



大数据技术与应用

网络与交换技术国家重点实验室

交换与智能控制研究中心

程祥

提纲-大数据处理和分析

1. 批处理计算

1.1 MapReduce

1.2 Spark

2. 流计算

2.1 Storm

3. 图计算

3.1 Pregel

4. 分析和挖掘

4.1 Hive

4.2 Mahout

1 基于Hadoop的数据仓库Hive

- 1.1 概述
- 1.2 系统详解
- 1.3 编程实践

1.1 概述

- 1.1.1 数据仓库
- 1.1.2 传统数据仓库面临挑战
- 1.1.3 Hive简介
- 1.1.4 Hive的历史由来
- 1.1.5 Hive在Hadoop生态系统中的位置

1.1.1 数据仓库

数据仓库 (Data Warehouse) 是一个面向主题的 (Subject Oriented)、集成的 (Integrated)、相对稳定的 (Non-Volatile)、反映历史变化 (Time Variant) 的数据集合，用于支持管理决策。

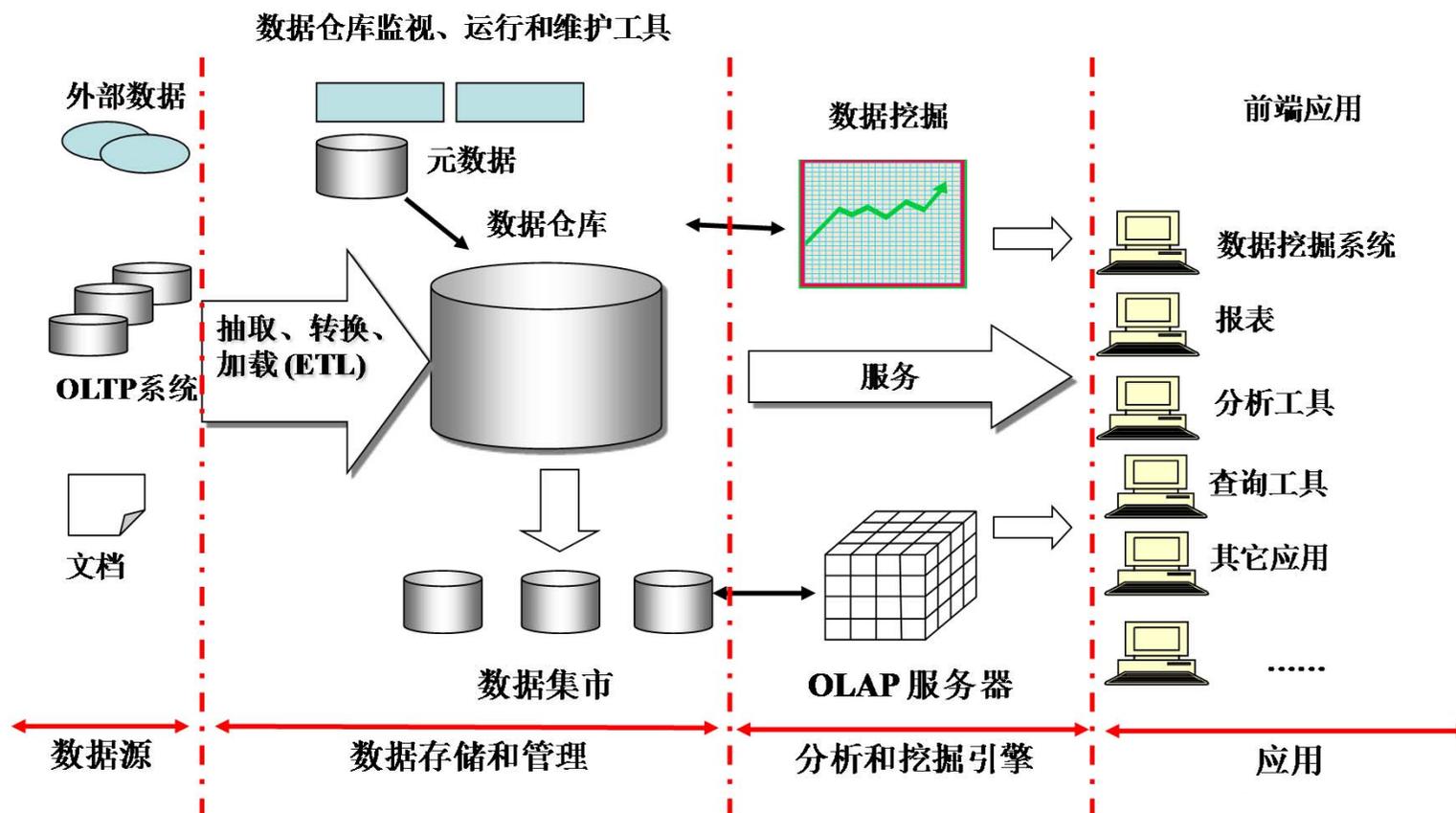


图 数据仓库的体系结构

1.1.2 传统数据仓库面临挑战

在大数据时代，传统的数据仓库主要面临以下挑战：

- ① 无法满足爆炸式增长的海量数据存储需求
- ② 数据量过大导致数据提取、处理速度慢
- ③ 往往依赖专用硬件，难于可扩展

1.1.3 Hive简介

- Hive是一个构建于Hadoop生态系统顶层的数据仓库工具
- 支持大规模数据存储、分析，具有良好的可扩展性
- 依赖分布式文件系统HDFS存储数据
- 依赖分布式并行计算模型MapReduce处理数据
- 定义了简单的类似SQL的查询语言——HiveQL
- 用户可以通过编写的HiveQL语句运行MapReduce任务
- 可以很容易把原来构建在关系数据库上的数据仓库应用程序移植到Hadoop平台上

1.1.4 Hive的历史由来

- Hive是Facebook开发的，构建于Hadoop集群之上的数据仓库应用
- 2008年Facebook将Hive项目贡献给Apache，成为开源项目



1.1.4 Hive的历史由来（续）

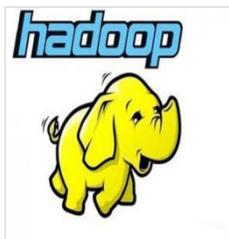
Hive成为Facebook数据仓库的历史



Facebook的数据仓库一开始是构建于MySQL之上的，随着数据量增加某些查询需要几个小时甚至几天才能完成。当数据达到1T时，MySQL进程跨掉。



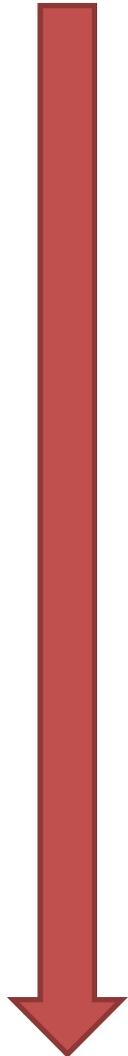
将数据仓库转移到Oracle，虽然可以支撑几个T的数据，但每天收集用户点击流数据（每天约400G）时，Oracle开始撑不住。



