MapReduce

Allen Wang (allen.g.wang@newegg.com)

2011-11-02

ONCE YOU KNOW, YOU NEWEGG.

Overview

- MapReduce基本概念
- MapReduce处理与数据流
- MapReduce是如何工作的?
- MapReduce应用开发
- MapReduce监控

MapReduce基本概念

- 什么是MapReduce?
  - MapReduce是一种用于数据处理的编程模型, 主要用于大规模数据集（大于1TB）的分布式并行运算。源自Google在2004发表的一篇文章 MapReduce: Simplified Data Processing on Large Clusters
  - MapReduce是一种通过多个结点并行完成一个任务的方法。
  - 每个结点仅处理存储在本结点上的数据。
  - 包含两个阶段
    - Map(映射)
    - Reduce(化简)

MapReduce基本概念

- 什么是Hadoop MapReduce?
  - 是基于MapReduce编程模型的一种实现。
  - 极大地方便了编程人员在不会分布式并行编程的情况下,将自己的程序运行在分布式系统上。

- Hadoop MapReduce的功能特性
  - Hadoop是基于Java的开源项目
  - MapReduce可用Java/Python...多种语言来编写。
  - 实现Map与Reduce函数非常简单。

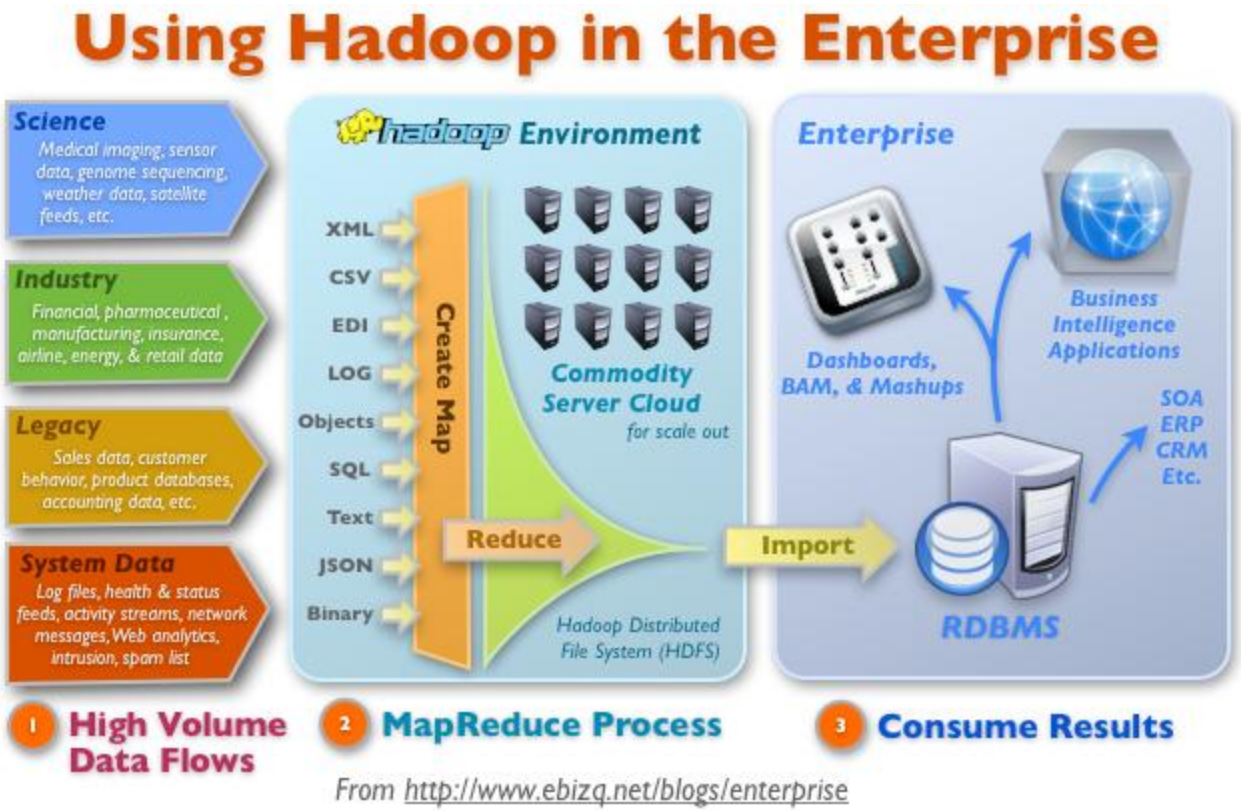
    map和reduce函数遵循如下常规格式

    map: (K1, V1) -> list(K2, V2)

    reduce: (K2, list(V2)) -> list(K3, V3)

