

基于改进匈牙利算法的云仿真资源调度

岳 璐, 傅妍芳, 钟联炯, 高武奇

(西安工业大学 计算机科学与工程学院, 陕西 西安 710032)

摘 要: 针对传统仿真系统平台的资源分配存在资源闲置、任务挤压和负载均衡等优化问题, 利用云计算技术的优势研究并提出了模块化的云仿真平台框架, 通过对云仿真资源调度策略研究, 提出了一种改进的匈牙利算法. 该算法克服了传统匈牙利算法只适用于一对一资源调度的不足, 实现了多对一的仿真任务与云仿真资源分配方案, 能尽量避免资源调度负载失衡. 通过扩展云计算仿真平台 CloudSim 实现了模拟算法仿真. 结果表明, 该调度策略能有效的减小云环境下计算机的负载, 提高了资源的利用率.

关键词: 云仿真; 匈牙利算法; 资源调度; CloudSim

中图分类号: TN47

文献标识码: A

文章编号: 1000-7180(2012)12-0022-05

Cloud Simulation Resource Schedule Based on an Improved Hungary Algorithm

YUE Lu, FU Yan-fang, ZHONG Lian-jiong, GAO Wu-qi

(School of Computer Science and Engineering, Xi'an Technological University, Xi'an 710072, China)

Abstract: Aiming at the optimization problem of idling resource, squeezing task and loading balance in the traditional resource schedule of the cloud simulation, this paper presents cloud simulation resource scheduling strategy based on an improved Hungary algorithm. It overcomes the traditional Hungarian algorithm applies only to the lack of one-to-one resource scheduling and achieves the distribution scheme between many-to-one simulation tasks and cloud simulation resources, avoiding the resource loading balance at best. By extending the Cloud Computing platform CloudSim to test the simulation, the results show that this method can reduce the loading of computers in the cloud computing, and the resource utilization is improved too.

Key words: cloud simulation; Hungary algorithm; resource scheduling; CloudSim

1 引言

随着计算机技术的发展, 云计算^[1]成为信息领域研究的热点. 可以网上实现计算和数据的存储, 将计算任务分布在大量计算机构成的资源池上, 使用户能够按需获取计算力、存储空间和信息服务. 2009 年李伯虎院士把云计算应用到仿真领域, 提出了云仿真的概念, 即使仿真资源是分布、复杂、动态和异构的, 仿真用户也可以通过云仿真平台对公有的资源进行灵活的使用. 为此, 应用云仿真平台的关键技术对仿真任务进行高效的调度成为云仿真^[2]所要解

决的重要问题.

当前国内外文献中解决资源调度^[3]的主要方法可以概括为以下几点: 先来先服务算法、基于粒子群优化的算法、基于 Petri 网的任务调度、基于遗传算法的任务调度和基于神经网络的任务调度等, 这些方法都存在问题描述过于复杂、正确性不高、收敛性能以及搜索全局最优解的能力低等^[4].

为了解决以往调度问题, 本文采用了匈牙利算法, 该算法可以求出问题的精确解^[5], 运行任务时间最少, 但是在使用时有约束条件. 因此, 针对云环境中任务调度的特点, 提出了一种基于改进匈牙利算

收稿日期: 2012-04-10; 修回日期: 2012-05-16

基金项目: “十二五”预研基金(40405010301); 陕西省教育厅基金(11JK1064); 陕西省科技厅攻关项目(2011K06-14)

法的云仿真资源调度算法,在任务和资源不对等的情况下突破了一对一的指派限制。

2 基于云仿真平台的体系结构

《建模与仿真系统的现状与未来》一文提出:在开发建模与仿真路线图时的另一个重要方面是,不要将大规模框架项目的开发置于所有其他建模与仿真项目的关键路径上。大的框架也是大的职责,为了降低风险,必须尽可能将大框架设计成模块化的,以易于修改和维护。所以,本平台需要具有的功能包括下面几点:

- (1)云仿真平台的总体设计;
- (2)能够监测分布式仿真系统的各个资源状态;
- (3)可靠的任务性能预测;
- (4)仿真任务的静态调度问题,以负载均衡为原则,实现仿真任务初始自动分配;
- (5)仿真任务的动态调度问题,并实现系统负载均衡,提高系统的容错性;
- (6)调度的最终目的实现任务的迁移;

总体上,云仿真平台^[6]主要具有三大功能,包括:注册、调度和监控功能。注册功能指将仿真用户和任务信息进行管理,在云仿真系统中,各类用户首先通过云仿真平台门户提交自己的任务需求,然后系统就会按照用户的需求查找资源,进而对这些资源进行动态的管理,完成整个仿真的任务。监控功能指能够监测分布式仿真系统的各个资源状态,支持仿真任务需求的表达。其中任务需求主要涉及任务计算消耗、任务之间依赖关系及交互的通信消耗、任务期望完成时间、任务执行对软硬件环境的要求等。一般需要提供图形和文本两种接口供用户表达任务需求;调度功能指能够对任务进行静态调度和动态调度,静态调度包括了根据任务的需求选择合适的网格节点、按某种调度策略将任务分配到网格节点上执行等。动态调度功能指能够支持仿真运行时的动态任务调度。动态调度主要考虑负载均衡和任务迁移;一般的建模仿真活动是一个标准化的过程,如开展效能仿真是按照HLA的FEDEP来开发联邦,都遵循一定的步骤。因而任务和资源的管理应该在软件上支持这种重用性,使各功能都具有模块化。

本平台是以HLA为基础根据分布式管理应用的特点,对HLA的框架进行了改进,使其适合仿真任务的执行,扩充对基于HLA/RTI的分布式仿真应用和并行仿真应用的支持。本平台可以实现系统/联邦中各类资源安全地按需共享与重用,实现网上

资源多用户按需协同互操作、共享与重用,实现系统/联邦动态优化调度运行。

平台框架如图1所示。HERTI是基于HLA Evolved的联邦运行支撑平台,由开源联邦运行支撑平台CERTI和WSPRC(Web Services Provider RTI Component 提供Web服务的RTI组件)构成。Federate为云仿真任务联邦成员,RTI需要提供一个WSPRC,使得一个或多个基于WSDL的联邦成员,能通过互联网连接。云仿真可以利用基于Internet的海量仿真资源,有三个仿真模块:仿真注册中心模块,进行仿真任务或对象的注册;云仿真监控模块,需要对仿真任务的性能和运行情况进行监控;云仿真调度模块,在仿真任务出现故障时进行动态迁移,也需要对仿真资源进行动态管理。

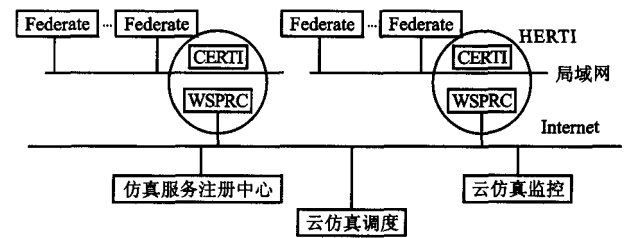


图1 基于HLA Evolved的云仿真平台

3 改进的匈牙利算法在云仿真资源调度的应用

3.1 匈牙利算法基本思想

匈牙利算法^[7]是由匈牙利数学家Edmonds于1965年提出,它是可以用于求解多种形式的指派问题的最简单、最有效的方法。在实际应用中,若每个工人在完成各项工作时效率不同,要求在工作分配的同时考虑总效率最高的问题,这就是最优匹配问题,又称指派问题。

