doi:10.3772/j.issn.1002-0470.2016.04.001

基于资源配置等效性的数据中心能耗优化①

孙发强② 鄢贵海 李华伟③ 韩银和

(中国科学院计算技术研究所 计算机体系结构国家重点实验室 北京 100190)

摘 要 针对数据中心服务器的低能效问题,进行了利用资源配置的等效性来优化服务器能效比的研究。研究发现,应用程序的多种资源分配方案具有相同的性能,但表现出较大的能耗差异,这种现象叫做“基于性能等效的资源配置”,简称“等效配置”。基于这种观察,提出了两种优化能效比的算法——SmartRank 算法和 SmartBalance 算法。SmartRank 算法使用资源等效替换的方法寻找能耗最低的资源配置,来达到局部最优的能效比;SmartBalance 算法通过评估资源需求向量与剩余资源间的关系来均衡资源分配,同时兼顾单个应用的能耗开销,从而达到全局最大能效比。实验表明,通过对这两个算法的优化,可实现平均节省 3% 的系统能耗,局部最大可以节省 12.5% 的能耗。

关键词 功耗管理,数据中心,资源等效替换,资源利用率,资源分配

0 引 言

数据中心 IT 设备的能耗通常占数据中心能耗的 50% 以上[1],IT 设备低能效是导致功耗成本大的主要原因。例如文献[2]中的实例表明,一座功耗 7600 万瓦的数据中心,提升 IT 设备 5.2% 的能效比就能够节省 100 万美元的功耗开销,因此提高 IT 设备能效至关重要。引起数据中心 IT 设备低能效的主要因素是数据中心服务器能耗与资源利用率不成比例,服务器资源利用率低[1,3]。针对数据中心服务器的低能效问题,本项目进行了利用资源配置的等效性来提高服务器能效比的研究,并提出了两种优化能效比的算法,即 SmartRank 算法和 SmartBalance 算法,而且通过实验验证其有效性。

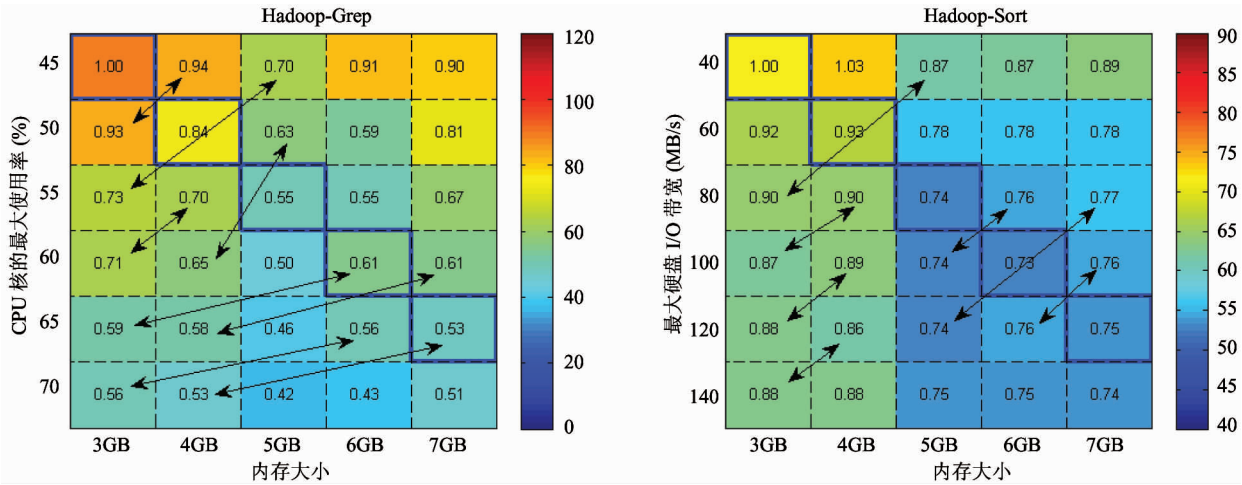
1 相关研究

服务器能耗与资源利用率不成比例的原因,除了服务器本身闲置(idle)态功耗过高外,更重要的是应用程序资源分配不当。不同的资源分配方案导致不同的资源使用行为,而不同的资源具有不同的功耗特性。在数据中心服务器中一个 CPU 的典型峰值功耗为 80 ~ 160W,一个 32G 内存的典型操作功耗为 6 ~ 12W,单碟硬盘的工作功耗约为 10W。因此不同资源配置可能具有较大的功耗差异。然而迥异的资源配置,可能具有相似的性能,如图 1(a)所示,Hadoop Grep 在不同资源配置下表现出相似的性能,例如资源配置为 3GB 内存,60% CPU 使用率和资源配置为 4GB 内存,55% CPU 使用率时的性能相近。而前者的能耗比后者高出 5%。显然为应用程序分配合适的资源,可以提高服务器的能效比。

服务器资源分配不均衡是导致服务器资源利用率低的主要原因之一。资源分配不均衡是指服务器的某种资源的分配量小于或大于其他资源的分配量。在资源隔离的环境下,一种资源过多的分配会导致其他应用程序由于该资源无法满足而不能分配到本服务器上,从而导致服务器低利用率和低吞吐

① 国家自然科学基金(61221062, 61376043, 61432017, 61572470, 61532017)资助项目。
② 男,1986 年生,博士;研究方向:数据中心能效优化和资源管理;E-mail: sunqle2010@gmail.com
③ 通讯作者,E-mail: lihuawei@ict.ac.cn
(收稿日期:2015-12-29)

