Ontario Tech UNIVERSITY

Faculty of Engineering & Applied Science

SOFE4790U – Distributed Systems

Group Work

Homework – Networking and Virtualization

Due Date: 10/02/2022

First Name	Last Name	Student ID
Abdul	Bhutta	100785884
Alexander	Campbell	100703650
Mamun	Hossain	100553073
Owen	Musselman	100657709

Part 1: Networking:

Review your course on computer networks. Can you try to summarize what you learned in that course? Focus on the 7 layers of the OSI interconnectivity model? What different detailed functionality does each layer provide? how does this help two applications to communicate?

In the Computer Networks course, some of the content learned was the OSI model, understanding TCP/IP and UDP connection and connectionless oriented protocols. We also became familiar with networking hardware, like routers, switches, hubs, bridges, and gateways. Additionally, subnetting, load balancing, DHCP and caching servers were discussed as well. All of these resulted in a firm understanding of how computer network's function.

The OSI model helps two different applications communicate by working their way through the seven layers. Where all these layers have a data protocol unit that communicates among two different nodes at each layer. For example, machine A is attempting to communicate with machine B, a message that is sent is handled by the first three layers which will then be segmented off once it reaches the transport layer. Next, the network layer creates a packet which goes to the data link layer where a frame is then created. Next, the physical layer transmits the data from machine A to machine B where the process is essentially reversed.

Layer Name	Layer Description
Layer 7 - Application layer	Interaction and access of the network and software are done using protocols allowing communication between each other. Examples: web browsers, email, messaging systems.
Layer 6 - Presentation layer	Where encryption and data compression are done. Decryption is also done on this layer as to be presented to the application layer, so it is then readable to the user.
Layer 5 - Session layer	Maintains sessions and is responsible for communication between devices. The session should be open when data is being transferred and then closed when transfers are done. Checkpoints are crucial here as if a session is halted, it can resume at the most recent checkpoint.
Layer 4 - Transport layer	Responsible for breaking the session layer data into segments. This layer will provide reliable data transfer as well. This is done with end-to-end error checking, using TCP/UDP protocols. TCP is connection-oriented which requires acknowledgement packets to proceed. UDP is a connectionless protocol which does not need acknowledgement packets to proceed.

Layer 3 - Network layer	Forwards packets from router to router, so the device can communicate with each other over a network. This layer will also find the best routing path to take for communication.
Layer 2 - Data link layer	Provides communication between two nodes while sending frames over a network. The frames contain the data. There are two subsections of this layer: Media Access Control, and Logical Link Control. MAC controls the frame and provides collision detection, and the LLC provides flow control, and automatic repeat requests.
Layer 1 - Physical Layer	Responsible for transmitting bit streams over a hardware medium (ethernet CAT6), determines the throughput latency, and error rates.

Part 2: Virtualization

SnowFlock

Cloud computing allows an end user to rent servers in the cloud and does not need a physical device such as a server, leading to lower costs. The main component of a cloud is providing a virtual machine (VM), which imitates a physical machine. Still, the end users are not observant of the fact that the devices are being shared by many other users or the location of the resources. The cloud provides user access and location transparency.

Cloud computing aims to help users set up their servers within seconds compared to purchasing, installing, and configuring the servers, which may take weeks or months. However, the cloud systems are not stand-alone and require a pool of resources to be added. To configure the servers, a user still needs to manage the cluster and add new servers. Snowflock solution to this issue is virtual machine cloning which uses API calls. It can solve the problems with a single logical operation with cloning, resource allocation, and cluster management. The VM makes copies of the cloud servers, which store the parent's operating system and application cache while adding them to a private cluster. These resources can be used at any time, depending on the number of resources required from the user. Snowflock has allowed the creation of VM in under 5 seconds and can process the load across many physical hosts. It has also integrated into the cloud-based-elastic servers, allowing the new worker node to come online and be loaded up 20 times faster. SnowFlock implements a similar process used in UNIX called a fork, which creates a new process of the parent (duplicate) within the system. In Snowflock, it copies the memory, processes, filesystem, and virtual devices, which can be used to create multiple copies and executed in a parallel sequence.

The way SnowFlock makes its approach is by using the following concepts: Virtualization (which allows for the cloud and machine cloning); Lazy Propagation (VM doesn't clone unless necessary); Multicast (Clones will all have similar attributes); Page Faults (Clone execution with missing memory won't run until page arrives); and Copy on Write (when memory is copied before being overwritten). With all this, it makes it possible to alleviate stress on the cloud servers and help with ensuring that they do not need to constantly boot new instances from the same template and aids with parallel computing, data mining, and serving web pages. Cloned virtual machines are missing most of their memory state by the time they are created by the descriptor:

Memserver process stops the VM so that its memory can settle. Once copied, clones can reconnect to the external world in their new enclosed VM. The memserver provides the bits of information the clone needs from its parent. Copy on write is used to circumvent the issue of corrupted memory, where the writing access from pages in memory is removed. A page fault will occur if a write is tried, and that notifies the hypervisor to copy the page. Importantly the parent VM can write to that page to keep running, but the memserver needs to use the copied read-only page.

Mcdist is used to distribute a packet to various receivers. It does this through IP multicasting which allows for network hardware parallelism, which in turn lowers the memory server load. Mcdist does not need to be a reliable method as the packets do not need to be received in a specific order, where the server multicasts for responses, and the clients timeout if/when they do not get a reply for a request they made and have to try again. There are 3 optimizations of mcdist:

- **Lockstep:** when temporal locality occurs, various clones will request the same page, and it will be done almost at the same time, and mcdist will ignore all but requests but the first one that reaches.
- **Flow control:** The server will limit its sending rate to be a weighted average of the client's receive rate. If this is not done, then the receiver will be overwhelmed with too many pages sent from the server.
- **End game:** At the time the server has transmitted most of its pages, the server will go back to unicast responses. Usually, anything at this point is requested retries, and having retries sent to every clone is a waste.

