边缘服务器上 QoS 感知的任务调度方法

李晓晓

华中科技大学计算机科学与技术学院 武汉 430074

摘 要 互联网规模的迅速扩展促使全球数据总量呈现爆炸式的增长。物联网等新的应用对数据存储及处理的实时性提出了更高的要求。为了减少发送到云端的数据量，减少延迟和计算成本，边缘计算被提出并广泛应用。但不同任务的 QoS 需求是不同的，为了满足异构的 QoS 要求，需要复杂的解决方案来调度资源有限的边缘服务器上的任务。本文研究了目前资源有限的边缘服务器上 QoS 感知的调度方法，探究了任务卸载的处理方案，并对未来的研究方向做了简要的描述。

关键词 边缘服务器；服务质量；任务调度

QoS-aware Task Scheduling Methods on Edge Servers

Li Xiaoxiao

(Department of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, 430074)

Abstract The rapid expansion of the scale of the Internet has led to the explosive growth of the global data volume. New applications such as the Internet of Things have put forward higher requirements for real-time data storage and processing. In order to reduce the amount of data sent to the cloud, reduce latency and computing costs, edge computing has been proposed and widely used. But the QoS requirements of different tasks are different. In order to meet the heterogeneous QoS requirements, a complex solution is needed to schedule tasks on edge servers with limited resources. This paper studies the QoS aware scheduling methods on the edge server with limited resources, explores the task unloading processing schemes, and gives a brief description of the future research direction.

Key words edge server; Quality of Service; task scheduling

1 引言

1.1 边缘计算

近年来，随着物联网（Internet of Things，IoT）和移动互联网的飞速发展，前端设备数量和产生的数据量呈现指数级的增长。事实上，目前的云基础设施本身无法支持大量当前的物联网应用程序，这主要有三个原因：首先，由于带宽限制、处理开销和传输成本，将大量生成的数据从创建位置（终端设备）传输到处理位置（云服务器）是不切实际的。其次，从终端设备到云服务器的严重端到端延迟会影响需要实时分析的应用程序的性能，如在线游戏、视频应用程序等。最后，一些数据可能涉及隐私和安全问题，跨越整个互联网的数据传输可能会带来一些隐患。

为了解决这些问题，边缘计算被提出，这种全新的计算模式能够避免网络瓶颈，克服通信开销，并且减少数据传输延迟。边缘计算将云计算和服务扩展到网络边缘，使处理、分析和存储更接近创建和使用请求的地方，其目标是减少发送到云端的数据量，减少延迟和计算成本。

对物联网而言，边缘计算技术取得突破，意味着许多控制将通过本地设备实现而无需交由云端，处理过程将在本地边缘计算层完成。这无疑将大大提升处理效率，减轻云端的负荷。由于更加靠近用户，还可为用户提供更快的响应，将需求在边缘端解决。

1.2 QoS需求

QoS（Quality of Service，服务质量）指一个网络能够利用各种基础技术，为指定的网络通信提供更好的服务能力，是网络的一种安全机制，是用来解决网络延迟和阻塞等问题的一种技术。

大多数基于深度神经网络的应用通常是资源密集型和计算密集型的，同时要求实时处理，这对资源有限的物联网设备的 QoS 要求提出了很大的挑战。为了减轻终端设备的计算负担，边缘智能通过利用大量附近的服务器和基础设施来提供计算服务，物联网设备可以通过无线连接将计算任务转移到边缘，以增强其计算能力。

为了支持不同的应用，边缘服务器应该为卸载的任务提供不同的 QoS 保证。交互式应用，如自动驾驶和增强现实等，通常有严格的时效性要求；相比之下，一些耐延迟应用，如监控视频分析，更注重吞吐量的提高，以更好地支持计算量大的数据分析。因此，需要一个复杂的解决方案来调度资源有限的边缘服务器上的任务，以满足异构的 QoS 要求。如图 1 是该问题的抽象场景。

