1 引言

而随着云技术的发展和云平台的广泛部署, 云中的工作流调度问题成为一个重要的研究课题[1], 在云计算服务交付的过程中, 由于用户直接面对的是虚拟机资源, 而真正解决问题的是虚拟机映射的实际的物理资源, 所以如何将任务合理分配到资源执行是我们所重点关注的.

(1) 云计算

云计算通过多种部署方式, 包括私有云、社区云、公有云、混合云 4 种部署模型[2], 通过基础设施即服务、平台即服务、软件即服务 3 种服务模式用户提供服务. 云计算有 2 个显著的特征, 快速弹性可扩展[3]、按需付费服务[4].

云计算关键技术[5]分别有: 虚拟化技术[6]、分布式数据存储技术[7]、大规模数据管理技术[7]、调度技术[8].

(2) 调度技术

云计算中的调度一般分为两个部分: ① 资源调度: 资源调度是指对物理资源进行合理有效的管理和使用等; ② 任务调度: 将任务合理分配到合适的计算资源执行.

简单点讲, 一次正常的用户服务流程为: 用户提交任务到云端, 任务调度器将任务分配到合适的计算资源上执行, 任务完成后再将结果反馈给用户. 其中, 任务分配这一环尤其重要, 一个合理高效的调度策略能够极大的提升云计算系统的性能. 下文重点关注的是任务调度.

随着云用户数量的不断增加, 为用户提供优质的服务势在必行. 因此, 任务调度非常重要, 因为它专注于在特定时间将任务分配给可用资源. 为了获得良好的用户服务质量, 需要高效的任务调度. 现有的云计算任务调度技术已经被研究者们提出, 但要实现高效的任务调度, 还需要进一步的改进. 所有算法的主要目标都是最大限度地利用资源, 最大限度地减少制造时间和成本, 提高性能.

(3) 云环境中的服务交付模型

云计算中的服务交付模型分为 4 部分, 如图 1 所示.

① 用户提交任务;

② 任务管理器将其拆分成多个子任务;

③ 任务调度器通过调度技术将子任务与物理资源建立映射关系;

④ 任务完成后进行汇总, 反馈到用户.

由此看出, 从任务出发到完成, 在整个任务执行过程中, 调度技术是重中之重, 它影响到了整个系统的运行效率、用户服务质量、系统负载均衡、系统能耗.

因此一个适合云计算环境的调度技术十分重要. 由于调度技术分为资源调度和任务调度, 本文只关注任务调度.

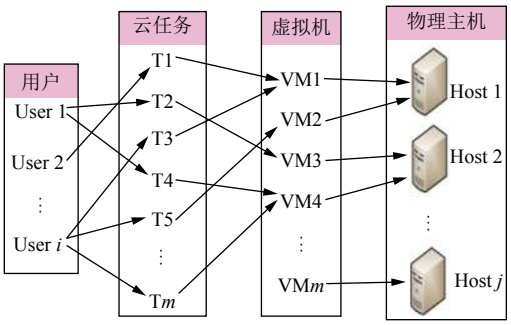

图 1 云计算服务交付模型图

2 任务调度

任务调度, 在云计算环境下其本质是一个映射的过程, 它在一定的约束条件下, 根据云计算环境下任务、资源两者的预测信息和状态将用户提交的互相独立的任务映射到相应的虚拟机资源上执行, 然后返回处理结果[9]. 简单点说, 就是将 x 个任务分配到 y 个计算机资源上执行的过程.

根据工作流和资源的可用信息以及任务分配给资源的时间可以将任务调度分为静态调度和动态调度[1].

2.1 调度分类

2.1.1 静态调度

静态调度中分为 4 类: 列表调度启发式[10,11]、聚类调度启发式[11]、重复调度启发式、元启发式.

(1) 列表启发式

基本思想是通过给任务分配一些优先级, 并根据优先级对其进行排序, 从而生成一个任务调度列表, 然后从调度列表中选择第一个任务, 再将任务分配给其对应的资源, 直到对所有任务进行调度.

调度列表可以静态或动态构造. 如果所有优先级都是在任务分配之前构造的, 则为静态列表调度; 如果在每个任务调度之后重新计算未调度任务的优先级, 则为动态列表调度. 但是无论是哪种列表调度, 都必须要有一个优先级的判断属性和资源选择策略, 并以此决定任务的优先级与其对应的最佳资源.

常见的列表调度启发式算法, 如最早时间优先

(Earliest Time First, ETF), 异类最早完成时间 (Predict Earliest Finish Time, PEFT)[10]等.

