软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(2):200−214 [doi: 10.3724/SP.J.1001.2012.04144]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种云计算环境下的能效模型和度量方法*

宋 杰, 李甜甜, 闫振兴, 那 俊, 朱志良+

(东北大学 软件学院,辽宁 沈阳 110819)

Energy-Efficiency Model and Measuring Approach for Cloud Computing

SONG Jie, LI Tian-Tian, YAN Zhen-Xing, NA Jun, ZHU Zhi-Liang+

(Software College, Northeastern University, Shenyang 110819, China)

+ Corresponding author: E-mail: zzl@mail.neu.edu.cn

Song J, Li TT, Yan ZX, Na J, Zhu ZL. Energy-Efficiency model and measuring approach for cloud computing. *Journal of Software*, 2012,23(2):200−214. http://www.jos.org.cn/1000-9825/4144.htm

Abstract: This paper presents an EE (energy efficiency) model and measuring approach for cloud computing. A mathematical expression of EE is first defined, as well as the measuring and calculation approaches, and the extreme conditions of EE are deduced. Then, to facilitate calculating EE, the mathematical expression between the computer power and the CPU working state is improved, thus EE can be calculated through CPU usage and CPU frequency, which simplifies the measurement of EE. In addition, the study designs and implements a series of experiments to verify the correctness of the proposed model. CPU-intensive, I/O intensive and interactive jobs are performed in both stand-alone environment and cloud environment to evaluate their EEs and to summarize their features as well as the optimization approaches. Both the theory and experiments have proved that the proposed EE model and measuring approach can accurately evaluate the EE of cloud systems and lay the foundation for EE optimization.

Key words: cloud computing; energy saving; energy efficiency model; energy efficiency measuring; energy calculation approach

摘 要: 提出一种云计算环境下的能效模型和度量方法.首先定义了能效的数学表达及其测量和计算方法,并推导出了能效最大值的发生条件;其次,为方便能效计算,改进了计算机功率和 CPU 工作状态之间关系的数学表达,通过 CPU 使用率和频率来计算能效,从而简化了能效测量方法.此外,还设计并实施了大量实验,验证了提出的能效模型的正确性;同时对单机环境,云计算环境中 CPU 密集型、I/O 密集型和交互型运算进行能效评估,总结其能效规律和优化办法.理论和实验证明,所提出的能效模型和计算方法能够准确地评估云系统的能效,并为能效优化奠定基础.

关键词: 云计算;节能;能效模型;能效度量;能耗计算方法

中图法分类号: TP316 **文献标识码**: A

计算机能耗问题刻不容缓,各种数据表明,计算机正在吞噬大量能源.云计算时代的到来并不会减少 IT 资

* 基金项目: 国家自然科学基金(61173028); 辽宁省自然科学基金(200102059)
收稿时间: 2011-07-17; 修改时间: 2011-09-06; 定稿时间: 2011-11-14

源,相反,计算机能耗在未来 10 年还会快速增长.一方面,硬件价格逐年降低,人们可以花费更少的经费购得更强大的服务器;另一方面,对云计算这种大规模计算模式的需求也将会迅速增加,需要更多硬件的投入.表面上看,这是一个良性发展,然而潜在的问题是,随着能源价格的逐年升高,廉价的基础设施将带来昂贵的能源开销.如果硬件能耗费用超过了硬件价格本身,则将极大地阻碍 IT 行业的发展.摩尔定律引发了计算机以高运算速率和高能耗为导向的发展,但未必符合环境成本,硬件成本的下降和能效(运算/能耗)的提高没有必然的联系.可以预测,高能效云计算将成为未来 10 年最为迫切、最具挑战性的研究课题之一.

尽管云计算被认为是一种绿色计算,但其本身并没有提供成熟的解决方案来评价和降低能耗,仍需要一种高能效方法来切实地实现绿色计算.对高能效云计算的研究,首先是要找到一种能效的度量模型和测量方法.我们迫切需要一个基准来衡量云计算的能效.我们拟制定一个能效模型和测量方法;现有软件性能评价模型都与运算速度和存储两个指标关联,没有考虑到能效因素,而目前的高能效研究也多采用单一的功率作为能耗模型.而本文拟确立的模型将从运算和能耗这两个角度来衡量软件的性能;有了能效模型,服务提供商可以准确地计算能源成本,服务使用者也准确地按需使用,按需付费,同时也为能效优化提供了依据.

本文着重研究能效的度量方法,包括能效的数学表达、测量方法、计算方法和理论最大值的发生条件,以及云系统的能效规律.我们定义能效是能源和效率之间的比值,测量能效最重要的是测量单位时间内处理的任务量和耗费的电能.对于前者,可以通过测量 CPU 运算时间和运算频率得出.对于后者的测量,存在如下困难:首先,计算机的有功功率不同于额定功率,其值是动态变化的,无法通过简单的计算得到;其次,一些电量测量设备都有固定的测量范围,无法适应云系统中大量节点;再次,云中分布式的节点环境也使电量计或电量传感器的安装复杂化;最后,若针对云中所有节点统一安装电量测量仪器,则无法区分每个节点的实际耗电量,测量粒度过粗,不利于能效优化算法的研究,若为每一个节点单独安装测量仪器,则需要大量仪器,且仪器间的数据和时钟同步也存在难题.因此,本文在能效模型的基础上提出了一种能效的计算公式,通过测量节点的CPU工作状态来计算其实时功率,进而计算能效.我们依据能效计算公式推导了能效最大值的存在和发生条件.我们组织了大量实验,首先研究一台计算机的能效规律,随后,基于 Hadoop 和 MapReduce 编程模型,对云计算中 CPU 密集型、I/O 密集型和交互型运算进行了不同层面的能效评估,总结其能效规律,证明了所提出的能效模型和计算方法的有效性和准确性,并提出了能效优化方法.

本文第 1 节介绍相关工作.第 2 节介绍能效模型的数学表达.第 3 节介绍能效的测量方法和计算方法,并从数学上推理能效最大值的发生条件.第 4 节首先通过实验推出能效计算公式中的参数并在单机环境下加以验证,随后在云计算环境中,通过对 CPU 密集型、I/O 密集型和交互型 3 种不同运算的能效计算和测量,验证能效公式和计算方法,并总结云计算中能效规律;最后对能效优化的思路和方法进行总结.第 5 节总结全文并提出进一步工作.

1 相关工作

2005 年以后兴起的绿色计算可被认为是一个崭新的研究领域.它从软件角度研究能耗优化问题.现有研究主要分为能源可感知的分布式系统[1−3]、以虚拟化方法为主的数据中心能效优化[4−6]和云计算环境下的能耗优化[7−9].如,文献[10]提出了 5 种策略来降低能耗:IVS,CVS,VOVO,VOVO-IVS 以及 VOVO-CVS,这 5 种策略均是根据服务器的负载情况来动态调整 CPU 电压、CPU 频率和节点个数以实现节能;文献[11]提出并论证了在 K 台服务器下,负载均分($f_i=L_T/K$)时能耗最小,并且给出了此时节点个数 K 与总负载以及参数 A,B 的关系式:

$$K=\sqrt[3]{2A/B}\times L_T.$$

无论采取何种优化方法,能耗优化首先都要定义能耗的度量模型和测量方法.文献[7]总结了实现绿色云环境下兼顾能源和可靠性的自优化框架所面临的挑战,提出应以 QoS/能耗作为衡量标准,但并没有提出实质的解决方案.文献[8]设定服务器的功率为 P 和运行时间 C,计算能耗为 $P\times C$,并没有考虑功率的动态性.文献[9]测量了云计算环境下在移动设备中使用 BitTorrent 传输文件比传统的 P2P 技术要节省客户端能耗,但仅列出实验数据,并没有给出实验方法.Orgerie[3,12,13]连续发表文章,用大量实验说明云计算的能效特性,并依托虚拟化和 Live

Migration 技术,提出了一种专注于云计算的高能效框架.文中统一使用功率的变化来衡量能效优化的效果,但其提出的能效度量是在固定负载下的计算机能耗,没有考虑运算任务的特点对能耗的影响.文献[14−16]在优化云系统能效中也都使用了功率或能量作为能耗单位.综上所述,大多研究都提出了自己的能耗模型,但近乎全部是简单的功率模型,既没有考虑约束下的能耗度量,也没有给出翔实的计算和测量方法,能源效率问题没有得到很好的重视.本文提出的能效模型就是在性能约束下的能耗度量,是单位能量内完成的运算量.能效模型的建立有助于更好地研究绿色云计算,为能效优化提供良好的评估模型.此外,现有方法通过控制CPU频率和电压的方法可以降低能耗,然而并不一定提高能效.本文虽未提出完整的能效优化方法,但却从数学上证明了能效极大值的发生条件,并通过实验数据验证其准确性.这些工作将对后续的能效优化研究起指导作用.

