Toward Scalable Transaction Processing

Evolution of Shore-MT

Anastasia Ailamaki Ryan Johnson[†] Ippokratis Pandis[*] Pınar Tözün

[†]University of Toronto
[*]IBM Almaden Research Center
École Polytechnique Fédérale de Lausanne

ABSTRACT

Designing scalable transaction processing systems on modern multicore hardware has been a challenge for almost a decade. The typical characteristics of transaction processing workloads lead to a high degree of unbounded communication on multicores for conventional system designs.

In this tutorial, we initially present a systematic way of eliminating scalability bottlenecks of a transaction processing system, which is based on minimizing the unbounded communication. Then, we show several techniques that apply the presented methodology to minimize logging, locking, latching etc. related bottlenecks of transaction processing systems. In parallel, we demonstrate the internals of the Shore-MT storage manager and how they have evolved over the years in terms of scalability on multicore hardware through such techniques. We also teach how to use Shore-MT with the various design options it offers through its application layer Shore-Kits and Metadata Frontend.

1. INTRODUCTION

In step with Moore's Law, hardware gives us more and more opportunities for parallelism rather than faster processors over the recent years. Exploiting this parallelism is crucial to utilize the available architectural resources and enable faster software. However, designing scalable systems that can take advantage of the underlying parallelism remains as a challenging task for the software developers from various fields.

Transaction processing systems exhibit high concurrency, and therefore, offer a good opportunity for more parallelism. However, the inherent communication in traditional high performance transaction processing systems lead to scalability bottlenecks on today's multicore hardware. Increased hardware parallelism does not automatically bring increased performance for transaction processing. Even systems that are able to scale very well on one generation of multicores might fail to scale-up on the next generation [7].

In this two hour tutorial, we initially teach a clear methodology for scaling-up transaction processing systems on multicore hardware. More specifically, we classify three types of communication in a typical transaction processing system: *unbounded*, *fixed*, and *cooperative* [3]. Our main observation is that not all the communication within a system is harmful in terms of scalability. 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.

Then, we show how effective our methodology is in practice for scaling-up transaction processing systems on multicore hardware. We give examples of some techniques that remove the *unbounded* communication step by step while solving the problem of locking [2, 7], logging [5, 6], and latching [8, 11] bottlenecks in a traditional transaction processing system. We observe how the Shore-MT storage manager [4, 10, 9] has evolved over the years through applying such techniques. We present the implementation of some of these techniques within Shore-MT, illustrating its internals in parallel.

Finally, we introduce the two powerful application layers of Shore-MT, Shore-Kits and Metadata Frontend. Shore-Kits [9] has the implementation of various database workloads for Shore-MT as well as the interface to run Shore-MT using its various design options. The Metadata Frontend, on the other hand, enables creating database tables interactively and running basic ad-hoc queries on top of Shore-MT. Moreover, it provides a brief API to easily wrap it up using tools like SWIG [1] and create simple benchmarks using different scripting languages.

The goal of this tutorial is to give guidelines for building scalable transaction processing systems by eliminating not all but only the unscalable communication in the system. In addition, it teaches how to use a state-of-the-art open-source scalable storage manager, Shore-MT, as a test-bed for future research. The basic methodology introduced here, however, can be applied to any software system that aims scalability on modern multicore hardware.

2. TUTORIAL OUTLINE

The tutorial is formed of five parts, which are given below:

1. Introduction:

This part of the tutorial takes 10 minutes. We first briefly go over the evolution of modern hardware. Then, we survey the related work on the scalability of transaction processing systems. Finally, we introduce Shore-MT storage manager and its features at the high level.

2. Eliminating the Unbounded Communication:
This part of the tutorial takes 50 minutes. Here we describe the communication types in transaction processing in detail. We layout our methodology for system scalability, which requires a thorough understanding of a system's communication patterns and aims to eliminate the unbounded communication. We also give examples on various techniques that apply this methodology to eliminate the locking, logging, and latching bottlenecks.

3. Hands-On – Shore-MT:
This part of the tutorial takes 15 minutes. We illustrate the internals of Shore-MT storage manager and the code parts from the different design options that implement the various scalability improvements mentioned during the previous part. We also show how to run basic unit tests and experiments with Shore-MT.

