Autonomic Provisioning with Self-Adaptive Neural Fuzzy Control for End-to-end Delay Guarantee

Palden Lama

Xiaobo Zhou

Department of Computer Science

University of Colorado at Colorado Springs

Performance Assurance in Virtualized Data Centers

- Modern data centers must provide performance assurance for complex system software such as multi-tier web applications.
- Web applications hosted on a data center provide business value based on Service Level Agreements (SLAs), which stipulate payments or penalties as function of one or more performance metrics. (e.g response time, throughput)
- It is important to guarantee performance based SLAs for providing customer satisfaction and increasing data center revenue.
- To guarantee SLAs in the face of dynamic and unpredictable Internet workloads is a challenging issue.

Performance Assurance in Virtualized Data Centers

- One approach for performance assurance is to allocate servers based on estimated peak resource demands of hosted applications.
- Predicting the peak workload of an Internet application and capacity provisioning based on these worst case estimates is notoriously difficult.
- There are numerous documented examples of Internet applications that faced an outage due to an unexpected overload. For instance, the normally well-provisioned Amazon.com site suffered a forty-minute down-time due to an overload during the popular holiday season in November 2000 [Amazon 2000].

Autonomic Resource Management for Performance Assurance

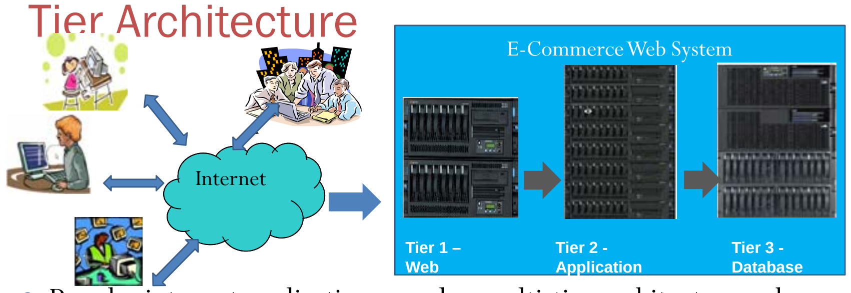
- Recently, lot of research interest focused on dynamic provisioning, which enables additional resources, such as servers, to be allocated to an application on-the-fly to handle workload increases.
- Dynamic provisioning facilitates autonomic resource management by reducing human involvement.
- Autonomic resource management of complex computing systems, such as Internet Data Centers, is critical and challenging due to rapidly growing scale and complexity of Internet services.
- Autonomic resource management needs to be Self-Adaptive to address dynamic Internet workloads and varying operating conditions of the data center.

We design a novel Self-Adaptive Neural Fuzzy Control that enables dynamic server provisioning for QoS assurance.

Virtualization

- Recently, virtualization technologies are used for server consolidation in modern data centers, resulting in significant power savings and improved resource utilization.
- Virtualization facilitates agile and dynamic server provisioning assuming there are pre-spawned pool of dormant VMs for each application in the system. As dormant VMs of an application are activated during an overload. [Urgaonkar-ACM TAAS 2008]
- Our work is focused on performance assurance of multi-tier web applications hosted on a virtualized data center.
- We apply a self-adaptive performance controller that dynamically allocates virtual machines to a multi-tier web application.

End-to-End Delay Guarantee on Multi-Tier Architecture



- Popular internet applications employ multi-tier architecture, where Tiers interact to carry out its part of overall request processing.
- E-commerce web system employ 3 tiers
  - Web Tier – Http request processing
  - App Tier – Core application functionality
  - DB Tier – Storing product catalogs and user orders
- End-to-End request delay bound is important for QoS, rather than delay at individual tier.

Percentile-Based Delay Guarantee

- Most previous research work has looked only at average performance (e.g., mean response time). However, average performance guarantees are not sufficient for many applications, in particular interactive ones.
- Providers of such services prefer percentile performance guarantees, such as that 95% of end users receive response times below an agreed upon threshold.
- For example, Amazon uses the 99.9th percentile to impose stringent requirements on the latency of their platform
- We select this metric, since the long tail of performance is more important to end users than average values.

