Methods for the Optimization of Parallel OLTP Systems

@andy_pavlo

Thesis Proposal
May 2012
Tellin’ him just be ready set
Pack ya shit up quick;
And when I hit,
be prepared to jet
Fast + Cheap
Legacy Systems

TPC-C NewOrder

- Real Work: 12.3%
- Buffer Pool: 29.6%
- Latching: 10.2%
- Locking: 18.7%
- Logging: 21.1%
- B-Tree Keys: 8.1%

OLTP Through the Looking Glass, and What We Found There
SIGMOD 2008
OLTP Transactions

Fast  Repetitive  Small
H-Store: A High-Performance, Distributed Main Memory Transaction Processing System

H-Store

Database Node

Core

Txn Coordinator

Execution Engine

Partition Data

Main Memory

Execution Engine

Partition Data

Client Application

Database Cluster
I wasn’t mad until these tricks played me,
It’s time to sanitize my posse.
Optimization #1: Partition database to reduce the number of distributed txns.
<table>
<thead>
<tr>
<th>c_id</th>
<th>c_w_id</th>
<th>c_l</th>
<th>c_id</th>
<th>c_w_id</th>
<th>c_last</th>
<th>...</th>
<th>o_c_id</th>
<th>o_w_id</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>5</td>
<td>RZA600614</td>
<td>78703</td>
<td>1004</td>
<td>5</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1002</td>
<td>3</td>
<td>GZA260923</td>
<td>78704</td>
<td>1002</td>
<td>3</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1003</td>
<td>12</td>
<td>Raekw476586</td>
<td>78705</td>
<td>1006</td>
<td>7</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1004</td>
<td>5</td>
<td>Deck578045</td>
<td>78706</td>
<td>1005</td>
<td>6</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1005</td>
<td>6</td>
<td>Killah47698</td>
<td>78707</td>
<td>1005</td>
<td>6</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1006</td>
<td>7</td>
<td>ODB780085</td>
<td>78708</td>
<td>1003</td>
<td>12</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Client Application

```
NewOrder(5, "Method Man", 1234)
```
NewOrder

SELECT * FROM WAREHOUSE WHERE W_ID = 10;

SELECT * FROM DISTRICT WHERE D_W_ID = 10 AND D_ID = 9;

INSERT INTO ORDERS (O_W_ID, O_D_ID, O_C_ID,...) VALUES (10, 9, 12345,...);

...
Algorithm Comparison

(cost estimate)

![Comparison Graphs]

- **TPC-C**: Execution Cost: Horticulture vs. State-of-the-Art; Skew Cost: Horticulture vs. State-of-the-Art
Throughput

- **Horticulture**
- **State-of-the-Art**

---

**TATP**
+88%

**TPC-C**
+16%

**TPC-C Skewed**
+183%
Search Times

- **TATP**
- **SEATS**
- **TPC-C**
- **TPC-C Skewed**
- **AuctionMark**
- **TPC-E**
H-Store

Client Application

Undo Log

Database Cluster
Grab your glocks when you see 2pac.
Call the cops when you see 2pac.
Optimization #2: Predict what txns will do before they execute.

On Predictive Modeling for Optimizing Transaction Execution in Parallel OLTP Systems
Current State

```
begin
```

Input Parameters:
- \( w_{id} = 0 \)
- \( i_{w\_id} = [0, 1] \)
- \( i_{ids} = [1001, 1002] \)

GetWarehouse:
```
SELECT * FROM WAREHOUSE
WHERE w_{id} = ?
```

GetWarehouse:
Count: 0
Partitions: \{ 0 \}
Previous: \( \emptyset \)

GetWarehouse:
Count: 0
Partitions: \{ 1 \}
Previous: \( \emptyset \)

CheckStock
Count: 0
Partitions: \{ 0 \}
Previous: \( \emptyset \)

CheckStock
Count: 0
Partitions: \{ 1 \}
Previous: \( \emptyset \)

CheckStock
Count: 1
Partitions: \{ 1 \}
Previous: \{ 0, 1 \}

CheckStock
Count: 1
Partitions: \{ 0 \}
Previous: \{ 0, 1 \}

Log

```
22
```
SELECT S_QTY FROM STOCK WHERE S_W_ID = ? AND S_I_ID = ?;

Input Parameters:
- \( w\_id = 0 \)
- \( i\_w\_ids = [0, 1] \)
- \( i\_ids = [1001, 1002] \)

CheckStock:

\[
\begin{align*}
\text{SELECT} & \quad S\_QTY \quad \text{FROM} \quad \text{STOCK} \\
\text{WHERE} & \quad S\_W\_ID = \_? \\
& \quad \text{AND} \quad S\_I\_ID = \_? ;
\end{align*}
\]

InsertOrder:

\[
\begin{align*}
\text{INSERT} & \quad \text{INTO} \quad \text{ORDERS} \\
(o\_id, o\_w\_id) & \quad \text{VALUES} \quad (\_?, \_?) ;
\end{align*}
\]
\[ w_{id}=0 \]
\[ i_w_{ids}=[0,1] \]
\[ i_{ids}=[1001,1002] \]
Throughput

(txn/s)

Houdini  Assume Single-Partitioned

TATP +57%

TPC-C +126%

AuctionMark +117%
Recap

Optimize Single-Partition Execution

H-Store: A High-Performance, Distributed Main Memory Transaction Processing System

Minimize Distributed Transactions

Skew-Aware Automatic Database Partitioning in Shared-Nothing, Parallel OLTP Systems
A. Pavlo, C. Curino, and S. Zdonik

Identify Distributed Transactions

On Predictive Modeling for Optimizing Transaction Execution in Parallel OLTP Systems
A. Pavlo, E. P. C. Jones, and S. Zdonik
Two-Phase Commit
Thesis Proposal:
Amortize distributed txn coordination with creative scheduling.

Dr. Zdonik – Or How I Learned to Stop Worrying And Love Distributed Transactions
Work in Progress
Transaction Batching

Two-Phase Commit
  Prepare

Database Cluster

Client Application

Transaction Batching

Client Application

Client Application
Can I lock partition #3?

Yes! Go ahead!

Non-Conflicting Txns

Speculative Transactions

Single Partitioning Txns
Speculative Queries

Can I lock partition #3?

Execute Query

Yes! Go ahead!
Conclusion: Achieving fast performance is more than just using only RAM.

Proposed Schedule:
Coding + Experiments (Summer)
Paper Submission (Fall/Winter)
Thesis Writing (Spring)
H-Store

http://github.com/apavlo/h-store

http://hstore.cs.brown.edu

http://wikipedia.org/wiki/H-Store