建立数学模型^[8]如下:设用 $C_{ij} > 0 (i, j = 1, 2, \dots, n)$ 表示指派第 i 人去完成第 j 项任务时所用的时间,定义决策变量:

$$x_{ij} = \begin{cases} 1 & \text{当指派第 } i \text{ 人去完成第 } j \text{ 任务时} \\ 0 & \text{不指派第 } i \text{ 人去完成第 } j \text{ 任务时} \end{cases} \quad (1)$$

则问题可转化为0-1线性规划问题:

$$\min z = \sum_{i=1}^n \sum_{j=1}^n C_{ij} x_{ij} \quad (2)$$

$$\text{s. t. } \begin{cases} \sum_{i=1}^n x_{ij} = 1, & i = 1, 2, 3, \dots, n \\ \sum_{j=1}^n x_{ij} = 1, & j = 1, 2, 3, \dots, n \\ x_{ij} = 1 \text{ 或 } 0, & i, j = 1, 2, 3, \dots, n \end{cases} \quad (3)$$

定理 1 如果从分配问题效率矩阵 $W(w_{ij})$ 的每一行元素中分别减去(或加上)一个常数 u_i (称为该行的位势), 从每一列分别减去(或加上)一个常数 v_j (称为该列的位势), 得到一个新效率矩阵 $B=(b_{ij})$, 若其中 $b_{ij}=w_{ij}-u_i-v_j$, 则二者的最优解等价。

定理 2 若矩阵 W 的元素可分成 0 和非 0 两部分, 则覆盖 0 元素的最少直线数等于位于不同行不同列的 0 元素的最大个数。

则匈牙利算法的基本思想是: 由定理 1 的方法不断变换效率矩阵 W , 使其产生尽可能多的 0 元素, 并且始终保持所有元素非负, 直到能从变换后的矩阵中找出 n 个位于不同行、不同列的 0 元为止, 这些 0 元对应的 $x_{ij}=1$, 其余 $x_{ij}=0$ 的方案为最优。“匈牙利算法”的实质就是寻找 n 个处在不同行不同列的零元素。

3.2 云仿真平台下的资源调度

3.2.1 改进的匈牙利算法数学模型

在云仿真中存在资源数目和仿真任务不等的问题, 对于这种不平衡的情况, 在某一时间范围内, 是不变的, 因此可以用改进匈牙利算法来进行求解, 将不平衡的指派问题转化成平衡指派问题。

现有这样的云仿真问题, 共有 m 台计算机, 有 n 项仿真任务, 考虑到每台机器完成其中的一项任务, 同时完成每项任务所需时间、机器的负载、网络带宽的不同, 建立数学模型如下:

为了便于分析问题, 将云仿真计算机上的运行时间 $T_k, k=1, 2, \dots, m$, 负载 $U_k, k=1, 2, \dots, m$, 网络延迟 $D_k, k=1, 2, \dots, m$. 设用 $W_{ij} > 0 (i, j=1, 2, \dots, n)$ 表示指派第 i 台计算机去完成第 j 项云仿真任务时综合权重, 其中 W_{ij} 越小表示云仿真任务在云仿真资源上运行的性能越好。

定义决策变量:

$$x_{ij} = \begin{cases} 1 & \text{当指派第 } i \text{ 个云仿真资源去} \\ & \text{完成第 } j \text{ 项云仿真任务时} \\ 0 & \text{当不指派第 } i \text{ 个云仿真资源去} \\ & \text{完成第 } j \text{ 项云仿真任务时} \end{cases} \quad (4)$$

则问题可转化为 0-1 线性规划问题:

$$\begin{aligned} \min z &= \sum_{i=1}^n \sum_{j=1}^n (T_{ij} + U_{ij} + D_{ij}) x_{ij} \\ &= \sum_{i=1}^n \sum_{j=1}^n W_{ij} x_{ij} \end{aligned} \quad (5)$$

$$\text{s. t. } \begin{cases} \sum_{i=1}^n x_{ij} = 1, & i=1, 2, 3, \dots, n \\ \sum_{j=1}^n x_{ij} = 1, & j=1, 2, 3, \dots, n \\ x_{ij} = 1 \text{ 或 } 0, & i, j=1, 2, 3, \dots, n \end{cases} \quad (6)$$

