基于稳定匹配的容器部署策略的优化

施 超 谢在鹏 柳 晗 吕 鑫
(河海大学计算机与信息学院 南京 211100)

摘 要 Docker的发展使得操作系统级虚拟化的容器渐渐兴起,容器即服务(CaaS)也越来越普及。随着容器技术的发展,容器将成为云环境中的主要部署模型,但针对容器的整合部署技术还未得到广泛的研究。容器化云环境中的容器数量众多,如何将众多的容器部署到合适的虚拟机以降低数据中心能耗,成为了一个亟待解决的问题。因此,文中创新性地将机器学习中的几种相似度计算方法作为稳定匹配算法的偏好规则,同时将已经拟分配过容器的虚拟机继续加入偏好列表,从而将一对一的稳定婚姻匹配算法改进为多对一的稳定匹配,解决了将容器整合到虚拟机上的初始化部署问题。仿真实验结果表明,采用优化的稳定匹配算法来初始化将部署容器时,不仅SLA违规较低,而且比FirstFit,MostFull以及Random算法分别约节能12.8%,34.6%和30.87%,其中使用欧氏距离作为稳定匹配算法偏好规则的节能效果最好。

关键词 容器云,容器即服务,稳定匹配,容器整合,能耗
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.04.021

Optimization of Container Deployment Strategy Based on Stable Matching

SHI Chao XIE Zai-peng LIU Han LV Xin
(College of Computer and Information, Hohai University, Nanjing 211100, China)

Abstract With the development of Docker, virtualization containers of operating system level are on the rise, and container-as-a-service(CaaS) is also becoming more and more popular. With the development of container technology, the container will become the main deployment model in the cloud environment, but the integrated deployment technology for the container has not been widely studied. In the cloud environment, how to deploy a large number of containers to a suitable virtual machine to reduce the energy consumption of the data center becomes an problem which needs to to be solved urgently. Therefore, several similarity calculation methods in machine learning are used as the preference rules of the stabilization matching algorithm, and the virtual machines that have been allocated to the container are added to the preference list at the same time, which makes the one-to-one stable marriage matching algorithm update to many-to-one stable match, solving the initial deployment problem of integrate container into the virtual machine. The experimental results show that the optimal storage efficiency is about 12.8%, 34.6% and 30.87% compared with the FirstFit, MostFull and Random methods respectively, and when the Euclidean distance is used as the preference rules of stabilization matching algorithm, the performance of energy saving is the best.

Keywords Containerized cloud, Container as a service, Stable matching, Container consolidation, Energy consumption

1 引言

近年来,Shepherd[1]提出了容器即服务(CaaS)的概念,以解决平台即服务(PaaS)环境中被开发的应用程序受平台规范限制的问题。容器技术[2]作为CaaS云模型的实现基础,提供了一种隔离的虚拟环境,而不需要中间监控介质,如虚拟机管理程序。容器提高了云资源利用率的效率,因为它们比虚拟机更密集。此外,容器共享主机操作系统内核,其通信通过系统标准调用来执行,这比虚拟机(VM)的基于管理程序的通信快得多。在如今云应用程序普及的情况下,容器技术被认为是云计算发展的下一个重要方向[3]。

随着容器(使用操作系统虚拟化)如Docker等系统的普及,容器技术成为部署、打包和管理应用程序的主要虚拟化技术[3]。与虚拟机相比,容器不仅是一种低开销的虚拟化技术,而且提高了性能。与针对虚拟化云数据中心的计算和网络资源的能效被广泛研究不同,只有少数学者调查了容器的节能

到稿日期:2017-05-24 返修日期:2017-06-17 本文受国家自然科学基金面上项目(61272543),NSFC-广东联合基金重点项目(U1301252)资助。
施 超(1991-),男,硕士生,主要研究方向为容器化云计算;谢在鹏(1982-),男,博士,讲师,CCF会员,主要研究方向为云计算与大数据平台、嵌入式系统架构,E-mail:zxiehhu@163.com(通信作者);柳 晗(1994-),男,主要研究方向为云计算;吕 鑫(1983-),男,博士,讲师,主要研究方向为密码学、网络信息安全。

