SIGMOD Officers, Committees, and Awardees

Chair	Vice-Chair	Secretary/Treasurer
Donald Kossmann Systems Group ETH Zürich Cab F 73 8092 Zuerich SWITZERLAND +41 44 632 29 40 <donaldk AT inf.ethz.ch>	Anastasia Ailamaki School of Computer and Communication Sciences, EPFL EPFL/IC/IIF/DIAS Station 14, CH-1015 Lausanne SWITZERLAND +41 21 693 75 64 <natassa AT epfl.ch>	Magdalena Balazinska Computer Science & Engineering University of Washington Box 352350 Seattle, WA USA +1 206-616-1069 <magda AT cs.washington.edu>

SIGMOD Executive Committee:

Anastasia Ailamaki, Magdalena Balazinska, K. Selçuk Candan, Curtis Dyreson, Donald Kossmann, Yannis Ioannidis, Richard Hull, and Ioana Manolescu.

Advisory Board:

Raghu Ramakrishnan (Chair), Yahoo! Research, <First8CharsOfLastName AT yahoo-inc.com>,
Amr El Abbadi, Serge Abiteboul, Ricardo Baeza-Yates, Phil Bernstein, Elisa Bertino, Mike Carey, Surajit Chaudhuri, Christos Faloutsos, Alon Halevy, Joe Hellerstein, Renée Miller, C. Mohan, Beng-Chin Ooi, Z. Meral Ozsoyoglu, Sunita Sarawagi, Min Wang, and Gerhard Weikum.

SIGMOD Information Director:

Curtis Dyreson, Utah State University, <curtis.dyreson AT usu.edu>

Associate Information Directors:

Manfred Jeusfeld, Georgia Koutrika, Michael Ley, Wim Martens, Mirella Moro, Rachel Pottinger, Altigran Soares da Silva, and Jun Yang.

SIGMOD Record Editor-in-Chief:

Ioana Manolescu, Inria Saclay—Île-de-France, <ioana.manolescu AT inria.fr>

SIGMOD Record Associate Editors:

Denilson Barbosa, Pablo Barceló, Vanessa Braganholo, Marco Brambilla, Chee Yong Chan, Rada Chirkova, Anish Das Sarma, Glenn Paulley, Alkis Simitsis, Nesime Tatbul, and Marianne Winslett.

SIGMOD Conference Coordinator:

K. Selçuk Candan, Arizona State University <candan AT asu.edu>

PODS Executive Committee: Rick Hull (chair), <hull AT research.ibm.com>, Michael Benedikt, Wenfei Fan, Maurizio Lenzerini, Jan Paradaens, and Thomas Schwentick.

Sister Society Liaisons:

Raghu Ramakhrishnan (SIGKDD), Yannis Ioannidis (EDBT Endowment).

Awards Committee:

Masaru Kitsuregawa (chair, University of Tokyo, <kitsure AT tk1.iis.u-tokyo.ac.jp>), Rakesh Agrawal, Elisa Bertino, Umesh Dayal, and Maurizio Lenzerini.

Jim Gray Doctoral Dissertation Award Committee:

Johannes Gehrke (Co-chair), Cornell Univ.; Beng Chin Ooi (Co-chair), National Univ. of Singapore, Alfons Kemper, Hank Korth, Alberto Laender, Boon Thau Loo, Timos Sellis, and Kyu-Young Whang.

[Last updated : September 24th, 2013]

SIGMOD Edgar F. Codd Innovations Award

For innovative and highly significant contributions of enduring value to the development, understanding, or use of database systems and databases. Formerly known as the "SIGMOD Innovations Award", it now honors Dr. E. F. (Ted) Codd (1923 - 2003) who invented the relational data model and was responsible for the significant development of the database field as a scientific discipline. Recipients of the award are the following:

Michael Stonebraker (1992)
David DeWitt (1995)
Serge Abiteboul (1998)
Rudolf Bayer (2001)
Ronald Fagin (2004)
Jennifer Widom (2007)
Umeshwar Dayal (2010)
Stefano Ceri (2013)

Jim Gray (1993)
C. Mohan (1996)
Hector Garcia-Molina (1999)
Patricia Selinger (2002)
Michael Carey (2005)
Moshe Y. Vardi (2008)
Surajit Chaudhuri (2011)

Philip Bernstein (1994)
David Maier (1997)
Rakesh Agrawal (2000)
Don Chamberlin (2003)
Jeffrey D. Ullman (2006)
Masaru Kitsuregawa (2009)
Bruce Lindsay (2012)

SIGMOD Contributions Award

For significant contributions to the field of database systems through research funding, education, and professional services. Recipients of the award are the following:

Maria Zemankova (1992)
Jeffrey Ullman (1996)
Raghu Ramakrishnan (1999)
Daniel Rosenkrantz (2001)
Surajit Chaudhuri (2004)
Hans-Jörg Schek (2007)
David Lomet (2010)
H.V. Jagadish (2013)

Gio Wiederhold (1995)
Avi Silberschatz (1997)
Michael Carey (2000)
Richard Snodgrass (2002)
Hongjun Lu (2005)
Klaus R. Dittrich (2008)
Gerhard Weikum (2011)

Yahiko Kambayashi (1995)
Won Kim (1998)
Laura Haas (2000)
Michael Ley (2003)
Tamer Özsu (2006)
Beng Chin Ooi (2009)
Marianne Winslett (2012)

SIGMOD Jim Gray Doctoral Dissertation Award

SIGMOD has established the annual SIGMOD Jim Gray Doctoral Dissertation Award to *recognize excellent research by doctoral candidates in the database field.* Recipients of the award are the following:

- **2006** *Winner*: Gerome Miklau, University of Washington. *Runners-up*: Marcelo Arenas and Yanlei Diao.
- **2007** *Winner*: Boon Thau Loo, University of California at Berkeley. *Honorable Mentions*: Xifeng Yan and Martin Theobald.
- **2008** *Winner*: Ariel Fuxman, University of Toronto. *Honorable Mentions*: Cong Yu and Nilesh Dalvi.
- **2009** *Winner*: Daniel Abadi, MIT. *Honorable Mentions*: Bee-Chung Chen and Ashwin Machanavajjhala.
- **2010** *Winner:* Christopher Ré, University of Washington. *Honorable Mentions*: Soumyadeb Mitra and Fabian Suchanek.
- **2011** *Winner*: Stratos Idreos, Centrum Wiskunde & Informatica. *Honorable Mentions*: Todd Green and Karl Schnaitterz.
- **2012** *Winner*: Ryan Johnson, Carnegie Mellon University. *Honorable Mention*: Bogdan Alexe.
- **2013** *Winner*: Sudipto Das, University of California – Santa Barbara. *Honorable Mention*: Herodotos Herodotou and Wenchao Zhou.

A complete listing of all SIGMOD Awards is available at: **http://www.sigmod.org/awards/**

[Last updated : September 24th, 2013]

Editor's Notes

Welcome to the March 2014 issue of the ACM SIGMOD Record!

First, be sure to peruse the Changes to SIGMOD Bylaws, which are notified to ACM members (also) through the SIGMOD Record. The changes are due to the fact that the SIGMOD Anthology and the SIGMOD DiSC are no longer produced in their original form, and instead replaced by an online collection. SIGMOD members may object to these changes until July 16, 2014; the details are provided right after these notes.

In the Database Principles column, Andrew McGregor surveys stream algorithms for processing very large graphs. State-of-the art algorithms are surveyed for a variety of popular problems, identifying common algorithmic techniques and the associated smart graph summary data structures, such as spanners and sparsifiers. This very carefully constructed survey can serve as the basis of graduate courses on graph processing algorithms.

The article by Tu, Wang, Zeng and Xu consider the problem of building an energy-efficient, yet performant DBMS. Their approach is based on two strategies: first, energy-aware optimization of queries and storage organization, and second, controlling the power modes of the hardware. The authors developed a system called E^2DBMS by augmenting the kernel of PostgreSQL, and report on initial promising results obtained experimenting with their system.
The second article of the issue is by Anglez, Boncz, Larriba-Pey, Fundulaki, Neumann, Erling, Neubauer, Martinez, Kotsev and Toma, which collaborate within the Linked Data Benchmark Council (LDBC) research and development project sponsored by the European Union. The article presents the project objectives and goals, together with an original process and design methodology in designing new benchmarks, with a particular focus on benchmark features challenging the RDF data management engine from a performance perspective.

In the Systems and Prototypes column, Luiz, Ling and Correia describe MITRA, a middleware for replicating DBMSs and making them tolerant to Byzantine faults. Further, MITRA is designed to support transparent management of replicas across DBMSs from several vendors. This feature is interesting when one is to guarantee against the failure of all nodes due to the same bug in the same system. The paper outlines the replication protocol, the system architecture, and presents a brief performance evaluation.

In the Research Centers column, Balazinska, Howe and Suciu present the database research group at the University of Washington. The group topics include scientific data management, big data systems including for instance innovative systems built on top of Dydra and Hadoop, user-facing tools such as the SQLShare project which aims at making the usage of databases simpler by automating or hiding from the users, all aspects but the querying, thus debunking the myth that writing SQL queries is the main issue preventing a wider adoption of SQL. The authors also summarize their research works on probabilistic databases, causality in data transformations and query pricing.

The Open Forum column features two articles. The first, by Pawlish, Varde, Robila and Ranganathan focuses on the energy efficiency of data centers. The authors gathered energy usage data on a real data center belonging to a mid-size university, over three years. A Case-Based Reasoning (CBR) approach is then presented to support deciding which (part of) the operation(s) should be kept in-house and which to delegate to an external cloud, thus forming a hybrid data center.

The second article, by Anciaux, Bouganim, Delot, Illari, Kloul, Mitton and Pucheral is highly original. It considers the problem of providing data services in Least Developed Countries (LDCs), under many

strong restrictions: mostly no internet connection, illiterate users, strong challenges to personal rights and safety through the lack of appropriate law enforcement to which users could appeal in case their personal information is stolen or malevolently used. The authors outline an architecture called Folk-IS, enabling individuals from least developed countries to implement a large distributed system for performing many useful tasks across areas such as banking, culture and leisure services etc.

The issue features two event reports. Wandelt, Deng, Gerdijkov, Mishra, Mitankin, Patil, Siragusa, Tiskin, Wang, Wang and Lese outline an impressive string similarity benchmark effort which they organized under the form of the International Competition on Scalable String Similarity Search and Join (S4) held in conjunction with EDBT/ICDT 2013. The workshop organizers have set up task definitions and a specification of the running environment, then the workshop allowed nine teams to compete on two specific string matching problems, namely k-approximate search and k-approximate join. The paper describes the problems, the methodology, and the teams' results. Second, Darmont and Pedersen report on the Second International Workshop on Cloud Intelligence (Cloud-I 2013), held in conjunction with the VLDB 2013 Conference in Riva del Garda, Italy. Cloud-I featured a keynote on AsterixDB, papers on MapReduce and distributed data provenance, and a panel on challenges for research and industry.

The issue closes with a call for submissions to the First International Workshop on Bringing the Value of "Big Data" to Users (DATA4U), to be held in conjunction with the VLDB conference, in 2014.

Editor-in-Chief change: Yanlei Diao to take over

This issue is a special one for me, since it is the last one I edit! It has been a long journey since the first I had been in charge of, namely March 2010. I am extremely grateful for the honor of serving SIGMOD in this unique position, and I thank first and foremost Yannis Ioannidis, the SIGMOD chair, and SIGMOD Executive committee members of that time, for trusting me with the fate of the Record. I have been fortunate to work with a great team of editors, and to recruit other great ones during these four years. I want to thank and congratulate them again for the patient, careful work in inviting, preparing, overseeing and editing the contributions we are giving back to you, our readers. My thanks go next to the current SIGMOD Executive Committee and in particular the newly elected chairs and co-chairs, Donald Kossmann, Anastasia Ailamaki and Magda Balazinska, who trusted me again with the Record and, when I requested to be relieved of this task, organized the selection process that lead to the designation of the next Editor-in-Chief: Yanlei Diao, from the University of Massachussets.

My best wishes to Yanlei and long live the Record, the most lively database journal I know of (and one with around 70K downloads over the last year: http://dl.acm.org/citation.cfm?id=J689)!

Your submissions to the Record are welcome via the submission site:

http://sigmod.hosting.acm.org/record

Prior to submitting, be sure to peruse the Editorial Policy on the SIGMOD Record's Web site (http://www.sigmod.org/publications/sigmod-record/sigmod-record-editorial-policy).

Ioana Manolescu

March 2014

Past SIGMOD Record Editors:

Harrison R. Morse (1969)
Daniel O'Connell (1971 – 1973)
Randall Rustin (1974-1975)
Douglas S. Kerr (1976-1978)
Thomas J. Cook (1981 – 1983)
Jon D. Clark (1984 – 1985)
Margaret H. Dunham (1986 – 1988)
Arie Segev (1989 – 1995)
Jennifer Widom (1995 – 1996)
Michael Franklin (1996 – 2000)
Ling Liu (2000 – 2004)
Mario Nascimento (2005 – 2007)
Alexandros Labrinidis (2007 – 2009)

Changes to SIGMOD Bylaws/Notification to Members

Dear SIGMOD member -

Recently several proposed changes to the SIGMOD bylaws were approved by the ACM SIG Governing Board Executive Committee. Under recently approved procedural changes by the SGB, certain minor changes in bylaws can be approved by the SGB EC and sent on to the ACM Executive committee for approval without a formal vote of the membership.

However, SIGMOD members have 30 days to register any objections. If at least 3% of SIGMOD members object, then the bylaws changes must be voted on by the entire membership.

Any challenges may be sent to Donna Cappo, Director of SIG Services at: cappo@hq.acm.org and must be received by: July 16, 2014.

Summary of changes:

These updates are related to the changes in SIGMOD's publications, because the Anthology and DiSC (in its original form) are no longer produced and are replaced by an online collection (still called DiSC for branding) within the ACM DL. The very last change is simply for purposes of syntax.

==========
Article 2 (i-iii)
==========
Current language
==========
i. Collecting and disseminating information in the specialty, through a newsletter (SIGMOD Record), a Web portal (SIGMOD Online), SIGMOD Digital Library (Anthology and Digital Symposium Collection), and other publications approved by the Publications Board of the ACM;
ii. Organizing sessions at conferences of the ACM;
iii Sponsoring conferences, symposia and workshops;
==========
Proposed language
==========
i. Collecting and disseminating information in the specialty, through a newsletter (SIGMOD Record), a Web portal (SIGMOD Online), an online digital library (Digital Symposium Collection), and other publications approved by the Publications Board of the ACM;
ii. Organizing sessions at conferences of the ACM;
iii. Sponsoring and offering in-cooperation status to conferences, symposia and workshops;
=========

=========
Article 8
=========
Current language
==========
a. SIGMOD develops and maintains a digital library covering important database publications (journals, proceedings of conferences, symposia and workshops, books and other publications).
b. The SIGMOD Digital Library is available on SIGMOD Online, is distributed on electronic media such as CD or DVD and is made available to members.
c. The SIGMOD Digital Library consists of two collections: (i) SIGMOD Anthology, which is produced as necessary and includes new additions to the library, and (ii) SIGMOD Digital Symposium (DiSC), which is produced annually and includes annual updates to the Anthology.
d. With the advice of the officers, the Chair appoints editors of SIGMOD Anthology and SIGMOD DiSC, both of whom are members of the SIGMOD Executive Committee.
==========
Proposed language
==========
a. SIGMOD develops and maintains the SIGMOD Digital Symposium Collection (DiSC), a digital library covering important database publications (journals, proceedings of conferences, symposia and workshops, books and other publications).
b. The SIGMOD Digital Library is accessible through the ACM Digital Library and is made available to members.
=========

=========
Article 9
=========
Current language
==========
a. The SIGMOD Executive Committee comprises the officers, the Editor of the SIGMOD newsletter, the Chair of the Advisory Board, the Editor of SIGMOD Anthology, the Editor of SIGMOD Digital Symposium Collection (DiSC), the Information Director, SIGMOD Conference Coordinator, and Chair of PODS Executive Committee. No person may hold two positions on the Executive Committee.
b. The general duties of the Executive Committee is to advise the Chair
on all matters of interest to the SIGMOD. Specific duties or responsibilities may be specified in these Bylaws or assigned by the Chair. The Executive Committee must approve all the major management policy decisions of the Group.
c. All members of, or candidates for, the Executive Committee must be voting members of ACM and of SIGMOD.
==========
Proposed language
==========
a. The SIGMOD Executive Committee comprises the officers, the Editor of the SIGMOD newsletter, the Chair of the Advisory Board, the Information Director, the SIGMOD Conference Coordinator, the Chair of the PODS Executive Committee, and the ACM TODS Editor in Chief. No person may hold two positions on the Executive Committee.

b. The general duties of the Executive Committee are to advise the Chair on all matters of interest to SIGMOD. Specific duties or responsibilities may be specified in these Bylaws or assigned by the Chair. The Executive Committee must approve all the major management policy decisions of the Group.
c. All members of, or candidates for, the Executive Committee must be members of ACM and of SIGMOD.
==========

==========
Article 14 (a)
==========
Current language
==========
a. By September 30 of the even-numbered year preceding the election year, the Chair appoints a nominating committee which will propose at least two consenting candidates for each elective office of SIGMOD. The slate of candidates elected by the nominating committee must be presented to all the SIGMOD members by the following January 31.
==========
Proposed language
==========
a. By September 30 of the even-numbered year preceding the election year, the Chair appoints a nominating committee that proposes at least two consenting candidates for each elective office of SIGMOD. The slate of candidates elected by the nominating committee must be presented to all the SIGMOD members by the following January 31.

==========
Rationale
==========
This is simply a linguistic change.

Current SIGMOD bylaws can be viewed at:
http://www.acm.org/sigs/bylaws-contents/mod_bylaws/

Graph Stream Algorithms: A Survey*

Andrew McGregor†
University of Massachusetts
mcgregor@cs.umass.edu

ABSTRACT

Over the last decade, there has been considerable interest in designing algorithms for processing massive graphs in the data stream model. The original motivation was two-fold: a) in many applications, the dynamic graphs that arise are too large to be stored in the main memory of a single machine and b) considering graph problems yields new insights into the complexity of stream computation. However, the techniques developed in this area are now finding applications in other areas including data structures for dynamic graphs, approximation algorithms, and distributed and parallel computation. We survey the state-of-the-art results; identify general techniques; and highlight some simple algorithms that illustrate basic ideas.

1. INTRODUCTION

Massive graphs arise in any application where there is data about both basic entities and the relationships between these entities, e.g., web-pages and hyperlinks; neurons and synapses; papers and citations; IP addresses and network flows; people and their friendships. Graphs have also become the de facto standard for representing many types of highly-structured data. However, analyzing these graphs via classical algorithms can be challenging given the sheer size of the graphs. For example, both the web graph and models of the human brain would use around 10^{10} nodes and IPv6 supports 2^{128} possible addresses.

One approach to handling such graphs is to process them in the *data stream model* where the input is defined by a stream of data. For example, the stream could consist of the edges of the graph. Algorithms in this model must process the input stream in the order it arrives while using only a limited amount memory. These constraints capture various challenges that arise when processing massive data sets, e.g., monitoring network traffic in real time or ensuring I/O efficiency when processing data that does not fit in main memory. Related questions that arise include how to trade-off size and accuracy when constructing data summaries and how to quickly update these summaries. Techniques that have been developed to the reduce the space use have also been useful in reducing communication in distributed systems. The model also has deep connections with a variety of areas in theoretical computer science including communication complexity, metric embeddings, compressed sensing, and approximation algorithms.

The data stream model has become increasingly popular over the last twenty years although the focus of much of the early work was on processing numerical data such as estimating quantiles, heavy hitters, or the number of distinct elements in the stream. The earliest work to explicitly consider graph problems was the influential by paper by Henzinger et al. [36] which considered problems related to following paths in directed graphs and connectivity. Most of the work on graph streams has occurred in the last decade and focuses on the *semi-streaming* model [27, 52]. In this model the data stream algorithm is permitted $O(n \operatorname{polylog} n)$ space where n is the number of nodes in the graph. This is because most problems are provably intractable if the available space is sub-linear in n, whereas many problems become feasible once there is memory roughly proportional to the number of nodes in the graph.

In this document we will survey the results known for processing graph streams. In doing so there are numerous goals including identifying the state-of-the-art results for a variety of popular problems and identifying general algorithmic techniques. It will also be natural to discuss some important summary data structures for graphs, such as spanners and sparsifiers. Throughout, we will present various simple algorithms that illustrate basic ideas and would be suitable for teaching in an undergraduate or graduate classroom setting.

Notation. Throughout this document we will use n and

*__Database Principles Column.__ Column editor: Pablo Barceló. Department of Computer Science, University of Chile, Santiago, Chile. E-mail: pbarcelo@dcc.uchile.cl

†__Support.__ The author is supported in part by NSF awards CCF-0953754, IIS-1251110, and CCF-1320719 and a Google Research Award.

	Insert-Only	Insert-Delete	Sliding Window (width w)
Connectivity	Deterministic [27]	Randomized [5]	Deterministic [22]
Bipartiteness	Deterministic [27]	Randomized [5]	Deterministic [22]
Cut Sparsifier	Deterministic [2, 8]	Randomized [6, 31]	Randomized [22]
Spectral Sparsifier	Deterministic [8, 46]	Randomized $\tilde{O}(n^{5/3})$ space [7]	Randomized $\tilde{O}(n^{5/3})$ space [22]
$(2t-1)$-Spanners	$O(n^{1+1/t})$ space [11, 23]	Only multiple pass results known [6]	$O(\sqrt{wn^{(1+1/t)}})$ space [22]
Min. Spanning Tree	Exact [27]	$(1+\epsilon)$-approx. [5] Exact in $O(\log n)$ passes [5]	$(1+\epsilon)$-approx. [22]
Unweighted Matching	2-approx. [27] 1.58 lower bound [42]	Only multiple pass results known [3, 4]	$(3+\epsilon)$-approx. [22]
Weighted Matching	4.911-approx. [25]	Only multiple pass results known [3, 4]	9.027-approx. [22]

Table 1: Single-Pass, Semi-Streaming Results: Algorithms use $O(n \operatorname{polylog} n)$ space unless noted otherwise. Results for approximating the frequency of subgraphs discussed in Section 2.3.

m to denote the number of nodes and edges in the graph under consideration. For any natural number k, we use $[k]$ to denote the set $\{1, 2, \ldots, k\}$. We write $a = b \pm c$ to denote $b - c \leq a \leq b + c$. Many of the algorithms are randomized and we refer to events occurring *with high probability* if the probability of the event is at least $1 - 1/\operatorname{poly}(n)$. We use $\tilde{O}(\cdot)$ to indicate that logarithmic factors have been omitted.

2. INSERT-ONLY STREAMS

In this section, we consider streams consisting of a sequence of unordered pairs $e = \{u, v\}$ where $u, v \in [n]$. Such a stream,

$$S = \langle e_1, e_2, \ldots, e_m \rangle$$

naturally defines an undirected graph $G = (V, E)$ where $V = [n]$ and $E = \{e_1, \ldots, e_m\}$. See Figure 1. For simplicity, we will assume that all stream elements are distinct and therefore the resulting graph is not a multigraph[1]. We will also consider weighted graphs where now each element of the stream, $(e, w(e))$, defines both an edge of the graph and its weight.

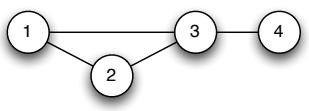


Figure 1: The graph on four nodes defined by the stream $S = \langle \{1,2\}, \{2,3\}, \{1,3\}, \{3,4\} \rangle$. It will be used to illustrate various definitions in later sections.

[1]Although many of the algorithms discussed immediately extend to the multigraph setting. Other problems such as estimating the number of triangles or distinct paths of length two require new ideas when edges have multiplicity [21, 38].

2.1 Connectivity, Trees, and Spanners

One of early motivations for considering the semi-streaming model is that $\tilde{\Theta}(n)$ space is necessary and sufficient to determine whether a graph is connected. The sufficiency follows from the following simple algorithm that constructs a spanning forest: we maintain a set of edges H and add the next edge in the stream $\{u, v\}$ to H if there is currently no path from u to v in H.

Spanners. A simple extension of the above algorithm also allows us to approximate the distance between any two nodes by constructing a *spanner*.

DEFINITION 1 (SPANNER). *Given a graph G, we say that a subgraph H is an α-spanner for G if for all $u, v \in V$,*

$$d_G(u, v) \leq d_H(u, v) \leq \alpha \cdot d_G(u, v) \, .$$

where $d_G(\cdot, \cdot)$ and $d_H(\cdot, \cdot)$ are lengths of the shortest paths in G and H respectively.

While the connectivity algorithm only added an edge if it did not complete a cycle, the algorithm for constructing a spanner will add an edge if it does not complete a short cycle.

Algorithm 1: Spanner

```
H ← ∅;
for each {u, v} ∈ S do
    If d_H(u, v) > α then H ← H ∪ {{u, v}};
return H
```

The fact that the resulting graph is an α-spanner follows because for each edge $(u, v) \in G \setminus H$, there must

have already been a path of length at most α in H. Hence, for any path in G of length d, including a shortest path, there is a corresponding path of length at most αd in H. The algorithm needs to store at most $O(n^{1+1/t})$ edges when $\alpha = 2t - 1$ for integral t. This follows because the shortest cycle in H has length $2t + 1$ and any such graph has at most $O(n^{1+1/t})$ edges [15]. A naive implementation of the above algorithm would be slow and more recent work has focused on developing faster algorithms [11, 23]. Other work [24] has considered constructing (α, β)-spanners where H is required to satisfy

$$d_G(u, v) \leq d_H(u, v) \leq \alpha \cdot d_G(u, v) + \beta \ .$$

Minimum Spanning Tree. Another generalization of the basic connectivity algorithm is to maintain a minimum spanning tree (or spanning forest if the graph is not connected).

Algorithm 2: Minimum Spanning Tree

```
1 H ← ∅;
2 for each {u, v} ∈ S do
3 |   H ← H ∪ {(u, v)};
4 |   If H includes a cycle, remove the largest weight
  |   edge in the cycle from H.
5 return H
```

By using the appropriate data structures, the above algorithm can be implemented such that each update takes $O(\log n)$ time [60].

2.2 Graph Sparsification

We next consider constructing graph sparsifiers in the data stream model. Rather than just determining whether a graph is connected, these sparsifiers will allow us to estimate a richer set of connectivity properties such as the size of *all* cuts in the graph. We will be interested in different types of sparsifier. First, Benczür and Karger [14] introduced the notion of cut sparsification.

DEFINITION 2 (CUT SPARSIFICATION). *We say that a weighted subgraph H is a $(1+\epsilon)$ cut sparsification of a graph G if*

$$\lambda_A(H) = (1 \pm \epsilon)\lambda_A(G) \ , \quad \forall A \subset V \ , \qquad (1)$$

where $\lambda_A(G)$ and $\lambda_A(H)$ is the weight of the cut $(A, V \setminus A)$ in G and H respectively.

Spielman and Teng [59] introduced the more general notion of spectral sparsification based on approximating the Laplacian of a graph.

DEFINITION 3 (LAPLACIAN). *The Laplacian of an undirected weighted graph $H = (V, E, w)$, is a matrix $L_H \in \mathbb{R}^{n \times n}$ where*

$$L_H(i, j) = \begin{cases} \sum_{\{i,k\} \in E} w(i, k) & \text{if } i = j \\ -w(i, j) & \text{otherwise} \end{cases}$$

and $w(i, j)$ is the weight of the edge between nodes i and j. If there is no such edge, let $w(i, j) = 0$.

DEFINITION 4 (SPECTRAL SPARSIFICATION). *We say that a weighted subgraph H is a $(1+\epsilon)$ spectral sparsification of a graph G if,*

$$x^T L_H x = (1 \pm \epsilon) x^T L_G x \ , \ \ \forall x \in \mathbb{R}^n \ , \qquad (2)$$

where L_G and L_H are the Laplacians of H and G.

Note that if we replace $\forall x \in \mathbb{R}^n$ in Equation 2 by $\forall x \in \{0, 1\}^n$ then we recover Equation 1. Hence, given a spectral sparsification of G, we can approximate the weight of all cuts in G. We can also approximate other "spectral properties" of G including the eigenvalues (via the Courant-Fischer Theorem), the effective resistances in the analogous electrical network, and various properties of random walks. Obviously, any graph G has a spectral sparsifier since G is a spectral sparsifier of itself. What is surprising is that there exists a $(1 + \epsilon)$ spectral sparsifier with at most $O(\epsilon^{-2} n)$ edges [12].

A Simple "Merge and Reduce" Approach. Not only do small spectral sparsifiers exist but they can also be constructed in the semi-streaming model [2, 46]. In this section, we present a simple algorithm that demonstrates the useful "merge and reduce" framework that has been useful for other data stream problems [8].

The following algorithm uses, as a black box, any existing algorithm that returns a $(1 + \gamma)$ spectral sparsifier. Let $\mathcal{A}$ be such an algorithm and let $\text{size}(\gamma)$ be an upper bound on the number of edges in the resulting sparsifier. As mentioned above, we may assume that $\text{size}(\gamma) = O(\gamma^{-2} n)$. We will also use the following easily verifiable properties of a spectral sparsifier:

- *Mergeable:* Suppose H_1 and H_2 are α spectral sparsifiers of two graphs G_1 and G_2 on the same set of nodes. Then $H_1 \cup H_2$ is an α spectral sparsifier of $G_1 \cup G_2$.
- *Composable:* If H_3 is an α spectral sparsifier for H_2 and H_2 is a β spectral sparsifier for H_1 then H_3 is an $\alpha\beta$ spectral sparsifier for H_1.

The algorithm is based on a hierarchical partitioning of the stream. First we partition the input stream of edges into $t = m/\text{size}(\gamma)$ segments of length $\text{size}(\gamma)$. For simplicity assume that t is a power of two. Let G_i^0 be the graph corresponding to the i-th segment of edges. For $i \in \{1, \ldots, \log_2 t\}$ and $j \in \{1, \ldots, t/2^i\}$, define

$$G_i^j = G_{2i-1}^{j-1} \cup G_{2i}^{j-1} \ .$$

For example, if $t = 4$, we have:

$$G_1^1 = G_1^0 \cup G_2^0 \ , \ G_2^1 = G_3^0 \cup G_4^0 \ ,$$

$$G_1^2 = G_1^1 \cup G_2^1 = G_1^0 \cup G_2^0 \cup G_3^0 \cup G_4^0 = G \ .$$

For each G_i^j, we define a weighted subgraph H_i^j using the sparsification algorithm $\mathcal{A}$ as follows:

$$H_i^0 = G_i^0 \text{ and } H_i^j = \mathcal{A}(H_{2i-1}^{j-1} \cup H_{2i}^{j-1}) \text{ for } j > 0 \ .$$

It follows from the mergeable and composeable properties that $H_1^{\log_2 t}$ is an $(1+\gamma)^{\log_2 t}$ sparsifier of G. If we set $\gamma = \epsilon/(2 \log_2 t)$ then this is a $(1+\epsilon)$ sparsifier. Furthermore, it is possible to compute $H_1^{\log_2 t}$ while only storing at most

$$2 \, \text{size}(\gamma) \log_2 t = O(\epsilon^{-2} n \log^3 n)$$

edges at any given time. This is because, as soon as we have constructed H_i^j, we can forget H_{2i-1}^{j-1} and H_{2i}^{j-1}. Hence, at any given time we will only need to store H_i^j for at most two values of i for each j.

2.3 Counting Subgraphs

Another problem that has received significant attention is counting the number of triangles, T_3, in a graph. This is closely related to the *transitivity coefficient*, the fraction of paths of length two that form a triangle, and the *clustering coefficient*, i.e.,

$$\frac{1}{n} \sum_v \frac{T_3(v)}{\binom{\deg(v)}{2}}$$

where $T_3(v)$ is the number of triangles that include the node v. Both statistics play an important role in the analysis of social networks. Unfortunately, it can be shown that determining whether a graph is triangle-free requires $\Omega(n^2)$ space even with a constant number of passes and more generally, $\Omega(m/T_3)$ space is required for any constant approximation [17]. Hence, research has focused on designing algorithms whose space will depend on a given lower bound $t \leq T_3$.

Vector-Based Approach. A number of approaches for estimating the number of triangles have been based on reducing the problem to a problem about vectors. Consider a vector $\mathbf{x}$ indexed by subsets T of $[n]$ of size three. Each T represents a triple of nodes and the entry corresponding to T is defined to be,

$$\mathbf{x}_T = |\{e \in S : e \subset T\}| \ .$$

For example, in the stream S corresponding to the graph in Figure 1, the entries of the vector are:

$$\mathbf{x}_{\{1,2,3\}} = 3, \ \mathbf{x}_{\{1,2,4\}} = 1, \ \mathbf{x}_{\{1,3,4\}} = 2, \ \mathbf{x}_{\{2,3,4\}} = 2 \ .$$

Note that the number of triangles T_3 in G equals the number entries of $\mathbf{x}$ that equal 3. Bar-Yossef et al. [10] presented an algorithm based on the following relationship between T_3 and the frequency moments of $\mathbf{x}$, i.e., $F_k = \sum_T \mathbf{x}_T^k$.

LEMMA 2.1. $T_3 = F_0 - 1.5F_1 + 0.5F_2$.

Let $\tilde{T}_3$ be the estimate of T_3 that results by combining $(1+\gamma)$-approximations of the relevant frequency moments with the above lemma. Then,

$$|\tilde{T}_3 - T_3| < \gamma \left(F_0 + 1.5F_1 + 0.5F_2\right) \leq 8\gamma mn$$

where the last inequality follows since

$$\max(F_0, F_2/9) \leq F_1 = m(n-2) \ .$$

It is possible to $(1+\gamma)$-approximate each of these frequency moments in $\tilde{O}(\gamma^{-2})$ space and so, by setting $\gamma = \epsilon/(8mn)$, this implies a $(1+\epsilon)$-approximation algorithm using $\tilde{O}(\epsilon^{-2}(mn/t)^2)$ space.

A more space-efficient approach proposed by Ahn et al. [6], is to use the ℓ_0 sampling technique [40]. An algorithm for ℓ_0 sampling uses $O(\text{polylog}\, n)$ space and returns a random non-zero element from $\mathbf{x}$. Let $X \in \{1,2,3\}$ be determined by picking a random non-zero element of v and returning the associated value of this element. Let $Y = 1$ if $X = 3$ and $Y = 0$ otherwise. Note that $E[Y] = T_3/F_0$. By an application of the Chernoff bound, the mean of $\tilde{O}(\epsilon^{-2}(mn/t))$ independent copies of Y equals $(1 \pm \epsilon)T_3/F_0$ with high probability. Multiplying this by an approximation of F_0 yields a good estimate of T_3. Note that an earlier algorithm using similar space was presented by Buriol et al. [18] but the above algorithm has the advantage that it is also applicable in the setting (discussed in a later section) where edges can be inserted and deleted.

Extensions and Other Approaches. Pavan et al. [54] developed the approach of Buriol et al. such that the dependence on n in the space used became a dependence on the maximum degree in the graph, and a tighter analysis is possible. Pagh and Tsourakakis [53] presented an algorithm based on randomly coloring the nodes and counting the number of monochromatic triangles exactly. Algorithms have also been developed for the multi-pass model including a two-pass algorithm using $\tilde{O}(m/t^{1/3})$ space [17] and an $O(\log n)$-pass semi-streaming algorithm [13]. Kutzkov and Pagh [49] and Jha et al. [37] also designed algorithms for estimating the clustering and transitivity coefficients directly.

Extending an approach used by Jowhari and Ghodsi [39], another line of work [39, 41, 50] makes clever use of complex-valued hash functions for counting longer cycles and other subgraphs. Lower bounds for finding short cycles were proved by Feigenbaum et al. [28]. Other related work includes approximating the size of cliques [35], independent sets [34], and dense components [9].

2.4 Matchings

A matching in a graph $G = (V, E)$ is a subset of edges $M \subseteq E$ such that no two edges in M share an endpoint. Well-studied problems including computing the matching of maximum cardinality or maximum total weight.

Greedy Single-Pass Algorithms. A simple algorithm that returns a 2-approximation for the unweighted problem is the following greedy algorithm.

Algorithm 3: Greedy Matching

```
M ← ∅;
for each e ∈ S do
    If M ∪ {e} is a matching, M ← M ∪ {e};
return M
```

The fact that the algorithm returns a 2-approximation follows from the fact that for every edge $\{u, v\}$ in a maximum cardinality matching, M must include an edge with at least one of u or v as an endpoint. At present this is the best approximation known for the problem! The strongest known lower bound is $e/(e-1) \approx 1.58$ which also applies when edges are grouped by endpoint [30,42]. Konrad et al. [47] considered a relaxation of the problem where the edges arrive in a random-order and, in this setting, they designed an algorithm that achieved a 1.98-approximation in expectation.

The greedy algorithm can easily be generalized to the weighted case as follows [27, 51]. Rather than only adding an edge if there are no "conflicting" edges, we also add the edge if its weight is at least some factor larger than the weight of the (at most two) conflicting edges and then remove these conflicting edges.

Algorithm 4: Greedy Weighted Matching

```
M ← ∅;
for each e ∈ S do
    Let C = {e' ∈ M : e' ∩ e ≠ ∅} ;
    If w(e)/w(C) ≥ (1 + γ) then M ← M ∪ {e} \ C;
return M
```

It would be reasonable to ask why we shouldn't add e if $w(e) \geq w(C)$, i.e., set $\gamma = 0$. However, consider what would happen if the stream consisted of edges

$$\{1, 2\}, \{2, 3\}, \{3, 4\}, \ldots, \{n-1, n\}$$

arriving in that order where the weight of edge $\{i, i+1\}$ is $1 + i\epsilon$ for some small value $\epsilon > 0$. The above algorithm would return the last edge with a total weight of $1 + (n-1)\epsilon$ whereas the optimal solution has weight

$$1+(n-1)\epsilon+1+(n-3)\epsilon+1+(n-5)\epsilon+\ldots > \frac{n-1}{2},$$

and hence decreasing ϵ makes the approximation factor arbitrarily large.

Roughly speaking, the problem with setting $\gamma = 0$ is that the weight of the "trail" of edges that are inserted into M but subsequently removed can be much larger than the weight of the final edges in M. By setting $\gamma > 0$, we ensure the weights in this trail are geometrically increasing. Specifically, let $T_e = C_1 \cup C_2 \cup \ldots$ where C_1 is the set of edges removed when e was added to M and C_{i+1} is the set of edges removed when an edge in C_i was added to M. Then, it is easy to show that for any edge e, the total weight of edges in T_e satisfies,

$$w(T_e) \leq w(e)/\gamma .$$

By a careful charging scheme [51], the weight of the optimal solution can be bounded in terms of the weight of the final edges and the trails:

$$w(\mathsf{OPT}) \leq (1+\gamma) \sum_{e \in M} (w(T_e) + 2w(e)) .$$

Substituting in the bound on $w(T_e)$ and optimizing over γ yields a 5.828-approximation. The analysis can be extended to sub-modular maximization problems [20].

The above algorithm is optimal among deterministic algorithms that only remember the edges of a valid matching at any time [61]. However, after a sequence of results [25, 26, 62] it is now known how to achieve a 4.91-approximation.

Multiple-Pass Algorithms. The above algorithm can be extended to a multiple-pass algorithm that achieves a $(2+\epsilon)$-approximation for weighted matchings. We simply set $\gamma = O(\epsilon)$ and take $O(\epsilon^{-3})$ passes over the data where, at the start of a pass, M is initiated to the matching returned at the end of the previous pass.

Guruswami and Onak showed that finding the size of the maximum cardinality matching exactly given p passes requires $n^{1+\Omega(1/p)}/p^{O(1)}$ space. No exact algorithm is known with these parameters but it is possible to find an arbitrarily good approximation. Ahn and Guha [3,4] showed that a $(1+\epsilon)$-approximation is possible using $O(n^{1+1/p})$ space and $O(p/\epsilon)$ passes. They also show a similar result for weighted matching if the graph is bipartite. Their results are based on adapting linear programming techniques and a careful analysis of the intrinsic adaptivity of primal-dual algorithms. In the node arrival setting where edges are grouped by endpoint, Kapralov presented an algorithm that achieved a $1/(1-1/\sqrt{2\pi p}+o(1/p))$-approximation ratio given p passes. This is achieved by a fractional load balancing approach.

2.5 Random Walks

A random walk in an unweighted graph from a node $u \in V$ is a random sequence of nodes $v_0, v_1, v_2, \ldots$ where $v_0 = u$ and v_i is a random node from the set $\Gamma(v_{i-1})$, the neighbors of v_{i-1}. For any fixed positive integer t, we can consider the distribution of $v_t \in V$. Call this distribution $\mu_t(u)$.

In this section, we present a semi-streaming algorithm by Das Sarma et al. [56] that returns a sample from $\mu_t(u)$. Note that it is trivial to sample from $\mu_t(u)$ with t passes; in the i-th pass we randomly select v_i from the neighbors of the node v_{i-1} determined in the previous pass. Das Sarma et al. show that it is possible to reduce the number of passes to $O(\sqrt{t})$. They also present algorithms that use less space at the expense of increasing the number of passes.

Algorithm. As noted above, it is trivial to perform length t walks in t passes. The main idea of the algorithm to build up a length t walk by "stitching" together short walks of length $\sqrt{t}$. Each of these short walks can be constructed in parallel in $\sqrt{t}$ passes and $O(n \log n)$ space. However, we will need to be careful to ensure that all the steps of the final walk are independent. Specifically, the algorithm starts as follows:

1. Let $T(v)$ be a node sampled from $\mu_{\sqrt{t}}(v)$.
2. Let $v = T^k(u) = T(\ldots T(T(u))\ldots)$ where k is maximal values such that the nodes in
$$U = \{u, T(u), T^2(u), \ldots, T^{k-1}(u)\}$$
are all distinct and $k \leq \sqrt{t}$.

