文章编号:1671-4598(2014)07-2199-04　　中图分类号:TP397　　文献标识码:A

云平台调度监控系统的设计与实现

王会霞，史丽燕
(河南广播电视大学 信息工程系，郑州 450046)

摘要：传统的云平台调度系统根据节点请求排队分配计算资源，效率低下；为提高云计算资源利用率，改进云软件服务能力，设计并实现了基于/proc文件面向云软件服务的监控与调度系统，系统采集并汇集节点内核信息，同时设计了分层调度，使系统可根据监控结果以及节点上传的请求情况来合理分配资源的调度系统，提高了系统资源利用率；最后进行了仿真实验，实验结果表明文章设计的系统资源利用率比传统系统提高了48%，具有极强的实际应用价值。

关键词：云平台；监控；资源调度；层次算法

Design and Implementation of Cloud Platform Scheduling Monitoring System

Wang Huixia, Shi Liyan
(Faculty of Information Engineering, Henan Radio & Television University, Zhengzhou 450046, China)

Abstract: Traditional cloud platform computing resource scheduling system based on node queue requests distribution, low efficiency. For improving the cloud computing resource utilization, and improving the cloud software service ability, designed and implemented based on the/proc file for cloud software service monitoring and dispatching system, system to collect and aggregate core node information, design the hierarchical scheduling at the same time, make the system according to monitoring results as well as the request of the node to upload to allocate resources scheduling system, improve the system resource utilization. Finally has carried on the simulation experiment, the experimental results show that compared with the conventional system, resource utilization of proposed system in this paper increased by 48%, with strong practical application value.

Keywords: cloud platforms; monitoring; resource scheduling; hierarchical algorithm

0 引言

云平台以及SaaS高速发展伴随着许多问题亟待解决：系统资源利用率低；开发过程繁琐；数据安全性存在问题；缺乏有效的计费机制等。这些问题的核心都是缺乏对集群中资源数据的有效监控以及调度[1]。现有的调度方法根据用户请求的等待队列，按照先到先服务的原则进行响应，这种调度的算法未考虑用户差异性，导致用户体验性差，并产生服务资源闲置[2]。本文使用/proc文件系统，设计了一套包含监控模块和算法模块的调度系统[3-5]。系统实现了对节点应用程序资源消耗的全方位监控，从而提供可靠的数据依据，并根据监控数据结果，而且实现了基于用户请求公平性、资源负载均衡的分层次高效调度方案[6-7]。

1 系统构成

云平台调度管理系统由现场监控、数据传输和调度管理三部分构成。现场监控部分由传感器检测模块、内核数据采集模块、自动识别模块、辅助照明模块、数据控制汇集模块和其他辅助模块组成；调度管理包括Web监控模块、调度处理中心模块和结果显示模块组成，调度管理系统的总体架构如图1所示。

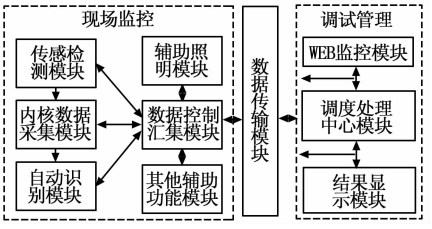

图1 监控调度管理系统总体架构图

监控调度管理系统首先由现场监控部分进行工作，通过传感检测模块对数据进行检测和预处理，之后由内核数据采集模块进行数据采集，对过自动误别模块，由数据控制汇集模块进行整理，通过无线网络ZigBee技术，借助远程通信GPRS传输到调度处理中心，进行数据的实时调度管理，最后由显示模块及时的把处理结果公布出来，以便决策者作出相应决策。

2 系统关键硬件模块结构设计

2.1 传感检测模块的硬件设计

传感检测模块主要负责传感器信号的接收与发送，为内核数据采集做准备。传感器信号收发选取了T1公司的CC1100，这种芯片的主要性能是功耗低且体积小，操作简单灵活，这种模块只需要很简单的外围电路就可以支持其进行工作，图2是模块以及外围电路硬件设计图。

如图2所示，DMX512设置了一个精确的输出偏电流，芯片STC12C5410AD组成一个平衡转换器。电路中的电容设计

收稿日期:2014-02-11；　**修回日期**:2014-04-11。
基金项目:2014年河南省攻关计划项目(1421002210041)；2014年河南省教育厅科学技术研究重点项目(14B520012)。
作者简介:王会霞(1977-)，女，河南郑州人，硕士研究生，讲师，主要从事计算机应用、网络方向的研究。

