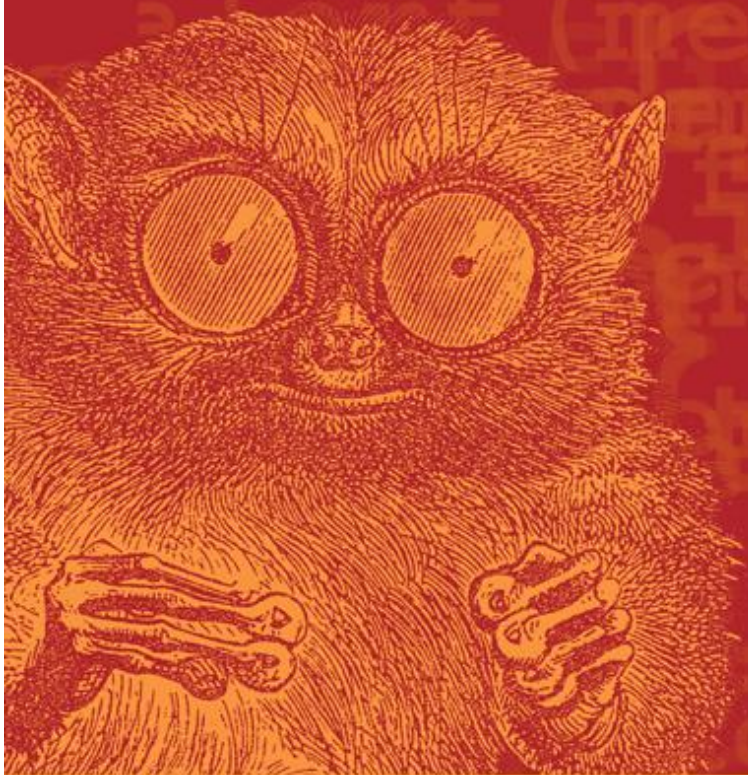


O'REILLY

OSCON[™]
Open Source Convention



Landscape of Open Source Transactional Storage Engines

Peter Zaitsev

Vadim Tkachenko

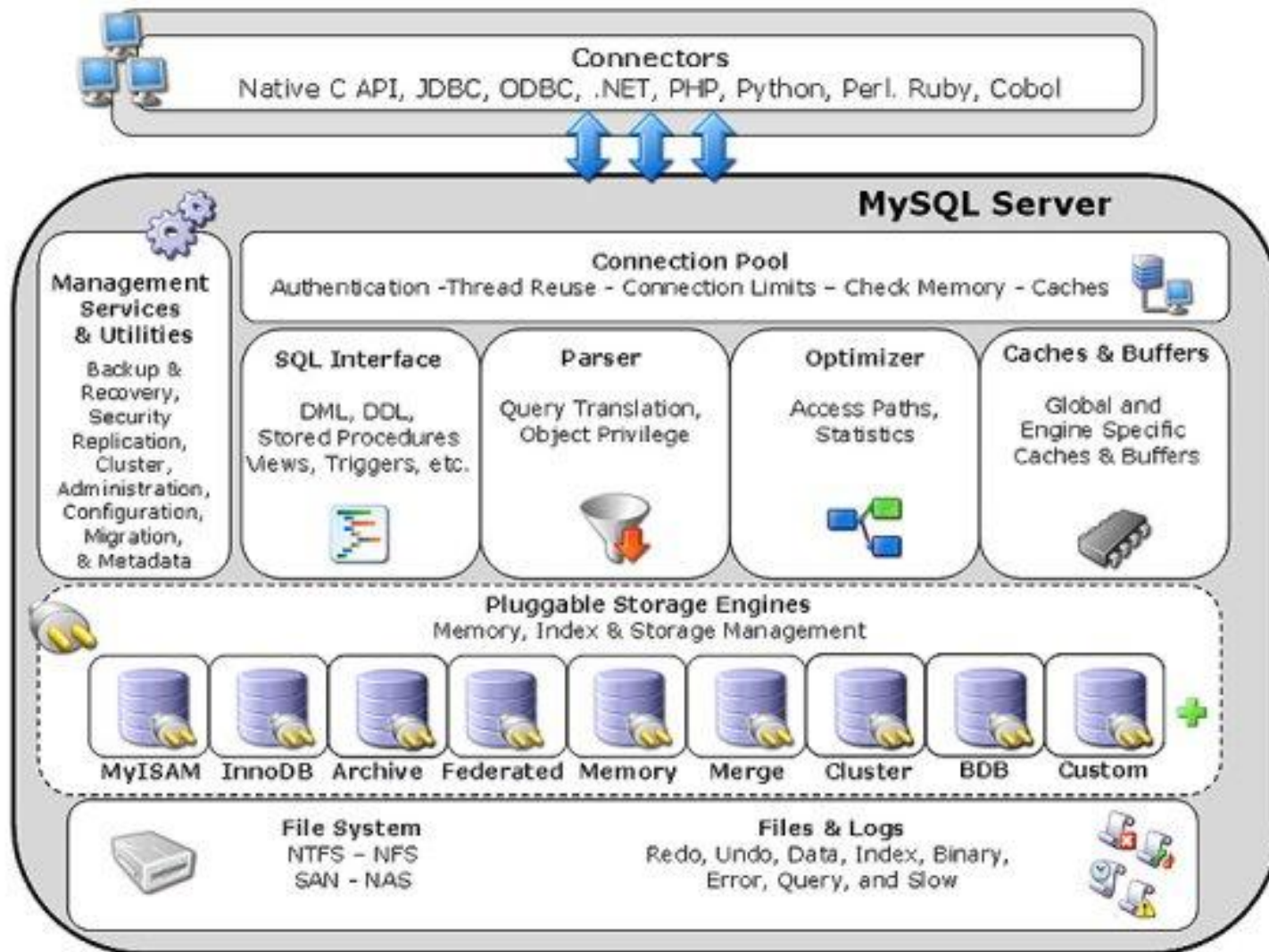
<http://MySQLPerformanceBlog.com>

About us

- Founders Percona Ltd
 - MySQL Performance Focused Consulting
 - <http://www.MySQLPerformanceBlog.com> - authors
- Worked for MySQL AB for years
 - Peter – lead of “High Performance Group”, Vadim his right hand
 - Long time MySQL users for bunch of personally involved projects



MySQL pluginable architecture



MySQL Transactional Engines

- **BDB** - Legacy Storage Engine, removed in 5.1 not tested
- **InnoDB** - “Most popular” (The only commonly used) storage engine by Innobase Oy.
- **SolidDB** - Storage Engine from Solid Information Technology
- **PBXT** - Storage Engine by SNAP Innovation (Paul McCullagh)
- **Falcon** - New Storage Engine by MySQL AB, Project lead by Jim Starkey
- **NDB** - MySQL Cluster is a whole other beast and not covered



InnoDB

- <http://www.innodb.com/>
- Mature Storage Engine, development started by Heikki Tuuri over 10 years ago.
- Heikki was looking for a ways to improve traditional databases performance
- Acquired by Oracle in the end of 2005
- The only Transactional storage engine available in MySQL 5.0 official release



solidDB

- <http://www.solidtech.com/solidDBforMySQL/>
- OpenSourced in 2006
- Existing Storage Engine technology “integrated” with MySQL
- Focused on reliability and Multiprocessor Scalability
- Currently shipped as production ready.



