Wisa Terhune-Praphruettam

CS435 Distributed Systems

Study Log

11/11/2020

**Top 20 Words:**

1. **Distributed System:**

Distributed system comprises of masking important backend services between different computers while enables the appearance as a single entity to the end users. By making the backend services transparent, the user can focus on interacting with the application through an interface. Such hidden backend services can include communication between different computers at the process levels.

1. **Processes:**

Processes are instructions that a computer executes using the Central Processing Unit (CPU). Each process is identified with a unique ID. The CPU will execute a process at a time and in order, so they are very resource intensive. Some computers will come with multiple CPUs to handle running multiple processes.

1. **Synchronous Calls:**

Synchronous calls do not emulate multi-tasking. It represents the communication calls between the client and server. Synchronous calls are better on a local level (such as process communications between a local device) because the client will make a call to the server and will *wait* for a response from the server. The response time efficiency becomes a problem at a larger distributed system, especially at an enterprise scale, because client-server calls can happen between distant geographical locations and the waiting time for a response can be longer.

1. **Asynchronous Calls:**

Asynchronous calls are complex than synchronous calls and do emulate multi-tasking communication calls between different clients and servers. Unlike synchronous calls, asynchronous calls must mimic non-blocking behaviors and able to handle server interruptions to pass the client a response when the request has been fulfilled.

1. **Inter-process Communication:**

Inter-process communication represents the exchanges of process communication between systems, either locally or via the network. The layout is less complex when processes are communicating with each other at a local level. This is because for local machines, processes can access shared memory. On the other hand, it is difficult for processes to run efficiently via the network because due to the lack of shared memory.

1. **Middleware:**

Middleware helps to alleviate a communication issue regarding inter-process communication between different application and network layers. A middleware is presented with a consistent user interface across multiple devices so that the user does not have to worry about the back-end service calls for requests. This can also include process coordination between messages using synchronous calls or asynchronous calls to receive and retrieve important information from the server.

1. **Event-Based Architecture:**

The event-based architecture is used to describe a communication style between the application, middleware, and the server. Typically, this relationship is summarized as “publisher/subscriber” relationship. This is because the server will “announce” an update. Any client that decided to listen to this announcement will receive the update message. This architecture style offers a great decoupling benefit. The server has less work to deliver the update since it does not have to worry about keeping track of the client (subscribers).

1. **User Datagram Protocol:**

User Datagram Protocol, UDP, is an Internet communication protocol involved in transmitting data between the clients, servers, and other devices in the network. The data is sent directly to a specific device. There is time efficiency with messages but UDP does not do the extra work to check if packets of messages have been received by the destination. This makes sending important information less reliable and can create frustrations for the destination location since it has to wait for a dataset. In addition, this can cause security issues because the messages can be intercepted by another party and the sender would not check or care at all.

1. **Transfer Communication Protocol:**

Transfer Communication Protocol, TCP, is another Internet communication protocol for sending and receiving messages across the network. It is different than UDP because it addresses reliability issues and has better message delivery. Before sending messages to a destination, TCP will establish a connection and will maintain the connection until the destination has received that message. This type of transmission checks the status of the message deliveries and reduces transmission errors. If the messages are not sent properly, then TCP can resend the messages again.

1. **Layered Architecture:**

Layered architecture provides a big overview of communication between the user and the server. As users might view a network as one big structure, the network architecture is multiple subsets interacting with each other. Within a layered architecture, the structure is split into multiple categories. Such categories can include the UI level (where users interact with most of the time), the processing level, and the data level. An example of this layered architecture would be our simple client/server assignments. Some layered architectures are considered thinner client while some are considered thicker client. In a thin client layered architecture, it is easier to do offline updates, but we are sending each object fields separately to the server. In contrast, a thick client sends all of the object fields to the server at the same time, but it is not smooth to do offline updates.

