第43卷 第8期 西 安 交 通 大 学 学 报 Vol. 43 №8
2009年8月 JOURNAL OF XI′AN JIAOTONG UNIVERSITY Aug. 2009

利用应用程序特征识别降低网格监控负载的研究

吴众欣[1]，王克[2]，钱德沛[1,2]
（1. 西安交通大学电子与信息工程学院，710049，西安；2. 北京航空航天大学计算机学院，100191，北京）

摘要：针对网络监控负载随着监控任务的增加而增加，从而影响集群有效管理、利用及监控系统性能的问题，展开了利用应用程序特征识别降低网格监控负载的研究. 采用主成分分析法分析程序执行过程中的监测数据，得到应用程序的主成分因子，再经由4种密集型基准程序构造的比对特征集合匹配，识别应用程序的主要特性，以此达到降低监控负载的目的. 在原型系统上的应用程序特征识别及监控负载评估实验表明，对于混合类型应用程序或某种密集型应用程序，监控负载可以减少20%～60%，对于巨大的采集数据，可以做到有效精炼.

关键词：网格监控；监控负载；主成分分析
中图分类号：TP393 **文献标志码**：A **文章编号**：0253-987X(2009)08-0011-06

Reducing the Grid Monitoring Workload Using Application Characteristics Identification

WU Zhongxin[1], WANG Ke[2], QIAN Depei[1,2]
(1. School of Electronics and Information Engineering, Xi′an Jiaotong University, Xi′an 710049, China;
2. School of Computer Science, Beihang University, Beijing 100191, China)

Abstract: The overhead of monitoring that increases with the increasing monitoring tasks is crucial to the effective management and efficient utilization of the cluster computers and to the performance of the monitoring system. Researches are made to reduce the overhead of monitoring by identifying the main characteristics of the application. Main factors of the application are dynamically identified by performing principal component analysis (PCA) on the fly of application execution. The set of the main characteristics is identified through matching up with the comparison characteristic set that is created by 4 categories resource intensive benchmark so that the monitoring workload is reduced. A prototype monitoring system adopting the proposed strategy is implemented. Experimental results show that the monitoring workload is decreased by 20% to 60% in hybrid applications or some resource intensive applications. Large collected data can be effectively refined and processed before inputting it to a monitoring system.

Keywords: grid monitoring; monitoring workload; principal component analysis

网格监控系统可以观测集群行为，分析采集数据，展示分析结果. 监控负载对系统的影响不可忽视，特别是当监控系统要求具有实时性或监控数据量大且节点繁忙时，会影响节点后续任务的运行[1]，因此减少监控负载是亟待解决的重要问题.

Zhang等人测定了MDS2等系统的性能，并观测到监控负载会引发系统变化[2]. Liu等人提出了网格采集数据的质量评估方法，但其未给出具体的实施步骤[3]. Yoshikazu利用差分算法，在一定容错概率下降低了监控负载[4].

一般，降低监控负载的方法有2种：①选择合适的目标变量集合，抛弃冗余信息[5]，但其要求监测的

收稿日期：2008-12-05. 作者简介：吴众欣(1974—)，男，博士生；钱德沛(联系人)，男，教授，博士生导师. 基金项目：国家高技术研究发展计划资助项目(2006AA01A118，2009AA01Z144)；科技部国际合作项目(2006DFA11080).

目标变量与运行时的应用程序特征相匹配;②减少监控数据的采集次数[6],但这种方法仅考虑了即时性,未考虑应用程序运行的整体过程变化.

本文采用主成分分析(PCA)方法来识别应用程序特征,并将其与4种密集型基准程序构建的比对特征集合进行匹配,匹配成功者,按密集型程序的目标变量采集数据,否则认为应用程序是混合型程序,应按照一定原则去除共有因子变量,得到目标变量.该方法考虑了应用程序整体运行过程的变化,也考虑了监控系统的实时性.

1 定 义

定义1 监控变量相互独立,不相关.设 n 是采集次数,p 是监控变量的个数,x_l 是监控变量,$\langle x_{l1},\cdots,x_{ln}\rangle$ 是对变量采集 n 次的数值组成的向量,则采集变量数值矩阵为

$$\boldsymbol{X}=\begin{pmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix}=(x_1,\cdots,x_l,\cdots,x_p)' \tag{1}$$

定义2 应用程序特征的主成分因子集合定义为 $\boldsymbol{F}=(F_1,..,F_j,\cdots,F_m)'(m<p)$,其中 F_j 是第 j 个主成分因子.多个 m_i 属于同一个 F_j.

