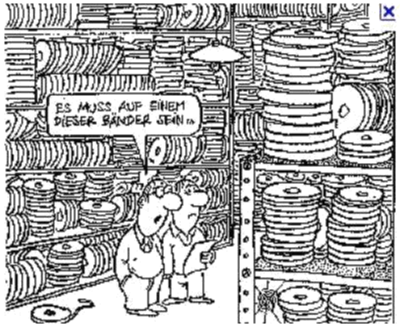
BENCHMARKING CLOUD DATABASES

CASE STUDY on HBASE, HADOOP and CASSANDRA USING YCSB

Planet Size Data!?

- Gartner's 10 key IT trends for 2012
  - unstructured data will grow some 80% over the course of the next five years



More (old) numbers!!!!

Facebook

- 1 billion active users, 1 in 3 Internet users have a Facebook account
- More than 30 billion pieces of content (web links, news stories, blog posts, notes, photo albums, etc.) shared each month. Holds 30PB of data for analysis, adds 12 TB of compressed data daily

Twitter

- 200 million users, 200 million daily tweets
- 1.6 billion search queries a day ,7 TB data for analysis generated daily

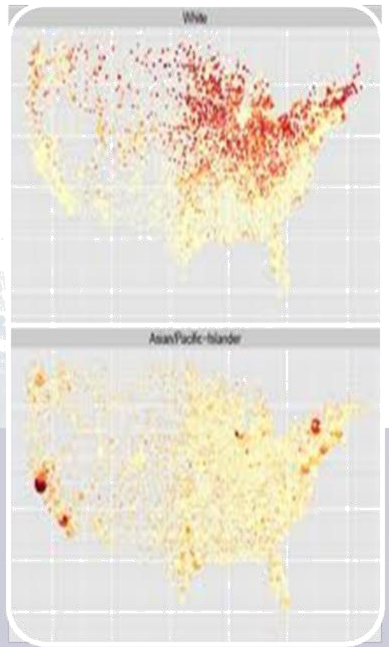
- **90 precent of the data in the world today has been created in the last two years alone**

- **Traditional data storage, techniques & analysis tools just do not work at these scales !**

- Source :http://www.rabidgremlin.com/data20/

Size matters but…

Performance



Geographic Distribution of users

Downtime measured from September 1, 2006 to September 1, 2007

Country	URL	Downtime in minutes	Uptime over a year
Brazil	www.google.com.br	3	99,999%
Netherlands	www.google.nl	11	99,998%
India	www.google.co.in	12	99,998%
Thailand	www.google.co.th	13	99,997%
Japan	www.google.co.jp	15	99,997%
Canada	www.google.ca	16	99,997%
Mexico	www.google.com.mx	16	99,997%
Egypt	www.google.com.eg	16	99,997%
Chile	www.google.cl	17	99,997%
France	www.google.fr	19	99,996%
Greece	www.google.gr	19	99,996%
United Arab Emirates	www.google.ae	20	99,996%
United Kingdom	www.google.co.uk	20	99,996%
Poland	www.google.pl	20	99,996%
Argentina	www.google.com.ar	21	99,996%
Hong Kong	www.google.com.hk	22	99,996%
Spain	www.google.es	22	99,996%
Italy	www.google.it	22	99,996%
Belgium	www.google.be	22	99,996%
Switzerland	www.google.ch	22	99,996%
Australia	www.google.com.au	26	99,995%
Romania	www.google.ro	27	99,995%
Saudi Arabia	www.google.com.sa	27	99,995%
Malaysia	www.google.com.my	28	99,995%
Germany	www.google.de	29	99,994%
United States	www.google.com	31	99,994%
[illegible]	www.google.cn	34	99,993%

Availability

Scalability

So many of them!