Compute Resource Consolidation Pattern

In a distributed system it often makes sense to logically partition resources into separate computation units, however, when it comes to scaling this up, we can sometimes find that this will lead to wasted computing power. If the separation is too much, then some units within a distributed system will be idle or underutilized. To solve this problem, we can use the Compute Resource Consolidation Pattern. In this pattern, we take similar resources and combine them into one consolidated computation unit to ensure that we get as much use as possible. By consolidating resources we can increase utilization and decrease overhead and costs. A common approach is to look at resources that are similar in both processing requirements and lifetime, meaning that they will scale similarly. When implementing the compute resource consolidation pattern, it is recommended to think about the topics below before implementing the design,

Topics	Description
Scalability and Elasticity	Do not group tasks that require the same level of scalability for a resource.
Lifetime	Design the tasks to have checkpoints and restart the tasks when interrupted or resource is resumed.
Release cadence	If one code requires to be updated or the system to be restarted, all other tasks will have to stop and be restarted as well!
Security	Each task running should have trust between each other without causing an issue or corrupting each other.
Fault Tolerance	If one system fails, all the tasks within the system are at a halt as well.
Contention	Do not use tasks that require the same resources such as two tasks using a lot of memory.
Complexity	Adding multiple tasks within the same system increases the complexity of the code.
Stable logical architecture	Design the code in a manner where it does not need to change even if the physical environment has changed.

Data Partitioning Guidance

Horizontal partitioning, also known as sharding, is storing tables in different databases while maintaining the same schema; each data partition is referred to as a shard in this case. This helps to reduce the number of rows in a specific table due to the tables being spread out. The difficulty of partitioning horizontally is in the sharding strategy itself, meaning that we want our databases to be evenly accessed and distributed. For example, all commonly accessed pieces of data being stored in one shard would mean this shard is accessed much more than the others; what we would want to do instead is distribute these commonly accessed pieces of data through the system.

Vertical partitioning is when you break up the columns of a table into several databases, meaning that the same key exists in several databases, but the full row is split up.

Functional partitioning is when you separate data based on its function in the distributed system. For example, partitioning data by read-only or write-only would be an example of functional partitioning.

In big data or a distributed system, it is impossible to keep all the data files on one resource, which is why partitioning is significant. Partitioning allows the data to be stored in separate storage and can have many benefits which are shown below,

Improvement	Description
Scalability	Increasing the data storage on a single server is limited where having partitions of many drives spread throughout on different nodes will achieve almost unlimited data storage.
Performance	Allows the user to access multiple files or operation in parallel.
Security	User can partition the data with sensitive material and apply various security protocols.
Provide management	Allow the user to manage, monitor, failure protection, and other tasks based on the partition and what each partition contains.
Availability	Allows the user to have access to files through redundancy or even if one partition fails, they will still have access to other partitions.
Match the data store to the pattern of use	Partitioning lets each individual partition to have a different type of data store. In other words, files could be stored using a document database, and binary data would be put into blob storage.

======================== END OF GROUP WORK ========================

Homework 2: Network and Virtualization
Member: Abdul Bhutta
Changes: RED

Part 1: Networking:

Review your course on computer networks. Can you try to summarize what you learned in that course? Focus on the 7 layers of the OSI interconnectivity model? What different detailed functionality does each layer provide? how does this help two applications to communicate?

The OSI layer helps two different applications communicate through the 7 layers. Each layer contains a protocol data unit used to communicate among two different nodes at each layer. When computer A is trying to communicate with computer B, the data or message being sent is managed by the first three-layer (Application, Presentation, Session) and then segmented at the transport layer. The network layer will produce a packet which will then proceed to the data link layer, which creates a frame for the packets that were received. The physical layer will send the information from computer A to computer B by converting it to bits and sending it over the medium. At computer B, it will unframe the packets at each layer following the same protocol used by computer A until the message is received. The seven OSI layers are shown below, with examples of devices being used at certain levels and protocols.

OSI Layer Level	Description	Example
Physical Layer (Layer 1)	Transmits bits over a medium and determines throughput, latency, error rate of a network link.	Ethernet Cables, Coax Cable, fiber cable, HUB
Datalink Layer (Layer 2)	It provides communication between two nodes while sending **frames** over a network link; within the frame is the data that is transmitted from one end to another end (Wire) which is encapsulated. The data link layer is divided into two subsections: MAC (Media access control) and LLC (Logical Link Layer). MAC allows for control of the frame and provides any detection of a collision of the frames while the LLC provides error checking such as checksum, parity bit, and CRC (cyclic redundancy codes).	MAC Address, Switch

Network Layer (Layer 3)	It provides the ability to move **packets** between the networks using routing protocols. The network layer allows other devices or networks to communicate with each other over multiple links (end-to-end) through packets and allows for network addressing (IP Address)	IP, Routing Protocols
Transport Layer (Layer 4)	The transport layer (4th layer) provides with reliable data transfer while segmenting them into packets. It provides error checking and flow control in an **end-to-end** network whereas in the link layer it provides error checking at the physical layer. The main protocol being used at the transportation layer is TCP/UDP. TCP is a connection-oriented protocol and reliable which requires an acknowledgement packet (HTTP) while UDP is a connection-less oriented protocol and unreliable which does not require any form of acknowledgement packet (VOIP).	TCP/UDP
Session Layer (Layer 5)	The session layer maintains the ongoing session (**ports**) while keeping it active and provides synchronization which allows the user to reconnect to the same checkpoint in case of a failure.	OS, Remote Procedure call (RPC), Gateway
Presentation Layer (Layer 6)	The encryption and data compression are done at the presentation layer such as ASCII coding.	ASCII, signed integer, unsigned integer
Application Layer (Layer 7)	Provides the user access to network resources/applications and allows protocols to communicate with each other.	HTTP, FTP. SMTP, DNS

Part 2: Virtualization

SnowFlock

Cloud computing allows an end user to rent servers in the cloud and does not need a physical device such as a server, leading to lower costs. The main component of a cloud is providing a virtual machine (VM), which imitates a physical machine. Still, the end users are not observant of the fact that the devices are being shared by many other users or the location of the resources. The cloud provides user access and location transparency.

