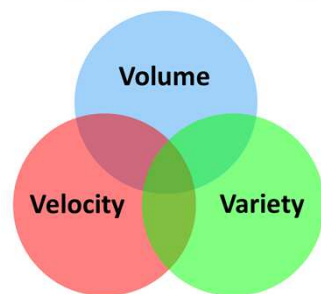


大数据运算系统 (1)



陈世敏


中科院计算所
计算机体系结构
国家重点实验室

©2015-2018 陈世敏

Outline

- MapReduce/Hadoop
 - 编程模型
 - 系统实现
 - 典型算法
- Microsoft Dryad

MapReduce/Hadoop简介

- MapReduce是目前云计算中最广泛使用的计算模型
 - 由Google于2004年提出
 - “MapReduce: Simplified Data Processing on Large Clusters”. Jeffrey Dean and Sanjay Ghemawat (Google). **OSDI 2004**.
- Hadoop是MapReduce的一个开源实现 
 - 2005年由Doug Cutting and Mike Cafarella 开始了Hadoop项目
 - 2006年Cutting成为Yahoo Lab的员工
 - Hadoop的开发主要由Yahoo Lab推动，后来成为Apache开源项目
 - 基于Java
 - 已经被广泛使用：Yahoo, Facebook, Twitter, Linkedin, Ebay, AOL, Hulu, 百度, 腾讯, 阿里, 天涯社区,

MapReduce编程模型

- 整体思路
- 数据模型
- Map-shuffle-Reduce
- Word count举例
- 与SQL Select语句的关系

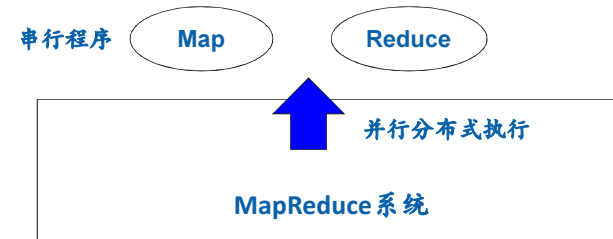
整体思路

- 并行分布式程序设计不容易
 - Multi-threading
 - Socket programming
 - Data distribution
 - Job distribution, coordination, load balancing
 - Fault tolerance
 - Debugging
- 需要有经验的程序员+编程调试时间
 - 上面每个方面都需要学习和经验积累
 - 调试分布式系统很时间和精力

整体思路

- 解决思路
 - 程序员写串行程序
 - 由系统完成并行分布式地执行

在本课中，程序员是指使用大数据平台实现上层应用功能的人员



整体思路

- 解决思路
 - 程序员写串行程序
 - 由系统完成并行分布式地执行
- 程序员保证串行程序的正确性
 - 编程序时不需要思考并行的问题
 - 调试时只需要保证串行执行正确
- 系统负责并行分布执行的正确性和效率
 - Multi-threading, Socket programming, Data distribution, Job distribution, coordination, load balancing, Fault tolerance
- 有什么问题吗？

整体思路

- 有什么问题吗？
- 牺牲了程序的功能！
 - 直接进行并行分布式编程，可以完成各种各样丰富的功能
 - 而一个编程模型实际上是限定了程序的功能类型
- 推论1：系统的编程模型必须有代表性
 - 必须代表一大类重要的应用才有生命力
- 推论2：一个成功的模型也就不可避免地被人们应用于原本不适合的情形
 - 需要扩展，需要新的模型
 - 这是后话

MapReduce的数据模型

- <key, value>

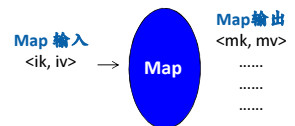
- 数据由一条一条的记录组成
- 记录之间是无序的
- 每一条记录有一个key, 和一个value
- key: 可以不唯一
- key与value的具体类型和内部结构由程序员决定, 系统基本上把它们看作黑匣

MapReduce

$\text{Map}(ik, iv) \rightarrow \{<mk, mv>\}$

$\text{Reduce}(mk, \{mv\}) \rightarrow \{<ok, ov>\}$

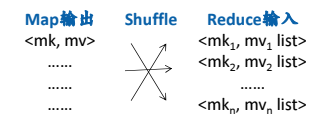
Map函数



$\text{Map}(ik, iv) \rightarrow \{<mk, mv>\}$

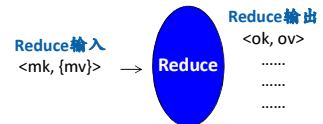
- 输入是一个key-value记录: <ik, iv>
 - 我们用'v'代表input
- 输出是0~多个key-value记录: <mk, mv>
 - 我们用'm'代表intermediate
- 注意: mk与ik很可能完全不同

