Automated Feedback-Based Vertical Elasticity for Heterogenous Soft Real-Time Workloads

Yu-An Chen*, Andrew J. Rittenbach†, Geoffrey Phi C. Tran†*, John Paul Walters†, and Stephen P. Crago*†

*Department of Electrical Engineering
University of Southern California, Los Angeles, CA 90089
Email: {chen116, geoffret}@usc.edu

†Information Sciences Institute
University of Southern California, Arlington, VA 22203
Email: {aritten, gtran, jwalters, crago}@isi.edu

***Abstract*—Cloud computing provides a virtualized platform for running various services, including soft real-time applications such as video streaming. To meet application's real-time requirements, system resource is often over allocated which results in system underutilization. To solve this problem, we present, ANCHORS, a framework that allows user to implement their own resource allocation algorithm that utilizes real-time performance feedback. We then present two resource allocation algorithms that are based on TCP congestion control and PID control respectively. We implement ANCHORS on a video streaming object detection application and show that ANCHORS can save more than 50% CPU utilization while still meeting deadlines 97% of the time. Finally, we present a Stride scheduling based algorithm to maintain application's real-time performance when the total resource demand is greater than systems available resource. Through experiments, we show that ANCHORS can avoid more than 50% of deadline misses in an oversubscribed system where no deadlines are met without ANCHORS.**

***Keywords*-soft real-time; vertical elasticity; feedback-based controller; resource allocation**

I. Introduction

sdf

II. Related Work

DART-C,POET, other POET, Adaptive Management of Virtualized Resources in Cloud Computing Using Feedback Control

Priority Based Dynamic Resource Allocation in Cloud Computing with Modified Waiting Queue

Adaptive Control of Virtualized Resources in Utility Computing Environments

Active Resource Allocation Management in Virtual Cloud Instances

Dynamic virtual machine allocation policy in cloud computing complying with service level agreement using CloudSim

Vertical and horizontal elasticity for dynamic virtual machine reconfiguration

Automatic memory-based vertical elasticity and oversubscription on cloud platforms

III. Background

Openstack
Xen Hypervisor
Soft Real-time
Perfomance monitoring:Hearbeat
Resource allocatoin/scheduling

IV. Anchors Framework

The main idea of Anchors is to dynamically allocate CPU resource to VMs by monitoring each VMs real-time performance as shown in figure meow. In this section, we will go over each Anchors frameworks module in detail.

A. Application's Real-Time Perfomance Feedback

1) Heartbeat API: We modified Heartbeat API to be used in Python.

2) Inter-Domain Communication: We utilize Xenstore for VM to send its heartbeat information to Dom0.

B. VM Monitor

1) Xen Scheduler: RTDS and credit scheduler

V. Controller Algorithm

1) AIMD: range
2) APID: log
3) Stride Scheduling: meow

VI. Evaluations and Discussion

A. Overhead

In this section we will measure the latency introduced in the application to record a heartbeat

Methodology: overhead

B. Application

Andy

C. Experiments

We will go over three different experiments to evaluate Anchors.

1) *Experiment 1: Monitor across Different Scheduler:*

In this experiments, we want to evaluate Anchors' ability to monitor and save CPU resource when VMs are running on different subset of cores with different scheduler.

- VM Configuration:
  Anchors requires the hypersvisor scheduler to be able to assign specific amount of CPU resource to the VMs. With one VM, under RTDS Scheduler, the amount CPU resource allocated to the VM can be specified by assigning proper period and budget. But for Credit Scheduler, which allocates CPU resource based on weights, it is required to create a dummy VM under the same CPU pool so precise CPU resource allocation can be achieved by assigning proper weights to the dummy and the main VM. To have a fair comparison, we also create a dummy VM for the CPU pool that runs RTDS Scheduler. Both dummy were assigned 1% of the CPU utilization before every trial. The complete VM configuration are shown in table I

Table I: My caption

	# of VCPUs	**Scheduler**	**Initial CPU Utilization**
VM1	5	RTDS	99%
Dummy1	5	RTDS	1%
VM2	5	Credit	99%
Dummy2	5	Credit	1%

- Application:
  In this experiment, we run the object detection application for on both VM1 and VM2. The duration of each trial is 120 seconds, the first 40 seconds are running under high workload mode, between 40-80 seconds the workload is switched to light, and finally between 80-120 seconds the workload is switched medium, as shown in figure meow.
- Methodology:
  For each scheduler, three different controllers, static, AIMD and APID are applied. The CPU utilization and FPS data are collected for analysis.
- Result:
  hi

2) *Experiment 2: Monitor across Multiple VMs:*

In this experiment, we use Anchors to monitor VM1 and VM2 that are running on the same set of PCPU with RTDS scheduler, and shows VM3 is able to benefit from the saved CPU utilization.

- VM Configuration:
  There are three VMs fo this experiment
- Application:
- Methodology:
- Result:

Experiment 3: Monitor under Overloading System

In this experiment, we show that Anchors is able to improve the overall real-time performance when the system is not able to provide enough CPU resource to satisfy the total demand from VMs.

- VM Configuration:
- Application:
- Methodology:
- Result:

VII. Conclusion and Future Work

[1]

container, multi-resource, multi-server and complex learning controller

1) We propose a task model that provides a platform to develop load balancing algorithm that does not require the knowledge of how an individual task is scheduled by moving the scheduling problem from device level to server level. It is impractical to obtain the scheduling information about each task because the traffic of the network make the arriving order of tasks a stochastic process and the problem size increase more rapidly.
2) We propose an optimization problem formulation for load balancing that minimizes deadline misses and total runtime for connected car system in fog computing.

References

[1] G. Karagiannis, O. Altintas, E. Ekici, G. Heijenk, B. Jarupan, K. Lin, and T. Weil, "Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions," *IEEE Communications Surveys Tutorials*, vol. 13, no. 4, pp. 584–616, Fourth 2011.