文章编号：0253-374X(2016)10-1636-09 DOI:10.11908/j.issn.0253-374x.2016.10.023

可变结构的并行计算中任务粒度细化可扩展方法

熊焕亮[1,2,3]，曾国荪[1,3]

(1.同济大学 电子与信息工程学院，上海 200092；2.江西农业大学 软件学院，江西 南昌 330045；
3.国家高性能计算机工程技术中心同济分中心，上海 200092)

摘要：首先评估并行任务及体系结构中影响可扩展性的关键因素，并对并行任务及体系结构进行图建模.然后，提出一种DAG任务粒度细化的可扩展方法，本质上是变换图的结构、调整图节点权值和边权值.进一步推导得出一些关于新扩展方法的有用结论.最后，应用网格模拟工具SimGrid开展实验，结果表明所提出的扩展方法，能实现可变结构并行计算的等速度效率扩展，对于并行计算扩展实践有指导意义.

关键词：并行计算；机器与算法；可变结构；扩展方法
中图分类号：TP338 **文献标志码**：A

Extension by Refining Task Granularity for Parallel Computation with Variable Structures

XIONG Huanliang[1,2,3], *ZENG Guosun*[1,3]

(1. College of Electronics and Information Engineering, Tongji University, Shanghai 201804, China; 2. Software College, Jiangxi Agricultural University, Nanchang 330045, China; 3. Tongji Branch National Engineering & Technology Center of High Performance Computer, Shanghai 201804, China)

Abstract: Aiming at such extension problem in parallel computation, this paper evaluates the key factors from parallel tasks and architecture which affect the scalability, and then models parallel tasks as well as architecture by the weighted graph. Especially, we propose the extension method of refining task granularity to realize an extension in parallel computation. The extension method transforms the graph's structure and adjusts the weights of its nodes and edges in essence. Additionally, by further derivation, some significant conclusions about the new extension methods are drawn. Finally, the simulative experiments are conducted on the platform SimGrid to verify the effectiveness of the proposed extension methods. The results show that the new methods can realize iso-speed-e extension in parallel computation with variable structures, which is helpful for its practical extension.

Key words: parallel computation; machine and algorithm; variable structures; extension method

---

并行计算系统常通过扩展其规模来获得更高的计算性能.可扩展性是评价并行计算系统能否扩展的一个极其重要性能指标[1].开展可扩展性研究首要问题是如何定义和度量并行计算可扩展性.关于并行计算可扩展性度量机制的研究中，比较有代表性的成果有：等效率(Iso-efficiency)[2]机制，等平均速度(Iso-speed)[3]机制，延迟度量(Latency metric)机制[4]等.国内对并行计算可扩展性的研究也取得了一些重要研究成果：李等提出了一种等并行计算时间与通信开销之比的可扩展性模型[5]，李等提出了一种近优可扩展模型[6]，杨等提出了一种加速比增大的倍数与机器规模增大的倍数之比的可扩展性模型[7]，孙等将等平均速度度量机制扩展至异构系统，提出了等平均速度效率度量机制(Iso-speed-e scalability metric)[8].杨学军院士特别指出可扩展性度量在并行计算发展进程中的重要地位[9].

可扩展性研究的另外一个重要内容是可扩展方法的研究.最初，可扩展方法的研究聚焦于机器的扩展，主要思路是通过增加诸如CPU、内存、输入/输出接口等重要处理部件的数量，从而实现机器的扩展.对于体系结构的可扩展研究，则主要关注节点间互连结构的可扩展性，例如树形的互连结构可扩展性良好，而超立方体结构却难以扩展.并行算法的可扩展方法则主要在于可扩展性度量和试验测量两方

收稿日期：2015-10-08
基金项目：上海市优秀学科带头人计划(10XD1404400)；江西省自然科学基金(20161BAB212047，20151BAB207040)；华为创新研究计划(IRP-2013-12-03)；高效能服务器和存储技术国家重点实验室开放基金(2014HSSA10)；江西省教育厅科研项目(GJJ150426).
第一作者：熊焕亮(1977—)，男，博士生，主要研究方向为并行分布处理.E-mail：2012_xionghuanliang@tongji.edu.cn
通讯作者：曾国荪(1964—)，男，教授，博士生导师，工学博士，主要研究方向为并行计算、可信软件和信息安全.
E-mail：gszeng@tongji.edu.cn

面，此类扩展方法中较突出的有 Speedup 方法[10-11]、Iso-efficiency 方法[2]、Iso-speed 方法[3]、Latency 方法[4]、以及同济大学曾国荪教授提出的等综合性能面积方法[12].

目前，改善并行计算系统的可扩展性取得一些新成果，包括硬件可编程可重构方法、程序编译优化和重写技术等. 硬件可编程重构技术通过硬件可编程适应计算任务需求的变化，以期达到最佳性能. Tessier 等[13]全面地综述了可重构体系结构，指出可重构体系结构在计算并行任务时，对于不同的软件实现，计算硬件均能获得较好的性能和能效. 程序编译优化等软件可重构技术则是通过优化任务程序算法，变换程序算法结构，以适应硬件体系结构的变化升级，较之硬件可重构技术，更为灵活和方便. 曾国荪等[14]给出了计算任务与体系结构匹配的异构计算等速度可扩展定义及可扩展性条件. 熊焕亮等[15]针对一类结构固定的并行计算，开展了等速度效率的可扩展性研究，针对并行任务的算法结构和并行机的体系结构同构情形，提出了一种变图权的等速度效率可扩展方法，而针对两者结构异构情形，提出了一种关键路径不变的等速度效率可扩展方法.

近年建设的并行计算系统在设计和建造时大多兼顾了后续可能的扩展升级. 而对于拥有源码等所有文档资料的并行任务求解程序，不受算法结构固定的约束，完全可根据硬件体系结构的特性，优化并行任务的粒度和算法结构. 本文对此类可变结构的并行计算，提炼其可扩展关联变量，对并行任务和体系结构进行图建模，给出满足实际扩展需求及可扩展性目标的扩展方法.

1　可变结构并行计算的概念及扩展性度量

在实际应用中，诸如地球模拟，气象预报等诸多领域，不断对计算性能提出了更高的挑战性需求. 大量并行计算机系统迫切需要扩展升级，相应地，并行任务的求解算法也急需扩展优化，以匹配扩展后的计算机系统. 此类并行计算机系统的软硬件升级往往需要改变软硬件的结构.

