Yahoo! Cloud Serving Benchmark

Overview and results – March 31, 2010

Brian F. Cooper

cooperb@yahoo-inc.com

Joint work with Adam Silberstein, Erwin Tam, Raghu Ramakrishnan and Russell Sears

System setup and tuning assistance from members of the Cassandra and HBase committers, and the Sherpa engineering team

Versions of this deck

- V4.1 – Original set of results from benchmark
- V4.2 – added Cassandra 0.5 versus 0.4.2 comparison, Cassandra range query results, and vary scan size results
- V4.3 – Added more scalability data points, and Sherpa elasticity data
- V4.4 – Complete results from final YCSB paper

Motivation

- There are many “cloud DB” and “nosql” systems out there
  - Sherpa/PNUTS
  - BigTable
    - HBase, Hypertable, HTable
  - Megastore
  - Azure
  - Cassandra
  - Amazon Web Services
    - S3, SimpleDB, EBS
  - CouchDB
  - Voldemort
  - Dynomite
  - Etc: Tokyo, Redis, MongoDB

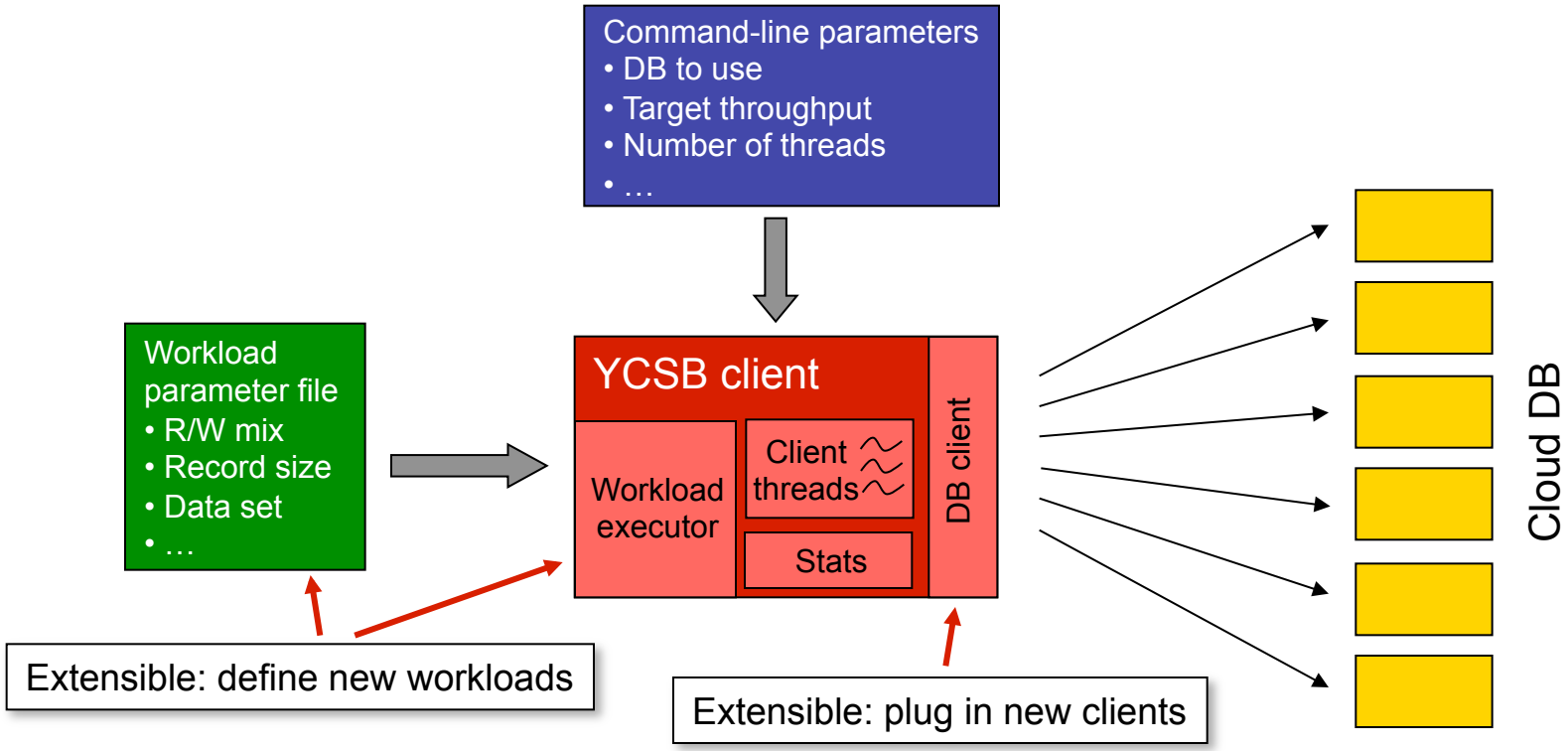
- How do they compare?
  - Feature tradeoffs
  - Performance tradeoffs
  - Not clear!

Goal

- Implement a standard benchmark
  - Evaluate different systems on common workloads
  - Focus on performance and scale out
    - Future additions – availability, replication
- Artifacts
  - Open source workload generator
  - Experimental study comparing several systems

Benchmark tool

- Java application
  - Many systems have Java APIs
  - Other systems via HTTP/REST, JNI or some other solution



Workloads

- Workload – particular combination of workload parameters, defining one workload
  - Defines read/write mix, request distribution, record size, …
  - Two ways to define workloads:
    - Adjust parameters to an existing workload (via properties file)
    - Define a new kind of workload (by writing Java code)
- Experiment – running a particular workload on a particular hardware setup to produce a single graph for 1 or N systems
  - Example – vary throughput and measure latency while running a workload against Cassandra and HBase
- Workload package – A collection of related workloads
  - Example: CoreWorkload – a set of basic read/write workloads