Shuffle (由系统完成)



- Shuffle = group by mk
- 对于所有的map函数的输出, 进行group by
- 将相同mk的所有mv都一起提供给Reduce

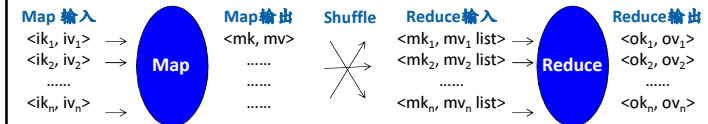
Reduce函数



$\text{Reduce}(\text{mk}, \{\text{mv}\}) \rightarrow \{\text{ok}, \text{ov}\}$

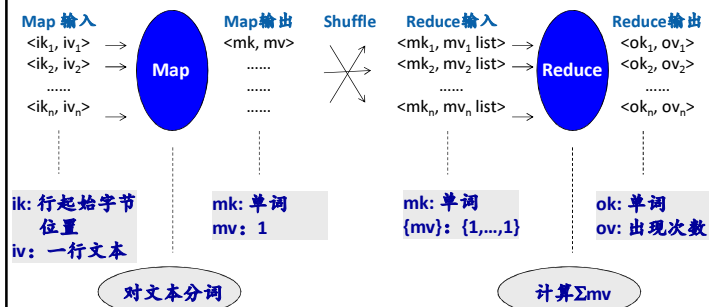
- 输入是一个mk和与之对应的所有mv
- 输出是0~多个key-value记录: <ok, ov>
 - 我们用'o'代表output
- 注意: ok与mk可能不同

Map-shuffle-Reduce



- 程序员编制串行的Map函数和Reduce函数
- 系统完成shuffle功能
 - shuffle = group by mk

Map-shuffle-Reduce: word count 举例



Word count: 统计文本中每个单词出现的次数

Hadoop Mapper.java (简化了exception handling)

```
public class Mapper<KEYIN, VALUEIN, KEYOUT, VALUEOUT> {
    protected void setup(Context context) {
        // NOTHING
    }
    protected void map(KEYIN key, VALUEIN value, Context context){
        context.write((KEYOUT) key, (VALUEOUT) value);
    }
    protected void cleanup(Context context) {
        // NOTHING
    }
    public void run(Context context) { //Hadoop调用run
        setup(context);
        while (context.nextKeyValue()) {
            map(context.getCurrentKey(), context.getCurrentValue(), context);
        }
        cleanup(context);
    }
}
```

Hadoop Reducer.java (简化了exception handling)

```
public class Reducer<KEYIN, VALUEIN, KEYOUT, VALUEOUT> {
    protected void setup(Context context) {
        // NOTHING
    }
    protected void reduce(
        KEYIN key, Iterable<VALUEIN> vlist, Context context) {
        for(VALUEIN v: vlist) context.write((KEYOUT)key, (VALUEOUT)v);
    }
    protected void cleanup(Context context) {
        // NOTHING
    }
    public void run(Context context) { //Hadoop调用run
        setup(context);
        while (context.nextKey()) {
            reduce(context.getCurrentKey(), context.getValues(), context);
        }
        cleanup(context);
    }
}
```

WordCount.java

```
public class WordCount {
    public static class TokenizerMapper
        extends Mapper<Object, Text, Text, IntWritable>{

        private final static IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(Object key, Text value, Context context
            ) throws IOException, InterruptedException {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens()) {
                word.set(itr.nextToken());
                context.write(word, one);
            }
        }
    }
}
```

WordCount.java

```
public static class IntSumReducer
    extends Reducer<Text, IntWritable, Text, IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
        Context context
        ) throws IOException, InterruptedException {

        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        result.set(sum);
        context.write(key, result);
    }
}
```

WordCount.java

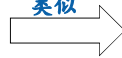
```
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf,
        args).getRemainingArgs();
    if (otherArgs.length != 2) {
        System.err.println("Usage: wordcount <in> <out>");
        System.exit(2);
    }
    Job job = new Job(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
    FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}
```

比较

MapReduce

- Map
- Shuffle
- Reduce
- 选择的功能更加丰富
 - 程序实现的
 - 类似最简单的SQL select
 - 但不支持join

类似

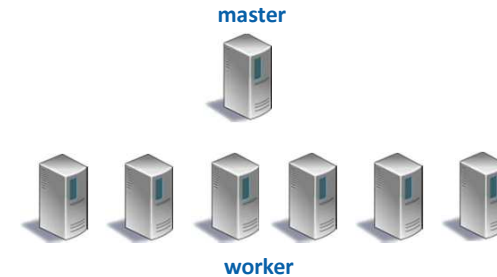


SQL Select

- Selection/projection
- Group by
- Aggregation, Having
- 功能由数据类型和SQL语言标准定义
 - 有UDF: user defined function
 - 但支持得不好