Cloud computing aims to help users set up their servers within seconds compared to purchasing, installing, and configuring the servers, which may take weeks or months. However, the cloud systems are not stand-alone and require a pool of resources to be added. To configure the servers, a user still needs to manage the cluster and add new servers. Snowflock solution to this issue is virtual machine cloning which uses API calls. It can solve the problems with a single logical operation with cloning, resource allocation, and cluster management. The VM makes copies of the cloud servers, which store the parent's operating system and application cache while adding them to a private cluster. These resources can be used at any time, depending on the number of resources required from the user. Snowflock has allowed the creation of VM in under 5 seconds and can process the load across many physical hosts. It has also integrated into the cloud-based-elastic servers, allowing the new worker node to come online and be loaded up 20 times faster. SnowFlock implements a similar process used in UNIX called a fork, which creates a new process of the parent (duplicate) within the system. In Snowflock, it copies the memory, processes, filesystem, and virtual devices, which can be used to create multiple copies and executed in a parallel sequence.

The way SnowFlock makes its approach is by using the following concepts: Virtualization (which allows for the cloud and machine cloning); Lazy Propagation (VM doesn’t clone unless necessary); Multicast (Clones will all have similar attributes); Page Faults (Clone execution with missing memory won’t run until page arrives); and Copy on Write (when memory is copied before being overwritten). With all this, it makes it possible to alleviate stress on the cloud servers and help with ensuring that they do not need to constantly boot new instances from the same template and aids with parallel computing, data mining, and serving web pages. Cloned virtual machines are missing most of their memory state by the time they are created by the descriptor:

Memserver process stops the VM so that its memory can settle. Once copied, clones can reconnect to the external world in their new enclosed VM. The memserver provides the bits of information the clone needs from its parent. Copy on write is used to circumvent the issue of corrupted memory, where the writing access from pages in memory is removed. A page fault will occur if a write is tried, and that notifies the hypervisor to copy the page. Importantly the parent VM can write to that page to keep running, but the memserver needs to use the copied read-only page.

Mcdist is used to distribute a packet to various receivers. It does this through IP multicasting which allows for network hardware parallelism, which in turn lowers the memory server load. Mcdist does not need to be a reliable method as the packets do not need to be received in a specific order, where the server multicasts for responses, and the clients timeout if/when they do not get a reply for a request they made and have to try again. There are 3 optimizations of mcdist:

- **Lockstep:** when temporal locality occurs, various clones will request the same page, and it will be done almost at the same time, and mcdist will ignore all but requests but the first one that reaches.
- **Flow control:** The server will limit its sending rate to be a weighted average of the client's receive rate. If this is not done, then the receiver will be overwhelmed with too many pages sent from the server.
- **End game:** At the time the server has transmitted most of its pages, the server will go back to unicast responses. Usually, anything at this point is requested retries, and having retries sent to every clone is a waste.

Compute Resource Consolidation Pattern

Grouping multiple tasks and operations into one unit to maximize resource utilization will allow the user to cut the cost and expenses of their business. Running various tasks on a different system with limited usage and charges per resource is not optimal. Grouping similar jobs with a similar scalability profile and processing requirement will help them scale together as one unit. The idlest situation is running systems that are not cost-effective and are frequently idle. When combining multiple tasks, tasks that require scalability or the computational level that fluctuates depending on the traffic should not be grouped with functions that do not require as many resources, which may lead to more tasks being done. When implementing the compute resource consolidation pattern, it is recommended to think about the topics below before implementing the design,

Topics	Description
Scalability and Elasticity	Do not group tasks that require the same level of scalability for a resource.
Lifetime	Design the tasks to have checkpoints and restart the tasks when interrupted or resource is resumed.
Release cadence	If one code requires to be updated or the system to be restarted, all other tasks will have to stop and be restarted as well!

Security	Each task running should have trust between each other without causing an issue or corrupting each other.
Fault Tolerance	If one system fails, all the tasks within the system are at a halt as well.
Contention	Do not use tasks that require the same resources such as two tasks using a lot of memory.
Complexity	Adding multiple tasks within the same system increases the complexity of the code.
Stable logical architecture	Design the code in a manner where it does not need to change even if the physical environment has changed.

Data Partitioning Guidance

In big data and distributed system, it is impossible to keep all the data files on one resource, which is why partitioning is significant. Partitioning allows the data to be stored in separate storage and can have many benefits shown below,

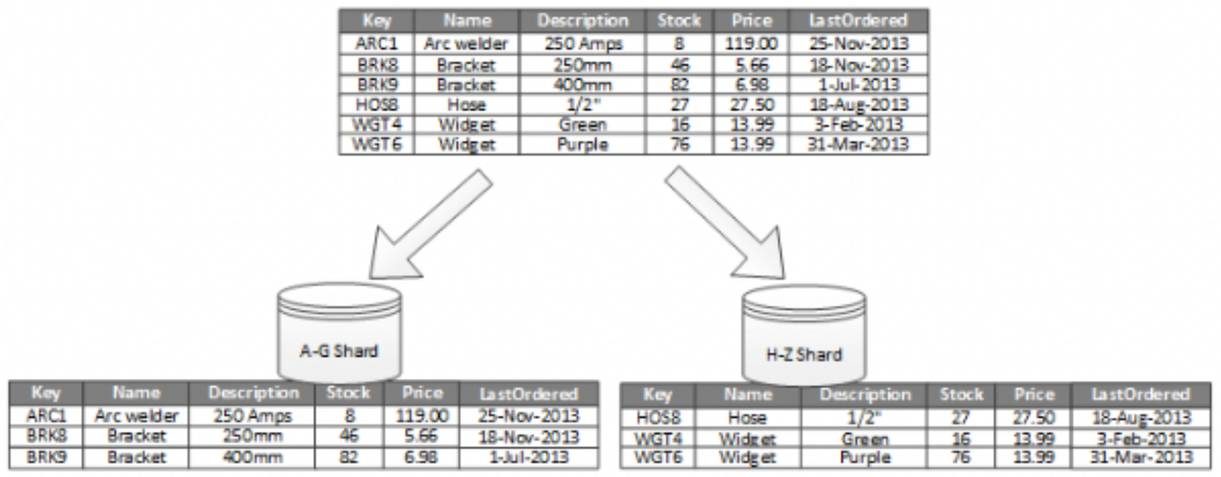
Improvement	Description
Scalability	Increasing the data storage on a single server is limited where having partitions of many drives spread throughout on different nodes will achieve almost unlimited data storage.
Performance	Allows the user to access multiple files or operation in parallel.
Security	User can partition the data with sensitive material and apply various security protocols.
Provide management	Allow the user to manage, monitor, failure protection, and other tasks based on the partition and what each partition contains.

