# Clustered Network Applications

Kiril Kamenev
Helsinki University of Technology
Department of Computer Science
kamenen@cc.hut.fi

## Abstract

The goal of this document is to provide the reader an introduction to clustered applications as well as guidelines to several commonly used techniques for implementing application clusters. Clustered solutions can be used to improve system availability, performance or distribute the load of the system. Each of these cases is discussed in more detail and accompanied by a real world example.

The document focuses on practical issues related to different clustering techniques rather than gives a theoretical justification of benefits and drawbacks of clustering.

## 1 Introduction

In today's world, there is a number of practical tasks in many human activity fields that require large amount of calculations to be done in a reasonable time. These tasks are ranging from an advanced simulation of some physical process to building an internet system capable of handling extremely large amounts of simultaneous requests. In early days of computing, supercomputers were built to solve these problems. These computers were extremely expensive and affordable only by big organizations[2]. Over time as computing technologies have been advancing, the price of standard workstation class computers decreased dramatically, while supercomputers were still expensive and limited to a narrow range of tasks. This lead to the idea of aggregating the power of several average power computers to perform complex tasks.

In computer world clustering means connecting two or more computers in such a way that they logically behave as one computer. Every computer in such a system is called a node, and the set of these computers is called a cluster.

There is a number of applications for clustered systems. The main advantages one can gain by switching from one machine to multiple machines configuration are [2]:

*High availability (HA)* aims to make a system less failure prone, including software and hardware failures. Typically, it is done by eliminating a single point of failure in the system.

*Scalability* means stepwise increasing system capacity in terms of computing power, storage space, etc.

*Load balancing (LB)* is used to share the load between nodes in the cluster, so that every node gets equally loaded.

A more detailed discussion of each of the above aspects of clustering is presented in subsequent sections.

This document focuses on clustering of network applications. The term "network application" implies that the application is utilising client-server architectural model. A special type of clusters, discussed in Chapter 6.2 is high performance clusters which aim to build a high performance system that isn't necessarily used for client-server application types.

## 2 High Availability

There are many applications whose nature requires them to be up-and-running 100% of time. An example of such an application can be one phone switch that serves a certain area. Should this switch go down due to hardware or software failure, the whole area served by it will loose phone connections. The classic nuclear reactor example gives us an idea of much more disastrous subsequences of system design having a single point of failure.

Real world environment is full of reasons for a particular part of a complex system to fail, ranging from natural disasters, to intentional attacks or technical failures. That's why it is very crucial to understand the importance of system's fail over capabilities.

A clustered solution addresses these issues by adding one or more backup nodes for every critical service, thus eliminating a single point of failure. If primary server goes down, the backup takes over the processing, thus preventing the whole system from failure. This is a conceptual idea utilised by the majority of clustered systems, however, the real world applications usually introduce more sophisticated requirements for such systems. For instance a clustered database might require fail over support within a transaction which adds lot of complexity to cluster design.

An important property of a HA system is standby type. Usually, in HA system one node is in active mode and another one is in standby mode. When in active mode, the node is performing its primary functions like processing requests. When in standby mode, node doesn't carry out any operations related to its primary function, but instead it monitors the state (also called health) of the active node.

If the standby node is able to take over the processing immediately after the active one fails, the standby mode is called hot standby. If some time is needed when to initialize the backup node, the standby mode is called cold[14].

## 3 Scalability

Scaling a system means increasing its performance by adding an additional module (e.g. server node). In fact, it is often more cost effective to build a clustered system consisting of nodes with average power instead of acquiring one extremely powerful machine. A good example of a scalable system is a global database. A resource that database management systems lack most is CPU power. If a database server gets too much requests and there's not enough power to serve them, it will result in long delays or even refused connections.

Adding an additional database server will increase the amount of served requests, though productivity growth is not linear.

There are two different types of scaling - vertical and horizontal scaling. Horizontal scaling means adding a new node to the cluster. Vertical scaling means adding running several instances of the network application on the same machine, e.g. using a different port number[2]. There are several reasons to use vertical scaling instead of horizontal:

- The solution can prevent the system from SW failures, while the hardware failures still remains an issue. On the other hand software crashes occur more often that hardware, and for many applications it is enough to tackle only software faults.
- Even though both instances of an application run on the same physical machine, performance of the system can still be increased. Some OS prevent a single process from consuming more than a certain amount of computing power. Creating several processes allows utilising CPU resources more efficiently.
- The solution is cheap to implement, since no investments in new hardware are needed. Only an additional application license is needed.

