CMPSCI 677 Distributed and Operating Systems **Spring 2019**

Lecture 7: Process, Code and VM Migration

Lecturer: Prashant Shenoy *Scribe:* ***Zeal Shah***

7.1 Announcements

- No class on Tuesday (02/19/2019).
- Guest lecture on Cloud Computing and Data Centers during the class on Wednesday (02/20/2019).
- Homework 2 and Lab 1 are out. Github classroom will be used for labs. Your code development history on Github classroom will be checked by the instructors. You can use Java, C++ or Python to do your labs. Please check with the instructors before you decide on using any other language for the labs.

7.2 Server Design Issues

7.2.1 Request Processing

There are two types of request processing methods in modern servers:

- **Iterative**: The server sits in a while loop serving requests. This method doesn't enable any concurrency and all requests are queued. This method is preferred in single-core single process servers.
- **Concurrent**: When a request arrives, the server will spawn a thread which will process the request. Preferred in multi-core multi-process systems.

7.2.2 How to locate an end-point?

How does the client figure out the port number to send the request to? One way is to hard-code server's IP address and port number into the client application but it is not recommended. A more flexible and better method of designing the system is to use a naming service where the server could register itself.

7.2.3 Stateful or Stateless

Servers can be classified into three categories in terms of state:

- **Stateful:** Maintains state of clients.It can be used to maintain sessions of user activity.
- **Stateless:** Doesn't maintain any client states.
- **Soft State:** Maintains state for limited amount of time by caching the data. Discarding the existing state does not change the correctness of the system.

7.2.4 Clustered Servers

For high volume services, scaling is achieved through tiering/clustering of servers. Each tiered level may be replicated and load is balanced across the clusters through a dispatcher which assigns each incoming request to a server in the cluster. There are two methods for request assignment in clusters:

- **Round Robin:** Requests are assigned in a round robin fashion. This approach isn't ideal or optimal for implementing user based sessions (activity by a single user in a small continuous time period).
- **Session based:** Requests from a user session are routed to the same server in order to maintain state about that session.

TCP splicing can be used for communication between the system and the client in a multi-tiered architecture. Client establishes a TCP connection with the Switch node that is the front node. Switch will then make a TCP connection with an actual server and will forward client's request to that server. Switch acts as an intermediate node that will basically receive requests from the client and will relay them to an actual server. The client can only see the first node to which it sends the request. The server which actually processes client's request remains transparent to the client.

Question: If the client has only established a connection with the switch and not the server then how is the server sending the response directly to the client?
Answer: Server is spoofing the IP address of the switch to keep everything transparent. When server sends the reply, it sends the reply with IP address of the Switch, so the client doesn't know that the response actually came from a third machine.

Question: How is the DNS configured for multiple servers?
Answer: DNS stands for Domain Name Service. DNS allows you to transfer name of the server to its IP address. In the case of clustered servers that we discussed above, url request will point only to the switch because the server replicas are transparent to the outside world and their internal IP addresses are not exposed to the outside world.

7.2.5 Server Architectures

There are primarily four types of server architectures:

- **Sequential**: Serves one request at a time and does not have concurrency. It does not take advantage of multiple cores.
- **Concurrent:** Spawns a new thread or process for each request. It can also assign one thread/process from the pre-spawned pool of threads/processes to an incoming request. Concurrency can be achieved. There exist two variants of concurrent models- thread based and process based. Both the models use master-slave mechanism to service the incoming requests.
- **Event-based Sequential:** Single process server, but is asynchronous and all the calls are non-blocking calls. Since, there is no blocking while a request is being processed, other requests can be processed concurrently. Thus, this model allows you to achieve concurrency without using threads by making the use of asynchronous programming. It is one of the most complicated models to develop.

For a single processor, if context switch overhead is used as a measure of efficiency, event-based architecture wins over process-based and thread-based architectures. Thread-based architecture involves thread context switching which is less expensive in comparison to process context switching. But event-based architecture has no context switching involved because everything happens inside a single process. Event-based architectures are best architectures for writing server applications.

But on the other hand, sequential and event-based being single-threaded cannot take advantage of multiple cores. Thread based and process based architectures can be more efficient in this case because they make use of multiple threads and processes. Additionally, one can develop a hybrid architecture where one thread is assigned to one core and each thread then performs event-based.

7.2.6 Scalability

Server capacity can be scaled by many different ways as follows:

- Buy a bigger machine.
- Replicate application on cluster of machines. Replicate means you run the complete code on multiple machines.
- Distribute data and algorithms. Distribute means you split a task into sub-tasks and run each sub-task on a different machine.
- Ship code instead of data.
- Cache.

Question: If you could distribute your algorithm across machines, would it not have scalability problem of not being replicated?
Answer: You can do both distribution and replication. An example of this is clustered servers. In clustered servers, functionalities are distributed across tiers and each tier is replicated.

Question: What does shipping code instead of data mean?
Answer: Rather than you sending data to the computation for execution, the computation will come where the data is present and will then execute.

7.3 Process, Code and VM Migration

The motivation behind developing techniques for code, process and VM migration is that migrating these components of a system helps in improved performance and flexibility.

There are two types of migration models:

- **Process Migration:** Also known as strong mobility, this includes the migration of all the components of a process i.e. code segment, resource segments and execution segment. An active process (an already executing program) on a machine is suspended, its resources like memory contents and register contents are migrated over to the new machine and then the process execution is restarted. It involves significant amount of data transfer over the network.
- **Code Migration:** Also known as weak mobility. In this model only the code is migrated and the process is restarted from the initial state on the destination machine. The network transfer overhead is low since only the code is transferred. Some examples of code migration are web-form, flash/java applets in browser, web search. Docker is also considered to be an example of code migration.

