

Scheduling Optimization for Resource-Intensive Web Requests on Server Clusters

Huican Zhu, Ben Smith and Tao Yang

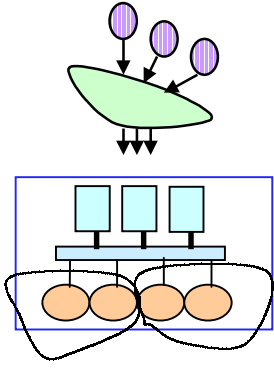
(hczhu, besmith, tyang)@cs.ucsb.edu

Department of Computer Science

University of California, Santa Barbara

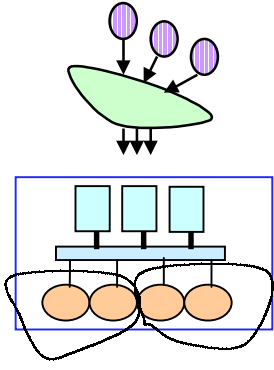
<http://www.cs.ucsb.edu/research/rcgi/>

ACM Symp. on Parallel Algo. & Architecture (SPAA99)



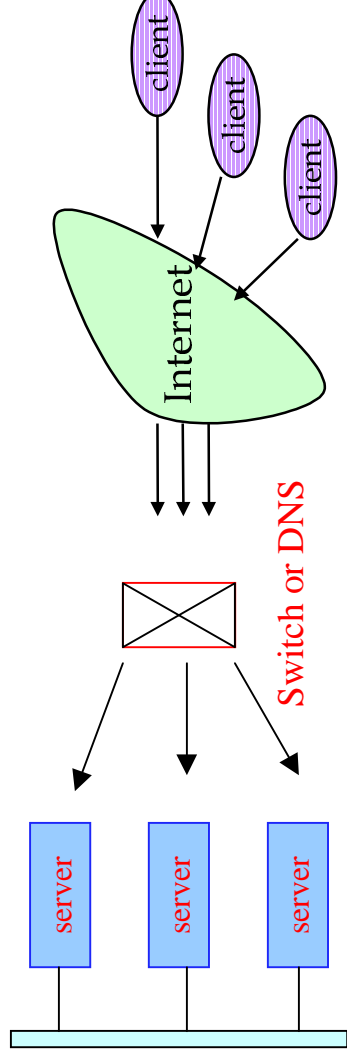
Motivation for clustering and scheduling optimization

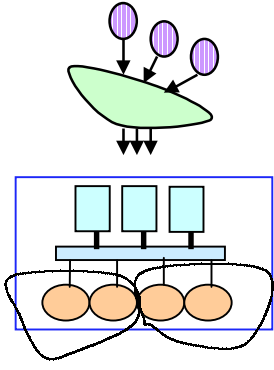
- ◆ Large request volumes
 - ◆ Yahoo: 250 million per day (3000/second)
 - ◆ Solution: Cluster servers to handle load
- ◆ Nonuniform resource requirements
 - ◆ Static (files) and dynamic (CGI) content
- ◆ Dynamic requests require more resources
 - ◆ 1 to 2 orders of magnitude longer processing time than static requests based on IBM Olympics and Alexandria Digital Library data



Flat architecture for Web server clustering

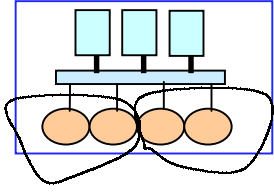
- ◆ Example systems: SWEB at UCSB, NCSA Web servers, LARD (Rice), Cisco LocalDirector
- ◆ Server nodes are linked to a network. All nodes can process dynamic or static requests.
- ◆ Requests are routed to one of the Web servers through DNS, single-IP switch or a special front-end node.



