doi:10.3969/j.issn.1001-2400.2016.01.009

考虑处理机释放时间的可分任务调度优化模型

王晓丽，王宇平，孟 坤
(西安电子科技大学 计算机学院,陕西 西安 710071)

摘要：可分任务调度是近年来信息技术领域研究的热点课题.已有的可分任务调度模型大多假设所有处理机在任务分配之初全部处于空闲状态,而实际上,当新的任务到来时很多处理机可能尚处于忙碌状态.每台处理机从忙碌状态转到空闲状态的时间不同,即处理机可能具有不同的释放时间.在充分考虑处理机释放时间不同的基础上,建立了一种新的混合时序约束的可分任务调度模型,并设计了高效的全局优化遗传算法求解该模型.实验结果表明了模型的合理性和算法的有效性.

关键词：可分任务调度;释放时间;混合时序约束;遗传算法
中图分类号:TP301 **文献标识码**:A **文章编号**:1001-2400(2016)01-0047-07

Release time aware divisible-load scheduling optimization model

WANG Xiaoli, WANG Yuping, MENG Kun
(School of Computer Science and Technology, Xidian Univ., Xi'an 710071, China)

Abstract: Divisible-load scheduling has become an increasingly hot subject in the research on information technologies in recent years. Most existing divisible-load scheduling models assume that all processors are idle at the beginning of workload assignment. In fact, many processors may still in the busy state when a new workload arrives. Processors may have different waiting times from the busy state to the idle, that is, processors have different release times. This paper proposes a new release time aware divisible-load scheduling model with hybrid time constraints and designs an effective global optimization genetic algorithm to solve it. Finally, experimental results show the effectiveness of the proposed model and the efficiency of the proposed algorithm.

Key Words: divisible-load scheduling; release time; hybrid time constraints; genetic algorithm

在并行与分布式系统中,大规模计算任务的处理逐渐成为了极具挑战性的课题.为系统建立合理且贴近实际的任务调度模型成了该领域最大的难点,其中可分任务调度模型[1-3]就是典型的代表之一.可分任务是指任务可以被切分为任意大小的若干子任务且子任务间不具有优先关系.在并行与分布式系统下,可分任务调度的主要目标是寻求合理的任务调度策略,使得任务的完成时间最短.

针对现实生活中常见的线性网路、总线型网络和树形网络,已经有很多文献研究可分任务调度方案的最优解.例如,文献[4]给出了齐次线性网络中任务完成时间的紧式耦合解,文献[5-6]给出了齐次树形网络与总线型网络下任务完成时间的渐进解.文献[7]给出了异构星形网络下任务最短完成时间的紧式耦合解,且证明了在遵循通信速率递减作为处理机调度顺序的情况下任务的完成时间最短.对于异构树形网络,文献[8]的研究表明处理机的调度顺序只依赖于节点的通信速率而非计算速率.然而,这些研究成果都没有考虑到处理机的计算启动开销和网络的通信启动开销.文献[9-11]研究了具有常启动开销的总线型网络中启动开销和处理机调度序列的变化对任务完成时间的影响,证明了当遵循计算速率递减作为处理机调度顺序时

收稿日期:2014-08-11 **网络出版时间**:2015-04-14
基金项目:国家自然科学基金资助项目(61402350,61472297,61272119);中央高校基本科研业务费专项资金资助项目(JB150307)
作者简介:王晓丽(1987－),女,讲师,博士,E-mail: wangxiaoli@mail.xidian.edu.cn.
网络出版地址:http://www.cnki.net/kcms/detail/61.1076.TN.20150414.2046.006.html

将获得任务的最短完成时间. 文献[12]的研究表明,在具有任意启动开销的异构分布式环境下,若任务足够大,则遵循通信速率递减作为处理机调度顺序时任务的完成时间最短. 文献[13] 将可分任务调度模型扩展到多源网格环境中,文献[14-15]将其扩展到云计算平台,文献[16]将其扩展到无线传感器网络中,文献[17-19]将其扩展到实时环境中.

