Dances with SAP HANA:
Innovations Towards In-Memory Transactional and Analytical Big Data Processing

Juchang Lee (juc.lee@sap.com, SAP Labs Korea)
December 2016

Agenda

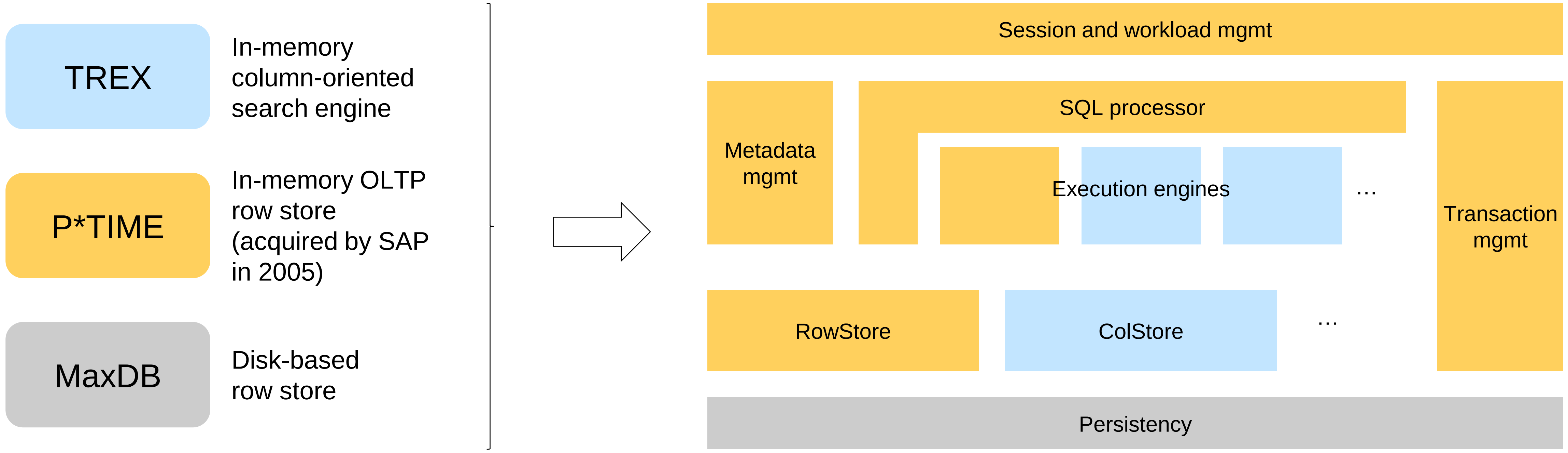
Brief history of SAP HANA

Challenges and Innovations with SAP HANA

- #1. *In-memory* database
- #2. In-memory *Big Data* processing
- #3. In-memory *Transactional and Analytical* Big Data processing
- #4. *HW+SW* co-innovation
- #5. Development and delivery *process* for high-performance DB engine

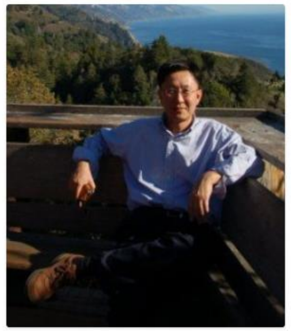
Further opportunities and call for idea-to-market

Brief History of SAP HANA



SAP HANA In-memory Database
for OLTP/OLAP workloads
(first product shipment in late Nov/2010)

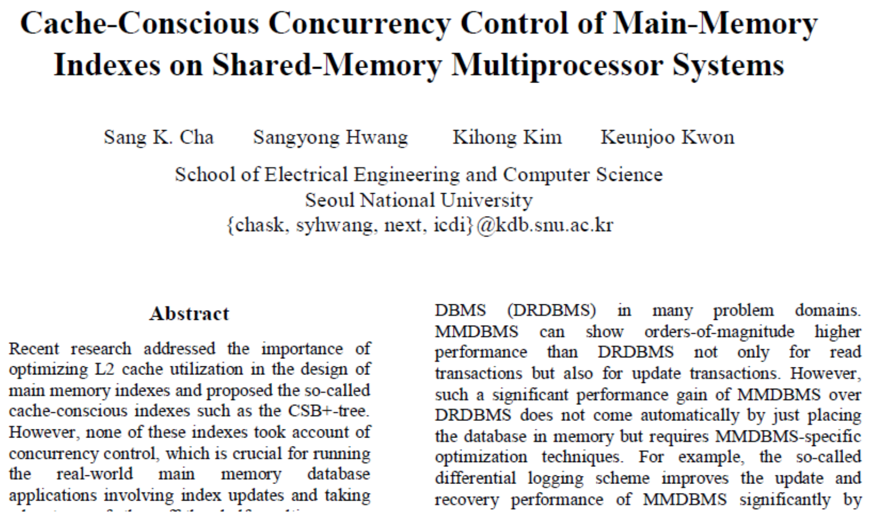
Acknowledgement



…

#1. In-Memory Database Engine

Paradigm shift of the DB engine optimization focus:



Cache-Conscious Concurrency Control of Main-Memory Indexes on Shared-Memory Multiprocessor Systems

Sang K. Cha Sangyong Hwang Kihong Kim Keunjoo Kwon

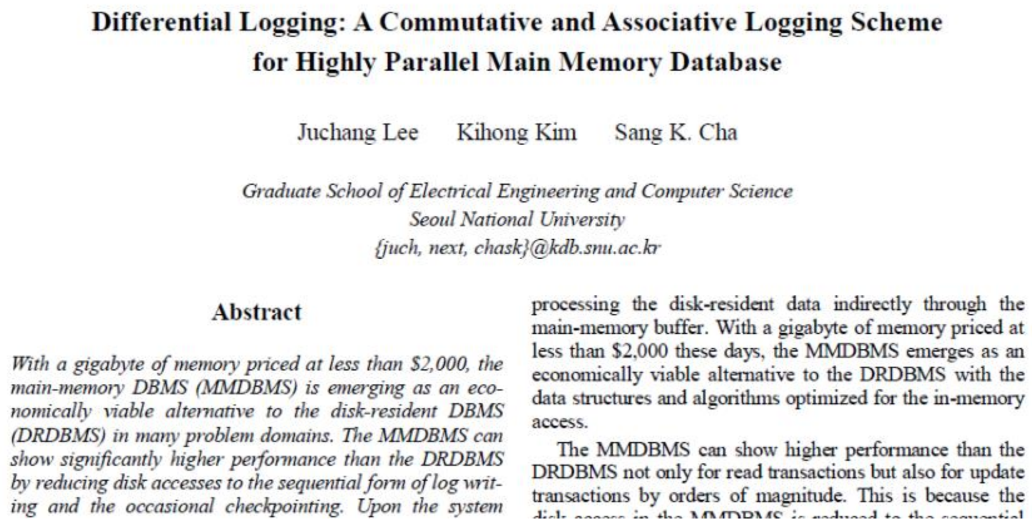
School of Electrical Engineering and Computer Science
Seoul National University
{chask, syhwang, next, icdi}@kdb.snu.ac.kr

