IT441: Network Servers & Infrastructure

CLASS 2: 12 Sep 2005
13:30 - 16:15

Resources: TAs

Prince William Campus:
Lunahsh Siddhu
Lunahsh@vgmuc.edu

check the course website http://netlab.vgmuc.edu/IT441 for updates

Last Time:

- NEW and how to get set up to use it
  - http://netlab.vgmuc.edu/tools
- Intro stuff
- Prereqs, grading, exams
- Servers
  - Definition of a server
  - Computers, characteristics that can affect servers
  - OS features
  - E.g. virtual memory, processes, and threads

This Time:

- OS features
  - Files, filesystems
  - Network interfaces
  - Server operation
  - Seen from perspective of performance
  - Kinds of data served
  - And how this affects performance

Life of a Process

Example: Different JVMs Handle Blocking
Utilization vs. Load

Thrashing

Using Disks

Using Disks

Using Disks

Using Disks
Optical Disks

- delays for CD-ROM/CDRW optical disks are same as for magnetic disks, but worse:
  - rotational delay: disk spins more slowly and
  - change rotation rate: varies from 400 to 200 RPM (nominal, for audio CD)
  - seeks are slower (typ. average 100-120 ms)
  - another possible delay: focus time once head “on station”
- and, for DVD:
  - layer change time

Using Disks

- to get information from a disk (reading):
  - info must previously have been put on the disk
  - specify start location: <cyl, sector, head>
  - specify length: number of sectors
- writing to a disk is analogous:
  - should the disk process requests in the order in which they are received?

<table>
<thead>
<tr>
<th>Request Number</th>
<th>Cylinder</th>
<th>Sector</th>
<th>Head</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0001</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>n+1</td>
<td>26</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>n+2</td>
<td>0003</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

- if processed in the order presented:
- takes 26 ms = data transfer time
- but, if requests need, would only take: 13 ms

Disk Errors

- most disks today use error correction techniques to
  - increase immunity to errors
  - allow higher density recording = higher capacities
- how frequently do errors occur?
  - non-recoverable data error: 1 in 10^11 bits read
  - sense error rate: 1 in 10^7 seek
  - MTBF: 1.2 x 10^6 hours
  - minimum start/stop cycles: 50,000
  - component life: 5 years

Managing Files on Disks

- file space typically distinct from space used for paging/virtual memory support
- file system manages files on disks
- what does file system have to do?
Managing Files on Disks

- File space typically distinct from space used for paging/virtual memory support
- File system manages files on disks

- What does file system have to do?
  1. Manage used blocks on disk (those contain files)
  2. Manage unused blocks on disk (those contain nothing)

An early linked-list scheme

<table>
<thead>
<tr>
<th>Filename</th>
<th>Link 1</th>
<th>Link 2</th>
<th>Data</th>
<th>Link 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>File 1</td>
<td>4</td>
<td>8</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>File 2</td>
<td>5</td>
<td>9</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>File 3</td>
<td>6</td>
<td>10</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Data</td>
<td>7</td>
<td>11</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

filename.txt is block 6 \rightarrow 8 \rightarrow 4 \rightarrow 2

Indexed File Management (UNIX)

- Instead of information per block on the disk (as linked list schemes), keep information per file
- What do you not see in the inode?

Indexed File Management (UNIX)

- File is not there
- A UNIX directory is a plain file with entries of form:
  - Inode, name, file type
- Is marked with attribute "directory"
Indexed File Management (UNIX)

- how many file bytes accessible in:
  - 1 seek: 30 direct access blocks from inode: 10 Kb
  - 2 seeks: 29 blocks: 256 Kb more
  - 3 seeks: 296 blocks: 65 Mb more
  - 4 seeks: 296 blocks: 16 Gb more
  assuming 3 Kb blocks

NTFS

- volume layout (partial)
  - Master File Table: one entry per file in volume
  - "system files"

Not Enough

- what do you do when you run out of disk space?
  - get more disk space
  - but which is better?
  - one more bigger capacity, more expensive disk?
  - one or more additional smaller disks that can be used collectively to appear as a bigger disk?