MapReduce基本概念

- MapReduce的应用

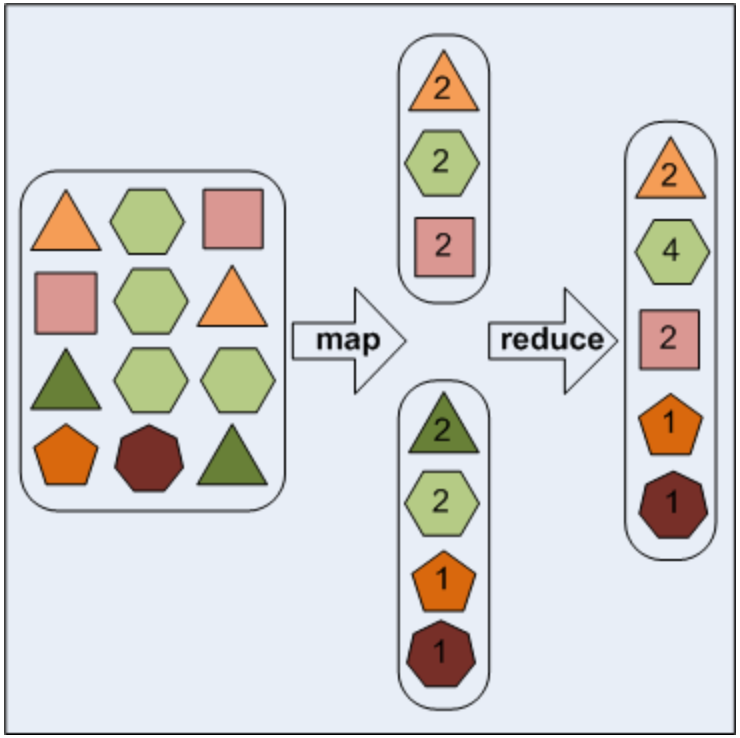


Overview

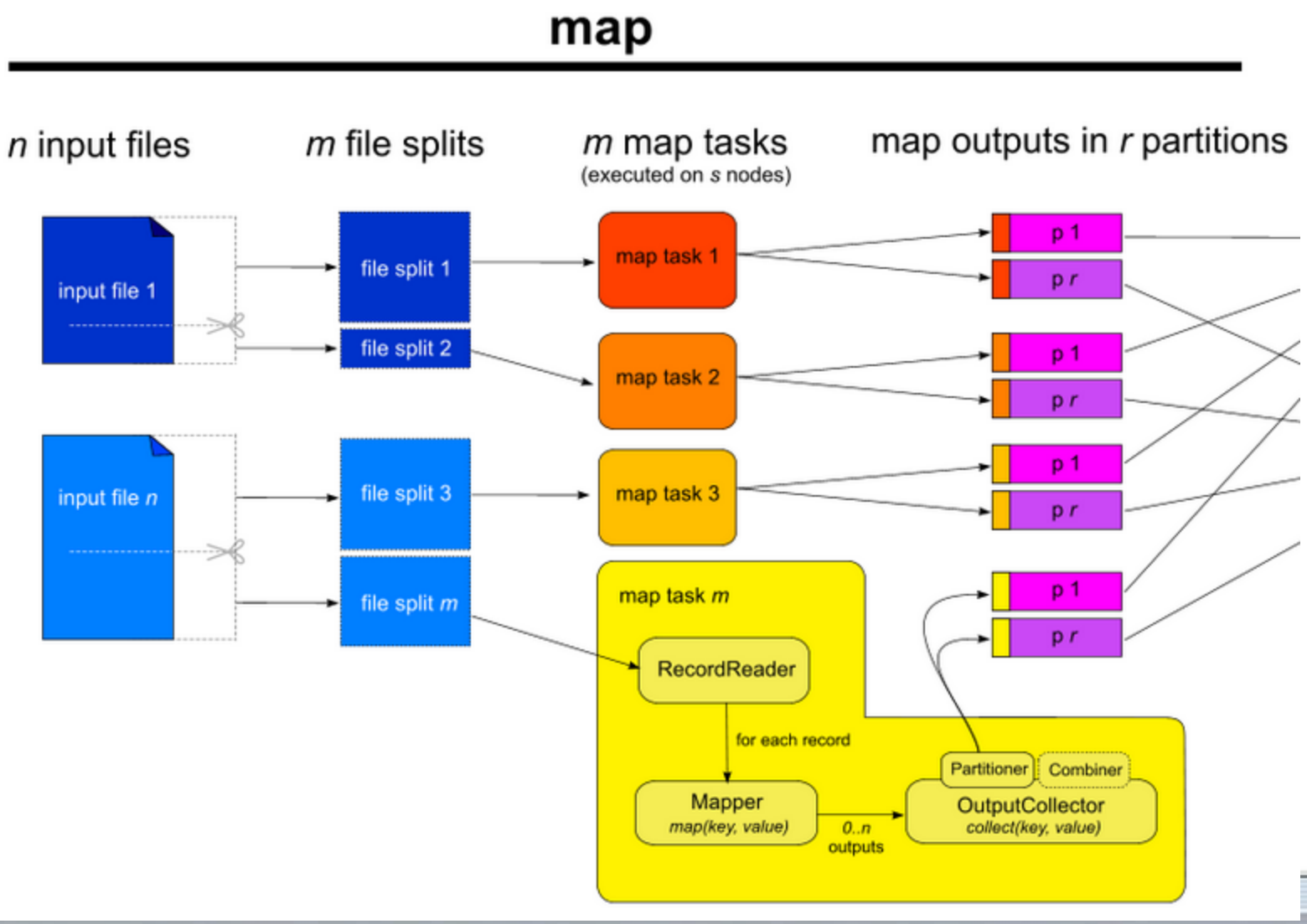
- MapReduce基本概念
- **MapReduce处理与数据流**
- MapReduce是如何工作的?
- MapReduce应用开发
- MapReduce监控

MapReduce处理与数据流

- 统计相同形状相同颜色图形的个数。



MapReduce处理与数据流



MapReduce处理与数据流

- Input

  存储在HDFS中文件，文件按block的拆分方式进行存储。默认一个block的大小为64Mb/128Mb。

- InputSplit

  Hadoop为每个分片构建一个map任务，为易于管理, 分片的大小趋向于HDFS中一个块的大小。

  RecordReader通过FileSplit从相关的InputFile中读取所有的record。

MapReduce处理与数据流

- Map

  通过RecordReader读取的每一条Record(key-value pair)都会调用map函数。

  key-value输入的格式由InputFormat来进行控制，默认是TextInputFormat。

  - TextInputFormat: 每条记录是一行输入。

    Key: 存储该行在整个文件中的字节偏移量。

    Value: 这行的内容, 不包括任何终止符(回车符和换行符)。

    Sample: 每个分片2条记录

    On the top of the Crumpetty Tree

    The Quangle Wangle stat,

    FileSplit会把上面的分片转化成下面的键值对:

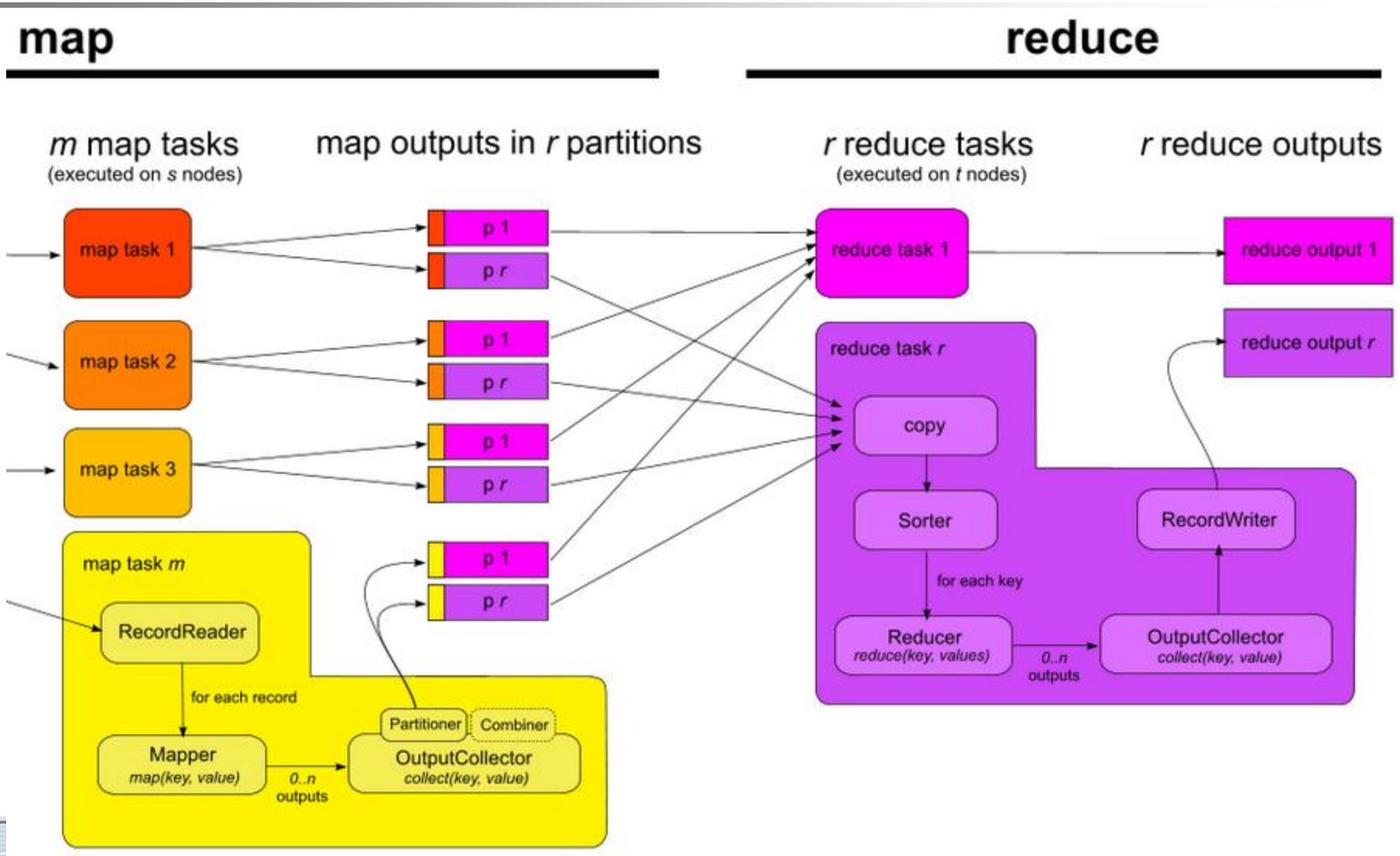
    (0, On the top of the Crumpetty Tree)

    (33, The Quangle Wangle stat,)