Availability	Allows the user to have access to files through redundancy or even if one partition fails, they will still have access to other partitions.
Match the data store to the pattern of use	Partitioning lets each individual partition to have a different type of data store. In other words, files could be stored using a document database, and binary data would be put into blob storage.

Designing Partitions

1. Horizontal partitioning (Sharding)

Each partition stores a portion of the data (rows of data), such as records 1-50, and the subsequent partition holds records 51 – 100, where each partition is known as a shard.



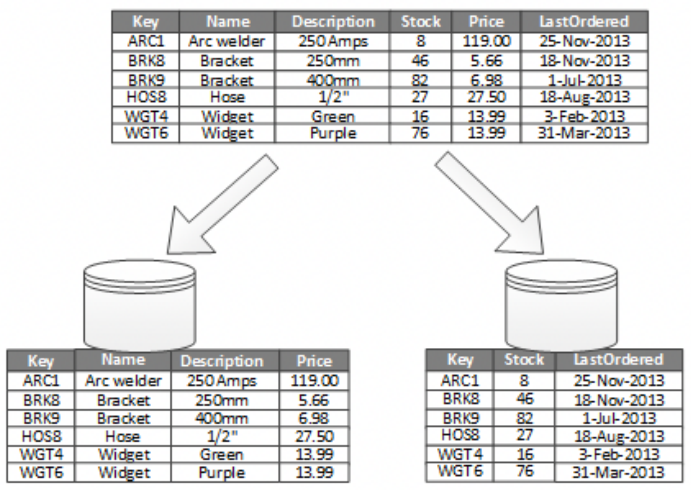
Key	Name	Description	Stock	Price	LastOrdered
ARC1	Arc welder	250 Amps	8	119.00	25-Nov-2013
BRK8	Bracket	250mm	46	5.66	18-Nov-2013
BRK9	Bracket	400mm	82	6.98	1-Jul-2013
HOS8	Hose	1/2"	27	27.50	18-Aug-2013
WGT4	Widget	Green	16	13.99	3-Feb-2013
WGT6	Widget	Purple	76	13.99	31-Mar-2013

Key	Name	Description	Stock	Price	LastOrdered
ARC1	Arc welder	250 Amps	8	119.00	25-Nov-2013
BRK8	Bracket	250mm	46	5.66	18-Nov-2013
BRK9	Bracket	400mm	82	6.98	1-Jul-2013

Key	Name	Description	Stock	Price	LastOrdered
HOS8	Hose	1/2"	27	27.50	18-Aug-2013
WGT4	Widget	Green	16	13.99	3-Feb-2013
WGT6	Widget	Purple	76	13.99	31-Mar-2013

2. Vertical partitioning

Each partition holds a table or a section of the table (columns), such as partition one will hold columns 1 – 10 while partition two will hold columns 11 – 20. Another example is partition one can save the most used columns while partition two will have the lesser used columns in the database.



Key	Name	Description	Stock	Price	LastOrdered
ARC1	Arc welder	250 Amps	8	119.00	25-Nov-2013
BRK8	Bracket	250mm	46	5.66	18-Nov-2013
BRK9	Bracket	400mm	82	6.98	1-Jul-2013
HOS8	Hose	1/2"	27	27.50	18-Aug-2013
WGT4	Widget	Green	16	13.99	3-Feb-2013
WGT6	Widget	Purple	76	13.99	31-Mar-2013

Key	Name	Description	Price
ARC1	Arc welder	250 Amps	119.00
BRK8	Bracket	250mm	5.66
BRK9	Bracket	400mm	6.98
HOS8	Hose	1/2"	27.50
WGT4	Widget	Green	13.99
WGT6	Widget	Purple	13.99

Key	Stock	LastOrdered
ARC1	8	25-Nov-2013
BRK8	46	18-Nov-2013
BRK9	82	1-Jul-2013
HOS8	27	18-Aug-2013
WGT4	16	3-Feb-2013
WGT6	76	31-Mar-2013

3. Functional partitioning

Partition is based on the purpose of the data being used, what category it comes under, or one primary goal. For example, having read-only data in one partition for faster access and another partition for read-write data. Designing the partition is challenging and should focus on the primary goal or main function of the partition. It is recommended that all three design partitions should be integrated into the schema.

Homework 2: Network and Virtualization
Member: Owen Musselman

Note: Changes made after group discussion are highlighted in red.

Part 1: Networking:
Review your course on computer networks, can you try to summarize what you learned in that course? Focus on the 7 layers of the OSI interconnectivity model? What different detailed functionality does each layer provide? How does this help two applications to communicate?

In the computer networks course, the content learned was the OSI model, the hardware used in creating the networks like routers, switches, hubs, modems, cables (ethernet, coax, fiber), calculating the best path to traverse the network, error checking by using checksums. DHCP and caching servers were also covered which would assist in load balancing and latency.

The Open System Interconnection (OSI) is a seven layer model that is used to communicate over a network. The OSI model's seven layers are as follows:

1. *Physical Layer*: Transmits the raw bit stream over the physical medium, like cables frequencies, pins. Layer also determines throughput latency, and error rates.
2. *Data Link Layer*: Node to node data transfer which is between two connected nodes. Additionally, the data link layer deals with error correction. Interestingly, this layer also has two sub layers which are the Media Access Control layer and the Logical Link Control layer (MAC and LLC respectively). MAC is responsible for interaction with the hardware interactions. LLC is responsible for flow control, and automatic repeat requests.
3. *Network Layer*: Where the layer is responsible for forwarding packets from router to router. This layer assists in increasing the efficiency of packet sending/receiving by finding good routing paths.
4. *Transport Layer*: Layer is responsible for breaking the data from the Session layer (Layer 5) into different segments. This layer also reassembles the segments on the receiving end and puts them together back into the data it was so that it is understood by the Session layer. This layer is also responsible for sending data at a correct rate so that it matches the speed of the device that is receiving the data (flow control). This layer will also error check and request for data to be resent in the case of an erroneous transmission.
5. *Session Layer*: Communication between devices and sessions. It will ensure that a session is open while data is being transferred (send or receive), this layer will also close the session after the job is finished. Checkpoints are an important part of this layer as if the session is stopped the device can start the transfer from the latest checkpoint that was made.
6. *Presentation Layer*: Layer defines how devices should keep their messages secret (encrypted) along with compressing data to ensure its proper transmission. Any data from the application layer is taken here and it is prepared to have it transmitted to the lower layers.
7. *Application Layer*: This layer gets data from the user directly through software like web browsers. This layer provides communication through the lower layers to make connections with different applications.

