基于GMA的轻量级高性能计算环境监控

王 伟 魏 勇 张晓霞 罗红兵

(北京应用物理与计算数学研究所 北京 100094)

摘 要 针对高性能计算环境监控的一般要求,分析对比现有的监控体系结构,设计了基于GMA体系结构的监控系统,详细介绍了设计中的几个关键问题,并给出了解决办法,最后实现了一个轻量级且高效的高性能计算环境作业监测系统。

关键词 高性能计算环境,GMA,监控系统

中图法分类号 TP316.4 **文献标识码** C

Lightweight High Performance Computing Environment Monitoring System Based on GMA

WANG Wei WEI Yong ZHANG Xiao-xia LUO Hong-bing

(Department of High Performance Computing Center, Institute of Applied Physics and Computational Mathematics, Beijing 100094, China)

Abstract For the general requirements of high-performance computing environment monitoring, we analyzed and contrasted the existing monitoring architecture. Based on grid monitoring architecture, we designed and realized a lightweight and efficient monitoring system, discussed several key issues of the system design in detail. Finally we realized such an efficient and lightweight monitoring system in high performance computing environment.

Keywords High performance computing environment, GMA, Monitor system

1 引言

高性能计算资源管理一直是高性能计算领域的重要研究课题。随着高性能计算机系统的发展以及应用需求的迅速增长,许多高性能计算中心安装了多台高性能计算机来满足用户多层面的计算服务需求,从而形成了一个复杂的高性能计算环境。例如,美国的劳伦斯·利弗莫尔国家实验室(Lawrence Livermore National Laboratory,简称LLNL)在2011年处于运行状态的高性能计算机有25套,总计算能力超过2.388千万亿次[1]。由于该类高性能计算环境涉及到多套计算机,系统规模庞大(从几千核到上万核)、用途多样(科学计算、数据可视化等),使得单纯使用高性能计算机自身的监控系统已不能满足该类计算环境管理和使用的需要。要实现计算资源的高效管理和合理利用,必须建立统一的系统监控,汇集各计算机运行状态、节点使用情况以及用户作业运行状态等信息,以方便管理员的管理和用户对系统的状态查询,为计算资源的统一调配奠定基础。

现在的监控工具有NWS[2]、MDS[3]、ParMon[4]、SuperMon[5]等。NWS(Network Weather Service)是University of California, Santa Barbara(UCSB)开发的周期性监测并动态预测性能的工具,能够完成广域环境下部分资源状态信息的收集,并提供系统的性能预测功能,为调度等工作提供了有力的支持。但是,NWS的监测信息不太完整、访问接口相对复杂,这些结构设计上的缺陷使它不能广泛应用[6]。MDS(Monitoring and Discovery Service)是Globus Toolkit(简称GT)的一个组件,它构建在轻量级目录访问协议(Lightweight Directory Access Protocol,简称LDAP)的基础之上,为访问底层信息提供者收集的性能数据提供了统一、灵活的接口。MDS采用了分散式的结构,因此具备了较强的可扩展性。但MDS对GT4容器的过分依赖,使得其安装使用非常复杂,进而限制了它的推广使用[6]。ParMon由印度高性能开发中心研制,采用Client/Server监控工作模式,ParMon在机群规模较大的情况下(如几百个结点),获得所有结点的状态信息比较耗时,使得监控所反映的系统状态信息的实时性稍差。SuperMon由Los Alamos国家高性能计算机实验室研制,采用了可扩展的层次化结构,并实现了多层之间的数据共享。Supermon在节点规模变大时速度会线性下降,同时存在Supermon这一系统单一失效点,监控系统的可靠性无法保证。而且,Supermon是通过预先设定的地址与每个mon连接,在不改变Supermon配置的情况下很难做到被监控点的动态增加。此外,还有许多监控工具,如HawKeye、GridView、CGSV、GridEye等[7]。网格监控软件都是针对网格这一广域、跨管理域环境的监控,监控信息的实时性普遍不强,系统开销也较大,并不适合园区内的高性能计算环境管理。

