SIGMOD Officers, Committees, and Awardees

Chair	Vice-Chair	Secretary/Treasurer
Yannis Ioannidis	Christian S. Jensen	Alexandros Labrinidis
University of Athens	Department of Computer Science	Department of Computer Science
Department of Informatics	Aarhus University	University of Pittsburgh
Panepistimioupolis, Informatics Bldg	Åbogade 34	Pittsburgh, PA 15260-9161
157 84 Ilissia, Athens	DK-8200 Århus N	PA 15260-9161
HELLAS	DENMARK	USA
+30 210 727 5224	+45 99 40 89 00	+1 412 624 8843
<yannis AT di.uoa.gr>	<csj AT cs.aau.dk >	<labrinid AT cs.pitt.edu>

SIGMOD Executive Committee:
Sihem Amer-Yahia, Curtis Dyreson, Christian S. Jensen, Yannis Ioannidis, Alexandros Labrinidis, Maurizio Lenzerini, Ioana Manolescu, Lisa Singh, Raghu Ramakrishnan, and Jeffrey Xu Yu.

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

Information Director, SIGMOD DiSC and SIGMOD Anthology Editor:
Curtis Dyreson, Washington State University, <cdyreson AT eecs.wsu.edu>

Associate Information Directors:
Denilson Barbosa, Ugur Cetintemel, Manfred Jeusfeld, Georgia Koutrika, Alexandros Labrinidis, Michael Ley, Wim Martens, Rachel Pottinger, Altigran Soares da Silva, and Jun Yang.

SIGMOD Record Editor:
Ioana Manolescu, INRIA Saclay, <ioana.manolescu AT inria.fr>

SIGMOD Record Associate Editors:
Magdalena Balazinska, Denilson Barbosa, Pablo Barceló, Vanessa Braganholo, Chee Yong Chan, Ugur Çetintemel, Brian Cooper, Cesar Galindo-Legaria, Glenn Paulley and Marianne Winslett.

SIGMOD Conference Coordinator:
Sihem Amer-Yahia, Qatar Computing Research Institute, <sihemameryahia AT acm.org>

PODS Executive: Maurizio Lenzerini (Chair), University of Roma 1, <lenzerini AT dis.uniroma1.it>, Phokion G. Kolaitis, Jan Paradaens, Thomas Schwentick, Jianwen Su and Dirk Van Gucht.

Sister Society Liaisons:
Raghu Ramakhrishnan (SIGKDD), Yannis Ioannidis (EDBT Endowment).

Awards Committee:
Laura Haas (Chair), IBM Almaden Research Center, <laura AT almaden.ibm.com>, Rakesh Agrawal, Peter Buneman, and Masaru Kitsuregawa.

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

[Last updated : October 10th, 2011]

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. Until 2003, this award was known as the "SIGMOD Innovations Award." In 2004, SIGMOD, with the unanimous approval of ACM Council, decided to rename the award to honor 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)	

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)	

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.* This award, which was previously known as the SIGMOD Doctoral Dissertation Award, was renamed in 2008 with the unanimous approval of ACM Council in honor of Dr. Jim Gray. Recipients of the award are the following:

- **2006** *Winner*: Gerome Miklau, University of Washington. *Runners-up*: Marcelo Arenas, University of Toronto; Yanlei Diao, University of California at Berkeley.
- **2007** *Winner*: Boon Thau Loo, University of California at Berkeley. *Honorable Mentions*: Xifeng Yan, University of Indiana at Urbana Champaign; Martin Theobald, Saarland University
- **2008** *Winner*: Ariel Fuxman, University of Toronto. *Honorable Mentions*: Cong Yu, University of Michigan; Nilesh Dalvi, University of Washington.
- **2009** *Winner*: Daniel Abadi, MIT. *Honorable Mentions*: Bee-Chung Chen, University of Wisconsin at Madison; Ashwin Machanavajjhala, Cornell University.
- **2010** *Winner:* Christopher Ré, University of Washington. *Honorable Mentions*: Soumyadeb Mitra, University of Illinois, Urbana-Champaign; Fabian Suchanek, Max-Planck Institute for Informatics.
- **2011** *Winner*: Stratos Idreos, Centrum Wiskunde & Informatica. *Honorable Mentions*: Todd Green, University of Pennsylvania; Karl Schnaitter, University of California in Santa Cruz.

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

[Last updated : October 10th, 2011]

Editor's Notes

Welcome to the December 2011 issue of the ACM SIGMOD Record! This issue appearing in December 2011 puts us back on the regular publication schedule.

I am glad to start this issue's notes by congratulating the members of our scientific community which have been recently named ACM Fellows (http://www.acm.org/press-room/news-releases/2011/fellows-2011): Serge Abiteboul, Divyakant Agrawal, Christian S. Jensen, Beng Chin Ooi, Margo Seltzer, Divesh Srivastava, Dan Suciu and Meral Özsoyoglu. Thanks to them all for representing our community within the larger computer science family, and many congratulations!

The first article by Lee, Lee and Park studies index scan operations on Flash Memory solid state drives (or SSDs, in short). The authors investigate the relation between the selectivity of an index scan and the performance that the operation can attain on an SSD. Index scans turn out to be inefficient for very selective look-ups, therefore the authors consider as an alternative a sorted index scan, which first sorts the index entries by record ID. However, the sorted index scan loses the index key order. To mitigate this, the authors propose a new index-based sort algorithm, which can sort the data in one pass regardless of the available sort memory size.

The survey by Lee, Lee, Choi, Chung and Moon is a very timely one: it provides an overview of parallel data processing techniques based on the principles of MapReduce. This is a field in which the state of the art moves faster than most of us manage to follow! The authors start by describing a generic MapReduce-based architecture, discuss advantages and pitfalls, before moving on to describe variants and improvements. Important extensions such as higher-level languages, flexible data flows, schema support, scheduling, I/O optimizations and others are discussed. An interesting list of applications is also presented, making the survey a valuable collection of information and pointers.

The system paper by Marcus, Bernstein, Badar, Karger, Madden and Miller is a nice example of how database technologies and a database-oriented mind set can help make the most out of data sources which exist today, but would have seemed outlandish in the days when the first bricks of our discipline were laid out! The authors devise a model (akin to data streams), a query language and elements of interface with a system that queries and integrates data from microblogs such as Twitter streams. TweeQL, the language they propose, mixes stream query features with built-in UDFs e.g. for text processing, geo-codes, classification and more.

Jiawei Han is this issue's guest in the Distinguished Profiles in Databases. The processing of Jiawei's interview transcript, through the meanders of the Record's editorial process (Marianne's assistant, Donna Coleman, does a great job at transcribing the audios gathered by Marianne, before Vanessa makes a second beautification pass), can be measured in time by the evolution of Jiawei's H-index: 76 at the time of the interview, it is 101 by the time of the publication! We are glad to send this issue to print before this number becomes obsolete. Read Jiawei's interview for many unique stories and insights: how he was admitted to an US university long before Chinese students could take GREs, where the border lies between data mining and databases and how to organize a very active research and supervisor career.

In the Research Centers column, Kersten, Manegold and Mullender summarize the twenty-six years of history of the CWI database research group from Amsterdam. Longtime a stalwart of the database community, CWI has recently enjoyed significant visibility in our community, such as, for instance, the 2011 SIGMOD Jim Gray Doctoral Dissertation Award to Stratos Idreos on database cracking. An important part of the CWI recent successes can be attributed to their longstanding work around MonetDB.

The paper surveys other areas of the group's work, including hardware-conscious data management, scientific databases, graphs and stream processing.

The Industry Perspective column is present with a report on HANA, SAP's new database, by Färber, Cha, Primsch, Bornhövd, Sigg and Lehner. HANA is a memory-centric database, built to leverage the capability of modern hardware (such as multi-core processors and SSDs) for the benefit of analytical and transactional applications. Interestingly, HANA supports both structured and semi-structured data, as well as entity extraction, or graph processing – clearly not the seventies' database! HANA is made to fit within the complex SAP Business Intelligence product suite. The authors end by discussing various levels of evolution and integration of HANA within the suite, perspectives for their work.

Three workshop reports are part of this issue.
First, Maurino, Cappiello, Vassiliadis and Sattler outline the proceedings of the 8th International Workshop on Quality in Databases (QDB 2010). The workshop focused on issues such as data quality assessment frameworks, data privacy and visualization, as well as the applications of data quality analysis to new domains such as the LOD (Linked Open Data) movement.
The report by Bizer, Boncz, Brodie and Erling results from discussions held at 2011 STI Semantic Summit within a group of twenty-five researchers, on the meaningful use of Big Data. The four authors each put forward a challenge: multi-disciplinary Big Data integration, the Billion Triple Challenge from the Semantic Web community, getting value out of government linked open data, and integrating linked data in regular DBMSs.
Last but not least, the report on the 4th Workshop on Very Large Digital Libraries, by Candela, Manghi and Ioannidis, highlights recent works at the confluence of very large digital libraries, and very large data archives. The workshop proceedings consider topics such as scalable support for digital libraries, archiving scientific data and data archive federations.

The call for papers of the 5th International Workshop on Testing Database System closes the issue.
On behalf of the Record's editorial board, let me wish you pleasant holidays and a Happy New Year!

Your contributions to the Record are welcome via the RECESS submission site (http://db.cs.pitt.edu/recess). Prior to submitting, be sure to peruse the Editorial Policy on the SIGMOD Record's Web site (http://www.sigmod.org/publications/sigmod-record/sigmod-record-editorial-policy).

Ioana Manolescu

December 2011

Past SIGMOD Record Editors:

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

Optimizing Index Scans on Flash Memory SSDs

Eun-Mi Lee[†] Sang-Won Lee[†] Sangwon Park[‡]

[†]School of Information & Communications Engineering
Sungkyunkwan University
Suwon, 440-746, Korea
{bbhammer,swlee}@skku.edu

[‡]Department of Information Communication Engineering
Hankook University of Foreign Studies
Yongin, 449-791, Korea
{swpark}@hufs.ac.kr

ABSTRACT

Unlike harddisks, flash memory SSDs have very fast latency in random reads and thus the relative bandwidth gap between sequential and random read is quite small, though not negligible. For this reason, it has been believed that index scan would become more attractive access method in flash memory storage devices. In reality, however, the existing index scan can outperform the full table scan only in very selective predicates.

In this paper, we investigate how to optimize the index scan on flash memory SSDs. First, we empirically show that the index scan underperforms the full table scan even when the selectivity of selection predicate is less than 5% and explain its reason. Second, we revisit the idea of sorted index scan and demonstrate that it can outperform the full table scan even when the selectivity is larger than 30%. However, one drawback of the sorted index scan is that it loses the sortedness of the retrieved records. Third, in order to efficiently resort the result from the sorted index scan, we propose a new external index-based sort algorithm, *partitioned sort*, which exploits the information of key value distribution in the index leaf nodes. It can sort data in one pass regardless of the available sort memory size.

Keywords-flash memory SSDs; sorted index scan; partitioned sort

1. INTRODUCTION

Magnetic harddisks have dominated the storage market for more than three decades, and most enterprise database systems assume harddisks as secondary storage. By the way, harddisk has the inevitable mechanical latency, and the access time for a small data page (e.g. 4KB) is thus dominated by the latency. In contrast, the bandwidth of sequential access in contemporary enterprise class harddisks is enormous (e.g. 200MB per second). From the historical perspective, we have witnessed that the improvement of the access latency in harddisk is lagging far behind that of the sequential access, and thus the performance gap between random access and sequential access is widening [3]. This bandwidth imbalance between random and sequential accesses in harddisk had made query optimizers prefer the full table scan to the index scan, except only when the selectivity of the given selection predicate is very low (e.g. 1%).

Meanwhile, flash memory based solid stated drives (hereafter, SSDs) are becoming popular as alternative storage to harddisks. In contrast to harddisk, SSD has very fast latency in random IOs since it has no mechanical part, and thus the performance gap between sequential and random access to all the data pages in a table is very small, though not negligible. Therefore, we can anticipate that, on top of SSDs, the index scan can outperform the full table scan even when the ratio of randomly accessed data pages in a table is very high (e.g. more than 50%). In reality, however, the existing index scan can outperform the full table scan only in very selective predicates.

In this paper, we investigate how to optimize the index scan on flash memory SSDs. First, we empirically demonstrate that the index scan underperforms the full table scan even when the selectivity of selection predicate is less than 10%, and explain that it is mainly due to repetitive reads of same data pages during the index scan with limited size of buffer cache. Second, we revisit the idea of sorted index scan which first sorts the index entries in the order of record identifier before fetching data pages and thus can avoid the repetitive reads of the same pages. Our experimental result shows that it can outperform the full table scan when the selectivity is larger than 30%. However, one drawback of the sorted index scan is that the retrieved records is not sorted any more in the index key order. Third, in order to efficiently re-sort the result from the sorted index scan, we propose a new index-based sort algorithm, *partitioned sort*, which exploits the information of key value distribution in the index leaf nodes while partitioning the data to be sorted. It can sort data in one pass regardless of the available sort memory size. In contrast, the traditional external sort might require multiple passes depending on the data size and the sort memory size.

2. BACKGROUND

In this section, we compare the performance characteristics of harddisks and SSDs according to the access pattern to data, then explain the parallelism adopted by contemporary SSDs and its implications on the performance of access methods, and briefly review two popular access methods, the full table scan and the index scan.

2.1 Harddisks and SSDs

Table 1: Harddisk vs. Flash SSD

Media	Harddisk[†]	Flash SSD[‡]
Latency (4KB)	3.5ms	0.2ms
IOPS (4KB)	600	35,000
Sequential bandwidth	125MB/sec	200MB/sec
Ratio between random and sequential bandwidth	0.02	0.7

[†]Seagate Cheetah 15K.5 ST373455SS
[‡]A Commercial SSD (SLC flash chip)

Table 1 presents the key performance metrics of a harddisk and an SSD used in our experiment. Because this paper focuses on read-intensive queries, Table 1 contains only the read-related performance metrics. The first row represents the access latency (i.e. response time) when a single process reads a page of 4KB. The second row shows maximum IOPS (IO per second) when concurrent read requests are made to each storage media. In other words, the second row represents the random read bandwidth of each storage media. Please note that throughput is not inverse of response time because each product has the native command queuing (NCQ) feature which can handle multiple requests at a time [3]. To be concrete, harddisks use the well-known elevator algorithm while SSDs distribute multiple requests to different flash chips so that each flash chip can handle each request in parallel. The third row represents the maximum bandwidth of each product when we read data in a purely sequential way. The fourth row shows the ratio between random and sequential bandwidth in each product. The ratio is calculated using the formula, *(maximum IOPS x 4KB) / sequential bandwidth*. Please note that while there is huge imbalance between sequential and random access in harddisk, the relative bandwidth gap between sequential and random read is quite small, though not negligible.

2.2 Parallelism inside SSDs

In order to achieve high bandwidth and better IOPS, every modern SSD adopts multi-channel and multi-way architecture and flash memory controller can read and write flash chips in parallel [1]. From the perspective of query throughput, we need to understand the impact of this parallelism inside SSDs. For simplicity, let us assume that the database tables are uniformly striped across multiple flash memory chips, and also that the data pages from a table are also evenly distributed over the chips. In fact, this assumption about the data distribution across multiple flash chips is similar to the RAID-0 striping. When a single query is accessing a table using the full table scan, all the flash chips would become active because each flash chip contains some data pages from the table, thus maximizing the bandwidth. In contrast, when a single query is accessing the table using the index scan which is implemented using *synchronous IO* call in most DBMS, only one of the flash chips is active while the others are idle at a point of time. Namely, the parallelism inside SSDs is under-utilized.

In order to fairly compare the performance of the full table scan and the index scan in SSDs, we need to run multiple queries at least as many as the parallelism degree inside SSDs, that is, the number of channels. We will illustrate the effect of parallelism in SSDs on query processing in Section 3.2.

2.3 Full Table Scan and Index Scan

Most DBMS provides two primitive access methods to tables, the full table scan and the index scan (hereafter, FTS and IDX, respectively) [5]. When harddisk is used as the database storage, the database query optimizer prefers FTS to IDX as the access method, except when the selection predicate for the target table is very selective. This is mainly due to the slow random IO latency, and IDX would, taking into account the widening performance gap between the sequential and random access, remain as the access method only when accessing a few records. In contrast, SSDs do not have any mechanical component and thus the access time in SSDs is nearly proportional to the data size being accessed. Therefore, an index based access method is expected to be promising when SSDs are used as database storage.

Despite using SSDs as database storage, however, there are two issues which might limit the efficacy of the traditional IDX: *index clusteredness* and *buffer size*. According to the clusteredness, there are clustered index and non-clustered index. If the physical ordering of data records of a table is the same or very close to the ordering of data entries in an index created on the table, the index is said to be clustered; otherwise non-clustered [5]. In case of clustered index, IDX would be very efficient because every data pages is read into buffer only once during index scan. In case of non-clustered index, IDX might read the same data pages from the storage into buffer cache several times because the size of buffer cache is limited and the data page can contain several data records to be retrieved by IDX. In fact, as we will illustrate in Section 3.2, IDX underperforms FTS even when the predicate selectivity is below 5% on SSDs.

3. SORTED INDEX SCAN

3.1 Sorted Index Scan

To improve the efficiency of the non-clustered index scan, there have been some approaches to sort the record identifiers (*rid*) in index entries in the order of page id before accessing the data records using the rids [2, 7]. By taking the *sorted index scan* with a non-clustered index, each data page is fetched once from the storage into buffer even with the limited buffer size. In harddisk based databases, however, the sorted index scan (hereafter, SIDX) has not been such popular and will become less effective because of the ever widening imbalance between sequential and random access. In case of SSDs, however, SIDX could be much more effective. For example, as will be shown later, SIDX outperforms FTS in the selectivity of 40% while pure IDX outperforms FTS in the selectivity of 5%. Therefore, we can make use of non-clustered indexes in a wider range of predicate selectivity.

The commercial database server used in our experiment does not directly support SIDX as its access method, and thus, in order to simulate the logical steps in SIDX using the DBMS, we used a query transformation approach. The two SQL queries in Figure 1 illustrate our query transformation approach using a sample range query which retrieves and aggregates tuples from table tab, each of whose a column value is between min and max. In the example, we assume that a non-clustered index exists on column a in table tab.

```
/* Before transformation */
SELECT *
FROM   tab
WHERE  a BETWEEN min AND max;

/* After transformation */
SELECT *
FROM  (SELECT /* use_nested_loop */ t1.*
       FROM (SELECT /* use_index */ rowid
             FROM   tab
             WHERE  a BETWEEN min AND max
             ORDER BY rowid) t1, tab t2
       WHERE t1.rowid = t2.rowid );
```

Figure 1: Query transformation for simulating the sorted index scan

The innermost SELECT statement in the transformed query finds all the index entries satisfying the predicate a BETWEEN min AND max, sorts them, and returns the result as a virtual table to the next outer query. The second inner SELECT statement accesses all the relevant data pages in table tab in the order of their page_ids, and returns the records satisfying the given predicate. And the outermost statement aggregates b values of all the records. In fact, this approach of query transformation will have run-time overhead over the natively implemented SIDX, but the overhead is not such big enough to change the main argument made in this paper.

3.2 Performance Evaluation

For the experiment, we used a commercial DBMS on Linux 2.6.18 with AMD 3.0GHz hexa-core processor and 8GB RAM. As data tablespace storage, we used the harddisk and the SSD from Table 1. In order to hide the interference on the data tablespace by the IOs in temporary tablespace, another harddisk was designated as the temporary tablespace device. We set database block size, buffer cache, and sort memory to 8KB, 256MB, and 1MB, respectively. The sort memory represents the memory area designated to each process for sorting. When either harddisk or SSD was used as stable storage, it was bound as a raw device to minimize the interference from data caching by the file system.

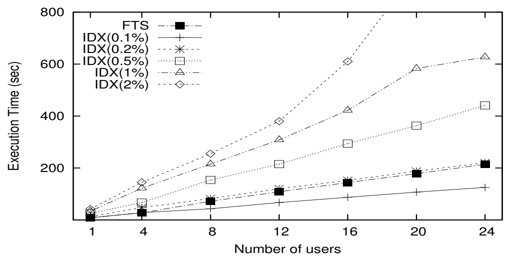
The sample table has 2.5 million tuples, and each tuple is 300B long. In order to create a non-clustered index on the table, we assigned a randomly generated distinct integer value between 1 and 2,500,000 to column a of each tuple, and created an index on column a.

For the sample range query in Figure 1 against the sample table and the non-clustered index, we measured the query execution time of three access methods, FTS, IDX, and SIDX, using harddisk and SSD, respectively, by varying the query selectivity and the number of concurrent users. When varying the number of concurrent users, we made 24 copies of both the sample table and the non-clustered index on column a and allowed each session to run the same sample query in Figure 1 against its own separate table and index. And, the performance result is plotted in Figure 2.

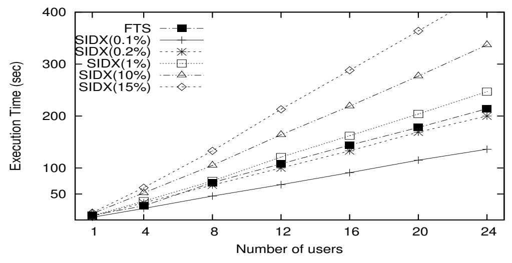
First, Figure 2(a) compares the performance of FTS and IDX in harddisk. As expected, FTS always outperforms IDX, except when the query predicate is very selective (less than 0.2%). This is mainly because the number of random reads increases as the query predicate becomes less selective and because the performance gap between sequential and random read operations in harddisk, as shown in Table 1, is large. Next, Figure 2(b) compares the performance of SIDX and FTS in harddisk. By comparing the performance of IDX and SIDX in Figure 2(a) and Figure 2(b), respectively, we know that SIDX improves IDX considerably. And this is mainly because SIDX reads each page to be accessed by queries only once while IDX might read each page several times. Despite the performance benefit of SIDX, we should note that SIDX, in case of harddisks, underperforms FTS except for very selective case (e.g. less than 1%) at which 6% of total pages in the table is accessed.

Now, let us compare the performance of three access methods on SSD. Figure 2(c) compares the performance of FTS and IDX on SSD. From Figure 2(c), we know that the break-even point between FTS and IDX in SSD moves to the selectivity of 5% at which 17% of total blocks in the table is accessed. Nevertheless, it is still disappointing to observe that the repetitive reads of the same pages make IDX to underperform FTS in SSD even when the selectivity is below 10%. In contrast, the performance of SIDX on SSD can outperform FTS for a wider range of selectivity. As plotted in Figure 2(d), the break-even point between SIDX and FTS moves up to the 40% selectivity when sixteen concurrent queries are running. In our sample table and non-clustered index, the number of distinct pages to be accessed by SIDX is about 60% of total page number in the table.

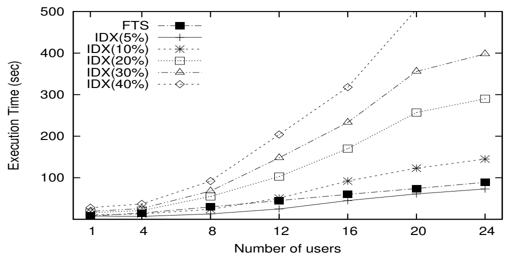
One interesting observation in Figure 2(d) is that the break-even selectivity between FTS and SIDX goes higher as the number of concurrent users increases. In order to understand this performance phenomenon in Figure 2(d), we need to remind the effect of parallelism inside SSD already explained in Section 2.2. When a single query runs in SIDX, SSD is under-utilized since only one flash chip is active at any point of time due to the synchronous IO call in SIDX. In contrast, in case of FTS, even if a single query is running, almost flash chips would be active because of the sequential access pattern in FTS. Therefore, when a single query runs, SIDX could underperform FTS even at the selectivity of 5%. Meanwhile, as more concurrent queries are running in SIDX, more flash chips inside SSD would be active because multiple random read operations are issued by concurrent queries, and thus the query throughput of SSDs gets higher. In contrast, in case of FTS, because of its sequential access pattern, the query execution time would increase in proportion to the number of concurrent queries. Thus, SIDX outperforms FTS at the selectivity of 40% when sixteen or more concurrent queries are running, at which case all flash chips inside SSD would be active in SIDX.



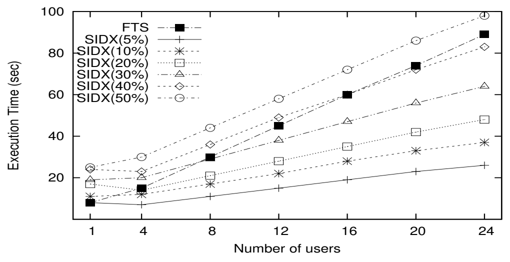
(a) Index Scan on HDD



(b) Sorted Index Scan on HDD



(c) Index Scan on SSD



(d) Sorted Index Scan on SSD

Figure 2: The break-even point between FTS, IDX and SIDX

4. INDEX-BASED PARTITIONED SORT

In the previous section, we have shown that SIDX is an effective access method on SSD. One downside of SIDX is, however, that the tuples produced by SIDX is not any more in the order of index key. Because the *interesting order* is not preserved [6], the output tuples from SIDX should be re-sorted to be used for `order by` or `group by` clause or as an input to the sort merge join.

In this case, the sorting overhead would be non-trivial, although it does not diminish the benefit of SIDX as an access method. But, fortunately we can efficiently sort the output of SIDX by exploiting the index itself. For this, we propose a new index-based sort algorithm, *partitioned sort (PS)* in this section.

4.1 Partitioned Sort

The key idea of PS is to map the records evenly into small partitions based on the ranges of values of the partitioning key so that all the records in each partition could fit into the available sort memory. Because each partition can fit in the sort memory, we can sort each partition using in-memory sort algorithm after reading it from SSD. And because all the partitions are range partitioned, we can simply concatenate the sorted output of each partition without any separate merge step such as in the existing external merge sort.

Then, the remaining key question is how to calculate the partitioning key values. In SIDX, we can calculate them just by scanning the relevant index entries in the leaf nodes because the index entries in the B+-tree leaf nodes are already sorted in key order. Meanwhile, without index on the key attribute, the only way to calculate the partitioning key values is, as far as we can imagine, to sort the whole records.

Below is listed the algorithm of our partitioned sort, and its overall process is depicted in Figure 3.

1. While scanning the index entries in SIDX, we calculate the partitioning key values so that the whole records of each partition can fit into the available sort memory. At the end of this step, we know how many partitions are necessary and the value range of each partition.
2. We create the partitions as necessary and specify the value range of each partition.
3. While retrieving the records using SIDX, we insert each record to its corresponding partition (*partitioning phase*).
4. For each partition (in the ascending order of partition range), we read all the records into the sort memory, sort them, and output the sorted records. (*sorting phase*).

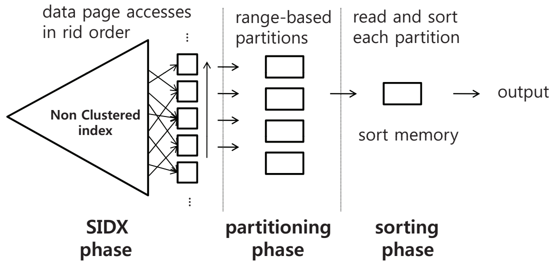


Figure 3: SIDX and partitioned sort

Please note that the step 3 writes all the records once and the step 4 reads all the records once, and thus our PS can, *regardless of the size of the available sort memory*, sort the retrieved records in *one pass* (i.e. one write and one read). Also note that this one-pass sort is possible because the partitions generated in step 3 is *skewless*, that is, the size of each partition is uniform. In comparison, the existing external merge sort [5] may require multiple passes depending on the size of the sort memory, especially when the sort memory is small compared to the size of records to be sorted.

