

TCSS 558: APPLIED DISTRIBUTED COMPUTING

Processes: Threads & Virtualization, Clients & Servers

Wes J. Lloyd
 School of Engineering
 & Technology (SET)
 University of Washington - Tacoma

1

OBJECTIVES - 1/26

- **Questions from 1/26**
- Assignment 0: Cloud Computing Infrastructure Tutorial
 - New testFibService.sh script
- Assignment 1: Key/Value Store - Next Week
- Chapter 3: Processes
 - Chapter 3.1: Threads
 - Context Switches
 - Threading Models
 - Multithreaded clients/servers
 - Chapter 3.2: Virtualization
 - Chapter 3.3: Clients
 - Chapter 3.4: Servers

January 26, 2023
TCSS558: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.2

2

ONLINE DAILY FEEDBACK SURVEY

- Daily Feedback Quiz in Canvas – Available After Each Class
- Extra credit available for completing surveys **ON TIME**
- Tuesday surveys: due by ~ Wed @ 10p
- Thursday surveys: due ~ Mon @ 10p

January 26, 2023
TCSS558: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.3

3

SURVEY QUESTIONS

- Survey has two questions:
- Add questions about the previous class to the **second question**
- 1st question: After today's class, comment on any new concepts that you learned about?
 - *It is helpful to know what is new, useful, less useful...*
- 2nd question: After today's class, what point(s) remain least clear to you?
 - >> **Please add questions HERE**

January 26, 2023
TCSS558: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.4

4

TCSS 558 - Online Daily Feedback Survey - 1/5

Due Jan 6 at 10pm Points 1 Questions 4
 Available Jan 5 at 1:30pm - Jan 6 at 11:59pm 1 day Time Limit None

Question 1 0.5 pts

On a scale of 1 to 10, please classify your perspective on material covered in today's class:

| | | | | | | | | | |
|---------------------|---|---|----------------------|---|---|---|------------------|---|----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Mostly Review To Me | | | Equal New and Review | | | | Mostly New To Me | | |

Question 2 0.5 pts

Please rate the pace of today's class:

| | | | | | | | | | |
|------|---|---|------------|---|---|---|------|---|----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Slow | | | Just Right | | | | Fast | | |

January 26, 2023
TCSS558: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.5

5

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (31 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average - 6.60 (↑ - previous 6.56)**
- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average - 5.77 (↑ - previous 5.58)**

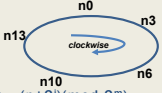
January 26, 2023
TCSS558: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.6

6

FEEDBACK FROM 1/24

- On slide 35, could you please clarify as to what is n in the formula to compute finger table, like in the example n was taken as 3 for $n3$, would it be $n=6$ for $n6$?
- n is the node ID
- Yes, n would be 6, for node 6 ($n6$)
- for $i=0$ to $m-1$
- i^{th} entry in FT for node with id n is **first node** $\geq (n+2^i) \pmod{2^m}$
- m = number of hash bits
- N3 finger table**
- | i (start) | $ft[i]$ (node) |
|-------------------------|--|
| $(3+2^0=4) \pmod{2^m}$ | $n6$ (first node going clockwise from i) |
| $(3+2^1=5) \pmod{2^m}$ | $n6$ (first node going clockwise from i) |
| $(3+2^2=7) \pmod{2^m}$ | $n10$ (first node going clockwise from i) |
| $(3+2^3=11) \pmod{2^m}$ | $n13$ (first node going clockwise from i) |

Note: the mod does nothing here ...




January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.7

7

FEEDBACK - 2

- Compute finger table for $n10$:
- The mod now must be considered
- n is the node ID
- m = number of hash bits = 4
- N10 finger table** for $i=0$ to $m-1$
- | i (start) | $ft[i]$ (node) |
|--------------------------|--|
| $(10+2^0=11) \pmod{2^m}$ | $n13$ (first node going clockwise from i) |
| $(10+2^1=12) \pmod{2^m}$ | $n13$ (first node going clockwise from i) |
| $(10+2^2=14) \pmod{2^m}$ | $n0$ (first node going clockwise from i) |
| $(10+2^3=18) \pmod{2^m}$ | $n3$ (first node going clockwise from i) |



January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.8

8

FEEDBACK - 2

- Are there any empirical studies showing the success of policy-based search methods over flooding or random walks?
- How should 'success' be defined?
- Success could mean many things.
- Success could be fulfilling functional requirements w/o bugs
- Success could be meeting various distributed systems goals: distribution transparency, accessibility, availability, etc.
- Success could be having better performance vs. unstructured peer-to-peer systems communicating with flooding or random walk - What is performance?
 - communication round-trip-time
 - volume/throughput (transactions/second)
 - network latency (overhead)

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.9

9

OBJECTIVES - 1/26

- Questions from 1/26
- Assignment 0: Cloud Computing Infrastructure Tutorial**
 - New testFibService.sh script**
- Assignment 1: Key/Value Store - Next Week
- Chapter 3: Processes
 - Chapter 3.1: Threads
 - Context Switches
 - Threading Models
 - Multithreaded clients/servers
 - Chapter 3.2: Virtualization
 - Chapter 3.3: Clients
 - Chapter 3.4: Servers

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.10

10

ASSIGNMENT 0

- Preparing for Assignment 0:**
 - Establish AWS Account
 - Standard account
 - Complete AWS Cloud Credits Survey and provide AWS account ID
 - Credits will be automatically loaded by Amazon into accounts
- Tasks:**
 - Task 1 - Establish local Linux/Ubuntu environment
 - Task 2 - AWS account setup, obtain user credentials
 - Task 3 - Intro to: Amazon EC2 & Docker: create Dockerfile for Apache Tomcat
 - Task 4 - Create Dockerfile for haproxy
 - Task 5 - Working with Docker-Machine
 - Task 6 - Config 3 multiple server configs to load balance requests for RESTful Fibonacci web service
 - Task 7 - Test configs and submit results

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.11

11

TESTING CONNECTIVITY TO SERVER

- testFibPar.sh** script is a parallel test script
- Orchestrates multiple threads on client to invoke server multiple times in parallel
- To simplify coordinate of parallel service calls in BASH, **testFibPar.sh** script ignores errors !!!
- To help test client-to-server connectivity, have created a new **testFibService.sh** script
- TEST 1: Network layer
 - Ping (ICMP)
- TEST 2: Transport layer
 - TCP: telnet (TCP Port 8080) - security group (firewall) test
- TEST 3: Application layer
 - HTTP REST - web service test

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.12

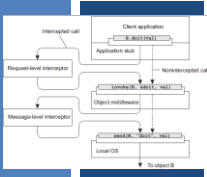
12

OBJECTIVES - 1/26

- **Questions from 1/26**
- **Assignment 0: Cloud Computing Infrastructure Tutorial**
 - New testFibService.sh script
- **Assignment 1: Key/Value Store - Next Week**
- **Chapter 3: Processes**
 - Chapter 3.1: Threads
 - Context Switches
 - Threading Models
 - Multithreaded clients/servers
 - Chapter 3.2: Virtualization
 - Chapter 3.3: Clients
 - Chapter 3.4: Servers

January 26, 2023 TCSS558: Applied Distributed Computing [Winter 2023]
 School of Engineering and Technology, University of Washington - Tacoma L8.13

13



CH 2.3: SYSTEM ARCHITECTURES

January 24, 2023 TCSS558: Applied Distributed Computing [Winter 2023]
 School of Engineering and Technology, University of Washington - Tacoma L7.14

