一个综合性集群监测模型 MCM 的设计与实现

秦海波[1], 魏晓辉[1], 李 博[1], 袁曙涛[2]
(1. 吉林大学 计算机科学与技术学院, 长春 130012; 2. 平台计算公司, 多伦多 L3R 3T7, 加拿大)

摘要: 根据已有的网络监测技术, 提出一个集群系统监测模型 MCM. MCM 将每个监测任务交给一个监测模块, 并可以灵活地加入和删除这些监测模块, 这种设计使得 MCM 可以有效地支持对分布式计算资源、服务以及异常事件的监测. MCM 为集群资源管理, 跨域并行作业, 网格资源协同分配和元调度算法提供了资源监测基础设施. 最后, 基于 MCM 和 Platform 公司的集群产品 EGO, 实现了一个高效的综合性集群监测系统.

关键词: 集群; 分布式; 资源管理; 监测
中图分类号: TP391 文献标识码: A 文章编号: 1671-5489(2008)01-0062-07

Design and Implementation of MCM, a Comprehensive Model of Cluster Monitoring

QIN Hai-bo[1], WEI Xiao-hui[1], LI Bo[1], YUAN Shu-tao[2]
(1. *College of Computer Science and Technology, Jilin University, Changchun 130012, China*;
2. *Platform Computing Inc., Toronto L3R 3T7, Canada*)

Abstract On the basis of existing network monitoring techniques, we presented a model of cluster monitoring called MCM. MCM resorts diverse monitoring jobs to a set of monitoring modules. MCM can add/remove the modules flexibly, and thus can monitor various resources, services and events efficiently. MCM constructs resources monitoring infrastructure for cluster resources management, cross domain parallel jobs, grid resources co-allocation and meta-schedule algorithm. Based on MCM and Platform Computing Corporation's production EGO, we implemented an efficient comprehensive cluster monitoring system.

Key words cluster; distributed; resources management; monitoring

集群和网格技术主要是通过高速网络和中间件技术将各种分布的计算设备联合起来协同工作, 高效地完成大规模的计算作业, 将分布的计算设备虚拟成一个超级计算机[1]. 集群计算中存在着分布式节点的资源管理和作业调度问题, 其中还包括集群服务的可靠性、集群的异步通知以及系统自我恢复等关键性问题[2,3]. 一套完善的解决方案需要考虑如下几方面:

首先, 集群节点的资源必须实现软件自动管理. 其次, 集群系统要保证其核心服务的可靠性, 需要对这些服务进行可靠性监测, 当服务出现异常时, 及时进行自恢复和重配置处理, 向管理设施发出异步通知. 对单个发生异常的节点, 集群系统也要探测到, 并进行实时处理[3].

目前, 集群监测的研究和商业开发得到很大推动和发展, 许多成果相继推出. 如集群监测系统 Supermony 依靠运行在节点上的 custom kernel module 返回集群监测系统; 集群监测系统 CARD[4] 使用

收稿日期: 2007-04-29
作者简介: 秦海波(1982~), 男, 汉族, 硕士, 从事分布式与网格系统的研究, E-mail: haibo.qin@gmail.com. 联系人: 魏晓辉(1972~), 男, 汉族, 博士, 教授, 从事分布式与网格系统的研究, E-mail: weixh@jlu.edu.cn
基金项目: 国家自然科学基金(批准号: 60703024)和吉林省自然科学基金(批准号: 20060532).

关系型数据库收集和存储监测结果. PARMON[5]是一个 C/S结构的集群监测系统, server端输出一系列节点的监测信息, client端则连接一个 server并解析数据. 还有较流行的商业软件 Big Brother, 是一套 C/S结构的分布式监测系统. Ganglia是其中较有代表性的网络监测系统, 是一个针对集群、网格等高性能计算系统的分布式监测系统.

