基于分布式和 GPU 加速技术的衍生品风险计量引擎实践研究

1. 课题背景

1.1. 衍生品风险指标计算现状

衍生品风险指标计算，往往需要针对不同的场景，对大量的持仓进行百万路径级别的蒙卡仿真。以我司为例，回报路径依赖的“雪球类”产品及其变种持仓约 1000 笔，如每笔估值使用 30 万次模拟，则全量估值需模拟约 3 亿次，如需要计算敏感性指标及 VaR 则需要模拟 700 亿次以上。当前基于单机的，CPU 的风险指标计算系统，完成上述指标的计算，通常需要耗费几个小时。实时的风险监控和报警更是无从谈起。

1.2. 分布式和 GPU 加速技术发展现状

在计算机领域，分布式技术从诞生至今，已经经历了相当长的时间。由于单机不可避免地存在性能和容量的上限，因此，计算机领域的专家和工程师，很早就尝试用多台计算机完成在单机上无法完成的计算任务。通过将一个大的计算任务按照一定的逻辑拆分为若干个小的任务，并分发到若干计算机上并行执行，能获得更高的计算速度，并突破单机的容量上线。当前，分布式技术最热门的话题之一，当属容器和 K8S。容器具有启动快，性能损耗小等优点，因而取代了虚拟机技术，成为了当前应用服务交付部署的首选。而 K8S 作为一种容器编排平台，其具有完备的集群管理能力，因而成为了时下最流行的分布式计算解决方案。

与分布式技术不同，GPU 加速是在单机上，针对部分向量化的，具有并行潜力的问题，使用 SIMD 架构进行加速运算。当前，英伟达

推出的通用并行计算架构 CUDA，是业界广泛使用的 GPGPU 加速平台。针对蒙卡问题，通过使用英伟达的显卡和 CUDA，可同时对数千条蒙卡路径进行并行计算处理，获得比 CPU 加速数十到上百倍的加速效果。

综上所述，为解决衍生品风险指标实时计算的问题，我们引入了分布式和 GPU 加速技术，设计并实现了一套分布式 GPU 风险计量引擎。该系统目前已经进入试运行阶段，取得了非常不错的加速效果。

2. 分布式 GPU 风险计量引擎实现方案

2.1. 总体设计方案

图 1 是分布式 GPU 风险计量引擎的总体技术架构。从图上可见，整个系统包含算法配置，任务管理，缓存管理，底层算力系统共四个模块:

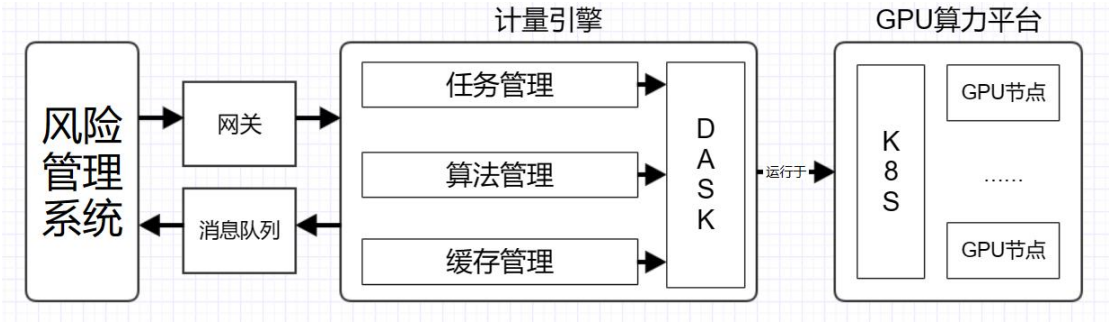

图 1 系统总体架构

- 算法配置：用户可通过此模块自行上传指标算法，并指定算法所需要的计算资源。
- 任务管理：用户可通过此模块启动指标计算任务，查询任务执行进度。
- 缓存管理：自动将指标计算所需的基础数据，分散缓存在算力系统的各个节点，以提升指标计算速度。
- 底层算力系统：一个分布式的 GPU 算力平台，为指标计算任务提供算力支持。

