云计算中心运维指标的集中管理①

冯佳丽[1], 刘 顺[2], 彭 璐[1], 张代兰[1]

[1](中国石化石油物探技术研究院 地球物理信息中心, 南京 211103)
[2](中国电子科技集团第二十八研究所, 南京 210007)

摘 要: 提出了云计算中心运维指标的集中管理模式, 利用 Quartz+MVC+Highcharts 技术实现了多个系统中关键运维指标的定时采集, 构建了运维数据的在线管理平台. 结果表明, 该管理模式能挖掘出具有实用性的运维数据, 提升了信息化管理水平.

关键词: 云计算中心; 运维指标; Quartz; 集中管理

Centralized Management for Cloud Computing Center Operation Index

FENG Jia-Li[1], LIU Shun[2], PENG Lu[1], ZHANG Dai-Lan[1]

[1](Center for Geophysical Intelligence Research, Sinopec Geophysical Research Institute, Nanjing 211103, China)
[2](The 28th Research Institute of China Electronics Technology Group Corporation, Nanjing 210007, China)

Abstract: A centralized management model of the cloud computing center operation index is proposed in this paper. The timed data collector and Web management platform are implemented by Quartz+MVC+Highcharts. The results proved that the informative level can be promoted by using this centralized management model.

Key words: cloud computing center; operation index; Quartz; centralized management

近年来, 云计算概念已进入到各个工作领域, 云计算中心也逐步由建设转向实际应用和维护. 大量的 IT 设备(集群、存储、服务器、网络)和辅助设备(空调、UPS、照明)需要在云计算中心机房中集中放置和管理[1]. 计算设备、环境设备、网络设备等都拥有自己单独的一套监控工具, 这些监控工具功能比较单一, 系统比较分散. 且大都只能保存近期数据, 如一周、一个月. 在此之前的数据会定期删除, 难以进行历史数据分析.

对云计算中心的运维管理人员而言, 想了解云计算中心整体情况需要逐一登入高性能设备监控系统[2,3]、网络监控系统、环境监控系统[4]等多个独立的程序, 分别查看不同类型设备的监控情况, 定期制作报表. 且无法获取历史数据或极值数据进行统计和问题分析. 这种分散管理、数据单一的模式阻碍了云计算中心运维效率的提升.

因此, 本文提出建设云计算中心运维指标管理系统, 有效地融合机房设备运维的重要指标, 实现对多种设备的“集中监控、集中运维”.

1 系统设计

云计算中心运维指标管理系统总体设计如图 1. 其中运维指标采集程序实现多数据源实时数据的定时采集和统计数据、历史数据的定时计算; 运维指标数据库实现多种运维指标原始数据和衍生数据的集中存储; 运维指标管理平台则实现跨平台的运维指标的图形化管理界面.

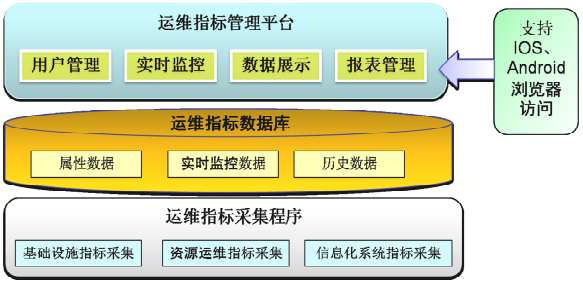

图 1 云计算中心运维指标管理系统架构图

① 收稿时间:2014-12-24:收到修改稿时间:2015-01-26

1.1 运维指标数据库

梳理机房设施、高性能集群系统、集中存储系统、网络系统、信息安全系统中的重要运维指标，建立关键运维指标目录. 建立运维指标数据库，将运维指标转化为结构化数据. 例如，为集群实时监控数据建立表结构，其中包含的数据项有记录 ID、采集时间、集群 ID、CPU 利用率、GPU 利用率、内存利用率、节点利用率、网络读写速率、磁盘读写速率等. 如图 2.

