Efficient Server Provisioning with Delay Guarantee on Multi-tier Clusters

Palden Lama
Xiaobo Zhou
Department of Computer Science
University of Colorado at Colorado Springs

Outline

- Motivation : QoS, multi-tier architecture, challenges/issues
- Related Work :

  "Controlling quality of service in multi-tier Web applications", [Diao-ICDCS 2006]

  "Agile dynamic provisioning of multi-tier Internet services and its applications",[Urgaonkar-ACM TAAS 2008]

  "eQoS:A Self-Tuning Fuzzy Control Approach for end-to-end QoS control on Internet Servers", [Wei-IWQoS 2005]
- Contribution : End-to-end resource optimization, average and 90th percentile end-to-end delay bound using model independent controller, reduce server switching cost, finer granularity control with non-uniform membership functions.
- Experiments and Results
- Conclusion

Motivation : QoS assurance

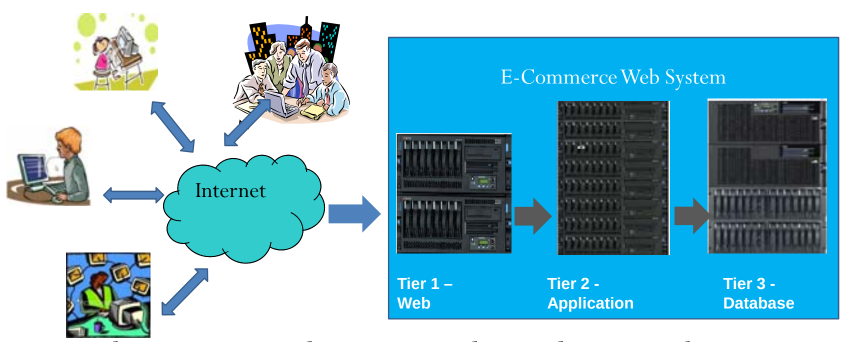
- Providing QoS in single-tier internet services and applications has been well studied in the past.
  - admission control
  - server provisioning
- Analytical modeling, machine learning and control theoretical approaches used for admission control and server provisioning.

  - "Performance guarantees for Web server end-systems: a control theoretical approach", [Abdelzaher-ITPDS 2002]

  - "Model based resource provisioning in a web service utility", [Doyle-USITS 2003]

  - "Autonomic provisioning of backend databases in dynamic content Web servers", [Chen-ICAC 2006]
- Extending mechanisms designed for single-tier to multi-tier architecture is non-trivial or even infeasible.

Motivation: Multi-Tier Architecture



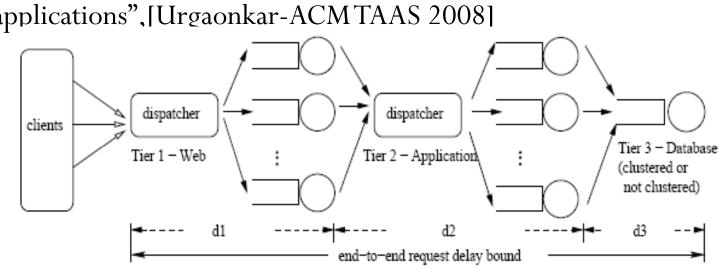
- Popular internet applications employ multi-tier architecture.
- Tiers interact to carry out its part of overall request processing.
- E-commerce web system employ 3 tiers
  - Web Tier – Http request processing
  - App Tier – Core application functionality
  - DB Tier – Storing product catalogs and user orders

Challenges/Issues

- End-to-End request delay bound is important for QoS, rather than delay at individual tier.
- Adding server to one tier does not necessarily increase effective system performance, due to inter-tier interaction, concurrency limits and cross-tier performance dependencies.
  -[Urgaonkar-ACM TAAS 2008]
- Efficiency in server provisioning for multi-tier architecture, in which each tier may be replicated and clustered for load sharing.

Related Work

- "Agile dynamic provisioning of multi-tier Internet services and its applications",[Urgaonkar-ACM TAAS 2008]



- It decomposes end-to-end delay guarantee into per-tier targets, then per-tier provisioning conducted based on queuing model to meet per-tier delay target.
- 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: How to determine those decomposition percentages?
Does not address efficiency of Server provisioning.
Application profiling is time consuming and model dependent.