Issues related to Percentile-Based Delay Guarantee

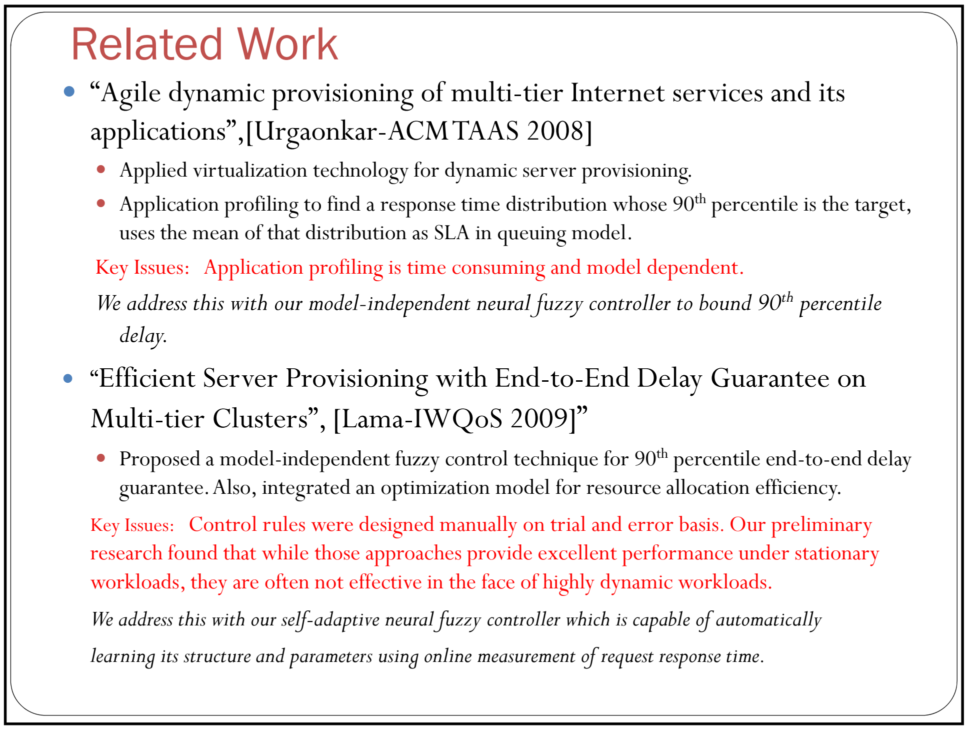
- Percentiles are not handled well by classical performance modeling techniques.
- Queueing model based server provisioning approach based can only guarantee the average end-to-end delay of requests.
- Recently, control theoretic techniques were applied for system performance control by performing linear approximation of system dynamics and estimation of system parameters.
- Such technique are not applicable to percentile based delay guarantee. Compared with the average delay, a percentile delay introduces much stronger nonlinearity to the system performance model.
- Furthermore, if the deployed system configuration or workload range deviates significantly from those used for system identification, the estimated system model used for control would become inaccurate.

A Novel Solution: Model Independence

- Our server provisioning approach addresses these issues by using a novel ***model-independent*** neural fuzzy controller.
- We apply fuzzy logic rules to determine the number of virtual servers to be added or removed for tracking the 95th percentile delay target.
- It saves the time and effort required to estimate non-linear performance model of complex web systems, in order to bound percentile-based delay guarantee.
- Traditionally, fuzzy rules are designed on trial-and-error basis or based on knowledge of human experts.

Self-Adaptiveness

- Our preliminary research found that static fuzzy rule-based approaches provide excellent performance under stationary workloads, but they are often not effective in the face of highly dynamic workloads.
- Our ***self-adaptive*** neural fuzzy controller combines the strength of machine learning and fuzzy control for ***robust*** performance assurance of Internet applications in the face of highly dynamic and unpredictable workloads.
- Our neural fuzzy controller is able to self-construct its structure and adapt its parameters to handle dynamic workload and varying performance targets.