定义3 应用程序特征由主成分因子体现,主成分因子由因子载荷获得,不能丢失主要信息.载荷大的主成分因子代表信息重要,其包含的变量必须监测,不可消减.

定义4 应用程序间具有相似性.Aashish 在 SPEC CPU2000 Benchmark 套装中选出8个程序来代替整个套装中的程序[7].

2 降低监控负载

依据上述定义,对采集到的数据矩阵 $\boldsymbol{X}$ 采用PCA方法降维,得到主成分因子集合 $\boldsymbol{F}$,再用主成分因子中的目标变量替代原始监控变量(未过滤相关变量的原始监控变量集合),期间不可丢弃有用信息.

2.1 监控负载 PCA 分析

对于应用程序,用 $\boldsymbol{X}=(X_1,..,X_l,\cdots,X_p)'$ 表示采集的数值矩阵,$E(\boldsymbol{X})=\boldsymbol{\mu},D(\boldsymbol{X})=\boldsymbol{\Sigma}$.设不可观测的随机向量为 $\boldsymbol{F}=(F_1,..,F_j,\cdots,F_m)'(m<p)$,其中 $E(\boldsymbol{F})=0,D(\boldsymbol{F})=\boldsymbol{I}_m$;又设 $\boldsymbol{\varepsilon}=(\varepsilon_1,..,\varepsilon_p)'$ 与 $\boldsymbol{F}$ 互不相关,且 $E(\boldsymbol{\varepsilon})=0,D(\boldsymbol{\varepsilon})=\boldsymbol{D}$(对角矩阵).则,$\boldsymbol{X}$ 满足以下模型

$$\left.\begin{aligned} X_1-\mu_1&=a_{11}F_1+a_{12}F_2+\cdots+a_{1m}F_m+\varepsilon_1\\ &\vdots\\ X_l-\mu_l&=a_{l1}F_1+a_{l2}F_2+\cdots+a_{lm}F_m+\varepsilon_l\\ &\vdots\\ X_p-\mu_p&=a_{p1}F_1+a_{p2}F_2+\cdots+a_{pm}F_m+\varepsilon_p \end{aligned}\right\} \tag{2}$$

可以称这个模型为正交因子模型,用矩阵表示为

$$\boldsymbol{X}=\mu+\boldsymbol{A}\boldsymbol{F}+\boldsymbol{\varepsilon} \tag{3}$$

式中:$\boldsymbol{F}=(F_1,\cdots,F_j,\cdots,F_m)'(m<p)$,$F_j$ 称为应用程序的主成分因子;$\boldsymbol{\varepsilon}=(\varepsilon_1,\cdots,\varepsilon_p)'$,$\varepsilon_1,\cdots,\varepsilon_p$ 称为 $\boldsymbol{X}$ 的特殊因子.主成分因子对 $\boldsymbol{X}$ 的每一个分量都起作用,而 $\boldsymbol{\varepsilon}$ 仅仅对 X_i 起作用.

矩阵 $\boldsymbol{A}=(a_{ij})_{p\times m}$ 是待估系数矩阵,也称为因子载荷矩阵,其中 $a_{ij}(i=1,\cdots,p;j=1,\cdots,m)$ 称为第 i 个变量的第 j 个因子载荷.

因为主成分因子彼此正交、不相关并具有单位方差,即 $D(\boldsymbol{F})=\boldsymbol{I}_m$,则由 $\boldsymbol{\Sigma}=D(\boldsymbol{X})=\boldsymbol{A}\boldsymbol{A}'+\boldsymbol{D}$ 可导出

$$\boldsymbol{\Sigma}-\boldsymbol{D}=\boldsymbol{A}\boldsymbol{A}' \tag{4}$$

因子分析首先是由样本协方差阵 $\hat{\boldsymbol{\Sigma}}$ 来估计 $\boldsymbol{\Sigma}$,然后分解式(4)求得 $\boldsymbol{A}$ 和 $\boldsymbol{D}$.也就是,从观测变量 $X_1,\cdots,X_p$ 给出的样本数据中,求出载荷矩阵 $\boldsymbol{A}$,然后预测主成分因子 F_j.$\mathrm{cov}(\boldsymbol{X},\boldsymbol{F})=\boldsymbol{A}$,其中 $\boldsymbol{A}$ 为 $p\times m$ 矩阵,$\boldsymbol{A}$ 中元素 a_{ij} 能够刻画变量 X_i 与 F_j 之间的相关性,而 X_i 是 F_j 上的因子载荷.