Abstract

Recent research addressed the importance of optimizing L2 cache utilization in the design of main memory indexes and proposed the so-called cache-conscious indexes such as the CSB+-tree. However, none of these indexes took account of concurrency control, which is crucial for running the real-world main memory database applications involving index updates and taking

DBMS (DRDBMS) in many problem domains. MMDBMS can show orders-of-magnitude higher performance than DRDBMS not only for read transactions but also for update transactions. However, such a significant performance gain of MMDBMS over DRDBMS does not come automatically by just placing the database in memory but requires MMDBMS-specific optimization techniques. For example, the so-called differential logging scheme improves the update and recovery performance of MMDBMS significantly by

Minimizing L2 cache misses
on concurrent index accesses
VLDB 2001



Differential Logging: A Commutative and Associative Logging Scheme for Highly Parallel Main Memory Database

Juchang Lee Kihong Kim Sang K. Cha

Graduate School of Electrical Engineering and Computer Science
Seoul National University
{juch, next, chask}@kdb.snu.ac.kr

Abstract

With a gigabyte of memory priced at less than $2,000, the main-memory DBMS (MMDBMS) is emerging as an economically viable alternative to the disk-resident DBMS (DRDBMS) in many problem domains. The MMDBMS can show significantly higher performance than the DRDBMS by reducing disk accesses to the sequential form of log writing and the occasional checkpointing. Upon the system

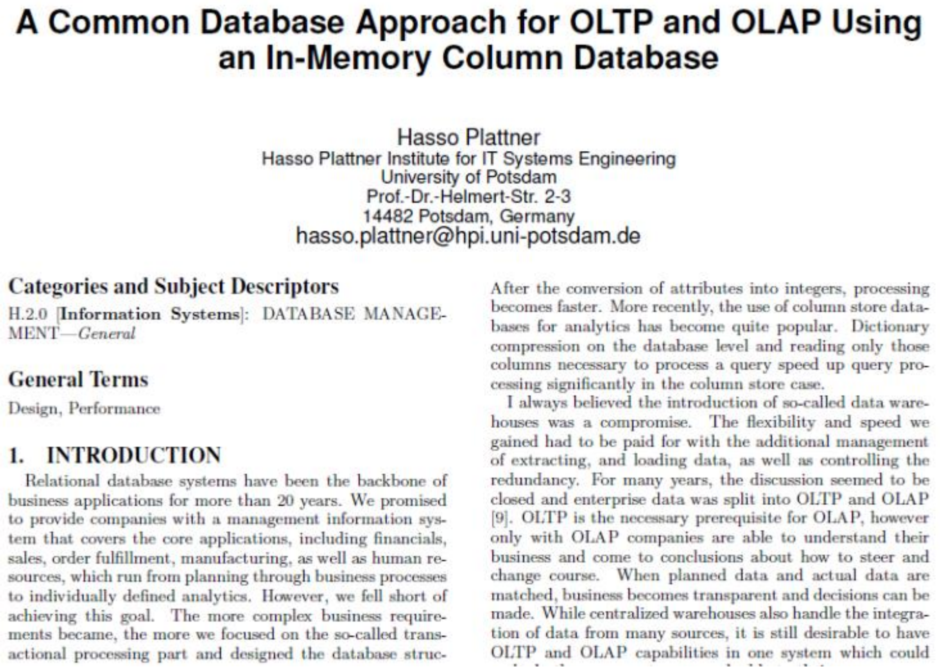
processing the disk-resident data indirectly through the main-memory buffer. With a gigabyte of memory priced at less than $2,000 these days, the MMDBMS emerges as an economically viable alternative to the DRDBMS with the data structures and algorithms optimized for the in-memory access.

The MMDBMS can show higher performance than the DRDBMS not only for read transactions but also for update transactions by orders of magnitude. This is because the

Resolving the bottleneck at log I/O during
transaction commit and database restart
ICDE 2001

#2. Big Data Processing with In-Memory Database Engine

Approach 1: compressed column store and additional acceleration on table scan



A Common Database Approach for OLTP and OLAP Using an In-Memory Column Database

Hasso Plattner
Hasso Plattner Institute for IT Systems Engineering
University of Potsdam
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam, Germany
hasso.plattner@hpi.uni-potsdam.de

Categories and Subject Descriptors
H.2.0 [**Information Systems**]: DATABASE MANAGEMENT—*General*