量[4]。例如分配给 CPU 密集型应用 WordCount 80% 的 CPU 资源[5]，将导致 CPU 需求率大于 20% 的应用程序无法分配，进而浪费掉大量的 I/O 资源。因此保持服务器资源均衡分配能够提高服务器利用率，进而提升服务器能效比。



（a）Hadoop Grep 应用中 CPU 与 Memory 组合的等效配置　　（b）Hadoop Sort 应用中硬盘 I/O 速率与 Memory 组合的等效配置

本例中允许 3% 的性能误差；图中双箭头代表资源组合的一种等效方案；方块中的数字表示相对运行时间

图 1　等效资源配置实例

国内外学者们通过功耗管理和任务调度来优化服务器的能效比。文献[6-10]通过动态电压频率调整（DVFS）和动态功耗管理（DPM）等功耗管理的方式来调整服务器功耗/能耗。DVFS 方式通过调整资源的供电电压和频率来优化性能和功耗；DPM 方法通过调整资源所处于的睡眠状态来优化服务器的性能和功耗。文献[6]发现使用 DVFS 方式可以节省 20% 的功耗；文献[7]使用 DPM 获得了 74% 的功耗减少。文献[6]使用反馈的机制，根据应用程序 QoS 的变化不断地调整 CPU 的电压和频率；文献[7,8]使用服务器的深度睡眠状态（S5）来减少服务器空闲时的功耗，进而提高服务器的能效比；文献[9]针对不同的应用程序，权衡开启 S3 睡眠状态引起的性能损失和功耗节省来提高能效比；文献[10]结合 DVFS 和 DPM 的特征，并且根据应用程序的特征和负载情况，动态调整服务器中各个芯片的功耗状态（power state）来达到高能效比。

在应用程序级，文献[11-15]通过任务调度将应用程序整合在一起的方式来提高资源的利用率，进而提高服务器能效比。文献[11,12]将具有资源互补性的应用程序整合到一起。文献[13-15]使用资源隔离的方式将不同应用程序整合到同一服务器上。

研究者还通过感知服务器资源利用率，使用虚拟机迁移的方式来提高服务器的利用率和能效比。虚拟机迁移的核心思想是将虚拟机从资源利用率低的服务器迁移到其它机器上，然后关闭资源利用率低的服务器。

然而这些方法或者针对单一的资源，如 CPU 等，动态调整资源的分配，或者着眼于任务调度，粗略地分配资源。然而不同的资源间存在着依赖关系，有关资源组合对功耗和性能的影响缺少研究，还有资源间的均衡分配对利用率的影响。因此将这些方法应用在数据中心这类资源分配复杂、应用多样化、以资源隔离为资源分配基础的系统上时存在明显的局限性。

本文通过研究各种资源组合对系统性能的影响，发现不同的资源组合在某些情况下存在“等效性能”的现象。基于这种等效特性，在优化能效比上，本文提出了两个核心算法：SmartRank 算法和 SmartBalance 算法。SmartRank 算法使用资源等效替换的方法寻找能耗最低的资源配置，来优化单个

应用程序(局部)的能效比。SmartBalance 算法通过评估资源需求向量与剩余资源间的关系来均衡资源分配,同时兼顾单个应用的能耗开销,从而达到服务器(全局)最大能效比。实验表明,通过这两个算法的优化,系统可以获得最大 12.5% 的局部能耗节省,平均3%的全局能耗节省。

2 等效资源配置概念

为了提高资源的利用率,现代数据中心采用多个应用程序共享一个物理主机,比如 Apache 的 Yarn 和 Twitter 的 Mesos 架构。在多程序共享的环境下,为了避免一个应用程序占有绝大部分的资源或保证应用程序足够的资源供应量,通常使用资源隔离的方式来满足服务需求,即限制应用程序使用资源的最大量。最常用的资源隔离的方法有虚拟机资源隔离、应用隔离及 Cgroup 隔离。这样应用程序的资源配置 $R = [s_1, s_2, \cdots, s_n]$ 和性能 P 间的关系可以记为 $P = f(R)$。

对于同一应用程序来说,相似的性能,有不同的资源配置方案。比如 Hadoop Grep 应用程序。如图 1(a)所示,对于不同的 CPU 配额(最大允许利用率)和内存大小,Hadoop Grep 具有不同的性能。随着 CPU 配额的增加,Hadoop Grep 的运行时间不断减少;同时在同一 CPU 配额下,随着内存容量的增加性能也相应地增加。总体来说,性能沿着资源分布图的斜对角线呈增长趋势。在对角线两侧或相邻的区域,Hadoop Grep 的性能相差很小,如图中双向箭头所示。如果忽略应用程序运行时的随机误差[16,17],我们可以把它们看作是相同的性能。Hadoop Sort 应用在内存与硬盘 I/O 带宽资源组合中,也存在这种资源配置与性能多对一的现象,如图 1(b)所示。我们把这种资源配置不同,但性能相近的现象叫做等效资源配置。也就是,给定资源配置 $\boldsymbol{R}_1$、$\boldsymbol{R}_2$ 及性能关系 $P_1 = f(R_1)$、$P_2 = f(R_2)$,如果 P_1 等于 P_2,那么 $\boldsymbol{R}_1$ 与 $\boldsymbol{R}_2$ 在性能上等效,称 $\boldsymbol{R}_1$、$\boldsymbol{R}_2$ 为等效资源配置。

