



云计算与数据挖掘

刘鹏 gloud@126.com

中国云计算: <http://www.chinacloud.cn>

中国网格: <http://www.chinagrid.net>

内 容 提 纲

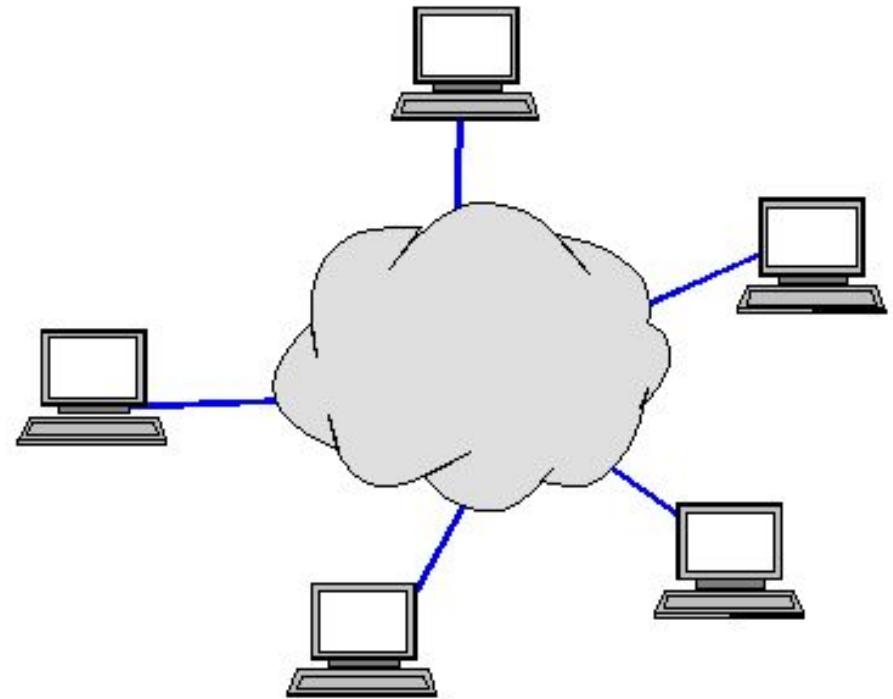
云计算概念与现状

»

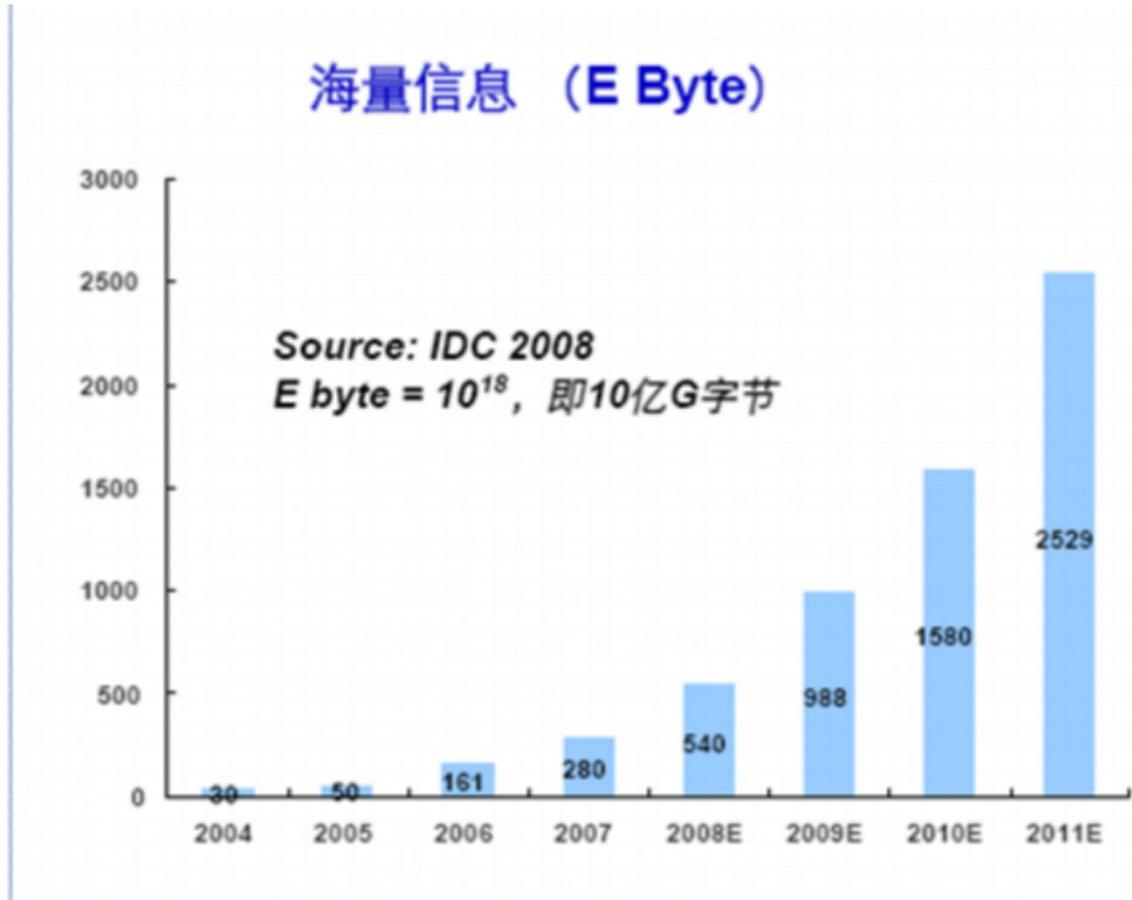
云计算的起源



The Network is the Computer™



云计算发展的驱动因素



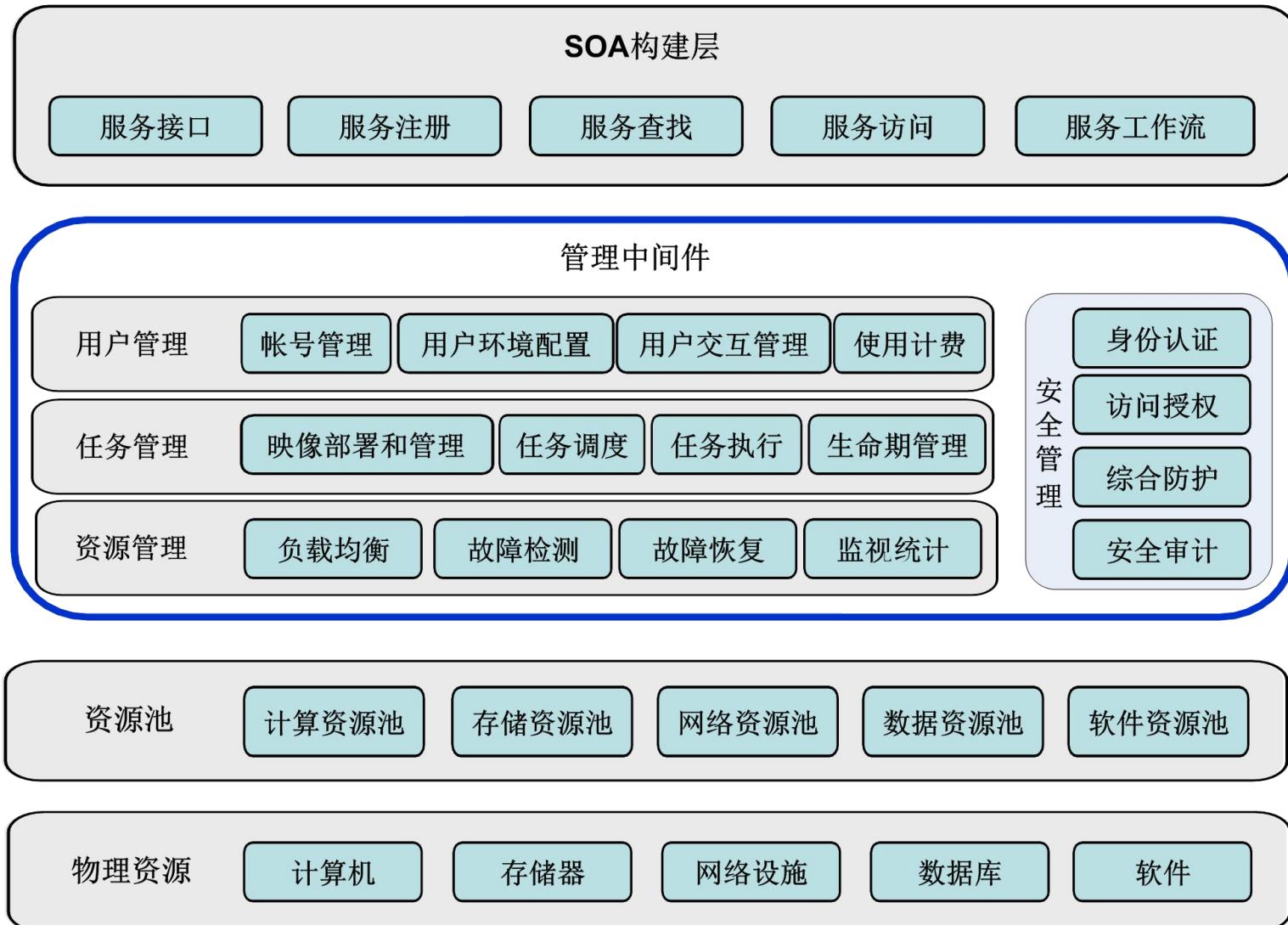
- **信息爆炸:** 2006年, 全球产生161 EB 的数据, 印成书是地球到太阳距离的10倍; 2007年, 全球产生280 EB 数据, 全世界平均每人45G; 而人类历史5000年的文字记载只有5EB



云计算的定义

云计算是一种商业计算模型。它将计算任务分布在大量计算机构成的资源池上，使各种应用系统能够根据需要获取计算力、存储空间和信息服务。

云计算技术体系结构



Google™



Google云计算关键技术



- Google文件系统GFS (Google File System)
- 并行数据处理MapReduce
- 结构化数据表BigTable
- 分布式锁管理Chubby







Google云计算原理

» 分布式文件系统GFS
Google File System

Google设计GFS的动机

- { Google需要一个支持海量存储的文件系统
 - 购置昂贵的分布式文件系统与硬件？



Google设计GFS的动机

} 为什么不使用当时现存的文件系统？

- Google所面临的问题与众不同
 ☞不同的工作负载，不同的设计优先级（廉价、不可靠的硬件）
- 需要设计与Google应用和负载相符的文件系统

GFS的假设与目标

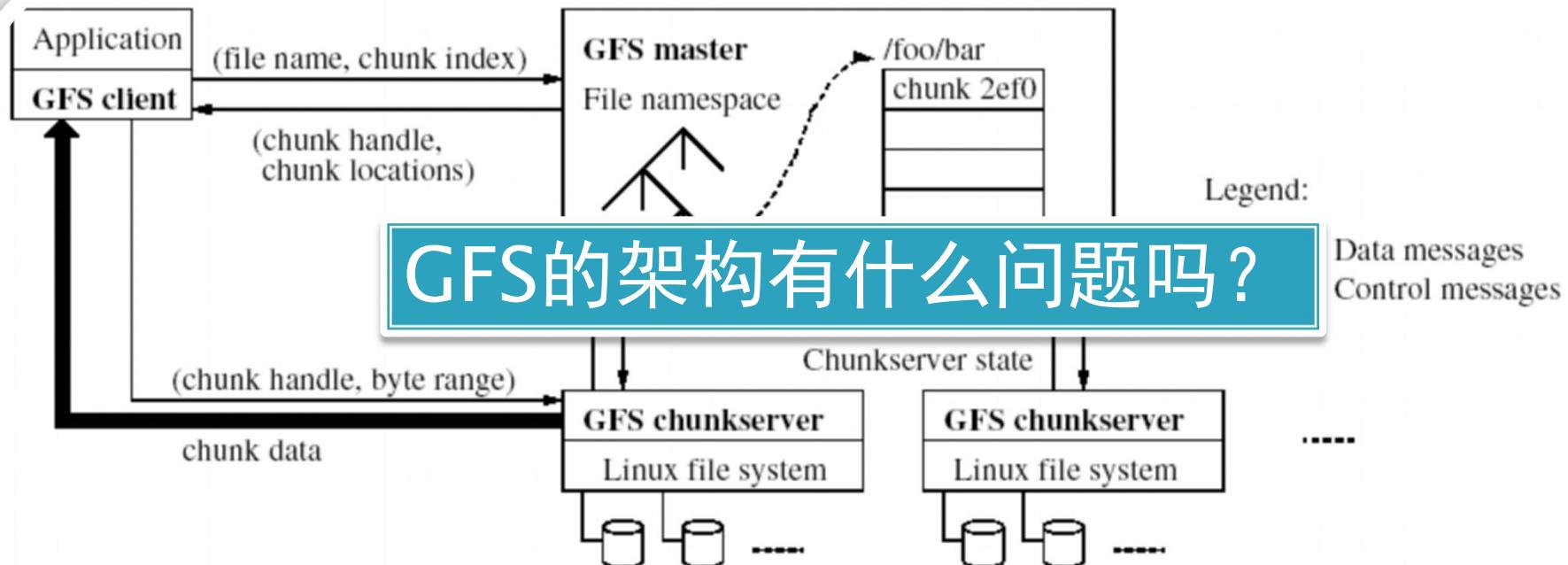
- { 硬件出错是正常而非异常
 - 系统应当由大量廉价、易损的硬件组成
 - 必须保持文件系统整体的可靠性
- { 主要负载是流数据读写
 - 主要用于程序处理批量数据，而非与用户的交互或随机读写
 - 数据写主要是“追加写”，“插入写”非常少
- { 需要存储大尺寸的文件
 - 存储的文件尺寸可能是GB或TB量级，而且应当能支持存储成千上万的大尺寸文件

GFS的设计思路

- { 将文件划分为若干块（Chunk）存储
 - 每个块固定大小（64M）
- { 通过冗余来提高可靠性
 - 每个数据块至少在3个数据块服务器上冗余
 - 数据块损坏概率？
- { 通过单个master来协调数据访问、元数据存储
 - 结构简单，容易保持元数据一致性
- { 无缓存
 - Why？

GFS的架构

} 单一Master, 若干ChunkServer



单一Master问题

- { 分布式系统设计告诉我们：
 - 这是单点故障
 - 这是性能瓶颈
- { GFS的解决办法
 - 单点故障问题

采用多个（如3个）影子Master节点进行热备，一旦主节点损坏，立刻选举一个新的主节点服务

单一Master问题

- { GFS的解决办法
 - 性能瓶颈问题

Simple, and good enough!

Master节点的任务

- { 存储元数据
- { 文件系统目录管理与加锁
- { 与ChunkServer进行周期性通信
 - 发送指令，搜集状态，跟踪数据块的完好性
- { 数据块创建、复制及负载均衡
 - 对ChunkServer的空间使用和访问速度进行负载均衡，平滑数据存储和访问请求的负载
 - 对数据块进行复制、分散到ChunkServer上
 - 一旦数据块冗余数小于最低数，就发起复制操作

Master节点的任务

} 垃圾回收

- 在日志中记录删除操作，并将文件改名隐藏
- 缓慢地回收隐藏文件
- 与传统文件删除相比更简单、更安全

} 陈旧数据块删除

- 探测陈旧的数据块，并删除

GFS架构的特点

} 采用中心服务器模式

- 可以方便地增加Chunk Server
- Master掌握系统内所有Chunk Server的情况，方便进行负载均衡
- 不存在元数据的一致性问题

GFS架构的特点

} 不缓存数据



GFS架构的特点

- { 在用户态下实现
 - 直接利用Chunk Server的文件系统存取Chunk， 实现简单
 - 用户态应用调试较为简单， 利于开发
 - 用户态的GFS不会影响Chunk Server的稳定性
- { 提供专用的访问接口
 - 未提供标准的POSIX访问接口
 - 降低GFS的实现复杂度

GFS的容错方法

{ GFS的容错机制

- Chunk Server容错

- 每个Chunk有多个存储副本（通常是3个），分别存储于不通的服务器上

- 每个Chunk又划分为若干Block（64KB），每个Block对应一个32bit的校验码，保证数据正确（若某个Block错误，则转移至其他Chunk副本）

GFS的性能

Cluster	A	B
Chuckservers	342	227
Available disk space	72 TB	180 TB
Used disk space	55 TB	155 TB
Number of Files	735 k	737 k
Number of Dead files	22 k	232 k
Number of Chunks	992 k	1550 k
Metadata at chuckservers	13 GB	21 GB
Metadata at master	48 MB	60 MB

Cluster	A	B
Read rate (last minute)	583 MB/s	380 MB/s
Read rate (last hour)	562 MB/s	384 MB/s
Read rate (since restart)	589 MB/s	49 MB/s
Write rate (last minute)	1 MB/s	101 MB/s
Write rate (last hour)	2 MB/s	117 MB/s
Write rate (since restart)	25 MB/s	13 MB/s
Master ops (last minute)	325 Ops/s	533 Ops/s
Master ops (last hour)	381 Ops/s	518 Ops/s
Master ops (since restart)	202 Ops/s	347 Ops/s