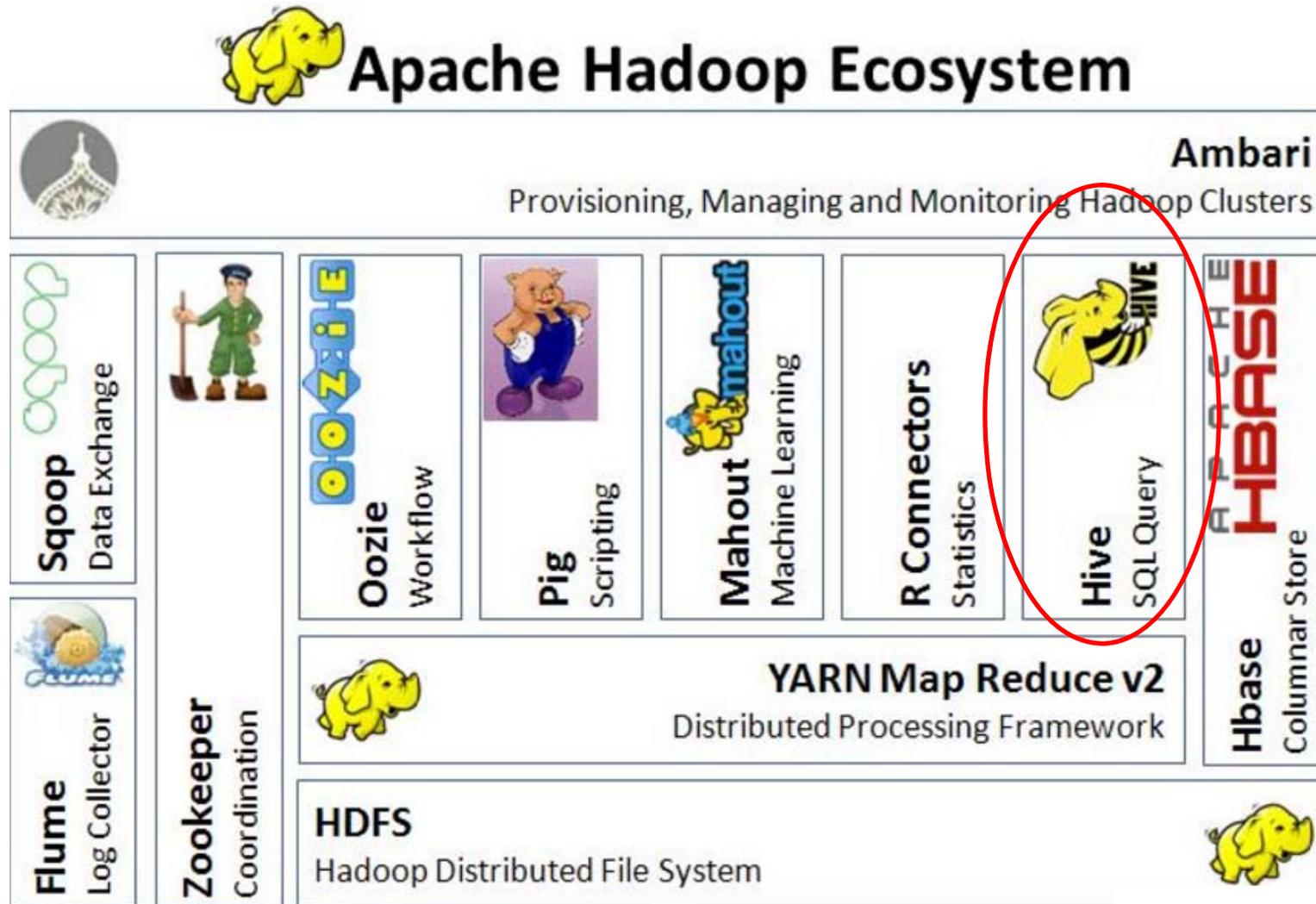
有效解决了大规模数据的存储与统计分析的问题，但是MapReduce程序对于普通分析人员的使用过于复杂和繁琐。



研发Hive，Hive对外提供了类似于SQL语法的HQL语句数据接口，自动将HQL语句编译转化为MR作业后在Hadoop上执行，降低了分析人员使用Hadoop进行数据分析的难度。



1.1.5 Hive在Hadoop生态系统中的位置



1.2 系统详解

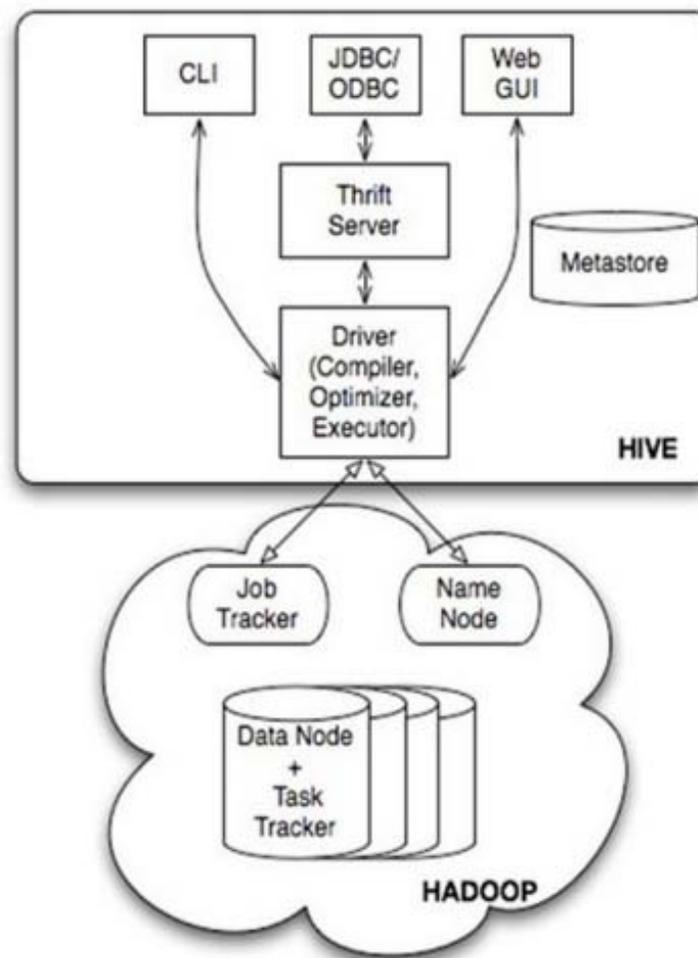
- 1.2.1 系统架构
- 1.2.2 HiveQL (HQL)
- 1.2.2 HQL to MapReduce Job
- 1.2.3 数据模型

