# Linux 性能监测工具测评研究

张慧嫱,程 华
(江南计算技术研究所,江苏 无锡 214083)

**摘 要**:性能监测是 Linux 系统运维和性能调优的关键,选择合适的性能监测工具十分必要。目前用户多从功能角度进行选择,对性能监测工具的资源占用情况考虑甚少。当持续监测高频程序时,工具自身负载不容忽视,应在前期选择时纳入考虑。针对性能监测工具缺乏资源分析的问题,设计了定量测评实验,提出了功能、性能、功耗三个维度的测评方案。选取覆盖系统主要部件的典型性能测试工具进行实验,包括 free、iostat、ifstat、sar 和 collectl。实验数据表明,监测单个部件时,使用子系统监测工具比使用全系统细粒度监测工具单点监测节省至少 30% CPU 资源,能耗至少降低 30%;同时监测系统多个部件时,全系统监控工具也有很大差异。对工具占用资源差异进行了分析,给出了工具选择策略,为用户选择、测评及设计性能监测工具提供参考。

**关键词**:Linux 系统;性能监测工具;功耗;测评
**中图分类号**:TP39 **文献标识码**:A **文章编号**:1673-629X(2018)09-0088-06
doi:10.3969/j.issn.1673-629X.2018.09.019

# Research on Evaluation of Linux Performance Monitoring Tools

ZHANG Hui-qiang, CHENG Hua
(Jiangnan Institute of Computing Technology, Wuxi 214083, China)

**Abstract**: Performance monitoring is the key to Linux system maintenance and performance optimization, and it is necessary to select an appropriate monitoring tool. At present, most users select tools according to their functions but consider little on their resource occupations. When monitoring high-frequency program continuously, the load of tools cannot be ignored and should be taken into account in the early selection. Due to the lack of resource analysis on performance monitoring tools, we design a quantitative evaluation experiment and propose an evaluation method on function, performance and power consumption. Some typical performance tools have been evaluated in the experiment including free, iostat, ifstat, sar and collectl. The experimental data shows that when monitoring a single unit, subsystem monitoring tools save at least 30% CPU resources and reduces energy consumption by at least 30% compared to full-system monitoring tools. When monitoring multiple units of system, different full-system monitoring tools vary a lot. Finally, we analyze the reason of resource consumption difference and propose some suggestion for selecting tools, which provides an effective reference for users on selecting, evaluating and designing performance monitoring tools.

**Key words**: Linux system; performance monitoring tools; power consumption; test and evaluation

## 0 引 言

性能监测工具可以实时监测系统的运行状态及资源使用情况[1-2],包括 CPU、内存、磁盘、网络等。近年来,随着“绿色计算”[3]概念的提出,低功耗成为软件设计、工具选择时的重要考量[4]。频繁监测使用高频的程序(如操作系统)时,监测工具给系统带来的性能、功耗负载不容忽视,必须引起重视。因此,除了实现的功能,了解性能监测工具运行中的性能、功耗情况十分必要,是选择合适的监测工具时必须考虑的重要因素。

Linux 系统中提供了丰富的性能监测工具[2],既有针对单个单元的监测工具,如 fdisk 查看硬盘分区信息,free 查看内存使用情况,iostat 实时记录整个 CPU 与接口设备的 I/O 状态;也有涵盖多个单元的全面系统监测工具,如 top 动态监测进程的工作状态及运行情况,sar 从多方面对系统活动进行报告,常用于全日

**收稿日期**:2017-10-07 **修回日期**:2018-02-27 **网络出版时间**:2018-05-16
**基金项目**:国家重点研发计划战略高技术重点专项(17-H863-01-ZT-004-009-01)
**作者简介**:张慧嫱(1992-),女,硕士研究生,研究方向为软件测试与评估、能耗评价;程 华,高级工程师,CCF 理事(30903M),研究方向为绿色计算、系统性能评价、软件测试与评估、能耗评价。
**网络出版地址**:http://kns.cnki.net/kcms/detail/61.1450.TP.20180515.1651.036.html