定义 1　可变结构的并行计算：指开展并行计算时，为提高并行计算系统的计算能力，允许并行机的体系结构做合理的调整，或者为了求解更大规模的问题以及更高精度的问题解，并行任务的求解算法结构可做合理的调整.

由定义 1 可知，对于可变结构的并行计算，结构可变意味着可在原体系结构基础上增加更多的计算节点，以及相应的通信链路，也可增加并行任务的计算负载，优化任务的计算粒度，实现可变结构并行计算的扩展. 可扩展性是并行计算的重要性能指标之一，是指一个并行计算系统随着其计算节点规模的扩展，其计算性能相应增强的能力，其度量如下定义.

定义 2　可扩展性度量函数：是指并行计算扩展实践中，用于计算及度量其可扩展性程度的关系表达式. 记并行任务扩展前、后的计算负载分别为 W_1，W_2，并行机体系结构扩展前、后的标记速度分别为 C_1，C_2，则可扩展性度量函数定义为 $\Phi(C_1,C_2)=(W_1/C_1)/(W_2/C_2)$.

定义 2 给出了并行计算等速度效率可扩展性的度量函数，其中体系结构的标记速度等于其每个计算节点分别运行基准测试程序所测得的计算速度之和，速度效率等于并行任务运行于体系结构上的平均速度与体系结构标记速度的比值. 函数 $\Phi(C_1,C_2)\in(0,1]$，在理想情况下，$\Phi(C_1,C_2)=1$. 一般情况下，$\Phi(C_1,C_2)<1$.

2　可变结构并行计算的图模型

2.1　可变结构体系结构的图描述

定义 3　可变结构的体系结构：是指一类建设好的、开放的，并已成功运行的并行计算机系统，即指由若干计算节点及网络设备，通过高速网络连接而成的并行计算环境，且其计算节点可再次增加、网络结构可相应调整，可表示为一个带权图 $HSG=\langle P,U,C,B\rangle$，其中：

(1)图顶点集合 $P=\{p^i\mid i\in[1,m]\}$ 表示体系结构中计算节点集，$m=|P|$ 表示计算节点数.

(2)图边集 $U=\{u^k\mid k\in[1,s]\}$ 表示体系结构中计算节点的互连情况，边数 $s=|U|$ 表示其中通信链路数. 边 $u^k=(p^i,p^j)$，$p^i,p^j\in P$ 表示计算节点 p^i，p^j 由高速通信网络连接在一起.

(3)权值集合 $C=\{c^i\mid i\in[1,m]\}$ 表示各计算节点标记速度集，其中 c^i 为计算节点 p^i 的标记速度.

(4)权值集合 $B=\{b^k\mid k\in[1,s]\}$ 表示互连计算节点间通信链路的带宽集. $\forall u^k=(p^i,p^j)\in U$，边 u^k 的权值 b^k 表示计算节点 p^i 和 p^j 间通信链路的带宽.

定义 3 中的结构可变是指扩展时可增加新的计算节点及通信链路，由此引起计算节点之间互连结构的变化，在体系结构图模型中表现为图的拓扑结

构是可变的.

2.2 可变结构并行任务的图描述

定义4 可变结构的并行任务：是指一类开发好的,可重构的,并已成功运行过的并行程序,即为适应扩展后的体系结构,程序任务的求解流程和算法可重构,如任务的求解粒度及任务依赖关系可重新调整,可表示为一个有向无环图 PTG=〈Q,L,E,Γ,A,Δ〉,其中：

(1)顶点集 $Q=\{q^i \mid i\in[1,n]\}$表示并行任务集,图阶 $n=|Q|$表示并行任务数,且 n 可变.

(2)有向边集 $L=\{l^h \mid h\in[1,r]\}$表示关联任务间的数据依赖集, $L\subseteq Q\times Q$. $\forall l^h=\langle q^i,q^j\rangle$, l^h 表示任务 q^j 对 q^i 存在数据依赖.

(3)权值集 $E=\{\varepsilon^i \mid i\in[1,n]\}$表示并行任务计算精度集,计算精度用相对误差率表示,如 $\varepsilon^1=0.1$ 表示计算结果精确到小数点后1位,精度 ε^1 提高10倍,则 ε^1 变为 $0.1/10=0.01$.

(4)权值集 $\Gamma=\{\gamma^i \mid i\in[1,n]\}$表示并行任务问题规模集,如在矩阵相乘算法中,问题规模为矩阵维数.

(5)权值集 $A=\{\alpha^i \mid i\in[1,n]\}$表示并行任务的串行分量集.

(6)权值集 $\Delta=\{\delta^h \mid h\in[1,r]\}$表示关联任务之间传输负载集.

通常,在改变任务程序的计算精度或问题规模时,会引起任务程序计算负载的变化,不妨设并行任务 q^i 的计算负载 w^i 为其计算精度 ε^i 和问题规模 γ^i 的函数 Ψ^i,即 $w^i=\Psi^i(\varepsilon^i,\gamma^i)$. 理论上任务之间的数据传输量与相关任务的计算负载密切相关,为便于扩展性分析,将数据传输量作为独立的参数.

3 可变结构DAG任务粒度细化的扩展方法

3.1 DAG任务粒度细化扩展的前提假设

(1)假设扩展前DAG并行任务已在原有并行机上成功运行,满足既定的功能和性能要求.

对于一DAG图并行任务 $PTG_1=\langle Q_1,L_1,E_1,\Gamma_1,A_1,\Delta_1\rangle$,并行任务总数为 n_1,数据依赖关系数为 r_1. 假设任务按照某种调度策略调度在体系结构 $HSG_1=\langle P_1,U_1,C_1,B_1\rangle$上成功地执行,体系结构中计算节点总数为 m_1,通信链路总数为 s_1,则 PTG_1 的执行时间 T_1 和速度效率 Ev_1 可通过如下分析计算得到.

$$\phi_j(i)=\begin{cases}1 & q^i\in \mathrm{Path}_j\\ 0 & q^i\notin \mathrm{Path}_j\end{cases}\quad i\in[1,m] \tag{1}$$

$$\varphi_j(h)=\begin{cases}1 & l^h\in \mathrm{Path}_j\\ 0 & l^h\notin \mathrm{Path}_j\end{cases}\quad h\in[1,r] \tag{2}$$