如果样本的协方差阵 $\hat{\boldsymbol{\Sigma}}$ 的特征值为

$$\lambda_1\geqslant\lambda_2\geqslant\cdots\geqslant\lambda_p\geqslant 0 \tag{5}$$

相应地,单位正交特征向量为 $\boldsymbol{l}_1,\boldsymbol{l}_2,\cdots,\boldsymbol{l}_p$,则 $\hat{\boldsymbol{\Sigma}}$ 的谱分解公式为

$$\hat{\boldsymbol{\Sigma}}=\sum_{i=1}^{p}\lambda_i\boldsymbol{l}_i\boldsymbol{l}_i'$$

它可以近似地分解为

$$\hat{\boldsymbol{\Sigma}}\approx\boldsymbol{A}\boldsymbol{A}'+\boldsymbol{D}$$

式中

$$\boldsymbol{A}=(\lambda_1^{1/2}\boldsymbol{l}_1,\cdots,\lambda_m^{1/2}\boldsymbol{l}_m)\overset{\mathrm{def}}{=}(a_{ij})_{p\times m} \tag{6}$$

由式(6)得到的 $\boldsymbol{A}$ 和 $\boldsymbol{D}$ 是因子模型的主成分解.主成分因子数由下式确定,即

$$\frac{\lambda_1+\cdots+\lambda_m}{\lambda_1+\cdots+\lambda_m+\cdots+\lambda_p}\geqslant P_0 \tag{7}$$

在本文实验中,$0.75\leqslant P_0<1$.

2.2 用突出主成分因子表示变量集合

采用正交旋转最大变异法来突出主成分因子.

若 $\boldsymbol{X}=\boldsymbol{AF}+\boldsymbol{\varepsilon}$，$\boldsymbol{F}=(F_1,\cdots,F_m)'$ 为主成分因子向量，令 $\boldsymbol{Z}=\boldsymbol{\Gamma}'\boldsymbol{F}$（$\boldsymbol{\Gamma}$ 为任意 m 正交矩阵），对 $\boldsymbol{F}$ 实施正交变换，则

$$\boldsymbol{X}=\boldsymbol{A\Gamma Z}+\boldsymbol{\varepsilon} \tag{8}$$

式中：$\boldsymbol{A\Gamma}$ 是主成分因子 $\boldsymbol{Z}$ 的因子载荷矩阵. 对初始 $\boldsymbol{A}$ 反复右乘正交矩阵 $\boldsymbol{\Gamma}$，可突出主成分因子.

2.3　主成分因子中的共有因子与特有因子

应用程序间具有共性，也有特性，因此它们的主成分因子会包含共有、特有 2 部分.

若有 k 种应用程序，则其采样变量矩阵与主成分因子的关系为

$$\left.\begin{aligned}
&Y_1\Rightarrow F_{11}+F_{12}+\cdots+F_{1a}\\
&\vdots\\
&Y_j\Rightarrow F_{j1}+F_{j2}+\cdots+F_{jd}\\
&\vdots\\
&Y_k\Rightarrow F_{k1}+F_{k2}+\cdots+F_{kg}
\end{aligned}\right\}$$

变量均相同的因子称为共有因子. 去除共有因子，Y_i 剩余的主成分因子就是特有因子. 获得特有因子的伪代码如图 1 所示.

```
//遍历种程序
for (i=1; i≤k-1; i++) {
    for (j=i+1; j≤k; j++) {
    //主成分因子对比
            for (i=1; x≤size0f (Yi); x++) {
                    for (y=1; y≤size0f (Yj);) {
                            if (Yi=Yj)
                            {add Yi to CommonFSet }
                    }
            }
    }
}
```

图 1　获得特有因子的伪代码

2.4　构建监控目标变量集合

构建监控目标变量集合需要遵从以下原则.

（1）保留主要信息. 根据定义 3 且由式（5）、式（7）可知，主成分因子序号在前的贡献大. 贡献大的主成分因子即使是共有因子也要保留.

（2）去除冗余信息. 在贡献小的主成分因子中，属于共有因子的变量要去除.

确定监控目标变量集合大致经以下 2 个阶段.

（1）构造比对特征集合（因子目标变量表）. 运行于集群上的信息传递接口（MPI）并行计算应用程序可以分为计算密集型、访存密集型、通讯密集型、I/O 密集型、混合类型. 依照上述的构建原则，可选择 4 种密集型的基准程序来构建比对特征集合.

