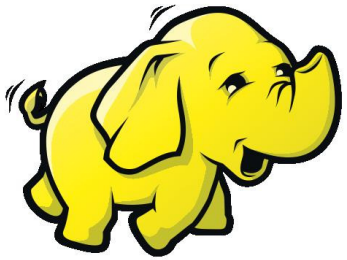
Scaling Storage and Computation with Apache Hadoop

Konstantin V. Shvachko
Yahoo!

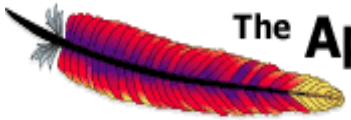
4 October 2010

What is Hadoop

- Hadoop is an ecosystem of tools for processing "Big Data"



- Hadoop is an open source project



- Yahoo! a primary developer of Hadoop since 2006

YAHOO!

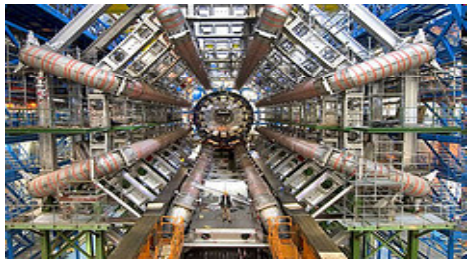
Big Data

- Big Data management, storage and analytics
- Large datasets (PBs) do not fit one computer
  - Internal (memory) sort
  - External (disk) sort
  - Distributed sort
- Computations that need a lot of compute power

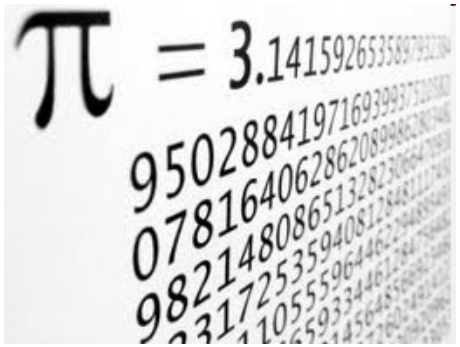
Big Data: Examples

- Search Webmap as of 2008 @ Y!
  - Raw disk used 5 PB
  - 1500 nodes

- Large Hadron Collider: PBs of events
  - 1 PB of data per sec, most filtered out



- 2 quadrillionth (10^{15}) digit of π is 0
  - Tsz-Wo (Nicholas) Sze
  - 23 days vs 2 years before
  - No data, pure CPU workload



Big Data: More Examples

- eHarmony
  - Soul matching

- Banking
  - Fraud detection

- Processing of astronomy data
  - Image Stacking and Mosaicing

Hadoop is the Solution

- Architecture principles:
  - Linear scaling
  - Reliability and Availability
  - Using unreliable commodity hardware
  - Computation is shipped to data
    No expensive data transfers
  - High performance

Hadoop Components

HDFS	Distributed file system
MapReduce	Distributed computation
Zookeeper	Distributed coordination
HBase	Column store
Pig	Dataflow language
Hive	Data warehouse
Avro	Data Serialization
Chukwa	Data Collection

