面向网格的性能工具的研究

时培植　李三立　都志辉　黄震春
（清华大学计算机系高性能研究所　北京 100084）

Research for Grid-Oriented Performance Tool

SHI Pei-Zhi　LI San-Li　DU Zhi-Hui　HUANG Zhen-Chun
(Institute of Computer High Performance, Tsinghua University, Beijing 100084)
spz@tirc.cs.tsinghua.edu.cn

Abstract　Performance is one of the key problems in either high performance computing or GRID application. Performance data must be collected and analyzed for co-allocating resource efficiently, obtaining high performance and fault toleration. Furthermore, with the development of Internet and GRID, the exchange of data between virtual organiztions is becoming more and more important, and the type of performance is increasing following the increasing of the resource type, which requires a proper representation of the performance data. This paper does some research on the collection, analysis and representation of the performance data, and presents a Grid-oriented performance tool prototype: THGPT, which can achieve the runtime performance data, describe the data in XML, and implement a browser-based visualization tool of performance data analysis.

Keywords　Performance tool, Performance data, Collect, Analyze, Represent, Grid

1　引言

“性能”的含义，已从单纯的计算速度，扩展到为一个包含计算、网络、磁盘读写速度甚至精密仪器性能的概念。性能也是网格应用程序最关心的问题之一，网格的工作流程中各个步骤都要考虑到性能问题，而对性能的研究则离不开与性能相关的数据。例如，Information service 应该能够提供性能数据，如资源当前负载状况，而不仅仅是该资源当前是否可用，以供资源协同分配、任务调度的需要；而 Fault detection service 也需要性能数据来判断各个资源是否工作正常。同时，性能还是传统的分布式计算和高性能计算的主要目的，性能数据也是负载平衡、消除性能瓶颈等工作所不可缺少的。

传统的性能工具应该能够采集各种性能数据，供程序员对其进行分析（这种分析应该是可视化的），以发现程序中的性能瓶颈，发掘性能潜力，最终提高性能。而 Grid 性能工具除了要能对性能数据进行采集和分析之外，虚拟组织内部甚至虚拟组织之间的数据交换不可避免；随着资源种类在不断增加，相应的新性能类型也在不断出现。因此，如何使性能数据能够用于不同场合和异构平台，如何以现有的格式描述未知的性能，如何将来自于跨越大范围网络的各种异构的平台、资源的数据以统一的格式进行描述，即，如何表示性能数据，已成为面向网格的性能工具的另一个必须解决的问题。

网格性能工具目前还没有成熟的产品，甚至连这个概念也尚在讨论之中。本文对性能数据的采集、分析和表示做了研究，介绍了一个面向 Grid 的性能工具的原型—THGPT，它可以获取应用程序的运行时性能数据（包括一些资源负载和程序状态），并实现了一个可以在浏览器上运行的可视化性能数据分析工具。最后，THGPT 还提出了一种性能数据的 XML 表示，使不同平台上的应用程序对性能数据能够以统一的格式进行描述和交换。

2　相关工作

由清华大学计算机系承担的“先进计算基础设施北京上海试点工程（简称‘清华 ACI 系统’）”已经于 2001 年 6 月通过了由著名计算机院士和专家组成的鉴定委员会的鉴定和验收委员会的验收。在清华 ACI 系统中，清华大学研制的高性能计算机“THNPSC-2”与上海大学研制的高性能计算机“自强 2000”等六个应用结点连接后，可以构成跨地区、跨学科的“虚拟实验室”研究环境。本文中的研究就是在该 ACI 系统上展开的。

Global Grid Forum 也对 Grid 中的性能分析做了很多研究，并提出了一种性能数据的 XML 语言表示[3,4]。这种表示方法初步解决了对性能数据的跨平台互操作性问题，但由于它对每一种性能都要定义相应的语义之后才能表示，因此只适合表示有限的几种性能，可延展性不强。