管理问题。Ghribi[4]讨论了数据中心中容器的能源效率管理问题,但是所提算法的有效性尚未得到评估。Spicuglia 等[5]提出的 OptiCA 简化了大数据应用在 CaaS 中的部署。Dong 等[6]提出了一个贪婪的容器放置方案,首先分配容器到最节能的机器。仿真结果显示,与随机调度方案相比,Dong 提出的方案可以显著降低能耗。Yaqub 等[7]指出 PaaS 中的部署模型基于托管各种软件服务的操作系统级容器,其主要贡献是将服务整合问题建模为多维装箱问题,并应用了模拟退火等元启发式算法。Ding[8]提出了一种针对云计算系统的资源推荐方法,该方法整合了价格效用、多属性匹配度量和群体客户评估。Jordi[9]简单介绍了获得 2012 年诺贝尔经济学奖的稳定匹配算法,包括双边匹配和资源分配的应用场景。赵道致等[10]根据供求主体的心理预期,基于前景理论的满意度提出了云制造资源双边匹配机制。

尽管在企业环境中云资源提供商对容器的兴趣激增,但是对容器的性能研究仍然是一个开放的课题,尤其是容器整合方面的研究还处于初级阶段。文献[4-7]初步调查了容器的资源整合问题,但是在算法层面上还没有深入研究。文献[8-10]利用推荐方法中的相似性度量或双边匹配思想对资源进行有效的分配,但当容器化云计算中的容器数量众多时如何优化这些资源的分配还处于初步探究阶段。因此,本文的主要目的是提出一种容器整合阶段的容器部署算法,创新性地将机器学习中的几种相似度计算方法作为稳定匹配算法的偏好规则,同时将已经拟分配过容器的虚拟机继续加入偏好列表,从而将一对一的稳定婚姻匹配算法改进为多对一的稳定匹配,以实现容器化云环境的资源分配,降低数据中心的能耗。通过在服务器和虚拟机上提升资源利用率,来解决容器化云环境中的能耗优化问题。

2 容器技术

系统级虚拟化中存在仿真硬件的“虚拟机”(VM),其具备在独立内核的主机管理程序之上运行自己的操作系统(OS)的能力。在操作系统级别上,存在与主机共享相同内核的所谓容器,被定义为轻量级虚拟环境,其在工作负载之间提供一层隔离,而没有基于管理程序的虚拟化开销。

基于这两种虚拟化类型,将研究云环境中的资源管理的技术分为 3 个主要类别,即轻量级容器、虚拟机和混合。轻量级容器类别包含假定任务/应用程序在容器内执行的技术。在虚拟机组中,应用程序在虚拟机内执行。在混合类别中,应用程序在容器内部执行,而容器映射在虚拟机而非服务器上。

操作系统级虚拟化或容器化本身分为两种不同的类型,即 OS 容器[11]和应用容器[12]。OS 容器可以共享主机操作系统的内核的 VM,同时提供隔离的用户空间。共享内核提高容器对资源的利用率,并减少启动和关闭容器的开销。表 1 中的 LXC 和 OpenVZ 就是该类容器的示例。与运行多个进程和服务的 OS 容器不同,应用容器专用于单个进程,并构建在 OS 容器上。每个容器中的单个进程是运行驻留应用程序的进程。应用容器可以被认为是云时代的一个革新,因为容器是轻量级的,更容易配置和管理,可以大大减少启动时间。表 1 中的 Docker 和 Rocket 是应用容器的示例。

表 1 硬件虚拟化分类及其代表技术

Table1 Hardware virtualization classification and representative technologies

虚拟化	组件	与硬件通信的方式	代表技术
操作系统级	OS 容器(轻量级虚拟机)	系统标准通信	LXC,OpenVZ
	应用容器		Docker,Rocket
系统级	虚拟机(VM)	Hypervisor	KVM,VMWare

在容器化云环境中,容器数量众多,容器整合技术可以有效减少启动的物理机数量,提高资源利用率,降低数据中心的功耗。容器越来越受欢迎,但针对容器的整合部署技术还没有得到广泛研究。本文重点研究操作系统级容器的部署技术。

3 容器整合

Google 和 Amazon Web Services 引入的容器即服务(CaaS)云模型越来越受欢迎,将成为主要的云服务模型之一。研究表明,相比于 Docker(容器)部署,使用 VM 容器混合配置能获得相近甚至更好的性能[13]。然而,如何提高 CaaS 数据中心的能源效率问题还没有得到深入研究。

服务器仍然是数据中心中最大的能源消费者[14]。因此,减少运行的服务器数量是研究的一大热点,可以通过容器整合来实现。与其他整合问题类似,容器整合问题也应被视为如下多阶段问题:何时触发迁移,要迁移哪个容器,迁移到哪里。

