高技术通讯 2012 年 第22 卷 第11 期:1143 ~ 1148

doi:10.3772/j.issn.1002-0470.2012.11.006

负载驱动的机群系统动态功耗管理研究①

梁爱华②[*,**] 肖利民③[*] 庞 瑜[*] 李勇男[*] 阮 利[*]

([*]北京航空航天大学计算机学院 北京 100191)

([**]北京联合大学计算机技术研究所 北京 100101)

摘 要 为了通过动态功耗管理方法解决高性能计算机系统功耗与性能之间的权衡问题,提出了一种适用于同构机群环境下负载驱动的动态功耗管理(WDPM)策略。该 WDPM 策略根据负载随时间变化的不均衡性,通过分析并行负载日志的到达特征,在改进超时动态功耗管理的基础上运用了基于负载预测的预唤醒方法和实时反馈修正机制。以真实负载日志和实际系统参数为源数据进行了模拟实验,而且做了 WDPM 策略与超时策略的对比测试。实验结果表明,WDPM 策略能在增加极少量功耗的情况下,有效减少作业平均等待时间和系统中节点状态切换次数,能更好地获得性能与功耗间的平衡。

关键词 动态功耗管理(WDPM),负载驱动,预唤醒,反馈机制,机群

0 引 言

在高性能计算领域,计算机群已被广泛用于多领域的大规模科学计算。长期以来,计算机性能一直应用中优先考虑的指标[1]。然而,随着系统规模的增大功耗急剧增大[2],功耗已成为制约高性能计算机系统发展的主要瓶颈之一[3,4],因此,功耗也就成为衡量高性能计算机的另一个重要指标,动态功耗管理必须进行。动态功耗管理是一种根据负载变化选择性地设置系统低功耗状态部件或关闭部分系统部件,以最小的活动部件数目或最小的部件功耗提供系统所需的服务和性能级别的方法[5]。这种管理最早被应用于单机系统中[6,7]的功耗管控。有效运用动态功耗管理的一个重要方面是合理选择系统中节点状态的调整时机。机群系统中,负载大都并行作业,并且负载量以及对节点的需求都随时间而变化,针对单个服务器的方法已不适用。而目前已有研究大都基于超时调整策略,未考虑负载动态变化带来的作业延迟的影响。此外,不合理的频繁开关机会导致更大的能耗损失[8]。因此,设计一种可以根据系统负载变化进行自适应调整的动态功耗管理方法很有必要。本文在对系统负载日志进行统计分析的基础上对超时策略进行了改进,提出了一种负载驱动的动态功耗管理(workload-driven dynamic power management, WDPM)策略。该策略在对负载进行建模的基础上采用了预唤醒方法,以减少由于节点状态切换导致的作业延迟。为使动态功耗管理能适应系统需求的动态变化,加入了反馈修正机制,以动态修正负载模型。模拟实验结果表明,该策略可以有效降低超时策略引起的作业延迟,并减少系统中节点状态的切换次数。

1 相关工作

广义上讲,动态功耗管理可在 CPU、单机系统和机群三个级别上实施[1]。CPU 级动态功耗管理也称为动态电压调节(dynamic voltage scaling, DVS),它被普遍运用在实时系统中。单机系统级动态功耗管理是指对选择系统部件功耗状态转换时机和选择何种状态做出决策,主要分为超时策略[9]、基于预测的启发式策略[10]和基于随机过程的优化策略[9,11]。在机群系统级,则是粗粒度地调节处于活动状态的服务器数量,即关闭系统中某些空闲的节点或将其调整至低功耗状态,以适应变化的系统

① 863 计划(2011AA01A205),国家自然科学基金(61003015),教育部博士点课题(20101102110018)和北京市自然科学基金(4122042)资助项目。

② 女,1979 年生,博士生,讲师;研究方向:高性能计算;E-mail: liangah@cse.buaa.edu.cn

③ 通讯作者,E-mail: xiaolm@buaa.edu.cn

(收稿日期:2011-12-26)

