Inspire…Educate…Transform.

Engineering Big Data

Hadoop Ecosystem, HDFS

Dr. Sreerama KV Murthy
CEO, Quadratyx

July 25, 2015

Wake-Up Quiz

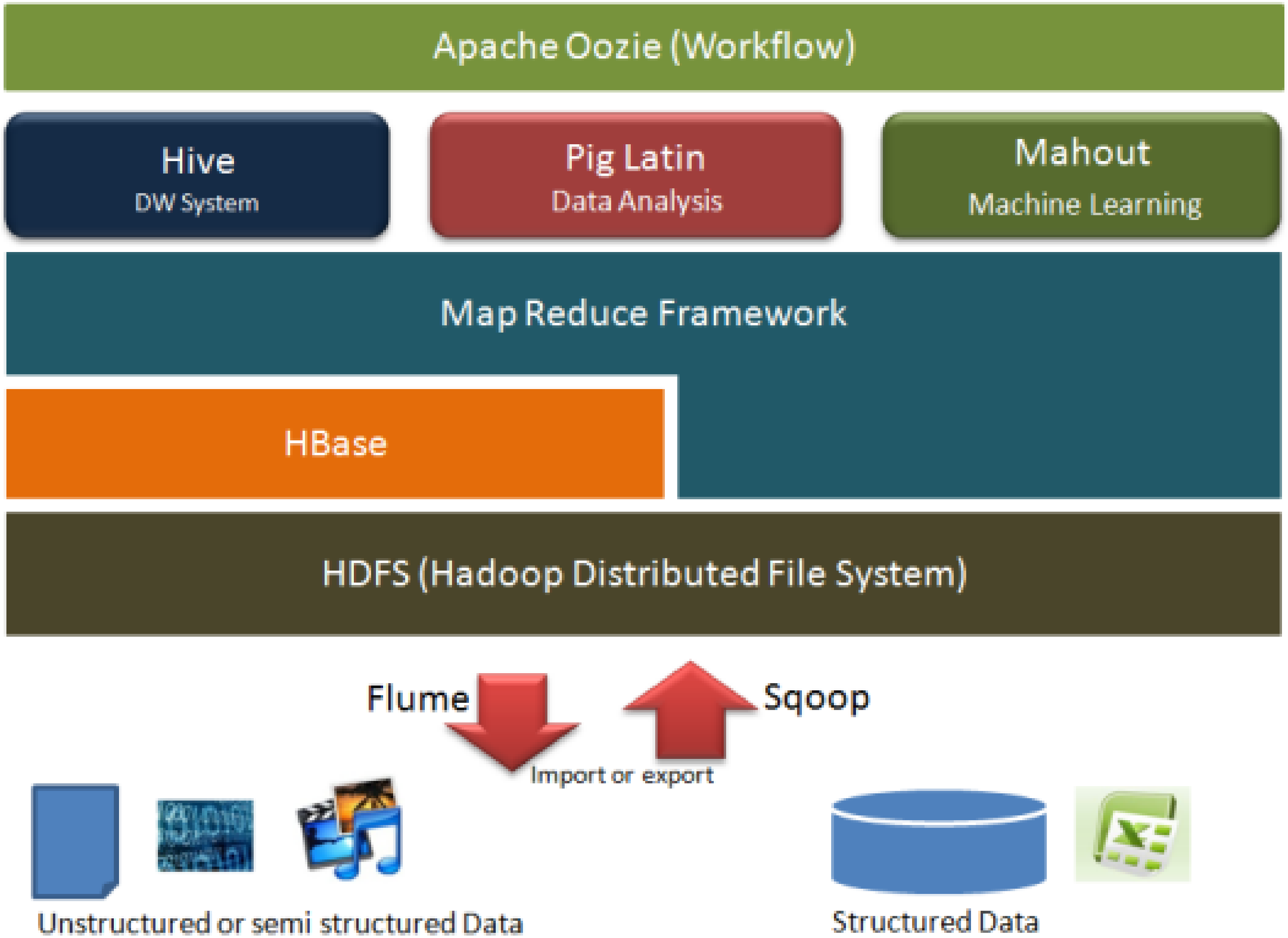
Hadoop Introduction

- **Open Source Apache Project**
  - http://hadoop.apache.org/
- **Written in Java**
  - Does work with other languages
- **Runs on**
  - Linux, Windows and more
  - Commodity hardware with high failure rate

Hadoop: 30,000 feet view

- Distribute data initially
  - Let processors / nodes work on local data
  - Minimize data transfer over network
  - Replicate data multiple times for increased availability
- Write applications at a high level
  - Programmers should not have to worry about network programming, temporal dependencies, low level infrastructure, etc
- Minimize talking between nodes (*share-nothing)*