General Terms
Design, Performance

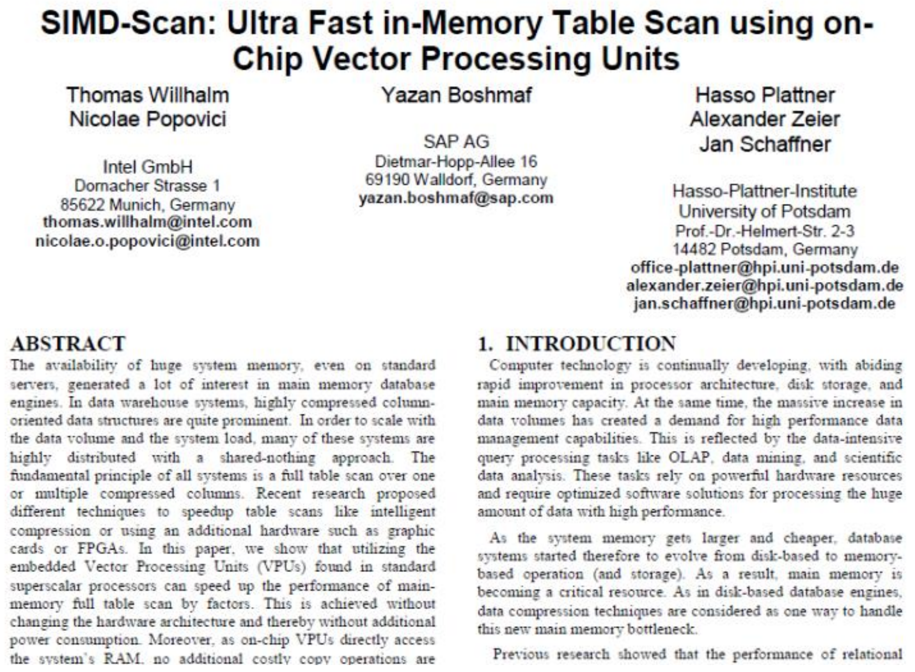
1. INTRODUCTION

Relational database systems have been the backbone of business applications for more than 20 years. We promised to provide companies with a management information system that covers the core applications, including financials, sales, order fulfillment, manufacturing, as well as human resources, which run from planning through business processes to individually defined analytics. However, we fell short of achieving this goal. The more complex business requirements became, the more we focused on the so-called transactional processing part and designed the database struc-

After the conversion of attributes into integers, processing becomes faster. More recently, the use of column store databases for analytics has become quite popular. Dictionary compression on the database level and reading only those columns necessary to process a query speed up query processing significantly in the column store case.

I always believed the introduction of so-called data warehouses was a compromise. The flexibility and speed we gained had to be paid for with the additional management of extracting, and loading data, as well as controlling the redundancy. For many years, the discussion seemed to be closed and enterprise data was split into OLTP and OLAP [9]. OLTP is the necessary prerequisite for OLAP, however only with OLAP companies are able to understand their business and come to conclusions about how to steer and change course. When planned data and actual data are matched, business becomes transparent and decisions can be made. While centralized warehouses also handle the integration of data from many sources, it is still desirable to have OLTP and OLAP capabilities in one system which could

SIGMOD 2009



SIMD-Scan: Ultra Fast in-Memory Table Scan using on-Chip Vector Processing Units

Thomas Willhalm
Nicolae Popovici
Intel GmbH
Dornacher Strasse 1
85622 Munich, Germany
thomas.willhalm@intel.com
nicolae.o.popovici@intel.com

Yazan Boshmaf
SAP AG
Dietmar-Hopp-Allee 16
69190 Walldorf, Germany
yazan.boshmaf@sap.com

Hasso Plattner
Alexander Zeier
Jan Schaffner
Hasso-Plattner-Institute
University of Potsdam
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam, Germany
office-plattner@hpi.uni-potsdam.de
alexander.zeier@hpi.uni-potsdam.de
jan.schaffner@hpi.uni-potsdam.de

ABSTRACT

The availability of huge system memory, even on standard servers, generated a lot of interest in main memory database engines. In data warehouse systems, highly compressed column-oriented data structures are quite prominent. In order to scale with the data volume and the system load, many of these systems are highly distributed with a shared-nothing approach. The fundamental principle of all systems is a full table scan over one or multiple compressed columns. Recent research proposed different techniques to speedup table scans like intelligent compression or using an additional hardware such as graphic cards or FPGAs. In this paper, we show that utilizing the embedded Vector Processing Units (VPUs) found in standard superscalar processors can speed up the performance of main-memory full table scan by factors. This is achieved without changing the hardware architecture and thereby without additional power consumption. Moreover, as on-chip VPUs directly access the system's RAM, no additional costly copy operations are

1. INTRODUCTION

Computer technology is continually developing, with abiding rapid improvement in processor architecture, disk storage, and main memory capacity. At the same time, the massive increase in data volumes has created a demand for high performance data management capabilities. This is reflected by the data-intensive query processing tasks like OLAP, data mining, and scientific data analysis. These tasks rely on powerful hardware resources and require optimized software solutions for processing the huge amount of data with high performance.

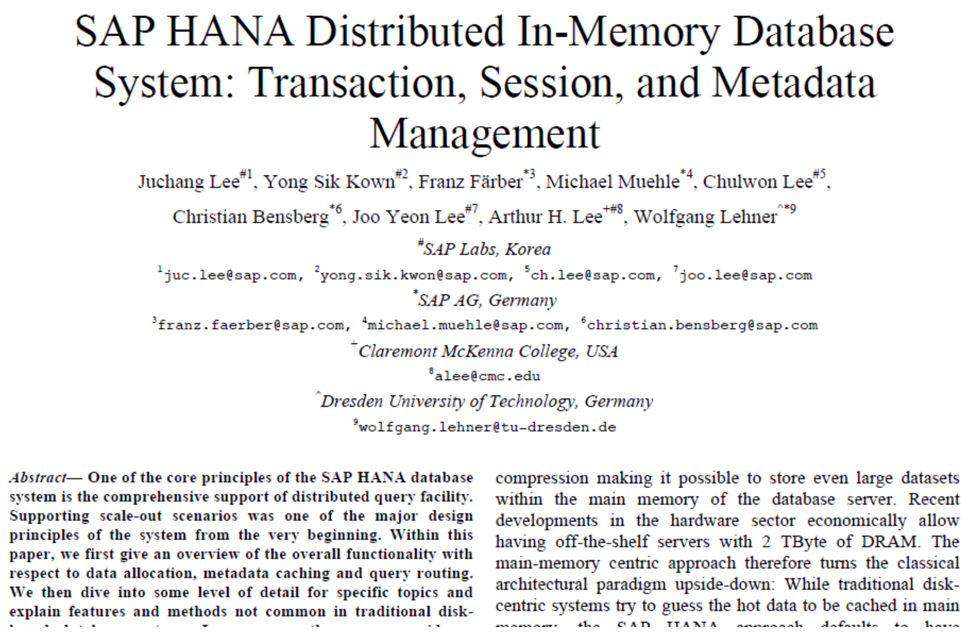
As the system memory gets larger and cheaper, database systems started therefore to evolve from disk-based to memory-based operation (and storage). As a result, main memory is becoming a critical resource. As in disk-based database engines, data compression techniques are considered as one way to handle this new main memory bottleneck.

