Artificial Intelligence and Robotics Research 人工智能与机器人研究, 2017, 6(2), 41-55
Published Online May 2017 in Hans. http://www.hanspub.org/journal/airr
https://doi.org/10.12677/airr.2017.62006

Hans汉斯

Research of Multi-Objective Optimization Based Algorithm for Docker-Microservices Placement

Tianyu Xia[1], Jiangqin Xu[2*], Min Jiang[1]

[1]Fujian Key Lab of the Brain-Like Computing and Applications, Xiamen University, Xiamen Fujian
[2]College of Foreign Language and Cultures, Xiamen University, Xiamen Fujian
Email: xiatianyu@stu.xmu.edu.cn, *xujiangqin@xmu.edu.cn, minjiang@xmu.edu.cn

Received: May 7th, 2017; accepted: May 22nd, 2017; published: May 25th, 2017

Abstract

Docker is an open-source cloud computing application container engine, because it can make a large number of applications run on the existing server, thus attracting a wide range of attention. Combining Docker technology with micro services can significantly improve performance, but it also brings about the problem of how to effectively deploy. In this paper, an algorithm called MOMDA-ABC is proposed based on distributed estimation algorithm and artificial bee colony algorithm. The algorithm can optimize the communication distance and host number between Docker containers that deploy micro services, which can improve the performance of cloud computing platform effectively. The experimental results also prove the effectiveness of the method.

Keywords

Docker, Micro-Services, Artificial Bee Colony Algorithm, Estimation of Distribution Algorithm

基于多目标优化的Docker-微服务部署研究

夏天宇[1]，徐姜琴[2*]，江　敏[1]

[1]福建省类脑计算技术及应用重点实验室(厦门大学)，福建　厦门
[2]厦门大学外文学院，福建　厦门
Email: xiatianyu@stu.xmu.edu.cn, *xujiangqin@xmu.edu.cn, minjiang@xmu.edu.cn

收稿日期：2017年5月7日；录用日期：2017年5月22日；发布日期：2017年5月25日

*通讯作者。

文章引用: 夏天宇, 徐姜琴, 江敏. 基于多目标优化的 Docker-微服务部署研究[J]. 人工智能与机器人研究, 2017, 6(2): 41-55. https://doi.org/10.12677/airr.2017.62006

摘 要

Docker是一个开源的云计算应用容器引擎，由于可以使数量巨大的应用程序在已有的服务器上运行，因此受到广泛的关注。将Docker技术与微服务相结合可以显著改善性能，但是也带来了如何有效部署的问题。本文在分布式估计算法和人工蜂群算法的基础上，提出了一个称为MOMDA-ABC的算法。该算法可以优化部署微服务的Docker容器间的通信距离和主机数，从而使得云计算平台的性能有效提升。实验结果也证明该方法的有效性。

关键词

Docker，微服务，人工蜂群算法，分布估计算法

1. 引言

云计算(Cloud Computing)，简单来说就是一种互联网的计算方式，根据用户需求为用户提供各种基于互联网的软硬件服务。大家所熟知的云计算几种服务模式：基础设施即服务、平台即服务以及软件即服务，很好的将云计算划分成各类按需提供服务的形式。当然，云计算能够提供超大规模的计算能力，能够为公司提供超高的可靠性和扩展性，保证数据的安全和对资源的伸缩利用，除此之外费用还极其廉价。但是云计算在使用过程中出现的问题：1) 无法提供多实体资源的切分，2) 计算机使用效率不高，3) 定制服务难满足需求。所以需要虚拟化技术来填补云计算中资源粒度划分问题。

虚拟化(Virtualization)，简而言之就是将一台计算机按照不同需求将实体资源(CPU、内存、硬盘存储空间和网络等)切割开，并且相互独立互不影响，这样可以有效的提高计算机的利用效率。但是现有的虚拟化技术随着互联网的发展也出现了相应的问题：1) 传统虚拟化资源划分粒度太大；2) 占用资源多；3) 并且迁移耗时耗力。所以需要新的虚拟化技术带来对上述问题的更好的解决办法，Docker 容器技术能够很好的解决这几个问题。

Docker 容器技术的出现，很好的解决了传统虚拟化技术的众多问题，Docker 容器具备如下特点：1) 独立性；2) 细粒度；3) 快速创建和销毁；4) 很多管理工具。但是容器的出现不能简单的用于替代传统虚拟化技术，需要寻找新的应用场景，所以可以通过利用容器技术来支持微服务开发架构，使得服务的维护更加方便快捷。微服务开发架构为开发者提供了可以更好开发应用程序的开发模式，它具备的特点：1) 各个微服务间相互独立；2) 每个微服务都是一个原子化的服务，即不能再划分成更小的服务；3) 并且各个微服务能够快速的组合重构成一个系统。随着 Docker 容器技术和微服务开发架构的完善，相信未来能够为云计算提供更加便捷的服务支持。

虽然 Docker 和微服务的结合能够很好的为现有的云计算提供更加便捷的服务支持，但是当结合 Docker 与微服务是也会产生相应的问题。本文着重考虑将以微服务架构开发的各个微小的服务部署到支持 Docker 容器虚拟化的服务器上将会产生的难点问题：将各个微服务部署到 Docker 平台上后各个微服务之间的通信时延问题，降低各个微服务之间的通信时延能够很大程度上降低服务访问的时间，提高用户体验。与此同时，考虑降低主机的数量，达到节约能源的效果。据此本文结合人工蜂群算法和分布估

计算法(EDABC)用于完成降低通信时延和减少主机数量。

本文的组织结构如下：第二节首先简单介绍一下相关背景知识，以及将问题模型化，最后介绍人工蜂群算法的基本概念。第三节介绍相关研究工作。第四节先介绍分布估计算法以及如何在人工蜂群算法中使用改进的 UMDA 构建一个概率模型。第五节实验部分对给出的实验结果进行分析。最后一节总结本文中的研究工作，以及对未来研究工作的展望。

2. 背景知识

2.1. Docker 与微服务

随着虚拟化技术的不断发展，人们对于虚拟化的技术革新需求越来越高，2013 年以来由 dotCloud 这个 Paas 提供商在 Github 上开源的容器引擎 Docker [1]进入虚拟行业的视野，相较于传统的 VM 虚拟化技术，Docker 以占用资源少，并且支持简单快速的构建、分发和运行获得包括亚马逊、Google、微软和阿里云等云计算服务提供商的青睐。

Docker (容器引擎)，简而言之是个用于将各种各样应用程序打包(pack)、分发(ship)和运行(run)在一个轻量容器中的发布在 github [2]社区的开源项目。Docker 容器是一个与硬件和平台都无关的虚拟化技术，那么可以在任何平台运行，你的个人电脑、工作站、远程服务器等。Docker 使用 C/S 架构模式，用户或者管理者使用远程 API 管理 Docker 容器，可以实现创建镜像、启动容器和分发容器等。

