

# Database Workload Management

Edited by

Shivnath Babu<sup>1</sup>, Goetz Graefe<sup>2</sup>, and Harumi A. Kuno<sup>3</sup>

**1** Duke University, USA [shivnath@cs.duke.edu](mailto:shivnath@cs.duke.edu)

**2** HP Labs, USA [goetz.graefe@hp.com](mailto:goetz.graefe@hp.com)

**3** HP Labs, USA [harumi.kuno@hp.com](mailto:harumi.kuno@hp.com)

---

## Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 12282 “Database Workload Management”. Dagstuhl Seminar 12282 was designed to provide a venue where researchers can engage in dialogue with industrial participants for an in-depth exploration of challenging industrial workloads, where industrial participants can challenge researchers to apply the lessons-learned from their large-scale experiments to multiple real systems, and that would facilitate the release of real workloads that can be used to drive future research, and concrete measures to evaluate and compare workload management techniques in the context of these workloads.

**Seminar** 09.–13. July, 2012 – [www.dagstuhl.de/12282](http://www.dagstuhl.de/12282)

**1998 ACM Subject Classification** H.2 Database Management, H.2.4 Systems, H.2.7 Database Administration

**Keywords and phrases** database workload management, robust query processing, cloud computing, query execution, hadoop, application availability, performance modeling


**Digital Object Identifier** 10.4230/DagRep.2.7.73

## 1 Executive Summary

*Shivnath Babu*

*Goetz Graefe*

*Harumi A. Kuno*

**License**  Creative Commons BY-NC-ND 3.0 Unported license  
© Shivnath Babu, Goetz Graefe, and Harumi A. Kuno

## Introduction

Much database research focuses on improving the performance of individual queries. Workload management focuses on a larger question – how to optimize the performance of the entire workload, as a whole. Workload management is one of the most expensive components of system administration. Gartner listed workload management as the first of two key challenges to emerge from the data warehouse market in 2009. However, we believe that even while both researchers and industry are building and experimenting with increasingly large-scale workloads, there is a disconnect between the OLTP/OLAP/Mixed/Hadoop/Map-Reduce workloads used in experimental research and the complex workloads that practitioners actually manage on large-scale data management systems.

One goal of this seminar was to bridge this gap between research and practice. Dagstuhl Seminar 12282 provided a venue where researchers can engage in dialogue with industrial participants for an in-depth exploration of challenging industrial workloads, where industrial participants can challenge researchers to apply the lessons-learned from their large-scale



Except where otherwise noted, content of this report is licensed under a Creative Commons BY-NC-ND 3.0 Unported license  
Database Workload Management, *Dagstuhl Reports*, Vol. 2, Issue 7, pp. 73–91  
Editors: Shivnath Babu, Goetz Graefe, and Harumi Anne Kuno



Dagstuhl Reports  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

experiments to multiple real systems, and that would facilitate the release of real workloads that can be used to drive future research, and concrete measures to evaluate and compare workload management techniques in the context of these workloads.

With regard to seminar participants, we took a system-centric focus, and invited participants who could speak to the management of workloads in a variety of systems. Seminar participants came from a variety of academic and commercial institutions: Cloudera, EMC/-Greenplum, LinkedIn, Microsoft, MIT, National University of Singapore, NEC, Queen's University, Stony Brook University, Teradata, Tokutek, TU Berlin, TU Ilmenau, TU München, UC Berkeley, Universität des Saarlandes, Universität Hamburg, University of Waterloo, and Yahoo.

## 2 Table of Contents

<b>Executive Summary</b>	
<i>Shivnath Babu, Goetz Graefe, and Harumi A. Kuno</i> . . . . .	73
<b>Overview of the week</b> . . . . .	76
<b>Monday: Individual presentations</b> . . . . .	76
Monday morning: commercial workload management . . . . .	76
Monday afternoon: benchmarks and workloads . . . . .	77
<b>Summarizing Challenges and Open Questions from Commercial Systems</b> .	78
Workload Characterization/ Performance Modeling . . . . .	78
Workload Isolation . . . . .	79
Scheduling Workloads for Concurrency . . . . .	81
<b>Tuesday: individual presentations</b> . . . . .	82
Tuesday morning: performance modeling/prediction, algorithms . . . . .	82
Tuesday afternoon: mixed workloads and robust query processing . . . . .	82
<b>Working Groups (Wednesday, Thursday, Friday)</b> . . . . .	83
Towards a Benchmark for Workload Management . . . . .	83
Scheduling Workloads for Concurrency . . . . .	85
Provably Good Scheduling for Database Workload Management . . . . .	85
Workload Characterization/ Performance Modeling . . . . .	86
Eliminating memory knobs . . . . .	87
Cache-Adaptive Algorithms for Databases . . . . .	88
Dataflow programming atop Cilk . . . . .	89
<b>Post-seminar outcomes</b> . . . . .	89
<b>Participants</b> . . . . .	91

### 3 Overview of the week

The week was structured around producing the following artifacts:

1. Descriptions of the most significant database workload management challenges facing industry, each defined in terms of a rough specification of a target workload and its objectives.
2. For some number of these challenges, a sample workload that would demonstrate the challenge, and that would allow solutions to the challenge to be validated and compared.
3. Descriptions of the "best" workload management techniques and best practices (both proven and unproven) that might apply to these challenges (both in practice and in research), as well as a partially-annotated bibliography that lists the papers that discuss those techniques and that summarizes their potential benefits and limitations.
4. Identify new synergies and opportunities for new techniques and new applications of existing techniques.

