Containment vs. Control: Keeping Busy Internet Servers Well-Behaved

Matt Welsh and David Culler

UC Berkeley Computer Science Division

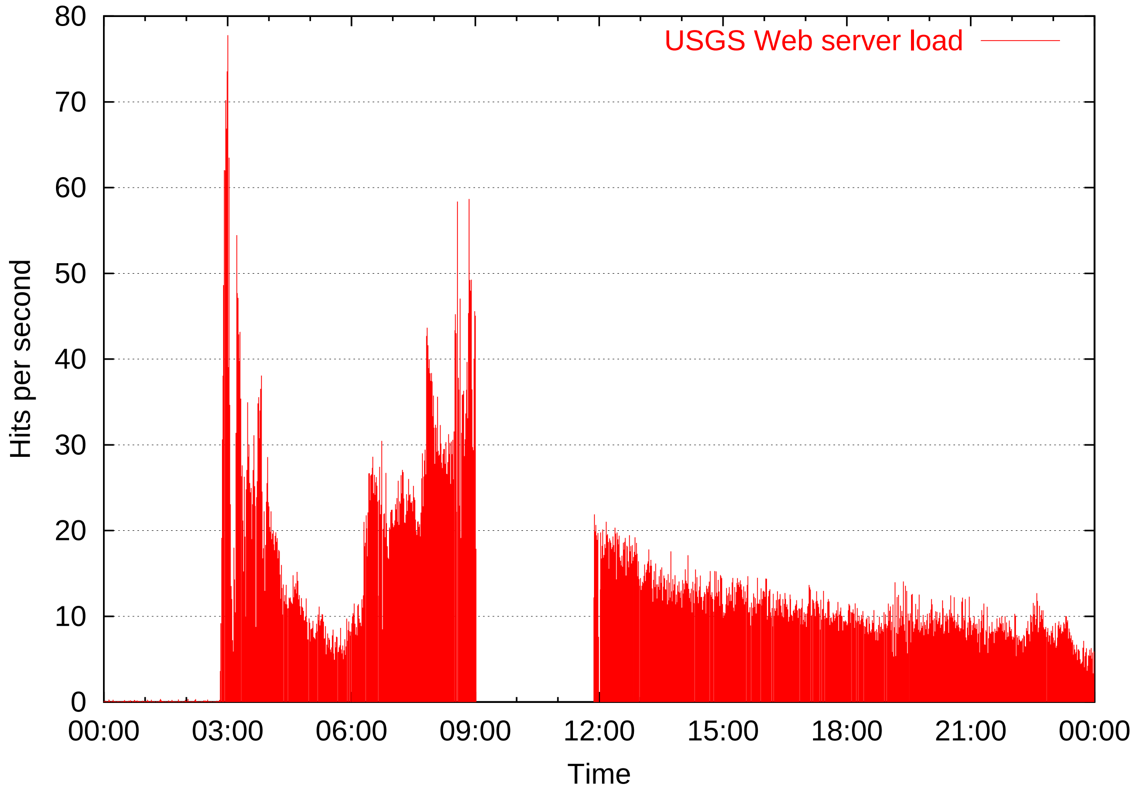
mdw@cs.berkeley.edu

USENIX'01, Boston, June 29, 2001

The Problem

Internet servers must be robust to extreme load

- Massive concurrency demands: 10,000s of users
- Continuous availability requirements
- Experience huge variations in load



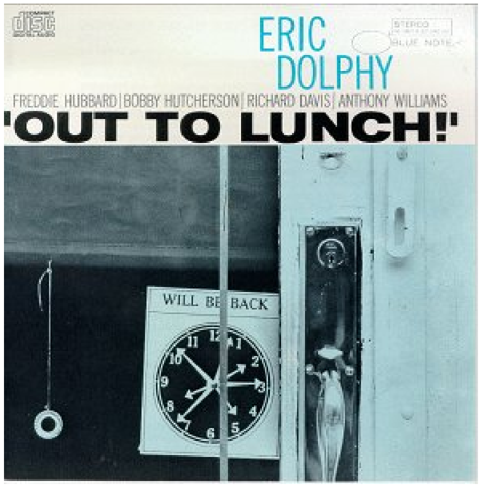
Keeping Servers Within Their Operating Regime

Admission control

- Don't let in more requests than you *think* you can handle
- Everyone queues up outside of the server
- TCP exponential backoff means you wait a long time!

Degraded service

- Service users more quickly, but provide less useful service
- e.g., ''We're busy, come back later''
- Lower satisfaction, but at least the site isn't dead


ERIC DOLPHY
FREDDIE HUBBARD | BOBBY HUTCHERSON | RICHARD DAVIS | ANTHONY WILLIAMS
'OUT TO LUNCH!'
WILL BE BACK

Mitigating Resource Demands

Classic approach: Resource containment

- Widely used by RTOSs, multimedia systems, etc.
- Divvy up resources to individual users/components by share
- Strictly *prevent* resources from being overcommitted