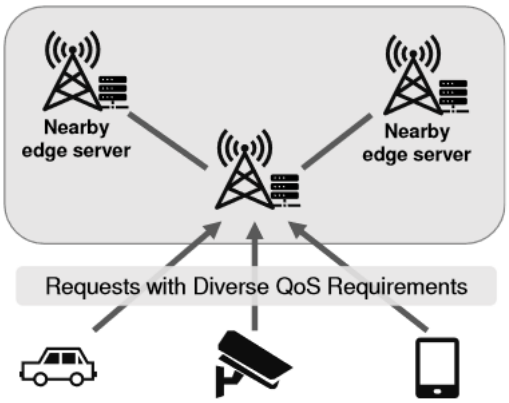

图 1　边缘服务器上异构 QoS 需求的任务场景

2 问题与挑战

2.1 时效性和吞吐量的矛盾

不同的 QoS 要求，如时效性和吞吐量，通常是相互矛盾的任务调度目标。以 DNN（Deep Neural Networks，深度神经网络）任务为例，提高系统吞吐量的常用方法是将一些 DNN 任务组合成一个批处理。批处理越大意味着边缘服务器的并行度越高，从而通过更高的计算资源利用率提高吞吐量。同时，批处理量大，执行延迟大，时效性不理想。为了在这两个目标之间进行权衡，单批并行和多批并行的方法陆续被提出[1]：

（1） 单批并行（SBP，Single Batch Parallelism）根据任务紧急情况和所需的计算资源等几个因素灵活地调整批大小，但只允许系统中同时运行单个批处理，在执行小批处理时可能会出现吞吐量损失。

（2） 多批并行（MBP，Multiple Batch Parallelism）允许多个批同时执行，小批负责紧急任务，以提供高响应性，而大批提供高系统吞吐量。但是，这些方法由于批次之间的干扰，在实时性方面容易导致 QoS 违反率较高；在系统级别，恶性资源争用通常发生在这些批之间；有可能大部分资源被非紧急任务占用，导致持续等待导致紧急任务响应能力低；同时，在调度层面，MBP 方法没有考虑异构的时效性需求，无法区分紧急任务和非紧急任务，导致紧急任务的截止日期违反率较高。

2.2 影响Qos的因素

在实际应用中，有一些因素会严重影响 QoS，比如动态系统负荷和突发负载。

（1） 动态系统负荷：由于业务数量的不断增加和工作负载的动态变化，分配给每个服务的计算资源在运行时也会动态变化。因此，在资源不足的情况下，紧急任务可能有很高的超时风险，这就导致 QoS 无法得到保障。

（2） 突发负载：由于边缘计算网络的计算资源往往不如云计算中心的计算资源充足，一旦边缘计算网络的负载突然增加，就会导致业务负载不平衡的问题[2]，即短时间内过多的请求会显著降低系统的 QoS。

3 研究现状分析

2021 年，Shida Lu 等[3]在实时人脸识别应用场景中，针对传统集中式云中心的不足，提出了云边缘融合的网络结构，它利用云强大的存储、查询和计算能力，为实时操作提供全面的查询和计算。基于网络架构，提出了一种分布式任务调度机制。提出了一种基于队列的任务调度算法，该算法考虑了数据传输时延和服务器负载，并进行了合理的任务分配策略。与云服务器相比，边缘服务器的处理能力仍然有限。因此，考虑到各服务器的负载和传输时延，采用多个不同地理分布的边缘服务器共同完成业务，将许多计算密集型任务合理地调度在每个

边缘服务器上。该调度系统是一个集群架构，由一个主服务器和多个从服务器组成。接收识别作业的边缘服务器充当主服务器，在调度任务的同时执行任务。其他边缘服务器充当从服务器，只负责执行任务。在每个边缘服务器中，维护一个等待队列 TQ 和一个备用队列 SQ。TQ 是边缘服务器等待执行的任务队列，SQ 是分配给边缘服务器但尚未传输输入数据的任务队列。此外，主机维护关于整个作业的信息，例如任务节点之间的依赖关系以及任务是否已被处理。