On Monday, industry participants spoke about the top workload management challenges commercial systems face, and other participants reviewed ongoing work on benchmarks that expose those challenges. Tuesday featured a series of presentations surveying the current state of the art in terms of research. On Wednesday, participants formed break out groups and considered the commercial challenges and how best to cast them as challenges that researchers could address. The break-out groups reformed on Thursday, and sketched paper abstracts of research papers. On Friday morning, these abstracts were presented.

### 4 Monday: Individual presentations

On Monday, we focused on commercial workload management systems, their open challenges, and ongoing work on benchmarks that expose those challenges.

#### 4.1 Monday morning: commercial workload management

Teradata has the possibly most well-established and sophisticated workload management system of any commercial database system, as evidenced by a recent keynote that credits Teradata's workload management as a key component of eBay's success. It was thus fitting that *Douglas Brown* from *Teradata* kicked off the first day with a presentation about Teradata's workload management capabilities.

*Rao Kakarlamudi* from *HP* presented an overview of HP Seaquest's workload management system. The HP Workload Management Services (WMS) provides the infrastructure to help you manage query workloads and system resources in the mixed workload environment of a SeaQuest Data Warehousing Platform for Business Intelligence. In the mixed workload environment of an enterprise data warehouse, like the SeaQuest platform, a variety of query workloads must run smoothly without interruptions to their throughput and performance. As a database administrator, you need to be able to control when different types of queries enter the system and how they proceed to execute, and you need to ensure that system resources remain available for query execution. You do not want a rogue query to monopolize system resources and slow down or prevent other queries from running in the system. Using WMS, you can influence when queries run and how many system resources they are allowed

to consume by assigning groups of queries (that is, query workloads) to WMS services. You can create your own services in WMS and configure them to have a relative priority and a set of thresholds for the execution environment. That way, you can maintain different query workloads and ensure that enough system resources are available to higher priority workloads while minimizing contention with lower priority workloads.

*Sivaramakrishnan Narayanan* from *EMC Greenplum* spoke about workload management at EMC Greenplum.

*Russell Sears* talked about his work at *Yahoo*: This talk summarizes Yahoo!'s current and next-generation data serving infrastructure. We provide an overview of three systems, Mobius, Walnut and bLSM, which promise to enable new classes of applications and to significantly improve the performance of Yahoo!'s existing workloads. However, fully leveraging these systems will require solutions to new workload management problems. We present the problems, and suggest new approaches to workload management. Mobius allows the workload manager to impact user-visible application semantics as well as performance SLAs. Walnut is designed to leverage extreme mismatches in system scale to provision services on spare capacity. Finally, unlike many data management systems, bLSM is designed to run in carefully provisioned environments, allowing its latency and throughput to be predicted directly from hardware parameters. We conclude the talk with a discussion of recent hardware trends.

*Robert Chansler* from *LinkedIn* spoke about LinkedIn Workflows In Large Hadoop Clusters. A workflow might be a coordinated collection of 20 job where each job might have a 1000 tasks (maps or reduces). A large Hadoop cluster will see thousands of jobs each day. For a carefully configured job, the number of tasks will be proportional to the input size (1 task 500 MB). Today the management of work load is limited to simple assignment of tasks to a small number of queues. Some conventional management tools are problematic for Hadoop—throttling and preemption have never been satisfactorily implemented

*Archana Ganapathi*, from *Splunk*, spoke about her experiences with managing Big Data Workloads in Splunk.

*Jingren Zhou* spoke about *Microsoft's* Bing infrastructure and workloads.

## 4.2 Monday afternoon: benchmarks and workloads

In the afternoon, we heard about benchmarks and workloads. *Kai-Uwe Sattler* from *TU Ilmenau* spoke about the Tractor Pulling for DBMS Benchmarking, which began as a collaboration into benchmarking robust query processing at the 2010 Dagstuhl Seminar 12282 on Robust Query Processing and presented at DBTest 2011.

*Michael Seibold* from *TU München* spoke of the Mixed Workload CH-benCHmark from the Technical University of Munich. While standardized and widely used benchmarks address either operational or real-time Business Intelligence (BI) workloads, the lack of a hybrid benchmark led us to the definition of a new, complex, mixed workload benchmark, called mixed workload CH-benCHmark. This benchmark bridges the gap between the established single-workload suites of TPC-C for OLTP and TPC-H for OLAP, and executes a complex mixed workload: a transactional workload based on the order entry processing of TPC-C and a corresponding TPC-H-equivalent OLAP query suite run in parallel on the same tables in a single database system. As it is derived from these two most widely used TPC benchmarks, the CH-benCHmark produces results highly relevant to both hybrid and classic single-workload systems. Like the "Tractor Pulling" benchmark, this work was initiated at

the 20101 Dagstuhl Seminar 12282 on Robust Query Processing and presented at DBTest 2011.

Finally, *Yanpei Chen* from *UC Berkeley* (he has since joined *Cloudera*) gave two presentations about on Hadoop/MapReduce workloads.

