CMPSCI 677 Operating Systems **Spring 2024**

Lecture 10: March 06

Lecturer: Prashant Shenoy *Scribe: Keerthy Kaushik Dasoju (2024), Diptyaroop Maji (2023), Samriddhi Raj (2022), Xingda Chen (2019)*

10.1 Migration Introduction

The motivation behind developing techniques for code, process and VM migration is that migrating these components of a system helps improve performance and flexibility. For example, in distributed scheduling, we submit a job on one machine. However, if that machine is overloaded, we would want to migrate either the code (of the submitted job) or the process itself to some other machine and improve performance. From a flexibility standpoint, migration helps in the sense that we can configure distributed systems dynamically. For example, clients can download software on demand from some driver repository and don't need the software to be preinstalled (refer Fig. 10.1).

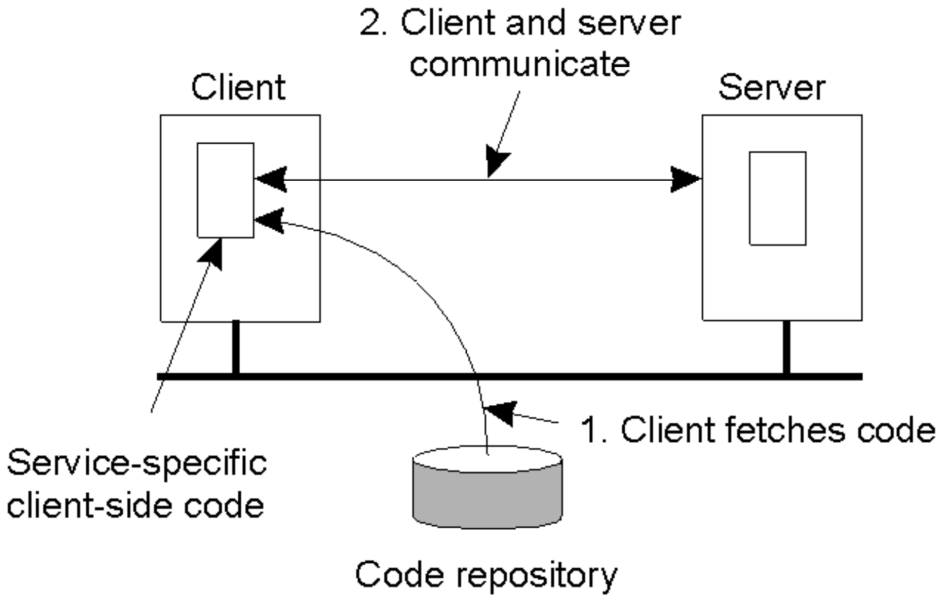


Figure 10.1: Motivation for migration – adds flexibility.

There are two types of migration models:

- **Process Migration (aka strong mobility):** This includes the migration of all the components of a process, i.e., code segments, resource segments and execution segments. 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 (aka 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. Many scenarios map to code migration. For example, filling out a web form and hitting submit is a form of code migration because the form becomes a small piece of code that goes to the server, which then processes it and executes. Another example is doing a web search. Web search can be thought of as a query for some words, and we want all the pages for that query. So, that's a "program" we made. The query actually moves to some other system (in this case, a search engine) and is executed there. Simple examples of code migration are taking a real Java program or a binary, moving it and executing it in another machine. Docker is also considered to be an example of code migration. Additionally, anything that is "download and execute" is also an example of code migration — for example, downloading device drivers on demand for new hardware.

  We can use code migration to get better parallelism (replicating same code across machines).

Question: When you do code migration, are you doing process migration?

Answer: Code migration simply means moving files, not processes. For example, Python code is a file. If we want to execute it remotely, we will copy that program somewhere else and run it. The process is created when we start executing that code on a remote machine.

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: The 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 the source and destination of the code or process.

10.1.1 Migration models:

Migration can be sender-initiated or receiver-initiated. An example of receiver-initiated migration is a browser downloading a Java applet or Flash application from the server. In sender-initiated, the sender process has the code and sends it somewhere else. An example of sender-initiated migration is a web search and database query. Fig. 10.2 shows a flowchart of the migration models.

A process can be migrated or cloned. In the case of migration, the complete process is moved to a different machine. In cloning, a copy of the process is created on a different machine, and we kill the process in the sending machine and start executing the process in the receiving machine. We can also allow both copies to execute. Cloning is a convenient way of replicating the process. An example of cloning is forking a process.

