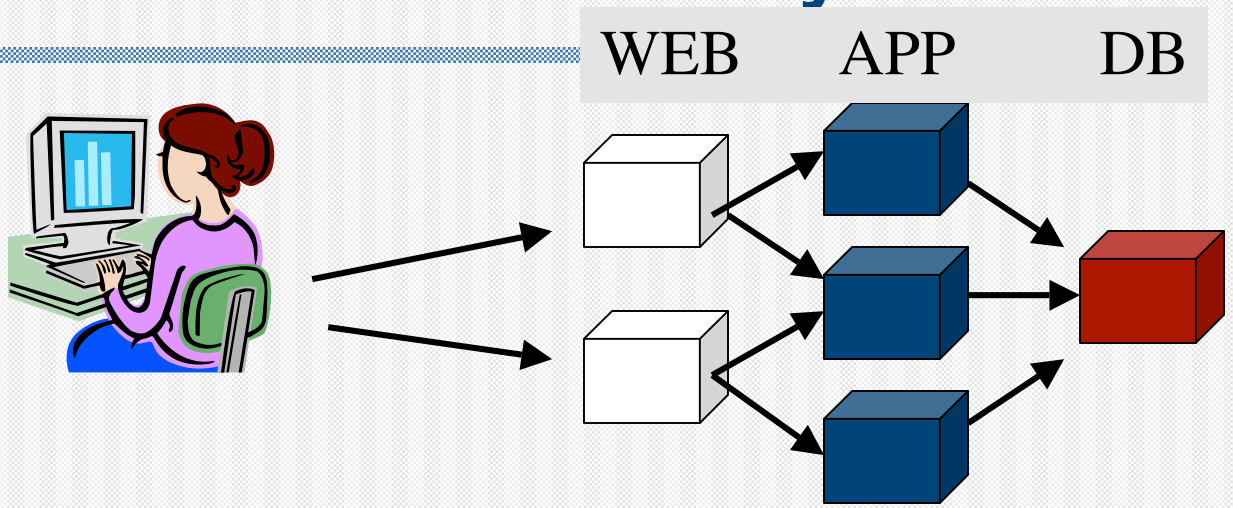
Automatic Classification of Requests to a 3-tier system using SLT

George Porter

Winter retreat

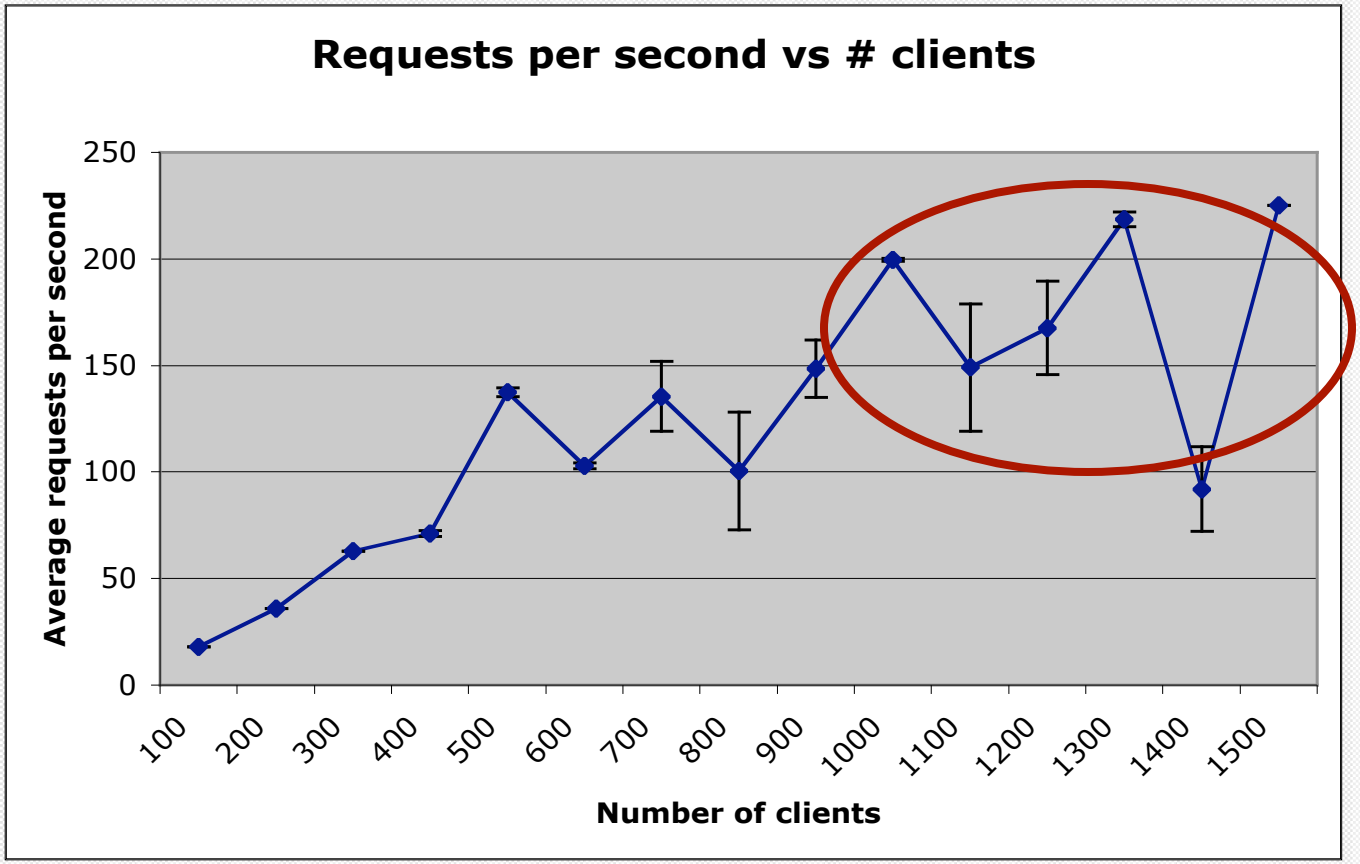
2005

Motivation: What is a 3 tier system?



