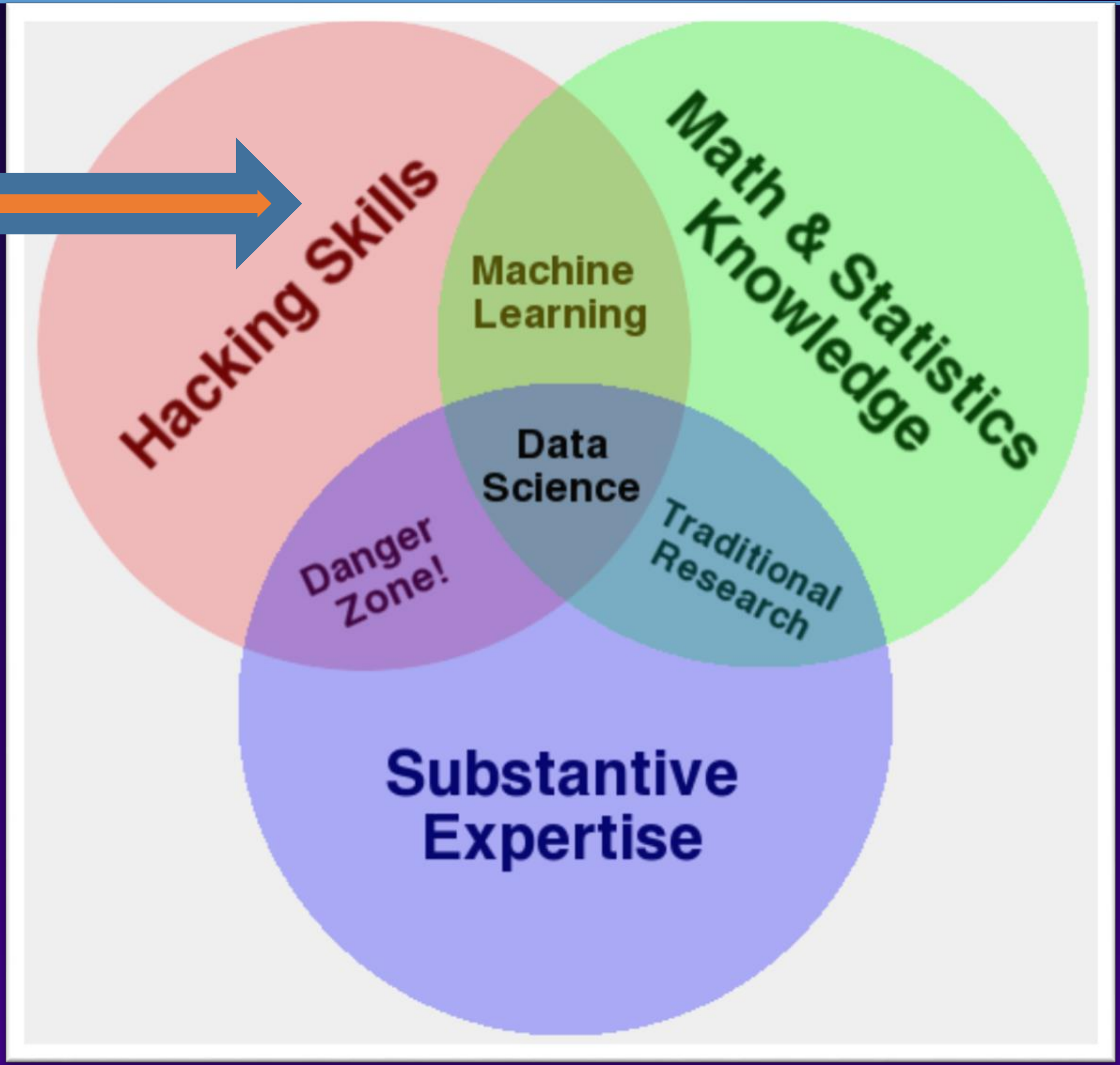
DATA MINING WITH HADOOP AND HIVE
Introduction to Architecture

Dr. Wlodek Zadrozny

(Most slides come from Prof. Akella's class in 2014)

Data Science

Hadoop
& Hive



Source: http://www.dataists.com/2010/09/the-data-science-venn-diagram/

Hadoop, Map-reduce, Hive, …

- A few slides today (with some updates by WZ).
- Full PDF of Prof. Akella’s slides on Moodle (104 slides)
- You’ll use it in your projects
- We’ll review and expand in future lectures (time permitting)

Scalable and Distributed Data Storage and Analysis

Srinivas Akella
Computer Science
UNC Charlotte

DSBA 6100: Big Data Analytics for Competitive Advantage
February 19, 2014

MapReduce and Hadoop

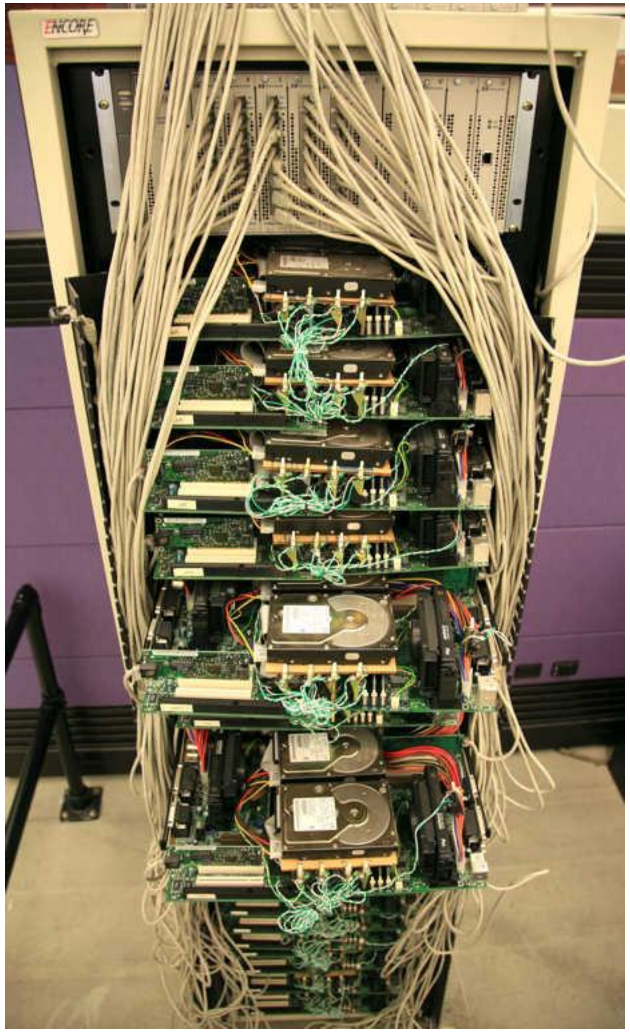
MapReduce

- MapReduce programming paradigm for clusters of commodity PCs
- Map computation across many inputs

- Fault-tolerant
- Scalable

- Machine independent programming model
- Permits programming at abstract level
- Runtime system handles scheduling, load balancing

