·智能交通专题· 文章编号:1000-3428(2017)05-0035-05 文献标志码:A 中图分类号:TP18

轨道交通系统中多客户端连接池动态分配策略

季一木[1,2,3],杨罗坤[1],柴博周[1],朱曈晖[1],李文峰[4]

(1.南京邮电大学 计算机学院,南京 210023;2.深圳大学 广东省普及型高性能计算机重点实验室,广东 深圳 518060;
3.南京理工大学 高维信息智能感知与系统教育部重点实验室,南京 210094;
4.江苏省城市轨道交通研究设计院股份有限公司,南京 211800)

摘 要: 针对轨道交通集群调度系统中数据库连接池参数一次性设定后不可修改的问题,设计面向多客户端的数据库连接池动态分配策略。通过使用动态分配算法,根据每个客户端访问频率的不同为当前客户端分配最优连接数,从而达到提高系统资源利用率的目的。实验结果表明,该动态分配策略能够缩短连接池响应时间,提高系统运行效率。

关键词: 数据库连接池;集群调度系统;多客户端;连接复用;动态分配

中文引用格式: 季一木,杨罗坤,柴博周,等. 轨道交通系统中多客户端连接池动态分配策略[J]. 计算机工程,2017,43(5):35-39,46.

英文引用格式: Ji Yimu,Yang Luokun,Chai Bozhou,et al. Dynamic Allocation Strategy for Multi-client Connection Pool in Rail Transit System[J]. Computer Engineering,2017,43(5):35-39,46.

Dynamic Allocation Strategy for Multi-client Connection Pool in Rail Transit System

JI Yimu[1,2,3],YANG Luokun[1],CHAI Bozhou[1],ZHU Tonghui[1],LI Wenfeng[4]

(1. College of Computer,Nanjing University of Posts and Telecommunications,Nanjing 210023,China;
2. Guangdong Key Laboratory of Popular High Performance Computers,Shenzhen University,Shenzhen,Guangdong 518060,China;
3. Key Laboratory of Intelligent Perception and Systems for High-dimensional Information of Ministry of Education,
Nanjing University of Science and Technology,Nanjing 210094,China;
4. Jiangsu Urban Rail Transit Research and Design Institute Co.,Ltd.,Nanjing 211800,China)

【Abstract】 In the rail transit clusters dispatch system,the parameters of database connection pool is one-time set and can not be modified. Aiming at this shortage,a dynamic allocation strategy of multi-client database connection pool is designed. It uses a dynamic allocation algorithm to allocate the optimal number of connections according to the different access frequency of each client for current client,so as to achieve the purpose of improving the utilization of system resources. Experimental results show that the dynamic allocation strategy proposed in this paper can shorten the response time of connection pool and improve the efficiency of system.

【Key words】 database connection pool;clusters dispatch system; multi-client; connection multiplexing; dynamic allocation

DOI: 10.3969/j.issn.1000-3428.2017.05.006

0 概述

城市轨道交通以其快捷、节能、方便、舒适等优点在城市交通运输行业有着广阔的发展前景并已取得巨大的发展[1]。地铁是城市快速轨道交通的先驱,地铁不仅具有运量大、速度快、安全、准时、节省能源、不污染环境等优点,而且可以在建筑群密集而不便于发展地面和高架轨道交通的地区大力发展。在地铁调度系统中,调度的响应时间直接影响城市轨道交通运营调度的效率。同时效率问题也是数据

基金项目: 江苏省自然科学基金青年基金(BK2013876);广东省普及型高性能计算机重点实验室开放基金(SZU-GDPHPCL2014);江苏省城市轨道交通研究设计院股份有限公司项目(JSUMT-JY-00-1512)。

作者简介: 季一木(1978—),男,教授、博士,主研方向为云计算、大数据、移动互联网;杨罗坤,硕士;柴博周、朱曈晖,硕士;李文峰,工程师。

收稿日期: 2016-06-12 **修回日期:** 2016-07-13 **E-mail:** jiym@njupt.edu.cn

库应用系统的一个重要衡量标准[2]。

在地铁交通调度台系统的开发过程中,连接的底层很多是通过 Socket 来通讯的。但建立 Socket 连接是非常耗时的,如果每个连接都在使用完后就断开 Socket 连接,需要时再重新建立 Socket 连接,这就构成了连接资源的浪费。如果频繁发生这种数据库操作,系统的性能必然会急剧下降,所以,应尽量复用连接以提高效率。而提高并发处理能力的基本思想就是更合理地分配和使用资源[3]。对于需要数据库支持的软件系统,采用数据库连接池技术就是为了合理利用资源,提高系统性能[4-5]。