- Composable building blocks to build web services
  - Web containers, various app/ejb containers, persistent state via automatically managed DB pools
- Problem: Open control loop/requests driven by users
  - Unusual requests, flash traffic, increased workload can overload components of the web service
  - Hard to provision; hard to make performance guarantees; **this leads to seemingly broken behavior to the end user**

Increasing load leads to perceived broken behavior

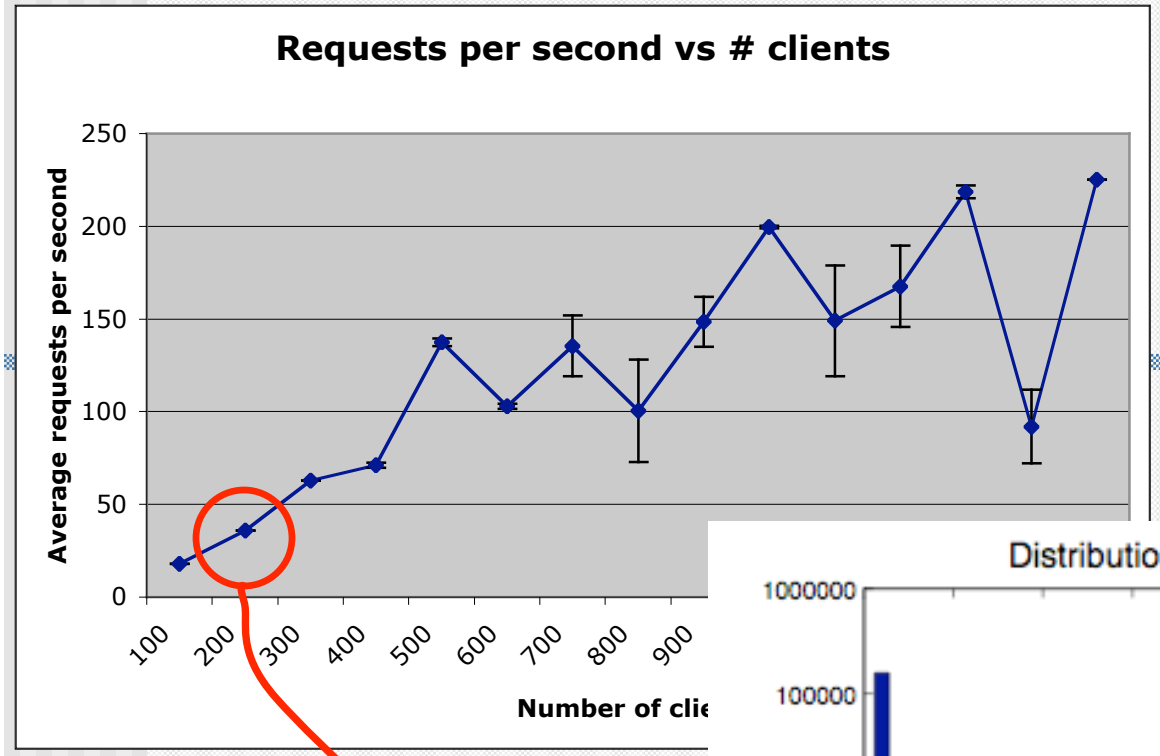


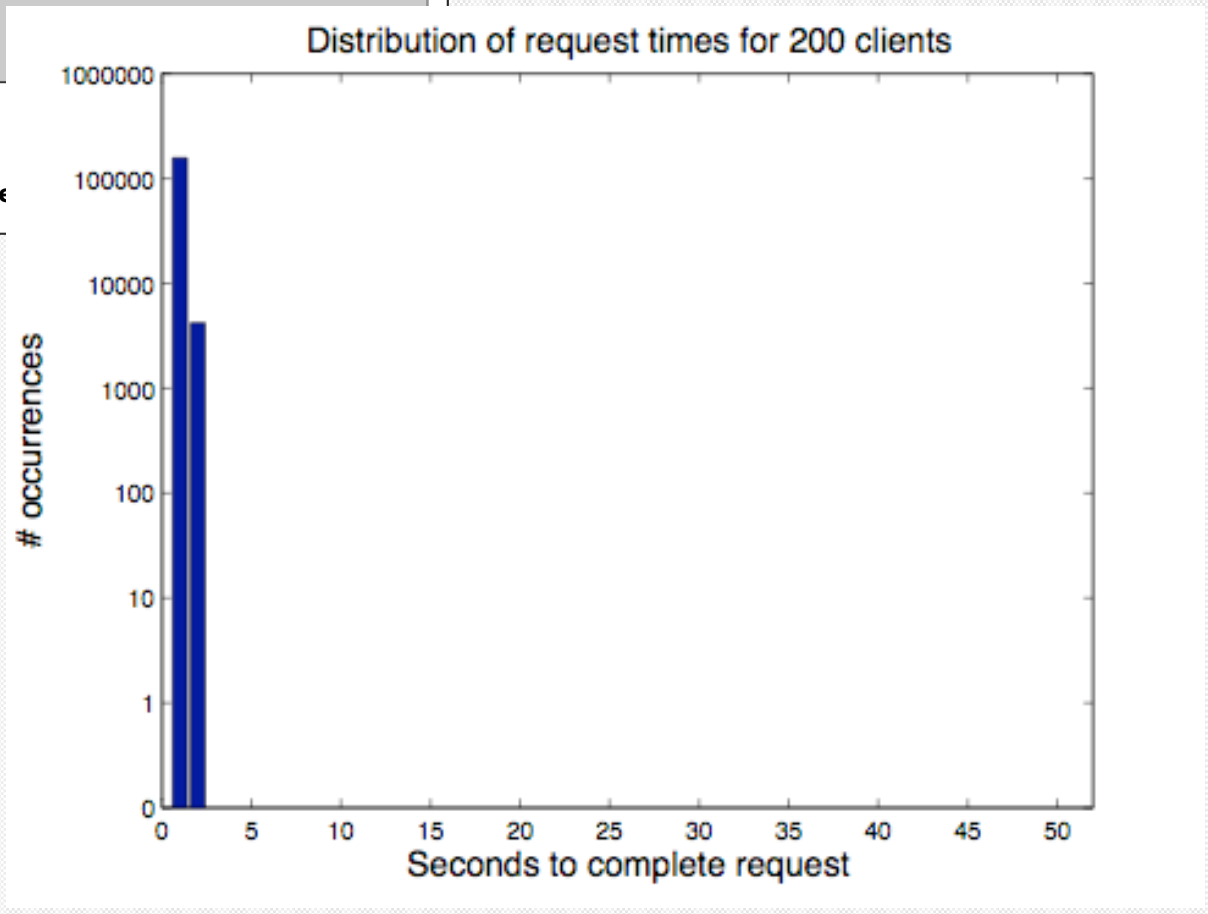
System in overload state…

…this leads to the following problem:

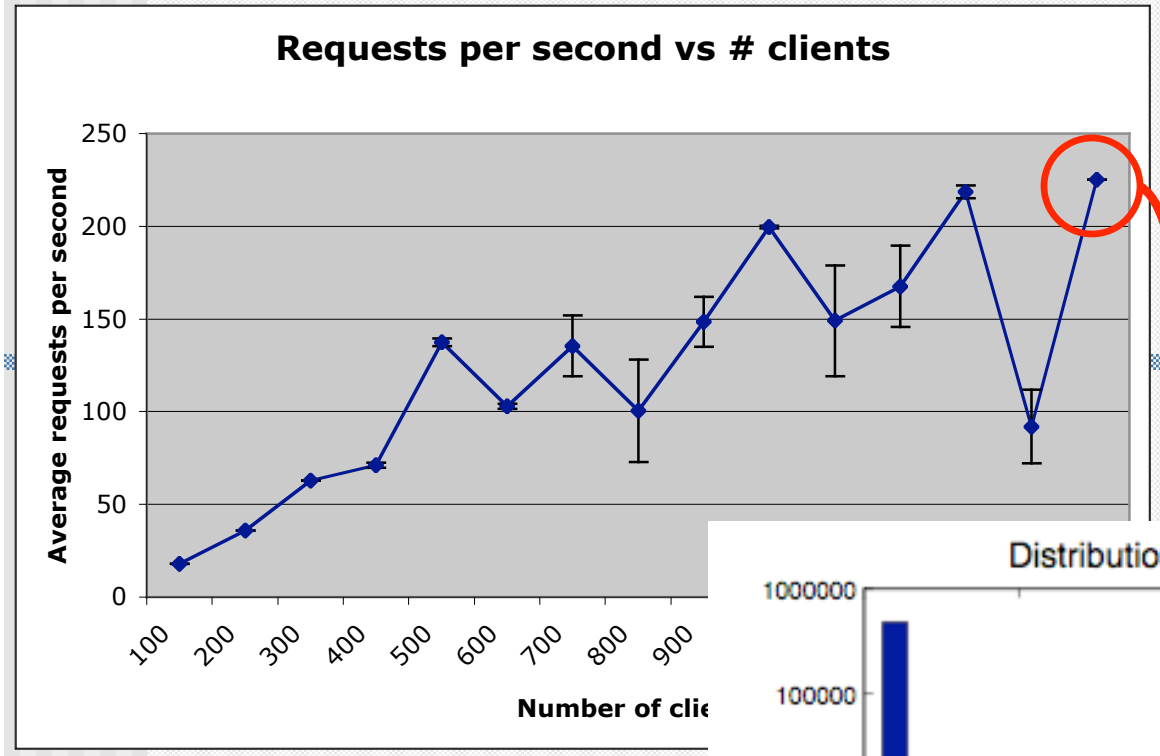
Results taken from RUBiS running on Emulab