## 4 Load Balancing

Load balancing is similar to scaling a system. A typical candidate for load balancing is a web- or database server. By distributing all incoming requests to different servers, the amount of requests served by one server is decreased, thus improving response times and increasing the amount of simultaneous connections.

Load balancing is closely related to high availability, since sharing the load between several nodes also eliminates a single point of failure. The difference is that in load balancing scenario all servers are active and loaded at the same time, while pure HA scenario implies that one or several servers are mostly in standby mode.

## 5 Clustered Applications Types

Clustered applications can be divided into two large groups:

- Cluster-aware applications
- Cluster-unaware applications

Cluster-aware applications are designed specifically for use in clustered environment. They know about existence of other nodes and are able to communicate with them. Clustered database is one example of such application. Instances of clustered database run in different nodes and have to notify other instances if they need to lock or modify some data.

Cluster-unaware applications don't know if they are running in a cluster or on a single node. Existence of a cluster is completely transparent for such applications, and some additional software is usually needed to set up a cluster. A web server is a typical cluster-unaware application. All servers in the cluster have the same content, and the client doesn't care from which server he/she gets the requested resource. A web server serving a dynamic application, where every user has own context must be cluster-aware, since user's session must be shared and synchronized among the nodes.

Usually a clustered solution doesn't cover only one of the above problems, but instead solves each of them to a certain extent[1]. Depending on the application nature and context, emphasis is put to the aspects that are most important for a particular application. The following sections discuss each of the problems listed above in more detail and provide a brief introduction to commonly used solutions.

### 5.1 DNS-based clustering

A clustered web server is perhaps the simplest example of an application that is a good candidate to be clustered. If one decides to set up a web server, and eventually the server becomes popular, the problem of server performance will arise sooner or later. For the end user this problem will result in extremely long response time or/and unavailability of the requested resources. The reason for that can be insufficient bandwidth, insufficient computing power or both of them. Naturally, these signs indicate that it is time to scale up the system[5, 12].

The simplest way to do that is to set up an additional web server with its own IP address, but bind this address to the same domain name as for the first server[5]. The DNS server should also be configured so that it uses the addresses defined for a certain name in a round-robin manner. This will result in one part of the users connecting to one server, while the rest will be connecting to another one. Thus, the average load of one server will be decreased.

Here are the main advantages of this solution:

- Easy to implement, since DNS round-robin is a standard function of most of DNS servers on all platforms
- It is transparent from both application and client point of view.
- Server is chosen only once in the beginning of communications.
- Clustering can be applied to all IP traffic (TCP, UDP)

Unfortunately this approach has also some major disadvantages:

- The traffic cannot be differentiated by port number, i.e. all traffic will be destined to the same server, no matter what destination port is used

- DNS server is not aware of the availability of servers in the cluster. Should one of the servers fail, users who will resolve the host name to IP address of the failed server will not be able to access the needed services
- DNS server is not aware of the load of servers, which may result in overloading or/and failing of one of the nodes
- Many network application clients cache IP addresses, which may result in a hot spot, or trying to contact a server that is currently unavailable, i.e. high availability cannot be achieved with this approach

The drawbacks of DNS-based clustering can be partly solved by adding some intelligence to the DNS server. Typically a company owns an authoritative name server that is resolving the names belonging to the top level domain owned by the company. This is where the DNS intelligence should be applied. In contrast with the traditional IP address rotation approach discussed above, the following enhancements can be offered[5]:

- IP address of unavailable server is not sent to the client
- IP address of overloaded server is not sent to the client
- users can be redirected to the most proximate sever
- A virtual IP address, not a real IP address of a back-end server can be issued to the user

However even these improvements do not solve all the problems existing in DNS-based clustering. For instance, client caching of IP addresses still remains a problem.

The intelligent DNS server is the first step towards the systems with dedicated load balancer, which is discussed in more detail in next section.

## 5.2 Using a dedicated load balancer

A more sophisticated approach to clustering is to introduce a separate network device that will serve as load balancer and distribute the network traffic between available servers. The level of intelligence that load balancer possesses can vary.

The simplest case is distributing the requests in round-robin fashion. More complex design may include some logic for determining the load of every server in the cluster and make a redirection decision based on that. Load balancer can also take care of keeping a track of server's status (available or not) and taking that into account before making a decision.

There are several basic techniques for determining the load of a server. For web server it would be enough to issue a simple HTTP request to each server and measure the response time. For more complex applications, a special agent can be placed onto every server. The agent should be capable of collecting all the information characterizing the server load, and returning that information to the load balancer[12].

The challenge of this approach is to make the load balancer transparent for the client. Several existing solutions for this problem are described in the following sections.