3.2.2 算法步骤及流程图

根据原匈牙利算法的不足, 结合具体的云仿真环境, 提出改进的匈牙利算法, 它的实现步骤^[9]如下:

第 0 步: 初始化效率矩阵。

第 1 步: 对系数矩阵进行变换, 使每行各列都出现 0 元素. 从系数矩阵每行(列)元素中减去该行(列)的最小元素。

第 2 步: 对云仿真资源和云任务数目进行比较, 如果相等进行步骤 3; 否则, 进行比较, 判断资源多还是任务多. 如果资源数目多, 添加和资源数目相等的虚拟任务(效率值为 0); 如果任务多, 先将效率矩阵分解为两个矩阵: 一个是以资源为维数的矩阵, 另一个是将剩余矩阵补充为以资源为维数的矩阵, 方法是添加和资源数目相等的虚拟任务(效率值为 0)。

第 3 步: 进行试指派, 以寻求最优解. 具体步骤如下:

①从只有一个 0 元素的行(列)开始, 给这个 0 元素作标记, 记作 0^* , 然后划去 0^* 所在列(行)的其他 0 元素, 记作 Φ ; ②给只有一个 0 元素列(行)的 0 元素作标记, 记作 0^* , 然后划去 0^* 所在行(列)的元素, 记作 Φ ; ③反复进行以上两步, 直到所有 0 元素被标记和划掉为止; ④若仍有未标记的 0 元素, 且同行(列)的 0 元素至少有两个, 则可用不同方案去试探。

第 4 步: 作最少的直线覆盖所有 0 元素, 以确定该系数矩阵中能找到最多的独立元素。

第 5 步: 为变换后的系数矩阵增加 0 元素. 在没有被直线覆盖的部分中找出最小元素, 然后在打 x 行各元素中都减去这个最小元素, 而在没打 x 列的各元素都加上这个最小元素, 以保证原来 0 元素不变. 若得到 n 个独立的 0 元素, 则已得到最优解, 否则回到第 3 步重复进行。

改进的匈牙利算法的流程图如图 2 所示。

4 实例验证

由于在资源调度的实际问题中, 经常会遇到资源少任务多的情况, 但经过上述转换后最终归结于资源和任务平衡情况, 现假定有 3 台云仿真计算机 P1, P2, P3, 有 5 个云仿真任务 Cloudlet1, Cloudlet2, Cloudlet3, Cloudlet4, Cloudlet5, 他们对于云仿真任务完成时间、云仿真计算机的负载和网络延迟的综合权重如表 1 所示。

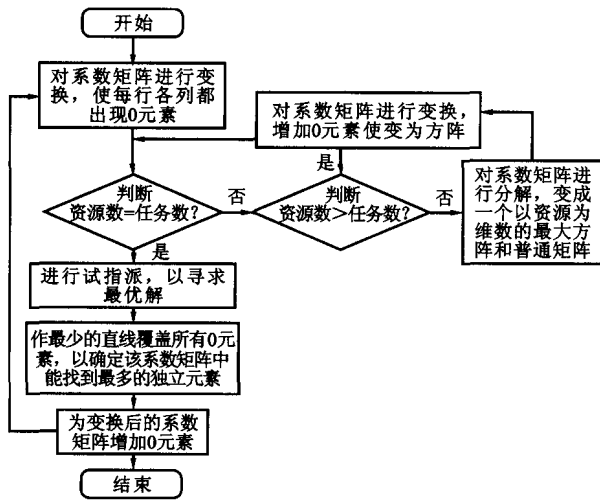


图2 改进的匈牙利算法的流程图
表1 计算机对云任务处理的综合权重

资源	Cloudlet1	Cloudlet2	Cloudlet3	Cloudlet4	Cloudlet5
p1	0.8	0.6	0.9	0.3	0.6
p2	0.8	1.0	0.9	0.5	0.7
p3	0.9	0.8	0.7	0.4	0.5

