华 中 科 技 大 学

研究生课程考试答题本

考生姓名 崔占普

考生学号 Y202202003

系、年级 研究生院 2022 级

类　　别 学 硕

考试科目 数据中心技术

考试日期 2022 年 12 月 20 日

边缘计算与云计算的资源调度优化问题研究报告

目录

摘要

近几十年来，新兴的移动应用和服务变得越来越需要资源和计算。便携式大小的移动设备无法提供这类服务。云计算一直是以可扩展的方式为移动服务提供可视化资源的核心计算技术。然而，不可预测的网络传输延迟使得云计算对于对时间敏感的移动服务来说不够有效，这就需要对底层计算平台进行重大更改。

随着物联网（IoT）的普及和无线网络的广泛渗透，对数据通信和计算的激增需求激增要求新兴的边缘计算范式。通过将位于云中的服务和功能移动到用户附近，边缘计算可以提供强大的通信、存储、网络和通信能力。边缘计算中的资源调度是边缘计算系统成功的关键，它引起了越来越多的研究兴趣。

一般来说，资源调度是指参与者用来有效地分配资源，并根据资源可用性实现参与者的目标的一组行动和方法。云计算的特点就在于 IT 资源的按需供求，边缘计算的资源存在于边缘网络中，通过这些资源提供了强大的可服务性，并可以完成任务。边缘网络中的资源可以分为三种类型，即通信资源、存储资源（也作为缓存资源）和计算资源。

本文作为课程综述报告，选定题目是边缘计算与云计算的资源调度优化问题研究报告，首先介绍了边缘计算以及云计算，并对其资源管理的重要性做了简单说明，对云计算与边缘计算的资源调度管理方面进行了讨论，介绍了现阶段的研究现状，并结合案例详细阐明两类计算的资源管理特点，在众多发展方向中介绍了边缘计算未来的分布式资源管理技术的使用以及云计算方向中机器学习的大规模应用。

1 研究原因

本人的论文研讨的内容是有关云计算下闲置资源的收集与分配，结合之前看过的几篇关于边缘计算资源管理的论文，重新选定方向，将综述报告的重点放在云计算与边缘计算在资源调度管理下的不同策略、不同选择以及不同发展方向上。在边缘计算逐渐补齐云计算短板，未来云边融合的趋势愈演愈烈的前提下，本篇综述报告尽量涵盖双方在资源调度下的内容。

自谷歌在 2008 年提出云计算的概念以来，云计算逐渐被接受并引入移动环境，突破了智能设备的资源限制，为用户提供了高要求的应用程序。云计算是一种经济有效的模型，它提供了丰富的应用程序和服务，同时使信息技术（IT）管理更容易访问，并更快地响应用户的需求。这些服务（计算、通信、存储和所有必要的服务）以一种简化的方式交付和实现：按需应变，无论用户的位置和智能设备的类型。

边缘计算是指允许在网络边缘、代表云服务的下游数据和代表物联网服务的上游数据进行计算的启用技术。边缘计算将原本位于云中的服务和功能移动到用户附近，这将云计算平台和网络集成起来，在网络的边缘提供强大的计算、存储、网络和通信能力。

数据中心满足 5G 超低时延、大带宽和海量连接的需求，5G 三大应用场景中，超低时延、大带宽和海量连接均需要边缘计算。边缘计算具有聚焦实时、短周期数据的分析能力，能更好地支撑本地业务的实时智能化处理与执行。在万物互联的 5G 智能化时代，如果仅采用中心云计算模式（中心云），已经不能满足高效地处理网络边缘端所产生的海量数据。因此分布在网络边缘端，提供实时数据处理、分析决策的小规模云数据中心的边缘云和边缘计算服务器显得尤其重要。

2 背景介绍

这一部分将简要介绍云计算与边缘计算的各自概念与关系，并对资源管理的重要性进行简要介绍。

2.1 边缘计算与云计算的关系

随着物联网规模的快速增长，集中式的数据存储、处理模式将面临难解的瓶颈和压力，此时在靠近数据产生的网络边缘提供数据处理的能力和服务，将是推动 ICT 产业发展的下一

个重要驱动力。边缘计算（Edge Computing）的概念由此而生。边缘计算成为新兴万物互联应用的支撑平台，目前已是大势所趋。

在有了云计算的同时，为什么还需要边缘计算？主要存在以下几点原因：

1） 网络带宽与计算吞吐量均成为云计算的性能瓶颈；

2） 物联网时代数据量激增，对数据安全提出更高的要求；

3） 终端设备产生海量“小数据”，需要实时处理。

如果说“云计算”所能实现的是大而全的话，那么“边缘计算”更多则是“小而美”，从数据源头入手，以“实时、快捷”的方式完成与“云计算”的应用互补。比较两者，云计算聚焦非实时、长周期数据的大数据分析，能够为业务决策支撑提供依据；边缘计算则聚焦实时、短周期数据的分析，能更好地支撑本地业务的实时智能化处理与执行。因此边云协同的概念被提出，图 1 就是边云协同总体的参考架构。

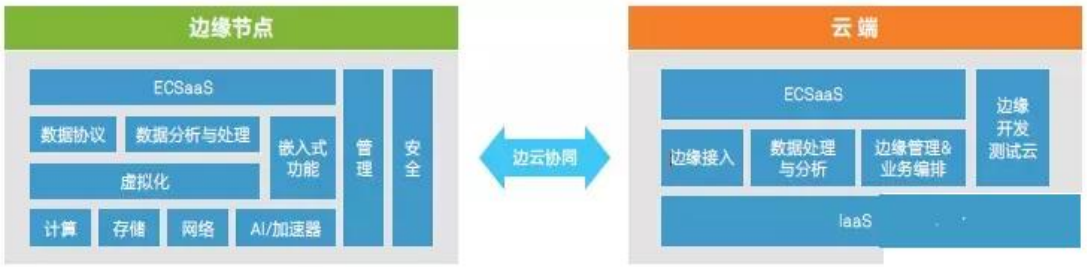

图 1 边云协同总体参考架构

2.2 资源管理优化的重要性

一般来说，资源调度是指参与者用来有效地分配资源，并根据资源可用性实现参与者的目标的一组行动和方法。

云计算的特点就在于 IT 资源的按需供求，边缘计算的资源存在于边缘网络中，通过这些资源提供了强大的可服务性，并可以完成任务。边缘网络中的资源可以分为三种类型，即通信资源、存储资源（也作为缓存资源）和计算资源。

