Effective “Small Site” Web Loadbalancing through Statistical Monitoring

George Porter, Randy H. Katz

Univ. of California Berkeley

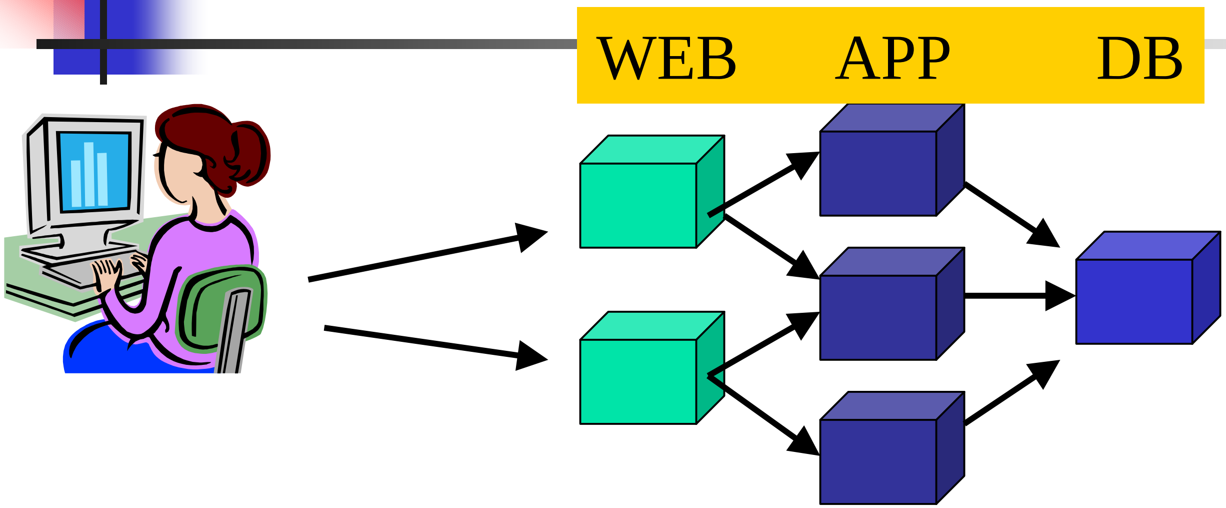
May 19, 2005

Selfman 2005 - Nice, France

Outline

- Motivation: Need for dynamic admission control for web services
  - Targeting “unmanaged” high variance sites such as open-source blogs
- My focus today is on the deployment problem via
  1. Black-box component monitoring
  2. Ultra-lightweight request effect discovery
  3. Visualizing correlations
  4. Network-level, selective request throttling
- Initial investigation with live 3-tier system
- Conclusion

Web services under excessive load



Examples:

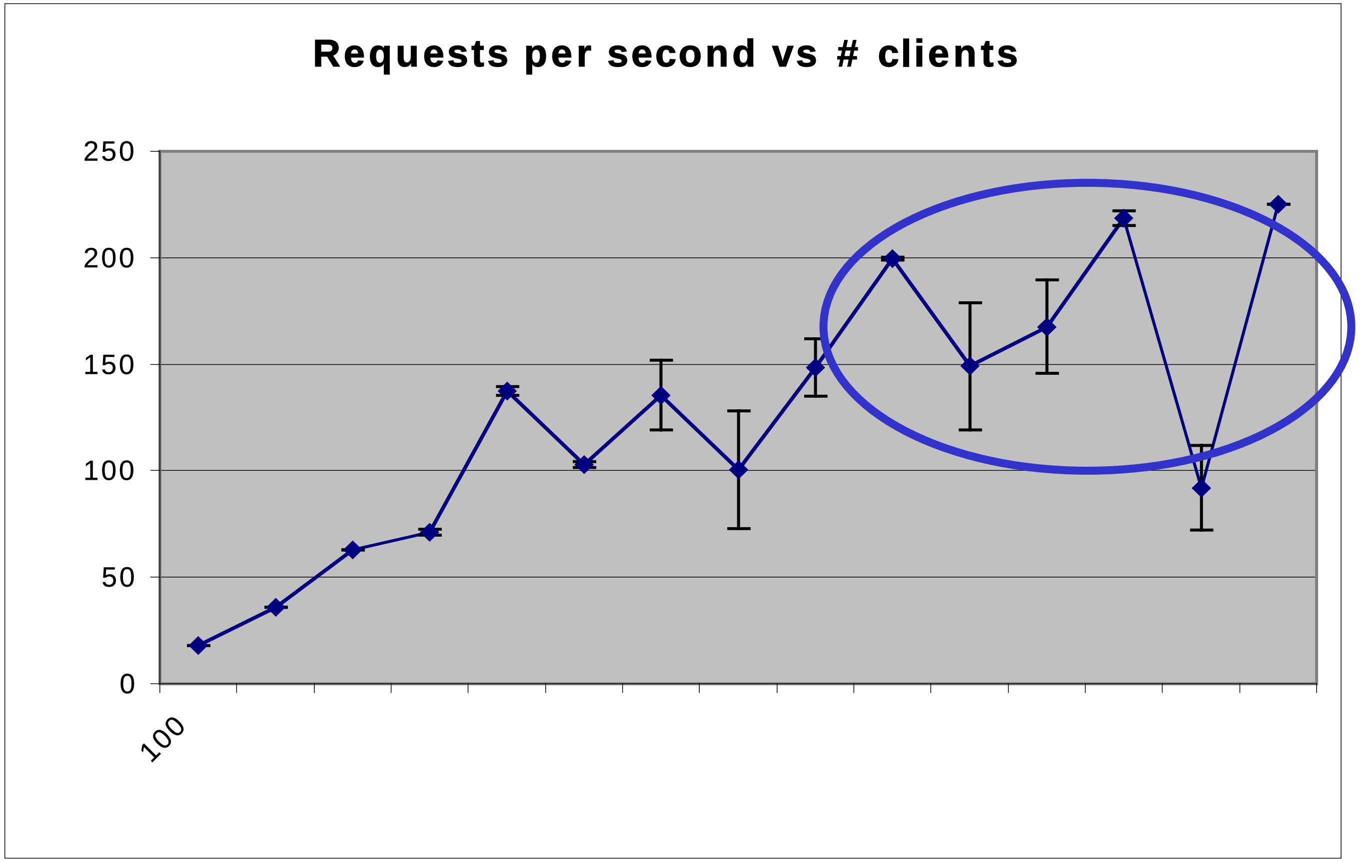
- Amazon, eBay.com
- Dailykos, alternet.org