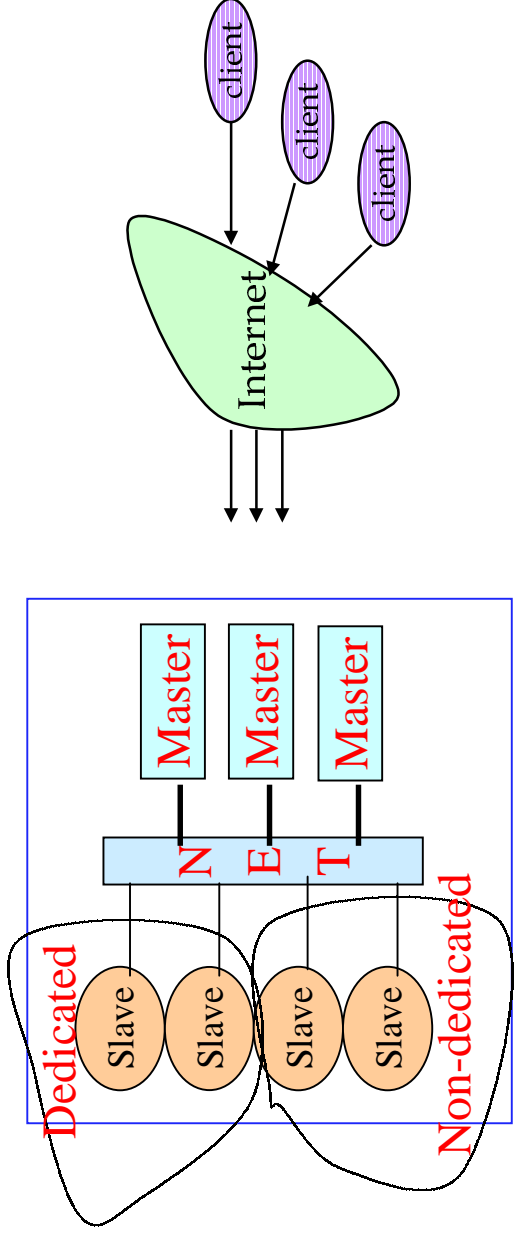
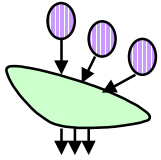


Layered architecture for network services

- ◆ The TACC architecture [SOSP'97] by Fox et al uses a two-level scheme (front-ends and workers) for network services
 - ◆ Incremental scalability and fault tolerance
 - ◆ No detailed design for Web servers
 - ◆ No study on performance differences between flat architecture and layered architecture
- ◆ Our work focuses on Web servers
 - ◆ Study when layered architecture is better than flat
 - ◆ Propose optimizations to ensure better performance



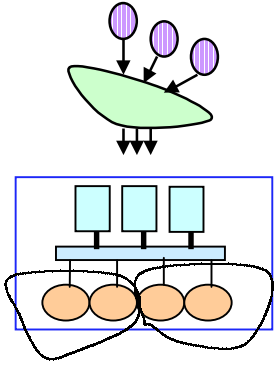
Our Master/Slave Architecture with Remote CGI



Server nodes are divided in two groups:

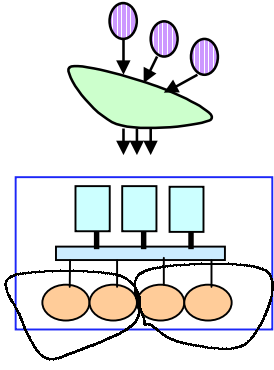
Slave group(dedicated or non-dedicated) processes dynamic requests

Master group handles static requests and part of dynamic requests



Justifications for M/S architecture

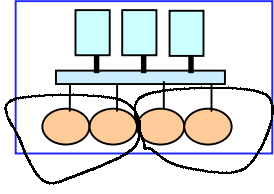
- ◆ Better fault tolerance and easy to recruit idle resources (advantages studied by TACC)
 - ◆ Minimize the number of IP addresses exposed to the Internet
- ◆ Few master servers can saturate outgoing bandwidth
 - ◆ Process static requests at > 2000 requests/sec.
 - ◆ Saturate network bandwidth $> 254\text{Mb/s}$.
 - ◆ Handling of dynamic requests is server bottleneck
- ◆ Separation of static content and dynamic requests can lead to better system performance
- ◆ Remote CGI can be more efficient than local CGI.



Remote CGI vs. Local CGI

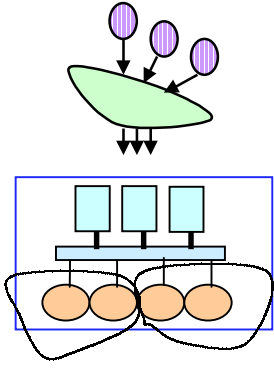
- ◆ **Implementation:**
 - ◆ Web server assembles request, redirects to a RCGI server. RCGI server fork's/exec's
- ◆ **Remote CGI is more efficient than local CGI**
 - ◆ Cheaper fork because of thin server on slave
 - ◆ Network overhead negligible