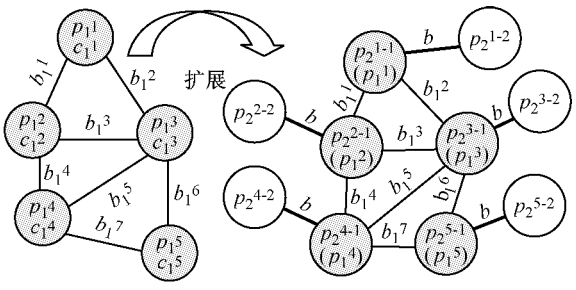
在并行任务DAG图中,从起始子任务至终止子任务存在多条执行路径,为便于描述每条路径的执行时间,分别定义任务节点隶属路径函数 $\phi_j(i)$,如式(1)所示,以及数据传输边隶属路径函数 $\phi_j(h)$,如式(2)所示. $\phi_j(i)=1$ 表示任务 q^i 在第 j 条路径 Path_j 上,$\phi_j(h)=1$ 表示数据依赖 l^h 在第 j 条路径 Path_j 上且 l^h 对应的两个任务不在同一计算节点. 记并行任务路径总数为 N_1,则整个并行任务的执行时间 T_1 为

$$T_1=\max_{j\in[1,N_1]}\Big(\sum_{i\in[1,n_1]}(\phi_j(i)w_1^i/c_1^{f(i)})+\sum_{h\in[1,s_1]}(\varphi_j(h)\delta_1^h/b_1^{g(h)})\Big) \tag{3}$$

其中 f 和 g 为并行任务在并行机上执行时采用的调度策略. 并行任务的执行时间 T_1 取决于关键路径的执行时间,关键路径执行时间可参考文献[15]中方法求解. 在并行任务的执行时间 T_1 确定后,任务执行的速度效率 Ev_1 为

$$Ev_1=W_1/(T_1C_1)=(\Sigma_{i=1}^{n_1}\Psi^i(\gamma_1^i,\varepsilon_1^i)/(T_1(\Sigma_{i=1}^{m_1}c_1^i)) \tag{4}$$

(2)并行机系统体系结构已扩展完成



a 扩展前的体系结构　　b 扩展后的体系结构

图1 体系结构的扩展

Fig.1 Extension of architecture

假设并行机的体系结构从 $HSG_1=\langle P_1,U_1,C_1,B_1\rangle$扩展为 $HSG_2=\langle P_2,U_2,C_2,B_2\rangle$,系统的标记速度从 C_1 扩展为 C_2,计算节点数从 m_1 增加至 m_2,通信链路数从 s_1 增加为 s_2,新增通信链路带宽足够大. 然后在此基础上,研究如何扩展重构并行任务以匹配新的体系结构.

3.2 DAG任务粒度细化的扩展过程

由假设(2)可知,体系结构扩展后,标记速度扩展了 $\beta=C_2/C_1$ 倍. 自然地,随体系结构性能的增强

将 DAG 并行任务中每个任务的计算负载也扩展 β 倍，以实现待求解问题更高的求解精度. 假设 DAG 任务的扩展过程中，串行计算部分保持不变，仅并行计算部分增加.

记扩展前 DAG 并行任务 PTG_1 的任务序列为 $Q_1=(q_1^1,q_1^2,\cdots,q_1^{n1})$，对应的计算负载序列为 $W_1=(w_1^1,w_1^2,\cdots,w_1^{n1})$，每个任务的串行分量序列为 $A=(\alpha^1,\alpha^2,\cdots,\alpha^{n1})$，则任务 q_1^i 扩展后的计算负载 $w_2^i=\alpha^i w_1^i+\beta(1-\alpha^i)w_1^i$，其中 $\beta(1-\alpha^i)w_1^i$ 为可并行计算负载.

在任务的可并行计算负载扩展后，需要将任务执行的粒度细化，以尽可能均衡任务的计算负载，使加速计算任务的执行成为可能. DAG 并行任务粒度细化的启发式方法如下：

(1) PTG_1 扩展后，若有 $\sum_{i=1}^{n_1}\beta(1-\alpha^i)w_1^i \ll \sum_{i=1}^{n_1}\alpha^i w_1^i$，即可并行计算负载总和远小于串行计算负载，不作任务粒度细化操作.

(2) 按每个任务的并行计算负载 $\beta(1-\alpha^i)w_1^i$ 降序排列任务系列 $(q_1^1,q_1^2,\cdots,q_1^{n1})$，对于并行计算负载相同的任务，则关键路径上的并行任务在前，从而得到并行任务序列 $(q_1^{i_1},q_1^{i_2},\cdots,q_1^{i_{n1}})$.

(3) 考察任务并行计算负载的波动情况，计算任务并行计算负载的波动因子 ρ，取 ρ 为 $(q_1^{i_1},q_1^{i_2},\cdots,q_1^{i_{n1}})$ 并行计算负载的方差，即 $\rho=(\beta(1-\alpha^{i_j})w_1^{i_j}-E(\beta(1-\alpha^{i_j})w_1^{i_j}))^2$.

(4) 计算 PTG_1 的扩展目标任务数 $n_2=\lfloor m_2 n_1/m_1\rfloor$，$PTG_1$ 中每个任务扩展的平均目标任务数 $k=\lfloor n_2/n_1\rfloor=\lfloor (m_2 n_1/m_1)/n_1\rfloor=\lfloor m_2/m_1\rfloor$，称 k 为粒度细化因子.

(5) 假设 PTG_1 中每个任务 $q_1^{i_j}$ 扩展细化为 k 个子并行任务，若任务并行计算负载的波动因子 ρ 较大($>\rho_0$)，则 $k=2\lfloor m_2/m_1\rfloor$，$j=i_1,i_2,\cdots,i_{\lfloor n1/2\rfloor}$，否则 $k=\lfloor m_2/m_1\rfloor$，$j=i_1,i_2,\cdots,i_{n1}$.

(6) 根据 PTG_1 的 DAG 图，采用算法 1 确定每个任务节点所在层的并发度.

(7) 根据任务 $q_1^{i_j}$ 所在层的并发度及已细化的任务节点计算 $q_1^{i_j}$ 当前所在层的任务数. 若任务 $q_1^{i_j}$ 扩展细化后，其所在层任务数大于 m_2，则任务 $q_1^{i_j}$ 的粒度不变，$j=j+1$，重复(7)，否则转(8).