1. **Application Programming Interface:**

Application Programming Interface is known as API. It is important in technology because it acts as an abstraction layer for the users to interact with the applications. There can also be language rules defined in the API. For instance, in the Java API, the interface (for example via intelliJ or Eclipse) has language syntax rules and libraries for users to use. Such items can include syntax to print out characters, create a list, sort a list, make random numbers, etc. when interacting with this abstraction layer represented by the API, the user would not need to worry about the behind-the-scenes interactions and code executions.

1. **Dynamic Host Configuration Protocol:**

DHCP is shorten for Dynamic Host Configuration Protocol. It is a protocol that is heavily involved with designating IP addresses to various devices across networks. The DHCP is active when a client calls the Internet and it will assign an active IP address to the user to surf the web. In a dynamic DHCP situation, the IP address will be given any time the user needs to surf the network and stores the IP address back into the server whenever the user finishes. If another client needs an IP address, then DHCP can reuse the IP address given to others previously.

1. **Global Clock:**

Besides not having a shared memory, keeping track of time in transactions is a daunting task in Distributed System. Due to the time zone differences across multiple regions of the world (with added complexity of certain regions change the clock to be ahead or backward by an hour every year), comparing time is a challenge. Having a timestamp for transactions is important because it helps us compare relationships between multiple events and analyze the cause and effect behaviors.

1. **Hadoop:**

Hadoop is a technology closely associated with Big Data. It has the capability to process large quantity of data, especially at an enterprise level. Processing these data can help a company analyzes trends in the dataset and adapt quickly to a changing market landscape against their competitors. Some of the Hadoop subset technologies include Hadoop Distributed File System (HDFS) and MapReduce. HDFS is useful in Hadoop because it can replicate a dataset across multiple servers so if a server experiences downtime, the information would still be available on another server. MapReduce enable the large dataset to be analyzed in a parallel manner instead of sequentially. Then MapReduce can organize the patterns of the data through shuffling or other techniques to show the results.

1. **Synchronization:**

Synchronizing instructions on a local machine is simpler than synchronizing instructions on a global network or even in a distributed system. The steps of processes become more complex, especially if there is no global clock used for referencing the ordered tasks needed to perform a program. The lack of global clock also causes concern for coordinating tasks and can slow down processing time to complete the instructions. A solution to this is to address global clock issue. One solution has been to transform the timestamp of transactions into Epoch format. Other solutions include using Cesium clocks, Stronium clocks, and Julian Calendar to address synchronization in a network.

1. **LAN:**

LAN stands for Local Area Network. It acts like a distributed system for a certain area or region. The components of the network can include routers, switches and servers that enable users to virtually connect remotely. There are many different sizes of LAN connections. An example of a LAN is at home where users are connecting their televisions, iPads, coffee machines, security camera system, and phones to a network. Another example of LAN would be multiple devices connecting to the Starbuck’s WIFI in a city. In this situation, the LAN architecture is considered to be client/server relationship because there are IT administrators involved providing Internet access to customers.

1. **Thread:**

A thread in a code collection helps make a program run more asynchronously. When the main program executes a line with “new Thread”, it will call the new Thread class to execute instructions behind the scenes. The goal is to reduce program run time because essentially, the program is splitting the tasks where the main task would continue as normal and the thread would run another set of instructions. Thus, the collaboration of multiple threads (with multi-tasking behaviors) working together to achieve an application end result helps the wait time for users to get a response back.

An example of a thread is below:

From the main class:

Boat x = new Boat();

x.start();

From the thread class:

Class Boat implements Runnable {

Public void run() { System.out.println(“This is thread.”);

}

1. **HTML:**

HTML stands for Hyper Text Markup Language. It has language formatting syntax rules for how documents and words should be displayed in multiple web browsers. The language uses a tagging concept for headers, paragraphs, lists, etc. Some of the useful features of HTML include adding links to a web browser, images, page colors, and text font sizes and colors. Combinations of these concepts can help users create a website.

To get started on writing HTML, a developer may use Sublime to create a basic website. An example of HTML is below:

Text: “Hello World! My name is Pepper and I like to eat cookies.”

