文章编号:1671-4598(2020)08-0193-05 DOI:10.16526/j.cnki.11—4762/tp.2020.08.040 中图分类号:TP393 文献标识码:A

基于 M/M/c 排队模型的云计算中心能耗管理策略

张书豪，张延华，王倩雯，王朱伟，李 萌
（北京工业大学 电子信息与控制工程学院，北京 100124）

摘要：由于云计算中心在降低能耗的同时还需要保证服务质量（QoS），针对用户访问云计算中心的排队机制，给出一种云计算任务排队模型，在此基础上提出一种基于 M/M/c 排队过程的云计算中心能耗管理算法，通过求解该模型获得了平均等待时间、阻塞概率等性能指标进而建立系统的能耗模型；同时用参量 ERP（Energy—Response time Product）作为排队网络的反馈量，引入反馈策略及服务器休眠预留机制，动态调整云计算中心服务器服务数；仿真结果表明，与其他策略进行比较该策略能够在保证 QoS 值的情况下，有效降低系统的能耗，避免了服务器资源浪费。

关键词：云计算中心；反馈控制；ERP；资源管理；排队论

Energy Consumption Strategy for Cloud Computing Center Based on M/M/c Queuing Model

Zhang Shuhao, Zhang Yanhua, Wang Qianwen, Wang Zhuwei, Li Meng
(Academy of Electronic Information and Control Engineering, Beijing University of Technology, Beijing 100124, China)

Abstract: Owing to the cloud computing center needs to guarantee the quality of service (QoS) while reducing the energy consumption, presenting a queuing model of cloud computing tasks based on the queuing mechanism of users accessing the cloud computing center. Based on this model, an energy consumption management algorithm of cloud computing center based on M / M / C queuing process is proposed. By solving the model, the average waiting time, blocking probability and other performance indexes are obtained. Then the energy consumption model of the system is established. At the same time, the parameter ERP (energy response time product) is used as the feedback quantity of the queuing network, and the feedback strategy and the server sleep reservation mechanism are introduced to dynamically adjust the number of server services in the cloud computing center. The simulation results show that compared with other strategies, this strategy can effectively reduce the energy consumption of the system and avoid the waste of server resources while ensuring the QoS value.

Keywords: cloud computing center; feedback control; ERP; resource management; queuing theory

0 引言

云计算作为一种新兴的技术，以能够提供弹性的计算资源和按需付费的方式，在信息与通信技术领域引起了新的革命[1]。云计算是由分布式计算、网格计算和并行处理等技术逐步发展而来[2]，它将大量存储、软件和计算等资源集中调度，构成超大规模、高性能且可扩充的虚拟 IT 资源池，再通过面向云用户的服务接口，提供按需计费的服务模式[3]。但是在云计算中，云端服务器集群能耗极高，如何快速响应和解决用户任务请求和云端服务器能耗之间的矛盾，对用户体验、云服务提供商的服务质量、运营成本控制等都有着重大的影响[4]。分析用户访问云计算中心的排队过程并研究云计算中心服务器的休眠策略，能够有效地提升云计算中心运行效率。

目前，有许多文献针对数据中心能耗管理以及资源分配问题开展研究，G. Huang，S. Wang 等人曾研究了云计算平台中的资源分配问题，通过建立模型预测了每个客户的到达时间，从而计算满足资源需求的最小资源量并提出自动扩展虚拟机的算法，进而减少资源的使用[5]，但他们在对虚拟机的调节上使用的是基于阈值的策略，而设置阈值又是一项艰巨的任务，一旦阈值设置不当，策略效果将受到极大影响。文献［6］从客户的服务需求入手，进行了预测，以便服务提供商避免为满足服务水平协议（service layer agreement，SLA）而过度配置资源，但客户请求是随机变动的，通过预测值动态配置计算资源来满足 SLA 和的同时达到资源的最佳利用是困难的；文献［7］使用 M/G/m 模型分析了平均队长、平均响应时间和服务单元数量之间的关系，但是没有获得响应时间概率分布函数的精确表达式。文献［8］通过对云数据中心的测试和分析，提出一种