*Interactive Analytical Processing in Big Data Systems: A Cross Industry Study of MapReduce Workloads.* Within the past few years, organizations in diverse industries have adopted MapReduce-based systems for large-scale data processing. Along with these new users, important new workloads have emerged which feature many small, short, and increasingly interactive jobs in addition to the large, long-running batch jobs for which MapReduce was originally designed. As interactive, large-scale query processing is a strength of the RDBMS community, it is important that lessons from that field be carried over and applied where possible in this new domain. However, these new workloads have not yet been described in the literature. We fill this gap with an empirical analysis of MapReduce traces from six separate business-critical deployments inside Facebook and at Cloudera customers in e-commerce, telecommunications, media, and retail. Our key contribution is a characterization of new MapReduce workloads which are driven in part by interactive analysis, and which make heavy use of query-like programming frameworks on top of MapReduce. These workloads display diverse behaviors which invalidate prior assumptions about MapReduce such as uniform data access, regular diurnal patterns, and prevalence of large jobs. A secondary contribution is a first step towards creating a TPC-like data processing benchmark for MapReduce.

*We Don't Know Enough to make a Big Data Benchmark Suite — An Academia-Industry View.* This is a position paper that comes from an unprecedented empirical analysis of seven production workloads of MapReduce, an important class of big data systems. The main lesson we learned is that we do not know much about real life use cases of big data systems at all. Without real life empirical insights, both vendors and customers often have incorrect assumptions about their own workloads. Scientifically speaking, we are not quite ready to declare anything to be worthy of the label “big data benchmark.” Nonetheless, we should encourage further measurement, exploration, and development of stopgap tools.

## 5 Summarizing Challenges and Open Questions from Commercial Systems

In the late afternoon, we broke into groups and focused on drilling down, categorizing and prioritizing the challenges we'd heard about from commercial systems. These challenges are captured below.

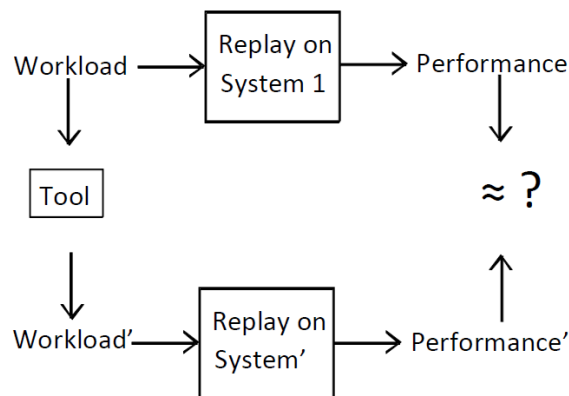
### 5.1 Workload Characterization/ Performance Modeling

*Doug Brown, Yanpei Chen, Jens Dittrich, Archana Ganapathi, Harumi A. Kuno, Barzan Mozafari, Awny Al-Omari, Norbert Ritter, Y.C. Tay*

#### 5.1.1 Industry challenges

1. What will be the impact of expected growth and planned changes?
2. How to set realistic service level goals (SLGs)?
3. How to change a workload's priority (relative weight) to meet SLGs?

4. How to set workloads' concurrency levels (TASM throttling) to meet SLGs?
5. How to justify tuning measures to meet SLGs?
6. How to predict new application implementation impact?
7. How to justify hardware upgrades required to meet SLGs?
8. How to compare actual performance with expected?
9. Memory vs. CPU isolation
10. Empirical validation
11. How to model all models
12. A tool to take a real workload and scale it up or down with "relativity" (specifics TVD).
13. How to achieve resource isolation between workloads.



■ **Figure 1** Sketch by the workload characterization/performance modeling breakout group summarizing the underlying challenge of workload characterization / performance modeling.

### 5.1.2 Top priorities (workload management)

1. How to change workload's priority (relative weight) to meet SLGs?
2. How to set workloads' concurrency level to meet SLGs?
3. How to predict new application's implementation impact?

### 5.1.3 Top Priorities (capacity planning)

1. How to change workload's priority (relative weight) to meet SLGs?
2. How to achieve resource isolation between workloads.
3. How to predict new application implementation impact?

## 5.2 Workload Isolation

*Ashraf Aboulnaga, Shivnath Babu, Robert Chansler, Hakan Hacigümus, Rao Kakarlamudi, and Michael Seibold*

The group defined workload isolation as:

1. *Space multiplexing of resources.* Each of the N workloads achieves the same performance that it would if it were running alone with unlimited computing resources

2. *Time multiplexing of resources.* Each of the N workloads is promised a share of the available computing resources, and it gets at least this share of the resources
3. *Application view of performance.* Each of the N workloads meets its SLOs

### 5.2.1 Research challenges

1. Level 1: Given 1 workload and its application level SLO, give me the resource requirements of this workload
2. Level 2: Given N workloads and application level SLOs, give me the resource requirements of each workload
3. Level 3: Given N workloads and application level SLOs that vary over time, give me the resource requirements of each workload
4. Workload Isolation Objective: Meet all requirements with minimum computing resources
  - It is easy to isolate workloads by overprovisioning

### 5.2.2 Open questions

1. What actions can a system take to ensure workload isolation?
  - Data replication, admission control, resource allocation, preemption, time multiplexing vs. space multiplexing
  - Not all actions applicable to all systems
2. Is virtualization a solution out of the box?
  - Dealing with multiple levels of virtualization that do not match with each other
3. Can elasticity be used as a mechanism to address overprovisioning?
  - How to provide elasticity in data intensive systems
  - Elasticity helps with changing workloads and SLAs
4. How to map from application level SLO to resource demands?
5. Quantifying and modeling the degree/impact of performance isolation.
  - Resource based metrics
  - SLA based metrics
6. Is data locality overrated given new developments in network
  - Inter-cluster bandwidth has gone up
  - Infiniband is becoming more popular
7. Can we just assume that data is spread across all available nodes and focus on compute-level isolation?