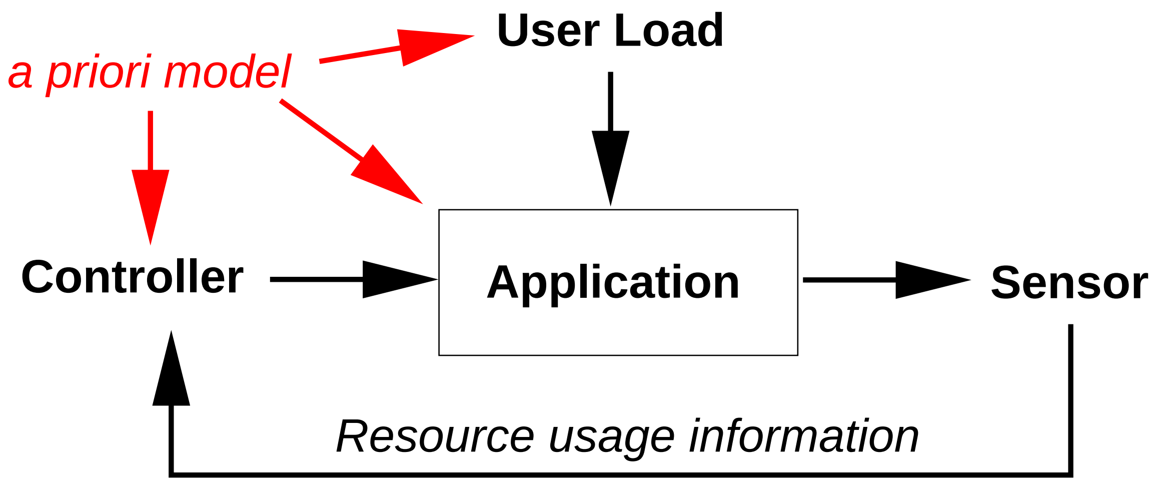
An alternative: Dynamic Control

- Observe behavior of system at runtime
- Exert *control* over resource usage, request stream, quality-of-service
- Allows temporary resource overcommittment and more flexible policy

Key problem:

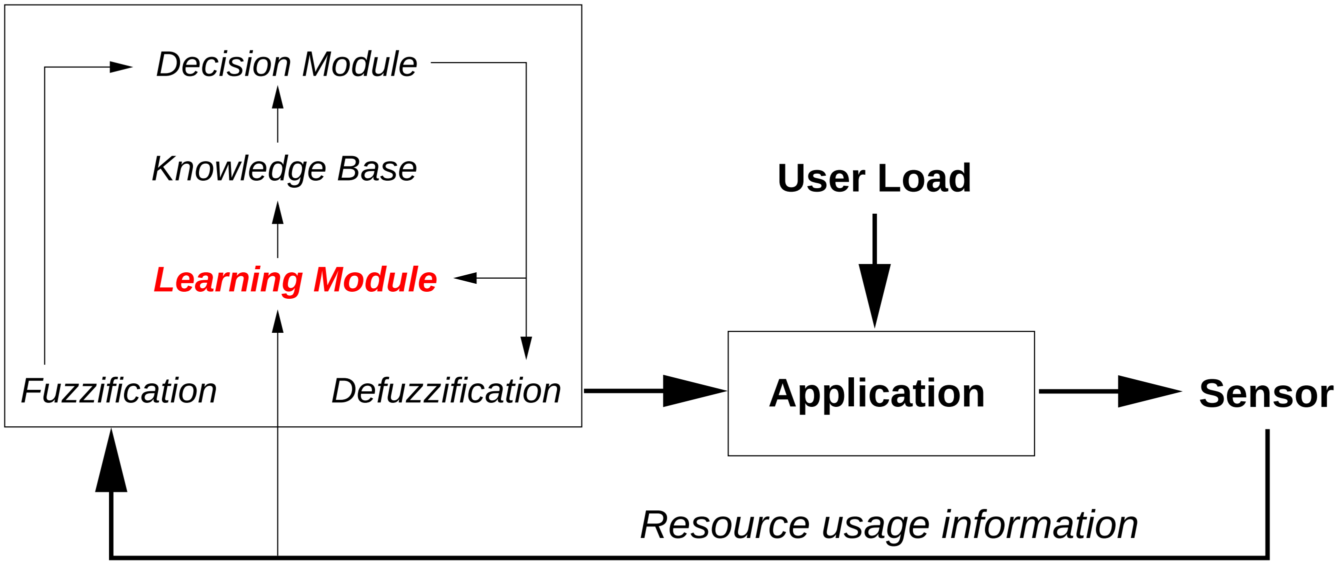
How do we know the demand that clients place on the system?

Classic Control-Theoretic Approach



- Construct accurate model of client load and system behavior
- Requires much a priori knowledge
  - ▷ *Difficult to model response of nonlinear systems!*
- Build controllers based on expected input/output relationship
  - ▷ *Can apply stability and performance analysis*

Self-organizing Fuzzy Control



Use fuzzy logic to drive control decisions

- Avoids many limitations of classic approach
- Requires knowledge base of heuristic control rules
- e.g., *If memory usage HIGH then set queue threshold LOW*

Automatically derive control logic from experience

- Too many parameters to define logic rules manually
- Alternative is self-organizing control
- Various approaches: genetic algorithms, neural systems, etc.

Summary

Busy Internet servers are exceedingly complex

- Dynamic content generation, system composition, failures
- Load is extremely bursty
- Overprovisioning is generally infeasible

How to keeping systems in their operating regime?

- Is it possible to accurately predict resource usage?
- Conservative approaches fail when load exceeds capacity
- Alternative: Observe and control system behavior externally

Classic control vs. modern approaches?

- Difficult to generate accurate (or linear) model of system behavior
- Draw from AI techniques to solve the problem
- Fuzzy/neural control -- is it worth the effort?

For more information:
`http://www.cs.berkeley.edu/~mdw/proj/seda/`