收稿日期:2019-12-25； **修回日期**:2020-01-06。
基金项目:国家自然科学基金面上项目(61571021);先进信息网络北京实验室(PXM2019_014204_500029);北京市教委科技计划面上项目(KM201610005004);中国博士后科学基金第 64 批面上项目(2018M640032);北京市博士后科研活动经费资助(ZZ2019—73)。
作者简介:张书豪(1993-),男,北京人,硕士,主要从事云计算、区块链等方向的研究。
通讯作者:张延华(1960-),男,北京人,教授,主要从事移动边缘计算,区块链,信息智能处理等方向的研究。

降低“等待能耗”的思路，并使用了 DPS（Dynamic Powering On/off）技术，但无法应对大量关闭服务器时访问量骤增的情况；文献［9］则通过对云计算中心建模从而对其服务性能进行分析，得出了用户请求的平均等待时间以及其他性能指标，给出了影响云计算中心性能的各种因素。文献［10］中提出了基于突发感知的云计算服务器资源预留机制，有效的解决了任务请求突然骤增情况下，休眠状态服务器无法及时被唤醒的问题，但是这会在一定程度上造成空闲资源的浪费。文献［11］中使用了双状态马尔可夫链对云计算虚拟机的资源需求进行建模，并且提出了基于静态分布的算法来调节虚拟机数量，但是对于现有的云计算中心而言，其虚拟机数量和任务到达量都是十分庞大的，而求解马尔可夫过程的计算量会随着基数的增加而指数增长。

本文提出一种基于 M/M/c 过程的云计算访问排队模型，通过对模型中的等待时间、队列长度等指标进行分析，获得了该模型响应时间的概率分布函数，在此基础上引入基于预留机制的 DPS 技术，并且嵌入以 ERP 为反馈量的闭环反馈策略，从而动态均衡系统能耗与 QoS 之间的关系。

1 系统结构及原理模型描述

1.1 云计算中心的工作机制

在云计算中心系统中，通常有大批任务请求进入数据中心。一旦有客户任务请求，云计算中心将根据客户的不同需求自适应提供不同类型的服务，云计算中心的工作机制如图 1 所示[12]。

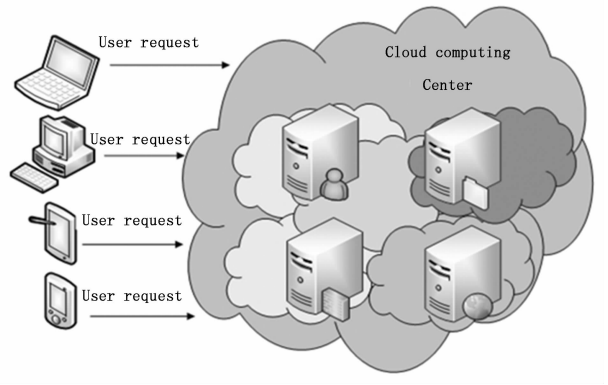

图 1 云计算中心的工作机制

典型的云计算环境可分为 4 层结构，分别是物理资源层、虚拟化资源层、管理中间件层和 SOA 服务层。云计算资源池属于资源虚拟化层，它屏蔽了物理资源层在分布上的细节，为其上层的管理中间件层提供服务支持。

1.2 云计算中心的任务队列模型

假设云计算资源池中有 c 台服务器（即 c 个计算单元），每个服务器都是独立运行的，服务器完成任务时不互相影响。客户请求到达的时间间隔服从参数为 λ 的泊松分布，并且每个服务器的服务速率服从参数为 μ 的指数分布。将这种系统称为具有多服务窗口的排队模型，简称 M/M/c 模型，M/M/c 排队模型运行方式如图 2 所示。

将所有用户每一次访问云服务器的动作，都当成一次

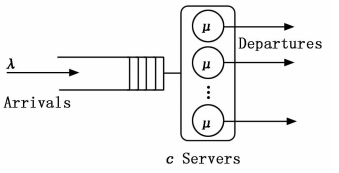

图 2 M/M/c 排队模型

顾客到达过程，并设到达的访问具有马尔可夫性，则可以将马尔可夫链引入云计算资源池中，预测分析未来时刻云计算资源池的状态[13]。由于访问过程服从参数为 λ 的泊松分布，且每次到来的任务所使用的资源量都是随机的，即可认为每个服务器的平均服务率遵循参数为 μ 的指数分布。除此之外，云服务器资源中物理机（PM）一般会被抽象为多个虚拟机（VM），客户实际分配到的是虚拟机资源，为简化起见本文将为每一个为客户服务的服务器都认为是一台虚拟机服务器 VM，则可以假设共有 c 台服务器。在不同时刻到达的用户服务请求中，服务顺序按先到先服务的规则。因此，客户访问云服务资源池的过程属于 M/M/c 排队过程。

