计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2017,26(10):53−60 [doi: 10.15888/j.cnki.csa.005978]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于爬虫的数据监控系统①

董　博, 李　翀, 刘学敏, 董科军

(中国科学院 计算机网络信息中心, 北京 100190)

摘　要: 近年来, 随着互联网技术的快速发展, 云计算服务开始全面普及, 大型互联网公司以及中小型企业已经开始提供自己的云计算服务, 中国科学院也提供了云计算服务平台——中科院“科技云”. 本文针对中国科学院 “科技云”项目的实际需求, 参考已有商业云监控系统的功能和运行模式, 设计并实现了一种基于爬虫的数据监控系统. 该系统相比商业云监控系统, 支持 URL(布尔) 类型数据监测的同时, 增加了对数值类型、文本类型的监测, 更好的支持第三方服务的监测, 并实现了服务故障警报, 监测数据可视化以及监测数据分布式存储.

关键词: 数据监控; 网络爬虫; 云计算; 分布式; 可视化

引用格式: 董博,李翀,刘学敏,董科军.基于爬虫的数据监控系统.计算机系统应用,2017,26(10):53–60. http://www.c-s-a.org.cn/1003-3254/5978.html

Spider-Based Data Monitor System

DONG Bo, LI Chong, LIU Xue-Min, DONG Ke-Jun

(Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: In recent years, with the rapid development of Internet technology, cloud computing services begin to widely spread. Large-scale Internet companies, small and medium enterprises have begun to provide their own cloud computing services. The Chinese Academy of Sciences also provides a cloud computing service platform - CAS “science-cloud”. Concerned with the actual needs of the “Science Cloud” project of the Chinese Academy of Sciences, this paper designs and implements a spider-based data monitoring system referencing the function and operation mode of the existing commercial cloud monitoring systems. Compared with the commercial cloud monitoring system, this system supports the monitoring of URL (Boolean) type data, adds the monitoring of numeric types and text types, supports better monitoring of the third party services, and realizes service failure alarm and monitoring data Visualization and monitoring data distributed storage.

Key words: data monitor; web crawler; cloud computing; distributed; visualization

随着信息技术和互联网产业创新的不断发展, 以云计算为代表的变革性技术正在迅速的普及, 互联网进入了大数据时代, 云计算技术得到了广泛的应用和快速的发展. 特别是近几年来, 随着云基础服务能力的提升, 基于云计算的互联网轻量级小规模服务不断增多, 大量的微小服务急需第三方支持的松耦合数据监控系统, 进行有效的服务监控和提醒[1].

本论文针对中国科学院 “科技云”项目的实际需求, 设计并实现了一种基于爬虫的数据监控系统. 此系统为“科技云”项目所所涵盖的基础设施服务、平台服务、软件即服务等公开发布的数据及服务状态, 提供有效的服务监控和数据汇聚. 以满足服务管理人员对服务进行实时监控、以及用户对服务状况进行了解, 并提供科技云服务门户监控服务.

① 基金项目: 中国科学院十二五信息化专项《科研信息化应用推进工程》(XXH12503)
收稿时间: 2017-01-10; 采用时间: 2017-02-13

1 研究现状和相关工作

云计算, 是一种基于互联网的计算方式, 通过此种方式, 可以按需把共享的软硬件资源和信息提供给计算机和其他设备. 云计算描述了一种基于互联网的新的 IT 服务增加、使用和交付模式, 其通常涉及通过互联网来提供动态易扩展且经常是虚拟化的资源[2].

根据美国国家标准和技术院对云计算服务的定义可明确三种服务模式[3]: Software-as-a-Service(SaaS, 软件即服务)、Platform-as-a-Service(PaaS, 平台即服务) 及 Infrastructure-as-a-Service(IaaS, 基础设施即服务). 如今的云计算服务已经非常普及, 包括大型互联网公司提供的公有云服务、中小型企业提供的私有云服务等. 他们在提供云服务的同时, 还提供负责监控云服务资源的使用情况、健康状况等信息的云服务监测功能, 如大型的云服务提供商 Amazon、阿里巴巴、腾讯等都有自己的云服务监控系统. 下面简单介绍一下以上三个云服务监控系统:

Amazon CloudWatch 是一项针对 AWS (Amazon web services) 云资源和在 AWS 上运行的应用程序进行监控的服务. Amazon CloudWatch 能够监控 Amazon EC2 实例、Amazon DynamoDB 表和 Amazon RDS DB 实例等各种 AWS 资源, 同时也能够监控应用程序和服务生成的定制指标, 以及应用程序生成的任何日志文件[4].

