遗传-蚁群算法在高性能计算任务调度中的应用

田智慧[1] 张帅永[2] 高 需[3]

[1](郑州大学地球科学与技术学院 河南 郑州 450052)

[2](郑州大学信息工程学院 河南 郑州 450001)

[3](郑州国家超级计算中心 河南 郑州 450001)

摘 要 针对目前高性能计算任务调度策略利用率低、负载不均衡等问题,设计一种基于遗传-蚁群算法的高性能计算任务调度算法(GA-ACO)。GA-ACO 分为两个阶段,第一阶段通过遗传算法缩小空间快速搜索到优秀解,紧接着将其转化为蚁群算法的初始信息素;第二阶段提出一种基于蚁群信息素的全局更新策略对收敛速度做出优化。实验分析表明,与蚁群算法和遗传算法相比,该算法缩短了任务完成时间,降低了节点负载率。

关键词 高性能计算 任务调度 遗传算法 蚁群算法 信息素

中图分类号 TP3 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2024.03.039

APPLICATION OF GENETIC ANT COLONY OPTIMIZATION IN HIGH PERFORMANCE COMPUTING TASK SCHEDULING

Tian Zhihui[1] Zhang Shuaiyong[2] Gao Xu[3]

[1](*School of Earth Science and Technology, Zhengzhou University, Zhengzhou 450052, Henan, China*)

[2](*School of Information Engineering, Zhengzhou University, Zhengzhou 450001, Henan, China*)

[3](*Zhengzhou National Supercomputing Center, Zhengzhou 450001, Henan, China*)

Abstract Aimed at the problems of low utilization rate and unbalanced load of current high performance computing task scheduling strategies, a high-performance computational task scheduling algorithm based on genetic ant colony optimization (GA-ACO) is designed. GA-ACO was divided into two stages. In the first stage, the genetic algorithm was used to narrow the space and quickly find the excellent solution, and then it was transformed into the initial pheromone of ant colony algorithm. In the second stage, a global update strategy based on ant colony pheromone was proposed to optimize the convergence speed. Experimental analysis shows that compared with ant colony algorithm and genetic algorithm, this algorithm shortens the task completion time and reduces the node load rate.

Keywords High performance computing Task scheduling Genetic algorithm Ant colony algorithm Pheromone

0 引 言

随着科技的进步,高性能计算已经在气候模拟、流体力学、分子动力学、生物信息等领域都得到了愈加广泛的应用[1]。针对高性能并发[2]、多个计算系统模型[3]、数据与云存储[4]下的高性能计算,数据处理的速度和用户需求响应的时间不能得到有效提高[5-6],如何对高性能计算系统平台进行资源分配[7]、节能调度[8]和负载均衡化[9]才是增强性能的核心。为了提高高性能计算系统利用率,以及降低系统负载不均衡,工程学者们做出很多调度算法研究。例如,Xu 等[10]提出了一种适用于异构计算系统任务调度的遗传算法(MPQGA),MPQGA 采用多个优先级队列来管理分解的任务,设计了一种新的遗传策略找出最佳可分配方案,并将计算任务分配给相应的异构计算节点,以获得最优的系统调度任务质量。张爱科等[11]提出一种遗传公平性比较均衡的遗传调度算法 FB-GSA。该算法

收稿日期:2021-01-07。国家重点研发计划项目(2018YFB0505004-03);郑州大学 2018 年科研启动基金项目(32210919)。**田智慧**,教授,主研领域:智慧城市与深度学习。**张帅永**,硕士生。**高需**,讲师。

通过解决空闲虚拟资源、占用任务的分配公平性和资源负荷等问题,有效缩短了任务总完成时间,提高分配公平性,最终实现负载均衡。Xu 等[12]提出了一种基于蚂蚁群模型的网络反馈机制选择算法(ANSA),通过向网络选择算法中加入反馈机制,再通过信息素浓度选择正确的路径。Keerthika[13]提出了一种多约束的负载均衡算法。该算法有效降低了调度的成本、最大完工时间、任务失败率,是一种优秀的负载算法。孟令玺等[14]提出了一种改进均衡蚁群的云计算调度方法,仿真结果显示,该方法能够提升系统资源的均衡性。饶健[15]提出了一种基于朴素贝叶斯的云计算预警信息调度法,该算法将节点负荷进行了分类,结合节点上下文环境,根据任务的优先级对调度进行仿真,并得出了较好的负载均衡性,提高了资源的利用率。