Hadoop Ecosystem : Overview

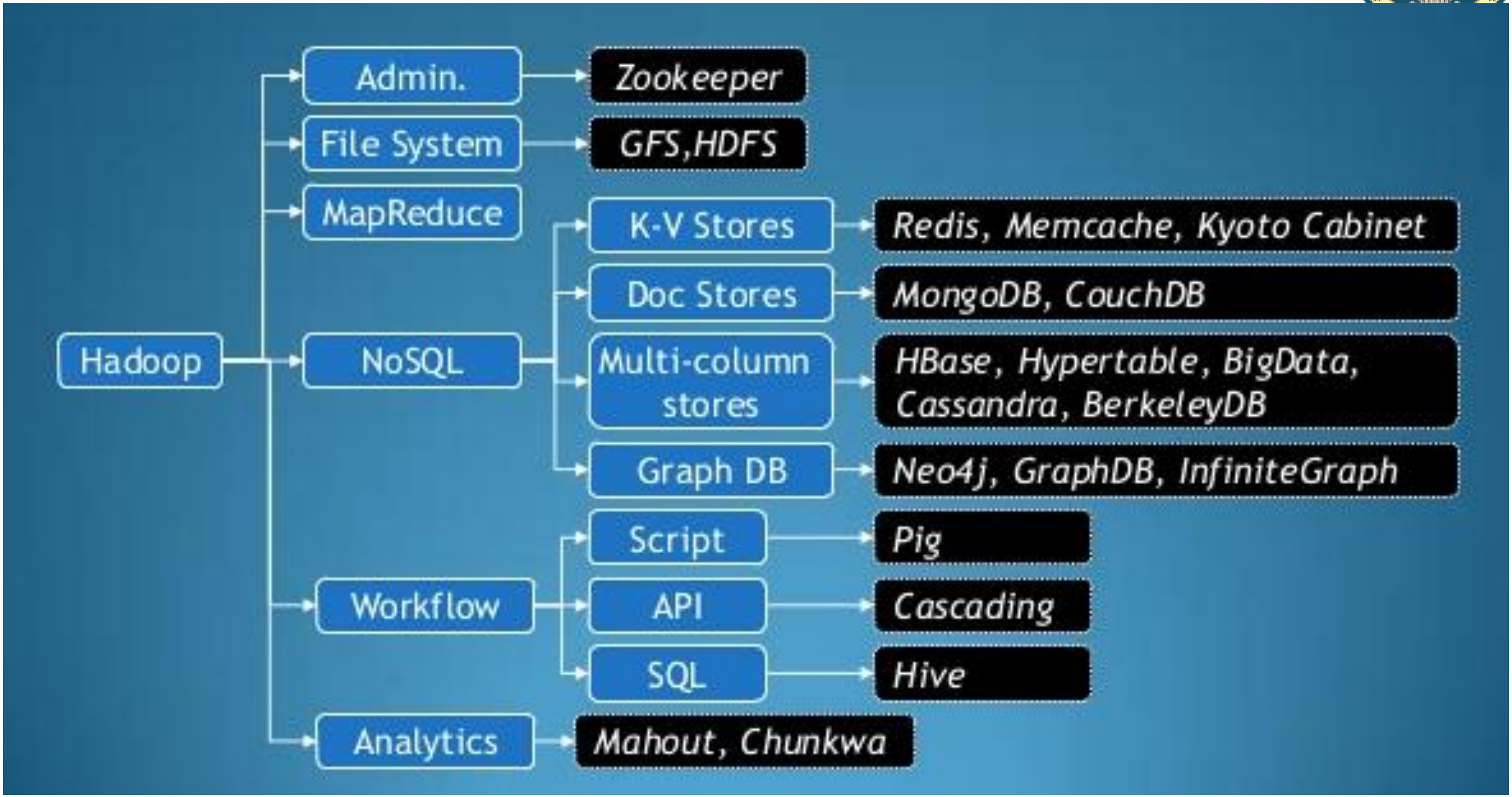


The Evolving Hadoop Landscape

Components		Description
mahout	R	Data Mining/machine learning tools used against Hadoop data to detect patterns and trends
Pig		Scripting language for analyzing large datasets. Compiles to MapReduce jobs
MapReduce (Hadoop 1.0)	YARN (Hadoop 2.0)	Programming model for processing large data sets. YARN performs overall resource mgmt
OOZIE		A workflow scheduler tool to manage Hadoop MapReduce jobs
Sqoop	HIVE	**Enable SQL for Hadoop data:** Sqoop - Data transfer between Hadoop and structured datastores. HIVE - data warehouse for Hadoop. Drill - open source, low latency SQL query engine for Hadoop and NoSQL.
APACHE DRILL		
ZooKeeper		Coordination of config. data, naming and synchronization of Hadoop projects

Components		Description
BigTop		Packaging services for Hadoop projects to ease testing and deployment
HBASE		A non-relational, distributed database that runs on top of HDFS
Thrift	AVRO	Schema-based data serialization system using RPC calls
Solr, elasticsearch	nutch	Indexing and search tools for data stored in HDFS for Hadoop
Kafka	Storm	Collect, aggregate, and move streaming data from multiple sources into Hadoop
Spark		AppDev tool for Hadoop apps combining batch, streaming, and interactive analytics
Ambari	chukwa	Monitoring & Management of Hadoop clusters and nodes

1


Hadoop
Admin.
Zookeeper
File System
GFS,HDFS
MapReduce
NoSQL
K-V Stores
Redis, Memcache, Kyoto Cabinet
Doc Stores
MongoDB, CouchDB
Multi-column stores
HBase, Hypertable, BigData, Cassandra, BerkeleyDB
Graph DB
Neo4j, GraphDB, InfiniteGraph
Workflow
Script
Pig
API
Cascading
SQL
Hive
Analytics
Mahout, Chunkwa

Hadoop Projects & Alternatives

Ingest/Propagate

Apache Flume Apache Kafka **Apache Sqoop**

Describe, Develop

Apache Crunch **Apache HCatalog Apache Hive Apache Pig**
Cascading Cloudera Hue DataFu **Dataguise** IBM Jaql

Compute, Search

Apache Blur Apache Giraph Apache Hama **Apache Lucene Apache MapReduce**
Apache SOLR Cloudera Impala HStreaming SQLstream Storm

Persist

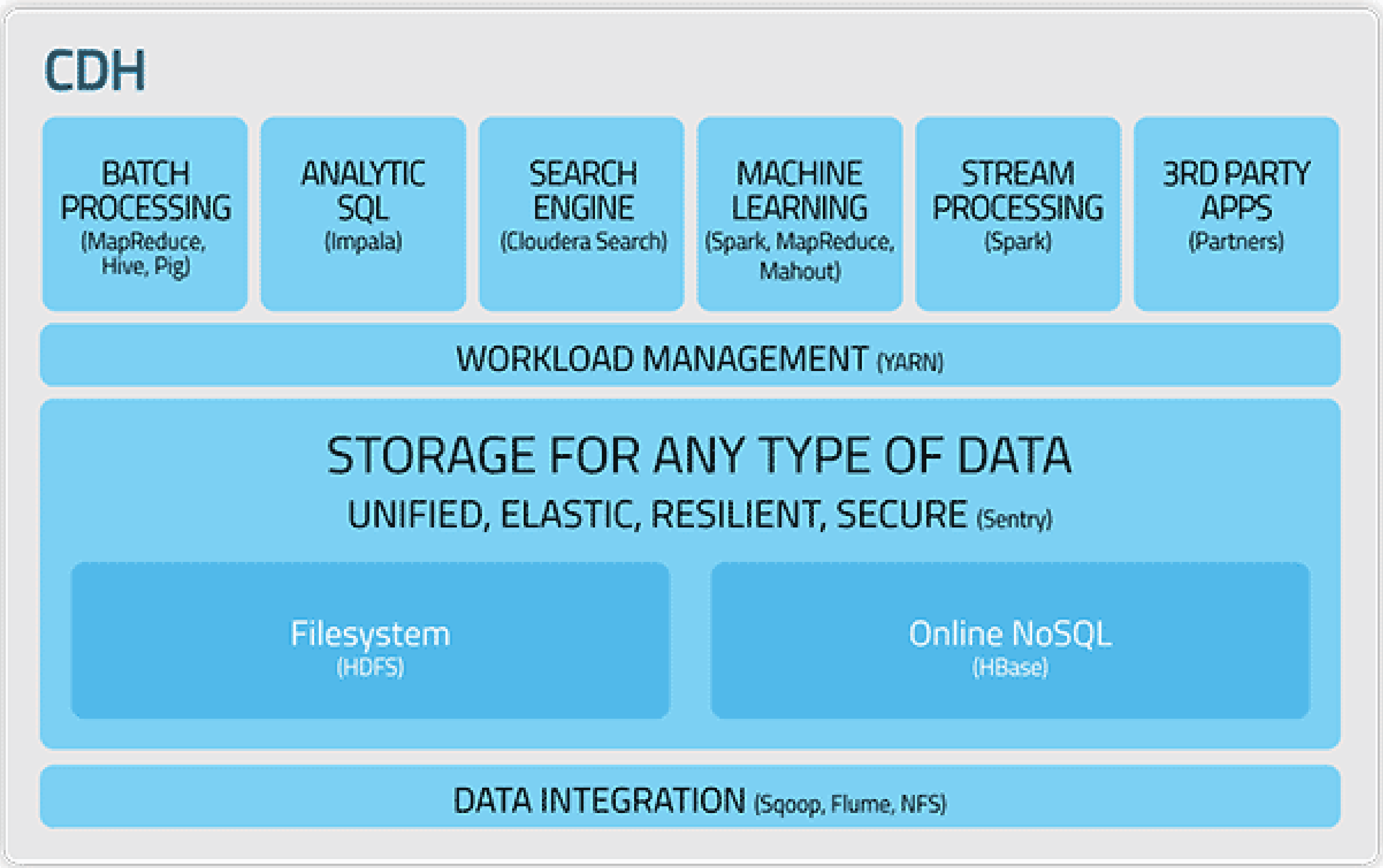
File System: Apache HDFS IBM GPFS **MapR Distributed File System**
Serialization: Apache Avro RCFile Sequencefile Text Trevni
DBMS: Apache Accumulo **Apache Cassandra Apache HBase** Hadapt Rainstor