	Fork(1 thread /0 MB)	Local CGI	RCGI
Ultra-30	20.0ms	27.9ms	20.3ms



Proposed Optimizations for M/S

- ◆ Cluster partitioning
 - ◆ Divide server nodes into Masters and Slaves: what is the optimized choice?
- ◆ Node selection (per request basis) for better load balance
 - ◆ Select node based on node I/O & CPU resource availability and request resource requirement.
- ◆ Reservation based scheduling
 - ◆ Reserve resources on master nodes for static requests
 - ◆ Short requests get fair share of resources
- ◆ Inexpensive request reassignment mechanism compared to redirection method

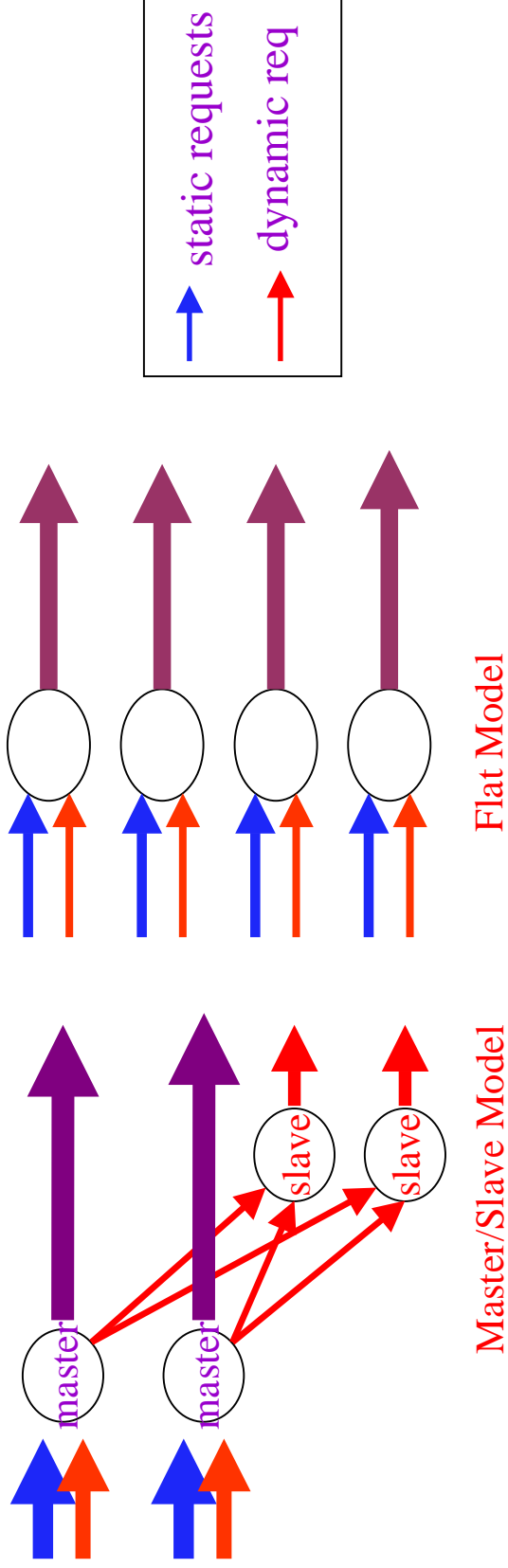


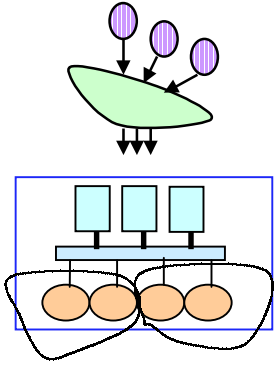
How to Partition a Server Cluster

Questions:

- 1: Given p nodes, what is the optimal number of master nodes?
- 2: What percentage of dynamic requests should be processed on masters?

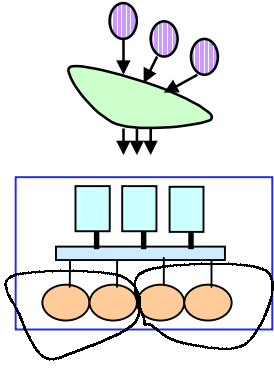
Goal: ensure the master/slave architecture outperforms flat architecture





Stretch Factor as Performance Metric

- ◆ **Stretch factor:** the ratio of response time at a particular load to that at no load
- ◆ Average stretch factor is more suited than average response time for systems with highly variable task sizes:
 - ◆ Majority of web requests are short, static requests.
 - ◆ Users are willing to wait longer for large tasks, but expect that small tasks complete quickly.
 - ◆ Average stretch factor indicates load of a system.



