NSTL全国服务体系监测平台的构建策略和关键技术*

王莉 郝春云 / 中国科学技术信息研究所 北京 100038
刘玉海 / 北京九瑞网络科技有限公司 北京 100101

摘要：文章提出NSTL全国服务体系监测平台的构建策略，包括监测指标、三级监测体系，以及从数据收集、跟踪、告警到存储的核心业务逻辑分析，对Shell编程、JMX框架、数据存储和报警机制等关键技术进行了描述。

关键词：NSTL监测平台，Shell编程，JMX框架，报警机制
DOI：10.3772/j.issn.1673—2286.2013.07.011

1 引言

国家科技图书文献中心（以下简称中心）经过十余年的发展，已经建成国内最大的公益性、普惠性科技文献信息资源保障与服务系统，并且通过地方建站、本地嵌入、接口、知识库以及集成揭示等多种方式辐射全国，形成既满足终端用户信息需求，又支持第三方科技信息机构的全国服务体系架构[1]。截至2012年年底，中心正式开通服务站41个，用户管理平台19个，覆盖全国除港澳台外的34个省市自治区[2]。这些已经开通的区域性站点究竟有没有达到预期的目的和效果？已经购买的资源对用户的价值到底有多大？随着服务规模的扩大，监测和评价的重要性日益突出。在这一背景下，2012年中心启动了全国服务体系监测项目，旨在对中心主站和全国站点展开服务性能和服务效果的全面监测，作为考量中心政策措施延续、改进和终止的重要依据。

对服务性能和效果的监测属于数字图书馆绩效评估的范畴，也一直是业界研究的热点问题。相关研究从早期偏重于投入逐渐发展到以用户为中心，关注服务影响和效果，并且出现了若干指标体系，涉及硬件设施和环境配置、信息资源建设和利用、服务获取、用户感知、教育等方方面面[3-8]。然而对于本项目而言，并没有一套完全适用的指标体系和评测方法，更重要的是理论联系实际，在参考借鉴已有研究成果和案例的基础上，结合NSTL自身特点，建立有针对性的、可操作的、持续有效的监测体系，通过来自全国不同站点相关指标的具体数据对比，支持站点间的自动测评。

2 构建策略

NSTL全国服务体系监测平台是一个工程性项目，围绕“NSTL各大系统和站点能否为用户提供稳定的信息服务”这样一个问题展开，在充分调研NSTL管理与运维需求以及相关技术的基础上，遵循不加重运维人员负担、不影响在线服务系统性能两大建设原则，采用整合已有的在线监测技术、挖掘日志文件、增强业务统计相结合的技术手段，快速实现对NSTL核心服务系统以及分布在全国各地服务站点的远程集中监测。

2.1 分级监测体系

（1）监测指标

指标体系的设立是评估的重要环节。项目组充分调研NSTL管理和运维需求，强调客观性、可操作性和可比性三项设计原则，从定量分析的角度提出操作系统、中间件、数据库、应用等四个层面的监测指标。前三个层面的监测主要是在线性能监测，为系统运维人

