基于虚拟化计算集群技术的 IPTAS分析子系统

王恒波[1]，丁伟[1,2]，夏震[2]

（1. 东南大学计算机科学与工程学院，南京， 211189；
2. 中国教育和科研计算机网华东（北）地区网络中心，南京， 210096）

摘 要：为解决被动网络测量中，一般个人或科研机构不具备面向海量测量数据的采集、存储、管理和计算能力，以及测量结果和分析算法难以共享的问题， CERNET 华东（北）地区网络中心自主开发了一个互联网流量分析平台 IPTAS 。 IPTAS 分析子系统是 IPTAS 平台的重要组成部分，支持可信任用户以提交分析任务的方式使用 IPTAS 平台的数据资源和计算资源，进行基于被动测量的研究。本文首先介绍了网络测量与 IPTAS 平台的联系，分析了 IPTAS 分析子系统目前存在的不足。随后提出一种基于虚拟化计算集群的改进方案，并给出了改进系统的框架结构和控制流程最后，对系统改进中关键机制的设计与实现，进行了详细的阐述。

关键词：被动测量； IPTAS ；虚拟化集群；任务调度；集群监管

IPTAS Analysis Sub-system based on Virtualization and Clustering

Wang Heng Bo[1], Ding Wei[1,2], Xia Zhen[2]

(1. School of Computer Science & Engineering, Southeast University, Nanjing, 211189;
2. CERNET Eastern China (North) Regional Network Center, 210096)

Abstract: To solve the problems in network passive measurement that general researchers or institutions are usually lack of the capability of collection、 storage 、 management and computing oriented to massive measurement data, and measurement results and analysis algorithms are difficult to share with others, CERNET Eastern China (North) Regional Network Center developed an IP Trace Analysis System via IPTAS. IPTAS analysis sub-system is an important part of IPTAS, which supports the trusted users carrying out their passive measurement studies using the data and computing resources in IPTAS in the form of submitting analysis task. Firstly, the paper introduces the association between network measurement and IPTAS, and makes a brief analysis about the shortcoming of existing IPTAS analysis sub-system. Then an improvement program based on virtualized computing cluster is proposed, containing the improvement framework and control process. Finally, the paper elaborates the design and implement of some key schemes involved in the system improvements.

Key words: Passive Measurement; IPTAS; Virtualized Cluster; Task Scheduling; Cluster Monitoring and Management

网络测量对于理解网络行为，规划网络发展具有重要意义。随着互联网流量数据的不断增长，网络测量，尤其是全流量被动测量开始遭遇如何有效管理测量数据的难题。国际上一些测量机构针对测
量数据的管理问题进行了相关研究，主要侧重于测量数据的跟踪、反馈和可用性的提高，如可扩展网
络测量知识库(SIMR)[1]、1ST-MOME[2]数据库、CAIDA 网络测最数据目录(IMDC)[3]等。这些系统除了提供关于测量数据、测量工具和分析结果的检索和下载服务外，对数据和分析算法的可重用性、存储和计算资源的可共享性考虑的很少。

互联网流量分析平台 IPTAS (IP Trace Analysis System)[4]，是CERNET华东（北）地区网络中心为解决被动测量中数据采集、存储和计算等问题而自主研发的一个以 IP Trace 数据为核心的数据管理与分析系统，为基于被动测量的网络行为学研究提供实测流量数据和共享计算平台。IPTAS 系统目前由四个子系统组成：发布子系统、

作者简介：王恒波，（1988-），男，硕士研究生， E-mail: hengbowang@njnet.edu.cn ；丁伟，（ 1962-），女，教授，博导， E-mail: wding@njnet.edu.cn；夏震，（ 1976- ），男，硕士研究生， E-mail: zhxia@njnet.edu.cn.