云计算创造了一个消费者使用软件和 IT 基础设施的环境，为计算作为第五个实用程序[1]的出现铺平了道路。数据中心的资源管理在云计算中仍然是一个重要的问题，而且它直接依赖于应用程序的工作负载。应用程序连接到传统云计算环境中的特定物理服务器，如数据中心，因此这些服务器经常被过度配置，以处理与最大工作负载[2]相关的问题。由于资源和建筑空间的浪费，数据中心在资源管理方面的运作费用高昂。另一方面，虚拟化技术已经证

明，它可以使数据中心更容易处理。该技术提供了多种好处，包括服务器整合和更高的服务器利用率。谷歌、微软和亚马逊等大型　巨头拥有大量的资源管理。

在这些数据中心中，一个服务器主机被分配了多个具有不同工作负载类型和数量的虚拟机。这种可变和不可预测的工作负载可能导致服务器被过度利用和未充分利用，从而导致分配给特定主机服务器上的 vm 的资源利用的不平衡。这可能会导致包括不一致的服务质量（QoS）、不平衡的能源使用和服务水平协议（SLA）违规等问题。根据一项关于不平衡工作负载的调查，平均 CPU 和内存利用率分别为 17.76%和 77.93%，谷歌数据中心的一项类似研究发现，谷歌集群的 CPU 和内存利用率分别不能超过 60%和 50%。

由于工作量不平衡，数据中心的生产力受到影响，导致能源消耗增加。它与数据中心的运营成本和财务损失成正比。这种过度的能源消耗对碳足迹有直接的影响，这应该被减少，因为一个理想的机器吸收了超过最大能源消耗的一半。根据能源信息管理局（EIA）的一项调查，2015 年数据中心消耗了约 35 小时（德拉瓦小时）的能源，预计到 2040 年这一数字将上升到 95 小时。

3 研究现状

3.1 云计算下的资源调度（以无服务器计算为例）

无服务器计算是云计算的一种，其自动化了细粒度的资源扩展，并简化了具有无状态函数的在线服务的开发和部署。但是，由于各种函数类型、依赖关系和输入大小，用户分配适当的资源仍然是非常简单的。资源分配的错误配置导致功能供应不足或过度，导致资源利用率持续低。

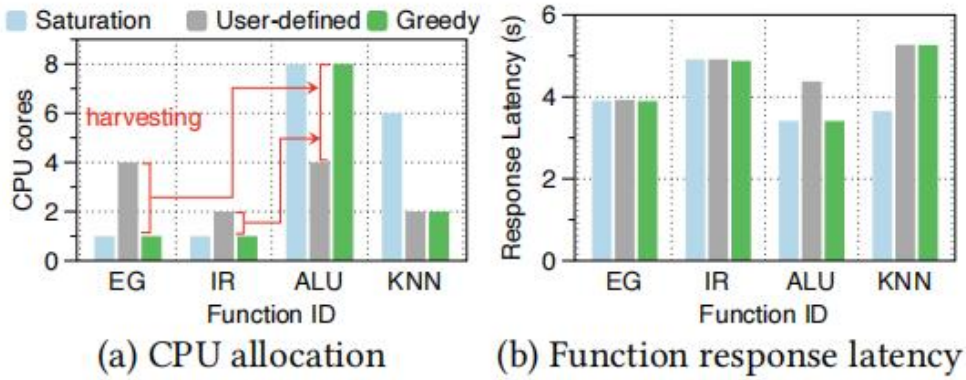

图 2 现有资源管理器对比图

比较了默认资源管理器（固定 RM）和（Greedy RM）在执行四个无服务器函数时所实现的函数响应延迟。固定 RM 只是简单地接受并应用由用户预定义的固定资源分配。Greedy RM 从过度配置的函数中动态获取 CPU 核，并根据从函数最近的资源利用中得到的估计功能饱和点，以先到先得的方式将获取的 CPU 核分配给配置不足的函数。

3.2 边缘计算下的资源调度

云计算的资源调度框架我们在案例中进行详细介绍，这一部分重点介绍边缘计算下的资源调度。我们可以清楚地将云服务厂商提供的 IT 资源（内存和 CPU）的调度优化是提高生产力的表现，那么在关注边缘计算的时候，我们肯定要提出一个重要问题，即边缘计算中为什么进行资源调度。

虽然边缘计算通过提供强大的计算、存储和通信能力，极大地增强了边缘网络的可服务能力，但它也需要从三个角度采取适当的资源调度策略。即从用户（来自不同应用场景的数据可能有不同的服务需求），服务提供商（以最小的成本获得最大的收入）以及边缘网络（有效地利用分散的资源）。

下图展示了边缘计算中的资源调度体系结构，包括事物层（即用户层）、边缘层和云层。这种架构的功能是为了说明构成边缘计算系统的组件之间的关系。事物层，也被称为用户层，由各种终端设备组成；边缘层，边缘层作为三层体系结构的核心，是介于事物层和云层之间的中间层。从硬件组成的角度来看，边缘层由各种网络和计算设备组成，如蜂窝通信塔、边缘服务器等组成；云层，云层由现有的云计算基础设施组成，如计算单元、存储单元和微数据中心。

云层是最强大的数据处理和存储中心。虽然边缘层中的 ESs 可以处理大量的数据，以减少延迟和能源消耗，但边缘计算范式仍然需要云的计算能力和高容量的存储基础设施来处理一些棘手的任务和全局信息。

