

基于 RAPL 的机群系统功耗限额控制

刘 嵩, 刘 轶, 杨海龙, 周彧聪

(北京航空航天大学 计算机学院, 北京 100191)

摘 要: 功耗管控是高性能计算系统和分布式数据中心管理的热点问题。当机房供电受限时需要对机群系统的功耗上限进行控制, 使有限的电力适应供电容量的动态变化。为此, 设计并实现一个基于 RAPL 的功耗限额控制系统。建立机群系统功耗模型, 利用 RAPL 对 CPU 功耗限额的控制能力并结合功耗差额测量方法, 将机群系统功耗上限控制在设定限额内, 在此基础上尽可能减少程序性能的损失。实验结果表明, 在较小的性能损失下, 该系统可有效降低峰值功耗并将其稳定在限额内。

关键词: 高性能计算; 分布式数据中心; 峰值功耗; 功耗限额; 差额测量; RAPL 技术

中文引用格式: 刘 嵩, 刘 轶, 杨海龙, 等. 基于 RAPL 的机群系统功耗限额控制[J]. 计算机工程, 2017, 43(5): 40-46.

英文引用格式: Liu Song, Liu Yi, Yang Hailong, et al. Power Capping Control for Cluster System Based on RAPL[J]. Computer Engineering, 2017, 43(5): 40-46.

Power Capping Control for Cluster System Based on RAPL

LIU Song, LIU Yi, YANG Hailong, ZHOU Yucong

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

[Abstract] The management and control of power has already become a hot issue in the area of management of High Performance Computing (HPC) system and distributed data center. In order to adapt the limited power to the dynamic change of the power supply capacity when the supply of energy is limited in the computer room, it is necessary to control the upper power limit of cluster system. Aiming at this problem, this paper designs and realizes a power capping control system based on RAPL. By constructing the power model of cluster system, utilizing RAPL's capability of controlling the power consumption limit of CPU and combining the method of measuring the difference of power, it sets the upper limit of energy consumption of cluster system within the previously set power cap. On this basis, it tries to reduce the losses of performance as much as possible. The result of experiment shows that this system can reduce the peak power effectively with slight performance and keep it below the power cap stably.

[Key words] High Performance Computing (HPC); distributed data center; peak power; power capping; difference measurement; RAPL technology

DOI: 10.3969/j.issn.1000-3428.2017.05.007

0 概述

随着高性能计算和云计算技术的快速发展, 机群系统能耗增长迅速。过高的功耗不但增加了系统运行成本, 也给散热/冷却设备带来了更多负荷, 同时高功耗产生的较高工作温度还会增加芯片等部件的故障率, 进而影响到系统的可靠性和可用性。因此, 功耗管控已成为近年来高性能计算系统和分布

式数据中心管理的热点问题之一。

低功耗管控的目的是在对性能不影响或影响较小的前提下尽可能降低系统的能耗。服务器/计算节点的功耗在不同的应用负载下呈现较大波动, 以当前主流双路服务器为例, 其空闲状态下功耗为 130 W 左右, 而当运行计算密集型应用时, 峰值功耗超过 300 W, 这将给机房供电系统带来较大压力。针对这一问题, 人们提出了功耗限额控制技术^[1], 其

基金项目: 国家“863”计划重大项目(2012AA01A302)。

作者简介: 刘 嵩(1989—), 男, 硕士, 主研方向为计算机系统结构、分布式系统、功耗控制技术; 刘 轶, 教授、博士; 杨海龙, 讲师、博士; 周彧聪, 硕士。

收稿日期: 2016-05-20

修回日期: 2016-06-24

E-mail: song.liu@buaa.edu.cn

目的是将系统运行功率控制在设定的上限值之下。该技术常被用于机房供电受限,而设备的总峰值功耗超过供电能力的情形(如设备扩容升级后^[2])。近年来,随着风能、太阳能等可再生能源供电的兴起,也要求系统具备功耗限额控制的能力,以适应供电容量的动态变化。

