DOI:10.20079/j.issn.1001-893x.220518003
引用格式:沈小龙,马金全,胡泽明,等.面向异构信号处理平台的负载均衡算法[J].电讯技术,2023,63(12):1978-1984.[SHEN X L,MA J Q,HU Z M,et al. A load balancing algorithm for heterogeneous signal processing platform[J]. Telecommunication Engineering,2023,63(12):1978-1984.]

面向异构信号处理平台的负载均衡算法*

沈小龙,马金全,胡泽明,李宇东
(信息工程大学 信息系统工程学院,郑州 450001)

摘 要:针对当前异构信号处理平台中信号处理应用的调度算法优化目标单一且调度结果中处理器负载不均衡的问题,提出了一种基于蚁群优化算法的负载均衡算法。该算法结合蚁群优化算法的快速搜索能力和组合优化能力,以信号处理应用的调度长度和处理器负载均衡为优化目标,对初始信息素矩阵和蚂蚁的遍历顺序进行改进,提出调度长度启发因子和负载均衡启发因子对处理器选择公式进行改进,利用轮盘赌策略确定信号处理应用各子任务分配的处理器,完成信号处理应用的调度。仿真结果表明,该算法得到调度结果在调度长度和负载均衡方面均有改进,可以充分发挥各处理器性能,提高异构信号处理平台的整体效率。

关键词:异构信号处理平台;任务调度;蚁群算法;负载均衡

开放科学(资源服务)标识码(OSID):

中图分类号:TN911 **文献标志码**:A **文章编号**:1001-893X(2023)12-1978-07

A Load Balancing Algorithm for Heterogeneous Signal Processing Platform

SHEN Xiaolong, MA Jinquan, HU Zeming, LI Yudong
(School of Information Systems Engineering, Information Engineering University, Zhengzhou 450001, China)

Abstract: For the problem that the scheduling algorithm of signal processing application in the current heterogeneous signal processing platform has a single optimization goal and the processor load is unbalanced in the scheduling results, a load balancing algorithm based on Ant Colony Optimization(ACO) algorithm is proposed. The algorithm combines the fast search ability and combinatorial optimization ability of ACO algorithm, takes the scheduling length of signal processing application and processor load balance as the optimization goal, improves the initial pheromone matrix and ant traversal order, proposes scheduling length heuristic factor and load balance heuristic factor to improve the processor selection formula, and uses Roulette strategy to determine the processor assigned to each subtask of signal processing application, thus completing the scheduling of signal processing applications. The simulation results show that the scheduling results obtained by the algorithm are improved in terms of scheduling length and load balancing. It can give full play to the performance of each processor and improve the overall efficiency of heterogeneous signal processing platform.

Key words: heterogeneous signal processing platform; task scheduling; ant colony algorithm; load balancing

* 收稿日期:2022-05-18;修回日期:2022-07-22
通信作者:沈小龙

0　引　言

信号处理应用通过有向无环图(Directed Acyclic Graph,DAG)建模为有依赖关系的任务集合。随着科技飞速发展,信号处理应用规模越来越大,功能日趋复杂,信号处理应用通过 DAG 抽象建模后的任务规模急剧上升,各任务功能不同,对处理器需求有差别,传统同构处理平台已经不能满足此需求,而异构信号处理平台包含丰富处理器资源,成为首要选择。任务在不同处理器上执行时间不同,调度算法决定任务分配情况,直接影响信号处理应用完成时间和平台处理器效率,因此调度算法研究尤为重要。

