Parallel & Distributed System

Server Clusters & Code Migration

Md. Biplob Hosen

Lecturer, IIT-JU

Email: biplob.hosen@juniv.edu

Contents

- **Servers Clusters**
  - ✓ Local-area Clusters
  - ✓ Wide-area Clusters
- **Code Migration**
  - ✓ Reasons for Migrating Code
  - ✓ Migration in Heterogeneous Systems

- Reference Books
  - ✓ Distributed Systems: Principles and Paradigms, 3rd Edition by Andrew S. Tanenbaum & Maarten van Steen, Publisher: Pearson Prentice Hall **[CH-03]**.

Server Clusters: Local-area Clusters

- Simply put, a server cluster is nothing else but a collection of machines connected through a network, where each machine runs one or more servers.
- The server clusters that we consider here, are the ones in which the machines are connected through a local-area network, often offering high bandwidth and low latency.

General organization:

- In many cases, a server cluster is logically organized into three tiers, as shown in the figure.

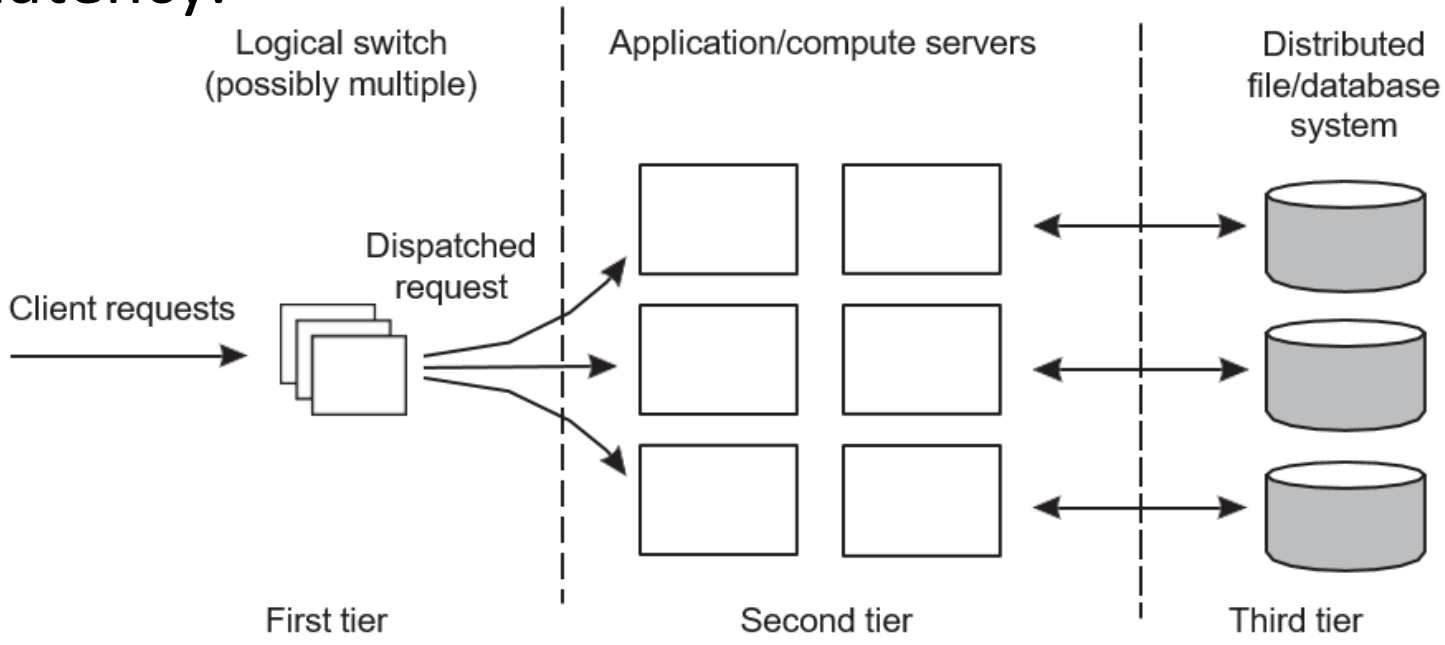

Figure. The general organization of a three-tiered server cluster.

Continue…

- The first tier consists of a (logical) switch through which client requests are routed. Such a switch can vary widely.
- For example, transport-layer switches accept incoming TCP connection requests and pass requests on to one of servers in the cluster.
- A completely different example is a Web server that accepts incoming HTTP requests, but that partly passes requests to application servers for further processing only to later collect results from those servers and return an HTTP response.
- In the case of enterprise server clusters, it may be the case that applications need only run on relatively low-end machines, as the required compute power is not the bottleneck, but access to storage is. This brings us the third tier, which consists of data-processing servers, notably file and database servers.

