MY DATABASE SYSTEM IS THE ONLY THING I CAN TRUST
Thirty Years Ago...
Interactive Transactions

Small # of CPU Cores

Small Memory Sizes
TPC-C BENCHMARK
Warehouse Order Processing

NewOrder Transaction
1. Check item stock level.
2. Create new order information.
3. Update item stock levels.
TPC-C BENCHMARK

Warehouse Order Processing

<table>
<thead>
<tr>
<th>CPU CORES</th>
<th>TXN/SEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>5,000</td>
</tr>
<tr>
<td>3</td>
<td>10,000</td>
</tr>
<tr>
<td>4</td>
<td>15,000</td>
</tr>
<tr>
<td>5</td>
<td>20,000</td>
</tr>
</tbody>
</table>

- MySQL
- Postgres
TRADITIONAL DBMS
Measured CPU Cycles

OLTP THROUGH THE LOOKING GLASS, AND WHAT WE FOUND THERE
HOW TO SCALE UP WITHOUT GIVING UP TRANSACTIONS?
H-Store: A High-Performance, Distributed Main Memory Transaction Processing System

DISK ORIENTED
MAIN MEMORY STORAGE

CONCURRENT EXECUTION
SERIAL EXECUTION

HEAVYWEIGHT RECOVERY
COMPACT LOGGING
PARTITIONS

SINGLE-THREADED EXECUTION ENGINES
STORED PROCEDURE

 VoteCount:

```sql
SELECT COUNT(*)
FROM votes
WHERE phone_num = ?;
```

 InsertVote:

```sql
INSERT INTO votes
VALUES (?, ?, ?, ?);
```

```java
run(phoneNum, contestantId, currentTime) {
    result = execute(VoteCount, phoneNum);
    if (result > MAX_VOTES) {
        return (ERROR);
    }
    execute(InsertVote, phoneNum, contestantId, currentTime);
    return (SUCCESS);
}
```
Transaction
Result

Application

SNAPSHOTS
TPC-C BENCHMARK
Warehouse Order Processing

- MySQL
- Postgres
- H-Store
DISTRIBUTED TRANSACTIONS
TPC-C BENCHMARK
8 Cores per Node
10% Distributed Transactions

- H-Store
DISTRIBUTED TRANSACTIONS

Application
DISTRIBUTED TRANSACTIONS

Application
DISTRIBUTED TRANSACTIONS

Query Count

P1
P2
P3
P4

Application
KNOW WHAT TRANSACTIONS WILL DO BEFORE THEY START
BUT PEOPLE ALWAYS GIVE ME BAD ADVICE
DON’T GET INVOLVED WITH COMPUTERS. YOU’LL NEVER MAKE ANY MONEY.
DON'T GET A PHD. EVERYONE WILL THINK YOU ARE A JERK.
THE DATABASE SYSTEM ALWAYS HAS MORE INFORMATION
DO USE MACHINE LEARNING TO PREDICT TRANSACTION BEHAVIOR.

ON PREDICTIVE MODELING FOR OPTIMIZING TRANSACTION EXECUTION IN PARALLEL OLTP SYSTEMS
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);

WORKLOAD

Feature Clusterer

Model Generator

Classifier

Decision Tree

Markov Models
DISTRIBUTED TRANSACTIONS
TPC-C BENCHMARK
8 Cores per Node
10% Distributed Transactions

- Naïve
- Houdini

OPTIMAL

TXN/SEC
0
5,000
10,000
15,000
20,000
25,000

NODES
1
2
4

2x
TPC-C BENCHMARK

8 Cores per Node
10% Distributed Transactions

Naïve  Houdini

TXN/SEC  NODES

1  2  4
DISTRIBUTED TRANSACTIONS

SP1 - Waiting for Query Result
DISTRIBUTED TRANSACTIONS

SP1 - Waiting for Query Result

SP2 - Waiting for Query Request
DISTRIBUTED TRANSACTIONS

SP1 - Waiting for Query Result
SP2 - Waiting for Query Request
SP3 - Two-Phase Commit
TRANSACTION STALL POINTS

**BASE PARTITION**
- SP1: Waiting for Query Result (45%)
- SP2: Waiting for Query Request (37%)
- SP3: Two-Phase Commit (18%)
- Real Work (73%)

**REMOTE PARTITION**
- SP2: Waiting for Query Request (73%)
- SP3: Two-Phase Commit (5%)
- Real Work (22%)

DO SOMETHING USEFUL WHEN STALLED
DON’T BE SURPRISED IF YOU & KB DON’T LAST THROUGH GRAD SCHOOL.
DON’T BE STAN’S STUDENT IF YOU GO TO BROWN.
DO USE MACHINE LEARNING TO SCHEDULE SPECULATIVE TASKS.
SERIALIZABLE SCHEDULE

Distributed Transaction

Single-Partition Transaction

Single-Partition Transaction
SERIALIZABLE SCHEDULE

Distributed Transaction

Speculative Transaction

VERIFY

Speculative Transaction
SPECULATIVE TRANSACTIONS

Transaction Queue

Speculation Candidate:

Distributed Transaction:

READ X

WRITE X

begin
QueryX
Count: 0
Partitions: { 0, 1 }
Previous: 0

QueryY
Count: 0
Partitions: { 1, 0 }
Previous: 0

QueryX
Count: 0
Partitions: { 0, 1 }
Previous: 0

QueryZ
Count: 0
Partitions: { 0, 1 }
Previous: 0

begin
Query1
Count: 0
Partitions: {} Previous: 0

Query3
Count: 1
Partitions: {} Previous: 0

Query3
Count: 2
Partitions: {} Previous: 0

commit

begin
Query3
Count: 0
Partitions: {} Previous: 0

commit

45
SPECULATIVE TRANSACTIONS

Transaction Queue

Speculation Candidate:

Distributed Transaction:

Hermes
SPECULATIVE QUERIES

Distributed Transaction:

begin
QueryX
  Counter: 0
  Partitions: (0)
  Previous: 0
QueryY
  Counter: 0
  Partitions: (1)
  Previous: 0
QueryX
  Counter: 0
  Partitions: (0, 1)
  Previous: 0, 1
QueryZ
  Counter: 0
  Partitions: (0, 1)
  Previous: 0, 1
QueryZ
  Counter: 0
  Partitions: (1)
  Previous: 1
commit
SPECFULATIVE QUERIES

Distributed Transaction:

begin  QueryX  Counter: 0  Partitions: {0}  Previous: 0
        QueryY  Counter: 0  Partitions: {1}  Previous: 0
        QueryX  Counter: 0  Partitions: {0}  Previous: 0
        QueryZ  Counter: 0  Partitions: {1}  Previous: 0
        QueryZ  Counter: 0  Partitions: {1}  Previous: 0
        commit

Hermes
SPECULATIVE QUERIES

Distributed Transaction:

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

QueryY:
SPECULATIVE QUERIES

Distributed Transaction:

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

QueryY:

- Counter: 0
- Partitions: (1)
- Previous: (0, 1)

QueryZ
- Counter: 0
- Partitions: (1)
- Previous: (0, 1)
begin

GetWarehouse
- Counter: 0
- Partitions: { 1 }
- Previous: ∅

GetWarehouse
- Counter: 0
- Partitions: { 0 }
- Previous: ∅

CheckStock
- Counter: 0
- Partitions: { 0 }
- Previous: { 1 }

CheckStock
- Counter: 0
- Partitions: { 1 }
- Previous: { 0 }

CheckStock
- Counter: 1
- Partitions: { 1 }
- Previous: { 0, 1 }

CheckStock
- Counter: 1
- Partitions: { 0 }
- Previous: { 0, 1 }

Transaction Parameters:
- w_id = 0
- i_w_ids = [1, 0]
- i_ids = [1001, 1002]

GetWarehouse:
SELECT * FROM WAREHOUSE
WHERE W_ID = ?
Transaction Parameters:

- `w_id=0`
- `i_w_ids=[1, 0]`
- `i_ids=[1001, 1002]`

CheckStock:

```sql
SELECT S_QTY FROM STOCK
WHERE S_W_ID = ?
AND S_I_ID = ?
```
begin

GetWarehouse
- Counter: 0
- Partitions: { 1 }
- Previous: ∅

GetWarehouse
- Counter: 0
- Partitions: { 0 }
- Previous: ∅

CheckStock
- Counter: 0
- Partitions: { 0 }
- Previous: { 1 }

CheckStock
- Counter: 0
- Partitions: { 1 }
- Previous: { 0 }

CheckStock
- Counter: 1
- Partitions: { 1 }
- Previous: { 0, 1 }

CheckStock
- Counter: 1
- Partitions: { 0 }
- Previous: { 0, 1 }

Transaction Parameters:
- w_id=0
- i_w_ids=[1, 0]
- i_ids=[1001, 1002]

CheckStock:
- SELECT S_QTY FROM STOCK
  WHERE S_W_ID = ?
  AND S_I_ID = ?;
VERIFICATION

Distributed Transaction

Speculative Transactions

Hermes
TPC-C BENCHMARK
8 Cores per Node
10% Distributed Transactions

Spec Queries
None Spec Queries Spec Txns All

TXN/SEC
1 2 4

NODES
0 10,000 20,000 30,000 40,000 50,000

TPC-C Benchmark

Hermes
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

Identify Distributed Transactions
ON PREDICTIVE MODELING FOR OPTIMIZING TRANSACTION EXECUTION IN PARALLEL OLTP SYSTEMS

Utilize Transaction Stalls
THE ART OF SPECULATIVE EXECUTION
In Progress (August 2013)
FUTURE WORK FUTURE WORK
One Size
Almost Fits All™
Escape From Planet Zdonik
(i.e., Andy Needs to Get Tenure)
Beyond the ‘Stores

• Non-Partitionable Workloads.
• The Poor Man’s Spanner.
• Scientific Databases.
DON’T MESS IT UP WITH KB.