通过容器的整合,可以从过载的主机迁移部分容器到资源空闲的虚拟机上,并关闭一部分欠载的物理机,从而减少启动的物理机数量,降低云数据中心的能耗。通过在虚拟机上整合容器,来提高这种新部署模型中服务器的能源效率。

数据中心在时刻 t 的能耗计算如下:

$$P_{dc}(t)=\sum_{i=1}^{N_s}P_i(t) \tag{1}$$

主要通过考虑 CPU 的功率的利用率来估计服务器的能耗,因为这是在其使用率方面呈现最大的能耗变化的部分。对于每个服务器 i,CPU 的利用率 $(U_i,t)=\sum_{j=1}^{N_{vm}}\sum_{k=1}^{N_c}U_{c(k,j,i)}(t)$,并且通过以下公式来估计服务器的能耗:

$$P_i(t)=\begin{cases}P_i^{\text{idle}}+(P_i^{\max}-P_i^{\text{idle}})*U_{i,t}, & N_{vm}>0\\ 0, & N_{vm}=0\end{cases} \tag{2}$$

在容器化云计算模型中,假设有 M 个容器、N 个虚拟机和 K 个服务器,则可以将能耗问题描述成:

$$\min(P_{dc}(t)=\sum_{i=1}^{N_s}P_i(t)) \tag{3}$$

该问题有以下限制条件:

$$\sum_{j=1}^{N_{vm}}U_{vm_{j,i}}(t)<S_{(i,r)},\forall i\in[1,N_s],\forall r\in\{\text{CPU}\} \tag{4}$$

$$\sum_{j=1}^{N_{vm}}vm_{(j,i,r)}<S_{(i,r)},\ \forall i\in[1,N_s],\ \forall r\in\{\text{BW, MEM, DISK}\} \tag{5}$$

$$\sum_{k=1}^{N_c}U_{(k,j,i)}(t)<vm(j,i,r),\forall j\in[1,N_{vm}],\forall i\in[1,N_s],\forall r\in\{\text{CPU}\} \tag{6}$$

$$\sum_{k=1}^{N_c}c_{(k,j,i,r)}<vm_{(j,i,r)},\ \forall j\in[1,N_{vm}],\ \forall i\in[1,N_s],$$

$$\forall r \in \{\mathrm{BW},\mathrm{MEM},\mathrm{DISK}\} \tag{7}$$

在云计算中,SLA 违规是一个重要的指标。服务提供商需要在满足 SLA 的情况下提供一系列服务给用户:

$$SLA=\sum_{i=1}^{N_s}\sum_{j=1}^{N_{vm}}\sum_{p=1}^{N_c}\frac{\mathrm{CPU}_r(vm_{j,i},t_p)-\mathrm{CPU}_a(vm_{j,i},t_p)}{\mathrm{CPU}_r(vm_{j,i},t_p)} \tag{8}$$

其中,$P_{dc}(t)$表示 t 时刻数据中心的能耗,P_i^{idle} 表示服务器 i 的闲置功耗,$P_i^{\max}$ 表示服务器 i 的最大功耗,N_c 表示容器数量,N_{vm}表示虚拟机数量,N_s 表示服务器数量,$S_{(i,r)}$表示服务器 i 的资源大小,$U_{vm_{j,i}}(t)$表示运行在服务器 i 中的虚拟机 j 在 t 时刻的 CPU 利用率,而 $vm_{(j,i,r)}$则表示运行在服务器 i 上的虚拟机 j 的资源大小,$\mathrm{CPU}_r(vm_{j,i},t_p)$表示在 t_p 时刻服务器 i 上的虚拟机 j 请求的 CPU 资源,$\mathrm{CPU}_a(vm_{j,i},t_p)$表示 t_p 时刻分配给虚拟机 j 的 CPU。

使用优化工具包尽管可以找到上述问题的近似最优解,但是计算时间和复杂性随容器数量呈指数增长。因此,第 4 节提出和评估了稳定匹配算法在容器化云计算中的应用和优化,创新性地将机器学习中的几种相似度计算方法作为稳定匹配算法的偏好规则,同时将已经拟分配过容器的虚拟机继续加入偏好列表,从而将一对一的稳定婚姻匹配算法改进为多对一的稳定匹配,获得具有较少计算开销的近似最优解。

