Distributed Computing: Hadoop and NoSQL

Gautam Singaraju
Ask Analytics

Presented at USFCS
10/20/2011

Why Distributed Computing?

- Two Reasons:
  - Let’s see what's happening in Industry.
  - Let’s see what’s happening in Academia.

Industry “Skills” Growth Rate

- C: -3% YOY# ↓
- C++: -7% YOY# ↓
- Java: -3% YOY# ↓
- Databases, DB Admin: -6% YOY# ↓
- Hadoop: +40% YOY# ↑
- HBase: +62% YOY# ↑
- Map Reduce: +39% YOY# ↑
- Text Analytics: +3% YOY# ↑

Ceveat: Skills shown in Red are important; the Green gives a competitive advantage.

YOY Stands for Year over Year. The figure does not represents number of jobs.

→From LinkedIn.com Skills beta.

The point is, if you want to work in the industry :

Programming Languages are the starting point. Develop additional skills.

Distributed computing could be one of them…

Academia Research

- Personally: Spam & Malware Detection
  - VeriSign TLD Data and Botnet Data
- Artificial Intelligence/Machine Learning
- Data Mining/Information Retrieval
- Bioinformatics/Medical Informatics
- Statistics: Recommendation Engines

http://atbrox.com/2010/05/08/mapreduce-hadoop-algorithms-in-academic-papers-may-2010-update/

If you want to work in academia and work with large scales of data:

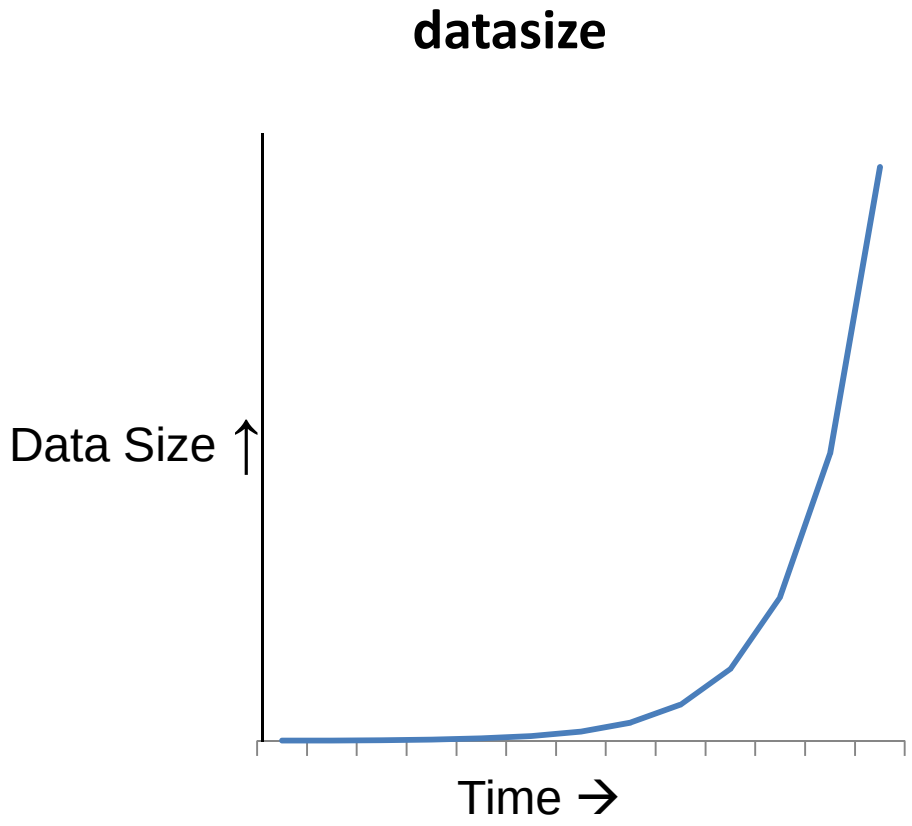
You might need distributed computing…

Outline

- Why Distributed Computing?
- Hadoop
  - Hadoop Distributed File System
  - Map Reduce
- NoSQL: HBase
  - CAP Theorem
  - HBase
    - Zookeeper
- Online Data Sources to work with.

Data Growth Rates

- Data is growing. Fast!
- Visit Ask.com; type "data growth trends"
- Types of data:
  - Financial
  - Census
  - Scientific
  - Astronomy
  - …



Why Distributed Computing?

- Processing on a single machine:
  - Will require significant time.
  - Sometimes, impossible with large sets of data.
- Distributed systems
  - Put together multiple machines in a cluster.
  - Increased aggregated bandwidth.