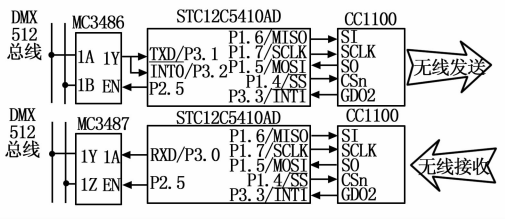

图 2 传感器模块硬件设计电路图

为了退偶，都尽量靠近电源引脚设计。无线收发模块与控制单片机都需要外围的电源供电，选取了 MC3486 进行电压转换，通过转换以后输出的电压变为 3.3 V 稳定电压。这种电路可以为收发模块、S3C2410 单片机同时供电，电源的输入输出都要经过滤波电容的处理，选取了 E1 与 E6 极性电容，这种模式能保证电源有足够的上电速度。

2.2 数据控制汇集模块的硬件设计

为了监控数据传输能达到监控高效、监控节点管理方便、数据收集效率高的目的。本文设计的数据控制汇聚模块采用集中式数据汇聚。设计中使用一个中间节点对数据采集节点收集到的数据进行集中的处理。这种汇聚方式相较分层的汇聚方式，部署起来可操作性更强，而且数据收集的效率更高。监测节点的硬件组成如图 3 所示。

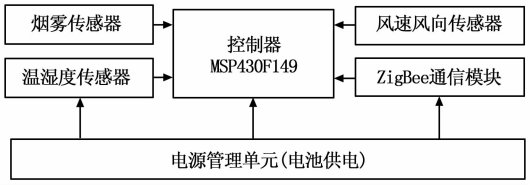

图 3 监测节点硬件组成

在监控节点中引入客户端数据汇聚进程，客户端数据汇聚进程负责系统性能数据的汇集和发送。数据基部节点的设计如图 4 所示，在服务器上引入数据汇聚进程，数据汇聚进程负责建立的数据汇聚集群。

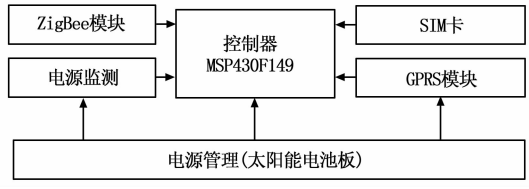

图 4 数据基站节点硬件结构

服务端数据汇聚进程通过控制命令的形式控制客户端进程的建立和数据汇聚集群的建立，并接收和处理客户端节点传输的性能数据[8]。客户端进程负责从节点获取采集到的性能数据，并对性能数据进行包装然后发送。

2.3 数据传输模块的硬件设计

数据控制汇集模块完成数据的汇集整理工作后，要通过网络传输到调度管理中心进行数据的调度处理。本系统采用无线网络传输，以克服山区等布线困难的限制。无线网络通信采用了 CC2430ZigBee 协议栈，Chipcon 公司生产的 Zigbee 管理芯片 CC2430，这是一款最为常用的无线传感网络管理芯片，芯片具有强大的内部实现能力，只需要很简单的外围电路就可以进行芯片的管理，电能的损耗较低适合大范围的应用实现。因为在 CC2591 芯片内部集成了 RF 匹配网络，本文将两种芯片进行组合扩展设计出如下的无线通信硬件设计电路图如图 5 所示。

图 5 中的 R1 与 R2 为两个偏置电阻，分别保证晶振中的合适的工作电流与电流的参考发生器。主控芯片的主时钟电路经过 XTAL1 与两个负载电容 C14 \ C15 提供，各个定时器都工作在该时钟中，时钟的控制通过 CLKCON 实现。内部的

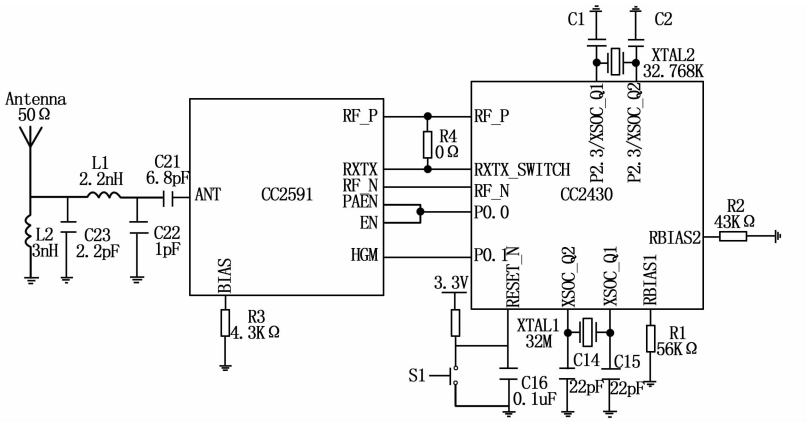

图 5 数据传输模块的硬件设计电路图

RC 振荡器也可以产生时钟，CC2591 的 EN 引脚、HGM 引脚、经过单片机的控制后接入到 CC2430 的闲置 I/O 口，通过控制这些接口的电平来控制 CC2591 的接收模式。