第一步,将表1所示的综合权重转换成效率矩阵,并对其进行行列约减有矩阵一,然后可以分解成两个矩阵:一个是 3×3 的A,另一个是 3×2 的矩阵B,即:

$$\begin{bmatrix} 0.2 & 0 & 0.3 & 0 & 0.2 \\ 0 & 0.2 & 0.2 & 0 & 0.1 \\ 0.2 & 0.1 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.2 & 0 & 0.3 \\ 0 & 0.2 & 0.2 \\ 0.2 & 0.1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0.2 \\ 0 & 0.1 \\ 0 & 0 \end{bmatrix}$$

矩阵一 A B

第二步,分别对矩阵A和B作最少的直线覆盖所有0元素,以确定该系数矩阵中能找到最多的独立元素.

$$\begin{bmatrix} 0.2 & 0 & 0.3 \\ 0 & 0.2 & 0.2 \\ 0.2 & 0.1 & 0 \end{bmatrix} \xrightarrow{\text{进行指派}} \begin{bmatrix} 0.2 & 0^* & 0.3 \\ 0^* & 0.2 & 0.2 \\ 0.2 & 0.1 & 0^* \end{bmatrix}$$

矩阵A 矩阵A'

$$\begin{bmatrix} 0 & 0.2 \\ 0 & 0.1 \\ 0 & 0 \end{bmatrix} \xrightarrow{\text{增添0元素}} \begin{bmatrix} 0 & 0.2 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \xrightarrow{\text{进行指派}} \begin{bmatrix} 0^* & 0.2 & 0^- \\ 0^- & 0.1 & 0^* \\ 0^- & 0^* & 0^- \end{bmatrix}$$

矩阵B 矩阵B'

第三步,修改效率矩阵,若已经构成了所有 0^* 元位于不同的行、不同列,已符合最优性条件,运算结束.

由矩阵A'和B'得到矩阵 $A' + B'$,将独立的0元素进行指派,用 0^* 做标志.

$$\begin{bmatrix} 0.2 & 0^* & 0.3 & 0^* & 0.2 & 0^- \\ 0^* & 0.2 & 0.2 & 0^- & 0.1 & 0^* \\ 0.2 & 0.1 & 0^* & 0^- & 0^* & 0^- \end{bmatrix}$$

矩阵 $A' + B'$

因为最后一列是虚拟任务,所以可以去掉最后一列,因此计算机对云任务处理的最终分配情况为:

$$\begin{aligned} p_1 &\rightarrow \text{Cloudlet2, Cloudlet4;} \\ p_2 &\rightarrow \text{Cloudlet1;} \\ p_3 &\rightarrow \text{Cloudlet3, Cloudlet5} \end{aligned}$$

在5项任务完成时考虑综合权重值为 $0.6 + 0.3 + 0.8 + 0.7 + 0.5 = 2.9$,而其他的调度算法得出来的结果均不小于2.9,考虑到权重指标越小越好,故此时得到的是最优分配方案.

5 仿真实验与分析

为了验证基于改进的匈牙利调度算法在云仿真资源调度策略的可行性,本文实验扩展了云仿真平台 CloudSim,编程工具为 Myeclipse6.0. DataCenterBroker 实现资源发现和信息交互,是模拟调度的核心.使用 Ant 工具重编译重写 DataCenterBroker、Cloudlet 等类,对上述算法进行模拟仿真.

为验证本算法的优越性,在本次实验中,有3台云仿真计算机,每次都有5个云仿真任务,现在有3种调度策略:先来先服务(FCFS),即如果云仿真计算机空闲,先到的云仿真任务先获得服务的机会;匈牙利算法,先给每个云仿真的资源最优分配云仿真任务,其余任务按照随机分配的方法;改进的匈牙利算法,即将所有的任务一次性分配给云仿真的资源,所有的任务同时进行.

