The background features several large, overlapping, curved shapes in shades of green, purple, and light blue. Interspersed among these are numerous small, yellow, triangular shapes pointing in various directions, creating a dynamic and abstract pattern.

Selective Early Request Termination for Busy Internet Services

Jingyu Zhou and Tao Yang

Ask.com

University of California, Santa Barbara

Multi-tier Internet Services

Firewall/
Web switch

Query
caches

Index servers
(partition 1)

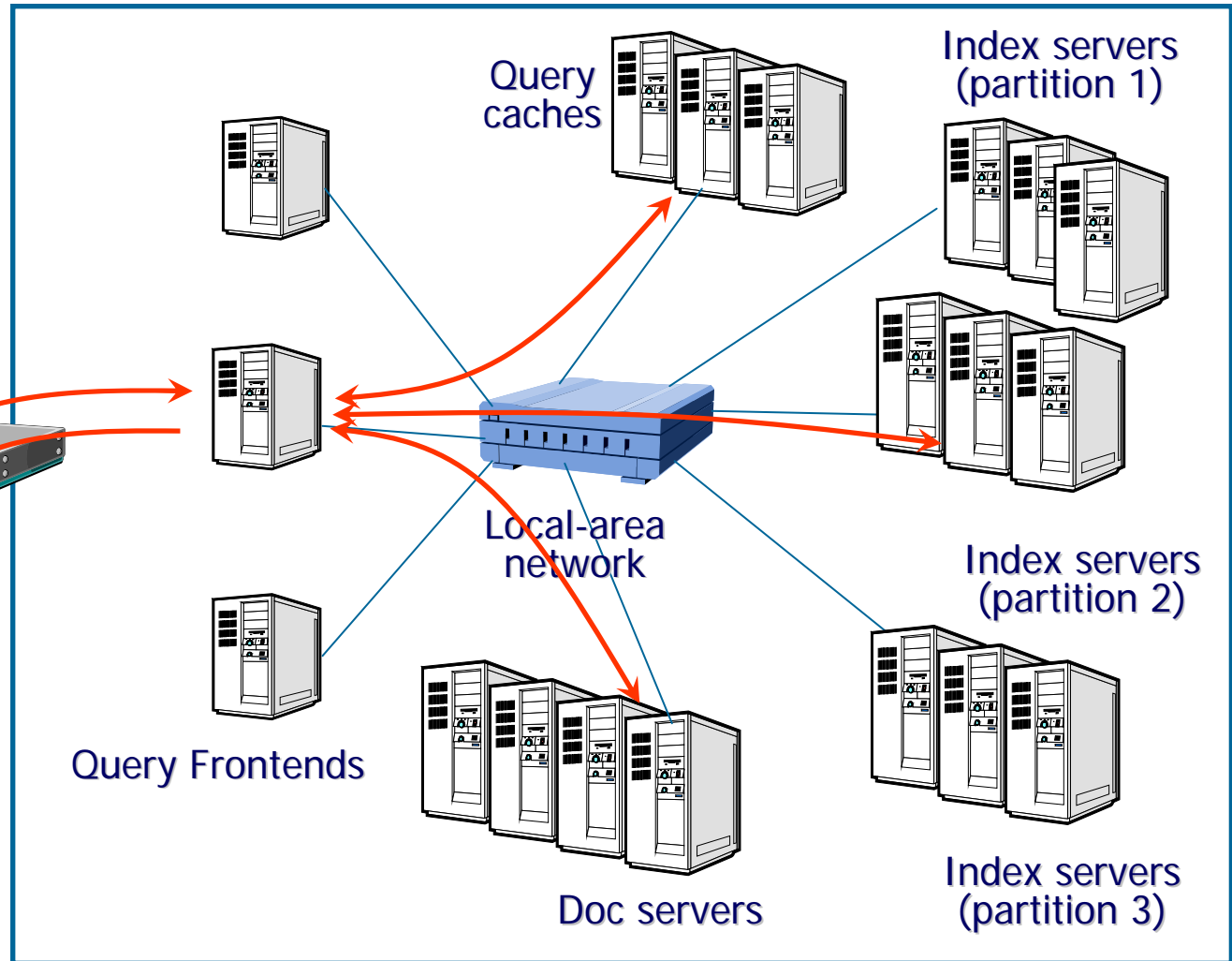
Local-area
network

Index servers
(partition 2)

