IT441: Network Servers & Infrastructure

CLASS 2 : 31 Jan 2005
13:30 – 16:15

Except where otherwise noted, these slides are © copyright Charles Snow, 2004. Permission to copy these slides is granted so long as each slide contains the copyright notice "© C. M. Snow 2004" appears on each page and the copies are made freely available, i.e., not sold. Students are advised that they should not use University consumables (ink, toner, paper) to produce printed versions of these slides. Rights for all other use of this material are reserved by the author; permission for any other use must be obtained in writing from the author.

Resources: TAs

Prince William Campus:
Santhosh Sridhar
ssridha1@gmu.edu

check the course website http://netlab.gmu.edu/it441 for updates
Last Time:

- NEW and how to get set up to use it
  - http://netlab.gmu.edu/disted
- intro stuff
  - pre-reqs, grading, exams
- servers:
  - definition of a server
  - computers, characteristics that can affect servers
  - OS features
    - esp. virtual memory, processes & 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

*thrashing*: point where most (eventually, all) of CPU time is spent waiting for and handling page faults, none for actual processing of jobs
Using Disks

- what are timing considerations?
- data rate: how fast do bits move by the head?
  - 3600 RPM, 60 sectors/track $\Rightarrow$ $\approx$ 1.75 Mbytes/sec
  - drives typically have buffer memory to smooth-out data rate
- have different data xfer rates:

<table>
<thead>
<tr>
<th>Peak medium xfer</th>
<th>Sustained read</th>
<th>Sustained write</th>
</tr>
</thead>
<tbody>
<tr>
<td>54.2</td>
<td>44.2</td>
<td>24.2</td>
</tr>
</tbody>
</table>

- but head has to be in right place…

Using Disks

- data rate: how fast do bits move by the head?
- moving the head called a **seek**:
  - from one track to an adjacent track, typically 5 ms)
  - from one edge of the disk to the other (max. seek distance) typically 20 ms
- usually work with **average seek time** as time to seek half-way across the disk
  - typically 8.5 ms
- and disk has to be in right place…
Using Disks

- data rate: how fast do bits move by the head?
- moving the head called a seek
- once head is “on station” have to wait for rotational delay: get the correct sector to where head is
  - determined by rotation speed
  - at 3600 RPM full rotation is 16.67 ms
  - average rotational delay is half a rotation: 8.33 ms

so average latency to access data is
  - average seek time + average rotational delay

these delays totally overwhelm data transfer delay times

anything we can do to mitigate these delays goes long way toward improving efficiency…
Optical Disks

- delays for CDROM/CDRW optical disks are same as for magnetic disks, but worse:
  - rotation delay: disk spins more slowly and
  - change rotation motor speed: varies from 400 to 200 RPM (nominally, 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, hd>
  - specify length: number of sectors
- writing to a disk is analogous
- should the disk process requests in the order in which they are received?
Using Disks

should the disk process requests in the order in which they are received?
consider:

<table>
<thead>
<tr>
<th>Request Number</th>
<th>Cylinder</th>
<th>Sector</th>
<th>Head</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>583</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>i+1</td>
<td>26</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>i+2</td>
<td>583</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

- takes 26 ms + data transfer time
- but if resequenced, would only take: 13 ms
Disk Errors

- most disks today use error-correction techniques to
  - promote immunity to errors
  - allow higher density recording ⇒ higher capacities
- how frequently do errors occur?
  - non-recoverable data error: 1 in $10^{14}$ bits read
  - seek error rate: 1 in $10^8$ seeks
  - MTBF: $1.2 \times 10^6$ 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? 2 essential things:
  1. manage used-blocks on disk (these contain files)
  2. manage unused-blocks on disk (these contain no files)

Disk Layout With Linked-List Scheme

- boot sector
- File Allocation Table
- root directory
- data blocks
An early linked-list scheme

filename.txt is blocks: 6 → 8 → 4 → 2

Indexed File Management (UNIX)

- instead of information per block on the disk (as linked list schemes), keep information per file
  - single indirect block ptrs
  - double indirect block ptrs
  - triple indirect block ptrs
- what do you not see in the inode?