Continue…

Request dispatching:

- Let us now take a closer look at the first tier, consisting of the switch, also known as the front end.
- An important design goal for server clusters is to hide the fact that there are multiple servers.
- This access transparency is invariably offered by means of a single access point, in turn implemented through some kind of hardware switch such as a dedicated machine.
- The switch forms the entry point for the server cluster, offering a single network address.
- For scalability and availability, a server cluster may have multiple access points, where each access point is then realized by a separate dedicated machine. We consider only the case of a single access point.

Continue…

- A standard way of accessing a server cluster is to set up a TCP connection over which application-level requests are then sent as part of a session. A session ends by tearing down the connection.
- In the case of transport-layer switches, the switch accepts incoming TCP connection requests, and hands off such connections to one of the servers. There are essentially two ways how the switch can operate.
- In the first case, the client sets up a TCP connection such that all requests and responses pass through the switch. The switch, in turn, will set up a TCP connection with a selected server and pass client requests to that server, and also accept server responses (which it will pass on to the client).
- In effect, the switch sits in the middle of a TCP connection between the client and a selected server, rewriting the source and destination addresses when passing TCP segments.
- This approach is a form of **network-address translation (NAT).**

Continue…

- Alternatively, the switch can actually hand off the connection to a selected server such that all responses are directly communicated to the client without passing through the switch.
- The principle working of what is commonly known as **TCP handoff** is shown in the following figure.

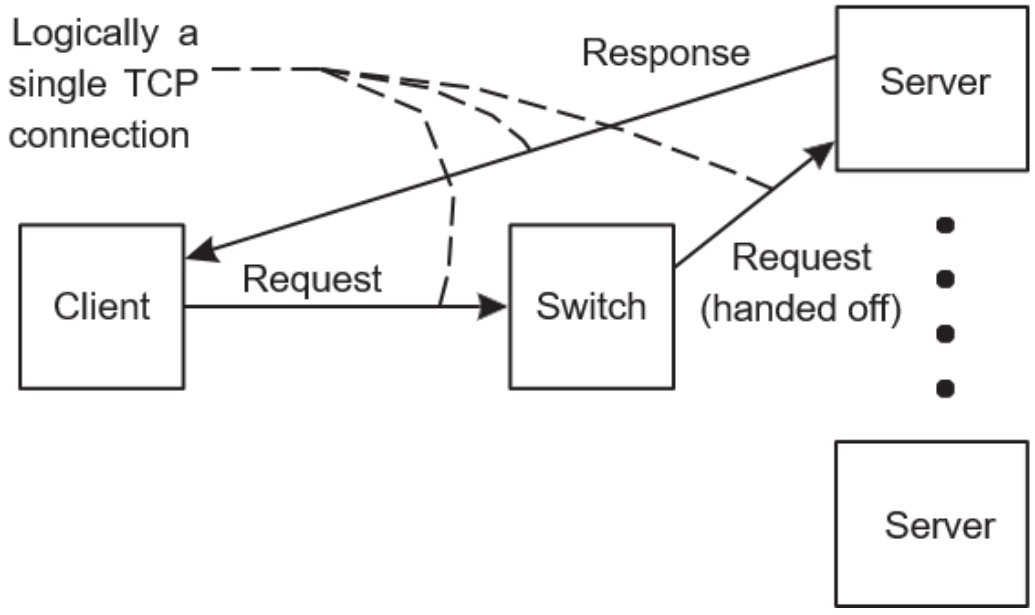

Figure. The principle of TCP handoff.

- When the switch receives a TCP connection request, it first identifies the best server for handling that request, and forwards the request packet to that server.
- The server, in turn, will send an acknowledgment back to the requesting client, but inserting the switch's IP address as the source field of the header of the IP packet carrying the TCP segment.

Continue…

- More advanced server selection criteria can be deployed as well. For example, assume multiple services are offered by the server cluster.
- If the switch can distinguish those services when a request comes in, it can then take informed decisions on where to forward the request to.
- This server selection can still take place at the transport level, provided services are distinguished by means of a port number.
- In the case of transport-level switches, decisions on where to forward an incoming request is based on transport-level information only.
- One step further is to have the switch actually inspect the payload of the incoming request. This content-aware request distribution can be applied only if it is known what that payload looks like.
- For example, in the case of Web servers, the switch can expect an HTTP request, based on which it can then decide who is to process it.

