OS-preresearch-RAY

于皓翔 PB23000055

2025 年 3 月 22 日

第一章　介绍

1.1　功能

下面是官方介绍网站

https://docs.ray.io/en/latest/ray-overview/index.html

Ray 是一个开源统一框架，用于扩展 AI 和 Python 应用程序（如机器学习）。它提供了用于并行处理的计算层，因此您无需成为分布式系统专家。Ray 使用以下组件最大限度地降低了运行分布式单个和端到端机器学习工作流的复杂性：

1. 可扩展的库，用于常见的机器学习任务，例如数据预处理、分布式训练、超参数调整、强化学习和模型服务。

2. 用于并行化和扩展 Python 应用程序分布式计算。

3. 用于将 Ray 集群与现有工具和基础设施（如 Kubernetes、AWS、GCP 和 Azure）集成并部署的集成和实用程序。

对于数据科学家和机器学习从业者来说，Ray 可以让你扩展作业而无需基础设施专业知识：

1. 轻松在多个节点和 GPU 之间并行化和分配 ML 工作负载。

2. 利用具有本机和可扩展集成的 ML 生态系统。

3. 对于 ML 平台构建者和 ML 工程师，Ray：

4. 提供计算抽象以创建可扩展且强大的 ML 平台。

5. 提供统一的 ML API，简化加入和与更广泛的 ML 生态系统的集成。

6. 通过使相同的 Python 代码从笔记本电脑无缝扩展到大型集群，减少开发和生产之间的摩擦。

对于分布式系统工程师，Ray 自动处理关键流程：

- 编排——管理分布式系统的各个组件。
- 调度——协调任务执行的时间和地点。
- 容错——无论不可避免的故障点如何，都能确保任务完成。
- 自动扩展——根据动态需求调整分配的资源数量。

现有的用于大数据任务或监督学习任务的框架按照所遵循的并行模型可以分为两类：块同步并行 (Bulk-synchronous Parallel) 和基于任务的并行 (Task Parallel)。块同步并行的代表是 MapReduce、Apache Spark 和 Dryad，这些框架是进程级并行的，因此他们的应用场景比较广泛，可用于训练、模型服务等场景，但对于更细粒度的并行（例如仿真和交互）则不适用。基于任务的并行的代表框架是 CIEL 和 Dask，它们虽然能做到细粒度的并行，但缺少分布式训练和服务的功能。而 TensorFlow 等深度学习框架虽然支持分布式训练，但也不能很自然地支持仿真和服务。最后，TensorFlow Serving 和 Clipper 框架支持模型服务，但是却不支持训练和仿真。Ray 则是一种通用的集群计算框架，既支持模型的训练，又支持对环境的仿真或与环境的交互，还支持模型服务。

为了满足这些任务的需求，Ray 实现了一套统一的接口，这套接口既能表达基于任务的并行计算 (task-parallel)，又能表达基于行动器的并行计算 (actor-based)。前者使得 Ray 能高效地、动态地对仿真、高维状态输入处理（如图像、视频）和错误恢复等任务进行负载均衡，后者行动器的设计使得 Ray 能有效地支持有状态的计算，例如模型训练、与客户端共享可变状态（如参数服务器）。Ray 在一个具有高可扩展性和容错性的动态执行引擎上实现了对任务和行动器的抽象。

1.2　框架

Ray AI 库———一组开源的、Python 的、特定领域的库，为 ML 工程师、数据科学家和研究人员提供可扩展且统一的 ML 应用程序工具包。

Ray Core –一个开源的、Python 的、通用的分布式计算库，使 ML 工程师和 Python 开发人员能够扩展 Python 应用程序并加速机器学习工作负载。

Ray 集群–一组连接到公共 Ray 头节点的工作节点。Ray 集群可以是固定大小，也可以根据集群上运行的应用程序所请求的资源自动扩展或缩小。

对于自定义应用程序，Ray Core 库可让 Python 开发人员轻松构建可在笔记本电脑、集群、云或 Kubernetes 上运行的可扩展分布式系统。它是 Ray AI 库和第三方集成（Ray 生态系统）的基础。

Ray 的架构由应用层和系统层组成，其中应用层实现了 Ray 的 API，作为前端供用户使用，而系统层则作为后端来保障 Ray 的高可扩展性和容错性。

此处请阅读 ***https://zhuanlan.zhihu.com/p/460600694***

1.3　工作原理

Ray 框架基于任务和行动器这两个重要需求，为用户提供了一套 API 及其编程范式。

任务是指在无状态的工作器中执行的远程函数。远程函数被调用时会立即返回一个 future 对象，而真正的返回值可以通过 ray.get(<future 对象>) 的方式来获取。这样的编程模型既允许用户编写并行计算代码，同时又提醒用户要关注数据之间的依赖性。