Previous research showed that the performance of relational

VLDB 2009

#2. Big Data Processing with In-Memory Database Engine

Approach 2: scaling out



SAP HANA Distributed In-Memory Database System: Transaction, Session, and Metadata Management

Juchang Lee[#1], Yong Sik Kown[#2], Franz Färber[*3], Michael Muehle[*4], Chulwon Lee[#5], Christian Bensberg[*6], Joo Yeon Lee[#7], Arthur H. Lee[+#8], Wolfgang Lehner[^*9]

[#]*SAP Labs, Korea*

[1]juc.lee@sap.com, [2]yong.sik.kwon@sap.com, [5]ch.lee@sap.com, [7]joo.lee@sap.com

[*]*SAP AG, Germany*

[3]franz.faerber@sap.com, [4]michael.muehle@sap.com, [6]christian.bensberg@sap.com

[+]*Claremont McKenna College, USA*

[8]alee@cmc.edu

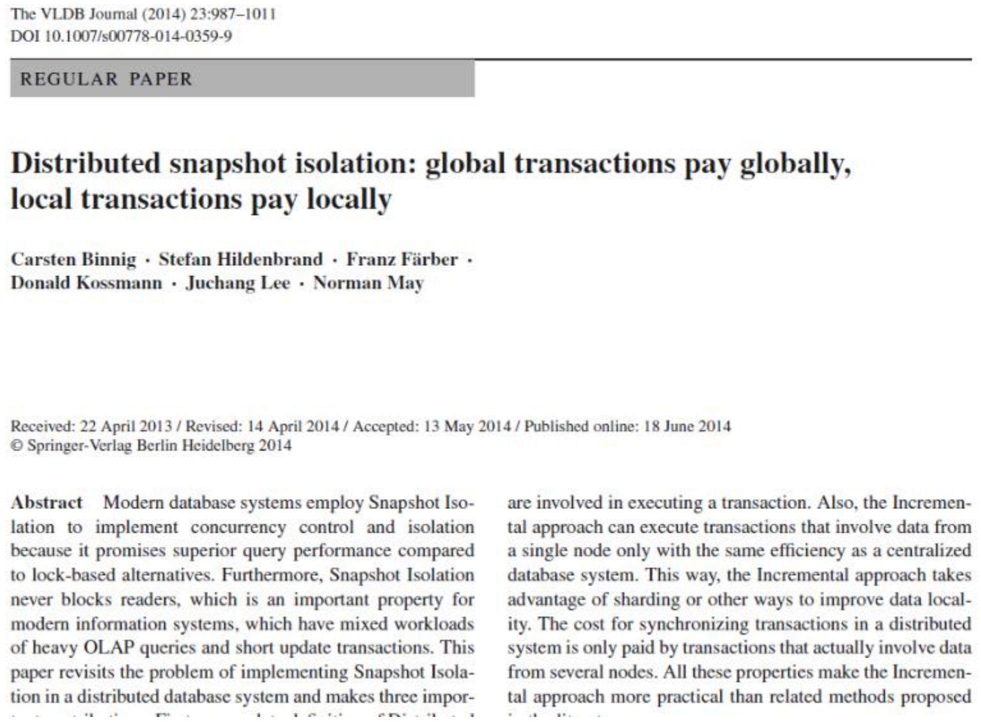
[^]*Dresden University of Technology, Germany*

[9]wolfgang.lehner@tu-dresden.de

Abstract— **One of the core principles of the SAP HANA database system is the comprehensive support of distributed query facility. Supporting scale-out scenarios was one of the major design principles of the system from the very beginning. Within this paper, we first give an overview of the overall functionality with respect to data allocation, metadata caching and query routing. We then dive into some level of detail for specific topics and explain features and methods not common in traditional disk-**

compression making it possible to store even large datasets within the main memory of the database server. Recent developments in the hardware sector economically allow having off-the-shelf servers with 2 TByte of DRAM. The main-memory centric approach therefore turns the classical architectural paradigm upside-down: While traditional disk-centric systems try to guess the hot data to be cached in main

Practical optimizations under shared-nothing distributed database systems
ICDE 2013



The VLDB Journal (2014) 23:987–1011
DOI 10.1007/s00778-014-0359-9

REGULAR PAPER

Distributed snapshot isolation: global transactions pay globally, local transactions pay locally

Carsten Binnig · Stefan Hildenbrand · Franz Färber · Donald Kossmann · Juchang Lee · Norman May

Received: 22 April 2013 / Revised: 14 April 2014 / Accepted: 13 May 2014 / Published online: 18 June 2014
© Springer-Verlag Berlin Heidelberg 2014

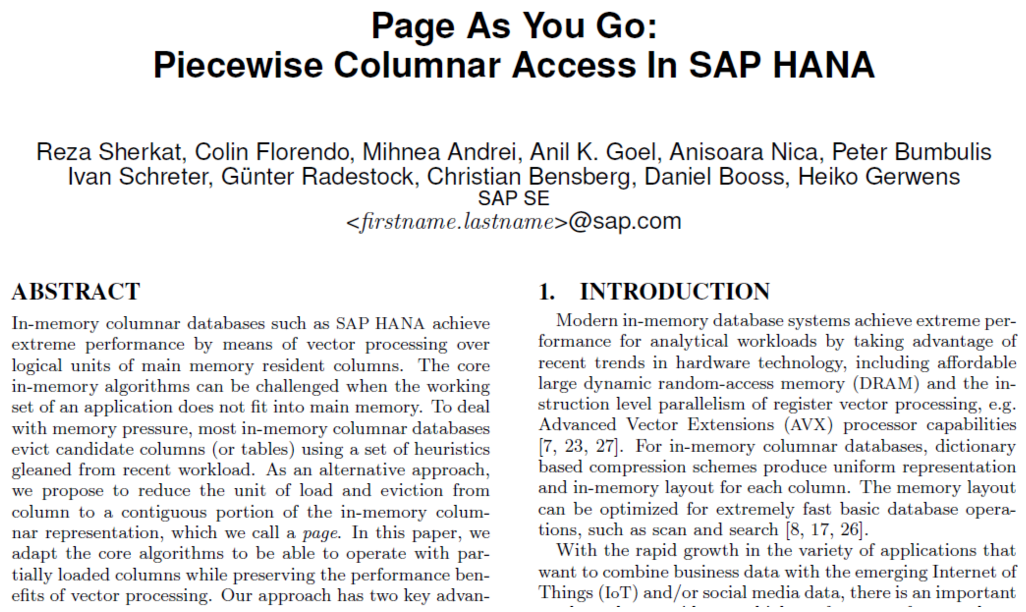
Abstract Modern database systems employ Snapshot Isolation to implement concurrency control and isolation because it promises superior query performance compared to lock-based alternatives. Furthermore, Snapshot Isolation never blocks readers, which is an important property for modern information systems, which have mixed workloads of heavy OLAP queries and short update transactions. This paper revisits the problem of implementing Snapshot Isolation in a distributed database system and makes three impor-

are involved in executing a transaction. Also, the Incremental approach can execute transactions that involve data from a single node only with the same efficiency as a centralized database system. This way, the Incremental approach takes advantage of sharding or other ways to improve data locality. The cost for synchronizing transactions in a distributed system is only paid by transactions that actually involve data from several nodes. All these properties make the Incremental approach more practical than related methods proposed