我们分别采用了 CloudSim 现有的 FCFS 调度算法、匈牙利算法、改进的匈牙利算法进行任务调度的对比.各任务在云仿真计算机上的负载率如图3所示.

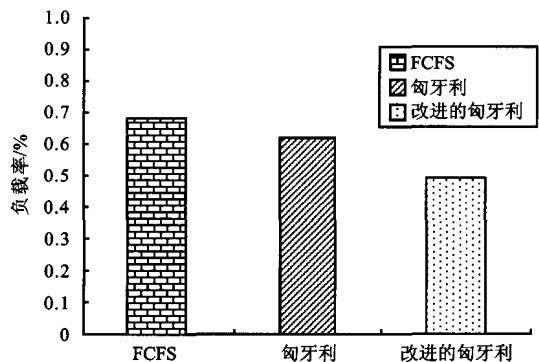


图3 3种不同的调度算法对计算机的负载率对比

在相同的实验条件下,FCFS算法对计算机的负载最大,其次是匈牙利算法,改进的匈牙利算法随着云任务的数量不断增加,负载率明显小于其他两种算法,效率更高.

6 结束语

通过研究在云仿真环境下的合理分配资源问题,提出基于改进的匈牙利算法的云仿真调度方法,同时综合考虑了在云仿真环境下每项任务运行时间、机器的负载、网络延迟的综合因素的影响,解决了传统匈牙利法只适合一对一的指派问题,为资源的一对多的分配提出了一种新的思路,实现了云仿真资源的合理、有效分配,提高了系统的效率。

参考文献:

- [1] 周相兵,马洪江,杨兴江,等.一种基于云计算的语义 Web 服务组合模型研究[J].微电子学与计算机,2009,26(8):206-210.
- [2] 张雅彬,李伯虎,柴旭东,等.基于虚拟化技术的云仿真资源迁移技术研究[J].系统仿真学报,2011,23(6):1268-1272.
- [3] 张千,梁鸿,李振,等.基于改进蚂蚁算法的网格资源管理的研究[J].微电子学与计算机,2009,26(9):71-74.
- [4] 刘万军,张孟华,郭文越,等.基于 MPSO 算法的云计算资源调度策略[J].计算机工程,2011,37(11):43-48.
- [5] 常庭懋,韩中庚.用“匈牙利算法”求解一类最优化问题[J].信息工程大学学报,2004,5(1):60-62.
- [6] 李伯虎,柴旭东,侯宝存,等.一种基于云计算理念的网络化建模与仿真平台——“云仿真平台”[J].系统仿真学报,2009,21(17):5292-5299.
- [7] 徐玖平,胡知能,王绥等.运筹学[M].2版.北京:科学出版社,2004.
- [8] 吴刚,陈振宇,庞妮,等.匈牙利算法实现雷达干扰资源分配[J].舰船电子工程,2011,199(1):87-94.
- [9] 殷红.基于模糊匈牙利算法的炮兵火力单位分配问题[J].2009,31(3):26-28.
- [10] Foster I, Zhao Yong, Raicu I. Cloud computing and grid computing 360-degree compared [C]//Proc. of Grid Computing Environments Workshop. Austin, USA: IEEE Press, 2008:1-10.

作者简介:



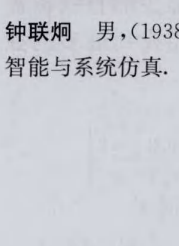
岳璐女,(1988—),硕士研究生.研究方向系统仿真与云计算技术.



傅妍芳女,(1978—),博士研究生,副教授.研究方向系统建模、分布交互仿真和云计算技术.



高武奇男,(1975—),博士研究生,讲师.研究方向为数据挖掘、分布交互仿真和 SOA 技术.



钟联炯男,(1938—),博士生导师,教授.研究方向为人工智能与系统仿真.