(8) 根据已有的并行源程序及文档信息，将并行任务 $q_1^{i_j}$ 扩展细化为 $(q_2^{i_j-1},q_2^{i_j-2},\cdots,q_2^{i_j-k})$，其中每个子任务主要包括两部分，一部分由原任务 $q_1^{i_j}$ 的串行部分复制重构得到，这部分程序的功能类似，可以认为他们的执行时间相互重叠；一部分由原任务 $q_1^{i_j}$ 扩展后的并行计算部分分割得到. 根据 $(q_2^{i_j-1},q_2^{i_j-2},\cdots,q_2^{i_j-k})$ 的数据规约情况，构造子任务 $q_2^{i_j-0}$，完成数据规约等，在前面 k 个子任务执行完之后执行. 权衡通信开销 T_o 和并行执行所节省的计算时间 T_s，若 $T_o>T_s$，则任务 $q_2^{i_j}$ 放弃粒度细化. $j=j+1$，转(9).

(9) 若 PTG_1 中所有任务节点均处理完毕，则 DAG 任务细化结束，否则转(7).

假设经过分析，DAG 并行任务中每个任务均实施扩展细化，则其扩展后的 DAG 并行任务如图 2b 所示，每个任务均扩展细化为 3 个子任务，即 2 个可并行计算的子任务，1 个规约任务，任务之间的依赖为粒度细化而产生的额外通信，例如任务 q_1^1，扩展为 $(q_2^{1-1},q_2^{1-2},q_2^{1-0})$ 3 个子任务，其中 q_2^{1-1},q_2^{1-2} 为可并行计算的子任务，q_2^{1-0} 为规约子任务.

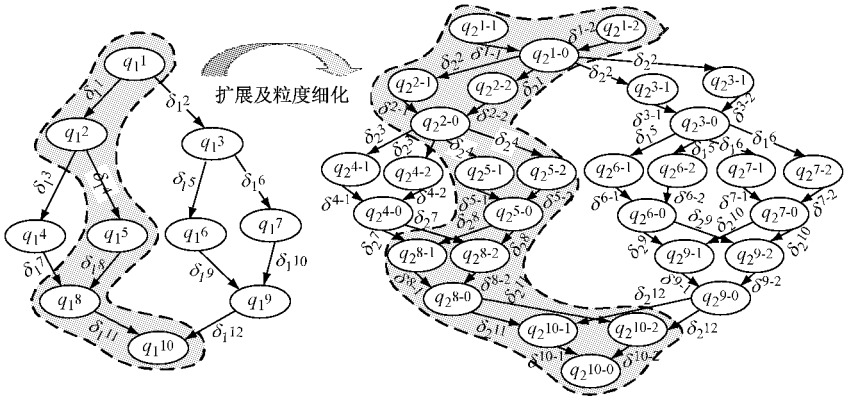
算法 1. ComputeDOPForLevel//计算 DAG 并行任务中每个任务所在层及该层并发度.

输入：初始的并行任务 DAG 图(以邻接表存储)

输出：DAG 并行任务中每个任务所在层 nodeLevel[]及每层的节点数 levelCount[]

```
{
  初始化任务队列 Queue，节点所在层号数组 nodeLevel[]，每层的计算节点数 levelCount[]全为 0;
  初始化变量 currentLevel=0，totalNodeNum=0;
  将 DAG 图中所有入度为 0 的任务节点入队;
  while(Queue 不为空){
  GetQueuelength;//取当前队列的长度
    totalNodeNum + = Queuelength;//计算 DAG 并行任务总任务数.
    for(int i=0;i<Queuelength;i++){
    队头节点 head 出队;
      nodeLevel [ head. nodeNumber ] = currentLevel;//标记节点所在层号.
    head 指向的全部节点入队;//head 节点的所有邻接节点入队.
    }
    currentLevel++;
  }
```

```
for(int i=0;i<totalNodeNum;j++){                    }
    levelCount[nodeLevel[i]]++;//计算每层          }
的节点数.                                          }
```



a 扩展前的并行任务　　b 扩展后的并行任务

图 2　DAG 并行任务的扩展及其粒度细化

Fig.2　Extension of DAG parallel task and its grain size refinement

4　可变结构并行计算 DAG 任务粒度细化扩展方法的性能评估

4.1　DAG 任务扩展及粒度细化后并行计算性能分析

4.1.1　可变结构并行计算扩展参数的详细假定

(1)并行机体系结构扩展参数的假定

假设并行计算机的体系结构已按假设(2)扩展好,为便于分析,假设扩展后的计算节点数 m_2 为扩展前节点数 m_1 的整数倍,即 $\lfloor m_2/m_1 \rfloor = m_2/m_1$,新增计算节点如图 1b 所示均匀地连接在原计算节点上,每个新增计算节点性能不低于与之直连的原计算节点,例如 $c_2^{1\text{-}2} \geqslant c_1^{1}$,新增通信链路带宽足够大,数据规约通信时间可忽略不计.

(2)DAG 并行任务扩展细化参数的假定

根据假设(1),扩展前 DAG 并行任务已成功调度至所匹配的计算节点.不妨假设任务 DAG 图的 N_1 条执行路径中,第 j 条路径为关键路径,则由式(3)可得 DAG 并行任务的执行时间 T_1 为

$$T_1 = \sum_{i\in[1,n_1]} (\phi_j(i) w_1^i / c_1^{f(i)}) + \sum_{h\in[1,s_1]} (\varphi_j(h) \delta_1^h / b_1^{g(h)}) \tag{5}$$

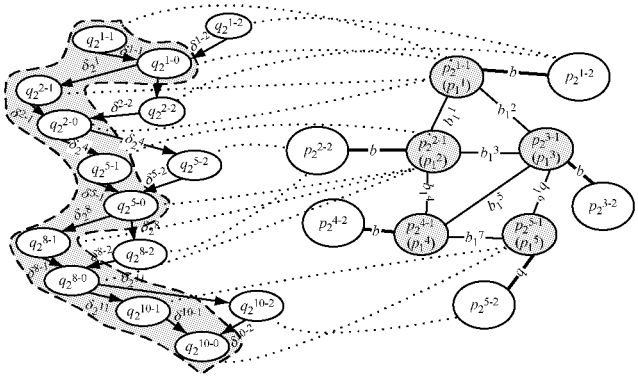
随着体系结构标记速度由 C_1 扩展提高为 C_2,DAG 并行任务中每个任务的并行计算负载相应地扩展 $\beta = C_2/C_1$ 倍,任务 q_1^i 扩展后的计算负载由原串行部分和扩展后的并行部分组成,即 $w_2^i = \alpha^i w_1^i + \beta(1-\alpha^i) w_1^i$.