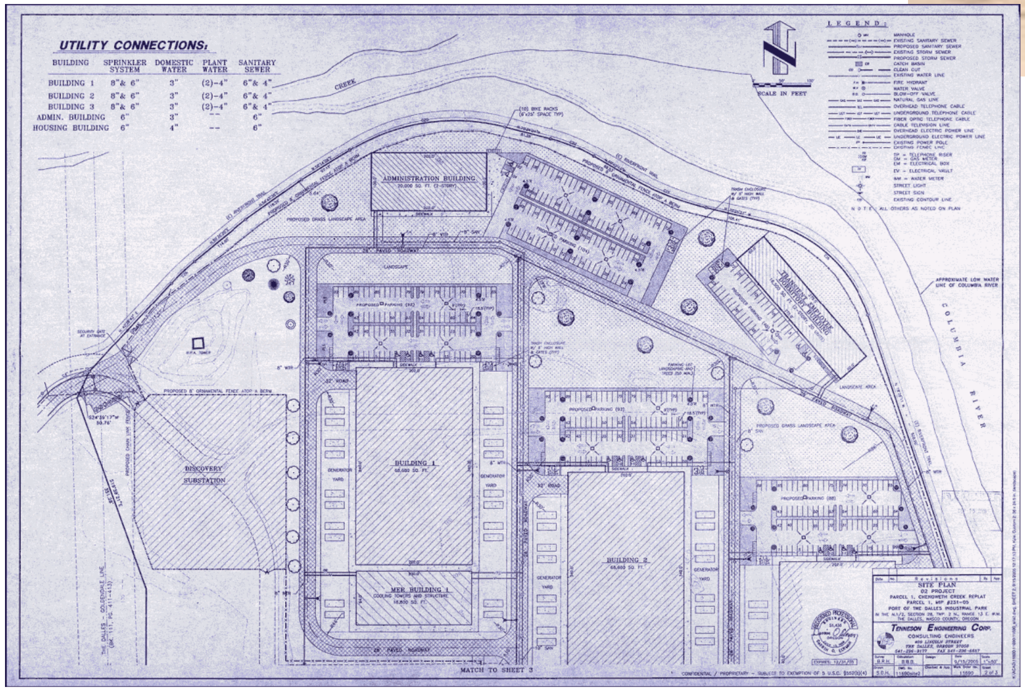
- First Google server
  ~1999



Computer History Museum

Data Centers

Yahoo data center



Google data center layout

Harpers, 3/2008

Motivation: Large Scale Data Processing

- Many tasks: Process lots of data to produce other data
- Want to use hundreds or thousands of CPUs

  ... but this needs to be easy

- MapReduce provides:
  - Automatic parallelization and distribution
  - Fault-tolerance
  - I/O scheduling
  - Status and monitoring

Example Tasks

- Finding all occurrences of a string on the web
- Finding all pages that point to a given page
- Data analysis of website access log files
- Clustering web pages

Functional Programming

- MapReduce: Based on Functional Programming paradigm that treats computation as evaluation of math functions

- **Map**
- map *result-type function sequence* &rest *more-sequences*
- The *function* must take as many arguments as there are sequences provided; at least one sequence must be provided. The result of map is a sequence such that element *j* is the result of applying *function* to element *j* of each of the argument sequences.
- Example: (map 'list #'- '(1 2 3 4)) => (-1 -2 -3 -4)

- **Reduce**
- reduce *function sequence* &key :from-end :start :end :initial-value
- The reduce function combines all the elements of a sequence using a binary operation; for example, using + one can add up all the elements.
- Example: (reduce #'+ '(1 2 3 4)) => 10

MapReduce Programming Model

- Input and Output: each a set of key/value pairs

- Programmer specifies two functions:

- map (in_key, in_value) -> list(out_key, intermediate_value)
  - Processes input key/value pair
  - Produces set of intermediate pairs

- reduce (out_key, list(intermediate_value)) -> list(out_value)
  - Combines all intermediate values for a particular key
  - Produces a set of merged output values (usually just one)

- Inspired by similar primitives in LISP and other languages

Example: Count word occurrences

```
map(String key, String value):
    // key: document name
    // value: document contents
    for each word w in value:
        EmitIntermediate(w, "1");

reduce(String key, Iterator values):
    // key: a word
    // values: a list of counts
    int result = 0;
    for each v in values:
        result += ParseInt(v);
    Emit(AsString(result));
```

MapReduce Operations

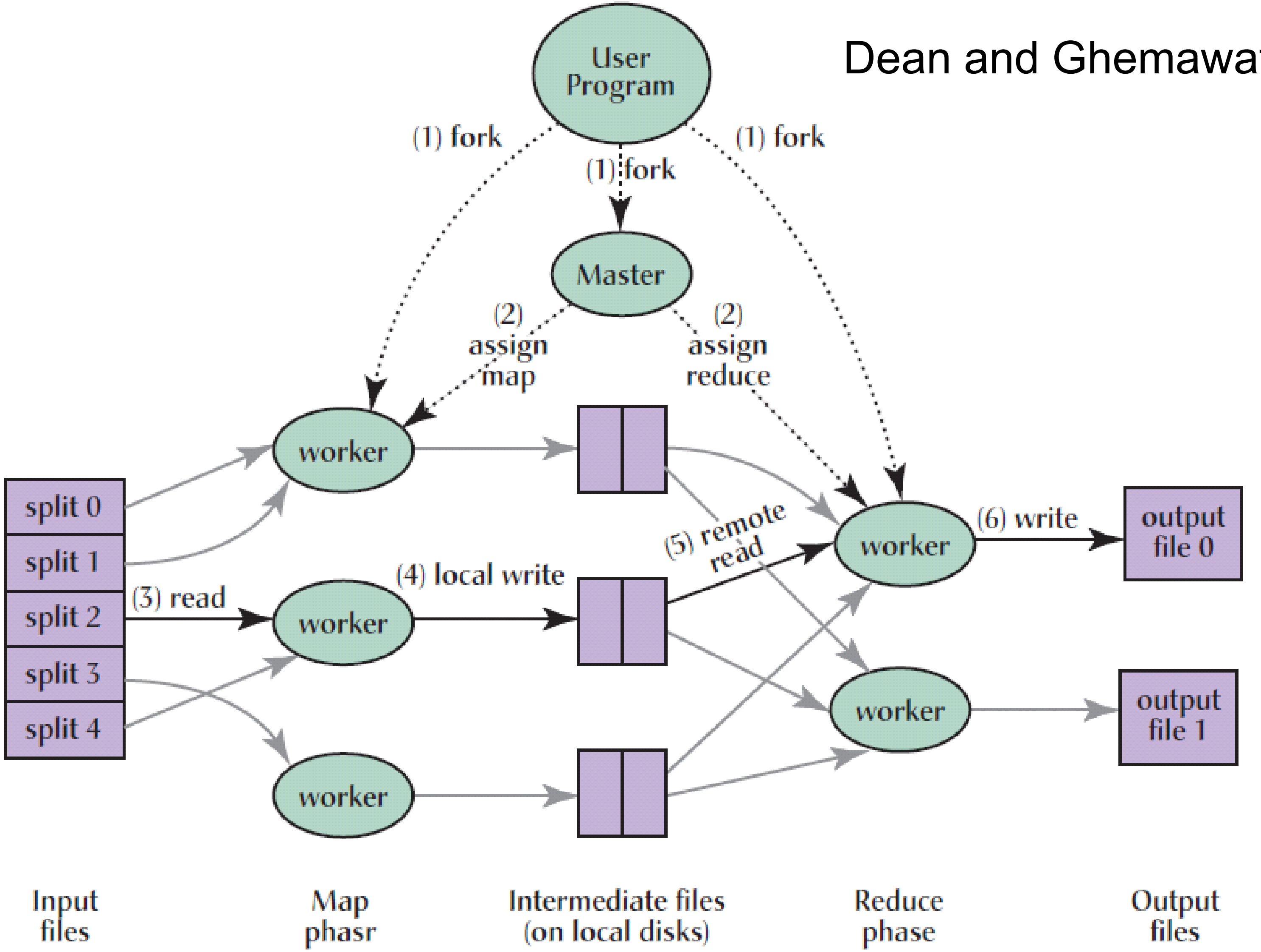
- Conceptual:
  - Map: (K1, V1) -> list(K2, V2)
  - Reduce: (K2, list(V2)) -> list(K3, V3)

