Benchmarking StrongDBMS

MALCOLM CROWE AND CALLUM FYFFE

JUNE 2019

Quick Intro: StrongDBMS

- Introduced in the paper
  - StrongDBMS: Built from immutable components
- Database file is the transaction log
  - Verifies serialization of transactions committed
- Relational client-server ACID design
  - First Committer Wins concurrency control
  - Under development: version 0.1
- Open Source: on github.com
  - Implemented in C# and Java

StrongDBMS is an RDBMS

- Most disk-based RDBMS use pagefiles
  - Pages Include data and indexes
  - For 1970s this saved on RAM
- Today's RAM is larger:
  - There are several memory-based RDBMS
  - But then there is no durability of commits
- StrongDBMS: log file is committed data
  - RAM contains indexes, uncommitted obs

TPC-C Benchmark

- In wide use since 1992 (tpc.org)
  - Current version 2010
  - For client-server systems
- OLTP simulation
  - Warehouse(s) for 100000 products
  - Each has 10 districts, 3000 customers each
  - Orders have 5-15 order lines
  - New Order typically involves 25 steps
  - Committed New Orders is the measure

Testing: Two main modes

- First: just see how fast the server copes
  - With a stream of New Order activity
- Second: model real clerk behaviour
  - Where each task takes minimum ~18 sec
  - Conflicts built in to the specification
    - The database must remain consistent
  - Run test for 10 minutes: same random gen
    - Allows comparison between products
- Third: Find # of clerks for saturation

Simulating Clerk activity

- Specification is for a mix of tasks
  - New Order
  - Order status
  - Stock level
  - Delivery
  - Delivery report
  - Payment
- Clerk processing phoned-in orders etc

The TPCC specification

- With several warehouses
  - And 1 clerk per warehouse
  - Design gives about 3-4% conflicts
- For me its important to test conflicts
- All clerks are for the same warehouse
  - This means that conflicts predominate

Task descriptions

- Each task has a screen-based form
  - Fixed-font fixed format specified
- Fields to be filled in
  - Feedback from server on each
- Order and payment commit data
  - Status and reports have only read steps
- The data is randomised, steps too
  - Specified mix: of 23 options
    - 10 new orders, 10 payment, 1 each of others

Task descriptions contd

- Minimum time between tasks
  - 0.5 sec new order, 3 sec payment
  - 2 sec each for the others
- Minimum time to start a task
  - 15 sec for new order, 3 sec payment
  - 10 sec order status, 2 sec each for others

Specified conflicts include

- The DISTRICT table has a NEXT_O_ID
  - Updated at start of New order
- Both DISTRICT and WAREHOUSE have YTD (year-to-date) columns
  - Updated by committing Payment task
- STOCK table has S_QUANTITY for item
  - Updated by entering Order line
- CUSTOMER table has BALANCE
  - Updated by committing New order

Testing method

- The initial state of the database
  - As specificed by TPCC
  - Used for every test
- First Test: Run 2000 new orders
  - Measure elapsed time (62 sec vs 20 sec)
- Second Test: Run for 10 minutes
  - With different numbers of clerks
  - I used 1, 10, 20, 30,… until no progress

Record Requests

```
33179.8569;1;2;0; begin transaction
33198.8361;2;2;1; select W_NAME,W_STREET_1,W_STREET_2,W_CITY,W_STATE,W_ZIP,W_YTD from
33225.8214;3;2;1; select D_NAME,D_STREET_1,D_STREET_2,D_CITY,D_STATE,D_ZIP,D_YTD from
33251.6347;4;2;1; select C_ID from CUSTOMER where C_W_ID=1 and C_D_ID=2 and C_LAST='CA
33273.8684;5;2;1; update DISTRICT where D_W_ID=1 and D_ID=2 set D_YTD = 97272.30
33281.8606;6;2;1; update CUSTOMER where C_W_ID = 1 and C_D_ID = 2 and C_ID = -1 set C_
33282.8598;7;2;1; update WAREHOUSE where W_ID=1 set W_YTD = 1326455.19
33283.8586;8;2;1; commit
33288.8537;9;2;1;104720371;104720625
36852.3125;10;2;0; begin transaction
36854.2798;11;2;2; select D_TAX,D_NEXT_O_ID from DISTRICT where D_W_ID=1 and D_ID=3
36857.2765;12;2;2; select W_TAX from WAREHOUSE where W_ID=1
36859.2753;13;2;2; update DISTRICT  where D_W_ID=1 and D_ID=3 set D_NEXT_O_ID=3036
37365.9339;14;2;2; select C_DISCOUNT,C_LAST,C_CREDIT from CUSTOMER where C_W_ID=1 and
```

Setup for Other DBMS

- Out of the box
- Create Database Tpcc
- All calls to BeginTransaction have
  - IsolationLevel.Serializable
- No tuning or lock requests