Query Frontends

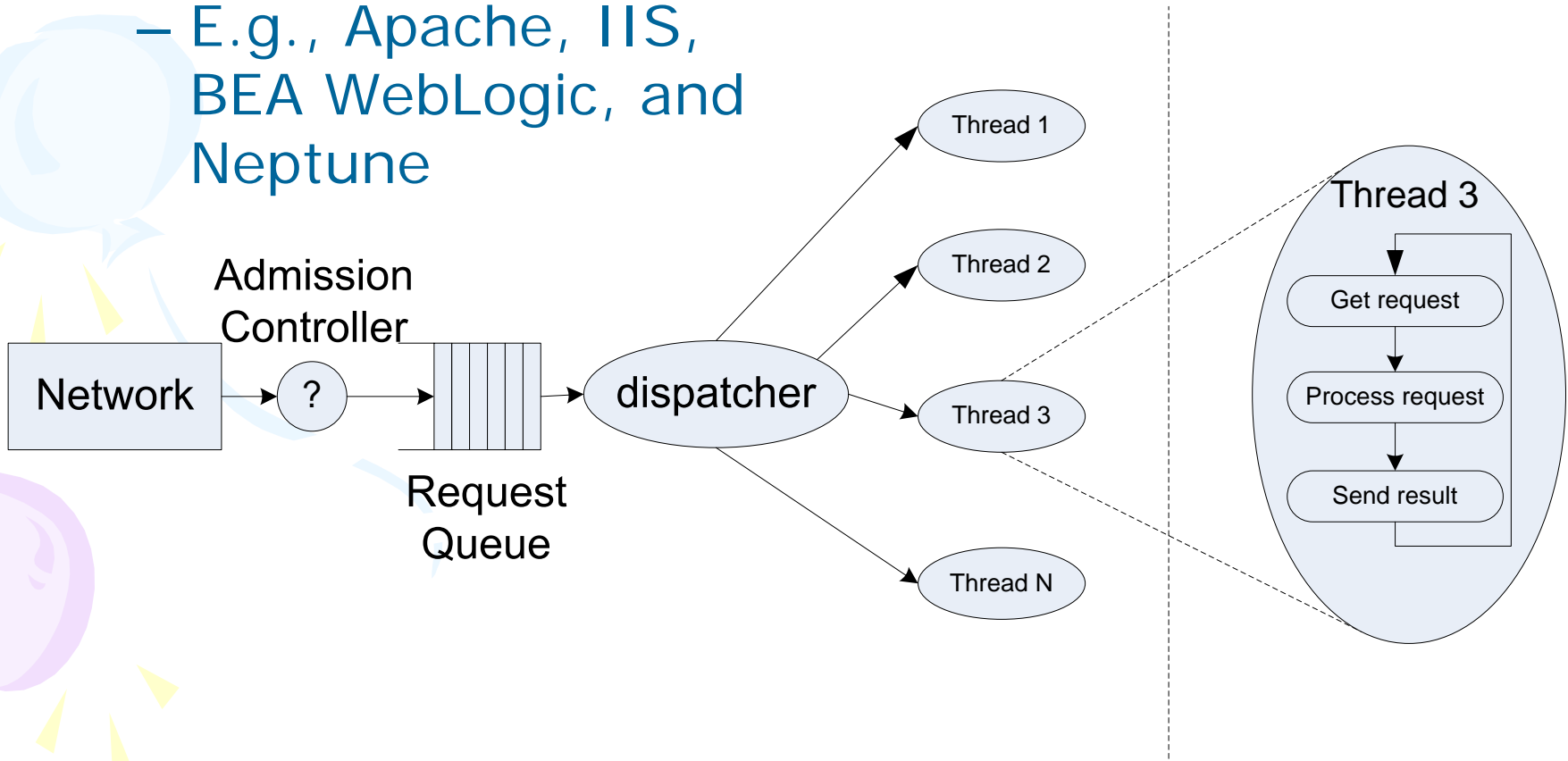
Doc servers

Index servers
(partition 3)



Multi-thread Programming Model for Request Processing

- Multi-threaded service tier
 - E.g., Apache, IIS, BEA WebLogic, and Neptune





Problem Statement

- Service-level agreement
 - E.g., 99% requests within 1s
- A QoS challenge to be met during
 - Flash-crowd type of high request rate
 - Size distribution shift: percentage of long requests increases