已有的研究大多假定处理机在任务到来时均处于空闲状态,然而在实际的并行与分布式应用场景中这个假设并非总是成立的. 当新的任务到来时,处理机可能尚未完成前一个任务,从而处于忙碌状态,不能立即参与新任务的计算. 笔者将新任务到来后,处理机由忙碌状态转变为空闲状态的时间间隔称为该处理机的释放时间. 考虑处理机释放时间的可分任务调度的研究尚处于起步阶段,文献[20]给出了一种穷举方法用于同构总线型网络的可分任务调度问题. 为了避免穷举法的巨额时间开销,笔者构建了一种新的考虑释放时间的可分任务调度智能优化模型,并提出了一种高效的遗传算法对模型进行求解.

1 考虑释放时间的可分任务调度优化模型

设 $N+1$ 台处理机通过星形网络互连,其中 P_0 为主处理机,$\{P_i \mid i \in (1,2,\cdots,N)\}$ 为从处理机,所有处理机的计算速率相同;从处理机 P_i 的释放时间记为 r_i. l_i 是连接 P_0 和 P_i 的通信链路,所有链路的传输速率相同. 设待分配的任务大小为 W_{total},参与计算的处理机数目为 n. P_0 仅负责将任务划分成 n 个子任务 $\alpha_1,\alpha_2,\cdots,\alpha_n$,并将这些子任务依次调度到从处理机 $P_1,P_2,\cdots,P_n$ 上完成并行计算,其中 $\sum_{i=1}^{n}\alpha_i=W_{\text{total}}$. 从处理机的调度顺序遵从释放时间递增的顺序,即 $r_1 \leqslant r_2 \leqslant \cdots \leqslant r_N$. P_0 传输大小为 α_i 的子任务给从处理机 P_i 所需的时间为 $z\alpha_i$,其中 z 为链路传输单位大小任务所花费的时间. 处理机 P_i 计算子任务 α_i 所需时间为 $w\alpha_i$,其中 w 为处理机计算单位任务所需的时间. 问题在于如何划分任务使得任务完成的时间达到最短.

称处理机 $P_i(i=1,2,\cdots,n)$ 开始从主处理机接收任务的时刻为 P_i 的开始时刻,记为 s_i. 给定所有处理机的释放时间,下面分 3 种情况讨论处理机释放时间与开始时间之间可能满足的约束条件:

(1) $r_{i+1} \leqslant s_i + z\alpha_i, i=1,2,\cdots,n-1$. 处理机 P_{i+1} 的释放时间 r_{i+1} 早于主处理机 P_0 给 P_{i+1} 分配任务的时刻,即 P_{i+1} 由忙碌状态转为空闲状态发生在 P_0 给 P_i 传输任务 α_i 的过程中.

文献[3]证明了只有当所有处理机同时完成计算时任务的完成时间最短,否则完全可以将后完成计算的处理机上的部分任务调度到先完成计算的处理机上执行. 由所有处理机同时完成计算,可以得到

$$s_i + z\alpha_i + w\alpha_i = s_{i+1} + z\alpha_{i+1} + w\alpha_{i+1} \quad , \quad i=1,2,\cdots,n-1 \quad . \tag{1}$$

由图 1 可以看出,P_i 的开始时间 s_i 满足

$$s_i = s_{i-1} + z\alpha_{i-1} = r_1 + z\left(\sum_{j=1}^{i-1}\alpha_j\right) \quad , \quad i=2,3,\cdots,n \quad . \tag{2}$$

由式(1)和式(2)可得

$$\alpha_i = q\alpha_{i-1} = q^{i-1}\alpha_1 \quad , \quad q=w/(z+w) \quad , \quad i=2,3,\cdots,n \quad . \tag{3}$$

已知 $\sum_{i=1}^{n}\alpha_i=W_{\text{total}}$,带入式(3),可得

$$\alpha_1 = W_{\text{total}}\Big/\left(\sum_{i=1}^{n}q^{i-1}\right) = W_{\text{total}}(1-q)/(1-q^n) \quad , \quad q=w/(z+w) \quad . \tag{4}$$

(2) $r_{i+1} > s_i + z\alpha_i$, $i=1,2,\cdots,n-1$. 处理机 P_{i+1} 的释放时间 r_{i+1} 晚于 P_0 给 P_i 传输完任务 α_i 的时刻,即 P_0 在给 P_i 传输完任务 α_i 后需要等待一段时间直到 P_{i+1} 恢复空闲才能为其传输任务 α_{i+1}. 同样地,由所有处理机同时完成计算可得式(1). 处理机 $P_i(i=1,2,\cdots,n)$ 的释放时间 r_i 和开始时间 s_i 是相等的,即 $s_i=r_i$. 将式(1) 中的 s_i 替换为 r_i 可得

$$\alpha_i = \alpha_1 + (r_1 - r_i)/(z+w) \quad , \quad i=2,3,\cdots,n \quad . \tag{5}$$

(3) 混合时序约束. 任意两个相邻的从处理机之间可能满足约束条件(1),也可能满足约束条件(2). 若有 n 台处理机参与计算,所有可能的情况有 $2^{(n-1)}$ 种,图 1 给出了其中一种可能的情况.