* 本文系2012年NSTL专项工作“NSTL服务系统监测及评价”的研究成果之一。

表1 监测指标

序号	监控项目	指标描述
操作系统层面：实时监测，可用于评估系统运行性能以及面对未来大并发量的承受能力。		
1	主机性能数据	包括CPU（使用率和平均负载）、内存使用率（swpd、free、buff、cache、paging rate、si、so等参数）和磁盘（磁盘空间使用率和磁盘I/O）。
2	系统连通性	包括探测次数、连通次数、最小响应时间、最大响应时间、平均响应时间、超时次数、平均超时时间、可用率等参数；对于连接失败的探测，进一步记录失败原因定期汇总。
中间件层面：针对WebLogic和Tomcat的实时性能监测，可用于评估服务运行的稳定性以及面对未来大并发量的承受能力。		
3	内存和线程使用情况	包括已经分配给队列的执行线程数、当前空闲的线程数、最长等待请求放入队列中的时间、队列中的等待请求数等重要信息。
4	数据库连接情况	包括连接缓冲池可包含的最大数据库连接数、当前使用的物理数据库连接数、当前使用的物理数据库连接数等信息。
数据库层面：针对Oracle和MySQL的实时监测与日志分析相结合，可用于评估服务运行的稳定性以及面对未来大并发量的承受能力。		
5	数据库性能数据	包括数据库吞吐量、数据库用户响应时间（系统服务时间+用户等待时间）等信息。
应用层面：日志分析，通过对各种信息服务量的客观统计，评估该项服务对用户的影响。		
6	网站流量统计	包括独立IP、独立访客、访问量、人均页面访问量、Online用户等参数。
7	检索量	对检索行为的计数。检索行为包括用户输入检索条件直接执行检索操作（一次检索结果查看中的翻页点击不计算在内），以及采用资源浏览的方式逐级点击的操作。
8	浏览量	对浏览行为的计数。浏览行为特指用户点击查看文献记录详细信息的操作。
9	全文传递量	包括全文传递、代查代借和全文直接下载的篇数。
10	参考咨询量	实时/非实时参考咨询完成数量，包括提问和回答这样一个完整的咨询过程才可计数。

员发现问题、解决问题服务。应用监测是对业务数据的分析统计，定期生成统计报告，为中心管理人员决策提供数据参考。详细监测指标如表1所示。

（2）三级监测体系

不同监测对象适用的监测粒度应该是不同的。从整体需求出发，NSTL全国服务体系监测平台划分为三级监测体系：

- Ⅰ级

NSTL中心站点各大服务系统，主要包括网络服务系统、回溯数据库、参考咨询、预印本、引文数据库和开放获取资源服务系统等，位于NSTL城域网内部，作为重点监测对象，列入Ⅰ级监测体系，实施全面监测，实时掌握系统运行情况。

- Ⅱ级

服务站和用户管理平台分布在互联网各地，硬件和操作系统级别的运维由本地机构负责，应用系统由NSTL网络管理中心集中运维，因此重点关注的是业务系统，如服务的连通性与服务的数量，侧重应用层面的监测。同时，从NSTL网管中心的角度看，虽然对操作系统的监测需求并不强烈，但是主机性能数据对于应用

系统的问题诊断是有帮助的，因此采用本地记录、定期收割分析的方式实现对这部分主机的监测功能。

● Ⅲ级

Ⅲ级监测体系针对NSTL订购的网络数据库以及集成的开放获取资源平台。这些资源/服务以Web方式提供，完全不可控，需要了解的仅仅是服务的连通性。这些数据对中心的资源采购及采购谈判具有重要价值。

2.2 监测平台框架

项目要求实现远程集中监测，整体架构采用B/S三层结构模式。服务器端完成“数据收集-跟踪-告警-存储”这一核心业务逻辑。数据呈现由Web服务器完成，用户采用浏览器方式访问Web服务器，完成与数据的交互。

对于服务站和用户管理平台而言，这些系统与NSTL主系统之间开通了特定访问权限，已经建立并使用特殊方式通信，属于可控对象。考虑互联网上数据传输的效率问题，采用C/S架构实现这部分主机的性能数据采集更为合适。因此，平台设计采用B/S与C/S混合架构，选用Java Spring开源框架实现。

图1展现了监测平台的业务逻辑，包括数据采集、跟踪、存储与呈现四个核心环节。

（1）数据采集

数据采集是从监测对象获得可用数据的流程，由采集器实现。监测对象和指标不同，采集方式也有所不同，一般来说，可以分为直接读取和推断测量两种方式。所谓直接读取，是指被监测对象提供指标数据的访问接口，或者直接记录指标数据，这时只需要拥有读取权限即可获得这些数据。例如，数据库监测中，NSTL应用系统采用了Oracle 10g数据库，Oracle提供了大量的性能视图，以计数和累计两种方式记录会话、实例、磁盘I/O、内存和系统竞争等相关信息。这些视图以v$开头，对它们进行关联和统计查询，可以从不同粒度上获得Oracle数据库的当前性能数据。所谓推断测量，是指针对被监测对象模拟某种操作，分析响应结果，进而得到需要的数据。典型的例子是对NSTL订购的网络数据库的可用性测试，基本思路是利用HTTP协议，访问指定URL，获取对方Web服务器返回信息，计算返回结果和发送请求的时间差，从而得到服务器的响应时间。这个时间可以作为系统可用性的推断测量。