- Composable building blocks to build web sites
  - Web containers, various app/ejb containers, persistent state via automatically managed DB pools
- Problem: Open control loop/requests driven by users
  - 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**

Target Environment

- High variability of workload
  - 300k/day visitors
  - Sometimes > 1M users/day
- Limited resources
  - Cannot turn on spare servers/blades
- Not business critical
  - But important that the service is available during flash traffic events (elections, news events, etc).

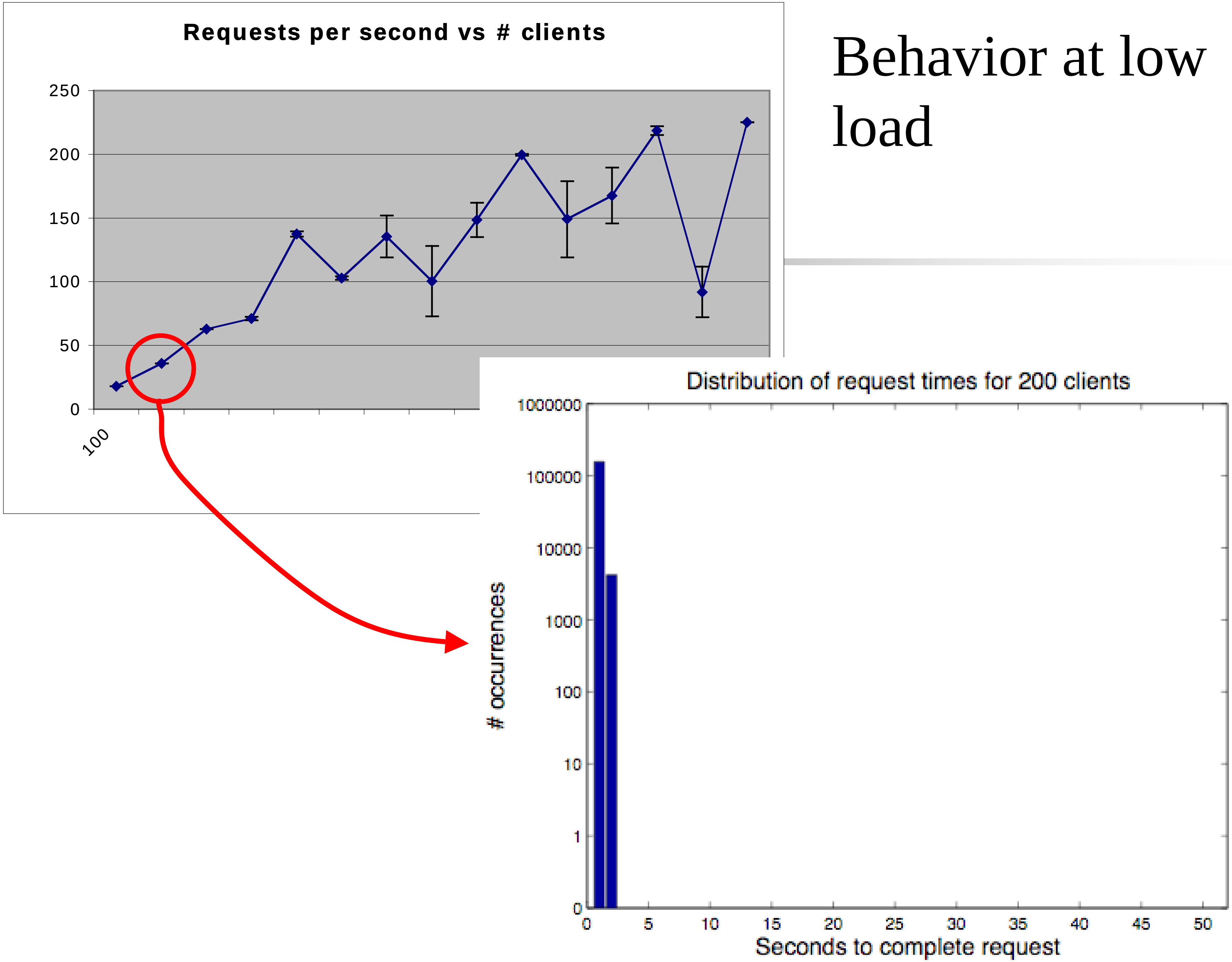
Increasing load leads to undesirable behavior



