***Task scheduling algorithm in edge computing:*** ***Status and prospects***

# 摘要

随着万物互联时代的到来，网络边缘设备产生的数据量快速增加，带来了更高的数据传输带宽 需求，同时，新型应用也对数据处理的实时性提出了更高要求，传统云计算模型已经无法有效应对，因 此，边缘计算应运而生．边缘计算的基本理念是将计算任务在接近数据源的计算资源上运行，可以有效 减小计算系统的延迟，减少数据传输带宽，缓解云计算中心压力，提高可用性，并能够保护数据安全和隐私．得益于这些优势，边缘计算从２０１４年以来迅速发展．在云计算场景下，任务调度的一般策略是将计算密集型任务迁移到资源充足的计算节点上执行. 但是在边缘计算场景下，边缘设备产生的海量数据无法通过现有的带宽资源传输到云计算中心进行集中式计算，且不同边缘设备的计算、存储能力均不 相同，因此，边缘计算系统需要根据任务类型和边缘设备的计算能力进行动态调度.本文首先介绍了边缘计算中任务调度的研究意义与背景，而后对任务调度简要概括说明，最后总结了当前对于任务调度算法的一些不同研究。

关键词：边缘计算、任务调度、延迟感知、蚁群算法

# 研究背景

随着物联网的普及，物联网终端设备所产生的数据量越来越庞大，为了解决

这些数据在传输、计算和存储过程中产生的计算负载和数据传输带宽不足的问题，

科研人员开始在终端设备上增加数据处理的功能，具有代表性的是移动边缘计算

（mobile edge computing，MEC）,雾计算（fog computing）和海云计算（cloud-sea

computing）[4]。

*万物互联时代的到来和智能应用的快速发展使得终端设备的数据量呈现爆发式增长，终端设备硬件的局限性和用户的移动性可能导致服务在稳定性和连续性方面的问题。边缘计算丰富了终端设备的计算和存储资源，提高了终端设备任务的处理效率。边缘计算的资源比终端设备上的更充足，但是计算、存储等支撑任务处理的资源仍然是有限的，合理的资源分配可以降低终端任务的处理时间，保证服务的稳定性和连续性.*

*随着万物互联时代的到来和应用程序的智能化发展，终端设备的数量快速增长，思科预测，到２０２３年全球终端设备的数量可能远远超过人口数量⑴。由于任务和数据的快速增长以及终端设备的移动性和资源的有限性，复杂的计算和分析任务难以在终端设备上快速执行。边缘计算的物理位置在靠近终端设备或者用户的地方，可以丰富终端设备的计算和存储资源，提高待处理任务的实时性和高效性。用户是终端设备的使用者，同时也是边缘计算服务提供商的客户和服务订阅者。边缘计算服务器通常由服务提供商部署和管理，可以作为一个小型的数据处理中心，通过网络资源、计算资源、存储资源来处理终端设备卸载的任务，提高任务的处理效率，降低时延敏感型任务的处理时间以及终端设备的能量消耗。与传统的云计算相比，边缘计算服务器中支撑任务传输和处理的计算、存储等资源是有限的，并且用户的终端设备处于移动状态，如果资源未能得到合理分配和利用，可能导致终端任务的处理时间长、效率低的问题，影响用户的服务体验、用户和服务提供商的效益平衡、服务质量等。在这种“端－边”*

伴随着移动互联网、物联网的等技术领域的飞速发展，云计算技术也得到了长足的发展，来越多的应用部署在云端服务器上，为低成本的终端设备提供服务。在“万物互联”的智能时代，越来越多的终端设备将接入互联网，云计算网络正在承受着越来越大的压力。根据思科公司的预测，2020 年将有超过 500 亿的设备接入网络，这意味着将有海量数据通过互联网发送到云计算中心。面对海量数据的处理，云计算正在承受着巨大的压力。物联网技术发展面临着瓶颈，亟需一项能提供低延迟的计算服务。边缘计算技术作为云计算技术的补充，通过在靠近终端设备的网络边缘部署具有计算能力的设备来提供计算服务，很好的弥补了云计算的不足。边缘计算的理念是将云计算扩展到网络边缘处理，使从边缘终端获取的数据能够在接近用户终端的地方完成计算，同时，不需要长期存储的数据也不必传输到云服务中心备份，相对减少了网络带宽的占用，降低了云端的功耗和负载。

