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 satisfy an applications real-time requirements, CPU resources are often allocated with the amount that satisfies an applications highest utilization, resulting in system underutilization. To solve this problem, we present ANCHORS, a Xen framework that allows users to implement their own resource allocation algorithm that utilizes real-time performance feedback from the applications that are running in the VMs. We then present and compare two example resource allocation algorithms that are based on TCP congestion control and PID control respectively. We apply ANCHORS to a video stream 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 control algorithm to maintain applications real-time performance when the total resource demand exceeds the systems available resource. Our results 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; Xen**

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

– Application:
  In this experiment, we run the object detection application for on both VM1 and VM2. The duration of each trial is 120 seconds. First 40 seconds, the application is running in high workload mode. The workload is switched to light mode from 40 to 80 seconds. Finally, between 80-120 seconds, the workload is switched to medium mode.
– VM Configuration:
  Anchors requires the hypervisor scheduler 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. As 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 for the main VM 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 VMs 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%

– Methodology:
  In each trial, after all four VMs are configured as shown in table I, VM1 and VM2 runs with application till it is finished. The CPU utilization and FPS data are collected for analysis. The same procedure is repeated with three different resource allocation algorithms: static, AIMD and APID.
– Result & Analysis:
  The results from experiment are shown in figure 1,2 and 3

2) Experiment 2: Monitoring Multiple VMs: In this experiment, we use Anchors to monitor VM1 and VM2, which are running on the same set of cores, and show that VM3 is able to benefit from the saved CPU utilization.

– Application:
  There are three VMs for this experiment. VM1 and VM2 run real-time application and VM3 runs non real-time application.
  o Random Video Sequence Object Detection: To