Monitor, Administer

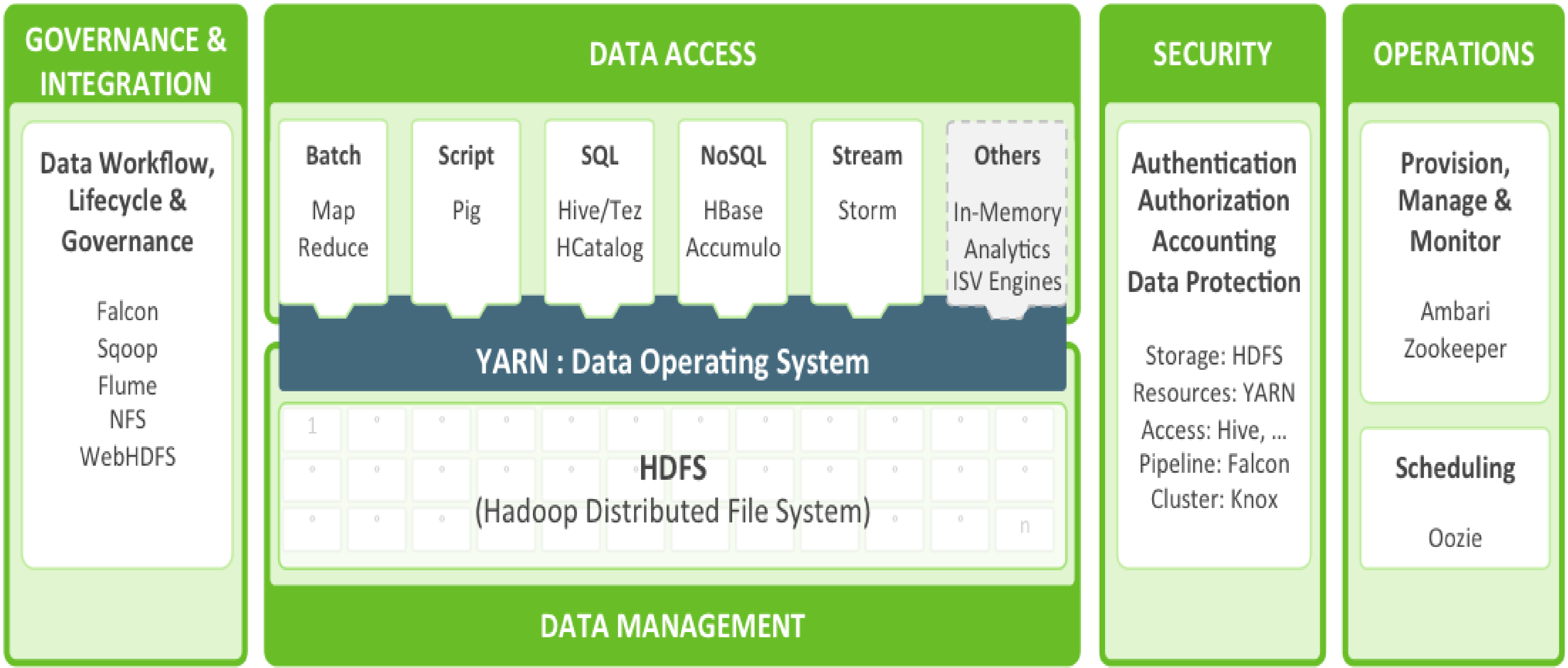
Apache Ambari Apache Bigtop Apache Chukwa **Apache Oozie** Apache Whirr
Apache Zookeeper Cloudera Manager Ganglia Nagios VMware Serengeti