本文通过使用 Intel RAPL 技术,设计实现一个针对机群系统的功耗上限控制系统。采用 RAPL 实现处理器运行功耗的限额控制,并通过测试并程序在不同 RAPL 限额下对系统总功耗的影响,建立基于 RAPL 的功耗限额控制模型,将机群系统的功耗控制在设定的限额内,以最大程度发挥程序的性能。

1 功耗控制

目前的功耗管控技术主要分为动态电压频率调整技术(Dynamic Voltage and Frequency Scaling, DVFS)、虚拟化技术和关闭/休眠技术,而对功耗限额的控制通常采用 DVFS 技术。

1.1 常用功耗管控技术

DVFS 技术是根据芯片所运行的应用程序对计算能力的不同需求,动态调整芯片的运行频率和电压,从而达到节能的目的^[3]。根据 CMOS 电路动态功率公式: $P = \alpha \times F \times V^2$ 可知,动态功率与频率和电压的平方成正比^[4],因此,降低处理器的频率可以降低处理器的动态功耗。而对于同一芯片,频率越高,需要的电压也越高,所以,对电压和频率的调整需要同步进行。Linux 内核中的 CPUFreq 子系统即采用 DVFS 技术^[5]。

虚拟化技术是一种资源管理技术,是将计算机的各种实体资源抽象出来,并以更好的方式重新分配。比如将一台计算机的资源划分为多个虚拟机,可以实现多个任务在同一计算机的不同虚拟机上运行,以提高资源利用率,减少所需计算机数量,进而降低能耗^[6]。DVFS 技术和虚拟化技术主要用于降低执行功耗(即有任务运行时的功耗)。

关闭/休眠技术可以通过关闭或者休眠一部分空闲的节点,让系统资源可以更加集中地被利用,降低空闲能耗,其主要优点是可以最大限度地降低空闲能耗,缺点是计算机在不同状态之间的转换需要较长时间,从而导致性能的下降^[7]。

针对功耗限额的控制技术要求对功耗的变化能够迅速地反应,因此,在这个领域 DVFS 技术使用得较多。传统 DVFS 功耗控制方案可以通过改变 CPU 的频率来降低系统整体的功耗,但未对 CPU 功耗的变化与机群总功耗变化量之间的关系进行讨论^[8]。采用 DVFS 可以建立总功耗和 CPU 频率之间关系的模型,进而通过给定的功耗限额计算出对应的

CPU 频率^[9],但由于 CPU 可设定的频率等级有限,因此无法进行更细粒度的调节。此外还有研究者结合 DVFS 和空闲节点的方法来进行功耗限制^[10-11],但是同样存在频率等级和空闲等级有限的问题。而单纯采用调节频率的方法也不能较好地维持功耗稳定,减少性能的损失^[12]。

1.2 RAPL 技术

RAPL 是 Intel 提出的一种功耗监测控制接口,最早应用于 Sandy-Bridge 处理器家族,可以监测和限制 CPU 在一定时间间隔内的平均运行功率。它结合自动的动态电压频率调整和时钟节流技术将功耗维持在用户设置的限额之下,RAPL 采用了一个功耗控制的内部模块,可以在一定时间窗口内估计平均的功耗,使用 DVFS 尽可能地让当前的功耗接近给定的功耗限额,并使用时钟节流技术精确地执行。RAPL 支持 4 个不同域(Domain)的功耗监测和控制,包括 PP0(CPU 核心与 Cache)、PP1(CPU 非核心部件,如 GPU)、Package(整个 CPU)和 DRAM(内存)^[13]。

RAPL 对传统的 DVFS 技术进行了 2 个主要方面的改进:1)整合了芯片内功耗的监测和控制,使其更精确和快速地识别和适应负载的变化;2)通过结合 DVFS 和时钟节流技术,能够提供比单纯 DVFS 更好的性能以及更细粒度的管控策略。RAPL 可以充分利用 DVFS 和时钟节流 2 种技术的优势使平均功耗尽可能维持稳定。当功耗限额设定在 DVFS 支持的范围内时,RAPL 主要采用 DVFS 来实现功耗调节,因而可以达到和 DVFS 相似的性能。当功耗限额超过 DVFS 控制的范围时,RAPL 仍然能够实现功耗限额控制,即使此时的性能损失可能更大^[12]。