- http://the-opt.com/?p=66

CouchDB relax

Cassandra

mongoDB

H·BASE

One Way – Benchmark

- Benchmark
  - On what parameters ?
    - Performance
    - Scalability
    - Availability /Fault tolerance

Get to know the giants

- HBase over Hadoop
- Cassandra

Cloud DB – Concepts

- DFS :-
  - **Distributed File System** is a file system implementation on top of the underlying operating file system which allows the **creation and access of file from multiple hosts across network.**
  - DFS creates **multiple replicas** of data blocks and distributes them on nodes throughout a cluster to enable **reliable, scalable and extremely rapid computations.**

Cloud DB – Concepts

- **Reliability is augmented** because of the replication of same data in multiple nodes
- DFS allows addition/removal of newer nodes to the cluster ensuring smooth scale up and scale down with minimal or no manual **intervention (easy scalability).**
- **Computation speed gets a boost** since now any node can theoretically access any data and we can make use of data locality for computation

Map Reduce

- programming model for expressing distributed computations on massive amounts of data and an execution framework for large-scale data processing on clusters of **commodity servers.**

  - instead of **transferring** huge amount of data to a **central location** and then processing the retrieved **data in a sequential way**,

  - send chunks **of small code to the nodes** (which ideally has the data needed for the computation), - run the computation making use of the **data locality** (mapping) - and send result back to **reducer** which coalesces the result. -

Map Reduce

- **no degradation of performance due to communication latency**
- many algorithms **cannot** be easily expressed as a single Map Reduce job.
- But theoretically lot of process can be broken down in to a sequence of mapping and reducer tasks can be now run **parallel** on multiple nodes on the **cluster**;
- reduce the time taken for the process by a **factor of nodes involved**.

Data Model

- Both HBase and Cassandra follow a columnar data model approach
- NO to normalization theories
- Yes to replication
- Arrange data for Queries is the guiding rule

Consistency

- Cassandra [default] – Eventual consistency
  - Expected behavior:
    - Very fast writes
    - Slower reads
- HBase – Strict consistency
  - Expected behavior:
    - Very fast reads
    - Near optimal writes (comparatively slower)

Architecture

- Hbase follows
  - a master slave model for ensuring **optimised resource** and task allocation
  - potential performance **bottleneck**
  - a potential candidate for **single point failure**

Architecture

- Cassandra
  - follows a **decentralized** model and focusses more on **availability** and **fault tolerance** rather than ensuring strict consistency of data

Metrics Used for Comparison

- *Performance:-*
  - number of transactions done per second. (basic)
  - might not give a clear picture for real world application where we have to factor in the latency.
  - Hence we measure performance not as throughput offered by the two services; but we compare the trade-off of **latency vs. through put**.

- **i.e: A service/system with better performance will achieve the desired latency and throughput with the same amount servers.**

Scalability

- System with good scale up features is that ; in which the latency should remain constant or reduce, as the
  - number of servers and
  - offered throughput scale upwards proportionally.

Experimental Setup

- **7 server-class** machines and one extra machine for clients.
- **Cassandra version V1.7.0**
- **HADOOP version 0.20.203.0 and HBASE version 0.92.1**
- No replication(other than defaults)
- Force updates to disk (except HBase, which Primarily commits to memory) (default behaviour YCSB)

Hard ware configuration

- Compute node Configuration:
  - Vendor_id : AuthenticAMD
  - Vpu family : 15
  - Model : 5
  - Model name : AMD Opteron(tm) Processor 246
  - Stepping : 8
  - **CPU MHz : 2004.296**
  - **Cache size : 1024 KB**
  - FPU : yes
  - Cache_alignment : 64
  - Address sizes : 40 bits physical, 48 bits virtual