The OSI model helps two applications communicate by having each layer add its information in the form of headers. Each layer is seen to handle messages from the layers above and below it which is accomplished by protocols. So each layer sends the processed data to the next layer, performs error checking like checksums to ensure the data is correct, ensuring proper communication between applications.

Part 2: Virtualization

Compute Resource Consolidation

Where websites, microservices and virtual machines are running in their own environment, causing inefficient resource usage even when idle, increasing management and runtime costs.

Tasks are able to be grouped together by means of scalability, processing capabilities and their lifetime. When this is done, they are able to scale together. A grouping method to be considered is tasks that need large amounts of CPU power in shorter bursts, and these could be grouped together into a single computational unit. It is noteworthy that there is balance between the need of keeping expensive resources being used and them being over stressed, as the more compute heavy tasks that are longer should not be grouped together as the resource is both being stressed, and used for extended periods of time.

- Scalability & Elasticity: Avoid grouping any tasks that are going to have scalability needs that will conflict within the same compute unit.
- Lifetime: Design tasks with checkpoint system which would let them halt operations in a clean manner, and then allow them to resume their operation before they are halted and the compute unit is restarted. Additionally, when multiple longer tasks that are in a compute unit it will more than likely be necessary to stop the unit from being recycled before these tasks are completed.
- Release cadence: When the config or the implementation of a task is altered frequently, it is most likely that the compute unit needs to be halted so that it can be reconfigured with these new changes and then restarted. If this is done, every other task within that compute unit needs to be restarted as well.
- Security: If there are multiple tasks in the same compute unit they could possibly share access to the same resources. In this case there needs to be a high trust between these tasks, as there is the possibility of having one task corrupt the operation of another task. Each task in the same compute unit is only as secure as the one with that is most vulnerable.
- Fault tolerance: In the case that one task within compute unit malfunctions or fails (even if there are multiple), it can have an effect on the other tasks within that same compute unit.
- Contention: Tasks should not have to compete for the same resources if they are in the same compute unit. For example two heavy computation tasks should not be grouped into the same compute unit as they would be playing tug-of-war with the resources.
- Complexity: When there are various tasks on the same compute unit, the complexity is increased, making the testability, debuggability, and the maintainability less than desired.
- Stable logical architecture: The code for each task should not need to be altered, even in the case of the physical environment it resides in has been altered.

Compute resource consolidation should be used when the tasks needed to run cannot do so cost effectively in their own separate compute units. For example tasks that spend vast amounts of time idle should not be in their own compute unit. This method of resource consolidation can potentially not be suitable if tasks are performing actions with sensitive data, or critical operations, in this case they are better off running in their own separate compute unit.

Data Partitioning
In distributed systems it is sensible to logically partition the resources into separate computational units, but this can lead the system to consume more power when not needed, like when idle. This problem is solved by using Compute Resource Consolidation and by doing so we can increase utilization, and decrease overheads and costs.

There are various reasons to perform data partitioning some are as follows:
- Improve scalability: Databases will eventually reach their limit in terms of hardware, this can be solved by having separate partitions located on different servers.
- Improve performance: performance is increased by having access operations occur over a smaller volume of data, like breaking up a large dataset into smaller sets instead of having to search through a single massive dataset.
- Improve security: More sensitive data can be separated in their own partitions which can then have different security measures put upon it.
- Provide optional flexibility: Can minimize the cost of operations, or maximizing the administrative efficiency. Additionally strategies used could be for management, backups/restoring, monitoring.
- Match the data store to the pattern of use: Partitioning lets each individual partition to have a different type of data store. In other words, files could be stored using a document database, and binary data would be put into blob storage.
- Improved availability: If the data is partitioned across multiple locations/serves, there is no single point of failure.

There are different types of partitioning methods three of which are:
- Horizontal partitioning (sharding): Each partition has its own data store, but those data stores have identical schemas. Each partition here has its own subset of the data. This reduces the number of rows in the tables since the tables are now spread out. The difficulty of partitioning horizontally is the sharding itself, as we want the databases to be accessed evenly as well as distributed evenly.
- Vertical partitioning: Kind of the opposite of Horizontal partitioning, where the partitions hold a subset of the field items in the data store. Where each of these fields are divided in terms of their pattern of use like how frequently they are used.
- Functional partitioning: Data is aggregated in terms of how it is used. Like in the case of an invoice, it would be stored in one partition and the stock of a product would be stored in another partition.

The above strategies can be combined, for example Functional partitioning and Vertical partitioning can be combined for a more specific use case.

Cloned VMs (SnowFlock)
Cloned virtual machines are missing most of their memory state by the time they are created by the descriptor:

Memserver process stops the VM so that its memory can settle. Once copied, clones can reconnect to the external world in their new enclosed VM. The memserver provides the bits of information the clone needs from its parent. Copy on write is used to circumvent the issue of corrupted memory, where the writing access from pages in memory are removed. A page fault will occur if a write is tried, and that notifies the hypervisor to copy the page. Importantly the parent VM can write to that page in order to keep running, but the memserver needs to use the copied read only page.

Mcdist is used to distribute a packet to various receivers. It does this through IP multicasting which allows for network hardware parallelism, which in turn lowers the memory server load. Mcdist does not need to be a reliable method as the packets do not need to be received in a specific order, where the server multicasts for responses, and the clients timeout if/when they do not get a reply for a request they made and have to try again. There are 3 optimizations of mcdist:

- Lockstep: when temporal locality occurs, various clones will request the same page, and it will be done almost at the same time, and mcdist will ignore all but requests but the first one that reaches.
- Flow control: The server will limit its sending rate to be a weighted average of the client's receive rate. If this is not done, then the receiver will be overwhelmed with too many pages sent from the server.
- End game: At the time the server has transmitted the majority of its pages, the server will go back to unicast responses. Usually anything at this point are requested retries, and having retries being sent to every clone is a waste.