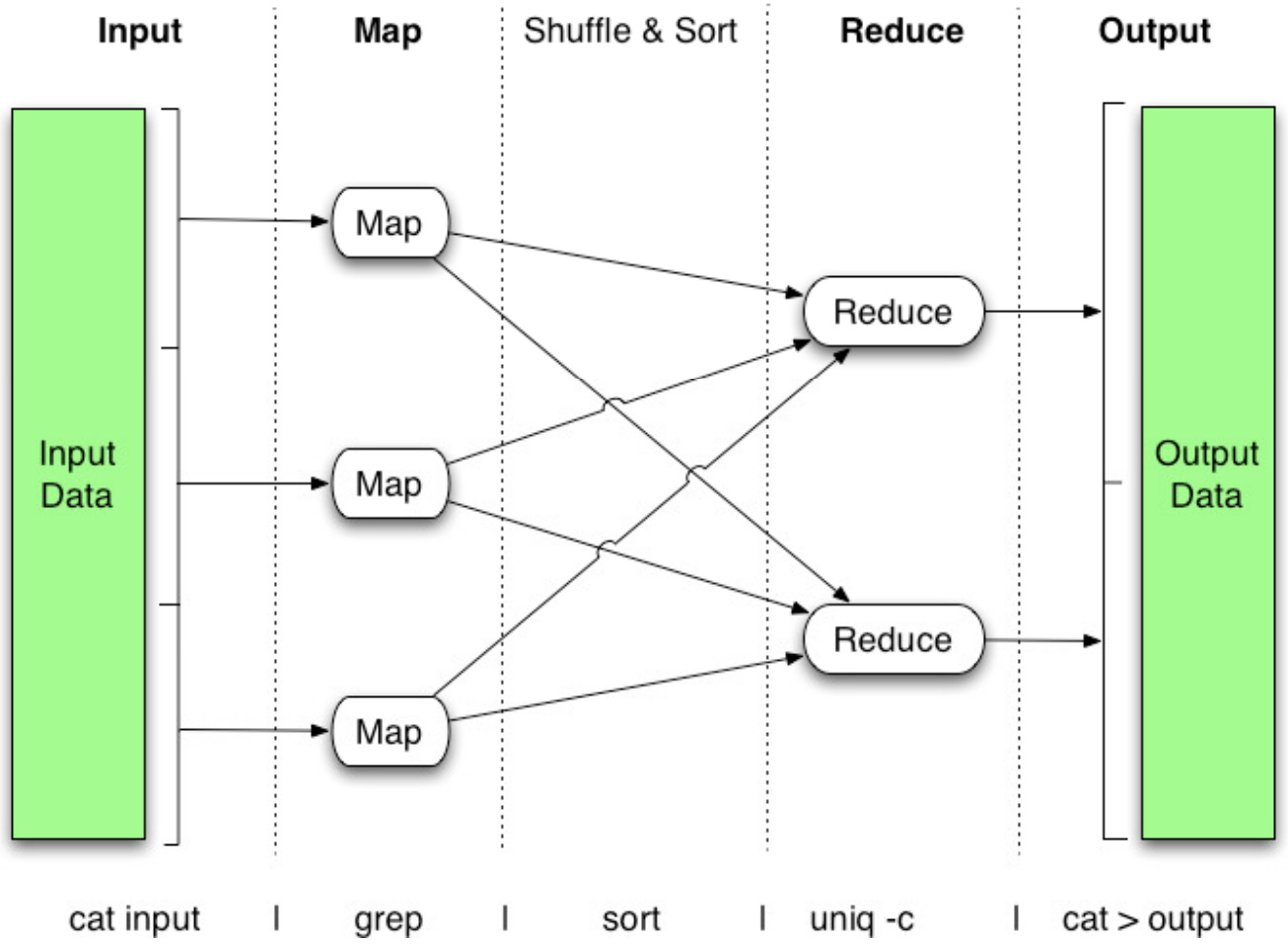
Hadoop Core

- A reliable, scalable, high performance distributed computing system
- Reliable storage layer
  - The Hadoop Distributed File System (HDFS)
  - With more sophisticated layers on top
- MapReduce – distributed computation framework
- Hadoop scales computation capacity, storage capacity, and I/O bandwidth by adding commodity servers.
- Divide-and-conquer using lots of commodity hardware

MapReduce

- MapReduce – distributed computation framework
  - Invented by Google researchers
- Two stages of a MR job
  - Map: {<Key,Value>} -> {<K',V'>}
  - Reduce: {<K',V'>} -> {<K'',V''>}
- Map – a truly distributed stage
  Reduce – an aggregation, may not be distributed
- Shuffle – sort and merge,
  transition from Map to Reduce
  invisible to user

MapReduce Workflow



Mean and Standard Deviation

- Mean $$\mu = \frac{1}{n}\sum_{1}^{n} x_i$$
- Standard deviation $$\sigma = \sqrt{\frac{1}{n}\sum_{1}^{n}(x_i - \mu)^2}$$

$$\sigma^2 = \frac{1}{n}\sum_{1}^{n} x_i^{\,2} - 2\mu\left(\frac{1}{n}\sum_{1}^{n} x_i\right) + \frac{1}{n}\sum_{1}^{n}\mu^2$$

$$\sigma^2 = \frac{1}{n}\sum_{1}^{n} x_i^{\,2} - \mu^2$$

Map Reduce Example: Mean and Standard Deviation

- Input: large text file
- Output: μ and σ

Mapper

- *Map input* is the set of words {w} in the partition
  - Key = Value = w
- *Map computes*
  - Number of words in the partition
  - Total length of the words ∑length(w)
  - The sum of length squares ∑length(w)^2
- Map output
  - <“count”, #words>
  - <“length”, #totalLength>
  - <“squared”, #sumLengthSquared>