现有数据库连接池的参数配置都是在初始化时一次设定,连接池在运行过程中其值不会改变[6],连接池中的内部管理机制也存在局限性,导致在实际应用中资源利用率低,过多并发连接也使系统运行效率下降[7]。此外基于平均数的动态连接池方法,其优点是方法简单、稳定性好,缺点是具有滞后性[8]。这些常规的数据库连接池管理技术虽然能提高系统的性能,但应用在轨道交通调度这个面向多个客户端访问的实际系统中时,无法进一步提高整个系统的数据库操作效率,原因是在集群调度系统中,由于每个集群调度用户在使用习惯上的差异性,导致了每个客户端的数据库操作频率的差异性,若每个客户端均使用固定的连接池参数,则不能随用户请求的多少而改变[9]。一方面,当用户请求频率很低时,空闲过多的连接会造成资源的浪费;另一方面,当用户请求频率很高时,可能预设的连接数不能满足要求,导致用户等待时间过长而出现异常[10-11]。

针对上述问题,本文在轨道交通集群调度系统中建立一个数据库连接池,并设计一个高效的连接池管理策略对数据库连接进行复用,避免数据库连接资源频繁建立和关闭,减少响应时间,从而提高调度系统的运行效率[12]。

1 数据库连接池工作原理

数据库连接池是面向众多连接对象的存储池,它提供一种连接管理策略,对连接对象的个数进行控制,同时提供应用程序的使用和获取连接的接口。数据库连接池主要由三部分组成:1)连接池的建立;2)连接池中连接的使用管理,包括对最大连接数、最小连接数的管理[13];3)连接池的关闭。图 1 给出一个传统的数据库连接池的原理图。当客户通过线程请求连接时,若连接池中有空余连接则分配连接响应请求,若没有则创建新的连接。当线程使用完连接后,将连接释放回连接池。

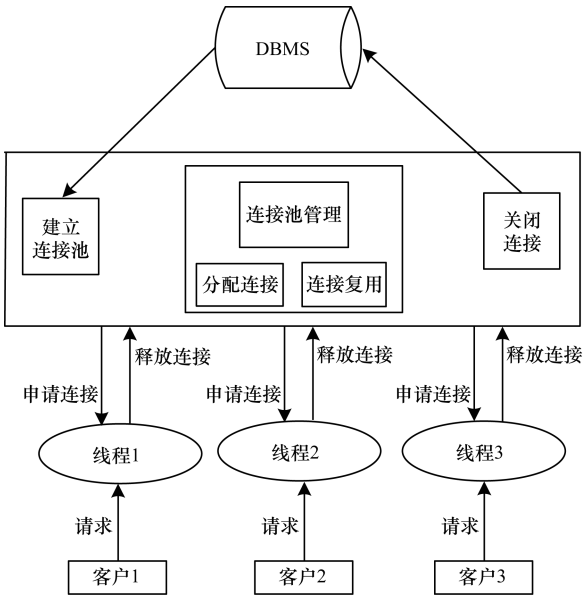

图 1 传统数据库连接池原理图

2 数据库连接池的设计和优化

集群调度系统采用 C/S 架构,在轨道交通调度系统中,由于不同用户使用习惯的差异,导致其使用客户端访问数据库的频率也存在差异,若所有客户端的连接池均使用固定参数,则整个系统在数据库处理方面无法达到最优性能[14]。为此,本文提出一种实时生成的动态参数技术,以提升数据库操作的性能。每个客户端在运行过程中产生的数据会实时发送至服务器,服务器根据每个客户端连接池的负载情况以及整个数据库服务器的负载情况计算出每个客户端的最大连接数和最小连接数,并发送至各个客户端,客户端根据最新的最小连接数和最大连接数分配数据库连接。

2.1 参数定义与说明

数据库连接池中的参数定义与说明如下:

1)最大连接数(*maxLinkCount*):客户端中连接池所能分配的连接池上限;