- WordCount example:
  - Map: (doc, contents) -> list(word_i, 1)
  - Reduce:

    (word_i, list(1,1,…)) -> list(word_i, count_i)

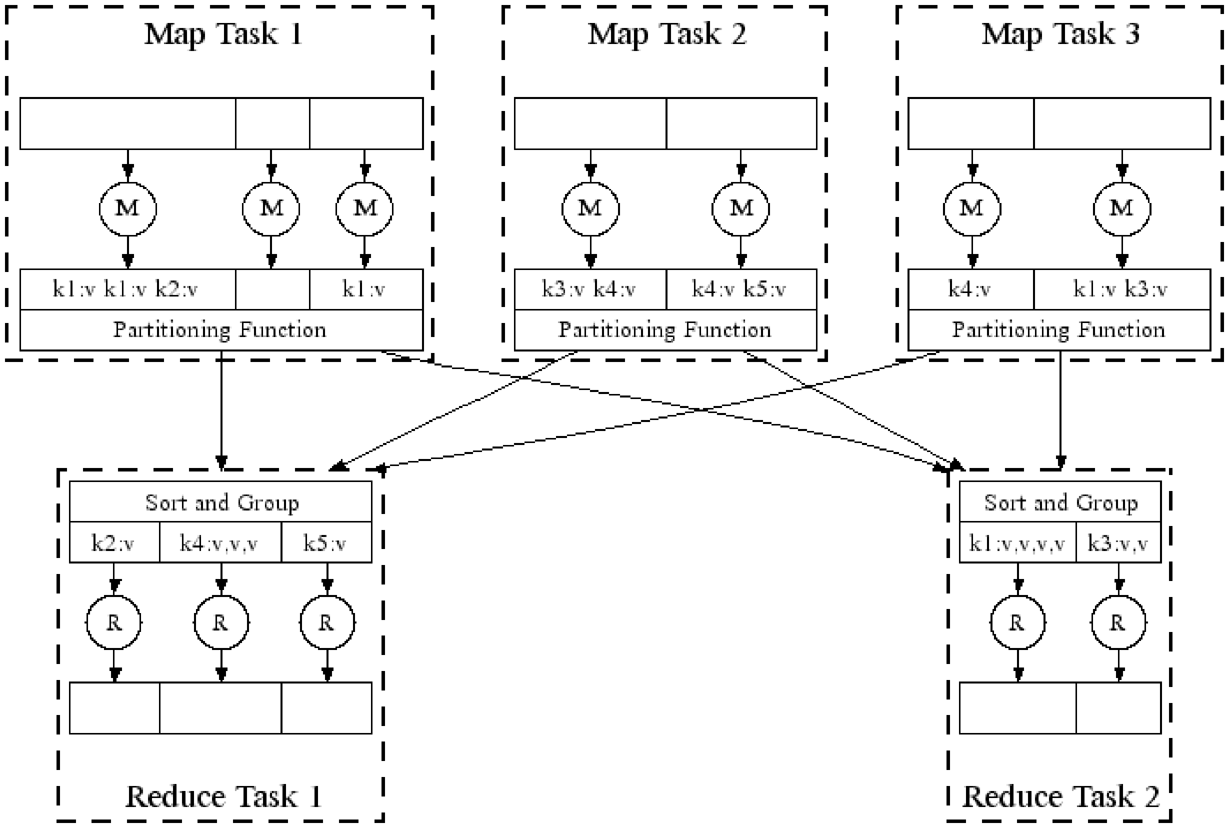
Execution Overview

Dean and Ghemawat, 2008



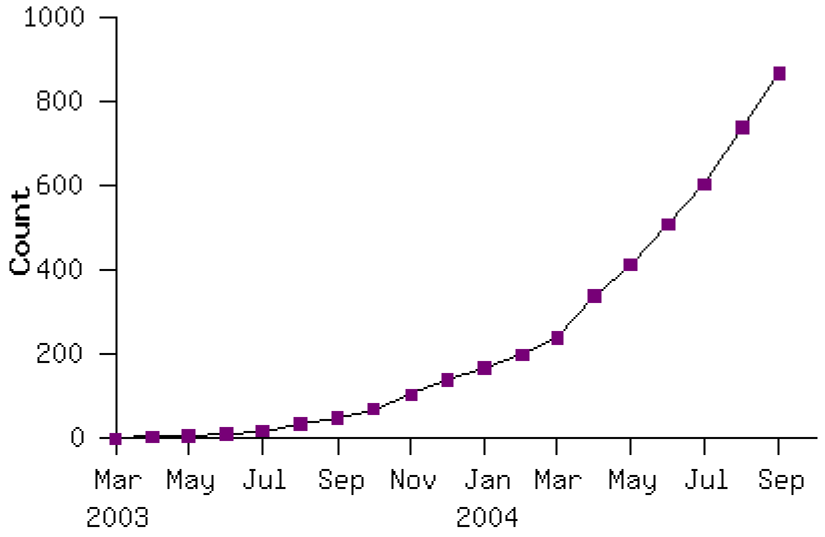
Parallel Execution

- 200,000 map/5000 reduce tasks w/ 2000 machines (Dean and Ghemawat, 2004)
- Over 1m/day at FB last year



Model has Broad Applicability

MapReduce Programs In Google Source Tree



Example uses:

distributed grep	distributed sort	web link-graph reversal
term-vector per host	web access log stats	inverted index construction
document clustering	machine learning	statistical machine translation

Usage at Google

Table I. MapReduce Statistics for Different Months.