We address these issues with our optimization based server provisioning approach and model-independent fuzzy controller to bound 90th percentile delay.

Related Work

- "eQoS: A Self-Tuning Fuzzy Control Approach for end-to-end QoS control on Internet Servers", [Wei-IWQoS 2005]
  - Successfully demonstrated that the approach outperforms linear PI controllers due to the model independence.
  - Work was done on processing rate allocation of a single server.
- Our Approach:
  - Fuzzy controller for dynamic server provisioning with (both average and 90th percentile) end-to-end delay guarantee in a multi-tier server architecture, together with an optimization model for resource allocation efficiency.
  - Consider use of non-uniform membership function for fine-granularity control of system performance.
  - Use a self-tuning component to reduce potential oscillations in server allocation due to switching latency.

Contribution: Optimization based server provisioning scheme

- Objective: Minimize total number of servers allocated

$$\sum_{i=1}^{n} m_i$$

- Average End-to-End delay target and resource utilization at each tier are expressed as constraints to the optimization problem

$$\sum_{i=1}^{n} d_i = U$$

$$0 < \rho_i < 1$$

Workload model: M/G/1 queuing system

- We consider session based traffic with arrival rate λ
- Average number of visits of a session to a tier denoted by v_i
- Request in different tiers demand different processing resources, r_i
- Assuming load balancer, $\rho_i = \lambda v_i r_i / m_i$
- According to Pollaczek-Khinchin formula,

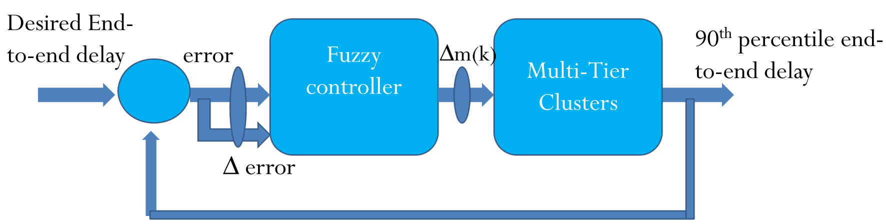
$$d_i = E[W_i] + E[X_i] = \frac{\rho_i E[X_i^2]}{2r_i(1-\rho_i)} + E[X_i]$$

  - $E[X_i]$: average of service time distribution
  - $E[X_i^2]$: 2nd moment of service time distribution
- Applying Lagrange multiplier technique for non-linear optimization

$$m_i = \lambda . v_i . r_i + \frac{\sum_{i=1}^{n} \sqrt{\lambda v_i . E[X_i^2]}}{U - \sum_{i=1}^{n} E[X_i]} \frac{\sqrt{\lambda v_i . E[X_i^2]}}{2}$$

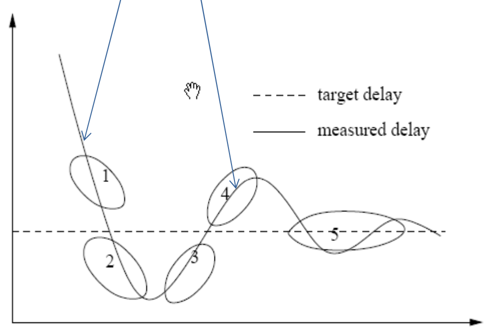
Contribution : 90th-percentile end-to-end delay using model-independent controller

- 90th percentile/95th percentile guarantee captures the user's perception of Internet service performance.
- We sample the End-to-end delay observed in the system on regular intervals and use it as a feedback to the fuzzy controller.