Related Work

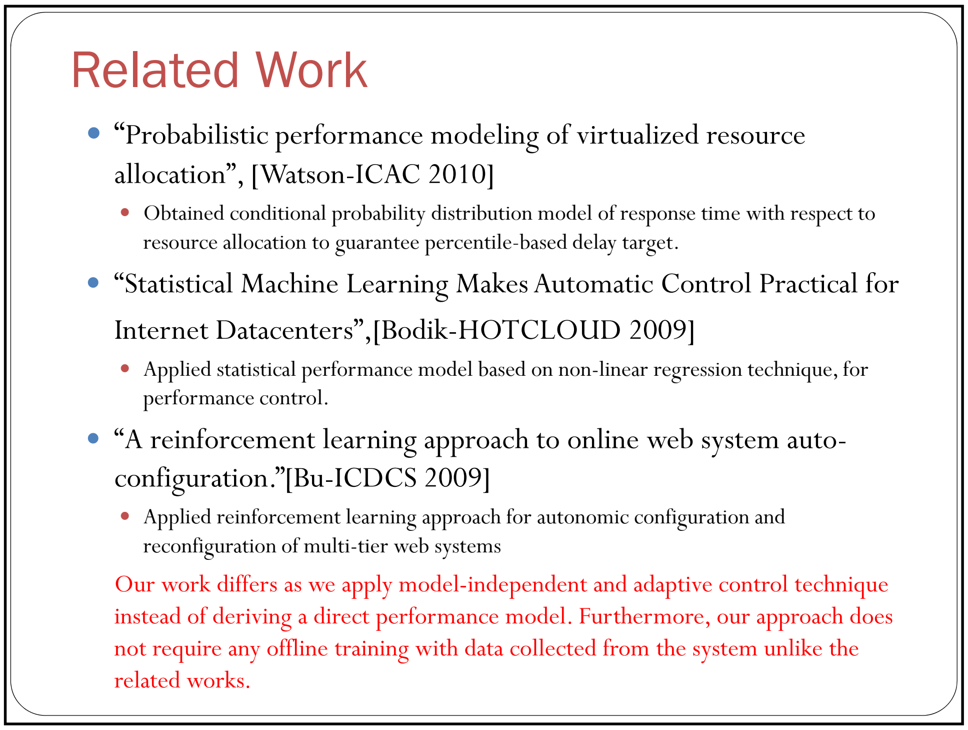
- "Agile dynamic provisioning of multi-tier Internet services and its applications",[Urgaonkar-ACM TAAS 2008]
  - Applied virtualization technology for dynamic server provisioning.
  - Application profiling to find a response time distribution whose 90th percentile is the target, uses the mean of that distribution as SLA in queuing model.

  Key Issues: Application profiling is time consuming and model dependent.

  We address this with our model-independent neural fuzzy controller to bound 90th percentile delay.
- "Efficient Server Provisioning with End-to-End Delay Guarantee on Multi-tier Clusters", [Lama-IWQoS 2009]"
  - Proposed a model-independent fuzzy control technique for 90th percentile end-to-end delay guarantee. Also, integrated an optimization model for resource allocation efficiency.

  Key Issues: Control rules were designed manually on trial and error basis. Our preliminary research found that while those approaches provide excellent performance under stationary workloads, they are often not effective in the face of highly dynamic workloads.

  We address this with our self-adaptive neural fuzzy controller which is capable of automatically learning its structure and parameters using online measurement of request response time.

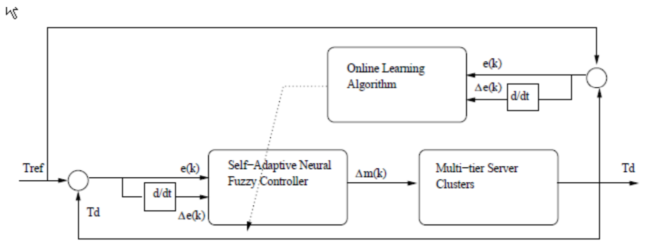


Related Work

- "Probabilistic performance modeling of virtualized resource allocation", [Watson-ICAC 2010]
  - Obtained conditional probability distribution model of response time with respect to resource allocation to guarantee percentile-based delay target.
- "Statistical Machine Learning Makes Automatic Control Practical for Internet Datacenters",[Bodik-HOTCLOUD 2009]
  - Applied statistical performance model based on non-linear regression technique, for performance control.
- "A reinforcement learning approach to online web system auto-configuration."[Bu-ICDCS 2009]
  - Applied reinforcement learning approach for autonomic configuration and reconfiguration of multi-tier web systems

  Our work differs as we apply model-independent and adaptive control technique instead of deriving a direct performance model. Furthermore, our approach does not require any offline training with data collected from the system unlike the related works.