Now let us illustrate the IO pattern generated from our PS with SIDX, and compare it with the IO pattern from the traditional external sorting (ES) with the full table scan (FTS). By the way, the commercial DBMS used in our experiment does not support either SIDX or PS, we again take the query transformation approach to simulate SIDX and PS, as shown in Figure 4. The first `CREATE` statement in Figure 4 creates range-based partitions and inserts the retrieved records using the transformed `SELECT` statement simulating SIDX in Figure 1. In other words, it represents the partitioning phase of our PS algorithm while retrieving the records using SIDX. The second `SELECT` statement in Figure 4 corresponds to the sorting phase of our PS algorithm. That is, it reads each partition, sorts its records using the `ORDER BY` clause, and then merges all the sorted records using the `UNION ALL` clause.

```
/* Before transformation */
SELECT *
FROM   tab
WHERE  a BETWEEN min AND max
ORDER BY a;

/* After transformation */
CREATE TABLE partioned PARTITION BY RANGE (a) (
  PARTITION p_1 VALUES LESS THAN (val1),
            ...
  PARTITION p_n VALUES LESS THAN (maxvalue)
) AS
SELECT statement in Figure 2;

SELECT * FROM ( /*+ use_no_merge */
  SELECT * FROM part_tab PARTITION (p_1) ORDER BY a
  UNION ALL
            ...
  SELECT * FROM part_tab PARTITION (p_n) ORDER BY a)
```

Figure 4: Query transformation for simulating partitioned sort

In order to validate that the query transformation in Figure 4 works as we intended, we traced the IO pattern while executing the transformed query and plotted them in Figure 5(a) [4]. The selectivity of the query was 30% and the size of the sort memory was set to 2 MB. A clear separation of two phases was observed in Figure 5(a). When partitions are created and populated during the partitioning phase, the data pages for the partitions were written sequentially to the data tablespace. In the second sorting phase, on the other hand, the data pages of each partition were read in randomly scattered manner, leading to random reads spread over the whole region of the time-address space corresponding to each partition. Figure 5(a) confirms that the PS algorithm, as we intended, can sort the records in one pass. To better explain the difference between PS and ES, we also traced the IO operations in the temporary

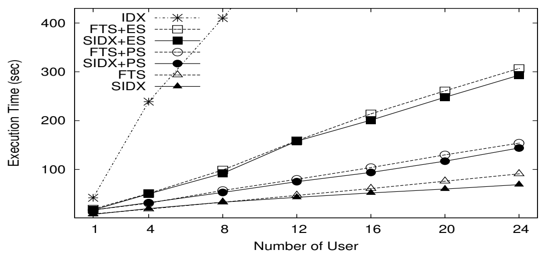


Figure 6: Performance of various query plans

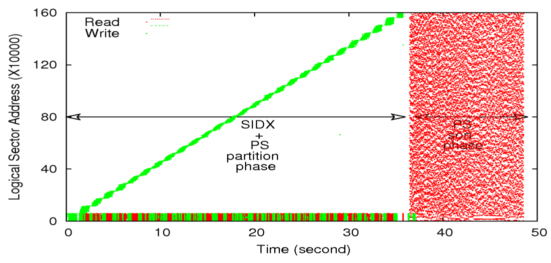
tablespace while executing the original `ORDER BY` query before the transformation in Figure 4 and plotted them in Figure 5(b). The table was accessed in FTS and the selected records were sorted using ES provided by the commercial DBMS. Figure 5(b) also has two phases. The first phase generates the initial runs while scanning the table from the data tablespace, and the second phase merges the runs to generate the final sorted output. In the merge phase, multiple passes are required to merge the runs, and this is because the sort memory is too small to merge all the runs at once. From this, we can observe that our PS can sort the data just in one pass regardless of the size of the available sort memory while ES may require multiple passes depending on the size of sort memory and the data size to be sorted.

The IO pattern of PS consists of sequential writes and random reads, which is preferable in SSD [4], and thus PS can provide good performance. In case of harddisk, however, as we confirmed throughout a separate set of experiments using the harddisk, the small random reads in the sort phase makes PS underperform ES.

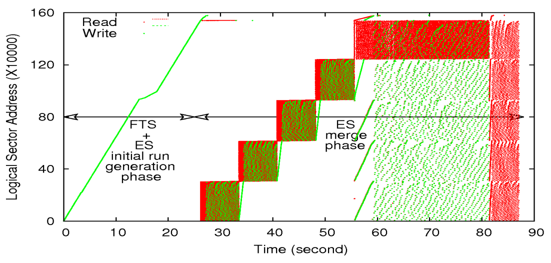
One interesting point is that our index-based partitioned sort can be used in combination with FTS as well as with SIDX. Thus, when the selectivity is higher (e.g. larger than 40%) and an index is available on the sorting attribute(s), PS combined with FTS (FTS + PS) can outperform the traditional ES combined with FTS (FTS + ES).

4.2 Performance Evaluation

The experimental setting here is same as that in Section 3.2. In order to compare the performance of various query execution plans for the `ORDER BY` query in Figure 4, we ran the query in five different ways, including IDX, SIDX+PS, SIDX+ES, FTS+PS and FTS+ES. The idea of PS can be, as stated earlier, applied to FTS as well as SIDX. In this experiment, the query selectivity was set to 30%, the size of the sort memory was 32 MB, the buffer caches was 128MB, and the number of concurrent users was varied from 1 to 24. The performance result is presented in Figure 6. In Figure 6, we also included the performance of SIDX and FTS for the query without `order by` clause so as to easily estimate the sort overhead in each query plan. From Figure 6, we can make a few important observations. First, although IDX preserves the sortedness of the result records and thus the separate sorting phase is not required, it performs much worse than the other four plans. Its overhead from the repetitive reads for the same pages is too



(a) SIDX+PS



(b) FTS+ES

Figure 5: IO patterns of PS and ES

large to be compensated by avoiding the explicit sorting step for the `order by` clause. The second observation is that our PS algorithms (SIDX+PS and FTS+PS) outperform the existing ES algorithms (SIDX+ES and FTS+ES). In particular, taking only the sort time into account, it can be roughly said that PS can sort the records two or three times faster than ES.

In order to test whether PS algorithm is effective in harddisk, we carried out a separate set of experiments using harddisk, but FTS + ES outperforms both SIDX + PS and FTS + PS consistently over almost every experimental configuration except only when the selectivity is lower than than 1%. This is because the overhead of frequent mechanical harddisk head movements for the random writes to multiple partitions during the partitioning phase in PS algorithm can not be compensated by the one pass sort.

Table 2: The effect of sort memory size

Access	2MB	4MB	8MB	16MB	32MB
FTS+ES	340	274	269	269	263
SIDX+PS	122	124	119	122	121
Ratio	2.8	2.2	2.3	2.2	2.2

Now, let us explain the effect of the sort memory size on the performance of two query plans, SIDX+PS and FTS+ES. We ran the transformed query in Figure 4 by varying the size of the sort memory from 2, 4, 8, 16 to 32 MB with the selectivity of 30% with 20 concurrent users. For comparison, we also ran the query in FTS+ES mode with the same configuration. The performance result is presented in Table 2. As expected, while the performance of FTS+ES varies considerably depending on the sort memory size, the performance of SIDX+PS is almost consistent regardless of the sort memory size.

5. CONCLUSION

In this paper, we revisited SIDX (sorted index scan) as a database access method when SSDs are used as the database storage, and have empirically shown that the break-even point between SIDX and FTS (full table scan) moves to higher selectivity (e.g. 40%) especially when multiple queries are concurrently running and thus the intrinsic parallelism inside SSDs are exploited. And, in order to efficiently re-sort the unsorted result of SIDX, we proposed an index-based partitioned sort algorithm, PS, and have experimentally shown that it can outperform the existing external merge sort (ES) and in particular its performance is, in contrast to ES, not sensitive to the size of available sort memory.

In this paper, we simulated SIDX and PS by modifying queries in a commercial DBMS because it does not support those operators directly. In fact, as far as we know, any commercial or open source DBMS is not equipped with either algorithm. However, considering that both SIDX and PS are effective in SSDs and that SSDs would be more actively adopted in the database market, both algorithms need to be incorporated into future DBMSs as first-class citizens; they should be implemented as built-in operators and could be chosen by query optimizers in a cost-based manner.

ACKNOWLEDGMENTS

This work was supported by ITRC NIPA-2011-(C1090-1121-0008) and NRF Grant NRF-2011-0026492.

6. REFERENCES

[1] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. Manasse, and R. Panigrahy. Design Tradeoffs for SSD Performance. In *USENIX 2008 Annual Technical Conference*, 2008.

[2] J. M. Cheng, D. J. Haderle, R. Hedges, B. R. Iyer, T. Messinger, C. Mohan, and Y. Wang. An Efficient Hybrid Join Algorithm: A DB2 Prototype. In *Proceedings of ICDE*, pages 171–180, 1991.

[3] S.-W. Lee, B. Moon, and C. Park. Advances in Flash Memory SSD Technology for Enterprise Database Applications. In *Proceedings of SIGMOD*, pages 863–870, 2009.

[4] S.-W. Lee, B. Moon, C. Park, J.-M. Kim, and S.-W. Kim. A Case for Flash Memory SSD in Enterprise Database Applications. In *Proceedings of SIGMOD*, pages 1075–1086, 2008.

[5] R. Ramakrishnan and J. Gehrke. *Database Management Systems(3rd ed.)*. McGraw Hill, 2002.

[6] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. Access Path Selection in a Relational Database Management System. In *Proceedings of SIGMOD*, pages 23–34, 1979.

[7] P. Valduriez. Join Indices. In *ACM Transactions on Database Systems*, pages 218–246, 1987.

Parallel Data Processing with MapReduce: A Survey

Kyong-Ha Lee
Yoon-Joon Lee
Department of Computer Science
KAIST
bart7449@gmail.com
yoonjoon.lee@kaist.ac.kr

Hyunsik Choi
Yon Dohn Chung
Department of Computer Science and Engineering
Korea University
hyunsik.choi@korea.ac.kr
ydchung@korea.ac.kr

Bongki Moon
Department of Computer Science
University of Arizona
bkmoon@cs.arizona.edu

ABSTRACT

A prominent parallel data processing tool MapReduce is gaining significant momentum from both industry and academia as the volume of data to analyze grows rapidly. While MapReduce is used in many areas where massive data analysis is required, there are still debates on its performance, efficiency per node, and simple abstraction. This survey intends to assist the database and open source communities in understanding various technical aspects of the MapReduce framework. In this survey, we characterize the MapReduce framework and discuss its inherent pros and cons. We then introduce its optimization strategies reported in the recent literature. We also discuss the open issues and challenges raised on parallel data analysis with MapReduce.

1. INTRODUCTION

In this age of data explosion, parallel processing is essential to processing a massive volume of data in a timely manner. MapReduce, which has been popularized by Google, is a scalable and fault-tolerant data processing tool that enables to process a massive volume of data in parallel with many low-end computing nodes[44, 38]. By virtue of its simplicity, scalability, and fault-tolerance, MapReduce is becoming ubiquitous, gaining significant momentum from both industry and academia. However, MapReduce has inherent limitations on its performance and efficiency. Therefore, many studies have endeavored to overcome the limitations of the MapReduce framework[10, 15, 51, 32, 23].

The goal of this survey is to provide a timely remark on the status of MapReduce studies and related work focusing on the current research aimed at improving and enhancing the MapReduce framework. We give an overview of major approaches and classify them with respect to their strategies. The rest of the survey is organized as follows. Section 2 reviews the architecture and the key concepts of MapReduce. Section 3 discusses the inherent pros and cons of MapReduce. Section 4 presents the classification and details of recent approaches to improving the MapReduce framework. In Section 5 and 6, we overview major application domains where the MapReduce framework is adopted and discuss open issues and challenges. Finally, Section 7 concludes this survey.

2. ARCHITECTURE

MapReduce is a programming model as well as a framework that supports the model. The main idea of the MapReduce model is to hide details of parallel execution and allow users to focus only on data processing strategies. The MapReduce model consists of two primitive functions: *Map* and *Reduce*. The input for MapReduce is a list of ($key1$, $value1$) pairs and `Map()` is applied to each pair to compute intermediate key-value pairs, ($key2$, $value2$). The intermediate key-value pairs are then grouped together on the key-equality basis, *i.e.* ($key2$, $list(value2)$). For each $key2$, `Reduce()` works on the list of all values, then produces zero or more aggregated results. Users can define the `Map()` and `Reduce()` functions however they want the MapReduce framework works.

MapReduce utilizes the Google File System(GFS) as an underlying storage layer to read input and store output[59]. GFS is a chunk-based distributed file system that supports fault-tolerance by data partitioning and replication. Apache Hadoop is an open-source Java implementation of MapReduce[81]. We proceed our explanation with Hadoop since Google's MapReduce code is not available to the public for its proprietary use. Other implementations (such as DISCO written in Erlang[6]) are also available, but not as popular as Hadoop. Like MapReduce, Hadoop consists of two layers: a data storage layer called Hadoop DFS(HDFS) and a data processing layer called Hadoop MapReduce Framework. HDFS is a block-structured file system managed by a single master node like Google's GFS. Each processing job in Hadoop is broken down to as many Map tasks as input data blocks and one or more Reduce tasks. Figure 1 illustrates an overview of the Hadoop architecture.

A single MapReduce(MR) job is performed in two phases: Map and Reduce stages. The master picks idle workers and assigns each one a map or a reduce task according to the stage. Before starting the Map task, an input file is loaded on the distributed file system. At loading, the file is partitioned into multiple data blocks which have the same size, typically 64MB, and each block is *triplicated* to guarantee fault-tolerance. Each block is then assigned to a mapper, a worker which is assigned a map task, and the mapper applies `Map()` to each record in the data block. The intermediate outputs produced by the mappers are then sorted locally for grouping key-value pairs sharing the same key. After local sort, `Combine()` is optionally applied to perform pre-aggregation on the grouped key-value pairs so that the communication cost taken to transfer all the intermediate outputs to reducers is minimized. Then the mapped outputs are stored in local disks of the mappers, partitioned into R, where R is the number of Reduce tasks in the MR job. This partitioning is basically done by a hash function *e.g.*, `hash(key) mod R`.

When all Map tasks are completed, the MapReduce scheduler assigns Reduce tasks to workers. The intermediate results are shuffled and assigned to reducers via HTTPS protocol. Since all mapped outputs are already partitioned and stored in local disks, each reducer performs the shuffling by simply pulling its partition of the mapped outputs from mappers. Basically, each record of the mapped outputs is assigned to only a single reducer by *one-to-one shuffling* strategy. Note that this data transfer is performed by reducers' pulling intermediate results. A reducer reads the intermediate results and merge them by the intermediate keys, *i.e.* $key2$, so that all values of the same key are grouped together. This grouping is done by *external merge-sort*. Then each reducer applies `Reduce()` to the intermediate values for each $key2$ it encounters. The output of reducers are stored and triplicated in HDFS.

Note that the number of Map tasks does not depend on the number of nodes, but the number of input blocks. Each block is assigned to a single Map task. However, all Map tasks do not need to be executed simultaneously and neither are Reduce tasks. For example, if an input is broken down into 400 blocks and there are 40 mappers in a cluster, the number of map tasks are 400 and the map tasks are executed through 10 waves of task runs. This behavior pattern is also reported in [60].

The MapReduce framework executes its tasks based on *runtime scheduling scheme*. It means that MapReduce does not build any execution plan that specifies which tasks will run on which nodes before execution. While DBMS generates a query plan tree for execution, a plan for executions in MapReduce is determined entirely at runtime. With the runtime scheduling, MapReduce achieves fault tolerance by detecting failures and reassigning tasks of failed nodes to other healthy nodes in the cluster. Nodes which have completed their tasks

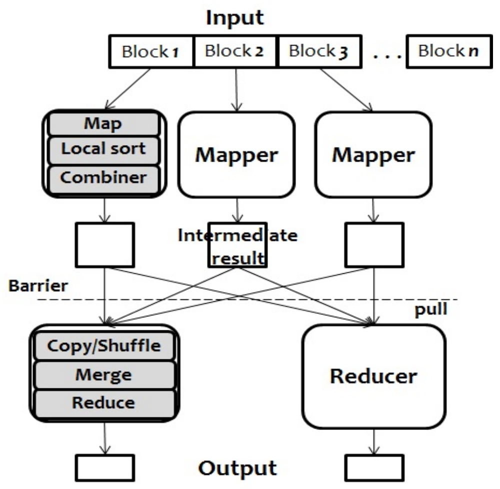


Figure 1: Hadoop Architecture

are assigned another input block. This scheme naturally achieves load balancing in that faster nodes will process more input chunks and slower nodes process less inputs in the next wave of execution. Furthermore, MapReduce scheduler utilizes a speculative and redundant execution. Tasks on straggling nodes are redundantly executed on other idle nodes that have finished their assigned tasks, although the tasks are not guaranteed to end earlier on the new assigned nodes than on the straggling nodes. Map and Reduce tasks are executed with no communication between other tasks. Thus, there is no contention arisen by synchronization and no communication cost between tasks during a MR job execution.

3. PROS AND CONS

3.1 Debates

As suggested by many researchers, commercial DBMSs have adopted "one size fits all" strategy and are not suited for solving extremely large scale data processing tasks. There has been a demand for special-purpose data processing tools that are tailored for such problems [79, 50, 72]. While MapReduce is referred to as a new way of processing big data in data-center computing [77], it is also criticized as a "major step backwards" in parallel data processing in comparison with DBMS [10, 15]. However, many MapReduce proponents in industry argue that MapReduce is not a DBMS and such an apple-to-orange comparison is unfair. As the technical debate continued, ACM recently invited both sides in January edition of CACM, 2010 [51, 39]. Panels in DOLAP'10 also discussed pros and cons of MapReduce and relational DB for data warehousing [23].

Pavlo *et al*'s comparison show that Hadoop is 2∼50 times slower than parallel DBMS except in the case of data loading [15]. Anderson *et al* also criticize that the current Hadoop system is scalable, but achieves very low efficiency per node, less than 5MB/s processing rates, repeating a mistake that previous studies on high-performance systems often made by "focusing on scalability but missing efficiency" [32]. This poor efficiency involves many issues such as performance, total cost of ownership(TCO) and energy. Although Hadoop won the 1st position in GraySort benchmark test for 100 TB sorting(1 trillion 100-byte records) in 2009, its winning was achieved with over 3,800 nodes [76]. MapReduce or Hadoop would not be a cheap solution if the cost for constructing and maintaining a cluster of that size was considered. Other studies on the performance of Hadoop are also found in literature [28, 61]. Analysis of 10-months of MR logs from Yahoo's M45 Hadoop cluster and MapReduce usage statistics at Google are also available [60, 9].

The studies exhibit a clear tradeoff between efficiency and fault-tolerance. MapReduce increases the fault tolerance of long-time analysis by frequent checkpoints of completed tasks and data replication. However, the frequent I/Os required for fault-tolerance reduce efficiency. Parallel DBMS aims at efficiency rather than fault tolerance. DBMS actively exploits pipelining intermediate results between query operators. However, it causes a potential danger that a large amount of operations need be redone when a failure happens. With this fundamental difference, we categorize the pros and cons of the MapReduce framework below.

3.2 Advantages

MapReduce is simple and efficient for computing aggregate. Thus, it is often compared with "*filtering then group-by aggregation*" query processing in a DBMS. Here are major advantages of the MapReduce framework for data processing.

Simple and easy to use The MapReduce model is simple but expressive. With MapReduce, a programmer defines his job with only Map and Reduce functions, without having to specify physical distribution of his job across nodes.

Flexible MapReduce does not have any dependency on data model and schema. With MapReduce a programmer can deal with irregular or unstructured data more easily than they do with DBMS.

Independent of the storage MapReduce is basically independent from underlying storage layers. Thus, MapReduce can work with different storage layers such as BigTable[35] and others.

Fault tolerance MapReduce is highly fault-tolerant. For example, it is reported that MapReduce can continue to work in spite of an average of 1.2 failures per analysis job at Google[44, 38].

High scalability The best advantage of using MapReduce is high scalability. Yahoo! reported that their Hadoop gear could scale out more than 4,000 nodes in 2008[4].

3.3 Pitfalls

Despite many advantages, MapReduce lacks some of the features that have proven paramount to data analysis in DBMS. In this respect, MapReduce is often characterized as an *Extract-Transform-Load*(ETL) tool[51]. We itemize the pitfalls of the MapReduce framework below, compared with DBMS.

No high-level language MapReduce itself does not support any high-level language like SQL in DBMS and any query optimization technique. Users should code their operations in Map and Reduce functions.

No schema and no index MapReduce is schema-free and index-free. An MR job can work right after its input is loaded into its storage. However, this impromptu processing throws away the benefits of data modeling. MapReduce requires to parse each item at reading input and transform it into data objects for data processing, causing performance degradation [15, 11].

A Single fixed dataflow MapReduce provides the ease of use with a simple abstraction, but in a fixed dataflow. Therefore, many complex algorithms are hard to implement with Map and Reduce only in an MR job. In addition, some algorithms that require multiple inputs are not well supported since the dataflow of MapReduce is originally designed to read a single input and generate a single output.

Low efficiency With fault-tolerance and scalability as its primary goals, MapReduce operations are not always optimized for I/O efficiency. (Consider for example sort-merge based grouping, materialization of intermediate results and data triplication on the distributed file system.) In addition, Map and Reduce are blocking operations. A transition to the next stage cannot be made until all the tasks of the current stage are finished. Consequently, pipeline parallelism may not be exploited. Moreover, block-level restarts, a one-to-one shuffling strategy, and a simple runtime scheduling can also lower the efficiency per node. MapReduce does not have specific execution plans and does not optimize plans like DBMS does to minimize data transfer across nodes. Therefore, MapReduce often shows poorer performance than DBMS[15]. In addition, the MapReduce framework has a latency problem that comes from its inherent batch processing nature. All of inputs for an MR job should be prepared in advance for processing.

Very young MapReduce has been popularized by Google since 2004. Compared to over 40 years of DBMS, codes are not mature yet and third-party tools available are still relatively few.

4. VARIANTS AND IMPROVEMENTS

We present details of approaches to improving the pitfalls of the MapReduce framework in this section.

4.1 High-level Languages

Microsoft SCOPE[53], Apache Pig[22, 18], and Apache Hive[16, 17] all aim at supporting declarative query languages for the MapReduce framework. The declarative query languages allow query independence from program logics, reuse of the queries and automatic query optimization features like SQL does for DBMS. SCOPE works on top of the Cosmos system, a Microsoft's clone of MapReduce, and provides functionality similar to SQL views. It is similar to SQL but comes with C# expressions. Operators in SCOPE are the same as Map, Reduce and Merge supported in [37].

Pig is an open source project that is intended to support ad-hoc analysis of very large data, motivated by Sawzall[55], a scripting language for Google's MapReduce. Pig consists of a high-level dataflow language called *Pig Latin* and its execution framework. Pig Latin supports a nested data model and a set of pre-defined UDFs that can be customized [22]. The Pig execution framework first generates a logical query plan from a Pig Latin program. Then it compiles the logical plan down into a series of MR jobs. Some optimization techniques are adopted to the compilation, but not described in detail[18]. Pig is built on top of Hadoop framework, and its usage requires no modification to Hadoop.

Hive is an open-source project that aims at providing data warehouse solutions on top of Hadoop, supporting ad-hoc queries with an SQL-like query language called HiveQL. Hive compiles a HiveQL query into a directed acyclic graph(DAG) of MR jobs. The HiveQL includes its own type system and data definition language(DDL) to manage data integrity. It also contains a system catalog, containing schema information and statistics, much like DBMS engines. Hive currently provides only a simple, naive rule-based optimizer.

Similarly, DryadLINQ[71, 49] is developed to translate LINQ expressions of a program into a distributed execution plan for Dryad, Microsoft's parallel data processing tool [48].

4.2 Schema Support

As described in Section 3.3, MapReduce does not provide any schema support. Thus, the MapReduce framework parses each data record at reading input, causing performance degradation [15, 51, 11]. Meanwhile, Jiang *et al* report that only immutable decoding that transforms records into immutable data objects severely causes performance degradation, rather than record parsing [28].

While MapReduce itself does not provide any schema support, data formats such as Google's Protocol Buffers, XML, JSON, Apache's Thrift, or other formats can be used for checking data integrity [39]. One notable thing about the formats is that they are self-describing formats that support a nested and irregular data model, rather than the relational model. A drawback of the use of the formats is that data size may grow as data contains schema information in itself. Data compression is considered to address the data size problem [47].

4.3 Flexible Data Flow

There are many algorithms which are hard to directly map into Map and Reduce functions. For example, some algorithms require global state information during their processing. Loop is a typical example that requires the state information for execution and termination. However, MapReduce does not treat state information during execution. Thus, MapReduce reads the same data iteratively and materializes intermediate results in local disks in each iteration, requiring lots of I/Os and unnecessary computations. HaLoop[66], Twister[42], and Pregel[36] are examples of systems that support loop programs in MapReduce.

HaLoop and Twister avoid reading unnecessary data repeatedly by identifying and keeping invariant data during iterations. Similarly, Lin *et al* propose an in-mapper combining technique that preserves mapped outputs in a memory buffer across multiple map calls, and emits aggregated outputs at the last iteration [75]. In addition, Twister avoids instantiating workers repeatedly during iterations. Previously instantiated workers are reused for the next iteration with different inputs in Twister. HaLoop is similar to Twister, and it also allows to cache both each stage's input and output to save more I/Os during iterations. Vanilla Hadoop also supports task JVM reuse to avoid the overhead of starting a new JVM for each task [81]. Pregel mainly targets to process graph data. Graph data processing are usually known to require lots of iterations. Pregel implements a programming model motivated by the Bulk Synchronous Parallel(BSP) model. In this model, each node has each own input and transfers only some messages which are required for next iteration to other nodes.

MapReduce reads a single input. However, many important relational operators are binary operators that require two inputs. Map-Reduce-Merge addresses the support of the relational operators by simply adding a third *merge* stage after reduce stage [37]. The merge stage combines two reduced outputs from two different MR jobs into one.

Clustera, Dryad and Nephele/PACT allow more flexible dataflow than MapReduce does [31, 48, 30, 26]. Clustera is a cluster management system that is designed to handle a variety of job types including MR-style jobs [31]. Job scheduler in Clustera handles MapReduce, workflow and SQL-type jobs, and each job can be connected to form a DAG or a pipeline for complex computations.

Dryad is a notable example of distributed data-parallel tool that allows to design and execute a dataflow graph as users' wish [48]. The dataflow in Dryad has a form of DAG that consists of vertices and channels. Each vertex represents a program and a channel connects the vertices. For execution, a logical dataflow graph is mapped onto physical resources by a job scheduler at runtime. A vertex runs when all its inputs are ready and outputs its results to the neighbor vertices via channels as defined in the dataflow graph. The channels can be either of files, TCP pipes, or shared-memory. Job executions are controlled by a central job scheduler. Redundant executions are also allowed to handle apparently very slow vertices, like MapReduce. Dryad also allows to define how to shuffle intermediate data specifically.

Nephele/PACT is another parallel execution engine and its programming model[30, 26]. The PACT model extends MapReduce to support more flexible dataflows. In the model, each mapper can have a separate input and a user can specify its dataflow with more various stages including Map and Reduce. Nephele transforms a PACT program into a physical DAG then executes the DAG across nodes. Executions in Nephele are scheduled at runtime, like MapReduce.

4.4 Blocking Operators

Map and Reduce functions are blocking operations in that all tasks should be completed to move forward to the next stage or job. The reason is that MapReduce relies on external merge sort for grouping intermediate results. This property causes performance degradation and makes it difficult to support online processing.

Logothetis *et al* address this problem for the first time when they build MapReduce abstraction onto their distributed stream engine for ad-hoc data processing[29]. Their *incremental* MapReduce framework processes data like streaming engines. Each task runs continuously with a sliding window. Their system generates MR outputs by reading the items within the window. This stream-based MapReduce processes arriving increments of update tuples, avoiding recomputation of all the tuples from the beginning.

MapReduce Online is devised to support online aggregation and continuous queries in MapReduce[63]. It raises an issue that pull-based communication and checkpoints of mapped outputs limit pipelined processing. To promote pipelining between tasks, they modify MapReduce architecture by making Mappers push their data temporarily stored in local storage to Reducers periodically in the same MR job. Map-side pre-aggregation is also used to reduce communication volumes further.

Li *et al* and Jiang *et al* have found that the merge sort in MapReduce is I/O intensive and dominantly affects the performance of MapReduce [21, 28]. This leads to the use of hash tables for better performance and also incremental processing [21]. In the study, as soon as each map task outputs its intermediate results, the results are hashed and pushed to hash tables held by reducers. Then, reducers perform aggregation on the values in each bucket. Since each bucket in the hash table holds all values which correspond to a distinct key, no grouping is required. In addition, reducers can perform aggregation on the fly even when all mappers are not completed yet.

4.5 I/O Optimization

There are also approaches to reducing I/O cost in MapReduce by using index structures, column-oriented storage, or data compression.

Hadoop++ provides an index-structured file format to improve the I/O cost of Hadoop [40]. However, as it needs to build an index for each file partition at data loading stage, loading time is significantly increased. If the input data are processed just once, the additional cost given by building index may not be justified. HadoopDB also benefits from DB indexes by leveraging DBMS as a storage in each node [11].

There are many studies that describe how column-oriented techniques can be leveraged to improve MapReduce's performance dramatically [35, 62, 68, 12, 69]. Google's BigTable proposes the concept of column family that groups one or more columns as a basic working unit[35]. Google's Dremel is a nested column-oriented storage that is designed to complement MapReduce[62]. The read-only nested data in Dremel are modeled with Protocol Buffers [47]. The data in Dremel are split into multiple columns and records are assembled via finite state machines for record-oriented requests. Dremel is also known to support ad-hoc queries like Hive [16].

Record Columnar File(RCFile), developed by Facebook and adopted by Hive and Pig, is a column-oriented file format on HDFS [68]. Data placement in HDFS is determined by the master node at runtime. Thus, it is argued that if each column in a relation is independently stored in a separate file on HDFS, all related fields in the same record cannot guarantee to be stored in the same node. To get around this, a file format that represents all values of a relation column-wise in a single file is devised. A RCFile consists of a set of row groups, which are acquired by partitioning a relation horizontally. Then in each row group, values are enumerated in column-wise, similar to PAX storage scheme [3].

Llama shows how column-wise data placement helps join processing [69]. A column-oriented file in Llama stores a particular column data with optional index information. It also witnesses that *late materialization* which delays record reconstruction until the column is necessary during query processing is no better than early materialization in many cases.

Floratou *et al* propose a binary column-oriented storage that boosts the performance of Hadoop by an order of magnitude[12]. Their storage format stores each column in a separate file but co-locate associated column files in the same node by changing data placement policy of Hadoop. They also suggest that late materialization with skiplist shows better performance than early materialization, contrary to the result of RCFile. Both Floratou's work and RCFile also use a column-wise data compression in each row group, and adopt a lazy decompression technique to avoid unnecessary decompression during query execution. Hadoop also supports the compression of mapped outputs to save I/Os during the checkpoints[81].

