IT441: Network Servers & Infrastructure

CLASS 1: 24 Jan 2005
13:30 – 16:15

IT441: Where and When

- live classroom presentation, Mondays 13:30 to 16:15
- originating from Prince William campus: BRH:131

IT441: Where and When

- live classroom presentation, Mondays 13:30 to 16:15
- live-by-Internet
- recorded as given in class, may be played back anytime

IT441: Where and When

- live-by-Internet uses GMU’s Network EducationWare new software
  for distance learning: http://netlab.gmu.edu/diated

NEW: Network EducationWare

- provides, live, in real-time:
  - overheads of class presentation with
  - instructor annotations and
  - instructor voice, synchronized to overheads
  - audiongraphies
  - live video of classroom
NEW: Network EducationWare

* provides, live, in real-time:
  * lets students ask questions:
    * by typing them in
    * by speaking them (if client does audio)
  * provides chat-room resources for students during lecture (at any time, actually)
  * entire lecture recorded for later Internet playback (excluding live video)

Using NEW

* need client software; currently runs only on Windows platforms (98, ME, NT, 2000, XP)
* minimal machine configuration:
  * P350 or faster + ≥ 64 Mb memory
  * IE 4.0 or later, Netscape (6.0 or later)
  * standard sound card
  * ≥ 56 kbps connection to Internet
* see http://netlab.gmu.edu/disted to try it and/or sign up (need password to get courses)

Getting and Using NEW

* go to http://netlab.gmu.edu/disted
Course Resources

- required text: 
  * Computer Networks and Internets with Internet Applications (4th edition) 
  * Douglas Comer 
- course web site: 
  http://netlab.gmu.edu/it441 
  * contains printed copies of slides (pdf) 
  * other course related info 
- background material

Resources: Instructor

- Instructor: Charles Snow 
  * email: csnow@netlab.gmu.edu 
- Office Hours 
  * Prince William Campus 
  * Bull Run Hall 122A 
  * Tuesday 10:00 – 12:00, 14:30 – 15:30

Resources: TAs

- check the course website for updates 
  * http://netlab.gmu.edu/it441

Students must:

- course prerequisites: 
  * IT341 Network Essentials 
  * MATH 108 Introductory Calculus with Business Applications 
  * MATH125 Discrete Mathematics

Course Project:

- two options for projects: 
  - Plan A: 
    * series of small lab exercises (4 or 5) configuring/operating network services on Windows and Linux platforms 
    * given out in second half of course 
    * each lab has exercise + questions to answer 
    * no teams: all labs are individual efforts

- two options for projects: 
  - Plan A: lab exercises 
  - Plan B: 
    * given a scenario, develop a feasible plan for implementing a solution 
    * involves finding/specifying all needed resources to implement solution 
    * interim project work required (e.g., status reports) 
    * produce a report and a short presentation on the solution 
    * can work in teams of up to 3 members 
    * but effort of each individual must be clearly identified in all work
Course Project:
- two options for projects
- students commit to one or the other plan at start of semester
- no changing of plan thereafter

Plan B Scenarios
1) airport traveler assistant:
   - provide infrastructure and client-side resources to give a traveller a nearly fully automated travel guide
2) tsunami relief network:
   - implement, in a short time, an Internet system on-the-ground in the area affected by the Dec 26th tsunami for the use of all involved in rescue, refugee management, and reconstruction
   - be aware that the full scenario description may not be nearly so detailed as statement-of-work descriptions you are used to from other courses
   - but you still have to provide a full solution...

IT441 Grading:

<table>
<thead>
<tr>
<th></th>
<th>Plan A</th>
<th>Plan B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homework Exercises</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Midterm</td>
<td>25</td>
<td>20</td>
</tr>
<tr>
<td>Project</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>Final</td>
<td>40</td>
<td>35</td>
</tr>
</tbody>
</table>

- assignments due by end-of-day on due date
- assignments handed via http://cs.gmu.edu/course-upload
- grades are proficiency based
- no extra credit work option

Important that you demonstrate:
- understanding of material
- ability to reason with it
- final exam
  - is comprehensive
  - scheduled for 16 May 2005, during class time
- basic rule: if we cover it in class, it can be on an exam
  - includes reading material suggested but not necessarily explicitly covered in class

IT441 Exams
- all students must be present, on campus, to write exams in the course
- both midterm and final
- bring GMU ID card and be prepared to show it
- arrangements for missing an exam must be made before the exam

What we look at

client
What we look at

• server-side questions:
  • who do I listen to? not listen to?
  • does it matter who they are?
  • what kinds of requests do I respond to?
  • how do I make a response?
  • all by myself, or do I get help from others?
  • what do I charge? how do I collect?
  • what formats can I respond with?

Server-side Considerations

• some network questions:
  • how connect?
  • route? does it matter?
  • how long stay connected?
  • context?
  • quality of service?