目前,RAPL 功能已经集成在最新的 Linux 内核中。在硬件实现中,RAPL 通过 rdmsr 和 wrmsr 指令访问特殊模块寄存器(Model Specific Registers,MSRs)来实现能耗的监测和功耗限额的写入^[14]。MSRs 的作用主要是设置 CPU 的工作环境和标示 CPU 的工作状态,包括温度控制、性能监控等,并能够提供 PP0 和 PP1 的能耗状态。图 1 是 RAPL 每个域支持的主要功能,其中有一部分功能是可选的^[15]。

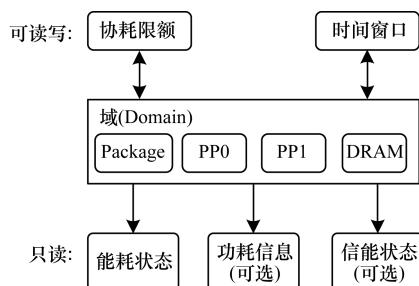


图 1 RAPL 的主要功能

在 CentOS 7 中, RAPL 针对每个独立的 CPU 设置了一套功耗控制系统。在 RAPL 框架下, 每个 CPU 整体为一个 Package, 对于每个 Package 的功耗可以设置 2 个限额, 由 $constraint_X_power_limit_uw$ ($X=0,1$) 参数表示。其中, $X=0$ 是长期 (long term) 限额, 表示在 long term 时间窗口内的平均功耗; $X=1$ 是短期 (short term) 限额, 表示在 short term 时间窗口内的平均功耗。对于时间窗口, 可以通过修改 $constraint_X_time_window_us$ ($X=0,1$) 参数改变长期和短期的时间窗口的大小。此外, RAPL 还可以读取每个 Package 当前的能耗。

2 基于 RAPL 的功耗限额控制方法

2.1 问题分析

为实现节约机群电力、降低功耗峰值的目标, 需要一种针对整个机群的功耗限额控制方法。机群由若干节点及一些外围设备构成, 例如制冷设备、网络设备 (如交换机) 等, 节点则由处理器、内存、硬盘等部件构成。由此可见, 想要控制整个机群的功耗, 需要考虑多方面因素。本文利用这些因素为整个机群建立功耗模型, 并在该模型的基础上讨论功耗限额的控制策略。

首先对单个节点建立功耗模型。每个节点的总功耗都是由处理器、内存、硬盘、主板、网卡、风扇等部件的功耗组成。因此, 用 n 表示机群系统节点的个数, 用 m 表示每个节点处理器的数量, 并将节点 i 中处理器 j 的功耗设为 $P_{cpu}(i, j)$ 。对于每个节点 i , 需要考虑的功耗参数分别为处理器总功耗 $\sum_{j=1}^m P_{cpu}(i, j)$ 、内存功耗 $P_{mem}(i)$ 、硬盘功耗 $P_{disk}(i)$ 、主板功耗 $P_{board}(i)$ 以及风扇等其他功耗 $P_{other}(i)$, 则整机功耗 $P(i)$ 可以由下式表示:

$$P(i) = \sum_{j=1}^m P_{cpu}(i, j) + P_{mem}(i) + P_{disk}(i) + P_{board}(i) + P_{other}(i)$$

对于拥有 n 个节点的机群系统, 除了要考虑每个节点的功耗外, 还要加上连接各个节点端交换机的功耗 P_{switch} , 因此, 机群总功耗可以表示为:

$$P = \sum_{i=1}^n P(i) + P_{switch}$$

在给定整个机群的功耗限额时, 需要知道每个节点的各个部件分配多少功耗以及交换机需要多少功耗, 然而在实际应用中, 对所有功耗一一进行控制并不现实。首先, 内存、硬盘等部件功耗难以单独测量; 其次, 这些部件的功耗也会受到 CPU 效率的影响, 单独控制可能会影响程序运行效率, 产生较大的性能损失。为此, 本文根据实验提出一种功耗控制方案, 仅通过控制 CPU 的功耗实现对整个机群系统