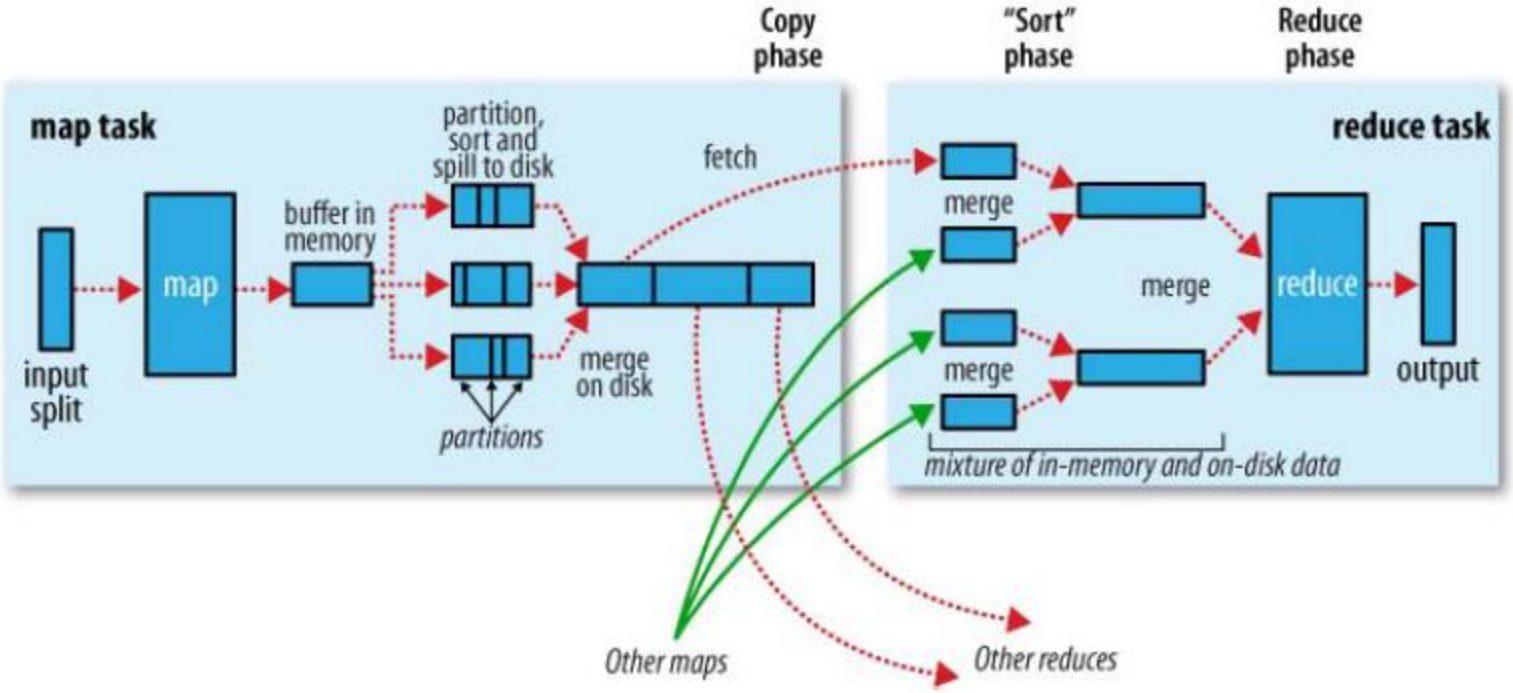
  - KeyValueInputFormat: 每条记录是一个键值对，以第一个\tab符作为分隔。
  - Sequence FileInputFormat: 用于处理二进制格式的数据。可自定义键值对规则。

NESC

MapReduce处理与数据流



MapReduce处理与数据流



MapReduce处理与数据流

- Shuffle
  - shuffle是从map产生输出到reduce的消化输入的整个过程。

- Partition & Sort
  - 基于Reducer的个数r(可配置)划分同样多的分区Partition。相同序号的Partition中数据会Copy到相同序号的Reducer上去。默认的分区算法是基于HashPartition，从map的outputs中的数据键会根据HashPartition算法被分配到不同的Partition中。
  - 假设键值是整型，Hash算法可以是整除取余算法，有3个分区，mod(0/3)=0 mod(3/3)=0 键值0和3会被分配到0分区。
  - 在每个分区中，后台线程按键进行内排序。