目前,根据搜索过程把任务调度算法分为启发式和元启发式调度算法。启发式调度算法分为 3 种:基于列表的调度算法[1-5]运算复杂度低,易于实现,适用于任务和处理器数量较少的情形;基于聚类的调度算法[6-8]可扩展性和鲁棒性较强,但要求聚类后的类别数量小于处理器数量;基于任务复制的调度算法[9-10]可对其他算法进行优化,扩展性较强,但会提高额外计算开销,增加算法复杂度。元启发式调度算法分为蚁群优化算法[11-12]、遗传算法[13-14]、粒子群算法[15-16]等,该类算法通过不断迭代从广泛解空间中找到符合目标性能的最优解,适用于任务较多情形。其中蚁群算法凭借着灵活性高、自适应性强且能够得到较优解的特点得到了广泛应用,在旅行商问题(Traveling Salesman Problem,TSP)、二次分配问题(Quadratic Assignment Problem,QAP)和作业车间调度(Job Shop Scheduling Problem,JSP)问题中取得了较好实验结果。但蚁群算法存在初始信息素不足、收敛速度慢、易陷入局部最优解等问题,且以上调度算法主要优化任务调度长度,属于单目标调度,在最终调度方案中,处理器负载不均衡,造成资源浪费。

基于以上背景,本文提出了一种基于蚁群优化算法的负载均衡调度算法(Ant Colony Optimization Load Balancing,ACOLB)。该算法针对蚁群算法初始信息素不足、搜索空间广泛、搜索时间较长等不足,采用优化初始信息素矩阵,指定蚂蚁遍历任务顺序,运用轮盘赌选择方式寻求最优解,并结合优化目标对启发因子和状态转移公式进行改进,在调度长度和负载均衡方面的性能均有明显改善。

1　系统模型

异构信号处理平台总体分为 4 层,如图 1 所示:底层为硬件层包含平台所有处理器资源,如中央处理器(Central Processing Unit,CPU)、图形处理器(Graphics Processing Unit,GPU)、数字信号处理器(Digital Signal Processor,DSP)、现场可编程门阵列(Field Programmable Gate Array,FPGA)等;底层之上为中间层,由操作系统层、板级支持包、平台抽象层等组成,通过对操作系统、驱动等建模,实现不同处理器间数据传输;中间层之上为组件层,包含项目组研发的所有组件,并根据组件特点分类管理;顶层为应用层,包含当前平台部署的信号处理应用程序。异构信号处理平台增加新信号处理应用时,首先进行功能分析从组件库中挑选组件搭建应用程序,然后利用调度算法为应用程序中组件分配处理器,最后通过中间层将组件部署到相应处理器,完成新应用程序在异构信号处理平台上的开发运行。

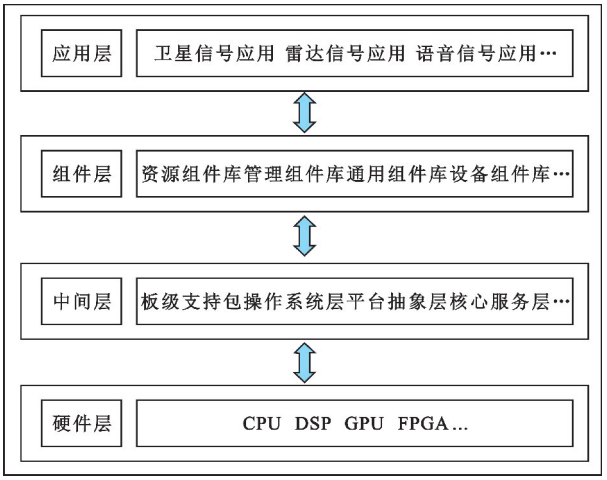

图 1　异构信号处理平台架构

信号处理应用在异构信号处理平台运行时,调度算法决定了其执行时间和各处理器的负载情况,对系统整体实时性和各处理器工作效率影响较大,因此调度算法尤为重要。调度问题研究中,将应用程序抽象成 DAG 图,定义为 $G=(V,C,P,W)$。经典 DAG 如图 2 所示。其中,$V=\{v_i\}$ 为任务集合与图中节点对应;$C=\{C_{ij}\}$ 为任务间平均通信开销与图中有向边对应,有向边表示任务间依赖关系和数据传递方向;$P=\{P_i\}$ 为处理器集合;$W=\{W_{ij}\}$ 为任务在处理器上的计算开销,与表 1 相对应。针对某通信信号处理应用,假设所有处理器是连接通畅的,每个任务不可再拆分必须分配到处理器执行,且任务执行期间其所在处理器不可以再执行其他任务,若两个任务分配到同一处理器,则任务间通信开销为零。