Question: What does it mean to "execute at target process" or "execute in separate process"?

Answer: Let's take an example to understand this. Say, we have a browser that contacts a server. The server sends the browser a small piece of code (say, JavaScript). After migration, if the Javascript executes in the same process of the browser, then it is "execute at target process". In contrast, if the browser process created a second process and Javascript was being executed in that second process, it is "execute in separate

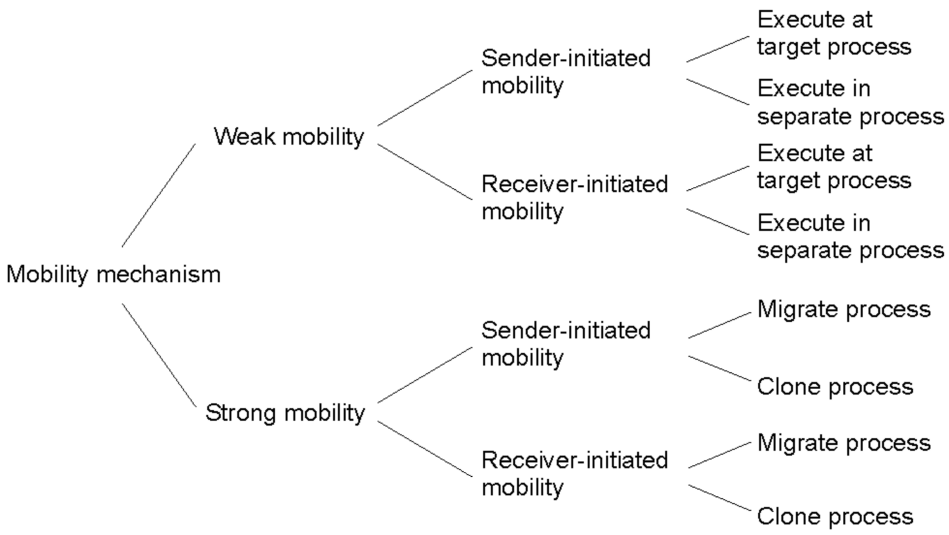


Figure 10.2: Migration models.

process".

Question: Whose decision is it to decide whether to do it in the target process or a separate process? Is it the receiver or the sender?

Answer: Typically it's neither. It is the developer's decision (application-level decision).

10.1.2 What happens to the resources that the process was accessing?

Let's say a process migrated to a remote machine was accessing a file on the local machine. The process needs to read/write to the file, but it is still in the previous machine. If the file is not present on the new machine, the process will throw an error. So, we need to deal with all kinds of resources that the process was accessing when we deal with code or process migration. How we are going to handle resources depends on what type of resource it is. The process must continue to have access to the resources even after migration to avoid any errors.

We classify the resources along two dimensions. To decide whether to migrate a resource attached to a process or not, first, we look at the nature of binding of resource to a process. There are three types of resources to process bindings:

- **Identifier:** Hard binding, one that you cannot substitute. Least flexibility. An example is URL for a website.
- **Value:** Slightly weaker binding. We can substitute it with another similar resource. Libraries used in Java are a good example. If the Java process was using JVM in the previous machine, as long as the new machine has a JVM of the same (or compatible) version, the Java process will work fine.
- **Type:** Weakest binding. We can substitute it with another resource which need not be exactly similar. Maximum flexibility. An example is a local device like a printer — different printer models will work just fine as long as there is a printer ready to accept print jobs.

Second, it is also necessary to look at the cost of moving resources. This can also be classified into three categories:

- **Unattached:** Very low cost of moving. E.g., files.
- **Fastened:** High cost of moving. E.g., databases. Databases can be moved, but if their size is large, moving them is expensive (in terms of time, or, network bandwidth cost, etc.)
- **Fixed:** Can't be moved. E.g., an ethernet card or an IP address.

10.1.3 Resource Migration Actions:

Different combinations of resource-to-machine binding (columns) and process-to-resource binding (rows) 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 (resource can't be moved, so giving it remote access), MV means moving the resources, CP means copying the resource and RB means rebinding process to locally available resource.