MapReduce处理与数据流

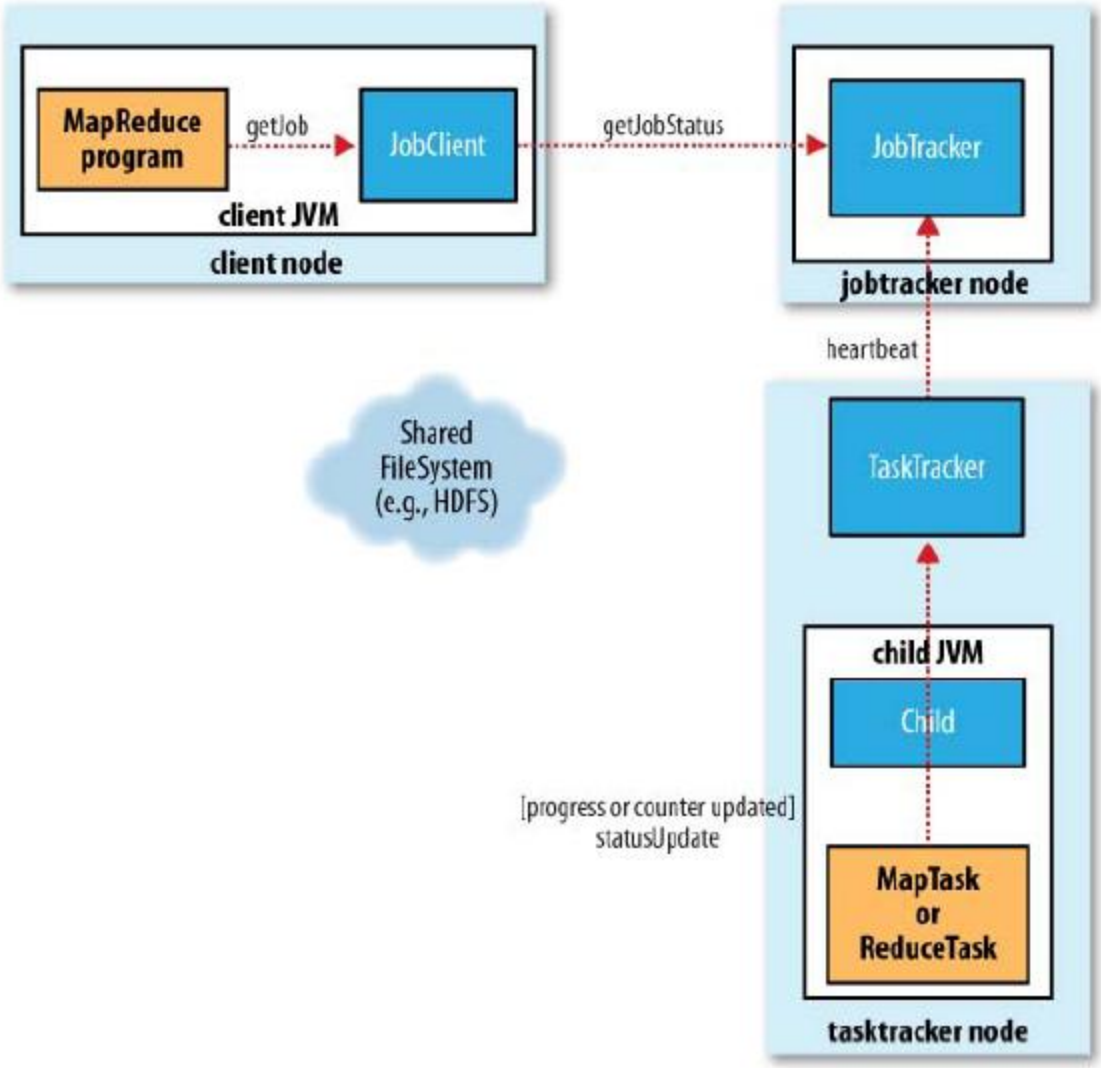
- Combine
  - 如果有一个Combiner， 在写入磁盘之前，它会在Partition & Sort后的输出上运行。
  - Combine可以运行多次，原则上最终Reduce的结果不能改变。
  - 通常Combine的算法与Reduce算法相同。

- Spill to disk
  - 每个map任务都有一个内存缓冲区，用于存储任务的输出。默认情况下，缓冲的大小为100MB, 一旦缓冲内容达到阈值(默认是80%), 一个后台线程便开始把内容写到(spill)磁盘中。在写磁盘过程中，map输出继续被写到缓冲区，但如果在此期间缓冲区被填满，map会阻塞直到写磁盘过程完成。

Overview

- MapReduce基本概念
- MapReduce处理与数据流
- **MapReduce是如何工作的?**
- MapReduce应用开发
- MapReduce监控

MapReduce是如何工作的?