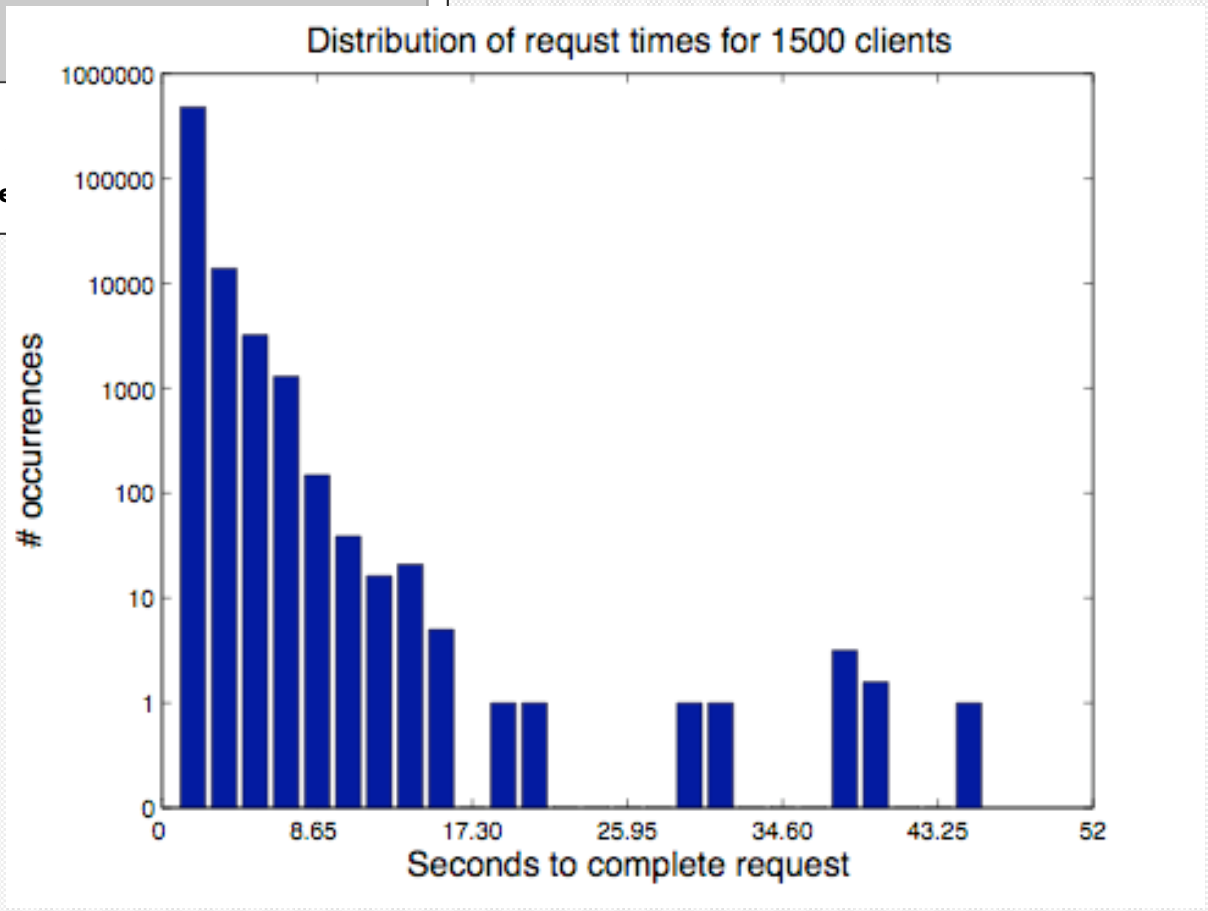
Behavior at low load





Behavior at high load

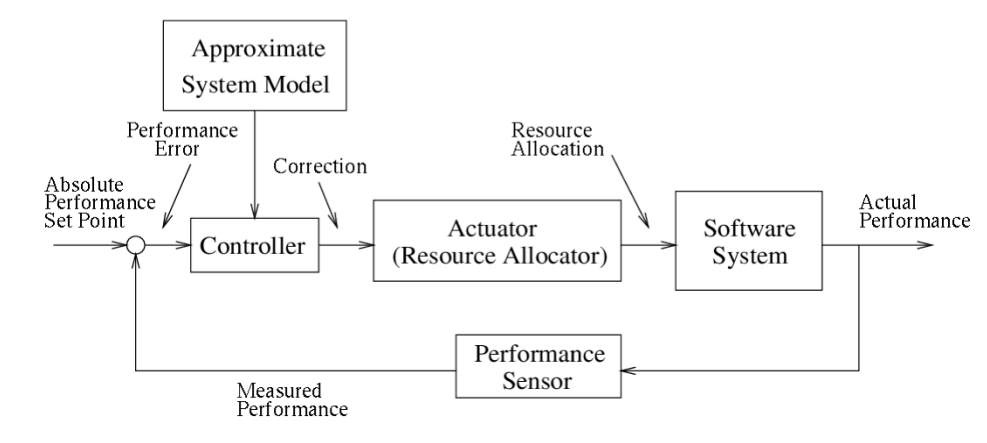




For users, the system seems defective in many cases

Attempts at apply CT to 3 tier systems

- Most relevant: ControlWare
  - Zhang, Lu, et al. - Univ of Virginia
  - Middleware system for mapping QoS goals into CT loops by controlling allocation of threads to sockets, cache space to buffers, etc. (opaque requests)
- But, all requests treated as the same -- homogeneous view
  - *This punishes "light" requests just as much as complex, CPU-intensive tasks*