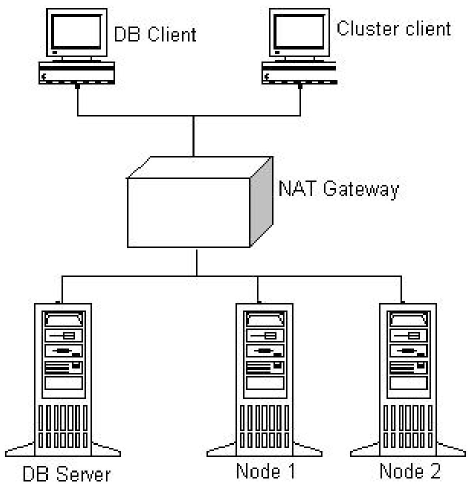

Figure 1: NAT forwarding might result in non-cluster traffic going through the gateway

### 5.2.1 Network Address Translation (NAT)

NAT is a technique that provides a way to "hide" a network behind one IP address. Originally it was developed to solve the problem of lack of IP addresses, but it also can be utilised for clustering network applications. When NAT is in use, the network lying behind the NAT gateway appears to the outer network as one IP address. All inbound and outbound traffic goes through the gateway, where the address translation is carried out. During the address translation, the gateway modifies the packet to substitute the source and destination IP addresses so that the packet reaches its final destination. This solution is completely transparent for the end user[1], since he/she is totally unaware of what happens behind the NAT gateway, and even of the existence of such a gateway at all[12].

However NAT approach has also some serious drawbacks. The main problem is that all traffic going in both directions must go through the gateway. The gateway is responsible for opening, modifying and forwarding of every packet it receives. This creates a significant processing overhead. The fact that for majority of server applications outbound traffic is much bigger than incoming makes this even a bigger problem.

Another aspect of this problem is that NAT gateway completely separates networks from each other, thus enforcing NAT translation for all other (not belonging to the cluster) hosts residing in the network. For instance, the clustered application server might use database for storing data. Setting up the database on the same network as the application server would make all other applications using the database communicate with it through the gateway. If the database is located in a different network, all cluster-database communication would use go trough NAT[12]. Figure 1 illustrates this issue.

[1]This is the case only with applications that don't use source IP address for other than transport purposes. E.g. IPSec also uses source IP to calculate the security checksum to ensure that the packet was not altered on its way to destination. As a consequence, IPSec doesn't work over NAT as it modifies the IP packet header.

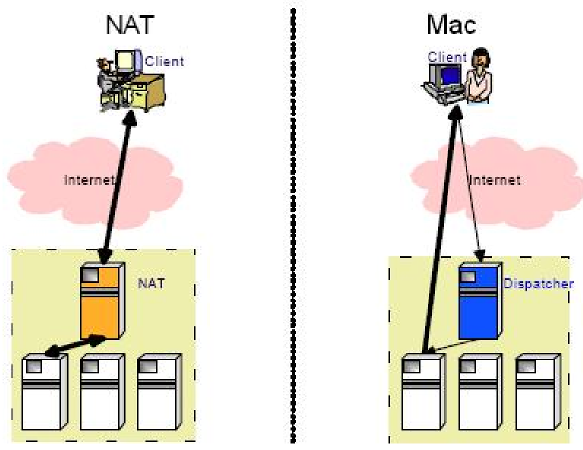

Figure 2: Differences between NAT and MAC routing

### 5.2.2 Media Access Control(MAC) based forwarding

MAC based forwarding is an alternative to NAT. This section provides a brief description of MAC forwarding, and discusses its differences from NAT.

MAC address is the physical address of the network interface card, which is used to deliver the packets over physical medium. IP addresses are used to route the packets between hosts which are in different physical networks, while the scope of MAC is limited to one physical network segment. MAC layer is located below IP layer in TCP/IP stack, and data presented in MAC headers is not relevant for higher level protocols.

MAC forwarding is based on substituting destination MAC address with the address of the host where the packet must be redirected. The server then receives the redirected packet and treats it as it was sent directly to this server. Since no modifications were made to the IP part of the packet, the application receives an exact copy of the IP packet sent by the client, including the header fields[12].

This means that the cluster node chosen to process the request believes that it is communicating directly with the client. It sends its reply directly to the client, bypassing the load balancer. The advantage of this approach is that it doesn't bring the overhead of return packet processing, thus saving the processing power of load balancer. As a subsequence it increases system performance. Figure 2 illustrates the differences between routes taken by packets with NAT and MAC.

