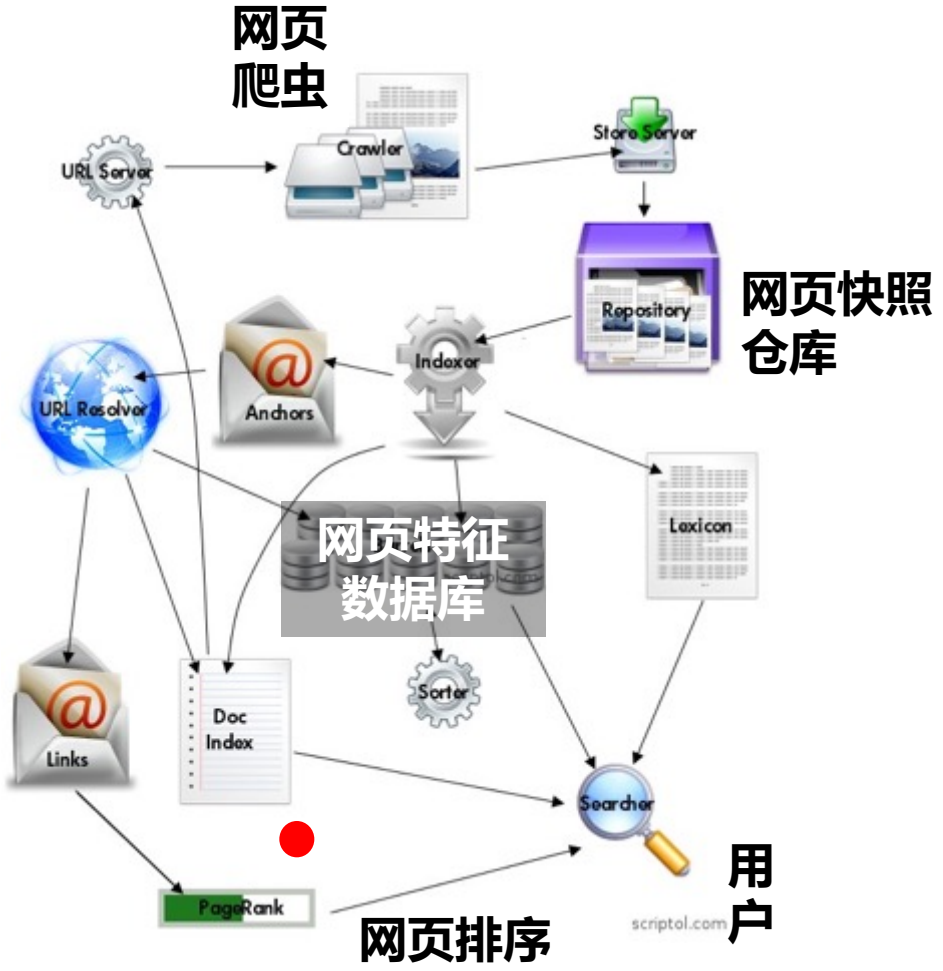
大数据软件栈

Stack

大数据生态系统

Google搜索的挑战

- 一次写、多次读
- 数据规模巨大，
- 价值密度很低，
- 成本十分敏感



Google硬件选型的理由

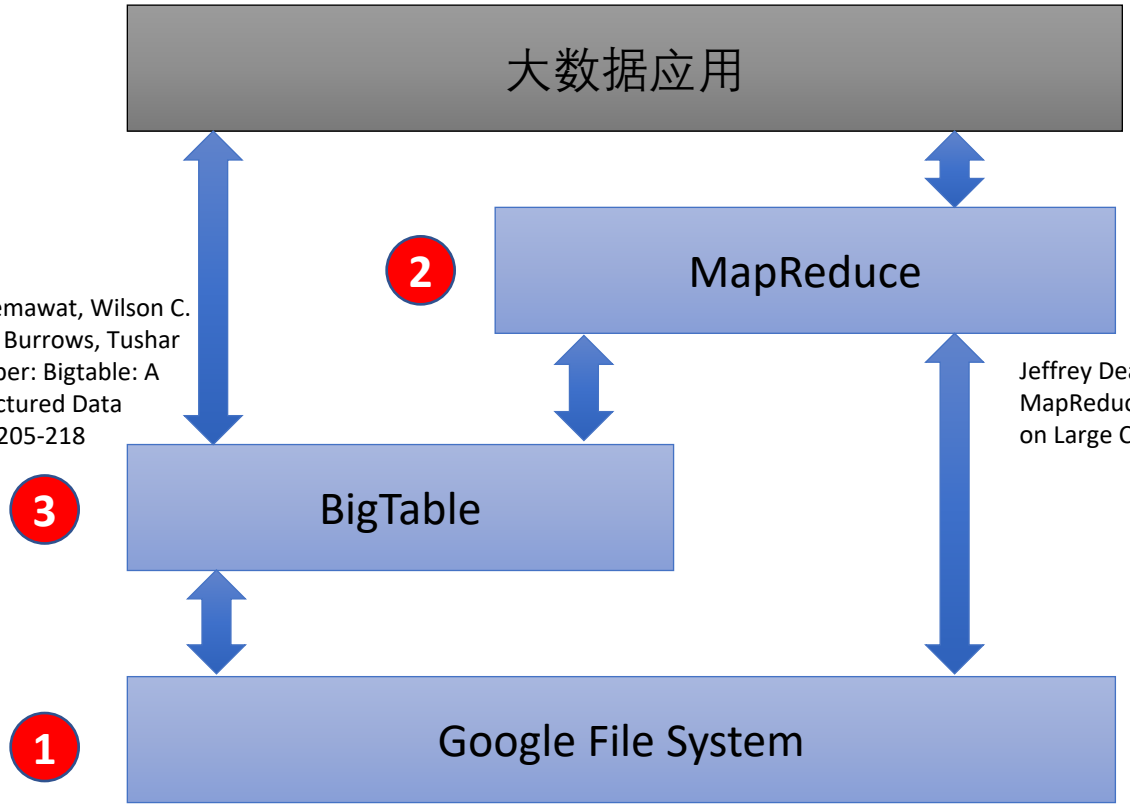
- 服务器 or PC机?

	服务器	PC
计算性能	高	低
数据存储	大	小
故障率	小	高
成本	昂贵	便宜

- 网页数据量虽大，但价值密度很低
  - 每个查询的成本 < 5美分

Google大数据软件

- 弥补硬件缺陷的“量身定制”的软件栈



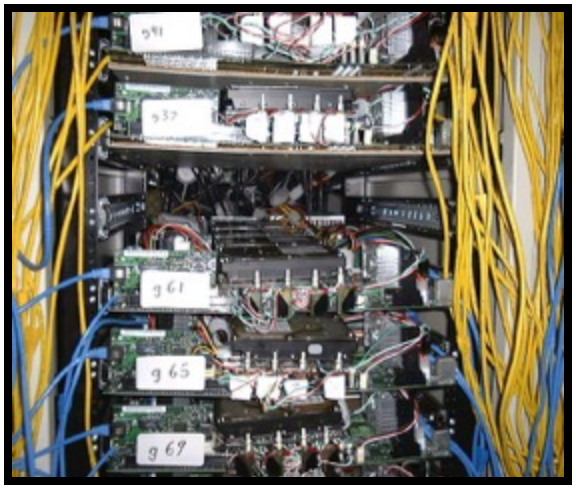
Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Michael Burrows, Tushar Chandra, Andrew Fikes, Robert Gruber: Bigtable: A Distributed Storage System for Structured Data (Awarded Best Paper!). OSDI 2006: 205-218

Jeffrey Dean, Sanjay Ghemawat: MapReduce: Simplified Data Processing on Large Clusters. OSDI2004: 137-150

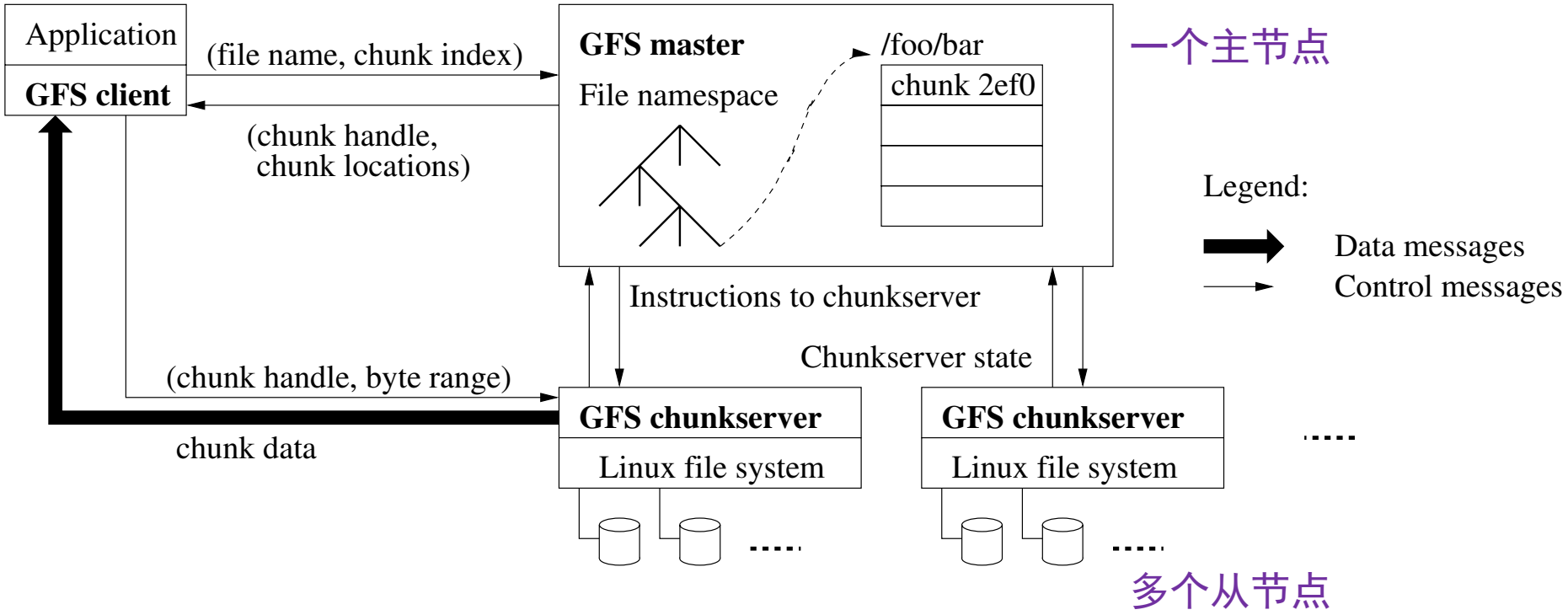
Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung: The Google file system. SOSP 2003: 29-43

