多核集群任务分配问题复杂性分析

谭国真,杨际祥,王 凡,潘 东
(大连理工大学计算机科学与技术学院,辽宁大连 116024)

摘 要: 传统任务分配问题通常以最小化计算代价和节点间通信代价的总代价为研究目标.在多核集群系统中,需要同时考虑节点内冲突代价.本文研究了以最小化计算代价、节点间通信代价和节点内冲突代价的总代价为目标的多核集群任务分配问题.通过建立任务分配问题与最小费用流问题的等价关系来分析节点内冲突代价对问题复杂性的影响关系.结果表明冲突代价成为影响问题复杂性的一个重要因素,给出并证明了冲突代价和节点间通信代价对问题复杂性的影响关系.最后,进一步讨论了各种复杂性下的多核集群任务分配问题的解法以及本文定理与结论的可应用性与有效性.

关键词: 任务分配; 复杂度分析; 最小费用流; 冲突代价; 多核集群
中图分类号: TP301 **文献标识码:** A **文章编号:** 0372-2112 (2012) 02-0241-06
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2012.02.006

Complexity Analysis of Task Assignment Problem on Multi-Core Clusters

TAN Guo-zhen, YANG Ji-xiang, WANG Fan, PAN Dong
(*School of Computer Science and Technology, Dalian University of Technology. Dalian, Liaoning* 116024, *China*)

Abstract: Traditional TAP (Task Assignment Problem) is generally to minimize total execution cost and inter-node communication cost. This paper investigates New TAP (NTAP) considering additive conflict cost in emerging multi-core cluster systems. We analyze the complexity of the NTAP with network flow method and conclude that the conflict cost is a key to the complexity of the NTAP, and demonstrate the relationships between the complexity and the two costs including conflict cost and inter-node communication cost. Moreover, the solutions to the NTAP, and the applicability and effectiveness of the theorems and conclusions are also discussed.

Key words: task assignment; complexity analysis; minimum cost flow; conflict cost; multi-core clusters

1 引言

并行与分布式系统中的任务分配问题是将一组任务分配到一组处理器或计算节点上以最小化计算代价和计算节点间的通信代价的总代价[1~6].并行计算性能的增长在未来至少 5 年的发展时间里将肯定来自于使用多核处理器技术的发展,系统趋向于由多核组成,多核集群以层次方式构成,基于多核节点的以层次方式构成的集群成为未来并行计算的一种发展趋势[7~10].传统的任务分配问题没有考虑多核集群的层次结构特性,适用于一般的分布式系统,不适用于多核集群,这是本文之所以针对多核集群系统进行研究的根本原因.

一方面,多核节点性能的获得主要来自于线程级并行的发展,而多线程对多核处理器和存储器等共享资源的访问将加剧节点内的访问冲突,为解决访问冲突而引入的同步操作开销或代价成为并行计算性能的一个瓶颈问题[7].另一方面,随着节点内处理器核的速度和数量的提高,更多的核访问存储系统将导致核之间对共享资源的争用加剧,同步代价增大.此外,节点内通信代价与节点内的通信量和通信延迟两个因素密切相关,文献[8,9]在通信量和通信延迟方面展示了节点内通信的重要性.在通信量方面,文献[8]指出,NAS 基准测试中超过 50%的消息是通过节点内通信完成的.在通信延迟方面,文献[9]表明,节点内通信延迟与节点间通信延迟之比随消息增大而增大.在本文中,为便于描述,把节点内通信代价以及为解决节点内的访问冲突而引入的同步代价统称为节点内冲突代价(简称冲突代价).因此,随着多核集群节点中核数和任务数的增多,冲突代价有增大的趋势,成为并行与分布式计算中的任务分配问题需要考虑的一个关键问题.

任务分配问题是并行与分布式计算研究领域的经典问题,一般的任务分配问题是 NP 完全问题[2].在其发展完善的过程中,研究者们针对该问题给出了大量的分析与求解方法,包括图论方法、数学规划方法、启发式

收稿日期:2010-12-15;修回日期:2011-05-20
基金项目:国家自然科学基金(No.60873256);中央高校基本科研业务费专项资金(No.DUT11SX09)

方法和元启发式方法[6].图论方法是将任务和处理器用图示的形式加以表达,并利用图论的方法建立任务和处理器之间的映射关系,产生有效的任务分配方案,与本文的研究内容密切相关,而网络流方法则是其中一种重要方法.网络流方法是将任务和处理器作为图中结点,根据任务模块之间的依赖关系、处理器之间的通信以及任务在处理器上的执行开销建立网络流模型,利用最大流-最小割算法进行求解.Stone[1]研究了考虑计算代价和通信代价的任务分配问题,应用最大流-最小割算法给出了一个双处理器系统上的最优任务分配方案.但是,该工作并没有解决多处理器的任务分配问题.Bokhari[2]对其进行了扩展,用于解决任务通信图为树形结构的任务分配问题.Towsley[4]又进一步把Bokhari的工作推广到了串并联结构.Bokhari[11]扩展了Stone的静态模型,研究了双处理器的动态任务分配模型,并提出了高效的最优化算法.Yadav等[12]将该模型推广到了多处理器系统.