Theoretical Results

Given :

average arrival rates of static and dynamic requests,
 average response time of static and dynamic requests,

We define :

m : number of master nodes

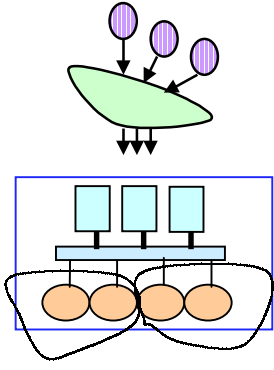
S_M : average stretch factors on the M/S architecture

S_F : average stretch factors on the Flat architecture

θ : percentage of dynamic requests executed on master nodes

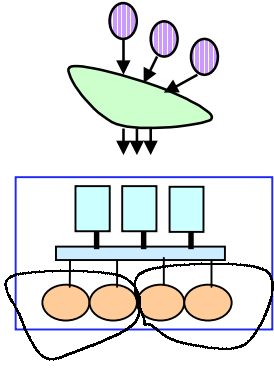
Results :

1. Given m sufficient to handle static requests, we can find a range for θ such that $S_M \leq S_F$.
2. The best m and θ can be calculated .



Cluster Scheduling with Practical Consideration

- ◆ Monitor arrival rates and response time of dynamic and static requests to adjust the reservation ratio dynamically
- ◆ Given a dynamic request received by a master, which node (slave or master) should process it?
 - ◆ Select a node with the least CPU/IO load, satisfying the resource reservation constraint. Combine two load indices using sampled weights.
 - ◆ Monitor CPU/IO usage and periodically exchange load information among nodes.



Selection of Best Node

- ◆ Best node is selected based on estimated stretch factor:

$$\frac{w}{CPUIdlePercentage} + \frac{1-w}{DiskAvailPercentage}$$

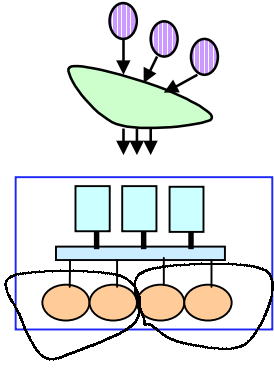
w: the average percentage of the time spent on CPU

1-w: the average percentage of the time spent on I/O

w and **1-w** is measured by offline sampling

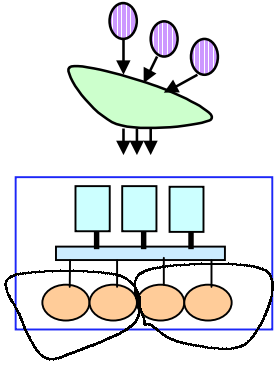
CPUIdlePercentage: percentage of CPU idle time

DiskAvailPercentage: percentage of disk bandwidth availability



Implementation of M/S and Performance Evaluation

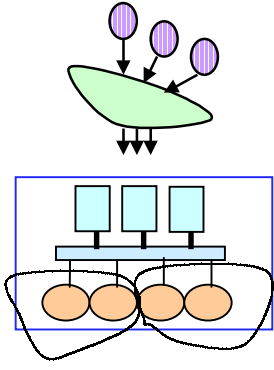
- ◆ M/S implemented on Solaris
- ◆ A Web simulator was designed to simulate behavior of large server clusters.
 - ◆ A Unix like CPU job scheduler and a storage access scheduler
 - ◆ Each job request is modeled as a sequence of CPU bursts and I/O bursts
 - ◆ Local network usage is not modeled
- ◆ Results validated on SUN UltraSparc cluster for small p.



Workload Description

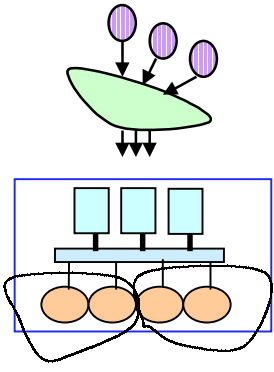
Log name	No. requests	CGI percentage
UCB	9.2 million	11.2
KSU	47364	29.1
ADL	73610	44.3

- ◆ UCB trace: CGI requests are replaced by artificial requests, **CPU intensive**.
- ◆ KSU trace: Search requests replaced by Webglimpse search, **Mixed**.
- ◆ ADL trace: Search requests are sent to local databases, **I/O intensive**.