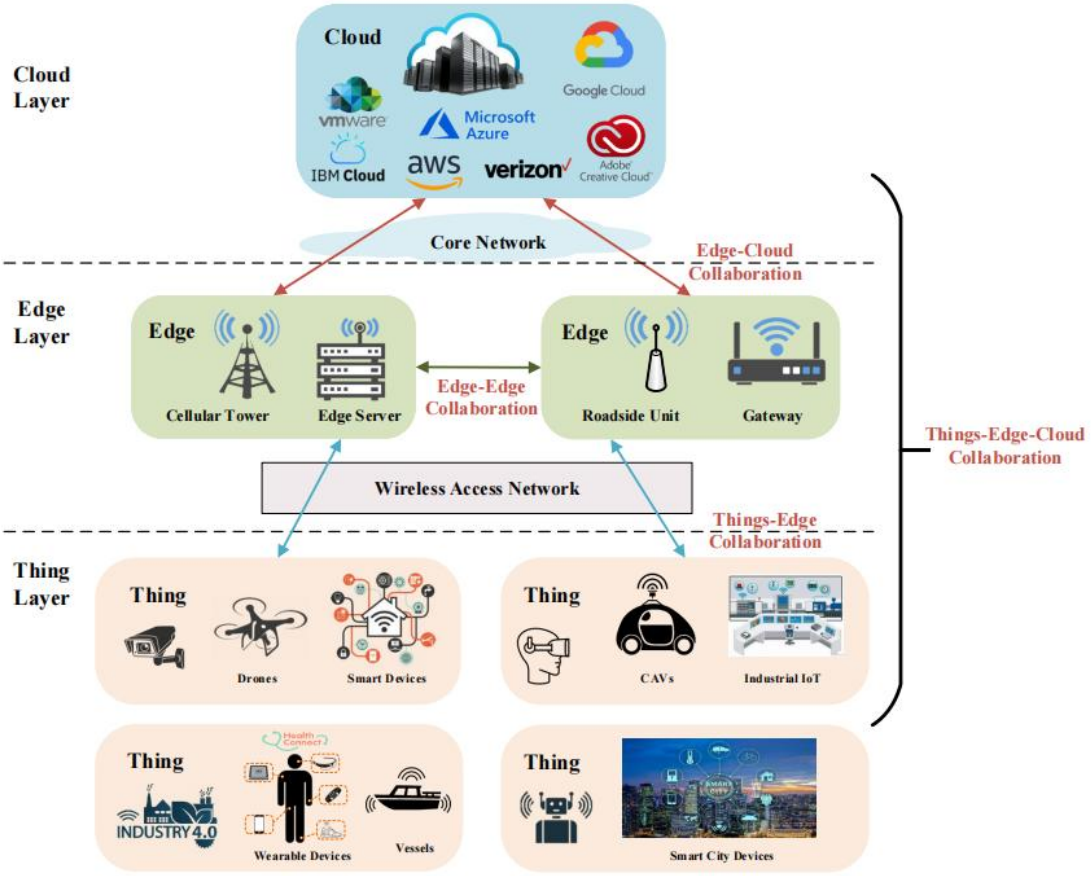

图 3 边缘计算中的资源调度体系结构

4 案例分析

4.1 无服务器计算优化案例

4.1.1 Freyr 架构与设计流程

这里主要介绍一个优化的资源管理器 Freyr，对其的架构与以及设计过程进行简单介绍，最后进行简单评价。

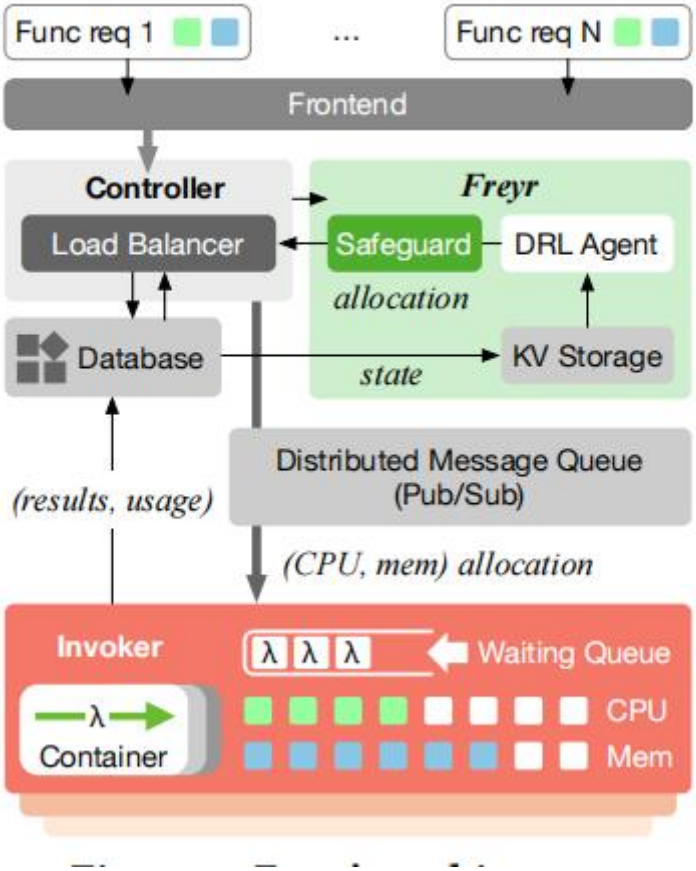

图 4 Freyr 架构图

首先并发函数请求到达前端，以使用用户定义的资源分配调用特定的函数。控制器允许函数请求，注册它们的配置，并将它们调度给调用器。在执行函数之前，Freyr 从无服务器平台数据库和做出资源收集和重新分配的决策。控制器指示调用者在执行函数调用时强制执行这些决策。