Rule Base

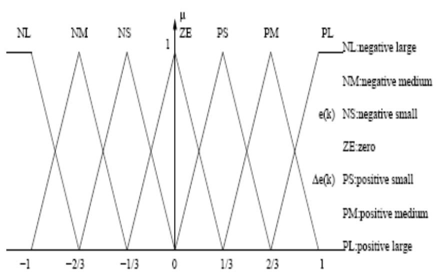
- Set of rules made to achieve desired output, depending on various combination of inputs.
  - *If error is NL and change in error is PL, then resource adjustment is ZE.*
  - *If error is NL and change in error is NL, then resource adjustment is PL.*

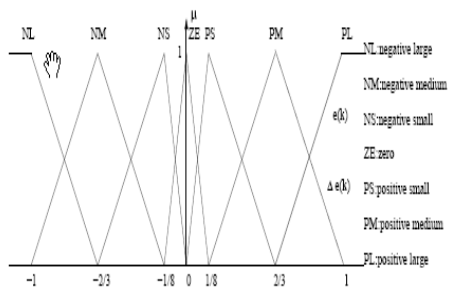


End-to-end delay transition, under the influence of fuzzy controller

Fuzzification:

- converts numeric values of inputs into fuzzy values (NL, PS, etc) using input membership functions.





- Non-uniform membership function enables finer granularity control.

Inference mechanism:

- Activates appropriate rules, based on fuzzified input.
- Standard max-min inference mechanism:

" activate set of rule(m,n) such that μ(m,n)>0
where μ(m,n) = min(μ(m), μ(n))"

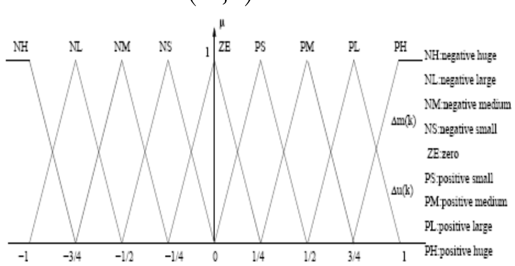
e.g if e(k) = 1/8 and Δe(k) = 1/16,
"e(k)" = PS, μ(PS) = 1
"Δe(k)" = PS, ZE with μ(PS) = 0.5, μ(ZE) = 0.5
therefore,
μ(PS,ZE) = μ(PS,PS) = 0.5
hence, rule(PS,ZE) and rule(PS,PS) are activated !

Defuzzification

- Calculates controller's numeric output (Δm(k)) based on fuzzy conclusions of activated rules.
- "center average" method to calculate controller output.

$$\Delta m(k) = \frac{\sum_{m,n} b(m,n).\mu(m,n)}{\sum_{m,n} \mu(m,n)}$$

- b(m,n): center of output membership function belonging to the result of rule(m,n)

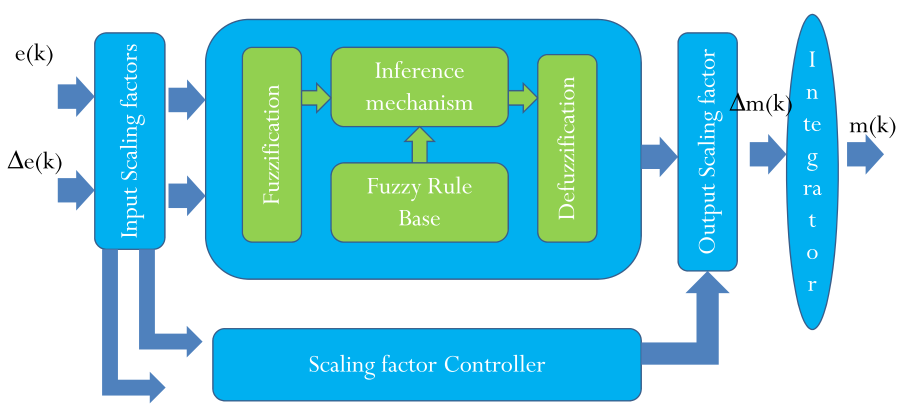


Output membership function

Contribution: Compensate Server Switching Cost

- Server switching cost: latency between allocating servers and accurately measuring the effect of provisioning on end-to-end delay.
  - e.g database replica addition goes through data migration and system stabilization phase. - "Autonomic provisioning of backend databases in dynamic content Web servers", [Chen-ICAC 2006]
- Self-tuning Ability: A controller was designed to adaptively adjust output scaling factor to compensate for server switching costs.
  - e.g. Rule Base for scaling factor controller is designed such that when error is big but has opposite sign as change in error, output scaling factor is tuned to a small value.

