

云计算下基于启发式混合算法的工作流调度研究

徐文忠¹, 彭志平², 左敬龙²

(1 江苏科技大学 计算机科学与工程学院, 江苏 镇江 212003;

2 广东石油化工学院 计算机与电子信息学院, 广东 茂名 525000)

摘 要: 针对当前云环境中资源利用率不高, 节点负载不均衡的问题, 提出了一种基于遗传算法的启发式混合算法用于工作流调度。该算法首先基于图的拓扑结构将任务按优先级排序, 然后融合 Best-Fit 方法创建最优初始种群, 最终结合交叉和变异操作来获得最佳解决方案。通过在 CloudSim 环境下的仿真分析和比较, 该算法在任务完成时间、负载均衡率以及加速比方面都优于其他算法。

关键词: 异构分布式计算系统; 云计算; 工作流调度; 启发式; 遗传算法

中图分类号: TP391

文献标识码: A

文章编号: 1000-7180(2015)04-0129-05

Research on Workflow Scheduling in Cloud Computing Based on Heuristic Hybrid Algorithm

XU Wen-zhong¹, PENG Zhi-ping², ZUO Jing-long²

(1 School of Computer Science and Engineering, Jiangsu University of Science and Technology,
Zhenjiang 212003, China; 2 School of Computer and Electronic Information, Guangdong University
of Petrochemical Technology, Maoming 525000, China)

Abstract: In view of the problem that resources utilization rate is not high and node load are imbalance in current cloud computing environment, this paper presents a heuristic hybrid method based on genetic algorithm for workflow scheduling. At first, the tasks are sorted by priority in this algorithm. Then, it merges Best-Fit method to make an optimal initial population, and apply crossover and mutation operations to achieve the best solution finally. This algorithm in the task completion time and load balancing rate and speedup is better than other algorithms.

Key words: heterogeneous distributed computing systems; cloud computing; workflow scheduling; heuristic; genetic algorithm

1 引言

云计算服务提供商整合大规模的网络服务器组成大型虚拟资源池, 基础设施层是硬件(服务器、存储和网络)和相关软件(操作系统虚拟化技术、文件系统)的交付, 并且以灵活和有效的方式向远程用户提供了大量的计算服务。在由数据中心、物理资源等组成的云计算设备中, 资源是以虚拟机部署的形式提供给用户, 以满足客户的请求。

由于一个好的调度方法会显著提高分布式系统的性能, 但没有直接的方法找到最佳解决方案, 所以调度决策必须依赖于启发式方法, 尽可能好地找到一个解决方案。运行在异构分布式计算系统中的软件称为应用程序, 每个应用程序是一个由相互连通的任务组成的工作流, 这些任务被调度运行在系统中的不同处理器上。

对于工作流调度^[1]这个问题已经提出了许多解决方法, 其中一些人使用了启发式方法和进化方法,

收稿日期: 2014-07-28; 修回日期: 2014-09-16

基金项目: 国家自然科学基金资助项目(61272382); 广东省自然科学基金资助项目(S2012010009963); 广东省高等学校科技创新项目(2013KJJCX0132)

这些方法往往侧重于有限的参数数量和主要目标(如所有任务的完成时间),如 Min-min 算法、Max-min 算法等.此外,一些经典的调度算法往往依靠基于列表的方法,如异构最早完成时间或关键路径,这经常被用于一些简单的模型系统中,并且不能准确地反映真正的并行系统.一个动态列表调度(DLS)方法也被提出用于云计算系统,它不考虑有向无环图(Directed Acyclic Graph, DAG)工作流的任务优先级,此外相关文献中也有一些有用的评估工作流调度的算法^[2].

在本文中,我们提出了一种基于遗传算法的后发式混合算法(HHGA),以便找到一个合适的调度解决方案,用于像云环境这样具有复杂计算应用程序的异构分布式计算系统.考虑到平衡资源的负载,提出的 HHGA 算法试图通过优化初始种群,在算法开始时减少遗传操作的迭代次数,该算法侧重降低应用程序的完成时间和错误率,并且使系统达到负载均衡^[3-6].