10.1.4 Migration in heterogeneous systems:

Here, the assumption is that the sender and receiver machines are heterogeneous (different hardware/OS etc.). If the OSes are different (say, Windows and Linux), we cannot just copy the memory of a Windows process to the Linux machine and expect it to run, due to different binary formats in the different OSes. If we have different hardware then the instruction sets will be different (say, Intel vs ARM). Thus, process migration is much harder due to these constraints. Code migration is easier. For example, as long as Python is installed on both machines, a Python program written on a Linux machine will run fine on a Mac.

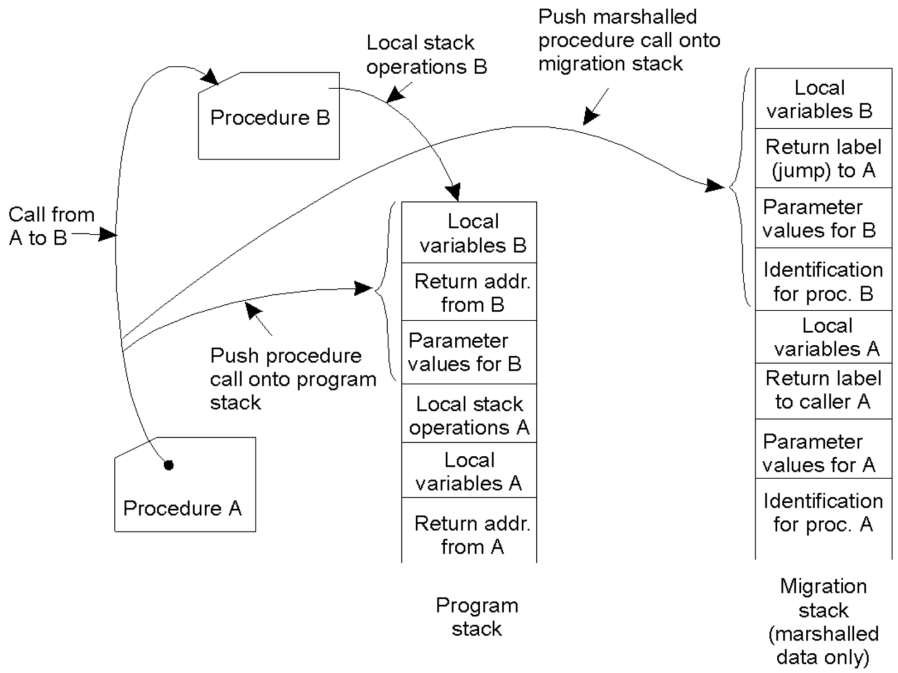
Only if we can abstract out the differences in OS/hardware, we can migrate processes. For example, Java. Java runs on JVM which is platform independent and abstracts out the OS/hardware differences. So, a Java process running on Windows can still run on some other platform.

Question: When you get a Docker image is that process migration or is that code migration?

Answer: If we take a Docker image and start it, that's code migration because we got the code for the program. It's just packaged in a container. On the other hand, if we have a running container and we move that running container to another machine, that's process migration because the container actually has active processes running.

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.

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.



Question: If JVM is a value resource, why can't we bind JVM of the new machine to newly migrated process ?
Answer: If exact same version of JVM as required by the process is available on the new machine, then we can bind it to the process. However a different JVM version can be incompatible and cause the program to crash.

10.2 Virtual Machine Migration

If a process is communicating with some other process, then process migration is much harder as the IP addresses of the sender and receiver machines are different and we cannot move IP addresses across physical machines. Virtual machine migration (VMM) gives us a way we can actually deal with even network connections of a process. Even the network connections will move when we migrate a VM. The IP address will also move and it will not change.

VMs can be migrated from one machine to another, irrespective of architectural differences. A VM consists of its OS and some applications running on this OS. So, in VM migration, the OS and these applications are migrated with negligible downtime. As the processes inside a VM will also move, VM migration also involves process migration. VM migration is usually done live; that is, it keeps executing during migration. Applications continue to run, nothing has gone down, and then after a while, the VM disappears from one machine and shows up on another machine with the applications still continuing to run.

There are two methods for VM migration which are Pre-copy migration and Post-copy migration.

