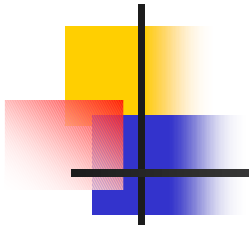


Cluster Reserves: A Mechanism for Resource Management in Cluster-based Network Servers

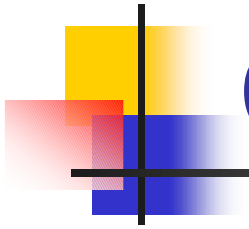
Mohit Aron

Peter Druschel
Department of Computer Science
Rice University
{aron,druschel,willy}@cs.rice.edu

Willy Zwaenepoel

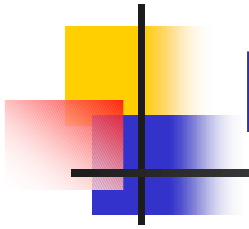
Srinath Rao

DARK Lab



Cluster-based Network Servers

- Why Cluster-based servers?
- Objective: Achieve maximum Throughput (Number of requests serviced per second)
- Load balancing among the server nodes
- Co-operative Caching of contents
- Different kinds
  - WRR
  - LARD
  - L2S
  - PRESS



Performance Isolation

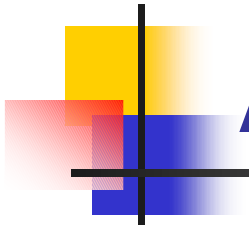
- Host variety of services

e.g. retrieval of static and dynamic Web pages, online databases for information retrieval

- Reserve minimal proportion of server resources for a service and/or a Client community

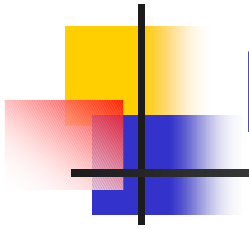
e.g. Web hosting, e-commerce site, organization

- Service Class
- Problem: How to allocate Cluster resources to achieve performance isolation?



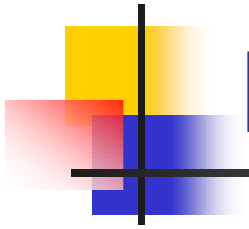
A solution

- Provide separate server nodes for each service class
  - lower avg utilization of resources
  - Higher avg request latencies



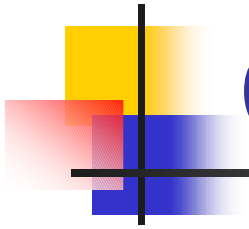
Desired solution

- Permit resources to be proportioned between the service classes
- Given sufficient load, a service class receives resources independent of the load on others
- Distribute idle resources amongst other service classes



Resource Containers

- Current OS features not effective
- Performance isolation in single-node Web servers
- OS abstraction for resource principals
- Principals compete with each other
- Allow accurate accounting and scheduling of resources



Cluster Reserves

- Cluster-wide resource principals
- Combine resource principals on individual nodes
- Cluster resource manager
- Service class – cluster resource principal
- Resource - CPU time, memory, disk, network bandwidth

