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:
Donald Kossmann (Chair), Anastasia Ailamaki (Vice-Chair), Magdalena Balazinska, K. Selçuk Candan, Yanlei Diao, Curtis Dyreson, Christian Jensen, Yannis Ioannidis, and Tova Milo.

Advisory Board:
Raghu Ramakrishnan (Chair, Microsoft), 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, Wim Martens, Mirella Moro, Rachel Pottinger, and Jun Yang.

SIGMOD Record Editor-in-Chief:
Yanlei Diao, University of Massachusetts Amherst <yanlei AT cs.umass.edu>

SIGMOD Record Associate Editors:
Pablo Barceló, Vanessa Braganholo, Marco Brambilla, Chee Yong Chan, Rada Chirkova, Anastasios Kementsietsidis, Olga Papaemmanouil, Aditya Parameswaran, Anish Das Sarma, Alkis Simitsis, Nesime Tatbul, Marianne Winslett, and Jun Yang.

SIGMOD Conference Coordinator:
K. Selçuk Candan, Arizona State University

PODS Executive Committee: Rick Hull (Chair, IBM Research), Michael Benedikt, Wenfei Fan, Martin Grohe, Maurizio Lenzerini, Jan Paradaens.

Sister Society Liaisons:
Raghu Ramakhrishnan (SIGKDD), Yannis Ioannidis (EDBT Endowment), Christian Jensen (IEEE TKDE).

Awards Committee:
Umesh Dayal (Chair, Hitachi America Ltd.), Elisa Bertino, Surajit Chaudhuri, Masaru Kitsuregawa, and Maurizio Lenzerini.

Jim Gray Doctoral Dissertation Award Committee:
Tova Milo (Co-Chair, Tel Aviv University), Timos Sellis (Co-Chair, RMIT University), Ashraf Aboulnaga, Sudipto Das, Juliana Freire, Minos Garofalakis, Dan Suciu, Kian-Lee Tan.

[Last updated : March 31st, 2015]

SIGMOD Officers, Committees, and Awardees (continued)

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)
Jim Gray (1993)
Philip Bernstein (1994)
David DeWitt (1995)
C. Mohan (1996)
David Maier (1997)
Serge Abiteboul (1998)
Hector Garcia-Molina (1999)
Rakesh Agrawal (2000)
Rudolf Bayer (2001)
Patricia Selinger (2002)
Don Chamberlin (2003)
Ronald Fagin (2004)
Michael Carey (2005)
Jeffrey D. Ullman (2006)
Jennifer Widom (2007)
Moshe Y. Vardi (2008)
Masaru Kitsuregawa (2009)
Umeshwar Dayal (2010)
Surajit Chaudhuri (2011)
Bruce Lindsay (2012)
Stefano Ceri (2013)
Martin Kersten (2014)

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)
Gio Wiederhold (1995)
Yahiko Kambayashi (1995)
Jeffrey Ullman (1996)
Avi Silberschatz (1997)
Won Kim (1998)
Raghu Ramakrishnan (1999)
Michael Carey (2000)
Laura Haas (2000)
Daniel Rosenkrantz (2001)
Richard Snodgrass (2002)
Michael Ley (2003)
Surajit Chaudhuri (2004)
Hongjun Lu (2005)
Tamer Özsu (2006)
Hans-Jörg Schek (2007)
Klaus R. Dittrich (2008)
Beng Chin Ooi (2009)
David Lomet (2010)
Gerhard Weikum (2011)
Marianne Winslett (2012)
H.V. Jagadish (2013)
Kyu-Young Whang (2014)

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.
- **2014** *Winners*: Aditya Parameswaran, Stanford University, and Andy Pavlo, Brown University.

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

[Last updated : March 31st, 2015]

Editor's Notes

Welcome to the March 2015 issue of the ACM SIGMOD Record!

First of all, the ACM SIGMOD Record Editorial Board welcomes four new Associate Editors, who started their terms on January 1st, 2015.

- Anastasios Kementsietsidis, Google Research (http://research.google.com/pubs/AnastasiosKementsietsidis.html)
- Jun Yang, Duke University (https://www.cs.duke.edu/~junyang/)
- Olga Papaemmanouil, Brandeis University (http://www.cs.brandeis.edu/~olga/home.html)
- Aditya Parameswaran, University of Illinois Urbana-Champaign (http://web.engr.illinois.edu/~adityagp/)

In addition, the ACM TODS Editorial Board welcomes six new Associate Editors, as described in the message from Christian Jensen, the Editor-in-Chief of TODS.

Second, we are very pleased to feature Michael Stonebraker as the 2014 ACM Turing Award winner! Stonebraker, currently Professor at the Massachusetts Institute of Technology, will receive the 2014 Turing Award for fundamental contributions to the concepts and practices underlying modern database systems. The article by Bruce Shriver and Marie Gentile in this issue of the SIGMOD Record has more on Stonebraker's contributions and the ACM Turing Award.

The issue continues with a Database Principles article by Luc Segoufin, which surveys recent results on the topic of enumerating answers to queries over a database. A new way to formulate the query evaluation process is to assume that tuples of the query result can be generated one by one with some regularity, for example by ensuring a fixed delay between two consecutive outputs once some necessary precomputation has been performed to construct a suitable index structure. This article focuses on such a case that the enumeration is performed with a constant delay between any two consecutive solutions, after linear-time preprocessing. While such constant delay enumeration cannot be always achieved, this article considers conjunctive queries and describes several scenarios in which it is indeed possible.

The Research and Vision Articles Column features a vision article, by Power et al., on "Implications of Emerging 3D GPU Architecture on the Scan Primitive". This article delivers a main message that as one projects into the future and examines 3D die-stacked systems, highly data-parallel architectures, such as General Purpose GPUs, can provide a large benefit over current CPU platforms as well as 3D-stacked CPU systems. It is a timely publication that calls for the attention of the database community to embrace high-bandwidth, highly data parallel architectures and perhaps rethink of database design in the future.

The Surveys Column features a survey by Kantorski et al. on "Automatic Filling of Hidden Web Forms". Today, a significant portion of the information on the Web is stored in online databases, in so-called Hidden Web and Deep Web. Access to information in the Hidden Web requires filling an HTML form that is submitted as a query to the underlying database. This article surveys a number of recent works on how to automate the process of filling forms by choosing appropriate values to fill the fields and retrieving nonempty result sets. Since this can be a challenging task due to the wide variety of forms and lack of prior knowledge of valid values for each field, this article presents 15 methods that are most influential in Web form filling, offering a good coverage and categorization of relevant techniques.

The Distinguished Profiles column features two winners of 2014 SIGMOD Jim Gray Doctoral Dissertation Award: Aditya Parameswaran and Andy Pavlo. Aditya graduated from Standard University, with a dissertation entitled "Human-Powered Data Management," and now is an assistant professor at the University of Illinois (UIUC). In the interview, Aditya speaks about his work, its impact in industry, and how he became determined to be an academic. Andy graduated from Brown University with a dissertation "On Scalable Transaction Execution in Partitioned Main Memory Database Systems." Andy is now an assistant professor at Carnegie Mellon University. In the interview, Andy talks about his work and his decision to stay focused on this single project in graduate school, which led to the success of his dissertation.

In the Research Centers Column, Stratos Idreos describes several areas of ongoing research at the DASlab at the Harvard School of Engineering and Applied Sciences. The long-term goal of DASlab is to assist in minimizing the time it takes to turn data into knowledge by designing and building novel data systems, tailored for the new and ever-evolving challenges of a data-driven world. Towards this goal, DASlab has ongoing research projects on self-designing data systems, auto-exploration systems, interactive and visual analytics, indexing in modern data systems, and hardware software co-design.

This issue features an event report, by Christophides and Palpanas, on the First International Workshop on Personal Data Analytics in the Internet of Things (PDA@IOT 2014). As one witnesses an increasing number of devices with embedded sensors and actuators becoming pervasive in everyday life, this workshop aims to spark research on data analytics, especially on how individual people can effectively exploit the data that they massively create in Cyber-Physical worlds. The workshop succeeded with two keynote talks, seven research papers, and a panel discussion.
Finally, the issue closes with the call for papers for the Sixth ACM Symposium on Cloud Computing (SoCC 2015), to be co-located with VLDB 2015, and call for participation for the Federated Computing Research Conference (FCRC 2015).

On behalf of the SIGMOD Record Editorial board, I hope that you all enjoy reading the March 2015 issue of the SIGMOD Record!

Your submissions to the Record are welcome via the submission site:
http://sigmod.hosting.acm.org/record

Prior to submitting, please read the Editorial Policy on the SIGMOD Record's Web site:
http://www.sigmod.org/publications/sigmod-record/sigmod-record-editorial-policy

Yanlei Diao

March 2015

Past SIGMOD Record Editors:

Ioana Manolescu (2009-2013)
Ling Liu (2000 – 2004)
Arie Segev (1989 – 1995)
Thomas J. Cook (1981 – 1983)
Daniel O'Connell (1971 – 1973)
Alexandros Labrinidis (2007 – 2009)
Michael Franklin (1996 – 2000)
Margaret H. Dunham (1986 – 1988)
Douglas S. Kerr (1976-1978)
Harrison R. Morse (1969)
Mario Nascimento (2005 – 2007)
Jennifer Widom (1995 – 1996)
Jon D. Clark (1984 – 1985)
Randall Rustin (1974-1975)

Changes to the TODS Editorial Board

Christian S. Jensen
csj@cs.aau.dk

It is of paramount importance for a scholarly journal such as *ACM Transactions on Database Systems* to have a strong editorial board of respected, world-class scholars. The editorial board plays a fundamental role in attracting the best submissions, in ensuring insightful and timely handling of submissions, in maintaining the high technical standards of the journal, and in maintaining the reputation of the journal. Indeed, the journals Associate Editors, along with the reviewers and authors they work with, are the primary reason that TODS is such a respected journal.

Retiring Associate Editors

As of February 1, 2015, four Associate Editors, George Kollios, Paul Larson, Yufei Tao, and Johannes Gehrke, have ended their terms, having served on the editorial board for seven, seven, seven, and four years, respectively. (They will stay on until they complete their current loads.) They have each provided very substantial, high-quality service to the journal and the database community. Specifically, I have never seen them compromise on quality when handling submissions, and I believe that they have uniformly made sound technical decisions. We are fortunate that they have donated their time and world-class expertise during these years.

New Associate Editors

Also as of February 1, 2015, six new Associate Editors have joined the editorial board:

- K. Selcuk Candan (http://aria.asu.edu/candan/)
- Amol Deshpande (http://www.cs.umd.edu/~amol)
- Daniel Kifer (http://www.cse.psu.edu/~dkifer)
- Mohamed Mokbel (http://www-users.cs.umn.edu/~mokbel)
- Jun Yang (http://www.cs.duke.edu/~junyang)
- Ke Yi (http://www.cse.ust.hk/~yike)

All six are highly regarded scholars in the field of database systems. We are very fortunate that these outstanding scholars are willing to volunteer their valuable time and indispensable expertise for handling manuscripts for the benefit of our scientific community. Indeed, I am gratified that they have committed to help TODS continue to improve, and I am looking forward to working with them.

Michael Stonebraker Wins the 2014 ACM Turing Award

Michael Stonebraker
Massachusetts Institute of Technology

Association for Computing Machinery

Advancing Computing as a Science & Profession

Contact: Bruce Shriver
212-626-0521
shriver@hq.acm.org

Marie Gentile
646-213-7249 or 917-679-6299
marie.gentile@finnpartners.com

ACM TURING AWARD GOES TO PIONEER IN DATABASE SYSTEMS ARCHITECTURE

MIT's Stonebraker Brought Relational Database Systems from Concept to Commercial Success, Set the Research Agenda for the Multibillion-Dollar Database Field for Decades

NEW YORK, March 25, 2015 – ACM, the Association for Computing Machinery, (www.acm.org) today named Michael Stonebraker of the Massachusetts Institute of Technology (MIT) recipient of the 2014 ACM A.M. Turing Award for fundamental contributions to the concepts and practices underlying modern database systems. Database systems are critical applications of computing and preserve much of the world's important data. Stonebraker invented many of the concepts that are used in almost all modern database systems. He demonstrated how to engineer database systems that support these concepts and released these systems as open software, which ensured their widespread adoption. Source code from Stonebraker's systems can be found in many modern database systems. During a career spanning four decades, Stonebraker founded numerous companies successfully commercializing his pioneering database technology work.

The ACM Turing Award, widely considered the "Nobel Prize in Computing," carries a $1 million prize with financial support provided by Google, Inc. It is named for Alan M. Turing, the British mathematician who articulated the mathematical foundation and limits of computing.

"Michael Stonebraker's work is an integral part of how business gets done today," said ACM President Alexander L. Wolf. "Moreover, through practical application of his innovative database management technologies and numerous business start-ups, he has continually demonstrated the role of the research university in driving economic development."

"The efficient and effective management of Big Data is crucial to our 21st century global economy," said Google Senior Vice President of Knowledge Alan Eustace. "Michael Stonebraker invented many of the architectures and strategies that are the foundation of virtually all modern database systems."

Database systems are among the most commercially successful software systems and are an essential part of the infrastructure of our global economy today. They are indispensable to business management, transaction processing, data analysis and electronic commerce, to name a few. Stonebraker is responsible for much of the software foundation of modern database systems and originated many of the key concepts of data management used in nearly all database systems today.

Stonebraker developed Ingres, proving the viability of the relational database theory. Ingres was one of the first two relational database systems (the other was IBM System R). With Ingres, Stonebraker made major contributions including query language design, query processing techniques, access methods, and concurrency control, and showed that query rewrite techniques (query modification) could be used to implement relational views and access control.

Stonebraker introduced the object-relational model of database architecture with the release of Postgres, integrating important ideas from object-oriented programming into the relational database context.

Postgres extended the relational database model, enabling users to define, store and manipulate rich objects with complex state and behavior.

Concepts introduced in Ingres and Postgres can be found in nearly all major database systems today. Ingres and Postgres were well engineered, built on UNIX, released as open software, and form the basis of many modern commercial database systems including Illustra, Informix, Netezza and Greenplum.

Stonebraker also developed lasting technical results on distributed query processing and transaction coordination protocols through development of Distributed Ingres, one of the first distributed database systems. Another highly influential project was XPRS, a parallel version of Postgres that explored the "shared nothing" approach to parallel database management. Mariposa, a massively-distributed federated database system, explored ideas such as opportunistic data replication and decentralized query processing.

Stonebraker set the course for the design of scalable data systems as an early advocate for the adoption of the shared nothing database architecture. This approach is widely viewed as the only way to achieve and maintain scale, and is employed by nearly every major database vendor and "big data" solution today.

More recently, Stonebraker has been an advocate of the "no size fits all" approach to database systems architecture and has developed database architectures for specialized purposes. He pioneered real-time processing over streaming data sources (Aurora/StreamBase). His work on column-oriented storage architecture resulted in systems optimized for complex queries (C-Store/Vertica). He developed a high throughput, distributed main-memory online transaction processing system (H-Store/VoltDB). Stonebraker has also developed an extreme-scale data management and data analysis system for science (SciDB).

Background

Michael Stonebraker is adjunct professor at the Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory (MIT CSAIL) where he is also co-founder and co-director of the Intel Science and Technology Center for Big Data. Prior to MIT, Stonebraker was professor of computer science at the University of California at Berkeley for 29 years. A graduate of Princeton University, Stonebraker earned his master's degree and his Ph.D. from the University of Michigan.

Stonebraker received the Software System Award with Gerald Held and Eugene Wong for the development of Ingres (IBM's System R was also recognized). He was the recipient of the inaugural SIGMOD Edgar F. Codd Innovations Award, and received the IEEE John von Neumann Medal. Stonebraker is an ACM Fellow and a member of the U.S. National Academy of Engineering.

Throughout his career, Stonebraker has proven the practical application of his research, founding several successful companies to commercialize his work, including Ingres Corporation (acquired by ASK and then Computer Associates), Illustra Corporation (acquired by Informix), Cohera Corporation (acquired by PeopleSoft), Streambase, Inc. (acquired by Tibco), Vertica Systems, Inc. (acquired by HP), VoltDB, Paradigm4 and Tamr, Inc.

ACM will present the 2014 A.M. Turing Award at its annual Awards Banquet on June 20 in San Francisco, Calif.

About the ACM A.M. Turing Award

The A.M. Turing Award **http://amturing.acm.org/** was named for Alan M. Turing, the British mathematician who articulated the mathematical foundation and limits of computing, and who was a key contributor to the Allied cryptanalysis of the German Enigma cipher and the German "Tunny" encoding machine in World War II. Since its

inception in 1966, the Turing Award has honored the computer scientists and engineers who created the systems and underlying theoretical foundations that have propelled the information technology industry.

About ACM

ACM, the Association for Computing Machinery **www.acm.org,** is the world's largest educational and scientific computing society, uniting computing educators, researchers and professionals to inspire dialogue, share resources and address the field's challenges. ACM strengthens the computing profession's collective voice through strong leadership, promotion of the highest standards, and recognition of technical excellence. ACM supports the professional growth of its members by providing opportunities for life-long learning, career development, and professional networking.

# # #

Constant delay enumeration for conjunctive queries

Luc Segoufin
INRIA and ENS Cachan

ABSTRACT

We survey some of the recent results about enumerating the answers to queries over a database. We focus on the case where the enumeration is performed with a constant delay between any two consecutive solutions, after a linear time preprocessing.

This cannot be always achieved. It requires restricting either the class of queries or the class of databases.

We consider conjunctive queries and describe several scenarios when this is possible.

1. INTRODUCTION

The evaluation of queries is a central problem in database management systems. Given a query q and a database $\mathcal{D}$ the evaluation of q over $\mathcal{D}$ consists in computing the set $q(\mathcal{D})$ of all answers to q on $\mathcal{D}$. The complexity of this problem has been widely studied. However most of the complexity bounds are extrapolated from the boolean case (aka the model checking problem) and expressed as a function of the sizes of q and $\mathcal{D}$. For non boolean queries it may be not satisfactory enough to express complexity results just in terms of the sizes of $\mathcal{D}$ and q. A simple observation shows that the set $q(\mathcal{D})$ may be huge, even larger than the database itself, as it can have a number of elements of the form $\|\mathcal{D}\|^l$, where $\|D\|$ is the size of the database and l the arity of the query. The fact that the solution set $q(\mathcal{D})$ may be of size exponential in the query is intuitively not sufficient to make the problem hard, and alternative complexity measures had to be found for query answering. For instance one could consider output-sensitive complexity measures expressed as a function of the sizes of q, $\mathcal{D}$ but also $q(\mathcal{D})$. In this direction, one way to define tractability is to assume that tuples of the query result can be generated one by one with some regularity, for example by ensuring a fixed delay between two consecutive outputs once a necessary precomputation has been done to construct a suitable index structure.

This approach, that considers query answering as an enumeration problem, has deserved some attention over the last few years. In this vein, the best that one can hope for is constant delay, i.e., the delay depends only on the size of q (but not on the size of $\mathcal{D}$). A number of query evaluation problems have been shown to admit constant delay algorithms, usually preceded by a preprocessing phase that is linear in the size of the database. We survey some of these results in this paper.

This imposes drastic constraints. In particular, the first answer is output after a time linear in the size of the database and once the enumeration starts a new answer is being output regularly at a speed independent from the size of the database. Altogether, the set $q(\mathcal{D})$ is entirely computed in time $f(q)(\|\mathcal{D}\| + |q(\mathcal{D})|)$ for some function f depending only on q and not on $\mathcal{D}$. In particular boolean queries can be evaluated in time linear in the size of the database. However, as shown in [5], the fact that evaluation of boolean queries is easy does not guarantee the existence of such efficient enumeration algorithms in general: under some reasonable complexity assumption, there is no constant delay algorithm with linear preprocessing enumerating the answers of acyclic conjunctive queries, although it is well-known that the model-checking of boolean acyclic queries can be done in linear time [29].

We stress that our study is theoretical. If most of the algorithms we will mention here are linear in the size of the database, the multiplicative factors are often very big, making any implementation difficult. However, we believe that the index structures designed for making these algorithms work are interesting and, with extra assumptions, could possibly be turned into something practical.

In this paper we concentrate on conjunctive queries, possibly with negated atoms. We will see how various forms of acyclicity play here a crucial role. Modulo reasonable complexity assumptions, we are actually able to characterize precisely those acyclic

conjunctive queries that can be enumerated with constant delay.

There are many related problems. Typically one could imagine computing the top-ℓ most relevant answers relative to some ranking function or to provide a sampling of $q(\mathcal{D})$ relative to some distribution. One could also imagine computing only the number of solutions $|q(\mathcal{D})|$ or providing an efficient test for whether a given tuple belongs to $q(\mathcal{D})$ or not. It is not clear a priori how these problems are related to constant delay enumeration. However, it turns out that in the scenarios where constant delay enumeration can be achieved, one can often also count the number of solutions in time linear in the size of the database and, after linear time preprocessing on the database, one can test in constant time whether a given tuple is part of the answers set.

This survey is by no means exhaustive. It is only intended to survey the major theoretical results concerning conjunctive queries and enumeration. Hopefully it will convince the reader that this is an important subject for research that still contains many interesting and challenging open problems.

Enumeration in general, and constant delay enumeration in particular, is a well identified subfield of algorithmics, and many non trivial enumeration algorithms exist for problems over graphs (like enumerating all spanning trees, all connected components, all cycles etc...) We will not discuss those results at all here.

2. PRELIMINARIES

2.1 Database and queries

In this paper a database is a finite relational structure. A *relational signature* is a tuple

$$\sigma = (R_1, \cdots, R_l),$$

each R_i being a relational symbol of arity r_i. A *relational structure* over σ is a tuple

$$\mathcal{D} = \left(D, R_1^{\mathcal{D}}, \ldots, R_l^{\mathcal{D}}\right),$$

where D is the *domain* of $\mathcal{D}$ and $R_i^{\mathcal{D}}$ is a subset of D^{r_i}. We define the *size* of $\mathcal{D}$ as

$$\|\mathcal{D}\| = |\sigma| + |D| + \sum_{R_i} |R_i^{\mathcal{D}}| r_i.$$

It corresponds to the size of a reasonable encoding of $\mathcal{D}$. The number of elements in the domain of $\mathcal{D}$ is denoted by $|\mathcal{D}|$.

A *query* takes as input a database of a given signature σ and returns a relation of a fixed arity, *the arity of the query*. A query is *boolean* if its arity is 0. The query is then either true or false on $\mathcal{D}$ and defines a property of $\mathcal{D}$. A query is *unary* if its arity is 1. If q is a query and $\bar{a}$ is in the image of q on $\mathcal{D}$, then we write $\mathcal{D} \models q(\bar{a})$. Finally we set

$$q(\mathcal{D}) = \{\bar{a} \mid \mathcal{D} \models q(\bar{a})\}.$$

Note that the size of $q(\mathcal{D})$ may be exponential in the arity of q. A query language is a class of queries. Typically it is defined as a logical formalism such as CQ (for *conjunctive* queries), FO (for *first-order* queries), MSO (for *monadic second-order* queries) and so on. As usual, $|q|$ denotes the size of q.

2.2 Model of computation

We use Random Access Machines (RAM) with addition and uniform cost measure as a model of computation, cf. [1]. Our algorithms will take as input a query q of size k and a database $\mathcal{D}$ of size n. We then say that an algorithm runs in *linear time* if it ends within $f(k)n$ steps, for some function f. It runs in *quasi-linear* time if it ends within $f(k)n \log n$ steps. It runs in *constant time* if it ends in $f(k)$ steps.

Given an $n \times n$ matrix, and two numbers $i, j \leq n$ the RAM model returns the content to the entry (i, j) of the matrix in constant time. Therefore when given the adjacency matrix of a graph it can test in constant time where two given nodes are adjacent or not. However our databases are encoded by the list of their tuples and we therefore do not have access to the adjacency matrix. Testing whether a tuple belongs to a relation may therefore require more than a constant time.

In the sequel we assume that the input database comes with a linear order on the domain. If not, we use the one induced by the encoding of the database as an input to the RAM. Whenever we iterate through all nodes of the domain, the iteration is with respect to the initial linear order.

An important observation is that the RAM model can sort m elements of size $O(\log m)$ in time $O(m \log m)$ [18]. In particular, we can sort lexicographically the tuples of a relation in linear time. As a consequence, a simple merge-sort algorithm we can compute the relation $\{\bar{x}\bar{y} \mid R(\bar{x}\bar{y}) \wedge S(\bar{x})\}$ in time linear in the sizes of R and S.