资源组合的这种等效特性,为能耗优化和资源均衡分配提供了优化空间。由于不同资源消耗的能耗差异较大,例如一个 CPU 的峰值功耗为 100 瓦,而硬盘单碟的典型工作功耗为 10 瓦,32GB 内存工作功耗典型值为 8 瓦左右,因此在这样的服务器上工作时,不同资源配置下同一应用程序具有不同的能耗开销。从这些等效配置中,选取能耗最小的配置就可以尽可能地提高单个应用程序的能效,即局部最优化。

资源配置在性能上等效的特性也可以均衡资源的使用,提高服务器的利用率和能效比。在资源分配不均衡的情况下,某一种资源过度分配,将导致其他应用程序无法获得相应的资源,降低服务器的利用率。如图 2 所示,在这里我们使用长方形来代表资源的一种配置方案,长方形的水平方向表示该配置最大允许使用的内存资源,垂直方向表示 CPU 资源的最大使用率,那么应用程序 APP_1、APP_2 的等效资源配置如图 2(a)所示,其中标记 E 代表应用程序在该资源配置下消耗多少个单位能耗;图 2(b) ~ 图 2(e)为 APP_1、APP_2 的四种可能的资源配置方案及相应的总能耗开销。如果一味地追求局部应用程序能效比的最大化,将导致服务器 CPU 资源无法满足 APP_1 的需求,使得 APP_2 无法分配到服务器上,最终降低利用率,如图 2(b)所示;而图 2(c) ~ 图 2(e)借助资源配置的等效性,使得 APP_1、APP_2 能够同时运行。然而不同的资源等效配置方案,能耗差异较大,图 2(d)能耗相对于图 2(e)则增长了 30%。因此在资源配置等效替换时,需要协调应用程序间的资源分配,才能提高服务器的利用率及能效比。

综上所述,从应用程序的性能出发,服务器的各种资源配置间存在等效替换的特性;在局部范围,资源配置的等效替换可以提高单个应用程序的能效比;协调应用间的资源配置可以优化服务器的能效比,属于组合优化问题。基于这些观察和发现,本文提出用 SmartRank 算法来提高单个应用程序的能效比,用 SmartBalance 算法来提高服务器能效比。

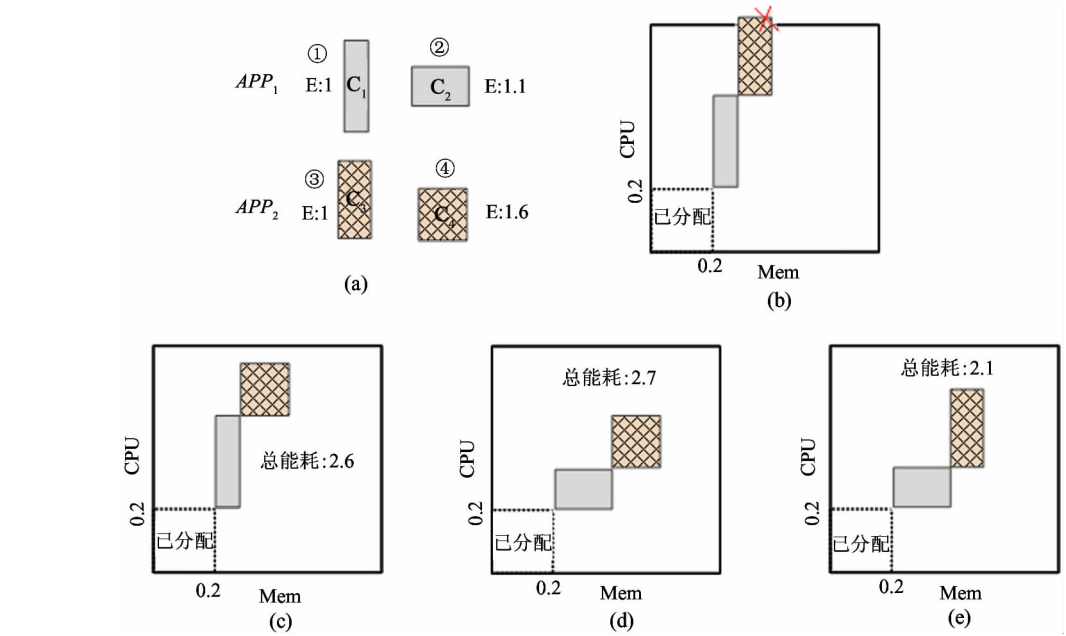


图 2　利用不同资源配置性能等效的特性优化全局资源分配(E 代表应用程序在该资源配置下消耗多少个单位能耗)

