Scalable Transaction Processing on Multicores
[Shore-MT & DORA]

Ippokratis Pandis

Ryan Johnson, Nikos Hardavellas, Anastasia Ailamaki
CMU & EPFL

HPTS - 26 Oct 2009
Multicore “cluster on a chip”

Then

CPU

Small Machine

CPU CPU CPU CPU
CPU CPU CPU CPU

Big Machine

Now

CPU

Multicore Machine

Core Core Core Core
Core Core Core Core

Parallelism of yesterday’s “big” machine on one chip
Database Engine Scalability

Best scalability just 30% of ideal
Shared Everything vs. Nothing

- Shared Everything
  - Hard to scale

- Shared Nothing
  - Multiple processes, physically separated data
  - Explicit contention control
  - Perfectly partitionable workload
  - Memory pressure: redundant data/structures

Two approaches complimentary

Focus on scalability of a single (shared everything) instance
Shore-MT

- Multithreaded version of Shore
- Why Shore?
  - State-of-the-art DBMS features
  - Two-phase row-level locking
  - ARIES-style logging/recovery
- Shore similar at instruction-level with commercial DBMSs

High-performing, scalable conventional engine
Available at: http://diaswww.epfl.ch/shore-mt/
Scalability on Even Higher Parallelism

- Lock manager overhead dominant
- Typical scenario: contention for compatible locks
Data-oriented Transaction Execution

- It is not the transaction which dictates what data the transaction-executing thread will access
- Break each transaction into smaller actions
  - Depending on the data they touch
- Execute actions by “data-owning” threads
- Distribute and privatize locking, data accesses across the chip

New data-oriented execution model
- Reduce overhead of locking and data accesses
DORA vs. Conventional – Throughput

Throughput (tps)

TPC-C Payment

Avoid expensive (centralized) lock manager operations

Immune to centr. lock manager

Higher performance in the entire load spectrum

Sun Niagara II 64 HW Contexts

Intra-xaction parallelism on light loads

DORA

BASELINE

2x

20%

40+%

Higher performance in the entire load spectrum
DORA vs. Conventional – At 100% CPU

- Eliminate contention on the centr. lock manager
- Significantly reduced work (lightweight locks)
Roadmap

• Introduction

• Conventional execution

• Data-oriented transaction execution

• Evaluation

• Conclusions
The higher the HW parallelism  →  Longer Queues of Requests  →  Longer CSs  →  Higher Contention
Conventional - Example

Transaction:
I   D
u(wh)  
u(cust)  
u(ord)  

I = Instruction
D = Data
Conventional - Access Pattern

TPC-C Payment - DISTRICT Records

- Unpredictable access pattern
- Source of contention
Roadmap

• Introduction
• Conventional execution
• Data-oriented transaction execution
• Evaluation
• Conclusions
Dora - Access Pattern

- Predictable access patterns
- Optimizations possible (e.g. no centralized locks)
Transaction Flow Graph

• Each transaction input is a graph of Actions & RVPs

• Actions
  – Identified by:
  – Table/Index it is accessing
  – Subset of primary key

• Rendezvous Points
  – Decision points (commit/abort)
  – Separate different phases
  – Counter of the # of actions to report
  – Last to report initiates next phase
  – Enqueue the actions of the next phase
Partitions & Executors

• Partitions at each table
  – Local lock table
    • Map \{partof(Key), LockMode\}
    • List of blocked actions
  – Input queue
    • New actions
  – Completed queue
    • On xct commit/abort
    • Remove from local lock table

• Executor thread
  – Loop completed/input queue
  – Asynchronous communication / event-based
Dora - Example

Transaction:
I  D
u(wh)  
u(cust)  
u(ord)  

CPU-0

CPU-1

CPU-2

CPU

L1

L2

MEM

I/O

WH

CUST

ORD

Centralized lock free

Improved data reuse
Dora vs. Shared-nothing

- No physical partition of data
- No duplicated data structures
- Smaller memory footprint
- A single log manager
- No need for distributed transactions
  - No need for 2PC

Dora is NOT a shared-nothing system

Combines benefits of both
Roadmap

- Introduction
- Conventional execution
- Data-oriented transaction execution
- Evaluation
- Conclusions
Experimental Setup

Hardware

- Sun Niagara II processor
- 8 cores with 8 HW contexts per core (64 HW ctxs)
- 32 GB main memory

Workloads

- Update-intensive, short-running transactions
- TPC-C – 100 warehouses (13GB)
- TM1 – 1M subscribers (1.5GB)
Eliminates contention on the lock manager

- Linear scalability to 64 HW ctxs
- Immune to oversaturation
Response time for single client

- Exploits intra-xct parallelism
- Lower response times on low-load
Higher peak performance
Always close to 100% CPU utilization
Roadmap

• Introduction
• Conventional execution
• Data-oriented transaction execution
• Evaluation
• Conclusions
Summary

• Large number of active threads stress scalability of database system

• Data-oriented transaction execution

• Benefits of shared-nothing w/o physical data partitioning

• Small modifications on a conventional storage engine

• Higher performance on the entire load spectrum