（2）识别应用程序特征. 经过主成分分析得到因子表，并与比对特征集合进行匹配. 匹配成功，则按照该密集型的目标变量进行负载消减；否则，程序是一种混合类型的程序，应采用上述构建原则消减共有因子变量，降低监控负载.

3　实　验

本文构建了 SmartMonitor 监控系统原型，以评估监控负载减小的程度.

3.1　SmartMonitor 系统与实验环境

SmartMonitor 系统架构如图 2 所示，其中心监控系统管理节点资源，包括分配、调度、监测子系统，子系统汇集监测信息并向中心系统传送. 系统使用 Ganglia 作为节点传感器.

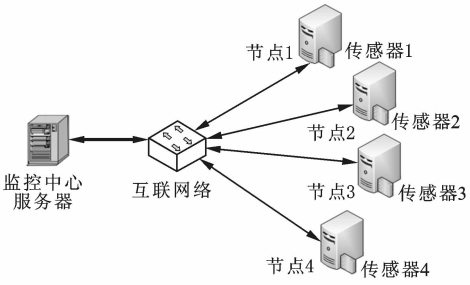


图 2　SmartMonitor 原型系统架构

节点的监控数据流向如图 3 所示. Ganglia 会采集多个变量（超过 15 个）的原始数据，再由过滤器滤掉不相关的变量数据. 批量处理模块制定采集时间窗口长度. 采集的数据先进入核心处理模块，后经 PCA 模块分析出主成分因子集合，根据程序特征，在共有、特有因子匹配模块中识别出应用程序特征，最后在负载消减模块执行去冗余操作，得到目标变量数据集合.

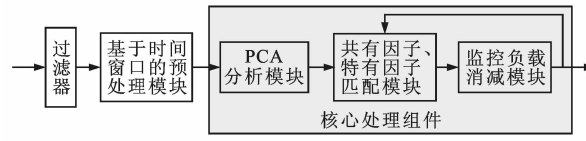


图 3　数据流向图

实验环境：每个节点（包括中心服务器）均配置 4 个 2.4 GHz 的 CPU、2 GB 内存、80 GB SCSI 硬盘，节点间以 1 000 MB 带宽的以太网互联，节点的软件环境为 Linux 操作系统、MPICH 执行框架.

3.2　监控变量

按照定义 1，由过滤器可以得到相互独立的变量，如表 1 所示.

表 1 相互独立的监控变量

变量名	含义	类型	变量名	含义	类型
bytes_in	每秒接收的字节数	Float (32 b)	load_one	1 min 负载平均值	Float (32 b)
bytes_out	每秒发送的字节数	Float (32 b)	mem_buffers	Buffer 层面的内存大小	Uint32 (32 b)
cpu_aidle	用户登录后 CPU 空闲率	Float (32 b)	mem_cached	Cache 层面的内存大小	Uint32 (32 b)
cpu_idle	CPU 空闲率	Float (32 b)	mem_free	可用内存大小	Uint32 (32 b)
cpu_system	CPU 系统占用率	Float (32 b)	pkts_in	每秒接收的数据包个数	Float (32 b)
cpu_user	CPU 用户占用率	Float (32 b)	pkts_out	每秒发送的数据包个数	Float (32 b)
disk_free	硬盘空闲空间大小	Double (64 b)	proc_run	正在运行的进程数目	Uint32 (32 b)
load_fifteen	15 min 负载平均值	Float (32 b)	proc_total	进程总数	Uint32 (32 b)
load_five	5 min 负载平均值	Float (32 b)	swap_free	交换分区内存大小	Uint32 (32 b)

3.3 选择比对基准程序

选择 4 种类型的基准程序作为比对程序.

计算密集型基准程序:选择密集并行(EP)基准程序,该程序主要测试数学函数的浮点运算性能,对通信不敏感.

通讯密集型基准程序:选择 NAS 并行基准程序包中的 BT 基准程序,该程序可求解三对角方程组,并以点到点的长消息通讯为主.

I/O 密集型基准程序:选择 b_eff_io 程序,该程序可以测试多访问模式下并行 MPI-I/O 应用程序的 I/O 带宽特性.

访存密集型基准程序:选择 Stream 来测量存储器持续带宽.

3.4 构造比对因子表

对比对基准程序的监测数据进行主成分分析,并用正交旋转最大变异法突出主因子,由此可得 4 种密集型应用程序的主成分因子,如表 2 所示.