What is Hadoop*?

- Distributed Computing framework
  - Developed in Java, open source.
  - Capable of handling large scales of data and nodes
  - Batch processing system.
  - Offline analysis of system.
- Essential Sub-systems:
  - Hadoop Distributed File System.
  - Map-Reduce

* http://hadoop.apache.org/

Hadoop Distributed File System

HDFS

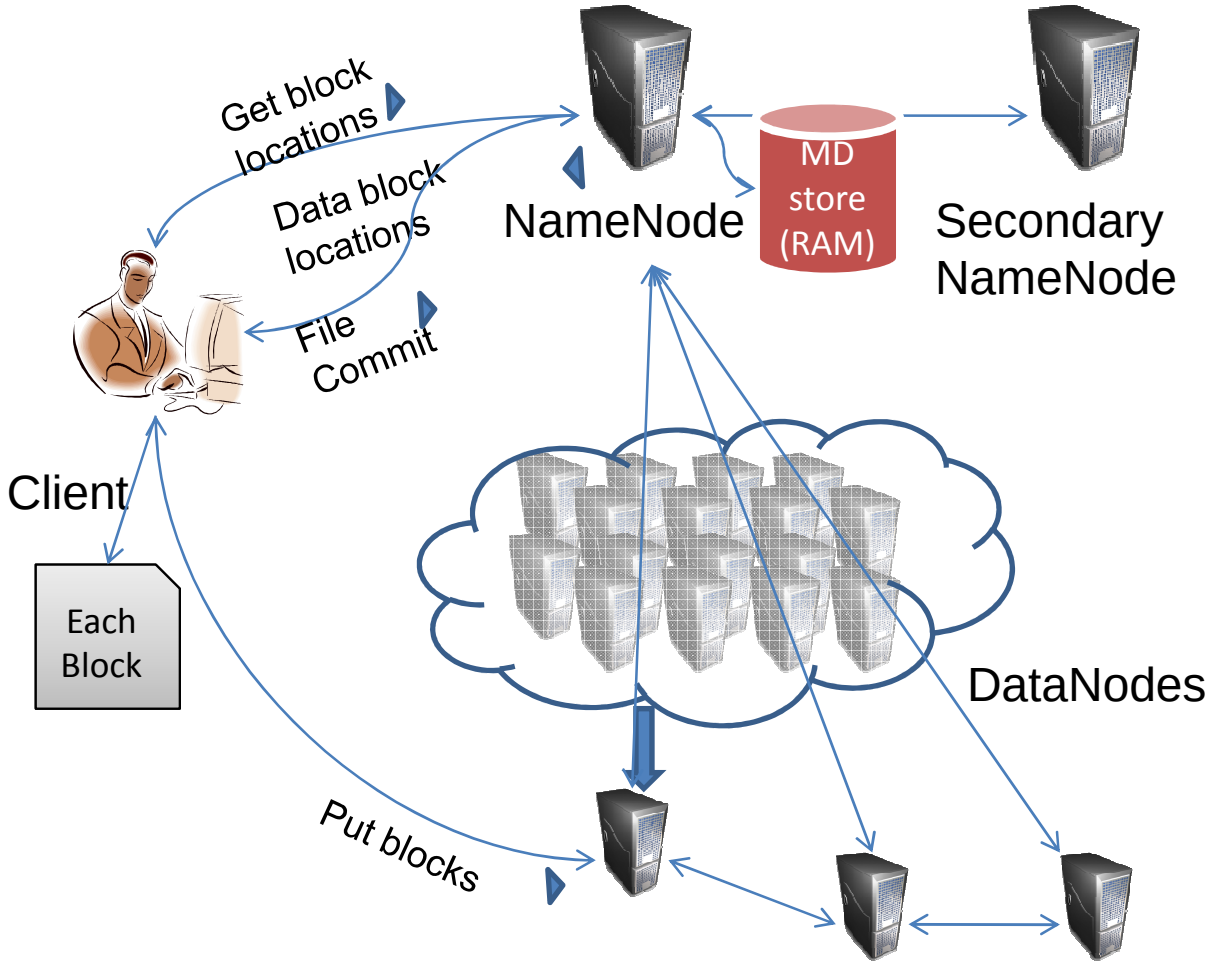
Goals of HDFS

- NOT Really a file System!
  - It is not POSIX compliant.
  - Runs on other Operating Systems; runs in user space.
  - Run it on local hard-disk; not NFS mounts!
- Commodity Hardware (each node is a few thousand $$)
  - No Raid for each node; but need RAID support for some.
  - Detect and recover from Node failures.
- Distributed File System
  - files broken into blocks. Each block replicated 3 times.
  - Each block is 64MB/128MB
- Optimized for Batch Processing
  - Computation near data resides
  - Provides very high aggregate bandwidth