Zaharia等[13]为解决集群调度数据局部性和公平性之间的冲突,提出了延迟调度算法.Lin等[14]研究了集群环境中的实时调度问题,以最小化任务执行时间和系统资源.Stillwell等[15]提出一个动态部分资源调度方法,以优化一个运行时已定义的性能指标.Casanova等[16]研究了同构单集群上的多并行任务图的离线调度问题,目标是优化性能,同时考虑任务图之间的公平性.Yu等[17]提出了一种故障感知工作流调度算法,通过降低平均的最大完成时间获得了较优的性能.

多核集群任务分配问题中,总代价为计算代价、节点间通信代价和节点内冲突代价之和,前人的研究文献中并未涉及到节点内冲突代价.本文的创新之处在于综合考虑三方面的代价总和来解决任务分配问题,提出了多核集群系统中的任务分配问题,目标是最小化计算代价、节点间通信代价和节点内冲突代价的总代价.

然而,如何分析该问题的复杂性以及新引入的冲突代价对问题复杂性的影响关系是什么?这是一个重要的理论问题,针对这一理论问题,本文开展了相关的研究工作.通过建立考虑冲突代价的多核集群任务分配问题与网络流中的最小费用流问题的等价关系来分析和证明了冲突代价与节点间通信代价对问题复杂性的影响关系,并进一步地讨论了各种复杂性下的多核集群任务分配问题的解法以及本文定理和结论的可应用性与有效性.

2 基本定义和假设

不失一般性,定义任务集合为 $T=\{t_1,t_2,\cdots,t_n\}$,计算节点集合为 $P=\{p_1,p_2,\cdots,p_m\}$,任务到处理器的分配可以形式化地描述成从任务集合到处理器集合的一个映射 $f:T\rightarrow P$,任务分配方案可通过向量 $\boldsymbol{Z}=(z_1,z_2,\cdots,z_n)\in\{1,2,\cdots,m\}^n$ 来表示,$\boldsymbol{Z}$ 的总代价表示为 $C(\boldsymbol{Z})$.如果任务 t_i 被分配到计算节点 p_k 上,则 $z_i=k,1\leqslant i\leqslant n$.定义 e_{iq}表示任务 t_i 在计算节点 p_q 上的计算代价,c_{ij}表示任务 t_i 和任务 t_j 被分配到不同计算节点上时产生的通信代价,I_{ij}表示任务 t_i 和任务 t_j 被分配到同一计算节点时产生的冲突代价,$1\leqslant j\leqslant n,1\leqslant q\leqslant m$.定义二元变量 $x_{iq}\in\{0,1\}$,当任务 t_i 未被分配到计算节点 p_q 时,$x_{iq}=0$;当 t_i 被分配到 p_q 时,$x_{iq}=1$.定义三元变量 x_{ijq},当任务 t_i 和任务 t_j 均未被分配到 p_q 上时,$x_{ijq}=0$;当 t_i 或 t_j 被分配到 p_q 时,$x_{ijq}=1$;当 t_i 和 t_j 均被分配到 p_q 时,$x_{ijq}=2$.设 x_q 表示分配到 p_q 的任务数.如果对于所有的 $t_i,t_j\in T$,有 $c_{ij}=c_0,I_{ij}=I_0$(c_0 和 I_0 为常数),则任务分配问题称为一致代价任务分配问题,否则,称为非一致代价任务分配问题.

本文假设多核节点以及节点间的互联网络是同构的.并且,$c_{ij}=c_{ji},I_{ij}=I_{ji}$;当 $z_i=z_j$ 时,有 $c_{ij}=0$;当 $z_i\neq z_j$ 时,有 $I_{ij}=0$.冲突代价(C_i)包含节点内通信代价(C_{intra})和同步代价(C_{syn}).同步操作可看作一种特殊形式的通信[18],C_{intra}和 C_{syn}是可加的,$C_i=C_{intra}+C_{syn}$.其中,C_i、C_{intra}和 C_{syn}都表示时间.

3 多核集群任务分配问题复杂性分析

3.1 单个计算节点上的节点间通信代价和节点内冲突代价分析

定理 1 对于一致代价任务分配问题和任意计算节点 p_q,若分配到 p_q 上的任务数为 x_q,则在 p_q 上产生的总的节点间通信代价为 $x_q(n-x_q)c_0$,总的节点内冲突代价为 $0.5x_q(x_q-1)I_0$.