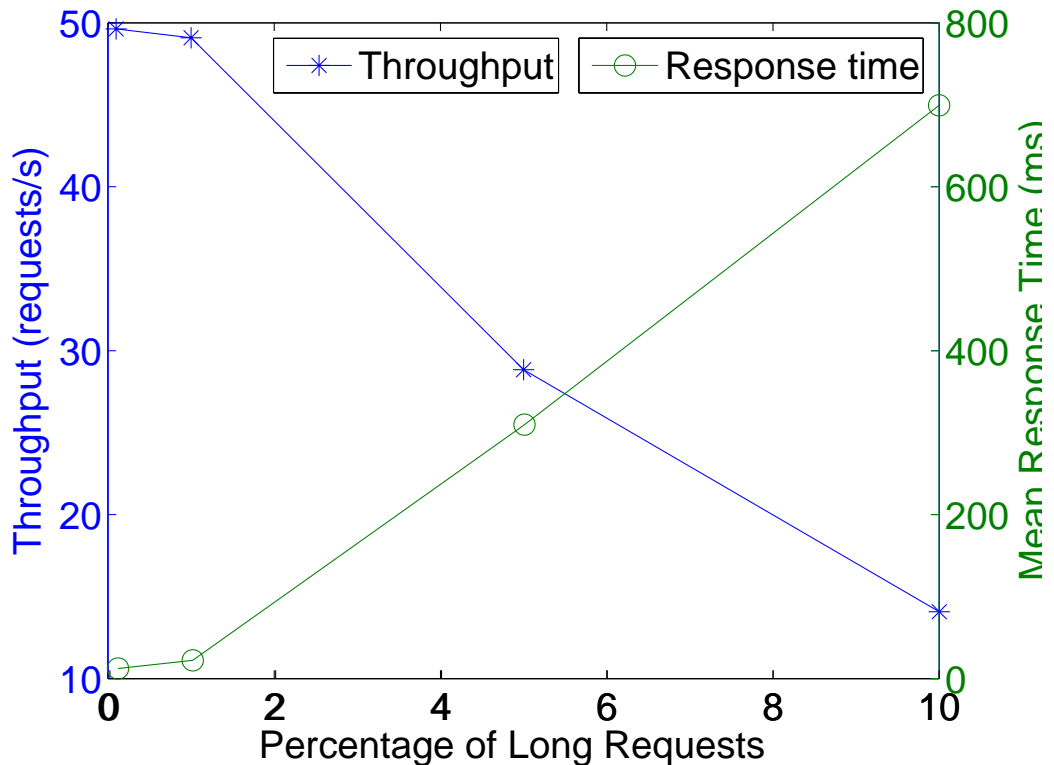
Motivating Example: Size Distribution Shift

- Settings

- 50 requests/s
- Two types of requests: 5ms and 500ms
- Long requests vary from 0.1% to 10%

- Results

- Significant throughput loss
- Magnitude increase of response time
- Admission control alone isn't enough





Current Techniques

- Admission control
 - Response time feedback (e.g., SEDA, Quorum)
 - Bounding request queue length (e.g., Neptune)
 - Policing TCP SYN packets (e.g., [Voigt'01])
- Adaptive service degradation
 - E.g., reduce image quality
- Size-based scheduling
 - Only for static content
 - File size as estimator