PrimeBase XT (PBXT)

- <http://www.primebase.com/xt/>
- Written mainly by Paul McCullagh since 2005
- Not a port of existing storage engine to MySQL but new writeup
- Uses number of unusual design decisions
- Only 50% transactional
- Focused on efficient BLOB storage
- <http://www.blobstreaming.org/>



Falcon

- <http://dev.mysql.com/doc/falcon/en/index.html>
- Based on “Netfrastructure” engine by Jim Starkey
- Purchased by MySQL AB in early 2006
- “Lightweight Design”
- Focused on Transactional needs of Web Application, efficient use of large amount of memory



O'REILLY

OSCON™
Open Source Convention

Design and Behavior



InnoDB design

- MVCC and very efficient row level locks
- Clustering by primary key, write to same pages
- non-compressed secondary indexes w. transaction info
- Single tablespace or tablespace per table
- Pessimistic locking
- Instant Deadlock detection
- Fuzzy Checkpointing
- “DoubleWrite” for partial page write protection



InnoDB

- DEADLOCK detection
- ```
Session 1: BEGIN;
Session 2: BEGIN;
Session 1: UPDATE test SET name='random1-1' WHERE id=1;
Session 2: UPDATE test SET name='random2-1' WHERE id=2;
Session 1: UPDATE test SET name='random1-2' WHERE id=2;
Session 2: UPDATE test SET name='random2-2' WHERE id=1;
```
- InnoDB detect deadlock (Error 1213) **Instantly** in second session
- Pessimistic locking:
- UPDATE the same row in two concurrent transaction – second transaction waits on COMMIT/ROLLBACK in first



# InnoDB Strengths

- Powerful MVCC
- Good performance on wide range of workloads
- Great Stability
- Great Data Protection
- Primary Key Clustering allows a lot of optimizations
- Transaction info in secondary indexes allow fast index only scans
- Adaptive Hash indexes and other advanced techniques



# InnoDB Weaknesses

- Slow Development pace in recent years
- Still having scalability issues with multiple CPUs
- Unscalable Auto-Increment, Broken Group Commit take very long to fix
- Large footprint, especially for secondary indexes
  - It turns out not so large as we compare
- Still messy integration with MySQL
  - How do you see how much space is free in InnoDB tablespace ?



# SolidDB Design

- MVCC and Row level locking
- Clustering by Primary Key
- New data stored in new pages
- “Bonsai Tree” used for Multi Versioning
- OPTIMISTIC and PESSIMISTIC locking specified on table level
- Online Backup (Not usable for Slave creation)
- High Available sync replication promised soon.



# solidDB - PESSIMISTIC

- DEADLOCK - DEADLOCK detected in first Session after 20 sec of waiting
  - Timeout based deadlocks
- UPDATE two rows – second session wait on first



# solidDB - OPTIMISTIC

- DEADLOCK - DEADLOCK detected in second Session immediately but with error 1205 – Lock wait timeout exceeded
- UPDATE two concurrent rows:
- SESSION 1: BEGIN;  
SESSION 2: BEGIN;  
SESSION 1: UPDATE test SET name = 'rnd' WHERE id=2;  
SESSION 2: UPDATE test SET name = 'rnd' WHERE id=2;
- In Session 2 we got:  
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
- This is OK for OPTIMISTIC engines, but may cause trouble in Web applications.



## SolidDB Strengths and Weakness

- Limited production usage to really tell
- Out of storage engines reviewed most similar in design to Innodb
- Choice of Optimistic vs Pessimistic is nice for some applications
- No instant deadlock detection
- So far available as special download only (not even a plugin)



# PBXT Design

- MVCC With row level locking
- “Per Database” Transactions
- No real durability yet, weak crash recovery
- OPTIMISTIC locking
- Write once, write sequentially to log
- Never update in place
- Data cache + Key cache
- Efficient BLOB Handling



# PBXT

- DEADLOCK detected in second session, 1213 error
- UPDATE two concurrent rows – optimistic,  
second session:  
ERROR 1020 (HY000): Record has changed since last  
read in table 'test2'



# PBXT Strengths and Weaknesses

- Not yet commonly used in production (we tried but got too many bugs)
- Very good performance for some workloads
- Efficient Storage, close to MyISAM
- Focused on BLOB efficient handling, extra features like Blob Streaming
- Still mainly one man project
- Large ToDo, a lot needs to be done, including Recovery
- Potentially large Purging overhead



# Falcon Design

- MVCC, row level locking (in practice, not in theory)
- PESSIMISTIC locking
- Not clustered by primary key
- Row cache (cache only rows you need)
- “Optimal” index traversal
- “Data Compression” - Nulls, Empty Strings
- Always needs to read row data (because of index structure)



# Falcon

- DEADLOCK:

In Session2:

ERROR 1020 (HY000): Record has changed since last read in table 'test2'

- Ann Harrison tells Falcon checks cycles in lock graph periodically rather than instantly on row lock wait

- UPDATE:

Second session waits



# Falcon Strengths in Weaknesses

- Still Alpha with many bugs – Early to judge
- Very active support from MySQL AB
- Fast development pace – bugs being fixed quickly, major performance improvements during last 3 months
- Good integration with MySQL, ie tables for performance data
- No Primary key clustering or covering index support
- Different design decisions can complicate migration from Innodb (though logical behavior became closer)



O'REILLY

OSCON™  
Open Source Convention

There are lies, big lies  
and there are  
**Benchmarks**



# Benchmarks – things to note

- Benchmarks may not be relevant for performance of your application
- Early versions we tried for Falcon, PBXT may change their performance properties before production
- There is not too much experience out where tuning Falcon, PBXT and Solid with MySQL as they are barely used in production
- We did less benchmarks than wanted – spent a lot of time fighting/reporting bugs and checking fixes