2)最小连接数(*minLinkCount*):客户端中连接池至少要分配的连接数;

3)当前连接数(*currentLinkCount*):客户端当前连接总数;

4)当前空闲连接数(*freeLinkCount*):客户端中

空闲连接数;

5)连接请求超时时间(*maxWaitTime*):客户端进程向连接池请求连接资源时等待的最长时间;

6)连接过期时间(*timeOut*):连接回收进程判断是否强制回收一个已分配的连接资源的依据;

7)连接请求失败率(*failRate*):单位时间内请求连接资源失败的次数与请求连接的总次数之比;

8)连接资源利用率(*usedRate*):当前已分配连接资源数与系统中总连接资源数之比;

9)数据库连接总数(*maxRate*):在不影响数据库操作前提下数据库能够分配的最大连接数。

2.2　连接资源的管理

在轨道交通调度这个多客户端访问的系统中,每个客户端均拥有一个独立的数据库连接池以管理本客户端的数据库连接服务。当客户端初始化时,数据库连接池按最小连接数创建连接。当系统需要进行数据库操作时,首先向数据库连接池请求连接。若连接池中存在空闲连接,则分配给请求进程,请求进程处理完数据库操作后将连接归还至连接池。若进程请求连接时连接池中没有空闲连接,则连接池判断当前已分配的连接数是否达到预设的最大连接数。若尚未达到最大连接数则创建一个新的连接并分配给请求进程。若已达到最大连接数,则请求进程须等待一定时间(连接请求超时时间),若超过该时间,则连接获取失败。若在连接请求超时时间内有其他进程归还连接,则将该连接分配给等待请求的进程[15]。另外还需创建一条额外线程,用于监控当前已分配连接的活跃度。若超过预设时间某一连接仍然没有数据交互,则连接池将强制收回该条连接,从而避免某一个已完成数据库操作的进程过长时间占据连接,进而提高连接利用率。

2.3　动态连接池的结构模型

为实时、动态地生成每个客户端的连接池参数,须使用一个服务器作为调度中心,它负责对各个客户端在一个周期内的请求率进行统计。若某客户端请求率较低时,则从该客户端回收相应连接,同时将这些连接分发给请求失败率高的客户端。动态连接池的系统结构如图 2 所示。

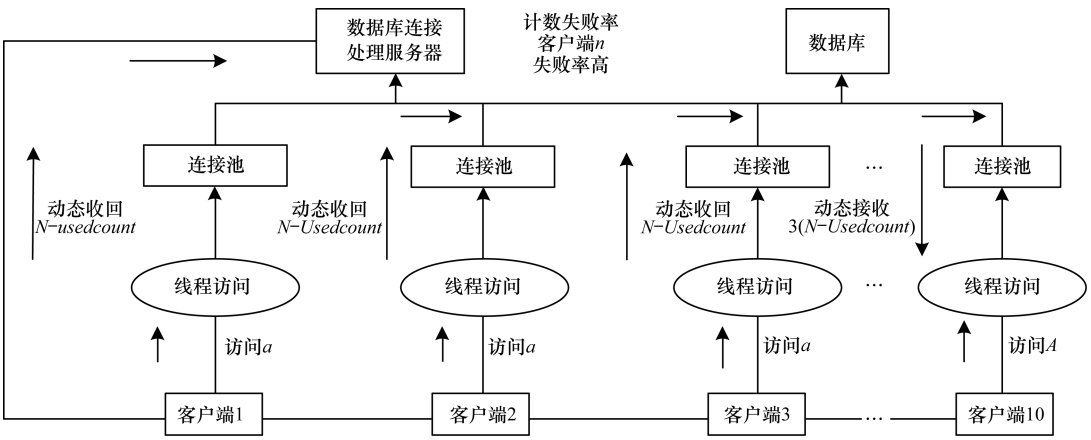

图 2　动态连接池系统结构

在图 2 中:1)初始设定最大连接数 N,最小连接数 n,在客户端 1 ~ 客户端 3 中的访问量为 a,$a < n$。在客户端 10 中的访问量为 A,$A >> N$。*UsedCount* 为当前连接数;2)客户端中访问量 a 很小,因此,此时动态回收(N – *Usedcount*)个连接,同理从客户端 2 和客户端 3 中回收(N – *Usedcount*)个连接。客户端 10中访问量为 A,$A >> N$,出现连接不够用,因此,回收 3 倍的(N – *Usedcount*)个连接给客户端 10。