变量相同的因子称为共有因子. 例如,计算密集型的程序要求节点间传送消息,所以其因子集合的一部分也会出现在通讯密集型应用程序的因子集合中. 整理得出的共有因子表如表 3 所示.

表 2 4 种密集型应用程序主成分因子

程序	变量				
	F_1	F_2	F_3	F_4	F_5
NPB-EP	cpu_aidle cpu_idle disk_free mem_buffers	bytes_in bytes_out pkts_in pkts_out load_fifteen load_five load_one	mem_free proc_total	swap_free	
Eff_I/O	bytes_in bytes_out pkts_in pkts_out	cpu_aidle disk_free mem_buffers mem_cached	load_fifteen load_five load_one	cpu_system cpu_user	proc_run proc_total
STREAM	cpu_aidle cpu_idle cpu_user proc_run load_fifteen load_five load_one	bytes_in bytes_out pkts_in pkts_out proc_total swap_free mem_buffers	mem_free		
NPB-BT	bytes_in bytes_out pkts_in pkts_out	mem_cached mem_free proc_total	cpu_aidle disk_free	load_fifteen load_fifteen load_five	cpu_system cpu_user

表 3 共有因子表

共有因子	F_2	F_3	F_4
变量	bytes_in bytes_out pkts_in pkts_out	load_fifteen load_five load_one	cpu_system cpu_user

根据 3.4 节的构建原则,如果应用程序的第一主成分因子(F_1)是共有因子,则仍需保留这个因子,因为根据式(5)、式(7),该主成分因子贡献最大,最能体现程序的特征. 将各类程序中不同的主成分因子分组称为特有因子. 若因子变量中除了共有因子变量外还有其他变量,则可认为这种因子不是共有因子. 修订得出应用程序的比对特征集合如表 4 所示.

表 4　比对特征集合

程序	变量		
	F_1	F_2	F_5
NPB－EP	cpu_aidle cpu_idle disk_free mem_buffers	mem_free proc_total	swap_free
Eff_I/O	bytes_in bytes_out pkts_in pkts_out	cpu_aidle disk_free mem_buffers mem_cached	proc_run proc_total
STREAM	cpu_aidle cpu_idle cpu_user proc_run load_fifteen load_five load_one	bytes_in bytes_out pkts_in pkts_out proc_total swap_free mem_buffers	mem_free
NPB-BT	bytes_in bytes_out pkts_in pkts_out	mem_cached mem_free proc_total	cpu_aidle disk_free

3.5　识别应用程序特征构建监测目标变量集合

采集应用程序执行期的监测数据进行主成分分析，用正交旋转最大变异法突出主成分因子，去除共有因子，得到应用程序的特征因子表. 将该表与表 4 比对，若与密集类型的程序特征因子相匹配，可按该类型的目标变量采集监测数据，否则认为该程序是一种混合类型程序（各种特征都不明显，或是某些特征的组合，多个特征的程序混合执行也会出现这种情况），应按照去除（除 F_1 之外）共有因子后的因子变量采集监控数据. 测定某一利用高斯消元法求解稠密矩阵的类 Linpack 程序（它是计算密集型程序与访存密集型程序的混合运行体，在这里称其为某混合型应用程序）的主成分因子如表 5 所示，经过消减，需监测的目标变量如表 6 所示. 对比表 1、表 5 变量可发现该程序的特征，其监控负载大为减小.

表 5　混合型应用程序的主成分因子及其变量

程序	变量					
	F_1	F_2	F_3	F_4	F_5	F_6
混合型应用程序	bytes_in bytes_out pkts_in pkts_out	disk_free mem_cached mem_free	load_fifteen load_five load_one	cpu_aidle proc_total	mem_buffers swap_free	cpu_idle cpu_user

表 6　表 5 经去除共有因子后的监控目标变量

程序	变量				
	F_1	F_2	F_4	F_5	F_6
混合型应用程序	bytes_in bytes_out pkts_in pkts_out	disk_free mem_cached mem_free	cpu_aidle proc_total	mem_buffers swap_free	cpu_idle cpu_user

3.6　监控负载减小率

监控目标变量是将原始采集变量的数量从 p' 减少到 m'，用 $L()$ 表示变量的数据长度（具体数值见表 1）或数据包的个数（不同的监控粒度）估算的监控负载减小率