关于能效的计算方法,文献[10]提出计算机功率和 CPU 工作状态的关系为 $P=P_{fix}+P_f f^3$.其中,P_{fix} 为常数,表示除 CPU 以外其他设备的功率;P_f 为 CPU 功率系数,f 为 CPU 频率.该计算式已被很多文献引用和证明[11,17,18].然而,该计算式基于两个重要假设:一是假设除 CPU 以外的其他系统组件的能耗为常量(该假设已被证实近似成立);二是假设 CPU 使用率为 100%.然而,CPU 使用率受任务量和算法类型的影响,一些算法由于 I/O 和网络瓶颈的存在,CPU 不能以 100%的使用率运行.简单假设使用率为 100%来考虑功率和 CPU 频率之间的关系是不完整的.本文弥补了上述缺陷,我们考虑功率和 CPU 使用率 ω、CPU 频率 f 的关系为 $P(f,\omega)$,推导并证明了 $P(f,\omega)$ 的数学表达.

现有的很多工作对 Hadoop 的性能进行基准测试,通过测试总结 Hadoop 性能特性,评估和优化云系统的性能.如 GridMix[19]、Hive 性能基准[20]、Yahoo 排序算法[21]、HiBeach[22]和 DFSIO[23].尽管这些基准测试并非用来评估能效,但我们认为,能效与负载类型是相关的,本研究将选用这些基准测试中常用的 CPU 密集型、I/O 密集型和交互型算法来测量云系统的能效.

由此可见,虽然国内外研究成果,特别是解决问题的方法有许多可供借鉴之处,但针对能效模型和能效计算这一新问题,目前的研究成果并不适用,或不能直接应用.本文在现有研究的基础上首先提出一个适用云计算的能效模型;随后定义了计算机功率和 CPU 使用率 ω、CPU 频率 f 之间的关系;而后提出了能效的测量和计算方法,推导了能效最大值的发生条件;最后通过实验验证了能效模型和计算方法的正确性,并评估了云计算中典型的 CPU 密集型、I/O 密集型和交互型运算的能效特点.我们提出的能效模型和计算方法能够准确地评估云系统的能效,并为能效优化提供测量依据.

2 能效模型

能效的度量对象归根结底包括 3 部分:计算机、网络和空调等辅助设备.本研究仅考虑计算机能效,网络设备和空调等辅助设备的能效本研究暂不涉及.直观地看,计算机能效是能源和效率这两个概念的综合.我们首先直观地定义能源和效率两个度量,然后进行具体化,以适应云系统.

人们通常用每秒所执行的浮点运算次数(floating point operations per second,简称 FLOPS),或者单字长定点指令平均执行速度(million instructions per second,简称 MIPS)来衡量计算机的效率,而使用瓦特(Watt)来衡量电功率及电路元件或设备在单位时间内吸收或发出的电能.因此,可以采用 FLOPS/Watt 来衡量计算机的能效,它表征单位能耗下的浮点运算次数.例如,早期的 UNIVAC I 计算机(1951)1 秒可以进行 1 905 次浮点运算,却消耗了 125 千焦的电能,FLOPS/Watt 约为 0.015,能效很低.

FLOPS/Watt 计算方法可以推导如下:

$$FLOPS/\text{Watt}=(Operations/\text{second})\ \text{per Watt}=Operations/\text{Joule}.$$

FLOPS/Watt 是一定时间内计算机完成的运算次数与消耗的能量之比.其中,*Operations* 表示执行浮点运算的次数.FLOPS 的值与 CPU 的主频相关,然而,CPU 主频只能代表处理器每秒进行加法运算的次数,如奔腾 IV 3GHz 的 CPU 每秒钟可以进行 3×10^9 次加法运算.FLOPS 的值还取决于处理器进行浮点运算的流水线条数,以及平均每次单精度浮点运算消耗的时钟周期.如,奔腾 IV 处理器采用的是 Prescott 核心,Prescott 核心具有两条浮点流水线,平均每次单精计算需要消耗 6 个时钟周期,则每秒钟可以执行的单精浮点计算理论值为 $3\times10^9\times2\div$

$6=10^9$,有功功率为 80Watt,能效约为 12 500 000FLOPS/Watt.

由此可见,使用浮点运算(FLOP)作为负载单位会导致能效数值过大、计算过于复杂且依赖多种因素.因此,我们设计一种新的负载单位.事实上,负载单位可以根据算法设计.如排序算法,我们可以定义排序 1 000 条数据的任务量作为单位 1,但这种方法不具有通用性,且与算法的复杂度相关.为简化模型,我们采用主频为 1GHz 的 CPU 在 1s 中的运算量作为 1 负载单位,记为 1U.

定义 1(负载单位). 负载单位(U)用来衡量计算任务量的大小,定义处理器在 1GHz 频率下 1s 完成的运算量为 1 负载单位,记为 1U.

由于各种 CPU 的运算能力差异,1GHz CPU 每秒的运算量对于不同种类 CPU 是不同的,不能笼统地定义为 1U.在实际测量中,采用 CPU 频率与使用率以及λ系数的乘积来衡量特定 CPU 完成的负载任务大小,即

$$\text{CPU 负载}=\text{CPU 频率}\times\text{CPU 使用率}\times\lambda_{\text{CPU}}.$$

从能效优化角度来说,只要使用统一定义的负载单位,就能确定能效规律,衡量优化效果.定义能效模型的主要目的是为了指导和评估能效的优化,负载单位 U 的定义不影响这一目的的实现.不失一般性,本研究设λ系数为 1.在实际测量比较时,可以根据云系统中各种 CPU 的运算能力差异而调整λ系数的取值.

用 T 时间内系统处理的任务 $L(T)$(单位 U)和能耗 $E(T)$(单位焦耳)分别替换 FLOPS 和 Watt,我们定义 T 时刻内能效$\eta(T)$为

$$\eta(T)=\frac{L(T)}{E(T)},\ E(T)\neq 0 \tag{1}$$

能效单位记为η,η=U/Joule,1m$\eta=10^{-3}\eta$.

对于云计算系统,设存在 N 个运算节点,记为 $c_i(1\leqslant i\leqslant N)$,对于 c_i 运算节点的 CPU 在 t 时刻频率为 $f_i(t)$,使用率为$\omega_i(t)$,功率(Watt)为 $p_i(t)$,则

$$L_i(T)=\sum_{i=1}^{N}\int_0^T f_i(t)\omega_i(t)\mathrm{d}t \tag{2}$$

$$E_i(T)=\sum_{i=1}^{N}\int_0^T p_i(t)\mathrm{d}t \tag{3}$$

$$\eta_i(t)=\frac{L_i(t)}{E_i(t)}=\frac{f_i(t)\omega_i(t)}{p_i(t)},\ \eta(T)=\frac{L(T)}{E(T)}=\frac{\sum_{i=1}^{N}\int_0^T f_i(t)\omega_i(t)\mathrm{d}t}{\sum_{i=1}^{N}\int_0^T p_i(t)\mathrm{d}t} \tag{4}$$

云系统中的每个计算机都有额定的功率和 CPU 频率,若 CPU 以满负荷运行,则认为能效的理论值为 CPU 频率和额定功率的比值.但在运行环境中,CPU 的使用率取决于运行的算法特点,CPU 频率会随着任务量的大小动态调整(CPU frequency scaling),而功率也会随之发生变化.因此,能效的计算需要依据实际测量值.

3 能效测量和计算方法

在前文定义的能效模型的基础上,本节介绍能效的测量方法.因为云系统能效的测量受仪器的限制而较为复杂,因此我们提出了一种能效计算方法,通过容易测量的数据,如 CPU 使用率和频率来计算能效.最后,我们根据计算公式推算了能效最大值及其发生条件.

