DATAGURU 炼数成金

Hadoop数据分析平台 第1周

2012.8.17

课程目标

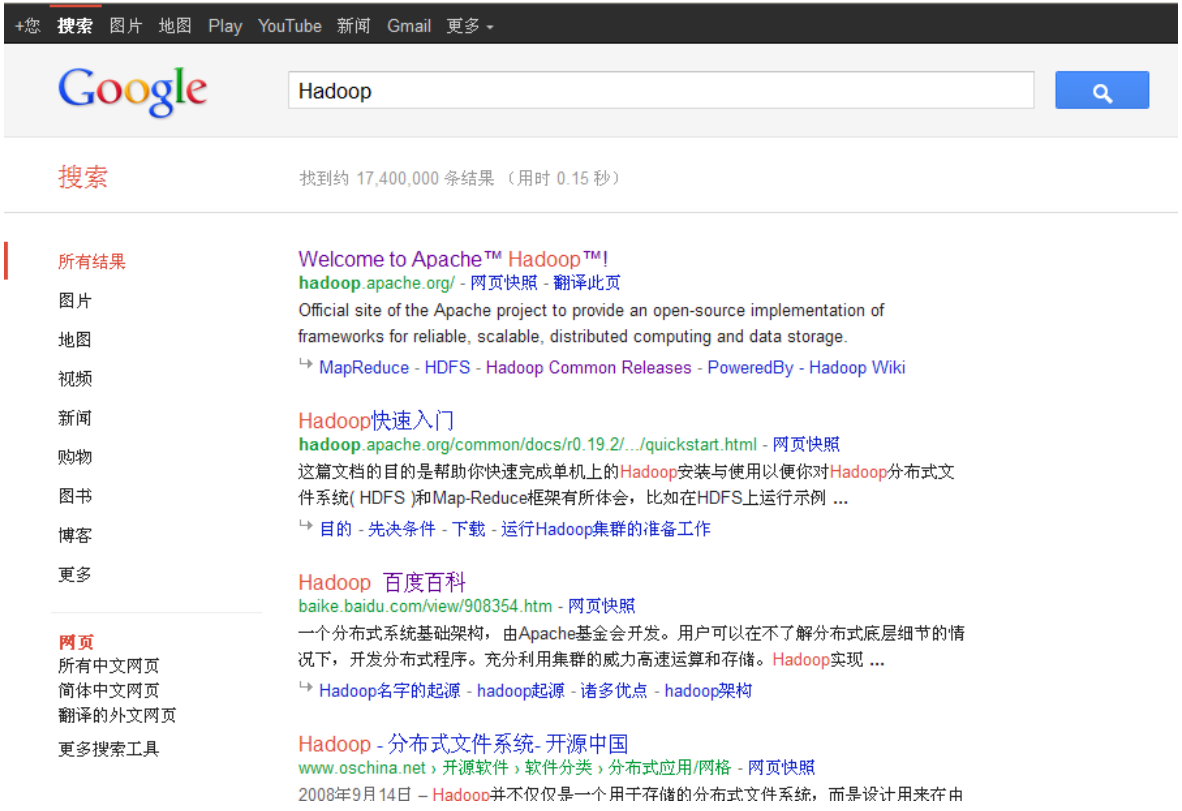
- 能独立熟练完成Hadoop的安装及熟悉Hadoop的配置与管理
- 熟练地在Hadoop和操作系统以及关系型数据库之前传递数据
- 能独立制定数据集成方案
- 熟练地向Hadoop提交作业以及查询作业运行情况
- 了解Map-Reduce原理，能书写Map-Reduce程序
- 了解HDFS原理，能熟练地对HDFS中的文件进行管理
- 能独立完成pig的安装并且利用pig做简单的数据分析工作
- 能独立完成Hbase的安装和配置
- 了解Hbase的原理并能进行简单的shell操作
- 能独立完成Hive的安装和配置
- 了解Hive的原理及进行HiveQL操作

一个典型的实验环境

- 服务器：ESXi，可以在上面部署10多台虚拟机，能同时启动4台
- PC：要求linux环境或windows+Cygwin，linux可以是standalone或者使用虚拟机
- SSH：windows下可以使用SecureCRT或putty等ssh client程序，作用是用来远程连接linux服务器，linux下可以直接使用ssh命令
- Vmware client：用于管理ESXi
- Hadoop：使用0.20.2

Hadoop的思想之源：Google

- Google搜索引擎，Gmail，安卓，AppspotGoogle Maps，Google earth，Google学术，Google翻译，Google+，下一步Google what？？



2012.8.17

Google的低成本之道

- 不使用超级计算机，不使用存储（淘宝的去i，去e，去o之路）
- 大量使用普通的pc服务器（去掉机箱，外设，硬盘），提供有冗余的集群服务
- 全世界多个数据中心，有些附带发电厂
- 运营商向Google倒付费