1.3 随机任务模型

由于服务请求符合参数为 λ 的泊松分布，每个服务器的平均服务率遵循参数为 μ 的指数分布，且有 c 台服务器，则云服务资源池的平均服务率为 $c\mu$，$1/\lambda$ 为平均到达间隔，$1/\mu$ 为每个服务器的平均服务时间。设 ρ 为服务强度，$\rho=\lambda/c\mu$，当 ρ 趋近于 0 时，用户任务的等待时间短，服务器节点有大量的空闲时间；反之当 ρ 趋近于 1 时，节点的空闲时间少，用户的等待时间长。$c\mu$ 应当大于等于 λ 即 $\rho\leqslant 1$，此时排队模型存在平稳分布。假设云计算资源池不限制任务数量，故云计算资源池中随机任务的可能状态集应为 $\Phi=\{0,1,2\cdots\}$。由此可以得出云计算资源池随机任务的状态流图，如图 3 所示。

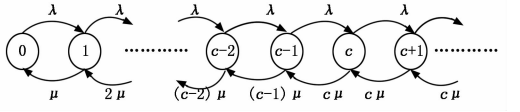

图 3 云计算资源池状态流图

其中：状态圆圈内的数字代表系统中服务进程数设为 n，当 $0\leqslant n\leqslant c$ 时，表示云计算资源池内有 n 个服务器单元正在被任务使用，其余 $c-n$ 个服务器单元空闲；当状态 $n>c$ 时（即达到云计算资源池的任务 n 超过 c 时），c 个进程都在被任务占用，而余下的任务排队等待服务。根据图 3，可得到系统状态之间的概率转移如下：

$$\lambda_n=\lambda \qquad for\ all\ n \tag{1}$$

$$\mu_n=\begin{cases} n\mu, & 1\leqslant n\leqslant c \\ c\mu, & n\geqslant c \end{cases} \tag{2}$$

由此可得概率系统状态转移矩阵：

$$\boldsymbol{P}=\begin{bmatrix} -\lambda & \lambda & 0 & & \\ \mu & -(\mu+\lambda) & \lambda & & \\ & 2\mu & -(2\mu+\lambda) & \lambda & \\ & & 3\mu & -(3\mu+\lambda) & \lambda \\ & & & \ddots & \ddots \end{bmatrix} \tag{3}$$

由于 n 为目前系统中的任务数，设 p_n 为此时系统中共有 n 个任务的概率，由此可知：

$$P_n = P_0 \frac{(c\rho)^n}{n!} \quad \text{for} \quad n \leqslant c \tag{4}$$

$$P_n = P_0 \frac{\rho^n c^c}{c!} \quad \text{for} \quad n \geqslant c \tag{5}$$

由正则性条件 $\sum_0^{\infty} P_n = 1$，当 $\rho < 1$ 时，有：

$$1 = \left(\sum_{n=0}^{c-1} \frac{(\rho_1{}^n)}{n!} + \sum_{n=c}^{\infty} \frac{\rho_1^n}{c!c^{n-c}} \right) P_0 = \left(\sum_{n=0}^{c-1} \frac{\rho_1^n)}{n!} + \frac{\rho_1^c}{c!} + \frac{1}{1-\rho} \right) P_0 \tag{6}$$

定义系统空闲概率为：

$$P_0 = \left[\sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{c!(1-\rho)} \right]^{(} - 1) \tag{7}$$