3 软件设计

3.1 内核数据采集模块软件的软件设计

内核数据采集模块主要是在传感检测模块完成信号检测后，对数据进行采集。数据采集主机以 Uboot 作为 BOOTLOADER，移植 Linux－2.6.38 作为嵌入操作系统，制作/proc 虚拟文件系统。由于/proc 文件系统是伪文件系统，它只利用 RAM，而不占 ROM，使用简单。它以文件系统的方式为访问系统内核操作提供接口，用户可以方便地通过/proc 得到系统的信息，所以使用/proc 建立 Linux 内核监控更加稳定。移植 Linux 操作系统之后，因 DM9000 驱动程序已经比较成熟，因此只需重点开发内核数据采集模块驱动程序、3G 模块的 USB 驱动程序，在有了这些驱动程序之后，再编写 SOCKET 通信程序完成 3G 通信，编写多媒体编解码程序完成视频编解码，编写 UART 程序完成 ZigBee 通信程序。

要进行内核数据进行高效的采集，首先要完成 TVP5150 在 Linux 的驱动程序。驱动程序要完成 TVP5150 的初始化，要通过 12C 总线配置其寄存器。如配置其选 择 数据采集端口输 入通道，当输入为 Composite 信号时，可以选择通道 A 或者通道 B，当输入为 S－Data 信号时，通道 A 输入 Luminance 信号，通道 B 输入 Chrominance 信号。在本系统中输出格式配置为 8 位 ITURBT. 656 格式的数据输出，可以按照 TVP5150 规格书的寄存器初始化表配置。

内核数据采集程序采用内存映射方式代替直接读取。直接读取是通过内核缓冲区来读取数据，而内存映射是通过把设备

文件映射到内存中，绕过了内核缓冲区，进程可以像访问普通内存一样对文件进行访问，加快了内核数据读取的速度。内核数据采集之前要先获得设备的属性信息，并对采集窗口等进行设置。内核数据编码采用 S3C6410 内部集成的多媒体编码模块 MFC 进行 H. 264 的内核数据压缩编码。其软件设计流程如图 6 所示。

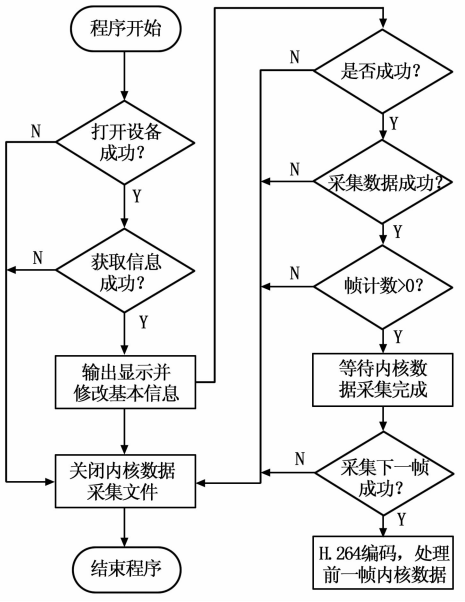

图 6　内核数据采集流程软件设计图

3.2　调度处理中心模块的软件实现

为了完成调度处理中心的有效数据调度，需要开发新的程序来与 Web 监控传输模块匹配。由于需要调度的应用有多种类型的资源消耗，本系统需将采集到的每一种资源（CPU、内存、I/O、网络）作为调度依据，并且能保证同时满足每一项资源调度的合理性。所以对应调度算法要区分层次，在主资源的基础上达到公平性。另外需要确定的是如何衡量一个节点的负载。本调度程序设计时将调度管理系统得到的集群实时的、所有的资源分量价值等同，给每一种资源一个可变层次权重。