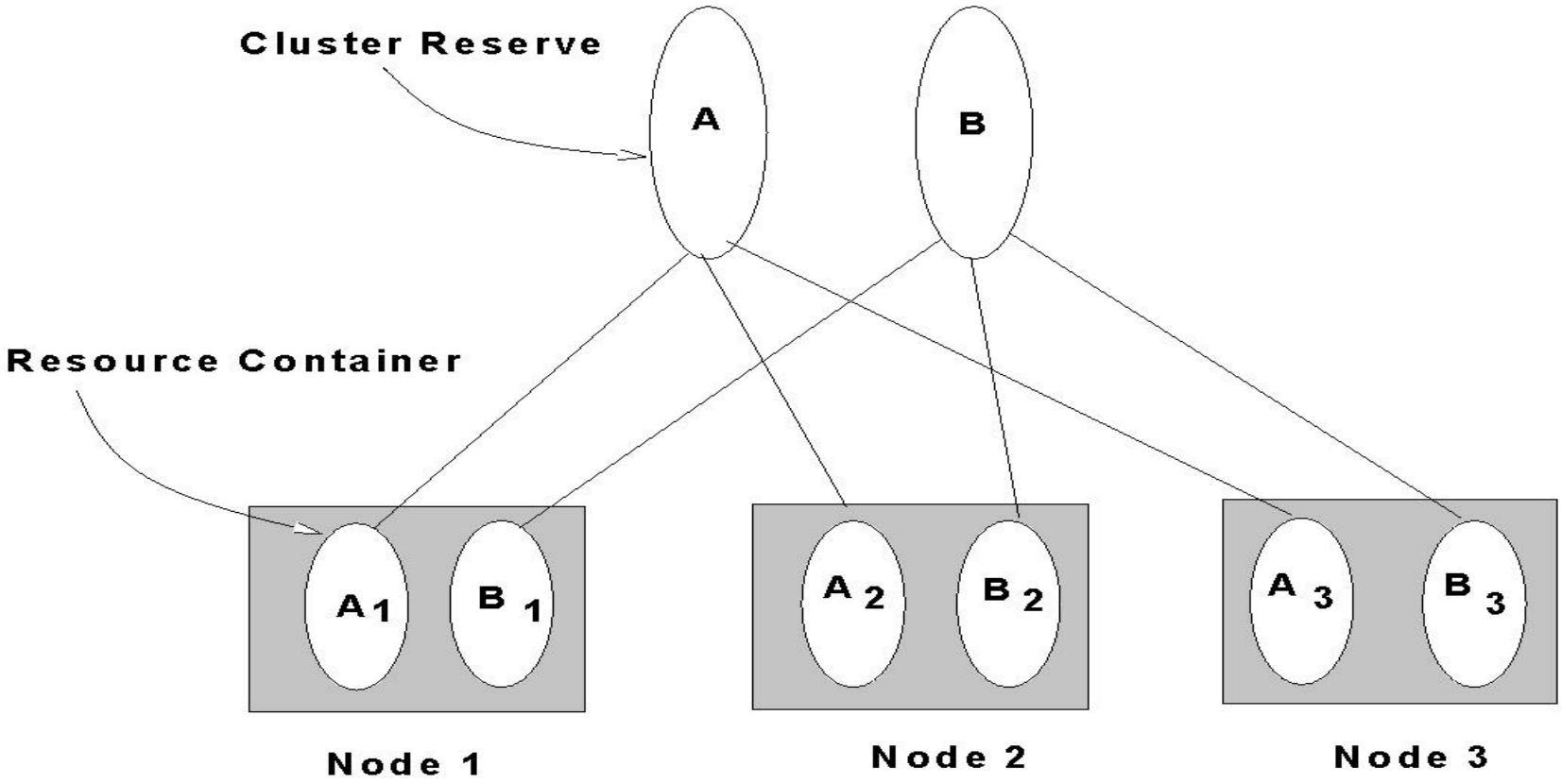
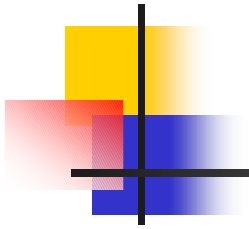


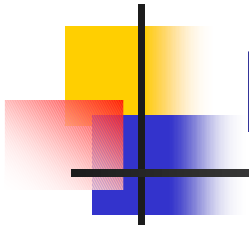
Figure 1: Cluster Reserves

A=B=50%
A1=B1=A2=B2=A3=B3=50%
A1=B2=100%,B1=A2=0%, A3=B3=50%



Cluster resource manager

- Computes partitioning - dynamic
- Collects resource usage statistics
- Target cluster allocations
- Maps the allocation problem to an equivalent constrained optimization problem
- Solution yields individual per-node resource allocations
- Resource sink



Maths

- N nodes, S service classes
- I/p

r,u: NxS matrices,D:vector of size S

$r_{i,j}$: % resource allocation

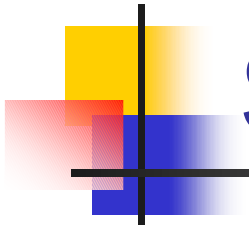
$u_{i,j}$: % resource usage

D_j: desired % resource allocation

- O/p

R: NxS matrix

$R_{i,j}$: new % resource allocation



Steps

- Compute the least feasible deviation between the desired and actual deviations
- Compute new resource allocations
  - Deviation computed is achieved
  - Resource allocations are close to service class usage
- Distribute unassigned cluster resources to idle service classes