### 5.2.3 Benchmarking Workload

- Analytical workload
- SQL or MapReduce both possible
  - Capability and complexity (SQL) vs. predictability (MapReduce)
  - TPC-H, WordCo-occurrence
- Vary the number of queries/jobs from 1 to N
  - Control the amount of data and degree of overlap in data access (e.g., by varying a filter on a time attribute)
  - Control degree of overlap in arrival times
  - Control degree of similarity between workloads



### 5.3 Scheduling Workloads for Concurrency

*Kai-Uwe Sattler and others*

This break-out group considered the top-three challenges in the area of scheduling theory, performance prediction, and no-knob data management.

1. Scheduling theory
  - Goal: provide a performance theorem for: Given a DAG in which the nodes have CPU work  $W$  & I/O work  $I$  running on a machine with  $P$  processors &  $D$  disks, a simple scheduler can run the DAG with good performance and memory bounds
  - Why:
    - Theoretical foundation for scheduling
    - Provable property
  - How to test:
    - By proving
    - Implement and validate the performance against the theorem
  - Why:
    - Theoretical foundation for scheduling
    - Provable property
  - Challenge: In analogy with Brent's theorem which states that a graph with work  $W$  and span  $S$  running on  $P$  processors can run in time less than  $W/P + S$ . Extras (may be harder):
    - $P$  and  $D$  change dynamically.
    - Processor and disk speeds change dynamically.
    - What is the locality problem for this kind of computation?
    - Given several such DAGs, how do you execute them concurrently (and what kind of bounds do you get)?
    - The DAG unfolds dynamically.
2. Performance prediction.
  - Goal: predict performance of individual requests as well as the overall workload
  - Why: required for scheduling, but difficult — see 30 years of research on query optimizers
  - No knobs
  - How to test: compare predictions to real execution times.
3. No knobs.
  - Goal: Eliminate tuning knobs in WLM (memory, parallelism, MPL, ...).
  - Why: Base for other techniques; cost issues, corrective actions.
  - First Step: Start with eliminating memory knobs (e.g., memory-adaptive sorting).
  - How to test: Compare with gold standard in static and dynamic settings.

For workloads, see powerpoint slideset in seminar materials.

## 6 Tuesday: individual presentations

Tuesday focused on research efforts relevant to the challenges highlighted on Monday.

### 6.1 Tuesday morning: performance modeling/prediction, algorithms

*Archana Ganapathi*, from *Splunk* but speaking of her work at *UC Berkeley*. Archana spoke about two topics: 1. Using Statistical Machine Learning to predict performance for a parallel database system. 2. Workload Synthesis and Replay for MapReduce.

*Y.C. Tay* from *National University of Singapore* presented his paper *Bottleneck Analysis for Cloud Transaction Architectures*. At the SIGMOD 2010 conference, Kossman, Kraska and Loesing presented an experimental comparison of four cloud architectures for transaction processing. The paper concluded that “It is still unclear whether the observed results are an artifact of the level of maturity of the studied services or fundamental to the chosen architecture”. This issue is addressed here via a theoretical analysis that focuses on the bottleneck in each architecture.

*Jens Dittrich* from *Universität des Saarlandes* presented his work on Hadoop++/HAIL.

*Norbert Ritter* from *Universität Hamburg* presented an alternative solution to the problem of DBS self-management, which avoids the drawbacks of the existing self-management functions (as described in the PhD thesis of Marc Holze). Instead of extending a DBMS with a set of component-specific self-management functions, the developed solution is designed as one single self-management loop, which has a system-wide view on all configuration decisions. As long as the workload of the system does not change and the goals are met, this self-management loop only performs very light-weight monitoring operations on the workload, performance, and state information. For this purpose, a workload shift detection solution is provided. This also comprises a workload classification solution, that groups similar workload events in order to further reduce the monitoring overhead. Furthermore, the workload information is analysed for cyclic patterns in order to predict upcoming workload shifts. Only when the system-wide self-management solution detects a workload shift or a violation of the goals, it performs a heavy-weight reconfiguration analysis. Given the current workload and state of the DBS, this reconfiguration analysis has to derive a new set of configuration parameter values that meet the goals in the best possible way. For this purpose the system-wide self-management solution employs a system model, which quantitatively describes the behaviour of the DBS using mathematical models. At runtime, the system model is evaluated by the self-management logic using multi-objective optimization techniques, where the goal values are represented as constraints.

### 6.2 Tuesday afternoon: mixed workloads and robust query processing

*Wendy Powley* from *Queen’s University – Kingston* provided an overview of the techniques and approaches that her research group has taken to workload management in DBMSs over the past 10 years.

*Ashraf Aboulnaga* from *University of Waterloo* provided an overview of his research on workload management, focusing on (1) query interactions, (2) workload management for Hadoop, and (3) virtualization.

*Bradley Kuzsmaul* (from MIT and Tokutek) and *Michael Bender* (from State University of New York at Stony Brook and Tokutek) spoke of Cilk, work stealing, write-optimized data structures, and scheduling algorithms.