4.6 Scheduling

MapReduce uses a block-level runtime scheduling with a speculative execution. A separate Map task is created to process a single data block. A node which finishes its task early gets more tasks. Tasks on a straggler node are redundantly executed on other idle nodes.

Hadoop scheduler implements the speculative task scheduling with a simple heuristic method which compares the progress of each task to the average progress. Tasks with the lowest progress compared to the average are selected for re-execution. However, this heuristic method is not well suited in a heterogeneous environment where each node has different computing power. In this environment, even a node whose task progresses further than others may be the last if the node's computing power is inferior to others. Longest Approximate Time to End(LATE) scheduling is devised to improve the response time of Hadoop in heterogeneous environments [52]. This scheduling scheme estimates the task progress with the *progress rate*, rather than simple progress score.

Parallax is devised to estimate job progress more precisely for a series of jobs compiled from a Pig program [45]. it pre-runs with sampled data for estimating the processing speeds of each stage. ParaTimer is an extended version of Parallax for DAG-style jobs written in Pig [46]. ParaTimer identifies a critical path that takes longer than others in a parallel query plan. It makes the indicator ignore other shorter paths when estimating progress since the longest path would contribute the overall execution time. Besides, it is reported that the more data blocks to be scheduled, the more cost the scheduler will pay [65]. Thus, a rule of thumb in industry – making the size of data block bigger makes Hadoop work faster – is credible.

We now look into multi-user environment whereby users simultaneously execute their jobs in a cluster. Hadoop implements two scheduling schemes: *fair scheduling* and *capacity scheduling*. The default fair scheduling works with a single queue of jobs. It assigns physical resources to jobs such that all jobs get an equal share of resources over time on average. In this scheduling scheme, if there is only a single MR job running in a cluster, The job solely uses entire resources in the cluster. Capacity sharing supports designing more sophisticated scheduling. It provides multiple queues each of which is guaranteed to possess a certain capacity of the cluster.

MRShare is a remarkable work for sharing multiple query executions in MapReduce [64]. MRShare, inspired by multi query optimization techniques in database, finds an optimal way of grouping a set of queries using dynamic programming. They suggest three sharing opportunities across multiple MR jobs in MapReduce, like found in Pig [18]: scan sharing, mapped outputs sharing, and Map function sharing. They also introduce a cost model for MR jobs and validate this with experiments. Their experiments show that intermediate result sharing improves the execution time significantly. In addition, they have found that sharing all scans yield poorer performance as the size of intermediate results increases, because of the complexity of the merge-sort operation in MapReduce. Suppose that $|D|$ is the size of input data that n MR jobs share. When sharing all scans, the cost of scanning inputs is reduced by $|D|$, compared to $n \cdot |D|$ for no sharing scans. However, as a result, the complexity of sorting the *combined* mapped output of all jobs will be $O(n \cdot |D| log(n \cdot |D|))$ since each job can generate its own mapped output with size $O(|D|)$. This cost can be bigger than the total cost of sorting n different jobs, $O(n \cdot |D| log |D|)$ in some cases.

4.7 Joins

Join is a popular operator that is not so well dealt with by Map and Reduce functions. Since MapReduce is designed for processing a single input, the support of joins that require more than two inputs with MapReduce has been an open issue. We roughly classify join methods within MapReduce into two groups: *Map-side join* and *Reduce-side join*. We also borrow some of terms from Blanas *et al*'s study, which compares many join techniques for analysis of clickstream logs at Facebook [57], for explaining join techniques.

Map-side Join

Map-Merge join is a common map-side join that works similarly to sort-merge join in DBMS. Map-Merge join performs in two steps. First, two input relations are partitioned and sorted on the join keys. Second, mappers read the inputs and merge them [81]. Broadcast join is another map-side join method, which is applicable when the size of one relation is small [57, 7]. The smaller relation is broadcast to each mapper and kept in memory. This avoids I/Os for moving and sorting on both relations. Broadcast join uses in-memory hash tables to store the smaller relation and to find matches via table lookup with values from the other input relation.

Reduce-side Join

Repartition join is the most general reduce-side join [81, 57]. Each mapper tags each row of two relations to identify which relation the row come from. After that, rows of which keys have the same key value are copied to the same reducer during shuffling. Finally, each reducer joins the rows on the key-equality basis. This way is akin to hash-join in DBMS. An improved version of the repartition join is also proposed to fix the buffering problem that all records for a given key need to be buffered in memory during the joining process[57].

Lin *et al* propose a scheme called "schimmy" to save I/O cost during reduce-side join[75]. The basic concept of the scheme is to separate messages from graph structure data, and shuffle only the message to avoid shuffling data, similar to Pregel [36]. In this scheme, mappers emit only messages. Reducers read graph structure data directly from HDFS and do reduce-side merge join between the data and the messages.

MapReduce Variants

Map-Reduce-Merge is the first that attempts to address join problem in the MapReduce framework [37]. To support binary operations including join, Map-Reduce-Merge extends MapReduce model by adding Merge stage after Reduce stage.

Map-Join-Reduce is another variant of MapReduce framework for one-phase joining [27]. The authors propose a filtering-join-aggregation model that adds *Join* stage before Reduce stage to perform joining within a single MR job. Each mapper reads tuples from a separate relation which take part in a join process. After that, the mapped outputs are shuffled and moved to joiners for actual joining, then the `Reduce()` function is applied. Joiners and reducers are actually run inside the same reduce task. An alternative that runs Map-Join-Reduce with two consecutive MR jobs is also proposed to avoid modifying MapReduce framework. For multi-way join, join chains are represented as a left-deep tree. Then previous joiners transfer joined tuples to the next joiner that is the parent operation of the previous joiners in the tree. For this, Map-Join-Reduce adopts one-to-many shuffling scheme that shuffles and assigns each mapped outputs to multiple joiners at a time.

Other Join Types

Joins may have more than two relations. If relations are simply hash-partitioned and fed to reducers, each reducer takes a different portion of the relations. However, the same relation must be copied to all reducers to avoid generating incomplete join results in the cases. For example, given a multi-way join that reads 4 relations and with 4 reducers, we can split only 2 relations making 4 partitions in total. The other relations need to be copied to all reducers. If more relations are involved into less reducers, we spend more communication costs. Afrati *et al* focus on how to minimize the sum of the communication cost of data that are transferred to Reducers for multi-way join [2]. They suggest a method based on Lagrangean multipliers to properly select which columns and how many of the columns should be partitioned for minimizing the sum of the communication costs. Lin *et al* propose the concurrent join that performs a multi-way join in parallel with MapReduce [69].

In addition to binary equal-join, other join types have been widely studied. Okcan *et al* propose how to efficiently perform θ-join with a single MR job only [14]. Their algorithm uses a Reducer-centered cost model that calculates the total cost of Cartesian product of mapped output. With the cost model, they assigns mapped output to reducers that minimizes job completion time. The support of Semi-join, *e.g.* $R \ltimes S$, is proposed in [57]. Vernica *et al* propose how to efficiently parallelize set-similarity joins with Mapreduce [56]. They utilize prefix filtering to filter out non-candidates before actual comparison. It requires to extract common prefixes sorted in a global order of frequency from tuples, each of which consists of a set of items.

4.8 Performance Tuning

Most of MapReduce programs are written for data analysis and they usually take much time to be finished. Thus, it is straightforward to provide the feature of automatic optimization for MapReduce programs. Babu *et al* suggest an automatic tuning approach to finding optimal system parameters for given input data [5]. It is based on speculative pre-runs with sampled data. Jahani *et al* suggest a static analysis approach called MANIMAL for automatic optimization of a single MapReduce job [34]. In their approach, an analyzer examines program codes before execution without any runtime information. Based on the rules found during the analysis, it creates a pre-computed B^+-tree index and slices input data column-wise for later use. In addition, some semantic-aware compression techniques are used for reducing I/O. Its limitation is that the optimization considers only selection and projection which are primarily implemented in Map function.

4.9 Energy Issues

Energy issue is important especially in this data-center computing era. Since the energy cost of data centers hits 23% of the total amortized monthly operating expenses, it is prudent to devise an energy-efficient way to control nodes in a data center when the nodes are idle[74]. In this respect, two extreme strategies for energy management in MapReduce clusters are examined [74, 43]. Covering-Set approach designates in advance some nodes that should keep at least a replica of each data block, and the other nodes are powered down during low-utilization periods [43]. Since the dedicated nodes always have more than one replica of data, all data across nodes are accessible in any cases except for multiple node failures. On the contrary, All-In strategy saves energy in an all-or-nothing fashion [74]. In the strategy, all MR jobs are queued until it reaches a threshold predetermined. If it exceeds, all nodes in the cluster run to finish all MR jobs and then all the nodes are powered down until new jobs are queued enough. Lang *et al* concluded that All-In strategy is superior to Covering-Set in that it does not require changing data placement policy and response time degradation. However, All-In strategy may not support an instant execution because of its batch nature. Similarly, Chen *et al* discuss the computation versus I/O tradeoffs when using data compressions in a MapReduce cluster in terms of energy efficiency [67].

4.10 Hybrid Systems

HadoopDB is a hybrid system that connects multiple single-node DBMS with MapReduce for combining MapReduce-style scalability and the performance of DBMS [11]. HadoopDB utilizes MapReduce as a distributing system which controls multiple nodes which run single-node DBMS engines. Queries are written in SQL, and distributed via MapReduce across nodes. Data processing is boosted by the features of single-node DBMS engines as workload is assigned to the DBMS engines as much as possible.

SQL/MapReduce is another hybrid framework that enables to execute UDF functions in SQL queries across multiple nodes in MapReduce-style [33]. UDFs extend a DBMS with customizing DB functionality. SQL/MapReduce presents an approach to implementing UDF that can be executed across multiple nodes in parallel by virtue of MapReduce. Greenplum also provides the ability to write MR functions in their parallel DBMS. Teradata makes its effort to combine Hadoop with their parallel DBMS [70]. The authors describe their three efforts toward tight and efficient integration of Hadoop and Teradata EDW: parallel loading of Hadoop data to EDW, retrieving EDW data from MR programs, and accessing Hadoop data from SQL via UDFs.

5. APPLICATIONS AND ADAPTATIONS

5.1 Applications

Mahout is an Apache project that aims at building scalable machine learning libraries which are executed in parallel by virtue of Hadoop [1]. RHIPE and Ricardo project are tools that integrate R statistical tool and Hadoop to support parallel data analysis [73, 58]. Cheetah is a data warehousing tool built on MapReduce with virtual view on top of the star or snowflake schemas and with some optimization techniques like data compression and columnar store [7]. Osprey is a shared-nothing database system that supports MapReduce-style fault tolerance [25]. Osprey does not directly use MapReduce or GFS. However, the fact table in star schema is partitioned and replicated like GFS, and tasks are scheduled by a central runtime scheduler like MapReduce. A difference is that Osprey does not checkpoint intermediate outputs. Instead, it uses a technique called chained declustering which limits data unavailability when node failures happen.

The use of MapReduce for data intensive scientific analysis and bioinformatics are well studied in [41, 80]. CloudBLAST parallelizes NCBI BLAST2 algorithm using Hadoop [13]. They break their input sequences down into multiple blocks and run an instance of the vanilla version of NCBI BLAST2 for each block, using the Hadoop Streaming utility [81]. CloudBurst is a parallel read-mapping tool that maps NGS read sequencing data to a reference genome for genotyping in parallel [78].

5.2 Adaptations to Intra-node Parallelism

Some studies use the MapReduce model for simplifying complex multi-thread programming on many-core systems such as multi-core[24, 65], GPU[20, 19], and Cell processors[8]. In the studies, mapped outputs are transferred to reducers via shared-memory rather than disks. In addition, a task execution is performed by a single core rather than a node. In this intra-node parallelism, fault-tolerance can be ignored since all cores are located in a single system. A combination of intra-node and inter-node parallelism by the MapReduce model is also suggested [54].

6. DISCUSSION AND CHALLENGES

MapReduce is becoming ubiquitous, even though its efficiency and performance are controversial. There is nothing new about principles used in MapReduce [10, 51]. However, MapReduce shows that many problems can be solved in the model at scale unprecedented before. Due to frequent checkpoints and runtime scheduling with speculative execution, MapReduce reveals low efficiency. However, such methods would be necessary to achieve high scalability and fault tolerance in massive data processing. Thus, how to increase efficiency guaranteeing the same level of scalability and fault tolerance is a major challenge. The efficiency problem is expected to be overcome in two ways: improving MapReduce itself and leveraging new hardware. How to utilize the features of modern hardware has not been answered in many areas. However, modern computing devices such as chip-level multiprocessors and Solid State Disk(SSD) can help reduce computations and I/Os in MapReduce significantly. The use of SSD in Hadoop with simple installation is briefly examined, but not in detail [21]. Self-tuning and job scheduling in multi-user environments are another issues that have not been well address yet. The size of MR clusters is continuously increasing. A 4,000-node cluster is not surprising any more. How to efficiently manage resources in the clusters of that size in multi-user environment is also challenging. Yahoo's M45 cluster reportedly shows only 5$\sim$10% resource utilization [60]. Energy efficiency in the clusters and achieving high utilizations of MR clusters are also important problems that we consider for achieving better TCO and return on investments in practice.

7. CONCLUSION

We discussed pros and cons of MapReduce and classified its improvements. MapReduce is simple but provides good scalability and fault-tolerance for massive data processing. However, MapReduce is unlikely to substitute DBMS even for data warehousing. Instead, we expect that MapReduce complements DBMS with scalable and flexible parallel processing for various data analysis such as scientific data processing. Nonetheless, efficiency, especially I/O costs of MapReduce still need to be addressed for successful implications.

Acknowledgement

This work is supported by the National Research Fund (NRF) grant funded by Korea government(MEST)(No. 2011-0016282).

8. REFERENCES

[1] Mahout: Scalable machine-learning and data-mining library. *http://mapout.apache.org*, 2010.
[2] F.N. Afrati and J.D. Ullman. Optimizing joins in a map-reduce environment. In *Proceedings of the 13th EDBT*, pages 99–110, 2010.
[3] A. Ailamaki, D.J. DeWitt, M.D. Hill, and M. Skounakis. Weaving relations for cache performance. *The VLDB Journal*, pages 169–180, 2001.
[4] A. Anand. Scaling Hadoop to 4000 nodes at Yahoo! *http://goo.gl/8dRMq*, 2008.
[5] S. Babu. Towards automatic optimization of mapreduce programs. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 137–142, 2010.
[6] Nokia Research Center. Disco: Massive data- minimal code. *http://discoproject.org*, 2010.
[7] S. Chen. Cheetah: a high performance, custom data warehouse on top of MapReduce. *Proceedings of the VLDB*, 3(1-2):1459–1468, 2010.
[8] M. de Kruijf and K. Sankaralingam. Mapreduce for the cell broadband engine architecture. *IBM Journal of Research and Development*, 53(5):10:1–10:12, 2009.
[9] J. Dean. Designs, lessons and advice from building large distributed systems. *Keynote from LADIS*, 2009.
[10] D. DeWitt and M. Stonebraker. MapReduce: A major step backwards. *The Database Column*, 1, 2008.
[11] A. Abouzeid *et al*. HadoopDB: An architectural hybrid of MapReduce and DBMS technologies for analytical workloads. *Proceedings of the VLDB Endowment*, 2(1):922–933, 2009.
[12] A. Floratou *et al*. Column-Oriented Storage Techniques for MapReduce. *Proceedings of the VLDB*, 4(7), 2011.
[13] A. Matsunaga *et al*. Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications. In *Fourth IEEE International Conference on eScience*, pages 222–229, 2008.
[14] A. Okcan *et al*. Processing Theta-Joins using MapReduce. In *Proceedings of the 2011 ACM SIGMOD*, 2011.
[15] A. Pavlo *et al*. A comparison of approaches to large-scale data analysis. In *Proceedings of the ACM SIGMOD*, pages 165–178, 2009.
[16] A. Thusoo *et al*. Hive: a warehousing solution over a map-reduce framework. *Proceedings of the VLDB Endowment*, 2(2):1626–1629, 2009.
[17] A. Thusoo *et al*. Hive-a petabyte scale data warehouse using Hadoop. In *Proceedings of the 26th IEEE ICDE*, pages 996–1005, 2010.
[18] A.F. Gates *et al*. Building a high-level dataflow system on top of Map-Reduce: the Pig experience. *Proceedings of the VLDB Endowment*, 2(2):1414–1425, 2009.
[19] B. Catanzaro *et al*. A map reduce framework for programming graphics processors. In *Workshop on Software Tools for MultiCore Systems*, 2008.
[20] B. He *et al*. Mars: a MapReduce framework on graphics processors. In *Proceedings of the 17th PACT*, pages 260–269, 2008.
[21] B. Li *et al*. A Platform for Scalable One-Pass Analytics using MapReduce. In *Proceedings of the 2011 ACM SIGMOD*, 2011.
[22] C. Olston *et al*. Pig latin: a not-so-foreign language for data processing. In *Proceedings of the ACM SIGMOD*, pages 1099–1110, 2008.
[23] C. Ordonez *et al*. Relational versus Non-Relational Database Systems for Data Warehousing. In *Proceedings of the ACM DOLAP*, pages 67–68, 2010.
[24] C. Ranger *et al*. Evaluating mapreduce for multi-core and multiprocessor systems. In *Proceedings of the 2007 IEEE HPCA*, pages 13–24, 2007.
[25] C. Yang *et al*. Osprey: Implementing MapReduce-style fault tolerance in a shared-nothing distributed database. In *Proceedings of the 26th IEEE ICDE*, 2010.
[26] D. Battré *et al*. Nephele/PACTs: a programming model and execution framework for web-scale analytical processing. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 119–130, 2010.
[27] D. Jiang *et al*. Map-join-reduce: Towards scalable and efficient data analysis on large clusters. *IEEE Transactions on Knowledge and Data Engineering*, 2010.
[28] D. Jiang *et al*. The performance of mapreduce: An in-depth study. *Proceedings of the VLDB Endowment*, 3(1-2):472–483, 2010.
[29] D. Logothetis *et al*. Ad-hoc data processing in the cloud. *Proceedings of the VLDB Endowment*, 1(2):1472–1475, 2008.
[30] D. Warneke *et al*. Nephele: efficient parallel data processing in the cloud. In *Proceedings of the 2nd MTAGS*, pages 1–10, 2009.
[31] D.J. DeWitt *et al*. Clustera: an integrated computation and data management system. *Proceedings of the VLDB Endowment*, 1(1):28–41, 2008.
[32] E. Anderson *et al*. Efficiency matters! *ACM SIGOPS Operating Systems Review*, 44(1):40–45, 2010.
[33] E. Friedman *et al*. SQL/MapReduce: A practical approach to self-describing, polymorphic, and parallelizable user-defined functions. *Proceedings of the VLDB*

Endowment, 2(2):1402–1413, 2009.

[34] E. Jahani *et al*. Automatic Optimization for MapReduce Programs. *Proceedings of the VLDB*, 4(6):385–396, 2011.

[35] F. Chang *et al*. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):1–26, 2008.

[36] G. Malewicz *et al*. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD*, pages 135–146, 2010.

[37] H. Yang *et al*. Map-reduce-merge: simplified relational data processing on large clusters. In *Proceedings of the 2007 ACM SIGMOD*, pages 1029–1040, 2007.

[38] J. Dean *et al*. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[39] J. Dean *et al*. MapReduce: a flexible data processing tool. *Communications of the ACM*, 53(1):72–77, 2010.

[40] J. Dittrich *et al*. Hadoop++: Making a yellow elephant run like a cheetah (without it even noticing). *Proceedings of the VLDB Endowment*, 3(1-2):515–529, 2010.

[41] J. Ekanayake *et al*. Mapreduce for data intensive scientific analyses. In *the 4th IEEE International Conference on eScience*, pages 277–284, 2008.

[42] J. Ekanayake *et al*. Twister: A runtime for iterative MapReduce. In *Proceedings of the 19th ACM HPDC*, pages 810–818, 2010.

[43] J. Leverich *et al*. On the energy (in) efficiency of hadoop clusters. *ACM SIGOPS Operating Systems Review*, 44(1):61–65, 2010.

[44] Jeffrey Dean *et al*. Mapreduce: Simplified data processing on large clusters. In *In Proceedings of the 6th USENIX OSDI*, pages 137–150, 2004.

[45] K. Morton *et al*. Estimating the Progress of MapReduce Pipelines. In *Proceedings of the the 26th IEEE ICDE*, pages 681–684, 2010.

[46] K. Morton *et al*. Paratimer: a progress indicator for mapreduce dags. In *Proceedings of the 2010 ACM SIGMOD*, pages 507–518, 2010.

[47] Kenton *et al*. Protocol Buffer- Google's data interchange format. *http://code.google.com/p/protobuf/*.

[48] M. Isard *et al*. Dryad: distributed data-parallel programs from sequential building blocks. In *Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, pages 59–72, 2007.

[49] M. Isard *et al*. Distributed data-parallel computing using a high-level programming language. In *Proceedings of the ACM SIGMOD*, pages 987–994, 2009.

[50] M. Stonebraker *et al*. One size fits all? Part 2: Benchmarking results. In *Conference on Innovative Data Systems Research (CIDR)*, 2007.

[51] M. Stonebraker *et al*. MapReduce and parallel DBMSs: friends or foes? *Communications of the ACM*, 53(1):64–71, 2010.

[52] M. Zaharia *et al*. Improving mapreduce performance in heterogeneous environments. In *Proceedings of the 8th USENIX OSDI*, pages 29–42, 2008.

[53] R. Chaiken *et al*. Scope: easy and efficient parallel processing of massive data sets. *Proceedings of the VLDB Endowment*, 1(2):1265–1276, 2008.

[54] R. Farivar *et al*. Mithra: Multiple data independent tasks on a heterogeneous resource architecture. In *IEEE International Conference on Cluster Computing and Workshops*, pages 1–10, 2009.

[55] R. Pike *et al*. Interpreting the data: Parallel analysis with Sawzall. *Scientific Programming*, 13(4):277–298, 2005.

[56] R. Vernica *et al*. Efficient parallel set-similarity joins using mapreduce. In *Proceedings of the 2010 ACM SIGMOD*, pages 495–506. ACM, 2010.

[57] S. Blanas *et al*. A comparison of join algorithms for log processing in MaPreduce. In *Proceedings of the 2010 ACM SIGMOD*, pages 975–986, 2010.

[58] S. Das *et al*. Ricardo: integrating R and Hadoop. In *Proceedings of the 2010 ACM SIGMOD*, pages 987–998, 2010.

[59] S. Ghemawat *et al*. The google file system. *ACM SIGOPS Operating Systems Review*, 37(5):29–43, 2003.

[60] S. Kavulya *et al*. An analysis of traces from a production mapreduce cluster. In *10th IEEE/ACM CCGrid*, pages 94–103, 2010.

[61] S. Loebman *et al*. Analyzing massive astrophysical datasets: Can Pig/Hadoop or a relational DBMS help? In *IEEE International Conference on Cluster Computing and Workshops*, pages 1–10, 2009.

[62] S. Melnik *et al*. Dremel: interactive analysis of web-scale datasets. *Proceedings of the VLDB Endowment*, 3(1-2):330–339, 2010.

[63] T. Condie *et al*. MapReduce online. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, pages 21–21, 2010.

[64] T. Nykiel *et al*. MRshare: Sharing across multiple queries in mapreduce. *Proceedings of the VLDB*, 3(1-2):494–505, 2010.

[65] W. Jiang *et al*. A Map-Reduce System with an Alternate API for Multi-core Environments. In *Proceedings of the 10th IEEE/ACM CCGrid*, pages 84–93, 2010.

[66] Y. Bu *et al*. HaLoop: Efficient iterative data processing on large clusters. *Proceedings of the VLDB Endowment*, 3(1-2):285–296, 2010.

[67] Y. Chen *et al*. To compress or not to compress - compute vs. IO tradeoffs for mapreduce energy efficiency. In *Proceedings of the first ACM SIGCOMM workshop on Green networking*, pages 23–28. ACM, 2010.

[68] Y. He *et al*. RCFile: A Fast and Space-efficient Data Placement Structure in MapReduce-based Warehouse Systems. In *Proceedings of the 2011 IEEE ICDE*, 2011.

[69] Y. Lin *et al*. Llama: Leveraging Columnar Storage for Scalable Join Processing in the MapReduce Framework. In *Proceedings of the 2011 ACM SIGMOD*, 2011.

[70] Y. Xu *et al*. Integrating Hadoop and parallel DBMS. In *Proceedings of the ACM SIGMOD*, pages 969–974, 2010.

[71] Y. Yu *et al*. DryadLINQ: A system for general-purpose distributed data-parallel computing using a high-level language. In *Proceedings of the 8th USENIX OSDI*, pages 1–14, 2008.

[72] D. Florescu and D. Kossmann. Rethinking cost and performance of database systems. *ACM SIGMOD Record*, 38(1):43–48, 2009.

[73] Saptarshi Guha. RHIPE- R and Hadoop Integrated Processing Environment. *http://www.stat.purdue.edu/ sguha/rhipe/*, 2010.

[74] W. Lang and J.M. Patel. Energy management for MapReduce clusters. *Proceedings of the VLDB*, 3(1-2):129–139, 2010.

[75] J. Lin and C. Dyer. Data-intensive text processing with MapReduce. *Synthesis Lectures on Human Language Technologies*, 3(1):1–177, 2010.

[76] O. O'Malley and A.C. Murthy. Winning a 60 second dash with a yellow elephant. *Proceedings of sort benchmark*, 2009.

[77] D.A. Patterson. Technical perspective: the data center is the computer. *Communications of the ACM*, 51(1):105–105, 2008.

[78] M.C. Schatz. CloudBurst: highly sensitive read mapping with MapReduce. *Bioinformatics*, 25(11):1363, 2009.

[79] M. Stonebraker and U. Cetintemel. One size fits all: An idea whose time has come and gone. In *Proceedings of the 21st IEEE ICDE*, pages 2–11. IEEE, 2005.

[80] R. Taylor. An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics. *BMC bioinformatics*, 11(Suppl 12):S1, 2010.

[81] T. White. *Hadoop: The Definitive Guide*. Yahoo Press, 2010.

Processing and Visualizing the Data in Tweets

Adam Marcus, Michael S. Bernstein, Osama Badar,
David R. Karger, Samuel Madden, Robert C. Miller
MIT CSAIL
{marcua, msbernst, badar, karger, madden, rcm}@csail.mit.edu

ABSTRACT

Microblogs such as Twitter provide a valuable stream of diverse user-generated data. While the data extracted from Twitter is generally timely and accurate, the process by which developers extract structured data from the tweet stream is ad-hoc and requires reimplementation of common data manipulation primitives. In this paper, we present two systems for querying and extracting structure from Twitter-embedded data. The first, TweeQL, provides a streaming SQL-like interface to the Twitter API, making common tweet processing tasks simpler. The second, TwitInfo, shows how end-users can interact with and understand aggregated data from the tweet stream, in addition to showcasing the power of the TweeQL language. Together these systems show the richness of content that can be extracted from Twitter.

1. INTRODUCTION

The Twitter messaging service is wildly popular, with millions of users posting more than 200 million *tweets* per day[1]. This stream of messages from a variety of users contains information on an array of topics, including conventional news stories, events of local interest (e.g., local sports scores), opinions, real-time events (e.g., earthquakes), and many others.

Unfortuantely, the Twitter interface does not make it easy to access this information. The majority of useful information is embedded in unstructured tweet text that is obfuscated by abbreviations (to overcome the 140-character text limit), social practices (e.g., prepending tweets from other users with *RT*), and references (e.g., URLs of full stories, or the @*usernames* of other users). Twitter's APIs provide access to tweets from a particular time range, from a particular user, with a particular keyword, or from a particular geographic region, but provides no facility to extract structure from tweets, and does not provide aggregate views of tweets on different topics (e.g., the frequency of tweets about a particular topic over time.)

[1]`http://blog.twitter.com/2011/06/200-million-tweets-per-day.html`

In this paper, we describe two approaches we have devised to help programmers and end-users make sense of the tweet stream. For programmers, we have built TweeQL, a SQL-like stream processor that provides streaming semantics and a collection of user-defined functions to extract and aggregate tweet-embedded data. For end-users, we built TwitInfo [7], a timeline-based visualization of events in the tweetstream, linked to raw tweet text, sentiment analysis, and maps.

2. TWEEQL

TweeQL provides a SQL-like query interface on top of the Twitter streaming API. The streaming API allows users to issue long-running HTTP requests with keyword, location, or userid filters, and receive tweets that appear on the stream and match these filters. TweeQL provides windowed select-project-join-aggregate queries over this stream, and utilizes user-defined functions for deeper processing of tweets and tweet text.

We begin by describing the TweeQL data model, and then illustrate its operation through a series of expamples. We close with a discussion of challenges with building TweeQL and future directions.

2.1 Data Model and Query Language

TweeQL is based on SQL's select-project-join-aggregate syntax. Its data model is relational, with both traditional table semantics as well as streaming semantics.

2.1.1 *Streams*

The primary stream that TweeQL provides is *twitter_stream*. TweeQL users define new streams based on this base stream using the *CREATE STREAM* statement, which creates a named substream of the main twitter stream that satistifies a particular set of filters. For example, the following statement creates a queriable stream of tweets containing the term *obama* generated from the *twitter_stream* streaming source:

```
CREATE STREAM obama_tweets
FROM twitter_stream
WHERE text contains 'obama';
```

While *twitter_stream* offers several fields (e.g., *text*, *username*, *userid*, *location*, *latitude*, *longitude*), the Twitter API only allows certain filters to be used as access methods for defining a stream. When defining a new stream on top of *twitter_stream*, the developer must provide a key lookup on *userid*, a text match on *text*, or a range lookup on *latitude* and *longitude*. If a user tries to create a stream from a streaming source but omits API-required filters, TweeQL will raise an error.