MapReduce是如何工作的?

- 整个模型的最上层有四个实体：
  - 客户端: 提交MapReduce Job
  - JobTracker: 协调作业运行, 任务分配。
  - TaskTracker：初始化任务并监控任务的执行。一个TaskTracker可以运行多个任务， 每个任务都有独立的JVM。
  - HDFS，用于共享job所需的文件

- JobTracker和TaskTracker用heartbeat机制来保持状态。

Overview

- MapReduce基本概念
- MapReduce处理与数据流
- MapReduce是如何工作的?
- **MapReduce应用开发**
- MapReduce监控

MapReduce应用开发

- 词频统计(Word Count)
  - 统计不同的词在一个文件集中出现的次数。
    - 比如, 有下面的的两个文件：
      - foo.txt: Sweet, this is the fool file
      - bar.txt: This is the bar file

      期望的输出：

      sweet 1
      this 2
      is 2
      the 2
      foo 1
      bar 1
      file 2

MapReduce应用开发

- Mapper Class

```
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);
    }
  }
}
```

MapReduce应用开发

- Reducer Class

```
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);
  }
}
```

MapReduce应用开发

- Driver Code

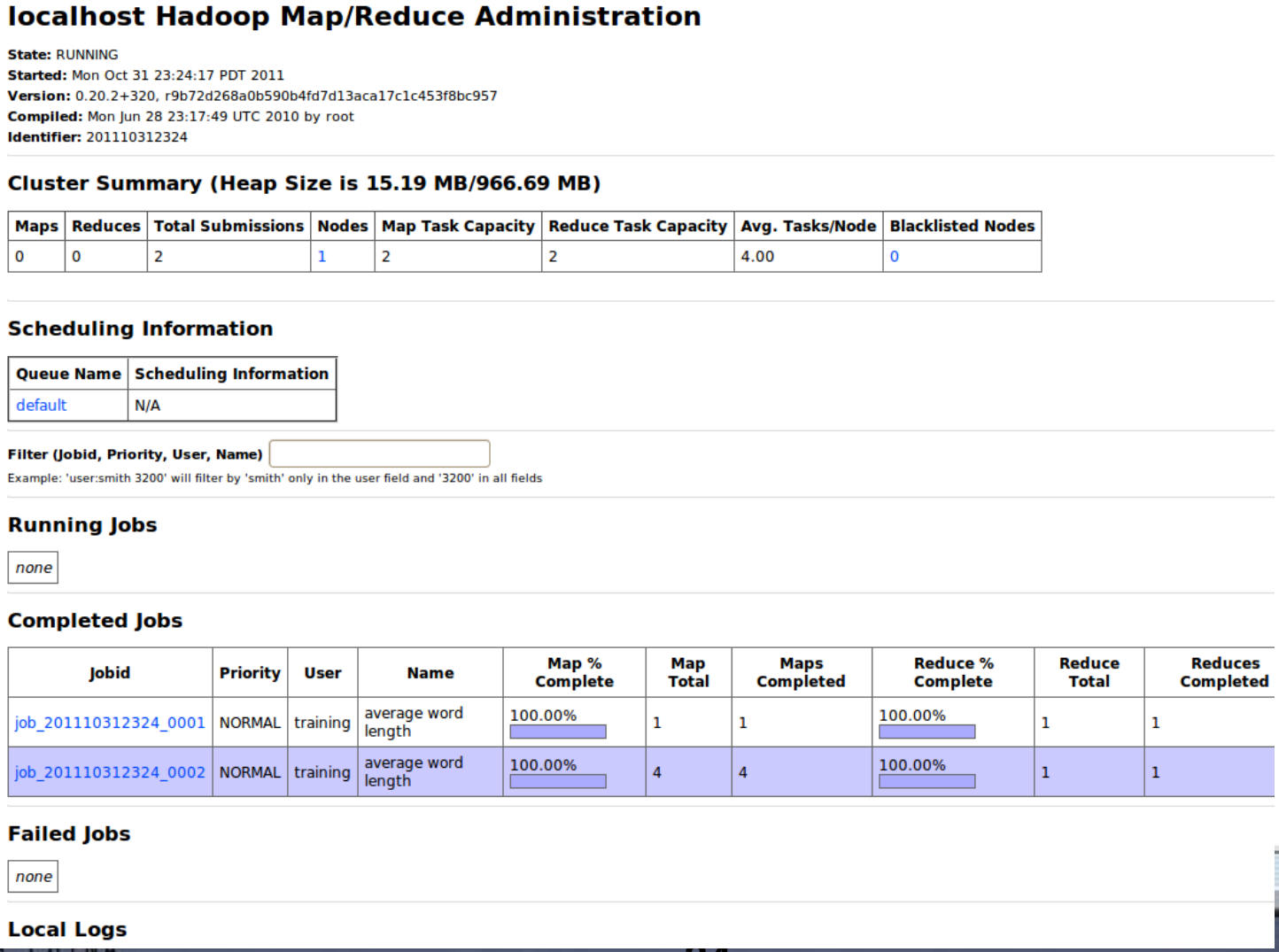
```
public class WordCount {
  public static void main(String[] args) throws Exception {
    if (args.length != 2) {
      System.err.println("Usage: wordcount <in> <out>");
      System.exit(1);
    }

    Configuration conf = new Configuration();
    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(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Overview

- MapReduce基本概念
- MapReduce处理与数据流
- MapReduce是如何工作的?
- MapReduce应用开发
- **MapReduce监控**

MapReduce监控

- http://[jobtrackerhost]:50030/jobtracker.jsp



localhost Hadoop Map/Reduce Administration

State: RUNNING
Started: Mon Oct 31 23:24:17 PDT 2011
Version: 0.20.2+320, r9b72d268a0b590b4fd7d13aca17c1c453f8bc957
Compiled: Mon Jun 28 23:17:49 UTC 2010 by root
Identifier: 201110312324

