Computer Science and Application 计算机科学与应用, 2021, 11(5), 1334-1340
Published Online May 2021 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2021.115135

Hans汉斯

基于Ray的服务于自动驾驶的远程分布式计算系统

邹星宇[1]，文　军[1]，陈　波[2]

[1]电子科技大学信息与软件工程学院，四川 成都
[2]智能终端四川省重点实验室，四川 宜宾
Email: wenjun@uestc.edu.cn

收稿日期：2021年4月18日；录用日期：2021年5月13日；发布日期：2021年5月20日

摘　要

随着自动驾驶技术的飞速发展，自动驾驶系统需要采集和处理更多的信息，车载的计算平台难以支撑高级别自动驾驶任务的算力需求。本文提出一种使用Ray框架来处理自动驾驶任务的远程分布式计算系统设计方案，克服单一计算平台的算力限制，为更高级别的自动驾驶提供支持。实验表明，Ray框架实现的分布式计算系统可以大幅减少自动驾驶任务的处理时间，提高系统吞吐量，保证自动驾驶任务的实时性。

关键词

深度学习，物联网，人工智能，自动驾驶，Ray

A Ray-Based Remote Distributed Computing System for Autonomous Driving

Xingyu Zou[1], Jun Wen[1], Bo Chen[2]

[1]School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu Sichuan
[2]Intelligent Terminal Key Laboratory of Sichuan Province, Yibin Sichuan
Email: wenjun@uestc.edu.cn

Received: Apr. 18th, 2021; accepted: May 13th, 2021; published: May 20th, 2021

Abstract

With the high development of autonomous driving technology, autonomous driving systems need to collect and process more information, and the on-board computing platform has been unable to

文章引用: 邹星宇, 文军, 陈波. 基于 Ray 的服务于自动驾驶的远程分布式计算系统[J]. 计算机科学与应用, 2021, 11(5): 1334-1340. DOI: 10.12677/csa.2021.115135

support the computing requirements of high-level autonomous driving tasks. Thus, we propose a remote distributed computing scheme that uses the Ray framework to implement such kinds of tasks to break through the computing power limitation of a single computing platform, and provide the support for higher-level autonomous driving. Experiments show that the distributed computing system implemented by the Ray framework can greatly reduce the processing time of autonomous driving tasks, increase system throughput, and ensure the real-time performance of autonomous driving tasks.

Keywords

Deep Learning, Internet of Things, Artificial Intelligence, Autonomous Driving, Ray

1. 介绍

自动驾驶依靠人工智能等技术[1] [2]实现不需要人工操纵的高效、安全的驾驶。未来自动驾驶和智慧道路等技术的深度整合将能够统筹协调区域内的车辆的路线与速度，大幅提高出行效率并减少能源浪费。自动驾驶同时还能避免人工驾驶中的各类安全隐患。

自动驾驶共有六个级别[3]，目前拥有自动驾驶能力的车辆的自动驾驶级别大都在 1 级或 2 级，距离高度自动驾驶或者完全自动驾驶有着很大的差距[4]。其中一个主要的问题就是车载处理器的算力不足的问题。要想实现完全自动驾驶，车载处理器至少要进行以下的几个计算任务，如：目标识别算法、车道线识别算法、端到端决策算法[5]和有限状态机决策算法等[6]。

但现在设计的自动驾驶车辆上搭载的计算平台很难满足实时地完成以上各类任务的算力需求。例如目前量产的带有自动驾驶功能的汽车中，特斯拉 HW3.0 硬件可以提供 144TOPS 的算力。Nvidia 的 Orin Soc 则能够达到 200TOPS 的算力，但是其自动驾驶仅能达到 L2 级，也就是说，想要达到更高级别的自动驾驶，必然会需要算力远大于现有产品的硬件，但必然会面对成本和功耗等因素的限制。

因此更高级别的自动驾驶向计算设备提出了以下两个需求：高性能需求，实时性需求。在车辆自动驾驶过程中有大量的计算任务，且其中很多的任务之间在一定时间内是没有相互依赖性的，也就是说在合适的时机并行处理这些任务可以大大避免因任务排队而造成的时延。但是仅靠单个汽车上的计算核心难以完成高效的任务调度和任务处理，导致计算耗费较长时间，难以达到实时性的要求。