Single Reducer

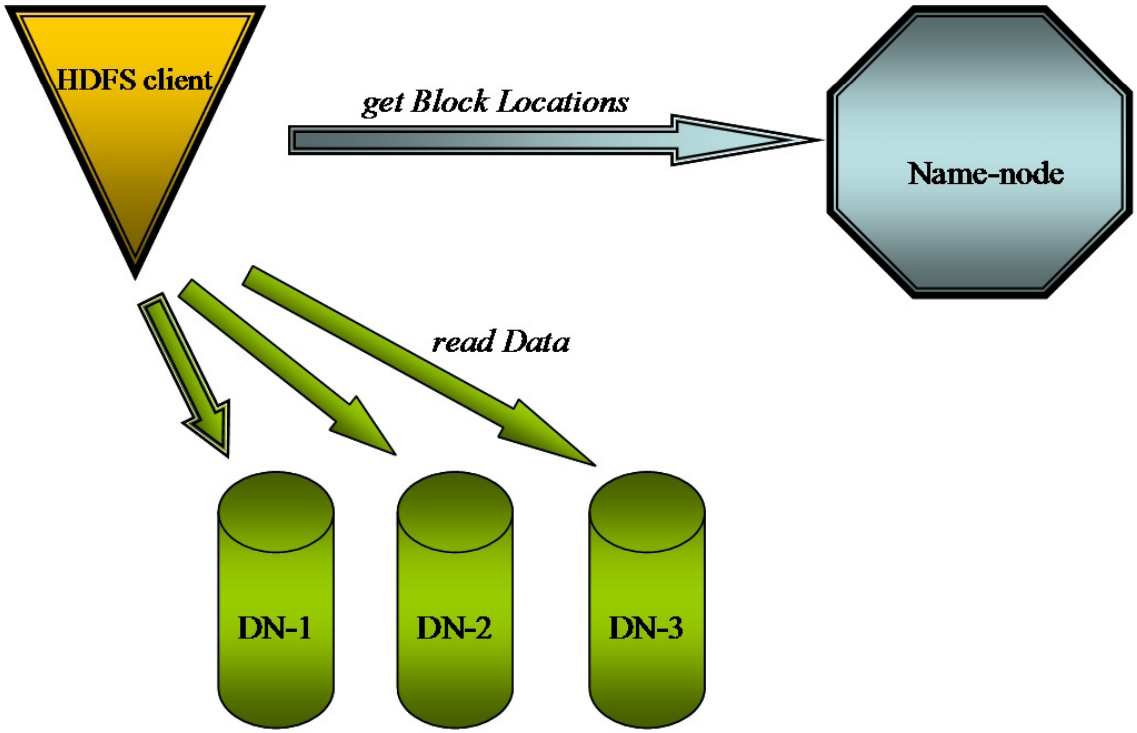
- *Reduce input*
  - {<key, value>}, where
  - key = “count”, “length”, “squared”
  - value is an integer
- *Reduce computes*
  - Total number of words: N = sum of all “count” values
  - Total length of words: L = sum of all “length” values
  - Sum of length squares: S = sum of all “squared” values
- *Reduce Output*
  - $\mu = L / N$
  - $\sigma = S / N - \mu^2$

Hadoop Distributed File System HDFS

- The name space is a hierarchy of files and directories
- Files are divided into blocks (typically 128 MB)
- Namespace (metadata) is decoupled from data
  - Lots of fast namespace operations, not slowed down by
  - Data streaming
- Single NameNode keeps the entire name space in RAM
- DataNodes store block replicas as files on local drives
- Blocks are replicated on 3 DataNodes for redundancy