到稿日期:2011-04-20 返修日期:2011-07-30 本文受国家自然科学基金(60803045),中国工程物理研究院科学技术发展基金资助项目(2010B0403058)资助。

王 伟(1984-),男,硕士,主要研究方向为高性能机管理及性能优化,E-mail:wang_wei@iapcm.ac.cn;魏 勇(1965-),男,技师,主要研究方向为高性能机管理及性能优化;张晓霞(1973-),女,高级工程师,主要研究方向为高性能机管理及性能优化;罗红兵(1968-),男,研究员,主要研究方向为高性能机管理及性能优化。

针对园区内单一管理域高性能计算环境监测需求，确定了以下设计要求：

(1)高性能。监测系统对被监测系统的资源占用应尽可能小，这样才能减少对被监控系统的干扰，从而获得比较准确的结果；同时，只有这样的监测系统才是实用的。

(2)可用性。监测系统应能获得足够的系统信息，并保证数据的准确性。但获得的信息越多、更新越及时，占用的资源也就越多。因而可用性与高性能的要求有一定冲突，需要根据实际情况进行权衡。

(3)健壮性。监测系统自身不能给被监测的系统带来安全隐患或影响其稳定运行。

针对以上要求，考虑到 GMA(Grid Monitoring Architecture，GMA)体系结构[8]具有对被监控资源影响较小、开放性好、可扩展性强以及低延迟和高效率等特点，本文基于 GMA 体系结构实现了一套作业监控系统(JSMON)。JSMON 采用基于 GMA 的体系结构，适用于园区局域网络内的高性能计算环境监控，JSMON 提供计算资源和用户作业的全局监控功能。目前，JSMON 已投入实际使用。

2 JSMON 的结构

GMA 是全球网格论坛(Global Grid Forum，GGF)提出的网格监控体系结构，它提出网格监听的生产者-消费者模型，也是目前网格监听体系结构常用的框架原型。在 GMA 中，监控数据的基本单位是一个事件，事件是一个经命名的、有时间戳的结构，此结构可以包含一个或多个数据条目，这些数据关联到一种或多种资源，如网络的使用率，或应用特定的数据，如应用的运行时间。产生可用事件数据的组件叫生产者，请求或接收事件数据的组件叫消费者，目录服务用于发布什么事件数据是可得的以及是从哪个生产者获得的。基于 GMA 体系结构，JSMON 由 Client、消费者、生产者、目录服务构成，各模块的功能以及关系如图 1 所示。

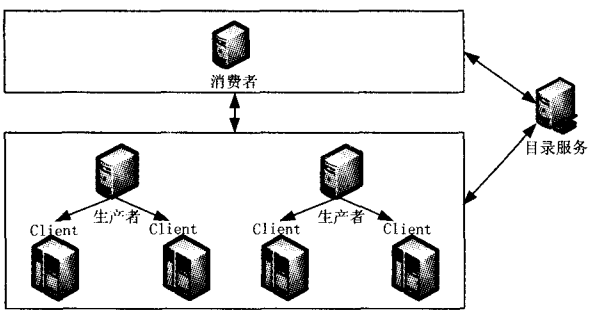

图 1 监控系统体系结构

(1) Client 运行在资源节点上的信息收集客户端，建立在高性能计算机系统的基础上，直接与高性能计算系统对接，利用高性能计算系统的监控接口，得到监测数据。它对高性能计算系统造成的负载最小，同时，机器的差异性由其负责屏蔽。

(2) 生产者运行在主要的资源节点上，接受来自各个 Client 的监控信息，并对信息进行一定的分析、处理、转换、存储并传送给消费者。生产者必须定时向目录服务注册来声明自己的存在，同时发布自己能够提供的数据类型。