如图 1 右边所示，Docker 具备的特点：1) 独立性：每个容器既是一个独立完整的执行环境，不依赖外部环境；2) 细粒度：每台物理机可以同时运行几百甚至上千的容器数量，每个容器的粒度很小；3) 快速创建和销毁：每个容器可以在以秒为单位的时间内创建以及销毁；4) 很多管理工具：具有数量众多的容器编排和管理工具，能够实现服务的组合调度。

如图 1 所示，相较于传统的虚拟化技术(VMs)，Docker 容器虚拟化具有非常明显的优势。首先从占用的存储大小上来看 VMs 占用非常大，这也导致很难用于调度，相反 Docker 容器的占用极小管理方便。其次运行 VMs 需要首先消耗大量的 CPU 和内存资源，留给应用程序的资源就会减少很多，相反 Docker 容器运行消耗的资源相较于 VMs 可以忽略不计。另外从可移植性角度来看 VMs 庞大不好移植，而 Docker 容器可以做到随时随地，以秒级别的速度运行起来。总的来说，相较于传统的虚拟化技术，Docker 容器具有非常明显的优势，随着时间的推移，Docker 容器技术将会越来越完善。Docker 容器技术的出现很好的解决了虚拟技术的相关问题，同时也需要为 Docker 容器虚拟化技术融入现有的云计算平台找到更多的应用场景，随着 Docker 容器技术的出现各个微服务可以使用更小粒度的容器进行部署，而且 Docker 容器的管理方便，能够很好的管理各个微服务。

微服务的出现是由于服务的疯狂增长使得现有的单体式软件开发架构急需改变，随着 Docker 容器技术的日趋完善，让微服务架构受到广大软件开发者和云服务提供商的重点关注。2016 年四月，Uber [3]退出基于模块的 monolithic 整体架构，转变为灵活的微服务架构。将服务拆分的微服务架构，采用相对独立的服务进行管理，相互之间使用同一的接口来进行交流，体现优势包括：1) 可控的复杂度。2) 各个微服务独立部署。3) 应用的扩展性增强。4) 各个微服务可以根据各自特点选择相应的技术。5) 增强了容错能力。

如图 2 所示，左边既是单体式架构，那么看到应用开发既是在同一个框架中开发。右边就是微服务开发架构，微服务具备的特点：1) 相互间独立：相较于单体式架构，每个服务既是一个独立的服务，也是一个完整的自治系统，能够独立提供服务。2) 原子化：与单体式架构不同的是每个微服务一定是一个原子化的服务，即服务不能再进行划分成比现在更小的服务。可以通过多个微服务组合成一个某个系统来达到某些目的。3) 组合和重构：微服务的特点既是能够快速的组合和重构，相互间组成一个系统，在

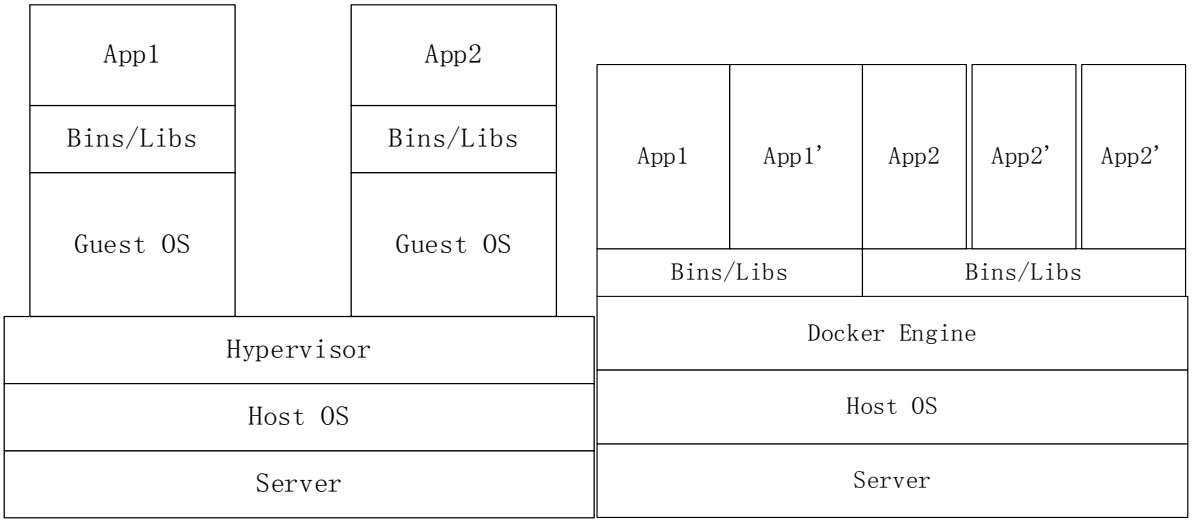

Figure 1. VMs vs. docker container
图 1. VMs vs. Docker 容器

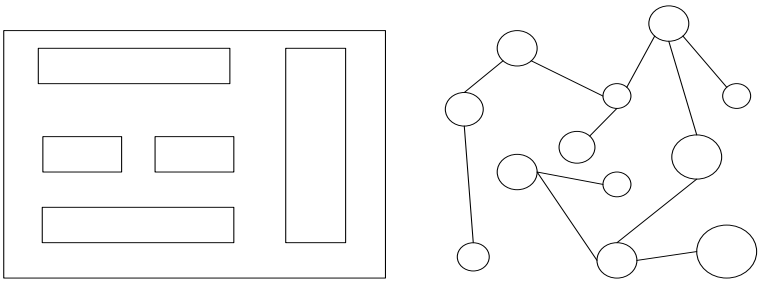

Figure 2. Monolithic architecture vs. micro-service architecture
图 2. 单体式架构 vs. 微服务架构

这个系统中的各个微服务是松耦合的关系，每个微服务机构简单，这样扩展性会更强。

2.2. 多目标优化问题定义

如[4]所描述的多目标问题如下所示：

$$\min y = \left[f_1(x), f_2(x), \cdots, f_k(x) \right] \tag{1}$$

$$\text{subject to } g(x) = \left[g_1(x), g_2(x), \cdots, g_m(x) \right] \le 0 \tag{2}$$

其中，n 维的向量 $x = (x_1, x_2, \cdots, x_n) \in X$ 和 k 维向量 $x = (x_1, x_2, \cdots, x_k) \in X$ 表示决策向量，X 表示决策空间，Y 是由目标向量形成的目标空间，$g(x)$ 是约束条件，决定决策向量的可行域。

可行解集 X_f 是满足约束条件的决策向量 x 构成的集，如下：

$$X_f = x \in X \mid g(x) \le 0 \tag{3}$$

那么目标空间对应如下：

$$Y_f = f\left(X_f\right) = Y_{x \in X_f} \left\{ f(x) \right\} \tag{4}$$

Pareto 最优解集和 Pareto 前沿定义：

对于集合 $A \subseteq X_f$，决策向量 $x \in X_f$ 是非劣的，当且仅当：$\exists a \in A : a$ 优于 x。

Pareto 前沿之间的关系是无差别关系，所有 Pareto 前沿的集合称作 Pareto 最优解，对应到二维目标函数的 Pareto 前沿解集对应曲线称作 Pareto 前沿面(Pareto Front)，如下图 3 所示。