功耗的控制。

2.2 解决方案

由于处理器的功耗随着处理器工作频率和利用率的变化而改变, 波动很明显, 而其他部件的功耗波动相对较小, 因此总功耗可以看作由动态功耗和静态功耗两部分组成, 处理器功耗作为总功耗的动态部分, 其他部件功耗作为静态部分。这样只需对动态功耗加以控制, 就可以很大程度上影响整个机群的功耗。所以, 本文将 CPU 以外静态功耗作为一个整体进行考虑, 则对于拥有 n 个节点的机群, 机群总功耗为:

$$P = \sum_{i=1}^n \sum_{j=1}^m P_{cpu}(i, j) + P_{other}$$

在给定总功耗时, 为了让机群始终运行在该功耗之下, 首先要知道应当给每个 CPU 分配多大的功耗。为了清晰表示机群总功耗与 CPU 功耗的关系, 根据负载均衡原则, 本文为每个 CPU 设置相同的功耗, 则机群整体的功耗可以简化为 $P = n \times m \times P_{cpu} + P_{blc}$ 。其中, P_{blc} 为机群总功耗与所有 CPU 功耗的差值, 定义为功耗差额。

在 CPU 功耗控制技术中, 钳制效果比较好的是 RAPL 技术^[12]。RAPL 技术可以稳定而有效地限制每个 CPU 的功耗^[16], 因此, 在已知机群总功耗时, 如果能够确定功耗差额的大小, 则可通过公式 $P_{cpu} = (P - P_{blc}) / (n \times m)$ 来确定每个 CPU 的功耗, 从而使用 RAPL 技术控制 CPU 功耗, 实现限制机群总功耗的目的。

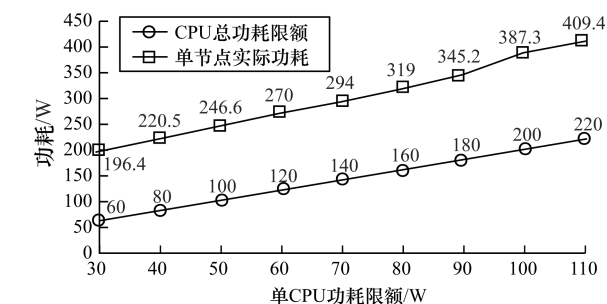
为获得机群总功耗与 RAPL 限额 (即单 CPU 功耗) 之间的关系, 确定功耗差额的大小, 本节针对不同的 RAPL 限额, 采用测量比较的方法进行总功耗的测试实验。实验方法是在节点端运行高性能并行测试程序, 以充分利用每个节点主机的所有核心, 使其达到满载的状态。在 CPU 满载的情况下, 通过改变 RAPL 的功耗限额, 分别测试不同 CPU 功耗与机群总功耗之间的关系, 从而建立 CPU 功耗与机群总功耗之间的关系模型。

实验机群包括 4 个节点, 每个节点有 2 个 CPU。实验中实时监控机群总功耗, 并执行以下步骤 (以下 RAPL 限额均针对每个 CPU):

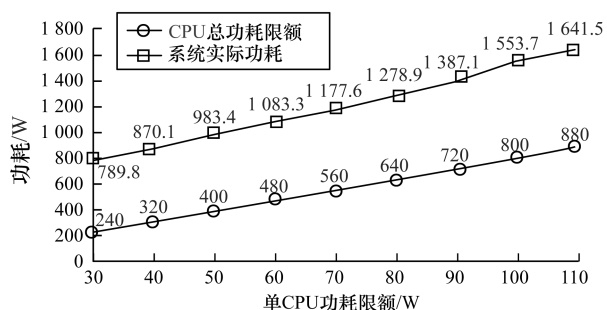
- 1) 将 RAPL 限额设置为每个 CPU 10 W;
- 2) 运行程序, 记录功耗最大值;
- 3) 以 10 W 为单位增加限额;
- 4) 重复步骤 2) 和步骤 3), 直到总功耗不再受限额影响;
- 5) 重复以上实验 3 次, 取平均值。