Fuzzy Controller



Contribution: Integration of fuzzy controller with optimization

Minimize $\sum_{i=1}^{n} m_i$

Subject to:

$$\sum_{i=1}^{n} d_i = U - \overline{U}$$

$$0 < \rho_i < 1$$

Solution:

$$m_i = \lambda . v_i . r_i + \frac{\sum_{i=1}^{n} \sqrt{\lambda v_i . E[X_i^2]}}{U - \overline{U} - \sum_{i=1}^{n} E[X_i]} \frac{\sqrt{\lambda v_i . E[X_i^2]}}{2}$$

- Increasing $\overline{U}$ implies smaller delay bound, hence more servers.
- We control $\overline{U}$ using Self-Tuning fuzzy controller to achieve 90th percentile end-to-end delay.
- Server allocation is optimized in each sampling interval of fuzzy control process.

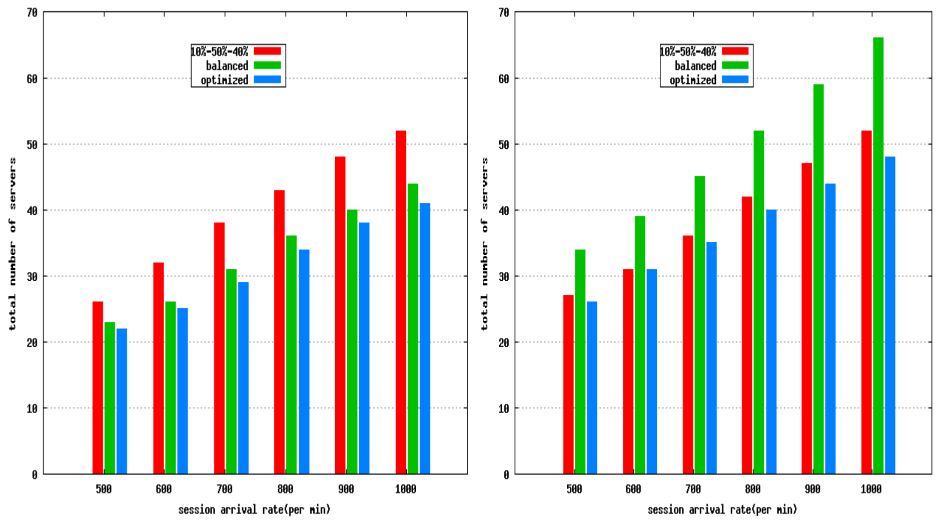
Performance Evaluation

- Extensive simulation of multi-tier cluster using synthetic session-based workload generator derived from CBMG of an online bookstore and also TPC-W benchmark.
- We adopt two sets of bounded pareto service time distribution.

Table 1: The characteristics of workload A and B.

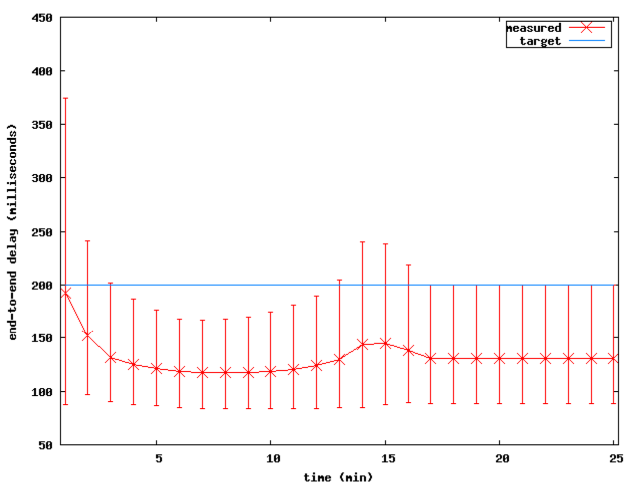
Workload A	WebTier	AppTier	DBTier
$E[X_i]$	24.163 ms	48.709 ms	34.403 ms
$E[X_i^2]$	591.864	2667.88	1191.77
Workload B	WebTier	AppTier	DBTier
$E[X_i]$	13.593 ms	67.962 ms	44.536 ms
$E[X_i^2]$	192.226	4805.66	1991.71