Globus 的 NWS[5]系统可以监测和预测 Grid 系统内的网络性能，但同样缺乏对其它性能的描述方法。

MPICH 软件包中包含一个性能评测工具 Jumpshot[6]，它可以对 MPI 应用程序的流程进行监视，同时也实现了可视化的界面。但它不能采集资源信息，而且只能用于机群环境。

Wisconsin 大学的 Paradyn 系统[7]在应用程序执行的过程中，动态地监控程序的运行状态，以 3W（When, Where, Why）为依据在二进制代码中搜索性能的瓶颈，并试图自动地消除瓶颈，但是它只能对特定平台上的软件进行优化，同时它也是个封闭的系统，并不面向 Web 和网格应用程序。

THPTii[8]是 THGPT 的早期版本，可以采集和分析 CPU 负载和内存利用率等性能数据，但只能用在机群系统中，不具备性能数据表示、Web 可视化分析模块等功能。

时培植　博士生，主要研究方向为高性能计算性能分析。李三立　清华大学计算机系教授，工程院院士。

3 THGPT 总体介绍

THGPT 系统由三部分组成，按照其各自的功能分别称为性能数据采集子系统、性能数据表示子系统和性能数据分析子系统。原始的性能数据是由性能数据采集子系统来采集的，之后通过表示子系统来将它书写成特定格式的 XML 文件，最后由分析子系统来进行可视化，提供给用户进行性能分析。如图 1 所示。

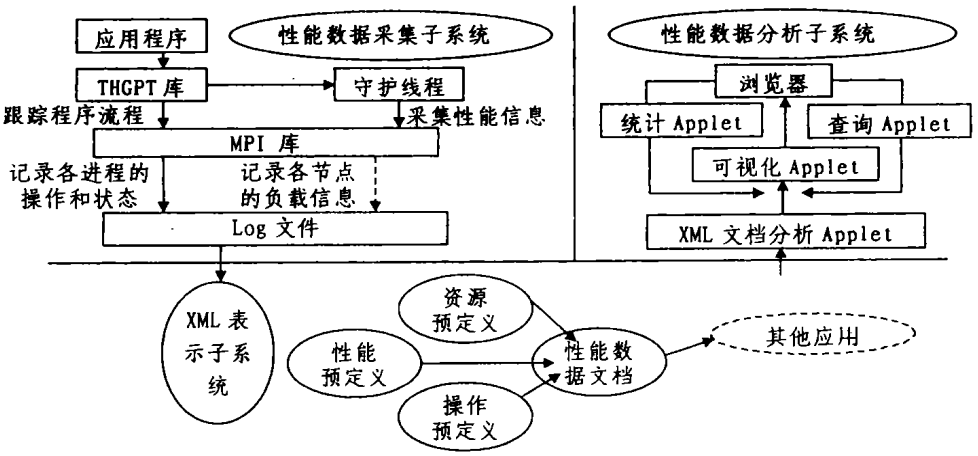


图 1 THGPT 框架示意图

4 性能数据的采集

网格系统的 Information service 必须提供资源的当前信息。而这些信息在进行性能分析或性能预测时往往还不够，还需要相应资源在过去一段时间内的性能信息，甚至需要知道这些资源在这段时间内的行为。

对于采集资源的性能数据，目前已经有了若干成型的软件。如 Globus 的 NWS 系统可以采集(并分析)网络的带宽和延迟等性能数据，清华大学计算机系的 CSIP 系统可以采集计算资源上的进程信息、CPU 信息等。THGPT 目前以 CPU 负载和内存利用率为例，实现了性能数据采集的一个实例原型。它采用了 THPT 中的性能数据采集模块，因为该模块不仅能采集资源的负载信息，还同时记录相应时间内运行于该资源上的应用程序的操作和状态。这些操作和状态显然与资源的负载信息密切相关，能为性能分析提供更多的线索。考虑到空闲时采集资源的性能数据意义不大，对性能数据的采集与应用程序同时启动同时终止。(现阶段它仅能对基于 MPI 平台--包括 MPI 的各种实现版本--的应用程序进行操作数据采集)。