Server Clusters: Wide-area Clusters

- A characteristic feature of local-area server clusters is that they are owned by a single organization.
- Deploying clusters across a wide-area network has traditionally been quite cumbersome as one had to generally deal with multiple administrative organizations such as ISPs (Internet Service Providers).
- With the advent of cloud computing, matters have changed and we are now witnessing an increase of wide-area distributed systems in which servers are spread across the Internet.
- The problems related to having to deal with multiple organizations are effectively circumvented by making use of the facilities of a single cloud provider.

Continue…

- Cloud providers like Amazon and Google manage several data centers placed at different locations worldwide. As such, they can offer an end user the ability to build a wide-area distributed system consisting of a potentially large collection of networked virtual machines, scattered across the Internet.
- An important reason for wanting such distributed systems is to provide locality: offering data and services that are close to clients.
- An example where such locality is important is streaming media: the closer a video server is located to a client, the easier it becomes to provide high-quality streams.

Continue…

Request dispatching:

- If wide-area locality is an issue, then request dispatching becomes important: if a client accesses a service, its request should be forwarded to a nearby server, that is, a server that will allow communication with that client to be fast. Deciding which server should handle the client's request is an issue of redirection policy.
- If we assume that a client will initially contact a request dispatcher analogous to the switch of local-area clusters, then that dispatcher will have to estimate the latency between the client and several servers.
- Once a server has been selected, the dispatcher will have to inform the client. Several redirection mechanisms are possible.

Continue…

- A popular one is when the dispatcher is actually a DNS name server. Internet or Web-based services are often looked up in the Domain Name System (DNS).
- A client provides a domain name such as service.organization.org to a local DNS server, which eventually returns an IP address of the associated service, possibly after having contacted other DNS servers.
- When sending its request to look up a name, a client also sends its own IP address (DNS requests are sent as UDP packets).
- In other words, the DNS server will also know the client's IP address which it can then subsequently use to select the best server for that client, and returning a close-by IP address.

Code Migration

- So far, we have been mainly concerned with distributed systems in which communication is limited to passing data.
- However, there are situations in which passing programs, sometimes even while they are being executed, simplifies the design of a distributed system.
- A typical example is searching for information in the Web.
- It is relatively simple to implement a 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 each off to different sites, we ma y be able to achieve a linear speedup compared to using just a single program instance.

Reasons for Migrating Code

- Traditionally, code migration in distributed systems took place in the form of process migration in which an entire process is moved from one machine to another.
- Moving a running process to a different machine is a costly and intricate task, but the reason has always been performance.
- The basic idea is that overall system performance can be improved if processes are moved from heavily-loaded to lightly-loaded machines.
- Load is often expressed in terms of the CPU queue length or CPU utilization, but other performance indicators are used as well.
- Process migration is no longer a viable option for improving distributed systems.
- However, instead of offloading machines, we can now witness that code is moved to make sure that a machine is sufficiently loaded.

Continue…

- Consider, as an example, 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 is better to ship part of the client application to the server and send only the results across the network.
- Otherwise, the network may be swamped with the transfer of data from the server to the client.
- In this case, code migration is based on the assumption that it generally makes sense to process data close to where those data reside.
- This same reason can be used for migrating parts of the server to the client. For example, in many interactive database applications, clients need to fill in forms that are subsequently translated into a series of database operations.
- Processing the form at the client side, and sending only the completed form to the server, can sometimes avoid that a relatively large number of small messages need to cross the network.

Continue…

- If code can move between different machines, it becomes possible to dynamically configure distributed systems.
- For example, suppose a server implements a standardized interface to a file system. To allow remote clients to access the file system, the server makes use of a proprietary protocol.
- Normally, the client-side implementation of the file system interface, which is based on that protocol, would need to be linked with the client application.
- This approach requires that the software be readily available to the client at the time the client application is being developed.
- An alternative is to let the server provide the client's implementation no sooner than is strictly necessary, that is, when the client binds to the server.
- At that point, the client dynamically downloads the implementation, goes through the necessary initialization steps, and subsequently invokes the server. This principle is shown in the following figure.

Continue…

- This model of dynamically moving code from a remote site does require that the protocol for downloading and initializing code is standardized.
- Also, it is necessary that the downloaded code can be executed on the client's machine.
- Typically, scripts that run in a virtual machine embedded in, for example, a Web browser, will do the trick.
- Arguably, this form of code migration has been key to the success of the dynamic Web.