To HTML:

<h> Hello World!</h>

<p> My name is Pepper and I live to eat cookies.</p>

1. **Blockchain:**

Blockchain technology uses encryption of data (called blocks). Each block contains everyday data, such as monetary transactions, along with sensitive information and timestamp fields of the transaction object. These fields are then concatenated together and encrypted (can be through SHA256). Afterwards they are added into an array list to form a chain. Block chain technology is often associated with Bitcoin but its application can be utilized by the finance sector and/or companies conducting financial transactions.

1. **UUID:**

UUID stands for Universally Unique Identifier. It is a collection of characters known to be unique and with no overlapping identifiers with others. This can be useful when we are creating IDs for our users so that we can uniquely identify them in our application. The collection of characters can include numbers and letters. In java, creating a UUID is not difficult. The below code shows how to generate a UUID in Java.

UUID uuid = UUID.randomUUID();

String string = uuid.toString();

1. **Network Address Translation**

NAT is short for Network Address Translation. It is similar to DHCP yet also different. It is similar to DHCP due to its involvement with IP addresses, but the IP addresses are considered fake for users. Often times, these addresses are given to multiple users so there can be duplicates. This system is suited on a local network and users can get assigned an official IP address by using a router to connect to the Internet through their internet provider.

**Lecture Notes**

**Lecture 1:**

1. Distributed System:
   1. can work on multiple computers like mobile, Internet, and wirelessly
   2. There is no global clock for processes and there will be compromises
   3. The user views multiple computers as one system
2. Intranet: like VPN and access to the Internet for authorized users only
3. Process:
   1. a direction for the Central Processing Unit to execute
   2. it does take up a lot of space on the computer
   3. has UUID, personal memory, security, and is a priority in the CPU
4. Thread:
   1. Not memory intensive as processes and is a part of a process
   2. Within the memory space of the process
   3. Like multiple children of a process
5. Open System:
   1. Open to the public and no one has to pay
   2. Has a defined interface
   3. Interface Definition Language (IDL):
      1. Defines the syntax rules of an interface: methods, parameters, responses, etc.
6. Considerations for Distributed System:
   1. There is no global clock
   2. One failed machine should not bring the entire computer system down
   3. Increase scaling can cause administration issue because the system design gets more complex
      1. One admin managing authentication or multiple people?
   4. Information is decentralized
7. Synchronous calls:
   1. Direct relationship between calls where a client will call a server and wait for a response back.
   2. Drawbacks: prone to network issues like distance and not time efficient.
8. Asynchronous calls:
   1. Non-blocking calls but the design is more complex; it is more time efficient because we would be “multi-tasking”
9. Transactions:
   1. Critical section – locking access to a shared resource while executing a transaction
   2. Semaphore – indicates if the critical section is available; T = wait || F = available
   3. Transactions are more challenging with Distributed transactions because there is no shared memory, harder to coordinate critical section and semaphores, messages are through the network where they could easily get lost, and you have to call multiple databases to formulate a response to the user
10. Inter-process Communication:
    1. Processes sharing memory space to conduct message exchanges.
    2. Locally – easy because of shared memory and easy do context switching
    3. Distributed system – difficult because messages are sent via the network and no shared memory
11. Universally Unique Identifiers (UUIDs):
    1. Unique collection of strings that no other machines would use
    2. Includes URLs and/or large randomized number
12. Middleware:
    1. Helps to resolve shared memory issue with inter-process communication when messages are sent over a network
    2. Locates in the middle of applications and OS libraries/instructions
    3. Shows the user an interface
13. Message Protocols:
    1. UDP – time efficient but messages might get lost
    2. TCP – time consuming but messages will be consistently received/sent
14. Bad Assumptions of Distributed System:
    1. Network is stable, reliable, safe, cheap, simple, and fast
15. Issues of wireless and wearable devices:
    1. Who should own/take care of healthcare data?
    2. Privacy leaks and security concerns