2.3. 双目标问题定义

将要定义的问题是在云计算 Docker 平台上部署微服务，这是一个装箱的问题，要被装箱的就是部署在 Docker 容器里的微服务，并且由于微服务数量众多模型化的变量众多，各个变量之间由于通信存在着关联，与此同时变量间存在着一个无向有环图。另外箱子就是代表着的服务器资源，包括：CPU、RAM 等。

首先需要考虑到在服务器上部署服务资源的有效利用是很重要的，需要用更少的主机来完成工作需求，并且不再使用二进制矩阵形式表示容器和主机间的关系而是使用 $y=\{y_1,y_2,\cdots,y_n\}$ 表示容器与主机的配对，n 表示容器即变量的数量。C 和 M 分别表示 RAM 和 CPU 资源，其中 R 表示预先设定的主机数量，k 即使某一台主机，用 $c=\{c_1,c_2,\cdots,c_h\}$ 表示在第 k 台主机上的容器占用 CPU 资源的集合。同时用 $m=\{m_1,m_2,\cdots,m_h\}$ 表示在第 k 台主机上的容器占用的 RAM 资源的集合。那么第一个需要优化的目标：

$$Minimize\ f_1(y)=Num(Unique(y)) \tag{5}$$

约束条件：

$$\begin{aligned}&\sum_{j=1}^{h}c_j\le C_k,\forall k\in R\\&\sum_{j=1}^{h}m_j\le M_k,\forall k\in R\end{aligned} \tag{6}$$

上述约束条件中(6)分别用于保证某一台主机不会产生 CPU 和 RAM 溢出。

其次需要考虑的是部署在服务器上的容器由于各个微服务之间的通信带来的时延问题，同样的使用 $y=\{y_1,y_2,\cdots,y_n\}$ 表示容器与主机的配对，n 表示容器即变量的数量，其中 u 代表了与 v 构成同一个应用的微服务标号，e 代表了时延因子。那么第二个需要优化的目标是：

$$Minimize\ f_2(y)=\sum_{v=1}^{n}\sum_{j=1}^{R}e*eq(y_{v,j},y_{u,j}) \tag{7}$$

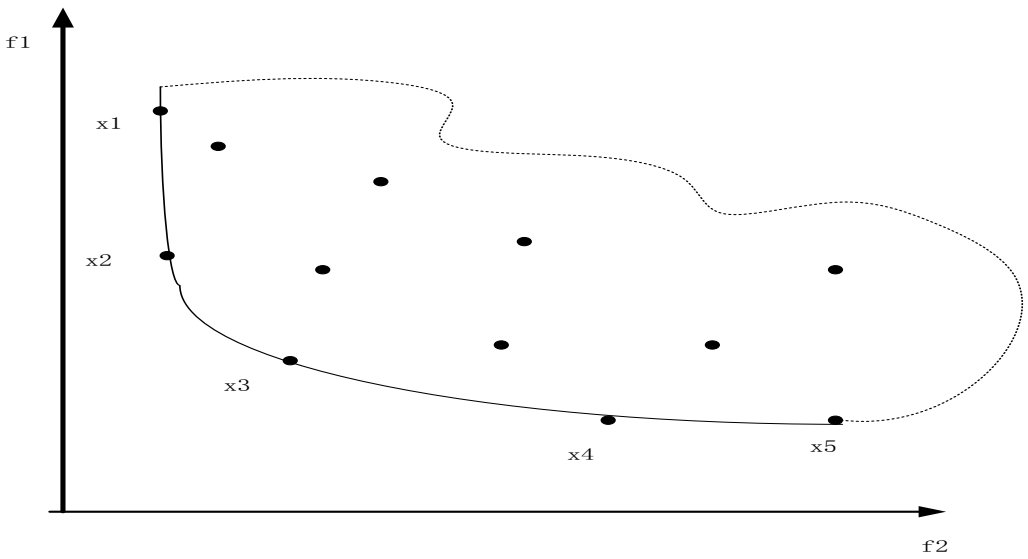

Figure 3. Double target Pareto frontier
图 3. 双目标 Pareto 前沿面

2.4. ABC 算法描述

人工蜂群算法(Artificial Bee Colony Algorithm, ABC)是由土耳其学者 KaraBoga [5]提出的模拟蜜蜂群体寻找优质蜜源的仿生智能计算方法。通过每个个体的分工以及信息共享，一起完成蜜蜂的采蜜工作。同时知道单只蜜蜂的能力有限，不过整个蜂群却总是能够找到优质蜜源。ABC 算法的主要特点是在搜索过程中只需要对问题进行 Pareto 支配原则判断作为进化依据。

人工蜂群算法的搜索模型包括的要素：蜜源、采蜜蜂、观察蜂和侦查蜂。蜜源代表了优化问题的可行解，采蜜蜂存有蜜源的有关信息，观察蜂通过蜂巢的跳舞区分享采蜜蜂的相关信息来选择蜜源，侦查蜂则随机搜索新的蜜源。

多目标人工蜂群算法的具体步骤：

1) 蜂群和参数的初始化，依据 Pareto 支配原则找到所有非劣解并更行外部档案完成初始化；

2) 将种群分为采蜜蜂和观察蜂，采蜜蜂根据外部档案的非劣解进行领域搜索，依据 Pareto 支配原则判断新的久的解的支配关系，保留优质解；观察蜂依据采蜜蜂进行领域搜索；

3) 侦查蜂当循环计数变量 trail 达到限定上限的解抛弃，并随机产生一个新的解；

4) 当前种群中 Pareto 最优解更新到外部档案中，可根据拥挤距离裁剪外部档案；

5) 循环第二步到第四步直至结束，得到结果。

3. 相关研究工作

目前，随着云计算的发展，虚拟化技术的应用越来越广泛。来与此同时，云计算的资源调度问题也日趋严峻。近些年，越来越多的学者开始使用进化计算(Evolutionary Computation, EC)算法用于处理云资源调度问题，这些算法可以在拥有众多变量的 NP-难问题提供全局解。ZHAN 和 LIU [6]等人总结了近几年将进化计算结合到云计算资源调度的研究，其中将云计算中资源调度的问题总结为三个大类：应用层(软件)调度问题、虚拟层(平台)调度问题和基础层调度问题。其中在虚拟层调度问题中可以划分为：负载平衡调度、能源减排调度和成本效益调度三个问题。

从平衡物理机负载来说，Hu [7]等人使用 GA 算法来进行在物理机上部署虚拟机以此达到负载平衡，他们使用不同的染色体编码方案，每个物理机可以托管多个虚拟机，所以他们的 GA 方法使用树型编码方案来指示在每个物理机上创建多少个虚拟机。在运行时，GA 方法重新映射虚拟机和物理机之间的关系，以此实现最大化的负载平衡。该方法的缺点是树型编码仅能支持 15 台虚拟机的调度。