	Aug. '04	Mar. '06	Sep. '07
Number of jobs (1000s)	29	171	2,217
Avg. completion time (secs)	634	874	395
Machine years used	217	2,002	11,081
map input data (TB)	3,288	52,254	403,152
map output data (TB)	758	6,743	34,774
reduce output data (TB)	193	2,970	14,018
Avg. machines per job	157	268	394
Unique implementations			
map	395	1958	4083
reduce	269	1208	2418

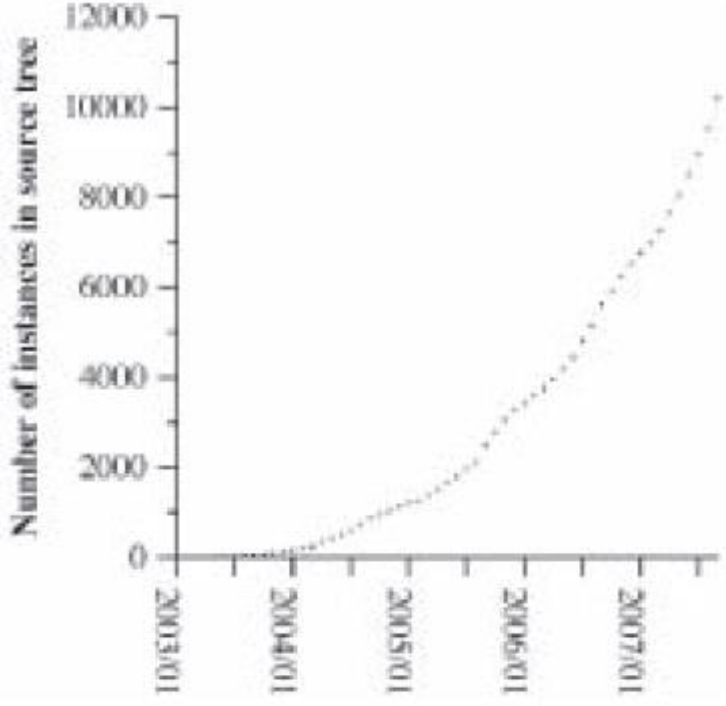
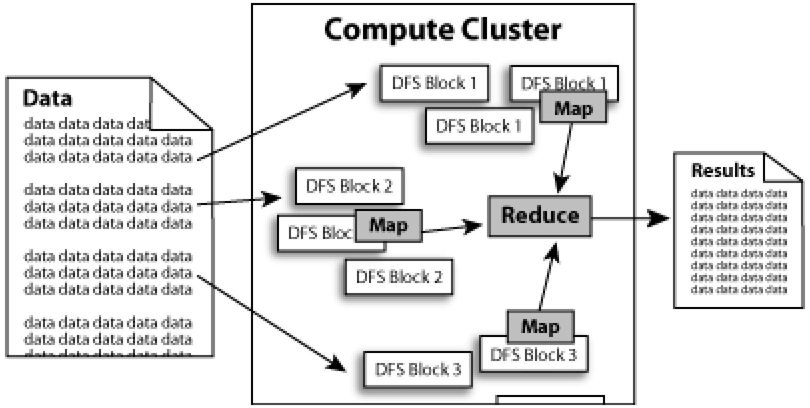


Fig. 4. MapReduce instances over time.

Hadoop

- Open Source Apache project
- Written in Java; runs on Linux, Windows, OS/X, Solaris
- Hadoop includes:
  - MapReduce: distributes applications
  - HDFS: distributes data



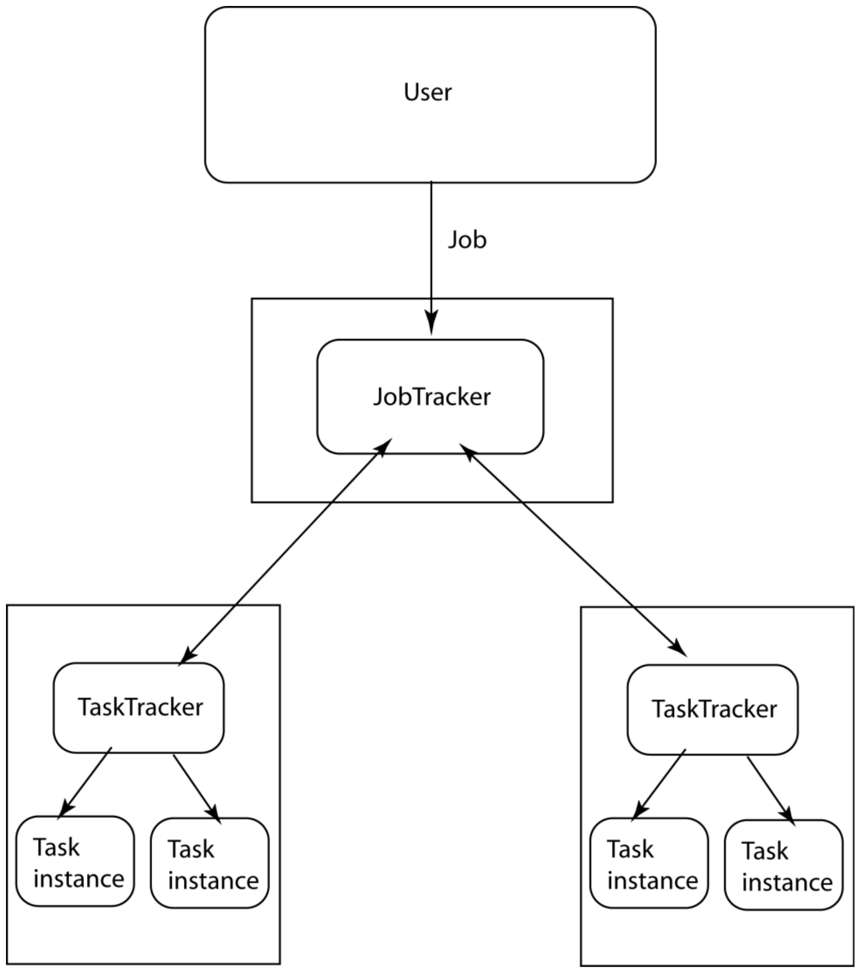
Hadoop Design Goals

- Storage of large data sets
- Running jobs in parallel
- Maximizing disk I/O
- Batch processing

Job Distribution

- Users submit mapreduce jobs to jobtracker
- Jobtracker puts jobs in queue, executes on first-come, first-served basis
- Jobtracker manages assignment of map and reduce tasks to tasktrackers
- Tasktrackers execute tasks upon instruction from jobtracker, and handle data transfer between map and reduce phases