*Goetz Graefe* from HP Labs spoke about robust query processing.

*Kostas Tzoumas* from TU Berlin spoke of his group's work on Stratosphere.

*Shivnath Babu* from Duke University presented "Perspectives on MapReduce Workload Management." Abstract: This talk gives an overview of problems that arise in workload management for MapReduce systems. We first discuss the reasons why the importance of MapReduce workload management has grown rapidly in recent years. We then present the preliminary techniques being used in the industry today for this problem, and why these techniques are inadequate. With the goal of laying out a research agenda, the talk concludes with an abstraction of the various aspects involved in MapReduce workload management.

## 7 Working Groups (Wednesday, Thursday, Friday)

For the second half of the week, participants worked in break out groups and considered the commercial challenges and how best to cast them as challenges that researchers could address.

### 7.1 Towards a Benchmark for Workload Management

*Ashraf Aboulnaga, Awny Al-Omari, Shivnath Babu, Robert J. Chansler, Hakan Hacigumus, Rao Kakarlamudi, and Michael Seibold*

**Summary:** This breakout group focused on (i) developing a framework for benchmarking workload management techniques, (ii) coming up with some example instantiations of this framework, and (iii) identifying use cases in which the proposed benchmark can be of value.

The focus was on benchmarking analytical or decision support systems, as opposed to transactional systems. The typical workflow in such systems is that data comes in through one or more feeds that are loaded into the system, and users submit analytical queries that are executed on the available data. The system must provide desired guarantees (SLAs) on data freshness (how fast is data loaded) and query response times. The system has to maintain these guarantees in the face of fluctuations in the workload and failures of the infrastructure; which is the task of workload management.

Given the above usage scenario, the benchmark framework that we developed requires benchmark creators to define the following six pluggable components:

1. Data feeds
2. Query workloads
3. SLAs and penalties for violating them
4. Failures (optional)
5. Temporal patterns for the arrival of data and queries, and for system failures
6. Scaling rules

A benchmark defined through this framework evaluates how well a workload management solution fulfills the requirements of the presented data feeds and query workloads while minimizing total cost. Thus, the benchmark measures performance metrics such as throughput and latency, and how they vary over time (a time series). The benchmark also measures

the number of SLA violations and the total cost of the system, defined as  $\text{Cost } C = R + P$ , where  $R$  is the cost of the infrastructure and  $P$  is the cost of SLA violation penalties.

**Temporal arrival patterns:** A key feature of the benchmark framework is that the arrival of data and queries is described by temporal arrival patterns. These temporal patterns are also used to describe when failures happen. The temporal patterns can be deterministic or probabilistic. If they are probabilistic, then they are generated prior to a benchmark run to ensure repeatability. The patterns can be uniform or bursty. If they are bursty, then it is important to control whether the peak arrival rates happen in a correlated fashion (e.g., simultaneously) or independently. Simultaneous peaks stress workload management solutions.

**Data feeds:** Benchmark data may have some schema, which needs to be specified. A benchmark also has one or more data feeds. The data has to have a temporal dimension (a timestamp attribute) and has to be generated in increasing timestamp order. The data that appears in the feeds is described by the distribution for data generation (uniform or skewed, data feeds correlated or not, etc.).

The arrival pattern of the data is specified by a temporal arrival pattern. The temporal arrival pattern can specify two types of data loading: batch loading which happens every once in a while (e.g., initial loading into the system or periodic updates to dimension tables), and continuous streams (e.g., updating the fact tables). Batch loading would be described by a spike in the temporal arrival pattern.

**Query workloads:** A benchmark has one or more query workloads. Each workload consists of a stream of analytical queries (simple queries for tactical workloads and more complex queries for reporting) that read controlled amounts of data. The queries, or at least a significant fraction of them, must have a temporal predicate that restricts them to touching some time window of the data. The arrival of queries on the different workloads is defined by temporal arrival patterns, one per workload.

Queries have SLAs on how fast they need to be processed (query completion time). The specification of the queries also controls how much data overlap there is, which can have a significant effect on workload management, and how fresh the data needs to be, which places a constraint on data loading.

**SLA penalties:** SLAs are defined by a penalty function for each query type and each workload. The penalty function defines different penalties for missing different response time requirements. Different penalty functions allow us to express different priorities for workloads. For example, a penalty function in which the penalty is infinite means that we cannot under any circumstance violate the SLA. In general, there are types of workloads where there is significant (even if not infinite) penalty for violating the SLA, and others where violating the SLA is less harmful.

The cost of SLA violation is included in the total cost that is reported in the benchmark score. Another option is to count SLA violations and report them in the benchmark score.

**Data overlap:** Different workloads are more likely to affect each other if they touch the same data. One way to control data overlap is to have different queries touch data in different time windows. For example, we can have two workloads that both always touch the most recent data, which creates a high degree of overlap. On the other hand, we can have one workload that touches the most recent data and another workload that touches some fixed data in the past, which results in zero overlap. It may also be possible to control data overlap in other ways, for example by having queries look at the same tables or different tables.

**Freshness:** The correlation between the query arrival patterns and the data arrival patterns controls how fresh the data has to be. If a query is supposed to see some recent

data, and this data has arrived but not been loaded yet, then the query will return a wrong answer.

