第 1 章　监控系统简介

本章阐述了监控系统的发展历程、监控系统的原理，以及监控系统的实现过程，对现有的开源监控解决方案进行了综合分析，目的是让读者全面了解监控系统，让读者更加深入地学习到监控系统的原理，为后面章节的学习做好准备。

1.1　为何需要监控系统

在一个 IT 环境中会存在各种各样的设备，例如，硬件设备、软件设备，其系统的构成也是非常复杂的，通常由如图 1-1 所示的模型构成。

图 1-1

多种应用构成复杂的 IT 业务系统，保证这些资源的正常运转，是一个公司 IT 部门的职责。而要让这些应用能够稳定地运行，则需要专业 IT 人员进行设计、架构、维护和调优。在这个过程中，为了及时掌控基础环境和业务应用系统的可用性，需要获取各个组件的运行状态，如 CPU 的利用率、系统的负载、服务的运行、端口的连通、带宽流量、网站访问状态码等信息。而这一切都离不开监控系统。

1.2　监控系统的实现

一个监控系统的组成大体可以分为两部分：**数据采集部分（客户端）和数据存储分析告警展示部分（服务器端）**，如图 1-2 所示。这两部分构成了监控系统的基本模型。

数据采集的工作模式可以分为**被动模式**（服务器端到客户端采集数据）和**主动**

图 1-2

模式（客户端主动上报数据到服务器端）。通常，大多数监控系统应该能同时支持这两种模式。被动模式对服务器的开销较大，适合小规模的监控环境；主动模式对服务器的开销较小，适合大规模的监控环境。

采集数据的协议方式可以分为两种：**专用客户端采集**和**公用协议采集**（SNMP、SSH、Telnet 等），如图 1-3 所示。

对于采集到的监控数据，可以将其存储到数据库或者文本或者其他方式，具体采用哪一种，应根据实际需求来决定。

怎么规划监控系统的架构设计呢？下面将详细分析。

对于一般的监控环境，被监控的节点不多，产生的数据较少，采用 C/S（Client/Server，客户端/服务器端）架构就足够了，如图 1-4 所示，这种架构适合于规模较小、处于同一地域的环境。

图 1-3

图 1-4

对于大规模的监控环境，被监控的节点多，且监控类型多，监控产生的数据和网络连接开销会非常巨大，而且由于跨地域等多种因素，需要分布式的解决方案，常见的方式为 C/P/S（Client/Proxy/Server，客户端/代理端/服务器端）架构（如图 1-5 所示），采用中间代理将大大提高监控服务器端的处理速度，从而能支撑构建大型分布式监控的环境。

图 1-5

监控系统更重要的功能是告警和故障处理，这对及时解决问题和故障自愈非常重要。告警的时候，需要考虑到故障的有效汇报和集中汇报，防止出现“告警洪水”，即同一类告警信息重复大量地发送。

1.3 监控系统的开源软件现状

前面简单介绍了监控的原理，下面看看已有的解决方案。

在监控软件中，开源的解决方案有流量监控（MRTG、Cacti、SmokePing、Graphite 等）和性能告警（Nagios、Zabbix、Zenoss Core、Ganglia、OpenTSDB 等）可供选择，并且每种软件都有自己的特点和功能，各自的侧重点和目标不完全相同，在设计理念和实现方法上也大同小异，但都具有共同特征，例如，采集数据、分析展示、告警以及简单的故障自动处理。最终都能达到对 IT 系统服务可用性的一个完全展示。下面将详细介绍各自的特点。

1.3.1 MRTG

MRTG（Multi Router Traffic Grapher）是一套可用来绘制网络流量图的软件，由瑞士奥尔滕的 Tobias Oetiker 与 Dave Rand 所开发，以 GPL 授权。

MRTG 最早的版本是在 1995 年春推出的，用 Perl 语言写成，可跨平台使用，数据采集用 SNMP 协议，MRTG 将收集到的数据通过 Web 页面以 GIF 或 PNG 格式绘制出图像，并以日、周、月为单位分别绘出，可以查询最大值和最小值，如图 1-6 所示。

