Methods of benchmarking NoSQL database systems

Ilya Bakulin
webmaster@kibab.com, kibab@FreeBSD.org

SMS Traffic

LVEE 2011
1. Introduction

2. YCSB benchmarking framework

3. YCSB practical usage

4. Results

5. Where to find further information
Why benchmarking NoSQL is necessary

- No guides / FAQs about performance are generally available, or are outdated
- NoSQL systems are actively developed
- Nobody wants to end up with crashed DB in production right before 2-week vacation
Why benchmarking NoSQL is complex

- RDBMS use SQL to provide access to data stored in them, while NoSQL systems don’t.
- Each NoSQL uses different protocol (Thrift, Memcached-style, own protocols).
- Existing benchmarks require SQL to work with database under inspection.
What is YCSB?

- YCSB stands for Yahoo Cloud Serving Benchmark
- Developed by Yahoo! Research group
- Open Source project, hosted on GitHub (178 watchers, 42 forks)
Java application

Shipped with ready-to-use adapters for several popular OpenSource databases

Command-line parameters:
- DB to use
- Target throughput
- Number of threads
- ...

Workload parameter file:
- R/W mix
- Record size
- Data set
- ...

Extensible: define new workloads

Extensible: plug in new clients

YCSB client

DB client

Workload executor

Stats

Client threads

Cloud DB
More on DB interface

- Simple operations: INSERT, UPDATE, REPLACE, DELETE, SCAN
- Does not use SQL
- ... but SQL support is available through contributed JDBC driver
- ... Even sharding configurations are possible

Command-line parameters
- DB to use
- Target throughput
- Number of threads
- ...

Workload parameter file
- R/W mix
- Record size
- Data set
- ...

Extensible: define new workloads
Extensible: plug in new clients

YCSB client

DB client

Stats

Client threads

Workload executor

Cloud DB

Ilya Bakulin (SMS Traffic)
More on DB interface

- Simple operations: INSERT, UPDATE, REPLACE, DELETE, SCAN
- Does not use SQL
- ... but SQL support is available through contributed JDBC driver
- ... Even sharding configurations are possible

![Diagram](Image)

- Workload parameter file
  - R/W mix
  - Record size
  - Data set
  - ...

- Command-line parameters
  - DB to use
  - Target throughput
  - Number of threads
  - ...

- YCSB client
  - Workload executor
  - Stats

- DB client

- Cloud DB
  - Extensible: plug in new clients
  - Extensible: define new workloads

YCSB client

DB client

Cloud DB

Extensible: plug in new clients
Extensible: define new workloads
More on DB interface

- Simple operations: INSERT, UPDATE, REPLACE, DELETE, SCAN
- Does not use SQL
- ... but SQL support is available through contributed JDBC driver
- ... Even sharding configurations are possible

![Diagram of YCSB client and Cloud DB with extensible components]

- Command-line parameters:
  - DB to use
  - Target throughput
  - Number of threads
  - ...

- Workload parameter file:
  - R/W mix
  - Record size
  - Data set
  - ...

- Extensible: define new workloads
- Extensible: plug in new clients
More on workload

- Specifies what DB operations are used by application
- Also defines request distribution
- It is possible to specify record size
- It’s possible to specify number of records and operations

![Graphs showing popularity distributions: Uniform, Zipfian, Latest]

- Uniform: Choice of records is completely random.
- Zipfian: Recent records have a high probability of being chosen, while older records have a low probability.
- Latest: The most recently inserted records are the most popular, while previously popular items become less popular over time.

The workload client must make many random choices when generating load: which operation to perform (Insert, Update, Read or Scan), which record to read or write, how often to perform it, and when to perform it (within a scan interval). The number of records to scan is determined randomly, allowing us to control the size of these intervals.
SMS service provider, several gateways, big clients (such as banks)

- 15% inserts, 65% updates, 15% reads
- Request distribution: latest SMS messages are the "hottest" ones
- Evaluated Cassandra and sharded MySQL as DB storage for the next generation of SMS sending platform
Testing process

- 3 instances of DBMS system on one server (Core Quad Q9400, 4GB RAM, SATA-II HDD, FreeBSD 8.2-amd64)
- Cassandra 0.7.4 (1GB Java heap / instance)
- MySQL 5.1 + InnoDB engine (1GB InnoDB buffer pool size / instance)
- Client: separate machine, 1Gb/s connection

Should avoid swapping and disk IO saturation
Some results: Workload "A": 50% read / 50% write

Workload A – Update heavy
• 50/50 Read/update

Comment: Cassandra is optimized for writes, and achieves higher throughput and lower latency. Sherpa and MySQL achieve roughly comparable performance, as both are limited by MySQL's capabilities.

HBase has good write latency, because of commits to memory, and somewhat higher read latency, because of the need to reconstruct records.

---

### Workload A - Read latency

<table>
<thead>
<tr>
<th>System</th>
<th>Average read latency (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cassandra</td>
<td>10</td>
</tr>
<tr>
<td>HBase</td>
<td>20</td>
</tr>
<tr>
<td>Sherpa</td>
<td>30</td>
</tr>
<tr>
<td>MySQL</td>
<td>40</td>
</tr>
</tbody>
</table>

### Workload A - Update latency

<table>
<thead>
<tr>
<th>System</th>
<th>Update latency (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cassandra</td>
<td>5</td>
</tr>
<tr>
<td>HBase</td>
<td>10</td>
</tr>
<tr>
<td>Sherpa</td>
<td>15</td>
</tr>
<tr>
<td>MySQL</td>
<td>20</td>
</tr>
</tbody>
</table>

---

Ilya Bakulin (SMS Traffic)  
NoSQL benchmarking  
July 2, 2011
Some results: Workload "A": 50% read / 50% write

- Workload A – Update heavy
  • 50/50 Read/update

Comment: Cassandra is optimized for writes, and achieves higher throughput and lower latency. Sherpa and MySQL achieve roughly comparable performance, as both are limited by MySQL's capabilities. HBase has good write latency, because of commits to memory, and somewhat higher read latency, because of the need to reconstruct records.
Some results: Workload "B": 95% read / 5% write

- 95/5 Read/update

Comment: Sherpa does very well here, with better read latency – only one lookup into a B-tree is needed for reads, unlike log-structured systems where records must be reconstructed. Cassandra also performs well, matching Sherpa until high throughputs. HBase does well also, although read time is higher.
Some results: Workload "B": 95% read / 5% write

- Comment: Sherpa does very well here, with better read latency – only one lookup into a B-tree is needed for reads, unlike log-structured systems where records must be reconstructed. Cassandra also performs well, matching Sherpa until high throughputs. HBase does well also, although read time is higher.
Links

- Yahoo Cloud Serving Benchmark: https://github.com/brianfrankcooper/YCSB
- google://
- webmaster@kibab.com, kibab@FreeBSD.org