(3) 目录服务包含所有可用数据的列表，使消费者可以发现当前可用的数据、数据的特性，以及应该和哪个生产者联系以获取指定类别的数据。目录服务本身并不存储数据，它仅仅包含关于提供的数据的名称、特性和位置等信息。这样，用户可以通过目录服务找到该数据的生产者的静态信息，比如数据类型、主机地址等，然后再向该生产者发出请求，获得相应数据。

(4) 消费者向生产者发送查询请求，从生产者获取数据后根据需要将数据整理、加工，以合适的形式提供给最终的用户或者上层应用。

JSMON 的工作流程如下：生产者在目录服务注册自己的存在并声明自己可以提供的服务及数据类型，之后消费者访问目录服务根据感兴趣的数据查找到生产者的信息，通过直接和生产者联系获得生产者的信息；Client 负责收集监控信息，并将收集到的监测信息传递给生产者，生产者将数据信息进行格式转换，与消费者进行通信，将转换后的信息传送给消费者，之后消费者响应用户的请求，将监测结果展示给用户。在实现的过程中，涉及到监控数据的表示、历史数据的存储、收集方式、数据传输模式和协议、信息的发布模式和服务机制等关键问题，将在下节进行详细介绍。

3 系统设计的关键问题

JSMON 采用基于可扩展标记语言(eXtensible Markup Language，简称 XML)[9]的数据格式来描述收集到的监控数据，使得系统的状态数据能跨越不同的系统平台进行交互，以解决系统异构性问题。Client 采集到的监测数据具有更新频繁、数据量大、时效性强等特点，将所有的监控数据进行长期存储是没有意义的，但是对于某些比较重要并且相对稳定的数据(如运算时间比较长的作业等)，有可能需要进行长期存储，用来进行历史分析。针对以上两种不同的数据，JSMON 采用了两种不同的存储方式：对于短期的性能数据采用循环数据库(Round Robin Database，简称 RRD)[10]来解决性能数据的存储。而对于一些较为重要并且相对稳定而要进行长期存储的数据，则采用 XML 的方式来进行存储。

3.1 数据收集方式

考虑到数据的收集过程需要一定时间，在进行数据收集时必须选择合理的时间间隔，否则就会由于数据溢出或丢失造成监测结果出现较大的偏差，从而影响数据的精确度。由于数据收集和数据分析处理本身都要消耗一定的网络带宽和设备资源，因此收集间隔时间必须在一个合理的范围内。收集频率过低，不能反映性能参数的真实变化情况；收集频率过高，将给整个网络带来巨大的负担。

定义系统一次服务的开销函数为

$$Cost = M \times N \times perCost \tag{1}$$

式中，M 为机器数，N 为每秒收集的次数，$perCost$ 为每次的收集开销。

每次收集开销一共包括 3 部分：生产者开销、传输开销和数据分析开销。

$$perCost = produceCost + transferCost + analyzeCost \tag{2}$$

由于使用了集群的监控系统接口，$produceCost$ 可以认为是个常量。$transferCost$ 和 $analyzeCost$ 都与监控数据的字节数有关。

$$transferCost = N \times perByteTransferCost \tag{3}$$

$$analyzeCost = N \times perByteAnalyzeCost \tag{4}$$

式中，N 为监测数据的字节数。

假设有两台机器，一个机器的采集时间为 0.1577s，另一台机器的采集时间为 0.1765s，如果采用顺序采集的方式，则按照上述公式，一次采集的时间开销为 0.1577＋0.1765＝0.2342s，这样可得每秒钟的采集频度上限为 4 次。如果采用多线程并行采集的过程，则以最大的采集时间 0.1765s 为每次采集的开销，从而可得每秒钟的采集频度上限为 5 次。因此，JSMON 系统采用多线程并行采集方式。