To give the system some slack in the time to load the data, the benchmark queries should be allowed to touch queries only  $F$  seconds in the past or older. That is, the time window predicates that are supposed to touch the most recent data in the different queries are of the form: “WHERE timestamp BETWEEN pasttimestamp AND NOW() –  $F$ ”. The higher the  $F$ , the more flexibility a system has in delaying the load of new data. In that sense,  $F$  is a freshness target.

A benchmark needs to specify a penalty cost that is incurred by a query if it does not see recent data that it is supposed to see. We can determine when this happens because the correct answer of each query – or how to obtain this correct answer – is part of the benchmark specification. The penalty cost for violating freshness is added into the total cost reported in the benchmark score. An infinite penalty cost means that a system is not allowed to violate freshness.

Failures: In today’s world, it is important to consider failures when talking about workloads and workload management. A lot of work these days goes into making systems fault tolerant, and systems are often overprovisioned or geographically replicated to enable them to handle failures. For example, if you consider a system that is mirrored for high availability, such a system is incurring a cost overhead of 100

We envision two ways to incorporate failures into the benchmark:

1. The benchmark can specify a temporal arrival pattern for failures chosen from a specific set of failure types (e.g., node and rack failures). For example, “After 30 minutes of running the benchmark, a rack fails for 10 minutes and then recovers.” The issue with this strategy is that specifying the meaning of failure and recovery from failure is not easy. Pulling the plug on the entire cluster is easy to understand, but what does it mean to lose 30

2. Instead of specifying failure types, the benchmark specifies the effect of these failures on the workload. For example, “After 30 minutes of running the benchmark, the data feed becomes unavailable for 10 minutes and, for the next 20 minutes the workload is double the past average as the system catches up.”

A benchmark needs to specify the expected behavior of data feeds and query workloads during the failures. It is recommended that the data feeds continue while the system is failed. Queries can either continue arriving and incur a penalty for failing, or we can model a case in which the query streams stop when the system fails.

Scaling rules: A benchmark must specify scaling rules. A simple way to do scaling is to say that each component (data, queries, arrival patterns) scales independently. However, we acknowledge that more elaborate scaling rules may be needed.

The next steps for the breakout group are to come up with some example instantiations of this framework, and to identifying use cases in which the proposed benchmark can be of value.

## 7.2 Scheduling Workloads for Concurrency

## 7.3 Provably Good Scheduling for Database Workload Management

*Bradley C. Kuszmaul (ed); Breakout Session B.*

**Problem 1.** How should you schedule the DAG that includes CPU work  $W$  and I/O work  $I$  on a machine with  $P$  processors and  $D$  disks? What performance and space guarantees can

you make? These variants may be useful: P and D may change dynamically. What if the processors or disks change speed dynamically? What is the locality problem for this kind of computation? Given several such DAGs, how do you schedule them when they are running at the same time? The DAG unfolds dynamically (as in Cilk).

**Definition 2.** The span of a DAG is the length longest path through the graph. (Sometimes this is referred to as depth or critical-path length.)

**Definition 3.** The span of the work of a graph G, written SW, is the length of the longest path through the graph, counting only the CPU work.

**Definition 4.** The span of the I/O of a graph G, written SI, counts only the I/O work.

**Conjecture 5.** A greedy schedule achieves time  $O(W=P+I=D+SW +SI)$ , where I/O and CPU are both measured in units of time. Idea for a proof. The idea is to extend Brent-Graham as follows: At any time step at least one of the following is true: half the processors are busy, half the disks are busy, or there are some idle processors and idle disks. The number of time steps that half the processors are busy is at most  $2W=P$ . The number of time steps that half the disks are busy is at most  $2I=D$ . If more than half the disks are idle and half the processors are idle, then ... Application 6. If a query optimizer has several query-plan choices, it needs to be able to estimate the run time of each plan. If we start running queries using work stealing, we need a way to predict the performance. Conjecture 5, if true, may provide a way for the user or the query planner to predict the performance of a plan on a machine with D disks and P processors.

## 7.4 Workload Characterization/ Performance Modeling

*Doug Brown, Yanpei Chen, Jens Dittrich, Archana Ganapathi, Harumi A. Kuno, Barzan Mozafari, Awmy Al-Omari, Norbert Ritter, Y.C. Tay*

Database vendors, developers, and administrators need to model and predict database performance given a workload and a system configuration in order to perform tasks such as workload management, capacity management, and system sizing. However, lack of data about workload characterization and workload performance is a significant obstacle for researchers working on modeling and prediction. Our breakout session produced a table showing functional workload types, logical workload characteristics that could be used to describe those types, and physical performance characteristics that could be observed from actual running workloads. We also proposed two tools. First, we proposed a workload characterization editor that could take an existing workload characterization and then manipulate one or more logical workload characteristics, producing a new workload characterization. We proposed that this tool could be made open and published, along with a repository of editable workload characterizations. Second, we proposed a tool that could take the resulting workload characterization and run the described workload. Finally, we described four use cases of how the workload editor could be used: (1) how to anticipate the performance impact of anticipated workload growth, (2) how to anticipate the performance impact of a workload throttling mechanism on a given workload; (3) how to compare the performance for two potential vendors (or systems); and (4) how to anticipate the performance impact of a new application.

A subset of the group co-authored a paper pursuing this idea further.