（2）跟踪

跟踪是指通过消息触发机制从采集器得到测量数据，并按照事先定义好的格式规范输出的过程。常见的触发机制有轮询和监听两种。轮询是指跟踪程序按固定频率调用采集器，获得指标的当前值，主要用于对那些频繁变化的主机性能数据进行实时监测，例如每分钟读取主机CPU利用率。监听是指采集器将其自身注册为被监测对象的事件监听程序，跟踪程序在事先不需要采用轮询方式周期性查看数据指标，当关注的事件发生时（例如数据超出设定阈值），监听程序能够立刻知晓，并执行相应的任务。对WebLogic中间件的监测就采用了这种方法，相关技术细节见本文“关键技术-JMX框架”部分。

（3）存储

收集到的指标数据采用“名称-值”形式表示，非常简单，其最大的特点在于时间相关性，随着时间的流逝数据量不断增长，越积越多，而应用最关心的总是最近一段时间的数据。处理这类数据最常见的工具是RRDTool，其核心思想是RRD数据库（Round Robin Database）。简单地说，在建立数据库时指定一个循环时间，当到达循环时间的时候，数据库会自动覆盖最老的数据，从而形成一个环形的数据区域[9]。由于RRDTool采用的是基于文件的数据存储，与系统的集成度不好，而且也不能很好地支持未来的数据挖掘与分析，因此本项目并没有直接集成RRDTool工具，而是借鉴RRD思想，采用纯Java编程在MySQL数据库中实现，主要的技术点是历史数据的

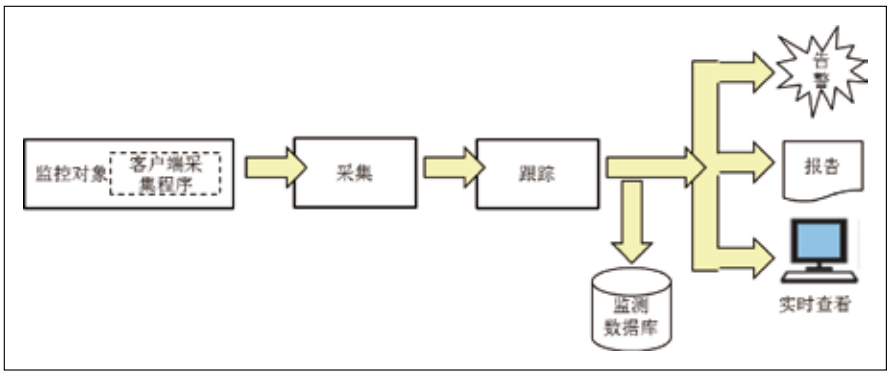

图1 监测平台业务流程图

归档机制，详见本文“关键技术-数据存储”部分。

（4）呈现

数据指标的呈现主要包括告警、实时显示和报告三种方式。告警是当某项指标数据超过设定阈值时，及时提醒相关人员注意，除了在控制中心实时显示之外，常用方法有电子邮件、IM和短信分发。具体的报警机制将在本文“关键技术”中深入讨论。实时显示是以Web方式近乎实时地显示特定指标数据，多采用图、表形式，强调可视化。报告则是从监测数据库中定期导出指定时间范围内的指标数据，通常规定了固定格式。本项目采用JFreeChart类库针对不同的指标类型生成不同形式的图表（例如用折线图展示CPU利用率的趋势变化），绘制的图表与PDF和EXCEL关联输出形成报告，能够很好地满足应用。

3 关键技术

3.1 Shell编程

NSTL全国服务体系集中采用AIX和Linux两种操作系统，且版本统一。AIX和Linux有很多内置的性能数据收集工具（例如vmstat、ps、sar、iostat），可以直接采用执行Shell脚本命令并解析响应的方式实现。