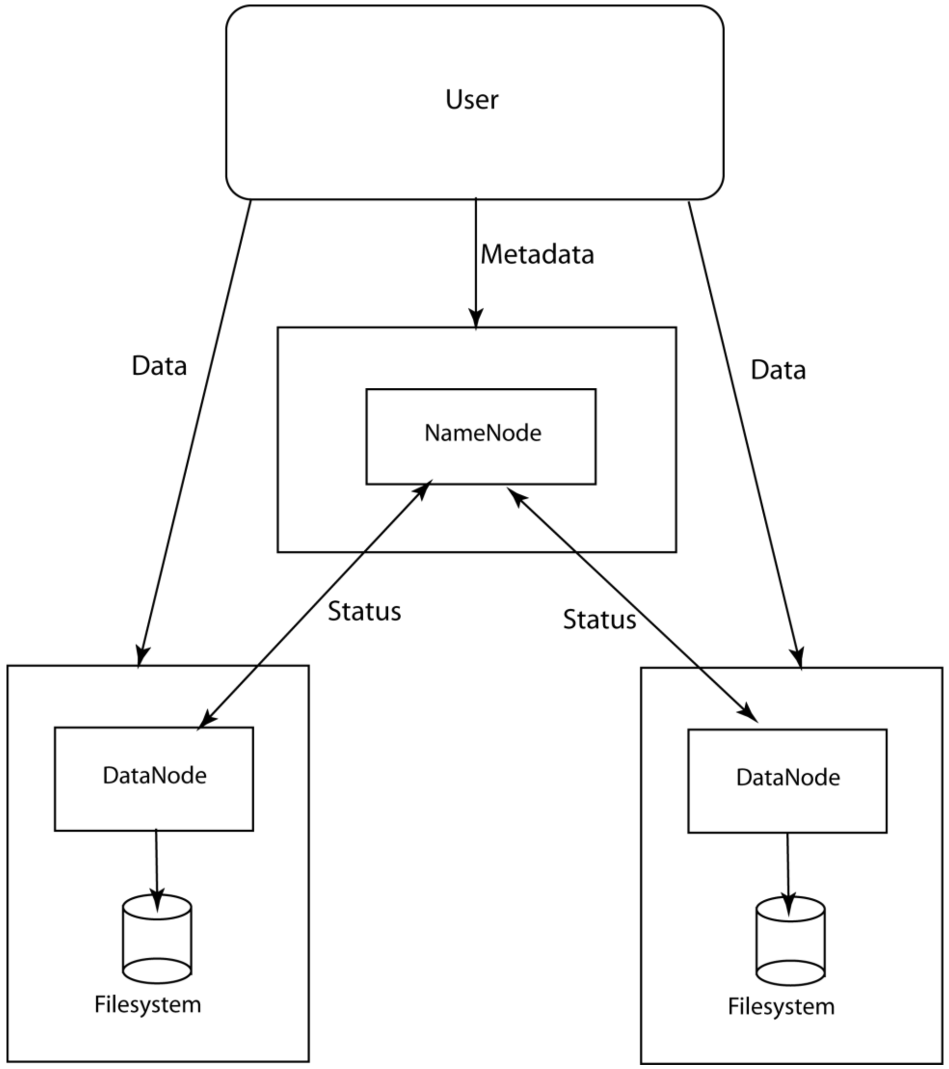
Hadoop MapReduce



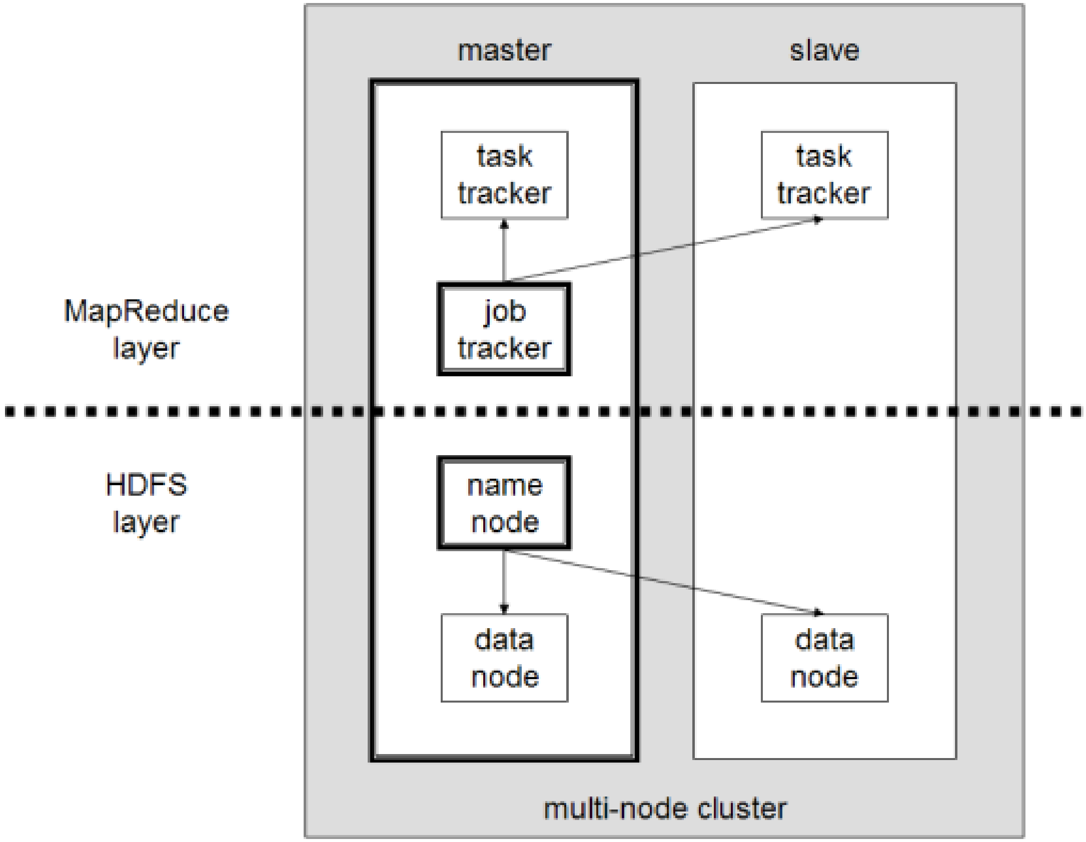
Data Distribution

- Data transfer handled implicitly by HDFS
- Move computation to where data is: data locality
- Map tasks are scheduled on same node that input data resides on
- If lots of data is on the same node, nearby nodes will map instead

Hadoop DFS (HDFS)