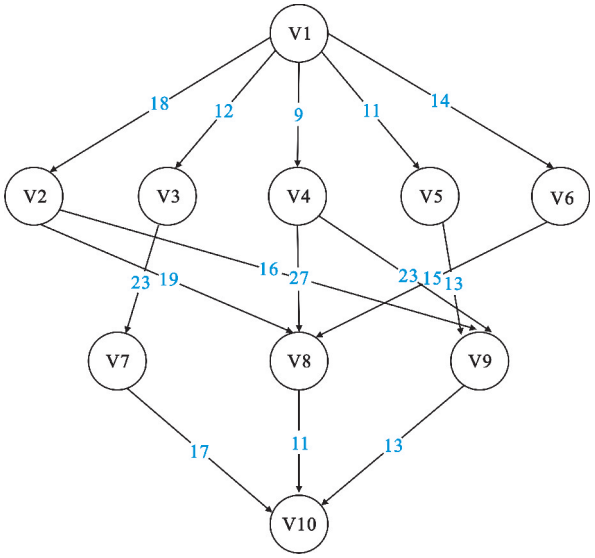

图 2 典型 DAG 任务图

表 1 典型任务图计算成本表

任务	P_1	P_2	P_3
1	17	16	9
2	13	19	22
3	11	13	19
4	13	8	17
5	12	13	10
6	13	16	9
7	7	13	11
8	5	11	14
9	18	12	20
10	21	7	16

2 ACOLB 算法

2.1 算法思想

蚁群算法灵感来自于真实世界中蚂蚁的觅食过程。蚁群会根据环境变换表现出不同行为特性。蚂蚁从 A 到 B 有 3 条路径可供选择,其长度关系是 2<1<3。蚂蚁在行走过程中释放信息素标记路径,信息素随着时间推移而蒸发,假设每条路径信息素蒸发系数和行进速度相同。对于最初蚂蚁,3 条路径初始信息素相同,随机选择路线,选择路线 2 耗时较短,残留信息素浓度较大;对于后面的蚂蚁,3 条线路信息素浓度有差异,选择 2 的概率明显提高,在动态正反馈机制作用下引导整个蚁群选择最佳路线,找到问题最优解。

在蚁群算法思想上构建异构信号处理平台中信号处理应用的调度模型,应用通过有向无环图抽象成 n 个任务、m 个处理器,蚂蚁需从任务列表中选择一个任务,通过处理器选择规则确定该任务运行的处理器,直到遍历完列表中所有任务。针对经典蚁群算法存在的不足,从任务优先级排序、信息素设置和更新、处理器选择三方面进行改进。

2.2 优先级排序

任务调度顺序对于最终调度结果十分关键,若有 n 个任务则调度顺序有 $n!$ 种,造成搜索空间广泛,收敛速度慢。本文采用异构最早完成时间(Heterogeneous Earliest Finish Time, HEFT)算法任务优先级排序规则,计算公式如下:

$$\mathrm{ranku}(i)=\overline{w_i}+\max_{v_j\in \mathrm{son}(v_i)}\left[C_{ij}+\mathrm{ranku}(j)\right]。\quad(1)$$

式中:$\overline{w_i}$ 为平均计算成本;$\mathrm{son}(v_i)$ 为任务 v_i 的子任务集合。按降序对 ranku 排序作为蚁群遍历任务顺序,缩小搜索范围,减少搜索时间,提高收敛速度。

2.3 信息素设置和更新

本文研究的通信密集型任务,任务间通信开销大于计算开销,降低通信开销有助于提升调度结果,而关键路径(Critical Path on a Processor, CPOP)算法采用关键路径任务分配到关键处理器思想,能极大减少通信开销,降低调度长度。因此,本文将 CPOP 调度结果作为初始信息素矩阵,解决初始信息素不足问题。以图 2 为例,CPOP 得到的任务分配情况如表 2 所示。根据该表对信息素矩阵 ***phe*** (pheromone)初始化设置,结果如公式(2)所示。