RAID Disk Drives

- disks may be fully independent within array
  - rotationally independent
  - access arms move independently
  - good for independent multiple I/O requests
  - or may be synchronized
  - spindles synchronized
  - access arms move together
  - good for high I/O speed

The server process

1. read the menu, make choice
2. waiter takes order
3. kitchen does magic
4. eat

The server process

1. read the menu, make choice
2. waiter takes order
3. kitchen does magic
4. eat
The server process

- read the menu, make choice
- waiter takes order
- kitchen does magic
- variable cost
- eat

The (computer) server process

- as with restaurant:
  - some costs are relatively fixed, others highly variable
  - client only cares about total service time
  - how to measure 'cost' for computer process:

- how to measure 'cost' for computer process:
  1. how much time to run process
  2. how much space the process needs

Typically are mutually exclusive: tradeoffs
- "hard" to optimize both simultaneously

A Server's view of Requests

- server receives request:
  - from another process on same machine
  - standard format for request
  - how defined by service protocol
  - usually implies fixed cost for processing
- clock starts ticking at server
  - but for client, clock started longer ago
A Server's view of Requests

1. process request and determine how to resolve
2. generate response
   - e.g., filesystem lookup of static file
   - e.g., "sub-contract" and ask another process to handle
     may have to re-process request
   - may have to format individual results
   - e.g., computation
     as in dynamic web pages, e.g., filling in date
3. deliver response
   - depending on how it was resolved, server may have to work
to deliver result
4. post-processing
   - garbage collection, reclaim and ready for re-use resources
     used to service request
   - accounting, update records, log files, etc.

Other Factors Affecting Service Time?

- scheduled downtime
- unscheduled downtime
- hardware failure
- software failure
- human error

What Servers Serve Up

- information service:
  - DHCP server provides IP address and related info
  - NTP server provides time correction time info
- content service:
  - web server provides access to content, text, audio, video, etc.
Kinds of Content

- Different content types differ in:
  - Representation format
  - How stored and accessed
  - Size, complexity
  - Compressability
- Questions:
  - How is content represented and stored on server?
  - How is it accessed and manipulated/preprocessed?
  - How is it represented for transmission?

Character Data

- ASCII became dominant coding scheme for US English language
  - Defines 128 characters — 7 bits per character
  - Unsuited to languages needing diacritical marks on glyphs, e.g., ā, ē, or special glyphs, e.g., ō.
  - Use other 128 positions available in 8-bit byte:
    - ISO 8859

Character Data

- Problems remain:
  - Some languages disallow these glyphs
  - Some languages need more than 256 glyphs
- Introduce 16-bit character representations: UNICODE

Character Data

- Irrespective of character set, these representations provide no formatting of text:
  - Other than whitespace and line break
  - Some text needs to be represented in a formatted form:
    - Adobe’s Portable Document Format (PDF) current leader
    - PDF documents unreadable without extensive software intervention; regular ASCII files easily readable

Numeric Data

- Two kinds:
  - Integer:
    - Can be represented exactly up to limit of representation size
    - E.g., byte: 0-255 signed: -128...0...127
    - E.g., 16-bit 0-65535 signed: -32768...0...32767
  - Floating point:
    - Can be represented highly accurately but not exactly
    - Usually not less than 32-bit representation
  - Each requires use of software to produce printable (human-readable) version of numeric value

Still Image

- A picture (photo) is represented by picture elements (pixels):
  - Image is digitized ("scanned") and represented as a matrix of pixels
  - Pictures may be represented as:
    - Monochrome: black and white
    - Grayscale: range of shades of gray from solid black to solid white
    - Colour: full range of colours
  - Depending on "pixel depth"
    - Number of bits per pixel
Still Image

- A picture (photo) is represented by picture elements (pixels)
  - Image is digitized ("scanned") and represented as a matrix of pixels
- Each pixel may represent:
  - Whether this point in picture is white or black
  - Usually thresholded to intermediate intensities to black or white
  - = Monochrome (1 bit per pixel)