1：GFS - Google文件系统

- “小机器”拼成“大系统”
- “差机器”变成“好系统”
- 为“并行计算”准备“分布数据”

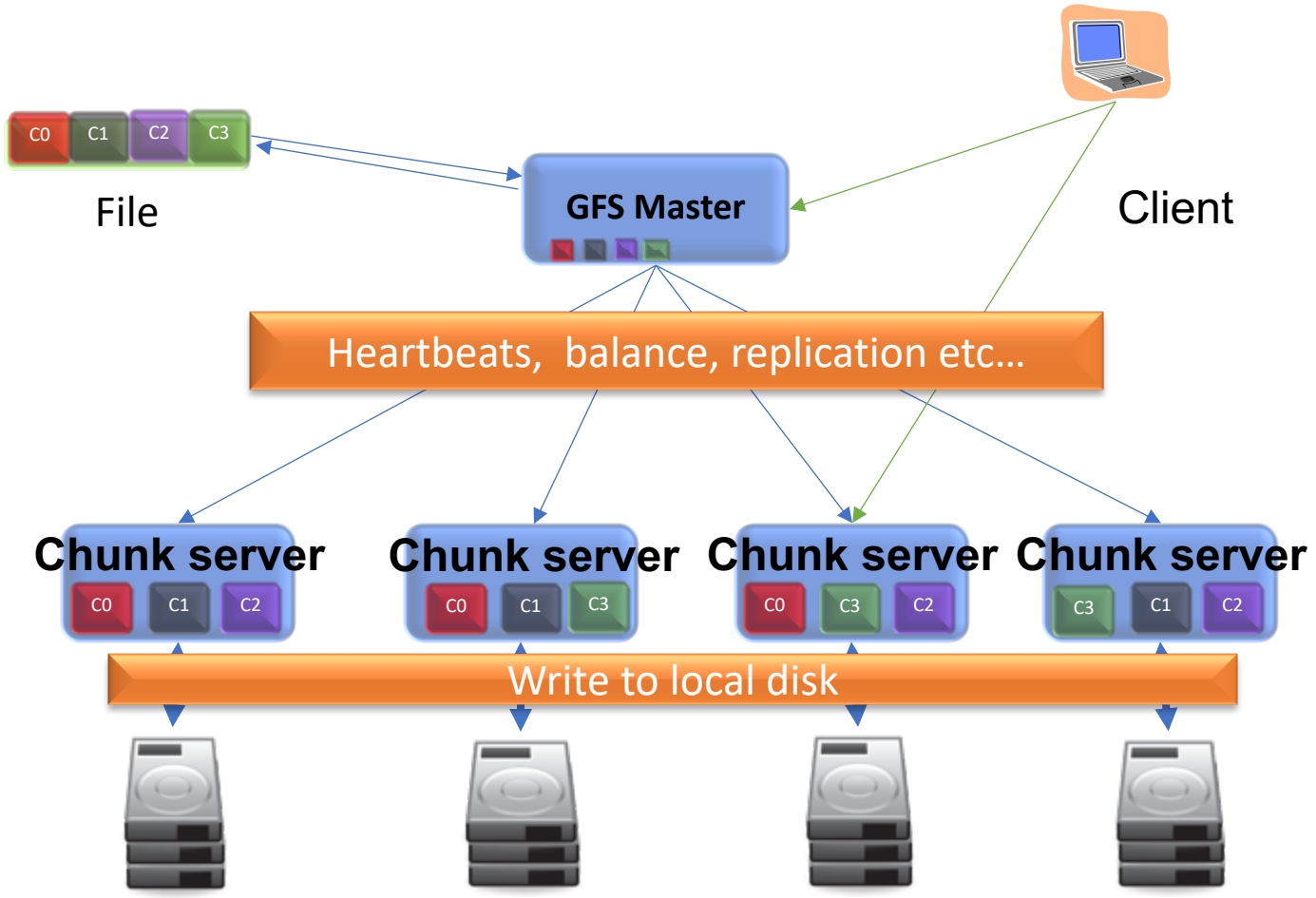


GFS采用主从架构

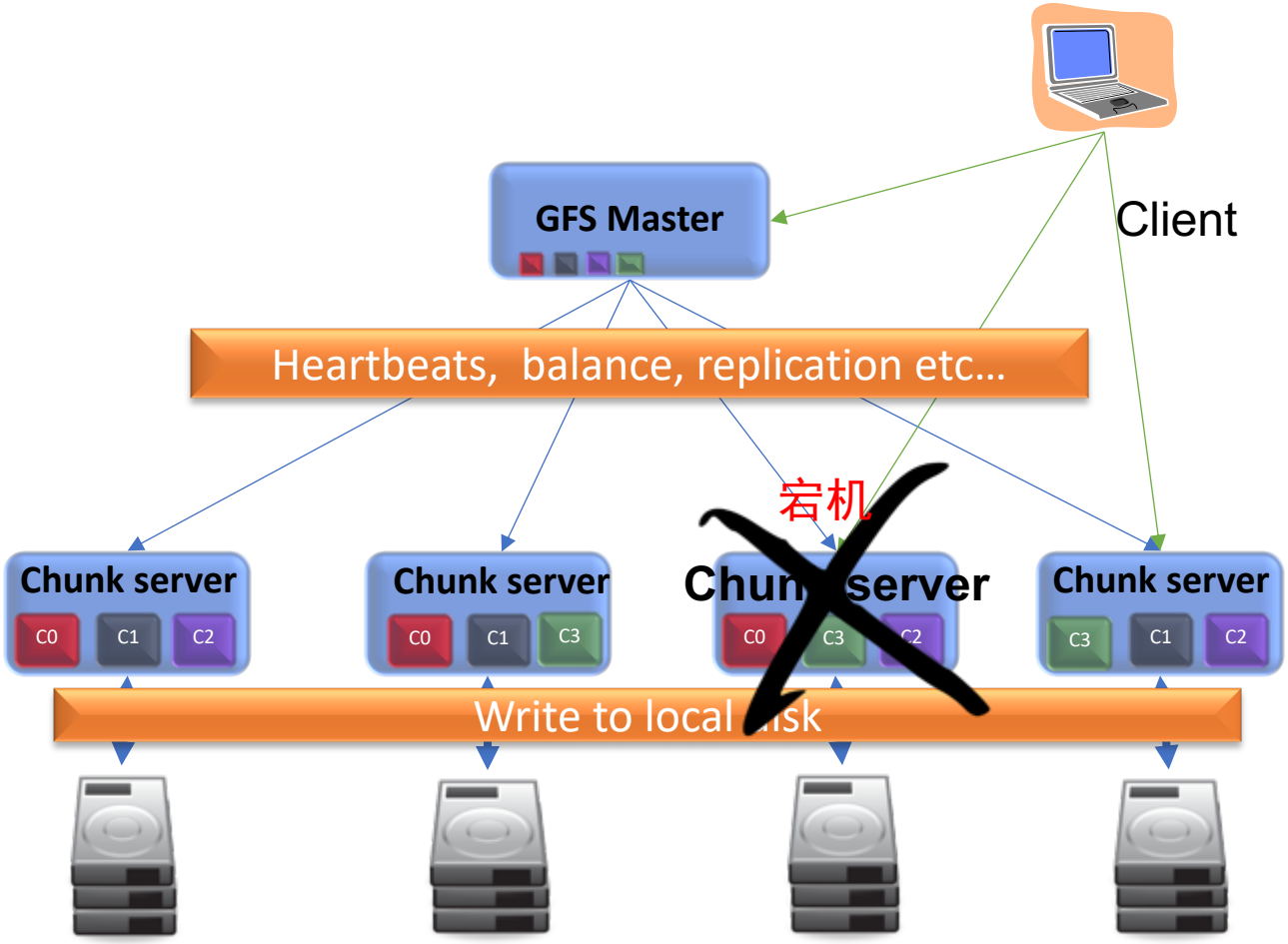


Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung: The Google file system. SOSP 2003: 29-43