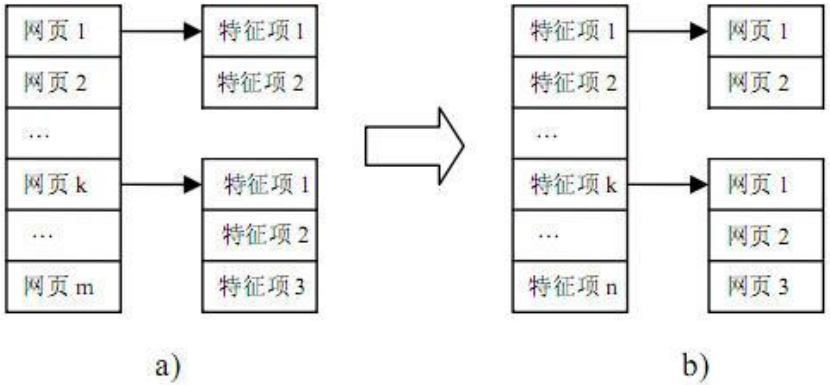
集装箱数据中心

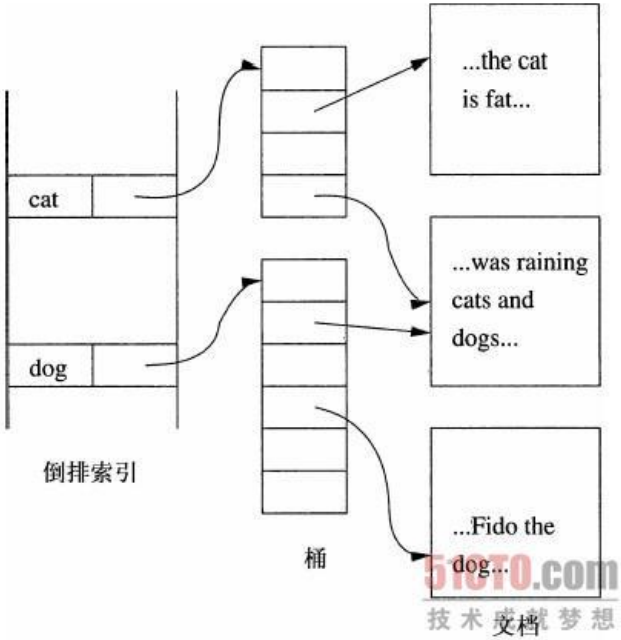
- 位于 Mountain View， Calif 总部的数据中心
- 总功率为10000千瓦，拥有45个集装箱，每个集装箱中有1160台服务器，该数据中心的能效比为1.25（ PUE 为 1 表示数据中心没有能源损耗，而根据2006年的统计，一般公司数据中心的能效比为 2.0 或更高。Google 的 1.16 已经低于美国能源部2011年的1.2 的目标）

Google面对的数据和计算难题

- 大量的网页怎么存储？
- 搜索算法
- Page-Rank计算问题

倒排索引





2012.8.17

Page Rank

- 这是Google最核心的算法，用于给每个网页价值评分，是Google“在垃圾中找黄金”的关键算法，这个算法成就了今天的Google

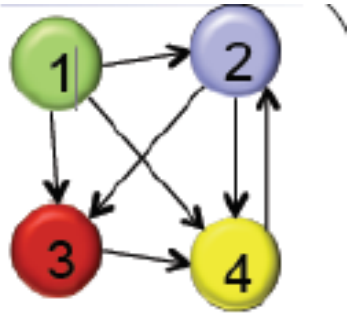
PageRank vector **q** is defined as $q = Gq$

where $G = \alpha S + (1-\alpha)\frac{1}{n}U$

- S is the destination-by-source stochastic matrix,
- U is all one matrix.
- n is the number of nodes
- α is the weight between 0 and 1 (e.g., 0.85)

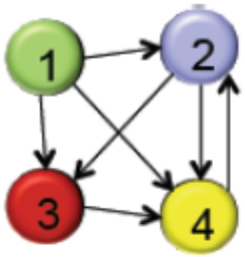
Algorithm: Iterative powering for finding the first eigen-vector

$$q^{next} = Gq^{cur}$$

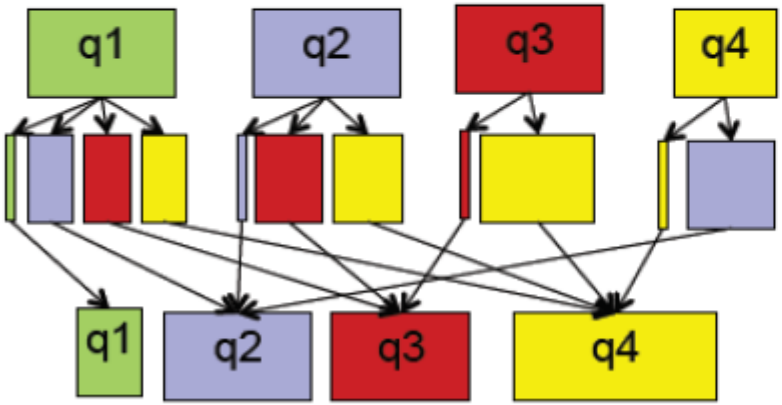


$$G = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 1 \\ 1/3 & 1/2 & 0 & 0 \\ 1/3 & 1/2 & 1 & 0 \end{bmatrix}$$

Map-reduce思想：计算PR



Map: distribute PageRank q_i



Reduce: update new PageRank

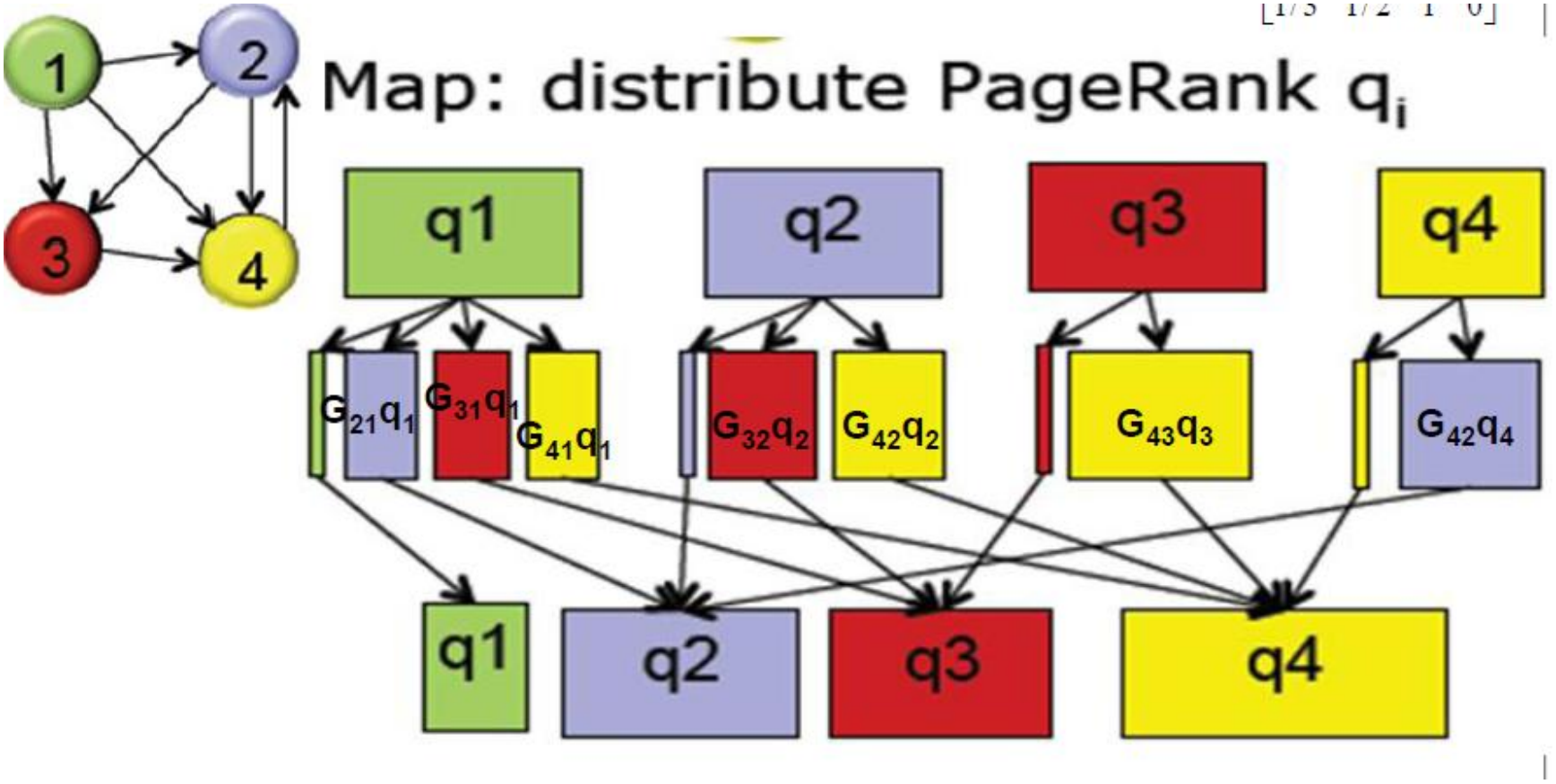
PageRank Map()