一般的数据采集间隔在 3～5 分钟之间即可。本系统采用 5 分钟作为收集间隔。

3.2 数据传输模式和协议

GMA 体系结构定义的生产者与消费者之间的数据传输有 3 种模式[11]：订阅模式(publish/subscribe)、请求响应模式(query/response)和通知模式(notification)。在订阅模式中，一般由消费者在生产者方声明感兴趣的数据和相关的传输参数，之后生产者向消费者发送数据。在请求响应模式中，消费者发出请求，随后生产者将消费者需要的数据一次发送出去。在通知模式中，生产者主动向消费者发送数据。

通过统计用户访问行为发现，用户访问主要集中在上班时间，采用订阅模式，在下班之后的时间段内，监控数据采集和传输是对资源和网络带宽的浪费。如果采用请求响应模式，在用户访问时才会请求监测数据，但在用户访问量大的时候，如果每来一个用户就请求一次，则会对资源和网络带宽造成极大的压力。如果采用通知模式，由于没有交互过程，用户无法定制自己感兴趣的数据。

上述 3 种工作方式单独使用都不能很好地解决监控问题，因此有必要将 3 者的优点综合考虑，并重新实现。因此 JSMON 的工作模式为：

当用户访问较少的时候，采用请求响应模式；

当用户访问量较大的时候，则采用订阅模式，即消费者声明数据，然后统一提供给用户使用。在客户端实现定制功能。

用户访问量的问题预定义在监控服务器端，可以由管理员对参数进行定制。这样可以根据实际情况将该工作模式调整到一个最佳的状态。

由于集群可能因为各种情况而出现无法响应的情况，因此在消费者与生产者通信的时候，需要判断生产者是否处于正常工作状态，我们给出了生产者无法响应或者不能提供信息的定义：

当生产者第一次接收请求之后没有响应开始，监控服务器端维持一个失败计数器，当计数器达到一定次数时(该次数可定制)，则可认为该集群无法响应，将该集群的状态置为不可用。

若一个服务器处于不可用状态时，监控服务器收到该集群的响应消息，监控服务器需要将该集群重新纳入到可以提供服务的状态，因此，监控服务器统计一个响应计数器，集群连续响应达到一定次数后，则认为集群正常，重新将集群的状态置为可用。

3.3 信息发布模式和服务机制

通过将开销函数扩展至一个工作日来看，开销公式有如下定义：

$$Cost=\sum_{i=0}^{n} i\times perCost \tag{5}$$

在 $perCost$ 中占最大比重的是 $transferCost$，其余的两项可以忽略不计，则上述公式可以写为

$$Cost\approx\sum_{i=0}^{n} i\times transferCost \tag{6}$$

式中，i 为请求的次数，无论采用何种工作模式，数据都需要在网络中传输，因此，为了减少网络传输的次数，系统根据用户的访问情况，提供如下不同的信息发布方式。

1. 当只有一个用户访问监控系统的时候，系统响应用户的第一次访问请求，之后如果用户没有操作，也不关闭监控页面，则消费者服务器端每隔一定周期去访问消费者。

2. 当多个用户访问的时候，系统响应第一个用户的访问请求，随后访问的用户系统将根据用户的访问时间和上次提供的服务时间进行比较，如果用户访问的时间仍然没有到下次访问间隔，则服务器返回上次的监控数据。

3. 用户访问一次页面，服务器请求一次监控数据。

多个用户访问的时候，服务器端接受用户的请求，根据用户的访问时间和监控数据的时间戳进行比较，如果用户的访问时间介于两者之间，则返回前次的数据监测结果，等到重新收集的时刻，刷新用户界面，完成数据监控。可以看出，多用户访问的开销近似为单用户单服务的开销，只是在服务器端需要处理用户服务，不需要占用服务器与集群之间的带宽和服务器的资源，系统负载最小。服务器端的刷新频率由用户定制，写到配置文件中，方便用户根据不同的情况选择定制，以达到最好的效果。