需求[12]。考虑到本文主题是机群系统的动态功耗管理,以下只对机群系统级相关研究做一简要描述,以利于对本研究的理解。

Elnozahy 等[13]对服务器机群采用的 5 种功耗管理策略进行了评价,这 5 种策略是:独立电压调节(independent voltage scaling, IVS)策略;协同电压调节(coordinated voltage scaling, CVS)策略;开关机(vary-on/ vary-off, VOVO)策略,IVS 和 VOVO 相结合的策略;CVS 和 VOVO 相结合的策略。文中对这些方法进行了比较分析。Dolz 等[14]提出了根据过去和未来用户请求数进行节点开关控制的软件模块,通过对节点空闲时间和等待作业数规定阈值共同控制节点的开关,等待作业数阈值可以在一定程度上反映负载量的变化,但不能预测负载的变化,负载量增大时的节点启动可能造成作业执行的延迟。Hu 等[15]提出了动态功耗管理和无线传感网络监控相结合方法的功耗与环境感知模块,它根据无线传感器监控温度情况设定节点超时阈值来控制节点的开关,由于超时阈值固定,会造成节点开启延迟,影响作业执行。文献[16]提出了一种可扩展的机群系统管理架构,不同状态的节点被划分到不同的节点池,并规定了每个节点池的最大节点数,提出可根据系统负载的自适应节点池转换机制,但未给出具体的实现方法。文献[17]提出,在负载较轻时,可以将负载整合到少数节点上运行,然后将多余的空闲节点关闭以节能。控制方法也是基于节点空闲的时间阈值。文献[18]提出功耗管理不能只关心功耗的降低,必须关注由此带来的性能影响,要同时满足能耗和服务质量(QoS)的双重要求,权衡好功耗与性能之间的关系。

本文通过对真实负载日志的分析,揭示了系统负载到达的统计特征,通过历史数据确定负载模型,并结合预唤醒方法和反馈修正机制,进行动态功耗管理。实证研究表明,负载驱动的动态功耗管理方法可以根据系统负载变化规律调整节点状态。相对于超时策略,能有效减少作业的平均等待时间和系统中节点切换次数。从而降低功耗控制同时导致的性能损失。

2 负载特征

机群系统中,负载是随时间变化的。对负载特征进行分析,并找出其变化规律是进行有效动态功耗管理的基础。本研究从希伯来大学并行系统实验室提供的并行工作负荷档案[19]中的负载日志入手对负载到达特征进行了分析。

2.1 负载到达模式分析

负载到达通常具有三个级别的周期性,分别是日循环(晚上很少工作)、周循环(周末很少工作)及年循环(节假日很少工作)[20],其中影响最大的是日周期。本文对负载特征的分析涉及日周期和周周期。

日循环曲线表示一天 24 小时中每个时段作业到达和处理器需求的变化。以 4 个系统的负载日志(OSC- Cluster、SDSC-BLUE、CTC-SP2、LLNL-Altas[19])为例,尽管各系统的负载变化各有不同,但有以下两个共同点:(1)不同用户提交作业时间具有不同特点,但绝大多数仍集中在正常工作时间内,遵守工作时间的活动周期;(2)由于科学计算中并行作业计算量较大,所占用时间较长,其负载变化曲线大都具有较长的尾巴,即负载量上升迅速,但下降较为缓慢。

目前国外已有大量对机群负载特征的研究[20,21],对负载到达的日周期,通常是从现有的统计分布中选择最适合的分布进行描述,常用的备选分布包括两阶段均匀分布、指数分布、超指数分布、伽马分布和超伽马分布。若将四、五点钟作为负载到达的最低点,并忽略某些系统中负载曲线的小波动,可将其归结为单峰分布,并结合并行负载的特点,最适合的是伽马分布[18]。因此,负载及其节点需求均可用伽马分布来表示,概率分布函数为

$$f(x;k,\theta) = x^{k-1}\frac{e^{-x/\theta}}{\theta^k \Gamma(k)} \quad (1)$$

其中 k 为形状参数,θ 为尺度参数。

负载抖动是指在相对较短的时间内大量作业到来的情况。应该指出的是,负载抖动可能是有意的或是错误导致的。为了整个系统的优化设计,这些不正常的负载抖动不应该被包括在负载模型中[21]。

周循环曲线则表示一周每天作业到达和处理器需求的变化,4 个系统中负载日志(OSC- Cluster、SDSC-BLUE、CTC-SP2、LLNL-Altas[19])中,在周一至周五的工作日中,负载具有不同特点且有波动,而在周末则相对于工作日有明显的下降。因此,本文对负载的周循环变化特征只做工作日和周末的区分。