HDFS Components

- Name Node
  - Single point of failure; maintains Metadata in-memory
  - Types of Metadata
    - Files, Blocks for file, Data Nodes for block. Other file attributes.
- Secondary Name Node
  - Not really a back-up Name Node!
  - Maintains edit logs of Name Node.
- Data Node
  - Store data on local file system.
  - Sends existing block report to Name Node

HDFS Architecture 1- Write

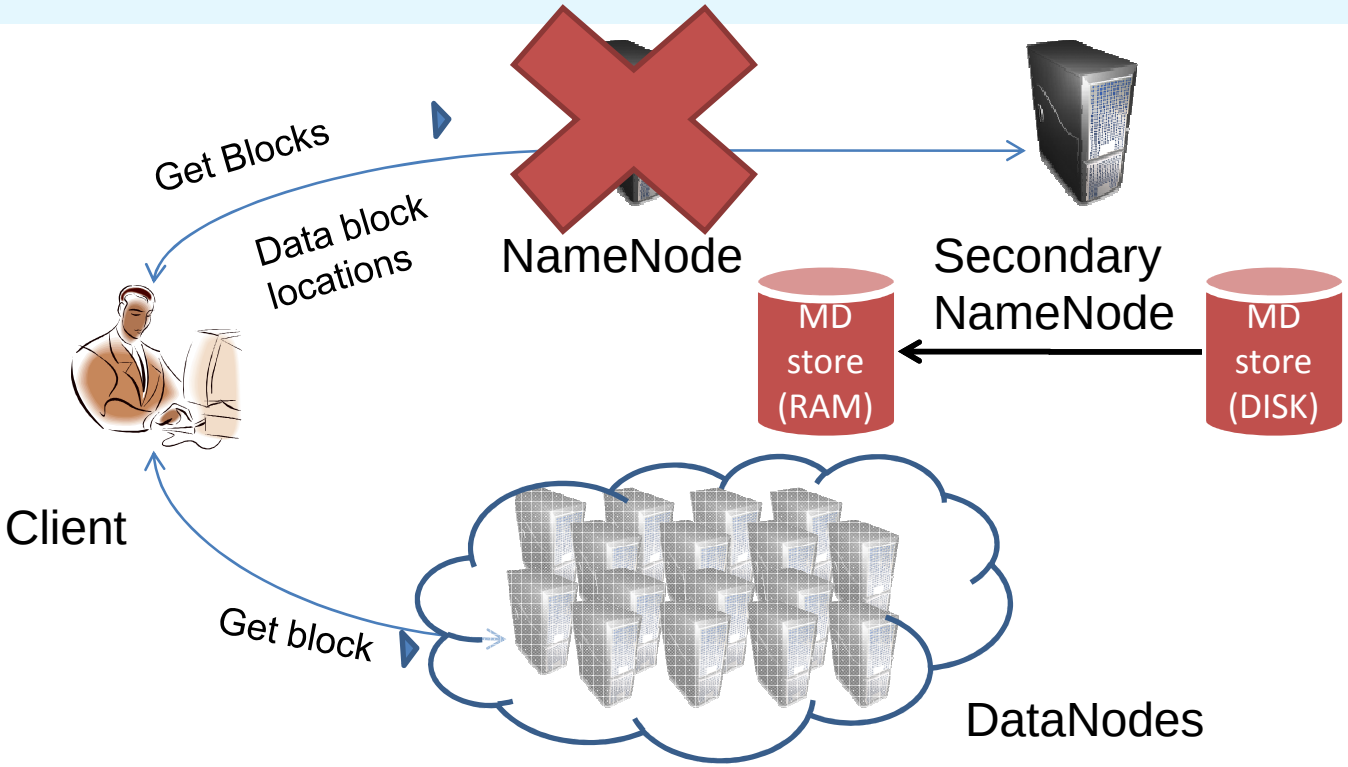


A sophisticated replica placement strategy.
Copy 1: on one machine.
Copy 2: on same rack.
Copy 3: on a different rack.

Periodically, the Namenode checks: if the # replica/block is less than replication number, another copy of the block is made.

CRC checks on blocks are done to verify block correctness.

HDFS Architecture 2 - Read



In case of Name Node failure, a 5-minute backup snapshot copies of meta-data are available on local disk and over NFS.

Whatever new data that has been added since the last backup metadata are “bad” blocks.