4　稳定算法应用到容器部署的策略优化

4.1　稳定婚姻算法

Alvin E. Roth 和 Lloyd S. Shapley 于 2012 年联合获得了诺贝尔经济学奖[15],他们对稳定分配理论和市场设计实践的贡献得到了肯定。稳定分配理论由一系列模型组成,用于研究分配问题,其中两个不相交的集必须匹配。例如,男性对妇女,工人对企业,学生对学校。如果没有任何子集能够通过仅在其间重新匹配来提高其所提出的匹配,则匹配是稳定的。稳定匹配意味着,在当前匹配中两者互为对方的最优选择而非第一选择,而在未匹配的元素中,不存在更优的匹配给任何一方。

一对一的稳定匹配过程可被描述为:对于两个不相交的个体集,其中每个人对来自另一组的一个子集严格地优先排序,每个个体至多可以与另一边的一个个体匹配。

一对一稳定算法的具体描述如算法 1 所示。

算法 1　稳定婚姻算法

```
FOR m =1 to M DO
companion[m]=NULL
  for w=1 to W DO
  companion[w]=NULL
  WHILE TRUE DO
    If there is no man m such that companion[m] =NULL THEN
      Return
    END IF
    Select such a man m;
    w= the first woman on m's list to whom m have not yet pro-
       posed;
    If companion[w]= NULL
      THEN
      companion[companion[w]]=NULL;
      companion[w]=m;
      companion[m]=w;
      END WHILE
```

4.2　稳定匹配算法的 3 种模式

稳定匹配问题(SM)一直是数学、运筹学、经济学和社会学等领域研究的热点问题[16]。稳定匹配问题通常以矩阵的形式出现,一般分为单边匹配、双边匹配和多边匹配。本文主要研究的是双边匹配问题中的稳定匹配算法。双边匹配问题的经典分类方法,主要是根据双边匹配理论的匹配稳定存在性而将其分为 3 种类型。

1)一对一匹配

每个对象最多可以与对方的一个成员匹配。最突出的实例是男女需要配偶的稳定婚姻问题。

2)多对一匹配

在其中一个集合中至少有一个对象可以匹配对方的多个对象,而在另一个集合中,每个对象都有一个匹配。一个典型的实例是学生入学问题,一个学生可以与一所大学匹配,而大学可以招收多个学生。

3)多对多匹配

两组集合中都至少有一个对象可以匹配另一组的多个成员。多对多匹配是最常见的问题类型,并且拥有许多示例,如在对等网络中创建伙伴关系。

4.3　容器和虚拟机的稳定匹配

将容器看成是一个集合,虚拟机看成是另一个集合,此时,容器集合和虚拟机集合就可以使用稳定匹配算法。但在云数据中心,容器的数量通常远远大于虚拟机的数量,因此不能直接应用一对一的稳定婚姻算法。

同时,在稳定婚姻算法中,如果子集遇到一个更好的匹配者,则其必须拒绝现在的匹配,因此那些已经有了匹配的子集可能会被拒绝,从而重新变成未匹配状态。这样的情形不太适合云计算,因为这意味着已经部署好的容器可能需要迁移,甚至部署一个容器时会导致所有已部署好的容器全部发生迁移并且重新部署,最坏的情况是这种迁移会不断迭代发生。

针对上述问题,本文对一对一的稳定婚姻算法进行了应用改进,第一轮将容器匹配到虚拟机上后,计算虚拟机的剩余资源,在下一轮中如果虚拟机的剩余资源足够部署其他容器,则依旧接受新的容器发送的部署请求。与传统稳定婚姻算法不同,改进算法第一轮接受了容器部署请求的虚拟机将被继续加入到下一轮容器集合的偏好列表中,直到所有的容器都得到了稳定的部署。根据该流程,一对一的稳定婚姻算法被扩展成了多对一模式的匹配算法。同时,本文采用"延迟接受稳定婚姻"算法,即在每一轮的选择匹配过程中,容器可以向自己最"偏好"的虚拟机发送部署请求,虚拟机如果有剩余计算空间和能力去支持该容器的部署,则选择同意部署。但是,该虚拟机并不是立即部署这个容器,而是等待其他容器发送请求,如果没有更合适的,则最终选择该容器进行部署;如果有比该容器更合适的,则拒绝该容器,接受新的容器,直到这一轮选中一个最适合的容器来部署。

容器选择哪个虚拟机发送部署请求,即容器的偏好规则

的选择，采用机器学习算法中协同过滤推荐算法的几种相似度计算方法来进行。

相似性的度量的方法有很多种，不同的度量方法的应用范围也不一样。常用的相似度的计算方法有：欧氏距离法、皮尔逊相关系数法、夹角余弦相似度法和 Tanimoto 度量法。

1)欧氏距离(Euclidean Distance)