Shuiguang Deng 等[4]探讨了移动边缘计算环境下基于微服务的应用程序的部署问题，并提出了一种在资源约束和性能要求的情况下帮助优化应用程序部署成本的方法。论文介绍了移动边缘计算模型，重点介绍了在边缘服务发放系统中部署基于微服务的应用的场景。在此基础上，将服务器上的微服务实例建模为排队节点，并提出了在满足应用程序平均响应时间要求的情况下寻找成本较低的最优部署方案的方法。此外，还探讨了可能影响结果的因素，并为开发人员部署基于微服务的应用程序提供了一些指导。由于在得到给定时间段内位置感知请求到达率的情况下，该方法可以为应用程序生成部署方案，因此应用程序开发人员可以在准确预测请求到达率时动态更新部署方案。

在边缘智能普及的推动下，DNN 服务已广泛部署在边缘，对边缘服务器造成了巨大的性能压力。如何提高边缘 DNN 服务的 QoS 成为一个关键和具有挑战性的问题。2022 年 Ziyan Fu 等[1]针对边缘服务器上的 DNN 任务进行了研究。首先，论文对空间多路复用和时间多路复用两种情况下的批干扰进行了深入的实验研究，清楚地展示了批大小和批干扰对 QoS 的影响：当系统中只有一个批处理在执行时，批处理大小较大的批处理具有较高的系统吞吐量。但是，随着执行时间的增加，响应性往往会变差。当同时执行多个批处理时，采用时间复用的系统具有较好的响应能力，但系统吞吐量较弱。空间多路复用则正好相反。BATCH[6]针对无服务器平台提出了一种自适应批处理调度方法，但没有考虑 QoS 要求和批次之间的干扰。基于这些观察，论文提出了 Kalmia 的调度框架。Kalmia 的目标是降低截止时间违反率，实现高系统吞吐量。为了实现这一点，Kalmia 采用了一种基于抢占的调度方法，包括两个调度策略，首先是 CNT 调度策略——以高吞吐量为目标调度非紧急任务；然后是 CUT 调度策略——组织紧急任务抢占 CNT 执行以保证紧急任务的时效性。同时，Kalmia 可以通过轻量级预测模型适应动态的系统负载，必要时将部分任务卸载到附近的边缘服务器。Kalmia 具有很好的适应动态系统负载的能力，并且可以很好地适应附近边缘服务器的数量。

此外，为了防止边缘计算 QoS 较差，在突发负载时，需要将某些业务负载迁移到其他边缘服务器上，以降低这些业务请求的整体延迟。

Shuiguang Deng 等[2]重点研究了在边缘环境下卸载工作时的突发负载疏散问题，提出了一种有效的策略来解决边缘环境中的突发负载。为了降低延迟并保持边缘网络可接受的 QoS，提出在遇到突发负载情况时所采取的两阶段策略：通过调度阶段的最优路由搜索，将任务从发生突发负载的服务器尽快迁移到其他服务器。随后，在远程服务器和边缘服务器的协助下，通过所提出的并行结构在调度阶段以最高效率处理这些任务。具体来说，论文首先将边缘计算网络划分为三层，如图 2 所示，即迁移协同层、计算层和远程辅助层。

