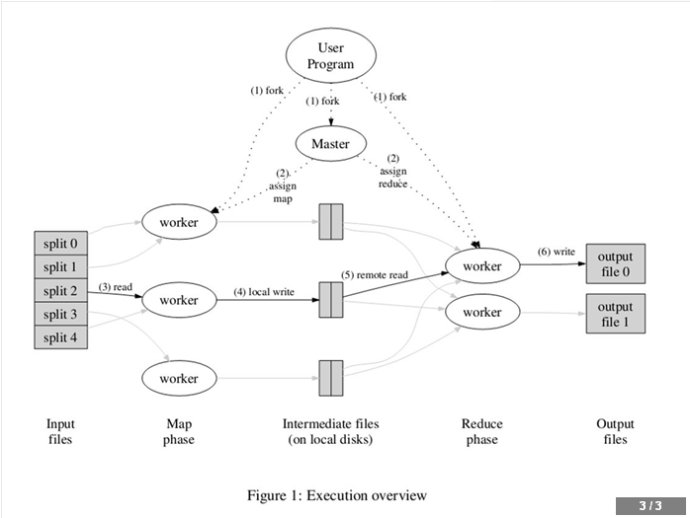
Hadoop简介  
  
Hadoop就是一个实现了Google云计算系统的开源系统，包括并行计算模型Map/Reduce，分布式文件系统HDFS，以及分布式数据库Hbase，同时Hadoop的相关项目也很丰富，包括ZooKeeper，Pig，Chukwa，Hive，Hbase，Mahout，flume等.  
  
这里详细分解这里面的概念让大家通过这篇文章了解到底是什么hadoop：  
  
  
  
1.什么是Map/Reduce，看下面的各种解释：  
  
(1)MapReduce是hadoop的核心组件之一，hadoop要分布式包括两部分，一是分布式文件系统hdfs,一部是分布式计算框，就是mapreduce,缺一不可，也就是说，可以通过mapreduce很容易在hadoop平台上进行分布式的计算编程。  
  
(2)Mapreduce是一种编程模型，是一种编程方法，抽象理论。  
  
(3)下面是一个关于一个程序员是如何个妻子讲解什么是MapReduce？文章很长请耐心的看。  
  
我问妻子：“你真的想要弄懂什么是MapReduce？” 她很坚定的回答说“是的”。 因此我问道：  
  
我： 你是如何准备洋葱辣椒酱的？（以下并非准确食谱，请勿在家尝试）  
  
妻子： 我会取一个洋葱，把它切碎，然后拌入盐和水，最后放进混合研磨机里研磨。这样就能得到洋葱辣椒酱了。  
  
  
妻子： 但这和MapReduce有什么关系？  
  
我： 你等一下。让我来编一个完整的情节，这样你肯定可以在15分钟内弄懂MapReduce.  
  
妻子： 好吧。  
  
我：现在，假设你想用薄荷、洋葱、番茄、辣椒、大蒜弄一瓶混合辣椒酱。你会怎么做呢？  
  
妻子： 我会取薄荷叶一撮，洋葱一个，番茄一个，辣椒一根，大蒜一根，切碎后加入适量的盐和水，再放入混合研磨机里研磨，这样你就可以得到一瓶混合辣椒酱了。  
  
我： 没错，让我们把MapReduce的概念应用到食谱上。Map和Reduce其实是两种操作，我来给你详细讲解下。  
Map（映射）: 把洋葱、番茄、辣椒和大蒜切碎，是各自作用在这些物体上的一个Map操作。所以你给Map一个洋葱，Map就会把洋葱切碎。 同样的，你把辣椒，大蒜和番茄一一地拿给Map，你也会得到各种碎块。 所以，当你在切像洋葱这样的蔬菜时，你执行就是一个Map操作。 Map操作适用于每一种蔬菜，它会相应地生产出一种或多种碎块，在我们的例子中生产的是蔬菜块。在Map操作中可能会出现有个洋葱坏掉了的情况，你只要把坏洋葱丢了就行了。所以，如果出现坏洋葱了，Map操作就会过滤掉坏洋葱而不会生产出任何的坏洋葱块。  
  
  
Reduce（化简）:在这一阶段，你将各种蔬菜碎都放入研磨机里进行研磨，你就可以得到一瓶辣椒酱了。这意味要制成一瓶辣椒酱，你得研磨所有的原料。因此，研磨机通常将map操作的蔬菜碎聚集在了一起。  
  
妻子： 所以，这就是MapReduce?  
  