3　基于资源配置等效性的能耗优化算法

3.1　问题描述

为了方便问题的形式化描述,我们首先定义三个关键变量——等效区间 ER、等效资源配置 EC 及资源约束函数 $d_j(R)$。我们将应用程序在某一性能点上允许波动的范围定义为等效区间 ER, 例如图 1 中性能区间(0.92 ~0.95)可以称为 Hadoop Grep 的一个 ER。我们使用 $EC(ER,A)$ 来表示应用程序集 A 在等效区间 ER 上的等效资源配置集,相应的 $EC(er_i,a_i)$ 表示应用程序 a_i 在等效区间 er_i 上的等效资源配置集。比如图 2 中 APP_1 的等效配置为 $\boldsymbol{C}_1$ 和 $\boldsymbol{C}_2$; APP_2 的等效配置为 $\boldsymbol{C}_3$ 和 $\boldsymbol{C}_4$;那么该应用程序组 $\{APP_1,APP_2\}$ 的等效配置为单个应用等效配置的组合$(\boldsymbol{C}_1,\boldsymbol{C}_3)$、$(\boldsymbol{C}_1,\boldsymbol{C}_4)$,$(\boldsymbol{C}_2,\boldsymbol{C}_3)$和$(\boldsymbol{C}_2,\boldsymbol{C}_4)$。$d_j(R)$ 表示资源配置 R 的第 j 种资源的需求量。我们将这些符号整理到表 1 中。

那么利用资源等效替换优化服务器能效比的目标函数为:

$$\min \quad E(R) \tag{1}$$

S. T.

$$R \in EC(ER,A) \tag{2}$$

$$d_j(R) \leqslant y_j,\ j = 1,\cdots,n \tag{3}$$

其中 y_j 表示服务器上第 j 种资源的上限, R 为应用程序集 A 在 ER 上的一种资源等效配置, $E(R)$ 代表该配置下的能耗值。

表 1　符号标记说明

$RR = [s_1,s_2,\cdots,s_n]$	资源配置向量。它代表了资源分配器分配给应用程序的资源容量。
$d_j(R)$	为资源配置向量 R 中第 j 种资源的容量
$EC(ER,A)$	应用程序 A 在等效区间 ER 上的等效资源配置向量集
$E(R)$	应用程序在资源配置 R 下的能耗

该优化目标面临一个挑战:获取应用程序等效配置集。获取应用程序的等效配置集,需要应用程序在不同的配置下进行运行。在实际生产环境中,很难满足这种要求。我们发现通过机器学习的方式可以获得某些应用程序的等效配置集,比如在线应用,入侵检测和日志分析等周期执行的服务。应用程序每次执行的时候,我们改变其资源配置并记录下相应性能和能耗。然后将性能落在相应等效区间的配置作为等效资源配置存储在相应的数据库中。这样我们解决了该挑战。

对于单个应用程序来说(局部优化),只需要算法遍历该应用的等效配置集,找到满足资源限制的能耗最小的配置即可;而全局优化则存在时间复杂度的问题。由上文可知,一组应用程序的等效资源配置集是由单个应用程序的等效资源配置组合而成。因而求解最优资源配置是一个组合优化问题,即 NP 问题。在实际的应用环境中,需要使用启发式算法来加快求解过程。本文针对这两种情况,分别提出了 SmartRank 算法优化局部资源配置,和启发式方法 SmartBalance 来优化全局资源配置。SmartBalance 算法通过引入 WET 标准,优先选择 WET 值小的配置来近似最优解。

3.2 SmartRank 算法

SmartRank 算法的主要思想是遍历所有等效资源配置,找到满足资源限制并且能耗最小的资源配置向量。该算法的复杂度为 $O(N)$,伪代码如下:

SmartRank 算法

输入:等效区间 er_i,资源与性能关系矩阵 $\boldsymbol{M}$,及相应能耗矩阵 $\boldsymbol{EN}$

输出:资源配置向量 $optR$

```
MinE = inf; optR = NULL
Ω_p = EC(er_i, a_i)
For each R_1 IN Ω_p
  If E(R_1) < MinE and Satisfy(R_1) == 1
    MinE = E(R_1)
    optR = R_1
  End
Done
```

其中 Satisfy($\boldsymbol{R}_1$)函数为判断资源配置 $\boldsymbol{R}_1$ 是否满足服务器物理资源限制,满足的话返回 1,否则返回 0。

3.3 SmartBalance 算法

由上文可知,使用资源配置等效替换来获取全局最优解是组合优化问题,因此在实际的应用环境中,需要使用启发式算法来加快求解过程。贪心的选择局部最优的资源配置向量,会导致某一资源分配过量,进而无法分配资源给其他应用程序,如图 2(b)所示。资源分配不均衡是导致贪心选择失效的主要原因。因此均衡资源分配是改善贪心选择方法低利用率的关键。鉴于此,本文提出了 SmartBalance 启发式算法来优化全局资源配置。SmartBalance 算法的基本思想是尽量保持各种资源相对均衡的分配,从而达到服务器有足够的资源满足未来的应用程序。通俗来讲,就是哪种资源剩余较多,分配哪种资源相应多点。

研究发现资源配置向量与服务器剩余资源向量间的夹角大小可以衡量资源分配的均衡性。资源分配越均衡,服务器资源利用率越高。如图 3 所示。在图 3(a)中,应用程序的资源分配向量 $\boldsymbol{R}$ 与服务器剩余资源向量 $\boldsymbol{Q}$ 有较大的夹角,最终导致了服务器只能再分配 2 个应用程序;而在图 3(b)中,资源分配向量与服务器资源余量始终保持一致的方向,资源分配相对均衡,因而在分配完 2 个应用后,还有足够的资源供其他应用程序使用。由此可知,保持与服务器资源余量相同的方向,资源分配相对均衡,有利于提高服务器的利用率。资源分配向量 $\boldsymbol{R}$ 与服务器剩余资源向量 $\boldsymbol{Q}$ 方向上的差异程度,本文使用它们之间的夹角 θ_R 来衡量,如下式所示:

$$\theta_R = \arccos \frac{< \boldsymbol{R}\ \boldsymbol{Q} >}{|\boldsymbol{R}| \cdot |\boldsymbol{Q}|} \tag{4}$$