Analytics, Machine Learning

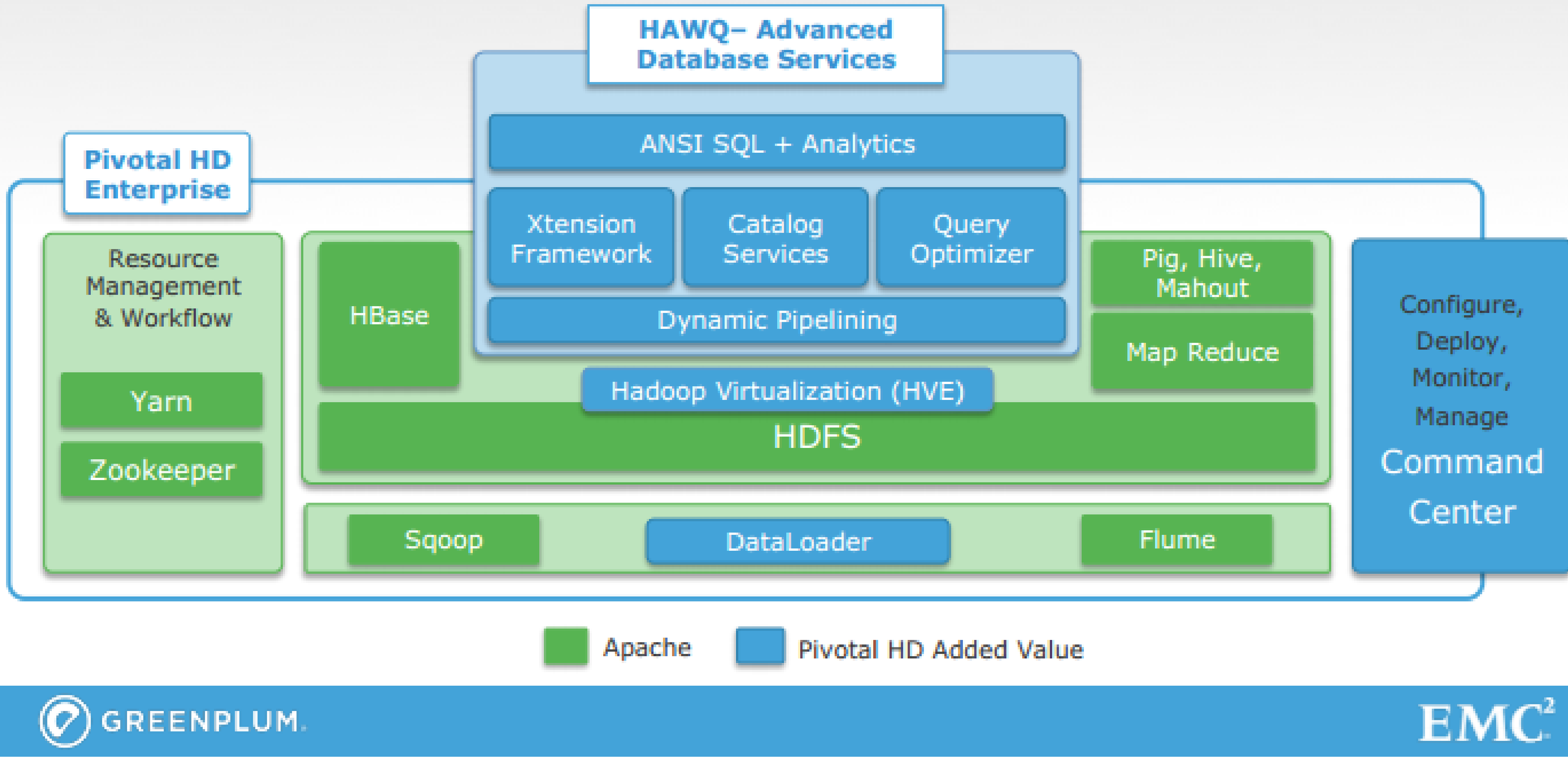
Apache Drill **Apache Mahout** Datameer IBM Big Sheets Karmasphere Platfora RHadoop

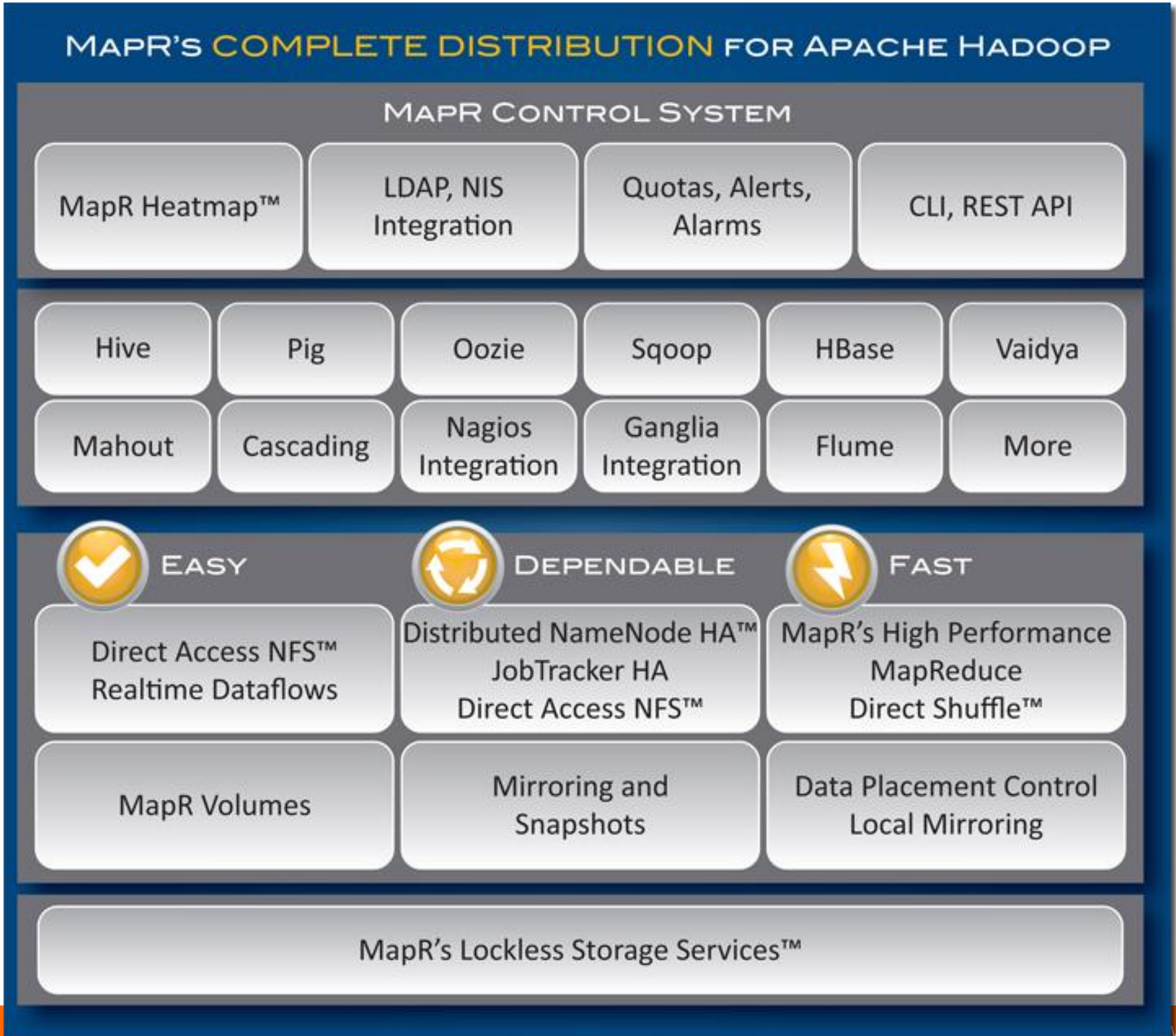

CDH
BATCH PROCESSING (MapReduce, Hive, Pig)
ANALYTIC SQL (Impala)
SEARCH ENGINE (Cloudera Search)
MACHINE LEARNING (Spark, MapReduce, Mahout)
STREAM PROCESSING (Spark)
3RD PARTY APPS (Partners)
WORKLOAD MANAGEMENT (YARN)
STORAGE FOR ANY TYPE OF DATA
UNIFIED, ELASTIC, RESILIENT, SECURE (Sentry)
Filesystem (HDFS)
Online NoSQL (HBase)
DATA INTEGRATION (Sqoop, Flume, NFS)