面对单机性能的瓶颈，大规模分布式系统能够为自动驾驶系统提供充足的算力，从而满足自动驾驶的高性能需求和实时性需求。

现在常用的大规模分布式的系统，如 Spark [7]、消息传递接口(MPI) [8]、MapReduce [9]等，均没有为自动驾驶提供算力加持的案例，因此这里基于其系统特征讨论其在自动驾驶系统中的适用性。Spark 和 MapReduce 采用了 BSP 执行模型，依赖分布式的文件存储系统对数据进行管理。它具备较强的鲁棒性，能支持大规模的扩展，但不适合实时计算；消息传递接口(MPI)是一套专门为并行计算的机器和大规模集群设计的信息传递接口。它通过点对点、群组收集或者群组广播的方式完成信息的交互和同步，提供了一套进程之间通信的协议。

为了突破车载计算核心性能的限制，我们基于 Ray 框架[10]作为分布式实现自动驾驶系统的基础工具，将汽车计算核心难以处理，或者需要消耗大量时间的计算任务交由远程分布式 Ray 集群处理。Ray

实现了动态任务图计算模型，它将应用程序抽象地建模为一个在运行过程中动态生成依赖的任务图。在此模型之上，Ray 提供了角色模型和并行任务模型的编程范式。Ray 的设计能够更高效地并行处理计算任务，帮助自动驾驶汽车快速完成计算任务，实时地完成决策。

2. 基于 Ray 的自动驾驶系统设计

自动驾驶的实现不仅要求车辆准确地采集环境信息，还要求车辆能够根据环境信息及时做出正确的驾驶决策。因此随着环境信息量的提升，车辆处理器需要更强的数据处理能力，更强的计算能力才能实时地完成决策。

当前已量产的拥有部分自动驾驶能力的汽车中的大部分是采集的视觉环境感知模型，其输入信息大多是经由摄像头采集的图像信息，但对于更高级别的自动驾驶来说这些信息是远远不够的。而在引入了基于激光雷达的环境感知模型和各类雷达数据之后，现有平台已经难以支撑对海量的数据的计算处理任务。随着通讯技术的发展，通讯延迟或将控制在车辆操作窗口时间以内[11] [12]，因此车辆的本地计算任务大部分可以交付于分布式云计算平台进行处理。借助分布式云计算平台更强大的算力以大大加强自动驾驶汽车的计算能力，使其能够面对更加复杂的环境，拥有更高级别的自动驾驶能力。

Ray 是一个高性能的分布式执行引擎，设计之初便致力于于支持人工智能应用的分布式执行，拥有相关的人工智能应用的相应支持与所需的 API。支持 TensorFlow [13]等机器学习框架尤其是对强化学习应用有着更高层次的支持。

远程自动驾驶系统中的远程分布式服务器集群要解决的就是汽车上计算平台的算力不足的问题，即要满足高性能和低时延的需求，但其实低时延的需求不仅要求远程分布式服务器集群的高性能以减少计算的时间，还对传输的网络延迟有要求。在这里不讨论网络延迟相关的技术，仅从降低计算时间出发，那么这两个需求可以合并成对高性能的需求。

通过分别与单机平台和基于 MPI 的分布式系统进行性能测试实验，我们发现基于 Ray 的分布式系统的确在计算能力上对两者均有着很大的优势。基于 Ray 的远程分布式服务器集群能够提供远大于汽车本地搭载的计算平台的算力，同时也拥有相对其他分布式系统更好的性能，能够更好地高级别自动驾驶的需求。

2.1. 远程自动驾驶系统的任务调度

远程自动驾驶系统可划分为车辆本地处理平台和远程分布式服务器集群，两部分之间的任务调度方式如图 1 所示，自动驾驶各个子任务由本地计算平台做预处理，而它将会将设计好的需要云平台负责的计算和数据处理的任务则交付于分布式计算平台进行处理。而这些任务借助基于 Ray 框架的分布式计算平台能够快速地完成，并且把相关任务处理完成后所需的结果交回车辆计算核心，控制车辆行驶。