(2) 聚类启发式

基本思想是通过在同一个集群中对任务进行聚类, 将任务映射到集群, 在同一个群集中排序任务. 尽量牺牲并行性来降低通信延迟, 优化数据相关任务之间的传输时间.

如果同一个集群中有两个独立的邻居任务, 则称为非线性集群; 否则称为线性集群. 对于非线性聚类, 对独立任务进行排序. 线性聚类保留了并行性, 不会增加并行执行时间. 非线性聚类通过对并行任务进行排序来减少并行性, 从而增加并行执行时间.

(3) 重复启发式

基本思想是在目标任务的同一资源上正确地复制任务, 从而避免这两个任务之间的执行时间冲突, 以及在不同的时间段内可能会发生一些资源闲置的情况.

因为在实际的调度中, 有些任务可能正在等待分配给其他资源的任务的数据, 只有接收到该数据之后, 任务才能开始执行, 如果在数据接收之前轮到该任务执行, 就会发生阻塞, 产生空闲时段, 直到数据到来, 影响到后面任务的执行. 如果使用有效的调度算法, 通过识别任务使得这些空闲时段可以有效地利用, 那么就可以缩短总的执行时间.

基于重复的调度通常可以缩短生成时间. 然而, 这也使得调度问题更加困难. 调度算法不仅需要观察任务之间的优先级约束, 还需要识别哪些任务需要复制, 以及如何将它们放入空闲时段.

(4) 元启发式

元启发式是启发式的进化版, 是对启发式局部搜索的改进, 元启发式方法通常提供了一种快速朝着一个非常好的解决方案前进的有效方法. 如遗传算法、蚁群算法、粒子群算法等.

与基于局部搜索的启发式方法相比, 元启发式方法能够在更大的解空间中搜索到任务解决方案. 因此, 在大多数情况下, 元启发式比基于局部搜索的启发式性能更好.

将以上局部的静态调度做了一个比较, 比较结果如表 1 所示.

列表启发式大部分都适用于资源数量有限的系统, 比较关注如何在保证能完成调度的同时降低系统的开销.

聚类启发式通常假定资源数量无限, 并将大量通信任务分配给同一类资源. 虽然它减少了资源间的通信, 但当任务在资源上按优先级约束排序时, 也可能导致负载不平衡或空闲时隙.

重复启发式算法旨在降低任务的执行时间, 避免大量的通信. 重复启发式通常在牺牲计算时间复杂性的情况下具有更好的调度质量.

元启发式比以上基于局部的启发式性能更好. 但是, 随着工作流任务数的增加, 元启发式算法的调度时间开销也会迅速增加.

表 1 静态调度比较表

方式项目	列表调度启发式	聚类调度启发式	重复调度启发式
特点	按优先级进行列表调度	将任务映射到集群	复制任务到空闲时段
适用场景	资源数量有限	资源数量无限	无限制
优点	降低计划开销	降低通信延迟	降低执行时间
缺点	增加任务完成时间	可能导致负载不平衡、空闲时段	增加计算时间复杂度

2.1.2 动态调度

动态调度[12]是为了解决调度信息的不可用性和资源与其他工作流或非工作流系统负载的争用而开发的, 目标是在可用资源队列之间平衡负载.

任务执行和通信时间由资源和工作流信息共同决定. 其中工作流信息包括工作流结构、任务执行工作量、通信数据量等, 资源信息包括云环境下的可用性、处理能力、通信带宽等. 在实际生产系统中它们都是不完整的、动态的. 要准确地评估每个队列的负载并不容易, 因此需要一个动态调度策略来处理这些不确定性.

总的来说, 在静态中, 所有关于任务的信息在执行之前都是调度程序已知的, 可以事先获取到更多信息, 对任务调度的方案进行优化, 使得任务实际执行时开销更小; 而在动态中, 关于任务的信息在执行之前不知道, 但运行时开销更大, 但是能更好地处理负载平衡的问题.

通过对调度的分析发现, 不管是静态还是动态调度, 由理论到实际解决问题都是依靠具体的任务调度算法执行的. 简单地说, 一种任务调度算法就是一种调度方式, 云计算中任务调度的优劣主要就体现在任务调度算法的优劣上.

合理的任务调度要从 2 个出发点进行考虑: 云计算提供商, 用户. 对提供商来说, 最关心的就是成本, 所

© 中国科学院软件研究所 http://www.c-s-a.org.cn

以要能够有效协调和分配资源, 降低成本; 对用户来说, 服务质量放在首位, 所以需要在完成需求的同时, 降低任务的总执行时间.