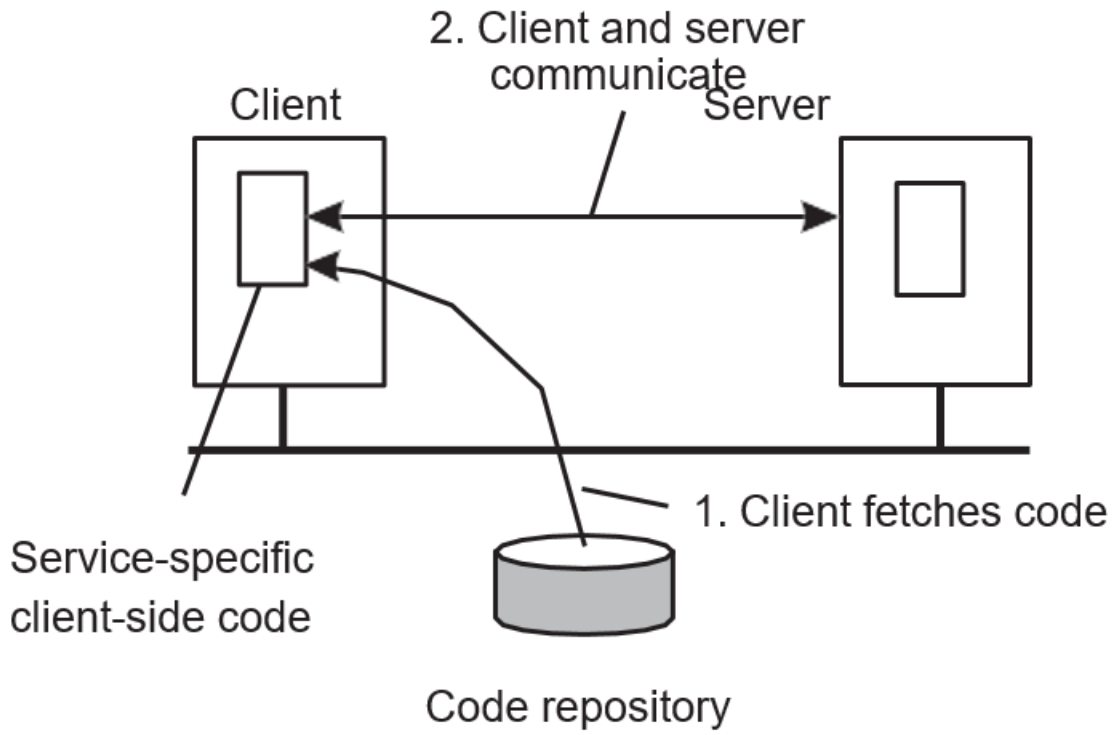

Figure. The principle of dynamically configuring a client to communicate with a server

Continue…

- The important advantage of this model of dynamically downloading clientside software is that clients need not have all the software preinstalled to talk to servers.
- Instead, the software can be moved in as necessary, and likewise, discarded when no longer needed.
- Another advantage is that as long as interfaces are standardized, we can change the client-server protocol and its implementation.
- Changes will not affect existing client applications relying on the server.
- There are, of course, also disadvantages. The most serious one has to do with security. Blindly trusting that the downloaded code implements only the advertised interface while accessing your unprotected hard disk and does not send the juiciest parts to heaven knows-who may not always be such a good idea. Fortunately, it is well understood how to protect the client against malicious, downloaded code.

Migration in Heterogeneous Systems

- So far, we have tacitly assumed that the migrated code can be easily executed at the target machine. This assumption is in order when dealing with homogeneous systems.
- In general, however, distributed systems are constructed on a heterogeneous collection of platforms, each having their own operating system and machine architecture.
- The problems coming from heterogeneity are in many respects the same as those of portability.
- Not surprisingly, solutions are also very similar. For example, at the end of the 1970s, a simple solution to alleviate many of the problems of porting Pascal to different machines was to generate machine independent intermediate code for an abstract virtual machine.
- That machine, of course, would need to be implemented on many platforms, but it would then allow Pascal programs to be run anywhere.

Continue…

- Although this simple idea was widely used for some years, it never really caught on as the general solution to portability problems for other languages, notably C.
- About 25 years later, code migration in heterogeneous systems is being tackled by scripting languages and highly portable languages such as Java.
- In essence, these solutions adopt the same approach as was done for porting Pascal.
- All such solutions have in common that they rely on a virtual machine that either directly interprets source code (as in the case of scripting languages), or otherwise interprets intermediate code generated by a compiler (as in Java).
- Further developments have weakened the dependency on programming languages.

Continue…

- In particular, solutions have been proposed to migrate not only processes, but to migrate entire 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 virtual machine monitors running an operating system and a suite of applications.
- With virtual machine 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 processes can remain ignorant of the migration itself: they need not be interrupted in their execution, nor should they experience any problems with used resources.
- The latter are either migrating along with a process, or the way that a process accesses a resource is left unaffected.

Thank You ☺