监测系统状态,collectl 则可以监测几乎全部的系统单元。不同监测工具采集不同种类和数量的硬件性能事件,占用的资源和消耗的能源也不同。使用时应根据需求选择合适的工具,才能有效实现系统监测与性能调试。

理想的性能监测工具应具备两方面的特性:(1)监测的全面性。应覆盖 CPU、磁盘、内存、网络等多个监测项目,实现系统单元的细粒度监测;(2)轻量级、低负载。占用系统资源尽量少,对系统运行的影响应尽可能小。当前,相关文献中对监测工具的分析主要集中在功能实现方面[5-7],对资源消耗情况的分析较少或不够全面[8-10]。

针对性能监测工具缺乏全面的资源、能耗分析这一问题,文中设计了定量测评实验。选取 free、iostat、ifstat、sar 和 collectl 等主流 Linux 性能监测工具,通过实验测量得到了典型监测工具的性能和功耗数据。

## 1　现状概述

### 1.1　性能监测原理

目前流行的大多数性能监测工具都是通过读取/proc文件系统下特定的编码文件[11],经过一定的编码解释后输出成可读的形式,来实现特定系统单元的实时性能监测。/proc 文件系统是 Linux 内核的一个虚拟文件系统[12],不仅提供了 Linux 内核的当前状态,还包含了系统硬件以及详细的进程信息,例如内存管理、进程相关、文件系统、设备驱动程序、系统总线、电源管理、终端、系统控制参数、网络等,几乎涵盖了内核的所有部分。/proc 系统以文件的形式提供了内核与进程之间的通信接口,用户或应用程序可以通过/proc 系统动态访问内核内部数据结构、读取系统的相关信息,还可以通过改变/proc 下相应的文件来改变内核设置。另一方面,计算机系统中提供了丰富的硬件性能计数器[13-14]和传感器[15-17],可直接采集系统的硬件信息和活动数据,并通过相应的驱动将检测到的硬件信息提供给操作系统,某些性能监测数据便是通过读取这些特定的配置文件获得。

### 1.2　监测工具分类

最早的性能监测工具产生于 Unix 系统中,便于系统管理员进行系统维护和性能调节。随着 Linux 系统的流行,这些工具大多经过一定程度的改写以适用于各个 Linux 系统发行版。随着 Linux 内核的不断升级、计算机硬件设备的不断完善,性能监测工具更加丰富,覆盖的监测项目更加全面,监测的数据也更加丰富和准确。根据监测单元的数目及监测数据的类型,现有的性能监测工具可大致分为以下三类:

(1)子系统性能监测工具:监测一个或少数几个子系统,提供特定子系统的全面性能监测数据。典型的如 free 和 iostat,其中 free 属于 procps 包,1992 年由 Brian Edmonds 发布,用于在终端显示内存和交换空间的使用情况,包括已用内存、buffers、cache 等,简易实用,几乎所有系统管理员日常都会用到;iostat 是 sysstat 系统监测软件包的一部分,由 Sebastien Godard 于 1998 年使用 C 语言发布,汇总了单个磁盘的使用信息,可报告 CPU 及磁盘使用情况,包括块设备的读和写,跟踪存储设备的性能,用于调整系统配置取得磁盘之间更好的 I/O 负载平衡。

(2)全系统细粒度性能监测工具:监测多个子系统,能提供多个子系统全面细粒度的性能监测数据。全系统细粒度监测工具功能强大,应用最为广泛的是 sar 和 collectl。其中,sar 工具也属于 Sysstat 包,最早写于 1999 年,随后不断完善,成为 Linux 系统监测的全能工具,可以统计实时数据或历史数据,收集、报告和保存系统的不同活动指标,包括 CPU 使用情况、文件读写情况、磁盘 I/O 状况等详细信息,帮助用户全方位监测系统,找出性能瓶颈;collectl 作为后起之秀,2003 年由 Mark Seger 采用 perl 语言写成,经过多年发展,如今已成为一个出色的轻量级全系统监测工具,可以覆盖众多系统单元、报告丰富资源信息,在功能方面既和 sar 有重合点,又展现出了很多不同于 sar 的细节和特性,例如增加了对计算机集群的监测支持,可以单独扮演 ps、top 等其他实用监测工具的角色。

