大型分布式计算中的分级节能调度

秦高德[1] 文高进[2]
(深圳职业技术学院计算机工程学院 深圳 518055)[1]
(中科院自动化研究所模式识别国家重点实验室 北京 100190)[2]

摘 要 随着云计算的快速发展,大型分布式计算被广泛应用。但是,其运行时的巨大能量消耗已经成为应用推广的难题。目前的节能研究主要提出通过调度来减少服务器的运行数量以节能,而没有考虑网络的能耗。提出的分级调度算法HAS(Hierarchical Scheduling Algorithm)针对各计算节点间可能出现任务调度的情况,以DMNS(Dynamic Maximum Node Sorting)调度方法将这些应用尽量分配到连接到同一级交换机的服务器中,然后,将应用数量少的计算节点上的任务转移到还能增加任务的节点,从而减少节点的数量。同时,调度时选择的是较少的数据交换量和较短的交换路径,以节约网络能耗。HAS算法的复杂度较好,且其稳定性也通过计算仿真得到验证。通过仿真数据对比表明,HAS比目前的其它方法更优。

关键词 分布式计算,节能调度,HAS,KMNS,DMNS
中图法分类号 TP302.7 **文献标识码** A

Hierarchical Scheduling of Large Scale Distributed Computation

QIN Gao-de[1] WEN Gao-jin[2]
(School of Computer Engineering, Shenzhen Polytechnic, Shenzhen 518055, China)[1]
(National Laboratory of Pattern Recognition, The Institute of Automation of the Chinese Academy of Sciences, Beijing 100190, China)[2]

Abstract With the rapid development of cloud computing, large scale distributed computation is used widely. However, recently, energy consumption of such distributed systems has become problem for further application. Existing methods for saving energy are developed mainly by decreasing the amount of running servers. However, these approaches do not consider the energy cost on network devices. This paper proposed a hierarchical scheduling algorithm. Our algorithm employs a dynamic maximum node sorting (DMNS) method to optimize the assignment of applications on servers which are connected to a switch in the same level. Secondly, we transfered the applications on the nodes with low load to the nodes which can handle more application in order to reduce the number of nodes. In addition, we chose the transfer path which bears less capacity of data exchange and less length which helps to reduce the energy consumption of network. As a result, both the running servers and the data transfer can be greatly reduced. The time complexity of HSA is satisfactory, and its stability is verified through simulations. Experimental results show that the performance of HSA outperforms existing methods.

Keywords Distributed computing, Energy-saving schedule, HAS, KMNS, DMNS

1 引言

现代巨型高速计算机系统,包括科学计算中心、云计算平台以及大型企业计算系统,使用了数以万计的计算节点,有数以十万计的计算内核。在将来随着追求更快速度的超型计算机的建设,计算节点和计算内核的数目还将以成倍的方式增长。这些巨型计算机系统运行和冷却时需消耗巨大的电力[1]。

作为一种有效的节能措施,在计算节点间进行任务调度的方法被广泛使用。通过调度,使任务在尽可能少的服务器中运行并使空闲的服务器处于休眠状态甚至关机,这能明显减少能源消耗[1-5]。不过,目前已有的任务调度研究基本都未考虑减少调度时网络传输的能源消耗因素。然而,在整个大规模计算机系统中,虽然建设的都是能提供高速对分宽带的数据传输网络,但计算任务通常只利用到了部分网络以及较低的网络传输能力。网络的使用情况依赖于数据转发策略,在任务调度时如果考虑网络的基础连接情况,可以减少能源的消耗。

到稿日期:2012-06-20 返修日期:2012-09-15 本文受国家高科技研发863计划(2009AA01A129-2),广东省科技计划(2010A090100028,201120510102),国家自然科学基金(60903116),中国科学院知识创新计划(KGCX2-YW-131)以及深圳市科技计划(JC200903170443A,ZD201006100023A,ZYC201006130310A)资助。

秦高德(1974-),男,硕士,讲师,主要研究方向为分布式计算、计算机网络应用,E-mail:qingaodesz@163.com;文高进(1978-),男,博士,高级工程师,主要研究方向为高性能计算、图像处理。