Pre-copy Migration: Fig. 10.3 shows pre-copy VMM (A and B are two physical machines). The process

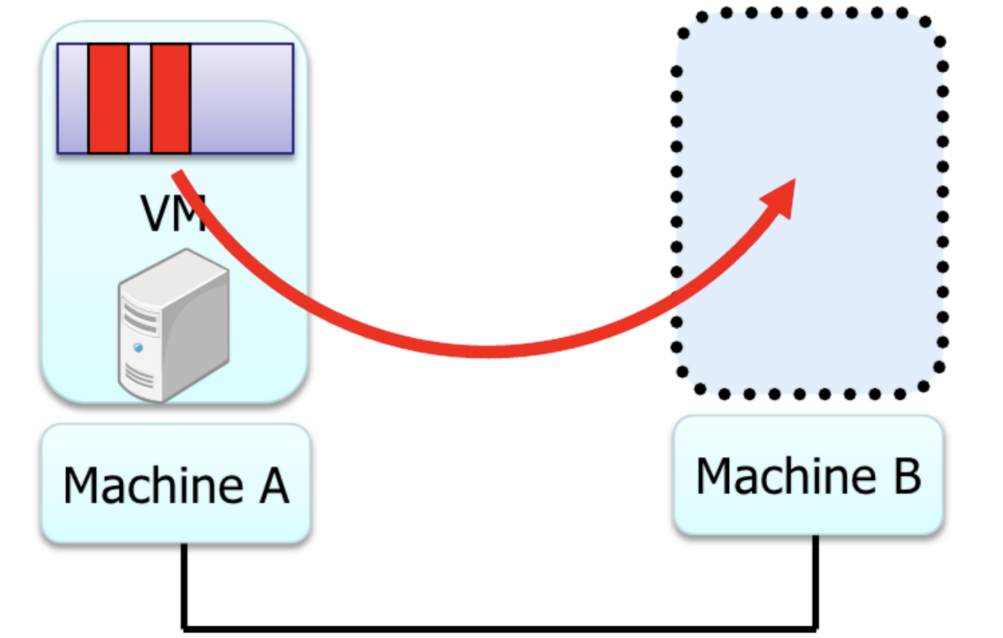


Figure 10.3: Pre-copy VMM.

of pre-copy migration involves the following steps:

1. Enable dirty page tracking. This is required to keep a track of pages which have been written to.
2. Copy all memory pages to destination.
3. Copy memory pages which were changed during the previous copy.
4. Repeat step 2 until the number of memory pages is small.
5. Stop VM, copy rest of memory pages at destination and start VM at the destination.
6. Send ARP packet to switch

Question: Is it a fair operation to assume that copy operations are faster than memory updates?

Answer: No, memory updates are much faster than copy operations.

Question: When copying in an operating system, is the entire thing is updated again?

Answer: In operating systems, there's a concept known as the working set of processes, which refers to a set of memory pages actively used by a process. Despite a process being allocated a significant amount of memory, it typically only accesses or modifies a small portion of it at any given time. Therefore, when copying data, it's not necessary to update the entire memory space; only the portions that have been modified need to be copied. In essence, only the relevant memory pages are changed during the copying process, rather than the entire memory space.

Question: What happens in 2nd round if a page which was not changed in round 1 got changed?

Answer: In the 2nd round, that page will also be considered. Our main goal here is to select all the pages which have a dirty mark on them and send them to another machine. However, it's worth noting that our assumption remains valid as the number of modified pages typically decreases from round to round.

Question: How do we qualify the downtime?

Answer: From the application perspective there is a downtime in this process but this is very small may be in milli-seconds which is not perceivable from the user's perspective.

Question: What happens if the page writes are many?

Answer: Despite the potential for many page writes, ultimately only a single page is changed. Therefore, the number of pages changed is not affected by the frequency of page writes. Consequently, there should be no issue with this pre-copy method. However, if there are random writes to different pages, the assumption that our converging changed pages process holds true is invalidated, rendering the process ineffective.

Question: Is there any synchronization here?

Answer: No, synchronization is not involved in this process. We are simply moving the memory content while keeping track of changed pages and moving those pages in the second round, continuing the process further.

Question: When a page changes are you going to send the entire page or are you only going to send what part of the page that changed?

Answer: In this case we'll only track dirty pages and send the entire page. If we want to send only the diff, then we have to keep the old copy and the new copy and do a difference, and that means even more memory allocation and added complexities.

Question: Till when do we repeat Step 4 above?