阿里云监控是一项针对阿里云资源和互联网应用进行监控的服务. 云监控服务可用于收集获取阿里云资源的监控指标, 探测互联网服务可用性, 以及针对指标设置警报. 阿里云监控服务能够监控云服务 ECS、云数据库 RDS 和负载均衡等各种云服务资源, 同时也能够通过 HTTP、CMP 等通用网络协议监控互联网应用的可用性[5].

腾讯云监控为腾讯云服务提供全方位监控, 借助于腾讯海量数据处理能力能够对于资源及成本进行智能化的分析, 直观展示各种云服务的资源使用状况、负载状况性能及系统健康状况等[6].

以上公有云服务平台提供的云监控服务, 存在以下两个方面的缺陷:

(1) 以企业自有的公有云服务为出发点. 例如阿里云监控主要针对阿里 ECS 等主机服务提供详细的云监控服务, 依赖于具体主机所提供的内部接口服务. 例如主机安全监控服务、网络流量监控服务等.

(2) 针对第三方的监控服务类型有限. 例如对于外部的第三方服务, 阿里云只能提供 URL 服务正常率访问监测等服务, 利用分布式站点来实现对第三方端口服务的定期探测, 为第三方服务提供服务故障邮件等提醒通知. 但是没有提供文本类型数据和数值类型数据的监测功能.

通过对以上云监控系统的分析可知, 目前还没有一款能够友好全面支持第三方服务的云监控系统. 根据中科院“科技云”的实际需求, 在能够定期监测服务健康率的同时, 还需实现对文本类型和数值类型数据的监测功能, 因此在已有商业云服务监控系统的功能和运行模式, 本课题增加了基于文本类型数据和数值类型数据的监测方法, 并对云服务资源和监控系统之间进行解耦, 使其能够友好地支持第三方服务监控, 同时满足对中科院“科技云”云服务的监控.

本文参考阿里云监控等已有商业云服务监控系统的功能和运行模式, 结合中科院“科技云”的实际需求, 设计并实现了一种基于爬虫的数据监控平台.

2 系统架构设计

根据中国科学院科技云服务监控需求, 科技云服务监控的类型划分为以下三种类型: 文本类型数据监控、数值类型数据监控以及布尔类型数据监控 (URL 类型数据监控). 其中 URL 类型数据指传统的服务可用性监控, 通常每次监控为布尔值数据, 即 0 或 1; 数值类型的数据指具体以浮点类型所监控到的数据, 例如某项服务的数据值等; 文本类型数据为字符串类型, 每种类型的文本类型数据可自定义解析代码, 在后期进行数据解析、处理和可视化.

根据每种数据类型的特点和网络爬虫技术, 分别实现对相应数据的爬取; 并为方便服务管理人员和用户快速聚焦到关注的服务和数据上, 而实现了数据的可视化, 以及发生故障的报警. 系统设计主要分为数据获取层、数据存储层及数据访问层三部分, 整体架构图如图 1 所示.

数据获取层通过爬虫引擎和相应的适配器获取各类资源数据, 然后按照对应的时间单位 (例如: 分钟, 小时, 天等) 拼接成 JSON 字符串, 每次爬取都会生成一个 JSON 字符串. 把 JSON 数据传输到数据存储层, 通过 Cobar 中间件交由分布式 MySQL 数据库进行存储. 当数据访问层访问数据时, 直接从数据存储层拉取对

应的 JSON 数据并解析, 对解析出来的数据进行可视化或者提供给接口访问等. 为了方便对数据的处理, 这三个模块之间统一利用 JSON 数据作为传输手段, 省去了各模块之间传输数据不一致造成的麻烦.

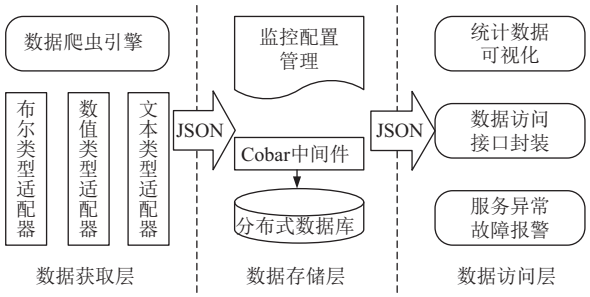

图 1 系统总体框图

2.1 数据获取层

数据获取层主要实现用户定制监测项、服务器定时监测指定监测项并爬取监测项数据. 根据监测的数据类型, 分别利用不同适配器进行监测和爬取数据. 适配器主要分为布尔类型、数值类型、文本类型以及数据库爬取等类型.