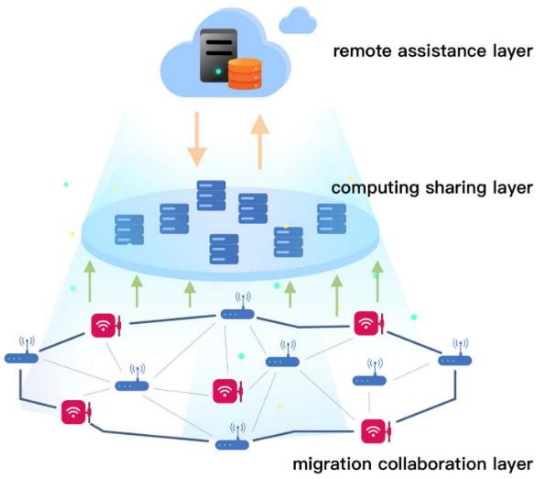

图 2　边缘计算网络的三层架构划分

论文提出了一种处理负荷疏散的快速疏散策略。这为边缘计算框架提供了一个实质性的扩展，使它们能够及时处理大量的流量负载。然后提供了在边缘网络资源有限时迁移突发负载的解决方案。并且在边缘服务器的计算共享层中提出了一种调度算法，使边缘环境中的任务并行处理。在不考虑所有任务处理时间的前提下，通过特定的调度来解决所提出的问题。最后在调度阶段提出了基于在线处理的算法，无需提前了解任务状态。利用边缘网络中边缘服务器的计算能力，在网络中空闲的计算资源下并行处理任务。

Qing Li 等[5]首先提出了统计计算模型和统计传输模型来量化统计 QoS 保证与任务卸载策略之间的相关性。然后将任务卸载问题表述为具有统计延迟约束的混合整数非线性规划问题，将统计延迟约束分别转化为 CPU 周期数约束和延迟指数约束。利用凸优化理论和吉布斯抽样方法，提出了一种为任务提供统计 QoS 保证的算法。实验证明，该算法高概率可以收敛到全局最优解。

4 调度框架 Kalmia

接下来以 Kalmia 为例，介绍最新的有关边缘服务器上 QoS 感知的任务调度方案研究。如图 3 所示，Kalmia 包含一个调度器、在线分析器和离线分析器、一个执行引擎和一个卸载引擎。

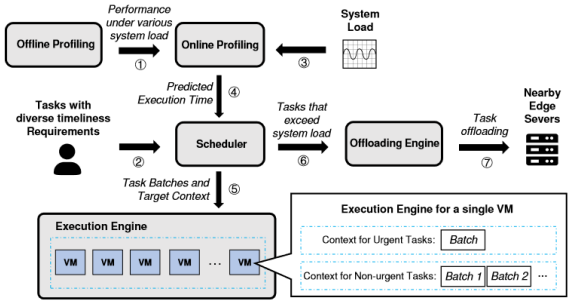

图 3　Kalmia 调度框架

Kalmia 的工作流程可以分为离线分析和在线调度两个阶段，具体流程如下：

（1）　首先模拟了几个不同级别的系统负载，并测量每个级别下的性能；

（2）　Kalmia 连续接收客户的任务。这些任务具有不同的时效性要求，调度器负责将多个任务组合成一个批处理执行；

（3）　在线分析根据历史数据获得系统负载；

（4）　根据来自离线分析的数据计算出给定批处理的执行时间；

（5）　通过利用 scheduler 给出的数据，执行引擎采用基于抢占的方法为紧急任务提供计算资源；

（6）　最后，在任务超出服务器能力的突发负载期间，卸载引擎将几个任务卸载到附近的边缘服务器，以降低截止时间违反率。

4.1　分析

在离线分析中，用不同系统负载条件和批处理大小下的性能数据训练一个回归模型。每个配置包括后台模型的数量和前台模型的大小，如图 4 所示，将每个配置运行 30 次，并从这些示例中获取执行时间的第 98 个百分位值。这确保了在在线分析期间，大多数任务可以在预测的执行时间内完成，从而保证了时效性。训练这个百分位值关于后台模型数目与前台模型大小的回归关系，得到一个回归模型。