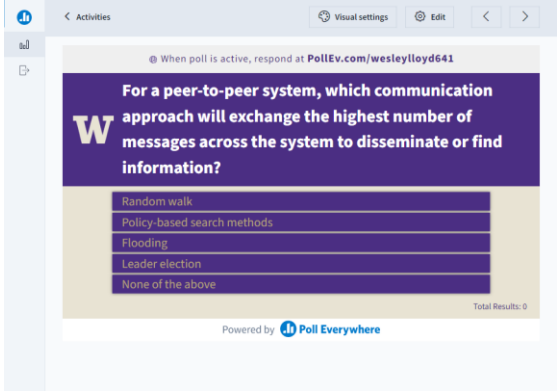
14

REVIEW QUESTIONS

- What is difference in finding/disseminating data in unstructured vs. structured peer-to-peer networks?
 - Structured: can grasp the number of messages required based on the organization and structure of the system
 - Fixed hypercube
 - Chord system
 - Unstructured: requires broadcast like message schemes that gossip to find/disseminate data: Flooding, Random walk

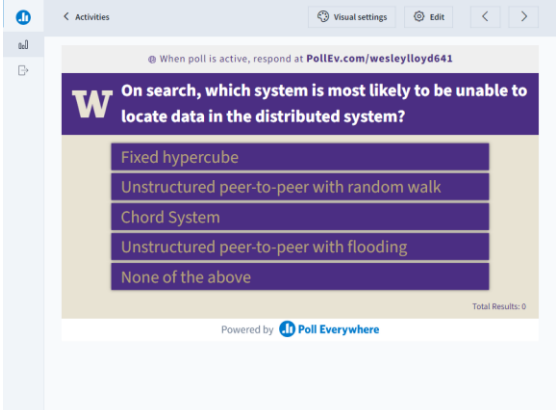
January 24, 2023 TCSS558: Applied Distributed Computing [Winter 2023]
 School of Engineering and Technology, University of Washington - Tacoma L7.15

15



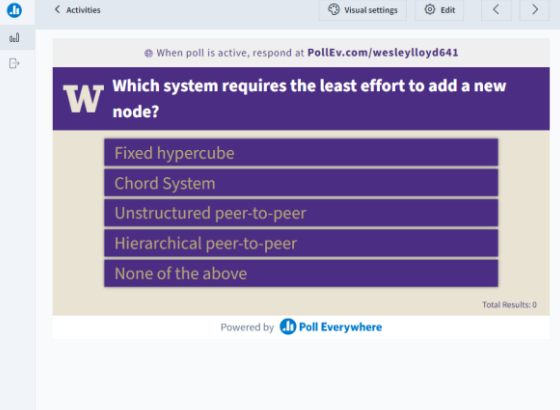
Powered by Poll Everywhere

16



Powered by Poll Everywhere

17



Powered by Poll Everywhere

18

W Which system features strict constraints regarding how the system can add or remove nodes?

- Fixed hypercube
- Chord System
- Unstructured peer to peer
- Hierarchical peer-to-peer
- Edge/Fog system
- None of the above

Total Results: 0

Powered by Poll Everywhere

19

W Which servers/nodes seek to store data very close to a user?

- Cloud computing server
- Super peers
- Fog server
- Edge server
- Weak peer

Total Results: 0

Powered by Poll Everywhere

20

W What are advantages of a decentralized structured peer-to-peer architecture?

No responses received yet. They will appear here...

Total Results: 0

Powered by Poll Everywhere

21

W What are advantages of a decentralized unstructured peer-to-peer architecture?

No responses received yet. They will appear here...

Total Results: 0

Powered by Poll Everywhere

22

OBJECTIVES - 1/26

- Questions from 1/26
- Assignment 0: Cloud Computing Infrastructure Tutorial
 - New testFibService.sh script
- Assignment 1: Key/Value Store - Next Week
- Chapter 3: Processes**
- Chapter 3.1: Threads
 - Context Switches
 - Threading Models
 - Multithreaded clients/servers
- Chapter 3.2: Virtualization
- Chapter 3.3: Clients
- Chapter 3.4: Servers

TCSS558: Applied Distributed Computing (Winter 2023)
School of Engineering and Technology, University of Washington - Tacoma

January 26, 2023

LS.23

23

CH. 3: PROCESSES
CH. 3.1: THREADS

Binary program count

Workload diversity (process types)

LS.24

24

CHAPTER 3

- Chapter 3 titled "processes"
- Covers variety of distributed system implementation details
- "Grab bag" of topics
 - Processes/threads
 - Virtualization
 - Clients
 - Servers
 - Code migration

January 26, 2023 TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma LR.25

25


OBJECTIVES - 1/26

- Questions from 1/26
- Assignment 0: Cloud Computing Infrastructure Tutorial
 - New testFibService.sh script
- Assignment 1: Key/Value Store - Next Week
- Chapter 3: Processes
 - Chapter 3.1: Threads**
 - Context Switches
 - Threading Models
 - Multithreaded clients/servers
 - Chapter 3.2: Virtualization
 - Chapter 3.3: Clients
 - Chapter 3.4: Servers

January 26, 2023 TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma LR.26

26

CH. 3.1 - THREADS




- For implementing a server (or client) threads offer many advantages vs. heavy weight processes
- What is the difference between a process and a thread?**
 - (review?) from Operating Systems
- Key difference: what do threads share amongst each other that processes do not.... ?**
- What are the segments of a program stored in memory?**
 - Heap segment (dynamic shared memory)
 - Code segment
 - Stack segment
 - Data segment (global variables)

January 26, 2023 TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma LR.27

27

THREADS - 2




- Do several processes on an operating system share...**
 - Heap segment?
 - Stack segment?
 - Code segment?
- Can we run multiple copies of the same code?**
- These may be managed as shared pages (across processes) in memory
- Processes are isolated from each other by the OS
 - Each has a separate heap, stack, code segment

January 26, 2023 TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma LR.28

28

THREADS - 3

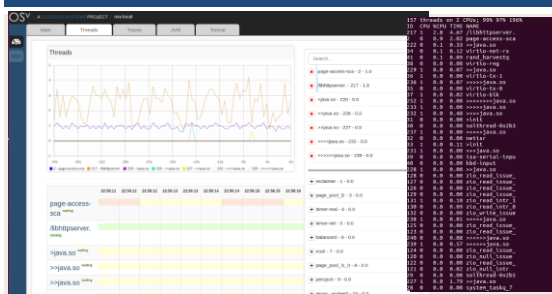


- Threads avoid the overhead of process creation
- No new heap or code segments required
- What is a context switch?**
- Context switching among threads is considered to be more efficient than context switching processes
- Less elements to swap-in and swap-out
- Unikernel: specialized single process OS for the cloud
- Example: Osv, Clive, MirageOS (see: <http://unikernel.org/projects/>)
- Single process operating system with many threads
- Developed for the cloud to run only one application at a time

January 26, 2023 TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma LR.29

29


OSV: ONE PROCESS, MANY THREADS



January 26, 2023 TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma LR.30

30

THREADS - 4



- Important implications with threads:
 - (1) multi-threading should lead to performance gains
 - (2) thread programming requires additional effort when threads share memory
 - Known as thread **synchronization**, or enabling **concurrency**
- Access to **critical sections** of code which modify shared variables must be **mutually exclusive**
 - No more than one thread can execute at any given time
 - Critical sections must run **atomically** on the CPU

January 26, 2023 TCCS558: Applied Distributed Computing [Winter 2023]
 School of Engineering and Technology, University of Washington - Tacoma L8.31