本文在已有网络监测技术的基础上, 提出一个集群监测模型 MCM, 包括表示层、控制逻辑层和监测器层, MCM 将监测工作交给监测模块, 可以灵活的在监测器层加入和删除这些监测模块, 有效地支持了对分布式计算资源、服务以及异常事件的监测. MCM 可以将监测数据提供给集群系统、可视化控制台等第三方应用, 并支持对监测到的集群异常进行实时处理. 这个集群监测模型可以综合地解决集群节点资源监测、集群服务监测、集群异步通知捕获和集群异常实时处理等问题, 并可移植于各种集群上. MCM 为集群资源管理、跨域并行作业、网格资源协同分配和元调度算法提供了资源监测基础设施.

1　集群监测模型 MCM

1.1　MCM 的总体结构

MCM 模型包含表示层、控制逻辑层和监测器层三层结构, 如图 1所示.

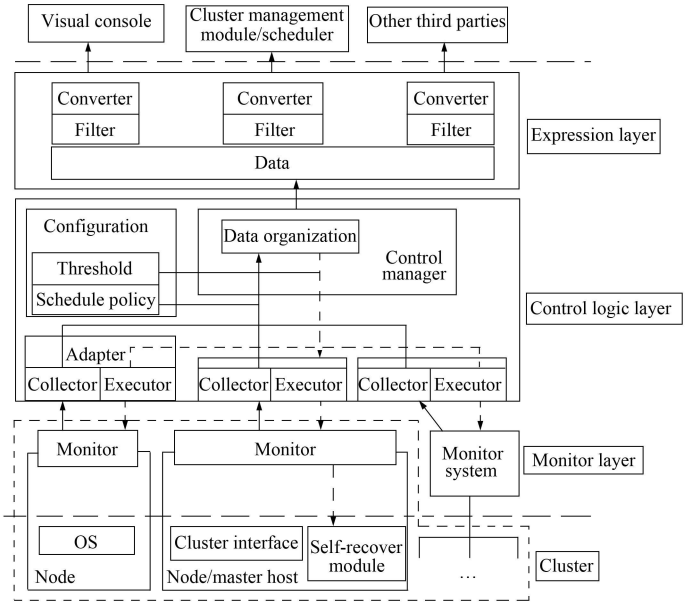

Fig 1　Architecture of MCM

1.2　表示层

表示层负责向集群系统、可视化控制台等第三方应用提供定制的监测数据 (如 CPU 利用率、空闲内存空间等), 表示层的主要任务是筛选和转化输出控制逻辑层的监测数据. 监测数据的存储和查询可以有多种实现方式, 如监测数据的最小单位为一个数据项, 由一对名字/值组成, 同一第三方定制的数据项组织在相同数据块中, 并且每个数据块由一个索引号标识. 其中第三方应用是通过位于控制逻辑层的配置文件定制所需数据, 且数据块的索引号对应着 MCM 当前的第三方应用. 表示层首先从监测数据中提取一个第三方应用定制的数据块, 该工作由滤选器完成. 每个第三方应用在表示层都配备一个滤选器, 滤选器根据所属第三方应用在 MCM 中的索引号, 从监测数据中找到并提取数据条目. 其次, 第三方应用, 尤其是集群系统, 都有内部的资源表示方式, 如网格搭建工具包 globus1.0使用的 RSL(Resource Specification Language)[6], 因此表示层为每个第三方应用都配备数据转换器, 将数据块的数据格式转换为第三方系统内部的数据格式. 实现 MCM 时可使用 XML格式表达监测数据, 对于不同的第三方应用, 利用丰富的 XML处理工具, 可以方便地在表示层为它们编写数据转换器 (见图 1). 同样, 对于后台使用数据库的第三方应用, 编写表示层输出与数据库接口的转换器, 把监测数据输出

到数据库. 表示层灵活的数据输出格式使MCM对第三方应用有良好的可扩展性.