1.2.1 系统架构

用户接口：Hive提供了三种用户接口：1) 命令行方式 (CLI)；2) 浏览器方式 (HWI)；3) 客户端方式。其中客户端即是使用JDBC/ODBC驱动通过Thrift server, 远程操作Hive。

驱动模块 (driver)：Hive的核心模块，包括编译器、优化器、执行器等，负责把HQL语句转换成一系列MapReduce作业。

元数据存储模块 (Metastore)：Hive的数据分为表数据和元数据，元存储在一个独立的关系型数据库（系统自带derby数据库，或使用MySQL数据库）。

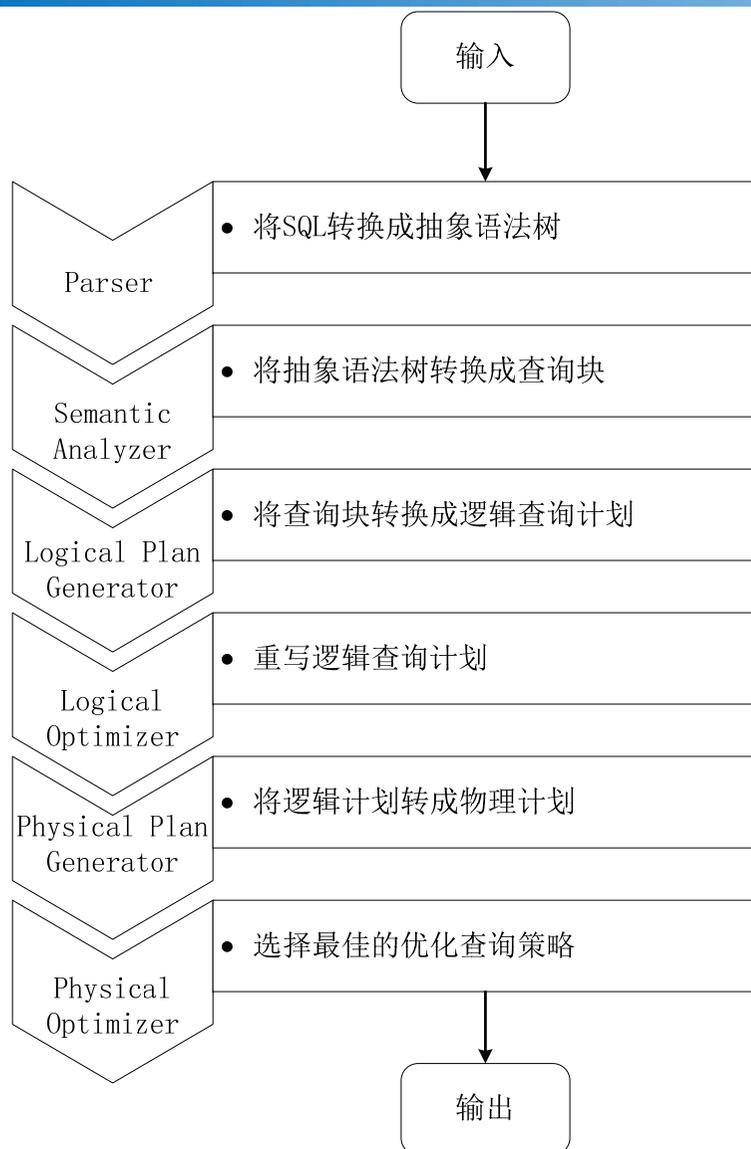


Hive系统架构

1.2.2 HiveQL (HQL)

- HQL是Hive的查询语言
- HQL是一种类SQL语言，与大部分的SQL语法兼容，但不完全支持SQL标准
- HQL不支持行级插入，更新和删除的操作，也不支持事务（老版本）
- HQL只支持少量的连接操作（例如，equality joins），不支持非等值连接，由于非等值连接难以转换为MapReduce作业

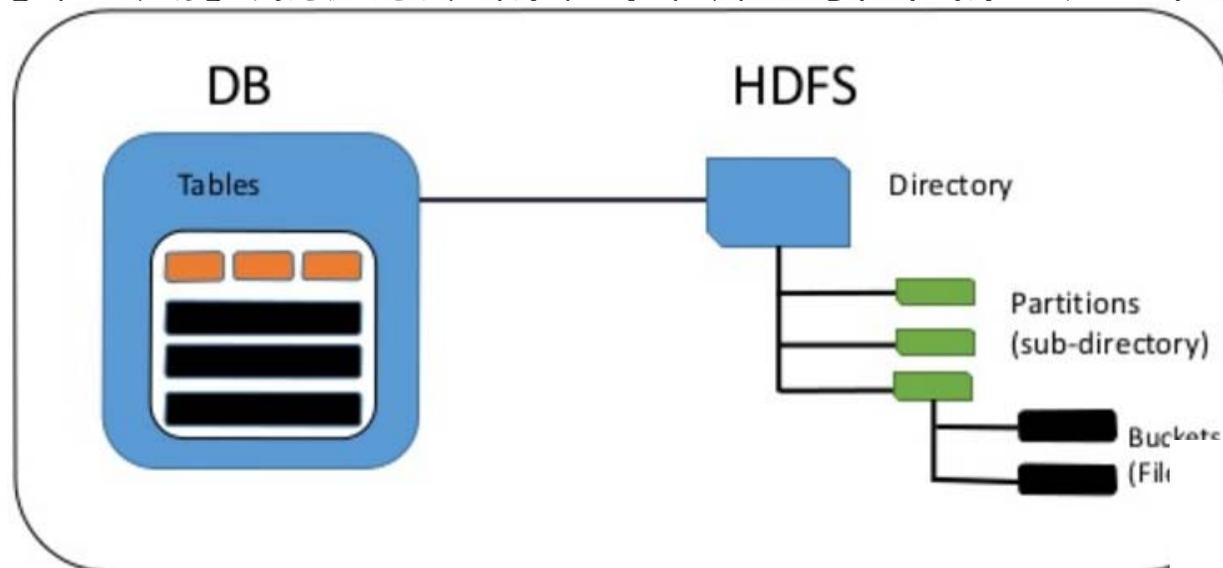
1.2.3 HQL to MapReduce Job



- 1) 由Hive驱动模块中的编译器对用户输入的HQL进行词法和语法解析，将HQL语句转化为抽象语法树的形式
- 2) 抽象语法树的结构仍很复杂，不方便直接翻译为MapReduce算法程序，因此，把抽象语法树进一步转化为查询块
- 3) 把查询块转换成逻辑查询计划，里面包含许多逻辑操作符
- 4) 重写逻辑查询计划，进行优化，合并多余操作，减少MapReduce任务数量
- 5) 将逻辑操作符转换成需要执行的具体MapReduce任务
- 6) 对生成的MapReduce任务进行优化，生成最终的MapReduce任务执行计划
- 7) 由Hive驱动模块中的执行器，对最终的MapReduce任务进行执行输出