31

BLOCKING THREADS

- Example: spreadsheet with formula to compute sum of column
 - User modifies values in column
- Multiple threads:
 - Supports interaction (UI) activity with user
 - Updates spreadsheet calculations in parallel
 - Continually backs up spreadsheet changes to disk
- Single core CPU
 - Tasks appear as if they are performed simultaneously
- Multi core CPU
 - Tasks **execute** simultaneously

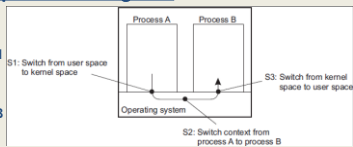
January 26, 2023 TCCS558: Applied Distributed Computing [Winter 2023]
 School of Engineering and Technology, University of Washington - Tacoma L8.32

32

INTERPROCESS COMMUNICATION

- IPC – mechanism using pipes, message queues, and shared memory segments
- IPC mechanisms incur context switching
 - Process I/O must execute in kernel mode
- How many context switches are required for process A to send a message to process B using IPC?**

- #1 C/S:**
Proc A → kernel thread
- #2 C/S:**
Kernel thread → Proc B



January 26, 2023 TCCS558: Applied Distributed Computing [Winter 2023]
 School of Engineering and Technology, University of Washington - Tacoma L8.33

33

OBJECTIVES – 1/26

- Questions from 1/26
- Assignment 0: Cloud Computing Infrastructure Tutorial
 - New testFibService.sh script
- Assignment 1: Key/Value Store - Next Week
- Chapter 3: Processes
 - Chapter 3.1: Threads
 - Context Switches**
 - Threading Models
 - Multithreaded clients/servers
 - Chapter 3.2: Virtualization
 - Chapter 3.3: Clients
 - Chapter 3.4: Servers

January 26, 2023 TCCS558: Applied Distributed Computing [Winter 2023]
 School of Engineering and Technology, University of Washington - Tacoma L8.34

34

CONTEXT SWITCHING

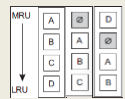
- Direct overhead**
 - Time spent not executing program code (user or kernel)
 - Time spent executing interrupt routines to swap memory segments of different processes (or threads) in the CPU
 - Stack, code, heap, registers, code pointers, stack pointers
 - Memory page cache invalidation
- Indirect overhead**
 - Overhead not directly attributed to the physical actions of the context switch
 - Captures performance degradation related to the side effects of context switching (e.g. rewriting of memory caches, etc.)
 - Primarily cache perturbation**

January 26, 2023 TCCS558: Applied Distributed Computing [Winter 2023]
 School of Engineering and Technology, University of Washington - Tacoma L8.35

35

CONTEXT SWITCH – CACHE PERTURBATION

- Refers to cache reorganization that occurs as a result of a context switch
- Cache is not clear, but elements from cache are removed as a result of another program running in the CPU
- 80% performance overhead from context switching results from this **"cache perturbation"**



January 26, 2023 TCCS558: Applied Distributed Computing [Winter 2023]
 School of Engineering and Technology, University of Washington - Tacoma L8.36

36

OBJECTIVES - 1/26

- **Questions from 1/26**
- **Assignment 0: Cloud Computing Infrastructure Tutorial**
 - New testFibService.sh script
- **Assignment 1: Key/Value Store - Next Week**
- **Chapter 3: Processes**
 - Chapter 3.1: Threads
 - Context Switches
 - **Threading Models**
 - Multithreaded clients/servers
 - Chapter 3.2: Virtualization
 - Chapter 3.3: Clients
 - Chapter 3.4: Servers

| | | |
|------------------|---|-------|
| January 26, 2023 | TCCSS58: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L8.37 |
|------------------|---|-------|

37

THREADING MODELS

- **Many-to-one threading:** multiple user-level threads per process
- Thread operations (create, delete, locks) run in user mode
- Multithreaded process mapped to single schedulable entity
- Only one thread per process runs at any given time
- Key take-away: thread management handled by user processes

- **What are some advantages of many-to-one threading?**

- **What are some disadvantages?**

| | | |
|------------------|---|-------|
| January 26, 2023 | TCCSS58: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L8.38 |
|------------------|---|-------|

38

THREADING MODELS - 2

- **One-to-one threading:** use of separate kernel threads for each user process - also called **kernel-level threads**
- The kernel API calls (e.g. I/O, locking) are farmed out to an existing kernel level thread

- Thread operations (create, delete, locks) run in kernel mode
- Threads scheduled individually by the OS
- System calls required, context switches as expensive as process context switching
- Idea is to have preinitialized kernel threads for user processes
- Linux uses this model...

- **What are some advantages of one-to-one threading?**

- **What are some disadvantages?**

| | | |
|------------------|---|-------|
| January 26, 2023 | TCCSS58: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L8.39 |
|------------------|---|-------|

39

APPLICATION EXAMPLES

- Google chrome: processes
- Apache tomcat webserver: threads

- Multiprocess programming avoids synchronization of concurrent access to shared data, by providing coordination and data sharing via interprocess communication (IPC)

- Each process maintains its own private memory

- **While this approach avoids synchronizing concurrent access to shared memory, what is the tradeoff(s) ??**
 - Replication instead of synchronization - must synchronize multiple copies of the data

- **Do distributed objects share memory?**

| | | |
|------------------|---|-------|
| January 26, 2023 | TCCSS58: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L8.40 |
|------------------|---|-------|

40

WE WILL RETURN AT 2:40PM



41

OBJECTIVES - 1/26

- **Questions from 1/26**
- **Assignment 0: Cloud Computing Infrastructure Tutorial**
 - New testFibService.sh script
- **Assignment 1: Key/Value Store - Next Week**
- **Chapter 3: Processes**
 - Chapter 3.1: Threads
 - Context Switches
 - Threading Models
 - **Multithreaded clients/servers**
 - Chapter 3.2: Virtualization
 - Chapter 3.3: Clients
 - Chapter 3.4: Servers

| | | |
|------------------|---|-------|
| January 26, 2023 | TCCSS58: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma | L8.42 |
|------------------|---|-------|

42

MULTITHREADED CLIENTS

- **Web browser**
- Uses threads to load and render portions of a web page to the user in parallel
- A client could have dozens of concurrent connections all loading in parallel
- **testFibPar.sh**
- Assignment 0 client script (GNU parallel)
- **Important benefits:**
- Several connections can be opened simultaneously
- Client: dozens of concurrent connections to the webserver all loading data in parallel

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
LR.43

43

MULTIPLE THREADS

- In Linux, threads also receive a process ID (PID)
- To display threads of a process in Linux:
- Identify parent process explicitly:
- `top -H -p <pid>`
- `htop -p <pid>`
- `ps -iT <pid>`
- Virtualbox process ~ 44 threads
- No mapping to guest # of processes/threads

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
LR.44

44

PROCESS METRICS

CPU

- `cpuUser`: CPU time in user mode
- `cpuKrn`: CPU time in kernel mode
- `cpuIdle`: CPU idle time
- `cpuIoWait`: CPU time waiting for I/O
- `cpuIntSrcv`: CPU time serving interrupts
- `cpuSoftIntSrcv`: CPU time serving soft interrupts
- `cpuNice`: CPU time executing prioritized processes
- `cpuSteal`: CPU ticks lost to virtualized guests
- `contextsw`: # of context switches
- `loadavg`: (avg # proc / 60 secs)

Disk

- `dscr`: disk sector reads
- `dsreads`: disk sector reads completed
- `dfrm`: merged adjacent disk reads
- `readtime`: time spent reading from disk
- `dsw`: disk sector writes
- `dswrites`: disk sector writes completed
- `dwm`: merged adjacent disk writes
- `writetime`: time spent writing to disk

Network

- `nbs`: network bytes sent
- `nbr`: network bytes received

45

LOAD AVERAGE

- Reported by: `top`, `htop`, `w`, `uptime`, and `/proc/loadavg`
- Updated every 5 seconds
- Average number of processes using or waiting for the CPU
- Three numbers show exponentially decaying usage for 1 minute, 5 minutes, and 15 minutes
- One minute average: exponentially decaying average
- Load average = $1 \cdot (\text{avg last minute load}) - 1/e \cdot (\text{avg load since boot})$
- 1.0 = 1-CPU core fully loaded
- 2.0 = 2-CPU cores
- 3.0 = 3-CPU cores . . .