所以一个既可以对虚拟资源进行有效的协调和分配, 降低提供商的成本; 又能快速、准确地完成任务, 给用户一个好的用户体验; 能让提供商和用户都满意的任务调度算法是云计算所迫切需要的.

而任务调度的约束条件就是依靠具体的任务调度算法来执行的, 所以选择合适的任务调度算法对策略的执行具有很重大的意义.

2.2 任务调度算法

任务调度算法根据调度目标的数量可以分为 2 类: 针对单一目标优化的传统任务调度算法, 针对多目标优化的启发式思想的智能化算法.

2.2.1 单目标优化的任务调度算法

单目标优化的任务调度算法主要包括最小完成时间 (Minimum Completion Time, MCT)、最小执行时间 (Minimum Execution Time, MET)、交换算法 (Switching Algorithm, SA)、贪心算法 (GReedy Algorithm, GRA)、先来先服务 (先进先出) 算法 (First Come First Server, FCFS/First In First Out, FIFO)、短作业优先算法 (Shortest Job First, SJF) 等.

其中 MET 算法考虑到任务的最短执行时间, 忽略了负载是否平衡的问题; MCT 算法考量到任务最早的完成时间, 但是可能导致工作时间变长; SJF 算法考虑选择用时较短的任务优先执行, 而忽略了任务的优先级.

从以上不难看出: 单目标优化的任务调度算法重点在于单一目标的最优解, 即某个实例的最优, 体现在“最”字上, 这样的话其他实例没在考虑的范围内, 就会不可避免的舍弃掉很多东西, 比较局限于某个目标最优, 而无法考虑到全局, 都是比较极端的算法, 为了达到目标, 可以不惜牺牲一切. 导致的后果是: 虽然该实例取得的效果最好, 但是因为要最大化的满足它而舍弃掉太多东西, 导致其他实例的效果就不是很理想甚至很糟糕, 使得结果具有很大的局限性而无法推广.

2.2.2 多目标优化的任务调度算法

在优化设计中, 要求多个目标达到最优的问题被称为多目标优化或者多约束问题. 在这种情况下, 基于启发式思想的智能化算法应运而生.

启发式思想的智能化算法思想在于: 解决多约束问题时, 在可以接受的花费的前提下, 得到一个解决方案, 给出尽量满足多个目标优化的一个可行解. 其核心点在于“多目标优化”, 即对于每一个实例来说, 也许当下解并不是它的最优解, 但却是多个实例在尽量满足需求条件下的极优解.

常用的启发式思想的智能化算法包括 2 类[13]:

(1) 基于生物启发 (Biological Inspiration, BI)

遗传算法 (Genetic Algorithm, GA)、模因算法 (Memetic Algorithm, MA)、狮子算法 (Lion Algorithm, LA)、帝国竞争算法 (Imperialist Competitive Algorithm, ICA), 是在云计算中与任务调度相关的少数生物启发算法.

(2) 基于群体智能 (Swarm Intelligence, SI)

蚁群 (Ant Colony Optimization, ACO) 算法、粒子群优化 (Particle Swarm Optimization, PSO) 算法、模拟退火 (Simulated Annealing, SA) 算法、人工蜂群 (Artificial Bee Colony, ABC) 算法、猫群优化 (Cat Swarm Optimization, CSO) 算法、蝙蝠算法 (Bat Algorithm, BA)、风驱动优化 (Wind Driven Optimization, WDO) 算法等[13].

3 任务调度算法

3.1 遗传算法

GA 的创造思路来源于达尔文著名的生物进化论中“优胜劣汰”的自然选择原则, 将自然生物的进化类比应用到实际问题中, 是通过模拟自然进化的过程来解决组合优化问题的一种搜索最优解的方法.

GA 对于给定问题的任何解决方案都由一组称为基因的元素组成的染色体代表, 即每一条染色体代表一个解决方案. 算法终止后, 最出色的染色体就是给定问题的最优解.

算法的优点: 进行选择、交叉、变异等操作的目标是选择区域中所有的染色体, 初期全局搜索能力强, 并行性强;

缺点: 要花费大量时间对染色体进行评估、选择、交叉、变异等操作, 后期求解效率低.