为便于分析,假设 DAG 并行任务中所有任务均细化为 $k+1$ 个子任务,$k = m_2/m_1$.任务 q_1^i 在扩展并实施粒度细化后,生成 k 个子并行任务($q_2^{i-1}, q_2^{i-2}, \cdots, q_2^{i-k}$),每个子任务的计算负载为扩展前任务 q_1^i 负载 w_1^i 的 η 倍,$\eta = \alpha + \beta(1-\alpha)/k$,即 $w_2^{i-1} = w_2^{i-2} = \cdots = w_2^{i-k} = \eta w_1^i$,以及 1 个规约子任务 q_2^{i-0}.

对于数据依赖关系 $l_1^h = \langle q_1^i, q_1^j \rangle$,不妨设 DAG 并行任务扩展后,传输给每个后继子任务的通信负载为扩展前通信负载 δ_1^h 的 η 倍,即为 $\eta\delta_1^h$.

(3)扩展及粒度细化后的 DAG 并行任务与新体系结构的匹配映射

扩展前,任务 q_1^i 映射至匹配的计算节点 $p_1^{f(i)}$,不妨设 q_1^i 扩展及粒度细化生成的子任务 q_2^{i-1} 及 q_2^{i-0} 映射至计算节点 $p_1^{f(i)}$),生成的其它子任务($q_2^{i-2}, \cdots, q_2^{i-k}$)映射至新增计算节点($p_2^{f(i)-2}, \cdots, p_2^{f(i)-k}$),如图 3 所示.



a 关键路径任务粒度细化　　b 扩展后的体系结构

图 3　原关键路径任务扩展及粒度细化后与匹配的体系结构

Fig.3　Task belonged to the original critical path after the extension and the matched architecture

扩展前，任务之间的数据通信 l_1^h 映射至链路 $u_1^{g(h)}$，扩展后，依赖数据在传输给新增计算节点时，需经原通信链路 $u_1^{g(h)}$ 传输至原计算节点，再经新增通信链路传输至新增计算节点，如图 3 所示.

4.1.2 可变结构并行计算 DAG 并行任务关键路径的扩展分析

根据假定(3)，子任务 q_2^{i-1} 与 q_2^{i-0} 在计算节点 $p_1^{f(i)}$ 上执行，由于与 $p_1^{f(i)}$ 直连的新增计算节点标记速度不低于 $p_1^{f(i)}$ 的标记速度，故 q_2^{i-1} 与 q_2^{i-0} 在 $p_1^{f(i)}$ 上的计算时间不小于其它子并行任务在新增节点上的计算时间，而 q_2^{i-0} 的计算时间可理解为子并行任务 $(q_2^{i-1},\cdots,q_2^{i-k})$ 之间的数据规约等产生的时间开销 T_o^i，$T_o^i=\max(\delta^{i-1}/b,\delta^{i-2}/b,\cdots,\delta^{i-k}/b)$，因此任务 q_1^i 扩展及粒度细化后，在匹配的计算节点上的执行时间为 $\eta w_1^j/c_1^{f(i)}+T_o^i$，其中 $\eta=\alpha+\beta(1-\alpha)/k$. 数据通信 l_1^h 的传输负载扩展 η 倍后，数据传输时间可近似为 $\eta\delta_1^h/b_1^{g(h)}+\eta\delta_1^h/b$.

命题 1　可变结构并行计算中，DAG 并行任务 $\mathrm{PTG}_1=\langle Q_1,L_1,E_1,\Gamma_1,A_1,\Delta_1\rangle$ 在匹配的体系结构 $\mathrm{HSG}_1<P_1,U_1,C_1,B_1\rangle$ 上执行的关键路径为 j，如果 HSG_1 按照假设(1)已扩展为 HSG_2，PTG_1 中每个任务的串行分量均为 α，并按照假设(2)实施粒度细化扩展为 PTG_2，且 PTG_2 和 HSG_2 已按照假设(3)匹配映射好，则 PTG_2 在 HSG_2 上执行的关键路径由原关键路径 j 上任务粒度细化生成的子并行任务组成.

证明：DAG 并行任务 PTG_1 存在多条执行路径，对于其中任一路径 r，记路径 r 的执行时间为 T_1^r，则 $T_1^r=\sum_{i\in[1,n_1]}(\phi_r(i)w_1^i/c_1^{f(i)})+\sum_{h\in[1,s_1]}(\varphi_r(h)\delta_1^h/b_1^{g(h)})$. 根据假设及上述分析可知，在由原路径 r 细化得到的多条路径中，记执行时间最长路径的执行时间为 T_2^r，则 $T_2^r=\sum_{i\in[1,n_1]}(\phi_r(i)(\eta w_1^i/c_1^{f(i)}+T_o^i))+\sum_{h\in[1,s_1]}(\varphi_r(h)\eta(\delta_1^h/b_1^{g(h)}+\delta_1^h/b))$，$T_o^i=\max(\delta^{i-1}/b,\delta^{i-2}/b,\cdots,\delta^{i-k}/b)$. 显然，$\mathrm{PTG}_1$ 的关键路径 j 扩展细化后的执行时间 $T_2^j=\sum_{i\in[1,n_1]}(\phi_j(i)(\eta w_1^i/c_1^{f(i)}+T_o^i))+\sum_{h\in[1,s_1]}(\varphi_j(h)\eta(\delta_1^h/b_1^{g(h)}+\delta_1^h/b))$. 比较 T_2^j 与 T_2^r 的表达式，对于 T_o^i 项，当新增链路带宽足够大时，任务的规约时间 T_o^i 及数据在新增链路上的传输时间 $\eta\delta_1^h/b$ 很小，不同任务之间的 T_o^i 差和 $\eta\delta_1^h/b$ 差则更小，故可忽略不计，$T_2^r\approx\sum_{i\in[1,n_1]}(\phi_r(i)\eta w_1^i/c_1^{f(i)})+\sum_{h\in[1,s_1]}(\varphi_r(h)\eta\delta_1^h/b_1^{g(h)})=\eta T_1^r$，$T_2^j\approx\eta T_1^j$，而 $T_1^j\geqslant T_1^r$，故 $T_2^j\geqslant T_2^r$，也即 PTG_2 在 HSG_2 上执行的关键路径由 PTG_1 关键路径 j 上任务粒度细化生成的子并行任务组成，证毕.