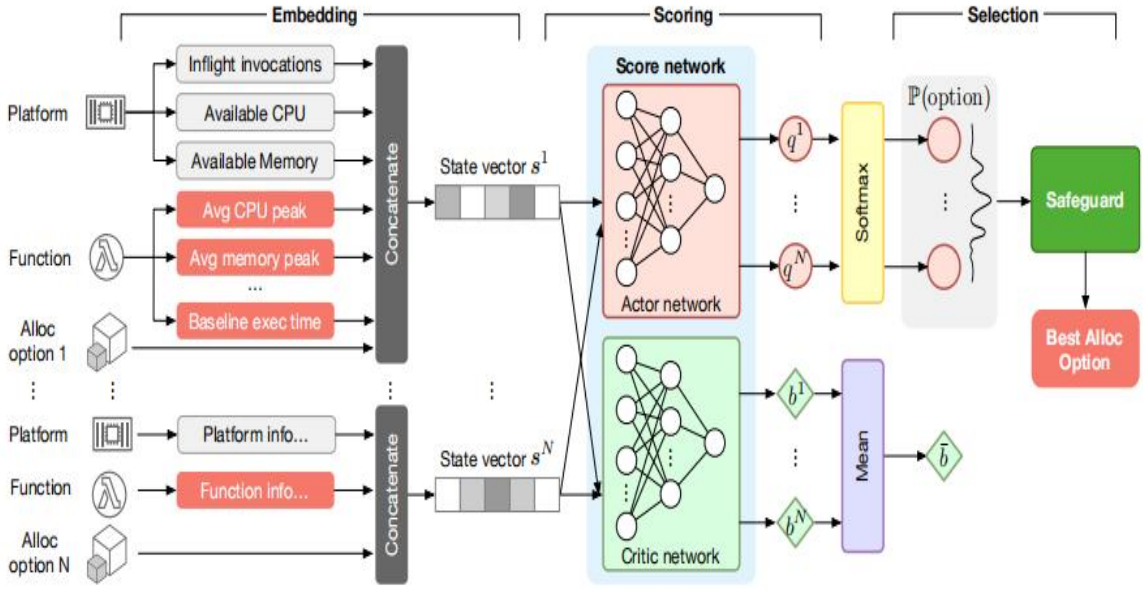

图 5　Freyr 工作流

Freyr 的工作流有三个阶段，装配阶段、评分阶段、选择阶段。

在第一阶段会收集信息，当为函数调用分配资源时，Freyr 从两个级别收集信息：平台级别和功能级别，具体来说，对于该平台，Freyr 捕获了系统中剩余的调用数量（即 inflight_request_num）、可用的 CPU 内核和可用的内存。对于传入的函数，Freyr 查询调用历史记录具有用户请求资源的平均 CPU 峰值、平均内存峰值、平均到达时间（IAT）、平均执行时间和执行时间的函数。一旦收集到这些信息，Freyr 就会用一个潜在的资源分配选项来封装它们。更准确地说，我们将信息和潜在的配置选项一起嵌入到一个平面状态向量中，作为 Freyr 代理的输入。

第二阶段我们采用下图所示的奖励函数进行评分，选出 E 值最大的一套选择分配方案，对其资源需求进行排序。

$$\mathbb{E}\left[\sum_{i=1}^{T}\gamma^{t-1}\left(-\sum_{f\in S|_{t_{i-1}}^{t_i}}\frac{e^i}{e^b}+R_{(slowdown<1)}-R_{(slowdown>1)}\right)\right].$$

图 6　奖励函数表达式

第三阶段最重要的就是保护算法，即要保证不能经过资源管理后某个任务的效率出现极大的下降。保护算法如下。

Algorithm 1: Safeguard mechanism atop *Freyr*.

```
1  while request_queue.notEmpty do
2      function_id ← request_queue.dequeue()
3      calibrate_baseline ← False
4      last_request ← QueryRequestHistory(function_id)
5      if last_request == None then
           /* Trigger safeguard */
6          range ← [user_defined]
7          calibrate_baseline ← True
8      else
9          threshold ← 0.8
10         last_alloc ← last_request.alloc
11         last_peak ← last_request.peak
12         recent_peak ← GetRecentPeak(function_id)
13         if last_peak < user_defined then
               /* Over-provisioned */
14             if last_peak / last_alloc ≥ threshold then
                   /* Trigger safeguard */
15                 range ← [user_defined]
16                 calibrate_baseline ← True
17             else
18                 range ← [recent_peak + 1, user_defined]
19             end
20         else
               /* Under-provisioned */
21             range ← [recent_peak + 1, max_per_function]
22         end
23     end
24     alloc_option ← Freyr(function_id, range)
25     Invoke(function_id, alloc_option, calibrate_baseline)
26 end
```

图 7 保护算法

第 5-7 行，之前没有调用时，Freyr 触发保护措施以获取资源使用，并校准方程；第 10-12 行，对于进一步的调用，Freyr 查询函数的历史，并轮询使用峰值、最后一次调用的分配以及自上次校准以来的峰值；第 13 行检查函数的当前状态，即配置过度或配置不足；第 14 行，假设资源使用量低于用户请求级别的 80%的函数被过度配置。对于过度配置（收获的）函数，Freyr 然后检查最后一次调用的使用峰值。如果使用峰值接近分配的 80%，我们怀疑可能会出现负载峰值，这可能比当前分配使用更多的资源。这将触发安全保护调用和基线重