终端设备具有了较为强大的数据处理功能，各种新型的移动应用不断出现在人们的眼前。例如人脸识别，路面监控，人机交互等应用。虽然这些应用分属于各个不同领域，但是他们却有着共同的特点，那就是对实时性要求比较高，也就是所谓的延迟敏感型应用。比如在工业生产中，机器的运行效率会直接影响到工厂产品的产出，所以通常情况下工厂会对机器进行实时监控，一旦发现工作现场出现问题要及时的作出响应，否则就可能造成重大的损失。 虽然移动终端设备，比如一些智能传感器、无人机、车载设备，通过不断更新换代，使处理器的数据处理能力越来越强大，但是面对海量的数据，它们仍然不能完全满足用户对服务质量的要求，对于复杂的计算任务很可能无法在短时间内完成，而且长时间进行计算会带来大量的电量损耗，导致移动终端的续航能力不足。为了解决这些问题，终端设备可以通过互联网将计算任务传送到中心服务器去处理[1]。

数据在中心服务器计算有几个优势：（1）中心服务器计算能力强大，能够处理复杂计算，产生的计算时延较低。（2）它可以运行复杂的应用程序，为数据计算提供可靠的保证（3）能够提供庞大的存储空间。

但是将计算任务交给中心服务器也有着明显的缺陷，那就是对网络环境要求较高，当网络通信环境较差或者终端设备距离中心服务器距离较远的时候将会产生很高的延迟[5]。所以如何动态的进行任务调度使数据处理产生的时延与能耗能够减少就成了一个急需解决的问题。

关键字：边缘计算、任务调度、蚁群算法

# 边缘计算

物联网技术是指利用无线通信技术、网络技术、智能技术等共同构造一种全球物体信息共享的万物互联的网络。随着互联网相关技术的发展，我们正在从物联网时代走入万物互联的时代，其突出特点是网络中的各种设备都将具有感知环境的功能，拥有更强的计算能力，网络中将存在数十亿甚至数百亿的连接节点物联网技术在提供更智能化功能的同时，也会在网络中产生海量的数据，如何处理海量的数据，是物联网技术发展过程中要解决的突出问题。基于云计算的解决方案，将需要处理的数据发送到云计算中心，利用云计算中心的强大的计算能力来集中解决数据的计算和存储问题[云计算解决方案的集中式处理方法，虽然解决了资源的限制，但由于终端设备和云服务器之间存在着较高的延迟，对于有适时性要求的任务,可能会带来巨大的延迟时间从而影响了服务质量在这种情况下，基于云计算不能高效的支持基于万物互联的应用服务，边缘计算技术作为云计算的补充可以较好地解决这些问题

在边缘计算网络中，与数据的存储和处理操作将在分布于网络边缘的边缘服务器中进行分布式处理，而不是在云服务器集中处理海量数据将在接近数据生产者的边缘服务器中进行，从而降低了对于主网络的使用，减少了任务处理的延迟时间.边缘计算网络采用的分布式的架构，数据的存储、处理将分布在网络边缘的服务器中进行，与云计算相比，将会极大地减少这些过程中的网络延迟.

物联网所产生的数据有近乎一半都产生在终端设备上，而且终端设备采集的数据大部分为多媒体数据，图像视频数据占了很大的比重，要处理这庞大的数据量，现在主要有两个方法，一种方法是将计算任务在终端设备上进行处理，另一种方法是将计算任务上传边缘服务器或中心服务器去处理。然而单纯使用这两种方式都存在一定的问题。首先是计算任务在终端设备上处理，终端设备一般存储较小，这就意味着终端设备不会拥有复杂的任务处理模型，也就是说终端设备数据处理能力有限，很难处理复杂的计算任务。其次，终端设备很大程度上受到电源的限制，终端设备需要在多种环境下工作，但是并不是每个地方都能够提供电源，大多数终端设备都是通过内部电源或者是太阳能电池板工作的，这就意味着终端设备供电能力不足，续航能力受到限制，无法进行长时间工作。而电量不足还会导致终端设备的计算速度受到限制，计算任务在终端设备上处理产生的时延就会比较高。所以单纯让终端设备来处理这些数据是不太合理的。

如果把计算任务上传到服务器处理同样会遇到许多问题，首先就是网络环境的问题。上边提到的很多终端设备使用电池或者太阳能电池板供电，由于供电量不足也就很少选择 3G 和 4G 这种高功耗的传输方式，往往选择功耗较低的 LoRa、ZigBee 等方式去和中心服务器进行数据交互。LoRa 和 ZigBee 都是低功耗的局域网无线标准，功耗低也就意味着传输速率慢，并且不能覆盖到较远的地方，如果要处理的数据是高清图片或者是其他占用资源较大的数据，使用低功耗传输协议进行传输的话可能要耗费很长的时间，甚至可能出现无法将数据上传到中心服务器的情况，这显然是不合理的。其次，上面提到了终端设备可能要在各种环境下工作，那就固然不能避免一些网络信号不好的地区，比如我们要检测森林的火控预警等情况，森林里的终端设备即使采用了 3G、4G 这种传输速率很高的传输方式，但是森林可能并不能完全覆盖 3G、4G 的信号，数据就未必能高效的传输到服务器上。如果终端设备是一个车载设备，它可能移动到任何地方，那么终端设备所处的网络环境就有可能时好时坏，网络的不稳定性导致了终端设备不能随时将数据上传到服务器，这时候计算任务优先在本地处理还是上传服务器处理就更不明确了。