Self-Adaptive Neural Fuzzy Controller


Online Learning Algorithm
e(k)
Δe(k)
d/dt
Tref
e(k)
d/dt
Δe(k)
Td
Self-Adaptive Neural Fuzzy Controller
Δm(k)
Multi-tier Server Clusters
Td

- T_{ref} = percentile-based request delay target (e.g 95%)
- T_d = measured percentile-based request delay
- $e(k) = T_{ref} - T_d$
- $\Delta e(k) = e(k) - e(k-1)$

Background on Fuzzy Control

- Control action based on set of fuzzy rules designed to achieve desired output. Exploits human intuitive reasoning.
- *Fuzzy Rule* (R_i)*: If* x_1 *is* A_1^i *and* x_2 *is* A_2^i *,then outcome*(R_i) *is* w_i
  - If e(k) is +LARGE and Δe(k) is –SMALL, then $(\Delta m(k))_1 = w_1$ ….. Rule 1
  - If e(k) is +MEDIUM and Δe(k) is ZERO, then $(\Delta m(k))_2 = w_2$ ….. Rule 2
- *Fuzzification* process maps numeric inputs e(k) and Δe(k) to different Linguistic/Fuzzy values using input membership functions.
  - If e(k) = 50, it is +MEDIUM with certainty 0.5 and +LARGE with certainty 0.5, where certainty is determined by corresponding membership functions, μ(MEDIUM) and μ(LARGE).

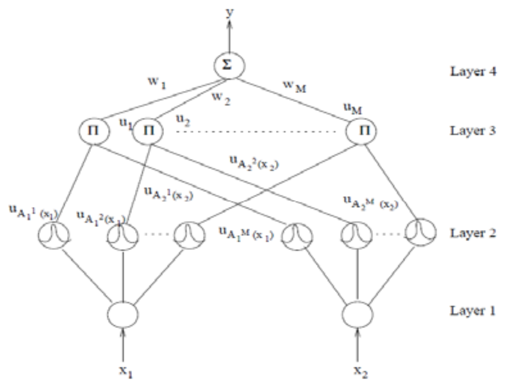
Background on Fuzzy Control

- *Inference* mechanism activates fuzzy rules based on fuzzified inputs.
  - Standard max-min Inference: the firing strength of an activated rule is given by $\mu(R_i) = \mu(A_1^{\,i}).\ \mu(A_2^{\,i})$
  - $\mu(R_1) = \mu(+\mathit{LARGE}).\ \mu(-\mathit{SMALL})$
  - $\mu(R_2) = \mu(+\mathit{MEDIUM}).\ \mu(-\mathit{ZERO})$

- *Defuzzification* process calculates controller's numeric output $\Delta m(k)$ based on conclusions of activated rules.
  - $\Delta m(k) = \sum_i (w_i . \mu(R_1)) / \sum_i \mu(R_1)$

Design of Neural Fuzzy Controller

- A neural fuzzy controller, performs the operations of a fuzzy controller using artificial neural network.
- Membership functions and rules dynamically construct and adapt themselves as the neural network grows and learns



Defuzzification $$y = \sum_{j=1}^{M} w_j \cdot u_j$$

Inference from Rule Base $$u_j = u_{A_1^j} \cdot u_{A_2^j} \cdots \cdot u_{A_n^j}$$

Fuzzification $$u_{A_i^j} = \exp\left(-\frac{(x_i - m_{ji})^2}{\sigma_{ji}^2}\right).$$

Input

Online Learning of Neural Fuzzy Controller

- Structure Learning: *dynamically determines proper input space fuzzy partitions and fuzzy logic rules, depending on the measured error and change in error in the 95th-percentile end-to-end delay.*
  - decides to add a new node in layer 2 and the associated rule node in layer 3, if all the existing rule nodes have firing strength smaller than a certain degree threshold.
  - we use a decaying degree threshold to limit the size of the neural network.
  - new node at layer 2 will have a membership function with a mean equal to the input x_i and standard deviation equal to a pre-specified or randomly generated value.

Benefit : our neural fuzzy controller performs consistently well for a wide range of input error and delay targets.

Online Learning of Neural Fuzzy Controller

- Parameter Learning: *adaptively modify the consequent part of existing fuzzy rules and the shape of membership functions to improve the controller's performance in the face of highly dynamic workload variation.*
  - Applies backpropagation learning rule to minimize the function

$$E = \frac{1}{2}(T_{ref} - T_d)^2 = \frac{1}{2}(e(k))^2$$

  - An error term is calculated as derivative of function E, with respect to a parameter of the network at each layer and propagated back.
  - We obtain parameter update rule as follows:

$$\Delta w_j = -\eta_w \frac{\delta E}{\delta w_j} = -\eta_w \frac{\delta E}{\delta y}\frac{\delta y}{\delta w_j} = \eta_w \delta^4 u_j$$

$$\Delta m_{ji} = -\eta_m \frac{\delta E}{\delta m_{ji}} = 2\eta_m \delta^2_{ji} \frac{(x_i - m_{ji})}{(\sigma_{ji})^2}.$$

$$\Delta \sigma_{ji} = -\eta_\sigma \frac{\delta E}{\delta \sigma_{ji}} = 2\eta_\sigma \delta^2_{ji} \frac{(x_i - m_{ji})^2}{(\sigma_{ji})^3}$$