除此 GA 方法以外，Lu [8]等人使用 ACO 方法快速的找到最近的无负载云资源。在他们的研究中，并不是在不同的云资源上调度虚拟机，而是在一些虚拟机超负荷时执行 ACO 来进行处理。ACO 过程就像蚂蚁搜索食物以便找到最佳物理资源一样(如：无负载节点)，并创建新的虚拟机来分担负载。该方法的缺点是在蚂蚁搜索食物过程中对超负荷虚拟机的标记工作的耗时。

另外，从尽可能减少物理资源的角度看，Chen [9]等人使用虚拟机和物理机映射的编码方案。每个基因代表一个虚拟机，基因会根据相应的虚拟机大小进行排序。这样如果染色体无效该方法可以使用贪婪策略将一些虚拟机移动到其他的物理机上，避免了同一台物理机上的虚拟机大小超过物理机大小。该方法的缺点是需要对虚拟机的大小进行排序，而且使用贪婪策略在时间上的消耗太多。

在减少物理机资源上除了 GA 方法以外，Feller [10] [11]等人通过使用 ACO 方法以实现在物理机上调度虚拟机，以此减少过物理机的数量来达到减少资源的效果。他们把资源调度问题建模为多维装箱问题，在蚁群算法的每个步骤中，根据信息素信息，蚂蚁会试图有效的选择下一个虚拟机放置在物理机上，目的为了达到尽可能的利用到每个物理机的资源，使得每个物理机上能够放置更多的虚拟机，以此减少物理机的数量。该算法的缺点是随着虚拟机的增加算法的时间需求呈现指数型增长，无法对更多数量的

虚拟机进行部署配置。

随着 Docker 容器的开源，Docker 公司也随后提供了集群管理工具 swarm [12] [13]，Swarm 提供了三种策略来对容器进行部署：spread、binpack 和 random。首先，spread 策略是 swarm 集群管理的默认策略，swarm 会根据各个节点正在运行的容器数量进行部署容器，在此策略下会选择容器数量少的节点来启动所需的容器，这样就可导致稀疏扩展容器。其次 binpack 策略是根据各个节点的资源使用情况来分配容器，因此此策略会留出空间来给更大的容器运行，可以避免碎片化的产生。另外，random 策略正如字面意思就是随机部署容器到各个节点上。

Google 公司的 Abbhishek 和 Luis [14]等人提出在谷歌的平台上使用 Borg 来进行对整个服务平台的管理。Borg 可以跨越多个集群支持成千上万的机器。他们提出 Borg 系统架构和重要的设计决策，对某些决策进行定量分析。该系统的缺点是过于复杂的语言规范以及对资源的自动化请求工作比较复杂。另外 Google 公司开源在 GitHub 上的一款 Docker 容器管理工具 Kubernetes [15]受到广大开发者的关注，其中也包括各大互联网公司(如亚马逊、阿里云和腾讯云等)的关注。Kubernetes 能够实现自动装箱，根据资源需求放置容器。而且还能够实现水平缩放以及服务发现和负载平衡等功能。Kubernetes 能够监控应用程序的运行状态并对应用程序配置进行更改。当然 kubernetes 缺点是没有想 swarm 那样容易上手，而是具备比较复杂的配置。

Docker 可以在云中运行复合应用程序提供良好的基础，特别是如果是云本地的。然而，Docker 专注于在一个主机上管理容器，但是 SaaS 提供商需要一个由多个主机组成的容器管理解决方案。Rene 和 Florian [16]等人对解决方案进行详细的分类，并将其映射到案例的研究中，并且确定了差距和集成的需求。他们利用他们自己的集成组件和工具增强功能弥补了这种差距，从而产生了目前最完整的管理套件。但是存在的缺点是需要映射到对应的案例中，需要针对相应的案例进行相应的研究。

Marcelo 和 David [17]等人对于使用 Docker 容器的微服务架构进行了性能的研究。在他们的研究中，主要工作目的就是比较在应用程序运行在单体式架构和微服务架构这两种架构的性能。比较在这两种架构的模型中的 CPU 和网络运行基准的性能，因此为系统设计者提供分析指导。但是该方法的缺点是智能提供相应的分析指导，还未能直接为系统设计者带来相应的方案。

通过回顾容器技术和微服务架构中的服务发现的挑战，Joe 和 Walter [18]等人通过介绍 Serfnode，一个完全分散的开源解决方案，基于 Serf 项目的服务发现问题。Serfnode 是一个非入侵性的 Docker 镜像，构成一个或者多个任意的容器。可以实现监视和自我修复。研究者将现有的解决方案用于服务发现问题，作为 Serfnode 的可扩展性的一个例子，展示了使用 Git 在 Docker 容器之间构建文件系统同步解决方案。

Xinjie 和 xili [19]等人设计一个 AODC 的资源分配框架，以最小化数据中心中的应用程序部署成本，并支持自动扩展，利用 Docker 容器的功能对 AODC 资源分配问题进行建模，并为具有多样化和动态应用程序以及大量物理资源的数据中心提出可扩展算法。Johanson 和 Dullo [20]等人在海洋观察系统收集气候相关事件序列数据，为了交互探索和分析这样的高维数据集，他们开发出 OceanTEA 软件。使用微服务开发架构并利用 Docker 平台实现通过机器学习方法的时间模式将交互式数据可视化。OceanTEA 的微服务架构确保充桌面计算机无缝扩展到云计算基础架构的可维护实施。

遗传算法(GAs)是解决硬优化问题的强大技术。但是，可扩展性问题可能会阻止它们应用于现实问题。利用云中的并行 GA 可能是一种经济的方法，可以获得时间有效的解决方案，利用云的吸引人的功能，如可扩展性、可靠性和容错成本效益。Pasquale 和 Filomena [21]等人提出一种使用全局并行化模型分发 GA 的方法，在 Docker 平台上利用软件容器及其云编排。他们还以 DevOps 方式设计了涵盖每个云 GAs 分布阶段(从资源分配到实际部署和执行)的概念工作流。Pasquale 和 Filomena [22]等人提出在云环境中开发、部署和执行并行遗传算法(Paralle-GA)。以分布式方式从资源分配到实际部署和执行的不同阶段。该

方法的缺点是扩展性不够，需要进行有计划的评估。

近两年在 Docker 容器虚拟化平台上的研究工作，以及人们对于微服务架构的研究，并且看到使用 GA 遗传算法在利用容器和云编排一起进行容器的管理。通过前人的研究本文在新的虚拟化技术 Docker 容器平台上基于资源的降低基础上提出了降低部署在容器中的各个微服务之间的通信距离问题。