证明 对于任意计算节点 p_q,若分配到该节点上的任务数为 x_q,则必然有 $n-x_q$ 个任务未被分配到该节点上.由于任务通信网络是完全图,因此这 x_q 个任务与其余 $n-x_q$ 个任务之间两两存在通信,共有 $x_q(n-x_q)$组通信.因为每组通信代价为 c_0,所以 p_q 上的总的节点间通信代价为 $x_q(n-x_q)c_0$.冲突代价只产生于分配到 p_q 上的 x_q 个任务之间,且每两个任务之间的冲突代价为 I_0,因此总的冲突代价为 $0.5x_q(x_q-1)I_0$.

推论 1 对于非一致代价任务分配问题,任务 t_i 和 t_j 在任意计算节点 p_q 上产生的节点间通信代价等于 $x_{ijq}(2-x_{ijq})c_{ij}$,产生的冲突代价等于 $0.5x_{ijq}(x_{ijq}-1)I_{ij}$.其中,$x_{ijq}=x_{iq}+x_{jq}$.

证明 根据定理 1,n 个节点间通信代价为常数 c_0 的任务在计算节点 p_q 上产生的总通信代价为 $x_q(n-x_q)c_0$.当仅考察任意两个任务 t_i 和 t_j 时,有 $n=2,c_0=c_{ij}$,且 $x_q=x_{iq}+x_{jq}=x_{ijq}$.因此,这两个任务在节点 p_q 上

产生的节点间通信代价为 $x_{ijq}(2-x_{ijq})c_{ij}$. 同理,这两个任务在 p_q 上产生的冲突代价为 $0.5x_{ijq}(x_{ijq}-1)I_{ij}$.

3.2 一致代价任务分配问题复杂性分析

定理 2 节点内冲突代价大于等于节点间通信代价的一致代价任务分配问题是 P 问题,可在多项式时间内求解.

证明 (1)通过如下方法将一致代价任务分配问题转化为图 G 上的最小费用流问题,如图 1 所示. 第 i 个任务对应图 G 中的结点 vt_i,所有任务对应的结点集为 $VT=\{vt_1,vt_2,\cdots,vt_n\}$. 第 q 个计算节点对应图 G 中的结点 vp_q,所有计算节点对应的结点集为 $VP=\{vp_1,vp_2,\cdots,vp_m\}$. 所有任务结点均与源点 r 相连,弧上容量为 1,费用为 0;所有计算结点均与汇点 s 相连,弧上容量为 n,费用函数为 $0.5x_q(n-x_q)c_0+0.5x_q(x_q-1)I_0$. 并且,每一个结点 vt_i 与每一个结点 vp_q 之间均有弧相连,弧上容量为 1,费用函数为 $e_{iq}x_{iq}$. 指定初始流量为 n,所有弧上的流均为整数流.

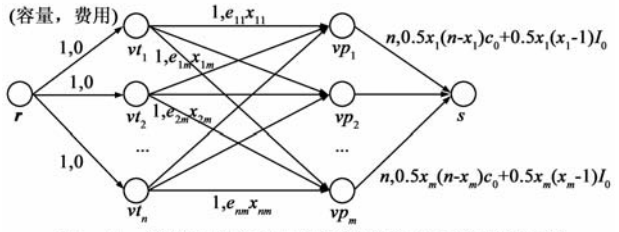

图1 与一致代价任务分配问题等价的最小费用流问题

(2)证明以上两问题的等价性. 首先,证明每一个可行流都对应一个任务分配方案. 由于初始流量为 n,所以对任意结点 $vt_i\in VT$,流入 vt_i 的流量均为 1. 根据流量守恒,流出 vt_i 的流量也为 1. 因为所有弧上的流均为整数流,所以从结点 vt_i 出发的弧中有且仅有一条弧上的流量为 1. 即,结点 vt_i 对应的任务被分配且仅被分配到唯一一个计算节点 p_{z_i} 上. 给定任意可行流 $\boldsymbol{F}^0$,不失一般性,假设所有从 VT 中的结点指向 VP 中结点的弧中流量为 1 的弧集为 $\{(vt_1,vp_{z_1}),(vt_2,vp_{z_2}),\cdots,(vt_n,vp_{z_n})\}$,则可行流 $\boldsymbol{F}^0$ 对应的任务分配方案 $\boldsymbol{Z}^0=(z_1,z_2,\cdots,z_n)$.