由于请求云计算资源池服务的作业最终都会完成服务，故丢失概率 $P_l = 0$。因此可知随机访问处于稳态时的各项系统指标，其计算公式如下：

排队队列中的平均客户数量：

$$L_q = \frac{(c\rho)^c \rho}{c!(1-\rho)^2} P_0 \tag{8}$$

系统中总的平均客户数量：

$$L_s = \frac{(c\rho)^c \rho}{c!(1-\rho)^2} P_0 + c\rho(\tag{9}$$

排队队列中的预期平均等待时间：

$$W_q = \frac{(c\rho)^c}{(c!(c\rho)(1-\rho)^2} P_0 \tag{10}$$

系统中总的预期平均等待时间：

$$W_s = \frac{(c\rho)^c}{(c!(c\rho)(1-\rho)^2} P_0 + \frac{1}{\mu} \tag{11}$$

1.4　建立实际场景模型

在云计算任务排队过程中，客户请求的任务量是随机变化的，若在大量服务器单元处于休眠状态时突遇用户任务请求骤增，由于唤醒服务器需要花费一定的时间，则可能无法及时地、有效地应付云用户的请求。基于带预留机制的 DPS（dynamic powering on \ off）策略，将所有服务器分为两组，其中永久运行的服务器构成服务主模块（base line module，BLM），等待启动的服务器构成服务预留模块（reserved line module，RLM）。系统共设置有 n_2 个服务器，BLM 中有 n_1 个服务器，RLM 中有 $n_2 - n_1$ 个服务器。其中 BLM 中服务器永久开启，RLM 中服务器可以在工作状态与休眠之间转变，当它的某个服务器处于休眠状态时，则该服务器仅消耗极少的维持休眠状态的能量。将缓冲区（Buffer）的设置为无限大，设置任务队列在缓冲区中也要消耗能量，但能量消耗较少。

从能耗方面考虑，RLM 中的服务器可以在服务器响应较快即缓冲区中作业数量较少时休眠一部分；同时为了保证用户体验即 QOS 值，RLM 服务器应在系统响应时间过长即缓冲区中作业较多时，重新开启一部分。一般地，由于云服务器数量足够多，且服务器唤醒——睡眠状态转化足够快，所以服务器的状态转换时间可忽略不计。需要注意的是，在对 RLM 中服务器进行休眠操作时，休眠策略是在 RLM 的控制器接收到指令后，休眠已经完成任务的服务器，对于 RLM 中正在服务却未完成服务的服务器则需要等待其完成当前工作后休眠，不能休眠正在为顾客服务的服务器。

为了符合实际场景，本文设置相同情况下，BLM 中服务器正常工作时的平均能耗等于 RLM 服务器，在请求到来时优先访问 BLM 中的服务器，避免 BLM 服务器资源的浪费。排队过程系统流程如图 4 所示。

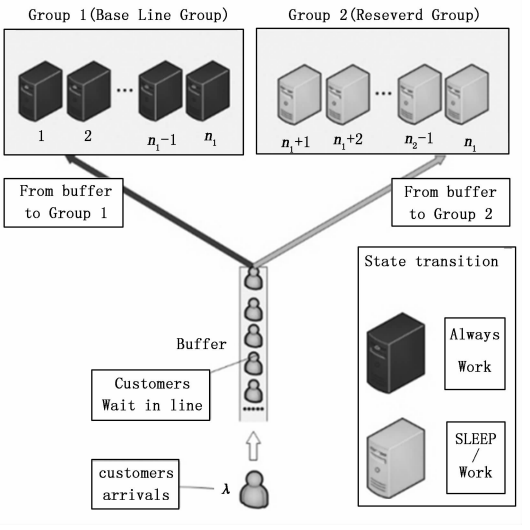

图 4　实际访问排队过程模型

1.5　建立系统能耗模型

考虑到实际问题，某些云服务器是不能够随意关闭的，关闭后可能会导致数据丢失或者重新配置数据库等一系列问题。所以设定 BLM 中服务器始终保持工作状态并保持能耗 W_1；为了简化起见，将 RLM 服务器的状态设置为两种：第一种是休眠状态，此时服务器功耗是 W_{21}，第二种是工作状态，服务器功耗是 W_{22}，W_{22} 要远大于 W_{21}，其总能耗记为 W_2；由于顾客请求在缓冲器中排队时也消耗能量，所以设置在缓冲器 Buffer 中的等待能耗为 W_3；由于前文中将 RLM 中服务器唤醒与休眠态之间的转化设置为瞬时的，所以转换状态的能耗 W_4 也可忽略不计。设 BLM 共有 n_1 台服务器，其中 n_1 台正在运行；BLM 共有 $n_2 - n_1$ 台服务器，其中 G 台正在运行则 $n_2 - n_1 - G$ 台处于休眠状态；设缓冲器队列中共有 D 个请求，故此系统的总能耗为：

$$W_A = W_1 \times n_1 + W_{22} \times G + W_{21} \times (n_2 - n_1 - G) + D \times W_3 \tag{12}$$