Google云计算原理

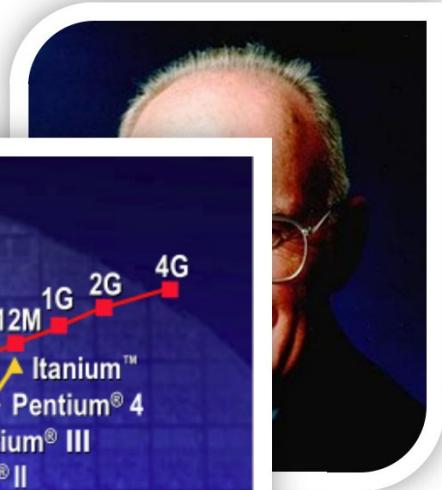
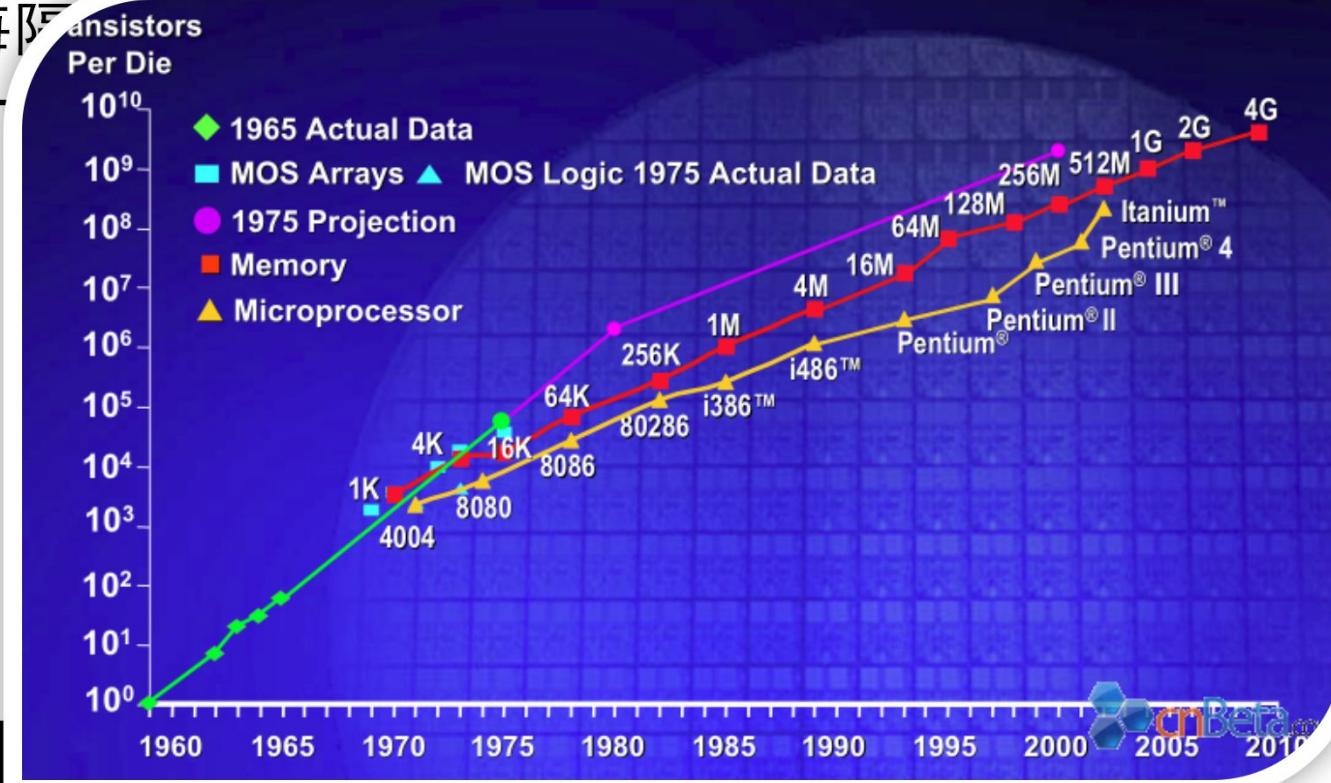
» 并行数据处理模型MapReduce

并行计算基础

{ 摩尔定律

- 集成电路芯片上所集成的电路的数目，

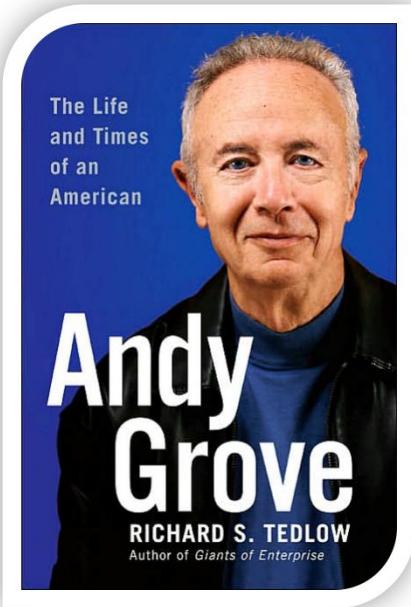
每



Gordon Moore

并行计算基础

- { “免费的性能大餐” ?
 - Andy given, and Bill taken away



Intel

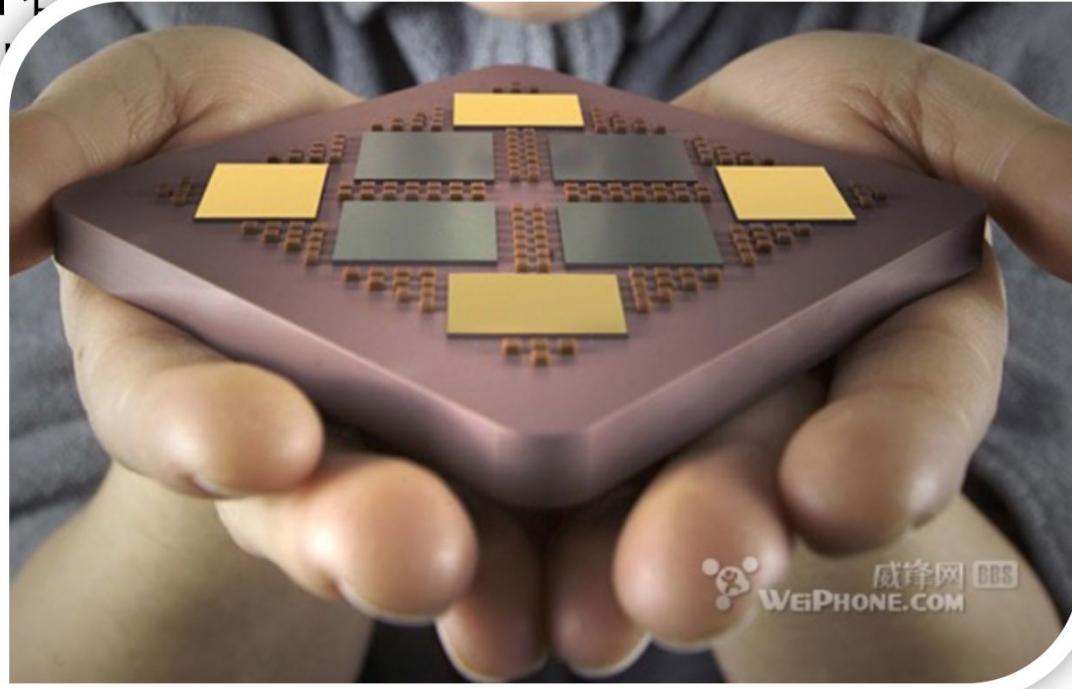


Microsoft

并行计算基础

{ 摩尔定律正在走向终结...

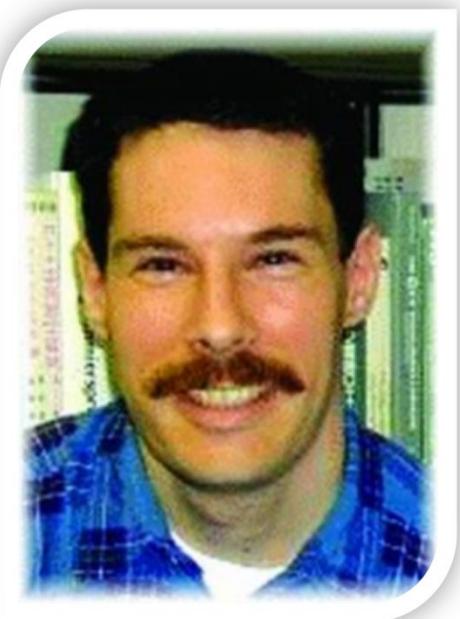
- 单芯片容纳晶体管的增加 叶生江生工廿廿日山西武
- CPU制
- CPU主
- 散热
- 功耗



未来的发展：多核

并行计算基础

- { 在多核时代生存，必须考虑并发问题
- { 不存在解决多核编程问题的银弹，
不存在可以简单地将并发编程问题化
解掉的工具， 开发高性能的并行程序
必须要求开发者从根本上改变其编程
方法
- { 从某种意义上来说， 这不仅仅仅是
要改变50年来顺序程序设计的工艺传统，
而且是要改变数百万年来人类顺序化思考问题的习
惯



Herb Sutter

并行计算基础

} 串行编程

- 早期的计算里，程序一般是被串行执行的
- 程序是指令的序列，在单处理器的机器里，程序从开始到结束，这些指令一条接一条的执行

} 并行编程

- 一道处理可以被划分为几部分,然后它们可以并发地执行
- 各部分的指令分别在不同的CPU上同时运行，这些CPU可以存在于单台机器中,也可以存在于多台机器上,它们通过连接起来共同运作

并行计算基础

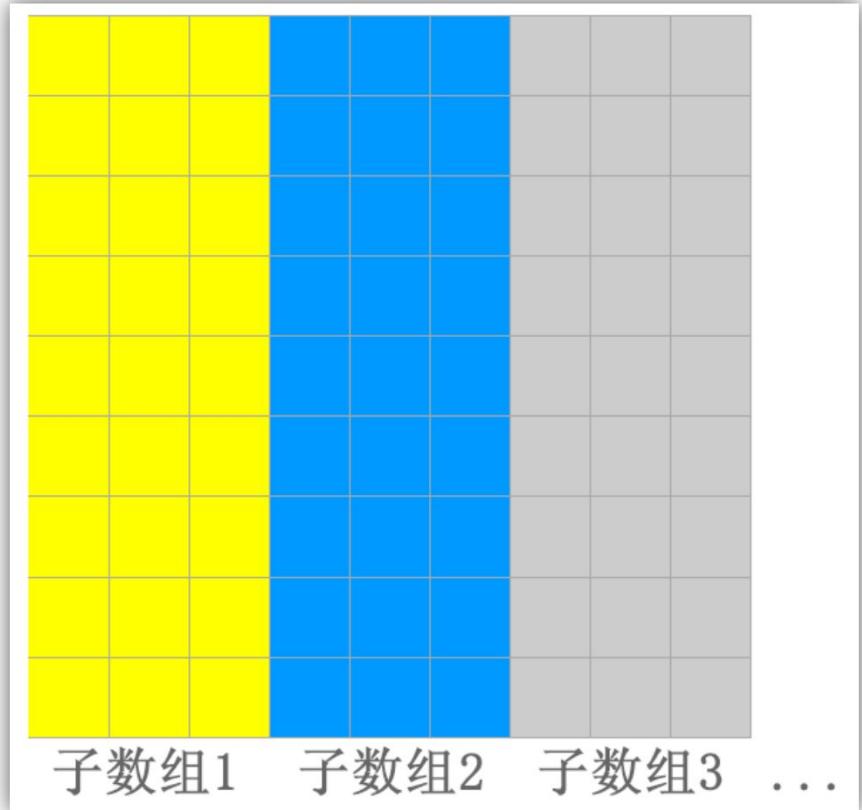
- { 什么样的问题适合并行计算?
 - 斐波那契序列(Fibonacci)的计算?

$$F_k = \begin{cases} 1, & k = 0 \\ 1, & k = 1 \\ F_{k-1} + F_{k-2}, & k \geq 2. \end{cases}$$

并行计算基础

{ 什么样的问题适合并行计算？

- 如果有大量结构一致的数据要处理，且数据可以分解成相同大小的部分，那我们就可以设法使这道处理变成并行



为什么需要MapReduce？

} 计算问题简单，但求解困难

- 待处理数据量巨大（PB级），只有分布在成百上千个节点上并行计算才能在可接受的时间内完成
- 如何进行并行分布式计算？
- 如何分发待处理数据？
- 如何处理分布式计算中的错误？

简单的问题，计算并不简单！

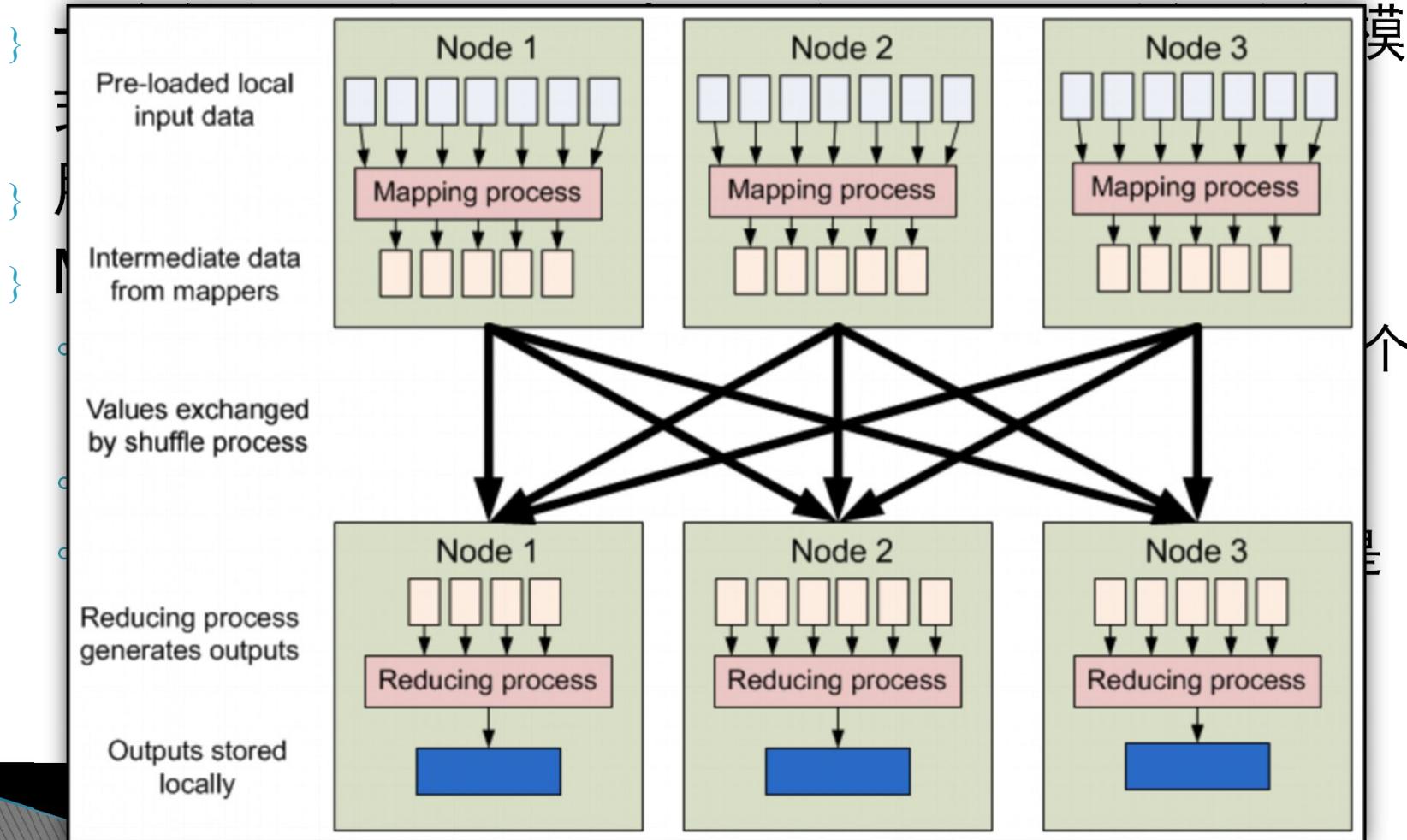
为什么需要MapReduce？

Jeffery Dean设计一个新的抽象模型，使我们只要执行的简单计算，而将并行化、容错、数据分布、负载均衡的等杂乱细节放在一个库里，使并行编程时不必关心它们这就是MapReduce



Google MapReduce
架构设计师
Jeffrey Dean

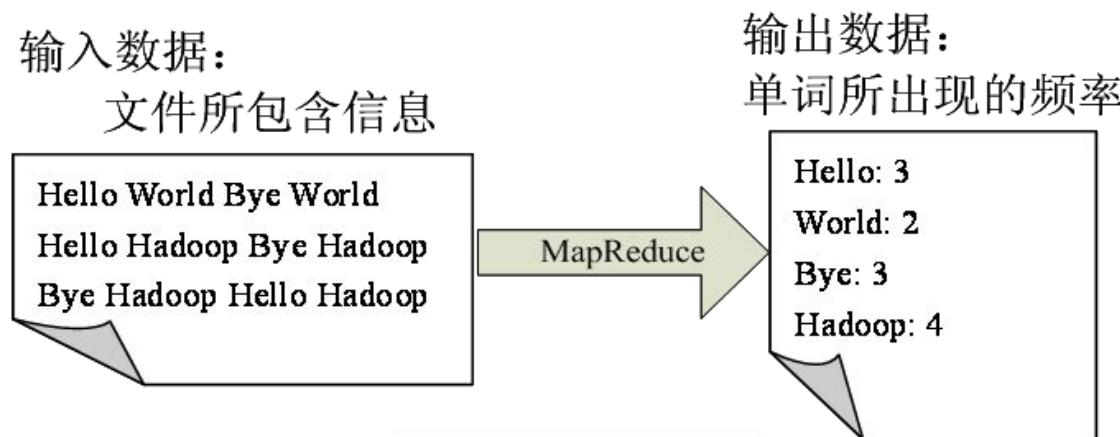
MapReduce



MapReduce示例：单词计数

{ 案例：单词记数问题(Word Count)