布尔类型适配器通过 http 协议访问[7], 定时探测 URL 地址, 分为 GET、POST、HEAD 访问, 根据访问获取的状态码和响应时间判断监控对象是否能够正常访问, 然后将其转换成布尔值并存储. 监测数值类型与文本类型数据的方法相一致, 都是通过 URL 地址去访问被监测对象的接口[8], 两者利用相同的适配器爬取其中的数据, 然后在后台分析数据所属类型, 并进行相应的存储. 数据库爬取适配器即根据用户的数据库配置信息连接远程数据库, 从中爬取数据, 并根据分析数据的所属类型进行存储.

2.2 数据存储层

数据存储层主要实现对监测数据的存储和查询. 由于单节点数据库无法满足对大量监测数据的存取, 且对服务器的性能要求较高, 因此本系统数据存储层采用分布式数据库存储技术[9]. 分布式数据库对网络中各结点计算机配置要求不高, 同时提高了系统的可靠性, 再者其非常便于扩充[10].

本系统的分布式数据库将数据表进行水平拆分, 均衡地分配给各个数据库结点, 然后利用中间件来管理各个数据库结点, 并将其作为应用层访问数据库的统一接口. 如图 2 所示.

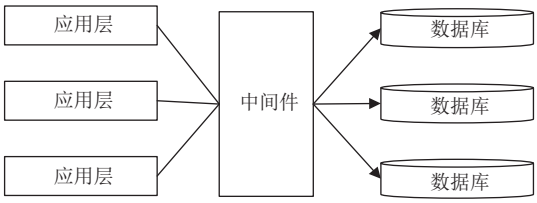

图 2 数据存储层架构

2.3 数据访问层

数据访问层包括接口封装和访问、可视化与展示、故障报警.

接口封装和访问, 是提供给用户的公共 URL 访问接口. 接口模块有两种用途: 第一, 用于供用户访问并获取监测的数据. 用户可以通过 URL 请求, 访问指定监测对象的监测数据, 数据以统一的 JSON 格式展现, 支持三种数据类型的数据. 第二, 用于给可视化模块提供接口访问, 完成数据的可视化, 数据格式的统一, 为数据可视化提供很大的便利; 可视化模块通过访问对应的接口, 利用统一的解析方法对数据进行解析, 最终可视化出相应的图表.

3 数据爬取与统计策略

本监控系统的监控流程为: 用户定制监测项 (包括监测时间粒度、监测地址、数据库配置信息等)、服务器提取监测项配置信息并按照配置进行定时监测、服务器把监测数据持久化、用户通过系统提供的接口查看监测数据以及对监测数据进行可视化.

本系统支持多类型数据的监测, 包括 URL 探测、对象接口以及 DB 的爬取. 监测不同类型的数据需要实现不同的适配器, 且需要在同样的任务调度逻辑中进行, 因此抽取共有的任务调度逻辑声明为接口 job, 其中调度方法为 call(). 不同类型数据的适配器通过实现 job 中的 call() 方法完成不同的监测逻辑, 且共用一致的调度逻辑, 这样任务调度不用关心每种类型监测项的监测逻辑的具体实现, 只需执行接口 job 中的 call() 方法即可. 如图 3 所示.

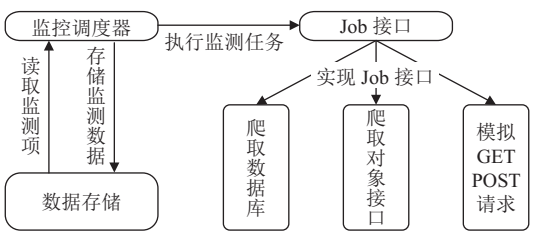

图 3 任务调度架构

3.1 数据爬取策略

本平台通过爬取不同类型的数据实现对各类云服务的监控. 数据类型有文本类型、数值类型、以及布尔类型数据, 爬取不同类型数据需要不同的适配器, 不同适配器的实现方式包括通过 HTTP 协议的 GET、POST、HEAD 请求访问监控对象, 爬取监控对象的接口, 直接爬取监控对象数据库等方式. 如图 4 所示.