4 JSMON 系统的实现及评价

4.1 系统实现

JSMON 系统采用 Java 开发，底层采用 Java 的本地调用(Java Native Interface，简称 JNI)技术调用高性能计算机的监控接口，高层采用 Web 方式提供服务，很好地实现了高性能计算环境的监控。在实验环境中，生产者(即服务器或集群等)采用 Linux 平台，客户端采用 Linux 平台。用户界面使用网页形式，采用 Tomcat 作为 Web 服务器。

JSMON 向用户提供了两种查询方式：

(1)通过目录服务查询到所要查询的数据的静态信息(如 IP 地址等)，然后按照层次结构从上至下依次解析这些静态信息从而找到数据所在的节点，由运行在该节点上的 Client 收集要查询的节点的状态信息，然后将监控到的数据依次向上一级的生产者发送，最高层的生产者将所有的监控数据根据需求进行处理后以适当的形式返回。

(2)如果已知监控数据的某些静态信息(如 IP 地址)，则可直接查询相应的生产者，由生产者直接返回 Client 监控到的数据，这种方式不用查询目录服务，但必须提前掌握监控数据的某些静态信息，这些静态信息必须能够惟一地标识出数据所在的生产者。

4.2 功能评价

JSMON 通过与各个具体的高性能计算机监控系统交互来获取高性能计算机资源的当前状态和资源使用情况。由于直接利用现有的高性能计算机监控模块，系统无需重新进行设计实现，减少了对系统的额外负担。

由于采用了缓存，系统在将用户任务提交之前，首先检查本地缓存，如果缓存失效，则直接将结果返回；如果缓存信息已过期，则通过安全证书和身份与全局生产者互相进行鉴别，

(下转第 311 页)

[2] Monien B, Preis R, Diekmann R. Quality matching and local improvement for multilevel graph-partitioning[J]. Parallel Computing, 2000, 26(12): 1609-1634

[3] Karypis G, Kumar V. METIS: unstructured graph partitioning and sparse matrix ordering system[R]. Department of Computer Science, University of Minnesota, 1995

[4] ANSYS FLUENT [OL]. http://www.ansys.com/Products/Simulation+Technology/Fluid+Dynamics/ANSYS+FLUENT

[5] ANSYS CFX [OL]. http://www.ansys.com/Products/Simulation+Technology/Fluid+Dynamics/ANSYS+CFX

(上接第 285 页)

通过安全的身份鉴别后，系统将监控任务提交到全局生产者，然后由全局生产者向下派发监控任务并最终得到返回的监控信息，将其进行处理后返回给用户。图 2 和图 3 是普通用户获得的 JSMON 系统监控结果的截图。

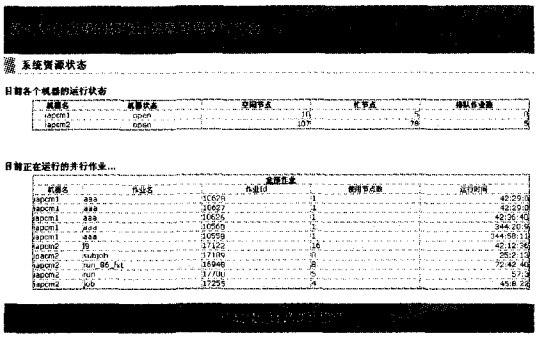

图 2 作业和资源运行状态

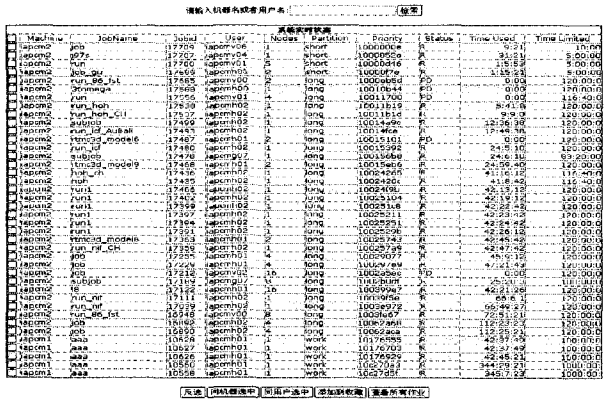

图 3 作业详细监控信息

4.3 性能评价

JSMON 获取所监控节点的资源运行状态和资源消耗情况。其对被监控节点的性能数据的采集是通过运行在节点上的资源采集 Client 来获得的，按照系统预定的周期，采集监控数据提交给生产者，以便上层应用使用。