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
LR.46

46

THREAD-LEVEL PARALLELISM

- Metric - measures degree of parallelism realized by running system, by calculating average utilization:

$$TLP = \frac{\sum_{i=1}^N i \cdot C_i}{1 - c_0}$$

- C_i - fraction of time that exactly i threads are executed
- N - maximum threads that can execute at any one time
- Web browsers found to have TLP from 1.5 to 2.5
- Clients for web browsing can utilize from 2 to 3 CPU cores
- Any more cores are redundant, and potentially wasteful
- **Measure TLP to understand how many CPUs to provision**

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
LR.47

47

MULTITHREADED SERVERS

- Multiple threads essential for servers in distributed systems
- Even on single-core machines greatly improves performance
- Take advantage of idle/blocking time
- Two designs:
 - Generate new thread for every request
 - Thread pool - pre-initialize set of threads to service requests

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
LR.48

48

SINGLE THREAD & FSM SERVERS

- Single thread server
 - A single thread handles all client requests
 - BLOCKS for I/O
 - All waiting requests are queued until thread is available
- Finite state machine
 - Server has a single thread of execution
 - I/O performing asynchronously (non-BLOCKing)
 - Server handles other requests while waiting for I/O
 - Interrupt fired with I/O completes
 - Single thread "jumps" back into context to finish request

January 26, 2023
TCCSS58: Applied Distributed Computing (Winter 2023)
School of Engineering and Technology, University of Washington - Tacoma
LS.49

49

SERVER DESIGN ALTERNATIVES

- A blocking system call implies that a thread servicing a request synchronously performs I/O
- The thread BLOCKS to wait on disk/network I/O before proceeding with request processing
- Consider the implications of these designs for responsiveness, availability, scalability. . .

| Model | Characteristics |
|----------------------|-------------------------------|
| Multithreading | Parallelism, blocking I/O |
| Single-thread | No parallelism, blocking I/O |
| Finite-state machine | Parallelism, non-blocking I/O |

January 26, 2023
TCCSS58: Applied Distributed Computing (Winter 2023)
School of Engineering and Technology, University of Washington - Tacoma
LS.50

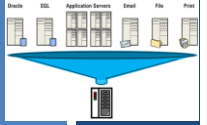
50

OBJECTIVES – 1/26

- Questions from 1/26
- Assignment 0: Cloud Computing Infrastructure Tutorial
 - New testFibService.sh script
- Assignment 1: Key/Value Store - Next Week
- Chapter 3: Processes
 - Chapter 3.1: Threads
 - Context Switches
 - Threading Models
 - Multithreaded clients/servers
 - Chapter 3.2: Virtualization
 - Chapter 3.3: Clients
 - Chapter 3.4: Servers

January 26, 2023
TCCSS58: Applied Distributed Computing (Winter 2023)
School of Engineering and Technology, University of Washington - Tacoma
LS.51

51




CH. 3.2: VIRTUALIZATION

LS.52

52

VIRTUALIZATION

- Initially introduced in the 1970s on IBM mainframe computers
- Legacy operating systems run in mainframe-based VMs
- Legacy software could be sustained by virtualizing legacy OSES
- 1970s virtualization went away as desktop/rack-based hardware became inexpensive
- Virtualization reappears in 2000s to leverage multi-core, multi-CPU processor systems
- VM-Ware virtual machines enable companies to host many virtual servers with mixed OSES on private clusters
- Cloud computing: Amazon offers VMs as-a-service (IaaS)

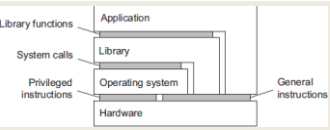


January 26, 2023
TCCSS58: Applied Distributed Computing (Winter 2023)
School of Engineering and Technology, University of Washington - Tacoma
LS.53

53

TYPES OF VIRTUALIZATION

- **Levels of Instructions:**
- **Hardware:** CPU
 - Privileged instructions
KERNEL MODE
 - General instructions
USER MODE
- **Operating system:** system calls
- **Library:** programming APIs: e.g. C/C++, C#, Java libraries
- **Application:**
- **Goal of virtualization:** mimic these interface to provide a virtual computer



January 26, 2023
TCCSS58: Applied Distributed Computing (Winter 2023)
School of Engineering and Technology, University of Washington - Tacoma
LS.54

54

TYPES OF VIRTUALIZATION - 2

- Process virtual machine**
 - Interpret instructions: (interpreters) (JavaVM) byte code → HW instructions
 - Emulate instructions: (emulators) (Wine) windows code → Linux code
- Native virtual machine monitor (VMM)**
 - Hypervisor (XEN): small OS with its own kernel
 - Provides an interface for multiple guest OSes
 - Facilitates sharing/scheduling of CPU, device I/O among many guests
 - Guest OSes require special kernel to interface w/ VMM
 - Supports **Paravirtualization** for performance boost to run code directly on the CPU
 - Type 1 hypervisor

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.55

55

TYPES OF VIRTUALIZATION - 3

- Hosted virtual machine monitor (VMM)**
 - Runs atop of hosted operating system
 - Uses host OS facilities for CPU scheduling, I/O
 - Full virtualization
 - Type 2 hypervisor
 - Virtualbox**
- Textbook: note 3.5 – good explanation of full vs. paravirtualization**
- GOAL:** run all user mode instructions directly on the CPU
- x86 instruction set has ~17 privileged user mode instructions
- Full virtualization:** scan the EXE, insert code around privileged instructions to divert control to the VMM
- Paravirtualization:** special OS kernel eliminates side effects of privileged instructions

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.56

56

EVOLUTION OF AWS VIRTUALIZATION

From <http://www.brendangregg.com/blog/2017-11-29/aws-ec2-virtualization-2017.html>

AWS EC2 Virtualization Types

| # | Tech | Type | With | VS | VS | VS | VS | VS | VS |
|---|------|---------------------|---------------------------|----|----|----|----|----|----|
| 1 | VM | Fully Emulated | | P | P | P | P | P | P |
| 2 | VM | Xen PV 3.0 | PV drivers | P | P | P | P | P | P |
| 3 | VM | Xen HVM 3.0 | PV drivers | VH | P | P | P | VS | VS |
| 4 | VM | Xen HVM 4.0.1 | PVHVM drivers | VH | P | P | P | P | VS |
| 5 | VM | Xen AWS 2013 | PVHVM + SR-IOV(net) | VH | VH | P | P | P | VS |
| 6 | VM | Xen AWS 2017 | PVHVM + SR-IOV(net, stor) | VH | VH | VH | P | P | VS |
| 7 | VM | AWS Nitro 2017 | | VH | VH | VH | VH | VH | VS |
| 8 | HW | AWS Bare Metal 2017 | | H | H | H | H | H | H |
| | HW | Bare Metal | | H | H | H | H | H | H |