21

MapReduce系统架构



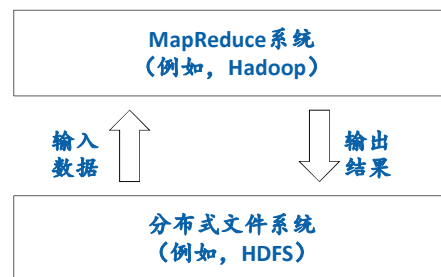
在OSDI'04文章中，基本上是1个master对应100~1000数量级的workers

大数据系统与大规模数据分析

22

©2015-2018 陈世敏(chensm@ict.ac.cn)

系统架构



大数据系统与大规模数据分析

23

©2015-2018 陈世敏(chensm@ict.ac.cn)

MapReduce / Hadoop系统架构



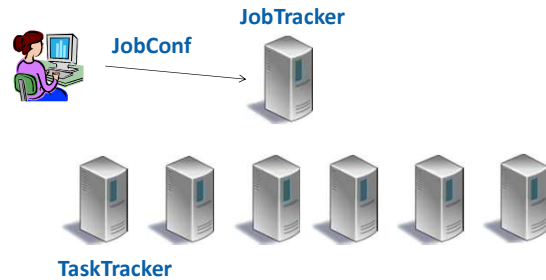
- 注意: JobTracker, TaskTracker, Name Node, Data Node都是进程, 所以可以在一台机器上同时运行JobTracker/Name Node, TaskTracker/Data Node
- Hadoop 2.x采用YARN代替了JobTracker, 但功能大同小异

大数据系统与大规模数据分析

24

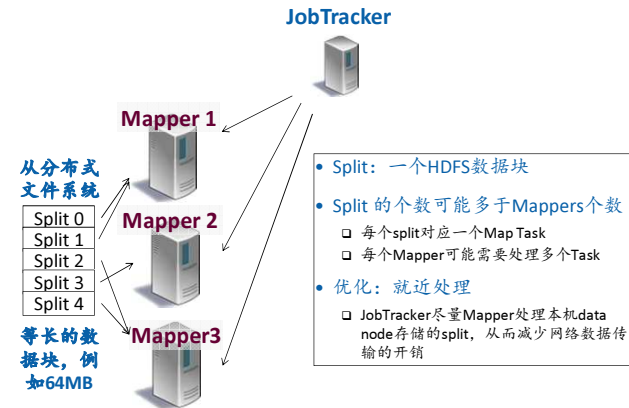
©2015-2018 陈世敏(chensm@ict.ac.cn)