- 给定一个巨大的文本（如1TB），如何计算单词出现的数目？



MapReduce示例：单词计数

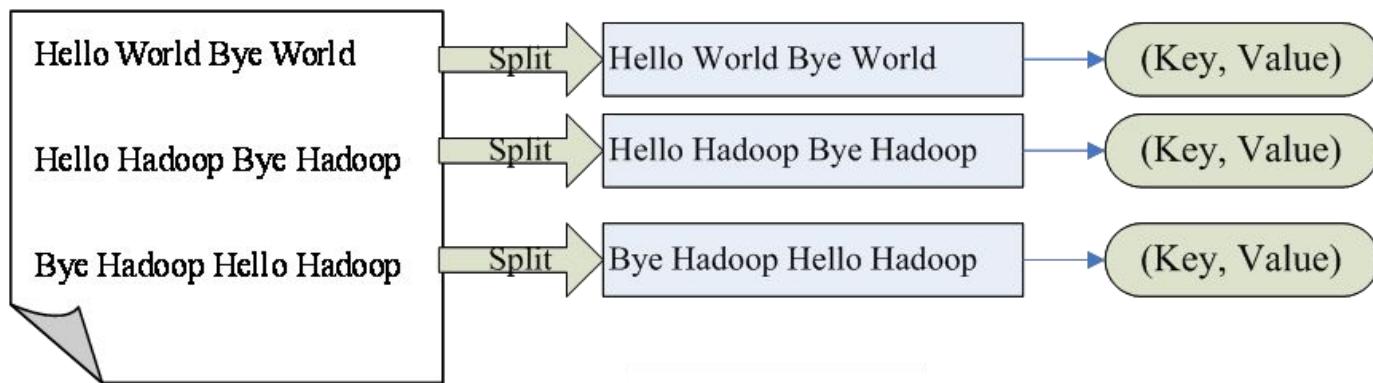
- } 使用MapReduce求解该问题
 - 定义Map和Reduce函数

```
Map (K, V) {  
    For each word w in V  
        Collect(w , 1);  
}  
  
Reduce (K, V[]) {  
    int count = 0;  
    For each v in V  
        count += v;  
    Collect(K , count);  
}
```

MapReduce示例：单词计数

} 使用MapReduce求解该问题

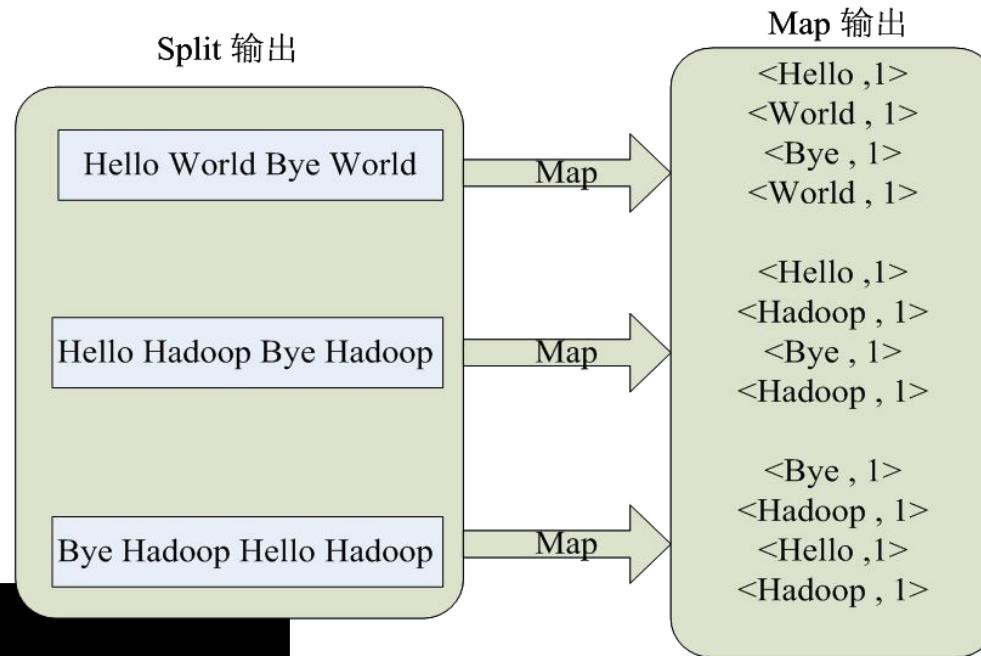
- Step 1: 自动对文本进行分割，形成初始的对



MapReduce示例：单词计数

} 使用MapReduce求解该问题

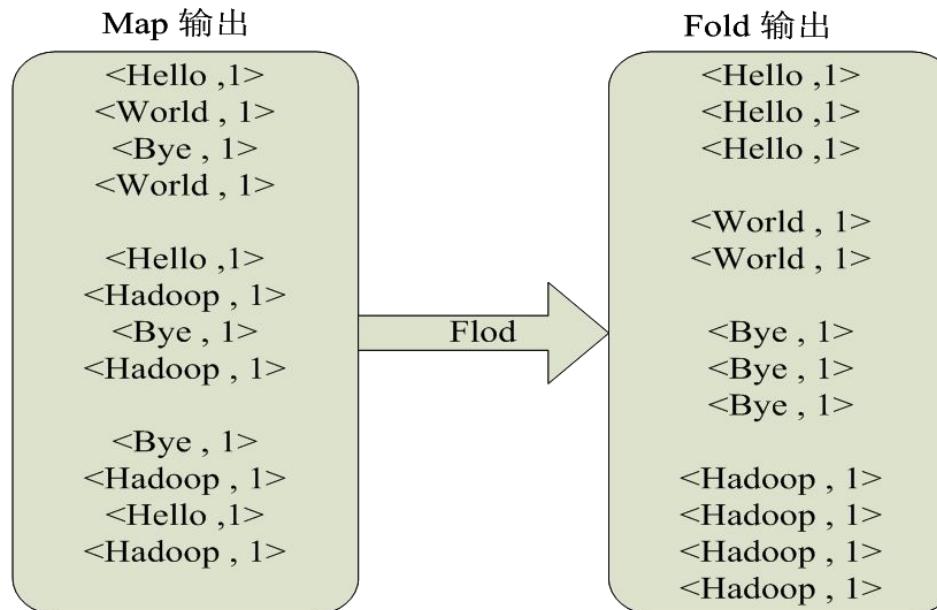
- Step 2: 在分割之后的每一对进行用户定义的Map进行处理，再生成新的对



MapReduce示例：单词计数

} 使用MapReduce求解该问题

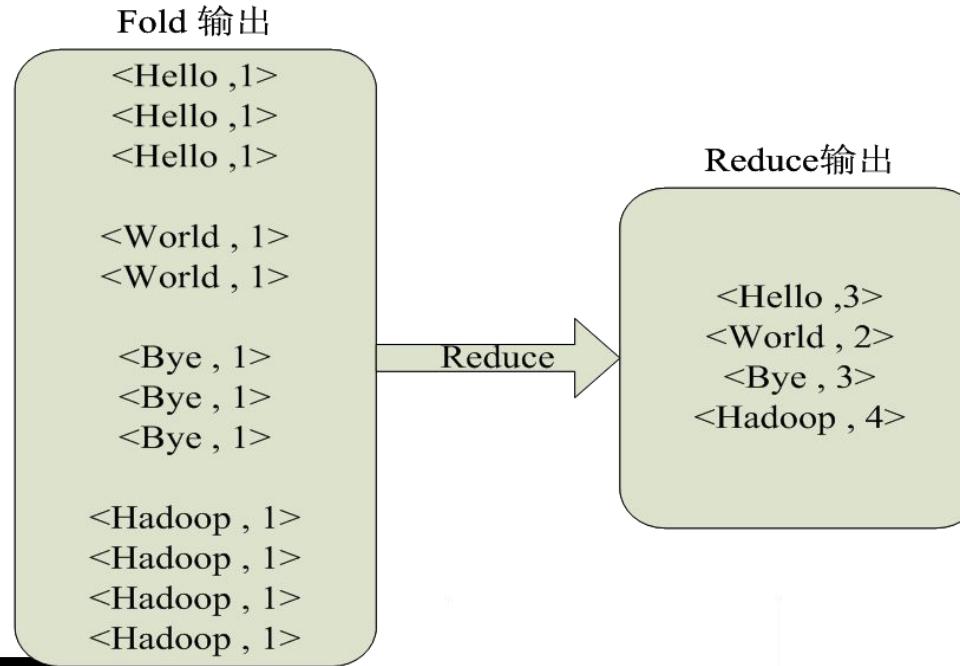
- Step 3: 对输出的结果集归拢、排序(系统自动完成)



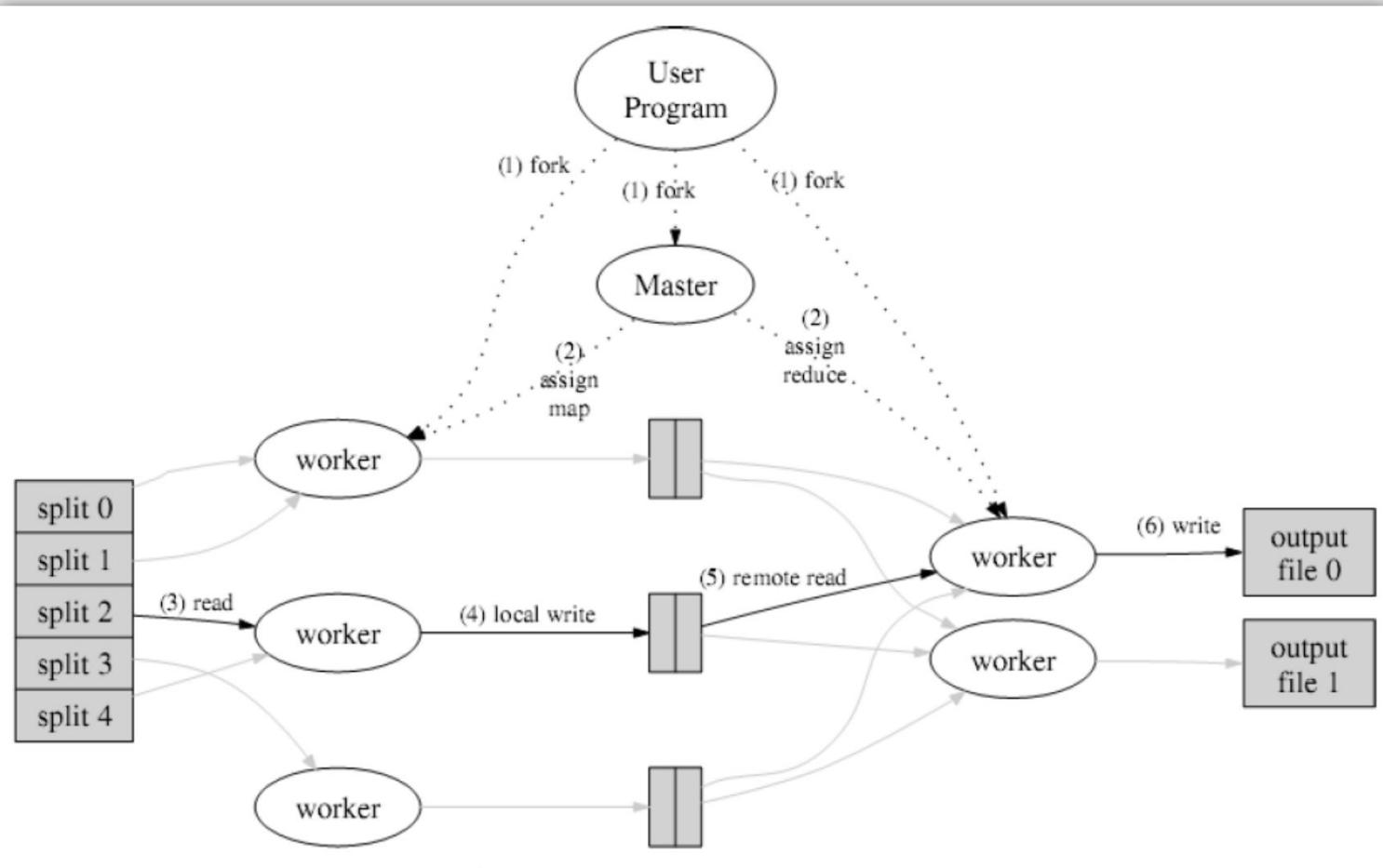
MapReduce示例：单词计数

{ 使用MapReduce求解该问题

- Step 4: 通过Reduce操作生成最后结果



Google MapReduce执行流程



文件存储位置

- { 源文件： GFS
- { Map处理结果： 本地存储
- { Reduce处理结果： GFS
- { 日志： GFS

思考

Google MapReduce计算架构有什么问题？

MapReduce的容错

} Worker故障

- Master 周期性的ping每个worker。如果master在一个确定的时间段内没有收到一个worker返回的信息，那么它将把这个worker当作死机处理。
- 重新执行该节点上已经执行或尚未执行的Map任务
- 重新执行该节点上未完成的Reduce任务，已完成的不再执行

WHY?

} Master故障

- 定期写入检查点数据
- 从检查点恢复

MapReduce的优化

} 任务备份机制

- 慢的workers 会严重地拖延整个执行完成的时间
 - 由于其他的任务占用了资源
 - 磁盘损坏
- 解决方案: 在临近结束的时候, 启动多个进程来执行尚未完成的任务
 - 谁先完成, 就算谁
- 可以十分显著地提高执行效率

MapReduce的优化

{ 本地处理

- Master 调度策略：
 - 向GFS询问获得输入文件blocks副本的位置信息
 - Map tasks 的输入数据通常按 64MB来划分 (GFS block 大小)
 - 按照blocks所在的机器或机器所在机架的范围进行调度
- 效果
 - 绝大部分机器从本地读取文件作为输入，节省大量带宽

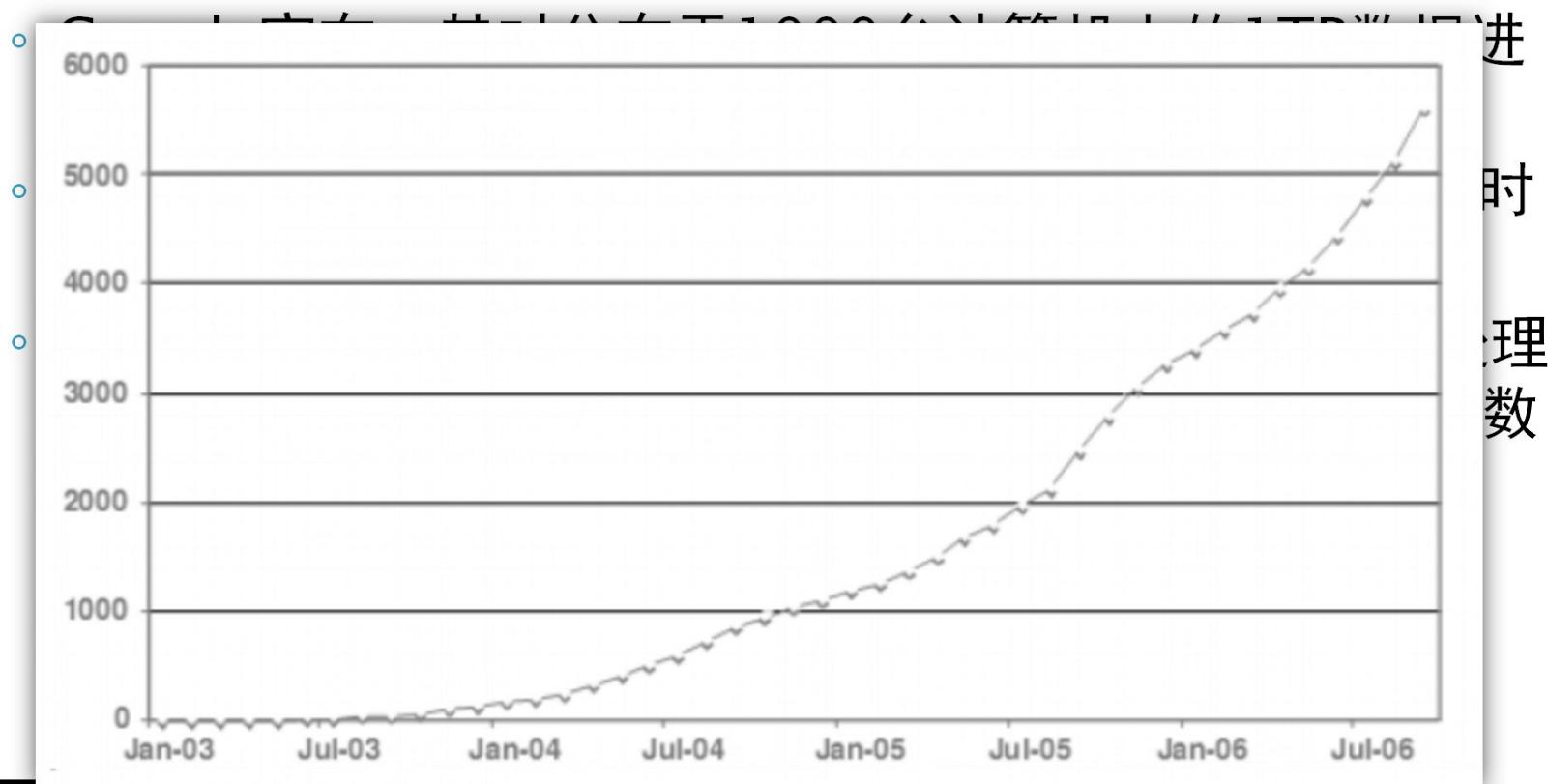
MapReduce的优化

} 跳过有问题的记录

- 一些特定的输入数据常导致Map/Reduce无法运行
- 最好的解决方法是调试或者修改
 ~~不一定可行~~~ 可能需要第三方库或源码
- 在每个worker里运行一个信号处理程序，捕获map或reduce任务崩溃时发出的信号，一旦捕获，就会向master报告，同时报告输入记录的编号信息。如果master看到一条记录有两次崩溃信息，那么就会对该记录进行标记，下次运行的时候，跳过该记录

“实践是检验真理的唯一标准”

} 实践证明，MapReduce是出色的分布式计算模型



Goolge的云计算

» 分布式数据表BigTable

BigTable

{ 为什么需要设计BigTable?