Hortonworks Data Platform

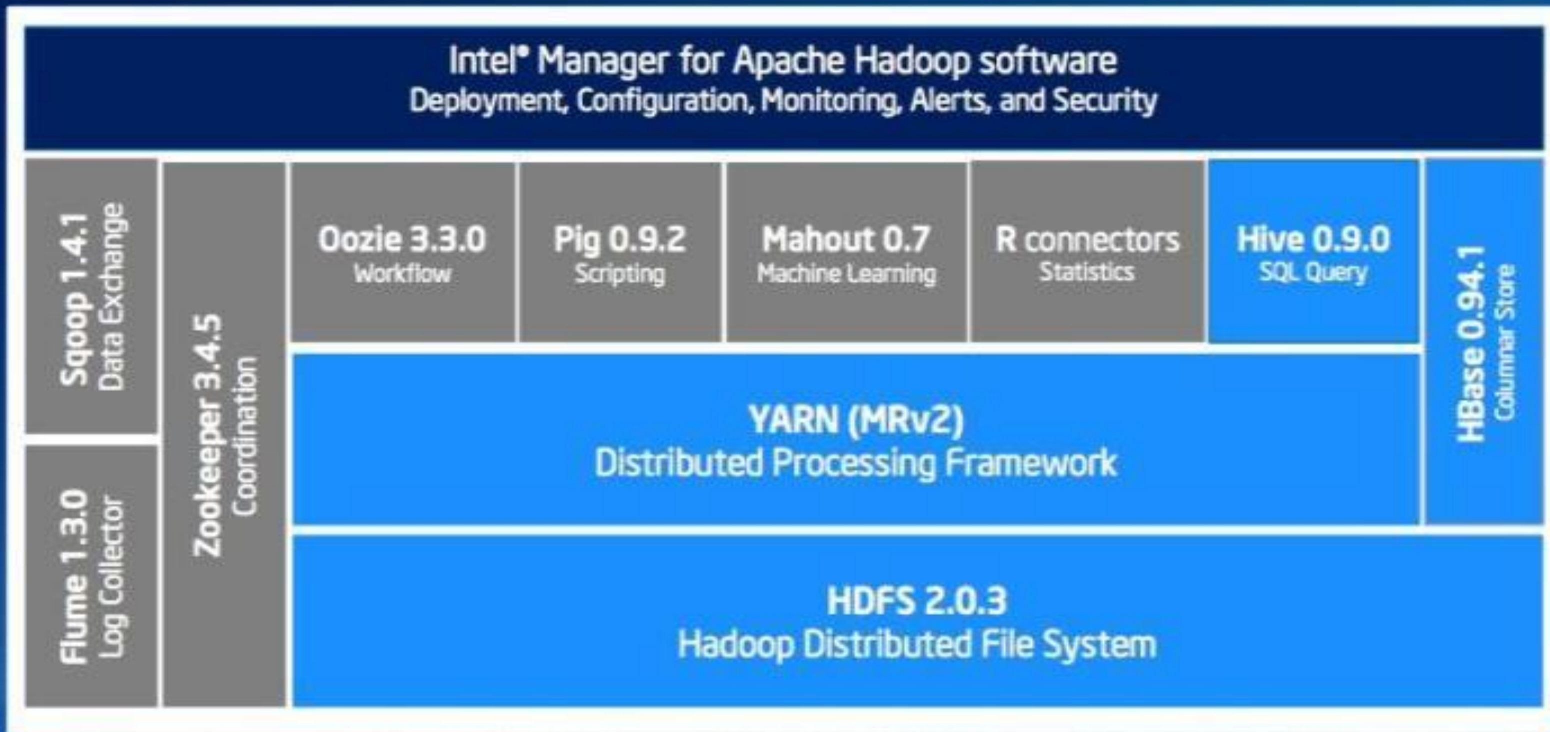


Pivotal HD Architecture




MapR's COMPLETE DISTRIBUTION FOR APACHE HADOOP
MapR CONTROL SYSTEM
MapR Heatmap™
LDAP, NIS Integration
Quotas, Alerts, Alarms
CLI, REST API
Hive
Pig
Oozie
Sqoop
HBase
Vaidya
Mahout
Cascading
Nagios Integration
Ganglia Integration
Flume
More
EASY
DEPENDABLE
FAST
Direct Access NFS™
Realtime Dataflows
Distributed NameNode HA™
JobTracker HA
Direct Access NFS™
MapR's High Performance MapReduce
Direct Shuffle™
MapR Volumes
Mirroring and Snapshots
Data Placement Control
Local Mirroring
MapR's Lockless Storage Services™

Intel® Distribution for Apache Hadoop* software



Intel proprietary

Intel enhancements contributed back to open source

Open source components included without change

All external names and brands are claimed as the property of others.

GFS & HDFS

Some slides adapted from Chris Hill's 2011 course slides at UMD.

GFS: Environment

Why did Google build its own file system?

- Google has unique FS requirements
  - huge volume of data
  - huge read/write bandwidth
  - reliability over tens of thousands of nodes with frequent failures
  - mostly operating on large data blocks
  - needs efficient distributed operations
- Google has somewhat of an unfair advantage…it has control over, and customizes, its:
  - applications
  - libraries
  - operating system
  - networks
  - even its computers!

Google File System (GFS) is unique even among DFSs

NFS, etc.

