

大数据处理综合实验

2024-2025学年 第2学期 复习大纲



01 并行计算与大数据处理技术概述

1. 为什么需要并行计算？

- 提高计算机性能有哪些基本技术手段
 - 提高字长，流水线微体系结构技术，提高集成度，提升主频
- 迫切需要发展并行计算技术的主要原因
 - 单处理器性能提升达到极限
 - 应用规模和数据量急剧增大，超大的计算量/计算复杂度

2. 并行计算技术的分类

- 有哪些主要的并行计算分类方法？
 - 按数据和指令处理结构：弗林(Flynn)分类
 - 按并行类型
 - 按存储访问构架
 - 按系统类型
 - 按计算特征
 - 按并行程序设计模型/方法



01 并行计算与大数据处理技术概述

3. 并行计算的主要技术问题

■ 并行计算有哪些方面的主要技术问题？

- 多核/多处理器网络互连结构技术
- 存储访问体系结构
- 分布式数据与文件管理
- 并行计算任务分解与算法设计
- 并行程序设计模型和方法
- 数据同步访问和通信控制
- 可靠性设计与容错技术
- 并行计算软件框架平台
- 系统性能评价和程序并行度评估

$$S = \frac{1}{(1 - P) + \frac{P}{N}}$$

■ 如何评估程序的可并行度(掌握Amdahl定律的重要作用)

- 一个并行程序可加速程度是有限制的，并非可无限加速，并非处理器越多越好



01 并行计算与大数据处理技术概述

4. MPI并行程序设计

- MPI功能与特点
- MPI程序结构
- MPI基本编程接口
- MPI编程实例

5. 为什么需要大规模数据并行处理?

- ▣ 处理数据的能力大幅落后于数据增长
- ▣ 海量数据隐含着更准确的事实



01 并行计算与大数据处理技术概述

6. 什么是**MapReduce**?

- ▣ 基于集群的高性能并行计算平台(Cluster Infrastructure)
- ▣ 并行程序开发与运行框架(Software Framework)
- ▣ 并行程序设计模型与方法(Programming Model & Methodology)

7. 为什么**MapReduce**如此重要?

- ▣ 高效的大规模数据处理方法
- ▣ 改变了大规模尺度上组织计算的方式
- ▣ 第一个不同于冯诺依曼结构的、基于集群而非单机的计算方式的重大突破
- ▣ 目前为止最为成功的基于大规模计算资源的并行计算抽象方法



02 MapReduce 简介

1. 对付大数据处理-分而治之

- 大数据分而治之的并行化计算
- 大数据任务划分和并行计算模型

2. 构建抽象模型：**Map**和**Reduce**

- 主要设计思想：
为大数据处理过程中的两个主要处理操作提供一种抽象机制
- 典型的流式大数据问题的特征
- **Map**和**Reduce**操作的抽象描述
提供一种抽象机制，把做什么和怎么做分开，程序员仅需要描述做什么，不需要关心怎么做
- 基于**Map**和**Reduce**的并行计算模型和计算过程



02 MapReduce 简介

3. 上升到构架-自动并行化并隐藏底层细节

- 主要需求、目标和设计思想
 - 实现自动并行化计算
 - 为程序员隐藏系统层细节
- MapReduce提供统一的构架并完成以下的主要功能
 - 任务调度
 - 数据/代码互定位
 - 出错处理
 - 分布式数据存储与文件管理
 - Combiner和Partitioner（设计目的和作用）



02 MapReduce 简介

4. MapReduce 的主要设计思想和特征

- ▣ 向“外”横向扩展，而非向“上”纵向扩展
- ▣ 失效被认为是常态
- ▣ 把计算处理向数据迁移
- ▣ 顺序处理数据、避免随机访问数据
- ▣ 为应用开发者隐藏系统层细节
- ▣ 平滑无缝的可扩展性