SERT Idea & Challenges

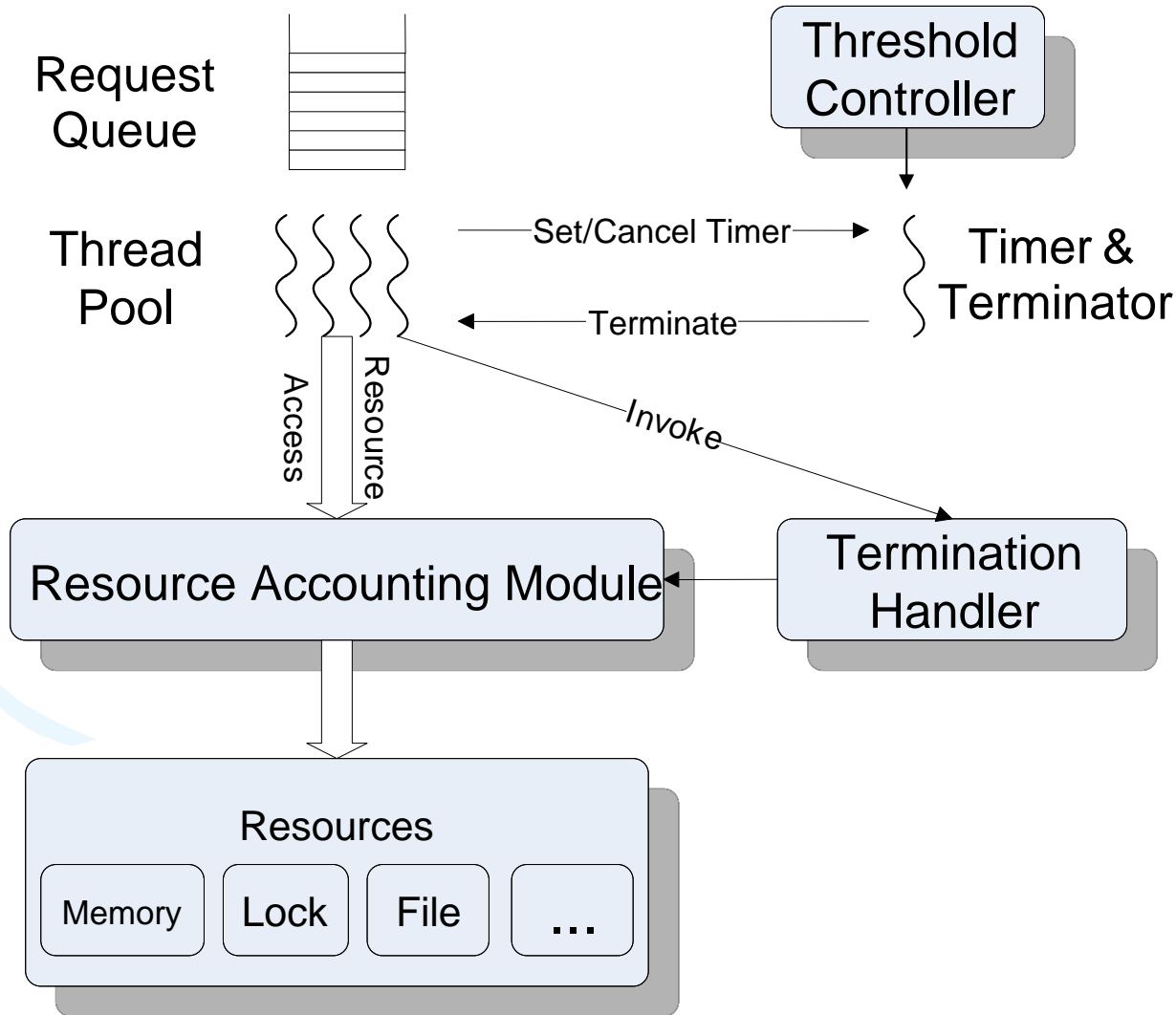
- Idea

- **Request-aware**: differentiate long and short requests
- **Early termination**: abort long requests during overload

- Challenges

- Detect long/short dynamic requests
 - Adaptive selection of termination threshold
 - Resource accounting for safety
 - Simplicity in programming
- 

SERT Architecture





Resource Accounting

- Targets a class of requests that are
 - Read-only
 - Stateless
- Resources
 - Memory: track heaps and memory mapped areas
 - Locks: use an integer counter
 - Sockets & file descriptors

Threshold Controller Adjusts Termination Threshold

- Ideas

- During light load allow execute longer: large threshold
- During heavy load terminate earlier: small threshold
- Load index p is throughput loss

- Formula

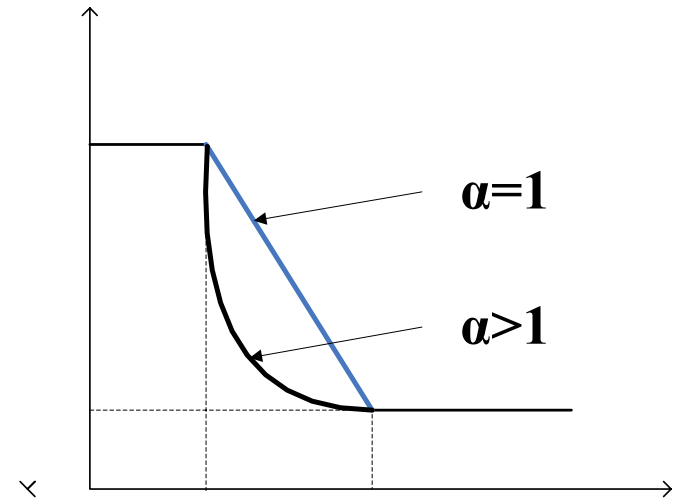
- $Threshold = LB + F(p) \times (UB - LB)$, where:
timeout range is $[LB, UB]$

$$F(p) = \begin{cases} 1 & p \leq LW \\ \left(\frac{HW - p}{HW - LW}\right)^\alpha & LW < p < HW \\ 0 & p \geq HW \end{cases}$$

$$p \leq LW$$

$$LW < p < HW$$

$$p \geq HW$$





Implementation & Usage

- Intercept Glibc/Pthread functions
 - Memory, Pthread locks, etc.
- POSIX signal for terminations
- Use sigsetjmp()/siglongjmp()
- Neptune middleware uses SERT APIs
- Applications link the SERT library with **no** code changes



SERT APIs

- Start timer thread and set signal type
extern int SERT_init_timer(int signum);
- Start & end of a request
extern void SERT_start();
extern void SERT_end();
- Set timeout value and controller parameters
extern void SERT_set_args(struct sert_arg *);
- Set the rollback point
extern void SERT_register_rollbackpoint(void *);