GFS

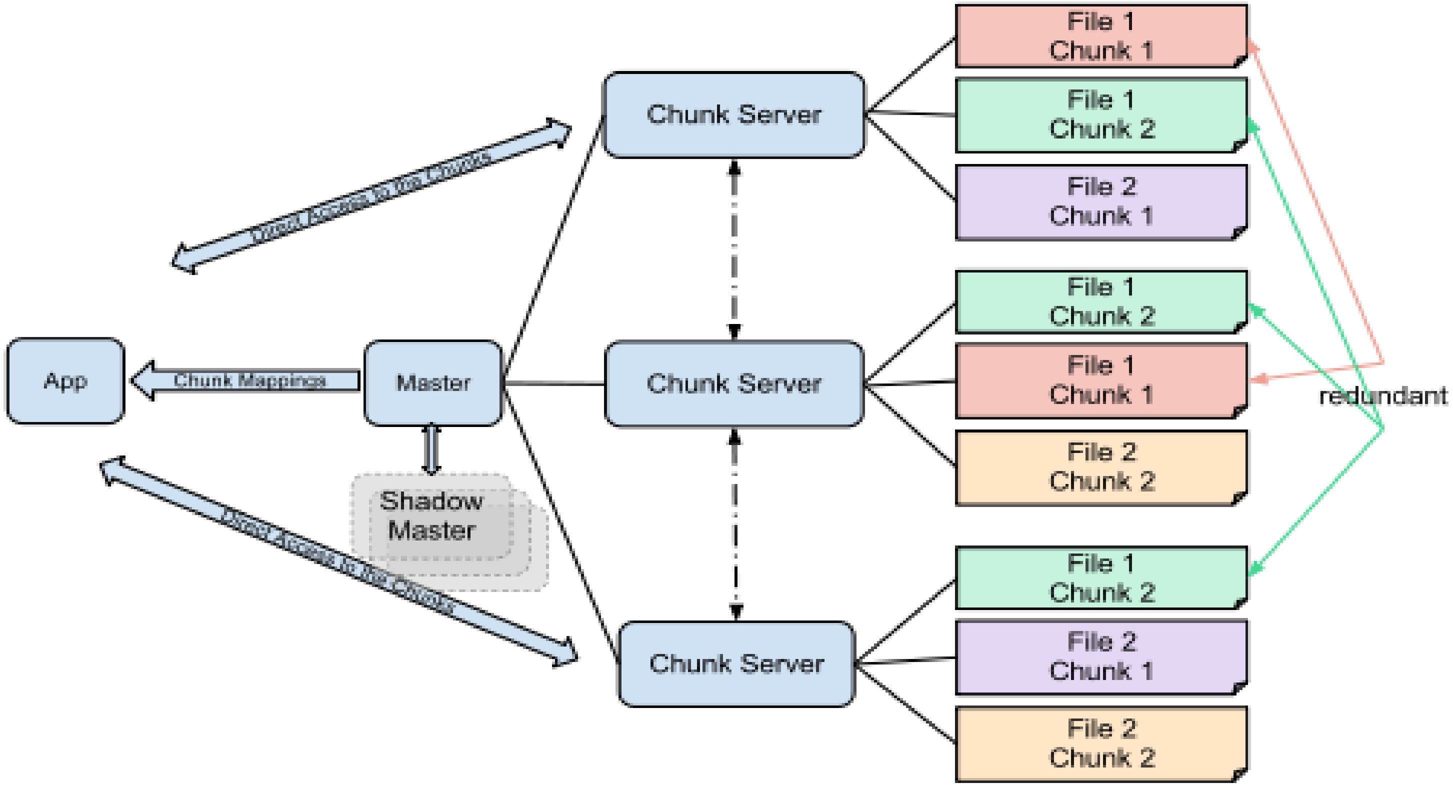
Independence
Small Scale
Variety of workloads

Cooperation
Large scale
Very specific, well-understood workloads

Google File System

- **High** component failure rates
  - Inexpensive commodity components fail all the time
- "Modest" number of HUGE files
  - Just a few million
  - Each is 100MB or larger; multi-GB files typical
- Files are write-once, mostly appended to
  - Perhaps concurrently
- Large streaming reads
- High sustained throughput favored over low latency

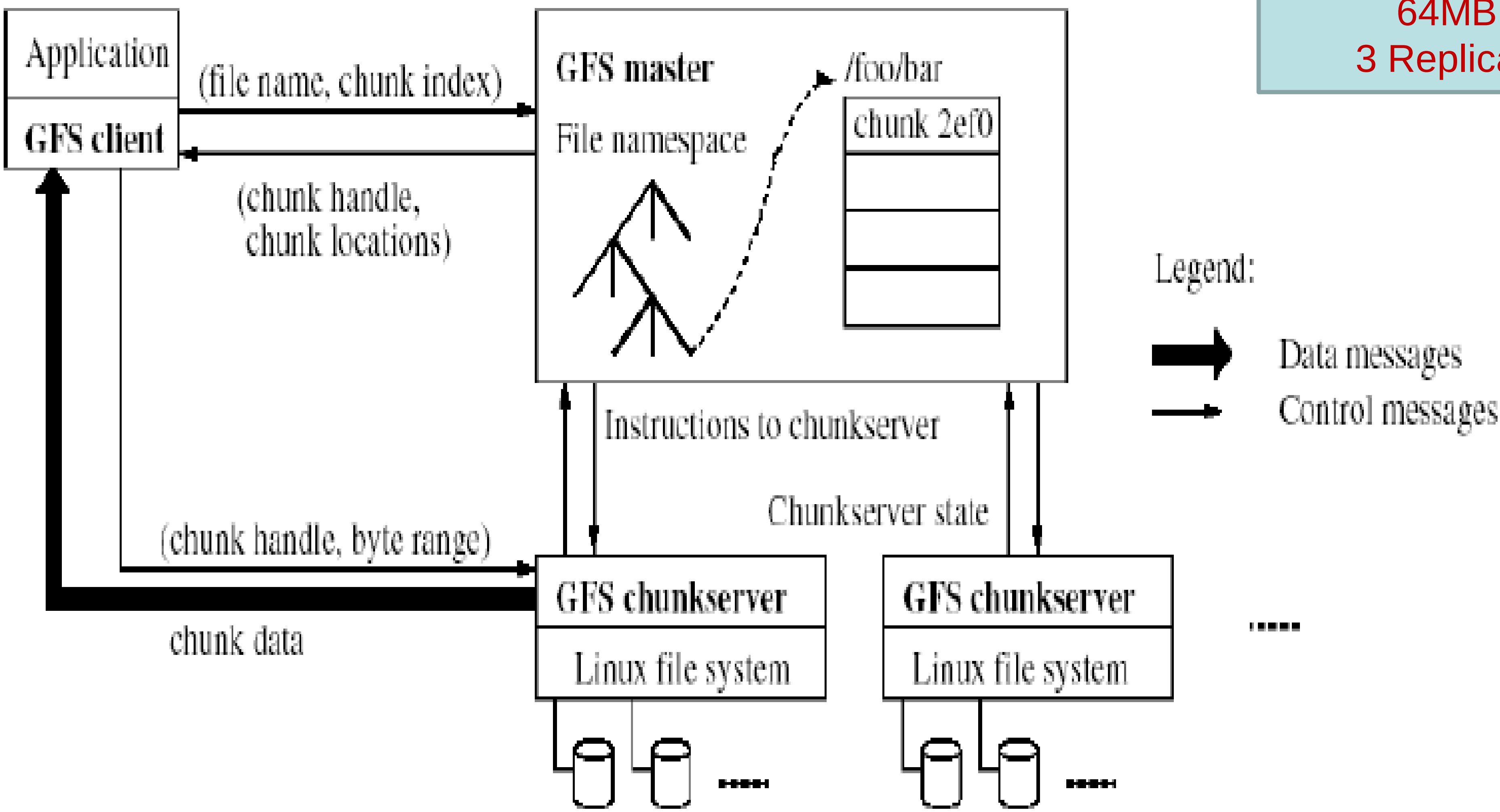
GFS: Architecture



- Masters manage metadata (naming, chunk location, etc.)
- Data transfers happen directly between clients/chunkservers
- Files are broken into chunks (typically 64 MB)
    - each chunk replicated on 3 chunkservers
- Clients do not cache data!

