How to Stop Under-Utilization and Love Multicores

Anastasia Ailamaki
EPFL
anastasia.ailamaki@epfl.ch

Erietta Liarou
EPFL
erietta.liarou@epfl.ch

Pınar Tözün
EPFL
pinar.tozun@epfl.ch

Danica Porobic
EPFL
danica.porobic@epfl.ch

Iraklis Psaroudakis
EPFL, SAP AG
iraklis.psaroudakis@epfl.ch

ABSTRACT

Designing scalable database management systems on modern hardware has been a challenge for almost a decade. Hardware trends oblige software to overcome three major challenges against systems scalability: (1) Exploiting the abundant thread-level parallelism provided by multicores, (2) Achieving predictively efficient execution despite the variability in communication latencies among cores on multisocket multicores, and (3) Taking advantage of the aggressive micro-architectural features.

In this tutorial, we shed light on the above three challenges and survey recent proposals to alleviate them. First, we present a systematic way of eliminating scalability bottlenecks based on minimizing unbounded communication and show several techniques that minimize bottlenecks in major components of database management systems. In addition, we demonstrate methods to parallelize major database operations. Then, we analyze the problems that arise from the non-uniform nature of communication latencies on modern multisockets and ways to address them for systems that already scale well on multicores. Finally, we examine the sources of under-utilization within a modern processor and present insights and techniques to better exploit the micro-architectural resources of a processor by improving cache locality at the right level of the memory hierarchy.

1. INTRODUCTION

Length: 3 hours

Target Audience: Researchers and developers in the field of data management systems who are non-experts on modern hardware and the challenges the emerging hardware poses on high-performance transaction and query processing, and PhD students who are interested in learning more about the underlying hardware and seeking a challenging and high-impact research topic on data management systems.

Related Previous Tutorials: The first part of this tutorial, scaling-up on multicores, is presented as part of the VLDB 2013 tutorial titled *Toward Scalable Transaction Processing - Evolution of Shore-MT* [1]. This tutorial, however, has broader scope and includes a range of data management systems and hardware platforms. More specifically, it surveys the concept of scalability for data management systems not just on multicores with uniform access latencies but also on multisockets with non-uniform memory accesses (NUMA) and at the micro-architectural level. In addition, it includes examples from a broader range of storage managers, not just from Shore-MT.

SIGMOD'14, June 22–27, 2014, Snowbird, UT, USA.

http://dx.doi.org/10.1145/2588555.2588892.

2. THREAD-LEVEL PARALLELISM

In step with Moore's Law, hardware gives us more and more opportunities for parallelism rather than faster processors since 2005. Exploiting parallelism is crucial for utilizing the available architectural resources and enabling faster software. However, designing scalable systems that can take advantage of the underlying parallelism remains a challenge. In traditional high performance transaction processing, the inherent communication leads to scalability bottlenecks on today's multicore and multisocket hardware. Even systems that scale very well on one generation of multicores might fail to scale-up on the next generation. On the other hand, in traditional online analytical processing, the database operators that were designed for unicore processors fail to exploit the abundant parallelism offered by modern hardware.

In this first part of the tutorial, we initially teach a methodology for scaling-up transaction processing systems on multicore hardware. More specifically, we identify three types of communication in a typical transaction processing system: *unbounded, fixed, and cooperative* [17]. We demonstrate that the key to achieve scalability on modern hardware, especially for transaction processing systems but also for any system that has similar communication patterns, depends on avoiding the unbounded communication points or downgrading them into fixed or cooperative ones. We show how effective our methodology is in practice by surveying related proposals from recent work (e.g., [10, 18, 21, 27, 28, 30]).

Traditional online analytical processing, however, is formed of read-only queries. Therefore, it does not suffer from the unbounded communication as in transaction processing. On the other hand, the database operators such as joins, scans, etc. are mainly optimized for single threaded execution. Therefore, they fail to exploit intra-query parallelism and cannot utilize several cores naïvely. In this tutorial, we also survey the recent techniques that aim at parallelising traditional database operations and exploring work and data

sharing opportunities among the concurrent queries (e.g. [6, 13, 15, 24]).