A Pseudo-code Example

```
void worker()
{
    while (1) {
        Request *request = get_request();
        jump_buf env;

        if (sigsetjmp(&env, 1) == 0) {
            SERT_register_rollbackpoint(&env);
        } else {
            /* longjmp back, resources has already
            been deallocated */
            continue;
        }

        SERT_start();
        process_request(request);
        SERT_end();

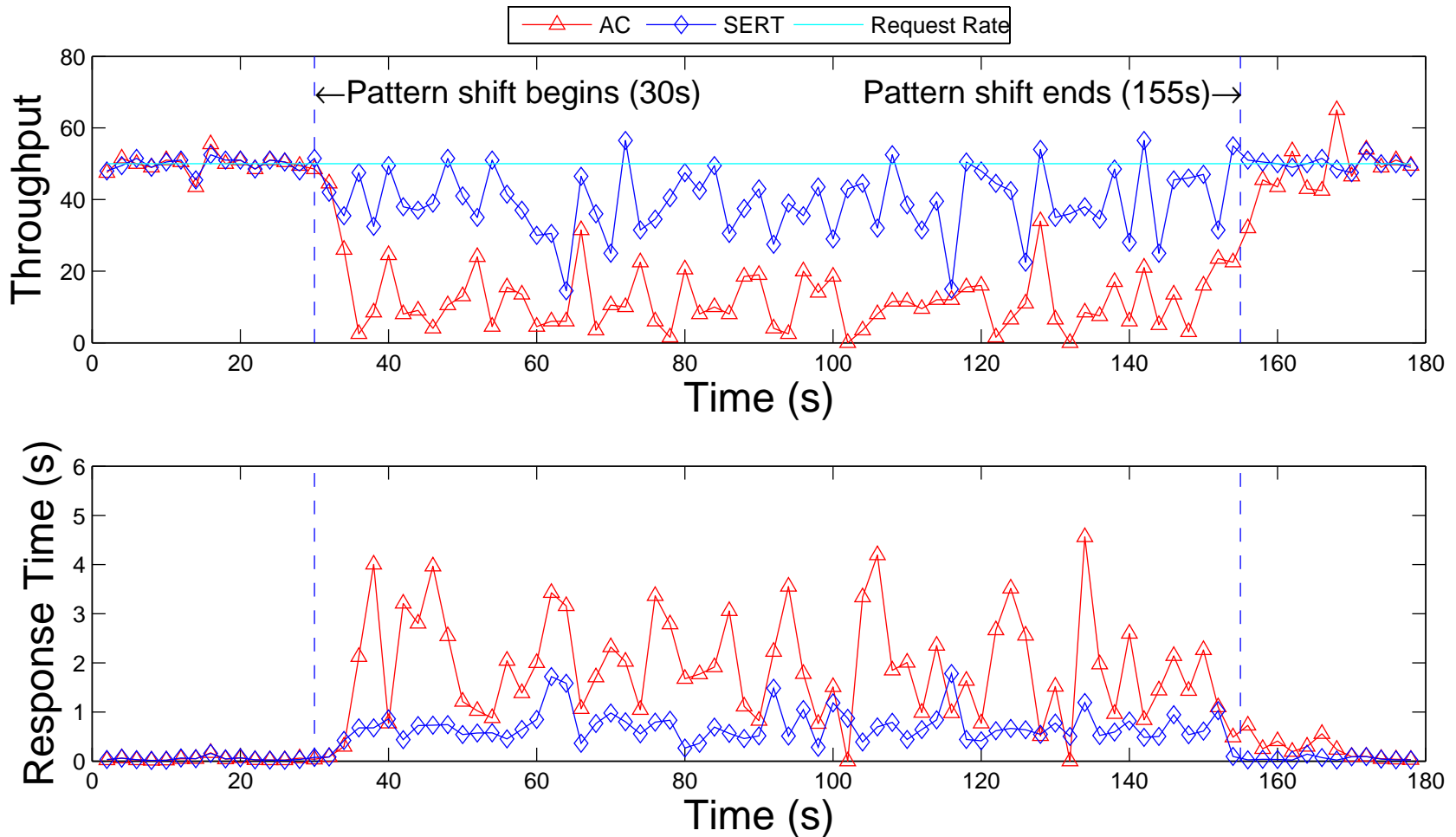
        send_result(request);
    }
}
```

Experimental Settings

- Hardware
 - 9 dual PIII 1.4GHz machines
 - Each has 4 GB RAM, 10K RPM SCSI disk
 - Fast Ethernet
- Applications from Ask.com
 - Index matching: find web pages containing key words; heavy-tailed; 2.1 GB warm data in memory
 - Ranking: rank page importance; exponential; in memory