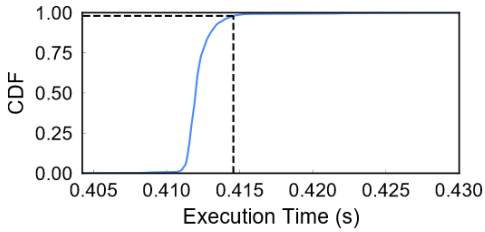

图 4　执行时间的累积分布函数

在在线分析期间，首先根据历史数据评估当前系统负载。此外，由于在离线分析期间在样本中取了执行时间的第 98 个百分位值，因此在在线分析期间使用实际执行时间来估计系统负载将导致系统负载被低估。为了解决这个问题，根据每个历史数据计算出一套系统负载估算值，并将其中的第 98 个百分位值作为最终的系统负载指标，确保大多数批可以在估计的执行时间内完成。

最后，当接收到调度器的预测执行时间的请求时，根据训练得到的回归关系来预测给定批处理的执行时间。

4.2　调度

Kalmia 将不断从用户设备接收到的任务放入优先级队列，任务按紧急程度排序（即任务截止日期）。调度器负责从优先级队列中选择任务，将它们组合成批处理，并将批处理发送到适当的上下文中。

4.3　执行引擎

将任务分为紧急任务和非紧急任务，执行引擎负责将它们分布到两个 CUDA 上下文中，分别是紧急任务上下文（context for urgent tasks，CUT）和非紧急任务上下文（context for non-urgent tasks，CNT）。为 CNT 开发了基于 MBP 的调度策略以提高系统吞吐量，并为 CUT 开发了基于 SBP 的调度策略以抢占 CNT 的执行并提供高响应性。

4.3.1　非紧急任务上下文

CNT 负责非紧急任务，以提高系统吞吐量为目标。它允许多个批处理同时执行，这些批处理通过空间复用共享计算资源。

（1） 首先根据任务截止日期的时间将任务分成几个块，并用 L 表示每个块的粒度。任务被划分成块，截止日期分别分布在[0, L)，[L, 2L)， [2L, 3L)，…。在 CNT 中，主要考虑截止日期大于 L 的任务。

（2） 根据在线分析所得到的数据来计算批大小，从而估计每个块中要完成的任务数量的上限，表示为 N。

（3） 接下来，Kalmia 决定哪些任务应该卸载到其他边缘服务器：首先计算系统在每个块中可以执行的任务数量的上限，如果一个块中的任务数量超过了这个上限，这些任务就被认为有违反时效性要求的风险。超过限制的部分任务将被卸载到其他边缘服务器。

（4） 然后，Kalmia 决定发送给执行引擎的下一批处理。如果队列中可用的非紧急任务超过上限 N，则只取 N 以内的任务进行组合；否则，表示当前任务队列中非紧急任务较少，在这种情况下，可以向批处理中添加一些紧急任务。

（5） 最后，Kalmia 将批处理发送到执行引擎。

其中，批大小需要保证时效性要求，即预测执行时间小于或等于批中最低的截止日期。论文中使用基于二进制搜索的策略来查找下一个批处理大小。

4.3.2 紧急任务上下文

CUT 负责紧急任务。它的目标是保证高响应性。在论文前期的观察环节中，已经总结了在相同环境下的不同任务批次之间会产生干扰，导致反应性较差，且批的响应性受其他上下文中批的影响小于同一上下文中批的影响。因此，使用专门的上下文来抢占 CNT，以提高紧急任务的响应能力。在 CUT 中，同一时间只允许执行一个批处理，以避免上下文内部的干扰。

与在 CNT 中使用固定长度（即 L）的块不同，在 CUT 中块的每个长度都基于任务的紧急程度。具体来说，首先找到最紧急的任务，并得到其截止时间，然后计算批大小。为了充分利用可用资源，这批任务的执行时间应该尽可能的大，但是为了保证时效性，它也受到截止时间的限制。最后，Kalmia 将这些任务发送给执行引擎。