**Lecture 2: Architectures**

1. Architecture styles:
   1. Layered – like middleware sitting in between application and network levels
   2. Object – like calls within a repository where objects call on each other
   3. Event Bus – like MuleSoft with publish/subscribe relationship
   4. Shared data space – like event bus but 24/7 and the data has to be stored consistently; decoupling time
2. Layered Architecture:
   1. UI level, processing level, data level
   2. Example is simple client/server call
   3. Thin client – sending fields of an object separately to the server; causes security concern when so many fields are being exposed; heavy on the database; easier to do offline updates
   4. Thick client – sending fields of an object at the same time to the server; is client-heavy; horrible to do offline updates
3. Structured Peer to Peer Architecture:
   1. DTH – Distributed Hash Table
   2. Generate a random number and insert node in a range
   3. Goal is to search for data by key
   4. Seeders – contribute to the group
   5. Leechers – mostly interested in downloading software without contributions

**Lecture 3: Processes/Virtualizations**

1. API – interface with library calls that will do system call to the OS; has machine instructions on the back end but the user does not have to worry about backend machine code details.
   1. Threads are involved with process calls to do an instruction
   2. Context switching – switching back and forth to run a process vs another
2. Layers of VM Architectures: Application, OS, VM monitor, and hardware
   1. Superserver – when receiving an incoming request from client, manages tasks to other servers
3. Reasons for migrating code: updates, data migration, etc.
   1. Client can fetch the new code and then communicate with the server
   2. Can migrate by identifier(strongest), value (standard), and type (weakest)
   3. Ways to migrate:
      1. Stop machines, migrate, and restart machine
      2. Machine pulls code on an ass needed basis
      3. Send the code to machine and send the modified ones during migration
4. IPC: Inter process communication
5. UDP – keep sending info to another location
   1. Not too difficult to implement but there is no acknowledgement that the information was sent properly
   2. Messages are only sent over once and it could be in an unsorted order
6. TCP – maintain continuous two-way communication between processes to communicate
7. Synchronous calls: communication where the client waits for a response from the server. Client cannot do anything until it receives a response back; good for local machine implementation but bad for distributed system
8. Asynchronous calls: client sends requests to server and then proceeds to do something else while waiting for a response. It is more complex to code and hard to synchronize tasks and/or handle interruptions. Uses next available port to run threads
9. Socket: takes in a request and connects to a port
10. ServerSocket: listens for incoming Socket requests and creates a socket whenever there is another request…usually has a maximum queue length
11. Marshalling data representation: transforming data representation from assembly language to bit strings and vice versa
    1. Very hard to automate, especially with dynamic data structures
12. Serialization: send string of data bits one right after another. They can be read in any order.
13. HTTP methods: GET, POST, PUT, DELETE

**Chapter 4: Communication**

1. Communications over a distributed network can comprise of layered protocols, middleware protocols, and Transmission Control Protocol.
2. Synchronous RPC – like synchronous calls, the client will wait for the response from the server
3. Asynchronous RPC – like asynchronous calls, the client will shift focus and be interrupted later where the response is returned from the server
4. Isochronous Transmission Mode – data has to be sent on time only
5. People are sensitive in their ears so have to be careful with audio synchronization
6. Peer to Peer Architecture:
   1. Nodes are placed in a circle next to each other like numbers on a clock
   2. The mapping can show the connections of nodes and its flexibility
   3. Associated with Distributed Hash Table, DHT

