异构集群下的任务调度算法研究

刘 莉[a], 姜明华[b]
(武汉纺织大学 a. 数学与计算机学院; b. 研究生处, 武汉 430073)

摘 要: 针对异构集群下高效节能的任务调度算法进行了研究,提出了一种基于复制的任务调度算法,在任务初始分配的基础上,分别从能源感知和性能—能源平衡两个角度考虑任务的复制。建立了由计算和通信造成的能源消耗的数学模型,并进行了大量的实验。实验结果表明,与已有的 BEATA 算法相比,该算法能明显地减少异构集群处理并行应用的调度长度和能耗。分析结果发现,任务复制的方法在减少调度长度的同时会增加相应的能耗,能同比优化调度长度和能耗的任务调度方法是今后的研究方向。

关键词: 任务调度; 任务复制; 异构性; 调度长度; 能源消耗
中图分类号: TP301.6 **文献标志码**: A **文章编号**: 1001-3695(2014)01-0080-05
doi:10.3969/j.issn.1001-3695.2014.01.018

Research of task scheduling algorithm on heterogeneous cluster

LIU Li[a], JIANG Ming-hua[b]
(*a. College of Mathematics & Computer Science, b. Graduate Department, Wuhan Textile University, Wuhan* 430073, *China*)

Abstract: Research for fast and energy-efficient task scheduling algorithm on heterogeneous cluster, this paper proposed a duplication-based task scheduling algorithm, which based on the initial allocation of tasks, duplicated tasks from two aspects of energy aware and performance-energy balance. It established mathematical models for the energy consumption caused by computing and communication, and did extensive experiments. Results show that compared with an existing algorithm-balanced energy-aware task allocation (BEATA), this method can significantly reduce the schedule length and energy consumption of processing parallel applications on heterogeneous cluster, and task duplication method can reduce the schedule length but meanwhile will increase energy consumption. Thus, the task scheduling method which can simultaneously reduce schedule length and energy consumption is the direction of future research.

Key words: task scheduling; task duplication; heterogeneity; schedule length; energy consumption

0 引言

中国的数据中心发展迅速。数据中心的应用由早期对数据的存储、访问逐步发展到深度处理、信息挖掘阶段,数据也从静态存储转向动态处理和交换,数据量大幅增长,应用的种类在不断增加。用户需求的快速增长,使得数据中心的规模不断扩大,数量不断增加,随之快速增长的是数据中心的能源消耗。2009 年,我国数据中心总耗电量约 364 亿 kW/h,占当年全国总电耗的 1%。未来几年,我国数据中心仍将快速发展,如果维持当前的低能效水平,到 2015 年,仅全国的数据中心就将消耗掉三峡电站 1 年的发电量[1]。各种规模的企业由于其数据中心与能源有关的开销正在经历重大的挑战,数据中心的能源危机抑制客户的业务增长,数据中心的满负荷生产,限制了一个公司的成长和作出必要的资本投资的能力。因此,数据中心能源消耗问题已成为阻碍数据中心发展的重要问题之一。

集群系统作为高性能并行系统的一种,能够被广泛地应用,与其自身的特点是分不开的。最初组建的集群一般是同构的,但随着应用需求的增加,集群的拓展可能导致集群的异构,例如增加了配置不同的硬件设备,如处理器、硬盘或交换机等,或增加了相同的硬件设备,但新旧硬件的工作效率可能不同。任务在各个计算节点上执行的开始时间可能不同,因为计算节点可能有不同的时钟频率[2]。另外,集群的高可用性即故障处理方面,由于相同的硬件一般具有相同的使用寿命和故障率,如存储设备,集群可能会配备不同品牌、不同厂家的同类产品,以防止突发故障集群性能的明显降低。虚拟化技术是数据中心必不可少的技术,当需要把虚拟机迁移到另外一台服务器上进行维护[3],或当集群中计算节点上的虚拟机数不同时,如节点 1 虚拟了四台,节点 2 虚拟了两台,则两个节点上虚拟机被调度处理同一个应用时,就可能造成集群的异构。

1 相关节能和调度技术

在处理器节能方面,动态电压缩放(dynamic voltage scaling,DVS)技术已被广泛利用,使处理器在可移植和不可移植的计算系统中都节能[4];动态频率调整(dynamic frequency scaling,DFS)技术,当处理器不完全利用时以较低的频率运行以节省电力[4,5];动态电压和频率缩放(dynamic voltage and frequency scaling,DVFS)技术,许多研究已证明它能降低集群和其他高性能计算平台的功耗,能为计算密集型应用节省相当数量的能源[5];动态电源管理(DPM)技术旨在以最低数量的活动部件或在这些部件上的最低负荷实现要求的性能,当系统组

收稿日期: 2013-03-15; **修回日期**: 2013-05-02

作者简介: 刘莉(1988-),女,硕士,主要研究方向为任务调度算法(ll_wuse@163.com);姜明华(1966-),男,教授,主要研究方向为信息集成、网络存储、服务计算.