新校准；第 21 行，对于配置不足的功能，允许 Freyr 从最近的峰值加上一个单元中选择到最大可用级别。

4.1.2 Freyr 评价

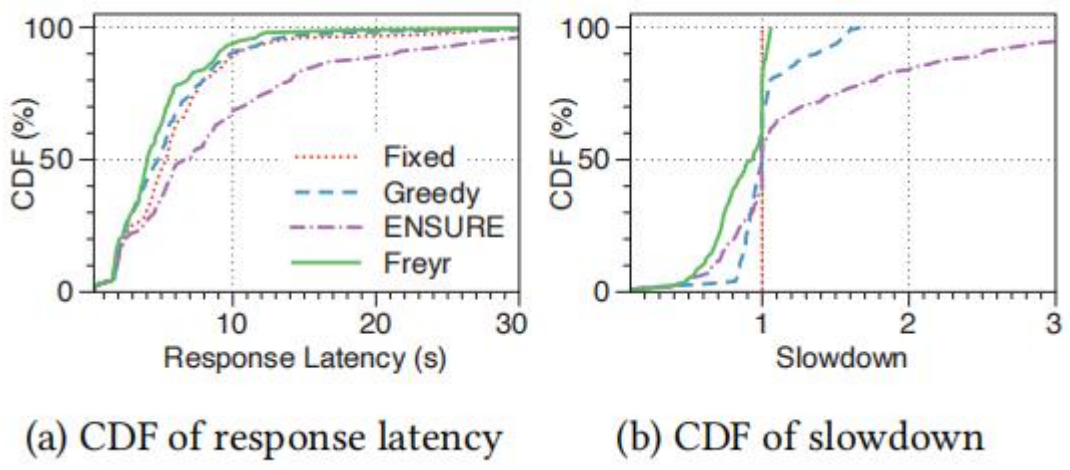

图 8 对比图

(a)显示了测试工作负载的函数响应延迟的 CDF。Freyr 的 99%功能响应延迟不到 19 秒，而固定 RM、贪婪 RM 和确保分别为 28、25 和 38 秒。

(b)显示了测试工作负载减缓的分布函数。Freyr 保持所有调用的 99%减速在 1.06 以下，而 Greedy 和 ENSURE 分别为 1.58 和 4.5。由于固定 RM 不调整资源，所有百分位的放缓保持 1.0。

在 OpenWhisk 集群上的实验结果表明，Freyr 优于其他 rm。Freyr 从 38.8%的功能调用中获取空闲资源，并加速了 39.2%的调用。与 OpenWhisk 中的默认 RM 相比，在相同的测试工作负载下，将 99%响应延迟减少了 32.1%。

4.2 基于区块链的 BCEdge 资源管理案例

如下图所示，BCEdge 包括以下四个步骤：注册、设施选择、任务处理和支付。在第一步中，每个设施所有者或移动用户都需要注册到系统。其次，移动用户提交一个包含资源需求的请求，并根据一些规则选择最优设施。一旦选择了一个设施，资源就会按照声明的方式进行分配。当数据存储在移动用户的终端上时，待处理的数据可以根据 D2D 定向传输到所选择的设施。完成任务后，生成一个支付交易并与所有设施共享，移动用户将向设施所有者

支付和奖励。BCEdge 可以减轻边缘云的负载，并通过集成的服务质量控制来奖励资源所有者。此外，它使每个资源所有者的贡献在系统中完全可见和透明。

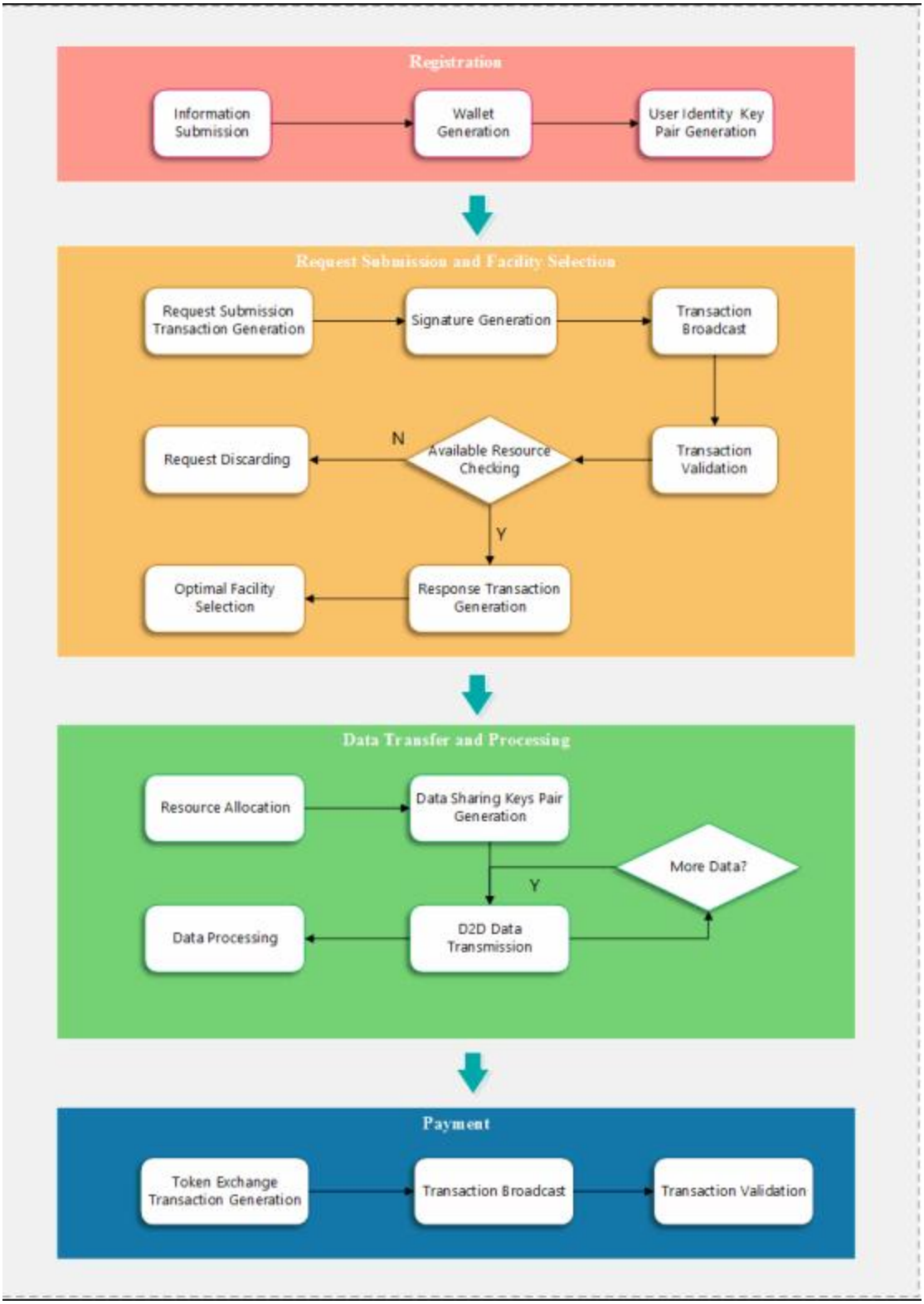

图 9　BCEdge 的工作流

注册部分，设施所有者或移动用户 U 注册后，为所有者或用户生成用户身份公/私钥对，

表示为（U. PKK，U. PRK）。每个用户或设施所有者都有一个钱包，用来存储与该用户相关的所有信息。当该用户注册时，将为该用户初始化该钱包。该钱包包含以下内容： (1)签名或识别交易所需的所有密钥对，以及(2)与用户相关的所有交易的信息。在最佳设备选择阶段或交易生成阶段，可以参考钱包来获取信息。