在单机和机群 (4 个节点, 共 8 个 CPU) 上分别进行以上实验, 根据实验所得数据, 绘出系统实际功耗和 CPU 总功耗的对比关系。

如图 2 所示,无论对单机还是机群,系统实际功耗与 CPU 总功耗呈现相同的变化趋势,并且有稳定的差值。尽管对于具有 4 个节点的机群 CPU 总功耗从 240 W 提升到了 880 W,但是机群中其他部件产生的功耗仅变化了 211.7 W。因此,只要能限制 CPU 的功耗,通过一定的换算就可以有效控制整个机群的功耗。



(a) 单机 CPU 总功耗限额与实际功耗关系



(b) 四节点机群 CPU 总功耗限额与实际功耗关系

图 2 CPU 总功耗限额与实际功耗的关系

对于每个机群,由于节点数量和配置的不同,总功耗与 CPU 功耗之间的关系也不一样。本文方法在功耗转换方法的基础上,提供机群总功耗和 CPU 功耗之间差额的动态测量功能,使得对于任意给定的功耗上限,都可以根据这个模型获得相应的 RAPL 限额,从而实现对整个机群的功耗限额控制。

3 基于 RAPL 的功耗控制系统实现

为更好地监控机群的实时功耗并便捷地控制机群的总功耗限额,本文开发一个机群功耗监控软件来实现这些功能。该软件包括监控端与节点端两部分,监控端为 Windows 系统下采用 C#编写的功耗查看和控制软件,节点端部署在每个节点 Linux 主机上,用来监听监控端发来的指令并执行查看或修改主频等参数的功能。

图 3 是机群功耗监控软件与机群系统各个节点和功率仪的交互关系图,监控端和节点端采用异步 Socket 实现通信,监控端和功率仪采用串口或者 SNMP 协议进行通信。

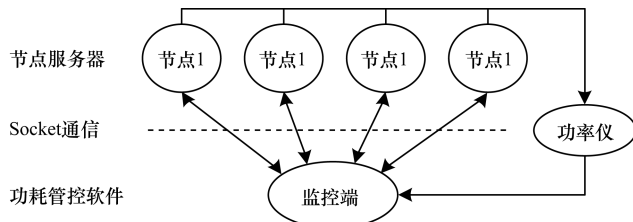


图 3 功耗管控系统结构

除了检测当前机群功率状态和各个节点的运行状态外,该软件还提供差额分析和限额设置功能,分别通过差额分析模块和功耗控制模块实现。在监控端可以为整个机群设置功耗上限,系统的功耗分析模块会根据当前机群的状态计算出分配给每个 CPU 的功耗大小,并设置 RAPL 限额,从而达到限制整个机群功耗的作用。图 4 是机群功耗管控系统各个模块的关系示意图。

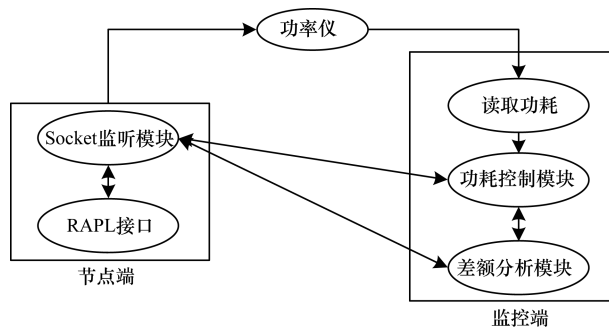


图 4 功耗管控系统模块关系

图 5 为机群功耗管控软件的运行界面,在软件中点击“读取”可以获得机群的实时功耗。在限额处输入为机群设置的功耗上限后,若为首次使用,则需勾选“分析差额”,开启差额分析功能,程序会根据 3.1 节差额分析模块给出的算法分析机群系统的最大差额(或差额表)并保存到配置文件中。当再次运行程序设置功耗限额后,系统会根据差额(或差额表)自动计算出每个 CPU 应当分配的功耗,并采用 RAPL 设置功耗限额。