Optimization based approach Vs. Per tier decomposition and balanced approach



Workload A   Workload B

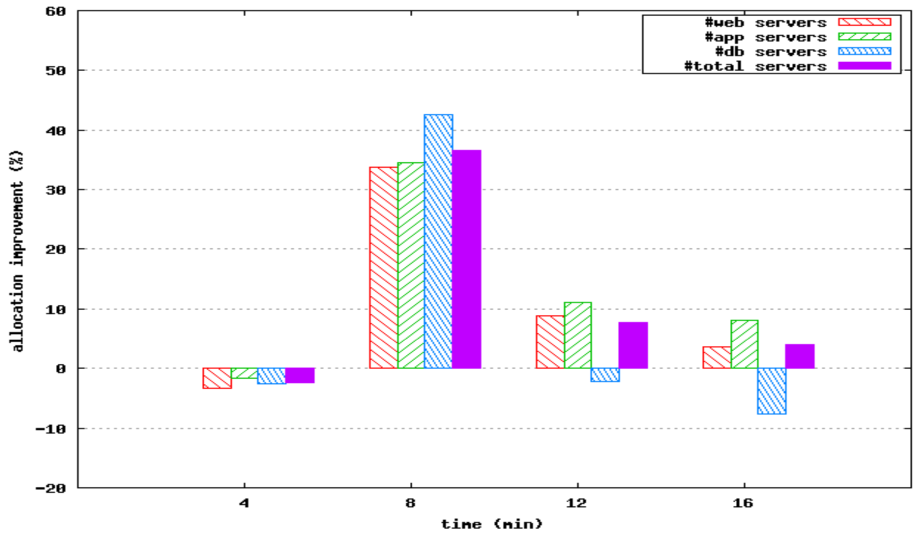
90th percentile end-to-end delay bound achieved using fuzzy controller



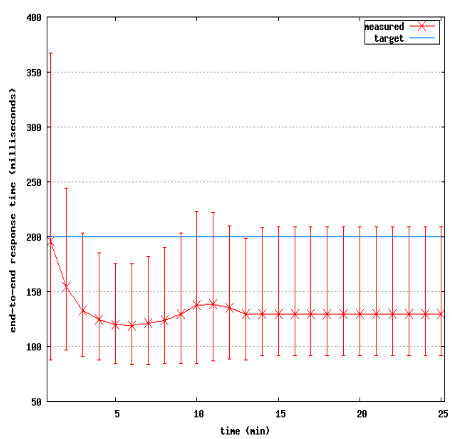
Workload A
-1600 sessions per minute

- Based on CBMG

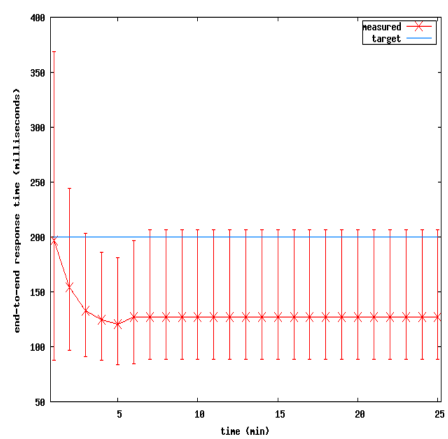
Non-uniform membership function improves server allocation