引擎运行流程一般为：

1）用户编写指标计算算法，通过算法配置模块提供的接口，上传到计量引擎；

2）用户通过任务管理模块，启动指标计算任务。计算任务由任务管理模块拆分为若干个子任务，并调度到算力平台的各个节点并行执行，在计算过程中，会使用到缓存管理模块提前缓存好的基础数据；

3）用户接收任务管理模块的任务进度通知，也可以通过接口主动查询任务进度；

基于这一套设计方案，目前我们已经实现了壁虎，折价建仓，大小雪球共 4 类产品的 PV，Delta，Gamma，Vega，Theta 等指标的计算。

2.2. GPU 算力平台的实现

从 2.1 节的设计方案介绍可知，整个系统的核心是图 1 中的分布式 GPU 算力平台。我们基于 K8S+Dask 集群实现了该平台。在 K8S 提供的容器编排功能的基础上，Dask 集群提供了分布式的任务调度和处理能力，数据共享能力。此外，为了实现 GPU 加速，我们在 Dask 集群的 Worker 节点上，引入了 Cupy 这一 GPU 加速计算库，使得在任务被调度到 Worker 节点上时，可以方便地利用 GPU 的加速能力。

2.3. 亮点、难点及相关解决方案

总体而言，分布式 GPU 风险计量引擎有如下几个亮点：

1）采用了分布式和 GPU 加速技术，极大地提高了风险指标计算的效率，对比传统的单机的计算系统，也极大地提高了系统的健壮性和可用性；

2）采用了插件式的设计，指标算法可以在系统运行的过程中随时上传/更新，方便风控人员针对新的产品类型，或新的指标类型添加新的算法；

3）采用了基于 Dask Dataset 的分布式缓存设计，大幅减少了指标计算过程中和数据库的交互，进一步提升了指标的计算效率；

而在系统研发和实施的过程中，我们也遇到了不少的难点：

1）Dask 调度机制和 GPU 容量限制的冲突问题。Dask Scheduler 节点总是将用户提交的任务尽快地分发到各个 Worker 节点执行。对单个 Worker 节点而言，由于其可用的 GPU 数量是有限的，因而 GPU 显存也是有限的。因此，当单个 Worker 的并行任务数超过一定数量的时候，就会超过 GPU 显存的容量限制，导致任务执行失败。为此，我们调整了 Dask Scheduler 的调度方式，从而将同一 Worker 并行运行的任务数限制在一定的数量以下，避免显存不足；

2）Cupy 显存泄露问题。Cupy 在某些特殊的情况下，不会释放已经分配的显存，久而久之，显存就会耗尽而无法执行新的任务；我们也是通过查阅了许多相关资料，并试验了若干方案，最后通过特定的编码方式规避了此问题，并形成了算法编写模板；

3. 性能优势

3.1. 测试方案

为了解和对比我们实现的分布式 GPU 风险计量引擎的实际性能，我们使用了具有以下配置的 GPU 算力平台进行测试：

- 带有两个 GPU 节点的 K8S 集群
- 每个 GPU 节点配置为：4 核+64G 内存+一块 nVidia A6000 GPU

作为对比的基准，我们使用单核+32G 内存的单机运行风险指标的 CPU 版本算法。

在测试任务的选取方面，我们选取了一个计算 658 个持仓的 PV 指标的计算任务。这 658 个持仓中，包含了壁虎，以及大小雪球产品。每个持仓使用 200 万个路径的蒙卡来计算 PV。

3.2. 性能测试结果

在 3.1 节的测试设置下，我们测得在单机配置上，完成 658 个持仓的 PV 计算的耗时为 18798.0 秒，平均每个持仓耗时约为 28.57 秒。而在我们的计量引擎上，完成计算仅需要 101 秒，平均每个持仓的耗时约为 0.15 秒，加速了 186 倍以上。

4. 进一步工作

目前，分布式 GPU 风险计量引擎的主要功能已经实现并投入了试运行，但是仍有许多功能需要进一步完善，包括完善算法管理模块，自动解决算法文件间的依赖关系，优化任务的调度排队机制等。这些都值得在后续的工作中继续探索。