1.2.4 数据模型

- Hive中主要包含以下几种数据模型：Database（数据库）、Table（表）、Partition（分区）、Bucket（桶）
- 数据库：相当于关系型数据库中的命名空间，作用是将用户和数据库的应用隔离到不同的数据库或模式中
- 表：表由存储的数据及描述表的一些元数据组成，存储的数据存储在分布式文件系统中（HDFS中与表名相同的目录下），元数据存储在关系型数据库中
- 分区：根据“分区列”的值对表的数据进行粗略划分的机制，分区数据存储在表的主目录下的一个子目录，可以避免扫描整表，加速查询
- 桶：对于每一个表或者分区，可以进一步（通过hash）组织成桶，也就是说桶是更为细粒度的数据范围划分，每个桶就是表（或分区）目录里的一个文件



1.3 编程实践

- 1.3.1 Hive的数据类型
- 1.3.2 Hive的基本操作
- 1.3.3 Hive编程示例：WordCount

1.3.1 Hive的数据类型

表 Hive的基本数据类型

类型	描述	示例
TINYINT	1个字节 (8位) 有符号整数	1
SMALLINT	2个字节 (16位) 有符号整数	1
INT	4个字节 (32位) 有符号整数	1
BIGINT	8个字节 (64位) 有符号整数	1
FLOAT	4个字节 (32位) 单精度浮点数	1.0
DOUBLE	8个字节 (64位) 双精度浮点数	1.0
BOOLEAN	布尔类型, true/false	true
STRING	字符串, 可以指定字符集	"xmu"
TIMESTAMP	整数、浮点数或者字符串	1327882394 (Unix新纪元秒)
BINARY	字节数组	[0,1,0,1,0,1,0,1]

1.3.1 Hive的数据类型

表 Hive的集合数据类型

类型	描述	示例
ARRAY	一组有序字段，字段的类型必须相同	Array(1,2)
MAP	一组无序的键/值对，键的类型必须是原子的，值可以是任何数据类型，同一个映射的键和值的类型必须相同	Map('a',1,'b',2)
STRUCT	一组命名的字段，字段类型可以不同	Struct('a',1,1,0)

1.3.2 Hive的基本操作

1. create: 创建数据库、表

- 创建数据库

- ① 创建数据库hive

```
hive> create database hive;
```

- ② 创建数据库hive。因为hive已经存在，所以会抛出异常，加上if not exists关键字，则不会抛出异常

```
hive> create database if not exists hive;
```

- 创建表

- ① 在hive数据库中，创建表usr，含三个属性id, name, age

```
hive> use hive;
```

```
hive> create table if not exists usr(id bigint, name string, age int);
```

1.3.2 Hive的基本操作

2. show: 查看数据库、表

- 查看数据库

- ① 查看Hive中包含的所有数据库

```
hive> show databases;
```

- ② 查看Hive中以h开头的数据库

```
hive> show databases like 'h.*' ;
```

- 查看表

- ① 查看数据库hive中所有表

```
hive> use hive;
```

```
hive> show tables;
```

- ② 查看数据库hive中以u开头的表

```
hive> show tables in hive like 'u.*' ;
```

1.3.2 Hive的基本操作

3. load: 向表中装载数据

- ① 把目录' /usr/local/data' 下的数据文件中的数据装载进usr表并覆盖原有数据

```
hive> load data local inpath '/usr/local/data' overwrite into table usr;
```

- ② 把目录' /usr/local/data' 下的数据文件中的数据装载进usr表不覆盖原有数据

```
hive> load data local inpath '/usr/local/data' into table usr;
```

- ③ 把分布式文件系统目录' hdfs://master_server/usr/local/data' 下的数据文件数据装载进usr表并覆盖原有数据

```
hive> load data inpath 'hdfs://master_server/usr/local/data' overwrite into table usr;
```

1.3.2 Hive的基本操作

4. insert: 向表中插入数据或从表中导出数据

- ① 向表usr1中插入来自usr表的数据并覆盖原有数据

```
hive> insert overwrite table usr1 select * from usr  
where age=10;
```

- ② 向表usr1中插入来自usr表的数据并追加在原有数据后

```
hive> insert into table usr1 select * from usr where  
age=10;
```

1.3.3 Hive编程示例：WordCount

- 首先，需要创建一个需要分析的输入数据文件
- 然后，编写HQL语句实现WordCount算法
- 具体步骤如下：

(1) 创建input目录，其中input为输入目录。命令如下：

```
$ cd /usr/local/hadoop  
$ mkdir input
```

(2) 在input文件夹中创建两个测试文件file1.txt和file2.txt，命令如下：

```
$ cd /usr/local/hadoop/input  
$ echo "hello world" > file1.txt  
$ echo "hello hadoop" > file2.txt
```

1.3.3 Hive编程示例：WordCount

(3) 进入hive命令行界面，编写HiveQL语句实现WordCount算法，命令如下：

```
$ hive
```

```
hive> create table docs(line string);
```

```
hive> load data inpath 'input' overwrite into table docs;
```

```
hive> create table word_count as
```

```
select word, count(1) as count from
```

```
(select explode(split(line, ' ')) as word from docs) w
```

```
group by word
```

```
order by word;
```

执行完成后，用select语句查看运行结果如下：

```
OK
Time taken: 2.662 seconds
hive> select * from word_count;
OK
hadoop 1
hello 2
world 1
Time taken: 0.043 seconds, Fetched: 3 row(s)
```

docs表

hello world
Hello hadoop

w表word列

word
hello
world
hello
hadoop

Hive小结

- Hive是一个构建于Hadoop之上的数据仓库工具，它依赖HDFS存储数据，依赖MapReduce处理数据。
- Hive的核心模块是驱动模块和元数据存储模块，驱动模块负责将用户提交的HQL转换为MapReduce作业，元数据存储模块负责管理表等信息
- Hive的数据模型主要包括数据库、表、分区和桶；表中的数据可以通过分区进行粗粒度划分，通过桶进行细粒度的划分；分区和桶都是为了加速查询