Cassandra Cluster

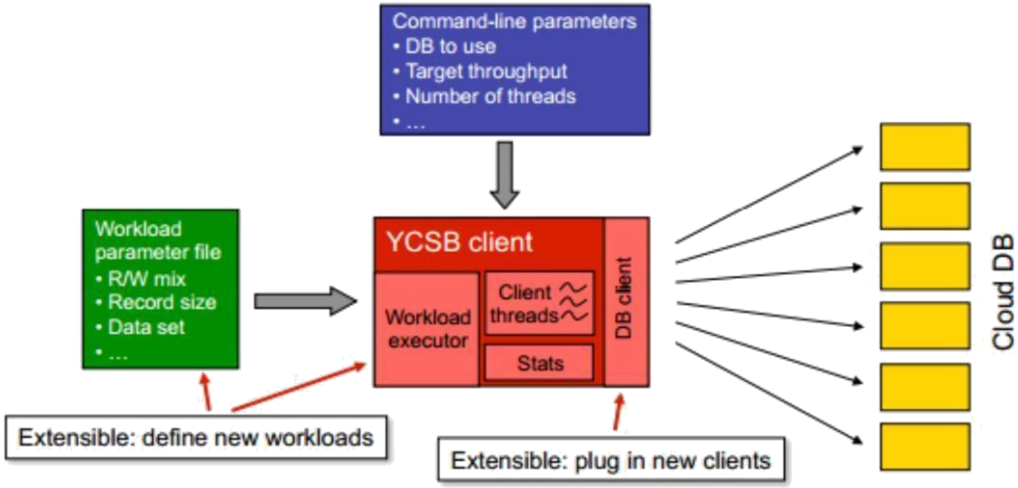
- All seven nodes were used as Cassandra nodes.
- We used **RandomPartitioner** to allow Cassandra to do the distributes rows across the cluster evenly and reduce extra overhead if any
- We also allocated 3 GB ram for Cassandra nodes.

HBase/Hadoop Cluster

- **HADOOP CLUSTER**: we have 1 **name node** (compute-0-11) and 3 **data nodes** (compute-0-11, compute-0-7, and compute-0-8).
- **HBASE CLUSTER**: we have 1 Master node (compute-0-3) and 4 region servers (compute-0-2, compute-0-3, compute-0-4, and compute-0-5).
- Hence in total we have dedicated seven servers for HBASE/ HADOOP Cluster.

- We allocated 1 GB ram to HADOOP and 3 GB ram to HBASE respectively

YCSB



The YCSB is a Java program for generating the data to be loaded to the database, and generating the operations which make up the workload.

YCSB

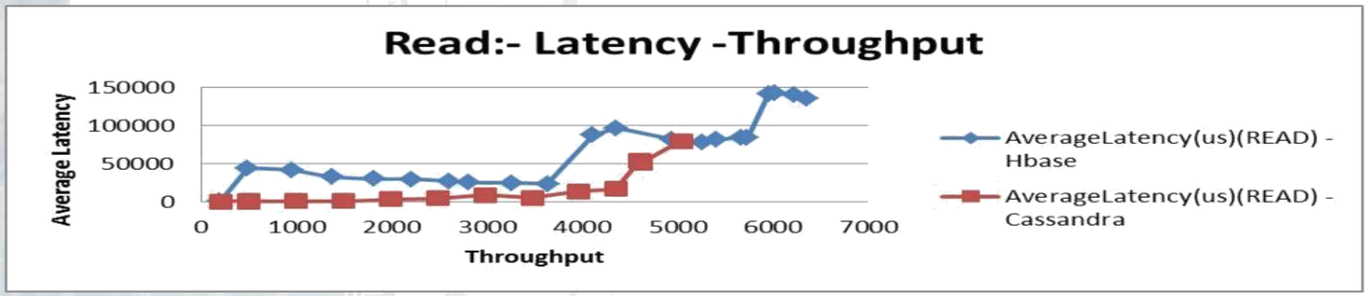
- Workload executor drives multiple client threads.
- Each thread executes a sequential series of operations
  - to load the database (the load phase)
  - and to execute the workload (the transaction phase