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.57

57

AWS VIRTUALIZATION - 2

- Full Virtualization - Fully Emulated**
 - Never used on EC2, before CPU extensions for virtualization
 - Can boot any unmodified OS
 - Support via slow emulation, performance 2x-10x slower
- Paravirtualization: Xen PV 3.0**
 - Software: Interrupts, timers
 - Paravirtual: CPU, Network I/O, Local+Network Storage
 - Requires special OS kernels, interfaces with hypervisor for I/O
 - Performance 1.1x – 1.5x slower than “bare metal”
 - Instance store instances: 1st & 2nd generation- m1.large, m2.xlarge
- Xen HVM 3.0**
 - Hardware virtualization: **CPU, memory (CPU VT-x required)**
 - Paravirtual: network, storage
 - Software: interrupts, timers
 - EBS backed instances
 - m1, c1 instances

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.58

58

AWS VIRTUALIZATION - 3

- XEN HVM 4.0.1**
 - Hardware virtualization: CPU, memory (**CPU VT-x required**)
 - Paravirtual: network, storage, **Interrupts, timers**
- XEN AWS 2013 (diverges from opensource XEN)**
 - Provides hardware virtualization for CPU, memory, **network**
 - Paravirtual: storage, **Interrupts, timers**
 - Called Single root I/O Virtualization (SR-IOV)
 - Allows sharing single physical PCI Express device (i.e. network adapter) with multiple VMs
 - Improves VM network performance
 - 3rd & 4th generation instances (c3 family)
 - Network speeds up to 10 Gbps and 25 Gbps
- XEN AWS 2017**
 - Provides hardware virtualization for CPU, memory, network, **local disk**
 - Paravirtual: remote storage, **Interrupts, timers**
 - Introduces hardware virtualization for EBS volumes (c4 instances)
 - Instance storage hardware virtualization (x1.32xlarge, i3 family)

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.59

59

AWS VIRTUALIZATION - 4

- AWS Nitro 2017**
 - Provides hardware virtualization for CPU, memory, network, **local disk, remote disk, Interrupts, timers**
 - All aspects of virtualization enhanced with HW-level support
 - November 2017
 - Goal: provide performance indistinguishable from “bare metal”
 - 5th generation instances – c5 instances (also c5d, c5n)
 - Based on KVM hypervisor
 - Overhead around ~1%

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.60

60

OBJECTIVES - 1/26

- **Questions from 1/26**
- **Assignment 0: Cloud Computing Infrastructure Tutorial**
 - New testFibService.sh script
- **Assignment 1: Key/Value Store - Next Week**
- **Chapter 3: Processes**
 - Chapter 3.1: Threads
 - Context Switches
 - Threading Models
 - Multithreaded clients/servers
 - Chapter 3.2: Virtualization
 - **Chapter 3.3: Clients**
 - Chapter 3.4: Servers

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.61

61

CH. 3.3: CLIENTS



January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.62

62

TYPES OF CLIENTS

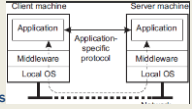
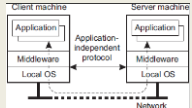
- **Thick clients**
 - Web browsers
 - Client-side scripting
 - Mobile apps
 - Multi-tier MVC apps
- **Thin clients**
 - Remote desktops/GUIs (very thin)

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.63

63

CLIENTS

- **Application specific protocol**
 - Thick clients
 - Clients maintain local data
 - Middleware (APIs)
 - Clients synchronize data with remote nodes
 - Example: shared calendar application
- **Application independent**
 - Thin clients
 - Client acts as a remote terminal
 - Provides interface to user (GUI / UI)
 - Server houses entire application stack

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.64

64

X WINDOWS

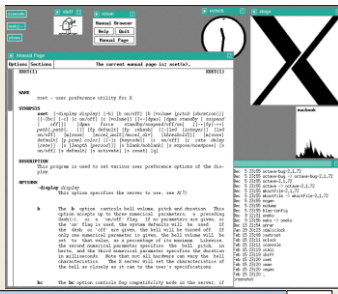
- Layered architecture to transport UI over network
- Remote desktop functionality for Linux/Unix systems
- X kernel acts as a server
 - Provides the **X protocol**: application level protocol
 - Xlib instances (client applications) exchange data and events with X kernels (servers)
 - Clients and servers on single machine → Linux GUI
 - Client and server communication transported over the network → remote Linux GUI

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.65

65

X WINDOWS - 2

- **Window manager:**
 - Application running atop of X-windows which provides flair
 - Many variants
 - Without X windows is quite bland



January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.66

66

- Layered architecture
- X-kernel: low level interface/APIs for controlling screen, capturing keyboard and mouse events (X window Server)
- Provided on Linux as Xlib
- Provides network enabled GUI
- Layering allows for use for custom window managers

January 26, 2023 TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma L8.67

67

EXAMPLE: VNC SERVER

- How to Install VNC server on Ubuntu EC2 Instance VM:
 - sudo apt-get update
 - # ubuntu 16.04
 - sudo apt-get install ubuntu-desktop
 - sudo apt-get install gnome-panel gnome-settings-daemon metacity nautilus gnome-terminal
 - # on ubuntu ≥ 18.04
 - sudo apt install xfce4 xfce4-goodies
 - sudo apt-get install tightvncserver # both
 - Start VNC server to create initial config file
 - vncserver :1

January 26, 2023 TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma L8.68

68

EXAMPLE: VNC SERVER - UBUNTU 16.04

- On the VM: edit config file: nano ~/.vnc/xstartup
- Replace contents as below (Ubuntu 16.04):

```
#!/bin/sh
export XKL_XMODMAP_DISABLE=1
unset SESSION_MANAGER
unset DBUS_SESSION_BUS_ADDRESS

[ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup
[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources
xsetroot -solid grey

vncconfig -iconic &
gnome-panel &
gnome-settings-daemon &
metacity &
nautilus &
gnome-terminal &
```

January 26, 2023 TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma L8.69

69

EXAMPLE: VNC SERVER - UBUNTU 18.04

- On the VM:
 - Edit config file: nano ~/.vnc/xstartup
 - Replace contents as below (Ubuntu 18.04):

```
#!/bin/bash
xrdb $HOME/.Xresources
startxfce4 &
```

January 26, 2023 TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma L8.70

70

EXAMPLE: VNC SERVER - 3

- On the VM: reload config by restarting server
- vncserver -kill :1
- vncserver :1
- Open port 22 & 5901 in EC2 security group:

January 26, 2023 TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma L8.71

71

EXAMPLE: VNC CLIENT

- On the client (e.g. laptop):
 - Create SSH connection to securely forward port 5901 on the EC2 instance to your localhost port 5901
 - This way your VNC client doesn't need an SSH key

```
ssh -i <ssh-keyfile> -L 5901:127.0.0.1:5901 -N -f -l <username> <EC2-instance ip_address>
```

- For example:

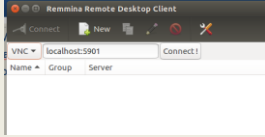

```
ssh -i mykey.pem -L 5901:127.0.0.1:5901 -N -f -l ubuntu 52.111.202.44
```

January 26, 2023 TCSS558: Applied Distributed Computing [Winter 2023] School of Engineering and Technology, University of Washington - Tacoma L8.72

72

EXAMPLE: VNC CLIENT - 2

- On the client (e.g. laptop):
- Use a VNC Client to connect
- Remmina is provided by default in Ubuntu
- Can "google" for many others
- Remmina login:
- Chose "VNC" protocol
- Log into "localhost:5901"

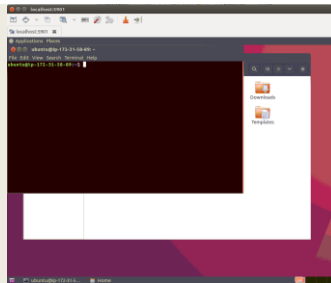


January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
LR.73

73

REMOTE COMPUTER IN THE CLOUD

- EC2 instance with a GUI...!!!