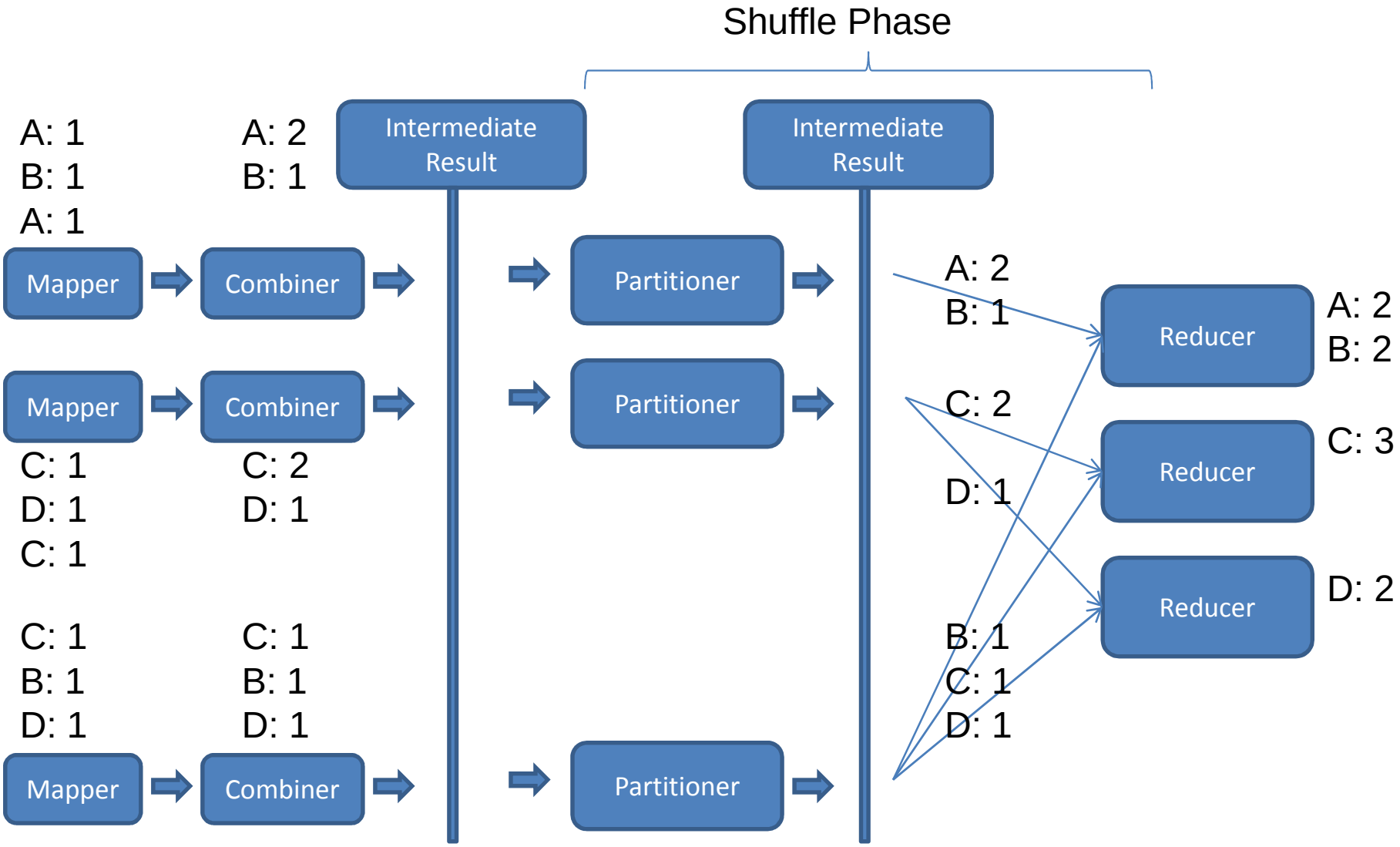
MAP REDUCE

Map/Reduce *

- Map/Reduce is an old concept.
- Map: Given a set of key - value pairs: Output none to more key, value pairs.
- Combiner: For a single key from the mapper, process all values and output 0 or more key, value pair.
- Shuffle Phase: Data is shuffled (parallel-sorted / exchanged between nodes).
- Reduce: For a single key from all mappers, process all values and output 0 or more key, value pair.