4. MOMDA-ABC

4.1. 分布估计算法描述

分布估计算法(Estimation of Distribution Algorithm, EDA) [23]对问题进行优化是通过将统计学习结合进化计算对搜索空间进行采样来产生新的优秀个体。分布估计算法通过构建概率模型可以清晰的表示出变量间的相互关系，帮助解决离散型的实际问题。对 EDAs 的划分，可以分成如下类别：1) 离散型；2) 实数型；3) 排列型；4) 遗传编程型；5) 多目标 EDA。

4.2. UMDA

UMDA (Univariate marginal distribution algorithm) [24] [25]是由德国学者 Muehlenbein 等人提出。在 EDAs 算法中是用于解决变量无关的问题。它通过在优势种群中构建概率模型用于估计联合概率分布并进行采样：

$$p_l(x) = \prod_{i=1}^{n} p_l(x_i) \tag{8}$$

在本论文的工作中，调整概率模型从优势种群中的估计联合概率的方式，用于针对使用人工蜂群算法解决多变量相关问题中建立变量相关的联合概率分布。需要根据变量间的关系构建一个集合 $a = \{x_1, x_2, \cdots, x_i, y_1, y_2, \cdots, y_i\}$，其中变量 x_i 集合是和当前选定的变量具有通信关系的变量集合用于保证能与相关的变量相互靠近以减少通信距离，变量 y_i 集合用于保证需要部署的所有微服务能够相互靠近进而减少部署过程中需要的主机数量。

4.3. MOMDA-ABC 算法描述

通过改进分布估计算法中的 UMDA 的概率模型构建，并结合进人工蜂群算法中构建多变量之间的相关性，从而解决之前提出的在 Docker 平台上部署带有通信的微服务所带来的多变量间的相关性。

1) 初始化

在方法中首先随机生成一个种群即初始种群。正如第二节中所述，改变种群个体的编码方式，种群个体中的变量代表了部署在容器中的微服务与主机的配对关系，即一串 n 个数字表示代替了二进制编码所要使用的矩阵需要 n^2 个二进制数字。不仅使得存储算法迭代过程存储空间的节省，也使的使用改进的 UMDA 和 ABC 算法更加方便清晰，同时需要在初始化之后对外部档案进行更新。

2) 采蜜蜂和观察蜂

采蜜蜂是代表了蜂群中最大的群体，在整个算法中用于保持对有前景的个体的继续探索。在每次迭代过程中，将会随机的选取外部档案中的一个解对当前的领域搜索进行指导。采蜜蜂将会根据改进的 UMDA 构建的概率模型进行领域搜索。同时观察蜂也会集合改进的 UMDA 构建的概率模型进行搜索用于保证迭代过程中的群体的多样性。

3) 侦查蜂和种群控制比例

在蜂群采蜜过程中，侦查蜂是种群中比例最小的群体，它一直处于一种随机的搜寻状态用于保证能在蜂群采蜜中寻找到更多的蜜源，也是用于提高种群的多样性。在种群的比例控制中，采蜜蜂占比 60%，观察蜂 30%，剩余的就是侦查蜂 10%。伪代码如算法 1。

算法 1：MOMDA-ABC

1. **输入：**种群大小 NP
2. **输出：**最终的外部档案 Rpop，其中包括：通讯距离和主机数量
3. 依据种群大小初始化种群 Foods，并设置环境参数；依据 Pareto 支配原则获得初始化后的外部档案
4. while(iter<maxCycle)
5. %%采蜜蜂采蜜过程
6. fori = 1:60%*NP
7. for j = 1:H
8. 构建变量间的关系集合 $a=\{x_1,x_2,\cdots,x_i,y_1,y_2,\cdots,y_i\}$
9. 依据集合 a 构建联合概率分布对采蜜蜂个体进行与外部档案的领域搜索
10. 依据 pareto 支配原则判断是否更新个体
11. end
12. end
13. %%观察蜂采蜜过程
14. fori = 60%*NP+1:90%*NP
15. for j = 1:H
16. 构建变量间的关系集合 $a=\{x_1,x_2,\cdots,x_i,y_1,y_2,\cdots,y_i\}$
17. 依据集合 a 构建联合概率分布对采蜜蜂个体进行与采蜜蜂的领域搜索
18. 依据 pareto 支配原则判断是否更新个体
19. end
20. end
21. %%侦查蜂采蜜过程
22. for90%*NP+1:NP
23. 随机生成个体进行个体判断
24. end
25. 更新外部档案
26. end

5. 实验

5.1. 设计与说明

在本节中将会对算法 MOMDA-ABC 进行模拟实验，通过与一个非常经常使用的算法 FFD 和原生的多目标人工蜂群算法进行比较，以此体现算法 MOMDA-ABC 对 Docker 容器平台的微服务的部署这个新的模型的性能表现。

在本实验中，设想一个在线教育系统，如图 4 所示。将会提供不同层次的在线课程给不同地方的学生。学生可以在系统中选择一系列的服务，包括课程选择服务、笔记功能服务、交流讨论服务等服务。这样在线教育系统可以服务很多的用户，将会生成更多的服务。

实验设置如下：模拟实验平台是 4 核 8GB 内存的 MAC 电脑。集群包括每台主机的资源：24 核和 50 GB 内存。在每台主机上其中的一个部署微服务的容器只能对应一台主机，不存在一个编号的容器部署到多个主机上的情况。其中模拟出多种容器的配置，包括 1 核、2 核、3 核和 4 核，以及 2 GB、4 GB、6 GB 和 8 GB 内存。

在此试验中参数的设置如下：NP = 150 (种群的数量)、Limit = 50 (单个个体未更新循环的极限次数)、maxCycle = 300 (总的循环次数)、以及根据上述的种群比例控制，采蜜蜂、观察蜂和侦查蜂分别占比 60%、30%