件闲置时,能自适应关闭这些组件或降低它们的性能[6]。Xie 等人[7]提出了一种平衡能源感知任务分配策略(balanced energy-aware task allocation,BEATA),旨在当把时间表长度局限于一个理想的范围时最大限度地减少整体能源消耗。Zong 等人[8]提出了同构集群下能源感知的复制调度算法(energy-aware duplication,EAD)和性能—能源平衡的复制调度算法(performance-energy balanced duplication,PEBD),在不明显增加能耗的情况下提高集群处理并行应用的性能。最近,李新等人[9]提出一种启发式处理器合并优化方法,按照任务最早开始时间和最早结束时间查找处理器时间空隙,将轻负载处理器上的任务重新分配到其他处理器上,从而减少使用的处理器数目,降低系统总体能耗。

在互连能耗方面,已有研究表明,集群的电力消耗独立于链路利用。例如,Gunaratne 等人[10]发现在以太网中闲置和充分利用的链路消耗大约相同量的电力。Zamani 等人[11]还指出,Myrinet-2000 设备没有电源管理技术,不管网络流量,互连消耗的网络能源仍然不变。但是高速互连确实能源消耗显著。例如,路由器和链路消耗 Mellanox 刀片服务器总功率预算的 37%[12]。随着下一代高速互连的出现如千兆以太网、InfiniBand、Myrinet 和 QsNet^2,互连能耗越来越大。Shang 等人提出了把 DVS 技术应用到链路的架构模型。Soteriou 等人[13]调查为响应通信流量的差异把 DVS 链路载入极端动态的开/关链路的潜能。Gunaratne 等人[10]调查自适应链路速率(adaptive link rate,ALR),通过利用自适应地改变链路数据传输速率来减少一个典型的以太网链路的能源消耗。

本文提出了一种基于复制的任务调度算法(duplication-based task scheduling algorithm for AOV network,DTSV),主要是根据 AOV 网(activity on vertex network)的特性对任务进行预处理和分配,再从能源感知和性能—能源平衡的角度考虑任务复制,以在不明显提高能耗甚至能减少能耗的情况下,减少整个应用的调度长度,提高异构集群系统处理并行应用的性能。

2　数学模型

本章用数学模型来表示一个异构的集群,具有优先约束的并行应用和计算节点及互连的能源消耗。

2.1　集群模型

本文中的异构集群是用由高速互连连接的一组计算节点的集合 $P=\{p_1,p_2,\cdots,p_m\}$ 来表示的。假设集群是异构的,各个计算节点具有不同处理能力的处理器,则相同任务在不同的计算节点上执行时间可能不同,不同处理器的能耗率也可能不同。互连也假设为异构的,不同节点之间的传递速率不同,则任务分配到不同的节点上时,其传递消息的通信时间根据不同的互连而不同。为了简化集群系统,假设所有节点是全相连的,分配到同一个计算节点上的两个优先约束的任务之间不计通信时间,集群系统是无故障的,各个任务的页故障服务时间被整合入任务的执行时间。在能源节省方面,每个节点的能耗率由单位时间内的焦尔(Joule)来衡量。

2.2　任务模型

由一组优先约束的任务集合表示的并行应用,以有向无环图 DAG 的形式体现,表示为 $G=(V,E)$。其中:$V=\{v_1,v_2,\cdots,v_n\}$ 表示一组优先约束的任务集合;E 是一组消息的集合,表示任务之间的通信和优先约束关系。在集合 V 中,任务 v_i 在计算节点 p_u 上的计算时间由 e_u^i 表示。$e_{ij}=(v_i,v_j)\in E$ 表示任务 v_i 与 v_j 之间传递的消息。c_{uv}^{ij} 表示任务 v_i 与 v_j 之间的通信时间,其中任务 v_i 在节点 p_u 上,v_j 在 p_v 上。当两个有依赖关系的任务在同一节点上时,不计其通信时间。