其次,证明每一个任务分配方案都对应一个可行流. 给定任意任务分配方案 $\boldsymbol{Z}^0=(z_1,z_2,\cdots,z_n)$,可以通过如下方式构造可行流 $\boldsymbol{F}^0$. 由于任务数为 n,所以初始流量为 n,即流入结点 vt_i 的流量均为 1. 若结点 vt_i 对应的任务被分配到计算节点 p_{z_i} 上,则弧 (vt_i,vp_{z_i}) 上的流量为 1. 由此,可构造出可行流 $\boldsymbol{F}^0$,所有从 VT 中的结点指向 VP 中的结点的弧中流量为 1 的弧集为 $\{(vt_1,vp_{z_1}),(vt_2,vp_{z_2}),\cdots,(vt_n,vp_{z_n})\}$.

最后,证明可行流的总费用等于对应任务分配方案的总代价,最小费用流对应最优任务分配方案. 易知,最小费用流问题中的费用函数 $0.5x_q(n-x_q)c_0+0.5x_q(x_q-1)I_0$ 对应任务分配问题中的节点间通信代价和节点内冲突代价之和,$e_{iq}x_{iq}$ 对应计算代价. 因此,任何一个可行流的总费用均等于对应任务分配方案的总代价. 另外,对于任意最小费用流 $\boldsymbol{F}^*$,假设 $\boldsymbol{F}^*$ 对应一个非最优的任务分配方案 $\boldsymbol{Z}^0$,即 $C(\boldsymbol{F}^*)=C(\boldsymbol{Z}^0)$,则存在一个最优任务分配方案 $\boldsymbol{Z}^*$,使得 $C(\boldsymbol{Z}^*)<C(\boldsymbol{Z}^0)$. 又因为 $\boldsymbol{Z}^*$ 对应一个可行流 $\boldsymbol{F}^0$,使得 $C(\boldsymbol{Z}^*)=C(\boldsymbol{F}^0)$,所以 $C(\boldsymbol{F}^0)<C(\boldsymbol{F}^*)$,与 $\boldsymbol{F}^*$ 是最小费用流相矛盾. 因此,每一个最小费用流对应一个最优任务分配方案.

(3)下面通过分析费用函数对最小费用流问题复杂性的影响关系来分析冲突代价对任务分配问题复杂性的影响关系. 根据构造过程,最小费用流问题中的费用函数可表示为:$0.5x_i(n-x_i)c_0+0.5x_i(x_i-1)I_0=0.5(I_0-c_0)x_i^2+0.5(nc_0-I_0)x_i$. 该二次费用函数的凹凸性由系数 I_0-c_0 决定. 根据 I_0-c_0 正负性的不同,可以将最小费用流问题区分为下面三类问题:

$$\begin{cases} I_0=c_0, & \text{线性费用网络最小费用流问题;}\\ I_0>c_0, & \text{凸费用网络最小费用流问题;}\\ I_0<c_0, & \text{凹费用网络最小费用流问题.}\end{cases}$$

其中,线性费用网络最小费用流问题和凸费用网络最小费用流问题是 P 问题,凹费用网络最小费用流问题是 NP-hard 问题. 由此可得出,节点内冲突代价大于等于节点间通信代价的一致代价任务分配问题是 P 问题,可转换为凸费用网络最小费用流问题,可在 $O(M\log N(M+N\log N))$ 时间内求解[19],M 表示边数,N 表示结点数. 因此,节点内冲突代价大于等于节点间通信代价的一致代价任务分配问题可在 $O(m^2n^2\log(m+n))$ 时间内求解,m 表示计算节点数,n 表示任务数.

3.3 非一致代价任务分配问题复杂性分析

定理 3 节点内冲突代价大于等于节点间通信代价的非一致代价任务分配问题是 P 问题,可在多项式时间内求解.

证明 (1)将非一致代价任务分配问题转换为广义网络最小费用流问题. 通过如下方法将非一致代价任务分配问题转化为图 G 上的广义网络最小费用流问题,如图 2 所示. G 中除了源点 r 和汇点 s 外,其它结点可分为三层. 第一层为任务结点层 $VT=\{vt_1,vt_2,\cdots,vt_n\}$,结点 vt_i 对应第 i 个任务. 第二层为任务分配结点层 $VA=\{va_{1:1},va_{1:2},\cdots,va_{n:m}\}$. 如果流经结点 $va_{i:q}$ 的流量为 1,则表示第 i 个任务被分配到了第 q 个计算节点上;反之,如果为 0,则表示第 i 个任务未被分配到第 q 个计算节点上. 第三层为任务对分配结点层 $VI=\{vi_{1,2:1},vi_{1,2:2},\cdots,vi_{n-1,n:m}\}$. 如果流经结点 $vi_{i,j:q}$ 的流量为 0,则表示第 i 个任务和第 j 个任务均未被分配到第 q 个计算节点上;如果流经结点 $vi_{i,j:q}$ 的流量为 1,则

表示第 i 个任务或第 j 个任务被分配到第 q 个计算节点上;如果流经结点 $vi_{i,j:q}$ 的流量为 2,则表示第 i 个任务和第 j 个任务均被分配到第 q 个计算节点上.