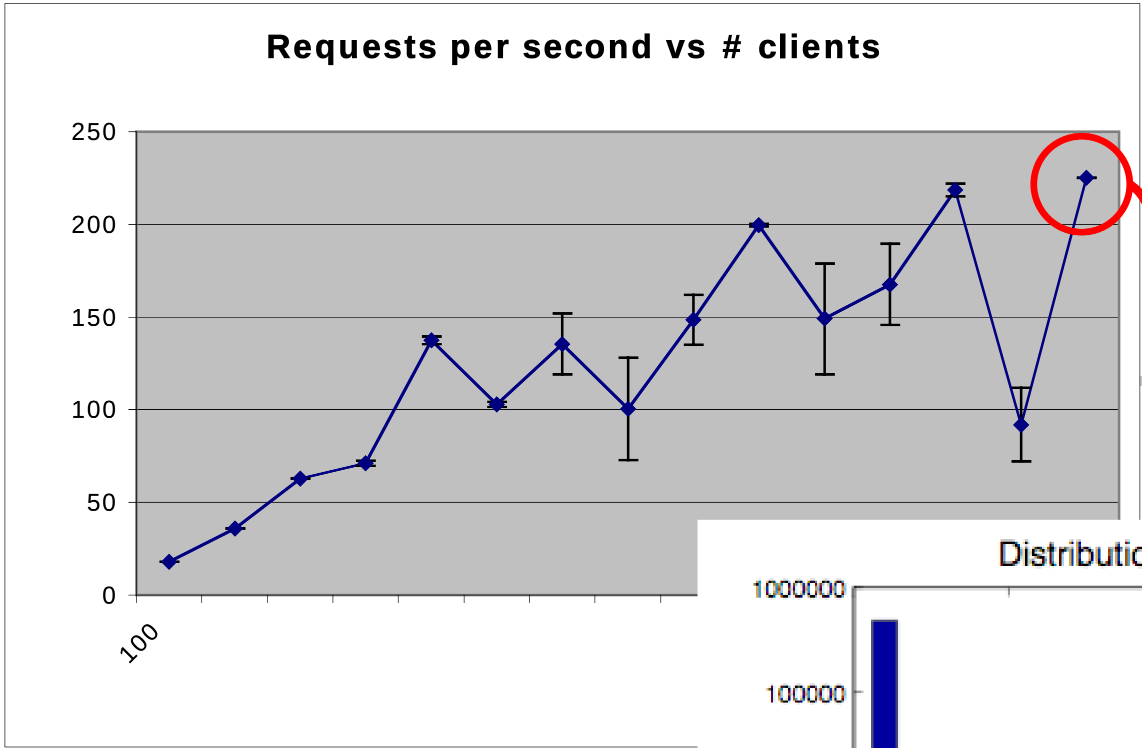
System in overload state…

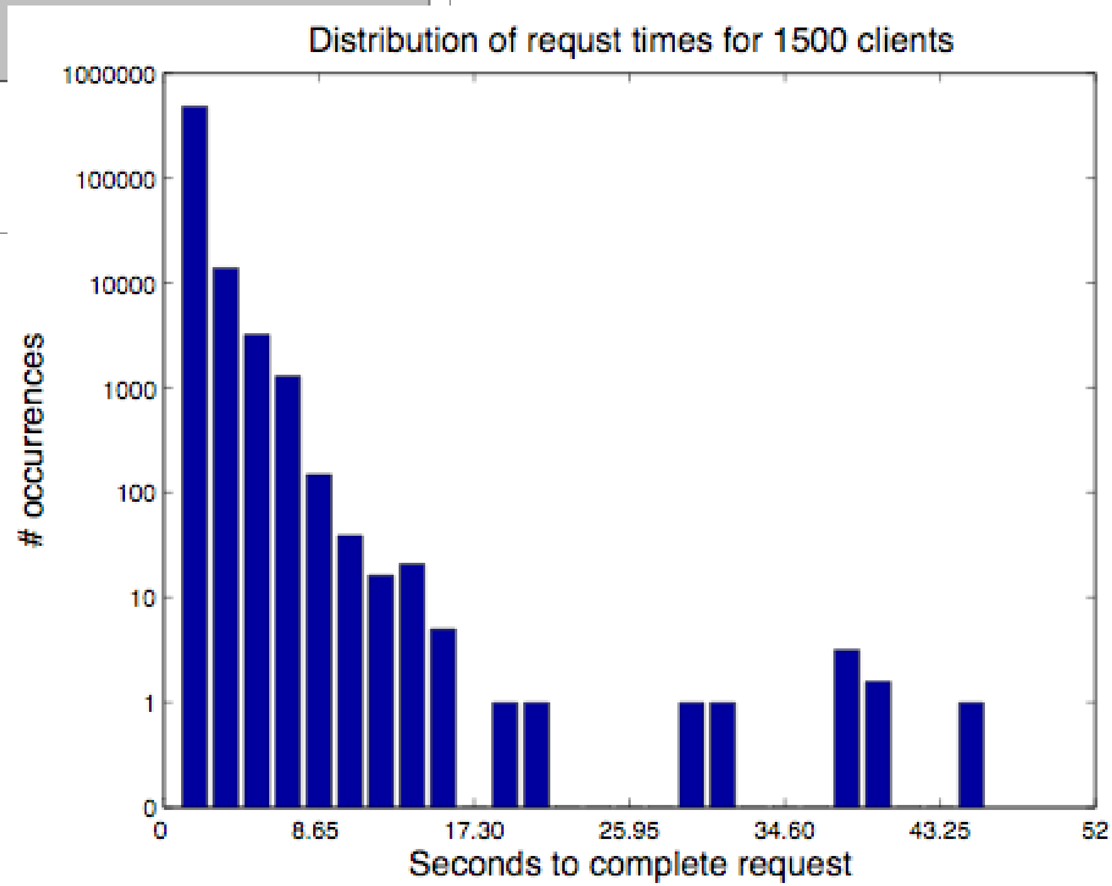
…this leads to the following problem:

Behavior at low load



Behavior at high load

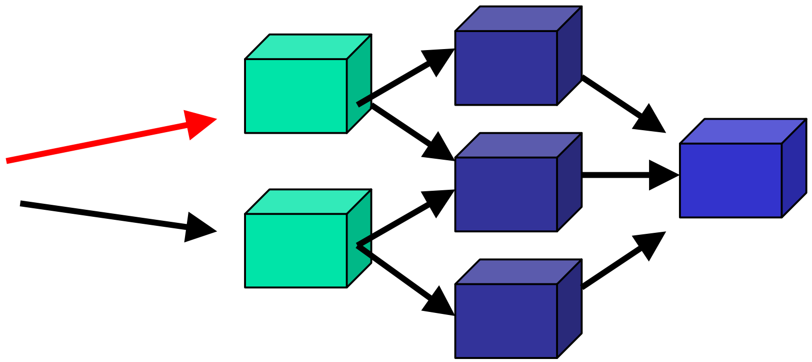




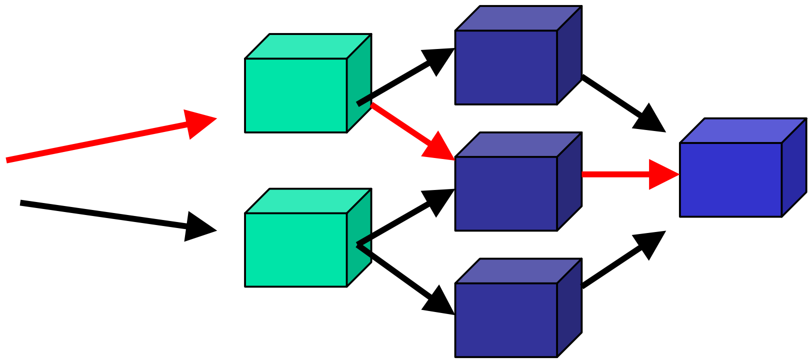
For users, the system seems defective in many cases

Observation: Ant and Elephant flows

- Ant flows
  - Invokes few components
  - Invokes inexpensive processing
  - Often more common



- Elephant flows
  - Touches several layers
  - Heavyweight processing / searching / DB joins