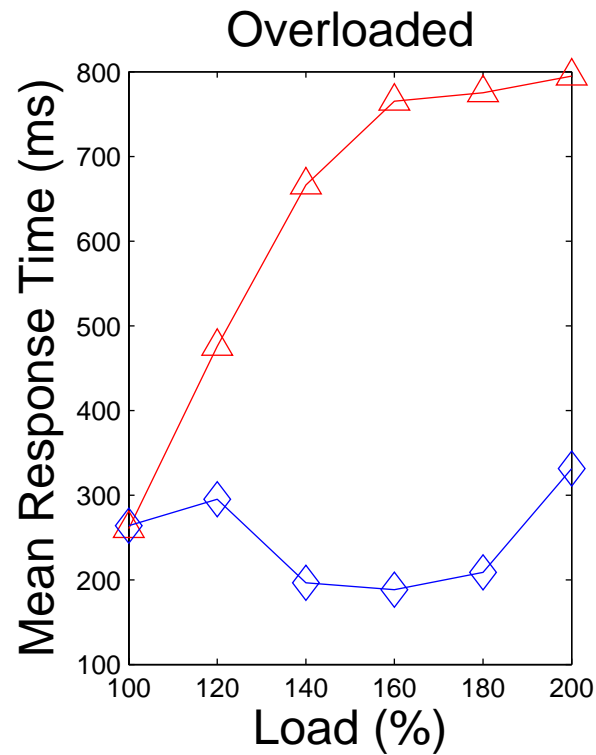
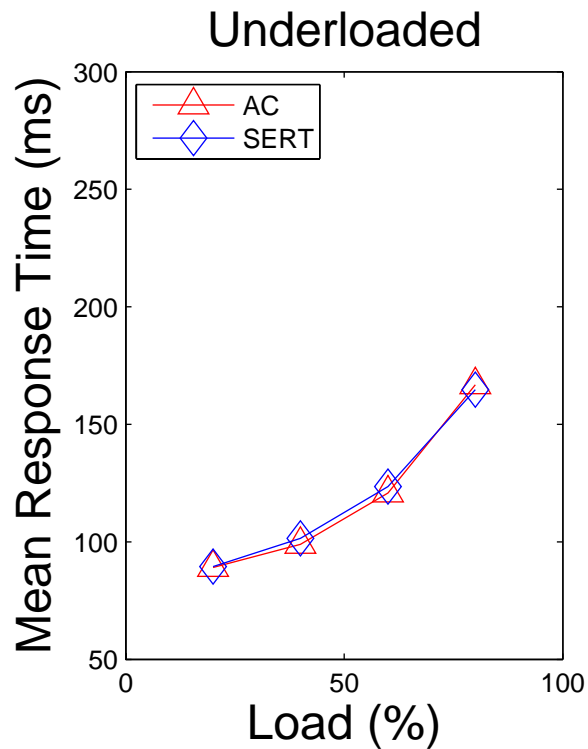
App.	Ave. (ms)	90% (ms)	Max. (ms)
Index Match	23.6	46	2,732
Ranking	93	212	14,035

Size Distribution Shift

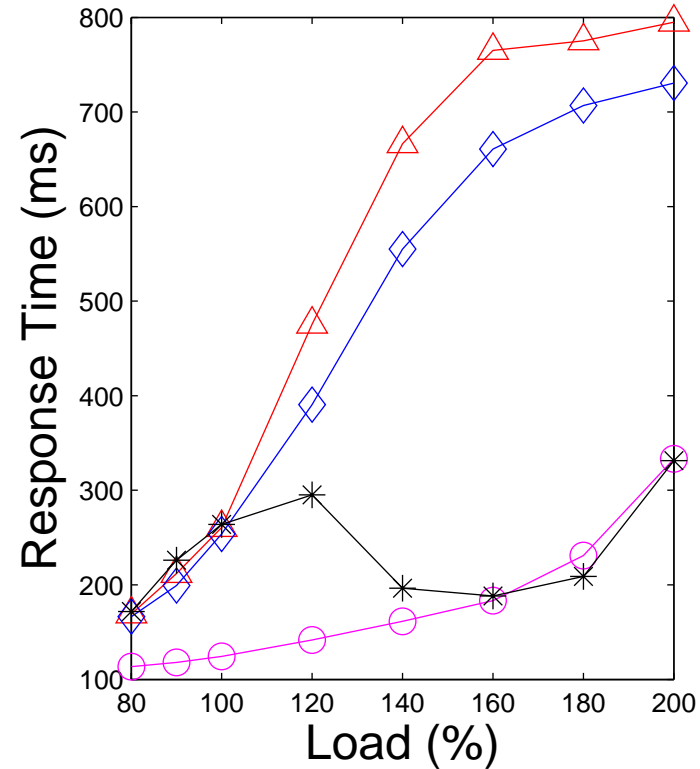
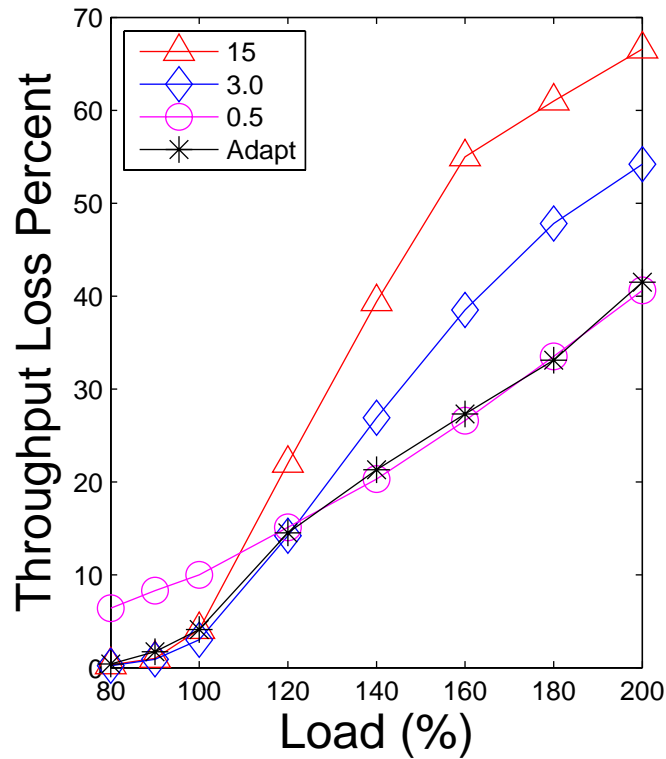