3. NON-UNIFORM MEMORY ACCESSES

Data management applications traditionally run on the highest performing servers of the day. Up until recently, such servers had uniform core-to-core communication latencies - multisocket uniprocessors communicate slowly with each other and cores on a multicore communicate fast. Now with multisocket multicores, for the first time we have *Islands*, i.e., groups of cores that communicate fast among themselves and slower with other groups. Currently, an Island is represented by a processor socket but soon, with dozens of cores on the same socket, we expect that Islands will form within a chip. In this setting, memory access times vary greatly depending on several factors including latency to access remote memory and contention for the memory hierarchy such as the shared last level caches, the memory controllers, and the interconnect bandwidth.

In the context of transaction processing, it can be appealing to regard multisocket as a distributed system and deploy multiple nodes in a shared-nothing configuration [18, 27]. While this approach works great for perfectly partitionable workloads, it is very sensitive to distributed transactions and the workload skew. At the same time, hardware-oblivious shared-everything systems suffer from non-uniform latencies that amplify bottlenecks in the critical path [23]. First, we present a set of best practices for choosing a good configuration based on different properties of workload and hardware topology. Then, we present a system that achieves scalability on multisockets by utilizing hardware topology-aware data structures and dynamically adapting to workload and hardware [22].

On the other hand, analytical workloads consist of ad-hoc, long running, and scan-heavy queries over relatively static data. In order to optimize performance, the execution engine needs to become NUMA-aware by tackling two main challenges: (a) employing a scheduling strategy for assigning multiple concurrent threads to cores in order to minimize remote memory accesses while avoiding contention on the memory hierarchy, and (b) dynamically deciding on the data placement in order to minimize the total memory access time of the workload. The two problems are not orthogonal, as data placement can affect scheduling decisions, while scheduling strategies need to take into account data placement. We review the requirements and recent techniques for highly concurrent NUMA-aware analytics that take into consideration data locality, parallelism, and resource allocation (e.g., [2, 5, 9, 20, 25]).

4. MICRO-ARCHITECTURAL BEHAVIOR

Recent studies analyzing the micro-architectural behavior of OLTP workloads on modern hardware emphasize that OLTP exploits modern micro-architectural resources very poorly. More than half of the execution time goes to memory stalls [11]; as a result, on processors that have the ability to execute four instructions in a cycle, which is the most common on modern commodity hardware, OLTP achieves around one instruction per cycle (IPC) [29]. Such underutilization of micro-architectural features is a great waste of hardware resources.

Several proposals have been made to reduce memory stalls through improving instruction and data locality to increase cache hit rates. These range from cache-conscious data structures and algorithms [8] to sophisticated data partitioning and thread scheduling for data [22], and from compilation optimizations [26], advanced prefetching [12], to computation spreading [3, 7] and transaction batching for instructions [4, 14]. We illustrate the strengths and weaknesses of each technique with examples from recent work as well as present the key insights behind each of them.

In addition, several recent proposals opt for hardware specialization for some of the database operations ([16, 19, 31]). We briefly go over these techniques and emphasize their impact for emerging hardware technologies.

5. TUTORIAL OUTLINE

- INTRODUCTION AND OVERVIEW (15 minutes)
  - Tutorial overview: goal, audience, and schedule
  - Hardware trends
  - Problem statement:
    - three dimensions of scalability
    - challenges traditional data management systems face on modern hardware
- EXPLOITING THREAD-LEVEL PARALLELISM (45 minutes)
  - Scaling up OLTP
    - Communication types in transaction processing
    - Recent work on scaling-up OLTP on modern hardware
    - Mapping state-of-the-art design principles to the communication types they eliminate
  - Intra- & Inter-Query Parallelism
    - Revisiting database operators on multicores
    - Exploiting sharing opportunities among concurrent queries
- NUMA-AWARE OLTP (30 minutes)
  - Assumptions modern server hardware with NUMA changes for data management systems
  - Quantifying the impact of non-uniform communication on OLTP performance using various design options and workloads
  - Dynamically adjusting to the hardware topology and workload characteristics while designing transaction processing systems that can scale across sockets
- NUMA-AWARE OLAP (30 minutes)
  - Memory access bottlenecks in multisocket multicore architectures
  - NUMA-aware analytical algorithms
  - Outline of the requirements of a NUMA-aware execution engine for highly concurrent analytical workloads
- MICRO-ARCHITECTURAL UTILIZATION (50 minutes)

- Results from recent workload characterization studies
- Techniques to improve data cache locality
- Techniques to improve instruction cache locality
- Toward specialized hardware

- CONCLUSIONS AND FUTURE DIRECTIONS (10 minutes)

6. BIOGRAPHY

Anastasia Ailamaki is a Professor of Computer Sciences at École polytechnique fédérale de Lausanne (EPFL) in Switzerland. Her research interests are in database systems and applications, and in particular (a) in strengthening the interaction between the database software and emerging hardware and I/O devices, and (b) in automating database management to support computationally-demanding and demanding data-intensive scientific applications. She has received a Finmeccanica endowed chair from the Computer Science Department at Carnegie Mellon (2007), a European Young Investigator Award from the European Science Foundation (2007), an Alfred P. Sloan Research Fellowship (2005), eight best-paper awards at top conferences (2001-2012), and an NSF CAREER award (2002).