Benchmark tiers

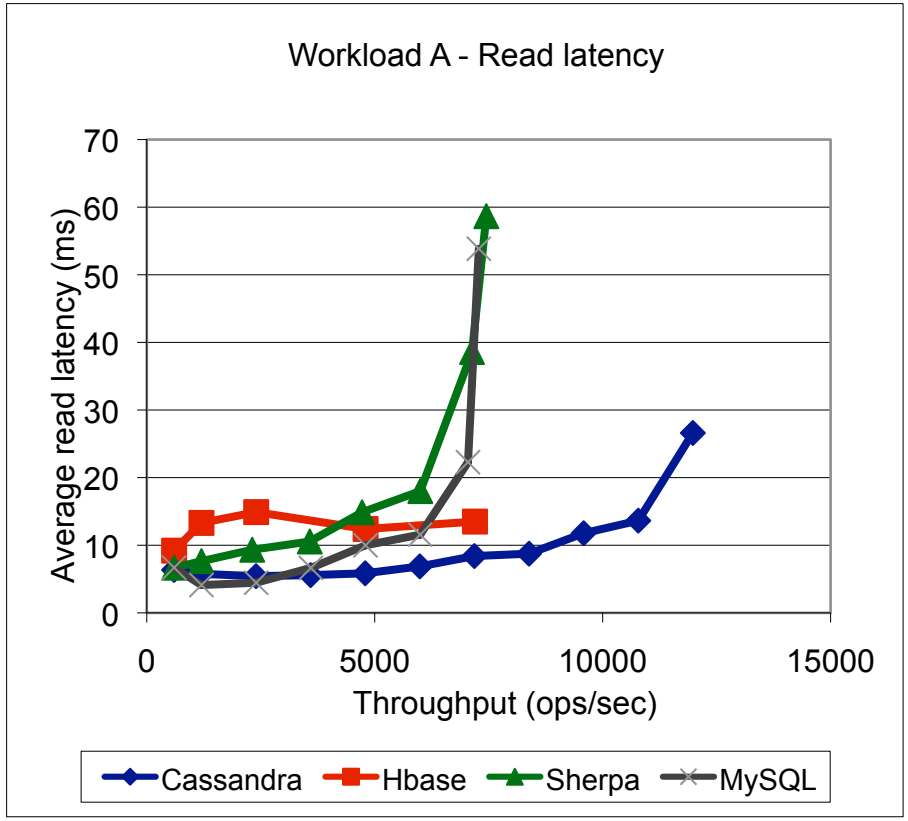
- Tier 1 – Performance
  - For constant hardware, increase offered throughput until saturation
  - Measure resulting latency/throughput curve
  - “Sizeup” in Wisconsin benchmark terminology
- Tier 2 – Scalability
  - Scaleup – Increase hardware, data size and workload proportionally. Measure latency; should be constant
  - Elastic speedup – Run workload against N servers; while workload is running att N+1th server; measure timeseries of latencies (should drop after adding server)

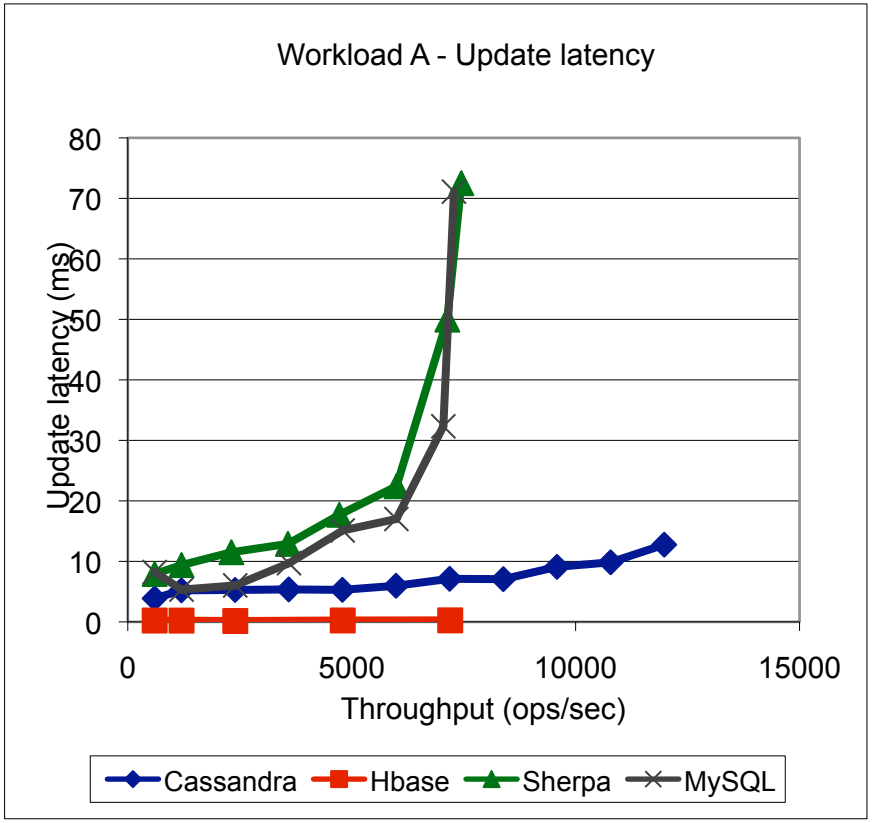
Test setup

- Setup
  - Six server-class machines
    - 8 cores (2 x quadcore) 2.5 GHz CPUs, 8 GB RAM, 6 x 146GB 15K RPM SAS drives in RAID 1+0, Gigabit ethernet, RHEL 4
  - Plus extra machines for clients, routers, controllers, etc.
  - Cassandra 0.5.0 (0.6.0-beta2 for range queries)
  - HBase 0.20.3
  - MySQL 5.1.32 organized into a sharded configuration
  - Sherpa 1.8 with MySQL 5.1.24
  - No replication; force updates to disk (except HBase, which primarily commits to memory)
- Workloads
  - 120 million 1 KB records = 20 GB per server
  - Reads retrieve whole record; updates write a single field
  - 100 or more client threads
- Caveats
  - Write performance would be improved for Sherpa, sharded MySQL and Cassandra with a dedicated log disk
  - We tuned each system as well as we knew how, with assistance from the teams of developers