Answer: We assume that we send all the pages the first time. Before sending again, only a fraction of the pages change. Since it is a subset, it will take less time than the previous round, as the network bandwidth is fixed. This is true for all rounds; that is, each round takes less time than its previous round (assuming no adversarial application). Eventually, after n iterations, we will have a small amount of memory that we need to move. At that point, we go to step 5, that is, stop the VM and copy the remaining pages to the destination physical machine. Since we stopped the VM, no new pages were dirtied during this round.

Note that although it says "live" migration, there is a small time during when the VM is stopped.

Question: What is the threshold where the stop time is small?

Answer: That's configurable. We want the pause time to be as small as possible so that it's almost not noticeable. So, it will depend on factors like the network bandwidth between the two machines.

Question: What happens if there's a network failure?

Answer: If anything fails, VM migration will stop, and it will not continue.

Handling "IP migration": The host machines have different IPs. This is why process migration doesn't work. However, VM migration works because now there are two different IPs — VM IP and host IP — and we can actually keep the VM IP the same while migrating the VM. The applications running in the VM are coupled with the VM IP, and hence these connections do not break even after VM migration. For everything to work after migration, there is one last step — an ARP packet is sent to the ethernet switch to update the ARP-MAC mapping (although IP remains the same, MAC address has changed since physical hosts are different).

Question: Is the IP address is the VM public or private?

Answer: It doesn't matter. The applications are actually tied to the VM IP address and not the host IP address. The only thing is VM IP should be different from the host IP (source/destination host). The application connections will not break as VM IP remains the same.

Post-copy Migration: This is also called lazy copy. The process of post-copy migration involves the following steps:

1. Stop VM and move non-memory VM states to destination
2. Start executing on new machine
3. In case of page faults in the new VM, copy the page from the source machine. Copying of other pages is also started in the background. Background copying also ensures that every page is copied even if it was required during a page fault before the VM is deleted from source.

One advantage of post-copy over pre-copy is that there is no iterative copying and each page has to be fetched only once. Here, we have a trade-off between memory overhead and performance overhead. Pre-copying is preferred in some cases when we do not want applications to keep waiting in case of page faults so there is less impact on application performance.

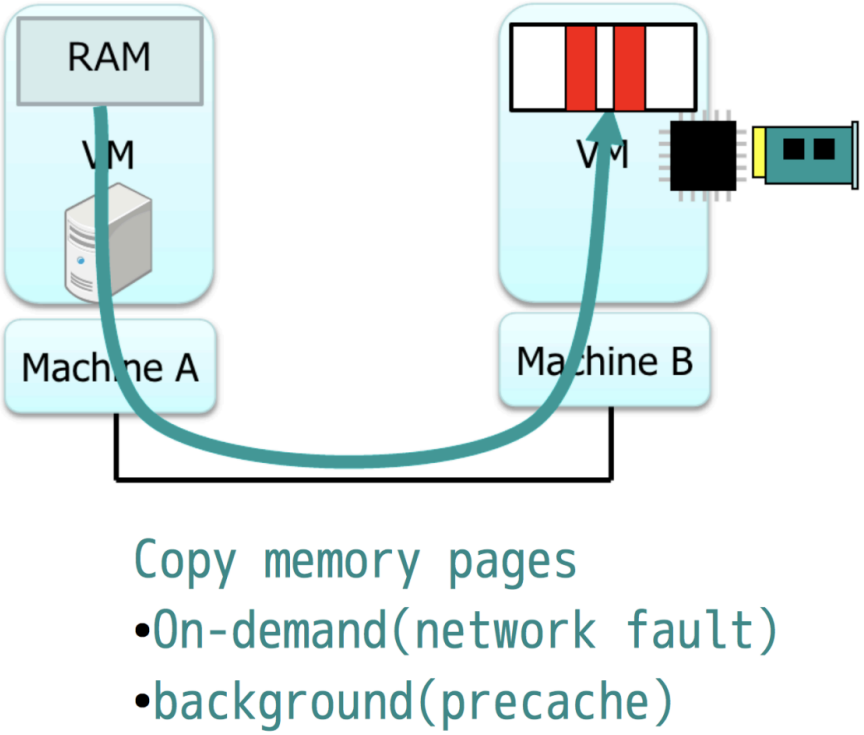


Figure 10.4: Post-copy VMM.

Question: Can a Machine B be part of different network or do they have to share a switch ?