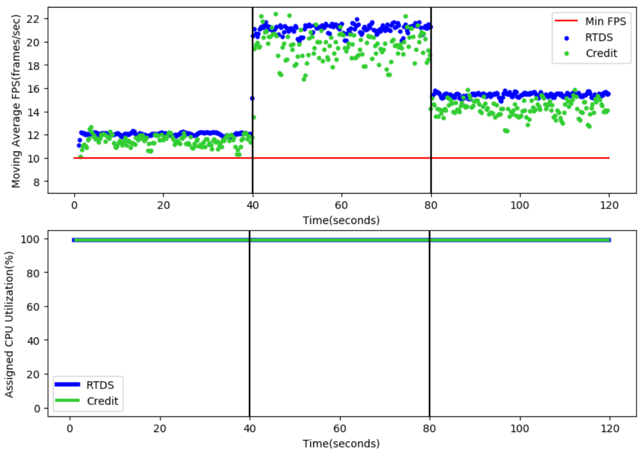


Figure 1: Static Algorithm with RTDS/Credit Scheduler

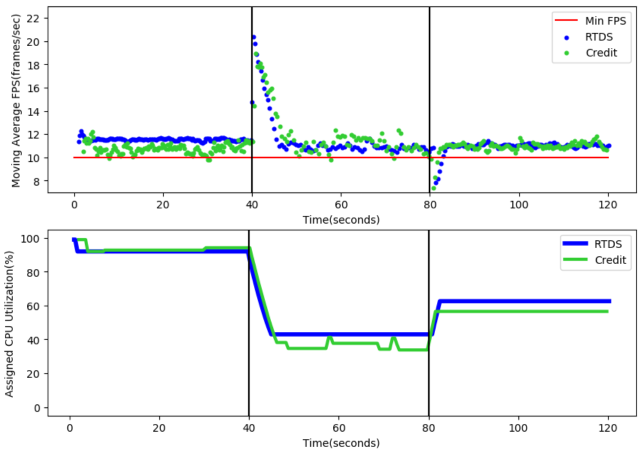


Figure 2: AIMD Algorithm with RTDS/Credit Scheduler

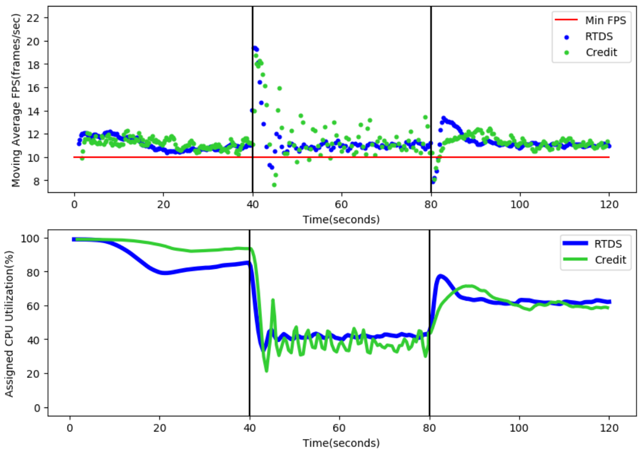
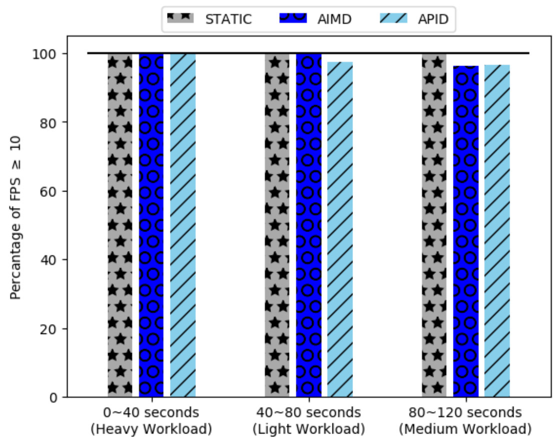
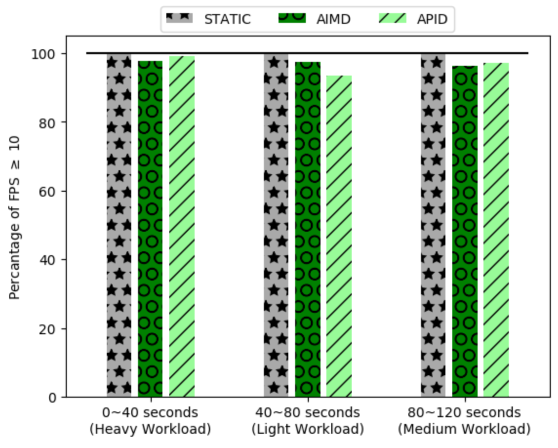


