《Mesos: A platform for fine-grained resource sharing in the data center》论文解读

随着数据处理的需求不断增长，越来越多的计算框架层出不穷，因此也出现了多种框架共享同一集群的现象。而之前的共享策略是将机器分区给不同的框架（类似虚拟机），这既不能实现高效运作，也没能实现有效的数据共享。这大大阻碍了计算能力的提升。可以说，缺乏一个更加合理的共享策略是这一领域的瓶颈之一。

为了解决这一瓶颈，Mesos 决定采用更加有效的共享策略。针对当前的状况，调度策略有着四种需求：支持现在的或者未来会出现的计算框架各自的资源调度需求、良好的可拓展性、高可用性与高容错性。为了满足这四种需求，Mesos 采用了一种灵活的细粒度资源分配模型。在这个模型中，每个框架分配资源的量由 Mesos 按照公平分享的原则决定，框架根据需求判断资源分配量是否合适，如何使用分配的资源也均由框架自身决定。这种资源调度策略称为委托控制。

委托控制调度策略的灵活性为 Mesos 带来两种优势：支持同一框架不同版本的多个实例同时运行；框架不必拘泥于 one-size-fits-all abstractions，而是可以专注于与框架相吻合的领域的研究。另外由于将部分资源划分工作交由了各个框架来做，Mesos 框架相对比较简洁。

Mesos 采用了主从架构来设计资源调度模式，这在提供了便利的同时，也对用户提出了更高的要求。在使用 Mesos 时，用户不仅要实现逻辑代码，还要实现特定的资源调度器。因此要求用户提高自身的技术水平。

在一些情况下，某些特定的资源对于申请资源的框架来说十分关键(例如可以快速读取)，而当 Mesos 分配的资源不满足这些条件时，框架有权利拒绝 Mesos 分配的资源，并继续等待下一次资源的分配。

由于 Mesos 内的资源没有分区而是由各个框架共享，因此为了防止出现资源耗尽的情况，Mesos 需要拥有对每个框架所占用的资源的回收权。Mesos 可以主动请求一些节点执行者去 kill 一些任务，并且拥有对任务的强制终止权。而对于不能被 kill 任务的框架，Mesos 则会设定一个最低限度资源给它们，当占有资源不超过这个最低限度时便不能被终止。

框架可以对 Mesos 所给的资源设定过滤器，从而得到最合适的资源；Mesos 也会对框架的资源使用情况进行记录，从而可以在之后的分配中为框架提供更好的资源分配。由此也规定框架不要接受不能使用的资源，防止 Mesos 下次将这些资源再次分配给这个框架，产生不必要的浪费。

Mesos 将框架分为 ealstic 与 rigid 两类。前者只需要得到部分所需资源即可运行相应的任务，并且任务完成后即可释放相应的资源；后者则需要获得全部所需资源后才可运行任务，并且所有任务运行完毕才可以释放资源。同时 Mesos 还将资源分为 mandatory 和 preferred 两类，前者是框架运行必需的，后者则是产生优化效果而非必需。这样 Mesos 可以将资源分配利用率尽可能最大化，提高整体的效率。

为了解决长时间任务与短时间任务混合作业的资源分配问题，Mesos 限制了每个节点上能运行的长时间任务的 slot 数，因此可以在每个节点为短时间任务保留 slot，以此避免短时间任务等待过长的情况。

另外，Mesos 给每个 slave 设置了 minimum offer size，从而防止突然出现的资源需求量大的框架无法取得足够资源的情况。

Mesos 不仅提高了计算效率，还真正实现了资源的共享，而非划分出固定的区域分配给不同框架。这无疑是资源调度策略的一场革命。