请求提交和设施选择阶段，当移动用户需要计算资源进行数据处理时，将生成请求提交事务。这些事务包括以下信息：计算资源需求，如内存大小、CPU 周期、预期占用时间和用户签名。首先，对事务的内容应用一个加密哈希函数。然后，使用私钥对生成的哈希值进行数字签名。

数据处理阶段，在添加包含响应事务的块后，设施所有者将分配所需的资源。由移动用户生成一个数据共享密钥对，用于数据传输。然后，通过使用 D2D 数据包将数据传输到选定的设施。数据包包含使用数据共享公钥和移动用户签名加密的数据。当设施所有者收到数据包时，设施所有者首先检查内容是否已被更改。最后，对数据进行解密以进行处理。

支付阶段在任务完成后，将生成一个令牌交换交易，以奖励或惩罚计算提供商。如果任务在没有违反 SLA 的情况下完成，则令牌将从请求者转移到计算提供者作为奖励。否则，计算提供程序将受到惩罚，并且令牌将从计算提供程序转移到请求者。

5 未来方向

目前云计算与边缘计算发展趋势迅猛，且云边协同的高效性与突出优势正在显现，那么他们未来发展方向也是多种多样，这里各选取一个较为热门实用的发展方向进行介绍。分别是边缘计算中的分布式资源调度方式（DRS），以及云计算的资源调度与机器学习（ML）的结合。

5.1 边缘计算与分布式资源调度

边缘计算中的 DRS 是指对边缘计算环境中可用的不同资源进行分布式调度，以优化系统和应用程序特定的目标。

边缘计算中的资源可以包括计算资源（CPU/GPU、内存等），存储资源（磁盘、闪存等）、网络资源（带宽、无线信道等）、甚至是通过组合多个资源而提供的服务。调度意味着在考虑约束条件的同时决定何时使用哪些资源。

框架如下图所示，框架由基础设施层、资源管理层、服务层和应用层四层组成。

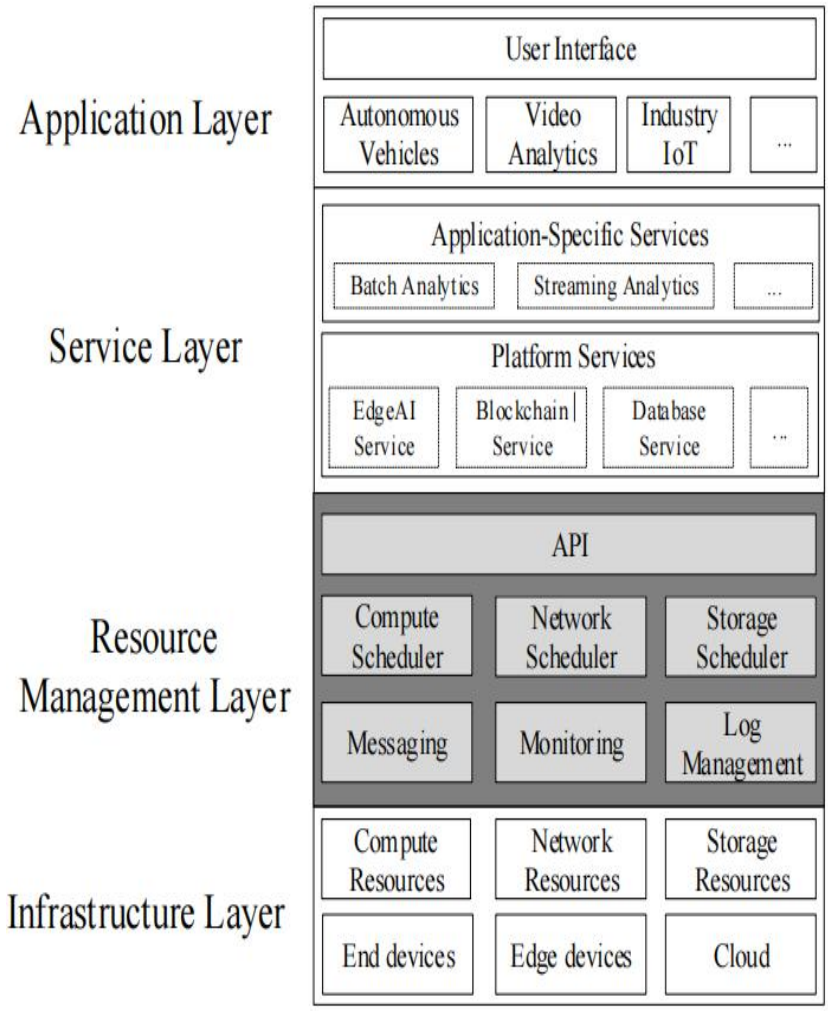

图 10 边缘计算架构图

DRS 意味着不同的设备可以是自主的，并且可以独立地对资源进行调度决策。在网状网络结构中，圆内有不同的边缘设备。每个边缘设备具有资源管理层中的所有组件，可以独立地进行调度决策。监控：负责监控资源的可用性和可用性；日志管理：负责记录调度决策和资源消耗；调度程序（计算、网络、存储等）：负责安排不同的资源；API：负责与框架中的上层进行接口，包括用户界面。

