Web Servers Outline

Chris Chin, Gregory Seidman, Denise Tso

March 19, 2001

I. Introduction

   A. What is a web server?
      1. is it anything that can be retrieved with an URL?
      2. (web service architecture diagram)
      3. web services (URL protocols)
         a. FTP
         b. gopher
         c. RTSP
         d. HTTP
      4. why concentrate on HTTP servers?
         a. HTTP is the predominant protocol on the web
         b. meta-information plus simplicity make it ideal
      5. compare and contrast: file servers vs. HTTP servers
         a. file servers are read/write
         b. HTTP servers are read-only
   B. Why do we want HTTP servers?
      1. standard, cross-platform, multimedia content delivery system
      2. enables the web as a universal information system
      3. everybody else is doing it
   C. What makes running a web server hard?
      1. maximize performance given available resources
      2. tune for varying definitions of "performance"
      3. end-to-end security concerns
   D. Contents overview
      1. basics of HTTP serving
      2. optimizing for performance and what we mean by performance
      3. price/performance tradeoffs
      4. security in web services
      5. case studies of web serving

II. HTTP Server Basics

   A. HTTP is an application-level protocol on top of TCP
   B. TCP is a connection-based, end-to-end Internet network protocol
   C. Simplest version: respond to HTTP requests with files on disk
      1. accept connection
      2. read request path
      3. translate requested path into file path
      4. read file from disk and write to network
      5. close connection
   D. More complicated versions involve
      1. generating responses dynamically
      2. caching responses in memory
      3. authentication and encryption
      4. multiple requests per connection (HTTP 1.1)

III. Performance Optimization

   A. Performance goals
      1. maximum reqs/second
      2. maximum throughput (bps)
      3. minimum latency (roundtrip time, a.k.a. RTT)
      4. minimum error rate (dropped/rejected connections)
      5. minimum downtime
      6. minimum cost ($)
   B. Limited resources
      1. network bandwidth
      2. disk bandwidth (and latency)
      3. memory capacity
      4. processor cycles
      5. queue length/load limit/thread limit
   C. Benchmarking and load testing
      1. GStone
      2. SPECweb99
      3. TPC-W
      4. WebBench
      5. WebStone
      6. Mercury Interactive
   D. Post-mortem analysis
      1. log analysis

        2. trace analysis
    E. Choosing a configuration
        1. performance studies
        2. experimental analysis
        3. symptoms of overload

IV. Tradeoffs Between a Single Server, Clustered Servers, and Big Iron

    A. The ideal web server...
        1. ...does not exist!
        2. depends on optimization goals
    B. Price/performance tradeoffs
        1. basic server system
            a. low cost
            b. single processor
            c. one or more disks
            d. large memory capacity
            e. single network interface
        2. clustering
            a. several single-processor servers appear to the outside world as a single server
                - mirroring
                - task factoring
                - data partitioning
                - (ref case studies)
            b. advantages
                - moderately low cost
                - more disk bandwidth
                - more memory capacity
                - more processing power
                - sometimes support failover
            c. disadvantages
                - may not survive failed elements
                - load may not be balanced
                - difficult to maintain a consistent overall state
            d. load balancing
                - fair and efficient allocation of workload to clustered elements
                - hard problem
                - depends on standards for inter-node communication

3. big iron
    a. single machine with multiple processors and disks
    b. advantages
        - more disk bandwidth
        - more memory capacity (?)
        - more processing power
    c. disadvantages
        - high cost
        - rarely survives failed elements

V. Security

A. Why bother?
    1. some information is sensitive
    2. only designated users should have access
    3. listening on the network should be difficult or impossible

B. Authorization and encryption
    1. password protection
    2. HTTPS and SSL

C. Firewalling sensitive data

D. Virtual private networks (VPNs) and extranets

VI. Web Server Software Architecture

A. Apache as a running example

B. Other server software should be mentioned

C. Software design choices

D. Algorithms
    1. scheduling/prioritizing requests
    2. thread pools

E. Integrated dynamic generation (mainly reference dyncon chapter)

VII. Case Studies

A. E-bay
    1. description
        a. premiere site for online auctions
    2. focus/motivation
        a. clustered servers and database backend
        b. heavily loaded 24/7
        c. interesting dynamic content and coherency problems
    3. details

        a. optimized for uptime
        b. coherency and freshness suffer
   B. HotMail
      1. description
         a. premiere web-based email service
      2. focus/motivation
         a. clustered servers
      3. details
         a. potentially heavy loads
         b. difficult to balance due to data transport

VIII. Advanced Topics

   A. Performance optimization is application-specific
      1. *dynamic content* generation
      2. web site *customization*
      3. *search engines*
   B. Interaction with *proxy caches*
      1. accurate expiration timestamps and cache coherency
      2. accurate page view statistics despite caches

IX. Summary/Conclusion

   A. Web services are currently important
   B. Hardware and software
      1. ...must be chosen carefully
      2. with an understanding of performance goals, good choices can be made
   C. Security
      1. sensitive information can be protected
   D. ...conclusion stuff based on previous prose