将处理机 P_{i-1} 和 P_i 满足约束条件(1)和条件(2)的情况分别记为 $\Gamma_i(1)$ 和 $\Gamma_i(2)$,其中 $i=2,3,\cdots,n$. 用 $C=(c_2,c_3,\cdots,c_n)$ 表示一种混合时序约束,其中 $c_i\in\{\Gamma_i(1),\Gamma_i(2)\}$. 若 $c_i=\Gamma_i(1)$,表明主处理机 P_0 在给 P_{i-1} 传输完数据后紧接着为 P_i 传输数据,中间没有空闲. 此时,处理机 P_i 的开始时间满足 $s_i=s_{i-1}+z\alpha_{i-1}$. 若 $c_i=\Gamma_i(2)$,表明主处理机 P_0 在给 P_{i-1} 传输完数据后需要等待 P_i 由忙碌转为空闲状态才能开始传输数据,中间存在空闲时间. 此时,处理机 P_i 的开始时间满足 $s_i=r_i$.

以图 1 为例,其混合时序约束 C 满足 $C=\big(\Gamma_2(1),\cdots,\Gamma_k(1),\Gamma_{k+1}(2),\Gamma_{k+2}(2),\cdots,\Gamma_{k+m}(2),\Gamma_{k+m+1}(1),\cdots,\Gamma_n(1)\big)$.

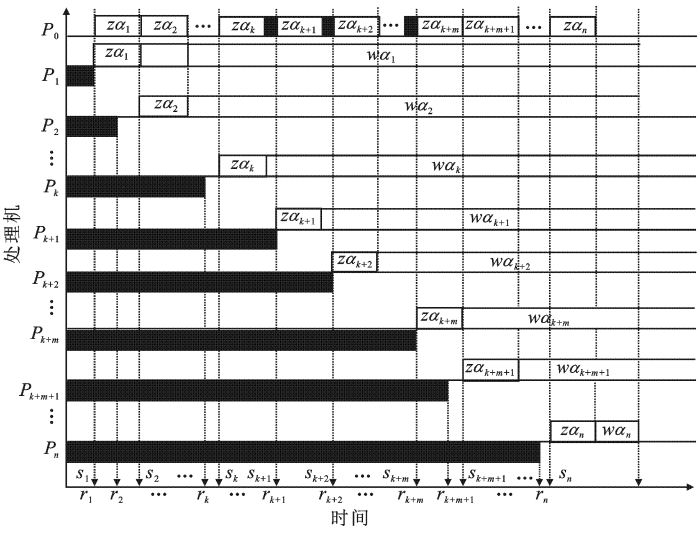


图 1　一种满足混合时序约束条件的可分任务调度图

根据混合时序约束 C,可以得到相邻处理机开始时间之间的关系:

$$\begin{cases} s_1=r_1 \quad , \\ s_2=s_1+z\alpha_1 \quad , \\ \vdots \\ s_k=s_{k-1}+z\alpha_{k-1} \quad , \\ s_{k+1}=r_{k+1} \quad , \\ s_{k+2}=r_{k+2} \quad , \\ \vdots \\ s_{k+m}=r_{k+m} \quad , \\ s_{k+m+1}=s_{k+m}+z\alpha_{k+m} \quad , \\ \vdots \\ s_n=s_{n-1}+z\alpha_{n-1} \quad . \end{cases} \tag{6}$$

将式(6)带入式(1),可得每台处理机分配的任务 α_i:

$$\begin{cases} \alpha_2=q\alpha_1 \quad , \\ \vdots \\ \alpha_k=q\alpha_{k-1} \quad , \\ \alpha_{k+1}=\alpha_1+(r_1-r_{k+1})/(z+w) \quad , \\ \alpha_{k+2}=\alpha_1+(r_1-r_{k+2})/(z+w) \quad , \\ \vdots \\ \alpha_{k+m}=\alpha_1+(r_1-r_{k+m})/(z+w) \quad , \\ \alpha_{k+m+1}=q\alpha_{k+m} \\ \vdots \\ \alpha_n=q\alpha_{n-1} \quad . \end{cases} \tag{7}$$