(3)全系统概要监测工具:可用于监测多个子系统,描述各系统总体运行情况的概要统计数据。前面介绍的 top 便是典型的全系统概要监测工具,top 提供了一系列系统范围的概要统计信息,包括 CPU 利用率、平均负载、内存及交换空间使用情况等,还可以监测运行最多的进程/任务状态,根据 CPU 用量等对进程进行排序,每隔一定间隔刷新屏幕。使用 top 命令,用户可快速掌握当前主要系统部件的大致性能情况,并对资源占用较高的进程实行专门监测。在此基础上,许多学者对 top 工具进行了不同的改进。由于 top 对/proc 系统拍快照,无法监测在快照之前结束的寿命较短的程序[1],针对此情况,Gerlof Langeveld 等于 2000 年发布了 atop 工具,使用进程核算技术,对短寿命的程序进行了捕捉显示;此外,Hisham Muhammad 提供了一个更为友好的 top 工具——htop,可以以更加多彩的方式显示更加丰富的统计信息,并对重要数据进行高亮显示。

有关各监测工具的使用方法和原理,官网或者 man 手册都有详细的说明信息,为用户选择合适的监测工具提供了有用的参考。然而,无论是官网还是学术界,目前很难找到针对各个性能监测工具自身性能、

功耗等占用资源情况的分析和比较。

## 2 实验设计

### 2.1 工具测评维度描述

功能往往是选择监测工具时的首要考虑因素,然而当频繁监测某个系统部件时,较小的性能监测工具也可能给系统负载带来不容忽视的影响。理想的性能监测工具不仅应能满足对系统单元的全面有效监测,还应尽可能少地占用系统资源,减少对系统的性能影响,提高监测数据的精确度。因此,在使用 Linux 系统进行部分复杂作业时,工具自身占用资源情况也应在选择性能监测工具时考虑在内。

结合监测工具特点及操作系统性能指标,文中将从功能、性能、功耗三个维度对监测工具进行测评。

#### 2.1.1 功 能

监测工具的功能主要体现在监测部件及监测项上,能否持续监测、监测时间粒度、输出格式等也是要考量的方面。一般来说,通过查看官网或 man 手册及相关文献,可以对工具的功能做到比较全面的了解。

#### 2.1.2 性 能

工具的性能体现在监控特定部件时,对 CPU 及内存的占用情况。文中采用 top 工具实时采样监测进程占用 CPU 及内存资源的情况,重复多次对采样结果取平均值,获得监测工具的性能评估。

#### 2.1.3 功 耗

为衡量各个监测工具使用时的功耗特性与差异,文中选用 Chroma66204 功耗分析仪对被测系统实时功耗进行测量。设定监测工具的监测频率为每秒一次,同时功率分析仪以每 0.25 秒一次的频率采集被测系统的实时功耗。设单独执行负载程序时,特定时间内记录的功耗数据为 $P_{test\_i}, i \in [1,N]$,负载的平均功耗为 $\bar{P}_{test} = \sum_{i=1}^{N} P_{test\_i}/N$。加上监控工具,重新运行负载,记录的功耗数据为 $P_{test+tool\_i}, i \in [1,N]$,平均功耗 $\bar{P}_{test+tool} = \sum_{i=1}^{N} P_{test+tool\_i}/N$,则监控工具在该负载场景下的平均功耗为 $\bar{P}_{tool} = \bar{P}_{test+tool} - \bar{P}_{test}$,所占总功耗的比例为 $ratio = \bar{P}_{tool}/\bar{P}_{test}$,重复多次,最终对计算结果取平均值。