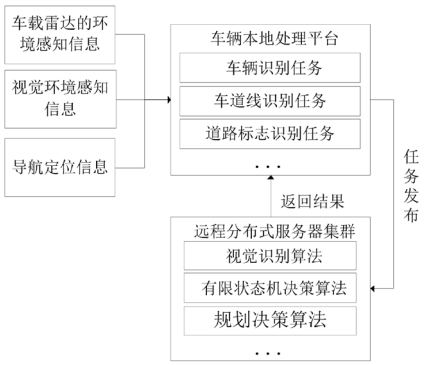

Figure 1. Remote autonomous system schematic diagram
图 1. 远程自动驾驶系统示意图

2.2. 基于 Ray 的自动驾驶框架

Ray 的体系结构如下图，它能够提供高可伸缩性和容错能力的系统层[7]。

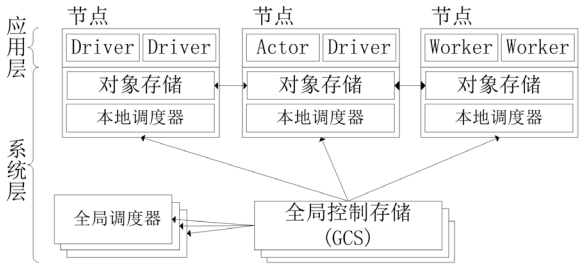

Figure 2. Ray's architecture diagram
图 2. Ray 的体系结构示意图

如图 2 所示，系统层和应用程序层两部分共同组成 Ray 的体系结构，系统层主要包含 GCS 和自底向上的分布式本地调度器。GCS 存储着系统中的所有控制状态，这样其他组件就可以是无状态的，方便容错，也方便其他组件的横向拓展。而本地调度器降低全局调度器的负担，增强了系统的吞吐量，他们共同调度各类任务给应用层处理，同时也对整个系统的数据进行管理。而应用层由 Driver、Worker 和 Actor 三类进程组成，他们分别处理用户程序、无状态任务和有状态任务。机器学习相关的 API 和计算模型在应用程序层实现。

正是有了以上机制，在自动驾驶系统中各类有状态的任务，如：路线规划任务、跟车任务等可以在分布式平台上不改变其本身各项依赖地完成处理。而各类无状态的任务，比如在进行视频处理时可以将视频分成图片的集合，一个 headnode 将图片分发给各个 workernode 加速处理，最后 headnode 汇总处理。而由少数 headnode 和大量 workernode 组成的集群便是基于 Ray 的分布式系统，如图 3 所示。

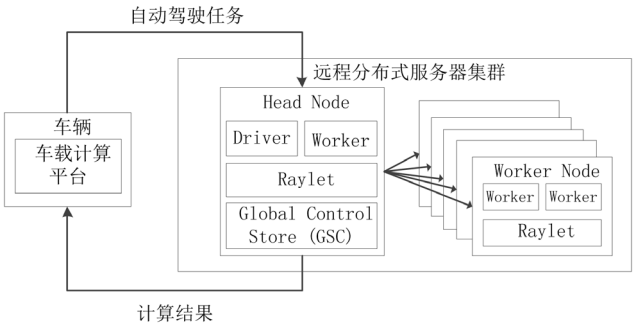

Figure 3. Diagram of remote distributed server cluster structure
图 3. 远程分布式服务器集群结构示意图

Ray 集群包括一组同类的工作节点和一个集中的分发结点。分发结点中的 Driver 进程是一种特殊的 Worker 进程，它负责执行用户程序，它将任务提交给其他 Worker 进程处理，因此能够了解程序各个部分的依赖关系，并且在合适的时间将任务分布式异步地在集群上进行处理。从而大大提升了单位时间内的计算速度。

Ray 框架对于任务引入了 Actor 模型。在 Ray 中的 Actor 有状态，不同于 Task (Ray 中无状态，可细分的任务)，它源于一种并发模型，也是共享内存并发模型的替代方案。因此在 Ray 中使用的 Actor 模型提供给 Ray 将传统复杂分布式事务给拆分成由事件驱动的线性处理流程的能力。因此使得 Ray 框架的性