Users are not allowed to directly query the raw *twitter_stream* because Twitter only provides access to tweets that contain a filter. If users wish to access an unrestricted stream, Twitter provides a sampled, unfiltered stream that TweeQL wraps as *twitter_sample*. A complete, unfiltered stream is not provided by Twitter for performance and financial reasons.

While our examples show users creating streams from the *twitter_stream* base stream, in principle one could also wrap other streaming sources, such as RSS feeds, a Facebook news feed, or a Google+ feed. Once wrapped, derived streams can be generated using techniques similar to the examples we provide.

2.1.2 UDFs

TweeQL also supports user-defined functions (UDFs). UDFs in TweeQL are designed to provide operations over unstructured data such as text blobs. To support such diversity in inputs and outputs, TweeQL UDFs accept and return array- or table-valued attributes. TweeQL UDFs also help wrap web APIs for various services, such as geocoding services.

Complex Data Types. TweeQL UDFs can accept array- or table-valued attributes as arguments. This is required because APIs often allow a variable number of parameters. For example, a geocoding API might allow multiple text locations to be be mapped to latitude/longitude pairs in a single web service request.

UDFs can also return several values at once. This behavior is needed both for batched APIs that submit multiple requests at once, and for many text-processing tasks that are important in unstructured text processing. For example, to build an index of words that appear in tweets, one can issue the following query:

```
SELECT tweetid, tokenize(text)
FROM obama_tweets;
```

The *tokenize* UDF returns an array of words that appear in the tweet text. For example, *tokenize("Tweet number one") = ["Tweet", "number", "one"]*. While arrays can be stored or passed to array-valued functions, users often wish to "relationalize" them. To maintain the relational model, we provide a *FLATTEN* operator (based on the operator of the same name from Olston et al.'s Pig Latin [8]). Users can wrap an array-valued function found in a *SELECT* clause with a *FLATTEN* to produce a result without arrays. For example, instead of the above query, the programmer could write:

```
SELECT tweetid, FLATTEN(tokenize(text))
FROM obama_tweets;
```

The resulting tuples for a tweet with *tweetid = 5* and *text = "Tweet number one"* would then be:

```
(5, 'Tweet')
(5, 'number')
(5, 'one')
```

Web Services as UDFs. Much of TweeQL's structure-extraction functionality is provided by third parties as web APIs. TweeQL allows UDF implementers to make calls to such web services to access their functionality. One such UDF is *geocode*, which returns the latitude and longitude for user-reported textual locations as described in Section 2.1.4. The benefit of wrapping such functionality in third party services is that often the functionality requires large datasets—good geocoding datasets can be upward of several gigabytes—that an implementer can not or does not wish to package with their UDF. Wrapping services comes at a cost, however, as service calls generally incur high latency, and service providers often limit how frequently a client can make requests to their service.

Because calls to other web services may be slow or rate-limited, a TweeQL UDF developer can specify several parameters in addition to the UDF implementation. For example, the developer can add a cache invalidation policy for cacheable UDF invocations, as well as any rate-limiting policies that the API they are wrapping allows. To ensure quality of service, the developer can also specify a timeout on wrapped APIs. When the timeout expires, the return token *TIMEOUT* is returned, which acts like a *NULL* value but can explicitly be fetched at a later time. Similarly, a *RATELIMIT* token can be returned for rate-limited UDFs.

2.1.3 Storing Data and Generating Streams

It is often useful for TweeQL developers to break their workflows into multiple steps and to write final results into a table. To support both of these operations, we allow the results of SELECT statements over streams to write data to named tables. In this way, intermediate steps can be named to allow subsequent queries in a workflow to utilize their results.

Output to a table and temporarily naming tuples is accomplished via the *INTO* operator. To save results, a programmer can add an *INTO TABLE tablename* clause to their query. To name a set of results that can be loaded as a stream by another query, the programmer can add an *INTO STREAM streamname* clause to their query. For example, consider the following three queries:

```
CREATE STREAM sampled
FROM twitter_sample;

SELECT text, sentiment(text) AS sent
FROM sampled
INTO STREAM text_sentiment;

SELECT text
FROM text_sentiment
WHERE sent > 0
INTO TABLE positive_sentiment;

SELECT text, sent
FROM text_sentiment
WHERE text contains 'obama'
INTO TABLE obama_sentiment;
```

The first query creates an unfiltered sampled stream called *sampled*. The second query retrieves all tweet text and its sentiment (described in Section 2.1.4), and places that text in a stream called *text_sentiment*. The third query stores all positive-sentiment tweet text from the *text_sentiment* stream in a table called *positive_sentiment*. The final query stores all tweet text from the *text_sentiment* stream containing the term *obama* in a table called *obama_sentiment*.

For testing purposes, it is also possible to select a stream *INTO STDOUT*, which outputs the contents of a stream to a user's console.

2.1.4 Structure Extraction UDFs

One key feature of our TweeQL implementation is that it provides a library of useful UDFs. One important class of operators are those that allow programmers to extract structure from unstructured content. The functions are described below.

String Processing. String functions help extract structure from text. We have already described one such UDF, *tokenize* in Section 2.1.2 that splits strings into a list of tokens. Other UDFs allow more complex string extraction, such as regular expressions that return lists of matches for each string.

Location. Tweets are annotated with location information in several ways. GPS-provided coordinates are most accurate, but only a small fraction of tweets are annotated with such precision (0.77% in mid-2010 [6]). More common is a self-reported location field, with values ranging from the nonsensical "Justin Bieber's heart" [6] to a potentially accurate "Boston, MA."

To extract structure from self-reported location strings, we offer a *geocode* UDF. The following query extracts the sentiment of tweets containing the term *obama* as well as the coordinates of the self-reported location:

```
SELECT sentiment(text) AS sent,
       geocode(loc).latitude AS lat,
       geocode(loc).longitude AS long
FROM obama_tweets
INTO STREAM obama_sent_loc;
```

The query also displays another feature of TweeQL UDFs. In addition to being able to return lists of fields to be flattened into a resultset, UDFs can return tuples rather than fields. In the example above, *geocode* returns a tuple of coordinates that the query projects into two fields, *lat* and *long*, in the result set.

Classification. Classifiers can be used to identify structure in unstructured text content. For example, social science researchers explore various ways to use the tweet stream as a proxy for public sentiment about various topics. TweeQL provides a *sentiment* UDF for classifying tweet text as expressing positive or negative sentiment. An example of this UDF can be seen in the *obama_sent_loc* stream example above. Other classifiers might identify the topic, language, or veracity of a tweet.

Named entity extraction. So far, we have identified tweets about President Obama by filtering tweets whose text contains the term *obama*. This approach may be unacceptable when two people with the same name might be confused. For example, searching for tweets containing the term *clinton* combines tweets such as "Secretary Clinton accepts Crowley resignation" and ones such as "Former President Clinton undergoes heart surgery."

To reduce ambiguity, TweeQL provides a *namedEntities* UDF that identifies potential entities in context. For example, *namedEntities("Secretary Clinton accepts Crowley resignation")* returns a list of fields *["Hillary Clinton", "P.J. Crowley"]* that can be filtered.

With the *namedEntities* UDF, we can refine our original *obama_tweets* example to identify tweets specifically involving Barack Obama.

```
CREATE STREAM obama_tweets
FROM twitter_stream
WHERE text contains 'obama';

SELECT text,
  FLATTEN(namedEntities(text)) AS entity
FROM obama_tweets
INTO STREAM obama_entities;

SELECT text
FROM obama_entities
WHERE entity = "Barack Obama"
INTO STREAM barack_obama_tweets;
```

The current implementation of *namedEntities* is an API wrapper around OpenCalais [2], a web service for

[2] http://www.opencalais.com/

performing named entity extraction and topic identification. OpenCalais was designed to handle longer text blobs (e.g., a newspaper article) for better contextual named entity extraction. One area of future work is to develop named entity extractors for tweets, which are significantly shorter.

2.1.5 Windowed Operators

Like other stream processing engines, TweeQL supports aggregates and joins on streams. Because streams are infinite, we attach sliding window semantics to them, as in other streaming systems [2, 1]. Windows are defined by a *WINDOW* parameter specifying the timeframe during which to calculate an aggregate or join.

Any streaming source must include a *__created_at* timestamp field. By default, tuples are timestamped with their creation time. On aggregates, an *EVERY* parameter specifies how frequently to emit *WINDOW*-sized aggregates. If an *EVERY* parameter is smaller than the *WINDOW* parameter, overlapping windows are emitted. The *__created_at* field of a tuple emitted from an aggregate is the time that the window begins.

The query below provides an example of the *WINDOW* and *EVERY* parameters for aggregates:

```
SELECT AVG(sent) AS sent,
       floor(lat) AS lat,
       floor(long) AS long
FROM obama_sent_loc
GROUP BY lat, long
WINDOW 3 hours
EVERY 1 hour
INTO STREAM obama_sent_by_area;
```

The query converts the *obama_sent_loc* stream of sentiment, latitude, and longitude into an average sentiment expressed in a 1°x 1°area. This average is computed over the course of three hours, and is emitted every hour.

2.1.6 Event Detection

As we explore with TwitInfo in Section 3, the number of tweets per minute mentioning a topic is a good signal of peaking interest in the topic. If the number of tweets per minute is significantly higher than recent history, it might suggest that an event of interest has just occurred.

To support event detection, we provide a *meanDeviation* UDF. The UDF takes a floating-point value as an argument. It returns the difference between the value and an exponentially weighted moving mean (EWMA) of recent values. This difference is called the mean deviation. Before returning the EWMA, it updates the EWMA with the floating-point value for future calls. The details of this algorithm are spelled out in [7]. The following example illustrates its use:

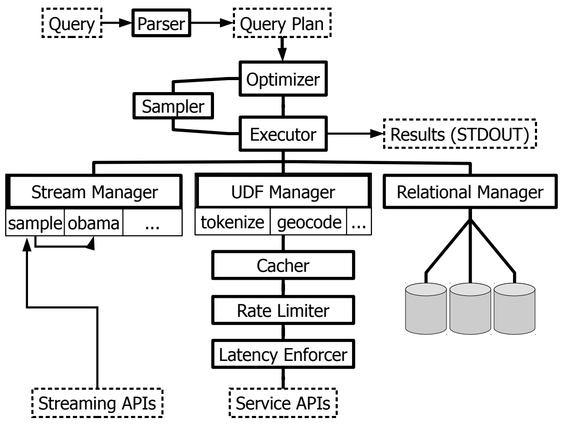


Figure 1: TweeQL architectural components.

```
SELECT COUNT(text) as count,
       __created_at as time
FROM obama_tweets
WINDOW 1 minute
EVERY 1 minute
INTO STREAM obama_counts;

SELECT meanDeviation(count) AS dev,
       time
FROM obama_counts
WHERE dev > 2
INTO TABLE obama_peaks;
```

The first query uses windowed aggregates, described in Section 2.1.5, to calculate the tweets per minute mentioning the term *obama*. The second query calculates the mean deviation of each tweets-per-minute value, and stores the time of deviations above 2 in a table *obama_peaks*.

The *meanDeviation* UDF is unique in that it stores state that is updated between calls. This makes the semantics of the UDF difficult to define, as calling *meanDeviation(count)* on the same *count* value with different histories will result in a different return value. In practice we found that the simple interface to the *meanDeviation* UDF makes it usable for event detection.

2.2 System Design

Figure 1 illustrates the key architectural components of the TweeQL stream processor.

TweeQL offers its SQL-like query language through a traditional query prompt or in batched query mode. All of the queries that make up a workflow (e.g., *sampled*, *text_sentiment*, *positive_sentiment*, and *obama_sentiment* in Section 2.1.3) are handled together and sent to the **Query Parser** to be processed at the same time.

The parser generates batches of dependent query trees, some of which store records in tables while others generate streams that other query trees depend on. The

Optimizer reorders operators as informed by selectivity and latency statistics collected by the **Sampler**. In addition to reordering operators, the optimizer also decides which filters to send to streaming APIs such as Twitter's to reduce the number of tuples returned. The sampler keeps statistics on all APIs and tables known to the database.

Optimized query tree batches are sent to the **Executor**. The query executor is iterator-based. As streaming sources asynchronously generate tuples, the streams are buffered by streaming access method operators that allow iterator access. Streamed tuples appear as tuples with a fixed schema to the rest of the query tree. As we see in Section 2.1.3, a stream (such as *text_sentiment*) can be used by multiple downstream query trees. Downstream query trees register themselves as listeners to named streams that send batches of tuples generated at their root to the streaming buffer of each query tree.

There are three data source managers from which the executor retrieves data: a stream manager, a UDF manager, and a relational manager.

The **Stream Manager** manages all streams generated with *CREATE STREAM* or *INTO STREAM*. It communicates with streaming APIs such as Twitter's, and informs streaming access method operators in query trees when new batches of tuples arrive from streaming sources.

The **UDF Manager** manages all UDF invocations. While traditional UDFs are executed as they are in traditional RDBMSs, the UDF Manager has special logic for handling UDFs which wrap web services. In addition to providing adapters that generate relational data from nonrelational services, it contains components that apply to all requests. The **Cacher** ensures that frequent service requests are cached, and supports age- and frequency-based cache eviction policies. The **Rate Limiter** enforces service-based rate limiting policies. These policies generally limit the number of requests per minute, hour, or day. Finally, the **Latency Enforcer** ensures that requests that run for too long are returned with *TIMEOUT* as discussed in Section 2.1.2. The latency enforcer still allows requests returned after a timeout to be cached for future performance benefits.

The **Relational Manager** simply wraps traditional relational data sources for querying, and stores tables generated with *INTO TABLE* syntax.

2.3 Current Status

TweeQL is implemented in Python, using about 2500 lines of code. The implementation is available as an open source distribution[3]. The distribution includes most of the features described in this paper. We are working to add the rate-limiting and latency-enforcing logic to web service UDF wrappers. The *CREATE STREAM* and *INTO STREAM* statements, which we realized were necessary as we wrapped streams for services other than Twitter, are available in experimental versions of TweeQL. Finally, we intend to add *FLATTEN* syntax in the next TweeQL release.

[3]https://github.com/marcua/tweeql

2.4 Challenges

In this section, we describe a number of challenges and open issues we encountered when building TweeQL.

Uncertain Selectivities. When creating a stream, TweeQL users may issue multiple filters that can be passed to the streaming API. Only one filter type can be submitted to the API, and selecting the most efficient one to send is difficult. For example, consider a user issuing the query:

```
CREATE STREAM obama_nyc
FROM twitter_stream
WHERE text contains 'obama';
  AND location in [bounding box for NYC];
```

The user wants to see all tweets containing the word *obama* that are tweeted from the New York City area. TweeQL must select between requesting all *obama* tweets, or all *NYC* tweets.

We benefit from having access to Twitter's historical API in this case. We can issue two requests for recent tweets with both filters applied, and determine which stream is less frequent. We are also exploring Eddies-style [3] dynamic operator reodering to adjust to changes in operator selectivity over time.

High-latency Operators. As discussed in Section 2.1.2, TweeQL UDFs can return *TIMEOUT* and *RATELIMIT* for long-running or rate-limited web services. Still, the high latency of operations is in tension with the traditional blocking iterator model of query execution.

Web service API requests such as geolocation can take hundreds of milliseconds apiece, but incur little processing cost on the query executor. Though the operations incur little computational cost, they often bottleneck blocking iterators. Caching responses and batching multiple requests when an API allows can reduce some request overhead.

We are exploring modifying iterators to operate asynchronously as described by Goldman and Widom [5]. This, in combination with a data model that allows partial results as described by Raman and Hellerstein [9], might be a sufficient solution.

Aggregate Classifiers are Misleading. In the development of TwitInfo, described in Section 3, we ran into an issue with running aggregates over the output of classifiers such as the *sentiment* UDF. We describe the problem and one solution in detail in [7].

One such example can be seen in the *obama_sent_by_area* stream in Section 2.1.5. Consider the case where the *sentiment* UDF simply outputs 1 for postive text and -1 for negative text. It is possible that the classifier powering *sentiment* has different recall (e.g., the fraction of text identified as positive in situations where the text is actually positive) for positive and negative classifications. In this case, *AVG(sent)* will be biased toward the class with higher recall. The solution described in [7] is to adjust for this bias by learning the positive and negative recall values ($recall_{positive}$ and $recall_{negative}$) on training data. With these values, we can return $\frac{1}{recall_{positive}}$ for positive text, and $\frac{-1}{recall_{negative}}$ for negative text. These values, when summed or averaged, adjust for overall recall differences.

3. TWITINFO

TwitInfo [7] is an application written on top of the TweeQL stream processor. TwitInfo is a user interface that summarizes events and people in the news by following what Twitter users say about those topics over time[4]. TwitInfo offers an example of how aggregate data extracted from tweets can be used in a user interface. Other systems, such as Vox Civitas [4], allow similar exploration, but TwitInfo focuses on the streaming nature of tweet data and uses event detection to relay a story.

3.1 Creating an Event

TwitInfo users define an *event* by specifying a Twitter keyword query. For example, for a soccer game, users might enter search keywords *soccer, football, premierleague,* and team names like *manchester* and *liverpool.* Users give the event a human-readable name like "Soccer: Manchester City vs. Liverpool" as well as an optional time window. When users are done entering the information, TwitInfo saves the event and begins logging tweets containing the keywords using a TweeQL query like the following:

```
CREATE STREAM twitinfo
FROM twitter_stream
WHERE text contains 'soccer'
   OR text contains 'football'
   OR text contains 'premierleague'
   OR text contains 'manchester'
   OR text contains 'liverpool';
```

This query results in some irrelevant tweets (e.g., tweets about American Football). In [7], we discuss how to remove noisy terms and rank tweets by relevance.

3.2 Timeline and Tweets

Once a user has created an event, TwitInfo creates a page on which the user can monitor the event. The TwitInfo interface (Figure 2) is a dashboard summarizing the event over time. The dashboard displays a timeline for the event, raw tweet text sampled from the event, an overview graph of tweet sentiment, and a map view displaying tweet sentiment and locations.

[4]The TwitInfo website with interactive visualizations is accessible at `http://twitinfo.csail.mit.edu/`

The event timeline (Figure 2.2) reports tweet activity by volume. The more tweets that match the query during a period of time, the higher the y-axis value on the timeline for that period. When many users are tweeting about a topic (e.g., a goal by Manchester City), the timeline spikes. TwitInfo's peak detection algorithm is implemented in a stateful TweeQL UDF described in Section 2.1.6. The algorithm identifies these spikes and flags them as peaks in the interface.

Peaks appear as flags in the timeline. TwitInfo automatically generates key terms that frequently appeared in tweets during a peak, and displays them to the right of the timeline. For example, in Figure 2.2, TwitInfo automatically tags one of the goals in the soccer game as peak "F" and annotates it on the right with representative terms in the tweets like '3-0' (the new score) and 'Tevez' (the soccer player who scored). Users can perform text search on this list of key terms to locate a specific peak.

As users click on peaks, the map, tweet list, links, and sentiment graph update to reflect tweets in the period covered by the peak.

The Relevant Tweets panel (Figure 2.4) contains the tweets that have the highest overlap with the event peak keywords. These tweets expand on the reason for the peak. The relevant tweets are color-coded red, blue, or white depending on whether the sentiment they display is negative, positive, or neutral.

3.3 Aggregate Metadata Views

A user may wish to see the general sentiment on Twitter about a given topic. The Overall Sentiment panel (Figure 2.6) displays a pie chart with the proportion of positive and negative tweets during an event.

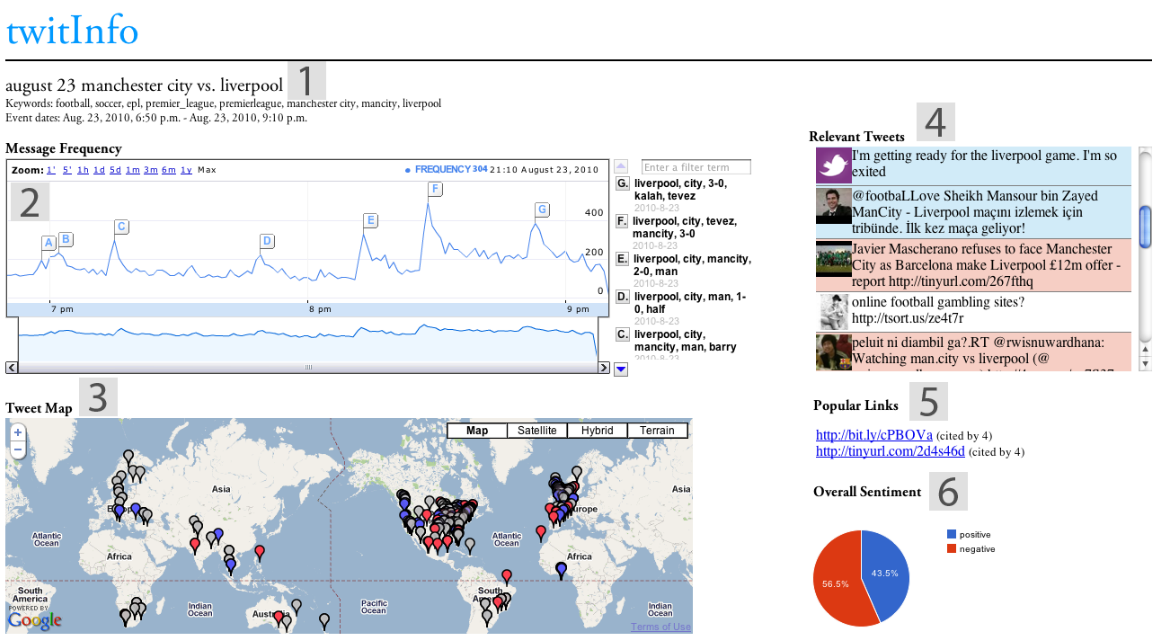
Twitter users share links as a story unfolds. The Popular Links panel (Figure 2.5) aggregates the top URLs extracted from tweets in the timeframe being explored.

Often, opinion on an event differs by geographic region. The Tweet Map (Figure 2.3) displays tweets that provide geolocation metadata. The marker for each tweet is colored according to its sentiment, and clicking on a pin reveals the associated tweet.

3.4 Uses and Study

As we developed TwitInfo, we tested its ability to identify meaningful events and its effectiveness at relaying extracted information to users.

We have tracked events of different duration and content using TwitInfo. In soccer matches, TwitInfo successfully identifies goals, half-time, the end of a game, and some penalties. The system successfully identified


twitInfo

august 23 manchester city vs. liverpool

Keywords: football, soccer, epl, premier_league, premierleague, manchester city, mancity, liverpool

Event dates: Aug. 23, 2010, 6:50 p.m. - Aug. 23, 2010, 9:10 p.m.

Message Frequency

G. liverpool, city, 3-0, kalah, tevez 2010-8-23

F. liverpool, city, tevez, mancity, 3-0 2010-8-23

E. liverpool, city, mancity, 2-0, man 2010-8-23

D. liverpool, city, man, 1-0, half 2010-8-23

C. liverpool, city, mancity, man, barry

Relevant Tweets

I'm getting ready for the liverpool game. I'm so exited

@footbaLLove Sheikh Mansour bin Zayed ManCity - Liverpool maçını izlemek için tribünde. İlk kez maça geliyor!

Javier Mascherano refuses to face Manchester City as Barcelona make Liverpool £12m offer - report http://tinyurl.com/267fthq

online football gambling sites? http://tsort.us/ze4t7r

peluit ni diambil ga?.RT @rwisnuwardhana: Watching man.city vs liverpool (@

Popular Links

http://bit.ly/cPBOVa (cited by 4)

http://tinyurl.com/2d4s46d (cited by 4)

Figure 2: The TwitInfo user interface summarizing a soccer game.

all major earthquakes over a 1-month timespan. Finally, we visualized sixteen days in Barack Obama's life and policymaking, and identified most newsmaking events in the interface. Examples of these visualizations can be found on the TwitInfo website.

We tested the TwitInfo interface with twelve users. We asked them to reconstruct either a soccer game or sixteen days in Barack Obama's life based solely on the TwitInfo user interface. Participants found the interface useful for such summaries, with one participant recounting in detail Obama's every activity over the timespan without having read any other news on the topic [7].

While users explained that TwitInfo provides them with a good summary of an event, they often described the summary as shallow. This is in part due to the short, fact-oriented nature of tweets.

A Pulitzer Prize-winning former Washington Post investigative reporter thought of two use-cases for TwitInfo in journalism. The first was in *backgrounding*: when a journalist starts researching a topic, it helps to have an overview of recent events of note. The second use was in finding eyewitnesses. While reporters are generally averse to trusting tweets at face value, a location-based view of tweets can help identify Twitter users that may have been at or near an event to follow up with in more detail.

4. CONCLUSION

Twitter offers a diverse source of timely facts and opinions. In order for the information in unstructured tweets to be useful, however, it must be tamed. We described two tools, TweeQL for programmers and TwitInfo for end-users, to make this information more accessible. More broadly, social streams offer the database community an opportunity to build systems for streaming, unstructured data, and social networks in the wild.

5. ACKNOWLEDGEMENTS

We thank Eugene Wu, who discussed the mechanics of stream creation and wrapping syntax.

6. REFERENCES

[1] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. Aurora: a new model and architecture for data stream management. *The VLDB Journal*, 2003.

[2] A. Arasu, S. Babu, and J. Widom. The CQL continuous query language: Semantic foundations and query execution. Technical Report 2003-67, Stanford InfoLab, 2003.

[3] R. Avnur and J. M. Hellerstein. Eddies: Continuously adaptive query processing. In *In SIGMOD 2000*.

[4] N. Diakopoulos, M. Naaman, and F. Kivran-Swaine. Diamonds in the rough: Social media visual analytics for journalistic inquiry. In *VAST*, 2010.

[5] R. Goldman and J. Widom. WSQ/DSQ: a practical approach for combined querying of databases and the web. *SIGMOD Record*, 2000.

[6] B. Hecht, L. Hong, B. Suh, and E. H. Chi. Tweets from justin bieber's heart: the dynamics of the location field in user profiles. CHI, 2011.

[7] A. Marcus, M. S. Bernstein, O. Badar, D. R. Karger, S. Madden, and R. C. Miller. Twitinfo: aggregating and visualizing microblogs for event exploration. CHI, 2011.

[8] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. Pig latin: a not-so-foreign language for data processing. SIGMOD, 2008.

[9] V. Raman and J. M. Hellerstein. Partial results for online query processing. In *SIGMOD*, 2002.

Jiawei Han Speaks Out
On data mining, privacy issues and managing students

by Marianne Winslett and Vanessa Braganholo

Jiawei Han
http://www.cs.uiuc.edu/~hanj/

Welcome to ACM SIGMOD Record's series of interviews with distinguished members of the database community. I'm Marianne Winslett, and today we are at the University of Illinois at Urbana-Champaign. I have here with me Jiawei Han, who is a professor of the Computer Science Department at the University of Illinois. Before joining Illinois, Jiawei was a professor at Simon Fraser University in Canada for many years, and briefly before that, he was a professor at Northwestern University. Jiawei's research interests lie in data mining. He is editor in chief of ACM Transactions on Knowledge Discovery from Data and he is the coauthor, with Michelline Kamber, of a popular textbook on data mining. Jiawei is an ACM Fellow and an IEEE Fellow. His PhD is from the University of Wisconsin at Madison. So, Jiawei, welcome!

Thank you! Thank you Marianne!

So Jiawei, according to Publish or Perish, you have an H-index of 76, which means that you have written 76 papers that have been cited 76 or more times. To put that in perspective, Publish *or Perish lists my own H-index as 27, and I'm pretty happy with that. But of course I'd be even happier if I were three times as productive and also had an H-index of 76. So tell us, what advice do you have for those of us who would like to be more influential?*

Actually, I myself think this is the first time I have heard about 76[1]! I remember when somebody told me there is an H-index… I went down there, I looked at it, it was around 54-55 something. I never knew I got 76. I think probably the best thing I can say is: if you choose a topic, choose something pretty exciting. Maybe it is tough, or maybe it may not be that tough, but I think, choose something which is a little fresh, a little meaningful, a little exciting, and try to find some good solutions. Probably, that is the best, because usually you get a paper you like people to read, and that you yourself like to work on. So if you got excited, it's likely that other people might get excited as well. Of course, not every paper can do things like that. Of course for me, if I got more papers published, I'd probably prefer every paper I write, just not write that many, but just write something more exciting. But since I am also supervising lots of students, you cannot expect every student here to write every paper not only exciting to the students themselves, but exciting to everybody. That's very hard. So we do have some papers, maybe a little incremental, that may not be so exciting. But I think overall, work with more exciting topics, and also try to work out some neat and elegant solutions, then probably people will feel more exciting to cite your paper.

So let's talk about data mining. Data mining is very important and popular today, but it is a very young field, with the first paper appearing only around 1992. What led you and the other KDD founders to start working in that new area?