1.3 控制逻辑层

控制逻辑层在MCM中负责收集和提交监测数据, 指导监测器对集群异常进行实时处理. 第三方应用的定制需求较复杂, 对于不同的被检测集群, 监测器的实现也不同, 为了使MCM具有良好的可扩展性和可移植性, 在设计控制逻辑层时采用开放式的框架结构. 各种实现方式的监测器只要配有专门的适配器就可以成为控制逻辑层中控制管理器的管理对象. 适配器负责衔接控制管理器与监测器之间通信, 包含收集器和执行器, 分别处理控制管理器与监测器之间的数据流和控制流. 因此, 不同的监测器只要配有专门的适配器, 便可提交监测数据, 接收控制管理器的指令. 反之, 只要按MCM内部协议提交监测数据、接收受控制管理器指令的系统, MCM便将其视作监测器. 通过适配器的这种协议转换, 控制逻辑层可以接纳监测器层中各种实现方式的监测器.

控制逻辑层的各种组件在控制管理器协同下工作. 控制管理器首先通过适配器读取监测器返回的监测数据, 将数据交给表示层, 再将监测数据与设定阈值进行比较, 判断有无异常发生, 当发现异常, 则生成指令, 指示监测器做异常处理. 适配器负责衔接控制管理器与监测器之间的交互, 其中包含的收集器和执行器分别处理数据流和控制流. 监测器层包含运行在集群节点或master主机上的监测器, 它们按控制管理器的指令进行监测和异常处理操作.

控制逻辑层的工作流程包括数据流和控制流, 分别代表监测数据在控制逻辑层的处理过程和控制管理器控制监测器执行指令的过程.

1.3.1 *数据流* 控制逻辑层与监测器间的数据流在图1中用实线表示. 监测器完成底层的监测操作后, 将得到的监测数据通过网际协议传送给收集器, 收集器再通过MCM内部协议传送给控制管理器. 面对众多监测器返回的监测数据, 控制管理器数据收集过程很容易成为整个模型的性能瓶颈, 所以控制管理器收集数据的过程要采用有效的调度策略, 实现并发机制[7]. 在MCM的设计中, 控制管理器将各种需要处理的事务看作是一组事件, 如从某个收集器读取监测数据, 向表示层提交处理好的数据等, 控制管理器将这些事件按设定的调度策略排在一个以时间标记的队列中, 当当前队列首事件的标记时间到达时, 控制管理器处理该事件, 否则主管模块接收新的事件并将其排在队列中. 在收集数据的过程中, 控制管理器先向所有监测器发出监测指令, 监测器得到指令后, 并行执行监测指令. 在监测器返回数据前, 控制管理器去处理别的事件, 监测器返回数据给适配器后, 控制管理器会收到通知, 控制管理器将读取该数据作为一个待处理事件排在队列中, 并标记处理时间. 当到达时间时, 控制管理器从适配器读取监测数据.

控制管理器对收集到的数据进行格式规范, 按照配置文件中第三方应用的定制条件将数据项组织成与第三方应用对应的数据块. 这样提交到表示层的数据就可以被进一步的筛选、转换.

1.3.2 *控制流* 控制逻辑层与监测器之间的控制流在图1中用虚线表示. 控制管理器通过执行器向监测器发出指令, 指导监测器进行监测和异常处理. 控制管理器在组织监测数据时, 会同时把监测数据与设定阈值作比较, 以判断是否出现集群异常. 当超出设定阈值时, 会把异常处理指令实时发给监测器, 而不需要排在控制管理器的事件队列中等待.

1.4 监测器层

监测器层由各种安装在集群节点/master主机的监测器构成, 监测器接收控制管理器的指令进行实际的监测和异常处理操作. MCM使用适配器连接监测器与控制逻辑层的控制管理器, 监测器的实现不受限制. 监测器的具体实现可以是一个可执行脚本, 一个监测进程, 甚至是另一个监测系统, 只要控制逻辑层为其配备了适配器, 它们就可以集成到MCM中. 这样监测一组节点的监测系统可以封装成MCM中的一个监测器, 不同设备的监测工具也可以组合地封装成MCM中的一个监测器, MCM监测器层的这种开放性使得MCM可支持分布式监测, 随集群规模的扩展而扩展.

监测器层的工作主要有三方面: 节点资源和集群服务监测、集群异步通知的监测和集群异常实时处理.