根据命题 1，可变结构并行计算中，体系结构 HSG_1 扩展为 HSG_2，DAG 并行任务 PTG_1 扩展为 PTG_2 后，PTG_2 的执行时间为 T_2，即

$$\begin{cases}\eta=\alpha+\beta(1-\alpha)/k\\ T_o^i=\max(\delta^{i-1}/b,\delta^{i-2}/b,\cdots,\delta^{i-k}/b)\\ T_2=\sum_{i\in[1,n_1]}(\phi_j(i)\eta w_1^i/c_1^{f(i)}+T_o^i)+\\ \qquad\sum_{h\in[1,s_1]}(\varphi_j(h)(\eta\delta_1^h/b_1^{g(h)}+\eta\delta_1^h/b))\end{cases}\tag{6}$$

DAG 并行任务 PTG_1 中每个任务 q_1^i，$(i=1,2,\cdots,n_1)$ 细化为 $k+1$ 个子并行任务 $(q_2^{i-0},q_2^{i-1},\cdots,q_2^{i-k})$，其中 q_2^{i-0} 主要是实现子并行任务 $(q_2^{i-1},\cdots,q_2^{i-k})$ 之间的数据规约，计算负载忽略不计，而 $(q_2^{i-1},\cdots,q_2^{i-k})$ 中每个子任务计算负载均为 ηw_1^i，故 $(q_2^{i-1},\cdots,q_2^{i-k})$ 的计算负载和为 $k\eta w_1^i$，因此 DAG 并行任务 PTG_2 的计算负载为 $W_2=\Sigma_{i=1}^{n_1}k\eta w_1^i$.

并行计算的速度效率近似为

$$Ev_2=W_2/(T_2C_2)=\Sigma_{i=1}^{n_1}(k\eta w_1^i)/((\eta T_1)\Sigma_{i=1}^{m_2}c_2^i)=kW_1/(T_1(C_1+\Sigma_{i=m_1+1}^{m_2}c_2^i))\tag{7}$$

其中，$\Sigma_{i=m_1+1}^{m_2}c_2^i$ 为 HSG_2 中新增计算节点的标记速度之和.

命题 2　可变结构并行计算中，DAG 并行任务 $\mathrm{PTG}_1=\langle Q_1,L_1,E_1,\Gamma_1,A_1,\Delta_1\rangle$ 在匹配的体系结构 $\mathrm{HSG}_1\langle P_1,U_1,C_1,B_1\rangle$ 上执行的关键路径为 j，如果 HSG_1 按照假设(1)已扩展为 HSG_2，PTG_1 中每个任务的串行分量均为 α，并按照假设(2)实施粒度细化扩展为 PTG_2，且 PTG_2 和 HSG_2 已按照假设(3)匹配映射好，则 PTG_2 在 HSG_2 上执行的速度效率 Ev_2 小于等于 Ev_1.

证明：根据命题 2 的扩展假设及公式(7)，易知 PTG_2 在 HSG_2 上的速度效率 $Ev_2=kW_1/(T_1(C_1+\Sigma_{i=m_1+1}^{m_2}c_2^i))$，当新增计算节点的标记速度均大于或等于原计算节点的标记速度时，$\Sigma_{i=m_1+1}^{m_2}c_2^i=\Sigma_{i=m_1+1}^{km_1}c_2^i\geqslant(k-1)\Sigma_{i=1}^{m_1}c_1^i=(k-1)C_1$，由此 $Ev_2=kW_1/(T_1(C_1+\Sigma_{i=m_1+1}^{m_2}c_2^i))\leqslant kW_1/(T_1kC_1)=Ev_1$，即 $Ev_2\leqslant Ev_1$，证毕.

由命题 2 可知，可变结构的并行计算实施任务

粒度细化扩展时,速度效率保持近似相等.根据定义2,等速度效率可扩展性函数 $\Phi=(W_1/C_1)/(W_2/C_2)=(C_2/C_1)/(W_2/W_1)=\beta/(k\alpha+\beta(1-\alpha)+\chi)=1/(\alpha k/\beta+(1-\alpha)+\chi/\beta)$,其中 χ 为规约子任务计算负载比重.若忽略规约子任务计算负载,即 $\chi=0$,且 α 一定时,显然有

$$\Phi(k,\beta)=\frac{1}{\frac{\alpha k}{\beta}+(1-\alpha)}\begin{cases}\leqslant 1, & \text{if } k\geqslant\beta\\ >1, & \text{if } k<\beta\end{cases} \tag{8}$$

当 $k<\beta$ 时,即总计算负载的增长较系统标记速度的增长更慢,故有 $\Phi>1$,反之,当 $k\geqslant\beta$ 时,总计算负载的增长较体系结构标记速度的增长更快,故有 $\Phi\leqslant 1$.

4.2 可变结构并行计算扩展的仿真实验

4.2.1 仿真实验工具及实验设计

为验证所提出可扩展方法的有效性,在实验室的Dell PC集群上,利用SimGrid3.10开展扩展模拟实验.集群平台包括32个计算节点,由千兆以太网互连组成,每个节点的CPU为Intel(R) Xeon™ 3.0 GHz,缓存4 GB,内存32 GB.操作系统为Redhat Linux9.0,并行计算软件环境为MPICH2.

可变结构的并行计算仿真实验中,扩展前的并行任务 PTG_1 如图2a所示,体系结构 HSG_1 如图1a所示,HSG_1 及 PTG_1 的具体参数取值如表1和表2所示,且实验中任务的计算负载与问题规模及计算精度的函数关系假设为 $w^i=10^5\gamma^i+10^6/\varepsilon^i$.表2中参数 $\alpha=1\%$ 为任务 PTG_1 的串行分量,任务 PTG_1 的计算规模和计算精度提升,任务计算负载相应地扩展 $\beta=C_2/C_1=3$ 倍,$k=m_2/m_1=2$ 为任务 PTG_1 的粒度细化因子,任务 PTG_1 粒度细化后,通信负载的扩展倍数 $\eta=\alpha+\beta(1-\alpha)/k=1.495$.