MR运行：提交作业

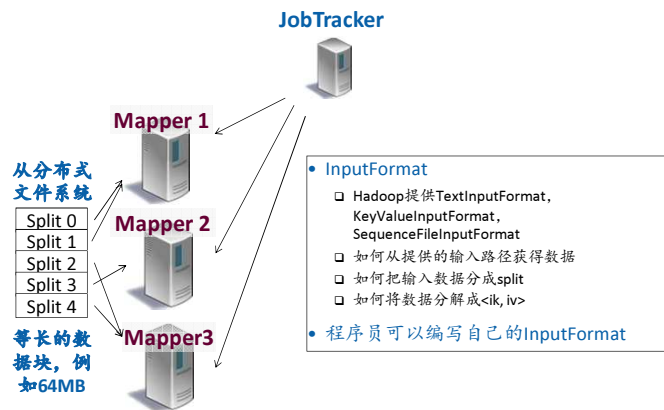


- 包括Map函数、Reduce函数(Jar)、配置信息(例如，几个Mappers，几个Reducers)、输入路径、输出路径等

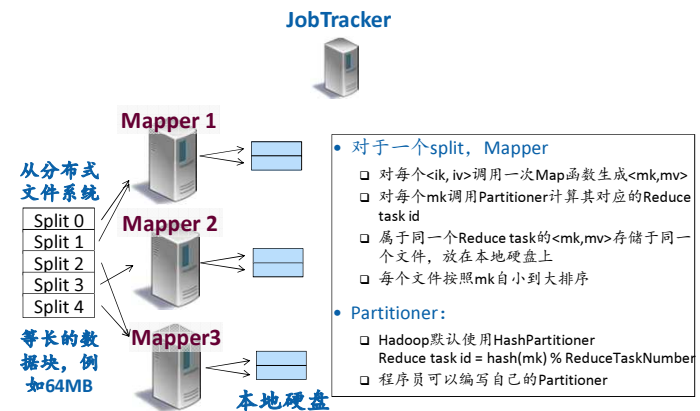
MR运行：Map Task 读数据



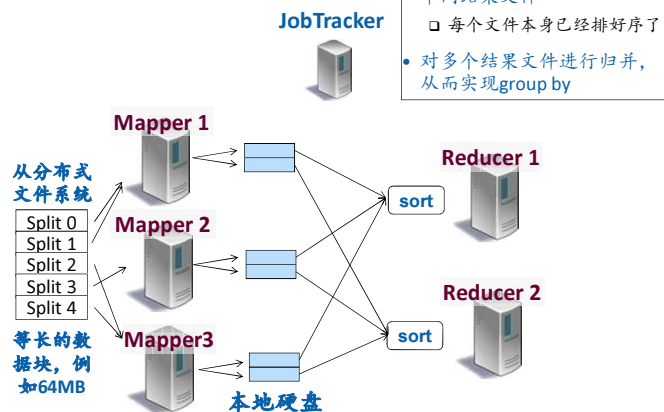
MR运行：Map Task 读数据



MR运行：Map Task 执行



MR运行: Shuffle

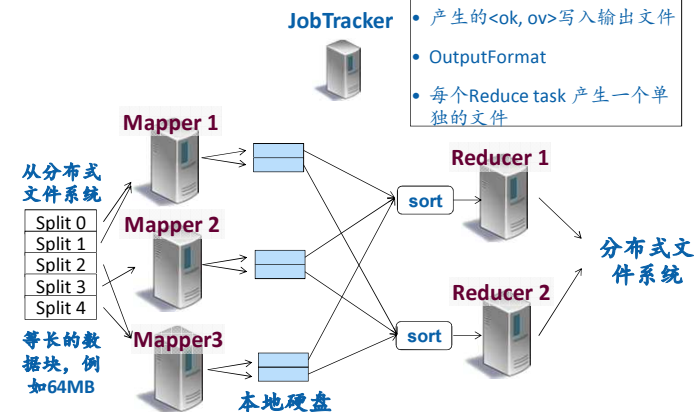


大数据系统与大规模数据分析

29

©2015-2018 陈世敏(chensm@ict.ac.cn)

MR运行: Reduce



大数据系统与大规模数据分析

30

©2015-2018 陈世敏(chensm@ict.ac.cn)

Combiner

- Combiner = partial reducer
 - Combiner(mk, {mv}) \rightarrow <mk, mv>
- 回顾word count
 - 传输很多<mk, 1>似乎有些浪费
 - 如果在Mapper一侧, 一个mk出现多次, 完全可以进行本地的累计, 这样只需要传输一个<mk, 本地次数>
 - 这可以用Combiner实现

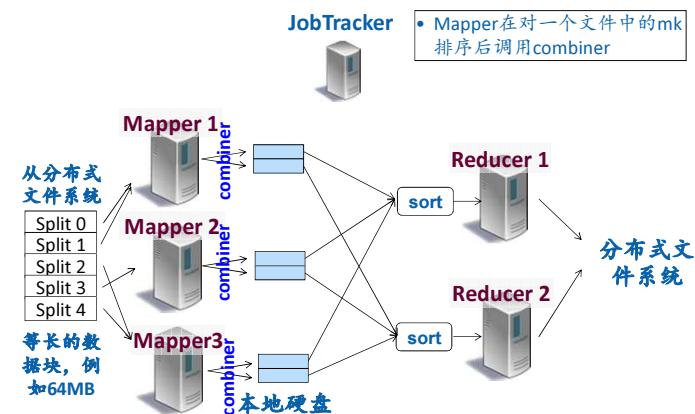


大数据系统与大规模数据分析

31

©2015-2018 陈世敏(chensm@ict.ac.cn)

MR运行: Combiner

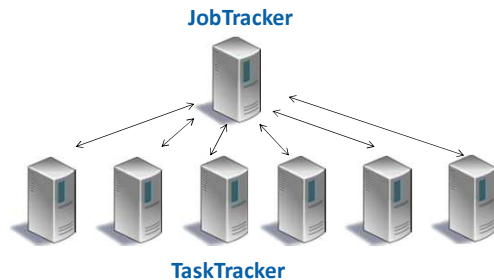


大数据系统与大规模数据分析

32

©2015-2018 陈世敏(chensm@ict.ac.cn)