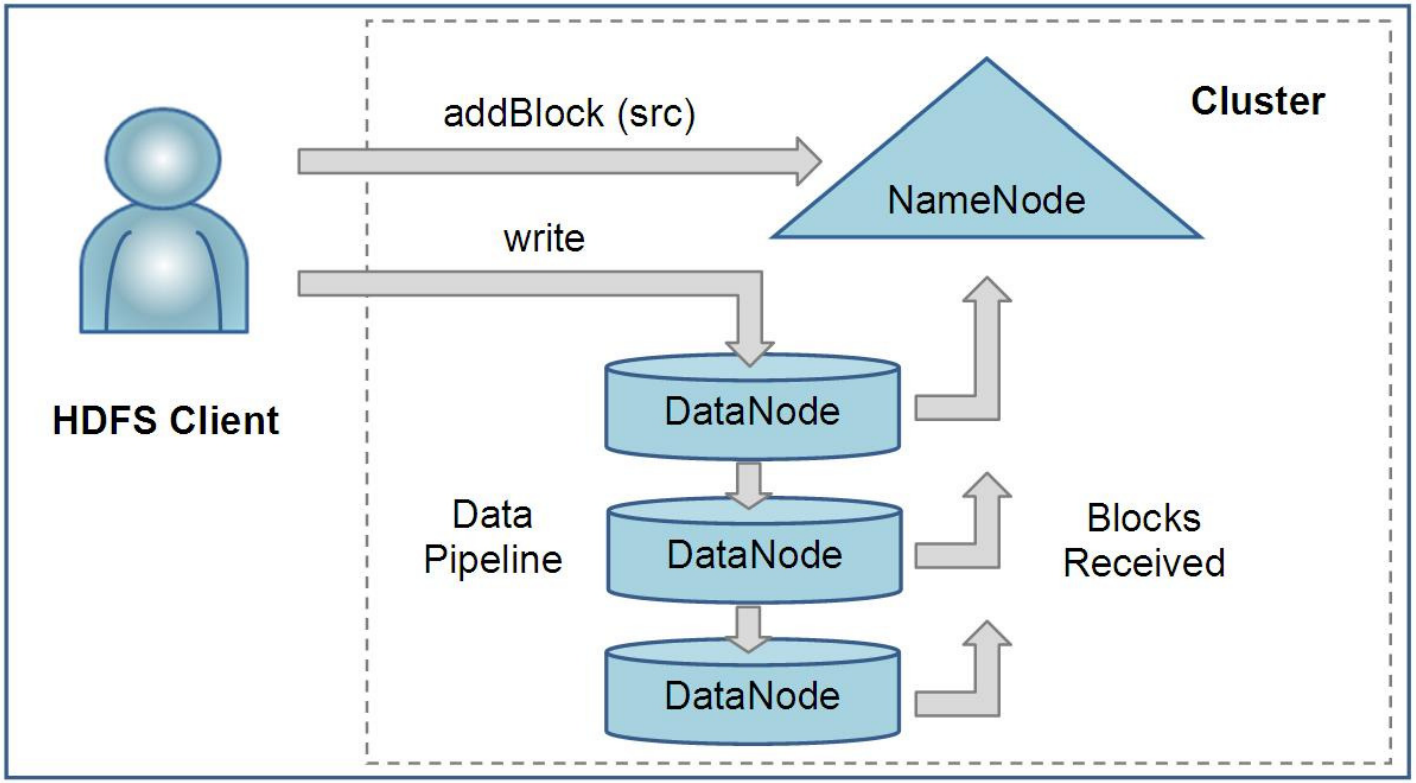
HDFS Read

- To read a block, the client requests the list of replica locations from the NameNode
- Then pulling data from a replica on one of the DataNodes