4. Hands-On – Shore-Kits:
This part of the tutorial takes 25 minutes. We first examine the internals of Shore-Kits. Then, we run Shore-Kits with its different configuration options to see how to enable/disable the various techniques prototyped inside Shore-MT. Finally, we go over how to extend Shore-Kits with different benchmarks.

5. Hands-On – Shore-Front:
This part of the tutorial takes 20 minutes. We initially demonstrate how to use the frontend to perform simple tasks. Then, we show how one can use different scripting languages to automatically create and run micro-benchmarks with this metadata frontend.

3. BIOGRAPHY

Anastasia Ailamaki (anastasia.ailamaki@epfl.ch)
Anastasia Ailamaki is a Professor of Computer Sciences at the 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). She earned her Ph.D. in Computer Science from the University of Wisconsin-Madison in 2000. She is a senior member of the IEEE and a member of the ACM, and has also been a CRA-W mentor.

Ryan Johnson (ryan.johnson@cs.utoronto.ca)
Ryan Johnson is an Assistant Professor at the University of Toronto specializing in systems aspects of database engines, particularly in the context of modern hardware. He contributed heavily to the initial development and performance tuning of Shore-MT. He graduated with M.S. and PhD degrees in Computer Engineering from Carnegie Mellon University in 2010, after completing a B.S. in Computer Engineering at Brigham Young University in 2004. In addition to his work with database systems, Johnson has interests in computer architecture, operating systems, compilers, and hardware design.

Ippokratis Pandis (ipandis@us.ibm.com)
Ippokratis Pandis is a Research Staff Member (RSM) at IBM Research Almaden. His research focuses on efficient, scalable data management and he is actively involved in IBMs DB2 BLU project. Prior joining IBM, Ippokratis graduated with a PhD in Electrical and Computer Engineering from Carnegie Mellon University where he worked on scalable transaction processing on multisocket and multicore hardware, contributing to the development of Shore-MT and Shore-Kits. His PhD thesis was on the data-oriented transaction processing architecture (DORA).

Pınar Tözün (pinar.tozun@epfl.ch)
Pınar Tözün is a fourth year PhD student at École Polytechnique Fédérale de Lausanne (EPFL) working under 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 and she actively contributes to the development and maintenance of Shore-MT and Shore-Kits. Before starting her PhD, she received her BSc degree in Computer Engineering department of Koç University in 2009 as the top student.

4. REFERENCES

[1] Simplified wrapper and interface generator (SWIG). Available at http://www.swig.org.

[2] R. Johnson, I. Pandis, and A. Ailamaki. Improving OLTP scalability using speculative lock inheritance. *PVLDB*, 2(1):479–489, 2009.

[3] R. Johnson, I. Pandis, and A. Ailamaki. Eliminating unscalable communication in transaction processing. *VLDB J.*, 2013.

[4] R. Johnson, I. Pandis, N. Hardavellas, A. Ailamaki, and B. Falsafi. Shore-MT: a scalable storage manager for the multicore era. In *EDBT*, pages 24–35, 2009.

[5] R. Johnson, I. Pandis, R. Stoica, M. Athanassoulis, and A. Ailamaki. Aether: a scalable approach to logging. *PVLDB*, 3:681–692, 2010.

[6] R. Johnson, I. Pandis, R. Stoica, M. Athanassoulis, and A. Ailamaki. Scalability of write-ahead logging on multicore and multisocket hardware. *VLDB J.*, 21:239–263, 2012.

[7] I. Pandis, R. Johnson, N. Hardavellas, and A. Ailamaki. Data-oriented transaction execution. *PVLDB*, 3(1):928–939, 2010.

[8] I. Pandis, P. Tözün, R. Johnson, and A. Ailamaki. PLP: page latch-free shared-everything OLTP. *PVLDB*, 4(10):610–621, 2011.

[9] Shore-MT. Shore-MT and Shore-Kits Code Repositories. Available at https://bitbucket.org/shoremt.

[10] Shore-MT. Shore-MT Official Website. Available at http://diaswww.epfl.ch/shore-mt/.

[11] P. Tözün, I. Pandis, R. Johnson, and A. Ailamaki. Scalable and dynamically balanced shared-everything OLTP with physiological partitioning. *VLDB J.*, 22(2):151–175, 2013.