遗传算法的优点主要在于其强力的全局搜索, 但是在算法的搜索效率以及算法解的多样性方面还是需要改进, 所以以下文献主要对算法的这 2 个部分进行优化, 如表 2 所示.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

表 2 遗传算法优化表

优化目标	文献	应用场景	优化内容	对比算法	优化结果	实验环境
算法搜索效率	[14]	云环境科学工作流调度	两类种群之间协同进化	Random; HEFT; GA; PSO	严格约束时, 满意度波动为 15% 左右, 更加稳定	Cloud Sim 支持的云工作流仿真工具包: WorkflowSim
	[15]	云任务调度	设计了多个优化目标的适应度函数	TGA; CGA	任务完成时间减少; 成本降低	Cloud Sim
	[16]	云任务调度	引入虚拟机相对适应度的概念, 使随机变异改进为有目标的变异	GA; Min-Min	收敛值降低 10%; 负载均衡标准差操持在 10 以下	CloudSim3.0
算法多样性	[17]	相控阵雷达任务调度	根据个体实时适应值动态调整交叉、变异概率调节公式; 采用 Logistic 方程对种群进行混沌初始化;	HGA; AGA; 启发式算法	调度成功率提升 40%; 实现价值率提升 20%; 时间利用率提升 70%; 时间偏移率减少 80%	Windows7, Inter(R)Core(TM) i74790 CPU(3.6 GHz), 4 GB 内存
	[18]	云环境科学工作流调度	调度方案中引入交叉变异操作	Random; HEFT; GA; Fussy-PSO	执行时间跨度更大; 执行代价更小	WorkflowSim

3.2 蚁群算法

ACO 最早是于 20 世纪 90 年代, Dorigo M 等[19]通过观察蚂蚁觅食行为, 并从中获得设计灵感而提出的, 常用于求解复杂组合优化问题的一种元启发式智能优化技术[20].

信息素的管理主要包括信息素的局部更新和信息素的全局更新 2 个部分[21].

算法的优点: 在后期确定了目标点和最优路径之后, 所有成员每次都会按照最优路径进行计算, 节省大量时间, 速度快.

缺点: 在初期确定目标点和最优路径时, 因为信息匮乏需要花费大量的时间做大量的摸索尝试, 速度慢.

蚁群个体之间通过信息素交流, 信息素积累起来之后, 搜索效率、解的精度极高, 但同时, 其缺点在于前期信息素的积累花费了太多时间, 由于收敛速度快, 也容易陷入局部最优, 所以以下文献主要在算法的解的多样性和算法前期的搜索效率方面进行优化, 如表 3 所示.

表 3 蚁群算法优化表

优化目标	文献	应用场景	优化内容	对比算法	优化结果	实验环境
算法多样性	[22]	云任务调度	引入了从属蚂蚁的多样化策略	ACOIACO	云任务为 700 时, 完工时间领先 ACO 50 s, IACO 20 s; 平均完工时间降低 20-40 s	文献表 2
	[23]	旅游路线规划	引入了随机选择概率; 动态设置启发式单元参数, 信息扩散系数	经典 ACO	节省交通成本, 缩短旅程时间, 提高旅客满意度	Matlab
	[24]	约束规划领域	强化了边缘信息, 充分利用动态变化信息	ACOSACODPRW	在解决大规模 CSP 时表现更突出	随机约束网络经典模型, Java
算法搜索速率	[21]	面向服务软件部署优化	设计了一种包含 5 种搜索规则的局部搜索策略	ACONSGA-II	解集更稳定; 求解质量更好; 响应时间降低利用率更高	Windows 7; MyEclipse 8.5 Java
	[25]	激光焊接轨迹规划	精英蚂蚁和全局更新相混合的信息素更新策略	经典 ACO	焊接轨迹长度缩短 65.3%	四轴焊接平台

3.3 粒子群算法

粒子群算法, 也称为鸟群、鱼群觅食算法, 是模拟鸟群、鱼群的觅食行为规律在搜索空间进行搜索最优解的方法. 最早由 Kennedy J 和 Eberhart R[26], 该算法最初是对一个简化的社会环境的模拟, 主要应用于求解连续搜索范围问题[27].

与遗传算法这种进化算法不同, 遗传算法会舍弃适应度较低的染色体, 而粒子群不使用选择, 不会舍弃任何一个解, 尽管这个解的质量很差, 算法会利用粒子间的信息交流对差解不断地优化, 所有的种群成员从试验开始一直存活到最后.

算法的优点: 搜索速度快、收敛速度快、参数少等.

缺点: 局部搜索能力差、初始化粒子随机性强.