## 7.5 Eliminating memory knobs

*Goetz Graefe, Wendy Powley, Kai-Uwe Sattler, Kostas Tzoumas, Jingren Zhou*

One of the tasks of workload management is to allocate resources to consumers with conflicting demands. Systems typically contain “knobs” that dictate resource allocation, for example how much memory should be allocated to the buffer pool, how much memory should be allocated to sorting, etc. Eliminating knobs for tuning data management systems is an important task to reduce maintenance costs and make the systems more usable to standard customers. Eliminating knobs makes the problem of workload management significantly easier by removing free variables.

We consider the problem of memory tuning in such systems as a building block towards this overall goal: given an externally defined amount of memory (which may vary over time), how should this memory be internally distributed among different heterogeneous consumers? Using a basic model taking utility functions of the different consumers into account, we formulate this problem as an optimization problem and present scenarios as well as metrics for comparing and evaluating different strategies. Furthermore, we argue that consumers should be memory-adaptive in order to provide a diminishing marginal utility function which simplifies the optimal memory distribution significantly. Finally, we discuss how this can be achieved for typical memory consumers in a database system.

Different memory consumers that we aim to model in the context of a DBMS are the buffer pool (including the case of several buffer pools for multiple page sizes, devices, tables vs. indexes, etc.), query execution (including sorting, hashing, exchange, etc), procedure cache (compiled query execution plans and scripts), metadata, database utilities (e.g., load, reorganization, compression, index creation, statistics/histograms, etc), log buffers, query optimizer, complex UDFs (DB2 application memory and database heap), and UDFs in garbage-collected languages such as C# or Java [1, 2].

A utility function describes the quantification of the performance for a given amount of resource. In the context of memory management, we can quantify performance as the reduction in IO rate. Utility functions are needed as an indication for performance prediction to trade resource allocations between different consumers. It is important that utility functions for all memory consumers are comparable, i.e., they are expressed in the same units. IO rate reduction can be predicted, e.g., in the case of query optimizer by predicting the expected reduction in execution plan cost if further exploration will take place.

A beautiful utility function obeys Gossen’s first law, i.e., is a monotonically increasing function with decreasing marginal gain. With beautiful utility functions, a simple strategy that trades resources based on possible gains converges to a solution that optimizes total utility [3]. Examples of non-beautiful utility functions include linear functions, step functions, and non-continuous functions. In the context of a DBMS, typical examples of operators with non-beautiful utility functions are sorting without graceful degradation, non-hybrid hashing, and LRU replacement strategy without scan protection.

Some memory consumers may be simply served by virtual memory paging to shrink their memory allocation. For other consumers, the internal data structures need to be adjusted. One research direction is changing the database engine in order to make utility functions beautiful. A core strategy for making utility functions beautiful is applying graceful degradation, i.e., the gradual and incremental use of page eviction, while retaining as much state as possible in the available memory. This enables a graceful transition from an in-memory to an external memory algorithm in presence of pressure. Examples include hybrid hashing [4] and adaptive-memory sort. For query optimization, an option would be a resource-guided query optimization enumeration strategy that prunes plans based on memory

constraints. Finally, for data exchange in parallel DBMSs [5], a possible strategy would be to reduce the network buffers using hierarchical partitioning [7].

A second research direction is formalizing and solving the allocation problem. We can formalize memory distribution as an optimization problem:

We maximize the overall utility (possibly weighted to express externally-dictated priorities) minus the cost of adjustment subject to the externally dictated memory constraints. Assuming that utility functions are beautiful and all weights are equal to one, the problem can be solved by a simple trading approach. While one consumer's pain is less than another consumer's gain, we trade memory. The cost of trading is the aggregated utilities penalties over the adjustment period. To avoid expensive oscillation a minimum gain/pain can be imposed.

Previous work in adaptive memory management [6] notes "[...] the difficulty in determining the suitable experiment to test the efficiency of automatic memory tuner," and "the absence of any standard metric for evaluation..." A further research direction is devising scenarios for testing the performance, scalability, and robustness of various methods that (claim to) eliminate memory knobs. A final research direction is generalizing the problem to include other resources. While our outset is memory knobs, this work can be possibly generalized to other knobs on data processing systems (such the degree of parallelism). Two issues must be addressed by this generalization. First, appropriate utility functions for the resource usage have to be defined. Second, the algorithms have to be adapted to graceful degradation strategy for dealing with changing resources.

## References

- 1 Dominic Battré, Stephan Ewen, Fabian Hueske, Odej Kao, Volker Markl, Daniel Warneke: Nephelē/PACTs: a programming model and execution framework for web-scale analytical processing. SoCC 2010:119-130
- 2 Ronnie Chaiken, Bob Jenkins, Per-Åke Larson, Bill Ramsey, Darren Shakib, Simon Weaver, Jingren Zhou: SCOPE: easy and efficient parallel processing of massive data sets. PVLDB 1(2):1265-1276 (2008)
- 3 Diane L. Davison, Goetz Graefe: Dynamic Resource Brokering for Multi-User Query Execution. SIGMOD 1995:281-292
- 4 Goetz Graefe: Query Evaluation Techniques for Large Databases. ACM Comput. Surv. (CSUR) 25(2):73-170 (1993)
- 5 Goetz Graefe: Encapsulation of Parallelism in the Volcano Query Processing System. SIGMOD 1990:102-111
- 6 Adam J. Storm, Christian Garcia-Arellano, Sam Lightstone, Yixin Diao, Maheswaran Sundarand: Adaptive Self-tuning Memory in DB2. VLDB 2006:1081-1092
- 7 Jingren Zhou, Nicolas Bruno, Wei Lin: Advanced partitioning techniques for massively distributed computation. SIGMOD 2012:13-24

## 7.6 Cache-Adaptive Algorithms for Databases

*Michael A. Bender and Siva Narayanan (Eds.) Breakout session B.*

Many commercial database execution engines reserve memory for operations statically and offer no opportunity to change this at runtime. This causes under- utilization and queries go slower than then can. Cache-adaptive algorithms offer a glimpse of hope.