1.4.1 *节点资源和集群服务的监测* 节点资源由本地操作系统直接管理, 因此监测器从操作系统提

供的接口获得节点资源信息. 而监测器要监测集群服务, 则需要集群系统提供相关接口. 监测器通过这些接口获取集群服务的各种元数据, 再据此判断集群服务的状态. 不同的集群提供不同的接口, MCM 中监测集群服务的监测器会因此失去可移植性, 如果能够制定出一套集群系统接口规范, 就能开发出可移植的集群服务监测器.

1.4.2 异步监测方式　集群监测不能只有轮询式的同步方式, 还要有中断式的异步方式. 同步方式的缺陷在于信息的实时性, 尤其是异常监测的实时性. 同步方式需要选择时间的间隔和询问顺序, 如果询问间隔太小, 则将产生太多不必要的通信量. 如果询问间隔太大, 并且在询问时顺序不合适, 则关于一些灾难性事件的通知又会太慢. 异步通知的方法可以实时通知监测系统, 但发送异步通知需要系统资源. 如果通知必须转发大量的信息, 则集群节点不得不消耗更多的系统资源发送通知, 减少了集群的可利用资源. 而且, 如果几个同类型的异常事件接连发生, 则大量网络带宽可能将被相同的通知所占用.

所以, 两种方式相结合是进行集群监测最有效的途径. 集群异步通知是集群系统通知外界集群事件发生的方法, 很多集群系统都支持异步通知机制, 采用不同的方法发出通知 (如日志记录、snmp trap 等). MCM 中的异步通知监测器能够捕捉集群异步通知, 这类监测器持续监听集群系统的通知发送通道, 控制逻辑层也为其配备异步工作方式的适配器. 即使异步通知在发送到监测器的过程中丢失, 同步方式的监测器也可以在后续的监测中发现问题[8].

1.4.3 实时异常处理　控制管理器在组织监测数据的同时判断当前数据项的值是否超过设定阈值, 阈值的超出将触发控制管理器的异常处理指令, 指示监测器进行实时异常处理. 实时异常处理需要集群节点 /master主机预装自我恢复模块, 监测器收到控制管理器异常处理指令后调用自我恢复模块, 其中引起集群异常的数据项会作为参数传给自我恢复模块, 自我恢复模块据此执行不同的恢复操作. 完整的集群系统大多含有与自我恢复相关的模块, MCM 的监测器在实现实时异常处理时可以充分利用它们. 自我恢复操作带有不可逆性 (如回滚、重初始化等). 为防止攻击, 集群安全机制和操作系统安全机制都会给自我恢复模块设定较高的权限, 监测器调用时将被要求多次认证. 因此, 要给这类监测器在节点本地操作系统设置高权限, 并在集群系统中建立一次登陆机制, 这样监测器才能在调用自我恢复模块时自动通过认证.

2 MCM 的实现

本文以开源软件 Nagios和 Platform公司的集群产品 EGO为设计环境, 给出 MCM 一个参考性的实现及其性能分析.

2.1 Nagios

Nagios是为了解决网络监测问题而开发的开源项目. Nagios网络监测框架由前台的 Web界面、中间的监测框架和后台的插件构成. Nagios的插件都是可执行程序, 由监测框架中的 Nagios主进程并行调用插件, 插件程序的返回值和标准输出作为监测结果的返回渠道. 对于插件的内部逻辑, 监测框架没有做任何约束. Nagios主进程采用内部数据格式将监测数据存储在文件 status dat中, 供前台的 cgi 程序读取. 前台的 cgi程序把读取的监测数据在动态页面中显示给用户. Nagios为了支持分布式监测, 为其他监测进程提供了一个有名管道 cmd, 其他监测进程可以将监测结果按 nagios内定格式写入此管道, 由 Nagios主进程读取[9]. Nagios的结构如图 2所示.