The limitation of MAC forwarding is that the load balancer and the cluster nodes have to reside on the same physical network segment, i.e. it cannot be used in a highly distributed environment.

Forwarding methods described above can be used unless there is no need to maintain some kind of persistence state across several requests. This approach suits e.g. for a web server serving static content. Unfortunately is doesn't work with the "shopping cart" type of applications that are very popular in the internet. The problem is that client specific information is available only on one node in a cluster, if the subsequent request happens to be forwarded to a different cluster, the application will not be able to access that information. To some extent this problem can be solved by adding a "sticky" connection functionality to the load balancer. Sticky connection means that all requests arriving from the same client will be forwarded to the same node.

However, even this solution doesn't solve all the possible problems. IP address is the easiest way to distinguish one client from another, but nowadays lots of users access the internet via various proxies, which results in many users having the same IP address. Under some circumstances it can result in a hot spot situation. Other proxies use a pool of IP addresses and assign a random IP address for every connection, which makes the sticky connection feature completely useless.

HTTP protocol has a special identifier called session ID that is used to distinguish between different connections even if they are coming from the same IP address. Some products utilise this to overcome the problems mentioned above. However, it is difficult to imagine a product that will support all the variety of existing stateful protocols.

## 6 Cluster-aware applications

In addition to the problems discussed in the previous chapter, a major disadvantage of cluster-unaware applications is impossibility to provide fail-over facilities for context sensitive applications. If user's session exists only on that node, other nodes don't know about it. If the serving node fails, load balancer will redirect the request to one of the servers that are up, but all the session data will be lost.

Cluster-aware applications solve this problem by adding a notification mechanism that keeps all the nodes in the cluster synchronized. In this case user's data is present on all nodes and each of them is able to take over the processing.

The following two sections present two examples of cluster-aware applications.

### 6.1 A clustered database

Database management system is a special case of cluster-aware clustered application. It is even more complex application type that the one described in the previous section. In case of a shopping cart application every user has its own session, and session is always modified only on one node. The only thing the server program has to take care about is replication of the session to other nodes. In case of a database, all cluster nodes work on the same data, and maintaining strict consistency of the underling data is the main challenge[10]. Additional feature that is specific only to database clusters is to provide transactional fail over, i.e. if one node fails during the transaction, it can be completed by another node.

EMIC provides a solution for creating database clusters based on MySQL database engine. Every node in the cluster has its own copy of the database and a replication engine that ensuring that data on all nodes is consistent. One of the main tasks of the replication engine is to preserve the global total order (GTO) of the arriving messages. Preserving GTO means that client messages will be applied to the data in the same order on every node. This guarantees that all the nodes will have the same data at all times[7].

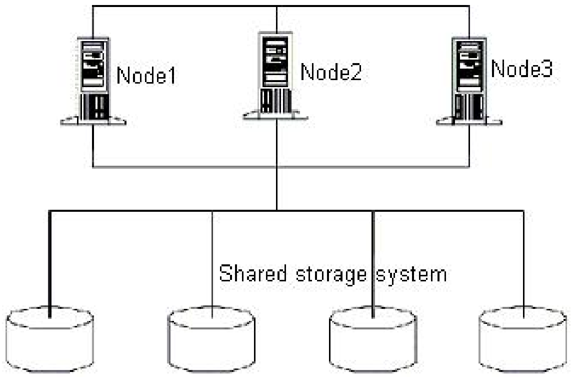

Figure 3: Oracle Real Application Cluster Architecture

Strict consistency of the data allows executing read queries locally without contacting other nodes, which greatly increases the performance of applications that are intensively reading data.

If the queries changes the data in the database, it is replicated to all nodes in the cluster using replication engine. The changes are applied to the data after safe delivery and GTO property are ensured.

Oracle provides a similar large-scale solution called Real Application Cluster. Oracle cluster is implemented by following the "share everything" paradigm. A cluster consists of processor nodes, cluster interconnect, and a disk subsystem. Disk subsystem is a data storage storage shared between processor nodes. Scaling the storage subsystem can increase the system capacity, and increasing the amount of processor nodes can reduce response times[11].

In such an environment every node is acting relatively independently, and it is very important to detect if the failure occurred on some node and isolate that node. For instance, due to failure of interconnect, a cluster might become just a set of servers groups isolated from each other. The cluster system must be able to decide which group will remain functioning as cluster, and which groups should temporary stop operations. This technique is called failure isolation.