分析子系统、测试子系统和数据管理子系统。其中，分析子系统[5]支持可信任用户以提交数据分析任务的形式使用IPTAS的数据资源和计算资源，进行面向海量 IP Trace 数据的统计分析工作，并共享分析代码与分析结果。

在实际运行中，IPTAS分析子系统的计算资源由单台高性能服务器提供，在大量分析任务集中提交时，存在任务等待时间长，系统任务吞吐率低的问题。IPTAS系统基于区域存储网络(IP-SAN)[6]的数据存储环境允许多机并发访问数据，使多机多任务并行成为可能。因此，为满足更多用户使用系统以及更大规模数据分析的需求，有必要对分析子系统现有的计算资源进行扩展，提高系统的服务能力。

1 改进方案

随着硬件成本的降低以及多核技术的发展，计算资源的结构不断趋于复杂，性能不断趋于强大，在这种背景下，采用单一操作系统对系统进行管理，将带来应用之间隔离性的问题，并可能导致计算资源的浪费和不合理分配。虚拟化技术[7]能够将单一系统按资源需求分割成多个相互隔离的系统，容纳更多的应用和用户操作环境。在提高系统资源利用率的同时，也降低了购置计算资源的资金投入。

针对现有IPTAS分析子系统计算能力不足的缺点，并考虑到目前网络中心内部存在一些未被充分利用的高性能服务器的实际，本文设计使用虚拟化的计算集群对系统进行结构和功能上的扩展，以集群环境下的多任务并行替代现有单机环境下的任务串行或低并行度的并行。改进后的分析子系统以主从模式(Mater-Slave)工作，支持用户分析任务在计算集群内合理调度与分配，以及对任务的跟踪和监控，并能够对计算集群进行有效地监管。改进后的系统由四个部分组成：

- 管理子系统：以Web页面的形式向用户提供的功能操作接口，并实现用户操作的后台处理与响应模块，包括测量数据检索与定制、分析代码上载、数据分析任务提交等核心功能。
- 执行子系统：实现对数据分析任务自用户提交到向用户返回结果整个生命周期的管理，包括任务调度、加载执行、运行时监控和结果回收。
- 监控子系统：以节点资源表的形式对计算集群内的所有节点的资源和状态信息进行存储和维护，为执行子系统的集群任务调度提供必要的依据。并提供可视化的集群节点性能监测功能。
- 虚拟化计算集群：由多台高性能服务器使用XEN[8]或KVM[9]虚拟化平台，虚拟出若干虚拟机通过千兆以太网互联而成。每台虚拟机通过IP-SAN访问IP Trace数据，向分析子系统提供计算服务，称为一个计算节点。
- IP-SAN存储：由支持iSCSI协议的DELL网络磁盘阵列和千兆以太网交换机互联而成，是IP Trace数据的实际存储环境。

图1和图2分别展示了改进后IPTAS分析子系统的逻辑结构和物理结构。逻辑上，三个子系统通过共享数据区交互。其中，管理子系统与执行子系统共享记录了用户任务信息的系统任务表，执行子系统雨监控子系统共享记录了节点信息的节点资源表；物理上，管理子系统部署在总控节点，由于改进系统分布式的特点，执行子系统和监控子系统都分布部署在总控节点和计算节点，以*-M和*-S分别表示总控节点和计算节点的部分，两部分间通过XML-RPC建立通信通道。

