自我感知技术在电源管理方面的应用研究

李 康,王 雷,罗燕京
(北京航空航天大学 计算机学院,北京 100191)

摘 要:文中利用现有的自我感知技术,以及电源管理的策略,提出了服务计算当中电源管理的新方案。具体通过自我感知框架感知服务的运行状态,对电源进行动态管理。达到在满足服务性能要求的基础上,对电源资源进行优化管理,缓解如今服务计算当中能耗急速增加的问题。通过实验发现基于自我感知的电源管理方案,能够在满足目标程序性能的基础上,通过调整程序使用的计算节点数目,达到节约电源、优化电源管理的目的。自我感知技术能够从程序的角度更好地反映程序的需求,满足关键程序对性能和资源的需求问题。

关键词:自我感知;电源管理;自适应系统
中图分类号:TP39 **文献标识码**:A **文章编号**:1673-629X(2012)12-0198-05

Research on Self-awareness Technology in Power Management

LI Kang, WANG Lei, LUO Yan-jing
(College of Computer, Beihang University, Beijing 100191, China)

Abstract: It uses existing self-awareness technology and power management strategy to provide a new scheme to manage power in service computing. Through self-aware framework, can dynamically get the state of running service and manage the power to optimize power resources management without performance sacrifice. This can be used to ease rapidly increasing energy consumption problem in service computing. Through experiments find that power management scheme proposed can save power by dynamically reducing computing nodes and meet the performance requirements of target programs. Self-aware technology can better reflect needs of program from the perspective of the program itself and satisfy the performance and resource needs of key program.

Key words: self-awareness; power management; adpative system

0 引 言

之前以分布性、动态性、成长性、自组织性、自适应性和异构性为特点的网络环境促使了面向服务计算(Service-Oriented Computing)这种新的计算泛型的提出与应用。面向服务计算中使用服务作为构建软件系统的基本元素,是针对分布式系统的新型计算模式,为解决资源异构问题以及构造动态、开放环境下耦合的集成化应用带来了诸多便利。面向服务计算将会成为IT领域未来发展的一个重要方向。

不过,由于面向服务的系统越来越复杂,并且松耦合的构架,导致比较低的性能和资源利用率,服务计算的代价较大。这是由于不能在运行时有效预测系统环境变化(例如服务负载的变化)带来的后果,并且按照变化调整系统的配置。

如今,计算机能源的限制变得越来越重要。不管是移动终端还是大规模计算,能源问题越来越突出。尤其是面向服务计算的数据中心的电源消耗正在急剧增长。直到2011年,美国所有的数据中心预测将消耗1000亿千瓦时点,每年的花费大概有74亿美元[1]。值得特别考虑的是,数据中心使用率普遍较低,一般只有在20%到30%之间[2,3]。这样会有很大的空转电源损耗(当前的服务器仅仅使用了60%的峰值功率),造成了大量的浪费。所以,需要对服务计算的服务器电源进行管理。

文中设计了一个电源使用进行动态管理的模型,并且实现了通过任务需求和服务器负载情况给任务动态分配计算节点数目,达到节约电源的目的。

1 相关研究工作

1.1 自我感知技术

自我感知技术的概念来源于自制系统。IBM最早于2001提出了使用自我管理系统[4],用来解决软件复杂度的不断增长以及软件管理成本的增长。自治系统是受人体的自治神经系统(Autonomic Nervous System)

收稿日期:2012-05-15;**修回日期**:2012-08-22
基金项目:国家"863"高技术发展计划项目(2011AA01A204)
作者简介:李 康(1988-),男,硕士研究生,CCF会员,主要研究领域为操作系统;王 雷,博士,副教授,主要研究领域为操作系统、编译技术和软件工程。

的启发。ANS 是神经系统的一部分,控制身体的潜意识行为,比如血液循环、肠胃活动等。这样可以将关注放在有意识的日常生活。

而要实现自制系统[5],自我感知技术是其中的基础。因为,只要系统能够了解自身所处的状态和当前的行为,才能对当前的状态进行评估、分析,进而决定下一步的行为和所要到达的状态。自我感知说明系统能够意识到自己的状态和行为。一般是基于自我监控(self-monitoring)反馈检测到数据来实现的[6,7]。