另外，性能数据的采集操作本身必须是低开销的。如果采集操作的开销大到了对资源的性能有明显影响的地步，那么这样的采集就失去了意义。THGPT 的性能时间采集子系统采用了多种手段以降低开销。

4.1 负载信息的获取

从图 1 中可以看出，采集负载信息的工作是由一个线程(Sysinfo Daemon)来完成的，该线程在应用程序启动时，由应用程序主线程派生。相对于进程，线程的开销要小得多。Sysinfo Daemon 启动后，作为另一条指令流与应用程序并发工作，采集性能数据。

在 Linux 系统中，每个正在运行的进程所占用的资源情况、整个系统当前内存总占用情况和开机以来 CPU 时间片的总分配情况，都保存在系统目录/proc 中。Sysinfo daemon 在绝大部分时间里睡眠以降低开销，只是每隔一定时间会醒来，然后访问/proc 目录中的相应文件，经过计算，得到该段时间系统的资源利用情况，包括 CPU 利用率、核心态 CPU 时间片和内存利用率，等等。

4.2 应用程序状态的跟踪

资源的性能情况往往和资源在该时刻的行为密切相关，因此 THGPT 对应用程序行为进行了跟踪。目前 THGPT 只能对指定的一些状态进行记录，如发送、接收、计算，等等。“状态”的跟踪是通过对“事件”的记录实现的。

定义 1 状态(State) 应用程序在运行过程中所执行的发送接收数据、阻塞等等行为，在本文被称为“状态”；

定义 2 事件(Event) 应用程序在两个状态之间切换时，就认为发生了一个事件。例如，在应用程序从计算状态切换到发送状态时，一个“开始发送”事件就发生了。反之，当应用程序结束发送状态，又回到计算状态时，一个“结束发送”事件就发生了。

除了计算之外，所有的状态都对应着两个事件，分别代表该状态的开始和结束。另一方面，只要记录了相应的一对起始和结束事件，那么在这两个事件之间，应用程序就处在对应的状态。而对于不在任何一对事件之间的程序段，就认为它处在“计算”状态，如图 2 所示。

THGPT 捕获程序中的某些函数调用，在该函数开始时记录一个开始事件后再执行函数功能，函数执行结束后，记录一个相应的结束事件后再返回。MPI 提供的 Wrapper 功能[1]使 THGPT 不必修改 MPI 应用程序的源代码就可以截断 MPI 调用，记录下计算、MPI 通信和阻塞等状态。

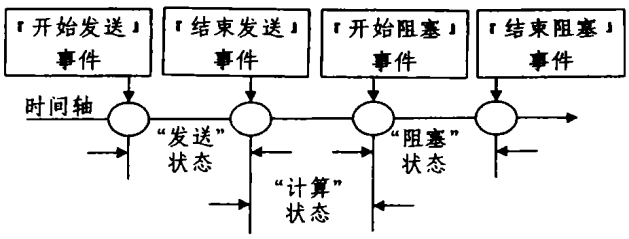


图 2 状态与事件示意图

5 性能数据的表示

Grid 系统中，性能数据有可能来自计算机群、存储服务、

网络设备，甚至目前还没有出现过的科学仪器设备，这些性能数据也有可能被各种各样的应用程序所使用。如果任何一对性能数据的生产者和消费者都需要事先进行复杂的交流以确定数据的格式，毫无疑问这将极大地降低系统的效率。因此，最好的办法就是将这些来自各种资源、描述各种性能的数据都以统一的、可跨平台的格式描述。