2.4　动态分配策略设计及实现

2.4.1　动态分配策略设计

动态分配策略的设计思想为:

1)当客户端初始化时,数据库连接调度服务器为该客户端分配一套默认的数据库连接参数(包括最小连接数、最大连接数),客户端的连接池基于此套参数进行初始化设置,按最小连接数创建相应的连接资源。

2)在客户端运行的过程中,需要实时计算本客户端的连接资源利用率和连接请求失败率,并将其发送至数据库连接调度服务器。数据库连接调度服务器实时记录所有客户端发来的数据,并基于这些数据为每个客户端计算出最小连接数和最大连接数。当数据库连接调度服务器感知到某些客户端连接请求失败率过高时,需要剥夺连接请求失败率较低的客户端的最小连接数和最大连接数,并将这一

部分连接资源分配给连接请求失败率较高的客户端,从而达到负载均衡。此外,若超过 80% 的客户端均出现连接请求失败率过高的情况,则数据库连接调度服务器需要在不超过数据库最大连接数的前提下增加整个系统的连接数,从而降低整个系统的连接请求失败率。

2.4.2 动态分配策略实现

动态分配策略的实现步骤如下:

1)设客户端初始化时设定的最大连接数为 N,最小连接数为 n,当前连接为 *currentLinkCount*,当前空闲的连接数 *freeLinkCount*。当某一客户端的 $currentLinkCount < n$,$freeLinkTime > 0$,同时 *usedRate* 较低,说明该客户端连接资源出现了冗余,应该回收该客户端的连接资源;当另一客户端中 *freeLink-Count* $=0$,同时 *failRate* 很高,说明该客户端中连接资源不够用,应该从其他空闲连接中回收连接资源,分配给该客户端。

2)对连接复用情况的判定需要获取以下数据:连接等待的时间(*firstTime*),连接等待时间间隔(*waitTime*),当前系统时间(*currentTime*),连接请求超时时间(*maxWaitTime*),连接过期时间(*timeOut*)。将这些时间以对象的形式存储到一个 *connectTime* 的结构体中。由于时间的产生是一个时间序列,因此还需要一个 *Linklist* 来存储这些时间。当客户端中连接的 $firstTime < waitTime$ 时,此时连接的使用率为 $usedRate = (currentLinkCount/minLinkCount)$。当 *usedRate* 很低时,说明此时该客户端上有连接冗余,出现连接资源的浪费,应回收一部分。当 $maxWaitTime > waitTime$ 时,此时连接使用率 $usedRate = 100\%$,同时用户请求等待时间超过 *maxWaitTime*,失败连接率 *failRate* 很高,此时需要给该客户端分配更多的连接资源。动态分配策略实现流程如图 3 所示。