2　算法描述

能量响应时间乘积 ERP（energy－response time product），被认为是寻找能量性能权衡的合适度量。它被定义为：$ERP^{\pi} = E[P^{\pi}] \times E[T^{\pi}]$[14]。其中 $E[P^{\pi}]$ 是控制策略 π 下的长期平均功耗，$E[T^{\pi}]$ 是策略 π 下的平均客户响应时间。最小化 ERP 可被视为最大化" 每瓦特性能"，此处性能被定义为平均响应时间的倒数。在控制领域，反馈算法作为闭环控制系统的核心，对系统各个节点状态的控制、调节起到了至关重要的作用。本文提出一种带预留机制的基于 ERP 标准

的反馈算法，实现对系统的全局控制和优化。

将 B 设置为合理的用户预期响应时间，在此本文设定了一个状态并称之为反馈算法开启临界状态，即：BLM 中所有服务器均开启，且 Buffer 中队列长度为 Lq_n^*1，此队列长度使实际响应时间刚好为 B，即：$\frac{Lq_{n1}^*}{n_1\times\mu}=B$，则 $Lq_{n1}^*=B\times n_1\times\mu$。进而根据公式（10）以及 μ、λ 等值，可以推算出平均最佳服务器总开启数量值 c^*，再根据 c^* 即可得此时 Buffer 中平均队列长度为 Lq^*。则保持等待时间为 B 的稳态状态下，此时中服务器的平均功耗为：

$$W^* = Lq^*\times W_3+(c^*-n_1)\times W_{22}+(n_2-c^*)\times W_{21}+n_1\times W_1 \tag{13}$$

而此时系统的实际消耗功率为：

$$W_a = L_at\times W_3+(c_{at}-n_1)\times W_{22}+(n_2-c_{at})\times W_{21}+n_1\times W_1 \tag{14}$$

其中：L_{at} 为此时 Buffer 中实际队列长度，c_{at} 是此时实际运行服务器台数。

具体算法如 Alg. 1 所示，系统框图如图 5 所示。

Algorithm 1 Feedback algorithm based on ERP

Input: $\lambda,\mu,B,Q,n_1,T_c,T_{\max}$;

Output: Number of servers to adjust;

set User satisfaction response time B

compute Lq_n^*1, Lq^*, c^* by λ, B, μ and Equs. (10)

while request in system > $Lq_n^*1+n_1$ and $T_c<T_{\max}$ do

compute W^* by Equs. (13)

get the Feedback system input $W^*\times B$

detect the actual power consumption W_a and actual response time T in T_c

get the Feedback system input $W_a\times T$

compute $\triangle W=W_a\times T-W^*\times B$

If $\triangle W>0$ then

wake up some servers by Linear programming Alg.

else if $\triangle W<0$

dormancy some servers by Linear programming Alg.

else

Dormancy all servers in RLM

End if

$T_c=T_c+10$

End while

return

3 实验结果与分析

3.1 参数设置及实验背景

在 Matlab 2019a 环境下，进行实验仿真，对本文提出的基于 ERP 的带预留机制的反馈算法进行仿真，从任务的响应时间和能耗两个方面进行仿真来评估算法。实验建立了一个由 100 个服务器组成的数据中心，其中 50 个 BLM 服务器，50 个等待启动的 RLM 服务器。排队规则设置为服从 M/M/c 排队模型，在任务排队的过程中，根据实时 ERP 的值，来决定是否开启 RLM 中服务器以及开启多少数量的 RLM 服务器。计算出系统在仿真时长内的能耗，以及系统平均响应时间，与 never－off 算法与静态阈值算法进行比较，仿真时长到达后，实验结束。

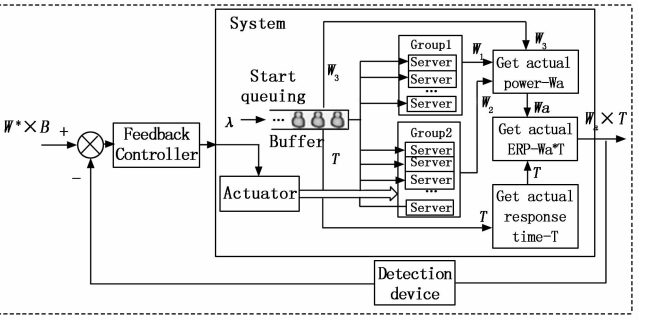

图 5 基于反馈控制调节的系统框图

实验的机器配置为：Intel（R）Core（TM）i7－4700HQ CPU @2.40GHz（8 CPUs）；8GB RAM；硬盘 1TB；windows 10 Professional 64 位。为了保证仿真系统的运行存在平稳状态，设置 $0<\frac{\lambda}{n_2\times\mu}<1$；实验环境涉及的相关参数以及取值如表 1 所示。

