

Apache Spark 3.0 将内置支持 GPU 调度

如今大数据和机器学习已经有了很大的结合，在机器学习里面，因为计算迭代的时间可能会很长，开发人员一般会选择使用 GPU、FPGA 或 TPU 来加速计算。在 Apache Hadoop 3.1 版本里面已经开始[内置原生支持 GPU 和 FPGA](#)了。作为通用计算引擎的 Spark 肯定也不甘落后，来自 Databricks、NVIDIA、Google 以及阿里巴巴的工程师们正在为 Apache Spark 添加原生的 GPU 调度支持，该方案填补了 Spark 在 GPU 资源的任务调度方面的空白，有机地融合了大数据处理和 AI 应用，扩展了 Spark 在深度学习、信号处理和各大数据应用的应用场景。这项工作的 issue 可以在 [SPARK-24615](#) 里面查看，相关的 SPIP (Spark Project Improvement Proposals) 文档可以参见 [SPIP: Accelerator-aware scheduling](#)



By <https://www.iteblog.com/>



如果想及时了

解 Spark、Hadoop 或者 Hbase 相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

目前 Apache Spark 支持的资源管理器 YARN 和 Kubernetes 已经支持了 GPU。为了让 Spark 也支持 GPUs，在技术层面上需要做出两个主要改变：

- 在 cluster manager 层面上，需要升级 cluster managers 来支持 GPU。并且给用户提供相关 API，使得用户可以控制 GPU 资源的使用和分配。
- 在 Spark 内部，需要在 scheduler 层面做出修改，使得 scheduler 可以在用户 task 请求中识别 GPU 的需求，然后根据 executor 上的 GPU 供给来完成分配。

因为让 Apache Spark 支持 GPU 是一个比较大的特性，所以项目分为了几个阶段。在 Apache Spark 3.0 版本，将支持在 standalone、YARN 以及 Kubernetes 资源管理器下支持 GPU，并且对现有正常的作业基本没影响。对于 TPU 的支持、Mesos 资源管理器中 GPU 的支持、以及 Windows 平台的 GPU 支持将不是这个版本的目标。而且对于一张 GPU 卡内的细粒度调度也不会在这个版本支持；Apache Spark 3.0 版本将把一张 GPU

卡和其内存作为不可分割的单元。

实现概括

Spark Scheduling

在这个层面，我们得允许从 RDD/PandasUDF API 中指定资源请求，这些请求应该在 DAGScheduler 中汇总。TaskSetManager 管理每个 Stage 挂起（pending）的任务，对于那些有 GPU 请求的任务，我们需要处理；对于那些不需要 GPU 的作业，其调度行为和效率应该和之前保持一致。

目前，CPUS_PER_TASK（spark.task.cpus）是一个 int 类型的全局配置，用于指定每个 task 应分配的 cores。为了支持 GPU 的配置，引入了 spark.task.gpus 参数用于指定每个 task 需要申请的 GPU 数。如果用户没有指定 spark.task.cpus 或 spark.task.gpus，那么 Spark 程序将使用默认的值；因为需要向后兼容，所以如果用户没指定 spark.task.cpus 或 spark.task.gpus，这两个参数的默认值分别为 1 和 空。

对于 ExecutorBackend，需要使得它可以识别和管理 GPU，并且把这些信息同步（比如修改现有的 RegisterExecutor 类）到 SchedulerBackend，然后 SchedulerBackend 可以根据这些 GPU 信息，为那些需要 GPU 资源的 task 进行资源分配。

Resource Manager

第一阶段将在 Standalone、YARN 以及 Kubernetes 上支持 GPU。Spark 需要在这三种资源管理上面做一些工作。

Standalone

Standalone 是 Spark 内置的资源管理模式，但是目前的 Standalone 部署模式并不能支持 GPU 等资源。为了能识别 GPU 信息，一种可行的方法是在配置文件里面对 GPU 资源进行配置，Worker 通过读取这些配置信息，并在内存结构里面维护 GPU 和 CPU 等可用资源等信息。同时，在 Master 上通过 allocateWorkerResourceToExecutors 方法对 Executors 申请的资源（包括 GPU）进行分配。

YARN

为了能够在 YARN 上支持 GPU，我们需要使用 YARN 3.1.2+ 版本；同时我们需要在 YARN 集群上做出相关配置，使得 YARN 启动了对 GPU 资源的支持，关于如何在 YARN 上配置 GPU 资源，请参见[这里](#)。

当为 Executors 申请 YARN 容器时，Spark 需要在 YARN 容器请求中将 executor 所需的 GPU 数量映射到 yarn.io/gpu 资源中。YARN 具有 GPU 隔离机制，所以无论是否使用 Docker 容器，对未分配给 YARN 容器的 GPU 资源的使用将会被阻止。

需要注意的是，截至目前 YARN 仅支持 Nvidia GPU。

Kubernetes

从 Kubernetes 1.8 版本开始，Kubernetes 使用设备插件模型（device plugin model）来支持 GPU、高性能NIC，FPGA 等设备。目前 Kubernetes 支持 Nvidia、AMD 和 Intel 的 GPU 设备。在 Spark + k8s 里面为 task 指定 GPU 的数量和在 Standalone 或 YARN 模式里面一样。也是支持 spark.task.gpus 和 spark.executor.gpus 的全局配置，也支持在 RDD stage 中为每个 task 设置。

相关参考文档

- [SPARK-24615 SPIP: Accelerator-aware task scheduling for Spark](#)
- [SPARK-27005 Design sketch: Accelerator-aware scheduling](#)
- [Accelerator-aware scheduling in Apache Spark 3.0](#)
- [SPIP: Accelerator-aware scheduling](#)

本博客文章除特别声明，全部都是原创！
原创文章版权归过往记忆大数据（[过往记忆](#)）所有，未经许可不得转载。
本文链接: [【】](#)（）