表 2 经典 DAG 图任务分配情况

任务	处理器
1	2
2	2
3	1
4	3
5	2
6	3
7	1
8	3
9	2
10	2

$$\boldsymbol{phe}=\begin{Bmatrix}1&2&1\\1&2&1\\2&1&1\\1&1&2\\1&2&1\\1&1&2\\2&1&1\\1&1&2\\1&2&1\\1&2&1\end{Bmatrix}。\tag{2}$$

式中:$\boldsymbol{phe}(j,i)$为任务 j 分配到处理器 i 的信息素浓度。受初始信息素矩阵引导,蚁群将向着降低通信开销方向不断迭代优化。每次迭代中蚂蚁为所有任务分配处理器,每只蚂蚁完成调度后的信息素增量 $\Delta\boldsymbol{phe}$ 计算公式如下:

$$\Delta\boldsymbol{phe}(j,i)=\Delta\boldsymbol{phe}(j,i)+\frac{Q}{L(i)}。\tag{3}$$

式中:$\Delta\boldsymbol{phe}(j,i)$为蚂蚁 i 为任务 j 分配处理器后引起信息素增量;Q 为信息素增加系数;$L(i)$为蚂蚁在本次迭代中取得的调度长度。为获得更加全面的信息素矩阵,充分利用蚁群群体行为优势,将每只蚂蚁的 $\Delta\boldsymbol{phe}$ 累加后对信息素 $\boldsymbol{phe}$ 进行更新,更新公式如下:

$$\boldsymbol{phe}=(1-\rho)\times\boldsymbol{phe}+\Delta\boldsymbol{phe}(j,i)。\tag{4}$$

式中:ρ 为信息素蒸发系数;$\boldsymbol{phe}$ 按照蚁群迭代频次进行更新。

2.4　处理器选择

ACOLB 算法通过对传统蚁群算法状态转移公式和启发因子改进,在调度长度基础上增加负载均衡优化目标。启发因子由 eta 和 etb 两部分组成。eta 计算公式如下:

$$\text{eta}=\frac{1}{\boldsymbol{EFT}}。\tag{5}$$

式中:$\boldsymbol{EFT}$ 为蚂蚁完成时间矩阵。etb 计算公式如下:

$$\text{etb}=\frac{1}{\boldsymbol{num\_process}}。\tag{6}$$

式中:$\boldsymbol{num\_process}$ 为处理器实时负载矩阵。ACOLB 状态转移公式如下:

$$p(i,k)=\frac{[\boldsymbol{phe}(i,k)]^{\alpha}\times[\text{eta}(i,k)]^{\beta}\times[\text{etb}(k)]^{\chi}}{\sum_{k=1}^{m}\{[\boldsymbol{phe}(i,k)]^{\alpha}\times[\text{eta}(i,k)]^{\beta}\times[\text{etb}(k)]^{\chi}\}}。\tag{7}$$

式中:$p(i,k)$,$\boldsymbol{phe}(i,k)$和 $\text{eta}(i,k)$分别为任务 i 分配到处理器 k 上的概率、信息素浓度和完成时间倒数;$\text{etb}(k)$为处理器 k 累计任务量倒数;α,β,χ 分别是 $\boldsymbol{phe}$,eta 和 etb 的重要程度因子。

得到任务选择处理器概率后,运用轮盘赌方法确定任务运行处理器。首先计算任务 i 分配到处理器 1~k 的概率之和 $p(i,k)'$,公式如下:

$$p(i,k)'=\sum_{j=1}^{k}p(i,j)。\tag{8}$$

之后,从(0,1)内生成一个随机数 rand,确定处理器集合 find_process,公式如下:

$$\text{find\_process}=p(i,k)'\geqslant\text{rand}。\tag{9}$$

选取 find_process 中编号最小处理器,作为任务 i 最终分配处理器。

2.5　算法步骤

ACOLB 算法流程如图 3 所示,具体步骤如下:

输入:DAG

输出:最优解

1 计算优先级列表和 CPOP 调度结果;

2 设置蚁群规模,迭代次数,$\boldsymbol{phe}$,α,β,χ,ρ 和 Q;

3 迭代次数加 1;

4 将上一次迭代中:EST,$\boldsymbol{EFT}$,$\boldsymbol{num\_process}$,eta 和 etb 进行重置;

5 任务数加 1;

6 蚂蚁数加 1;

7 分配处理器;

8 更新 eta 和 etb;

9 如果有蚂蚁未对该任务进行调度,转到步骤 6;

10 如果有任务未分配处理器,转到步骤 5;

11 记录此次迭代中最佳调度结果;

12 更新信息素矩阵;

13 如果迭代次数不大于最大迭代值,转到步骤 3。

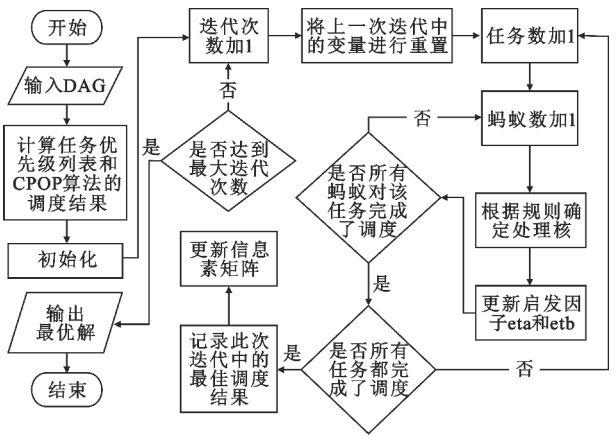

图 3　ACOLB 算法流程

3　仿真实验及性能分析

实验平台是 CPU 为 Intel(R) Core(TM) i5-1135G7 @ 2.40 GHz 、内存为 16 GB 的 64 位操作系

统。随机 DAG 生成程序参数设置为任务数 $V=\{10,20,40,60,80\}$,处理器数 $P=\{3,4,5,6\}$,通信计算比 CCR = {0. 1,0. 25,0. 5,0. 75,1,2. 5,5,7. 5,10};并行因子 Parallel = {0. 1,0. 5,1,5,10}。CPOP 算法对于通信密集型任务调度效果较好,因此将其作为对比算法,通过图 1 所示的典型 DAG 和随机 DAG,验证 ACOLB 的有效性和可靠性。

3.1 优化目标

本文所提算法优化目标有两个,即调度长度 makespan 和负载均衡系数 pld。makespan 表示算法整体执行时间,越小越好,计算公式如下:

$$\text{makespan}=\text{EFT}(v_{\text{exit}})。\tag{10}$$

式中:v_{exit} 为出口任务。

pld 表示系统内所有处理器负载均衡程度,越小越好,其计算公式如下:

$$\text{pld}=\sqrt{\frac{\sum_{i=1}^{m}(\text{nump}(i)-\overline{\text{nump}})^2}{m}}。\tag{11}$$

式中:nump(i)为处理器最终分配任务数;$\overline{\text{nump}}$ 为处理器平均分配任务数。

3.2 参数设置

设置 50 只蚂蚁,进行 1 000 次迭代,信息素增加系数为 1,信息素蒸发系数为 0. 25,phe 重要程度因子 α 为 1,eta 重要程度因子 β 取值范围为 0~2,etb 重要程度 χ 为 0~1,其中 α,β,χ 取值范围是通过大量实验确定的。

3.3 ACOLB 有效性分析

为验证算法有效性,采用经典 DAG 和随机 DAG 进行实验,从调度长度和负载均衡两方面进行对比。

实验 1:经典 DAG 如图 1 所示,其计算成本如表 1 所示。ACOLB 调度长度如图 4 所示,CPOP 负载情况如图 5 所示,ACOLB 如图 6 所示。