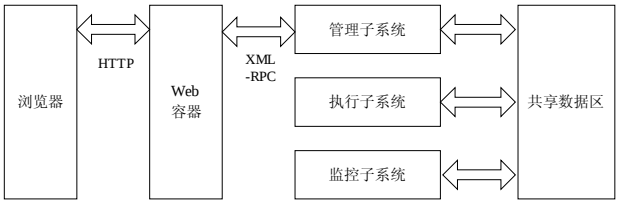

图 1 系统逻辑结构

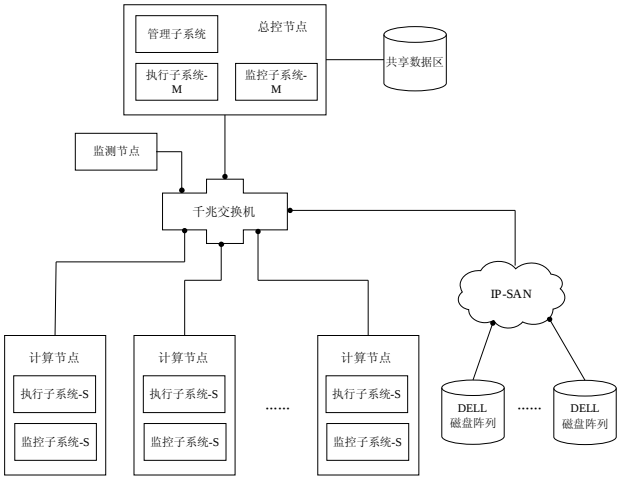

图 2 系统物理结构

2 控制流程

管理子系统是响应用户页面操作的触发式系统，而监控子系统负责节点信息维护，在时间域上是一个周期性触发的系统，控制流程都相对简单。限于篇幅，在此着重介绍执行子系统的控制流程，如图 3 所示。

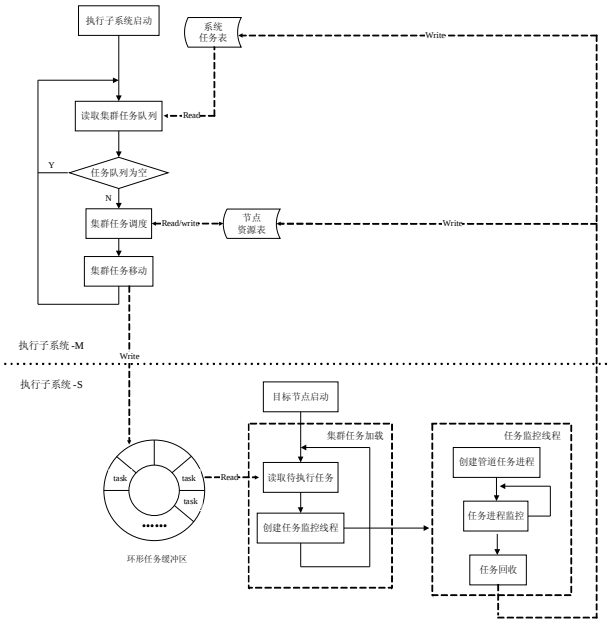

图 3 执行子系统系统控制流程

执行子系统的主要控制流程简述如下：

(1) 以合理的策略调度系统任务表中的所有等待任务，根据节点资源表为每个任务分配适合的目标节点。

(2) 将已完成调度的任务相关数据文件和控制信息发送到目标节点的环形任务缓冲区中。

(3)
目标节点从任务缓冲区中读取任务，并加载执行。

(4)
监控任务进程的运行状态。

(5)
捕获任务终止状态，返回相应的任务完成信息。

其中，过程 (1) 合 (2) 由执行子系统 -M 的相应模块完成，过程 (3) 、 (4) 和 (5) 由执行子系统-S的相应模块完成。

3 系统关键技术

3.1 集群环境下多任务调度

集群环境下，如何规划一组到达的、相互独立的任务的调度顺序，并为任务分配合适的计算节点，从而实现降低任务平均等待时间，提高系统的任务吞吐率 (Throughput) 等优化目标是系统设计中的关键问题。联系 IPTAS 分析子系统的实际，任务调度的具体目标如下：

目标 1 ： 系统任务吞吐率较大

目标 2 ： 任务平均平均等待时间较小

目标 3 ： 满足分析任务对系统资源的需求

目标 4 ： 保证多用户使用系统资源的公平性

不同的分析任务对系统资源的需求量和占用时间可能存在较大差异，不同计算节点在服务能力上也不尽相同。因此，区分分析任务的粒度和评估计算节点的服务能力，是制定合适策略与算法的前提。

