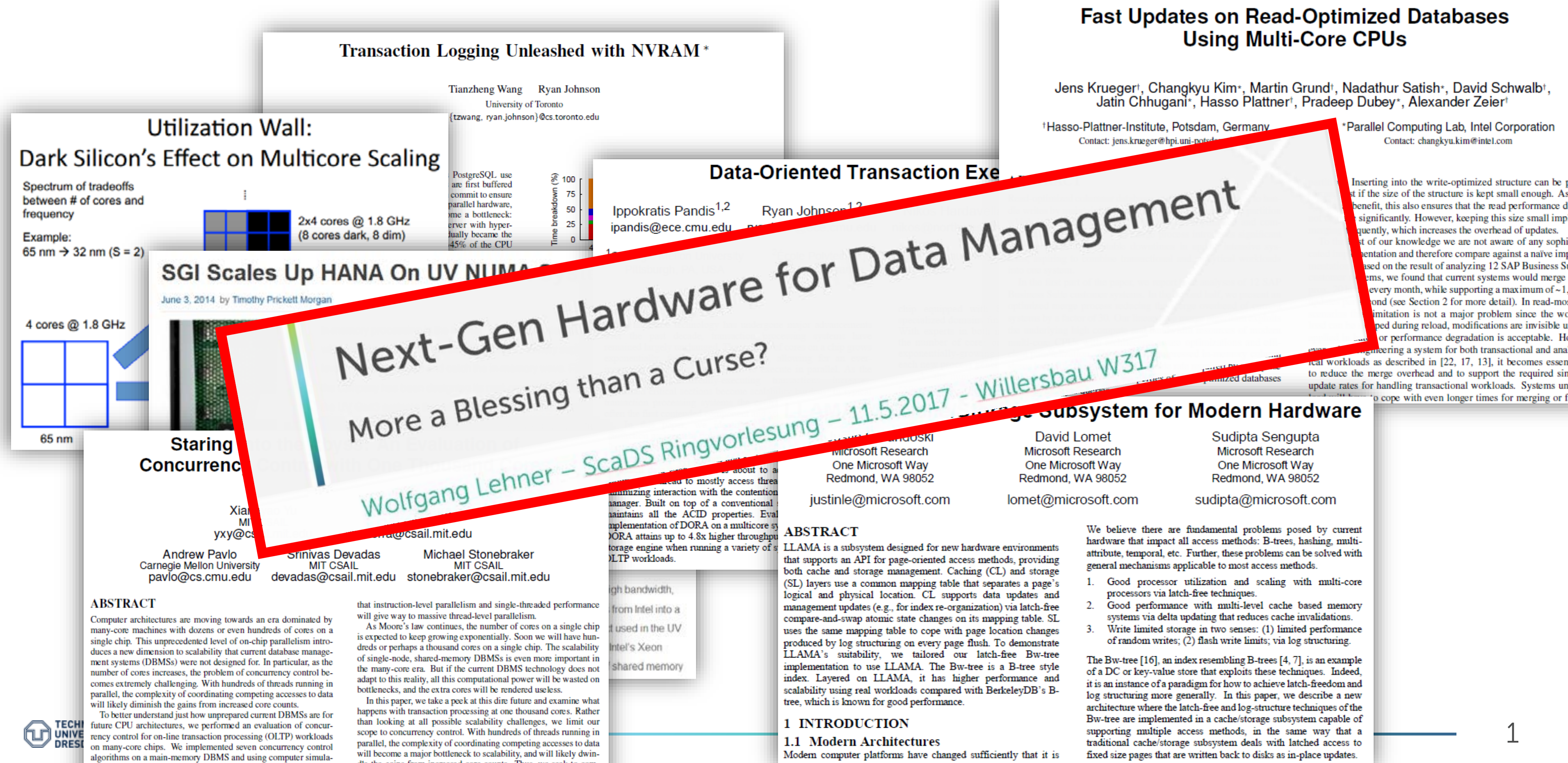
Modern Hardware – All over the place…

Next-Gen Hardware for Data Management

More a Blessing than a Curse?

Wolfgang Lehner – ScaDS Ringvorlesung – 11.5.2017 - Willersbau W317



Transaction Logging Unleashed with NVRAM *

Tianzheng Wang Ryan Johnson
University of Toronto
{tzwang, ryan.johnson}@cs.toronto.edu

PostgreSQL use
are first buffered
commit to ensure
parallel hardware,
ome a bottleneck:
erver with hyper-
dually became the
45% of the CPU

Utilization Wall:
Dark Silicon's Effect on Multicore Scaling

Spectrum of tradeoffs
between # of cores and
frequency

Example:
65 nm → 32 nm (S = 2)

Data-Oriented Transaction Exe

Ippokratis Pandis[1,2]
ipandis@ece.cmu.edu

Ryan Johnson[1,2]

about to a
to mostly access threa
inimizing interaction with the contention
nanager. Built on top of a conventional
naintains all the ACID properties. Eval
nplementation of DORA on a multicore sy
DORA attains up to 4.8x higher throughpu
torage engine when running a variety of s
OLTP workloads.