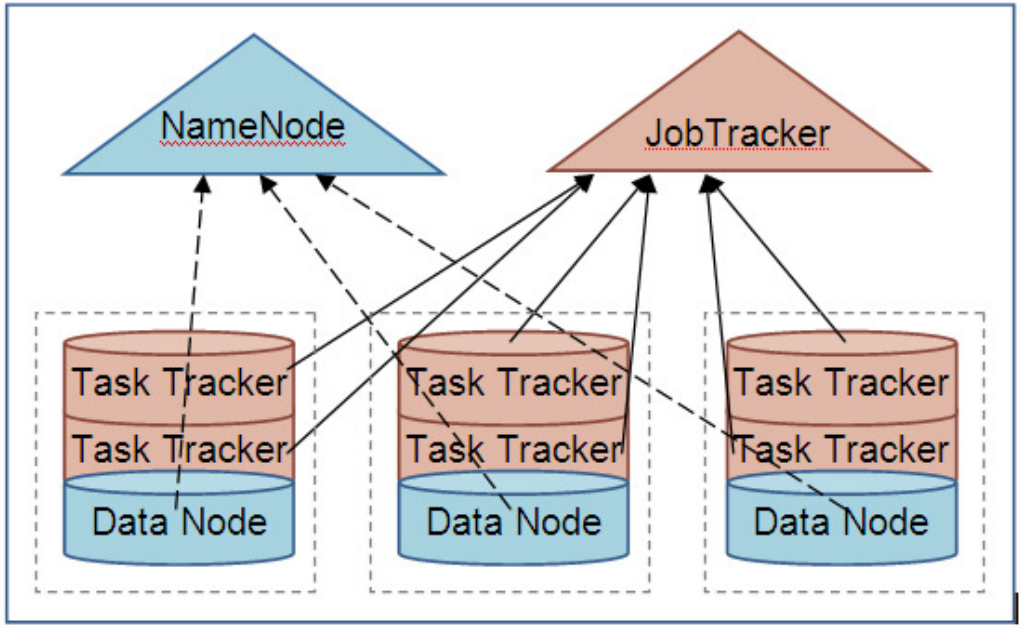
HDFS Write

- To write a block of a file, the client requests a list of candidate DataNodes from the NameNode, and organizes a write pipeline.



Replica Location Awareness

- MapReduce schedules a task assigned to process block B to a DataNode possessing a replica of B
- Data are large, programs are small
- Local access to data



Name Node

- NameNode keeps 3 types of information
  - Hierarchical namespace
  - Block manager: block to data-node mapping
  - List of DataNodes
- The durability of the name space is maintained by a write-ahead journal and checkpoints
  - A BackupNode creates periodic checkpoints
  - A journal transaction is guaranteed to be persisted before replying to the client
  - Block locations are not persisted, but rather discovered from DataNode during startup via block reports.

Data Nodes

- DataNodes register with the NameNode, and provide periodic block reports that list the block replicas on hand
- DataNodes send heartbeats to the NameNode
  - Heartbeat responses give instructions for managing replicas
- If no heartbeat is received during a 10-minute interval, the node is presumed to be lost, and the replicas hosted by that node to be unavailable
  - NameNode schedules re-replication of lost replicas

Quiz:
What Is the Common Attribute?

HDFS size

- Y! cluster
  - 70 million files, 80 million blocks
  - 15 PB capacity
  - 4000 nodes. 24,000 clients
  - 41 GB heap for NN
- Data warehouse Hadoop cluster at Facebook
  - 55 million files, 80 million blocks
  - 21 PB capacity
  - 2000 nodes. 30,000 clients
  - 57 GB heap for NN