设定任务集 V 中只有一个入口任务 v_{entry} 和一个出口任务 v_{exit}。任务分配矩阵(如 $\boldsymbol{X}$)是一个 $n\times m$ 的二进制矩阵,用来反映 n 个任务到集群中 m 个计算节点上的映射。如果元素 x_{ij} 为 1,则表示任务 v_i 分配在计算节点 p_j 上;反之,元素值为 0。

2.3　能源消耗模型

设 en_u^i 为任务 v_i 在计算节点 p_u 上的计算能耗,用 PN_u^{high} 来表示节点 p_u 工作时的能耗率,则能源消耗 en_u^i 表示为

$$en_u^i=PN_u^{\text{high}}\times e_u^i \tag{1}$$

给出一个并行应用和它的任务集 V,以及任务分配矩阵 $\boldsymbol{X}$,则可计算 V 中所有任务在处理器上的计算能耗:

$$EN_{\text{high}}=\sum_{i=1}^{|V|}\sum_{u=1}^{|P|}en_u^i\times x_{iu}=\sum_{i=1}^{n}\sum_{u=1}^{m}x_{iu}\times PN_u^{\text{high}}\times e_u^i=\sum_{u=1}^{m}\left(PN_u^{\text{high}}\times\sum_{i=1}^{n}x_{iu}\times e_u^i\right) \tag{2}$$

式(2)只计算了处理器活动时的能源消耗,使 PN_u^{low} 为计算节点空闲时的能耗率,定义 f_i 为任务 v_i 的完成时间。节点空闲时的能耗就是该计算节点空闲时的能耗率和节点空闲时长的产物,则计算节点 p_u 的空闲能耗计算为

$$EN_{\text{low}}^u=PN_u^{\text{low}}\times\left(\max_{i=1}^{n}f_i-\sum_{i=1}^{n}x_{iu}\times e_u^i\right) \tag{3}$$

其中:$\max_{i=1}^{n}f_i$ 是时间表长度,即出口任务的完成时间;而 $\left(\max_{i=1}^{n}f_i-\sum_{i=1}^{n}x_{iu}\times e_u^i\right)$ 是节点 p_u 上的空闲时间。所有计算节点的空闲时间总能耗为

$$EN_{\text{low}}=\sum_{u=1}^{m}EN_{\text{low}}^u=\sum_{u=1}^{m}\left(PN_u^{\text{low}}\times\left(\max_{i=1}^{n}f_i-\sum_{i=1}^{n}x_{iu}\times e_u^i\right)\right) \tag{4}$$

于是,运行在异构集群上的并行应用的总能耗可由式(2)和(4)派生出来,即

$$EN=EN_{\text{high}}+EN_{\text{low}}=\sum_{u=1}^{m}\left(PN_u^{\text{high}}\times\sum_{i=1}^{n}x_{iu}\times e_u^i\right)+\sum_{u=1}^{m}\left(PN_u^{\text{low}}\times\left(\max_{i=1}^{n}f_i-\sum_{i=1}^{n}x_{iu}\times e_u^i\right)\right) \tag{5}$$

互连上能耗的计算方式与任务在计算节点上的计算能耗类似。用 el_{ij} 表示消息 $e_{ij}=(v_i,v_j)\in E$ 传递所产生的能源消耗。假设任务 v_i 和 v_j 分别在计算节点 p_u 和 p_v 上,PL_{uv}^{high} 为节点 p_u 与 p_v 之间链路的能耗率,则传递消息 e_{ij} 所产生的通信能耗为

$$el_{uv}^{ij}=PL_{uv}^{\text{high}}\times c_{uv}^{ij} \tag{6}$$

集群的互连在这个研究中被假设为异构的,所以通过网络互连传递的消息可能有不同的传输速率。通信计算比 CCR 的大小体现了应用的类型,$CCR>1$ 表示通信密集型应用,反之则表示计算密集型应用。CCR 表示如下:

$$CCR=\frac{\frac{1}{|E|}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\frac{1}{m(m-1)}\sum_{u=1}^{m}\sum_{v=1,v\neq u}^{m}c_{uv}^{ij}\right)}{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{m}\sum_{u=1}^{m}t_u^i\right)} \tag{7}$$