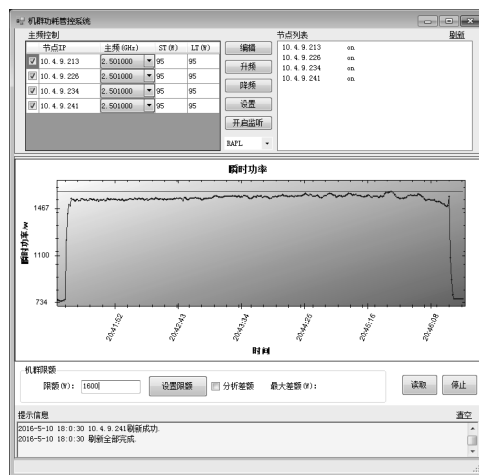


图 5 功耗管控系统用户界面

3.1 差额分析模块

差额分析模块主要的功能是在给定机群总功耗的情况下,对最大功耗差额的测量与分析。通过上文测试实验可以看出,机群功耗和 CPU 总功耗的差额随着 RAPL 限额的增长而不断增长,但总体变化范围不大,因此,只要能获得最大的功耗差额,就可以利用 RAPL 技术限制整个机群的功耗上限,其中计算密集型程序的差额分析效果最佳。

为获取任意机群的最大功耗差额,本文在机群功耗管控系统中加入了功耗差额分析功能。当用户在机群功耗管控系统中开启差额分析功能后,初始状态将 RAPL 上限设定在一个较低水平(默认 60 W,可在配置文件中修改),并给定一个最大允许的功耗限额,然后运行测试程序,通过比较当前实时功耗和功耗上限之间的关系,系统自动调节 RAPL 上限,使得实时功耗不断逼近功耗限额,并计算出整个过程中最大的功耗差额作为差额的基准值。图 6 为本文采用到的差额分析方法流程,其中, N 为集群 CPU 个数; $P_{\max}$ 为最大允许功耗; P_b 为当前功耗差额; $P_{b_{\max}}$ 为最大功耗差额; P_r 为当前 RAPL 限额。

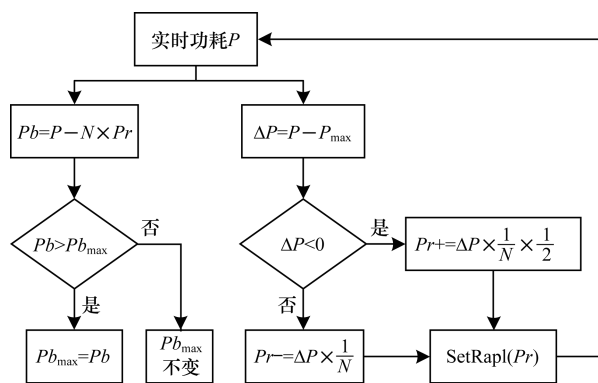


图 6 差额分析方法流程

假设此时功耗上限为 P_{cap} ,最大功耗差额为 P_{blc} ,则对于拥有 $n \times m$ 个 CPU 的机群,每个 CPU 的 RAPL 限额为 $P_{\text{rapl}} = (P_{\text{cap}} - P_{\text{blc}}) / (n \times m)$ 。此时对于给定的功耗限额,均可以保证总功耗在限额以下。

该模块提供 2 种差额分析方案,一种是“唯一差额法”:对于一个机群,确定一个唯一最大的功耗差额,无论使用中功耗上限设置为多少,都采用这个差额确定 RAPL 限额;另一种是“分段差额法”:将功耗分为几个连续的区间,求出每个区间内的最大功耗差额,当用户设置机群总功耗时,根据该数据所在区间选取最大功耗差额来计算 RAPL 限额。

3.2 功耗控制模块

功耗控制模块的主要功能是根据用户提供的机群总功耗限额 P_{cap} ,以及由差额分析模块得出的最大

功耗差额 P_{blc} ,通过 $P_{\text{rapl}} = (P_{\text{cap}} - P_{\text{blc}}) / (n \times m)$ 计算出适合的 RAPL 限额,并对所有 CPU 进行统一设置。这样可以保证机群始终运行在该限额以下,并尽可能维持功耗稳定,降低程序性能的损失。同时,如果因为某些特殊情况导致功耗仍然超过给定的限额,则功耗控制模块会自动降低 RAPL 限额并更新功耗差额数据。