Monochrome Image

Grayscale Image

Colour Image

- Each pixel may represent:
  - Amount of brightness in image at that point
  - Between full black (0) and full white (255)
  - Use 1 byte per pixel to give 256 levels of gray
  - = grayscale image

Still Image

- Each pixel may represent:
  - Amount of brightness in image at that point for each of red, green and blue
  - Between no RGB or full RGB
  - Can use 2 bytes per pixel to give 65536 colour = intensity values
    - 4 = 5-4 bytes
    - 1 = 5-6 bytes
  - Or 1 byte per colour per pixel to give 256 levels of each of R, G, and B = 3 bytes per pixel for 16,777,216 colour = intensity values
  - = Colour image
Image Sizes

- Quality of image representation depends on how many pixels per row, number of rows used to represent image = resolution.
- Typical computer screen: 100 pixels per inch.
- High-quality photo print: > 1000 pixels per inch.
- Scan: 1 square inch image:
  - 300 dpi grayscale = 10 Kb; colour = 30 Kb.
  - 1000 dpi grayscale = 100 Kb; colour = 300 Kb.

Moving Pictures: Video

- Cinema film motion picture:
  - 24 frames per second.
- Television:
  - 30 frames per second (NTSC).
- So, at 1000 dpi per image in colour X 24 frames per second = 72 Mb/sec.
  - Actually would be higher: why?

Audio

- Digitize audio signal by measuring loudness s times per second.
- Nyquist theorem says that s samples per second captures frequencies in analog signal up to $\frac{s}{2}$ Hz.
  - Human hearing range: 20 - 20,000 Hz.
  - E.g., audio CD sampled at 44,100 samples/second.
  - Captures up to 22,050 Hz.
- Accuracy depends on number of bits used to measure loudness in each sample.
  - Need at least 16 bit high-quality audio.

Audio

- Data rate for standard audio CD:
  $\frac{44,100 \text{ samples/second}}{16 \text{ bits/sample}} \times 2 \text{ channels} = \frac{1411200 \text{ bits/second}}{\text{second}}$
  - Or: 176,400 bytes/second for stereo.

Audio

- Alternative sampling: measure change in s and represent it as
  - 1: increased since last measurement.
  - 0: decreased since last measurement.
- Need to measure much more often, but each measurement produces only 1 bit.
- This technique called delta modulation.
- Previous technique (Nyquist) called PCM.

Data Representation

- Tradeoff:
  - High-quality representation of images/sound requires very high bit rates.
  - Users want high-quality but don’t want high bit rates (too slow).
- Solution: compression.
  - Image and audio data contains redundant information that can be discarded.
  - We only represent what’s leftover.
  - Experts group study how much can be discarded and still leave acceptable quality result.
Compressed Audio

- two different demands/markets:
  1. telephone
     - bit rate most critical factor
     - choose aggressive compression
     - accept poorer audio quality "muddy"
     - have to compress "on the fly"
     - highly compressed speech has "noisy" quality
     - common telephony standards include GSM compression (used by NIM)
  2. web/multimedia
     - bit rate less critical factor
     - can pre-compress, or compress not in real-time
     - quality essential as most traffic not just voice
     - multi-channel sound

mp3 is a currently popular format
- developed by Moving Picture Experts Group
- achieves good compression with minimal compromise on audio quality
- patience: 1/10 or better overall audio
- is proprietary format
- pay fee to access format
- ogg is popular open-source format
- compression as good as mp3
- audio quality slightly better

Compressing Video

- much more CPU intensive than compressing audio
- currently popular format: MPEG
- named for group that developed it
- uses complex mathematical compression procedure on elements of each video frame
- also exists frame-by-frame variation
- includes audio (mp3) encoded
- audio is compressed independently of video

Compressing Audio

- mp3, ogg or aac compression of audio data is CPU intensive
- as a server should share these pre-compressed
- special purpose hardware/software makes play is feasible/expensive
- e.g., Apple iPod