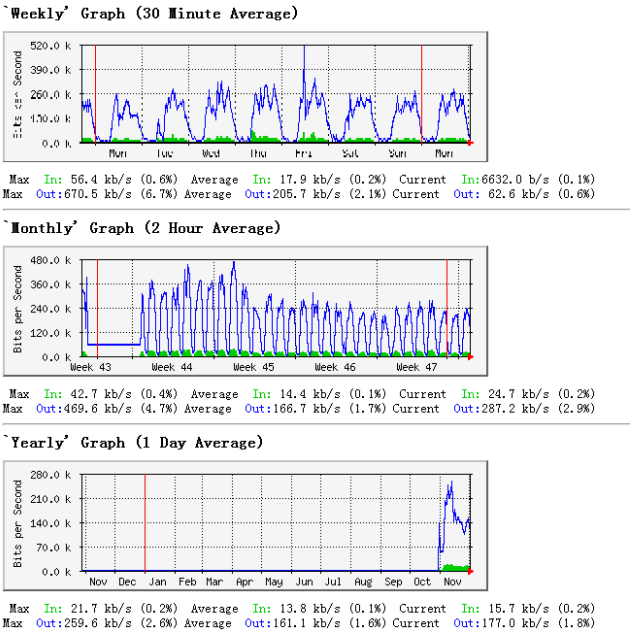

图 1-6

MRTG 原本只能绘出网络设备的流量图，后来发展出了各种插件。因此，网

络以外的设备也可由 MRTG 监控，例如，服务器的硬盘使用量、CPU 的负载等。

1.3.2　Cacti

Cacti（英文含义为仙人掌）是一套基于 PHP、MySQL、SNMP 和 RRDtool 开发的网络流量监测图形分析工具，如图 1-7 所示。它通过 snmpget 来获取数据，使用 RRDtool 绘图，但使用者无须了解 RRDtool 复杂的参数。它提供了非常强大的数据和用户管理功能，可以指定每一个用户能查看树状结构、主机设备以及任何一张图，还可以与 LDAP 结合进行用户认证，同时也能自定义模板，在历史数据的展示监控方面，其功能相当不错。

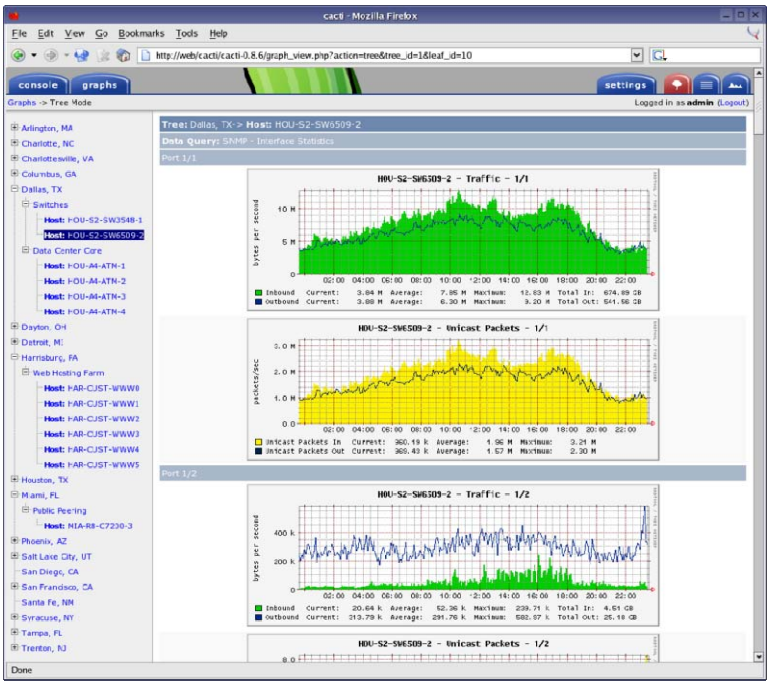

图 1-7

Cacti 通过添加模板，使不同设备的监控添加具有可复用性，并且具备可自定义绘图的功能，具有强大的运算能力（数据的叠加功能）。

1.3.3　SmokePing

SmokePing 主要用于监视网络性能，包括常规的 ping、WWW 服务器性能、DNS 查询性能、SSH 性能等。底层也是用 RRDtool 做支持，特点是绘制的图非常漂亮，网络丢包和延迟用颜色和阴影来表示，支持将多张图叠放在一起，如图 1-8