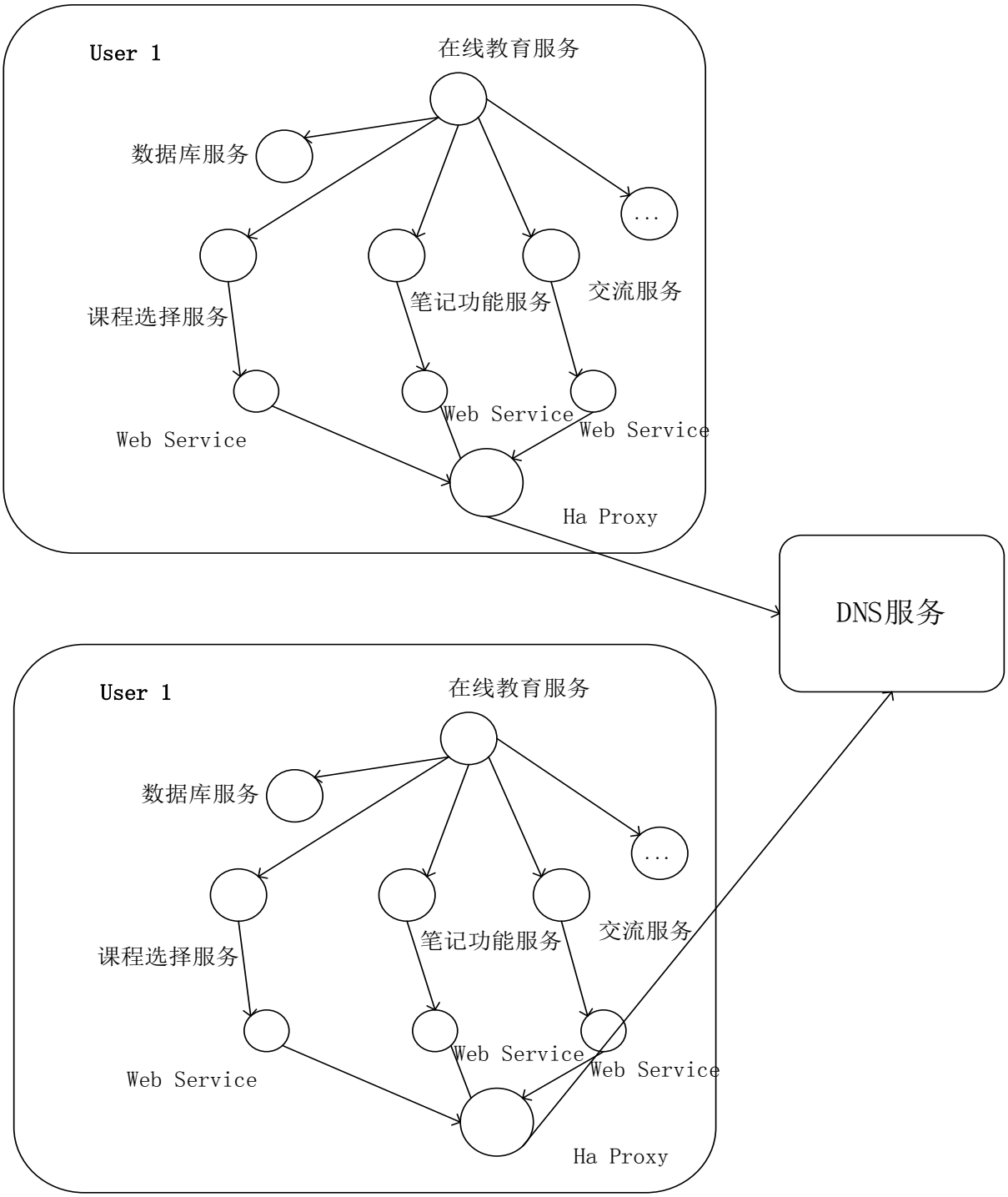

Figure 4. The example of online education system
图 4. 在线教育系统示例

和 10%。通过在 MAC 电脑上运行 Matlab 程序，获得输出的外部档案 Rpop，其中包括通讯距离和主机数量。

5.2. 实验结果与分析

实验结果图 5 展示了使用 MOMDA-ABC 算法对双目标通信距离和主机数量进行优化所得的 Pareto

前沿面，其中 f1 表示通信距离，f2 表示主机数量。此图表示的数据是容器数量 160 个，每个容器的 cpu 和内存大小分别在 1 核、2 核、3 核和 4 核，以及 2 GB、4 GB、6 GB 和 8 GB 内存。

实验结果表 1 展示了 FFD、ABC、MOMDA-ABC 三种不同的方法对不同的容器数量进行优化的结果。实验的结果可以通过通讯距离和主机数量来进行评价。其中通讯距离越短表明通讯成本越低，而主机数量越小则表明相应的运营的维护成本越低，通讯距离和主机数量是通过程序运行的输出获得的。实验结果表明，MOMDA-ABC 方法较现有的方法随着容器数量的增多能够较好的对双目标通讯距离和主机

Table 1. FFD, ABC and MOMDA_ABC algorithms
表 1. FFD、ABC 和 MOMDA_ABC 算法比较

80	FFD	145	10
	ABC	124	11
	MOMDA-ABC	85	11
100	FFD	183	12
	ABC	170	14
	MOMDA-ABC	124	13
120	FFD	218	15
	ABC	195	18
	MOMDA-ABC	161	17
140	FFD	253	18
	ABC	244	22
	MOMDA-ABC	201	20
160	FFD	291	21
	ABC	286	25
	MOMDA-ABC	238	23

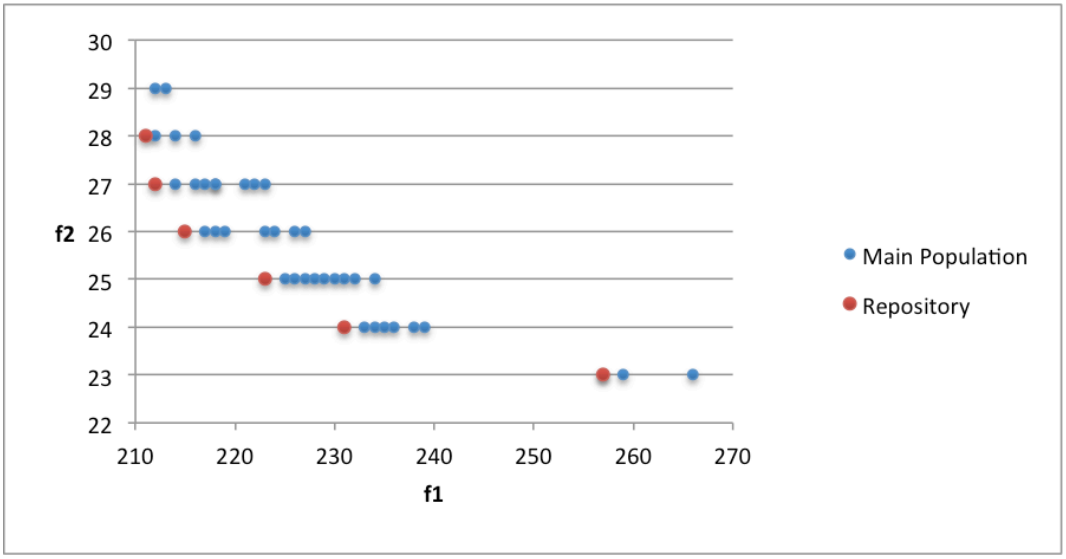

Figure 5. The Pareto frontier of MOMDA-ABC
图 5. MOMDA-ABC 算法所得前沿面

数量都进行很好的优化。

实验结果图 6 展示了使用方法 MOMDA-ABC 在运行时间变化情况。实验结果可以通过时间曲线进行评价。其中时间表明随着容器数量的增长算法运行时间的增长变化。实验结果表明，相较于 Feller [10] 等人使用 ACO 算法运行时间指数型增长情况，使用 MOMDA-ABC 方法能够在增长到 500 个容器数量情况下实现线性增长，表明算法的运行时间具有更高的效率。

实验结果图 7 展示了使用方法 MOMDA-ABC 针对参数 hostParam 的变化满足不同用户群体的需求。实验结果可以通过通讯距离均值、主机数均值和预设主机参数 hostParam 来进行评价。其中通讯距离均值越短表明通讯成本越低，而主机数量均值越小则表明相应的运营的维护成本越低，同时预设主机参数 hostParam 的增长体现对通讯距离和主机数的影响情况。实验结果表明，从图中能够直观的发现当预设主