虽然在异构的平台下进行性能数据的交换，除了以统一的格式对性能数据进行描述外，还离不开性能数据的发布、目录、费用、传输、安全等方面，但这些方面的研究与性能数据本身没有关系，在 Information service 或者其它服务、应用中，完全可以将性能数据与其它数据等同看待。因此，这里只对性能数据的表示进行了研究。

性能并不仅仅是指计算性能，因此这种表示必须满足以下几个条件：1)它必须是自描述的，可以描述目前的多种性能；2)它必须是可扩展的，因为新的性能类型随时可能出现；3)它必须是跨平台的，因为不同的平台对于数据的理解很可能会不同。

因此，我们最终选择了 XML 作为对性能数据进行描述的语言，其优势在于：

1)*设计新的标记语言* XML 允许人们开发与特定领域相关的标记语言，这使得该领域的人们可以交换数据，而不必担心接收数据的人是否装备了特定的软件。

2)*自描述性* XML 在底层使用的是非常简单的文本格式，丢失一部分也仍然可以保持语义；从高层来说，XML 是自描述的，人们很容易从数据中读取它们的真实含义。而 XML 又是非常规范的，不易误解。

3)*应用间交换数据* XML 是通用的并且易于读写，它非常适合作为在不同的应用之间交换数据的格式。

4)*结构化和集成的数据* 用户可以指定一个定义了文档中的元素的词汇表，还可以指定元素之间的关系。

由此可以看出，XML 语言完全满足表示性能数据的要求。

我们在附录一中给出了 XML 语言的文档类型定义(DTD)，它规定了 XML 文档的语法，即描述性能数据必须满足的一些要素，例如必须列出该性能的名称、该资源的名称及唯一位置、该数据所描述性能的起始时间和结束时间，等等。

前面已经提到，由于新的性能类型随时可能出现，因此性能数据的表示不能仅仅局限于只表示某一种性能，这很不利于扩展，如文[3][4]，为每种性能都在 DTD 中定义相应的元素，这样不利于增加新的性能类型。THGPT 在这两个方面考虑到了这一点。

从图 2 中可以看出，性能数据的 XML 文档由四部分组成，其中只有性能数据文档是从传感器采集来的性能数据所生成的，其它三个文档则分别包含了资源、性能和操作的若干预定义。其中资源预定义包括 Information service 提供的各种资源，性能的预定义则定义了 CPU 性能、磁盘读速率、网络带宽等，操作预定义(是个可选项，因为并非所有应用都需要资源上的行为信息)则定义了发送、广播、阻塞等操作。这些预定义可以保证性能数据文档中所涉及到的资源、性能、操作名称的合法性，而将它们分开则有利于扩展。新的性能、操作或资源出现时，只需要在相应的预定义文档中简单地增加几个元素，即可书写相应的新性能类型的性能数据文档，完全不必修改整个文档的结构。

另外，THGPT 抽取了所有性能数据的共性，采用了精心设计的格式，可以对所有类型的性能数据都以统一的格式描述。它们的差别仅仅在性能预定义文档中体现，而在性能数据文档中的格式则完全相同。这种共性归纳起来有三点：1)所有资源的性能都有一个理论上的确定峰值；2)所有资源的性能都有一个确定的度量；3)所有资源在某时刻的性能都有一个与峰值的确定百分比，时间戳也应记录下来。

因此，THGPT 将与相关性能类型对应的峰值、度量都写在性能预定义文档中，而性能数据文档中则只记录某时刻的性能。DTD 中相应的两处关键定义如下：

```
<! ELEMENT PerfDataList (PerformanceType, FullPerf,
    Measurement, PerfData+)>
<! ELEMENT PerfData (StartTime, EndTime, AveragePe-
    rfRatio, MaxPerfRatio?, MinPerfRatio?)>
```

第一句定义了每种性能数据所必需的组成部分，包括一个性能类型描述、峰值、度量单位，以及若干性能数据单元。第二句则定义了每个性能数据单元的结构，应该包括采样的起始时间戳、结束时间戳，以及这段时间内的平均性能(与峰值的百分比)，还可能包括这段时间内性能的最高值与最低值。