# Benchmarks

- Read-Only on typical table for web-application
- DBT2 – TPC-C emulation
- Dell DVD Store – emulation of e-commerce site
- Sysbench – OLTP transactions
- Sqlbench - small data set, single user, typical query patterns



# Box

- Dell PowerEdge 2950
- CentOS release 4.5
- 4 CPU
  - model name : Intel(R) Xeon(R) CPU 5148 @  
2.33GHz
  - stepping : 6
  - cpu MHz : 2327.529
  - cache size : 4096 KB
- 16 GB of RAM
- RAID 10 (6 10K RPM 3.5" SAS hard drives)



# MySQL Versions

- Yes, this means version affects performance not only storage engine but we could not get all storage engine working with same MySQL version.
- InnoDB and PBXT  
5.1.19
- Falcon  
6.0.1-alpha, bk tree from 10-Jul
- SolidDB  
5.0.41-0073



# Engines parameters

- 12 GB of RAM for buffers
- InnoDB **--innodb\_buffer\_pool\_size=12G**  
**--innodb\_flush\_method=O\_DIRECT**  
**--innodb-log-file-size=100M**
- SolidDB **--soliddb-cache-size=12G**
- Falcon  
**--falcon\_min\_record\_memory=2G**  
**--falcon\_max\_record\_memory=4G**  
**--falcon\_page\_cache\_size=8G**
- PBXT  
**pbxt\_index\_cache\_size=8G**  
**pbxt\_record\_cache\_size=4G**

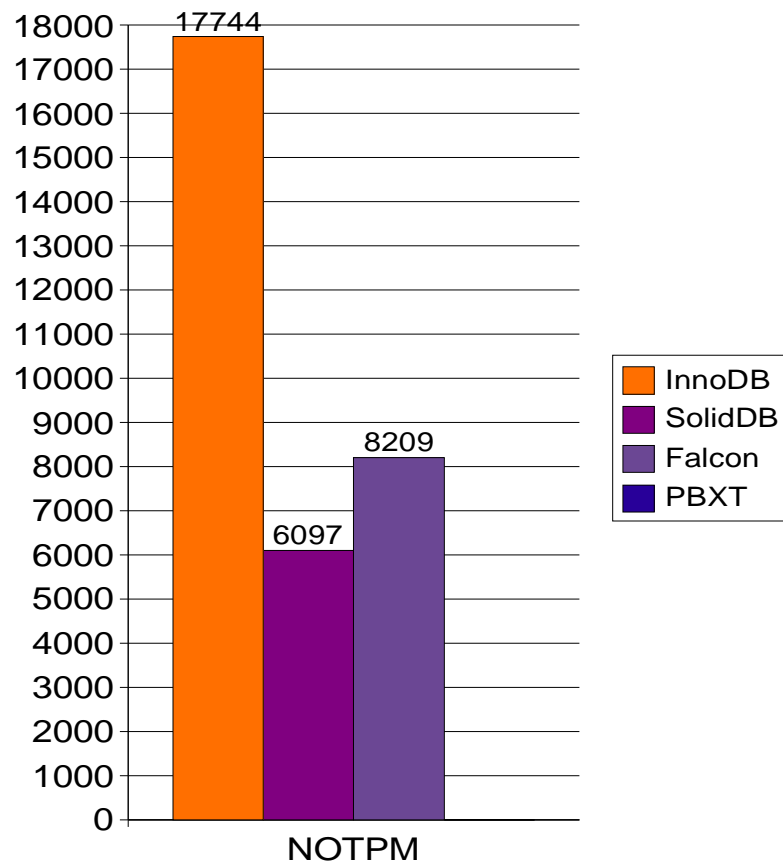


# DBT2 Configuration Details

- DBT2
  - <http://osdl/dbt.sourceforge.net/>
  - 10 Concurrent users (about 2 for each CPU core and disk)
  - “Zero Delay” to fully load MySQL Server
  - In 400W configuration reduced available memory to 4G by locking 12GB of memory to have it IO bound.
    - Buffer sizes were reduced to 2GB



# DBT2 – 10 warehouses



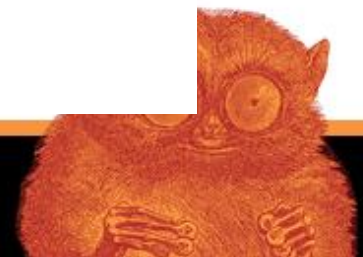
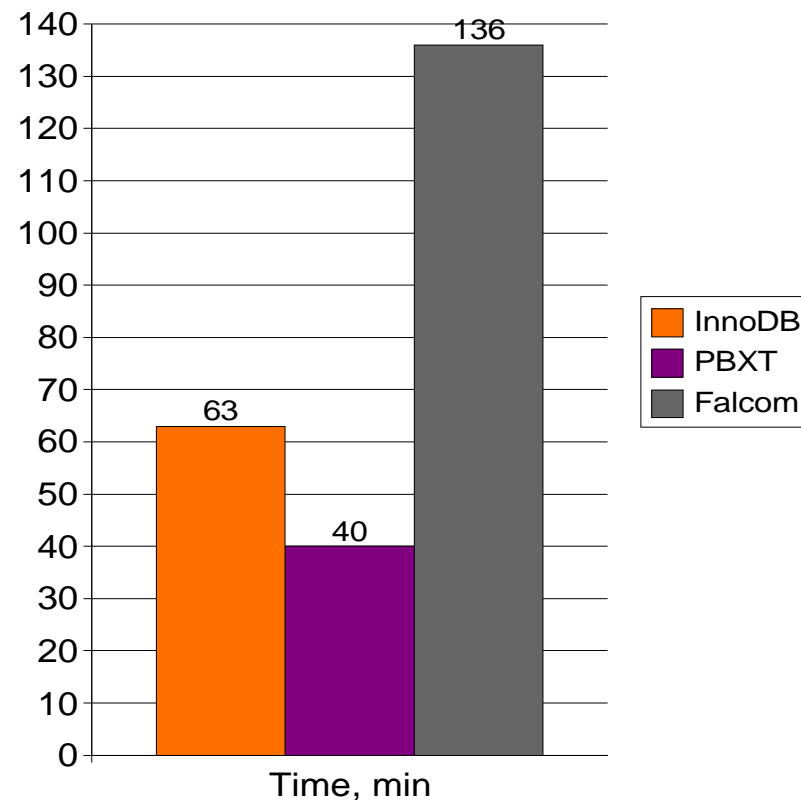
- 10 warehouses, 10 clients (datasize ~ 700M)
- Result in New Order Transaction Per Minute, more is better
- PBXT crashed
- Old version of Falcon had ~1100 NOTPM
- Great improvement !