- Input:
  - key = page x,
  - value = (q_x, links[$y_1 \ldots y_m$])
- Output:
  - key = page x,
  - value = $partial_x$

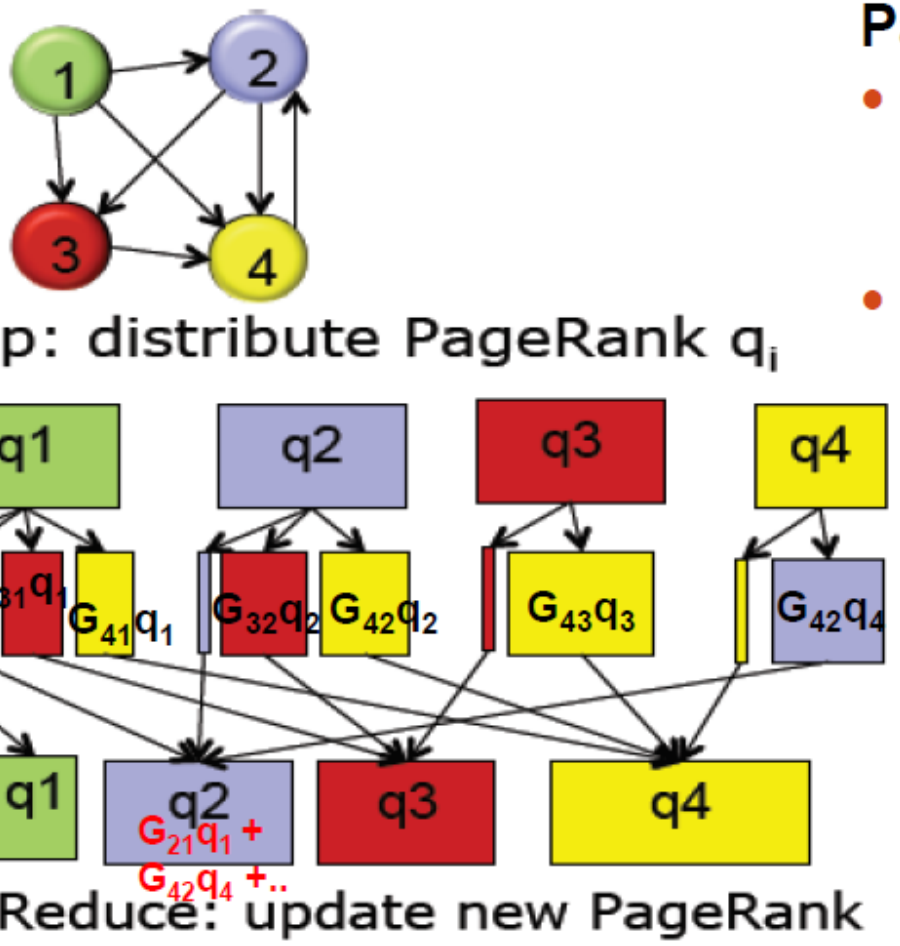
  1. Emit(x, 0) //guarantee all pages will be emitte
  2. For each outgoing link y_i:

     Emit(y_i, $G_{ix}q_x$)

计算PR值


1
2
3
4
Map: distribute PageRank q_i
q1
q2
q3
q4
$G_{21}q_1$
$G_{31}q_1$
$G_{41}q_1$
$G_{32}q_2$
$G_{42}q_2$
$G_{43}q_3$
$G_{42}q_4$
q1
q2
q3
q4

计算PR值



PageRank Reduce()