该性能数据文档除了用于性能分析之外，还可以提供给其它应用(如资源协同分配、容错服务等)。统一的 XML 表示使性能数据在各种应用中共享成为可能。

图3是用 Internet Explore 5.0对该 XML 文档的显示情况。

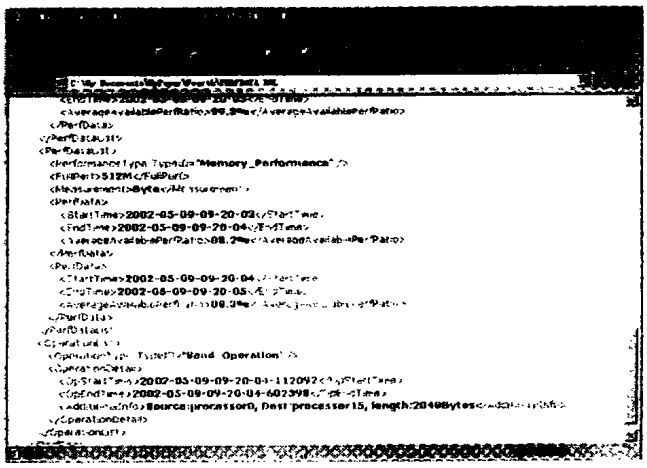


图3 IE5.0显示范例性能数据的 XML 文档

6 性能数据的分析

在性能工具中，性能数据的采集和表示最终是为了供程序员进行性能分析。我们采用了 JAVA applet 技术来对性能数据进行可视化。目前我们还没有完全意义上的网格应用程序，因此性能分析部分所用的是一个涉及多个计算节点资源的并行计算程序：利用分子模拟方法中的 Monte Carlo 法研究狭缝炭孔内吸附的临界特性算法的运算实例[2]。

性能分析部分的核心是它形成的性能分析图(即画面中从左下开始的部分，占据了窗口的大部分面积)，本文中称它为 PAG(Performance Analysis Graph)。PAG 被划分成 np 个子图(np 即参与应用程序运行的计算节点资源的个数)，每个子图表示相应计算节点资源上某段时间内的应用程序行为和资源负载，子图包括上半部分以矩形方式表现的进程行为重现图 RSRG(Resource Stauts Recurring Graph)和下半部分以折线方式表现的应用程序占用系统资源图 PRG(Perfor-

mance Record Graph)。应用程序在某时间内的状态表现为相应PSRG内的矩形,以不同的颜色区分,矩形在时间轴上的位置对应着此状态的持续时间。而PRG中也有若干条折线,同样以颜色区分,代表该资源在这段时间内的各种性能记录。窗口的上部是若干标签,列出了颜色和状态、性能的关系。

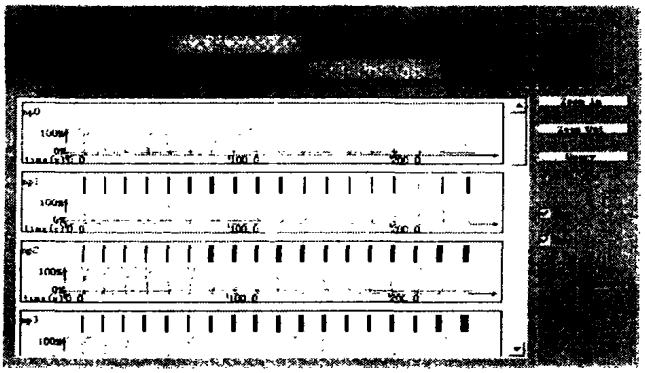
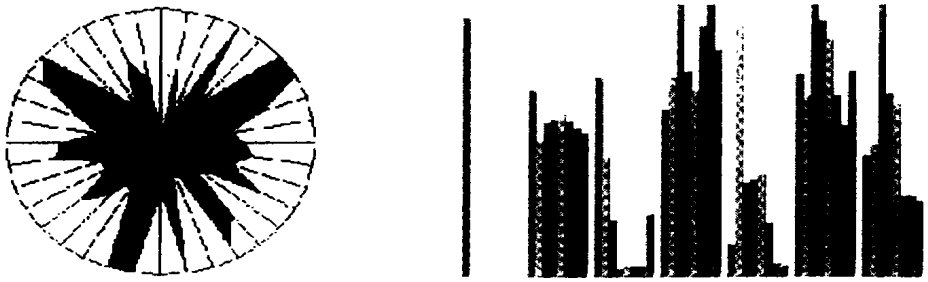