然而,遗传算法有更好的搜索空间解能力,并且所需局部参数较多,容易获得局部优秀解;蚁群算法具有更好的搜索精确解能力,但由于初始信息素过少,导致初始搜索解较慢。为了解决遗传与蚁群出现的缺点,提出了一种遗传蚁群高性能计算任务调度算法(GA-ACO),结合遗传算法的搜索空间解能力以及蚁群算法的求解精度优秀,通过模拟实验证明了该算法可以很好地适应于高性能计算任务调度。

1 高性能计算任务调度数学模型

高性能计算系统中资源调度策略的定义如下:设跨度 X 表示一种有效的资源调度策略,即 $T(X)$ 就表示资源 T 在资源调度集群节点 S 上执行任务完成的总时间,为了达到任务资源调度的最终目标最优时间跨度,用 $T(X)$ 表示所要求的最优时间跨度,即求 $T(X)$ 最小化。

设 L 表示负载均衡指标的集合,集合 $L=\{L_1,L_2,\cdots,L_n\}$,则 $L_j(X)$ 就表示在调度策略 X 下的集群节点 j 的负载指标。则 $L_j(X)$ 定义为:

$$L_j(X)=\frac{1}{T(X)}\sum_{i=1}^{m}M_{ij}\times W_{ij} \tag{1}$$

式中:$j\in\{1,2,\cdots,m\}$,$M_{ij}\neq -1$,则可以得到 $0<L_j(X)\leqslant 1$。设 $\rho(X)$ 表示负载指标均方差,则定义 $\rho(X)$ 为:

$$\rho(X)=\sqrt{\frac{1}{n}(L_j(X)-\frac{1}{n}\sum_{j=1}^{n}(L_j(X))^2} \tag{2}$$

根据 $0<L_j(X)\leqslant 1$ 可以得到 $0\leqslant\rho(X)<0.5$,设 $\delta(X)$ 表示负载指标标准差,则定义 $\delta(X)$ 为:

$$\delta(X)=1-2\times\rho(X) \tag{3}$$

由 $0\leqslant\rho(X)<0.5$ 可以得到 $0<\delta(X)\leqslant 1$。

通过数学模型可以得出,找到一种资源调度策略,它可以让 $T(X)$ 达到最小,$\delta(X)$ 达到最大,以此找到优秀解。

2 算法分析

2.1 遗传模型选择

遗传算法(GA)是模拟生物进化的智能算法,它具有较强的搜索空间解能力,因此任务调度分配问题很适合染色体种群中的编码问题。首先使用适应度函数值对任务分配问题进行评估,再利用选择、交叉和变异等操作建立迭代过程,通过不断迭代进化产生新的任务个体,逐步求解出任务调度的优秀解。

2.1.1 染色体编码和解码

针对高性能计算任务调度所存在的问题,需要编码多个染色体,每种染色体都代表特定的任务调度方案。通过多次模拟实验选择间接编码方法。其具体工作流程如下:首先对每个任务占用的节点编码。染色体长度代表任务节点的总数,基因数值表示占用资源的任务节点数量。使用式(4)计算子任务的数量:

$$N_{\text{subtaskNum}}=\sum_{t=1}^{m}taskNum(t) \tag{4}$$

式中:m 表示任务数量;t 表示任务执行顺序;$taskNum(t)$ 表示分配给任务 t 的子任务数量。

例如,假设有三个任务,那么 $m=3$。三个资源表示 $n=3$,三个任务分别被分成三份作业,即 $taskNum(1)=3$、$taskNum(2)=4$、$taskNum(3)=2$,所以 $N_{\text{subtaskNum}}=9$。这表示染色体的总长度是 9。将基因的取值范围定为(1,3)。然后通过间接编码的方式得到一组染色体{2,3,1,2,3,1,1,1}。我们以把第一份作业传输到第二份作业,第二份作业传输到第三份作业的方式,依此类推。然后对这些染色体进行解码,得到任务上各种资源的分布,$w1$:{3,6,8,9},$w2$:{1,4,7},$w3$:{2,5}。