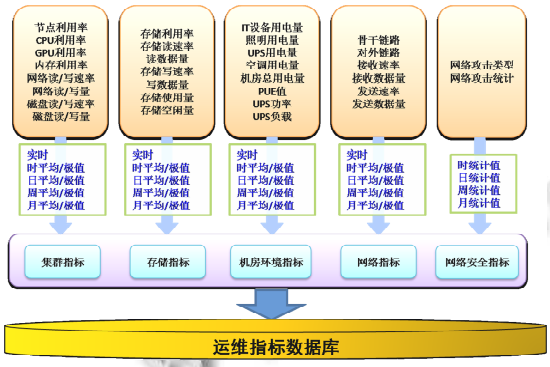

图 2 云计算中心运维指标数据库目录

1.2 数据自动采集

云平台高性能设备监控模块实现了集群、存储的实时监控，采样间隔为 5 分钟；网络管理系统实现了机房内部骨干链路的流量监控. 网络安全管理系统实现了网络攻击的监控，攻击记录存储在该系统数据库中. 机房环境监控系统实现了空调、UPS 等设备的实时监控. 为了实现数据的统一管理及历史数据的回溯，运维指标采集程序需要访问这些数据源，对其中的实时监控数据进行筛选、统计、转存，并通过计算生成运维管理所需的历史平均数据、极值数据. 如图 3，采集程序每 5 分钟定时读取集群实时监控数据并计算所有集群的平均实时数据，每小时、每天、每周、每月定时计算历史数据.

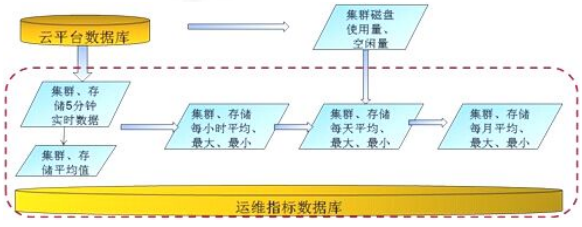

图 3 高性能设备数据采集流程

1.3 运维指标管理平台

运维指标管理平台实现运维指标的规范统一、图形化管理. 实现用户管理、实时数据展示、历史数据检索、报表管理. 使得信息化工作相关人员、决策人员，只需登录该平台，即可快速了解云计算中心所有设备的运行状态和运维指标，进行大量运维指标的在线分析管理，避免多平台操作的复杂性，提高运维管理工作的效率及灵活性. 主要功能架构如图 4.

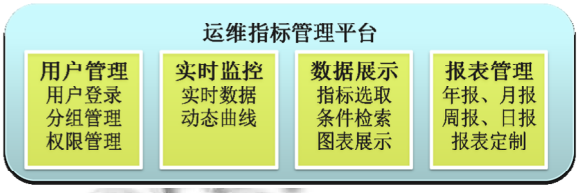

图 4 运维指标管理平台功能架构

(1) 用户管理：该模块实现平台用户的注册、管理，提供用户角色配置、权限配置，使得不同用户能够管理不同运维指标数据，保证数据的安全性.

(2) 实时监控：实现实时数据的动态监控曲线.

(3) 数据展示：实现各种采样间隔数据的展示和大量历史数据的追溯，提供多种检索方式，通过丰富的图表形式直观的展现数据情况.

(4) 报表管理：实现日报、周报、月报、年报的用户定制、分析与导出，部分有权限用户可实现具体数据的导出.

2 技术背景

2.1 MVC 开发模式

MVC 是“Model-View-Controller”(“模型-视图-控制器”)的缩写，这三个层各司其职，互不干涉[5]，有利于开发分工，有利于组件的重用.

模型代表业务流程、业务规则、数据对象模型，是 MVC 模式的核心. 模型对业务流程的处理过程对其它层来说是黑盒操作，它接受视图收集来的数据，并返回最终的处理结果，有利于模型的重构和提高重用性.

视图代表用户交互界面. MVC 模式对于视图的处理仅限于视图上数据的展示和获取，以及收集用户的请求，而不包括业务流程的处理.

控制器将模型与视图匹配在一起，共同完成用户的请求. 控制器不做任何的数据处理，它类似于一个分发器，分派用户的请求并选择恰当的视图用于显示，同时解释用户的输入并将它们映射为模型层可执行的操作.

2.2 定时任务框架 Quartz

Quartz 是一个开源的作业调度框架,它具有很大的可定制性, 在程序中能够用它来实现简单的或复杂的作业调度[6]. Quartz 可以管理几十, 几百, 甚至成千上万的简单或者复杂的作业调度. 这些作业可以是任何标准的 Java 组件或者是 EJB[7]. Quartz 框架的核心是 Quartz Scheduler, 它负责管理 Quartz 的各种组件和实施调度的规则.