图4 性能分析子系统的可视化界面

用户除了直观地观察可视化结果图之外,还可以获取若干以文字表示的额外信息,如某一时刻结点资源所处的详细运行状态(例如,对于一个Send状态,用户就可以知道Send动作发生的时刻、传递信息的长度、所花费的时间等)、该资源的硬件峰值性能和某时刻的性能数据。

从图4中可以看出,这个应用程序的负载均衡程度很高--大部分时间内,所有节点的CPU资源都被充分利用,利用率达到或接近100%。但是从图中同样可以看出,一旦切换到通信状态(即图中的有色矩形部分),该计算节点资源的CPU利用率就会急剧下降。

另外,单纯地罗列出所有性能数据的可视化结果,有时对性能分析没有太大意义,因此THGPT提供了若干查询和统计工具,可以查询指定的一个或多个资源的一种或多种性能数据子集并可视化之,或者对某种性能数据进行统计,如某节点资源上某段时间内的平均负载,所有资源的平均总通信时间,等等。这将有助于程序员发现性能的瓶颈所在,进而最终提供满意的性能。图5(a)是某计算节点资源在进行Laplace变换时,某段时间内CPU性能随时间变化的雷达图表示;图5(b)是在一次LU分解中,所有参与计算节点资源在某段时间内CPU性能统计平均值的直方图表示,各个方柱的高低代表该节点CPU利用率的高低。从统计和查询的结果中,可以看出系统的负载是否均衡,通信或同步开销是否过大,等等。从图5(b)中可以看出,在这段时间内,各计算节点资源的负载并不平衡,有的节点几乎是完全空闲,而有的节点则几乎是满负荷工作。



(a) (b)

图5 性能数据的统计、查询分析结果举例

总结 "网格性能工具"目前还只是一个探讨中的概念。本文提出它不仅需要具备采集和分析性能数据的功能,还应该能提供性能数据的跨平台的表示方法。本文对网格性能工具做了研究,并实现了一个面向网格的性能工具原型:THGPT。它可以获取应用程序的运行时性能数据,并以XML语言对性能数据进行跨平台、统一格式的描述,最后实现了一个可以在浏览器上运行的可视化性能数据分析工具。

参考文献

1 http://www-unix.mcs.anl.gov/mpi/mpich/papers/mpicharticle/node31.html

2 Jiang S Y,Rhykerd C L,Gubbins K E. Layering, Freezing Transitions, Capillary Condensation and Diffusion of Methane in Slit Carbon Pores. Mol. Phys.,1993,79:373

3 Aydt R,et al. Simple Case Study of a Grid Performance System. http://www-didc.lbl.gov/GGF-PERF/GMA-WG/papers/GWD-GP-9-2.pdf

4 Smith W,Gunter D,Quesnel D. Simple XML Producer/Consumer Protocol; http://www-didc.lbl.gov/GGF-PERF/GMA-WG/papers/GWD-GP-8-2.pdf

5 http://nws.npaci.edu/NWS/

6 http://www-unix.mcs.anl.gov/perfvis/software/viewers/

7 Pancake C M,Simmons M L,Yan J C. Performance evaluation tools for parallel and distributed systems. IEEE Parallel and Distributed Technology 3 4 Winter 1995 IEEE,1995.14～20,1063～6552