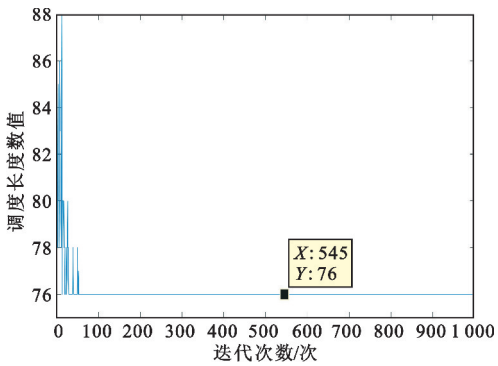

图 4 ACOLB 的调度长度

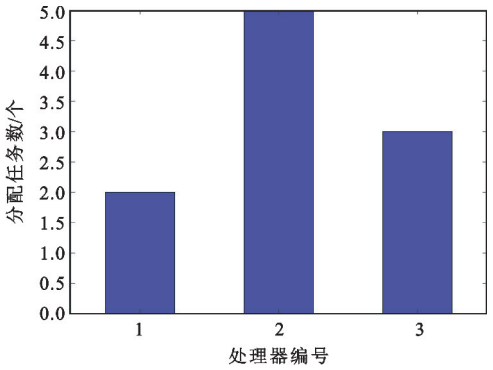

图 5 CPOP 处理器负载情况

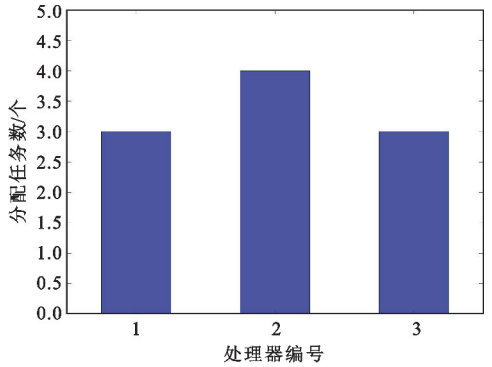

图 6 ACOLB 处理器负载情况

仿真得到 CPOP 调度长度为 86,负载均衡系数为 1. 247 2;ACOLB 调度长度为 76,负载均衡系数为 0. 471 4,ACOLB 调度长度和负载均衡均优于 CPOP。ACOLB 前期调度结果下降到某个值时,陷入局部最优出现振荡,后通过轮盘赌策略跳出,继续寻找全局最优解。ACOLB 负载情况遵循 CPOP 调度结果引导,2 号处理器仍是关键处理器,但较 CPOP 处理器负载分配情况,能更加均衡最大发挥各处理器性能,提高效率。

实验 2:随机生成一个任务数为 20,处理器数为 5 的 DAG,应用两种算法得到的调度结果如表 3 所示。

表 3 调度结果

算法	makespan	pld
CPOP	298	3. 162 3
ACOLB	194	1. 414 2

从随机 DAG 结果可以看出,ACOLB 调度长度和负载均衡系数均优于 CPOP。对处理器负载均衡之后,关键处理器同 CPOP 保持一致,能够极大减少任务间通信开销。

从以上两个实验可以得出,ACOLB 不仅对于经典 DAG 有改进效果,对于随机 DAG 同样可以起到

优化调度长度和负载均衡的作用,证明了 ACOLB 算法的有效性。

3.4　ACOLB 可靠性分析

为验证算法可靠性,对处理器和任务数量采用控制变量原则,从调度长度和负载均衡两方面进行对比。

实验 3:随机生成一组任务数为 20,处理器数分别为 3,4,5,6 的 DAG。两种算法调度长度结果对比如图 7 所示,处理器负载均衡系数对比如图 8 所示,对结果进行计算得到调度长度优化率和负载均衡优化率。

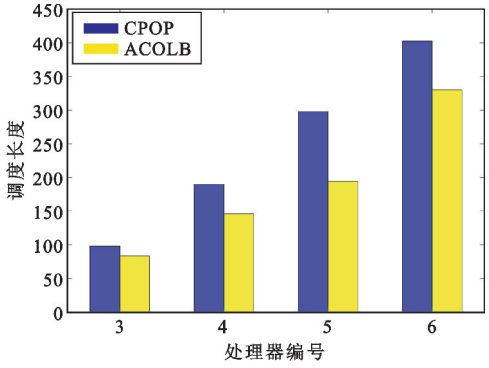

图 7　任务调度长度对比(实验 3)