Network Considerations

What we look at

• what do we need:
  • to make this work
  • to make it work better? faster? cheaper? bigger?
  • to expand to do more?

Technology Focus

• this course focuses on the technology needed to support client ↔ server interaction
  • theory
  • hardware
  • software
  • costs
    • capital
    • on-going
  • operating
  • expanding/contracting
Why Theory?

What's a Server?

- something that provides a service
  elements:
  * hardware
  * software
  * energy
  * $impose constraints on ability to provide service

What's a Server?

- something that provides a service
  elements:
  * hardware
  * software
  * energy
  * $impose constraints on ability to provide service

Server: Inside to Outside

- examine briefly a server from the inside out
- server's purpose is to provide a service
- service is provided by a program running on the server

A basic computer:
**A basic computer:**

- **CPU:**
  - executes instructions retrieved from memory.
  - has registers and status word(s).
  - is a bus master most of the time.
  - may surrender bus to other devices for brief intervals.
  - can generate traps for failure conditions.

- **MEMORY:**
  - stores bit patterns (instructions, data).
  - organized as flat, linear sequence of bytes.
  - is byte addressable (usually).
  - may generate traps for error conditions.

**Memories...**

- place to hold a value (data or instruction) while it is worked on or until it is needed to be worked on.

- may require electrical power to hold a value: volatile memory

- may not require electrical power: non-volatile memory

  - earliest computer memories, magnetic core, were non-volatile

- not all memories are the same:
  - smaller access time ⇔ higher cost per bit
  - bigger capacity ⇔ smaller cost per bit
  - higher capacity ⇔ higher access time
Memories...

REGISTERS:
- contained within CPU
- GPRs: general purpose registers
- PC: program counter
- PSW: program status word
- access at CPU speed
- volatile

Memories...

CACHE:
- small amount of high-speed mem between CPU and bus
- may be multi-staged
- volatile

Memories...

MAIN MEMORY:
- byte addressable
- flat, linear address space
- typical sizes: 256 MB to 2 GB
- volatile

Memories...

DISKS (magnetic, optical):
- are peripheral devices
- interact through interfaces/ controllers
- magnetics: now in 100s of GB
- opticals: now in 10s of GB
- mechanical access method (e.g., read/write head must be moved to correct position)
- non-volatile
- magnetics subject to magnetic fields

Memories...

MAGNETIC TAPE:
- are peripheral devices
- interact through interfaces/ controllers
- now in 100s of GB
- mechanical access method: tape must be moved to correct position
- sequential access only
- non-volatile
- magnetics subject to magnetic fields

Memories...

DECREASING SIZE OF DATA-CHUNK TRANSFERRED
- lower cost per bit
- higher capacity
- higher access time
- lower frequency of access by CPU

Memories...

SMALL, expensive, fast memory supplemented by larger, cheaper, slower memory
Choreography

- so you have this hardware
  - cpu, cache, bus, memory, i/o devices ...
  - do you have a server?

- need some software to operate the hardware
  - wrap an operating system around the hardware
  - manages resources
  - creates virtual resources
  - so now it's a server?

Choreography

- add 'applications' like httpd and you have a web server
- so:
  - server is a program
  - program runs as app under OS

- does everything needed by computer fit into main memory?
- what if it doesn't?

Virtual Memory

- memory usually organized as set of pages or segments
- or combinations of these
- part of program being run placed into memory
  - don't need it all there at once
  - keep other parts elsewhere (somewhere cheaper)
- meaning?

- use of disk space to provide additional memory space
- tradeoffs:
  - memory pricing: $170/Gb
  - disk pricing: $0.5875/Gb
  - memory avg access: 10s of ns
  - disk avg access: 13 ms (13,000,000 ns)
  - when program needs a page not resident, must wait > 10^4 memory-times for it
  - can be twice this bad
Virtual Memory

- Incidence of these page faults not reliably predictable
- So bad news for real-time applications
- Is this still a big deal?
- 32-bit processors can only access 4 GB anyway, so for $700 can have all RAM
- What about 64-bit processors? (can access 17,179,869,184 GB)
- Bottom line: these days, use RAM to span entire address space for 32-bit machines

Running a Program

- A running program = process
- What resources does a process need?
  - CPU
  - Memory
  - Devices
- Once "running," does a process run non-stop until done?
  - What happens if it doesn't?

Running a Program

- OS needs to switch from the context of the waiting process to one that is runnable
- Context switches are $\$$ overhead
- Other context switches?

Running a Program

- Devices are not directly accessible to processes
- Accessed via system calls
  - Ask the OS to work on process' behalf
- System call = context switch
  - At least two
  - What does process do while waiting for system call to return?