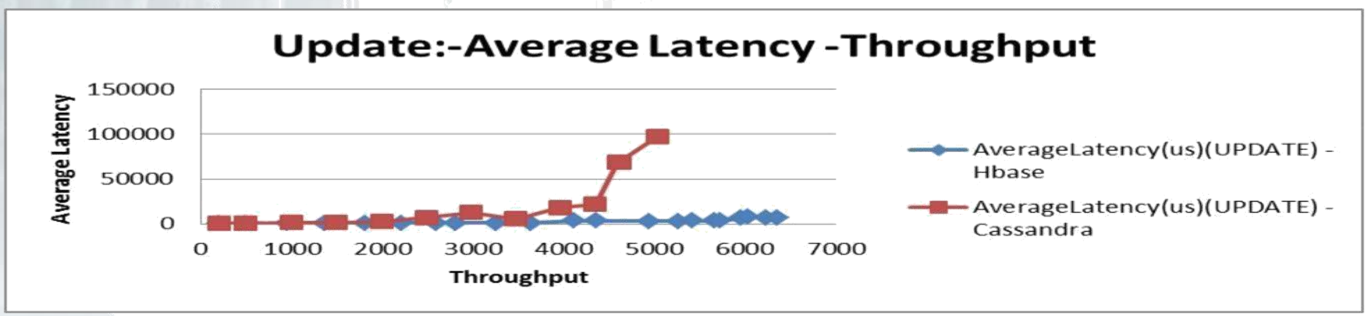
YCSB

- At the end of the experiment, the statistics module aggregates the measurements and reports
  - average, 95th and 99th percentile latencies,
  - and either a histogram or time series of the latencies

Results : Work load A

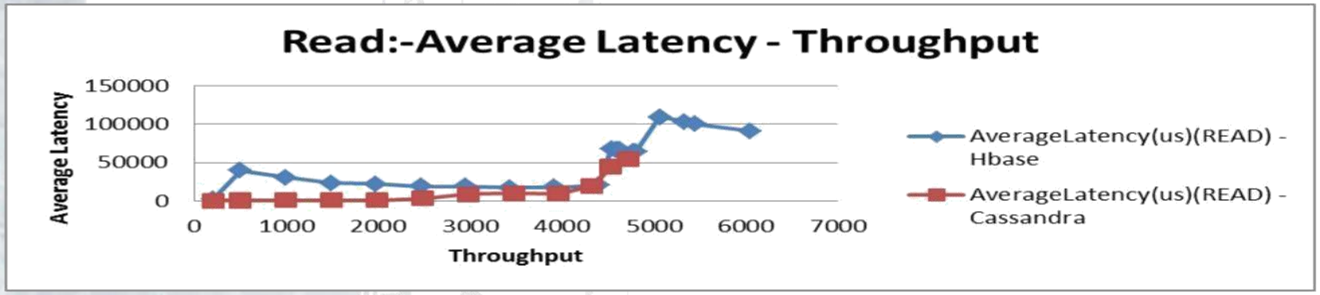
- *Workload A – **50% READ-50% UPDATE***

