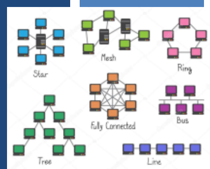
TCSS 558: APPLIED DISTRIBUTED COMPUTING

Processes: Servers - II

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



1

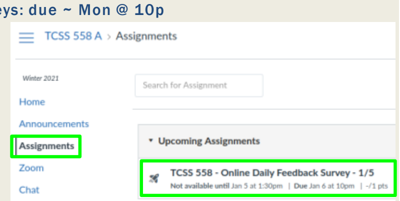
OBJECTIVES – 2/6

- Questions from 2/1
- Assignment 1: Cloud Computing Infrastructure Tutorial
- Assignment 2: Key/Value Store
  - Java Maven project template files posted
- Midterm Thursday February 8
- Chapter 3: Processes
  - Chapter 3.4: Servers
- Midterm Thursday February 8
  - 2nd hour – practice midterm questions

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

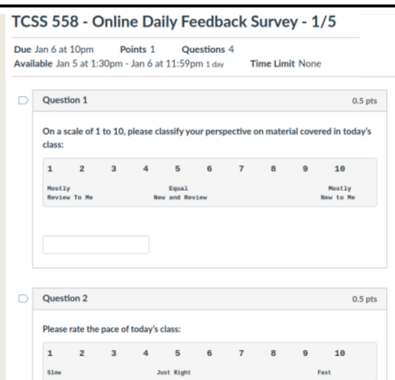


3

SURVEY QUESTIONS

- Survey has two questions:
- Be sure to add your questions about the previous class to the ***second question***
- 1st question: After today's class, comment on any new concepts that you learned about?
  - *Have been getting questions here…*
- 2nd question: After today's class, what point(s) remain least clear to you?
  - **>> Please add questions HERE**

4



5

MATERIAL / PACE

- Please classify your perspective on material covered in today's class (24 respondents):
- 1-mostly review, 5-equal new/review, 10-mostly new
- **Average – 6.04** (↓ - *previous 6.60*)
- Please rate the pace of today's class:
- 1-slow, 5-just right, 10-fast
- **Average – 5.52** (↑ - *previous 5.80*)

6

FEEDBACK FROM 2/1

- ***Consider a stateless server that doesn't maintain a soft state. In this scenario, what is the use of maintaining cache in stateless server? Is it even applicable? Can you please provide an example?***
  - An application data cache may not be used to store session state data since it is transient – but cached data can help a server in many ways
  - A server can cache results of common operations and database queries for immediate recall to improve performance for common / repeated requests.
  - Database servers automatically cache recent query results. For example: 'select count(*) from table;'
  - Frequently use memory pages from shared libraries and application program code can be cached in Level 1, Level 2, and Level 3 CPU caches to speed-up program performance of the server
  - Linux uses a virtual memory subsystem, and caching memory page translations for commonly accessed virtual to physical page address translations can increase performance of the server

February 6, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L10.7

7

FEEDBACK - 2

- ***Can you please provide an example of a well known application that uses object based servers?***
- *"Distributed objects were popular in the late 1990s and early 2000s, but have since fallen out of favor.", from:* https://martinfowler.com/articles/distributed-objects-microservices.html
- Visualforce Remote Objects provide client proxy objects (in a web browser) for basic data manipulation operations on sObjects hosted on a remote server.
- Every Salesforce sObject corresponds to a database table where it is stored

February 6, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L10.8

8

FEEDBACK - 3

- ***Consider a scenario where two people using the same debit card have made an online purchase at the same time to the extent of a milli second. The total amount available is 30$. Person A makes a purchase of 30$ and Person B makes a purchase of 25$. In this scenario we may look at payment server as handling multiple concurrent requests. Can you please provide your inputs on how an effective payment server should handle this scenario?***
- We will discuss a solution to this sort of problem known as total-ordered multicasting using logical clocks in Chapter 6

February 6, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L10.9

9

OBJECTIVES – 2/6

- Questions from 2/1
- **Assignment 1: Cloud Computing Infrastructure Tutorial**
- Assignment 2: Key/Value Store
  - Java Maven project template files posted
- Midterm Thursday February 8
- Chapter 3: Processes
  - Chapter 3.4: Servers
- Midterm Thursday February 8
  - 2nd hour – practice midterm questions