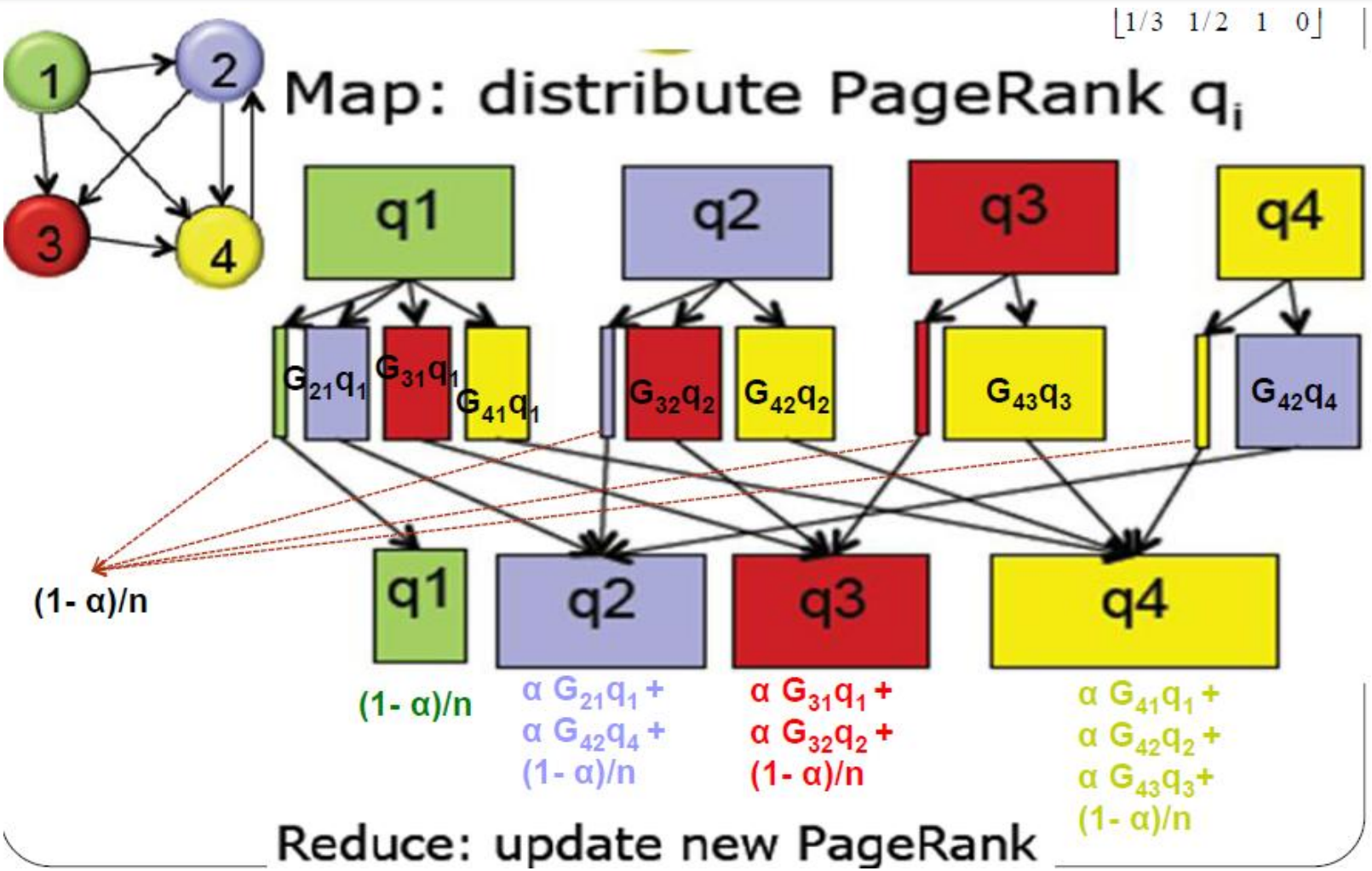
- Input:
  - key = page x,
  - value = the list of [$partial_x$]
- Output:
  - key = page x,
  - value = PageRank q_x

1. $q_x = 0$
2. For each partial value d in the list:
   q_x += d
3. q_x = α q_x + (1- α)/n
4. Emit(x, q_x)

$$q^{next} = Gq = \alpha Sq + (1-\alpha)\frac{1}{n}Uq$$

2012.8.17

计算PR值


1/3 1/2 1 0
1
2
3
4
Map: distribute PageRank q_i
q1
q2
q3
q4
$G_{21}q_1$
$G_{31}q_1$
$G_{41}q_1$
$G_{32}q_2$
$G_{42}q_2$
$G_{43}q_3$
$G_{42}q_4$
(1- α)/n
q1
q2
q3
q4
(1- α)/n
α $G_{21}q_1$ +
α $G_{42}q_4$ +
(1- α)/n
α $G_{31}q_1$ +
α $G_{32}q_2$ +
(1- α)/n
α $G_{41}q_1$ +
α $G_{42}q_2$ +
α $G_{43}q_3$ +
(1- α)/n
Reduce: update new PageRank

Google带给我们的关键技术和思想

- GFS
- Map-Reduce
- Bigtable（后面讲）

Hadoop的源起——Lucene

- Doug Cutting开创的开源软件，用java书写代码，实现与Google类似的全文搜索功能，它提供了全文检索引擎的架构，包括完整的查询引擎和索引引擎
- 早期发布在个人网站和SourceForge，2001年年底成为apache软件基金会jakarta的一个子项目
- Lucene的目的是为软件开发人员提供一个简单易用的工具包，以方便的在目标系统中实现全文检索的功能，或者是以此为基础建立起完整的全文检索引擎
- 对于大数量的场景，Lucene面对与Google同样的困难。迫使Doug Cutting学习和模仿Google解决这些问题的办法
- 一个微缩版：Nutch

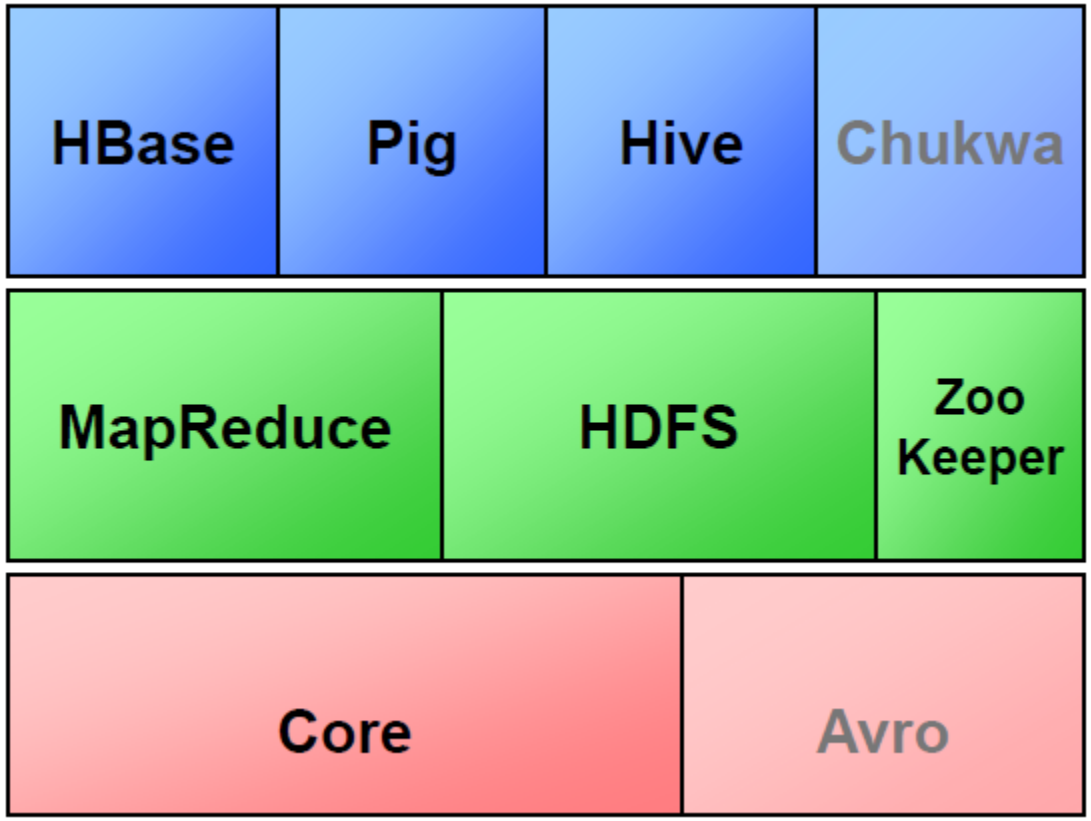
从lucene到nutch，从nutch到hadoop

- 2003-2004年，Google公开了部分GFS和Mapreduce思想的细节，以此为基础Doug Cutting等人用了2年业余时间实现了DFS和Mapreduce机制，使Nutch性能飙升
- Yahoo招安Doug Cutting及其项目
- Hadoop 于 2005 年秋天作为 Lucene的子项目 Nutch的 一部分正式引入Apache基金会。2006 年 3 月份，Map-Reduce 和 Nutch Distributed File System (NDFS) 分别被纳入称为 Hadoop 的项目中
- 名字来源于Doug Cutting儿子的玩具大象