G 中的弧可以分为四层.第一层为弧集 $E_1=\{(r, vt_i)\}$,弧上的容量为 1,费用为 0,增益为 1.第二层为弧集 $E_2=\{(vt_i, va_{i:q})\}$,弧上的容量为 1,费用函数为 $e_{iq}x_{iq}$,增益为 $n-1$.第三层为弧集 $E_3=\{(va_{i:q}, vi_{i,j:q})\}\cup\{(va_{j:q}, vi_{i,j:q})\}$,弧上的容量为 1,费用为 0,增益为 1.第四层为弧集 $E_4=\{(vi_{i,j:q}, s)\}$,弧上的容量为 2,费用为 $0.5x_{ijq}(2-x_{ijq})c_{ij}+0.5x_{ijq}(x_{ijq}-1)I_{ij}$,增益为 1.因为弧上的增益系数不全为 1,所以该费用网络为广义费用网络.指定初始流量为 n,所有弧上的流均为整数流.

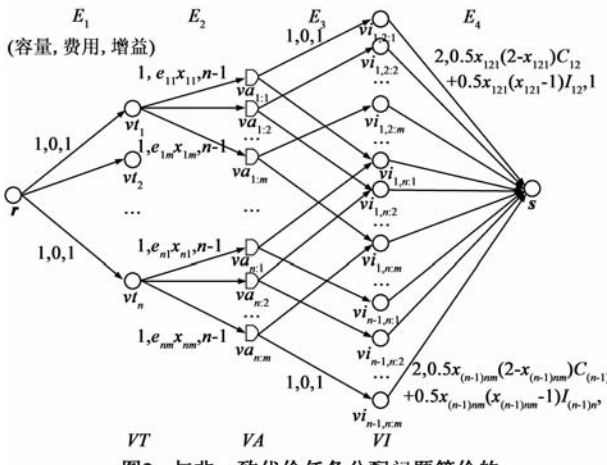

图2 与非一致代价任务分配问题等价的广义网络最小费用流问题

我们以图 2 中的弧 (r, vt_1)、$(vt_1, va_{1:1})$、$(va_{1:1}, vi_{1,2:1})$和$(vi_{1,2:1}, s)$上的费用函数的计算过程来说明从各个节点到汇点的费用函数是如何计算的,其它弧上的费用函数的计算方法与此相同.每段弧按顺序包含了三项计算,即:容量、费用和增益.对于弧(r, vt_1),由于没有费用发生,因此指定费用函数为 0;对于弧$(vt_1, va_{1:1})$,费用函数为任务 t_1 分配到计算节点 p_1 上发生的执行代价,即 $e_{11}x_{11}$;对于弧$(va_{1:1}, vi_{1,2:1})$,由于没有费用可发生,因此费用函数为 0;对于弧$(vi_{1,2:1}, s)$,根据任务 t_1 和 t_2 分配到计算节点 p_1 上的情况,在该弧上的费用为节点间通信代价与冲突代价之和,即:$0.5x_{121}(2-x_{121})c_{12}+0.5x_{121}(x_{121}-1)I_{12}$.

(2)证明以上两问题的等价性.首先,证明每一个可行流都对应一个任务分配方案.由于初始流量为 n,所以对于所有的 $vt_i\in VT$,流进该结点的流量为 1.根据流量守恒,从 vt_i 出发的 m 条弧中有且仅有一条弧上的流量为 1,其它弧上的流量均为 0.即,结点 vt_i 对应的第 i 个任务被分配且仅被分配到唯一一个计算节点 p_{z_i} 上.给定任意可行流 $\boldsymbol{F}^0$,不失一般性,假设所有从任务结点指向任务分配结点的弧中流量为 1 的弧集为$\{(vt_1, vp_{z_1}), (vt_2, vp_{z_2}), \cdots, (vt_n, vp_{z_n})\}$,则可行流 $\boldsymbol{F}^0$ 对应的任务分配方案 $\boldsymbol{Z}^0=(z_1, z_2, \cdots, z_n)$.

其次,证明每一个任务分配方案都对应一个可行流.给定任意任务分配方案 $\boldsymbol{Z}^0=(z_1, z_2, \cdots, z_n)$,可以通过如下方式构造可行流 $\boldsymbol{F}^0$.由于任务数为 n,所以初始流量为 n,即流入任意结点 $vt_i\in VT$ 的流量均为 1.若任务 t_i 对应的第 i 个任务被分配到计算节点 p_{z_i} 上,则弧 $(vt_i, va_{i:z_i})\in E_2$ 上的流量为 1.在确定了弧集 E_2 中弧上流量之后,弧集 E_3 中弧上的流量也随之确定.这是因为,对于任意一条流量为 1 的弧$(vt_i, va_{i:z_i})\in E_2$,弧上的增益系数为 $n-1$,从而使得流出结点 $va_{i:z_i}$ 的流量为 $n-1$,于是从结点 $va_{i:z_i}$ 出发的 $n-1$ 条容量均为 1 的弧上的流量也为 1.由此,可以构造出可行流 $\boldsymbol{F}^0$,其中弧集 E_2 中流量为 1 的弧集为$\{(vt_1, va_{1:z_1}), (vt_2, va_{2:z_2}), \cdots, (vt_n, va_{n:z_n})\}$.