图2给出了采用SSH远程调用Shell命令的基本框架，用于B/S架构实时采集主机性能数据。关于SSH的Java实现已经有很多成熟的方式，例如JSch[①]、Ganymed SSH-2 for Java[②]、Apache Ant中的SSHEXEC和SSHXCUTE[③]等，可以在一个SSH连接会话中完成多项指标数据的实时抓取任务，持续监测则通过任务管理以定期轮询的方式实现。整个方法简单有效，并且对服务器开销较低。图3是本项目实现的对NSTL中心站检索服务器CPU使用率的折线展示。

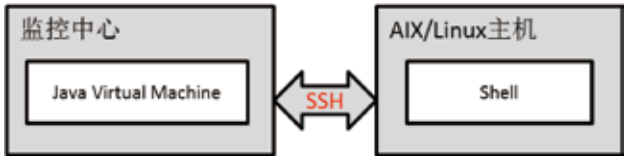

图2 远程调用Shell命令（B/S架构）

分布在互联网各地的服务站和用户管理平台采用客户端代理程序完成本地主机数据的实时抓取与记录任务，由监测中心定期收割日志文件。图4给出了这种本地执行Shell脚本文件的C/S采集架构。其中，采集到的指标数据以.log后缀命名，采集过程中产生的错误也保存在相应目录下，以.err后缀命名。在该模式下，持续监测是通过在被监测主机中设置cron定时任务实现的。

3.2 JMX框架

JMX（Java Management extensions）是SUN公司提出的一套管理框架，其核心是MBean，简单地说，MBean就是一个Java类，提供了控制和监视资源的管理

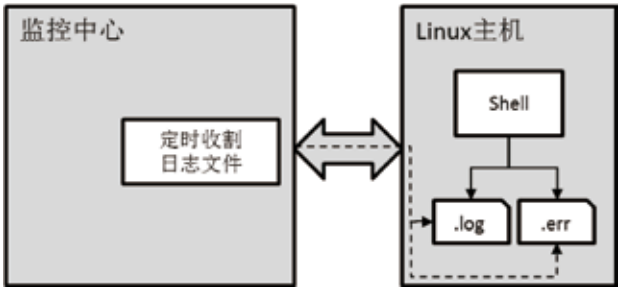

图4 本地执行Shell脚本文件（C/S架构）

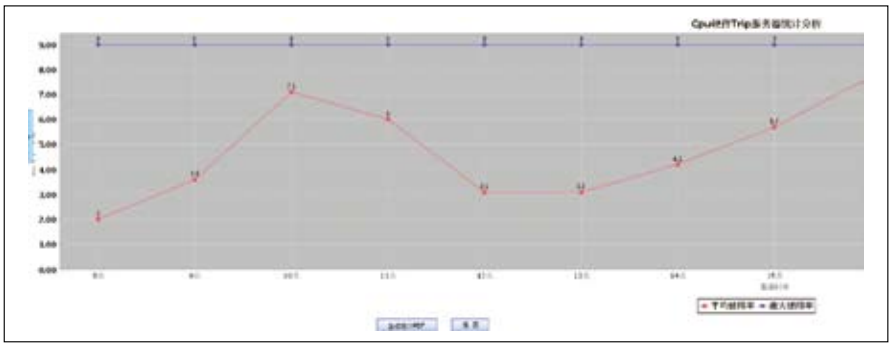

图3 CPU使用率折线图（系统截图）

① http://www.jcraft.com/jsch/
② http://www.ganymed.ethz.ch/ssh2/
③ http://code.google.com/p/sshxcute/