Objective

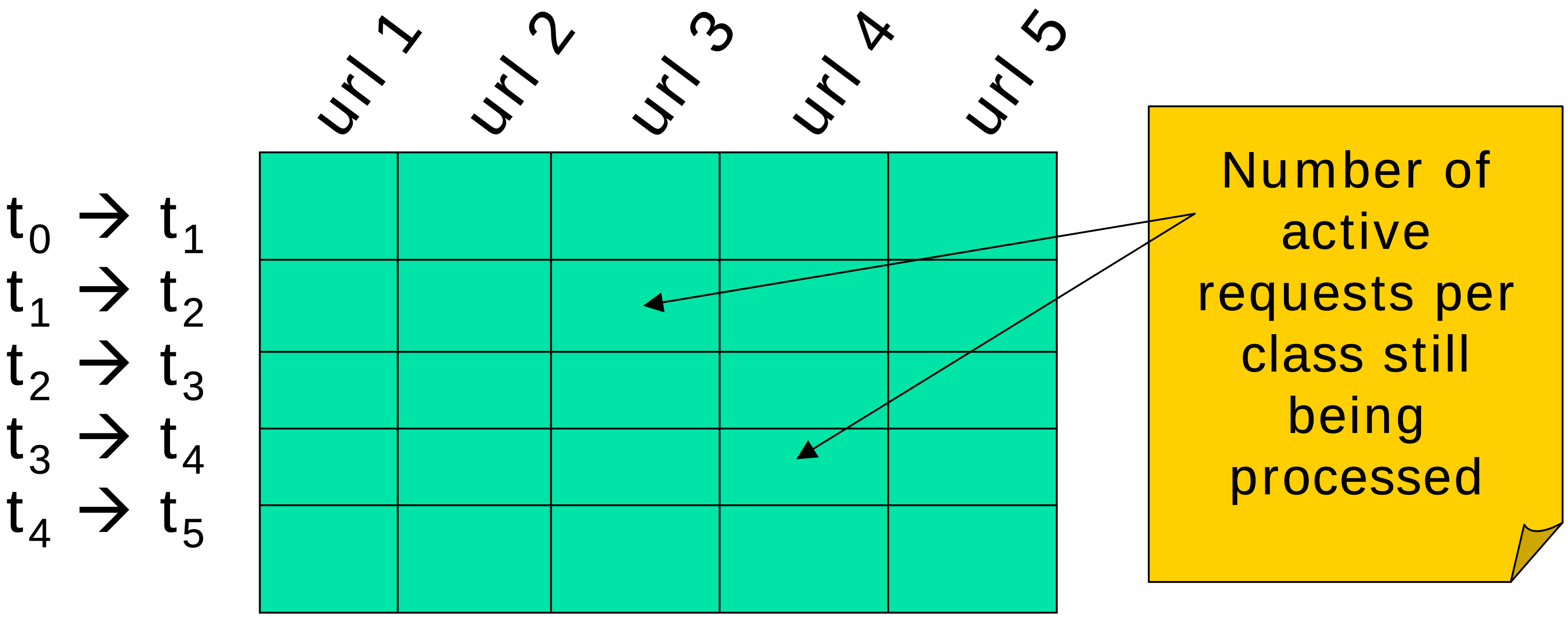
- Discover Elephant flows
  - Approach: Black box analysis of running system with statistical learning theory (SLT)
  - Minimal disruption to running system
    - Why? Fast-growing sites based on unmanaged open source software; hardware/software platforms which undergo frequent change
- Selective Admission Control
  - Goal in this case is a responsive system, even if “heavy” requests take more time
  - Approach: Network-level bandwidth shaping of elephant flows
    - Web-server independent actuator
    - HTTP-level pushback ok as well

1- Black box component monitoring

- Goal is to provide operator with hints of elephant flows
- Without fine-grained O/S instrumentation
  - Underlying components often change
  - Hooks often os/driver specific
  - Heisenberg principle (at least perception of)
- [Cohen04, Barham04] Finer-grained instrumentation of components leads to better request effect discovery
  - Their approach complementary to this work

1- Black box component monitoring

- From Web server's Apache logs:

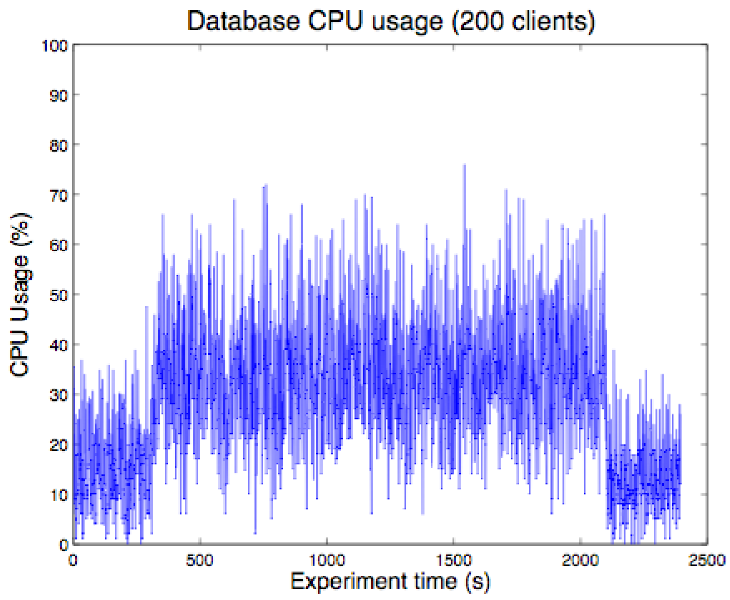


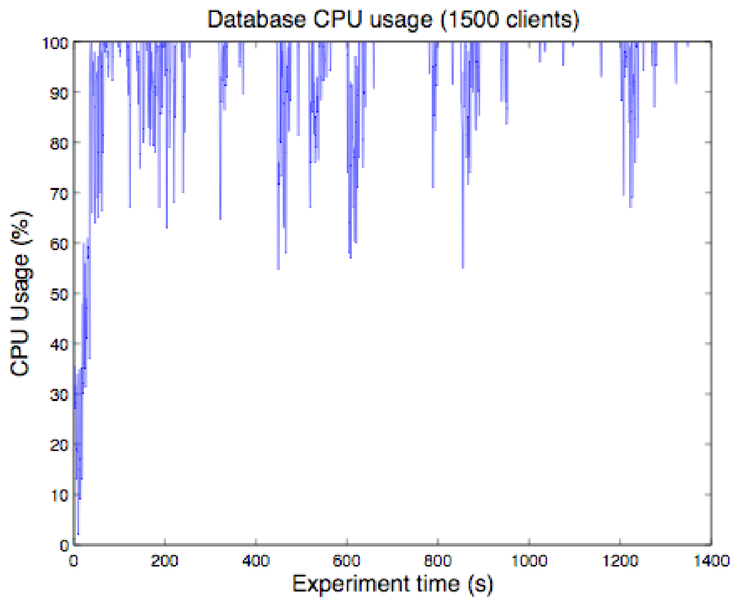

```
10.1.1.2 20296 + 1377 1102213360 0 /PHP/RUBiS_logo.jpg
10.1.1.2 1393 + 1375 1102213360 0 /PHP/SearchItemsByCategory.php
10.1.1.2 3736 + 1390 1102213360 0 /PHP/BrowseCategories.php
```

Request duration

Data collected: 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





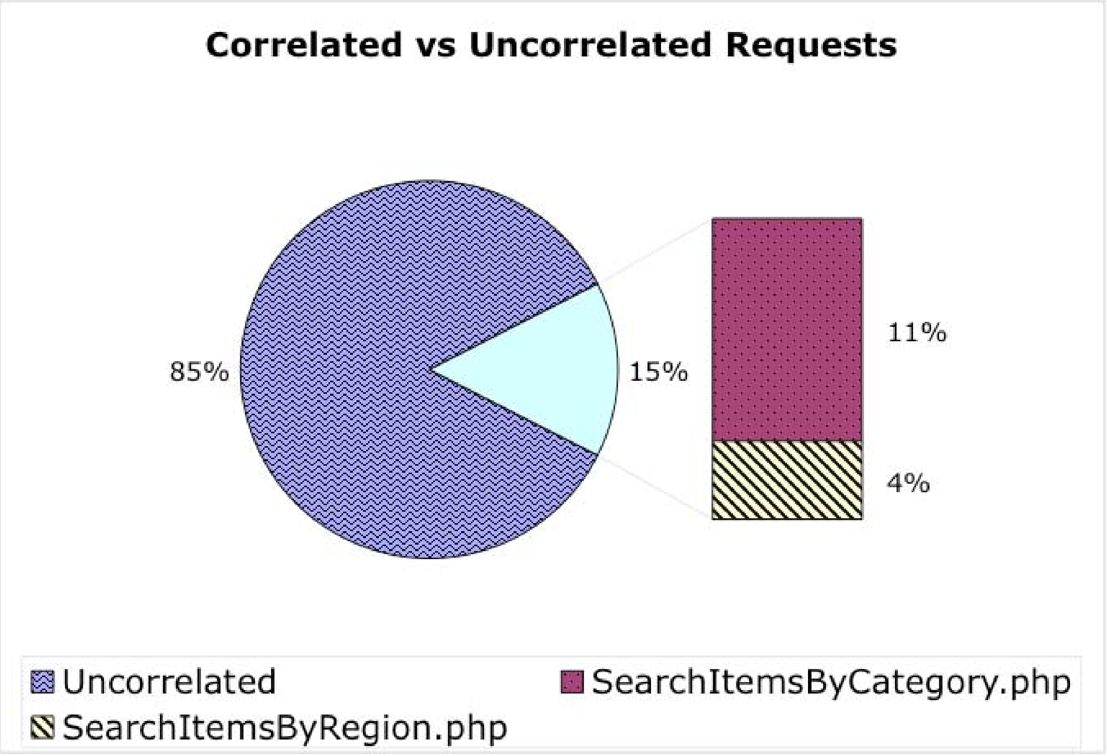
2- Finding correlated requests (elephants)

- Pearson's correlation coefficient
  - Easy to use, quick
- Run periodically for each measured parameter
  - Web server CPU, DB cpu, disk I/o activity, O/S context switching
- Produces candidate set of requests
- Some unexpected results--namely, fewer correlated URLs than expected
  - Browse seemed to be a superset of search, for example