GFS Architecture – Close-up

64MB
3 Replicas



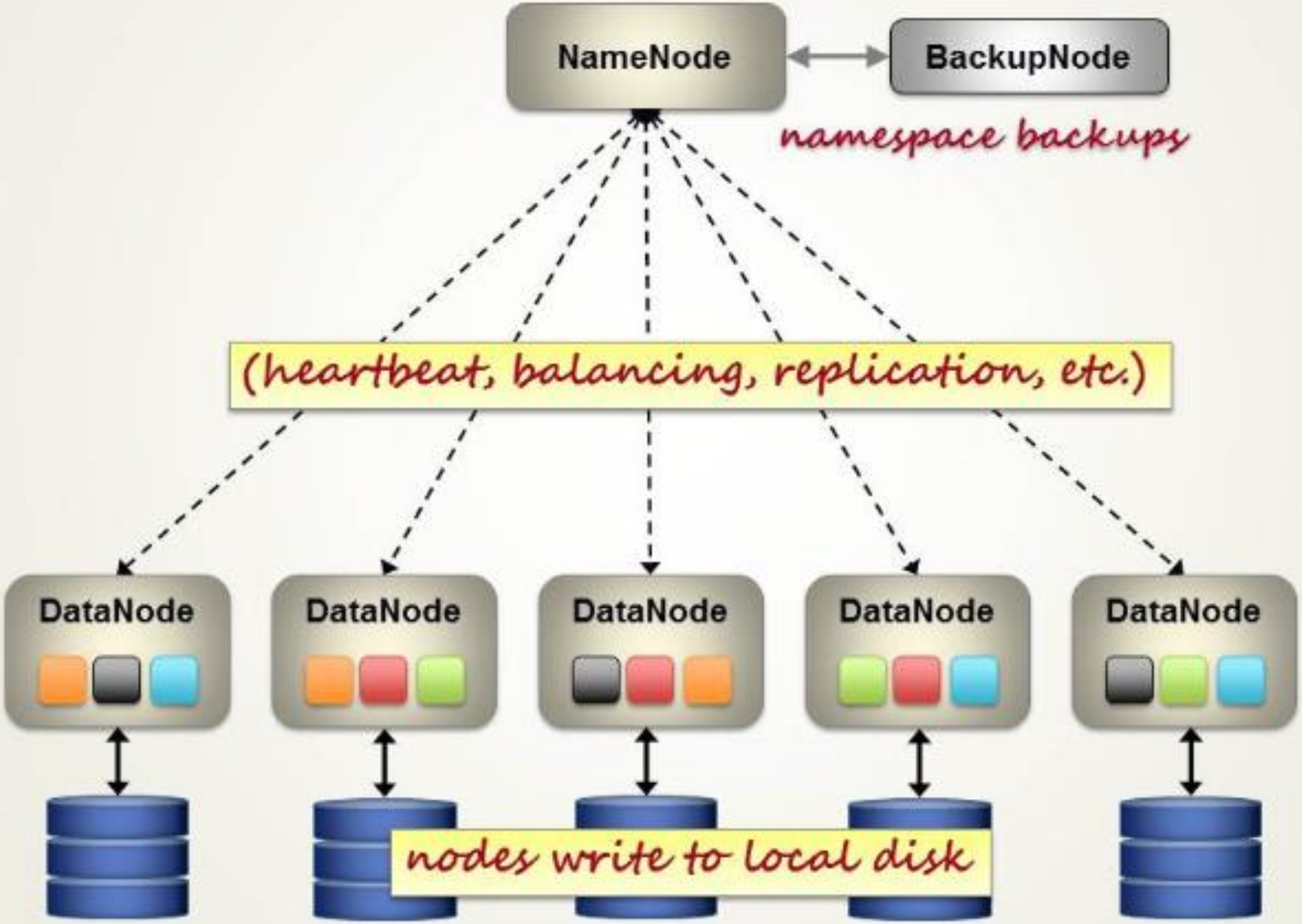
Master's Responsibilities

- Metadata storage
- Namespace management/locking
- Periodic communication with chunkservers
  - give instructions, collect state, track cluster health
- Chunk creation, re-replication, rebalancing
  - balance space utilization and access speed
  - spread replicas across racks to reduce correlated failures
  - re-replicate data if redundancy falls below threshold
  - rebalance data to smooth out storage and request load

Master's Responsibilities (contd.)

- Garbage Collection
  - simpler, more reliable than traditional file delete
  - master logs the deletion, renames the file to a hidden name
  - lazily garbage collects hidden files
- Stale replica deletion
  - detect "stale" replicas using chunk version numbers

HDFS Architecture



22

Ref: www.ProjectM80.com

HDFS* vs. GFS

- Namenode ***(Master)***
  - Metadata:
    - Where file blocks are stored (namespace image)
    - Edit ***(Operation)*** log
  - Secondary namenode ***(Shadow master)***
- Datanode ***(Chunkserver)***
  - Stores and retrieves blocks
    - …by client or namenode.
  - Reports to namenode with list of blocks they are storing

*CDH3

HDFS vs. GFS (contd.)

- Only single-writers per file.
  - No ***record append*** operation.
- Open source
  - Provides *many* interfaces and libraries for different file systems.
    - S3, KFS, etc.
    - Thrift (C++, Python, ...), *libhdfs* (C), FUSE