在基于节能的计算任务调度过程中，本文进一步考虑减少网络能耗，提出了分层调度算法 HAS(Hierarchical Scheduling Algorithm)，通过调度减少计算节点数来节能，同时采用优化策略来减少调度时在网络设备上产生的能耗。

2 相关工作

在集群计算环境中，服务器中的计算任务通常采用对象驱动的方式，所有的服务器全部被保持在工作状态，并且计算任务被分散到尽可能多的服务器中，以提供最大的计算能力。这种操作模式会使得集群计算环境中设备的利用率较低，因而能耗效率低。已有研究指出，使用负载调度算法动态地调节，使空闲服务器进入节能模式，能显著地减少电力的消耗[1-3,12]。

负载调度算法广泛地以在线装箱问题(Bin Packing Problem，BPP)来进行研究[4,6-8,10]。每台服务器被看作是一个箱子，拥有可用的资源，如 CPU、内存和网络带宽等。计算任务被看作是一个需装箱的物品，重点考虑其计算资源需求。调度算法需确定在要求时间内完成任务的情况下，用最少的计算服务器数量来并发运行这些计算任务。在服务器间预先完成应用实时转移的调度，再来完成计算任务的策略，有助于显著提高资源的利用率。这种策略对于节能研究也是非常重要的，是绿色计算的一个常用技术[1,9]。

3 问题和方法

在本节中，针对节能提出了一个调度算法模型，并为调度算法的优劣提出了评估依据。

3.1 问题描述

输入：要完成计算任务的应用列 $A=(a_1,a_2,\cdots,a_m)$，准备参与计算的计算节点列 $Nd=(node_1,node_2,\cdots,node_n)$。其中，$a_i$ 表示第 i 个应用所需的资源(内存、CPU 和 I/O 等)，假定 $0<a_i\leqslant 100$。

如图 1 所示，各计算节点间常通过高速的网络交换设备相连。2 个计算节点间在交换数据时，连接的交换设备数量将会影响能量的消耗多少，交换设备数越少，消耗的能量也越少[11]。

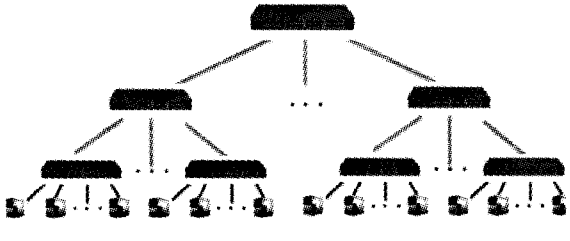

图 1 分布式计算中的网络结构

输出：应用列 A 按调度算法被调度到 Nd 的一个子集中，以减少最终参与计算的节点数，同时在调度过程中尽量减少网络中所产生的数据转移能耗(转移的数据量少，且经过的交换设备数少)。

3.2 节点状态描述

以整型向量 $S^t=(s_1^t,s_2^t,\cdots,s_m^t)$ 表示应用列 A 被分配的计算节点位置，其中 $s_i^t\in S$ 表示应用 a_i 在 t 时间段内被分配到节点 $node_{s_i^t}$，S^0 代表 A 的初始位置。现简单地用 $S=(s_1,s_2,\cdots,s_i)$ 表示 A 的节点列位置，其中 $1\leqslant s_i\leqslant n$，$s_i$ 表示第 i 个应用将从原始位置 $node_{s_i^0}$ 调度转移到 s_i 节点。

3.3 目标函数

调度算法由以下 2 个公式来评价其调度效果。

(1)节点数

$$NU_{cost}=\|S\|$$

式中，$\|S\|$ 表示计算节点向量 S 中被应用列 A 使用的个数，显然节点数越少，能耗越小。

(2)数据转移能耗

为减少计算节点数而进行调度任务时，数据转移能耗由原始节点到目标节点间的数据转移量和经由交换设备时的网络能耗量决定。假定 $P=\{p_{i,j}\}$，$0\leqslant i,j\leqslant m$ 为网络能耗总量，其中 $p_{i,j}$ 表示一个单位的应用从 i 节点转移到 j 节点时的网络能耗。则数据转移能耗可由下面公式表示：

$$DT_{cost}(s)=\sum_{i=1}^{m}\|a_i*\sigma(s_i-s_i^0)*p_{s_i^0,s_i}\|$$

式中，s_i^0 表示应用 a_i 所在的初始计算节点，s_i 表示 a_i 所去的目标节点，且规定 $\sigma(x)=0(x=0)$，$\sigma(x)=1(x\neq 0,x\in\mathbf{R})$。