我： 你可以说是，也可以说不是。 其实这只是MapReduce的一部分，MapReduce的强大在于分布式计算。  
  
妻子： 分布式计算？ 那是什么？请给我解释下吧。  
  
我： 没问题。  
  
我： 假设你参加了一个辣椒酱比赛并且你的食谱赢得了最佳辣椒酱奖。得奖之后，辣椒酱食谱大受欢迎，于是你想要开始出售自制品牌的辣椒酱。假设你每天需要生产10000瓶辣椒酱，你会怎么办呢？  
  
妻子： 我会找一个能为我大量提供原料的供应商。  
  
我：是的..就是那样的。那你能否独自完成制作呢？也就是说，独自将原料都切碎？ 仅仅一部研磨机又是否能满足需要？而且现在，我们还需要供应不同种类的辣椒酱，像洋葱辣椒酱、青椒辣椒酱、番茄辣椒酱等等。  
  
妻子： 当然不能了，我会雇佣更多的工人来切蔬菜。我还需要更多的研磨机，这样我就可以更快地生产辣椒酱了。  
我：没错，所以现在你就不得不分配工作了，你将需要几个人一起切蔬菜。每个人都要处理满满一袋的蔬菜，而每一个人都相当于在执行一个简单的Map操作。每一个人都将不断的从袋子里拿出蔬菜来，并且每次只对一种蔬菜进行处理，也就是将它们切碎，直到袋子空了为止。  
这样，当所有的工人都切完以后，工作台（每个人工作的地方）上就有了洋葱块、番茄块、和蒜蓉等等。  
  
妻子：但是我怎么会制造出不同种类的番茄酱呢？  
  
我：现在你会看到MapReduce遗漏的阶段—搅拌阶段。MapReduce将所有输出的蔬菜碎都搅拌在了一起，这些蔬菜碎都是在以key为基础的 map操作下产生的。搅拌将自动完成，你可以假设key是一种原料的名字，就像洋葱一样。 所以全部的洋葱keys都会搅拌在一起，并转移到研磨洋葱的研磨器里。这样，你就能得到洋葱辣椒酱了。同样地，所有的番茄也会被转移到标记着番茄的研磨器里，并制造出番茄辣椒酱。  
  
（4）上面都是从理论上来说明什么是MapReduce，那么咱们在MapReduce产生的过程和代码的角度来理解这个问题。  
如果想统计下过去10年计算机论文出现最多的几个单词，看看大家都在研究些什么，那收集好论文后，该怎么办呢？   
  
　　方法一：  
      我可以写一个小程序，把所有论文按顺序遍历一遍，统计每一个遇到的单词的出现次数，最后就可以知道哪几个单词最热门了。 这种方法在数据集比较小时，是非常有效的，而且实现最简单，用来解决这个问题很合适。   
  
　方法二：  
       写一个多线程程序，并发遍历论文。   
　　这个问题理论上是可以高度并发的，因为统计一个文件时不会影响统计另一个文件。当我们的机器是多核或者多处理器，方法二肯定比方法一高效。但是写一个多线程程序要比方法一困难多了，我们必须自己同步共享数据，比如要防止两个线程重复统计文件。   
  
方法三：  
      把作业交给多个计算机去完成。   
　　我们可以使用方法一的程序，部署到N台机器上去，然后把论文集分成N份，一台机器跑一个作业。这个方法跑得足够快，但是部署起来很麻烦，我们要人工把程序copy到别的机器，要人工把论文集分开，最痛苦的是还要把N个运行结果进行整合（当然我们也可以再写一个程序）。   
  
　　方法四：  
      让MapReduce来帮帮我们吧！   
  
　　MapReduce本质上就是方法三，但是如何拆分文件集，如何copy程序，如何整合结果这些都是框架定义好的。我们只要定义好这个任务（用户程序），其它都交给MapReduce。  
  
  
map函数和reduce函数　　  
  
  
map函数和reduce函数是交给用户实现的，这两个函数定义了任务本身。   
  
　　map函数：接受一个键值对（key-value pair），产生一组中间键值对。MapReduce框架会将map函数产生的中间键值对里键相同的值传递给一个reduce函数。   
  
　　reduce函数：接受一个键，以及相关的一组值，将这组值进行合并产生一组规模更小的值（通常只有一个或零个值）。   
  
　　统计词频的MapReduce函数的核心代码非常简短，主要就是实现这两个函数。   
  
　　map(String key, String value):   
  