### 2.2 实验流程

当前 Linux 系统应用领域越来越广,其运行载荷也随之复杂多样化;同时,针对 Linux 系统的性能监测工具越来越多,无法对所有工具一一测评。因此,文中采取变量递增式的方法,针对几种常用工具的性能功耗进行测评,旨在为选择 Linux 性能监测工具提供思路。



实验使用不同监测工具,对同一部件进行监测,并在不同运行负载的条件下,对比各工具性能及功耗指标,最终通过结果分析,给出工具选择策略。实验过程可以描述为

$$\{G_1, G_2, \cdots, G_n, P\} = H(x\_load, y\_tool)$$

其中,x_load 为运行负载;y_tool 为测评对象工具;$H$ 为 2.1 节中的测试过程;$G_1 \sim G_n$ 分别表示不同部件在当前环境下的性能指标;$P$ 为当前环境下的工具功耗。

具体实验步骤如下:

Step1:相同负载条件下,针对不同的同部件监测工具测评与比较。

实验过程可以描述为:

$$\{G_1, G_2, \cdots, G_n, P\} = H(x\_load = c, y\_tool)$$

其中,$c$ 表示常量,即负载不变。

对一款工具性能进行测评与横向比较,必须限定其使用条件,在同环境下运行不同工具,能直观反映某一工具对具体使用环境的匹配程度;但同时,此方法得出的实验结论只能适用于相同或者相近的工具使用环境,在使用条件变化时,该方法得出的结论不具有通用性。

Step2:不同负载条件下,针对不同的同部件监测工具测评与比较。

为进一步验证第一步的结论,本步骤中加入了负载条件的变化,实验过程可以描述为:

$$\{G_1, G_2, \cdots, G_n, P\} = H(x\_load, y\_tool)$$

当使用环境变化时,不同工具的性能功耗也随之改变,只有在对各种使用环境和限制下的工具进行比较后,才能找到不同条件下监测工具选择的最优解。

## 3 实 验

### 3.1 实验对象与环境

#### 3.1.1 实验对象

针对 CPU、内存、磁盘、网络等关键系统部件,从子系统、全系统细粒度性能监测工具中各选取了一些典型的性能监测工具作为测评对象,包括 iostat、free、ifstat、sar 以及 collectl。

它们的主要监控命令如表 1 所示。

表 1 常用监控命令

| Unit | free | iostat | ifstat | sar | collectl |
|---|---|---|---|---|---|
| CPU | -- | iostat -c | | sar-u ALL | collectl-sc |
| Memory | free | -- | | sar-r ALL | collectl-sm |
| Disk | -- | iostat-d | | sar-b | collectl-sd |
| Network | -- | -- | ifstat-a | sar-n DEV | collectl-sn |
| ALL | -- | -- | | sar-A | collectl-ALL |

free、iostat 及 ifstat 这三个子系统监测工具的功能比较单一,只能监测一个或少数几个系统单元,sar 和 collectl 的功能相对强大,监测单元众多,监测数据丰富,涵盖了常用的分系统。和 sar 相比,collectl 还可以监测 nfs、lustre 以及内部通信数据,用于计算机集群的性能监测;此外,collectl 还提供 slabs(系统对象缓存)数据的采集。

3.1.2　实验环境

实验选用台式微机的具体软硬件环境参数如表 2 所示。

选取 Mersenne、lmbench、unixbench 等以及系统空载状态作为 CPU、内存、磁盘、系统整体的负载场景。其中,Mersenne 为计算和存储密集型程序,系统接近满负荷运行;lmbench[17] 与 unixbench 为经典的系统级 Benchmark,平均负载低于 1,空载时只有一些守护进程运行,系统负载接近 0。由于网络测试的特殊性,单独选取 netperf 和系统空载状态用于测评网络监测工具。