- Google需要存储的数据种类繁多
 - 网页，地图数据，邮件.....
 - 如何使用统一的方式存储各类数据？
- 海量的服务请求
 - 如何快速地从海量信息中寻找需要的数据？

{ BigTable：基于GFS和Chubby的分布式存储系统

- 对数据进行结构化存储和管理
- 与GFS的联系

Google的需求

- { 数据存储可靠性
- { 高速数据检索与读取
- { 存储海量的记录（若干TB）
- { 可以保存记录的多个版本

假设

- { 与写操作相比，数据记录读操作占绝大多数工作负载
- { 单个节点故障损坏是常见的
- { 磁盘是廉价的
- { 可以不提供标准接口
 - Google既能控制数据库设计，又能进行应用系统设计

设计目标

- { 具有广泛的适应性
 - 支持Google系列产品的存储需求
- { 具有很强的可扩展性
 - 根据需要随时加入或撤销服务器
 - 应对不断增多的访问请求
- { 高可用性
 - 单个节点易损，但要确保几乎所有的情况下系统都可用
- { 简单性
 - 简单的底层系统可减少系统出错概率，为上层开发带来便利

逻辑视图

关系数据库中的表是什么样的？有什么特征？

关系数据库中的表设计需要遵循什么原则？

数据模型

} 行

- 每行数据有一个可排序的关键字和任意列项
- 字符串、整数、二进制串甚至可串行化的结构都可以作为行键
- 表按照行键的“逐字节排序”顺序对行进行有序化处理
- 表内数据非常‘稀疏’，不同的行的列的数完全目可以大不相同
- URL是较为常见的行键，存储时需要倒排
 统一地址域的网页连续存储，便于查找、分析和压缩

mp3.baidu.com/index.asp→com.baidu.mp3/index.asp

数据模型

{ 列

- 特定含义的数据的集合，如图片、链接等
- 可将多个列归并为一组，称为族（family）
- 采用 族:限定词 的语法规则进行定义
 fileattr:owning_group”, “fileattr:owning_user”, etc
- 同一个族的数据被压缩在一起保存
- 族是必须的，是BigTable中访问控制的基本单元

数据模型

{ 时间戳

- 保存不同时期的数据，如“网页快照”

{ “A big table”

- 表中通过(row, col, timestamp)查询
- 表中通过(row, col, MOST_RECENT)查询
- 表中的数据行数可以无限地增加

数据模型

} 无数据校验

- 每行都可存储任意数目的列
 ✖ BigTable不对列的最少数目进行约束
- 任意类型的数据均可存储
 ✖ BigTable将所有数据均看作为字符串
- 数据的有效性校验由构建于其上的应用系统完成

} 一致性

- 针对同一行的多个操作可以分组合并
 ✖ 不支持对多行进行修改的操作符

物理视图

Row Key	Time Stamp	Column Contents	Column Anchor		Column "mime"
			cnnsi.com	my.look.ca	
"com.cnn.www"	T9		CNN		
	T8			CNN.COM	
	T6	"<html>.. "			Text/html
	T5	"<html>.. "			
	t3	"<html>.. "			

Row Key	Time Stamp	Column: Contents
Com.cnn.www	T6	"<html>.. "
	T5	"<html>.. "
	T3	"<html>.. "

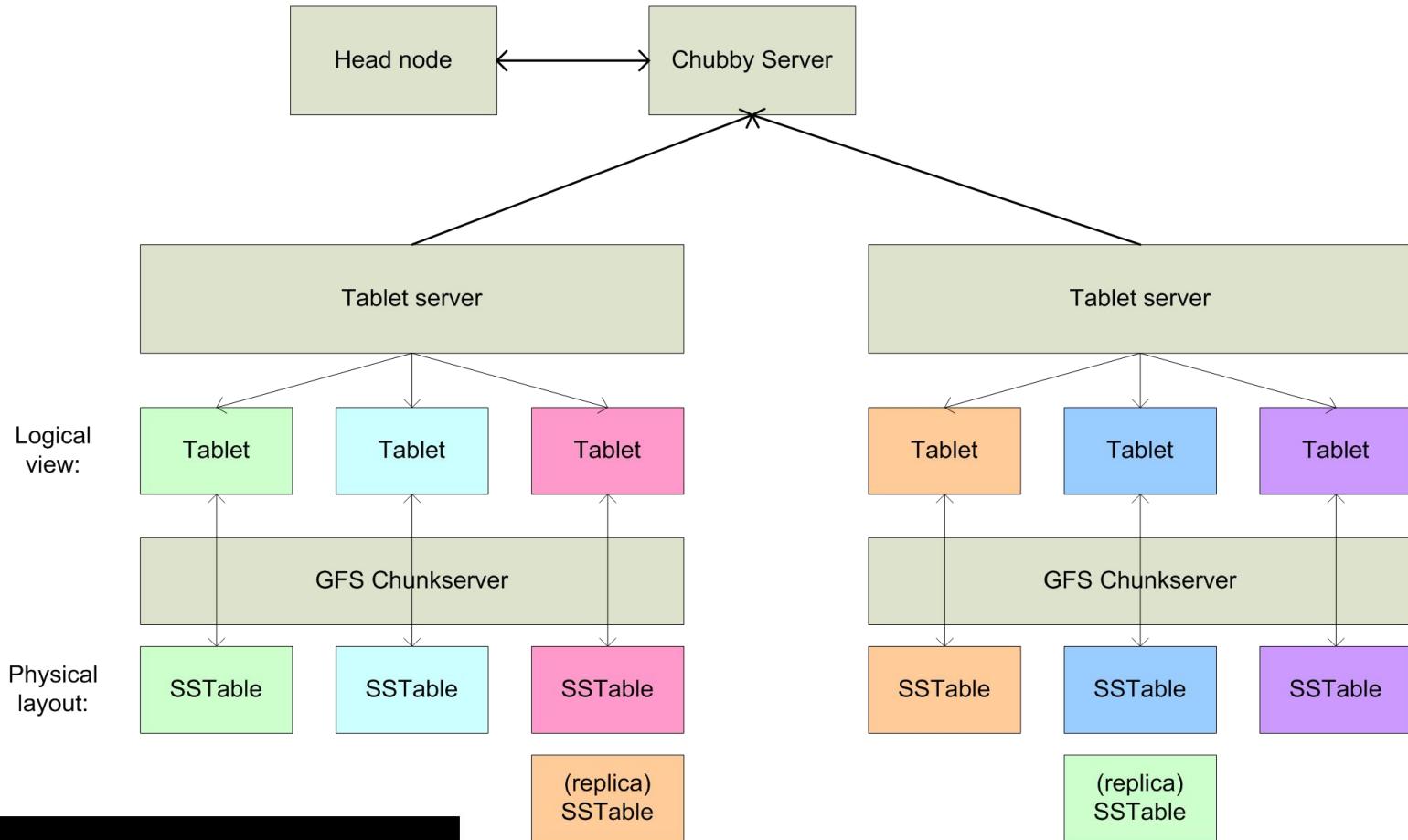
Row Key	Time Stamp	Column: Anchor	
Com.cnn.www	T9	Anchor:cnnsi.com	CNN
	T5	Anchor:my.look.ca	CNN.COM

Row Key	Time Stamp	Column: mime
Com.cnn.www	T6	text/html

物理视图

- { 逻辑上的“表”被划分为若干子表（Tablet）
 - 每个Tablet由多个SSTable文件组成
 - SSTable文件存储在GFS之上
- { 每个子表存储了table的一部分行
 - 元数据：起始行键、终止行键
 - 如果子表体积超过了阈值（如200M），则进行分割

体系结构

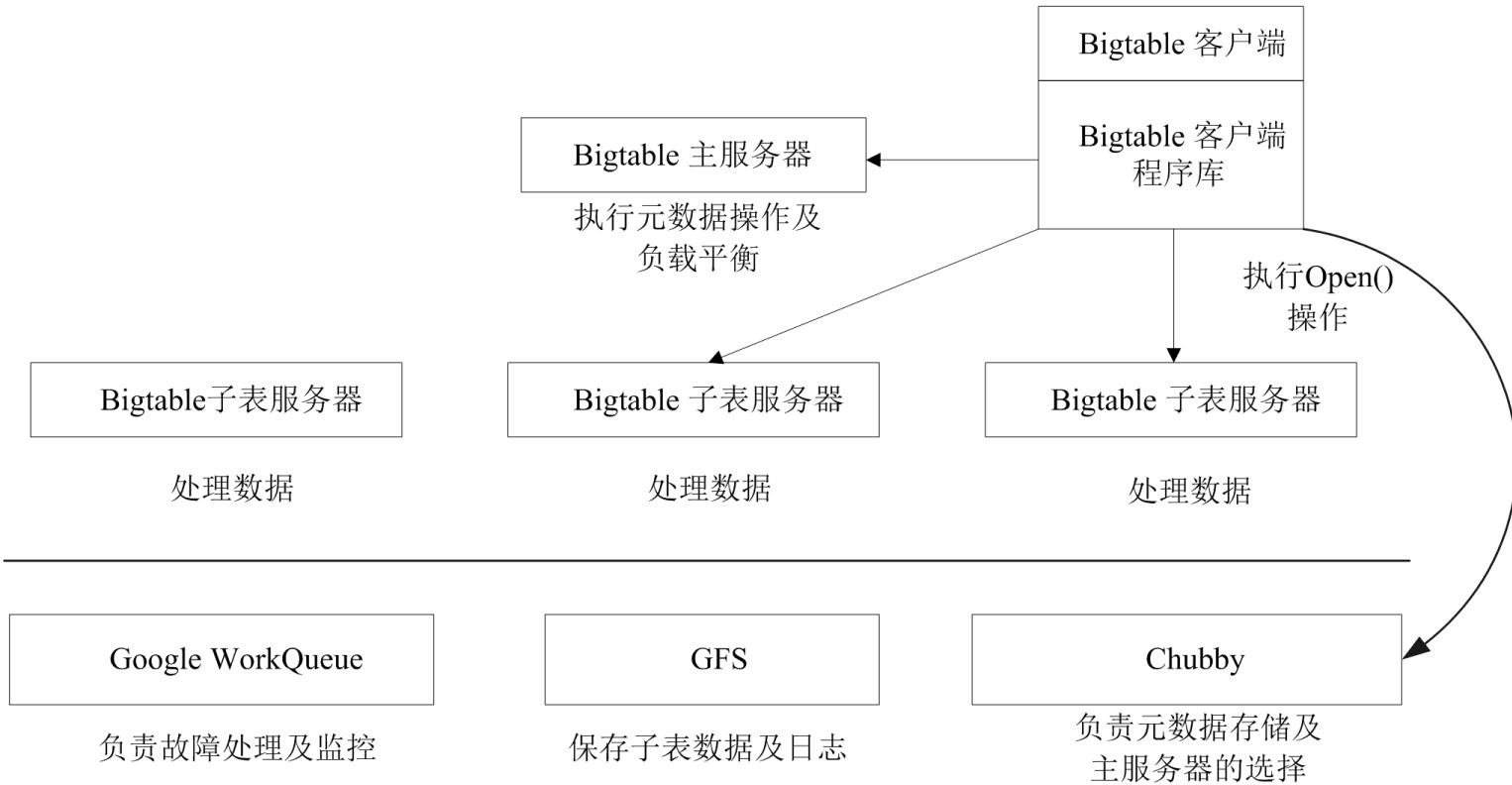


主节点的职责

- { 为每个子表服务器分配子表，对外提供服务
- { 与GFS垃圾回收进行交互，收回废弃的SSTable
- { 探测子表服务器的故障与恢复
- { 负载均衡

有效缓解单点故障

BigTable小结



综合讨论

- { Google云计算架构中GFS、MapReduce和BigTable中是否存在集群节点复用的情况？
- { 如何复用？
- { 节点复用的好处有哪些？
- { Google云计算架构的设计对你有哪些启发？有哪些收获？

中移动大云平台Big Cloud

» BC-PDM分布式数据挖掘系统

“大云”研发大事记

- } 2007年3月，确定了大云(Big Cloud)计划，即中国移动研究院为打造中国移动云计算基础设施而实施的关键技术研究及原型系统开发计划。
- } 2007年7月，利用闲置的15台PC服务器，基于开源软件搭建了海量数据处理试验平台，并成功运行搜索引擎软件。
- } 2008年10月，建立256节点的大规模运算实验室，并运行数据挖掘工具和相关应用。
- } 2009年9月，Big Cloud 0.5版本在中国移动研究院内部发布试用。
- } 2009年12月，试验平台进一步扩容，达到1000台服务器、5000个CPU、3000TB的存储规模。

大云数据挖掘系统(BC-PDM)

} 并行数据挖掘工具(BC-PDM)是一套高性能、低成本、高可靠性、高可伸缩性的海量数据处理、分析和挖掘系统。该工具提供海量数据并行ETL和并行挖掘能力，支持企业的BI应用和精准营销;提供业务逻辑复杂的SQL 能力，支持海量数据的清洗、转换、关联、汇总等操作，支持生成企业报表、KPI、挖掘等应用;提供基于Web的SaaS服务模式，降低企业IT系统投资。



中国移动通信
CHINA MOBILE

并行数据挖掘
BC-PDM的web化服务工具

用户名:

密 码:

登录



登录后界面

中国移动通信
CHINA MOBILE

并行数据挖掘
BC-PDM web服务工具

欢迎 admin 登录 退出
当前项目组-- itemA
切换项目组 itemA

操作工作流
数据压缩与传输
已有流程调度
用户权限管理

用户流程管理和布局 工作区 |

数据加载与导出 ETL操作 挖掘算法 挖掘连接组件 结果展示 用户

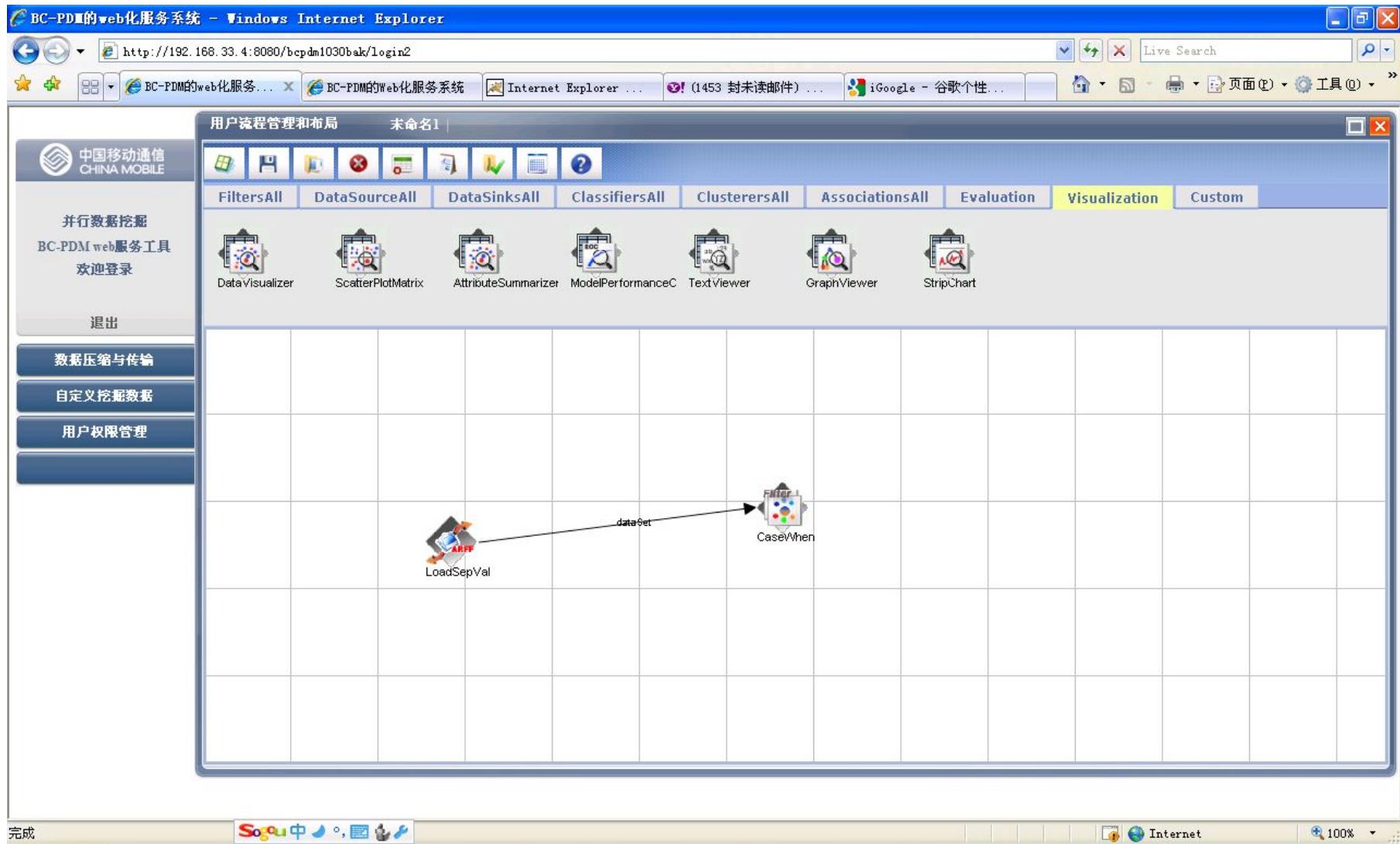
导入 导出 普通ETL 链式ETL 分类 聚类 关联规则 训练测试集 结果显示 用户