* http://hadoop.apache.org/mapreduce/

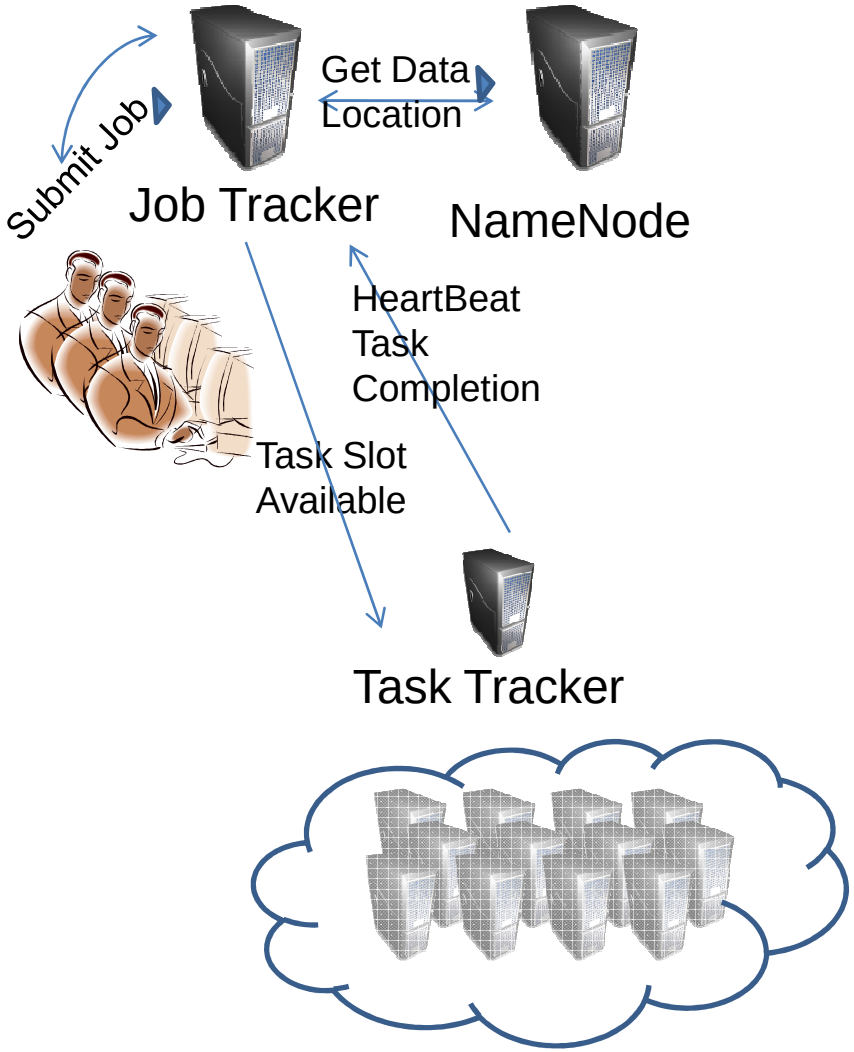
Map/Reduce Framework Explained



More Components

- Job Tracker:
  - Clients submits a job to job tracker.
  - Tracks the progress of a job.
- Task Tracker
  - Tracks the progress each task.

Map/Reduce Framework Explained



Task Tracker runs on each node.
Users can check the job progress at the Job tracker.