8 Shi Pei-zhi, Li San-li, Meng Jie. THPTii: A Visualization Tool for Load Information Collecting and Performance Analyse in High Performance Computing System. Mini-Micro Systems, Aug. 2002,23:902～906

附录一 文档格式定义

资源预定义格式文档(ResourceRef.dtd):

```
<!ELEMENT Resources (ResourceDescription) * >
<! ELEMENT ResourceDescription (Location, Provider, Description)>
<!ATTLIST ResourceDescription ResourceID ID #REQUIRED>
<!ELEMENT Location (#PCDATA)>
<!ELEMENT Provider (#PCDATA)>
<!ELEMENT Description (#PCDATA)>
```

性能预定义格式文档(PerformanceTypeID.dtd):

```
<!ELEMENT PerformanceTypes (PerfType) * >
<!ELEMENT PerfType (TypeDescription)>
```

```
<!ATTLIST PerfType TypeID ID #REQUIRED>
<!ELEMENT TypeDescription (#PCDATA)>
```

操作预定义格式文档(Operation.dtd):

```
<!ELEMENT OperationTypes (OpType)*>
<!ELEMENT OpType (OpTypeDescription)>
<!ATTLIST OpType TypeID ID #REQUIRED>
<!ELEMENT OpTypeDescription (#PCDATA)>
```

性能数据格式定义文档(Perfdata.dtd):

```
<!ENTITY % ResourceRef-DTD SYSTEM "ResourceRef.dtd">
%ResourceRef-DTD;
<!ENTITY % PerformanceTypeID-DTD SYSTEM "PerformanceTypeID.dtd">
%PerformanceTypeID-DTD;
<!ENTITY % Operation-DTD SYSTEM "Operation.dtd">
%Operation-DTD;
<!ENTITY PerformanceTypeIDs SYSTEM "PerformanceTypeID.xml">
<!ENTITY OperationTypeIDs SYSTEM "Operation.xml">
<!ENTITY ResourceRef SYSTEM "ResourceRef.xml">
<!ELEMENT PerformanDoc (Resources, PerformanceTypes, OperationTypes, (ResourceReference, PerfDataList+, OperationList*)+)>
<!ELEMENT ResourceReference EMPTY>
<!ATTLIST ResourceReference ResourceID IDREF #REQUIRED>
<!ELEMENT PerfDataList (PerformanceType, FullPerf, Measurement, PerfData+)>
<!ELEMENT PerformanceType EMPTY>
<!ATTLIST PerformanceType TypeID IDREF #REQUIRED>
<!ELEMENT FullPerf (#PCDATA)>
<!ELEMENT Measurement (#PCDATA)>
<!ELEMENT PerfData (StartTime, EndTime, AveragePerfRatio, Max PerfRatio?, MinPerfRatio?)>
<!ELEMENT StartTime (#PCDATA)>
<!ELEMENT EndTime (#PCDATA)>
<!ELEMENT AveragePerfRatio (#PCDATA)>
<!ELEMENT MinPerfRatio (#PCDATA)>
<!ELEMENT MaxPerfRatio (#PCDATA)>
<!ELEMENT OperationList (OperationType, OperationDetail*)+>
<!ELEMENT OperationType EMPTY>
<!ATTLIST OperationType TypeID IDREF #REQUIRED>
<!ELEMENT OperationDetail (OpStartTime, OpEndTime, AdditionalInfo)>
<!ELEMENT OpStartTime (#PCDATA)>
<!ELEMENT OpEndTime (#PCDATA)>
<!ELEMENT AdditionalInfo (#PCDATA)>
```

附录二 范例文档(下列文档均为节选)