将每一个节点的多资源等同于单资源，将这个换算之后的值作为节点的剩余资源量，资源量的大小与该节点的负载成反比关系。通过这种换算，得到每个节点的负载情况的描述。

层次程序设计的主要流程如下：

（1）第一层为对用户的选择。根据主资源最小的原则，选择所有等待队列中用户主资源占用率最小的，作为非受限调度的对象，并通过调度管理系统中相应应用的资源消耗情况更改用户主资源占用率；

（2）第二层为节点的选择。通过调度管理系统实时的节点监控数据，根据每一个节点的负载，选择一个负载最小的节点，将（1）中选定的应用程序调度到该节点运行；

（3）重复（1）（2），直到将用户的请求全部调度或者集群剩余资源不足。

层次设计中，保证调度的公平性非常复杂。本程序设计时引入了可变权重，实现了内核监控结果的量化处理，为调度的公平性创造了前提。并结合可变权重均衡资源的调度空间使系统运行可以更加流畅。软件流程如图 4 所示。

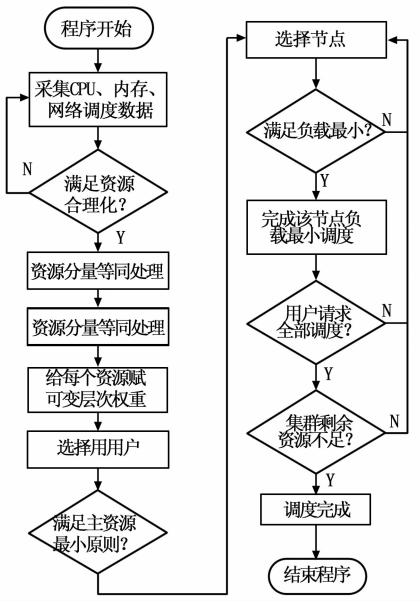

图 7　调度处理中心流程软件设计图

4　系统实验与分析

实验环境为 5 台计算机，1 台计算机模拟云服务器，其他 4 台模拟客户端，操作系统为 Debian Lenny，CPU 为 Intel（R）Core（TM）i5－2400 @3.10 GHz，4 GBRAM。

运行调度系统，首先在计算机上运行单进程应用 application，启动进程监控程序对 application 进行监控，启动命令如下：

test02@debian：～/monitor＃./procmonitor1 application。调度结构如图 8 所示。

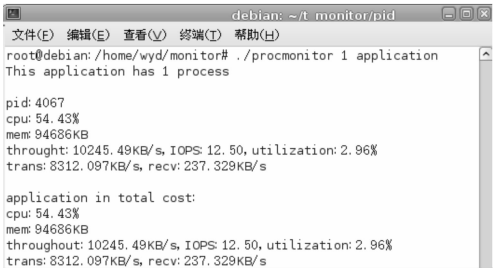

图 8　对客户端的监控结果

调度管理系统可完整管理控节点 CPU、内存、I/O 以及网络信息，表明调度管理系统运行正常。

根据客户端监控结果，执行调度程序进行层次分析，产生表 1 结果。

表 1　层次分析结果

编号	角色	CPU 利用率(%)	请求数(条/s)	层次
Test00	服务器	31	—	总机
Test01	客户端	43	20	1 层
Test02	客户端	86	20	2 层
Test03	客户端	51	5	3 层
Test04	客户端	82	5	4 层

表 1 中，Test01 的高请求数，低 CPU 利用率，被合理划

入第一层次，Test04的低请求数，高CPU利用率需求较低，被划入第四层次。可知分层调度可以根据客户端情况合理分配资源，在保证客户端得到良好服务的基础之上，保证服务高效性。

为验证基于节点监控的调度系统优于常规排队调度系统，设计实验，以1 h内的服务器平均CPU利用率和服务器传输的数据量作为指标，衡量常规排队、无监控调度以及监控调度三种调度模式的工作效率，其中无监控调度模式，单纯根据客户端请求数来进行资源分配。

表2结果表明，对客户端内核运行情况的监视结合分层算法相对于无监控的调度模式以及常规排队模式可有效提高云模拟服务器的系统利用率提高，降低资源的闲置，提高云平台的服务能力。

表2 服务器在不同调度模式下利用情况

模式	服务器CPU平均利用率(%)
常规排队模式	24
无监控调度模式	32
监控调度模式	72

通过表3的数据可以清晰看出，监控调度模式有效提升服务器的服务效率，1 h内服务器数据传输量达到6.8 GB，远高于无监控调度模式与常规排队模式。

表3 不同调度模式接收服务数据量

模式	服务数据量(GB)
常规排队模式	2.1
无监控调度模式	3.3
监控调度模式	6.8

通过对比试验，监控调度模式在提升平台CPU利用率与平台服务数据传输效率上均高于传统的云调度算法，证明了基于节点进程监控的云平台调度系统具有现实可行性。

5 结束语

本文设计并实现了基于/proc节点进程监控的云平台调度管理系统。最后模拟运行调度系统，证实内核信息采集模块工作良好，并与常规排队调度和无监控调度进行对比证实了监控调度系统可有效提高服务器的CPU平均利用率，并在固定时间内提供更多的数据服务，从而证明了基于节点进程监控的云平台调度系统具有现实可行性。