对于要卸载的任务，首先计算一个批处理所花费的最小时间，也就是大小为 1 的批处理的执行时间。由于 Kalmia 最多允许一个批同时执行，因此 deadline 比批处理中最后一个任务 deadline 还晚的任务以及 deadline 比批处理中第一个任务的 deadline 加上最短执行时间还要早的任务均无法在期限内完成，它们将被卸载到其他边缘服务器。CUT 调度策略是在 CUT 空闲时触发的，即当系统开始启动或前一批处理完成时触发。

Kalmia 可以根据实时异构时效性需求和系统负载自适应地确定批处理大小和目标上下文，从而自动实现响应性和系统吞吐量之间的权衡。

4.4 卸载引擎

4.4.1 任务卸载

对于一个新的批处理，若任务无法在期限内完成，则被卸载到其他边缘服务器。

当调度器决定执行任务卸载到其他边缘服务器时，它将任务解除队列并将它们发送到卸载引擎。卸载引擎同时向附近的所有边缘服务器发送任务请求，然后将任务数据发送到接受该任务的第一个边缘服务器。

同时，Kalmia 考虑到这些任务的紧迫性，预先计算了在 CUT 中剩余要执行的批次，确定违反截止时间风险较高的任务，并将其卸载到其他服务器上。

4.4.2 任务接收

（1）对于非紧急任务，Kalmia 首先根据任务的截止时间找到该块，如果块大小没有达到上限，则接受该块。

（2）对于紧急任务，Kalmia 需要在给定的任务截止时间内检查计算资源是否可用。

5 未来研究方向

突发负载是任务调度中一个棘手的难题。在现实场景中，任务卸载可能受到许多复杂因素的影响，如网络状况和实时系统状态。

Kalmia 中的负载卸载并没有考虑服务器的选择问题，这是框架中的不足之处。Shuiguang Deng 等提出的算法兼顾了用户生成任务的效率和公平性，但考虑到在实践中任务的延迟值可能会被不准确估计的情况，需要设计更有效的策略来解决负载疏散问题。

因此，构思有效的不同边缘服务器之间的任务卸载调度策略，以进一步提高任务卸载性能，是未来可以研究的方向。

6 结论

本文研究了目前资源有限的边缘服务器上 QoS 感知的调度方法，探究了突发负载时的处理方案。为了满足异构的 QoS 要求，需要复杂的解决方案来调度资源有限的边缘服务器上的任务，而调度方案需要考虑的因素有很多，包括动态系统负荷、突发负载、并发进行的任务之间可能存在的互相干扰等。最后，以最新的研究成果为例介绍了 QoS 感知的任务调度框架 Kalmia，并对未来的研究方向做了简要的描述。

参 考 文 献

[1] Fu Z, Ren J, Zhang D, et al. Kalmia: A heterogeneous QoS-aware scheduling framework for DNN tasks on edge servers[C]//IEEE INFOCOM 2022-IEEE Conference on Computer Communications. London, United Kingdom, 2022: 780-789.

[2] Deng S, Zhang C, Li C, et al. Burst load evacuation based on dispatching and scheduling in distributed edge networks[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 32(8): 1918-1932.

[3] Lu S, Gu R, Jin H, et al. QoS-aware task scheduling in cloud-edge environment[J]. IEEE Access, 2021, 9: 56496-56505.

[4] Deng S, Xiang Z, Taheri J, et al. Optimal application deployment in resource constrained distributed edges[J]. IEEE Transactions on Mobile Computing, 2020, 20(5): 1907-1923.

[5] Li Q, Wang S, Zhou A, et al. QoS driven task offloading with statistical guarantee in mobile edge computing[J]. IEEE Transactions on Mobile Computing, 2020, 21(1): 278-290.

[6] Ali A, Pinciroli R, Yan F, et al. Batch: Machine learning inference serving on serverless platforms with adaptive batching[C]//SC20: International Conference for High Performance Computing, Networking, Storage and Analysis. Atlanta, GA, USA, 2020: 1-15.