同时，服务器还存在负载上限，如果请求中心服务器的终端设备过多，服务器的资源可能不够分配，结果就是造成数据处理产生很高的延时。其次，服务器所处位置往往比较分散，有些服务器很可能距离终端设备较远，如果数据选择上传到服务器，会在传输上产生较高的延时，如果应用的实时性要求比较高，那么肯定会影响到用户的体验。

综上所述，由于终端设备和服务器存在各种条件的制约，很难确定计算任务是在终端设备处理还是上传到服务器去处理，因此需要研究一个动态的处理办法，根据终端设备的数据处理速度和存储容量、供电情况以及网络状态、服务器的负载等因素，动态的选择数据的处理位置，尽可能的减少数据处理产生的时延与能耗。

通过将一组服务器部署到靠近用户设备的附近，智能设备应用能够方便获得后端计算服务的支持，解决了当前的智能设备的计算存储能力和电源供给不足的问题，又能够使得分配给云端执行的任务延迟不会太高。我们把这样一组处在边缘网络的服务器叫做边缘计算节点。边缘计算节点部署在智能设备和终端用户附近，可以减少数据传播延迟和ＷＡＮ带宽使用。当前的边缘计算节点一般由一个统一的计算平台或者运营商进行管理，例如亚马逊［１３］，阿里云［１４］，华为云［１５］和微软［１６］等提供的边缘计算服务，都是由运营商统一管理整个平台的资源分配。而这些平台通常将平台的管理与服务的操作分离开来，既方便了企业也方便了客户，这也是许多的现代云计算服务的典型需求。

边缘计算与云计算的结合将是物联网网络架构发展的趋势，借助边缘服务器接近数据生产者的优势和云服务器的强大的计算、存储能力，将对未来物联网环境所产生的海量数据进行有效的存储和处理。边缘计算架构中，网络边缘部分由边缘服务器组成相互连接的网络，相临近的边缘服务器各自连接为一个网络，通过核心网络与云服务器相连接，所有终端设备都有与其相临近的边缘服务器用于接受其可能传输的任务。边缘服务器的计算能力、存储能力、上传带宽、下载带宽以及计算单位计算量的能耗成本各自不同，云服务器的计算能力与存储能力与边缘服务器相比，远远超越后者。

边缘计算与云计算相结合的网络架构为物联网技术的发展提供了有力的保障，也带来了新的挑战。与云计算不同，在边云结合的网络架构内，任务处理的位置可以有选择，对于不同需求、不同特性的任务，其所适合处理的位置也不同所以云计算中使用的调度策略并不能直接适用于当前边云结合的计算架构。资源分配与任务调度算法则在对任务处理位置和顺序的安排中将起到十分关键的作用。综上所述，为边云结合架构寻找适合的资源分配与任务调度算法，将为边缘计算和物联网技术的发展做出重要贡献。



# 任务调度

在云计算场景下，任务调度的一般策略是将计 算密集型任务迁移到资源充足的计算节点上执行． 但是在边缘计算场景下，边缘设备产生的海量数据 无法通过现有的带宽资源传输到云计算中心进行 集中式计算，且不同边缘设备的计算、存储能力均不 相同，因此，边缘计算系统需要根据任务类型和边缘设备的计算能力进行动态调度．

调度包括２个层面：

１）云计算中心和边缘设备之前的调度；

２）边缘设备之间的调度．

云计 算 中 心 与 边 缘 设 备 间 的 调 度 分 为 ２种 方 式：自下而上和自上而下．自下而上是在网络边缘处 将边缘设备采集或者产生的数据进行部分或者全部 的预处理，过滤无用数据，以此降低传输带宽；自上 而下是指将云计算中心所执行的复杂计算任务进行 分割，然后分配给边缘设备执行，以此充分利用边缘 设备的计算资源，减少整个计算系统的延迟和能耗． ２０１７年，Ｋａｎｇ等人设计了一个轻量级的调度器 Ｎｅｕｒｏｓｕｒｇｅｏｎ，它可以将深度神经网络不同层的计 算任务在移动设备和数据中心间自动分配，使得移 动设备功耗最多降低了９４．７％，系统延迟最多加快 了４０．７倍，并且数据中心的吞吐量最多增加了６．７ 倍．边缘设备间也需要动态调度．边缘设备的计算、 存储能力本身是不同的，并且会随着时间的变化而 变化，而它们承担的任务类型也是不一样的，因此需 要动态调度边缘设备上的任务，提高整体系统性能， 防止出现计算任务调度到一个系统任务过载情况下 的设备．Ｚｈａｎｇ等人［１４３］针对延迟敏感性的社会感知 任务设计了 一 个 边 缘 任 务 调 度 框 架 ＣｏＧＴＡ，实 验 证明该框架可以满足应用和边缘设备的需求．