将式(7)中的 $n-1$ 个等式同 $\sum_{i=1}^{n}\alpha_i=W_{\text{total}}$ 共 n 个等式表示成标准形式,即

$$\boldsymbol{A}\cdot\boldsymbol{\alpha}=\boldsymbol{b} \quad ,$$

其中 $\boldsymbol{\alpha}$ 表示的是待求的 $n\times1$ 维解变量 $(\alpha_1,\alpha_2,\cdots,\alpha_n)$,$\boldsymbol{A}$ 是 $n\times n$ 的系数矩阵,$\boldsymbol{b}$ 是 $n\times1$ 维的向量. 在图 1 给出的任务调度图中,$\boldsymbol{A}$ 和 $\boldsymbol{b}$ 可以分别表示为

$$\boldsymbol{A}=\begin{bmatrix}
-q & 1 & 0 & \cdots & & & & & & & 0\\
\vdots & \ddots & \ddots & \ddots & \cdots & & & & & & \vdots\\
\vdots & \ddots & \ddots & \ddots & \ddots & \cdots & & & & & \vdots\\
0 & \cdots & 0 & -q & 1 & 0 & \cdots & & & & 0\\
-1 & 0 & \cdots & & 0 & 1 & 0 & \cdots & & & 0\\
-1 & 0 & & \cdots & & 0 & 1 & 0 & \cdots & & 0\\
\vdots & \vdots & & & \cdots & & \ddots & \ddots & & \cdots & \vdots\\
-1 & 0 & & & & \cdots & & 1 & 0 & \cdots & 0\\
0 & 0 & & & & & \cdots & -q & 1 & & \\
\vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots\\
0 & & & & & & & \cdots & 0 & -q & 1 & 0\\
0 & 0 & & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & -q & 1\\
1 & 1 & & & \cdots & 1 & 1 & & & \cdots & 1 & 1
\end{bmatrix}, \tag{8}$$

$$\boldsymbol{b}=\left[0,\cdots,0,\frac{r_1-r_{k+1}}{z+w},\frac{r_1-r_{k+2}}{z+w},\cdots,\frac{r_1-r_{k+m}}{z+w},0,\cdots,0,W_{\text{total}}\right]^{\mathrm{T}}. \tag{9}$$

给定一种混合时序约束 $C=(c_2,c_3,\cdots,c_n)$,就可以求得一个形如 $\boldsymbol{A}\cdot\boldsymbol{\alpha}=\boldsymbol{b}$ 的标准式,进而可以采用线性规划的方法求出任务分配方案 $\boldsymbol{\alpha}$ 的解. 下面给出考虑释放时间的可分任务调度模型:

$$\begin{cases}
\min\limits_{n,C}(\boldsymbol{T})=\min\left(r_1+(z+w)\alpha_1\right), \\
\text{s.t.}\begin{cases}
①\ 0<n\leqslant N,\text{其中 } N \text{ 为处理机的总数},n \text{ 为参与计算的处理机数量};\\
②\ 0<\alpha_i\leqslant W_{\text{total}},\quad i=1,2,\cdots,n;\\
③\ \sum\limits_{i=1}^{n}\alpha_i=W_{\text{total}};\\
④\ s_i+z\alpha_i+w\alpha_i=s_{i+1}+z\alpha_{i+1}+w\alpha_{i+1},\quad i=1,2,\cdots,n-1;\\
⑤\ C=(c_2,c_3,\cdots,c_n),c_i\in\{\Gamma_i(1),\Gamma_i(2)\},\quad i=2,3,\cdots,n;\\
⑥\ \text{若 } c_i=\Gamma_i(1),\ r_{i+1}\leqslant s_i+z\alpha_i;\ \text{若 } c_i=\Gamma_i(2),\ r_{i+1}>s_i+z\alpha_i,\text{其中},i=2,3,\cdots,n;\\
⑦\ \text{若 } c_i=\Gamma_i(1),\ s_i=s_{i-1}+z\alpha_{i-1};\ \text{若 } c_i=\Gamma_i(2),\ s_i=r_i,\text{其中},i=2,3,\cdots,n.
\end{cases}
\end{cases}$$

模型的目标是任务的完成时间最短. 约束①表示并非所有的处理机都参与计算;约束②和③表示从处理机分配的子任务均非负,且所有子任务之和为总任务 W_{total};约束④表示所有处理机必须同时完成计算;约束⑤和⑥限定了混合时序约束条件 C 的取值范围;约束⑦限定了处理机 P_i 的开始时间 s_i 的取值范围.