2.3 Parametrized complexity

The database $\mathcal{D}$ and the query q play different roles as input of our problems. It is often assumed that $|\mathcal{D}|$ is large while $|q|$ is small. Hence it is useful to distinguish them in the input of the query answering problem. Parametrized complexity is a suitable framework for analyzing such situations.

We only provide here the basics of parametrized complexity needed for understanding this paper. The interested reader is referred to the monograph [17].

We view the problem of boolean query evaluation as a parametrized problem where the input is a database $\mathcal{D}$ and a boolean query q, the parameter is $|q|$ and the problem asks whether $\mathcal{D} \models q$.

A parametrized problem is Fixed Parameter Tractable, i.e. can be solved in FPT, if it can be solved in time $f(q)\|\mathcal{D}\|^c$ for some suitable computable function f and constant c. The idea behind this definition is that it is often preferable to have an algorithm working in $2^{|q|}\|\mathcal{D}\|^2$ rather than $\|\mathcal{D}\|^{|q|}$.

In parametrized complexity there is also a suitable notion of reduction, called FPT-reduction. FPT is closed under FPT-reductions and there are some hard classes of parametrized problems, closed under FPT-reductions, containing problems with no known FPT algorithms and that are believed to be different from FPT.

We distinguish two important hard classes denoted W[1] and AW[$*$]. W[1] plays in parametrized complexity the role of NP in classical complexity. A typical problem which is complete for W[1] is the parametrized boolean query evaluation problem for CQ [24]. AW[$*$] plays in parametrized complexity the role of PSpace in classical complexity. A typical problem which is complete for AW[$*$] is the parametrized boolean query evaluation problem for FO [24].

2.4 The enumeration class CD∘Lin

Let $\mathcal{L}$ be a query language and $\mathcal{C}$ be a class of databases. We say that the *enumeration problem* for $\mathcal{L}$ over $\mathcal{C}$ can be solved with *constant delay after linear preprocessing* (is in CD∘Lin), if it can be solved by a RAM algorithm which, on input $q \in \mathcal{L}$ and $\mathcal{D} \in \mathcal{C}$, can be decomposed into two phases:

- a preprocessing phase that is performed in time linear in the size of the database, and
- an enumeration phase that outputs $q(\mathcal{D})$ with no repetition and a delay depending only on q between any two consecutive outputs. The enumeration phase has full read access to the output of the preprocessing phase and can use extra memory whose size depends only on q.

The definition of CD∘Lin requires a preprocessing time linear in $\|\mathcal{D}\|$ and a delay not depending on $\mathcal{D}$. There are hidden multiplicative factors that are functions on the size of the query. These factors may be huge. We will refer to them in the sequel as the *multiplicative factors*.

Before we proceed with the technical presentation of the results, it is worth spending some time with examples.

Example 1. *Consider a database schema containing a binary relational symbol R and the query*

$$q(x, y) := \neg R(x, y).$$

On input database $\mathcal{D}$, the following simple algorithm enumerates $q(\mathcal{D})$:

> Go through all pairs (a, b);
> test if it is a fact of $R^{\mathcal{D}}$;
> if so skip this pair;
> otherwise output it.

However, a simple complexity analysis shows that the delay between any two outputs is not constant. There are two reasons for this. First, arbitrarily long sequences of pairs can be skipped. Second, it is not obvious how to test whether $(a, b) \in R^{\mathcal{D}}$ in constant time (i.e. without going through the whole relation $R^{\mathcal{D}}$). In order to enumerate this query with constant delay it is necessary to perform a preprocessing computing an index structure that can be used for enumeration. This is done as follows.

We first decide on an arbitrary linear order on the domain of $\mathcal{D}$. We then order all $R^{\mathcal{D}}$ according to the lexicographical order. Recall that with the RAM model this can be done in time linear in $\|\mathcal{D}\|$.

We then compute for each tuple $\bar{u}$ of $R^{\mathcal{D}}$ the tuples $\bar{v} = f(\bar{u})$ and $\bar{v}' = g(\bar{u})$ such that $\bar{v}$ is the smallest (relative to the lexicographical order) element not in $R^{\mathcal{D}}$ that is bigger than $\bar{u}$ (hence all tuples between $\bar{u}$ and $\bar{v}$ are in $R^{\mathcal{D}}$) and $\bar{v}'$ is the smallest (relative to the lexicographical order) element in $\in R^{\mathcal{D}}$ that is bigger than $\bar{v}$. These functions can be computed in time linear in $\|\mathcal{D}\|$ by a simple pass on the ordered list of $R^{\mathcal{D}}$ from its last element to the first one.

This concludes the preprocessing phase, the index consists in those precomputed functions. Note that the RAM model is such that once a function h is computed, on input $\bar{u}$, $h(\bar{u})$ is returned in constant time.

Using the precomputed functions, the enumeration phase is now simple. We maintain two pairs of elements of $\mathcal{D}$: one is initialized with the smallest pair according to the lexicographical order, the other one with the smallest pair in $R^{\mathcal{D}}$. The second pair will always be pointing to an element of $R^{\mathcal{D}}$. Assuming the current pairs are $\langle \bar{u}, \bar{v} \rangle$, we then do the following until $\bar{u}$ is maximal.

If $\bar{u} = \bar{v}$ then we move to $\langle f(\bar{v}), g(\bar{v}) \rangle$. Note that f and g are such that for all $\bar{v}$, $f(\bar{v}) \neq g(\bar{v})$.

If $\bar{u} \neq \bar{v}$ we output $\bar{u}$ and replace it by its successor in the lexicographical order without changing $\bar{v}$.

This algorithm is constant delay as an output is performed at least every other step and each step can be performed in constant time as all the relevant functions have been precomputed. All output tuples are clearly not in $R^{\mathcal{D}}$ and the reader can check that all skipped tuples are in $R^{\mathcal{D}}$.

EXAMPLE 2. *Same schema but the query is now computing the pairs of nodes at distance 2:*

$$q(x, y) := \exists z R(x, z) \wedge R(z, y).$$

We will see in Section 3 that it is likely that this query cannot be enumerated with constant delay. However, if we assume that R has degree bounded by d, then for any node u of the graph, at most d^2 nodes v are at distance 2 from u. Moreover, it is easy to see that we can compute in time linear in $\|\mathcal{D}\|$ the function $f(u)$ associating to u the list of its nodes at distance 1. An extra linear pass based on the function f computes the function $g(u)$ associating to u the list of its nodes at distance 2. From there the enumeration algorithm with constant delay is trivial.

REMARK 1. Notice that if the enumeration problem for $\mathcal{L}$ over $\mathcal{C}$ is in CD$\circ$LIN, then all answers can be output in time $O(\|\mathcal{D}\| + |q(\mathcal{D})|)$ and the first output is computed in time linear in $\|\mathcal{D}\|$. In particular the evaluation problem for boolean queries of $\mathcal{L}$ is in FPT. Hence unless FPT = W[1] any language $\mathcal{L}$ whose evaluation problem for boolean queries is hard for W[1] cannot be enumerated in CD$\circ$LIN. In particular this holds for CQ and FO.

Notice that if the arity of q is less or equal to 1, then $|q(\mathcal{D})| \leq |\mathcal{D}| \leq \|\mathcal{D}\|$. It is then plausible that the whole set of answers can be computed in time linear in $\|\mathcal{D}\|$. If this is the case then we have a simple constant delay algorithm that precomputes all answers during the precomputation phase and then scans the set of answers and outputs them one by one during the enumeration phase. Hence enumeration often becomes relevant when the arity of q is at least 2. In this case $q(\mathcal{D})$ can be quadratic in $\|\mathcal{D}\|$ and hence can certainly not be computed within the linear time constraint of the precomputation phase. The index structure built during the preprocessing phase is then a non trivial object. One can also view this index structure as a compact (of linear size) representation of the set $q(\mathcal{D})$ (that can be of polynomial size) and the enumeration algorithm as an output streaming decompression algorithm.

3. CONJUNCTIVE QUERIES AND ENUMERATION

In this section we consider the evaluation of conjunctive queries with possibly negated atoms. We start with the case with no negated atoms.

3.1 Conjunctive queries

Recall that a conjunctive query (CQ) is a query of the form

$$q(\bar{x}) := \exists y_1 \cdots \exists y_l \quad \bigwedge_i R_i(\bar{z}_i)$$

where $R_i(\bar{z}_i)$ is an *positive atom* of q, R_i being a relational symbol and $\bar{z}_i$ containing variables from $\bar{x}$ or $\bar{y}$.

A typical example is the distance 2 query of Example 2. Another example is the query returning all triangles in a graph.

As evaluating boolean conjunctive queries is hard for W[1], we restrict our attention to *acyclic* conjunctive queries that can be evaluated in time $|q| \cdot \|\mathcal{D}\| \cdot |q(\mathcal{D})|$ [29]. We will see that it is very unlikely that constant delay enumeration can be done even for acyclic conjunctive queries. It is only achieved for a subset of them called *free-connex*. We start with the necessary definitions.

To a conjunctive query q, we associate an hypergraph $H(q) = (V, S)$ whose vertices V are the variables of q and whose hyperedges S are the set of variables occurring in a single atom of q, i.e. $S = \{\{x_1, \cdots, x_p\} \mid R(x_1, \cdots, x_p) \text{ is an atom of } q\}$.

A *join tree* of $q \in$ CQ is a tree T whose nodes are atoms of q and such that

(i) each atom of q is the label of exactly one node of T,

(ii) for each variable x of q, the set of nodes of T in which x occurs is connected.

A conjunctive query q is said to be *acyclic* if it has a join tree. In graph theoretical terms this is equivalent to saying that the hypergraph $H(q)$ is *α-acyclic.*

A boolean query associated to a join tree can be evaluated in time linear in $\|\mathcal{D}\|$ using a simple bottom-up traversal of the join tree. If the query is non boolean, the possible valuations of the free variables need to be stored at each step, hence a multiplicative extra factor of $|q(\mathcal{D})|$. The result of [29] follows.

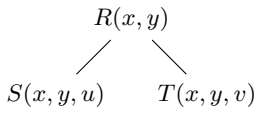
An acyclic conjunctive query $q(\bar{x})$ is said to be *free-connex* if the query $q(\bar{x}) \wedge R(\bar{x})$ is also acyclic, where $\bar{x}$ are the free variables of q and R is a new

symbol of appropriate arity[1]. Note that all boolean acyclic queries are free-connex.

For example the acyclic conjunctive query

$$q(x, y) := \exists u, v \quad S(x, y, u) \wedge T(x, y, v)$$

is free-connex because the following join tree shows acyclicity of the extended query:



However the distance 2 query

$$q(x, y) := \exists z \quad S(x, z) \wedge S(z, y)$$

is acyclic but not free-connex as the query

$$\exists z \quad S(x, z) \wedge S(z, y) \wedge R(x, y)$$

is clearly cyclic.

Free-connexity implies the existence of a join tree where all the free variables occur at the root. Hence the bottom-up traversal of the join tree can be performed without having to remember the valuations of the free variables until the last steps. The multiplicative factor of $|q(\mathcal{D})|$ then become an additive factor. With little extra effort this can be turned into a constant delay enumeration algorithm.

THEOREM 1. *[5] The enumeration for free-connex acyclic conjunctive queries is in* CD∘LIN.

We stress that the multiplicative factors involved in Theorem 1 are polynomial in the query size.

The result of Theorem 1 also holds if the queries contain inequalities. In this case atoms with inequalities are not involved when building the (generalized) join trees. In the presence of inequalities, the multiplicative factors are now exponential in the query size.

It turns out that free-connexity characterizes exactly those acyclic queries that can be enumerated in constant delay, assuming boolean matrix multiplication cannot be done in quadratic time. Boolean matrix multiplication is the problem of given two $n \times n$ matrices with boolean entries M, N to compute their product MN. The best known algorithms so far (based on the Coppersmith–Winograd algorithm [11]) require more than $n^{2.37}$ steps.

THEOREM 2. *[5] If boolean matrix multiplication cannot be done in quadratic time then the following are equivalent for acyclic conjunctive queries q:*

1. *q is free-connex*
2. *q can be enumerated in* CD∘LIN
3. *q can be evaluated in time $O(\|\mathcal{D}\| + |q(\mathcal{D})|)$.*

In particular the distance 2 query cannot be enumerated with constant delay after linear time preprocessing unless boolean matrix multiplication can be done in quadratic time.

With a stronger hypothesis we can even show that acyclicity itself is necessary for having constant delay enumeration. This hypothesis requires that it is not possible to test the existence of a triangle in a hypergraph of n vertices in time $O(n^2)$ and that for any k testing the presence of a k-dimensional tetrahedron cannot be tested in linear time (see [9] for precise definitions).

THEOREM 3. *[9] If the above hypothesis is true then the following are equivalent for $q \in$* CQ*:*

1. *q is acyclic free-connex*
2. *q can be enumerated in* CD∘LIN

3.2 Signed conjunctive queries

We are now interested in evaluating signed conjunctive queries. Those extend the syntax of conjunctive queries by allowing negated atoms. In other words they are of the form

$$q(\bar{x}) := \exists \bar{y} \quad q^+(\bar{x}\bar{y}) \wedge q^-(\bar{x}\bar{y})$$

where q^+ is a conjunction of positive atoms while q^- is a conjunction of negated atoms.

When q^- is empty we have seen in the previous section that q can be enumerated with constant delay after a linear preprocessing as soon as $H(q^+)$ is α-acyclic and q^+ free-connex. When q^+ is empty it has been shown in [8, 9] that constant delay enumeration can be achieved if $H(q^-)$ is *β-acyclic* and q^- free-connex. β-acyclicity is the hereditary extension of α-acyclicity. It requires that the hypergraph and all its subhypergraphs are α-acyclic. When neither q^+ nor q^- are empty then a notion of *signed-acyclicity* was introduced in [9]. It yields α-acyclicity and β-acyclicity in the corresponding limit cases. It also allows for tractable enumeration algorithms.

THEOREM 4. *[9] The enumeration for free-connex signed-acyclic conjunctive queries can be done with constant delay after a preprocessing time of the form $\|\mathcal{D}\|(\log \|\mathcal{D}\|)^{|q|}$.*

The enumeration for free-connex signed-acyclic conjunctive queries can be done with logarithmic delay after a quasi-linear time preprocessing.

[1]This is not the initial definition of free-connex as given in [5]. This presentation is from Brault-Baron [9]

The multiplicative factors are exponential in the size of the query for the constant delay result but polynomial in the logarithmic delay result. As in the previous section, modulo complexity hypothesis, typically that testing the existence of a triangle cannot be done in $O(n^2 \log n)$ time on a graph of size n, the signed-acyclicity hypothesis and the free-connexity hypothesis cannot be avoided [9].

3.3 Longer delay

We could consider enumeration algorithms allowing for non constant delay.

Delay linear in the size of the database. In this setting, the preprocessing phase remains linear in the size of the database but the delay between any two consecutive outputs is now linear in the size of the database. Notice that linear delay still implies that the associated model checking problem is in FPT, hence CQ cannot be enumerated with linear delay unless W[1] =FPT.

One can then consider restricting the class of databases. A class of databases, called $\underline{\text{X}}$-databases, has been exhibited such that CQ can be enumerated over it with linear delay. We will not define $\underline{\text{X}}$-databases in this note. Typical examples are grids and trees with all XPath axis.

THEOREM 5. *[4]. The enumeration for* CQ *over $\underline{\text{X}}$-structures can be done with linear delay.*

For acyclic conjunctive queries linear delay enumeration can be obtained with no restriction on the databases.

THEOREM 6. *[5]. The enumeration for acyclic* CQ *over all databases can be done with linear delay.*

Polynomial delay. One could also allow polynomial precomputation and polynomial delay. This notion is maybe less relevant in the database context. Indeed, the degree of the polynomial could depend on the size of the query and in this case the preprocessing phase can often precompute all solutions. This notion is however relevant when considering first-order queries with free second-order variables. In this case, for Σ_1-queries, polynomial delay enumeration can be achieved [16].

4. NEARBY PROBLEMS

It turns out that the index structures build for enumeration can be used with little modifications for solving several related problems, like counting the number of solutions, or in the presence of an order, directly pointing to the j^{th}-solution. We briefly survey those results here.

Counting the number of solutions.

Given a query q and a database $\mathcal{D}$, the counting problem is to compute $|q(\mathcal{D})|$.

Given a query language $\mathcal{L}$, we say that the counting problem of $\mathcal{L}$ is solvable in time $f(n)$ if for any $q \in \mathcal{L}$ and any database $\mathcal{D}$, $|q(\mathcal{D})|$ can be computed in time $g(q)f(\|\mathcal{D}\|)$ for some computable function g. Note that f does not depend on q. If f is polynomial then the associated parametrized computational problem is in the class FPT.

For conjunctive queries, actually even for acyclic conjunctive queries, counting the number of solutions of a query is a hard problem. Already for acyclic conjunctive queries the combined complexity is $\#P$-complete [25] and only the quantifier free acyclic conjunctive queries can be solved in time linear in $\|\mathcal{D}\|$ [3]. Adding just one quantifier already make already the problem hard [26].

For this reason, [14] introduced a new parameter named *quantified-star size*. It measures "how the free variables are spread in the formula" and bounding this parameter yields tractable counting problem for acyclic conjunctive queries.

THEOREM 7. *[14] For each number s, the counting problem for acyclic conjunctive queries of quantified-star size bounded by s can be solved in time polynomial in both the size of the query and of the database.*

It turns out that this parameter characterizes exactly the class of acyclic conjunctive queries having a tractable counting problem. If a class of acyclic conjunctive query does not have a bounded quantified-star size, then its associated counting problem is #W[1]-hard [14]. In particular, it cannot be solved in FPT.

It is possible to perform counting efficiently beyond acyclic conjunctive queries. For instance it is known that the boolean case is tractable for CQ having *bounded width* for various notions of width. In order to capture also non boolean queries the notion of *quantified-star size* was extended for various notions of width [13]. Based on this definition for hypertreewidth, the result reads as follows:

THEOREM 8. *[13] Let $\mathcal{C}$ be a class of* CQ *of bounded generalized hypertreewidth.*

Assuming W[1] $\neq$ FPT *the following are equivalent:*

1. *The counting problem for queries in $\mathcal{C}$ is solvable in polynomial time*

2. *$\mathcal{C}$ has bounded quantified-star size*

If the schema is not part of the input, or more generally if we assume a bound on the arity of the predicates used in queries of $\mathcal{C}$, they building on [19] we get the following stronger result that also shows that the bounded treewidth hypothesis is necessary:

THEOREM 9. *[13] Let $\mathcal{C}$ be a class of* CQ *using predicates of bounded arity. Assuming* W[1] $\neq$ FPT *the following are equivalent:*

1. *The counting problem for queries in $\mathcal{C}$ is solvable in polynomial time*
2. *$\mathcal{C}$ has bounded treewidth and bounded quantified-star size*

5. DISCUSSION

5.1 More expressive queries

In order to be able to enumerate more expressive queries, one need restrictions on the class of databases under consideration. Several restrictions have been investigated, like bounded degree [20, 12], bounded expansion [21] and low degree [15] for FO queries, bounded tree-width [2, 22] for MSO queries and XPath queries over data trees [7]. The interested reader is referred to [27, 28] for a more detailed overview of these results.

5.2 The impact of order

The definition of CD$\circ$LIN presented here does not specify the order in which the answers are output. One could require a specific order, relevant to the context in which the query is evaluated. For instance, if there is a linear order on the domain of the database, one could require that the tuples of the result are output in lexicographical order. Another typical example is when there is a relevance measure associated to each tuple and one would like the answers to the query to be output in the order of their relevance.

Requiring a specific order when outputting the answers to a query may have a dramatic impact on the existence of constant delay algorithms. This is not surprising as the index built during the preprocessing phase is designed for a particular order.

6. CONCLUSIONS

We mentioned several results concerning constant delay enumeration of conjunctive queries. We hope that we succeeded to convince the reader that this is a very interesting topic. We conclude with several research directions.

One could for instance consider relaxing the "no duplicate" constraint during the enumeration phase and enumerate conjunctive queries with the "bag semantics", i.e. each answer occurs as many times as there are valuations witnessing it. This has not been investigated and is clearly relevant for database queries with aggregate predicates.

The situation of the lower bounds mentioned in this paper is not completely satisfactory as they are based on complexity or algorithmics hypothesis. Of course one can construct artificial problems, based on the fact that there exist quadratic but not linear problems, that do not admit constant delay enumeration algorithms. For the queries mentioned in this note, like the distance 2 one, the lower bounds requires an assumption. It is plausible (i.e. there are no known drastic consequences in complexity theory nor in algorithmic) that the non existence of constant delay enumeration algorithms could be proved with no assumptions. We believe this is an interesting and challenging question.

7. REFERENCES

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[2] G. Bagan. MSO Queries on Tree Decomposable Structures Are Computable with Linear Delay. In *Conf. on Computer Science Logic (CSL)*, pages 167–181, 2006.

[3] G. Bagan. *Algorithmes et complexité des problèmes d'énumération pour l'évaluation de requêtes logiques*. PhD thesis, Université de Caen, 2009.

[4] G. Bagan, A. Durand, E. Filiot, and O. Gauwin. Efficient Enumeration for Conjunctive Queries over X-underbar Structures. In *Conf. on Computer Science Logic (CSL)*, pages 80–94, 2010.

[5] G. Bagan, A. Durand, and E. Grandjean. On Acyclic Conjunctive Queries and Constant Delay Enumeration. In *Conf. on Computer Science Logic (CSL)*, pages 208–222, 2007.

[6] G. Bagan, A. Durand, E. Grandjean, and F. Olive. Computing the jth solution of a first-order query. *RAIRO Theoretical Informatics and Applications*, 42(1):147–164, 2008.

[7] M. Bojańczyk and P. Parys. XPath evaluation in linear time. *J. of the ACM*, 58(4), 2011.

[8] J. Brault-Baron. A Negative Conjunctive Query is Easy if and only if it is Beta-Acyclic. In *Conf. on Computer Science Logic (CSL)*, pages 137–151, 2012.

[9] J. Brault-Baron. *De la pertinence de l'énumération : complexité en logiques*

propositionnelle et du premier ordre. PhD thesis, Université de Caen, 2013.
[10] T. Colcombet. A Combinatorial Theorem for Trees. In *Intl. Coll. on Automata, Languages and Programming (ICALP)*, pages 901–912, 2007.
[11] D. Coppersmith and S. Winograd. Matrix Multiplication via Arithmetic Progressions. *J. on Symbolic Computation*, 9(3):251–280, 1990.
[12] A. Durand and E. Grandjean. First-order queries on structures of bounded degree are computable with constant delay. *ACM Trans. on Computational Logic (ToCL)*, 8(4), 2007.
[13] A. Durand and S. Mengel. Structural tractability of counting of solutions to conjunctive queries. In *Intl. Conf. on Database Theory*, pages 81–92, 2013.
[14] A. Durand and S. Mengel. On Polynomials Defined by Acyclic Conjunctive Queries and Weighted Counting Problems. *J. on Computer and System Sciences (JCSS)*, 80(1):277–296, 2014.
[15] A. Durand, N. Schweikardt, and L. Segoufin. Enumerating first-order queries over databases of low degree. In *Symp. on Principles of Database Systems (PODS)*, 2014.
[16] A. Durand and Y. Strozecki. Enumeration Complexity of Logical Query Problems with Second-order Variables. In *Conf. on Computer Science Logic (CSL)*, pages 189–202, 2011.
[17] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.
[18] E. Grandjean. Sorting, Linear Time and the Satisfiability Problem. *Annals of Mathematics and Artificial Intelligence*, 16:183–236, 1996.
[19] M. Grohe, T. Schwentick, and L. Segoufin. When is the evaluation of conjunctive queries tractable? In *Symposium on Theory of Computing (STOC)*, pages 657–666, 2001.
[20] W. Kazana and L. Segoufin. First-order query evaluation on structures of bounded degree. *Logical Methods in Computer Science (LMCS)*, 7(2), 2011.
[21] W. Kazana and L. Segoufin. Enumeration of first-order queries on classes of structures with bounded expansion. *Symp. on Principles of Database Systems (PODS)*, 2013.
[22] W. Kazana and L. Segoufin. Enumeration of monadic second-order queries on trees. *ACM Transactions on Computational Logic*, 14(4), 2013.
[23] S. Lindell. A Normal Form for First-Order Logic over Doubly-Linked Data Structures. *Int. J. Found. Comput. Sci.*, 19(1):205–217, 2008.
[24] C. H. Papadimitriou and M. Yannakakis. On the Complexity of Database Queries. *J. on Computer and System Sciences (JCSS)*, 58(3):407–427, 1999.
[25] R. Pichler and S. Skritek. Tractable Counting of the Answers to Conjunctive Queries. In *Alberto Mendelzon Intl. Workshop on Foundations of Data Management (AMW)*, 2011.
[26] R. Pichler and S. Skritek. Tractable counting of the answers to conjunctive queries. *J. Comput. Syst. Sci.*, 79(6):984–1001, 2013.
[27] L. Segoufin. Enumerating with constant delay the answers to a query. In *Intl. Conf. on Database Theory*, pages 10–20, 2013.
[28] L. Segoufin. A glimpse on constant delay enumeration. In *Intl. Symp. on Theoretical Aspects of Computer Science (STACS)*, pages 13–27, 2014.
[29] M. Yannakakis. Algorithms for Acyclic Database Schemes. In *Intl. Conf. on Very Large Databases (VLDB)*, pages 82–94, 1981.

