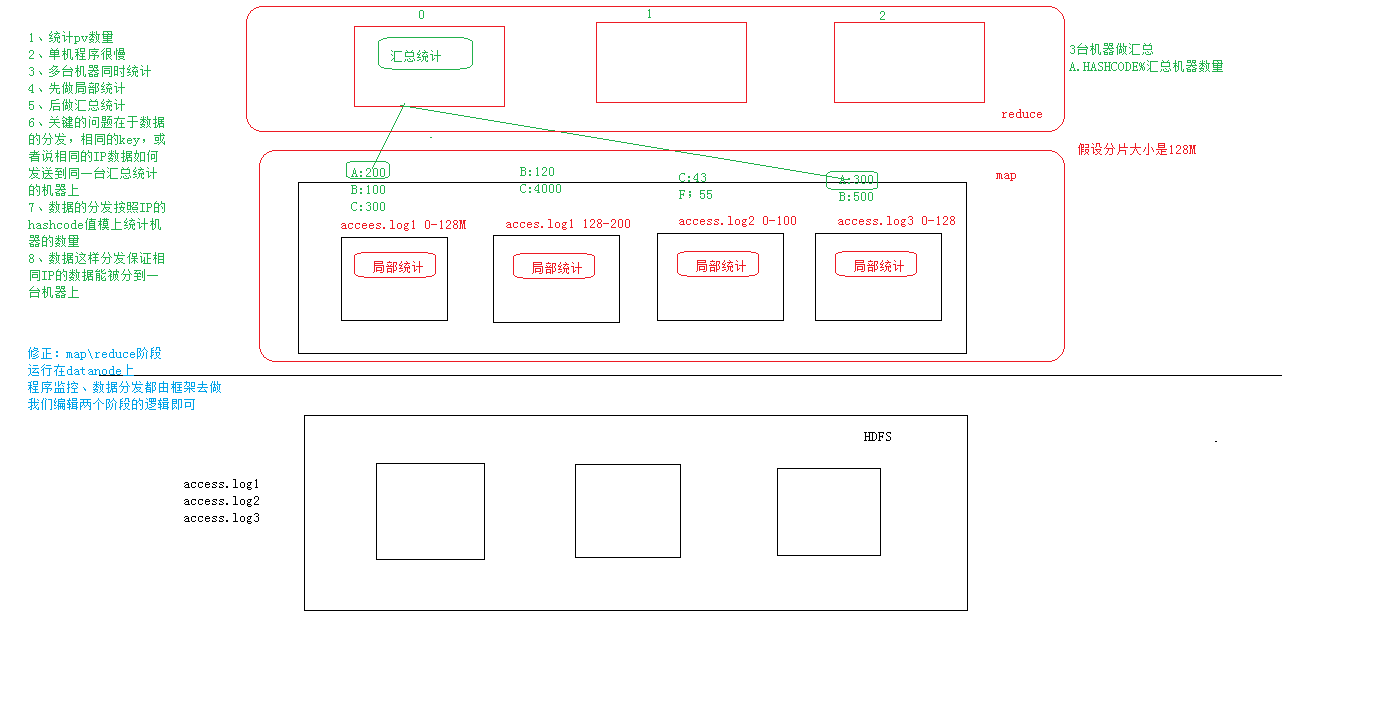
### MR原型解析



### Yarn的安装

* 1. 集群上会运行多个程序，不会把所有资源都分配给一个程序，比如说各个部门的程序都需要运行，也就是说都需要资源，这时候我们就需要一个资源分配和调度程序，帮我们分配资源：cpu,内存，磁盘
  2. Yarn是属于hadoop的一个组件，不需要再单独安装,hadoop中已经存在，只需要配置
  3. Yarn的配置
     1. Yarn也是一个集群,有主节点和从节点
     2. Nodemanager提供资源，ResourceManager负责分配资源
     3. 配置配置文件
        1. 配置map-site.xml

Cp mapred-site.xml.template mapred-site.xml

Vi mapred-site.xml

配置如下

<configuration>

<!--用于执行MapReduce作业的运行时框架-->

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

</configuration>

* + - 1. 配置yarn-site.xml

<!--配置resourcemanager的主机-->

<property>

<name>yarn.resourcemanager.hostname</name>

<value>mini1</value>

</property>

<!--NodeManager上运行的附属服务-->

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<!--配置resourcemanager的scheduler的内部通讯地址-->