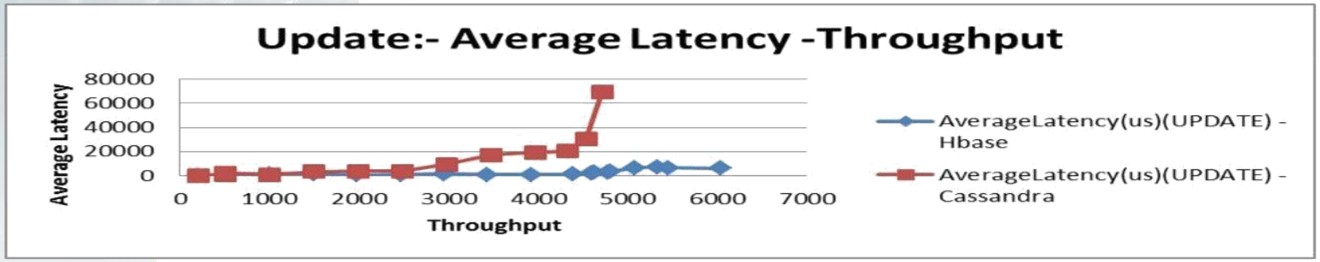




Results : Work load B

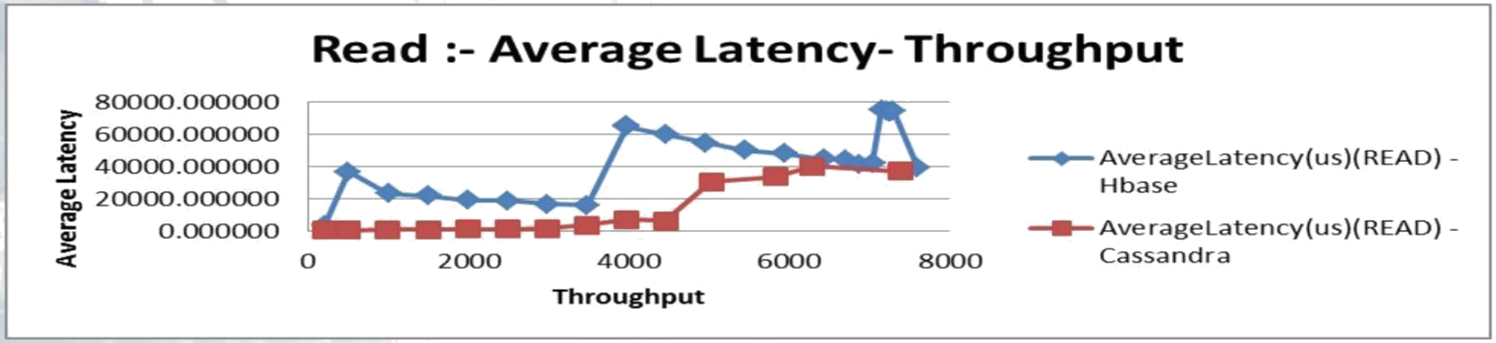
- ***Read Heavy 95% READ-5% UPDATE***





Results : Work load C

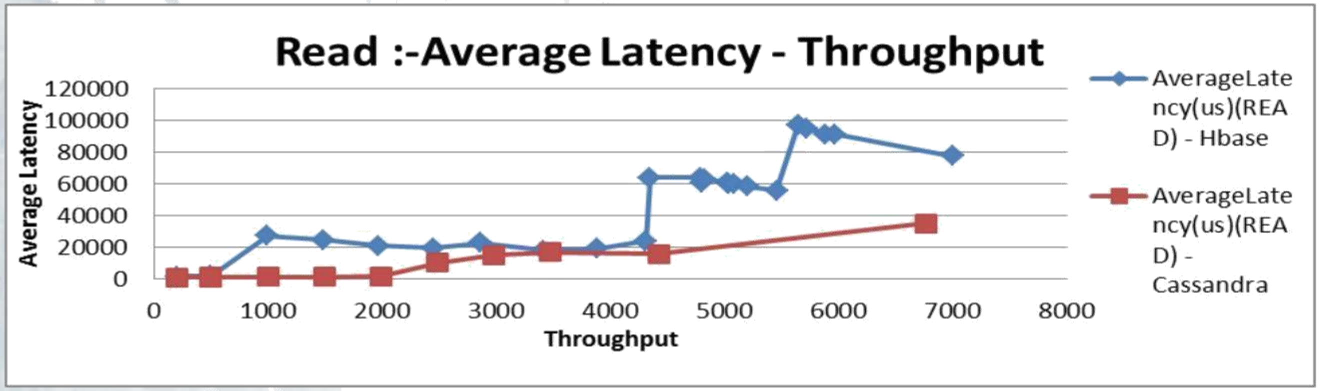
- **Read 100%**



Hence in a system with high reads and low updates Cassandra is slightly better than HBASE over HADOOP (if inconsistency of data is not much of an issue)