表 2　实验环境配置

| 硬件环境 | | |
|---|---|---|
| 处理器 | 内存 | 硬盘 |
| Intel(R) Core(TM) i5-2400 CPU 4 核 | 4 GB | WDC WD10EZEX-22B　1T |
| **软件环境** | | |
| Linux 版本 | 内核版本 | 编译器版本 |
| Ubuntu 16.04 | 4.4.0-42-generic | gcc 5.4.0 |

### 3.2　实验结果

使用子系统监测工具 iostat 监测 CPU 与磁盘,free 监测内存,ifstat 监测网络,全系统监测工具 sar、collectl 依次监测前述系统部件及系统整体。将监测同一部件的不同工具数据进行对比。

3.2.1　性能测试结果

性能测评实验的数据如表 3 所示。

表 3　监控工具占用资源数据

| 部件 | 测试激励 | free | | iostat | | ifstat | | collectl | | sar | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | CPU/% | Mem/% | CPU/% | Mem/% | CPU/% | Mem/% | CPU/% | Mem/% | CPU/% | Mem/% |
| CPU | Me1rsenne | -- | -- | 1.001 | 0.48 | -- | -- | 0.228 | 0.7 | 0.234 | 0 |
| | Lmbench | | | 0.123 | 0.21 | | | 0.050 | 0.7 | 0.196 | 0 |
| | Unixbench | | | 0.419 | 0.38 | | | 0.050 | 0.7 | 0.189 | 0 |
| | Idle | | | 0.039 | -- | | | 0.072 | 0.7 | 0.224 | 0 |
| Memory | Mersenne | 0.012 | 0 | -- | -- | -- | -- | 0.133 | 0.7 | 0.25 | 0 |
| | Lmbench | 0.005 | 0 | | | | | 0.094 | 0.7 | 0.188 | 0 |
| | Unixbench | 0.003 | 0 | | | | | 0.107 | 0.7 | 0.186 | 0 |
| | Idle | 0.006 | 0 | | | | | 0.168 | 0.7 | 0.224 | 0 |
| Disk | Mersenne | -- | -- | 0.059 | 0 | -- | -- | 0.093 | 0.7 | 0.238 | 0 |
| | Lmbench | | | 0.033 | 0 | | | 0.048 | 0.7 | 0.191 | 0 |
| | Unixbench | | | 0.033 | 0 | | | 0.044 | 0.7 | 0.195 | 0 |
| | Idle | | | 0.047 | 0 | | | 0.066 | 0.7 | 0.214 | 0 |
| Network | Netperf | | | | | 0.037 | 0 | 0.104 | 0.7 | 0.194 | 0 |
| | Idle | | | | | 0.011 | 0 | 0.121 | 0.7 | 0.186 | 0 |
| ALL | Mersenne | -- | -- | -- | -- | -- | -- | 0.666 | 0.7 | 0.983 | 0 |
| | Lmbench | | | | | | | 0.496 | 0.7 | 0.679 | 0 |
| | Unixbench | | | | | | | 0.615 | 0.7 | 0.826 | 0 |
| | Idle | | | | | | | 0.858 | 0.7 | 1.004 | 0 |

整体来看,五种监测工具使用时占用的系统资源都比较小,CPU 利用率最大为 1.004%。随着系统由满负荷运行降低到空载状态,各监测工具的 CPU 利用率均呈现了先减小后增大的趋势。这是因为 Mersenne 时单一任务运行,监测工具等待队列较短,Idle 时系统空载,监测工具优先级较高[2,18],这两种情况下性能工具占用较多的 CPU 资源,而 Lmbench、Unixbench 时多任务执行,监测工具优先级较低,等待队列较长,因此占用 CPU 资源较少。Netperf 与系统空载状态接近,工具占用 CPU 变化情况不明显。