inode

- mode information
- link count
- uid owner
- gid
- file size in bytes
  - direct block ptrs
- atime
- mtime
- last inode mod time
Indexed File Management (UNIX)

- filename isn’t there
- a UNIX directory is a plain file with entries of form: 
  `<inode_num><filename>`
- is marked with attribute “directory”

inode

<table>
<thead>
<tr>
<th>mode information</th>
<th>link count</th>
<th>uid owner</th>
<th>gid</th>
</tr>
</thead>
<tbody>
<tr>
<td>file size in bytes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>atime</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mtime</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>last inode mod time</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Indexed File Management (UNIX)
Indexed File Management (UNIX)

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

assuming 1 Kb blocks

NTFS

- volume layout (partial):

<table>
<thead>
<tr>
<th>MFT</th>
<th>MFT mirror</th>
<th>log file</th>
<th>volume info</th>
<th>attributes definition table</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
<td>----------</td>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
<td>----------</td>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
<td>----------</td>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
<td>----------</td>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
<td>----------</td>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
<td>----------</td>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
<td>----------</td>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
<td>----------</td>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
<td>----------</td>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
<td>----------</td>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
<td>----------</td>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
<td>----------</td>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
<td>----------</td>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
<td>----------</td>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
<td>----------</td>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
<td>----------</td>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------------</td>
<td>----------</td>
<td>-------------</td>
<td>-----------------------------</td>
</tr>
</tbody>
</table>

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 concurrent multiple I/O requests
- or may be synchronized
  - spindles synchronized
  - access arms move together
  - good for high xfer speed
The server process

read the menu, make choice -> waiter takes order -> kitchen does magic

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:
The (computer) server process

- how to measure ‘cost’ for computer process:
  1. how much time to run process
     - CPU time
       - how improve?
     - waiting time, e.g., when blocked
  2. how much space the process needs
The (computer) server 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 network
  - from another process on same machine
  - standard format for request
    - how defined? by a service protocol
    - usually implies fixed cost for processing
  - clock starts ticking at server
    - but for client, clock started ‘long’ 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
A Server’s view of Requests

1. process request and determine how to resolve
2. generate response
3. deliver response
   ▪ depending on how it was received, 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 down time
- unscheduled down time
  - hardware failure
  - software failure
  - human error

What Servers Serve Up

- *information* service:
  - DHCP server provides IP address and related info
  - NTP server provides drift-corrected 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
  - compressibility
- questions:
  - how is content represented and stored on server?
  - how is it accessed and manipulated/processed?
  - 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., é, â, or special glyphs: ø, ß
- use other 128 positions available in 8-bit byte:
  - ISO-8859
Character Data

- problems remain:
  - some languages don't use 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:
  - **integers**: 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 representations
  - 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 threshold to force intermediate intensities to black or white
  - \( \Rightarrow \) monochrome (1 bit per pixel)

Monochrome Image
Still Image

- each pixel may represent:
  - amount of brightness in image at that point
  - between full black (0) and full white (0xFF)
  - use 1-byte per pixel to give 256 levels of gray
  - $\Rightarrow$ grayscale image

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
    - do 5-5-5 bits
    - do 5-6-5 bits
  - 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:
  - 100 dpi: grayscale ≈ 10 Kb, colour ≈ 30 Kb
  - 1000 dpi: grayscale ≈ 1000 Kb, colour ≈ 3000 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 \( s/2 \) Hz
  - human hearing ranges 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 for high-quality audio

Audio

- data rate for standard audio CD:
  \[
  \frac{44100 \text{ samples}}{\text{second}} \times 16 \frac{\text{bits}}{\text{sample}} \times 2 \text{ channels} = 1411200 \frac{\text{bits}}{\text{second}}
  \]
- or: 176,400 bytes/second for stereo
Audio

- alternative sampling: measure change in $s$ and represent it as
  - 1: $s$ increased since last measurement
  - 0: $s$ 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 image/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 left over
  - expert groups study how much can be discarded and still leave acceptable quality result
Compressed Audio