GFS文件的分片与副本

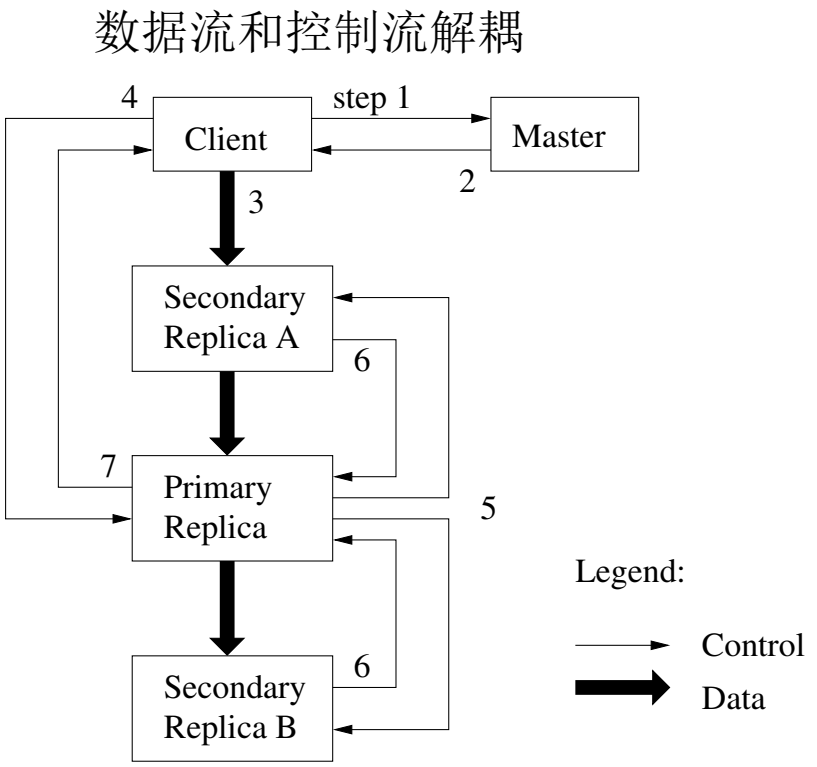


GFS副本：“读”的收益


GFS Master
Client
Heartbeats, balance, replication etc…
宕机
Chunk server
C0
C1
C2
Chunk server
C0
C1
C3
Chunk server
C0
C3
C2
Chunk server
C3
C1
C2
Write to local disk

GFS副本：“写”的代价

- 原则:
  - “写”必须覆盖所有副本
  - 主(Master)节点负载最小化
- 机制:
  - 主节点指定主副本
  - 主副本获得“写”锁
  - 主副本确定“传播链”
  - 所有副本遵循同一顺序



2：Google MapReduce

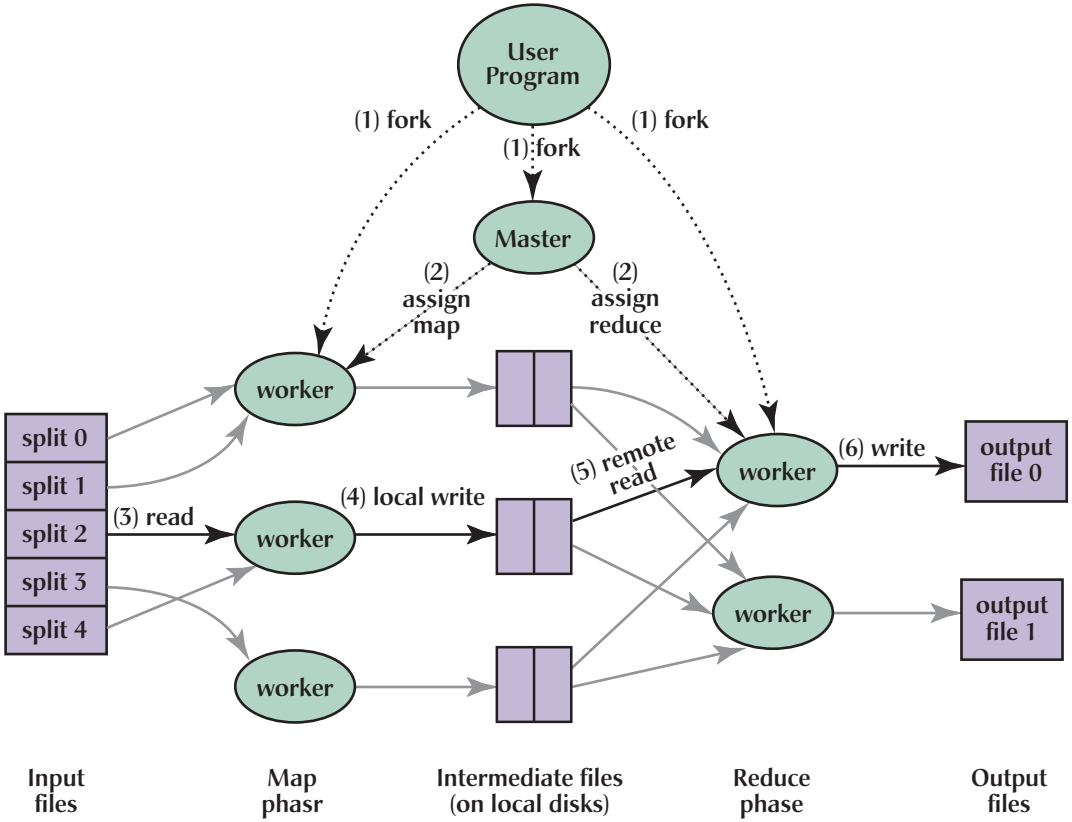
- Google提出的一个用于大规模数据集的、适于低成本硬件机群的、可靠容错的、并行运算软件框架

- 处理大规模数据（大于1PB）
- 大规模并行处理（一千个节点以上）
- 使并行编程实现更加容易

Jeffrey Dean, Sanjay Ghemawat: MapReduce: Simplified Data Processing on Large Clusters. OSDI2004: 137-150

MR让程序找数据

- GFS分片机制为 “MR程序” 准备好 “分布的数据”
  - 输入和输出数据都放在GFS中

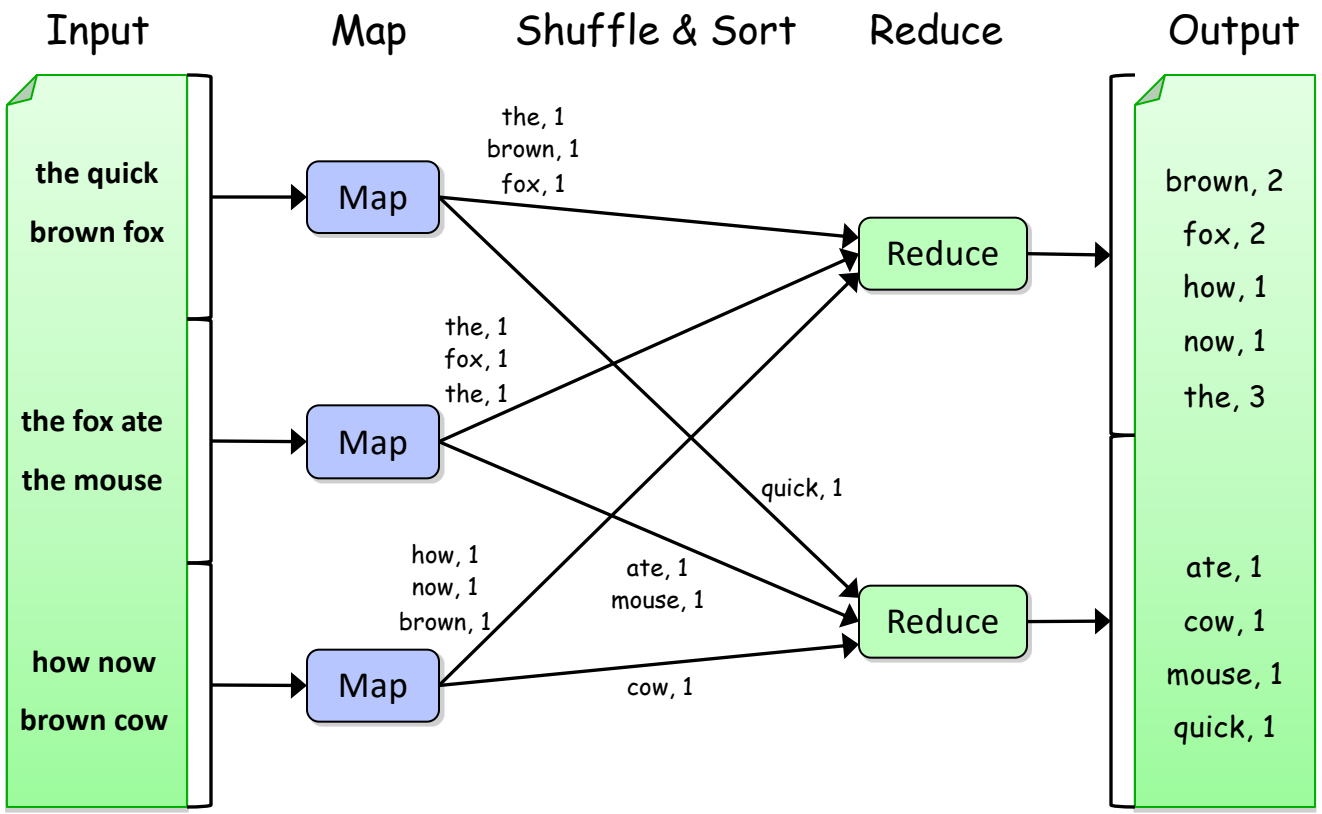


MR实例: Word Count

- MR框架，负责变“串行”为“并行”

```
def mapper(line):
    foreach word in line.split():
        output(word, 1)

def reducer(key, values):
    output(key, sum(values))
```

Word Count 执行过程



MR的三个阶段

- Map – 在DataNode节点上执行Map()函数
- Shuffle – 根据Key将M输出的KV传到R节点上
- Reduce – 对不同的KV组执行Reduce()函数