粒子群和蚁群算法在某些方面比较相似, 搜索速度快, 但是容易陷入极值, 所以以下文献主要在算法多样性和精度方面进行优化, 如表 4 所示.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

表 4 粒子群算法多样性优化表

文献	应用场景	优化内容	对比算法	优化结果	实验环境
[28]	云任务调度	加入混沌扰动机制	PSO; SPSO	收敛更快; 任务完成时间、成本更低	CloudSim
[29]	云任务调度	引入 VNS 算法	PSO	平均运行时间减少 5.5%;	Microsoft Windows XP; Matlab R2009b
[30]	云资源调度	加入 Sobol 随机序列	QPSO	随着迭代次数增加到 200, 任务完成时间减少 20 s、平均成本降低 13 s	Windows XP, Matlab R2012b
[31]	分数阶系统参数辨识	基于 Tent 映射的改进	GAIPSOACPSO	求解精度更高; 收敛速度提高	Matlab 7.11, Intel(R)Core(TM)15-2320 CPU, 3.00 GHz 和 104 GB RAM
[32]	算法研发	融合了全局和局部两种常用的邻居拓扑结构; 将原始种群分为 3 个子种群并赋予其不同的加速因子	FDRPSO; FIPSO; CLPSO; PSODDS; SRPSO; CCPSO-ISM	在 50% 的函数上求得最优解, 求解精度更高; 收敛速度更快; 显著性检验出性能更优	CEC2013 测试集
[33]	市区最优路径规划	引入平滑度存储路径和节点	PSO; A*	平均距离降低 40%	Microsoft Windows

3.4 模拟退火算法

模拟退火算法是一种随机搜索算法, 是对热力学中固体物质退火过程的模拟. 文献[34]中介绍模拟退火算法最早的思想是由 Metropolis 等提出. Kirkpatrick 等成功地将退火思想引入到组合优化领域.

固体物质退火过程与一般组合优化问题之间的相似性如表 5 所示.

表 5 模拟退火算法与组合优化问题的相似性

组合优化问题	固体退火
解	物质状态
控制参数	温度
目标函数	内能
求解过程	退火过程
Metropolis 准则	等温过程

SA 对于给定问题的任何解决方案都由退火过程中物质的状态代表, 即每个温度下物质的状态代表一个解决方案, 当内能到达最低时, 物质的状态就代表最优解[35].

算法的优点: 稳定性好, 在局部最优时容易有概率跳出并趋于全局最优;

缺点: 对参数依赖性强, 收敛速度慢, 搜索时间长. 如果冷却过程足够慢, 模拟运行时间足够长, 模拟退火几乎可以保证找到最优的解决方案.

与其它智能算法不同的是, SA 在迭代过程中会以一定的概率接受与当前解相比较差的解, 接受概率随着温度的降低减小. 由于在搜索过程中接受差解, 所以有可能导致遗失掉最优解; 另一方面, 这种处理可以在一定程度上避免算法陷入局部最优解.

以下文献主要在使用 Metropolis 准则对新解进行一个概率接受的部分进行优化, 减小陷入局部最优的可能性. 如表 6 所示.

3.5 算法对比

由以上第 4 章对各算法的阐述, 发现各个算法都有其自身的优点和不足, 这里对其能力的各个方面进行比较, 如表 7 所示.

算法搜索过程中会进行信息交互的算法 ACO, PSO 都会面临一个问题: 信息交互后, 每个个体都会受到优秀个体搜索结果的影响, 都会向着优秀的个体“看齐”, 这样就会导致算法收敛速度大大加快. 算法收敛速度快会产生 2 种结果:

① 算法整体的进程节奏加快, 更早地达到终止条件;

② 算法丧失多样性, 容易陷入局部最优.

与之相反, GA, SA 的收敛速度都相对比较慢, 其原因是 GA 会对种群中的大量个体执行操作, SA 则会对温度变化过程中产生的每一个新解都进行操作, 花费了大量的时间, 收敛速度自然就快不起来. 算法收敛速度慢会产生 2 种结果:

③ 算法整体的进程节奏过慢, 算法搜索效率低;

④ 加强了算法的全局搜索能力, 有效降低陷入局部最优的可能性.

所以如何将算法的收敛性利用好, 在加速算法进程的同时增加算法解决方案的多样性是值得进行研究的.

启发式思想的智能化算法相比较于传统的算法优化更全面, 适用性更广, 虽然各自有各自的特点, 但同

© 中国科学院软件研究所 http://www.c-s-a.org.cn

时也有自身的不足, 要想使其具有更强的生命力, 就需要不断自我提升; 同时也需要借鉴其他算法的优势, 彼此相互学习, 优劣互补来弥补自身的不足, 同时达到更优的效果.