　　// key: document name   
  
　　// value: document contents   
  
　　for each word w in value:   
  
　　EmitIntermediate(w, "1");   
  
　　reduce(String key, Iterator values):   
  
　　// key: a word   
  
　　// values: a list of counts   
  
　　int result = 0;   
  
　　for each v in values:   
  
　　result += ParseInt(v);   
  
　　Emit(AsString(result));   
  
　　在统计词频的例子里，map函数接受的键是文件名，值是文件的内容，map逐个遍历单词，每遇到一个单词w，就产生一个中间键值对<w, "1">，这表示单词w咱又找到了一个；MapReduce将键相同（都是单词w）的键值对传给reduce函数，这样reduce函数接受的键就是单词w，值是一串"1"（最基本的实现是这样，但可以优化），个数等于键为w的键值对的个数，然后将这些“1”累加就得到单词w的出现次数。最后这些单词的出现次数会被写到用户定义的位置，存储在底层的分布式存储系统（GFS或HDFS）。   
  
  
  
工作原理  
  
   
  
　　上图是论文里给出的流程图。一切都是从最上方的user program开始的，user program链接了MapReduce库，实现了最基本的Map函数和Reduce函数。图中执行的顺序都用数字标记了。  
  
　　1.MapReduce库先把user program的输入文件划分为M份（M为用户定义），每一份通常有16MB到64MB，如图左方所示分成了split0~4；然后使用fork将用户进程拷贝到集群内其它机器上。   
  
　　2.user program的副本中有一个称为master，其余称为worker，master是负责调度的，为空闲worker分配作业（Map作业或者Reduce作业），worker的数量也是可以由用户指定的。   
  
　　3.被分配了Map作业的worker，开始读取对应分片的输入数据，Map作业数量是由M决定的，和split一一对应；Map作业从输入数据中抽取出键值对，每一个键值对都作为参数传递给map函数，map函数产生的中间键值对被缓存在内存中。   
  
　　4.缓存的中间键值对会被定期写入本地磁盘，而且被分为R个区，R的大小是由用户定义的，将来每个区会对应一个Reduce作业；这些中间键值对的位置会被通报给master，master负责将信息转发给Reduce worker。   
  
　　5.master通知分配了Reduce作业的worker它负责的分区在什么位置（肯定不止一个地方，每个Map作业产生的中间键值对都可能映射到所有R个不同分区），当Reduce worker把所有它负责的中间键值对都读过来后，先对它们进行排序，使得相同键的键值对聚集在一起。因为不同的键可能会映射到同一个分区也就是同一个Reduce作业（谁让分区少呢），所以排序是必须的。   
  
　　6.reduce worker遍历排序后的中间键值对，对于每个唯一的键，都将键与关联的值传递给reduce函数，reduce函数产生的输出会添加到这个分区的输出文件中。   
  
　　6.当所有的Map和Reduce作业都完成了，master唤醒正版的user program，MapReduce函数调用返回user program的代码。   
  
　　所有执行完毕后，MapReduce输出放在了R个分区的输出文件中（分别对应一个Reduce作业）。用户通常并不需要合并这R个文件，而是将其作为输入交给另一个MapReduce程序处理。整个过程中，输入数据是来自底层分布式文件系统（GFS）的，中间数据是放在本地文件系统的，最终输出数据是写入底层分布式文件系统（GFS）的。而且我们要注意Map/Reduce作业和map/reduce函数的区别：Map作业处理一个输入数据的分片，可能需要调用多次map函数来处理每个输入键值对；Reduce作业处理一个分区的中间键值对，期间要对每个不同的键调用一次reduce函数，Reduce作业最终也对应一个输出文件。  
  
总结：  
  
通过以上你是否了解什么是MapReduce了那，什么是key,怎么过滤有效数据，怎么得到自己想要的数据。  
MapReduce是一种编程思想，可以使用java来实现，C++来实现。Map的作用是过滤一些原始数据，Reduce则是处理这些数据，得到我们想要的结果，比如你想造出番茄辣椒酱。也就是我们使用hadoop，比方来进行日志处理之后，得到我们想要的关心的数据