2 全局优化遗传算法

遗传算法已经被证明能很好地解决工程技术领域的优化、设计、控制等实际应用问题,尤其是针对任务调度等 NP-hard 的组合优化问题[21]. 因此,笔者设计了一种新的全局优化遗传算法来求解上文提出的混合时序约束可分任务调度模型.

2.1 编码与解码

笔者采用实数编码方式,将混合时序约束的可分任务调度问题表示成一个向量 $\boldsymbol{I}=(n,\boldsymbol{H})$,其中 n 表示参与计算的处理机数目,种群初始化时设为处理机总数 N;$\boldsymbol{H}=(h_2,h_3,\cdots,h_N)$,表示一种混合时序约束,$h_i\in\{1,0\}$. 若 $h_i=1$,表示处理机 P_{i-1} 和 P_i 满足约束条件(1);反之,$h_i=0$,表示满足约束条件(2).

举例说明:假设共有 6 台处理机,一种可能的编码方案为 $\boldsymbol{I}=(6,1,0,1,1,0)$,其中第 1 位 $n=6$,表示参与计算的处理机数目(初始时设定为处理机总数);$h_2=1$,表示第 1 和 2 台处理机满足约束(1);$h_3=0$,表示第 2 和 3 台处理机满足约束(2),第 3 和 4 台满足约束(1),第 5 和 6 台满足约束(1),第 7 和 8 台满足约束(2).

对于任意一个个体 $\boldsymbol{I}$,给定 n 和 $\boldsymbol{H}$,就可以得到 $n\times n$ 的系数矩阵 $\boldsymbol{A}$ 和对应的向量 $\boldsymbol{b}$,并将问题转化为

$\boldsymbol{A}\cdot\boldsymbol{\alpha}=\boldsymbol{b}$ 的标准形式. 通过线性规划方法求解该标准式,可以得到任务分配方案 $\boldsymbol{\alpha}$ 的解. 验证求解出的 $\boldsymbol{\alpha}$ 是否满足模型所有的约束条件,尤其是约束②和⑤. 若满足模型的所有约束条件,则个体 $\boldsymbol{I}$ 对应惟一一个类似于图 1 的混合时序可分任务调度图,调度方案 $\boldsymbol{\alpha}$ 即为可行解. 将该方案对应的任务完成时间 $T=r_1+(z+w)\alpha_1$ 作为个体 $\boldsymbol{I}$ 的适应度值. 如果 $\boldsymbol{\alpha}$ 不满足模型的部分约束条件,则表明并不需要这么多的处理机参与计算,令 $n=n-1$,更新个体 $\boldsymbol{I}$, 重复这个过程,直至求出一个满足模型全部约束条件的解.

2.2　交叉与变异

交叉操作是遗传算法的主要进化手段,且模拟了生物繁殖和基因重组机理,通过两两染色体基因交叉互换,繁殖两个新的子代个体,并将父代好的基因遗传至子代个体. 交叉操作保持了遗传算法种群个体的多样性和全局搜索能力. 笔者采用两点交叉方式生成新个体: 随机生成两个整数 p 和 q,满足 $2\leqslant p<q\leqslant N$,作为交叉点,将两个父代个体交叉点之间的基因进行交换,生成两个后代个体. 由于个体第 1 位表示参与计算的处理机数目,因此交叉后的后代个体第 1 位均置为处理机总数 N.

变异操作是推动遗传算法进化的重要手段. 通过一定概率的基因变异,保持种群多样性,强化了遗传算法的局部搜索能力. 笔者采用单点变异方式生成新个体: 随机生成一个整数 p,满足 $2\leqslant p\leqslant N$,作为变异点,将个体在该点的基因位取反,产生新的后代个体;同时,将后代个体的第 1 位置为处理机总数 N.