行动器用来表达有状态的计算任务。每个行动器都会暴露一些可供远程调用的方法，类似于任务中的远程函数，不同的是，使用 f.remote 顺序

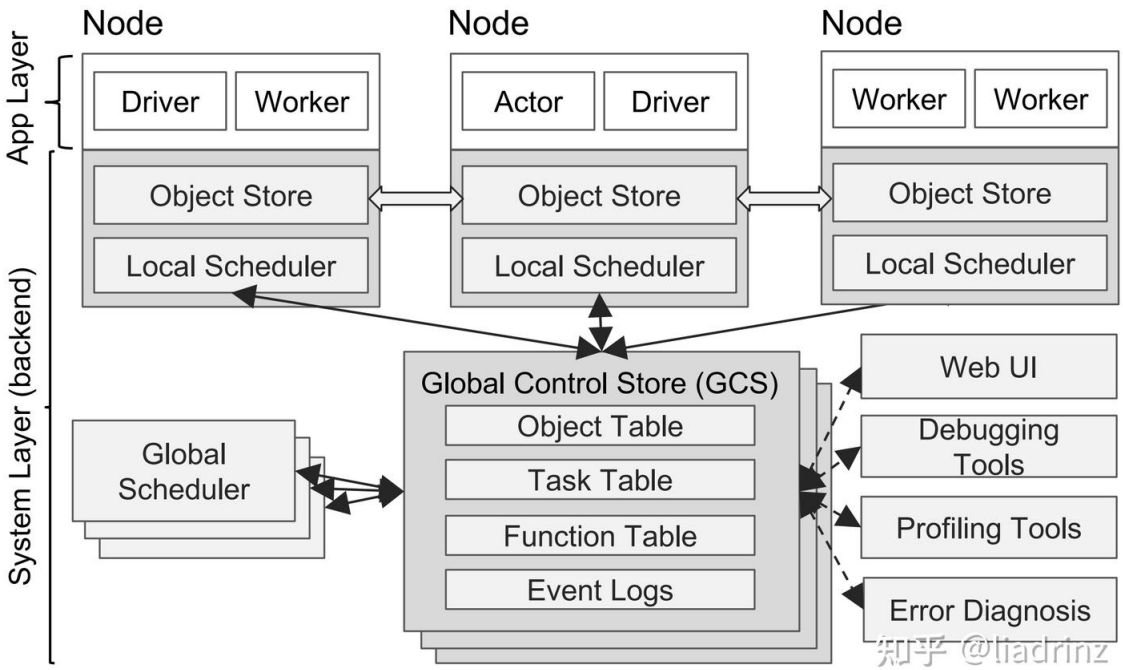
Node
Node
Node
App Layer
Driver
Worker
Actor
Driver
Worker
Worker
Object Store
Local Scheduler
Object Store
Local Scheduler
Object Store
Local Scheduler
System Layer (backend)
Global Control Store (GCS)
Object Table
Task Table
Function Table
Event Logs
Global Scheduler
Web UI
Debugging Tools
Profiling Tools
Error Diagnosis
知乎 @liadrinz

图 1.1: Ray 架构图

地提交若干个远程函数后，这些函数是并行执行的，但在同一个 actor 下使用 actor.method.remote 顺序地提交若干个远程方法后，这些方法将串行地执行。但是，不同 actor 之间的调用是可以并行的。

1.4　总结

Ray 能方便地部署分布式计算，特别是模型的分布式训练与推理。

1.5　OSH 已有的 Ray 相关项目

1.5.1　Ray-LLM 部署优化

https://github.com/OSH-2024/Team$_{S}wanGeese$

Ray+vLLM 多卡推理、Ray + deepspeed、自动数据分发的实现

1.5.2　ROS2 搭建分布式系统

https://github.com/OSH-2022/x-DelayNoMore

基于 ROS 2 搭建一个分布式系统，并使用 Ray 提高系统的分布式性能，提高系统的并行度，从而提高 ROS 2 各个节点的性能和稳定性

1.6　科学与工业界应用

下面的公众号链接是 Ray Forward 2024 活动回顾

https://mp.weixin.qq.com/s/e9zwPOzubRkVBdTNUybT6A

如今 Ray 项目面临的主要挑战与机遇：包括支持 GPU 和 CPU 的混合计算、满足 GPU 密集型和数据密集型工作负载的需求，以及构建强大的库生态系统；强调了 AI 领域内模型规模与数据量的不断扩大、AI 从辅助

决策到成为核心决策工具的转变，以及对硬件加速器和计算架构多样化的需求。

微信的 AI 应用场景广泛，包括语音识别、文字翻译等，面临的挑战包括应用规模庞大、部署复杂及运维难度高等。为应对这些挑战，微信选择了 Ray 平台，创建 Astra-Ray 的整体架构，因其能统一分布式底层，融合不同计算范式，构建完整生态。

随后分享了一个针对百万级节点集群的管理架构设计，该设计包括节点资源预分配至 K8s 集群、使用联邦集群架构、每个应用具有独立调度器以及共享调度机制，以支持高并发调度、资源利用率最大化和降低应用部署复杂度。该架构通过多层调度架构、节点状态的高效同步以及高性能资源管理，实现对庞大集群的有效管理和高效调度。

基于 Ray Workflows 和高效异构调度加速蛋白质结构预测性能，基于 Ray Data 和 Streaming 调度加速蛋白质生成设计性能，基于 Ray 在生命科学领域场景融合计算架构。

第二章　考虑选择的小方向

利用 Ray 进行算力调控，提高**分布式系统**的性能与稳定性

利用 Ray 部署强化学习模型，处理 GPU 和 CPU 的混合计算等问题

第三章　参考文献

1. https://docs.ray.io/en/latest/ray-overview/index.html
2. https://zhuanlan.zhihu.com/p/460600694
3. https://github.com/OSH-2024/Team$_S wan Geese$
4. https://github.com/OSH-2022/x-DelayNoMore
5. https://mp.weixin.qq.com/s/e9zwPOzubRkVBdTNUybT6A