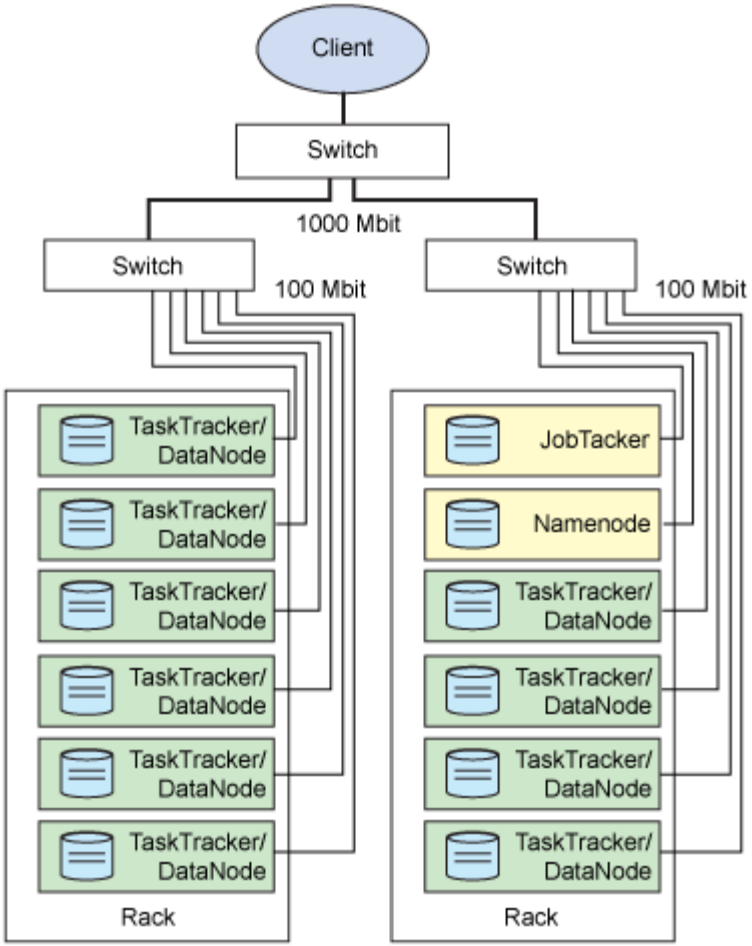
目前Hadoop达到的高度

- 实现云计算的事实标准开源软件
- 包含数十个具有强大生命力的子项目
- 已经能在数千节点上运行，处理数据量和排序时间不断打破世界纪录

Hadoop子项目家族

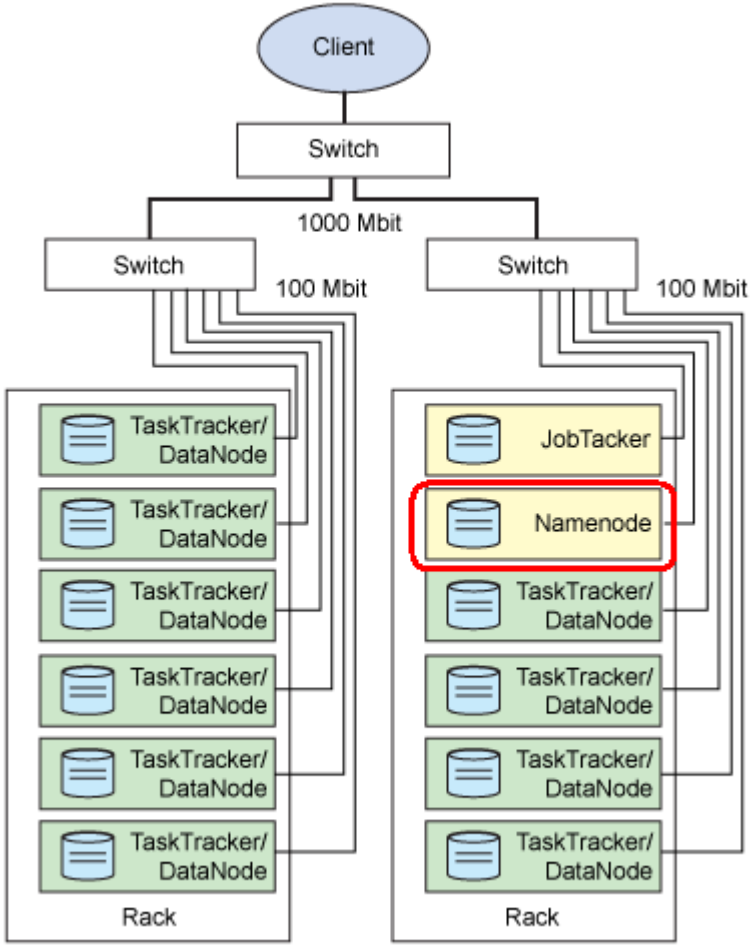


Hadoop的架构


Client
Switch
1000 Mbit
Switch
100 Mbit
Switch
100 Mbit
TaskTracker/
DataNode
TaskTracker/
DataNode
TaskTracker/
DataNode
TaskTracker/
DataNode
TaskTracker/
DataNode
TaskTracker/
DataNode
Rack
JobTacker
Namenode
TaskTracker/
DataNode
TaskTracker/
DataNode
TaskTracker/
DataNode
TaskTracker/
DataNode
Rack