4 调度算法框架

我们的调度算法策略由 2 部分构成：节点子集内的调度和分级调度。

4.1 节点子集与节点层次

这里所说的计算节点子集应满足一个条件：在子集中任意 2 个节点间进行任务转移调度时的路由能耗是相同的，比如它们都直接连接到同一交换机，在图 2 中列出了 2 种不同的节点子集。

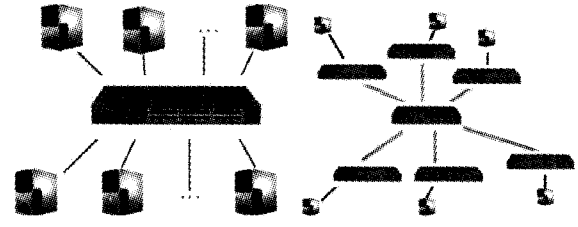

图 2 2 种节点子集

用子集内任意 2 节点间的任务调度平均能耗等级作为节点列中各子集的等级标识，显然，级别越低，其平均能耗越小。图 2 中的 2 种节点子集的等级标识不一样，在图 3 中列出了节点等级的一种区分方法。

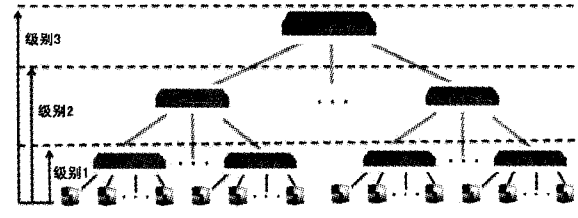

图 3 一种节点等级区分法

4.2 节点子集内的调度

在一个节点子集内进行节能调度算法研究时，考虑到子集中的任意 2 节点间在进行任务转移调度时的能耗是一样的，因此节能调度简化为如何减少参与运算的节点数和调度时产生的数据总量。显然，被转移的节点中包含的数据量越

小越好，所以包含较多的应用任务数据量的节点应该保持不动，且用来接收从其它节点转移来的任务。子集内的调度方法将在第5节详细描述。

4.3 分级调度

在各子集内的任务调度结束后，所有节点按事先约定的阈值被分为3类：Nd_{empty}、Nd_{unfull}和Nd_{full}。Nd_{empty}中节点原先分配的任务已全部被移走，暂定不再参与计算以节能；Nd_{full}中节点所包含的任务量都大于阈值，在将来不再参与调度而直接参与计算；Nd_{unfull}中节点则分配有任务但还可以接收更多的任务或其任务将被转移，在将来还需进一步调度。接着，将Nd_{unfull}中节点按能耗等级分成若干子集，在每一子集内再次进行类似的任务调度。如此，这样一种分级调度不断进行，直到几乎所有节点要么没有被分配任务，要么是"满"任务状态。详细的分级调度方法将在第6节说明。

5 子集内的调度

用Nd^f代表某节点子集内安排用来接收任务的那些节点，而其余需转移任务的节点列假设为Nd^t。

5.1 递增取 K 调度法(KMNS, the K-th Max Node Sorting Method)

按各节点初始分配的计算任务量将节点列进行升序排列，然后取前K个节点列为Nd^t，其余的为Nd^f，最后用降序最佳适配法BFD(Best Fit Descending)将任务从Nd^t转移到Nd^f(在装箱问题的调度算法中，BFD比其他适配法更优[10])。

具体来说，在转移过程中，Nd^t中节点按任务量进行降序排列，然后以最佳适配方法BF(Best Fit)按任务量从大到小的顺序将任务分别从Nd^t转移到Nd^f。如果Nd^t有一节点的任务在Nd^f无法找到适配的接收节点，则在Nd^f中插入一个新的空节点来接收。