在 Nagios框架的基础上, 依照 MCM 的设计实现了一个集群监测框架, 框架含有新的控制管理进程和数据表示模块, 取代了原来的 Nagios主进程和文件 status dat 本文保留 Nagios前台作为一个第三方应用.

2.2 EGO

EGO(Enterprise Grid Orchestrator)是世界主要网格计算产品供应商之一 Platform Computing公司所开发的集群产品, 其目标是将集群里每个节点的计算能力、存储空间、I/O 带宽等虚拟化成一个 IT 资源的网格平台, 这个平台为用户提供一个单一的管理环境, 可在地理位置分散的站点之间, 为所有的

关键任务应用、服务及工作任务集中分配共享资源, 以切实降低企业的 IT总成本[10].

集群对用户提供了若干用于操控关键任务的服务接口, 可使用户对关键任务进行控制、定向和部署等. EGO为上层的元调度器、资源代理、虚拟机服务和第三方软件集成等提供了一套多种语言协议(SOAP, C, Java等)的 SDK接口, 这些接口与集群核心进程 vemkd交互, 完成集群的信息查询、资源分配和作业执行等操作. EGO拥有完整的安全认证机制和异步通知机制. EGO的结构如图 3所示.

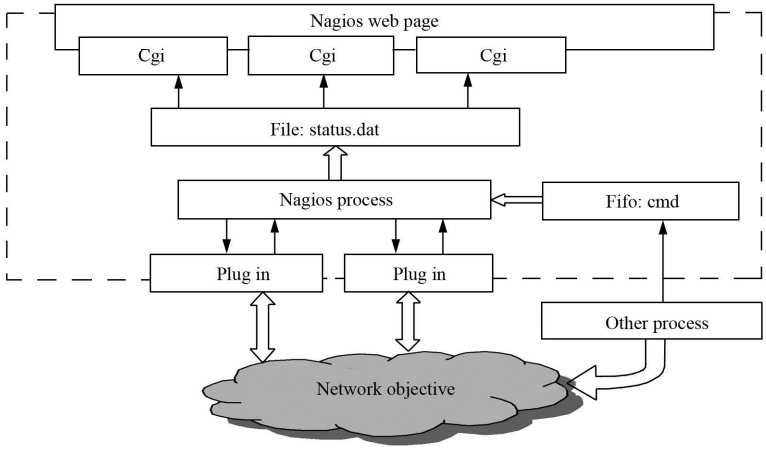

Fig. 2 Architecture of Nagios

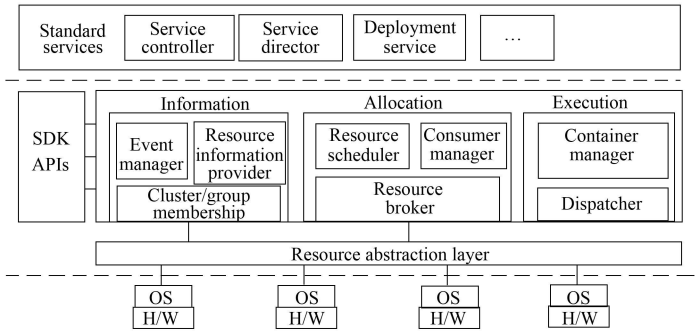

Fig. 3 Illustration of EGO

EGO的系统管理员需要监测每个节点的硬件资源、节点的集群参数、节点的作业负载情况以及EGO核心服务的运行状态和 EGO的异步通知等.

2.3 集群监测实现的效果和性能

图 3中的 Resource Information Provider模块可提供 EGO的节点资源信息, 且开发者可以通过 SDK接口与其交互. 据此我们开发了若干节点资源监测器, 同时开发了相应适配器并以插件的形式插入集群监测框架. 控制管理进程从适配器读取监测数据后, 进行组织并通过数据表示模块提交给 Nagios前台的 cgi程序, 节点的资源信息最终显示在 Nagios的 Web界面上. 类似的, 集群核心服务的状态可通过 EGO的 Service Controller获得, 利用 Service Controller的接口开发集群核心服务监测器, 并在 Nagios框架中插入对应的适配器插件. 图 4为 Nagios界面的截图, 显示了两组监测项目, 第一组监测集群的