Map Reduce and HDFS



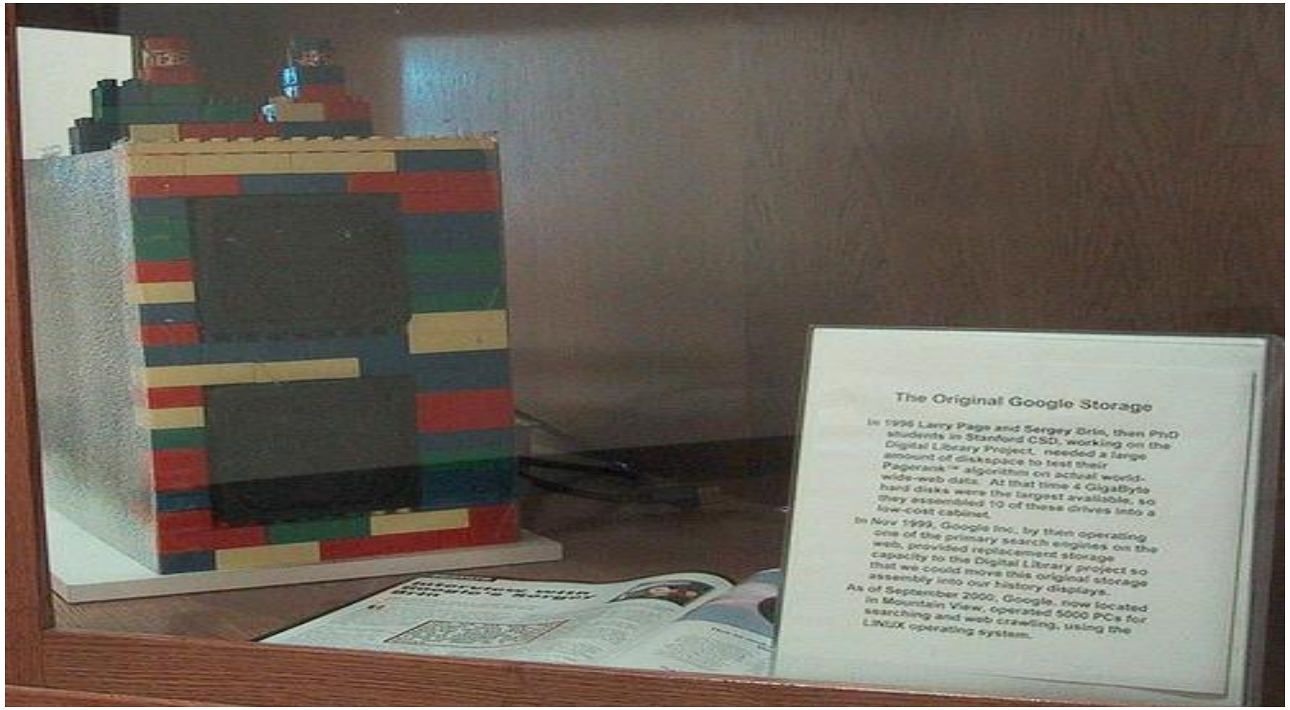
http://www.michael-noll.com/wiki/Running_Hadoop_On_Ubuntu_Linux_(Multi-Node_Cluster)

Data Access

- CPU and transfer speed, RAM and disk size double every 18-24 months
- Disk seek time is nearly constant (~5% per year)
- Time to read entire disk is growing

- Scalable computing should not be limited by disk seek time
- Throughput more important than latency

Original Google Storage



Source: Computer History Museum

HDFS

- Inspired by Google File System (GFS)
- Follows master/slave architecture
- HDFS installation has one Namenode and one or more Datanodes (one per node in cluster)
- Namenode: Manages filesystem namespace and regulates file access by clients. Makes filesystem namespace operations (open/close/rename of files and directories) available via RPC
- Datanode: Responsible for serving read/write requests from filesystem clients. Also perform block creation/deletion/replication (upon instruction from Namenode)

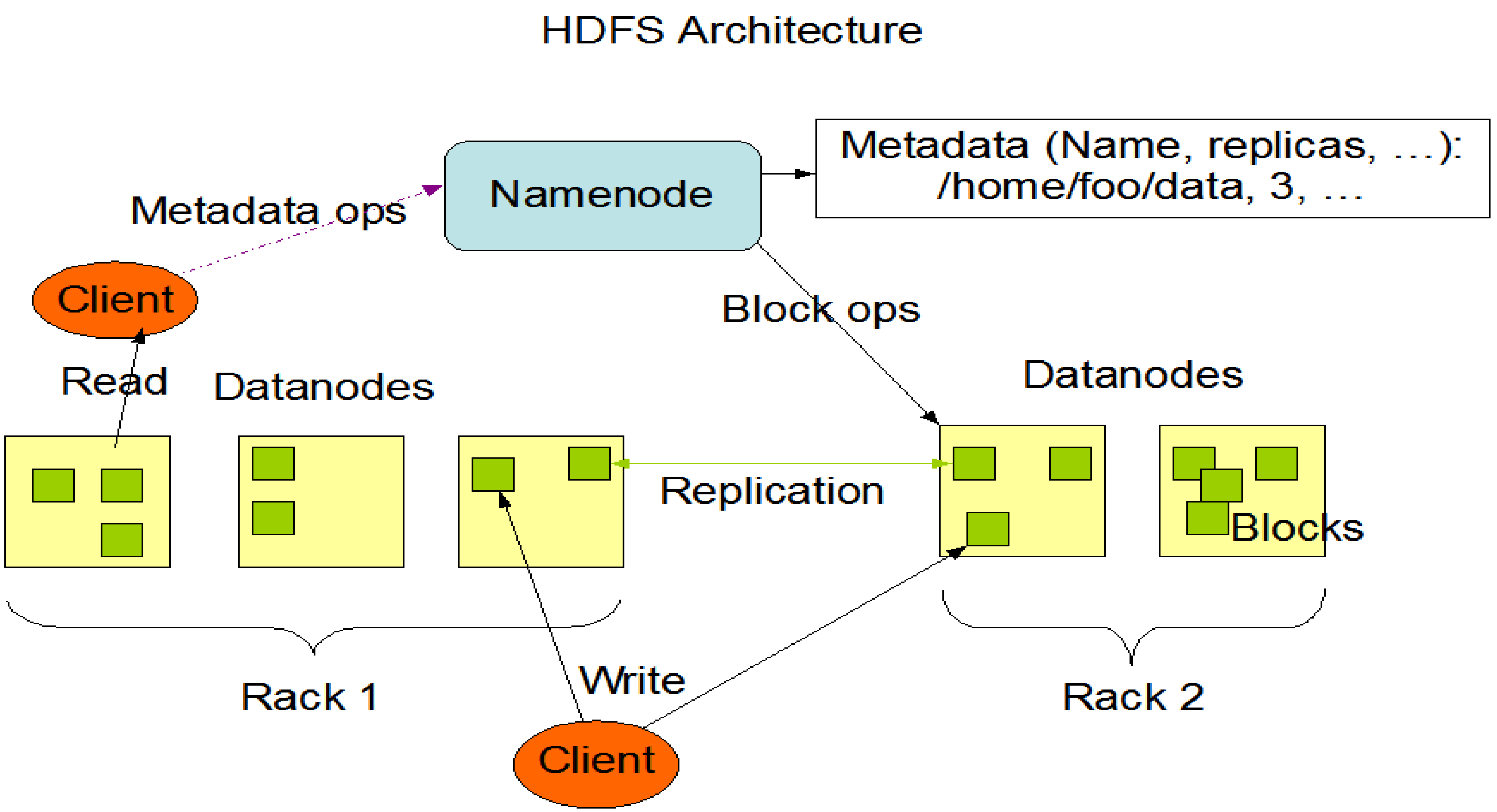
HDFS Design Goals

- Very large files:
  - Files may be GB or TB in size
- Streaming data access:
  - Write once, read many times
  - Throughput more important than latency
- Commodity hardware
  - Node failure may occur

HDFS

- Files are broken into blocks of 64MB (but can be user specified)
- Default replication factor is 3x
- Block placement algorithm is rack-aware
- Dynamic control of replication factor

HDFS



Source: http://lucene.apache.org/hadoop/hdfs_design.html

Example HDFS Installation

- Facebook, 2010 (Largest HDFS installation at the time)
- 2000 machines, 22,400 cores
- 24 TB / machine, (21 PB total)
- Writing 12TB/day
- Reading 800TB/day
- 25K MapReduce jobs/day
- 65 Million HDFS files
- 30K simultaneous clients.

2014: Facebook generates 4 new petabyes of data and runs 600,000 queries and 1 million map-reduce jobs per day.