- During shift, about 10% requests are 500+ms
- SERT
 - 209.1% higher throughput
 - 54.7% response time reduction

Ranking Service Evaluation

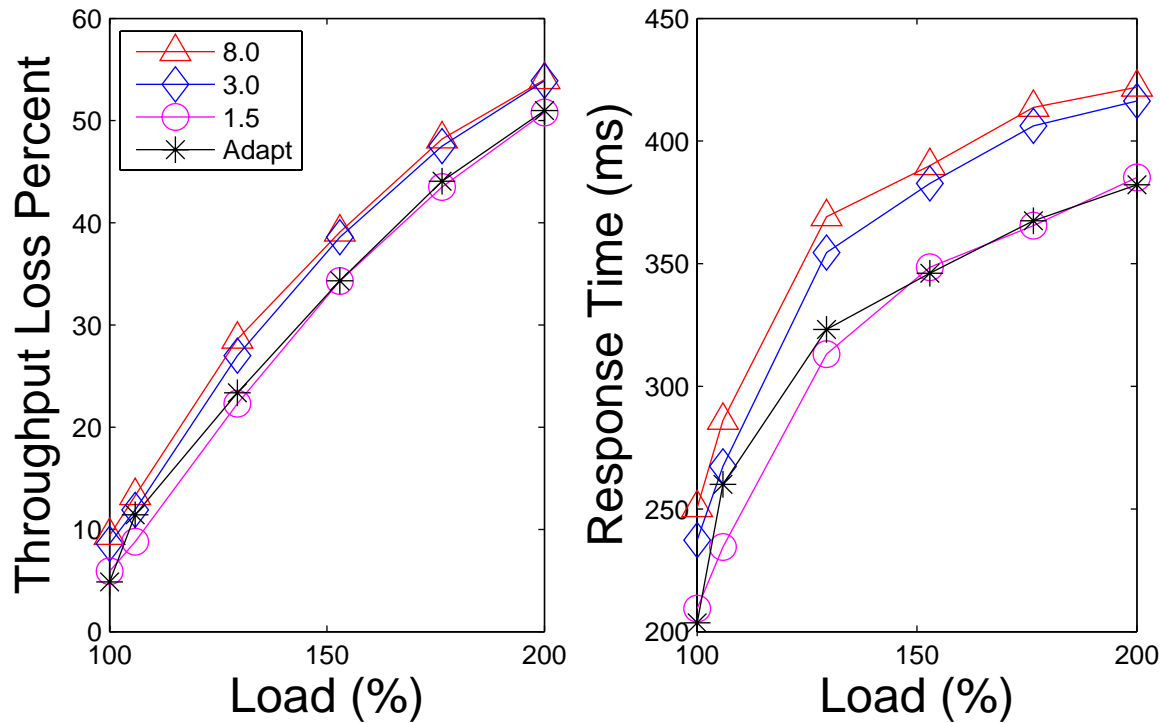


Evaluation of Threshold Controller for Ranking Service



- Adaptive controller vs. fixed threshold of 0.5s, 3.0s, 15s

Evaluation of Threshold Controller for Index Matching



- Adaptive controller vs. fixed threshold of 0.5s, 3.0s, 15s



Related Work

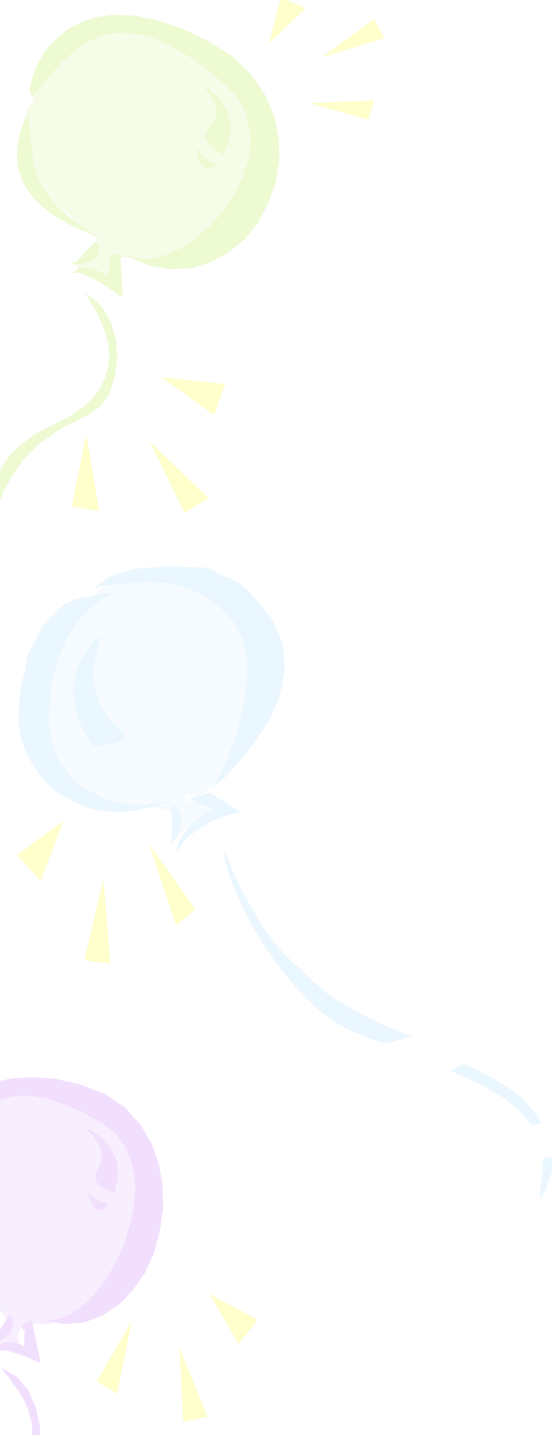
- Real-time database systems [Kuo'00, Lin'90, Shu'94]
 - Higher priority transaction aborts lower ones
 - UNDO/REDO log for recovery
- Recoverable memory libraries