- 2008年1月，Google MR每天数据处理量是20PB，
  - 相当于美国国会图书馆当年5月份存档数据的240倍

3：BigTable – Google大表

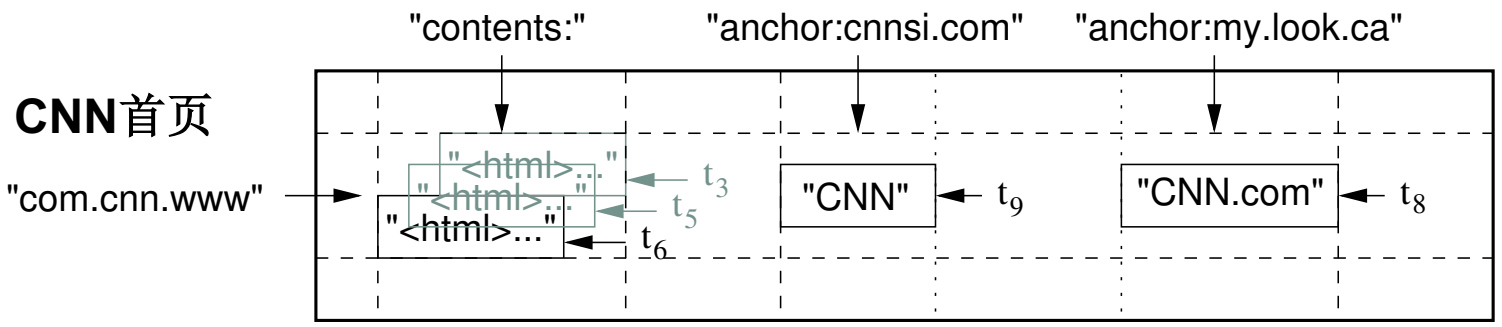
- Google具有的海量的（半）结构化数据
- 超过同时代商用数据库处理能力

- 网页数据（数十亿网页，多个版本，约20K/版本）
  - URL，内容，元数据，链接，锚点 …
- 用户数据（数亿计用户，数千次查/秒）
  - 用户画像，访问记录 …
- 地理数据（数百TB）
  - 实体（商店,餐馆等），街道，卫星图像 …

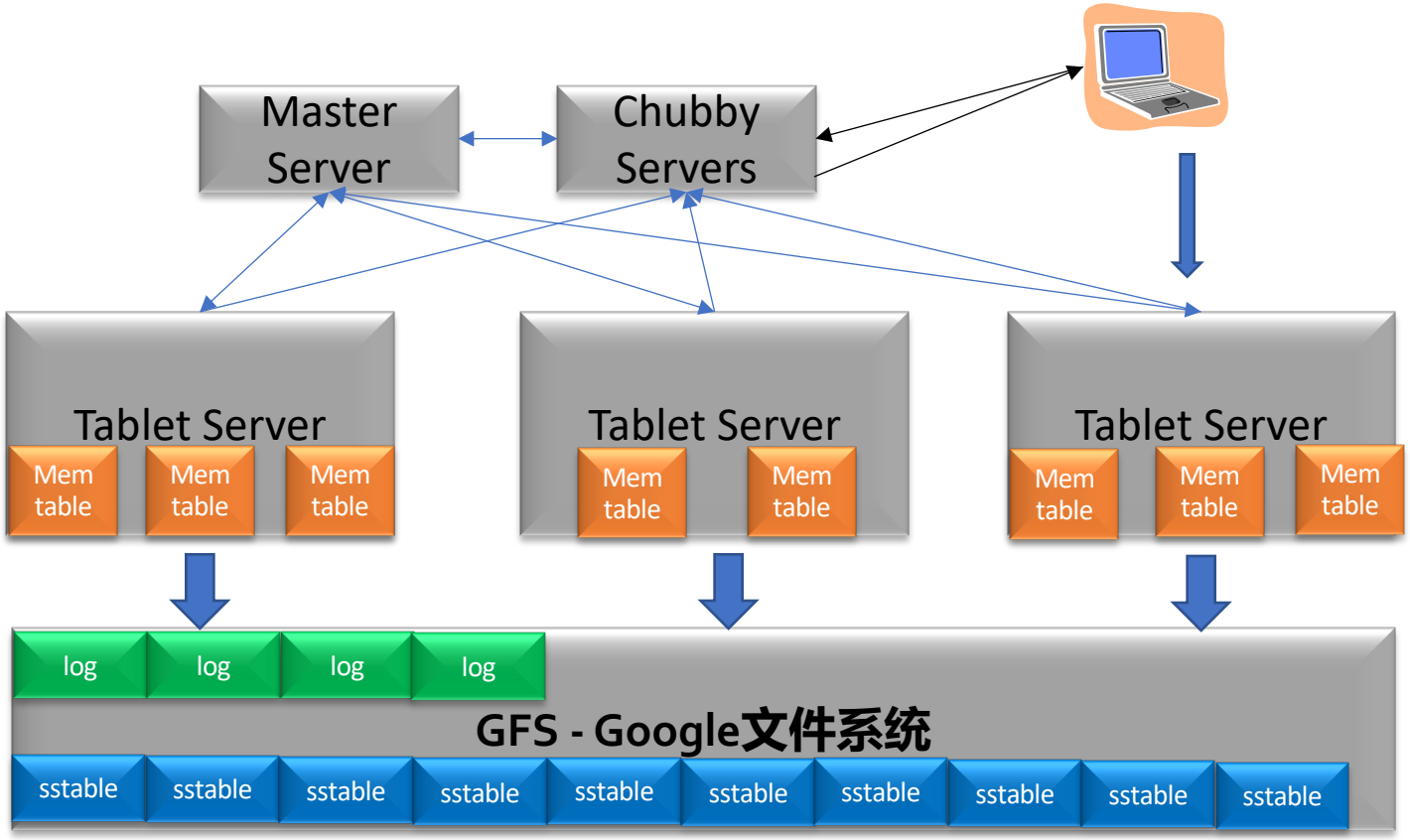
Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Michael Burrows, Tushar Chandra, Andrew Fikes, Robert Gruber: Bigtable: A Distributed Storage System for Structured Data (Awarded Best Paper!). OSDI 2006: 205-218

BT例子：WebTable

- WebTable - 保存Google索引的所有网页及相关信息
  - 网页URL作为行键（Row Key）
  - 网页的属性（如Content）作为列名（Column Name)
  - 爬取时间戳（Timestamp）作为网页版本号
- BT特点：稀疏、多维键（行/列/时刻）、有序



基于GFS实现BigTable



BigTable – “酸” - “碱” 平衡

- BigTable保证单行操作的一致性，不支持跨行事务

Atomicity 原子性
Consistency 一致性
Isolation 隔离性
Durability 持久性

Basically Available **基本可用**
Soft-state **柔性事务**
Eventual consistency **最终一致性**

关系数据库 **大数据系统**

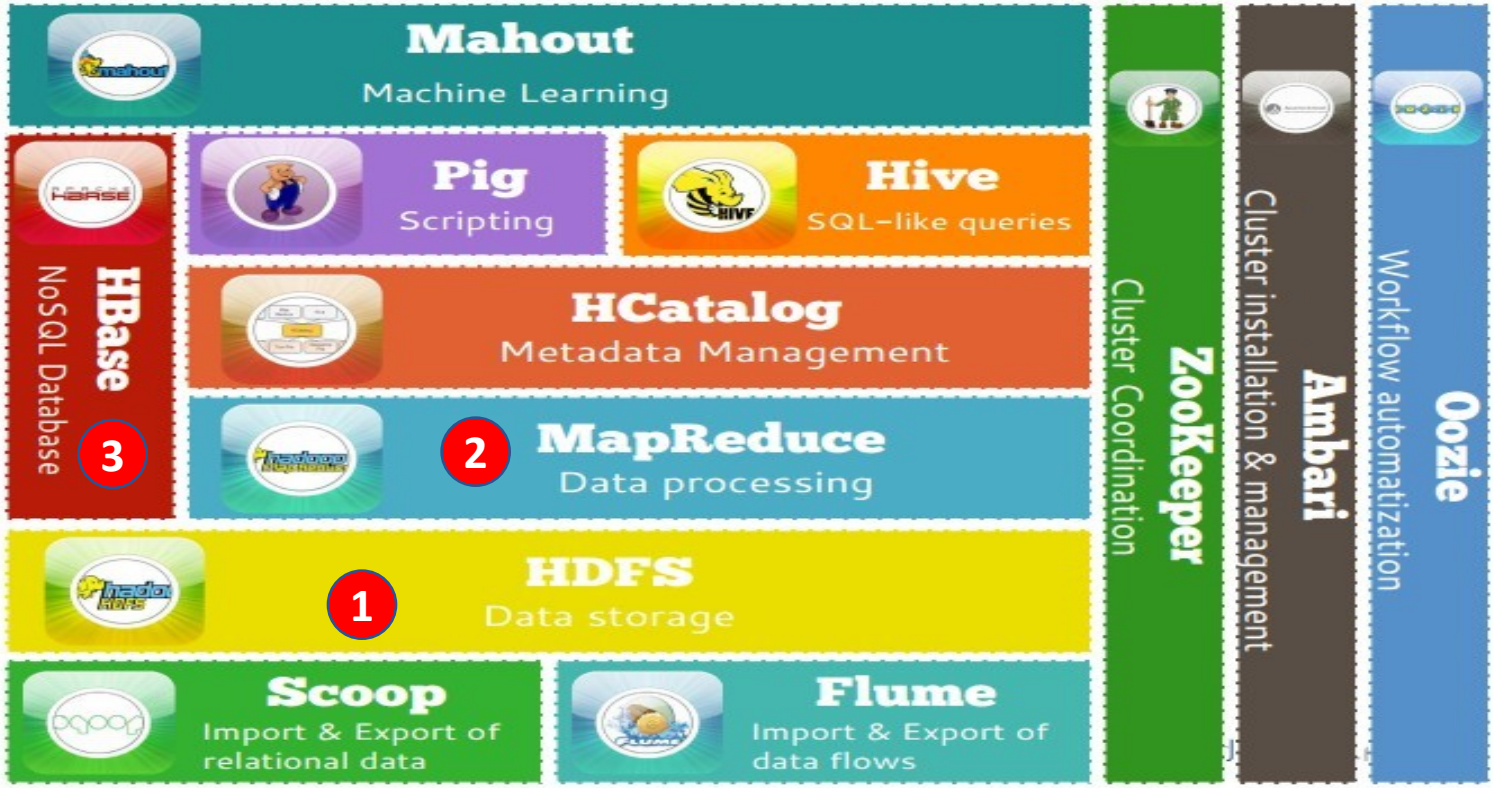
Google的贡献

- 颠覆
  - One size fits all 的惯性思维
  - IBM、Oracle、Microsoft产业巨头的垄断
- 贡献
  - 引爆了开源大数据系统的野蛮生长

Google and Big Data. History.