The reason that we insist that the nodes in U are disjoint is because otherwise the next steps of the random walk will not be independent of the previous steps. So far we have generated a sample v from $\mu_\ell(u)$ where $\ell = k\sqrt{t}$. We then enter the following loop:

3. While $\ell \leq \sqrt{t}$
   (a) If $v \notin U$, let $v \leftarrow T(v), \ell \leftarrow \sqrt{t} + \ell, U \leftarrow U \cup \{v\}$
   (b) Otherwise, sample $\sqrt{t}$ edges with replacement incident on each node in U. Find the maximal path from v such that on the i-th visit to node x, we take the i-th edge that was sampled for node x. The path terminates either when a node in U is visited more than $\sqrt{t}$ times or we reach a node that is not in U. Reset v to be the final node of this path and increase ℓ by the length of the path. If we complete the length t random walk during this process we may terminate at this point and return the current node.
4. Perform the remaining $O(\sqrt{t})$ steps of the walk using the trivial algorithm.

Analysis. First note that $|U|$ is never larger than $\sqrt{t}$ because $|U|$ is only incremented when ℓ increases by at least $\sqrt{t}$ and we know that $\ell \leq t$. The total space required to store the vertices T is $\tilde{O}(n)$. When we sample $\sqrt{t}$ edges incident on each node in U, this requires $\tilde{O}(|U|\sqrt{t}) = \tilde{O}(t)$ space. Hence, the total space is $\tilde{O}(n + t)$. For the number of passes, note that when we need to take a pass to sample edges incident on U, we make $O(\sqrt{t})$ hops of progress because either we reach a node with an unused short walk or the walk uses $\Omega(\sqrt{t})$ samples edges. Hence, including the $O(\sqrt{t})$ passes used at the start and end of the algorithm, the total number of passes is $O(\sqrt{t})$.

3. GRAPH SKETCHES

In this section, we consider dynamic graph streams where edges can be both added and removed. The input is a sequence

$$S = \langle a_1, a_2, \ldots \rangle \quad \text{where} \quad a_i = (e_i, \Delta_i)$$

where e_i encodes an undirected edge as before and $\Delta_i \in \{-1, 1\}$. The multiplicity of an edge e is defined as $f_e = \sum_{i:e_i=e} \Delta_i$. For simplicity, we restrict our attention to the case where $f_e \in \{0, 1\}$ for all edges e.

Linear Sketches. An important type of data stream algorithms are *linear sketches*. Such algorithms maintain a random linear projection, or "sketch", of the input. We want to be able to a) infer relevant properties of the input from the sketch and b) maintain the sketch in small space. The second property follows from the linearity of the sketch if the dimensionality of the projection is small. Specifically, suppose

$$\mathbf{f} \in \{0, 1\}^{\binom{n}{2}}$$

is the vector with entries equalling the current values of f_e and let $\mathcal{A}(\mathbf{f}) \in \mathbb{R}^d$ be the sketch of this vector where we call d the dimensionality of the sketch. Then, when (e, Δ) arrives we can simple update $\mathcal{A}(\mathbf{f})$ as follows:

$$\mathcal{A}(\mathbf{f}) \leftarrow \mathcal{A}(\mathbf{f}) + \Delta \cdot \mathcal{A}(\mathbf{i}^e)$$

where $\mathbf{i}^e$ is the vector whose only non-zero entry is a "1" in the position corresponding to e. Hence, it suffices to store the current sketch and any random bits needed to compute the projection. The main challenge is therefore to design low dimensional sketches.

Homomorphic Sketches. Many of the graph sketches that have been designed so far are built up from sketches of the rows of the adjacency matrix for the graph G. Specifically, let $\mathbf{f}^v \in \{0, 1\}^{n-1}$ be the vector $\mathbf{f}$ restricted

to coordinates that involve node v. Then, the sketches are formed by concatenating sketches of each $\mathbf{f}^v$, i.e.,

$$\mathcal{A}(\mathbf{f}) = \mathcal{A}_1(\mathbf{f}^{v_1}) \circ \mathcal{A}_2(\mathbf{f}^{v_2}) \circ \ldots \circ \mathcal{A}_n(\mathbf{f}^{v_n}) .$$

Note that the random projections for different $\mathcal{A}_i$ need not be independent but that these sketches can still be updated as before.

The algorithms discussed in subsequent sections all fit the following template. First, we consider a basic algorithm for the graph problem in question. Second, we design sketches $\mathcal{A}_i$ such that it is possible to emulate the basic algorithm given only the projections $\mathcal{A}_i(\mathbf{f}^{v_i})$. The challenge is to ensure that the sketches are homomorphic with respect to the operations of the basic algorithm, i.e., for each operation on the original graph, there is a corresponding operation on the sketches.

3.1 Connectivity

We start with a simple algorithm for finding a spanning forest of a graph and then show how to emulate this algorithm via sketches.

Basic Non-Sketch Algorithm. The algorithm is based on the following simple $O(\log n)$ stage process. In the first stage, we find an arbitrary incident edge for each node. We then collapse each of the resulting connected components into a "supernode". In each subsequent stage, we find an edge from every supernode to another supernode (if one exists) and collapse the connected components into new supernodes. It is not hard to argue that this process terminates after $O(\log n)$ stages and that the set of edges used to connect supernodes in the different stages include a spanning forest of the graph. From this we can obviously deduce whether the graph is connected.

Emulation via Sketches. There are two main steps to constructing the sketches for the connectivity algorithm:

1. *An Appropriate Graph Representation.* For each node $v_i \in V$, define a vector $\mathbf{a}_i \in \{-1, 0, 1\}^{\binom{n}{2}}$:

$$\mathbf{a}^i_{\{j,k\}} = \begin{cases} 1 & \text{if } i = j < k \text{ and } \{v_j, v_k\} \in E \\ -1 & \text{if } j < k = i \text{ and } \{v_j, v_k\} \in E \\ 0 & \text{otherwise} \end{cases}$$

   These vectors then have the useful property that for any subset of nodes $\{v_i\}_{i \in S}$, the non-zero entries of $\sum_{i \in S} \mathbf{a}^i$ correspond exactly to the edges across the cut $(S, V \setminus S)$.

2. *ℓ_0-Sampling via Linear Sketches:* As mentioned earlier, the goal of ℓ_0-sampling is to take a non-zero vector $\mathbf{x} \in \mathbb{R}^d$ and return a sample j where

$$\Pr_r[\text{sample equals } j] = \begin{cases} \frac{1}{|F_0(\mathbf{x})|} & \text{if } \mathbf{x}_j \neq 0 \\ 0 & \text{if } \mathbf{x}_j = 0 \end{cases} .$$

   A useful feature of existing work [40] on ℓ_0 sampling is that it can be performed via linear projections, i.e., for any string r there exists $M_r \in \mathbb{R}^{k \times d}$ such that the sample can be reconstructed from $M_r\mathbf{x}$. For the process to be successful with constant probability $k = O(\log^2 n)$ suffices. Consequently, given $M_r\mathbf{x}$ and $M_r\mathbf{y}$ we have enough information to determine a random sample from the set $\{i : x_i + y_i \neq 0\}$ since

$$M_r(\mathbf{x} + \mathbf{y}) = M_r\mathbf{x} + M_r\mathbf{y} .$$

For example, for the graph in Figure 1, we have

$$\begin{array}{rcl} \mathbf{a}^1 & = & (\ \ 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0\ \ ) \\ \mathbf{a}^2 & = & (\ -1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0\ \ ) \\ \mathbf{a}^3 & = & (\ \ 0 \quad -1 \quad 0 \quad -1 \quad 0 \quad 1\ \ ) \\ \mathbf{a}^4 & = & (\ \ 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad -1\ \ ) \end{array}$$

where the entries correspond to the sets $\{1,2\}$, $\{1,3\}$, $\{1,4\}$, $\{2,3\}$, $\{2,4\}$, $\{3,4\}$ in that order. Note that the non-zero entries of

$$\mathbf{a}^1 + \mathbf{a}^2 \ = \ (\ \ 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0\ \ )$$

correspond to $\{1,3\}$ and $\{2,3\}$ which are exactly the edges across the cut $(\{1,2\},\{3,4\})$.

The resulting algorithm for connectivity is relatively simple but makes use of linearity in an essential way:

1. *In a single pass, compute the sketches:* Choose $t = O(\log n)$ random strings $r_1, \ldots, r_t$ and construct the ℓ_0-sampling projections $M_{r_j}\mathbf{a}^i$ for $i \in [n]$, $j \in [t]$. Then,

$$\mathcal{A}_i(\mathbf{f}^{v_i}) = \left(M_{r_1}\mathbf{a}^i\right) \circ \left(M_{r_2}\mathbf{a}^i\right) \ldots \circ \left(M_{r_t}\mathbf{a}^i\right) .$$

2. *In post-processing, emulate the original algorithm:*

   (a) Let $\hat{V} = V$ be the initial set of "supernodes".

   (b) For $i = 1, \ldots, t$: for each supernode $S \in \hat{V}$, use $\sum_{i \in S} M_{r_j}\mathbf{a}^i = M_{r_j}(\sum_{i \in S} \mathbf{a}^i)$ to sample an edge between S and another supernode. Collapse the connected supernodes to form a new set of supernodes.

Since each sketch $\mathcal{A}_i$ has dimension $O(\text{polylog}\, n)$ and there are n such sketches to be computed, the final connectivity algorithm uses $O(n \cdot \text{polylog}\, n)$ space.

Extensions and Further Work. Note that the above algorithm has $O(\text{polylog}\, n)$ update time but a connectivity query may take $\Omega(n)$ time. This was addressed in subsequent work by Kapron et al. [44].

An easy corollary of the above the result is that it is also possible to test whether a graph is bipartite. This follows by running the connectivity algorithm on the both G and the *bipartite double cover* of G. The bipartite double cover of a graph is formed by making two

copies u_1, u_2 of every node u of G and adding edges $\{u_1, v_2\}, \{u_2, v_1\}$ for every edge $\{u, v\}$ of G. It can be shown that G is bipartite iff the number of connected components in the double cover is exactly twice the number of connected components in G.

3.2 k-Connectivity

We next present an extension to test k-connectivity, i.e, determining whether every cut in the graph includes at least k edges. This algorithm builds upon ideas in the previous section and exploits the linearity of the sketches to an even greater extent.

Basic Non-Sketch Algorithm. The starting point for the algorithm is the following basic k phase algorithm:

1. For $i = 1$ to k: Let F_i be a spanning forest of $(V, E \setminus \bigcup_{j=1}^{i-1} F_j)$
2. Then $(V, F_1 \cup F_2 \cup \ldots \cup F_k)$ is k-edge-connected iff $G = (V, E)$ is at least k-edge-connected.

The correctness is easy to show by arguing that for any cut, every F_i contains an edge across this cut or

$$F_1 \cup \ldots \cup F_{i-1}$$

already contains *all* the edges across the cut. Hence, if $F_1 \cup F_2 \cup \ldots \cup F_k$ does not contain all the edges across the cut, it includes at least k of them. We call a set of edges with this property a *k-skeleton*.

Emulation via Sketches. As with the connectivity algorithm, we compute the entire set of sketches and then emulate the algorithm on the compressed form. The important observation is that if we have computed a sketch $\mathcal{A}(G)$, but subsequently need the sketch $\mathcal{A}(G - F)$ for some set of edges F we have discovered, then this can be computed as $\mathcal{A}(G - F) = \mathcal{A}(G) - \mathcal{A}(F)$.

1. *In a single pass, compute the sketches:* Let $\mathcal{A}^1(G)$, $\mathcal{A}^2(G), \ldots, \mathcal{A}^k(G)$ be k independent sketches for finding a spanning forest.
2. *In post-processing, emulate the original algorithm:* For $i \in [k]$, construct a spanning forest F_i of $(V, E \setminus F_1 \cup \ldots \cup F_{i-1})$ using

$$\mathcal{A}^i(G - F_1 - F_2 \ldots - F_{i-1}) = \mathcal{A}^i(G) - \sum_{j=1}^{i-1} \mathcal{A}^i(F_j)\,.$$

Since computing each spanning forest sketch used $O(n \cdot \text{polylog}\, n)$, the total space used by the algorithm for k-connectivity is $O(k \cdot n \cdot \text{polylog}\, n)$.

3.3 Min-Cut and Sparsification

In this section we revisit graph sparsification in the context of dynamic graphs. To do this we will need to discuss the offline algorithms for sparsification in more detail.

Sparsification via Sampling. The results in this section are based on the following generic sampling algorithm:

Algorithm 5: Generic Sparsification Algorithm

1 Sample each edge e with probability p_e;
2 Weight each sampled edge e by $1/p_e$;

It is obvious that the size of any cut is preserved in expectation by the above process. However, if p_e is sufficiently large it can be shown that a range of properties, including the size of cuts are approximately preserved with high probability. In particular, Karger [45] showed that for some constant c_1, if

$$p_e \geq q := \min\left\{1, c_1 \lambda^{-1} \epsilon^{-2} \log n\right\}$$

where λ is the size of the minimum cut of the graph then the resulting graph is a cut sparsifier with high probability. Fung et al. [29] strengthened this to show that the sampling probability need only scale with λ_e^{-1} where λ_e is the size of the minimum cut that separates the end points of e. Specifically, they showed that for some constant c_2, if

$$p_e \geq \min\left\{1, c_2 \lambda_e^{-1} \epsilon^{-2} \log^2 n\right\}$$

then the resulting graph is a cut sparsifier with high probability. Spielman and Srivistava [58] showed[2] that the resulting graph is a spectral sparsifier if

$$p_e \geq \min\left\{1, c_3 r_e \epsilon^{-2} \log n\right\}$$

for some constant c_3 where r_e is the *effective resistance* of e. The effective resistance of an edge $\{u, v\}$ is the voltage difference that would need to be applied between u and v for 1 amp to flow between u and v in the electrical network formed by replacing each edge by a 1 ohm resister. The effective resistance r_e is a more nuanced quantity than λ_e in the sense that λ_e only depends on the number of edge-disjoint paths between the endpoints of e whereas the lengths of these paths are also relevant when calculating the effective resistance r_e. However, the two quantities are related by the following inequality [7],

$$\lambda_e^{-1} \leq r_e = O(n^{2/3}) \lambda_e^{-1}\,. \tag{3}$$

Minimum Cut. As a warm-up, we show how to estimate the minimum cut λ of a dynamic graph [6]. To do this we use the algorithm for constructing k-skeletons described in the previous section in conjunction with Karger's sampling result. In addition to computing a

[2]Note that their result is actually proved for a slightly different sampling with replacement procedure.

skeleton on the entire graph, we also construct skeletons for subsampled graphs. Specifically, let G_i be the graph formed from G by including each edge with probability $1/2^i$ and let

$$H_i = \text{skeleton}_k(G_i) \ ,$$

be a k-skeleton of G_i where $k = 3c_1\epsilon^{-2}\log n$. Then, for

$$j = \min\{i : \text{mincut}(H_i) < k\} \ ,$$

we claim that

$$2^j \, \text{mincut}(H_j) = (1 \pm \epsilon)\lambda \ . \tag{4}$$

For $i \leq \lfloor \log_2 1/q \rfloor$, Karger's result implies that all cuts are approximately preserved and, in particular,

$$2^i \cdot \text{mincut}(H_i) = (1 \pm \epsilon) \, \text{mincut}(G_i) \ .$$

However, for $i = \lfloor \log_2 1/q \rfloor$,

$$E\left[\text{mincut}(H_i)\right] \leq 2^{-i}\lambda \leq 2q\lambda \leq 2c_1\epsilon^{-2}\log n$$

and hence, by an application of the Chernoff bound, we have that $\text{mincut}(H_i) < k$ with high probability. Hence, $j \leq \lfloor \log_2 1/q \rfloor$ with high probability and Equation 4 follows.

Sparsification. To construct a sparsifier, the basic idea is to sample edges with probability $q_e = \min\{1, t/\lambda_e\}$ for some value of t. If $t = \Theta(\epsilon^{-2}\log^2 n)$ then the resulting graph is a combinatorial sparsifier by appealing to the aforementioned result of Fung et al. [29]. If $t = \Theta(\epsilon^{-2}n^{2/3}\log n)$ then the resulting graph can be shown to be a spectral sparsifier by combining Equation 3 with the aforementioned sampling result of Spielman and Srivistava [58]. In this section, we briefly outline how to perform such sampling. We refer the reader to Ahn et al. [6, 7] for details regarding independence issues and how to reweight the edges.

The challenge is that we do not know the values of λ_e ahead of time. To get around this we take a very similar approach to that used above for estimating the minimum cut. Specifically, let G_i be defined as above and let $H_i = \text{skeleton}_{3t}(G_i)$. For simplicity, we assume $\lambda_e \geq t$. We claim that

$$\Pr\left[e \in H_0 \cup H_1 \cup \ldots \cup H_{2\log n}\right] \geq t/\lambda_e \ .$$

This follows because the above probability is at least $\Pr\left[e \in H_j\right]$ for $j = \lfloor \log \lambda_e/t \rfloor$. But the expected size of the minimum cut separating $e = \{u, v\}$ in H_j is at most $2t$ and appealing to the Chernoff bound, it has size at most $3t$ with high probability. Hence,

$$\Pr\left[e \in H_j\right] \approx \Pr\left[e \in G_j\right]$$

since H_j was a $3t$-skeleton. The claim follows since $\Pr\left[e \in G_j\right] \geq t/\lambda_e$.

4. SLIDING WINDOW

In this section, we consider processing graphs in the sliding window model. In this model we consider an *infinite* stream of edges $\langle e_1, e_2, \ldots \rangle$ but at time t we only consider the graph whose edge set consists of the last w edges,

$$W = \{e_{t-w+1}, \ldots, e_t\} \ .$$

We call these the *active* edges and we will consider the case where $w \geq n$. The results in this section were proved by Crouch et al. [22]. Note that some of sampling-based algorithms for counting small subgraphs are also applicable in this model.

4.1 Connectivity

We first consider testing whether the graph is k-edge connected for a given $k \in \{1, 2, 3, \ldots\}$. Note that $k = 1$ corresponds to testing connectivity. To do this, it is sufficient to maintain a set of edges $F \subseteq \{e_1, e_2, \ldots, e_t\}$ along with the time-of-arrival $\text{toa}(e)$ for each $e \in F$ such that for any cut, F contains the most recent k edges across the cut (or all the edges across the cut if there are less than k of them). Then, we can easily tell whether the graph of active edges is k-connected by checking whether F would be k-connected once we remove all edges $e \in F$ where $\text{toa}(e) \leq t - w$. This follows because if there are k or more edges among the last w edges across a cut, F will include the k most recent of these edges.

The following simple algorithm maintains the set

$$F = F_1 \cup F_2 \cup \ldots \cup F_k$$

where the F_i are disjoint and each is acyclic. We add the new edge e to F_1. If it completes a cycle, we remove the oldest edge in this cycle and add that edge to F_2. If we now have a cycle in F_2, we remove the oldest edge in this cycle and add that edge to F_3. And so forth.

Therefore, it is possible to test k-connectivity in the sliding window model using $O(kn\log n)$ space. Furthermore, by reducing other problems to k-connectivity, as discussed in the previous sections, this also implies the existence of algorithms for testing bipartiteness and constructing sparsifiers.

4.2 Matchings

We next consider the problem of finding large matchings in the sliding-window model. We will focus on the unweighted case and describe a $(3 + \epsilon)$-approximation. It is also possible to get a 9.027-approximation for the weighted case by combing this algorithm with a randomized rounding technique by Epstein et al. [25].

Algorithm. The approach for estimating the size of the maximum cardinality matching is based on the "smooth histograms" technique of Braverman and Ostrovsky [16].

The algorithm maintains maximal matchings over various "buckets" $B_1, \ldots, B_k$ where each bucket comprises of the edges in some suffix of the stream that have arrived so far. The buckets will always satisfy

$$B_1 \supseteq W \supsetneq B_2 \supsetneq \cdots \supsetneq B_k \tag{5}$$

where W is the set of active edges. Equation 5 implies that

$$m(B_1) \geq m(W) \geq m(B_2) \geq \ldots \geq m(B_k) \ ,$$

where $m(\cdot)$ denotes the size of the maximum matching on a sequence of edges.

Within each bucket B, we construct a greedy matching $\hat{M}(B)$ whose size we denote by $\hat{m}(B)$. There is potentially a bucket for each of the w suffixes and keeping a matching for each suffix would use too much space. To reduce the space usage, whenever two non-adjacent buckets have greedy matchings whose matching size is within a factor of $1-\beta$ where $\beta = \epsilon/4$, we will delete the intermediate buckets. Specifically, when a new edge e arrives, we update the buckets and matchings as follows:

Algorithm 6: Procedure for Updating Buckets

1 Create a new empty bucket B_{k+1};
2 Add e to each $\hat{M}(B_i)$ if possible;
3 **for** $i = 1, \ldots, k-2$ **do**
4 Find the largest $j > i$ such that

$$\hat{m}(B_j) \geq (1-\beta)\hat{m}(B_i)$$

Discard intermediate buckets and renumber;
5 If $B_2 = W$, discard B_1. Renumber the buckets;

Analysis. We will prove the invariant that for any $i < k$,

$$\hat{m}(B_{i+1}) \geq m(B_i)/(3+\epsilon)$$

or $|B_i| = |B_{i+1}| + 1$ or both. If $|B_i| \neq |B_{i+1}| + 1$, then we must have deleted some bucket B such that $B_i \subsetneq B \subsetneq B_{i+1}$. For this to have happened it must have been the case that $\hat{m}(B_{i+1}) \geq (1-\beta)\hat{m}(B_i)$ at the time. The next lemma shows that the optimal matching on the sequence of edges starting with B_i is not significantly larger than the greedy matching we find if we start with only start with B_{i+1}.

LEMMA 4.1. *For any sequence of edges C,*

$$m(B_i C) \leq \left(3 + \frac{2\beta}{1-\beta}\right) \hat{m}(B_{i+1}C) \ ,$$

where B_iC is the concatenation of B_i and C.

And hence we currently satisfy:

$$m(B_i) \leq \left(3 + \frac{2\beta}{1-\beta}\right) \hat{m}(B_{i+1}) \leq (3+\epsilon)\hat{m}(B_{i+1}) \ .$$

Therefore, either $W = B_1$ and $\hat{m}(B_1)$ is a 2-approximation for $m(W)$, or we have

$$m(B_1) \geq m(W) \geq m(B_2) \geq \hat{m}(B_2) \geq \frac{m(B_1)}{3+\epsilon}$$

and thus $\hat{m}(B_2)$ is a $(3+\epsilon)$-approximation of $m(W)$.

The fact that the algorithm does not use too much space follows from the way that the algorithm deletes buckets. Specifically, we ensure that for all $i \leq k-2$ we have $\hat{m}(B_{i+2}) < (1-\beta)\hat{m}(B_i)$. Since the maximum matching has size at most n, this ensures that the number of buckets is $O(\epsilon^{-1}\log n)$. Hence, the total number of bits used to maintain all k greedy matchings is $O(\epsilon^{-1} n \log^2 n)$.

5. CONCLUSIONS AND DIRECTIONS

There is now a large body of work on the design and analysis of algorithms for processing graphs in the data stream model. Problems that have received considerable attention include estimating connectivity properties, approximating graph distances, finding approximate matching, and counting the frequency of sub-graphs. The resulting algorithms combine existing data stream techniques with ideas from approximation algorithms and graph theory. By both identifying the state-of-the-art results and illustrating some of the techniques behind these results, it is hoped that this survey will be useful to both researchers that may want to use existing algorithms and to those that want to develop new algorithms for different problems.

There are numerous possible directions for future research. Naturally, it would be interesting to improve existing results. For example, does there exist a semi-streaming algorithm for constructing a spectral sparsifier when there are both edge insertions and deletions? What is the optimal approximation ratio for estimating the size and weight of maximum matchings? Other specific questions can be found at the wiki,

`sublinear.info` .

More general, open-ended research directions include:

1. *Directed Graphs.* Relatively little is known about processing directed graphs and yet many natural graphs are directed. For example, it is known that any semi-streaming algorithm testing s-t connectivity requires $\Omega(\log n)$ passes [33] but is this number of passes sufficient? If we could estimate the size of flows in directed graphs, this could lead to better algorithms for approximating the size of bipartite matchings.

2. *Communication Complexity.* The recent results on graph sketching imply surprisingly efficient communication protocols; if the rows of an adjacency

matrix are partitioned between n players, then the connectivity properties of the graph can be inferred from a $O(\text{polylog}\, n)$ bit message from each player. In contrast, if the partition of the entries is arbitrary, the players needs to send $\tilde{\Omega}(n)$ on average [55]. What other graph problems can be solved using only short messages? What if each player also knows the neighbors of the neighbors of a node? From a different perspective, establishing reductions from communication complexity problems is a popular approach for proving lower bounds in the data stream model. But less is known about graph stream lower bounds because it is often harder to decompose graph problems into multiple simpler "independent" problems and use existing communication complexity techniques.

3. *Stream Ordering.* The analysis of stream algorithms has traditionally been "doubly worst case" in the sense that the contents of the stream *and* the ordering of the stream are both chosen adversarially. If we relax the second assumption and assume that the stream is ordered randomly (or that the stream is stochastically generated), can we design algorithms that take advantage of this? Some recent work is already considering this direction [19, 43, 47]. Alternatively, it may be interesting to further explore the complexity of various graph problems under specific edge orderings, e.g., sorted-by-weight, grouped-by-endpoint, or orderings tailored to the problem at hand [57].

4. *More or Less Space.* Research focusing on the semi-streaming model has been very fruitful and many interesting techniques have been developed that have had applications beyond stream computation. However, the model itself is not suited to process sparse graphs where $m = \tilde{O}(n)$. While many basic problems require $\Omega(n)$ space, this does not preclude smaller-space algorithms if we may make assumptions about the input or only need to "property test" the input [32], i.e., we just need to distinguish graphs with a given property from graphs that are "far" from having the property. Do such algorithms exist? Alternatively, what if we are permitted *more* than $\tilde{O}(n\, \text{polylog}\, n)$ space? Various lower bounds, such as those for approximate unweighted matching, are very sensitive to the exact amount of space available and nothing is known if we may use $O(n^{1.1})$ space for example.

Acknowledgements. Thanks to Graham Cormode, Sagar Kale, Hoa Vu, and an anonymous reviewer for numerous helpful comments.

6. REFERENCES

[1] K. J. Ahn. *Analyzing massive graphs in the semi-streaming model*. PhD thesis, University of Pennsylvania, Philadelphia, Pennsylvania, Jan. 2013.

[2] K. J. Ahn and S. Guha. Graph sparsification in the semi-streaming model. In *International Colloquium on Automata, Languages and Programming*, pages 328–338, 2009.

[3] K. J. Ahn and S. Guha. Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints. *CoRR*, abs/1307.4359, 2013.

[4] K. J. Ahn and S. Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. *Inf. Comput.*, 222:59–79, 2013.

[5] K. J. Ahn, S. Guha, and A. McGregor. Analyzing graph structure via linear measurements. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 459–467, 2012.

[6] K. J. Ahn, S. Guha, and A. McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *ACM Symposium on Principles of Database Systems*, pages 5–14, 2012.

[7] K. J. Ahn, S. Guha, and A. McGregor. Spectral sparsification of dynamic graph streams. In *International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, 2013.

[8] M. Badoiu, A. Sidiropoulos, and V. Vaikuntanathan. Computing s-t min-cuts in a semi-streaming model. *Manuscript*.

[9] B. Bahmani, R. Kumar, and S. Vassilvitskii. Densest subgraph in streaming and mapreduce. *PVLDB*, 5(5):454–465, 2012.

[10] Z. Bar-Yossef, R. Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 623–632, 2002.

[11] S. Baswana. Streaming algorithm for graph spanners - single pass and constant processing time per edge. *Inf. Process. Lett.*, 106(3):110–114, 2008.

[12] J. D. Batson, D. A. Spielman, and N. Srivastava. Twice-ramanujan sparsifiers. *SIAM J. Comput.*, 41(6):1704–1721, 2012.

[13] L. Becchetti, P. Boldi, C. Castillo, and A. Gionis. Efficient algorithms for large-scale local triangle counting. *TKDD*, 4(3), 2010.

[14] A. A. Benczúr and D. R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In *ACM Symposium on Theory of Computing*, pages 47–55, 1996.

[15] B. Bollobás. *Extremal Graph Theory*. Academic Press, New York, 1978.

[16] V. Braverman and R. Ostrovsky. Smooth histograms for sliding windows. In *IEEE Symposium on Foundations of Computer Science*, pages 283–293, 2007.

[17] V. Braverman, R. Ostrovsky, and D. Vilenchik. How hard is counting triangles in the streaming model? In *International Colloquium on Automata, Languages and Programming*, pages 244–254, 2013.

[18] L. S. Buriol, G. Frahling, S. Leonardi, A. Marchetti-Spaccamela, and C. Sohler. Counting triangles in data streams. In *ACM Symposium on Principles of Database Systems*, pages 253–262, 2006.

[19] A. Chakrabarti, G. Cormode, and A. McGregor. Robust lower bounds for communication and stream computation. In *ACM Symposium on Theory of Computing*, pages 641–650, 2008.

[20] A. Chakrabarti and S. Kale. Submodular maximization meets streaming: Matchings, matroids, and more. *CoRR*, arXiv:1309.2038, 2013.

[21] G. Cormode and S. Muthukrishnan. Space efficient mining of multigraph streams. In *ACM Symposium on Principles of Database Systems*, pages 271–282, 2005.

[22] M. S. Crouch, A. McGregor, and D. Stubbs. Dynamic graphs in the sliding-window model. In *European Symposium on Algorithms*, pages 337–348, 2013.

[23] M. Elkin. Streaming and fully dynamic centralized algorithms for constructing and maintaining sparse spanners. *ACM Transactions on Algorithms*, 7(2):20, 2011.

[24] M. Elkin and J. Zhang. Efficient algorithms for constructing $(1+\epsilon, \beta)$-spanners in the distributed and streaming models. *Distributed Computing*, 18(5):375–385, 2006.
[25] L. Epstein, A. Levin, J. Mestre, and D. Segev. Improved approximation guarantees for weighted matching in the semi-streaming model. *SIAM J. Discrete Math.*, 25(3):1251–1265, 2011.
[26] L. Epstein, A. Levin, D. Segev, and O. Weimann. Improved bounds for online preemptive matching. In *STACS*, pages 389–399, 2013.
[27] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2-3):207–216, 2005.
[28] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. Graph distances in the data-stream model. *SIAM Journal on Computing*, 38(5):1709–1727, 2008.
[29] W. S. Fung, R. Hariharan, N. J. A. Harvey, and D. Panigrahi. A general framework for graph sparsification. In *ACM Symposium on Theory of Computing*, pages 71–80, 2011.
[30] A. Goel, M. Kapralov, and S. Khanna. On the communication and streaming complexity of maximum bipartite matching. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 468–485, 2012.
[31] A. Goel, M. Kapralov, and I. Post. Single pass sparsification in the streaming model with edge deletions. *CoRR*, abs/1203.4900, 2012.
[32] O. Goldreich. Introduction to testing graph properties. In O. Goldreich, editor, *Studies in Complexity and Cryptography*, volume 6650 of *Lecture Notes in Computer Science*, pages 470–506. Springer, 2011.
[33] V. Guruswami and K. Onak. Superlinear lower bounds for multipass graph processing. In *IEEE Conference on Computational Complexity*, pages 287–298, 2013.
[34] B. V. Halldórsson, M. M. Halldórsson, E. Losievskaja, and M. Szegedy. Streaming algorithms for independent sets. In *International Colloquium on Automata, Languages and Programming*, pages 641–652, 2010.
[35] M. M. Halldórsson, X. Sun, M. Szegedy, and C. Wang. Streaming and communication complexity of clique approximation. In *International Colloquium on Automata, Languages and Programming*, pages 449–460, 2012.
[36] M. R. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. *External memory algorithms*, pages 107–118, 1999.
[37] M. Jha, C. Seshadhri, and A. Pinar. A space efficient streaming algorithm for triangle counting using the birthday paradox. In *KDD*, pages 589–597, 2013.
[38] M. Jha, C. Seshadhri, and A. Pinar. When a graph is not so simple: Counting triangles in multigraph streams. *CoRR*, arXiv:1310.7665, 2013.
[39] H. Jowhari and M. Ghodsi. New streaming algorithms for counting triangles in graphs. In *COCOON*, pages 710–716, 2005.
[40] H. Jowhari, M. Saglam, and G. Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In *ACM Symposium on Principles of Database Systems*, pages 49–58, 2011.
[41] D. M. Kane, K. Mehlhorn, T. Sauerwald, and H. Sun. Counting arbitrary subgraphs in data streams. In *International Colloquium on Automata, Languages and Programming*, pages 598–609, 2012.
[42] M. Kapralov. Better bounds for matchings in the streaming model. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1679–1697, 2013.
[43] M. Kapralov, S. Khanna, and M. Sudan. Approximating matching size from random streams. In *ACM-SIAM Symposium on Discrete Algorithms*, 2014.
[44] B. M. Kapron, V. King, and B. Mountjoy. Dynamic graph connectivity in polylogarithmic worst case time. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1131–1142, 2013.
[45] D. R. Karger. Random sampling in cut, flow, and network design problems. In *ACM Symposium on Theory of Computing*, pages 648–657, 1994.
[46] J. A. Kelner and A. Levin. Spectral sparsification in the semi-streaming setting. *Theory Comput. Syst.*, 53(2):243–262, 2013.
[47] C. Konrad, F. Magniez, and C. Mathieu. Maximum matching in semi-streaming with few passes. In *APPROX-RANDOM*, pages 231–242, 2012.
[48] C. Konrad and A. Rosén. Approximating semi-matchings in streaming and in two-party communication. In *International Colloquium on Automata, Languages and Programming*, pages 637–649, 2013.
[49] K. Kutzkov and R. Pagh. On the streaming complexity of computing local clustering coefficients. In *WSDM*, pages 677–686, 2013.
[50] M. Manjunath, K. Mehlhorn, K. Panagiotou, and H. Sun. Approximate counting of cycles in streams. In *European Symposium on Algorithms*, pages 677–688, 2011.
[51] A. McGregor. Finding graph matchings in data streams. In *APPROX-RANDOM*, pages 170–181, 2005.
[52] S. Muthukrishnan. *Data Streams: Algorithms and Applications*. Now Publishers, 2006.
[53] R. Pagh and C. E. Tsourakakis. Colorful triangle counting and a mapreduce implementation. *Inf. Process. Lett.*, 112(7):277–281, 2012.
[54] A. Pavan, K. Tangwongsan, S. Tirthapura, and K.-L. Wu. Counting and sampling triangles from a graph stream. In *International Conference on Very Large Data Bases*, 2013.
[55] J. M. Phillips, E. Verbin, and Q. Zhang. Lower bounds for number-in-hand multiparty communication complexity, made easy. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 486–501, 2012.
[56] A. D. Sarma, S. Gollapudi, and R. Panigrahy. Estimating pagerank on graph streams. *J. ACM*, 58(3):13, 2011.
[57] A. D. Sarma, R. J. Lipton, and D. Nanongkai. Best-order streaming model. *Theor. Comput. Sci.*, 412(23):2544–2555, 2011.
[58] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, 2011.
[59] D. A. Spielman and S.-H. Teng. Spectral sparsification of graphs. *SIAM J. Comput.*, 40(4):981–1025, 2011.
[60] R. Tarjan. *Data Structures and Network Algorithms*. SIAM, Philadelphia, 1983.
[61] A. B. Varadaraja. Buyback problem - approximate matroid intersection with cancellation costs. In *International Colloquium on Automata, Languages and Programming*, pages 379–390, 2011.
[62] M. Zelke. Weighted matching in the semi-streaming model. *Algorithmica*, 62(1-2):1–20, 2012.

A System for Energy-Efficient Data Management

Yi-Cheng Tu [1], Xiaorui Wang [2], Bo Zeng [3], and Zichen Xu [2]
[1] *Dept. of Computer Science and Engineering, University of South Florida, U.S.A.*
[2] *Dept. of Electrical and Computer Engineering, The Ohio State University, U.S.A.*
[3] *Dept. of Industrial and Management Systems Engineering, University of South Florida, U.S.A.*
Emails: ytu@cse.usf.edu, xwang@ece.osu.edu, bzeng@usf.edu, xu.925@osu.edu

ABSTRACT

Energy consumption of computer systems has increased at a steep rate in recent years. Following extensive energy-related research and practice in the hardware and OS communities, much attention has been paid to developing energy-efficient applications. With database systems being a heavy energy consumer in modern data centers, we face the challenge of designing DBMSs with energy as a first-class performance goal. This paper presents our on-going work in designing and implementing a DBMS that enables significant energy conservations while maintaining other performance targets. We follow two new strategies in DBMS implementation to achieve our system design goal. The first one is to change the resource consumption patterns via energy-aware query optimization and reorganizing data records to enable load consolidation in disks. The second strategy is active control of power modes of hardware (i.e., CPU and hard disks) toward energy reduction. Specifically, we use control-theoretic techniques to allow dynamic adjustment of CPU frequency and online data migration to achieve disk load consolidation. Preliminary results have shown the effectiveness of our design.

1. INTRODUCTION

The past decade has witnessed prosperity in green computing research, with a focus on making low-level components (i.e., hardware and OS) of a computing system energy-efficient. Much interest has been shifted to the application level in the recent few years, with the belief that system-level energy management mechanisms can be ineffective without the assistance of energy-aware applications. We believe databases serve as a salient example that supports this view: a DBMS is much like an OS itself by managing resources (e.g., disk, memory buffer) acquired from the real OS for different tasks (i.e., query processing, indexing). Therefore, there is the need to develop DBMSs with energy efficiency as a first-class performance goal. Such work is also well motivated by the substantial economical/social benefits it brings, as databases are found to be a major energy sink in a typical data center [11].

This paper presents our on-going work in designing and implementing an energy-efficient DBMS (E^2-DBMS) that enables significant energy conservation as compared to traditional DBMSs. Specifically, the design goal of E^2DBMS can be formulated as an optimization problem: *minimize energy consumption while maintaining a certain level of query processing performance*. Note that the constraint of the problem is essential as efficient query processing is always the main concern in a database service.

Related Work. Our community has shown much interest in green databases over the past few years. Earlier work [5, 7] provided high-level ideas on improving energy efficiency of data- bases, and both emphasized energy-aware query optimization. Another work [9] proposed query rescheduling and CPU frequency control as two means for green data management and supported their claims with experimental results. Various other topics such as energy quantification of database servers [11, 12, 15], benchmarking [13], cost-based query plan evaluation [8] are also reported. As compared to the above projects, we focus on a systematic framework for energy conservation inside a DBMS rather than individual means. The key challenge is to identify the main components in existing DBMSs that lead to low energy efficiency and relevant mechanisms to alleviate the problem. For that, work by Lang *et al.* [9] is close in spirit to ours. The proposed E^2DBMS system, however, is unique in that: (1) we consider energy-aware storage management as a means for energy reduction, and (2) we use formal control-theoretic methods (instead of heuristics) to ensure performance goals are met.

2. OVERVIEW OF OUR APPROACH

In E^2DBMS, we mainly explore the following two strategies to improve database energy efficiency.

Modifying resource consumption patterns. The

main idea is to tune the DBMS towards energy-efficient computational paths and system configurations. The intuition behind this is: traditional DBMSs are optimized towards query performance only, therefore we can search the space of all system states to locate those that carry low power cost and a small performance penalty in database operations. Clearly, the effectiveness of such an idea depends on the existence of plans/configurations in the search space that provide aforementioned tradeoff between energy and performance. While one report [15] showed concern on the existence of such tradeoffs, we believe there are abundant opportunities, as supported by our previous work [16] and evidence shown in this paper.

Harnessing low-power modes of hardware. While modifying resource consumption pattern achieves platform-free energy saving, controling the low-power modes of hardware may derive more dramatic gains. An effective way to reduce energy consumption of computer systems is to transition the hardware components from high-power states to low-power states [3] when performance allows. Most components in a modern database server such as processors [14], main memory [4], and hard disks [6] have adjustable power states. Components are fully operational, but consume much less power while having degraded functionality in low-power states.

In the following sections, we elaborate on concrete mechanisms to implement the above strategies in a DBMS as well as the technical challenges. There are two important observations about our solutions. First, the mechanisms we propse do not aim at finding the fastest execution path for queries. Instead, they often involve identifying the best tradeoff between energy and performance. This also means that existing DBMSs optimized towards query processing time are not energy-efficient platforms. Second, such mechanisms will not be effective if deployed at the OS level, since the modeling, monitoring, and configuration of database system can only be realized inside the DBMS. This necessitates DBMS-level efforts in making databases green.

The following discussions are organized based on the target (hardware) resource of the relevant mechanisms. Previous work [11, 15, 16] has shown that CPU and storage system are the main consumers of *active energy* [1] in a typical database system. Memory also draws significant amount of power but we do not believe it provides meaningful energy saving

[1] The portion of energy consumed by database workload thus can be controlled by E^{2}DBMS. In opposite to that is the *base energy* – energy consumed by the system when no workload is issued and all hardware run on their lowest power modes.

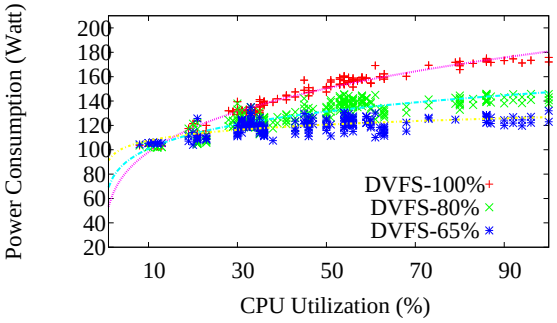


Figure 1: CPU power consumption in a database server under different workload intensities and DVFS levels.

opportunities. Since the performance of database depends heavily on the amount of memory, turning memory chips into sleep mode will lead to excessively long query processing time, thus eating the gain of running in low-power modes.