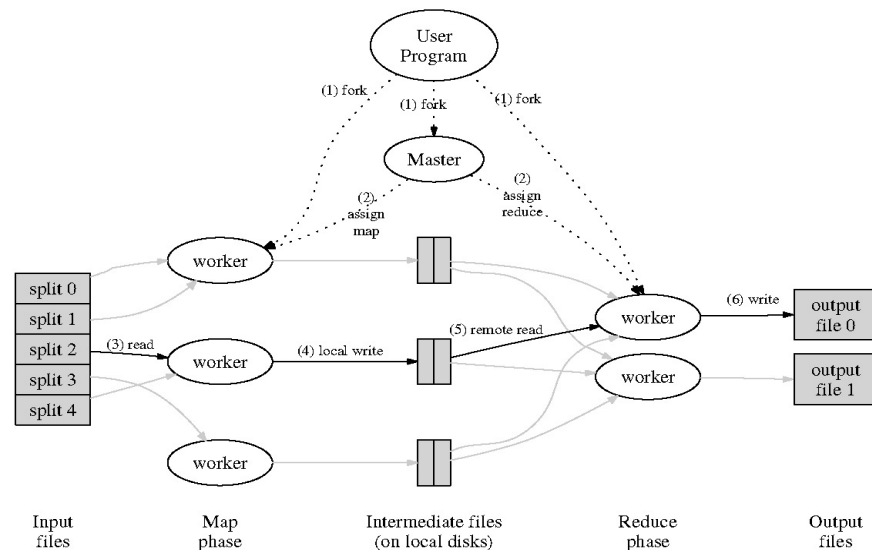


03-1 Google MapReduce基本架构

1. MapReduce的基本模型和处理思想

2. Google MapReduce的基本工作原理

- Google MapReduce并行处理的基本过程
- 带宽优化（**Combiner**的设计目的和作用）
- 用数据分区解决数据相关性问题（**Partitioner**的设计目的和作用）
- 失效处理
- 计算优化





03-1 Google MapReduce基本架构

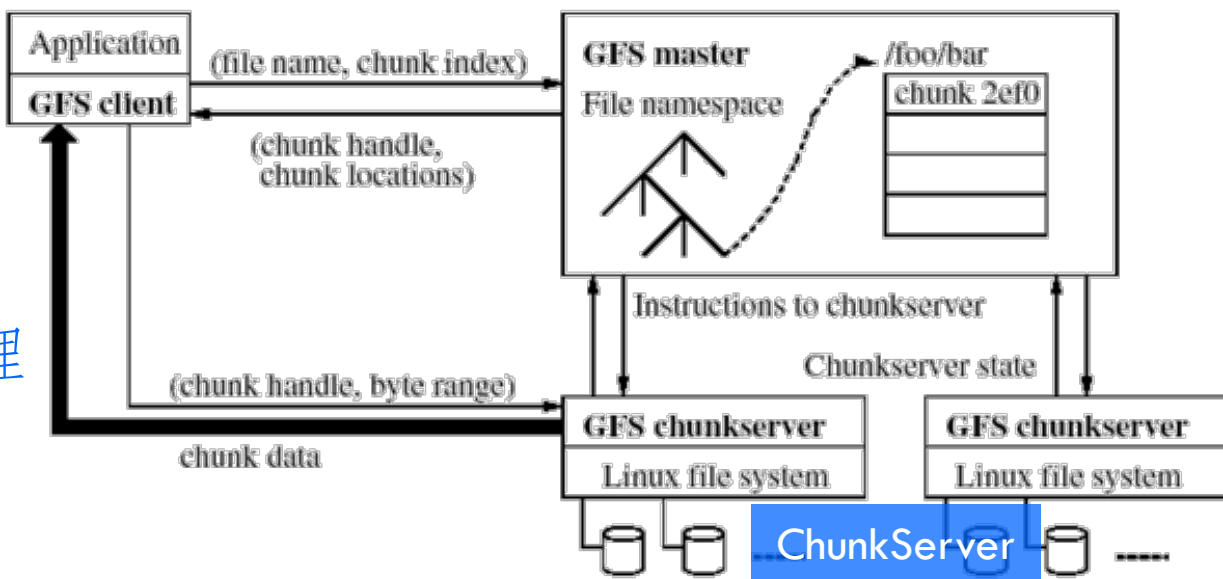
3. 分布式文件系统GFS的基本工作原理

■ Google GFS的基本设计原则

- 廉价本地磁盘分布存储
- 多数据自动备份解决可靠性
- 为上层的MapReduce计算框架提供支撑

■ Google GFS的基本构架和工作原理

- GFS Master的主要作用
- GFS ChunkServer的主要作用
- 数据访问工作过程
- GFS的系统管理技术





03-1 Google MapReduce基本架构

4. 分布式结构化数据表BigTable

- BigTable的基本作用和设计思想
- BigTable设计动机和目标
 - 需要存储管理海量的结构化半结构化数据
 - 海量的服务请求
 - 商用数据库无法适用