欧氏距离是使用得较多的一种相似性度量方法，k-Means 中就是使用欧氏距离作为相似性的评价指标。最初，其用于计算欧几里德空间中两个点的距离，即假设 x,y 是 n 维空间的两个点，则它们之间的欧几里德距离：

$$d(x,y)=\sqrt{(\sum(x_i-y_i)^2)} \tag{9}$$

2)皮尔逊相关系数(Pearson Correlation)

皮尔逊相关系数主要用于量化两个量之间的依赖程度。根据 Pearson 分析，如果存在由 x_i 和 y_i 表示的 n 个样本的两个随机变量 X 和 Y，则使用式(10)计算相关系数，其中，$\bar{x}$和 $\bar{y}$ 分别表示 X 和 Y 的样本均值，r 的取值范围为[−1，+1]。

$$r(x,y)=\frac{\langle x_i-\bar{x},y_i-\bar{y}\rangle}{\|x_i-\bar{x}\|\ \|y_i-\bar{y}\|}=\frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2\sum_{i=1}^{n}(y_i-\bar{y})^2}} \tag{10}$$

3)余弦相似度(Cosine Similarity)

余弦相似度有着与皮尔逊相似度相同的性质，对量级不敏感，用于计算两个向量的夹角。夹角余弦的取值范围为[−1，1]，其值越大表示两个向量的夹角越小，也即两者越相似。

$$cosSim(x,y)=\frac{\langle x,y\rangle}{\|x\|\ \|y\|}=\frac{\sum_{i=1}^{n}x_{1i}x_{2i}}{\sqrt{\sum_{i=1}^{n}x_{1i}^2}\sqrt{\sum_{i=1}^{n}x_{2i}^2}} \tag{11}$$

4)Tanimoto 系数(Tanimoto Coefficient)

Tanimoto 系数也被称为 Jaccard 系数，是 Cosine 相似度的扩展，多用于计算文档数据的相似度。

$$T(x,y)=\frac{x\cdot y}{\|x\|^2\|y\|^2-x\cdot y}=\frac{\sum x_iy_i}{\sqrt{\sum x_i^2+\sqrt{\sum y_i^2}-\sum x_iy_i}} \tag{12}$$

针对容器选择哪个虚拟机发送部署请求即容器的偏好规则问题，本文实验采用几种不同的相似度计算方法；而虚拟机对容器的偏好规则，则是计算虚拟机的资源是否能够支持该容器的部署，并按照资源利用率的大小从高到低排序。容器部署到虚拟机的稳定匹配算法的具体描述如算法 2 所示。

算法 2 基于稳定婚姻的多对一容器初始化部署算法

```
Foreach object[c]=NULL
FOR c =1 to container DO
object[c]=NULL
  FOR vm=1 to VM DO
    object[w]=NULL
  WHILE TRUE DO
    IF there is no container c such that object[c]=NULL THEN
    Return
    END IF
    Select such a container c;
    vm=the first vm on c's list to which c have not yet proposed;
    IF object[vm]= NULL
    THEN
    vm. containerlist. sortByUtilization;
    object[object[vm]]=NULL
    object[vm]=m;
    object[c]=w;
    calculate remain[vm];
    IF(remain[vm]>0)
    BREAK;
END WHILE
```

1)将所有容器标记为尚未部署。若存在尚未部署的容器，则进入下一步。

2)每一个尚未部署的容器根据相似度度量方法计算适合自己部署的虚拟机排名，并对最合适的且从没有拒绝过自己的虚拟机发送部署请求。

3)在每一轮中，每一个虚拟机将正在发送部署请求的容器与自己当前延迟接受的容器进行比较，并根据自己的资源利用率排名来选择部署到自己空间的容器：如果该容器部署优于当前的容器部署的资源利用率，则抛弃当前容器；否则保留当前容器，拒绝发送部署请求的容器。

4)计算部署过容器的虚拟机的剩余资源，继续运行下一轮的容器部署的匹配算法，新容器(或者是在之前被拒绝部署的容器)可以继续向已经部署过容器但从来没有拒绝过自己的以及未部署容器的虚拟机发送部署请求。

5)被抛弃或者被拒绝的容器依然保留未部署标记。

6)重复上述步骤，直到不存在未部署容器。

5 实验结果和分析

为了评估容器云数据中心中调度和分配策略的性能，需要构建可扩展和可重复实验的评估环境。仿真技术能提供可重复和可控的环境，满足实验要求。ContainerCloudSim[17]是作为 CloudSim 仿真工具包的扩展而开发的，最新的 CloudSim4.0 版本支持容器的仿真。它为测试和评估资源管理技术提供了一个环境，支持容器化云计算环境的建模，如容器调度、布局和容器整合。