## 6.2 High performance clusters

High performance cluster is an alternative to one extremely powerful machine needed to perform a large amount of calculations. All HPCs fall into one of two large groups:

- Tightly coupled clusters
- Loosely coupled clusters

Loosely coupled clusters suite very well for large tasks that can be easily divided into small independent tasks. Such clusters usually have one or more master servers, which distribute work units between the computational nodes and collect the work results. Nodes in such clusters can be distributed over large areas, e.g. internet. If a specific task is hardly dividable into small independent tasks, then tightly coupled cluster is a more suitable solution for the problem. In a tightly coupled cluster every node depends on one or more other nodes, and data flow includes several nodes before a task is completed[6].

Loosely- and tightly-coupled clusters concepts are closely related to the concept of commodity and constellation clusters.

Commodity clusters are build by using standard commercial off the shelf (COTS) products. For instance cluster nodes can be widely available workstation class computers with one of the popular operating systems. Standard LAN (or WAN) technologies can be used for communications within cluster. Utilising standard technologies allows creating clusters of extremely large size. Well-known examples of a large scale computing clusters are DNA decoding and search for extraterrestrial life programmes. In each of these programmes every internet user around the globe could make his computer a cluster node by installing a client application. Digital video rendering is another example of a task that can be carried out by a loosely-coupled cluster. Every frame can be rendered independently, which makes the division of work very easy[2].

A special care of networking issues has to be taken. If a cluster contains a large number of nodes, it is likely that at some point many nodes will start submitting their computation results to the master node, which may result in a performance bottleneck. High capacity networking solution can be used to tackle this problem[6].

The difference between commodity and constellation clusters is that in constellation clusters number of processors in one cluster node exceeds the number of nodes in the cluster. Another difference is that the interconnecting network doesn't necessarily have to be standard; it can be a proprietary or custom designed network.

Constellation clusters suite well for the same tasks as tightly-coupled clusters. Usage of non-standard networks makes them limited to a small area. On the other hand, strict control over the configuration of every node makes it possible to create a highly predictable system with minimal deviations from the expected performance. High dependency between nodes in a tightly coupled cluster increases the affect of network latency on the overall system performance. This is the reason for employing non-standard technologies which can provide desired level of control[2].

# 7 Conclusion

The amount of computational tasks that can be carried out only by a very powerful computer system is growing very rapidly. Many of these tasks were introduced by internet technologies and services gaining more and more popularity every day. Clustering is a cost-effective alternative to supercomputers in this field.

Three main benefits that one can gain by clustering a client-server application are high availability, scalability and load balancing. Non client-server applications can use clustering for building a high performance computational systems called high performance clusters (HPC).

Clustering of a network application can be implemented in a variety of ways, depending on the system requirements, available investments, etc. Some solutions like DNS based load balancing are very easy to implement, but have some limitations that might be crucial for big systems. There are also complete solutions for building a highly available, scalable application clusters suitable for big organizations.

## References

[1] Jon Björklund, *Adaptive Load Balancing in a Cluster Computer*, October 2002, UNIVERSITY OF HELSINKI, Department of Computer Science

[2] Mark Baker, *Cluster Computing White Paper*, December 2000, University of Portsmouth, UK

[3] Connie U. Smith, Lloyd G. Williams, *Performance and Scalability of Distributed Software Architectures*, September 2000

[4] Mark Baker, Rajkumar Buyya, *Cluster Computing: A High-Performance Contender*, July 1999, University of Portsmouth, Monash University

[5] Cisco Systems, *The Global Server Load Balancing Primer*, 2004, http://www.cisco.com

[6] Extreme Networks, *Network Considerations for Clustered Servers*, 2003, http://www.extremenetworks.com/

[7] EMIC Networks, *High Availability Cluster with Dynamic Load*, November 2003, http://www.emicnetworks.com/

[8] Alcatel Internetworking, *Server Load Balancing*, 2002, http://www.alcatel.com

[9] John McArthur and Dennis Byron, International Data Systems, *Building Successful Enterprise Application Suites on an Enterprise Storage Foundation*, 1999, http://www.idc.com

[10] Oracle, *Database Architecture:Federated vs. Clustered*, March 2002 http://www.oracle.com

[11] Oracle, *Building Highly Available Database Servers Using Oracle Real Application Clusters* May 2002 http://www.oracle.com

[12] IBM Corporation, *Network Dispatcher White Paper*, June 2000, http://www.ibm.com

[13] M. Patino-Martinez, R. Jimenez-Peres, B.Kemme, G. Alonso, *Scalable Replication in Database Clusters*

[14] David W. Coit, *Cold-standby redundancy optimization for nonrepairable systems* Department of Industrial Engineering, Rutgers University, 2000