表1 可变结构的体系结构扩展前后的性能参数(带下划线项为关键路径对应的数据项)

Tab.1 Performance parameters before and after the extension of architecture with the variable structure (the underlined data items corresponding to the key path)

扩展前($m_1=5$,$s_1=7$,$C_1=0.6$GIPS)			扩展及粒度细化后($m_2=10$,$s_2=12$,$C_2=1.235$GIPS)		
编号	计算节点 c_1^i/MIPS	通信带宽 b_1^i/Gbps	编号	计算节点 c_2^i/MIPS	通信带宽 b_2^i/Gbps
1	100	0.15	1—1	100	0.15
			1—2	110	10
2	120	0.25	2—1	120	0.25
			2—2	130	10
3	110	0.45	3—1	110	0.45
			3—2	135	10
4	115	0.35	4—1	115	0.35
			4—2	125	10
5	125	0.30	5—1	125	0.30
			5—2	135	10
6	—	0.1	6—1	—	0.1
7	—	0.2	7—1	—	0.2

实验中,并行任务 q^i 调度至计算节点 $p^{f(i)}$,表示为二元组$(q^i,p^{f(i)})$,DAG并行任务 PTG_1 与体系结构 HSG_1 中计算节点的映射关系具体为:(q_1^1,p_1^1),(q_1^2,p_1^1),(q_1^3,p_1^3),(q_1^4,p_1^1),(q_1^5,p_1^2),(q_1^6,p_1^4),(q_1^7,p_1^3),(q_1^8,p_1^2),(q_1^9,p_1^5),(q_1^{10},p_1^5).在体系结构 HSG_1 扩展为 HSG_2,并行任务 PTG_1 扩展细化为 PTG_2 后,各并行子任务与计算节点的映射参照扩展之前的匹配映射,例如,任务 q_1^1 细化为 $(q_2^{1-1},q_2^{1-2},q_2^{1-0})$,其中 q_2^{1-1} 和 q_2^{1-0} 映射至计算节点 p_2^{1-1}(即原计算节点 p_1^1),q_2^{1-2} 映射至新增计算节点 p_2^{1-2},其它子任务与计算节点的映射类似.

4.2.2 仿真实验结果及分析

对于可变结构的并行计算,在体系结构已扩展升级好之后,并行任务采用任务粒度细化的扩展方法进行模拟扩展,在SimGrid3.1上运行模拟程序Simulator,实验结果如表3所示.

表3中扩展后的速度效率0.361近似于扩展前的速度效率0.373,表明实验中任务粒度细化扩展方法基本上实现了等速度效率扩展,扩展后的速度效率略低于扩展前的速度效率,主要是由于细化后的子任务之间规约开销,从而影响了实际运行速度.另外,实验中 $k=2$,$\beta=3.0$,根据公式(8),有 $\Phi>1$,但实际上规约子任务计算负载总是客观存在,实验中取 $\chi=0.14$,$\chi W_1=390$GI,从而有 $\Phi=0.959$.

为揭示不同的(k,β)对计算密集型、通信密集型及计算通信平衡型三类并行任务速度效率及可扩展性的影响,我们以表1中所描述的体系结构,表2所描述的并行任务算法结构为基准,以高通信负载比重模拟通信密集型任务,变换 k 和 β,实施多次模拟扩展实验.扩展实验的速度效率变化曲线如图4所示,可扩展性变化曲线如图5所示.图4中速度效率的变化曲线表明,在任务的结构和计算负载,扩展参数(k,β)均相同时,通信密集型任务因通信延迟开销较大,速度效率较计算密集型任务的速度效率明显偏低,而计算通信平衡型任务的速度效率则介于两者之间.

根据式(8),当上述两类任务的串行比 α,以及参数 k,β 相同时,其可扩展性理应相同,但图5中扩展性曲线表明,任务类型为通信密集型的并行计算,其

表 2　可变结构的并行任务扩展前后的性能参数(带下划线项为关键路径包含的数据项)

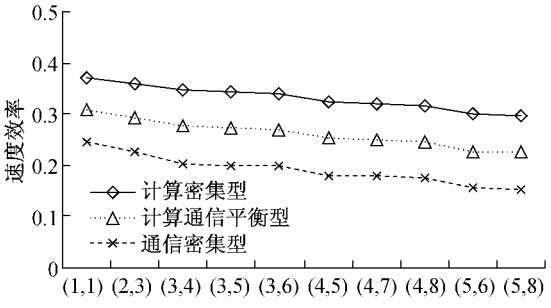
Tab.2　Performance paramenters before and after the extension of parallel task with the variable structure (the underlined data items corresponding to the key path)

扩展前($n_1=10, r_1=12, \alpha=1\%$)					扩展及粒度细化后($k=2, \beta=3, \eta=1.495$)				
编号	并行任务 $q_1{}^i$			数据传输量 $\delta_1{}^i$/MB	编号	并行任务 $q_2{}^i$			数据传输量 $\delta_2{}^i$/MB
	$\gamma_1{}^i$	$\varepsilon_1{}^i$	$w_i{}^i$/GI			$\gamma_2{}^i$	$\varepsilon_2{}^i$	$w_2{}^i$/GI	
1	1×10^6	1×10^{-5}	2×10^2	1	1—1	3.00×10^6	3.33×10^{-6}	299	1.50
					1—2	3.00×10^6	3.33×10^{-6}	299	1.50
2	1×10^7	1×10^{-6}	2×10^3	3	2—1	3.00×10^7	3.33×10^{-7}	2 990	4.49
					2—2	3.00×10^7	3.33×10^{-7}	2 990	4.49
3	1×10^6	1×10^{-5}	2×10^2	2	3—1	3.00×10^6	3.33×10^{-6}	299	2.99
					3—2	3.00×10^6	3.33×10^{-6}	299	2.99
4	1×10^6	1×10^{-3}	101	3.5	4—1	3.00×10^6	3.33×10^{-4}	150.99	5.23
					4—2	3.00×10^6	3.33×10^{-4}	150.99	5.23
5	1×10^5	1×10^{-4}	20	4	5—1	3.00×10^5	3.33×10^{-5}	29.9	5.98
					5—2	3.00×10^5	3.33×10^{-5}	29.9	5.98
6	1×10^5	1×10^{-5}	110	3	6—1	3.00×10^5	3.33×10^{-6}	164.45	4.49
					6—2	3.00×10^5	3.33×10^{-6}	164.45	4.49
7	1×10^6	1×10^{-5}	2×10^2	3.2	7—1	3.00×10^6	3.33×10^{-6}	299	4.78
					7—2	3.00×10^6	3.33×10^{-6}	299	4.78
8	1×10^5	1×10^{-4}	20	1.5	8—1	3.00×10^5	3.33×10^{-5}	29.9	2.24
					8—2	3.00×10^5	3.33×10^{-5}	29.9	2.24
9	1×10^5	1×10^{-3}	11	2.1	9—1	3.00×10^5	3.33×10^{-4}	16.44	3.14
					9—2	3.00×10^5	3.33×10^{-4}	16.44	3.14
10	1×10^3	1×10^{-2}	4	0.9	10—1	3.00×10^3	3.33×10^{-3}	5.98	1.35
					10—2	3.00×10^3	3.33×10^{-3}	5.98	1.35
11	—	—	—	0.8	11—1	—	—	—	1.20
					11—2	—	—	—	1.20
12	—	—	—	0.6	12—1	—	—	—	0.90
					12—2	—	—	—	0.90