分析工具和技术同样能够帮助设计理想的感应器[8]。另外一个令人注目的想法是脉搏和心跳监控(Heartbeat and pulse monitoring)[9]、Martina Maggio 等[10],使用 Heartbeats 监测框架来研究了一个自我感知,自我控制的自适应系统的模型。Heartbeats 不仅被程序设计者用来指示程序的性能目标,而且用来测试运行程序当前的执行性能。

1.2　电源管理技术

为了处理空闲能源损耗,研究者认为系统组件消耗的能源应该跟他们的使用率成正比。DVFS(Dynamic Voltage and Frequency Scaling)[11,12]是如今经常使用在处理器上的电源管理技术,这个技术就展示了这个概念。

在现代的计算平台,可以在多个支持的电源状态之间切换,每一个都有不同的电压、平率和电源消耗特点。不幸的是,切换到一个电源状态减少将会减少交付的资源,会影响甚至会使程序失去交付正确响应服务的能力。这个跟程序本身密切相关(例如视频压缩器或搜索引擎),这些程序需要对人类用户提供快速交互服务。

使用软件策略节约能源[13],分为三种:转移(transition),负载改变(load-change),自适应(adaption)。转移问题是决定何时切换到低能耗,削减功能的状态。负载改变是决定如何改变一个组件的负载,可以在未来使用低能耗模式。自适应问题是如何创造软件让它们按照新颖的、低功耗的方式工作。

Henry Hoffmann 等研究了代码穿孔(code perforation)的方法[14]。通过降低程序运行的准确性来增加程序的性能。代码穿孔针对一些对数据准确度和精度要求不是特别高的应用,例如视频编码、三维图像处理、搜索等应用中。通过减少主要迭代循环的数目这种思想,来获得性能上的提高。

对于很多程序可以通过降低结果的准确度(或者质量)来增加性能,这个最终会转化为能源节省。程序经常会暴露一个静态的接口(以配置参数的形式),这样会允许用户来控制程序的性能与服务质量(QoS)的均衡。需要在负载或者交付的计算资源波动的情况下仍可以正常运行,这种情况下,静态的配置就出现问题——结束,然后重新启动程序,修改一些配置参数,这种方式不是大家所想要的或者是不能接受的,对于那些长时间运行的程序或者程序重新启动时间太长。

1.3　存在问题

网络上访问经常会有大量的高峰爆发,而且网络服务的访问通常有时延要求,所以,造成了大量低使用率的机器。现有的电源管理机制经常会有一些不足。通过使用虚拟机进行服务整合,是对于非关键服务的通用做法,但是不能保证能够按照要求的响应速度提供服务。关闭空闲服务器,甚至使用低功耗模式,都会导致同样的问题。

需要建立一种机制,在节约能源的同时能够满足服务的性能要求,提供高质量的服务。因此,需要对服务的性能进行监控,进行动态的资源配置,不仅可以使用现有策略节约电源,而且不影响服务的质量。并且能够在服务负载高峰的时期,优化资源配置,满足服务性能。

2　基于自我感知技术的电源管理设计

2.1　基于自我感知技术的服务执行性能获取

在如今的数据中心中,基于 SOA 模式的数据服务和程序一般都是运行在专门的服务器上,服务器需要保证能够在服务请求峰值时候满足性能的需求。不过,由于提高能源效率和减少操作损耗需求的驱动,公司越来越多的采用了服务器虚拟化和服务器合并的技术。不过,采用虚拟化技术,往往会增加系统的复杂度和动态性。复杂性的增加是由于虚拟资源的引入,和因此导致的逻辑资源与物力资源分配的差距。动态性的增加是由于缺乏对于底层服务硬件的直接控制,而且程序与负载的复杂交互共享了整个物理设施。由于缺乏能力来预测这样的复杂交互,并且很难一致地调整程序来提供稳定的服务质量来满足性能和可用性。服务提供者经常会遇到这样的问题:虚拟设施上新部署的服务应该提供什么样的性能,这个服务需要分配多少资源。将一个服务从一个虚拟机上转移到另外一个虚拟机上的代价是多少?当客户复杂变化的时候如何去调整配置来满足性能?要回答这些问题,必须要能够感知运行服务的运行时性能,并且系统配置的变化和负载变化会对性能产生何种影响。

