Code Migration

Passing programs, sometimes while they are executing.

Reasons for migrating code

Early Form: Process migration, entire processes were moved one node to another. It is a costly and intricate task. Reason has always been **performance**. The basic idea is that overal system performance can be improved if processes are moved from heavily loaded machines to lightly loaded machines. Load is often expressed in terms of CPU queue length or CPU utilization and some other factors. Process migration was not found to be viable after some time.

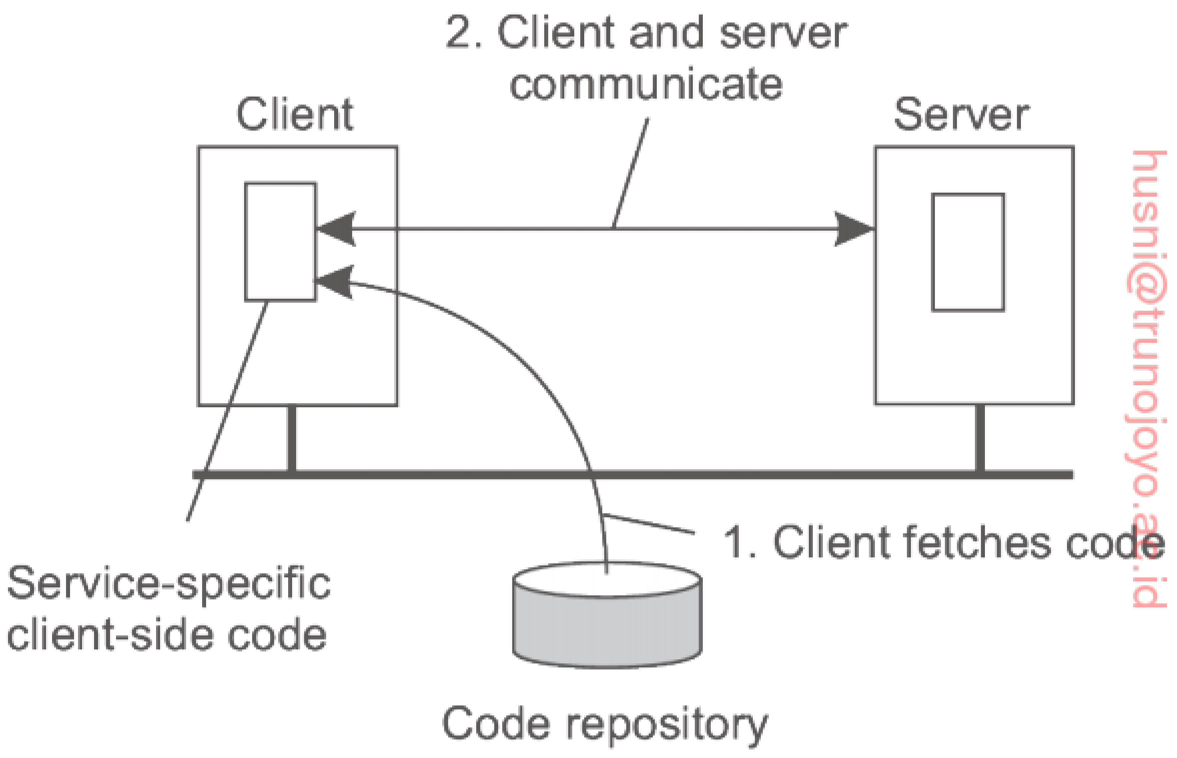
Instead of offloading machines, code is moved. Migrating complete virtual machines with their suite of applications to lightly loaded in order to minimize the total number of nodes being used is a common practice in optimizing energy usage in data centers.

`complexity for migrating process >>> complexity for migrating VM`

Example: Consider, a client-server system in which the server manages a huge database. If a client application needs to perform many database operations involving large quantities of data, it may be better to ship part of the client application to the server and send only the result across the network. In this case, code migration is based on the assumption that it generally makes sense to process data close to where that data resides. The same reason can be used for migrating parts of the client to the server. Example: Sending out a complete form instead of performing single database operations.

Support for code migration can also help improve performance by exploiting parallelism, but without the intricacies involved with it. Example: Searching for information on the web. Implementing search query in the form of a small mobile program, called a **mobile agent**, that moves from site to site. By making several copies of such a program, and sending it off to different sites simultaneously in parallel can result in linear speedup.

Besides performance, **flexibility**. Distributed computing is partitioning the application into different parts, and deciding on advance where each should execute. However, if code can move between different machines, it becomes possible to dynamically configure distributed systems.


2. Client and server communicate
Client
Server
1. Client fetches code
Service-specific client-side code
Code repository

In the above, the client fetches the implementation, intializes, and subsequently invokes the server. Protocol for downloading and initializing should be standardized (generic). Also, it is neccessary that the downloaded code can be executed on the client's machine.

Advantages of dynamically loading client-side software

- Clients need not have all the software preinstalled to talk to servers. Instead, software can be moved in as necessary, and likewise, discarded when no longer needed.
- As long as the interfaces are standardized, we can change the client-server protocol and its implementation as often as we like. Changes will not affect existing client apps that rely on the server.

Disadvantages

- Blindly trusting the downloaded code can cause serious harm to unprotected data.

Migration in heterogeneous systems

Heterogeneous systems = systems with different OSes and machine architectures.

Problems similar to problems of portability. Same case with solutions. Languages like Java, Pascal implement some of those solutions (machine independent code for abstract virtual machines).

Solutions have been proposed to not only migrate processes, but also complete computing environments. The basic idea is to compartmentalize the overall environment and to provide processes in the same part their own view on their computing environment. That compartmentalization takes place in the form of VM monitors running an OS and a suite of applications.

With VM migration, it becomes possible to decouple a computing environment from the underlying system and actually migrate it to another machine. A major **advantage** of this approach is that process can remain ignorant of the migration itself: they need not be interrupted in their execution, nor should they experience any problems with used resources.

Migrating the entire memory image

3 ways to handle migration:

- Pushing memory pages to the new machine and resending the ones that are later modified during the migration process.
- Stopping the current VM; migrate memory, and start the new VM.
- Letting the new VM pull in the new pages as needed, that is, let the processes start on the new VM immediately and copy the memory pages on demand.

The 2^{nd} option can lead to unacceptable downtimes, if the running VM is a live service.
The 3^{rd} option may extensively prolong the migration period, but may also lead to poor performance because it takes a long time before the working set of the migrated processes has been moved to a new machine.

ALTERNATIVE: Pre-copy approach which combines the 1^{st} option, along with a brief stop-and-copy phase as represented by the 2^{nd} option.