MR: Fault Tolerance (容错)



- HeartBeat(心跳)消息
 - 定期发送, 向JobTracker汇报进度
- JobTracker可以及时发现不响应的机器或速度非常慢的机器
 - 这些异常机器被称作Stragglers

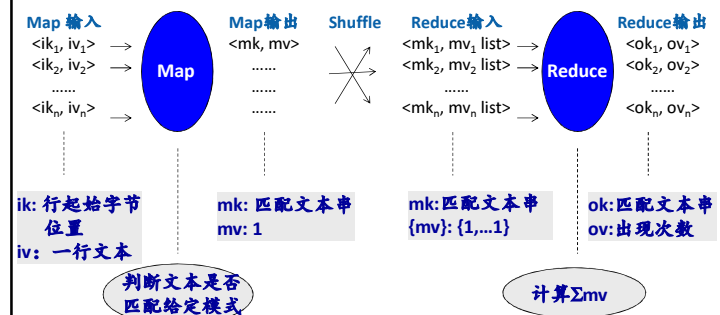
MR: Fault Tolerance (容错)

- 一旦发现Straggler
 - JobTracker就将它需要做的工作分配给另一个worker
- Straggler是Mapper, 将所对应的splits分配给其它的Mapper
 - 输入数据是分布式文件, 所以不需要特殊处理
 - 通知所有的Reducer这些splits的新对应Mapper
 - Shuffle时从新对应的Mapper传输数据
- Straggler是Reducer, 在另一个TaskTracker执行这个Reducer
 - 这个Reducer需要重新从Mappers传输数据
 - 注意: 因为Mapper的输出是在本地文件中的, 所以可以多次传输

典型算法

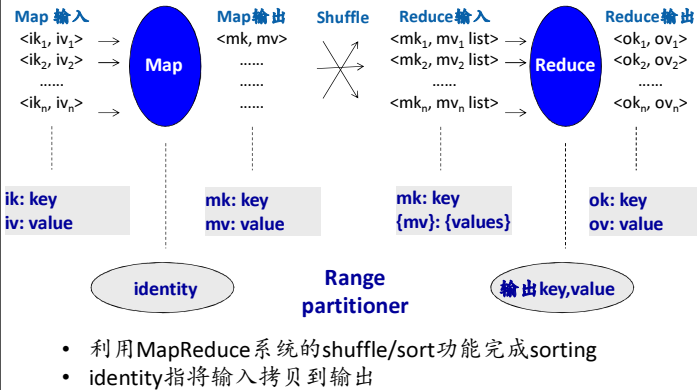
- Grep
- Sorting
- Join

举例: Grep (找到符合特定模式的文本)



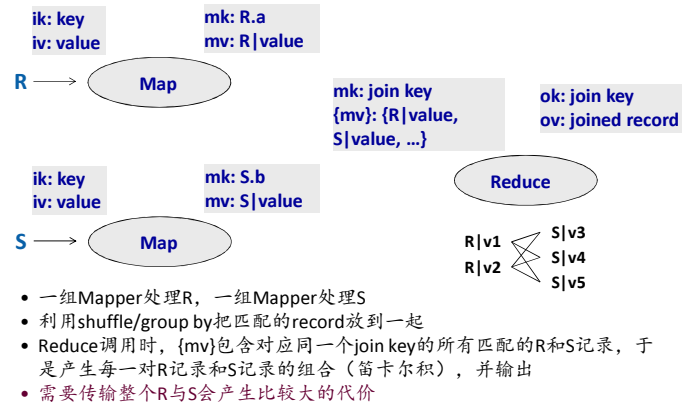
与word count类似

举例：Sorting



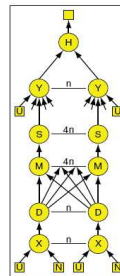
举例：Equi-Join

$$R \bowtie_{R.a = S.b} S$$



Microsoft Dryad

- Dryad是对MapReduce模型的一种扩展
 - 组成单元不仅是Map和Reduce，可以是多种节点
 - 节点之间形成一个有向无环图DAG(Directed Acyclic Graph)，以表达所需要的计算
 - 节点之间的数据传输模式更加多样
 - 可以是类似Map/Reduce中的shuffle
 - 也可以是直接1:1、1:多、多:1传输
 - 比MapReduce更加灵活，但也更复杂
 - 需要程序员规定计算的DAG



Dryad paper
[Eurosys'07]

- Microsoft内部云计算系统Cosmos基于Dryad

小结

- MapReduce/Hadoop
 - 编程模型
 - 系统实现
 - 典型算法
- Microsoft Dryad