2.1.2 目标函数和适应度函数

使用解码序列等矩阵计算每个资源执行任务的执行时间。因此,完成资源调度任务的总时间为:

$$F(x)=\max_{r=1}^{n}\sum_{i=1}^{W}work(r,i) \tag{5}$$

式中:$work(r,i)$ 表示资源 r 执行该资源上的子任务 i 所花费的时间;W 表示分配给资源的子任务的数量。式(5)定义为目标函数。

我们利用适应度函数评价染色体,其值越大,染色体越优良,任务调度方案求解越好。由于适应度函数的值与目标函数的密切相关,因此适应度函数定义如式(6)所示。

$$f(x)=\frac{1}{F(x)} \tag{6}$$

2.1.3 遗传操作

选择、交叉和变异是遗传算法中的常有操作。通过这些操作,它不断产生新的个体,以找到优秀解。

1) 选择操作。根据适应度函数来确定每个个体进入下一代种群的概率。计算选择概率为:

$$p(i)=\frac{f(i)}{\sum_{j=1}^{S} f(j)} \tag{7}$$

2) 交叉操作。本文采用自适应的交叉方式。首次在每个交叉个体间按较大的交叉概率在个体之间交换一些位元,这样可以避免过早的交叉发生。在算法的后半阶段,由于交叉概率的下降,更加容易直接产生新的优良属性个体,进而使算法的收敛速度变快。

3) 变异操作。本文采用单点突变的方法,以较小的概率去转变群体中的个别位,如"1"到"0","0"到"1"。

在实际操作中,经过多次递归迭代,去除适应度函数值小于平均值的新个体,得到某组的优秀解,作为信息素优化的基础。

2.2 蚁群模型

蚁群算法(ACO)是由意大利学者 Dorigo 等从群体生物进化的角度提出的一种搜索算法。该算法具有较好的鲁棒性、并行性和正反馈特性,因此能够很好地适用于任务调度问题。但是当求解问题较大的时,由于蚁群算法的前期搜索能力过慢,导致信息素不足,从而使得算法收敛速度变慢,所以采取遗传算法前期搜索空间解快的能力实现对任务分配策略的优秀求解。

2.2.1 算法初始化

在染色体群体评价中,我们以连续五代进化率的大小来评价。遗传算法完成后,根据种群个体对该种群的适应程度,依次对该种群个体进行排序,最终将其前 10% 的初始种群作为优化解,并将它转换成蚁群初始信息素。具体的初始化规则如下:

$$T_i^G(0)=\rho S_n \tag{8}$$

式中:ρ 表示设定的一个常数;S_n 表示遗传算法的优化解。通过遗传算法所产生的结果,可以得到信息素的分布情况。资源信息素的初始值为:

$$T_i(0)=r_i+T_i^G(0) \tag{9}$$

式中:r_i 表示资源的处理能力;$T_i^G(0)$ 表示当前遗传算法结束时,转化为信息素的值。

2.2.2 资源选择概率

根据当前资源的信息素浓度值决定下一个资源的概率为:

$$P_k(i,j)=\frac{[T_j(t)]^{\alpha}[\eta_j]^{\beta}}{\sum_{u\in U}[T_u(t)]^{\alpha}[\eta_u]^{\beta}} \tag{10}$$