Step 1

Objective:

$$Minimize \sum_{j=1}^{S} | \sum_{i=1}^{N} R_{ij} - N * D_j |$$

Constraints:

$$\forall_{i=1}^{N} \sum_{j=1}^{S} R_{ij} \leq 100$$

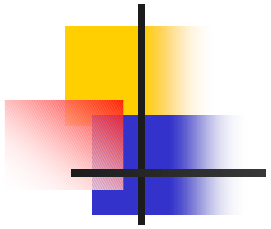
$$\forall_{i,j} \; R_{ij} \leq u_{ij} \; if \; r_{ij} > u_{ij}$$

$$\forall_{i,j} \; R_{ij} \geq 1$$

Step 2

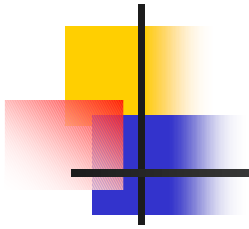
- Add another constraint

$$Minimize \sum_{i=1}^{N} \sum_{j=1}^{S} (R_{ij} - (u_{ij} + k_{ij}))^2$$



Call #		% usage (sink) Svc 1	% usage (sink) Svc 2	% allocation Svc 1	% allocation Svc 2
1	Node 1	50 (1)	50 (1)	50	50
	Node 2	50 (1)	50 (1)	50	50
2	Node 1	40 (0)	60 (1)	40	60
	Node 2	50 (1)	50 (1)	60	40
3	Node 1	40 (1)	60 (1)	41	59
	Node 2	60 (1)	40 (1)	59	41
4	Node 1	41 (1)	59 (1)	42	58
	Node 2	59 (1)	41 (1)	58	42
5	Node 1	40 (0)	60 (1)	40	60
	Node 2	58 (1)	42 (1)	60	40
6	Node 1	40 (1)	60 (1)	41	59
	Node 2	60 (1)	40 (1)	59	41
27	Node 1	48.9 (1)	51.1 (1)	49	51
	Node 2	51.1 (1)	48.9 (1)	51	49

Table 1: Dynamics of the Resource Manager



Implementation

- 300 MHz PII machines, 128 MB of RAM, Free BSD-2.2.6 OS
- 7 Pentium Pro 166 MHz client machines
- Resource containers implemented with lottery scheduling scheme
- Resource manager runs on a separate cluster node
- Tracker – communicates with resource manager
- S-client client program
- Apache web server at server nodes
- LOQO tool for solving optimization problem
- Switched 100Mbps ethernet