最后，求出参与计算的节点数和转移的数据总量。

子集内调度的算法过程如下。

算法1 递增取K调度算法KMNS

输入：具有N个节点的子集Nd^s。

输出：节点列B，满足最后参与计算的节点数较少，且转移任务产生的数据量较小。

过程：

(1)根据Nd^s中每个节点被分配的任务量将其按升序排序，得到节点列$Nd^{s'}$。

(2)根据预设的K值，取$Nd^t=Nd^{s'}_{i=1,\cdots,K}$，$Nd^f=Nd^{s'}_{i=K+1,\cdots,N}$。

(3)采用递减最佳适配法BFD将任务从Nd^t转移到Nd^f。

(4)找到Nd^f与Nd^s间的对应关系。

(5)计算最后所得Nd^f的节点数NU^K_{cost}和转移数据总量DT^K_{cost}。

性质1 如果$K=N$，则第(1)步中就不需对节点列进行升序排序，这里的KMNS算法简化成了BFD算法。

性质2 有$NU^K_{cost}\geqslant N_{opt}$。其中最佳节点数$N_{opt}$可由式(1)求出：

$$N_{opt}=\left\lceil\frac{TA}{ACN}\right\rceil \tag{1}$$

式中，TA代表此节点子集总的任务量，ACN代表节点的平均接收能力。由式(1)可知，最少N_{opt}个节点就可以接收节点子集的所有任务。

引理1 在节点子集中，假如任务总量为T，节点总数为N，用来接收任务的节点数为NU^K_{cost}，如果$K\leqslant N-NU^K_{cost}$，则转移数据量的估计值为$DTE(K)=T-\frac{NU^K_{cost}*T}{N}$。

引理2 如果转移的数据总量为DT^K_{cost}，则有

$DT^K_{cost}\leqslant DTE(K)$

引理2给出了在最佳调度算法中转移数据量的上界值。

5.2 KMNS的性能

现引入以下3个度量，来研究KMNS调度算法的性能。

(1)节点使用率：$RoNU(K)=NU^K_{cost}/N$

显然，NU^K_{cost}比N小而比N_{opt}大，所以有$\frac{N_{opt}}{N}\leqslant RoNU(K)\leqslant 1$。

(2)数据转移率：$RoDT(K)=DT^K_{cost}/T$

在调度中需转移的数据DT^K_{cost}，最佳的情况是0，即无数据需转移，而最坏的情况是$\frac{N-1}{N}*T\to T(N\to+\infty)$，即几乎所有数据都进行了转移，所以有$0\leqslant RoDT(K)<1$。

(3)预估数据转移率：$RoDTE(K)=\frac{DTE(K)}{T}=1-RoNU(K)$。

由引理1可知，如果$K\leqslant N-NU^K_{cost}$，则有$RoDTE(K)\geqslant RoDT(K)$。

随着输入节点数N的不同，在图4和图5中分别显示了KMNS的性能情况。由图中可以看出，随着K取值的增大，$RoNU(K)$逐渐变小，但不小于$\frac{N_{opt}}{N}$，而$RoDT(K)$则逐渐增大。当K增大到某个值时，$RoNU(K)$与$RoDT(K)$的综合值将得到最佳结果。

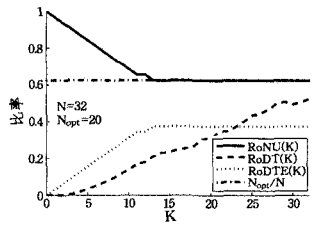

图4 KMNS的性能曲线(节点数$N=32$)

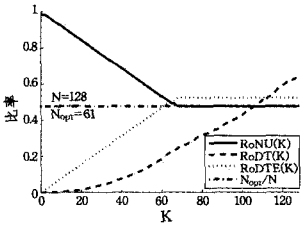

图5 KMNS的性能曲线(节点数$N=128$)

5.3 动态取 K 调度算法(Dynamic Max Node Sorting Method, DMNS)