- 2003: Google File System. Scalable distributed file system for data intensive applications built over cheap commodity hardware.
- 2004: Map-Reduce. Programming model for large data sets. Automatically parallelizes jobs in large cluster of commodity machines. Inspiration for Hadoop.
- 2006: Bigtable. Distributed storage system for structured data. Inspiration for NoSQL databases
- 2010: Dremel. Ner real time solution with SQL-like language. Model for Google BigQuery.
- 2010: Pregel. Scalable Graph Computing. Mining data from social graphs.

Hadoop的生态圈



可视化

统计分析

分布式计算

数据存储

ETL

You Say, “tomato…”

Google calls it:	**Hadoop** equivalent:
MapReduce	Hadoop
GFS	HDFS
Bigtable	HBase

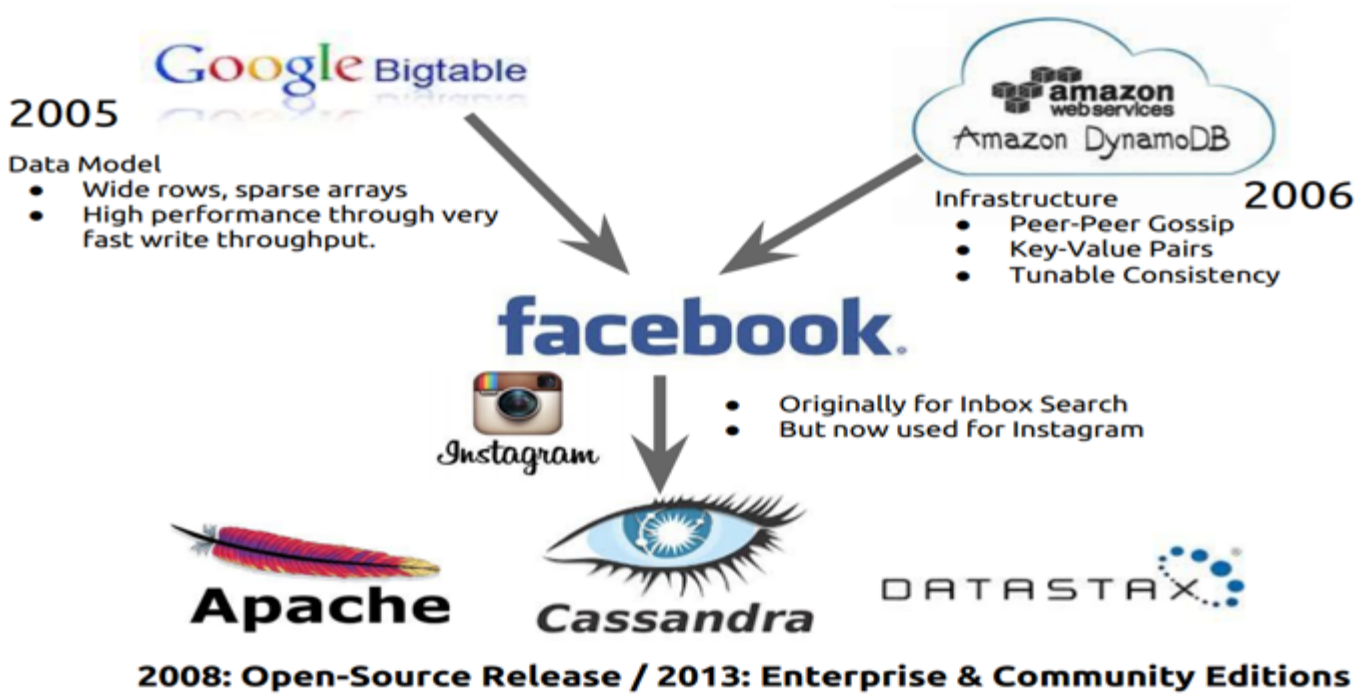
Google/Hadoop的技术弱点

- 主从架构
- 单点故障
- ……

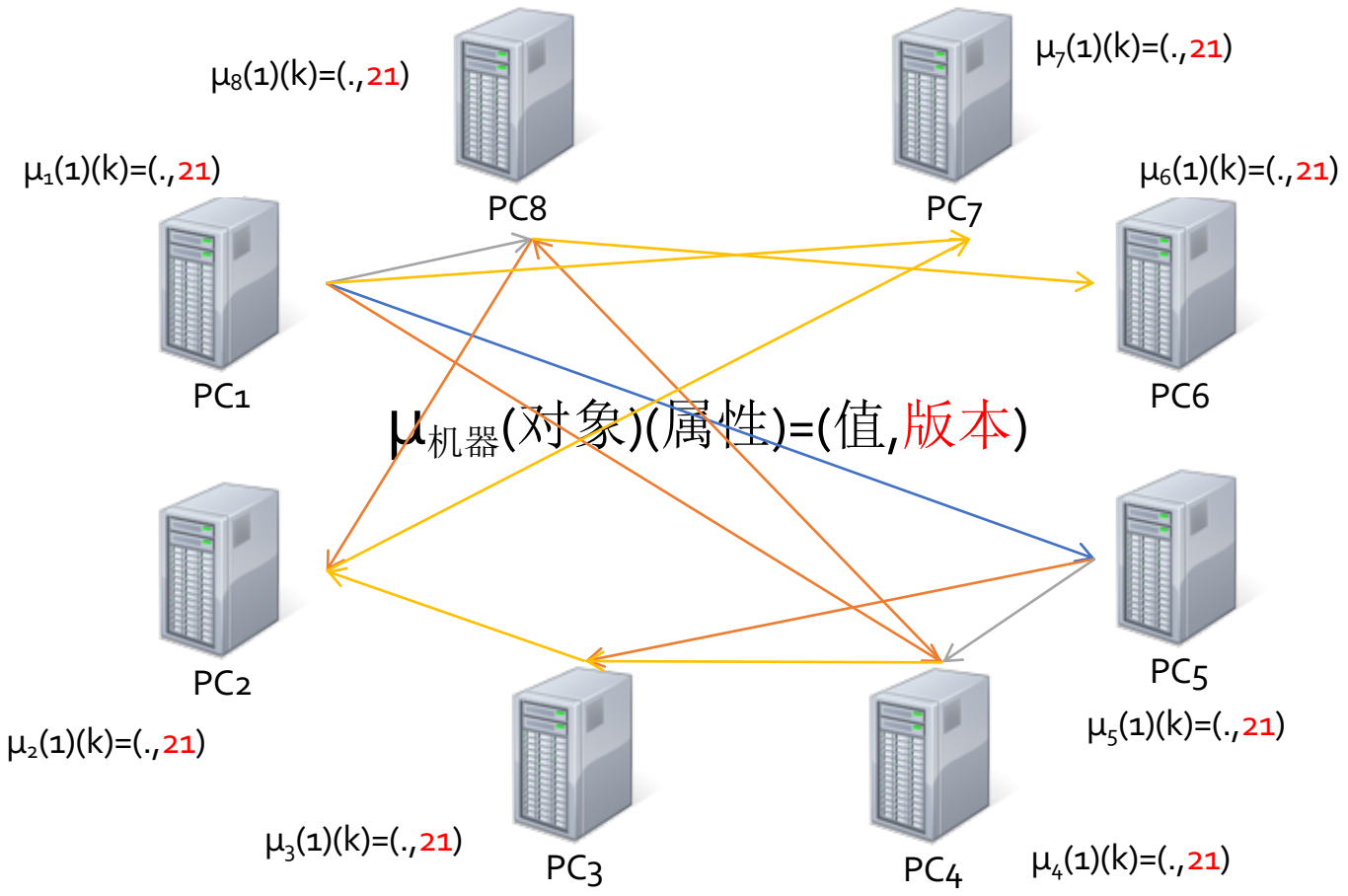
Cassandra

2008年开源，社交网、物联网领域的典型大数据系统

2015年，苹果公司用Cassandra管理了七万五千个节点



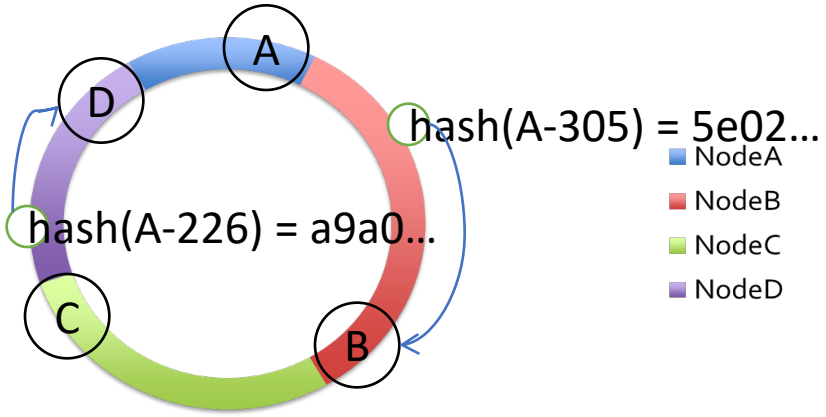
P2P Gossip 协议 – 流言传播



一致性哈希 - 改善系统弹性

- 原理 –把数据对象的Key映射到一个足够大的环上
  - 同时把每个机器也映射到环上的一个点
  - 每个机器负责环上的一段数据
- 优点 – 系统弹性好
  - 机器进入或离开时只影响有限个数据段