所示。其作者（Tobi Oetiker）还开发了 MRTG 和 RRDtool 等工具。

SmokePing 的站点为：http://tobi.oetiker.ch/hp/。

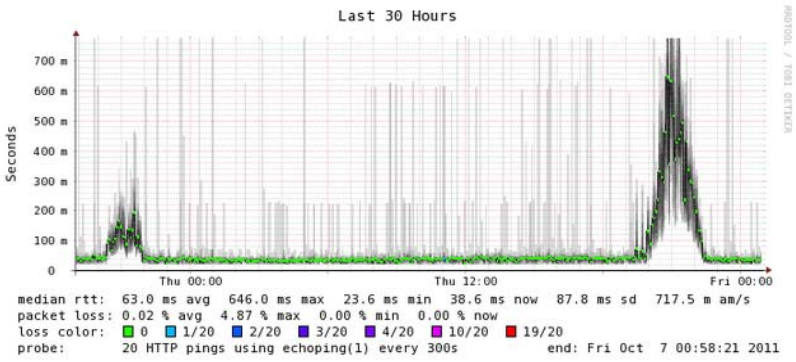

图 1-8

1.3.4 Graphite

Graphite 是一个用于采集网站实时信息并进行统计的开源项目。Graphite 服务支持平均每分钟 4800 次更新操作，采用简单文本协议，具有绘图功能，其即插即用的功能可方便地用于任何需要监控的系统上。Graphite 的界面如图 1-9 所示。

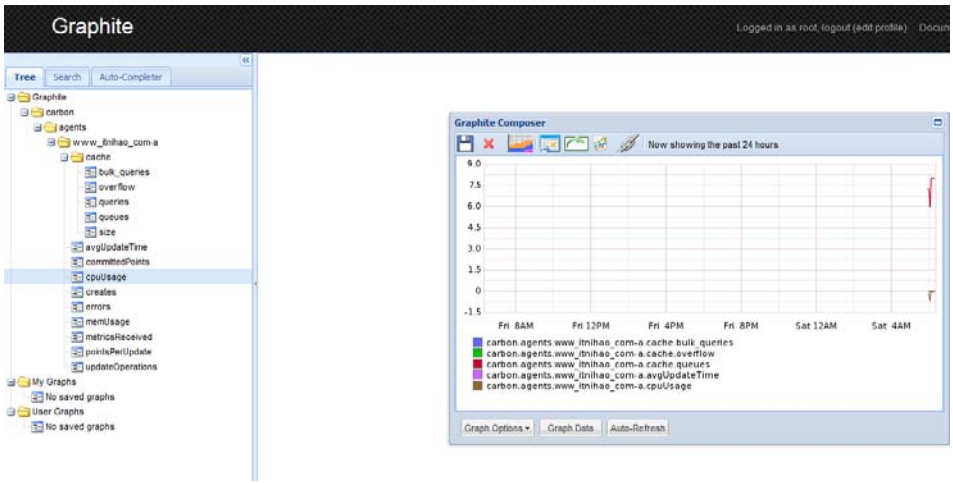

图 1-9

和其他监控工具不同的是，Graphite 本身并不收集具体的数据，这些数据收集的工作通常由第三方工具或插件完成（如 Ganglia、Nagios、collectd、statsd、Collectl 等）。因此，可以说，Graphite 是一个绘图工具。

值得一提的是，Graphite 用 Python 语言编写，采用 Django 框架，对于熟悉

Python 的用户（通常是运维人员）来说，将是一个不错的绘图工具选择。

简单地说，Graphite 做两件事：存储数据和按需绘图。

Graphite 的官方站点为：http://graphite.wikidot.com/。

1.3.5　Nagios

Nagios 是一个企业级的监控系统，可监控服务的运行状态和网络信息等，并能监视所指定的本地或远程主机参数以及服务，同时提供异常告警通知功能等。

Nagios 可运行在 Linux 和 UNIX 平台上，同时提供一个可选的基于浏览器的 Web 界面，以方便系统管理人员查看网络状态、各种系统问题，以及日志等，如图 1-10 所示。

Nagios 的功能侧重于监控服务的可用性，能及时根据触发条件告警。

目前，Nagios 也占领了一定的市场份额，不过从笔者的观察来看，Nagios 并没有与时俱进，已经不能满足于多变的监控需求，架构的扩展性和使用的便捷性有待增强，其高级功能集成在商业版 Nagios XI 中。