CPU Resource Control. The mechanism we harness to change CPU usage pattern is *energy-aware query optimization*. CPU is a *power proportional* hardware, meaning its power consumption is (roughly) proportional to the intensity of the computational load it handles (Fig. 1). Significant power can be saved if we can find query execution plans that do not require as much CPU time as those found by existing query optimizers. Such plans will also be energy-efficient if they require the same or slightly longer I/O time, and they are often ignored by the query optimizer in existing DBMSs since the latter only considers performance in the query cost model. In our previous work [16], we systematically studied the power profiles of various database benchmarks, and found that queries with such low-CPU plans that are missed by traditional query optimizers are very common (e.g., in 10 out of the 19 queries we studied in TPC-H). Therefore, we can design a novel query optimizer to identify such plans upon evaluating the energy cost of alternative plans. Note that hard disks are not energy-proportional therefore this mechanism has little effects on disk energy consumption.

For modern CPUs, it is well-known that the Dynamic Voltage and Frequency Scaling (DVFS) technique can allow *cubic reductions* in power density relative to performance loss [14]. This has been verified using various database workloads in previous work [9] and our experiments (Fig. 1). Therefore, we design a DVFS controller of CPUs as an integrated part of E^{2}DBMS to harvest the power savings generated by low DVFS levels. The power saving potential from adjusting DVFS is significant: a

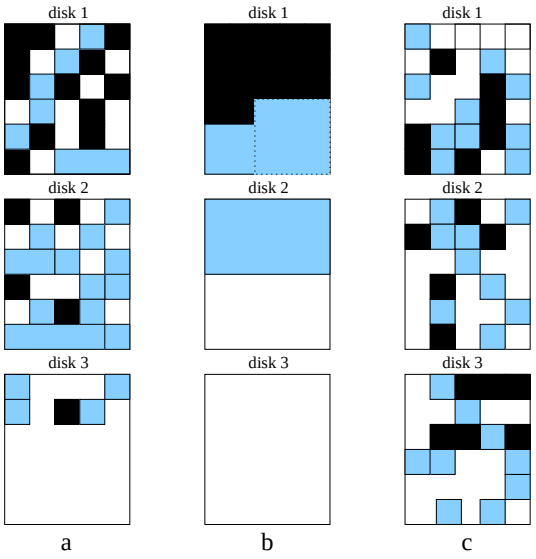


Figure 2: Three distributions of data blocks on three hard disks. Darker color means higher data popularity in block

60W power difference can be observed in the server we tested (Fig. 1). This confirms the results from [9], in which a 45% reduction of active energy is reported. On the other hand, since query processing time is often dominated by I/O time (especially in I/O-bound workloads), a performance penalty associated with low-power CPU modes will not affect system performance dramatically. Modern CPUs typically provide OS-level handles for efficient adjustment of DVFS levels, making DVFS control extremely light-weighted. Note that the proposed CPU control will not be effective if deployed on the OS level, since the control decisions depend heavily on the collection and analysis of database states.

Storage Management. Traditional multi-disk storage systems seek *load balancing* to achieve best I/O performance. In E^{2}DBMS, however, the storage manager follows a *load consolidation* approach by putting most I/O load into a subset of the physical disks. By this, energy saving opportunities are created since disks with low I/O load can spin down or enter low power/performance mode. Specifically, we explore an *energy-aware data placement* mechanism that changes the data access patterns in individual disks via dynamically reconfiguring data placement at runtime. Note that the key problem in load consolidation is to balance energy savings and disk contention (i.e., data access performance). For example, the placement scheme shown in Fig. 2b will maximize power savings but yield unacceptable performance for disk 1. On the contrary, Fig. 2a shows a better solution by distributing hot items into two disks. In the load balancing scheme (Fig. 2c), blocks of all color depth will be randomly distributed into all disks thus energy reduction will be minimized. The key issue for the data placement algorithm is to provide the highest level of load consolidation with a small overhead, and maintain the service quality. Obviously, this mechanism only provides higher level of load consolidation - it requires the collaboration of low-level mechanisms that can change the power modes of disks to actually save energy.

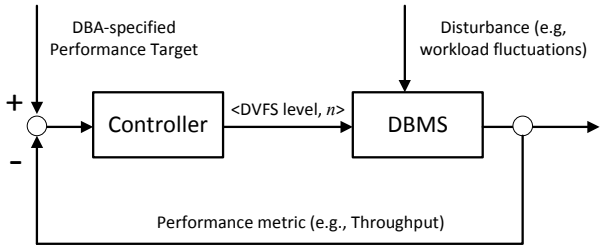


Figure 3: The feedback control loop for energy-efficient CPU usage

Traditional disk farms are notoriously energy inefficient: all disks used to hold the database keep spinning at full speed at all times while user accesses are often skewed towards a very small fraction of the database [10]. This clearly implies a big waste of energy and also opportunities for optimization. Fortunately, most hard disks shipped today can be transitioned to low-power modes with a fraction of the power consumption in active mode. Furthermore, the emergence of multi-speed hard disks [6] offer more flexible power/performance tradeoffs. With the aforementioned dynamic data placement mechanism to generate opportunities for load consolidation, the main problem is to develop a dynamic power management (DPM) module to determine the exact power mode for each disk.

3. E^{2}DBMS DESIGN ISSUES

This section briefly introduces the technical challenges and our solutions in implementing E^{2}DBMS towards its design goal at runtime.

Energy-aware CPU control. For CPUs, we map the design problem into a feedback control loop (Fig. 3) that periodically changes the database and hardware configurations (i.e., the input signal in the loop) such that the system performance (i.e., the output signal) traces a target specified by the DBA. In this feedback loop, the system input (i.e., control signal) is a multi-dimensional value: the DVFS level of CPUs and a parameter n that determines the preference of power over performance in query

optimization . The output signal is a system-level performance metric such as query response time or throughput. The intuition of our design comes from the fact that *there is a monotonic relationship between power and performance* when the system runs under different configurations. Therefore, if the query processing performance of the DBMS is unnecessarily better than the threshold, we can decrease the DVFS level of the CPUs for power savings, or be more aggressive in choosing power-efficient query plans. If the system performance drops below the threshold, we can do the opposite to achieve faster query processing. *As long as we keep the actual system performance on the desirable level, system power will be minimized.* The first challenge is an actuator of the control signal, i.e., mechanisms to ensure the DBMS run in the state specified by the signal. The second challenge is the choice of the input signal whose value enables different tradeoffs between power and performance in query optimization. More importantly, we need to make the system trace the desired performance threshold in a dynamic environment.

The DVFS part of the control signal can be easily enforced by a light-weighted system call. In the Intel CPU we tested, this can be done by writing the DVFS level into a cached system file with a 20 microsecond overhead. To enforce the tradeoff set by the parameter n, our previous work [16] presents an *energy-aware query optimizer* that selects query plans based on this signal. For each query plan, the optimizer evaluates its running time T and power cost P, and gives a score as $C = PT^n$ – the plan with the lowest score is selected. Here a big problem is the accurate estimation of P, we propose a two-level approach for this: we first use standard simple workloads to generate the static power cost of individual database operators, and use those as parameters to derive the total cost of an entire plan. Then such parameters are updated periodically following a *Recursive Least Square* manner to gain better accuracy in a dynamic environment.

The second challenge is often seen in database tuning problems. We use well-established control-theoretic system modeling and controller design techniques to achieve our design goal. As compared to traditional heuristics-based database tuning methods, control theory provides: (1) guaranteed control performance such as accuracy, stability, and short settling time; and (2) tolerance of modeling errors and fluctuations of system/workload features. The control loop design will begin by modeling the relationship between system input (i.e., control signal) and output (i.e., performance). We will use *system identification* techniques to derive such models via experiments with standard inputs. The next step is to design the controller towards guaranteed performance. Our plan is to apply the Proportional-Integral (PI) control theory to achieve stability and zero steady state control – the latter ensures maximal energy saving.

Energy-aware storage system. For the storage system, the main issue is to find the proper level of load consolidation and power modes in disk arrays to meet the performance and power requirements. Here our system design goal can be viewed as a classical optimization with performance target as a constraint and energy as the optimization target. As an input to this problem, the access rate of individual data items typically changes over time in a database system. This requires the optimization solution to be recomputed therefore efficient algorithm is needed. Another fact that complicates the problem is that the realization of a new optimization solution, unlike the adjustment of DVFS and query optimizer parameter, bears very high costs. For example, changing the data placement scheme (to reach a new level of load consolidation) involves significant I/O (and power) cost by moving data around. Transitioning disks to another state also involves a performance penalty on the level of seconds [6]. In this project, we tackle load consolidation in storage systems by treating it as a *dynamic optimization* problem and propose efficient algorithms with provable accuracy bounds.

The main task is to develop an integrated optimization model for data migration and disk state adjustment and efficient algorithms to achieve the optimal (or near-optimal) balance between power consumption and query performance. Starting from an M/G/1 queueing model for single-disk behaviour, we have developed a mixed integer programming model over multiple periods for inter-disk fragment migration and have tested it with industrial solvers. The next step would be to analyze the model to identify structural properties of optimal solutions for the purpose of creating guidelines to reduce search space for computing algorithms.

Another interesting problem is a DBMS mechanism to reorganize data records for greater chances of load consolidation. Our studies show that the data popularity can be traced down to individual tuple level in a typical database. Without extra efforts, a data page will hold records with different levels of popularity thus all pages have similar aggregated page-level popularity. Referring this to Fig. 2, all blocks will have similar color depth and load consolidation becomes impossible. We will

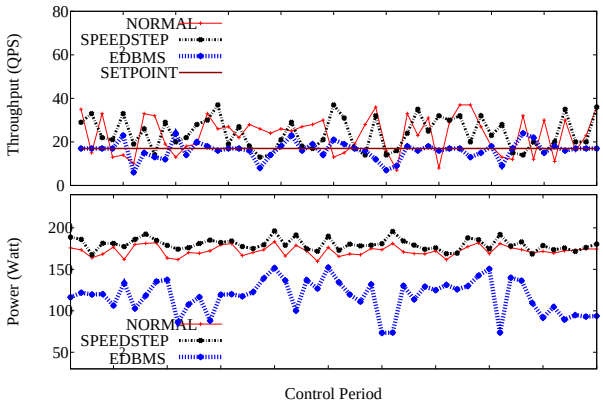


Figure 4: Database throughput and active power consumption in 50 control periods

use a *popularity-based intra-disk data fragmentation* method to dynamically organize database records in a disk such that data items with similar temperature are stored in consecutive pages. Specifically, we divide data on a disk into many fragments, each of which occupy a fixed number of consecutive pages. At runtime, we monitor the access frequency of data records and dynamically place records into a proper fragment according to their popularity. Such fragments will be used as the basic unit of data migration mentioned above. We propose an *incremental restoration* algorithm to solve such a tuple packing problem – the main idea is to constrain data reorganization to resident (in-memory) pages as much as we can, thus minimizing extra I/O cost.

4. PROGRESS AND INITIAL RESULTS

We are in the process of actively studying the above issues and implementing the system. An early version of E^{2}DBMS that augments the PostgreSQL kernel with the energy-aware query optimizer was made public [17]. In this section, we report part of our experimental results with a focus on revealing potential of the E^{2}DBMS system.

We have built an initial version of the control framework proposed in Figure 3. In this framework, we designed and implemented a proportional-integral (PI) controller that tracks system throughput to a reference value via adjusting DVFS levels of the CPU. Such a design is based on a dynamic DBMS system model derived from comprehensive system identification study. The implemented framework was tested for its robustness under various database workloads and environmental setups. For example, Fig. 4 shows the performance and power consumption in 50 control periods during one experiment, in which the throughput setpoint is 18

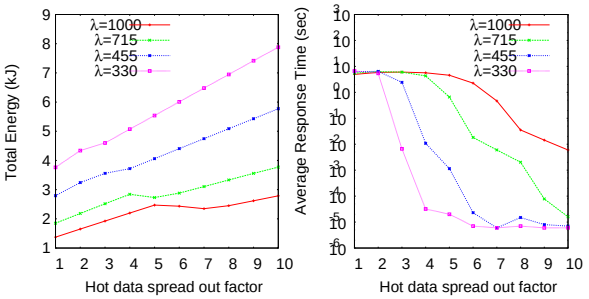


Figure 5: Energy cost and response time of a read-only workload on a 10-disk system

queries per second. We fed the database with a composite workload that consists of TPC-C and TPC-H queries in a "racing" environment, in which multiple CPU-intensive programs compete with the DBMS for CPU resources. The two baselines are "Normal" that always sets the DVFS level to 100% and "SpeedStep" – an Intel built-in technique to adjust DVFS on the OS level. Compared with those two baselines, our controller (i.e., "E^{2}DBMS") maintains the throughput (blue dotted line) around the setpoint without introducing too many overshoots thus delivers significant energy savings – 51.3% more savings than the OS-level mechanism "SpeedStep" was observed. Other empirical results and controller stability analysis of the CPU control framework can be found in [18].

We also performed simulation experiments to verify our ideas in power-aware storage management. In this set of experiments, we simulate a database table whose tuples span 10 hard disks. We follow the well-established b/c model [10] in generating random read workloads. It is well known that database access patterns are highly skewed and can be described as an 80/20 or even a 90/10 model. Given the popularity of tuples, we implement a static data fragmentation scheme such that the c% hot tuples are placed into a variable number F (called *data spread-out factor*) of the 10 disks. Disk specifications are based on those of a commercial product – the Hitachi UltraStar 7K4000 [1]. Such disks have two power modes: a high power mode that consumes 11.4 watts at 7200rpm spinning speed and a low power mode with 4.6 watts at 6300rpm. We assume the hot disks always run at 11.4w/7200rpm and the cold disks at 4.6w/6300rpm.

Fig. 5 shows the energy consumption and performance of such schemes under four 90/10 workloads with task arrival rate ranging from 330/s to 1000/s. By comparing the total consolidation ($F = 1$) case with the load balancing ($F = 10$) case, energy savings of 50.8% (for $\lambda = 1000$) to 52.2% (for $\lambda = 330$)

can be observed.[2] However, the $F = 1$ cases show bad performance - average read response time of all disks is on the level of seconds. However, the response time drops quickly when we spread the load to more disks. For example, it reaches the level of milliseconds for $\lambda = 330$ when the hot items are spread into only three disks. We believe the above results clearly show the potential of our idea of performing DBMS-level load consolidation. Specifically, those F values in the middle range provide desirable tradeoffs between energy and performance - about 50% energy can be saved with little performance degradation. OS-level consolidation, which is simulated by those with high F vales, can only reach a state of balanced load without saving any energy. Note that all results presented in this section only considered direct energy savings. If we consider energy savings from cooling systems,[3] the total energy reduction will be even higher.

5. SUMMARY

In this paper, we argue for the great potential of energy-efficient database systems. Such potential can only be realized via designing DBMSs with energy consumption as a first-class performance goal. We present our work in building such a DBMS named E^{2}DBMS, which achieves high energy efficiency via two strategies: modifying resource consumption patterns and controlling power modes of hardware. Such strategies are implemented through a series of mechanisms that target at the main active energy consumers in a typical database server: CPU and disks. Experiments run on an initial E^{2}DBMS prototype and simulation platform demonstrate significant energy savings.

Acknowledgements

This work is supported by US National Science Foundation (NSF) grants IIS-1117699 and IIS-1156435.

6. REFERENCES

[1] Ultrastar 7k4000 OEM specification. http://www.hgst.com/tech/techlib.nsf/products/ultrastar_7k4000.

[2] F. Ahmad and T. N. Vijaykumar. Joint optimization of idle and cooling power in data centers while maintaining response time. In *ASPLOS'10*, pages 243–256, 2010.

[2]The low intensity workload consumed more energy because our workload contains a fixed number (20,000) of operations. Disk idle time is longer in low intensity cases. This also shows that a higher percentage of energy can be saved when system is not heavily loaded.

[3]According to [2], for one watt of direct power saving, one to two watts can be saved from cooling.

[3] R. Bianchini and R. Rajamony. Power and energy management for server systems. *IEEE Computer*, 37(11):68–74, 2004.

[4] W. Felter *et al.* A performance-conserving approach for reducing peak power consumption in server systems. In *Procs. Intl. Conf. Supercomputing (ICS)*, 2005.

[5] G. Graefe. Database servers tailored to improve energy efficiency. In *Procs. EDBT Workshop on Software Engineering for Tailor-made Data Management*, 2008.

[6] S. Gurumurthi *et al.* DRPM: Dynamic Speed Control for Power Management in Server Class Disks. In *ISCA*, pages 169–179, 2003.

[7] S. Harizopoulos *et al.* Energy efficiency: The new holy grail of data management systems research. In *CIDR*, 2009.

[8] M. Kunjir *et al.* Peak Power Plays in Database Engines. In *EDBT*, 2012.

[9] W. Lang and J. Patel. Towards eco-friendly database management systems. In *CIDR*, 2009.

[10] M. Nicola and M. Jarke. Performance modeling of distributed and replicated databases. *IEEE Trans. Knowledge and Data Engineering*, 12(4):645–672, Jul/Aug 2000.

[11] M. Poess and R. Nambiar. Energy cost, the key challenge of today's data centers: a power consumption analysis of TPC-C results. *Proc. of VLDB*, 1(2):1229–1240, 2008.

[12] M. Poess and R. Nambiar. Tuning servers, storage and database for energy efficient data warehouses. In *ICDE*, 2010.

[13] M. Poess *et al.* Energy benchmarks: a detailed analysis. In *TPCTC*, pages 131–140, 2010.

[14] K. Skadron *et al.* Temperature-aware microarchitecture: Modeling and implementation. *ACM Trans. Architecture and Code Optimization*, 1(1), 2004.

[15] D. Tsirogiannis *et al.* Analyzing the energy efficiency of a database server. In *SIGMOD*, pages 231–242, 2010.

[16] Z. Xu, Y.-C. Tu, and X. Wang. Exploring power- performance tradeoffs in database systems. In *ICDE*, pages 485–496, 2010.

[17] Z. Xu, Y.-C. Tu, and X. Wang. PET: Reducing database energy cost via query optimization. In *Procs. VLDB Endowment*, 2012.

[18] Z. Xu, X. Wang, and Y.-C. Tu. Power-Aware Throughput Control for Database Management Systems. In *the 10th International Conference on Autonomic Computing*, pages 315–324, 2013.

The Linked Data Benchmark Council: a Graph and RDF industry benchmarking effort

Renzo Angles[1,2], Peter Boncz[3], Josep Larriba-Pey[4], Irini Fundulaki[5], Thomas Neumann[6], Orri Erling[7], Peter Neubauer[8], Norbert Martinez-Bazan[9], Venelin Kotsev[10], Ioan Toma[11]
[1]Universidad de Talca, Chile; [2]VU University Amsterdam, The Netherlands;
[3]CWI, Amsterdam, The Netherlands; [4]Universitat Politècnica de Catalunya, Spain;
[5]FORTH, Heraklion, Greece; [6]Technische Universität München, Munich, Germany;
[7]OpenLink Software, United Kingdom; [8]Neo Technology, Sweden;
[9]Sparsity Technologies, Spain; [10]Ontotext AD, Bulgaria; [11]Universität Innsbruck, Austria
rangles@utalca.cl, boncz@cwi.nl, larri@ac.upc.edu, fundul@ics.forth.gr,
neumann@in.tum.de, oerling@openlinksw.com, peter.neubauer@neotechnology.com,
norbert@sparsity-technologies.com, venelin.kotsev@ontotext.com, ioan.toma@sti2.at

ABSTRACT

The Linked Data Benchmark Council (LDBC) is an EU project that aims to develop industry-strength benchmarks for graph and RDF data management systems. It includes the creation of a non-profit LDBC organization, where industry players and academia come together for managing the development of benchmarks as well as auditing and publishing official results. We present an overview of the project including its goals and organization, and describe its process and design methodology for benchmark development. We introduce so-called "choke-point" based benchmark development through which experts identify key technical challenges, and introduce them in the benchmark workload. Finally, we present the status of two benchmarks currently in development, one targeting graph data management systems using a social network data case, and the other targeting RDF systems using a data publishing case.

1. INTRODUCTION

Graph and RDF databases have been created to support the storing and analysis of complex relationships in highly interconnected data occurring in application domains like e.g. social network analysis, linked open data, etc. Both data management technologies hold the notion of graph as their main abstraction mechanism for data modeling and querying. These technologies are relatively young, certainly in their tradition of industry-strength products supporting this graph data model, yet have aroused significant interest from users looking beyond the relational model. In order for practitioners to compare these new data management systems with each other, and with established relational technology, benchmarks can play a helpful role. Relevant benchmark challenges can further help spur technological progress, to more quickly mature these nascent industries.

Current graph and RDF benchmarks, however, do not fully attain all the desirable characteristics [5] (i.e., relevant, repeatable, fair, verifiable and economical), and sometimes neglect the particularities and requirements in RDF and graph data management [2, 3, 4, 7] (e.g. complex graph queries over irregularly shaped and correlated data).

The Linked Data Benchmark Council (LDBC)[1] is an EU project that brings together a community of academic researchers and industry, whose main objective is the development of open source, yet industrial grade, benchmarks for graph and RDF databases. The founding industry members of LDBC are the graph database companies Neo Technologies and Sparsity Technologies, and the RDF database companies Ontotext and OpenLink Systems. A result of the project will be the LDBC non-profit organization, open for worldwide industry participation, which during and after the end of the EU project will supervise the creation and maintenance of the benchmarks as well as the activities for obtaining, auditing and publishing the benchmarking results.

This paper describes the goals of the LDBC as well as its organizational structure. It also presents the status of two benchmarks in current development: a graph data management benchmark based on the social network use-case, and an industry-strength RDF benchmark for semantic data enrichment based on a real-life semantic publishing use case (the BBC semantic publishing platform).

[1]Linked Data Benchmark Council is EU project FP7-317548 (see `http://ldbc.eu`). Renzo Angles was funded by Fondecyt Chile grant 11100364.

In this paper we also describe a process for developing benchmarks based on technical challenges called "choke points", developed by LDBC. This methodology depends on a combination of workload input by end users, and access to true technical experts in the architecture of the systems being benchmarked. The overall goal of the choke-point based approach is to ensure that a benchmark workload covers a spectrum of technical challenges, forcing systems onto a path of technological innovation in order to score good results.

2. ORGANIZATIONAL STRUCTURE

The LDBC is organized in three platforms:

The board of LDBC members. This is formed by one representative (director) per each member organization of LDBC with voice and one single vote per organization in their assembly. Meetings of the directors make all policy decisions, though certain decisions, among which the adoption of new benchmarks, is handled through a written vote and requires an absolute majority of all members.

Technical User Community (TUC)[2]. It is an open organization that brings together users of RDF and graph technologies, researchers, industry participants, and delegates of the LDBC members in physical events (TUC meetings) to discuss about possible benchmark use cases and scenarios and to assess the quality of the benchmark proposals and the adequacy to their needs. LDBC organizes multiple TUC meetings per year, providing logistics and travel assistance to external invitees.

Task Forces. A Task Force is an internal LDBC structure that carries the development of a benchmark from beginning to end. It is formed by experts from member organizations of LDBC, and it works on the proposal, creation of the use case and scenario based on the TUC discussions, choke points suggested by technical experts and the implementation of the different parts of the benchmark (e.g. data and workload generation).

3. DEVELOPMENT METHODOLOGY

The development of a benchmark in LDBC is initiated by board decision, designed and constructed by a task force, and supported by TUC input and feedback. This process results in the creation of four main elements: (1) the data schema, which defines the structure of the data used by the benchmark; (2) the workload, which defines the set of operations that the system under benchmarking has to perform during the benchmark execution; (3) performance metrics, which are used to measure (quantitatively) the performance of the systems; and (4) execution rules, which are defined to assure that the results from different executions of the benchmark are valid and comparable.

The final result, a benchmark specification, consists of both textual documentation and - insofar possible - a standard implementation (i.e. data generator, workload generator and test driver). LDBC software is open source, and is disseminated through GitHub (see `https://github.com/ldbc`).

Choke Point based Design. Literature until now has described the technical work required when designing a good database system benchmark in relatively vague terms. LDBC intends to formalize some of the best practices and raise the state-of-the-art in this area, in its guidelines for benchmark development.

On the surface, a benchmark models a particular scenario, and this should be believable, in the sense that users of the benchmark must be able to understand the scenario and believe that this use-case matches a larger class of use cases appearing in practice. On a deeper – technical – level, however, a benchmark exposes technology to a workload. Here, a benchmark is valuable if its workload stresses important technical functionality of actual systems. This stress on elements of particular technical functionality we call "choke points". To understand benchmarks on this technical level, intimate knowledge of actual system architectures is needed. The LDBC consortium was set up to gain access to those architects of the initial LDBC industry members, as well as to the architects of database systems Dex[3], Neo4j[4], Virtuoso[5], RDF-3X[6], Hyper[7], MonetDB[8] and Vectorwise[9].

LDBC authors [1] analyzed the relational TPC-H benchmark in terms of 28 different choke points; providing both a good illustration of the choke point concept, and an interesting to-do list for those optimizing a system for TPC-H. Specific examples among those 28 are choke points like exploiting functional dependencies in group-by, foreign-key joins with a low match ratio (to be exploited by e.g. bloom filters), and discovering correlation among key attributes in a clustered index (e.g. using zone maps).

Choke points can be an important design ele-

[2] `http://www.ldbc.eu:8090/display/TUC/`

[3] `http://www.sparsity-technologies.com`
[4] `http://www.neo4j.org`
[5] `http://virtuoso.openlinksw.com`
[6] `https://www.mpi-inf.mpg.de/~neumann/rdf3x/`
[7] `http://hyper-db.de`
[8] `http://www.monetdb.org`
[9] http://www.actian.com/vectorwise

ment during benchmark definition. The technical experts in a task force identify choke points relevant for a scenario, and document these explicitly. Subsequently, as the benchmark workload evolves during the process of its definition, a close watch is kept on which queries in the workload test which choke point to which extent, aiming for complete coverage using a limited amount of queries. Choke points thus can ensure that existent techniques are present in a system, but can also be used to reward future systems that improve performance on still open technical challenges.

3.1 Phases of the design process

Analysis. This phase is oriented to determine the requirements of the benchmark based on the analysis of the application domain, workload characterization, and selection and definition of choke points. The workload characterization abstracts the selected real-life scenario into a basic data schema, identifying the typical data structures found and their relationships. Workload requirements are derived from captured static and dynamic behavior of real workloads (e.g. obtained from members or through the TUC). The identification of choke points comes from the architectural analysis in existing systems from expert knowledge.

Design. In this phase, a detailed schema is formulated, including design of attribute value distributions and correlations, and join connectivity between different schematic elements. Also, the workload is fleshed out into concrete sets of queries with example parameter bindings. A check is made on which of the queries in the workload hit which choke points, ensuring that all are covered. Finally, we define the metrics for measuring performance, and rules for benchmark execution and result reporting.

Implementation. In this phase, the needed software tools are designed and implemented, particularly the data generator, the workload generator, and drivers for one or more systems. Data generators must conform to certain minimum standards, for instance, in case of benchmarks at scale, these should be designed with parallelism in mind. Special properties and relationships in the data must be specified and implemented (e.g. data consistency, data distributions and correlations).

The workload generator needs to chose substitution parameters for the operations of the workload. Proper selection of substitution parameters is steered by the data distributions and their correlations as generated by the test driver, and by the choke points behind the individual queries in the workload, and may require post-generation dataset analysis. An important point is that different substitution parameters for one query should always lead to (roughly) the same data access and execution characteristics (in terms of e.g. cardinalities, locality and optimal query plans) such that behavior is stable and understandable.

Testing. With the ability to run the workload on one or more existing systems comes the task of testing it. A basic aim is *verification* to ensure that the implementation yields the correct and intended results. A second aim is *validation* by measuring not only the performance, but all relevant quantitative and qualitative features of the benchmark (cardinalities, query plans, operators used, detailed performance profiles). This experimentation thus provides insight in how far the benchmark indeed tests the choke points that were targeted.

Considering that the data generator, the workload generator and the methods for substitution parameter generation are mutually dependent, benchmark development is by necessity iterative, such that we may fall back to the Design stage to refine these aspects.

Distribution. This phase is oriented to prepare the benchmark for wider usage. It consists of cleaning up the design and implementation to include only the relevant pieces of software. It includes all the operations to package the test drivers, datasets and/or data generators as well as documentation (including execution, auditing and reporting rules).

4. ONGOING DEVELOPMENT

4.1 Social Network Benchmark

The Social Network Benchmark (SNB)[10] is designed for evaluating a broad range of technologies for tackling graph data management workloads. The systems targeted are quite broad: from graph, RDF, and relational database systems to Pregel-like graph programming frameworks.

The scenario of the benchmark, a social network, is chosen with the following goals in mind: it should be understandable to a large audience, and this audience should also understand the relevance of managing such data; the scenario in the benchmark should cover the complete range of interesting challenges, according to the benchmark scope; and the query challenges in it should be realistic in the sense that, though synthetic, similar data and workloads are encountered in practice.

[10] `http://www.ldbc.eu:8090/display/TUC/Social+network+benchmark+task+force`

SNB includes a data generator[11] that enables the creation of synthetic social network data with the following characteristics: the data schema is representative of a real social network (see Figure 1); the data generated includes properties occurring in real data, e.g. irregular structure, structure/value correlations and power-law distributions; and the software generator is easy-to-use, configurable and scalable.

The requirement to generate at scale a complex social graph with special data distributions that at the same time exhibits certain interesting value correlations (e.g. German people having predominantly German names) and structural correlations (e.g. friends being mostly people living near), poses an interesting challenge. The SNB data generator builds on the work on correlated social network generation in S3G2 [6], whose source code has been adapted to the SNB data schema. S3G2 comes with the ability to leverage parallelism through Hadoop, ensuring fast and scalable generation of huge datasets.

SNB is intended to cover all main aspects of social network data management, and therefore splits into three separate workloads:

– **Interactive workload.** This workload tests system throughput with relatively simple queries and concurrent updates. The workloads test ACID features and scalability in an online operational setting. Given the high write intensity, this workload may also be used to let the dataset grow, which will be implemented by pre-generating data in the generator but only importing the data corresponding to one time point in the bulk load, and playing out the rest of the modifications in the update workload.

– **Business intelligence workload.** This workload consists of complex structured queries for analyzing online behavior of users for marketing purposes. The workload stresses query execution and optimization. The targeted systems are expected to be those that offer an abstract query language. Queries typically touch a large fraction of the data and do not require repeatable read.

– **Graph Analytics Workload.** This workload tests the functionality and scalability of the systems for graph analytics that typically cannot be expressed in a query language. The analytics is done on most of the data in the graph as a single operation and produces large intermediate results. The analysis is not expected to be transactional or need isolation. This workload targets graph programming frameworks, though systems with a query-language might compete using iterative implementations that repeatedly fire queries and keep intermediate results in temporary data structures.

A benchmarked system does not need to run all workloads. Each workload in SNB produces a single metric for performance at the given scale and a price/performance metric at the scale.

4.2 Semantic Publishing Benchmark

The Semantic Publishing Benchmark (SPB)[12] simulates the management and consumption of RDF metadata that describes media assets, or creative works. The scenario is a media organization that maintains RDF descriptions of its catalogue of creative works (e.g. from the BBC). The benchmark is designed to reflect a scenario where a large number of aggregation agents provide the heavy query workload, while at the same time a steady stream of creative work description management operations are in progress. This benchmark plainly targets RDF database systems, which support at least basic forms of semantic inference.

The RDF descriptions of this benchmark use an ontology that defines numerous properties for content, for example: date of creation, short/long descriptions, etc. Furthermore, a tagging ontology is used to connect individual creative work descriptions to instances from reference datasets, including sports, geographical, or political information. The data used will fall under the following categories: reference data, which is a combination of several Linked Open Data datasets, e.g. GeoNames and DBpedia; domain ontologies, that are specialist ontologies used to describe certain areas of expertise of the publishing, e.g., sport and education; publication asset ontologies, that describe the structure and form of the assets that are published, e.g., news stories, photos, video, audio, etc.; and tagging ontologies and the metadata, that links assets with reference/domain ontologies.

The data generator is initialized by using several ontologies and datasets. The instance data collected from these datasets are then used at several points during the execution of the benchmark. Data generation is performed by generating SPARQL fragments for create operations on creative works and executing them against the RDF database system.

Two separate workloads are modeled in SPB:

– **Editorial workload.** It simulates creating, updating and deleting creative work metadata descriptions. Media companies use both manual and semi-automated processes for efficiently and correctly managing asset descriptions, as well as annotating them

[11] https://github.com/ldbc/ldbc_socialnet_bm/tree/master/ldbc_socialnet_dbgen

[12] http://www.ldbc.eu:8090/display/TUC/Semantic+Publishing+Task+Force

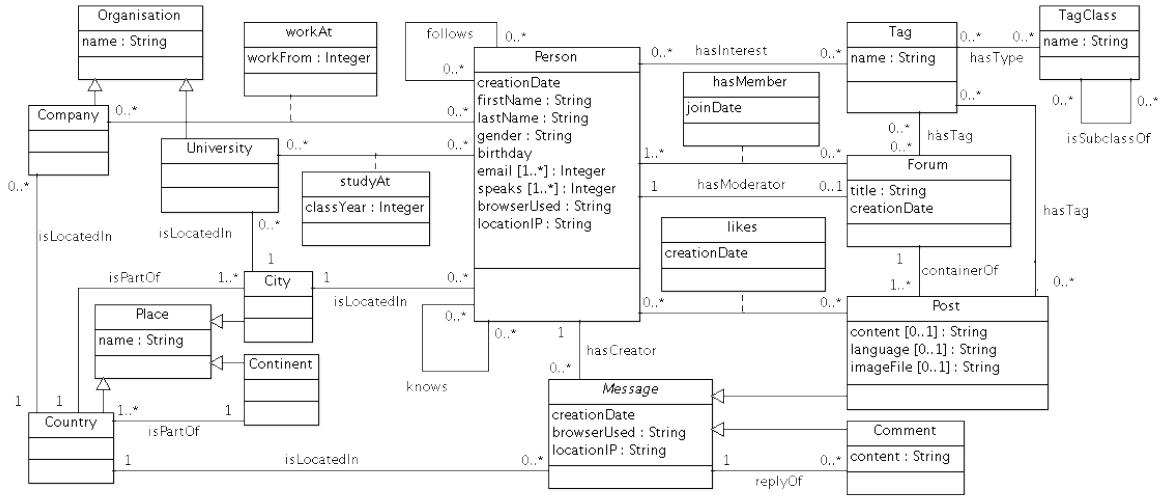


Figure 1: The data schema of the Social Network Benchmark represented in UML.

with relevant instances from reference ontologies.

– **Aggregation workload.** It simulates the dynamic aggregation of content for consumption by the distribution pipelines (e.g. a web-site). The publishing activity is described as "dynamic", because the content is not manually selected and arranged on, say, a web page. Instead, templates for pages are defined and the content is selected when a consumer accesses the page. In this workload, SPARQL queries are used to find relevant content.

Measurement in SPB is performed on the update/retrieve rate of queries executed by editorial and aggregation agents for a fixed amount of time. Metrics describe the queries per second rate that each RDF database system is capable to sustain during the benchmarking period.

5. CONCLUSIONS

In this paper we presented the Linked Data Benchmark Council (LDBC), a new initiative towards for benchmarking Graph and RDF data management systems. LDBC aims to bring some of the best practices of the TPC to the small but growing graph and RDF database industry. A main technical advance is its "choke point" driven benchmark design, which ensures that interesting and well-chosen technical challenges will emerge from implementing the benchmarks. The LDBC currently has two benchmarks under development by its "task forces": the Social Network Benchmark (SNB) and the Semantic Publishing Benchmark (SPB). The latter focuses on testing RDF database systems, whereas the former actually splits into three different sub-benchmarks (the interactive workload, the BI workload, and the graph analytics workload) that all work on a shared dataset. This benchmark thus targets graph, RDF and relational database systems, as well as graph programming frameworks.

We hereby invite the reader to join the Technical User Community (TUC) to influence the LDBC.

6. REFERENCES

[1] P. Boncz, T. Neumann, and O. Erling. TPC-H Analyzed: Hidden Messages and Lessons Learned from an Influential Benchmark. In *TPCTC*, 2013.

[2] D. Dominguez-Sal, N. Martinez-Bazan, V. Muntes-Mulero, P. Baleta, and J. L. Larriba-Pey. A Discussion on the Design of Graph Database Benchmarks. In *TPCTC*, 2010.

[3] S. Duan, A. Kementsietsidis, K. Srinivas, and O. Udrea. Apples and Oranges: A Comparison of RDF Benchmarks and Real RDF Datasets. In *SIGMOD*. ACM Press, 2011.

[4] Y. Guo, A. Qasem, Z. Pan, and J. Heflin. A Requirements Driven Framework for Benchmarking Semantic Web Knowledge Base Systems. *TKDE*, 19(2):297–309, 2007.

[5] K. Huppler. The Art of Building a Good Benchmark. In *TPCTC*, August 2009.

[6] M.-D. Pham, P. A. Boncz, and O. Erling. S3G2: A Scalable Structure-Correlated Social Graph Generator. In *TPCTC*, 2012.

[7] T. Weithöner, T. Liebig, M. Luther, and S. Böhm. What's Wrong with OWL Benchmarks. In *SSWS*, 2006.

MITRA: Byzantine Fault-Tolerant Middleware for Transaction Processing on Replicated Databases

Aldelir Fernando Luiz
Federal University of Santa Catarina – Brazil
aldelir.luiz@posgrad.ufsc.br

Lau Cheuk Lung
Federal University of Santa Catarina – Brazil
lau.lung@ufsc.br

Miguel Correia
University of Lisboa – IST, INESC-ID – Portugal
miguel.p.correia@tecnico.ulisboa.pt

ABSTRACT

Replication is often considered a cost-effective solution for building dependable systems with off-the-shelf hardware. Replication software is usually designed to tolerate crash faults, but Byzantine (or arbitrary) faults such as software bugs are well-known to affect transactional database management systems (DBMSs) as many other classes of software. Despite the maturity of replication technology, Byzantine fault-tolerant replication of databases remains a challenging problem. The paper presents MITRA, a middleware for replicating DBMSs and making them tolerant to Byzantine faults. MITRA is designed to offer transparent replication of off-the-shelf DBMSs with replicas from different vendors.

1. INTRODUCTION

For many years, relational database management systems, often called simply *database management systems* (DBMSs), have been a key component of applications of a wide-range of business areas. We believe this will continue to be true for many applications for years ahead. Applications based on DBMSs usually rely on the reliability of transaction processing and on the availability of the data stored in the database. Therefore, fault tolerance is a desirable property for DBMSs.

Replication is a well-known approach to make services fault-tolerant, which has already been applied to DBMSs [8, 18]. The idea is that the service is executed in a set of servers in such a way that if some of them fail, the service as a whole stays operational and clients continue to be able to execute transactions. However, DBMS vendors usually do not provide native support for replication or hooks for third-party replication protocols. This puts on third-parties the burden of either modifying DBMS source code (if available) or to develop middleware that intercepts client requests and delivers them to the servers. The latter is the approach followed in this paper.

The replication of databases has been studied both in the databases and distributed systems research communities. Although Gray commented that it can be hard to achieve strong consistency in replicated databases [8], Schiper and Raynal have shown that transactions on replicated databases have common properties with group communication primitives such as ordering and atomicity [15]. After that result, several researchers studied the use of group communication systems and middleware to support database replication [12, 4, 10, 16, 6, 13]. Most of the solutions for database replication tolerate only crash faults [12, 4, 10]. Although these are arguably the most common faults, Byzantine or arbitrary faults are also common in today's systems. Faults such as data corruption in disk or RAM due to physical effects or in software due to bugs are Byzantine faults, not crashes. Interestingly, many bugs have historically been found in DBMSs [7].

In the literature there are a few proposals of Byzantine fault-tolerant (BFT) protocols to replicate databases [7, 16, 6, 13]. However, they are focused on a specific problem, based on assumptions hard to substantiate in practice, or simply not the best for certain applications. Specifically, [7] does not allow concurrent transactions; HRDB [16] depends on a centralized controller; Byzantium [6] adopts a consistency criterion that may cause anomalies violating the semantics of some applications (snapshot isolation); and BFT-DUR [13] does neither handle relational databases nor Byzantine clients.

This paper presents the design of a middleware for BFT database replication, MITRA (Middleware for Interoperability on TRAnsactional replicated databases). MITRA supports design diversity [14], i.e., different replicas can run different DBMSs. This is an important mechanism to avoid common mode failures caused, e.g., by a bug existing in all replicas. The middleware supports concurrent transactions, has no centralized components, and provides serializability. The paper does not delve into the details of the protocol at the core of the middleware, which has already been presented elsewhere [11], but on its

design and architectural aspects.

MITRA is modular in the sense that it does not require changes on the DBMSs and it encapsulates all complexity of the BFT replication. The middleware is written in Java and modularity is achieved by following the Java Database Connectivity (JDBC) specification. The rest of this paper is organized as follows: Section 2 discusses some concepts on database replication; Section 3 introduces MITRA and its design; in Section 4, we describe some implementation details and an experimental evaluation; Section 5 concludes the paper.

2. DATABASE REPLICATION

There are several taxonomies of database replication schemes in the literature [8, 17]. A particularly interesting one classifies protocols in terms of their update propagation strategy, i.e., of the way in which they make the state of the replicas converge. This taxonomy classifies database replication schemes in three classes: synchronous (or eager replication), asynchronous (or lazy replication), and certification-based.

A *synchronous* or *eager replication* protocol propagates the updates of a transaction by applying them on the replicas before the transaction commits [8]. More specifically, such a scheme provides strong consistency and fault tolerance by ensuring that updates are stable at multiple replicas before replying to the clients.

An *asynchronous* or *lazy replication* protocol executes and commits each transaction at a single replica, delaying the propagation of updates until the transaction has committed [8]. Lazy replication tends to perform better than eager replication because it avoids the communication overhead of the later during normal execution. However, lazy replication can let replicas diverge and lose the effects of some transactions [18].