So a KDD paper… If you say it appeared around 1992, it was probably in a database conference, like the SIGMOD or VLDB conferences. But even in other conferences, I probably can trace back to 1989, when Piatetsky-Shapiro organized the First International Workshop on Knowledge Discovery in Databases. That one I still remember because I sent a paper there, I attended the workshop. At that time it was a very small group, about 30 people. It was actually attached to IJCAI, that was in Detroit. At that time everybody felt this could be a big fish, a big direction. I myself also felt this way because I worked, as my PhD thesis, I actually worked on deductive databases. At that time, logic programming, and database was very hot.

I got a big influence by Randy Katz. Randy at that time was a professor in Wisconsin. He actually once gave a seminar. At that time, Japan got a 5th generation computer project. That was 1983 or 1982, I forgot. He actually, in the seminar, he even put a Japanese sword right on the table. He said it was a Japanese challenge. So he said the Japanese wanted to work out a Prolog machine, which is highly parallel, that can do a lot of database searching, inferencing. He said we needed to face the challenge. So I got a deep impression on that. That time also was the

[1] Editors' note: at the time this interview was published, his H-index was 101 (we used Publish or Perish with the search string "Jiawei Han" and area "Engineering, Computer Science, Mathematics").

time I selected my research topics. So, I got pretty excited, actually, once I got there. I did my thesis on that, published quite a few papers, including both database conferences or even logic programming conferences. So it is a rather different field, you will see resolution, you know, (Herbrand Universe), recursion, recursive query processing, compilation, all of those things probably in the other domain. And when I went to Simon Fraser, I was also interested in looking for good topics to extend the scope, because, I myself feel if we just rely on expert rules to derive knowledge, or new knowledge, it is far less efficient; you really need tools to dig through the data to get knowledge rather than just relying on expert rules. So that is the reason I am actually quite interested in integrating, like machine learning induction into the database as well. That's the part where we started.

We got a paper, I actually originally did not really know where to send, because we got the paper, and there were no such conferences, actually. Of course, we could send it to database conferences, but I did not really know whether people would like it. At that time, Gregory Piatetsky-Shapiro organized one. He just sent emails that said he was organizing the first international workshop on knowledge discovery in databases. I figured this one actually was a very good match of the algorithm we worked out. At that time we called this as attribute-oriented induction. It works in our databases going up and down, then you can derive some generalized knowledge. So we sent the paper down there, it was taken very quickly, because it was a workshop. That was the first paper we got down there, so then we did lots of improvements, and extension, and later we got it into VLDB 1992[2], that is the one you mentioned.

In 1992 the database conferences started taking data mining papers. But the interesting thing is, I think there were quite a few milestone topics on data mining before the formal, like the KDD conference formed up. Actually, almost all appeared in database conferences. It was very highly cited, very excited thing on this. I think the reason could be this: I believe the database people like to work out the algorithm, really working on very huge amounts of data, scalable algorithms, and also they really worry if it's effective, you get all the performance time. That is the reason you'll probably see, I can give you a few good examples… Of course, Rakesh Agrawal, their associational mining paper was probably the most cited paper even in the database conference history, and Raghu Ramakrishnan, their paper on BIRCH, that one is also on clustering algorithms... Even Johannes Gehrke, on the Rainforest algorithm. There are lots of very highly cited papers that actually appeared first in database conferences. So to that extent, I should say, data mining actually grew, of course, you could say it actually grew out from many different places, from machine learning, statistics, database, but database conferences really took a lot of very good papers, those are milestone papers in the KDD history.

What application areas for data mining are just around the corner?

[2] Jiawei Han, Yandong Cai, Nick Cercone: Knowledge Discovery in Databases: An Attribute-Oriented Approach. VLDB 1992: 547-559

Oh, data mining, I should say, can almost apply to anywhere. You probably can see it. For example on web search, people use, say a PageRank or HITS algorithm. Essentially, PageRank or HITS is also doing data mining, because if you found a page that was pointed out by other ones, it really carries semantics, carries importance. That is the reason you finally can find very interesting, very relevant pages. So, my feeling is, the first thing we should think data mining is invisible data mining. That probably is the most interesting but most effective mining method. There are people using it. You think about Amazon.com, they say "People buying this book also buy other books". They are using for example collaborative filtering algorithm or some other data mining algorithms. You think about Google, people search on the web, they are using some mining results. So those invisible data mining, even if they do not say they are really doing data mining, actually they are using this methodology. I think this is probably the most interesting thing to see. There will be many things coming out.

Would you think someday that we will have domain-independent approaches to data mining, like a unified discriminative ranking model that's independent of semantic issues, independent of the particular application? For example, if you see a customer behaving in an unusual manner, maybe that means fraud, maybe it means that might be a big spender, or maybe it's just noise. Do you see domain-independent approaches to basic mining tasks?

Actually, for data mining as a discipline, you want to work out some general principles. To that extent, you don't want to stick with very, very concrete, you know, say, my methods are just working on this particular problem. You want to be a little general, somewhat domain independent. But on the other hand, because of different kinds of data, you really need different methods. They are so different. For example, just mining sequences. The sequence on transaction databases, like shopping sequences, and sequences on biological data, like DNA sequences, biological sequences. or mining text sequences. could be very, very different, because they are looking for very different patterns. So if you say, my algorithm works on all kinds of sequences, probably it is good for nothing, you know, it really cannot find patterns! To that extent, I think we can say the algorithm is first tailored to this particular application. Then you work out a very effective algorithm, maybe you try to extend your scope, to work on other applications which could be somewhat domain independent.

I remember, actually, we worked out one method I think called CloSpan, that's the one. We first worked out PrefixSpan by Jian Pei with me at Simon Fraser, and Xifeng worked out this CloSpan algorithm. I remember one professor in Purdue, he or she took this algorithm, and actually tried to use it on biodata, and also found something interesting. And I remember in our CS 591 seminar just a few weeks ago, there were some Japanese researchers, they do, I think it's more like web log, or web blog mining. They first used our PrefixSpan. I did not actually even realized, they first used this, so to that extent, this one can be used in multiple domains. So I like things to be more domain independent, but I think for particular questions, for particular problems, we have to first focus to make it more specialized, to make it work, then think how to generalize it.

Does that mean, does data mining have any general principles, like databases do?

I think the problem in database and data mining could be a little different, especially if you think about relational databases. The data is more like really well structured, and for this structured data, you can work on like selection, join, query processing or transaction management, reasonably easily. Actually if we just work on highly structured data, there could be some algorithm you can transfer to different domains . But even that, people are looking for different patterns. For the same structured data, you may look at different kinds, whether you want to find clusters, you want to find you know like regression, or evolution. Since people are finding different things, likely those algorithms will be tailored, or for different applications, they could be rather different. That's why, some people dream we have on-the-shelf, data mining tools. You just download it, and every pattern will be there. At least, at this point, I'd say, it is not realistic. It may not be quite effective. So you have to know the domain better, and you really know what's the pattern you want to find. and what's the trick you can use your knowledge. I think that's, it is far from like just using a very simple language, like SQL or SQL mining, you can solve all the problems.

Ok! So, two people suggested that I ask you about the ethics of data mining. One person gave me ChoicePoint as one example. I looked up ChoicePoint on Wikipedia, and it says, "ChoicePoint [...] is a data aggregation company [...] that acts as a private intelligence service to government and industry. [...] ChoicePoint combines personal data sourced from multiple public and private databases for sale to the government and the private sector. The firm maintains more than 17 billion records of individuals and businesses, which it sells to an estimated 100,000 clients [...] However, this data has not been secured sufficiently to prevent theft of data on at least one occasion. [...] The company has also been the subject of lawsuits for maintaining inaccurate data, inquiries whether it allowed political bias to influence its performance of government contracts and accused of illegally selling the data of overseas citizens to the US Government." So of course there will always going to be mistakes and inaccuracies in mined information, and there will always be some people who are greedy or corruptible. If we look forward to the future, how can we address these problems for data mining?

Actually, I read a lot of newspapers or some different controversial things on data mining, so the first thing I should say is, for any research, for example, when you apply for an NSF grant, they will ask you "are dealing with human subjects or non-human subjects?". Data mining is actually dealing with both things. A lot of data mining things are not dealing with human subjects at all. For example, if you try to mine some astronomy pictures (like what Jim Gray did, it was astronomy databases), you still need a lot of data mining. You probably will never worry about disclosing any stars privacy. So to that extent, there's no real privacy issue. Actually data could be public to anywhere in the world, anybody can share. So there are lots of such data mining tasks which we do not have to worry about the privacy.

But on the other hand, there are human subjects. For example, you mine data related to the people. So definitely once we get into this one, we have to think about privacy and security, all those issues. What I feel is, with data mining, usually you can think in two ways to do data mining. One way is, you got an in-house data mining software, you mine by yourself. For example, Wal-Mart can have some data miners, they sit in the Wal-Mart database, they do all the data mining. Then, they may have a choice, what they should publish, what they do not publish. Even for the in-house data mining, there could be issues whether you could look at any personal data or not. So as long as you have some appropriate measures for in-house data mining, for example, you say, I mined a customer record. Even you can take, say a particular credit card number linked with a previous same credit card number, and you find the shopping sequences. As long as you say, this employee or this data miner has no way or is not permitted to look for further links from this number, then you treat this number as a dummy number, like RFID, so you finally find something. That way, based on my viewpoint, you still haven't violated anybody's privacy yet. But the problem mainly is, what things you can publish? For example, if you publish things in a more statistic term, for example, you see US Statistic Bureau. They regularly publish lots of things. You can buy a CD-ROM, and you can have many years of data. You can go down to zip code, but zip code is still quite big. In most cases, one zip code may cover say thousand, or tens of thousands of people. If you say K-anonymity, this K could be ten thousand. And there is no way you could build links, so you publish this kind of data is still safe.

Based on my view, if you do in-house data mining, and then you carefully publish your data, which make your K or make all these privacy preserving things quite big, you are reasonably safe. You still can use this data. I think many people are using it, like US Statistic Bureau, they publish lots of data, lots of people are using it. I believe these things are necessary because an administrator like Obama wants to know some concrete statistics. Anybody who makes decisions need to base it on your data. So for those kinds of privacy, if you make this K quite big, you should not worry too much.

But you can charge more for your products if you make K smaller.

Yes, that is exactly that. You get a little dangerous then. If you make it too small, people start identifying something sensitive, something that may really violate people's privacy. That's exactly why privacy preserving publishing and data mining actually becomes a very important topic because people want to do both, want to find deeper information, but in the meantime, protect people's privacy, and these two could be conflicting goals. But another very important thing is (some people discussing about this) out-of-source data mining. I feel this out-of-source data mining could be a little dangerous.

What is out of source data mining?

That means that you ship your data to other people to mine it. Then, the other people mine it, and you don't want this miner to dig up more information than you want. This thing, my feeling is, it is very easy to go out of control. There are lots of research papers. My feeling is, some research papers say you can do k-anonymity, do l-diversity, do t-closeness, doing all these, and there is one professor from UT Austin who actually showed: if you very secure to guarantee this, then you do data mining, probably then you cannot really find good patterns. Actually, it could be even worse than the intruders or something. So I think this could be true. To completely make your data like have no characteristics, then ship it to other people to mine, you probably won't really find very interesting patterns. But on the hand, if you ship the raw data or some data that really contains sensitive information to a third party to mine, I don't feel it is a good idea.

One person commented that startups are much easier to do today, and wanted to know when you plan to start a company, and what your product will be.

The first thing, about the startup and whether it is easy to do or it's hard to do: different people may have different opinions or experience. My feeling is this: there are, of course, people working on database or data mining, a very practical domain. There could be lots of applications. Those applications may promote lots of startups. Whether researchers need to do startup or some other people may take the idea to do a startup, that completely, different people may have different opinions. For me, I actually like to concentrate on research. That's the reason I'm not that interested to set up a startup or something. For research-wise, I already think I got quite exhausted. If I do startup, I probably would have no time to sleep!

Have you found any challenges in your graduate students being distracted by industry or by startups?

I think for the students, actually being in UIUC at Urbana-Champaign is much better than in a big company or in a big city. For example, somewhere really in the center of the bay area or in Seattle or some places. Those places may attract the students in a very easy way because they just give a phone call, they just ride a bike, or drive a car, you can go there. I believe here, I do not feel the students here really are distracted by those companies. To a certain extent, it is good to learn something about a company's needs, to see the real world. I encourage students to go out to do summer intern, especially go to a real company, go to research labs, to do summer interns. I feel this is a very good practice, because you learn something about outside world, about applications, about industry, about research labs. When you come back, you probably have different research problems, different ideas, you build up your social network, research network. I think these all will help students.

With industry and academia working so closely now, what guidelines do you recommend for young graduates to choose between working in academia and industry?

I think students have different thinking, different preferences. Some students really like to go to industry. I got one student, Zheng Shao, who is very smart. Unfortunately, he did not finish the PhD. In the middle, he actually was attracted by Yahoo! first. He left, he went down there, he did very well, he is very successful. He actually came here I believe a few weeks ago doing some recruiting. I think the students really liked to go to industry. I still like them to finish PhD, because once you get better knowledge, and you master the research area, and also you get into PhD, likely you get into a little more like a research or development, or more invention or some kind of position where you use your talent better. To that extent, I encourage every student to finish PhD before going outside.

But on the other hand, I know there are lots of students that really want to do research. I told them for doing research, basically you have two major choices, one is going to university, because doing research in a university is not just doing teaching, you actually really dash forward, by working with graduate students you can do a lot of very interesting research. And another one is actually is good industry research labs, like IBM Research, Microsoft Research, Yahoo! Research, Google Research. There are lots of such research institutes or research labs. I think those research centers are very exciting as well because they got a lot of people, they all have PhDs, with different talents, and they are usually really good. They work together. And they also can work with real industry, and work with a lot of professors who may go down there for sabbatical or for doing some joint work. You really can, to some extent, extend your scope. For example, like Xifeng Yan. He did probably two or three years at IBM Research as a researcher, then recently he joined UC Santa Barbara, actually as a Chair Assistant Professor. He got very good training on both academia and research labs.

You were a young person in China during the Cultural Revolution, and then suddenly you were a graduate student in computer science at Wisconsin! How did you make that transformation?

I think this could be a pretty heavy topic! So, it is true, at that time it was a very unusual transition, I should say. Not only I was young when the Cultural Revolution broke out, it was a pretty dark time, in the sense, my family was intellectual, so I was almost at the bottom of society to some extent. I labored in the countryside for quite a few years. It was not easy. Actually, not only me, the whole country, to a certain extent,,the university was closed for almost 12 years: 1966 to 1978. For me, myself, of course, it was not easy.

Probably the really breaking point was in early 1978. China restored the Graduate study, Graduate School. I was bold enough to try the Chinese Academy of Sciences, and I passed the exam, and also the English one. So, I probably could say 1979 was the first year China and the US got diplomatic relationship, and I went to the University of Wisconsin in 1979. So I was very fortunate, but also it was not easy. It was interesting because those many years, China, I should say, almost closed doors for 30 years. That was the first time actually people even went to campuses to see the students from China. Even Wisconsin I think was bold enough to take those students. The reason was at China at that time there was no TOEFL, no GRE. There was no exam system at all. But I remember that a University of Wisconsin Professor told me they were bold enough because they saw China was a very big country, and it is a very big country as well, and there must be many talented people, Actually, one professor who wrote recommendation letters for me, got a PhD in UIUC, I should say. But also very unfortunately, he was labeled as a rightist and was almost forbidden to do any research work for 25 years. But he did write a letter for me and for Hongjun Lu. At that time, when we first came, Wisconsin actually first put these several students as international special. not formal graduate students, because they could not judge us. But they said, since the professors said these are the very best selected students, got into the Chinese Academy of Sciences graduate school, so they must be good. So they just blindly believed that. They took us, and we easily passed the first year, so then finally we got through that.

Actually, in those several years, there were many such cases. I remember YY's (Yuanyuan Zhou) advisor, Kai Li. YY actually invited him to come to give a talk, and she also invited both me and a few other professors to go down to her house. When Kai met me, he was joking, he said, "You know, we were classmates together with Ming Li," he says, "You see, that year, that two years, probably now only a few Chinese came out. Among probably four or five Chinese ACM fellows, we 3 are from the same class." That simply says, even we three people, it was just a tough competition, I probably should say, when you finally got there. I just say, China opened up, brought really a lot of changes for China itself, but also for lots of students.

You spent your sabbatical year (2007-8) in your office doing research and writing proposals. Why not get away and do something different instead?

I think with so many students working with me, it is a little hard—you are out of students' scope, going out for a very long time. And also there are so many things waiting for me to get done. I think for that sabbatical I did a lot of things on not only research but also enhanced my book, there were lots of other things. So, I just have very little time, hopefully in the future, I may get a chance.

I had a question suggestion I found very amusing. Are you ready?

Yes*!*

Which of the following would you find the most rewarding: (a) writing a landmark book in the field; (b) graduating lots of outstanding PhDs; (c) coming up with your own good algorithms; or (d) winning lots of prestigious awards?

That's a little hard to say, but I probably should say, training a lot of outstanding graduate students, that is probably the most exciting thing. The reason is, not only you train a lot of students, but also they will become the new seed for the whole field. If you don't train a good number of students, the whole field problems really cannot be solved by a small number of people. So I think that is really the most exciting thing.

How many graduate students, postdocs, and visitors do you have right now?

Okay, so, I think I can't roughly count, maybe. It depends on how we count it. I think I have 17 PhD students, and I have 2 master students, 1 visiting scholar so far, and 1 visiting student.

So, that's a great leading for my next question! Someone told me, "I am amazed by Jiawei's time management skills and his enthusiasm in research. Even with his large group, he still answers most students' emails within hours, no matter whether it is in the morning, noon or midnight." So, how do you do that, with so many?

I think, the first thing, for students supervision, with many students, there are disadvantages and advantages. The challenge could be how could you handle every student because everyone is different. But from my viewpoint, you should not do everything just by yourself. The students can work together. Another thing is, the students can organize by themselves as well. So in many cases, no matter in Canada or in US, I feel there are always, there are some students, who really have the talent, who can lead, who can be the future, who will probably be a leader or professor, so they can really organize things. You should really let those students to play some role. In the meantime, I meet many students actually in groups. To some extent, these groups are dynamic, in the sense that, once we have some research topic, sometimes I send an email and say "who is interested in these topics?". There will always be a few people volunteering, and some are very enthusiastic. I will ask those enthusiastic students to lead, and then we form research groups. We finish this, or even before finishing this, there could be some new topics, some student, very energetic can join 3 or 4 different research groups. I think this probably can really help reduce my load.

But for answering email so quick, I think the first thing is I try to, sometimes try to forget, otherwise email will pile up, and I have to read it again and again. It may not be a very good habit. I think Johannes Gehrke actually, I remember once in SIGMOD Record he wrote an article, he said he tried not to be distracted by emails. He tried to pile up the email until a certain time, like 3 o'clock. I think that could be a good habit, because you really can concentrate more on your research. Sometimes I could be distracted.

Do you have any words of advice for fledgling or midcareer database researchers or practitioners?

Yeah! I think one thing probably is choosing promising and good research topics. Usually, I should say the midcareer researcher should be bold enough to challenge some new things. That's the one, see, I started working on data mining to some extent to challenge myself. If I feel this is a very good topic, I would like to jump in. I think for midcareer researcher, especially if you already got tenure, you really should be bold enough to find something you think is challenging, is exciting, then you jump in. Of course, in the meantime, if you originally have a very good background in certain things, you may also like to keep going. Sometimes you make a complete swap, but sometimes you may swap back, but by doing something new, I think it will always give you some more credit and also more capability.

Among all your past research, do you have a favorite piece of work?

Oh, of course, something that got you excited, you feel is interesting even no matter whether other people got the same excitement or not? For example, in the early days, when I was doing deductive database, I got this, like a recursion, a very irregular recursion compilation, and I can solve, for example, the N-Queen problem in a rather declarative way, and I got very excited about that. But of course, you know, there are lots of new research topics you can work on. Like the first piece of work we did, this attribute-oriented induction. I also got pretty excited, at least at the time. Then the later ones… I probably believe we got this pattern growth method to solve like frequent patterns, sequential patterns, graph patterns, I believe those are pretty exciting things. And also those things we can see from the citation.

If you magically had enough extra time to do one additional thing at work that you are not doing now, what would it be?

You mean outside of the research, altogether, or something related to research?

It could be either way.

Actually, if you say out of research, I actually quite like to travel, or climbing mountains, those things when I was young, I was quite energetic on those things.

Oh, I see, so when you said research or not research, I was imagining by not research you meant like you would volunteer to be the chair of SIGKDD or something, but you meant really outside the research, outside of work entirely! So you climb mountains!

Yes! But on the other hand, for research, I think you always try to find something exciting to work on. Quite often, I like to read for example, Scientific American. I always think there are a lot of different research topics you would love to know, and also you may try it. For example, in a lot of research, sometimes ideas you get are from those readings. You feel "oh, why this biologist can do this, why can't I do something similar?". I think a lot of research or knowledge can crossbreed.

If you could change one thing about yourself as a computer science researcher, what would it be?

Actually, in my early days, of course during the Cultural Revolution, it really broke out my dream. I actually would like to be a physicist, but I've never been able to get a chance. But I think it is interesting to read those things. But that is a different thing, I think once I got into computer science, I really love it.

Great! Thank you very much for talking with me today.

Thank you very much!

The Database Architectures Research Group at CWI

Martin Kersten Stefan Manegold Sjoerd Mullender
CWI
Amsterdam, The Netherlands
first.last@cwi.nl
http://www.cwi.nl/research-groups/Database-Architectures/

1. INTRODUCTION

The Database research group at CWI was established in 1985. It has steadily grown from two PhD students to a group of 17 people ultimo 2011. The group is supported by a scientific programmer and a system engineer to keep our machines running. In this short note, we look back at our past and highlight the multitude of topics being addressed.

2. THE MONETDB ANTHOLOGY

The workhorse and focal point for our research is *MonetDB*, an open-source columnar database system. Its development goes back as far as the early eighties when our first relational kernel, called *Troll*, was shipped as an open-source product. It spread to ca. 1000 sites world-wide and became part of a software case-tool until the beginning of the nineties. None of the code of this system has survived, but ideas and experiences on how to obtain a fast relational kernel by simplification and explicit materialization found their origin during this period.

The second half of the eighties was spent on building the first distributed main-memory database system in the context of the national *Prisma* project. A fully functional system of 100 processors and a, for that time, wealthy 1 GB of main memory showed the road to develop database technology from a different perspective. Design from the processor to the slow disk, rather than the other way around.

Immediately after the Prisma project, a new kernel based on Binary Association Tables (*BATs*) was laid out. This storage engine became accessible through *MIL*, a scripting language intended as a target for compiling SQL queries. The target application domain was to better support scientific databases with their (archaic) file structures. It quickly shifted to a more urgent and emerging area.

Several datamining projects called for better database support. It culminated in our first spin-off company, Data Distilleries, in 1995, which based their analytical customer relationship suite on the power provided by the early MonetDB implementations. In the years following, many technical innovations were paired with strong industrial maturing of the software base. Data Distilleries became a subsidiary of SPSS in 2003, which in turn was acquired by IBM in 2009.

Moving MonetDB Version 4 into the open-source domain required a large number of extensions to the code base. It became of the utmost importance to support a mature implementation of the SQL-03 standard, and the bulk of application programming interfaces (PHP, JDBC, Python, Perl, ODBC, RoR). The result of this activity was the first official open-source release in 2004. A very strong XQuery front-end was developed with partners and released in 2005 [1].

MonetDB remains a product well-supported by the group. All its members carry out part of the development and maintenance work, handling user inquiries, or act as guinea pigs of the newly added features. A thorough daily regression testing infrastructure ensures that changes applied to the code base survive an attack of ca. 20 platform configurations, including several Linux flavors, Windows, FreeBSD, Solaris, and MacOS X. A monthly bugfix release and ca. 3 feature releases per year support our ever growing user community. The web portal[1] provides access to this treasure chest of modern database technology. It all helped us to create and maintain a stable platform for innovative research directions, as summarized below. The MonetDB spin-off company was set up to support its market take-up, to provide a foundation for QA, support, and development activities that are hard to justify in a research institute on an ongoing basis.

3. HARDWARE-CONSCIOUS DATABASE TECHNOLOGY

A key innovation in the MonetDB code base is its reliance on hardware conscious algorithms. For,

[1] `http://www.monetdb.org/`

advances in speed of commodity CPUs have far outpaced advances in RAM latency. Main-memory access has therefore become a performance bottleneck for many computer applications, including database management systems; a phenomenon widely known as the "memory wall." A revolutionary redesign of database architecture was called for in order to take advantage of modern hardware, and in particular to avoid hitting this memory wall. Our pioneering research on columnar and hardware-aware database technology, as materialized in MonetDB, is widely recognized, as indicated by the VLDB 2009 10-year Best Paper Award [19, 2] and two DaMoN best paper awards [22, 6]. Here, we briefly highlight important milestones.

Vertical Storage. Whereas traditionally, relational database systems store data in a row-wise fashion (which favors single record lookups), MonetDB uses a columnar storage, which favors analysis queries by better using CPU cache lines.

Bulk Query Algebra. Much like the CISC vs. RISC idea applied to CPU design, the MonetDB relational algebra is deliberately simplified compared to the traditional relational set algebra. Paired with an operator-at-a-time bulk execution model, rather than the traditional tuple-at-a-time pipelining model, this allows for much faster implementation on modern hardware, as the code requires far fewer function calls and conditional branches.

Cache-conscious Algorithms. The crucial aspect to overcome the memory wall is good use of CPU caches, for which careful tuning of memory access patterns is needed. This led to a new breed of query processing algorithms. Their key requirement is to restrict any random data access pattern to data regions that fit into the CPU caches to avoid cache misses, and thus, performance degradation. For instance, *partitioned hash-join* [2] first partitions both relations into H separate clusters that each fit into the CPU caches. The join is then performed per pair of matching clusters, building and probing the hash-table on the inner relation entirely inside the CPU cache. With large relations and small CPU caches, efficiently creating a large number of clusters can become a problem in itself. If H exceeds the number of TLB entries or cache lines, each memory reference will trigger a TLB or cache miss, compromising the performance significantly. With *radix-cluster* [17], we prevent that problem by performing the clustering in multiple passes, such that each pass creates at most as many new sub-clusters as there are TLB entries or cache lines. With *radix-decluster* [18], we complement partitioned hash-join with a projection (tuple reconstruction) algorithm with a cache-friendly data access pattern.

Memory Access Cost Modeling. For query optimization to work in a cache-conscious environment, and to enable automatic tuning of our cache-conscious algorithms on different types of hardware, we developed a methodology for creating cost models that takes the cost of memory access into account [16]. The key idea is to abstract data structures as *data regions* and model the complex data access patterns of database algorithms in terms of simple compounds of a *few basic data access patterns*. We developed cost functions to estimate the cache misses for each basic pattern, and rules to combine basic cost functions and derive the cost functions of arbitrarily complex patterns. The total cost is then the number of cache misses multiplied by their latency. In order to work on diverse computer architectures, these models are parametrized at run-time using automatic *calibration* techniques.

Vectorized Execution. In the "X100" project, we explored a compromise between classical tuple-at-a-time pipelining and operator-at-a-time bulk processing [3]. The idea of vectorized execution is to operate on chunks (vectors) of data that are large enough to amortize function call overheads, but small enough to fit in CPU caches to avoid materialization into main memory. Combined with just-in-time light-weight compression, it lowers the memory wall somewhat. The X100 project has been commercialized into the Actian/VectorWise company and product line [2].

4. DISTRIBUTED PROCESSING

After more than a decade of rest at the frontier of distributed database processing, we embarked upon several innovative projects in this area again.

Armada. An adventurous project was called Armada where we searched for technology to create a fully autonomous and self regulating distributed database system [5]. The research hypothesis was to organize a large collection of database instances around a dynamically partitioned database. Each time an instance ran out of resources, it could solicit a spare machine and decide autonomously on what portion to delegate to its peer. The decisions were reflected in the SQL catalog which triggered continuous adaptive query modification to hunt after the portions in the loosely connected network of workers. It never matured as part of the MonetDB distribution, because at that time we did not have all the basic tools to let it fly.

Since, the Merovingian toolkit developed and now provides the basis for massive distributed process-

[2] `http://www.actian.com/products/vectorwise/`

ing. It provides server administration, server discovery features, client proxying and funneling to accommodate large numbers of (web) clients, basic distributed (multiplex) query processing, and fail-over functionality for a large number of MonetDB servers in a network. It is the toolkit used by partner companies to build distributed datawarehouse solutions. With Merovingian we were able to open two new research tracks: DataCyclotron and Octopus. Our new machine cluster[3] provides a basis to explore both routes in depth.

DataCyclotron. The grand challenge of distributed query processing is to devise a self-organizing architecture which exploits all hardware resources optimally to manage the database hot-set, to minimize query response time, and to maximize throughput without single point global coordination. The Data Cyclotron architecture [4] addresses this challenge using turbulent data movement through a storage ring built from distributed main memory and capitalizing on the functionality offered by modern remote-DMA network facilities. Queries assigned to individual nodes interact with the storage ring by picking up data fragments that are continuously flowing around, i.e., the hot-set.

The storage ring is steered by the *level of interest* (*LOI*) attached to each data fragment. The LOI represents the cumulative query interest as it passes around the ring multiple times. A fragment with *LOI* below a given threshold, inversely proportional to the ring load, is pulled out to free up resources. This threshold is dynamically adjusted in a fully distributed manner based on ring characteristics and locally observed query behavior. It optimizes resource utilization by keeping the average data access latency low. The approach is illustrated using an extensive and validated simulation study. The results underpin the fragment hot-set management robustness in turbulent workload scenarios.