Life of a Process

- To manage processes, OS needs to keep information about the proc
  - Size of memory used
  - Context info (suspend/resume)
  - Others?
- Where does OS keep all this?
  - Process control block (PCB)
- Each new process requires creating a new PCB and initializing it
Workload on Server

- server is a process running on a computer
- process has certain resources:
  - CPU
  - memory
  - I/O access
- process runs instructions along exactly one execution path

Another view of a Process

- distinct:
  - process has
  - process does
- the term "task" often used to refer to what the process has
- look at the "does" part...

Another view of a Process

- suppose we can watch the process as it wanders through memory:
  - imagine we can leave a dot at each location the process visits for instructions:

Another view of a Process

- and if we connect the dots, we get a stringy line tracing the execution of the process: a thread

- "normal" processes we're used to have exactly 1 thread of execution
- can we have > 1?

Another view of a Process

- can we have > 1?
- YES
  - but what do we need to make that work?

Threads, Processes

- each thread needs to have:
  - a PC value
  - a PSW value
  - a GPR set
  - tiny amount of stack space
  - a TID: thread ID

Multi-threaded Processes

- now can have multiple threads of execution in the same memory space
Multi-threaded Processes

- now can have multiple threads of execution in the **same** memory space
- so all variables are shared across all threads in the process:
  - **there is only one copy of the data**

Multi-threaded Processes

- now can have multiple threads of execution in the **same** memory space
- so all variables are shared across all threads in the process:
  - **there is only one copy of the data**
  - great for sharing data between different threads
  - different threads within a process may do different things but need to exchange info
  - fraught with peril
  - what 1 thread does to data affects every thread using that data

Multi-threaded Processes

- now can have multiple threads of execution in the **same** memory space
- so all variables are shared across all threads in the process:
  - **there is only one copy of the data**

Threads

- so a thread is a **unit of execution** within a process
- where do threads come from?
  - created by parent threads
  - a server may need to invoke a process to handle a client request
    - e.g., a database lookup
    - e.g., perform some computation with client data to produce result for response
  - or, perhaps, create a new thread to handle request

Thread Control: a User Issue

- how expensive is a thread-switch compared to a process-switch?
- not all threads are created equal
  - user-level threads
    - kernel doesn’t know they exist
    - user process controls creation/switching between threads
    - problem when thread does I/O or makes syscall
  - kernel-level threads
    - kernel does know they exist
    - kernel controls creation/switching between threads

Where the Threads are...

- kernels need not know anything about threads
  - all the thread support exists in a set of library functions, e.g., pthreads
  - these threads often called **user-level threads**
- disadvantage:
  - when thread blocks, all threads block
  - can’t really benefit from multiple CPUs if avai
  - advantages?
User Level Threads

- advantages:
  1. thread switching is very fast, happens entirely within user process/application, no mode changes
  2. thread scheduling handled by user/application so can implement scheduling model best suited to app
  3. library supported, so highly portable

Where Else the Threads are

- move all thread support into kernel: kernel level threads
  - no thread management code in application
  - app uses API to kernel thread functions
  - e.g., Linux, Win32
  - kernel schedules individual threads
  - thread-level, not process-level, scheduling granularity

Kernel level threads

- advantages:
  1. kernel can schedule multiple threads concurrently on multiple CPUs if avid
  2. blocking one thread need not impact processing of other threads in the process

- disadvantages?
  - thread switching involves kernel intervention, mode changes

Thread Performance

- how do user-level/kernel-level threads compare
  - with each other?
  - with use of separate processes?
  - measurements reported by Anderson et. al. (1992)
    - on a VAX where a procedure invocation takes 7μs
    - kernel trap takes 17 μs
    - tested:
      - null-fork: time to create child thread/process that, once created does nothing except exit
      - signal-wait: time for a thread/process to signal a waiting thread/process and wait for an event

Thread Performance

- results: (times in microseconds)

<table>
<thead>
<tr>
<th>TEST</th>
<th>USER-LEVEL THREADS</th>
<th>KERNEL-LEVEL THREADS</th>
<th>PROCESSES</th>
</tr>
</thead>
<tbody>
<tr>
<td>null-fork</td>
<td>34</td>
<td>949</td>
<td>11301</td>
</tr>
<tr>
<td>signal-wait</td>
<td>37</td>
<td>441</td>
<td>1840</td>
</tr>
</tbody>
</table>

Server Performance

- how do servers handle requests?
Server Performance

- how do servers handle requests?
  - queue up and process sequentially through single server instance

Server Performance

- how do servers handle requests?
  - queue up and process sequentially through single server instance
  - server makes copy of itself to handle a received request
    - i.e., a new server process instance for that request

Simple Interaction Example

- http request
- http response

Less Simple Interaction Example

- http request
- db server
- http server
- http response

Interaction Example

- http request
- db server
- http server
- http response
Role of Middleware

Middleware server knows what computations to do to get the data and then how to generate the required HTML to return to client.

Interaction Example

For next time...

- Assignment 1 is available on the course web site
- It is due 30 Jan 2005
- See instructions on course web site for how to submit