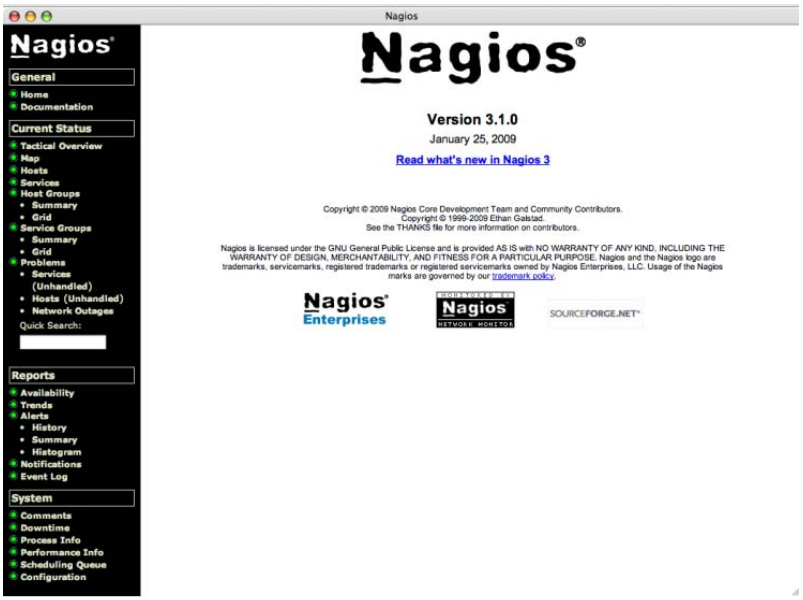

图 1-10

1.3.6　Zenoss Core

Zenoss Core（简称 Zenoss）是开源企业级 IT 管理软件，它允许 IT 管理员依靠单一的 Web 控制台来监控网络架构的状态和健康度。

Zenoss Core 的强大功能来自深入的列表与配置管理数据库，用于发现和管理公司 IT 环境的各类资产（包括服务器、网络和其他结构设备）。Zenoss 可以创建关键资产清单和对应的组件级别（接口、服务、进程、已安装的软件等）。建立好模型后，Zenoss 就可以监控和报告 IT 架构中各种资源的状态和性能状况了。同时还提供与 CMDB 关联的事件和错误管理系统，以协助提高各类事件和提醒的管理效率，以此提高 IT 管理人员的效率。

Zenoss Core 采用 SNMP 来采集数据，其界面如图 1-11 所示。

图 1-11

1.3.7 Ganglia

Ganglia 是一个跨平台的、可扩展的、高性能的分布式监控系统，如集群和网格。它基于分层设计，使用广泛的技术，用 RRDtool 存储数据，具有可视化界面，适合于对集群系统的自动化监控。其精心设计的数据结构和算法使得监控端到被监控端的连接开销非常低。目前已经有成千上万的集群正在使用这个监控系统，可以轻松地处理 2000 个节点的集群环境。该软件的部分截图如图 1-12 所示。