3.1 测量方法

能效的测量分为两个部分:一是测量 $L(T)$,二是测量 $E(T)$.首先我们考虑测量 $L(T)$,由公式(2)可知,对 $f_c(t)$和$\omega_c(t)$在 T 时间上积分得到 $L(T)$.其中,$f(t)$取决于任务量大小和操作系统的 Frequency Scaling 算法.如 Linux 操作系统下 CPU 可以工作在 5 种模式(Performance,Powersave,Ondemand,Conservative,Userspace)下.而$\omega_c(t)$则取决于算法本身,如执行 CPU 密集型算法时 CPU 的使用率会接近 100%;而执行 I/O 密集型算法时,由于 CPU 要等待 I/O,因此使用率通常不高.由此可见,我们无法确切地找到 $f_c(t)$和$\omega_c(t)$的函数表达,但 $L(T)$可以测量得到.

主流操作系统都提供了测量 CPU 频率和使用率的接口,我们在云系统中每个节点上部署 CPU 监控代理,云计算中分布式的编程环境可以很容易地同步这些监控代理并汇总数据.设代理每隔Δt 时间对节点进行一次 CPU 频率 f 和使用率ω的采样,从 0 时刻开始至 T 时刻共采样 M 次.由定积分的定义可知:

$$L(T)=\sum_{i=1}^{N}\int_0^T f_i(t)\omega_i(t)\mathrm{d}t\approx\sum_{i=1}^{N}\sum_{j=1}^{M}f_i(\Delta t\cdot j)\omega_i(\Delta t\cdot j)\Delta t \tag{5}$$

当Δt 足够小时,公式(5)的值即为 $L(T)$的值.其中,$f_i(\Delta t\cdot j)$和$\omega_i(\Delta t\cdot j)$均可通过监控代理测得.代理以$\Delta t$ 时间间隔不停地采集 CPU 频率及使用率.T 时间后汇总运算节点的测量值,按公式(5)计算得到整个云系统的负载.

测量 $E(T)$的方法有很多种.所有计算机都有额定功率,但实际功率却是动态变化的,因此很难找到 $p_i(t)$的数学表达,并按公式(3)计算能耗 $E(T)$.我们同样可以每隔Δt 时间对节点进行一次实时功率的采集,类似于公式(5).

$$E(T)=\sum_{i=1}^{N}\int_0^T p_i(t)\mathrm{d}t\approx\sum_{i=1}^{N}\sum_{j=1}^{M}p_i(\Delta t\cdot j)\Delta t \tag{6}$$

当Δt 足够小时,公式(6)的值即为 $E(T)$的值.但大部分计算机设备或操作系统都未提供实时功率的测量接口,测量 $p(\Delta t\cdot j)$存在一定的难度.若为云内每个计算机节点安装功率传感器,则需要大量传感器,且它们之间的通信(时钟同步、数据汇总)也较为复杂.最简单的方法是使用电量计直接测量 T 时间经过电缆线的电量,或者测量整个云计算系统的实时功率 $p_{cloud}(\Delta t\cdot j)$,按 $\sum_{j=1}^{M}p_{cloud}(\Delta t\cdot j)\Delta t$ 得到整体 $E(T)$值.

然而,电量计的方法也存在一定的不足:

① 电量计都有一定的功率测量范围,普通的电量计测量的最大功率在 2 500 瓦左右,对于一个包含大量用电设备的云系统,需要大量电量计,并且电量计的控制、同步和数据汇总都存在难题.

② 电量计无法区分每台计算机的能耗.建立能效模型的目的是能效优化和能效相关的 SLA 制定,因此需要确切地知道每运算节点,或每用户请求的耗能,而电量计测量能耗的方法粒度过大.

由此可见,云计算中能耗的测量需要额外的设备,测量方法比CPU频率和使用率的测量要复杂许多.若能根据实时 CPU 频率和使用率来推算实时功率,进而按照公式(6)计算能耗,则将极大地简化能效的测量过程,下节将介绍这种计算方法.

3.2 计算方法

根据第 3.1 节的论述,本节着重考虑能耗 $E(T)$的计算方法.我们假设云是由同构的运算节点构成的及 $E(T)$的计算方法适用于同构的云.可以把异构的云看作是由多个同构的子云组成的,在每一个同构子云上计算 $E(T)$,最后汇总得到整体 $E(T)$值.

根据文献[10]的定义,当 CPU 以频率 f 满负荷运行时,功率和 f 之间的关系为 $P=P_{fix}+P_f f^3$.其中,P_{fix} 为常数,表示除 CPU 以外其他设备的功率,这些设备功率较稳定;P_f 为 CPU 功率系数.然而,文献[10]假设使用率为 100%,但计算机 CPU 使用率随任务量和算法类型的影响而变化,因此,仅简单考虑功率和频率之间的关系是不完整的.

我们假设功率和 CPU 使用率ω、CPU 频率f的关系为 $P(f,\omega)$,推导 $P(f,\omega)$的数学表达.已知当ω固定时,$p_c=A_1+B_1f^3$;另外,假设当 f 固定时,$p_c=A_2+B_2\omega$,其中,A_2 表示 CPU 空闲时系统的功率,B_2 为使用率系数(实验 1 将证明该假设的正确性).为简化推导,我们把 f^3 作为一个自变量,考虑 $P(f^3,\omega)$的数学表达:

$$\left.\begin{aligned}\frac{\partial P(f^3,\omega)}{\partial\omega}&=A_1+B_1f^3\\ \frac{\partial P(f^3,\omega)}{\partial f^3}&=A_2+B_2\omega\end{aligned}\right\}\Rightarrow P(f^3,\omega)=Af^3+B\omega f^3+C\omega+D \tag{7}$$

公式(7)中,A,B,C,D 为系数,其值可以在单机环境下通过测量得到(参见实验 1).由于我们考虑云系统中节点间是同构的,因此 A,B,C,D 的值对每个节点来说都是相同的.结合公式(3)和公式(7),可以得出多个节点 T 时段的能耗 $E(T)$:

$$E(T)=\sum_{i=1}^{N}\int_0^T p_i(t)\mathrm{d}t=\sum_{i=1}^{N}\int_0^T[Af_i(t)^3+B\omega_i(t)f_i(t)^3+C\omega_i(t)+D]\mathrm{d}t \tag{8}$$

$$E(T)=\sum_{i=1}^{N}\sum_{j=1}^{M}[Af_i(\Delta t\cdot j)^3+B\omega_i(\Delta t\cdot j)f_i(\Delta t\cdot j)^3+C\omega_i(\Delta t\cdot j)+D]\Delta t \tag{9}$$

$$\eta(T)=\frac{L(T)}{E(T)}=\frac{\sum_{i=1}^{N}\sum_{j=1}^{M}f_i(\Delta t\cdot j)\omega_i(\Delta t\cdot j)\Delta t}{\sum_{i=1}^{N}\sum_{j=1}^{M}[Af_i(\Delta t\cdot j)^3+B\omega_i(\Delta t\cdot j)f_i(\Delta t\cdot j)^3+C\omega_i(\Delta t\cdot j)+D]\Delta t} \tag{10}$$

由第 3.1 节介绍的测量方法可以测得 $f(\Delta t\cdot j)$和$\omega(\Delta t\cdot j)$的值;根据公式(10),可以计算得出 T 时间内云系统的能效.每节点的CPU的频率和使用率可以很方便地测量,计算方法不仅可以计算云系统的总能效,而且可以独立计算每一个节点的能效,因此,计算方法具有粒度细致、实现容易等优点.

异构性是云计算的一个显著特点,本方法同样可以运用到异构的环境中,其原因和优势有以下几点:

① 异构云环境可以划分为若干个同构子环境,这些同构子环境的数量绝非海量,因此对每一种同构子环境确定参数值是可行的.

② 对于每个同构子环境,只要选取一个节点确定参数,相对于使用仪器测量每个节点的功率,参数的测量方法是较为简单的.

③ 参数确定后就会不发生变化,使用仪器监测电量或是功率则是一种长期往复的过程.用单次测量参数的方法使较为复杂的监测计算机功率转换成较简单的监测计算机频率和使用率,是本节提出的计算方法的优势.

④ 本节提出的模型和方法有助于能效优化的研究,从这个角度,某些参数的确定与否并不影响对能效优化方法的评价.