A *certification-based* protocol uses group communication primitives such as total order multicast for ordering transactions and propagating write and read sets to the group of replicas [12, 18]. This approach is optimistic since reads and updates are first executed on a single replica without any synchronization with the rest, being a transaction committed only if there are no conflicting updates. MITRA follows this approach since it has shown good performance in the past [12, 10].

3. THE MITRA MIDDLEWARE

As explained, MITRA is a middleware that aims to tolerate Byzantine faults on database systems following the certification-based replication approach. It encapsulates the fault tolerance mechanisms by implementing the JDBC API specification, in order to provide a heterogeneous and replicated environment that appears to the clients as a single virtual DBMS. The ability to support DBMS diversity is important because different systems are unlikely to share the same bugs or vulnerabilities [7].

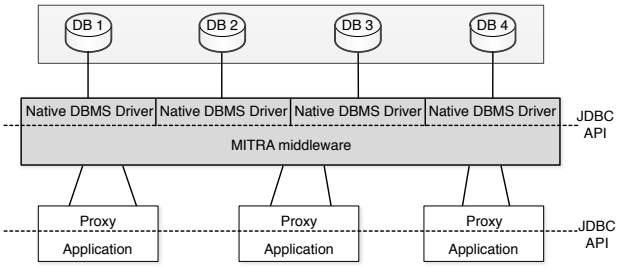


Figure 1: Basic architecture of MITRA

Figure 1 presents the basic architecture of MITRA. The client application interacts with the database through a proxy that exports the JDBC API and replaces what would normally be a DBMS-specific JDBC driver (e.g., a MySQL or a PostgreSQL JDBC driver). The middleware is implemented essentially on the server-side, meaning that the protocol is mostly executed by the servers where the database is replicated. The figure represents it abstractly in the form of a mid-layer, but there are server-side replication processes running in all the servers and communicating through the network. The middleware at each server makes calls to a DBMS driver – again a JDBC driver – that hides the specifics of the DBMS from the replication process. These drivers are readily available for a wide range of DBMSs (http://devapp.sun.com/product/jdbc/drivers). The use of different DBMSs in different servers provides diversity.

3.1 Assumptions

We consider a system composed of an arbitrary, finite, set of clients $C = \{c_1, c_2, ..., c_n\}$ and a set of n replicas $S = \{r_1, r_2, ..., r_n\}$. These entities communicate by message passing, through the network. We assume that an unlimited number of clients and up to $f = \lfloor \frac{n-1}{3} \rfloor$ replicas can be faulty, i.e., can deviate arbitrarily from their specification (Byzantine faults).

Let a database be a collection of data items $D = \{x_1, x_2, ..., x_n\}$. A transaction is a sequence of read and/or write operations on these data items, initiating by a *begin transaction* operation and ending with a *commit* or an *abort* operation. To support certification-based database replication, we make some assumptions about the replica DBMSs: (i) they are relational and transactional; (ii) they support rollback of operations; (iii) they implement

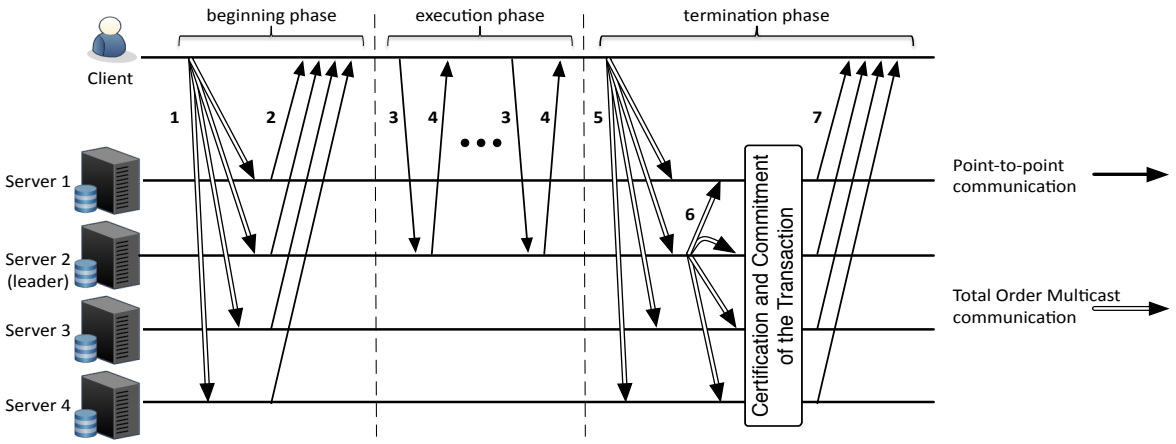


Figure 2: MITRA's replication protocol with its three phases.

strict two-phase locking and serializable isolation; (*iv*) statements modify the database atomically, without side effects.

As mentioned, MITRA is based on a group communication primitive. Specifically, MITRA uses a BFT *total order multicast* primitive based on a consensus protocol [3]. We make a weak assumption about the synchrony of the system to ensure the termination of this primitive: communication delays do not grow exponentially. This is required by the impossibility of solving consensus deterministically in asynchronous systems [5]. We also assume the existence of a collision-resistant cryptographic hash function and a message authentication function to ensure the integrity and authenticity of messages.

3.2 Replication Protocol

The replication strategy adopted in MITRA is an extension of the original certification-based replication scheme [18] to handle Byzantine faults. In this sense, MITRA operates by letting transactions execute first on a single replica in an optimistic way; when the termination of the transaction is requested, that replica total order multicasts the transaction's reads and writes to the rest of the replicas for certification and commitment. The use of this multicast primitive guarantees that all replicas execute these operations in the same order despite the existence of concurrency [1]. This section briefly presents the protocol (details in [11]).

The execution of MITRA's protocol for a single transaction is represented in the time diagram of Figure 2. For each transaction, a replica is selected to be the leader and the rest are followers. The protocol has three phases that we explain next: beginning, execution, and commitment.

Beginning. A transaction begins with a client opening a connection to the database. The client-side part of the middleware sends a *begin transaction* message to all replicas using the total order multicast primitive (step 1). Upon delivering this message, every replica applies a deterministic criterion to select a single replica to be the leader and execute the transaction's operations in a speculative way. Then, every replica sends an acknowledgement to the client with the identifiers of the leader and the transaction, concluding this phase (step 2).

Execution. Figure 2 illustrates this phase in steps 3 and 4, which are repeated for all the statements of the transaction. On receiving the begining-phase acknowledgment from at least $f + 1$ replicas, the client becomes aware of which replica is the transaction's leader. The client connects to the leader and submits its read and write operations to that replica (step 3). After receiving a statement, the leader executes it and returns the result to the client (step 4). Note that during this phase, there is no interaction among replicas since our protocol relies on optimistic concurrency control [9]; the other replicas do not even receive the statements. This phase ends when the client requests the commitment of the transaction.

Termination. To request commitment of a transaction, the client totally order multicasts a *request commit* message to all replicas (step 5). This message carries the statements issued in the transaction and a hash of the results received by the client. Upon delivering the *request commit* message, every replica changes the transaction to a state indicating that it is ready to commit. Then, the leader totally order multicasts to all replicas a message with the following data (step 6): a hash of the transaction's statements received and processed; a hash of the transaction's state; the transaction's readset and writeset. Upon delivering the *commit* message, all replicas verify the data in the *request commit*

and *commit* messages. If they match, every replica starts a *certification* test where it checks the validity of transaction's readset in the database. These checks are needed to guarantee the transaction integrity and validity, and to preserve the serializability. These checks also prevent a Byzantine client from either forging a transaction or committing a spurious transaction. If any of the checks fails, the transaction is aborted.

If the transaction passes these checks and there is a concurrent transaction (or more), the termination still involves another *certification* step. Consider two transactions, T_i and T_j. We say that T_i precedes T_j if the *commit* message for T_i is delivered by the replicas before they deliver the *request commit* message for T_j. We say that these transactions are concurrent if neither T_i precedes T_j nor T_j precedes T_i. The transaction T_i being committed passes the certification test iff for any concurrent transaction T_j, $readset(T_i) \cap writeset(T_j) = \emptyset$. When the transaction passes this test, we can be assured that it does not violate serializability, so its writes are applied in the databases at all replicas. The certifications test is deterministic so every replica will reach the same outcome for T_i.

The protocol and the transaction end when all non-faulty replicas reply to the client with the outcome of the transaction (step 7). The client accepts the outcome if it receives the same reply from at least $f+1$ distinct replicas. No more than f replicas are faulty, so this avoids that the client accept the outcome of a faulty replica that unilaterally commits a non-serializable transaction.

A final remark. In a serializable execution where T_j is executed before a concurrent transaction T_i, T_i would see all of T_j's writes. Our approach is conservative in the sense that T_i and T_j could be executed optimistically, concurrently, in different replicas, but the certification test allows T_i to commit only if T_i did not read any item written/updated by T_j (a read-write conflict and dirty read).

3.3 Middleware Components

Figure 3 presents a detailed architecture of the server-side of the middleware. Recall that the middleware runs at each server, so the representation of the middleware as a single box is an abstraction of reality. The client-side is not detailed as it is much simpler, e.g., it encapsulates calls to the database as messages to the replication protocol.

Client Connection Manager. The *client connection manager* is the interface with the clients, i.e., it is the component that receives messages from the clients and forwards them messages. It works at the

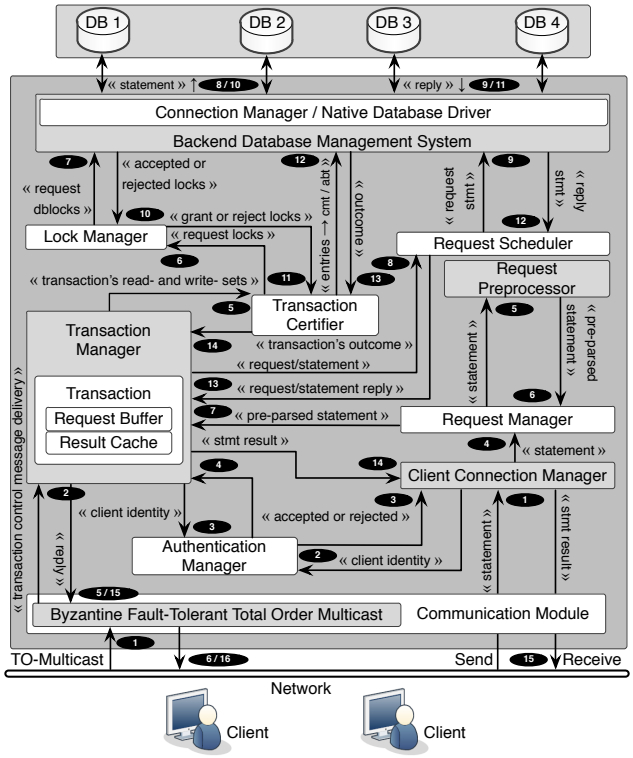


Figure 3: MITRA's components.

level of communication abstractions such as sockets and channels. Upon receiving a client's message it forwards it to the *authentication manager*, to verify if its content is valid and belongs to some active transaction. If that is the case, it forwards the message to the *request manager* for proper treatment; otherwise, it discards the message and sends an exception to the client.

Request Manager. The *request manager* manages all requests received by the server. When a statement is received, the module does basic syntax checking to verify if it is well-formed, then invokes the next module to preprocess the statement. Then, it invokes the *transaction manager* to link the statement with its transaction, as there can be several concurrent transactions being executed through the middleware, and process it.

Request Preprocessor. This component is responsible for the just mentioned syntax checking and preprocessing. MITRA has to support diversity of DBMSs, so there are compatibility issues that must be solved. The middleware supports the standard way of interaction with databases using SQL (Structured Query Language), so the use of different dialects of this language is a problem when diversity is needed. A solution would be to restrict statements to ANSI SQL, but our experience shows that this is too limitative. Therefore, our middleware solves

this problem by translating the SQL statements issued by the clients into the native SQL dialect of the back-end database replica running at the server. This translation is a complex task, but there are software packages that are able to do it for most SQL dialects, e.g., the SwisSQL API (http://www.swissql.com/products/sql-translator/sql-converter.html) or the SQL-Translator (http://search.cpan.org/~frew/SQL-Translator-0.11018).

Transaction Manager. This module controls transaction execution and keeps data concerning active and committed transactions. This module is responsible for: (*i*) scheduling the statements' executions in the database (by invoking the *request scheduler*); and (*ii*) executing the actions requested by control messages. When the total order multicast protocol delivers a control message, this message is passed to the transaction manager that does the following: *begin transaction* – starts a new transaction on the database; *request commit* – changes the state of the transaction and sends a *commit* message if it is the leader; *commit* – starts the transaction certification test in order to either commit or abort the transaction.

Transaction Certifier. The *transaction certifier* executes two tasks: (*i*) checks the validity of every data item read by a transaction; and (*ii*) checks whether the data items read in an optimistic way are up to date when commit is requested. These tasks are done during the termination phase of the protocol to ensure consistency and serializability. In this way, the *transaction certifier* guarantees that a transaction only commits if it is consistent, it has read valid data items, and its reads do not conflict with writes of any concurrent transaction already committed (see Section 3.2). Note that both tasks are necessary because a Byzantine replica may send spurious data items to the client, or may reply to statements with obsolete data. It is noteworthy that due to the use of total order multicast all replicas deliver the *commit* message to the same transaction in the same order, thus they can certify and commit the transaction in the same way. Lastly, since the *transaction certifier* does transaction certification and commitment in a serialized way, this ensures determinism and that the replica's states do not diverge.

Lock Manager. The *lock manager* is responsible for acquiring the read and write locks on the data items of a given transaction against the database, according to the read and write sets for that transaction. This module acts as a scheduler for requesting the locks in the database. All locks for a transaction are requested and acquired in the same order in all replicas, since this happens when these replicas deliver the *commit* message for that transaction. In this way, the order in which all replicas do it is the order imposed by the total order multicast protocol.

Request Scheduler. When the leader replica receives a valid read or write statement from a client, it passes that statement to the *request manager*. Next, the statement goes to the *transaction manager* and, finally, to the *request scheduler* that executes it in the local database. The scheduler is responsible for dealing with concurrency control issues in cooperation with the local DBMS. All operations are executed synchronously, in the sense that the scheduler waits for a result from the DBMS to send it to the client (by replying to the *transaction manager* that then sends it to the *client connection manager*). The scheduler waits for a response for a given interval of time and returns an exception to the client if that time expires without receiving it.

Authentication Manager. The authentication manager maps the authentication credentials provided by the user – login and password in the current version – with the credentials used to access the local DBMS in the server. The login/password provided by the user are not the ones used in the local databases, but a form of access to the virtual database provided by the middleware. This module also verifies the authenticity of the messages received from the clients.

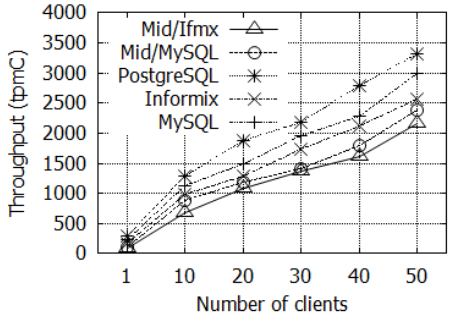
4. PROTOTYPE AND EVALUATION

We implemented the MITRA prototype fully in Java, as a replicated middleware on top of database replicas. As we said in Section 3, our implementation provides a standard JDBC interface, in order to be transparent to client applications. MITRA's JDBC driver is the client-side part of the middleware. It encapsulates the database requests in protocol messages that it forwards to the server-side part of the middleware, the replicas. The replicas access the databases through their native Type IV JDBC drivers. As BFT total order multicast protocol we used BFT-SMaRt, a stable and efficient BFT replication library written in Java (http://code.google.com/p/bft-smart) [3]. Although the code was implemented carefully, at this stage no attempt was made to make it efficient to the point of being usable in real systems.

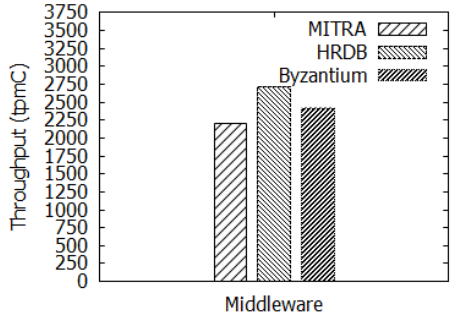
We present some experimental results of the execution of MITRA. Other evaluation aspects such as message complexity can be found in [11]. We evaluate MITRA with the industry standard Online

Transaction Processing TPC-C benchmark (http://www.tpc.org/tpcc). In our evaluation, we compare MITRA, HRDB (with CBS configuration), Byzantium (our own implementation with single-master configuration) and standalone DBMSs. We did not run BFT-DUR because it relies on a key/value database, and the TPC benchmarks are not compatible with this type of databases.

All experiments were done in a LAN with nine machines running CentOS 5.8 and IBM's JDK 6.0, each with 4GB of RAM, an Intel Core 2 Duo E8400 2.67GHz CPU and an Intel 82567LM-3 Gigabit Ethernet interface. The replicas and the clients were attached to the same network switch (Gigabit Ethernet). In order to have database diversity, we used MySQL 5.5.8 (InnoDB) and Informix Innovator-C Edition 11.70-UC1 with MITRA and PostgreSQL 8.4 with Byzantium. For the TPC-C experiments, we loaded the databases (every replica) with 10 warehouses and 10 districts per warehouse. We consider only the case of $f = 1$, which is the typical value used in the evaluation of BFT protocols, as replicas are expensive. Therefore, we executed both MITRA and Byzantium with 4 replicas, and HRDB with 3. The remaining 5 machines were used to run up 50 clients issuing transactions with an interval of 200 ms. The values reported in Figure 4 are the averages of 25 experiments.



(a) Throughput without replication.



(b) Throughput with replication.

Figure 4: Experimental results for standard TPC-C workload (with no batches).

Figure 4(a) aims to show the overhead introduced by the middleware without replication. The label Mid/MySQL corresponds to MITRA with MySQL as DBMS and Mid/Ifmx to MITRA with Informix. In both cases, the clients interact with a single database server through MITRA. These two configurations are not fault-tolerant so they aim simply to provide a lower bound on the performance that our prototype can achieve. The experiments labeled MySQL, Informix and PostgreSQL were obtained through the native JDBC drivers for these DBMSs without involving the middleware. A brief comparison of MITRA's JDBC driver and native DBMSs JDBC drivers shows that the overhead introduced by the middleware is at most 35%, depending on the DBMS. We believe this overhead can be reduced by better engineering the source code, but at this stage our objective was only to have a proof-of-concept prototype.

Figure 4(b) presents the results of running TPC-C with MITRA, HRDB, and our own implementation of Byzantium, with 50 clients producing transactions. In these experiments, both MITRA and HRDB used MySQL as DBMS, while Byzantium used PostgreSQL. We used different DMBSs for MITRA and Byzantium because they have different requirements. MITRA needs DBMSs that supports serializable isolation, whereas Byzantium needs a DBMS that supports snapshot isolation. The figure shows that MITRA has a slightly worse performance than the rest, which was expected for the following reasons. HRDB relies on a centralized coordinator, so it does not need to use a multicast between replicas or to run a certification in every replica, reducing much the overhead involved. However, the benefits of MITRA in relation to HRDB are clear: the failure of up to any f servers does not preclude our middleware from continuing to process transactions; in HRDB the failure of the coordinator stops the system. Byzantium scales better for three reasons, all related to snapshot isolation: it needs only write sets to certify transactions; it needs just one atomic multicast to do commit; it does not need to acquire locks on read operations. However, snapshot isolation is weaker than serializability and under certain circumstances, it can produce incorrect results by violating integrity constraints [2], which not happen in MITRA.

5. FINAL REMARKS

The paper presented a flexible middleware for Byzantine fault-tolerant replication of databases using heterogeneous DBMSs. Due to the use of JDBC, MITRA is compatible with applications that use this interface to interact with the database, being

transparent to the DBMSs, except for issues related to dialects of SQL. Although MITRA is not the only solution for BFT database replication, none of the alternatives provides simultaneously serializability and full distribution. The performance of MITRA is slightly worse than others, which was expected due to the characteristics of that protocol and the fact that we did not make a strong effort to make the prototype efficient. Nevertheless, the results are promising and the costs seem to provide an adequate tradeoff with the benefits, at least for some applications. We believe that these overheads are not overly onerous, especially given the increased robustness and fault tolerance that MITRA offers for an application.

Acknowledgments. This work was partially supported by the CAPES through project Lead Clouds (A039_2013) and by the CNPq through PDI 560258/2010-0 and by the FCT through contract PEst-OE/EEI/LA0021/2013 (INESC-ID).

6. REFERENCES

[1] D. Agrawal, G. Alonso, A. E. Abbadi, and I. Stanoi. Exploiting atomic broadcast in replicated databases. In *Proceedings of the 3rd International European Conference on Parallel Processing*, pages 496–503, 1997.

[2] H. Berenson, P. Bernstein, J. Gray, J. Melton, E. O'Neil, and P. O'Neil. A Critique of ANSI SQL Isolation Levels. *ACM SIGMOD Record*, 24(2):1–10, 1995.

[3] A. Bessani, J. Sousa, and E. Alchieri. State Machine Replication for the Masses with BFT-SMaRt. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks*, 2014.

[4] E. Cecchet, J. Marguerite, and W. Zwaenepoel. C-JDBC: Flexible database clustering middleware. In *Proceedings of the USENIX Annual Technical Conference*, pages 9–18, 2004.

[5] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, 1985.

[6] R. Garcia, R. Rodrigues, and N. Preguiça. Efficient middleware for Byzantine fault-tolerant database replication. In *Proceedings of the 6th European Conference on Computer Systems*, pages 107–122, 2011.

[7] I. Gashi, P. T. Popov, and L. Strigini. Fault tolerance via diversity for off-the-shelf products: A study with SQL database servers. *IEEE Transactions on Dependable and Secure Computing*, 4(4):280–294, 2007.

[8] J. Gray, P. Helland, P. O'Neil, and D. Shasha. The dangers of replication and a solution. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 173–182, 1996.

[9] H. T. Kung and J. T. Robinson. On optimistic methods for concurrency control. *ACM Transactions on Database Systems*, 6:213–226, 1981.

[10] Y. Lin, B. Kemme, M. P. no Martínez, and R. Jiménez-Peris. Middleware based data replication providing snapshot isolation. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 419–430, 2005.

[11] A. F. Luiz, L. C. Lung, and M. Correia. Byzantine fault-tolerant transaction processing for replicated databases. In *Proceedings of the 10th IEEE International Symposium on Network Computing and Applications*, pages 83–90, 2011.

[12] F. Pedone, R. Guerraoui, and A. Schiper. The database state machine approach. *Distributed and Parallel Databases*, 14(1):71–98, 2003.

[13] F. Pedone, N. Schiper, and J. Armendáriz-Iñigo. Byzantine fault-tolerant deferred update replication. In *Proceedings of the 5th Latin-American Symposium on Dependable Computing*, pages 7–16, 2011.

[14] B. Randell. System structure for software fault tolerance. *IEEE Transactions on Software Engineering*, 1(2):221–232, 1975.

[15] A. Schiper and M. Raynal. From group communication to transactions in distributed systems. *Communications of the ACM*, 39:84–87, 1996.

[16] B. Vandiver, H. Balakrishnan, B. Liskov, and S. Madden. Tolerating Byzantine faults in transaction processing systems using commit barrier scheduling. In *Proceedings of 21st ACM Symposium on Operating Systems Principles*, pages 59–72, 2007.

[17] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, and G. Alonso. Database replication techniques: a three parameter classification. In *Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems*, pages 206–215, 2000.

[18] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, and G. Alonso. Understanding replication in databases and distributed systems. In *Proceedings of the 20th International Conference on Distributed Computing Systems*, pages 464–474, 2000.

The Database Group at the University of Washington

Magdalena Balazinska, Bill Howe, and Dan Suciu
{magda,billhowe,suciu}@cs.washington.edu
Dept. of Computer Science & Engineering
University of Washington

The database group at the University of Washington (UW) was founded in 1998 when the department hired Alon Halevy (now at Google). The group currently consists of about twenty researchers: three faculty members (the authors), four postdocs, and fifteen students. Alumni include faculty members at Computer Science Departments at British Columbia, Michigan, Pennsylvania, Stanford, UMass, Wisconsin, one faculty member at the CMU Tepper School of Business, and several researchers and engineers at Facebook, Google, Microsoft, Nokia, Twitter, and other technology companies. The group has funding from NSF, the Gordon and Betty Moore Foundation, the Alfred P. Sloan Foundation, and several companies including Amazon, EMC, Google, HP, Intel, Microsoft, NEC, and Yahoo. The group has been recognized through several best paper awards and two ACM SIGMOD Best Dissertation Awards.

We conduct research mostly in small groups and tackle a diverse set of data management challenges. Some of our projects result from collaborations with domain scientists on the UW campus; others are sparked by novel theoretical breakthroughs that lead to new approaches to data management challenges; many are the results of both. We give here a short overview of the recent research themes in our group; more details are available on our website:

`http://db.cs.washington.edu/`

1. SCIENTIFIC DATA MANAGEMENT

Our research agenda is partially derived from collaborations with scientists across the University of Washington and beyond, leveraging our close connection with the University of Washington eScience Institute [6].

The eScience Institute was founded in 2005 with the goal of advancing the research and practice of data-intensive discovery across all fields of science. With the advent of new, high-bandwidth data sources (survey telescopes, high-throughput sequencers, ubiquitous sensor networks, planetary-scale simulations), data management research became recognized as a critical driver of scientific discovery. As a result, the database group and the eScience Institute became close partners, and were able to initiate and maintain multiple long-term collaborations with scientists.

In 2008, we founded an inter-disciplinary research group called AstroDB [1]. This group brings together faculty, research scientists, postdocs, and students from the Astronomy department and our database group. In 2009, we initiated an independent collaboration with a marine microbiology lab. Thanks to the sustained nature of these partnerships, both have led to a series of joint research projects. We give examples in the following sections.

Our inter-disciplinary collaborations have also allowed us to collect a curated repository of datasets and use cases that anyone can use in their research: A repository of MapReduce applications [15], a public repository of scientific datasets equipped with a SQL interface [19], and a number of parallel analytics use cases that go beyond MapReduce [14]. We are continuously working on expanding these collections of applications.

2. BIG DATA SYSTEMS

Motivated by partnerships in both science and industry, we have invested deeply in research on systems that can empower non-specialists to extract knowledge from large, complex, and noisy datasets.

As one approach, we worked on leveraging modern tools such as Dryad and Hadoop (see our Nuage project website for details and papers [15]). As an example, we helped our AstroDB collaborators to develop a method for carrying out an important analysis step, which involved data clustering, using both Dryad and Hadoop. We found, however, that it was far from trivial to use these systems in a way that resulted in high performance: while both Dryad and Hadoop made it easy to express the user-defined functions that we needed, both lead to problems with uneven load distribution, also called *skew*. This observation lead us to build the SkewReduce and SkewTune [13] systems; both are publicly available and have been very well received by sci-

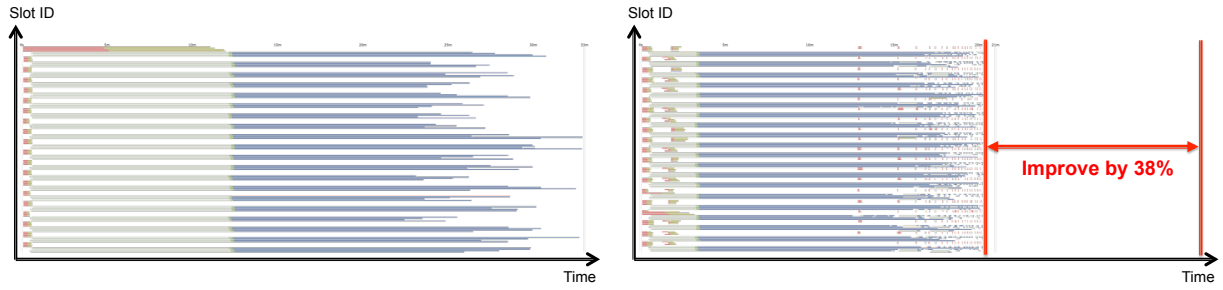


Figure 1: Timing chart illustrating how SkewTune can help squeeze the execution of a MapReduce job. In this example, runtime decreases by 38% with SkewTune.

ence and industry. Figure 1 illustrates the benefits of SkewTune on the execution of a MapReduce job. In that same line of work, we developed progress indicators for MapReduce workflows and a fault-tolerance optimizer. These approaches were designed to facilitate and accelerate data analysis using Hadoop.

Simultaneously, we became interested in extending existing Big Data systems with iterative capabilities to support more general classes of complex analytics tasks. The HaLoop system extended Hadoop with inter-iteration caching that led to orders of magnitude performance improvements for long-running iterative jobs [3]. We followed up that work by showing how to implement and optimize a recursive query language — Datalog — using the HaLoop runtime.

Our work became quickly noted in the community: the HaLoop paper is currently the most-cited paper in VLDB 2010, SkewTune and one of our MapReduce progress estimation papers are among the top cited in SIGMOD 2012 and ICDE 2010[1], while a related paper on the study of skew in MapReduce won the Best Student Paper at the Open Cirrus Summit 2011.

Over the past two years, our group has moved beyond Hadoop. We have built our own, parallel data management system called Myria [14]. An important aspect of our system is that Myria is set up as a Cloud service that users access directly from their browsers. In the Myria project, we are studying *both* the systems challenges related to efficiently supporting modern data management and analytics needs *and* what it means to operate such as system as a Cloud service. Myria now runs on 100-node Amazon EC2 deployments and processes terabytes of data from applications in astronomy, oceanography, social media, and cybersecurity, as well as standard benchmarks. Figure 2 shows a screenshot of the Myria graphical interface.

Myria's goal is to allow users to simply upload their data and become self-sufficient data science experts: "self-serve analytics." Myria accepts queries (online in a

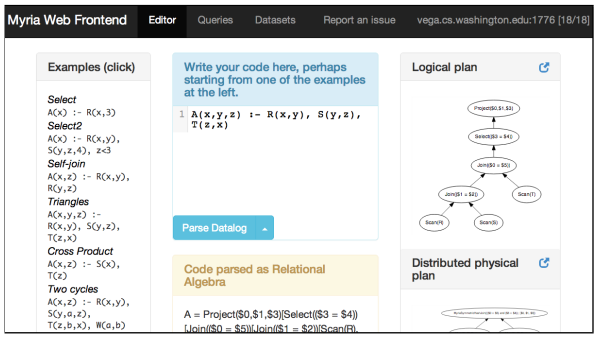


Figure 2: Myria browser-based front-end.

browser or programmtically through a REST API) written in SQL, Datalog, and a new, iterative, hybrid declarative/imperative language we call MyriaL. All three languages are compiled to the same intermediate representation based on an extension of RA+While, then optimized to produce a parallel physical plan for execution on a cluster. Based on our experience with SQLShare [9] (described below), we know that science users can and will write data analysis tasks in declarative languages, but we seek new language features to capture a greater proportion of their tasks. Myria's execution layer, MyriaX, adopts state-of-the-art system design principles: it uses a pipelined, possibly cyclic graph of dataflow operators that make efficient use of I/O and memory, and it has built-in support for asynchronous evaluation of recursive queries. In addition, we are innovating in the space of massively distributed query processing techniques, by building on recent theoretical breakthroughs connecting conjunctive queries with the fractional edge cover (for sequential processing) or fractional edge packing (for parallel processing). The suite a techniques lead to significant reductions in the amount of data communication during the computation of multiway join queries; some of our work in this space is here [2], more is under way. Figure 3 shows the high-level Myria architecture. We will be demonstrating Myria at SIGMOD 2014.

Separately from Myria, we are also working on a

[1] `http://arnetminer.org/conferencebestpapers`

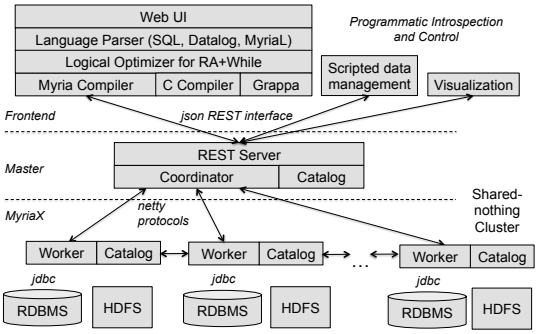


Figure 3: Myria system architecture.

project to manage non-relational data in the context of scientific data management. In astronomy, for example, research involves the analysis of telescope images, such as those produced by the Sloan Digital Sky Survey or the Large Synoptic Survey Telescope. To address the general problem of analyzing multidimensional datasets (from images, simulations, and others) commonly used in science, our group has partnered with the SciDB project [20]. Our focus has been on the storage and execution layers of this engine. In the context of the AstroDB collaboration, we also built the AscotDB system [23] that enables the efficient analysis of raw telescope images using SciDB. AscotDB extends SciDB with efficient iterative processing features, support for spherical coordinates, and an integrated set of graphical and Python interfaces geared toward telescope image analysis.

3. USER-FACING TOOLS

The number of users who need to manage and analyze large databases is growing much faster than the number of database administrators and developers. Users today thus need to be self-sufficient and modern data management systems must support these users effectively. In response to this trend, our group studies how to facilitate each step of the data management and analysis cycle; we briefly describe here four projects that our group conducted in this space.

SQLShare began as an experiment [8] to try and understand why databases tend to be underused in science — despite a natural fit for question-and-answer hypothesis testing. Common explanations claim a mismatch between scientific data and the models and languages of relational databases, or simply that "scientists won't write SQL." Our hypothesis, instead, was that the core models and languages were more than adequate, but it was too much work to set up a database and get the data in and out. To test the hypothesis we developed a Web-based interface to SQL Azure that reduced the use of databases to a simple upload-query-share workflow that emphasized views and view sharing as the first class interaction mode. The SQLShare experiment has been remarkably successful in demonstrating the utility of databases in new contexts; it currently has hundreds of science users who have uploaded several thousand datasets of varying size and complexity and issued tens of thousands of hand-written SQL queries. We have seen collections of scripts written in R and Python replaced with a handful of SQL queries, simplifying collaborative analysis to the exchange of links into SQLShare [9]. We have seen SQLShare used to facilitate open data and complexity hiding: at least one public dataset is a view that joins 50 distinct tables. In one experiment, we participated in a workshop between 40+ oceanographers from various subfields, using SQLShare to perform "SQL stenography": writing queries in real time to integrate data, test hypotheses, and populate visualizations in response to the live scientific discussion. We found that that ability to write queries in real-time significantly improved the productivity of the meeting, changing it from a "planning" meeting to a science meeting [7].

In a different project, we addressed the core challenge in database usability: assisting inexperienced users in formulating complex SQL queries. This has been a challenge for both research and industry from the early years of relational database systems. Several commercial products include visual query building tools, based on the Query-By-Example visual paradigm, but these are tools aimed at helping users formulate *simple* queries. Our project, SnipSuggest [12], helps users formulate *complex* queries, by taking a radically different approach. As a user types a query, SnipSuggest offers recommendations, by suggesting small *SQL snippets*, such as a list of k relevant predicates for the WHERE clause, or a list of UDFs. The key contribution is in selecting relevant recommendations. SnipSuggest produces *context-aware suggestions*: when generating its recommendations, SnipSuggest considers the partial query that the user has typed so far, then draws its recommendations from similar past queries authored by other users, thus leveraging a growing, shared, body of experience. We found this simple idea to have a dramatic impact in practice.

In the context of Cloud service usability, we are currently re-thinking the interface between users and data management services in public Clouds [16]. Today, when a user wants to use a Cloud service, the service asks her to pick a set of resources, such as the number of instances and their specific sizes. When selecting resources, however, users are left wondering: Will the configuration that I pick be fast enough for me? Will I incur any unexpected charges? Will I be able to express the queries that I need (if the service supports one of many existing "SQL-like" languages). To address

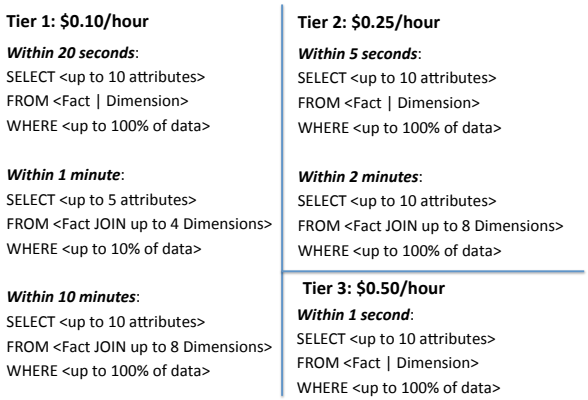


Figure 4: Example PSLA with three service tiers.

this mis-match between the resource-centered view of Cloud services and the need-centered view of the users, we are currently developing a new concept of "Personalized Service Level Agreements (PSLAs)" between users and Clouds. Users specify only the schema of their database and the table sizes, while the system generates a PSLA for their database. The PSLA shows a set of price-performance options for queries expressed on the given data. Figure 4 shows an illustrative example of a toy PSLA. We are actively working on this project and incorporating it in Myria.

Finally, in the RFID Ecosystem project [17], we have explored challenges related to helping users manage sensor data. For this, we put together a building-wide RFID infrastructure with 160 antennas covering 7 floors of the CSE building and 67 users carrying over 300 tags. We developed Cascadia, a complete software stack for managing RFID data [24], built a battery of applications, and conducted longitudinal studies. Through these studies, we were the first to characterize the performance of a user-centered, real-life RFID deployment.

4. PROBABILISTIC DATABASES

Probabilistic databases have emerged as a general abstraction for modeling uncertain data, where the presence of records or the values of attributes may be uncertain [21]. When a query is evaluated on such data, each row returned by the query is annotated with a probability. The key research challenge is to compute these probabilities efficiently, hopefully within current query processing engines. This problem has been shown to be closely related to lifted probabilistic inference in Statistical Relational Models, such as in the popular Markov Logic Networks, also developed at the University of Washington [5]. In *exact inference* one asks for the correct output probabilities, while in *approximate inference* one can tolerate errors.

Over the last decade our group has fully described the complexity of exact inference problem in probabilistic databases. First, for some queries, computing the output probability is #P-hard in the size of the database; the simplest example is:

```
select distinct R.A
from R, S, T
where R.B = S.B and S.C = T.C
```

If each record in `R` and `T` is an independent random variable, with a given probability of being present, then computing the probability of any output tuple is #P-hard in the size of the relations `R`, `S`, `T`. Other queries, however, can be computed in polynomial time (drop the relation `T` in the query above and the new query is in PTIME), and, moreover, they can be computed by a query plan where each query operator manipulates the probabilities explicitly (e.g. a join would multiply the probabilities). The main outcome of our research is a complete classification of Unions of Conjunctive Queries (a.k.a. select/project/union/join queries) into two classes, #P-hard or PTIME, forming a dichotomy [4]. A compiler can decide, for any given query, in which of these two classes the query belongs, through a simple static analysis.

Computing the output probabilities is equivalent to weighted model counting on the query lineage[2], on which there exists a rich literature. An alternative approach to pushing probabilistic inference inside a database engine is to first compute its lineage, then use one of the weighted model counting algorithms. We have proven, however, that there exists queries that are computable in polynomial time, yet classic weighted model counters based on the Davis-Putnam-Logemann-Loveland (DPLL) will take exponential time, even with modern extensions such as caching and components. The results of this research are shown in Figure 5. The query evaluation algorithm that resulted from our research is both more powerful than traditional weighted model counters[3], and has the extra benefit that it can be performed within existing query engines.

Our current research focuses on extending the dichotomy to queries with negation, and approximate inference algorithms for #P-hard queries.

5. CAUSES AND EXPLANATIONS

Our group has conducted pioneering research into understanding causality in data transformation, and computing explanations for observed query outputs. Figure 6 shows the number of publications in SIGMOD during a moving five years window, broken down into papers published by authors from industry and papers

[2]In Statistical Relational Models the lineage is called *grounding*.

[3]Our algorithm uses the inclusion/exclusion formula on the query expression, which has no efficient counterpart on the grounded representation (lineage).

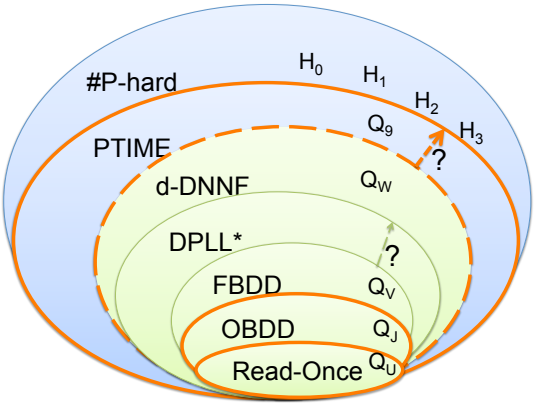


Figure 5: Classification of queries by their complexity on probabilistic databases: some queries are #P-hard, others can be computed in PTIME. Solid lines mean that a complete syntactic characterization of queries in that class is known; dashed line means it is conjectured; thin lines mean it is open. DPLL$*$ means DPLL extended with caching and components.

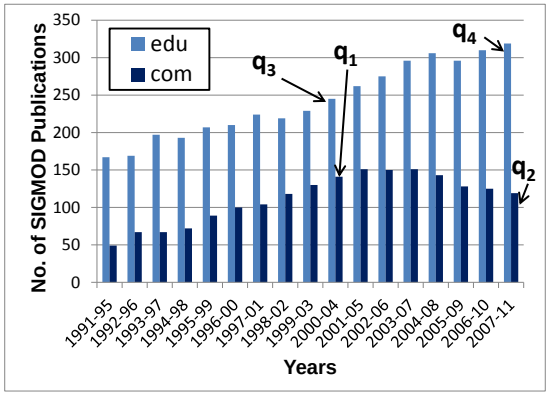


Figure 6: Number of SIGMOD publications in five years windows, broken down into papers from industry ('com') and academia ('edu'). (Affiliation data was available only for some authors, therefore the graph does not include all papers, while papers with authors from both industry and academia are included in both bars.) While both increase until 2000-2007, afterwards the number of papers from academia continues to increase while that from industry decreases. In our research, we study concepts and develop tools that help users find explanations for observed query outputs.

published by authors from academia. Around 2000-2007 one can notice an interesting bump: while before that period both the number of academic and industrial papers increase over time, after that period the number of academic papers continues to increase, but that of industrial papers decreases. Often users face such unexpected query outcomes and would like explanations to their observed outcome.