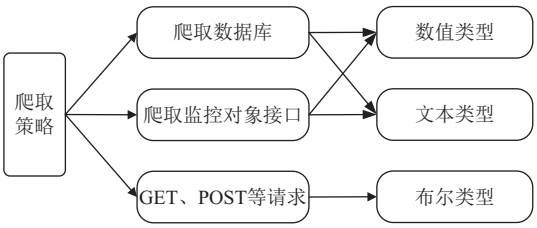

图 4 爬取策略

根据 GET、POST、HEAD 请求访问服务器, 监测服务器响应时间、状态码等数据, 并根据以上数据判断服务是否能够正常访问, 此类数据归为布尔类型数据, 即用 1、0 分别表示服务正常与否, 主要用来监测云服务的健康度, 这也是主流的监控系统所具备的通用功能, 如阿里云监控、腾讯云监控等.

爬取数据库是通过服务器端直接访问监测对象数据库, 并利用用户设置的 SQL 语句直接查询监测对象的数据库, 并把数据存储到监控系统中, 由于数据库中存储着多种类型的数据, 包括文本类型、数值类型等, 不能把爬取的数据统一归为某一数据类型, 必须对爬取的数据进行解析并判断所属类型, 然后存储至相应的数据库或表中.

爬取监控对象接口是通过访问监控对象提供的接口获取数据. 服务提供的数据接口主要以 JSON 数据返回, 但是直接将 JSON 数据存储到监控系统中无法从中挖掘到监控信息, 必须对 JSON 数据进行解析, 从而获取有价值的数据, 并分辨出数据类型, 然后把数据存储至对应的数据库或表中, 供监控系统分析监控对象的状况.

数据的爬取策略主要目的是通过对爬取到的各类型数据进行统一的解析、过滤, 从中挖掘出有价值的数据, 并按照对应的类型存储至相应的数据库或表中. 通过对数据进行统一地处理、存储, 大大降低了监控系统对监测对象数据分析的难度, 监控系统只需从数据库提取数据, 并对数值类型、文本类型、布尔类型这三种数据进行处理分析即可, 而无需关心数据的来源. 此外, 接口访问模块只需提供这三类数据接口, 同时可视化只要针对这三种类型数据进行可视化. 数据的爬取策略从整体上减少了各模块的耦合, 降低了开发成本.

以下是爬取的三类数据示例:

(1) 该类数据中只有单纯的数值, 这类数据归为数值类型数据.

```
{
    count: 460
}
```

(2) 这类数据中除了数值数据外, 还有其它文本数据, 不能对其进行统一的解析, 必须为其实现单独的解析策略, 因此将其归为文本类型数据.

```
{
  jobs: [{
    username: “ucas*”,
    studyarea: “化学”,
    hpcname: “era",
    updatetime: 1462957204,
    corenum: 1000,
    walltime: 9025000
  }],
  status_code: 0,
  status_str: “success”
}
```

(3) 该类数据是通过 http 协议访问 URL 地址获取的数据, 用来监测服务是否正常, 其中 health 表示服务是否正常 (1 表示正常), avgReponseTime 表示响应时间, 该类数据归为 URL 监测数据, 即布尔类型数据.

```
{
  avgReponseTime: 8.74214,
  method: “GET”,
  day: “2017-01-06”,
  health: 1
}
```

3.2 数据统计策略

本系统主要爬取三种类型数据, 因此存储时需对三类数据进行分别存储, 同时为方便对三类监测数据进行访问以及可视化, 需对这些数据分别进行统计, 以此满足系统的需求.

监控系统爬取的数据是以用户或管理人员配置监测项的时间粒度为单位, 以五分钟、十分钟、三十分钟或者一个小时等为时间间隔, 每个监测项的时间粒度不一定一致, 但为方便数据的统计以及可视化, 需要统一这些监测数据的统计单位, 例如, 全部转换为以天、周或者月等为单位的监测数据, 这样将极大地简化后续的数据统计及可视化. 本系统采用的统计单位为天, 即把各个监测数据最终按天进行统计存储.

爬取的数据类型不同, 统一统计单位时所用的策略也就不同, 下面对布尔类型数据和数值类型数据的统计策略进行简单介绍.

(1) 布尔类型数据, 其数据内容主要由 1、0 组成, 其代表所监测的服务健康与否. 监测的时间粒度一般为一小时以内, 但按此时间粒度为单位进行统计, 统计的数据偶然性较大且统计单位过小, 不利于从大范围观察服务一段时间内的整体运行情况.

布尔类型数据按天为单位进行统计时, 云服务管理人员和用户关心的是云服务一天的健康情况, 即一天内服务正常运行所在的比率——健康率, 因此我们计算一天内服务正常的次数与上一天的监测次数的比值, 算出一天的健康率.

(2) 数值类型数据, 其数据内容为整型或浮点型数据, 其通常表示一些服务的注册人数、登录人数、资源使用率等情况.