Hive is Facebook's data warehouse, with 300 petabytes of data in 800,000 tables

More at: https://research.facebook.com/blog/1522692927972019/facebook-s-top-open-data-problems/

1B users per day
(http://fortune.com/2015/08/28/1-billion-facebook/)

Companies to first use MapReduce/Hadoop

- Google
- Yahoo
- Microsoft
- Amazon
- Facebook
- Twitter
- Fox interactive media
- AOL
- Hulu

LinkedIn
Disney
NY Times
Ebay
Quantcast
Veoh
Joost
Last.fm

Any company with enough data

Hadoop Vendors

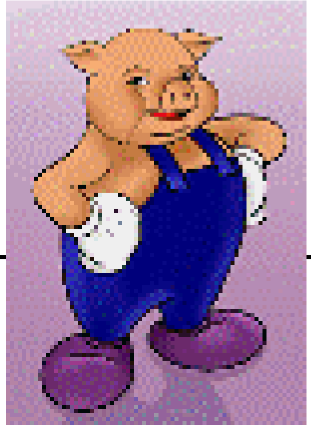
- Cloudera
- Hortonworks
- MapR
- IBM BigInsights

Hive

- **Apache Hive** is a data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis.
- Developed at Facebook to enable analysts to query Hadoop data
- MapReduce for computation, HDFS for storage, RDBMS for metadata

- **Can use Hive to perform SQL style queries on Hadoop data**
- Most Hive queries generate MapReduce jobs
- Can perform parallel queries over massive data sets

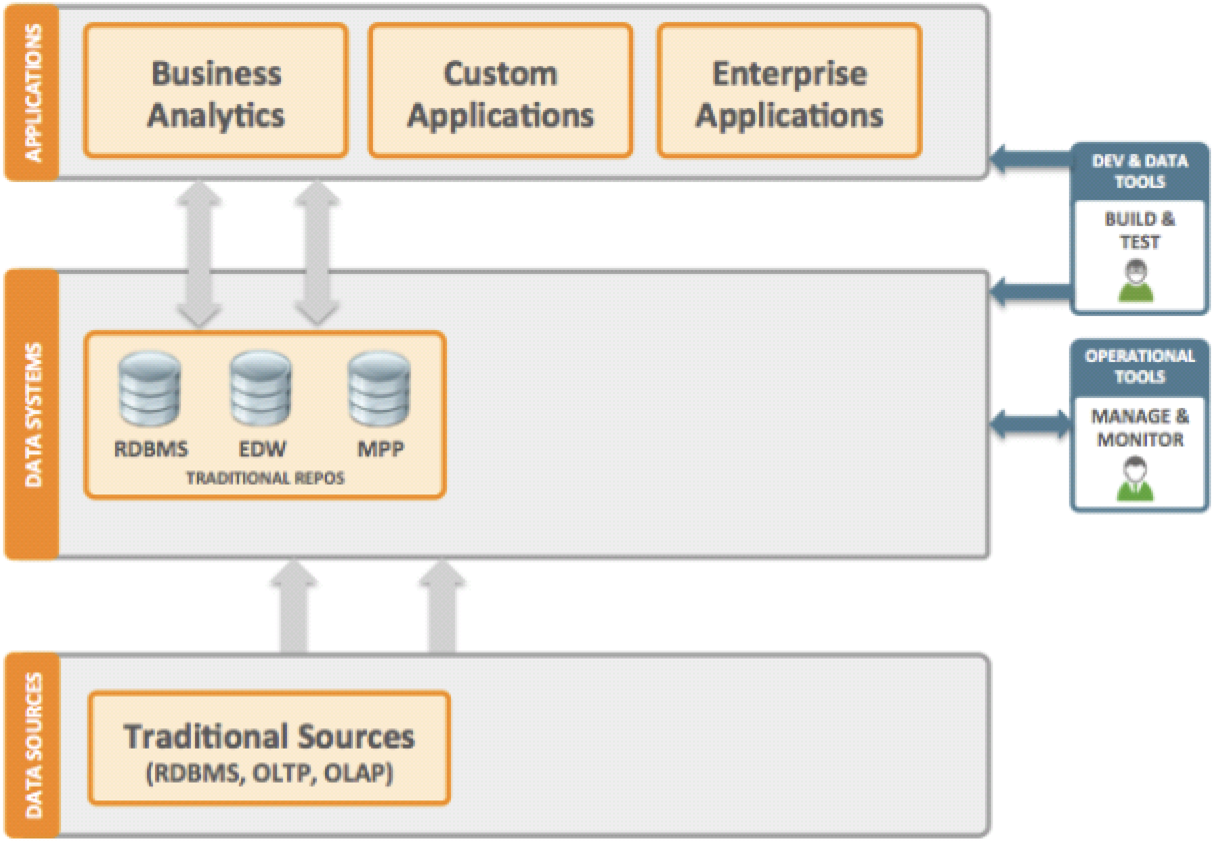
Apache Pig



- Pig: High-level platform for creating MapReduce programs
- Developed at Yahoo
- Pig Latin: Procedural language for Pig

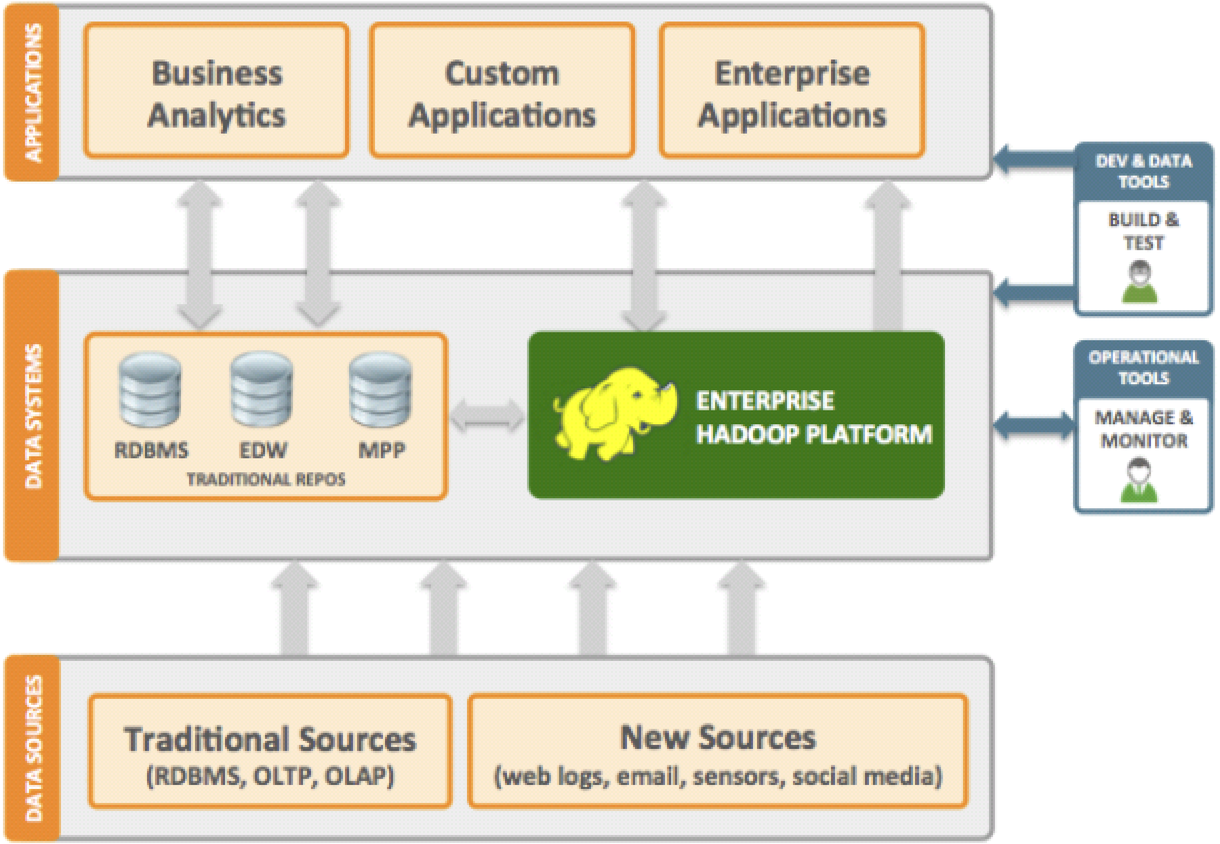
- Ability to use user code at any point in pipeline makes it good for pipeline development