当监测相同的部件时,子系统监测工具比 sar 和 collectl 使用特定命令监测具有更大的优势,前者的 CPU 利用率均低于 0.1%,CPU 占用率至少降低 30%。对比两个全系统细粒度的性能监测工具,相同负载下监测相同的子系统,collectl 的 CPU 利用率相较 sar 至少降低 15%。在内存占用率上,无论系统负载高低,无论被测单元种类,free、iostat 以及 sar 占用的内存资源非常小,接近 0,collectl 则稳定在 0.7%。

3.2.2　功耗测试结果

功耗测评实验的数据如表 4 所示。

表 4 监测工具平均功耗及占总功耗比例

| 部件 | 测试激励 | free | | iostat | | ifstat | | collectl | | sar | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 功耗/W | 比例 | 功耗/W | 比例 | 功耗/W | 比例 | 功耗/W | 比例 | 功耗/W | 比例 |
| CPU | Mersenne | -- | -- | 1.001 | 0.48% | -- | -- | 2.062 | 0.98% | 2.058 | 0.98% |
| | Lmbench | | | 0.123 | 0.21% | | | 0.819 | 1.40% | 0.719 | 1.23% |
| | Unixbench | | | 0.419 | 0.38% | | | 0.375 | 0.34% | 0.465 | 0.41% |
| | Idle | | | 0.039 | -- | | | 0.554 | -- | 0.270 | -- |
| Memory | Mersenne | 0.441 | 0.21% | -- | -- | -- | -- | 3.086 | 1.46% | 3.177 | 1.51% |
| | Lmbench | 0.173 | 0.30% | | | | | 1.011 | 1.73% | 0.958 | 1.63% |
| | Unixbench | 0.748 | 0.67% | | | | | 1.059 | 0.95% | 0.620 | 0.55% |
| | Idle | 0.054 | -- | | | | | 0.002 | -- | 0.029 | -- |
| Disk | Mersenne | -- | -- | 1.115 | 0.53% | -- | -- | 2.507 | 1.19% | 2.890 | 1.37% |
| | Lmbench | | | 0.086 | 0.15% | | | 1.049 | 1.79% | 0.745 | 1.27% |
| | Unixbench | | | 0.295 | 0.26% | | | 0.926 | 0.83% | 0.27 | 0.2% |
| | Idle | | | 0.024 | -- | | | 0.100 | -- | 0.030 | -- |
| Network | Netperf | | | | | 0.312 | 0.31% | 1.054 | 1.06% | 0.895 | 0.90% |
| | Idle | | | | | 0.045 | -- | 0.649 | -- | 0.054 | -- |
| ALL | Mersenne | -- | -- | -- | -- | -- | -- | 3.693 | 1.75% | 3.936 | 1.87% |
| | Lmbench | | | | | | | 1.430 | 2.44% | 1.767 | 3.02% |
| | Unixbench | | | | | | | 1.17 | 1.01% | 1.320 | 1.17% |
| | Idle | | | | | | | 2.340 | -- | 3.921 | -- |

注:因空载时系统功耗接近 0,在计算占总功耗比例时未把 idle 情况考虑入内

排除一些奇异点,总体而言,所有监测工具对系统的能耗负载较小,最大不超过 4 W,所占总功耗比例最大为 3.02%。子系统监测工具的平均功耗普遍比对应的全系统工具监测命令降低至少 30%,占总功耗比例不超过 0.67;sar 和 collectl 这两个全系统监测工具在监测特定单元时功耗差异不明显,普遍维持在 2% 以下,但当监测系统整体时,collectl 在各种系统负载情况下的耗能表现都稍优于 sar,普遍低了 0.3 ~ 1.5 W,至少减少了 7% 的能耗。

## 4 结果分析及结论

从实验结果可以看出,当监测系统分部件时,子系统监测工具的性能及功耗均优于使用相应命令的全系统监测工具。这是因为子系统监测工具框架更为简单,只需读取少数的系统文件;而全系统监测工具因为功能的多样性,底层框架更为复杂,尽管只监测单一部件,依然需要占用较多资源维护整体框架。当监测系统整体时,与 sar 相比,collectl 在性能和功耗方面具有较明显的优势。这得益于 collectl 使用了“实用报表提取语言”Perl 写成,与写成 sar 的 C 语言相比,Perl 语言集成了强大的正则表达式功能,能够很容易地操作文本、提取有用的信息、快速处理大批量数据,这在读取/proc系统、解码处理相关文件时具有很大优势。