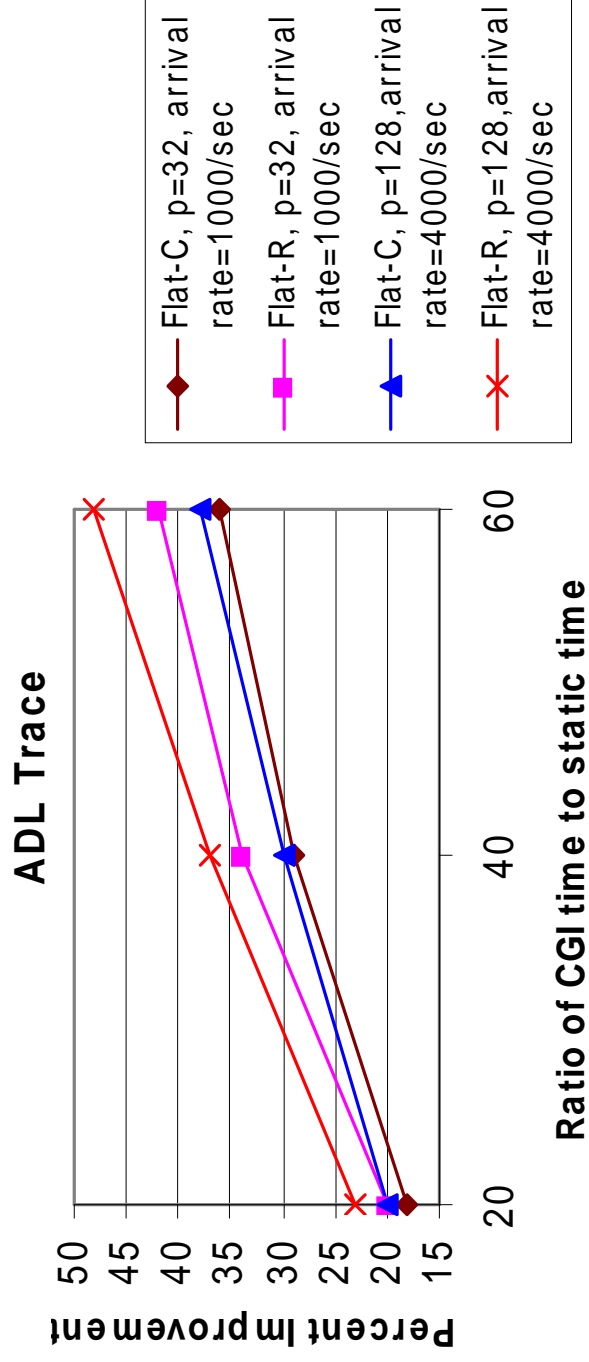
Different Scheduling Methods Evaluated

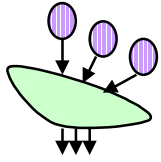
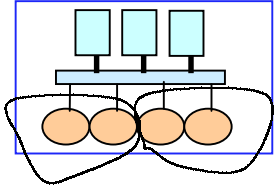
- ◆ For flat architecture:
 - ◆ Flat-R: Round-robin scheduling.
 - ◆ Flat-C: Least connection based scheduling.
- ◆ For M/S architecture:
 - ◆ M/S: Scheduling with resource reservation and request I/O and CPU resource requirement awareness.
 - ◆ M/S-ns: no sampling (Do not consider specific request I/O and CPU resource requirement patterns)
 - ◆ M/S-nr: no resource reservation for static requests.



