Server Clusters

Local Area Clusters

A server cluster is nothing else but a collection of machines connected through a network, where each machine runs one or more servers. The machines are connected through LAN in this case, often offering high bandwidth and low latency.

General Organisation

A server cluster is locally organised into three tiers:

1. This tier consists of a (logical) switch through which client requests are routed. Such a switch can vary widely. Example: transport layer switches accept incoming TCP connection requests and pass requests to one of the servers in the cluster.
2. Consists of application / compute servers which offer servers running on high-performance hardware in the case of cluster computing and servers running on relatively low-end hardware in the case of enterprise servers.
3. Consists of data processing servers, notably file and database servers.

It is frequently the case where machines are equipped with their own local storage, often integrating application and data processing in a single server leading to a two-tiered architecture. When a server cluster offers mutliple services, it may happen that different machines run different application servers. As a consequence, the switch will have to distinguish services or otherwise it cannot forward requests to the proper machines. As a consequence, several machines may become idle, while others are overloading with requests. So it may be configured to distribute requests to the idle machines, balancing the load.

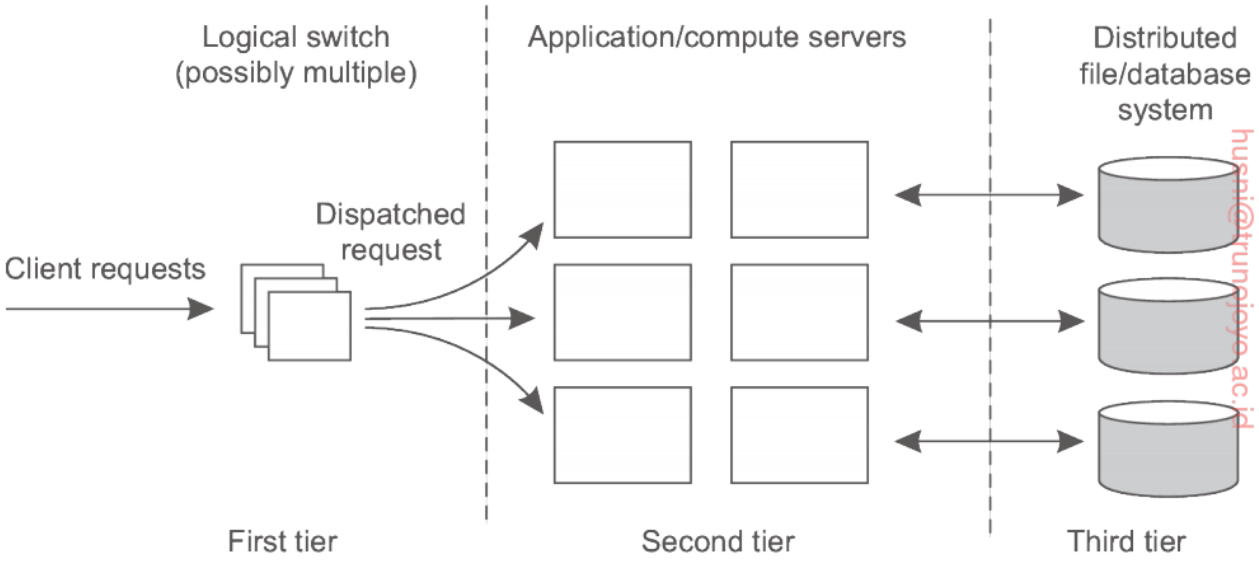

Figure 3.18: The general organization of a three-tiered server cluster.

Request Dispatching

The First tier, consisting of the switch, is known as the **front-end**. **Important Design goal**: Hide the fact that there are multiple servers. This transparency is invariably offered by the means of a single access point (switch).

The switch forms the entry point of 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 seperate dedicated machine.

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 cases of **transport layer switches**, the switch accepts incoming TCP connection requests, and hands off such connection to one of the servers.

2 ways in which switches can operate:

1. A client sets up a TCP connection such that all requests and responses pass through the switch. The switch, in turn, sets up a TCP connection with a selected server and pass client requests to that server, and also accept server responses. This approach is a form of **network address translation (NAT)**.
2. Alternatively, the switch can actually hand off the connection to the selected server such that all responses are directly communicated to the client without passing through the server. This principle working of what is commonly known as **TCP handoff**.