最后,证明可行流的总费用等于对应任务分配方案的总代价,最小费用流对应最优任务分配方案.易知,广义网络最小费用流问题中的费用函数 $0.5x_{ijq}(2-x_{ijq})c_{ij}+0.5x_{ijq}(x_{ijq}-1)I_{ij}$对应任务分配问题中的节点间通信代价和节点内冲突代价之和,$e_{iq}x_{iq}$对应计算代价.因此,任何一个可行流的总费用均等于对应任务分配方案的总代价.对于任意最小费用流 $\boldsymbol{F}^*$,假设 $\boldsymbol{F}^*$对应一个非最优的任务分配方案 $\boldsymbol{Z}^0$,即 $C(\boldsymbol{F}^*)=C(\boldsymbol{Z}^0)$,则存在一个最优任务分配方案 $\boldsymbol{Z}^*$,使得 $C(\boldsymbol{Z}^*)<C(\boldsymbol{Z}^0)$.又因为 $\boldsymbol{Z}^*$对应一个可行流 $\boldsymbol{F}^0$,使得 $C(\boldsymbol{Z}^*)=C(\boldsymbol{F}^0)$,因此 $C(\boldsymbol{F}^0)<C(\boldsymbol{F}^*)$,与 $\boldsymbol{F}^*$是最小费用流相矛盾.故,每一个最小费用流对应一个最优任务分配方案.

(3)分析冲突代价对问题复杂性的影响关系.下面通过分析费用函数对广义网络最小费用流问题复杂性的影响关系来分析冲突代价对任务分配问题复杂性的影响关系.根据构造过程,广义网络最小费用流问题中的二次费用函数可表示为:$0.5x_{ijq}(2-x_{ijq})c_{ij}+0.5x_{ijq}(x_{ijq}-1)I_{ij}=0.5(I_{ij}-c_{ij})x_{ijq}^2+(c_{ij}-0.5I_{ij})x_{ijq}$.该二次费用函数的凹凸性由系数 $I_{ij}-c_{ij}$决定.根据 $I_{ij}-c_{ij}$正负性的不同,可以将最小费用流问题区分为下面三类问题:

$$\begin{cases} I_{ij}=c_{ij}, & \text{广义线性费用网络最小费用流问题;} \\ I_{ij}>c_{ij}, & \text{广义凸费用网络最小费用流问题;} \\ I_{ij}<c_{ij}, & \text{广义凹费用网络最小费用流问题.} \end{cases}$$

其中,广义线性费用网络最小费用流问题和广义凸费用网络最小费用流问题是 P 问题,广义凹费用网络最小费用流问题是 NP-hard 问题.由此可得出,冲突代价大于等于节点间通信代价的非一致代价任务分配问题是 P 问题,可转换为广义凸费用网络最小费用流

问题,可以在 $O(MN)$时间内求解[19],M 表示边数,N 表示结点数.因此,冲突代价大于等于节点间通信代价的非一致代价任务分配问题可在 $O(m^2n^4)$时间内求解,m 表示计算节点数,n 表示任务数.

4 问题解法及定理结论可应用性与有效性的讨论

前文从理论上分析并证明了多核集群中节点内冲突代价对问题复杂性的影响关系,本节讨论各种复杂性下的多核集群任务分配问题的解法以及本文定理与结论的可应用性与有效性.

4.1 多核集群任务分配问题的解法

多核集群任务分配问题可利用广义网络流规划模型来进行描述,并可通过最小费用流进行求解.然而,凹凸性不同的费用网络的最小费用流问题之间的复杂性存在很大差异[20].在一般情况下,多核集群任务分配问题是 NP-hard 问题,不存在多项式解法,可采用近似或启发式的次优算法进行求解.当节点内冲突代价等于节点间通信代价时,可通过流量增广法或原本法[20]进行求解;而当节点内冲突代价大于节点间通信代价时,可将凸费用转化为线性费用,进而使用流量增广法或原本法进行求解.对图 2 所代表的模型中的弧集 E_4 中的弧上的凸费用进行分段线性近似表示,每条凸费用弧可以用两条线性费用弧替代.如此,最小凸费用流问题转化为最小线性费用流问题,从而可利用流量增广法或原本法进行求解.