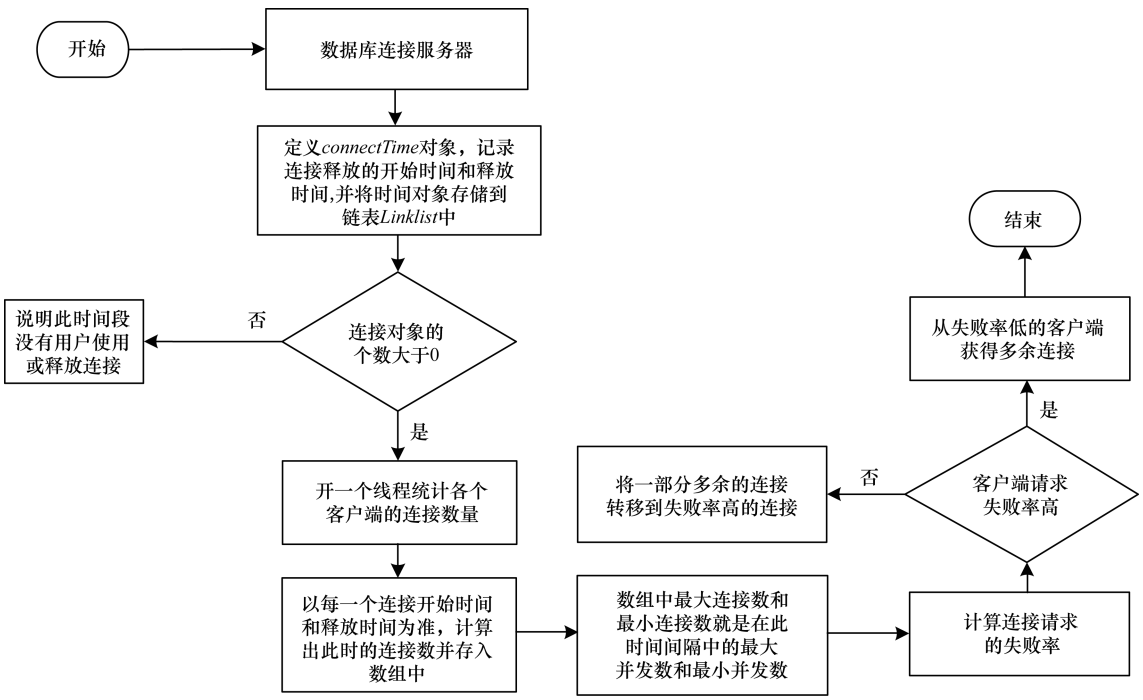

图 3 动态分配策略流程

2.5 算法描述

动态分配策略的算法描述如下:

输入 每个客户端中的线程请求数,分别为 $A_1, A_2, \cdots$

输出 系统在一定周期内完成连接的响应时间 T,为每个客户端响应时间 $t_1, t_2, \cdots$ 之和

```
1. 初始化最大连接数 N 和最小连接数 n
2. if( CurrentLinkCount > n && CurrentLinkCount < N)
/* 判断当前空闲连接数,比较当前连接 CurrentLinkCount 是否在最大连接数 N 和最小连接数 n 之间 */
3. if( FreeLinkCount > 0 )
/* 检测空闲连接 FreeLinkCount */
4. failRate = ( N – CurrentLinkCount)/N
/* 计算连接的失败率 */
5. if( failRate > 50% ) then
/* 如果失败率过高大于 50% */
6. Distribute( )/* 通过调用 Distribute( ) 函数从失败率高的客户端从服务器端获取更多连接,同时服务器端从失败率低的客户端回收空闲连接,连接用完之后就释放掉 */
```

7. Linklist(t)/ * 建立一个 Linklist 链表来存储时间序列,记录一个测试周期的时间 * /

8. FirstTime = CurrentTime/ * 复位连接时间,初始化连接的等待时间,方便下一次进行连接的计算 * /

9. close( )/ * 测试结束,关闭连接 * /

3　性能测试

3.1　测试环境与数据

对本文策略进行性能测试,服务器与客户端设置如下:服务器 CPU 为 Intel Xeon 7500,内存为 DDR3 ECC Registered 32 GB,硬盘为 20T SATA,操作系统为 Windows Server 2008,数据库为 MySql5.6.3,编程语言为 Qt5.4.0,编译器为 MinGW 4.9.1。

客户机 CPU 为 Intel Core I5,内存为 4 GB,硬盘为 500 GB,操作系统为 Windows 7,编程语言为 Qt5.4.0,编译器为 MinGW 4.9.1。

实验中,通过 4 个客户端来进行访问,A 表示每个客户端访问的进程数,A_1 ~ A_4 分别表示客户端 1 ~ 客户端 4 的进程数;N 表示最大连接数;n 表示最小连接数。使用普通的数据库连接池时,设定数据库连接池的初始值 $N=100$,$n=30$,不能修改。然后采用动态分配策略进行测试。实验采用如表 1 所示的 7 组测试数据。

表 1　测试数据(进程数)

实验编号	A_1	A_2	A_3	A_4
1	15	30	80	150
2	130	20	70	100
3	60	150	100	20
4	90	40	170	30
5	40	70	150	80
6	130	30	50	110
7	150	30	70	120

3.2　测试结果与分析

本次实验用了 7 组数据进行实验,在一个监控周期内分别对参数固定的数据库连接池、基于平均数分配的数据库连接池和动态分配的数据库连接池在轨道交通调度系统中进行实验结果比较。实验结果如图 4 所示,可以看出,当调度系统中用户请求不固定时,采用动态分配策略的数据库连接池响应时间最短,效果优于普通连接池和基于平均数的动态分配策略。