Figure 3: APID Algorithm with RTDS/Credit Scheduler

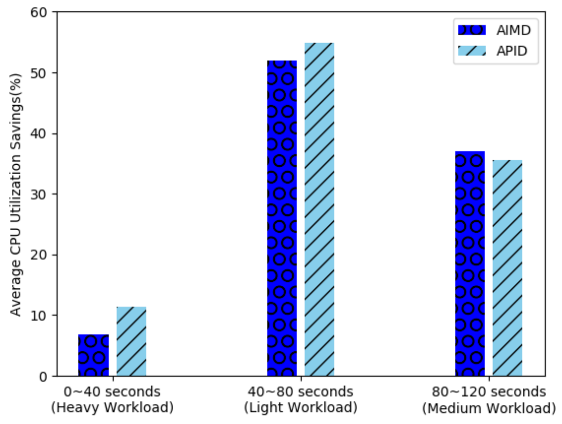


(a) RTDS Scheduler

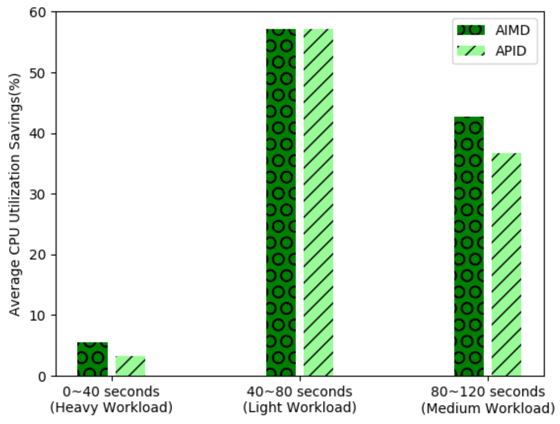


(b) Credit Scheduler

Figure 4: Soft Real-time Performance Comparison



(a) RTDS Scheduler



(b) Credit Scheduler

Figure 5: CPU Savings Comparison

mimic a real-world video surveillance application, this real-time application read in video streams and if a person is detected in the current frame, the sample frequency is switched from low to medium. If no one is detected in the current frame, the sampling frequency is switched to low frequency. Each trial is run for 120 seconds. The waiting time for a person to show in the frame follows an exponential distribution with $\beta = 30$ seconds and the time for a person to stay in the frame follows an exponential distribution with $\beta = 15$ seconds.

o Matrix Multiplication: a non real-tie application doing loops of 500 by 500 matrix multiplication.

– VM Configuration:
VM1, VM2 and VM3 all run under the same set of PCPU with RTDS Scheduler. VM1 and VM2 are initially assigned with 45% CPU utilization and VM3 is assigned with 10% CPU utilization. The configuration is shown in table II

Table II: My caption

	# of VCPUs	**Scheduler**	**Initial CPU Utilization**
VM1	5	RTDS	45%
VM2	5	RTDS	45%
VM3	5	RTDS	10%

– Methodology:
First we generate the two random sequences of frames with and without a person for VM1 and VM2 respectively. We run VM1 and VM2 with these two video sequences three times, each time with different a different controller: static, AIMD and APID. FPS and CPU utilizations are recorded for analysis.
As for VM3, during each trial, VM3 starts running

matrix multiplication when VM1 or VM2 sends their first heartbeat to Anchors. VM3 stops and reports the numbers of matrix multiplication completed when VM1 and VM1 notify Anchors that their applications are completed.

– Result & Analysis:

hi meow

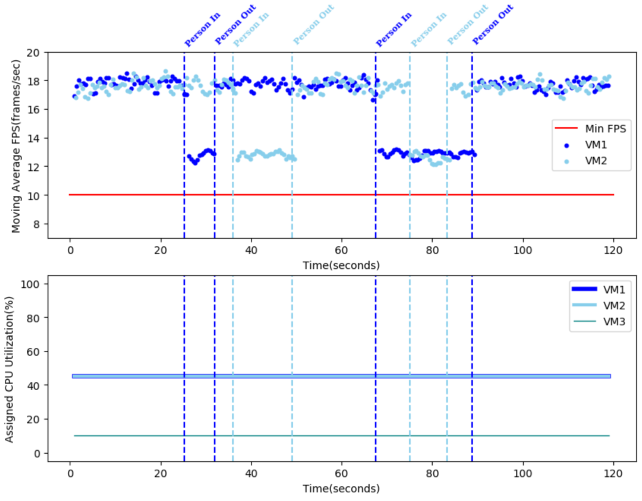


Figure 6: Static Algorithm with RTDS Scheduler

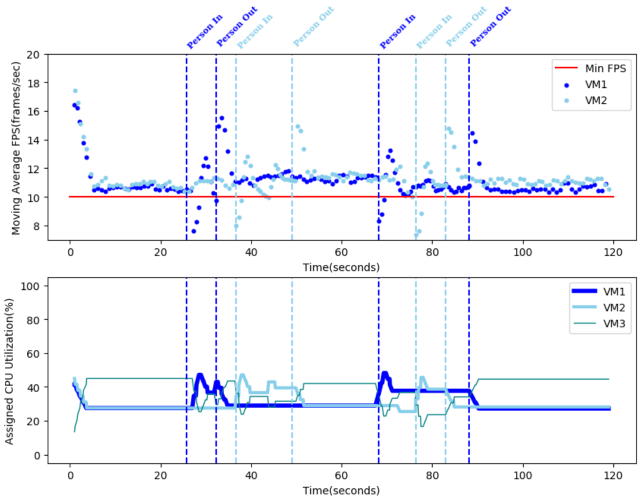


Figure 7: AIMD Algorithm with RTDS Scheduler

3) *Experiment 3: Monitoring VMs in an Overloading System:* In this experiment, we show that even under the worst case scenario, where the system does not have enough CPU resource to satisfy the total demand from the VMs, Anchors is able to improve the overall real-time performance with Stride Scheduling.