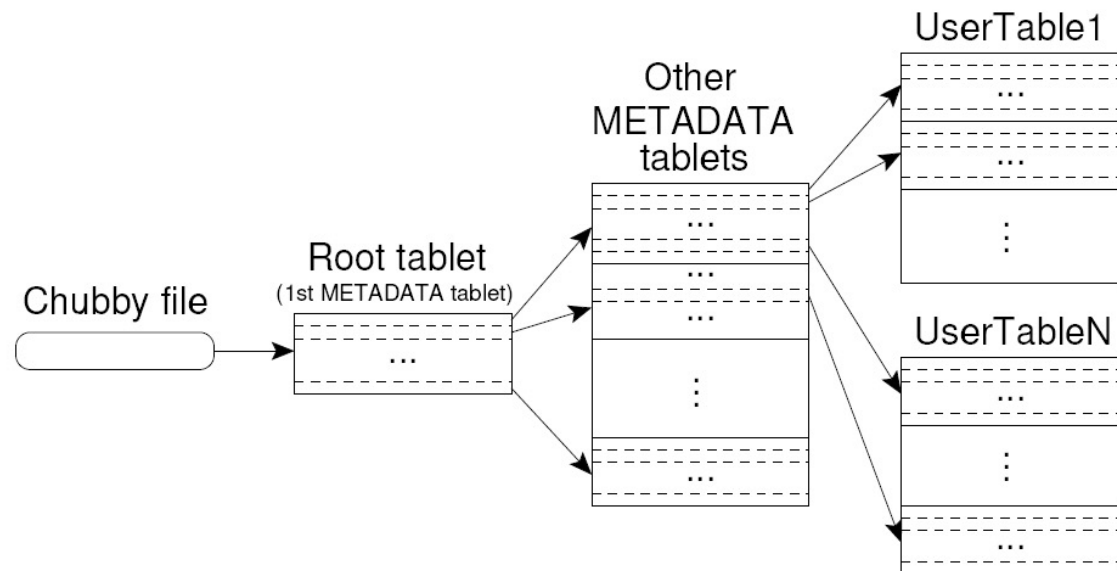
03-1 Google MapReduce基本架构

■ BigTable数据模型—多维表

- 一个行关键字(row key)
- 一个列关键字(column key)
- 一个时间戳(time stamp)