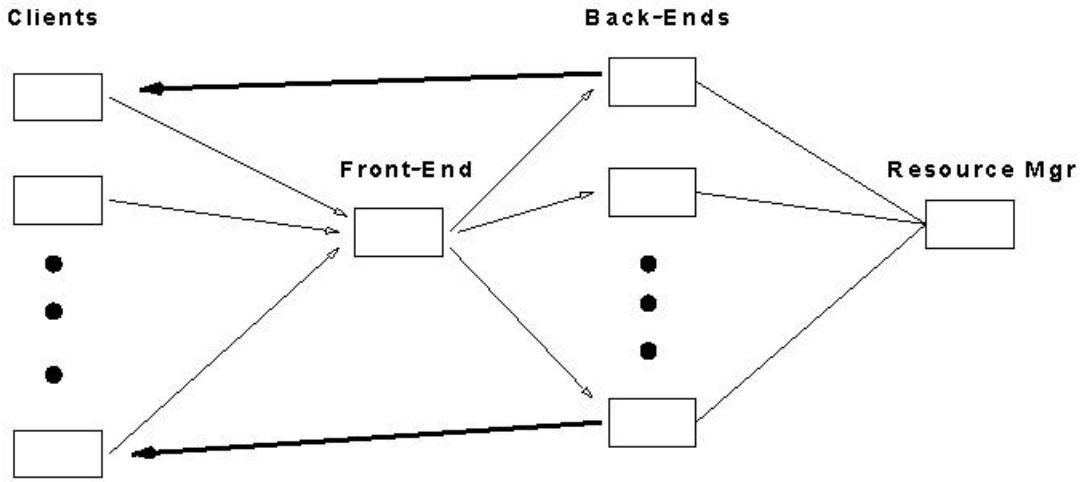


Figure 2: LAN configuration

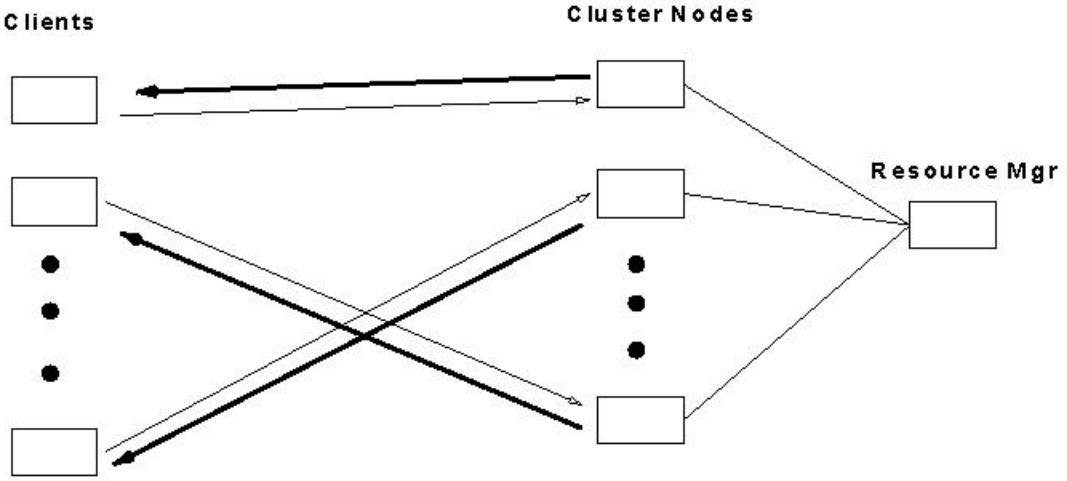


Figure 3: Geographically Distributed Nodes

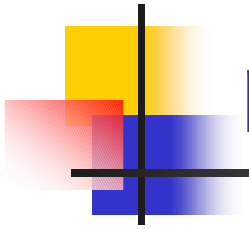
Performance isolation via node separation

- 4 different web server logs
- 4 service classes
- Each service allotted 25%

	Disjoint	Shared	
		WRR	LARD
Xput (conn/s)	252 (1.0)	517 (2.0)	1214 (4.8)
CPU util. (%)	15	35	60

Table 3: Disjoint vs shared cluster use

Performance isolation via per-node resource allocations



- Static per-node resource assignment
- WRR strategy
- 4 back-end nodes, 7 client machines
- 5 service classes
- 20% allocation for each class
- Synthetic trace : repeated set of 5 requests