本文使用的数据集来自于 29 天内收集的 Google 云跟踪日志[18]的第二版本。日志由描述机器、作业和任务的数据表组成。此外，为了评估容器整合算法，使用从真实世界的工作负载跟踪获得的数据进行实验。工作负载数据来自 PlanetLab[19]基础设施监控到的 CoMon 项目的 CPU 利用率。

实验中，稳定匹配算法中的偏好规则分别使用了欧氏距离、余弦相似度以及 Tanimoto 系数 3 种相似度计算方法(利用皮尔逊相似度进行计算时，每次都需要计算样本均值以降低效率，因此不予考虑)。当偏好规则采用欧氏距离时，任务数量分别设置为 2000～8000，任务长度统一设置为 500～2000。本实验因为只考虑容器如何初始化部署到虚拟机上的

问题,不考虑容器的迁移问题,所以实验中禁止了容器迁移的模块。在 CloudSim4.0 上进行了多次实验,在上述 7 种任务数量下,分别使用 firstfit 算法和基于欧氏距离偏好规则的多对一稳定匹配(SM)算法初始化部署容器,实验结果如图 1 所示。可以看出,相比于 firstfit 算法,本文提出的算法在当前配置和应用场景下的平均能耗比降低了 45.13%。

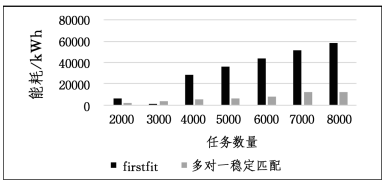

图 1　firstfit 与多对一稳定匹配算法的能耗比较

Fig. 1　Energy consumption comparison of firstfit and many-to-one stable matching algorithm

针对 3 种不同偏好规则下的稳定匹配算法,选取任务数量分别为 4000,6000 和 8000 共 3 组仿真配置,任务长度设置为 500 到 2000,每一组配置应用不同的偏好规则进行了多次实验,实验结果如图 2 所示。由图可知,使用欧氏距离的方法的能耗平均降低了 45.54%,使用 Tanimoto 方法的能耗平均降低了 45.37%,使用余弦相似度方法的能耗平均降低了 44.74%。在其他配置场景下的多次实验结果同样证明,使用基于欧氏距离的方法作为多对一 SM 算法的偏好规则时能最大程度地降低能耗。

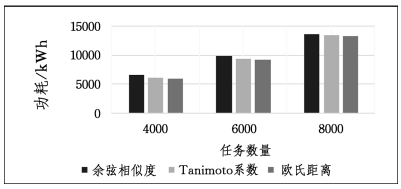

图 2　3 种偏好规则下的稳定匹配算法的能耗比较

Fig. 2　Energy consumption comparison of stable matching algorithm under three kinds of preference rules

将仿真实验中的配置 A 到配置 H 的任务大小取值的跨度分别设置为 500～3000,500～4000,500～5000,500～6000,500～7000,500～8000,500～9000,500～10000。在 A,C,D,F,G 这 5 组中,基于 Tanimoto 偏好规则的 SM 算法的能耗都低于 firstfit 算法的能耗,且在 B,E,H 3 组实验中的能耗也相差不多。如图 3 所示,重复多次实验,总体上,对于不同大小的任务跨度,firstfit 算法的能耗使用情况波动较大;而本文提出的算法基本稳定,能有效解决云计算环境中任务大小多变的实际容器部署问题,降低了数据中心的整体能耗。

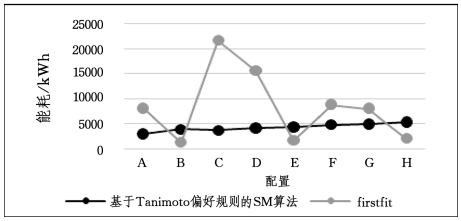

图 3　不同大小的任务跨度算法的能耗比较

Fig. 3　Energy consumption comparison of different algorithms

考虑所有不同的任务大小、任务数量以及数据中心的容器、虚拟机以及物理机数量等情况,重复多次实验并取均值,实验结果如图 4 所示。可以看出,在同等实验条件下初始化部署容器时,多对一 SM 算法分别比 firstfit,MostFull,Random 算法平均约降低了 12.8%,34.6%和 30.87%的能耗。