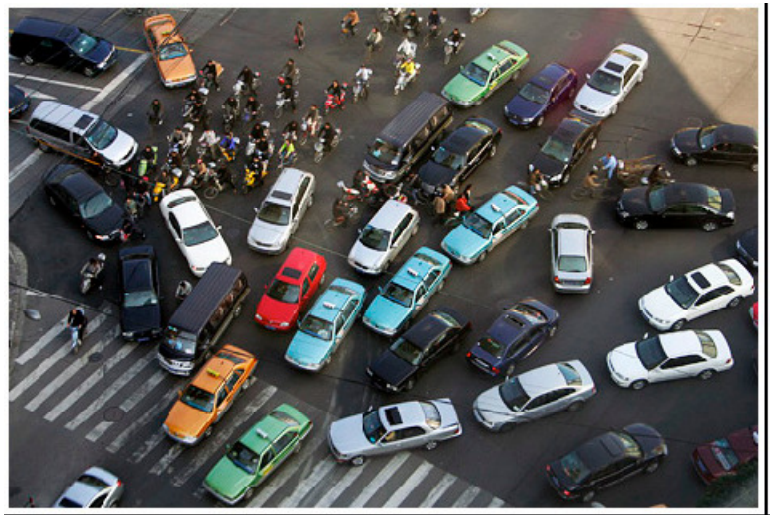
Benchmarks

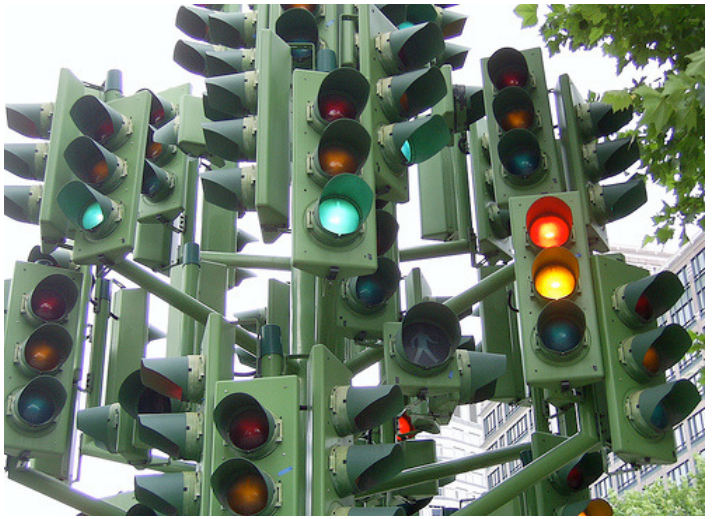
- **DFSIO**
  - Read: 66 MB/s
  - Write: 40 MB/s
- **Observed on busy cluster**
  - Read: 1.02 MB/s
  - Write: 1.09 MB/s
- **Sort ("Very carefully tuned user application")**

Bytes (TB)	Nodes	Maps	Reduces	Time	HDFS I/O Bytes/s	
					Aggregate (GB/s)	Per Node (MB/s)
1	1460	8000	2700	62 s	32	22.1
1000	3558	80,000	20,000	58,500 s	34.2	9.35

ZooKeeper

- A distributed coordination service for distributed apps
  - Event coordination and notification
  - Leader election
  - Distributed locking
- ZooKeeper can help build HA systems