The golden standard for an explanation is the *actual cause* of the observed outcome, which has been recently studied and defined algorithmically by Judea Pearl. At the core, causality is defined in terms of intervention: an input is said to be a cause if we can affect the output by changing just the value of that input, while keeping all others unchanged. An *explanation* lowers the bar of causality, and only requires that a change in the input affects the output: the more it affects the output, the better the explanation.

Research in our group ranges from theoretical investigations of the complexity of causality, to query explanation systems. PerfXplain, gives explanations for performance observations in MapReduce systems [11]. PerfXplain logs a number of parameters during the execution of MapReduce programs, then allows the user to ask questions such as *why did my job take the same amount of time even though I have doubled the number of workers?*, and returns ranked answers, such as *because the block size was too large* (and, therefore, there was not enough parallelism available). In a more recent system we have extended explanations to arbitrary SQL queries, such as the one that produces the graph in Figure 6: the user asks *why is the graph of SIGMOD publications from academia always increasing, while that from industry is increasing much less?* (we refer the reader to [18] for the answers).

6. MANAGING VALUABLE DATA

We are entering an era where the value of technology shifts from the hardware and software artifacts to the data: the most successful companies are those that hold, can produce, or acquire unique and valuable data, such as geographical data (Google maps), social data (Facebook), corporate data (Dun&Bradstreed), maps (Navteq), or miscellaneous data extracted, integrated, and cleaned (Factual). Data has value, and users are keenly aware of that. Some of our most forward-looking research projects develop revolutionary concepts to reason about and manage data with explicit value. We have pioneered the concept of *arbitrage-free query pricing*. Instead of charging per data item, or for an entire dataset, we have proposed a framework where data sellers can charge for every query, based on the information content of its answer. A key property that must be satisfied by such a framework is that the price function be arbitrage-free: if a query Q_1 can be answered entirely from the output of some other query Q_2, then its price should not be larger (otherwise a middleman would pur-

chase Q_2 and resell Q_1 at a profit). This concept should be quite familiar to the database community: it is precisely when Q_1 can be answered entirely from a materialized view Q_2.

In another project, we have looked at the terms of use that accompany data bought and sold on the Web. When major corporations acquire data, they must enforce that their employees follow these terms, and this is difficult: in an informal survey of 13 data providers we found that the average length of terms of use is about 8.3 pages of language full of legal terms, almost impossible to understand and remember by a typical programmer. We are currently developing a system, called DataLawyer, where such policies can be specified declaratively (in SQL) and which can enforce them automatically at query time, by adding only a small overhead to the query processing time [22].

7. EDUCATION

Our group places a strong emphasis on teaching. At the undergraduate level, we teach a course that focuses on using data management systems and building database applications (CSE 344) and another that focuses on building data management systems (CSE 444).

At the graduate level, funded by an NSF IGERT grant, together with colleagues in Computer Science, Astronomy, Oceanography, Genome Sciences, Statistics, and Chemical Engineering, we are putting in place a new PhD program in Data Science [10]. The goal of the program is to create an inter-disciplinary cohort of PhD students who will all specialize in both methods for data science and at least one application domain. Our graduate database course (CSE544) is one of the required courses in this new program.

Through our partnership with the eScience Institute, we have also been engaged with multiple efforts to develop "data science" curricula for both majors and non-majors. One of us developed a Massively Open Online Course (MOOC) called Introduction to Data Science that saw over 9,000 students complete all assignments and over 7,000 earn a certificate. Unlike many courses in this area, we strongly emphasized the central role of database formalisms and technology in data science, pointing out, for example, the ubiquity of the relational algebra even among "NoSQL" technologies. In addition to this online course, we have bootstrapped certificate programs targeting working professionals in cloud computing and data science, we co-developed a new data-oriented introductory programming course for non-majors called Introduction to Data Programming, and we have co-hosted multiple workshops for introductory programming for scientists and engineers through Software Carpentry[4]).

[4] http://software-carpentry.org/

8. REFERENCES

[1] AstroDB: methods and tools for Big Data astronomy. http://db.cs.washington.edu/astrodb/.

[2] P. Beame, P. Koutris, and D. Suciu. Communication steps for parallel query processing. In *PODS*, pages 273–284, 2013.

[3] Y. Bu, B. Howe, M. Balazinska, and M. Ernst. The HaLoop approach to large-scale iterative data analysis. *VLDB Journal (Special Issue: Best of VLDB 2010)*, 21(2), 2012.

[4] N. N. Dalvi and D. Suciu. The dichotomy of probabilistic inference for unions of conjunctive queries. *J. ACM*, 59(6):30, 2012.

[5] P. Domingos and D. Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2009.

[6] University of Washington eScience Instittute. http://www.escience.washington.edu.

[7] D. Halperin, K. Weitz, B. Howe, F. Ribalet, M. A. Saito, and E. V. Armbrust. Real-time collaborative analysis with (almost) pure SQL: A case study in biogeochemical oceanography. In *SSDBM*, 2013.

[8] B. Howe, G. Cole, E. Souroush, P. Koutris, A. Key, N. Khoussainova, and L. Battle. Database-as-a-service for long tail science. In *SSDBM*, 2011.

[9] B. Howe, F. Ribalet, D. Halperin, S. Chitnis, and E. V. Armbrust. Collaborative science workflows in SQL. *Computing in Science & Engineering, Special Issue on Science Data Management*, 15(2), May/June 2013.

[10] Big Data U: PhD Program in Big Data and Data Science. http://data.washington.edu/.

[11] N. Khoussainova, M. Balazinska, and D. Suciu. Perfxplain: Debugging MapReduce job performance. *PVLDB*, 5(7):598–609, 2012.

[12] N. Khoussainova, Y. Kwon, M. Balazinska, and D. Suciu. SnipSuggest: Context-aware autocompletion for SQL. *PVLDB*, 4(1):22–33, 2010.

[13] Y. Kwon, M. Balazinska, B. Howe, and J. Rolia. SkewTune: Mitigating skew in MapReduce applications. In *SIGMOD*, 2012.

[14] Myria: Big Data management as a Cloud service. http://myria.cs.washington.edu/.

[15] Nuage: Scientific data management in the cloud. http://nuage.cs.washington.edu/.

[16] J. Ortiz, V. T. de Almeida, and M. Balazinska. A vision for personalized service level agreements in the cloud. In *Proc. of the DanaC Workshop*, 2013.

[17] The RFID Ecosystem. http://rfid.cs.washington.edu/.

[18] S. Roy and D. Suciu. A formal approach to finding explanations for database queries. In *SIGMOD*, 2014. To appear.

[19] SQL Share. http://sqlshare.escience.washington.edu.

[20] UW SciDB Branch. http://scidb.cs.washington.edu/.

[21] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[22] P. Upadhyaya, N. R. Anderson, M. Balazinska, B. Howe, R. Kaushik, R. Ramamurthy, and D. Suciu. The power of data use management in action. In *SIGMOD*, pages 1117–1120, 2013.

[23] J. Vanderplas, E. Soroush, S. Krughoff, M. Balazinska, and A. Connolly. Squeezing a big orange into little boxes: The AscotDB system for parallel processing of data on a sphere. *IEEE Data Eng. Bulletin*, 36(4):11–20, 2013.

[24] E. Welbourne, K. Koscher, E. Soroush, M. Balazinska, and G. Borriello. Longitudinal study of a building-wide RFID Ecosystem. In *MOBISYS*, 2009.

A Call for Energy Efficiency in Data Centers

Michael Pawlish
Montclair State University
Dept. of Env. Mgmt.
Montclair, NJ, USA
pawlishm1@montclair.edu

Aparna S. Varde
Montclair State University
Dept. of Computer Science
Montclair, NJ, USA
vardea@montclair.edu

Stefan A. Robila
Montclair State University
Dept. of Computer Science
Montclair, NJ, USA
robilas@montclair.edu

Anand Ranganathan
IBM Thomas J. Watson Research Center,
Hawthorne, NY USA
arangana@us.ibm.com

Abstract

In this paper, we explore a data center's performance with a call for energy efficiency through green computing. Some performance metrics we examine in data centers are server energy usage, Power Usage Effectiveness and utilization rate, i.e., the extent to which data center servers are being used. Recent literature indicates that utilization rates at many internal data centers are quite low, resulting in poor usage of resources such as energy and materials. Based on our study, we attribute these low utilization rates to not fully taking advantage of virtualization, and not retiring phantom (unused) servers. This paper describes our initiative corroborated with real data in a university setting. We suggest that future data centers will need to increase their utilization rates for better energy efficiency, and moving towards a cloud provider would help. However, we argue that neither a pure in-house data center or cloud model is the best solution. Instead we recommend, from a decision support perspective, a hybrid model in data center management to lower costs and increase services, while also providing greater energy efficiency.

Keywords: Cloud, Data Centers, Green IT, Utilization Rates, Energy Efficiency

1. INTRODUCTION

The data center is the backbone of the Internet that has provided tremendous communication gains; however, at the same time energy efficiency in data centers is often a secondary concern. The management of data centers is increasingly becoming more complex from dealing with legacy equipment, developments in technology such as blade servers and virtualization, and the present push to outsource much of the data center through cloud providers; all while top management has been keeping budgets level or seeking cuts. Traditionally, energy efficiency has therefore not been a top priority with data center managers, due to the aforementioned challenges of operating a data center. In this paper, we claim that following a hybrid business model that takes advantage of cloud technologies and the existing in-house data center will assist in developing a more effective strategy for energy efficiency.

There are a number of reasons to seek energy efficiency in computing facilities. First, in many places in the world energy consumption is increasing at a faster rate than new energy sources are being developed. In the United States, there is a tremendous push back by the public to any type of new large-scale energy production facilities. This push back results in delays in construction of new facilities, and according to supply and demand should result in future elevated energy costs due to the increasing demand [15].

A second reason for seeking data center energy efficiency is the pure economics of squeezing out inefficiencies in current systems [2]. In the rush to build data centers in the first decade of the 21st century, energy efficiency had a low priority. Now that the market has matured, there is a need to find gains such as low hanging fruit, for example, increasing the temperature in the data center or placing the lighting on motion detectors. By making data centers more efficient, or lowering the cost and environmental impact, management will see improvement in their operating costs.

A third reason to pursue an energy efficiency strategy is to keep current with emerging technology advances. For example, virtualization that allows more applications to run on fewer servers is an important technological development from an energy efficiency perspective [3, 12]. Virtualization has allowed the retirement of a number of servers, or basically has permitted more processing power to be computed with less electrical consumption [7]. Servers are therefore continuing to be built that are smaller and more powerful from previous generations.

Finally, another reason to seek out energy efficiency is public perception. In a recent cover issue of the Sunday *New York Times*, the data center industry was presented as the next wasteful and polluting industry of the 21st century [5]. This perception of the Information Age is contrary to the positive reputation that many individuals hold towards the Internet, and the article exposed many efficiency problems, including particularly the low utilization rate in data centers that is addressed in this paper.

This paper presents a detailed analysis over a three-year period of energy usage, and documents the low utilization rate in a mid-size university data center similar to a typical computing facility described by previously published literature [1, 8, 11]. Data mining techniques such as Case Based Reasoning (CBR) and decision trees are provided as approaches for decision support in the management of the center. Results from the analysis and mining support the arguments in favor of a hybrid data center. Here, existing local capacity is combined with an outside cloud provider as the most efficient strategy to pursue for enhanced service, low cost, and a more energy efficient model.

2. PARAMETERS IN DATA ANALYSIS

We obtained our data from our university data center, typical of most organizational data centers, in that the servers are not homogeneous. As characteristic of most in-house data centers, legacy equipment is the norm with differing vintages of servers and cooling components. Sampling was conducted manually by visiting the data center and recording energy usage over a three-month period during the spring semester of each year for three years. The purpose of documenting server energy usage was to establish a base line study, and document the carbon emissions. We focus on certain parameters for analysis as described next.

Table 1. Server Energy Usage

Date	PDU 1 kWh	PDU 2 kWh	PDU 3 kWh	PDU 4 kWh	Total
3/01/10-6/01/10	66,598	46,838	90,527	80,382	284,345
3/01/11-6/01/11	50,680	36,093	85,994	75,381	248,148
3/01/12-6/01/12	40,433	26,061	86,615	78,547	231,656

2.1 Utilization Rate

The utilization rate if defined as the extent to which the CPU is busy at any given instance of time, as stated in the Equation 1 herewith:

$$U = \frac{\sum_{n=1}^{T}(CPU\ Rate)}{T} \quad \text{--- 1}$$

Here U represents the utilization rate calculated as an efficiency ratio that sums up each instance of the CPU rate over a total time span T, such that CPU rate is the extent to which the CPU is busy at a given instance of time. Utilization rate gives management an idea of how much the data center is being used, and can be expressed as a percentage. Based on this, it is clear that it is desirable to increase the utilization rate for energy efficiency.

1) Observations from A Data Center Host

We consider a data center with two hosts that continually shift user demand for optimal performance. As an example we hereby present utilization rate calculation for a single day. The CPU rate per minute is emailed to us in a file based on continuous monitoring of data center hosts. We sum up this CPU rate and divide it by the total number of minutes per day to get the daily utilization rate.

Host 1-Thursday 6/14/12
$\sum$ CPU Rate = 49,350
Utilization Rate = 49,350/ 1440 =34%

Based on such calculations, Tables 2 and 3 give a broader picture of utilization rates for the first six months of 2012.

Table 2. Utilization Rates

2012	Host 1		
Month	*Average Utilization rate*	*Monthly low*	*Monthly high*
Jan.	*38%*	*7%*	*86%*
Feb.	*34%*	*10%*	*85%*
March	*30%*	*7%*	*60%*
April	*35%*	*8%*	*68%*
May	*35%*	*10%*	*63%*
June	*29%*	*9%*	*60%*

Table 3. Utilization Rates

2012	Host 2		
Month	*Average Utilization rate*	*Monthly low*	*Monthly high*
Jan.	*42%*	*20%*	*86%*
Feb.	*35%*	*25%*	*90%*
March	*38%*	*21%*	*87%*
April	*35%*	*9%*	*82%*
May	*38%*	*21%*	*84%*
June	*42%*	*18%*	*90%*

An initial observation is that average utilization rates are around 30% to 42%, which we believe is on the low side. To enhance energy efficiency, our argument is that data centers need to operate at higher utilization rates than these. From an economic perspective the cost of running data centers, as per our analysis, is that the data center is running at an optimal operation point only around 1/3 of the time. This is an apparent waste of resources that unnecessarily contributes to carbon emissions when fossil fuels are used for generating the required electricity. After examining the utilization rate,

we delve further into our case study through a metric called Power Usage Effectiveness as explained next.

2.2 Power Usage Effectiveness

The Power Usage Effectiveness (PUE) is an efficiency ratio of data centers that was developed by the industry, and is defined in the following Equation 2:

$$PUE = \frac{Total\ Facility\ Power}{IT\ Equipment\ Power} \quad \text{--- 2}$$

In theory, if the PUE equaled 1.0, the data center would be considered perfectly efficient since Total Facility Power would equal IT Equipment Power. In reality or practice, a PUE slightly above 1.0 has been observed in some ultra efficient data centers, for example, Facebook's Prineville data center located in Oregon. Presently, a PUE of around 2.0 seems to be the industry average since there is power lost in Total Facility Power for energy use by such components as lighting and cooling. Using such measures as efficient design factors, for example, airside economizing (free cooling) that uses outside air to lower the data center room's temperature, and therefore uses less power than traditional air-conditioning is a typical method to lower the PUE and the energy usage. There also seems to be a growing trend of locating data centers in higher latitudes to take advantage of the cooler climates. One such example has been the growth trend in data centers in Sweden, due to such factors as a stable government with cheap electricity that is derived from hydropower that does not contribute to carbon dioxide emissions.

In the fall of 2013, the second phase of our study was initiated with the installation of meters to measure the energy consumption of the data center. Due to relatively large PUE values observed and considering the installation of temperature/relative humidity sensors, a future research question would focus on how to lower the PUE. A next step will be to raise the temperature in the data center by 2 degrees Fahrenheit. The research team feels confident with the sensors in place to prevent hot spotting, and the team is curious of the savings in the carbon footprint and electricity cost. The ultra efficient cloud data centers are able to operate with a PUE slightly above 1.0 and that further supports our argument that hybrid computing is more energy efficient as discussed later in this paper. In the next sub-section, the carbon footprint of the data center is analyzed in order to assess its carbon dioxide emissions.

2.3 Carbon Footprint

From an energy management perspective, perhaps the most important parameter is the carbon footprint of an organization that represents the atmospheric carbon dioxide emissions that directly correlates with energy usage. More specifically, the carbon footprint of an organization is the estimated total of the output of carbon dioxide released in the atmosphere from primarily burning fossil fuels to supply the power for operations. In this case, we refer to the operations of the data center. Currently, the estimated amount of CO_2 released from data centers worldwide is approximately 2% which is a growing concern [4]. The standard formula to calculate the carbon footprint is given in Equation 3 as follows:

$$C = \frac{E*N}{T} \quad \text{--- 3}$$

Where C represents carbon footprint, E represents electrical usage in kWh per year, N represents national CO_2 emissions, and T represents metric tons (1 metric ton equals 2,204.6 lbs.) In our evaluation, we have recorded the energy usage of our data center servers and calculated the carbon footprint using the given formula. These values are summarized for a three month period in Table 1. Based on this, the total carbon footprint for data center servers at our university is calculated per year as stated in Equation 4 below. Consider that:

$$E_{year}=E_{sample}*4 \quad \text{--- 4}$$

Where E_{year} represents total yearly energy used in 2012, E_{sample} represents a sample of the total energy consumption over the three month period. The results are thus as follows for the energy usage of the servers in 2012.

$$E_{2012}=231{,}656\ kWh * 4 = 926{,}624\ kWh$$

The carbon footprint for the servers C_S is therefore calculated using Equation 1, considering N = 1.34 lbs/kWh as the national average of US CO_2 emissions [13].

$$C_S = 926{,}624\ kWh * 1.34\ lbs/kWh * 1\ metric\ ton/2{,}204.6lbs = 563\ metric\ tons/year$$

A metric ton conversion ratio is used because CO_2 emissions are commonly expressed in the international community in metric tons. Now consider the carbon footprint for cooling or air conditioning. The estimated electrical usage is 58 kW per hour with three air conditioning units running 7 days a week, and 365 days per year. The electrical power usage for air conditioning is 1,524,240 kWh/year. Thus, for example, the total carbon footprint for air conditioning C_{AC} in our data center is calculated as:

$$C_{AC} = 1{,}524{,}240\ kWh/year * 1.34\ lbs/kWh * 1\ metric\ ton/2{,}204.6lbs = 926\ metric\ tons/year$$

From Table 1 and the air conditioning power usage calculation presented above, we also obtain the combined power usage for 2012 including data center servers and air conditioning. This is calculated as 926,656 kWh/year (servers) + 1,524,240 kWh/year (cooling) = 2,450,864 kWh/year. Therefore, based on our measurements and estimations, our data center is contributing approximately 1,500 metric tons per year of CO_2 into the atmosphere that is not a good indicator. Especially considering that due to low utilization rates

presented in the next section of this paper, the majority of the time CO_2 emissions are being wasted on idle servers and the concerned cooling.

Given this analysis of parameters, we now consider case based reasoning and decision trees in addressing the problem of energy efficiency in data centers.

3. DEPLOYMENT OF CASE BASED REASONING

The data-mining paradigm of Case Based Reasoning (CBR) has been deployed in our work. CBR discovers knowledge from previous cases or examples and uses that for reasoning about other similar cases in the future. A typical CBR model uses the R4 cycle: Retrieve, Reuse, Revise and Retain. In R4, we retrieve a similar past case, reuse it to fit the current scenario as far as possible, revise it using methods in the field of "adaption in CBR", and then retain the adapted learned case as for future cases.

In our study we use CBR in various examples, one of which is shown in Figure 1. In this example, we examine the case where there is inefficient use of energy in data centers. Following Figure 1 in a clockwise rotation based on the R4 cycle yields a four step process as follows. The first step in this cycle retrieves relevant information pertaining to the potential to lower CO_2 and energy usage by $1/50^{th}$ by shifting email operations to a cloud provider. This estimation is calculated by considering that this data center has approximately 50 data racks, and the student email system takes up about one full rack. (Note that the employee and faculty email were not outsourced earlier due to legislation and privacy issues). The second step in the R4 cycle involves reusing the information that recommends the use of higher energy efficiency in cloud providers that will result in more efficient resource use. The third step in the cycle is to revise the case with the recommendation of our main argument for a hybrid model. This suggests using the existing data center through higher in-house server utilization, plus backup provided by a third party cloud company. A hybrid model will resize existing data centers, and shift spikes in demand to an outside cloud provider. The final step in the CBR cycle is to retain the new knowledge for the future as the learned case. This places an emphasis in continual data center management that measures and monitors metrics such as server sprawl, energy usage and utilization rates, while using a portfolio management approach to determining which applications are candidates for a cloud provider.

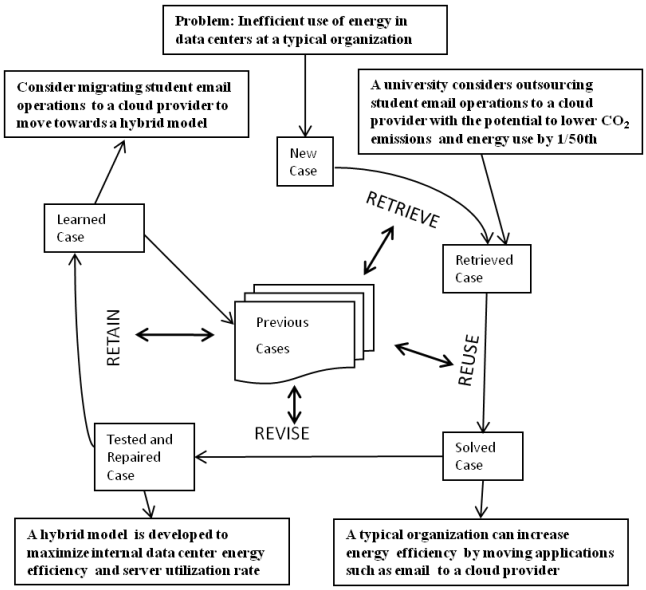


Figure 1: CBR for Energy Efficiency in a University Data Center

Based on our CBR model where $1/50^{th}$ of the electricity and resulting carbon emissions could be transferred to a cloud provider by outsourcing the student email, the question remains if the cloud provider could be more energy efficient than the internal data center. If there was greater energy efficiency in a cloud provider by utilizing, for example, hydro-electric power or by utilizing resources more efficiently, there would be a net benefit. The equation for this translation of the net carbon benefit would be the following, i.e. Equation 5:

$$C_{Benefit}=1/50*1500\ CO_2\ tons=30\ CO_2\ tons \quad ---5$$

The net carbon benefit, $C_{Benefit}$ would result in $1/50^{th}$ of 1,500 calculated metric tons of CO_2 from our data center which translates to approximately 30 metric tons per year of CO_2 savings (minus the addition of any CO_2 from the cloud provider). Currently, the data on a rack level or server basis is not provided by cloud companies, and we realize that our argument is based on the assumption that cloud providers are more resource-efficient, since that is a key operating goal of cloud providers. However, in all scenarios this may not be the case, e.g., when the cloud provider is using fossil fuels as an energy source.

4. ANALYSIS WITH DECISION TREES

While CBR examines specific cases, a decision tree follows a logical path on more of a general problem. Thus, decision trees have a specific starting point and flow through a series of questions to a recommended strategy. In the decision tree in Figure 2, the starting point examines whether the PUE is greater than 2.5, which is set as a baseline. This is because it has been found from our discussions with data center

personnel that industry standards for PUE are usually below this number. Energy usage is increasingly becoming an important factor for management to measure in order to achieve a more energy efficient data center, and the PUE is an efficiency ratio of energy use.

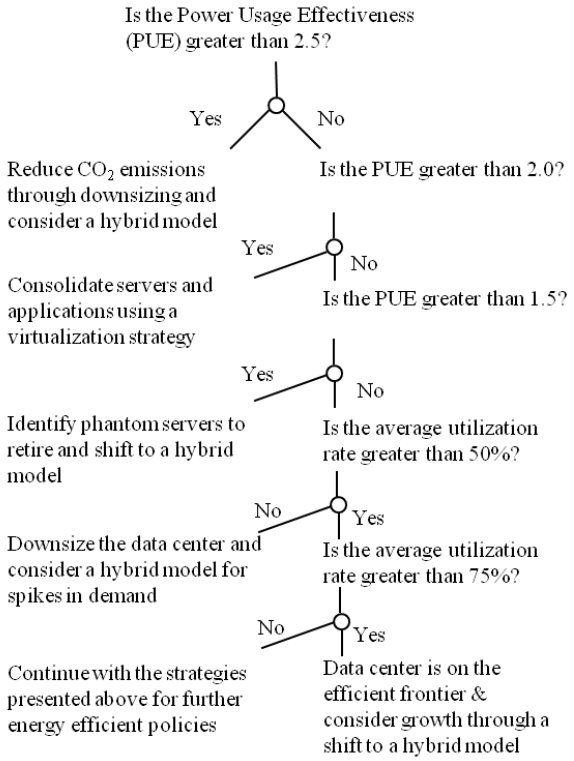


Figure 2: Decision Tree Examining PUE & Utilization Rate

As illustrated in this decision tree, a PUE of 2.5 was selected for initial comparison since currently most in-house data centers are operating at a higher level than cloud or external data centers. Next, the decision tree moves on to follow paths to achieve greater energy efficiency, with a second step to see if a virtualization strategy has been developed to reduce servers by moving more applications to fewer servers. Traditionally in the past, the general rule of thumb was to have one application per server, but this has proved to be costly and inefficient from a natural resource perspective. The consolidation of applications to fewer servers is a first step in a series of solutions that can be implemented simultaneously with other strategies, such as retiring phantom servers. Examples of phantom servers are servers that are still in operation that are not completing useful work, that were typically left on from previous administrators.

The decision tree moves on from the PUE analysis to analyze the utilization rate in blocks of 25% higher utilization rates. In each decision, further strategies are identified while optimizing the data center towards a hybrid strategy. The more efficient data centers will operate at around a 75% utilization rate with applications and spikes in user demand shifted to an external data center, i.e., cloud.

5. PROPOSAL FOR A HYBRID MODEL

Increasing utilization rates and lowering the PUE in data centers for enhanced energy efficiency is important for lowering the carbon footprint of organizations. In addition we propose a new paradigm of running a data center on a hybrid model as presented in the CBR example in Figure 3. The model begins with the first step of reducing the number of servers by 25%, through shifting spikes in demand to a cloud provider in step two. The goal in the third step is to increase the utilization rate to 70-80%, with the final objective achieved by greater virtualization and lowering the number of physical servers.

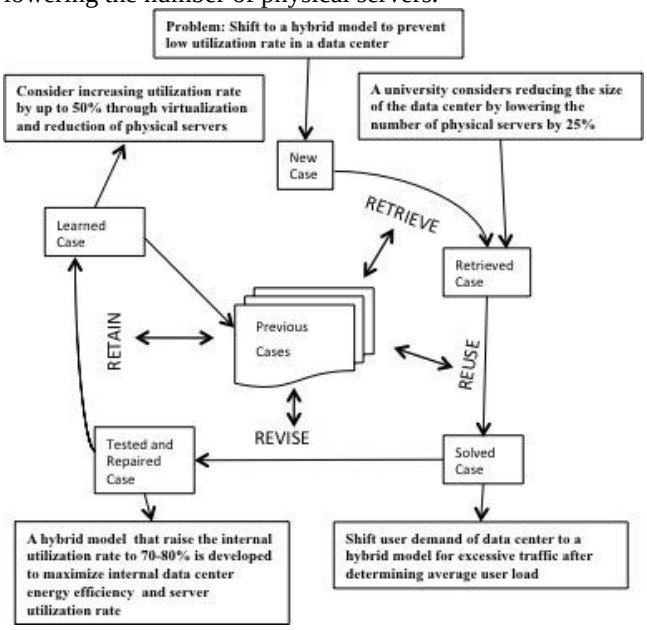


Figure 3: CBR for Shifting to a Hybrid Model

The reasons for using a hybrid model are the following based on our analysis between the trade-offs of an in-house data center and an external data center, i.e. cloud:

- Due to economies of scale most cloud providers can operate with a lower PUE and a higher utilization rate through having data centers geographically distributed.
- The PUE of in-house data centers tend to be higher than PUEs found in a typical external or cloud data center.
- An argument for keeping an in-house data center essentially boils down to security and privacy issues that industry and society will continue to develop into the future. For example, if health care records were kept on the cloud it may be more efficient, but people would be concerned that insurance

companies could obtain their records and deny coverage.

- Resistance to change by personnel is yet another issue. For example, data center managers are familiar with SQL based packages in a traditional database setting. Migrating to the cloud and using packages such as Hadoop/Hive could involve additional training.
- Our final argument is that current in-house data centers are overbuilt, since these data centers have been designed for peak usage. Typically, peak usage only occurs a few days of a year such as at the end of an accounting period, or during peak shopping seasons.

Therefore, while the usage of cloud computing is presently debated, we believe that cloud computing presents the next large wave in information technology. The economies of scale of cloud computing has brought forth an age where it is no longer necessary to provision computing needs for the future combined with elastic demand while all being instantaneous. In many cases, these benefits of the cloud outweigh the fixed costs of owning expensive capital and the operational costs of internal data centers depending on the organization. One of the most important factors is the flexibility provided by cloud computing which could lead to a competitive advantage in organizations depending on implementation of strategy.

We thus put forth a proposition that the answer for mid to large size organizations is a hybrid model of operating a data center, and we present the idea in both bullet point and a decision tree format. To transition to a hybrid model, we recommend four strategies as stated below:

1. First determine the rate of growth of the data center. To accomplish this energy usage needs to be recorded. For example, in the decision tree in Figure 4 an arbitrary number of 5% growth is selected, and each organization can select a goal to contain its energy usage accordingly.
2. Phase out 25 to 50% of servers due to low utilization rates, depending on organizational goals and objectives. The objective would be to match average utilization rates per month with actual server usage. Once again, in cases of excess demand an outside cloud provider would be secured.
3. Develop a data center strategy of keeping mission critical information on local servers and down size the data center by shifting non-critical information or applications to a cloud provider. To provide for back up in the local data center, a secure strategy would call for a cloud provider to additionally provide support for mission critical data.
4. Shift to public applications that are run on the cloud. For example, many applications such as payroll, human resource management, email, and customer service management are now provided by cloud software. We believe that this trend of cloud-based software will be the future technology that will have implications on the local data center by decreasing demand on present operations. We suggest that operations involving high security and privacy issues be retained on the internal data center servers.

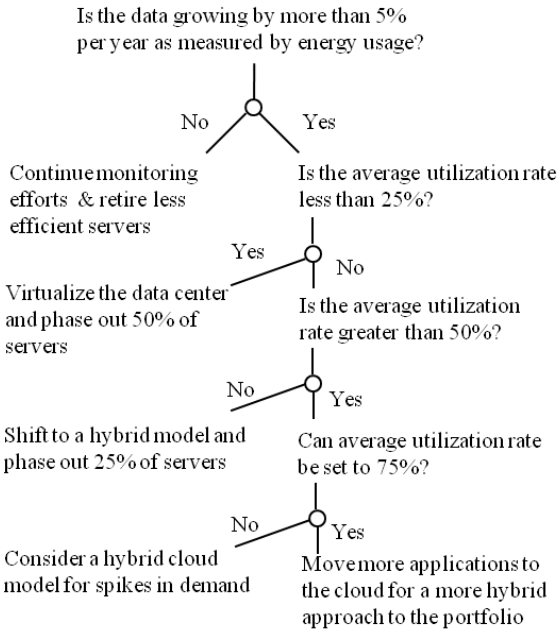


Figure 4: Decision Tree for Moving To a Hybrid Model

A more robust hybrid model as we envision it, would combine all four of the above mentioned strategies, and as we also envision future internal data centers operating at higher average utilization levels of 70 to 80% with spikes in demand and redundancy for backup supported by a cloud provider such as Amazon, Rackspace, Microsoft, Google or similar. These companies have cloud facilities that are geographically diverse while shifting demand to operate at higher utilization rates that support more efficient energy management. These commercial cloud providers generally do not make available information on their energy use or utilization rate performance due to releasing strategic information to competitors, but it would be expected that these cloud providers would be efficiently operating their facilities to reduce such factors as server sprawl, and increase such factors of virtualization, since that is their main operational goal.

6. CONCLUSIONS AND FUTURE WORK

From our analysis of a typical data center, utilization rates have been documented as operating on the low side, and the literature on this subject also documents other data centers operating with low utilization rates. From a broad perspective this is a

societal problem, since resources in the form of energy and materials are being wasted, and the energy used is producing unnecessary carbon dioxide emissions when fossil fuels are the fuel source. To solve this problem, we recommended a shift in thinking of data center operations to a hybrid model with the following advantages:

- A shift to a hybrid model is that existing data centers are more fully developed by gaining higher utilization rates, or in other words the servers are more efficiently run.
- This hybrid strategy would involve the increasing use of virtualization with more applications running on fewer machines.
- The strategy would also rely on cloud providers to provide backup for mission critical operations, as well as providing for increased spikes in user demand.
- Operating the data center from a hybrid model would enhance energy efficiency, and we believe contribute to enhanced use of natural resources.

Finally, from a strategic perspective, the most important characteristic of implementing the cloud is the flexibility gained. The ability to have a variable cost instead of a fixed cost or asset will provide growth for innovation and experimentation on different business models. While it is impossible to predict new businesses that may develop in the future, the ability to be flexible and innovative have proven over time to be successful characteristics of organizational growth.

7. ACKNOWLEDGMENTS

This research has been supported by a grant from the PSE&G Company. We would like to thank PSE&G for their encouragement and support.

8. References

[1] Anderson, S. (2010) Improving Data Center Efficiency, *Energy Engineering*, Vol. 107, No. 5, pp. 42-63.

[2] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2010) A View of Cloud Computing, *Communications of the ACM*, April, Vol. 53, No. 4., pp. 50-58.

[3] Donnellan, B., Sheridan, C., and Curry, E. (2011) A Capability Maturity Framework for Sustainable Information and Communication Technology, *ITPro*, January/February, pp. 33-40.

[4] Forge, S. (2007) Powering Down; Remedies for unsustainable ICT, *Foresight*, Vol. 9, No. 4, pp. 3-21.

[5] Glanz, J. (2012) Power, Pollution and the Internet, *New York Times*, Sept. 23, pp A.1.

[6] Greenberg, A., Hamilton, J., Maltz, D., and Patel, P. (2009) The Cost of the Cloud: Research Problems in Data Center Networks, *ACM SIGCOMM Computer Communication Review*, Vol. 39, No. 1, Jan., pp. 68-73.

[7] Kansal, A., Zhao, F., Liu, J., Kothari, N., Bhattacharya, A. (2010) Virtual Machine Power Metering and Provisioning, *Proceedings from ACM Symposium on Cloud Computing*, June 2010, pp. 39-50.

[8] Kapoor, R., Porter, G., Tewari, M., Voelker, G., Vahdat, A. (2012) Chronos: Predictable Low Latency for Data Center Applications, *ACM Symposium on Cloud Computing*, Oct 2012.

[9] Koomey, J. (2011) *Growth in Data Center Electricity Use 2005 to 2010*, Analytics Press, Oakland, CA.

[10] Pawlish, M. & Varde A. S. (2010) A Decision Support System for Green Data Centers**,** *ACM CIKM Workshops Program*, Oct 2010, pp. 47-56.

[11] Pawlish, M., Varde, A. S., & Robila, S. A. (2012) Cloud Computing for Environment-Friendly Data Centers, *ACM CIKM Workshops Program*, Oct 2012, pp. 43- 48.

[12] Pedersen, T.B. (2010) Research challenges for cloud intelligence: invited talk, *EDBT Workshops Program*, March 2010, pp. 6:1.

[13] Schulz, G. (2009) *The Green and Virtual Data Center*, Auerbach Publications, Boca Raton, Fl., pg. 11.

[14] Siegele, L. (2008) Let It Rise: A Special Report on Corporate IT. *The Economist*, October, pp 1-14.

[15] World Commission on Environment and Development (1987) *Our Common Future*. Oxford: Oxford University Press**.**

Opportunistic Data Services in Least Developed Countries: Benefits, Challenges and Feasibility Issues

Nicolas Anciaux[1, 2], Luc Bouganim[1, 2], Thierry Delot[3,1], Sergio Ilarri[4], Leila Kloul[2], Nathalie Mitton[1], Philippe Pucheral[1, 2]

[(1)]INRIA, France
fname.lname@inria.fr

[(2)]PRISM, UVSQ, France
fname.lname@prism.uvsq.fr

[(3)]LAMIH, UVHC, France
Thierry.Delot@inria.fr

[(4)]Univ. of Zaragoza, Spain
silarri@unizar.es

ABSTRACT

According to many studies, IT should become a key facilitator in establishing primary education, reducing mortality or supporting commercial initiatives in Least Developed Countries. The main barrier to the development of IT services in these regions is not only the lack of communication facilities, but also the lack of consistent information systems, security procedures, economic and legal support, as well as political commitment. In this paper, we propose the vision of an infrastructure-less data platform well suited for the development of innovative IT services in Least Developed Countries. We propose a participatory approach, called Folk-IS, where each individual implements a small subset of a complete information system thanks to highly secure, portable and low-cost personal devices as well as opportunistic networking, without the need for any form of infrastructure. In this paper, we focus on the exploitation and feasibility analysis of the Folk-IS vision. We also review the technical challenges that are specific to this approach.

1. INTRODUCTION

Least Developed Countries (LDCs) are 50 countries with nearly 1 billion people, which meet UN criteria in terms of poverty, human resource weaknesses and economic vulnerability. LDCs have no global access to the Internet and are more than ever left by the wayside with respect to Information and Communication Technologies (ICT).

Does this mean that e-services are a superfluous luxury for LDCs? On the contrary, several reports [14, 20, 31] make evident that ICT are called to play a catalytic role in these countries: ICT can help to achieve universal primary education, promote gender equality and empower women, reduce child mortality, combat HIV and other diseases, ensure environmental sustainability, and develop a global partnership for development.

However, the challenge of making e-services practical and affordable still remains. While 60% of the population in LDCs is already covered by a mobile cellular signal, only 0.5% has a mobile broadband subscription and a 3G service is offered only in at most 25% of the LDCs, very often at a prohibitive cost [20]. Hence, mobile phones in these areas are primarily feature phones used for voice and SMSs. Several innovative solutions are thus based on the use of mobile phones and text messages to address specific issues like keeping track of vaccine cold chains [13], improving agriculture [33], or easing administrative procedures [29], but cannot be generalized to the broader scope of data-driven applications. Major industrial actors like Google (project Loon for All) and Facebook (Internet.org) are also conducting pioneering projects in this field, showing the potential for new applications and business in these areas. However, providing general data services in LDCs faces many obstacles which go far beyond low connectivity: severe lack of IS infrastructure, high initial investment, difficulty to maintain the system operational, reluctance to use the system due to security concerns, etc.

According to Non-Governmental Organizations (NGOs), four main requirements must be met to build a practical technical solution: (1) **privacy protection**: this is a major prerequisite due to local opaque practices and the lack of any security infrastructure (coercive laws, secured servers, trusted authorities, etc.), leading to a self-enforcement of privacy principles; (2) **immediate personal benefit**: because of the lack of strong economical or political incentives to impose the solution in the field, an immediate benefit must be provided to each user; (3) **self-sufficiency**: the solution must not rely on an hypothetic improvement of the existing software and hardware infrastructure; and (4) **very low deployment cost**: the usual scale being a few dollars per user. Besides this, users' empowerment is known to be crucial to make the solution sustainable in LDCs, and its maintenance should ideally generate a source of revenue for new local jobs.

In [3], we proposed the vision of trusted cells, a data platform for personal data services where the shared infrastructure (typically the Cloud) is untrusted, while personal devices (such as smart phones, tablets or set-top boxes) are trusted execution environments. In this paper, we revisit this vision with the goal of providing data services to the inhabitants of LDCs. We propose a new paradigm, that we call *Folk-enabled Information System* (Folk-IS), based on a fully decentralized and participatory approach, where each individual implements a small subset of a complete information system without the need for a shared networked infrastructure. As trusted cells, Folk-IS builds upon the emergence of highly-secure, portable and low-cost storage and computing devices, called hereafter

Smart Tokens. Here, however, the focus is on the low-cost of ownership, deployment and maintenance, and on the absence of a networked infrastructure. In Folk-IS, smart tokens have the capability to host the digital history of their owners and to exchange information in a privacy-preserving manner. This enables people to transparently and opportunistically perform data management and networking tasks as they physically move, so that IT services are truly delivered by the crowd.

The potential advantages are important for the inhabitants (develop economics activities, hold a digital folder, communicate with friends, with NGOs or administrations, etc.), for the community (have communication means with inhabitants used to feed cultural programs or launch e-admin procedures like census, detect epidemics, etc.), and for the NGOs (conduct monitoring programs, consult/fill inhabitants folders when visiting them, etc.).