3.1.1任务粒度划分

数据分析任务由分析算法和测量数据集组成。分析算法是针对测量数据的抽样、统计、建模等功能性算法，如报文抽样、流特征提取、报文测度等。系统目前提供 NetFlow 流数据和 IP Trace 原始报文数据两类测量数据。可以认为，分析算法的计算复杂性越高，则需要的系统资源越多；输入数据量越大，则占用系统资源的时间越长。对分析任务粒度的划分，需要从这两方面考虑。

一般认为，分析算法存在稳定的资源需求量，不会随输入规模的增长而无限增长。因此，要求用户在提交分析任务时，显式地声明本次任务需要的资源需求向量 δ，如CPU核心数、最大内存

容量、最大磁盘空间等。相应地，系统设定一个资源需求阈值向量 δ_0，根据分析任务的资源需求向量是否超过阈值向量，将任务划分为两种资源需求粒度：大任务和小任务。

系统提供的测量数据集按时间进行组织，如持续时间分别为1h、2h和 24h 的 IP Trace 数据集。事实上，用户通常对所选数据集中某个时间跨度内的数据进行分析。因此，要求用户提交任务时声明大致使用的数据量 K。根据 K 的取值，将任务分为三种时间粒度：短任务、中长任务和长任务。

综上所述，任务的粒度划分如表 1 所示，其中，δ_0， K_1 ， K_2 为经验值，可根据系统运行状况调整。

表 1. 分析任务的粒度划分

粒度	资源需求	数据量级(GB)
小任务	$\delta < \delta_0$	*
大任务	$\delta >= \delta_0$	*
短任务	*	$0 < K <= K_1$
中长任务	*	$K_1 < K <= K_2$
长任务	*	$K > K_2$

3.1.2计算节点服务能力评价

从计算节点资源拥有量的角度来描述节点的服务能力，认为资源拥有量越多的节点，能同时容纳的分析任务越多，服务能力越强。

定义节点能力权值为 W_μ ，反映在无任务负载的情况下节点资源拥有量的差异。由于节点所处网络环境的一致性，只考虑CPU资源和内存资源，按式 (1)(2)(3) 计算。式中， W_{cpu} 、 W_{ram} 分别表示节点的 CPU 能力权值和内存能力权值，λ_1 、 λ_2 表示 W_{cpu} 、 W_{ram} 对 W_μ 的影响程度。 n 表示集群内节点的总数量， Speed(i) 、 Vcpu(i) 、 RAM(i) 分别表示节点 i 的CPU运算速率、虚拟CPU数量和内存容量。

$$W_\mu(i) = \sqrt{\lambda_1 * W_{cpu}(i)^2 + \lambda_2 * W_{mem}(i)^2} \qquad (1)$$

$$W_{\text{cpu}}(i) = \frac{Speed(i) * Vcpu(i)}{\max_{k=1}^{n} Speed(k) * Vcpu(k)} \qquad (2)$$

$$W_{\text{ram}}(i) = \frac{RAM(i)}{\max_{k=1}^{n} RAM(k)} \qquad (3)$$

定义节点资源容余向量 ξ 为节点各项资源当前的剩余量，与分析任务的资源需求向量 δ 相对应，体现节点当前对分析任务的容纳能力。节点可接受一个任务，当且仅当 ξ>=δ 。分析子系统为每个节点 i 维护 ξ(i) ，并按 (4) 式随着节点完成任务 t 和任务 t 到达节点而动态地增减。

$$\zeta(i) \leftarrow \zeta(i) \pm \delta(t) \qquad (4)$$

3.1.3多任务调度策略与算法

集群环境下的多调度策略由任务调度策略和资源匹配策略组成：调度策略决定任务调度的时序，匹配策略决定为任务分配的计算节点。

为满足目标 1， 2 和 4，拟定以下调度策略：

- 时间粒度较小的短任务优先调度
- 长时间等待的“饥饿”任务优先调度
- 正在运行任务数较少的用户的后续任务优先调度

为满足目标 2 和 3，应尽可能快速地定位能够满足任务资源需求的节点，并为大任务“预留”足够的资源，避免被小任务抢占。资源匹配策略采用一种基于能力权值排序的首次适应 (First Fit) 策略：