2.2 负载模型确定

通过上节分析,负载的日循环特征用伽马分布表示。形状参数 k 和尺度参数 θ 需根据实际系统确定。为使模型适用具体系统,在系统实际运行过程

中,需要对负载信息进行收集,由此得到历史数据,包括各个时段的负载量、节点需求等。这些数据可作为确定负载模型的依据。日循环曲线的确定过程如图1所示。为保证数据能真实反映系统的使用情况,对收集的历史数据按照各时段计算得到的平均值作为采样数据。

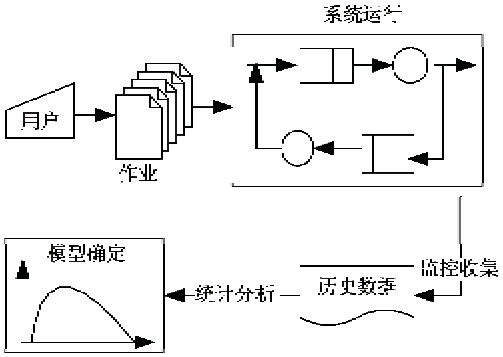

图1 日循环模型确定过程

采样数据确定后,需要确定伽马函数中的形状参数和尺度参数,并进行参数估计。本文采用统计学中的极大似然估计方法。对于伽马分布,首先得到24个独立同分布的随机变量(x_1, …, x_{24}),对应一天中各时段(每小时)的负载量,由此可得到其似然函数,进而得到其对数似然函数,最后求得尺度参数θ的极大似然估计为

$$\hat{\theta} = \frac{1}{24k}\sum_{i=1}^{24} x_i \tag{2}$$

对k没有精确解,可以通过牛顿迭代方法得到其近似解。k的初始值可以通过矩量法得到,令

$$s = \ln\left(\frac{1}{24}\sum_{i=1}^{24} x_i\right) - \frac{1}{24}\sum_{i=1}^{24}\ln(x_i) \tag{3}$$

得到k的近似值为

$$k \approx \frac{3 - s + \sqrt{(s-3)^2 + 24s}}{12s} \tag{4}$$

形状参数k和尺度参数θ确定后,即可得到日循环统计模型。对于周循环曲线,通过历史数据分别计算得到工作日和周末负载量的平均值($AvgN_{weekday}$和$AvgN_{weekend}$),得到周末相对于工作日的比例系数λ,即

$$\lambda = AvgN_{weekend} / AvgN_{weekday} \tag{5}$$

3 负载驱动的动态功耗管理

3.1 节点状态

服务器通常包含多种低功耗状态。例如,在高级配置与电源接口(advanced configuration and power interface, ACPI)中定义了三种节能方式,分别是挂起(Suspend)、挂起到内存(suspend to RAM, STR)、挂起到硬盘(suspend to Disk, STD)。按节能效果从小到大依次为Supsend、STR、STD。相反,唤醒所需时间依次增加。

本文中节点状态包含3个,分别是忙碌、空闲、休眠。忙碌表示节点处于运行状态。由于目前服务器均为多核处理器,本文设定忙碌是指处理器的每个核均满负荷运行。空闲表示节点没有运行作业。由于挂起(Suspend)只对显示屏断电,对无显示器的机群系统中的服务器并不适用。服务器一般采用挂起到内存(STR)或挂起到硬盘(STD)两种方式。STR需要硬件的支持,因此本文采用STD休眠方式。

为说明系统中节点状态转换关系,我们将与超时策略进行对比说明。图2(a)为超时策略转换图,图2(b)为负载驱动策略转换图。二者的区别在于空

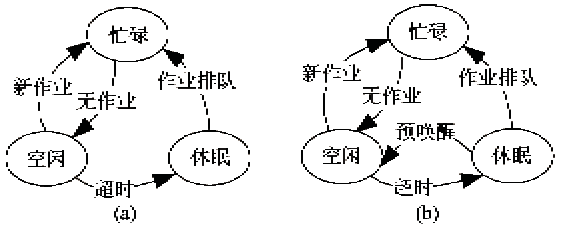

图2 超时和负载驱动策略的节点状态转换

闲与休眠的转换。采用超时策略时,若节点空闲时间达到设定阈值,则转为休眠状态,当由于资源不足作业排队时,休眠节点被唤醒。若用负载驱动策略,空闲节点在达到超时阈值时转为休眠。休眠状态的节点除作业排队时被唤醒外,还可能被预唤醒。即提前预测系统需求,当空闲节点不能满足系统需求时,则提前唤醒部分休眠节点。本文描述中所用的符号如表1所示。