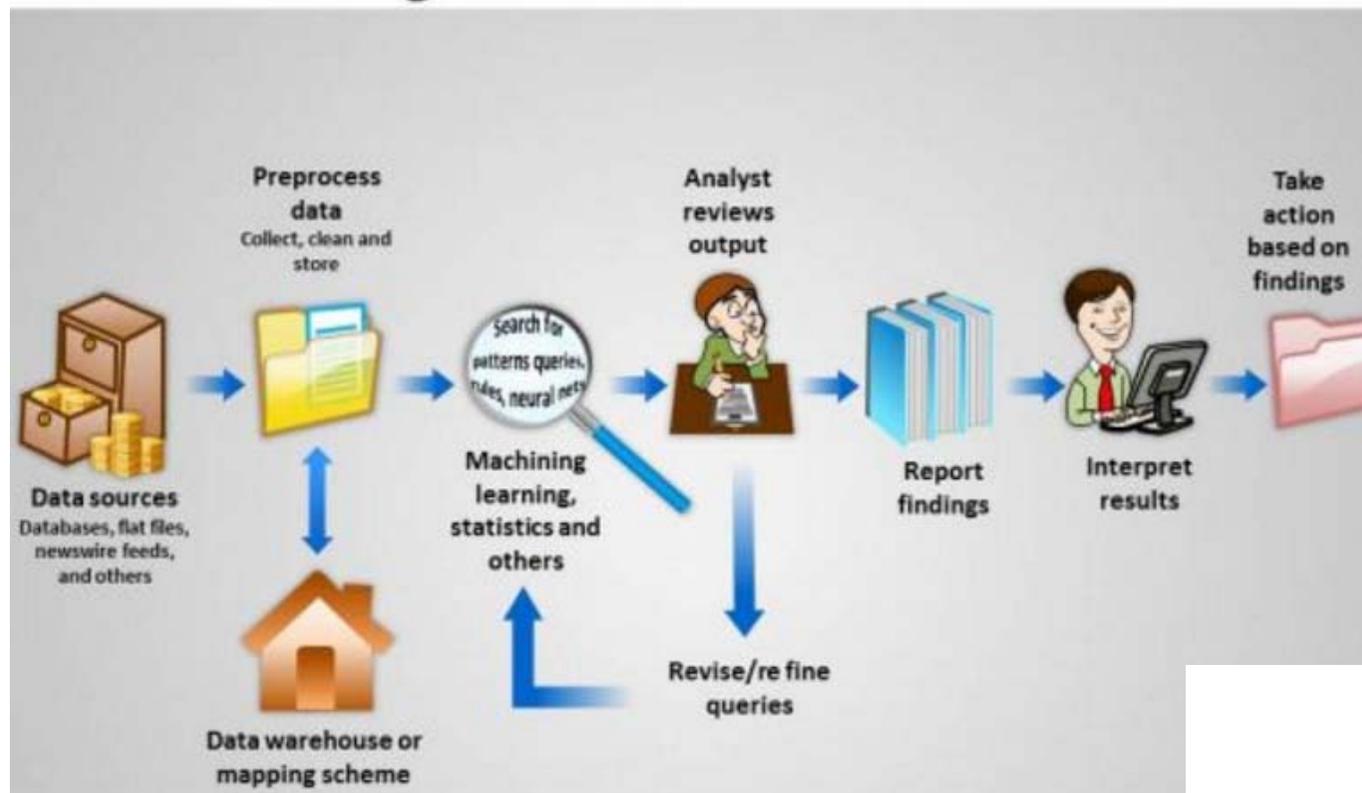
2. 基于Hadoop的数据挖掘工具Mahout

- 2.1 数据挖掘
- 2.2 大数据时代
- 2.3 Mahout

2.1 数据挖掘

- 数据挖掘一般是指从大量的数据中发现有意义的模式或知识的过程。
- 数据挖掘通常与计算机科学有关，并通过统计、机器学习等诸多方法来实现
- 数据挖掘具在互联网、电信、金融、生物医药等领域有着广泛应用

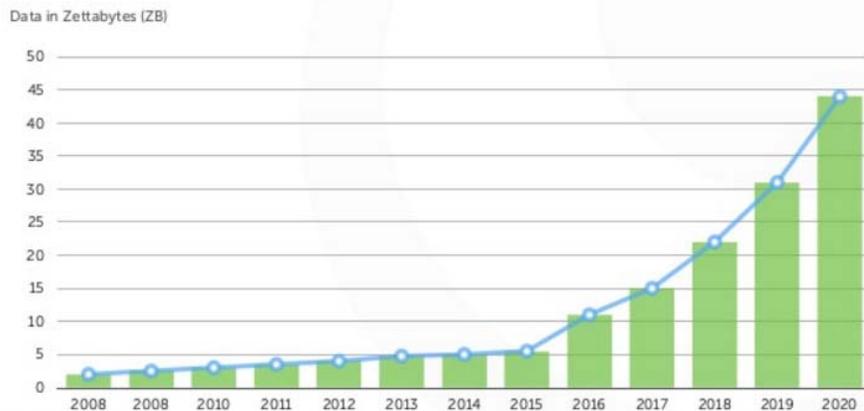
Data Mining Process



2.2 大数据时代

- 近年来，随着互联网、物联网、云计算等技术的快速发展，以及智能终端、网络社会、数字地球等信息体的普及和建设，全球数据量出现爆炸式增长，人们已经步入“大数据时代”。
- 麦肯锡在2011年5月发布的一份报告：《大数据：创新、竞争和生产力的下一个新领域》。在这份报告里麦肯锡指出，数据已经渗透到每一个行业和业务职能领域，逐渐成为重要的生产因素；而人们对于海量数据的运用将预示着新一波生产率增长和消费者盈余浪潮的到来。

Data is growing at a 40 percent compound annual rate, reaching nearly 45ZB by 2020

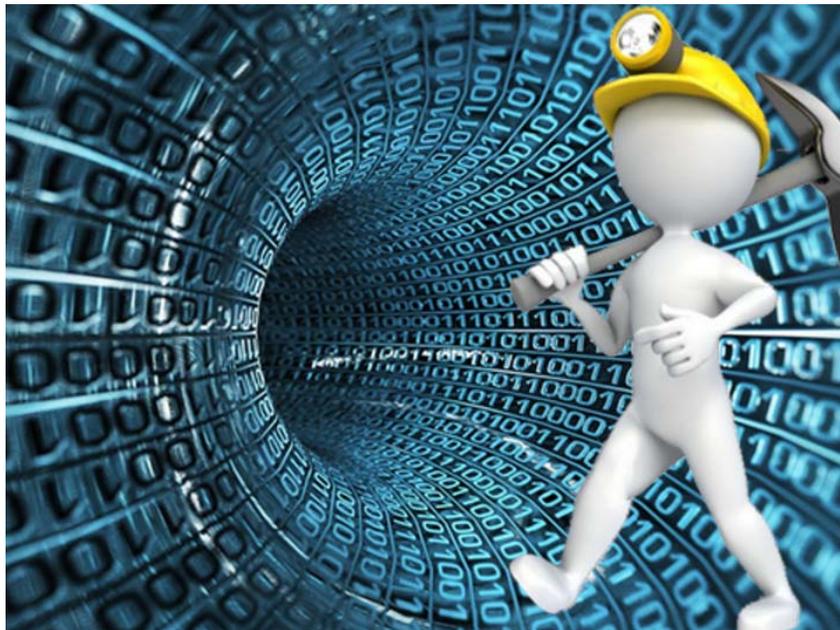


Source: Oracle, 2012



2.2 大数据时代

- 大数据中蕴含着巨大价值，被看做是世纪“新石油”。通过对大数据的分析和挖掘来提供信息服务，已成为信息服务提供的主要手段。例如，通过对大量互联网用户的网上的行为记录的分析 and 挖掘可以实现精准的内容推荐。