Workload A – Update heavy

- 50/50 Read/update

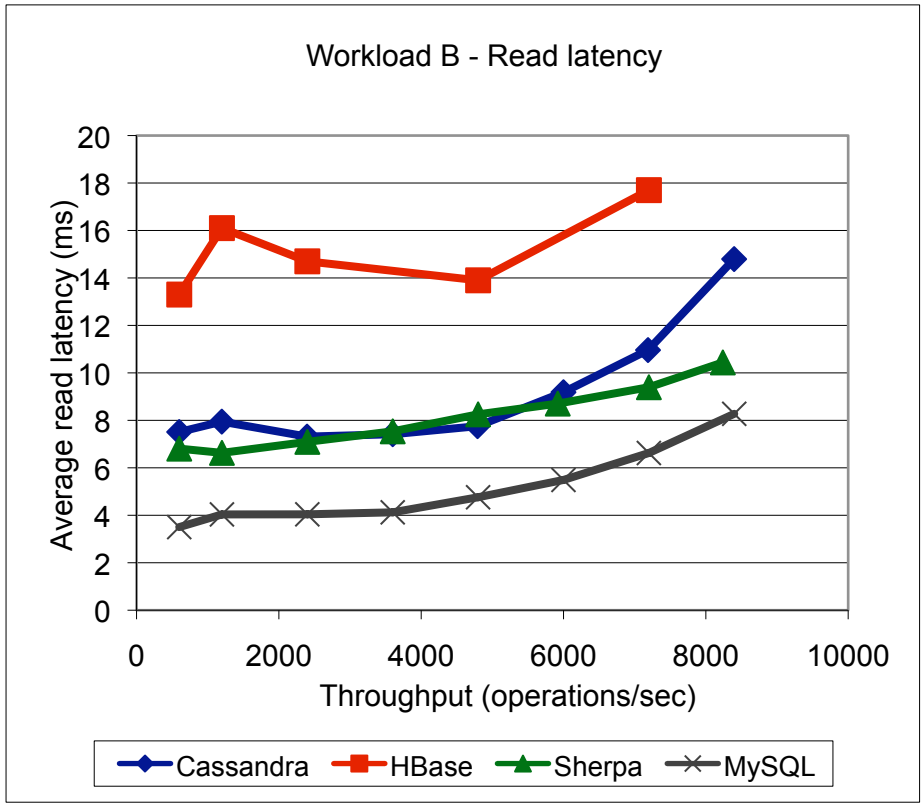


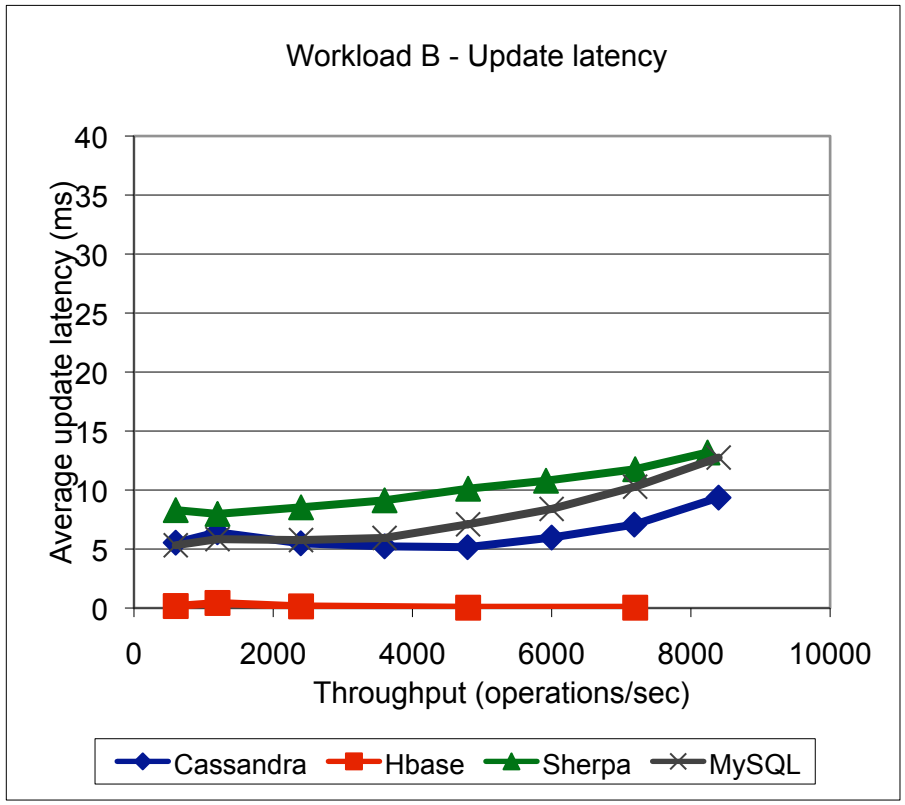


Comment: Cassandra is optimized for writes, and achieves higher throughput and lower latency. Sherpa and MySQL achieve roughly comparable performance, as both are limited by MySQL's capabilities. HBase has good write latency, because of commits to memory, and somewhat higher read latency, because of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