为了让参与计算的节点最少和被转移的数据量最小，提出了一种动态取K调度算法，如算法2。

算法2 动态取K调度算法

输入：初始节点列$A=\{a_i\}^N_{i=1}$

输出：节点列C，满足最少的计算节点NU_{cost}和最小的数据传输量DT_{cost}

初始化：$NU_{cost}=N$，$DT_{cost}=+\infty$，C=null

主程序：

对于节点列A中的所有应用使用BFD调度算法，最后计算NU^{BFD}_{cost}。

For K = 0 to N do

应用KMNS调度算法，计算NU^K_{cost}、DT^K_{cost}，得到临时节点列B。

if $NU_{cost}>NU^K_{cost}$ then

$NU_{cost}=NU^K_{cost}$，$DT_{cost}=DT^K_{cost}$，C=B

else if $NU_{cost}==NU^K_{cost}$ then

if $DT_{cost}>DT^{K}_{cost}$ then

$DT_{cost}=DT^{K}_{cost}$, C=B

end if

end if

end for

由性质 1 得知，BFD 是 DMNS 的一种特殊情况，故有 $NU_{cost}\leqslant NU^{BFD}_{cost}$。由文献[10]得知，$BFD(L)=\frac{11}{9}OPT+4$。DMNS 的上界可由以下的引理来计算。

引理 3 $DMNS(L)=\frac{11}{9}OPT+4$

6 分级调度

在算法 3 中给出了分级调度算法的计算过程和结果。

算法 3 分级调度算法(Hierarchy Scheduling of Applications, HAS)

输入：节点列 $A=\{a_i\}^{N}_{i=1}$，数据转移价值矩阵 CM

输出：满足最少的计算节点和最少的数据转移能耗的节点列 C

初始化：级别 l=1，阈值 y=0.95，C=null

主程序：

While A != null do

根据数据转移价值矩阵 CM 将 A 分解为子集 $\{B_i\}^{N_l}_{i=1}$

for i=1:N_l do

使用算法 2(DMNS 算法)对子集 B_i 进行调度

根据阈值 y 计算 Nd_{unfull} 和 Nd_{full}

将 Nd_{full} 从 A 中移除，$C=C\cup Nd_{full}$，

end for

级别 l=l+1

end while

(1)复杂度

从总体上来看，分级调度算法在调度时，在同一级别将节点列先拆分成子集，然后对子集内节点随着级别增加来调度。在有 n 个节点和交换机为 16 口的调度中，最大的级别是 $\log_{16}n$，在级别 m 时，有 $\frac{n}{(16*m)}$ 个子集。假设算法 2(DMNS)的时间复杂度是 $\Theta(1)$，则算法 3(HAS)的时间复杂度是 $\sum_{m=1}^{\log_{16}n} n/(16*m)=\Theta(n*\log(\log n))$。

(2)稳定性

如果某节点的应用发生改变，则此节点所在的子集被重新调度且重新产生一个新的 Nd_{unfull}，从级别 2 到 n 的调度也将重新进行。分级调度算法的稳定性可由本地数据转移比率来评估：

$$RoLDT=Ns*L/T+DT_{2-n}/T$$

式中，T 代表节点列的应用数据总量，L 代表节点的最大数据装载量，Ns 代表参与转移的节点数，DT_{2-n} 代表从第 2 级到最后一级调度过程中产生的数据转移总量。不同的参与转移的节点数 Ns 导致不同的调度稳定性，当 Ns 递增时，DT_{2-n} 将递减。当 Ns 与 DT_{2-n} 达到一个较好的平衡时，$RoLDT$ 将得到一个较小的值。$RoLDT$ 越小，分级调度算法越稳定，当改变某节点的应用时，所带来的调度结果摆动情况也更小。

7 仿真实验和讨论

我们将分级调度算法 HAS 用 C++语言予以编程实现，并在 P-IV 2.8GHz CPU 和 2GB 内存的 PC 机上测试。数据转移价值矩阵 CM 根据节点间的距离和网络连接情况来设定。2 种输入节点的取值都在 0 到 100 之间，分别由一致分布和正态分布(零平均值和标准差为 0.3)来随机产生。

7.1 KMNS 调度的性能

分别以 32、64、128 和 256 个节点来测试 KMNS 的调度情况，32 和 128 个节点的评估结果见图 4 和图 5。可见，$RoNU(K)$ 随着 K 的增大而减小，但不小于 N_{opt}/N，但 $RoDT(K)$ 却随着 K 的增大而增大。