式中:$T_j(t)$ 表示资源 j 在 t 时刻的信息素值;α、β 分别代表信息素和资源性质的重要程度;η 表示可见性,即:

$$\eta=amP+Bb \tag{11}$$

式中:a、b 表示资源的处理能力和带宽;P、B 表示资源待处理数量和带宽数量。

2.2.3 信息素更新

通过比较蚁群算法的性能,提出了一种基于信息素的全局更新算法,以提高算法的收敛效率。当蚂蚁成功完成资源选择时,信息素随之发生变化。信息素浓度更新如下所示:

$$T_j^{\text{new}}=\rho T_j^{\text{old}}+\Delta T_j \tag{12}$$

式中:ρ 表示信息素挥发系数。

2.2.4 算法终止条件

终止条件与蚂蚁迭代的次数、运行结果有关。在蚁群算法搜索中,如果连续 N 代适应值都没有明显的变化,则该算法就终止。

$$|F(x)_i-F(x)_{i-1}|<\varepsilon \tag{13}$$

式中:ε 是预先设定充分小的数。

2.3 GA-ACO 中的算法描述

根据遗传算法的全局搜索能力,以及通过改进蚁群算法的信息素更新,得到 GA-ACO 在高性能计算任务调度问题求解流程描述如下:

Step1 设置 GA-ACO 的基本参数。

Step2 初始化高性能任务调度解种群数和代数 $G=0$。

Step3 计算当前的适应度值 $f(x)$。

Step4 根据适应度值 $f(x)$,选择最好的个体进入下一代。

Step5 通过概率去实现遗传操作。

Step6 对新种群中的个体适应度值进行计算。

Step7 进化代数加 1,即 $G=G+1$。

Step8 判断是否满足遗传算法的结束条件,是则进入下一步,否则就一直重复 Step4 - Step7。

Step9 在染色体 G 代中挑选适应度最高的个体作为优化解。

Step10 用 Step9 的优化解来初始化蚁群信息素值。

Step11 将 m 只蚂蚁随机地散布于 n 个资源上。

Step12 计算蚂蚁资源分配结果中每个目标函数值,得到优秀解。

Step13 根据全局信息素更新算法对每个资源进行信息素更新。

Step14 判断是否达到终止条件,是则退出,否则,返

回 Step11 继续执行。

高性能计算任务调度的具体求解流程如图 1 所示。

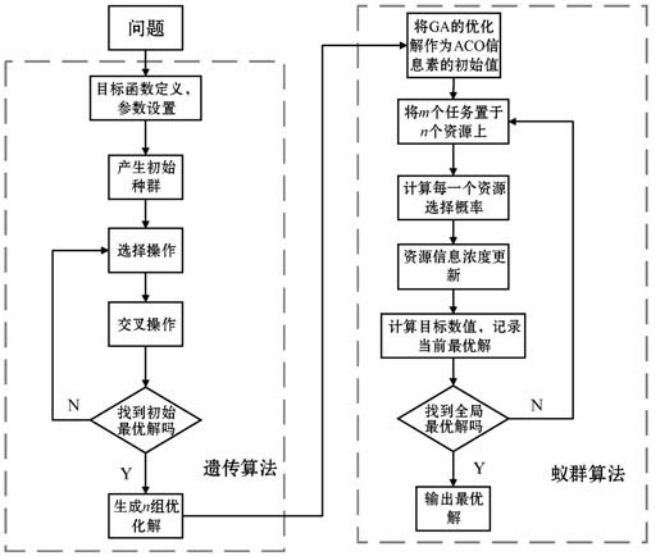

图 1 GA-ACO 的资源调度流程

3 实验与结果分析

为了验证 GA-ACO 的在高性能计算任务调度的优化效果。将实验结果与遗传算法和蚁群算法在相同环境下进行比较。

3.1 参数设置

实验采用河南省超算中心高性能计算平台作为实验的硬件环境，每个节点拥有 2 个 CPU 处理器，128 GB 内存。服务器节点为 20，任务数为 50 ~ 300，分别采用 GA、ACO、GA-ACO 方法进行对比实验。针对遗传算法和蚁群算法每次搜索求解都需要重复实验的优点，在实验过程中考虑了不同参数值的不同场景数。表 1 总结了这些实验中使用的仿真参数。

表 1 算法参数设置

算法	参数	值
GA	人口数量	100
	交叉率	0.6
	突变率	0.1
ACO	蚂蚁数量	100
	α	1
	β	1
	ρ	0.4
	a	0.8
	b	0.6