表 6 模拟退火算法优化表

优化目标	文献	应用场景	优化内容	对比算法	优化结果	实验环境
概率接受环节	[34]	学校排课问题	对当前最优解增加记忆处理; 设置双阈值减少计算量	传统 SA	收敛速度提高三倍; 运行速率提高	Windows 7, Eclipse, My SQL, Java
	[36]	共享孔径技术	对当前最优解增加记忆、对比处理	迭代 FFT	运行时间减少 0.06%	天线阵列
	[37]	导航卫星激光星间链路拓扑动态优化	设计一种避免冲突的链路交叉更新算法快速生成链路拓扑	SA	链路利用率提高 36~41%	BDS 的卫星星座, MEO 层为 Walker 24/3/1 星座构型
	[38]	柔性调度	个体调换、局部颠倒的局部搜索	SA	计划维护时间降低 89%, 计划编制总时间减少 98%	Matlab
局部搜索	[39]	工程模型修正技术	对当前最优解增加记忆、返回功能; 增加淬火过程	SAA	前 3 阶计算、试验模态频率相对误差降低 12.74%, MAC 值升高到 0.32	MSC Patran/Nastran 平台; DMAP 语言

表 7 算法对比

比较项目	ACO	GA	PSO	SA
解的形式	找到目标后留下信息素的路径适应值	染色体编码的适应值	粒子的位置和速度适应值	固体内能
代理	蚂蚁	染色体	粒子	粒子
信息交流	有	无	有	无
解的更新方式	信息交流后选择下一节点	染色体交叉、变异	信息交流后调整位置、速度信息	温度变化
解的多样性	无	染色体交叉、变异	无	Metropolis 准则

4 结束语

对云计算作了一个概述, 阐述了云计算环境下的任务调度模型, 并对其进行了分析, 再由任务调度引出 4 个比较完善、具有代表性的任务调度算法 ACO、GA、PSO、SA, 分别对它们做了详细分析, 包括基本思想、算法特点以及可改进的方式, 针对各个算法的特点, 归纳了一些各个算法两两之间可以进行改进以及融合方法; 再对相比前面 4 种算法, 比较新颖的 ACO、ICA、BA、CSO 做了分析, 包括算法的基本思想、特点和可改进的方式; 由于从理论想要应用到实践, 就必须要经过实验对其进行验证, 而直接投入实际环境中进行实验会花费大量的时间、资源等.

随着信息技术的不断进步, 物联网在我们的日常生活中发挥着越加重要的作用, 人们生活水平的提高, 对各种物联网应用的要求也越来越高, 对云计算的挑战也随之到来. 在传统的云计算中, 每一次服务交付都要将计算过程上传到云, 有限的带宽和网络资源被大量数据传输所占用, 然而云是远离用户的, 这就直接造成了云计算所面临的第一大难点: 网络延迟. 此外, 物联网设备和传感器的功率是有限的, 大量的数据传输也给物理设备带来了压力, 这也是云计算所面临的第二大难点: 功耗成本.

为了延长设备的使用寿命, 就很有必要通过将计算调度到具有更高功率和计算能力的设备来平衡功耗, 即通过改善任务调度可以有效地降低云计算的功耗成本. 因此, 任务的调度和处理分配就成为了云计算需要解决的关键问题.

虽然通过调度方案的优化可以减少云计算的功耗, 但是由于云计算服务中, 每一次服务交付都要将计算过程上传到云端, 即使改进了调度方案, 还是无法有效地解决网络延迟的问题.

为了解决云计算网络延迟的问题, 边缘计算[40]出现了. 边缘计算不同于云计算, 很直观地, 边缘计算就是在网络的“边缘”进行服务交付, 执行的数据计算和存储在用户附近. 相比较云计算与用户的距离更近近, 最直观导致的结果就是降低了网络延迟、网络的带宽需求、数据计算或存储期间的传输延迟, 并且有效地降低了物理设备损耗速度. 此外, 与云计算相比, 边缘

© 中国科学院软件研究所 http://www.c-s-a.org.cn

计算可以将计算和通信开销从具有有限电池或电源的节点迁移到具有大量功率资源的边缘节点.