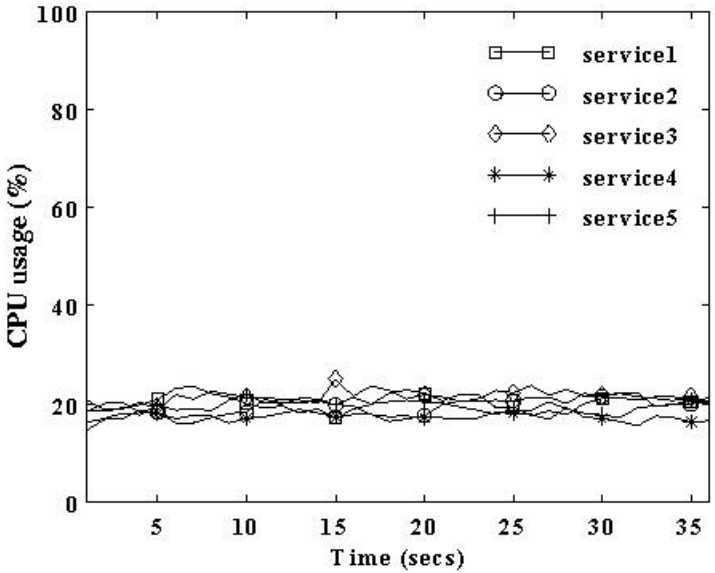


Figure 4: Typical node usage

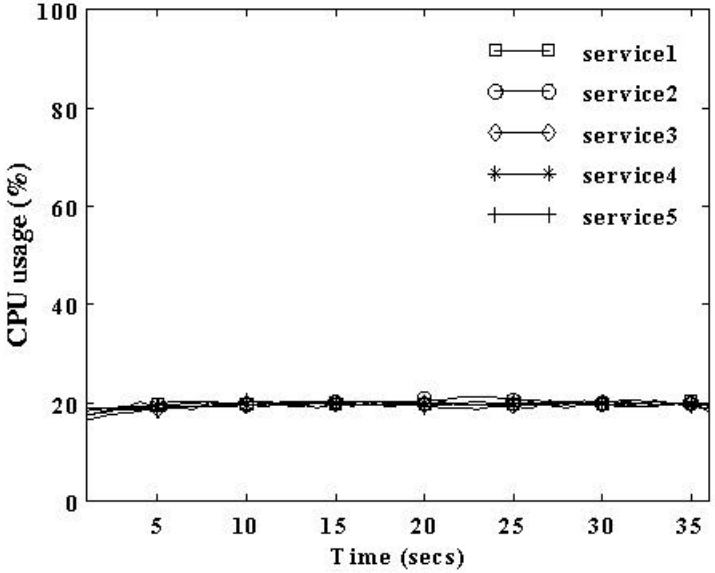
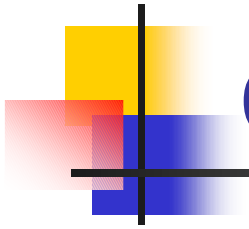


Figure 5: Cluster-wide usage

Geographically distributed clusters



- 4 cluster nodes
- 5 service classes
- First 2 nodes receive request for the first service class, all receive requests for all other service classes
- 20% allocation for each service class