使用 Application Heartbeat 的自我感知框架提供了一个简单和标准的程序接口,程序能够使用这个接口来查询程序的性能。这个框架不仅能够使程序本身查看自己的性能,而且外部的观察者也能查询。但是,实现对程序的性能监控,必须对程序代码中添加一些额外代码。因为,底层的程序性能的测试。例如,检测一

段时间周期内的指令数目,但是,这个并不能有效的说明这些指令在做有用的工作,计算机或许在做有用的工作,但是也有可能在一个锁上进行忙等待。测试 CPU 利用率或者 cache 命中率有同样的问题。这些方法的缺陷是他们从低层次的机器的性能来推测高层次的程序本身的性能。Heartbeats 的思路是让程序在运行过程中让程序本身来告诉外界自己执行的性能。因此,这种方法有较好的可移植性和通用性。通过在程序的关键循环的外部插入 Heartbeats API 来释放心跳,每执行一次心跳 API,就代表程序向前执行了一步。这种方式能有效的测量到程序本身执行的进度。并且可以使用每秒的心跳数目来衡量和指定程序的性能。

2.2 系统控制模型

考虑一个使用了 heartbeat 框架的系统。用 k 来统计心跳信号的次数。$h_t(k)$ 是在第 $k-1$ 次到第 k 次测试中间消逝的时间。w_σ 用来表示该程序的负载,即在满足最少资源的情况下连续两次心跳的期望值。并且假设负载不随着程序的运行而发生变化,可以认为是一个已知的常量。基于以上的一些假设,控制系统的模型是这个样子。

$$h_t(k) = w_\sigma u(k-1) + \delta w(k-1) \tag{1}$$

这里 δw 表示的是外部的干扰,u 表示的控制信号。控制信号 $u(k)$ 的范围应该在(0,1]。如果 $u(k)=1$ 说明给程序最少的资源。$u(k)$ 的值越小说明给程序的资源越多,来保证程序获得最高的性能。注意,控制信号是依赖于实现策略的,同样的控制信号可能实现的策略是不一样的。比如,一个控制信号可能配置了更多的计算机内核来加快计算速度,另外一个可能是增加了 DRAM 的带宽增加内存受限制的程序执行速度。

使用的模型是非常简单的,但是它却抓住了所有必须的动态信息来适当的控制系统,外部干扰对于负载的影响是没有办法提前预测的。这个模型需要将 $u(k)$ 映射到有效地控制动作中去。映射在概念上是控制机制的一部分,但是,如果没有合理考虑的话,将会使整个模型失效。

寻找一个好的行动策略是一个复杂的问题。为了利用现有的模型分析的优点,有必要描述控制变量 $u(k)$ 和有效行动的关系(有效的行动有:程序使用的计算机内核数目,CPU 频率伸缩机制)。如果这个关系对当前程序无效,控制系统将不能正常运行。

2.3 利用反馈机制的电源管理优化

当获取服务当前运行的状态之后,需要对当前程序的运行状况进行评价。首先,对比当前的运行状态跟目标运行状态。如果,不能满足目标运行状态,则需要自动进行调解。需要利用现有的程序和硬件配置方法,对当前的运行状态进行配置。调整,程序当前的性能,占用的资源,如 CPU 数目、内存等。

使用一种通用的反馈机制模型[6],

$$h(t+1) = b \cdot s(t) \tag{2}$$

$h(t)$ 表示在时间 t 时程序的心跳率,b 是当程序满足基本资源配置时后的执行速度,$s(t)$ 是在时间 t 时对程序执行的加速度。通过给定的模型,控制系统计算在时间 t 时应用的加速度。

$$e(t) = g - h(t) \tag{3}$$

$$s(t) = s(t-1) + \frac{e(t)}{b} \tag{4}$$

$e(t)$ 在这里表示在时间 t 时,程序的执行速度与目标速度的差距,g 表示程序的目标执行速率。

这里,需要程序设计者提供一些可供调解配置的接口。并且,最好能够说明这些接口使用之后,可能对程序执行造成的影响。但是,一般情况下,程序设计者不会提供这些接口。所以,就需要寻找可用的调解程序运行的策略。自我感知技术调整程序运行状态流程如图 1 所示:

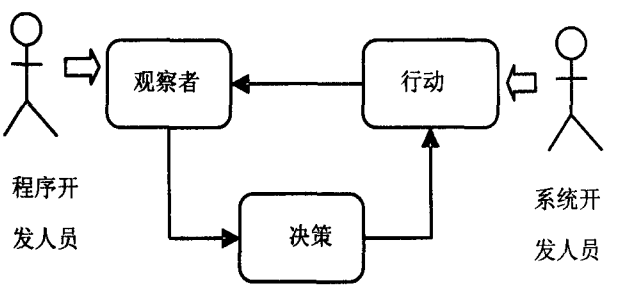

图 1 自我感知技术调整程序运行状态的流程

程序一般都会提供一些可配置的参数,以及这些参数可以调解的范围。这些参数的组合都代表着对程序的不同的控制。通过一组独立的输入,让目标程序在不同的组合状态运行,得到这些状态变化对程序状态的影响。处理这些程序状态的信息,来标记参数可用的配置点,来标识可用的对程序状态的控制点。当需要对程序进行调整来满足目标的时候,可以在这些控制点中选择合适的策略,来达到目标要求。

3 系统实现

设计了一个简单的系统来对自我感知技术的电源管理方案进行验证。使用一个分布式运行环境来运行示例程序,其中,分布式环境有 8 个计算节点。目标分布式程序有一个设定好的性能标准,即在给定的时间间隔的心跳率的范围。将在满足性能目标的前提下,调整服务占用的计算节点的数目。例如,在计算节点负载比较高的时候,就需要为程序分配比较多的计算节点来满足性能要求。当计算节点负载较低的时候,就可以减少程序占用的计算节点,以此降低额外计算

节点带来的损耗。

这种实现的优点是,系统能够在没有人员干预的情况下,自动按照设定好的目标,调整程序的运行状态。在满足的目标的前提下,优化电源的管理,达到节约能源的目的。系统结构如图 2 所示:

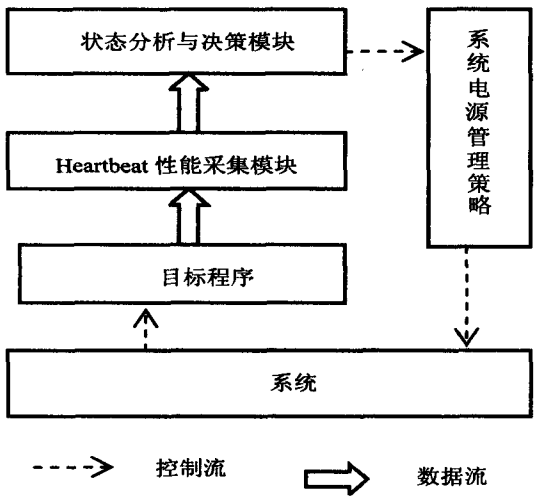

图 2　自我感知电源管理系统结构图

3.1　HeartBeat 性能采集模块

Heartbeat 的实现原理是,通过采集程序关键循环当中的时间戳,来获取程序实际的执行进度。然后,通过度量两次相邻循环的间隔,可以简洁有效地获取程序当前的性能以及执行进度。对于循环执行较快的程序,可以划定一个窗口,规定一个窗口大小,只有当运行窗口数目循环之后,再采集时间点。这样可以避免过多采集数据对目标程序带来干扰。表 1 中列出了 Heartbeat API 函数。

在目标程序中,插入 HeartBeat 的采集点。这些 Heartbeat 节点必须在目标程序运行的过程中的关键循环当中,这样才可以准确地反映程序的执行进度,而且通过周期时间内产生的 heartbeat,来反映程序当前的执行效率。观察程序运行状态优化系统参数如图 3 所示。

表 1　heartbeat API 函数

函数名	参数	作用
initialize	window[int]	初始化心跳系统,指定窗口大小来计算平均心跳率
heartbeat	tag[int]	生成心跳来表示程序的执行
current_rate	windows[int]	返回上一个窗口中的平均心跳率
set_target_rate	max[int],min[int]	设置目标心跳速率范围