Workload B – Read heavy

- 95/5 Read/update

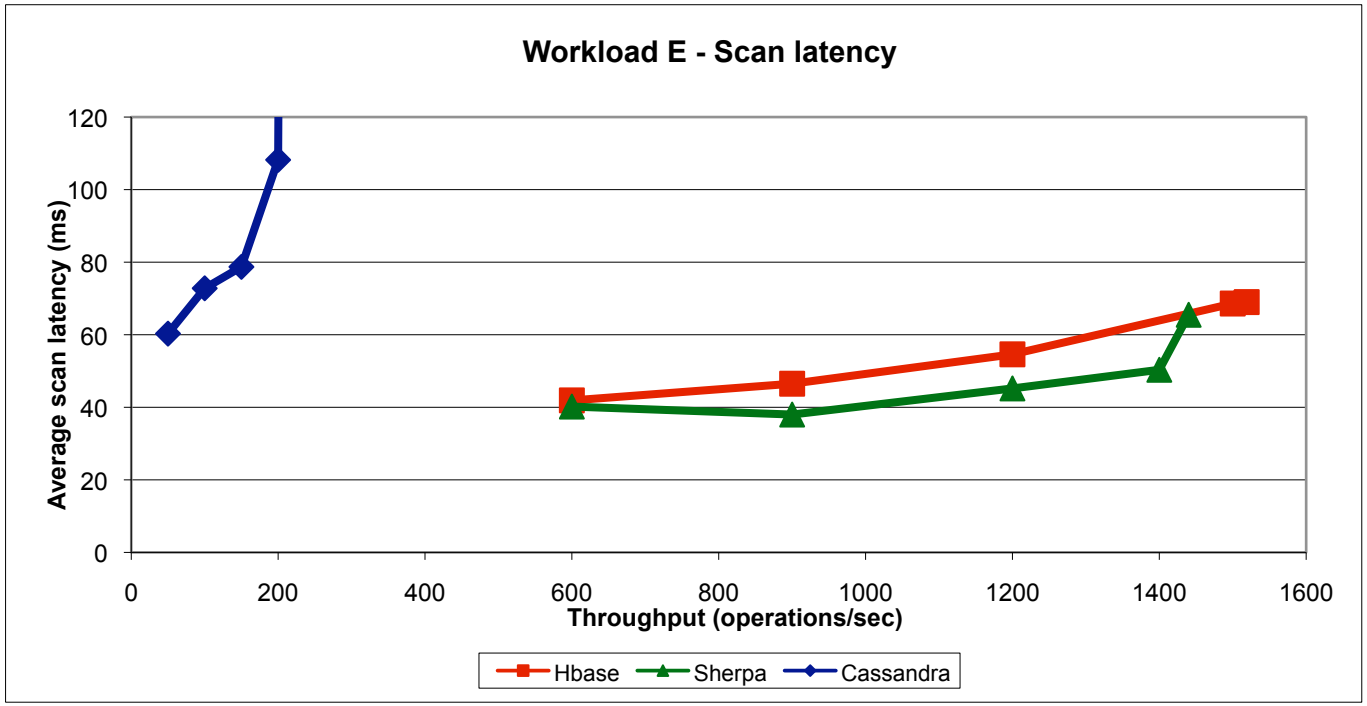




Comment: Sherpa does very well here, with better read latency – only one lookup into a B-tree is needed for reads, unlike log-structured systems where records must be reconstructed. Cassandra also performs well, matching Sherpa until high throughputs. HBase does well also, although read time is higher.