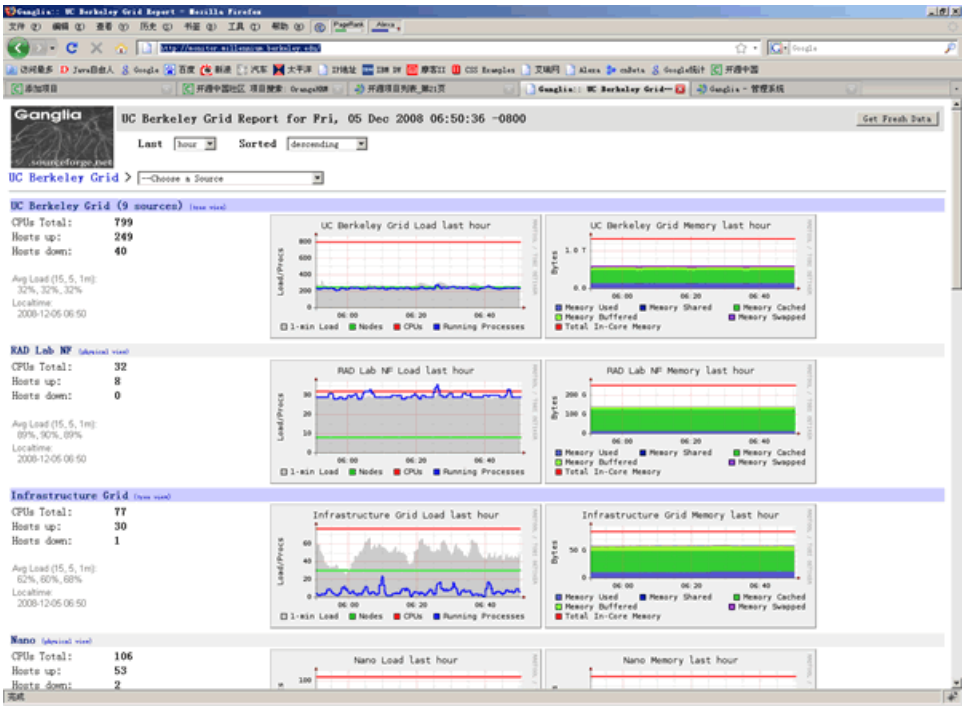

图 1-12

1.3.8　OpenTSDB

开源监控系统 OpenTSDB 用 HBase 存储所有时序（无须采样）的数据，来构建一个分布式、可伸缩的时间序列数据库。它支持秒级数据采集，支持永久存储，可以做容量规划，并很容易地接入到现有的告警系统里（如图 1-13 所示）。OpenTSDB 可以从大规模的集群（包括集群中的网络设备、操作系统、应用程序）中获取相应的采集指标，并进行存储、索引和服务，从而使这些数据更容易让人理解，如 Web 化、图形化等。

在对实时性要求比较高的场合，OpenTSDB 是一个很好的选择。它支持秒级别的数据采集，这在其他监控系统中是无法想象的。因得益于其存储系统的选择，所以它支持大数据分析。因此，这个开源软件在未来的环境中会有更多的用户，也会获得更广泛的支持。

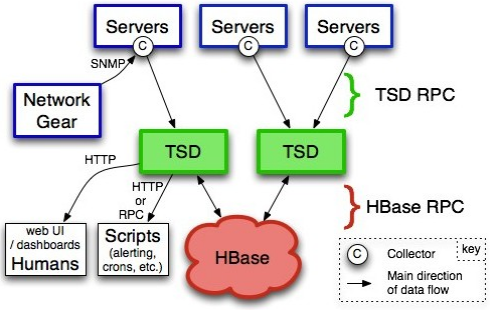

图 1-13

1.3.9 Zabbix

Zabbix 是一个分布式监控系统，支持多种采集方式和采集客户端，有专用的 Agent（代理），也可以支持 SNMP、IPMI、JMX、Telnet、SSH 等多种协议，它将采集到的数据存放到数据库，然后对其进行分析整理，达到条件触发告警。其灵活的扩展性和丰富的功能是其他监控系统所不能比的。相对来说，它的总体功能做得非常优秀，其界面如图 1-14 和图 1-15 所示。

