ISEngineering

Technische Universität Berlin

Enterprise Computing: Exercise 6 – Benchmarking

Marco Peise, Stefan Tai

ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

Agenda

Exercise 6 Benchmarking

- Cassandra
- MongoDB

Exercise 6

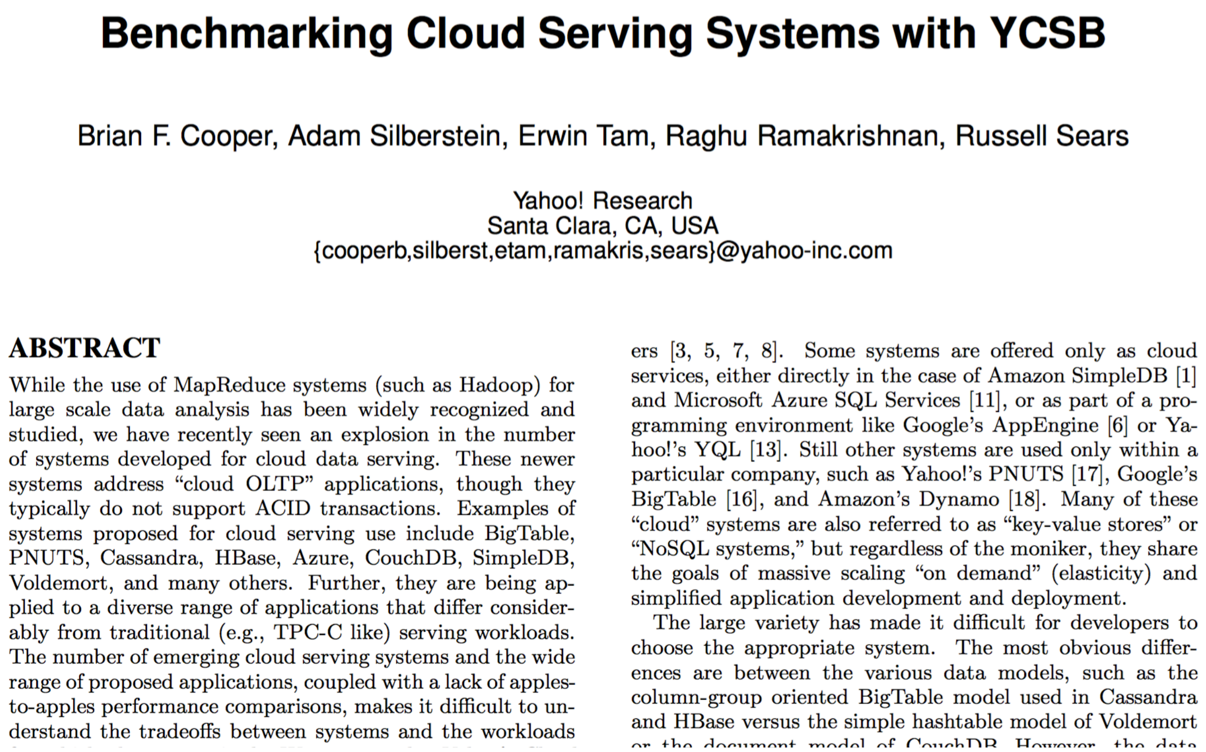
Task 1

Name four relevant reasons why transactional web benchmarks are not optimal for benchmarking cloud database systems or cloud database services.

.

ISEngineering
Wirtschaftsinformatik –
Information Systems Engineering

YCSB



Benchmarking Cloud Serving Systems with YCSB

Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, Russell Sears

Yahoo! Research
Santa Clara, CA, USA
{cooperb,silberst,etam,ramakris,sears}@yahoo-inc.com

ABSTRACT

While the use of MapReduce systems (such as Hadoop) for large scale data analysis has been widely recognized and studied, we have recently seen an explosion in the number of systems developed for cloud data serving. These newer systems address "cloud OLTP" applications, though they typically do not support ACID transactions. Examples of systems proposed for cloud serving use include BigTable, PNUTS, Cassandra, HBase, Azure, CouchDB, SimpleDB, Voldemort, and many others. Further, they are being applied to a diverse range of applications that differ considerably from traditional (e.g., TPC-C like) serving workloads. The number of emerging cloud serving systems and the wide range of proposed applications, coupled with a lack of apples-to-apples performance comparisons, makes it difficult to understand the tradeoffs between systems and the workloads

ers [3, 5, 7, 8]. Some systems are offered only as cloud services, either directly in the case of Amazon SimpleDB [1] and Microsoft Azure SQL Services [11], or as part of a programming environment like Google's AppEngine [6] or Yahoo!'s YQL [13]. Still other systems are used only within a particular company, such as Yahoo!'s PNUTS [17], Google's BigTable [16], and Amazon's Dynamo [18]. Many of these "cloud" systems are also referred to as "key-value stores" or "NoSQL systems," but regardless of the moniker, they share the goals of massive scaling "on demand" (elasticity) and simplified application development and deployment.