■ BigTable基本构架

- 子表服务器
- 子表存储结构SSTable（对应于GFS数据块）
- 子表数据格式
- 子表寻址：3级B+树

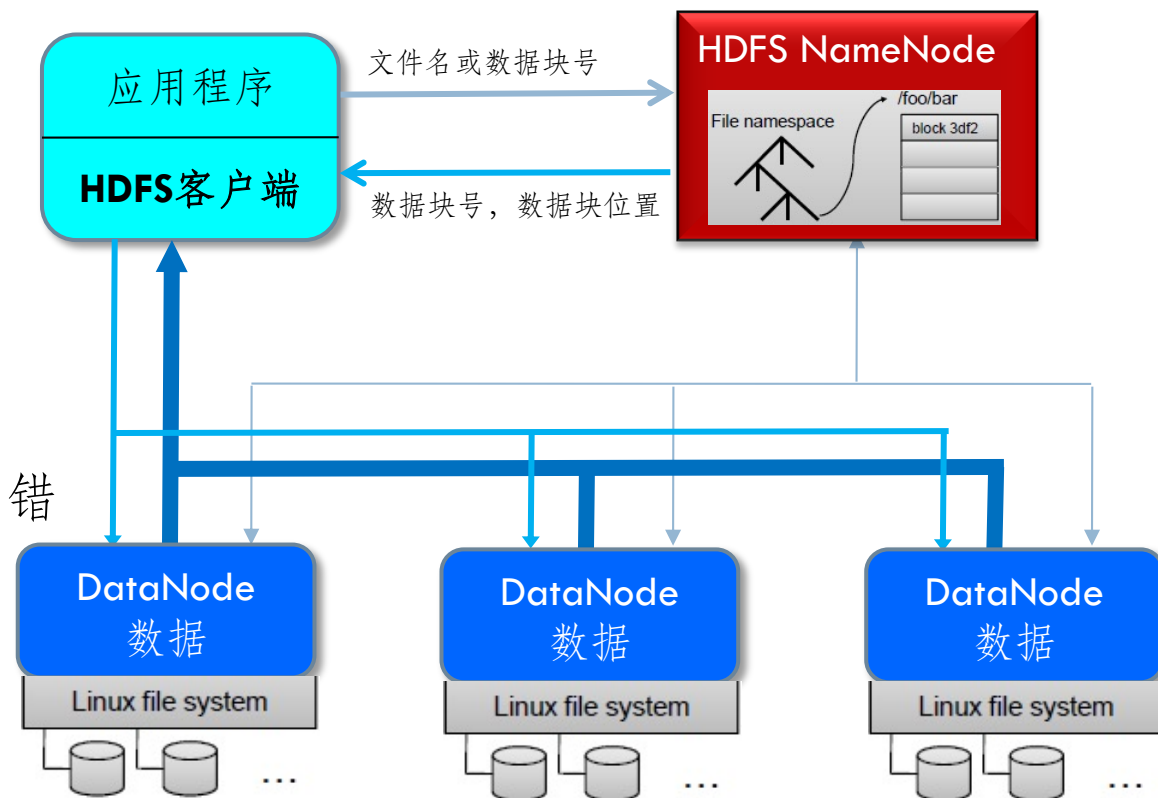




03-2 Hadoop MapReduce基本架构

1. Hadoop 分布式文件系统HDFS

- HDFS的基本特征
- HDFS基本构架
 - NameNode的作用； DataNode的作用
- HDFS数据分布设计
- HDFS可靠性与出错恢复
 - NameNode出错； DataNode出错； 数据出错
- HDFS的安装和启动
- HDFS文件系统操作命令





03-2 Hadoop MapReduce基本架构

2. Hadoop MapReduce的基本工作原理

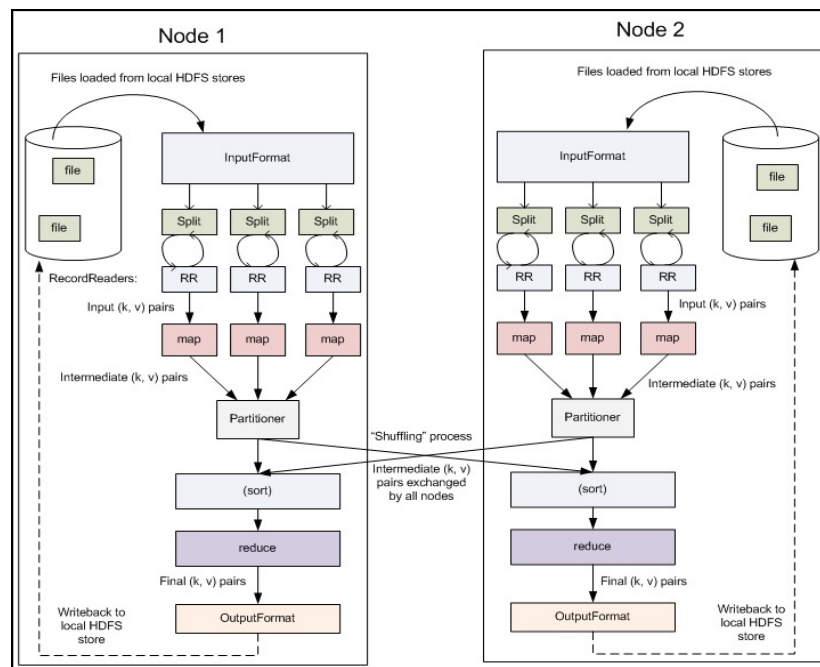
■ Hadoop MapReduce基本构架与工作过程

- 经典版：JobTracker的作用； TaskTracker的作用
- YARN：ResourceManager的作用； NodeManager的作用； ApplicationMaster的作用
- MapReduce作业执行过程

■ Hadoop MapReduce主要组件

- 文件输入格式InputFormat
- 输入数据分块InputSplits
- 数据记录读入RecordReader
- **Mapper**：多少个mapper？
- **Combiner**
- **Partitioner**
- **Reducer**：多少个reducer？
- 文件输出格式OutputFormat

■ 容错处理与计算性能优化





03-2 Hadoop MapReduce基本架构

■ 程序执行时的容错处理与计算性能优化

- 由Hadoop系统自己解决
- 主要方法是将失败的任务进行再次执行
- TaskTracker会把状态信息汇报给JobTracker，最终由JobTracker决定重新执行哪一个任务
- 为了加快执行的速度，Hadoop也会自动重复执行同一个任务，以最先执行成功的为准（投机执行）
- `mapreduce.map.speculative`：是否为Map Task打开推测执行机制，默认为false
- `mapreduce.reduce.speculative`：是否为Reduce Task打开推测执行机制，默认为false