# DBT2 – 400 warehouses

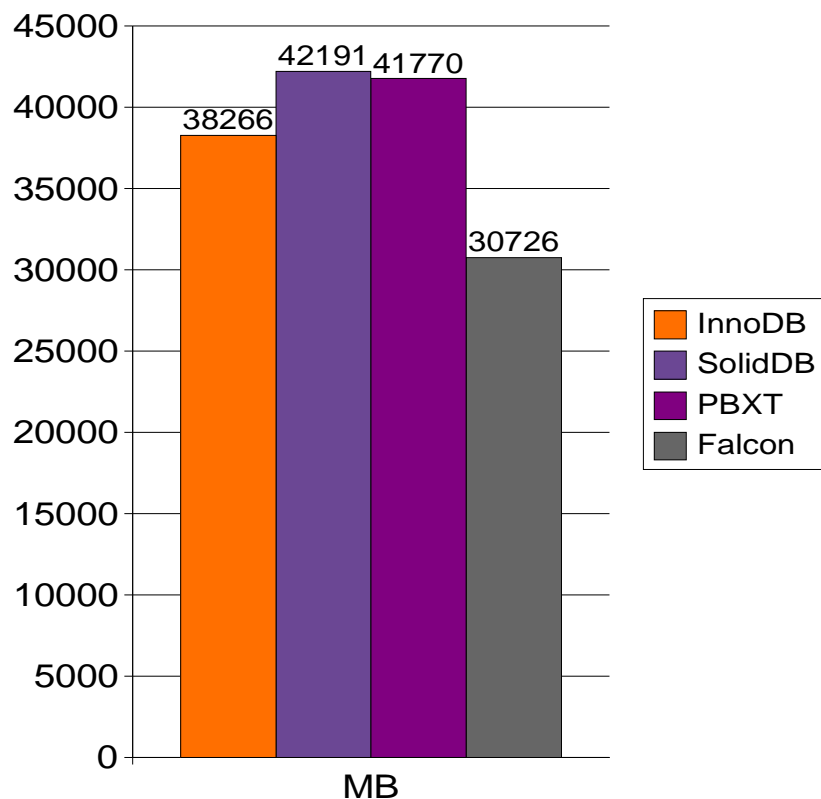
- Data size ~ 29GB
- SolidDB crashed after 336 mins
- Did Not disable logs on SolidDB to have things comparable.

Load time



# DBT2, 400W, Data size

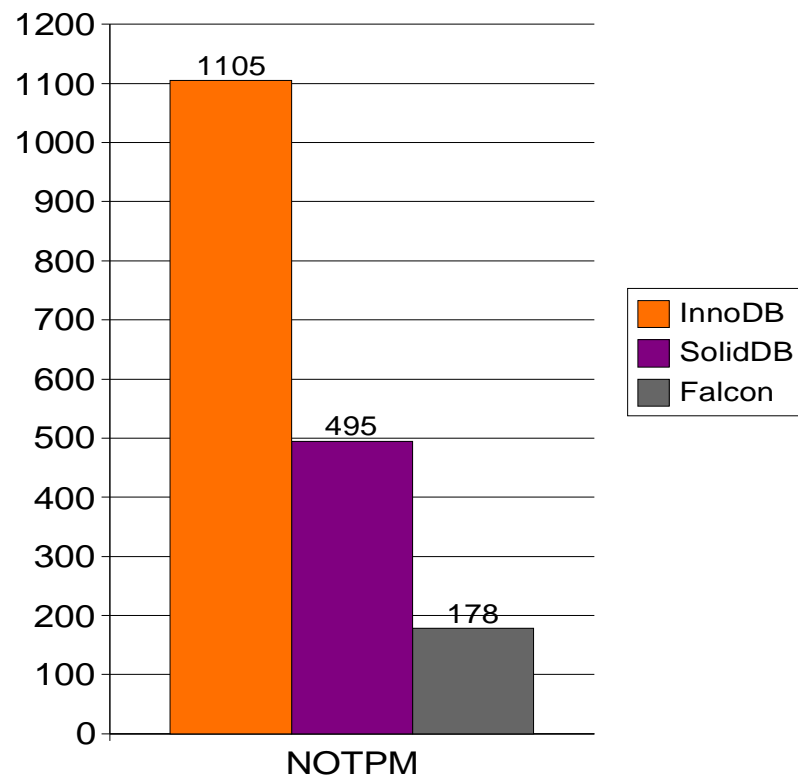
Size of loaded data



- Surprisingly large size from PBXT
- SolidDB – tables were loaded into MyISAM and then converted to SolidDB
- It was crashing otherwise



# DBT2, 400W, Results

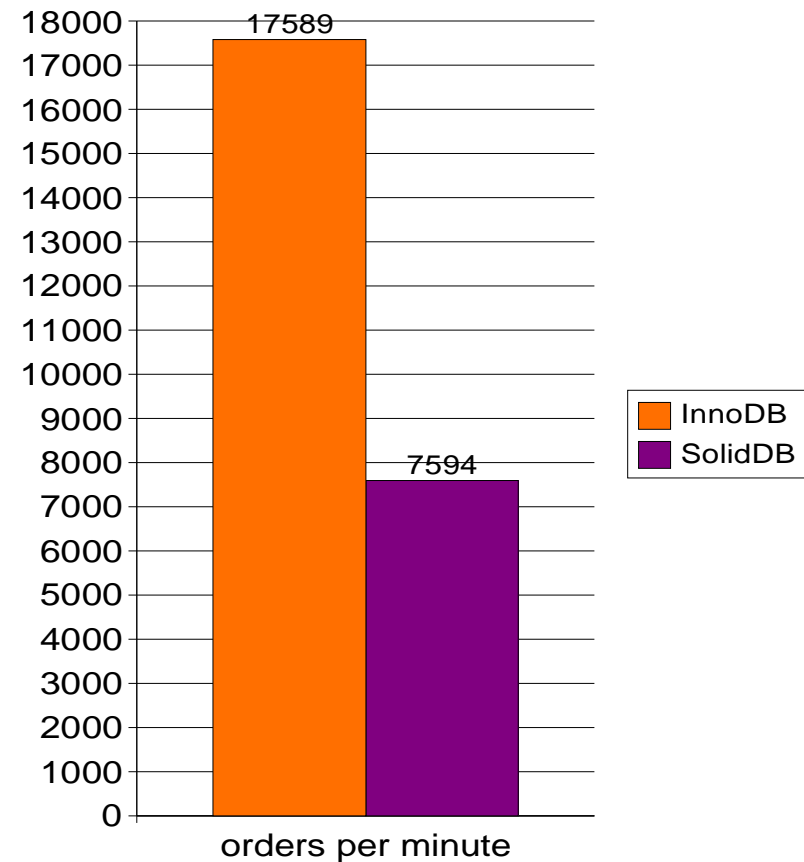


- PBXT crashed
- Result in New Order Transaction Per Minute, more is better



# Dell DVD Store

- Datasize  
Medium 1 GB  
2,000,000 Customers  
100,000 Products
- Falcon – crashed
- PBXT – a lot of errors
- Result in New Orders  
per minute, more is  
better