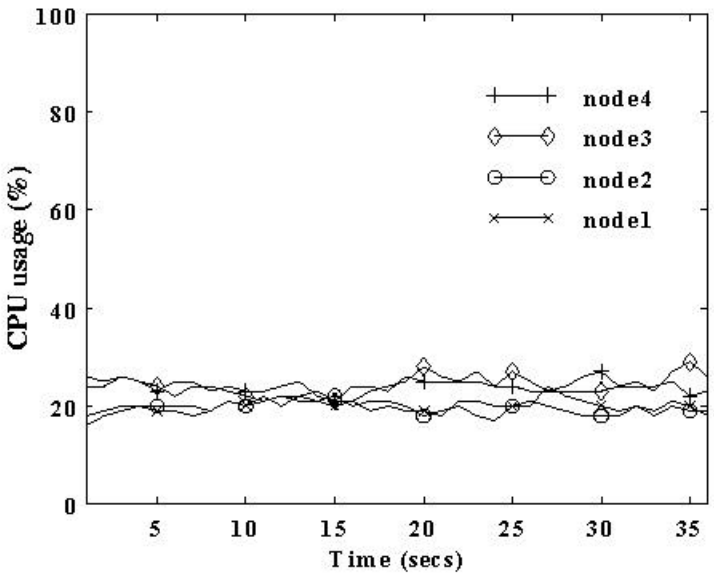


Figure 6: Typical service usage

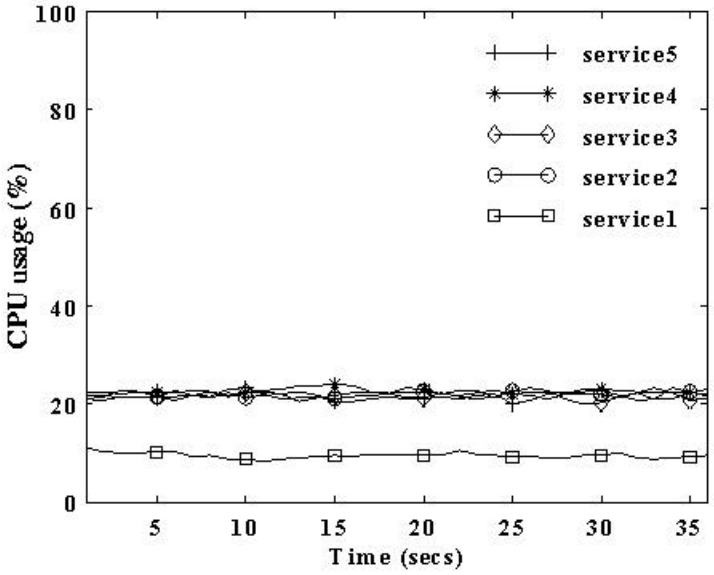


Figure 7: Cluster-wide usage

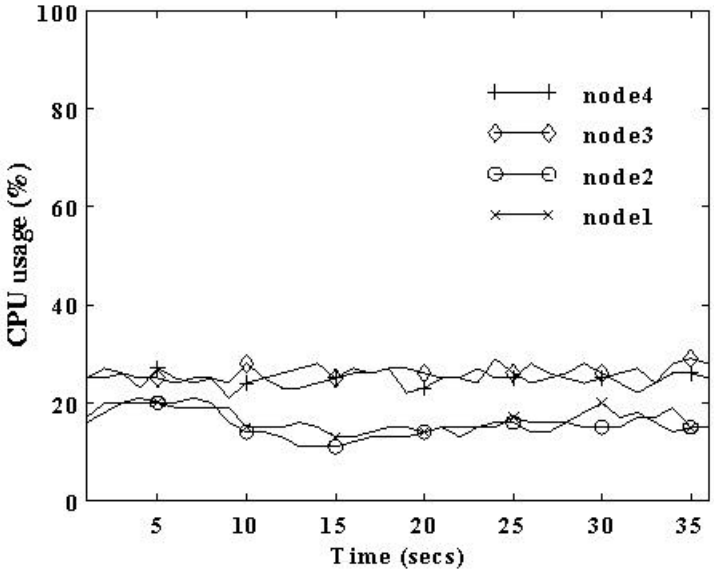


Figure 8: Typical service usage

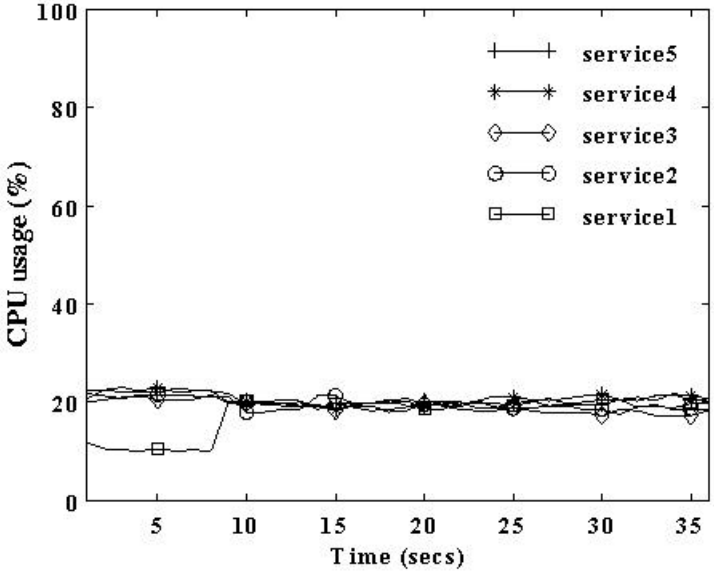
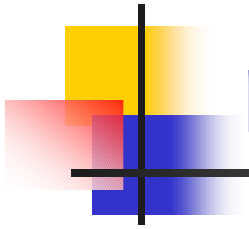


Figure 9: Cluster-wide usage

Sparse, resource intensive requests



- E.g. rendering of maps
- 5 service class, 4 back-end nodes, WRR strategy, 20% allocation scheme
- Service class 1 requests access CGI script that runs for 10 seconds. Only one such outstanding request
- Others 6KB static file