Erietta Liarou is a postdoctoral researcher at the Data-Intensive Applications and Systems (DIAS) lab of EPFL led by Prof. Anastasia Ailamaki. Her primary research interests include database architectures, transaction processing on modern hardware, data-stream processing, distributed query processing, and data analytics with emphasis on very large data management. She received her PhD in Computer Science from the University of Amsterdam, The Netherlands, in 2013, and she has also been with the System S group in IBM T.J.Watson Research Center, Hawthorne, NY, USA and the Intelligent Systems Laboratory in Technical University of Crete, Greece.

Pınar Tözün is a fifth year PhD student at École polytechnique fédérale de Lausanne (EPFL) working under the supervision of Prof. Anastasia Ailamaki in Data-Intensive Applications and Systems (DIAS) Laboratory. Her research focuses on scalability and efficiency of transaction processing systems on modern hardware. She received her BSc degree in Computer Engineering department of Koç University in 2009.

Danica Porobic is a fourth year PhD student at École polytechnique fédérale de Lausanne (EPFL) working under the supervision of Prof. Anastasia Ailamaki in Data-Intensive Applications and Systems (DIAS) Laboratory. Her research focuses on designing scalable transaction processing systems for non-uniform hardware. She has graduated top of her class with MSc and BSc in Informatics from University of Novi Sad and has worked at Oracle Labs and Microsoft SQL Server.

Iraklis Psaroudakis is a third year PhD student at École polytechnique fédérale de Lausanne (EPFL) working under the supervision of Prof. Anastasia Ailamaki in Data-Intensive Applications and Systems (DIAS) Laboratory. His research focuses on scheduling highly concurrent analytical workloads and he also co-operates with the SAP HANA database team. He has received his diploma from the School of Electrical and Computer Engineering of the National Technical University of Athens.

7. REFERENCES

[1] A. Ailamaki, R. Johnson, I. Pandis, and P. Tözün. Toward Scalable Transaction Processing: Evolution of Shore-MT. *PVLDB*, 6(11):1192–1193, 2013.

[2] M.-C. Albutiu, A. Kemper, and T. Neumann. Massively Parallel Sort-merge Joins in Main Memory Multi-core Database Systems. *PVLDB*, 5(10):1064–1075, 2012.

[3] I. Atta, P. Tözün, A. Ailamaki, and A. Moshovos. SLICC: Self-Assembly of Instruction Cache Collectives for OLTP Workloads. In *MICRO*, pages 188–198, 2012.

[4] I. Atta, P. Tözün, X. Tong, A. Ailamaki, and A. Moshovos. STREX: Boosting Instruction Cache Reuse in OLTP Workloads through Stratified Transaction Execution. In *ISCA*, pages 273–284, 2013.

[5] C. Balkesen, G. Alonso, J. Teubner, and M. T. Ozsu. Multi-Core, Main-Memory Joins: Sort vs. Hash Revisited. *PVLDB*, 7(1), 2014.

[6] G. Candea, N. Polyzotis, and R. Vingralek. A Scalable, Predictable Join Operator for Highly Concurrent Data Warehouses. *PVLDB*, 2(1):277–288, 2009.

[7] K. Chakraborty, P. M. Wells, and G. S. Sohi. Computation Spreading: Employing Hardware Migration to Specialize CMP Cores On-the-fly. In *ASPLOS*, pages 283–292, 2006.

[8] S. Chen, P. B. Gibbons, T. C. Mowry, and G. Valentin. Fractal Prefetching B+-Trees: Optimizing Both Cache and Disk Performance. In *SIGMOD*, pages 157–168, 2002.

[9] M. Dashti, A. Fedorova, J. Funston, F. Gaud, R. Lachaize, B. Lepers, V. Quéma, and M. Roth. Traffic Management: A Holistic Approach to Memory Placement on NUMA Systems. In *ASPLOS*, pages 381–394, 2013.