Enhancements on Neural Fuzzy Controller : Compensate Switching Cost

- Server switching cost: latency between allocating servers and accurately measuring the effect of provisioning on end-to-end delay.
  - affects the parameter learning process, which depends on measurement of delay error.
  - the current measurement of delay error may actually be caused by the weights and outputs of the neural fuzzy controller that existed a few sampling intervals earlier.
- Enhancements:
  - utilize the weights and outputs of the neural fuzzy controller stored few sampling intervals earlier.
  - integrate a self-tuning component that adjusts the output to pro-actively compensate for the server switching effects.

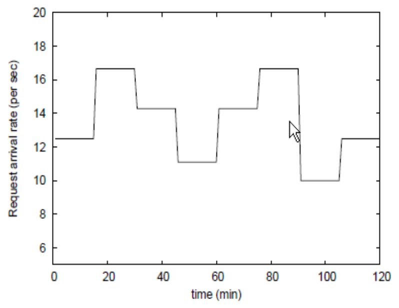
Performance Evaluation

- Extensive simulation of multi-tier cluster using synthetic session-based workload generator derived from CBMG of an online bookstore.
- We generate a synthetic G/G/1 workload using Pareto distributions of request inter-arrival time and service time.
- We choose the workload characteristics of a three-tier application reported in [Urgaonkar-ACM TAAS 2008]

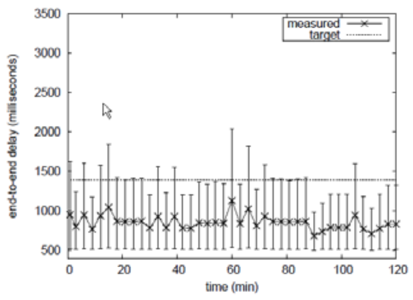
TABLE I
WORKLOAD CHARACTERISTICS.

Parameter	WebTier	AppTier	DBTier
s_i	20 ms	294 ms	254 ms
σ_i^2	848	2304	1876

Effectiveness of our provisioning approach

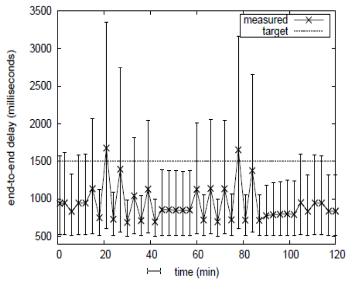


Highly dynamic workload

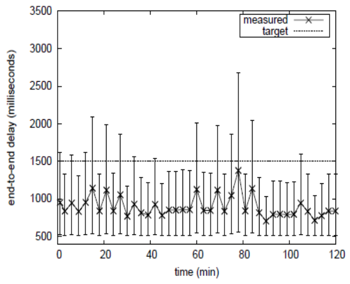


End-to-End Delay

Comparison with rule-based fuzzy controllers

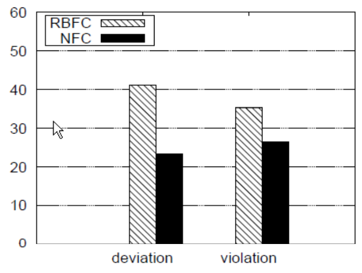


(a) with rule based fuzzy control.



(b) with neural fuzzy control.

Comparison with rule-based fuzzy controllers



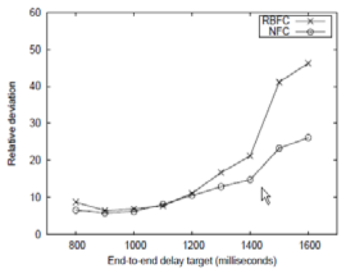
(c) performance comparison.

$$R(e) = \frac{\sqrt{\sum_{k=1}^{n} e(k)^2/n}}{T_{ref}}.$$

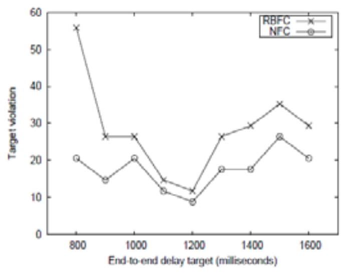
$$T(v) = \frac{\sum_{k=1}^{n} v(k)}{n}$$

Comparison with rule-based fuzzy controllers

Comparison for various delay targets

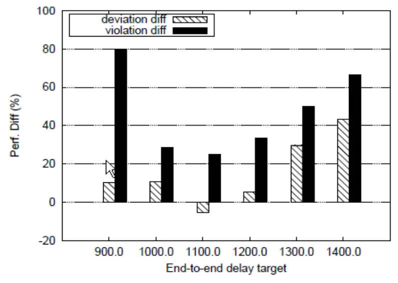


(a) relative delay deviation.