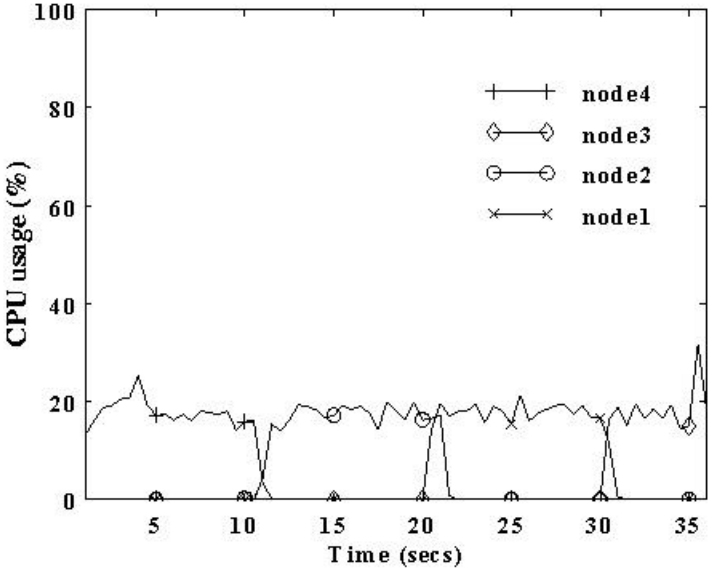


Figure 10: Usage for service 1

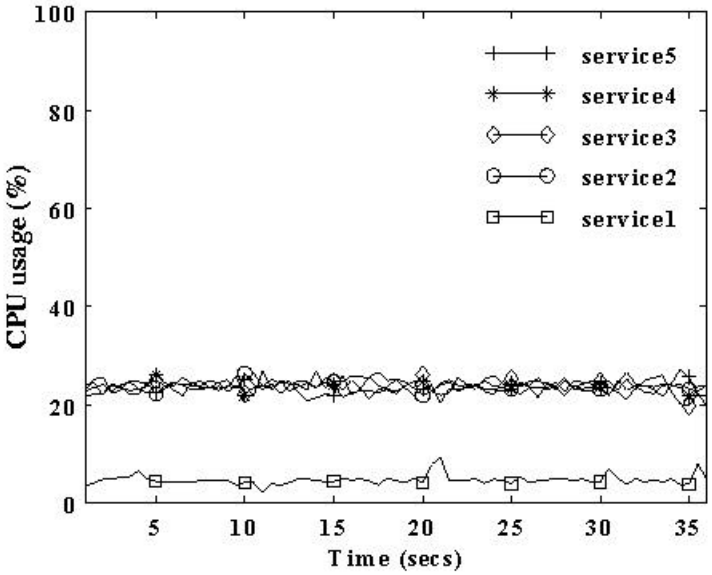


Figure 11: Cluster-wide usage

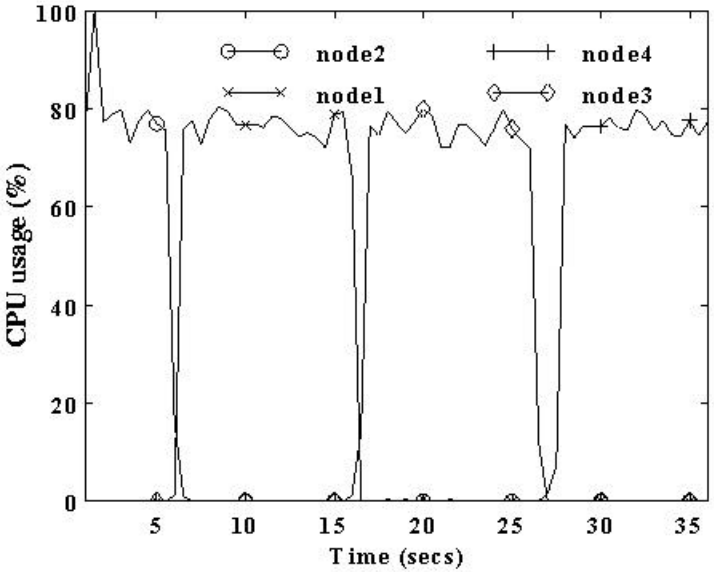


Figure 12: Usage for service 1

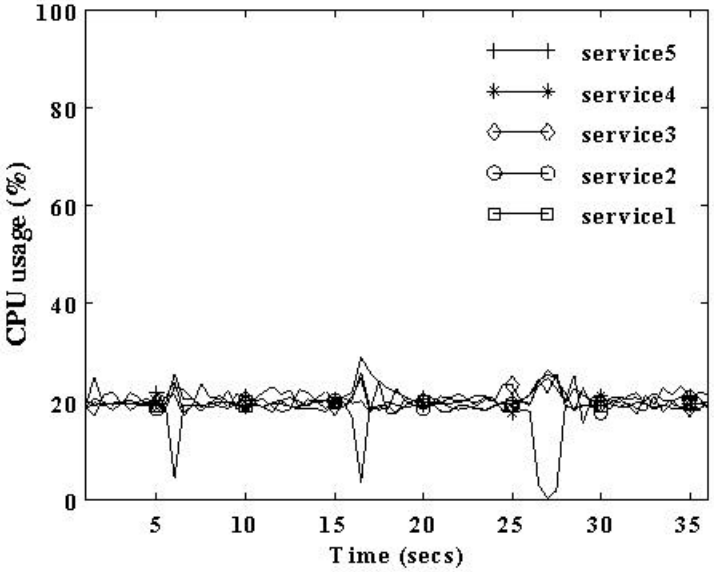
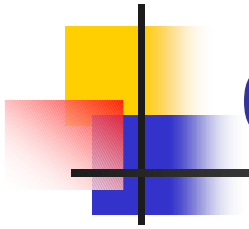