[10] C. Diaconu, C. Freedman, E. Ismert, P.-A. Larson, P. Mittal, R. Stonecipher, N. Verma, and M. Zwilling. Hekaton: SQL Server's Memory-optimized OLTP Engine. In *SIGMOD*, pages 1243–1254, 2013.

[11] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi. Clearing the Clouds: A Study of Emerging Scale-out Workloads on Modern Hardware. In *ASPLOS*, pages 37–48, 2012.

[12] M. Ferdman, C. Kaynak, and B. Falsafi. Proactive Instruction Fetch. In *MICRO*, pages 152–162, 2011.

[13] G. Giannikis, G. Alonso, and D. Kossmann. SharedDB: Killing One Thousand Queries with One Stone. *PVLDB*, 5(6):526–537, 2012.

[14] S. Harizopoulos and A. Ailamaki. STEPS Towards Cache-Resident Transaction Processing. In *VLDB*, pages 660–671, 2004.

[15] S. Harizopoulos, V. Shkapenyuk, and A. Ailamaki. QPipe: A Simultaneously Pipelined Relational Query Engine. In *SIGMOD*, pages 383–394, 2005.

[16] R. Johnson and I. Pandis. The bionic dbms is coming, but what will it look like? In *CIDR*, 2013.

[17] R. Johnson, I. Pandis, and A. Ailamaki. Eliminating unscalable communication in transaction processing. *VLDBJ*, 23(1):1–23, 2014.

[18] A. Kemper and T. Neumann. HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots. In *ICDE*, pages 195–206, 2011.

[19] O. Kocberber, B. Grot, J. Picorel, B. Falsafi, K. Lim, and P. Ranganathan. Meet the Walkers: Accelerating Index Traversals for In-memory Databases. In *MICRO*, pages 468–479, 2013.

[20] Y. Li, I. Pandis, R. Mueller, V. Raman, and G. Lohman. NUMA-aware Algorithms: The Case of Data Shuffling. In *CIDR*, 2013.

[21] I. Pandis, P. Tözün, R. Johnson, and A. Ailamaki. PLP: Page Latch-free Shared-everything OLTP. *PVLDB*, 4(10):610–621, 2011.

[22] D. Porobic, E. Liarou, P. Tözün, and A. Ailamaki. ATraPos: Adaptive Transaction Processing on Hardware Islands. In *ICDE*, 2014.

[23] D. Porobic, I. Pandis, M. Branco, P. Tözün, and A. Ailamaki. OLTP on Hardware Islands. *PVLDB*, 5(11):1447–1458, 2012.

[24] I. Psaroudakis, M. Athanassoulis, and A. Ailamaki. Sharing Data and Work Across Concurrent Analytical Queries. *PVLDB*, 6(9):637–648, 2013.

[25] I. Psaroudakis, T. Scheuer, N. May, and A. Ailamaki. Task Scheduling for Highly Concurrent Analytical and Transactional Main-Memory Workloads. *ADMS*, 2013.

[26] A. Ramirez, L. A. Barroso, K. Gharachorloo, R. Cohn, J. Larriba-Pey, P. G. Lowney, and M. Valero. Code Layout Optimizations for Transaction Processing Workloads. In *ISCA*, pages 155–164, 2001.

[27] M. Stonebraker, S. Madden, D. J. Abadi, S. Harizopoulos, N. Hachem, and P. Helland. The End of an Architectural Era: (It's Time for a Complete Rewrite). In *VLDB*, pages 1150–1160, 2007.

[28] A. Thomson, T. Diamond, S.-C. Weng, K. Ren, P. Shao, and D. J. Abadi. Calvin: Fast Distributed Transactions for Partitioned Database Systems. In *SIGMOD*, pages 1–12, 2012.

[29] P. Tözün, I. Pandis, C. Kaynak, D. Jevdjic, and A. Ailamaki. From A to E: Analyzing TPC's OLTP Benchmarks – The obsolete, the ubiquitous, the unexplored. In *EDBT*, pages 17–28, 2013.

[30] S. Tu, W. Zheng, E. Kohler, B. Liskov, and S. Madden. Speedy Transactions in Multicore In-memory Databases. In *SOSP*, pages 18–32, 2013.

[31] L. Wu, A. Lottarini, T. K. Paine, M. A. Kim, and K. A. Ross. Q100: The Architecture and Design of a Database Processing Unit. In *ASPLOS*, pages 255–268, 2014.