7.2 DMNS 调度的对比

我们也引入 DMAS(Dynamic Max Application Sorting[10])和 BFD(Best Fit Descending)来与 DMNS 进行对比。在 DMAS 中，节点只根据节点所装载应用数量的最大比率(未考虑数据转移的能耗)排序。表 1 显示了 DMNS、DMAS 和 BFD 这 3 个调度在输入节点的数据量以正态分布随机产生时的调度结果，表 2 则显示了在一致分布时的调度结果。由表 1 和表 2 可知，3 种调度在最后都能使节点数达到相同的最小值。而由图 6 可见，DMNS 产生的数据转移量也是最少的。

表 1 以正态分布随机产生节点列时的性能对比

N	项目	DMNS	DMAS	BFD
32	Nd_{cost}	16.58	16.58	16.58
32	DT_{cost}	599.77	659.01	1073.85
64	Nd_{cost}	32.26	32.26	32.26
64	DT_{cost}	1194.32	1318.99	2176.14
128	Nd_{cost}	63.82	63.82	63.82
128	DT_{cost}	2427.37	2712.19	4438.20
256	Nd_{cost}	127.23	127.23	127.23
256	DT_{cost}	4853.39	5366.64	8887.95

表 2 以一致分布随机产生节点列时的性能对比

N	项目	DMNS	DMPS	BFD
32	Nd_{cost}	16.16	16.16	16.16
32	DT_{cost}	394.46	440.73	932.07
64	Nd_{cost}	31.72	31.72	31.72
64	DT_{cost}	806.39	887.54	1909.15
128	Nd_{cost}	63.64	63.64	63.64
128	DT_{cost}	1607.06	1800.07	3929.89
256	Nd_{cost}	127.27	127.27	127.27
256	DT_{cost}	3237.21	3616.10	7943.63

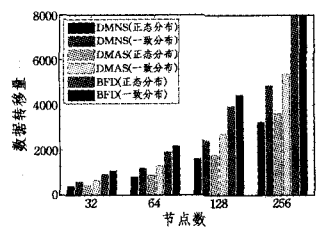

图 6 HAS 调度按节点数时的数据转移量对比

7.3 HAS 调度的性能

7.3.1 最小的数据转移量

为测试算法 3 中实现的基于 DMNS 的 HAS 调度算法，仍然引入 DMNS、DMAS 和 BFD 调度，用满足正态分布和一致分布的方式产生 4096 个节点中的数据，并分别用于 32、64、128、256 口的交换机。经测试发现，以上 3 种调度都可以得到相同的最少节点数，而不同的数据转移量对比如图 7 所示。

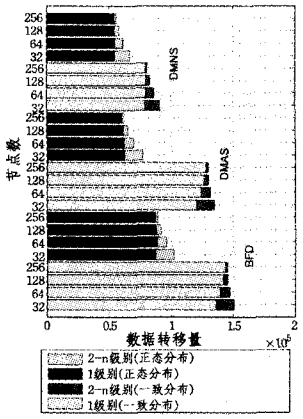

图 7 HAS 调度按级别时的数据转移量对比

从总体上看，DMNS 在数据转移量上仍然最少。也可以看出，在级别 1 产生的数据转移量比级别 $2-n$ 要大得多，且正态分布时的数据转移量比标准分布情况下要多。但在级别 $2-n$，分别在 2 种分布之间比较以及在 3 种调度之间比较，所产生的数据转移量却差不多。对于 3 种调度，交换机端口数越多，级别 $2-n$ 中产生的数据转移量越少。

7.3.2 稳定性对比

基于 DMNS、DMAS 和 BFD 这 3 种调度的 HSA 的稳定性比较如图 8 所示。可见随着交换机端口数的变化，稳定性也不同。当端口数为 64 时，3 种调度的稳定性都达到最好。总之，当端口数为 64 时基于 DMNS 的 HAS 具有最佳稳定性。

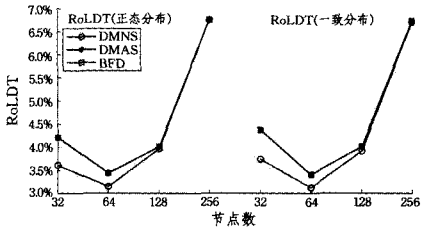

图 8 HAS 调度的 RoLDT 对比

结束语 在大型分布式计算中心的应用调度时，为了使参与计算的服务器最少和产生的数据转移量最少，我们提出了 DMNS 调度方法。在 DMNS 的基础上进一步提出的 HAS 调度算法稳定性好，时间复杂度为 $\Theta(n * \log(\log(n)))$。将 HAS 的性能与已有类似的调度算法进行仿真比较，结果显示其在稳定性和能耗方面都有占优。