表1 符号表示

符号	描述	符号	描述
φ	忙碌状态节点功耗	σ	空闲状态节点功耗
μ	休眠状态节点功耗	ω	唤醒节点所需功耗
ϕ	当前时间	τ	调度周期
ξ	当前空闲节点数	γ	处于唤醒状态的节点数
t	唤醒节点所用时间	T	超时阈值
λ	周循环系数		

超时策略对超时阈值的确定通常采用预先确定一组备选值,利用滑动窗口的方法进行调整。负载驱动策略的超时阈值确定方法与通常的超时策略有所不同。节点从休眠状态唤醒到空闲状态需要一定的功耗,唤醒一个节点所需功耗为 ω,休眠一个节点的功耗则可假定为 0[10]。从直观上讲,节点不应该处于空闲太长时间,因为这样会产生不必要的功耗。然而,如果空闲节点马上休眠则可能会由于不久又被唤醒而产生更大的功耗,因此,空闲节点应等待一段时间,以使其功耗等同于唤醒所需功耗[10],即可假定以下公式成立:

$$\sigma \cdot T = \omega \cdot i \tag{6}$$

由此求得 T 即定为本策略的超时阈值。

3.2 预唤醒

STD 所需的唤醒时间较长,一定程度上会导致作业延迟。因此,若根据负载变化曲线预测节点需求,并通过预唤醒方法提前唤醒相应数量的节点,可以减少作业延迟。

在每个调度周期,根据负载模型预测得到下个调度周期的节点需求,若当前节点数不能满足需求,则预唤醒相应数量的节点。预唤醒(Pre-wakeup)算法描述如下:

Pre-wakeup 算法

1. 获取当前处于空闲状态节点数 ξ 和唤醒状态的节点数 γ
2. 通过负载模型计算 $(\phi + \tau)$ 所需的空闲节点数,即 $\eta = \lambda \cdot f(\phi + \tau; k, \theta)$
3. 判断是否是周末,若是, $\eta = \lambda \cdot f(\phi + \tau; k, \theta)$
   If $(\xi + \gamma < \eta)$
   唤醒 $(\eta - \xi - \gamma)$ 个节点
   Else
   $(\xi + \gamma - \eta)$ 个节点进入超时队列
4. 结束

3.3 反馈修正

负载模型是对历史数据通过运用统计方法而确定的。在系统运行过程中,负载到达的特征可能发生变化,使实际情况与预先确定的负载模型产生误差。如果负载曲线在运行中始终不变,则可能导致误差越来越大。因此,为使负载模型自适应具体系统,本策略加入了反馈修正机制:在系统运行中实时收集负载数据,即在每个调度周期,收集数据并与预定的负载曲线比较,根据比较结果修正负载模型参数。修正过程如图 3 所示。

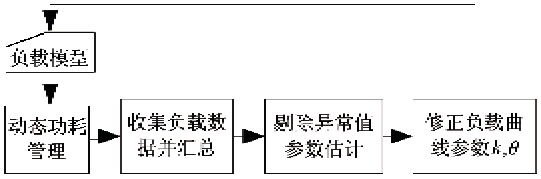

图 3 反馈修正过程

在反馈修正过程中,需要将负载抖动导致的异常值剔除,由于负载抖动是在短时间内发生的,时段设置为 30min。判断异常值采用格拉布斯检验法。反馈(Feedback)修正算法描述如下:

Feedback 算法

输入:负载 $(x_1 + x_2 + \cdots + x_{48})$,原参数 k, θ

输出:新参数 k, θ

1. 收集负载数据,并汇总每个时段数据
2. 检验统计的负载数据 $(x_1, x_2, \cdots, x_{48})$ 中是否存在异常值,若存在则剔除(格拉布斯检验法)

   计算统计量 $\mu = (x_1 + x_2 + \cdots + x_{48}) / 48$

   $s = (\sum (x_i - \mu)/(n-1))^{1/2} (i = 1, 2, \cdots, 48)$

   $G_i = (x_i - \mu)/s$

   确定检出水平 α

   If $(G_i > G_{1-\alpha}(48))$

   x_i 为异常值,将其剔除

   Else

   汇总每时段(每小时)负载数据

   确定负载分布的新参数

   修正参数 k, θ
3. 结束

4 模拟实验

本文利用C++实现了负载驱动的动态功耗管理(WDPM)。采用 4 个并行负载日志[17](OSC-Cluster, SDSC- BLUE, CTC-SP2, LLNL-Altas)的标准格式 CLEANED 版进行了模拟实验。4 个日志的信息如表2所示。实验中将前30天的负载日志作为