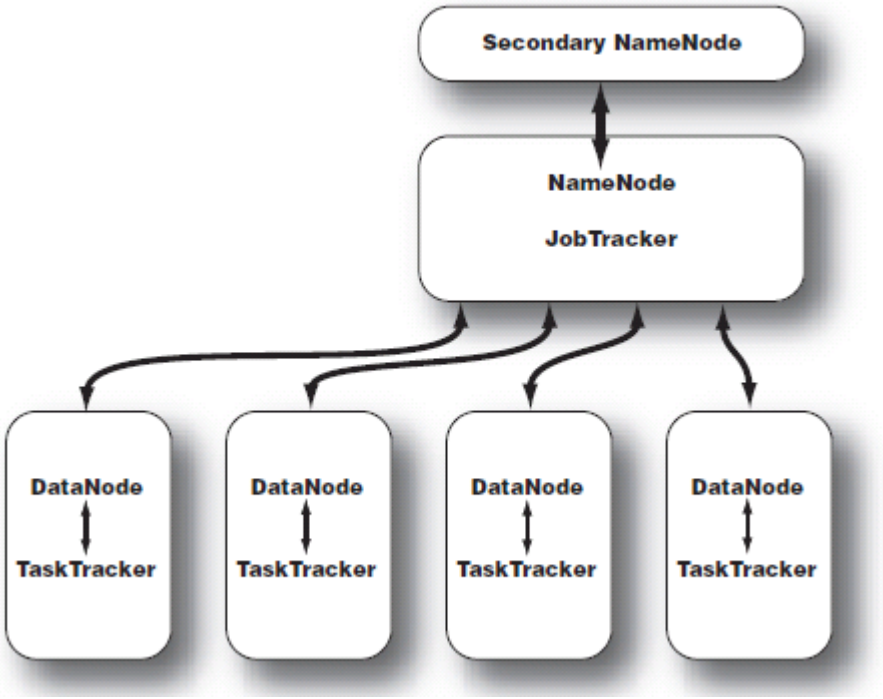
Namenode

- HDFS的守护程序
- 纪录文件是如何分割成数据块的，以及这些数据块被存储到哪些节点上
- 对内存和I/O进行集中管理
- 是个单点，发生故障将使集群崩溃



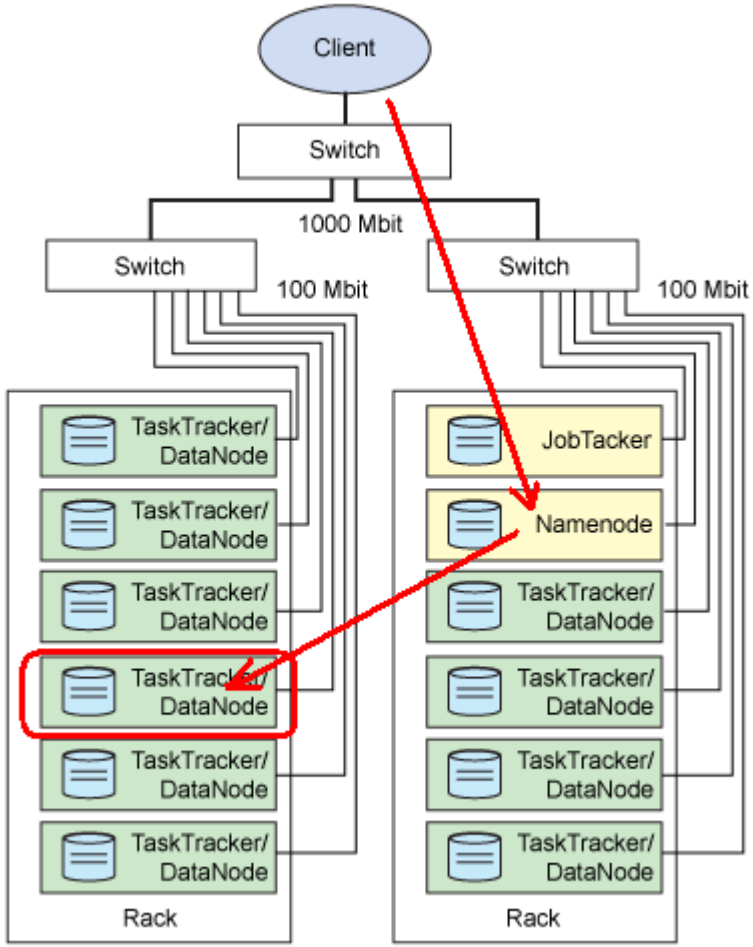
Secondary Namenode

- 监控HDFS状态的辅助后台程序
- 每个集群都有一个
- 与NameNode进行通讯，定期保存HDFS元数据快照
- 当NameNode故障可以作为备用NameNode使用