# sysbench

- Older Falcon used in this test. New one crashes :(
- Couple of READ-ONLY queries against typical table for Web-applications – info of user account:

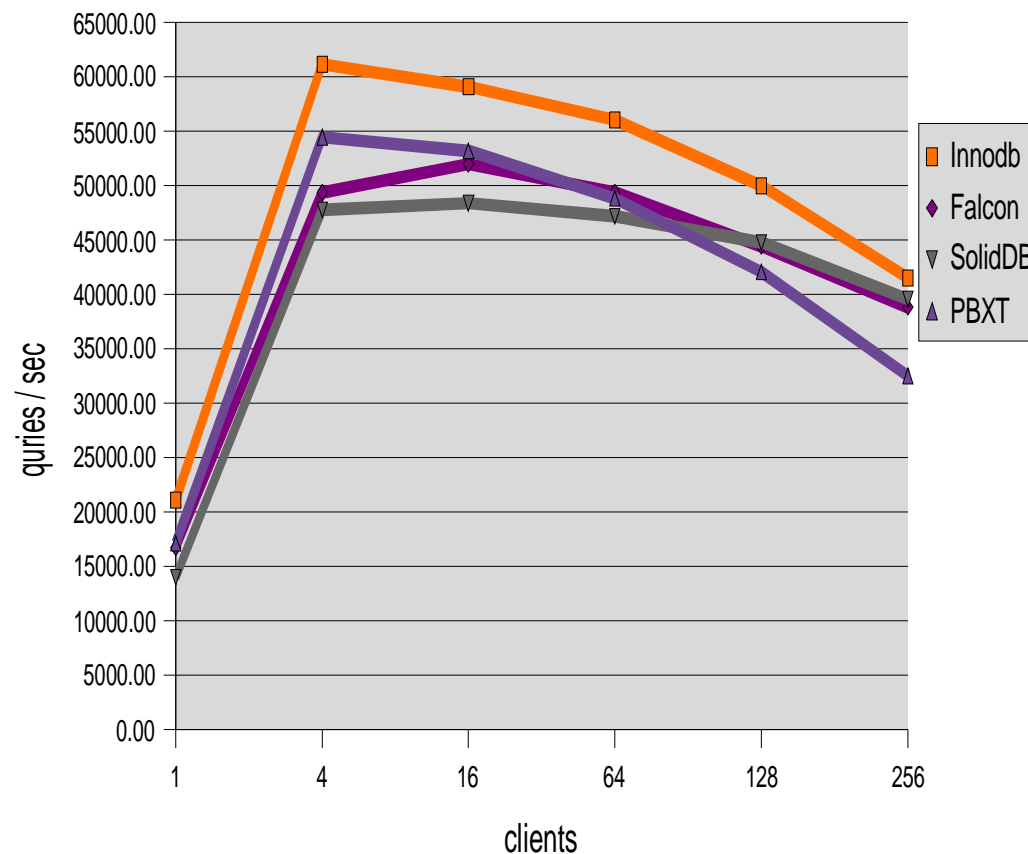
```
CREATE TABLE IF NOT EXISTS sbtest (
 id int(10) unsigned NOT NULL auto_increment,
 name varchar(64) NOT NULL default '',
 email varchar(64) NOT NULL default '',
 password varchar(64) NOT NULL default '',
 dob date default NULL,
 address varchar(128) NOT NULL default '',
 city varchar(64) NOT NULL default '',
 state_id tinyint(3) unsigned NOT NULL default '0',
 zip varchar(8) NOT NULL default '',
 country_id smallint(5) unsigned NOT NULL default '0',
 PRIMARY KEY (id),
 KEY `country_id` (country_id,state_id,city)
```



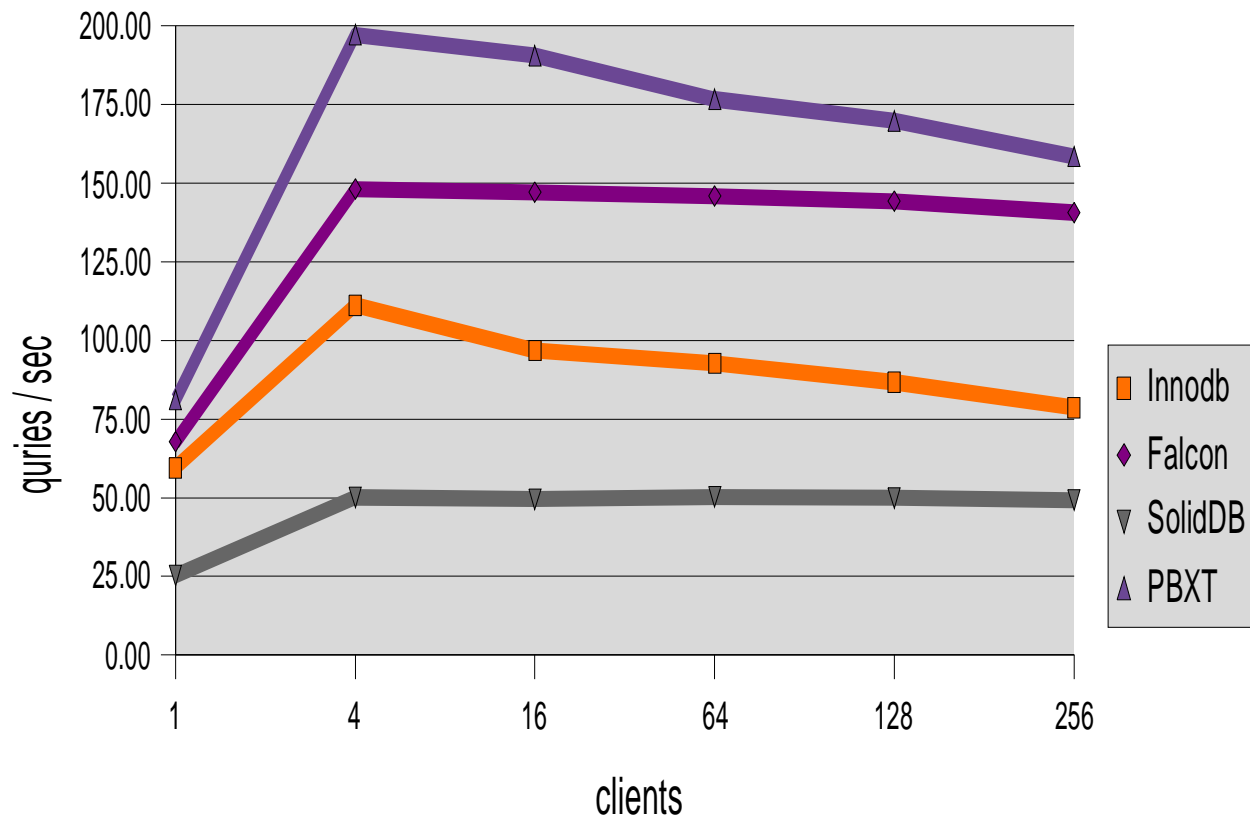
# sysbench, read by primary key

```
·SELECT name
FROM sbtest
WHERE id=?
```

```
·InnoDB and
Solid have
sweet spot
being
clustered by
PK
```