04 Hadoop安装运行与程序开发

1. Hadoop安装方式

- 单机方式
- 单机伪分布方式
- 集群分布模式

2. 单机Hadoop系统安装基本步骤

3. 集群Hadoop系统安装基本步骤

4. Hadoop集群远程作业提交与执行

- 程序开发与提交作业基本过程
- 集群分布方式下远程提交作业

5. Hadoop MapReduce程序开发



04 Hadoop 安装运行与程序开发

6. Hadoop 分布式文件系统HDFS编程

- **FileSystem**基类
- 创建文件
- 打开文件
- 获取文件信息
- 获取目录信息
- 文件读取
- 文件写入
- 关闭
- 删除



05 MapReduce基础编程

1. MapReduce可解决哪些算法问题？

- 基本算法
- 复杂算法或应用

2. 回顾： **MapReduce**流水线

3. MapReduce排序算法和二级排序

4. MapReduce单词同现分析算法

5. **MapReduce**文档倒排索引算法

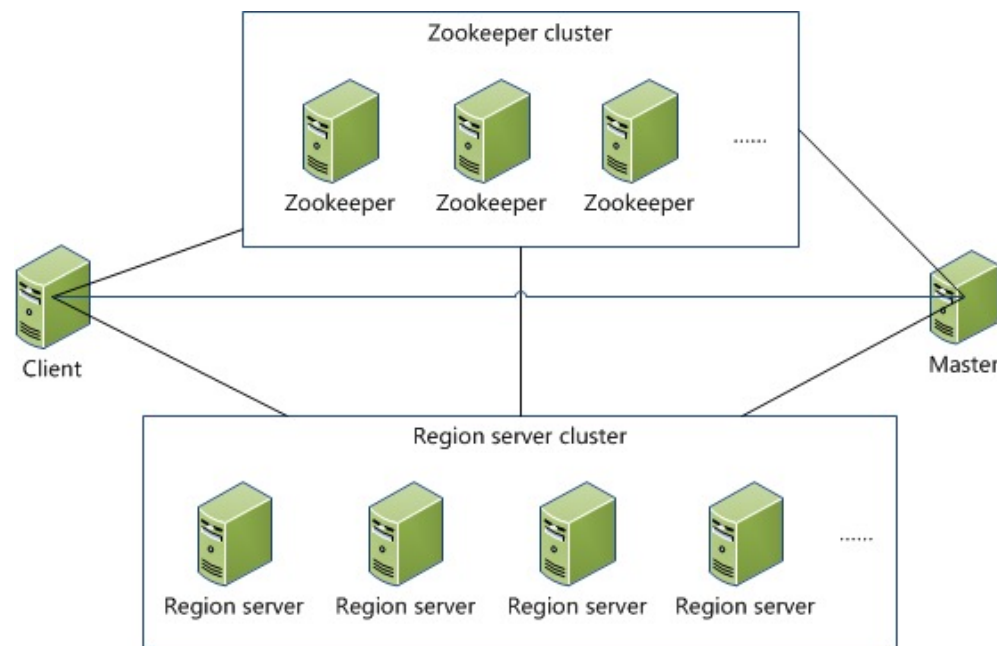
6. MapReduce专利文献数据分析



06-1 HBase基本原理与程序设计

1. HBase基本原理

- HBase的设计目标和功能特点
- HBase数据模型
- HBase的基本构架
- HBase的数据存储和管理





06-1 HBase基本原理与程序设计

2. HBase基本操作

- HBase shell 操作
 - 创建表格与列举表格
 - 插入数据
 - 描述表信息
 - 输入数据与扫描数据
 - 限制列进行扫描
 - HBase中的disable和enable