数值类型数据按天为单位统计时, 服务管理人员或用户关心的是服务在一天内增加的个数、峰值、最小值等信息. 关注的是某个点的数据, 而不是比率数据, 因此统计数值类型数据时, 主要统计一天内最后监测的值、最大值以及最小值等数据.

4 数据访问与可视化

4.1 数据访问

数据访问通过接口访问模块进行, 接口访问模块主要供用户获取监测数据, 以及给可视化提供数据来源. 本平台通过对监测数据进行处理、解析、格式化, 最终转换为统一的 JSON 格式. 系统对每种类型的监测数据进行单独的解析, 抽取出对应的关键数据, 并把这些数据拼接成统一格式的 JSON 数据, 返回给访问者, 这样极大地简化了解析通过接口获取监测数据的过程.

通过上一章可知, 系统会对爬取的数据进行统计. 系统会在每天凌晨一点对前一天的布尔类型、数值类型及文本类型三类数据按不同逻辑进行统计, 并把实时的监测数据和统计的监测数据分开存储至数据库中. 这样当访问统计结果时不用临时计算, 而是直接从数据库中查询即可.

对于布尔类型、数值类型及文本类型这三类数据, 系统提供了基于 JSON 的数据访问方式, 并提供了基于标准数据访问的可视化访问. 用户通过 URL 地址访问系统接口来进行数据访问和可视化, 分别通过以下几种方式进行访问 (productName 为产品名称、collectorName 为监测项名称):

(1) 最新数据访问

访问最新数据时, 系统会从数据库中查询存放实时监测数据的数据表, 查询出距访问时间最近的一次监测数据, 系统将查询结果拼接成 JSON 数据返回给访问者. 因为三种类型的监测数据存储在不同的数据表中, 所以通过发送不同的 URL 请求进行访问.

http://{hostname}/probe/api/data/{productName}/{collectorName}, 访问最新数值和文本类型数据; http://{hostname}/probe/api/url/{productName}/{collectorName}, 访问最新布尔类型数据, 如下所示:

```
[{
  watchTime: “2017-01-11 10:17:54”,
  data: [{
    count: 86703
  }],
  collectorId: 7
}]
```

获取最新的数据类型数据, 其中 watchTime 是这条数据的获取时间, data 字段是监控值.

(2) 统计数据访问

用户可访问最近 n 天、n 月或者 n 年的统计数据. 系统把各类统计数据存放在单独的数据表中, 本系统只存储以天为单位的统计数据. 当用户访问统计数据时, 系统通过解析 URL 地址判断访问的数据类型, 并判断访问统计数据的计量单位 (天、月、年等). 若以天为单位进行访问, 系统直接根据 URL 中指定的天数, 从存放统计数据的表中查询最近的 n 天数据; 若按月或者年为单位进行访问时, 系统依然从存放统计数据的表中查询, 由于表中只存放以天为单位的统计数据, 因此系统需对查询的统计数据进行计算, 并把结果拼

接成 JSON 数据返回给用户.

不同类型的数据, 其计算方式不同. 如上一章所述, 若为布尔类型数据, 对查询的最近 n 月或者 n 年数据按时间段求平均值; 若为数值类型或文本类型, 则查询最近 n 月或者 n 年的数据中每个时间段的最后的监测数据, 或者最大值、最小值等数据.

http://{hostname}/probe/api/stats/{productName}/{collectorName}?[day|month|year]=value, 访问最近一段时间的监测数据, 其中参数“day|month|year”表示“天|月|年”为单位, 例如 day=30 表示之前 30 天的数值统计. 如下所示:

```
[{
   watchTime: “2017-01-11 10:17:54”,
   data: [
   {count:86703, day: “2017-01-06”},
   {count:86710, day: “2017-01-07”},
   {count:86803, day: “2017-01-08”},
   {count:87021, day: “2017-01-09”},
   {count: 87309, day: “2017-01-10”}
   ],
   collectorId: 7
}]
```

获取监测项最近 5 天的统计数据, data 中的五组数据为每天的统计结果.

4.2 可视化

可视化主要为方便用户排除冗余信息并快速聚焦到所关心的问题上. 目前, 本系统实现了对数值类型数据、布尔类型数据的可视化, 这些数据的内容都是由可计算的数据组成, 因此可以进行可视化.