根据已有的研究结果——集群中高速互连在空闲和完全使用时消耗的能源基本上相同,设定 PL_{uv} 表示链路的能耗率,并将互连上总的通信能耗表示如下:

$$EL=\sum_{u=1}^{m}\sum_{v=1,v\neq u}^{m}\max_{i=1}^{n}(f_i)\times PL_{uv} \tag{8}$$

最后在异构集群上运行一个并行应用会产生的总能耗便是计算节点上的总能耗与互连上的总能耗之和:

$$E = EN + EL \tag{9}$$

2.4 异构模型

在异构集群中,计算节点各异的处理能力使得一个任务在不同节点上的计算时间各不相同,这涉及到计算的异构性。任务计算时间的异构性是异构集群的一个重要特征,在分配算法中需要考虑到任务在不同节点上的计算时间不同来决定分配策略。

任务 v_i 在节点 p_u 上的计算权值(如 w_u^i)定义为 v_i 在 p_u 上的计算时间与集群中执行 v_i 最快的节点的比值,即 $w_u^i = \frac{e_u^i}{\min\limits_{k=1}^{m}(e_k^i)}$。任务 v_i 计算异构度表示如下:

$$HD_C^i = \sqrt{\frac{1}{m}\sum_{u=1}^{m}(w_i^{\text{avg}} - w_u^i)^2}, w_i^{\text{avg}} = \frac{(\sum_{u=1}^{m} w_u^i)}{m} \tag{10}$$

任务集 V 的计算异构度为

$$HD_C = \frac{1}{|V|}\sum_{i=1}^{n} HD_C^i \tag{11}$$

3 任务调度算法

任务调度问题是一个 NP 难问题,因此 DTSV 算法是启发式的,仅能提供较优的结果。DTSV 算法主要包括任务初始分配和任务复制两个部分。DTSV 以 AOV 网特性对任务集进行初始分配,即对任务集进行拓扑排序,再从能源感知和性能—能源平衡两个角度去考虑任务复制。

3.1 拓扑排序

对 DAG 中所有的任务进行拓扑排序,步骤如下:

a)统计所有任务的前任数 parents(1∶N)。

b)找到所有入口任务,放入列表中,一般为任务 v_1,将任务放入前任列表 predecessor( )中;

c)逐一取出前任列表中的任务 predecessor(i),找到该任务所有的后继,放入后继列表中的任务 successor( ),清空 predecessor( )。

d)逐一取出后继列表中的任务 successor(i)。如果该任务的 parents(i) =1,则把该任务放入拓扑有序列表 list( )中,同时放入 predecessor( ),parents(i) =0;如果当前后继的 parents(i) >1,则 parents(i) - -,清空 successor( )。

e)回到步骤 c)进行循环,直至所有任务已处理。

3.2 参数计算

在本研究中,衡量算法优劣的一个重要标准是异构集群上处理应用的调度时间长度。为了计算调度长度,需要计算和记录一些重要的参数,如计算节点的可用时间,记为 $P_a(u)$,$u \in P$,各个任务在各计算节点上的开始时间 st_u^i,$u \in P$ 且 $i \in V$,任务的最早开始时间 est(i,u),$i \in V$ 且分配到节点 u 上,及任务的最早完成时间 ect(i,u),$i \in V$ 且分配到节点 u 上。

本研究中异构集群下任务间的约束条件为当前任务必须在所有前任都完成后才可以开始执行。算法第一步是任务初步分配,第二步是任务复制。所以参数计算也分为两个部分,第一部分为初步分配的参数,计算如下:

$$st_v^j = \begin{cases} \max(P_a(v), \text{ect}(i,u) + c_{uv}^{ij}) & u \neq v \\ P_a(v) & u = v \end{cases} \tag{12}$$

式(12)是各个任务在各计算节点上的开始时间 st_v^j。在找到最晚完成的前任 v_i 及其所在计算节点 p_u 后,根据节点号 v 是否等于 u,即任务 v_i 和 v_j 是否在同一计算节点上,来决定当前任务 v_j 在各计算节点 $p_v(v=1,2,\cdots,m)$ 上的开始时间。

$$\text{est}(i,u) = \min(st_v^i) \quad i=1,2,\cdots,n \text{ 且 } v=1,2,\cdots,m \tag{13}$$