Job Scheduler Algorithms
1. FIFO Scheduler
2. Fair Scheduler

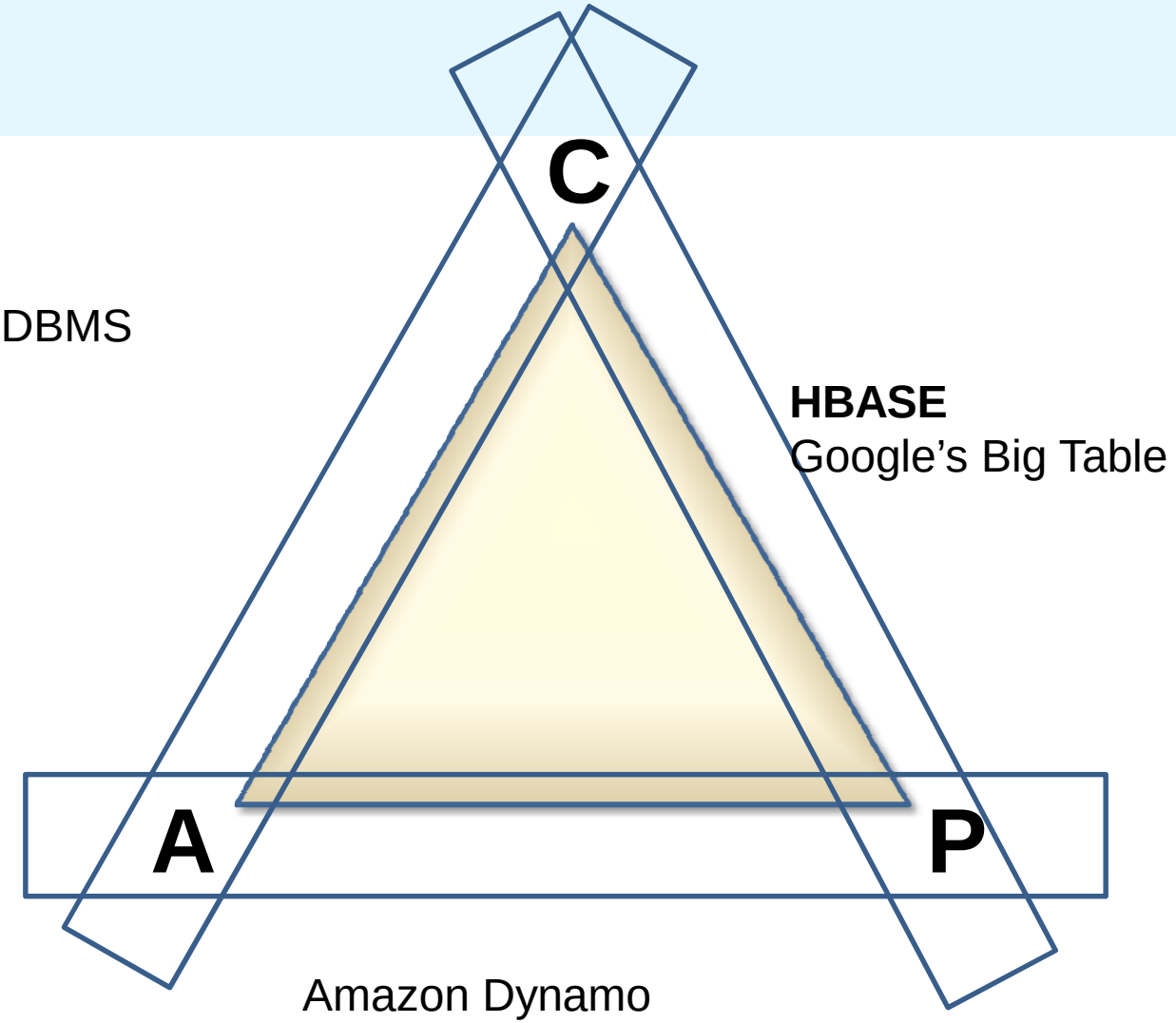
Other Frameworks

- If you want to use Hadoop, you might have to:
- build your own or download open-source frameworks for:
  - Data Organization and Location Framework.
  - Job Management.
  - Result Delivery.

HBASE

CAP Theorem

- Brewer's CAP Theorem
  - Consistency
  - Availability
  - Partition Tolerance
- Prof. Eric Brewer, UCB at the ACM Symposium on the PoDC 2000.
- 2002 was formal proved by Dr. Seth Gilbert and Prof. Nancy Lynch.
- Amazon:
  - 0.1 second increase in latency drops 1% of the traffic.
- Google:
  - 0.5 second increase in latency drops 5% of the traffic.



RDBMS

HBASE
Google's Big Table

Amazon Dynamo
Cassandra
CouchDB
Project Voldemort

HBase *

- Random, real-time read/write access to Hadoop
- Modeled after Google' Bigtable.
- Optimizations for real time queries.
- No SQL, No joins.