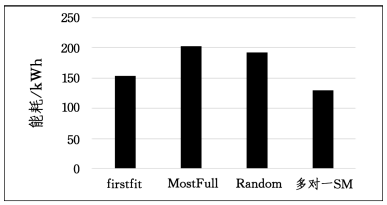

图 4　firstfit,MostFull,Random 和多对一 SM 算法的能耗比较

Fig. 4　Energy consumption comparison of firstfit, MostFull, Random and many-to-one SM algorithm

在云计算领域,SLA 是衡量算法性能和有效性的一个重要指标。根据式(8)和实验,与文献[20]中的 3 种算法相比,多对一 SM 算法只比 Random 算法的 SLA 违规高,而比 firstfit 和 MostFull 两种容器部署算法的 SLA 违规都要低。实验结果如图 5 所示。

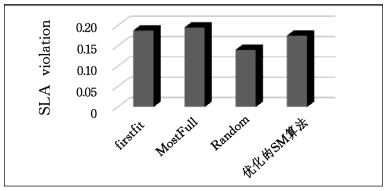

图 5　firstfit,MostFull,Random 和多对一 SM 算法的 SLA 比较

Fig. 5　SLA comparison of firstfit, MostFull, Random and many-to-one SM algorithm

结束语　提高云数据中心的能源利用效率是一场持久战,有效措施包括增加云提供商的投资回报率,减少加速全球变暖的二氧化碳的排放量等。本文通过将机器学习中的相似度计算方法作为稳定匹配算法的偏好规则,并且在下一轮继续将已经拟分配容器的虚拟机加入偏好列表,使得一对一稳定婚姻匹配被优化成多对一稳定匹配,解决了 CaaS 环境中的容器初始化部署问题,降低了能效。结果表明,使用多对一 SM 算法的能耗比 firstfit,MostFull 以及 Random 算法的能耗平均约降低了12.8%,34.6%和 30.87%。基于不同的偏好规则,将欧氏距离作为偏好规则时效果最佳。本文主要研究容器的初始化部署问题,未来可以考虑进一步改进该算法以解决因负载均衡容器发生迁移情况后的容器整合问题。

参 考 文 献

[1] SHEPHERD D. Containers as a Service(CaaS) is the cloud operating system-i build the cloud[EB/OL]. http://www.ibuildthecloud.com/blog/2014/08/19/containers-as-a-service-caas-is-the-cloud-operating-system.

[2] DUA R, RAJA A R, KAKADI D. Virtualization vs Containerization to Support PaaS[C]// 2014 IEEE International Conference on Cloud Engineering(IC2E). IEEE, 2014: 610-614.

[3] RUAN B, HUANG H, WU S, et al. A Performance Study of Containers in Cloud Environment[C]// Advances in Services

Computing: 10th Asia-Pacific Services Computing Conference (APSCC 2016). Springer International Publishing, 2016: 343-356.

[4] VAKILINIA S, HEIDARPOUR B, CHERIET M. Energy efficient resource allocation in cloud computing environments[J]. IEEE Access, 2017, 4(99): 8544-8577.

[5] SPICUGLIA S, CHEN L Y, BIRKE R, et al. Optimizing capacity allocation for big data applications in cloud datacenters[C]// Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management(IM 2015). 2015: 511-517.

[6] DONG Z, ZHUANG W, ROJAS-CESSA R. Energy-aware scheduling schemes for cloud data centers on Google trace data[C]// 2014 IEEE Online Conference on Green Communications (OnlineGreencomm 2014). 2014: 1-6.

[7] YAQUB E, YAHYAPOUR R, WIEDER P, et al. Metaheuristics-based planning and optimization for SLA-Aware resource managementin PaaS clouds[C]// 7th IEEE/ACM International Conferenceon Utility and Cloud Computing (UCC 2014). 2014: 288-297.

[8] DING S, XIA C, CAI Q, et al. QoS-aware resource matching and recommendation for cloud computing systems [J]. Applied Mathematics and Computation, 2014, 247(C): 941-950.

[9] MASSÓ J. The theory of stable allocations and the practice of market design: The Nobel Prize in Economics 2012 for Alvin E. Roth and Lloyd S. Shapley[J]. Contributions to Science, 2015, 11: 103-112.

[10] ZHAO D Z, LI R. The Bilateral Matching Mechanism of Cloud Manufacturing Resources Considering the Main Body's Psychological Expectation[J]. Control and Decision, 2017, 32(5): 871-878. (in Chinese)
赵道致, 李锐. 考虑主体心理预期的云制造资源双边匹配机制[J]. 控制与决策, 2017, 32(5): 871-878.