2 问题定义

在云计算环境中,数据中心系统用于收集和保存资源和任务的静态和动态信息,一些关键的静态信息是:物理内存存储空间,虚拟内存存储空间,磁盘存储空间等,一些动态信息如节点自身的平均负载,运行任务的数量以及 CPU 利用率等,这些信息定期被捕获,并且通过通信组件发送给主节点的数据接受者.这些数据经常被更新,并且以实时的形式提供响应作为服务,显然,云系统中的这种异质性是有效的响应请求.

我们可以使用一些静态和动态的信息用于调度,由于在云系统中的每个时间会收到许多任务,所以工作量和其他动态信息会选择一个好的候选资源用于任务调度.

分布式系统中的每个应用程序是一个包含一组任务的工作流,其中任务通过优先约束互相连接,每个任务都将被执行并给出了一个输出数据集,然后这个数据集被发送到由工作流结构所定义的下一个任务.当工作流的大小增加时,处理时间可能变得很长,由于它们的异质性,云平台的主机将具有不同的性质和计算能力.

某些应用程序是计算密集型,而有些是通信密集型,通信计算比(CCR)是一个度量,能够表明一个任务图是计算密集型、通信密集型还是介于两者之间,CCR 系数等于平均通信成本除以目标系统的平

均计算成本.本文所使用的符号如表 1 所示.

表 1 符号定义

符号	定义
V	节点集合,代表应用程序图中的任务
E	边集合或者应用程序图中任务之间的依赖关系
v_i	应用程序图中的一个任务
$\omega(e_{i,j})$	图中任务 v_i 和 v_j 之间边的度量
p_j^{MIPS}	每分钟处理的资源指令数量
$T_{\text{pred}}(v_i)$	任务 v_i 的前驱任务的集合
$T_{\text{succ}}(v_i)$	任务 v_i 的后继任务的集合
$T_{\text{ready}}(v_i, p_j)$	任务 v_i 的所有前驱任务被执行的时间
$T_{\text{avail}}(p_j)$	处理器 p_j 用于执行任务 V_i 的时间
$T_{\text{ST}}(v_i, p_j)$	任务 v_i 的开始时间
$T_{\text{FT}}(v_i, p_j)$	任务 v_i 的结束时间
$T_{\text{priority}}(v_i)$	任务 v_i 的优先级
$CC(v_i, v_j)$	任务 v_i 和 v_j 数据传输的成本
$T_c(v_i, v_j)$	从任务 v_i 到 v_j 输出数据的大小
$C_{k,l}$	资源 p_k 和 p_l 之间链路的带宽
$d(v_i)$	关键路径中任务 v_i 的深度
p_i^f	单位时间任务失败的概率
p_i^{fr}	资源的失败因素
p_i^t	在当前调度中,资源 p_i 完成工作流中所有分配任务的时间

一般来说,一个具有优先约束任务的应用程序图可以由一个有向无环图(DAG)表示,即 $G = (V, E)$,其中 V 是代表任务的节点集合, E 是边集合或者任务之间的依赖关系.在工作流图中,每个任务直到其前驱任务完成之后才能调度,因此每个任务的开始时间为

$$T_{\text{ST}}(v_i, p_j) = \max\{T_{\text{avail}}(p_j), T_{\text{ready}}(v_i, p_j)\} \quad (1)$$

式中, $T_{\text{ready}}(v_i, p_j)$ 可以按如下公式计算:

$$T_{\text{ready}}(v_i, p_j) = \max\{T_{\text{FT}}(v_k) + \omega(e_{k,i})\}, \quad v_k \in T_{\text{pred}}(v_i) \quad (2)$$

每个任务的完成时间为

$$T_{\text{FT}}(v_i, p_j) = T_{\text{ST}}(v_i, p_j) + (\omega(v_i)/p_j^{\text{MIPS}}) \quad (3)$$

式中, $\omega(v_i)$ 是任务 v_i 的长度,在工作流图中, $T_{\text{pred}} = \varphi$ 的任务是开始任务, $T_{\text{succ}} = \varphi$ 的任务是退出任务.

此外,在我们的研究中,资源被不同的链接完全连接,当任务分配到资源时,两个任务之间数据传输的成本可以按以下公式计算:

$$CC(v_i, v_j) = T_c(v_i, v_j)/C_{k,l} \quad (4)$$

在调度过程中,云系统有一些属性需要考虑,方法中使用的重要约束有:

- 请求的数量总是多于资源的数量,每个资源一次能够处理多个请求.
- 请求的特性总是可变的和不确定的,如到达时间、执行时间等.
- 云环境是异构资源的集合.
- 资源具有动态的硬件和软件特性,如节点上的平均工作负载、CPU 利用率、故障率等.