# 现有的工作

**4.1 基于使用量预测的边缘计算资源与分配算法**

4.1.1背景

终端任务的处理通常伴随着边缘计算服务器的资源调度，因此很多现有的研宄都依赖于资源分配来完成终端任务的卸载和执行。不同的任务处理需要的资源量和资源情况不同。一些任务对服务器的资源情况没有特别的需求，在不同处理能力的服务器上完成处理的区别仅仅是时间的不同；有些任务需要的资源具有确定性，处理该类型任务需要大量的资源，当服务器的可用资源难以满足其所需资源的最低要求时，该类型任务将难以被处理。针对后一种任务类型，需要提前进行资源的预留和预分配，来保证该类型任务的及时处理。

资源的预分配可以为大型不可分割的任务实现资源的预留，保证这类型任务的及时处理。目前对边缘计算服务器资源预分配的研究比较有限，因此更早出现的针对云计算的资源预取的研究文献也值得探讨，这些文献一般侧重于对任务负载的预测。文献结合系统的现状和资源分布，利用虚拟化技术将任务调度问题抽象为虚拟机部署问题，对系统未来的工作负载进行预测，提出了一种基于保守控制策略的三次指数平滑算法。在此基础上，他们提出了基于概率匹配的多目标约束功耗。优化模型和低能耗资源分配算法。有的文献提出了一种集群和预取技术来有效地管理云存储资源，通过自动分类来识别每个类别中不同请求流的趋势，以预测的方式完成资源的预分配。上述文献研究了云计算资源的预取技术，但是由于边缘计算服务器的分布式的特性和资源的有限性，现有的针对集中式的云计算资源预分配的研宄可能难以适用于边缘计算场景。

己经有一些文献对云计算资源的预分配问题进行了研究，但对边缘计算服务器资源预分配问题的研究还比较匮乏，很多问题有待研究：第一，大型计算任务的处理需要一定的资源量，覆盖该终端设备的边缘计算服务器的可用资源量可能不足以处理该任务，此时可能存在任务处于等待状态直到资源量满足需求的情况，导致该类型任务长时间得不到处理。第二，现有的研究着重于系统性能的优化，很少从属于不同类型用户和服务提供商的满意度出发。第三，在实际的场景中，用户的任务到达时间并不完全同步，所以卸载的任务和资源分配算法难以通过批处理的方式来完成。在资源预分配的过程中，资源交易可能需要第三方的参与，导致敏感信息的泄露和资源的浪费。



4.1.2 Results

提出的基于双边拍卖的边缘计算资源预分配算法，根据不同的个性特征为参与资源交易的双方探索满意度最高的资源拍卖方案。该算法避免了已有资源交易场景中的第三方介入对资源的浪费问题，能兼顾用户和服务提供商的角色需求，同时考虑到终端设备任务的处理时延和成本以及服务提供商的收益，通过针对场景的特殊需求调节参数来提高双方的总满意度，在保护终端隐私信息的前提下，能准确的预测移动终端设备的目的地和资源需求量，并降低终端任务的处理

针对边缘计算中的资源预分配问题，提出了基于使用量预测的边缘计算资源预分配ＤＡＥＲ算法。首先，考虑到终端设备的高速移动性，建立了状态预测模型来预测终端设备的阶段性目的地。其次，建立了资源需求模型和拍卖模型来描述用户与服务提供商之间的关系。在建模过程中，考虑了用户和服务提供商的个体意识和行为偏好，根据满意度评价标准将用户和服务提供商划分为多种类型。最后，基于遗传算法得到最优的资源预分配方案，使用户和服务提供商的总体满意度达到最大。



**4.2 基于延迟感知的边缘计算任务调度策略**

**4.2.1 背景**

**边缘计算网络的优势在于为数据生产者就近提供了处理数据资源，大大减少了向上层服务器传输的数据量。在边缘计算系统中的传统静态任务调度策略缺少对同处在网络边缘的服务器的连接，终端设备将原始数据传递给与其直接通信的边缘服务器时，如果这台服务器无法处理应用程序的所有模块，便将超出能力的模块向上放置，忽略了同层中通信延迟更小的边缘服务器。针对传统静态任务调度的不足，面向边缘计算网络架构，提出了一种基于延迟感知的动态任务调度策略。并与传统的静态任务调度策略进行了对比分析。**

**提出的基于延迟感知的动态任务调度策略（Latency Aware Online task scheduling strategy, LAO）运行在部署在网络边缘的边缘服务器之上，在边缘服务器之间不设置监控节点。这个任务调度策略的目标是在边缘计算网络中找到一个合适的服务器来处理任务，以实现任务处理过程中的低延迟。由于使用了动态的任务调度策略，所以在服务器中允许动态地创建虚拟机和为虚拟机动态地分配资源。**