(b) temporal target violation.

Comparison with rule-based fuzzy controllers

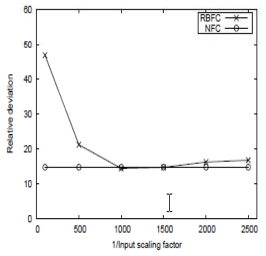


(c) performance difference.

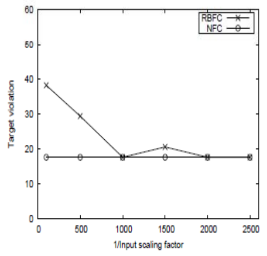
$$PD_{deviation} = \frac{R(e)_{RBFC} - R(e)_{NFC}}{R(e)_{NFC}}$$

$$PD_{violation} = \frac{T(v)_{RBFC} - T(v)_{NFC}}{T(v)_{NFC}}$$

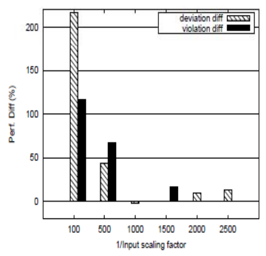
Effect of input scaling factor in rule-based fuzzy controller



(a) relative delay deviation.



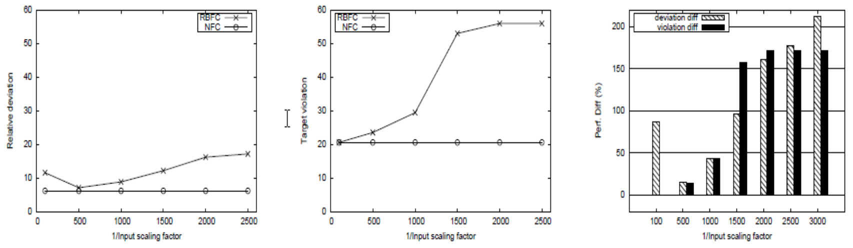
(b) temporal target violation.



(c) performance difference.

Fig. 14. Performance comparison for various input scaling factors with delay target 1400 ms.

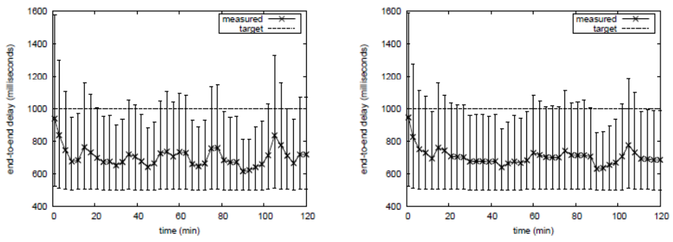
Effect of input scaling factor in rule-based fuzzy controller



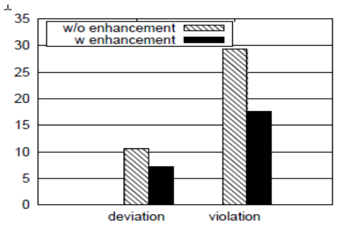
(a) relative delay deviation. (b) temporal target violation. (c) performance difference.

Fig. 15. Performance comparison for various input scaling factors with delay target 1000 ms.

Effect of server-switching delay



(a) without the enhancements. (b) with the enhancements.



(c) performance comparison.

Conclusion

- We designed a novel self-adaptive neural fuzzy control based server provisioning approach to guarantee the 95th-percentile end-to-end delay of requests flowing through a multi-tier server cluster.
- Simulation results demonstrate that the neural fuzzy controller is robust to highly dynamic workloads and changes in delay target.
- Importantly, the neural fuzzy control demonstrated its promise of being a self-adaptive approach for autonomic computing in virtualized data centers.
- Our future work will be on the implementation and evaluation of the approach in a prototype data center.

Acknowledgement

- This research was supported in part by US NSF CAREER award CNS-0844983 and research grant CNS-0720524.