如上面所提到的,在一些方法中,遗传算法用于在 NP 问题(如工作流调度)中寻找一个合适的解决方案,通常情况下,正常的遗传算法步骤是这样的:

- (1)考虑任务图的拓扑结构,创建初始种群.
- (2)通过适应度函数评估解决方案.
- (3)为下一代选择一些解决方案.
- (4)交叉和变异操作.
- (5)如果不满足停止条件,重复步骤(2)~(5).

3 基于启发式混合算法的工作流调度

本文在遗传算法的基础上提出了一种启发式混合算法,并且考虑了云系统的特征,具体算法流程见算法 1.

算法 1 HHGA 算法

输入:可用资源的集合和一个应用程序未映射的任务;
输出:一个优化的调度解决方案.

1. 在应用程序图中找到每个任务的深度;
2. 每个任务根据(5)设置优先级;
3. 更新资源属性;
4. 创建初始种群;
5. 对于每条染色体,每个任务根据执行时间找到最适合的资源;
6. 选择资源列表中的下一个资源;
7. 如果索引计数器指向上一个资源,选择资源列表中的第一个资源;
8. 使用(8)评估所有染色体;
9. 如果不满足停止条件,则重复以下循环,否则转步骤 10;
- (1)随机选择基因交叉操作;
- (2)变异操作;
- (3)选择最好的染色体作为精英;
10. 保存最佳解决方案;
11. 由于获得了最佳解决方案,因此在候选资源上调度所有映射任务.

3.1 编码

在遗传算法中,每一个解决方案被编码为一条染色体,每条染色体有 N 个基因,即染色体长度.在工作流调度中,每一个调度以染色体的形式出现,包含应用程序的任务和相关的候选资源.首先,图中的任务基于它们对其他任务的影响按优先级进行排序,任务应该从一组可用资源映射到合适的资源.

在算法 1 中,根据数据中心的信息,为了产生一个染色体,每个任务映射到可用资源的虚拟列表中的一个选定的资源,虚拟列表在某些操作(如初始化种群)中将被更新.

3.2 初始化种群

假设一组多个可能的解决方案被称为种群,初始种群在标准遗传算法中随机生成.制作一个好的和以快速的方式响应以目标为导向的初始种群是我们所关注的,为了建立初始种群,当任务按优先级排序之后,它们将被放置在染色体中基因的第一行,并且为每个任务从虚拟资源列表选择一个具有最小运行时间的合适资源.以这种方式,对于每个基因的第一条染色体,该算法从虚拟列表的第一个地方选择最适合的资源,而对于第二条染色体,则从虚拟列表的第二个地方找到最好的资源.

该方法可确保所有资源将被选中用于创建种群,因此,所有可能的解决方案几乎能够平衡资源的负载.当每个可用资源被选择作为候选资源时,根据当前资源对指定任务的处理能力,虚拟列表将被更新.

3.3 任务优先级

因为复杂应用程序图上的任务不能很容易分割,并且每个任务的长度和后继任务与其他任务不同,所以基于图的拓扑选择任务是一个重要的问题.任务的完成时间影响应用程序的完成时间,因此根据以下公式来计算任务的优先级:

$$T_{\text{priority}}(v_i) = \omega(v_i) + \sum_{d(v_j)=a}^{\beta} (d(v_j) * \omega(e_{i,j})),$$

$$v_j \in T_{\text{pred}}(v_i) \quad (5)$$

$$\alpha = \beta - \text{floor} \quad (6)$$

3.4 交叉操作

此处随机选择两个亲本和它们的一些基因,然后通过改变选定基因的部分资源来创建两个其他的解决方案.例如,在一些随机选定的基因中,如第二、第四和最后一个基因,候选资源在两个随机选定的染色体中被每个以太网网络改变.

3.5 变异操作

随机选择一条染色体和它的一个基因,并且从虚拟资源列表选择一个资源,如果该资源的失败率高于上一个候选资源,那么所选择的资源将被替换为选定的基因.资源的失败因素可以如下计算:

$$p_i^{\text{fr}} = p_i^f / p_i^{\text{MIPS}} \quad (7)$$

由于变异操作的原因,遗传算法不会停留在局部最小值,而且这种方法在变异中能够快速找到一种好

的解决方案.