⑤ 在异构的生产环境中,我们可以运用在同构的实验环境中得出的能效规律和优化方法,部分优化方法与具体设备无关.但这并不代表同构系统中的能效规律在异构系统中一定发生,一定完全一致,后者存在新的规律,需要进一步研究.

3.3 能效极值分析

由公式(10)可以看出,能效与$f(t)$和$\omega(t)$有关,本节从理论角度论证云计算的能效优化方法.直观上,降低CPU频率或使CPU空闲可以降低能耗,但未必提高能效.若云系统中每个节点的能效都达到最大值,则认为整个系统的能效最优.因此,我们考虑单一节点的能效.在 T 时间段内,若对于任意时刻 t,$\eta(t)$都为最大值,则$\eta(T)$最大,因此,

$$\eta(t)=\frac{L(t)}{E(t)}=\frac{f(t)\omega(t)}{Af(t)^3+B\omega(t)f(t)^3+C\omega(t)+D} \tag{11}$$

我们首先研究$\eta(t)$与$\omega(t)$之间的关系,把公式(11)的分子分母都除以$\omega(t)$:

$$\eta(t)=\frac{f(t)}{\frac{Af(t)^3}{\omega(t)}+Bf(t)^3+C+\frac{D}{\omega(t)}} \tag{12}$$

由公式(12)可以看出,$f(t)$恒大于0,则$\eta(t)$随$\omega(t)$单调递增,即CPU使用率越高,能效越高.由于$\omega(t)$的取值范围在[0,1]区间,因此,当$\omega(t)$=1 时能效达到最大.

随后研究$\eta(t)$与$f(t)$之间的关系,对公式(11)中的$f(t)$求导:

$$\frac{\partial\eta(t)}{\partial f(t)}=0\Rightarrow f(t)=\sqrt[3]{\frac{C\omega(t)+D}{2[A+B\omega(t)]}} \tag{13}$$

由公式(13)知,当$\omega(t)$固定时,公式(11)存在一个极值,我们可以证明该值为极大值.

由上述推导可知,当$\omega(t)$=1,$f(t)$=$\sqrt[3]{\frac{C+D}{2(A+B)}}$ 时,能效达到极大值.计算机能效极值发生在 CPU 以某一特定频

率 100%运行的时刻.该结果仅仅是通过数学推理得到的,其准确性将在实验部分得到证实.

4 实验分析和结果展示

根据第 3.1 节的论述,测量云系统的能效需要测量每个节点的 CPU 实时频率和实时使用率,以及整体能耗.而计算能效则只需要测量每个同构子云中节点的 CPU 实时频率和实时使用率.为了验证上述理论和数学表达的正确性,我们选用 10 台 PC 机构建了基于 Hadoop HDFS 的云计算环境,实验环境见表 1.

Table 1 Description of experiment environment

表 1 实验环境描述

项	描述
节点	1 个管理节点,9 个运算节点,同构的清华同方超翔 Z900 计算机,CPU 为 Inter Core i5-2300 2.80GHz,8GB 内存,1TB 硬盘,主板集成声卡、显卡和网卡,所有能耗数据均不包含显示器、鼠标和键盘的能耗
操作系统	CentOS 5.6, Linux 2.6.18 内核
CPU 工作模式	单机环境:频率固定在用户设定的模式下(Userspace); 云环境:平滑地根据任务量调整 CPU 频率(Conservative)
编程环境	Java 6
云计算环境	Hadoop 0.20.2
电量计	北电电力监测仪(专业版),依据 GB/T17215-2003,功率误差值±0.01~0.1W,功率测量上限为 2 200W,数据采样频率在 1.5s~3s 之间,能耗测量单位为 kWh(保留 2 位小数)
单机环境用例	圆周率蒙特卡罗算法(概率法)[24],通过调整线程个数和强制线程休眠控制 CPU 使用率
MapReduce 用例	WordCount[25]:CPU 密集型运算,网络和磁盘 I/O 负载较轻,数据量每节点为 100MB,250MB,500MB,750MB,1 000MB; Sort[21]:I/O 密集型运算,数据量每节点为 100MB,250MB,500MB,750MB,1 000MB; MRBench[26]:交互式运算,较小的作业(jobs)高并发的执行,实验更正 Hadoop MRBench 并行执行的 Bug[27],扩展顺序启动多作业(jobs)为并发启动多作业,并发作业数为 1,10,20,50,60,75,85,100,110,135,145,150,200,300
ω和 f 采集频率Δt	1 秒采集 10 次数据(Δt=0.1s)
测量单位	效率:按照定义 1 中定义的单位 U; 能耗:焦耳(Joule),我们用功率表示实时能耗; 功率:测量一律采用有功功率,单位瓦特(Watt); 能效:能效单位记为η,η=U/Joule,实验采用 mη作为单位,1mη=$10^{-3}\eta$; CPU 频率:GHz; CPU 使用率:比例ω,其中,$0\leqslant\omega\leqslant 1$,对于 Linux 系统,$\omega$为 CPU 用户使用率和系统使用率之和,即(1−CPU 空闲率)

4.1 单机实验

我们在单机环境下对 CPU 使用率$\omega(t)$和计算机功率 $p(t)$进行测量.首先我们验证公式(7)中提出的当 $f(t)$固定时,$\omega(t)$和 $p(t)$的线性关系;随后验证当$\omega(t)$固定时,$p(t)$和 $f(t)$的三次方之间的线性关系;最后,推算公式(8)中的 A,B,C,D 这 4 个系数的值,并比较计算能效和测量能效之间的关系.

实验 1. 频率 f 固定时,计算机功率 $p(t)$和能效$\eta(t)$与使用率$\omega(t)$之间的关系.

在单机环境中,我们采用强制线程休眠一定时间的办法,使 CPU 使用率按照正弦曲线的方式变动(如图 1 所示的ω曲线,算法略),得到大量连续的ω值,并测量其对应的计算机功率,测量结果如图 1 和图 2 所示.

由图 1 可以看出,使用率曲线和功率曲线的峰值、谷值都同时发生,变化趋势也一致.图 2 是图 1 中 0~30s 的图像放大图.由图 1 和图 2 可以看出,无论 CPU 工作在何种频率下,计算功率曲线和使用率曲线的趋势都是吻合的,成线性关系.细致观察:图 2 中个别功率极值和使用率极值的发生时间存在误差,这是因为使用仪器测量功率存在一定程度的滞后;图 2 中很多时刻功率值是相同的,曲线程阶梯上升形态,这是因为仪器测量功率的频率要小于监控代理采集 CPU 使用率的频率.

综上所述,我们可以验证公式(7)中提出的当 f 固定时,式 $p_c=A_2+B_2\omega$是正确的,参数 A_2 和 B_2 与计算机相关.我们在实验 2 中讨论参数的测定.

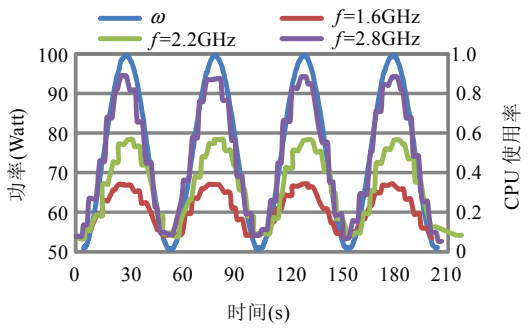


Fig.1 Sine curve of CPU usage and its corresponding power curve

图 1 使用率正弦曲线和对应的功率曲线

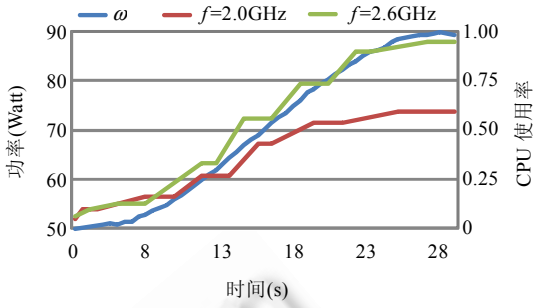


Fig.2 Smoothly increased CPU usage and its corresponding power curve

图 2 CPU 使用率平稳增加时对应的功率曲线

进一步地,我们使用圆周率蒙特卡罗算法[24]作为运算任务,通过调整算法并行的线程数目来控制 CPU 的使用率,我们把ω设定在 0,0.25,0.5,0.75 和 1 这 5 个值上,通过 Linux 的 CPU 运行模式(Userspace),可以使设定 CPU 频率稳定工作在 1.6GHz~2.8GHz 之间,使用电量计对计算机功率测量.我们计算了该实验条件下的实时能效.根据第 3.3 节的数学分析,CPU 使用率越大,能效越高.图 3 展示了根据公式(4)计算的实时能效,无论 CPU 工作在何种频率下,最大能效均发生在ω=1 时刻,与理论推导吻合.