其中 $<\boldsymbol{R}\ \boldsymbol{Q}>$ 为 $\boldsymbol{R}$ 和 $\boldsymbol{Q}$ 的内积,$|\boldsymbol{Q}|$ 为 $\boldsymbol{Q}$ 的模。θ_R 越小代表资源分配越均衡,反之资源分配越不均衡。为了提高服务器利用率,应该尽量减小 θ_R 的值。

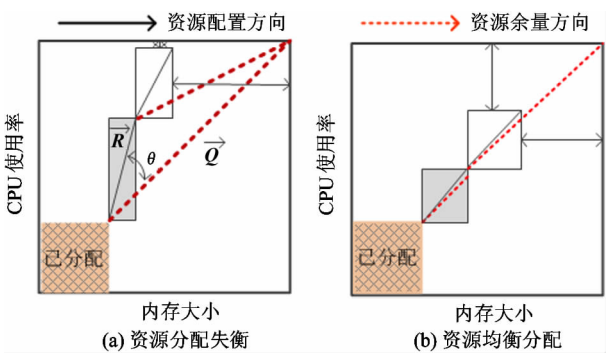


图 3 不同的资源分配方案(相同颜色块为等效配置)

优先分配能耗差异较大的应用程序,能够使服务器获得较少的能耗。这是因为,即使随后的应用程序的资源配置方案不是能耗最小的,该应用程序消耗的能耗与最优解相比也相差较小,降低对整体能耗的影响。而且全局最优解是整体能耗开销最小的资源配置方案,将每个应用程序的等效资源配置

按照能耗从小到大排序,可以加快搜索速度。同时全局优化需要兼顾资源分配均衡。因此对于等效资源配置的优先选择上,需要兼顾能耗和整体资源分配均衡。本文使用加权能量通量(weighted energy throughput,WET)来作为等效资源配置优先级衡量的标准,定义如下:

$$WET = (1-\rho)\cdot E(R) + \rho\cdot\theta_R \quad (5)$$

其中 ρ 为资源分配均衡程度占整体的比重。它表示在选择资源配置时,我们更倾向于能耗还是利用率。当 $\rho = 0$ 时,贪心选择等价于能耗优先,当 $\rho = 1$ 时,贪心选择等价于利用率优先。为了降低搜索空间,对每一个应用程序我们取 WET 最小的 N 个等效资源配置。那么 SmartBalance 算法过程如下。

SmartBalance 算法

输入: 应用程序集 A,服务器资源余量 $\boldsymbol{Q}$
输出: 资源配置方案 $optR$

```
1  optR = [ ]; flag = 0; cand = { }
2  B = Sorted( A )
3  Iteration(1)
4  optR = MinOpt (cand)
```

Iteration(j) 函数

```
1  If j > m
2    cand = cand ∪ optR
3  ELSE
4      K = EC(er_j, B_j); flag = 0
5    For each conf IN TopN(K)
6      If Satisfy(conf) = = 1 then
7          optR = optR ∪ conf; Q = Q − conf;
8          Iteration(j + 1); flag = 1
9      End
10   If flag = = 1 then
11 Q = Q + conf
12         optR = optR − conf; flag = 0
13 End
14   End
15 End
```

其中 Sorted( ) 函数按照应用程序在等效配置集中的能耗变化,由大到小排序程序;TopN( ) 为 WET 最小的 N 个配置;MinOpt( ) 函数是在所有可行的资源配置方案中寻找能耗最小的资源配置。Iteration( ) 函数为回溯迭代过程,遍历搜索空间寻找所有能够满足资源约束的资源配置方案,一旦找到就将其保存到全局变量 $cand$ 中,该过程的算法复杂度为 $O(N^{m-1})$,其中 m 为应用程序的个数,N 为最大的等效资源配置数;SmartBalance 算法关键在于回溯迭代寻找可行解,因此算法的复杂度为 $O(N^m)$。虽然算法的复杂度看起来比较高,但是我们发现,当 N 取 3 时,SmartBalance 算法的解与最优解的差距平均为0.5%(如图 6 所示)。因此本算法在实际应用中,可操作性很强。

4 实验结果与分析

4.1 实验平台及测试程序

为了验证资源配置等效替换的优势,本文使用 Cgroup 来控制 KVM 虚拟机 CPU 使用率、硬盘 I/O 读写速率。KVM 运行在对称多处理器(SMP)结构的曙光服务器上,服务器总共具有 24 个处理器核、24GB 内存,硬盘平均读速率为 170MB/s、写速率为 140MB/s。KVM 使用 2 个处理器核,启动 KVM 后,使用 Cgroup 将 KVM 绑定到相应的处理器核上,同时限制处理器的 CPU 最大使用率、硬盘的 I/O 最大读写速率;使用 libvirt 来控制 KVM 的运行内存的大小。