3.2　系统参数调节接口

在这个部分,将实现具体如何调整程序与系统状态的接口。包括将任务分发到更多的计算节点上,以及调整具体某个 CPU 的执行模式。具体可调整程序以及系统状态的实现,是实现优化管理的基础。只有可以有效调整系统电源使用的情况,才能最终达到节约电源的目的。而节约电源肯定会在一定程度上影响程序的执行效率。

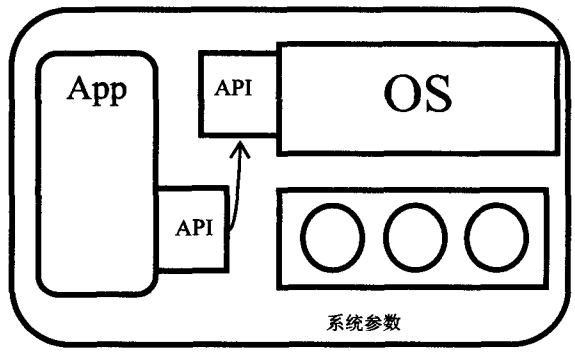

图 3　观察程序运行状态优化系统参数

3.3　状态分析与决策模块

将使用 heartbeat 采集到的效率信息与提前设定好的目标效率进行比较,决定下一个计算周期的激励策略,来满足目标性能。假如 h 是当前观测到的心跳速率,同时 g 是系统的目标心跳速率。同时各种激励策略能够提高执行速率来满足目标心跳速率,当 $h > g$ 时,说明系统的执行速度已经满足了用户的请求。这时候,可以通过一些降低执行速度的策略来降低执行速度,比如,减少分配给该程序的节点数目,或者使用 DFVS 技术降低 CPU 的电压和频率,这些措施在满足程序性能目标的前提下带来能源的节约。

4　实验结果

使用一个分布式运行环境来运行示例程序,其中,分布式环境有 8 个计算节点。目标分布式程序有一个设定好的性能标准,即在给定的时间间隔的心跳率的范围。在目标程序运行的时候,将面临服务器负载的波动,在满足性能目标的前提下,调整服务占用的计算节点的数目。对目标程序设定的目标执行心跳速率是 14 ~ 22 之间,发现目标程序的性能能够满足设定的目标性能,在目标范围中间波动。当任务性能超过目标心跳率范围时候,资源控制器将会相应增加或者减少分配给任务的计算节点,使目标程序满足性能要求。

因为,一个程序不可能独占整个执行环境,服务器上需要不停的接受服务,因为别的服务的干扰,目标程序的性能就可能受到影响,这时候目标程序需要更多的资源来满足其服务性能。

服务的性能不是越快越好,因为,要交付越高速的服务,就需要更多的资源。但是,当服务能够满足用户的情况下,可以考虑节约管理资源,尤其是电源。

在实验中,可以看到自我感知系统可以在服务器负载波动的情况下,通过调整资源满足程序的性能目标,而且在满足性能目标的情况下,优化管理系统资源,节约能源。

5 结束语

文中旨在提出新的针对服务计算的电源管理策略，能够有效利用现有的电源管理策略，又不影响整个程序的执行状况。文中通过使用 heartbeat 技术监控程序的执行状态，使用反馈机制改变程序的运行状态，实现电源管理。

通过实验，发现这种机制可以用来缓解服务计算当中能源消耗急速增加的现状。

参考文献：

[1] EPA, Lawrence Berkeley National Laboratory. Report to congress on server and data center energy efficiency: Public law 109-431[R]. [s. l.]: [s. n.], 2008.

[2] Barroso L, Holzle U. The Case for Energy-Proportional Computing[J]. IEEE Computer, 2007, 40(12): 33-37.

[3] Meisner D, Gold B. PowerNap: elimnating server idle power [J]. ACM SIGPLAN Notice, 2009, 44(3): 205-216.

[4] Hinchey M G, Sterritt R. Self-managing Software[J]. Computer, 2006, 39(2): 107-109.

[5] Khalid A, Haye A M. Survey of Framework, Architecture and Techniques in Autonomic Computing[C]//Fifth International Conference on Autonomic and Autonomous Systems. Valencia: [s. n.], 2009.