2.3　全局优化遗传算法

下面给出具体的遗传算法框架.

算法 1　求解可分任务调度模型的全局优化遗传算法.

(1) 初始化. 根据编码规则随机生成初始种群 $P(0)$. 令进化代数 $t=0$.

(2) 交叉. 以概率 p_{cros} 从 $P(t)$ 之中选择父代个体进行两点交叉. 交叉获得的全部后代个体定义为集合 O_1.

(3) 变异. 以概率 p_{mut} 从集合 O_1 中选择个体进行变异. 新的后代个体定义为集合 O_2.

(4) 选择. 从集合 $P(t)\cup O_1\cup O_2$ 中选择最优的 E 个个体直接保留到下一代种群以加快收敛速度. 使用轮盘赌选择操作从集合 $P(t)\cup O_1\cup O_2$ 选择 $N-E$ 个个体保留到下一代种群 $P(t+1)$ 中. 令 $t=t+1$.

(5) 终止条件. 如果满足终止条件,则终止算法; 否则,转向步骤(2).

3　实验与结果分析

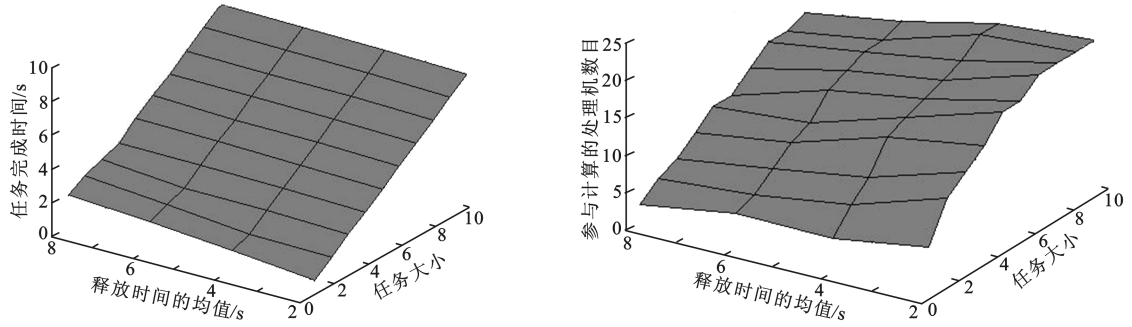
针对笔者提出的模型和算法,对比已有算法进行了多组实验. 系统参数设置如下: 处理机个数 $N=20$, $z=0.8$, $w=1.2$,处理机 $P_1\sim P_{20}$ 的释放时间 $r_1\sim r_{20}$ 是指数分布的随机数. 遗传算法中参数设置如下: 种群大小 $S_{\mathrm{P}}=100$,交叉概率 $p_{\mathrm{cros}}=0.6$,变异概率 $p_{\mathrm{mut}}=0.02$,精英保留个数 $E=5$,终止条件为进化代数 $t=100$.

表 1 给出了不同任务量情况下 ($W_{\mathrm{total}}=1.0\sim10.0$) 两种算法对比的实验结果,其中 GA 代表笔者提出的全局优化遗传算法,EA 代表文献[20]提出的穷举算法(Exhaustive Algorithm). 从表 1 可以看出,两个算法针对同样大小的任务,调度后得到的参与计算的处理机数目和任务的完成时间完全相同,可见笔者提出的算法能够有效地求出任务的最优调度策略. 在算法运行时间方面,笔者提出的算法 GA 的运行时间远远小于穷举算法的时间,可见笔者提出的算法不仅有效而且高效.