由于采用了多线程对数据进行采集来控制传输时间，因此在采集部分和解析部分数据可以并行处理，而在网络条件较好的时候，传输也可以并行进行，这样数据的最大时间从之前的加和模式变成了最大模式，在对两台高性能计算机的监控测试的时候发现，数据监控时间开销从 0.2342s 下降到 0.1765s。

通过引入用户访问模式来减少整个监控系统的监控次数，以达到降低负载和网络带宽的目的，在下一步的工作中将对这部分工作进行测试。

结束语 监控系统对于高性能计算环境管理来说是至关重要的。通过监控可以及时发现产生故障的主机、网络设备、应用程序等，分析系统性能瓶颈，帮助用户在最短的时间内调整或恢复系统。基于 GMA 我们设计并实现了一套基于高性能计算环境的监控系统 JSMON，并投入使用。该监控系统具有如下特点：

1. 系统可移植性强。系统采用 Java 组件技术，使用通用的标准 XML 技术传输信息，使用 B/S 结构作为 Web 服务功能，使得信息的获取、产生、传输、显示标准化，从而具备了很好的跨平台性。

2. 监控数据的可定制性，根据用户定制作业监控信息，实现作业监控信息的筛选，用户得到自己最关心的数据信息。用户以及系统管理员可以根据作业监控系统提供的数据了解监控域内每个集群或各个主机节点的运行情况。

3. 系统具有方便可用性。作业监控系统针对系统管理员和用户提供不同的监视视图。

4. 系统轻量级。由于采用高性能计算机现有的监控接口，不需要重新配置一套监控服务，因此对于集群本身的监控负载最小。

5. 系统实时性好。由于园区局域网络传输速度快、数据传输稳定、处于同一管理域内以及相对安全等特点，较之互联网，监控系统需要考虑的因素相对简单，则实时性会较互联网的监控系统好。

下一步将对监控系统的性能和安全性做进一步的研究，使得监控系统更加完善。

参 考 文 献

[1] Blaise Barney. Introduction to Livermore Comput- ing Resources. [EB/OL]. https://computing.llnl.gov/tutorials/lc_resources/, 2011-08

[2] Network Weather Service[EB/OL]. http://nws.cs.ucsb.edu

[3] GT 4.0 Index Service: User's Guide [EB/OL]. http://www-unix.globus.org/toolkit/docs/4.0/info/index/user-index.html

[4] Buyya R. PARMON: a portable and scalable monitoring system for clusters[J]. Software Practice and Experience, 2000, 30(7): 723-739

[5] Sottile M, Minnich R. Supermon: A Highspeed Cluster Monitoring System[C]//Proceedings of Cluster. 2002

[6] 王翠，刘方爱. 基于 GMA 的教育资源网格监控研究[J]. 计算机工程与设计，2009，30(16)：3914-3917

[7] 黄鑫，刘梅梅，朱巧明，等. 网格监控系统研究综述[J]. 计算机应用研究，2009，26(1)：1-3，18

[8] A Grid Monitoring Architecture [EB/OL]. http://www-didc.lbl.gov/GGF-PERF/GMA-WG/papers/GWD-GP-16-2.pdf

[9] XML 系列教程[EB/OL]. http://www.w3school.com.cn/x.asp

[10] About RRD tool [EB/OL]. http://oss.oetiker.ch/rrdtool/, 2008

[11] 张宏海. 网格监控系统[J]. 超级计算通讯，2006，4(4)：43-55