**4.2.2 成果**

**针对边缘计算系统中的任务调度问题，设计了一种基于延迟感知的动态任务调度策略，该策略考虑了边缘网络中同一区域内的边缘服务器的空闲资源，通过对任务截止时间的限制将系统中应用程序的响应延迟保持在一个较低水平。最后，通过模拟环境下的仿真实验，证明了该策略的有效性。**

**4.3 基于改进蚁群算法的任务调度算法**

4.3.1 背景

由于边缘计算中数据的异构性、计算和存储资源的局限性，导致利用有限的资源实现对于时延敏感应用的有效支撑是目前要亟待解决的关键问题。那么，能否实现任务的合理调度是解决上述问题的思路之一。目前针对任务调度已有部分研究者提出了一些解决方案，具体有：有的使用动态资源调度算法实现边缘端的资源分配；利用资源匹配算法来将云数据中心的资源分配在边缘上，通过二者的协作实现边缘计算的低时延和高服务质量。有的通过改变Storm的Task实例的排序分配方式以及二者的映射关系实现全局的优化调度；同时，针对拓扑任务并发度的缺陷，提出了一种基于蝙蝠算法的调度策略，以满足边缘节点之间的高实时性处理要求。有的是在边缘网络上搭建一个实时数据处理的应用程序，希望通过这种方式减少数据传输量来降低能耗。有的将数据块的最佳放置和任务的最佳调度相结合，以减少提交任务的计算延迟和响应时间，改善边缘计算用户的体验。以上这些算法从不同的需求角度提高了任务的完成效率，但较少有人关注由于任务调度的不合理导致的计算资源的负载不均衡，使得边缘环境下有限的资源无法得到充分利用，进而提高了用户的响应时间。文中将在蚁群算法的基础上，以任务的资源需求为约束实现边缘端任务分配的负载均衡，让有限的边缘资源发挥最大的功效。

该算法将要求解的问题抽象为如下模型：

在某一时刻一个边缘节点接收到的任务数为n,表示为TN={tn1,tn2,…,tnn};任务i对于资源的需求描述为(dcpui,dgpui,dmemi),其中dcpui表示该任务对于CPU的需求，dgpui表示该任务对于GPU的需求，dmemi表示该任务对于内存的需求；而该节点需要将这些任务合理分配至其所在的边缘网络当中，边缘网络中可分配的计算节点数量为m,表示为PM={pm1,pm2,…,pmn};节点j中可用的资源描述为(rcpuj,rgpuj,rmemj),其中rcpuj表示该节点CPU可用量，rgpuj表示该节点GPU的可用量，rmemj表示该节点内存的可用量。

任务TN到节点PM的分配关系可以用矩阵A表示为：

|  |  |
| --- | --- |
|  |  |

这里的aij表示为任务Ti和节点Pj的对应关系，aij∈{0,1},i∈{0,1,…,n},j∈{0,1,…,n},如果aij=1则表示任务i分配在了节点j上。

4.3.2 成果

针对边缘计算环境下边缘节点间资源差距过大且任务分配的负载不均衡等问题，提出了一种基于蚁群优化算法的任务调度方法。方法以不同任务对于CPU、内存、带宽等计算资源的需求情况的差异作为任务选择边缘节点的约束条件，以边缘云达到整体的负载均衡为目标，通过改进启发式因子、信息素的更新等条件提高算法的整体计算效率，降低计算时间，最后通过利用蚁群算法实现任务在边缘环境下的合理分配得出最优分配方式。方法能够避免相同类型的任务部署在同一节点中以提高任务执行效率和运算资源利用率。仿真实验结果表明，该算法在相同的节点数量下可以分配更多的任务，而且相同的任务数量下边缘节点整体具有较低的负载不均衡度，提高了资源的利用率，降低了任务整体的计算时间。

提出的算法相对轮询算法(RR)、传统蚁群算法(ACO)提高了最大任务可承载数，提高了边缘节点的使用效率，从而降低了网络中心总流量，降低了对用户的响应需求的平均响应时间；而且IAC算法每次会有多个近似最优解生成，可以根据需求自由选择。通过三个图的对比可以看出，全部为小型任务时各算法差距很小，当加入大型任务时可以看出文中提出的算法更加优秀。同时若是对负载需求不高的情景也可以通过减少蚂蚁数的方式提高分配效率，降低算法的运行时间。

在边缘节点数不变的情况下，提出的IAC算法的负载均衡度在随着任务数的增长过程中均优于剩余的两种算法，且优秀程度也有着一个较为明显的差距。这是由于随着任务数量的增多在分配过程中分配的可能性也就越多，也更有可能出现优秀的分配方式，所以算法的整体负载均衡度呈现一个下降的趋势。综上可以得出结论，IAC算法无论是从负载均衡度还是任务的分配数上相对于剩余两种算法均有一个良好的表现。