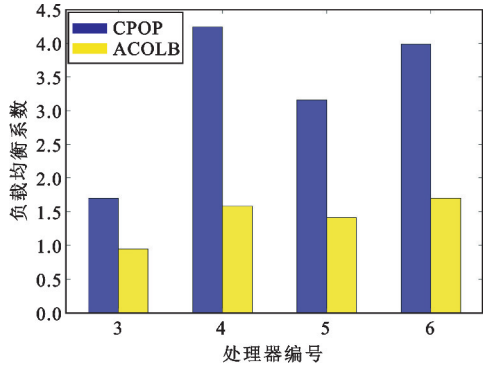

图 8　处理器负载均衡系数对比(实验 3)

从实验结果可看出,任务数目相同处理器数目不同时 ACOLB 算法依然可靠,相比于 CPOP 算法,其在调度长度和负载均衡方面均有改进,本组实验中调度长度优化率最小值是 15.3%,负载均衡优化率最小值是 44.5%。

实验 4:随机生成一组处理器数为 6,任务数分别为 20,40,60 随机任务图。两种算法调度长度结果对比如图 9 所示,处理器负载均衡系数对比如图 10 所示,对结果进行计算得到调度长度优化率和负载均衡优化率。

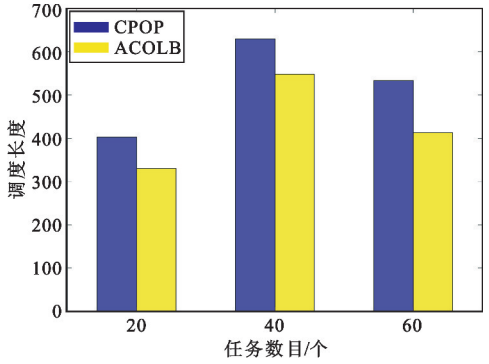

图 9　任务调度长度对比(实验 4)

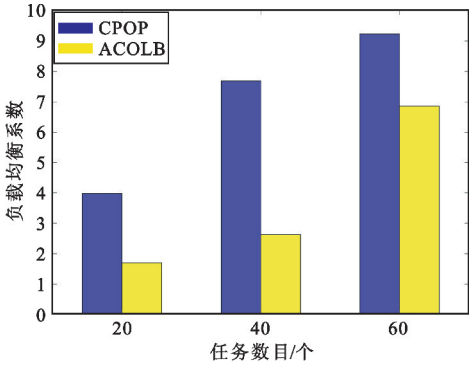

图 10　处理器负载均衡系数对比(实验 4)

从实验结果看出,处理器数相同任务数不同时,ACOLB 算法依然可靠,本组实验中调度长度优化率最小值是 13%,负载均衡优化率最小值 25%。

从实验 3 和实验 4 调度结果可看出,对于随机 DAG,ACOLB 算法仍然可以达到优化效果,其中调度长度优化率在 13% 以上;负载均衡改善效果明显,优化率在 25% 以上,实现了处理器负载均衡,提高了系统整体效能。

4　结束语

本文针对当前异构信号处理平台信号处理应用的调度算法中处理器负载不均衡造成系统资源利用不合理的问题,提出了 ACOLB 算法。该算法在蚁群优化算法基础上,利用 CPOP 调度结果解决初始信息素不足问题;利用 HEFT 调度列表缩小搜索空间,提升搜索速度;利用轮盘赌策略跳出局部最优解;增加负载均衡启发因子均衡调度结果。通过仿真得出,ACOLB 算法在任务调度长度和负载均衡方面均有改进,证明了算法的有效性和可靠性。其中调度长度优化率为 13%,优化效果一般,是下一步研究方向;负载均衡优化率为 25%,优化效果明显。运用该算法可解决处理器负载不均衡问题,使异构

信号处理平台负载更加均衡,发挥最大效能。

参考文献:

[1] TOPCUOGLU H, HARIRI S, WU M Y. Performance-effective and low-complexity task scheduling for heterogeneous computing [J]. IEEE Transactions on Parallel and Distributed Systems,2002,13(3):260-274.