- 所有计算节点按照能力权值 W_μ 从低到高的顺序排序，构成资源队列
- 小任务从队头开始，向后搜索第一个满足 ξ>=δ 的节点，大任务从队尾向队头，向前搜索第一个满足 ξ>=δ 的节点

综合上述的调度策略和资源匹配策略，设计基于调度轮次和动态任务优先级的调度算法。

系统周期性地从信息数据库取出到达系统的所有任务，构成等待任务队列，对这一组等待任务的调度称为一轮(round)。每轮按照任务优先级大小的顺序调度每个任务，并根据资源匹配的策略为任务选择合适的计算节点。任务的优先级定义如下：

$$pri(t) = w_1 * pri\_static(t) + w_2 * pri\_user(t) + w_3 * pri\_round(t) \qquad (5)$$

$$pri\_user(t)=\begin{cases} penalty, & if\ L_u \geq L_0 \\ 0, & other \end{cases} \quad (6)$$

$$pri\_round(t) \leftarrow pri\_round(t)+\Delta r \quad (7)$$

式 (5) 中， pri(t) 表示分析任务 t 的优先级。pri_static(t) 表示由 t 的时间粒度所决定的静态优先级，时间粒度较小的任务具有较高的静态优先级。 pri_user(t) 表示由任务 t 所属用户 u 的用户优先级，每轮调度开始时按式 (6) 更新，L_u，L_0分别为用户 u 在每轮调度开始时正在运行的任务数和系统设定的阈值，L_u随用户 u 的任务的运行和完成而动态更新， penalty<0 ，为用户优先级惩罚值。pri_round(t)表示轮次优先级，反应任务 t 的“饥饿“程度，因本轮未能完成资源匹配而进入下轮的任务，按式 (7) 更新轮次优先级， Δr>0 ，为轮次优先级增量。w_1 、 w_2 、 w_3 分别表示三个优先级因子的影响程度。

对第 i 轮调度过程描述如下：

1. 系统进入第 i 轮调度，读取所有到达的任务到任务队列；
2. 按式 (5)(6)(7)计算每个任务在本轮的优先级；
3. 按任务优先级从大到小的顺序调度任务；
4. 对当前任务 t ，按资源匹配策略分配计算节点。若成功， t 所属用户的 L_u 自增 1；若不成功，调度下一个任务，直到队列中所有任务都得到了一次调度的机会；
5. 第 i 轮调度结束，未完成资源匹配的任务进入第 i+1 轮调度，按式 (7)更新这些任务的轮次优先级。

3.2 分析任务执行、监控与回收

计算节点上的执行子系统-S维护一个环形任务缓冲区，接收执行子系统 -M 根据任务调度的结果发送来的分析任务，并负责任务的加载执行、监控和回收。

由于任务到达的随机性，为避免任务在缓冲区中过度等待，执行子系统-S采用多线程设计，主线程迭代地从任务缓冲区中读取任务而不必阻塞，并为每个任务创建一个监控线程。由于每个分析代码本身就是可独立运行的程序，通过在监控线程中启动管道进程(PIPE)的方法执行一个分析任务。同时，任务监控线程对管道任务进程进行实时地监控，根据系统设定的中止条件决定是否中止任务。在任务进程结束后，任务监控线程捕获管道任务进程的结束状态，进行判定和相应的处理。