2016 年 5 月, 美国自然科学基金委 (National Science Foundation, NSF) 在计算机系统研究中将边缘计算替换云计算, 列为突出领域[41]. 虽然其发展和技术还不太成熟, 但是由于其自身的特点, 边缘计算将会成为继云计算之后的又一大热点领域. 7 个关键技术包括: 网络、隔离技术、体系结构、边缘操作系统、算法执行框架、数据处理平台以及安全和隐私; 未来几年迫切需要解决的 6 个方向问题: 编程模型、软硬件选型、基准程序与标准、动态调度、与垂直行业的紧密结合以及边缘节点的落地[41].

参考文献

1 Wu FH, Wu QB, Tan YS. Workflow scheduling in cloud: A survey. The Journal of Supercomputing, 2015, 71(9): 3373–3418. [doi: 10.1007/s11227-015-1438-4]

2 Sabi HM, Uzoka FME, Mlay SV. Staff perception towards cloud computing adoption at universities in a developing country. Education and Information Technologies, 2018, 23(5): 1825–1848. [doi: 10.1007/s10639-018-9692-8]

3 Al-Dhuraibi Y, Paraiso F, Djarallah N, *et al*. Elasticity in cloud computing: State of the art and research challenges. IEEE Transactions on Services Computing, 2018, 11(2): 430–447. [doi: 10.1109/TSC.2017.2711009]

4 Weinman J. The economics of pay-per-use pricing. IEEE Cloud Computing, 2018, 5(5): 101–c3. [doi: 10.1109/MCC.2018.053711671]

5 刘永. 云计算技术研究综述. 软件导刊, 2015, 14(9): 4–6.

6 Xu ZC, Liang WF, Xia QF. Efficient embedding of virtual networks to distributed clouds via exploring periodic resource demands. IEEE Transactions on Cloud Computing, 2018, 6(3): 694–707. [doi: 10.1109/TCC.2016.2535215]

7 Li RX, Shen CL, He H, *et al*. A lightweight secure data sharing scheme for mobile cloud computing. IEEE Transactions on Cloud Computing, 2018, 6(2): 344–357. [doi: 10.1109/TCC.2017.2649685]

8 Anushree B, Arul Xavier VM. Comparative analysis of latest task scheduling techniques in cloud computing environment. Proceedings of 2018 Second International Conference on Computing Methodologies and Communication. Erode, India. 2018. 608–611.

9 王治东. 云计算环境下任务调度研究综述. 中国新通信, 2017, 19(9): 78. [doi: 10.3969/j.issn.1673-4866.2017.09.066]

10 Arabnejad H, Barbosa JG. List scheduling algorithm for heterogeneous systems by an optimistic cost table. IEEE Transactions on Parallel and Distributed Systems, 2014, 25(3): 682–694. [doi: 10.1109/TPDS.2013.57]

11 Wang HJ, Sinnen O. List-scheduling versus clus-ter-scheduling. IEEE Transactions on Parallel and Distributed Systems, 2018, 29(8): 1736–1749. [doi: 10.1109/TPDS.2018.2808959]

12 Sonmez O, Yigitbasi N, Abrishami S, *et al*. Performance analysis of dynamic workflow scheduling in multicluster grids. Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing. Chicago, IL, USA. 2010. 49–60.

13 Singh P, Dutta M, Aggarwal N. A review of task scheduling based on meta-heuristics approach in cloud computing. Knowledge and Information Systems, 2017, 52(1): 1–51. [doi: 10.1007/s10115-017-1044-2]

14 徐健锐, 朱会娟. 基于自适应惩罚函数的云工作流调度协同进化遗传算法. 计算机科学, 2018, 45(8): 105–112.

15 李超, 戴炳荣, 旷志光, 等. 云计算环境下基于改进遗传算法的多维约束任务调度研究. 小型微型计算机系统, 2017, 38(9): 1945–1949. [doi: 10.3969/j.issn.1000-1220.2017.09.005]

16 任金霞, 黄艺培, 钟小康. 基于遗传算法的云任务调度改进算法. 江西理工大学学报, 2018, 39(3): 90–94.

17 张浩为, 谢军伟, 张昭建, 等. 基于混合自适应遗传算法的相控阵雷达任务调度. 兵工学报, 2017, 38(9): 1761–1770. [doi: 10.3969/j.issn.1000-1093.2017.09.013]

18 王国豪, 李庆华, 刘安丰. 多目标最优化云工作流调度进化遗传算法. 计算机科学, 2018, 45(5): 31–37, 48.

19 Dorigo M, Gambardella LM. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 53–66. [doi: 10.1109/4235.585892]

20 Dorigo M, Stützle T. Ant colony optimization: Overview and recent advances. Gendreau M, Potvin JY. Handbook of Metaheuristics. Cham: Springer, 2019. 311−351.