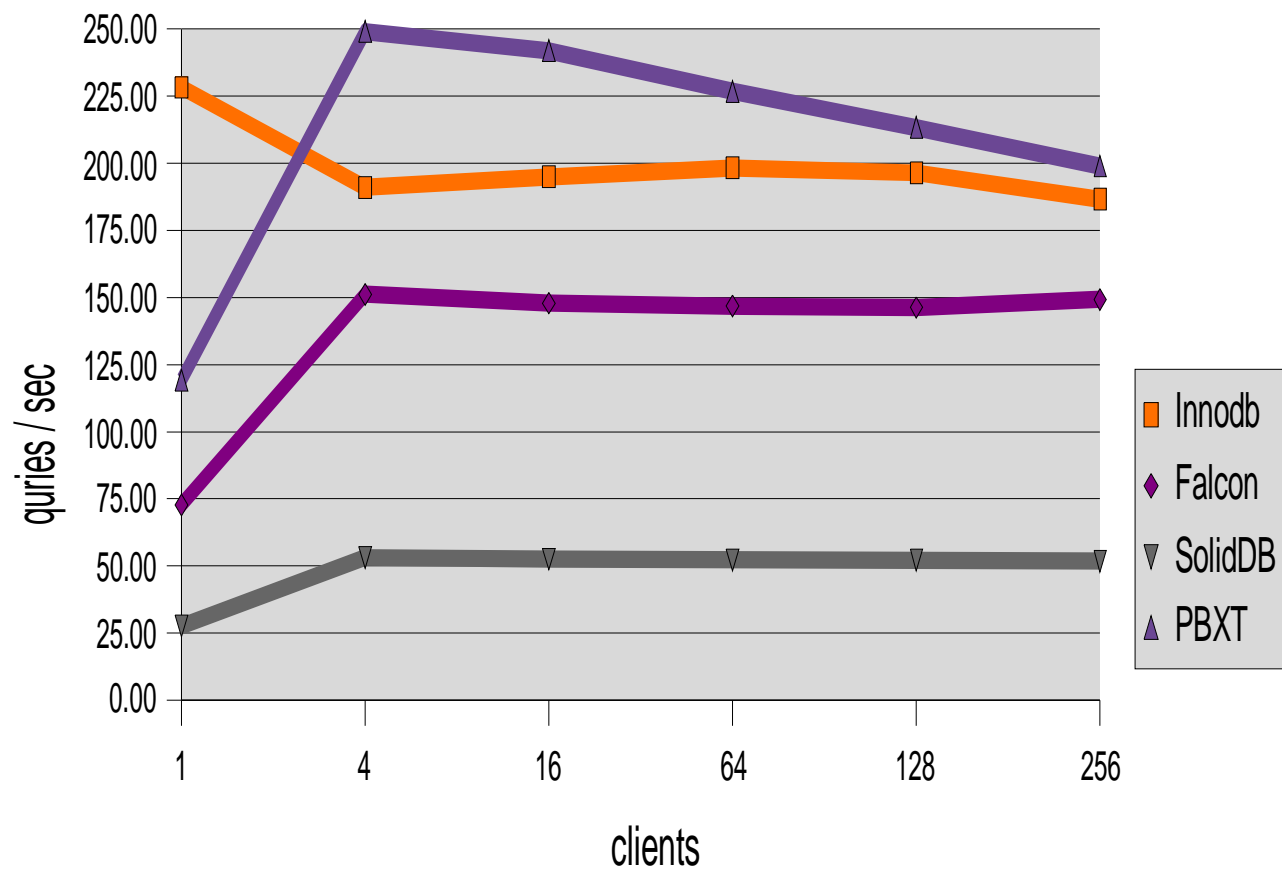
# sysbench, read by index



- `SELECT name FROM sbtest WHERE country_id=?`
- PBXT Excels
- Falcon comes next



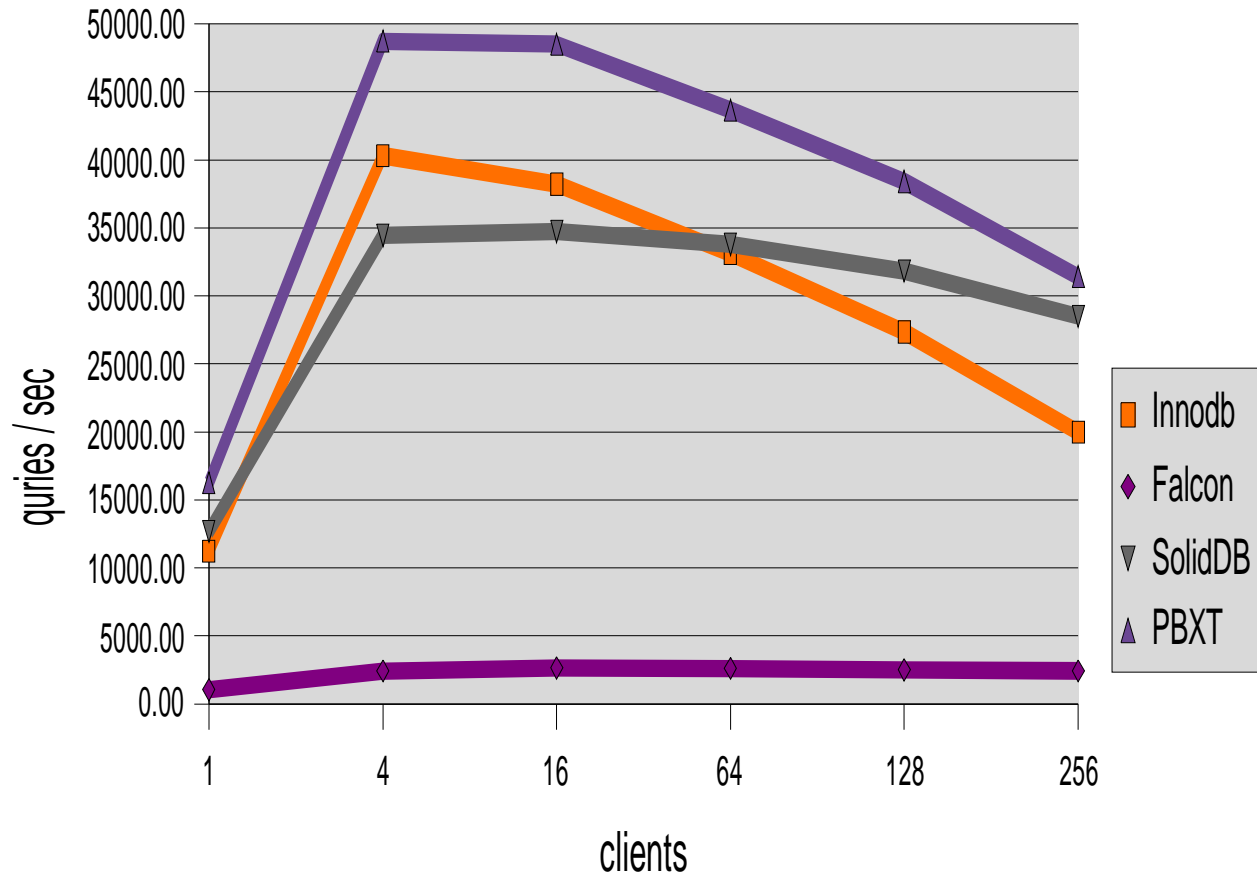
# sysbench, read by covered index



- `SELECT state_id FROM sbtest WHERE country_id=?`
- PBXT still best
- Falcon can't use covered index



# sysbench, read by index, LIMIT 20



```
