H-Store: A Specialized Architecture for High-throughput OLTP Applications

Evan Jones (MIT)
Andrew Pavlo (Brown)
13th Intl. Workshop on High Performance Transaction Systems
October 26, 2009
Intel Xeon E5540 (Nehalem/Core i7)

Source: Intel 64 and IA-32 Architectures Optimization Reference Manual
Distributed Clusters
Scaling OLTP on Multi-Core?

*Use a distributed shared-nothing design*
How to Make a Faster OLTP DBMS

- Main memory storage
  - Replication for durability

- Explicitly partition the data

- Specialized concurrency control
  - Stored procedures only
  - Single partition: execute one transaction at a time
  - Multiple partitions: supported but slow
OLTP: Where does the time go?

Users Rely on Partitioning

December, 2002

What about multi-core?

- Traditional approach:
  - One database process
  - Thread per connection
  - Shared-memory, locks and latches

- H-Store approach:
  - Thread per partition
  - Distributed transactions
Example Microbenchmark

- One table per client
  ```
  Table(id INTEGER, counter INTEGER)
  ```
- Each client executes the following query:
  ```
  UPDATE Table
  SET counter = counter + 1
  WHERE id = 0;
  ```
- Add clients to find maximum throughput
- Data on RAM disk
Experimental Configuration
Partitions versus Threads

Relative Speed

CPUs

October 26, 2009
Scalability Analysis

Partitions scale better than threads.

- Threads: contention for shared resources [1]
- Partitions: memory bottleneck causes sublinear scaling

**H-Store: Not just for distributed shared-nothing clusters**

Multi-core Design Problem

- How to automatically create a data placement scheme to improve multi-core throughput?

- Data Partitioning:
  - Maximize the number of single-partition transactions.

- Data Placement:
  - Maximize the number of single-node transactions.
Database Partitioning

- Select partitioning keys and construct schema tree.

TPC-C Schema

- WAREHOUSE
- DISTRICT
- CUSTOMER
- ORDERS
- ITEM
- STOCK
- ORDER_ITEM

Schema Tree

- WAREHOUSE
- DISTRICT
- CUSTOMER
- ORDERS
- ORDER_ITEM
- STOCK
- ITEM

Replicated
Database Partitioning

- Combine table fragments into partitions.

**Schema Tree**

- WAREHOUSE
- DISTRICT
- CUSTOMER
- ORDERS
- ORDER_ITEM

**Partitions**

- P1
- P2
- P3
- P4
- P5

*Replicated*
Data Placement

- Assign partitions to cores on each node.

Partitions

Cluster Node

Partition Affinity

Node 1

Node \(n\)
H-Store’s Future

- New Name. New Company.
- Six full-time developers.
- Open-source project (GPL)
- Beta by end of 2009
  - Multiple deployments in financial service areas.
More Information

- H-Store Info + Papers:
  - http://db.cs.yale.edu/hstore/

- VoltDB Project Information:
  - http://www.voltodb.org/