3.6 评估和选择解决方案

为了评估一个解决方案的价值,我们通过带有有效参数的适应度函数来对其进行评估,在遗传算法中,适应度函数应用于所有解决方案并计算其值,然后基于参数放置策略获得一个具有最佳值的解决方案作为最佳解决方案.每个解决方案的适应度值通过式(8)进行计算:

$$Fitness = \sum_{i=1}^m (p_i^t \times p_i^f) \quad (8)$$

目标是:

$$\min imizing (fitness) \quad (9)$$

该算法通过交叉和变异操作尽可能的最小化适应度值,试图找到最佳解决方案,一些最好的染色体将通过精英主义的方法被选择用于下一代中.

4 实验仿真结果及分析

实验采用了云计算仿真平台 CloudSim3.0.3,并对其进行了扩展,重写了 DatacenterBroker、Cloudlet 等类,对本文提出的算法进行了模拟仿真.实验中将本文提出的算法(HHGA)和另外两种算法(LAGA 和 NGA)在任务的完成时间、资源的负载均衡率以及加速比方面进行了对比,实验所用的参数如表 2 和表 3 所示,实验对比的结果如图 1、图 2 和图 3 所示.

表 2 仿真参数

参数	值
任务数量	20~100
任务长度	10~50($\times 10^5$ MI)
资源数量	50
资源处理速度	500~1 000(MIPS)
资源之间的带宽	10~100(mb/s)
资源的失败率	$10^{-4} \sim 10^{-3}$
CCR 值	0.25

表 3 HHGA 算法参数

参数	值
种群大小	50
交叉率	0.5
变异率	0.5
精英主义选择率	0.8
迭代次数	50

图 1 表明:与 LAGA 算法相比,使用 HHGA 算法在任务完成时间上缩短了将近 19.15%,而与

NGA 算法相比,使用 HHGA 算法则缩短了将近 34.39%.

图 2 表明:与 LAGA 算法相比,使用 HHGA 算法在负载均衡率上提高了将近 10.38%,而与 NGA 算法相比,使用 HHGA 算法则提高了将近 11.88%.

图 3 表明:与 LAGA 算法相比,使用 HHGA 算法在加速比上提高了将近 18.22%,而与 NGA 算法相比,使用 HHGA 算法则提高了将近 33.77%.