本文使用 BigDataBench[18] 中的测试程序来获取应用程序在不同的资源配置向量下的性能,应用负载见表 2;使用 CPU 利用率来获取应用程序运行时消耗的能耗[19]。本文以单核 5% 的粒度来调整 CPU 的利用率、20MB/s 的速率改变硬盘 I/O 的读写性能、1GB 的大小改变内存资源分配。测试用例来自四类:微基准测试(Micro Bechmarks)、web 应用、机器学习算法及 HDFS 基准测试。Hadoop 版本为 2.7.0,默认块大小为 128MB;Mahout 版本为 0.6。KVM 虚拟机运行在 Ubuntu 14.04,Java 版本为 1.7.0。

表 2　资源配置及负载

Cgroup 资源配置		
CPU 单核最大使用率范围		[0.4 ~1]
硬盘 I/O 读写速率范围(MB/s)		[40 ~140]
内存大小(GB)		[2 3 4 5 6 7]
应用程序及负载		
微基准测试	Hadoop Grep Hadoop Sort	输入:1GB
Web 应用	Hadoop PageRank Hadoop Connect	节点数:32768 边数:99489
机器学习算法	Mahout KMeans Mahout Bayesian	输入:1GB
HDFS 测试程序	Hadoop DFSIOTest	输入:6 个 256MB 文件
W1	Sort、DFSIOTest、Grep、Kmeans	
W2	DFSIOTest、Grep、Kmeans、Bayesian	
W3	Grep、Kmeans、Bayesian、Connect	
W4	Kmeans、Bayesian、Connect、PageRank	

4.2　结果分析

我们分别从局部和全局来分析等效资源替换在能耗上的优化结果。能耗优化的程度不但取决于应用程序本身,而且还取决于应用程序的等效区间和算法的资源选择。

应用程序等效区间间的能耗变化迥异。我们选用较小的性能片段作为应用程序的等效区间,如图 4中的 ER1、ER2 和 ER3。由图 4 可知,不同的等效区间上,应用程序的能耗差异较大。如 ER2 上 Hadoop dfsio 能耗差异达到 6%,而在 ER3 上能耗差异只有 2.5%。这将导致应用程序使用等效资源配置时,有些应用程序能耗降低明显,有的不明显,如图 5所示。该发现启发我们求解近似全局最优解时,优先选择能耗降低明显的应用程序。

本文对单个应用程序进行能耗优化研究发现,等效资源配置使应用程序获得了不同程度的能耗节省。实验结果如图 5 所示(这里我们设置等效区间长度为 0.05)。利用资源等效配置,SmartRank 使 Hadoop Sort 最大可以获得 12.5% 的能耗节省,而 Hadoop PageRank 最大获得 6.8% 的能耗节省。然而 SmartRank 算法仅考虑单个应用的能耗节省,没有考虑服务器整体的利用率,即忽略了资源分配的均衡性。我们在此基础上使用 WET 来均衡资源分配。实验结果显示,在服务器剩余资源为[80,100,5]的情况下,SmartRank 算法也能获得较高的能耗节省,如图 5 深色柱体所示。在局部使用 WET 优化能耗,虽然不能够保证应用程序获得最大的能耗节省,但是能够尽可能地接近最优值。如 Hadoop Sort、Hadoop Grep 与 Hadoop PageRank 获得了最大的能耗节省,Hadoop Bayesian、Hadoop Conn 和 Hadoop Kmeans 能耗节省仅次于最优解,而 Hadoop dfsio 与最优解相差 30%。这是因为 WET 参数不但注重能耗优化的量,而且还兼顾系统的利用率。当最优资源配置与当前的资源剩余量有较大的差异时,相对于能耗值来说,资源的均衡使用占据主导,SmartRank 选择了与资源剩余量相近,并且能耗较小的