Workload E – short scans

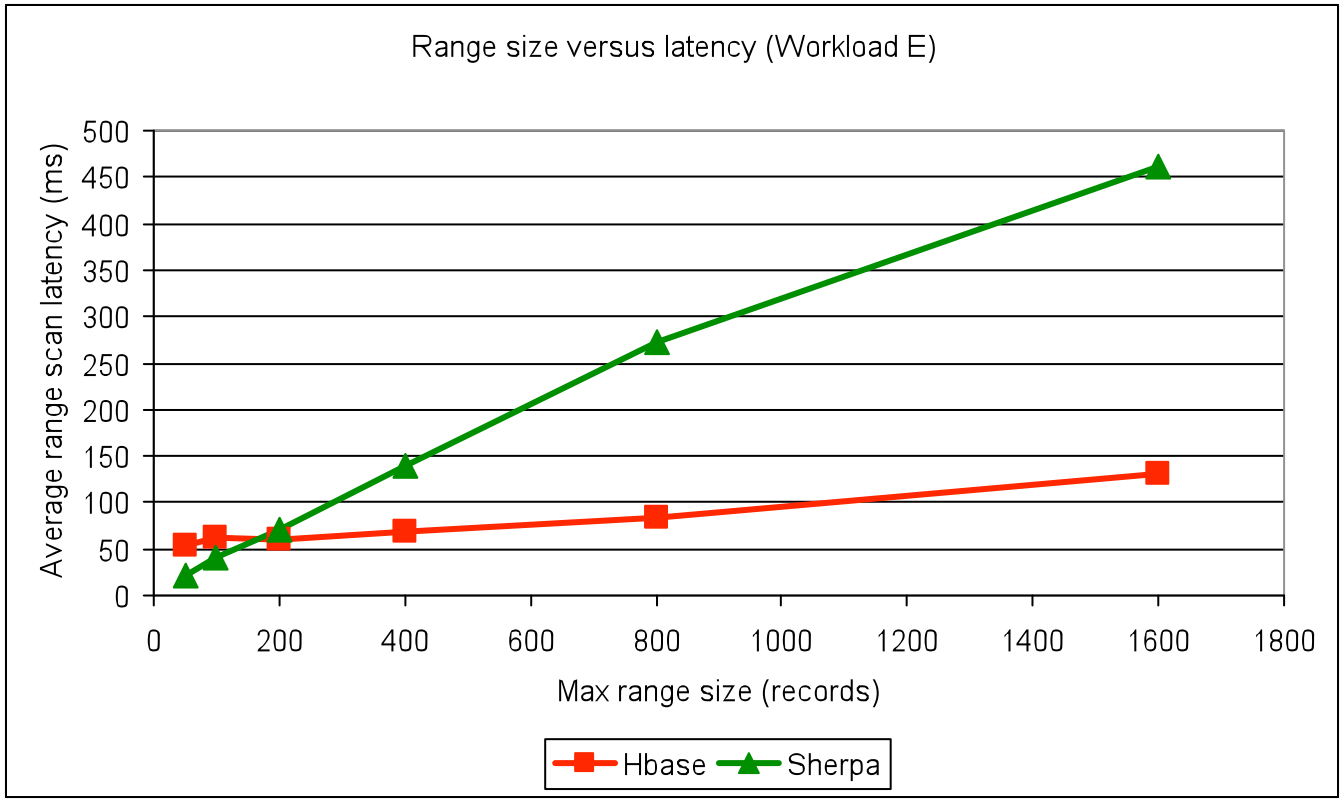
- Scans of 1-100 records of size 1KB



Comment: HBase and Sherpa are roughly equivalent for latency and peak throughput, even though HBase is “meant” for scans. Cassandra’s performance is poor, but the development team notes that many optimizations still need to be done.

Workload E – range size

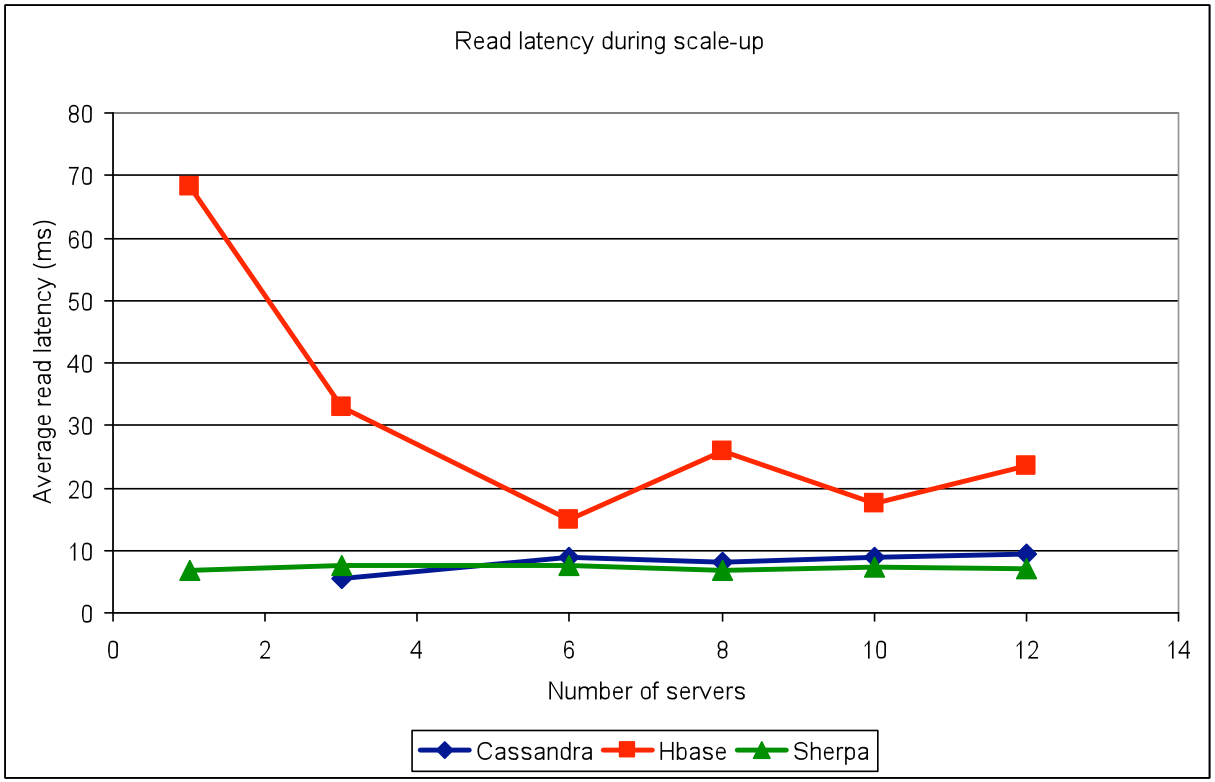
- Vary size of range scans



Comment: For small ranges, queries are similar to random lookups; Sherpa is efficient for random lookups and does well. As range increases, HBase begins to perform better since it is optimized for large scans