2.3 Highcharts

Highcharts 是一个用纯 JavaScript 编写的图表库, 能够很简单便捷的在 web 网站或是 web 应用程序添加具有交互性功能的图表[8]. Highcharts 不需要像 Flash 和 Java 那样需要插件才可以运行, 而且运行速度快. 另外 Highcharts 还有很好的兼容性, 能够支持当前大多数 PC 浏览器, 以及主流手机、平板设备的浏览器.

3 系统实现

3.1 运维指标定时采集

运维指标采集程序运用 Quartz 框架, 实现多个数据库的定时读取、计算任务. Quartz 实现作业调度需要进行构建 Scheduler, 实现 Job 作业、制定 Trigger 规则.

(1) 构建 Scheduler

Quartz Schduler 是任务调度器, 所有的定时任务都通过 Schduler 注册; 到达指定时间点时, 由 Schduler 调用该 Job 的 execute()方法. Schduler 可以使用默认创建也可以通过配置文件创建. 后者通过 web.xml 文件引入 Quartz 框架功能以及 quartz.properties 配置文件, 再由配置文件创建 Schduler. 其中关键代码内容如下:

```
<servlet>
<servlet-name>QuartzInitializer</servlet-name>
    <servlet-class>
 org.quartz.ee.servlet.QuartzInitializerServlet
    </servlet-class>
    <init-param>
<param-name>config-file</param-name><param-value>/
quartz.properties</param-value>
    </init-param>
    <init-param>
  <param-name>shutdown-on-unload</param-name>
        <param-value>true</param-value>
     </init-param>
    <init-param>
<param-name>start-scheduler-on-load</param-name>
        <param-value>true</param-value>
    </init-param>
    <load-on-startup>2</load-on-startup>
</servlet>

# Configure Main Scheduler Properties
org.quartz.scheduler.instanceName = QuartzScheduler
org.quartz.scheduler.instanceId = AUTO
```

(2) 实现 Job 作业

Job 的实现需继承 Quartz 框架中的 Job 类, 实现父类的 execute 函数, 在该函数中调用具体的程序内容, 使该任务按照 Quartz 机制执行. 本文中根据不同的任务执行规则, 建立了五个定时任务类, 包括 Job5min、Job10min、Job1hour、Job1day、Job1week、Job1month, 定期执行实时数据及历史数据的计算, 将数据写入运维指标数据库. 如 Job1day 定时任务中执行集群、存储、网络、用电量等多条计算任务.

```
public void execute(JobExecutionContext context)
throws JobExecutionException
{   this.executeCluster1day ();
    this.executeCluster1daymaxmin();
    this.executeStorage1day ();
    this.executeStorage1daymaxmin();
    this.executeNet1day ();
    this.executeNetSec1day ();
    this.executeUPS1day ();
    this.executeYDL1day ();
    this.executeKT1day ();
    ……      }
```

(3) 制定 Trigger 规则

任务内容实现后, 需要创建 Quartz Trigger, Quartz Trigger 负责定义任务的执行时间, 时间到了就触发对应的 Job 中的 execute 函数去执行. 在 job.xml 文件中利用<job-detail>标签指定任务类, 利用<trigger>标签为该任务类配置触发器, 并将 Job 与相应的 Trigger 进行绑定. trigger 执行时间规则由 cron 语句进行描述, 如 Job1day 对应的配置如下:

```
<job>
    <job-detail>
     <name>Job1day</name>
     <group>DEFAULT</group>
     <description>
         A job that repeat-interval is 1 day
     </description>
     <job-class>
             com.quartz.Job1day
     </job-class>
  </job-detail>
  <trigger>
    <cron>
         <name>Job1dayTrigger1</name>
         <group>DEFAULT</group>
         <job-name>Job1day</job-name>
         <job-group>DEFAULT</job-group>
<cron-expression>0 0 2 * * ?</cron-expression>
<!-- 每天凌晨 2 点触发一次 -->
       </cron>
  </trigger>
</job>
```

3.2 运维指标管理系统

运维指标实时数据及历史数据定时采集进入数据库后，通过运维指标管理系统实现图形化界面，对大量的指标数据进行展示、分析.

