文献综述

计算机技术 硕1708班 李冬 M201773208

1. 引言

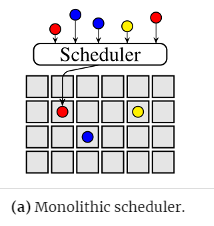
集群调度程序是现代基础设施的重要组成部分，在过去几年中发展迅速，从单一的设计转变为更加灵活的分布式设计。早期的调度程序，例如，Hadoop调度程序，Borg调度程序和Kubernets调度程序，都是单片，单个调度程序进程在一台机器上运行，所有的工作负载由同一个调度程序处理，所有任务都通过相同的调度逻辑运行。但是，随着现代大数据时代的到来，数据量越来越大，对计算的速度和精度要求都比较高，单纯的通过不断增加处理器的数量来增强单个计算机的计算能力已经达不到预想的效果，单个的计算机已经难以满足数据处理的要求，大数据处理的方向逐渐的朝着分布式的计算集群来发展，将分布在不同空间的计算机通过网络相互连接组成一个有机的集群，然后将需要处理的大量数据分散到这个集群中，交由分散系统内的计算机组，同时计算，最后将这些计算结果合并得到最终的结果，因此，需要集群调度程序，来为不同的应用程序分配资源，集群调度的程序的优劣也就决定了集群资源的利用率以及运行效率。

但是现在的集群调度程序并不支持高效的任务抢占机制，例如，YARN和Mesos只支持基于自杀式的任务抢占机制，在这种情况下，被抢占的任务会被终止，如果要继续运行被抢占的任务，则需要重新开始运行，这就导致了大量长时间作业运行缓慢，也就导致了集群资源的严重浪费，在这种背景之下，本篇论文的作者们就提出了新的两种任务调度机制，IP（immediate preemption）和GP（graceful preemption），同时也证明了IP机制适用于MapReduce，GP机制适用于Spark。

1. 调度程序的发展

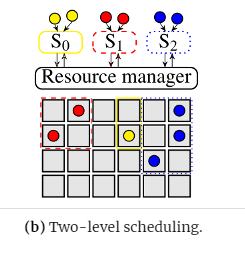
灰框代表集群机器，圆圈代表任务，S代表调度器

1. 单片调度器



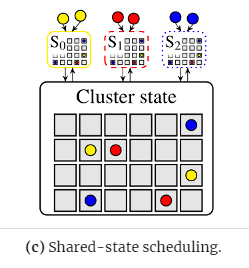
单个调度程序进程在一台机器上运行，例如，JobTracker在Hadoop v1和kube-scheduler Kubernetes中，并将任务分配给机器，所有的工作负载有同一个调度程序处理，所有的任务都通过相同的调度逻辑运行。

1. 两级调度



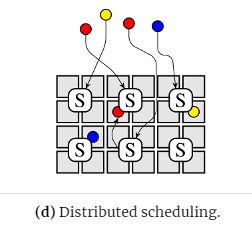
二级调度结构通过分离资源分配和任务安排来解决这个问题。这使得任务放置逻辑可以针对特定的应用程序进行定制，而且还能够在他们之间共享集群。Mesos集群管理器率先推出这一做法，在Mesos中，资源被提供给应用程序级别的调度程序，而YARN则允许应用程序级别的调度程序请求资源。特定与工作负载的调度程序与资源管理器进行交互，为每个工作负载划分集群资源的动态分区，允许定制特定与工作负载的调度策略。这样也带来一个缺点，应用程序级的调度程序失去了onmiscience，也就是优先抢占（较高优先级的任务抢占较低任务级的任务）变得难以实现。

1. 共享状态调度



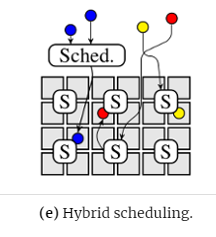
共享状态设计最突出的例子是Google的Omega，微软的Apollo，以及Hashicorp的Nomad容器调度程序。所有这些在一个位置实现了共享集群状态：Omega中的“单元状态”，Apollo中的“资源监视器”以及Nomad的“计划队列”。Apollo不同于其他两个，因为它的共享状态是只读的，调度事物直接提交给集群机器。但是，共享状态体系也有一些缺点，在高争用的情况下可能会遇到性能下降。