Scale-up

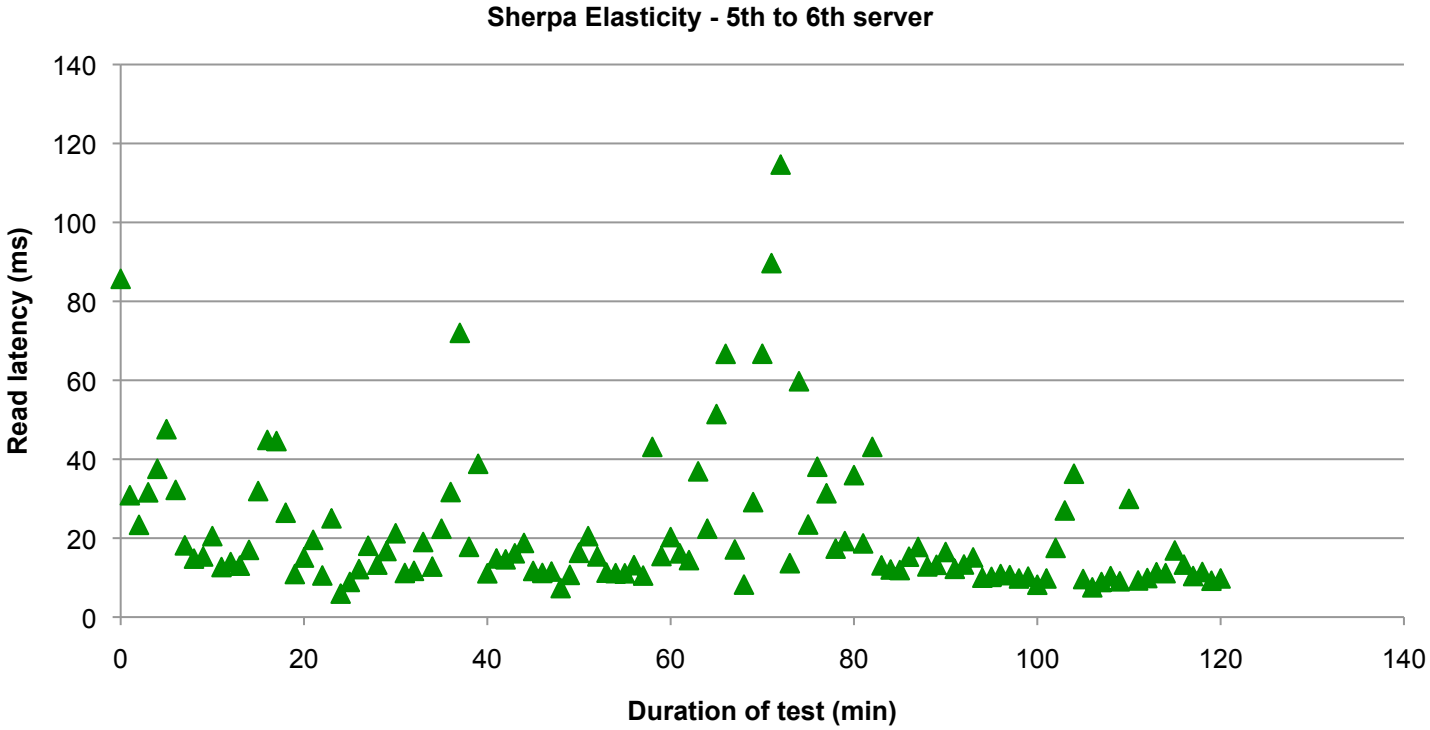
- Read heavy workload with varying hardware



Comment: Sherpa and Casandra scale well, with flat latency as system size increases. HBase is very unstable; 3 servers or less performs very poorly.

Elasticity

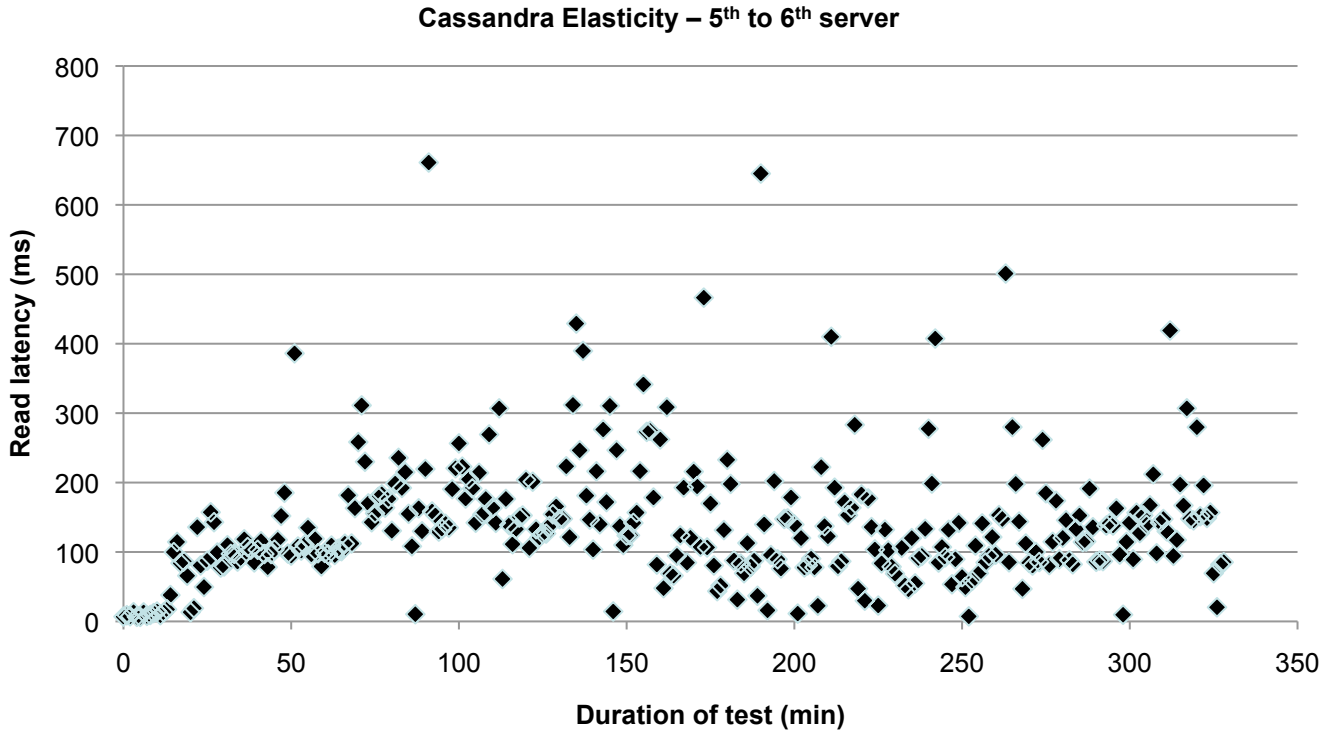
- Run a read-heavy workload on 2 servers; add a 4th, then 5th, then 6th server.



Comment: Sherpa shows variance in response time as tablets are moving, but after the data moves, it settles into an average that is faster than before the sixth server was added.