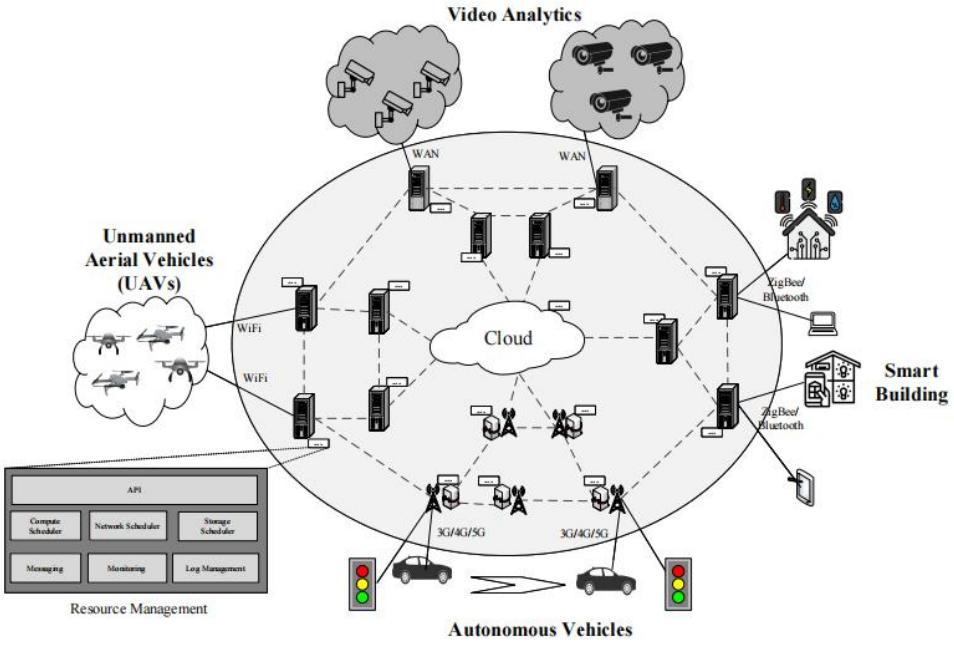

图 11 边缘计算中 DRS 的示例

5.1.1 边缘计算的 DRS 的场景应用

边缘计算的 DRS 有许多场景应用，例如自动驾驶、实时视频处理以及工业物联网。

在自动驾驶场景中可以为自动驾驶汽车确定的不同用例需要支持许多需求，包括高的和受限的移动性、实时处理、可靠性、数据的空间和时间依赖性等，使用集中式资源调度来支持这些需求具有挑战性。决定避障的车辆不能使用一小时前来自不同路段的数据。不能使用集中式控制器对车辆进行资源调度决策，如果需要从所有车辆发送数据，就会导致通信瓶颈和严重的延迟。此外，车辆通常有不同的车主，他们可能由于隐私问题而不愿意将数据共享给集中控制器。

在实时处理场景中，实时视频分析需要支持关键应用程序的实时处理，如 AR/VR、车辆网络等。视频分析涉及到具有高带宽和计算要求的流式数据。这些需求使得使用集中式控制器进行处理具有挑战性。例如，在警察的应用程序中，实时视频分析可以专门用于在监视区域内实时检测可疑的罪犯。警察还可以使用实时分析，通过全市的视频监控来寻找失踪儿童。消防应用程序可以使用实时视频分析来监控森林火灾或火灾期间的紧急建筑疏散。

工业物联网，不同行业物联网应用有一些共同的基本要求，包括高可伸缩性、实时处理、自动控制、可靠性、灵活性、协作、安全和隐私等。这些问题是使用集中式资源调度解决方案所难以解决的。

5.1.2 边缘计算中的 DRS 的发展方向

未来工作的一个主要机会是提出集成平台，考虑对边缘计算中不同类型资源的分布式调度。需要考虑的一个问题是为集成平台设计体系结构。设计接口，使体系结构中的不同组件之间以及与应用程序用户之间的交互。集成的平台还应该包括中间件，以便为应用程序用户提供一个抽象。

构建模拟工具包，可以专门为 DRS 解决方案提供基准模型和解决方案。这将有助于研究人员对所提出的解决方案进行更容易和更公平的比较。未来的工作还可以考虑提供混合测试平台。

联合优化，许多现有的 DRS 工作提供了一个管理个人资源的解决方案。很少有工作探索计算卸载、缓存或资源分配问题的联合优化。

5.2 云计算资源调度与机器学习

传统上，资源管理是通过静态策略来完成的，这在各种动态场景中施加了一定的限制，促使云服务提供商采用数据驱动的、基于机器学习的方法。机器学习正被用于处理各种资源管理任务，包括工作负载估计、任务调度、虚拟机整合、资源优化和能量优化等。

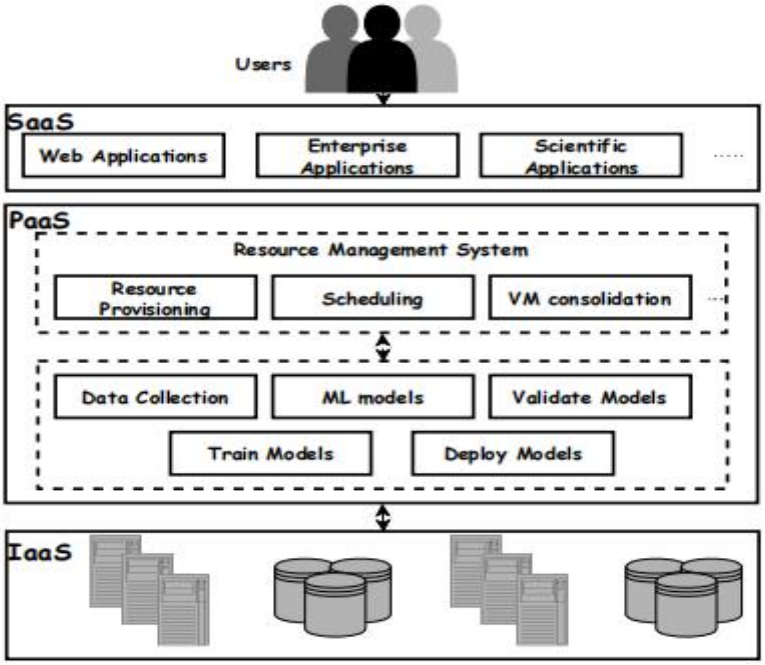

图 12 使用机器学习的云计算范式的组件

基于建模目标和问题，机器学习算法被分为监督学习、半监督学习（SSL）、无监督学习和强化学习（RL）。无监督学习被归类为聚类和降维，而有监督学习被归类为分类问题（如句子分类、图像分类等）和回归问题。