清洗类 转换类

数据类型检查 外键约束 主键约束 缺值处理 空值域约束 去重 CaseWhen组件 计数区间化 字段类型

The screenshot shows the BC-PDM web service tool's user interface. The top navigation bar includes tabs for '工作区' (Work Area), '数据加载与导出' (Data Import/Export), 'ETL操作' (ETL Operations), '挖掘算法' (Mining Algorithms), '挖掘连接组件' (Mining Connection Components), '结果展示' (Result Display), and '用户' (User). Below this is a sub-navigation bar with tabs for '导入' (Import), '导出' (Export), '普通ETL' (Normal ETL), '链式ETL' (Chain ETL), '分类' (Classification), '聚类' (Clustering), '关联规则' (Association Rules), '训练测试集' (Training/Test Set), '结果显示' (Result Display), and '用户' (User). A toolbar above the tabs contains icons for various functions. The main area is divided into sections for '清洗类' (Cleaning) and '转换类' (Transformation), each containing several components represented by icons with 'Filter' labels. The bottom half of the screen is a large, empty grid for defining data flows or processes.

工作流画布 (Knowledge Flow Layout)



job监控框(Job monitor)

The screenshot shows the Hadoop Job Monitor interface. At the top, there is a flow diagram with two components: '导入不定长' (Import Unbounded Length) and 'join组件' (join component). An arrow labeled 'dataSet' points from the first component to the second. Below the diagram, a status message reads: 'join组件 正在运行...' (join component is running...).

Below the flow diagram, a blue header bar displays job information: `jobid=job_201003031725_0231 numnodes=49 jobname=chain_Join totalInputsize=1.27248097E10`.

To the left, a sidebar provides configuration details:

```
reduce =20  
2010-4-  
15  
10:10:1  
6:  
map=9  
5  
reduce  
=21
```

On the right, there are two main sections: a bar chart and a table.

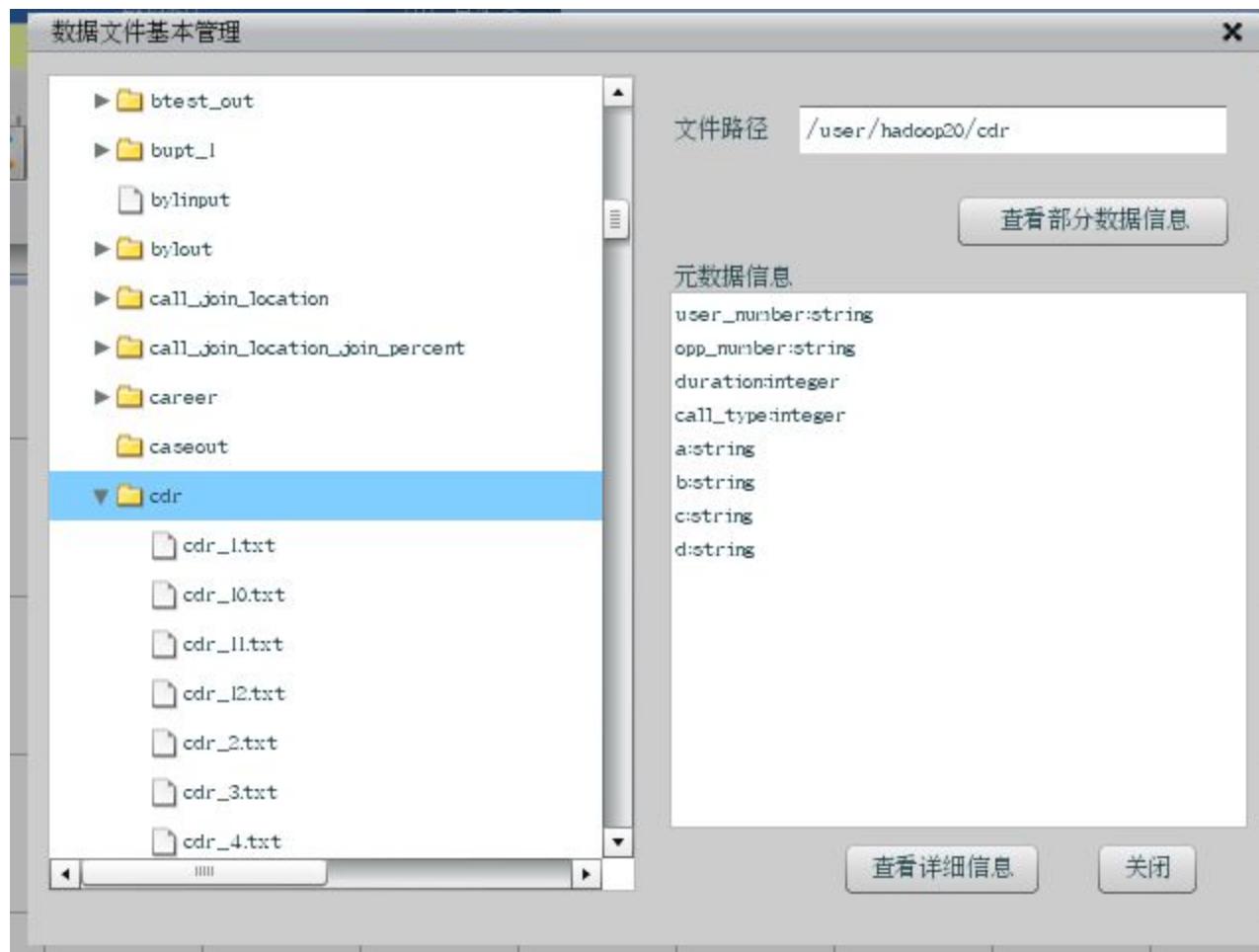
Bar Chart: The chart compares 'map' and 'reduce' tasks. The 'map' task has a value of approximately 95, and the 'reduce' task has a value of approximately 20.

组件	值
map	95
reduce	20

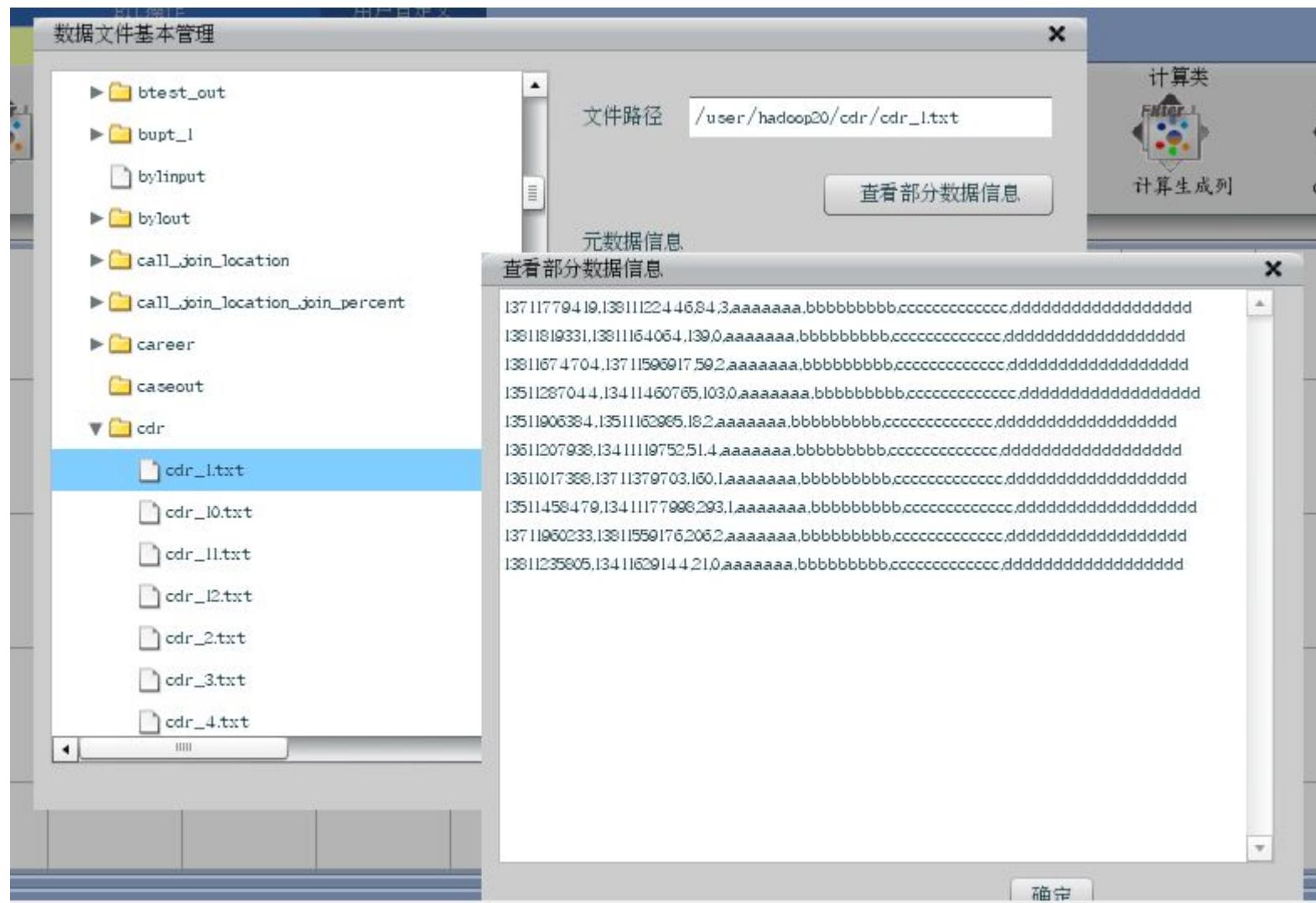
Table: A table with columns for job metrics. The first column is highlighted in yellow and contains the text '导入不定长' (Import Unbounded Length), which corresponds to the 'join组件' component shown in the flow diagram above.

组件开始时间	组件结束时间	组件持续时间	job名稱	job提交时间	job开始时间	job结束时间	job持续时间	job状态	jobID
			导入不定长						

数据查看与管理



预览数据



数据文件基本管理
interval

输入文件 /user/hadoop20/dm_call.txt
区间数 10

元数据信息
基本信息
字段

文件： /user/hadoop20/dm_call.txt
字段数量： 0

字段	Max	Min
total_fee	1140303.05	0.0
basic_fee	2164.91	-83194.51
p2p_sms_fee	11657.05	-1.66
data_fee	1140303.05	-0.66
value_added_fee	2000	10

sub_id:string
total_fee:float
basic_fee:float

p2p_sms_fee:float
data_fee:float
value_added_fee:float

call_cnt:int
call_cell:int
out_call_cnt:int

out_call_cell:int
in_call_cnt:int
In_call_cell:int

local_call_cnt:int

字段名称
sub_id
total_fee

basic_fee
p2p_sms_fee
data_fee

value_added_fee

统计信息
合计：

均值： 73.85978268253263

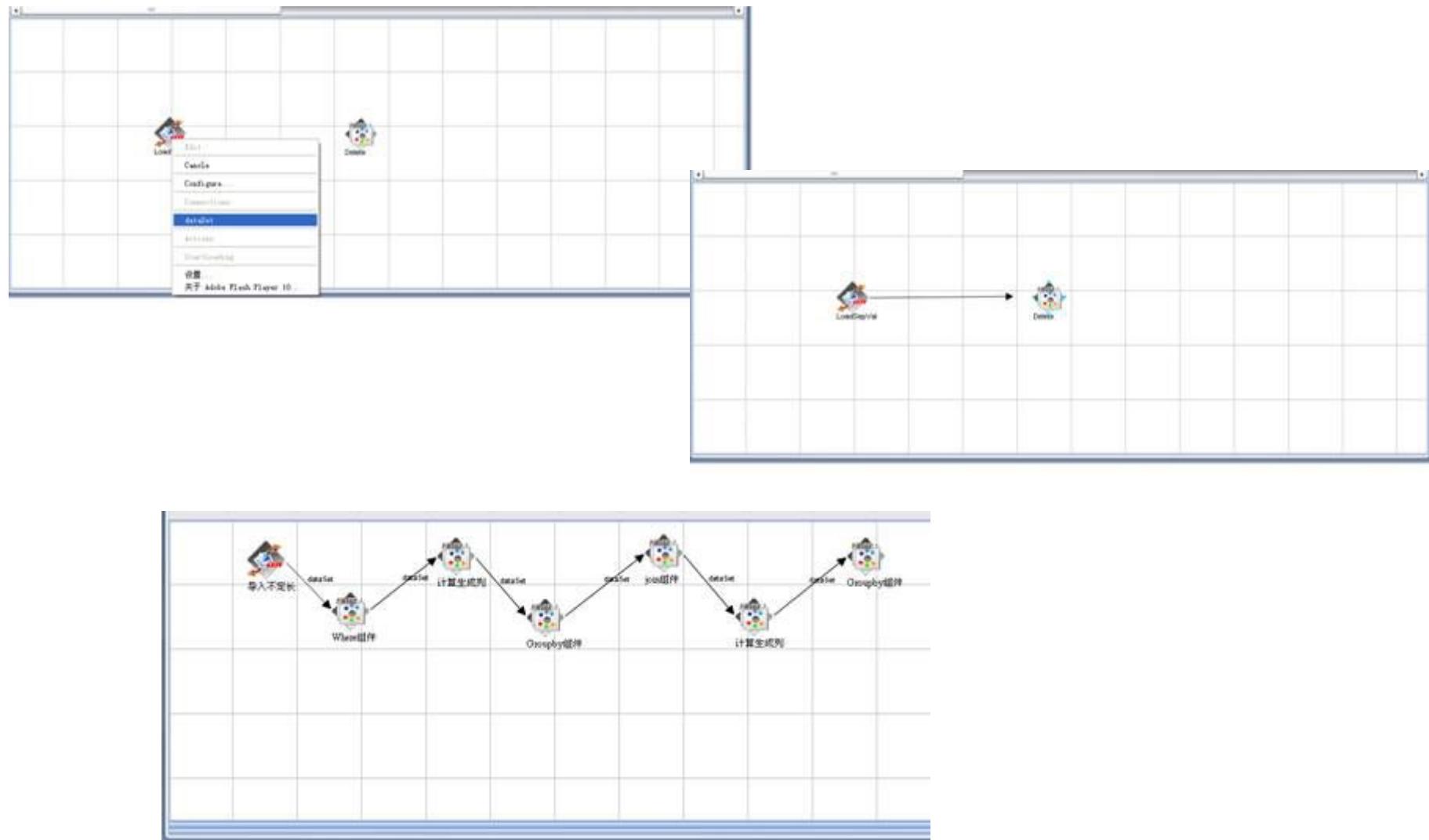
最大值： 1140303.05

最小值： 0.0

方差： 73.85978268253263

确定

创建工作流





数据加载与导出 ETL操作 挖掘算法 挖掘连接组件 结果展示 用户
导入 导出 普通ETL 链式ETL 分类 聚类 关联规则 训练测试集 结果显示 用户



导入定长



导入不定长



打开文件列表

文件列表属性：

文件名称	文件大小
gd_yijing_gprs4.xml	9181
wwwwww.xml	4508
limm_excute_dm_call_ir	8109
limm_excute_dm_call_	8164
lr_0120_create.xml	5922
eeee.xml	10385
jzx_dm_sphere_temp11	6763
jzx_dm_lac_ci_200910.	6921

删除

全选

全不选

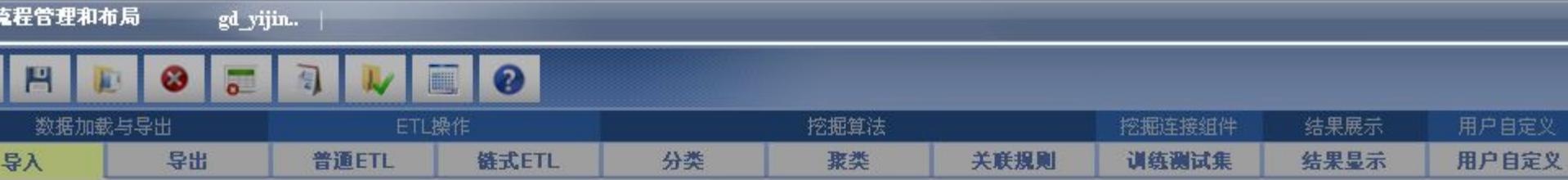
确认

配置工作流

用户流程管理和布局

gd_yijin..





where

输入文件： 浏览...