•SELECT name
FROM sbtest
WHERE
country_id=?
LIMIT 20
```

•Falcon Does  
not optimize  
Limit

•Innodb  
Scales  
poorly



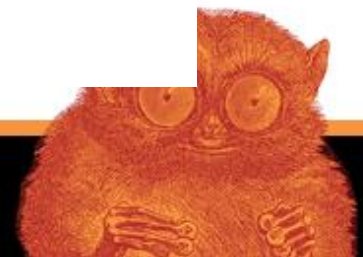
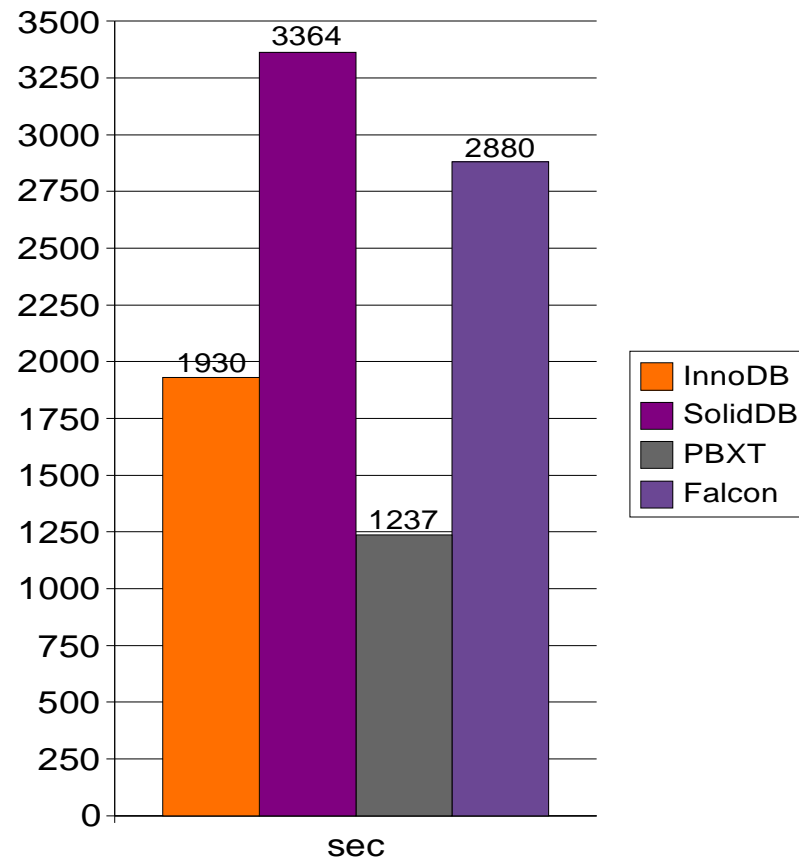
# Sysbench OLTP

- Datasize  
100,000,000 rows  
~25GB
- Uniform distribution
- I/O-bound load
- read / write transactions
- Reduced available memory by locking 12GB out of 16GB