此外,图 2 所代表的模型对应的数学规划模型可表示为式(1).因此,多核集群任务分配问题也可以利用数学规划方法来求解.其中,$E^+(i)$表示结点 i 的出弧集,$E^-(i)$表示结点 i 的入弧集,$q_k=e_{ij}$,k 表示弧$(vt_i, va_{i:j})$;$u_k=[(1-0.5x_{ijq})c_{ij}+0.5(x_{ijq}-1)I_{ij}]x_{ijq}$,$k$ 表示弧$(vi_{i,j:q}, s)$,f_k 表示弧 k 上的流量,V 为图 G 的顶点集.

事实上,文献[21]中的实验结果表明了本文方法具有优势.

$$
\begin{cases}
\text{Min} \sum\limits_{k\in E_2} q_k x_k + \sum\limits_{k\in E_4} u_k x_k, \\
\text{s.t.} -\sum\limits_{k\in E^-(i)} f_k + \sum\limits_{k\in E^+(i)} f_k = 0, i \in V \setminus \{r, s, VA\}, \\
\quad -\sum\limits_{k\in E^-(i)} (n-1) f_k + \sum\limits_{k\in E^+(i)} f_k = 0, i \in VA, \\
\quad -\sum\limits_{k\in E^-(r)} f_k + \sum\limits_{k\in E^+(r)} f_k = n, \\
\quad -\sum\limits_{k\in E^-(s)} f_k + \sum\limits_{k\in E^+(s)} f_k = -n(n-1), \\
\quad 0 \leqslant f_k \leqslant 2, k \in E_4, \\
\quad 0 \leqslant f_k \leqslant 1, k \in E_1 \cup E_2 \cup E_3.
\end{cases} \tag{1}
$$

4.2 定理与结论的可应用性与有效性

本文定理与结论从理论上揭示了冲突代价对任务分配问题复杂性的影响关系,其成立具有严格的约束条件.在实际求解任务分配问题的过程中,由于所求解问题的非规则性、机器拓扑结构的异构性和可用计算资源的动态性等因素[10],无法也不必考虑每个任务的节点内冲突代价和节点间通信代价.因此,根据实际情况可放宽假设条件,例如,可通过平均的节点内冲突代价和节点间通信代价来讨论它们对任务分配问题复杂性的大致影响关系.在本小节中,结合任务分配和负载均衡过程中的实际问题,讨论本文定理和结论的可应用性与有效性,主要体现在以下几个方面.

(1)如果分配到各个节点上的任务之间的依赖度较小,则意味着节点之间的通信代价较小,同步代价相对较高,冲突代价相对较高,并随节点内核数和任务数的增多而增大,从而可满足冲突代价大于或等于节点间通信代价条件,任务分配问题可转化为凸费用网络最小费用流问题或线性费用网络最小费用流问题而求解.事实上,这种情况下的并行计算实际加速比可达到线性加速比.

(2)当任务规模较小且任务之间的依赖度较高时,意味着任务之间的通信量较大,使用更多的计算节点得不偿失.使用较少的节点参与计算,节点间通信代价较小,节点内的任务之间由于存在较高的依赖度而产生较高的同步代价和节点内通信代价,冲突代价随之增大,从而可满足冲突代价大于或等于节点间通信代价条件,任务分配问题因此可转化为凸费用网络最小费用流问题或线性费用网络最小费用流问题而求解.

(3)当任务规模较大且任务之间的依赖度较高时,意味着通信量较大,但并非意味着需要使用更大规模的节点参与计算.事实上,很少的应用需要使用多于1000 个的处理器进行并行求解.考虑到通信和同步开销,存在一个能够有效求解一特定问题的计算节点数目.那么,可把依赖度高的多个任务聚合为一个任务并分配到同一节点上以减小节点间通信代价,而这同时会增大节点内冲突代价.对于这种情形,当每个节点内的核数较大时,参与计算的节点数较少,仍可满足冲突代价大于或等于节点间通信代价条件,任务分配问题从而可转化为凸费用网络或线性费用网络的最小费用流问题而求解.

(4)当大量的任务之间的依赖度很高且每个节点上的核数较小时,为了获得并行性,通常需要把这些任务分配给较多的小节点,从而使得节点间通信代价较高,冲突代价相对较小,导致冲突代价小于节点间通信代价,计算复杂性高.事实上,在这种情况下,应用问题本身就不具有可并行性.

5 结论

利用最小费用流理论分析并证明了多核集群节点

内冲突代价对任务分配问题复杂性的影响关系.结果表明:(1)当冲突代价等于节点间通信代价时,任务分配问题可转换为广义线性费用网络最小费用流问题,可在 $O(m^2n^4)$时间内求解;(2)当冲突代价大于节点间通信代价时,任务分配问题可转换为广义凸费用网络最小费用流问题,可在多项式时间内求解.特别地,对于一致代价的任务分配问题,当冲突代价大于等于节点间通信代价时,可转换为凸费用网络最小费用流问题,可在 $O(m^2n^2\log(m+n))$时间内求解;(3)当冲突代价小于节点间通信代价时的任务分配问题是 NP-hard 问题,求解该问题的高效多项式启发式算法有待研究.