Traditional Data Enterprise Architecture



Hortonworks, 2013

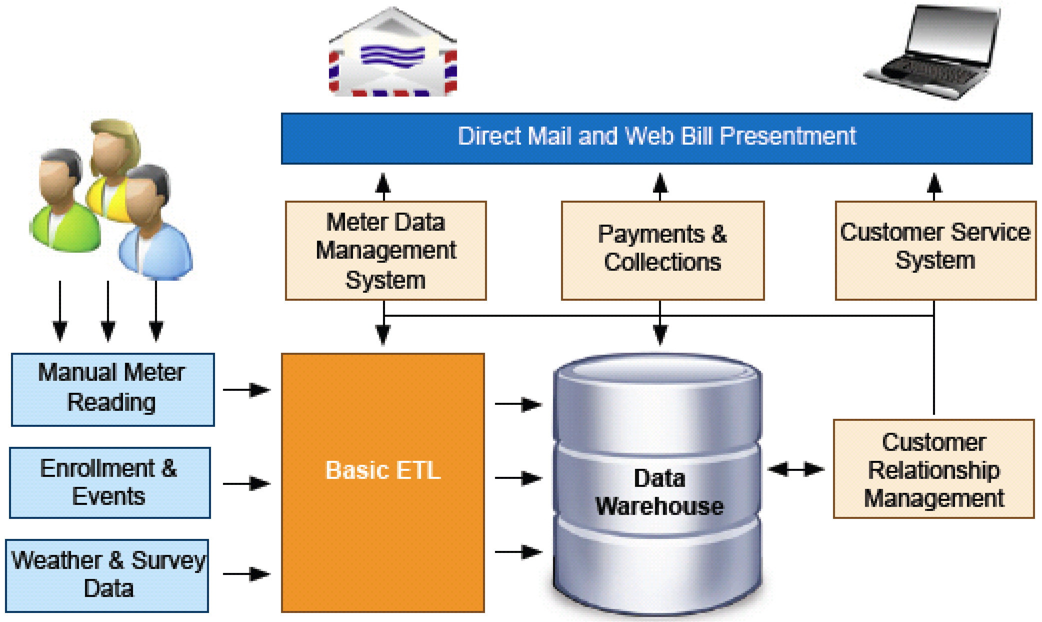
Emerging Big Data Architecture



1. Collect data
2. Clean and process using Hadoop
3. Push data to Data Warehouse, or use directly in Enterprise applications

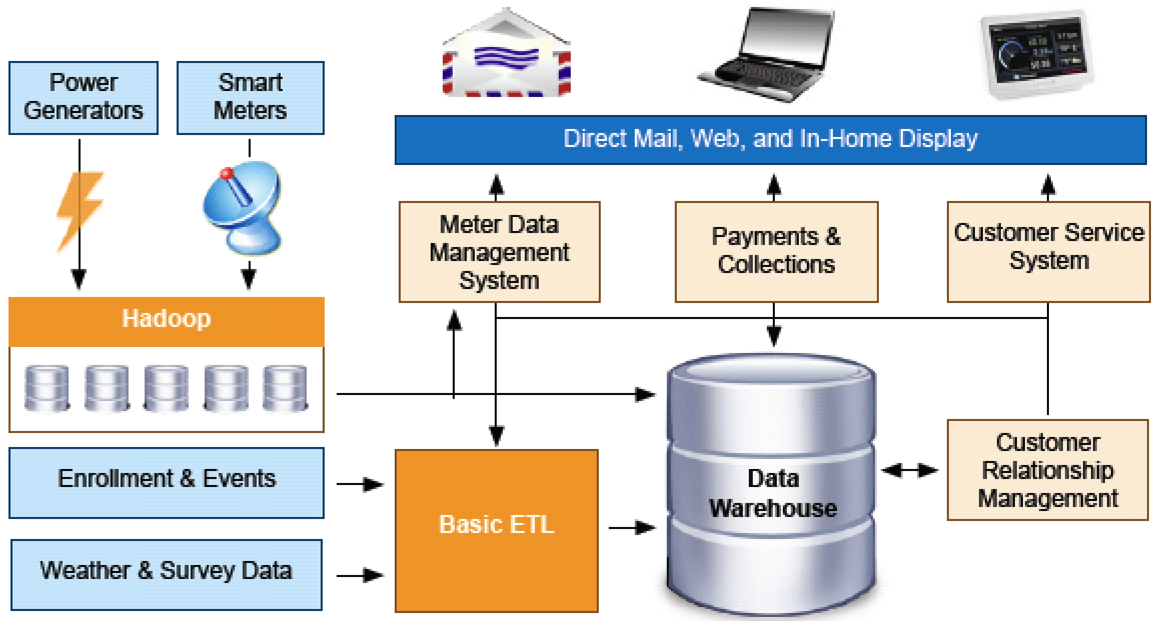
Hortonworks, 2013

- Data flow of meter done manually



Awadallah and Graham

- Automatic update of meter readings every 15 mins



Awadallah and Graham

DW and Hadoop Use Cases

Requirement	Data Warehouse	Hadoop
Low latency, interactive reports, and OLAP	●	
ANSI 2003 SQL compliance is required	●	
Preprocessing or exploration of raw unstructured data		●
Online archives alternative to tape		●
High-quality cleansed and consistent data	●	
100s to 1000s of concurrent users	●	●*
Discover unknown relationships in the data	●	●
Parallel complex process logic		●
CPU intense analysis	●	●
System, users, and data governance	●	
Many flexible programming languages running in parallel		●
Unrestricted, ungoverned sand box explorations		●
Analysis of provisional data		●
Extensive security and regulatory compliance	●	
Real time data loading and 1 second tactical queries	●	●*

Figure 4. Requirements match to platforms

*HBase

Awadallah and Graham