表 2 模拟所用的日志信息

日志	节点数	作业总数	30 天作业数	90 天作业数
OSC-Cluster	57	80714	1020	3537
SDSC-BLUE	144	250440	6430	19133
CTC-SP2	430	79302	6639	20254
LLNL-Altas	1152	60332	1326	9149

历史数据,用于负载模型建立。在此基础上完成30天日志和90天日志实验。

实验的指标包括性能和功耗两个方面。性能方面的指标是作业的平均等待时间,功耗方面是系统中节点平均功耗。另外测试中统计了节点的平均休眠/唤醒次数。节点在各种状态下的功耗值(φ,σ,μ,ω)等参数均以由功率表对8核的联想万全服务器R510测得的数据作为依据。本实验将WDPM策略和基本超时策略进行了对比。为保证同等条件下进行比较,设置相同的超时阈值,即按照式(6)计算得到的T。

表3和表4分别是30天负载和90天负载的模拟结果。其中包括WDPM策略与超时策略对比后的百分比。从结果来看,WDPM策略均很大程度地减少了作业的平均等待时间,从而减小了作业延迟。节点的平均休眠/唤醒次数也均有一定程度的降低。

表3 30天负载实验结果

日志	策略	节点平均功耗(kW)	作业平均等待时间(s)	节点平均休眠/唤醒次数(次)
OSC-Cluster	超时	56.0	60	31
	WDPM	58.4	39	30
	对比	+4.3%	-35%	-3.2%
SDSC-BLUE	超时	501.1	67	515
	WDPM	503.9	19	458
	对比	+0.5%	-71.6%	-11%
CTC-SP2	超时	471.7	29	448
	WDPM	475.0	10	391
	对比	+0.7%	-65.5%	-12.7%
LLNL-Atlas	超时	99	52	36
	WDPM	101.2	36	35
	对比	+2.2%	-30.8%	-2.8%

表4 90天负载实验结果

日志	策略	节点平均功耗(kW)	作业平均等待时间(s)	节点平均休眠/唤醒次数(次)
OSC-Cluster	超时	461.0	72	103
	WDPM	480.3	40	99
	对比	+4.2%	-44.4%	-3.9%
SDSC-BLUE	超时	1791.1247	37	2429
	WDPM	1791.1248	10	1969
	对比	+0.00001%	-73%	-18.9%
CTC-SP2	超时	1586.1	33	1529
	WDPM	1596.5	13	1328
	对比	+0.66%	-60.1%	-13.1%
LLNL-Atlas	超时	969.4	37	266
	WDPM	988.9	19	258
	对比	+2.0%	-48.6%	-3.0%

在功耗方面,WDPM策略相对于超时策略只有少量的增加。从百分比来看,90天负载的模拟结果要略优于30天负载。这与WDPM策略的反馈修正机制有关。也就是说,从长远来看,由于WDPM策略可以自适应系统负载的变化,因而更易达到能耗与性能的平衡。

5 结 论

负载到达特征与用户使用习惯紧密相关,不同系统的负载模型有所不同,若初始分布不易确定,则需要对系统运行的历史数据进行曲线拟合来确定初始的负载变化曲线。并在系统运行中动态修正负载模型。另外,若负载到达模型呈现多峰特征,则需要进行分段处理。只要初始负载模型确定,则可采用预唤醒策略和反馈修正机制进行动态功耗管理。因此,该策略适用于各种高性能计算机系统。

冷却开销在系统的能量消耗中占很大比例。由于空调系统布局可能造成局部过热现象,因此优先选择哪些节点进行状态调整也会对整个系统功耗产生影响。今后可以从系统全局出发设计热量感知的动态功耗管理方法,并通过建立合理的模型,将其转换为优化问题的求解,力求找到能耗和性能的最优平衡。

参考文献

[1] Chedid W, Yu C. Survey on power management techniques for energy efficient computer systems. Technical report, Mobile Computing Research Lab, Cleveland State University, 2002

[2] TOP500 Team. The 37th Edition of TOP500 List. http://www.top500.org/lists/2011/11, Nov 2011