Optimizations for delayed snapshot isolation
VLDB Journal 2014

#2. Big Data Processing with In-Memory Database Engine

Approach 3: offloading cold data to disks



Page As You Go:
Piecewise Columnar Access In SAP HANA

Reza Sherkat, Colin Florendo, Mihnea Andrei, Anil K. Goel, Anisoara Nica, Peter Bumbulis
Ivan Schreter, Günter Radestock, Christian Bensberg, Daniel Booss, Heiko Gerwens
SAP SE
<*firstname.lastname*>@sap.com

ABSTRACT

In-memory columnar databases such as SAP HANA achieve extreme performance by means of vector processing over logical units of main memory resident columns. The core in-memory algorithms can be challenged when the working set of an application does not fit into main memory. To deal with memory pressure, most in-memory columnar databases evict candidate columns (or tables) using a set of heuristics gleaned from recent workload. As an alternative approach, we propose to reduce the unit of load and eviction from column to a contiguous portion of the in-memory columnar representation, which we call a *page*. In this paper, we adapt the core algorithms to be able to operate with partially loaded columns while preserving the performance benefits of vector processing. Our approach has two key advan-

1. INTRODUCTION

Modern in-memory database systems achieve extreme performance for analytical workloads by taking advantage of recent trends in hardware technology, including affordable large dynamic random-access memory (DRAM) and the instruction level parallelism of register vector processing, e.g. Advanced Vector Extensions (AVX) processor capabilities [7, 23, 27]. For in-memory columnar databases, dictionary based compression schemes produce uniform representation and in-memory layout for each column. The memory layout can be optimized for extremely fast basic database operations, such as scan and search [8, 17, 26].

With the rapid growth in the variety of applications that want to combine business data with the emerging Internet of Things (IoT) and/or social media data, there is an important

SIGMOD 2016

#3. OLTP and OLAP together in a single Database Engine

Hybrid Garbage Collection for Multi-Version Concurrency Control in SAP HANA

Juchang Lee [#]
juc.lee@sap.com

Hyungyu Shin [‡]
hgshin@dblab.postech.ac.kr

Chang Gyoo Park [#]
chang.gyoo.park@sap.com

Seongyun Ko [‡]
syko@dblab.postech.ac.kr

Jaeyun Noh [#]
jaeyun.noh@sap.com

Yongjae Chuh [#]
yongjae.chuh@sap.com

Wolfgang Stephan [†]
wolfgang.stephan@sap.com

Wook-Shin Han [‡*]
wshan@dblab.postech.ac.kr

[#]SAP Labs Korea, Korea
[‡]Pohang University of Science and Technology (POSTECH), Korea
[†]SAP SE, Germany

ABSTRACT

While multi-version concurrency control (MVCC) supports fast and robust performance in in-memory, relational databases, it has the potential problem of a growing number of versions over time due to obsolete versions. Although a few TB of main memory is available for enterprise machines, the memory resource should be used carefully for economic and practical reasons. Thus, in order to maintain the necessary number of versions in MVCC, versions which will no longer be used need to be deleted. This process is called garbage collection. MVCC uses the concept of *visibility* to define garbage. A set of versions for each record is first identified as candidate if their version timestamps are lower than the minimum value of snapshot timestamps of active snapshots

group garbage collection, table garbage collection, and interval garbage collection. Through experiments using mixed OLTP and OLAP workloads, we show that HybridGC effectively and efficiently collects garbage versions with negligible overhead.

Keywords

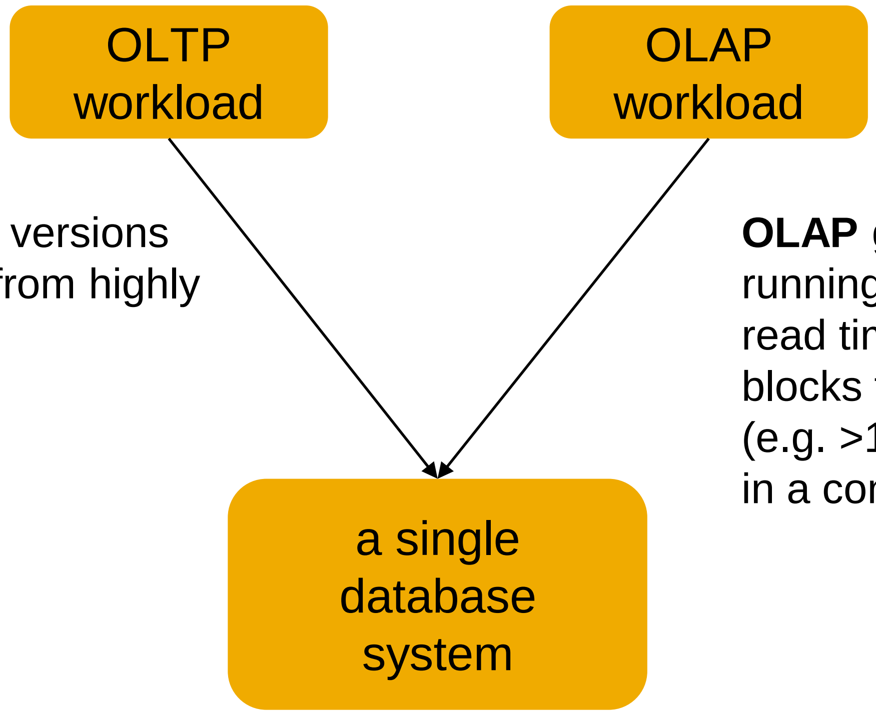
Garbage collection; multi-version concurrency control; SAP HANA