06-1 HBase基本原理与程序设计

3. HBase编程方法示例

■ HBase的Java编程

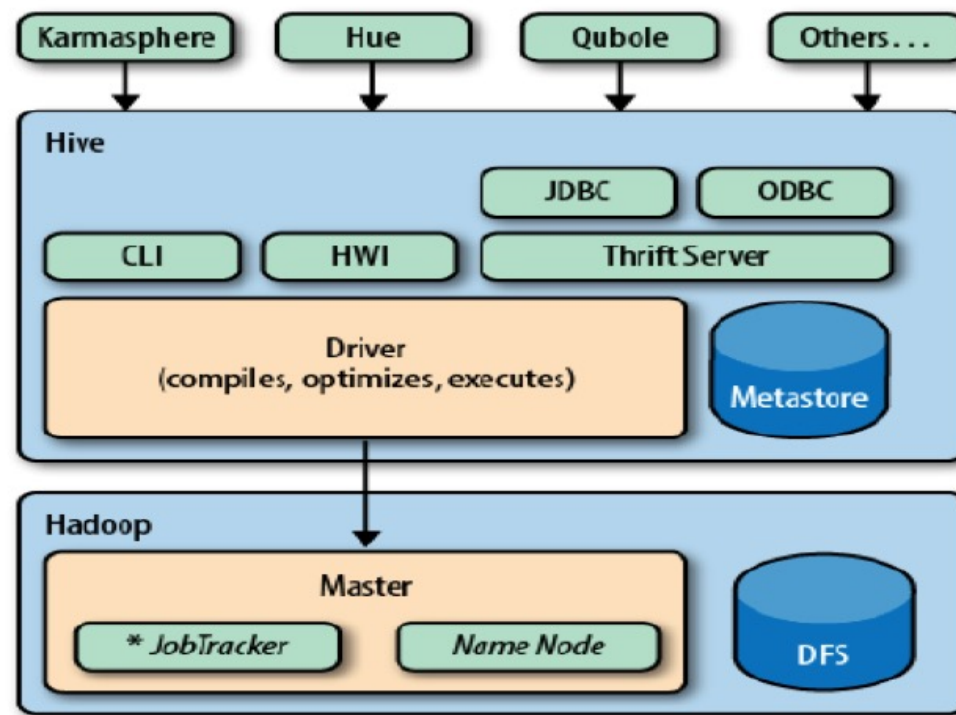
- 创建表
- 删除表
- 查询数据
- 插入数据
- 删除数据
- 切分表



06-2 Hive简介

1. Hive基本工作原理

- 在Hadoop上用SQL进行数据分析
- Hive的组成模块
- Hive的系统结构
- Hive的数据模型
- 元数据存储：Metastore
- 数据的物理分布情况
 - 表，分区，分桶
- Hive系统的配置





06-2 Hive简介

2. Hive基本操作示例

- 启动Hive的命令行界面shell
- 创建数据表
- 装入数据
- SELECTS and FILTERS
- Group By
- Join
- 分区 vs 分桶



07 MapReduce 高级编程

1. 复合键值对的使用

- 用复合键让系统完成排序
- 把小的键值对合并成大的键值对

2. 用户自定义数据类型

- Hadoop 内置的数据类型
- 用户自定义数据类型
 - 需要实现 **Writable** 接口
 - 作为 **key** 或者需要比较大小时则需要实现 **WritableComparable** 接口



07 MapReduce 高级编程

3. 用户自定义输入输出格式

- Hadoop 内置的文件输入格式
 - **TextInputFormat**
 - **KeyValueTextInputFormat**
- Hadoop 内置的 **RecordReader**
 - **LineRecordReader**
 - **KeyValueLineRecordReader**



07 MapReduce 高级编程

3. 用户自定义输入输出格式

- 用户自定义 **InputFormat** 和 **RecordReader** 的方法
 - *NewInputFormat* extends `FileInputFormat<Text, Text>`
 - *NewRecordReader* extends `RecordReader<Text, Text>`
 - `job.setInputFormatClass(NewInputFormat.class)`
- Hadoop 内置的文件输出格式
 - **TextOutputFormat**
 - **KeyValueTextOutputFormat**
- Hadoop 内置的 **RecordWriter**
 - **LineRecordWriter**
- 用户自定义 **OutputFormat** 和 **RecordWriter** 的方法
 - *NewOutputFormat* extends `FileOutputFormat<Text, Text>`
 - *NewRecordWriter* extends `RecordWriter<Text, Text>`
 - `job.setOutputFormatClass(NewOutputFormat.class)`



07 MapReduce 高级编程

4. 用户自定义 **Partitioner** 和 **Combiner**

■ 定制 **Partitioner**

```
class NewPartitioner extends HashPartitioner<K,V>
{ // override the method
    getPartition(K key, V value, int numReduceTasks)
    { .....
    }
}
```

Job. setPartitionerClass(NewPartitioner)

■ 定制 **Combiner**

```
public static class NewCombiner extends Reducer < Text, IntWritable, Text, IntWritable >
{ // 实现reduce方法
    public void reduce(Text key, Iterable<IntWritable> values, Context context)
        throws IOException, InterruptedException
    { ...
    }
}
```

Job. setCombinerClass(NewCombiner)

特别注意: **Combiner** 输出时不能改变其输入键值对的格式, 必须保持一致!