表 1　不同任务量情况下全局优化遗传算法(GA)和穷举算法(EA)对比的实验结果

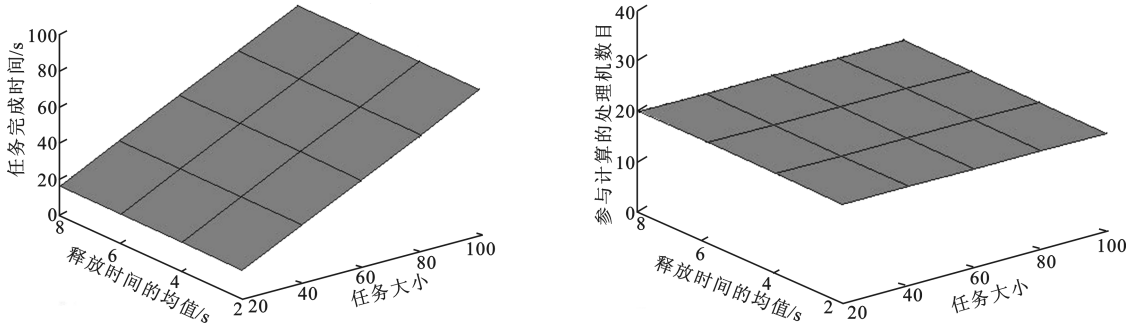
W_{total}	n		完成时间/s		算法运行时间/s		W_{total}	n		完成时间/s		算法运行时间/s	
	GA	EA	GA	EA	GA	EA		GA	EA	GA	EA	GA	EA
1.0	6	6	0.849 151	0.849 151	3.90	636.62	5.0	18	18	4.010 41	4.810 49	3.24	633.94
2.0	11	11	1.615 830	1.615 830	3.80	668.41	6.0	18	18	4.810 49	4.810 49	3.11	628.25
3.0	13	13	2.413 140	2.413 140	3.62	625.11	7.0	20	20	5.610 20	5.610 20	2.86	624.58
4.0	15	15	3.211 510	3.211 510	3.39	626.75	8.0	20	20	6.410 23	7.210 26	3.01	626.41
5.0	18	18	4.010 410	4.010 410	3.24	633.94	9.0	20	20	7.210 26	7.210 26	2.95	629.14
6.0	18	18	4.810 490	4.810 490	3.11	628.25	10.0	20	20	8.010 29	8.010 29	2.77	655.93

下面分两种情况对考虑释放时间的可分任务调度优化模型进行定性分析,着重考查任务的最短完成时间受处理机释放时间的影响.图 2 和图 3 分别表示任务较小($W_{total}=1.0\sim10.0$)和任务较大($W_{total}=20.0\sim100.0$)两种情况下,任务最短完成时间和参与计算的处理机数目随任务大小和处理机平均释放时间的变化趋势.



(a)任务最短完成时间的变化趋势 (b)参与计算的处理机数目的变化趋势

图 2 在任务较小的情况下最短完成时间和参与计算的处理机数目随任务大小和处理机平均释放时间的变化趋势



(a)任务最短完成时间的变化趋势 (b)参与计算的处理机数目的变化趋势

图 3 在任务较大的情况下最短完成时间和参与计算的处理机数目随任务大小和处理机平均释放时间的变化趋势

由图 2(a)可以看出,在任务较小的情况下,随着处理机平均释放时间和任务的逐渐增大,任务的最短完成时间也在逐渐增大.由图 2(b)可以看出,对于同样大小的任务,参与计算的处理机数目随处理机平均释放时间的增大而逐渐减少.这是由于某些处理机的释放时间过大,超过了任务的最短完成时间,因而无法参与计算.随着任务的增大,任务的最短完成时间也逐渐增大,会有更多的处理机参与任务的计算.通过上面的分析可知,在任务较小的情况下,处理机的释放时间会较大程度地影响任务的最短完成时间和参与计算的处理机数目.

由图 3 可以看出,当任务量足够大时,所有的处理机都参与计算,任务的最短完成时间随任务量的增加近似呈线性增长趋势,处理机的释放时间对任务最短完成时间的影响几乎可以忽略.这主要是由于模型采用的是阻塞通信模式,后分配任务的处理机需要等待先分配任务的处理机完成数据的传输后才能开始接收任务,当任务量足够大时,等待时间已经超过了处理机的释放时间,所以释放时间对任务的最短完成时间不再构成影响.

4 结 束 语

在考虑实际并行与分布式环境下处理机存在释放时间的基础上,笔者详细分析了 3 种不同约束条件下的任务调度过程,进而提出了一种新的混合时序约束的可分任务调度模型,并设计了高效的全局优化遗传算法对其进行求解,实验结果表明了模型的合理性和算法的有效性.

参考文献:

[1] BHARADWAJ V, GHOSE D, MANI V, et al. Scheduling Divisible Loads in Parallel and Distributed Systems [M]. Los Alamitos: IEEE Computer Society Press, 1996.

[2] BHARADWAJ V, GHOSE D, ROBERTAZZI T G. Divisible Load Theory: A New Paradigm for Load Scheduling in Distributed Systems [J]. Cluster Computing, 2003, 6(1): 7-18.