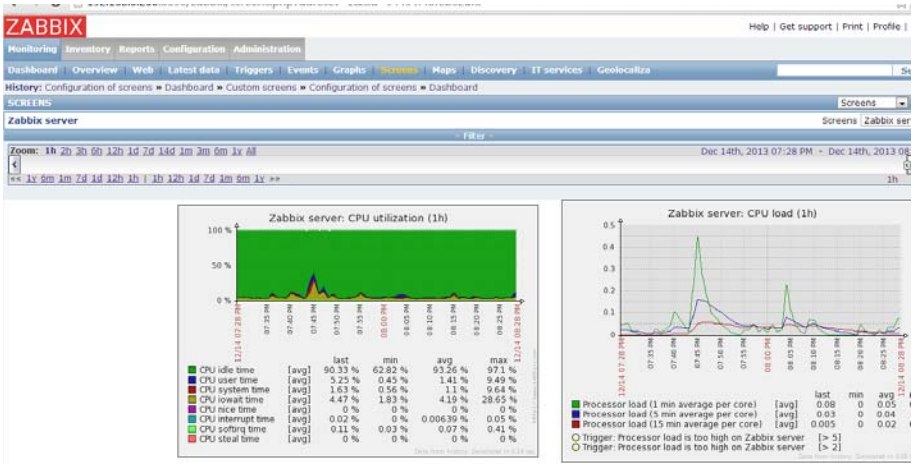

图 1-14

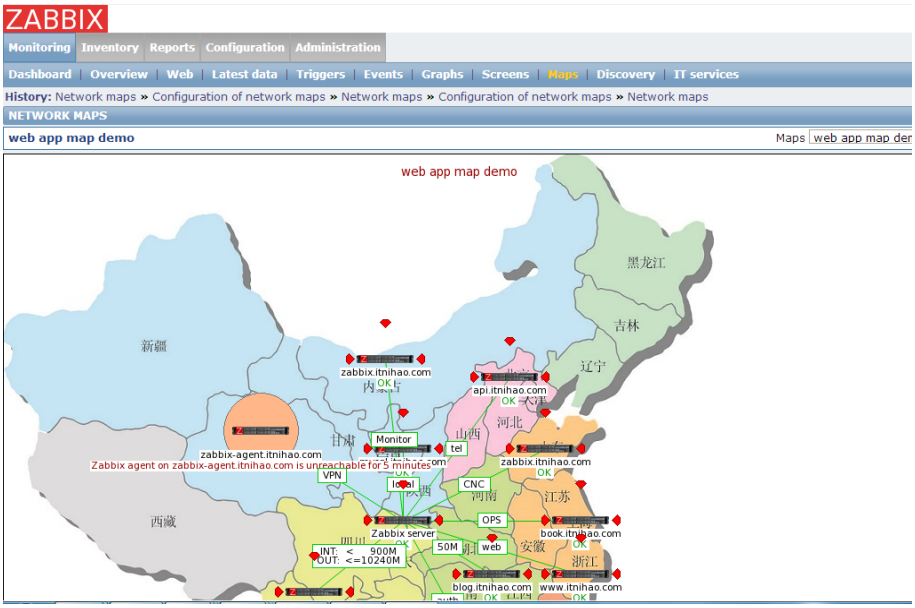

图 1-15

该监控工具也是本书的重点，具体功能将在后面详细介绍。

从以上各种监控系统的对比来看，Zabbix 都是具有优势的，其丰富的功能、可扩展的能力、二次开发的能力和简单易用的特点，读者只要稍加学习，即可构建起自己的监控系统。

1.4　监控系统的原理探究

前面已经介绍了为什么需要监控系统，下面将深入介绍监控系统的原理。

1．监控系统的诞生

首先来看一下监控系统的使用者都有哪些？

试想一个很小的网站，刚上线也许就只有一个 VPS 或者一台服务器。随着业务的发展，它逐渐成为一个大型的网站，服务器从一台发展到多台，运维人员从一个发展到多个，业务战线也从一条发展到多条，这时候出现故障的概率就会大大增加。当有一天，你作为该网站的运维人员，你的老板问你为何某个服务不可使用，为何出现故障时，而身为运维人员的你却不知道故障出现在哪里，此时你是不称职的。如果换一个角度，在未发生故障的时候就已经把故障解决了，那你就是幕后英雄。

有一个故事是讲神医扁鹊的。有人对扁鹊说：“你是天下的神医，能治天下各种疑难杂症，很了不起。”而扁鹊却说：“你知道吗？我和我长兄比起来，可差远啦！我长兄给人治病，是治人于病未发之前，在稍见端倪的时候就能防患于未然，而我只能治人病于膏肓之中，往往是病人受尽折磨才来找我。”（引自魏文王问扁鹊——《鹖冠子》世贤第十六）

监控系统就是这么一个“神医”，救人于膏肓之中（发送故障告警，或者自动修复故障），在发生故障之前，监控系统中会隐约显现故障的前期迹象，这也适用于其他事物，即任何事物的发生必有其原因和条件。有经验的人会从这小小的迹象中发现更大的问题，例如，突发的流量增长，突发的访问量增大，某台服务器的瞬间负载变高，都表明了即将出现异常情况。对出现的故障，能及时通知告警和故障的自动修复，对运维人员响应处理故障的速度会大大加快，甚至在异常严重的故障情况下，对及时采用应急预案，有不可或缺的作用。如果没有监控，故障的反馈往往来自用户的报告，然后才由运维人员处理。