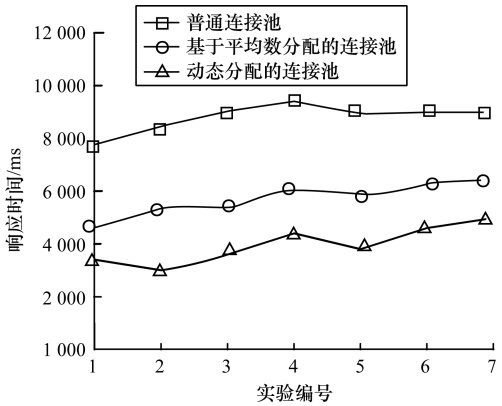

图 4　响应时间对比

4　结束语

本文分析传统数据库连接池原理,针对轨道交通调度系统提出一种面向多客户端的数据库连接池动态分配策略。通过数据库连接调度服务器实时记录所有客户端发来的数据,并基于这些数据为每个客户端计算当前连接数和连接请求失败率。当数据库连接调度服务器感知到某些客户端连接请求失败率过高时,则取消失败率较低客户端的连接,并将这一部分连接资源分配给失败率较高的客户端,从而达到负载均衡,提升整个系统的运行效率。下一步将研究如何通过机器学习的方式对自适应参数进行智能分配。

参考文献

[1] 周光军.基于云计算技术的城市轨道交通信息系统统一开发测试平台应用架构浅析[J].信息安全与技术,2015,1(1):43-47.

[2] 王学慧.基于 SQL Server 的数据库应用系统性能优化研究[J].电子科学技术,2015,15(1):95-99.

[3] 王霓虹,张　涛.基于 Java 的数据库连接池设计方案的研究[J].信息技术,2006,27(3):102-106.

[4] 欧阳洪基,葛　萌,赵　蔷.JDBC 与设计模式的数据库连接池的实现方法[J].计算机技术与发展,2011,21(1):84-88.

[5] Pyarali I, Aspivak M, Cytron R, et al. Evaluating and Optimizing Thread Pool Strategies for Real-time CORBA[J]. ACM SIGPLAN Notices, 2010, 36(8): 214-222.

[6] 王国亮,安世全.一种数据库连接池的动态控制策略[J].重庆邮电大学学报(自然科学版),2009,19(4):446-449.

[7] 孟培超,胡圣波,舒　恒,等.基于 ADO 数据库连接池优化策略[J].计算机工程与设计,2013,35(4):1706-1710.

(下转第 46 页)

5 结束语

本文针对高性能计算系统和分布式数据中心功耗过高和功耗峰值的问题,提出一种基于 RAPL 的功耗限额控制方法,使得在给定机群的功耗限额下可以最大化地发挥程序的性能。该方案不仅可以解决功耗峰值的问题,还可以为机群功耗规划和节点数量规划提供一种可行的手段。通过实验可以看出,采用基于 RAPL 的功耗限额控制方法,在性能损失 7.8% 的情况下,可以将功耗峰值降低 200 W,并保持功率稳定;在性能损失 19.45% 的情况下,可以将功耗峰值降低 471 W。下一步将在实验分析的基础上比较不同类型负载在受到限额控制时的性能变化,进而采取更细致的功耗管控策略。

参考文献

[1] Bhattacharya A A, Culler D, Kansal A, et al. The Need for Speed and Stability in Data Center Power Capping[J]. Sustainable Computing: Informatics and Systems, 2013, 3(3): 183-193.

[2] Fan Xiaobo, Weber W D, Barroso L A. Power Provisioning for a Warehouse-sized Computer[J]. ACM SIGARCH Computer Architecture News, 2007, 35(2): 13-23.

[3] 卢春鹏. 动态电压与频率调节在降低功耗中的作用[J]. 单片机与嵌入式系统应用, 2007, 7(5): 12-14.

[4] 周亦敏, 沈云龙, 曹丽东. 基于异构多核平台 H.264 解码的 DVFS 算法[J]. 计算机工程, 2013, 39(11): 268-271.