Compressing Decompressing

- usually done with combination of special purpose hw + firmware
- can be done entirely in software on general-purpose hardware
- but三人disproportionately higher cost
- however done, is said to use a codec
- Coder/Decoder
- codecs can be proprietary or open-source
CODECS

- proprietary codecs for voice telephony
  - many from ITU-T
  - licensing fees for use in software or hardware
- some examples for voice:
  - BroadVoice SB family, new, designed for VoIP
  - g722.1-law, std speech codecs
- some examples for audio:
  - AAC: Advanced Audio Coding, used in iTunes
  - Windows Media Audio 9 family

More CODECS

- some codecs for video:
  - DVI
  - Windows Media Video 9
  - Sorenson Video
- useful site with summaries & info:
  - http://www.silicon一番reetable.com/ffmpeg/video/cod
  - e.pdf

About Compression

- for pictures & audio we use **lossy** compression
  - discards parts of real information; cannot be recaptured
  - lossless then loss data won’t be missed
- not all compression applications accept lossy compression –
  - some need **lossless** compression
    - e.g., email
    - e.g., transporter

Lossy Compression

- JPEG is a lossy image compression algorithm
- degree of compression is selectable
  - higher compression = more loss

<table>
<thead>
<tr>
<th>Original image (27.67 KB)</th>
<th>JPEG image, &quot;75%&quot; (4.48 KB)</th>
</tr>
</thead>
</table>

How Bad is Lossy?

Compressed Text?

- yes, text can also be compressed
- use lossless compression techniques
  - RLE: run-length encoding
  - LZ
  - Huffman
- usually done to conserve space, not as concession to bit
  - rate concerns
- practically speaking, given current cost per-byte of disk
  - space, not worthwhile compressing text
  - unless is text that is rarely modified and perhaps rarely read
**IO in the UNIX World**

- `open()`
- `read()`
- `write()`
- `close()`

**UNIX Networking**

- use of network connections follows this UNIX pattern:
  - `open()`: set up the resources needed to establish a connection
  - `read()`: move data back and forth on open connection as long as you need to
  - `close()`: shutdown the connection
- actual function calls slightly different in name, but same in spirit

**Network Targets**

- opening a file is easy
  - give name of file as target
  - what is target in network?
    - a *host identifier* (IP address)

**Addresses**

**Network Targets**

- opening a file is easy
  - give name of file as target
- what is target in network?
  - a *host identifier* (IP address)
  - a *port* (where a process is supposed to be listening)
  - host > port = *socket*
Socket Summary

Client Basics

/* Create a socket */
sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
if (sock < 0) {
  perror(&sock, "socket creation failed\n");
  exit(1);
}
/* Create the socket to the specified server. */
if (connect(sockfd, (struct sockaddr *)&sa, sizeof(sa)) < 0) {
  perror(&sock, "Connect failed\n");
  exit(2);
}

/* Bind a local address to the socket */
if (bind(sockfd, (struct sockaddr *)&sa, sizeof(sa)) < 0) {
  perror(&sock, "Bind failed\n");
  exit(3);
}
/* Specify size of request queue */
if (listen(sockfd, 50) < 0) {
  perror(&sock, "Listen failed\n");
  exit(4);
}

Server Basics

while (1) {
  /* Accept the connection */
  if (accept(sockfd, (struct sockaddr *)&caddr, &len) < 0) {
    perror(&sock, "Accept failed\n");
    exit(5);
  }
  /* handle the connection */
  /* This server has been contacted \n  send(sockfd, "Hello, world\n", strlen(argv[0]));
  close(sockfd); */
}

Running It

* when server runs, we observe (from UNIX "netstat"):
  listen 6000 0.0.0.0 6000 \n  * LISTEN

* and if we run the client, we get the message:
  This server has been contacted 1 time.

* and a second time, etc.
  This server has been contacted 2 times.
Getting to a socket

- how does the request get to a server socket?
  - travels over network
  - originated on same computer reachadle by network
- how did the network do this?
  - many hard-working layers of software