– VM Configuration:
VM1 and VM2 run under the same set of PCPU with RTDS Scheduler. Both VMs are initially assigned with

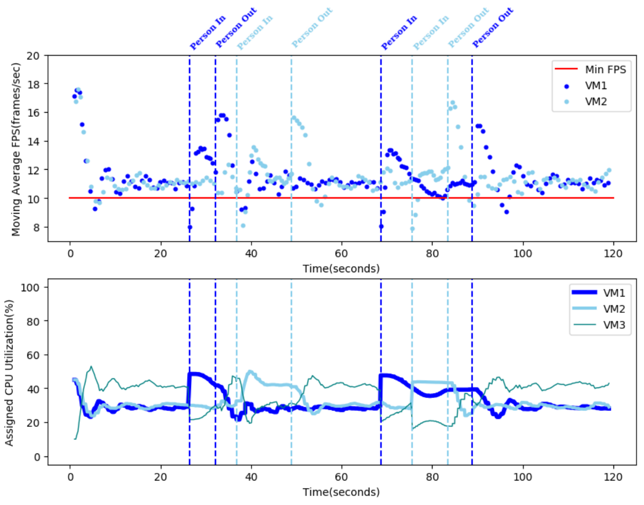


Figure 8: APID Algorithm with RTDS Scheduler

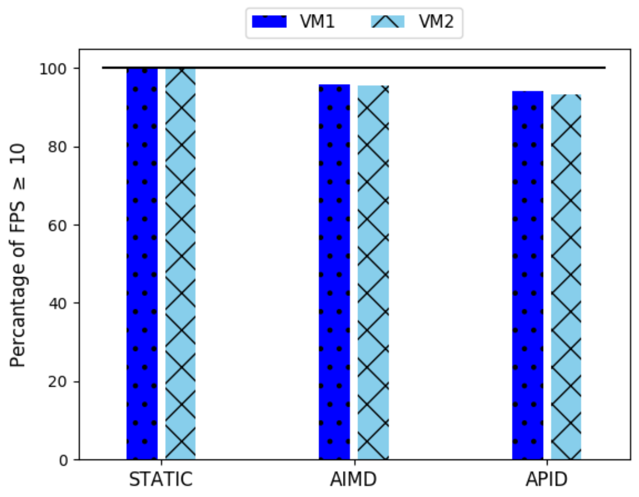


Figure 9: Soft Real-time Perforamnce Comparison

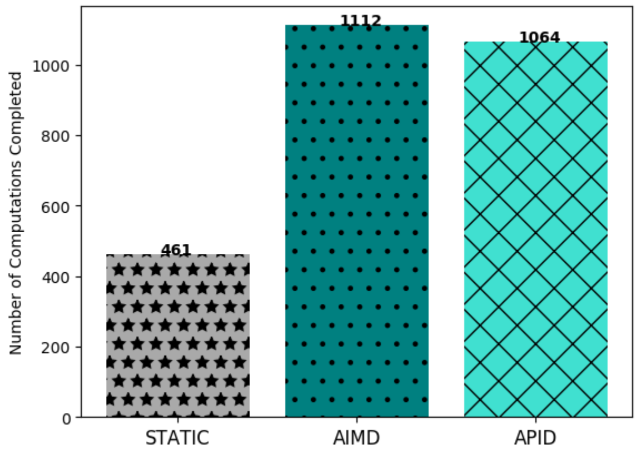


Figure 10: VM3 Computation Counts Comparison

50% CPU utilization. The configuration is shown in table III.