机器	开始	结束
A	e590…c	12a3…e
B	12a3…f	7310…3
C	7310…4	a331…1
D	a331…2	e590…b



Cassandra推动了NoSQL运动

NOSQL meetup
Last.fm
June 11, 2009

Jon Oskarsson,
Last.fm

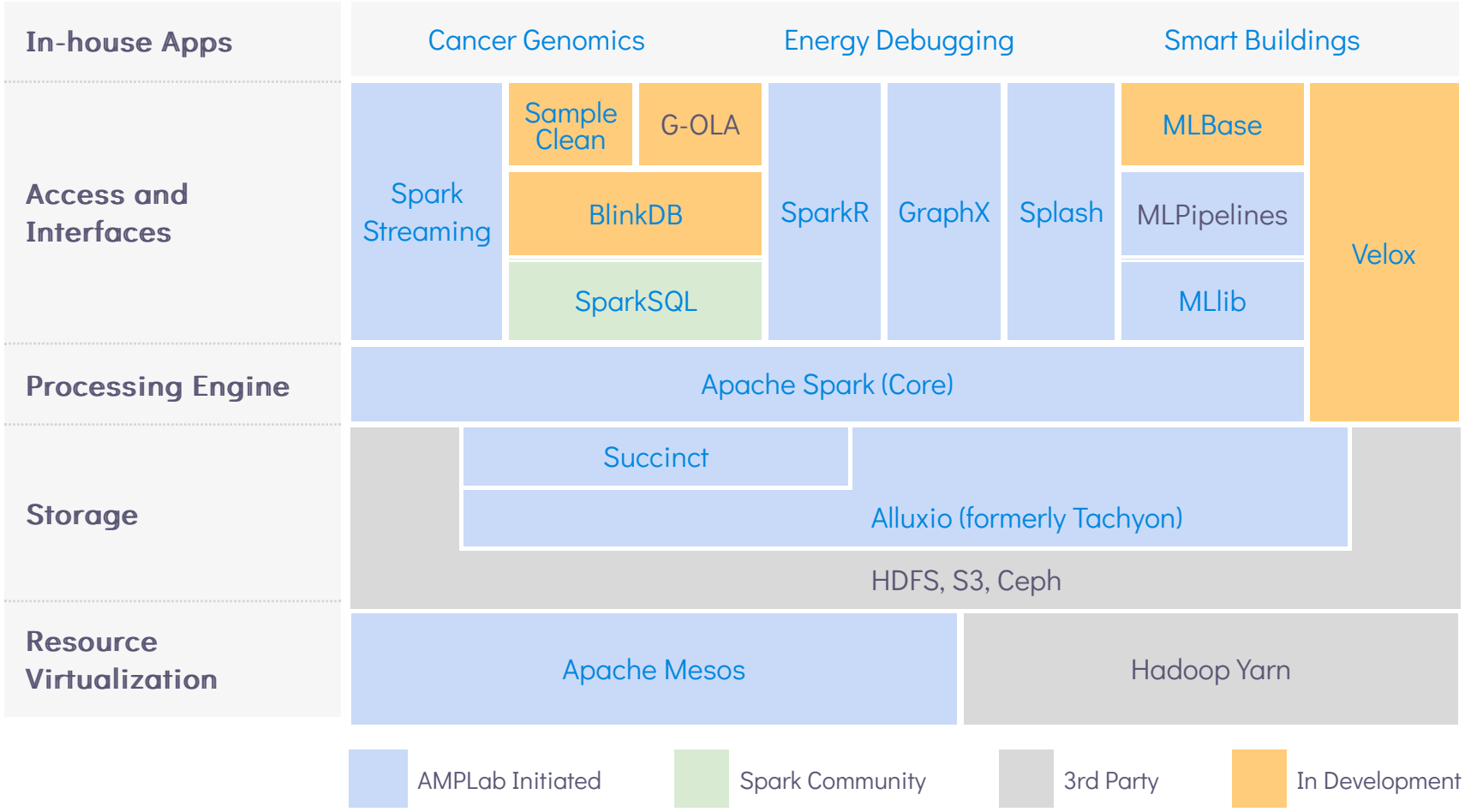
CBS Interactive是哥伦比亚广播集团（**CBS Corporation**）旗下的互动媒体公司。

10.00: **Intro session** (**Todd Lipcon**, Cloudera)
10.40: Voldemort (Jay Kreps, Linkedin)
11.20: Short break
11.30: Cassandra (Avinash Lakshman, Facebook)
12.10: Free lunch (sponsored by Last.fm)
13.10: Dynomite (Cliff Moon, Powerset)
13.50: HBase (Ryan Rawson, Stumbleupon)
14.30: Short break
14.40: Hypertable (Doug Judd, Zvents)
15.20: CouchDB (Chris Anderson, couch.io)
…

DB-ENGINES 2015年2月排名

排名	DBMS	Score
1	Cassandra	107.08
2	HBase	57.15

BDAS, the Berkeley Data Analytics Stack



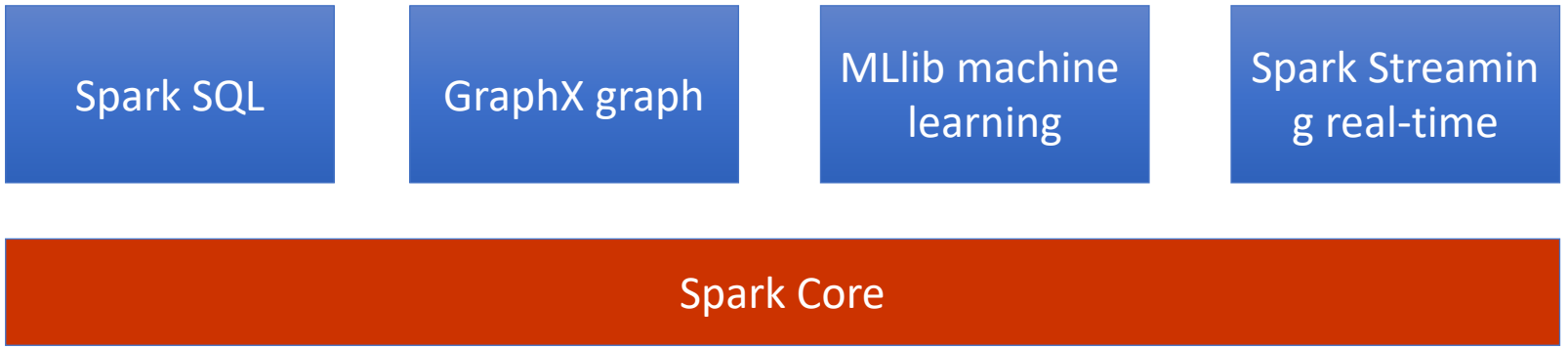
What is so great about Spark?

- We belive that Spark is the first system that allows a general-pupose programming language to be used at interactive speeds for in-memory data mining on clusters.

Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauly, Michael J. Franklin, Scott Shenker, Ion Stoica: Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. NSDI2012: 15-28

OK, but what exactly is Spark?

- Distributed data analytics engine, generalizing Map Reduce
- Core engine, with streaming, SQL, machine learning, and graph processing modules

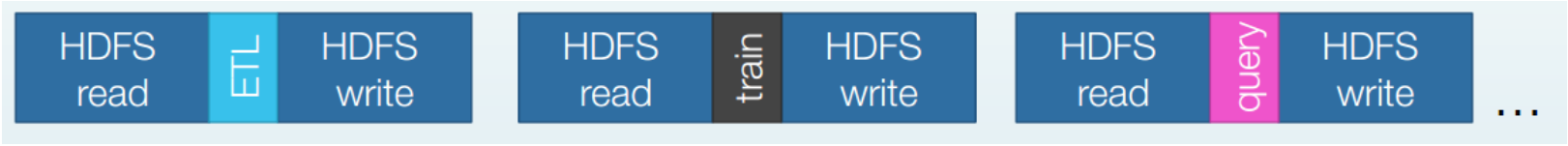


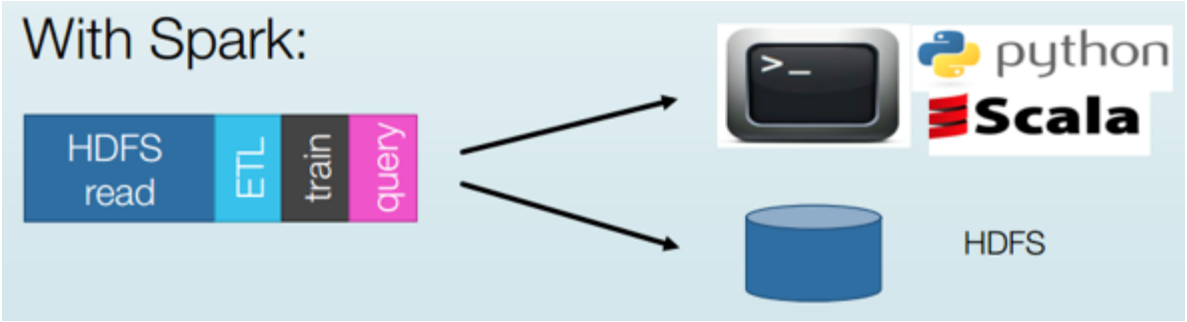
Spark Core:
RDDs, Transformations & Actions