边缘节点的负载均衡是边缘计算下一个十分重要的问题。目前方法在资源分配方面主要以时延为准则忽略了边缘环境的局限性，文中针对边缘环境下资源的局限性，首先对边缘节点的任务分配问题进行描述；然后设计了一种改进的蚁群算法用来解决该问题。该算法将任务对资源的需求作为启发因子，改变了信息素更新方式，加快了收敛速度，提高了最终解的质量。通过仿真实验结果表明，算法可以有效降低节点的负载不均衡度。

**4.4 面向实时流数据处理的边缘计算资源调度算法**

4.4.1背景

随着物联网技术的快速发展，越来越多的终端设备如无 人机、植入式医疗设备、车载雷达等被大规模部署在各种生活 场景，并源源不断地采集图像、视频等环境信息，来支持电力 巡检［1］ 、智能医疗［2］ 、无人驾驶［3］ 等实时数据处理任务。然而， 由于终端设备的计算能力有限，而与数据中心（核心网）的通 信代价又太高，传统的“终端设备-数据中心”的架构很难满足 的实时数据计算任务对低时延的需求。近年来，研究者开始 使用智能边缘计算［4-6］ 的框架来实现任务分流，其基本思想是 在靠近数据源的网络边缘部署计算设备，形成“终端设备-边 缘节点-数据中心”的体系架构，并基于监控到的任务负载信 息，将应用动态地部署在边缘节点或数据中心，既可以充分利 用比终端设备计算能力更强的边缘设备，也能够有效避免数 据在核心网中传输造成的高时延，从而有效提升服务质量。 在智能边缘计算中，受限于边缘设备的带宽等资源，一个 非常关键的问题是如何合理利用监控到的任务负载进行资源 调度，来降低应用的时延。

在边缘计算进行任务调度的过程中，不仅要实时地 监控任务运行的信息，也需要考虑到与其他的边缘节点、甚至 是数据中心中的其他任务的交互过程中的数据传输问题，因 此属于一个多级、多数据流、任务依赖的资源调度问题。而目 前大多数研究工作主要面向的是数据流单任务处理的应用，当需要处理分布在多个边缘节点覆盖下的终端设备产 生的多条数据流时，传统的方法往往忽略任务之间的通信关 系，带来额外的通信开销并导致时延的增加，往往很难适用。

我们将系统抽象为如下模型：

系统包含若干个计算 节点，记其数目为 M，其集合为 D = { d1 ，d2 ，⋯，dM}，这些计算 节点包括1个数据中心和M - 1个边缘计算设备，令计算设备 d1表示数据中心。其中，对于任意边缘计算节点di ，其直接覆 盖若干个终端智能设备，每个设备作为数据生产者会产生数 据流，数据流会在该边缘计算节点处理或经由该边缘计算节 点转发至其他计算设备进行处理，即边缘计算节点 di所覆盖 的终端智能设备所产生的数据流的集合为Qdi 。

系统中，需要部署的流数据处理应用数目为N，即该集合 为 S = { s1 ，s2 ，⋯，sN}。对于任一流数据处理应用 si ，其数据生 产者可能有多个，每个数据生产者产生一条数据流，记其集合 为 Qsi ；而数据消费者只有一个，本文使用变量 xij 表示流数据 处理应用 si的数据消费者是否为 dj ∈ D，若是，则xij = 1，否则 xij = 0，同时，每个任务只能部署在单个的节点上。显然，有以 下约束条件： xij ∈ { 0，1}；∀i ∈ {1，2，⋯，N }，j ∈ {1，2，⋯，M } （1） ∑ j = 1 M xij = 1；∀i ∈ {1，2，⋯，N } （2）

由于不同任务会竞争带宽资源，因此需要在任务部署的 过程中考虑节点之间的带宽限制。令Pk ij表示应用si部署在计 算节点dj时，计算节点dk为支持应用si所消耗的带宽。显然， Pk ij可表示为： Pk ij = q ∈ { ∑ Qs i ∩ Qdk } rq； ∀i ∈ {1，2⋯，N }，∀j，k ∈ {1，2，⋯，M }，j ≠ k （3） Pk ij = ∑ q ∈ { } Qs i - Qs i ∩ Qdj rq； ∀i ∈ {1，2，⋯，N}，∀j，k ∈ {1，2，⋯，M}，j = k （4） 其中：rq 表示数据流 q 所消耗的带宽，Qsi ∩ Qdk 表示经由计算 设备 dk 转发的交由应用 si 处理的数据流集合，Qsi - Qsi ∩ Qdj 表示交由应用si处理的不在计算设备dj覆盖范围内的数据流 集合。本文假设终端智能设备的位置和数据流所消耗的带宽 资源是固定的，因此，Pk ij在系统中表现为一个定值。 对于任一计算节点 dj ，其带宽承载能力是固定的，记为 Uj 。对于任意应用 si ，假如其部署在计算节点 dj上，则消耗计 算节点 dj 的带宽资源的大小为 P j ij ；假如其部署在计算节点 dk ( j ≠ k) 上，则消耗计算节点 dj的带宽资源的大小为 P j ik。要 求每一计算节点消耗的带宽不超过该计算节点所能提供的带 宽，因此有如下约束： ∑i = 1 N ∑k = 1 M (xikP )j ik ≤ Uj；∀j ∈ {1，2，⋯，M } （5）