思考题: 请同学们解释为什么要有这个要求? 如果不转手这个要求会发生什么问题?



07 MapReduce 高级编程

5. 多数据源的连接

■ 用 `DataJoin` 类实现 `Reduce` 端 `Join`

- 连接主键 `GroupKey` 与 记录标签 `Tag`
- `DataJoinMapperBase`
- `DataJoinReducerBase`
- `TaggedMapOutput`

▣ `Mapper` 需要实现的抽象方法

- `abstract Text generateInputTag(String inputFile)`
- `abstract TaggedMapOutput generateTaggedMapOutput(Object value)`



07 MapReduce 高级编程

5. 多数据源的连接

■ 用文件复制实现Map端Join

用distributed cache将一个或多个小数据量文件分布复制到所有节点上

■ Job类中:

`public void addCacheFile(URI uri)` //将一个文件放到distributed cache file中

■ Mapper或Reducer的context类中:

`public Path[] getLocalCacheFiles()` //获取设置在distributed cache files中的文件路径

■ Replicated Joins方法的变化使用

可以参考Word Count 2.0源代码

■ 带Map端过滤的Reduce端Join

■ 多数据源连接解决方法的限制

- 以上的多数据源Join只能是具有相同主键/外键的数据源间的连接，如果数据源两两之间具有多个不同的主键/外键的连接，则需要使用多次MapReduce过程完成不同主/外键间的连接



07 MapReduce 高级编程

6. 全局参数/数据文件的传递

- 全局作业参数的传递
 - Configuration类专门提供以下用于保存和获取属性的方法
 - mapper/reducer类初始化方法setup()从configuration对象中读出属性
- 全局数据文件的传递(见上distributed cache的使用)

可以参考Word Count 2.0源代码



07 MapReduce 高级编程

7. 其它处理技术

- 查询任务相关信息
- 划分多个输出文件集合
- 输入输出到关系数据库
- 从数据库中输入数据
 - DBInputFormat: 提供从数据库读取数据的格式
 - DBRecordReader: 提供读取数据记录的接口
- 向数据库中输出计算结果
 - DBOutputFormat: 提供向数据库输出数据的格式
 - DBRecordWriter: 提供向数据库写入数据记录的接口
 - DBConfiguration 提供数据库配置和创建连接的接口
- 创建数据库连接
 - `public static void configured (Job job, String driverClass, String dbUrl, String userName, String passwd)`
 - 指定写入的数据表和字段:
 - `public static void setOutput (Job job, String tableName, String... fieldNames)`



08 基于MapReduce的搜索引擎算法

网页排名图算法PageRank

- PageRank的基本设计思想和设计原则
- PageRank的简化模型及其缺陷
 - Rank leak和Rank sink
- PageRank的随机浏览模型
- 随机浏览模型的PageRank公式
- 用MapReduce实现PageRank
 - Phase1: GraphBuilder
 - Phase2: PageRankIter
 - Phase3: Rankviewer
 - PageRank迭代终止条件