January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
LR.74

74

THIN CLIENTS

- Thin clients
 - X windows protocol
 - A variety of other remote desktop protocols exist:

Remote desktop protocols include the following:

- Apple Remote Desktop Protocol (ARD) – Original protocol for Apple Remote Desktop on macOS machines.
- Applance Link Protocol (ALP) – a Sun Microsystems-specific protocol featuring audio (play and record), remote printing, remote USB, accelerated video
- HP Remote Graphics Software (RGS) – a proprietary protocol designed by Hewlett-Packard specifically for high end workstation remoting and collaboration.
- Independent Computing Architecture (ICA) – a proprietary protocol designed by Citrix Systems
- NX technology (NoMachine NX) – Cross platform protocol featuring audio, video, remote printing, remote USB, H264-enabled.
- PC-over-IP (PCoIP) – a proprietary protocol used by VMware (licensed from Teradici)^[7]
- Remote Desktop Protocol (RDP) – a Windows-specific protocol featuring audio and remote printing
- Remote Frame Buffer Protocol (RFB) – A framebuffer level cross-platform protocol that VNC is based on.
- SPICE (Simple Protocol for Independent Computing Environments) – remote-display system built for virtual environments by Qumranet, now Red Hat
- Splashtop – a high performance remote desktop protocol developed by Splashtop, fully optimized for hardware (H.264) including Intel / AMD chipsets, NVIDIA of media codecs. Splashtop can deliver high frame rates with low latency, and also low power consumption.
- X Window System (X11) – a well-established cross-platform protocol mainly used for displaying local applications; X11 is network transparent

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
LR.75

75

THIN CLIENTS - 2

- Applications should separate application logic from UI
- When application logic and UI interaction are tightly coupled many requests get sent to X kernel
- Client must wait for response
- Synchronous behavior and app-to-UI coupling adversely affects performance of WAN / Internet
- **Protocol optimizations:** reduce bandwidth by shrinking size of X protocol messages
 - Send only differences between messages with same identifier
 - Optimizations enable connections with 9600 kbps

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
LR.76

76

THIN CLIENTS - 3

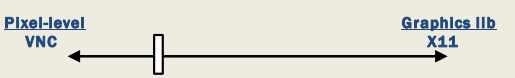
- Virtual network computing (VNC)
- Send display over the network at the pixel level (instead of X lib events)
- Reduce pixel encodings to save bandwidth – fewer colors
- Pixel-based approaches loose application semantics
- Can transport any GUI this way
- **THINC**- hybrid approach
 - Send video device driver commands over network
 - More powerful than pixel based operations
 - Less powerful compared to protocols such as X

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
LR.77

77

TRADEOFFS: ABSTRACTION OF REMOTE DISPLAY PROTOCOLS

- Tradeoff space: abstraction level of remote display protocols



The diagram shows a horizontal double-headed arrow representing a tradeoff space. On the left end, it is labeled 'Pixel-level VNC'. On the right end, it is labeled 'Graphics lib X11'. A vertical bar is positioned in the middle of the arrow, indicating a specific point or range within this tradeoff space.

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
LR.78

78

TRADEOFFS: ABSTRACTION OF REMOTE DISPLAY PROTOCOLS

Tradeoff space: abstraction level of remote display protocols

Pixel-level
VNC

Graphics IIb
X11

- Generic – no app context
- Graphics data
- Higher network bandwidth
- Fewer colors
- Utilize graphics compression
- More network traffic

- Application context is available
- UI data/operations
- Lower network bandwidth
- More colors

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.79

79

CLIENT ROLES IN PROVIDING DISTRIBUTION TRANSPARENCY

- Clients help enable distribution transparency of servers
- Replication transparency
 - Client aggregates responses from multiple servers
 - Only the client knows of replicas

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.80

80

CLIENT ROLES IN PROVIDING DISTRIBUTION TRANSPARENCY - 2

- Location/relocation/migration transparency
 - Harness convenient naming system to allow client to infer new locations
 - Server inform client of moves / Client reconnects to new endpoint
 - Client hides network address of server, and reconnects as needed
 - May involve temporary loss in performance
- Replication transparency
 - Client aggregates responses from multiple servers
- Failure transparency
 - Client retries, or maps to another server, or uses cached data
- Concurrency transparency
 - Transaction servers abstract coordination of multithreading

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.81

81

OBJECTIVES – 1/26

- **Questions from 1/26**
- Assignment 0: Cloud Computing Infrastructure Tutorial
 - New testFibService.sh script
- Assignment 1: Key/Value Store - Next Week
- Chapter 3: Processes
 - Chapter 3.1: Threads
 - Context Switches
 - Threading Models
 - Multithreaded clients/servers
 - Chapter 3.2: Virtualization
 - Chapter 3.3: Clients
 - **Chapter 3.4: Servers**

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.82

82

CH. 3.4: SERVERS

L8.83

83

SERVERS

- Cloud & Distributed Systems – rely on **Linux**
- <http://www.zdnet.com/article/it-runs-on-the-cloud-and-the-cloud-runs-on-linux-any-questions/>
- IT is moving to the cloud. And, what powers the cloud?
 - **Linux**
- Uptime Institute survey - 1,000 IT executives (2016)
 - 50% of IT executives – plan to migrate majority of IT workloads to off-premise to cloud or colocation sites
 - 23% expect the shift in 2017, 70% by 2020...
- Docker on Windows / Mac OS X
 - Based on **Linux**
 - Mac: Hyperkit Linux VM
 - Windows: Hyper-V Linux VM

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.84

84

SERVERS - 2

- Servers implement a specific service for a collection of clients
- Servers wait for incoming requests, and respond accordingly
- Server types**
 - Iterative:** immediately handle client requests
 - Concurrent:** Pass client request to separate thread
- Multithreaded servers are concurrent servers
 - E.g. Apache Tomcat
- Alternative:** fork a new process for each incoming request
- Hybrid:** mix the use of multiple processes with thread pools

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.85

85

END POINTS

- Clients connect to servers via:
 - IP Address** and **Port Number**
- How do ports get assigned?
 - Many protocols support "default" port numbers
 - Client must find IP address(es) of servers
 - A single server often hosts multiple end points (servers/services)
 - When designing new TCP client/servers must be careful not to repurpose ports already commonly used by others