EGO cluster	Core service	OK	12-22-2006 10:37:46	5 d 17 h 33 m 19 s	1/4	Connected! clustername: cluster 1: master: test007.grid.org: version: 1.10
Node 1	EGO set parameter	OK	12-22-2006 10:34:54	0 d 0 h 5 m 1 s	1/4	CPU factor: 13.20: total slot(s): 1: free slot(s): 1:
	EGOHost hardware resource	OK	12-22-2006 10:37:03	0 d 0 h 2 m 52 s	1/4	Available mem: 615.00 M total mem: 1 022 M available swap space: 1 755.00 M total swap space: 2 458 M 1CPU(s) 2disk(s)
	EGOHost system info	OK	12-22-2006 10:39:11	0 d 0 h 0 m 44 s	1/4	OS: NTX86: mode: PC450: ldle time: 0.00 min login session(s): 1:

Fig. 4 View of cluster core services and node resources

核心服务，结果显示服务处于 OK 状态，在这种情况下监测器返回了集群的名称、master主机名称等. 第二组则在监测到节点 node1处于 OK状态时返回了 node1的负载情况、硬件资源以及系统信息.

EGO的内部事件由 EventManager模块管理，每个事件由来源、事件代号、级别、参数等若干字段组成，通知的发送方式支持日志记录和 snmp trap. 本文实现了一个集成了 snmp monitor的监测器，该监测器使用 snmp monitor捕捉 EGO以 snmp trap封装的异步通知，而 snmp monitor配置的 trap handler脚本扮演适配器的角色，它解析异步通知里各字段内容，并将数据写入有名管道 cmd. 这样监测器捕获的 EGO异步通知经过控制管理进程和数据表示模块的处理，显示在 Nagios的 Web界面上. 监测器 GetNotification捕捉到 EGO的异步通知后，在 Nagios界面显示出来的效果如图 5所示.

Host	Service	Status	Last check	Duration	Attempt	Status information
EGOmaster	Getnotification	OK	12-22-2006 18:55:52	0 d 0 h 0 m 42 s	1/1	Component: VEM Event code: SYS_HOST_DOWN Level: WARN Desc: VEM detects a host is down Cluster name: cluster 1　　host name: node 2.jlugrid.org

Fig 5　View of EGO asynchronous notifications

由图 5可见，监测器捕获到 EGO master主机发出的异步通知，监测器返回了捕获通知的时间和通知里包含的消息内容，该消息报告 vemkd探测到集群 cluster1中的主机 node2. jlugrid. org关机. 该参考实现中实现了丰富的监测功能(如图 6所示)，Monitoring Service为一个包含若干监测项目的监测服务. 如图 4所示，第一个 Monitoring Service命名 Core Services，监测的对象是运行在 EGO master的核心进程，其中包括 4项监测项目：监测 EGO master主机是否能连接上；获取集群名称；获取 EGO master的主机名；获取当前 EGO的版本号. 由图 4可见，第一行的尾部正确地返回了在 EGO master运行正常时的所有监测结果.

Gore service
- If the master can beconnected
- Cluster name
- Master's host name
- EGO version

EGO host summary
- Status summary of nodes in the cluster

Host's hardware resource
- Amount of CPU
- Memory space
- Swap space
- Total memory space
- Total swap space
- Amount of disks

Parameters set by EGO
- Amount of slots
- Amount of free slots
- CPU factor

Host's overload
- I/O capability
- 15-second CPU run queue length
- Minute CPU run queue length
- 15-minute CPU run queue length
- Minute CPU utilization
- Paging rate

EGO's notification
- All EGO notifications

Fig 6　Monitoring items in our reference implementation

为了验证监测异常的能力，关闭 EGO master上的 EGO核心进程，图 4中的第一行返回值如图 7所示.

由图 7可见，界面返回了异常级别、发生时间等摘要信息，点击监测服务名称，可看到详细诊断信息. 深色背景为了提醒用户异常的发生.