With the use of lLazy propagation, virtualization, page faults, copy on write, multicast. With all of these present it is possible to reduce the stress on the cloud servers as they do no need to reboot new instances continuously. This helps with serving web pages, and parallel computing as well.

Generic Questions:

- What resources are managed and how?

  The resources managed in the system are CPU, memory, storage, VMs. They are managed through data partitioning, vm cloning, and resource consolidation.
- How does the system described in the article provide the following requirements:
  - Resiliency in case of failures: Data partitioning can allow for no single point of failure, allowing for other partitions to be in use if another fails. If a VM were to fail a new clone can be made in a respectable amount of time to fill in for the failure.
  - Scalability in the case of expansion in various parameters: More VMs can be cloned if needed (scalable and elastic). With data partitioning it allows the data to be hosted on separate servers. With consolidation, compute units should not have conflicting scalability requirements within them.
- How are transparencies provisioned within the described image sharing system?
  Transparency within data partitioning is that the users do not know how or where the data

is partitioned, all they know is that it is a seamless experience and it seems like it is a single implementation. For VM cloning with lazy state replication, the users do not see what VM they are in. These cloned VMs with lazy replication can increase the efficiency as only the state that is used is copied, but the user is not concerned, just if the service is usable. As for compute resource consolidation, when a user is actively using the system, they don’t see or care to know what compute unit they are utilizing, they only want their workload to be dealt with, hence what resources are apart of the compute unit, and what compute unit they are using is hidden from the user.

Homework 2: Network and Virtualization
Member: Alexander Campbell

Part 1: Networking

The OSI model is the Open Systems Interconnection model which describes the architecture that allows for various machines to communicate with each other using standardized protocols. The OSI model can be broken up into 7 abstracted layers, each with one being layered on top of the other, each with their own unique function for universal communication. Each layer can only communicate with the layers it is directly adjacent to, so each communication up or down. The layers from top to bottom are: Application layer, Presentation layer, Session layer, Transport layer, Network layer, Data Link layer, and finally the physical layer.

The application layer is the highest level layer and is used for interaction with the actual software application and the user controlling the machine. However due to this being a network architecture the application layer does not actually include the softwares the end user uses, but the components and protocols that connect the user to the network. These can include things such as HTTP, STMP, FTP, etc. The application layer receives this data from the presentation layer.

The presentation layer's main function is to package or present data taken or given from the application layer. It does so by encrypting, decrypting, compressing, or decompressing the data given to it depending on the layer that needs the data. For example, the presentation layer may receive data from the application layer that is too large to be sent over the network so it would be the presentation layer's job to package the data neatly in the required size before sending it to the layer below it. It may also decode messages sent from the layer below it so that it can be presented on the application layer in a form readable to the end user.

The session layer is the layer below the presentation layer, it creates the communication between two devices such that it will initialize, maintain, and terminate the connection between devices. It is known as the session layer as a "session" is the duration of communication between two machines. Many authentication protocols will also utilize the session layer.
This layer also allows for segmenting of data so in the case a crash occurs, we do not have to download the entire file again and only the missing segments. The transport layer is used to pass these messages from machine to machine.

The transport layer is used for communication of data between machines in the form of connectionless communication such as UDP or connected-oriented communication such as TCP. This data can be in variable-length formats and also be received at variable time lengths. This layer is also responsible for both error control and flow control. Error control means the transport layer ensures that the data it's receiving is not corrupted, for example by the use of checksums and asking for a repeat sending of data if anything is incorrect. Flow control ensures that a machine is not receiving more data than it can manage, for example when establishing a TCP connection the receiving machine will acknowledge the reception of data, this reception is done by the transport layer.

The network layer's function is to actually send the data that was packaged in the transport layer as packets to the network. Packets are a segmented form of data in smaller units that, upon reception, a computer can reassemble in order to have readable, in-order data. This layer also performs routing, which is to find the least cost path to the destination machine.

The data link layer is a node-to-node transfer. This means it has a similar use-case to the network layer however the data link layer provides communication between machines within the same network. Similar to the transport layer, the data link layer provides the functionality for communication and flow control between machines.

Lastly the physical layer is the actual real transmission of data bits such as 1s and 0s through some sort of data transfer device such as a switch. This layer will also establish a signal convention between the two devices to ensure that it receives the correct information, such as a high signal is a 1 and low signal is a 0.

Part 2:

Compute Resource Consolidation Pattern

In a distributed system it often makes sense to logically partition resources into separate computation units, however when it comes to scaling this up we can sometimes find that this will lead to wasted computing power. If the separation is too much, then some units within a distributed system will be idle or underutilized. To solve this problem we can use the Compute Resource Consolidation Pattern.

In this pattern, we take similar resources and combine them into one consolidated computation unit to ensure that we get as much use as possible. By consolidating resources we can increase utilization and decrease overhead and costs. A common approach is to look at resources that are similar in both processing requirements and lifetime, meaning that they will scale similarly.

Compute Partitioning Guidance

We can partition data in a distributed system to improve scalability, performance, security, flexibility, and availability. Partitioning data is essentially when split up data into sections that are accessed and managed separately from each other. This improves scalability by distributing the data across multiple computers so there is no memory limit. This also increases the availability and performance since smaller data sets being spread out makes access more efficient and less prone to complete failures. At the same time, certain partitions can be made harder to access than others, making it more secure. Within partitions there are three main strategies: Horizontal partitioning, Vertical partitioning, and Functional partitioning.

Horizontal partitioning, also known as sharding, is storing tables in different databases while maintaining the same schema, each data partition being referred to as a shard in this case. This helps to reduce the amount of rows in a specific table due to the tables being spread out. The difficulty of partitioning horizontally is in the sharding strategy itself, meaning that we want our databases to be evenly accessed and distributed. For example, all commonly accessed pieces of data being stored in one shard would mean this shard is accessed much more than the others, what we would want to do instead is distribute these commonly accessed pieces of data through the system.

Vertical partitioning is when you break up the columns of a table into several databases, meaning that the same key exists in several databases but the full row is split up.