Implications of Emerging 3D GPU Architecture on the Scan Primitive

Jason Power
powerjg@cs.wisc.edu

Yinan Li
yinan@cs.wisc.edu

Mark D. Hill
markhill@cs.wisc.edu

Jignesh M. Patel
jignesh@cs.wisc.edu

David A. Wood
david@cs.wisc.edu

Department of Computer Sciences
University of Wisconsin–Madison

ABSTRACT

Analytic database workloads are growing in data size and query complexity. At the same time, computer architects are struggling to continue the meteoric increase in performance enabled by Moore's Law. We explore the impact of two emerging architectural trends which may help continue the Moore's Law performance trend for analytic database workloads, namely 3D die-stacking and tight accelerator-CPU integration, specifically GPUs. GPUs have evolved from fixed-function units, to programmable discrete chips, and now are integrated with CPUs in most manufactured chips. Past efforts to use GPUs for analytic query processing have not had widespread practical impact, but it is time to re-examine and re-optimize database algorithms for massively data-parallel architectures. We argue that high-throughput data-parallel accelerators are likely to play a big role in future systems as they can be easily exploited by database systems and are becoming ubiquitous. Using the simple scan primitive as an example, we create a starting point for this discussion. We project the performance of both CPUs and GPUs in emerging 3D systems and show that the high-throughput data-parallel architecture of GPUs is more efficient in these future systems. We show that if database designers embrace emerging 3D architectures, there is possibly an order of magnitude performance and energy efficiency gain.

1. INTRODUCTION

Due to recent technology trends including the continuation of Moore's law [16] and the breakdown of Dennard scaling [4], computing has become energy-limited. Although device manufacturers are continuing to add more transistors per chip (Moore's law), the threshold voltage of the transistors is not decreasing at the same relative rate (the breakdown in Dennard scaling) [4]. Unlike in the past, database designers cannot rely on computer architects to give increased performance for free; future devices with double performance will consume almost double the energy. Using ITRS projections, Esmaeilzadeh et al. found that by 2024 we can only expect a 7.9× average speedup compared to today's processors from traditional architectural optimizations, more than 24× less than if performance had continued to follow Moore's law [7].

However, there are two architectural trends that can help mitigate this bleak projection: 3D-die stacking and tightly-integrated accelerators. With 3D die-stacking, multiple silicon dies are stacked on top of one another (see Section 4 for more details). This allows tightly-integrating accelerators with traditional CPUs to become more economical and common. Additionally, 3D die-stacking enables very high-bandwidth memory systems, up to 1 TB/s in some projections [1], and decreases the energy for communication by a factor of 3× [3].

Computer architects are already creating tightly-integrated general-purpose commodity hardware accelerators. SIMD hardware (short vector units e.g., SSE and AVX) is one example of this trend. Now, like SIMD units, general-purpose graphics processing units (GPGPUs) are moving onto the CPU die and becoming first-order compute platforms. Today 99% of Intel's and 67% of AMD's desktop CPUs ship with an on-die GPU [18], and server chips with integrated GPUs have been announced [2]. In addition to tightly integrating GPUs and CPUs physically, new programming models, like heterogeneous system architecture (HSA) [19] enable these systems to be easily programmed.

SIMD units, GPGPUs, and other data-parallel architectures can have increased performance and energy efficiency compared to CPU platforms because they have lower overhead per processing element. For instance, in NVIDA's Kepler GPU architecture, 192 processing elements share a fetch,

decode, and scheduling unit, compared to 8 processing elements sharing a front-end in Intel's Haswell architecture. CPUs can waste 90% of the instruction execution energy on the front-end pipeline [11], while GPU architecture amortizes this overhead across tens or hundreds of hardware threads. Additionally, through software-managed caches, large register files, data-caches optimized for throughput, and a high-degree of hardware multithreading, GPUs can effectively use hundreds of GB/s of main memory bandwidth, significantly improving performance and energy for bandwidth-bound applications. Current CPU memory systems are optimized for minimizing instruction latency with small buffers and large general-purpose caches. Thus, for data-parallel applications, CPUs are less efficient than optimized for high-throughput data-parallel hardware.

In this work, we focus on the scan operation. Scans are an important primitive and the workhorse in high-performance in-memory database systems like SAP HANA, Oracle Exalytics, IBM DB2 BLU and Facebook's Scuba. Scans are data-parallel operations, and a series of scan algorithms have been developed in the database community to exploit this parallelism using hardware artifacts such as the parallelism within regular ALU words (e.g. [14]), and SIMD to accelerate scans (e.g. [14, 21, 12, 5, 22]).

However, the parallelism from using CPU SIMD extensions is limited. SIMD hardware sits in an architecture design that is not optimized for high-throughput data processing. The latency-centric CPU memory hierarchy and execution model limit the effectiveness of the SIMD units. Using an architecture explicitly designed to efficiently execute data-parallel applications can significantly increase the efficiency of processing database scans.

Figure 1 shows the energy and performance tradeoff between CPU and GPU architectures. Even though discrete GPUs have more computation resources and higher bandwidth than CPUs, the performance improvement is overwhelmed by overheads due to copying data and operating system interaction. In fact, the discrete GPU results in higher response time (4×) and energy (6×) than a four core CPU; this behavior validates why GPUs have largely been ignored by database systems for scan processing today. However, we find that today's integrated GPUs mitigate many of these problems. But, they only provide a marginal benefit (17%) over the CPU since they are limited by the same memory interface. Thus, the benefit of integrated GPUs is also limited for database scan operations. However, as we project into the future and examine 3D die-stacked systems, GPUs, or other highly

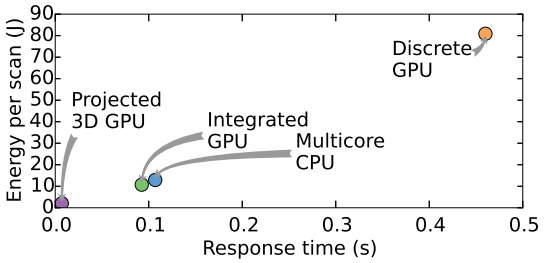


Figure 1: Performance and energy of scans on a multicore CPU, discrete and integrated GPUs, and a projection on a future high-bandwidth, highly data-parallel architecture. Lower and to the left is best.

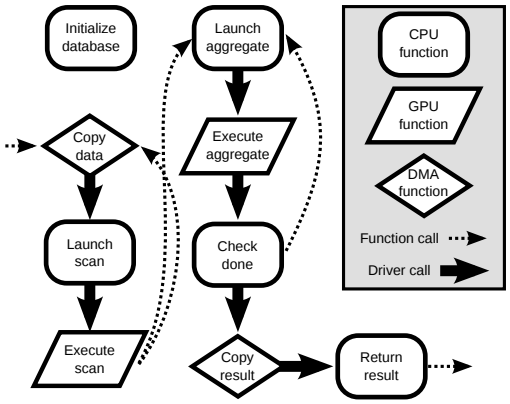
data-parallel architectures, can provide a large benefit over current CPU platforms (15×) and also 3D-stacked CPU systems (4×). Thus, the use of GPU-like high-throughput hardware can not be ignored by database designer in the near future—this is the central message of this paper. We provide initial evidence in support of this position by examining the implications for the database scan operator. (We acknowledge that future work is essential to expand this argument to other data processing operations).

2. GPGPU BACKGROUND

Graphics processing units (GPUs) have recently become more easily programmable, creating the general purpose GPU (GPGPU) computing landscape. This trend began with 1st-generation *discrete GPUs* that are connected to the system via the PCIe bus which has higher latency and lower bandwidth than main memory. Now, 2nd-generation *integrated GPUs* that share the same silicon chip are becoming mainstream. With emerging 3D die-stacking technology, we will soon see 3rd-generation *3D GPUs* that combine the high performance of discrete GPUs with the low latency and simple programming models of integrated GPUs.

There are three key differences between GPUs and CPUs that make GPUs, especially future GPUs, an important platform for database designers to consider. First, GPGPUs employ very wide data-parallel hardware. For instance, an AMD HD7970 can operate on 131,072 bits in parallel, compared to only 256–512 bits in modern CPU's SIMD hardware. Second, GPGPUs are programmed with SIMT (single-instruction multiple-thread) instead of SIMD (single-instruction multiple-data). The SIMT model simplifies programming the GPU's wide data-parallel hardware. For instance, SIMT allows arbitrary control flow between individual data-parallel lanes.

Finally, and importantly, GPU architecture can be more energy-efficient than CPU architecture for



(a) OpenCL flow for the discrete GPU.

(b) HSA flow for the integrated GPU.

Figure 2: Flowchart showing how the CPU and GPU interact in BitWarp.

certain workloads (e.g., database scans). Since many data-parallel lanes share a single front-end (instruction fetch, decode, etc.), this per-instruction energy overhead is amortized. On CPU architectures, the execution front-end and data movement consumes 20–40× more energy than the actual instruction [13]. Additionally, all of the parallelism is explicit for GPUs through the programming model, while CPUs require high energy hardware (like the re-order buffer and parallel instruction issue) to implicitly generate instruction-level parallelism, which wastes energy for data-parallel workloads.

As GPU hardware has become more closely integrated with CPU hardware, the GPU programming models have become more integrated with CPU programming models as well. In this paper, we use the heterogeneous system architecture (HSA) runtime to program the GPU. HSA presents the programmer with a coherent and unified view of memory and low-latency user-level API for using the GPU [19].

Past GPU APIs have significant overheads because they assume the discrete GPU model with high latency and low bandwidth communication between the CPU and GPU. These overheads can significantly affect application performance. We find that the discrete GPU performance is 16× slower than its potential because of these overheads.

Figure 2 shows the difference between the traditional GPU APIs (2a), and new programming models like HSA (2b). This figure shows the steps required to execute a full query on these two systems. The HSA flowchart elides both the operating system driver overheads and the data copies. Because of these changes, not only does the application perform better, but it is simpler to program as well.

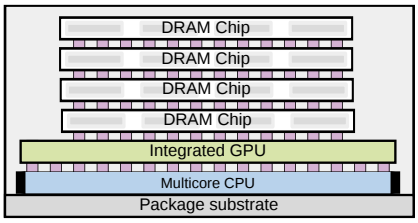


Figure 3: A potential future 3D architecture. All components shown would occupy a single socket.

3. BITWARP IMPLEMENTATION

To study scans on GPUs, we leverage previous work accelerating analytical query processing on the CPU: the BitWeaving scan algorithm [14]. BitWeaving outperformed state-of-the-art scan algorithms by leveraging intra-word parallelism on CPU hardware and is similar to some industry solutions. We have modified the BitWeaving algorithm to execute efficiently on the GPU. The main change in the algorithm is increasing the logical execution width to the size of the GPU's data-parallel units.

BitWeaving uses a coded columnar layout, packing multiple codes per word. This mechanism allows BitWeaving to leverage the 64-bit wide CPU word to execute the scan predicate on many column codes in a single cycle. In BitWarp, we leverage the entire 4096-bit lane width of the GPU. BitWeaving includes two different algorithms, horizontal and vertical, which refer to the way the codes are packed. In this work, we focus only on BitWeaving vertical, as it performs better than horizontal in all cases on the GPU. BitWeaving vertical packs the codes such that one bit of each code is in each consecutive word. The paper by Li and Patel details the algorithm and implementation of BitWeaving [14].

4. 3D ARCHITECTURE

A recent architectural trend is that "Die-stacking is happening" [3] due to advances in fabrication, cooling, and other technologies. There are many wide-ranging implications of die-stacking from device manufacturing to system design. Die-stacking has two important consequences for database designers: higher memory bandwidth and increased compute capability. Some project 1 TB/s of bandwidth, more than 40× the bandwidth available today for CPUs [1, 6, 17]! Die-stacking also moves memory closer to the computation resulting in lower energy: a 3× reduction for first-generation devices [3]. Also, die stacking allows multiple compute chips (e.g. CPU and GPU dies) to be packaged together.

Figure 3 shows a potential architecture which leverages 3D die-stacking. In 3D integration, separate silicon dies are stacked directly on top of one an-

other and connected by through-silicon vias (TSVs). TSVs provide a high-bandwidth, low-latency, and low-energy interconnect. In the architecture in Figure 3, TSVs connect the CPU and GPU, yielding performance similar to an integrated die. Additionally, TSVs connect both the CPU and GPU to die-stacked DRAM. In such a system, we expect performance similar to a discrete GPU, without any overheads, and power similar to today's integrated chip. Figure 3 represents what is in one CPU socket in a motherboard in future systems. 2.5D integration (not shown) is similar to 3D integration, except chips are connected by TSVs in a silicon interposer instead of directly stacked. 2.5D die-stacking has similar characteristics to 3D stacking.

In addition to leveraging the high bandwidth of TSVs, 3D die-stacked architectures also allow CPU and GPU cores to use manufacturing processes customized specifically for CPU or GPU needs. For current CPU-only chips, hardware manufacturers optimize their CMOS manufacturing process to reduce latency, using faster, less energy-efficient transistors and thicker, wider wires that limit bandwidth in exchange for lower latency. For current GPU-only chips, hardware manufacturers optimize their CMOS processes to increase bandwidth (at the expense of latency), using slower, lower-leakage transistors and thinner, narrower wires that maximize intra-chip bandwidth, but also significantly increase latency. For current integrated architectures—where the CPU and GPU share the same silicon die—hardware manufacturers must strike a compromise between the incompatible, conflicting demands of CPUs and GPUs. 3D die-stacking allows the CPU and GPU to be manufactured on separate chips using appropriately optimized manufacturing processes, increasing the performance and energy-efficiency of both CPUs and GPUs.

Figure 3 shows one of many possible architectures which leverage 3D or 2.5D die-stacking to increase memory bandwidth and integrate a highly data-parallel architecture. There are many ways to architect the system to take advantage of these trends of increased bandwidth and compute capability. Determining the best system architecture for 3D-stacked systems is an interesting direction for future work.

5. METHODOLOGY

There are many possible designs for a highly data-parallel system. Some examples include re-architecting CPUs by increasing the vector SIMD width and changing their cache management policies, Intel's Xeon Phi processor which has 72 simple in-order cores with wide SIMD lanes [9], and GPU architecture which is an example of a highly data-parallel architecture that has already shown economic viability. Although there are many possible embodiments, we believe that any highly data-parallel architecture will share many characteristics with GPGPUs. These characteristics include:

- Large number of simultaneous threads to generate the memory-level parallelism needed to hide memory latency,
- Wide vector execution units that can issue vector memory accesses, and
- Many execution units to operate on data at memory-speed.

Additionally, it is likely that programming these devices will be similar to programming current GPGPUs. For instance, OpenCL is a flexible language which can execute on GPUs, CPUs, and other accelerators like the Xeon Phi. We focus on GPGPUs as an example data-parallel architecture that has already shown economic viability.

For a constant comparison point, we use AMD CPU (A10-7850K) and GPU (HD7970) platforms in our evaluation. We use a four core CPU at 3.7 GHz and two different GPUs, an integrated GPU that is on the same die as the CPU, and a discrete GPU connected to the CPU via the PCIe bus. The GPUs have 8 CUs at 720 MHz and 32 CUs at 1125 MHz, respectively. The theoretical memory bandwidth for the CPU-GPU chip is 21 GB/s, and the discrete GPU's memory bandwidth is 264 GB/s. We use a single-socket system in our evaluation, but integrated CPU-GPU chips should scale to multiple sockets similar to CPU-only chips.

We use the discrete GPU as a model to predict the performance of the future die-stacked system. According to the projection in [1, 6, 17], 264 GB/s is a reasonable assumption for the memory bandwidth in a first-generation die-stacked system, and the 32 CU chip in the discrete GPU will fit into a package like Figure 3.

6. RESULTS

To measure the efficacy of the GPU for the scan operation, we measured the performance, power, and energy consumed for the CPU and GPU hardware. Figure 4 shows the time per scan over a 1 billion entry column (about 1 GB of data).

The multicore CPU is not an efficient platform for performing the scan primitive. Figure 4 shows that the speedup of four cores over one core is only 60%. Scan is an embarrassingly parallel operation, so we expect almost perfect scaling from the scan algorithm. The reason the CPU does not scale linearly is that the CPU memory system is not designed to

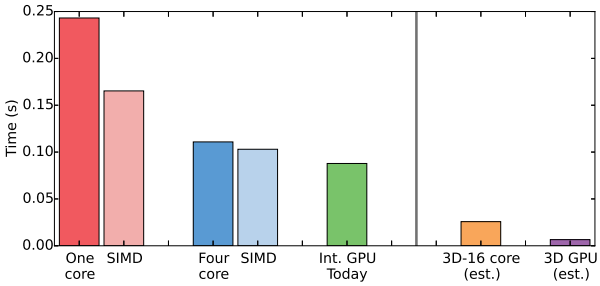


Figure 4: Performance of scan on 1 billion 10-bit codes. Averaged over 1000 scans.

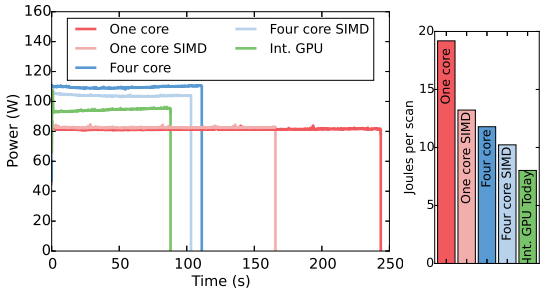


Figure 5: Power for 1000 scans and energy per scan.

support high bandwidth applications. CPU caches are built for low-latency, not high bandwidth.

The integrated GPU sees a small improvement over the multicore with SIMD, a 17% speedup. This improvement is because the integrated GPU's more efficient cache hierarchy. GPUs are designed for these types of high-bandwidth applications and have a separate memory controller which can exploit the memory-level parallelism of the application better than the CPU.

Figure 5 shows the power and energy consumed by the CPU and GPU when performing 1000 scans. The data was obtained by measuring the full system power; thus, it includes all of the system components (e.g., disk, motherboard, DRAM, etc.). The right side of the figure shows the total energy consumed (power integrated over time).

Figure 5 shows that even though the four core configuration and the integrated GPU take more power than the one core CPU, they execute much faster, resulting in lower overall energy. Additionally, the integrated GPU is more efficient than the four core CPU in power and performance.

In the future, it's likely that the GPU will become even more energy efficient compared to the CPU. Each GPU compute unit (CU) (similar to a CPU core) has lower power per performance than a CPU core [13]. Also, each GPU CU is smaller area per performance than CPU cores. Thus, there can be many more GPU CUs than CPU cores, exemplified by our test platform.

6.1 3D Architecture Performance

There are two characteristics of 3D architecture that significantly affects the performance of the scan primitive. First, by stacking a GPU die with the CPU die, each processor type is more highly optimized and can take more total area, thus increasing the overall compute capability of the system. Second, since the chip-memory interconnect uses TSVs, the memory bandwidth is orders of magnitude higher than current systems. These attributes together create a higher performance and more energy efficient system than today's architectures.

The right side of Figure 4 shows the estimated performance of the CPU and the GPU in a 3D die-stacked system. To estimate the CPU's speedup, we assume that adding 4× more cores will result in a 4× speedup. This is a very aggressive estimate; the CPU is unlikely to get perfect speedup when adding more cores. There is only a 60% speedup from one to four cores; however, the added bandwidth will increase the CPU performance too. We err on the side of overestimating the CPU performance.

To estimate the 3D GPU's performance, we run the scan operation on a current discrete GPU with 32 CUs (4× more execution resources than the integrated GPU) with a bandwidth similar to future die-stacked systems [6]. We discard all of the overheads associated with using the discrete GPU, since on a die-stacked system, the overheads will be minimal, as we found with the integrated GPU.

Using this projection, we find that a die-stacked GPU system can provide 15.7× higher performance than today's non-stacked multicore CPU system. The 3D GPU is also 3.9× faster than the aggressive 3D CPU estimate.

In addition to the performance gains, 3D die-stacked DRAM enables lower energy per access by using smaller and shorter wires than off-chip DRAM. Coupled with the GPU's efficiency, it is likely that the energy benefit of this system is much higher than the performance benefit.

We predict that to take advantage of the increasing memory bandwidth from 3D die-stacking, database designers must embrace high-bandwidth, highly data-parallel architectures, and we have shown that the GPU is a good candidate. Looking forward, CPU architecture will continue to become more efficient, including for data-parallel workloads. However, GPUs will also increase in efficiency at a similar rate. We believe that as these trends come to fruition, it will become increasingly important for database designers to leverage high-bandwidth data-parallel architectures to keep pace with the highest possible performance.

7. CONCLUSIONS AND FUTURE WORK

Previous works have shown the huge potential of using GPUs for database operations (e.g., [8, 10, 15, 20]). However, many of these works have not included the large discrete GPU overheads when operating on large in-memory databases. We show current physically and logically integrated GPUs mitigate the problems with discrete GPUs and show a modest speedup and energy reduction over multicore CPUs for scan operations.

Looking forward, computer architects are pursuing many interesting avenues to increase the memory bandwidth significantly, such as 3D die-stacking. However, conventional multicore CPU architecture is not well suited to efficiently use this increased memory bandwidth. We advocate that to take advantage of these architectural trends, database designers should look to data-parallel accelerators, of which the GPU is one example. If database designers embrace these new architectures, there is possibly an order of magnitude performance and energy efficiency gain for scans. Examining these issues for a wider range of database workloads is an interesting direction for future work.

8. ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation (CCF-1218323, CNS-1302260, CCF-1438992, IIS-0963993, IIS-1110948, and IIS-1250886), the Anthony Klug NCR Fellowship, Cisco Systems Distinguished Graduate Fellowship, Google, and the University of Wisconsin (Kellett award and Named professorship to Hill). Hill and Wood have a significant financial interest in AMD and Google. Patel has a significant financial interest in Microsoft and Quickstep Technologies.

9. REFERENCES

[1] 3D-ICs. `http://www.jedec.org/category/technology-focus-area/3d-ics-0`, 2012. Accessed: 2014-6-24.

[2] AMD. AMD Opteron X2100 Series APU. `http://www.amd.com/en-us/products/server/x2100`. Accessed: 2014-5-28.

[3] B. Black. MICRO 46 Keynote: Die Stacking is Happening, 2013.

[4] M. Bohr. A 30 Year Retrospective on Dennard's MOSFET Scaling Paper. *IEEE Solid-State Circuits Newsletter*, 12(1):11–13, Jan. 2007.

[5] Z. Chen, J. Gehrke, and F. Korn. Query optimization in compressed database systems. In *SIGMOD Conference*, pages 271–282, 2001.

[6] H. Consortium. Hybrid Memory Cube Specification 1.0, 2013.

[7] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. In *ISCA Conference*, pages 365–376, 2011.

[8] N. K. Govindaraju, B. Lloyd, W. Wang, M. Lin, and D. Manocha. Fast computation of database operations using graphics processors. In *SIGMOD Conference*, page 215, 2004.

[9] R. Hazra. Accelerating insights in the technical computing transformation. In *Intl. Supercomputing Conf.*, 2014.

[10] B. He, K. Yang, R. Fang, M. Lu, N. Govindaraju, Q. Luo, and P. Sander. Relational joins on graphics processors. In *SIGMOD Conference*, page 511, 2008.

[11] M. Horowitz. Computing's Energy Problem (and what we can do about it). In *IEEE International Solid-State Circuits Conference*, pages 10–14, 2014.

[12] R. Johnson, V. Raman, R. Sidle, and G. Swart. Row-wise parallel predicate evaluation. *PVLDB*, 1(1):622–634, 2008.

[13] S. W. Keckler. Life after Dennard and How I Learned to Love the Picojoule. In *MICRO 44 Keynote*, 2011.

[14] Y. Li and J. M. Patel. BitWeaving: fast scans for main memory data processing. In *SIGMOD Conference*, pages 289–300, 2013.