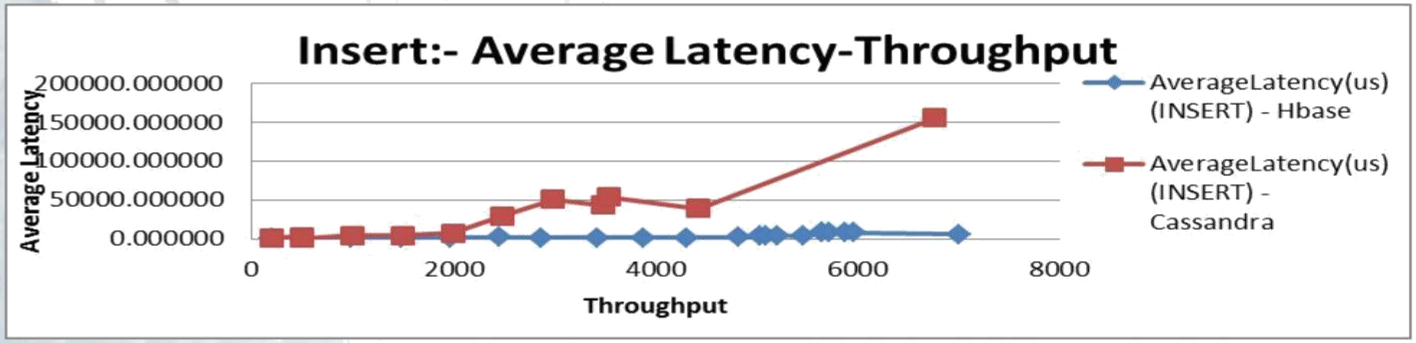
Results : Work load D

- ***INSERT 5% READ-95% UPDATE***



social networking sites where there are some inserts followed by heavy read operations on those insert.

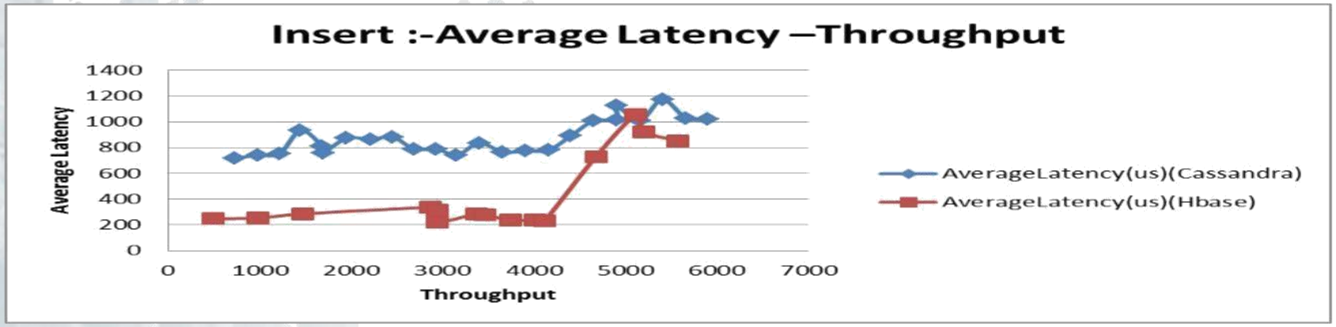
Results : Work load D



social networking sites where there are some inserts followed by heavy read operations on those insert.