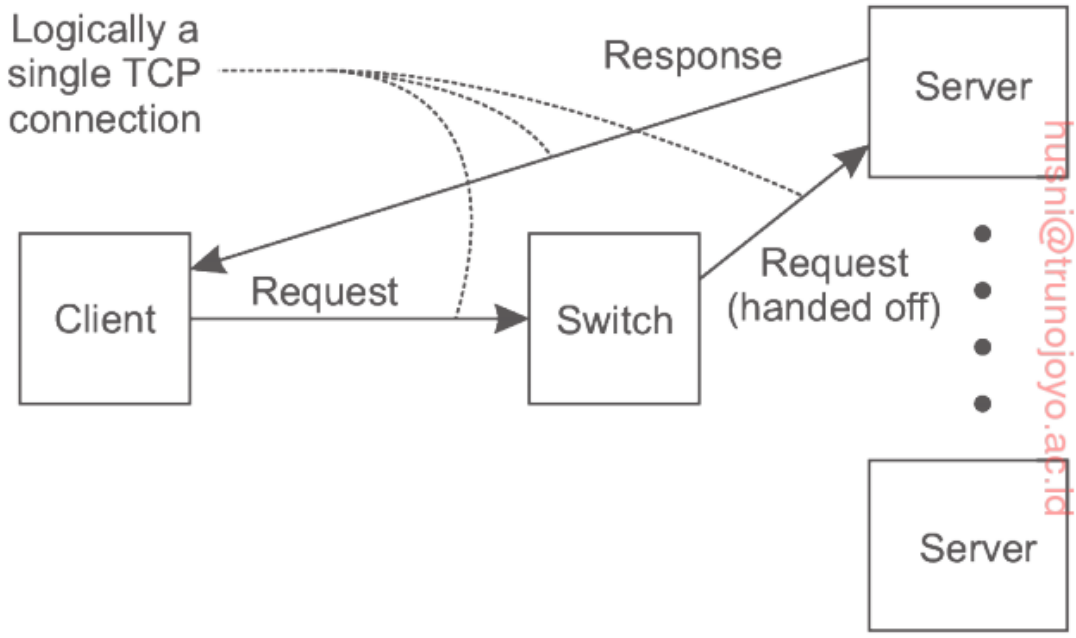

Figure 3.19: The principle of TCP handoff.

When the switch recieves a TCP connection request, it first identifies the best server for handling that request, and forwards the request packet to the server. The server, in turn, will send an acknowledgement 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, since it is expecting an answer from the switch instead of some arbitary server it has never heard of before. Clearly, it requires OS level modifications. TCP handoff is especially effective when responses are much larger than requests.

The simplest load balancing policy that the switch can follow is round robin: each time it picks the next server from its list to forward a request to.

Content-aware request distribution: switch may actually inspect the payload of the incoming request.

Wide Area Clusters

Generally not owned by a single organisation unlike local area. Deploying clusters can be cumbersome as one had to generally deal with multiple administrative organizations such as ISPs. Cloud Computing has changed this. The problems related to having deal with multiple organisations are effectively circumvented by making use of facilities of a single cloud provider. Wide area provide the benefit offering data and services close to clients.

Request Dispatching

If a client accesses a service, its request should be forwarded to a nearby server, that is, a server that will allow communication with the client to be fast. Deciding which server should handle that client's request is an issue of **redirection policy**. If we assume that a client will intially contact a request

dispatcher, then that dispatcher will to 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. A popular one is when the dispatcher is actually a **DNS name server**.

DNS

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 send 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.

This scheme is not perfect for 2 reasons:

- Rather than sending the client's IP address, what happens is that the local DNS server that is contacted by the client acts as a proxy for that client. So the DNS server's IP address is used to identify the location of the client not the client's.
- Depending on the scheme that is used for resolving a domain name, it may even be the case the address of the local DNS server is not even being used. Instead, it may happen that the DNS server is deciding on which IP address to return, may be fooled by the fact that the requester is yet another DNS server acting as an intermediate between the original client and the deciding DNS server. In those cases, the locality awareness has been completely lost.