[15] M. D. Lieberman, J. Sankaranarayanan, and H. Samet. A Fast Similarity Join Algorithm Using Graphics Processing Units. In *ICDE Conference*, pages 1111–1120, Apr. 2008.

[16] G. Moore. Cramming More Components Onto Integrated Circuits. *Electronics*, pages 114–117, Jan. 1965.

[17] J. T. Pawlowski. Hybrid Memory Cube (HMC). In *Hot Chips 23*, 2011.

[18] J. Peddie. Market Watch. Technical Report Q1'04, Jon Peddie Research, 2014.

[19] P. Rogers. Heterogeneous System Architecture Overview. In *Hot Chips 25*, 2013.

[20] N. Satish, C. Kim, J. Chhugani, A. D. Nguyen, V. W. Lee, D. Kim, and P. Dubey. Fast sort on cpus and gpus: a case for bandwidth oblivious SIMD sort. In *SIGMOD Conference*, pages 351–362, 2010.

[21] T. Willhalm, N. Popovici, Y. Boshmaf, H. Plattner, A. Zeier, and J. Schaffner. SIMD-Scan: Ultra fast in-memory table scan using on-chip vector processing units. *PVLDB*, 2(1):385–394, 2009.

[22] J. Zhou and K. A. Ross. Implementing database operations using SIMD instructions. In *SIGMOD Conference*, pages 145–156, 2002.

Automatic Filling of Hidden Web Forms: A Survey

Gustavo Zanini Kantorski Viviane Pereira Moreira Carlos Alberto Heuser
{gzkantorski,viviane,heuser}@inf.ufrgs.br
Institute of Informatics – UFRGS – Porto Alegre – Brazil

ABSTRACT

A significant part of the information available on the Web is stored in online databases which compose what is known as *Hidden Web* or *Deep Web*. In order to access information from the Hidden Web, one must fill an HTML form that is submitted as a query to the underlying database. In recent years, many works have focused on how to automate the process of form filling by creating methods for choosing values to fill the fields in the forms. This is a challenging task since forms may contain fields for which there are no predefined values to choose from. This article presents a survey of methods for Web Form Filling, analyzing the existing solutions with respect to the type of forms that they handle and the filling strategy adopted. We provide a comparative analysis of 15 key works in this area and discuss directions for future research.

1. INTRODUCTION

The Hidden Web [11] or Deep Web [4] is the part of the Web that is not accessible through traditional crawling (*i.e.,* direct link navigation) [19]. The contents of the Hidden Web can only be accessed by filling out Web forms, such as the ones in Figure 1, which are then submitted as queries to the online database behind the form. The Hidden Web covers several topic domains, such as government, education, entertainment, business, health, news, and sports. There are thousands of online databases for each of those domains – most of them containing structured information [7]. Exposing the contents of an online database can be achieved by designing wrappers, *i.e.,* programs that extract data from a specific Web site. However, since wrappers are specific for each site, this solution is not feasible when dealing with thousands of hidden Web sites. Thus, a more scalable approach is to use a Hidden Web Crawler to automatically identify and retrieve data from online databases.

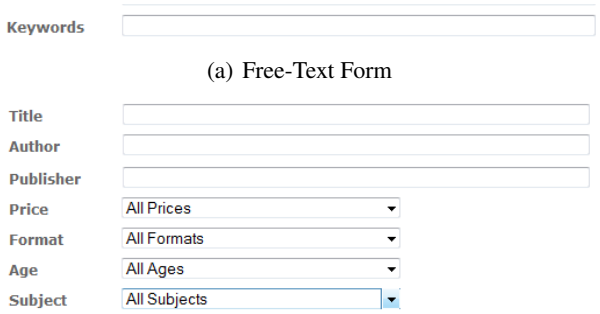
Hidden Web Crawling has many applications. The discovered content can be indexed by generic search engines, such as Google, Bing, and Yahoo!. Furthermore, it can be used to create vertical search engines, which focus on a specific segment such as real estate, online auction, books, airline tickets, etc.

Hidden Web crawling can be divided into three key phases: (i) discovery of the entry points to the Hidden Web, *i.e.,* Web forms that allow searching on-line databases [3, 21–23]; (ii) identification, filling, and submission of forms [1,2,9,14,15,20,22,25–27,29–31]; and (iii) data extraction from the results of submissions [5, 6, 8, 17, 32, 33].

This survey is focused on the second phase. The identification of the fields is reasonably straightforward and can be achieved by parsing the HTML code of the page containing the form looking for specific tags such as `input` and `select`. The challenge is *Web Form Filling* (WFF), *i.e.,* how to automatically fill the fields using suitable values in order to retrieve meaningful data. This is a critical step since filling a form with unsuitable values will result in blank or error pages representing a waste of resources. The goal is not to find all possible values, but to select a subset of values so as to minimize the number of submissions and maximize the coverage, i.e., retrieve more distinct data behind the form. The fact that forms are designed to be handled by human users and this leads to a diversity of designs poses further difficulties to automatic processing.

Figure 2 shows a fragment of the HTML code for the Web form shown in Figure 1(b). An HTML form is defined within a <*form*> and a <*/form*> tag (see Figure 2, lines 1 and 23). Form fields can be text boxes, selection lists, checkboxes, radio buttons, or submit buttons. Selection lists, radio buttons, and checkboxes show a list of options to the user. On the other hand, a text box field does not contain options, so the values need to be discovered somehow. More concisely, fields can be grouped into two types: *fields with a finite domain* such as selection lists (see Figure 2, lines 8 to 21); and *fields with an infinite domain*, such as text fields, in which a user can type any value (see Figure 2, lines 2 to 7), which brings further challenges to automatic filling.

Authors classify forms under different names with respect to the kinds of fields that they have. Forms with a



(a) Free-Text Form

(b) Complex Form

Figure 1: (a) Free-Text and (b) Complex Web Forms

single text field are called *simple* [24,25] or *free-text* [28]. Forms with several fields are called *multi-field* [25], *advanced* [24], or complex [28]. In this survey, we adopt the terminology by Tjin-Kam-Jet [28] which refers to the former as *free-text* and as *complex* to the latter.

Although the Hidden Web also has unstructured content (*e.g.*, text, images, videos, etc.), structured contents are the most prevalent (over 75%, according to Chang *et al.* [7]). These are usually relational databases containing attribute-value pairs (*e.g.*, a movie Web site that returns movie information such as director, actors, title, etc.). The vast majority of the existing work on WFF is focused on uncovering structured data, so this represents the primary focus of this survey.

Contributions. This article contains a survey of the key works on WFF. We analyzed the existing literature on Hidden Web Crawling and report on 15 works we believe to be the most influential in WFF. These works are discussed with respect to some key features such as text field seed generation, value generation, prior knowledge, human intervention, and submission method. While distinct works employ different filling strategies, two aspects are common to all: (i) filling method and (ii) form type. In this article, we consider two categories for filling method: *heuristics* and *machine learning*; and two categories for form type: *free text Web forms* and *complex Web forms*. We present a comparative study under two perspectives: a *holistic view*, in which each method is treated in its fullness; and a *Cartesian view*, in which each method is studied as a collection of parts. To the best of our knowledge, this is the first survey to address WFF in the Hidden Web.

2. DEFINITIONS AND PROBLEM OVERVIEW

Web Form Filling (WFF) is the process of selecting values for filling the fields in a Web form. Generating values for fields with finite domains is fairly easy as the possible values are found in the form itself. For fields with infinite domains, the values have to be predicted. This process may be divided into two sub-problems: (i) selecting an appropriate set of initial values (*seeds*); and (ii) selecting an appropriate set of input values.

```
1<form method="get" action="/results.html">
2     <span>Title</span>
3     <input type="text" name="title" />
4     <span>Author </span>
5     <input type="text" name="authorName" />
6     <span>Publisher</span>
7     <input type="text" name="publisher" />
8     <label for="price">Price</label>
9       <select name="prc" id="prc">
10        <option value="all">All Prices</option>
11        <option value="1">under $10</option>
12        <option value="2">$10-$25</option>
13        ..........
14      </select>
15    <label for="select">Format</label>
16      <select name="fmt" id="fmt">
17        <option value="all">All Formats</option>
18        <option value="0">Hardcover</option>
19        <option value="1">Paperback</option>
20        ..........
21      </select>
22      ................
23</form>
```

Figure 2: Fragment of HTML Code from Figure 1(b)

Existing solutions for WFF can be classified with respect to their reliance on prior knowledge. Approaches which rely on prior knowledge need to build the knowledge base beforehand and generate values according to the domain (*i.e.,* topic) of the form. On the other hand, methods that do not rely on prior knowledge typically generate new candidate values by analyzing the results from previous submissions.

The problem in which this survey is focused may be formulated as "given an HTML form, identify the fields and find suitable values to fill these fields so that the form retrieves meaningful data". A form F is represented by two sets. The first set describes the fields and their values. The second set represents the results of form submissions as a single table. More formally, the fields and values of a form F are represented as a set of *(field, domain)* pairs:
$F = \{(f_1, dom(f_1)), (f_2, dom(f_2)), ..., (f_n, dom(f_n))\}$
where the f_i's are the form fields and the $dom(f_i)$ are the domains.

A form field represents input objects, such as selection lists, checkboxes, text boxes. The domain of a field is the set of values it can take. For example, if f_j is a selection list ($\langle select \rangle$ tag in HTML), then $dom(f_j)$ are the values in the list $\langle option \rangle$ tag in the HTML.

A form field is usually associated with a *label* which has a descriptive text that helps understanding the field.

Thus, $label(f_i)$ refers to the label associated with the i^{th} form field. For example, the label *Title* is associated with the first field in the form shown in Figure 1(b).

The second set in form F is represented as a single table R with m records $r_1, r_2, ..., r_m$ over a set of k attributes $A = a_1, a_2, ..., a_k$, in which each attribute has $dom(f_k)$. Then, we have:
$F = \{(f_1, dom(f_1)), (f_2, dom(f_2)), .., (f_n, dom(f_n))\}$
$R = \{r_1, r_2, ..., r_m\}$ and $A = \{a_1, a_2, ..., a_k\}$

An HTML form can be submitted by two methods, *get* or *post*. In the *get* method, field values are included as part of the URL in the HTTP request. For example, suppose a user entered the value 'Hidden Web' in the field 'Title' in the form illustrated in Figure 1(b). The value entered for the title field is appended to URL generating '*../results.html?title=Hidden+Web*'. In the *post* method, field values are sent inside the HTTP request and the URL contains only the action that is about to be performed.

3. APPROACHES FOR WEB FORM FILLING

We surveyed 15 works that address the problem of filling fields in Web forms. This survey organizes the solutions found in the literature according to two aspects: *filling method* and *type of interface*. There are two possible filling methods: *heuristic-based* and *machine learning-based*. Heuristic-based approaches are usually guided by statistical information (such as term frequencies, number of rows retrieved, etc.) and apply thresholds as stopping criteria. Machine learning approaches use this statistical information to create a model which decides which values to use to fill the forms. Similarly, there are two types of interface: *free-text* Web form where users type a list of keywords in a single search text box and *complex* Web forms which contain several fields. We analyze each existing approach according to its filling method and the type of interface that it supports. It is important to notice that this analysis does not separate the surveyed approaches into disjoint groups, as it is possible for an approach to use both filling methods and/or types of interfaces.

Existing works are grouped into four categories: (i) heuristics applied to free text forms; (ii) heuristics applied to complex Web forms; (iii) machine learning applied to complex Web forms; and (iv) overlapping combinations. We classify under *overlapping combinations* approaches which tackle more than one type of interface and/or filling methods. We found no works which apply machine learning applied to free text forms. Thus, there is no subsection here dealing with this category. Also, we found only one approach that works with complex Web forms and machine learning. In the next subsections, we analyze each of these categories.

3.1 Heuristics Applied to Free Text Forms

Since free text forms have only one keyword text field, the formalism presented in Section 2 can be reduced. As a result, a Web form can be simply represented as $F = \{(f_1, dom(f_1)\}$, where f_1 is a keyword field and $dom(f_1)$ is the set of values for form field f_1. In general, heuristic-based methods rely on statistical information about the submissions.

Barbosa and Freire [2] select a set of keywords with high frequency to build queries with high coverage for form field f_1. The discovery of values for the domain $dom(f_1)$ is based on the data coming from the database itself instead of a random word generation. The approach is composed of two steps. The first step is the selection of initial keywords from the page that contains the form. The selected keywords are used to fill the form.

The algorithm proceeds to find additional keywords by iteratively submitting keywords obtained in the results of previous submissions. The goal is to select high-frequency keywords and use them to construct a query that has high coverage The stopping condition is determined by two parameters: *maxterms* or *maxSubmissions* probe queries submitted. The best choice for these parameters depends on the database. The rationale is that values found in the database are more likely to result in higher coverage than randomly selected values.

The main advantage of this method is that Web forms with keyword fields do not need detailed knowledge of the data structure or schema. Experiments on different domains and form sizes show that using *stopwords* increases coverage. This happens because *stopwords* have high frequencies (*i.e.,* appear in many documents).

Soulemane *et al.* [27] present a method which selects values for form field f_1 based on term frequency. The rationale is that frequency determines the importance of a value in a set of documents (or database rows).

The focus of Soulemane's work is on providing an automatic indexing mechanism for dynamic Web contents. The method comprises form detection, selection of search keywords, dynamic content extraction, and detection of duplicate URLs. Form detection consists in identifying Web forms with a single general input text field. Selection of search keywords tries to generate an optimized search result. Dynamic content extraction is the extraction of the data from the result pages. Detection of duplicate URLs deals with cases in which two distinct values may generate the same URL twice.

As in [2], the initial keyword values are selected from the page containing the form. After obtaining the first results, values for the domain $dom(f_1)$ are chosen from the successfully retrieved pages. A threshold *max* submissions per form prevents the crawler from falling into

an infinite loop. But if the method fails to obtain a convenient keyword from a given page, a value is chosen from the repository containing results from previous submissions. As a last resort, an external dictionary can be queried to provide values.

Term frequency measures how often a value is found in a collection. Suppose a value V_i occurs n_p times within a Web page P which contains N_p values. Term frequency is given by $F_{tf} = \frac{n_p}{N_p}$.

Ntoulas *et al.* [25] describe an adaptive algorithm based on results from previous submissions which adapts its query selection policy automatically based on such results. They assume that the crawler downloads pages from a Web site that has a set of pages S. Each potential query q_i which may be issued can be treated as a subset of S. Each subset is associated with a weight that represents the cost of issuing the query. Thus, the goal is to find which subsets cover the maximum number of Web pages with the minimum total weight (cost).

The heuristics employed by this method include the cost and the amount of new data returned for a query that has not been retrieved by previous queries. The maximum total weight takes a number of factors: the cost of submitting the query to the form, the cost of retrieving the result index page, and the cost of downloading the actual pages. The authors assume that submitting a query incurs a fixed cost of c_q. The cost c_d to download a matching item is also fixed, while the cost of downloading the result index page is proportional to the number of retrieved results. Then the overall cost of query q_i is as described in Eq. 1:

$$cost(q_i) = c_q + c_r P(q_i) + c_d P_{new}(q_i) \quad (1)$$

where $P_{new}(q_i)$ is the fraction of the new documents from q_i that have not been retrieved from previous queries.

Based on the cost and the amount of data retrieved, the authors use the efficiency metric to quantify the desirability of the query q_i (Eq. 2):

$$Efficiency(q_i) = P_{new}(q_i)/cost(q_i) \quad (2)$$

where $P_{new}(q_i)$ is the number of new documents returned for q_i and $cost(q_i)$ is the cost of issuing the query q_i.

Efficiency measures how many new documents are retrieved per unit cost and it can be used as an indicator of how well the resources are spent when submitting q_i. Thus, the crawler can estimate the efficiency of every candidate q_i and choose the one with the highest value. For estimating efficiency, the method has a *query statistics table*. This table stores the counts of how many times a value q_i appears within the documents downloaded from $q_1, ..., q_{i-1}$. The set of q_i's determines the domain $dom(f_1)$.

One issue here is the choice of the keyword to be used as the first query. The selection is not done by the adaptive algorithm as it has to be manually set because the query statistics table has not been populated yet. Thus, the selection is generally arbitrary. So, for the purpose of fully automating the whole process, the authors describe that some additional investigation is necessary.

Some Web Hidden Web sites limit the number of results returned for a query. Thus, if a query has a large number of matching results, only a fraction will be returned (e.g., the first 1000). This is problematic since the probability that a query q_i appears in the pages from $q_1, q_2, ..., q_{i-1}$ considers the entire database. To solve this issue, the approach assumes that the results returned are a random sample of the complete set of results which match the query and adjust the estimates to calculate $P(q_{i+1} | \, q_1 \vee ... \vee q_i)$.

Wu *et al.* [31] present a form filling method based on feedback of the previously submitted values. The form is treated as a single table, referred to as *DB*, with a set of queriable attributes $AS = \{attr_{s1}, attr_{s2}, ..., attr_{sm}\}$ and a set of result attributes $AR = \{attr_{r1}, attr_{r2}, ..., attr_{rn}\}$. Table *AS* is equivalent to set F and table *AR* is equivalent to sets R and A.

The set of distinct attribute values (DAV) consists of all distinct attribute values in *DB*. An attribute-value graph (AVG), *G(V,E)* for *DB* is a non-directional graph that is built as follows: for each distinct value a_{vi} in *DAV* there is only one vertex $v_i \in V$. A non-directional edge $(v_i, v_j) \in E$, if, and only if, av_i and av_j coexist in a relational instance $t_k \in DB$. Each edge in AVG represents a relational link between av_i and av_j. These values compose the domain $dom(f_1)$. The determination of values that are used for form filling is reached by the use of a seed value and, from the results, values related to the one used in the query are extracted. The authors rely on a heuristic cost to evaluate the query. The cost of a query q_i in the *DB* database is defined as in Eq. 3:

$$cost(q_i, DB) = \frac{num(q_i, DB)}{k} \quad (3)$$

where *num(q_i,DB)* represents the total number of records in *DB* matching q_i, and k corresponds to the maximum number of records in each result page.

For selecting the next query, the authors define a new metric called *query harvest rate* to capture the productivity of each candidate query. Given a target Web database *DB*, and a local database DB_{local} containing the data records already crawled from *DB*, the harvest rate of q_i is defined as in Eq. 4:

$$HR(q_i) = \frac{[num(q_i, DB) - num(q_i, DB_{local})]}{cost(q_i, DB)} \quad (4)$$

where $num(q_i, DB)$ and $num(q_i, DB_{local})$ is the number data records matched by q_i in *DB* and DB_{local}, respectively; $cost(q_i)$ stands for the cost of obtaining all the

result pages. The goal is to select the attribute value with the highest harvest rate as the next query.

Furthermore, the authors also integrate domain knowledge to query selection. The method uses a domain statistics table *DT* of a domain *DM*. Table *DT* consists of a collection of entries in the form of $< q_i, P(q_i, DM) >$, where q_i stands for a candidate query and $P(q_i, DM)$ is the domain probability that q_i occurs in *DM*. With table *DT*, two groups of queries appear: Q_{DB} and Q_{DT}. Q_{DB} consists of queries whose corresponding attribute values have been discovered in the target database *DB* from the previous results, and Q_{DT} corresponds to the queries in the domain table *DT*, but not yet seen by *DB*.

Wang *et al.* [30] gather a set of documents as a sample that represents the original database. From the sample, they choose a set of values representing the domain $dom(f_1)$ that cover most of the items in the sample with a low cost (*i.e.,* retrieve the most results with the fewest submissions). These values are used to extract data from the original database.

The solution considers two heuristics: the *hit rate*, which denotes the set of data retrieved from the database, and the *overlapping rate*, which refers to the amount of duplicated data retrieved. More formally, the hit rate of a set of queries Q in a database, denoted by $HR(Q, DB)$, is defined as the ratio between the number of unique data items collected by sending the queries in Q to *DB* and the size of the database *DB*. The overlapping rate of Q in *DB*, denoted by $OR(Q, DB)$, is defined as the ratio between the total number of collected links and the number of unique links retrieved by sending queries in Q to *DB*.

The algorithm runs with the database *DB* as input, the sample size s, and the query pool size p. The sample should have an appropriate size to produce a satisfactory query list in the query pool. In order to select the queries to issue, the method creates a query pool using the terms found in a random sample from the database.

The values selected from the query pool are those that have document frequency (*df*) ranging between 2% and 20% of the sample size. Values that occur in fewer than 2% of the sample are most probably rare values, while values that appear in more than 20% of the documents are too common to consider. Then, the relative query pool size of a set of queries Q on database *DB*, denoted by $poolSize(Q, DB)$, is defined as in Eq. 5:

$$poolSize(Q, DB) = \sum_{q \in Q} df(q, DB)/|DB| \qquad (5)$$

where $df(q, DB)$ is the document frequency of q in *DB*, *i.e.,* the number of items in *DB* matching query q.

Once the query pool is populated, values from this pool are selected and sent to *TotalDB*. The selection criteria are to have Hit Rate equal to 1 and the minimum Overlapping Rate available in the sample. These values are the domain $dom(f_1)$ of form field f_1.

In practice, the total number of documents in a real deep Web database is unknown; hence the calculation of *HR* becomes impossible.

3.2 Heuristics applied to Complex Web Forms

Lage *et al.* [18] present a method to fill fields using a set of heuristics and a sample data repository for automatically finding forms, filling them out, and collecting pages containing useful data. The method starts crawling from the main page of the site looking for forms in a blind search (*i.e.,* the search is not guided by any heuristics). A set of heuristics is used to discard non-query forms. Next, it extracts the labels from the remaining forms and, using a sample data repository, it tries to learn how to fill them out. Finally, it submits all filled forms in order to identify data-rich pages. The process ends when these pages are not found.

A key component is the *sample data repository*. It is used to identify evidences in the traversed pages that such pages belong to a specific application domain. The repository is a set of attribute-value pairs of the form $< label(f_i), dom(f_i) >$ that describe objects from the application domain. The repositories are generated by extracting data from Web sources of specific domains.

The task of form filling consists in finding a mapping between form fields and repository attributes. Heuristics extract the labels which are above input fields. If the labels do not match the attributes in the repository, the form is disregarded.

Mapping is straightforward for search forms which contain only fields with finite domains, since one can easily obtain the matching attributes, which are explicitly available in the *select* HTML tags. If matches are found, the agent knows how to fill the form and all necessary information to submit it is already available, so the process continues.

Raghavan *et al.* [26] propose a task-specific Hidden Web crawler called *Hidden Web Exposer* (HiWE). The approach was developed to automatically process, analyze and submit forms through an internal model of forms and form submissions.

The model treats a form F as a set of $(element, domain)$ pairs: $F = (E_1, D_1), (E_2, D_2), ..., (E_n, D_n)$ where the E_i is the element and the D_i is the domain. The elements E_n are the form fields f_i and domains D_n are the field domains $dom(f_i)$.

The values used to fill out forms are maintained in a special table called *Label Value Set* (LVS). LVS tables are associated to form fields. Each row in LVS table is a pair (L, V), where L is a label and $V = \{v_1, v_2, ..., v_n\}$ is

a value set assigned to label L. The V set has a M_v function that associates weights, between 0 and 1, to each set member. Each v_i is a possible value that is assigned to form field E if $label(E)$ matches with L. The estimated value $MV(v_i)$ represents the correctness of value v_i in relation to element E.

The main issue in this method is filling LVS table with the desired values for queries and, after this, the association of values to form fields. The LVS table also allows *label aliasing*, *i.e.,* two or more labels may share the same value set V.

The HiWE crawler supports four strategies for populating the LVS table: *(i) explicit initialization*, in which it can be supplied with labels and associated value sets at startup time, *(ii) built-in categories*, which it has built-in entries in the LVS table for some common categories, such as times, months, dates, days of week, etc., *(iii) wrapped data source*, in which entries for the LVS table are used for querying data sources through a well-defined interface, and *(iv) crawling experience*, which provides useful information which can be used when crawling new sites.

Liddle *et al.* [20] perform automatic form filling by assigning a default value to form fields. The authors have created a prototype tool that automatically retrieves the data behind a specific HTML form.

The strategy involves three steps: *(i)* issue the default query, *(ii)* retrieve a sample to determine whether the default query produces acceptable results, and *(iii)* analyze the retrieved information and submit new queries exhaustively until a limiting threshold is reached.

The authors heuristically select a reasonable minimum number of submissions to maximize the coverage. In order to do that, the size of the database behind the form is estimated, and then queries are issued until a certain percentage of completeness is reached.

Heuristics include the percentage of data retrieved, the number of queries issued, the number of bytes retrieved, the amount of time spent, and the number of consecutive empty queries. Each of these thresholds constitutes a sequential stopping criterion that can terminate the crawl before trying all possible queries (*i.e.,* all combinations of values for fields with finite domains).