[2] DOROSTKAR F, MIRZAKUCHAKI S. List scheduling for heterogeneous computing systems introducing a performance-effective definition for critical path [C]// Proceedings of 2019 9th International Conference on Computer and Knowledge Engineering. Mashhad: IEEE, 2020:356-362.

[3] WANG G, GUO H, WANG Y. A novel heterogeneous scheduling algorithm with improved task priority [C]// Proceedings of 2015 IEEE 17th International Conference on High Performance Computing and Communications. New York: IEEE,2015:1826-1831.

[4] IJAZ S, MUNIR E U. MOPT: list-based heuristic for scheduling workflows in cloud environment [J]. The Journal of Supercomputing,2019,75(7):3740-3768.

[5] BITTENCOURT L F, SAKELLARIOU R, MADEIRA E R M. DAG scheduling using a lookahead variant of the heterogeneous earliest finish time algorithm [C]// Proceedings of 2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing. Pisa: IEEE,2010:27-34.

[6] ZHANG L, YU D, ZHENG H. Optimization of cloud workflow scheduling based on balanced clustering[C]// Proceedings of 2017 International Symposium on Cyberspace Safety and Security. Cham: Springer, 2017: 352-366.

[7] YU D, YING Y, ZHANG L, et al. Balanced scheduling of distributed workflow tasks based on clustering [J]. Knowledge-Based Systems,2020,199:1059301-1059313.

[8] KANEMITSU H, HANADA M, NAKAZATO H. Clustering-based task scheduling in a large number of heterogeneous processors [J]. IEEE Transactions on Parallel and Distributed Systems,2016,27(11):3144-3157.

[9] 刘静. 考虑通信和时间限制的异构多核系统调度理论与方法[D]. 长沙:湖南大学,2015.

[10] HE K, MENG X, PAN Z, et al. A novel task-duplication based clustering algorithm for heterogeneous computing environments[J]. IEEE Transactions on Parallel and Distributed Systems,2019,30(1):2-14.

[11] 王裕健. 云计算任务调度中改进的蚁群算法的研究[D]. 北京:华北电力大学,2021.

[12] 丛慧,夏永祥,李悦,等. 同步卫星转发器二维时频资源蚁群调度算法[J]. 电讯技术,2017,57(5):540-547.

[13] FANG Y, XIAO X, GE J. Cloud computing task scheduling algorithm based on improved genetic algorithm [C]//Proceedings of 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference. Chengdu: IEEE, 2019: 852-856.

[14] ZHOU Z, LI F, ZHU H, et al. An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments [J]. Neural Computing and Applications,2020,32(6):1531-1541.

[15] JANA B, CHAKRABORTY M, MANDAL T. Soft computing: theories and applications [M]. Singapore: Springer,2019.

[16] ZHANG Y, YANG R. Cloud computing task scheduling based on improved particle swarm optimization algorithm [C]//Proceedings of the 43rd Annual Conference of IEEE Industrial Electronics Society. Beijing: IEEE, 2017:8768-8772.

作者简介:

沈小龙 男,1994 年生于内蒙古商都,2017 年获学士学位,现为硕士研究生,主要研究方向为异构信号处理平台、资源调度。

马金全 男,1975 年生于甘肃张掖,2013 年获博士学位,现为副教授,主要研究方向为通信信号处理与软件无线电。

胡泽明 男,1977 年生于河南新县,2008 年获博士学位,现为副教授,主要研究方向为软件无线电和异构平台信息处理技术。

李宇东 男,1995 年生于河北易县,2017 年获学士学位,现为硕士研究生,主要研究方向为异构信号处理平台任务部署。