The large variety has made it difficult for developers to choose the appropriate system. The most obvious differences are between the various data models, such as the column-group oriented BigTable model used in Cassandra and HBase versus the simple hashtable model of Voldemort

References:
- http://wiki.github.com/brianfrankcooper/YCSB/
- B.F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, R. Sears "Benchmarking cloud serving systems with YCSB.", ACM SOCC, 2010

Task 2 Perform YCSB (Cassandra & MongoDB)

Install YCSB, the YCSB Cassandra & MongoDB bindings and perform the following tasks against a local Cassandra server (1-node cluster) and a local MongoDB server:

Cassandra: Load the Cassandra table “usertable” in keyspace “ycsb” with 1 million records of default record size (i.e., 10 fields à 100 Bytes = 1 KB).

Load a “transaction” performance benchmark with workload “workloada” (50% read and 50% write operations) with 1 million operations and 10 Threads on your local machine. Use “recordcount=1000000” as one workload property.

Run with it with “operationcount=1000000” as one workload property.

Task 2 – Install YCSB

# download YCSB 0.11.0

https://github.com/brianfrankcooper/YCSB/releases/download/0.11.0/ycsb-0.11.0.tar.gz

# download Cassandra 3.9

http://cassandra.apache.org/download/

# download MongoDB 3.4.0

https://www.mongodb.com/download-center#community

Task 2 – Cassandra Configuration
Create Keyspace and Table

```
cat >create_usertable <<END_OF_FILE
create keyspace ycsb WITH REPLICATION = {'class' :
     'SimpleStrategy', 'replication_factor': 1 };
USE ycsb;
create table usertable (y_id varchar primary key,
    field0 varchar, field1 varchar, field2 varchar, field3 varchar,
    field4 varchar, field5 varchar, field6 varchar, field7 varchar,
    field8 varchar, field9 varchar);
END_OF_FILE
```

```
Cqlsh -f create_usertable
```

Task 2 – Load and Run!

```
$ ./ycsb-0.11.0/bin/ycsb load cassandra-2-cql …

$ ./ycsb-0.11.0/bin/ycsb run cassandra-2-cql …

$ ./ycsb-0.11.0/bin/ycsb load mongodb ...

$ ./ycsb-0.11.0/bin/ycsb run mongodb ...
```

Task 2 a)

a) What are your results of the “transaction” performance benchmark for Cassandra?

What happens if you change read or write consistency levels?

YCSB Metric	Results
Average READ Latency [µs]	
Average UPDATE Latency [µs]	
READ MinLatency [µs]	
Throughput [ops/sec]	

Task 2 b)

b) What are your results of the “transaction” performance benchmark for MongoDB?

YCSB Metric	Results
Average READ Latency [µs]	
Average UPDATE Latency [µs]	
READ MinLatency [µs]	
Throughput [ops/sec]	

Task 2 c)

c) Use Cassandra's nodetool utility to collect the following metrics after your benchmark run:

Cassandra Metric	Results
Local read count	
Local read latency	
Local write count	
Local write latency	

Task 2 d)

d) Explain the read count and write count numbers that you collected in c). Do the numbers make sense?

Task 2 e)

e) What equivalent **command in MongoDB** to Cassandra's nodetool utility could you use to collect the following metrics after your benchmark run:

YCSB Metric	Results
Average Object Size	
Number of all Documents in DB	

Task 2 f)

f) When switching your MongoDB engine from MMAPv1 to WiredTiger Engine and run your benchmark again, how many ”Insert Calls” is YCSB performing? Use the same tool as in 2 e).

Does the number for “Insert Calls” make sense? Explain why!

YCSB Metric	Results
Insert Calls	

Task 2 f) switching from WiredTiger engine to MMAPV1

https://docs.mongodb.com/manual/core/mmapv1/

1.
2.
3.
4.
5.
6.

Task 2 g)

g) What are your results of the “transaction” performance benchmark for MongoDB with MMAPv1 activated?

Explain the difference between “MMAPv1” and “WiredTiger” results.

YCSB Metric	Results
Average READ Latency [µs]	
Average UPDATE Latency [µs]	
READ MinLatency [µs]	
Throughput [ops/sec]	