Question: Why is a search query an example of code migration?
Answer: Keywords typed in a search bar basically become part of a query and query is a program. On pressing submit, the query is sent to another machine and is executed there. This illustrates migration of code from client machine to the server machine.

Question: In process migration, if you suspended an active process and migrated it, how do you take care of its state?
Answer: Specific state of process like its memory contents can simply be written onto a disk and the process can be resumed elsewhere. Debuggers perform a similar operation.

Question: Does the migration have to be only between client and server or can it be between a cluster of servers?
Answer: Migration is not limited to just client and server. Migration is independent of source and destination of code or process.

Examples of sender-initiated migration: Web searches, database queries. Examples of receiver-initiated migration: Browser downloading Java applet or flash application from the server

A process can be migrated or cloned. In case of migration, complete process is moved to a different machine. In cloning, a copy of the process is created on a different machine and you allow both the copies to execute. Cloning is a convenient way of replicating the process. An example for cloning is fork of a process.

To decide whether to migrate a resource attached to a process or not, we look at the nature of binding of resource to process. There are three types of resource to process bindings:

- **Identifier:** Hard binding, one that you cannot substitute. Least flexibility. Example is URL for a website
- **Value:** Slightly weaker binding. Libraries used in Java are a good example.
- **Type:** Weakest binding, one that can be substituted. Maximum flexibility. Example is a local device like printer.

Apart from the resource binding, it is also necessary to look at the cost of moving resources which can also be classified into three categories:

- **Unattached:** Very low cost of moving. Example- files.
- **Fastened:** High low cost of moving. Example- databases.
- **Fixed:** Can't be moved. Example- local devices.

Different combinations of resource-to-machine binding and process-to-resource binding are tabulated below:

	Unattached	**Fastened**	**Fixed**
By Identifier	MV (or GR)	GR (or MV)	GR
By Value	CP (or MV, GR)	GR (or CP)	GR
By Type	RB (or GR, CP)	RB (or GR, CP)	RB (or GR)

where GR means establishing global system-wide references, MV means moving the resources, CP means copying the resource and RB means rebinding process to locally available resource.

Question: Would you want to do checkpoint and restart instead of migrating a process?
Answer: Checkpoint and restart is a standard way to implement process-migration and not an alternative to process-migration.

Heterogeneous systems are systems in which the source and client machines can have different architecture and even different OS. Code-migration is possible if the code we are trying to migrate is interpretable code, for example, Python code. But, if the code is binary code then it will require recompilation because directly migrating binary code will not work. Process migration in heterogeneous systems will require recompilation of code platform, transfer of data and necessary translations. Even after achieving this, the process might work across heterogeneous systems under limitations. On the other hand, migrating Java code across heterogeneous platforms will not be a problem because all JVMs provide the same abstraction even though they are running on different operating systems.

7.4 VM Migration

Process and code migration in heterogeneous systems pose a challenge. Migration via interpreted codes is one possibility but it is often clumsy in practice and is almost never used. Moreover, interpreted code migration only supports weak mobility. In such cases, it is beneficial to look at techniques for VM migration. VMs can be migrated from one machine to another irrespective of architectural differences, without any noticeable down-time. One scenario where VM migration will prove to be helpful is when one VM is running on a server which does not have enough resources to cater to the increased traffic. In this case, one can simply migrate that VM to a bigger server with more resources so that now the VM cans service more requests. VM migration can be live unlike process migration.

There are two methods for VM migration:

- **Pre-copy Migration:** The process of pre-copy migration can be listed down as:
  1. Copy all memory pages to destination
  2. Copy memory pages which were changed during the previous copy
  3. Repeat step 2 until number of memory pages is small.
  4. Stop VM, copy rest of memory pages at destination and start VM at the destination.
  5. Send ARP request to switch.

  Machine has 2 IP addresses- one IP for the actual machine and one logical IP address for the virtual machine running on it. VM's IP address is portable and so it can be decoupled from the source machine and can be sent over to the new machine. This is done soon after the memory contents have been copied at the destination. It is followed by sending an ARP request to inform switch about the change in port number associated with VM's IP address. Thus, now the switch will send all the packets addressed to VM's IP to the new port as opposed to the old port. NOTE: Logical Ethernet card is mapped onto the physical interface card. So, whenever you send a packet, it gets sent to the physical interface which is then delivered to the right logical Ethernet based on its IP address.

- **Post-copy Migration:** The process of post-copy migration can be listed as:
  1. Stop VM
  2. Copy the non-memory VM states to destination. These states are enough to atleast start executing the VM at destination.
  3. Resume VM at destination
  4. Copy memory pages in the background.

  In the last step, if a required memory page is not found, then it will lead to asynchronous page faults. During page faults, VM will stall until the page has been fetched over. Thus, the pages which were not transferred initially can be fetched asynchronously in the background.

Pre-copy migration does not have any performance overhead. But if the memory of your system is changing very quickly, pre-copy migration can take longer time and so it has a high copying overhead.

Post-copy migration has relatively less memory copying overhead because the memory is no longer changing. However, post-copy migration has a higher performance overhead because of the stalls introduced by networked page faults.

Question: How much copying is enough to restart VM at the destination?
Answer: Moving atleast all the registers, a few OS pages that program counter is pointing to, should be good enough to restart the VM. More details depend on the hardware architecture.

Question: Programs often have spatial and temporal locality of references, can we use it to intelligently figure out what to pre-fetch?
Answer: Post-copy migration can make use of such optimizations where for example, one can get the working set of the programs first and then fetch the rest.