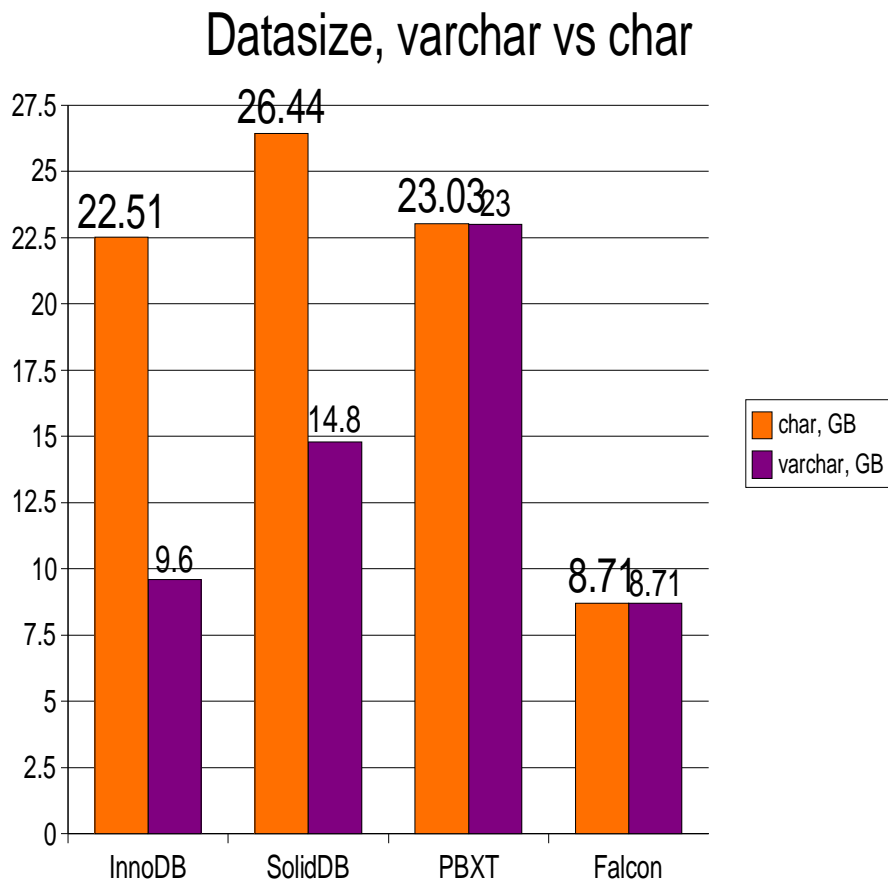


# Sysbench OLTP, time to load data

- Using multi-value INSERTs rather than LOAD DATA INFILE
- Solid and Falcon are even slower than Innodb which is known to be slow compared to MyISAM for data load.



# Sysbench OLTP, Datasize

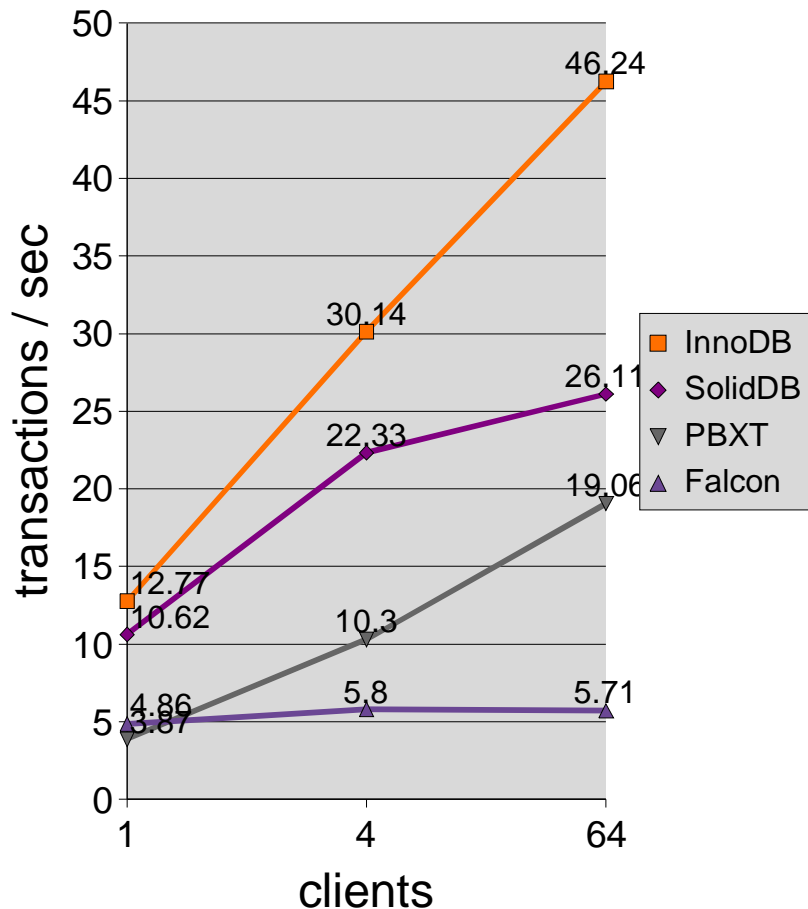


- Comparison of storages of char and varchar columns in the table
- Falcon uses dynamic length rows anyway
- PBXT surprisingly has same huge size in both cases



# Sysbench OLTP, results

I/O bound



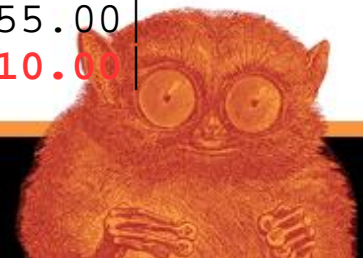
- Memory limited to 4GB, 2GB for buffers
- InnoDB and SolidDB have benefit due to clustering by primary key
- All but Falcon scale well for IO bound workload with this amount of hard drives.



# Selected sqlbench results

- single operation repeated N times, total time in secs. less is better

| Operation                             | 1       | 2       | 3       |
|---------------------------------------|---------|---------|---------|
|                                       | innodb_ | pbxt_fa | soliddb |
| alter_table_add (100)                 | 8.00    | 3.00    | 32.00   |
| count (100)                           | 12.00   | 8.00    | 28.00   |
| count_distinct (1000)                 | 6.00    | 8.00    | 74.00   |
| count_distinct_2 (1000)               | 11.00   | 11.00   | 16.00   |
| count_group_on_key_parts (1000)       | 7.00    | 10.00   | 83.00   |
| count_on_key (50100)                  | 70.00   | 94.00   | 210.00  |
| delete_all_many_keys (1)              | 17.00   | 2.00    | 28.00   |
| insert (350768)                       | 6.00    | 5.00    | 21.00   |
| outer_join (10)                       | 14.00   | 7.00    | 61.00   |
| select_key2_return_prim (200000)      | 30.00   | 29.00   | 25.00   |
| select_many_fields (2000)             | 8.00    | 6.00    | 5.00    |
| update_big (10)                       | 18.00   | 56.00   | 727.00  |
| update_of_key_big (501)               | 19.00   | 6.00    | 165.00  |
| update_of_primary_key_many_keys (256) | 44.00   | 17.00   | 55.00   |
| update_with_key_prefix (100000)       | 19.00   | 8.00    | 10.00   |



# Conclusion

- All reviewed storage engines but InnoDB are currently too unstable for production use. SolidDB comes closest.
- InnoDB is still winner in majority of tests
- Falcon has some issues with LIMIT optimization and IO bound scalability
- PBXT and Falcon win in certain tests
- SolidDB is currently an outsider in terms of Performance
- Need to revisit when production versions of all storage engines are ready.



# The End

- Thanks for coming !
- Slides will be published at <http://www.mysqlperformanceblog.com/>
- Feel free to approach us with your question
- MySQL Performance Optimization Consulting Available
- <http://www.mysqlperformanceblog.com/mysql-consulting/>