[3] ROBERTAZZI T G. Ten Reasons to Use Divisible Load Theory [J]. Computer, 2003, 36: 63-68.

[4] MANI V, GHOSE D. Distributed Computation in Linear Networks: Closed Form Solutions [J]. IEEE Transactions on Aerospace and Electronic Systems, 1994, 30: 471-483.

[5] BATAINEH S, ROBERTAZZI T G. Ultimate Performance Limits for Networks of Load Sharing Processors [C]// Proceedings of the Conference on Information Sciences and Systems. New York: Princeton University, 1992: 794-799.

[6] GHOSE D, MANI V. Distributed Computation with Communication Delays: Asymptotic Performance Analysis [J]. Journal of Parallel and Distributed Computing, 1994, 23: 293-305.

[7] BHARADWAJ V, GHOSE D, MANI V. Optimal Sequencing and Arrangement in Distributed Single-Level Networks with Communication Delays [J]. IEEE Transactions on Parallel and Distributed Systems, 1994, 5: 968-976.

[8] KIM H J, JEE G I, LEE J G. Optimal Load Distribution for Tree Network Processors [J]. IEEE Transactions on Aerospace and Electronic Systems, 1996, 32(2): 607-612.

[9] SURESH S, MANI V, OMKAR S N. The Effect of Start-up Delays in Scheduling Divisible Load on Bus Networks: an Alternate Approach [J]. Journal of Computational and Applied Mathematics, 2003, 46(10/11): 1545-1557.

[10] BHARADWAJ V, LI X L, CHUNG C K. On the Influence of Start-up Costs in Scheduling Divisible Load on Bus Networks [J]. IEEE Transactions on Parallel and Distributed Systems, 2000, 11(12): 1288-1305.

[11] SOHN J, ROBERTAZZI T G. Optimal Load Sharing for a Divisible Job on a Bus Network [J]. IEEE Transactions on Aerospace and Electronic Systems, 1996, 32(1): 34-40.

[12] SHANG M S. Optimal Algorithm for Scheduling Large Divisible Workload on Heterogeneous System [J]. Applied Mathematical Modeling, 2008, 32: 1682-1695.

[13] MURUGESAN G, CHELLAPPAN C. Multi-source Task Scheduling in Grid Computing Environment Using Linear Programming [J]. International Journal of Computer Science and Engineering, 2014, 9(1): 80-85.

[14] IYER G N, VEERAVALLI B, KRISHNAMOORTHY S G. On Handling Large-scale Polynomial Multiplication in Compute Cloud Environments using Divisible Load Paradigm [J]. IEEE Transactions on Aerospace and Electronic Systems, 2012, 48(1): 820-831.

[15] LIN W, LIANG C, WANG J Z, et al. Bandwidth-aware Divisible Task Scheduling for Cloud Computing [J]. Software: Practice and Experience, 2014, 44(2): 163-174.

[16] SHI H Y, WANG W L, KWOK N M, et al. Adaptive Indexed Divisible Load Theory for Wireless Sensor Network Workload Allocation [J]. International Journal of Distributed Sensor Networks, 2013, 2013: 484796.

[17] MAMAT A, LU Y, DEOGUN J, et al. Efficient Real-time Divisible Load Scheduling [J]. Journal of Parallel and Distributed Computing, 2012, 72(12): 1603-1616.

[18] HU M, VEERAVALLI B. Dynamic Scheduling of Hybrid Real-time Tasks on Clusters [J]. IEEE Transactions on Computers, 2013, 99: 1.

[19] HU M, VEERAVALLI B. Requirement-aware Strategies for Scheduling Real-time Divisible Loads on Clusters [J]. Journal of Parallel and Distributed Computing, 2013, 73(8): 1083-1091.

[20] CHOI K, ROBERTAZZI T G. An Exhaustive Approach to Release Time Aware Divisible Load Scheduling [J]. International Journal of Internet and Distributed Computing Systems, 2011, 1(2): 40-50.

[21] COSTA A, CAPPADONNA F A, FICHERA S. A Novel Genetic Algorithm for the Hybrid Flow Shop Scheduling with Parallel Batching and Eligibility Constraints [J]. The International Journal of Advanced Manufacturing Technology, 2014, 75(5-8): 833-847.

(编辑:郭　华)