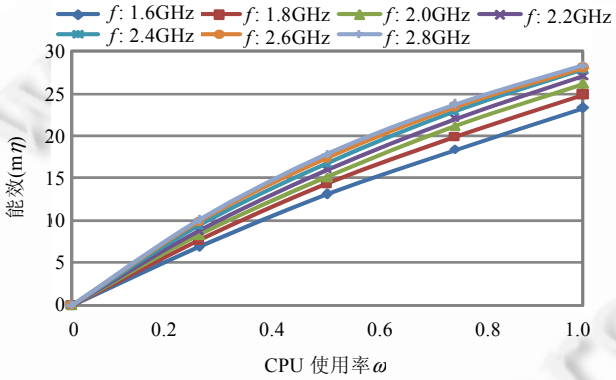


Fig.3 Relation between CPU usage and real-time energy efficiency under different CPU frequency

图 3 不同频率下 CPU 使用率和实时能效之间的关系

实验 2. 使用率ω固定时,计算机功率 $p(t)$和能效$\eta(t)$与频率$f(t)$之间的关系.

我们把ω设定在 0,0.25,0.5,0.75 和 1 这 5 个值上,通过 Linux 的 CPU 运行模式(Userspace)可使 CPU 稳定工作在设定频率 1.6,1.8,2.0,2.2,2.4,2.6 和 2.8GHz.使用电量计对计算机实时功率进行测量,得出如图 4 所示的曲线.

从图 4 中功率曲线可以得出,当 CPU 使用率不为 0 时,CPU 频率越大,功率越大,基本符合 $p_c=A_1+B_1f^3$ 的规律,然而与频率三次方的关系还需要进一步验证(见实验 3).当 CPU 使用率为 0 时,功率近似为常量.进一步地,我们计算了图 4 实验条件下的实时能效$\eta(t)$,由于当ω=0 时计算机没有进行运算,因此能效为 0.图 5 仅显示了ω=0.25, 0.5,0.75,1 时的能效与 CPU 频率之间的关系.

根据第 3.3 节的数学分析,当 $f(t)^{\max}=\sqrt[3]{\dfrac{C\omega(t)+D}{2[A+B\omega(t)]}}$ 时,能效达到最大值.我们根据实验 1 和实验 2 得到的数据计算得出,对于本实验使用的计算机,A=0.15,B=1.4,C=15.2,D=50.8,于是,

$$f^{\max}=\sqrt[3]{\frac{C\omega(t)+D}{2[A+B\omega]}}\approx\begin{cases}3.7937\text{GHz}, & \omega=0.25\\ 3.2508\text{GHz}, & \omega=0.5\\ 2.9593\text{GHz}, & \omega=0.75\\ 2.7716\text{GHz}, & \omega=1\end{cases} \tag{14}$$

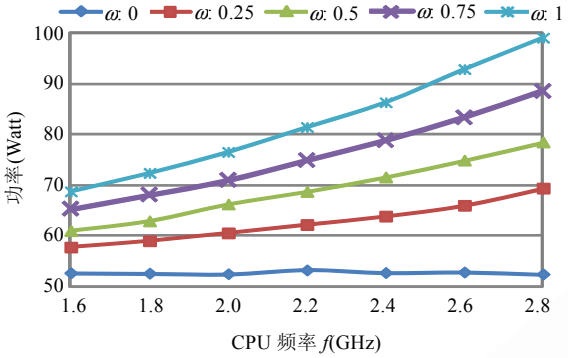


Fig.4 Relation between frequency and computer power under different usage

图 4 不同使用率下频率和计算机功率之间的关系

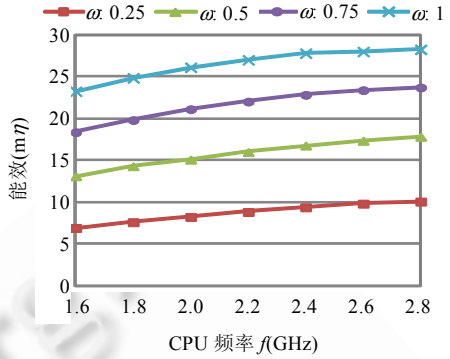


Fig.5 Relation between frequency and real-time energy efficiency under different usage

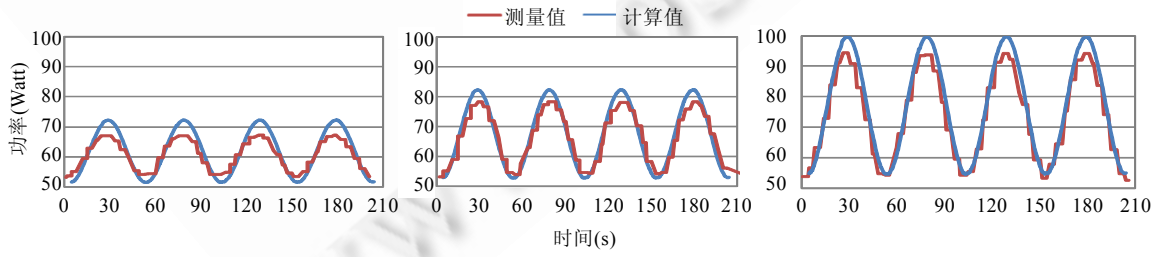
图 5 不同使用率下频率和实时能效之间的关系

测试使用的计算机 CPU 最大频率为 2.8GHz,理论能效最大值发生时,CPU 理论频率均超过或接近 CPU 的最大频率,因此,图 5 中的曲线呈上升趋势,最大能效均发生在 f=2.8GHz 处,与理论推导吻合.进一步分析,根据公式(7),功率 $p=P(f,\omega)=Af^3+B\omega f^3+C\omega+D$.令 ω=1,$p=P(f,\omega)=(A+B)f^3+(C+D)$,其中,$C+D$ 为除 CPU 以外的设备的能耗常量.对于特定 CPU,A 和 B 的值已固定,$f^{\max}$ 的取值取决于 $C+D$ 的值.可以这样理解,计算机外设耗能越多,CPU 越需要更快的工作去抵消这些额外能耗,以提高能效.本文所有实验中均未计入显示器的能耗,若计入,重新计算 A=0.15,B=1.4,C=16,D=70,当 ω=1 时 $f^{\max}$=3.0272,远超出 CPU 的最大频率.

实验 3. 计算能效和测量能效之间的比较.

本实验比较测量能效和计算能效之间的差异.本文第 3.2 节提出了一种根据 CPU 频率和使用率推算计算机功率,进而推算能效的方法.若单机环境下计算方法是准确的,则可以推广到云计算环境中.本实验首先比较了能耗的计算.能耗计算本质是功率的推算,我们使用电量计测量计算机的实时功率,并使用程序实时测量 CPU 频率和使用率,根据公式(7),$p=P(f,\omega)=Af^3+B\omega f^3+C\omega+D$($A$=0.15,$B$=1.4,$C$=15.2,$D$=50.8),计算实时功率并与测量的功率进行比较.

依据实验 1 中模拟的 CPU 使用率按正弦曲线形式变化的测试用例,我们分别比较了 f=1.6,2.2,2.8GHz 时计算功率和测量功率,比较结果如图 6 所示.



(a) f=1.6GHz (b) f=2.2GHz (c) f=2.8GHz

Fig.6 Comparison of calculated power and measured power

图 6 计算功率和测量功率的比较

由图 6 可以看出,总体上功率计算值和测量值是吻合的.因为本实验 f 值固定,采集了大量连续 ω 值(参见图 2 的 ω 曲线),因此功率计算值曲线平滑;而功率测量值受仪器采集频率和数据延迟的影响,曲线不平滑且呈阶梯状.图 6 中,功率计算值的峰值均略高于测量值,谷值则略低于测量值,即功率测量值较为平均化.这是因为仪器测量 Δt 时间的功率平均值作为瞬时值,而该 Δt 要远大于 ω 的测量时间间隔,因此在 ω 达到峰值谷值的一瞬间,对应的功率测量并没有发生.监控代理每 0.1s 测量一次 ω 值,而仪器最小间隔为 1.5s 测量一次功率,ω 值变化对功率的影响在功率采样过程中被平均化,因此导致图 6 中功率计算曲线和测量曲线的差异.此外,造成这一现象的原因是我们假设除 CPU 之外的其他组件功率为常量,而实际情况下,这些组件的功率仍存在一定的浮动.