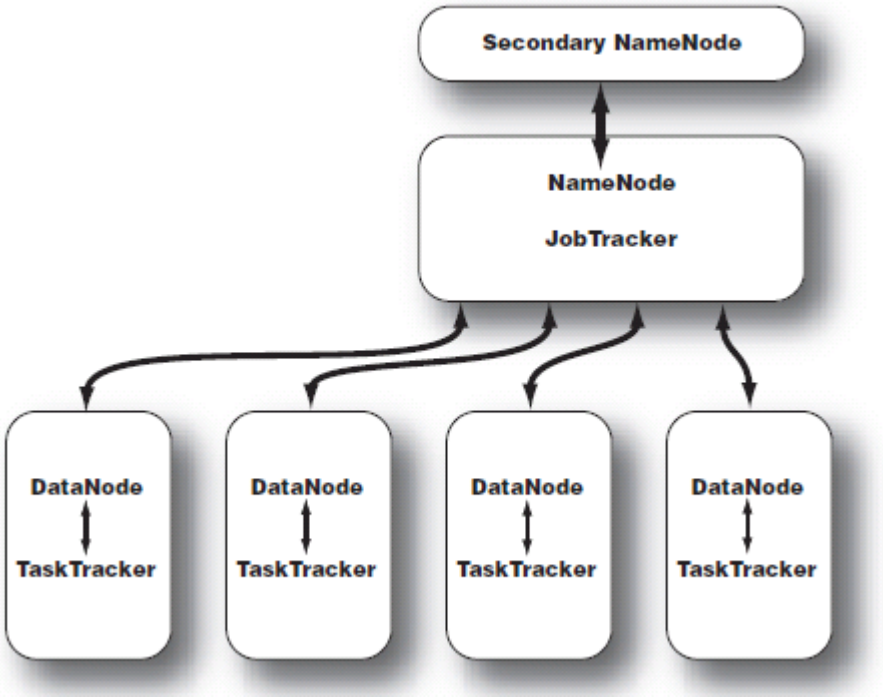
DataNode

- 每台从服务器都运行一个
- 负责把HDFS数据块读写到本地文件系统



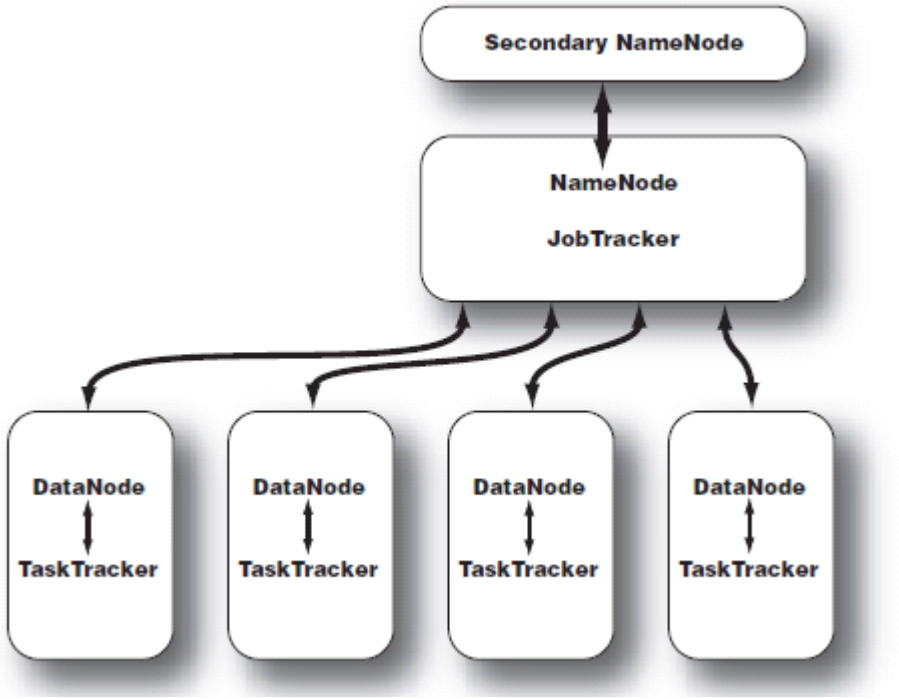
JobTracker

- 用于处理作业（用户提交代码）的后台程序
- 决定有哪些文件参与处理，然后切割task并分配节点
- 监控task，重启失败的task（于不同的节点）
- 每个集群只有唯一一个JobTracker，位于Master节点

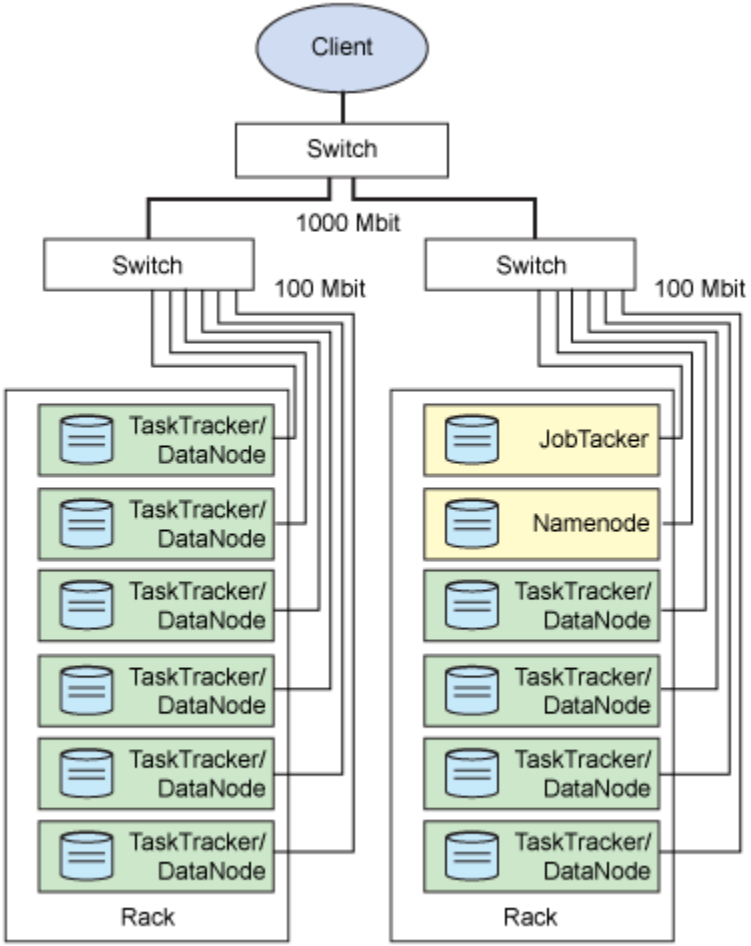


TaskTracker

- 位于slave节点上，与datanode结合（代码与数据一起的原则）
- 管理各自节点上的task（由jobtracker分配）
- 每个节点只有一个tasktracker，但一个tasktracker可以启动多个JVM，用于并行执行map或reduce任务
- 与jobtracker交互

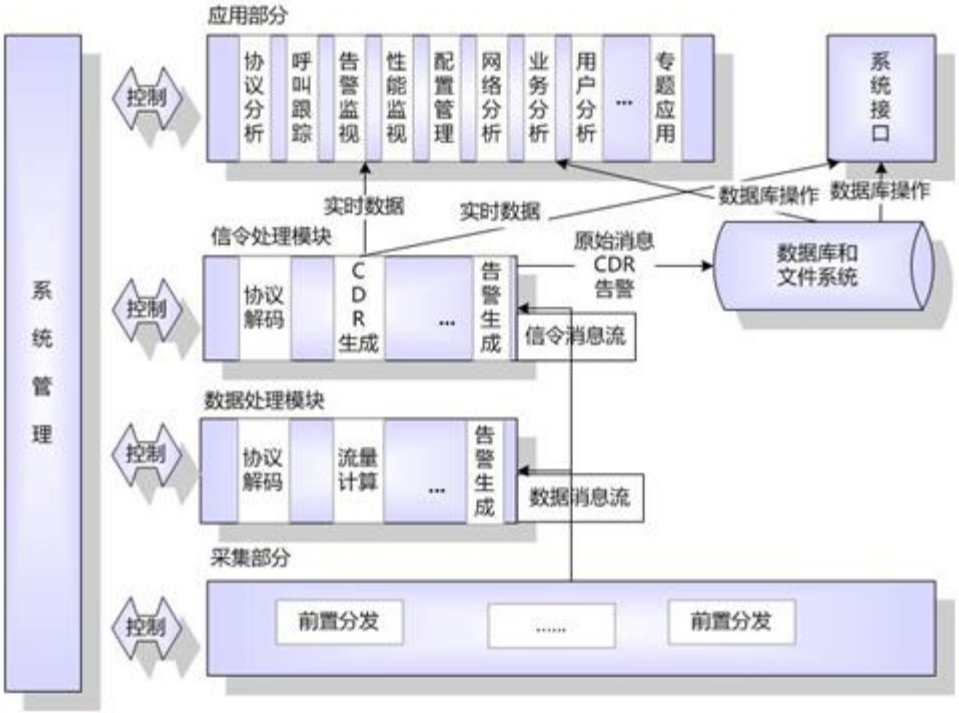


Master与Slave



- Master：Namenode、Secondary Namenode、Jobtracker。浏览器（用于观看 管理界面），其它Hadoop工具
- Slave：Tasktracker、Datanode
- Master不是唯一的

Why hadoop ?