式(13)表示任务 $v_i(i=1,2,\cdots,n)$ 的最早开始时间 est(i,u),根据其在各计算节点上的开始时间 $st_v^i(v=1,2,\cdots,m)$,取最小值,u 则记录任务 v_i 最终分配的计算节点号。

$$\text{ect}(i,u) = \text{est}(i,u) + e_u^i \tag{14}$$

式(14)表示任务 v_i 的最早完成时间 ect(i,u)就是最早开始时间 est(i,u)加上其在当前节点 p_u 上的执行时间 e_u^i。

$$P_a(u) = \text{ect}(i,u) \quad u=1,2,\cdots,m \tag{15}$$

式(15)表示当前节点 p_u 的可用时间 $P_a(u)$ 是最近分配到该节点的任务 v_i 的最早完成时间 ect(i,u)。

第二部分是任务复制,在这个部分任务可能被复制到其他计算节点上,各个任务在各计算节点上的最早开始时间和最早完成时间可能改变,其中最早开始时间 est_d 计算如下:

$$\text{est}_d(j,v) = \begin{cases} \max(P_a(v), \text{ect}(i,u) + c_{uv}^{ij}) & \text{无复制} \\ \max(\text{ect}(i,v), \text{ect}(i,u)) & \text{ect}(i,v) = P_a(v) + e_v^i \end{cases} \tag{16}$$

当任务 v_j 的前任 v_i 被复制时,v_i 在当前计算节点 p_v 上的开始时间为当前节点的可用时间 $P_a(v)$,完成时间为 $P_a(v) + e_v^i$。当前任务 v_j 的最早开始时间取节点可用时间,即 ect(i,v)和 v_i 在原始分配节点上的完成时间 ect(i,u)的最大值,以确保 v_i 在 v_j 前完成。

3.3 算法描述

在所有任务初始分配到各计算节点后,DTSV 将进行任务复制。分别从能源感知和性能—能源平衡两个角度去考虑任务的复制。

算法 1 能源感知的 DTSV

```
找到入口任务 v1;
i=2;拓扑有序列表的索引
while i≤n 遍历所有任务
  v=list(i);当前任务
  flag=0;0 表示所有前任与 v 都不在同一节点上,1 表示至少有一个在同一节点上
  for j=1 to i-1 遍历寻找 v 的前任
    u=list(j);
    if u 是 v 的前任且在同一个计算节点上
      flag=1;
    else 记录最晚完成的任务时间和任务号;
  if flag==0
      if 最晚完成的前任的完成时间+通信时间>当前节点的可用时间+前任在当前节点的计算时间
          增加的能耗=计算能耗-通信能耗;
          if 增加的能耗<门限值
              复制最晚完成的前任到当前计算节点;
所有任务都处理后,循环结束,返回任务分配矩阵、总能耗和调度时间
```

算法 2 性能—能源平衡的 DTSV

```
找到入口任务 v1;
i=2;拓扑有序列表的索引
while i≤n 遍历所有任务
  v=list(i);当前任务
  flag=0;0 表示所有前任与 v 都不在同一节点上,1 表示至少有一个在同一节点上
  for j=1 to i-1 遍历寻找 v 的前任
    u=list(j);
    if u 是 v 的前任且在同一个计算节点上
      flag=1;
```

else 记录最晚完成的任务时间和任务号;
if flag == 0
if 最晚完成的前任的完成时间 + 通信时间 > 当前节点的可用时间 + 前任在当前节点的计算时间
增加的能耗 = 计算能耗 - 通信能耗;
减少的时间 = 前任的最晚完成时间 + 通信时间 -(当前节点的可用时间 + 前任在当前节点的计算时间);
计算增加的能耗与减少的时间比 = 增加的能耗/减少的时间
if 比值 < 门限值
复制前任;
所有任务都处理后,循环结束,返回任务分配矩阵、总能耗和调度时间

3.4　时间复杂度分析

时间复杂度是衡量一个算法质量的重要指标。因为关系到算法执行的效率,下面对 DTSV 算法进行时间复杂度的分析。

定理 1　给出一个具有优先约束的并行应用,算法 DTSV 对该应用作出调度决策的时间复杂度为 $T(\mathrm{DTSV}) = O(n(m \lg m + kq) + |E|)$。其中:$|V|$表示任务的数量,值为 n;$|P|$表示计算节点的数量,值为 m;$|E|$表示任务图中边的数量;k 为预选最佳分配节点的节点数;q 为任务图中最大的度。