* http://hbase.apache.org/

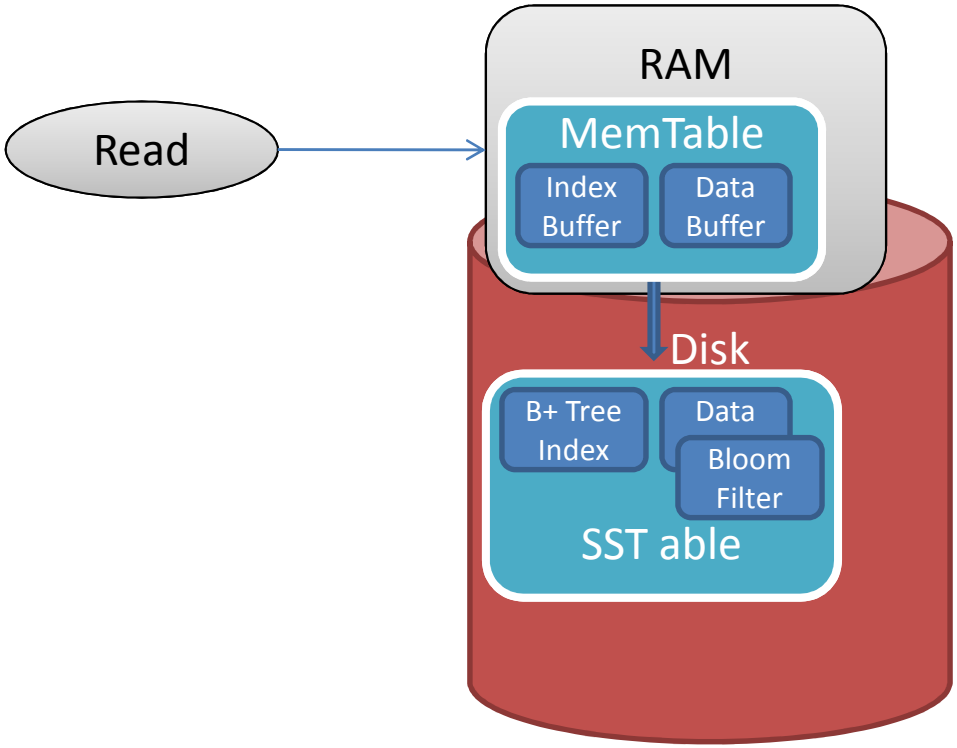
HBase Benefits

- Automatic Key Space Partitioning
  - Frequently accessed key space will be split into two regions and one region is moved to a different region server for increased performance.
- Automatically scales linearly with new nodes.
- **Hadoop advantages**:
  - Commodity hardware.
  - Fault tolerance.
- Run Map/Reduce over it!
  - Super quick!

Frameworks

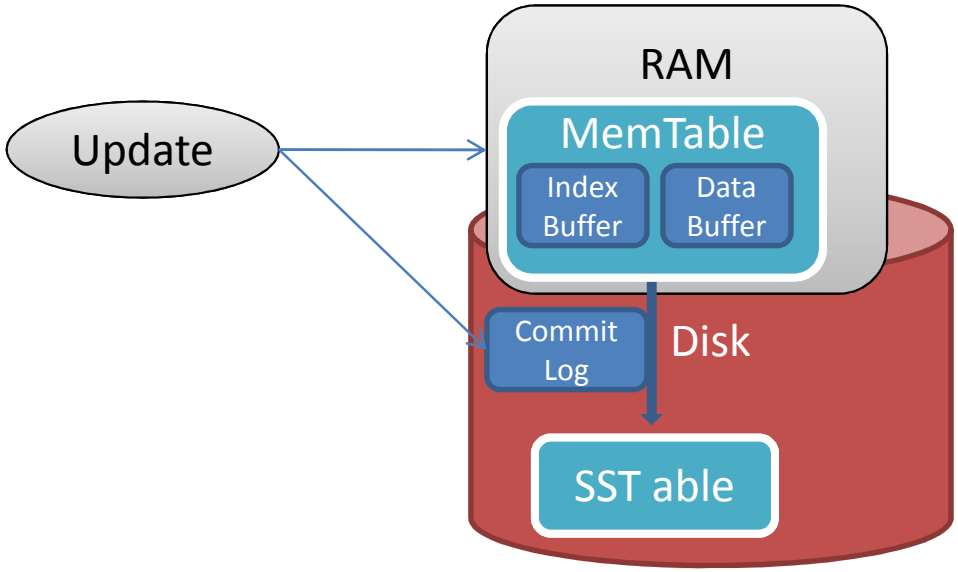
- Zookeeper
  - A coordination framework for distributed applications.
- HBase Components:
  - Master: Monitoring, load balances region servers and Redirect client to correct region servers
  - Region Server (each node): Serving client requests (Write/Read/Scan), Send heart beats to Master.
  - Regions (key space)

HBase Read Architecture

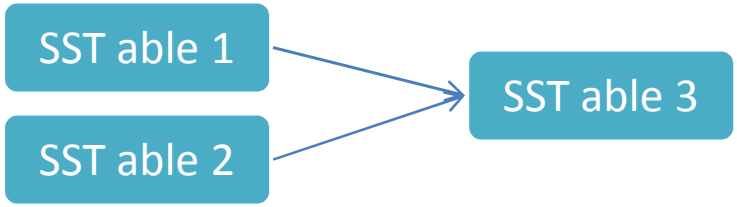


- Check the newest copy in memory, if not available, hit the disk.
- On disk, the data is stored in a combination of B+ tree Index; and data chucks are checked with the help of Bloom filter.
- The combination of B+ tree index, bloom filter gives increased performance for on disk operations.