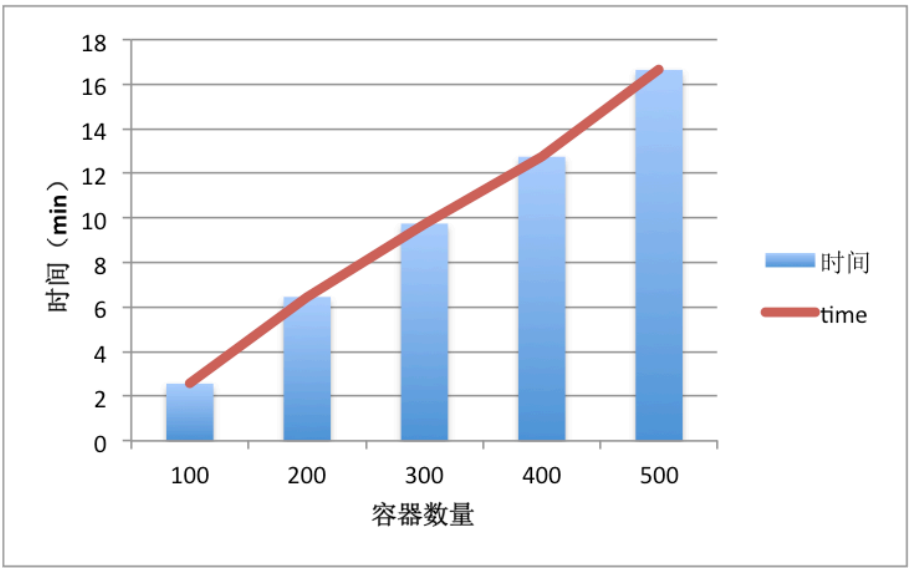

Figure 6. MOMDA-ABC algorithm operation time
图 6. MOMDA-ABC 算法运算时间

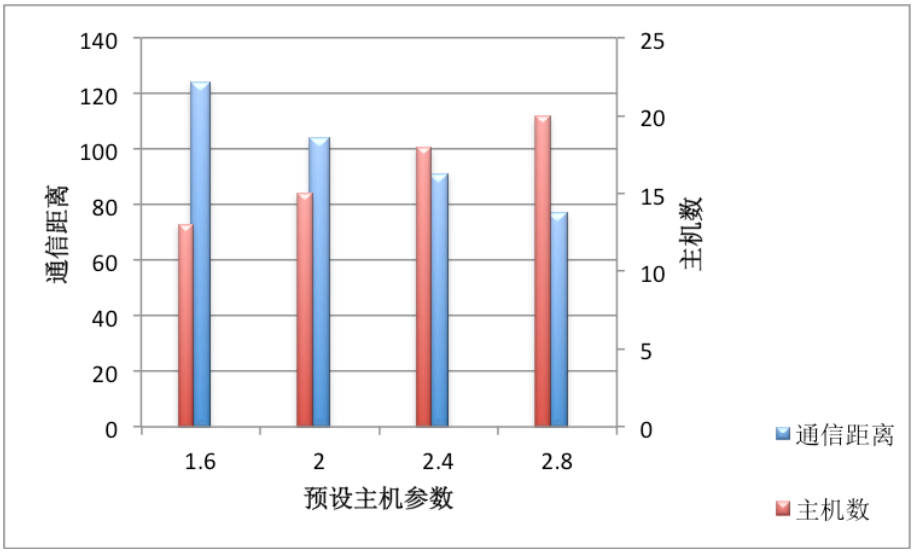

Figure 7. Changes of preset host parameter host Param
图 7. 预设主机参数 host Param 的变化

机参数 hostParam 增长的时候通讯距离均值会相应的降低(如图中蓝色的柱形降低)，而当参数 hostParam 增长则主机数量均值则会相应的增加(如图中红色的柱形增长)。对于不同的用户可以根据自己需求配置不同的主机参数满足相应的需求。

实验图表展示。

6. 总结与展望

本文通过使用 MOMDA-ABC 方法运用于 Docker 和微服务平台，通过实现对容器间的通讯距离和主机数量双目标优化，进而实现对容器间通讯距离的降低和对容器主机数量使用量的降低。实验表明本文中的方法能够有效的优化通讯距离和主机数量，并能使得算法的实验时间得到显著提高。本文的研究能够对 Docker 与微服务的结合起到促进的作用，使得在应用微服务架构开发应用程序部署到 Docker 平台后能够有效的解决通讯和能耗的问题。同时对于云计算的发展起到一定的促进作用，让更多的人能够使用微服务开发架构，并使用云计算提供平台支持。

通过本文的研究，可以发现分布估计算法与启发式算法的结合能够有效的提升算法的时间，相信可以结合其他的不同启发式算法或改进分布估计算法的概率模型应用在更多的问题上，在未来的工作中，考虑结合进来多种分类器[26] [27] [28]对 Docker 容器的分类进行研究工作，与此同时可以将 MOMDA-ABC 算法结合逻辑[29] [30] [31] [32]进行机器人的研究，另外在机器人情感分析[33] [34]和机器视觉[35]中结合 MOMDA-ABC 算法进行深层次研究，最后在领域学习中[36]结合此算法也将会是一个研究方向。

基金项目

该研究项目受到国家自然科学基金 61673328/F030603 资助。

参考文献 (References)

[1] https://www.docker.com

[2] https://github.com/docker/docker

[3] https://eng.uber.com/building-tincup/

[4] Deb, K. (2014) Multi-Objective Optimization. In: Burke, E.K. and Kendall, G., Eds., *Search Methodologies*, Springer, Berlin, 403-449. https://doi.org/10.1007/978-1-4614-6940-7_15

[5] Karaboga, D. (2005) An Idea Based on Honey Bee Swarm for Numerical Optimization. Computers Engineering Depart ment, Engineering Faculty, Erciyes University.

[6] Zhan, Z.-H., Liu, X.-F., *et al.* (2015) Cloud Computing Resource Scheduling and a Survey of Its Evolutionary Approaches. *ACM Computing Surveys*, **47**, Article No. 63. https://doi.org/10.1145/2788397

[7] Hu, H., Gu, J.H., Sun, G.F. and Zhao, T.H. (2010) A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment. *Proceedings of the* 3*rd International Symposium on Parallel Architectures, Algorithms and Programming*, Dalian, 18-20 December 2010, 89-96.

[8] Lu, X. and Gu, Z.L. (2011) A Load-Adaptive Cloud Resource Scheduling Model Based on Ant Colony Algorithm. *Proceedings of the IEEE International Conference on Cloud Computing and Intelligence Systems*, Beijing, 15-17 September 2011, 296-300.