The sampling batch needs to be large enough to cover the margins of the sample space. Let $f_1, f_2, ...f_n$ be the n be the fields with finite domains, and let $|f_i|$ represent the number of values for the i^{th} factor. $|f_i|$ stands for the field domain $dom(f_i)$. Then, the total number of possible combinations N for this form is $\prod |f_i|$, and the cardinality c of the largest field is $max(|f_1|, |f_2|, ..., |f_n|)$. Next, C is defined as the size of a sampling batch. C is calculated as $max(c, log_2 N)$.

This accounts for the cases in which there are many fields of small cardinality. If the C sample queries yield new data, the method proceeds by sampling additional batches of C queries at a time, until it reaches one of the user-specified thresholds or it exhausts all the possible combinations.

This method does not handle text fields, ignoring them whenever possible. If they are mandatory, and thus cannot be ignored, user's intervention is requested.

Alvarez *et al.* [1] propose an architecture called *DeepBot* for crawling the Hidden Web. The crawler works in three steps: (i) for every domain, the system tries to match its attributes with the fields of the form, using visual distance and text similarity metrics, (ii) by using the output of the previous step, the system determines whether the form is relevant with respect to the domain and, (iii) if the form is relevant, the crawler uses it to run the queries defined in the domain.

For each query, the system obtains a new URL to add to the list of URLs. The authors describe the domain definitions used to guide the data collection task. A domain definition is composed of a set of attributes $A = a_1, a_2, ..., a_n$, a set of queries $Q = q_1, q_2, ..., q_m$, and a relevance threshold μ.

The method uses an attribute set that represents form fields and a query set associated with the domain. A set of attributes has a name, a nickname list, and a specificity index s_i. The nickname list represents alternative labels that may identify the attribute in a query form. For instance, the attribute *author*, from a domain used for collecting data about books, could have nicknames such as *writer* or *writtenby*. The specificity index s_i is a number between 0 and 1 indicating how likely a query form containing such an attribute is actually relevant to the domain. For instance, in the book domain, the attribute ISBN would have a very high s_i, since the presence of this attribute in a form is a strong evidence that it deals with book search. On the other hand, *price* would have a low s_i value, since it could be related to any type of product.

The set of queries is a list of pairs *(attribute, value)* where *attribute* is an attribute a_i from the set of attributes A and *value* is a string. The query set is run on the discovered relevant forms. The labels of the fields are extracted and compared to attributes of the domain through textual similarity. The domain attribute values that match the fields are selected for filling the form. For that, heuristics based on visual distance measures between the form fields and the texts surrounding them are used.

Finally, the domain also includes a relevance threshold μ. The specificity indexes and threshold will be used to determine whether a given form is relevant to a domain. The authors do not present details of how a query list for each attribute is built.

3.3 Machine Learning applied to Complex Web Forms

The methods that employ machine learning techniques generally rely on manually labeled data to serve as training instances. As a consequence, user intervention is needed to validate the data and to make corrections on the labeling.

Toda *et al.* [29] describe a method, called *iForm*, for WFF based on value extraction from free text documents. The method is divided into two sub-problems: extracting values from the input text; and filling the form field using the extracted values. It automatically chooses segments from the input text and assigns them to the appropriate form fields. Free text documents are treated as sequences of tokens $t_1, t_2, ..., t_N$, representing individual words or punctuation. The extraction task consists in identifying segments from the documents, *i.e.,* a sequence of contiguous tokens, which are suitable for the fields in the form.

The method exploits features related to the content and the style of the values. These are combined in a Bayesian framework. The conditional probabilities (probability using content related features and probability using style related features) of a field associated with an extracted value of the text document are computed. The final conditional probability can be computed using a disjunctive operator *or* over the probabilities derived from each feature.

The probabilities are assigned to form fields f_i and the extracted values are the domains $dom(f_i)$. The approach relies on the knowledge obtained from the values of previous submissions for each field and on manual textual input (to correct errors). The authors do not report on experiments run on search forms.

3.4 Overlapping Combinations

Jian *et al.* [13, 14] present a method for Hidden Web crawling. The method combines heuristics and machine learning techniques for filling free-text Web forms. Therefore, we can reduce the model presented in Section 2 to a single form field f_1. The domain $dom(f_1)$ is the set of values with the highest harvest rates.

In this approach, the harvest rate for each query is encoded as a tuple representing its linguistic, statistic, and HTML features. The linguistic features are *part of speech* which represents the category of the word (noun, verb, adjective, etc.);the *length*, which represents the length of values in number of characters; and the language that the values fall into (useful for multilingual crawls). Statistical features include term frequency (TF), document frequency (DF), Term Frequency times Inverse Document Frequency (TF$\times$ IDF), and the Residual IDF (RIDF). Finally, the HTML features are the *TAG*, which consists of the HTML tags and attribute information, the *Location*, which represents the location information of node in the DOM tree derived from the HTML document, and the *Markedness* which determines how much the word stands out from the normal text in the HTML document (e.g. bold, underlined, italic, etc). The keywords with high harvest rates are used to train a machine learning model which will be applied to estimate the harvest rates for issued keywords which have not been submitted yet.

Jian *et al.* [13] present a framework based on Reinforcement Learning for Deep Web crawling. In the framework, a crawler is regarded as an agent and the Hidden Web database is the environment. The agent perceives its current state and selects an action (query) to submit to the environment (database) according to a long-term reward. The environment responds by giving the agent some reward, *i.e.,* new records, and changing it into the next state. The rewards of unexecuted actions are evaluated by their executed neighbors. Because of the learning policy, a crawler can avoid using unpromising queries, as long as some of them have been issued. Zheng [34] extends the work by [13, 14] by developing a Q-value approximation algorithm that allows the crawler to select a query by learning from the experience of previous queries. The Q-value is the metric that estimates the long-term rewards.

Dong and Li [9], similarly to Jian *et al.* [14], work with free-text forms applying machine learning and heuristics. Consequently, the model can be represented as a single form field f_1 and the domain $dom(f_1)$ is represented by the values according to the query harvest rate.

A sample if taken from the target database behind the form to get a sampling database. The same measures used in Wu *et al.* [31] are applied. Then, it automatically chooses several types of features (number of records retrieved, the length of the values) from the sampling database. Next, it learns a query harvest model from a multi-linear regression approach and employs the model to select queries to submit to the form.

The heuristics include a cost model, a coverage rate, and a query harvest rate. The cost of crawling a Web database as the total number of communication rounds between the crawler and the Web server. It is important to distinguish between the total number of communication rounds and the total number of queries issued. This is because each result page can typically hold a fixed number k of matched records and thus every initiated connection retrieves at most k data records.

The crawling cost $cost(q_i, DB)$ of querying the database DB with query q_i is defined as in Eq. 6:

$$cost(q_i, DB) = \frac{|R(q_i, DB)|}{k} \tag{6}$$

where $|R(q_i, DB)|$ stands for the number of all records in DB matched q_i and k corresponds to the maximum number of records displayed in each result page from the target Web site.

The coverage rate of a query q_i is defined as in Eq. 7:

$$CR(q_i, DB) = \frac{|R(q_i, DB)|}{|DB|} \quad (7)$$

where $|R(q_i, DB)|$ stands for the number of all the records in DB matched q_i and $|DB|$ corresponds to the number of total records in the Web database DB.

Finally, given a target Web database DB and a local database DB_{local} containing the data records already crawled from DB, the query harvest rate of q_i $HR(q_i, DB)$ is defined as in Eq. 8:

$$HR(q_i, DB) = k \times (1 - |R(q_i, DB_{local})|/|R(q_i, DB)|) \quad (8)$$

where $|R(q_i, DB_{local})|$ and $|R(q_i, DB)|$ correspond to the number of data records matched by q_i in DB_{local} and DB, respectively.

The training set is constructed by simulating the Hidden Web crawling using the sampling database. It uses features of the query result in each round to generate the features for candidate queries. Since the sample database is known, the method calculates the harvest rate of each candidate query. Finally, it selects the query with the highest harvest rate as the next query and continues the construction of the training set until all records in the sample database are crawled.

Madhavan *et al.* [22] employs just heuristics to fill free-text and complex Web forms. The goal of the method is to index the resulting HTML pages.

The authors present an algorithm to select input values for text search interfaces that accept keywords and an algorithm for identifying inputs that take only values of a specific type. HTML forms have n inputs and the method introduces the *query template* concept for Web forms. A query template fills a subset of the inputs, called *binding inputs*. The remaining are regarded as *free inputs* and discarded. The number of inputs that make up a template will be referred to as the *dimension* of the template. Multiple form submissions can be generated by assigning different values to the binding inputs. There are no details on how the values are assigned to a template. Each query template and its values are submitted and the results are evaluated to check how much information is retrieved.

A signature function is calculated for the results of submissions, which are compared against each other. A query template is considered *informative* if the generated result pages are sufficiently distinct. Otherwise, it is *uninformative* and thus discarded.

For infinite domain fields, the authors adopt an iterative probing approach to identify the candidate keywords for a field. At a high level, they assign an initial seed set of words as values for the text field and build a query template with the text field as a single binding input. The method exploits the values from a page by identifying the most relevant values to its contents. Thus, the technique uses TF×IDF to choose the values. For the initial values, the top $N_{initial}$ words on the form page are selected.

For the candidate keywords in iteration $i+1$, assume that W_i is the set of all Web pages generated and analyzed until iteration i. Let C_i be the set of words that occur in the top N_{probe} words on any page in W_i. From C_i, the words discarded are those that have so far occurred in too many pages in W_i (since they are likely to correspond to boilerplate HTML that is is found on all pages on the form site), or those that occur only in one page in W_i (since they may be too specific and thus not representative of the contents of the site). The field domains $dom(f_k)$'s of form fields f_k are the remaining values in C_i.

Kantorski *et al.* [15] present an automatic method that combines heuristics for filling both free-text and complex Web forms. The method explores two strategies. The first is how to select good values, or queries, to submit to a particular form in order to retrieve more data with fewer submissions. The second strategy is how to fill the fields efficiently, especially text fields.

The authors employ the concepts of *template* (similar to Madhavan *et al.*'s [22] notion of query template) and *template instance*. Templates are represented by form fields and their combinations. A template instance assigns a value to each field considered for the form submission. A template is *informative* if its template instances retrieve enough distinct data and *uninformative* templates are discarded. The idea is to use information from previous submissions to avoid wasteful submissions (*i.e.,* which do not add new information to the existing set). The informativeness evaluation is repeated for all generated templates and avoids unnecessary submissions in templates of higher order. An instance template considered non-informative will cause instance templates of higher order being discarded.

The choice of values for fields with infinite domains is based on a feedback loop, in which each element has an effect on the next one, until the last element produces feedback on the first element. The idea is use information from the form itself plus the data retrieved from previous submissions as input to future submissions. Heuristics include ranking functions, such as the collection frequency (CF), IDF, and the number of distinct records retrieved (nr) combined into two scores $r1 = cf_t \times idf_t$ and $r2 = nr \times idf_t$.

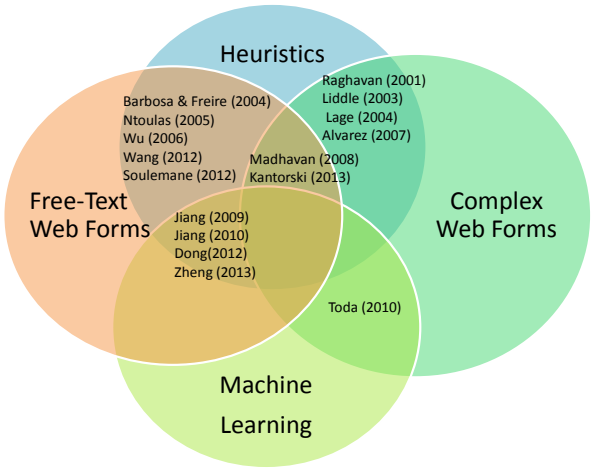


Figure 3: Holistic View

4. COMPARATIVE ANALYSIS

This section presents a comparative analysis of the surveyed methods according to two perspectives. In the *holistic analysis*, each method is studied in its entirety considering the aspects presented in Section 2, *i.e.,* the filling method and the type of interface. In the *Cartesian analysis*, the methods are studied as a collection of dissociated parts.

Holistic Analysis. The surveyed approaches apply heuristics or machine learning techniques or a combination of both. Figure 3 shows how the surveyed approaches are classified under the holistic view. Heuristics are used to simplify the process of WFF and yield good results for both types of interface. Machine learning techniques were added to improve the results reached using heuristics. The surveyed approaches report achieving similar scores using the evaluation metrics, regardless of whether they apply heuristics or machine learning.

Looking at Figure 3, we notice that the research is condensed around certain parts of the space as most of the existing WFF methods [1,2,15,18,20,22,25–27,30, 31] employ heuristics for selecting values for fields in free-text and complex Web forms. The reason for that is heuristics simplify the process of WFF and yet they yield good results.

Methods that rely solely on machine learning techniques (such as [29]) have the limitation of requiring manual labeling. To avoid human intervention, some methods [9, 14] combine heuristics and machine learning. Yet, there are methods that combine heuristics and machine learning to handle free-text Web forms only [9,13,14]. Finally, there are no approaches that combine heuristics and machine learning for handling both type of interfaces (free-text and complex Web forms). This probably happens because the existing methods reach good results for free-text forms. This is not true for complex Web forms, as approaches have yet to be proposed to choose good values for them.

While there are many methods that handle one type of interface, a gap is still open in the selection of proper values for both kinds of forms. Only two methods [15, 22] select values for both Web forms and both use only heuristics [15,22].

Figure 3 also shows the evolution of WFF in the past decade. In the early 2000s, progress was made in terms of free-text and complex forms, separately, using heuristics. In the late 2000s, solutions that work with both free text and complex types of interface, were developed. Since 2010, a considerable progress has been made in terms of machine learning techniques. The adoption of machine learning shows an increase in the level of sophistication of WFF.

Cartesian Analysis. Table 1 presents a summarized view of the methods showing a number of aspects: (i) how they generate seed (initial) values; (ii) how they generate the remaining values; (iii) how much they rely on prior knowledge; (iv) the type of submission method (get/post) of the forms handled by the approach; and (v) dependency on human intervention. Table 1 also contains information about the experiments reported on the surveyed works, such as number of forms and evaluation results. These values cannot be directly used to compare the approaches since experiments have been performed on different forms and evaluated under different metrics. Our goal is just to provide an indication of quality.

Regarding the generation of the initial values for text fields, several methods [1, 18, 25, 26, 30, 31] depend on a predefined list of values. This list is, generally, defined manually or previously built for each form domain. This is shown in the column entitled "prior knowledge". Other methods [2,15,22,27] extract the information from the HTML page where the form is located.

The advantage of getting the initial values from a previously assembled list is that these values tend to retrieve valid results. The disadvantage is that such lists need to be assembled for each form, which becomes prohibitive when dealing with a large number of forms. On the other hand, methods that do not rely on predefined lists have the advantage of being automatic. Their problem is to discover what are the good initial values for filling fields. Also, there is a higher submission cost associated with the task of discovering these values.

Distinct criteria are used by the approaches to select the remaining values. Liddle *et al.* [20] do not select values automatically for text fields as user intervention is needed. Madhavan *et al.* [22] use traditional information retrieval metrics such as TFxIDF while [2, 27] adopt only the term frequency. Kantorski *et al.* [15]

Table 1: Summary of the surveyed works

Method	Text Field Seed Generation	Value Generation	Prior Knowledge	Submission Method	Human Intervention	# Forms	Evaluation Metric
Barbosa and Freire[2]	page containing the form	iteratively submitting queries using values obtained in previous iterations based on term frequency	No	unknown	No	8	Coverage (79% - 8 forms)
Soulemane *et al.[27]*	page containing the form	iteratively submitting queries using values obtained in previous iterations based on term frequency	No	get	No	1	Number of values
Ntoulas *et al.[25]*	manually defined	iteratively submitting queries using values obtained in previous iterations based on term probability	No	get/post	Yes	4	Coverage (84%)
Wang *et al.[30]*	random sample of domain corpora	iteratively submitting queries using values obtained in previous iterations based on set of data retrieved	No	get/post	No	4	Sample Size (2,000)
Lage *et al.[18]*	sample data repository	sample data repository	Yes	unknown	No	27	Precision and recall (93%)
Raghavan *et al.[26]*	label value set table	label value set table	Yes	get/post	Yes	50	Submission Efficiency
Liddle *et al.[20]*	default values	not applicable	No	get	Yes	13	Coverage (80%)
Wu *et al.[31]*	randomly selected values from a pre-exisitng database	Attribute Value Graph Domain Statistics Table (DT)	Yes	get/post	No	5	Coverage (without DT 90%) (with DT 95%)
Alvarez *et al.[1]*	pre-defined domain attributes	domain attributes table	Yes	unknown	Yes	30	Precision and recall (>90%) except in one case
Toda *et al.[29]*	textual document given by user	probability of a field given a word n-gram	Yes	get/post	Yes	5	Precision, recall and F-Measure (73%)
Jian *et al.[13,14]*	page containing the form	query harvest rate using features of values	No	unknown	No	3	Coverage (>80%)
Dong and Li[9]	sample data repository	query harvest rate using features of values	No	get/post	No	3	Coverage (95%)
Kantorski *et al.[15]*	page containing the form	iteratively submitting queries using values obtained in previous iterations based on CFxIDF and distinct rows retrieved	No	get/post	No	11	Coverage and Efficiency (>82%)
Madhavan *et al.[22]*	page containing the form	iteratively submitting queries using values obtained in previous iterations based on TFxIDF	No	get	No	10	Coverage (> 55%)

combine CF together with IDF in the CF×IDF measures and the total number of distinct rows retrieved for choosing values. Wu *et al.* [31] adopt the HR. Wang *et al.* [30] match HR and OR. The value of HR is a limitation in Wang *et al.*'s work [30] because the total number of rows behind the form is needed and, for most real Hidden Web sources, this number is unknown. Finally, there are some methods [9,13,14] that use features about the submissions to select values.

Most approaches handle the *get* [20, 22, 27] submission method, while some can work for both *get* and *post* [9, 15, 25, 26, 29–31]. This information, however, is not evident in some of the surveyed works [1, 2, 14, 18]. We believe that approaches which do not clearly state the submission method use only the *get* method. This submission method has an advantage compared to the *post* method as field values are included as part of the URL in the HTTP request. As a result, the URLs identified can be directly indexed by the search engines.

The number of forms used in the experiments varies considerably (from 1 to 50). Experimenting with real Web forms is tricky as they frequently change, may have high response times or even be unavailable for some periods. These difficulties impact the scale of the tests. Approaches that use machine learning techniques report good evaluation results (averaged across all forms). However, they performed experiments with a small number of Web forms.

Considering the two methods used by the state-of-the-art in WFF (*i.e.,* heuristics and machine learning), we notice that heuristic-based methods have been favored over machine learning. However, many of the approaches rely on user intervention. Both heuristic [1, 18, 20, 25, 26] and machine learning [29] techniques require manual specification of initial values or annotation of the training data. Only one method [9] is completely automatic; however, it handles only free-text forms.

In terms of interface, methods for one type of form are more common than for both. There was also a logical transition from free-text to complex and then to both types. The only approach that handles complex Web forms and uses machine learning is the one by Toda *et al.* [29]. This approach, however, is designed for forms that add rows to a database and not search forms.

5. FUTURE DIRECTIONS

This section presents some insights into trends and future directions in WFF.

Initial seed generation. Most methods report evaluation results in terms of coverage [2, 9, 14, 15, 20, 22, 25, 31]. Precision, recall, and F-measure are also used. [1, 18, 29]. Others propose new evaluation measures [26, 27, 30]. We analyzed the the number of forms used in the experimental evaluation of approaches that apply the coverage metric [2, 9, 14, 15, 20, 22, 25, 31] and how each approach generates the initial values (automatically or using a predefined list). The average coverage for both approaches is similar ($\tilde{8}2\%$); however, experiments with automatic seed generation have been done on more forms. This suggests automatic solutions for generating initial values are more scalable and thus should gain more attention in future approaches.

Reducing human intervention. Human intervention is required by 5 out of the 15 surveyed methods [1, 20, 25, 26, 29]. The amount of intervention varies from labeling data to having to input the values manually. While human intervention may be feasible when dealing with a reduced number of forms, it poses a bottleneck on the approach. Removing human intervention while keeping good results is likely to be the goal of future approaches which aim at being scalable.

Handling complex forms. In complex Web forms, future research could concentrate on modeling the relationship among the multiple attributes in the form. In order to tackle that, understanding Hidden Web forms is necessary. Form understanding is the process of extracting semantic information from an interface [10, 12, 16]. The combination of filling methods and semantic services could be an alternative for improving the automatic filling of complex Web forms.

Using Machine Learning. Regarding machine learning approaches, future investigations could encompass the evaluation of several learning models to determine which is the best suited to address WFF. Furthermore, meta-learning approaches could also include the identification of the appropriate subset of learning algorithms is recommended for the task of choosing field values.

Machine Learning applied to complex forms. Figure 3 shows that there is no method that combines both types of interface with heuristics and machine learning. Thus, a possible future direction may be filling this gap by creating a technique that works for both types of Web forms and applies both types of filling method. Future approaches could adopt heuristic rules for extracting meta information (features) about the submissions and employ machine learning techniques to obtain values for the fields, without human intervention.

Allowing incremental updates. A limitation of all surveyed approaches is the crawling process is always repeated from the start. This is undesirable when dealing with large databases as it has the overhead of discovering already crawled data. Future approaches should aim for incremental updates so as to optimize the process.

Dealing with specific domains. Finally, most WFF approaches have been designed for general crawling. We found no methods that deal with WFF that is focused on a specific topic. Domain-specific features could be explored so as to yield improved results in WFF.

6. CONCLUSION

The Hidden Web represents an important portion of the Web which can only be reached by filling a form. Uncovering Hidden Web data is a challenging task. A scalable approach to gather such data depends on automatically filling forms. The goal is to choose suitable values so that meaningful data can be retrieved.

In this article, we present a survey dedicated to works that address the problem of Web form filling. Two types of analysis were performed over 15 key works in this area. The *holistic analysis* describing each method according to the filling method and the type form that they handle, and a *Cartesian analysis* which considers how the methods perform each of the subtasks involved in the process. This work concludes presenting future directions in this area.

Acknowledgments

The authors thank the anonymous reviewers for their helpful suggestions. This research was partially supported by CNPq/Brazil, project 478979/2012-6. G. Kantorski received a CAPES-Brazil scholarship.

7. REFERENCES

[1] M. Álvarez, J. Raposo, A. Pan, F. Cacheda, F. Bellas, and V. Carneiro. Crawling the content hidden behind web forms. *ICCSA*, pages 322–333, 2007.

[2] L. Barbosa and J. Freire. Siphoning hidden-web data through keyword-based interfaces. In *Brazilian Symposium on Databases*, pages 309–321, 2004.

[3] L. Barbosa and J. Freire. An adaptive crawler for locating hidden-web entry points. In *WWW*, pages 441–450, 2007.

[4] M. Bergman. The deep web: Surfacing hidden value. *Journal of Electronic Publishing*, 7(1):07–01, 2001.

[5] J. Caverlee, L. Liu, and D. Buttler. Probe, cluster, and discover: focused extraction of qa-pagelets from the deep web. In *Data Engineering*, pages 103 – 114, march-2 april 2004.