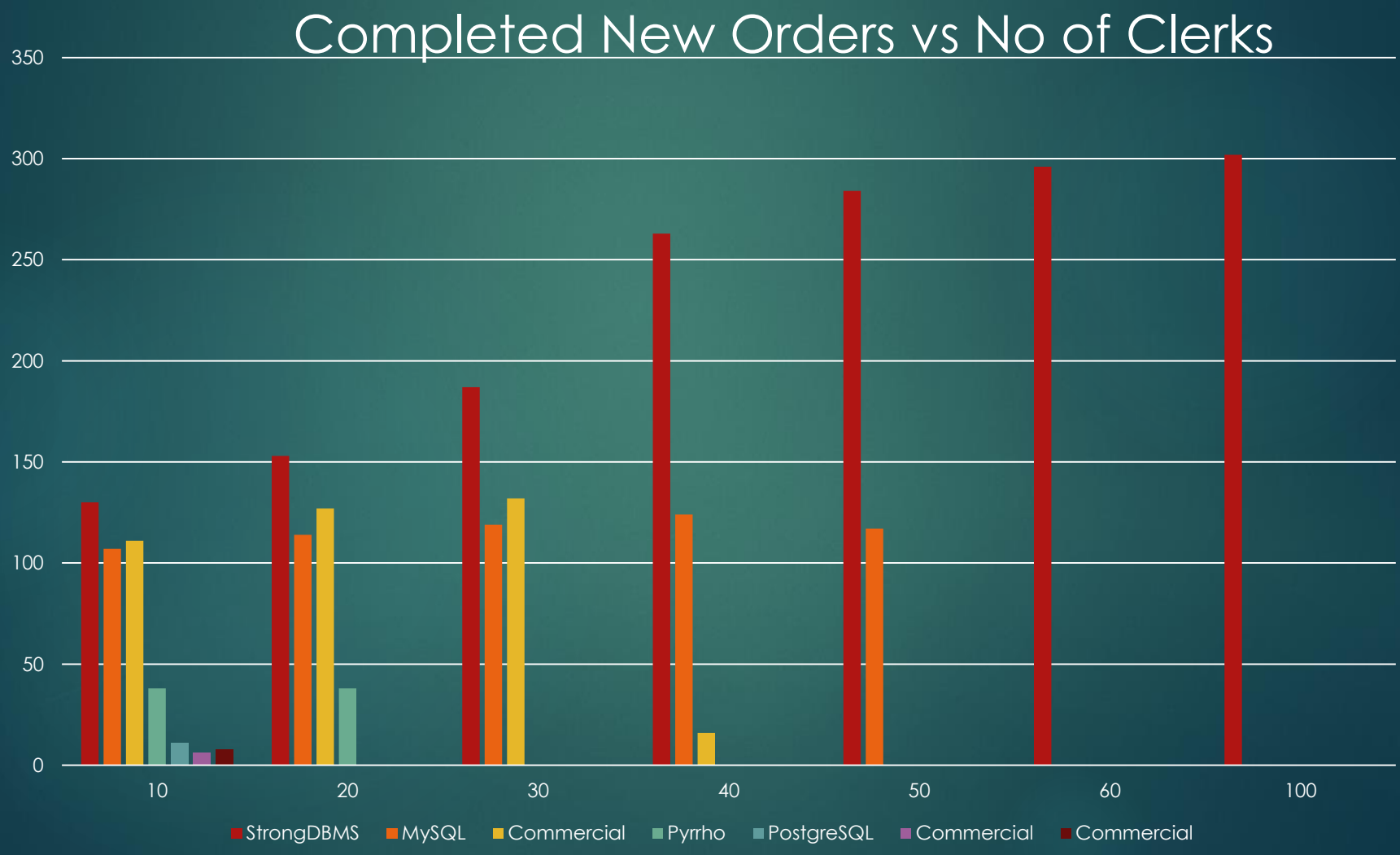
Results

Clerks	10	20	30	40	50	60	100
StrongDBMS	130	153	187	263	284	296	302
MySQL	107	114	119	124	117		
Commercial	111	127	132	16			
Pyrrho	38	38					
PostgreSQL	11						
Commercial	6						
Commercial	8						

Analysis of results

- First: StrongDBMS is slower than the commercial DBMS
  - 30% as fast in Test 1
- Second: StrongDBMS handles concurrency better
  - Assuming competitors use SERIALIZABLE
- For 1 clerk performance is identical

Outcomes: summary



Demos (Strong, Other)

- Building initial database (25 min,10 min)
- Cold Start (23 sec,-)
- 2000 Order transactions (3 min, 1 min)
- Simulated clerks (all 10 min)
  - Any number you like
- Other databases
  - Don't perform well with Serializable
  - PostgreSQL RepeatableRead 364 for 98 clerks

Questions?

- https://github.com/MalcolmCrowe/ShareableDataStructures
- Strongdbms.com
- Shareabledata.org
- @MalcolmCrowe
- #StrongDBMS
- #ShareableDataStructures