综上,得出如下结论:

(1)作为典型的分系统监测工具,free、iostat 和 ifstat 简单实用,实现特定子系统的性能监测的同时,占用的系统资源、能耗都很小;sar 和 collectl 作为全系统细粒度监测工具,功能强大,不仅实现了丰富的系统单元监测项目,支持丰富的数据输出和表现形式,而且在不同的系统负载测试下,均展现了出色的轻量级低负载特性。

(2)实验设定四种监测工具的采集频率均为每秒一次,实际使用时根据需要调整到合适的监测频率,如每分钟一次等,此时监测工具对系统的资源、能耗负载将会更低。

(3)给出工具选择策略:如需监测特定的子系统,选择对应的分系统监测工具,会比使用特定的全系统监测命令占用更少的资源;而如果需要同时监测多个子系统或者监测系统整体,全系统细粒度监测工具将是不错的选择,首先按照具体的功能差异进行选择,当功能特性均满足时,collectl 将比 sar 更具性能及功耗上的优势。

## 5 结束语

性能监测是系统运维调优的重要一环,随着计算机硬件的不断完善、Linux 内核的不断升级,涌现出一大批功能各异、性能出色的性能监测工具。从计算机硬件到计算机软件,从 CPU 到内存到硬盘到网络,多样的性能监测工具覆盖了面向各系统单元的监测需求,且不断完善改进,向着高精度、低负载的方向发展。

文中选取了几款典型的 Linux 系统性能监测工具,设计了针对性的测评实验,从功能、性能、功耗多个维度进行了测试分析,发现不同的工具在不同使用环境下所表现出的性能功耗不尽相同。在使用时,应根

据实际情况选择满足功能需求、占用系统资源较少、消耗能源较低的合适的性能监测工具,尽量降低持续监测给系统带来的运行负载和性能影响。

目前,面向单个物理主机的性能监测工具已基本满足应用需求,亟需在面向资源共享的虚拟化环境和面向资源整合的大规模数据中心环境下,提供精准、低负载、全系统视图的性能监测工具。随着大规模数据中心的兴起,虚拟化技术盛行,如何衡量同一物理主机上每台虚拟机的实际资源使用情况,对单个虚拟机的细粒度性能事件进行实时监测,是亟待解决的问题。

**参考文献:**


[1] GREGG B. Systems performance:enterprise and the cloud[M]. Upper Saddle River:Prentice Hall Press,2013.

[2] CILIENDO E, KUNIMASA T, BRASWELL B. Linux performance and tuning guidelines[M]. [s. l. ]:[s. n. ],2007.

[3] 郭　兵,沈　艳,邵子立. 绿色计算的重定义与若干探讨[J]. 计算机学报,2009,32(12):2311-2319.

[4] 罗　旻,杨　波,高德远,等. 基于结构级的低功耗设计方法[J]. 小型微型计算机系统,2004,25(3):329-333.

[5] MOSBERGER D,JIN T. Httperf-a tool for measuring web server performance[J]. ACM SIGMETRICS Performance Evaluation Review,1998,26(3):31-37.

[6] 赵永刚,付立东. 基于 pfmon 的性能测试与分析工具 Codemon[J]. 计算机工程,2008,34(19):271-273.

[7] 温莎莎,刘　轶,刘弢宋,等. PPAT:一种 Pthread 并行程序线程性能分析工具[J]. 计算机应用与软件,2012,29(11):43-47.

[8] MINNICH R G. Supermon:high-performance monitoring for Linux clusters[C]//Linux showcase & conference. [s. l. ]:USENIX Association,2001.