布尔类型数据主要展示一个服务的正常率, 反映一个网站或服务近期的健康状态, 为用户维护网站或服务提供参考. 监控系统对每个监控对象的监测数据 (主要包括状态码, 响应时间等) 进行统计, 计算出监控对象每天、每周、每月、每年等不同时间粒度的服务正常率, 即对监测数据按不同时间粒度求平均值. 该值通过访问系统的统计数据接口获得, 然后结合时间粒度的数据, 组成坐标轴中的 X 轴、Y 轴 (平均值为 Y 轴, 时间粒度为 X 轴), 通过折线图、柱状图等形式可视化, 以此反映最近一段时间内不同时间粒度下的服务正常率.

数值类型数据的可视化主要是展示一些随着时间变化、且对用户有参考价值的数值数据或者展示某一类资源的实时状态. 数值类型数据的可视化由二维数据构成, 其中 X 轴为一组数据, Y 轴为一组数据, 通过折线图或者柱状图展示两者之间的关系. 例如, 注册用户数、网站访问量、服务器资源利用率、超级计算机的作业数量等都是数值类型数据的可视化.

通过该 URL 地址, http://{hostname}/probe/api/chart/{productName}/{collectorName}, 访问监测对象的可视化结果, 图 5 为某布尔类型数据的可视化结果.

5 系统实现与应用

5.1 系统实现

本监控系统基于 Spring MVC 框架, 利用分布式数据库存储服务监测项以及服务监测数据, 采用 Spring 中集成的 Quartz——开源作业调度框架实现服务平台的定时任务功能, 借用 Echarts——纯 Javascript 的图表库, 设计并实现监测数据的可视化.

数据访问层利用 Spring MVC 框架实现. Spring MVC 框架是一个基于驱动的 MVC 框架, 通过实现 Model-View-Controller 模式很好的将数据、业务与展现进行分离. 用户在视图层可以配置监测对象信息, 在模型层进行任务调度、数据爬取以及存储逻辑.

数据爬取通过定时执行多种适配器来爬取监测数据, 本系统利用 Quartz 来实现任务调度和定时任务. Quartz 是 OpenSymphony 开源组织在 Job scheduling 领域的一个开源项目, 其开源作业调度框架完全由 java 编写, 它可以与 J2EE 和 J2SE 应用程序相结合也可以单独使用. Quartz 具有很大的灵活性而又不失简单性, 用户可以通过它来为一个作业创建简单的或复杂的调度[11].

数据存储层利用分布式数据库存储数据. 本系统的分布式数据库由 Cobar 中间件和多个 MySQL 数据库实例组成.

Cobar 是 Alibaba 开源的 MySQL 分布式处理中间件, 是阿里巴巴 B2B 前台应用访问数据库的统一入口[12], 它可以在分布式的环境下像传统数据库一样提供海量数据服务. Cobar 支持将一张表水平拆分成多份并分别放入不同的库, 以此实现表的水平拆分, 也支持将不同的表放入不同的库. 本系统主要使用第一种方式, 将监测数据表进行水平拆分, 均衡地分配到每个数

据库实例中, 每个数据库中保存被分配的数据分片, 以及其他数据分片的备份, 在保证高性能的同时提高稳定性.

应用层通过 Cobar 进行数据访问, Cobar 根据解析收到的 SQL 语句, 判断该语句所涉及的数据分布在哪些分库上, 然后分发到各个分库执行, Cobar 将分库的执行结果进行合并、处理, 最后返回给应用层.

可视化模块通过百度的 Echarts——纯 Javascript 的图表库实现, 此图表库可以流畅的运行在 PC 和移动端, 兼容当前主流浏览器 (IE8/9/10/11, Chrome, Firefox, Safari 等), 底层依赖轻量级的 Canvas 类库 ZRender, 提供直观、生动、可交互、可高度个性化定制的数据可视化图表.

本系统将 Echarts 集成到系统中, 并基于 Echarts 实现更多的定制功能, 让其在同一表中实现对多种时间粒度的可视化, 并可任意切换. 如图 5 所示可同时对以日、周、月、年为时间粒度进行可视化.

5.2 应用

目前, 本监控系统已部署在服务器上并应用于中科院科技云中, 用户可以根据配置信息定时监控科技云的用户数据、服务资源利用率、服务健康率等数据. 服务管理人员和用户可以通过监控系统定制监控对象, 监控时间粒度来监测关注的服务. 该系统除监控科技云服务外, 可对第三方服务进行监控, 用户只需正确配置监控对象信息, 系统便可执行监控, 实现了对布尔类型、数值类型、文本类型的监控. 本监控系统除可用于中科院科技云监控外, 也可作为企业和公司内部私有云的监控系统.

图 6 列出了中科院“科技云”托管的云服务的运行情况. 可视化出了各个服务每天的健康运行情况.