输出文件： 浏览...

表达式：

编辑

numOfmap numOfreduce

确定

取消

运行工作流

用户流程管理和布局 gd_hivet.. |

工具

数据加载与导出 ETL操作 挖掘算法 挖掘连接组件 结果展示 用户自定义

导入 导出 普通ETL 链式ETL 分类 聚类 关联规则 训练测试集 结果显示 用户自定义

导入定长 导入不定长

正在运行...

jobid=job_201003021439_0184 numnodes=25 jobname=chain_group by totalInputsizes=1.24225505E10

组件开始时间	组件结束时间	组件持续时间	job名称	job提交时间	job开始时间	job结束时间	job持续时间	job状态	jobId
2010-4-13 9:00:00-00:00	0d 0h 1m	chain_g	2010-4-13 9:00:00	2010-4-13 9:00:00	2010-4-13 9:00:00	0d 0h 1m	succ	job_201003021439_0184	

组件异常信息

查看工作流结果

CMRI-Big Cloud-PDM Tools

DataSources DataSinks Filters Classifiers4MR Clusterers4MR Associations4MR Evaluation Visualization

unsupervised

Knowledge Flow Layout

```
graph LR; dm_ca11[dm_ca11] -- "dataSet" --> PNormalize[PNormalize]; PNormalize -- "dataSet" --> Train[Training Set Maker]; Train -- "trainings" --> PKmeans[PKmeans]; PKmeans -- "text" --> GraphViewer[GraphViewer];
```

Job monitor ClusterSize: 30 nodes & InputDataSize: 7.4GB

```
16:06:49: map 100% reduce 99%
16:06:50: map 100% reduce 100%
16:06:51: Job complete: job_200906091821_0111
```

Log

```
16:03:13: Task Id : attempt_200906091821_0111_m_000018_0, Status : FAILED
16:03:13: Task Id : attempt_200906091821_0111_m_000038_0, Status : FAILED
16:03:19: Task Id : attempt_200906091821_0111_m_000038_1, Status : FAIL FD
16:06:51: Job complete: job_200906091821_0111
```

Status

OK

map:100% reduce:100%

结果展示

2009-11-24 10

```
Build Classifier:  
Index(0-first):0,1,3,4,5,6,7,8,9  
Target Index(0-first): 8  
Initial g's v=10615.775364196445  
Initial weight vector:  
-0.5740493607788926  
-0.08045057117522388  
-6.115925247660133E-4  
-0.01783267712607373  
6.041072202240968E-4  
0.023380936166284595  
8.81201105942224E-4  
-0.0031770112006717505  
0.030217379481376865  
0.030523803428856747  
  
Begin Iteration: 1g's v=929.306733443305  
s's v=0.012509965561592775  
w Vector:  
-0.5739969732964518 -0.11404020583503545
```

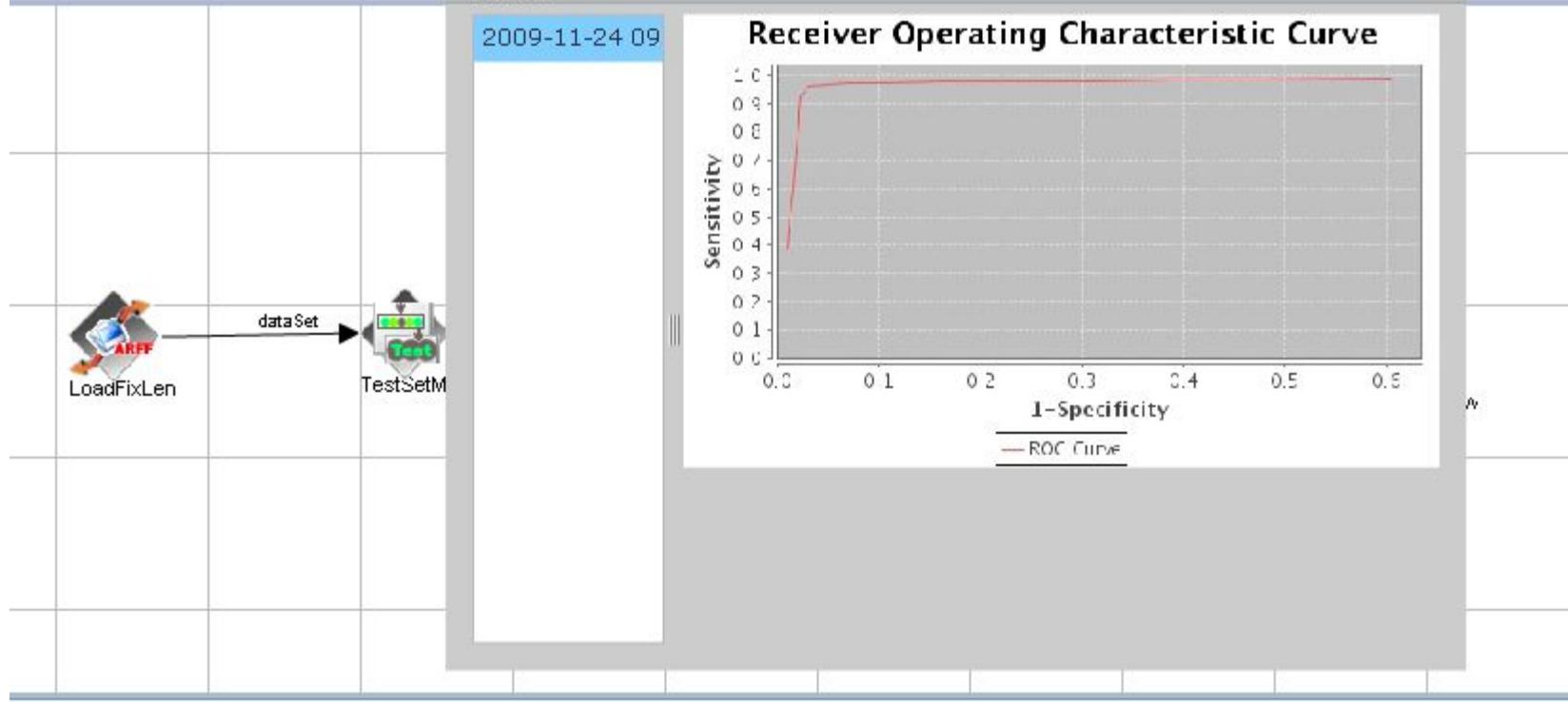
文本

EmptyAssociator4M PAwtits

PFPgrowth

结果展示

2009-11-24 09



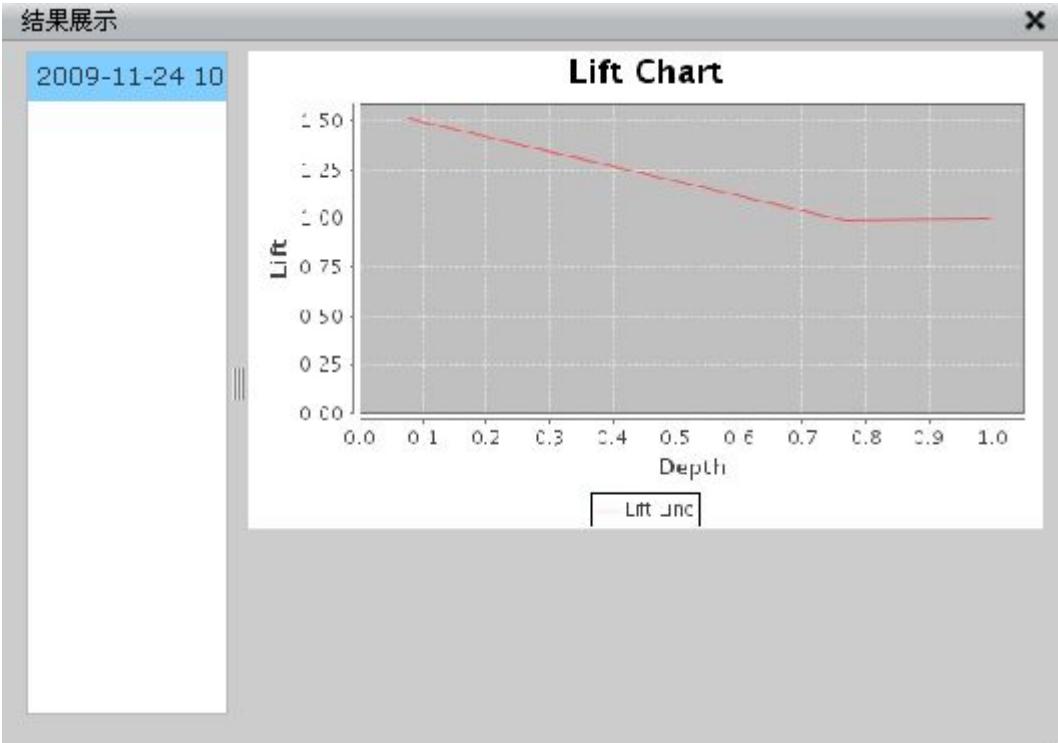
1171523_0910 jobname=TestJob totalInputsize=1680156.0

educe=0

educe=0



ROC图



Lift图

工作流调度情况

单流程调度模式

<切换到流程批处理模式>

已有模型	监听状态	监听次数	操作
select1.xml	未被调度	共调度0次(等待运行0；正在运行0；结束运行0)	自定义 /详细调度
session-3.xml	未被调度	共调度0次(等待运行0；正在运行0；结束运行0)	自定义 /详细调度
delete3.xml	未被调度	共调度0次(等待运行0；正在运行0；结束运行0)	自定义 /详细调度
delete.xml	未被调度	共调度0次(等待运行0；正在运行0；结束运行0)	自定义 /详细调度
seq_2.xml	未被调度	共调度0次(等待运行0；正在运行0；结束运行0)	自定义 /详细调度
gt.xml	未被调度	共调度0次(等待运行0；正在运行0；结束运行0)	自定义 /详细调度
delete2.xml	未被调度	共调度0次(等待运行0；正在运行0；结束运行0)	自定义 /详细调度
delete1.xml	未被调度	共调度0次(等待运行0；正在运行0；结束运行0)	自定义 /详细调度
session-2.xml	未被调度	共调度0次(等待运行0；正在运行0；结束运行0)	自定义 /详细调度
seq_1.xml	未被调度	共调度0次(等待运行0；正在运行0；结束运行0)	自定义 /详细调度
session-1.xml	未被调度	共调度0次(等待运行0；正在运行0；结束运行0)	自定义 /详细调度
delete4.xml	未被调度	共调度0次(等待运行0；正在运行0；结束运行0)	自定义 /详细调度
lr2.xml	未被调度	共调度0次(等待运行0；正在运行0；结束运行0)	自定义 /详细调度
seq_3.xml	未被调度	共调度0次(等待运行0；正在运行0；结束运行0)	自定义 /详细调度

BC-PDM的数据装载和导出

数据加载是将分布式文件系统（DFS）上的没有元数据文件的数据生成元数据文件，或对数据进行断行、空行等初步处理，或作为工作流运行的起始组件。

数据导出是将处理后的数据文件转换成指定的格式或分隔符，数据仍然保存在DFS上。

数据加载（导入）

加载不定长格式数据

源路径

Digitized by srujanika@gmail.com

浏览

目标路径

[View Details](#) | [Edit](#) | [Delete](#)

浏览

○ 输入元数据信息

字段	字段类型

增加、刪除、修改

● 加载元数据文件

浏览

分隔符

异常数据处理

另存路径

浏览...