能更好，吞吐量更高，从而响应速度更快。

在远程自动驾驶系统的应用中 Ray 集群将位于服务器端，它将接收来自车辆的数据，并且由一个集中的 GCS 控制，调用集群中可用的单元为车辆进行自动驾驶的计算。

通过 Ray 的独特设计，它能够通过调用多个工作节点来为远程自动驾驶系统提供以下功能：

1) 支持毫秒级的任务，同时拥有每秒处理百万级别任务数的能力。

2) 嵌套的并行性。Ray 可以提供将一个任务拆分成数个并行处理的子任务，并且将这些子任务进一步拆分为更小的并行子任务。

3) 在运行的时候动态地确认任务的依赖。

4) 在共享的可变状态下运行任务。

5) 支持异构资源，如 CPU、GPU 等。

Ray 提供的以上的特性能够满足自动驾驶的以下需求：

1) 自动驾驶对系统性能的要求。自动驾驶的各个任务需要动态执行，这些任务可能相互之间有着耦合，例如高精度地图、车道识别、路权优先级判断、长距离路径规划、短距离路径规划等等。嵌套的并行性能够使得这些任务以恰当的方式并行处理。而这种强大的并行能力又不会影响到有着共享的可变状态以及有着动态的依赖的任务。正是因为其强大的并行能力，因此分布式集群能大大加速任务处理的速度。

2) 自动驾驶复杂任务对异构硬件的需求。例如车道规划、图像识别、车辆控制等等任务，处理这些任务需要用到异构硬件。而 Ray 能够通过 Actor 来调用诸如 CPU、GPU 等异构资源，满足自动驾驶的需求。

2.3. 实验仿真

本文实验基于 Python 实现。硬件配置：4 台相同配置计算机，CPU 均为 10900k，GPU 均为 RTX2080TI；软件配置：运行环境为 Python 3.8.5；基于 Ray 1.1.0 框架和 Redis 3.5.3 数据库。若未特别说明则本文实验均基于此实验环境。

我们首先准备了 48 次基于 SHA256 哈希函数的计算任务用以模拟自动驾驶中所需要进行的任务。在分别由基于 Ray 的 4 台计算机系统、2 台计算机系统以及单机平台上调用不同 worker 数量的情况下，统计完成计算任务的耗时。结果如图 4 所示，可以看到在 16 个 worker 的情况下 Ray 框架还能运用上 CPU 超线程的特性对计算进行加速。与调用单个 Worker 的计算耗时相对比，Ray 的分布式集群大大降低了计算耗时，且随着节点和 worker 数量的增多节约更多计算时间。

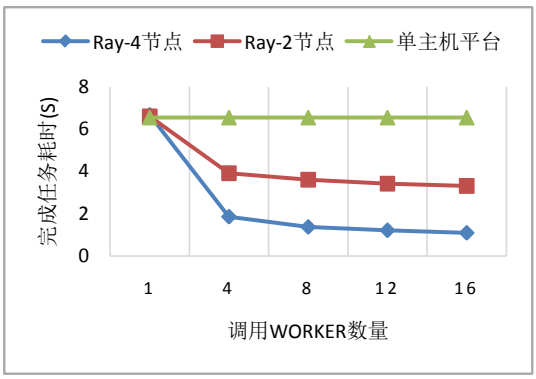

Figure 4. Comparison results of task time consuming between ray cluster and single host platform
图 4. Ray 集群同单主机平台完成任务耗时结果对比

图 4 显示在处理相同的复杂计算任务集合时，单主机平台完成计算任务所需时间与 Ray 分布式集群仅调用一个 Worker 的情况下所需时间相同，而随着分布式集群调用更多 Worker 后其所耗时间大幅度降低。将复杂运算交由基于 Ray 框架的分布式集群以加速运算是可以实现且效果很好的。以上面的任务为例，当有多个计算任务需要处理的时候单机平台受限于自身处理设备的并发量和计算能力，其算力是有一个明显的上限的。

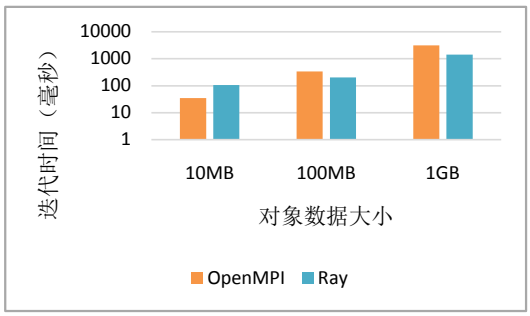

Figure 5. Average execution time of Ray and MPI in AllReduce
图 5. Ray 与 MPI 在 AllReduce 中的平均执行时间