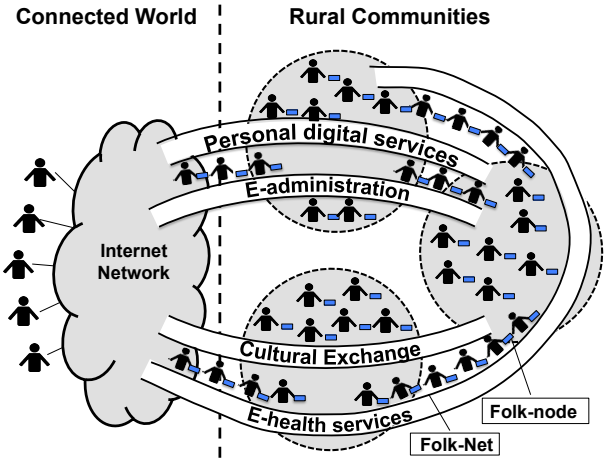


Figure 1. Folk-enabled information system

We do not argue that Folk-IS is the ultimate solution. The future of IT in LDC will probably be multiform, the problem being important and complex enough to leave room for complementary initiatives. Folk-IS has the salient characteristics to enable a smooth and incremental deployment of an information system in a purely infrastructure-less context while taking advantage of existing elements of infrastructure, if any, to improve its own behavior.

In [4], we presented our overall Folk-IS vision. This paper details this vision and shows that this paradigm technically makes sense, conforms to the requirements mentioned above and opens important and exciting research challenges for the database community.

2. REPRESENTATIVE ICT INITIATIVES IN LDCs

Applying ICT in developing countries has recently attracted considerable attention, reflected in recent special issues, such as [14, 31]. Several of the proposed applications aim at making use of mobile ICT services to resolve key challenges such as providing content that is locally relevant and mitigating access barriers linked to literacy and language [40]. However, when developing data-driven solutions for LDCs, important difficulties arise. The coverage of mobile phones in LDCs is increasing, but phones are primarily used for voice and SMSs, not for data-driven applications. Sir Tim Berners-Lee mentioned a recent study that showed that in Mozambique "*using just 1GB of data can cost well over two months wages for the average citizen*"[1]. According to many analysts, this situation cannot evolve rapidly due to a combination of intrinsic barriers, not only technical, but also economical and organizational [9].

Many isolated initiatives have been launched with specific strategies for getting around these barriers, and propose ad-hoc solutions for specific problems. In relation to health issues, for example, we could mention *FoneAstra*, *CVDMagic*, NextDrop, and *mWash*. *FoneAstra* [13] is a low-cost sensing system using mobile phones to keep track of vaccine cold chains. *CVDMagic* [34] proposes the use of mobile-based approaches to detect Cardio-Vascular Diseases (CVD) in India. *NextDrop*[2] provides SMS alerts indicating when a tap will have water, at least 60 minutes in advance. Finally, *mWash*[3] and [10] consider the problem of collecting and disseminating information to ensure a safe access to sanitized water. Other interesting topics tackled in relation to developing countries are related to agriculture [33], and e-government [29], among others; see [28] for more examples of interesting ICT applications for developing countries. These solutions address key issues in developing countries and bring actual improvements. However, despite their undisputed interest, these initiatives usually exploit only text messages for exchanging data, and remain confined to specific applications. Their generalization also faces other barriers like the lack of global information system infrastructures, of security procedures, and economic and political support.

Supporting more powerful form of data-driven applications to provide global e-services undeniably requires improved network connectivity. However, according to many analysts, even if increasing the number of "urban" spots would be an obvious solution, their deployment is very challenging as it would require more base stations, and hence more energy with collateral difficulties, like the need to carry gas-oil and protect it against theft [20]. To overcome the limitations of Internet coverage, several pioneering proposals do not rely on a hypothetic future improvement of the infrastructure, but propose new alternatives. Thus, proposals [30, 35] exploit mobile ad-hoc networks to provide asynchronous connectivity for data-oriented services. Specifically, in *DakNet* [30] mobile access points are mounted on physical means of

[1] http://a4ai.org/press-centre/what-our-supporters-say/
[2] http://nextdrop.org/
[3] http://www.pacinst.org/reports/mwash

transportation (e.g., buses, motorcycles) to transport data or information requests from one place to another. The related approach in [35] suggests the use of buses and cars to carry data between villages and Internet gateways. In relation to these proposals, the so-called *Delay-Tolerant Networks (DTNs)* [15] are very interesting, as they suit well with the idea of benefitting from opportunistic links when possible. Moreover, they show that huge benefits (e.g., email services) can be provided to users by means of asynchronous communications, even if this is at the expense of a potentially (unavoidable) high latency.

Google project Loon[4] is another ambitious attempt to connect people in rural and remote areas and bring people back online after disasters thanks to a network of high-altitude balloons. This is promising, although it still remains unclear if such a network infrastructure is durable, since current balloons can only function for a few weeks. In Cape Town (South Africa), Google also participated in *The Cape Town TV White Spaces (TVWS) trial*[5], which evaluated the possibility to benefit from unused parts of the frequency bands used for television to enable Internet access [28]. For its part, *Internet.org* promotes low-cost wireless handsets, compression of Web pages, and data caching on the edge of the network, to reduce data transfer. It is complemented by the Alliance for Affordable Internet[6], a coalition including the public sector and private companies (such as Google and Facebook) as well as civil organizations, which pushes towards progresses in regulation and policy.

These different initiatives demonstrate the growing interest, as well as the high difficulty, to integrate LDCs in global IT networks. Obviously, it is still not possible to predict which of those attempts may succeed, and when. Loon project architects believe that this is going to work commercially, but confess that it may be very slow going. In addition, low connectivity is clearly not the ultimate and unique barrier. As already mentioned, the generalization and deployment of e-services in these regions faces many other obstacles: the lack of consistent information systems, security procedures, economic and legal support as well as political commitment, and the difficulty to ensure the maintenance of the system.

The specificity of our proposal is to consider the aforementioned obstacles as characteristics of LDCs and to propose a holistic and general data management solution that supports secure data-intensive applications, while taking advantage of existing elements of infrastructure, if any, to improve its own behavior. So, we consider initiatives focusing on communication aspects, like DakNet or project Loon, as complementary to Folk-IS, since any additional means to reduce the network latency and increase connectivity would benefit our vision.

[4] http://www.google.es/loon/
[5] http://newweb.tenet.ac.za/tvws/.
[6] http://a4ai.org/

3. FOLK-IS ARCHITECTURE

As stated in the introduction, the Folk-IS paradigm builds upon the emergence of smart tokens, i.e., highly secure, portable and low-cost storage and computing devices, which may have various form factors and characteristics. In this section, we consider an abstract *Smart Token*, to focus on the foundation of Folk-IS and on its mode of operation with no consideration for specific technical choices. We postpone the discussion on real hardware platforms and concrete architectures to Section 5.

Smart Tokens: We consider Smart Tokens embedding at least: (1) enough stable storage to host the complete digital environment of its holder, (2) enough tamper-resistant computing resources to run a server managing the data and enforcing access control rules, and (3) a biometric sensor to authenticate users (e.g., a fingerprint reader, well adapted to illiterate people). Smart Tokens require also input/output capabilities to interact with users and communication facilities (e.g., short-range communications) to exchange messages. To meet the low-cost requirements, a smart token may inherit part of these functionalities from a terminal it connects to, assuming that it does not introduce security breaches.

Folk-enabled Network (c): since Folk-IS cannot rely on existing communication infrastructures, an ad hoc network of smart tokens is used for transferring messages (i.e., network packets) among smart tokens and from smart tokens to Internet access points. This can be achieved either by using short-range communications between smart tokens (assuming they are equipped with such communication facilities) or through terminals where smart tokens connect to (assuming messages are uploaded on the terminal and retrieved by others smart tokens which will next connect to it). So, communications are opportunistic by nature.

Folk-IS Personal Node (Folk-node): it is associated with each individual and refers to the combination of a Smart Token and of the embedded software components required to manage the holder's data, act as a network node, and enforce the security of the whole system.

Folk-enabled Information System (Folk-IS): based on the existence of Folk-nodes, we can devise a system implementing the three main functions of an information system as follows:

- *Communication management:* by carrying and routing data, each Folk-node acts as a node in the Folk-Net. Assuming that each Folk-node maintains a history of its moves (e.g., by gathering the GPS coordinates of all the devices it connects to), it can forward messages to encountered Folk-nodes (directly, or through terminals) whose moving profile best matches the recipient's location. Such georouting protocol provides a much better resource utilization of the Folk-Net than a basic flooding protocol.
- *Data management:* each individual centralizes in his Folk-node all his personal data (e.g., medical records,

administrative documents, credentials). Each time the holder interacts with a data source (e.g., by physically meeting a data provider, like a doctor, or by receiving a document through the Folk-Net), the application simply inserts these data in a local Folk-node database. From the information system viewpoint, the global database is the "union" of all these local databases. Each Folk-node is assumed to participate in common services, like processing global queries (broadcasted through the Folk-Net) and ensuring data durability (thanks to data replication among Folk-nodes).

- *Access control & authentication:* each Folk-node controls the access to the data it hosts and strongly authenticates the requesters (and its owner). Performing this control on the user's side is the ultimate solution to increase the holder's confidence, enforce his consent, and minimize the benefit/cost ratio of an attack. Indeed, the complexity of attacking the system is increased by the Smart Token tamper-resistance and by the obligation to be physically in contact with it to attack it. In parallel, the benefit of the attack is limited to disclosing/corrupting the data of a single individual. Note that the holder himself must authenticate and does not have all the privileges on his own Folk-node (e.g., he cannot tamper his own medical data to prescribe himself new drugs). In addition, messages carried by a Smart Token or replicas from other individuals stored locally are encrypted with keys never available to that Smart Token.

Hence, the complete system (data storage and network facilities) progressively deploys itself as Folk-nodes and shared devices are distributed to the participants. Folk-IS is by construction highly-redundant and robust, it does not require any central administration, and its global cost is directly proportional to the scale of the targeted population. As discussed next, some Folk-IS functionalities could be delegated to specific workers (e.g., people renting terminals or acting as postmen) in order to improve the quality of service (QoS) while creating a new local economic model (e.g., like people renting their cell phones in LDCs [9]). If elements of infrastructure are available, the Folk-IS QoS also improves accordingly, decreasing the network latency by shortening the route to the destination (or to the nearest Internet access point).

Figure 2 illustrates the Folk-IS mode of operation. It shows two rural communities, residents (black icons with letters) and their possible moves (grey icons), a *school* and a *first aid* room used by residents from both communities, and an Internet access point. Data exchanges through terminals are represented by dashed lines (e.g., the Folk-node of person A is used from the terminal of nurse N in the infirmary), and short-range data exchanges are represented by pink halos (e.g., between A and B in *Community 2*). E serves as a *netman* (i.e., a postman conveying digital messages) and carries data when travelling from place to place (e.g., the school, the infirmary and the Internet access point) thanks to his Folk-node. Different routing paths to transmit a message from A to the Internet are represented, e.g., the path ABTE: person A → person B → T (teacher) →E (netman) → Internet. These paths benefit from the physical mobility of people and their devices.

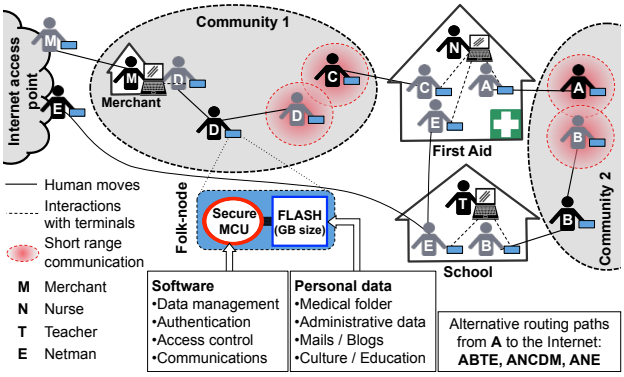


Figure 2. Folk-IS: mode of operation

4. SCENARIOS

Based on this mode of operation, several concrete scenarios can be envisioned. For illustration, we show examples below.

Healthcare scenario: In most LDCs, healthcare is provided by local nurses/doctors working in first aid rooms or intervening in rural communities during drought periods or health programs. They only possess very basic medicines and equipment. Needless to say, they do not benefit from an Electronic Health Records (EHRs) system, which is already highly difficult to organize in developed countries. Even paper-based medical records are too complex to maintain because many people have no identity document to link them to their records and people move according to seasons and drought periods. By using a Folk-node, each patient owns his complete and up-to-date medical folder. Without any Internet connection, a nurse can access the fingerprint-authenticated patient's medical folder, and appends diagnoses and treatment information. She may request medical advice from a distant clinic, by transmitting documents (e.g., a picture of the patient's injury) from the patient's Folk-node through the Folk-Net. She can notify the patient a few days later, still through the Folk-Net, in case a serious problem has been detected. In this case, the same architecture could also be used to detect and anticipate epidemics to better limit and fight them. Global queries can also be broadcasted through the Folk-Net to conduct epidemiological studies or maintain health indicators on populations unreachable so far. Hence, each Folk-node acts as a smart Personally Controlled Electronic Health Record (PCEHR).

Cultural heritage: Benefiting from the distributed and traveling Folk-IS architecture, new practices could be put in place to produce and share pieces of cultural heritage,

within communities or towards the outside world through virtual museums. Thus, with the help of cultural organizations, users able to produce pieces of a cultural heritage, related to traditional stories, songs, beliefs or cultural events, could be identified and equipped with Folk-nodes. Other users able to describe or translate such productions could also be equipped, and share/publish the enriched cultural content to virtual museum. The exact contribution of each actor (e.g., content producer, descriptors, translators, etc.) could be recorded, watermarked and certified by his tamper-resistant Folk-node. This may generate a new economic model where actors could be paid according to their contributions.

Generalization: Many usual scenarios can be transposed to rural communities thanks to Folk-IS, provided that they accommodate high-latency data exchanges: scholar folders for children, e-administration procedures (e.g., drought warnings, recording births and deaths), personal data applications (e.g., email, social or networking services), etc. This may also generate a new economic model where every actor could be paid according to his contribution, as exemplified in the cultural heritage scenario. It could also be the case for people playing the role of a *netman*, who could be paid according to the volume of transported data, the amount of population touched and the distance covered.

5. CONCRETE FOLK-IS INSTANCES

Let us now go from abstract smart tokens to the concrete hardware platforms. Given our target, an essential element in the design of the Folk-node's hardware is its cost, since it determines alone the overall cost of the Folk-IS solution. To decrease this cost, we study whether some of the Folk-node's hardware resources can be shared while minimizing the impact on the Folk-node's functionalities. To this end, we distinguish between Folk-node hardware platforms associated with individuals and shared devices which are either made publicly available in specific places or owned by local workers (e.g., nurses, teachers, administration officers, merchants). In the following, we first describe each Folk-node's resource arguing the pros and cons of sharing it. Then we examine the design of shared devices. Finally, we consider the whole Folk-IS architecture and discuss some deployment scenarios.

5.1 Folk-IS Hardware Platforms

5.1.1 Resources in Folk-nodes

As mentioned in Section 3, Folk-nodes are expected to embed storage, security, communication, location, and I/O resources.

Storage resources. Flash memory is undoubtedly the most adequate storage medium, thanks to its robustness, compact form factor, low cost, and low energy consumption characteristics. The main storage resources must be located on the Folk-node itself for obvious availability reasons (e.g., the holder's medical folder must be accessible anytime and anywhere when carried by the holder) and then cannot be shared. The storage capacity must afford the storage of all the holder's data as well as the persistent data required by common services (e.g., durability data, network packets, etc.).

Security resources. Secure microcontrollers (SMCUs) [11] can provide tangible security guarantees, and are today available at very low cost, in a small form factor (e.g., a SIM card), generally combined with an external (unsecure) flash chip. Existing SMCUs typically embed a relatively powerful processor (e.g., clocked at about 50 MHz), a cryptographic co-processor, and an internal secure stable storage (around 1 MB) to store the embedded code (e.g., the OS, a data management engine, the authentication mechanism, etc.) and sensitive metadata (e.g., cryptographic material used to protect the holder's data). Since we cannot assume that each user has an electronic identity (this would require certificates and a PKI –Public Key Infrastructure–, which contradicts our infrastructure-less assumption), we consider the use of a biometric sensor to prove the identity of the user. Biometric sensors (e.g., fingerprint readers) seem well adapted to illiterate users since this avoid remembering and typing a password or PIN code. At the personalization step (i.e., before the first use), the user's fingerprint is stored inside the SMCU's secure memory and it is then checked at each access.

SMCUs and fingerprint readers are at the heart of the security system of Folk-IS. They cannot be disassociated from the Folk-nodes (i.e., associated with the terminals the Folk-nodes connect with) without introducing the risk of class-breaking attacks (i.e., breaking a single-shared terminal would enable access to the sensitive data of all the Folk-nodes sharing that terminal). The privacy and security requirement is so fulfilled, due to a combination of factors: (1) the guarantee of an implicit user's consent (no access to the data without the holder's fingerprint check) and of a strong authentication of local workers; (2) the obligation to be physically in contact with the device to attack it; (3) the tamper-resistance of the Folk-node's processing and storage units, which makes hardware and side-channel attacks highly difficult; (4) the certification of the embedded software platform, which makes software attacks also highly difficult; and (5) the impossibility, even for the Folk-node's holder, to directly access the data stored locally or involved into local computations (he must authenticate and can only get data according to his privileges).

Communication resources. To enable the human network described in Section 3, Folk-nodes need a means to communicate together. This could be achieved either by using short-range communications or by physically plugging Folk-nodes in shared devices (messages can be temporarily stored on the shared device and delivered to Folk-nodes that will be plugged later). The advantage of enabling Folk-nodes with wireless capabilities is that messages can be exchanged without help from shared devices. However, wireless communications need energy, thus imposing the need of a battery and some mechanism to

charge it (e.g., energy harvesting technologies). Solutions with a limited amount of energy can also be exploited to improve the connectivity of the Folk-net, as discussed in Section 6.2.

Location resources. The network routing will be done based on the geographical location of the devices, raising the need to obtain at least an approximate location of Folk-nodes. Different alternatives could avoid embedding a costly positioning mechanism in each Folk-node. For example, the location information can be deduced from interactions with a small subset of localized Folk-nodes, shared devices, or fixed nodes. The user's home address can also be stored on each Folk-node at the personalization step, giving approximate static information about its location.

Input/Output (I/O) resources. Classical I/O resources must be available to enable user interactions with a Folk-node. A screen, a keyboard (or a touch screen), a microphone and a speaker are obvious I/O elements. They have however different form-factor and cost/energy requirements. Fortunately, most user interactions will happen with local workers and thus can make use of their own device's I/O resources. In addition, illiterate users will probably underuse the keyboard or the screen. Such I/O resources can thus be located on shared devices without a significant impact on the platform usability, while keeping a speaker and microphone on the Folk-node would allow basic users' interactions autonomously.

5.1.2 Energy considerations and concrete Folk-nodes

In an infrastructure-less context, the electrical power network is limited and leads to consider energy as a particularly scarce resource. To this respect, let us first consider two extreme designs for the Folk-nodes: passive and active ones (see the top of Figure 3).

The ***passive*** Folk-node embeds only mandatory resources (i.e., related to storage and security) and has no internal battery, and thus cannot be used without a shared device. It achieves the lowest cost at the expense of needing a shared device to be used. The intrinsic characteristics of a passive Folk-node are its low cost and robustness, inherited from its simplicity. It is very similar to a smart token product, provided by Gemalto, that we used in a field experiment [2, 3, 5]. This kind of smart token is available for a few dollars, in a SIM card form factor (plugged in a USB key casing), and simply needs to be extended with a fingerprint reader, in a way similar to traditional secure USB keys.

The ***active*** Folk-node is more complex, since it embeds a battery and the required means to charge it and power a wireless communication element (Bluetooth, led-light communications, ...). It is also equipped with I/O resources to allow basic users' interactions autonomously. Typically, we keep a speaker and microphone, given their small form factor, robustness and low-cost/low-energy profiles. Thus, active Folk-nodes provide certain functionalities to end-users even without being connected to a shared terminal, and they can communicate (typically, they can exchange messages wirelessly with surrounding active folk-nodes).

Capitalizing on various energy harvesting technologies (Piezo-electric, solar, etc.), other intermediate designs would make sense, leading to Folk-nodes being active intermittently, then called ***partially active*** Folk-node. For example, a Folk-node could be endowed with piezoelectric energy harvesting technologies and could become active at the will of its owner, a mechanical interaction with the equipment being needed to power the device for short durations.

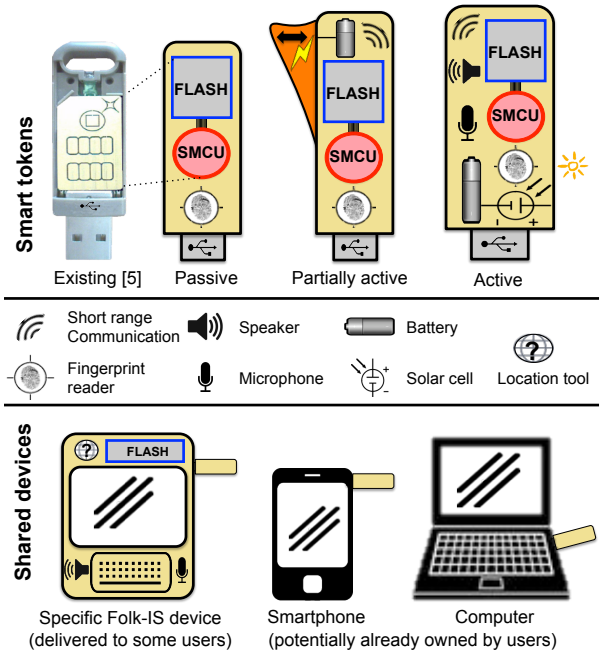


Figure 3. Folk-IS architecture elements

5.2 Shared Devices

Any hardware resource mentioned in the previous section and not present in Folk-nodes should be available in shared devices, that is: (1) for the passive Folk-nodes, all the I/O resources (screen, keyboard, speaker and microphone), some flash memory (for durability and communication purposes), and location mechanisms (for helping routing messages); (2) for the active Folk-nodes, a keyboard, a screen, and a location mechanism.

The bottom of Figure 3 shows three possible instances of shared devices. The specific Folk-IS device (on the left) is designed especially for the Folk-IS architecture. The idea is to increase the robustness of the deployed solution by providing a closed architecture, i.e., a device with no possibility to install any software or hardware components different from those developed and validated by Folk-IS (note that this does not mean proprietary hardware or software). An alternative is to make use of the devices that local workers potentially already own (e.g., a smartphone

or a computer, as shown on the right of Figure 3) in order to further reduce the overall cost of the Folk-IS architecture. Indeed, the required shared resources are generally present in these devices. If a device misses some resource (e.g., it has no location facilities, or has a small available storage), then the service will be locally degraded, but this does not hinder the global architecture (typically, the estimated location will be more imprecise, the durability will be provided elsewhere, etc.).

5.3 Concrete Architectures and Deployment Scenarios

Based on the discussion above, different combinations of Folk-nodes / shared devices and different deployment scenarios can be envisioned. The initial deployment of Folk-IS should be based on the most simple and robust combination, in order to minimize the risks of a too-open or too-complex architecture. Thus, we first consider the combination passive Folk-nodes / specific Folk-IS shared device, whose main advantages are robustness and low cost (a few dollars for the Folk-node, less than US$100 for the shared device, with the cost of shared devices having a small impact on the cost of the overall solution).

Then, more expensive partially active or active Folk-nodes (of a few tens of dollars) could be progressively acquired and distributed to unequipped individuals requiring a lower latency of communications or wishing to access their Folk-nodes without the need of a shared device (e.g., by adding a microphone or speaker).

An orthogonal direction for enhancing the initial deployment is to open the Folk-IS solution to existing devices potentially owned by users or local workers. This means allowing Folk-nodes to connect to a larger set of devices (hardware adaptors may be needed to enable physical connections) and to upload applications (retrieved through the Folk-Net) to these devices. Such open architecture is more complex due to device heterogeneity but it brings many benefits: (1) it lowers the overall cost of the Folk-IS architecture; (2) it increases the number of potential terminals, thereby increasing data availability and reducing nrtwork latency; and (3) it allows third parties to develop interesting user-centric applications, thus increasing the benefit to the users.

These two directions (i.e., more active Folk-nodes and open Folk-IS) are not exclusive. We believe that a final Folk-IS architecture will indeed include passive Folk-nodes, active ones, specific Folk-IS shared devices, and existing devices, their proportion being highly dependent on the context.

6. DATABASES CHALLENGES

Several challenges of Folk-IS are at the crossroad of Computer Science, Economics, and Social Science. A first one is to formalize and quantify the impact of different Folk-IS configurations on e-inclusion, and more generally on social and economic progresses in LDCs. A second important challenge is linked to the study of specific human-computer interactions, made complex by the fact that some users might be illiterate. More generally, low level of education, lack of training, specific beliefs and norms of potential users will impose to study adapted Folk-IS devices interfaces and usages. This section however focuses on database related challenges.

6.1 Co-design of the Embedded Data Management Engine

A primary role of a Folk-node is to ensure secure storage and sharing of all data forming the holder's *digital environment*: user authentication, secure data storage, query evaluation and access control enforcement. The fingerprint reader enables authenticating the requester (e.g., a nurse) at each access, so that privileges can be properly associated with him/her in order to protect data confidentiality and integrity. In a data management system, fine-grained privileges granted to subjects on objects are usually identified by expressions (i.e., queries) over the data. These queries determine the information that can be accessed by the subject (other data remain hidden). Hence, answering accurately the Folk-IS privacy and security requirement leads to embed a data management engine within the Folk-node, in charge of evaluating queries over the data, checking the integrity of any data involved in the computation, and delivering the result to the grantee subject. When considering a relational database, the queries are formulated using the relational algebra (involving selection, projection, joins and aggregation operators). Whenever documents are managed, queries are usually defined using expressions over tags/labels associated to the documents.

Evaluating queries within a Folk-node, even expressed using simple expressions over tags, is very challenging due to the strong and conflicting inherent hardware constraints of smart tokens. Whatever their exact characteristics, low cost and low energy, embedded devices are always endowed with: (1) a small amount of RAM (because of its low density and its high relative impact on the overall cost of the device); and (2) NAND flash memory (the most adopted persistent memory for embedded devices, due to its high density and high robustness), which by nature badly supports random writes. In addition, in the context of the Folk-node, the raw data require to be (3) protected using cryptographic features to ensure the confidentiality and integrity of the stored data. These constraints lead to contradictory objectives: executing queries with acceptable performance with a tiny RAM entails indexing massively the embedded database in NAND Flash, while index updates generate fine-grained random writes, and then unacceptable NAND flash write costs and cryptographic overhead.

These statements make any state-of-the-art solution impractical, and call for the design of brand-new data management techniques. Existing embedded DBMSs (e.g., SQLite, BerkeleyDB) and light versions of popular DBMSs

(e.g., DB2 Everyplace, Oracle Database Mobile Server) target devices far more powerful than smart tokens (like smart phones or tablets), and do not tackle the abovementioned constraints. Most of the recent proposals addressing the problem of storage and indexing in Flash, like [1, 24], adapt the traditional B+-Tree to Flash memory by relying on a Flash-resident log to delay the index updates. When the log is large enough, the updates are committed into the B+-Tree in batch mode, to amortize the Flash write cost. The log must be indexed in RAM to ensure performance. To amortize the write cost by large factors, the log is seldom committed, leading to a higher RAM consumption. Conversely, limiting the RAM size means increasing the commit frequency, thus generating more random writes. Under strong RAM limitations, the commit frequency becomes de facto very high and the gain on random writes vanishes. Flash-aware implementation of key-value stores [39, 25] also use a log structure in Flash to store key-value pairs and exploit sequential writes. But again, an index must be maintained in RAM, with a relatively large size (~1B per key-value pair) to obtain acceptable performances.

Recent proposals more specifically handle the smart token constraints. Microsearch [38] is a search engine designed for sensor nodes, answering full text search queries in a pure pipeline fashion, thus fulfilling the tiny RAM constraint. MiloDB [6] is a database machine for future generations of SIM cards with large storage capacities (a NAND Flash chip is superposed to the secure microcontroller of the SIM card). The database is organized into sequential *Log Containers* in NAND Flash, supports a massively indexed database while avoiding random writes by definition, and is able to compute SQL star join queries within the microcontroller. These proposals demonstrate the growing interest of storing and querying large amounts of data in smart tokens, but remain restricted to particular data models and specific types of smart tokens.

In Folk-IS, a much broader scope in terms of data models and hardware components have to be investigated to co-design and implement the best storage, indexing and query engine considering a triple objective: reducing the overall cost of the device, reducing its energy consumption, and offering a high level of flexibility. To this end, other hardware choices will have to be envisioned. For example, a combination of a very low cost secure (tamper-resistant) microcontroller coupled with a regular microcontroller could be preferred. It would provide the expected level of security, would be more flexible, and would consume less energy (regular microcontrollers have a larger RAM size with respect to tamper-resistant ones, which increases performance and thus reduces energy consumption). Regarding persistent memory (NAND Flash), a micro-SD card would undoubtedly offer better flexibility and lower cost than a raw Flash die: the card could be replaced at will by a cheaper, larger or more efficient one, and it could also be kept when replacing a deficient device. Supporting micro-SD cards has a strong impact on the software design since it leads to access the NAND Flash through an FTL. An FTL is a software layer, proprietary, and potentially different for each micro-SD card, leading to different behaviors in terms of access time, read/write performance ratios, number of sequential data files that can be written sequentially, etc. Designing data management techniques for such a variable environment leads to propose adaptive techniques: frequency of index refreshment would have to adapt to the performance of the micro-SD card, as well as many parameters regarding the storage and indexing structures.

6.2 Folk-Net Routing and Mobility Prediction

The Folk-Net routing protocol aims at delivering the data exchanged among Folk-nodes, workers' devices, and remote servers. The communication can be established either directly between active Folk-nodes (i.e., both equipped with communication facilities) or indirectly through a shared device (e.g., encrypted messages are exchanged as passive Folk-nodes successively connect to the same device, acting as a relay at a clinic, a school, etc.). Moreover, some humans may play the role of "netmen", gathering data to relay ("letters") and physically carrying them to other shared devices (or "intermediate delivery points") for further distribution. Folk-Net can be homogeneous (i.e., exclusively composed of active or passive Folk-nodes) or hybrid, in which case network latency and throughput will depend on the proportion of each type of node and of netmen. Folk-Net relies on *opportunistic communications* [19] by nature and poses a set of challenges in terms of routing, mobility prediction and energy consumption.

Routing protocols

The choice of the best candidate(s) to carry the encrypted data towards their destination is a primary concern to avoid flooding the human network. Many routing protocols have been proposed for wireless networks. *OLSR* and *AODV*, designed for *MANETs (Mobile Ad hoc Networks)* [23], are disqualified in the Folk-IS context due to the great amount of messages they require. *ZigBee* cannot be considered either. Indeed, it relies on a routing tree and thus is not reliable against node mobility. Geographic routing protocols are more interesting candidates as they do not need to maintain routing tables and work nearly stateless [32]. However, they rely on the following assumptions: (1) the location of the packets' recipient is known, and (2) every node capable of routing information knows its own position and trajectory. Both assumptions cannot be assumed in the Folk-Net context, motivating new research efforts.

Folk-IS also shares some similarities with *VANETs (Vehicular Ad Hoc Networks)* [18]. Whereas some features differ significantly (e.g., the underlying communication standard, the communication range, the speed and power of nodes, etc.), both environments share the same lack of infrastructure and dynamic topology. Hence, we believe

that protocols designed in the VANET context, like *carry-and-forward* [16] or *content-based dissemination* [12], could be adapted to Folk-IS. Carry-and-forward protocols are well suited to sparse networks. They exploit some nodes as "data mules" to carry the information and deliver it to other nodes encountered along their trip. On the contrary, content-based dissemination protocols limit the number of nodes relaying the message in dense networks. In the Folk-IS context, carry-and-forward protocols could be investigated to deliver messages (e.g., a query result), whereas content-based dissemination could be considered to efficiently disseminate a query to the nodes holding potential results.

Mobility prediction and profiling

As commented in Section 5.1.1, no exact GPS location can be assumed for the Folk-IS nodes. Instead, their location must be approximated from interactions with a small subset of localized Folk-nodes, either shared devices or fixed nodes. Besides, active Folk-nodes can also compute an approximate location based on the position of their neighbors using techniques based on *RSSI (Received Signal Strength Indicator)*.

Approximate locations can be combined with mobility prediction techniques to predict the future movements of Folk-nodes. Many mobility prediction techniques only rely on the history of the user's mobility patterns and thus are insensitive to the changes of the user behavior. Other techniques are based on both the history of the mobility patterns and formal models (e.g., [26]). In [7, 8], a new approach is investigated combining the notions of local and global profiles. While local profiles capture the recurrent behaviors of each user (like going to the office every day at the same hour and using, more likely, the same paths), global profiles capture a set of frequently-followed paths by a certain percentage of users (main roads, trains, etc.). This approach achieves a much better prediction successful rate and could lead to great delivery and energy performance in Folk-IS georouting. Local profiles could be maintained in each Folk-IS node for privacy concern while global profiles could be anonymized and hosted in shared devices during a certain period of time.

Routing with mobility prediction

New routing protocols mixing geolocation approximation, mobility prediction strategies and social interactions (e.g., [17, 41]) definitely deserve to be studied to select the best "data mules" for each message. In relation to this topic, it is worth mentioning that the problem of detecting and tracking "familiar strangers" (unknown people that one may frequently encounter during his/her everyday activities) has attracted the attention of research, due to the interest in understanding human behavior and diffusion processes [37]. This emphasizes the interest of considering the social behavior in the design of opportunistic routing protocols. As another example, [22] distinguishes between *stable nodes* that have frequent contacts with others within the same community (chosen as *community coordinators* for searching information within the community) and *highly mobile nodes* that frequently visit other communities (chosen as *community ambassadors* for searching information in other communities). These concepts could be applied similarly in the context of Folk-IS.

Energy saving

Active Folk-nodes have a limited autonomy in terms of energy, have to detect neighbors at low cost and form an ad hoc network whose topology and density may change over time (e.g., the network may be quite stable and dense in a village, quite sparse elsewhere, a dynamic neighborhood may become stable when owners of Folk-nodes enter the same bus, etc.). Every task should thus integrate the energy consumption. This can be done through range adjustment when communicating and set up of duty cycling. The former operation allows the reduction of the energy spent for sending (the message is sent to a closer node at a lower power) and for overhearing (it will reach less nodes), and generates less interferences. It has to be included in the georouting protocol (like in [27]). The latter operation is based on the node activity and its neighborhood. If a node does not need to communicate and there are enough active nodes in its neighborhood to ensure the relay of other nodes' messages, it can simply sleep and save energy. This operation can also be enhanced through the use of prediction models.

6.3 Application, Data Model and Access Control Deployment

As discussed in Section 4, Folk-IS enables many important applications, like email services, healthcare, and e-administration, some of them novel by their purpose (e.g., sharing pieces of a cultural heritage) or by their target (e.g., epidemiological studies on populations unreachable so far). The infrastructure-less context introduces new challenges with respect to application deployment, unified data modeling, identity verification, access control, and user's consent. Typically, data standardization, central application stores or central authorities identifying people, delivering certificates and enforcing access control rules on central servers, cannot be assumed. Conversely, a salient feature of Folk-IS is the ability to push the control at the edge of the network, that is to say (1) within each tamper-resistant Smart Token, and (2) through face-to-face interactions between users, implementing a de-facto user's consent. This may enable the reestablishment of local spheres of trust (e.g., NGO producing applications, data models, home-made identification information, access control policies, and pushing them at the Folk-node level through Folk-Net). This paves the way to a semi-decentralized way of managing and deploying applications, each Folk-node guaranteeing that (i) data produced by a given organization will not leak outside that organization, and (ii) the data owners' privacy is always respected.

6.4 Evaluation of Global Queries on a Population of Folk-nodes

Traditional techniques used to process queries in distributed databases or in peer-to-peer networks are not suitable, as they assume a good knowledge about the location of the data items in the network. Here the data items will be dynamically distributed over a network of nodes that may be accessible or not at a certain moment, and the carrier of a replica of a data item may change at any time. Moreover, we cannot assume that all the data items relevant to a given query will always be available and retrievable in a reasonable time period. So, we should adopt the open-world assumption, admit the possibility of approximate answers, decide how to identify the relevant data sources without overloading the network (e.g., based on spatial conditions), how to route queries and results to their recipients (while preserving autonomy) using geographic routing and the physical mobility of nodes, and how to determine when a query and its associated routing tasks have to be finished. Moreover, new types of queries could be of interest in this context, requiring new query processing techniques; for example, reachability queries [36] could be used to study the possibility of propagation of a disease by analyzing past trajectories.

6.5 Structure and Calibration of the Folk-IS Architecture

Folk-IS is built on a large number of highly-secure but seldom-available Folk-nodes (active or passive) on the one side, and on less secure but more accessible and powerful shared devices on the other side. This unusual asymmetric architecture requires deeply rethinking the overall organization of an information system. This means distributing software resources on the hardware elements of the architecture, such that local and global applications can run and provide results with acceptable performance, security and resiliency. This also means associating different responsibilities to different devices (e.g., *super-nodes*), associating specific roles to humans (e.g., *netmen* to reduce latency), and designing new distributed protocols to implement global functionalities like query evaluation or data durability. For the latter, data replication is required because Folk-nodes may be lost, stolen, broken, etc. Data could be replicated based on the mobility profiles of the users. Confidentiality of replicas can be achieved through encryption, leading to the problem of choosing the most adequate set of Folk-nodes to hold a copy of the keys or of key shares (e.g., based on trust or similar mobility profiles). Finally, new simulation models are needed to calibrate IT resources according to the target applications and the local habits of residents. The objective is to determine suitable network topologies (given a certain density of Folk-nodes, communication frequency, etc.) with viable incremental deployments. Those simulation models will also help quantifying the expected gain in terms of social and economic sustainability, which is key for their adoption in the long term.

7. CONCLUSION

As mentioned in [9], time has come for research works, not only commercial initiatives, addressing the expectations of 80% of the population living outside developed countries. The *Folk-enabled Information System (Folk-IS)* paradigm combines low-cost secure devices, embedded software components and opportunistic communications, to meet fundamental requirements of LDCs. The promise of the solution is to fulfill the requirements stated in the introduction:

- *Privacy protection*: Folk-IS inherits its security and privacy protection from the tamper-resistance of each token and from the delegation of the access control and authentication tasks to each Folk-node, at the user's side. Moreover, any user must authenticate to the Folk-node, which securely records any performed action in an audit log, making the complete system accountable. The holder himself may be authenticated by the Folk-node, and may not have all the privileges on his own records.
- *Immediate personal benefit*: Folk-IS makes the Folk-node of each individual play the role of a passport for a wide range of critical applications: healthcare, e-administration, education, market rates for farmers, e-mails, etc. In parallel, governments and NGOs benefit from a new channel to pull/push information from/towards communities previously unreachable.
- *Self-sufficiency*: Folk-IS is infrastructure-less by construction. The complete system (data storage and network facilities) progressively and automatically deploys itself as Folk-nodes are distributed to the participants, with the ability to benefit from future infrastructure improvements.
- *Very low deployment cost*: Folk-nodes are hosted in low-cost smart tokens and the global cost of the system is directly proportional to the scale of the targeted population. In addition Folk-IS is by construction a highly-redundant and robust system that does not require any central administration. The maintenance and performance improvement of the system can be a source of empowerment with new local jobs (e.g., people renting terminals or acting as postmen), crucial to make the solution sustainable.

Hence, we expect that this paper could pave the way to exciting future works for our research community as well as to fruitfull experiments in the field as soon as the major challenges are tackled.

8. ACKNOWLEDGMENTS

We warmly thank Léo Dayan, scientific director of APREIS NGO and of the Nomadic World University for Sustainable Development, as well as Pascale Pollack, director of E-NEXUS NGO, for precious discussions and feedback on the requirements of IT solutions for LDCs.

9. REFERENCES

[1] Agrawal, D., Ganesan, D., Sitaraman, R., Diao, Y., and Singh, S. 2009. Lazy-adaptive tree: An optimized index structure for flash devices. In *PVLDB*, 2(1).

[2] Allard, T., Anciaux, N., Bouganim, L., Pucheral, P., and Thion, R. 2009. Trustworthiness of Pervasive Healthcare Folders. *Pervasive and Smart Technologies for Healthcare*, Information Science Reference.

[3] Anciaux, N., Bonnet, P., Bouganim, L., Nguyen, B., Popa, I. S., and Pucheral, P. 2013. Trusted Cells: A Sea Change for Personal Data Services. In *CIDR*.

[4] Anciaux, N., Bouganim, L., Delot, T., Ilarri, S., Kloul, L., Mitton, N., and Pucheral, P. 2014. Folk-IS: Opportunistic Data Services in Least Developed Countries. In *PVLDB*, 7. http://www.vldb.org/pvldb/vol7/p425-anciaux.pdf

[5] Anciaux, N., Bouganim, L., Guo, Y., Pucheral, P., Vandewalle, J.-J., and Yin, S. 2010. Pluggable Personal Data Servers. In *ACM SIGMOD*.

[6] Anciaux, N., Bouganim, L., Pucheral, P., Guo, Y., Le Folgoc, L., and Yin, S. 2013. MILo-DB: A personal, secure and portable database machine. *Distributed and Parallel Databases*.

[7] Barth, D., Bellahsene, S., and Kloul, L. 2011. Mobility Prediction Using Mobile User Profiles. In *IEEE MASCOTS*.

[8] Barth, D., Bellahsene, S., and Kloul, L. 2012. Combining Local and Global Profiles for Mobility Prediction in LTE. In *ACM MSWiM*.

[9] Brewer, E., Demmer, M., Du, B., Ho, M., Kam, M., Nedevschi, S., Pal, J., Patra, R., Surana, S., and Fall, K. 2005. The Case for Technology in Developing Regions. IEEE *Computer* 38(6).

[10] Brown, D., Marsden, G., and Rivett, U. 2012. WATER Alert!: Using Mobile Phones to Improve Community Perspective on Drinking Water Quality in South Africa. In *ACM ICTD*.

