关于分布式计算的认识

雷哲轩 1700011714

【摘要】 分布式计算是一种将需要较大算力来解决的问题分割成较小的部分，并分配给许多计算处理，最后再将各计算结果汇总综合得到最终结果的计算技术。随着大数据时代的到来，即使是超级计算机，作为单一的计算机，在某些数据的处理和计算上也会显得力不从心。这时，分布式计算就显得尤为重要。甚至对于一些过于庞大的问题，还需要借助大众个人计算机的算力来进行，构成超大规模分布式计算。本文通过对分布式计算的几种模型的简单介绍，来建立对分布式计算初步的认识。

【关键词】 分布式计算；大数据；MapReduce；Actor；Pipeline

1 引言

分布式计算作为划分大任务量并进行分配与结果汇总的计算方法，在大数据时代的重要性不言而喻。在当今的信息时代，个人计算机已经深入千家万户，这些大部分时间均空闲的算力成为大量可被利用的计算资源。于是乎，利用公众的个人计算机提供算力来支持某些大型项目的运行成为有效可行的方法，大大降低了大规模数据处理的难度。现在著名的向公众开放的分布式计算项目有 GIMPS（寻找梅森素数）、Folding@Home（研究蛋白质的折叠结构）、SETI@Home（寻找地外文明）等。

如何高效地分配计算任务、汇总计算结果是分布式计算模型考虑的问题。下面简单讨论一些分布式计算模型，对分布式计算面临的困难与取得成就做一个初步认识。

2 MapReduce：分割与合并

如何将数据量庞大的问题划分为小问题“各个击破”？有一类问题，数据类型较少，可以分割为数据量较小的子问题，从而降低每个子问题的数据量，但这些子问题与原问题的类型一致，相互独立。这样一来，就容易对这些子问题分别求解而不需要考虑其相互关系，最后把结果汇总即可得到原问题的解，这与并行计算是高度相似的，而这种方法称为分治法。Google 提出的 MapReduce 分布式计算模型是分治的典型代表。MapReduce 的分布式计算模式如图 1 所示[1]。

其中任务下发给负责管理的 Master 后，被 Master 按照不同的模式分割为子任务并分配给各个 Mapper 与 Reducer（数目由用户以及任务类型决定），Mapper 根据数据的键与值完成计算后将结果送入键值对缓存区，最后由 Reducer 对计算结果进行统计汇总，输出最终的结果。

以当下的新冠肺炎病例数目统计为例。假设需要统计某日武汉市附近的黄石与荆门的新增确诊人数，输入的数据为该两市内各医院该日的确诊人数，键值对为 (城市, 人数)。当 Master 获取相关任务 (Step1) 后，会对任务进行分割与分配 (Step2)，假定将输入的数据按照 Mapper 的算力分为三组分配给三个 Mapper(Step3)，将 Reducer 分为黄石与荆州两类统计方式。Mapper 按照键类型将计算得到的新增确诊病例数据分别输送给缓冲区 (Step4)，例如将键为黄石的输送给第一个缓冲区，荆州的输送给第二个缓冲区，最后数据被 Reducer 进行加和统计 (Step5)，输出两个地区的新增确诊病例。

可以想象，这种分治的方案是有所限制的，因为它要求切分之后的子任务之间相互不影响，且与原问题类型相同。但这种分治的思想在面对实际的大量数据处理问题时的确是简单且适用的。