- RDDs
  - Distributed collection of objects
  - Can be cached in memory
- Built via parallel transformations (map, filter, …)
  - Automatically rebuilt on failure based on lineage
- DAGs of RDDs and Transformations can be (lazily) executed via actions
  - Examples: Export to HDFS, count number of objects

Why Application Developers love Spark

- Building a real-world big data application without and with Spark:

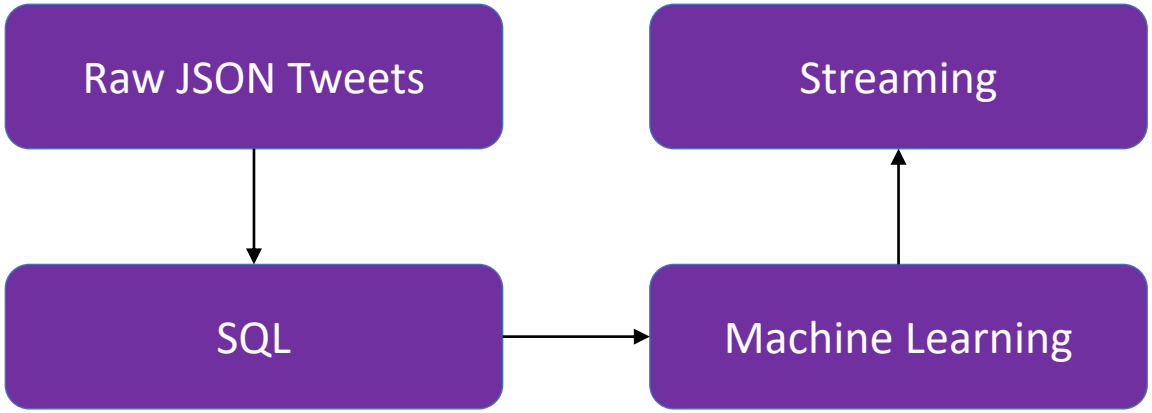




Interactive analysis

An Example App

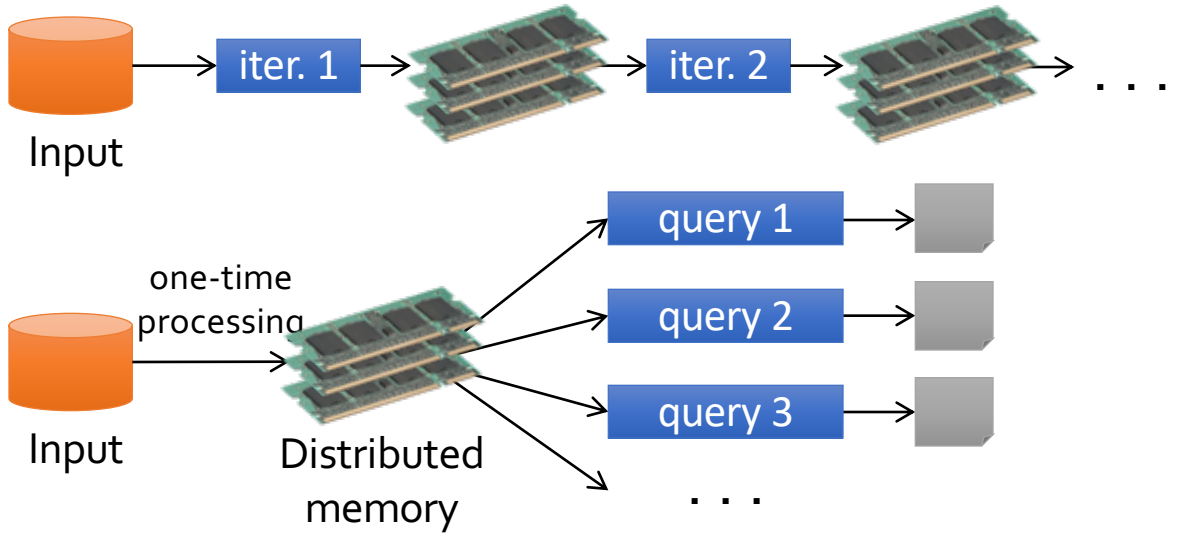
- Building a real-world big data application without and with Spark:



Fits on one slide

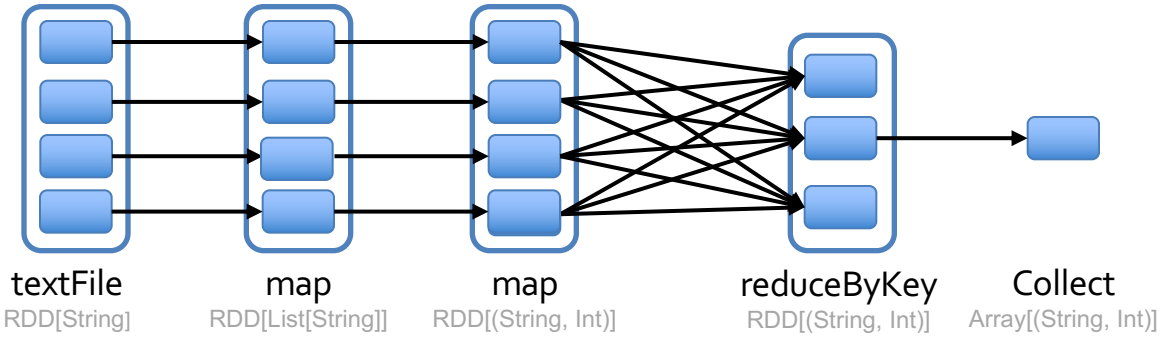
```
import org.apache.spark.sql._
val ctx = new org.apache.spark.sql.SQLContext(sc)
val tweets = sc.textFile("hdfs:/twitter")
val tweetTable = JsonTable.fromRDD(sqlContext, tweets, Some(0.1))
tweetTable.registerAsTable("tweetTable")

ctx.sql("SELECT text FROM tweetTable LIMIT 5").collect.foreach(println)
ctx.sql("SELECT lang, COUNT(*) AS cnt FROM tweetTable \
  GROUP BY lang ORDER BY cnt DESC LIMIT 10").collect.foreach(println)
val texts = sql("SELECT text FROM tweetTable").map(_.head.toString)

def featurize(str: String): Vector = { ... }
val vectors = texts.map(featurize).cache()
val model = KMeans.train(vectors, 10, 10)

sc.makeRDD(model.clusterCenters, 10).saveAsObjectFile("hdfs:/model")
val ssc = new StreamingContext(new SparkConf(), Seconds(1))

val model = new KMeansModel(
    ssc.sparkContext.objectFile(modelFile).collect())

// Streaming
val tweets = TwitterUtils.createStream(ssc, /* auth */)
val statuses = tweets.map(_.getText)
val filteredTweets = statuses.filter {
    t => model.predict(featurize(t)) == clusterNumber
}
filteredTweets.print()

ssc.start()
```

RDD模型：用内存保存数据



RDD模型：数据的抽象

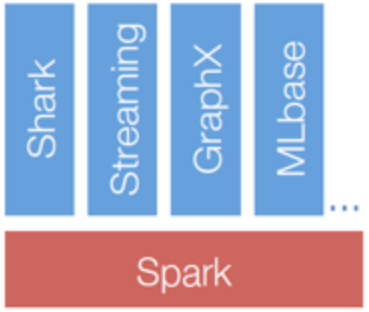
```
sc.textFile("/some-hdfs-data")                  RDD[String]
.map(line => line.split("\t"))                  RDD[List[String]]
.map(parts => (parts[0], int(parts[1])))        RDD[(String, Int)]
.reduceByKey(_ + _, 3)                          RDD[(String, Int)]
.collect()                                      Array[(String, Int)]
```



比MapReduce更快、更强！

	MapReduce	Spark
框架开销	执行容器多次创建与销毁， 基于心跳的通信机制	一次创建多次使用， 基于事件驱动的通信机制
数据IO	从HDFS（硬盘）中读取， 中间结果写入到HDFS中	从HDFS中读取至内存， 每次迭代从内存中读取， 中间结果写入到内存
Shuffle Sort	必须	可选
计算操作	map、reduce、combine	map、reduce、combine、join、 groupBy、union、intersection、 distinct、……

BDAS的优势



- 综合性的解决方案：在统一的框架内开发大数据应用
  - 对用户：易用性提升
  - 对开发者：代码精简（基于Spark的分布式程序的代码量通常仅为同类程序的十分之一）
- 高效的解决方案：BDAS的目标是快速处理大量数据

	主流解决方案	BDAS解决方案	BDAS的优势
文件系统	HDFS	Tachyon	数据读写速度提高**300**倍
MapReduce	Hadoop	Spark	运行速度提高**10-100**倍
SQL查询	Hive	Spark SQL	查询速度提高**40**倍
处理数据流	Storm	Spark Streaming	处理速度提高**2**倍