[11] Bursky, D. 2012. Secure Microcontrollers Keep Data Safe. PRN Engineering Services, http://tinyurl.com/secureMCU

[12] Cenerario, N., Delot, T., and Ilarri, S. 2011. A Content-Based Dissemination Protocol for VANETs: Exploiting the Encounter Probability. *IEEE Trans. on Intelligent Transportation Systems*, 12(3).

[13] Chaudhri, R., Borriello, G., and Anderson, R. J. 2012. Monitoring Vaccine Cold Chains in Developing Countries. *IEEE Pervasive Computing*, 11(3).

[14] Coceres, R., Belding, E. M., Parikh, T. S., and Subramanian, L. 2012. Information and Communication Technologies for Development - Guest Editors' Introduction. IEEE *Pervasive Computing*, 11(3).

[15] Fall, K. R. 2003. A Delay-Tolerant Network Architecture for Challenged Internets. In *ACM SIGCOMM*.

[16] Fenu, G. and Nitti, M. 2011. Strategies to Carry and Forward Packets in VANET. In *Digital Information and Communication Technology and Its Applications, Communications in Computer and Information Science,* 166.

[17] Gao, L., Li, M., Bonti, A., Zhou, W., and Yu, S. 2013. Multidimensional Routing Protocol in Human-Associated Delay-Tolerant Networks. *IEEE Transactions on Mobile Computing*, 12(11).

[18] Hartenstein H. and Laberteaux, K. 2010. *Vehicular Applications and Inter-Networking Technologies*. John Wiley & Sons.

[19] Huang, C.M., Lan, K.C., and Tsai, C.Z. 2008. A Survey of Opportunistic Networks. In *Advanced Information Networking and Applications-Workshops*.

[20] ITU. 2011. The Role of ICT in Advancing Growth in Least Developed Countries – Trends, Challenges and Opportunities. http://www.itu.int/pub/D-LDC-ICTLDC.2011

[21] Jumira, O., Wolhuter, R., and Mitton, N. 2012. Prediction Model For Solar Energy Harvesting Wireless Sensors. In *Africomm*.

[22] Kang, C., Haiying, S., and Haibo, Z. 2014. Leveraging Social Networks for P2P Content-Based File Sharing in Disconnected MANETs. *IEEE Transactions on Mobile Computing*, 13(2).

[23] Kiess, W. and Mauve, M. 2007. A survey on real-world implementations of mobile ad-hoc networks. *Ad Hoc Networks*, 5(3).

[24] Li, Y., He, B., Yang, R. J., Luo, Q., and Yi, K. 2010. Tree indexing on solid state drives. In *PVLDB*, 3(1-2).

[25] Lim, H., Fan, B., Andersen, D. G., and Kaminsky, M. 2011. SILT: A memory-efficient, high-performance key-value store. In *ACM SOSP*.

[26] Liu, G. and Maguire, G. 1996. A Class of Mobile Motion Prediction Algorithms for Wireless Mobile Computing and Communication. *Mobile Networks and Applications*, 1(2).

[27] Mitton, N., Simplot, D., and Zheng, J. 2013. Guaranteed Delivery in k-Anycast Routing in Multi-Sink Wireless Networks ICST Mobile Communications and Applications. In *ICST*.

[28] Murugesan, S. 2013. Mobile Apps in Africa. *IT Professional*, 15(5).

[29] Ndou, V. 2004. E-Government for Developing Countries: Opportunities and Challenges. *The Electronic Journal of Information Systems in Developing Countries*, 18.

[30] Pentland, A., Fletcher, R., and Hasson, A. 2004. DakNet: Rethinking Connectivity in Developing Nations. IEEE *Computer*, 37(1).

[31] Rossi, G., Murugesan, S., and Godbole, N. 2012. IT in Emerging Markets. *IT Professional*, 14(4).

[32] Rührup, S. 2009. Theory and Practice of Geographic Routing. *Ad Hoc and Sensor Wireless Networks: Architectures, Algorithms and Protocols*.

[33] Sahilu, H., Villafiorita, A., Weldemariam, K., Belachew, M., and Zewge, A. 2012. Designing Distributed Agricultural Information Services for Developing Countries. In *ACM DEV*.

[34] Singh, P., Singh, A., Naik, V., and Lal, S. 2012. CVDMagic: A Mobile Based Study for CVD Risk Detection in Rural India. In *ACM ICTD*.

[35] Seth, A., Kroeker, D., Zaharia, M. A., Guo, S., and Keshav, S. 2006. Low-Cost Communication for Rural Internet Kiosks Using Mechanical Backhaul. In *ACM MOBICOM*.

[36] Shirani-Mehr, H., Kashani, F. B., and Shahabi, C. 2012. Efficient Reachability Query Evaluation in Large Spatiotemporal Contact Datasets. In *PVLDB*, 5(9).

[37] Suna, L., Axhausena, K. W., Lee, D., and Huanga, X. 2013. Understanding metropolitan patterns of daily encounters. *Proceedings of the National Academy of Sciences,* 110(34).

[38] Tan, C.C., Sheng, B. , Wang, H., and Li, Q. 2010. Microsearch: A search engine for embedded devices used in pervasive computing. ACM *Transactions on Embedded Computing Systems*, 9(4).

[39] Vo, H. T., Wang, S., Agrawal, D., Chen, G., and Ooi, B. C. 2012. LogBase: A scalable log-structured database system in the cloud. In *PVLDB*, 5(10).

[40] VOICE project. VOIce-based Community-cEntric mobile Services for social development (VOICE). ICT-2009.9.1. http://www.mvoices.eu/

[41] Wang, Y., Yang, W.S., and Wu, J. 2013. Analysis of a Hypercube-Based Social Feature Multipath Routing in Delay Tolerant Networks. IEEE *TPDS*, 24(9).

State-of-the-art in String Similarity Search and Join

Sebastian Wandelt
Knowledge Management in Bioinformatics, HU Berlin, Berlin, Germany

Dong Deng
Tsinghua University, Beijing, China

Stefan Gerdjikov
FMI Sofia University, Sofia, Bulgaria

Shashwat Mishra
Special Interest Group in Data, IIT Kanpur, Kanpur, India

Petar Mitankin
IICT Bulgarian Academy of Sciences, FMI Sofia University, Sofia, Bulgaria

Manish Patil
Louisiana State University, Louisiana, USA

Enrico Siragusa
Algorithmic Bioinformatics, FU Berlin, Berlin, Germany

Alexander Tiskin
Department of Computer Science, University of Warwick, United Kingdom

Wei Wang
University of New South Wales, New South Wales, Australia

Jiaying Wang
Northeastern University Shenyang, China

Ulf Leser
Knowledge Management in Bioinformatics, HU Berlin, Berlin, Germany

ABSTRACT

String similarity search and its variants are fundamental problems with many applications in areas such as data integration, data quality, computational linguistics, or bioinformatics. A plethora of methods have been developed over the last decades. Obtaining an overview of the state-of-the-art in this field is difficult, as results are published in various domains without much cross-talk, papers use different data sets and often study subtle variations of the core problems, and the sheer number of proposed methods exceeds the capacity of a single research group. In this paper, we report on the results of the probably largest benchmark ever performed in this field. To overcome the resource bottleneck, we organized the benchmark as an international competition, a workshop at EDBT/ICDT 2013. Various teams from different fields and from all over the world developed or tuned programs for two crisply defined problems. All algorithms were evaluated by an external group on two machines. Altogether, we compared 14 different programs on two string matching problems (k-approximate search and k-approximate join) using data sets of increasing sizes and with different characteristics from two different domains. We compare programs primarily by wall clock time, but also provide results on memory usage, indexing time, batch query effects and scalability in terms of CPU cores. Results were averaged over several runs and confirmed on a second, different hardware platform. A particularly interesting observation is that disciplines can and should learn more from each other, with the three best teams rooting in computational linguistics, databases, and bioinformatics, respectively.

Keywords

String search, String join, Scalability, Comparison

1. INTRODUCTION

Approximate search and join operations over large collections of strings are fundamental problems with many applications. String similarity search is used, for instance, to identify entities in natural language texts [29], to align DNA sequences produced in modern DNA sequencing with substrings of a reference genome [16, 17], or to perform pattern matching in time series represented as sequences of symbols [10]. String similarity joins are building blocks in the detection of duplicate Web pages [13], in collaborative filtering [2], or in entity reconciliation [7]. Research in this field dates back to the early days of computer science and the area is still highly active today. Literally hundreds of methods have been proposed.

For string similarity search and join, fundamental techniques include seed-and-extend methods (turning similarity search into an exact search problem of smaller strings, e.g. All-Pairs [2], ED-Join [31], and PPJoin [32]), partitioning techniques (e.g. Pass-Join [15], NGPP [29], and PartEnum [1]), prefix-filtering methods (e.g. Trie-Join [8] and PEARL [23]), and other methods (e.g. M-Tree [5], LSH [12], SSI [9], and FASTSS [25]).

Research in the field has been carried out in various

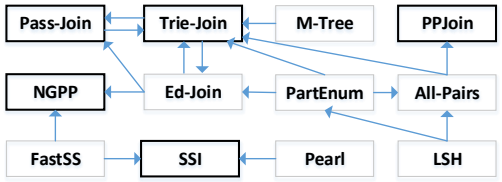


Figure 1: Recent work on string similarity search and join with edit distance constraints. An edge from method M1 to M2 visualizes that M2 was found to be superior to M1. Marked approaches are non-dominated, i.e. not reported strictly slower than any other method.

scientific disciplines, the most important ones probably being algorithms for pattern matching, computational linguistics, bioinformatics, and database / data integration. There are subtle differences between the problems being studied, for instance varying in the concrete similarity measure (edit distance, Jaccard, Hamming etc.), the type of string comparisons (global or local alignment, approximate substring search etc.), the amount of indexing being allowed (online in the queries and/or the database). Methods often are tuned for specific ranges of allowed error thresholds or query lengths, specific hardware properties, specific alphabet sizes, or specific distributions of errors. Though newly published methods mostly compare to some prior works, selection of these works is often suboptimal and comparisons are carried out on different data sets; data sets all too often are not made publicly available, which means that results are not reproducible. In Figure 1, we show existing evaluation results for the most relevant work on string similarity search/join with edit distance constraints. As a consequence of the heterogeneity of approaches and problems, the lack of common benchmarks, and the dispersal of research in different communities, today it is hardly possible to choose the best algorithm for a given problem.

In this work, we report on the (to the best of our knowledge) most comprehensive benchmark in two specific string similarity match problems to date: k-approximate search and k-approximate join (with k as an edit distance threshold; see below for exact definition). We organized this benchmark using a rather uncommon approach: The *International competition on Scalable String Similarity Search and Join (S4)*[1] held as a workshop in conjunction with EDBT/ICDT 2013. We made an open, world-wide call for contributions and provided crisp task definitions, a loose hardware specification and example data. Nine teams from different communities participated, including databases, natural language processing, and bioinformatics. Thus, for the first time, we were able to evaluate different highly competitive implementations of search and join algorithms on the same evaluation platform (hardware, operating system, and datasets). In addition, organizing the benchmark as a competition, where teams developed and tuned their own programs independently, allowed us to compare original and optimized programs instead of unverified and potentially unoptimized re-implementations.

All submitted programs were tested on different datasets (DNA sequences and geographical names) of different sizes (a few KB up to a few GB) with different error thresholds (edit distance k between 0 and 16). We performed experiments in two different hardware settings: a commodity PC with 8 cores/64 GB RAM and a server with 80 cores/1 TB RAM. For the top performing programs we performed additional analyses with different number of threads to investigate the possibility to parallelize algorithms. Furthermore, we compared submissions with a number of publicly available algorithms of groups that did not participate, showing that the best ranked programs from our competition are several orders of magnitude faster. Altogether, 14 different programs or configurations were evaluated with differences in runtime of factors of more than 1000 between the fastest and slowest program. We are confident that our results give a fairly representative picture of the state-of-the-art in string similarity search. The evaluation of all programs and datasets took more than three months of raw processing time.

The wealth of experiments we performed and the significant number of programs we compared allows us to draw several interesting conclusions about scalability, batch procession effects, index size, main memory usage, and the possibility to parallelize techniques.

The purpose of this paper is not only to report on efficiency of algorithms in string similarity search, but also to promote competitions as an effective, joyful, and comprehensive means to evaluate the state-of-the-art on a given problem. Actually, competitions are quite common in many related disciplines, such as information extraction, information retrieval, data analysis etc., but, to our knowledge, represent a novel approach within the database community. The only comparable effort we are aware of is the SIGMOD programming contest. However, it only addresses graduates and the focus is more on education (and probably recruitment). In contrast, the main purpose of S4 was to identify the fastest methods available. Clearly, the most critical point for a competition like S4 is the measurement of wall clock time, which is dependent on the concrete implementation and the machine being used for measurements, instead of quality metrics independent of the concrete implementation and evaluation environment (such as precision or recall). We will expand on this issue in Section 6.

The remainder of this paper is organized as follows. We describe the concrete problems we benchmarked, the datasets, and the benchmarking methodology in Section 2. All submitted methods are briefly presented in Section 3. Evaluation results for approximate string searching are presented in Section 4 and for approximate string join in Section 5. In Section 6, we discuss the results of the competition and compare results to three external programs, Flamingo [3], Pearl [23], and SSI [9], which were evaluated after our competition was finished. The paper is concluded with Section 7.

[1] `http://www2.informatik.hu-berlin.de/~wandelt/searchjoincompetition2013/`

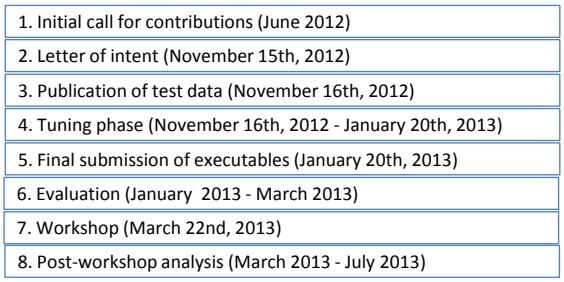


Figure 2: Phases of the competition

2. BACKGROUND

We define the problems of approximate string searching and approximate string join. Our competition and evaluation methodology is introduced together with a description of datasets and evaluation environments.

2.1 Formal problem statement

DEFINITION 1 (STRINGS). *A* string *s is a finite sequence of symbols over an alphabet Σ. The length of a string s is denoted by $|s|$ and the substring starting at position i with length n is denoted by $s(i, n)$. We write $s(i)$ as an abbreviation for $s(i, 1)$. All positions in a sequence are zero-based, i.e., the first character of s is $s(0)$.*

As a distance function between two strings we use unweighted edit distance for different error thresholds k.

DEFINITION 2 (STRING SIMILARITY). *Given strings s and t, s is k-*approximately similar to *t, denoted $s \sim_k t$, if and only if s can be transformed into t by at most k edit operations. The edit operations are: replacing one symbol in s, deleting one symbol from s, and inserting one symbol into s.*

We investigate two problems: string similarity search and string similarity join.

DEFINITION 3 (SIMILARITY SEARCH). *Given a collection of strings $S = \{s_1, ..., s_n\}$, a query string q, and an edit distance threshold k, the result of* string similarity search of q in S *is defined as*
$SEARCH(S, q, k) = \{i \mid s_i \in S \wedge s_i \sim_k q\}$.

For instance, given a collection $S = \{ACA, TGA, AC\}$, a query string $q = ACA$, and $k = 1$, the result of string similarity search is $SEARCH(S, q, k) = \{1, 3\}$.

DEFINITION 4 (SIMILARITY (SELF) JOIN). *Given a collection of strings $S = \{s_1, ..., s_n\}$ and an edit distance threshold k, the result of* string similarity self-join of S *is defined as $JOIN(S, k) = \{(i, j) \mid s_i \in S \wedge s_j \in S \wedge s_i \sim_k s_j\}$.*

For instance, the result of a string similarity self-join on data set S from above with $k = 1$ is $JOIN(S, 1) = \{(1, 1), (1, 3), (2, 2), (3, 1), (3, 3)\}$. Note that we explicitly include the reflexive and symmetric closure in our definition. We note that a self-join is comparable to a join between two different sets as we make no assumptions about the a priori average level of similarity of the strings in a set. In the following we will often use the term join instead of self-join.

2.2 Competition and methodology

This competition brought together researchers and practitioners from database research, natural language processing, and bioinformatics. The challenge for all participants was to perform string similarity search and join over unseen data and query sets with varying error thresholds k as fast as possible. The call for the competition was circulated by email through various lists addressing the different areas dealing with string matching, in particular databases, algorithms, computational linguistics, and bioinformatics. We also contacted directly a few dozen researchers known for their contributions to the field. The different phases of the competition are shown in Figure 2.

In total we received initial expressions of interest from 22 teams, out of which 11 teams officially submitted a program. One team failed to hand in a complete paper describing their approach on time, and another group withdrew shortly before the final deadline. Thus, we eventually compared programs from 9 teams (see Table 1). All these teams gathered at a workshop colocates with EDBT/ICDT 2014 in Genoa, where each team presented its approach and the results of our evaluation were discussed. This format led to a workshop in the best sense of the word - as all presentations essentially covered the same problems, talks were highly focused and intensive discussions and exchanges of ideas emerged naturally. We also organized culinary prices for the best teams which were immediately shared with the entire audience.

We succeeded in reaching out to different research communities: two teams have their home in bioinformatics, two in computational linguistics, one in algorithms/ computational complexity, and the remaining four are best described as database groups. Contributions came from four continents and seven countries. At least six teams (Team 1, 3–5, 7–9) published highly influential papers on string matching problems before [15, 19, 22, 24, 26, 28, 33], while two teams (Team 2 and Team 6) can be considered as newcomers. As Table 1 shows, the techniques used cover a broad range and thus subsume a large fraction of previous research in k-approximate string matching. Out of the five non-dominated methods in Figure 1, four methods are directly represented by corresponding authors in our competition.

The competition consisted of two tracks:

Track 1: Given a set of strings S, a query string q and an error threshold k, compute $SEARCH(S, q, k)$.

Track 2: Given a set of strings S and an error threshold k, compute $JOIN(S, k)$.

Small subsets of the final evaluation datasets (around 5%) were made available for the contestants for preparation of their submissions. It was announced that these strings are representative for the final evaluation datasets. Furthermore, we announced a description of the evaluation hardware and provided a virtual machine mirroring the software environment used for evaluation. Thus, all teams could develop and tune their programs before submission. Each program was allowed to use any number of threads, with the restriction that the official evaluation environment System 1 (see below) has 8 cores,

Team	Affiliation	General approach	Indexing?	Indexing queries?
1	Tsinghua University, China	Partitioning and pruning [15](Pass-Join, Trie-Join)	yes	no
2	Magdeburg University, Germany	Sequential search	no	no
3	University of Warwick, UK	Bit-parallel LCS computation [26]	no	yes
4	Sofia University, Bulgaria	Directed acyclic word graph [19]	yes	no
5	FU Berlin, Germany	Approximate partitioning [24]	yes/no	yes/no
6	IIT Kanpur, India	Deletion neighborhoods / hashing	yes	no
7	Louisiana State University, USA	Q-gram indexing with filtering	yes	no
8	University of NSW, Australia	Trie-index with filtering [33] (PPJoin,NGPP)	yes	no
9	Northeastern University, China	cache-aware BWT	yes	no

Table 1: Teams which participated in the competition

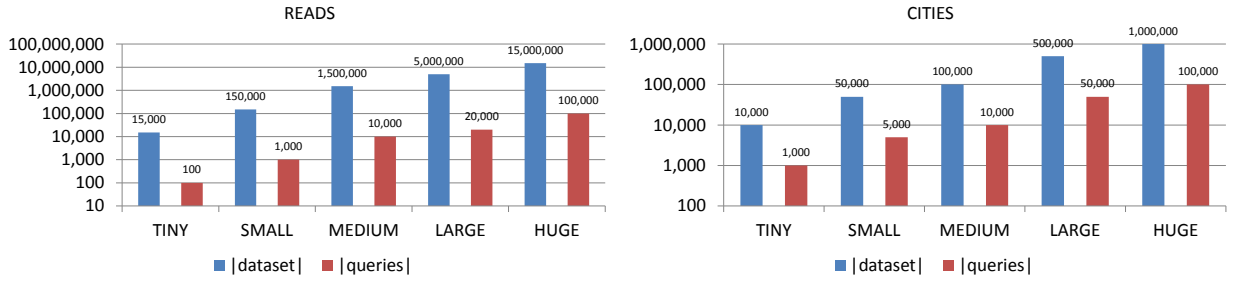


Figure 3: Size of dataset and number of queries used for evaluation (READS and CITIES)

and a maximum of 48 GB of main memory. Details on CPU, clock rate, cache sizes, disks etc. were not provided to prevent hardware specific tuning; note that this implies that further improvements could be possible taking the specific hardware into account [20]. Programs were allowed to have two phases, one for indexing the data set, and one for evaluating a set of queries on the set (or the index). The main evaluation criterion was measured wall clock time. In general, we ranked programs based on average runtime over three independent runs; variations in runtime were very low and are not reported here. If programs ran much longer than most of the competitors, experiments were only performed once. We also measured the indexing time and report it here, but we did not take it into account for ranking.

2.3 Datasets

We used two different types of datasets, for evaluation in both tracks, to cover different alphabets and string lengths. Each type of dataset contains five distinct, highly-similar datasets of increasing size, for evaluating scalability.

READS: These data sets contain reads obtained from a human genome. The data is characterized by a small alphabet (5 symbols) and quite uniform length of strings (around 100 symbols per string).

CITIES: These data sets are based on geographical names taken from World Gazetteer. The data is characterized by a larger alphabet (around 200 symbols) and non-uniform length of strings (5-64).

Considered values for k depend on the dataset. For READS, we announced and used $k \in \{0, 4, 8, 12, 16\}$; for CITIES $k \in \{0, 1, 2, 3, 4\}$. Thus the maximum error rate for READS is around $\frac{1}{6}$ and for CITIES around $\frac{4}{5}$. The size of each dataset and the number of queries for Track 1 are shown in Figure 3. For READS, the number of reads starts with 15,000 (TINY) and ends with 15,000,000 (HUGE). For CITIES, the number of cities starts with 10,000 (TINY) and ends with 1,000,000 (HUGE). For READS and CITIES, the maximum number of queries in HUGE is 100,000.

2.4 Evaluation Environments

After the development phase of the competition, participants submitted their final programs which were evaluated on two different platforms.

System 1: A computer with 8 cores (processor: AMD FX-8320) and 64 GB RAM. The operating system (Fedora Scientific 17 x86_64) was installed on a SSD with 128 GB. The SSD contained the datasets as well as the programs. Each program serialized its results to an external USB 3.0 hard disk with 3 TB. This system was announced beforehand and results for this system were used for ranking.

System 2: A server with 80 cores (processors: Intel Xeon CPU E7 - 4870) and 1 TB RAM. The operating system was openSUSE 12.1 x86_64. All datasets, programs, and serialized results were put on a local hard disk with a total storage capacity of 10 TB. This system was introduced only during evaluation for (a) performing experiments with more cores / memories and for (b) confirming results on a separate hardware with different architecture and CPUs.

Most of the experiments were run on System 1. We have used System 2 only for an extended evaluation, inves-

tigating the scalability with the number of threads (for top performing methods on System 1). In our evaluation below, we will mention explicitly if System 2 was used.

3. METHODS

This section describes the methods used by each team in their submissions to the competition.

3.1 Team 1

PassJoin (Tsinghua University) adopts a partition-based framework for string similarity search and joins. The basic idea is that given two datasets R and S, and an edit distance threshold k, each string in R is split into $k+1$ disjoint segments. For each string in S, PassJoin checks if it contains any substring matching the segments of R. If no, PassJoin prunes the string; otherwise the string and those strings whose segments match the substrings of the string are verified. There are two challenges in the partition-based method. The first one is how to select the substrings. A position-aware substring selection method and a multi-match-aware substring selection method have been proposed. It has been proven the multi-match-aware substring selection method selects the minimum number of substrings. And it is the only way to select the minimum number of substrings when the string length is longer than $2 * k + 1$. The second one is how to verify each candidate pair. PassJoin uses a length-based verification method, an improved early termination technique, and an extension-based verification method.
Team 1 submitted two programs: **Program 1_A** and **Program 1_B**. Both programs of Team 1 were evaluated for both tracks and both datasets.

3.2 Team 2

Team 2 (Magdeburg University) tries to outperform conventional index-searches by a sequential search algorithm. Starting from a naive algorithm for computing edit distances, several optimizations are introduced. Calculation of the edit distance is improved by using length-heuristics. If the computation of a dot matrix cannot be avoided, the program applies several heuristics to prune the search space early. Further optimizations include the use of reference-based semantics over value-based semantics and the use of simple data types. They devise simple scheduling strategies depending on the current workload.
Team 2 submitted only one program: **Program 2_A**, which was evaluated for Track 1 only.

3.3 Team 3

The Waterfall algorithm of Team 3 (University of Warwick) solves the competition challenge without indexing or any other preprocessing of the database strings. First, a reduction of the edit distance problem to the longest common subsequence (LCS) problem between the database string and the query string, both suitably modified, is applied. The strings' LCS score is then computed by a bit-parallel algorithm, based on [6]. This technique is extended so that a database string can be tested simultaneously against multiple query strings, by a subword-parallel technique similar to that of [14], which was further developed in the waterfall algorithm. Due to the self-imposed restriction of not preprocessing the database, the algorithm runs significantly slower than other competitors, which do index the database strings before answering the queries. However, the approach chosen by Team 3 can prove useful in a situation where input preprocessing is not possible. Such a situation occurs e.g. when the string database is replaced by a continuous stream of input strings, each of which needs to be matched against a small set of query strings in real time.
Team 3 submitted only one program: **Program 3_A**, which was evaluated for both tracks and both datasets.

3.4 Team 4

The WallBreaker of Team 4 (Sofia University) is a new sequential algorithm for the similarity search problem in a finite set of words. It reduces and essentially overcomes the wall-effect caused by the redundantly generated false candidates. To achieve this the query is split into smaller subqueries with smaller threshold. This allows to start with an exact match and then extend these exact matches to longer candidates whereas the threshold increases slowly in a stepwise manner. In order to implement this idea in practice two kind of resources are used: (i) a linear space representation of the infixes in the finite set of words that enables a left/right extension of an infix in constant time per character; and (ii) efficient filters, universal Levenshtein automata [18], sychnorised Levenshtein automata [19] and standard Ukkonen filter [27], that prune the unsuccessful candidates as soon as a clear evidence for this occurs. In the index structure information about the possible lengths of longest/shortest left/right possible extensions are encoded. This information is then used as an additional length-filter.
As a result a breaking-the-wall-effect is achieved. In the beginning the WallBreaker considers only small neighborhoods of short words which keeps the searching space modest. Afterwards, while increasing the potential size of the neighborhoods, longer infixes are generated that are much more informative than shorter ones and suppress the searching space for their own sake. For further details the reader is refered to [11], where besides the standard Levenshtein edit-distance also the generalized Levenshtein edit-distance is handled.
Team 4 submitted two programs:
Program 4_A: It uses **16** threads, the additional length-filter, and applies universal Levenshtein automata for thresholds $\leq$ **5**, and synchronised Levenshtein automata for thresholds $\leq$ **3**.
Program 4_B: It uses **16** threads, ignores the additional length-filter, and applies universal Levenshtein automata for thresholds $\leq$ **5**, and synchronised Levenshtein automata for thresholds $\leq$ **3**.
Both programs of Team 4 were evaluated for both tracks and both datasets.

3.5 Team 5

The methods of Team 5 (FU Berlin) are variations of those applied in Masai [24], a tool for mapping high-throughput DNA sequencing data. First an online solution for computing edit distances using a banded version of the Myers bit-vector algorithm [21] is proposed. Team 5 is able to check in time $O\big(\frac{(k+1)(n+|\Sigma|)}{w}\big)$, where w is the CPU word size and Σ the string alphabet, if two strings of length m and n (w.l.o.g. $m < n$) are within edit distance k. Then they propose to index multiple queries in a radix tree and backtrack them into the radix (or suffix) tree of the database. In practice, radix (and suffix) trees are replaced by simpler radix (and suffix) arrays. Multiple backtracking is parallelized with static load balancing and work queues. Finally, as proposed by Navarro and Baeza-Yates [21], a filtering method partitioning queries into approximate seeds is implemented. Such a filtering method combines the previous two methods and works well up to moderate error rates. The programs are implemented in C++ and OpenMP using the SeqAn library.
Team 5 submitted four programs:
Program 5_A: An online algorithm.
Program 5_B: Partitioning with minimum seed length (10 for READS, 4 for CITIES)
Program 5_C: Partitioning with minimum seed length (13 for READS, 5 for CITIES)
Program 5_D: Partitioning with minimum seed length (15 for READS, 6 for CITIES)

3.6 Team 6

The submission of Team 6 (IIT Kanpur) uses deletion neighborhoods [25]. A k-neighborhood is generated for every string $s \in S$. Every string in the k-neighborhood is referred to as a key. The underlying index structure is a hash-table which maintains an inverted index on the keys. In order to circumvent the large space requirement, the program only indexes an L_s-length suffix for each key. Given a query string q and an edit distance threshold k, first the k-neighborhood of q, N_q, is generated. The list corresponding to every key in N_q is obtained from the index structure. A union of these lists is guaranteed to be a superset of the answer set $SEARCH(S, q, k)$. For each string s in the generated candidate list, the program uses a length-threshold aware distance computation to verify s. In a multi-core environment, the program partitions the entire workload into k equal parts and each part is handled by a single, dedicated thread. Team 6's idea is that deletion neighborhoods offer a powerful, selective signature scheme to process edit distance queries. Team 6 only participated in Track 1 of the competition. Further, since deletion neighborhoods are only suited for scenarios with larger alphabet size, Team 6's submission **Program 6_A** was only evaluated on CITIES dataset.

3.7 Team 7

The index structure of Team 7 (Louisiana State University) consists of a generalized suffix tree (GST) and a two-level wavelet tree (WT) on its leaves. The first level WT maintains an array of starting positions of all suffixes of GST. For each leaf of this WT, another WT for the difference between the starting position of the suffix and the string length to which it belongs to is maintained. Given τ, r, Team 7 obtains $\tau + k$ disjoint partitions of r aiming to balance selectivity of count filtering and frequency of partitioned segments. Then GST and WT are used to obtain inverted list of each partition pre-filtered by "Position Restricted Alignment" that combines the well-know length and position filters. All inverted lists are then merged to retrieve the strings similar to r.
Team 7 submitted only one program: **Program 7_A**, which was evaluated for Track 1 with READS only.

3.8 Team 8

Team 8 (University of NSW) presents a solution based on tries, which have the advantages of small indexing space, freeness of verification, and computation sharing among strings with common prefixes. The method proposed is a simple adaptation of trie-based error-tolerant prefix matching [30]. Existing trie-based methods process a query by incrementally traversing the trie and maintaining a set of trie nodes (called active nodes) for each prefix of the query. One common drawback is that they have to maintain a large number of active nodes. Instead, Team 8 records only a small number of potentially feasible nodes as "active nodes" during query processing, which reduces the overhead of maintaining nodes and reporting results. In addition, Team 8 characterizes the essence of edit distance computation by a novel data structure named edit vector automaton, which substantially accelerates the state transition of active nodes, and therefore, improves the total query performance. Naive parallelization is added to exploit multi-core CPUs.
Team 8 submitted only one program: **Program 8_A**, which was evaluated for Track 1 with CITIES only.

3.9 Team 9

BWTSearcher of Team 9 (Northeastern University) takes advantage of a cache-aware multicore framework using Burrows-Wheeler-Transform [4]. BWTSearcher segments the whole collection of database sequences to fit to the CPU cache lines. The approximate string search algorithm is based on a partition approach. The query is decomposed into $\tau + 1$ chunks. If P matches the text with at most τ errors, at least one of the parts will match a substring of the text exactly. A new data structure called BWTPA is proposed to find the matching candidates. Length filter and position filter are used to prune the candidates. Team 9 proposed a reversed segment trie to merge the identical segments, which can save much duplicated computation. In addition, a look ahead algorithm is developed to support bounded edit distance and improve the verification of the candidate strings. BWTSearcher can search on any dataset, but is not optimized on DNA data, yet.
Team 9 has only one participating program: **Program 9_A**, which was evaluated on all datasets for Track 1.

	TINY		SMALL		MEDIUM		LARGE		HUGE	
Prog.	I	S	I	S	I	S	I	S	I	S
1_A	0.4	0.2	1.1	0.4	10.3	**4.3**	34.0	**24.5**	108.0	312.1
1_B	0.4	0.2	1.2	0.4	10.5	9.5	33.6	64.9	100.9	924.7
2_A	0.1	2.4	1.3	185.7	-	-	-	-	-	-
3_A	0.0	1.5	0.0	4.5	0.3	289.8	0.7	1,979.8	2.0	30,898.0
4_A	2.5	0.5	29.3	**0.2**	291.0	4.6	872.5	24.6	2,251.8	**232.5**
4_B	1.7	0.3	23.0	0.5	235.2	5.4	710.3	27.8	1,754.5	249.0
5_A	0.0	0.5	0.1	23.9	0.9	2,802.1	-	-	-	-
5_B	1.4	**0.1**	2.4	0.7	15.8	8.7	55.4	51.6	192.2	580.8
5_C	1.4	**0.1**	2.4	1.7	15.7	31.4	55.3	95.8	193.9	761.2
5_D	1.4	**0.1**	2.3	2.7	15.5	52.5	55.7	138.9	193.7	900.3
7_A	0.5	0.5	1.1	0.4	168.4	13.2	567.8	62.9	2,710.9	1,587.8
9_A	0.3	0.2	2.4	9.2	26.5	532.5	85.6	3,269.4	465.6	42,866.6

Figure 4: Indexing (I) and search (S) times for different READS datasets (Track 1, k varies per query) [time in seconds].

4. EVALUATING APPROXIMATE STRING SEARCH METHODS

In the following section we report results for all submissions for Track 1: approximate string search. We present results for READS datasets first and then for CITIES.

4.1 Similarity Search for READS

In Figure 4, we show the indexing and search times for the READS dataset and random values for k (for each query in the dataset we have assigned a random number out of $\{0, 4, 8, 12, 16\}$). For READS-TINY and READS-SMALL most of the programs compute the results within a few seconds, with two exceptions. 2_A, the index-less approach, needs already 185 seconds for answering READS-SMALL. For READS-MEDIUM, 2_A did not compute a result within several hours, so it was not evaluated on the larger datasets. Program 5_A, another index-less approach, needs 23.9 seconds for READS-SMALL and around 45 minutes for READS-MEDIUM. Therefore, 5_A was not tested on READS-LARGE and READS-HUGE.

The fastest programs for READS-HUGE are 4_A and 4_B, taking 232.5 and 249.0 seconds, respectively. The third program is 1_A, which needs 312.1 seconds. However, the indexing time of 1_A is around 20 times shorter than the indexing time for 4_A and 4_B. Programs 1_B, 5_B, 5_C, and 5_D need 10 to 15 minutes for READS-HUGE. Program 3_A, which does not use an index structure, already needs 8 hours to compute all solutions for READS-HUGE.

In Figure 16, we show search times for different values of k and the dataset READS-MEDIUM. The indexing time for all the programs is independent of the value of k, and is shown in Figure 4. Except 3_A and 9_A, all programs can compute the results set for $k \leq 8$ within few seconds. The best program for $k = 16$ is 4_A, needing only 17.8 seconds, followed by 4_B and 1_A. For all values of k, 4_A is among the fastest programs, only clearly outperformed by 1_A for $k = 12$.

We have further analyzed the effect of batch-processing for all programs for READS-MEDIUM and $k = 4$, except 2_A. In Figure 15, the average time per query for different numbers of queries is shown. It can be seen that for most programs, the average query answering time per query is reduced, if the number of queries is increased. For a large number of queries, the programs of Team 1 and Team 4 have the shortest time per query. We have further evaluated the three top-performing programs on our second evaluation environment System 2 with a different number of threads. Each program was preset to use 8, 24, and 80 threads, respectively. In Figure 5, the results of the evaluation are shown. It can be seen that 1_A and 4_A scale quite well with the number of threads: if the number of threads is increased by 3 (8 to 24), the search time is reduced by a factor larger than 2. The improvement from 24 threads to 80 threads is not as big any more. For 5_B there is almost no effect when increasing the number of threads. Their multiple backtracking algorithm is not straightforward to parallelize and the static load-balancing approach doesn't scale well. In this scenario it is probably easier to abandon multiple backtracking and go back to "standard" single backtracking, to allow a query-by-query parallelization.

		8 threads		24 threads		80 threads	
READS-	Prog.	I	S	I	S	I	S
MEDIUM	1_A	16.6	**4.1**	15.8	**1.8**	14.6	**1.2**
	4_A	510.6	4.9	527.2	2.0	639.4	1.5
	5_B	25.0	14.7	24.9	17.4	18.1	16.6
LARGE	1_A	47.4	**26.3**	48.3	**10.9**	47.8	**7.0**
	4_A	1,851.3	27.0	1,518.6	12.8	1,740.8	8.1
	5_B	93.7	80.0	66.1	81.8	66.0	91.2
HUGE	1_A	131.8	371.7	134.9	137.7	131.1	82.1
	4_A	4,290.4	**245.3**	3,718.7	**87.2**	4,096.2	**42.8**
	5_B	301.2	1,237.5	240.3	1,186.4	2,172.2	1,403.7

Figure 5: Search times for READS on System 2 [time in seconds].

	TINY		SMALL		MEDIUM		LARGE		HUGE	
Prog.	I	S	I	S	I	S	I	S	I	S
1_A	0.1	0.5	0.1	**0.4**	0.2	**0.9**	0.9	18.2	1.9	59.9
1_B	0.1	0.4	0.1	**0.4**	0.2	**0.9**	0.9	17.7	1.7	**46.8**
2_A	0.0	0.5	0.0	4.0	0.1	23.6	0.2	228.3	-	-
3_A	0.0	1.5	0.0	3.0	0.0	6.1	0.1	41.2	0.2	109.6
4_A	2.3	**0.2**	3.9	0.7	7.0	1.6	25.0	28.5	39.7	69.2
4_B	1.1	0.5	3.9	0.7	7.0	1.6	24.5	28.4	39.9	67.3
5_A	0.0	2.0	0.0	39.0	0.0	176.5	0.1	3,623.9	-	-
5_B	2.4	1.1	2.4	14.6	2.5	53.8	2.7	1,018.9	3.1	4,903.0
5_C	2.4	1.1	2.4	13.6	2.4	44.7	2.7	1,088.8	3.2	4,387.4
5_D	2.4	1.6	2.4	14.6	2.5	43.2	2.7	1,062.3	3.1	3,097.0
6_A	13.0	0.5	63.2	1.3	126.3	2.8	562.4	16.0	1,206.3	248.3
8_A	0.0	0.5	0.1	1.4	0.2	5.4	1.0	107.9	2.0	445.5
9_A	0.1	0.5	0.1	0.9	0.2	2.5	1.1	**15.2**	1.6	137.5

Figure 6: Indexing (I) and search (S) times for different CITIES datasets [time in seconds].

4.2 Similarity Search for CITIES

In Figure 6, we show the indexing and search times for the CITIES dataset and random values for k. For CITIES-TINY and CITIES-SMALL most of the programs compute the results within a few seconds. The only exception are the programs of Team 5, which need already 13.6 -39.0 seconds for CITIES-SMALL. All programs were tested on all datasets, with two exceptions. Programs 2_A and 5_A did not return a result for CITIES-

		8 threads		24 threads		80 threads	
CITIES-	Progr.	I	S	I	S	I	S
MEDIUM	1_A	0.21	**0.57**	0.22	**0.24**	0.27	**0.19**
	4_A	10.25	0.95	10.26	0.38	10.31	0.23
	5_B	0.77	158.15	1.20	133.68	1.36	103.95
LARGE	1_A	1.123	**12.84**	1.042	**5.341**	1.143	**3.222**
	4_A	33.283	17.68	33.353	7.297	33.686	4.377
HUGE	1_A	2.225	**43.615**	2.226	**19.679**	2.247	**11.529**
	4_A	52.903	57.473	53.53	28.283	53.175	21.057

Figure 7: Search times for CITIES on System 2 [time in seconds].

HUGE within several hours. Indexing times are quite short for all programs, except 6_A, which almost spends 20 minutes on indexing CITIES-HUGE.
The fastest program for CITIES-HUGE is 1_B, needing 46.8 seconds. It is closely followed by 1_A, 4_A, and 4_B. The programs of Team 5 are the slowest for CITIES, which probably means that their approach is better suited to deal with small-alphabets.
In Figure 17, the search times for CITIES-MEDIUM and different values of k are shown. Programs 1_A and 1_B are always among the fastest.
We have further evaluated the three top-performing programs on our second evaluation environment System 2 with a different number of threads. In Figure 7, the results are shown. The results are very similar to the results of READS: Program 1_A and 4_A scale well from 8 to 24 threads and quite good for 24 threads to 80 threads. Program 5_B does not scale as well as the other two (and was not tested for CITIES-LARGE and CITIES-HUGE).
Figure 8 shows a comparison of indexing times vs. search times for READS-HUGE and CITIES-HUGE for System 1.

5. EVALUATING APPROXIMATE STRING JOIN METHODS

In the following section we report on results of all submissions for Track 2: approximate string join. Again, we present results for READS datasets first and then for CITIES.

5.1 Similarity Join for READS

In Figure 18 and Figure 20, we show the join times for the READS dataset, for $k = 0$ (a) and $k = 16$ (b), respectively.
For $k = 0$, all programs have been tested for all datasets, except from 5_A. Program 5_A already needs around 30 minutes to perform a join on READS-SMALL. The fastest programs need less than 10 seconds to perform a self-join on READS-HUGE: 1_A and 1_B. For k=16, most programs could only be tested until READS-SMALL. Two programs were evaluated in READS-HUGE: Program 1_A needed 22.9 hours and Program 4_A needed 41.5 hours.
We report the join times for READS-HUGE and different values for k in Figure 9. Programs 3_A and 9_A already need more than 20 hours to perform a 4-approximate self-join on READ-HUGE. The best performing method is implemented in Program 1_A.
We have further evaluated the three top-performing programs on our second evaluation environment System 2 with a different number of threads. In Figure 21, the results are shown. For all programs a higher number of threads reduces the runtime. It is interesting to see that with an increasing value of k, the effect is bigger than with small numbers. We conjecture that the overhead of setting up the threads and synchronization is dominating for smaller k.