We introduce class of cache-adaptive algorithms. Cache-adaptive algorithms adapt to memory parameters that change over time, e.g., the available memory  $M$  and the block-transfer



size  $B$ . We exhibit optimal cache-adaptive algorithms for sorting, matrix multiplication, and sort-merge-join and hash join.

We first prove that so-called cache-oblivious algorithms are also cache-adaptive. Cache-oblivious algorithms are memory-hierarchy universal, i.e., the same algorithm works simultaneously on all memory-hierarchies and with no memory-specific parameterization. We then exhibit algorithms, whose parameters depend on  $B$  and  $M$ , and show that they are adaptive.

We first model how  $B$  and  $M$  are allowed to change. We then give a formal definition what it means to be optimal.

**If** we can prove this:

- Query execution engine can adapt its memory usage.
- Utilization can be maximized.
- Workload management can be more intelligent about memory as a resource.

## 7.7 Dataflow programming atop Cilk

*Russell Sears, Bradley Kuzsmaul, Michael Bender (and others?).*

Cilk is a superset of the C++ programming language for parallel programming tasks. It introduces two new keywords, "spawn" and "sync" that allow the runtime environment to optionally run portions of a computation in parallel on a second processor.

Dataflow programs are inherently parallel, and are therefore a seemingly natural fit for languages such as Cilk. The most natural way to encode relational query trees in Cilk is to treat each relational operator as an iterator embedded in a C function call graph. However, doing so yields invalid, deadlock-prone Cilk programs.

The problem is due to a mismatch between Cilk's "strict" semantics, which require function arguments to be executed before the function call itself, and dataflow programming semantics, which require each operator to be executed in parallel. Certain dataflow operators, such as merge join, stream data from multiple inputs without materializing intermediate results. In order to execute such programs, we must be able to guarantee that each operator makes progress independent of the other.

In a serial execution, it is possible that one of the join subqueries will be partially executed, and then block until its output is consumed. Similarly, the join algorithm will block until the other output produces data. Because the execution is serial, the second subquery will never be invoked, and the program will deadlock.

The solution seems to be to explicitly spawn a new pthread for each such operator. The Cilk runtime simultaneously guarantees that all pthread threads make progress, and that reasonable runtime scheduling decisions will be made, leveraging the inherent parallelism of each query operator.

In this work, we will examine the performance tradeoffs between conventional, dataflow-oriented execution environments and our new Cilk-style approach, characterizing queries for which each approach outperforms the other.

## 8 Post-seminar outcomes

A number of collaborations were begun during the seminar that continued after the seminar's end. Ashraf Abounaga and Shivnath Babu collaborated on a tutorial proposal, "Workload Management for Big Data Analytics", that has been accepted for ICDE 2013. Douglas Brown from Teradata began a collaboration with Barzan Mozafari from MIT exploring performance modeling. Douglas Brown, Yanpei Chen, Jens Dittrich, Archana Ganapathi,

Harumi A. Kuno, and Y. C. Tay from Teradata, Cloudera, Saarland University, Splunk Inc, HP Labs, National University of Singapore developed their break-out group's abstract into a vision statement (they are currently considering where to publish it). Robert Chansler from LinkedIn began a collaboration with Shivnath Babu's Starfish project at Duke University.

## Participants

- Ashraf Aboulnaga  
University of Waterloo, CA
- Awny Al-Omari  
Teradata, US
- Shivnath Babu  
Duke University, US
- Michael A. Bender  
State University of New York at  
Stony Brook, US
- Douglas Brown  
Teradata – Rancho Santa Fe, US
- Robert J. Chansler  
Linkedin, US
- Yanpei Chen  
University of California –  
Berkeley, US
- Jens Dittrich  
Universität des Saarlandes, DE
- Archana Ganapathi  
Splunk Inc., USA
- Goetz Graefe  
HP Labs – Madison, USA
- Hakan Haciguemus  
NEC Laboratories America, Inc.  
– Cupertino, US
- Rao Kakarlamudi  
HP – Palo Alto, US
- Harumi A. Kuno  
HP Labs – Palo Alto, US
- Bradley C. Kuszmaul  
MIT – Cambridge, US
- Barzan Mozafari  
MIT – Cambridge, US
- Sivaramakrishnan Narayanan  
EMC Corp. – Bangalore, IN
- Wendy Powley  
Queen’s Univ. – Kingston, CA
- Norbert Ritter  
Universität Hamburg, DE
- Kai-Uwe Sattler  
TU Ilmenau, Germany
- Russell Sears  
Microsoft Research –  
Mountain View, US
- Michael Seibold  
TU München, US
- Y. C. Tay  
National Univ. of Singapore, SG
- Kostas Tzoumas  
TU Berlin, DE
- Jingren Zhou  
Microsoft Research –  
Redmond, US