[9] Chen, S., Wu, J. and Lu, Z.H. (2012) A Cloud Computing Resource Scheduling Policy Based on Genetic Algorithm with Multiple Fitness. *Proceedings of the IEEE* 12*th International Conference on Computer and Information Technology*, Chengdu, 27-29 October 2012, 177-184.

[10] Feller, E., Rilling, L. and Morin, C. (2011) Energy-Aware Ant Colony Based Workload Placement in Clouds. *Proceedings of the* 12*th IEEE/ACM International Conference on Grid Computing*, Lyon, 21-23 September 2011, 26-33. https://doi.org/10.1109/grid.2011.13

[11] Feller, E. and Morin, C. (2012) Autonomous and Energy-Aware Management of Large-Scale Cloud Infrastructures. P*roceedings of the IEEE* 26*th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*,

Shanghai, 21-25 May 2012, 2542-2545.

[12] https://docs.docker.com/engine/swarm/

[13] https://github.com/docker/swarm

[14] Verma, A., Pedrosa, L., Korupolu, M., *et al.* (2015) Large-Scale Cluster Management at Google with Borg. *Proceedings of the Tenth European Conference on Computer Systems*, ACM, 18.

[15] https://kubernetes.io/

[16] Peinl, R., Holzschuher, F. and Pfitzer, F. (2016) Docker Cluster Management for the Cloud—Survey Results and Own Solution. *Journal of Grid Computing*, **14**, 265-282. https://doi.org/10.1007/s10723-016-9366-y

[17] Amaral, M., Polo, J., Carrera, D., *et al.* (2015) Performance Evaluation of Microservices Architectures Using Containers. *IEEE* 14*th International Symposium on Network Computing and Applications* (*NCA*), Cambridge, MA, 28-30 September 2015, 27-34. https://doi.org/10.1109/nca.2015.49

[18] Stubbs, J., Moreira, W. and Dooley, R. (2015) Distributed Systems of Microservices Using Docker and Serfnode. 7*th International Workshop on Science Gateways* (*IWSG*), Budapest, 3-5 June 2015, 34-39. https://doi.org/10.1109/iwsg.2015.16

[19] Guan, X., Wan, X., Choi, B.Y., *et al.* (2017) Application Oriented Dynamic Resource Allocation for Data Centers Using Docker Containers. *IEEE Communications Letters*, **21**, 504-507.

[20] Johanson, A., Flögel, S., Dullo, C., *et al.* (2016) OceanTEA: Exploring Ocean-Derived Climate Data Using Microservices.

[21] Salza, P. and Ferrucci, F. (2016) An Approach for Parallel Genetic Algorithms in the Cloud Using Software Containers. arXivpreprint arXiv:1606.06961

[22] Salza, P., Ferrucci, F. and Sarro, F. (2016) Develop, Deploy and Execute Parallel Genetic Algorithms in the Cloud. *Proceedings of the* 2016 *on Genetic and Evolutionary Computation Conference Companion*, Denver, Colorado, 20-24 July 2016, 121-122.

[23] Larranaga, P. and Lozano, J.A. (2002) Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Press, Boston. https://doi.org/10.1007/978-1-4615-1539-5

[24] Muhlenbein, H. and Paass, G. (1996) From Recombination of Genes to the Estimation of Distributions I. Binary Parameters. In: Voigt, H.M., Ebeling, W., Rechenberg, I. and Schwefel, H.P., Eds., *Parallel Problem Solving from Nature —PPSN IV*. *PPSN* 1996. *Lecture Notes in Computer Science*, Vol. 1141, Springer, Berlin, Heidelberg, 178-187. https://doi.org/10.1007/3-540-61723-X_982

[25] Muhlenbein, H. (1997) The Equation for Response to Selection and Its Use for Prediction. *Evolutionary Computation*, **5**, 303-346. https://doi.org/10.1162/evco.1997.5.3.303

[26] Jiang, M., Ding, Y., Goertzel, B., *et al.* (2014) Improving Machine Vision via Incorporating Expectation-Maximization into Deep Spatio-Temporal Learning. 2014 *International Joint Conference on Neural Networks* (*IJCNN*), Beijing, 6-11 July 2014, 1804-1811.

[27] Chao, F., Sun, Y., Wang, Z., *et al.* (2014) A Reduced Classifier Ensemble Approach to Human Gesture Classification for Robotic Chinese Handwriting. 2014 *IEEE International Conference on Fuzzy Systems* (*FUZZ-IEEE*), Beijing, 6-11 July 2014, 1720-1727.

[28] Yao, G., Chao, F., Zeng, H., *et al.* (2014) Integrate Classifier Diversity Evaluation to Feature Selection Based Classifier Ensemble Reduction. 2014 14*th UK Workshop on Computational Intelligence* (*UKCI*), Bradford, 8-10 September 2014, 1-7.

[29] Lee, G., Tolia, N., Ranganathan, P. and Katz, R.H. (2011) Topology-Aware Resource Allocation for Data-Intensive Workloads. *ACM SIGCOMM Computer Communication Review*, **41**, 120-124. https://doi.org/10.1145/1925861.1925881

[30] Zhang, X., Jiang, M., Zhou, C., *et al.* (2012) Graded BDI Models for Agent Architectures Based on Łukasiewicz Logic and Propositional Dynamic Logic. *International Conference on Web Information Systems and Mining*, Chengdu, 26-28 October 2012, 439-450.

[31] Chao, F., Hu, L., Shi, M. and Jiang, M. (2011) Robotic 3D Reaching through a Development-Driven Double Neural Network Architecture. In: Wang, Y. and Li, T., Eds., *Knowledge Engineering and Management. Advances in Intelligent and Soft Computing*, Vol. 123, Springer, Berlin, Heidelberg, 179-184.

[32] Wu, Y., Jiang, M., Huang, Z., Chao, F. and Zhou, C. (2015) An NP-Complete Fragment of Fibring Logic. *Annals of Mathematics and Artificial Intelligence*, **75**, 391-417. https://doi.org/10.1007/s10472-015-9468-4

[33] Jiang, M., Yu, Y., Chao, F., *et al.* (2013) A Connectionist Model for 2-Dimensional Modal Logic. 2013 *IEEE Symposium on Computational Intelligence for Human-Like Intelligence* (*CIHLI*), Singapore, 16-19 April 2013, 54-59.

[34] Cai, Z., Goertzel, B., Zhou, C., *et al.* (2013) OpenPsi: A Novel Computational Affective Model and Its Application in Video Games. *Engineering Applications of Artificial Intelligence*, **26**, 1-12.

[35] Chao, F., Wang, Z., Shang, C., *et al.* (2014) A Developmental Approach to Robotic Pointing via Human-Robot Interaction. *Information Sciences*, **283**, 288-303.

[36] Jiang, M., Huang, W., Huang, Z. and Yen, G.G. (2017) Integration of Global and Local Metrics for Domain Adaptation Learning via Dimensionality Reduction. *IEEE Transactions on Cybernetics*, **47**, 38-51. https://doi.org/10.1109/TCYB.2015.2502483