A fully functional prototype of the proposed architecture has been implemented using modest extensions to MonetDB and runs within a multi-rack cluster equipped with Infiniband. Extensive experimentation using both micro benchmarks and high-volume workloads based on TPC-H demonstrates its feasibility. The Data Cyclotron architecture and experiments open a new vista for modern in-the-network distributed database architectures with a plethora of research challenges.

Octopus. In the Octopus project, we deviate from the predominant approach in distributed database processing, where the data is spread across a number of machines one way or another before any query processing can take place. We start from a single master node in control of the database, and with a variable number of worker nodes to be used for delegated query processing. Data is shipped just-in-time to the worker nodes using a need-to-know policy, and reused, if possible, in subsequent queries. A bidding mechanism among the workers yields the most efficient reuse of parts of the original data, available on the workers from previous queries.

The adaptive distributed architecture uses the master/workers paradigm: the master hosts the database and computes a query by generating distributed subqueries for as many workers as it has currently available. The workers recycle the data they have processed in the past as much as possible to minimize the data transfer costs. Due to the just-in-time replication, the system easily harvests non-dedicated computational resources, while supporting full SQL query expressiveness.

Our experiments show that the proposed adaptive distributed architecture is a viable and flexible approach for improving the query performance of a dedicated database server by using non-dedicated worker nodes, reaching benefits comparable to traditional distributed databases.

5. ADAPTIVE INDEXING

Query performance strongly depends on finding an execution plan that touches as few superfluous tuples as possible. The access structures deployed for this purpose, however, are non-discriminative. They assume every subset of the domain being indexed is equally important, and their structures cause a high maintenance overhead during updates. Moreover, while hard in general, the task of finding the optimal set of indices becomes virtually impossible in scenarios with unpredictable workloads.

With **Database Cracking**, we take a completely different approach. Database cracking combines features of automatic index selection and partial indexes. Instead of requiring a priori workload knowledge to build entire indices prior to query processing, it takes each query predicate as a hint how to physically reorganize the data. Continuous physical data reorganization is performed on-the-fly during query processing, integrated in the query operators. When a column is queried by a predicate for the first time, a new cracker index is initialized. As the column is used in the predicates of further queries, the cracker index is refined by range partitioning until sequentially searching a partition is faster than binary searching in the AVL tree guiding a search to the appropriate partition.

[3]`http://www.scilens.org/platform/`

Keys in a cracker index are partitioned into disjoint key ranges, but left unsorted within each partition. Each range query analyzes the cracker index, scans key ranges that fall entirely within the query range, and uses the two end points of the query range to further partition the appropriate two key ranges. Thus, in most cases, each partitioning step creates two new sub-partitions using logic similar to partitioning in quicksort. A range is partitioned into 3 sub-partitions if both end points fall into the same key range. This happens in the first partitioning step in a cracker index (because there is only one key range encompassing all key values) but is unlikely thereafter [7].

Updates and their efficient integration into the data structure are covered in [8]. Multi-column indexes to support selections, tuple reconstructions and general complex queries are covered in [9]. In addition, [9] supports partial materialization and adaptive space management via partial cracking.

While database cracking comes with very low overhead but slow convergence towards a fully optimized index, *adaptive merging* features faster convergence at the expense of a significantly higher overhead. **Hybrid adaptive indexing** aims at achieving a faster convergence while keeping the overhead low as with database cracking [10].

With **stochastic cracking**, we introduce a significantly more resilient approach to adaptive indexing. Stochastic cracking does use each query as advice on how to reorganize data, but not blindly so; it gains in resilience and avoids performance bottlenecks by allowing for lax and arbitrary choices in its decision-making. Thereby, we bring adaptive indexing forward to a mature formulation that confers the workload-robustness previous approaches lacked.

Ongoing work aims at combining adaptive indexing techniques with the ideas of physical design and auto-tuning tools. The goal is to exploit workload knowledge to steer adaptive indexing where possible, but keep the flexibility and instant adaptation to changing workloads of adaptive indexing.

6. SCIENTIFIC DATABASES

After the first open-source release of MonetDB, we were keen to check its behavior on real-life examples beyond the classical benchmarks. The largest, well-documented and publicly available datawarehouse was the Sloan Digital Sky Survey (SDSS) / SkyServer. Embarking on its re-implementation was a challenge. None of the other DBMSs had accomplished a working implementation, either due to its complexity or lack of resources (business drive).

Skyserver. We achieved a fully functional implementation of SkyServer [4]. It proved that the column store approach of MonetDB has a great potential in the world of scientific databases. However, the application also challenged the functionality of our implementation and revealed that a fully operational SQL environment is needed, e.g., including persistent stored modules. Its initial performance was competitive to the reference platform, Microsoft SQL Server 2005, and the analysis of SDSS query traces hinted at several techniques to boost performance by utilizing repetitive behavior and zoom-in/zoom-out access patterns that were not captured by the system.

Recycler. An immediate follow up project focused on developing a *recycler* component to MonetDB. It acts as an intelligent cache of all intermediate results. Avoiding recomputing of any subquery as often as possible, within the confines of the storage set aside for the intermediates. The results were published in 2009 at SIGMOD and received the runner up best paper award [11].

Recycling can be considered an adaptive materialized view scheme. Any subquery can be re-used, there is no a priori decision needed by a human DBA. It is also more effective than recycling only the final query result sets. Integration of the recycler with the SDSS application showed that a few materialized views had been forgotten in the original design, which would have improved throughput significantly. This was found without human intervention.

SciBORQ. Scientific discovery has shifted from being an exercise of theory and computation, to become the exploration of an ocean of observational data. This transformation was identified by Jim Gray as the 4th paradigm of scientific discovery. State-of-the-art observatories, digital sensors, and modern scientific instruments produce Petabytes of information every day. This scientific data is stored in massive data centers for later analysis. But even from the data management viewpoint, the capture, curating, and analysis of data is not a computation intensive process any more, but a data intensive one. The explosion in the amount of scientific data presents a new "stress test" for database design. Meanwhile, the scientists are confronted with new questions, how can relevant and compact information be found from such a flood of data?

Data warehouses underlying Virtual Observatories stress the capabilities of database management systems in many ways. They are filled on a daily basis with gigabytes of factual information, derived from large data scrubbing and computational in-

[4]see `http://www.scilens.org/`

tensive feature extraction pipelines. The predominant data processing techniques focus on massive parallel loads and map-reduce algorithms. Such a brute force approach, albeit effective in many cases, is costly.

In the SciBORQ project, we explore a different route [21]. One based on the knowledge that only a small fraction of the data is of real value for any specific task. This fraction becomes the focus of scientific reflection through an iterative process of adhoc query refinement. However, querying a multiterabyte database requires a sizable computing cluster, while ideally the initial investigation should run on the scientist's laptop.

We work on strategies on how to make biased *snapshots* of a science warehouse such that data exploration can be instigated using precise control over all resources. These snapshots, constructed with novel sampling techniques, are called *impressions*. An impression is selected such that either the statistical error of a query answer remains low, or an answer can be produced within strict time bounds. Impressions differ from previous sampling approaches because of their *bias* towards the focal point of the scientist's data exploration.

7. STREAMING

DataCell. Streaming applications have been en vogue for over a decade now and continuous query processing has emerged as a promising paradigm with numerous applications. A more recent development is the need to handle both streaming queries and typical one-time queries in the same application setting, e.g., complex event processing (CEP). For example, data warehousing can greatly benefit from the integration of stream semantics, i.e., online analysis of incoming data and combination with existing data. This is especially useful to provide low latency in data intensive analysis in big data warehouses that are augmented with new data on a daily basis.

However, state-of-the-art database technology cannot handle streams efficiently due to their "continuous" nature. At the same time, state-of-the-art stream technology is purely focused on stream applications. The research efforts are mostly geared towards the creation of specialized stream management systems built with a different philosophy than a DBMS. The drawback of this approach is the limited opportunities to exploit successful past data processing technology, e.g., query optimization techniques.

For this new problem we combine the best of both worlds. In the DataCell project [14] we take a different route by designing a stream engine on top of an existing relational database kernel [15]. This includes reuse of both its storage/execution engine and its optimizer infrastructure. The major challenge then becomes the efficient support for specialized stream features.

We focus on incremental window-based processing, arguably the most crucial stream-specific requirement. In order to maintain and reuse the generic storage and execution model of the DBMS, we elevate the problem to the query plan level. Proper optimizer rules, scheduling and intermediate result caching and reuse, allow us to modify the DBMS query plans for efficient incremental processing. In extensive experiments, DataCell demonstrates efficient performance even compared to specialized stream engines, especially when scalability becomes a crucial factor.

8. GRAPH DATABASES

As database kernel hackers we can not escape the semantic web wave. RDF and triple stores requirements are also challenging the MonetDB kernel. In a recent paper [20], we showed how existing database technology can provide a sound basis for these environments. The base performance of MonetDB for graph database is superb, but perhaps we may find novel tricks when a complete SPARQL front-end emerges on top of it. Most likely, we can re-use many of the techniques developed in the context of MonetDB/XQuery, in particular run-time query optimization [12]. Nevertheless, we did not chicken out and got ourselves lured into European development projects to promote Linked-Open-Data. A step towards this goal is to carve out a benchmark that would shed light on the requirements in this field.

9. FUTURE

Despite the broad portfolio of topics, there is a strong drive and interest in pushing the boundaries of our knowledge by seeking areas hitherto unexplored. The mission for the future is to seek solutions where the DBMS *interpret queries by their intent*, rather than as a contract carved in stone for complete and correct answers. The result set should aid the user in understanding the database's content and provide guidance to continue his data exploration journey. A scientist can stepwise explore deeper and deeper into the database, and stop when the result content and quality reaches his satisfaction point. At the same time, response times should be close to instant such that they allow a scientist to *interact* with the system and explore the data in a contextualized way.

In our recent VLDB 2011 Challenges & Visions paper [13], we chartered a route for such groundbreaking database research along five dimensions:

- One-minute DBMS for real-time performance.
- Multi-scale query processing.
- Post processing for conveying meaningful data.
- Query morphing to adjust for proximity results.
- Query alternatives for lack of providence.

Each direction would serve several PhDs and produce a database system with little resemblance to what we have built over the last thirty years. We look forward to seeing members of the database research community join our mission and take up the challenges expressed.

10. ACKNOWLEDGMENTS

This research was made possible with the dedication of the 2011 research team comprised of Niels Nes, Peter Boncz, Fabian Groffen, Ying Zhang, Stratos Idreos, Milena Ivanova, Romulo Goncavles, Erietta Liarou, Lefteris Sidirourgos, Arjen de Rijke, Holger Pirk, Phan Minh Duc, Bart Scheers, Eleni Petraki, Thibault Sellam, Yagiz Kargin.

11. REFERENCES

[1] P. A. Boncz, T. Grust, M. van Keulen, S. Manegold, J. Rittinger, and J. Teubner. MonetDB/XQuery: A Fast XQuery Processor Powered by a Relational Engine. In *Proc. of the ACM Int'l Conf. on Management of Data (SIGMOD)*, pages 479–490, June 2006.

[2] P. A. Boncz, S. Manegold, and M. L. Kersten. Database Architecture Optimized for the New Bottleneck: Memory Access. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB)*, pages 54–65, Sept. 1999.

[3] P. A. Boncz, M. Zukowski, and N. Nes. MonetDB/X100: Hyper-Pipelining Query Execution. In *Proc. of the Int'l Conf. on Innovative Data Systems Research (CIDR)*, pages 225–237, Jan. 2005.

[4] R. Goncalves and M. L. Kersten. The Data Cyclotron Query Processing Scheme. In *Proc. of the Int'l Conf. on Extending Database Technology (EDBT)*, pages 75–86, Mar. 2010.

[5] F. Groffen, M. L. Kersten, and S. Manegold. Armada: a Reference Model for an Evolving Database System. In *Proc. of the GI-Fachtagung Datenbanksysteme in Business, Technologie und Web (BTW)*, pages 417–435, Mar. 2007.

[6] S. Héman, N. Nes, M. Zukowski, and P. A. Boncz. Vectorized data processing on the cell broadband engine. In *Proc. of the Int'l Workshop on Data Management on New Hardware (DaMoN)*, page 4, June 2007.

[7] S. Idreos, M. L. Kersten, and S. Manegold. Database Cracking. In *Proc. of the Int'l Conf. on Innovative Data Systems Research (CIDR)*, pages 68–78, Jan. 2007.

[8] S. Idreos, M. L. Kersten, and S. Manegold. Updating a Cracked Database. In *Proc. of the ACM Int'l Conf. on Management of Data (SIGMOD)*, pages 413–424, June 2007.

[9] S. Idreos, M. L. Kersten, and S. Manegold. Self-organizing Tuple Reconstruction in Column-stores. In *Proc. of the ACM Int'l Conf. on Management of Data (SIGMOD)*, pages 297–308, June 2009.

[10] 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 VLDB Endowment (PVLDB)*, 4(9):585–597, June 2011.

[11] M. Ivanova, M. L. Kersten, N. Nes, and R. Goncalves. An Architecture for Recycling Intermediates in a Column-store. In *Proc. of the ACM Int'l Conf. on Management of Data (SIGMOD)*, pages 309–320, June 2009.

[12] R. A. Kader, P. A. Boncz, S. Manegold, and M. van Keulen. ROX: Run-time Optimization of XQueries. In *Proc. of the ACM Int'l Conf. on Management of Data (SIGMOD)*, pages 615–626, June 2009.

[13] M. L. 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 VLDB Endowment (PVLDB)*, 4(12):1474–1477, Aug. 2011.

[14] M. L. Kersten, E. Liarou, and R. Goncalves. A Query Language for a Data Refinery Cell. In *Proc. of the Int'l Workshop on Event-driven Architecture, Processing and Systems (EDA-PS)*, Sept. 2007.

[15] E. Liarou, R. Goncalves, and S. Idreos. Exploiting the Power of Relational Databases for Efficient Stream Processing. In *Proc. of the Int'l Conf. on Extending Database Technology (EDBT)*, pages 323–334, Mar. 2009.

[16] S. Manegold, P. A. Boncz, and M. L. Kersten. Generic Database Cost Models for Hierarchical Memory Systems. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB)*, pages 191–202, Aug. 2002.

[17] S. Manegold, P. A. Boncz, and M. L. Kersten. Optimizing Main-Memory Join On Modern Hardware. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 14(4):709–730, July 2002.

[18] S. Manegold, P. A. Boncz, N. Nes, and M. L. Kersten. Cache-Conscious Radix-Decluster Projections. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB)*, pages 684–695, Aug. 2004.

[19] S. Manegold, M. L. Kersten, and P. A. Boncz. Database Architecture Evolution: Mammals Flourished long before Dinosaurs became Extinct. *Proceedings of the VLDB Endowment (PVLDB)*, 2(2):1648–1653, Aug. 2009.

[20] L. Sidirourgos, R. Goncalves, M. L. Kersten, N. Nes, and S. Manegold. Column-Store Support for RDF Data Management: not all swans are white. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB)*, pages 1553–1563, Sept. 2008.

[21] L. Sidirourgos, M. L. Kersten, and P. A. Boncz. SciBORQ: Scientific data management with Bounds On Runtime and Quality. In *Proc. of the Int'l Conf. on Innovative Data Systems Research (CIDR)*, pages 296–301, Jan. 2011.

[22] M. Zukowski, N. Nes, and P. A. Boncz. DSM vs. NSM: CPU performance tradeoffs in block-oriented query processing. In *Proc. of the Int'l Workshop on Data Management on New Hardware (DaMoN)*, pages 47–54, June 2008.

SAP HANA Database - Data Management for Modern Business Applications

Franz Färber [#1], Sang Kyun Cha [+2], Jürgen Primsch [?3],
Christof Bornhövd [*4], Stefan Sigg [#5], Wolfgang Lehner [#6]

[#]*SAP – Dietmar-Hopp-Allee 16 – 69190, Walldorf, Germany*
[+]*SAP – 63-7 Banpo 4-dong, Seochoku – 137-804, Seoul, Korea*
[?]*SAP – Rosenthaler Str. 30 – 10178, Berlin, Germany*
[*]*SAP – 3412 Hillview Ave – Palo Alto, CA 94304, USA*

[1]franz.faerber@sap.com [2]sang.k.cha@sap.com [3]j.primsch@sap.com
[4]christof.bornhoevd@sap.com [5]stefan.sigg@sap.com [6]wolfgang.lehner@sap.com

ABSTRACT

The SAP HANA database is positioned as the core of the SAP HANA Appliance to support complex business analytical processes in combination with transactionally consistent operational workloads. Within this paper, we outline the basic characteristics of the SAP HANA database, emphasizing the distinctive features that differentiate the SAP HANA database from other classical relational database management systems. On the technical side, the SAP HANA database consists of multiple data processing engines with a distributed query processing environment to provide the full spectrum of data processing – from classical relational data supporting both row- and column-oriented physical representations in a hybrid engine, to graph and text processing for semi- and unstructured data management within the same system.

From a more application-oriented perspective, we outline the specific support provided by the SAP HANA database of multiple domain-specific languages with a built-in set of natively implemented business functions. SQL – as the lingua franca for relational database systems – can no longer be considered to meet all requirements of modern applications, which demand the tight interaction with the data management layer. Therefore, the SAP HANA database permits the exchange of application semantics with the underlying data management platform that can be exploited to increase query expressiveness and to reduce the number of individual application-to-database round trips.

1. INTRODUCTION

Data management requirements for enterprise applications have changed significantly in the past few years. For example, it is no longer reasonable to continue the classical distinction between transactional and analytical access patterns. From a business perspective, queries in transactional environments on the one hand are building the sums of already-delivered orders, or calculating the overall liabilities per customer. On the other hand, analytical queries require the immediate availability of operational data to enable accurate insights and real-time decision making. Furthermore, applications demand a holistic, consistent, and detailed view of its underlying business processes, thus leading to huge data volumes that have to be kept online, ready for querying and analytics. Moreover, non-standard applications like planning or simulations require a flexible and graph-based data model, e.g., to compute the maximum throughput of typical business relationship patterns within a partner network. Finally, text retrieval technology is a must-have in state-of-the-art data management platforms to link unstructured or semi-structured data or results of information retrieval queries to structured business-related contents.

In a nutshell, the spectrum of required application support is tremendously heterogeneous and exhibits a huge variety of interaction patterns. Since classical SQL-based data management engines are too narrow for these application requirements, the SAP HANA database presents itself as a first step towards a holistic data management platform providing robust and efficient data management services for the specific needs of modern business applications [5].

The SAP HANA database is a component of the overall SAP HANA Appliance that provides the data management foundation for renovated and newly developed SAP applications (see Section 4). Figure 1 outlines the different components of the SAP HANA Appliance. The SAP HANA Appliance comprises replication and data transformation services to easily move SAP and non-SAP data into the HANA system, modeling services to create the business models that can be deployed and leveraged during runtime, and the SAP

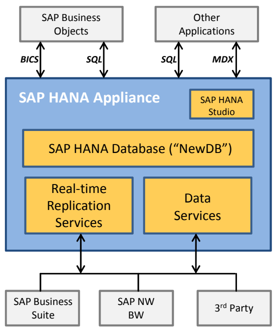


Figure 1: Components of the SAP HANA Appliance

HANA database as its core. For the rest of this paper, we specifically focus on the SAP HANA database.

Core Distinctive Features of the SAP HANA Database

Before diving into the details, we will outline some general distinctive features and design guidelines to show the key differentiators with respect to common relational, SQL-based database management systems. We believe that these features represent the cornerstones of the philosophy behind the SAP HANA database:

- *Multi-engine query processing environment*: In order to cope with the requirements of managing enterprise data with different characteristics in different ways, the SAP HANA database comprises a multi-engine query processing environment. In order to support the core features of enterprise applications, the SAP HANA database provides SQL-based access to relationally structured data with full transactional support. Since more and more applications require the enrichment of classically structured data with semi-structured, unstructured, or text data, the SAP HANA database provides a text search engine in addition to its classical relational query engine. The HANA database engine supports "joining" semi-structured data to relations in the classical model, in addition to supporting direct entity extraction procedures on semi-structured data. Finally, a graph engine natively provides the capability to run graph algorithms on networks of data entities to support business applications like production planning, supply chain optimization, or social network analyses. Section 2 will outline some of the details.

- *Representation of application-specific business objects*: In contrast to classical relational databases, the SAP HANA database is able to provide a deep understanding of the business objects used in the application layer. The SAP HANA database makes it possible to register "semantic models" inside the database engine to push down more application semantics into the data management layer. In addition to registering semantically richer data structures (e.g., OLAP cubes with measures and dimensions), SAP HANA also provides access to specific business logics implemented directly deep inside the database engine. The SAP HANA Business Function Library encapsulates those application procedures. Section 3 will explain this feature from different perspectives.

- *Exploitation of current hardware developments*: Modern data management systems must consider current developments with respect to large amounts of available main memory, the number of cores per node, cluster configurations, and SSD/flash storage characteristics in order to efficiently leverage modern hardware resources and to guarantee good query performance. The SAP HANA database is built from the ground up to execute in parallel and main-memory-centric environments. In particular, providing scalable parallelism is the overall design criteria for both system-level up to application-level algorithms [6, 7].

- *Efficient communication with the application layer*: In addition to running generic application modules inside the database, the system is required to communicate efficiently with the application layer. To meet this requirement, plans within SAP HANA development are, on the one hand, to provide shared-memory communication with SAP proprietary application servers and more closely align the data types used within each. On the other hand, we plan to integrate novel application server technology directly into the SAP HANA database cluster infrastructure to enable interweaved execution of application logic and database management functionality.

2. ARCHITECTURE OVERVIEW

The SAP HANA database is a memory-centric data management system that leverages the capabilities of modern hardware, especially very large amounts of main memory, multi-core CPUs, and SDD storage, in order to improve the performance of analytical and transactional applications. The HANA database provides the high-performance data storage and processing engine within the HANA Appliance.

Figure 2 shows the architecture of the HANA database system. The Connection and Session Management component creates and manages sessions and connections for the database clients. Once a session has been established, database clients can use SQL (via

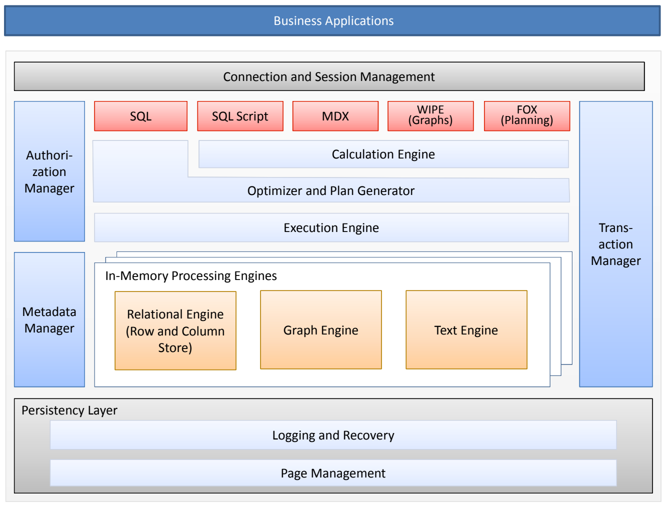


Figure 2: The SAP HANA database architecture

JDBC or ODBC), SQL Script, MDX or other domain-specific languages like SAP's proprietary language FOX for planning applications, or WIPE, which combines graph traversal and manipulation with BI-like data aggregation to communicate with the HANA database. SQL Script is a powerful scripting language to describe application-specific calculations inside the database.

SQL Script is based on side-effect-free functions that operate on database tables using SQL queries, and it has been designed to enable optimization and parallelization.

As outlined in our introduction, the SAP HANA database provides full ACID transactions. The Transaction Manager coordinates database transactions, controls transactional isolation, and keeps track of running and closed transactions. For concurrency control, the SAP HANA database implements the classical MVCC principle that allows long-running read transactions without blocking update transactions. MVCC, in combination with a time-travel mechanism, allows temporal queries inside the Relational Engine.

Client requests are parsed and optimized in the Optimizer and Plan Generator layer. Based on the optimized execution plan, the Execution Engine invokes the different In-Memory Processing Engines and routes intermediate results between consecutive execution steps.

SQL Script and supported domain-specific languages are translated by their specific compilers into an internal representation called the "Calculation Model". The execution of these calculation models is performed by the Calculation Engine. The use of calculation models facilitates the combination of data stored in different In-Memory Storage Engines as well as the easy implementation of application-specific operators in the database engine.

The Authorization Manager is invoked by other HANA database components to check whether a user has the required privileges to execute the requested operations. A privilege grants the right to perform a specified operation (such as create, update, select, or execute). The database also supports analytical privileges that represent filters or hierarchy drill-down limitations for analytical queries as well as control access to values with a certain combination of dimension attributes. Users are either authenticated by the database itself, or the authentication is delegated to an external authentication provider, such as an LDAP directory.

Metadata in the HANA database, such as table definitions, views, indexes, and the definition of SQL Script functions, are managed by the Metadata Manager. Such metadata of different types is stored in one common catalogue for all underlying storage engines.

The center of Figure 2 shows the three In-Memory Storage Engines of the HANA database, i.e., the Relational Engine, the Graph Engine, and the Text Engine. The Relational Engine supports both row- and column-oriented physical representations of relational tables. The Relational Engine combines SAP's P*Time database engine and SAP's TREX engine currently being marketed as SAP BWA to accelerate BI queries in the context of SAP BW. Column-oriented data is stored in a highly compressed format in order to improve the efficiency of memory resource usage and to speed up the data transfer from storage to memory or from memory to CPU. A system administrator specifies at definition time whether a new table is to be stored in a row- or in a column-oriented format. Row- and column-oriented database tables can be seamlessly combined into one SQL statement, and subsequently, tables can be moved from one representation form to the other [4]. As a rule of thumb, user and application data is stored in a column-oriented format to benefit from the high compression rate and from the highly optimized access for selection and aggregation queries. Metadata or data with very few accesses is stored in a row-oriented format.

The Graph Engine supports the efficient representation and processing of data graphs with a flexible typing system. A new dedicated storage structure and a set of optimized base operations are introduced to enable efficient graph operations via the domain-specific WIPE query and manipulation language. The Graph Engine is positioned to optimally support resource planning applications with huge numbers of individual resources and complex mash-up interdependencies. The flexible type system additionally supports the efficient execution of transformation processes, like data cleansing steps in data-warehouse scenarios, to adjust the types of the individual data entries, and it enables the ad-hoc integration of data from different sources.

The Text Engine provides text indexing and search capabilities, such as exact search for words and phrases, fuzzy search (which tolerates typing errors), and linguistic search (which finds variations of words based on linguistic rules). In addition, search results can be ranked and federated search capabilities support searching across multiple tables and views. This functionality is available to applications via specific SQL extensions. For text analyses, a separate Preprocessor Server is used that leverages SAP's Text Analysis library.

The Persistency Layer, illustrated at the bottom of Figure 2, is responsible for the durability and atomicity of transactions. It manages data and log volumes on disk and provides interfaces for writing and reading data that are leveraged by all storage engines. This layer is based on the proven persistency layer of MaxDB, SAP's commercialized disk-centric relational database. The persistency layer ensures that the database is restored to the most recent committed state after a restart and that transactions are either completely executed or completely undone. To achieve this efficiently, it uses a combination of write-ahead logs, shadow paging, and savepoints.

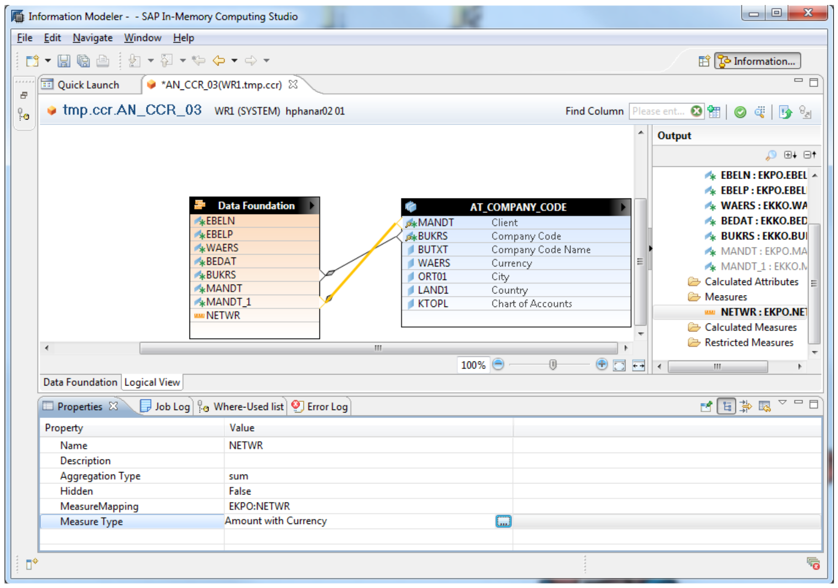
To enable scalability in terms of data volumes and the number of application requests, the SAP HANA database supports scale-up and scale-out. For scale-up scalability, all algorithms and data structures are designed to work on large multi-core architectures especially focusing on cache-aware data structures and code fragments. For scale-out scalability, the SAP HANA database is designed to run on a cluster of individual machines allowing the distribution of data and query processing across multiple nodes. The scalability features of the SAP HANA database are heavily based on the proven technology of the SAP BWA product.