Table III: My caption

	# of VCPUs	Scheduler	Initial CPU Utilization
VM1	5	RTDS	50%
VM2	5	RTDS	50%

– Application:
Both VM1 and VM2 run the same object detection application but with deterministic sequence of person in the frame or not. As shown in figure meow. The reason for using deterministic sequence is because we want to emphasize how Anchors controller handle overloading system. To create a overloading system, medium sampling frequency is used when a person is not in a frame, and high sampling frequency is used when a person is detected in a frame.
– Methodology:
In each trial, after all VM1 and VM2 are configured as shown in table III, VM1 and VM2 runs with application till it is finished. The CPU utilization and FPS data are collected for analysis. The same procedure is repeated with three different controllers: static, AIMD and APID.
– Result & Analysis:

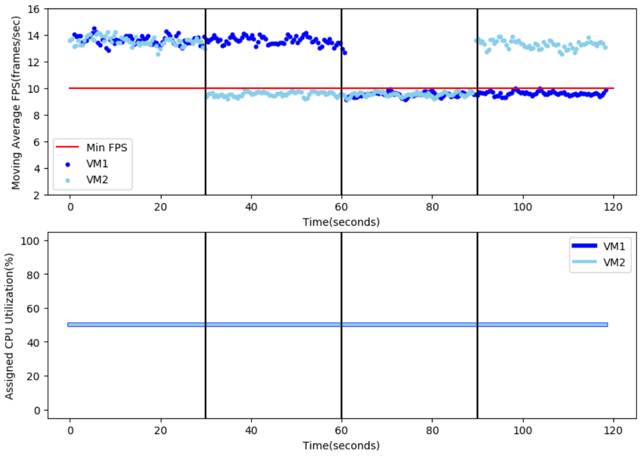


Figure 11: Static Algorithm with RTDS/Credit Scheduler

VII. Conclusion and Future Work

[1]

In this work we develop Anchors, a framework in Xen that provides user a platform at the hypervisor level to implement any CPU resource allocation algorithm that uses feedback from soft real-time application. We also present two resource allocation algorithms, AIMD and APID, that