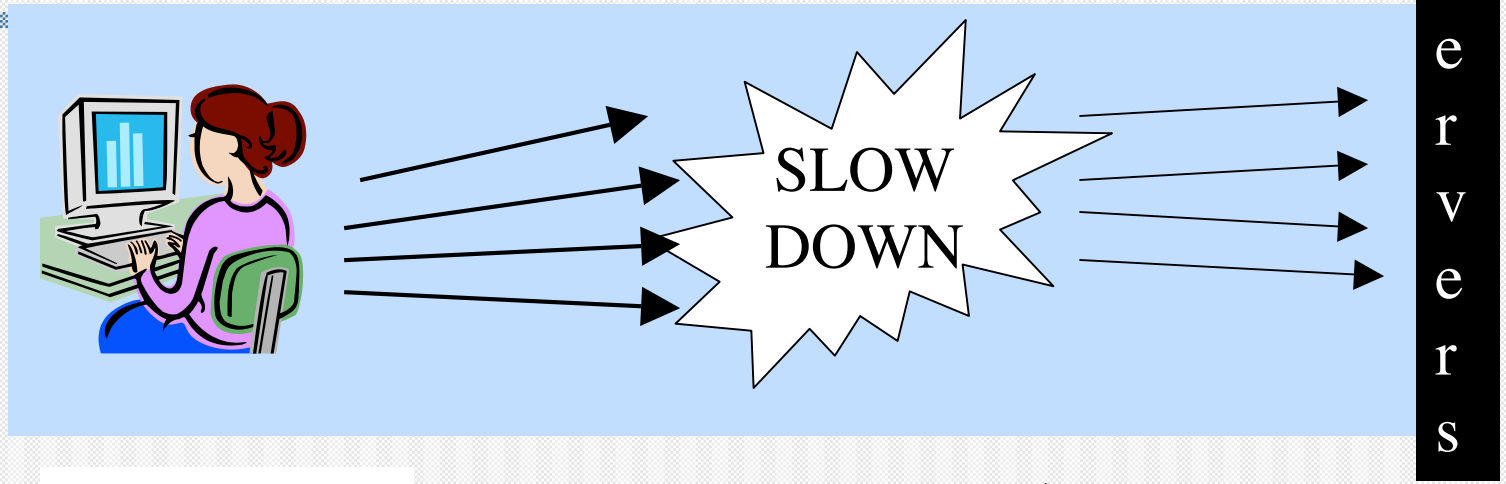


IDEA: Use SLT to classify requests based on their effect

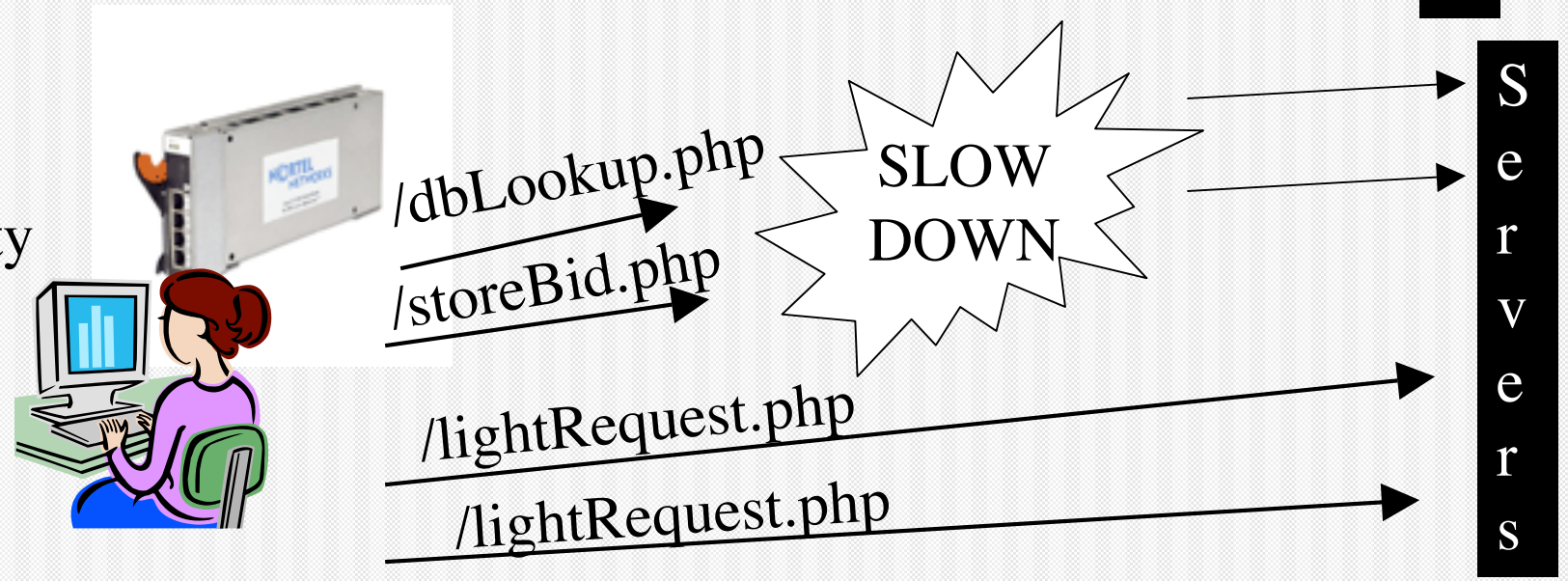
- Analysis: group requests into
  - Those that affect the bottleneck
  - Those that don't
- …using the technique of ***linear regression***
  - Classify requests based on correlations to DB CPU utilization (the bottleneck in my system)
- Find candidate list of requests that are correlated with bottleneck
- (in progress) separate these requests into a separate, bandwidth-shaped path
- Assumption: reduce avg service time by delaying "elephant flows"

Role for iBoxes

No Network Visibility

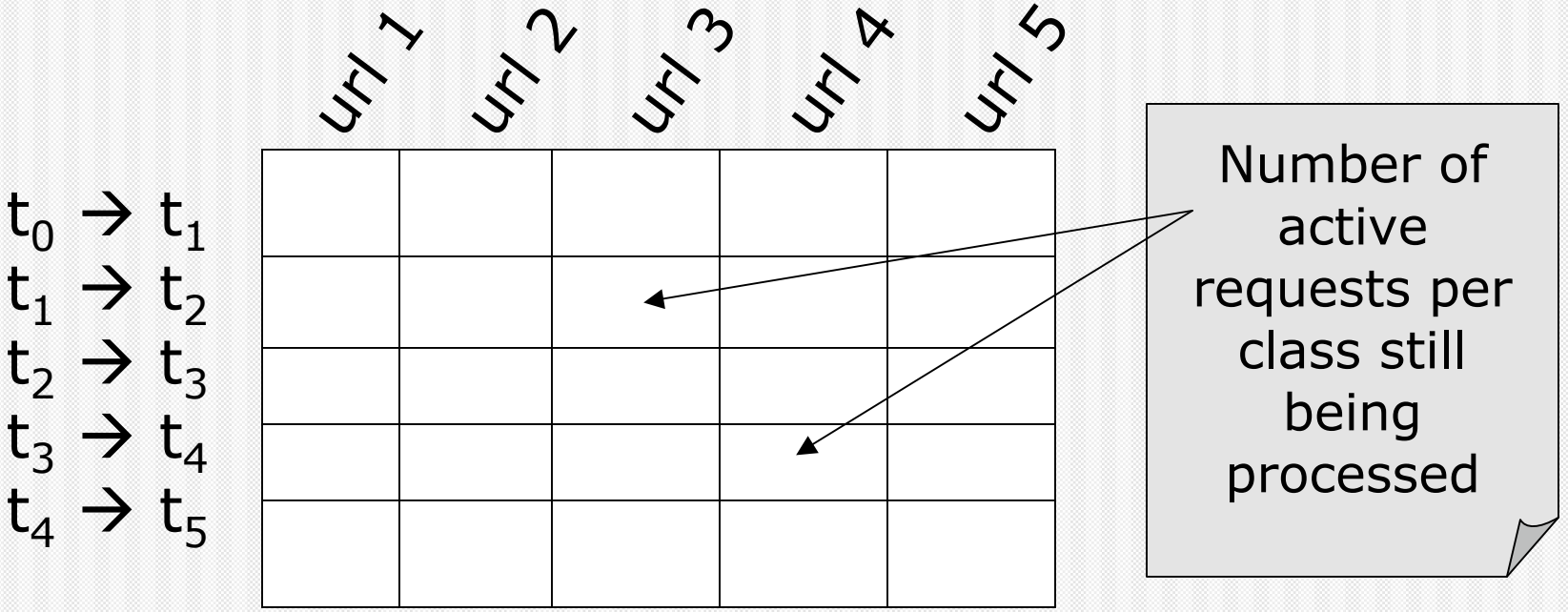


HTTP Header Visibility



Observe: web requests

- From Web server's Apache logs:

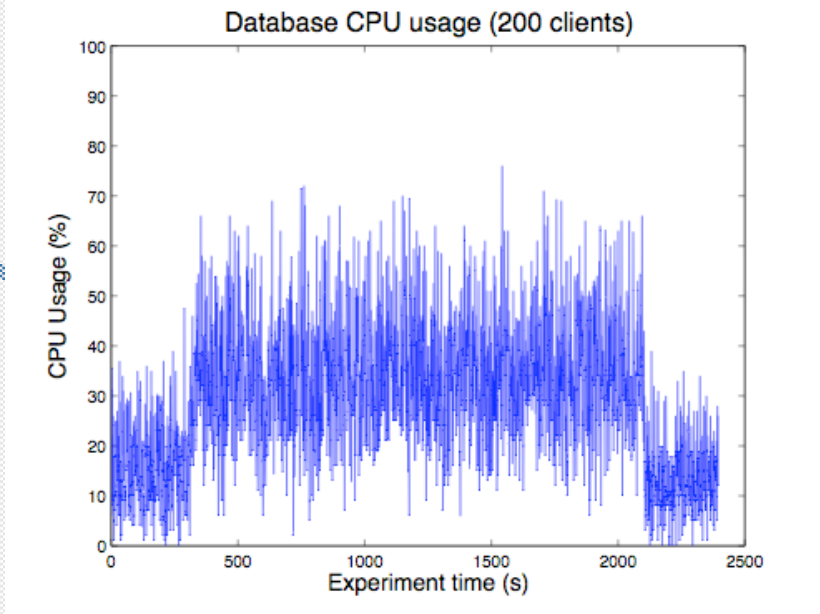


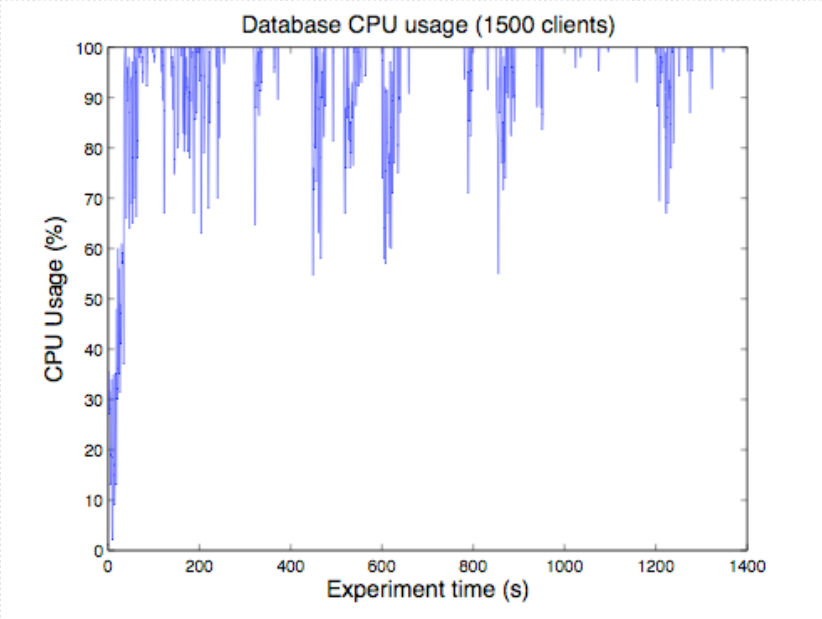

```
10.1.1.2 20296 + 1377 1102213360 0 /PHP/RUBiS_logo.jpg
10.1.1.2 1393 + 1375 1102213360 0 /PHP/SearchItemsByCategory.php
10.1.1.2 3736 + 1390 1102213360 0 /PHP/BrowseCategories.php
```

Request duration

Observe: servers

- Utilized *sysstat*
- Collected for web, db:
  - CPU idle, system, user, busy
  - Network traffic between tiers
  - Context switches
  - Disk I/O operations
- This work focuses on DB CPU, which in my deployment was the bottleneck