- two different demands/markets:
  1. telephony:
     - bit rate most critical factor
     - choose aggressive compression
     - emphasize “intelligibility” of speech
     - accept poorer audio quality “fidelity”
     - have to compress “on the fly”
     - highly compressed speech has “ringy” quality to it
     - common telephony standards include GSM compression (used by NEW)
  2. web/multi-media
     - 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
Compressed Audio

- mp3 is a currently popular format
  - developed by Motion Picture Experts Group
    http://www.chiariglione.org/mpeg
  - achieves good compression with ‘minimal’ compromise on audio quality
    - typically 10:1 or better over raw 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 Audio

- mp3, ogg or aac compression of audio data is CPU intensive
  - so a server should store these pre-compressed
- special purpose hardware/firmware makes players feasible/inexpensive
  - e.g., Apple i-pod
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 exploits frame-to-frame variation
  - includes audio (mp3) encoded
  - audio is compressed independently of video

Compressing Decompressing

- usually done with combination special-purpose hw + firmware
- can be done entirely in software on general-purpose hardware
  - but at 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:
  - BroadVoice16: fairly new, designed for VOIP
  - µlaw/A-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:
  - DivX
  - Windows Media Video 9
  - Sorenson video
- useful site with summaries & info:
About Compression

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

Lossy Compression

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

<table>
<thead>
<tr>
<th>Image Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>original image, no compression (tiff)</td>
<td>527,676 bytes</td>
</tr>
<tr>
<td>jpeg image, “75%”</td>
<td>24,485 bytes</td>
</tr>
</tbody>
</table>
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
I/O in the UNIX World

- open(): set up the resources needed to establish a connection
- read()/write(): 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

UNIX Networking

- use of network connections follows this UNIX pattern:
  - open: set up the resources needed to establish a connection
  - read/write: 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?

Addresses
Network Targets

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

- host + port = socket
Socket Summary

Client Server Example (Comer 30)

- Comer provides complete socket example programs in chapter 30
  - client
  - server
- client contacts server and is told how many times server has been contacted
Client Basics

/* Create a socket. */

sd = socket(PF_INET, SOCK_STREAM, ptrp->p_proto);
if (sd < 0) {
    fprintf(stderr, "socket creation failed\n");
    exit(1);
}

/* Connect the socket to the specified server. */

if (connect(sd, (struct sockaddr *)&sad, sizeof(sad)) < 0) {
    fprintf(stderr,"connect failed\n");
    exit(1);
}

Client Basics

n = recv(sd, buf, sizeof(buf), 0);
while (n > 0) {
    write(l,buf,n);
    n = recv(sd, buf, sizeof(buf), 0);
}

close (sd);
Server Basics

/* Create a socket */

sd = socket(PF_INET, SOCK_STREAM, ptrp->p_proto);
if (sd < 0) {
    fprintf(stderr, "socket creation failed\n");
    exit(1);
}

/* Bind a local address to the socket */

if (bind(sd, (struct sockaddr *)&sad, sizeof(sad)) < 0) {
    fprintf(stderr,"bind failed\n");
    exit(1);
}

/* Specify size of request queue */

if (listen(sd, QLEN) < 0) {
    fprintf(stderr,"listen failed\n");
    exit(1);
}

while (1) {
    alen = sizeof(cad);
    if (sd2 = accept(sd,(struct sockaddr *)&cad,&alen)) < 0) {
        fprintf(stderr, "accept failed\n");
        exit(1);
    }
    visits++;
    sprintf(buf,"This server has been contacted %d time%s\n",visits, (visits==1?":":"s.");
    send(sd2,buf,strlen(buf),0);
    close(sd2);
}
Running It

- when server runs, we observe (from UNIX “netstat”):

<table>
<thead>
<tr>
<th>Proto</th>
<th>Recv-Q</th>
<th>Send-Q</th>
<th>Local Address</th>
<th>Foreign Address</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcp</td>
<td>0</td>
<td>0</td>
<td>*:5193</td>
<td><em>:</em></td>
<td>LISTEN</td>
</tr>
</tbody>
</table>

- 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 some computer reachable by network

- how did the network do this?
  - many hard-working layers of software
A Layered Networking Model

- Application
- Presentation
- Session
- Transport
- Network
- DLC
- Physical

- Application
- Presentation
- Session
- Transport
- Network
- DLC
- Physical