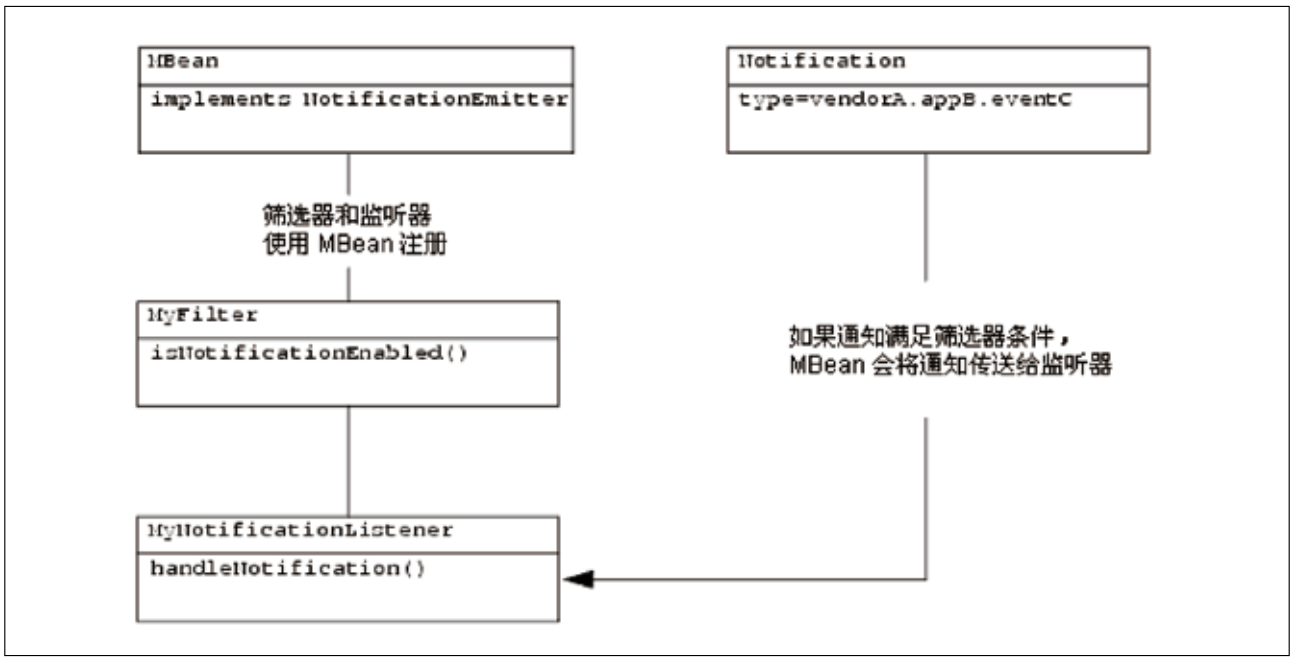

图5 接收来自MBean的通知[10]

图6 实时监测到的线程信息（系统截图）

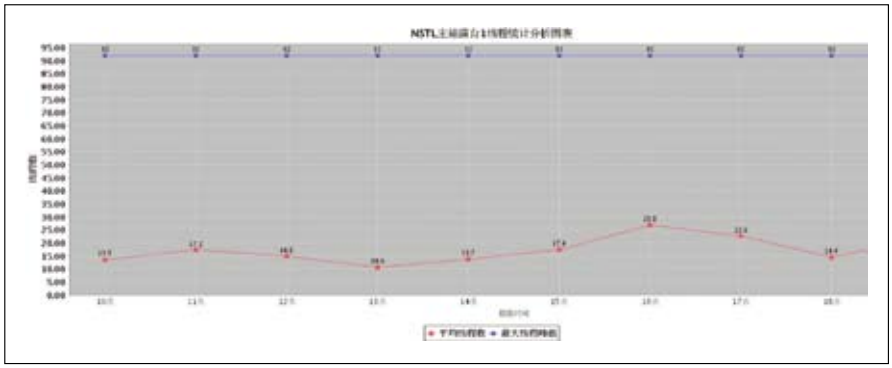

图7 线程变化趋势（系统截图）

能力[10]。图5显示了通过MBean传递消息的基本模型。

首先，通过MBean的javax.management.NotificationEmitter接口发出通知。第二步，创建并注册监听器，采用

javax.management.NotificationListener.handleNotification()方法实现收到通知时执行的操作。第三步，创建并注册监听器的筛选器，仅接收那些需要关注的事件。