[6] Huebscher M C, McCann J A. A survey of autonomic computing-degrees, models, and application[J]. ACM Computer. Sruvey, 2008, 44(3): 1-31.

[7] Kaiser G, Parekh J, Gross P, et al. Kinesthetics extreme: An external infrastructure for monitoring distributed legacy system [C]//Proc of Autonomic Computer Workshop. Seattle: IEEE, 2003.

[8] Liu Hua, Parasha M. A component based programming framework for autonomic applications[C]//Proceedings of the International Conference on Automatic Computing. Prague Czech Republic: IEEE Systems, Man, and Cybernetics Society, 2004.

[9] Sterritt R. Pulse Monioring: Extending the Health-check for the Autonomic GRID[C]//Proceeding of IEEE Workshop on Autonomic Computing Principles and Architectures. Alberta, Canada: IEEE Industrial Informatics, 2003.

[10] Maggio M, Hoffmann H. Controlling software application via resource allocation with the Heartbeats framework[C]//Conference on Decision and Control. Atlanta, GA: IEEE, 2010.

[11] Lorch J R, Smith A J. Software Strategies for Portable Computer Energy Management[J]. Personal Communication, 1998, 5(3): 60-73.

[12] Hoffmann H, Misailovic S. Using Code Perforation to Improve Performance, Reduce Energy Consumption, and Respond to Failures [J/OL]. 2009. http://dspace. mit. edu/handle/1721.1/46709? show=full.

[13] Kounev S, Brosig F. Towards self-aware performance and resource management in modern service-oriented system[C]// Proceedings of the International Conference on Services Computing. Miami, FL: IEEE, 2010.

(上接第 197 页)

WebService 服务器的设计将加入混合型 Agent 的思想，即自主地反应、推送、处理信息的思想[13]。而移动平台的开发因为服务器设计中 WebService 的便捷性，可扩展至 iOS 等系统。

参考文献：

[1] 罗辉琼，聂瑞华，林怀恭，等. 基于 SOA 架构的数字校园研究与实现[J]. 计算机技术与发展，2009，19(5)：224-227.

[2] 方蔚涛，杨 丹，李 珩，等. 数字化校园信息门户的设计研究[J]. 计算机科学，2007(3)：9-11.

[3] 王晨辉. 基于 Android 平台校园信息发布系统[J]. 数字技术与应用，2010(8)：123-123.

[4] Fabien B, Valérie M. A concrete solution for Web services adaptability using policies and aspocts[C]//Proceedings of the 2nd International Conference on Oil Service Oriented Computing. New York, NY, USA: ACM, 2004: 134-142.

[5] 王 健，冯锡炜，张永攀. 基于 WAP 技术的校园手机网的设计与实现[J]. 科学技术与工程，2011，11(10)：2338-2341.

[6] Philipp L. The strategic impact of service oriented architectures[C]//Proceedings of the 14th Annual IEEE International Conference and Workshops on Engineering of Computer-based Systems. Washington, DC, USA: IEEE Computer Society, 2007: 475-484.

[7] 李 菲，张新家，袁 林. 基于 Web Services 的群组数据交换系统的研究与实现[J]. 计算机技术与发展，2011，21(12)：186-190.

[8] 焦方俊. 基于 Web Services 的校园信息系统集成平台的研究与设计[D]. 厦门：华侨大学，2007.

[9] 刘壮业，姚 郑. 基于 Web 服务的教师管理系统的设计与实现[J]. 中国科学院研究生院学报，2009，26(1)：127-130.

[10] Gosu A K. Analysis of Web Services on J2EE Application Servers[D]. Texas: University of North Texas, 2004.

[11] 岳 昆，王晓玲，周傲英. Web 服务核心支撑技术：研究综述[J]. 软件学报，2004，15(3)：428-442.

[12] 朱永生，王 军. 基于 Web 内容的一种数据自动抽取方法[J]. 计算机技术与发展，2012，22(5)：87-89.

[13] 刘大有，杨 鲲，陈建中. Agent 研究现状与发展趋势[J]. 软件学报，2000，11(3)：315-321.