3. SAP HANA DATABASE: BEYOND SQL

As outlined in our introduction, the SAP HANA database is positioned as a modern data management and processing layer to support complex enterprise-scale applications and data-intensive business processes. In addition to all optimizations and enhancements at the technical layer (modern hardware exploitation, columnar and row-oriented storage, support for text and irregularly structured data, etc.), the core benefit of the system is its ability to understand and directly work with business objects stored inside the database. Being able to exploit the knowledge of complex-structured business objects and to perform highly SAP application-specific business logic steps deep inside the engine is an important differentiator of the SAP HANA database with respect to classical relational stores.

More specifically, the "Beyond SQL" features of the SAP HANA database are revealed in multiple ways. On a smaller scale, specific SQL extensions enable the exposure of the capabilities of the specific query processing engines. For example, an extension in the WHERE clause allows the expression of fuzzy search queries against the text engine. An explicit "session" concept supports the planning of processes and *What if?* analyses. Furthermore, SQL Script provides a flexible programming language environment as a combination of imperative and functional expressions of SQL snippets. The imperative part allows one to easily express data and control flow logic by using DDL, DML, and SQL-Query statements as well as imperative language constructs like loops and conditionals. Functional expressions, on the other hand, are used to express declarative logics for the efficient execution of data-intensive computations. Such logic is internally represented as data flows that can be executed in parallel. As a consequence, operations in a data flow graph must be free of side effects and must not change any global states, neither in the database nor in the application. This condition is enforced by allowing only a limited subset of language features to express the logic of the procedure.

On a larger scale, domain-specific languages can be supported by specific compilers to the same logical construct of a "Calculation Model" [2]. For example, MDX will be natively translated into the internal query processing structures by resolving complex dimensional expressions during the compile step as much as possible by consulting the registered business object structures stored in the metadata catalog. In contrast to classical BI application stacks, there is no need for an extra OLAP server to generate complex SQL statements. In addition, the database optimizer is not required to "guess" the semantics of the SQL statements in order to generate the best plan – the SAP HANA database can directly ex-



tmp.ccr.AN_CCR_03 WR1 (SYSTEM) hphanar02 01

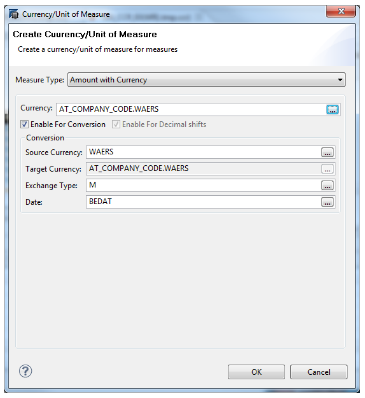
Data Foundation
EBELN
EBELP
WAERS
BEDAT
BUKRS
MANDT
MANDT_1
NETWR

AT_COMPANY_CODE	
MANDT	Client
BUKRS	Company Code
BUTXT	Company Code Name
WAERS	Currency
ORT01	City
LAND1	Country
KTOPL	Chart of Accounts

EBELN : EKPO.EBEL
EBELP : EKPO.EBEL
WAERS : EKKO.WA
BEDAT : EKKO.BED
BUKRS : EKKO.BU
MANDT : EKPO.MA
MANDT_1 : EKKO.M
Calculated Attributes
Measures
NETWR : EKPO.NE
Calculated Measures
Restricted Measures

Property	Value
Name	NETWR
Description	
Aggregation Type	sum
Hidden	False
MeasureMapping	EKPO:NETWR
Measure Type	Amount with Currency

(a) SAP Information Modeler



Create Cuurency/Unit of Measure

Create a currency/unit of measure for measures

Measure Type: Amount with Currency

Currency: AT_COMPANY_CODE.WAERS

[✓ Enable For Conversion] [✓ Enable For Decimal shifts]

Conversion

Source Currency: WAERS

Target Currency: AT_COMPANY_CODE.WAERS

Exchange Type: M

Date: BEDAT

(b) Concurrency conversion dialog

Figure 3: Modeling of currency conversion within SAP HANA

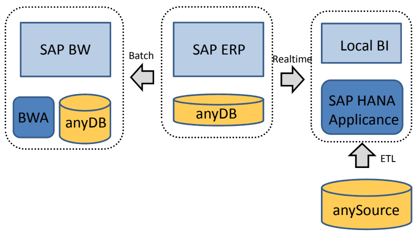
ploit the knowledge of the OLAP models carrying much more semantics compared to plain relational structures.

As an additional example of this "Beyond SQL" feature, consider the disaggregation step in financial planning processes [3]. In order to distribute coarse-grained planning figures to atomic entries—for example, from business unit level to department level—different distribution schemes have to be supported: relative to the actual values of the previous period, following constant distribution factors, etc. Since disaggregation is such a crucial operation in planning, the SAP HANA database provides a special operator, available within its domain-specific programming language, for planning scenarios. Obviously, such an operator is not directly accessible via SQL. Following this principle, the SAP HANA database also provides a connector framework to work with "external" language packages like the statistical programming environment R [1].

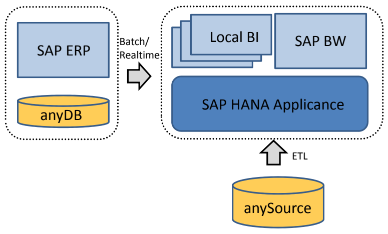
In addition to specifically tailored operators, the SAP HANA database also provides a built-in Business Function Library (BFL) that offers SAP-specific application code. All business logic modules are natively integrated into the database kernel with a maximum degree of parallelism. Compared to classical stored procedures or stored functions, the BFL is included in the database engine using all the technical advantages of deep integration. A prominent example of an application-specific algorithm is the procedure of currency conversion. Though supposedly simple in nature—a scalar multiplication of a monetary figure with the conversion rate—the actual implementation covering the complete application semantics of currency conversion comprises more than 1,000 lines of code. Figure 3(a) illustrates the graphical tool to create a "Calculation Model" in the SAP HANA database by applying a currency conversion function to an incoming data stream. As can be seen, the data source itself comprises not only simple columns but also comprehensive metadata such as type information with respect to plain, calculated, or derived measures.

The application designer creates a logical view using the Information Modeler and applies pre-defined application logics provided by the BFL. As shown in the modeling dialog of Figure 3(b), the parameters of the currency conversion function can be set in multiple ways to instrument the business logic. In the current example, the function performs a conversion to the currency with respect to the specific company code (given in column `AT_COMPANY_CODE.WAERS`).

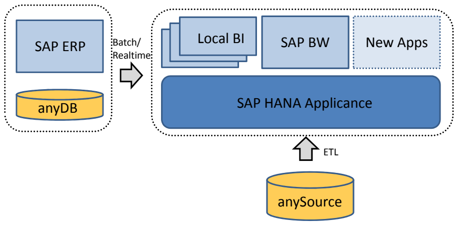
To summarize, the SAP HANA database provides a classical SQL interface including all transactional properties required from a classical database management system. In addition, the SAP HANA database positions itself as a system "Beyond SQL" by providing an ecosystem for domain-specific languages with particular internal support on the level of individual operators. Moreover, the concept of a BFL to provide a set of complex, performance-critical, and standardized application logic modules deep inside the database kernel creates clear benefits for SAP and customer-specific applications.



(a) Supporting local BI



(b) Running SAP BW



(c) Platform for new applications

Figure 4: Planned SAP HANA roadmap

4. THE HANA ROADMAP

Although, from a technology perspective, the SAP HANA database is based on the SAP BWA system with its outstanding record of successful installations, the generally novel approach of a highly distributed system with an understanding of semantic business models requires time for customers to fully leverage their data management infrastructure. SAP intends to pursue an evolutionary, step-wise approach to introduce the technology to the market.

In a first step, the SAP HANA Appliance is positioned to support local BI scenarios. During this step, customers can familiarize themselves with the technology exploiting the power of the new solution without taking any risk for existing mission-critical applications. SAP data of ERP systems will be replicated to the SAP HANA Appliance in real-time fashion. Data within the SAP HANA Appliance can be optionally enhanced by external non-SAP data sources and consumed using the SAP BOBJ analytical tools. Aside from new analytical applications on top of HANA, the primary use case here is the acceleration of operational reporting processes directly on top of ERP data.

The plan for the second phase of the roadmap, as shown in Figure 4(b), comprises supporting the full SAP BW application stack. Step by step, the customer is able to move more critical applications (like data warehousing) to the SAP HANA Appliance. This phase also positions the SAP HANA Appliance as the primary persistent storage layer for managed analytical data. Switching the data management platform will be a non-disruptive move from the application's point of view. In addition to providing the data management layer for a centralized data-warehouse infrastructure, the SAP HANA Appliance is also planned to be used to consolidate local BI data marts exploiting a built-in multi-tenancy feature.

The third step in the current roadmap – introducing the SAP HANA Appliance to the market in an evolutionary way – consists of extending the HANA ecosystem with new applications using the modeling and programming paradigm of the SAP HANA database in combination with application servers. Depending on the specific customer setup, long-term plans are to put HANA also under the classical SAP ERP software stack.

To summarize, the basic steps behind the HANA roadmap are designed to integrate with customers' SAP installations without disrupting existing software landscapes. Starting small with local BI installations, putting the complete BW stack on top of HANA in combination with a framework to consolidate local BI installations, is considered a cornerstone in the SAP HANA roadmap.

5. SUMMARY

Providing efficient solutions for enterprise-scale applications requires a robust and efficient data management and processing platform with specialized support for transaction, analytical, graph traversal, and text retrieval processing. Within the SAP HANA Appliance, the HANA database represents the first step towards a new generation of database systems designed specifically to provide answers to questions raised by demanding enterprise applications. The SAP HANA database, therefore, should not be compared to classical SQL or typical key-value, document-centric, or graph-based NoSQL databases. HANA is a flexible data storage, manipulation, and analysis platform, comprehensively exploiting current trends in hardware to achieve outstanding query performance and throughput at the same time. The different engines within the distributed data processing framework provide an adequate solution for different application requirements. In this paper, we outlined our overall idea of the SAP HANA database, sketched out its general architecture, and finally gave some examples to illustrate how an SAP HANA database positions itself "Beyond SQL" by natively supporting performance-critical application logics as an integral part of the database engine.

6. ACKNOWLEDGMENTS

We would like to express our sincere thanks to all of our NewDB colleagues for making the HANA story a reality. We also would like to thank Glenn Pauley, SIGMOD Record Editor for Industrial Perspectives, for his helpful comments.

7. REFERENCES

[1] P. Grosse, W. Lehner, T. Weichert, F. Färber, and W.-S. Li. Bridging two worlds with RICE. In *VLDB Conference*, 2011.

[2] B. Jaecksch, F. Färber, F. Rosenthal, and W. Lehner. Hybrid Data-Flow Graphs for Procedural Domain-Specific Query Languages. In *SSDBM Conference*, pages 577–578, 2011.

[3] B. Jaecksch, W. Lehner, and F. Färber. A plan for OLAP. In *EDBT conference*, pages 681–686, 2010.

[4] J. Krüger, M. Grund, C. Tinnefeld, H. Plattner, A. Zeier, and F. Faerber. Optimizing Write Performance for Read Optimized Databases. In *DASFAA Conference*, pages 291–305, 2010.

[5] H. Plattner and A. Zeier. *In-Memory Data Management: An Inflection Point for Enterprise Applications*. Springer, Berlin Heidelberg, 2011.

[6] J. Schaffner, B. Eckart, D. Jacobs, C. Schwarz, H. Plattner, and A. Zeier. Predicting In-Memory Database Performance for Automating Cluster Management Tasks. In *ICDE Conference*, pages 1264–1275, 2011.

[7] C. Weyerhäuser, T. Mindnich, F. Färber, and W. Lehner. Exploiting Graphic Card Processor Technology to Accelerate Data Mining Queries in SAP NetWeaver BIA. In *ICDM Workshops*, pages 506–515, 2008.

Report on the 8th International Workshop on Quality in Databases (QDB10)

Andrea Maurino
Department of Informatics Systems and Communication
University of Milano - Bicocca
Via Bicocca degli Arcimboldi 8
20136, Milano, Italy
maurino@disco.unimib.it

Cinzia Cappiello
Department of Electronics and Information
Politecnico di Milano
Via Ponzio 34/5
20122, Milano, Italy
cappiello@elet.polimi.it

Panos Vassiliadis
Dept. of Computer Science
University of Ioannina
Ioannina, 45110, Hellas
pvassil@cs.uoi.gr

Kai-Uwe Sattler
FG Datenbanken und Informationssysteme
Ilmenau University of Technology
Postfach 100 565
D-98684 Ilmenau
kus@tu-ilmenau.de

1. INTRODUCTION

The eighth international workshop on Quality in Database was held in Singapore, on September 13th, 2010 and co-located with the 36th Conference on Very Large DataBase (VLDB). The main objective of the workshop was to address the challenge to detect data anomalies and assess, monitor, improve, and maintain the quality of information.

The workshop attracted 12 submissions from Asia, Australia, Europe, and the United States, out of which the Program Committee finally accepted 9 full papers. The accepted papers focused on important issues especially related to Data Quality assessment, Entity Matching, and Information Overloading.

2. QUALITY IN DATABASES: OPEN ISSUES

QDB 2010 was the eighth workshop addressing the challenges of quality in databases. Significant research contributions were presented in this edition. Anyway, the existing research works in the area of data and information quality are still far from maturity and significant room for progress exists. The participants agreed that many open challenges still remain. It is possible to classify them into three general areas:

- Improvement of the comparison of algorithms and evaluation through DQ standards
- Further investigation of well known DQ issues (privacy and visualization)
- Application of DQ in new domains (such as linked open data, and Data as a Service)

Concerning the first point, discussions with participants highlighted the lack of standards for comparing different solutions, algorithms and tools. In fact, a lot of papers create their own homemade benchmarks or golden rules to demonstrate the efficiency and the effectiveness of algorithms, but it is very rare that such data are shared to support cross comparison analysis. Participants suggested to organize specific data quality events to create occasions in which data quality researchers can compare and discuss novel ideas such as an international contest about data quality evaluation similar to the KDD cup `http://www.sigkdd.org/kddcup/` or the semantic Web service challenge `http://sws-challenge.org/wiki/index.php/Main_Page`. One concrete example, related to personal data only, is the "name game" workshop series organized within the APE-INV project `http://www.academicpatenting.eu`.

As regards the second open problem, participants agreed that DQ research should focus more on visualization and privacy issues. In fact, the visualization of the results of DQ activities (ranging from assessment to record linkage and data improvement) is a crucial point for a larger dissemination of DQ researches. It includes typical problems related to the visualization of large data sets (as in data mining field), but also it needs more tailored solutions to underline errors or to interprete the results of DQ activities. In this field, mashups seem to be a promising technology for easily integrating DQ results and supporting improvement decisions. Moreover, an important aspect when integrating data from a large number of heterogeneous sources under diverse ownerships is the provenance of data or parts thereof[2]; provenance denotes the origin of data and can also include information on processing or reasoning operations carried out on the data. In addition, provenance enables effective support of trust mechanisms and policies for privacy and rights management. In the last years many solutions for provenance models and management mechanisms have been published[4]. However, according to participants a lot of problems are still opens. For example, there is the need for a trust management system for improving accountability, building on provenance, especially in the context of data aggregation.

Finally, workshop paticipants also agreed that data quality research should also focus on the definition of methods for the quality assessment and improvement of data modelled on the basis of new paradigms for information management. Linked open data, for example, is a set of principles to share in the Web environment open data [1]. This can bring a paradigmatic shift from the classical relational data integration architecture towards new Web based solutions. In this specific scenario data are retrieved from heterogeneous data sources (e.g., relational, graph and stream-like data) and there are not appropriate methods to assess, preserve and improve the quality of such data sets. In fact, while the requirements for the quality assessment of closed (mostly corporate) data sources are well understood, several open issues raise when quality assessment has to be performed in autonomous and distributed data sources where quality-related meta-information is typically sparse and the quality of the meta-information is uncertain.

Another new paradigm for data management that it is worth to consider is the Data as a Service. Until now, "data" and "services" have been always considered two different concepts characterized by different problems, approaches, models and tools. Nevertheless in the last years there is a growing interest to the Xaas approach. X as a services, where X could be software [6] or a platform [3], introduces the possibility to consider Data as a Service and consequently data can be managed by means of typical service oriented solutions. In this context, data quality could play, for example, a fundamental role in the selection of services that is commonly based on other functional and non-functional properties (e..g., response time, availability). The convergence of data and service approaches could be the first steps toward the definition of a new and holistic theory where data and service are considered as two faces of the same problem [5].

3. KEYNOTE PRESENTATION

The keynote speech, titled "SOLOMON: Seeking the Truth Via Copying Detection" was delivered by Xin Luna Dong, AT&T. In the information era, a large amount of information sources are available and easily accessible. However, freely accessible information is often unreliable: it is often accessed by data quality problems in terms of relevance, accuracy, or authority. Moreover, the information diffusion enabled by Web technologies negatively impacts on data quality issues since errors can be easily propagated. The identification of copying content between information sources could be a valuable help for the users to filter relevant data. In this keynote Xin Luna Dong presented the SOLOMON tool that supports the discovery of copying relationships between structured data source to improve data integration features. She also explained which are the research open issues for leveraging redundancy and obtain quality from Web sources. Open issues in this field are mainly related to source selection (e.g., how many sources are sufficient for aggregation?), source integration (e.g., source ordering in online query answering) and data visualization (e.g., task-driven source exploration).

4. RESEARCH PAPERS

The technical paper session consisted of nine presentations, whose main points are summarized next. Together, they give a glimpse to the exciting new developments on data and information quality.

The paper titled "Quality Assessment Social Networks: A Novel Approach for Assessing the Quality of Information on the Web" by Tomas Knap, Irena Mlynkova introduces a Web Quality Assessment model, which is a model for the ranking of Web resources on the grounds of a quality assessment (QA) score, involving profiles and policies for the management of the resources. Since people in social networks might benefit from adopting profile properties from other people with which they are linked, the paper introduces the concept of QA social networks and algorithms for the successive application of relevant QA policies (i.e., policies of trusted users) to a person's retrieved resources.

The goal of "Deriving Effectiveness Measures for Data Quality Rules" by Lei Jiang, Alex Borgida, Daniele Barone, John Mylopoulos is the evaluation of data quality rules. Broadly speaking, data quality rules detect errors and inconsistencies and they play an important role in data quality assessment. Starting from the results of previous research, the authors propose a quantitative framework for measuring and comparing data quality rules in terms of their effectiveness. Effectiveness formulas are built from variables that represent probabilistic assumptions about the occurrence of errors in data values, and earlier work gave examples of how to derive these formulas in an ad-hoc fashion. The presented approach involves several steps, including building Bayesian network graphs, adding (symbolic) probabilities to the nodes in the graph, and deriving effectiveness formulas. The approach is implemented in Python, and the paper reports its evaluation results.

Soumaya Ben Hassine-Guetari, Jérôme Darmont, Jean-Hugues Chauchat with the paper "Aggregation of data quality metrics using the Choquet integral" present a solution that uses the Choquet integral to aggregate different data quality metrics into a single score. When comparing different (data) items, many quality dimensions might be used along with their respective metrics. When two items A and B have different scores over different dimensions, it is not straightforward to compute a global qualifying score to facilitate their comparison. In this perspective, the aggregation of data quality metrics can be the solution for computing a global and objective data quality score. The authors contribute to the solution of the problem by suggesting how the Choquet integral might provide an answer.

The paper "Data Partitioning for Parallel Entity Matching" written by Toralf Kirsten, Lars Kolb, Michael Hartung, Anika Groß, Hanna Köpcke, and Erhard Rahm deals with an important problem of entity matching, which is an important and difficult step for integrating Web data. In order to reduce the execution time for matching algorithms, the authors investigate how entity matching can be performed in parallel on a distributed infrastructure. The paper proposes different strategies to partition the input data and generate multiple match tasks that can be independently executed. One of the suggested strategies supports both blocking to reduce the search space for matching and parallel matching to improve efficiency. Special attention is given to the number and size of data partitions as they impact the overall communication overhead and memory requirements of individual match tasks. The caching of input entities and affinity-based scheduling of matching tasks is also considered. The authors also discuss the tool that they have developed in a service-based, distributed infrastructure as well as the detailed evaluation of their method.

An interesting tool for duplicate detection is proposed in "DuDe: The Duplicate Detection Toolkit" by Uwe Draisbach and Felix Naumann. Duplicate detection, also known as entity matching or record linkage, has been a research topic for several decades. The challenge is to effectively and efficiently identify pairs of records that represent the same real world entity. Researchers have developed and described a variety of methods to measure the similarity of records and/or to reduce the number of required comparisons. Comparing these methods to each other is essential to assess their quality and efficiency. However, it is still difficult to compare results, as differences can always be found in the evaluated data sets, the similarity measures, the implementation of the algorithms, or simply the hardware on which the code is executed. To face this challenge, the paper discusses the development of a comprehensive duplicate detection toolkit named DuDe. DuDe provides multiple methods and data sets for duplicate detection and consists of several components with clear interfaces that can be easily served with individual code.

An original problem is raised by Wolfgang Gottesheim, Norbert Baumgartner, Stefan Mitsch, Werner Retschitzegger, and Wieland Schwinger with the paper "Improving Situation Awareness" related to information overloading. Information overload is a severe problem for operators of large-scale control systems, as such systems typically provide a vast amount of information about a large number of real-world objects. Systems supporting situation awareness have recently gained attention as way to help operators to grasp the overall meaning of available information. To fulfill this task, data quality has to be ensured by assessment and improvement strategies. In this paper, a vision towards a methodology for data quality assessment and improvement for situation awareness systems is presented.

The management of conditional functional dependencies is an important topic described in "Extending Matching Rules with Conditions" by Shaoxu Song, Lei Chen, and Jeffrey Yu. Matching dependencies (mds) have recently been proposed in order to make dependencies tolerant to various information representations, and proved useful in data quality applications such as record matching. Instead of a strict identification function in traditional dependency syntax (e.g., functional dependencies), mds specify dependencies based on similarity matching quality. However, in practice, mds may still be too strict and only hold in a subset of tuples in a relation. Thereby, the paper proposes conditional matching dependencies (cmds), which bind matching dependencies only in a certain part of a table. Compared to mds, cmds have more expressive power that enables them to satisfy wider application needs. The paper includes a discussion of both theoretical and practical issues of cmds, including inferring cmds, irreducible cmds with less redundancy and, the discovery of cmds from data as well as the experimental evaluation of cmd discovery algorithms.

Finally, Peter Yeh, and Colin Puri show in "Discovering Conditional Functional Dependencies to Detect Data Inconsistencies" an approach that exploits conditional functional dependencies for detecting inconsistencies in data and hence improves data quality. The approach has been empirically evaluated on three real-world data sets, and the paper discusses the performance of the proposed approach in terms of precision, recall, and runtime. Moreover, a comparison between the presented approach and an established, state-of-the-art solution shows that the presented approach outperforms this solution across the previously mentioned dimensions. Finally, the paper describes efforts to deploy the approach as part of an enterprise tool to accelerate data quality efforts such as data profiling and cleansing.

5. ACKNOWLEDGMENTS

We would like to thank the program committee members, keynote speakers, authors and attendees, for making QDB 2010 a successful workshop. Finally, we also express our great appreciation for the support from the VLDB 2010 organization.

6. REFERENCES

[1] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.

[2] P. Buneman and W. C. Tan. Provenance in databases. In C. Y. Chan, B. C. Ooi, and A. Zhou, editors, *SIGMOD Conference*, pages 1171–1173. ACM, 2007.

[3] E. Keller and J. Rexford. The "platform as a service" model for networking. In *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, INM/WREN'10, pages 4–4, Berkeley, CA, USA, 2010. USENIX Association.

[4] D. L. McGuinness, J. Michaelis, and L. Moreau, editors. *Provenance and Annotation of Data and Processes - Third International Provenance and Annotation Workshop, IPAW 2010, Troy, NY, USA, June 15-16, 2010. Revised Selected Papers*, volume 6378 of *Lecture Notes in Computer Science*. Springer, 2010.

[5] M. Palmonari, A. Sala, A. Maurino, F. Guerra, G. Pasi, and G. Frisoni. Aggregated search of data and services. *Inf. Syst.*, 36(2):134–150, 2011.

[6] W. Sun, K. Zhang, S.-K. Chen, X. Zhang, and H. Liang. Software as a service: An integration perspective. In B. Kramer, K.-J. Lin, and P. Narasimhan, editors, *Service-Oriented Computing ICSOC 2007*, volume 4749 of *Lecture Notes in Computer Science*, pages 558–569. Springer Berlin / Heidelberg, 2007.

The Meaningful Use of Big Data: Four Perspectives – Four Challenges

Christian Bizer[1], Peter Boncz[2], Michael L. Brodie[3], Orri Erling[4]
[1]Web-based Systems Group, Freie Universität Berlin; [2]Centrum Wiskunde & Informatica, Amsterdam; [3]Verizon Communications, USA; [4]OpenLink Software, Utrecht
christian.bizer@fu-berlin.de, P.Boncz@cwi.nl, michael.brodie@verizon.com, oerling@openlinksw.com

Abstract

Twenty-five Semantic Web and Database researchers met at the 2011 STI Semantic Summit in Riga, Latvia July 6-8, 2011[1] to discuss the opportunities and challenges posed by Big Data for the Semantic Web, Semantic Technologies, and Database communities. The unanimous conclusion was that the greatest shared challenge was not only engineering Big Data, but also doing so meaningfully. The following are four expressions of that challenge from different perspectives.

Michael's Challenge: Big Data Integration is Multi-disciplinary

The exploding world of Big Data poses, more than ever, two challenge classes: engineering - efficiently managing data at unimaginable scale; and semantics – finding and meaningfully combining information that is relevant to your concern. Without the meaningful use of data, data engineering is just a bunch of cool tricks. Since every computer science discipline and every application domain has a vested interest, Big Data becomes a use case for multi-disciplinary problem solving[2]. The challenge posed here is of the meaningful use of Big Data regardless of the implementation technology or the application domain.

Emerging data-driven approaches in the US Healthcare Big Data World[3] involves over *50 million* patient databases distributed US-wide for which the US Government defines *Meaningful Use* and the medical community has identified challenges [4] across queries such as: "For every 54-year-old white, female high school dropout with a baseline blood pressure of 150 over 80 in the beta blocker group who had these *two* concurrent conditions and took these *three* mediations. Magid matched her to another 54-year-old female high school dropout with a baseline blood pressure of 150 over 80 in the ACE inhibitor group, who had the same drugs."[5]

In this Big Data World information is unbelievably large in scale, scope, distribution, heterogeneity, and supporting technologies. Regardless of the daunting engineering challenges, meaningful data integration takes the following form (step order can vary):

- **Define** the concern – the problem to be solved - the query to be answered, e.g., efficacy of a drug for 54-year-old hypertensive women.
- **Search** the Big Data space for candidate data elements that map to the concern; e.g., all hypertensive 54-year-old women.
- **Transform** Extract, Transform, and Load (ETL) the relevant parts of the candidate data elements into appropriate formats and stores for processing for processing.
- **Entity Resolution**: Verify that data elements are unique, relevant, and comprehensive, e.g., all hypertensive 54-year-old women. Since unique identification is practically and technically infeasible, not all candidate data elements will refer to the entity of concern. More challenging are data elements that describe aspects of the entity of concern at different level of abstraction and from different perspectives, e.g., data elements on myriad details of hypertensive 54-year-old women, e.g., physiology, social network membership, salary, education.
- **Answer the query/solve the problem**: Having selected the data elements relevant to the entity of concern, compute the answer using domain-specific computations, e.g., efficacy of the drug.

It is hard to conceive of the scope and scale of data elements in the Big Data World. The above method has worked amazing well for more than 30 years in the $27 billion per year relational database world with blinding efficiency over ever expanding database sizes from gigabytes, to terabytes, to petabytes, and now exabytes. Data elements that are genuinely relational constitute less than 10% of the Big Data World and that share is falling rapidly.

The rare properties of single value of truth, global schema, and view update of semantically homogeneous relational databases are often underlying assumptions of relational database integration. However, few relational databases are semantically homogeneous and like most data stores, they lack these properties. Hence, meaningful data integration solutions cannot be based on these properties without supporting evidence that must be derived manually. Since the real world involves multiple truths over every concern, relational data integration has semantic (correctness) and engineering (efficiency) limits.

My challenge is meaningful data integration in the real, messy, often schema-less, and complex Big Data World of databases and the (Semantic) Web using multi-disciplinary, multi-technology methods.

Chris' Challenge: The Billion Triple Challenge

Over the past few years, an increasing number of web sites have started to publish structured data on the Web according to the Linked Data principles. This trend has led to the extension of the Web with a global data space – the Web of Data [6].

Topology of the Web of Data Like the classic document Web, the Web of Data covers a wide variety of topics ranging from data describing people, organizations and events, over products and reviews to statistical data provided by governments as well as research data from various scientific disciplines.
W3C Linking Open Data (LOD) community effort has started to catalog known Linked Data sources in the CKAN data catalog and regularly generates statistics about the content of the data space [7]. According to these statistics, the Web of Data currently contains around 31 billion RDF triples. A total of 466 million of these triples are RDF links which connect data between different data sources. Major topical areas are government data (13 billion triples), geographic data (6 billion triples), publication and media (4.6 billion triples), life science (3 billion triples).