我们利用 AllReduce 来模拟分布式情况下的机器学习工作负载，在拥有 4 个节点的 Ray 集群上使用 Ray 的 API 实现的 Ring AllReduce，又使用 OpenMPI 来进行 Ring AllReduce，结果如图 5 所示，Ray 集群在 100 MB 时和 1 GB 时均明显好于 OpenMPI 的成绩。这将在进行大量的数据处理时为自动驾驶任务节约出宝贵的时间。

而基于 Ray 框架的分布式系统则因为 Ray 的优秀的任务调度设计和分布式所带来的算力资源的极大丰富，其处理能力大大强于基于 MPI 的分布式计算系统以及单机系统。

3. 结论

本文针对自动驾驶对计算平台高性能、低时延的需求，采用 Ray 构架实现远程分布式计算系统来处理自动驾驶任务，实验证明它能够将复杂的自动驾驶任务以较少的时间完成的，为自动驾驶提供一个更实用的平台。但由于自动驾驶中的实时数据量较大，存在车辆与服务器之间传输数据的延迟问题，未来做好车辆和服务器的数据传输的保障，使用更高效的通信方式，则能够提供更安全和更高效的驾驶系统支持，让自动驾驶汽车真正成为可靠的工具。

致　谢

本项目得到厅市共建智能终端四川省重点实验室开放课题 SCITLAB-0013 资助。

参考文献

[1] Geoffrey, H., Yann, L. and Yoshua, B. (2015) Deep Learning. *Nature*, **521**, 436. https://doi.org/10.1038/nature14539

[2] Jordan, M.I. and Mitchell, T.M. (2015) Machine Learning: Trends, Perspectives, and Prospects. *Science*, **349**, 255-260. https://doi.org/10.1126/science.aaa8415

[3] SAE International (2016) Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. SAE International, Warrendale, PA.

[4] Geiger, A., Lenz, P. and Urtasun, R. (2012) Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite. 2012 *IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, 16-21 June 2012, 3354-3361. https://doi.org/10.1109/CVPR.2012.6248074

[5] Bojarski, M., Del Testa, D., Dworakowski, D., *et al*. (2016) End to End Learning for Self-Driving Cars. http://arxiv.org/pdf/1604.07316.pdf

[6] 张新钰, 高洪波, 赵建辉, 等. 基于深度学习的自动驾驶技术综述[J]. 清华大学学报(自然科学版), 2018, 58(4): 438-444.

[7] Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M.J., Shenker, S., and Stoica, I. (2012) Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. *Proceedings of the* 9*th USENIX Conference on Networked Systems Design and Implementation* (*2012*), USENIX Association, 15-28.

[8] Gabriel, E., Fagg, G.E., Bosilca, G., Angskun, T., Dongarra, J.J., Squyres, J.M., Sahay, V., Kambadur, P., Barrett, B., Lumsdaine, A., Castain, R.H., Daniel, D.J., Graham, R.L. and Woodall, T.S. (2004) Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation. *Lecture Notes in Computer Science*, **3241**, 97-104. https://doi.org/10.1007/978-3-540-30218-6_19

[9] Dean, J. and Ghemawat, S. (2008) MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, **51**, 107-113. https://doi.org/10.1145/1327452.1327492

[10] Moritz, P., Nishihara, R., Wang, S., *et al.* (2017) Ray: A Distributed Framework for Emerging AI Applications. *Proceedings of the* 13*th USENIX Symposium on Operating Systems Design and Implementation* (*OSDI*'18), 8-10 October 2018, Carlsbad, CA, 561-577.

[11] Chang, K. (2015) Wireless Communications for Vehicular Safety. *IEEE Wireless Communications*, **22**, 6-7. https://doi.org/10.1109/MWC.2015.7054711

[12] Xiong, K., Leng, S., Chen, X., Huang, C., Yuen, C. and Guan, Y.L. (2020) Communication and Computing Resource Optimization for Connected Autonomous Driving. *IEEE Transactions on Vehicular Technology*, **69**, 12652-12663. https://doi.org/10.1109/TVT.2020.3029109

[13] Abadi, M., Barham, P., Chen, J., *et al*. (2016) TensorFlow: A System for Large-Scale Machine Learning. *Proceedings of the* 12*th USENIX Symposium on Operating Systems Design and Implementation* (*OSDI*), Savannah, 2-4 November 2016, 265-283.