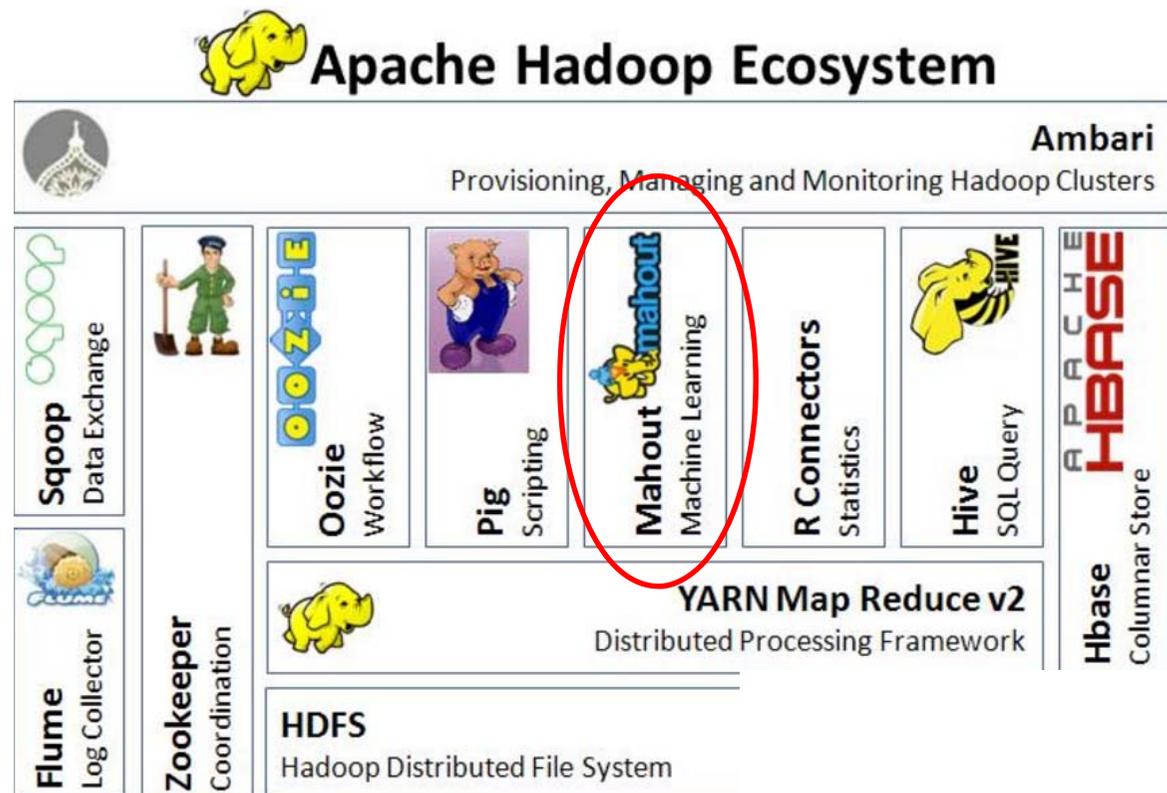
传统的数据挖掘工具难以满足“大数据”挖掘需求！

2.3 Mahout

- 2.3.1 Mahout简介
- 2.3.2 Mahout典型应用
- 2.3.3 Mahout编程示例

2.3.1 Mahout简介

- Mahout是Apache Software Foundation (ASF) 旗下的一个开源项目，提供多种可扩展的机器学习领域经典算法的实现，旨在帮助开发人员更加方便快捷地创建智能应用程序。
- Mahout基于hadoop平台，它利用MapReduce计算框架将单机算法扩展到分布式计算环境，大大提升了算法可处理数据的规模和处理性能。



2.3.2 Mahout典型应用

What Can I do with Mahout Right Now?

3C + FPM + O = 

3C: **C**ollaborative Filtering, **C**lustering, **C**lassification

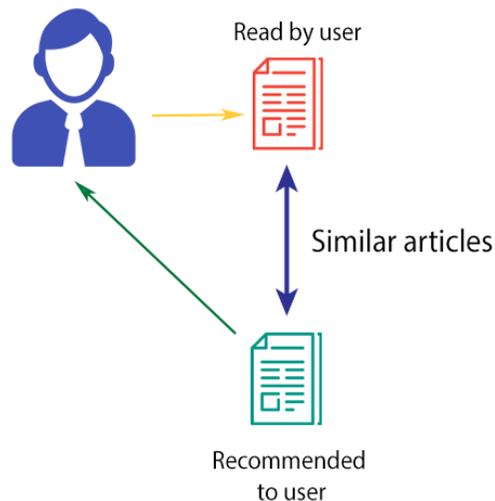
FPM: **F**requent **P**attern **M**ining

O: **O**thers

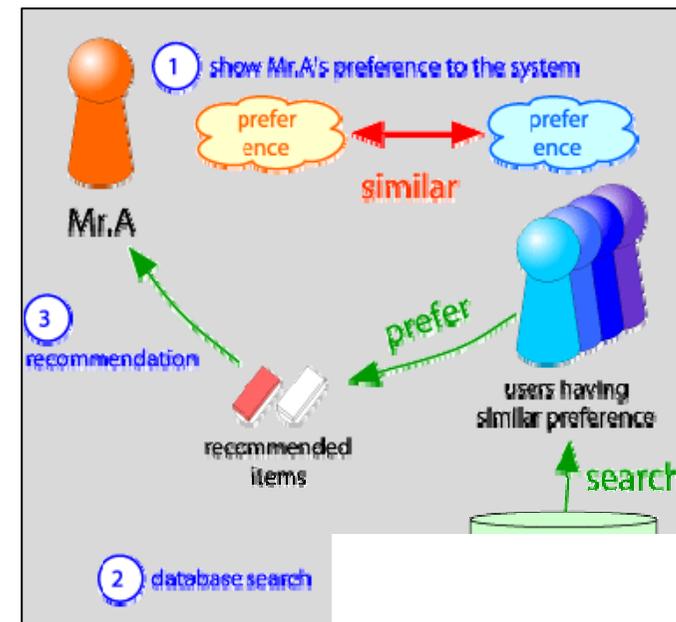
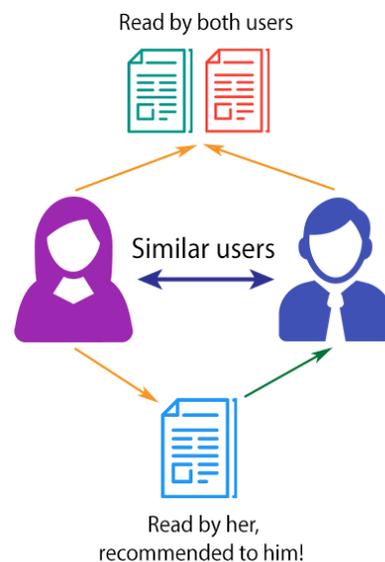
2.3.2 C1: Collaborative Filtering