1. INTRODUCTION

Commercial database management systems (DBMSs) including SAP HANA employ multi-version concurrency control (MVCC) due to robust and better performance for var-

SIGMOD 2016

Problem: Garbage Collection under OLTP/OLAP-mixed Workloads



OLTP generates versions with high speed from highly concurrent write transactions

OLAP generates long-running queries holding old read timestamps which blocks the garbage collector (e.g. >1hr of cursor life time in a commercial system)

Consequence

- Version space grows – more critical especially in in-memory DBMS
- Version lookup cost increases, i.e. performance drops

Solution: Interval Garbage Collection and Hybrid Garbage Collection

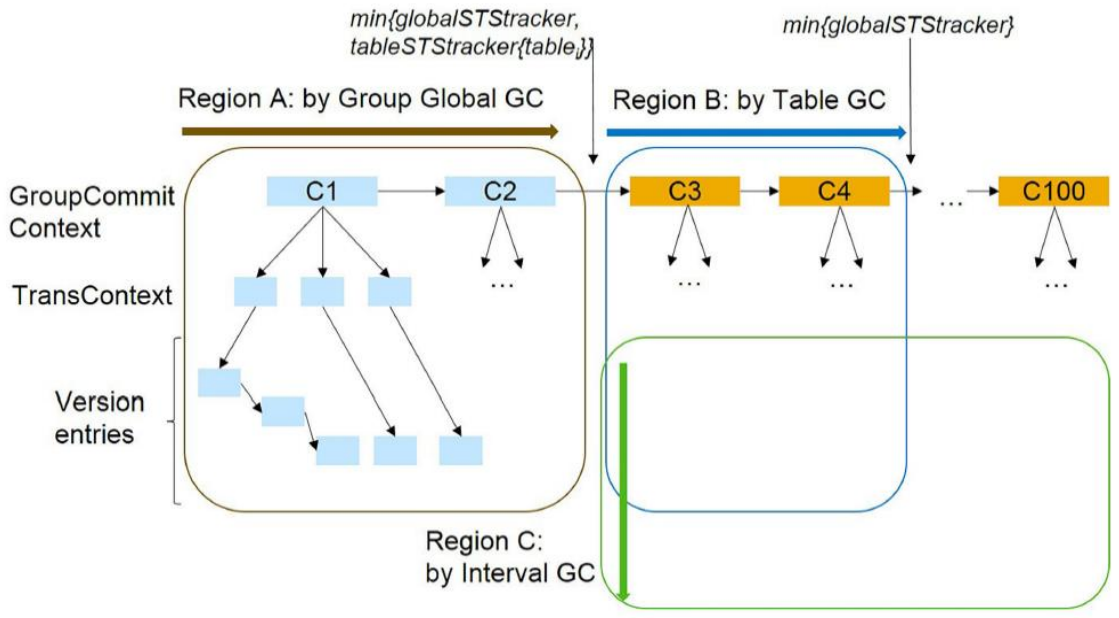


Figure 9: Hybrid garbage collection.

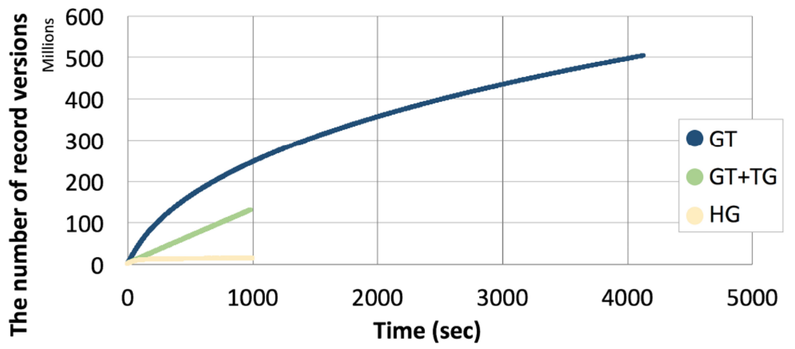


Figure 10: The number of record versions with a long-duration cursor.

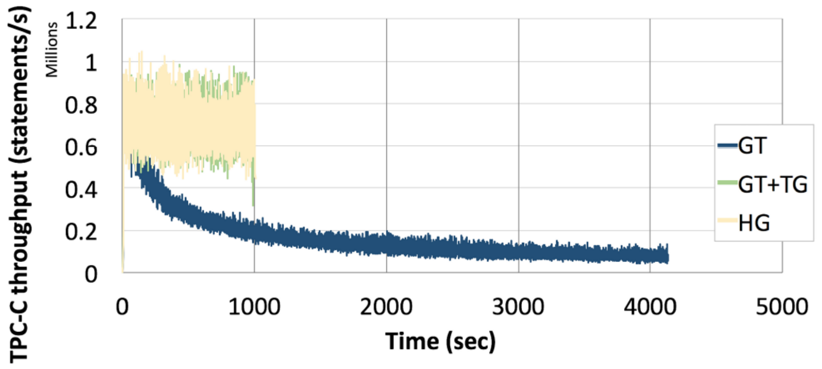
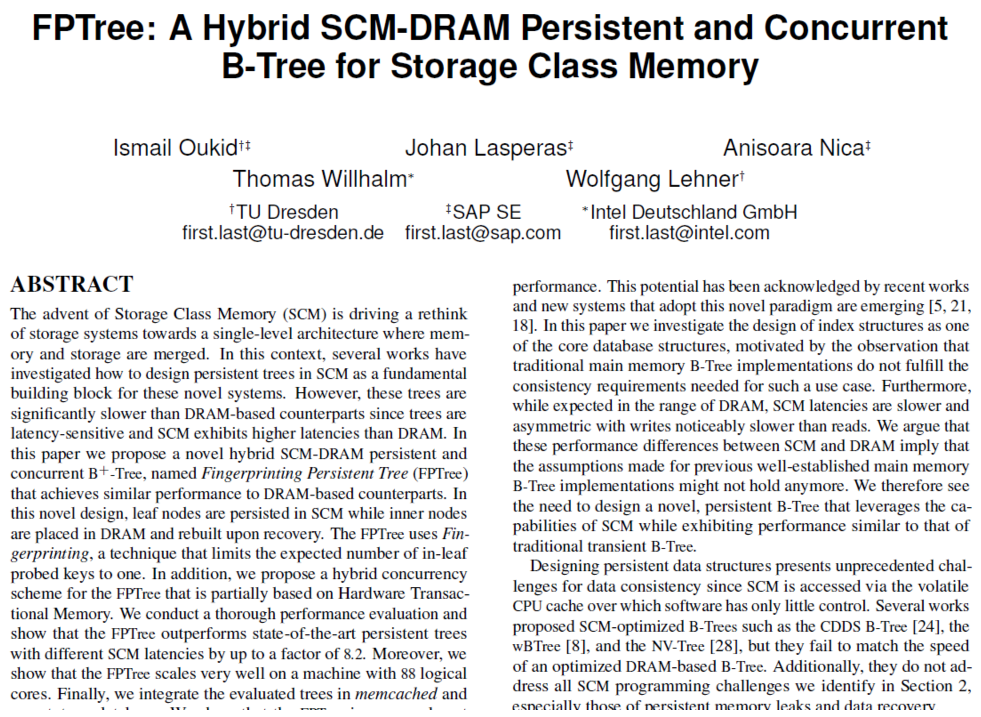