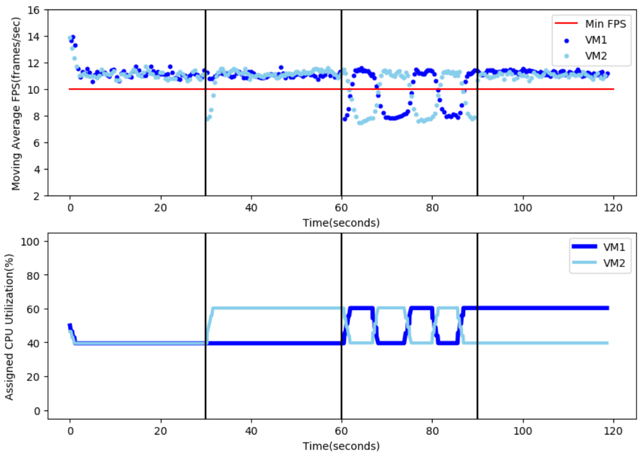


Figure 12: AIMD Algorithm with RTDS/Credit Scheduler

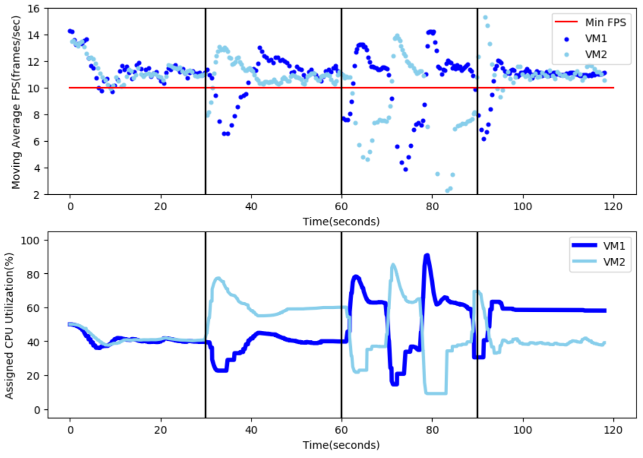


Figure 13: APID Algorithm with RTDS/Credit Scheduler

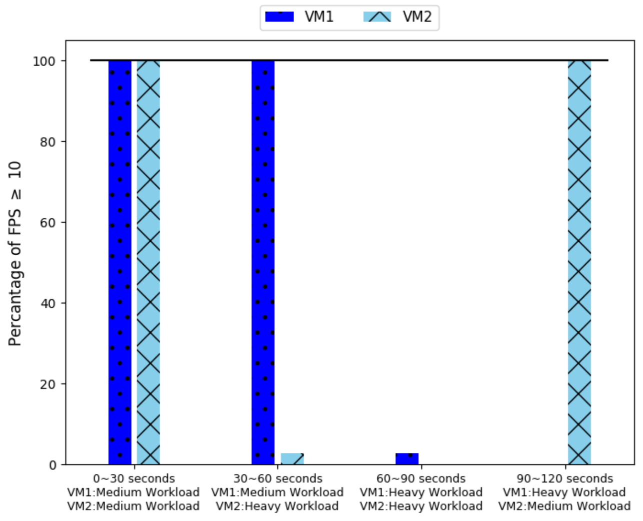


Figure 14: Static Algorithm with RTDS/Credit Scheduler

dynamically adjusting CPU utilization to the VMs based

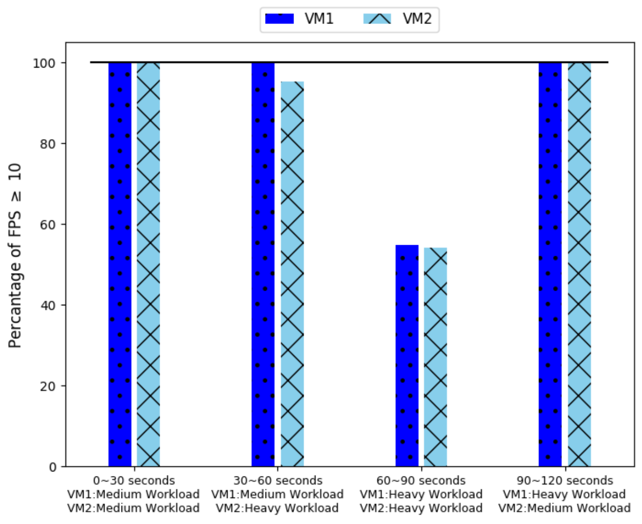


Figure 15: AIMD Algorithm with RTDS/Credit Scheduler

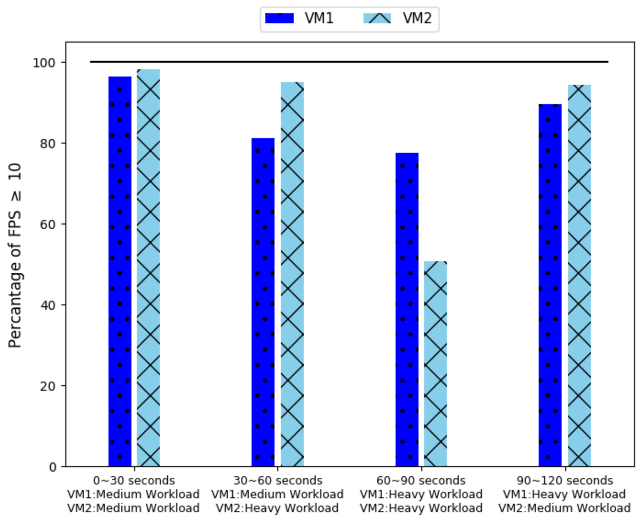


Figure 16: APID Algorithm with RTDS/Credit Scheduler

on their application's soft real-time performance. Finally, we present a Stride scheduling based control algorithm to improve real-time performance in an overloading system. Future work includes implementing Anchors in different visualization platforms such as containers or a multi-servers system. We also plan to expand Anchors to monitor over multiple resources such as memory and network. Finally, learning based methods such as Q-learning and regression can be used to develop new resource allocation algorithms.

This work was supported by the Office of Naval Research grant #N00014-16-1-2887.

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.