February 6, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L10.10

10

AWS CLOUD CREDITS UPDATE

- We have received AWS CLOUD CREDITS for TCSS 558 – W2024
- Credits will be provided by email request
  - Please include: 12-digit AWS account ID, and AWS account email
- Credits will first be provided for students not in F'23 TCSS562
- Request codes by sending an email with the subject: "**AWS CREDIT REQUEST**" to **wlloyd@uw.edu**
- Codes can also be obtained in person (or zoom), in the class, during the breaks, after class, during office hours, by appt
- Credit codes are carefully exchanged, and not shared by IM
- For students ***unable*** to create a standard AWS account: Please contact instructor by email - *Instructor will work to create hosted IAM user account*

February 6, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L10.11

11

ASSIGNMENT 1

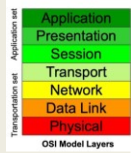
- ***Preparing for Assignment 1: Intro to Cloud Computing Infrastructure and Load Balancing***
  - Establish AWS Account - Standard account
- Now posted:
  - Task 0 - Establish local Linux/Ubuntu environment
  - Task 1 –AWS account setup, obtain user credentials
  - Task 2 – Intro to: Amazon EC2 & Docker: create Dockerfile for Apache Tomcat
  - Task 3 – Create Dockerfile for haproxy (software load balancer)
  - Task 4 – Working with Docker-Machine
  - Task 5 – Submit Results of testing alternate server configs

February 6, 2024	TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma	L10.12

12

TESTING CONNECTIVITY TO SERVER (PG 16-18)

- testFibPar.sh script is a parallel test script
- Orchestrates multiple threads on client to invoke server multiple times in parallel
- To simplify coordination of parallel service calls in BASH, testFibPar.sh script ignores errors !!!
- To help test client-to-server connectivity, there is also a testFibService.sh script that supports 3 tests
- TEST 1: Network layer test
  - Ping (ICMP)
- TEST 2: Transport layer test
  - TCP: telnet (TCP Port 8080) – security group (fw) test
- TEST 3: Application layer test
  - HTTP REST – web service test



February 6, 2024 | TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L10.13

13

OBJECTIVES – 2/6

- Questions from 2/1
- Assignment 1: Cloud Computing Infrastructure Tutorial
- **Assignment 2: Key/Value Store**
  - **Java Maven project template files posted**
- Midterm Thursday February 8
  - 2nd hour - Tuesday February 7 – practice midterm questions
- Chapter 3: Processes
  - Chapter 3.4: Servers
- Midterm Thursday February 8
  - 2nd hour – practice midterm questions

February 6, 2024 | TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L10.14

14

ASSIGNMENT 2

- Multi-protocol TCP/UDP/RMI Key Value Store
- Implement a "GenericNode" where the application can be launched to assume the role of a client or server for a Key/Value Store data store
- Recommended in Java (11)
- Client node program interacts with server node to put, get, delete, or list items in a key/value store
- Multi-threaded or single-threaded server

February 6, 2024 | TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L10.15

15

OBJECTIVES – 2/6

- Questions from 2/1
- Assignment 1: Cloud Computing Infrastructure Tutorial
- Assignment 2: Key/Value Store
  - Java Maven project template files posted
- **Midterm Thursday February 8**
- Chapter 3: Processes
  - Chapter 3.4: Servers
- Midterm Thursday February 8
  - 2nd hour – practice midterm questions

February 6, 2024 | TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L10.16

16

OBJECTIVES – 2/6

- Questions from 2/1
- Assignment 1: Cloud Computing Infrastructure Tutorial
- Assignment 2: Key/Value Store
  - Java Maven project template files posted
- **Midterm Thursday February 8**
- Chapter 3: Processes
  - Chapter 3.4: Servers
- Midterm Thursday February 8
  - 2nd hour – practice midterm questions

February 6, 2024 | TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L10.17

17

L10.18

18

TYPES OF SERVERS

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

19

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

20

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?**

21

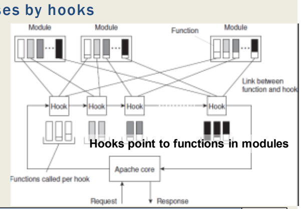
EJB – ENTERPRISE JAVA BEANS

- EJB- specialized Java object hosted by a EJB web container
- 4 types: Session (**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