证明　DAG 中的任务拓扑排序的时间复杂度为 $O(|V| + |E|)$。初始分配任务的时间复杂度为:对每个任务在各计算节点上的完成时间排序的时间复杂度为 $O(|P|\lg|P|)$,为每个任务选取最佳分配的计算节点所需的时间复杂度为 $O(kq)$。考虑任务复制的时间复杂度为 $O(|V| + |E|)$,所以,DTSV 算法的总体时间复杂度为

$$O(|V| + |E| + |V|(|P|\lg|P| + kq) + |V| + |E|) = O(n(m \lg m + kq) + |E|)$$

4　性能评估

在本研究中,并行应用由一组带有优先约束的任务集来表示,任务间的执行约束条件为:对当前任务,必须保证其所有前任都完成后才可以开始。根据 OTOP(once tuning one parameter)的规则,将 DTSV 与 BEATA 算法的结果进行比较,从几个重要参数的改变来分析各个参数对算法结果的影响及算法的性能。

4.1　BEATA 算法

BEATA 算法是针对运行在异构的网络嵌入式系统的协作应用的一种平衡的能量感知任务分配策略,旨在将能源延迟高效方案整合到任务分配中,从而在节约能源与调度长度之间作最好的权衡。其中一个关键参数是能源自适应窗口(energy adaptive window,EAW),通过微调 EAW 的大小,可以方便地定制 BEATA,以满足特定的能量延迟权衡需求。根据任务之间的依赖关系,进行拓扑排序来重构任务列表,计算各个任务在各个计算节点上的计算时间并进行升序排序,最后根据 EAW 所划出的节点,根据能源感知策略来决定任务最终分配的节点。

4.2　仿真配置

遵循 OTOP 原则依次改变计算节点的数量、通信计算比(CCR)、应用类型和计算异构度来比较和总结算法的效果。在仿真过程中计算任务调度时间长度和总能耗,作为评估算法在性能和能效方面优劣的衡量点。

从 STG 集合中选取了四种应用类型,两种实际中的应用:robot control,任务数为 88;fpppp,任务数为 334。另外,随机选用了两个有 500 个任务数的合成应用,命名为 random1 和 random2[14]。仿真数据中采用了真实处理器的能耗参数来进行计算,所选用的处理器类型及忙碌和空闲时的能耗率[8] 如表 1 所示。为了观察互连的能耗情况,使用了在实际的集群中广泛使用的四种高速互联网络 Gigabit Ethernet、InfiniBand、Myrinet 和 OsNet^2,其能耗参数[8] 如表 2 所示。

表 1　处理器能耗参数

处理器类型	空闲模式能耗率/W	工作模式能耗率/W
Athlon 64 X2 4600 + 85W	15	104
Athlon 64 X2 4600 + 65W	14	75
Athlon 64 X2 3800 + 35W	11	47
Core 2 Duo E6300	26	44

表 2　网络能耗参数

网络类型	空闲能耗率/W	工作能耗率/W
Gigabit Ethernet	75	75
InfiniBand	25	25
Myrinet	55.2	55.2
OsNet^2	42	36

总结了四个主要的可变配置参数,如表 3 所示。为了更全面地观察和分析 DTSV 算法在异构集群环境下处理并行应用的性能,一次只改变一个参数,并记录该参数选取各个值时的实验结果,用来分析各个参数的改变对算法性能的影响。

表 3　重要参数特征

参数	值(固定值) -(选值)
任务数	(334) -(88,334,500,500)
节点数	(32) -(32,64)
计算异构度	(2) -(2,3,4,5)
CCR	(0.5) -(0.1,0.5,1,5)

4.3　性能分析

根据四个参数的几组数值,分别进行了计算,将数据结果转换为图表,从而更直观地观察和分析 DTSV 算法的效率。图 1 ~4 中,每组数据从左至右分别为:BEATA 算法的结果、能源感知的任务复制结果(表示为 $\mathrm{DTSV_{EAD}}$)以及性能—能源平衡的任务复制结果(表示为 $\mathrm{DTSV_{PEBD}}$)。

4.3.1　不同应用的性能

在算法中保持其他参数不变,依次更改四种应用,得到的结果如图 1 所示。

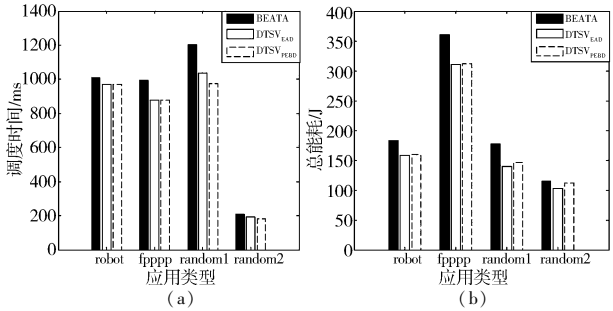

图 1　四种应用的调度长度和能耗

图 1 中算法 BEATA 的调度长度明显大于 DTSV 算法的结果,而性能—能源平衡的任务复制策略要稍优于能源感知策略。在能耗方面,DTSV 的能耗明显少于 BEATA 算法的结果,其中性能—能源平衡的能耗要稍多于能源感知的调度策略。