	Pval	coeff
BrowseCategories.php	0.1747	-0.035
BrowseRegions.php	0.0926	-0.0434
SearchItemsByCategory.php	**0**	**0.5654**
SearchItemsByRegion.php	**0.0034**	**0.0756**
AboutMe.php	0.7702	0.0075
RegisterUser.php	0.4876	-0.0179
SellItemForm.php	0.4891	0.0179
RegisterItem.php	0.8767	0.004
ViewItem.php	0.0953	-0.0431
PutComment.php	0.5157	-0.0168
ViewUserInfo.php	0.4646	-0.0189
PutBidAuth.php	0.8641	-0.0044
PutBid.php	0.2566	-0.0293
BuyNowAuth.php	0.971	-0.0009
BuyNow.php	0.1206	0.0401
ViewBidHistory.php	0.9741	-0.0008

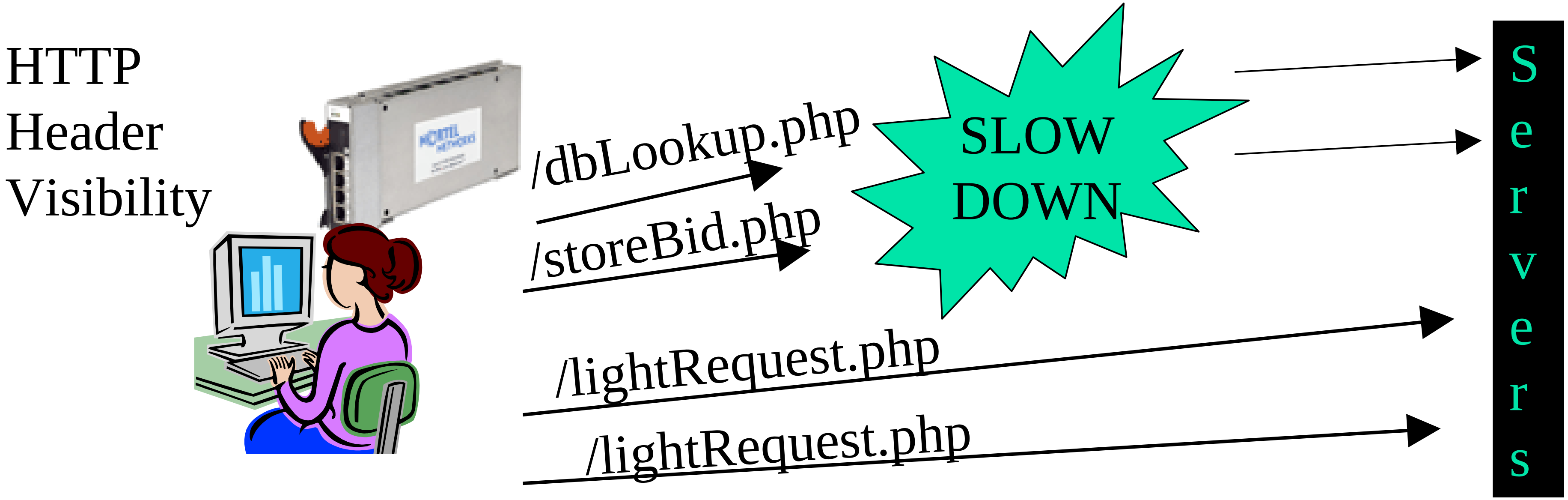
3- Visualization tool for results

- Allows network operators to include domain-specific knowledge
- Entry-point for operator in the loop
- Can enhance “top talkers” graph
- Development in progress



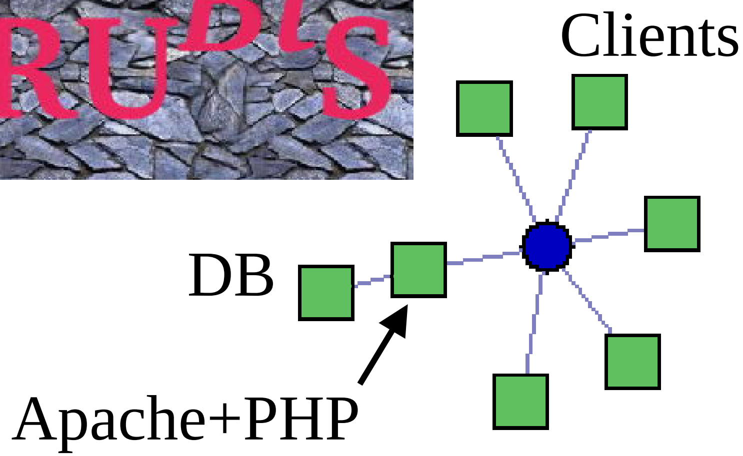
4- Effective actuators for new policies

- Need for network-level action point with HTTP header visibility
- Commercial products such as Nortel Alteon Web Switch
- Part of “iBox” project at Berkeley
  - Per-session packet tagging and bandwidth fencing system
  - In our BladeCenter testbed, use of 802.1q VLAN tags and Linux ‘tc’ extensions