<property>

<name>yarn.resourcemanager.scheduler.address</name>

<value>mini1:8030</value>

</property>

<!--配置resoucemanager的资源调度的内部通讯地址-->

<property>

<name>yarn.resourcemanager.resource-tracker.address</name>

<value>mini1:8031</value>

</property>

<!--配置resourcemanager的内部通讯地址-->

<property>

<name>yarn.resourcemanager.address</name>

<value>mini1:8032</value>

</property>

<!--配置resourcemanager的管理员的内部通讯地址-->

<property>

<name>yarn.resourcemanager.admin.address</name>

<value>mini1:8033</value>

</property>

<!--配置resourcemanager的web ui 的监控页面-->

<property>

<name>yarn.resourcemanager.webapp.address</name>

<value>mini1:8088</value>

</property>

在resourcemanager上配置slaves

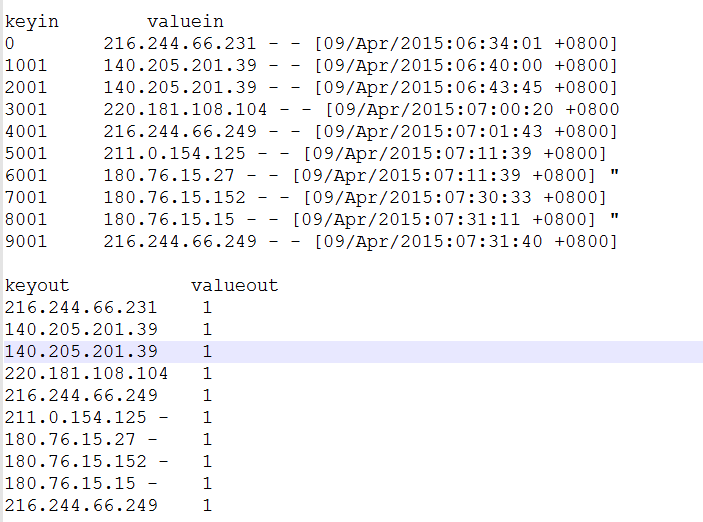
将所有的nodemanager写入

### Yarn的介绍

* 1. 思想：将资源管理和作业调度/监控功能划分为单独的进程，其思想是拥有一个全局的RM和每个应用程序的applicationMaster,应用程序可以是单个作业也可以使一组作业
  2. RM和NM构成了数据计算框架，RM负责分配资源，NM负责管理每台机器的资源，负责监视容器的资源使用情况，资源（cpu,memory，disk,network）
  3. 每个应用程序ApplicationMaster实际上是一个特定的库，他的任务是与RM协商资源，并与NM一起执行和监视任务
  4. Yarn的启动
     1. Yarn-daemon.sh start\stop resourcemanger
     2. Yarn-daemon.sh start\stop nodemanger
     3. Start-yarn.sh
     4. Stop-yarn.sh

### 第一个MR程序

思路：先编写局部统计程序，这个阶段我们称之为映射，会将数据内容转换为key-value,要使用一个框架通常都是实现一个接口或者继承抽象类重写方法，那么这个局部统计程序就需要继承框架系统的mapper类，然后重新map（）方法，map()方法来统计IP出现次数



