Hierarchical cluster for scalable web servers

Presented at Cluster 2001, Newport Beach, CA., October 9, 2001

Dongseung Kim & Jonghuck Hong

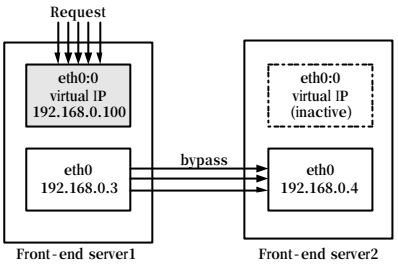
Korea University

Motivation

- Web servers & scalability
  - cluster: easy expansion
  - content based server switching
  - easy job migration
- High efficiency and throughput of web servers
  - High throughput requirement for peak load time
  - Job dispatching bottleneck in the dispatcher when the arrival is burst

Expansion & performance scalability

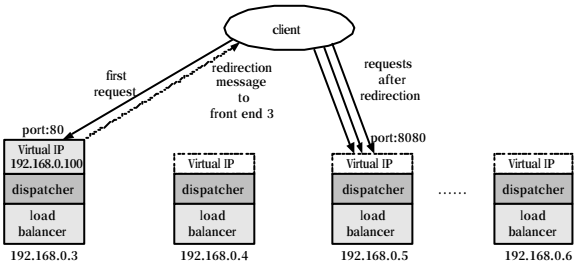
- Expansion by front-end nodes
  - multiple front-end node for dispatching
  - load distribution among front-end nodes
    - 1. Bypass



    - 2. Redirection

HTTP redirection-Front-end nodes

- One virtual IP – transparent to outside users
- HTTP redirection - expandablity
  - First contact thru virtual IP port no. 80
  - second and later connection: port 8080
  - connection consistency - designated servers
  - Two connections needed

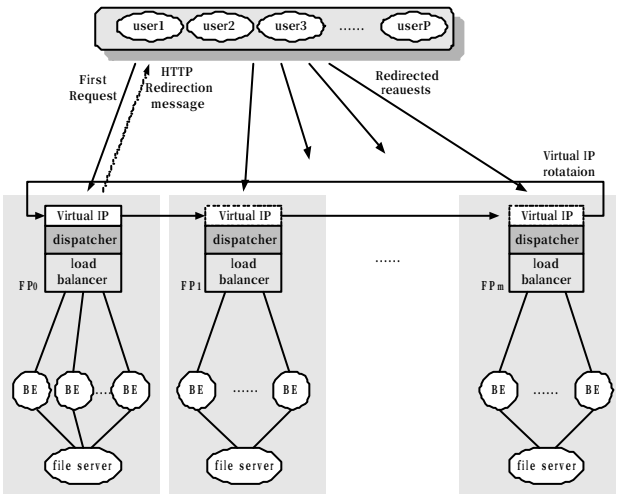


Load balancing / distribution

- Imbalance among front-end nodes
  - weighted round-robin scheduling using HTTP redirection
- Among backend nodes
  - Weighed round robin scheduling
    - load index: average file size transmitted
    - Normalized load index used for weight computation

5

Hierarchical Architecture



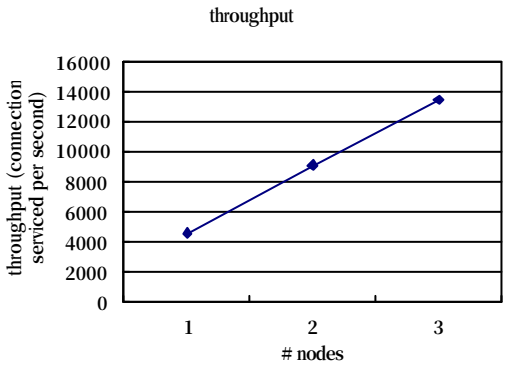
6

Experiments

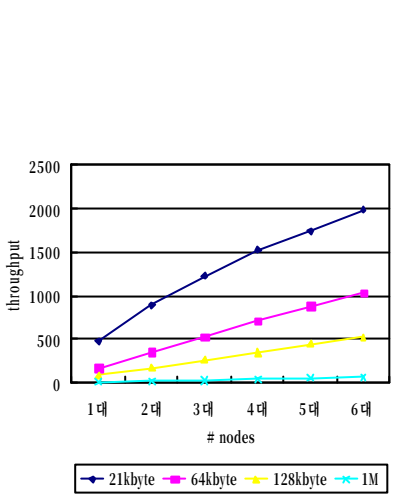
- Hardware & software
  - Cluster of 16 Linux PCs
  - Pentium II CPU with 128 MB memory
  - 100Mbps Ethernet switch
  - Partition to both servers and clients
  - Documents with various file sizes