Elasticity

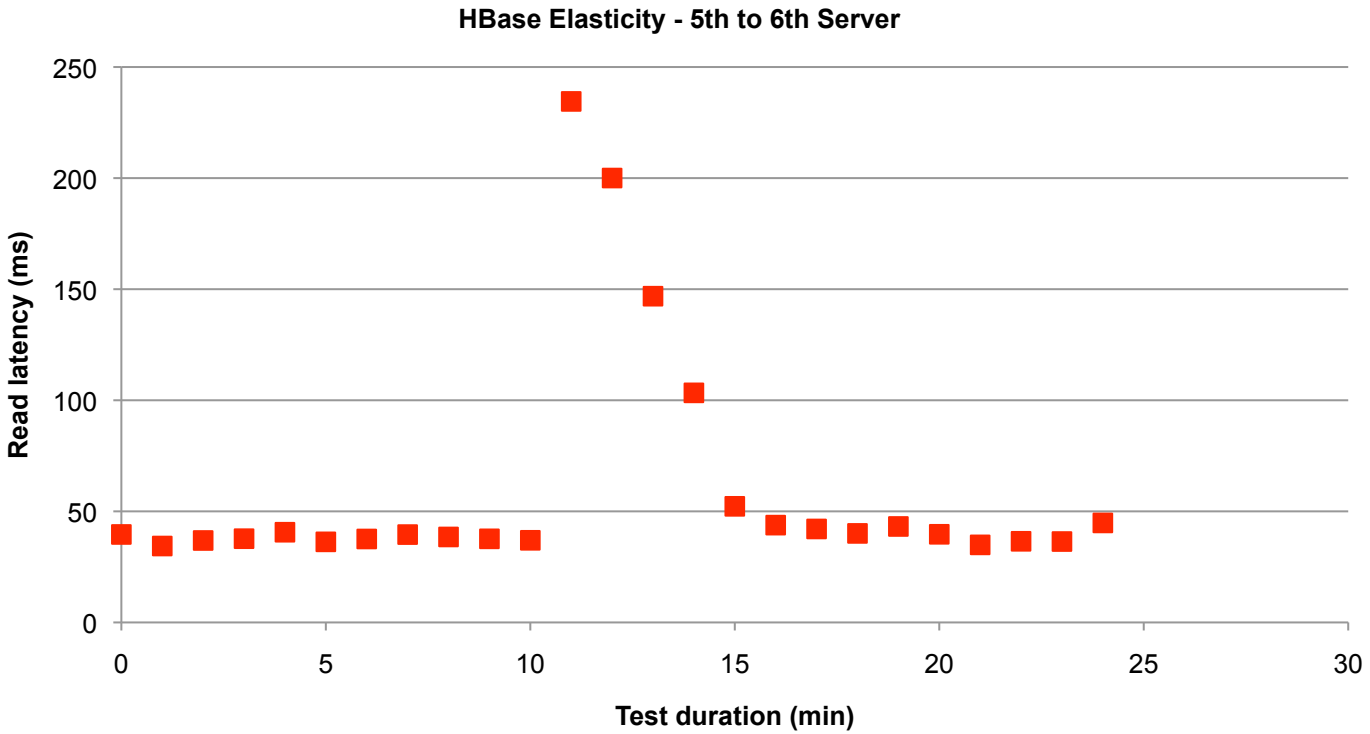
- Run a read-heavy workload on 2 servers; add a 4th, then 5th, then 6th server.



Comment: Cassandra shows lots of latency variance as it moves data to the new server, and takes multiple hours to stabilize