运维指标 Web 管理系统采用 MVC 模式，利用轻量级的 SSI 框架(Struts+Spring+IBatis)实现数据的访问、业务逻辑、用户交互. 并利用 Highcharts 实现大量数据的图形化展示.

系统读取页面表现层设定的数据查询条件，提交到 Struts 控制层的 action 中进行用户操作响应. action 调用 Spring 业务逻辑层的 Service 执行数据检索. 通过执行 Service 中的查询函数，调用 iBatis 持久层，实现数据库连接和数据读取、计算. 以集群年报为例，按用户需求调用不同的 Service，实现按天、按周、按月的多维度报表展示，其中部分关键代码如下.

```
public class ClusterAction extends BaseAction {
public String    clusterYear()
 {     ICluster1dService cluster1dService;
      ICluster1dmaxService cluster1dmaxService;
      ICluster1dminService cluster1dminService;
         ICluster1wService cluster1dService;
      ICluster1wmaxService cluster1dmaxService;
      ICluster1wminService cluster1dminService;
      ICluster1mService cluster1mService;
      ICluster1mmaxService cluster1mmaxService;
      ICluster1mminService cluster1mminService;
      ……

this.setavgList(cluster1dService.queryForList2(paramMa
p));

this.setmaxList(cluster1dmaxService.queryForList2(para
mMap));

this.setminList(cluster1dminService.queryForList2(para
mMap));
      ……
      jsonData.put("Xdata", xdataList);
      jsonData.put("cpudata", cpulist);
      ……
      }
      }
```

获取的数据传回 action 后，存入了 json 数组中，将其传回 jsp 页面中，按照 Highcharts 规则编写 Java Script，为图形提供 X、Y 轴数据. 例如集群 CPU 利用率平均值及波动范围图的关键代码如下:

```
var chart;
$(function() {
$('#container').highcharts({
      chart: {zoomType: 'xy'},
      title: {text: 'CPU 利用率(%)'},
      xAxis: [{categories: ${XAxisCategories},
            labels: {rotation: -45}}],
      yAxis: [{labels: {
                formatter: function() {return
this.value ;}},
            title: {text: 'CPU 利用率(%)',} }],
      series: [{
            name: 'CPU 利用率平均值',
```

```
            type: 'spline',
            data: ${Y_CPU},},
{name: '波动范围',
            color: 'red',
            type: 'errorbar',
            data: ${Y_CPUmm},      }]
});
});
```

在该 jsp 页面的 form 元素中添加“container”对象，即可显示此图形，效果如图 5.

图 5 运维指标管理平台数据展示

4 结论

本文提出的“运维指标管理系统”分析了云计算中心对运维数据的需求，通过对多个独立监控系统数据的定时采集，将重要的运维指标监控数据集中存储于一个数据库中，并计算生成大量具有实用价值的历史统计数据. 最后建立 Web 图形化界面系统，将运维指标数据进行多方位展示.

从实现效果来讲，该平台将分散的运维指标数据进行统一管理，使得用户能够了解云计算中心所有设备的运行情况，能够快速检索获取所关注的数据，并快速制作报表，为日常运维和决策提供量化依据. 系统选择的开发模式及工具具有较强的兼容性，能支持 PC 及移动设备，便于随时随地对云计算中心情况进行查看.

参考文献

1 王如荣.常见网络监控技术及分析.电脑知识与技术,2011,33(7):8168–8169.
2 范军涛,李国庆.实用的机群监控系统.计算机工程与设计,2008,29(1):190–192.
3 李超,梁阿磊,管海兵,李小勇.海量存储系统的性能管理与监测方法研究.计算机应用与软件,2012,29(7):78–80.
4 高军,陈伟斌,孙成柱.通用性机房集中监控系统的设计与实现.计算机工程与设计,2011,32(4):1499–1502.
5 田娟,徐钊.基于J2EE的MVC设计模式的分析与思考.计算机与现代化,2010,10:54–58.
6 顾宁,杨灵敏,李佳佳.基于XML和Web Service的数据交换在分布式店里企业中的应用.计算机工程与科学,2010,8:98–99.
7 王崟,董志勇.基于Quartz的网管系统任务调度的实现.电脑开发与应用,2011,24(7):23–24.
8 张建军,刘虎,倪芳英.基于 SSH 与 Highcharts 整合架构的 Web 应用研究.计算机技术与发展,2013,23(9):245–247.