■ 是否忽略第一行

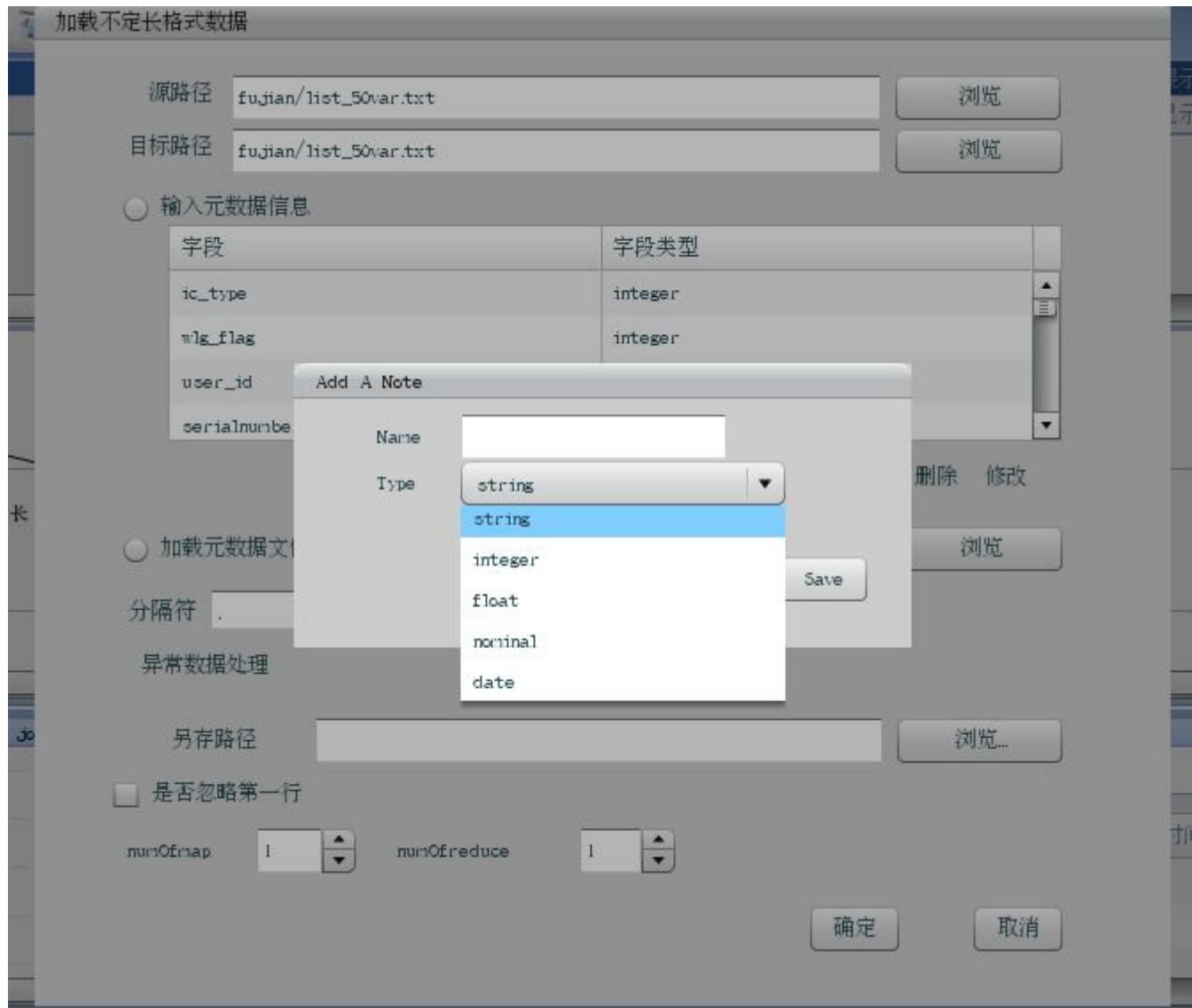
numOfmap 1

numOfmap 1 numOfreduce 1

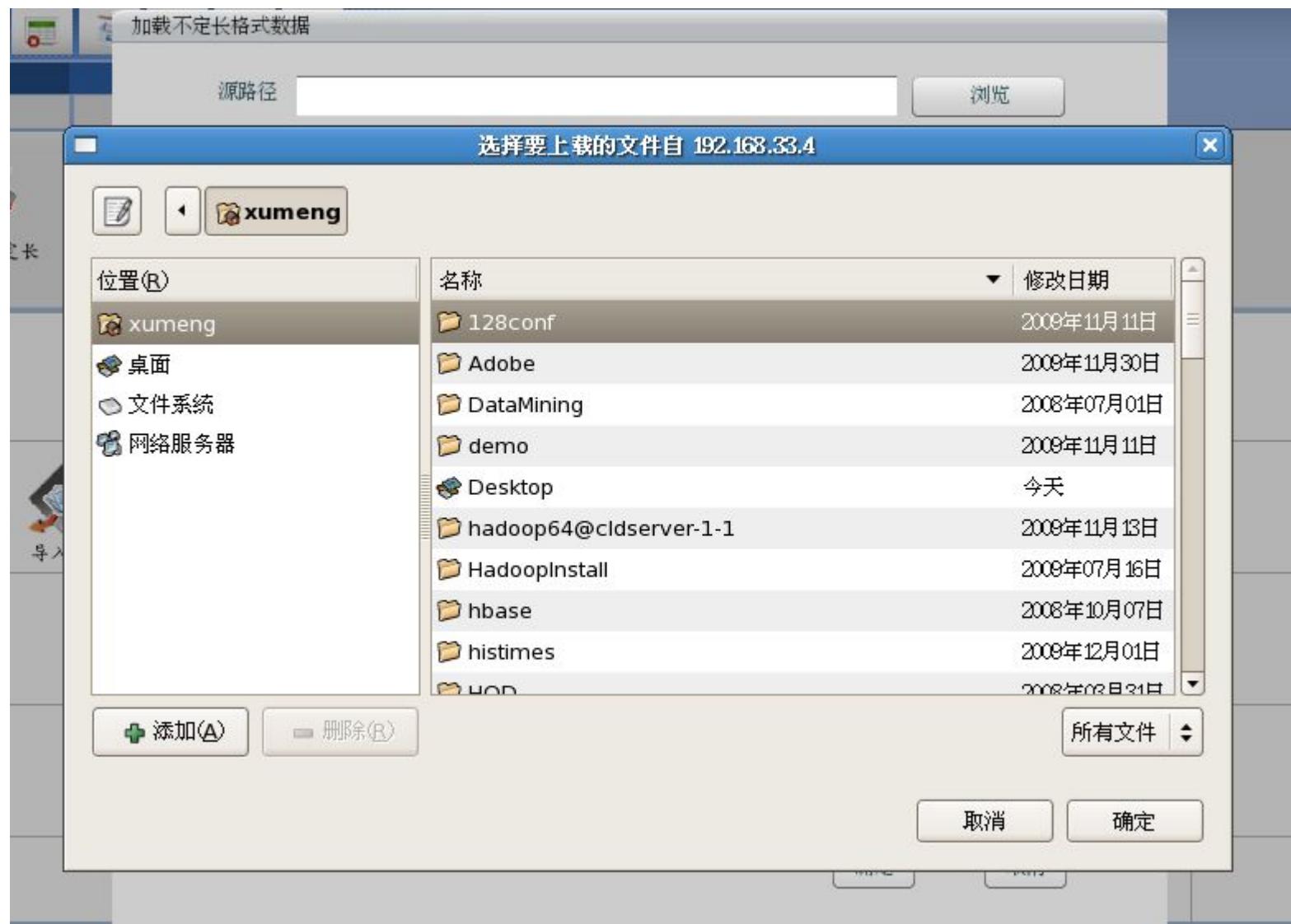
确定

取消

数据加载（导入）



导入头文件



头文件举例

start_time,date,开始时间
imsi,VARCHAR(10),IMSI
calling,VARCHAR(10),用户号码
user_ip,VARCHAR(10),用户IP地址
APN,VARCHAR(10),访问方式
IMEI,VARCHAR(10),终端标识号
rat,int,2G/3G网络标识
App_type,int,应用类型
LAC,VARCHAR(10),xm
Cell_ID,VARCHAR(10),xm
source_ip,VARCHAR(10),源IP地址
dest_ip,VARCHAR(10),目的地址

数据导出

导出定长格式数据 ×

源路径 浏览...

目标路径 浏览...

输入字段长度

字段名	字段长度

使用配置文件 浏览...

numOfmap numOfreduce

并行数据处理(ETL)

ETL操作可以分为普通ETL和链式ETL。

普通ETL和链式ETL分别又细分为：清洗类、转换类、集成类、计算类、抽样类、集合类、更新类、及其它类8大类。



类别	组件名	功能
清洗类	数据类型检查	对输入文件逐条记录地检查每个字段的数据是否与元数据中的类型相符合
	外键约束	升级主键表或升级外键表后的外键约束检查
	主键约束	对数据表的主键约束检查，包括主键非空和主键唯一
	缺值处理	按照指定的替换值填补数据文件中的缺值或Null值等
	空值域约束	包括非空值约束、值域范围检查和自定义约束检查
	去重	将完全重复的数据行丢弃
转换类	Casewhen	将符合条件的数据按指定进行转换，类似SQL的case when
	计数区间化	按计数将指定字段值区间化为N个区间，每个区间数据个数相等，并为该字段按不同区间设置特定值
	字段类型转换	支持对多个字段进行字段名或字段类型的修改，提供多种数据类型字段之间的强制转换
	数值区间化	按数值将指定字段值区间化为N个区间，每个区间数据取值范围相等，并为该字段按不同区间设置特定值
	归一化	对指定字段按该字段的均值和标准偏差，进行zscore归一化
	属性交换	将属性的两列互换
	关联规则数据生成	将业务订购情况数据生成购物篮数据供关联规则算法使用
	PCA主成分分析	将输入数据的属性由高维降到较低的维度
集成类	Delete组件	删除符合一定表达式条件的记录
	Join组件	可将多个表按指定的字段关联，包括主键join、维表join和普通join三个组件，针对不同关联情况使用不同组件
	Sort组件	按用户指定排序关键字字段进行排序
	Where组件	找出满足用户定义的表达式条件的记录。
计算类	计算生成列	通过对现有多字段混合计算生成的新字段
	Groupby组件	对数据按照用户指定的属性聚集、汇总
	统计	计算每个字段的统计信息
抽样类	分层抽样	供分类目标字段数据平衡使用
	采样	按比例随机抽样数据
集合类	集合差	根据用户指定的数据文件和集合运算表达式进行2个集合的差运算
	集合交并	根据用户指定的数据文件和集合运算表达式进行多个集合间的交，并运算。
更新类	Update组件	更新，类似数据库Update
	Insertupdate组件	增量更新，类似数据库Insertupdate
输出类	数据分发组件	根据其包含的数据分发方式将数据分为两个阶段

清洗类-数据类型检查

根据元数据中各个字段的数据类型，对输入文件逐条记录地检查每个字段的数据是否与元数据中的类型相符合，支持多种日期类型。对发现不满足数据类型的记录，应用异常数据处理规则。并在有效性验证后提供一个验证报告，包括丢弃了多少数据、对数据进行了什么处理等信息。



参数	说明
输入文件	设置输入文件的地址及文件名
输出文件	设置输出文件的地址及文件名
另存路径	设置异常数据文件的地址和文件名
numOfmap	设置Map个数
numOfreduce	设置Reduce个数

转换类-**caseWhen**

根据用户输入的条件，将指定字段的值进行转换，类似SQL的**case when**。用户指定转换的字段与转换规则。支持对多字段进行转换，支持对某个字段多个转换规则，支持**default**规则。可以配置转换生成列的元数据信息。



参数	说明
输入文件	设置输入文件的地址及文件名
输出文件	设置输出文件的地址及文件名
生成字段名	设置要生成字段的名字
字段类型	设置要生成字段的类型
条件表达式	当条件表达式为真时，执行替换表达式
替换表达式	设置替换表达式
numOfmap	设置Map个数
numOfreduce	设置Reduce个数

BC-PDM挖掘算法

1 并行分类算法Classifiers4MR

分类功能应以用户提供的历史消费清单作为训练数据，这些数据中有一个属性作为分

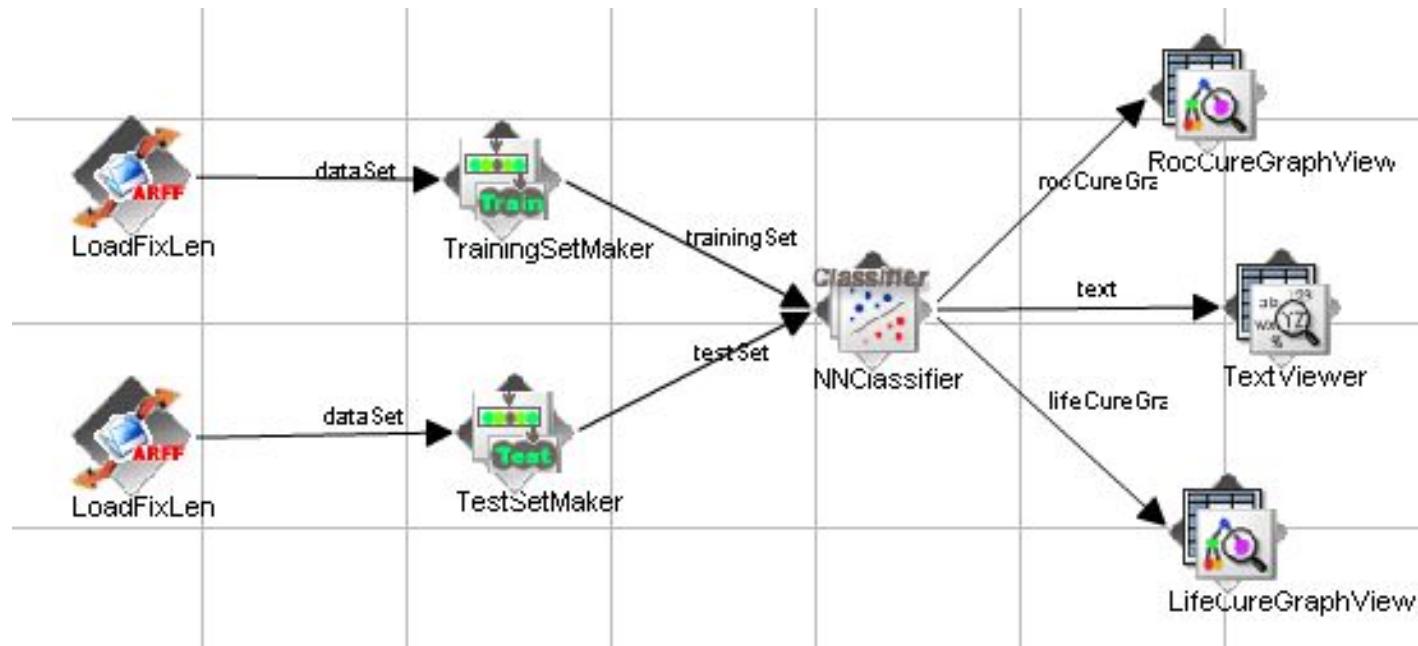
类属性，且每条记录已标明分类属性的值。分类算法应提供两方面基本功能：

(1) 学习分类模型：从训练集数据中发现潜在的分类模型，并以特定方式表达

(2) 预测：对用户新提供的数据集，依据分类模型预测出所属类别
并行分类算法包括：EmptyClassifier4M、C45决策树、CBC分类、
CBR分类、K近邻、朴素贝叶斯、层次C45决策树、线性回归分类、
神经网络算法。

神经网络算法

把整个神经网络的神经元划分成不同层次，对同层次内的不同神经元进行并行参数计算，并通过统一调度和精度控制对神经元进行快速的并行化训练。训练完毕后，对于每一个输入，通过并行化神经网络快速地得到输出



NNClassifier

traininputpath	pku/nn/train.txt	浏览
testinputpath	pku/nn/train.txt	浏览
predictinputpath	pku/nn/train.txt	浏览
outputpath	pku/nn/output	浏览
nummaptasks	2	
numreducetasks	1	
learningrate	0.6	
momentum	0.9	
middleenum	1	
middlepopulation	5	选择
min_success_ratio	0.7	
index		选择
targetindex		选择
modelpath	./model/nnInfo.txt	选择

确定

参数	设置
trainInputPath	设置训练集在DFS上的路径
testInputPath	设置测试集在DFS上的路径
predictInputPath	设置预测测试集在DFS上的路径
outputPath	设置结果输出在DFS的路径
numMapTasks	设置Map的个数，一般取计算集群核个数的4倍
numReduceTasks	设置Reduce的个数，一般取计算集群核个数的2倍
learningRate	设置神经网络的学习率， 默认为0.6
moment	设置神经网络的学习冲量， 默认为0.9
middleNum	设置神经网络的中间层数目， 默认为5
middlePopulation	设置神经网络的各个中间层上节点的数目，每一层的节点数用,隔开。比如有两个中间层，各有x个和y个节，则参数配置为x,y
min_success_ratio	期望达到的最小成功率， 默认为0.7
index	预测利用的属性， 默认为出来目标属性外所有的整形和浮点型的属性
TargetIndex	要预测的目标属性， 默认为最后一维
modelPath	训练中用来存储模型的路径，或者测试时要利用的模型文件的路径

2 并行聚类算法Clusterers4MR

聚类算法应对用户提供的全体数据集，按照一定的聚类原则，自动聚成几簇。每个簇内的数据应具有很高的相似性。应提供的功能包括：

- (1) 自动聚类：将数据集形成簇模型；
 - (2) 预测：依据形成的簇模型，对新数据判定所属的簇。
- 并行聚类规则算法包括：k均值算法、Clara聚类算法、DBScan聚类算法。



3 并行关联规则算法Associations4MR

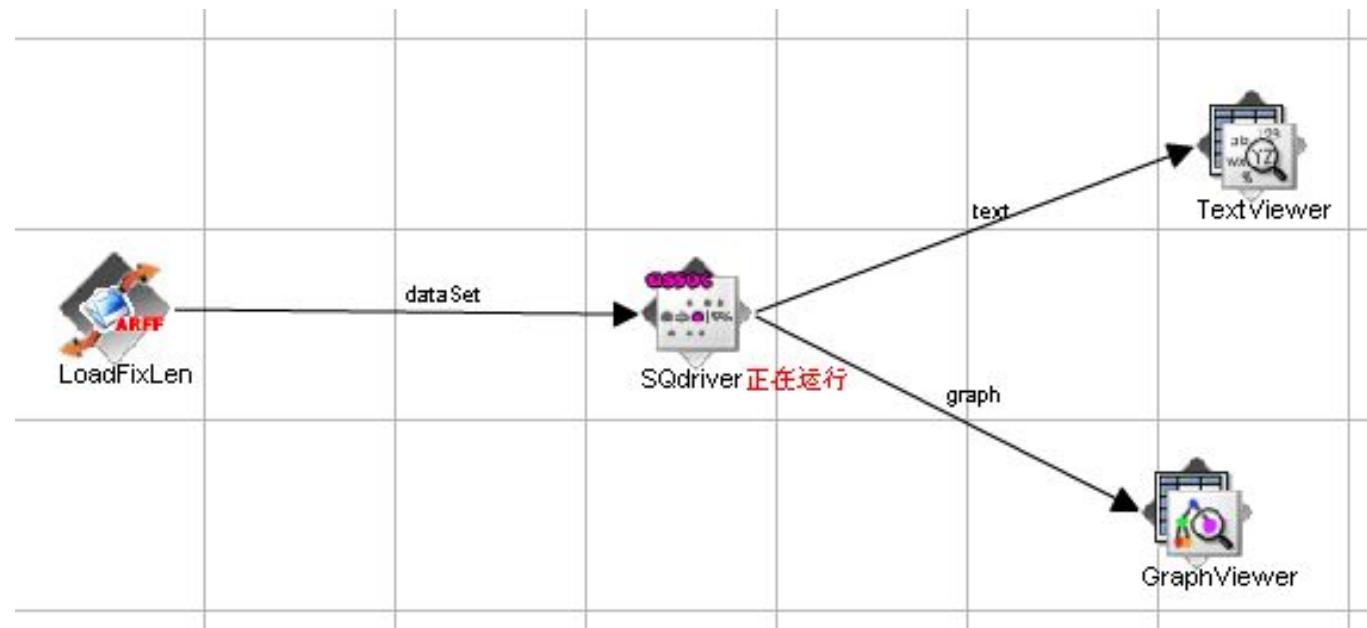
关联规则挖掘工具以选定的客户消费记录全集作为输入，分析消费记录中各消费项间依赖关系，产生规则描述各消费项之间同时出现的规律。