Figure 12: TPC-C throughput with a long-duration cursor.

#4. HW+SW Co-Innovation



FPTree: A Hybrid SCM-DRAM Persistent and Concurrent B-Tree for Storage Class Memory

Ismail Oukid[†‡] Johan Lasperas[‡] Anisoara Nica[‡]
Thomas Willhalm[*] Wolfgang Lehner[†]

[†]TU Dresden first.last@tu-dresden.de
[‡]SAP SE first.last@sap.com
[*]Intel Deutschland GmbH first.last@intel.com

ABSTRACT

The advent of Storage Class Memory (SCM) is driving a rethink of storage systems towards a single-level architecture where memory and storage are merged. In this context, several works have investigated how to design persistent trees in SCM as a fundamental building block for these novel systems. However, these trees are significantly slower than DRAM-based counterparts since trees are latency-sensitive and SCM exhibits higher latencies than DRAM. In this paper we propose a novel hybrid SCM-DRAM persistent and concurrent B$^+$-Tree, named *Fingerprinting Persistent Tree* (FPTree) that achieves similar performance to DRAM-based counterparts. In this novel design, leaf nodes are persisted in SCM while inner nodes are placed in DRAM and rebuilt upon recovery. The FPTree uses *Fingerprinting*, a technique that limits the expected number of in-leaf probed keys to one. In addition, we propose a hybrid concurrency scheme for the FPTree that is partially based on Hardware Transactional Memory. We conduct a thorough performance evaluation and show that the FPTree outperforms state-of-the-art persistent trees with different SCM latencies by up to a factor of 8.2. Moreover, we show that the FPTree scales very well on a machine with 88 logical cores. Finally, we integrate the evaluated trees in *memcached* and

performance. This potential has been acknowledged by recent works and new systems that adopt this novel paradigm are emerging [5, 21, 18]. In this paper we investigate the design of index structures as one of the core database structures, motivated by the observation that traditional main memory B-Tree implementations do not fulfill the consistency requirements needed for such a use case. Furthermore, while expected in the range of DRAM, SCM latencies are slower and asymmetric with writes noticeably slower than reads. We argue that these performance differences between SCM and DRAM imply that the assumptions made for previous well-established main memory B-Tree implementations might not hold anymore. We therefore see the need to design a novel, persistent B-Tree that leverages the capabilities of SCM while exhibiting performance similar to that of traditional transient B-Tree.

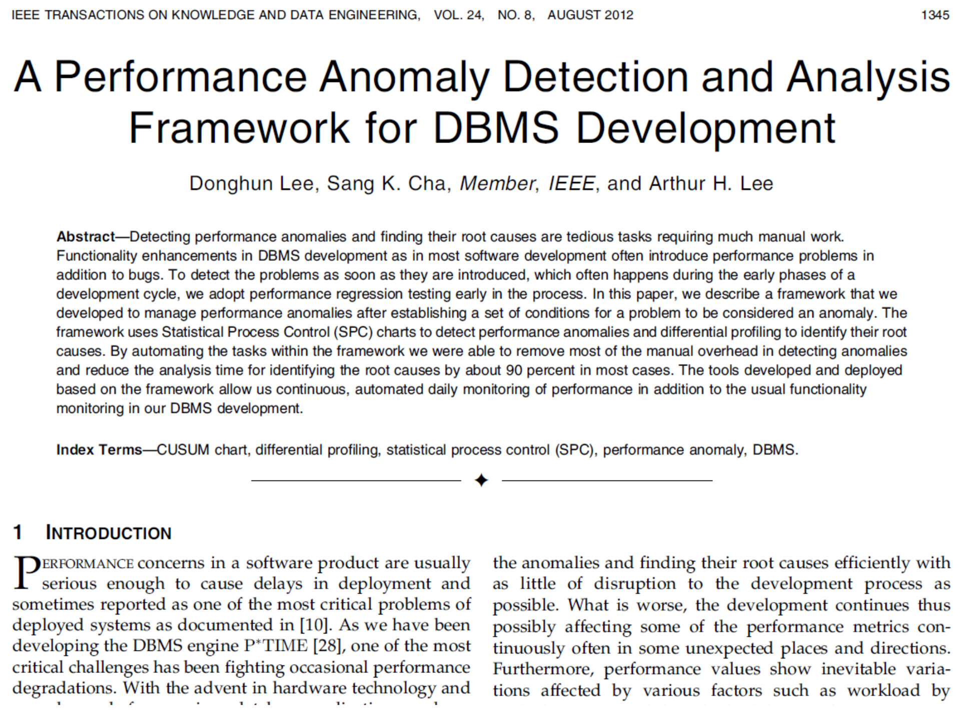
Designing persistent data structures presents unprecedented challenges for data consistency since SCM is accessed via the volatile CPU cache over which software has only little control. Several works proposed SCM-optimized B-Trees such as the CDDS B-Tree [24], the wBTree [8], and the NV-Tree [28], but they fail to match the speed of an optimized DRAM-based B-Tree. Additionally, they do not address all SCM programming challenges we identify in Section 2, especially those of persistent memory leaks and data recovery