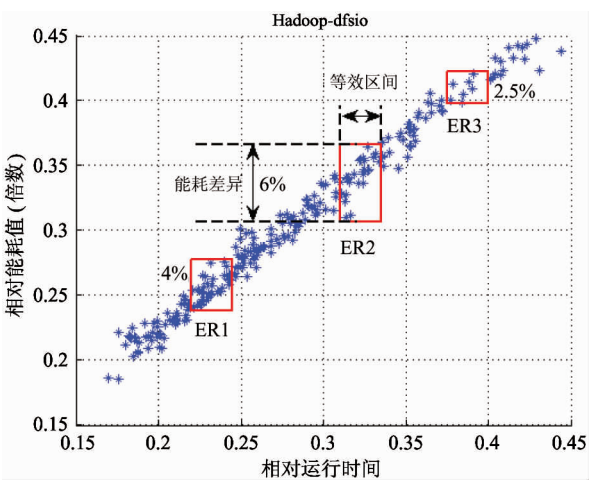


图 4　Hadoop-dfsio 应用在不同资源配置下的能耗差异分布

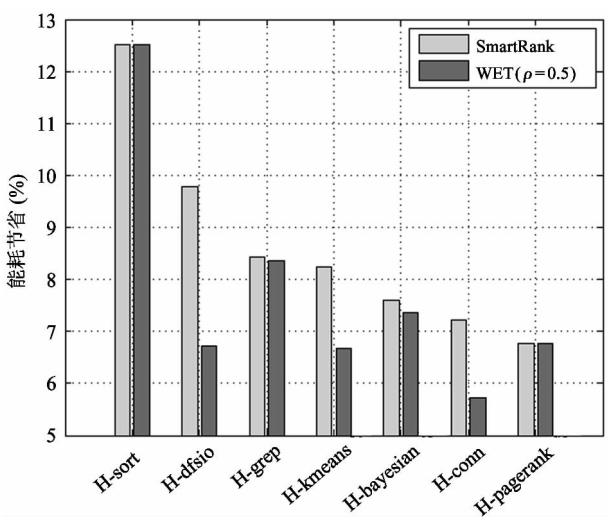


图 5　SmartRank 能耗优化评估。“H”代表 Hadoop 程序

配置。能耗优化量与资源分配均衡间的比重可以使用参数 ρ 来调节;下文中我们将研究 ρ 对能耗节省的影响。

接下来我们研究 SmartBalance 对服务器能耗的优化效果。我们通过对 4 个应用负载 W1、W2、W3 和 W4(如表 2 所示)在总资源配置为[420,620,34],服务器资源余量为[400,580,30]的情况下的能耗变化来分析 SmartBalance 的有效性。为了比较 SmartBalance 算法在能耗优化上的优势,我们使用随机选择算法(Random),即在所有可能的资源配置方案中,随机选择一种配置,穷举算法(Oracle)作对比。同时为了避免随机(Random)算法选取最坏的情况或最优的情况,我们选择能耗节省处于平均值的资源配置。如图 6 所示,SmartBalance 算法整体能耗仅比最优值多 0.5% 的功耗,而随机算法平均比最优解多 3.3% 的能耗开销。不同的负载下,不同的资源配置具有不同的能耗差异,导致 SmartBalance 算法具有不同的优化空间。比如在负载 W1 下,不同资源配置平均具有 4.5% 的能耗差异;W2 只有 2.8% 的能耗差异。因此 SmartBalace 算法最大减少 4% 的能耗,最小获得 2.5% 的能耗节省;总体来说,SmartBalace 算法平均获得 3% 的能耗节省。

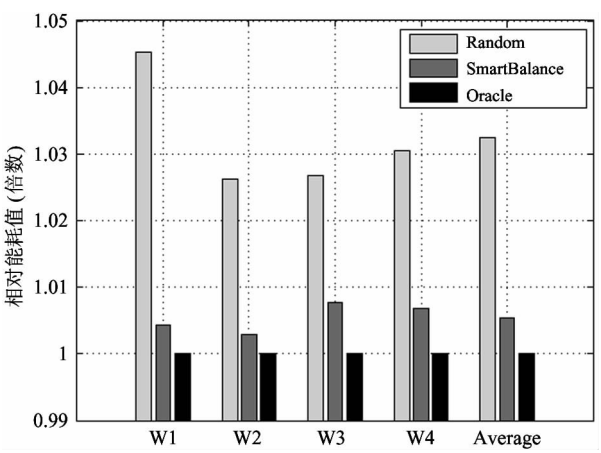


图 6　随机、SmartBalance 和穷举算法的能耗优化对比

在本文中,我们使用权重值 ρ 来调节资源均衡使用和能耗节省之间的重要程度。ρ 值取 0 时,SmartBalance 算法相当于能耗优先的算法;当 ρ 取 1 时,SmartBalance 算法相当于利用率优先的算法。我们在服务器剩余资源为[80,100,5]下,研究 ρ 对局部最优解的能耗影响。如图 7 所示,随着 ρ 值的增大,能耗开销不断地增大;Hadoop Grep、Hadoop Connect component 及 Hadoop PageRank 在 ρ 值大于 0.54 时,能耗已经达到最大值;而 Hadoop Bayesian、Hadoop sort、Hadoop kmeans 及 Hadoop dfsio 在 ρ 值达到 0.78 以上时,能耗逐渐接近最大值。权衡资源使用与能耗支出,ρ 应该小于 0.54,在本文中我们倾向选择 ρ =0.3。

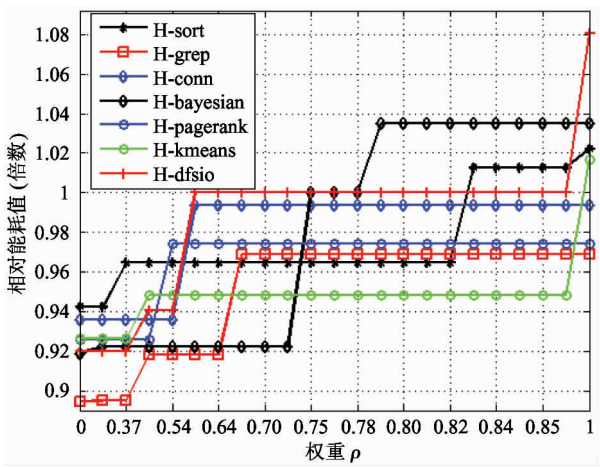


图 7　权重 ρ 对能耗优化的影响

5　结　论

本文使用资源配置等效替换的策略,改善在局部由于资源分配不当引起的额外能耗开销,和整体资源分配时,由于资源分配不均衡导致的服务器低利用率、低吞吐量等性能损失,来提高数据中心服务器的能效比。针对局部资源不当导致的能耗浪费,本文提出了 SmartRank 算法,该算法能降低 12.5% 的能耗浪费;在全局上,我们提出了启发式算法 SmartBalance,该算法获得与最优资源配置相似的能耗节省,仅差0.5%。

本文通过资源重新组合的方法来提高能效比,弥补了传统功耗管理单一资源调整等不足,及资源分配不均引起的利用率低等问题,因此本方法与这些方法是正交的。协同传统功耗管理方法和资源配置等效替换方法,最大化服务器能效比是我们将来需要探寻的方向。