- 与传统的基于内容过滤直接分析内容进行推荐不同，协同过滤分析用户兴趣，在用户群中找到指定用户的相似（兴趣）用户，综合这些相似用户对某一信息的评价，形成系统对该指定用户对此信息的喜好程度预测，并进行推荐。
- Mahout实现包括user-based CF和item-based CF在内的多种CF算法，并支持多种相似度计算（例如，Cosine相似度等）

CONTENT-BASED FILTERING

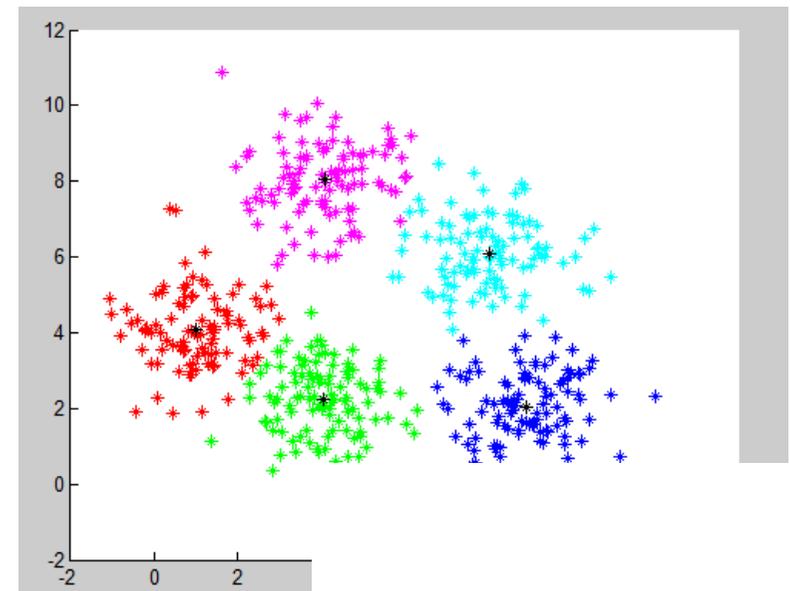
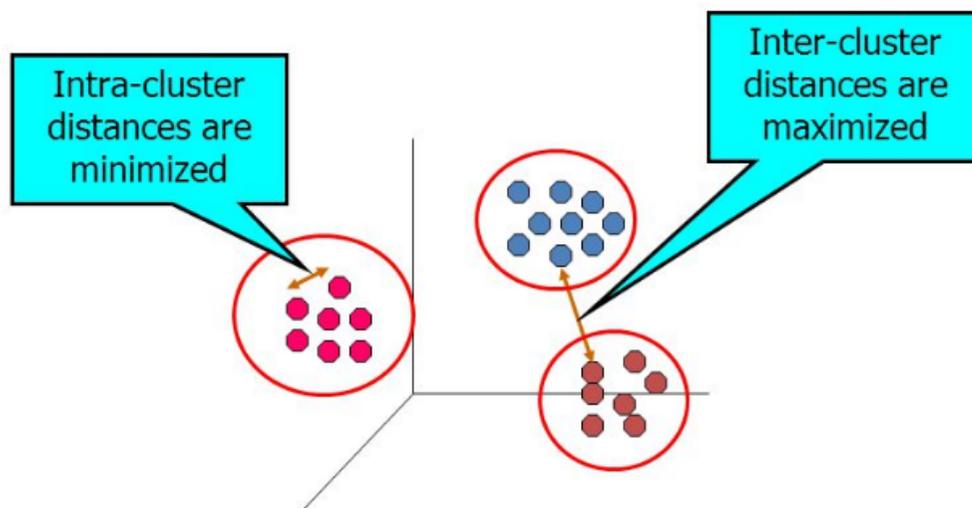


COLLABORATIVE FILTERING



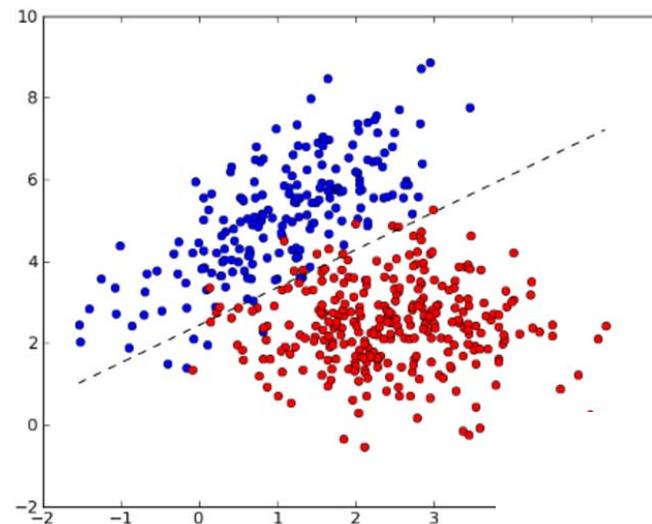
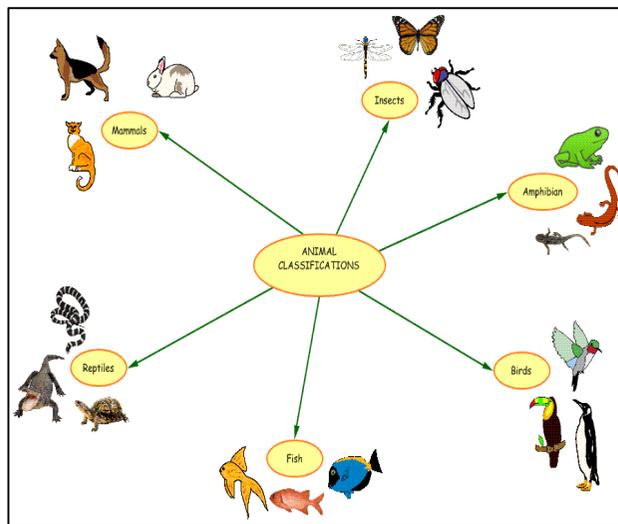
2.3.2 C2: Clustering

- 聚类 (Clustering) 是将一组元素划分为若干个簇 (类), 使得簇内元素距离最小化, 簇外元素最大化
- Clustering是无监督学习, 无需有标签数据进行训练
- 具有广泛应用, 例如, 用户群划分, 新闻文档聚类等
- Mahout实现了包括K-Means, Fuzzy K-Means, Density-Based在内的多种聚类算法, 并支持包括欧几里得、马哈顿距离在内的多种距离计算