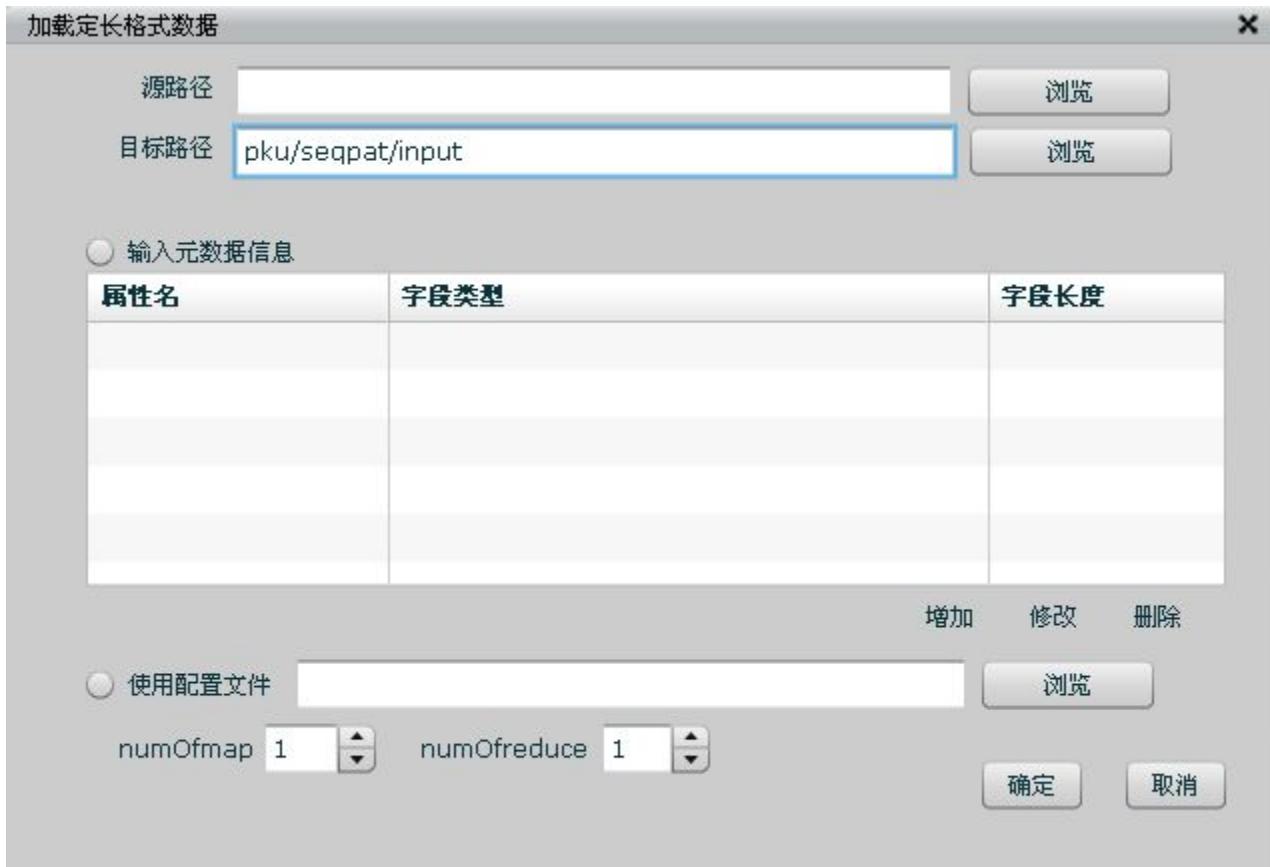
关联规则算法包括：并行PApriori算法、并行PFPgrowth算法、并行PAwfits算、时序关联规则。

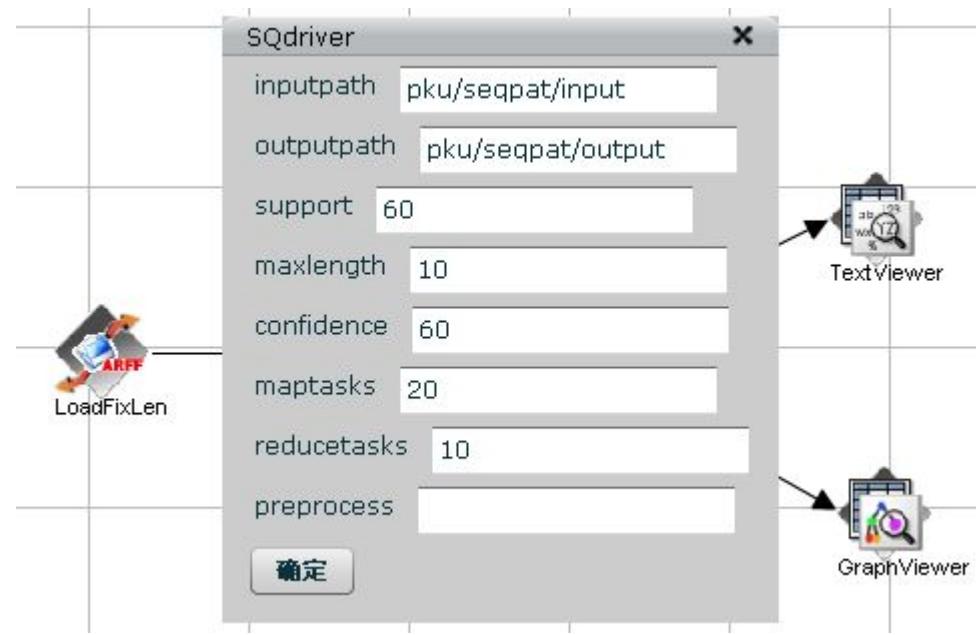


时序关联规则算法

时序关联规则挖掘算法主要可以用于发现交易序列中的频繁模式。因为交易序列具有时间性，因此各数据项集之间存在时间上的先后关系，“买A又买B的顾客往往还会买C”的规则转变成了“买A后又买B的顾客往往会在接下来再买C”或者“买B后又买A的顾客往往会在接下来再买C”，在时序关联规则算法中，这后两者是不同的两个关联规则。







参数	设置
inputpath	设置测试集在DFS上的输入路径, 例如: pku/seqpatt/input
outputpath	设置DFS上的结果输出路径, 例如: pku/seqpatt/output
support	设置序列的支持度(%), 取值范围为: 0 – 100, 一般为20
maxlength	设置序列关联规则的最大长度, 一般取值为6
confidence	设置规则的置信度(%), 若小于该阈值, 则不保存该规则, 否则保存到规则文件中。该阈值取值范围为0~100, 一般取80
maptasks	设置Map的个数, 一般取计算集群核个数的4倍
reducetasks	设置Reduce的个数, 一般取计算集群核个数的2倍
preprocess	设置预处理的输入路径, 预处理的输出路径为inputpath, 若路径为空值, 则不进行预处理过程

聚类算法结果展示

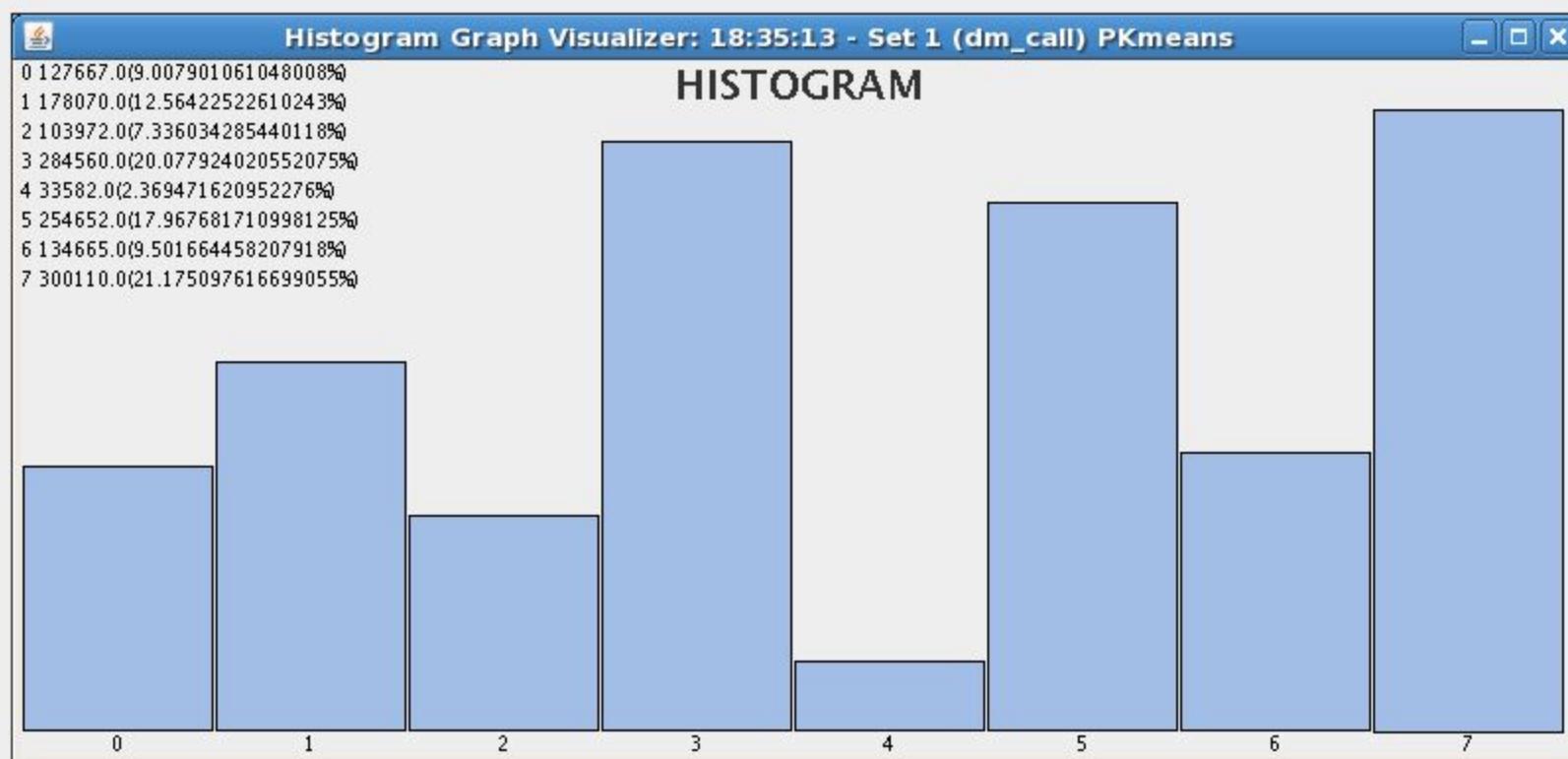


Graph Viewer

Job list

4:45 - Set 1 (dm_call) PKmeans
5:13 - Set 1 (dm_call) PKmeans**CMRI-Blue Carrier-PDM Tools**

Classifiers4MR Clusterers4MR Associations4MR Evaluation Visualization

Arff
Loader4MR

决策树算法结果展示



Result list

08:56:30 - Model: PC45

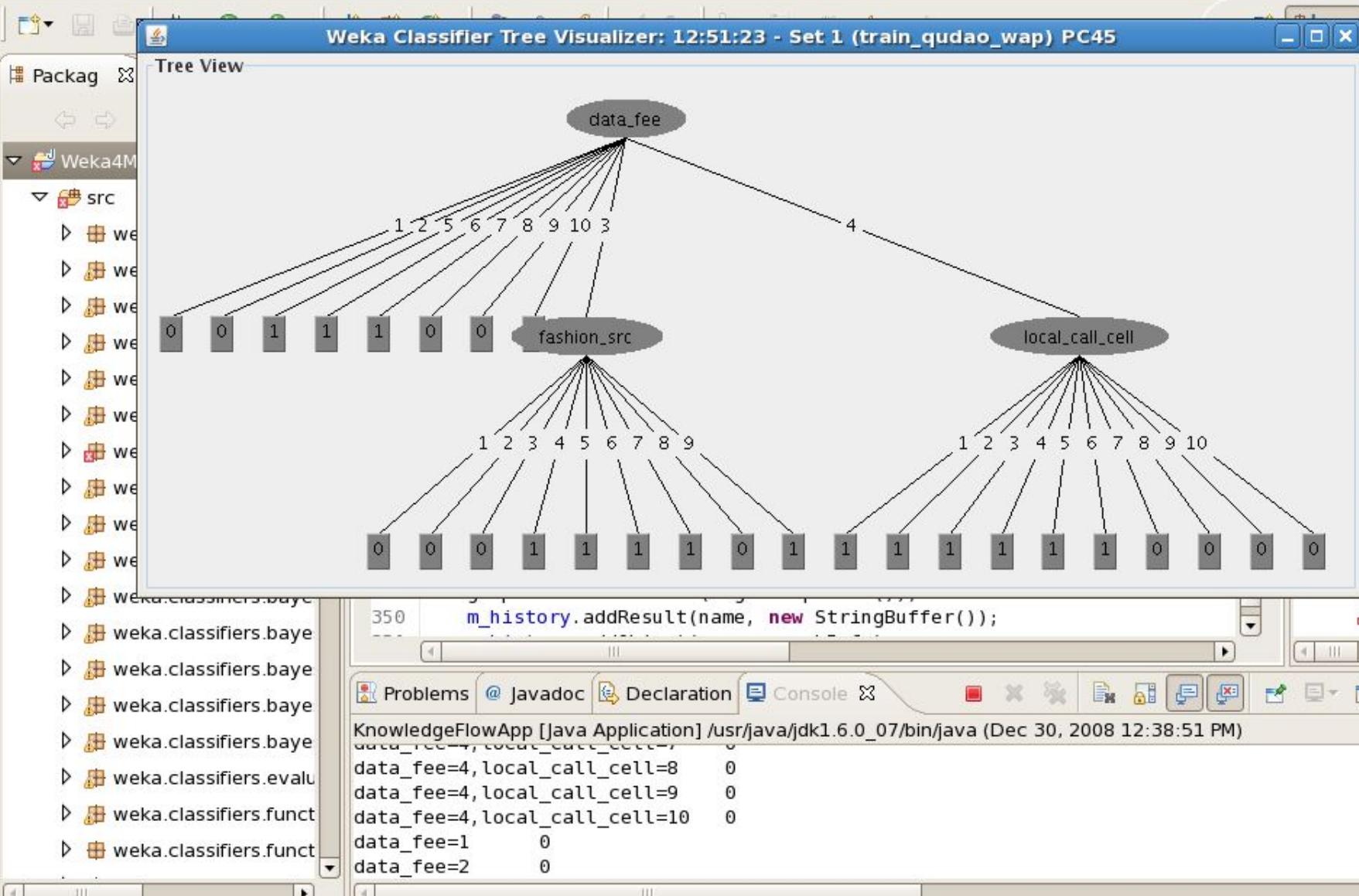
Text

Tree Model

```
call_cell=2      0
call_cell=3      0
call_cell=4      0
call_cell=1
|   data_fee=2   1
|   data_fee=3   0
|   data_fee=4   0
|   data_fee=5   0
|   data_fee=1
|   |   online_id=3   0
|   |   online_id=4   0
|   |   online_id=5   0
call_cell=5
|   local_call_cell=1   0
|   local_call_cell=2   1
|   local_call_cell=3   0
|   local_call_cell=4   0
|   local_call_cell=5
|       day_call_duration=1   0
|       day_call_duration=2   0
|       day_call_duration=3   0
|       day_call_duration=4   0
|       day_call_duration=5
|           In_call_cell=1   0
|           In_call_cell=2   0
|           In_call_cell=3   0
|           In_call_cell=4   0
|           In_call_cell=5
|               |   out_call_cell=1   1
|               |   out_call_cell=2   0
|               |   out_call_cell=3   0
|               |   out_call_cell=4   0
|               |   out_call_cell=5
|                   |   call_cnt=3   0
|                   |   call_cnt=4   0
```

Java - Weka4MR081105/src/weka/gui/beans/GraphViewer.java - Eclipse Platform

File Edit Source Refactor Navigate Search Project Run Window Help



进一步了解云计算

收藏夹 中国云计算 - 中国最大的云计算资料和交流中心

中国云计算 chinacloud.cn

加入收藏 网站地图 网站搜索 简 繁 默

首页 热点 最新 中国云计算大会 概念 国内 国际 名家视点 技术评论 技术实现 网格 云存储 云安全 论文 会议 应用 专题 下载 基础知识 论坛

头条内容



华为对云计算的深刻认识

社会智能化形成的500亿M2B的各种机器终端将实现互联。海量的信息如何高效处理？未来10年，网络数据流量的增长将高达70~100倍，而带来的收入每年仅增长5~10%。如何结构性地将单位流量的成本降低到原来的1/10甚至1/100、以降低投资压力？

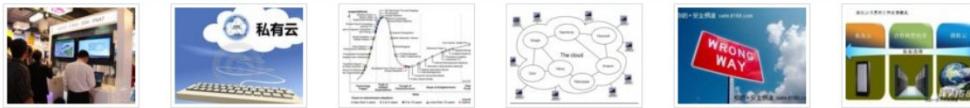
- 实战第一个云程序
- 云计算游戏或致PS、Xbox失败
- 中电信加入全球云计算平台
- 微软中国云计算创新中心落户上海
- 面向企业的云计算技术培训课程
- [PPT]刘鹏：云计算发展的内在动力
- [PPT]刘鹏在中国互联网大会演讲《绿色云计算》
- 《云计算》教材配套14个PPT免费公开
- 云计算：程序员重回个人英雄时代
- 40多本云计算电子书，欢迎下载！
- 对空谈概念已经生厌了？那就看《云计算》教材吧
- 苏州电博会云计算、物联网分论坛
- 云计算让圆周率精确到小数点后2千万亿位

刘鹏：云计算应提升五倍性价比 1 2 3 4

今日热门

- 不懂这显然您不专业！云计算术语大
- 刘鹏：网格计算与云计算(PPT)
- hadoop-0.20.1+120 hive-0.3.99.1+0
- 云计算的20个基本定义
- Hadoop:hbase的搭建

图片内容



随时随地分享数据 中国移动 CIO热捧私有云 成本支 Gartner新兴技术评估 实战第一个云程序 云安全忽视不得 安全 微软私有云 中国企业

◎ 国内第一本系统讲述云计算技术的专业书籍
◎ 由中国云计算专家委员会委员 刘鹏教授 主编



云计算 CLOUD COMPUTING 刘鹏 编著

电子工业出版社 PUBLISHING HOUSE OF ELECTRONICS INDUSTRY http://www.phei.com.cn

<http://www.chinacloud.cn>



谢 谢!

刘鹏

gloud@126.com

中国云计算：<http://www.chinacloud.cn>

中国网格：<http://www.chinagrid.net>