资源预定义范例文档(ResourceRef.xml):

```
<?xml version="1.0" encoding="utf-8"?>
<Resources>
    <ResourceDescription ResourceID="tp1">
        <Location>computer14.cluster2.hpclab.cs.tsinghua.edu.cn</Location>
        <Provider>hpclab, dept of computer science, Tsinghua univ, China</Provider>
        <Description>This is a traditional computing resource, computer inside a cluster.</Description>
    </ResourceDescription>
</Resources>
```

性能预定义范例文档(PerformanceTypeID.xml):

```
<?xml version="1.0" encoding="utf-8"?>
<PerformanceTypes>
    <PerfType TypeID="Check_sys_CPU_info">
    <TypeDescription>Available CPU power for computation.</TypeDescription>
    </PerfType>
    <PerfType TypeID="Check_sys_MEM_info">
        <TypeDescription>Available Memory capability in a computer for computation.</TypeDescription>
    </PerfType>
    <PerfType TypeID="ReadRate_Performance">
        <TypeDescription>Data transfer rate while reading from a storage device.</TypeDescription>
    </PerfType>
    <PerfType TypeID="P2P_Bandwidth_Performance">
        <TypeDescription>Bandwidth between two peers.</TypeDescription>
    </PerfType>
</PerformanceTypes>
```

操作预定义范例文档(Operation.xml):

```
<?xml version="1.0" encoding="utf-8" ?>
<OperationTypes>
    <OpType TypeID="Send">
        <OpTypeDescription>
            Send between two processors. Additional information includes source, dest, length, etc.
        </OpTypeDescription>
    </OpType>
    <OpType TypeID="Recv">
        <OpTypeDescription>
            Recv between two processors. Additional information includes source, dest, length, etc.
        </OpTypeDescription>
    </OpType>
    <OpType TypeID="Barrier">
        <OpTypeDescription>
            Barrier, synchronization overhead.
        </OpTypeDescription>
    </OpType>
    <OpType TypeID="Computing">
        <OpTypeDescription>Just computing.</OpTypeDescription>
    </OpType>
</OperationTypes>
```

性能数据范例文档(Perfdata.xml):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE PerformanDoc SYSTEM "Perfdata.dtd">
<PerformanDoc>
    &ResourceRef;
    &PerformanceTypeIDs;
    &OperationTypeIDs;
    <ResourceReference ResourceID="tp1"/>
    <PerfDataList>
        <PerformanceType TypeID="Check_proc_CPU_info"/>
        <FullPerf>500</FullPerf>
        <Measurement>Mhz</Measurement>
        <PerfData>
            <StartTime>241389161</StartTime>
            <EndTime>241399675</EndTime>
            <AveragePerfRatio>99.13</AveragePerfRatio>
        </PerfData>
        <PerfData>
            <StartTime>243409263</StartTime>
            <EndTime>243419234</EndTime>
            <AveragePerfRatio>99.91</AveragePerfRatio>
        </PerfData>
    </PerfDataList>
    <PerfDataList>
        <PerformanceType TypeID="Check_proc_MEM_info"/>
        <FullPerf>256</FullPerf>
        <Measurement>MBytes</Measurement>
        <PerfData>
            <StartTime>240379161</StartTime>
            <EndTime>240389675</EndTime>
            <AveragePerfRatio>1.05</AveragePerfRatio>
        </PerfData>
        <PerfData>
            <StartTime>241389162</StartTime>
            <EndTime>241399676</EndTime>
            <AveragePerfRatio>1.05</AveragePerfRatio>
        </PerfData>
    </PerfDataList>
    <OperationList>
    <OperationType TypeID="Sendrecv"/>
        <OperationDetail>
            <OpStartTime>174960281</OpStartTime>
            <OpEndTime>174970097</OpEndTime>
            <AdditionalInfo>1 0 6 2 1 0 2 6 Sendrecv, datatype, length, sender, receiver
            </AdditionalInfo>
        </OperationDetail>
    </OperationList>
</PerformanDoc>
```