我们进一步考察能效计算值和测量值的差异.依据实验 2 的结果,图 7 展示了当 ω=0.25,0.5,0.75,1,且 f=1.6,1.8,2.0,2.2,2.4,2.6,2.8GHz 时能效计算值和测量值的差异(Y 轴为累积能效值).直观地,在 ω 和 f 的各种组合下,能效计算值和测量值相差无几,统计知平均误差率为 2%(主要原因是忽略了除 CPU 之外的其他组件功率的浮动).单机实验证明了本文提出的能效模型和能效计算方法的准确性.我们采用相同的设备构建了一个小型的云计算环境,继续验证本文提出的能效模型和测量方法在云环境中的准确性.

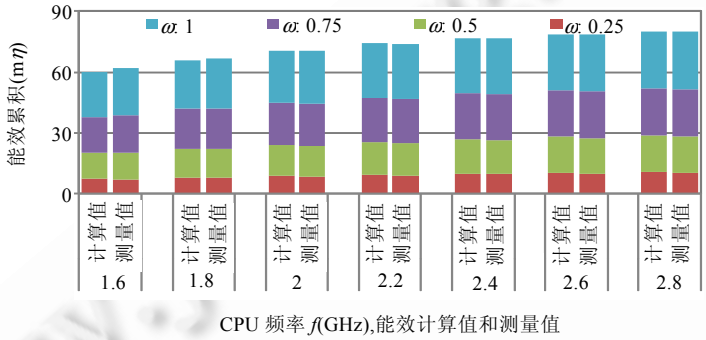


Fig.7 Comparison of calculated energy efficiency and measured energy efficiency

图 7 计算能效和测量能效的比较

4.2 云环境实验

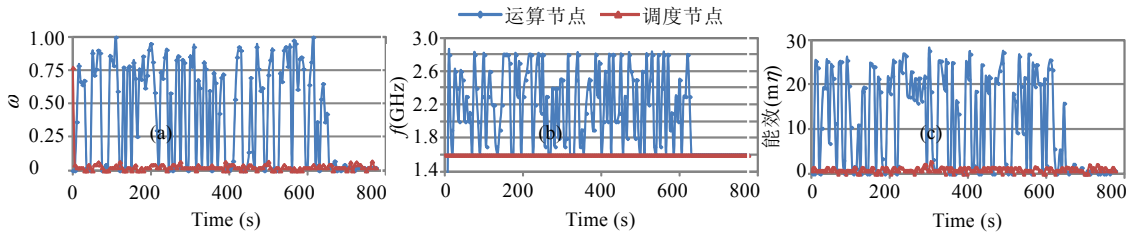
根据表 1 的实验环境,我们在云系统中对不同任务量的 WordCount(CPU 密集型计算)[25]、Sort(I/O 密集型计算)[21]和 MRBench(交互式任务,小作业高并发的执行)[26]进行了测试,研究其能效规律.

实验 4. 3 种类型运算的 CPU 频率、使用率和能效的基本特征.

为了更好地研究能效,我们分析 3 种类型的运算在特定任务量的执行时间内,调度节点(仅 1 个)和运算节点(选取 1 个)的实时 CPU 频率曲线、使用率曲线和能效曲线.由于选取的 500MB 每节点的 WordCount 和 Sort 运算执行时间在 800s 左右,数据采集过于密集,无法在图中显示,我们对测量的数据进行采样,每 5s(50 条数据)随机采样 1 次.而 50 作业并发条件下的 MRBench 运算则仅执行 50s 左右,无须对数据进行采样.WordCount 运算的 $f(t),\omega(t),\eta(t)$ 曲线如图 8 所示.

由图 8 可以看出,在 WordCount 运算中,调度节点比运算节点空闲很多.调度节点的 CPU 使用率很低,频率保持在最低频率 1.6GHz,仅运算开始时频率有短暂的上升,调度节点的能效也很低.WordCount 是一种 CPU 密集型运算,观察运算节点的能效规律可以发现,在 600s 之前,$f(t),\omega(t),\eta(t)$ 都很高,随之急剧下降.这符合 MapReduce 算法的特点,Map 阶段着重于运算,而 Reduce 阶段有大量的 I/O 操作和网络操作,Map 运算的中间结果在运算节点中重新重新分布,于是 CPU 空闲,能效下降.

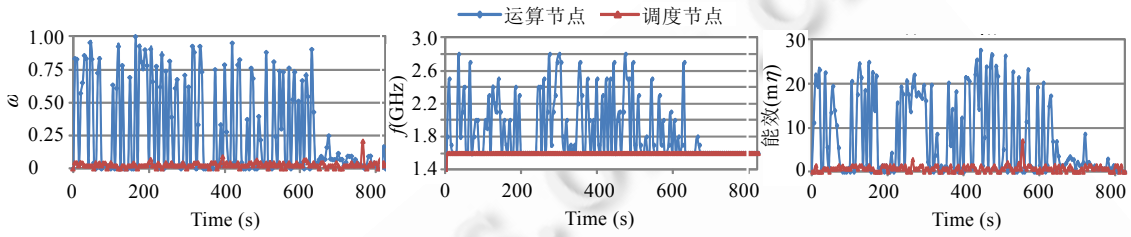
对于每节点 500MB 数据条件下的 Sort 运算,图 9 的基本规律与图 8 类似,Map 阶段的 $f(t),\omega(t),\eta(t)$ 高于 Reduce 阶段,调度节点仅在运算启动时调度任务,结束时返回结果,因此参与的计算比运算节点少很多.图 10 的并发 MRBench 运算则有所不同,虽然调度节点总体上能效值低于运算节点,但其 $f(t),\omega(t)$ 较前两者都接近运算节点,在并发作业执行初期还高于运算节点.



(a) CPU 使用率$\omega(t)$ (b) CPU 频率$f(t)$ (c) 节点能效$\eta(t)$

Fig.8 $f(t)$, $\omega(t)$, $\eta(t)$ curves of WordCount under 500MB data per node

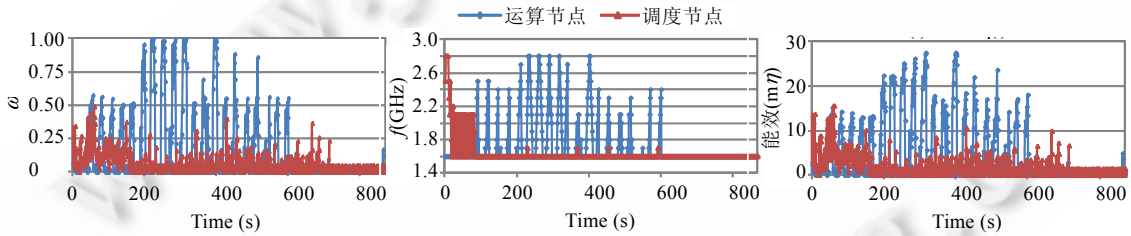
图 8 每节点 500MB 数据的 WordCount 运算$f(t),\omega(t),\eta(t)$曲线



(a) CPU 使用率$\omega(t)$ (b) CPU 频率$f(t)$ (c) 节点能效$\eta(t)$

Fig.9 $f(t)$, $\omega(t)$, $\eta(t)$ curves of Sort under 500MB data per node

图 9 每节点 500MB 数据的 Sort 运算$f(t),\omega(t),\eta(t)$曲线



(a) CPU 使用率$\omega(t)$ (b) CPU 频率$f(t)$ (c) 节点能效$\eta(t)$

Fig.10 $f(t)$, $\omega(t)$, $\eta(t)$ curve of MRBench under 50 concurrent jobs

图 10 50 个并发作业下,MRBench 运算的$f(t),\omega(t),\eta(t)$曲线

实验 5. 3 种类型运算的 CPU 频率和使用率比较.