SGI Scales Up HANA On UV NUMA

June 3, 2014 by Timothy Prickett Morgan

igh bandwidth,
from Intel into a
t used in the UV
Intel's Xeon
shared memory

Staring
Concurrenc

Xia
MI
yxy@c

a@csail.mit.edu

Andrew Pavlo
Carnegie Mellon University
pavlo@cs.cmu.edu

Srinivas Devadas
MIT CSAIL
devadas@csail.mit.edu

Michael Stonebraker
MIT CSAIL
stonebraker@csail.mit.edu

ABSTRACT

Computer architectures are moving towards an era dominated by many-core machines with dozens or even hundreds of cores on a single chip. This unprecedented level of on-chip parallelism introduces a new dimension to scalability that current database management systems (DBMSs) were not designed for. In particular, as the number of cores increases, the problem of concurrency control becomes extremely challenging. With hundreds of threads running in parallel, the complexity of coordinating competing accesses to data will likely diminish the gains from increased core counts.

To better understand just how unprepared current DBMSs are for future CPU architectures, we performed an evaluation of concurrency control for on-line transaction processing (OLTP) workloads on many-core chips. We implemented seven concurrency control algorithms on a main-memory DBMS and using computer simula-

that instruction-level parallelism and single-threaded performance will give way to massive thread-level parallelism.

As Moore's law continues, the number of cores on a single chip is expected to keep growing exponentially. Soon we will have hundreds or perhaps a thousand cores on a single chip. The scalability of single-node, shared-memory DBMSs is even more important in the many-core era. But if the current DBMS technology does not adapt to this reality, all this computational power will be wasted on bottlenecks, and the extra cores will be rendered useless.

In this paper, we take a peek at this dire future and examine what happens with transaction processing at one thousand cores. Rather than looking at all possible scalability challenges, we limit our scope to concurrency control. With hundreds of threads running in parallel, the complexity of coordinating competing accesses to data will become a major bottleneck to scalability, and will likely dwin-

Fast Updates on Read-Optimized Databases
Using Multi-Core CPUs

Jens Krueger[†], Changkyu Kim[*], Martin Grund[†], Nadathur Satish[*], David Schwalb[†],
Jatin Chhugani[*], Hasso Plattner[†], Pradeep Dubey[*], Alexander Zeier[†]

[†]Hasso-Plattner-Institute, Potsdam, Germany
Contact: jens.krueger@hpi.uni-potsd

[*]Parallel Computing Lab, Intel Corporation
Contact: changkyu.kim@intel.com

Inserting into the write-optimized structure can be
t if the size of the structure is kept small enough. As
benefit, this also ensures that the read performance d
significantly. However, keeping this size small imp
quently, which increases the overhead of updates.
t of our knowledge we are not aware of any sophi
nentation and therefore compare against a naïve imp
ased on the result of analyzing 12 SAP Business S
ems, we found that current systems would merge
every month, while supporting a maximum of ~1,
ond (see Section 2 for more detail). In read-mos
imitation is not a major problem since the wo
ped during reload, modifications are invisible u
or performance degradation is acceptable. H
gineering a system for both transactional and ana
ical workloads as described in [22, 17, 13], it becomes essen
to reduce the merge overhead and to support the required si
update rates for handling transactional workloads. Systems un
to cope with even longer times for merging or f

mized databases

ge Subsystem for Modern Hardware

Microsoft Research
One Microsoft Way
Redmond, WA 98052
justinle@microsoft.com

David Lomet
Microsoft Research
One Microsoft Way
Redmond, WA 98052
lomet@microsoft.com

Sudipta Sengupta
Microsoft Research
One Microsoft Way
Redmond, WA 98052
sudipta@microsoft.com

ABSTRACT

LLAMA is a subsystem designed for new hardware environments that supports an API for page-oriented access methods, providing both cache and storage management. Caching (CL) and storage (SL) layers use a common mapping table that separates a page's logical and physical location. CL supports data updates and management updates (e.g., for index re-organization) via latch-free compare-and-swap atomic state changes on its mapping table. SL uses the same mapping table to cope with page location changes produced by log structuring on every page flush. To demonstrate LLAMA's suitability, we tailored our latch-free Bw-tree implementation to use LLAMA. The Bw-tree is a B-tree style index. Layered on LLAMA, it has higher performance and scalability using real workloads compared with BerkeleyDB's B-tree, which is known for good performance.

1 INTRODUCTION

1.1 Modern Architectures

Modern computer platforms have changed sufficiently that it is

We believe there are fundamental problems posed by current hardware that impact all access methods: B-trees, hashing, multi-attribute, temporal, etc. Further, these problems can be solved with general mechanisms applicable to most access methods.

1. Good processor utilization and scaling with multi-core processors via latch-free techniques.
2. Good performance with multi-level cache based memory systems via delta updating that reduces cache invalidations.
3. Write limited storage in two senses: (1) limited performance of random writes; (2) flash write limits; via log structuring.

The Bw-tree [16], an index resembling B-trees [4, 7], is an example of a DC or key-value store that exploits these techniques. Indeed, it is an instance of a paradigm for how to achieve latch-freedom and log structuring more generally. In this paper, we describe a new architecture where the latch-free and log-structure techniques of the Bw-tree are implemented in a cache/storage subsystem capable of supporting multiple access methods, in the same way that a traditional cache/storage subsystem deals with latched access to fixed size pages that are written back to disks as in-place updates.