5.2 Similarity Join for CITIES

Join times for the CITIES dataset are reported in Figure 22 for $k = 0$ and in Figure 19 for $k = 4$. Apart from Program 5_A, all programs finished to compute an exact self-join on all CITIES datasets. Program 1_A is the fastest program in each case. Team 4's programs are ranked second. Program 3_A finishes third, which is quite remarkable for an index-less approach.
The join times for CITIES-HUGE and different values of k are reported in Figure 10. Program 1_A is the best for all values of k, except for $k = 1$, where it is outperformed slightly by 1_B. We did not test the index-less approach 5_A.
We have further evaluated the three top-performing programs on our second evaluation environment with a different number of threads. In Figure 23, the results are shown. For all programs a higher number of threads reduces the runtime. The results show a similar behavior as when joining READS: it seems that performing a join with a small k usually is better with a small number of threads, while for larger values of k it makes indeed sense to use parallelism.

6. POST-COMPETITION ANALYSIS

The main results of our competition are shown in Figure 11. For each task and dataset we list the techniques used by the three top performing teams. The partitioning and pruning techniques of Team 1 show the best performance for three out of four problems. Only for searching our READS dataset, the acyclic word graph of Team 4 slightly outperforms Team 1's techniques.
In the following we discuss our results and relate them to existing work not covered by the competition.

6.1 Additional algorithms

We compare the results of the competition to existing tools for approximate string search. We only take into account non-dominated methods from Figure 1, for which no participant of our competition had a direct contribution. The only such non-dominated method is SSI [9]. In addition, we test two other methods: Flamingo [3], which is often used as baseline for evaluation, and Pearl [23], a prefix tree index. The results are shown in Figure 12, together with the comparison of the best three ranked programs from our competition.
Unfortunately, Flamingo has only implemented approximate search, no approximate join. We run Flamingo with the standard configuration (filters as set by the Getting-Started-example) and different length of q-grams. Index and search times are considerably longer than many of the competitors in our competition. However,

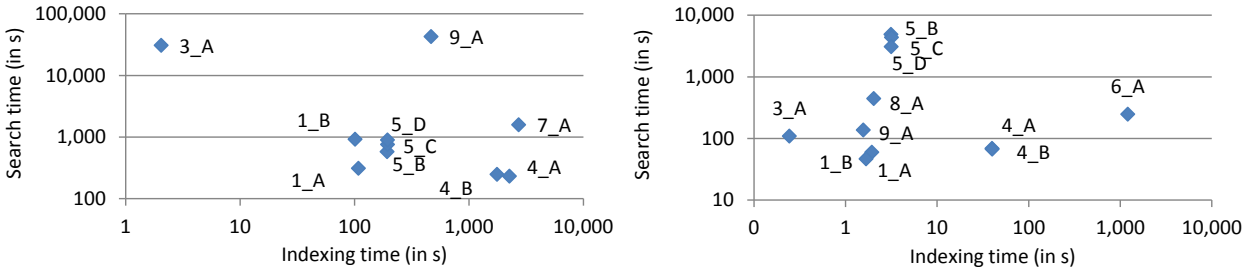


Figure 8: Search/Indexing times for READS-HUGE (left) and CITIES-HUGE (right) [time in seconds].

	READS-HUGE (time in minutes!)				
Prog.	**k=0**	**k=4**	**k=8**	**k=12**	**k=16**
1_A	**0.2**	1.0	**3.7**	**84.4**	**1,377.3**
1_B	**0.2**	**0.9**	8.9	231.0	-
3_A	258.8	5,760.0	-	-	-
4_A	37.6	41.3	81.2	220.8	2,489.1
4_B	29.4	31.4	75.7	214.1	-
5_A	-	-	-	-	-
5_B	0.5	12.4	126.7	2,590.4	-
5_C	0.5	12.1	111.9	-	-
5_D	0.5	12.3	74.9	-	-
9_A	5.5	1,197.5	-	-	-

Figure 9: Join times for READS-HUGE and different k [time in minutes].

	CITIES-HUGE				
Prog.	**k=0**	**k=1**	**k=2**	**k=3**	**k=4**
1_A	**1.0**	1.9	**6.1**	**50.1**	**345.5**
1_B	1.1	**1.8**	6.8	53.8	353.0
3_A	588.1	564.1	655.8	847.6	1,700.0
4_A	40.9	45.5	81.2	440.6	945.0
4_B	39.7	42.2	78.8	418.3	942.0
5_B	11.3	78.3	1,719.2	-	-
5_C	11.3	37.1	726.2	11,462.5	-
5_D	11.4	32.8	785.9	-	-
8_A	3.3	21.2	218.2	3,339.2	21,230.0
9_A	10.9	28.9	198.7	1,912.9	-

Figure 10: Join times for CITIES-HUGE and different k [time in seconds].

	place	READS	CITIES
search	1	4_A (acyclic word graph)	1_A (partitioning and pruning)
	2	1_A (partitioning and pruning)	4_A (acyclic word graph)
	3	5_B (radix/Suffix trees)	3_A (bit-parallel LCS computation)
join	1	1_A (partitioning and pruning)	1_A (partitioning and pruning)
	2	4_A (acyclic word graph)	4_B (acyclic word graph)
	3	5_B (radix/Suffix trees)	3_A (bit-parallel LCS computation)

Figure 11: Overall ranking for search and join.

CITIES:	**SMALL**		**MEDIUM**		**HUGE**	
Prog.	**Index**	**Search**	**Index**	**Search**	**Index**	**Search**
Flamingo |q|=2	0.1	4.7	0.2	26.6	-	-
Flamingo |q|=3	0.2	7.6	0.4	42.8	-	-
Flamingo |q|=4	0.2	9.7	0.5	55.4	-	-
Pearl	2.9	33.5	6.4	74.9	99.4	2,541.1
SSI						
1_B	0.1	0.4	0.2	0.9	1.7	46.8
4_B	3.9	0.7	7.0	1.6	39.9	67.3
3_A	0.0	3.0	0.0	6.1	0.2	109.6

READS:	**SMALL**		**MEDIUM**		**HUGE**	
Prog.	**Index**	**Search**	**Index**	**Search**	**Index**	**Search**
Flamingo |q|=5	2.1	45.9	-	-	-	-
Flamingo |q|=6	2.3	44.1	36.8	6,052.2	-	-
Flamingo |q|=7	3.0	443.7	-	-	-	-
Pearl	10.1	3,567.0	-	-	-	-
SSI	0.4	27.7	1.2	5,032.1	-	-
4_A	29.3	0.2	291.0	4.6	2,251.8	232.5
1_A	1.1	0.4	10.3	4.3	108.0	312.1
5_B	2.4	0.7	15.8	8.7	192.2	580.8

Figure 12: Indexing and Search times for Flamingo, Pearl, and SSI [time in seconds].

note that Flamingo makes only use of one thread and the memory footprint seems to be very small. Possibly, performance of Flamingo can be further improved by additional filters. We have tested SSI only on the READS datasets. For each CITIES dataset, SSI stopped with a insufficient memory exception. This might be a bug affecting the handling of large alphabets.

The best programs from our competition outperform these tools by a factor of 1000 and more for READS-MEDIUM and a factor of 50 and more for CITIES-HUGE. In addition, we have evaluated Pearl for joining GEONAMES datasets: Even for GEONAMES-MEDIUM and $k = 4$, Pearl needs more than 1 hour to compute the self-join, while 1_A needs less than 2 minutes. Given the existing evaluation results from Figure 1, for each non-dominated method either one of its authors has contributed to our competition (Pass-Join, Trie-Join, PPJoin, NGPP) or the method (SSI) was shown to be way less scalable than our best programs. Therefore, we believe that our analysis represents the state-of-the-art in string similarity search and join.

6.2 Memory usage

We show the peak main memory usage with respect to READS-Huge in Figure 24. Programs 5_B, 5_C, and 5_D only use around 13.6 GB of main memory, followed by 3_A with 15.6 GB. The maximum amount of main memory is used by 9_A with 40.6 GB. The average main memory is 24.2 GB, which means that all the programs make use of roughly half of the main memory available.. In Figure 25, the peak main memory usage for the dataset CITIES-HUGE is shown. Most of the programs show modest memory usage; the average is only 6 GB. The most main memory is used by Program 6_A: 24.7 GB, followed by 4_A and 4_B with 12-13 GB. Program 9_A only uses 0.6 GB of main memory. Thus, most of the main memory is left unused. We conjecture that it might be possible to further improve query answering times for some techniques by pre-computation of more sophisticated index structures.

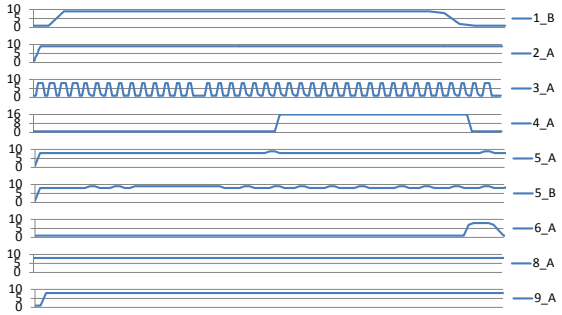


Figure 13: Searching CITIES-LARGE: number of active threads from the beginning of the program until its termination. Note that all the programs had a different run time, the x-axis has a different scale for each program.

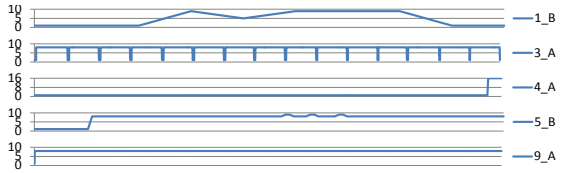


Figure 14: Joining READS-Medium with k=4: number of active threads from the beginning of the program until its termination.

6.3 CPU utilization

In Figure 13, the number of active threads is shown over time when searching CITIES-LARGE. The graphs of 1_A, 4_B, 5_C, and 5_D are not shown since they are very similar to 1_B, 4_A, 5_B, and 5_B, respectively. Most of the programs start preprocessing with one thread and then increase the number of threads. Program 3_A is the only program which does not follow this pattern. Load scheduling of programs 1_B and 4_A can possibly be improved, since these programs do not make constant use of the full number of available cores. Program 4_A has a long single-thread preprocessing phase; queries are answered using 16 threads.

In Figure 14, the number of active threads is shown over time when joining READS-MEDIUM with $k = 4$. The graphs of 1_A, 4_B, 5_C, and 5_D are not shown since they are very similar to 1_B, 4_A, 5_B, and 5_B, respectively. The overall join time for 1_B is only few seconds, so the graph is not as stable as the other ones. For Program 4_A and 5_B the preprocessing phase can be clearly identified (with only one thread). Program 4_A makes use of 16 threads again instead of only 8. Program 9_A uses 8 threads for most of the time. (only the first few seconds are run with only one thread).

6.4 Redundancy

The official rules allowed to serialize the same answer several times: sometimes the same result is found by different components of a search algorithm independently. In Figure 26, we analyze the redundancy in the results. The programs of Team 4 and Team 5 report answers several times (in average 4-6 times). All other programs report each answer only once (baseline 100 percent).

7. CONCLUSION

We believe that our evaluation gives a fairly representative picture of the state-of-the-art in string similarity search and join for different data set sizes, different alphabet sizes, and different error thresholds.. Based on our datasets and competing programs, we conclude that an error rate of 20-25% pushes today's techniques to the limit. For instance, self-joining a set of 15.000.000 sequence reads of length 100 with an edit distance threshold $k = 16$ takes almost one day even for the best participant. However, the final result has more than 50.000.000 entries, which makes the usefulness of such queries in real applications questionable.

Our experiments showed that many participants used less main memory than available. The effect is perspicuous for our CITIES dataset: more than half of the competitors used less than 10 percent of the main memory. An interesting lead for future research are indexing strategies that make full use of existing main memory. Even for smaller datasets, query answering times might be further reduced by more precomputation at indexing time.

Although we have ranked programs based on search time, we have also measured indexing time separately. We found that indexing times vary a lot between implementations; in addition many programs use only one thread for indexing. Another point that could be improved as revealed by our analysis is to improve thread utilization, especially for current hardware with their quickly increasing number of cores.

It is interesting to note that the three top performing teams use difference techniques. Combining these techniques, e.g. the bit-parallel LCS computation from Team 3 with the pruning techniques of Team 1, could probably reduce search and join times beyond the state-of-the-art.

8. ACKNOWLEDGMENTS

We thank Nikolaus Augsten for his insightful comments on a draft version of this paper. In addition, we thank Thomas Stoltmann for providing us Figure 1.

9. REFERENCES

[1] A. Arasu, V. Ganti, and R. Kaushik. Efficient exact set-similarity joins. In *PVLDB*, VLDB '06, pages 918–929. VLDB Endowment, 2006.

[2] R. J. Bayardo, Y. Ma, and R. Srikant. Scaling up all pairs similarity search. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 131–140, New York, NY, USA, 2007. ACM.

[3] A. Behm, R. Vernica, S. Alsubaiee, S. Ji, J. Lu, L. Jin, Y. Lu, and C. Li. UCI Flamingo Package 4.1, 2010.

[4] M. Burrows and D. J. Wheeler. A block-sorting lossless data compression algorithm. Technical report, Digital SRC Research Report, 1994.

[5] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *PVLDB*, VLDB '97, pages 426–435, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[6] M. Crochemore, C. S. Iliopoulos, Y. J. Pinzon, and J. F. Reid. A fast and practical bit-vector algorithm for the Longest Common Subsequence problem. *Information Processing Letters*, 80(6), Dec. 2001.

[7] D. Dey, S. Sarkar, and P. De. A distance-based approach to entity reconciliation in heterogeneous databases. *IEEE Trans. Knowl. Data Eng.*, 14(3):567–582, 2002.

[8] J. Feng, J. Wang, and G. Li. Trie-join: a trie-based method for efficient string similarity joins. *The VLDB Journal*, 21(4):437–461, 2012.

[9] D. Fenz, D. Lange, A. Rheinländer, F. Naumann, and U. Leser. Efficient similarity search in very large string sets. In A. Ailamaki and S. Bowers, editors, *Scientific and Statistical Database Management*, volume 7338 of *Lecture Notes in Computer Science*, pages 262–279. Springer Berlin Heidelberg, 2012.

[10] X. Ge and P. Smyth. Deformable Markov model templates for time-series pattern matching. In *Proceedings of SIGKDD*, pages 81–90, New York, NY, USA, 2000. ACM.

[11] S. Gerdjikov, S. Mihov, P. Mitankin, and K. U. Schulz. Good parts first - a new algorithm for approximate search in lexica and string databases. *ArXiv e-prints*, Jan. 2013.

[12] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *PVLDB*, VLDB '99, pages 518–529, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[13] M. Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. SIGIR '06, pages 284–291, New York, NY, USA, 2006. ACM.

[14] H. Hyyrö, K. Fredriksson, and G. Navarro. Increased bit-parallelism for approximate and multiple string matching. *ACM Journal of Experimental Algorithmics*, 10, 2005.

[15] G. Li, D. Deng, J. Wang, and J. Feng. Pass-join: A partition-based method for similarity joins. *PVLDB*, 5(3):253–264, 2011.

[16] H. Li and R. Durbin. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics (Oxford, England)*, 25(14):1754–1760, 2009.

[17] Y. Li, A. Terrell, and J. M. Patel. WHAM: a high-throughput sequence alignment method. SIGMOD '11, pages 445–456. ACM, 2011.

[18] S. Mihov and K. U. Schulz. Fast approximate search in large dictionaries. *Computational Linguistics*, 30(4):451–477, 2004.

[19] P. Mitankin, S. Mihov, and K. U. Schulz. Deciding word neighborhood with universal neighborhood automata. *Theoretical Computer Science*, 412(22):2340 – 2355, 2011.

[20] I. Moraru and D. G. Andersen. Exact pattern matching with feed-forward Bloom filters. *J. Exp. Algorithmics*, 17(1):3.4:3.1–3.4:3.18, Sept. 2012.

[21] G. Navarro and R. Baeza-Yates. A hybrid indexing method for approximate string matching. *Journal of Discrete Algorithms*, 1(1):205–239, 2000.

[22] M. Patil, S. V. Thankachan, R. Shah, W.-K. Hon, J. S. Vitter, and S. Chandrasekaran. Inverted indexes for phrases and strings. In *SIGIR 2011*, pages 555–564, 2011.

[23] A. Rheinländer and U. Leser. Scalable sequence similarity search and join in main memory on multi-cores. In *Proceedings of the 2011 international conference on Parallel Processing - Volume 2*, Euro-Par'11, pages 13–22, Berlin, Heidelberg, 2012. Springer-Verlag.

[24] E. Siragusa, D. Weese, and K. Reinert. Fast and accurate read mapping with approximate seeds and multiple backtracking. *Nucleic acids research*, Jan. 2013.

[25] B. S. T. Bocek, E. Hunt. Fast Similarity Search in Large Dictionaries. Technical Report ifi-2007.02, April 2007. http://fastss.csg.uzh.ch/.

[26] A. Tiskin. Semi-local longest common subsequences in subquadratic time. *J. Discrete Algorithms*, 6(4):570–581, 2008.

[27] E. Ukkonen. Algorithms for approximate string matching. *Information Control*, 64:100–18, 1985.

[28] G. Wang, B. Wang, X. Yang, and G. Yu. Efficiently indexing large sparse graphs for similarity search. *IEEE Trans. Knowl. Data Eng.*, 24(3):440–451, 2012.

[29] W. Wang, C. Xiao, X. Lin, and C. Zhang. Efficient approximate entity extraction with edit distance constraints. SIGMOD '09, pages 759–770, New York, NY, USA, 2009. ACM.

[30] C. Xiao, J. Qin, W. Wang, Y. Ishikawa, K. Tsuda, and K. Sadakane. Efficient error-tolerant query autocompletion. *PVLDB*, 2013.

[31] C. Xiao, W. Wang, and X. Lin. Ed-join: an efficient algorithm for similarity joins with edit distance constraints. *PVLDB*, 1(1):933–944, Aug. 2008.

[32] C. Xiao, W. Wang, X. Lin, and J. X. Yu. Efficient similarity joins for near duplicate detection. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 131–140, New York, NY, USA, 2008. ACM.

[33] C. Xiao, W. Wang, X. Lin, J. X. Yu, and G. Wang. Efficient similarity joins for near-duplicate detection. *ACM Trans. Database Syst.*, 36(3):15, 2011.

	READS-MEDIUM - Number of queries				
Prog.	1	100	10,000	100,000	200,000
1_A	199.0000	1.9900	0.0225	0.0042	0.0031
1_B	205.0000	2.0100	0.0220	0.0048	0.0032
2_A	-	-	-	-	-
3_A	1,625.0000	18.3100	3.0682	4.2107	3.8523
4_A	83.0000	0.7800	**0.0101**	**0.0041**	0.0035
4_B	107.0000	0.8700	**0.0101**	0.0043	**0.0030**
5_A	50.0000	234.5200	-	-	-
5_B	52.0000	0.2200	0.0211	0.0160	0.0142
5_C	**38.0000**	**0.1800**	0.0228	0.0174	0.0138
5_D	44.0000	0.2100	0.0214	0.0155	0.0144
7_A	116.0000	1.5600	0.5538	0.5519	0.5423
9_A	279.0000	25.7800	24.0174	25.8930	24.8394

Figure 15: Batch effect for READS-MEDIUM: Time per query for a different number of total queries (1-200,000 queries) [time in milliseconds].

	MEDIUM				
Prog.	k=0	k=4	k=8	k=12	k=16
1_A	0.2	0.2	**0.3**	**1.5**	25.4
1_B	0.2	0.2	0.4	3.1	42.1
2_A	-	-	-	-	-
3_A	2.9	30.9	136.2	335.8	972.6
4_A	**0.1**	**0.1**	0.4	3.3	**17.8**
4_B	**0.1**	**0.1**	0.4	3.5	20.1
5_A	-	-	-	-	-
5_B	**0.1**	0.2	0.9	19.5	56.4
5_C	**0.1**	0.2	3.9	9.1	108.4
5_D	**0.1**	0.2	5.2	44.7	160.8
7_A	0.4	5.6	6.4	20.5	30.5
9_A	117.3	242.0	242.5	311.2	1,749.3

Figure 16: Search times for READS-MEDIUM and different values of k [time in seconds].

	MEDIUM				
Prog.	k=0	k=1	k=2	k=3	k=4
1_A	**0.0**	**0.0**	**0.1**	**0.5**	3.5
1_B	**0.0**	**0.0**	**0.1**	0.6	**3.0**
2_A	8.0	7.0	7.2	16.7	21.3
3_A	5.3	5.2	5.5	6.0	8.0
4_A	**0.0**	**0.0**	**0.1**	0.9	6.2
4_B	**0.0**	**0.0**	0.2	0.9	5.9
5_A	178.4	172.8	154.3	159.9	194.7
5_B	**0.0**	0.6	6.2	63.3	206.1
5_C	**0.0**	0.7	9.2	39.1	199.1
5_D	13.6	11.9	24.6	58.4	119.0
6_A	0.3	2.3	5.4	7.8	15.4
8_A	0.1	0.1	0.6	4.0	18.4
9_A	**0.0**	0.1	0.3	2.5	9.1

Figure 17: Search times for CITIES-MEDIUM [time in seconds].

	READS k=0				
Prog.	TINY	SMALL	MEDIUM	LARGE	HUGE
1_A	**0.5**	1.1	**1.6**	**4.4**	**9.6**
1_B	**0.5**	**0.6**	1.8	4.6	9.9
3_A	2.0	8.3	200.3	1,836.1	15,531.2
4_A	2.5	29.8	288.5	870.0	2,258.0
4_B	2.0	23.8	234.5	709.9	1,764.5
5_A	19.5	1,813.8	-	-	-
5_B	2.5	3.3	5.2	9.5	30.8
5_C	2.5	3.3	4.7	9.2	30.9
5_D	2.5	4.0	5.1	9.2	30.6
9_A	**0.5**	1.2	7.0	9.1	328.7

Figure 18: Join times for READS and $k = 0$ [time in seconds].

	CITIES k=4				
Prog.	TINY	SMALL	MEDIUM	LARGE	HUGE
1_A	**0.7**	**3.0**	**10.5**	**117.0**	**345.5**
1_B	0.9	**3.0**	11.0	119.5	353.0
3_A	6.5	31.0	68.5	577.0	1,700.0
4_A	2.0	17.0	54.0	807.0	945.0
4_B	2.5	17.0	57.5	810.0	942.0
5_A	10.4	205.5	982.5	-	-
5_B	13.8	241.0	920.5	-	-
5_C	15.0	226.5	926.0	-	-
5_D	22.6	266.0	838.5	2,401.0	-
8_A	6.0	141.5	532.5	3,585.0	21,230.0
9_A	16.1	193.5	578.5	-	-

Figure 19: Join times for CITIES and k=4 [time in seconds].

	READS k=16				
Prog.	TINY	SMALL	MEDIUM	LARGE	HUGE
1_A	**0.5**	**9.8**	**1,028.3**	**11,283.9**	**82,636.5**
1_B	**0.5**	26.0	2,941.0	33,055.5	-
3_A	26.0	1,732.3	-	-	-
4_A	33.1	362.8	4,048.4	25,823.9	149,344.1
4_B	32.5	361.7	-	-	-
5_A	19.8	2,217.3	-	-	-
5_B	4.1	50.8	4,200.9	-	-
5_C	31.0	431.0	-	-	-
5_D	40.0	625.0	-	-	-
9_A	159.7	9,327.3	-	-	-

Figure 20: Join times for READS and $k = 16$ [time in seconds].

		READS-MEDIUM				
Threads	Progr.	k=0	k=4	k=8	k=12	k=16
8	1_A	**1.22**	**8.51**	**16.22**	**87.95**	**1,000.14**
	4_A	460.37	470.45	633.01	1,724.68	6,077.06
	5_B	3.04	80.29	213.45	3,538.78	10,230.97
24	1_A	**1.23**	**6.76**	**10.76**	**33.96**	**381.07**
	4_A	460.26	462.56	576.99	869.19	2,354.78
	5_B	5.63	55.41	162.01	3,679.70	9,808.58
80	1_A	**1.22**	**6.64**	**9.57**	**23.61**	**335.73**
	4_A	469.87	460.93	486.23	645.42	1,318.72
	5_B	3.76	52.55	188.61	3,437.48	5,157.10

Figure 21: Join times for READS-MEDIUM on System 2 [time in seconds].

	CITIES k=0				
Prog.	TINY	SMALL	MEDIUM	LARGE	HUGE
1_A	**0.6**	**0.6**	**0.6**	**0.6**	**1.0**
1_B	0.8	0.7	0.7	**0.6**	1.1
3_A	5.8	28.4	56.7	287.2	588.1
4_A	1.9	4.2	7.0	24.9	40.9
4_B	1.7	4.3	7.2	25.0	39.7
5_A	7.7	175.1	850.1	-	-
5_B	4.7	4.6	4.5	6.7	11.3
5_C	4.6	4.7	4.8	6.3	11.3
5_D	4.9	4.8	4.8	6.2	11.4
8_A	1.0	**0.6**	0.9	1.4	3.3
9_A	0.7	1.0	**0.6**	3.4	10.9

Figure 22: Join times for CITIES and k=0 [time in seconds].

		CITIES-MEDIUM				
Threads	**Progr.**	**k=0**	**k=1**	**k=2**	**k=3**	**k=4**
8	1_A	**0.06**	**0.30**	**0.53**	**1.83**	**8.12**
	4_A	10.40	10.35	11.15	17.06	46.00
	5_B	1.45	4.17	56.12	376.39	2,513.64
24	1_A	**0.08**	**0.27**	**0.38**	**0.94**	**3.14**
	4_A	10.42	10.37	10.69	12.60	22.76
	5_B	5.76	4.07	65.28	760.71	2,353.97
80	1_A	**0.11**	**0.31**	**0.39**	**0.85**	**2.42**
	4_A	10.47	10.46	10.48	11.37	16.76
	5_B	2.38	3.92	42.47	532.91	2,051.15

Figure 23: Join times for CITIES-MEDIUM on System 2 [time in seconds].

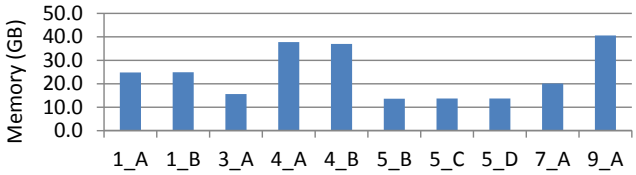


Figure 24: Peak main memory usage for READS-HUGE [memory in GB].

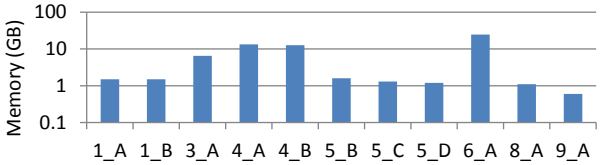


Figure 25: Peak main memory usage for CITIES-HUGE [memory in GB].

	READS-MEDIUM - Number of queries				
Prog.	**1**	**100**	**10,000**	**100,000**	**200,000**
1_A	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**
1_B	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**
2_A	-	-	-	-	-
3_A	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**
4_A	**100.0%**	200.0%	645.8%	479.2%	609.1%
4_B	**100.0%**	200.0%	645.8%	479.2%	609.1%
5_A	**100.0%**	200.0%	445.8%	479.8%	465.7%
5_B	**100.0%**	200.0%	445.8%	479.8%	465.7%
5_C	**100.0%**	200.0%	445.8%	479.8%	465.7%
5_D	**100.0%**	200.0%	445.8%	479.8%	465.7%
7_A	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**
9_A	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**

Figure 26: Result redundancy: Searching READS-MEDIUM with k=4 for different number of queries (1-200,000) [redundancy in percent; 100% stands for no redundant results; 200% means that in average each result is reported twice].

Report on the Second International Workshop on Cloud Intelligence (Cloud-I 2013)

Jérôme Darmont
Université de Lyon (Laboratoire ERIC)
5 avenue Pierre Mendès-France
F-69676 Bron Cedex – France
jerome.darmont@univ-lyon2.fr

Torben Bach Pedersen
Aalborg University (Daisy)
Selma Lagerløfs Vej 300
DK-9220 Aalborg Ø – Denmark
tbp@cs.aau.dk

http://eric.univ-lyon2.fr/cloud-i/

1. INTRODUCTION

Business intelligence (BI) is a broad field related to integrating, storing and analyzing data to help decision-makers in many domains (from "real" business to administration, health, and environment) make better decisions. Front-end analytics methods include reporting, on-line analytical processing (OLAP), and data mining.

With the increasing success of cloud computing, cloud BI "as a service" offerings have started appearing, both from cloud start-ups and major BI industry vendors. Beyond porting BI features into the cloud, which already implies numerous issues (e.g., BigData/NoSQL database modeling and storage, data localization, security and privacy, performance, cost and usage models...), this trend also poses new, broader challenges for making data analytics available to small and middle-size enterprises (SMEs), non-governmental organizations, Web communities (e.g., supported by social networks), and even the average citizen; this vision presumably requiring a mixture of both private and open data.

Thus, Cloud Intelligence is not only a current technological and research challenge, but also an important economic and societal stake, since people increasingly demand open data, which must be easily accessible on the Web, possibly mixed with private data, and analyzed with intelligible on-line tools with advanced collaborative features enabling users to share and reuse BI concepts and analyses in large scale fashion. Analysis results can then be be shared world-wide.

The Second International Workshop on Cloud Intelligence (Cloud-I 2013) [3] was held in conjunction with VLDB 2013 in Riva del Garda, Italy on August 26, 2013. In continuation of the first edition, it brought together researchers and engineers from academia and industry to discuss and exchange ideas related to Cloud Intelligence. The workshop featured a joint keynote with the BIRTE workshop and two research sessions, the latter including a panel discussion. The topics of this year's accepted papers mainly focused on MapReduce-based computations and indexing.

2. KEYNOTE

The keynote entitled "AsterixDB: A New Platform for Real-Time Big Data BI" was given by Prof. Michael J. Carey, from the University of California, Irvine. AsterixDB is a so-called "BDMS (Big Data Management System)" grounded in both parallel database systems and data-intensive computing frameworks (i.e., MapReduce and NoSQL-like frameworks) [2]. AsterixDB notably features a flexible, semi-structured data model (Asterix Data Model - ADM) based on JSON, a declarative query language (Asterix Query Language - AQL) for expressing a wide range of BI queries, and a parallel runtime engine, Hyracks, that has been scale-tested to thousands of cores. This feature set makes AsterixDB ideally suited to modern needs such as Web data warehousing, social data storage and analysis, and other real-time BI applications.

3. RESEARCH PAPERS

3.1 Session 1: MapReduce

Guoliang Zhou, Yongli Zhu and Guilan Wang, in "Cache Conscious Star-Join in MapReduce Environments", propose two join strategies for star schemas in MapReduce environments. Two algorithms, namely Multi-Fragment-Replication Join (MFRJ) and MapReduce-Invisible Join (MRIJ), avoid fact table data movement and are cache conscious in each MapReduce node. In addition, in MFRJ, the fact table is partitioned into several column groups for cache optimization; one group contains all the foreign key columns and each measure column becomes a separate group. In MRIJ, each column is processed separately one by one, giving a higher cache utilization and avoiding frequent cache misses from one column to the other column. Experimental results in cluster environments show that MFRJ and MRIJ outperform existing approaches in the Hive system.

In "Toward Intersection Filter-Based Optimization for Joins in MapReduce", Thuong-Cang Phan, Laurent d'Orazio

and Philippe Rigaux criticize MapReduce for not directly supporting join operations with multiple inputs. To address this problem, they propose a Bloom filter-based intersection filter that exploits probabilistic models to remove most disjoint elements between two datasets. The cost of two-way and cascade joins is analyzed to minimize disk I/O and communication costs. Comparison experiments show that this approach is more effective than existing state-of-the-art solutions.

Yanfeng Zhang and Shimin Chen, in the position paper entitled "i^{2}MapReduce: Incremental Iterative MapReduce", aim to to support incremental iterative computation (e.g., PageRank) on constantly changing datasets (e.g., the Web graph). Since in many cases, data changes impact only a very small fraction of the datasets, and the new iteratively converged state is quite close to the previously converged state, i^{2}MapReduce saves recomputation by starting from the previously converged state, and by performing incremental updates on changing data. Based on this, i^{2}MapReduce achieves a significant performance improvement over recomputing iterative jobs in MapReduce.

3.2 Session 2: Emerging Topics

In the paper entitled "Bloofi: A Hierarchical Bloom Filter Index with Applications to Distributed Data Provenance", Adina Crainiceanu considers the use of Bloom filters in federated cloud environments. With hundreds of geographically distributed clouds participating in a federation, information needs to be shared by the semi-autonomous cloud providers. This can be done by encoding the information using Bloom filters and sharing the Bloom filters with a central coordinator. Bloofi, an efficiently constructed and maintained hierarchical index structure for Bloom filters, is thus proposed to speed-up the search process. Theoretical and experimental results show that Bloofi provides a scalable and efficient solution for searching through a large number of Bloom filters.

4. PANEL

Finally, Jérôme Darmont and Torben Bach Pedersen launched a panel discussion themed "Cloud Intelligence – Challenges for Research and Industry", by presenting Fusion Cubes [1], a collaborative framework aiming to support *self-service business intelligence*. Self-service BI enables non-expert users to make well-informed decisions by enriching the decision process with situational data, i.e., data that have a narrow use for a specific business problem, typically a short lifespan for a small group of users, and are usually not owned and controlled by the decision maker. This way of working can even be pushed towards *personal intelligence*, i.e., cloud decision-support services (in SaaS mode) that would be accessible to individuals and very small companies, from "NoETL" data integration to easy OLAP, including the automatic multidimensional modeling of the underlying datamart [4].

Morten Middelfart, from the BI vendor TARGIT, also stressed the need to allow easy integration of existing BI data with external data using a "NoETL" paradigm.

Adina Crainiceanu questioned how to increase the impact of academic research, which fueled a stimulating discussion between academics and industrials about building systems *vs.* writing scientific publications; and the value and quality of (seldom well-documented) code produced by researchers *vs.* academia-industry collaborations.

Finally, the challenges ahead for cloud intelligence systems were discussed, such as extracting data and/or knowledge from low-availability sites, making self-service BI user-friendly, mobile cloud issues, hybrid SQL/NoSQL systems and interoperability, and finally privacy issues that integrating aggregate data can solve while complying to various international regulations.

5. DISCUSSION AND OUTLOOK

If we first look at the topics of the presented papers, we see that 3 out of 4 accepted papers concerns processing various types of queries in using a MapReduce platform. This witnesses the fact that one of the key challenges in cloud intelligence is the ability to process really massive datasets within an acceptable time frame. This is even more the case than in cloud computing in general, since cloud intelligence use cases tend to involve very expensive "deep analytics" computations.

The keynote addresses the same challenge, but in a broader and more advanced sense, since it presents a considerably more advanced platform than standard MapReduce, including a more advanced data model and query language, and much more advanced optimization techniques. This shows that the cloud intelligence community is, in some sense, returning to the "old virtues" of semantic data models and high-level query languages, only now with the key difference that massive scalability must be supported at all times.

The fourth paper considered scalability in a somewhat different sense, namely in large scale and wide area distributed cloud environments. This is a reminder of the fact that cloud intelligence is not only about scaling up within a single data center, but also to enable efficient integration of geographically dispersed information resources.

The panel considered wider issues such as personal and collaborative cloud intelligence, including ETL and data integration for cloud intelligence, human computer interaction and user friendliness, and privacy issues. In comparison, these topics are more open and the exact problems and solutions perhaps less obvious, than for the issue of scaling up and out specific computations. Thus, we expect that solutions will eventually emerge, but that it will take some time. The same is true for some topics from the Call for Papers that were not presented or discussed at all. These include developing novel payment models for amortizing the cost of cloud intelligence solutions over time and different users and how to provide cloud intelligence as a service. We attribute the lack of papers on such issues to the fact that the cloud intelligence field is still young and that other issues are more pressing for the majority of users.

Summing up, we conclude that there is a lot of interesting work going on in the area of cloud intelligence,

but that many challenges are still remaining. Thus, there is a continued need for venues that focus on this issue. We hope to continue the Cloud-I workshop series in connection with future VLDB conferences. For the third edition of the workshop, it is again the intention to organize a special issue of a journal for extended versions of the best papers.

6. ACKNOWLEDGEMENTS

The Cloud-I Chairs would like to thank all the authors of submitted papers for their interest in the workshop and the high quality of the submitted papers. We would also like to thank all the referees (both PC members and external reviewers) for their careful and dedicated work, both during the reviewing and the discussion phases. Working in cooperation with this Program Committee has been both an honor and a pleasure. Finally, we would like to express our gratitude to the members of the Organizing Committee of VLDB 2013, especially the Workshop Chairs Tiziana Catarci, AnHai Doan and Tova Milo, for their support in organizing this workshop; and to Craig Rodkin, Publications Operations Manager at the ACM, for his assistance in editing the proceedings.

7. REFERENCES

[1] A. Abello, J. Darmont, L. Etcheverry, M. Golfarelli, J.-N. Mazon, F. Naumann, T.-B. Pedersen, S. Rizzi, J. Trujillo, P. Vassiliadis, and G. Vossen. Fusion Cubes: Towards Self-Service Business Intelligence. *International Journal of Data Warehousing and Mining*, 9(2):66–88, April-June 2013.

[2] S. Alsubaiee, Y. Altowim, H. Altwaijry, A. Behm, V. R. Borkar, Y. Bu, M. J. Carey, R. Grover, Z. Heilbron, Y.-S. Kim, C. Li, N. Onose, P. Pirzadeh, R. Vernica, and J. Wen. ASTERIX: An Open Source System for "Big Data" Management and Analysis. *PVLDB*, 5(12):1898–1901, 2012.

[3] J. Darmont and T. B. Pedersen, editors. *2nd International Workshop on Cloud Intelligence (colocated with VLDB 2013), Cloud-I '13, Riva del Garda, Italy, August 26, 2013*. ACM, 2013.

[4] C. Phipps and K. C. Davis. Automating data warehouse conceptual schema design and evaluation. In *4th International Workshop on Design and Management of Data Warehouses (DMDW 2002), Toronto, Canada*, volume 58 of *CEUR Workshop Proceedings*, pages 23–32, 2002.

Call for papers: DATA4U 2014

First International Workshop on Bringing the Value of "Big Data" to Users
http://sites.google.com/site/data4u2014/
Hangzhou, China, in conjunction with VLDB 2014

IMPORTANT DATES

Abstracts due: June 2, 2014
Papers due: June 9, 2014
Notification: July 7, 2014
Camera-ready copies: July 21, 2014
Workshop: September 1, 2014

DESCRIPTION

The trend of bigger and bigger data --- in terms of volume, velocity, and variety --- is inevitable. Ultimately, how "big data" will impact the broad population of users rests on what value we can bring to them. Historically, the database community has focused primarily on efficient processing of structured queries posed by expert users on pre-organized data. But this focus only addresses one of the many different challenges in bringing the value of big data to users. Besides making queries and analysis faster and more scalable, we must address the pain points before and after analytics --- that is, how to put together data from diverse sources and "wrangle" it into representations appropriate for analyses, and how to communicate results and insights effectively. To broaden the impact of big data, we must also move beyond our traditional notions of "users," such as programmers and analysts, to a much wider range of new user archetypes, such as non-expert users who want to "get something" from their data, or ordinary citizens who wish to play a more active role in understanding public data.

The main goals of the workshop are to help expand the scope of database research to encompass a more complete picture of how to deal with big data, and to promote new, alternative viewpoints on what the database community should work on, if it is to play a bigger role in bringing the benefits of big data to the public. The workshop is designed to bring together researchers with similar interests, foster discussion of work in progress, encourage ideas that are "off the beaten track," and engage non-traditional users of database research. Our emphasis is on the importance of usability to a wider range of user archetypes. Other communities are also actively studying usability issues in big data; we believe that it is high time for the database community to begin contributing its expertise and perspectives to this important problem.

TOPICS OF INTEREST

We interpret "big data" broadly as data "too big for users." Areas of particular interest for the workshop encompass all aspects of dealing with big data, such as acquisition, cleansing, integration, "wrangling," access, querying, mining, statistical analysis, visualization, and storytelling with data. We call for research and experience papers, as well as for demonstration proposals, in the realm of these topics of interest. Papers describing systems, platforms, and applications with explicit foci on usability are especially welcome.

SUBMISSION GUIDELINES

Submissions should use the same document templates as PVLDB. The length of a workshop paper should not exceed 6 pages; the length of a demonstration proposal should not exceed 4 pages. A demonstration proposal should be clearly marked as such on its first page. All submissions will be handled electronically by the Conference Management Toolkit (CMT): https://cmt2.research.microsoft.com/DATA2014
The workshop proceedings will be published by VLDB and indexed via DBLP.

Workshop Co-Chairs

Rada Chirkova, NC State University, USA
Jun Yang, Duke University, USA

PC Members (tentative)

Azza Abouzied, NYU Abu Dhabi, United Arab Emirates
Minos Garofalakis, Technical University of Crete, Greece
H.V. Jagadish, University of Michigan, USA
Ioana Manolescu, INRIA, France
Eugene Wu, MIT, USA
Tiziana Catarci, Universita di Roma, Italy
Zachary Ives, University of Pennsylvania, USA
Chengkai Li, University of Texas at Arlington, USA
Li Eric Qian, Facebook, USA
Meihui Zhang, National University of Singapore, Singapore