本文提出的能效计算方法取决于 CPU 频率和使用率,因此我们对 3 种类型运算的f和ω值进一步做了比较.本实验对每节点 500MB 数据的 WordCount,Sort 运算和 50 个并发作业下 MRBench 运算的实时f和ω值进行统计,计算每种数值出现的频度比率.我们分别对比了 3 种运算的调度节点(仅 1 个)和运算节点(选取 1 个)的 CPU 频率和使用率的统计值,如图 11 所示.

由图 11 可以得出以下结论:

① 对于运算节点(如图 11(a)、图 11(c)所示),CPU 工作在高使用率的时间 WordCount 最多,Sort 次之,MRBench 最少,因为 CPU 密集型运算对 CPU 的占有率高,而 I/O 密集型运算 CPU 过多的等待 I/O 操作;同理,CPU 工作在高频率的时间同样是 WordCount 最多,Sort 次之,MRBench 最少.

② 对于调度节点(如图 11(b)、图 11(d)所示),3 种运算的调度节点都比较空闲,唯独在 MRBench 中,调度节点的 CPU 使用率和频率有所上升(如图 10 所示).

③ 调度节点可以管理大量运算节点,实验环境中,调度节点和运算节点 1 比 9 的关系导致其空闲,影响了云系统的整体能效.

④ 3 种运算类型的运算节点 CPU 使用率和频率(如图 11(a)、图 11(c)所示)均是正向倾斜(positively skew)及 CPU 的使用率和频率较低的时间居多.结合前文推导的能效最优条件,我们认为,若使图 11(a)、图 11(c)负向倾斜(negatively skew)及 CPU 的使用率和频率较高的时间居多,就可以优化能效.

⑤ 造成 CPU 频率和使用率不高的原因:一是任务量不饱满;二是 CPU 在等待其他资源,包括 I/O 操作、网络通信和调度.结合运算种类和用户的访问量等不同因素,对云计算系统和 MapReduce 任务从 I/O 操作、网络通信和任务调度等角度优化,可以提高云系统能效.

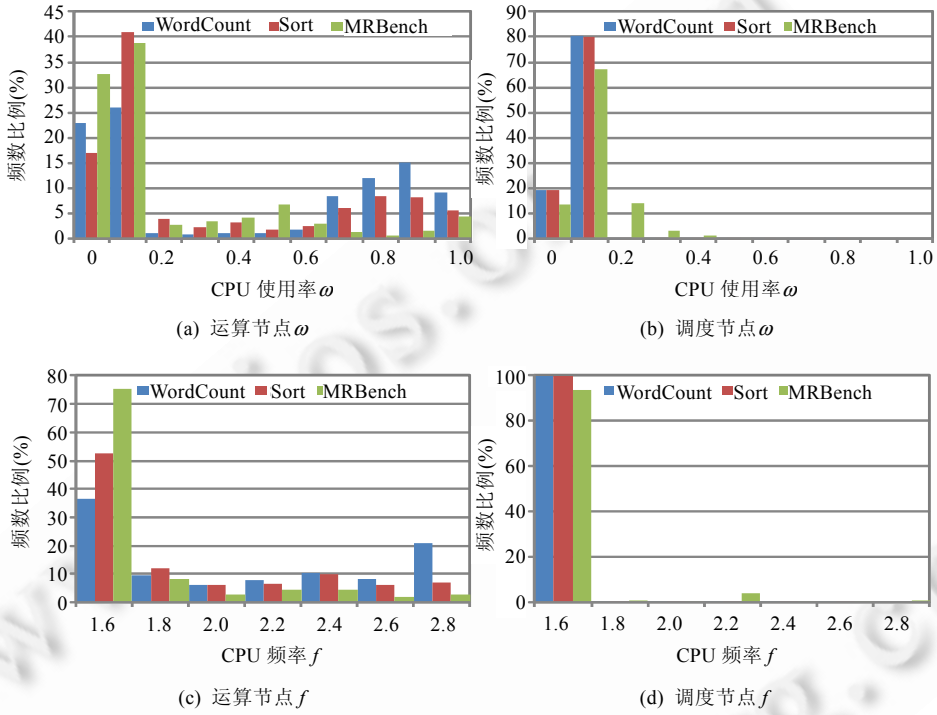


Fig.11 Statistic comparison on $f(t)$, $\omega(t)$ of WordCount, Sort and MRBench under condition of experiment 4

图 11 实验 4 条件下,WordCount,Sort 和 MRBench 的 $f(t)$,$\omega(t)$统计值对比

实验 6. 3 种类型运算的整体能效计算值和测量值的比较.

为了进一步验证本文提出的能效计算方法的正确性以及对比 3 种运算类型下云系统的总体能效,我们对云系统能效的计算值和测量值进行比较.计算值是按照公式(10),通过测得的实时 CPU 频率 $f(\Delta t\cdot j)$和 CPU 使用率 $\omega(\Delta t\cdot j)$计算得出的(Δt=0.1s);测量值则是使用电量计,直接测量 T 时间经过电缆线的电量作为云系统的消耗的总电量,带入公式(10)的分母部分而计算得出的.实验结果如图 12 所示.

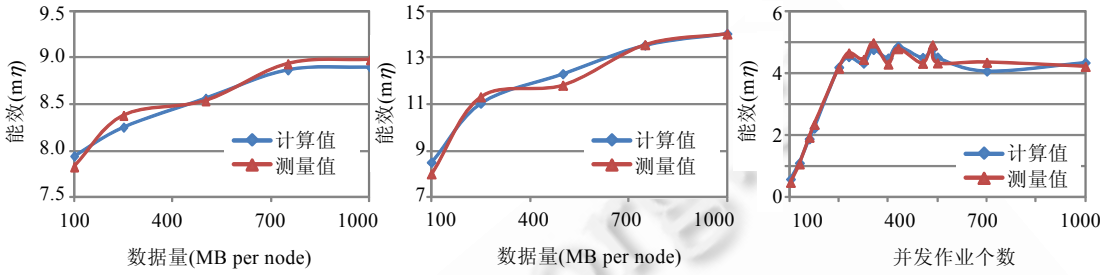
由图 12 可以得出 3 个结论:

① 总体上,测量值和计算值误差很小,经统计误差率在 1%~5%之间,由于电量计的电量单位是千瓦时(保留 2 位小数),在测量时运算和仪器的同步需要手工控制,因此小范围的误差是可以接受的.

② CPU 密集型运算(WordCount)的能效要大于 I/O 密集型运算(Sort),两者都大于交互型(MRBench)运算,这符合实验 4、实验 5 中 CPU 的使用率和频率曲线.

③ 对于密集型运算 WordCount 和 Sort,数据量的增加表示任务量的增大,能效曲线随任务量增大而缓慢增

加,总体变化趋势不大,并逐渐趋于平稳.结合实验 4 的分析知,通过调整任务的大小优化能效,效果并不明显,必须改进任务在运算节点中的执行算法,进而提高 CPU 的使用率和频率,提高能效.对于交互型运算,能效随并发请求个数增加快速增大,达到最大值后趋于平稳.对交互型运算的能效优化,单纯通过增加并发作业数目效果不明显,而是要通过调整作业分配和执行算法等其他方法提高能效,同时保证一定的并发数目,最终实现能效优化.图 12(c)曲线达到峰值后并不平稳,而是在一定范围($\pm 0.5m\eta$)波动.初步分析,一是测量点较密集,数据单位较小,能效值波动属正常现象;二是测量仪器的误差所致;三是在并发作业超过 100 以后,少数作业会异常终止,这也在一定程度上影响了能效值.



(a) WordCount (b) Sort (c) MRBench

Fig.12 Comparison between the calculated and measured value of overall energy-efficiency of WordCount, Sort and MRBench under different workloads

图 12 不同任务量下,WordCount,Sort 和 MRBench 的整体能效的计算值和测量值比较

5 结论和进一步工作

本文描述了一种云计算环境下的能效模型和度量方法.文中详细阐述了能效模型的数学表达、测量方法、计算方法、极大值发生条件以及云计算环境中 CPU 密集型运算、I/O 密集型运算和交互型运算的能效特点,并提出了进一步优化能效的思路.本文得出以下结论:

- 云计算系统的能耗不应该孤立地考察,度量模型应该是有约束的,而最有效的约束就是运算速率,能效是衡量单位能量完成的运算.基于此,低功率运行的计算机耗能可能较小,但能效未必较高,因为低功率会导致运算速率的下降.本文给出了能效的数学表达,并且推导其存在极大值的发生条件,又通过实验证明了上述推导的正确性.
- 测量云计算系统中,每个节点的实时功率难以实施,而本文提出的计算机功率和 CPU 工作状态关系的计算公式,弥补了现有计算公式没有考虑 CPU 使用率这一明显不足.基于此,可以通过测量 CPU 频率和使用率来计算功率,进而计算能效,这种方法简单易行.文中大量的实验数据证明了计算方法的准 确性.
- 实验证明,典型任务(CPU 密集型、I/O 密集型或交互型)下的云系统能效很低,有很大的提升空间.原因主要是 CPU 空闲,通过加大任务量和请求数目可以一定程度上的提高能效.最有效的途径是研究任务的调度和执行算法,优化 I/O 操作、网络通信和节点调度,缩短 CPU 对资源的等待时间.