Figure 13: Cluster-wide usage

Content-based request distribution



- 4 back-end nodes, LARD scheme
- 3 service classes, 33% allocation
- 358 MB, 24 MB, 193 MB dataset

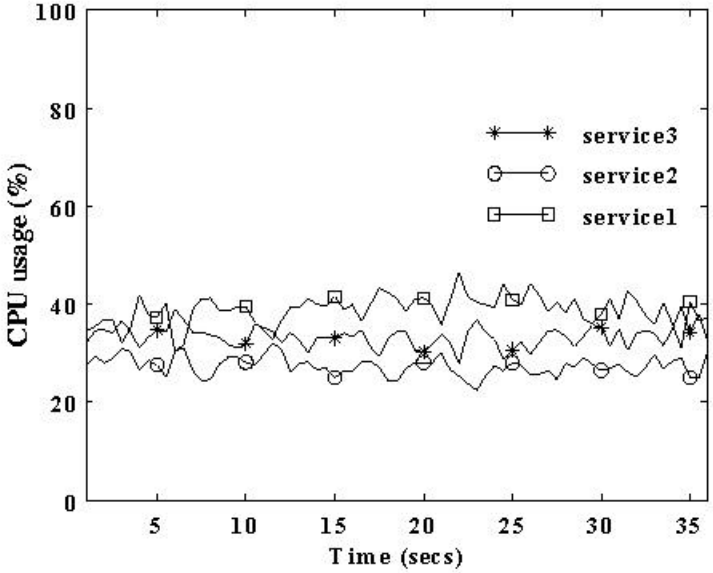


Figure 14: No Cluster Reserves

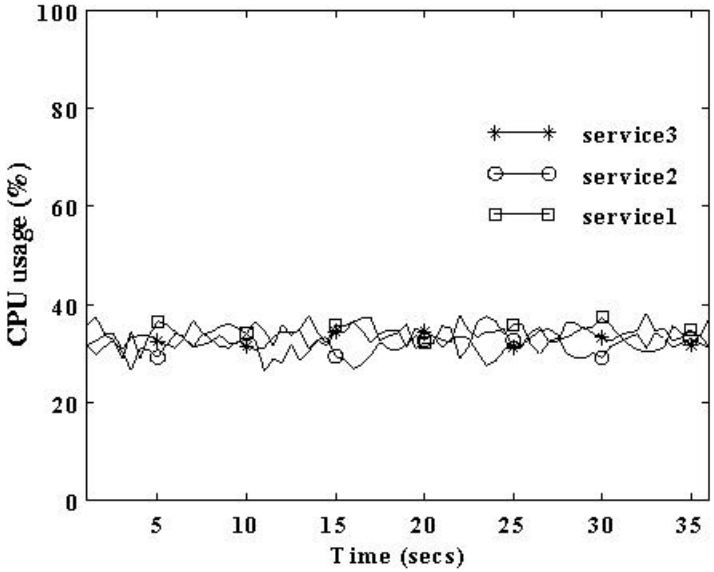
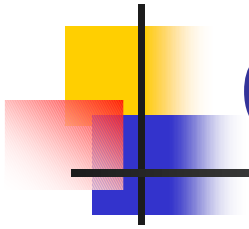


Figure 15: With Cluster Reserves



Conclusions

- Extends existing mechanisms for performance isolation to a cluster
- Resource management needed among services that share a set of clusters
- Results show higher resource utilization and improved performance
- An effective solution for providing performance isolation in cluster-based web servers