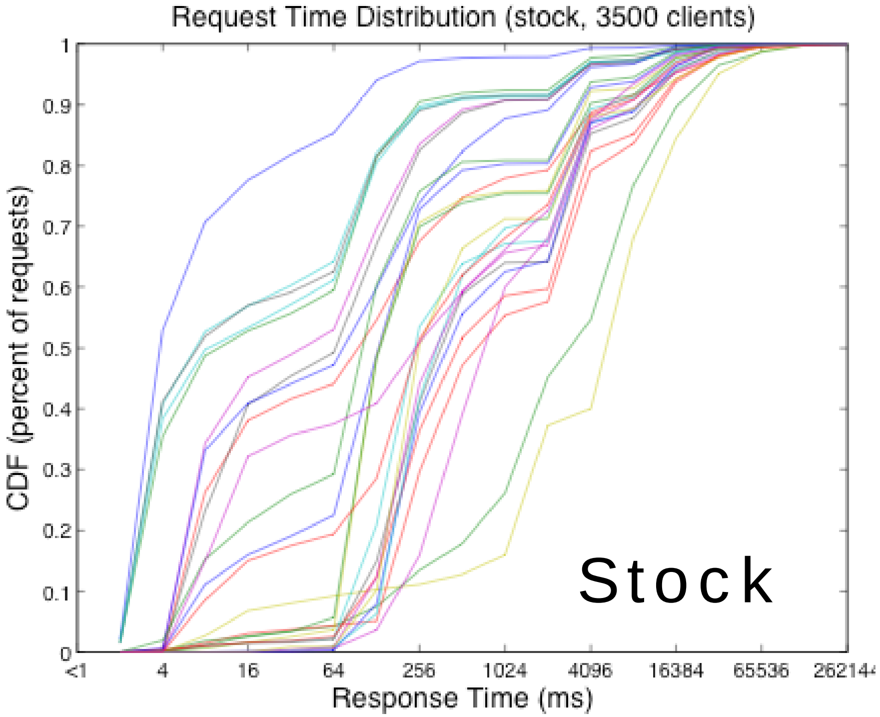


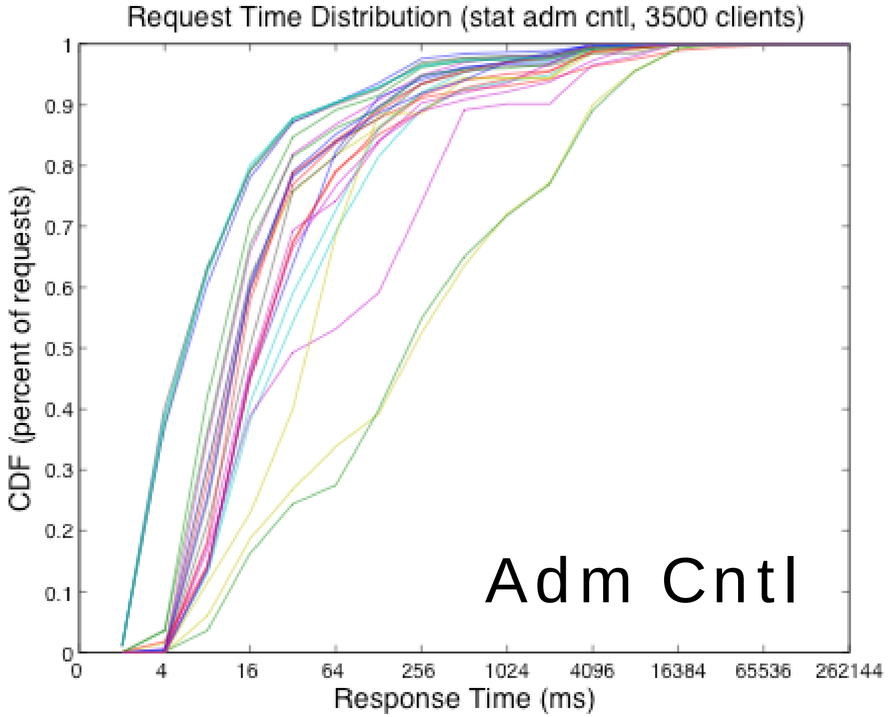
Experimental setup

- IBM BladeCenter testbed
  - Reconfigurable interconnect, linux-based platform, 12 2x3 Ghz Pentiums
- RUBiS (Rice Univ. Bidding System)
  - eBay like workload, transition matrix driven
  - Default matrix, 7 sec
- 10 client machines
- Apache + PHP app
- MySQL DB server
- Nortel Alteon HTTP parsing with 802.1q VLAN tagging + Linux tc extentions for b/w shaping



Request time distribution results





	stock	adm control
total requests	756137	1143264
correlated URLs	112521	105964
req/sec (avg)	462	782
session time (avg)	670	872
max request time	154.7	32.7

Conclusions

- Need for more self-managed web services
- Role for ultra-lightweight mechanisms in addition to fine-grained solutions
- Four mechanisms to enable this
    1. Black-box component monitoring
    2. Ultra-lightweight request effect discovery
    3. Visualizing correlations
    4. Network-level, selective request throttling
- Operator in the loop beneficial for many web service operators