2.3.2 C3: Classification

- 分类 (Classification) 是将新的元素按照某种标准 (分类器) 打上标签, 并根据该标签区分类别
- Classification是有监督学习, 需要有标签的数据作为训练集训练分类器
- 具有广泛应用, 例如, 垃圾邮件识别、广告点击率预估等
- Mahout实现了包括朴素贝叶斯、决策树、Logistic回归、神经网络等在内的多种分类算法



2.3.2 FPM : Frequent Pattern Mining

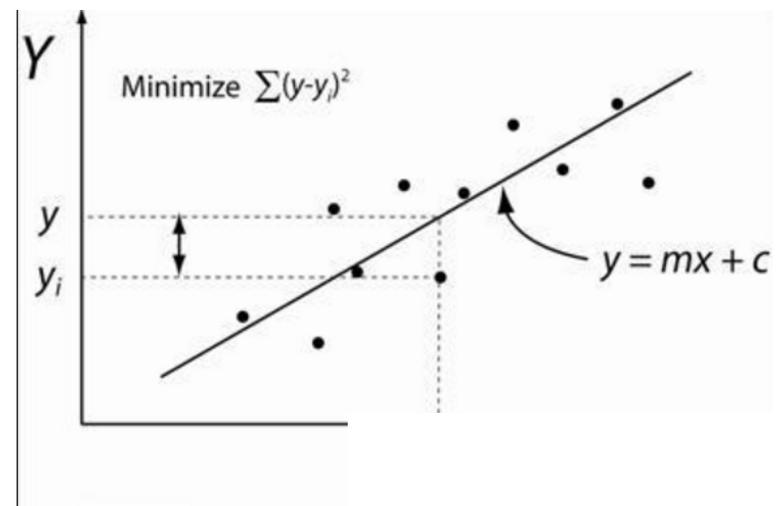
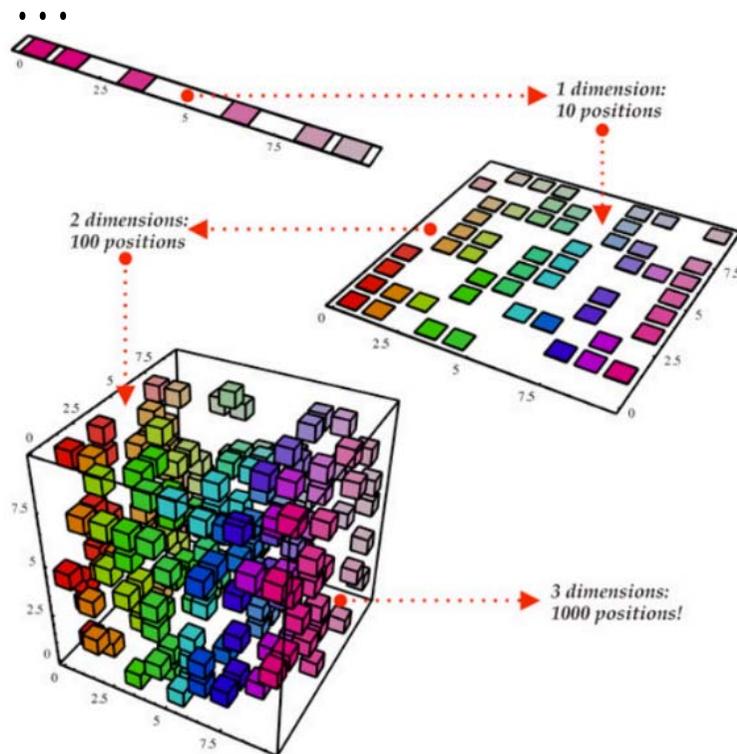
- 频繁模式挖掘 (Frequent Pattern Mining) 是从大量记录中挖掘频繁出现的模式
- 具有广泛应用, 例如, 购物篮分析 (尿布和啤酒的故事), Web点击流分析, DNA序列分析等
- Mahout实现了经典的频繁项集挖掘算法FP-Growth



2.3.2 O : Others...

- 数据降维 (Dimension Reduction)
- 线性回归 (Linear Regression)
- 数学运算 (Math Library)

•



2.3.3 Mahout编程示例

- 任务使用Mahout编写一个user-based CF的推荐程序
- 输入数据存放在c:\input.txt, 格式为(用户id, 商品id, 打分), 具体内容:

```
1,101,5  
1,102,3  
1,103,2.5  
2,101,2  
2,102,2.5  
2,103,5  
2,104,2  
3,101,2.5  
3,104,4  
3,105,4.5  
3,107,5  
4,101,5  
4,103,3  
4,104,4.5  
4,106,4  
5,101,4  
5,102,3  
5,103,2  
5,104,4  
5,105,3.5  
5,106,4
```

2.3.3 Mahout编程示例

- 主函数
- ```
public class TestMahout {
 // private TestMahout(){};
 public static void main(String args[]) throws Exception {
 TestMahout testMahout = new TestMahout();
 System.out.println("The UBCF Result:");
 testMahout.UBCF();
 }
}
```

## 2.3.3 Mahout编程示例

- baseUserCF函数片段

- ```
public void UBCF(){
    DataModel dataModel;
    try{
        dataModel = new FileDataModel(new File("c://input.txt")); // 1.读入数据
        UserSimilarity userSimilarity = new PearsonCorrelationSimilarity(dataModel); // 2.相似度
        UserNeighborhood userNeighborhood = new NearestNUserNeighborhood(2, userSimilarity,
            dataModel); // 3.查找k紧邻
        Recommender recommender = new GenericUserBasedRecommender(dataModel,
            userNeighborhood, userSimilarity); // 4.推荐引擎构建
        // 为用户i推荐两个Item
        for (int i = 1; i < 6; i++) {
            System.out.println("recommand for user:" + i);
            List<RecommendedItem> recommendations = recommender.recommend(i, 2);
            for (RecommendedItem recommendation : recommendations) {
                System.out.println(recommendation);
            }
        }
        ...
    }
}
```

2.3.3 Mahout编程示例

- 推荐结果

recommand for user:1

RecommendedItem[item:104, value:4.257081]

RecommendedItem[item:106, value:4.0]

recommand for user:2

recommand for user:3

RecommendedItem[item:106, value:4.0]

RecommendedItem[item:103, value:2.5905366]

recommand for user:4

RecommendedItem[item:102, value:3.0]

recommand for user:5

Mahout小结

- Mahout是开源的、基于Hadoop的数据挖掘工具
- Mahout利用MapReduce计算框架实现了常用的数据挖掘算法，包括分类、聚类、频繁模式挖掘等算法
- 用户可以基于Mahout快速研发智能应用程序

谢谢