为了实现最优的任务部署，本文的目标是最小化系统延 迟。系统延迟可以定义为所有数据流从数据源节点到任务目 标节点的延迟之和，由于不同任务所在的计算节点和数据源 节点往往并不相同，因此需要合理地规划任务部署的位置，实 现最优的系统延迟。 对于任意流数据处理应用 si ，其处理的数据流有若干条， 每条数据流在终端智能设备上发送，经直接覆盖该智能设备 的计算节点 dj 转发，传输到部署 si 的计算节点 dk 进行处理。 由于智能设备到边缘节点的传输时延远远小于边缘节点到边 缘节点间的传输时延，本文的延迟代价只考虑计算节点之间 的传输时延。令lij表示数据流在计算节点di与dj之间传输的 延迟，当 i = j 时，lij = 0，否则，lij > 0，由于边缘节点之间的位 置相对固定，因此假设 lij为一个常数。令 Lij表示数据流处理 应用si部署在计算设备dj上时，应用si处理的数据流的时延之 和，可表示为： Lij =∑k = 1 M ∑q ∈ { } Qdk ∩ Qs i lkj ； ∀i ∈ {1，2，⋯，N }，∀j ∈ {1，2，⋯，M } （6） 则在任一任务部署方案 x下，记系统的总延迟代价为 V，可表 示为： V =∑i = 1 N ∑ j = 1 M xijLij （7）

基于以上的定义，本文考虑如下问题：在一个多边缘计算 节点‒单数据中心的计算网络中，面向数据流处理应用，寻求 一种任务放置和带宽资源分配方案，使得在满足带宽资源约 束的条件下，系统的总时延最小。该问题可以形式化地定义 为问题P1： min { X } V =∑i = 1 N ∑ j = 1 M xijLij s. t. C1：xij ∈ { 0，1}；∀i ∈ {1，2，⋯，N }，j ∈ {1，2，⋯，M } C2：∑ j = 1 M xij = 1；∀i ∈ {1，2，⋯，N } C3：∑i = 1 N ∑k = 1 M ( xikP j ik ) ≤ Uj ；∀j ∈ {1，2，⋯，M } 其中，优化的目标为部署方案 x下的总延迟代价。约束 C1表 示决策变量 X = {xij |i ∈ {1，2，⋯，N }，j ∈ {1，2，⋯，M } }的取值 范围为二元的离散值{ 0，1}，xij = 1 表示应用 si部署在计算节 点dj上；约束C2表示任意应用si ，必须也只能部署在某一计算 节点上，计算节点包括 1 个数据中心和 M - 1 个边缘计算几 点；约束 C3 表示任意计算设备 dj ，使用的带宽资源不超过其 可用的带宽资源。

4.4.2成果

主要研究了“终端设备-边缘节点-数 据中心”网络架构中的物联网多数据流的应用部署问题，目标 是通过合理的应用部署，使得在满足网络资源约束的条件下， 相关流数据处理应用的总服务时延最小。本文首先证明了该 问题是非确定性多项式难（Non-deterministic Polynomial，NP） 难的。为了实现高效的任务分配，本文设计了一个动态的资 源 调 度 方 案 FFS+IPFS（Find Feasible Solution+ImProve Feasible Solution），其主要思想为首先选取占用带宽资源最少 的应用和边缘节点进行部署，得到近似最优的可行解，然后在 此基础上，通过局部搜索算法对现有方案进行迭代优化，使得 优化后的部署方案能够尽可能保持同质性，即具有相互依赖 的任务被部署在相近的边缘节点，从而减少任务通信带来的 开销。本文进行了大量的模拟实验，将该算法与 ODS（Only Data Center）、Hash 和 Closest 等主流算法进行比较，实验结果 显示，在不同的负载情况下，本文提出的 FFS+IPFS 算法都可 以有效实现任务调度，并且时延比其他算法低 23% 以上，验 证了本文算法的有效性。