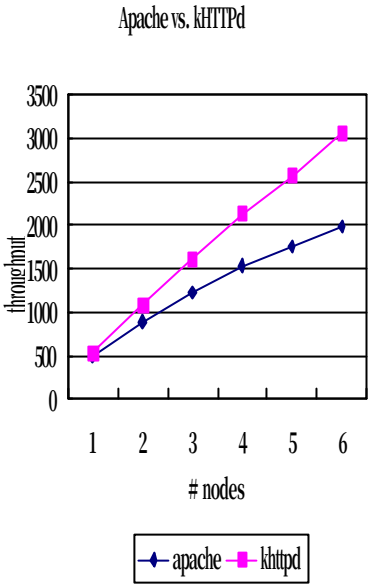
Scalability of front end nodes

- Configuration
  - 1-3 front end nodes
  - 6 backend
  - 6 client no



Scalability in backend nodes



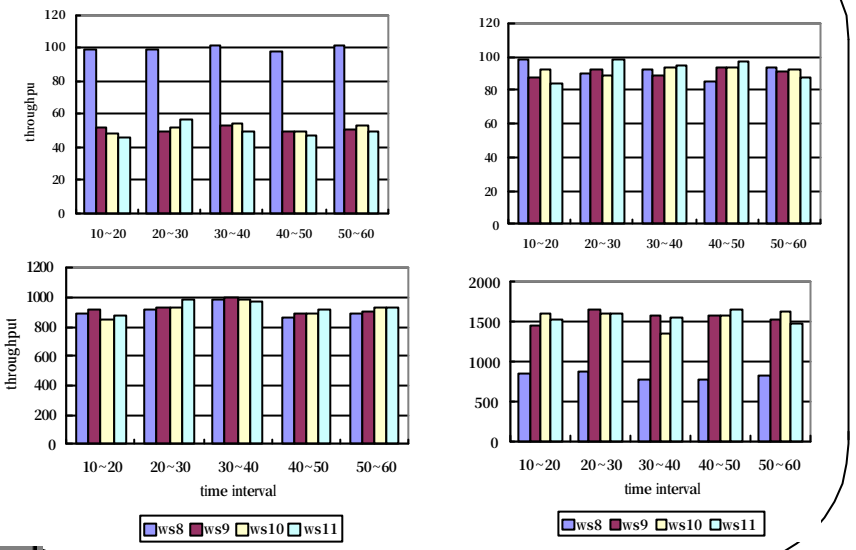


Load balancing in backend nodes

- configuration
  - Four client nodes
  - One front end dispatcher
  - Four web server nodes
- Load level
  - Full-loaded by client requests
- Document sizes
  - 16k, 21k, 64k, 128k, 256k
- Forced imbalance if Round Robin scheduling is used
  - 2:1:1:1
  - 8:1:1:1