功耗控制模块和差额分析模块均采用 Socket 接口将 RAPL 限额发送给节点端,节点端通过修改 `constraint_X_power_limit_uw` ($X = 0, 1$) 参数的值来实现 RAPL 限额的设定,并将设置成功后的结果返回给监控端。

4 实验结果与分析

本文实验包括差额分析实验和功耗控制实验,差额分析实验采用分段差额法测量各个区间的最大差额,作为后续功耗控制实验中的基准差额。功耗控制实验采用 Linpack 验证模型的功耗控制效果,并比较使用限额控制前后的性能损失。

4.1 实验环境

实验的环境包括被测系统(即被测的节点端)、功率仪和监控端。监控端通过读取功率仪中的数据获得实时功率和总体能耗。功率仪采用 GDW1200C 电参数测量仪,是采集被测系统实时功率和能耗的专业设备。

监控端系统为 Windows 7,编程环境为 Visual Studio 2013。机群包含 4 个节点,系统均为 CentOS 7 (内核: Linux 3. 10. 0-327. el7. x86_64, 处理器: Intel(R) Xeon(R) CPU E5-2680 v3 @ 2. 50 GHz, 双路 CPU, 共 24 个核心)。

网络环境:采用 InfiniBand 高速互连网络连接。

测试程序:实验中采用 Intel MKL 安装包中 Intel® Optimized Linpack Benchmark 作为测试程序。

4.2 实验方法

根据负载平衡的原则,可以认为每个 CPU 处于平等的地位,因而对它们设置相同的功耗限额。同时,为了体现 RAPL 限额对功耗的钳制效果,这里将长期和短期的平均功率设为相同值,并且不改变默认的时间窗口大小。

4.2.1 差额分析实验

差额分析需要在节点端运行高性能并行程序,来充分利用每个节点主机的所有核心,使其达到满载的状态。同时设置当前机群可能达到的最大功耗上限。实验开始前监控端打开监控端软件,与节点端分别建立连接,并开启实时功率监控和差额分析功能,此时系统开始自动分析并计算最大的功耗差额。

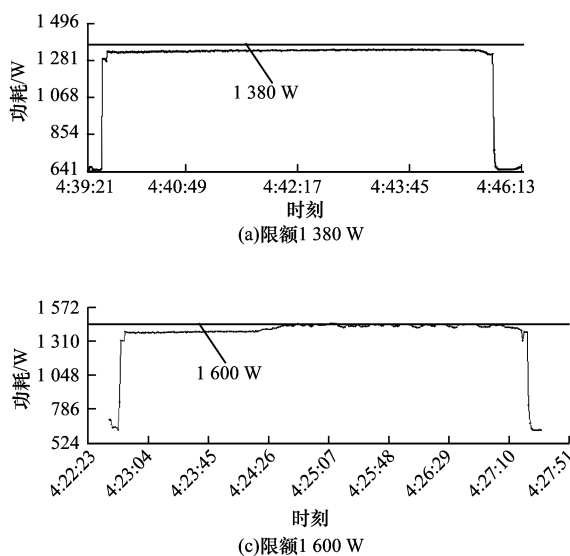
本实验采用分段差额法,测量总功耗从 1 000 W 到 1 700 W 每隔 100 W 的最大功耗差额,并记录到配置文件中,这样当再次运行程序时,可以根据总功耗限额所在区间最大的功耗差额来计算 RAPL 限额,由于每个区间的平均功耗差额不到 30 W,因此这样选取功耗差额既能避免频繁地调节 RAPL 上限,又能较大程度地减少程序性能损失。

测试程序采用 Intel LINPACK,选取任务规模 $N = 81\ 920$, $NB = 192$, $P \times Q = 12 \times 8$,运行时 Linpack 开启 96 个进程,充分利用 4 个节点的 96 个核。