本段主要贡献可以概括为：

l）针对边缘计算场景的多数据流任务调度进行建模，证 明了在网络资源约束的情况下，该问题是NP难的。

2）利用边缘计算场景中存在大量对资源需求少的任务 的特点，提出了一种动态的资源调度方案 FFS+IPFS，通过迭 代优化的方法有效提升了算法的性能。

3）在真实的边缘计算任务数据集中进行了模拟实验，实 验结果显示，本文提出的 FFS+IPFS 可以有效实现任务调度， 平均时延相比其他主流算法降低23%以上。

**4.5 多目标优化的边缘计算系统资源分配方案**

4.5.1 背景

由于边缘计算网络中物理服务器的异构性，以及每个层级的物理距离会很大程度上影响边缘计算系统中应用的延迟和设备的功耗，所以需要设计一种有效的资源分配方法，以减少应用程序处理的响应延迟和该过程中产生的能耗。目前，针对边缘计算网络的研究已经广泛地开展，并且取得了初步的成果，然而针对边缘计算资源分配方案的研究所取得的结果却不理想，这主要是因为边缘计算网络中的资源分配过程是一个 NP 难问题，理论上只有通过穷举所有方案才能得到全局最优解，但这一过程会随着问题规模的扩大消耗越来越多的时间。如何在可接受的范围内找到理想的解决方案是边缘计算资源分配算法需要考虑的问题。传统的解决方案是通过一定的规则将应用模块映射到终端设备到云数据中心的垂直路径上，这种方案很难顾及到边缘网络中的其他服务器，容易造成边缘计算网络中资源的浪费和带来较大的延迟，从而影响优化结果。 针对以上问题，本文提出了一种以低延迟、低能耗为优化目标的边缘计算应用模块映射算法。该算法针对于网络边缘的物理服务器可以相互通信的边缘计算网络，同时将模拟退火算法引入到了应用模块映射过程中，具有高效且能够搜索到全局最优解的优势。通过该算法所生成的映射方案，可以充分利用边缘计算网络中的资源，并且更偏向于产生能耗较低的物理服务器，从而降低系统的功耗和延迟。将这种算法称之为 SAP 算法（Simulated-Annealing module placement），并且在模拟退火算法中加入了记忆功能，可以将迭代过程中产生的最优解储存下来，避免在迭代过程中跳出最优解后被遗失，从而得到近似的全局最优解。

4.5.2 成果

针对边缘计算系统中的应用模块映射问题，提出了一种结合模块链和模拟退火算法的模块映射方案 SAP 算法。这种方案充分挖掘了边缘计算系统的潜在资源和发挥了模拟退火算法的快速寻找近似全局最优解的优势，并且参照二者的特性，对二者进行了改进，使得二者可以更好地结合。其中，模块链这一概念可以将传感器与应用程序更好的对应，应用程序所拥有的一部分传感器可以从统一的处理位置中脱离出来，寻找到具有更低延迟、更低功耗的服务器来放置自己对应的应用模块。传统的模块映射问题将一个网关所属的传感器对应的统一应用模块放置在统一的位置，而模块链的出现使得边缘计算系统中的模块映射问题规模更加庞大，依靠传统的模块映射算法很难解决。利用模拟退火算法可以寻找全局最优解的优势，可以较快的在解空间中找到理想的模块映射方案。此外，SAP 算法增加了记录最优解的功能，可以将迭代过程中优化程度最好的映射方案记录下来，避免在跳出局部最优解的过程中遗失最优解。

面向边缘计算系统中的低延迟、低功耗的需求，提出了一种结合了模拟退火算法的资源分配方案 SAP。该方案以低延迟、低功耗为优化目标，凭借模拟退火算法能够跳过局部最优解，快速地找到近似的全局最优解的优势，从而实现在保持系统低延迟的前提下最大程度地降低系统整体能耗。最后，经过实验设计和模拟仿真证明了该策略的优化效果。

# 未来展望

综上所述，现阶段的边缘计算有如下优点：

- 低延迟：计算能力部署在设备侧附近，设备请求实时响应；

- 低带宽运行：将工作迁移至更接近于用户或是数据采集终端的能力能够降低站点带宽限制所带来的影响。尤其是当边缘节点服务减少了向中枢发送大量数据处理的请求时。

- 隐私保护：数据本地采集，本地分析，本地处理，有效减少了数据暴露在公共网络的机会，保护了数据隐私。

综上所述，动态调度的目标是为应用程序调度 边缘设备上的计算资源，以实现数据传输开销最小 化和应用程序执行性能的最大化．设计调度程序时 应该考虑：任务是否可拆分可调度、调度应该采取什 么策略、哪些任务需要调度等．动态调度需要在边缘 设备能耗、计算延时、传输数据量、带宽等指标之间 寻找最优平衡．根据目前的工作，如何设计和实现一 种有效降低边缘设备任务执行延迟的动态调度策略 是一个急需解决的问题．