所有的机器学习算法都是优化问题，其目标是评估一个目标函数的极值。模型和逻辑目标函数的发展是机器学习方法的第一步。所确定的目标函数通常采用适当的数值优化方法来求解优化问题。

上面 4.1 提到的 Freyr 就是采用深度学习的资源管理框架。

5.2.1 资源调度出现的问题

虚拟机整合中的多种资源使用情况。虚拟机整合方法试图在更少的主机上整合更多的虚拟机，以便关闭剩余的主机并节省能源。大多数研究人员使用当前的 CPU 利用率来确定主机在这个过程中是否过载。这可能会导致不必要的虚拟机迁移和主机电源模式转换，从而降低整合过程的效率。用于迁移虚拟机的目标主机是 CPU 利用率最高的主机，但由于缺乏未来的估计，这可能会导致过度利用。因此，未来的资源利用估计可以解决这个问题。除了 CPU 的使用外，其他资源的消耗，如内存和磁盘，也可能会导致主机超载，使整合过程变得困难和具有挑战性。

错误的主机超载检测。当前资源利用率预测会导致不可靠的超载主机检测，特别是在当前资源利用率超过阈值的情况下。在决定是否应该迁移分配给该主机的虚拟机时出现了挑战，因为负载在很短的时间后迅速减少，从而导致错误的热检测点，即错误的过载主机检测。但是，当负载退化的持续时间足够大时，则需要迁移虚拟机，以避免过度使用。这种虚拟机整合机制对资源管理系统提出了一个独特的挑战，以避免不必要的虚拟机迁移开销。

时间序列预测数据，现代数据中心的工作负载遵循一个时间序列模式。因此，时间序列预测的模型应该根据历史数据进行训练，因为它假定未来的趋势将与以前看到的相同。然而，数据中心经历了非常非线性的工作负载变化，这就是为什么经常出现新的趋势，使模型难以精确地学习。由于缺乏适合于所有类型的时间序列预测数据的单一模型。此外，大多数时间序列预测的集成模型都是基于固定预测器的集合，无论是同质的还是异构的，这使得模型难以学习时间序列预测中的模式变化。

5.2.2 未来发展方向

性能和在线分析。智能资源管理系统的效率由多种因素决定，包括预测模型的准确性和时间复杂度。谷歌、微软、亚马逊等大公司负责管理极其复杂的数据中心，工作负载范围广泛。因此，在虚拟机存在如此高可变或非线性的工作负荷下，通过使用更复杂的 ML 和 DL 模式，更准确地估计之前的工作量是未来的研究方向。此外，算法的时间复杂度是根据运行输入代码所需的时间来衡量其性能的。因此，该算法应该被设计成在时间复杂度方面尽可能的简单。

时间序列预测数据开发针对云时间序列工作负载数据中任何类型的数据集的通用集成框架是未来的研究方向。一般来说，深度学习（DL）是一个快速扩展和广泛的研究领域，涉及到新的架构。然而，研究人员永远不确定他们需要适应哪种方法。使用全局神经网络模型在某些时间序列中容易出现离群误差。因此，必须以层次模型的形式发展出同时包含个体时间序列的全局和局部参数的新模型。这些模型可以与集成相结合，用相同的数据集训练多个模型。此外，cnn 长期以来一直被用于图像处理，

6 总结

在本文中，我们首先介绍了边缘计算以及云计算，并对其资源管理的重要性做了简单说明，对云计算与边缘计算的资源调度管理方面进行了讨论，介绍了现阶段的研究现状，并结合案例详细阐明两类计算的资源管理特点，在众多发展方向中介绍了边缘计算未来的分布式资源管理技术 的使用以及云计算方向中机器学习的大规模应用。

本篇综述只能算管中窥豹，在查资料的过程中也发现自己目前对云计算与边缘计算哪怕是资源管理这一个方向都是知之甚少，各种绝妙的 idea 以及问题的解决方案都让我受益匪浅，文献里提到的未来发展方向也让我心潮澎湃，深度学习、机器学习、区块链等等被作为工具，切实提高了资源利用效率，降低了成本。

7 参考文献

[1]Quyuan Luo, Shihong Hu, Changle Li, Senior Member, Resource Scheduling in Edge Computing: A Survey ,IEEE Communications Surveys & Tutorials,2021

[2] A. Zhou, Q. Sun, J. Li, Bcedge: Blockchain-based resource management in d2d-assisted mobile edge computing, Software: Practice and Experience,2021

[3]Hanfei Yu, Hao Wang, Jian Li, Xu Yuan, and Seung-Jong Park. 2022. Accelerating Serverless Computing by Harvesting Idle Resources. In Proceedings of the ACM Web Conference 2022 (WWW ’22), April 25–29, 2022, Virtual Event, Lyon, France. ACM, New York, NY, USA, 11 pages.

[4]Tahseen Khana , Wenhong Tiana , Rajkumar Buyyab,Machine Learning (ML)-Centric Resource Management in Cloud Computing: A Review and Future Directions,Journal of Network and Computer Applications Volume 204, August 2022, 103405

[5]Yuvraj Sahni,Lei Yang,Distributed resource scheduling in edge computing: Problems, solutions, and opportunities ,Computer Networks,2022

[6] Yuchang Mo , Senior Member, IEEE and Liudong Xing , Senior Member,Effificient Analysis of Resource Availability for Cloud Computing Systems to Reduce SLA Violations,IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 19, NO. 6, NOVEMBER/DECEMBER 2022