HBase Update Architecture



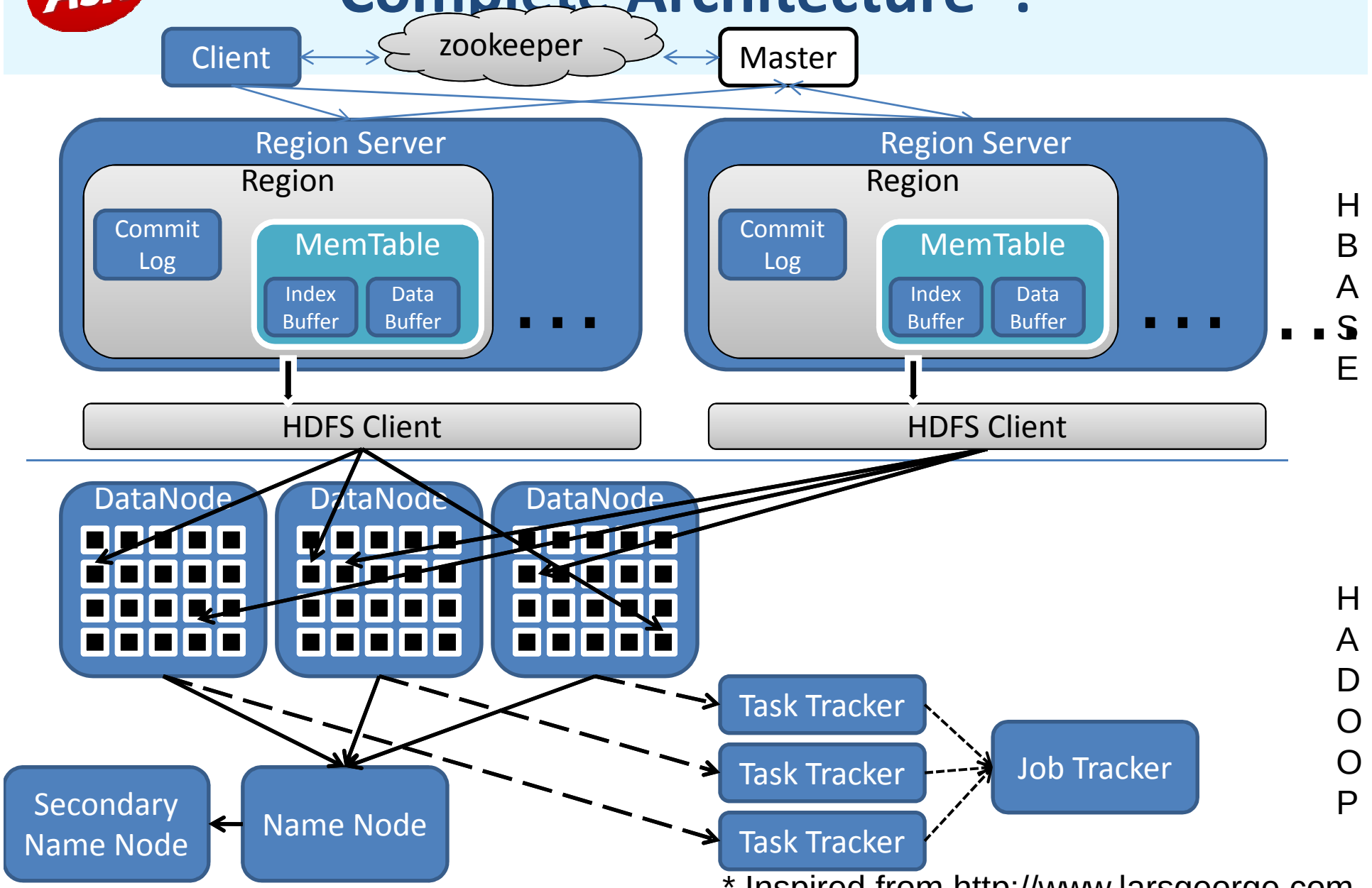
Compactions



Compactions are used to purge any deleted keys. This is typically a time consuming process and is done during reduced traffic times or by copying the data onto a secondary cluster.

- On disk, the each column data is stored sequentially in a SSTable.
- When an update operation is performed, the update is applied in memory and a commit log is maintained on disk.
- When a system is down, a transaction log is replayed and the most current version is stored in memory.

Complete Architecture*!



* Inspired from http://www.larsgeorge.com

Great Data Sets

http://www.data.gov

http://www.usa.gov/

http://www.infochimps.org

http://2010.census.gov/

You can connect with me at:

gautam@singaraju.com

http://www.linkedin.com/in/gautams

Questions?

THANK YOU