MapReduce: Simplifified Data Processing on Large Clusters

读后感

MapReduce是一个用于处理和生成大型数据集的编程模型和相关实现。用户指定处理键/valuepair的map函数来生成一组intermediatekey/value对，以及合并与同一中间键关联的所有中间值的reduce函数。

computation接受一组输入键/值对，并生成一组输出键/值对。mapreducelibrary2的用户用两个函数表示计算:Map和Reduce。Map由用户编写，接受一个输入对并生成一组中间键/值对。MapReducelibrarygroupstogetherall intermediatevaluesassociated with The same intermediate key I并将它们传递给Reduce函数。Reduce函数也是由用户编写的，它接受中间键I和该键的一组值。它将这些值合并在一起，形成一个可能更小的值集。通常，每次Reduce调用只生成0或1个输出值。中间值通过迭代器提供给用户的reduce函数。这允许我们处理那些大到无法装入内存的值列表。

MapReduce接口有许多不同的实现。正确的选择取决于环境。谷歌中广泛使用的计算环境的实现:与交换以太网连接在一起的largeclustersofcommodity pc。

映射调用通过自动将出现在OSDI 2004 3中的输入数据划分为一组M分割，分布在多台机器上。输入分块可以由不同的机器并行处理。Reduce调用是通过使用一个分区函数(如hash(key) mod R)将中间键空间划分为R个块来分配的，分区的数量(R)和分区函数由用户指定。

成功完成后，mapreduce执行的输出在R输出文件中可用(每个reduce任务一个输出文件，文件名由用户指定)。通常，用户不需要将这些R输出文件合并到一个文件中——他们通常将这些文件作为输入传递给另一个MapReduce调用，或者从另一个分布式应用程序中使用它们，该应用程序能够处理分割成多个文件的输入。

主程序保存几个数据结构。对于每个map任务和reduce任务，它存储状态(空闲、正在进行或已完成)和工作机器的标识(对于非空闲任务)。master是将中间文件区域的位置从map任务传播到reduce任务的管道。因此，对于每个完成的映射任务，主存储由映射任务生成的R中间文件区域的位置和大小。当地图任务完成时，将接收该位置和大小信息的更新。信息被逐步地推送给正在进行减少任务的工作人员。

容错：

1. 人工错误：主处理器定期ping每一个worker。如果在一定的时间内没有收到worker的响应，则master将该worker标记为failed。worker完成的任何映射任务都被重置为初始空闲状态，因此可以对其他worker进行调度。
2. 主服务器失败：让主服务器写入上述主数据结构的定期检查点很容易。如果主任务终止，则可以从最后一个检查点状态启动一个新副本。然而，考虑到只有一个主处理器，它失败的可能性不大;因此，如果主服务器失败，我们当前的实现将中止MapReduce计算。
3. 语义故障：我们的分布式实现产生相同的输出，就像将会产生一个非故障的连续执行的整个程序。我们依赖于map的原子提交和reduce任务输出来实现这个属性。

导致MapReduce操作用时延长的一个常见原因是出现了“落后者”：某台机器在完成最后的一个map或reduce任务时花费了反常的漫长时间。很多原因都会导致落后者的产生。我们有一个通用的机制来减轻落后者问题。当MapReduce操作接近完成时，主节点会将仍在处理中的剩余任务调度给其它机器作备用执行。原本的执行和备用执行中的任一个完成时都会将对应任务标记为已完成。我们已经调整过这个机制，使它因这个机制而增加的计算资源消耗通常只有一点点。我们已经观察到这一机制有效地减少了大型MapReduce操作花费的时间。

MapReduce的用户指定他们想要的reduce任务/输出文件的数量。通过划分函数可以将数据按中间key划分给各个reduce任务。在给定的划分中，中间key/value对是按增序排列的。这个顺序保证使每个划分产生一个有序的输出文件变得很容易，当输出文件的格式需要支持高效的按key随机访问，或用户需要输出数据有序时，这一性质会非常有用。

合并函数由每个执行map任务的机器调用。通常合并函数与reduce函数的实现代码是相同的。它们唯一的区别就是MapReduce库处理函数输出的方式。reduce函数的输出会写到最终的输出文件中，而合并函数的输出会写到中间文件中，并在随后发送给一个reduce任务。

MapReduce库支持多种不同格式输入数据的读取。用户也可以通过实现一个简单的reader接口来提供对新的输入类型的支持，尽管大多数用户只是使用为数不多的几种预定义输入类型中的一种。reader的实现不一定要提供从文件中读数据的功能。有些情况下，MapReduce的用户发现从他们的map或reduce操作中产生一些额外的辅助文件很有帮助。我们依赖于应用作者来确保这样的边界效应是原子且幂等的。通常应用会写一个临时文件，并在文件生成完毕时将其原子的更名。

参考文献：MapReduce: Simplifified Data Processing on Large Clusters