表 3　可变结构并行计算任务粒度细化扩展实验结果

Tab.3　The experimental result of grain size refinement extension for parallel task with the variable structure

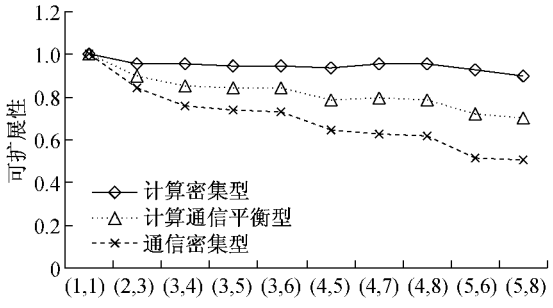
	总计算负载 W/GI	标记速度 C/MIPS	并行执行时间/s	实际运行速度 V/MIPS	速度效率 $Ev=V/C$	可扩展程度 $\Phi=(W_1/C_1)/(W_2/C_2)$
扩展前	2 866	570	13 805.12	212.59	0.373	0.959
扩展后	8 569+390	1 710	14 865.02	618.94	0.361	



(k,β)，k：任务粒度细化因子；β：计算负载扩展因子

图 4　速度效率变化趋势

Fig.4　The trend of speed efficiency



(k,β)，k：任务粒度细化因子；β：计算负载扩展因子

图 5　可扩展性变化趋势

Fig.5　The trend of scalability

可扩展性较计算密集型更小，计算通信平衡型任务的可扩展性介于两者之间，这是由于通信密集型任务计算中通信负载比重较大，任务粒度细化生成的规约任务的计算负载明显增加，致使其可扩展性降低. 另外，图 5 中计算密集型任务与通信密集型任务相比较，其可扩展性基本保持不变，这是因为计算密集型任务粒度细化扩展方法中，产生的额外通信、规约等开销极少，可扩展性主要取决于参数 k 和 β.

5 结束语

对于一些成熟的、已成功运行的并行计算应用系统,为获得更高的计算性能,通常会扩展计算节点规模,增大任务计算负载,优化任务算法结构.针对此类可变结构并行计算的可扩展问题,本文分析影响其可扩展性的并行任务因素及体系结构因素,对并行任务及体系结构进行图建模.特别针对体系结构业已扩展升级好,如何增大并行任务的计算负载,优化任务算法结构的扩展问题,提出了DAG任务粒度细化的可扩展方法,实质上为变换图结构的可扩展方法.通过分析扩展前后的速度效率性能,证明了DAG任务粒度细化的扩展方法可基本实现可变结构并行计算的等速度效率扩展.最后,为验证所提出可扩展方法的有效性,编写了基于网格计算模拟工具SimGrid3.10的C语言程序Simulator,模拟任务在体系结构上的运行,从而实现模拟扩展实验,结果表明所提出的扩展方法实现了可变结构并行计算的等速度效率扩展.

参考文献:

[1] 李晓梅,莫则尧,胡庆丰,等.可扩展并行算法的设计和分析[M].长沙:国防工业出版社,2000.
LI Xiaomei, MAO Zeyao, HU Qingfeng, *et al*. Design & analysis of scalable parallel algorithms [M]. Changsha: National Defense Industry Press, 2000.

[2] Grama A, Gupta A, Kumar V. Iso-efficiency: measuring the scalability of parallel algorithms and architectures[J]. IEEE Parallel & Distributed Technology, 1993, 1(3): 12.

[3] Sun X H, Rover D T. Scalability of parallel algorithm machine combinations[J]. IEEE Transactions on Parallel Distributed Systems, 1994, 5(6):599.

[4] Zhang X D, Yan Y, He K. Latency metric: an experimental method for measuring and evaluating parallel program and architecture scalability[J]. Journal of Parallel and Distributed Computing, 1994, 22(3): 392.

[5] Wu X F, Li W. Performance models for scalable cluster computing[J]. Journal of Systems Architecture, 1997, 44(3): 189.

[6] 陈军.李晓梅.近优可扩展性:一种实用的可扩展性度量[J].计算机学报,2001,24(2):179.
CHEN Jun, LI Xiaomei. Near-optimal scalability: a practical scalability metric[J]. Chinese Journal of Computers, 2001, 24(2):179.

[7] 王与力,杨晓东.一种更有效的并行系统可扩展性模型[J].计算机学报,2001,24(1):84.
WANG Yuli, YANG Xiaodong. A more effective scalability model for parallel system[J]. Chinese Journal of Computers, 2001, 24(1): 84.

[8] Chen Y, Sun X H, Wu M. Algorithm-system scalability of heterogeneous computing [J]. Journal of Parallel and Distributed Computing, 2008, 68(11): 1403.

[9] Yang X J, Wang Z Y, Xue J L. The reliability wall for exascale supercomputing[J]. IEEE Transactions on Computers, 2012, 61(6):767.

[10] Sun X H, Ni L M. Scalable problems and memory-bounded speedup[J]. Journal of Parallel and Distributed Computing, 1993, 19(1): 27.

[11] Yang X J, Du J, Wang Z Y. An effective speedup metric for measuring productivity in large-scale parallel computer systems [J]. Journal of Supercomputing, 2011, 56(2):164.

[12] Xiong H L, Zeng G S, Zeng Y. A novel scalability metric about iso-area of performance for parallel computing[J]. Journal of Supercomputing, 2014, 68(2):652.

[13] Tessier R, Pocek K, DeHon A. Reconfigurable computing architectures[J]. Proceedings of the IEEE, 2015, 103(3): 332.

[14] 郝水侠,曾国荪,谭一鸣.计算任务与体系结构匹配的异构计算可扩展性分析[J].电子学报,2010,38(11):2585.
HAO Shuixia, ZENG Guosun, TAN Yiming. Scalability analysis of heterogeneous computing based on computation task and architecture to match[J]. *Acta Electronica Sinica*, 2010, 38(11): 2585.

[15] Xiong H L, Zeng G S, Ding C L, *et al*. Extensions by graph weight for parallel computing under the constraint of fixed structure[J]. IETE Journal of Research, 2015:doi:10.1080/03772063.2015.1093967, 2015.