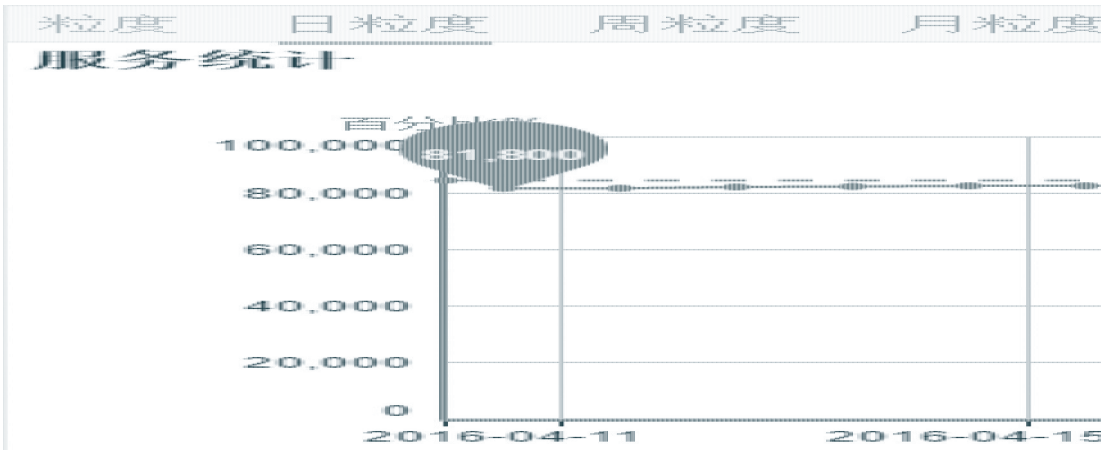

图 5 监测对象可视化结果

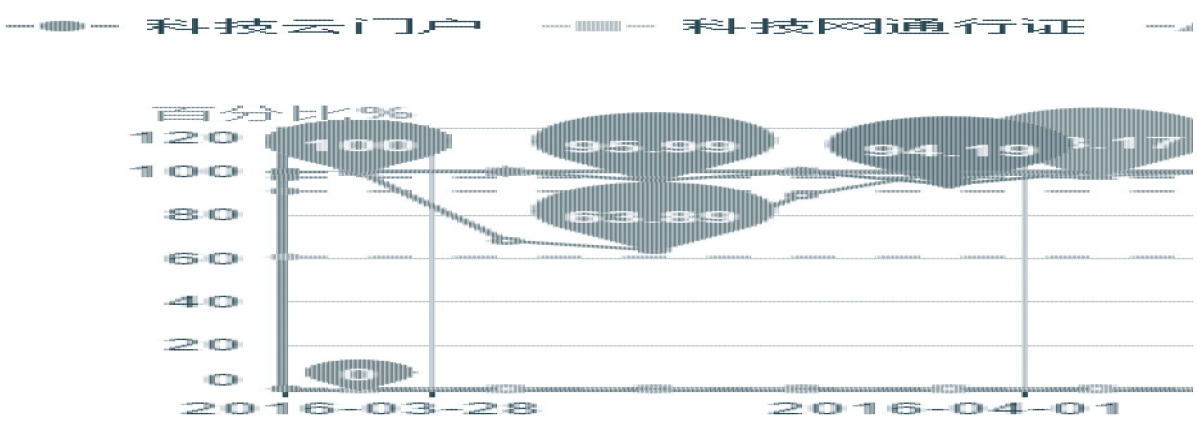

图 6 服务资源使用情况

图 7 和图 8 是中科院计算机网络信息中心下超级计算机中的作业运行情况和资源利用率等信息, 包含了中科院内一些研究所利用超级计算机的情况. 图 9 和图 10 是中科院“科技云”下团队文档库的. 团队文档库是计算机网络信息中心对外提供的团队协作服务, 图中分别列出了团队文档库的月活跃用户数和日访问量.