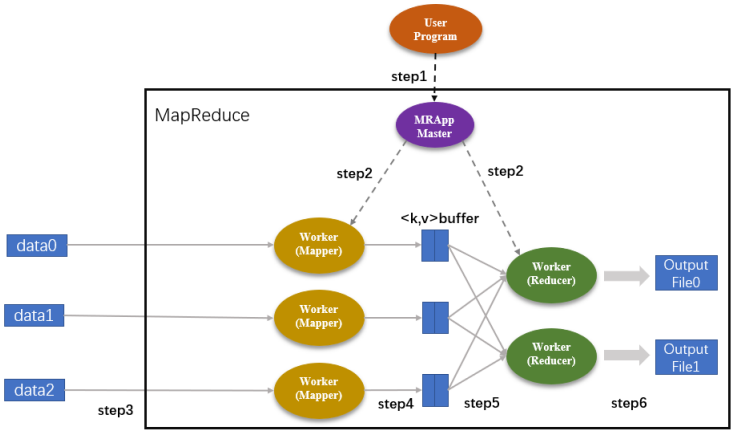

图 1: MapReduce 分布式计算模型

3 Actor：交流与独立

上述提到了 MapReduce 在各个 Mapper 之间进行相互交流的不足，在处理复杂问题上会有所欠缺。同时，MapReduce 中需要有一个 Master 统筹安排各进程的协调与调配。Actor 模型提供了一种各进程拥有各自的内部运算逻辑，同时又有相互交流的机制的方案，在解决了各进程交流问题的同时保持了进程状态行为的独立性[1,2]。在 Actor 模型中，所有的对象都可以看成 Actor，拥有自己的状态、行为与消息，前两者涉及 Actor 本身的性质，后者则涉及不同 Actor 之间的交互。如果将处理数据的任务按照任务类型进行划分，不同的进程模块在处理数据的过程中其不同的作用，那么这些模块可以看作一系列 Actor。这些 Actor 拥有各自的运算规则与方式，同时他们之间的数据传递可以通过 Actor 的消息 Mailbox 进行。一个 Actor 处理收到的消息是按收到的顺序进行的，所以避免了竞争和可能出现的锁死；而且 Actor 处理消息是非同步的，即一个 Actor 发送消息后无需等待响应，可以继续执行其他的本地任务，大幅提升工作效率。

总结来说，Actor 模型中各个 Actor 单元的状态行为被封装，保证了各 Actor 进程运行的稳定与独立，同时与其他 Actor 的交流通过 Mailbox 进行，其次对于消息的处理是按收到顺序的，避免了各消息请求的竞争。当然，Actor 模型中存在的交流非同步性导致其在一些要求数据处理有步骤先后的问题中有所局限，因为一个模块的消息发出后需要等待响应才能执行下一部分任务。

4 Pipeline：顺序与并行

如果采用将任务拆分为多个步骤，按顺序分别执行的方法来处理问题，则需要采用并行的方式进行，否则下一个进程需要等待上一个进程结束后再开始运行，效率较低。可以将数据进行划分，第一个进程对一部分数据进行处理，结果输出给第二个进程，同时开始处理第二部分数据。依次类推，各进程模块实现了并发运算，提高了效率。事实上，这种数据并行的思想与 MapReduce 对数据的分割有相似之处。在大批量输入数据的机器学习项目中，这种 Pipeline 的方法是比较实用的。对于其中存在的数据预处理、模型训练、结果计算输出等方面可以划分给不同的进程，并将输入数据进行划分，

数据预处理模块一份一份地处理数据并输出给模型训练模块进行训练，这种并行的方式减少了整体训练耗时[1,3,4]。

5 认识与感想

我对分布式计算的关注来自于第一周的测验选择题的最后一题。我在第一次做的时候认为超大规模分布式计算与智慧众包相同，并不属于用算法解决问题的范畴，因为我当时认为分布式计算只是简单地划分计算任务并分配给多个计算机，然而事实并非如此。分布式计算中任务分配汇总也有很大的学问，已有多种模型理论，使用多个计算机进行算力分配并非轻而易举。作为数据结构与算法的初学者，我对这些理论只是触及皮毛的，当然也只能有表面的、定性的理解。希望随着课程的深入，所学知识的增多，能够对这些理论模型有更深刻的认识。

参考文献

[1] https://time.geekbang.org/column/article/155575

[2] https://www.cnblogs.com/snailclimb/p/fenbushijisuan.html

[3] https://blog.csdn.net/rise51/article/details/51172195

[4] https://blog.csdn.net/WitsMakeMen/article/details/85000456