本项目中对消息中间件WebLogic和Tomcat的性能监测就是采用JMX框架实现的。Weblogic的指标参数非常多，主要关注服务器当前状态、服务器监听端口上的活动、内存和线程使用情况、数据库连接等运行时的统计信息[11]，图6是在线捕获的线程基本信息，图7则

以折线图形式展示了线程的变化趋势。

对Tomcat的远程监测基本原理与WebLogic监测相同，不同点在于WebLogic监测采用的是WebLogic Server类从远程JMX客户端发起连接，对Tomcat则需要通过JDK中的类进行JMX远程管理[12]。

以上讨论的是对基于JMX框架的应用系统的监测。如果要监测非JMX架构的Java应用，则需要添加JMX支持，可以采取插装Java类和资源的方法实现[13]，不需要修改源代码即可为应用添加监管能力。对于基于Spring的应用程序，还可以利用Spring JMX扩展特性为方法和属性透明地增加通知事件，这种方法虽然需要修改源代码和数据库结构，但是并不会搞乱Java对象[14]。NSTL网络服务系统是基于Spring框架开发的，理论上可以添加JMX监测能力，例如，可以在文献传递环节为下订单这一行为添加JMX监测能力，实现对文献订购量的实时观测。由于NSTL应用系统本身非常复杂，在本项目实施过程中并没有采用该方案，业务数据的收集由应用系统以日志文件和数据库的方式记录下来，由控制中心定期抓取。

3.3 数据存储

数据存储由XML文件和MySQL数据库两部分组成，XML文件存储采集相关的配置信息，包括数据源、指标类型、单位、采集频率等，支持监测任务的管理；MySQL数据库存储收集到的指标数据。虽然指标数据本身形式简单，但是随着时间的流逝数据量会不断增长，并且，在观察较长时间的指标变化时并不是数据越多就越清晰，例如，查看过去1年文献检索量的变化趋势，使用月平均检索量更为合适。因此在指标数据的存储中，重点讨论历史数据的归档机制。

（1）保存最近收集的细粒度数据（原始指标数据），按照特定指标类型分别存储，每个值都记录与之相关联的时间戳和数据来源。

（2）使用最大、最小和平均三种方式对原始数据聚合，按小时/日/周/月/年的层次分表存储，形成不同粒度的数据集。需要注意的是，并不是所有指标都按照小时-日-周-月-年这样一个层次归档，需要为特定指标类型设定特定的归档粒度。

（3）按照先进先出的原则，循环清除时间较长的原始数据，以保证数据库查询效率。简单地说，假设原始数据保留时间为60天，当数据采集到第61天时，新采集到的数据将覆盖第1天的数据，保证表存储空间不会无限制增长。同样地，特定指标类型也应该根据实际需要设置特定的保存期限。

3.4 报警机制

控制中心将接收到的数据写入数据库中，同时与预设阈值进行比较，如果超出设置范围，则启动报警机制，这里主要讨论消息分级和消息传递策略问题。

（1）消息分级

报警消息按照来源指标数据可以分为操作系统、中间件、数据库、应用四个层面，但是这种归类方式是非常粗糙的，需要重新按照重要性设置不同级别，加强对重要报警的监测，及时发现安全隐患。

判断一条警示消息的重要程度，不仅与监测到的数据绝对值相关，更与该值的变化趋势关系密切。例如，CPU平均负载描述系统当前的繁忙程度，它反映了系统中所有等待运行的进程数，但是进程数量多并不一定会影响服务器的正常工作，监测的重点应该放在变化趋势上，突增或突减都需要警示，并且应该根据变化幅度大小设置不同的重要级别。因此，判断一条告警消息的重要性时需要采用绝对值和变化值相结合的方式。

（2）消息传递策略

很多时候系统自动监测到的异常情况会持续到问题的解决，如果运维人员没有及时得到通知，并进行干预，那么告警消息可能变成一种干扰，直接影响监测系统的性能；采用邮件告警的情况下，可能会导致邮件服务器被列入黑名单；采用短信告警则会迅速侵吞手机存储空间，影响正常使用。因此，合理分发也是报警机制中要重点考虑的问题，需要结合固定分发频率、指标数据的变化趋势综合考量。