3.2 实验结果分析

由图 2 可以看出，随着任务数的逐步增加，三种算法的迭代次数也在持续增加，但是 GA-ACO 的迭代次数一直低于另外两种算法，说明 GA-ACO 在任务调度中收敛速度最优，能够更快地搜索到优秀解证。

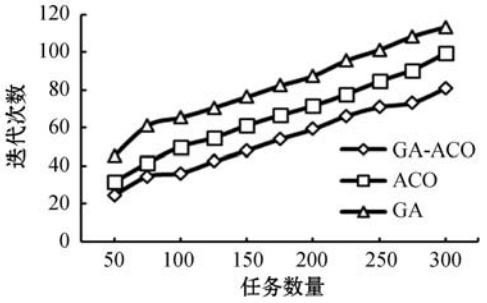

图 2 迭代次数对比

由图 3 可以看出，随着任务数量的增加，三种算法的任务执行时间差异越来越明显，其中 GA-ACO 的任务执行时间随着任务数的增多一直是最快的。原因是 GA-ACO 通过 GA 产生的优化解转化为 ACO 的信息素，避免了 GA 局部搜索和 ACO 缺乏初始信息素的缺点，使搜索空间解的能力增强，提高了收敛速度。

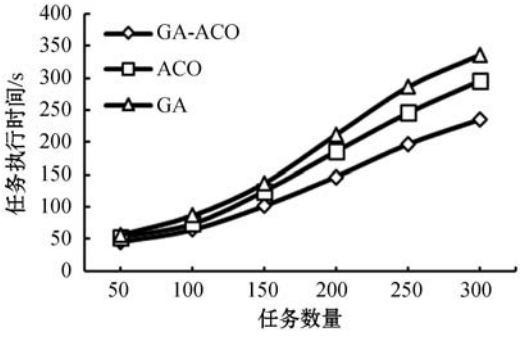

图 3 任务执行时间对比

图 3 是每种算法随着任务数量增多，负载指标的变化，负载指标越小，则任务调度系统的负载均衡性越好。GA-ACO 的负载指标始终低于 ACO、GA，验证了 GA-ACO 对高性能计算任务调度的负载均衡性能的优化。

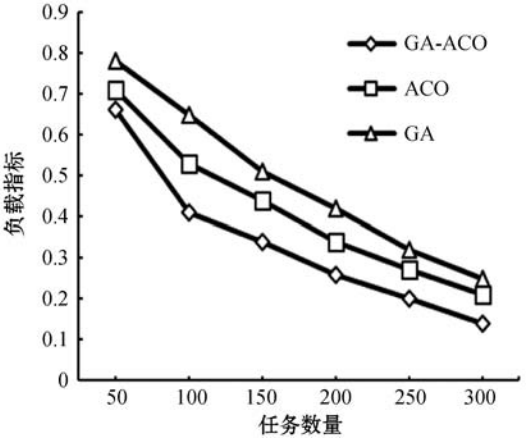

图 4 负载指标的标准差对比

4 结　语

针对目前高性能计算任务调度中资源利用不足以及负载不均衡问题,提出一种遗传-蚁群算法的高性能计算任务调度算法,利用遗传算法的搜索空间解能力强,获得更好的解决方案,然后将它转换成蚁群初始信息素,最终得到更好的全局搜索性和反馈性。实验结果表明,GA-ACO 拥有更快的收敛速度,任务的执行时间更短,负载均衡性更好,从而说明 GA-ACO 有利于在高性能计算中解决任务调度问题。

参 考 文 献

[1] Borghesi A, Bartolini A, Lombardi M, et al. Scheduling-based power capping in high performance computing systems [J]. Sustainable Computing: Informatics and Systems, 2018,19:1 – 13.

[2] Sun H Y, Stolf P, Pierson J M. Spatio-temporal thermal-aware scheduling for homogeneous high-performance computing datacenters[J]. Future Generation Computer Systems, 2017,71:157 – 170.

[3] Thoman P, Dichev K, Heller T, et al. A taxonomy of task-based parallel programming technologies for high-performance computing[J]. The Journal of Supercomputing, 2018, 74(4):1422 – 1434.

[4] Zhou A F. Genetic ant colony algorithm improves resource scheduling in cloud computing[C]//3rd International Conference on Information Science and System, 2020:106 – 109.