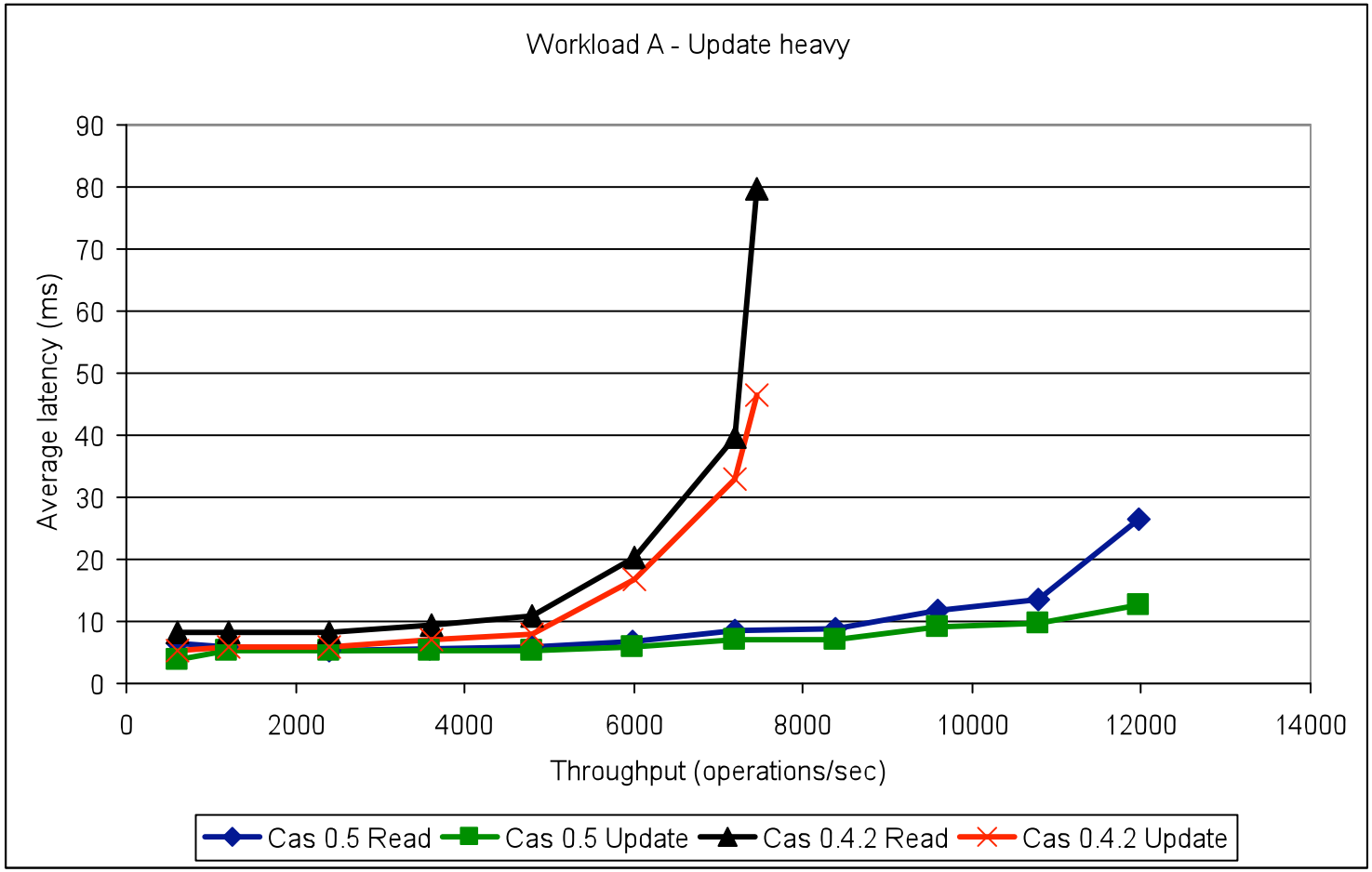
Elasticity

- Run a read-heavy workload on 2 servers; add a 4th, then 5th, then 6th server.

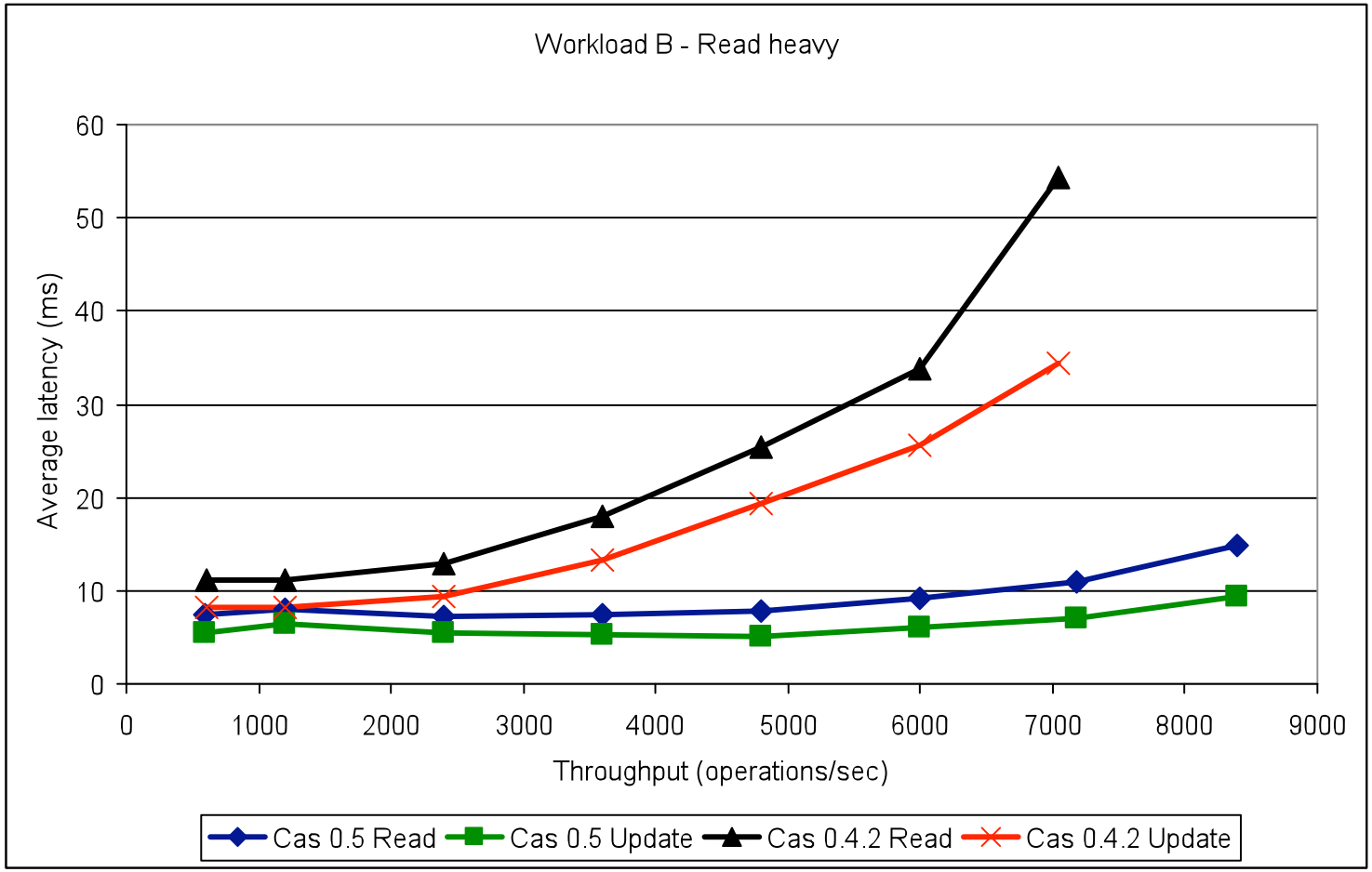


Comment: HBase shows a small latency bump as the cluster reconfigures. But data is not moved to the new server until a compaction is performed (not shown in the graph)

Cassandra 0.4.2 vs 0.5



Cassandra 0.4.2 vs 0.5



For more information

- Contact: Brian Cooper (cooperb@yahoo-inc.com)
- Detailed writeup of benchmark: http://www.brianfrankcooper.net/pubs/ycsb.pdf
- Open source YCSB tool coming soon (watch http://research.yahoo.com/Web_Information_Management/YCSB)