参考文献

[1] H S Stone. Multiprocessor scheduling with the aid of network flow algorithms[J]. IEEE Transactions on Software Engineering, 1977, 3(1): 85 – 93.

[2] S H Bokhari. On the mapping problem[J]. IEEE Transactions on Computers, 1981, 30(5): 207 – 214.

[3] C C Shen, et al. A graph matching approach to optimal task assignment in distributed computing systems using a minimax criterion[J]. IEEE Transactions on Computers, 1985, 34(3): 197 – 203.

[4] D F Towsley. Allocating programs containing branches and loops within a multiple processor system[J]. IEEE Transactions on Software Engineering, 1986, 12(10): 1018 – 1024.

[5] S Menon. Effective reformulations for task allocation in distributed systems with a large number of communicating tasks [J]. IEEE Transactions on Knowledge and Data Engineering, 2004, 16(12): 1497 – 1508.

[6] A Ernst, et al. Exact solutions to task allocation problems[J]. Management Science, 2006, 52(10): 1634 – 1646.

[7] 杨际祥,等.多核软件的几个关键问题及其研究进展[J].电子学报,2010,38(9):2140 – 2146.
YANG Ji-xiang, et al. Some key issues and their research progress in multi-core software[J]. Acta Electronica Sinica, 2010, 38(9): 2140 – 2146. (in Chinese)

[8] L Chai, et al. Understanding the impact of multi-core architecture in cluster computing: A case study with Intel dual-core system[A]. Proc. 7th IEEE CCGRID[C]. Washington: IEEE CS Press, 2007. 471 – 478.

[9] 涂碧波,等.多核处理器机群 Memory 层次化并行计算模型研究[J].计算机学报,2008,31(11):1948 – 1955.
TU Bi-bo, et al. Research on parallel computation model with memory hierarchy on multi-core clusters[J]. Chinese Journal of Computers, 2008, 31(11): 1948 – 1955. (in Chinese)

[10] 杨际祥,等.并行与分布式计算动态负载均衡策略综述[J].电子学报,2010,38(5):1122 – 1130.
YANG Ji-xiang, et al. A survey of dynamic load balancing strategies for parallel and distributed computing[J]. Acta Electronica Sinica, 2010, 38(5): 1122 – 1130. (in Chinese)

[11] S H Bokhari. Dual processor scheduling with dynamic reassignment[J]. IEEE Transactions on Software Engineering, 1979, 5(4): 341 – 349.

[12] P K Yadav, et al. Scheduling algorithm: Tasks scheduling algorithm for multiple processors with dynamic reassignment [J]. Journal of Computer Systems, Networks, and Communications, 2008. 1 – 9.

[13] M Zaharia, et al. Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling[A]. Proc. EuroSys'10[C]. New York: ACM Press, 2010. 265 – 278.

[14] X Lin, et al. Real-time scheduling of divisible loads in cluster computing environments[J]. Journal of Parallel and Distributed Computing, 2010, 70(3): 296 – 308.

[15] M Stillwell, et al. Dynamic fractional resource scheduling for cluster platforms[A]. Proc IPDPS Workshops[C]. Washington: IEEE CS Press, 2010. 1 – 4.

[16] H Casanova, et al. On cluster resource allocation for multiple parallel task graphs[J]. Journal of Parallel and Distributed Computing, 2010, 70(12): 1193 – 1203.

[17] Z F Yu, et al. Failure-aware workflow scheduling in cluster environments[J]. Cluster Computing, 2010, 13(4): 421 – 434.

[18] J Parrow. An introduction to the π-calculus. Handbook of Process Algebra[M]. Amsterdam: Elsevier Science, 2001. 479 – 543.

[19] R K Ahuja, et al. Network Flows: Theory, Algorithms, and Applications[M]. New Jersey: Prentice Hall Press, 1993.

[20] P A Jensen, et al. Network Flow Programming[M]. Malabar, Fla: Krieger Pub Co, 1987.

[21] YANG Ji-xiang, et al. Solution to new task allocation problem on multi-core clusters[J]. Journal of Computational Information Systems, 2011, 7(5): 1691 – 1697.

作者简介

谭国真 男,1960 年生,教授,博士生导师.研究方向为集群计算、网络优化、智能交通系统与无线传感器网络.
E-mail: gztan@dlut.edu.cn

杨际祥 男,1975 年生,博士研究 生.研究方向为集群计算与多核计算.
E-mail: jixiang_yang@126.com