Results : Work load E

- **100% INSERTS**

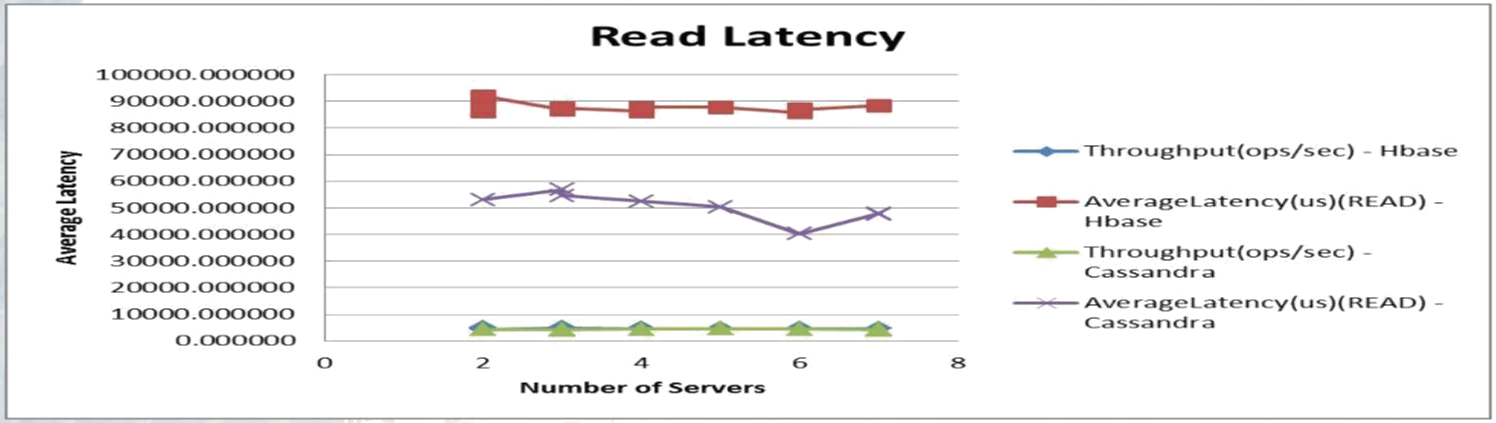


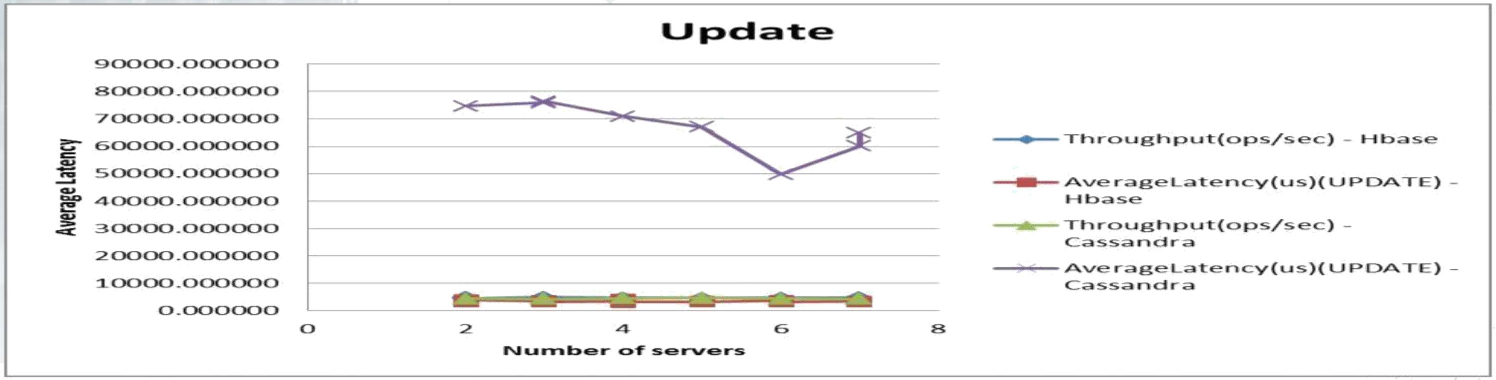
In a write heavy workload we see that at higher throughputs there is very little difference between HBase and Cassandra

Work load –Target App

	Workload	Operation	Target Application
1	A–Update heavy	Read:50% Update: 50%	**Session store recording recent actions in a user session**
2	B–Read heavy	Read: 95% Update: 5%	**Photo tagging; add a tag is an update, but most operations**
3	C–Read only	Read: 100%	**User profile cache, where profiles are constructed elsewhere**
4	F INSERTS Only	Inserts 100%	**Logging applications, Data transformation applications**

Scalability





Conclusion /Confusion

- HBase has good **write latency**, and somewhat **higher read latency**.
- Cassandra achieves **higher throughput** and **lower latency in** a comparable way for both writes/reads and updates.
- With respect to scalability for less number of servers (< 4) Cassandra scales better than HBASE

What next?

- Different cluster set ups
- Newer versions of Both Db’s
- YCSB – Map Reduce!?

Right tool for the Job

Thank you