参考文献：

[1] Sottile M, Minnich R. Supermon: a high-speed cluster monitoring system [A]. Proceedings of Cluster 2002 [C], 2002.
[2] 段琳琳. 共享平台数据属性重合下的LDAP多身份认证 [J]. 科技通报，2013，8：91-93.
[3] Laadan O, Baratto R A, Phung D B, et al. DejaView: A Personal Virtual Computer Recorder [C], 2007.
[4] 怀进鹏，李 沁，胡春明. 基于虚拟机的虚拟计算环境研究与设计 [J]. 软件学报，2007，18 (8)：2016-2026.
[5] 韩燕波，王桂玲，等. 互联网计算的原理与实践：探究网格、云和Web X.0背后的本质问题和关键技术 [M]. 北京：科学出版社，2010.
[6] 范金阳. 面向虚拟网络的数据包捕获及展现系统的研究与实现 [D]. 北京：北京航空航天大学，2011.
[7] 梁东莺，高 潮. 云计算及其应用 [J]. 计算机测量与控制，2011，19 (8)：1958-1961.
[8] Richard Stevens W, Stephen A. Rago. UNIX环境高级编程 [M]. 北京：人民邮电出版社，2009.

(上接第2198页)

去驱动图形，用户输入尺寸后，单击确定，便可以在Autocad里得到承重减震装置的正视图，如图6所示。

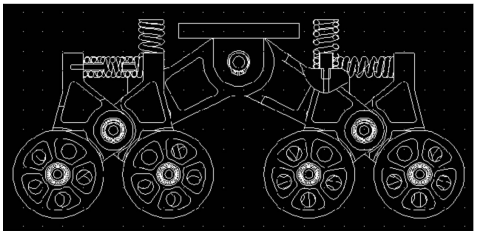

图6 承重减震装置正视图

5 结语

为了使新型反恐防暴机器人在各种复杂地形条件下都能够平稳行驶，设计了一种承重减震装置，通过在Adams中的仿真分析，该承重减震装置对崎岖路面上的行驶震动有明显的缓冲吸收效果。为了提高摆臂弹簧的使用寿命，以摆臂弹簧最小弹力值为最优目标，笔者对承重减震装置进行了优化，得到最优尺寸。笔者又以AutoCAD2010为依托平台，使用VS2010平台的C++语言编写软件，利用ObjectARX2010对该承重减震装置所在的子系统进行了二次开发，可实现参数化驱动图形。ObjectARX与MFC对话框的结合，使编写的软件留有友好的人机交互界面，设计者在对话框中输入最优尺寸，便可得到承重减震装置的正视图、俯视图、左视图，大大提高了设计工作的效率。

参考文献：

[1] 衣 勇，宋雪萍. 机器人仿真研究的现状与发展趋势 [J]. 机械工程师，2009，(7)：63-65.
[2] 王月梅，周义清，常列珍. 履带车辆的一种动力学建模 [J]. 华北工学院学报，2005，26 (3)：164-166.
[3] 韩宝坤，李晓雷，孙逢春. 履带车辆动力学仿真技术的发展与展望 [J]. 兵工学报，2003，24 (2)：246-249.
[4] 郭卫东. 虚拟样机技术与ADAMS应用实例教程 [M]. 北京：北京航空航天大学出版社，2008.
[5] Zhang L, Wang L, Wang F, et al. Gait simulation of new robot for human walking on sand [J]. Journal of Central South University of Technology, 2009, 16 (6): 971.
[6] 喻 涛，李文蔚. ADAMS约束问题讨论 [J]. 机械，2007，34 (12)：50-51.
[7] 张 帆. ObjectARX开发实例教程 [Z]. 北京：智帆高科科技有限公司，2007.
[8] 甘 辉. 利用ObjectARX与MFC实现应用程序开发 [J]. 通用机械，2007，(1)：65-87.
[9] 李长勋. AutoCAD ObjectARX程序开发技术 [M]. 北京：国防工业出版社，2005.