Functional partitioning is when you separate data based on its function to the distributed system. For example, partitioning data by read-only or write-only would be an example of functional partitioning.

Homework 2: Network and Virtualization
Member: Mamun Hossain

Introduction:

To understand networking, we have to know what networking is exactly. We have previously learned that computer networking refers to the interconnection between devices that have the ability to transfer and exchange data and share their resources amongst each other. All devices have the ability to network with each other in some capacity.
You will have devices such as switches and routers that are the skeleton of the computer network. These devices will then connect to a simple device, like your computer or printer, which then will send data to the router so it can choose the best route for the network to pass through.

Switches and routers are very different but they have a distinct way of knowing how they interact with different devices. Some switches have a permanent pre assigned MAC address, whereas some routers will have a network assigned external IP address.
The reason why computer networks prove to be so important is that across organizations, it allows the ease of sharing information, it saves a lot of money, storage, and also adds a high degree of data security, and to this day it is ever evolving, as we have been learning in our distributed course.

Computer Networks Course Summary:

In our previous year, we had begun an introduction to Computer Networks where we had learned and applied our knowledge of C programming and Linux as well as understanding the TCP/IP reference models. We also learned briefly about the OSI network layers and how it interacts with the commands we had instructed in our labs. The goal of the course was to be able to use Cisco hardware and software tools to build small networks, understand the functions, protocols and timing of network devices such as hubs, bridges, switches, routers, and gateways, and to understand ethernet coding and protocol structure. The goal was to increase our understanding of computer networks and being able to even create our own computer network in the future.

7 Layers of the OSI interconnectivity model:

In the OSI interconnectivity model, there are 7 different layers. But it wasn’t always the case as several decades ago there were two different models that depicted Computer Networks. That being said, the OSI model goes from layer 7 to layer 1. We will be discussing the following layers: Application (Layer 7); Presentation (Layer 6); Session (Layer 5); Transport (Layer 4); Network (Layer 3); Data Link (Layer 2); Physical (Layer 1). We will begin discussing with layer 7.

Layer 7 - Application:
With the application layer, it will provide a means for the end user to interact with a device that is connected to the specified network. The way the end user interacts can be done in a way such as opening a web browser or network based application (email). To put it simply, the application layer is the data the user can see while the application is running. Some examples of this layer could be, but not limited to: email, instant messaging, browsing the internet, printing documents etc.

Layer 6 - Presentation:
The purpose of the presentation layer is to send the data from an application and convert that in a readable form for the network to access. This basically allows the system to understand and relay the information to the end user. Examples of the session are: Encryption and expansion of a message to allow efficient travel across the network, content translation, graphics formatting, etc.

Layer 5 - Session:
The session layer provides different values to the number of bytes in each session that the end user is using during that period. The main purpose of the session layer is to ensure applications can work on devices, and can manage and terminate bytes and code through a network. Some examples are: Synchronization of data flow, Partitioning of services into functional groups, and creating dialog units.

Layer 4 - Transport:
The transport layer will allow different devices to connect and interact to each other if they are on the same network. Dependant on the network and the application being used, the transport layer can communicate with the devices which is reliable, secure, and offer the best effort communications. Examples of this are: Application identification, Detecting errors in the transmissions, and being able to identify client level entities.

Layer 3 - Network:
The network layer makes sure that the packets of information can be sent through the layer 2 networks and sent to the right locations, also known as logical addresses. With the evolution of modern technology, IP addresses have been made in a way to easily access and connect different networks with each other, as we have seen with different server machines across the world. Using different things called subnets (separation of the network layer into smaller parts -Figure 1) will help the router use the subnet of the network's IP address to connect to all the devices that are sharing that server information. Now the way the router can accomplish this is by finding the shortest path in the network and calculating the best way to reach each specified network in the organization. An example of this would be from going from one router to another on the network to create the fastest connection. Now a router traveling from one router to another can cause fragmentation, but all the fragmented packets are repaired and put together at the final destination system of the layer network. Some basic security functionality can also be set by the filtering traffic using router addresses or something similar.

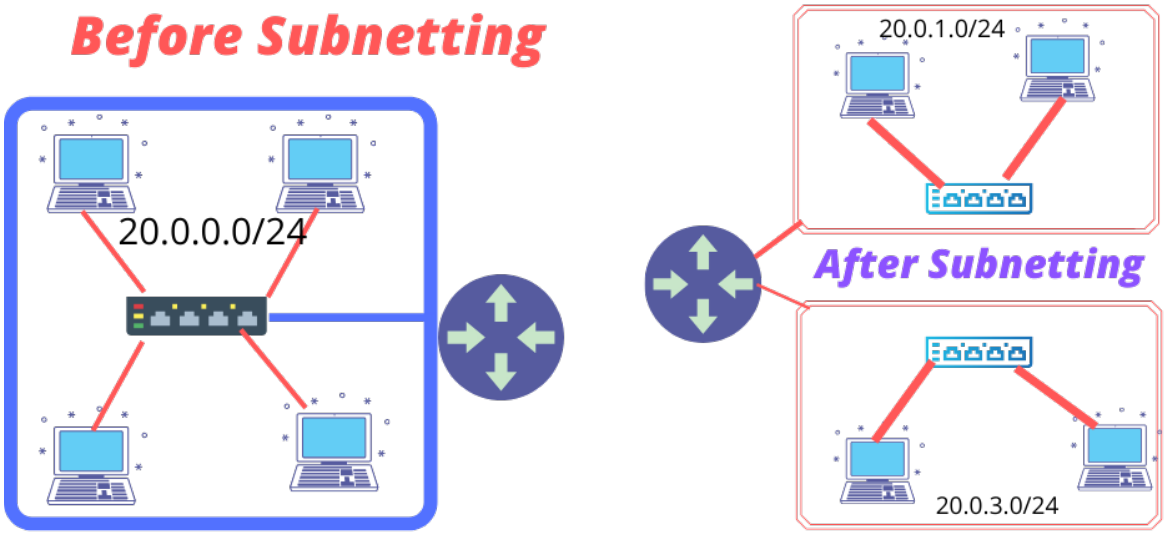


Figure 1: Subnet Example

Layer 2 - Data Link:
For layer 2 of the OSI model, it is only part of a LAN. The way it works is that will allow the network the ability to send and receive messages and packets across a server, while also providing a physical address so the devices can send the information across a network. Some components that are required are Ethernet ports and bridges and network interface cards (NIC). NICs also have assigned MAC addresses that helps with the traffic of information making sure the network doesn't get overloaded with information. Refer to Figure 2 to see how the MAC addresses work in cohesion with the Ethernet switches.