[6] C.-H. Chang, M. Kayed, R. Girgis, and K. F. Shaalan. A survey of web information extraction

systems. *TKDE*, 18(10):1411–1428, 2006.
[7] K. Chang, B. He, C. Li, M. Patel, and Z. Zhang. Structured databases on the web: Observations and implications. *SIGMOD Rec*, 33(3):61–70, 2004.
[8] V. Crescenzi, G. Mecca, P. Merialdo, et al. Roadrunner: Towards automatic data extraction from large web sites. In *VLDB*, volume 1, pages 109–118, 2001.
[9] Y. Dong and Q. Li. A deep web crawling approach based on query harvest model. *Journal of Computational Information Systems*, 8(3):973–981, 2012.
[10] E. C. Dragut, W. Meng, and C. T. Yu. *Deep Web Query Interface Understanding and Integration*. Morgan & Claypool Publishers, 2012.
[11] D. Florescu, A. Levy, and A. Mendelzon. Database techniques for the world-wide web: a survey. *SIGMOD Rec*, 27:59–74, September 1998.
[12] T. Furche, G. Gottlob, G. Grasso, X. Guo, G. Orsi, and C. Schallhart. Opal: automated form understanding for the deep web. In *WWW*, pages 829–838, 2012.
[13] L. Jiang, Z. Wu, Q. Feng, J. Liu, and Q. Zheng. Efficient deep web crawling using reinforcement learning. In *PAKDD*, pages 428–439, 2010.
[14] L. Jiang, Z. Wu, Q. Zheng, and J. Liu. Learning deep web crawling with diverse features. In *WI/IAT-Volume 01*, pages 572–575, 2009.
[15] G. Z. Kantorski, T. Moraes, V. Moreira, and C. Heuser. Choosing values for text fields in web forms. In *ADBIS*, pages 125–136, 2013.
[16] R. Khare, Y. An, and I. Song. Understanding deep web search interfaces: A survey. *SIGMOD Rec*, 39(1):33–40, 2010.
[17] A. H. Laender, B. A. Ribeiro-Neto, A. S. da Silva, and J. S. Teixeira. A brief survey of web data extraction tools. *SIGMOD Rec*, 31(2):84–93, 2002.
[18] J. P. Lage, A. S. da Silva, P. B. Golgher, and A. H. F. Laender. Automatic generation of agents for collecting hidden web pages for data extraction. *Data Knowl. Eng.*, 49(2):177–196, May 2004.
[19] S. Lawrence and C. L. Giles. Searching the world wide web. *Science*, 280(5360):98–100, 1998.
[20] S. Liddle, D. Embley, D. Scott, and S. Yau. Extracting data behind web forms. *Advanced Conceptual Modeling Techniques*, pages 402–413, 2003.
[21] Y. Lu, H. He, H. Zhao, W. Meng, and C. Yu. Annotating structured data of the deep web. In *ICDE*, pages 376–385. IEEE, 2007.
[22] J. Madhavan, D. Ko, Ł. Kot, V. Ganapathy, A. Rasmussen, and A. Halevy. Google's deep web crawl. *Proc. of the VLDB Endowment*, 1(2):1241–1252, 2008.
[23] M. C. Moraes, C. A. Heuser, V. P. Moreira, and D. Barbosa. Pre-query discovery of domain-specific query forms: A survey. *TKDE*, 25(8), 2013.
[24] U. Noor, Z. Rashid, and A. Rauf. A survey of automatic deep web classification techniques. *International Journal of Computer Applications*, 19(6):43–50, Apr. 2011.
[25] A. Ntoulas, P. Zerfos, and J. Cho. Downloading textual hidden web content through keyword queries. In *JCDL*, pages 100–109, 2005.
[26] S. Raghavan and H. Garcia-Molina. Crawling the hidden web. In *VLDB*, pages 129–138, 2001.
[27] M. Soulemane, M. Rafiuzzaman, and H. Mahmud. Article: Crawling the hidden web: An approach to dynamic web indexing. *International Journal of Computer Applications*, 55(1):7–15, Oct. 2012.
[28] K. Tjin-Kam-Jet, D. Trieschnigg, and D. Hiemstra. Free-text search over complex web forms. In *Multidisciplinary Information Retrieval*, pages 94–107. Springer, 2011.
[29] G. Toda, E. Cortez, A. da Silva, and E. de Moura. A probabilistic approach for automatically filling form-based web interfaces. *Proc. of the VLDB Endowment*, 4(3):151–160, 2010.
[30] Y. Wang, J. Lu, J. Liang, J. Chen, and J. Liu. Selecting queries from sample to crawl deep web data sources. *Web Intelligence and Agent Systems*, 10:75–88, January 2012.
[31] P. Wu, J. Wen, H. Liu, and W. Ma. Query selection techniques for efficient crawling of structured web sources. In *ICDE*, pages 47–47, 2006.
[32] Y. Zhai and B. Liu. Structured data extraction from the web based on partial tree alignment. *TKDE*, 18(12):1614–1628, 2006.
[33] H. Zhao, W. Meng, Z. Wu, V. Raghavan, and C. Yu. Fully automatic wrapper generation for search engines. In *WWW*, pages 66–75, 2005.
[34] Q. Zheng, Z. Wu, X. Cheng, L. Jiang, and J. Liu. Learning to crawl deep web. *Information Systems*, 38(6):801–819, 2013.

Aditya Parameswaran Speaks Out on Human-Powered Computation

Marianne Winslett and Vanessa Braganholo

Aditya Parameswaran
http://web.engr.illinois.edu/~adityagp/

Welcome to ACM SIGMOD Record's Series of Interviews with distinguished members of the database community. I'm Marianne Winslett and today we are in Snowbird, Utah, USA, site of the 2014 SIGMOD and PODS conference. I have here with me, Aditya Parameswaran, who is an assistant professor at the University of Illinois at Urbana-Champaign. Aditya received the 2014 SIGMOD Jim Gray Doctoral Dissertation Award for his thesis entitled, "Human-Powered Data Management". Aditya's PhD is from Stanford, where he worked with Hector Garcia-Molina.

So Aditya, welcome!

Thank you! I'm happy to be here!

Tell me about your dissertation

My dissertation is on Human-Powered Data Management as the title says.

The question is: how do you process large quantities of unstructured data (images, videos and text) with the help of humans? So here, we are talking about using crowdsourcing.

The goal of my dissertation was to figure out the fundamental primitives underlying the techniques you would use to process data with humans. So we figured out that there is a fundamental trade-off in this space, a trade-off between cost (you do need to pay humans if they help you process data), accuracy (humans make mistakes and you do need to take that into account) and finally, latency (humans take a lot of time). So there is a three-way trade-off that naturally appears in this setting. Given this three-way trade-off, our focus was on designing fundamental data processing algorithms, or rather, revisiting fundamental algorithms for data processing. Things like sorting, finding the max, filtering, they all have to be revisited under these new assumptions. The second goal was on how to use these algorithms in data processing systems. So we built a database system that uses humans as a data source (just like any other data source), and also a crowd-powered search engine, that uses humans to process data for you.

We have been recently using Amazon Mechanical Turk in my group, and I see jobs posted there – tasks that involve using a search engine to look up something and to rank the results. Are those coming from you guys?

It is possible! It is possible. The tool that we built for crowd-powered searching, which we call DataSift, starts with a query that a user might issue. This could contain images, for instance, "give me cables that connect to a socket that I took a photo of using my iPhone", and so here is a query that contains some rich information. Ordinary search engines cannot deal with these types of queries. As a result, we need to rely on humans as an integral part of the computation. So my system, DataSift, figures out the right way of decomposing this query into the set of small tasks that are done by humans, as well as automated tasks, which are done by the algorithm, and then combining the two to give accurate results.

How would you decompose the cable/plug question?

The workflow that we found to work well in this scenario is a workflow that we call "Gather-Retrieve-Filter-Retrieve-Filter". It's a mouthful, but the underlying idea is the following… You start by asking the crowd for fully textual reformulations of this query. In order to be able to use a traditional keyboard search API, you do need text. If you have images, there is no way you can use a traditional keyboard search API. So you ask the crowd for textual reformulations of this query. Maybe they may give you "this is a USB socket", or a more complex socket than I probably would not be able to identify. So they give me these textual reformulations. Starting from these textual reformulations, I (as Datasift) go and retrieve a few items that correspond to these textual reformulations using my keyword search API, so this is an automated step. Once I retrieve those items, I can then have humans evaluate those items to see whether they satisfy the query or not. Maybe if it indeed was a USB socket, "USB socket" would be a great keyword to retrieve things from and the items that you retrieve are all likely to be correct so people would say, "Yeah, those are good answers". On the other hand, if you start with a wrong answer, and you have people coming up with "three plug-pin sockets" (I just made that up), you're likely to get the wrong answers and as a result the crowd workers are going to identify that "Hey, these are returning wrong results. You should probably not use this". Starting from that, I can go back and reweight my reformulations. I can focus on the reformulations that gave me the most mileage from the sampling phase. Maybe I might focus on the USB sockets rather than the three plug-pin sockets. So once I narrow down on the reformulations that give me the most bang for the buck, I can retrieve a lot more items for that and once again have humans evaluate those items to finally compose the results for my query.

Most companies […] would not be willing to admit that it's actually humans in the background doing work for you.

Okay! I have some questions for you. How much duplication? How much would you have to pay? How long does it take? And how many duplicates would you

need to have a fairly high confidence for a query like the one we're talking about?

You have a workflow, which contains the six components. Some of them are automatic and some of them are crowdsourced. So internally for some of these components, specifically the filter component, we have developed algorithmic techniques that tell us when to ask additional questions, like how to trade-off between cost, latency, and accuracy. It will tell you, "Hey, look, there is a lot of disagreement about this item, based on the answers that we've gotten so far, so you should probably ask an additional human". On the other hand, there may be cases where you arrive at an agreement between the workers very quickly and therefore you do not need to ask additional humans. These individual operators in this workflow are optimized in the sense that given certain parameters, they optimize the others. Now, in terms of the overall budgeting, you start by having the user of such a system specify the amount of money they want to spend on this workflow. So, along with a query I provide my credit card and I say, "Hey, use $2". For queries that require domain knowledge, I may certainly be willing to spend those $2 to decompose a query into small units of work that are then answered by humans and then get results for the query. You could certainly pay a lot more, so you could get results faster, but when you pay around $2, you end up getting results within half-an-hour… that is the number that we typically see.

So you have to be willing to wait a while. You have to be willing to be a little patient. Basically, the crowds can help you in the cases when it's either a really hard query that you don't know the answer to or when the query requires so much labor that you're not willing to put in, and your time is more valuable. Presumably, this could be a useful building block for an apartment search engine. I have spent hours searching for apartments and if I could just specify what I want and the crowd could figure out…

> ***[…] don't rule out any options at the onset and be strategic.***

You'd like that one!

Yup! So these are the sorts of use cases -- anywhere it requires domain expertise or hard labor. Those are cases that the crowd can help you with.

How much of that half-hour spent is waiting for people to pick up the task versus formulating it? How does that half-an-hour break down?

I've had this happen many times. I would start off with a complicated query that I wanted to issue on Google search. I would start by posing that query and often would find that even with the first five pages of results, I didn't get anything useful. I would then reformulate it and formulate a different query and then go through the same process with that, and keep repeating the process until I get to the results that I want. Oftentimes it takes me half-an-hour or more; I would rather spend that $2 and have someone in the crowd help me out with it.

Okay so most of that time is spent with the person doing what you would have done.

Potentially, in the case where it is a labor-intensive task… In the case where it is a domain specific task like a "USB socket", I may not know the answer myself. Someone who may not be very electronically savvy might want to use a service like this just because they would be able to get answers that they don't already know.

That would be great!

This could be a solution to local IT support.

Yes! Instead of "what would Google say?" which is the best answer to give, it would be, "what would Amazon Mechanical Turk say?"

Yup!

Very good! What industrial impact do you see your dissertation work is likely to have?

I'm glad you've asked that question. We are currently conducting a survey of a number of companies that use crowdsourcing at a very large scale. As it turns out, a lot of companies use crowdsourcing at a large scale. So companies like Microsoft, Google, Facebook, all of them use crowdsourcing at a large scale and they are often ashamed to admit it because it is their secret sauce. Most companies prefer when the clever technology they are using is an algorithm, or better hardware, or something like that, right? They would not be willing to admit that it's actually humans in the background doing work for you. So a lot of companies use crowdsourcing as their secret sauce.

Secret sauce for what kinds of work?

Google, for instance… I'm probably missing out a lot of use cases, but [they use crowdsourcing as a secret sauce] for almost anything that requires training data. Any scenario where you require training data for your machine learning algorithm, that's a scenario where you could use crowds. So for content moderation, for spam detection, for search relevance, all of these are use cases for crowds. Oftentimes, they make subtle tweaks to the algorithm, and they have to then evaluate the results using crowds. So that is like a verification step rather than a training step. Each of these companies use crowds at a very large scale and that's what we've been discovering when we've been talking to these people. In fact, a lot of them are certainly trying to optimize for the tradeoff between costs, latency and accuracy, but some of them have not even gotten the basics right. So the techniques that myself and my collaborators have developed could certainly benefit these companies because they are doing this at scale and if they use optimized plugins or the algorithms that we've developed, they could certainly get a lot more mileage from the same dollar spent. So they could get the results quicker, they could get results of a higher quality, and so on.

When you say "at scale" do you mean like millions of worker tasks per day coming from these places? What does "at scale" mean nowadays?

So I don't think I'm allowed to talk about how many tasks these companies pose, but at the very least it is millions of tasks every week…and a lot of companies do even more. It suffices to say there's a lot of crowdsourcing being done and often for a lot of these companies this is at a scale larger than Mechanical Turk. Mechanical Turk is a toy example for them. These guys actually use outsourcing firms in India, Philippines and so on, and these outsourcing firms are middlemen. They will then hire employees who come and work for them 9-to-5, doing these micro tasks day in and day out. So given that you have these workers in-house, you have the ability to track their progress, and you have the ability to incentivize (bonuses that you could provide to the guys that are doing well). So it's a different set-up, but that's how some of these companies operate at scale.

Very interesting! Is there something that you know now that wish you had known earlier in your grad school and post-doc career?

When I started my PhD, I was hell-bent on either doing a startup or joining a research lab. Those were my only two career goals when I stared my PhD. Very soon I realized that my heart lay in research rather than doing a startup, at least at the start. I was passionate about getting to the bottom of things rather than dealing with management and dealing with all different issues that come up when doing a startup. By year two, I was all for joining a research lab. Year three was when Yahoo Research collapsed, and that's when the bubble burst for me. Yahoo Research was the place to go because there were a lot of really smart people, they had access to real problems, they had access to real data, and they were given the freedom to do whatever they wanted. That was how it was back then. The bubble burst because this is not a sustainable model. If you do not contribute back to the company, then it's not going to work out in the long run. That was when I started thinking and introspecting as to whether I really wanted to be in a research lab or would I rather have students of my own, to leave a legacy, to champion an area, to have a research vision, to have people working on fragments of that research vision, and moving the field forward with something that I can truly call my own rather than my company's or my team's. That's when I started thinking seriously about academia. It was not until year three when I started thinking seriously about academia.

If I had to do it all over again, I would have not eliminated that as a potential career goal at the start. So in the first two years, I was just having fun as a grad student. Not having fun in the sense of not doing work, but I was having fun working on all sorts of problems. I was going for breadth rather than depth. I was collaborating with people at Yahoo. I was collaborating with people at Microsoft. I was having collaborations with folks not in my research group but in other research groups at Stanford and I was having fun. But this was not getting me deep into a research topic such that I could have something substantial and meaty to say during my job talk. And that is something that happened more along the way. I wish I had figured this out a little earlier and mentally prepared myself a little earlier. I am not sad with where I am right now, but I would have mentally prepared myself to be an academic a little earlier.

I don't know about that. Your dissertation won that prize, so there's got to be meat there and we hired you! So I hope that was a happy end to the job hunt. I'm not sure that you really needed to start thinking pre-professionally any earlier than you did, but still, that's your advice and it stands.

So at the very least, maybe I got here by chance, because I got onto an area that was relatively unexplored and therefore I was lucky. But at least to others I would suggest that they don't rule out any options at the onset and be strategic. Have fun, have a lot of collaborators, have fun collaborating with a lot

of smart people, but be strategic and think long term at the very least. So it worked out for me at the end, but I wouldn't have expected it, right? In year three and year four I was panicking because I didn't know what was going on. I didn't have a dissertation topic, and that's when I chanced upon this exciting new field and very quickly we published a number of papers on it. If that had not happened, I don't think I would have landed an academic job.

Well thank you very much for talking with me today.

Thank you so much!

Andy Pavlo Speaks Out on Main Memory Database Systems

Marianne Winslett and Vanessa Braganholo

Andy Pavlo

http://www.cs.cmu.edu/~pavlo/

Welcome to ACM SIGMOD Record Series of Interviews with distinguished members of the database community. I'm Marianne Winslett and today we're in Snowbird, Utah, USA, cite of the 2014 SIGMOD and PODS conference. I have here with me Andy Pavlo, who is a professor at Carnegie Mellon University. Andy received the 2014 SIGMOD Jim Gray Doctoral Dissertation Award for his thesis entitled "On Scalable Transaction Execution in Partitioned Main Memory Database Systems". Andy's PhD is from Brown University where he worked with Stan Zdonik.

So Andy, what do you got there? Can you rotate it so that we can read it in the camera?

This is my Jim Gray Doctoral Dissertation Award.

Woah! Look at that! Now, is this body armor?

No, no, this is my belt. I've only won two awards in my life, okay? I won class clown in high school and then I won this. And you know what? I may never win another award ever again, so I'm going to relish this as much as possible. So I don't want to take away from Aditya because he won as well, but he's won best paper award and things like that. So this is it for me. So I'm going to sleep with this every night for the next year and then someone else will win for 2015.

So you just strap it around your waist?

Well, maybe, but when you give a talk you have to hold it like this because people can't see above the podium, right? And then I'll have my students hold it up behind me and walk in with it in a procession whenever I give a talk. Just for one year, and then after that, it will be retired. It will go in a shelf in my office at Carnegie Mellon.

> ***Your job talk needs to have tons of graphs. Lots of jokes, lots of graphs.***
>
> ***Michael Stonebraker***

Okay! And which parts of it did SIGMOD give you?

Well SIGMOD gave me the plaque. The belt part, you know, I made myself.

You made it yourself?

Well, I got it lasered.

Okay. Well it looks pretty spiffy. Maybe we should do that for every award winner.

Again, so Aditya… he worked hard, he got the award too. He had the option to get a belt. He could have made a gold chain to hold his. He decided not to. I decided to do it. So there you go.

Now by saying that you got the award for class clown, you're setting the bar very high for this interview.

I don't know about that much, but okay.

Okay, well let me ask you first. What's your dissertation about?

We started around 2007, 2008 when the NoSQL movement was gaining prominence. They were all out there saying, "Well, the only way that you can scale up a large scale distributed database system to support a large number of concurrent users with concurrent operations is if you give up transactions entirely", right? So you see these in systems like Google's BigTable or Amazon's DynamoDB, and in the open source world there's Cassandra, MongoDB and Riak, sort of implementations based on those ideas. So when we started we said, "Well, let's not give up transactions. Let's see what we can do in a modern architecture. Can we still have a distributed database system that can still support strong ACID guarantees?" So that's sort of what culminated in the system called H-Store that I helped develop with people at Brown, at MIT and Yale and then what eventually became VoltDB, but it was originally out of Vertica.

The basic concept of the system was that we were going to have a main memory execution environment. We were going to have serial execution transactions across multiple nodes. We would support stored procedures only, we would use a real lightweight logging scheme. So that allowed us to be able to support the large number of concurrent users that you need in these modern Internet applications without having to give up transactions. So I think that was a pretty significant contribution.

And did you achieve that totally?

To some extent, yes. I mean, there are certain aspects of applications where H-Store is not the right architecture. I'm totally upfront about that, right? But I think we've seen this in the commercial space with VoltDB. VoltDB is the commercial implementation of H-Store's architecture. There are a large number of applications where the design we used in H-Store is absolutely appropriate and it works really well. But there are a large number of applications, for instance, anything with a social graph like Facebook or Twitter where you have arbitrary users connected together, that don't partition very well, so H-Store is not the right architecture for that.

So what were the key architectural or implementation choices that you made that make it a success?

The main one was that it's main memory storage engine. That's not a new idea. The first work around this concept was in early 1980s. What makes it different this time around is that we're finally at the point where the price and the capacity of DRAM made it possible to store all but the largest OLTP databases (these front end transaction processing databases) entirely in main memory. Once you have everything in main memory, a lot of the design decisions that came out of the original databases from the 1970s don't make sense anymore – the stuff from Ingres and System R. So for example, you don't need a heavy weight concurrency control scheme with locks and latches in order to mask the latency of disk because there is no disk. So in a traditional system, a transaction could stall anytime because you had to touch data that wasn't in main memory, that was in disk, and therefore you had to allow other transactions to run at the same time in order to mask that stall. But now since everything is in main memory, you're never going to have a disk stall, so it doesn't make sense to allow multiple transactions to run at the same time.

Another key concept that we did in H-Store was that we use a lightweight concurrency control scheme based on partition level locks where each partition is going to get assigned a single threaded execution engine for transactions. And so, because it is single threaded, when it executes a transaction it knows that no other transaction and no other thread is running at the same time to touch that same data and therefore you don't need any locks and latches at the lower stuff, more at the fine grained level. So that allows you to run really fast.

What about cache misses?

There are no cache misses. I mean are you talking about L1, L2? Those are so fast. Having to go to DRAM it's significantly faster than having to go to disk, so those aren't really a big issue for us.

And what part of that giant system did your thesis focus on?

The thesis itself wasn't just, "Hey, we built this system, ta da", right? The fundamental part was "Hey, there is this new architecture, we can do better than a traditional system". Then going beyond that was, "What are all the problems where this doesn't work out?". So the rest of the thesis is saying and identifying, "Well here is the issues we have when working in this kind of operating environment and what we can do to fix it".

Three main parts came after the basic system design. The first one was coming up with automatic techniques to take an arbitrary application and figure out the best partitioning scheme and how to split it up across a cluster of nodes so that you maximize the percentage of the transactions that only touch data at a single partition at a single node. Then you avoid any of the slowness involved with two-phase commit like Paxos.

The one thing that I think I realized in grad school that made me successful is really focusing on this single project.

The second part is related to multi-partition transactions. It happens that for some applications you are not able to get rid of these distributed transactions, these multi-partition transactions entirely. This could be either because the application simply does not partition in a good way (the Twitter/Facebook example is a good one), but also might be because of weird legal reasons. So to give an example, we've visited PayPal in their early days and they had this weird legal restriction where customer accounts from different countries could not be on the same physical hardware for some reason. So that means if you were using a system like H-Store, this would never work because that's always going to be a multi-partition transaction. And so the second piece of the thesis resided in using machine leaning techniques to figure out, when a transaction request comes in, if it's a single partition, a distributed transaction and what partitions it actually needs to touch so that we only need to allocate or lock the bare minimum of resources we need for that transaction. Then while it's running, we can identify when it's done with those resources, go ahead and release them and let the next transactions to start running. This is sort of doing an early two-phase commit process. So that was the second piece.

We've tried to minimize the number of distributed transactions, we have identified when the transactions come in and whether they're distributed or not, but we still have these distributed transactions. We have these points where the execution nodes in the cluster are idle because they're locked by some other guy in another node and therefore they're waiting for the next request to work on, and they're sitting and doing nothing. This is because we're using this serial partition level locking scheme. So the third piece of the thesis is a speculative execution technique where we identify when we're idle at a remote node because of a distributed transaction. We go ahead and peek in our queue for that partition and try to pick out transactions that commute with what work the distributed

transactions have done so far. We show that by using these machine learning techniques that we use for the second part, we can identify transactions that will finish in enough time and won't interfere at all with the distributed transactions. So everybody can do a root commit in the end and we'd all claim we don't violate any isolation consistency guarantees.

Do you know anything now that you wish you had known when you were a grad student? Or during your job-hunt?

Well, that's two questions… For grad school, there was this really critical point in my second and a half year where I needed to decide whether I wanted to go and continue building H-Store (the academic version of the system) or whether I should just leverage what the VoltDB guys were doing in the commercial side. The H-Store relationship with VoltDB is kind of incestuous where we were separate projects that came together as one project then we separated again then we came back together. I thought I was going to go forward using everything of VoltDB, but my advisor to his credit, he said, "look I really think that you'd be better off from a research standpoint, if you did another fork, pulled back some changes from VoltDB in the H-Store code but then rewrote a lot of the stuff that you need for research". This was a hard choice, right? Because this means that for two years or so I'd be writing a lot of code and not getting publications done, but he really pushed me to do this. In hindsight it was the right thing to do. To his credit, he really stood by me for those two years when I was not getting any publications done because I was spending all this time writing the system. He just let me go and do my thing and thankfully, it all worked out. So I'm very grateful for his faith in me in pulling this off.

The one thing that I think I realized in grad school that made me successful is really focusing on this single project. A lot of times I see grad students that are focusing on different things, different projects, and when it comes the time to go on the job market, whether it's an academic or industrial position, they have this tenuous or weak connection of trying to say, "I did this project and this project that are all together in the same package", right? I think it's kind of transparent. Whereas in my case, I was able to go on the market and say, "Hey look, I built the H-Store system. I'm the H-Store guy. Here is all the work based on a single system". I can talk to length about any part about the system because I spent so much time working on it. I credit this idea from Dan Abadi, who's now at Yale, but he was at MIT working on the C-Store project, which was the predecessor to H-Store. So when he was in the job market, whether he knows this or not, he was the C-Store guy and he was really successful in that regard. So I try to emulate that or copy that idea of being the H-Store guy when I was in grad student and I think I was pretty successful with it.