本项目采用的基本分发策略是：第一，报警行为由预设阈值触发，阈值可以是绝对值，也可以是相对值。第二，相同的告警消息仅发送3次。需要注意的是，这里的“相同”并不是同一次监测行为产生的同一条消息，而是分发告警消息之后，紧接着下一次监测得到的指标数据触发相同的告警。

3.5 运行效率分析

“监测系统启用之后不能影响在线服务系统的性能”是指导整个项目实施的重要原则。监测行为可能对

监测对象性能产生影响是在数据采集环节，项目在确定技术路线时就充分考虑这个问题，并在具体实施中采取若干技术手段减低数据采集对监测对象系统资源的占用。

（1）Agent和Agentless在主机性能数据采集中各司其职，以适应“NSTL主站-服务站”模式。

Agent监测方式也就是在框架设计部分提到的嵌入总体B/S框架中的C/S部分，主要完成对分布在互联网各地的全国服务站点主机性能数据的采集工作。Agentless监测方式针对的是NSTL城域网内中心主服务系统。采用不同的实现模式重点考虑的是对网络带宽资源的消耗问题。中心主服务系统部署在NSTL千兆城域网内，网络带宽资源充足，完全可以支持实时采集数据未经压缩和汇总就直接传输给监测服务器。服务站分布在互联网上，各个站点网络条件差异很大，而NSTL网管中心对这些站点的实时监控需求并不强烈，因此使用Agent采集并记录日志，由监控中心定期收割。

（2）高效利用操作系统内置工具，以Shell脚本方式完成主机性能数据采集任务。

Agentless监测对监测主机的影响主要在建立访问连接时产生的，其最大优点就是占用系统资源少。主站系统采集模块在实现中采用一个SSH连接会话完成多项指标数据的实时抓取任务，持续监测通过任务管理以定期轮询的方式实现。整个方法简单有效，并且对服务器开销较低，单模块测试中观察CPU占用率小于1%。

Agent运行在被监测端，对其性能和可靠性有较高的要求。为保证主机和监测应用都能正常运行，大多数软件采用效率比较高的C代码，也有采用JAVA实现的（如IBM）。相比之下，JAVA程序会占用较多内存，对主机系统资源的开销较大。为了快速构建，主机性能采集模块高效利用操作系统内置工具，以Shell脚本方式实现。

Shell是一种解释型语言，尽管从执行方式方面考虑，解释型程序的效率比不上C语言类的编译型程序，但脚本具有简便快捷的特点，并且在NSTL应用系统的长期运维工作中，已经形成一些监测脚本文件，其运行效率和监测能力是经过实践检验的。将这些经验集成到监测工作流中，加快了系统建设进度。

（3）按需制定合理的监测指标集，将中间件监测行为对监测对象的影响控制在一个可接受的范围内。

本项目对中间件进行监测的主要目的是为了掌握其运行的健康状况，出现问题能够及时发现，至于发现问题之后的诊断排查则需要配合更复杂的工具完成（如profiler）。目前对主站前台系统配置了会话、线程、Java虚拟机三类基本监测指标，技术上采用JMX框架编程实现，在监测对象中将会启用Java进程，因此在性能分析时主要采用观察Java进程占用系统资源的方法，对比监测系统运行和不运行两种状态下的表现。测试结果显示，启用和不启用监测系统并未对监测主机上Java进程的CPU使用率和内存使用率造成明显变化，CPU和内存平均使用率的变化小于1%。

（4）采用若干技术细节提升中间件监测模块的执行效率，降低对监测对象的性能影响。

中间件监测模块基于JMX框架实现，在具体编程时主要采用单一连接和一次请求多个属性的方法提升执行效率。其中，单一连接即为采集器保留一个单独连接，以此减少轮询并重新连接对目标主机的开销。实践中证明该方法对性能提升的效果明显。