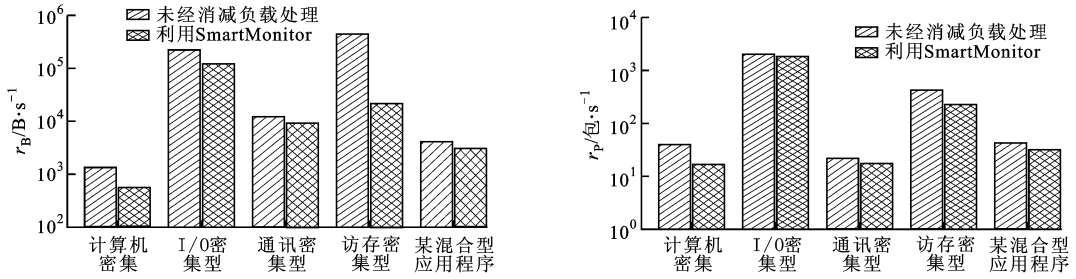
$$u=\frac{1}{n}\sum_{n}\frac{\sum L(p')-\sum L(m')}{\sum L(p')} \tag{10}$$

式中：n 是实验次数. 在实验中，定义监控负载为监测行为带来的每秒进出系统的字节数或每秒进出系统的数据包个数. 当 $n=10$ 时，平均负载减小的百分比如表 7 和图 4 所示. 从中看出，访存密集型程序可以减小 20%左右的负载，而计算密集型、通讯密集型或 I/O 密集型应用程序可以减小 45%～60%的负载. 对于混合类型应用程序，以 3.5 节的解线性方程为例，当 $n=10$ 时，监控负载减小了近 26%. 可见，本文方法非常有效.

表 7　监控负载减小率比较

程序类型	r_B/B·s^{-1}	r_P/包·s^{-1}	r_B/B·s^{-1}	r_P/包·s^{-1}	负载减小率/%
	未经消减负载处理		利用 SmartMonitor		
计算密集型	1 365.51	40.65	573.51	16.85	59
I/O 密集型	221 626.50	2 658.65	12 1597.37	1 827.88	45
访存密集型	12 263.58	21.83	9 265.59	17.43	22
通讯密集型	430 183.19	425.27	21 393.42	225. 39	48
某混合型应用程序	4 197.30	43.11	3 144.72	31.82	26

注：r_B 表示字节接收率；r_P 表示数据包接收率.



(a)字节 (b)数据包

r_B:字节接收率;r_P 数据包接收率

图4 负载减小结果

4 结束语

本文采用应用程序特征识别法来消减监控负载.今后的工作是对作业的执行过程按照监控信息进行合理调度,提高资源利用率和多作业并行执行的能力.

参考文献:

[1] 王寅峰,董小社,何秀强,等.网格测试引擎:一种构建网格测试环境的基础架构[J].西安交通大学学报,2007,41(8):884-888.
WANG Yinfeng, DONG Xiaoshe, HE Xiuqiang, et al. Grid testing engine: an infrastructure for constructing grid testing environment [J]. Journal of Xi'an Jiaotong University, 2007, 41(8), 884-888.

[2] ZHANG Xuehai, FRESCHL J L, SCHOPF J M. A performance study of monitoring and information services for distributed systems [C]// 12th IEEE International Symposium on High Performance Distributed Computing. Los Alamitos, CA, USA: IEEE Computer Society, 2003:270.

[3] LIU Wei, LUO Tiejian, SONG Jinliang, et al. Improving grid monitoring with data quality assessment [C]// International Symposium on Communications and Information Technologies. Los Alamitos, CA, USA: IEEE Computer Society, 2007:1534-1539.

[4] YOSHIKAZU K, KENJIRO T. Scalable data gathering for real-time monitoring systems on distributed computing [C]// 8th IEEE International Symposium on Cluster Computing and the Grid. Piscataway, NJ, USA: IEEE, 2008: 425-432.

[5] YANG Lingyun, SCHOPF J M, DUMITRESCU C L, et al. Statistical data reduction for efficient application performance monitoring [C]// Proceedings of the Grid Workshop. Los Alamitos, CA, USA: IEEE Computer Society, 2006:327-334.

[6] DUESTERWALD E, CASCAVAL C, DWARKADAS S. Characterizing and predicting program behavior and its variability [C] // Proceedings of the 12th International Conference on Parallel Architectures and Compilation Techniques. Los Alamitos, CA, USA: IEEE Computer Society, 2003:220.

[7] PHANSALKAR A, JOSHI A, EECKHOUT L, et al. Measuring program similarity: experiments with SPEC CPU benchmark suites [C]// IEEE International Symposium on Performance Analysis of Systems and Software. Los Alamitos, CA, USA: IEEE Computer Society, 2005:10-20.

(编辑 苗凌)