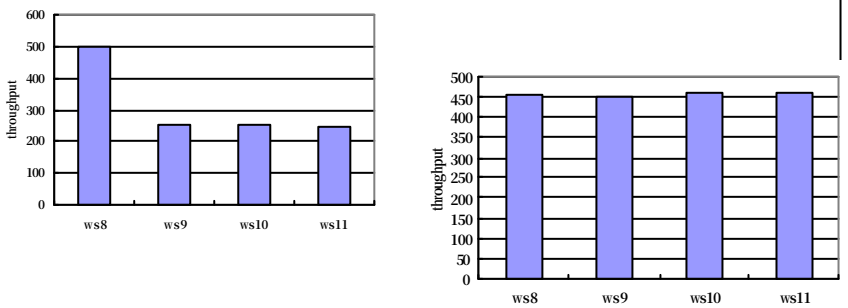
Load distribution - 2:1:1:1

- RR scheduling
- Dynamic scheduling



Load distribution - 2:1:1:1

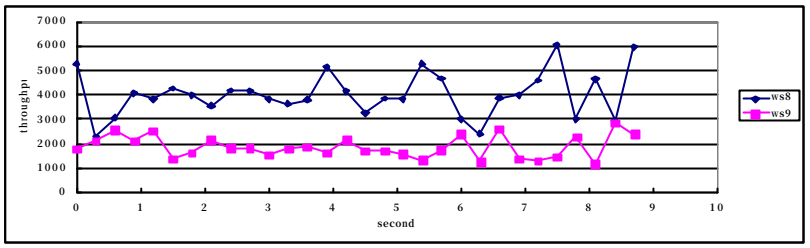
- RR
- Dynamic load balancing



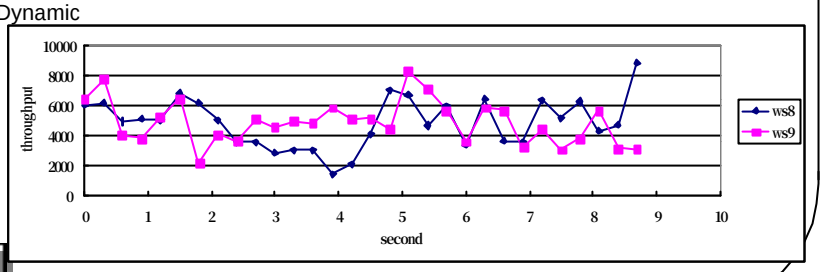
Snapshot of load variation - 2:1:1:1

(load measured in every 0.3 second)

- RR



- Dynamic

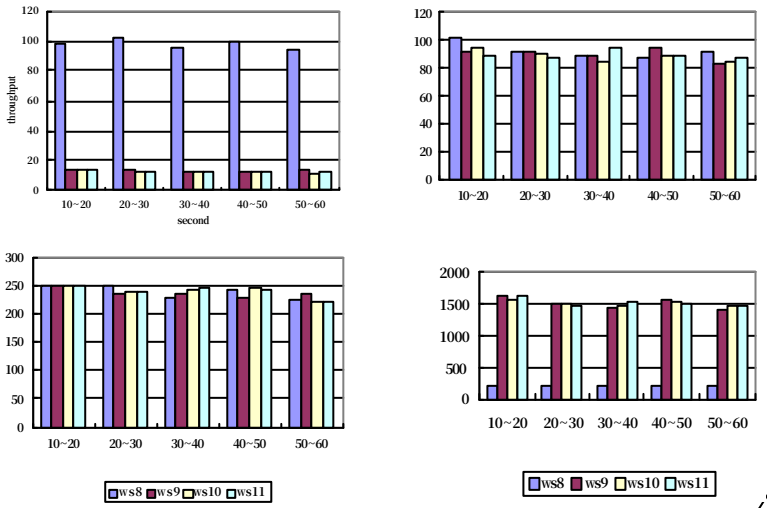


13

Parallel Computing & Architecture Lab. Korea Univ.

8:1:1:1

- RR
- Dynamic scheduling

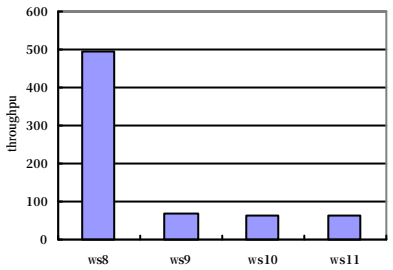


14

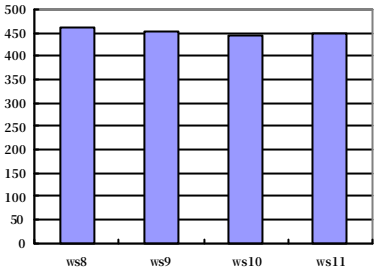
Parallel Computing & Architecture Lab. Korea Univ.

- Throughput with static load of 8:1:1:1

- RR



- Dynamic scheduling



15

Parallel Computing & Architecture Lab. Korea Univ.

Conclusions

- Multiple front-end nodes for concurrent job dispatching
  - hierarchical architecture for content-based switching
  - Bottleneck in dispatcher removed by using multiple front end nodes with HTTP redirection
- Redirection in back-end server nodes
  - Weighted Round-robin scheduling
  - Load index- number of average sized file served

16

Parallel Computing & Architecture Lab. Korea Univ.