表 1 实验参数设置

参数名	取值设置	含义
n_2/个	100	服务器总数量
n_1/个	50	BLM 中服务器数量
λ	700	任务的平均到达率
μ	10	任务的平均服务速率
W_1/W	100	BLM 中服务器能耗
W_{21}/W	10	RLM 中服务器休眠时能耗
W_{22}/W	100	RLM 中服务器工作时能耗
W_3/W	5	缓冲器 BUFFER 中的能耗
W_4/W	0	RLM 中服务器状态转换能耗
B/s	0.2	设定的顾客理想等待时间
$T_{\max}$/s	100	总仿真时长
T_c/s	10	检测周期

3.2 实验结果及比较

首先，仿真完成了在上述参数及规则下的基于 ERP 的带预留机制的反馈算法，其次比较了 never－ off 算法、静态阈值算法、本文算法的任务平均响应时间和系统执行任务的功耗。设 $\lambda=700$ 后，可得到任务到达符合泊松分布的数据，如图 6 所示。仿真实验比较了在总仿真时间为 100 s 时，3 种算法的平均响应时间，如图 7 所示，never－off 算法的平均响应时间最短，静态阈值算法最长，而本文算法的平均响应时间介于上述两种算法之间。这是由于在 never－off 算法中，云计算中心开启的服务器数量是根据排队理论预测出来的，云数据中心的服务器不能动态唤醒或休眠，所有任务都可以尽快地分配到服务器；在静态阈值算法中，虽然服务器可以动态唤醒休眠，但开启服务器的数量在每个阶段都有固定值，在该阶段内即使任务数增加，服务器数量也不会增加，此时便会导致平均响应时间增长，无法较好地满足云用户任务响应时间需求；而在本文算法中，

云计算中心的运行服务器数量能够根据 ERP 变动动态改变，能够较好地满足云用户任务响应时间需求。

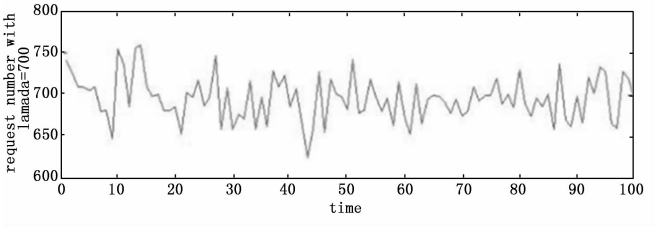

图 6　泊松分布下任务请求数量到达变化

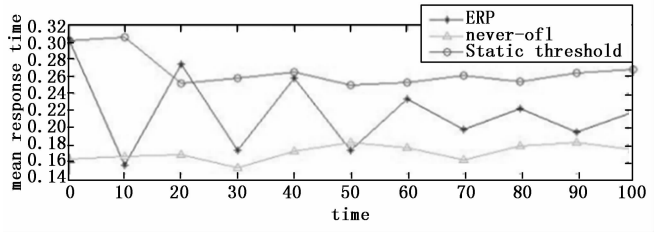

图 7　3 种算法响应时间比较

由图 8 可知，在相同条件下，本文算法的消耗的能耗最小，never－off 算法的消耗的能耗最大，静态阈值算法消耗的能耗介于上述两个算法之间。这是由于在 never－off 算法中，云计算中心的服务器总数量是根据排队论预测值计算得出的，并始终全部开启，在任务到来较少时产生了大量无用能耗，因此其节能效果最差；在静态阈值算法中，只有当队列中的任务超过设定的阈值时，部分服务器才开启并提供服务，这在一定程度上避免了大量空闲能耗的产生，相对 never－off 算法也有一定的改进，然而该算法需要设置阈值，阈值设置本身就困难，阈值设置不当很可能会导致系统性能不佳。在本文算法中，可以根据 ERP 值动态调整并决策出相对较优的运行服务器数量，避免开启过多 RLM 中的服务器，有利于降低云数据中心空闲能耗。通过结果对 3 种算法进行对比，可知 never－off 算法能提供最优的系统性能，但其消耗的总能耗最多，不能满足云计算中心的利益；静态阈值算法在一定程度上节约了能耗，照顾到了云计算中心的利益，但一旦系统中任务数过多，就会使任务的平均等待时间上升，降低了 *QOS* 值；而本文提出的策略能够兼顾云计算中心和用户的利益，相对较优。图 9 中，actual *ERP* 是实际 *ERP* 的变化，ERP^* 是期望。