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.86

86

COMMON PORTS

packetlife.net

| TCP/UDP Port Numbers | | | |
|----------------------|--------------------|-------------|-------------------|
| 7 | Echo | 554 | RTSP |
| 19 | Chargen | 546-547 | DHCPv6 |
| 20-21 | FTP | 560 | monitor |
| 22 | SSH/SFTP | 563 | SMTP over SSL |
| 23 | Telnet | 587 | SMTP |
| 25 | SMTP | 591 | FileMaker |
| 42 | WINS Replication | 593 | Microsoft DCOM |
| 43 | WINS | 632 | Internet Printing |
| 49 | TACACS | 636 | LDAP over SSL |
| 53 | DNS | 639 | MSDP (PM) |
| 67-68 | DHCP/BOOTP | 646 | LDP (MPLS) |
| 69 | TFTP | 691 | MS Exchange |
| 70 | Gopher | 860 | ISCSI |
| 79 | Finger | 873 | rsync |
| 80 | HTTP | 902 | VMware Server |
| 88 | Kerberos | 989-990 | FTP over SSL |
| 102 | MS Exchange | 993 | IMAP4 over SSL |
| 110 | POP3 | 995 | POP3 over SSL |
| 133 | Ident | 1025 | Microsoft RPC |
| 139 | NNTP (Usenet) | 1026-1029 | Windows Messenger |
| 143 | IMAP4 | 1080 | SOCKS Proxy |
| 161-162 | SNMP | 1241 | Nessus |
| 177 | XDMCP | 1311 | Dist OpenManage |
| 178 | BDP | 1317 | IMAP |
| 2745 | Segnet | 2745 | Segnet |
| 2967 | Symantec AV | 3050 | Interbase DB |
| 3074 | SOX Live | 3124 | HTTP Proxy |
| 3127 | MSBlast | 3128 | HTTP Proxy |
| 3223 | CLBP | 3260 | ISCSI Target |
| 3306 | MySQL | 3389 | Terminal Server |
| 3689 | iTunes | 3690 | Subversion |
| 3724 | Microsoft Internet | 3784-3785 | Ventrilo |
| 4333 | mSQL | 4444 | Center |
| 4644 | Center | 4664 | Google Desktop |
| 4672 | Shake | 4899 | Radmin |
| 5001 | Slingbox | 5000 | UPnP |
| 5001 | iperf | 5004-5005 | RTP |
| 5004-5005 | RTP | 5050 | Yahoo! Messenger |
| 5060 | SIP | 5060 | SIP |
| 5150 | BitTorrent | 5150 | BitTorrent |
| 6091-6901 | Windows Live | 6970 | Quicktime |
| 7212 | GhostSurf | 8000 | Internet Radio |
| 8080 | HTTP Proxy | 8086-8087 | Kaspersky AV |
| 8118 | Private | 8200 | VMware Server |
| 8500 | Adobe ColdFusion | 8767 | TeamSpeak |
| 8866 | Scale B | 9100 | HP JetDirect |
| 9101-9103 | Beatsite | 9119 | MSRPC |
| 9800 | WebDAV | 9898 | Sabber |
| 9988 | BotSploit | 9999 | Urchin |
| 10000 | Webmin | 12035-12036 | Second Life |
| 10000 | BackupExec | 12345 | NetBus |
| 10113-10116 | NetIQ | 13720-13721 | NetBackup |
| 11371 | OpenPGP | 14567 | MailServer |

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.88

87

TYPES OF SERVERS

- Daemon server**
 - Example: NTP server
- Superserver**
- Stateless server**
 - Example: Apache server
- Stateful server**
- Object servers**
- EJB servers**

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.88

88

NTP EXAMPLE

- Daemon servers**
 - Run locally on Linux
 - Track current server end points (outside servers)
 - Example: network time protocol (ntp) daemon
 - Listen locally on specific port (ntp is 123)
 - Daemons routes local client traffic to the configured endpoint servers
 - University of Washington: time.u.washington.edu
 - Example "ntpq -p"
 - Queries local ntp daemon, routes traffic to configured server(s)

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.89

89

SUPERSERVER

- Linux inetd / xinetd**
 - Single superserver
 - Extended internet service daemon
 - Not installed by default on Ubuntu
 - Intended for use on server machines
 - Used to configure box as a server for multiple internet services
 - E.g. ftp, pop, telnet
 - inetd daemon responds to multiple endpoints for multiple services
 - Requests fork a process to run required executable program
- Check what ports you're listening on:
 - sudo netstat -tap | grep LISTEN

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
L8.90

90

INTERRUPTING A SERVER

- Server design issue:
 - Active client/server communication is taking place over a port
 - How can the server / data transfer protocol support interruption?
- Consider transferring a 1 GB image, how do you pass a unrelated message in this stream?
 1. **Out-of-band** data: special messages sent in-stream to support interrupting the server (*TCP urgent data*)
 2. Use a separate connection (different port) for admin control info
- Example: sftp secure file transfer protocol
 - Once a file transfer is started, can't be stopped easily
 - Must kill the client and/or server

January 26, 2023 TCCS558: Applied Distributed Computing (Winter 2023)
School of Engineering and Technology, University of Washington - Tacoma L8.91

91

STATELESS SERVERS

- Data about state of clients is not stored
- Example: web application servers are typically stateless
 - Also function-as-a-service (FaaS) platforms
- Many servers maintain information on clients (e.g. log files)
- Loss of stateless data doesn't disrupt server availability
 - Losing log files typically has minimal consequences
- **Soft state**: server maintains state on the client for a limited time (*to support sessions*)
- Soft state information expires and is deleted

January 26, 2023 TCCS558: Applied Distributed Computing (Winter 2023)
School of Engineering and Technology, University of Washington - Tacoma L8.92

92

STATEFUL SERVERS

- Maintain persistent information about clients
- Information must be explicitly deleted by the server
- Example:
File server - allows clients to keep local file copies for RW
- Server tracks client file permissions and most recent versions
 - Table of (client, file) entries
- If server crashes data must be recovered
- Entire state before a crash must be restored
- Fault tolerance - *Ch. 8*

January 26, 2023 TCCS558: Applied Distributed Computing (Winter 2023)
School of Engineering and Technology, University of Washington - Tacoma L8.93

93

STATEFUL SERVERS - 2

- Session state
 - Tracks series of operations by a single user
 - Maintained temporarily, not indefinitely
 - Often retained for multi-tier client server applications
 - Minimal consequence if session state is lost
 - Clients must start over, reinitialize sessions
- Permanent state
 - Customer information, software keys
- Client-side cookies
 - When servers don't maintain client state, clients can store state locally in "cookies"
 - Cookies are not executable, simply client-side data

January 26, 2023 TCCS558: Applied Distributed Computing (Winter 2023)
School of Engineering and Technology, University of Washington - Tacoma L8.94

94

OBJECT SERVERS

- **OBJECTIVE**: Host objects and enable remote client access
- Do not provide a specific service
 - Do nothing if there are no objects to host
- Support adding/removing hosted objects
- Provide a home where objects live
- Objects, *themselves*, provide "services"
- Object parts
 - State data
 - Code (methods, etc.)
- **Transient object(s)**
 - Objects with limited lifetime (< server)
 - Created at first invocation, destroyed when no longer used (i.e. no clients remain "bound").
 - Disadvantage: initialization may be expensive
 - Alternative: preinitialize and retain objects on server start-up

January 26, 2023 TCCS558: Applied Distributed Computing (Winter 2023)
School of Engineering and Technology, University of Washington - Tacoma L8.95

95

OBJECT SERVERS - 2

- **Should object servers isolate memory for object instances?**
 - Share neither code nor data
 - May be necessary if objects couple data and implementation
- Object server threading designs:
 - Single thread of control for object server
 - One thread for each object
 - Servers use separate thread for client requests
- Threads created on demand **vs.** Server maintains pool of threads
- **What are the tradeoffs for creating server threads on demand vs. using a thread pool?**

January 26, 2023 TCCS558: Applied Distributed Computing (Winter 2023)
School of Engineering and Technology, University of Washington - Tacoma L8.96