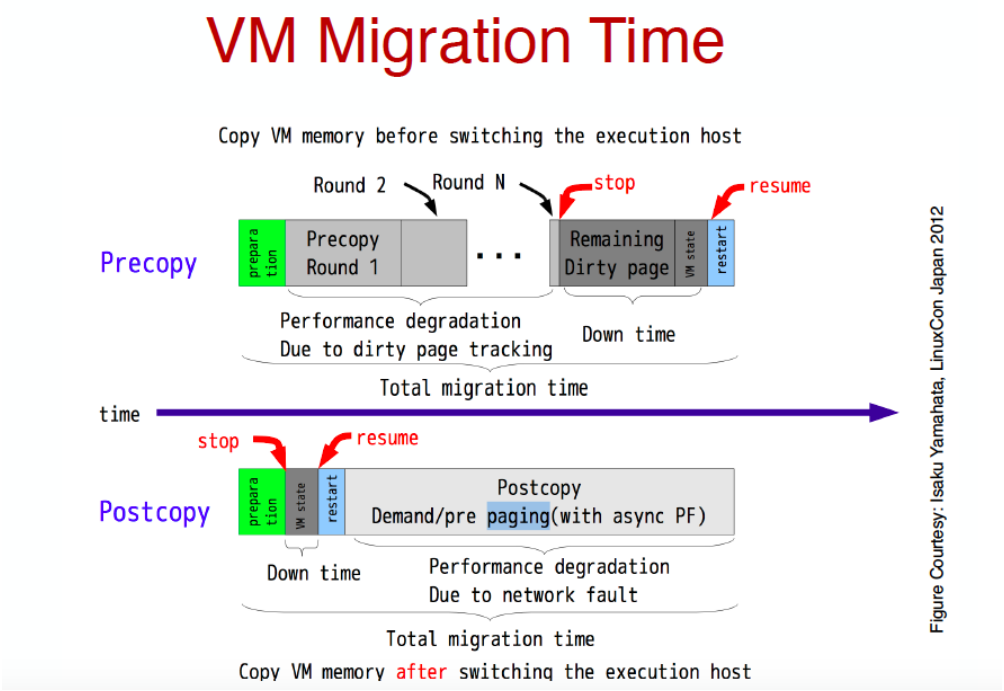


Figure 10.5: Visualization and comparison of pre-copy and post-copy VM Migration.

Answer: This process works on one cluster.You are not migrating In this process something on the internet to some completely different location.that is WAN. What here is LAN migration(Local Area Network).So here it assumes that Machine A and B are part of the same LAN.Presumably they are part of the same subnet. They don't have to be on the same switch. In WAN,In a complete different address space,it won't work since routers have to send packets to the right port.Only works for Local Area Migration.

Question: What about the files which are on the disk?
Answer: files are storing in virtual disk. 1)virtual disk image is stored on the file server. Can simply access from the file server from a different machine.Most common and fast approach. 2)virtual disk image is local to machine A. We need to move the virtual disk as well. Copy the file page by page by precopy mechanism.Slow process.

Question: Given that it is happening on cluster, Is there any restriction on hypervisor or OS?

Answer: Only restriction is VM should be executable on the new hypervisor.We assumes that machines are running the same hypervisor.

Question: What are the pros and cons of precopy and postcopy migrations? **Answer**: In post copy,since you are fetching memory on demand,Its gonna take some time to fetch each page since the process which got faulted when it was trying to execute and access that page gonna pause untill that page is brought over.So it causes some performance penality.But each page is transferred exactly once.

In precopy, there is no performance penality.But the pages has to trasnfer multiple times if it has changed after being copied over.The total amount of data copying in precopy is going to be much higher.Also there is a small chance that precopy may not terminate,especially when the VM behave badly,it keeps churning

memmory and after each round you have transfer a huge amount of memory and it never converge.

Question:Why do you have to tranasfer the VM?

Answer: Vm can allow you to increase the resources When your application get overloaded.So load balancing is a good reason.

Question: In post copy ,can there be a scenario where some pages are never accessed by processes and no page faults?

Answer:To prevent these scenarios, there are 2 ways we are bringing memory, 1)Fault on Demand 2)A background process which is fetching remaining pages even if they are not faulted on.

Question:In post copy ,are files migrated in the same manner?

Answer:you can assume the files are on the server and don't move it at all. or can assume that files are on the virtual disk local to the VM and either move it over through postcopy or precopy. easiest approach is keep things on the file server and don't move.

Question: How much copying is enough to restart VM at the destination?
Answer: Moving at least 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.