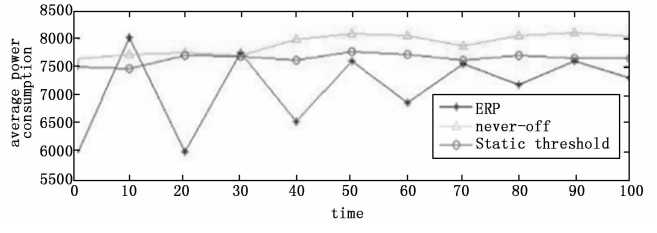

图 8　3 种算法功率消耗比较

从图 9 可以看出，本文的算法得出的实际 *ERP* 值始终在预估值的附近波动，而且越来越趋近于预期 *ERP*，故可

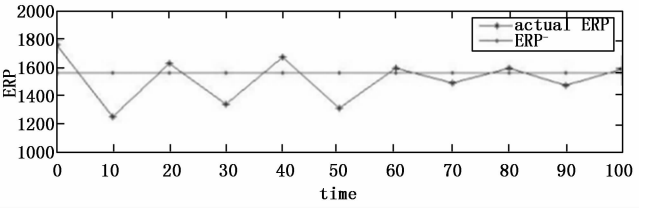

图 9　*ERP* 变化比较图

知仿真结果符合预期。值得注意的是，在图 6 和图 7 中，本文算法的初期波动较大，图 8 中实际 *ERP* 值也呈现出折线状波动，这是由于本文算法在初期调整时为了轻量化计算量选用了反馈算法而造成的现象。但经过短暂的波动期后，数据指标便会趋于稳定。

4　结束语

针对现有云计算中心能耗浪费问题，对云计算中心的工作机制进行了分析，得出了云计算中心的工作特点，利用 M/M/c 排队模型对云计算中心用户访问过程进行了建模，并得出各项重要指标。引入了动态开关策略（DPS），在此基础上提出了基于 ERP 标准的带预留机制的反馈控制策略。基于 Markov 过程的状态空间及其状态转换关系的控制策略会随着服务器数量的增加导致策略计算指数型增加，本文提出的轻量级算法在云服务器中心服务器计算单元巨大的实际情况下，具有快速得出策略控制结果的优势。以 ERP 为衡量标准，均衡考虑了能耗与响应时间两项重要指标，最大化“每瓦特性能”，寻找最佳的能耗绩效权衡。实验证明，该策略能够在保证 QoS 的情况下，有效降低系统能耗。但是也存在得出的控制结果不够精确，仿真初期震荡较大的缺点。下一步将重点考虑引入参数随时间变化的泊松分布，在不过多增加计算量的前提下，研究更加精确的控制方法，提出本算法的补充算法，进一步对云计算中心的服务质量和能耗进行优化管理。

参考文献：

[1] Fan Q，Liu L. A survey of challenging issues and approaches in mobile cloud computing [A]. 17th International Conference on Parallel and Distributed Computing，Applications and Technologies (PDCAT) [C]. IEEE，2016.

[2] Buyya R. Cloud computing：the next revolution in information technology [M]. IEEE，2010.

[3] 傅颖勋，罗圣美，舒继武. 安全云存储系统与关键技术综述 [J]. 计算机研究与发展，2013 (1)：138－147.

[4] Yao D，Yu C，Yang L，et al，Using crowdsourcing to provide QoS for mobile cloud computing [J]. IEEE Trans. Cloud Comput.，2019，7 (2)：344－356.

[5] Huang G，Wang S，Zhang M，et al. Auto scaling virtual machines for web applications with queueing theory [A]. International Conference on Systems & Informatics [C]. IEEE，2017.

[6] Anithakumari S，Chandrasekaran K. Negotiation and monitoring of service level agreements in cloud computing services [M]. Springer Singapore，2017.

（下转第 237 页）