96

EJB – ENTERPRISE JAVA BEANS

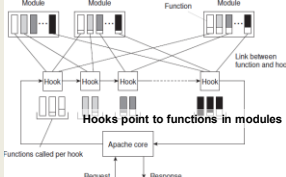
- EJB- specialized Java object hosted by a EJB web container
- 4 types: stateless, stateful, entity, and message-driven beans
- Provides "middleware" standard (framework) for implementing back-ends of enterprise applications
- EJB web application containers integrate support for:
 - Transaction processing
 - Persistence
 - Concurrency
 - Event-driven programming
 - Asynchronous method invocation
 - Job scheduling
 - Naming and discovery services (JNDI)
 - Interprocess communication
 - Security
 - Software component deployment to an application server

January 26, 2023
TCCS558: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
LS.97

97

APACHE WEB SERVER

- Highly configurable, extensible, platform independent
- Supports TCP HTTP protocol communication
- Uses hooks – placeholders for group of functions
- Requests processed in phases by hooks
- Many hooks:
 - Translate a URL
 - Write info to log
 - Check client ID
 - Check access rights
- Hooks processed in order enforcing flow-of-control
- Functions in replaceable modules

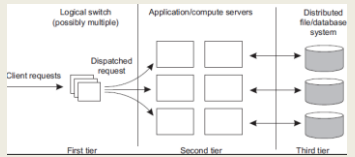


January 26, 2023
TCCS558: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
LS.98

98

SERVER CLUSTERS

- Hosted across an LAN or WAN
- Collection of interconnected machines
- Can be organized in tiers:
 - Web server → app server → DB server
 - App and DB server sometimes integrated



January 26, 2023
TCCS558: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
LS.99

99

LAN REQUEST DISPATCHING

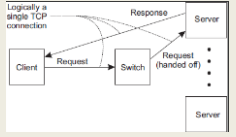
- Front end of three tier architecture (logical switch) provides distribution transparency – hides multiple servers
- Transport-layer switches: switch accepts TCP connection requests, hands off to a server
 - Example: hardware load balancer (F5 networks – Seattle)
 - HW Load balancer - OSI layers 4-7
- Network-address-translation (NAT) approach:
 - All requests pass through switch
 - Switch sits in the middle of the client/server TCP connection
 - Maps (rewrites) source and destination addresses
- Connection hand-off approach:
 - **TCP Handoff:** switch hands of connection to a selected server

January 26, 2023
TCCS558: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
LS.100

100

LAN REQUEST DISPATCHING - 2

- Who is the best server to handle the request?
- Switch plays important role in distributing requests
- Implements load balancing
- **Round-robin** – routes client requests to servers in a looping fashion
- **Transport-level** – route client requests based on TCP port number
- **Content-aware request distribution** – route requests based on inspecting data payload and determining which server node should process the request



January 26, 2023
TCCS558: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
LS.101

101

WIDE AREA CLUSTERS

- Deployed across the internet
- Leverage resource/infrastructure from Internet Service Providers (ISPs)
- Cloud computing simplifies building WAN clusters
- Resource from a single cloud provider can be combined to form a cluster
- For deploying a cloud-based cluster (WAN), what are the implications of deploying nodes to:
 - (1) a single availability zone (e.g. us-east-1e)?
 - (2) across multiple availability zones?

January 26, 2023
TCCS558: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
LS.102

102

WAN REQUEST DISPATCHING

- Goal: minimize network latency using WANs (e.g. Internet)
- Send requests to nearby servers
- Request dispatcher: routes requests to nearby server
- Example:** Domain Name System
 - Hierarchical decentralized naming system
- Linux: find your DNS servers:


```
# Find you device name of interest
nmcli dev
# Show device configuration
nmcli device show <device name>
```

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
LS.103

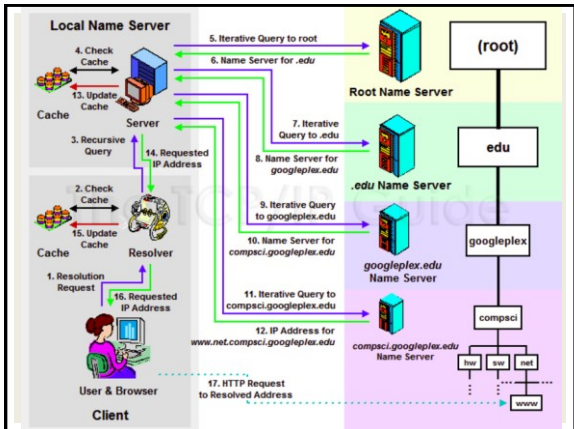
103

DNS LOOKUP

- First query local server(s) for address
- Typically there are (2) local DNS servers
 - One is backup
- Hostname may be cached at local DNS server
 - E.g. www.google.com
- If not found, local DNS server routes to other servers
- Routing based on components of the hostname
- DNS servers down the chain mask the client IP, and use the originating DNS server IP to identify a local host
- Weakness:** client may be far from DNS server used. Resolved hostname is close to DNS server, but not necessarily close to the client

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
LS.104

104



105

DNS: LINUX COMMANDS

- `nslookup <ip addr / hostname>`
- Name server lookup - translates hostname or IP to the inverse
- `traceroute <ip addr / hostname>`
- Traces network path to destination
- By default, output is limited to 30 hops, can be increased

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
LS.106

106

DNS EXAMPLE - WAN DISPATCHING

- Ping www.google.com in WA from wireless network:
 - `nslookup`: 6 alternate addresses returned, choose (74.125.28.147)
 - Ping 74.125.28.147: Average RTT = **22.458 ms (11 attempts, 22 hops)**
- Ping www.google.com in VA (us-east-1) from EC2 instance:
 - `nslookup`: 1 address returned, choose 172.217.9.196
 - Ping 172.217.9.196: Average RTT = 1.278 ms (11 attempts, 13 hops)
- From VA EC2 instance, ping WA [www.google](http://www.google.com) server
 - Ping 74.125.28.147: Average RTT 62.349ms (11 attempts, 27 hops)
 - Pinging the WA-local server is ~60x slower from VA
- From local wireless network, ping VA us-east-1 google :
 - Ping 172.217.9.196: Average RTT=81.637ms (11 attempts, 15 hops)

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
LS.107

107

DNS EXAMPLE - WAN DISPATCHING

- Ping www.google.com in WA from wireless network:
 - `nslookup`: 6 alternate addresses returned, choose (74.125.28.147)

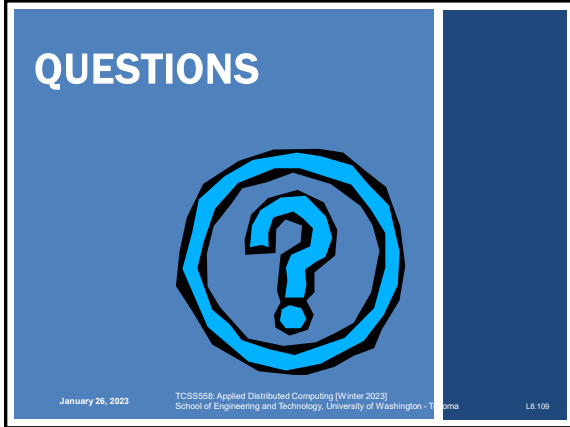
Latency to ping VA server in WA: ~3.63x
 WA client: local-google 22.458ms to VA-google 81.637ms

Latency to ping WA server in VA: ~48.7x
 VA client: local-google 1.278ms to WA-google 62.349!

- From local wireless network, ping VA us-east-1 google :
 - Ping 172.217.9.196: Average RTT=81.637ms (11 attempts, 15 hops)

January 26, 2023
TCCSS58: Applied Distributed Computing [Winter 2023]
School of Engineering and Technology, University of Washington - Tacoma
LS.108

108



109