参考文献

[1] Zhang Q, Shi W S. UPS-aware workload placement in enterprise data centers. *IEEE Computer Magazine*, 2015, (1):1-7

[2] Barroso L A, Holzle U. The case for energy-proportional computing. *Computer*, 2007, 40(12): 33-37

[3] Barroso L A, Clidaras J, Hoelzle U. The Datacenter as A Computer: An Introduction to The Design of Warehouse-scale Machines. Morgan & Claypool Publishers, 2013. 1-154

[4] Lee S, Panigrahy R, Prabhakaran V, et al. Validating heuristics for virtual machines consolidation, MSR-TR-2011-9. Microsoft Research, 2011

[5] Huang S S, Huang J, Dai J Q, et al. The HiBench benchmark suite: Characterization of the MapReduce-based data analysis. In: Proceedings of the 2010 IEEE 26th International Conference on Data Engineering Workshops, California, USA, 2010. 41-51

[6] Lo D, Cheng L Q, Govindaraju R, et al. Towards energy proportionality for large-scale latency-critical workloads. In: Proceeding of the 41st Annual International Symposium on Computer Architecuture, Minneapolis, USA, 2014. 301-312

[7] Meisner D, Gold B T, Wenisch T F. PowerNap: eliminating server idle power. In: Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems, Washington, USA , 2009. 205-216

[8] Meisner D, Wenisch T F. DreamWeaver: architectural support for deep sleep. In: Proceedings of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems, London, United Kingdom, 2012. 313-324

[9] Isci C, McIntosh S, Kephart J, et al. Agile, efficient virtualization power management with low-latency server power states. In: Proceedings of the 40th Annual International Symposium on Computer Architecture, Tel-Aviv, Israel, 2013. 96-107

[10] Liu Y P, Draper S C, Kim N S. SleepScale: Runtime joint speed scaling and sleep states management for power efficient data centers. In: Proceedings of the 41st Annual International Symposium on Computer Architecture, Minneapolis, USA, 2014. 313-324

[11] Delimitrou C, Kozyrakis C. Quasar: resource-efficient and QoS-aware cluster management. In: Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Salt Lake City, USA, 2014. 127-144

[12] Delimitrou C, Kozyrakis C, Paragon: QoS-aware scheduling for heterogeneous datacenters. In: Proceedings of the 18th International Conference on Architectural Support for Programming Languages and Operating Systems, Houston, USA, 2013. 77-88

[13] Lo D, Cheng L Q, Govindaraju R, et al. Heracles: improving resource efficiency at scale. In: Proceedings of the 42nd Annual International Symposium on Computer Architecture, Portland, USA, 2015. 450-462

[14] Ma J, Sui X F, Sun N H, et al. Supporting differentiated services in computers via programmable architecture for resourcing-on-demand(PARD). In: Proceedings of the 20th International Conference on Architectural Support for Programming Languages and Operating Systems, Istanbul, Turkey, 2015. 131-143

[15] Yang H L, Breslow A, Mars J, et al. Bubble-flux: precise online QoS management for increased utilization in warehouse scale computers. In: Proceedings of the 40th Annual International Symposium on Computer Architecture, Tel-Aviv, Israel, 2013. 607-618

[16] Chen T S, Guo Q, Temam O, et al. Statistical performance comparisons of computers. *IEEE Transactions on Computers*, 2015, 64(5): 1442-1455

[17] Yan G H, Sun F Q, Li H W, et al. CoreRank: Redeeming "Sick Silicon" by dynamically quantifying core-level healthy condition. *IEEE Transactions on Computers*, 2015, 65(3):716-729

[18] Wang L, Zhan J F, Luo C J,et al. BigDataBench: A big data benchmark suite from internet services. In: Proceedings of the 20th IEEE International Symposium On High Performance Computer Architecture, Orlando, USA, 2014. 488-499

[19] Fan X, Weber W D, Barroso L A. Power provisioning for a warehouse-sized computer. In: Proceedings of the 34th Annual International Symposium on Computer Architecture, New York, USA, 2007. 13-23

Improvement of datacenters' energy efficiency based on resource allocation equivalency

Sun Faqiang, Yan Guihai, Li Huawei, Han Yinhe
(Key Laboratory of Computer System and Architecture, Institute of Computing Technology, Chinese Academy of Science, Beijing 100190)

Abstract

The equivalence of resource allocation was used to improve the energy-efficiency proportion of servers to solve the low energy efficiency problem of modern datacenters. The study observed that variety of resource allocation schemes for applied programs have the same performance but they are apparently different in energy consumption. This phenomenon was called the equivalent allocation. Based on the observation, two algorithms for improving the energy-efficiency proportion, named SmartRank and SmartBalance, were proposed. The SmartRank algorithm uses the resource equivalent replacement to seek the resource allocation consuming lowest energy to achieve the energy-efficiency proportion of local optimization. The SmartBalance algorithm balances the resource allocation through evaluation of the relation between the vector of resource demand and the surplus resources while takes account of the cost for energy consumption to achieve the maximum local energy-efficiency proportion. The experiments show that the SmartRank algorithm can reduce the energy of a single application as much as 12.5%, and the SmartBalance algorithm can save 3% of energy of the whole system on average.

Key words: power management, datacenter, resource equivalent replacement, resource utilization, resource allocation