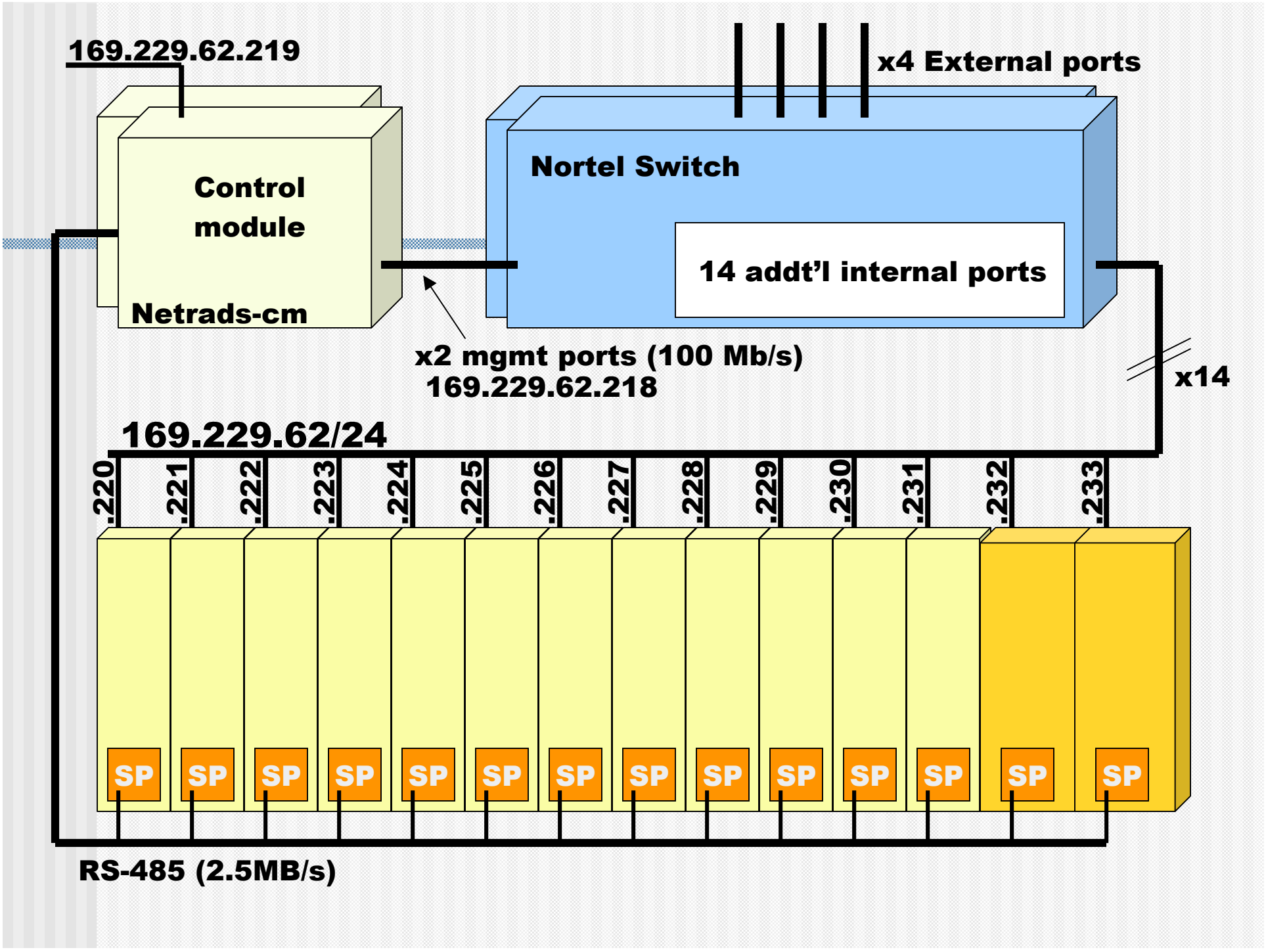


Analyze

- Linear regression
  - “Black box approach”
  - No modification of O/S or apps
  - Minimal interference (capture Apache logs, use the *sysstat* system utility)
  - No need to tag requests, match requests with effect, or match observations at the web server with observations at the DB server
- Additive, linear model
  - At high level, load on CPU is the sum of work given to it
  - Smaller order effects like CPU scheduling, caching, paging, disk arm activity, etc., important, but not in my model
- For the model, it is only important that these effects are not correlated to the class of request

New “Act” Opportunity: iBoxes

- Deep packet inspection of HTTP headers per flow
  - *Nortel 2-7 switch*
- Per-flow/per-vlan bandwidth shaping
  - Packeteer PacketShaper
- Currently this work only classifies requests, although integration with the above PNEs is in progress in our BladeCenter


169.229.62.219
x4 External ports
Control
module
Netrads-cm
Nortel Switch
14 addt'l internal ports
x2 mgmt ports (100 Mb/s)
169.229.62.218
x14
169.229.62/24
.220
.221
.222
.223
.224
.225
.226
.227
.228
.229
.230
.231
.232
.233
SP
RS-485 (2.5MB/s)

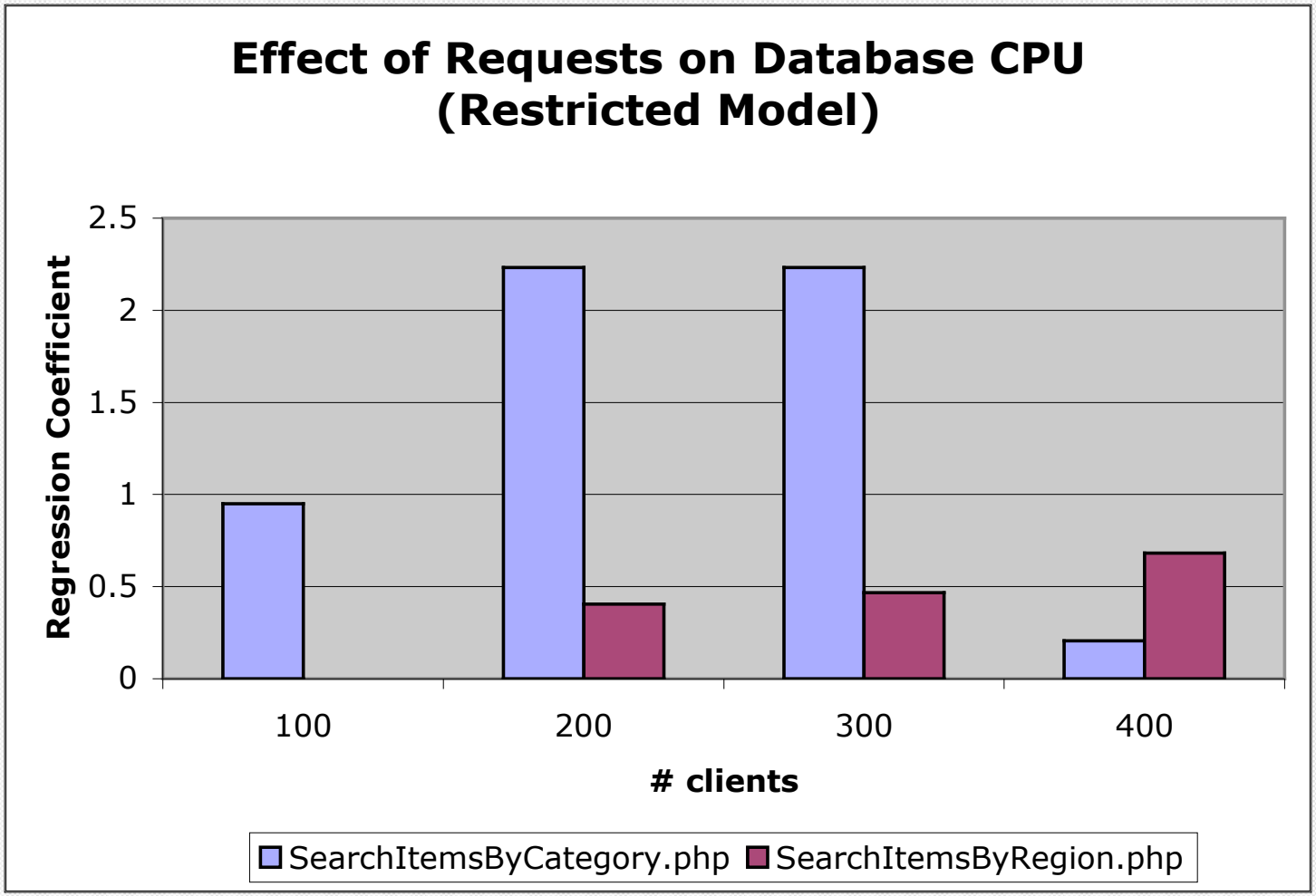
Summary: Observe/Analyze/Act Framework

- Observe
  - Apache web logs / Systems measurements
  - HTTP headers in requests
- Analyze
  - Offline (periodic) linear regression
  - Output: subset of URLs positively correlated to bottleneck
- Act (in-progress)
  - Use Nortel switch to segregate correlated requests into their own VLAN
  - Use Packeteer box to throttle that VLAN

Experimental setup

- Emulab testbed
  - Reconfigurable interconnect, linux-based platform, Utah
- RUBiS (Rice Univ. Bidding System)
  - eBay like workload, transition matrix driven
  - Default matrix, 7 sec
- 5 clients
- Apache + PHP app
- MySQL DB server

Results of regression



Only statistically significant coefficients shown

Unexpected Results

- Even the simple RUBiS system has numerous request types
  - I assumed *a priori* that several of the requests would be correlated, but weren't
- Real systems have many, many more request pathways
  - Given a list of 40 URLs, which are correlated to load?
- Experimentally we found a more narrow set of candidate URLs than expected