Characteristics of the Web of Data The Web of Data has several unique characteristics which make it an interesting use case for research on data integration as well as the Big Data processing:

- **Widely-used vs. proprietary vocabularies.** Many Linked Data sources reuse terms from widely-used vocabularies to represent data about common types of entities such as people, products, reviews, publications, and other creative works. In addition, they use their own, proprietary terms for representing aspects that are not covered by the widely used vocabularies. This partial agreement on terms makes it easier for applications to understand data from different data sources and is a valuable starting point for mining additional correspondences.
- **Identity and vocabulary links.** Many Linked Data sources set identity links (owl:sameAs) pointing at data about the same entity within other data sources. In addition data sources as well as vocabulary maintainers publish vocabulary links that represent correspondences between terms from different vocabularies (owl:equivalentClass, owl:equivalentProp-erty, rdfs:subClassOf, rdfs:subPropertyOf). Applications can treat these links as integration hints which help them to translate data into their target schema as well as to fuse data from different sources describing the same entity.
- **Data Quality:** The Web is an open medium in which everybody can publish data on the Web. As the classic document Web, the Web of Data contains data that is outdated, conflicting, or intentionally wrong (SPAM). Thus, one of the main challenges that Linked Data applications need to handle is to assess the quality of Web data and determine the subset of the available data that should be treated as trustworthy.

Pre-Crawled Data Sets One approach to obtain a corpus of Linked Data is to use publicly available software, such as LDSpider, to crawl the Web of Data. However, there exist already a number of publicly available data sets that have been crawled from the Web of Data and can be promptly used for evaluation and experimentation.

- **BTC 2011.** The Billion Triple Challenge 2011 data set (BTC 2011) has been crawled in May/June 2011 and consists of 2 billion RDF triples from Linked Data sources. There are also two older versions of the data set available which have been crawled in 2009 and 2010. The BTC data sets are employed in the Semantic Web Challenge, an academic competition that is part of the International Semantic Web Conference. The BTC data sets can be downloaded from the Semantic Web Challenge website [8].
- **Sindice 2011.** The Sindice 2011 data set has been crawled from the Web by the Sindice search engine. The data set consists of 11 billion RDF triples which (1) originate from Linked Data sources and (2) have been extracted from 230 million Web documents containing RDFa and Microformats markup. The data set contains descriptions of about 1.7 billion entities and can be downloaded from [9].

Now then, the task A concrete task, which touches all challenges around data integration, large-scale RDF processing, and data quality assessment that arise in the context of the Web of Data, is to (1) find all data that describes people (in whatever role) as well as creative works produced by these people (ranging from books, films, musical works to scientific publications) in the BTC 2011 or the Sindice 2011 data set; (2) translate this data from the different vocabularies that are used on the Web into a single target vocabulary, (3) discover all resources that describe the same real-world entity (identity resolution), and (4) fuse these descriptions into an integrated representation of all data that is available about the entity using a general or several domain-specific trust heuristics.

Success metrics: Success metrics for this task are the number of people and creative works discovered in the data set and on the other hand the completeness and consistency of the integrated data.

Peter's Challenge: The LOD Ripper

Motivation. For broader adoption of semantic web techniques, two main challenges arguably exist: (I) lack of good use cases (ii) ever existing data integration troubles that makes creating links so hard. The LOD Ripper idea originates from the thought that the best window of opportunity is linked open government data. If it became easy for people and companies to earn money and reap value from this high-quality & free information out there, linked open data might break through in this domain. If this fails to catch on soon, linked open government data investment in the early adaptor countries might drop, and might altogether fail to take off in the rest. Use cases outside government or academic data are much harder to find as one then faces the issue of an economic model for LOD production. So, better to succeed here.

Success Metric. A side note on what success could be. Success is not only achieved when the IT world switches to semantic-everything technology. Given the value of installed base, this is unrealistic. Success is already achieved when people combine multiple LOD datasets, and link them to their own data, but then import the result e.g. as a flat relational table (via CSV, XML, etc.) for use in existing infrastructure and tools. Think of existing enterprise middleware, business logic, data warehouses, and OLAP and data mining tools: technology that has been invested heavily in, and which would profit from enrichment by linked open government data. The semantic success will be in the fact that semantic technology has made data integration easier and partially automatic. Data integration is one of the highest cost issues in IT, worth tens of billions of dollars per year. Therefore, my name for the project, the "**LOD Ripper**": a technology to rip valuable data out of LOD sources. Admittedly, this is intended to be provocative to the Semantic Web community and to emphasize practicality. But, you could also use the LOD ripper to extract data in triple form, of course. The LOD ripper could also search non-LOD open government datasets, just like CKAN. Mapping these together may trigger the incremental LOD-ification of such datasets.

Now then, the proposal: the LOD Ripper is a vision of a web portal, driven by goals similar to CKAN, however going way beyond CKAN in its practical support for an information engineer in finding and combining useful open government data, and integrating it with his own. The portal would do the maximum possible, given a vague information need on the part of the information engineer, to put him as quickly as possible into hands-on mode with real data (snippets) from the entire data collection. This means among other things that one of the main ways to interact with the system is keyword search, which would search in (1) ontologies/schemas (2) the data itself and (3) mappings/views provided by earlier users of the portal. The goal of the portal is to assist the information engineer in obtaining a useful mapping that allows him to retrieve ("rip") a derived dataset that is valuable for his problem space. Point (3) stresses that this portal should facilitate a pay-as-you-go process.

Mappings. Obtaining a mapping may happen by finding an existing mapping, by combining multiple existing ones into a new one, or by fresh composition. The resulting mapping should be made available again for future users. Mapping languages are hence an important aspect, and user interfaces to compose mappings and mapping systems, as well as entity resolution algorithms are part of such systems. Mappings are not only specifications, but in the end will also take the form of new data, new or better triples, that add meaning in and between existing dataset(s). Such new triples may be generated by a mapping system following a mapping specification automatically, but should be materializable as triple sets, because often these need to be manually curated as well. Note that mappings need provenance tracking, at least in the form of a simple version tracking system.

Ranking. As we search schema, mapping and data, we need also ways to usefully rank these. On the one hand, ranking could be based on precision of match with keywords, but on the other, should be based on usefulness/quality assessment by previous users of the datasets and dataset elements.

Visualization. To show results of a search, we need good snippets or summaries of what we find. In the case of ontologies, one would use dataset summarization techniques, to visualize the most common structures in a dataset and where the keyword search matched in that. If we look for multiple types of data, one would also visualize the structure of any existing mappings between the hits, leaving out irrelevant details as much as possible. When searching for views/mappings, these should similarly be visually summarized. It should be one click to switch from looking at schema visualizations to see representative samples of underlying data occurring in the wild. There should be strong support for generating tabular data views out of the LOD sources. The ability to extract tables, using all the mapping machinery, is the prime output of the LOD Ripper portal.

Key Matching. The system should allow users to define keys, and upload possibly a large number of key values, which typically come from the users' own environment. Think for instance of a column containing city names as a potential key column. One purpose of such a key column in the LOD Ripper is to measure the overall effectiveness of finding useful data ("how many of my cities did I find info for?"). It also provides a concrete starting point for instance-driven data integration ("find me matching city properties anywhere!"). Note that this works on the instance level, and one needs algorithms to quickly search for similar and overlapping data distributions.

Snappiness. Visualizing results and creating mappings *interactively* is going to be very important. This means emphasis on cool GUI design as well as low-latency performance. This requires a solid LOD warehouse with advanced indexing performed in the background. A technique probably useful for instance-level data matching would be NGRAM indexing (to speed up partial string and distribution matching) as well as massive pre-computation of entity resolution methods.

Call to action: Can we organize such a portal? Do you have ideas and time, or even components available?

Orri's Challenge: Demonstrate the Value of Semantics: Let Data Integration Drive DBMS Technology

Advances in database technology will continue to facilitate dealing with large volumes of heterogeneous data. Linked data and RDF have a place in this, as they are a schema-less model with global identifiers and a certain culture of, or at least wish for, reuse of modeling.

Systematic adoption of DBMS innovation into the semantic data field, backed by systematic benchmarking, will make the schema-less flexibility of these technologies increasingly affordable.

These developments set the stage for the real challenge:

- **Demonstrate the benefit of semantics for data integration**. The RDF/data world does not exist in a bubble, in any real life situation it will be compared to alternatives.
- **Meaningfully combine DBMS and reasoning functions**. Identify real-world problems where there is real benefit in having logics more expressive than SQL or SPARQL close to the data. We have talked extensively about smarter databases but the actual requirement remains vague. We do not think of OWL or RIF as such answers for data integration even if they may be a part of it.
- **Bring Linked Data and RDF into the regular data-engineering stack:** Use existing query and visualization tools against heterogeneous data. There are many interactive SPARQL builders but are these performs comparable to MS Query for SQL? Since data here is schema-less, data set summarization will have to play the role that the schema plays with relational tools. There are many RDF bound UI widgets but few bind to Excel for business graphics?

We know how to make DBMS's. To get to the next level we need use cases that represent real needs, e.g. data integration. This information is required to determine what ought to be optimized or in what way the existing query languages / logics / processing models fail to measure up to the challenge.

So, users / practitioners, does there exist functionality that belongs with the data but cannot be expressed in queries? What about entity resolution frameworks? What about inference? What kind of inference? What of the many things people do in map/reduce, is there a better way? How about Berkeley Orders of Magnitude (BOOM) work for declarative data centric engineering for big data? I envision expanding the Semdata benchmarks activity to include specific use cases that come from you. What did you always want to do with a DBMS but never dared ask?

This could result in a set of use cases with model solutions with different tools and techniques. We are not talking about fully formalized benchmarks but about samples of problems motivating DBMS advances beyond standard query languages.

This in turn would bring us closer to quantifying the benefits of semantic technology for real world problems, which is after all our value proposition. Of course, this involves also non-RDF approaches, as we do not believe that there ought to be a separate RDF enclave but that technologies should be appreciated according to their merits. It is no wonder the bulk of database research has been drawn to the performance aspect, as success in this is fairly unambiguous to define and the rationale needs no explaining. But when we move to a more diverse field like data integration, which indubitably is the core question of big data, we need more stakeholder involvement.

Tell us what you need and we'll see how this shapes the future of DBMS.

If you are struggling with doing things that DBMS' s ought to do but do not support, let us know. Chances are that these problems could be couched in terms of open government data even if your application domain is entirely different, thus alleviating processes confidentiality problems.

References

[1] 2011 STI Semantic Summit, Riga, Latvia, http://www.sti2.org/events/2011-sti-semantic-summit

[2] M.L. Brodie, M. Greaves and J.A. Hendler, Databases and AI: The Twain Just Met, 2011 STI Semantic Summit, Riga, Latvia, July 6-8, 2011

[3] Preliminary Observations on Information Technology Needs and Priorities at the Centers for Medicare and Medicaid Services: An Interim Report, Committee on Future Information Architectures, Processes, and Strategies for the Centers for Medicare and Medicaid Services; CSTB; National Research Council of the Academies of Science, 2010

[4] E.M. Borycki, A.W. Kushniruk, S. Kuwata, J. Kannry, Engineering the electronic health record for safety: a multi-level video-based approach to diagnosing and preventing technology-induced error arising from usability problems, Stud Health Technol Inform. 2011;166:197-205.

[5] S. Begley, The Best Medicine: The Quiet revolution in comparative effectiveness research just might save us from soaring medical costs, Scientific American **305**, 50 - 55 (2011)

[6] C. Bizer, T. Heath, and T. Berners-Lee: Linked Data - The Story So Far. International Journal on Semantic Web & Information Systems, Vol. 5, Issue 3, Pages 1-22, 2009.

[7] C. Bizer, A. Jentzsch, and R. Cyganiak: State of the LOD Cloud. http://www4.wiwiss.fu-berlin.de/lodcloud/state/

[8] Semantic Web Challenge website. http://challenge.semanticweb.org/

[9] Sindice-2011 Dataset for TREC Entity Track. http://data.sindice.com/trec2011/

Fourth Workshop on Very Large Digital Libraries

On the marriage between Very Large Digital Libraries and Very Large Data Archives

Leonardo Candela
ISTI - Consiglio Nazionale delle Ricerche
Pisa, Italy
candela@isti.cnr.it

Paolo Manghi
ISTI - Consiglio Nazionale delle Ricerche
Pisa, Italy
manghi@isti.cnr.it

Yannis Ioannidis
University of Athens
Athens, Greece
yannis@di.uoa.gr

1. INTRODUCTION

The workshop series on Very Large Digital Libraries (VLDLs) started in 2008 [12] with the aim of fostering and initiating systematic and constructive discussions on the specific and rather novel research area of "very large digital libraries". Before this, building on long experience in the field of digital libraries and data infrastructures, the authors have spent efforts in the definition of the Digital Library Reference Model [4, 3] and in the definition of the Digital Library Technology and Methodology Cookbook [1]. Both initiatives had the common goal of consolidating digital library research as an independent and well established research field with peculiarities characterized by shared foundations. In line with this path, the VLDL workshop series has a twofold target. On the one hand delineating the boundaries of this research area, in an attempt to motivate its existence as an independent avenue of investigation. On the other hand identifying its foundations and grand challenges, so that research results could be classified and compared in a constructive confrontation. The long-term and ambitious goal is to discuss the foundations of VLDLs and establish it as a research field on its own, with well-defined areas, models, trends, open problems, and technology.

The main outcome of the workshop series [12, 8, 9, 5], also published as SIGMOD Record reports [11, 10], was consolidation of its mission. The presentations and following discussions collected in the years clearly promoted VLDLs as a chapter of their own in computer science research. VLDLs cannot be simply regarded as very large databases storing Digital Library (DL) content, as one may be tempted to assess. In fact, as the Reference Model for Digital Libraries well justifies, DL systems cannot be approached from the perspective of content management only; the dimensions of user, functionality, policy, quality, and architecture management are equally important. Accordingly, DLs become Very Large DLs (VLDLs) when any one of these aspects reaches a magnitude that requires specialized technologies or approaches. Actually, the appellation of "very large" is acquired whenever one of the following features apply to one of the dimensions above:

- *volume*, i.e. the dimension in terms of number of entities to be managed or size is huge;
- *velocity*, i.e. the speed requirements for collecting, processing and using entities is demanding; and
- *variety*, i.e. the heterogeneity in terms of entity types to be managed and sources to be merged is high.

However, there is not yet any threshold or indicator with respect to these features agreed by the community in the large that might be used to clearly discriminate very large digital libraries from digital libraries. Regardless of this, very large digital libraries have been developed – e.g. the Library of Congress[1], the National Science Digital Library[2], Europeana[3], DRIVER[4] – and the demand for infrastructures and services promoting collaboration and knowledge sharing on large scale is growing [6, 13].

The fourth VLDL workshop has been organized in conjunction with the 15th edition of the TPDL 2011 conference [7], which is part of the series of *European Conference on Research and Advanced Technology for Digital Libraries (ECDL)* started in 1997. This year workshop called for topics on theory and practice of VLDLs. More specifically, theoretical or foundational topics covered definitional models and measures (content, functionality, users, and policies), architectural models, and design methodologies for VLDLs. Practical or systemic topics covered ideas, experiments, and practical experiences in system design and implementation. Of particular interests were: integration and federation of DLs, user management, security, sustainability, scalability, distribution, interoperability for content, functionality, quality of service, storage, indexes, and preservation.

Moreover, unlike previous editions, this year the workshop proposed the traversal topic "...on the marriage between Very Large Digital Libraries and Very Large Data Archives ...". The idea was to call for contributions proposing research issues and solutions regarding VLDLs in relationship with research data. Research data is today an "hot" area in the field of DLs. Scientists are more and more realizing the need of tools capable of dealing with the so-called "tsunami"

[1]http://www.loc.gov
[2]http://nsdl.org
[3]http://www.europeana.eu
[4]http://search.driver.research-infrastructures.eu/

of data in order to make it accessible, searchable, reusable, or linked with the research publications it is related with [2]. The digital nature of research data, its requirements for several metadata descriptions (e.g. geo-reference, provenance), efficient-scalable-secure storage/access, advanced visualization and management tools, interoperability solutions, show many overlaps with the main DLs issues and, due to the multidisciplinary nature and cross-organizational character of data archives, with the very large nature of DLs.

2. WORKSHOP PRESENTATIONS

All submitted contributions were peer reviewed by two of the six members of the Program Committee and six were accepted. The workshop structure comprised an invited speakers session followed by the presentation of the six contributions. Each session is analyzed in a separate subsection below.

2.1 Invited talks

This session featured two invited talks. The first focused on indexing and search challenges in the area of very large visual archives. The second focused on interoperability challenges in the construction of a European data archive infrastructure, arising from the EUDAT project.

Amato in the talk entitled "Dealing with Very Large Visual Document Archives" presented state of the art, issues and open research directions related to content based retrieval in very large datasets of visual documents. Content based retrieval is typically performed searching by similarity on the visual (vectorial) features extracted from images. In the last decades researchers have investigated techniques for executing similarity search efficiently and in a scalable way, mostly based on extraction of global visual features [14]. Several techniques were presented, each resulting as an improvement of the existing ones: tree-based access, approximate similarity search, and permutation-based methods. Finally, approaches based on local visual features were presented. These offer much higher retrieval quality, but introduce efficiency issue which are orders of magnitude more difficult.

Thiemann in the talk entitled "The EUDAT Initiative: Challenges and Opportunities" presented EUDAT, a three-year project starting in October 2011 and funded through FP7 e-Infrastructure Call 9, Data infrastructure for e-Science. Its consortium consists of 23 partners from 13 countries and represents 15 user communities from a wide range of scientific disciplines. Emphasis is on development towards a Collaborative Data Infrastructure across scientific communities. The talk highlighted challenges and opportunities as been seen in the current data infrastructure landscape in Europe and addressed within EUDAT.

2.2 Presentation of contributions

This session included the presentations from the six contributions, which covered very large issues on digital libraries in combination with data archives. In particular, the majority of the accepted papers focused on problems related to large scale content storage and management.

In reference to large scale content storage, Jurik and Zierau in the paper entitled "*Different Mass Processing Services in a Bit Repository*", analyzed the requirements of a general bit repository mass processing service that should capable of abstracting over several programming models and platforms. The service is typically needed in large data archives and libraries, where different ways of doing mass processing is needed for different digital library tasks. The investigation shows that the execution environment has an heavy influence on mass processing requirements, hence a general purpose approach is only possible with respect to a given scenario, where common service parameters and organizational issues can be identified. Thompson, Bainbridge, and Suleman in the paper entitled "*Using TDB in Greenstone to Support Scalable Digital Libraries*", discussed about the issues affecting one of the most diffuse digital library software when dealing with large collections. In particular, they evaluated the behavior of the open source Greenstone digital library software when exploited in parallel tasks, identified a drawback residing in the database component and propose some strategies essentially based on the exploitation of a database supporting parallel access by obtaining significant benefits in terms of import time. Finally, Praczyk, Nogueras-Iso, Kaplun, and Šimko in the paper entitled "*A storage model for supporting figures and other artifacts in scientific libraries, the case study of Invenio*", presented an extension of the data storage model of Invenio, a software platform for building a web-based (document) repository developed at CERN. The extension addresses the requirements arising while extending INSPIRE, the information resource in High Energy Physics, to store figures and preserving data tables on which publications are based. Such requirements are in line with current digital libraries challenges to facilitate discovery and access to digital objects distinct from the traditional full-text documents, e.g. figures, data sets or software related to scientific developments.

With regard to large scale content management, Lemire and Vellino in the paper entitled "*Extracting, Transforming and Archiving Scientific Data*", proposed a scalable strategy for automatically addressing research-data problems, ranging from the extraction of legacy data to its long-term storage. The automation of these tasks faces three major challenges: (*i*) research data and data sources are highly heterogeneous, (*ii*) future research needs are difficult to anticipate, (*iii*) data is hard to index. To address these problems, the authors reviewed existing solutions in the business world and proposed the Extract, Transform and Archive (ETA) model for managing and mechanizing the curation of research data. Freitas and Ramalho in the paper entitled "*Relational Databases Conceptual Preservation*" addressed the digital preservation of relational databases by focusing on the conceptual model of the database, hence considering database semantics as an important aspect of preservation "property". This technique enhances previous approaches, which were based on raw format preservation of relational database data and structure. The method is based on Web Ontology Language (OWL) ontologies used to express database semantics as inferred by special algorithms devised by the authors into a prototype. Finally, Elbers and Broeder in the paper entitled "*Federating Live Archives*" described The Language Archive (TLA) infrastructure and its transition towards open federated archive environment by means of openness to novel metadata formats. In this federated archive environment both data and services are synchronized to multiple sites in order to provide long-term persistency on both levels. The change towards a new metadata format opens the possibilities for other domains to define their own metadata components and structure and thus join the infrastructure. However, this increase in flexibility requires a major update of the archiving and exploitation tools.

3. WORKSHOP DISCUSSION

As in the previous edition, the concluding brainstorming session confirmed the general agreement that a "very-large" Digital Library can be considered as such if any of the axes user management, content management, functionality management, and policy management becomes "very large" with respect to *volume*, *velocity*, or *variety*. This statement gives a particular flavour to this research field, which distinguished it from digital libraries and very large databases. In fact, a digital library with a small-size content base may be considered very large because of its large users base or because of its challenging evolving and unpredictable functional requirements. From this claim, the discussion moved towards the question "what does very-large mean w.r.t. the four axes or to any permutation of them?". Again, the discussion converged on believing that very-largeness is a matter of thresholds and hard-challenges which today shape the limits of our solutions and tomorrow will be hopefully tackled to evolve into newer and harder problems. Very large digital library foundations are still in an early stage and do not help in giving a formal specification to these challenges. As a matter of fact, VLDL limits, hence today's very-large issues, manifests themselves only in real-case scenarios, which researchers are still unable to classify w.r.t. a general theory of VLDL. Consequently, the same holds for the solutions proposed by researchers, which find it hard to confront their work with that of others. More generally, researchers cannot decide to tackle a problem of VLDL research starting from a broader perspective, given clearly stated and agreed on VLDL problematics.

In order to start this re-organization, the audience suggested to pursue a pragmatic approach by first trying to identify common "grand challenges" in the field and subsequently narrow the scope of research to a list of "focused challenges", over which researchers can measure their competences and compare ideas and solutions. This discussion will continue during the next year, through collaborative web tools and based on the volunteering work of researchers. The intention is for the Fifth Workshop on Very Large Digital Libraries to bear these grand and focused challenges as list of topics for article submission.

4. CONCLUSIONS

The main conclusion drawn from all workshop deliberations was that VLDL research has all the attributes to candidate as an independent research field. Not only, its DL flavour rotating around the four dimensions of users, content, functionality and policies, only overlaps in some sense with very large databases (on the content issues) and makes its particularly interesting and innovative in terms of challenges. It was agreed that next year's workshop questions should move one step forward. On the one hand into identifying, across the axes of investigation, a categorization of very large problems for DLs, and on the other hand to continue the quest on solutions to these issues. The LinkedIn group "Open Forum on Very Large Digital Libraries"[5] is today open for researchers willing to cooperate on the first task so as to pave the way to a more constructive confrontation on common research avenues next year, in the next edition.

5. ACKNOWLEDGMENTS

We would like to thank all those who contributed directly or indirectly to the event, especially our colleagues at ISTI-CNR Donatella Castelli and Pasquale Pagano for their advices.

Special thanks are also due to the members of the program committee: *Daan Broeder* (Max Planck Institute for Psycholinguistics, The Netherlands), *Kat Hagedorn* (OAIster System, University of Michigan Digital Library Production Service, USA), *Norbert Fuhr* (Department of Computer Science, University of Duisburg-Essen, Germany), *Leonid Andreevich Kalinichenko* (Institute of Informatics Problems of the Russian Academy of Science, Moscow), *Fabrizio Silvestri* (Institute of Information Science and Technologies, National Research Council, Italy), *Hussein Suleman* (Department of Computer Science, University of Cape Town, South Africa). Their active participation, enthusiasm and research experience largely contributed in making this workshop an attractive venue and in the end a constructive experience for all authors, participants and the community in the large. Finally, our sincere gratitude goes to all participants, invited speakers and authors, whose enthusiasm and vision constitute the soul of this series of workshops.

Workshop proceedings [5] were funded by ISTI-CNR.

6. REFERENCES

[1] G. Athanasopoulos, L. Candela, D. Castelli, K. E. Raheb, P. Innocenti, Y. Ioannidis, A. Katifori, A. Nika, S. Ross, A. Tani, E. Toli, C. Thanos, and G. Vullo. Digital Library Technology and Methodology Cookbook. Deliverable D3.4, DL.org, April 2011.

[2] C. L. Borgman. The conundrum of sharing research data. *Journal of the American Society for Information Science and Technology*, pages 1–40, 2011.

[3] L. Candela, G. Athanasopoulos, D. Castelli, K. E. Raheb, P. Innocenti, Y. Ioannidis, A. Katifori, A. Nika, G. Vullo, and S. Ross. The Digital Library Reference Model. Deliverable D3.2b, DL.org, April 2011.

[4] L. Candela, D. Castelli, Y. Ioannidis, G. Koutrika, P. Pagano, S. Ross, H.-J. Schek, H. Schuldt, and C. Thanos. Setting the Foundations of Digital Libraries – The DELOS Manifesto. *D-Lib Magazine*, 13(3/4), March/April 2007.

[5] L. Candela, Y. Ioannidis, and P. Manghi, editors. *Proceedings of the Fourth Workshop on Very Large Digital Libraries (VLDL 2011)*, Berlin, Germany, 2011. ISBN 978 88 95534 11 4.

[5] `http://www.linkedin.com/groups/Open-Forum-on-Very-Large-4118623`

[6] E. Commission. Communication from the commission to the european parliament, the council, the european economic and social committee and the committee of the regions a digital agenda for europe. Technical report, European Commission, August 2010.

[7] S. Gradmann, F. Borri, C. Meghini, and H. Schuldt, editors. *Research and Advanced Technology for Digital Libraries - International Conference on Theory and Practice of Digital Libraries, TPDL 2011, Berlin, Germany, September 26-28, 2011. Proceedings*, volume 6966 of *Lecture Notes in Computer Science*. Springer, 2011.

[8] Y. Ioannidis, P. Manghi, and P. Pagano, editors. *Proceedings of the Second Workshop on Very Large Digital Libraries (VLDL 2009)*, Corfu, Greece, 2009.

[9] Y. Ioannidis, P. Manghi, and P. Pagano, editors. *Proceedings of the Third Workshop on Very Large Digital Libraries (VLDL 2010)*, Glasgow, Scotland, UK, 2010. ISSN 1818-8044.

[10] P. Manghi, P. Pagano, and Y. E. Ioannidis. Second workshop on very large digital libraries: in conjunction with the european conference on digital libraries corfu, greece, 2 october 2009. *SIGMOD Record*, 38(4):46–48, 2009.

[11] P. Manghi, P. Pagano, and P. Zezula. First workshop on very large digital libraries – vldl 2008. *SIGMOD Record*, 37(4):115–117, 2008.

[12] P. Manghi, P. Pagano, and P. Zezula, editors. *Proceedings of the First Workshop on Very Large Digital Libraries (VLDL 2008)*, Aarhus, Denmark, 2008.

[13] C. Thanos. Global Research Data Infrastructures: The GRDI2020 Vision. Technical report, GRDI2010 `www.grdi2020.eu`, 2011.

[14] P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity Search - The Metric Space Approach*, volume 32 of *Advances in Database Systems*. Springer, 2006.

Call for Papers

Fifth International Workshop on Testing Database Systems (DBTest)

In conjunction with ACM SIGMOD/POCS Conference 2012
Scottsdale, USA

http://dbtest2012.comp.polyu.edu.hk

The annual Workshop on Testing Database Systems, DBTest, is the premier forum for researchers, practitioners, developers, and users to explore cutting-edge technology and exchange tools, techniques, knowledge, and experiences revolving around the testing of data management systems.
We invite the submission of original contributions relating to all aspects of testing of data management systems defined broadly.

Topics of interest include but are not limited to:

- Benchmarking and performance evaluation
- Cloud databases and multi-tenant data management
- Concurrency control
- Data stream management systems
- Database applications
- Distributed data processing frameworks (e.g., Hadoop)
- Generation of test artifacts (e.g. data generation)
- Randomized test methodologies
- Robust query processing
- Reliability and availability of database systems
- Software metrics and software engineering processes
- Test automation
- Tuning and calibration of complex systems
- Usability and maintainability
- War stories and field experience

In particular, we encourage the submission of vision papers as well as papers illustrating day-to-day challenges in industrial database application scenarios that will help bring practical issues to the attention of a larger research community.

All aspects of the submission and notification process will be handled electronically. Papers of up to six pages should be submitted via CMT at https://cmt.research.microsoft.com/DBTEST2012
For additional information and formatting guidelines, please visit the workshop website.

Paper Submission
March 12, 2012

Notification of Acceptance
April 5, 2012

Camera-ready due
April 22, 2012

Workshop
May 21, 2012

Program Committee
Carsten Binnig (DHBW)
Glenn Paulley (Sybase)
Guy Lohman (IBM)
Jayant Haritsa (IISC Bangalore)
Jens Dittrich (Univ. d. Saarlands)
Meikel Poess (Oracle)
Neoklis Polyzotis (UC Santa Cruz)
Rimma Nehme (Microsoft)
S Sudarshan (IIT Bombay)
Thomas Neumann (TU Munich)
Zhendong Su (UC Davis)

Steering Committee
Leo Giakoumakis (Microsoft)
Donald Kossmann (ETHZ)

Organization
Eric Lo (Hong Kong Polytechnic University)
Florian Waas (EMC/Greenplum)

Contact
ericlo@comp.polyu.edu.hk
florian.waas@emc.com