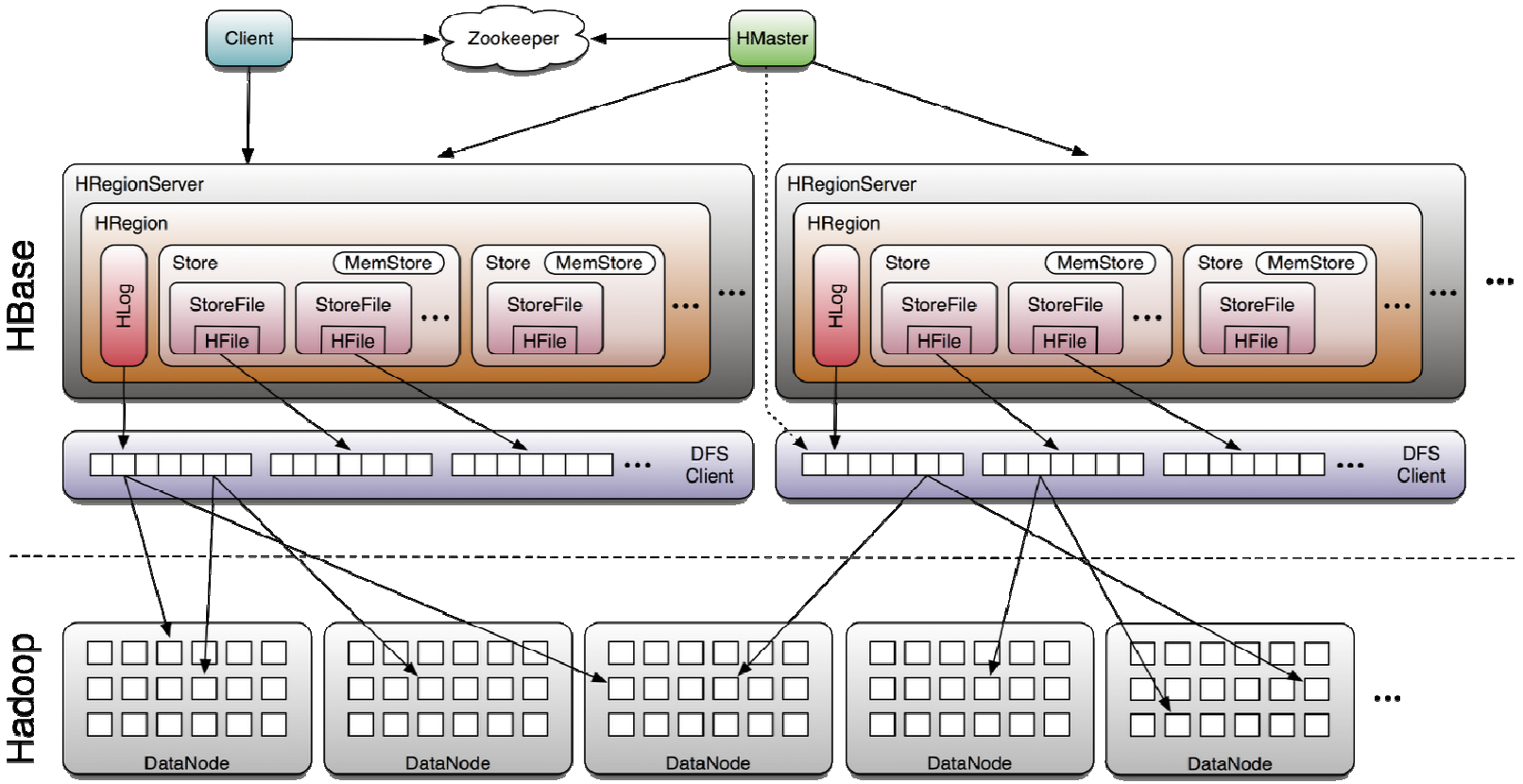
HBase

- Distributed table store on top of HDFS
  - An implementation of Googl's BigTable
- Big table is Big Data, cannot be stored on a single node
- Tables: big, sparse, loosely structured.
  - Consist of rows, having unique row keys
  - Has arbitrary number of columns,
  - grouped into small number of column families
  - Dynamic column creation
- Table is partitioned into regions
  - Horizontally across rows; vertically across column families
- HBase provides structured yet flexible access to data

HBase Functionality

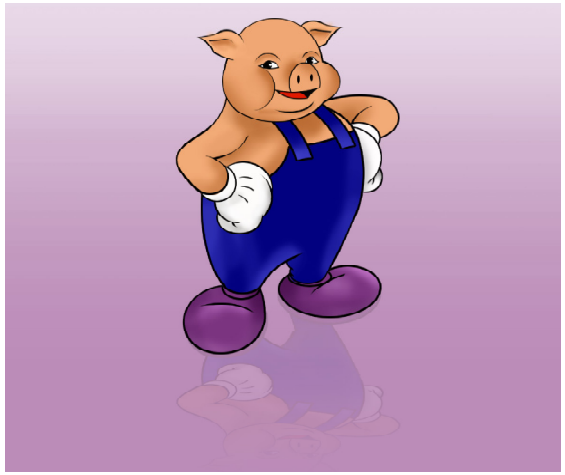
- HBaseAdmin: administrative functions
  - Create, delete, list tables
  - Create, update, delete columns, families
  - Split, compact, flush
- HTable: access table data
  - Result HTable.get(Get g) // get cells of a row
  - void HTable.put(Put p) // update a row
  - void HTable.put(Put[] p) // batch update of rows
  - void HTable.delete(Delete d) // delete cells/row
  - ResultScanner getScanner(family) // scan col family

HBase Architecture



Pig

- A language on top of and to simplify MapReduce
- Pig speaks Pig Latin
- SQL-like language
- Pig programs are translated into a series of MapReduce jobs



Hive

- Serves the same purpose as Pig
- Closely follows SQL standards
- Keeps metadata about Hive tables in MySQL DRBM

Hadoop User Groups