在将来的调度算法研究工作中，将考虑更多节能方面的要素。例如在大型分布式计算中心每个计算节点的降温模式不相同，转移应用时，如果选择温度更低的节点，也将节约能耗和提高计算稳定性，这就成为有温度介入的调度算法研究。

参 考 文 献

[1] 林伟伟，齐德昱. 云计算资源调度研究综述[J]. 计算机科学，2012，39(10)：1-6

[2] Orgerie A-C, Lefevre L, Gelas J-P. Demystifying energy consumption in Grids and Clouds[A]// Green Computing Conference, 2010 International, 2010[C]. Chicago, IL: IEEE Conference Publications, 2010: 335-342

[3] Chase J S, Anderson D C, Thakar P N, et al. Managing energy and server resources in hosting centers[A]// Proceedings of the eighteenth ACM symposium on Operating systems principles (SOSP'01), 2001[C]. New York, NY, USA: ACM, 2001: 103-116

[4] Yao C-C. New algorithm for bin-packing[J]. Journal of ACM, 1980, 27(2): 207-227

[5] Verma A, Ahuia P, Neoqi A. Power-aware dynamic placement of HPC applications[A]// Proceedings of the 22nd annual international conference on Supercomputing (ICS'08), 2008[C]. New York, NY, USA: ACM, 2008: 175-184

[6] Johnson D. Near-Optimal Bin Packing Algorithms [M]. MIT, Cambridge, Massachusetts, 1973: 56

[7] Gambosi G, Postiglione A, Talamo M. Algorithms for the relaxed online bin-packing model[J]. SIAM Journal on Computing, 2000, 30(5): 1532-1551

[8] Sanders P, Sivadasan N, Skutella M. Online Scheduling with Bounded Migration[J]. Mathematics of Operations Research, 2009, 34(2): 481-498

[9] Srikantaiah S, Kansal A, Zhao Feng. Energy aware consolidation for cloud computing[A]// Proceedings of the 2008 conference on Power aware computing and systems (HotPower'08), 2008 [C]. Berkeley, CA, USA, USENIX Association, 2008: 10

[10] Johnson D S, Demers A, Ullman J D. et al. Worst-case Performance Bounds for Simple One-Dimensional Packing Algorithms [J]. SZAM Journal on Computing, 1974, 3(4): 299-325

[11] Baliga J, Ayre R W A, Hinton K. et al. Green Cloud Computing: Balancing Energy in Processing, Storage, and Transport[J]. Proceedings of the IEEE, 2011, 99(1): 149-167

[12] Younge A J, von Laszewski G, Wang Li-zhe, et al. Efficient resource management for Cloud computing environments[A]// Proceedings of the International Conference on Green Computing (GREENCOMP'10), 2010[C]. Washington, DC, USA, IEEE Computer Society, 2010: 357-364

(上接第 63 页)

[8] Pinheiro E, Bianchini R, Carrera E, et al. Load balancing and unbalancing for power and performance in cluster-based systems [R]. DCS-TR-440. Department of Computer Science, Rutgers University, May 2001

[9] Gandhi A, Harchol-Balter M, Kozuch M A. The case for sleep states in servers [C]// Proceedings of the 4th Workshop on Power-Aware Computing and Systems (HotPower'11). Cascais, Portugal, 2011: 6-10

[10] Horvath T, Skadron K. Multi-mode Energy Management for Multi-tier Server Clusters[C]// Proceedings of the 17th International Conference on Parallel Architecture and Compilation Techniques (PACT'08). Toronto, Canada, 2008: 270-279

[11] Xue Zheng-hua, Dong Xiao-she, Ma Si-yuan, et al. An energy-efficient management mechanism for large-scale server clusters [C]// Proceedings of the 2007 IEEE Asia-Pacific Services Computing Conference (APSCC'07). Tsukuba Science City, Japan, 2007: 509-516

[12] Feitelson D. Parallel Workloads Archive [OL]. http://www.cs.huji.ac.il/labs/parallel/workload/l_anl_int/ANL-Intrepid-2009-1.swf.gz, 2011