4.3.2　计算节点数量的影响

计算节点的数量为 64 时,四种应用的调度时间和总能耗的结果如图 2 所示。

与图 1(计算节点数为 32)相比,在 64 个节点的异构集群系统下,DTSV 算法处理四个应用的执行时间都明显少于 32 个节点的系统,说明了在异构集群中计算节点的数量对处理并行

应用的性能有一定的影响。在能耗方面,64 个计算节点的系统与 32 个节点的系统能耗相当,其中计算密集型应用 fpppp 的能耗有明显区别,这是因为较少的计算节点导致较长的调度时间。

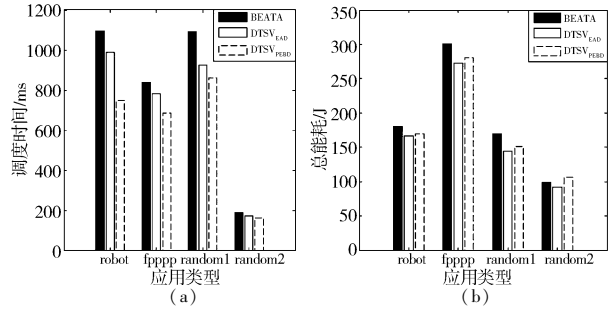

图2 节点数为64时,四种应用的调度长度和能耗

4.3.3 异构性的影响

衡量计算异构度对结果的影响时,选择了 fpppp 应用,计算结果如图3所示。

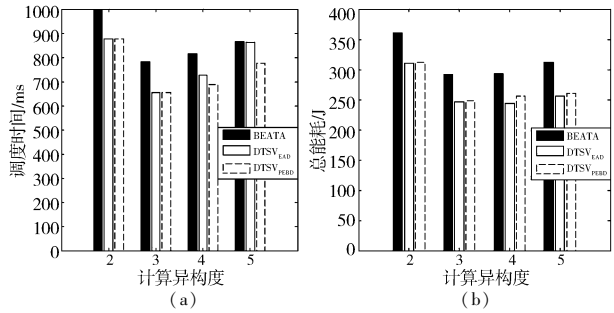

图3 不同计算异构度的调度长度和能耗

从图3中可以看出,DTSV 算法的调度长度明显小于 BEATA 算法,而且性能—能源平衡的方法要优于能源感知调度的结果。在能耗方面,DTSV 算法调度的总能源也明显地少于 BEATA 算法。fpppp 是计算密集型应用,可以看到当计算异构度为3时,调度时间最少,能耗也相对较小。

4.3.4 不同 CCR 的影响

通信计算比 CCR 对异构集群上并行应用调度性能影响的结果如图4所示,选择了合成应用 random2。

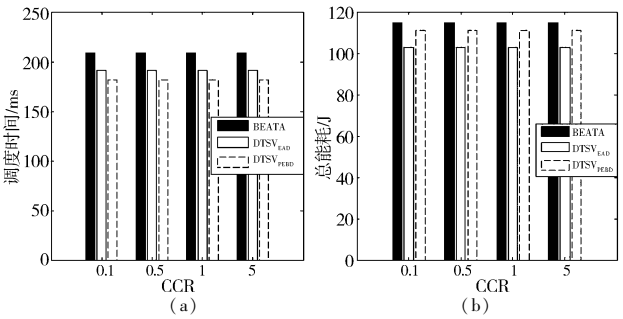

图4 不同 CCR 的调度长度和能耗

从图4中可以看出,不同的 CCR 对算法的性能没有任何影响,意味着异构集群对通信密集型应用和计算密集型应用的处理性能相当。同时 DTSV 算法的性能要优于 BEATA 算法,但在调度时间上,性能—能源平衡的策略要优于能源感知策略,而在能耗方面后者要优于前者。