大数据生态系统

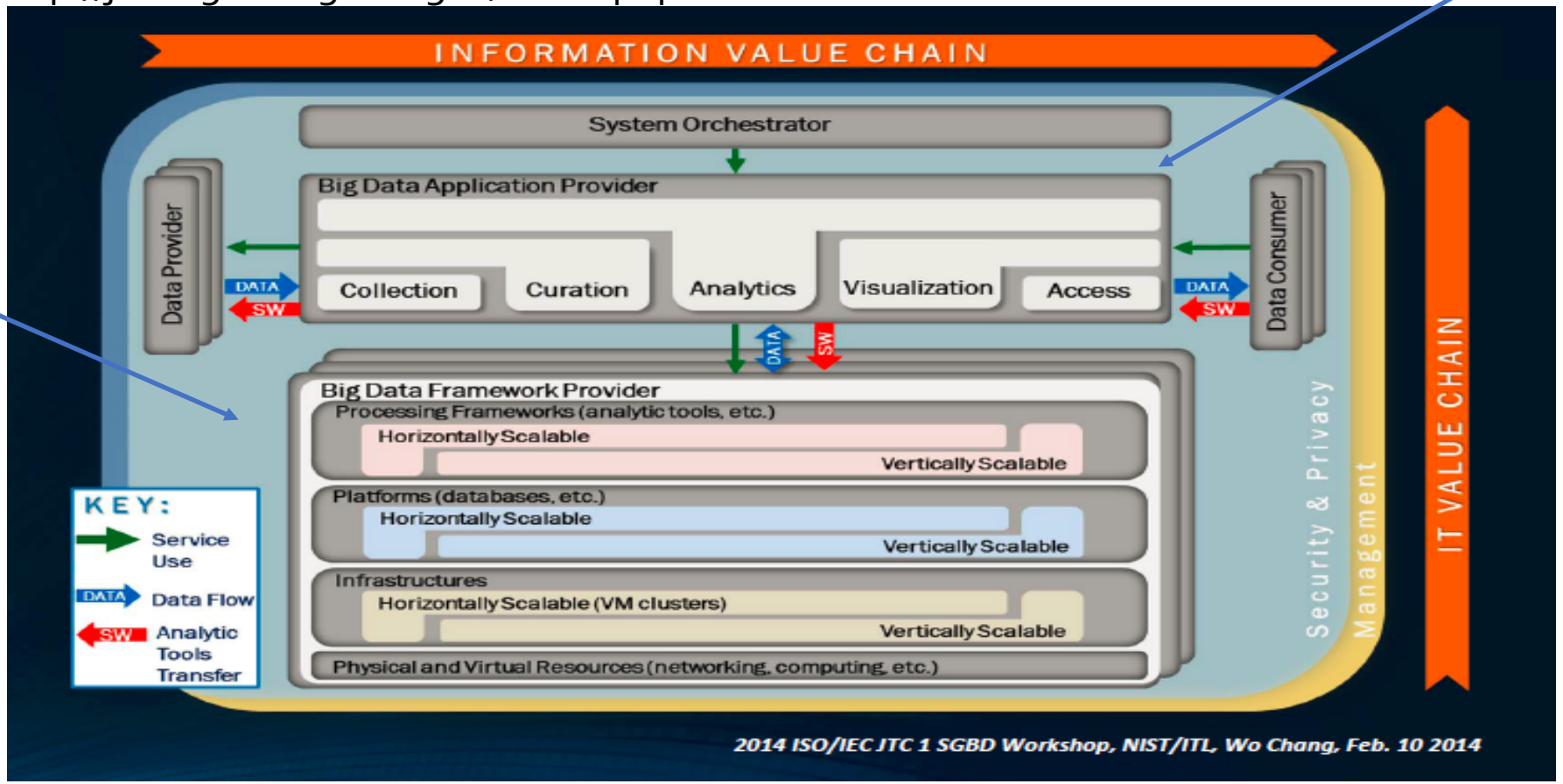

Data Analysis & Platforms
Hadoop
ParAccel
Storm
HPCC Systems
GridGain
Dremel
Hortonworks
Apache Drill
calpont
ORACLE TIMESTEN
HD
Zettaset
Databases / Data warehousing
INFOBRIGHT
Cassandra
HBASE
Hibari
riak
Infinispan
Bigdata®
orientDB
Neo4j
HYPERTABLE
redis
Globals
Operational
Versant JPA
MarkLogic
mobject
Multivalue database
Rocket
U2 REVELATION
northgate
QM
jBASE INTERNATIONAL
Business Intelligence
talend
open data solutions
JASPERSOFT
SpagoBI
Jedox
pentaho
Palo
BIRT Exchange
ACTUATE
KNIME
Data Mining
RAPID MINER
mahout
orange
RAPID ANALYTICS
WEKA
KEEL
togaware
Social
Apache Kafka
ThinkUp
Corona
Big Data search
Lucene
Apache Solr
elasticsearch.
Data aggregation
oqoop
zohmg
Multidimensional
GT.M
SciDB
rasdaman
raster data manager
KeyValue
EROSPIKE
leveldb
GENIEDB
Chordless
Beta
Tokyo Cabinet
Scalaris
SCALIEN
Project Voldemort
A distributed database.
hamsterdb
RAPTORDB
FairCom
STSDB
HyperDex
IQLECT
OpenLDAP
ioremap.net
STORAGE AND BEYOND
Document Store
mongoDB
CouchBase
CouchDB
Raven DB
CLUSTERPOINT
RaptorDB
EJDB
djon DB
JasDB
SchemafreeDB
sisodb
denso db
Graphs
Gephi
InfiniteGraph
FlockDB
AllegroGraph® 4.9
GraphBuilder
Gremlin
INFO GRID
HYPERGRAPHDB
dex
GraphBase
BrightstarDB
Grid Solutions
GIGASPACES
HAZELCAST
Galaxy
Object databases
db4objects
ZOPE
NEOPPOD
STARCOUNTER
Magma
Sterling
Picolisp
siaqodb
MORANTEX
INFORMATION SYSTEMS
EyeDB
HSS Database™
RAMER D
NDatabase
C# Lightweight Object Database
Multimodel
ArangoDB
alchemydatabase
XML Databases
existdb
BASE X
Qizx
sedna
xindice
Created by: www.bigdata-startups.com

大数据系统参考框架

http://jtc1bigdatasg.nist.gov/home.php

大数据生命周期

大数据系统软件栈



大数据技术挑战

Beckman Report

October 14-15, 2013

beckman.cs.wisc.edu/

Daniel Abadi, Rakesh Agrawal, Anastasia Ailamaki, Magdalena Balazinska, Philip A. Bernstein, Michael J. Carey, Surajit Chaudhuri, Jerey Dean, AnHai Doan, Michael J. Franklin, Johannes Gehrke, Laura M. Haas, Alon Y. Halevy, Joseph M. Hellerstein, Yannis E. Ioannidis, H.V. Jagadish, Donald Kossmann, Samuel Madden, Sharad Mehrotra, Tova Milo, Jerey F. Naughton, Raghu Ramakrishnan, Volker Markl, Christopher Olston, Beng Chin Ooi, Christopher Re, Dan Suciu, Michael Stonebraker, Todd Walter, Jennifer Widom

Past DB Submits

- 1988→Future Directions in DBMS Research-The Laguna Beach Participants
- 1990→Database Systems: Achievements and Opportunities
- 1995→Database Research: Achievements and Opportunities into the 21st Century
- 1996→Strategic Directions in Database Systems-Breaking Out of the Box
- 1998→The Asilomar Report on Database Research
- 2003→The Lowell DB Research Self-Assessment
- 2008→The Claremont Report

The Claremont Report on Database Research

- This is the 7th submit.
- Just before the submission deadline of CIDR 2009.
- Big heads are invited to attend.

- Participants(27):
- Rakesh Agrawal, Anastasia Ailamaki, Philip A. Bernstein, Eric A. Brewer, Michael J. Carey, Surajit Chaudhuri, AnHai Doan, Daniela Florescu, Michael J. Franklin, Hector Garcia-Molina, Johannes Gehrke, Le Gruenwald, Laura M. Haas, Alon Y. Halevy, Joseph M. Hellerstein, Yannis E. Ioannidis, Hank F. Korth, Donald Kossmann, Samuel Madden, Roger Magoulas, Beng Chin Ooi, Tim O'Reilly, Raghu Ramakrishnan, Sunita Sarawagi, Michael Stonebraker, Alexander S. Szalay, Gerhard Weikum

http://db.cs.berkeley.edu/claremont/

Trends Reported in 2008 Claremont

- We are at a turning point in the history of the field, due both to an explosion of data and usage scenarios, and to major shifts in computing hardware and platforms.
- Motivating factors:
  - Breadth of excitement about Big Data
  - Data analysis as a profit center
  - Ubiquity of structured and unstructured data
  - Expanded developer demands
  - Architectural shifts in computing

Opportunities Reported in 2008

- Revisiting Database Engines
- Declarative Programming for Emerging Platforms
- The Interplay of Structured and Unstructured Data
- Cloud Data Services
- Mobile Applications and Virtual Worlds

大数据在泥泞中前行 - Reported in Beckman Report

- 人们普遍认识到的数据“大”（Volume），不是数据科学面临的全部挑战，甚至不是主要挑战。来自不同数据源的、不同类型、不同语义（Variety）的数据集合的深度综合与融合问题远没有解决，同时，物联网、传感网、穿戴设备等机器数据的快速到达（Velocity），对数据处理的时效性提出了更大的挑战，除此之外数据隐私与可用性（包括数据质量）问题更是存在挑战
- 数据科学（包括大数据技术）的创新与探索刚刚起步，并行进在泥泞当中