Simulation Results: M/S vs Flat

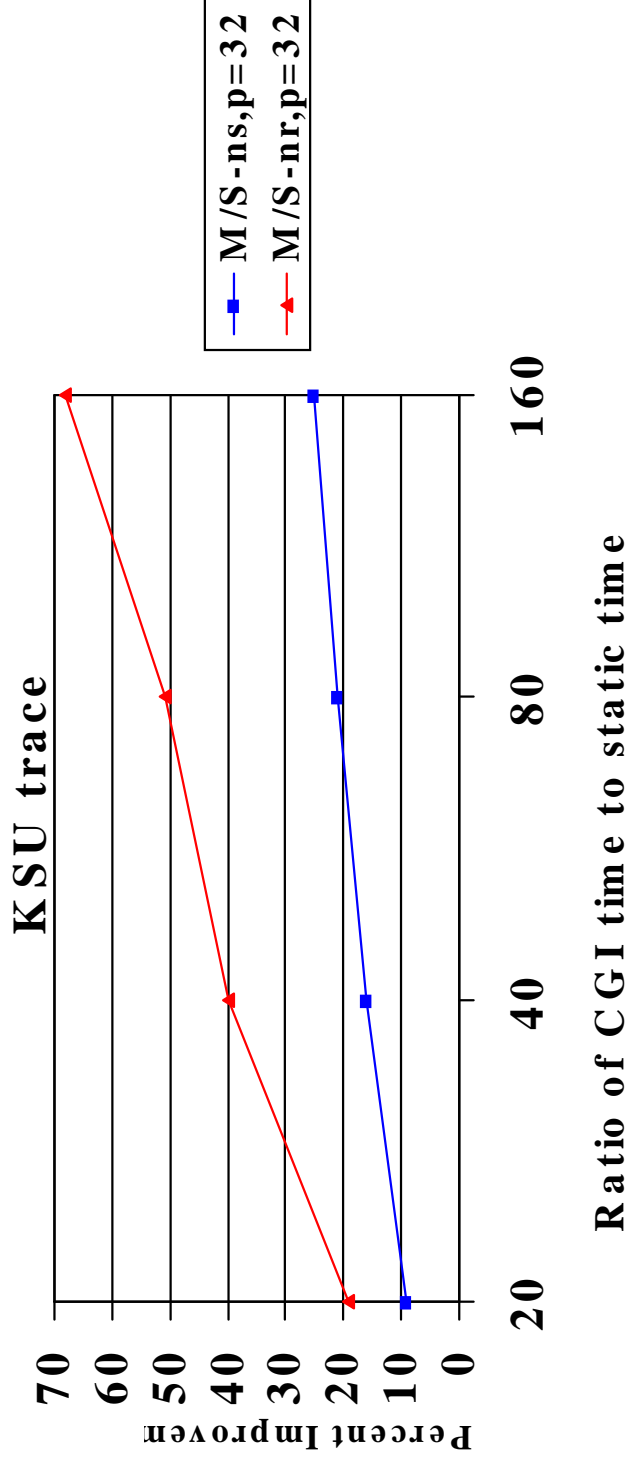
Improvement of average stretch factor by M/S over Flat-C and Flat-R.
p is the number of servers. (similar results for other traces)

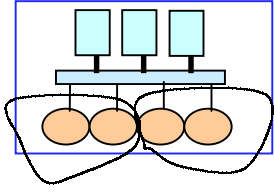




Simulation Results: Benefits of Optimization Strategies

Performance improvement of M/S over other less optimized scheduling methods (similar results for other traces)



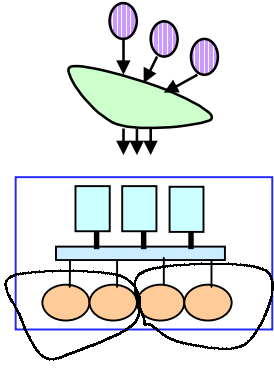


Experimental Validation on a SUN UltraSparc Cluster

- ◆ M/S implementation based on Apache 1.3
- ◆ Server cluster consists of 6 Sun UltraSparc I connected via 100Mb switched Ethernet.
- ◆ 3% variation of simulated vs. experimental results

Reduction of Stretch factors by M/S over other methods

Trace	Number of masters	M/S-ns	M/S-nr	Flat-C	Flat-R
UCB, 40/s	3	13%	18%	18%	26%
KSU, 40/s	2	18%	32%	30%	41%
ADL, 40/s	1	16%	37%	27%	35%



Concluding Remarks

- ◆ Evaluation results:
 - ◆ M/S: up to 69% performance improvement over flat
 - ◆ Separation of dynamic and static content
 - ◆ Resource reservation: up to 68% improvement
 - ◆ Resource requirement sampling: up to 23% improvement
- ◆ Contributions:
 - ◆ Designed and evaluated layered architecture for web server clusters with trace-driven simulation and experimentation
 - ◆ Proposed several scheduling optimizations
 - ◆ Results applicable to other dynamic content methods