[5] Li Shuai, Broekaert F. Low-power Scheduling with DVFS for Common RTOS on Multicore Plat-forms[J]. ACM SIGBED Review, 2014, 11(1): 32-37.

[6] 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.

[7] Hom J, Kremer U. Energy Management of Virtual Memory on Diskless Devices [M]//Benini L, Kandemir M, Ramanujam J. Compilers and Operating Systems for Low Power. Berlin, Germany: Springer, 2003: 95-113.

[8] Cochran R, Hankendi C, Coskun A K, et al. Pack & Cap: Adaptive DVFS and Thread Packing Under Power Caps[C]//Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture. New York, USA: ACM Press, 2011: 175-185.

[9] Saravana M, Govidan S, Lefurgy C, et al. Using On-line Power Modeling for Server Power Capping [C]// Proceedings of 2009 Workshop on Energy-efficient Design. Austin, USA: [s. n.], 2009.

[10] Gandhi A, Harchol-Balter M, Das R, et al. Power Capping via Forced Idleness [C]//Proceedings of Workshop on Energy-efficient Design. Austin, USA: [s. n.], 2009.

[11] Georgiou Y, Glesser D, Trystram D. Adaptive Resource and Job Management for Limited Power Consumption[C]// Proceedings of 2015 IEEE International Parallel and Distributed Processing Symposium Workshop. Washington D. C., USA: IEEE Press, 2015: 863-870.

[12] Petoumenos P, Mukhanov L, Wang Zheng, et al. Power Capping: What Works, What Does Not [C]// Proceedings of the 21st International Conference on Parallel and Distributed Systems. Washington D. C., USA: IEEE Press, 2015: 525-534.

[13] Rountree B, Ahn D H, de Supinski B R, et al. Beyond DVFS: A First Look at Performance Under a Hardware-enforced Power Bound [C]//Proceedings of the 26th IEEE International Parallel and Distributed Processing Symposium Workshops. Washington D. C., USA: IEEE Press, 2012: 947-953.

[14] Hähnel M, Döbel B, Völp M, et al. Measuring Energy Consumption for Short Code Paths Using RAPL[J]. ACM SIGMETRICS Performance Evaluation Review, 2012, 40(3): 13-17.

[15] Intel Corporation. Intel® 64 and IA-32 Architectures Software Developer's Manual[Z]. 2011.

[16] Zhang Huazhe, Hoffman H. A Quantitative Evaluation of the RAPL Power Control System[C]//Proceedings of the 10th International Workshop on Feedback Computing. Seattle, USA: [s. n.], 2015: 1-6.

编辑 金胡考

(上接第 39 页)

[8] Kang D, Han S, Yoo S, et al. Prediction-based Dynamic Thread Pool Scheme for Efficient Resource Usage[C]// Proceedings of the 8th International Conference on Computer and Information Technology. Washington D. C., USA: IEEE Press, 2008: 159-164.

[9] Wang Qingyang, Kanemasa Y, Li Jie, et al. Response Time Reliability in Cloud Environments: An Empirical Study of n-Tier Applications at High Resource Utilization [C]// Proceedings of the 31st Symposium on Reliable Distributed Systems. Washington D. C., USA: IEEE Computer Society, 2012: 378-383.

[10] 王俊峰, 谢冬青. 基于线程池技术集群接入点的应用研究[J]. 微计算机信息, 2009, 25(24): 133-135.

[11] 孙旭东, 韩江洪, 刘征宇, 等. 基于分段的线程池尺寸自适应调整算法[J]. 计算机工程, 2011, 37(2): 44-47.

[12] Eichenberger A E. Using Advanced Compiler Technology to Exploit the Performance of the Cell Broadband Engine Architecture[J]. IBM Systems Journal, 2006, 45(1): 59-84.

[13] 王宇杰, 王 峰, 杨文宾. 基于数据库连接池的数据库访问性能对比测试研究[J]. 工业控制计算机, 2010, 23(6): 92-94.

[14] 薛正华, 董小社, 伍卫国, 等. 自适应大规模服务器集群监控系统的构建[J]. 西安交通大学学报, 2008, 42(2): 400-402.

[15] 杨开杰, 刘秋菊, 徐汀荣. 线程池的多线程并发控制技术研究[J]. 计算机应用与软件, 2010, 27(1): 168-172.

编辑 金胡考