2．监控系统的实现

监控系统往往需要对物理硬件和应用软件的性能和参数进行数据汇集，实现集中管理和统一分析。

在一个监控系统中，构成要素为监控服务器端程序、数据存储、被采集节点

等相关模块，其告警分析和自动故障处理功能由服务器端执行。在数据采集完成之后，需要对采集到的数据进行分析和处理，判断是否有异常，是否属于告警条件。告警条件如何设置呢？通常是根据实际的经验值、业务需求来设置告警阈值。达到告警条件时，则发送告警信息给管理人员，然而，对于有些故障，我们希望程序能自动处理，减少人工干预，让程序自动修复，只在出现严重故障、程序无法判断的时候，才告警通知管理人员处理。

一个监控系统往往需要集成资产管理，可以从逻辑上展示业务和功能的信息，通过对其进行数据分析，做到对投资与回报的一个反馈展示，为资产的合理规划与使用提供了依据。

从其工作模式来看，监控系统的数据采集可以分为两种：主动监控和被动监控，在 1.2 节已经详述，一个理想的监控系统，其采集端支持的采集方式越多，其扩张能力越强大，适用的环境场合越多。

监控系统需要提供一个 API，方便其他功能系统对监控数据进行操作管理，这在业务系统精密的情况下显得特别重要，通常能对外提供 API 功能的软件，其用户群会更广，产品会越做越好。API 一般可以分为 RESTful、SOAP 等形式，数据类型有 JSON、XML 等多种。从目前的趋势来看，RESTful 已经成为绝大多数 API 首选的方式。

监控系统需要对故障数据进行分析汇总，从故障中分析出现的概率，从而可以积累经验，避免以后出现类似的问题。例如，由于机器硬件导致的故障，其概率有多大，哪些部件最容易出问题，出问题的影响概率多大，问题解决的概率有多大。从监控的数据中就可以分析并发现相关数据，在此基础上进行分析汇总，可以整理出相应的对策和相应的技术应急方案。

常见的监控系统性能采集指标如表 1-1 所示。

表 1-1

监 控 项 目	详 细 内 容
主机监控	CPU、内存、磁盘的剩余空间/利用率和 I/O、SWAP 使用率、系统 UP 时间、进程数、负载
网卡监控	Ping 的往返时间及包成功率、网卡流量，包括流入/流出量和错误的数据包数
文件监控	监控文件大小、Hash 值，匹配查询、字符串存在与否
URL 监控	监测指定 URL 访问过程中的返回码、下载时间及文件大小，支持内容匹配
应用程序	端口和内存使用率、CPU 使用率、服务状态、请求数、并发连接数、消息队列的字节数、Client 事务处理数、Service 状态等
数据库	监测数据库中指定的表空间、数据库的游标数、Session 数、事务数、死锁数、缓冲池命中率、库 Cache 命中率、当前连接数、进程的内存利用率等性能参数
日志	错误日志匹配、特定字符串匹配
硬件	温度、风扇转速、电压等

3．监控系统对时间的要求

监控系统需要根据实际应用的需求，实时/非实时地采集和展示数据。另外，包括历史趋势数据的展示和分析，以及容量报表、可用性报告的生成。

4．监控系统的告警需求

支持**多种方式，如短信、邮件、IM 和其他接口。具备可定制化功能**，对第三方告警介质提供可编程接口。这一点在很多场合非常重要，例如，将告警结果发送到专用的告警分析系统。

支持**对告警内容的分析自动处理，防止误报、漏报，以及防止抖动**。这一点对大多数监控系统都是一个值得挑战和研究的课题。例如，一个机房网络发生故障，按照常规告警内容，会收到无数条告警信息，内容是每个设备的故障，而对于更高级的告警信息，我们希望收到的是“某机房存在网络故障，受影响的设备的 IP 是 X.X.X.X～X.X.X.X，受影响的业务是 XXX.”，这样做的目的是让告警信息更智能、更有效，防止“告警炸弹”的产生。

简而言之，监控数据的采集、存储、分析和故障报告是每个监控系统的基本功能，其他复杂的附加功能则是监控系统的增值业务。

在本书中，你将会看到以上功能的具体体现，无论你是使用 Zabbix，还是自行开发监控系统，本书都具有良好的参考价值。