3.2.1中止条件的选择与检查

中止条件应该体现多用户使用系统资源的公平性，避免超量使用和长时间占用的情况，因此，设定以下的中止条件：

中止条件 1：对资源的占用超过资源需求向量 δ 所声明的。

中止条件 2：持续运行时间超过任务 t 的超时阈值$T_0(t)$，按式 (8) 和 (9) 计算。

$$T_0(t)=request\_time(t)*\varepsilon*(1+\alpha) \quad (8)$$

$$\varepsilon=\frac{\sum_{t=1}^{m} actual\_time(t)}{\sum_{t=1}^{m} request\_time(t)} \quad (9)$$

式 (9) 中， request_time(t) 为由用户预估的任务 t 的请求运行时间， ε 为修正因子，为该用户的历史任务 (m 个) 的实际运行时间与请求时间的比值。 α 为弹性因子， $\alpha\in(0,1]$ ，体现超时阈值的宽容性。

任务监控线程通过定时读取 linux 操作系统下 /proc/<pid>/stat 文件的方法对任务中止条件进行检查。

3.2.2结束状态的判定与处理

任务的结束状态存在三种情况，由任务监控线程捕获管道进程的返回代码来判断：

情况 1：任务正常结束；

情况 2：任务运行过程触发了中止条件，被系统中止；

情况 3：由于代码自身问题或其他未知因素导致任务异常中止。

对于情况 1，由任务监控线程生成任务完成信息，并连同任务产生的结果文件，返回给分析子系统。对于情况 2，向分析子系统返回异常说明，对于情况 3，系统对用户任务具有一定的宽容性，提供首次出错重试机制，给予首次异常中止的任务重新运行的机会，若情况 3 不是首次出现，则直接返回异常说明。

3.3 计算集群系统监管

随着系统的运行，计算节点出现故障的概率随之增大，为提高系统的可靠性，监控子系统需要进行失效节点检测，及时发现问题节点，指导执行子系统做出正确的调度决策。同时，为协助系统管理者宏观把握集群系统的运行状况，需要设计一个覆盖集群范围的性能监测框架，采集、整理并呈现所有计算节点的历史性能数据。

3.3.1节点失效检测

失效检测(Failture Detection)主要基于心跳(Heartbeat)机制，即被监控进程 p 定时向监控进程 q 发送心跳消息，表示其运行正常。在主从节点模型中，失效检测存在主节点定时轮询和从节点主动上报两种工作模式。定时轮询实现简单，能够立即判定节点状态，但随着从节点规模增大，主节点定时轮询的资源消耗也将增大。主动上报实时性较好，主节点压力减轻，但一旦心跳消息由于网络等原因延迟到达，主节点容易对节点状态产生误判。

基于分析子系统的计算集群规模有限的事实，监控子系统采用定时轮询策略。各个计算节点上运被监控进程 p，总控节点上运行监控进程 q。q 在每经 $\Delta_{interval}$ 时间以远程过程调用RPC的方式启动各个计算节点上的 p，检查计算节点对 IP-SAN 存储的访问是否出现异常。发生以下情况，监控子系统判定计算节点失效：

- 未收到计算节点上的 p 的回应
- p 的回应显示节点无法访问 IP-SAN 存储

以上两种情况下，计算节点均无法顺利接受或执行任务。监控子系统一旦判定某个节点失效，则更新节点资源表中该节点为不可用，执行子系统在进行任务调度时将不会考虑该节点。

3.3.2集群性能监测

针对虚拟化计算集群实体机和虚拟机并存的特点，性能监测工作应集群性能监测工作从“多粒度，多维度”上展开。多粒度指监测的对象包含集群整体、实体机和虚拟机三个粒度，多维度指监测的性能测度涵盖进程负载、CPU、内存、磁盘 I/O 、网络 I/O 使用情况等多个维度。

遵循“分散采集，集中管理”的原则，使用开源分布式监控软件 Ganglia[10] 和 Host sFlow[11] 设计一个可变粒度的集中式集群性能监测框架，实现集群整体、实体机和虚拟机三个粒度上，多性能测度的采集、处理和呈现。性能监测的框架结构如下：