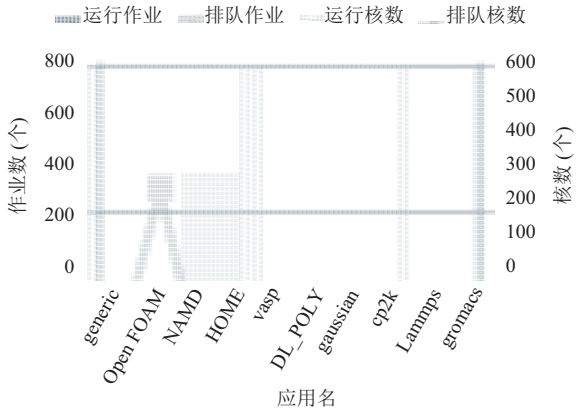

图 7 超级计算机作业运行情况

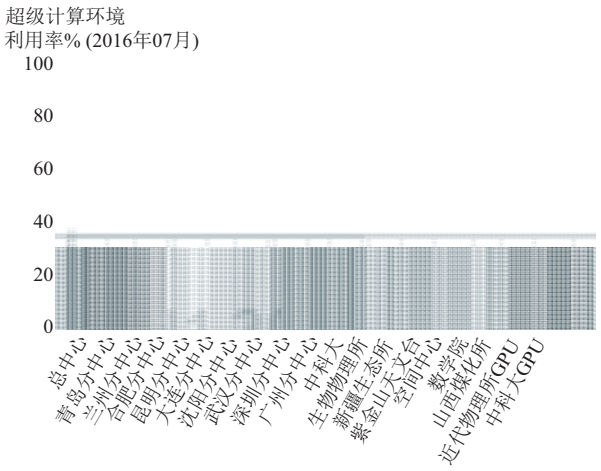

图 8 超级计算机资源使用情况

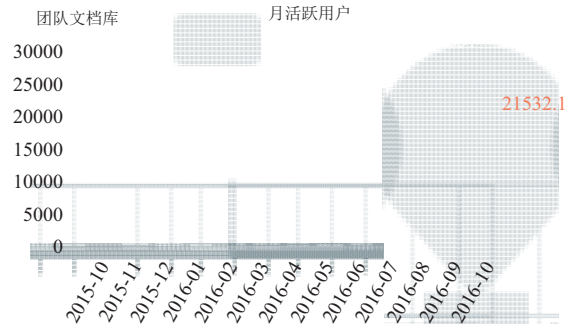

图 9 团队文档库月活跃用户

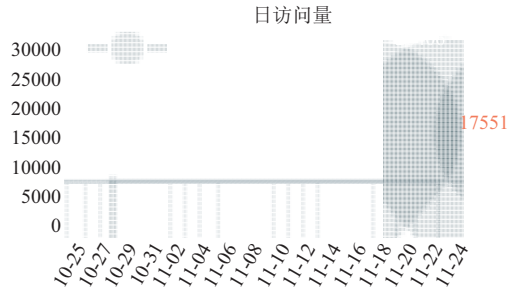

图 10 团队文档库日访问量

6 结语

本文根据中科院科技云的实际情况, 基于网络爬虫技术设计并实现了数据监控系统. 结合科技云的服务和数据的特点, 重点设计了数据爬取和统计策略、监控和存取逻辑, 以适应对多种类型数据的爬取和存储; 同时, 为保证服务正常、高效地运行, 设计和实现了对数据的可视化以及故障报警功能. 该系统兼容多类型数据的爬取, 具备可定制性强、可视化界面简洁重点突出、实时故障报警等特点, 能够满足服务管理人员和用户对科技云服务监控的需求.

参考文献

1 Aceto G, Botta A, DE Donato W, *et al*. Cloud monitoring: A survey. Computer Networks, 2013, 57(9): 2093–2115. [doi: 10.1016/j.comnet.2013.04.001]
2 Armbrust M, Fox A, Griffith R, *et al*. A view of cloud computing. Communications of the ACM, 2010, 53(4): 50–58. [doi: 10.1145/1721654]
3 Mell P, Grance T. The NIST definition of cloud computing. Communications of the ACM, 2010, 53(6): 50–50.
4 Varia J, Mathew S. Overview of amazon web services. Amazon Web Services, 2014.
5 云监控. http://www.aliyun.com/product/jiankong/.
6 基础监控 BCM. https://www.qcloud.com/product/cm.html#wiki.
7 刘继红, 吴军华, 任明鑫. 基于改进的网络蜘蛛算法抽取 Web 站点结构的方法. 江南大学学报 (自然科学版), 2009, 8(5): 555–559.
8 徐远超, 刘江华, 刘丽珍, 等. 基于 Web 的网络爬虫的设计与实现. 微计算机信息, 2007, 23(21): 119–121. [doi: 10.3969/j.issn.1008-0570.2007.21.048]
9 邵佩英. 分布式数据库系统及其应用. 北京: 科学出版社, 2000.
10 庞惠, 翟正利. 论分布式数据库. 电脑知识与技术, 2011, 7(2): 271–273.
11 胡利强, 周冬初, 王伟. Quartz 调度器与 Web 程序整合的研究和应用. 计算机与现代化, 2010, (8): 98–99, 104.