EGO cluster	Core service	CRITICAL	12-22-2006 10:40:10	0 d 0 h 3 m 10 s	1/4	EGO master cannot be contected!

Fig 7　View of exceptions

为了验证监测数据的真实性，本文在 node1上运行样本作业 SJ. 该作业申请 100 M 内存，运行 SJ后，界面上返回的结果如图 8所示.

Node 1	EGO set parameter	OK	12-22-2006 10:40:03	0 d 0 h 6 m 11 s	1/4	CPU factor: 13.20: total slot(s): 1: free slot(s): 1:
	EGO host hardware resource	OK	12-22-2006 10:42:03	0 d 0 h 3 m 08 s	1/4	Available mem: 510.00 M total mem: 1 022 M available Swap space: 1 700.00 M total swap space: 2 050 M 1 CPU(s) 2disk(s)

Fig 8　View of available memory space

由图8可见,第二行尾部显示可用内存510M.与图4相比,返回的结果完全符合节点node1的资源使用情况.

参考实现控制管理进程含有事件触发指令的代码,当所监测关键任务的状态由OK转到非OK时,会触发自我恢复指令.仍以上面的例子为假设,当监测到EGO核心进程连接不上时,自恢复模块将试图重启EGO master.同其他集群系统类似,EGO提供了接口服务Service Controller,可提供关键任务的当前状态与启动和停止关键任务的操作接口,本文针对EGO开发了自我恢复模块,监测器发现异常时会立即调用它们,重初始化集群系统的关键任务.这一参考实现能够实时监测EGO地域上分布的所有节点的计算资源与集群核心服务的状态,捕捉EGO发出的异步通知,对非正常状态的关键任务可以实时恢复.该实现保留了Nagios的插件机制,可连接各种实现方式下的监测器,满足各种监测需求.新的数据表示模块可使该集群监测框架与多种第三方应用进行集成.

参考文献

[1] Massie M L, Chun B N, Culler D E. The Ganglia Distributed Monitoring System: Design, Implementation, and Experience [EB/OL]. 2004-07-15. http://www.cs.cmu.edu/~15849g/readings/massie04.pdf

[2] Smith C, Henry D. High-performance Linux Cluster Monitoring Using Java [EB/OL]. 2006-04-12. http://www.linux-clustersinstitute.org/Linux-HPC-Revolution/Archive/PDF02/32-Smith_C.pdf

[3] Roney T, Bailey A, Fullop J. Cluster Monitoring at NCSA [EB/OL]. 2006-05-14. http://www.linuxclustersinstitute.org/Linux-HPC-Revolution/Archive/PDF01/Roney_NCSA.pdf

[4] Anderson E, Patterson D. Extensible, Scalable Monitoring for Clusters of Computers [C]//Proceedings of the 11th Systems Administration Conference. San Diego, California: USENIX Association, 1997: 9-16.

[5] Buyya R. PARMON: a Portable and Scalable Monitoring System for Clusters [J]. Software: Practice & Experience (SPE), 2000, 30(7): 723-739.

[6] Czajkowski K, Foster I. A Resource Management Architecture for Metacomputing Systems [EB/OL]. 1998-04-02. ftp://ftp.globus.org/pub/globus/papers/gram97.pdf

[7] Argent Software Inc. Argent Engineering Note: Cluster Monitoring [M/OL]. 2001-05-12. http://www.argent.com/doc/gd/argent_cluster_monitoring_b01.pdf

[8] Case J, Fedor M. rfc1067 [EB/OL]. 2006-07-13. http://www.ietf.org/rfc/rfc1067.txt

[9] Galstad E. Nagios Documentation [M/OL]. 2006-03-17. http://nagios.sourceforge.net/docs/2_0/.

[10] Platform Computing Inc. EGO Whitepaper [EB/OL]. 2005-08-13. http://www.platform.com/NR/rdonlyres/FED88940-8381-406A-948C-CC676407BB9E/0/EGO_WP2.pdf

(责任编辑:韩啸)