[5] Kumar A M, Venkatesan M. Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment[J]. Wireless Personal Communications, 2019,107(4):1835 – 1848.

[6] Jiang J, Zhang J, Zhang L J, et al. Passive location resource scheduling based on an improved genetic algorithm [J]. Sensors, 2018,18(7):2093.

[7] Wang D Y, An X S, Zhou X W, et al. Data cache optimization model based on cyclic genetic ant colony algorithm in edge computing environment [J]. International Journal of Distributed Sensor Networks, 2019,15(8):155 – 164.

[8] Xue H, Kim K T, Youn H Y. Dynamic load balancing of software-defined networking based on genetic-ant colony optimization[J]. Sensors, 2019,19(2):311.

[9] Rousset A, Herrmann B, Lang C, et al. A survey on parallel and distributed multi-agent systems for high performance computing simulations[J]. Computer Science Review, 2016, 22:27 – 46.

[10] Xu Y M, Li K L, Hu J T, et al. A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues[J]. Information Sciences, 2014,270:255 – 287

[11] 张爱科,谢翠兰. 基于公平性和负载均衡的云计算任务调度算法[J]. 计算机应用与软件,2015,32(2):268 – 271.

[12] Xu M H, Zi Y. ACO-based network selection algorithm[J]. Computer Engineering and Applications, 2012,48(5):84 – 88.

[13] Keerthika P, Suresh P. A multi-constrained grid scheduling algorithm with load balancing and fault tolerance [J]. The Scientific World Journal, 2015,2015:349576.

[14] 孟令玺,孟令威. 云计算下的资源负载均衡性调度仿真[J]. 计算机仿真,2018,35(4):386 – 389.

[15] 饶健. 云计算平台网络应急信息优先调度仿真研究[J]. 计算机仿真,2018,35(6):353 – 356.

(上接第 245 页)

[2] Lu Y, Liu L, Zhu J, et al. Architecture, challenges and applications of dynamic reconfigurable computing[J]. Journal of Semiconductors, 2020,41(2):021401.

[3] Bo Z. A level sensitive cluster based partitioning algorithms for reconfigurable systems[J]. Journal of Computer Aided Design & Computer Graphics, 2006,18(5):667 – 673.

[4] 陈乃金. 基于深度优先贪婪搜索的可重构硬件任务划分算法[J]. 计算机应用,2012,32(1):158 – 162.

[5] Miyazawa K, Sakate H, Sekiguchi S I, et al. Real-time hardware implementation of HEVC video encoder for 1080p HD video[C]//Picture Coding Symposium, 2013.

[6] Zhu Y, Jiang L, Shi P, et al. Parallelization of intra prediction algorithm based on array processor[J]. High Technology Letters, 2019,25(1):74 – 80.

[7] 王飞龙,刘新闯,刘鹏,等. HEVC 帧内预测算法加速设计与实现[J]. 计算机应用与软件,2020,37(1):151 – 156.

[8] Bross B, Han W J, Sullivan G J, et al. High efficiency video coding (HEVC) text specification draft 10 (JCTVC-L1003)[C]//JCT-VC Meeting (Joint Collaborative Team of ISO/IEC MPEG & ITU-T VCEG), 2013.

[9] Lakshmi, Aparna P. Efficient architectures for planar and DC modes of intra prediction in HEVC[C]//2020 7th International Conference on Signal Processing and Integrated Networks (SPIN), 2020.

[10] Choudhury R, Rangababu P. Design and implementation of mixed parallel and dataflow architecture for intra-prediction hardware in HEVC decoder[C]//International Symposium on VLSI Design and Test, 2017.

[11] Kurdi A H, Grantner J L, Abdel-Qader I. Hardware accelerator for edge detection[C]//2020 IEEE 24th International Conference on Intelligent Engineering Systems, 2020.

[12] Xiuzhi Y, Min Z, Longzhao S, et al. Research and implementation of fast algorithm for intra prediction mode selection oriented to hardware[J]. Journal of Computer-Aided Design & Computer Graphics, 2019,31(1):158 – 164.