[3] Moore J, Chase J, Ranganathan P, et al. Making scheduling cool: temperature-aware resource assignment in data centers. In: Proceedings of Usenix Annual Technical Conference, Anaheim, USA, 2005. 61-75

[4] Feng W C. Making a Case for Efficient Supercomputing. *ACM Queue*, 2003, 1(7): 54-64

[5] Lorch J R, Smith A J. Software strategies for portable computer energy management. *IEEE Personal Communications*, 1998, 5: 60-73

[6] Benini L, Bogliolo A, Micheli G D. A survey of design techniques for system-level dynamic power management. *IEEE Transactions on Very Large Scale Integration Systems*, 2000, 8(3): 299-316

[7] Augustine J, Irani S, Swamy C. Optimal power-down strategies. *SIAM Journal on Computing*, 2008, 37(5): 1499-1516

[8] Platform. Green HPC. Dynamic power management in HPC. A Technology Whitepaper, 2008

[9] 江琦, 奚宏生, 殷保群. 动态电源管理超时策略自适应优化算法. 控制与决策, 2008, 4(10): 372-377

[10] Hwang C H, Wu A. A predictive system shutdown method for energy saving of event-driven computation. *ACM Transactions on Design Automation of Electronic Systems*, 2000, 5(2): 226-241

[11] 储毅, 赵敏. 基于马尔可夫决策的动态电源管理技术. 电子科技大学学报, 2007, 36(3): 521-523

[12] Lam T W, Lee L K, Ting H F. Sleep with Guilt and Work Faster to Minimize Flow Plus Energy. In: Proceedings of the 36th International Colloquium on Automata, Languages and Programming, Rhodes, Greece, 2009. 665-676

[13] Elnozahy E N, Kistler M, Rajamony R. Energy-Efficient Server Clusters. In: Proceedings of the 2nd Workshop on Power Aware Computing Systems, Cambridge, USA, 2002. 179-196

[14] Dolz M F, Fernández J C, Mayo R, et al. EnergySaving Cluster Roll: Power saving system for clusters. In: Proceedings of the 23rd Architecture of Computing Systems, Hannover, Germany, 2010. 162-173

[15] Hu F P, Evans J J. Power and environment aware control of Beowulf clusters. *Cluster Computing -the journal of networks software tools and applications*, 2009, 12(3): 299-308

[16] Xue Z, Dong X, Ma Si, et al. An energy-efficient management mechanism for large-scale server clusters. In: Proceedings of the 2nd IEEE Asia- Pacific Services Computing Conference, Tsukuba, Japan, 2007. 509-516

[17] Bernhard S. Energy optimization of existing datacenters-Save the planet and you budget. Open Grid Forum, Catania, Italy, 2009

[18] 林闯, 田源, 姚敏. 绿色网络和绿色评价: 节能机制, 模型和评价. 计算机学报, 2011, 34(4): 593-612

[19] Parallel workloads archive. http://www.cs.huji.ac.il/labs/parallel/workload/, Dec. 2011

[20] Lublin U, Feitelson D G. The workload on parallel supercomputers: modeling the characteristics of rigid jobs. *Journal of Parallel and Distributed Computing*, 2003, 63(11): 1105-1122

[21] Feitelson D. Workload modeling for computer systems performance evaluation. http://www.cs.huji.ac.il/~feit/wlmod, July 2011

Study of workload-driven dynamic power management for high performance computing clusters

Liang Aihua*,**, Xiao Limin*, Pang Yu*, Li Yongnan*, Ruan Li*

(*School of Computer Science and Engineering, Beihang University, Beijing 100191)

(**Institute of Computer Technology, Beijing Union University, Beijing 100101)

Abstract

A workload-driven dynamic power management (WDPM) strategy is proposed to improve the tradeoff between power consumption and performance in homogeneous computing clusters. Through the empirical analysis of real workload logs in production systems, the WDPM strategy integrates the load prediction-based pre-wakeup approach and the dynamic feedback-based revising mechanism based on the improvement of the timeout strategy. By using the data from real workload logs and practical systems, the extensive simulations were conducted to investigate the performance and energy consumption of the strategy. The experimental results indicate that, as compared with the timeout strategy, the WDPM strategy can effectively reduce the average wait time of jobs and the node state switching times of a system with a very little increase of power consumption. Therefore, it can alleviate the performance loss and achieve the better tradeoff of performance and power consumption.

Key words: dynamic power management (WDPM), workload-driven, pre-wakeup, feedback mechanism, cluster