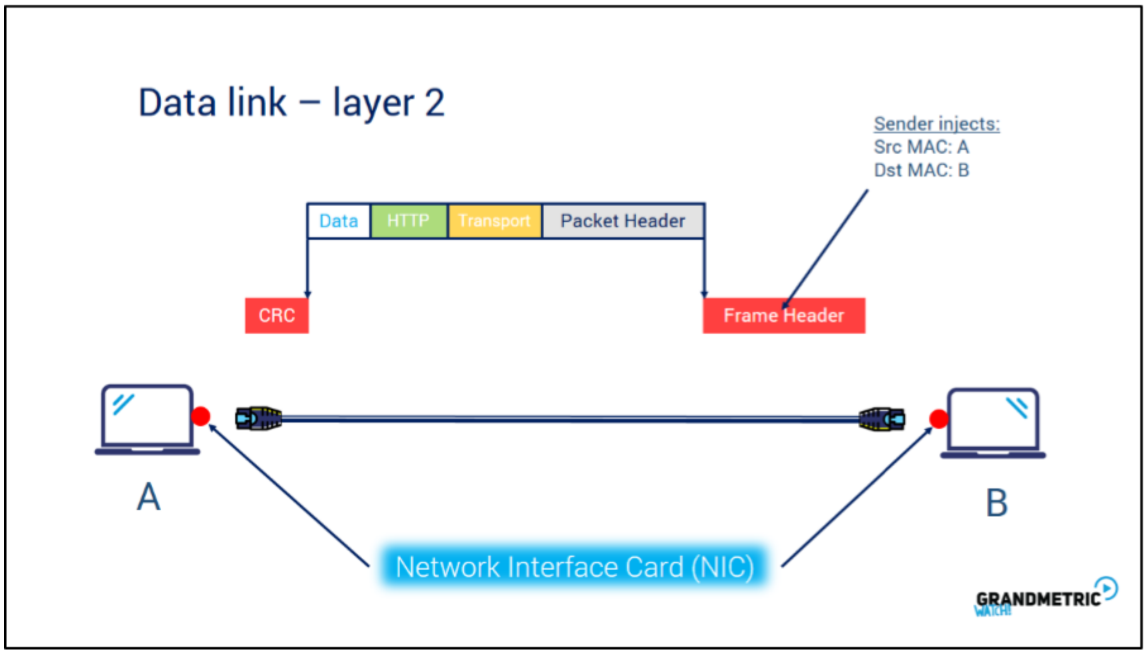


Figure 2: Data link layer working with NIC

Layer 1 - Physical:
The way the physical layer works is taht you have the device send information through the connector which will then go to the patch panel and to the hub and repeater. There are different requirements for the connector, such as the cable that will provide
information routing (ex. CAT 6 cable). There are several electrical, mechanical, functional, and procedural requirements before any information can be relayed onto the network. Some examples of the components required are the following: Adapters that can connect the media, Network Interface Cards, Hub, Repeater, Cabling system materials, devices that can connect to the network.
The way this ultimately helps two applications to communicate is because if host connections are not established, if data can't be translated, and if data can't be transferred or received, how will any information across different applications ever be sent? The OSI model solves all these problems and more, making communication between applications seamless.

Part 2: Virtualization Resource Consolidation:
We have learned that in a distributed system, we partition our resources into computation units. The way this works is that resource utilization can increase and in turn will reduce the cost of the management and associated with the compute processing in different applications, mainly cloud based.

Resource Partitioning:
A lot of large scale solutions can be divided into partitioning to taht way the data can be accessed in separate way. A good example of this is in our systems programming course last year we had to partition our hard drives to allocate our system to be able to use Ubuntu. In doing so, we were able to improve scalability and optimize performance instead of using a Virtual Machine which would have used more computing power.
The reason we partition data is to make sure that we can improve scalability, improve security, improve performance, improve availability, provide flexibility, and match the data to the pattern of use. In doing all these things it would make our usage of the programs and applications much more comfortable.
When we design partitions, there are three strategies: Horizontal partitioning; where we separate the data store but the partitions retain the same schema. Then we have vertical partitioning; where the fields are divided according to the way they are being used. Lastly, we have functional partitioning; where the data is changed in correlation to how the data is being bound in the system itself.

Snow Flocks:
With the issue of needing a cloud server being needed to manage cluster membership and addition of new servers, SnowFlock alleviates these problems with a cloud API call. It uses this API called VM cloning which in turn allows for resource allocation, cluster management, and application logic to apply to these issues immediately.
The way SnowFlock makes its approach is by using the following concepts: Virtualization (which allows for the cloud and machine cloning); Lazy Propagation (VM doesn't clone unless necessary); Multicast (Clones will all have similar attributes); Page Faults (Clone execution with missing memory won't run until page arrives); and Copy on Write (when memory is copied

before being overwritten). With all this, it makes it possible to alleviate stress on the cloud servers and help with ensuring that they do not need to constantly boot new instances from the same template and aids with parallel computing, data mining, and serving web pages.

Generic Homework Questions: What resources are managed and how?
The resources that are managed are the data that is being partitioned. The way it is being managed is by three different ways, one being the SnowFlock which creates the VM cloning, also with partitioning with consolidation as well as improving the information within resource partitioning.

How does the system described in the article provide the following requirements: Resiliency in case of failures?

SnowFlock shows resilience in the case of failure by being the counter measure by using VM Cloning when it is necessary.

Scalability in case of expansion in various parameters?
Because of the necessary mutations and the centralization for network traffic, the SnowFlock method causes issues in expansion in various parameters.

Maintainability in response to continuous changes?
On a large scale change, there will be many failures amongst the VM clones. So the reliability on a large scale is very difficult as the server only cares about maintaining its own flow control But because of using the mcdist system, SnowFlock can scale well