10.3 Container Migration

Containers are light weight VMs. When you are migrating a container you are migrating only the processes and some resources that it accesses. The underlying OS is not getting migrated.

10.3.1 Types of Migration

There are three types of migration:

Cold Migration

1. VM is not running. There is just an image of VM/container or virtual image of VM/container on the disk.
2. Copy this image and data files to new machine.
3. Start on new machine.
4. No state preserved.

Warm Migration

Incurs downtime to migrate state from previous instance, but preserves the state.

1. Suspend running VM/container to disk.
2. Copy image, data, suspended memory state.
3. Resume execution of suspended VM.

Hot/Live Migration

Migrate state but with no downtime so copy state while VM executes. Most complex to execute.

10.3.2 Snapshots

A snapshot is a copy of some object (file, disk, VM, container) at a certain point in time (point-in-time copy). We can preserve the contents of the VM(the memory and disk content) as they existed at that point, if one takes a snapshot of a VM at a certain instant. Snapshots are also known as checkpoints. Snapshots help in rolling back to a point in time and creating backups.

There are two ways to create a snapshot.

1. Full (Real) Snapshots - Actually make a real copy. Very inefficient.
2. Virtual Snapshots - Here a virtual copy is created. Instead of making a real second copy of a file we just copied the metadata and all of that is pointing to the previous copy. The previous copy can continue to change because the VM can write on the disk. So, in case of virtual snapshots, copy-on-write is used. Whenever a VM is about to write to a previous copy, a new copy is created first and pointers are shifted to this new copy. Virtual snapshots are very efficient as compared to full snapshots.

Question: Is the snapshots are periodic diff?
Answer:Periodic diffs are not necessarily copy and write. copy and write is diff on demand.You preserve the copy of the page and then you write.

Question: Snapshots are useful for migration. Is it only valid for real snapshot real copies as opposed to a virtual copy?
Answer: Even in a virtual copy you have access to all of the data because it's just pointing to the same blocks. So, it will not matter whether you took a virtual snapshot or a real copy for migration purposes.

10.3.3 CheckPoint and Restore

This is a warm container migration technique. Many containers actually support checkpoint and restore as the first option because live migration is a bit more complicated. Migration in containers is a little more complicated than VMs as in containers we only migrate processes. Steps of CheckPoint and Restore are :

1. Pause container execution.
2. Checkpoint (save) memory contents of container to disk.
3. Copy checkpoint to new machine (memory + disk image).
4. Resume execution on new machine.

Question: Is a checkpoint the same as a snapshot?
Answer: Checkpoints are like real snapshots.

10.3.4 Linux CRIU

1. Linux CRIU (Checkpoint Restore In User Space) : It is used for warm or live migrations, snapshots and debugging. CRIU is not a container-specific technique, it is a process-specific technique. As a container is a collection of processes, it suspends all of the processes in the container and writes it out on the disk one by one.
2. CRIU uses `/proc` file system to gather all info about each process in the container. In the proc file linux keeps its OS data structures. It's like a file system—you can go and look at the files. There's information about every active process in the system.
3. CRIU copies saved state to another machine.
4. CRIU restorer
   (a) Use fork to recreate processes to be restored
   (b) Restorer also restores the resources being used by processes; for container, restores namespace
   (c) If any network connections were being used, we have to do extra work if we have migrated the container to a new machine. We can migrate active sockets only if the IP address moves along with the container to the new machine. To do so, we can use virtual network devices in containers and move them.

Question: Why can't the process bring the IP address of the previous machine?
Answer: We can not do this because the socket connection is always tied to the IP address on which socket was established.

Question: If we assign a new IP address to the container, how to make sure it is unique?
Answer: We can have either a public IP or a private IP. If it's a public IP by definition it has to be unique. Many containers will not have a second public IP. They'll give themselves a 192 address which is a private IP address. So when you move a container over a new machine, to make this private IP work, the new machine should not have the same IP address running on it already.

10.3.5 Case Study: Viruses and Malware

- Viruses and malware, classified as code migration, propagate between machines, often initiated either by the sender or the receiver.
- Sender-initiated threats, such as proactive viruses, actively seek out vulnerable machines to infect, deploying autonomous code to accomplish their objectives.
- Receiver-initiated threats occur when users unknowingly trigger malicious code by interacting with infected web URLs or opening email attachments containing malware.