Self-tuning controller causes faster convergence

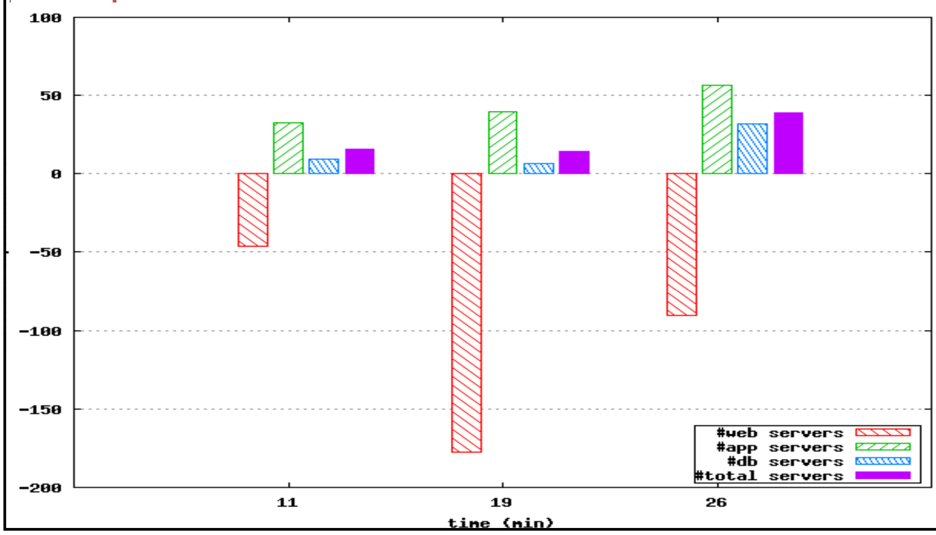


Without self-tuning

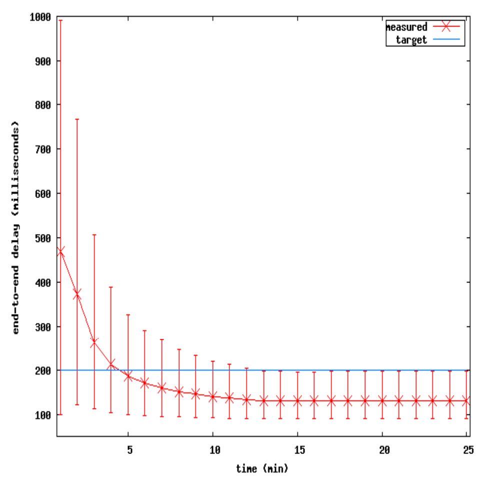


With self-tuning

Integrated approach optimizes server allocation at each step of fuzzy control process



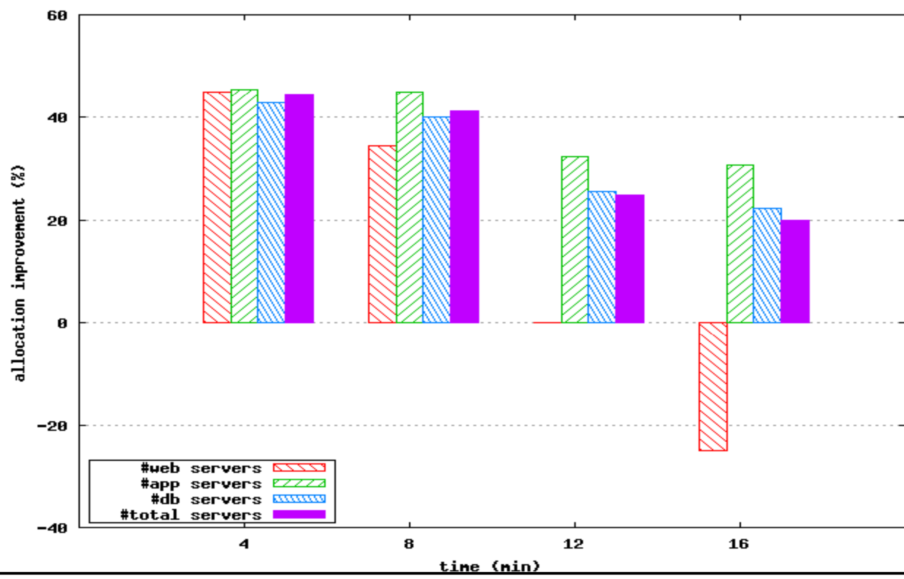
Performance with TPC-W workload



- 500 simulated users
- Ordering traffic mix
- TPC-W 14 state customer behavior model

- unpredictable workload model in terms of average number of visits that a session will make to a state
- exponentially distributed USMD

Impact of integrated approach on modified TPC-W workload



Conclusion

- We proposed an efficient server provisioning approach based on end-to-end resource optimization model.
- We designed a model-independent self-tuning controller to provide 90th percentile end-to-end delay guarantee.
- Integration of optimization model and model-independent fuzzy controller provides superior performance in resource allocation efficiency and end-to-end delay assurance.

Thanks

- Questions ?