As far as what I wish I knew now about being in the job market… Stonebreaker told me when I got invites to go interview at some schools, at IBM Research, Intel Labs, really awesome research places. He basically said, "Look, you are the only database systems person on the market. Your job talk needs to have tons of graphs. Lots of jokes, lots of graphs". So my job talk was essentially about my gambling addiction at the greyhound dog track. So I went this whole thing about how I go gambling, go see the dog track, and from that, on how it gave me ideas on how to make my database run faster. Actually I don't go to the dog track, and greyhound racing is deplorable. It was a joke, right? And only at one place, one guy thought the joke was real. Everyone knew, that guy is just joking that's fine. There was one guy at IBM who came up to me afterwards, "Hey man, I really like going to the dog track. When do you want to go?" I was like, "no, you totally misread that". I didn't start grad school thinking like, "oh, I absolutely need to go to Carnegie Mellon or a top school like that". I didn't set out to do this. I was just around some really smart people who had a lot of good guidance and I think I worked pretty hard to build this thing and everything worked out.

I've only won two awards in my life, okay? I won class clown in high school and then I won this. And you know what? I may never win another award ever again, so I'm going to relish this as much as possible.

You described it as a big project and you're the only systems guy, does that mean you had like a dozen bosses?

No, so the project started off being myself, another PhD student's at Brown, and two PhD students' at MIT. It was a lot of people at the very beginning. This is like 2007, 2008. We all worked for about a year building the core basic system out. Then around 2008, they went out and forked the code and made VoltDB

(the company). Then all the other PhD students went off and did other things. I, myself, went off and did some stuff with the MapReduce with Stonebreaker, David Dewitt and Sam Madden. All the while thinking that VoltDB was going to add the stuff that I needed. They kept saying, "oh, next quarter, next quarter we'll do it don't worry". Then finally when I went back after doing the MapReduce stuff to go work on H-Store again, I asked, "Hey I need *x*, *y* and *z*, are you guys going to do it?" and they said, "No, this is not what customers are asking for. We're not going to do it". I'm not faulting them, that is a business decision and that's fine. That was the point where Stan was like, "look you should go and fork your code and do your own thing". That's when I did that. When I went back to the system, everyone else was gone. I worked with Evan Jones a little bit but he was sort of off doing the relational cloud stuff up at MIT. So about a year or so, I was trying to cobble together whatever resources I could get, masters students, undergrads, all at Brown to try to help me build this thing…but it was a lot of time, a lot of late nights. I did eventually borrow some code from VoltDB but a lot of the core transactional stuff was rewritten (twice actually) from scratch. I don't recommend it. It probably was not a healthy lifestyle and it was certainly not sustainable. I'm not keeping that same pace at Carnegie Mellon because I have other things to work on. But yeah, it was a lot of code in over a two or three year period to make this all work. It wasn't just me, but for that one period I got to get whatever resources I could to make this work.

Thank you very much for talking with me today.

Thanks for having me. It's a blast. Thanks.

DASlab: The Data Systems Laboratory at Harvard SEAS

Stratos Idreos

Harvard University
http://daslab.seas.harvard.edu

ABSTRACT

DASlab is a new laboratory at the Harvard School of Engineering and Applied Sciences (SEAS). The lab was formed in January 2014 when Stratos Idreos joined Harvard SEAS. DASlab curently consists of 3 PhD students, 1 postdoctoral researcher and 9 undergraduate researchers while it is set to double its graduate student population in the next one to two years. The lab is part of a growing community of systems and computer science researchers at Harvard; computer science faculty is scheduled to grow by 50% in the next few years.

The main focus of DASlab is on designing data systems that (a) make it easy to extract knowledge out of increasingly diverse and growing data sets and (b) can stand the test of time.

1. INTRODUCTION

At DASlab our long-term goal is to assist in minimizing the time it takes to turn data into knowledge by designing and building novel data systems, tailored for the new and ever-evolving challenges of a data-driven world.

Today knowledge hides in plain sight. It is quite likely that we already have all the raw data to discover solutions for many of the worlds big scientific challenges and we just do not have the proper systems (i.e., algorithms and data structures) to analyze the existing data or we simply do not know what to look for. As the size of the data we collect grows, data systems have become one of the most fundamental bottlenecks for data analytics; storing, accessing and analyzing raw data has to happen before any meaningful observations can take place and as it stands today we cannot even move, let alone store, access and analyze all our data.

In this document, we provide a brief overview of some of the research projects that are currently under way at DASlab and discuss our long term goal and motivation. Our research is driven by two fundamental goals outlined below.

Everyone should be a data-scientist. Knowledge is power and should be accessible to everyone. The more data we collect, though, the harder it becomes to make sense of the data. Today analyzing data requires expertise and resources that very few individuals hold. What if we made it equally easy for data scientists, data systems experts, astronomers, biologists, statisticians, as well as the general public to extract knowledge out of data via auto-tuning, interactive and intuitive data systems?

We should not have to design new data systems every decade. Computer science is a fast moving field. Every few years the environment changes dramatically with new hardware and new application requirements (meaning new data formats, models, etc.). What if we did not have to completely and manually redesign our data systems, algorithms and data structures every decade? What if we spend this energy designing more powerful ways to analyze data with systems that can automatically evolve to match the new environment?

The two high level goals above are strongly interconnected and they both lead to a vast array of fundamental computer science challenges. Our focus is on designing big data systems that are easy to use, i.e., with as few knobs as possible and with as little human input as possible; systems that just work, no matter the underlying hardware properties. Our vision is that data systems should be intuitive to use and interactive, guiding the user towards the interesting patterns in the data, doing just enough to generate the maximum insight within a limited time and resources budget. Indexing, tuning, optimization and other complex technical features should all be actions that are continuously active but always hidden from the users.

2. RESEARCH DIRECTIONS

In this section, we briefly describe some of the ongoing research projects at DASlab, focusing on discussing motivation and goals.

2.1 Self-designing Data Systems

Different applications and different hardware require different system designs to achieve optimal performance (i.e., throughput, query-latency, energy-performance etc.). Yet, so far, all data system architectures are static, operating within a single and narrowly defined design space (e.g., NoSQL, NewSQL, SQL, column-stores, row-stores, etc.) and hardware profile. Historically, a new system architecture requires at least a decade to reach a stable design. However, as we go deeper into the big data era, hardware and applications evolve continuously, leaving data-driven applications locked with sub-optimal systems. The more our businesses, sciences and every day life become data-driven, the more this becomes a fundamental shortcoming.

In this project, we ask the following question: *How many aspects of data system design can be abstracted and automated?* The end goal is that such systems take the "shape" of the data and queries with minimum human intervention during the design phase. This leads to systems that are fully tailored for a specific scenario and hardware profile, yet they are fast to design and implement. The ideal end result is fully self-designed and self-implemented systems.

We are currently experimenting with several ideas and designs about how data systems can self-design some of their most critical components. For example, inspired by the theory of evolution, a self-designing system may deploy multiple competing solutions down at the low level of its architecture such as using various combinations of data layouts, access methods and execution strategies. Then "the fittest design wins" and becomes the dominant architecture until the environment (workload and hardware) changes again. As new data and queries come in, a system evolves such that its architecture matches the properties of the incoming workload. Other research directions include the adoption of ideas from machine learning to make system design decisions, the use of advanced statistic models and what if analysis, as well as applying these concepts in combination.

Essentially, there are two distinct opportunities with this line of work. The first one targets system design: Given a set of data, queries and hardware, return a data system. The second one targets tuning of an existing system in an environment with diverse workload and hardware properties. In such cases, the complexity of the system tuning options makes it impractical to manually tune as the environment often changes over time as well.

As a more concrete example to what a self-designed system can be useful for, imagine the scenario where a research lab or a company has multiple different workloads that require very different system architectures while new requirements appear frequently. With self-designed data systems the data scientists would only need to point the system to the different datasets and it would take the appropriate shape for each one of them.

What does the appropriate shape mean? Query languages, data models, data layouts, physical layout, query processing operators are only a few of the design dimensions that a self-designed data system needs to decide upon. In our research, we started working throughout the whole stack by keeping the least common denominator so as the resulting system is generic and expandable enough. We consider the relational data model as well as RDF and other hierarchical data models, and we currently investigate the physical storage of such data, including the traditional row-store and column-store layouts as well as hybrid layouts.

This line of research creates numerous opportunities to bootstrap new applications, to automatically create systems that are tailored for specific scenarios, to minimize system footprint and automatically adapt to new hardware.

2.2 Queriosity: Auto-exploration

Data exploration is the natural paradigm for extracting knowledge out of data. It involves a series of steps such as: take a look at the data, try out a few models to see if they fit, look for outliers, learn from this experience and data seen so far, zoom into or out of this data set and repeat until satisfied with the knowledge gained. The output of each step is the input of the next one. Yet modern data systems are not designed with data exploration in mind.

Modern data systems are based on strict forms of interaction and they are designed for expert users who know enough both about the application domain and about how to set-up, tune and interact with data systems. This state of affairs restricts significantly the range of people who can actually explore data as well as the time it takes to extract knowledge out of growing data sets.

In this project, we ask the following question: *What if we had systems that can automatically explore rich data sets and report back interesting facts?* We are designing an autonomous "data robot", termed Queriosity, a portmanteau of *query* and *curiosity*.

Queriosity's goal is to explore data sets and figure out interesting patterns. It continuously learns about the fastest way to explore a given data set,

observes how interesting its findings are for the user and adjusts its strategy accordingly. As a result, with Queriosity the demanding task of data exploration is reduced to just the following two steps on the part of the user:

Here is a data set I want to explore;
Show me something interesting.

No more input from the user is needed other than confirming that a given insight is useful or not. Queriosity finds new insights as well as new relevant data sets automatically.

The design of an autonomous data exploration system that learns continuously presents various challenges on the conceptual as well as the system level. On the conceptual ground, for instance, we need to answer questions such as the following: What statistical properties of a dataset mirror user's interest? How can these indicators be determined without a priori information of either the dataset, the domain or both? What does it mean for an exploratory system to learn from experience? On the other hand, building the first autonomous data exploration system we are confronted with system level challenges such as the following: How to design a system that remains interactive while exploring and learning from potentially Petabytes of data? What form should the eventual system take i.e., collaborative, stand-alone or a hybrid? How does such a system integrate with myriads of existing data systems and data representations? How can it be designed to remain relevant decades from now?

With the prevalence of paradigms such as data-intensive science, Internet of things and information governance, we envision Queriosity finding application in virtually all domains as a *personal data scientist* that assists, data scientists in businesses and scientific research as well as people in every day life that try to make sense of the data around us.

2.3 Interactive and Visual Analytics

Why should researchers and data scientists have to learn complex languages and interfaces to use a data system? Why should they wait for several hours or days to analyze big piles of data if they only care about a small part? A data scientist should be able to simply point to the data and start extracting knowledge immediately.

The fundamental problem we are addressing in this project is that it is not easy anymore to extract knowledge out of data. Modern data scientists are confronted with complex systems and tools that lead to the following problems: (a) they are slow as they are designed with the traditional notion of processing all data to always give complete answers, (b) they rely on complex interfaces and languages which are meant only for experts of a specific system category and (c) they require expertise in terms of system tuning. The side effect is that data analysis becomes slow and requires expertise which becomes harder and harder for a single person to acquire. This directly translates to a significant financial overhead for businesses and scientific research labs. For example, a delay in the acquisition of knowledge implies lost business opportunities, while the need for more expertise implies that more personnel is required.

In this project we enable data analytics via intuitive touch interfaces and gestures and work towards a new class of data systems that are designed from the ground up to be interactive and tailored for data exploration. A data scientist can see, touch and explore data directly on a tablet device and can fully drive low-level query processing actions and complex analytics via gestures. The new paradigm is that the system continuously adjusts its storage and data access patterns to match the exploration path, it accesses only as much data as needed to instantly provide enough visual feedback to the data scientist, introducing a radically new data processing paradigm that is tailored for interactive exploration. Instead of having to learn complex languages and interfaces, data scientists interact with the data system via intuitive gestures. Instead of having to wait to set up and tune the system and its low level knobs, the system automatically adjusts to the tasks a data scientist performs. Instead of having to wait long stretches of time to get a complete answer over a big data set, the system gives a quick answer back and data exploration takes place as a continuous interaction between the data system and the data scientist which step by step leads to the interesting part in the data.

For example, in our current working prototype a data scientist can work directly over an iPad tablet using touch gestures. Data is visualized in the form of various geometric shapes (e.g., rectangles) and a data scientist can declare various queries/actions such as scans and aggregations and can start touching the data with slide and zoom-in/out gestures as opposed to using complex query languages. As data is being touched, the system plots interactive graphs to communicate the observed data patterns only for the actions performed and only for the data touched. As the data scientist adjusts the gesture characteristics and area of the data touched, the graphs get increasingly more complete and the data patterns become more and more apparent. The system is interactive enough such that every sin-

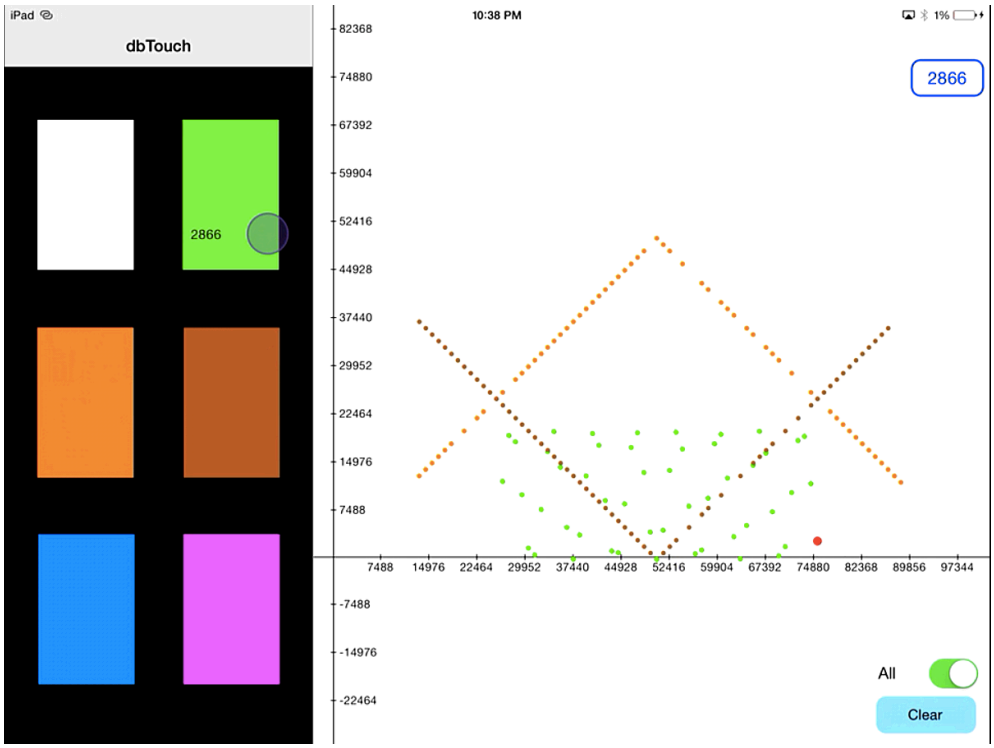


Figure 1: Interactive gesture-based exploration with dbTouch [18,26].

gle touch instantly translates to a visual change in the interactive graphs and data scientists can immediately use this information as feedback on which parts of the data they should focus next. Figure 1 depicts a screen-shot of our current prototype; the grey circle shows where a finger touches a data object which results in data plotted on the right hand side. With a few simple slide gestures we can get a quick understanding about the data distribution in the data set represented by the rectangle objects. The x-axis in the plot represents row-ids while the y-axis represents data values.

We are currently exploring similar ideas for data analytics via 3D gestures and in virtual reality environments. The common challenge is the design of data systems kernels that are interactive at their core regardless of the data sizes as opposed to state of the art systems that are monolithic.

2.4 Indexing in Modern Data Systems

Modern data systems do not rely on indexing in the same way past systems did. At a first glance this is for a good reason. For example, modern systems that target analytical workloads can perform very efficient scans using technologies such as column-storage which allows for reading fewer data items than a traditional row-store. In addition, relying on fixed width and dense columns allows for scans with tight for-loops which give excellent opportunities for prefetching and eliminate if statements and branch misses from the critical path. Similarly, exploiting SIMD and modern multi-core CPUs, as well as working directly over compressed data, allow for extremely fast scans. Another significant trend is that with larger memory sizes, data systems applications will store all hot data in memory, removing the major bottleneck of reading data from disk. At the same time, advanced query processing techniques allow for scans to answer multiple queries at the same time in environments where concurrent query requests are the norm; essentially we can answer N queries with the cost of a single scan.

In light of all these developments here we ask the question: *Are secondary indexes still useful today and if so in what form?* In our research we consider multiple variations of secondary indexes, from full indexes like a B$^+$-Tree to partial indexes like zonemaps. We analyze modern data systems using both models and experimentation. Our initial findings suggest that the decision of index use requires not only the traditional parameter of query selectivity but also other system conditions. This is especially true in light of a new trend in modern analytical workloads: increasing query concurrency.

We also find that designing strict index structures sacrifices performance when the workload changes. This leads to the investigation of dynamic/morphable data structures that can balance read, write and space amplification on-the-fly.

2.5 Hardware Software Co-design

Typically, software is being designed based on, apart from the workload, the available hardware. In this line of research, we investigate opportunities to reverse this relationship. The driving force of hardware development has been the applications, hence, in a perfect world hardware should be co-designed with the software. As a concept this is not a new idea; database machines have been studied extensively in the past and have been revisited in recent years. There were good reasons to abandon such ideas in the past and there are good reasons to explore new opportunities now. However, rather than proposing a full hardware design, in this project we investigate more hybrid solutions where the architecture of a database system remains generic enough and can be assisted by strategically placed accelerators that exploit game-changing modern hardware properties. Below we describe two of the directions we are pursuing.

Near Memory Processing. As the cost of accessing main memory becomes increasingly expensive and the memory size becomes larger, the main bottleneck of any data system is caused by fetching data to the processor. Here we ask the question: *Which parts of the processing can be offloaded in a generic enough way to functional units near memory or disk?* We design near-data query operators like select and project and we currently investigate the design of other operators, like hashing, sorting, and aggregations. Envisioning a complete architecture for near memory processing hardware is not as straightforward as pushing all actions close to the data. For example, while it is easy to see that it makes sense to push all selections, when it comes to more complex operators such as joins the discussion becomes more interesting; data after a join may be bigger than before, and so we may end up pushing more data up the memory hierarchy.

The Relational Disk. One of the most challenging questions in data management recently has been the bridging of the requirements of transactional and analytical workloads. There have been many proposals on the data systems layer, while, arguably, it has proven to be very difficult to provide a system capable of doing both analytical and transactional processing equally efficiently. Here we ask the question: *What is needed from the hardware in order to build a data system capable of bridging the analytical and transactional processing?* A disk-subsystem capable of delivering column-store performance when accessing a single column, yet supporting row-wise updates is the ultimate goal. Such a storage subsystem which we call *The Relational Disk* requires radically new hardware design assuming, instead of generic file structure, relational file organization, in order to be feasible to build in hardware the necessary technology that would give both row-wise and column-wise accesses.

3. INSPIRATION AND PAST WORK

For our work at DASlab we draw inspiration from several lines of work in the DB community. Most prominently we align with other exciting work on database architectures, adaptive systems and data exploration. Our own past work lies on a breadth of topics and is joint work with several labs, more frequently with colleagues from the database groups at CWI, EPFL, HP Labs, Microsoft Research, IBM Research, Google, Paris Descartes University, and NUS. The big chunk of our past work has focused on a series of topics on how to minimize the cost of bootstrapping database systems. Below we briefly point to some of these research projects.

With our work on adaptive indexing we have studied the design of modern data systems where indexing requires zero human intervention and tuning. Indexes are created adaptively and incrementally as a side-effect of query processing [10, 15, 17, 9, 16, 7, 20, 32, 35, 31, 8]. Building on top of such adaptive ideas our work on column/row hybrids proposes a system with adaptive storage components [4] that can choose the optimal layout on-the-fly.

Similarly, our work on adaptive loading presents the design of systems that do not require data loading up front. Instead, such systems can work directly on top of raw data, while matching the performance of traditional systems [12, 2, 3].

Our work on touch-based interactive systems introduced the idea of systems that are interactive enough to immediately react on gestures that represent query actions over big data sets [18, 26].

Our past work also includes work on core architecture topics such as column-store architectures [1, 5, 13], exploiting compressed indexes [14], designing column-stores that support stream processing [25, 29, 30], effectively utilizing compression during join processing [24] as well as proposing statistics oblivious access paths [6].

Furthermore, our past work has also focused on distributed query processing, designing distributed systems for several different data models, e.g., relational [22, 19], information retrieval [33, 34] and RDF [28, 27].

Finally, much of the above work can be captured under the umbrella of data exploration techniques in an effort to design database kernels that support data exploration at their core [11, 23, 21].

4. ACKNOWLEDGMENTS

Our work is currently funded by the Harvard School of Engineering and Applied Sciences, the US National Science Foundation, the Swiss National Science Foundation, Facebook, and Google.

5. REFERENCES

[1] D. Abadi, P. A. Boncz, S. Harizopoulos, S. Idreos, and S. Madden. The design and implementation of modern column-oriented database systems. *Foundations and Trends in Databases*, 5(3):197–280, 2013.

[2] I. Alagiannis, R. Borovica, M. Branco, S. Idreos, and A. Ailamaki. Nodb: efficient query execution on raw data files. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 241–252, 2012.

[3] I. Alagiannis, R. Borovica, M. Branco, S. Idreos, and A. Ailamaki. Nodb: efficient query execution on raw data files. In *Communications of the ACM, Research Highlights*, 2015.

[4] I. Alagiannis, S. Idreos, and A. Ailamaki. H2O: a hands-free adaptive store. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 1103–1114, 2014.

[5] R. Barber, P. Bendel, M. Czech, O. Draese, F. Ho, N. Hrle, S. Idreos, M.-S. Kim, O. Koeth, J.-G. Lee, T. T. Li, G. M. Lohman, K. Morfonios, R. Müller, K. Murthy, I. Pandis, L. Qiao, V. Raman, R. Sidle, K. Stolze, and S. Szabo. Business Analytics in (a) Blink. *IEEE Data Eng. Bull.*, 35(1):9–14, 2012.

[6] R. Borovica, S. Idreos, A. Ailamaki, M. Zukowski, and C. Fraser. Smooth scan: Statistics-oblivious access paths. In *Proceedings of the International Conference on Data Endineering (ICDE)*, 2015.

[7] G. Graefe, F. Halim, S. Idreos, H. Kuno, and S. Manegold. Concurrency Control for Adaptive Indexing. *Proceedings of the Very Large Data Bases Endowment (PVLDB)*, 5(7):656–667, 2012.

[8] G. Graefe, S. Idreos, H. Kuno, and S. Manegold. Benchmarking adaptive indexing. In *Proceedings of the TPC Technology Conference on Performance Evaluation and Benchmarking (TPCTC)*, pages 169–184, 2010.

[9] F. Halim, S. Idreos, P. Karras, and R. H. C. Yap. Stochastic Database Cracking: Towards Robust Adaptive Indexing in Main-Memory Column-Stores. *Proceedings of the Very Large Data Bases Endowment (PVLDB)*, 5(6):502–513, 2012.

[10] S. Idreos. Database Cracking: Towards Auto-tuning Database Kernels. *CWI, PhD Thesis*, 2010.

[11] S. Idreos. *Big Data Exploration*. Taylor and Francis, 2013.

[12] S. Idreos, I. Alagiannis, R. Johnson, and A. Ailamaki. Here are my Data Files. Here are my Queries. Where are my Results? In *Proceedings of the biennial Conference on Innovative Data Systems Research (CIDR)*, 2011.

[13] S. Idreos, F. Groffen, N. Nes, S. Manegold, S. Mullender, and M. L. Kersten. MonetDB: Two Decades of Research in Column-oriented Database Architectures. *IEEE Data Eng. Bull.*, 35(1):40–45, 2012.

[14] S. Idreos, R. Kaushik, V. R. Narasayya, and R. Ramamurthy. Estimating the compression fraction of an index using sampling. In *Proceedings of the International Conference on Data Endineering (ICDE)*, pages 441–444, 2010.

[15] S. Idreos, M. L. Kersten, and S. Manegold. Database cracking. In *Proceedings of the biennial Conference on Innovative Data Systems Research (CIDR)*, 2007.

[16] S. Idreos, M. L. Kersten, and S. Manegold. Updating a cracked database. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 413–424, 2007.