**Lecture 7/Chapter 6: Synchronization**

1. Synchronization: Implementing the discrete steps of a process operation
   1. Time is in epoch format
      1. This format helps because telling time is difficult since there is no globally shared memory for time
   2. Speed causes concern for coordination problem, especially when there is no global clock…so there will be precision issues
      1. Each CPU will have skewed clock compared to another CPU
   3. Cesium clocks: standard clock but not fast enough now
      1. Current clock is Stronium clocks
   4. Julian calendar: used by the West
2. Distributed time: coordinating time while in a distributed system
   1. UTC is used now
   2. Clocks under development: Cesium 133, Optical clocks, strontium clock
3. Ideally, we want all clocks to be synchronized (C = t)
4. Network Time Protocol: Time for request and time for response
5. Berkeley Algorithm: Since the machines don’t have a reliable clock, we manually set the time to the average based upon an agreed time
   1. Issue: certain machines cannot readjust their time to the average because they might end up going backward
   2. Resolution: if clock is ahead, slow down the time gradually
6. Lamport Logical Clock:
   1. transitive relationship: a -> b, b -> c then a -> c
   2. concurrent: x and y processes are independent then they are concurrent and do not affect each other
   3. C(x) = clock of x process
   4. Will update(adjust) the transaction time when there is an incoming response in the return message to the local time
7. Can make an event unique by adding a digit after decimal
8. When doing updates, make sure that all the servers are applying the same ORDER of updates to get the same resulting updates
   1. For example, PEMDAS
9. Multicast – good for server replication
10. Vector clocks – like Lamport Time stamps but includes an extra integer…to update servers
    1. Vector[0] = [6,7,9]o would be sent to other servers to update
    2. Might be good to pair vector clock middleware with casually ordered communication since it is cheapest and simplest
11. Casual ordering: sending a message to a distributed system happens before a returning message
12. Token Ring Algorithm: processes in a circular ring network design
    1. Token is sent around the ring network
13. Election algorithm: when a process fails, another process picks it up
14. Bully algorithm: if a process fails, then the higher process number will take over

**Lecture 8:**

1. DHCP: dynamic Host Configuration Protocol
   1. When register devices on the network; recycling IP addresses
   2. The DHCP server sends an IP address whenever the client calls
   3. Dynamic allocation method: give any available IP address whenever the client calls, most common method; efficient because if a device is not being used, then we can give the IP address back to the server
   4. Automatic allocation method: gives the same IP address when client is called
   5. Static allocation method: MAC address assigns IP address in the table
2. NAT: Network Address Translation
   1. Not the same as DHCP
   2. Gives fake IP address so computers can surf on the Internet
   3. An example would be a user has an IP address and with a router, can access the Internet
   4. Works well on local network
   5. The router will replace the temporary IP NAT address with an official IP address from the internet provider and connects to the internet
3. ISP: Internet Service Provider
   1. Dynamic IP address whenever we move our laptop to work in a different location…causes issues because the Internet needs to send the information to you
4. IP Tunneling: passing data is packaged inside a bigger packet
   1. How VPN is set up
5. Mobile IP: IP address for mobile devices
   1. IP address is from static address and/or current location
   2. Home agent (reroute packet) and foreign agent
6. IPv4 vs IPv6: headers are different
   1. v6 allows more network addresses and geolocation in addresses with security in the header
   2. IPv6 data is wrapped inside IPv4 for migration and then unwrapped
   3. Strong mobility – moving data when processes are running
   4. Weak mobility – moving data at the beginning
7. Mobile agent: code and state moving to different locations to do work
   1. Cheaper to send code than ship data…especially in the case of Big Data

**Lecture 9**

1. IPsec – IP security
   1. Security below application layer like SSL and SSH and uses internet key exchange
   2. IPv6 compatible and encrypts traffic, authentication, anti-replay, set up session keys
2. VPN:
   1. Inspired by IP security, exchanges a key with the firewall (can be multiple firewalls) to gain access to work network
3. Spread spectrum technology
   1. CDMA – allows multiple applications to share the same bandwidth, such as Bluetooth technology
4. Asynchronous Transfer Mode, ATM: packets of data being received with different delivery time
   1. Competitor to IP
   2. Solution: have bigger buffer but will have delays
   3. Fast switching of packets for voice data
5. Bandwidth: bits sent in the network on average
6. Hadoop: grouping of libraries and utilities handling big data computation
   1. HDFS: storage
   2. Hadoop MapReduce software: handing big data
      1. Extract field from big data, filter and sort and map fields to a storage
      2. Shuffle and reduce steps