22

APACHE WEB SERVER

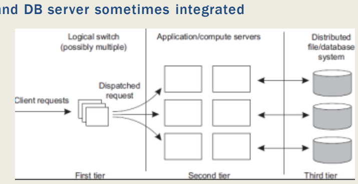
- Highly configurable, extensible, platform independent
- Supports TCP HTTP protocol communication
- Uses hooks – placeholders for group of functions
- Requests processed in phases by hooks
- Many hooks:
  - Translate a URL
  - Write info to log
  - Check client ID
  - Check access rights
- Hooks processed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



23

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



24

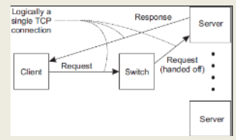
LAN REQUEST DISPATCHING

- 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

25

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



26

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?

27

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:

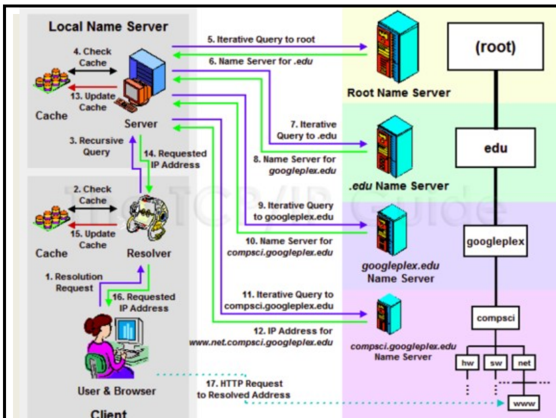
```
# Find you device name of interest
nmcli dev
# Show device configuration
nmcli device show <device name>
```


28

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*

29



30

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

February 6, 2024 | TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L10.31

31

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* 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)

February 6, 2024 | TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L10.32

32

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)

February 6, 2024 | TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L10.33

33

WE WILL RETURN AT 5:00PM

34

OBJECTIVES – 2/6

- Questions from 2/1
- Assignment 1: Cloud Computing Infrastructure Tutorial
- Assignment 2: Key/Value Store
  - Java Maven project template files posted
- Midterm Thursday February 8
  - 2nd hour - Tuesday February 7 – practice midterm questions
- Chapter 3: Processes
  - Chapter 3.4: Servers
- Midterm Thursday February 8
  - **2nd hour – practice midterm questions**

February 6, 2024 | TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L10.35

35

PRACTICE MIDTERM

- Envisioned as a class activity

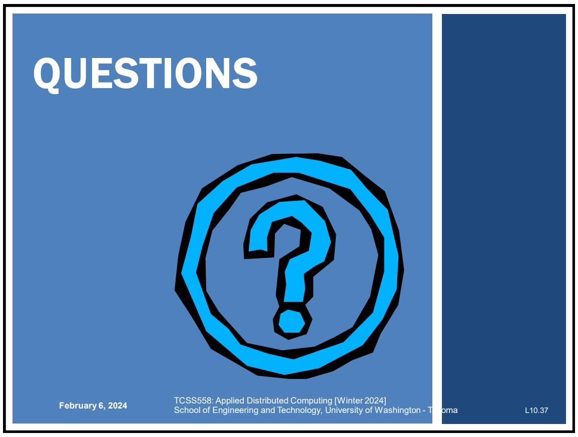
- Form groups of 1-3 in class or via Zoom breakout rooms
- Each group works on 2 assigned questions
- Group submits PDF solution by end of day today

- Instructor shares solutions provided from the class submission via Canvas by Wednesday morning
  - Subject to updates for late submissions

- Outcome:
  All students have access to solutions for review and practice
- Class Activity:
  https://canvas.uw.edu/files/115846587/

February 6, 2024 | TCSS558: Applied Distributed Computing [Winter 2024] School of Engineering and Technology, University of Washington - Tacoma | L10.36

36


QUESTIONS
February 6, 2024
TCSS558: Applied Distributed Computing [Winter 2024]
School of Engineering and Technology, University of Washington - Tacoma
L10.37

37