5 结束语

本文目标是对运行在异构集群上的并行应用提出一种调度算法,旨在不明显增加能耗的情况下,减少调度长度,甚至能同时减少调度长度和能耗,以提高异构集群处理并行应用的性能。为了实现这个目标,提出了一种对于 AOV 网型应用,基于复制的任务调度算法 DTSV,并建立了描述异构集群、带有优先约束的并行应用以及计算节点与互连的能源消耗的数学模型。本文选取了四种应用,包括实际应用和合成的并行应用,采用了真实的处理器和高速互连的能耗率,并进行了广泛的实验。实验结果表明,与已有的 BEATA 算法相比,DTSV 算法能明显地减少调度长度和能源消耗,可以有效地提高异构集群处理并行应用的性能并能改善能耗问题。DTSV 从能源感知和性能—能源平衡两个角度考虑任务复制,产生了两组不同的数据结果。分析发现,性能—能源平衡 DTSV 的调度长度要优于能源感知的 DTSV,但相对地,前者在能耗方面要劣于后者。可见,依靠任务复制减少调度长度的同时会相应增加能耗,则能在增加较少能耗的情况下明显减少调度长度或能同比减少调度长度和能耗的任务调度算法可以作为进一步的研究目标。

我国的数据中心在不断发展,处理复杂、动态、大数据的应用性能和能耗需要不断地探索和改善。从实际的应用出发,例如 Hadoop、MapReduce 也有相关的作业调度算法,但在大数据时代,可以探究能带来更好性能和更少能量消耗的算法。而任务复制问题是一个 NP 完全问题,研究者们也会不断地去改进基于任务复制的调度算法。

参考文献:

[1] 田宝华,蒋句平,李宝峰,等. 基于统一资源管理的超级计算机系统节能方案[J]. 计算机应用, 2012, 32(3):835-838.

[2] ZONG Zi-liang, QIN Xiao, RUAN Xiao-jun, *et al*. Energy-efficient scheduling for parallel applications running on heterogeneous clusters [C]//Proc of the 36th International Conference on Parallel Processing. 2007:19.

[3] 叶可江,吴朝晖,姜晓红,等. 虚拟化云计算平台的能耗管理[J]. 计算机学报, 2012, 35(6):1262-1285.

[4] GRUNWALD D, LEVIS P, FARKAS K I, *et al*. Policies for dynamic clock scheduling[C]// Proc of the 4th Symposium on Operating Systems Design and Implementation. 2000:73-86.

[5] LORCH J R, SMITH A J. Improving dynamic voltage scaling algorithms with PACE[J]. ACM SIGMETRICS Performance Evaluation Review, 2001, 29(1):50-61.

[6] BENINI L, BOGLIOLO A, MICHELI G D. A survey of design techniques for system-level dynamic power management[J]. IEEE Trans on Very Large Scale Integration Systems, 2000, 8(3): 299-316.

[7] XIE Tao, QIN Xiao. An energy-delay tunable task allocation strategy for collaborative applications in networked embedded systems[J]. IEEE Trans on Computers, 2008, 57(3):329-343.

[8] ZONG Zi-liang, MANZANARES A, RUAN Xiao-jun, *et al*. EAD and PEBD: two energy-aware duplication scheduling algorithms for parallel tasks on homogeneous clusters[J]. IEEE Trans on Computers, 2011, 60(3):360-374.

[9] 李新,贾智平,鞠雷,等. 一种面向同构集群系统的并行任务节能调度优化方法[J]. 计算机学报, 2012,35(3):591-602.

[10] GUNARATNE C, CHRISTENSEN K, NORDMAN B, *et al*. Reducing the energy consumption of Ethernet with adaptive link rate (ALR)[J]. IEEE Trans on Computers, 2008, 57(4):448-461.

[11] ZAMANI R, AFSAHI A, QIAN Ying, *et al*. A feasibility analysis of power-awareness and energy minimization in modern interconnects for high-performance computing[C]//Proc of the 9th IEEE International Conference on Cluster Computing. 2007:118-128.

[12] Mellanox Technologies Inc. Mellanox performance, price, power, volumn metric (PPPV)[R/OL]. (2004) [2012-03-15]. http://www.mellanox.com/products/shared/PPPV.pdf.

[13] SOTERIOU V, PEH L S. Dynamic power management for power optimization of interconnection networks using on/off links[C]//Proc of the 11th Symposium on High Performance Interconnects. 2003:15-20.

[14] Standard task graph set Web site[EB/OL]. [2013-03-15]. http://www.kasahara.elec.waseda.ac.jp/schedule/.