Read-write workload (transition_7.txt)

```
/PHP/RUBiS_logo.jpg
/PHP/SearchItemsByCategory.php
/PHP/index.html
/PHP/BrowseCategories.php
/PHP/browse.html
/PHP/SearchItemsByRegion.php
/PHP/BrowseRegions.php
/PHP/about_me.html
/PHP/AboutMe.php
/PHP/bid_now.jpg
/PHP/RegisterUser.php
/PHP/register.html

(70,851 requests total)
```

```
/PHP/ViewItem.php
/PHP/sell.html
/PHP/PutBidAuth.php
/PHP/PutBid.php
/PHP/ViewUserInfo.php
/PHP/BuyNow.php
/PHP/BuyNowAuth.php
/PHP/ViewBidHistory.php
/PHP/PutComment.php
/PHP/SellItemForm.php
/PHP/RegisterItem.php
/PHP/StoreComment.php
/PHP/StoreBid.php
```

Review

- SLT was able to classify requests to a web service based on their effect on the system
- Linear regression techniques:
  - Were able to discover statistically significant, positively correlated relationships between search URLs and load on the DB server
  - Avoid the need to modify the system
  - Don't require matching observations at the web server with observations at the db
- Better QoS by throttling back requests
  - Correlations discovered by SLT narrow down the list of URLs to throttle
  - This throttling places the most delay on those users causing the most load, while not throttling other users
  - (work in progress)
- Leads to perceived higher reliability

Questions?

- Thanks to Alice Zheng and Gert Lanckriet
- Thanks to the Emulab group

Backup Slides

Control theory implications

- We have candidate list of requests to pass through Packeteer PNE for throttling
- Our choice is inherently monotonic
  - Throttling requests of any type will reduce load on the system
  - First reduce URLs with pos correlation, then, if necessary, other URLs
- Several options for throttling choice:
  - URL with highest correlation
  - Dial for those URLs with pos. correlations
  - Implemented with SLB groups on a load balancer

Analyse: The model

- Model:
  - $Y = \beta x + \varepsilon$
  - Y is MxN
  - X is NxC
- Result:
  - ßHat is then MxC
- OLS:
  - Yhat = X*ßHat
  - e = Yhat - Y
  - RSS = $\Sigma_i e_i$
  - SE = sqrt(RSS)

- Variables
  - N: number of time epochs (output variable measurements)
  - M: # output variables
  - C: # of classes (# urls)

Stepwise regression

- Find covariate with highest correlation to Y, and add if p-value < 0.05
  - Continue adding variables to the model until all remaining covariates have p-value >= 0.05
- The result is a linear equation containing only stat. significant terms

Stepwise regression example

```
Initial columns included:  none
Step 1, added column 3, p=0
Step 2, added column 8, p=0
Step 3, added column 12, p=3.52833e-05
Step 4, added column 4, p=0.00339175
Step 5, added column 5, p=0.0122998
Final columns included:  3 4 5 8 12

ans =

    'Coeff'          'Std.Err.'       'Status'    'P'
    [1.2259e+10]     [7.9807e+12]     'Out'       [    0.9988]
    [   -0.3189]     [    0.1721]     'Out'       [    0.0640]
    [    0.3076]     [    0.0853]     'In'        [3.1994e-04]
    [   -0.5781]     [    0.1608]     'In'        [3.3473e-04]
    [   -0.3890]     [    0.1552]     'In'        [    0.0123]
    [    0.3193]     [    0.3989]     'Out'       [    0.4236]
    [   -0.2388]     [    0.2567]     'Out'       [    0.3525]
    [    1.8627]     [    0.1295]     'In'        [         0]
    [   -0.3436]     [    0.3851]     'Out'       [    0.3723]
    [   -0.3088]     [    0.3018]     'Out'       [    0.3064]
    [   -0.3817]     [    0.2984]     'Out'       [    0.2011]
    [    0.5235]     [    0.1893]     'In'        [    0.0057]
    [   -0.8143]     [    0.4918]     'Out'       [    0.0980]
```

Results (con't)

- Experiments with 100 to 1500 clients
  - But at 500 the DB server became the bottleneck
- Strong positive correlations with searching urls

References

- [1] R. Zhang, C. Lu, T. Abdelzaher, J. Stankovic. ControlWare: A Middleware Architecture for Feedback Control of Software Performance. In Proceedings of the 2002 International Conference on Distributed Computing Systems, Vienna, Austria, July 2002.
- [2] A. Goel, D. Steere, C. Pu, and J. Walpole. Swift: A feedback control and dynamic reconfiguration toolkit. Technical Report CSE-98-009, Oregon Graduate Institute, Portland, OR, June 1998.
- [3] E. Cecchet, A. Chanda, S. Elnikety, J. Marguerite and W. Zwaenepoel. Performance Comparison of Middlware Architectures for Generating Dynamic Web Content. 4th ACM/IFIP/USENIX International Middleware Conference. Rio de Janeiro, Brazil, June 16-20, 2003.
- [4] E. Cecchet, J. Marguerite and W. Zwaenepoel. Performance and scalability of EJB applications. 17th ACM Conference on Object-oriented Programming, Systems, Languages and Applications (OOpsla 2002), Seattle, WA. Nov 4-8, 2002.
- [5] C. Amza, E. Cecchet, A. Chanda, A. Cox, S. Elnikety, R. Gil, J. Marguerite. Specification and Implementation of Dynamic Web Site Benchmarks IEEE 5th Annual Workshop on Workload Characterization (WWC-5). Austin, TX. Nov 2002.

Default_7.txt Workload Categories

```
59512 /PHP/RUBiS_logo.jpg
13060 /PHP/SearchItemsByCategory.php
12934 /PHP/index.html
10233 /PHP/BrowseCategories.php
 9761 /PHP/browse.html
 5469 /PHP/SearchItemsByRegion.php
 2857 /PHP/BrowseRegions.php
 2102 /PHP/about_me.html
 2057 /PHP/AboutMe.php
 1209 /PHP/register.html
 1207 /PHP/RegisterUser.php
  675 /PHP/sell.html
    3 /PHP/ViewUserInfo.php
```

Request distribution for 1400 clients