1. 分布式调度



完全分布式架构，它们之间根本没有调度程序之间的协调，并使用许多独立的调度程序在处理传入的工作负载。作业通常可以提交给任何调度程序，每个调度程序可以将任务放在集群中的任何位置。与二级调度程序不同，每一个调度程序都负责不同分区。由于它们被设计用于基于最少知识的快读决策，因此分布式调度程序无法支持或负担复杂或特定于应用程序的调度策略。

1. 混合式调度

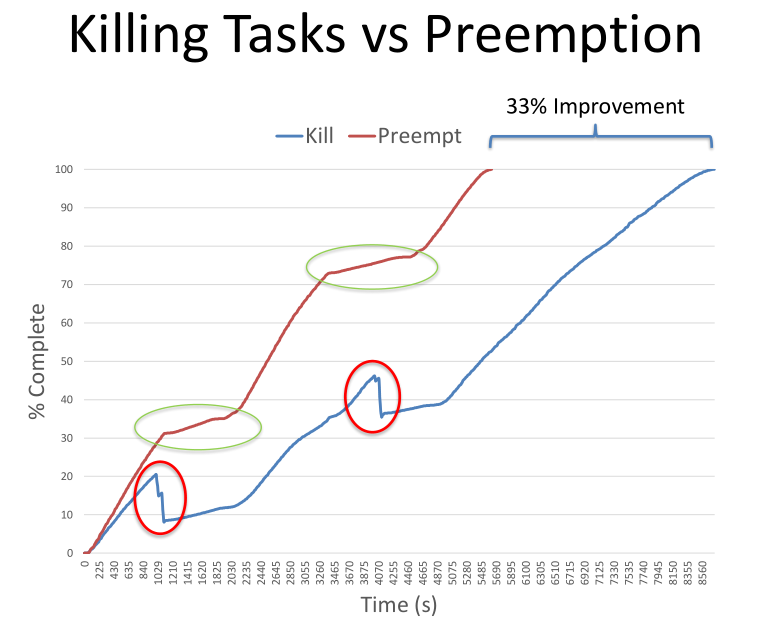


混合架构是最近的发明，主要是学术，试图通过将单片和共享状态设计结合来解决完全分布式的缺点，工作方式有两种调度路径，部分工作量的分布式调度路径，非常短的任务或低优先级的批量工作负载，集中休息一个。但是，在目前，还没有在生产环境中部署混合调度程序。

1. 调度策略

Mesos引入两级调度的思想，将每个应用程序的调度工作委托给应用程序本身，而Mesos仍然负责应用程序之间的资源分配并强化总体公平性，Mesos根据优先级或公平份额策略为应用程序提供资源，应用程序然后计算它如何使用它们，并告诉Mesos他想要的报价是什么资源，这为应用程序提供了很大的灵活性，因为它可以选择现在运行一部分任务，稍后等待更大的分配（组合调度），或者调整任务的大小来适应可用的任务，但是，由于报价是有时间限制的，在浪费资源的长时间任务，在时间限制之后，Mesos会将其杀死，这样虽然激励了使用短期任务，对公平有好处，但是同时也浪费了资源。

为了提高集群资源利用率以及集群的工作效率，人们在调度策略上面做了研究，为了解决killing-based的调度策略带来的资源利用率低的问题，2012年，Sriram Rao提出了抢占式的调度策略，并实现了提出框架Amoeba，实现了弹性的轻量级机制，通过checkpoint保存任务的输出，避免了计算资源的浪费，并通过实验结果证明了抢占式调度策略可以提高作业效率33%。



在Sriram Rao这篇文章的基础上，本文的作者们提出来BIG-C，基于容器的资源管理框架，并且提出了两种任务调度机制，来解决资源利用率与低延时的问题，IP（immediate preemption）机制和GP(graceful preemption)机制,IP机制是将被抢占的任务的资源瞬间降到最低，但是不杀死这个任务，GP机制是逐渐降低被抢占的任务的资源，并在基于YARN调度器上实现了这两个机制，结果表明了BIG-C在短任务延迟和资源利用率上取得了平衡。