4 结语

本文从NSTL管理需求入手，结合现有服务体系，提出采用B/S和C/S混合架构，基于Java Spring开源框架建设远程集中监测平台。目前，该平台完成了数据采集、跟踪、存储与呈现等核心功能的开发测试，部署在主系统测试环境（上线前测试）的后台管理框架中，由一台IBM X3850机器支撑运行。试运行阶段启用了对主站网络服务系统生产环境的操作系统、中间件、数据库和应用四个层面的监测；针对外地站点仅启用了通断监测功能。虽然该平台并不具备评价功能，但是通过最常见的统计方法，可以快速了解情况、发现问题；同时，在统一的指标体系下，各种统计分析数据汇聚形成监测报告，为中心决策提供重要而客观的数据支持。

参考文献

[1] 袁海波.改革创新 开放联合 谱写科技文献共建共享新篇章[M].国家科技图书文献中心成立十周年文集.北京:科学技术文献出版社,2010:15-18.
[2] NSTL网管中心.2012年NSTL年终工作总结[R].2012.
[3] 向远媛.图书馆评估研究综述[J].新世纪图书馆,2011(1):59-61,16.
[4] BERTOT J C. Library Network Statistics and performance Measures: Approaches and Issues [J]. Library Quarterly, 2001(11): 229-230.
[5] GONÇALVES M A, MOREIRA B L, FOX E A, et al. "What is a good digital library?" - a quality model for digital libraries [J]. Information Processing & Management, 2007, 43(5): 1416-1437.
[6] KHOO M, DONAHUE R A. Evaluating digital libraries with webmetrics [C]// Proceeding JCDL '07 Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries. Vancouver, British Columbia, Canada: ACM Press, 2007: 484-484.
[7] The COUNTER code of practice for e-resources: release 4 [EB/OL]. (2013-02) [2013-0311]. http://www.projectcounter.org/code_practice.html.
[8] 曾照云.LibQUAL+TM—图书馆服务质量评价研究综述[J].情报杂志,2009,28(12):95-98.
[9] rrdtutorial [EB/OL]. [2013-04-26]. http://oss.oetiker.ch/rrdtool/tut/rrdtutorial.en.html.
[10] 使用JMX开发可管理的应用程序[EB/OL]. [2013-03-11]. http://edocs.weblogicfans.net/wls/docs92/jmxinst/understanding.html#wp1100632.
[11] 最佳实践：监听 WebLogic Server 事件[EB/OL]. [2013-04-26]. http://edocs.weblogicfans.net/wls/docs92/jmx/notifications.html.
[12] Apache Tomcat 6.0: Monitoring and Managing Tomcat [EB/OL]. [2013-04-26]. http://tomcat.apache.org/tomcat-6.0-doc/monitoring.html.
[13] WHITEHEAD N. Java运行时监测 第2部分:编译后插装和性能监测[EB/OL]. (2008-08-19) [2013-03-11]. http://www.ibm.com/developerworks/cn/java/j-rtm1/.
[14] DUGUAY C.扩展Spring的JMX支持[EB/OL]. (2005-11-24). [2013-03-11]. http://www.ibm.com/developerworks/cn/java/j-springjmx/.

作者简介

王莉，中国科学技术信息研究所研究员，全国信息和文献标准化技术委员会技术协作分技术委员会委员。研究课题：数字图书馆。E-mail: wangli@istic.ac.cn
郝春云，中国科学技术信息研究所高级工程师。E-mail: chyhao@istic.ac.cn
刘玉海，北京九瑞网络科技有限公司项目经理。E-mail: beijixing830@163.com

The Construction Strategy and Key Technologies of NSTL Monitoring Platform

Wang Li, Hao Chunyun / Institute of Scientific & Technical Information of China, Beijing, 100038
Liu Yuhai / Ninemax Network Technology Co., Ltd., Beijing, 100101

Abstract: This paper proposes the strategy of NSLT Monitoring Platform, including monitoring indicators, three-level framework, and the core business logic of data collection, tracking, alarm and storage, and describes key technologies such as Shell programming, JMX framework, data storage and alarm mechanisms.

Keywords: NSLT monitoring platform, Shell programming, JMX framework, Alarm mechanisms

（收稿日期：2013-05-14）