4.2.2 功耗控制实验

实验建立在差额分析实验获得的最大功耗差额表的基础之上,根据用户设定的功耗上限计算出适当的 RAPL 限额,并分配给每个 CPU。

测试程序采用 Intel LINPACK,选取 $N = 100\ 000$, $NB = 192$, $P \times Q = 8 \times 12$ 。选取机群功耗上限分别为 1 380 W, 1 530 W, 1 600 W, 1 650 W,验证功耗限制效果并计算性能损失。



4.3 结果分析

4.3.1 差额分析实验

由差额分析实验可以得出各个功率区间的最大功耗差额,如图 7 所示。将该数据存放在机群管理系统的配置文件中,以便功耗控制实验使用。

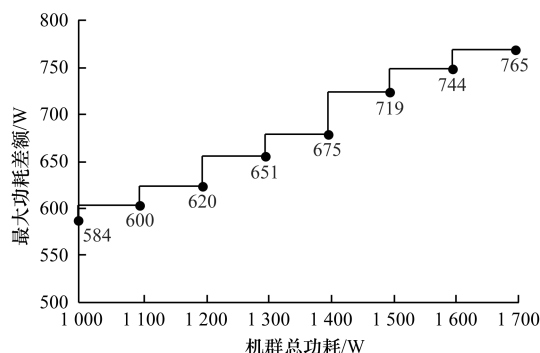


图 7 分段差额法功耗-差额关系图

4.3.2 功耗控制实验

图 8 为功耗管控系统监测到的功耗限额控制实验效果,分别展示了 1 380 W, 1 530 W, 1 600 W 和 1 650 W 限额下机群的实时功耗曲线。

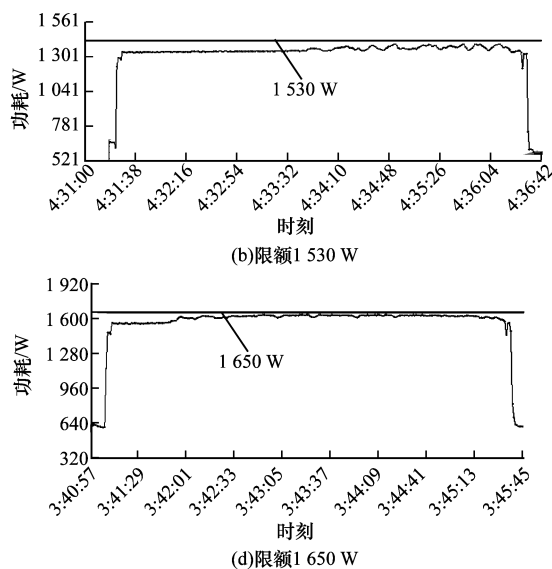


图 8 功耗限额实验结果

本实验针对不同的机群功耗限额,选取相应区间最大的功耗差额,并计算出分配给每个 CPU 的 RAPL 限额。经实验表明,4 种情况下机群的总功耗均维持在给定限额以下,且基本保持稳定,其中当限额为 1 600 W 时,由于恰好处在区间的端点处,此时最大功耗最接近限额。

对于每次实验,表 1 记录了 Linpack 的运行时间和 GFLOPS,设机群总功耗限额为 cap 时的性能损失率为: $\eta = 1 - GFLOPS_{cap} / GFLOPS_{none}$,则可以随着限额的降低, Linpack 的性能损失不断增加。但是

在一定的性能损失范围内,基于 RAPL 的功耗限额控制方法可以很大程度上降低功耗峰值。

表 1 功耗限额控制实验数据

机群功耗 限额/W	功耗峰值 /W	平均功耗 /W	Linpack 运行时间/s	GFLOPS	性能损失 /%
1 380	1 371.1	1 365.2	311.32	2 141.49	19.45
1 530	1 510.5	1 492.8	292.38	2 280.15	14.23
1 600	1 598.7	1 579.8	276.53	2 410.85	9.31
1 650	1 643.8	1 619.2	271.96	2 451.41	7.79
无限额	1 842.7	1 763.1	250.78	2 658.44	-

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 Platforms[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, Völz 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[J]. 计算机应用与软件, 2010, 27(1): 168-172.

编辑 金胡考