21 李琳, 应时, 董波. 一种求解面向服务软件部署优化问题的多目标蚁群算法. 中南大学学报 (自然科学版), 2017, 48(9): 2376–2387. [doi: 10.11817/j.issn.1672-7207.2017.09.017]

22 Moon YJ, Yu HC, Gil JM, *et al*. A slave ants based ant colony optimization algorithm for task scheduling in cloud computing environments. Human-Centric Computing and Information Sciences, 2017, 7(1): 28. [doi: 10.1186/s13673-017-0109-2]

© 中国科学院软件研究所 http://www.c-s-a.org.cn

23 Mao XM. Study on ant colony optimization algorithm for “one-day tour” traffic line. Cluster Computing, 2019, 22(S2): 3673–3680. [doi: 10.1007/s10586-018-2217-9]

24 Zhang Q, Zhang CS. An improved ant colony optimization algorithm with strengthened pheromone updating mechanism for constraint satisfaction problem. Neural Computing and Applications, 2018, 30(10): 3209–3220. [doi: 10.1007/s00521-017-2912-0]

25 林哲骋, 许力. 一种应用于激光焊接轨迹规划的改进蚁群算法. 焊接学报, 2018, 39(1): 107–110. [doi: 10.12073/j.hjxb.2018390024]

26 Kennedy J, Eberhart R. Particle swarm optimization. Proceedings of the IEEE International Conference on Neural Networks. Perth, Australia. 1995. 1942–1948.

27 Sammut C, Webb GI. Encyclopedia of Machine Learning and Data Mining. Boston, MA, USA: Springer, 2017: 1–10.

28 宋寒冰. 云计算环境下基于改进 PSO 算法的任务调度研究[硕士学位论文]. 长春: 吉林大学, 2017.

29 Guo LZ, Wang YJ, Zhao SG, *et al*. Particle swarm optimization embedded in variable neighborhood search for task scheduling in cloud computing. Journal of Donghua University (English Edition), 2013, 30(2): 145–152.

30 赵莉. 基于改进量子粒子群算法的云计算资源调度. 南京理工大学学报, 2016, 40(2): 223–228.

31 Liu L, Shan L, Jiang C, *et al*. Parameter identification of the fractional-order systems based on a modified PSO algorithm. Journal of Southeast University (English Edition), 2018, 34(1): 6–14.

32 方昕. 一种新型启发式 PSO 算法求解市区最优路径规划研究. 计算机与数字工程, 2018, 46(2): 270–275. [doi: 10.3969/j.issn.1672-9722.2018.02.013]

33 邓先礼, 魏波, 曾辉, 等. 基于多种群的自适应迁移 PSO 算法. 电子学报, 2018, 46(8): 1858–1865. [doi: 10.3969/j.issn.0372-2112.2018.08.009]

34 高健, 高培. 一种改进的模拟退火算法求解中学排课问题. 工业计量, 2018, 28(4): 72–75, 91.

35 姚壹壹, 王玲鹏, 金科扬, 等. 基于模拟退火算法最优物流配送问题的应用. 宁波工程学院学报, 2018, 30(1): 39–44. [doi: 10.3969/j.issn.1008-7109.2018.01.007]

36 刘新星, 张贞凯, 费晓. 模拟退火算法的共享孔径多波束形成. 电光与控制, 2018, 25(11): 57–61. [doi: 10.3969/j.issn.1671-637X.2018.11.011]

37 董明佶, 林宝军, 刘迎春, 等. 基于多目标模拟退火算法的导航卫星激光星间链路拓扑动态优化. 中国激光, 2018, 45(7): 0706004.

38 黄海松, 刘凯, 初光勇. 改进模拟退火算法在柔性调度中的应用. 组合机床与自动化加工技术, 2018, (2): 148–151, 156.

39 杜大华, 贺尔铭, 李磊. 改进模拟退火算法的喷管动力学模型修正. 宇航学报, 2018, 39(6): 632–638.

40 Yu W, Liang F, He XF, *et al*. A survey on the edge computing for the internet of things. IEEE Access, 2018, 6: 6900–6919. [doi: 10.1109/ACCESS.2017.2778504]

41 施巍松, 张星洲, 王一帆, 等. 边缘计算: 现状与展望. 计算机研究与发展, 2019, 56(1): 69–89. [doi: 10.7544/issn1000-1239.2019.20180760]

© 中国科学院软件研究所 http://www.c-s-a.org.cn