 - Recoverable virtual memory [Saty.'94], Rio Vista [Lowell'97]
 - Application modifications needed
- Process checkpointing and rollback
 - Fault tolerance [Li'90], program replay [Srinivasan'04] and debugging [Qin'05]



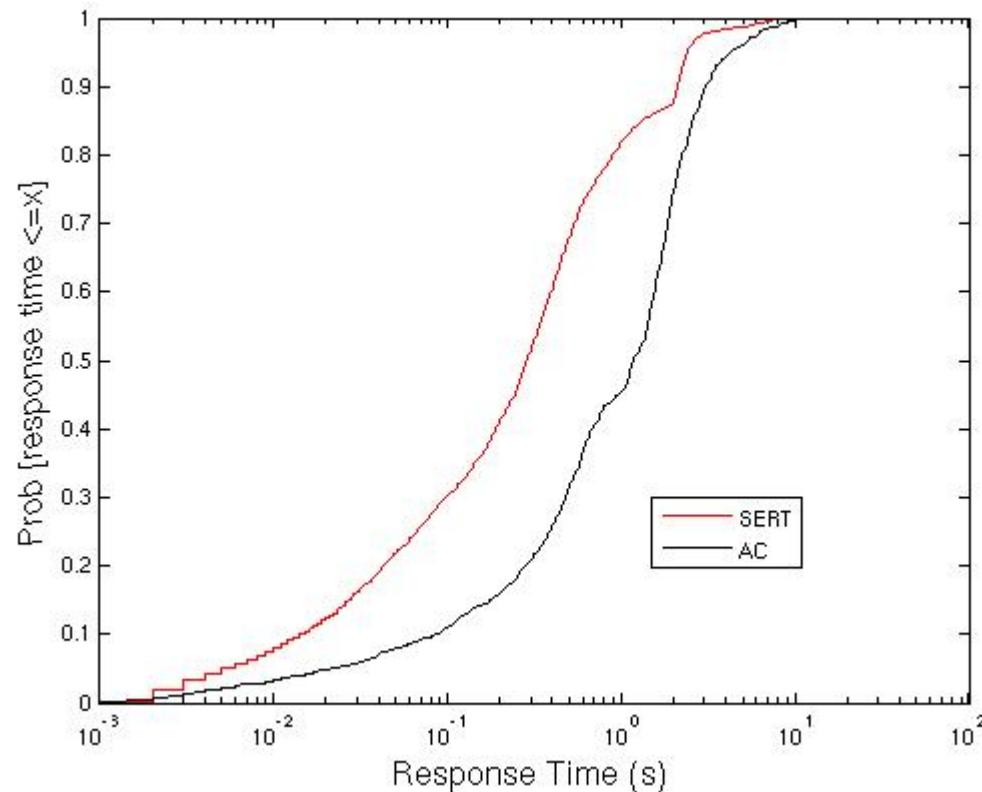
Conclusions

- Contribution: an early termination scheme for busy Internet services
 - Dynamically select termination threshold
 - Safely terminate requests early
 - Provide API for multi-threaded services
- Future work
 - Perform cooperative early-termination across different nodes and tiers

Questions?



CDF of Response Time during Size Distribution Shift



E.g., completed within one second

- SERT 81.7%
- AC 45.3%