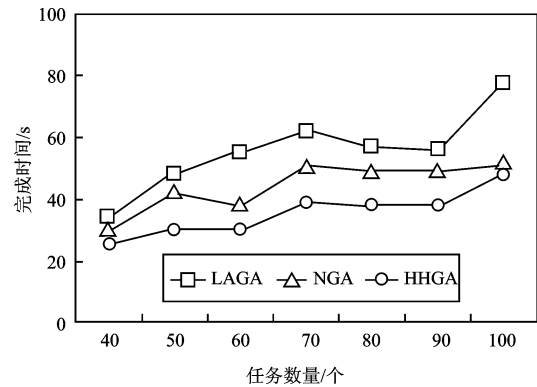


图 1 任务完成时间对比

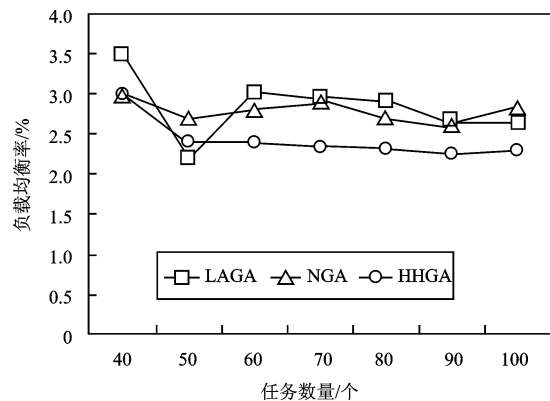


图 2 负载均衡率对比

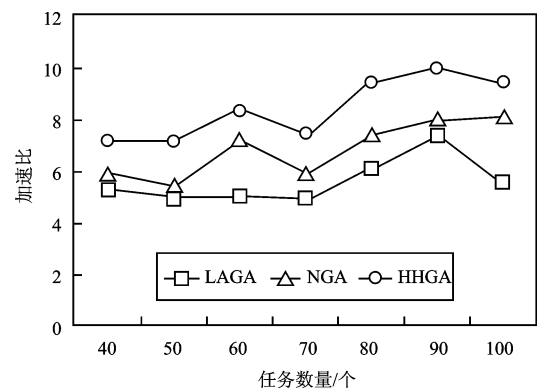


图 3 算法加速比对比

5 结束语

本文所提出的算法(HHGA)通过虚拟资源列表及其更新的属性创建以目标为导向的初始种群,为了创建初始种群,采用两个阶段进行评估.在第一阶段,基于图的拓扑结构,应用程序的所有任务将按优先级排序;在第二阶段,通过 Best-Fit 方法来选择好的候选资源.因此,这种方法在兼顾负载均衡的时候能够快速找到好的解决方案,而且我们在适应度函数中使用一些适当的参数来增加种群评估的质量.

实验结果对比表明,本文提出的算法不仅能够减少任务的完成时间,而且还能降低资源的负载均衡率,极大地提高了资源利用率.在接下来的工作中,我们将着重研究通信密集型应用程序中的工作流调度问题.

参考文献:

- [1] 贵飞,汪芸.云计算下基于容错 QoS 的虚拟机资源分配策略[J].微电子学与计算机,2013,30(3):136-139.
- [2] Casanova H, Desprez F, Suter F. On cluster resource allocation for multiple parallel task graphs[J]. Parallel

Distrib Comput, 2010, 70(2):1193-1203.

- [3] Xinpeng Du, Lizhi Chen, Daiqiang Chen. A simulated annealing algorithm for sparse recovery by l_0 minimization[J]. Neurocomputing, 2014, 131(5):98-104.
- [4] Omara FA, Arafa M M. Genetic algorithms for task scheduling problem[J]. Parallel Distrib Comput, 2010, 70(2):13-22.
- [5] Wang X, Yeo C S, Buyya R, et al. Optimizing the makespan and reliability for workflow applications with reputation and a look-head genetic algorithm[J]. Comput Syst, 2011, 27(1):1124-1134.
- [6] Singh J, Singh H. Efficient tasks scheduling for heterogeneous multiprocessor using genetic algorithm with node duplication [J]. Comput Sci, 2011, 11(3):388-403.

作者简介:

徐文忠 男,(1991-),硕士,研究方向为机器学习和智能信息处理、计算资源管理策略. E-mail:455947770@qq.com.

彭志平 男,(1969-),教授、博士.研究方向为多主体技术、机器学习、自主计算、语义 Web 服务和云计算.

左敬龙 男,(1975-),讲师、硕士.研究方向为人工智能和信息安全.

(上接第 128 页)

- [6] Jussi-Pekka Jasson, Antti Mantyniemi, Juha Kostamovaara. Multiplying delay locked loop (MDLL) in time-to-digital conversion[C]// 2009 Instrumentation and Measurement Technology Conference, I2MTC Singapore: IEEE, 2009:1226-1231.
- [7] Chen-Yu Liu, Wei-Zen Chen. Spur suppression technique for multiplied delay locked loop[C]// 2011 IEEE International Symposium on Radio-Frequency Integration Technology, RFIT. Beijing: IEEE, 2011: 133-136.
- [8] Amr Elshazly, Rajesh Inti, Brian Young, et al. Clock Multiplication Techniques Using Digital Multiplying Delay-Locked Loops[J]. IEEE Journal of Solid-State Circuits, 2013, 40(6):1416-1428.

- [9] Kuo Hsing Cheng, Yu Lung Lo. A fast-lock wide-range delay-locked loop using frequency-range selector for multiphase clock generator[J]. IEEE Transactions on Circuits and Systems, 2007, 54(7):562-565.
- [10] Doo-Chan Lee, Kyu-Young Kim, Young-Jae Min, et al. A jitter and power analysis on DCO[J]. IEEE Transactiona on Circuits and Systems, 2011, 58(9):560-564.

作者简介:

文治平 男,(1962-),研究员.研究方向为微电子技术.

王浩弛 男,(1988-),硕士,助理工程师.研究方向为 FPGA.