[9] 闫　洁,徐恒阳,安　虹,等. Pview:一种基于 PMU 的支持并行程序性能分析的新方法[J]. 计算机科学,2011,38(2):288-292.

[10] 徐　建,张　琨,刘凤玉. 基于 Linux 的计算系统性能监控[J]. 南京理工大学学报:自然科学版,2007,31(5):622-627.

[11] BENEDICT S. Performance issues and performance analysis tools for HPC cloud applications:a survey[J]. Computing,2013,95(2):89-108.

[12] 赵付强,李允俊,宫彦磊. Proc 文件系统的研究与应用[J]. 计算机系统应用,2013,22(1):87-90.

[13] LONDON K,MOORE S, MUCCI P,et al. The PAPI cross-platform interface to hardware performance counters[C]// Department of defense users group conference proceedings. Los Alamitos,Calif. :IEEE,2001:18-21.

[14] LONDON K S,DONGARRA J,MOORE S,et al. End-user tools for application performance analysis using hardware counters[C]//International conference on parallel and distributed computing systems. Richardson, Texas, USA:[s. n. ],2001:460-465.

[15] CHATZI E G. Recent developments in hardware sensors for the on-line monitoring of polymerization reactions[J]. Journal of Macromolecular Science Part C,1999,39(1):57-134.

[16] HILL J,SZEWCZYK R,WOO A,et al. System architecture directions for networked sensors[J]. ACM SIGOPS Operating Systems Review,2000,34(5):93-104.

[17] MCVOY L, STAELIN C. Imbench:portable tools for performance analysis[C]//USENIX technical conference. [s. l. ]:USENIX Association,1996.

[18] SILBERSCHATZ A,GALVIN P B,GAGNE G,et al. Operating system concepts[M]. Reading:Addison-Wesley,1998.


(上接第 87 页)


[6] 刘林静,楼文高,冯国珍. 基于用户相似性的加权 Slope One 算法[J]. 计算机应用研究,2016,33(9):2708-2711.

[7] 张玉芳,代金龙,熊忠阳. 分步填充缓解数据稀疏性的协同过滤算法[J]. 计算机应用研究,2013,30(9):2602-2605.

[8] 郑　丹,王名扬,陈广胜. 基于 Weighted-slope One 的用户聚类推荐算法研究[J]. 计算机技术与发展,2016,26(4):51-55.

[9] 李桃迎,李　墨,李鹏辉. 基于加权 Slope one 的协同过滤个性化推荐算法[J]. 计算机应用研究,2017,34(8):2264-2268.

[10] 张　鹏,葛小青. 融合标签相似度的 k 近邻 Slope One 算法[J]. 重庆邮电大学学报:自然科学版,2016,28(4):518-524.

[11] 杜茂康,刘　苗,李韶华,等. 基于邻近项目的 Slope One 协同过滤算法[J]. 重庆邮电大学学报:自然科学版,2014,26(3):421-426.

[12] FRANCESCO R,LIOR R,BRACHA S,et al. Recommender systems handbook[M]. New York:Springer,2011.

[13] 刘建国,周　涛,汪秉宏. 个性化推荐系统的研究进展[J]. 自然科学进展,2009,19(1):1-15.

[14] 王玉斌,孟祥武,胡　勋. 一种基于信息老化的协同过滤推荐算法[J]. 电子与信息学报,2013,35(10):2391-2396.

[15] HERLOCKER J L,KONSTAN J A,TERVEEN L G,et al. Evaluating collaborative filtering recommender systems[J]. ACM Transactions on Information Systems,2004,22(1):5-53.

[16] HAN Jiawei, PEI Jian, YIN Yiwen, et al. Mining frequent patterns without candidate generation:a frequent-pattern tree approach[J]. Data Mining &Knowledge Discovery,2004,8(1):53-87.

[17] TAN Pangning, STEINBACH M, KUMAR V. 数据挖掘导论[M]. 范　明,范宏建,译. 北京:人民邮电出版社,2011:223-228.