### MR程序的总结

一种分布式运算程序，分为map阶段和reduce阶段

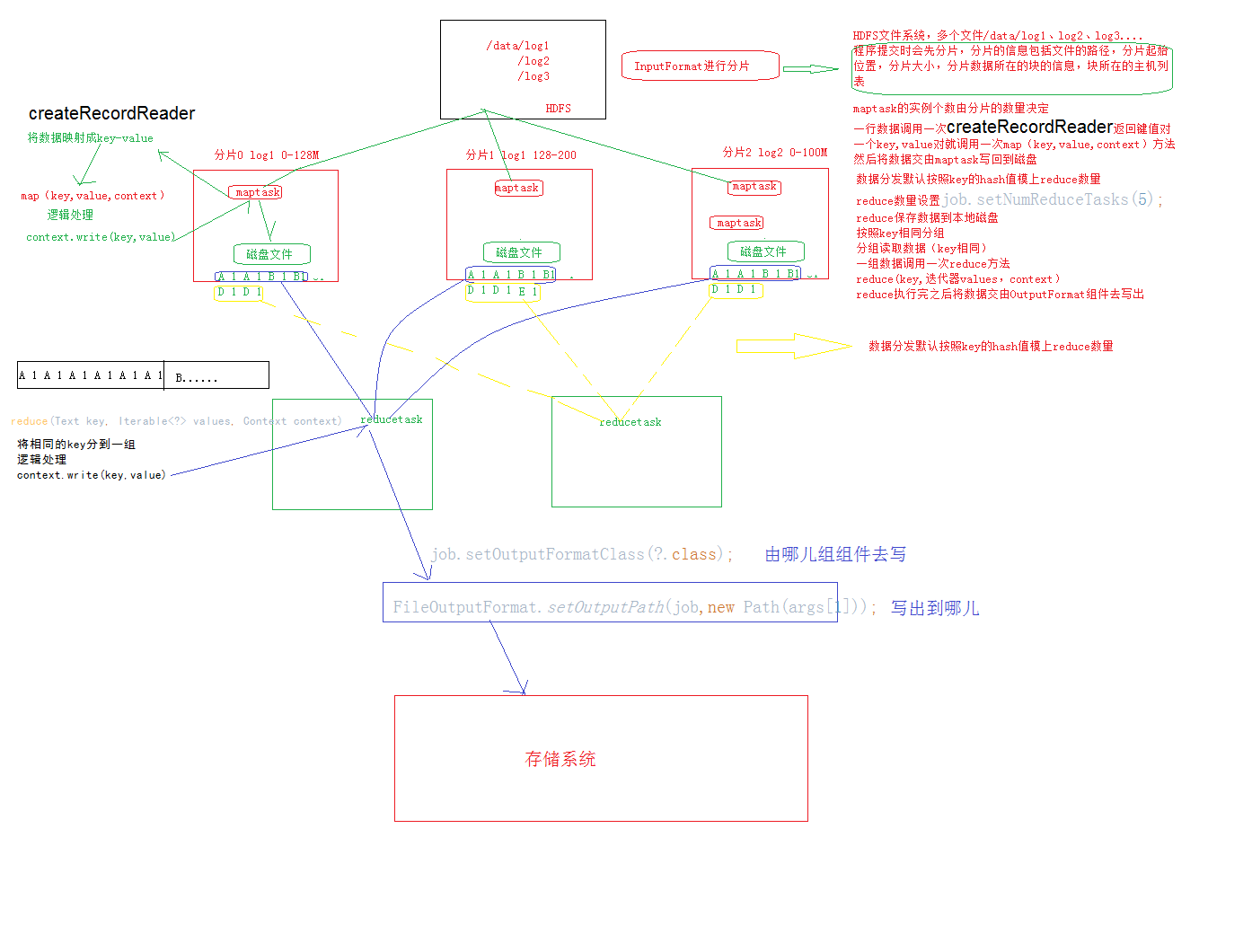
Map阶段会有一个实体程序，不需要我们自己开发，用户只需要维护map方法就可以

默认情况下map程序读取一行数据就会调用一次map方法，而且会将这一行数据的偏移量作为key，这一行数据的内容作为value返回给框架，然后由框架写出context.write(key,value)

Reduce 阶段会有一个实体程序，不要我们自己开发我们需要维护reduce方法

Reduce程序会接受map端输出的中间结果数据，而且相同的key的数据会到达同一个reduce实例中去，每个reduce实例会处理多个key的数据。Reduce程序会将自己收集的数据按照key相同进行分组，对一组数据调用一次reduce方法，并且将参数传给reduce(key,迭代器values，context)，然后写出

### MR运行流程简析



1、分片：逻辑概念，分片信息包括起始偏移量，分片大小，分片数据所在的块的信息,块所在的主机列表。每一个分片对应着一个maptask,通过调整分片的大小可以调整maptask的数量，也就是调整map阶段的并行度。

long minSize = Math.max(getFormatMinSplitSize(), getMinSplitSize(job));//返回1

long maxSize = getMaxSplitSize(job);//返回long的最大值

long splitSize = computeSplitSize(blockSize, minSize, maxSize)

return Math.max(minSize, Math.min(maxSize, blockSize));

//计算分片大小

while (((double) bytesRemaining)/splitSize > SPLIT\_SLOP) {} //分片有一个1.1倍的冗余

1. 分片机制的补充
   1. 分片读取规则
      1. 第一个分片从第一行开始读取，读到分片末尾，再读取下一个分片的第一行
      2. 既不是第一个分片也不是最后一个分片，第一行数据舍去，读到分片末尾，再继续读取下一个分片的第一行数据
      3. 最后一个分片舍去第一行，读到分片末尾
2. 分片大小的设置
   1. FileInputFormat.setMinInputSplitSize(job,1000);
   2. FileInputFormat.setMaxInputSplitSize(job,1000000);
3. 判断是否可以分片，文件若是压缩的需要判断

### 七、wordcount的编写