SIGMOD 2016

SAMSUNG
SAMSUNG - SAP Research Center Grand Opening Celebration
29 September 2016
SAMSUNG-SAP Research Center

Samsung-SAP Research Center
(since 29/Sep/2016)

#5. Development and Delivery Process for High-Performance DB Engine



IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 24, NO. 8, AUGUST 2012 1345

A Performance Anomaly Detection and Analysis Framework for DBMS Development

Donghun Lee, Sang K. Cha, *Member*, *IEEE*, and Arthur H. Lee

Abstract—Detecting performance anomalies and finding their root causes are tedious tasks requiring much manual work. Functionality enhancements in DBMS development as in most software development often introduce performance problems in addition to bugs. To detect the problems as soon as they are introduced, which often happens during the early phases of a development cycle, we adopt performance regression testing early in the process. In this paper, we describe a framework that we developed to manage performance anomalies after establishing a set of conditions for a problem to be considered an anomaly. The framework uses Statistical Process Control (SPC) charts to detect performance anomalies and differential profiling to identify their root causes. By automating the tasks within the framework we were able to remove most of the manual overhead in detecting anomalies and reduce the analysis time for identifying the root causes by about 90 percent in most cases. The tools developed and deployed based on the framework allow us continuous, automated daily monitoring of performance in addition to the usual functionality monitoring in our DBMS development.

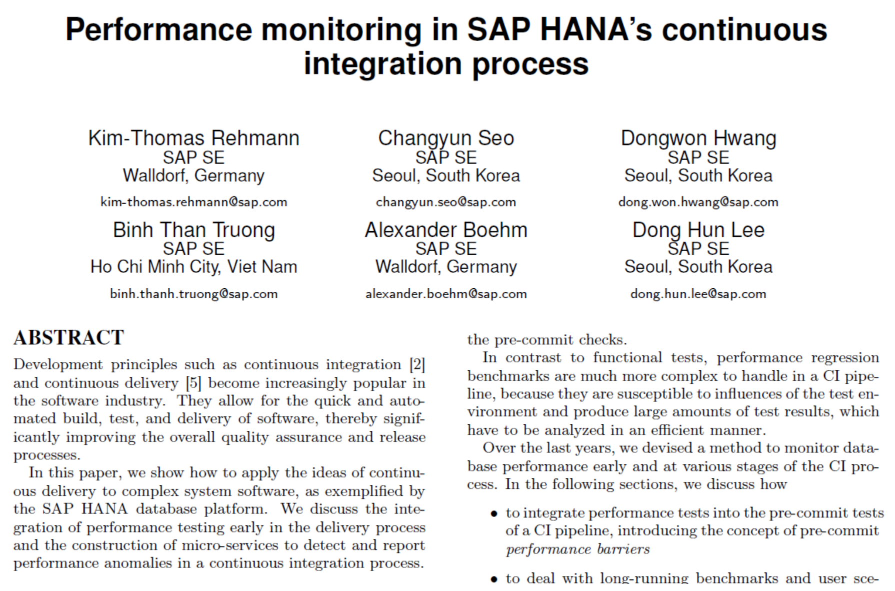
Index Terms—CUSUM chart, differential profiling, statistical process control (SPC), performance anomaly, DBMS.

1 INTRODUCTION

PERFORMANCE concerns in a software product are usually serious enough to cause delays in deployment and sometimes reported as one of the most critical problems of deployed systems as documented in [10]. As we have been developing the DBMS engine P*TIME [28], one of the most critical challenges has been fighting occasional performance degradations. With the advent in hardware technology and

the anomalies and finding their root causes efficiently with as little of disruption to the development process as possible. What is worse, the development continues thus possibly affecting some of the performance metrics continuously often in some unexpected places and directions. Furthermore, performance values show inevitable variations affected by various factors such as workload by

IEEE TKDE 2012



Performance monitoring in SAP HANA's continuous integration process

Kim-Thomas Rehmann
SAP SE
Walldorf, Germany
kim-thomas.rehmann@sap.com

Changyun Seo
SAP SE
Seoul, South Korea
changyun.seo@sap.com

Dongwon Hwang
SAP SE
Seoul, South Korea
dong.won.hwang@sap.com

Binh Than Truong
SAP SE
Ho Chi Minh City, Viet Nam
binh.thanh.truong@sap.com

Alexander Boehm
SAP SE
Walldorf, Germany
alexander.boehm@sap.com

Dong Hun Lee
SAP SE
Seoul, South Korea
dong.hun.lee@sap.com

ABSTRACT

Development principles such as continuous integration [2] and continuous delivery [5] become increasingly popular in the software industry. They allow for the quick and automated build, test, and delivery of software, thereby significantly improving the overall quality assurance and release processes.

In this paper, we show how to apply the ideas of continuous delivery to complex system software, as exemplified by the SAP HANA database platform. We discuss the integration of performance testing early in the delivery process and the construction of micro-services to detect and report performance anomalies in a continuous integration process.

the pre-commit checks.

In contrast to functional tests, performance regression benchmarks are much more complex to handle in a CI pipeline, because they are susceptible to influences of the test environment and produce large amounts of test results, which have to be analyzed in an efficient manner.

Over the last years, we devised a method to monitor database performance early and at various stages of the CI process. In the following sections, we discuss how

- to integrate performance tests into the pre-commit tests of a CI pipeline, introducing the concept of pre-commit *performance barriers*
- to deal with long-running benchmarks and user sce-

SIGMETRICS 2016

Further Opportunities and Call for Idea-to-Market

Industry-University co-research

Open job positions

- DB kernel developers (transaction, replication, session, metadata, in-memory store):
  - https://jobs.sap.com/job/Seoul-C%2B%2B-%28Senior%29-Developer-for-SAP-HANA-Database-Kernel-Team-Job-11/321176501/
- SQL engine developers (SQL/SQLScript executor, optimizer):
  - https://jobs.sap.com/job/Bangalore-C%2B%2B-%28Senior%29-Developer-for-SAP-HANA-SQL-engine-team-Job-11/321330101/
- Quality specialist, development infrastructure, development support
  - https://jobs.sap.com/job/Seoul-Quality-Specialist-Job-11/320983201/
  - https://jobs.sap.com/job/Seoul-SAP-HANA-Platform-Dev-Support-Job-11/321336901/