- sFlow agent：运行在实体机上的守护进程，负责采集实体机及内部虚拟机的性能数据，并以UDP单播周期性地向监测节点的Gmond发送。
- Gmond (Ganglia monitoring daemon)：Gmond是 Ganglia 体系结构中运行在被监测节点上，采集性能数据的守护进程。在该集成的结构中，Gmond数据采集的功能已为sFlow agent所替代，转而作为 sFlow 的数据收集器，运行在监测节点端，负责接收来自各个sFlow agent的性能数据，存储在内存数据库中。

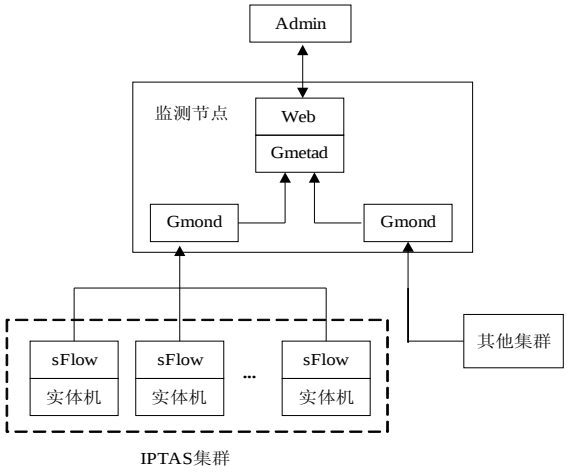

图 4 计算集群性能监测框架

- Gmetad (Ganglia meta daemon)：运行在监测节点的守护进程，周期性轮询Gmond的内存数据库，以 Web页面和图表的形式将性能数据呈现给管理者。

4 结论

本文分析了 IPTAS 分析子系统的缺陷，提出了一种基于虚拟化计算集群的改进方案，并对改进工作中的关键技术要点进行了详细的分析和设计，对于系统改进工作的具体实施具有重要的指导意义。

参考文献

[1]
Mark Allman， Ethan Blanton, Wesley M Eddy. A Scalable System for Sharing Intemet Measurements[C]. In Proceeding of the 2002 Passive and Active Measurement Workshop, Fort Collins, USA, March 2002.

[2]
The MOME Project Consortium. Information Technologies Society-Cluster of European Projects aimed at Monitoring and Measurement [EB/OL]. http://www.ist-mome.org/.

[3]
Colleen Shannon, David Moore, Ken Keys, et a1. The Internet Measurement Data Catalog[J]. ACM SIGCOMM Computer Communication Review, Oct, 2005: 35(5).

[4]
高亚东 ．大规模高速网络数据分析系统的设计与实现 [D]. 南京 ： 东南大学 , 2007.

[5]
张寒 ．面向代码移动的 IP_TASCM 分析集群内子系统的设计与实现 [D]. 南京 ： 东南大学 , 2009.

[6]
iSCSI. http://en.wikipedia.org/wiki/ISCSI.

[7]
R.Figueiredo, P.Dinda, J.Fortes. Resource Virtualization Renaissance[C]. IEEE Computer, 2005, 28-31.

[8]
Barham P, Dragovic B, Fraser K, et al. Xen and the art of virtualization[C]. Proceedings of the 19th ACM Symposium on Operating Systems Principles. New York, 2003: 164-177.

[9]
Kivity A, Kamay Y, Laor D, et al. KVM: the Linux virtual machine monito[C]. In Proceedings of the Linux Symposium 2007, Linux Symposium, 2007: 255-230.

[10] Matthew L.Massie, Brent N.Chun, David E.Culler. The ganglia distributed monitoring system: design, implementation, and experience[J]. Parallel Computing， 2004, 30: 817–840.

[11] Host sFlow: http://host-sflow.sourceforge.net/.