Cluster Summary (Heap Size is 15.19 MB/966.69 MB)

Maps	Reduces	Total Submissions	Nodes	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes
0	0	2	1	2	2	4.00	0

Scheduling Information

Queue Name	Scheduling Information
default	N/A

Filter (Jobid, Priority, User, Name)

Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Running Jobs

none

Completed Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed
job_201110312324_0001	NORMAL	training	average word length	100.00%	1	1	100.00%	1	1
job_201110312324_0002	NORMAL	training	average word length	100.00%	4	4	100.00%	1	1

Failed Jobs

none

Local Logs

MapReduce监控

Directory: /logs/userlogs /attempt_201110312324_0002_m_000001_0/

Parent Directory		
log.index	87 bytes	Nov 1, 2011 2:13:35 AM
stderr	0 bytes	Nov 1, 2011 2:13:19 AM
stdout	3582317 bytes	Nov 1, 2011 2:13:33 AM
syslog	1471 bytes	Nov 1, 2011 2:13:35 AM

- stderr -> System.err.print
- stdout -> System.out.print
- syslog -> hadoop系统日志

MapReduce监控

- http://[namenodehost]:50070/dfshealth.jsp

NameNode 'localhost:8020'

Started:	Mon Oct 31 23:24:16 PDT 2011
Version:	0.20.2+320, r9b72d268a0b590b4fd7d13aca17c1c453f8bc957
Compiled:	Mon Jun 28 23:17:49 UTC 2010 by root
Upgrades:	There are no upgrades in progress.

Browse the filesystem
Namenode Logs

Cluster Summary

44 files and directories, 15 blocks = 59 total. Heap Size is 15.31 MB / 966.69 MB (1%)

Configured Capacity	:	18.82 GB
DFS Used	:	5.31 MB
Non DFS Used	:	5.16 GB
DFS Remaining	:	13.65 GB
DFS Used%	:	0.03 %
DFS Remaining%	:	72.54 %
Live Nodes	:	1
Dead Nodes	:	0

NameNode Storage:

Storage Directory	Type	State
/var/lib/hadoop-0.20/cache/hadoop/dfs/name	IMAGE_AND_EDITS	Active

Thanks!