[11] PRICE D, TUCKER A. Solaris Zones: Operating system support for consolidating commercial workloads [C] // 18th USENIX Conference on System Administration (LISA 2004). 2004: 241-254.

[12] NAGY G. Operating system containers vs. application containers [OL]. https://blog. risingstack. com/operating-system-containers-vs-application-containers.

[13] ALI Q. Scaling web 2. 0 applications using Docker containers on vSphere 6. 0 [OL]. http://blogs. vmware. com/performance/2015/04/scaling-web-2-0-applications-using-docker-containers-vsphere-6-0. html.

[14] GREENBERG A, HAMILTON J, MALTZ D A, et al. The cost of a cloud: research problems in data center networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 39(1): 68-73.

[15] CARRERAS J M. The theory of stable allocations and the practice of market design: The Nobel Prize in Economics 2012 for Alvin E. Roth and Lloyd S. Shapley[J]. Contributions to Science, 2015, 11(1): 103-112.

[16] LEI L Z. An abstract argumentation based study of stable matching problems[D]. Hangzhou: Zhejiang University, 2014. (in Chinese)
雷丽赟. 基于抽象论辩理论的稳定匹配问题研究[D]. 杭州: 浙江大学, 2014.

[17] PIRAGHAJ S F, DASTJERDI A V, CALHEIROS R N, et al. ContainerCloudSim: An environment for modeling and simulation of containers in cloud data centers[J]. Software: Practiceand Experience, 2017, 47(4): 505-521.

[18] MISHRA A K, HELLERSTEIN J L, CIME W, et al. Towards characterizing cloud backend workloads: insights from Google compute clusters[J]. ACM SIGMETRICS Performance Evaluation Review, 2010, 37(4): 34-41.

[19] PARK K, PAI V S. CoMon: Amostly-scalable monitoring system for planetlab[J]. SIGOPS Operating System Review, 2006, 40(1): 65-74.

[20] PIRAGHAJ S F, DASTJERDI A V, CALHEIROS R N, et al. A framework and algorithm for energy efficient container consolidation in cloud data centers[C]// 2015 IEEE International Conference on Data Science and Data Intensive Systems (DSDIS). IEEE, 2015: 368-375.

(上接第 121 页)

[9] MOTTER A E, LAI Y C. Cascade-based attacks on complex networks[J]. Plysical Review E, 2002, 66(6): 065102.

[10] WANG W X, CHEN G R. Universal robustness characteristic of weighted networks against cascading failure[J]. Physical Review E, 2008, 77(2): 026101.

[11] LI T, PEI W J, WANG S P. Optimal traffic routing strategy on scale-free complex networks[J]. Acta Phys. Sin., 2009, 58(9): 5903-5910. (in Chinese)
李涛, 裴文江, 王少平. 无标度复杂网络负载传输优化策略[J]. 物理学报, 2009, 58(9): 5903-5910.

[12] LI C D, DENG Y, YUAN Z F, et al. Dynamic information-based load reallocation strategy for cascading failure networks [J]. Journal of South China University of Technology (Natural Science Edition), 2016, 44(5): 22-28. (in Chinese)
李从东, 邓原, 原智峰, 等. 基于动态信息的级联失效负载重分配策略[J]. 华南理工大学学报, 2016, 44(5): 22-28.

[13] DING L, ZHANG S Y. Cascading failures-oriented weighting strategies on complex networks[J]. Control and Decision, 2013, 28(9): 1399-1408. (in Chinese)
丁琳, 张嗣瀛. 面向级联失效的复杂网络加权策略[J]. 控制与决策, 2013, 28(9): 1399-1408.

[14] LI Z, GUO Y H, XU G A, et al. Analysis of cascading dynamics in complex networks with an emergency recovery mechanism [J]. Acta Phys. Sin., 2014, 63(15): 158901. (in Chinese)
李钊, 郭燕慧, 徐国爱, 等. 复杂网络中带有应急恢复机理的级联动力学分析[J]. 物理学报, 2014, 63(15): 158901.

[15] TRAN H T, BALCHANOS M, DOMERCAN J C, et al. A framework for the quantitative assessment of performance-based system resilience[J]. Reliability Engineering and System Safety, 2017(158): 73-84.

[16] GOH K I, KAHNG V, KIM D. Universal behavior of load distribution in scale-free network[J]. Physical Review Letters, 2001, 87(27): 278701.