本研究还处于初步阶段.首先,定义能效模型的目的是为了更好地优化能效,因此我们将依据本文得出的结论,进一步研究能效优化方法.就能效模型本身而言,拟将本文提出的模型和 QoS 模型整合在一起,提出一个 QoS/能效(QoS per energy efficiency)模型.QoS/能效模型可以衡量达到某种 QoS 总体指标所需要的能源效率,反之亦然.并可以根据 QoS/能效模型定义能效优化目标,明确 QoS 和能效之间的取舍关系,解决如降低 QoS 何种指标能带来能源的明显节约,如何以最小的能源代价换取 QoS 的提升等问题.

References:

[1] Chen G, He WB, Liu J, Nath S, Rigas L, Xiao L, Zhao F. Energy-Aware server provisioning and load dispatching for connection-intensive Internet services. In: Crowcroft J, Dahlin M, eds. Proc. of the 5th USENIX Symp. on Networked Systems Design and Implementation (NSDI). San Francisco: USENIX Association, 2008. 337−350.

[2] Urgaonkar B, Shenoy PJ, Chandra A, Goyal P, Wood T. Agile dynamic provisioning of multi-tier Internet applications. Trans. on Autonomous and Adaptive Systems, 2008,3(1):1−39. [doi: 10.1145/1342171.1342172]

[3] Orgerie AC, Lefèvre L, Gelas JP. Save Watts in your grid: Green strategies for energy-aware framework in large scale distributed systems. In: Proc. of the 14th Int'l Conf. on Parallel and Distributed Systems (ICPADS 2008). Melbourne: IEEE, 2008. 171−178. [doi: 10.1109/ICPADS.2008.97]

[4] IBM project big green. http://www-03.ibm.com/press/us/en/pressrelease/21524.wss

[5] Using virtualization to improve data center efficiency. http://www.thegreengrid.org/Global/Content/white-papers/Using-Virtualization-to-Improve-Data-Center-Efficiency

[6] Rivoire S, Shah MA, Ranganathan P, Kozyrakis C. JouleSort: A balanced energy-efficiency benchmark. In: Chan CY, Qoi BC, Zhou A, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. Beijing: ACM Press, 2007. 365−376. [doi: 10.1145/1247480.1247522]

[7] Bahsoon R. Green cloud: Towards a framework for dynamic self-optimization of power and dependability requirements in green cloud architectures. In: Babar MA, Gorton I, eds. Proc. of the 4th European Conf. on Software Architecture (ECSA 2010). Copenhagen, 2010. 510−514.

[8] Kumar K, Lu YH. Cloud computing for mobile users: Can offloading computation save energy? IEEE Computer, 2010,43(4): 51−56. [doi: 10.1109/MC.2010.98]

[9] Kelenyi I, Nurminen JK. CloudTorrent—Energy-Efficient BitTorrent content sharing for mobile devices via cloud services. In: Proc. of the 7th IEEE on Consumer Communications and Networking Conf. (CCNC). 2010. 1−2.

[10] Elnozahy EN, Kistler M, Rajamony R. Energy-Efficient server clusters. In: Falsafi B, Vijaykumar TN, eds. Proc. of the 2nd Int'l Workshop on Power-Aware Computer Systems (PACS 2002). Cambridge: Springer-Verlag, 2003. 179−197. [doi: 10.1007/3-540-36612-1_12]

[11] Abdelsalam HS, Maly K, Mukkamala R, Zubair M, Kaminsky D. Analysis of energy efficiency in clouds. In: Proc. of Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns (COMPUTATIONWORLD 2009). 2009. 416−421. [doi: 10.1109/ComputationWorld.2009.38]

[12] Lefèvre L, Orgerie AC. Designing and evaluating an energy efficient cloud. The Journal of Supercomputing, 2010,51(3):352−373. [doi: 10.1007/s11227-010-0414-2]

[13] Orgerie AC, Lefèvre L. When clouds become green: The green open cloud architecture. In: Proc. of the Int'l Conf. on Parallel Computing: From Multicores and GPU's to Petascale. 2010. 228−237.

[14] Li B, Li JX, Huai JP, Wo TY, Li Q, Zhong L. EnaCloud: An energy-saving application live placement approach for cloud computing environments. In: Proc. of the IEEE Int'l Conf. on Cloud Computing (CLOUD 2009). Bangalore: IEEE, 2009. 17−24. [doi: 10.1109/CLOUD.2009.72]

[15] Younge AJ, von Laszewski G, Wang LZ, Lopez-Alarcon S, Carithers W. Efficient resource management for cloud computing environments. In: Proc. of the Int'l Green Computing Conf. Chicago: IEEE, 2010. 357−364. [doi: 10.1109/GREENCOMP.2010.5598294]

[16] Srikantaiah S, Kansal A, Zhao F. Energy aware consolidation for cloud computing. In: Proc. of the 2008 Conf. on Power Aware Computing and Systems. Berkeley: USENIX Association, 2008. 10−14.

[17] Lee YC, Zomaya AY. Energy conscious scheduling for distributed computing systems under different operating conditions. IEEE Trans. on Parallel and Distributed Systems, 2011,22(8):1374−1381. [doi: 10.1109/TPDS.2010.208]

[18] Chen YY, Das A, Qin WB, Sivasubramaniam A, Wang Q, Gautam N. Managing server energy and operational costs in hosting centers. In: Eager DL, Williamson CL, Borst SC, Lui JCS, eds. Proc. of the Int'l Conf. on Measurements and Modeling of Computer Systems (SIGMETRICS). Banff: ACM Press, 2005. 303−314. [doi: 10.1145/1064212.1064253]

[19] GridMix program. Hadoop source distribution: src/benchmarks/gridmix

[20] Jia Y, Shao Z: A benchmark for hive, PIG and hadoop. http://issues.apache.org/jira/browse/HIVE-396

[21] Sort program. Hadoop source distribution: src/examples/org/apache/hadoop/examples/sort

[22] Huang SS, Huang J, Dai JQ, Xie T, Huang B. The HiBench benchmark suite: Characterization of the mapreduce-based data analysis. In: Proc. of the 26th Int'l Conf. on Data Engineering Workshops (ICDEW). Long Beach: IEEE, 2010. 41−51. [doi: 10.1109/ICDEW.2010.5452747]

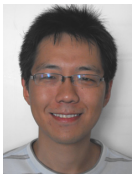
[23] DFSIO program. Hadoop source distribution: src/test/org/apache/hadoop/fs/TestDFSIO

[24] Module for Monte Carlo Pi. http://math.fullerton.edu/mathews/n2003/montecarlopimod.html

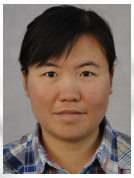
[25] WordCount program. Hadoop source distribution: src/examples/org/apache/hadoop/examples/WordCount

[26] MRBench program. Hadoop source distribution: src/examples/org/apache/hadoop/mapred/MRBench

[27] MRBench: Setting the baseDir parameter has no effect. https://issues.apache.org/jira/browse/MAPREDUCE-2398



宋杰(1980－),男,安徽淮北人,博士,副教授,CCF 会员,主要研究领域为云计算,高能效计算.



那俊(1980－),女,博士生,讲师,主要研究领域为云计算,服务计算.

李甜甜(1989－),女,硕士生,主要研究领域为高能效云计算.

朱志良(1962－),男,博士,教授,博士生导师,主要研究领域为 Web 服务,复杂网络.

闫振兴(1987－),男,硕士生,主要研究领域为云数据分析,数据管理.