应用部分
协议分析
呼叫跟踪
告警监视
性能监视
配置管理
网络分析
业务分析
用户分析
...
专题应用
系统接口
控制
实时数据
实时数据
数据库操作
数据库操作
信令处理模块
协议解码
CDR生成
...
告警生成
原始消息
CDR
告警
数据库和文件系统
信令消息流
系统管理
数据处理模块
协议解码
流量计算
...
告警生成
数据消息流
采集部分
前置分发
.....
前置分发

场景：电信运营商信令分析与监测

- 原数据库服务器配置：HP小型机，128G内存，48颗CPU，2节点RAC，其中一个节点用于入库，另外一个节点用于查询
- 存储：HP虚拟化存储，>1000个盘
- 数据库架构采用Oracle双节点RAC
- 问题：1 **入库瓶颈** 2 **查询瓶颈**

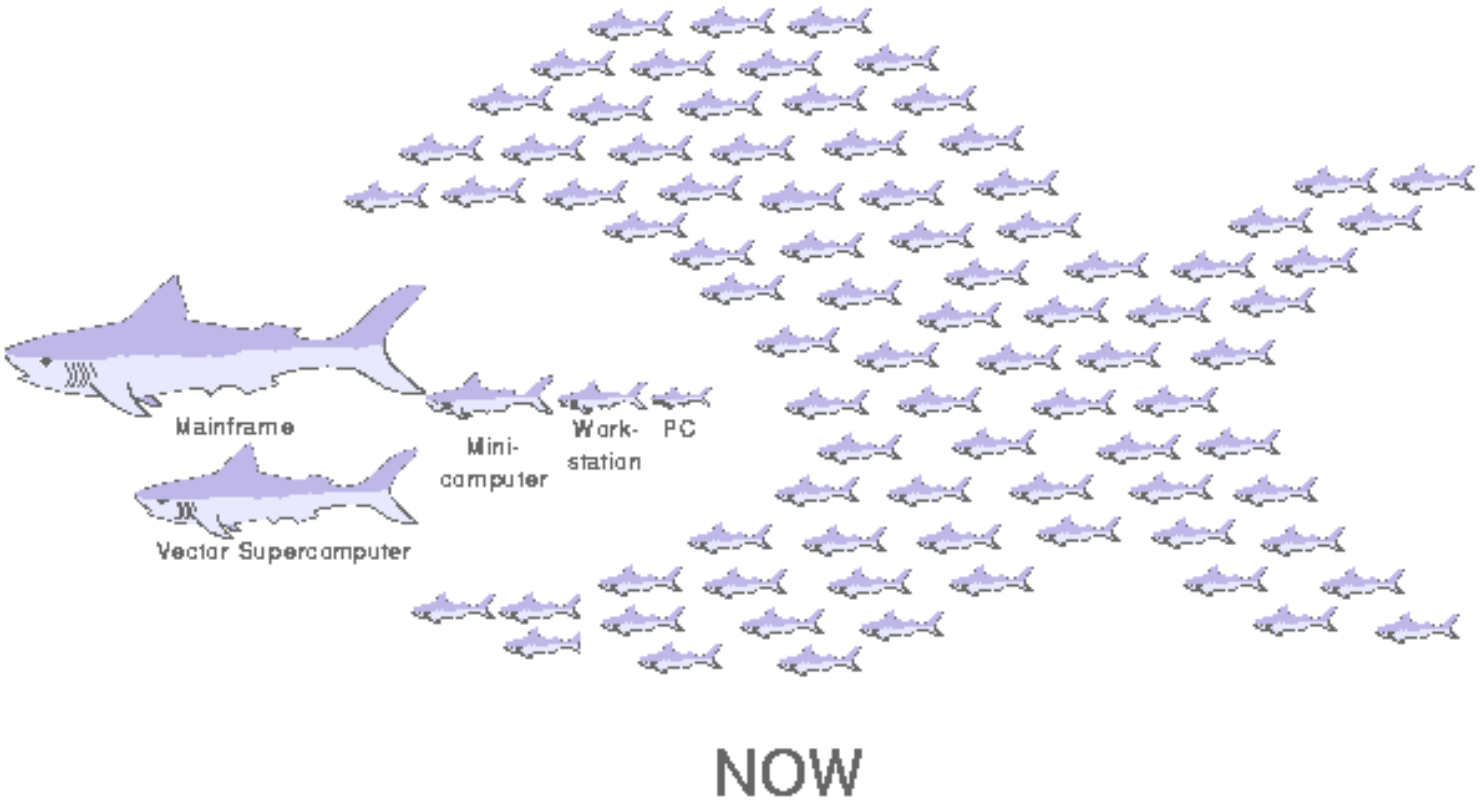
数据分析者面临的问题

- 数据日趋庞大，无论是入库和查询，都出现性能瓶颈
- 用户的应用和分析结果呈整合趋势，对实时性和响应时间要求越来越高
- 使用的模型越来越复杂，计算量指数级上升

数据分析者期待的解决方案

- 完美解决性能瓶颈，在可见未来不容易出现新瓶颈
- 过去所拥有的技能可以平稳过渡。比如SQL、R
- 转移平台的成本有多高？平台软硬件成本，再开发成本，技能再培养成本，维护成本

Hadoop的思想


Mainframe
Mini-
computer
Work-
station
PC
Vector Supercomputer
NOW

Why not Hadoop ?

- Java ?
- 难以驾驭 ?
- 数据集成困难 ?
- Hadoop vs Oracle

Hadoop体系下的分析手段

- 主流：Java程序
- 轻量级的脚本语言：Pig
- SQL技巧平稳过渡：Hive
- NoSQL：HBase

Thanks

FAQ时间