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$



09 MapReduce 数据挖掘基础算法

1. 基于MapReduce的K-Means并行化算法

- 数据挖掘并行算法研究的重要性
- K-Means 聚类算法
- 基于MapReduce的K-Means并行算法设计
- 聚类算法应用实例



09 MapReduce 数据挖掘基础算法

2. 基于MapReduce的分类并行化算法

- 分类问题的基本描述
- **K-最近邻分类并行化算法**
- 朴素贝叶斯分类并行化算法

3. 基于MapReduce的频繁项集挖掘算法

- 频繁项集挖掘基本问题
- 现有算法概述：**Apriori**算法，子集求取算法，**SON**算法
- **PSON**：基于MapReduce的并行算法



10-1 Spark 系统简介

- 为什么会有Spark?
- Spark的基本构架和组件

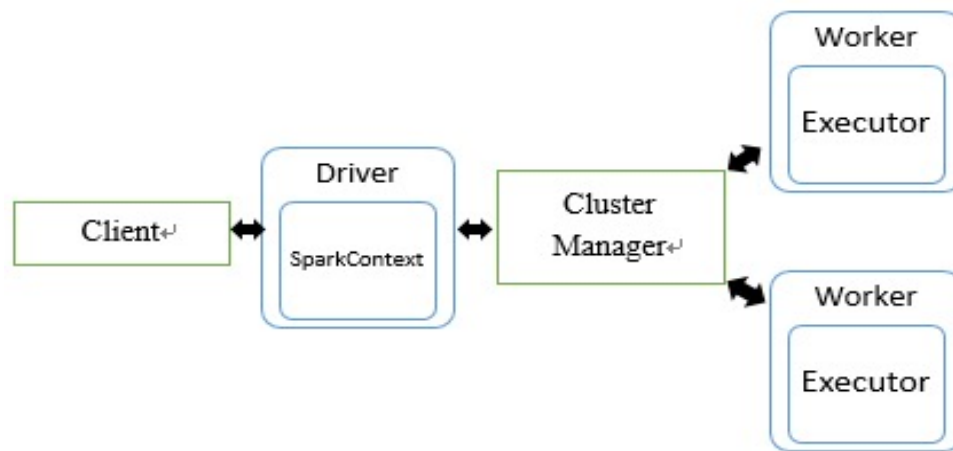
- Master node + Driver
- Worker node + Executors

- Spark的程序执行过程

- Application (Driver + Executor), Job, Stage, Task

- Spark的技术特点

- 基于内存计算的弹性分布式数据集(RDD)
- 灵活的计算流图(DAG)
- 覆盖多种计算模式 (查询分析, 批处理, 流式计算, 迭代计算, 图计算, 内存计算)





10-2 Spark编程

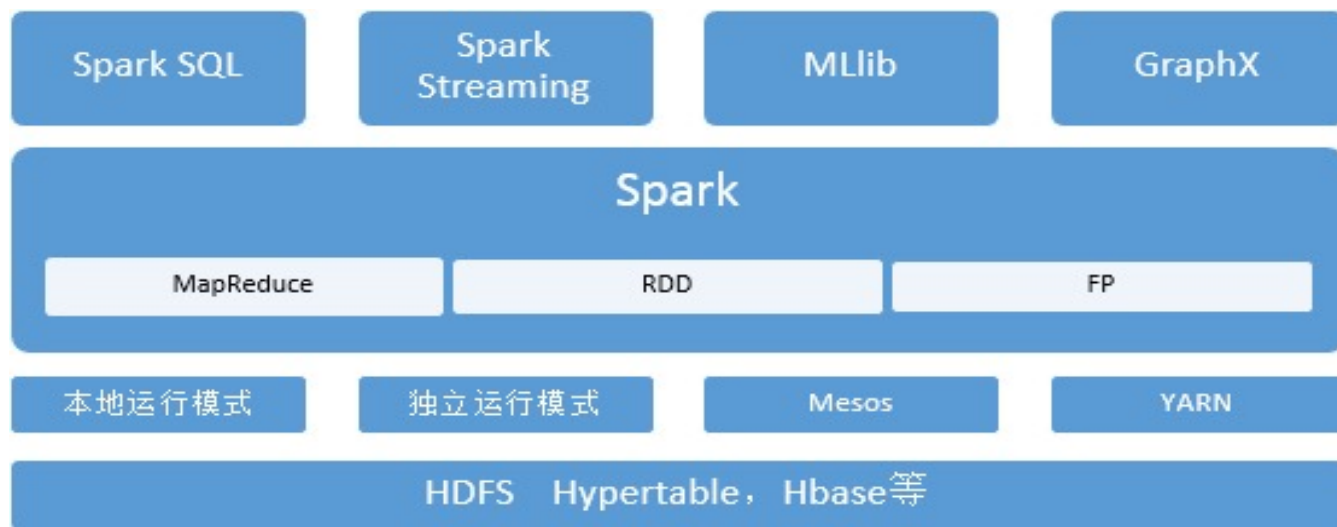
- Spark编程模型与编程接口
 - RDD 操作: Transformation & Action
 - RDD的容错, 宽依赖/窄依赖, 持久化
 - Scala, Java, Python
- Spark编程示例
 - WordCount
 - 二次排序
 - KMeans
- Spark的安装运行模式
 - Standalone, Yarn, Mesos, Kubernetes



10-3 Spark上层组件

Spark环境中其它功能组件简介

- Spark SQL
 - RDD vs. DataFrame vs. Dataset
- Spark Streaming
 - DStream
- MLlib
- GraphX
 - Resilient Distributed Property Graph
 - Table视图和Graph视图





考核方式

- 课程实验：4次，共占**25%**
- 课程设计：1项，占**25%**
- 期末考试：笔试，闭卷，占**50%**

THANK YOU



南京大學
NANJING UNIVERSITY

南京大学计算机软件研究所
Institute of Computer Software, Nanjing University