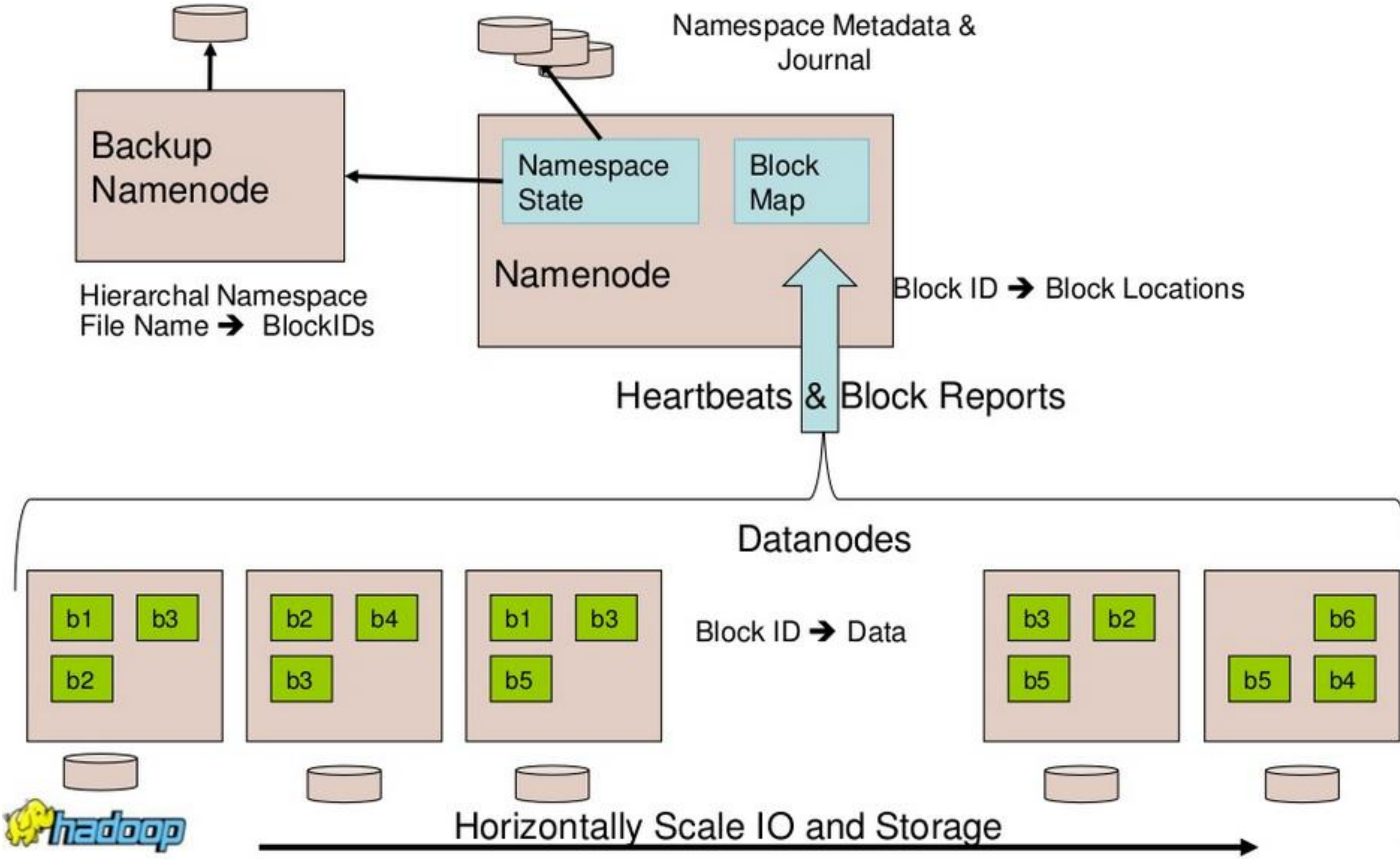
NameNode Metadata

- Metadata in Memory
  - The entire metadata is in main memory
  - No demand paging of metadata
- Types of metadata
  - List of files
  - List of Blocks for each file
  - List of DataNodes for each block
  - File attributes, e.g. creation time, replication factor
- A Transaction Log
  - Records file creations, file deletions etc

The Problem: Single Master

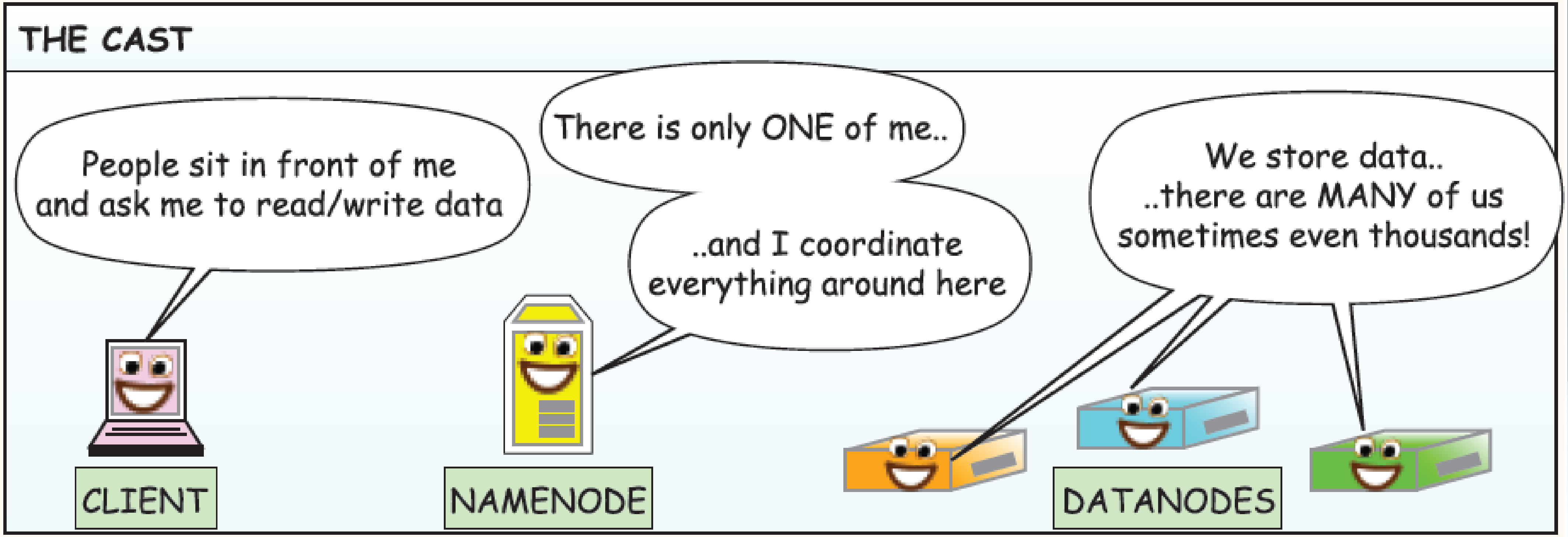
- Problem:
  - **Single** point of failure
  - Scalability bottleneck
- GFS solutions:
  - *Shadow* masters
  - Minimize master involvement
    - **never** move data through it, use only for metadata
      - and cache metadata at clients
    - large chunk size
    - master delegates authority to primary replicas in data mutations (chunk leases)
- Simple, and good enough for Google's concerns

Recap: CDH3 HDFS Architecture



Reference: Apache Hadoop India Summit 2011, Sanjay Radia talk

See the Cartoon!



Uploaded to Piazza.

Block Placement

- Current Strategy
  - One replica on local node
  - Second replica on a remote rack
  - Third replica on same remote rack
  - Additional replicas are randomly placed
- Clients read from nearest replicas
- Would like to make this policy pluggable

Heartbeats

- DataNodes send hearbeat to the NameNode
  - Once every 3 seconds
- NameNode uses heartbeats to detect DataNode failure

Replication Engine

- NameNode detects DataNode failures
  - Chooses new DataNodes for new replicas
  - Balances disk usage
  - Balances communication traffic to DataNodes

Rebalancer

- Goal: % disk full on DataNodes should be similar
  - Usually run when new DataNodes are added
  - Cluster is online when Rebalancer is active
  - Rebalancer is throttled to avoid network congestion
  - Command line tool

Inspire...Educate...Transform.

International School of Engineering

Plot 63/A, 1st Floor, Road # 13, Film Nagar, Jubilee Hills, Hyderabad - 500 033

For Individuals:	+91-9502334561/63 or 040-65743991
For Corporates:	+91-9618483483
Web:	http://www.insofe.edu.in
Facebook:	https://www.facebook.com/insofe
Twitter:	https://twitter.com/Insofeedu
YouTube:	http://www.youtube.com/InsofeVideos
SlideShare:	http://www.slideshare.net/INSOFE
LinkedIn:	http://www.linkedin.com/company/international-school-of-engineering