[17] S. Idreos, M. L. Kersten, and S. Manegold. Self-organizing tuple reconstruction in column stores. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 297–308, 2009.

[18] S. Idreos and E. Liarou. dbTouch: Analytics at your Fingertips. In *Proceedings of the biennial Conference on Innovative Data Systems Research (CIDR)*, 2013.

[19] S. Idreos, E. Liarou, and M. Koubarakis. Continuous multi-way joins over distributed hash tables. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, pages 594–605, 2008.

[20] S. Idreos, S. Manegold, H. Kuno, and G. Graefe. Merging What's Cracked, Cracking What's Merged: Adaptive Indexing in Main-Memory Column-Stores. *Proceedings of the Very Large Data Bases Endowment (PVLDB)*, 4(9):585–597, 2011.

[21] S. Idreos, O. Papaemmanouil, and S. Chaudhuri. Overview of Data Exploration Techniques. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, 2015.

[22] S. Idreos, C. Tryfonopoulos, and M. Koubarakis. Distributed evaluation of continuous equi-join queries over large structured overlay networks. In *Proceedings of the International Conference on Data Endineering (ICDE)*, page 43, 2006.

[23] M. Kersten, S. Idreos, S. Manegold, and E. Liarou. The Researcher's Guide to the Data Deluge: Querying a Scientific Database in Just a Few Seconds. *Proceedings of the Very Large Data Bases Endowment (PVLDB)*, 4(12):1474–1477, 2011.

[24] J. Lee, G. K. Attaluri, R. Barber, N. Chainani, O. Draese, F. Ho, S. Idreos, M. Kim, S. Lightstone, G. M. Lohman, K. Morfonios, K. Murthy, I. Pandis, L. Qiao, V. Raman, V. K. Samy, R. Sidle, K. Stolze, and L. Zhang. Joins on encoded and partitioned data. *Proceedings of the Very Large Data Bases Endowment (PVLDB)*, 7(13):1355–1366.

[25] E. Liarou, R. Goncalves, and S. Idreos. Exploiting the power of relational databases for efficient stream processing. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, pages 323–334, 2009.

[26] E. Liarou and S. Idreos. dbTouch in Action: Database kernels for touch-based data exploration. In *Proceedings of the International Conference on Data Endineering (ICDE)*, pages 1262–1265, 2014.

[27] E. Liarou, S. Idreos, and M. Koubarakis. Evaluating conjunctive triple pattern queries over large structured overlay networks. In *International Semantic Web Conference*, pages 399–413, 2006.

[28] E. Liarou, S. Idreos, and M. Koubarakis. Continuous rdf query processing over dhts. In *ISWC/ASWC*, pages 324–339, 2007.

[29] E. Liarou, S. Idreos, S. Manegold, and M. L. Kersten. Monetdb/datacell: Online analytics in a streaming column-store. *Proceedings of the Very Large Data Bases Endowment (PVLDB)*, 5(12):1910–1913, 2012.

[30] E. Liarou, S. Idreos, S. Manegold, and M. L. Kersten. Enhanced stream processing in a DBMS kernel. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, pages 501–512, 2013.

[31] E. Petraki, S. Idreos, and S. Manegold. Holistic indexing in main-memory column-stores. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, 2015.

[32] H. Pirk, E. Petraki, S. Idreos, S. Manegold, and M. L. Kersten. Database cracking: fancy scan, not poor man's sort! In *Proceedings of the International Workshop on Data Management on New Hardware (DAMON)*, 2014.

[33] C. Tryfonopoulos, S. Idreos, and M. Koubarakis. Publish/subscribe functionality in ir environments using structured overlay networks. In *ACM SIGIR Special Interest Group on Information Retrieval*, pages 322–329, 2005.

[34] C. Tryfonopoulos, S. Idreos, M. Koubarakis, and P. Raftopoulou. Distributed large-scale information filtering. *T. Large-Scale Data- and Knowledge-Centered Systems*, 13:91–122, 2014.

[35] K. Zoumpatianos, S. Idreos, and T. Palpanas. Indexing for interactive exploration of big data series. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 1555–1566, 2014.

Report on the First International Workshop on Personal Data Analytics in the Internet of Things (PDA@IOT 2014)

Vassilis Christophides
R&I Center, Technicolor, France
vassilis.christophides@technicolor.com

Themis Palpanas
Paris Descartes University, France
themis@mi.parisdescartes.fr

1. INTRODUCTION

In the last few years, we are witnessing a wave of connected things (i.e., devices) that are flooding our everyday living spaces. These networks of physical objects with *embedded sensors* and *actuators* that communicate with other objects, databases, and in some cases with people, are often described under the umbrella term *Internet of Things* (IoT). Recent studies[1] estimate that by 2020 the IoT will scale up to 50 billion daily-use objects, while its economic value will reach $14.4 trillion dollars, spread across all sectors (both consumer[2] and industrial[3] markets). According to the IoT vision, "smart things" will "disappear" in our living environments (aka *Pervasive Computing*), or be embodied in humans (aka *Wearable Computing*, with the goal of building disruptive services for human-beings, offering a *seamless* and *implicit interaction* between the *real* and the *virtual* worlds.

In this changing world, every human action and living context will create "information shadows" on the IoT via active, or passive monitoring. For example, *Quantified-self*[4]) devices enable self-tracking of any kind of biological, physical, or behavioral information, *smart home sensors*[5] capture accurate indoor environmental information and energy consumption habits, *residential gateways* record in real-time usage information of communication and entertainment devices, while *smart cars and phones* trace the places and trajectories of our everyday life. Fine-grained personal data are already being massively created, permeating almost every facet of human interaction to objects (*user awareness*), to their physical environment (*ambient awareness*) and to other humans (*social awareness*). Such personal data are mostly analyzed today by application silos tightly coupled with the employed "smart things" (in health and well-being, home automation and entertainment, etc.). Individuals are striving today for tools that can help them to gather, process and make sense of all the data they produce in private, as well as public spaces.

The 1st International Workshop on Personal Data Analytics in the Internet of Things (PDA@IOT), held in conjunction with VLDB 2014, aims at *sparking research on data analytics, shifting the focus from business to consumers services*. While much of the public and academic discourse about personal data has been dominated by a focus on the privacy concerns and the risks they raise to the individual, especially when they are seen as the new oil of the global economy[6], PDA@IOT focus on how persons could effectively exploit the data they massively create in Cyber-Physical worlds. It relies on the following assumptions[7]:

- My personal data is most *valuable to me*!
- People want to *benefit from their data– not hide it away* - but they need control, trust & transparency
- Personal data economy is primarily a *human experience economy*
- Personal data intend to *empower people*

We believe that the full potential of the IoT goes far beyond connecting "things" to the Internet: it is about using data to create new value for people. In a *People-centric computing paradigm*[8], both small scale personal data and large scale aggregated data should be exploited to identify unmet needs and proactively offer them to users. PDA@IOT seeks to address current technology barriers that impede existing personal data processing and analytics solutions to empower people in *personal decision making*[9].

The PDA@IOT ambition is to provide a unique forum for researchers and practitioners that approach personal data from different angles, ranging from data management and processing, to data mining and human-data interaction, as well as to nourish the interdisciplinary synergies required to tackle the

[1] D. Evans "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything", Cisco IBSG 2011.

[2] M. Vallve "The Internet Of Things' Will Change Everything About The Global Consumer Economy", BII May 2014

[3] P. C. Evans, M. Annunziata "Industrial Internet: Pushing the Boundaries of Minds and Machines", GE, Nov. 26, 2012

[4] Z. Yumak, P. Pu "Survey of Sensor-Based Personal Wellness Management Systems" BioNanoScience 3(3) Sept. 2013

[5] X. Zhang, T.Kato, T.Matsuyama Learning a context-aware personal model of appliance usage patterns in smart home. Innovative Smart Grid Technologies(ISGT) - Asia, 2014

[6] A. Pentland et al. "Personal Data: The Emergence of a New Asset Class", World Economic Forum. January 2011.

[7] A. Margolis "A people-centred approach to Data and the Internet of Things", Claro Partners, 2014.

[8] Similar paradigm shifts have been recently proposed in other domains such as *Vendor Relationship Management* (VRM) in contrast to *Customer Relationship Management* (CRM) or the *Intention* in contrast to the *Attention Economy*.

[9] J. Duggan. "The Case for Personal Data-Driven Decision Making". In PVLDB, vol. 7, 2014.

challenges and problems emerging in People-centric Computing. A list of core research issues considered by the first edition of the workshop includes:

- *Trusted architectures* for personal data collection balancing data privacy & utility
- *Interpretation of raw personal data streams* originating from heterogeneous sensors
- Descriptive, predictive & cognizant *data personal analytics*
- *Offline/online* personal data *processing & curation*.

The workshop program included 2 keynote talks, 7 research papers, and a panel discussion. In this report, we summarize the ideas that were presented and discussed in the workshop. The detailed program and the slides for the talks are available at the PDA@IOT web page: http://www2.thlab.net/pda-iot2014/.

2. Keynote Talks

The first keynote talk, "Objective Self", was given by Prof. Ramesh Jain from the University of California at Irvine (USA). Prof. Jain presented how it is now possible to analyze and understand a person's life style in order to build a model for an individual, called the "Objective Self". Most people use phones with a multitude of sensors that continuously generate data streams related to several different aspects of their lives. Other powerful sources of information for a person are social networks, e-mail, calendar systems, and lately wearable and home sensors. By integrating these multi-sensory data streams, it is possible to create an accurate chronicle of a person's life. In this talk, Prof. Jain focused on the context of a person's health: by correlating life events to health related events obtained using wearable sensors and other sources of information, one can build the health persona of a person, which is a long-term objective characterization of a person's health. Several data streams can be used in this respect, such as motion tracking, location tracking, activity level, wearable health sensors, and personal calendar data. Prof. Jain illustrated how recognition algorithms can be applied to the Life Event detection problem and then build an objective personal chronicle, called personicle. By building a personicle over a long period and applying data mining, it is possible to create a model of the person that could result in actionable insights and alerts in everyday life.

The second invited speaker was Prof. Dimitrios Georgakopoulos from the RMIT University (Australia) and his talk was entitled "Distilling High Value Personalized Information from the Internet of Things and Social Networks". He highlighted that Cyber-Physical-Social (CPS) Computing encompasses real-time extraction of high value and personalized information from social networks and millions of cyber-physical systems in the IoT, as well as the development of specialized cloud-based services. Despite the expanding array of applications that require distilling knowledge from big CPS data, there is currently no easy way to manage and exploit such big data, personalize the results, or develop cloud services that make this possible anywhere via mobile devices. Therefore, CPS requires the development of novel solutions for discovering on-line cyber-physical and social media sources, dynamically integrating such sources and their data, and analyzing or personalizing billions of data streams and tens of years of historical data form on-line sources anywhere and in real time. Prof. Georgakopoulos was particularly interested in context-aware techniques proposed by research and commercial systems that can help individuals understand IoT data. Finally, he outlined a unified approach for CPS data management and analysis involving real-time summarization and personalization as the main way of extracting high value information from big data in various domains: digital agriculture, smart energy grids, and disaster management.

3. Research Papers

3.1 Healthcare and Wellbeing

The first session of the workshop was devoted to research papers that addressed *personal decision making in healthcare and wellbeing*.

In the paper entitled "*kHealth: Proactive Personalized Actionable Information for Better Healthcare*", Amit Sheth, Pramod Anantharam, and Krishnaprasad Thirunarayan focused on health aficionados and patients with chronic conditions that increasingly rely on sensors and mobile devices to track sleep, food, physical activity, and other physiological observations (e.g., weight, heart rate, blood pressure). They highlighted a paradigm shift in healthcare from reactive medicine to predictive, preventative, personalized, and participatory medicine, in which individuals are empowered to fully participate in health related decision making. To facilitate this transformation, there is a need to understand the richness and nuances of fine-grained healthcare data produced by a variety of mobile devices and wearable sensors, related to a person and the population in general. The majority of existing data analytics techniques focuses on finding discrepancies in a single stream of observations without much insight into the problem and actionable knowledge to the individuals. The authors presented the kHealth system for analyzing observations from passive (no human involvement in data collection) and active (human input involved in data collection) sensors in order to provide explanations that are intelligible to individuals and their clinicians for well-informed decision making.

In the same direction, Emmanouil G. Spanakis, Feng Dong, Manolis Tsiknakis, Vangelis Sakkalis, Kostas Marias and Dimitris Kafetzopoulos, in their work entitled "*MyHealthAvatar: The digital patient perspective using Internet of Things technologies*", focused on how health-related information in the IoT can be locally aggregated and transmitted for remote monitoring and response. They stressed that the interrelation of such heterogeneous data sources opens new directions for providing a comprehensive picture related to health, thus triggering interventions by medical staff when necessary, and realizing preventive care. They described MyHealthAvatar (MHA), a personal digital health related collection bag, carried by individual citizens throughout their lifetime to sustain in a meaningful manner all information related to their health, empowering users through a number of patient-centred healthcare services. They outlined the MHA challenges for accessing, collecting and sharing long term multilevel personal health data through an integrated environment including: clinical data, genetic data, medical sensor data and devices, human behavior data and activity data for clinical data analysis, prediction and prevention for the individual citizen.

Finally, Xinpeng Zhang, Hiroki Kitabayashi, Yasuhito Asano and Masatoshi Yoshikawa, presented "*A Health-aware Pedestrian Navigation System by Analysis of Spatiotemporal Vital Signs Data*", discussing how health-related data can empower individuals in their outdoor social activities. The authors focused on a health-aware pedestrian navigation system to assist the outdoor activities for vulnerable pedestrians, such as senior people, or people with disabilities. Examples of health related profiling of jogging itineraries include the recognition of when pedestrians get tired while climbing a slope, or if they feel stressed when walking some road. The system offers essentially two services. First, it monitors the personal health condition from individual vital signs data. Since a pedestrian's health condition varies by location and time due to fatigue and stress, it is spatio-temporally contextualized. Second, the system recommends jogging itineraries according to the results of the previous analysis. It can therefore guide multiple pedestrians to walk together, promoting conversation and strengthening mental health.

3.2 Life Logging and Citizen Services

The second session of the workshop focused on extracting *individual and collective behaviors for life logging and citizen services*.

Laleh Jalali, Da Huo, Hyungik Oh, Mingfan Tang, Siripen Pongpaichet and Ramesh Jain, in their work entitled "*Personicle: Personal Chronicle of Life Events*", presented an overview and early results of their project aiming at integrating, aggregating, and analyzing heterogeneous personal data streams to build a persona, and use it to recognize evolving personal situations. In particular, the authors are interested in recognizing movements and personal situations in order to empower individuals with actionable information and insights. They demonstrated how high-level life events can be extracted through simultaneous use of asynchronous observations consisting of continuous GPS and accelerometer measurements. To create a chronicle of life events, called Personicle, they detect low level physical activities using data produced by various unobtrusive sensors embedded in a mobile phone, and use hierarchical classification techniques for identifying high level life events using location context and physical activity. The authors plan to collect, store, and analyze data from a large number of heterogeneous sensors, and make publicly available the test data sets.

Yuzuru Tanaka, Hajime Imura and Jonas Sjabergh, in their work entitled "*Exploratory Visual Analytics for Winter Road Management using Statistically Preprocessed ProbeCar Data*", show how aggregated probe car data can be statistically preprocessed over road links for an urban-scale area, in order to visualize the dynamic change of traffic flow in terms of the divergence and the flow vector field. This analysis provides insights regarding the dynamic change of traffic hotspots, main traffic streams, and route selection preference. This work is motivated by new citizen services, in particular improving winter road management in Sapporo. They extend well-known exploratory visual analytics techniques to multiple coordinated views by integrating different analysis tools with their result visualization views into the same operational framework. These newly added views may coordinate with others, and allow users to directly select patterns calculated at runtime to further quantify the underlying database view. Exploratory visual analytics with such an environment enables us to detect road links for effective, pinpoint snow-removal.

3.3 Wearable Devices

The third session of the workshop program comprised papers *analyzing data from wearable devices* to recognize individuals' emotional arousal, or train their brain's fast and slow thinking abilities.

Julien Fleureau, Philippe Guillotel and Izabela Orlac in their work entitled "Affective Profiles of Movies and Opera Based on the Physiological Responses of the Audience" proposed an objective study of the emotional impact on real audience of various audiovisual shows and live events. An affective benchmarking solution was presented, making use of a low-intrusive measurement of the ElectroDermal

Activity (EDA), which is known to be linked to the unconscious reactions of the nervous system, and thus can be used as a measure of emotional and sympathetic arousal responses. A dedicated processing of this biosignal produces a time-variant and normalized affective profile related to the significant excitation variations of the audience. Besides the methodology, the originality of this work stems from the evaluation of the proposed framework on 62 real audience members during special screenings of a film festival and one opera. The authors show that the resulting "Affective Profile" is strongly correlated to the events occurring during the shows, that some shows have a higher intensity than others, and that the extracted affective profile is consistent with the artistic rules from the creative intent.

Finally, Melanie Swan, Takashi Kido and Minna Ruckenstein in their work entitled "BRAINY: Multi-modal Brain Training App for Google Glass" presented a new Google Glass application for brain training. This work is motivated by the observation that there are over 50 brain fitness training companies, but there is not yet a brain fitness training app for Glass. BRAINY initially targets the improvement of memory function, and additional cognitive training modules can be added later for attention, processing speed, flexibility, problem solving, and other areas. BRAINY takes advantage of the audio-visual and voice command functionality of the Glass platform to create multi-modal memory games that can be played easily by users during waiting time, or other down time. The more advanced memory modules of BRAINY target neuroplasticity and memory updating, and train the brain's fast and slow thinking systems.

4. Panel Discussion

Given that personal data processing and analytics is still in its infancy, the panel attempted to sketch a first roadmap of the emerging core research issues. The moderator, Vassilis Christophides (Technicolor R&I, France), challenged three panelists on the building blocks required to understand and make sense of Personal Data harvested in the IoT for enabling *smarter individual choices* and *behavioral changes,* as well as *resource-conscious automation and optimization of everyday life*: (a) Prof. Ramesh Jain (Univ. of California at Irvine, USA) (b) Prof. Yuzuru Tanaka (Hokkaido Univ., Japan), and (c) Prof. Themis Palpanas (Paris Descartes Univ., France).

Prof. Ramesh Jain emphasized the need for collecting data from various wearable devices, mobile phones, and social networks, in a way that users do not feel threatened for their privacy. Current data sources are very limited, because people do not see a good reason to take the risk and share their data. He emphasized the need for developing simple applications that clearly show that data sharing results in tangible benefits, in order to encourage people to share their data for the long-term benefits to individuals, as well as to the society. Another issue that Prof. Jain raised was a sustainable business model to be used for personal data. Some people have talked about trading personal data for value depending on who wants access to it. This is a fundamental issue that assumes that personal data is a new currency of some kind that could be traded for the appropriate value.

Prof. Yuzuru Tanaka pointed out that urban-scale monitoring of collective human activities is nowadays conducted through private sector sensing (smart-phone, probe-car and card usage data), public sector sensing (traffic jam/accident, public transportation and surveillance-camera image data), and crowd-source sensing (smart-phone application usage and Social Network data). Unfortunately, these data reside in different silos, mainly due to data privacy concerns. Statistical data processing may be a solution. However, what kind of statistical processing may allow what kind of analyses to discover what kind of knowledge about collective human activities is still an open issue. More generally, he argued that there exists a big gap between the state-of-the-art analytical methodologies and the complex big data application requirements, such as optimization of urban-scale infrastructures. In these settings, we need to analyze systems of heterogeneous systems, where each component can be mathematically modeled, but the whole system cannot. To find an appropriate analysis scenario is itself a research task. In this respect, he advocated the need for new machine learning algorithms capable to deal with these challenges, and that, for the moment, a pragmatic alternative seems to be exploratory visual analytics.

Prof. Palpanas stressed the need for non-technical users to be able to easily manage their personal data, as well as perform a range of analysis tasks on them. This would give them insights with respect to their habits and personal choices, and enable them to make informed decisions about their lifestyles. He argued that this goal could be achieved by the development of a new generation of systems and applications having usability at the core of their design. He offered as an example the domain of data series: even though most of the popular personal monitoring devices produce data in the form of time series, there are no suitable systems that can empower non-technical users (or developers) to take full advantage of the wealth of information hidden in these data.

ACKNOWLEDGMENTS

Special thanks go to Technicolor R&I Center (France) for their support in the organization of this workshop.

Sixth ACM Symposium on Cloud Computing

Hilton – Waikoloa, Big Island, HI

Aug 26 – Aug 29, 2015

Call For Papers

The ACM Symposium on Cloud Computing 2015 (ACM SoCC 2015) will be the sixth in a new series of symposia that brings together researchers, developers, users, and practitioners interested in cloud computing. ACM SoCC is the premier conference on cloud computing; it is the only conference co-sponsored by the ACM Special Interest Groups on Management of Data (SIGMOD) and on Operating Systems (SIGOPS). SoCC will be held at the Hilton – Waikoloa in Hawaii. It will be co-located with VLDB 2015.

The scope of SoCC is broad and encompasses diverse data management and systems topics such as software as a service, virtualization, scalable cloud data services, and data management and analytics at scale. Many facets of systems and data management issues must be revisited in the context of cloud computing. Suggested topics for paper submissions include but are not limited to:

- Administration and Manageability
- Data Privacy
- Data Services Architecture
- Distributed and Parallel Query Processing
- Energy Management
- Distributed and Cloud Management
- High Availability and Reliability
- Infrastructure Technologies
- Large Scale Cloud Applications
- Multi-Tenancy
- Programming Models
- Provisioning and Metering
- Resource Management and Performance
- Scientific Data Management
- Security of Services
- Service Level Agreements
- Storage Architectures
- Transactional Models
- Virtualization Technologies

Paper Submission

Authors are invited to submit original papers that are not being considered for publication in any other forum. All SoCC submissions will be held to a high quality standard, and evaluated based on their originality, technical merit, topical relevance, value to the community, and likelihood of leading to insightful technical discussions at the symposium. For additional details, please visit our website at:
https://sites.google.com/site/acm2015socc/home/call-for-papers

Important Dates

Papers Due	April 24, 2015 9 PM PDT
Acceptance	June 19, 2015
Posters Due	June 26, 2015
Camera Ready Due	July 20, 2015

Conference Officers

General Chair	Shahram Ghandeharizadeh
Program Chairs	Magdalena Balazinska Mike Freedman
Steering Committee	Phil Bernstein Mike Carey Joe Hellerstein Marvin Theimer John Wilkes Willy Zwaenepoel

Call for Participation for FCRC 2015
(Federated Computing Research Conference)

Call for Participation
Federated Computing Research Conference (FCRC 2015)
June 13-20, 2015, Portland, Oregon, USA
Web Site: http://fcrc.acm.org

Early bird registration is now open for FCRC 2015, the event that assembles a spectrum of affiliated research conferences and workshops into a week-long coordinated meeting. FCRC 2015 will be held at the Oregon Convention Center, in Portland, Oregon from June 13 – 19, 2015. Early Registration ends May 18th.

This year's plenary speakers include:

- Michael Stonebraker, MIT CSAIL, 2014 ACM Turing Award Winner, who will deliver the 2014 ACM Turing Lecture;
- Andrew Yao, Tsinghua University;
- Olivier Temam, Google;
- Don Syme, Microsoft Research;
- Kathy Yelick, University of California at Berkeley; and
- Balaji Prabhakar, Stanford University.

Conferences include:

- Computational Complexity Conference
- CRA-W 2015 Career Mentoring Workshops
- EC 2015 [The 16th ACM Conference on Economics and Computation]
- HPDC 2015 [The 24th International Symposium on High-Performance Parallel and Distributed Computing]
- ISCA 2015 [The 42nd International Symposium on Computer Architecture]
- ISMM 2015 [ACM SIGPLAN International Symposium on Memory Management]
- IWQoS 2015 [IEEE/ACM International Symposium on Quality of Service]
- LCTES 2015 [ACM SIGPLAN/SIGBED International Conference on Languages, Compilers and Tools for Embedded Systems]
- PADL 2015 [17th Symposium on Practical Aspects of Declarative Languages]
- PLDI 2015 [36th ACM SIGPLAN Conference on Programming Language Design and Implementation]
- SIGMETRICS 2015 [International Conference on Measurement and Modeling of Computer Systems]
- SPAA 2015 [ACM Symposium on Parallelism in Algorithms and Architectures]
- STOC 2015 [47th ACM Symposium on Theory of Computing]
- TRANSACT [10th ACM SIGPLAN Workshop on Transactional Computing]

For more details, please visit the website:

http://fcrc.acm.org