Enhancing Computational Performance through Load Balancing in Parallel and Distributed Systems

Nguyen Van Tuan[1] and Pham Thi Hoa[2]

[1]Department of Information Systems, Hanoi University of Science and Technology, Hanoi, Vietnam
[2]Department of Computer Science, University of Da Nang, Da Nang, Vietnam

ABSTRACT

In computing, load balancing improves the distribution of workloads across multiple computing resources, such as computers, a computer cluster, network links, central processing units, or disk drives.Load balancing aims to optimize resource use, maximize throughput, minimize response time, and avoid overload of any single resource.Parallel computing is a type of computation in which many calculations or the execution of processes are carried out simultaneously. Large problems can often be divided into smaller ones, which can then be solved at the same time.

Keywords: Distributed Computing,Distributed Programming,parallel computing ,load balancing,client-server load balancing .

I. INTRODUCTION

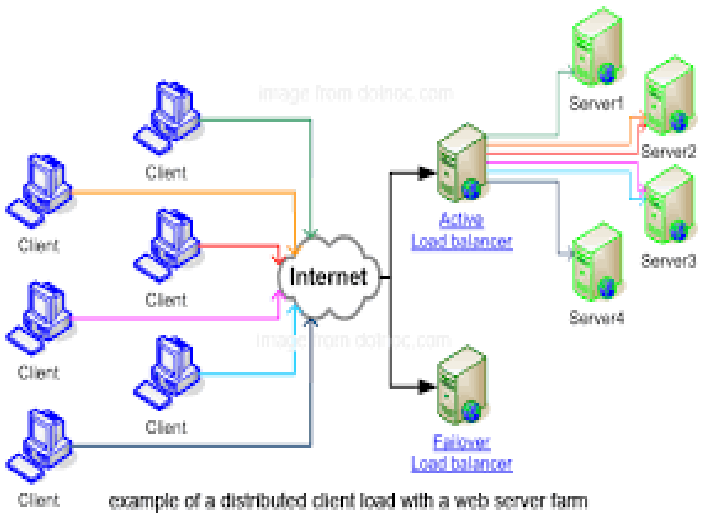
Using multiple components with load balancing instead of a single component may increase reliability and availability through redundancy. Load balancing usually involves dedicated software or hardware, such as a multilayer switch or a Domain Name System server process.
Load balancing differs from channel bonding in that load balancing divides traffic between network interfaces on a network socket (OSI model layer 4) basis, while channel bonding implies a division of traffic between physical interfaces at a lower level, either per packet (OSI model Layer 3) or on a data link (OSI modelLayer 2) basis with a protocol like shortest path bridging.

Parallel computing

There are several different forms of parallel computing: bit-level, instruction-level, data, and task parallelism. Parallelism has long been employed in high-performance computing, but it's gaining broader interest due to the physical constraints preventing frequency scaling.[2] As power consumption (and consequently heat generation) by computers has become a concern in recent years,[3] parallel computing has become the dominant paradigm in computer architecture, mainly in the form of multi-core processors.[4]

Parallel computing is closely related to concurrent computing—they are frequently used together, and often conflated, though the two are distinct: it is possible to have parallelism without concurrency (such as bit-level parallelism), and concurrency without parallelism (such as multitasking by time-sharing on a single-core CPU).[5][6] In parallel computing, a computational task is typically broken down into several, often many, very similar subtasks that can be processed independently and whose results are combined afterwards, upon completion. In contrast, in concurrent computing, the various processes often do not address related tasks; when they do, as is typical in distributed computing, the separate tasks may have a varied nature and often require some inter-process communicationduring execution.

Parallel computers can be roughly classified according to the level at which the hardware supports parallelism, with multi-core and multi-processor computers having multiple processing elements within a single machine, while clusters, MPPs, and grids use multiple computers to work on the same task. Specialized parallel computer architectures are sometimes used alongside traditional processors, for accelerating specific tasks.
In some cases parallelism is transparent to the programmer, such as in bit-level or instruction-level parallelism, but explicitly parallel algorithms, particularly those that use concurrency, are more difficult to write than sequential ones,[7] because concurrency introduces several new classes of potential software bugs, of which race conditions are the most common. Communication and synchronization between the different subtasks are typically some of the greatest obstacles to getting good parallel program performance.



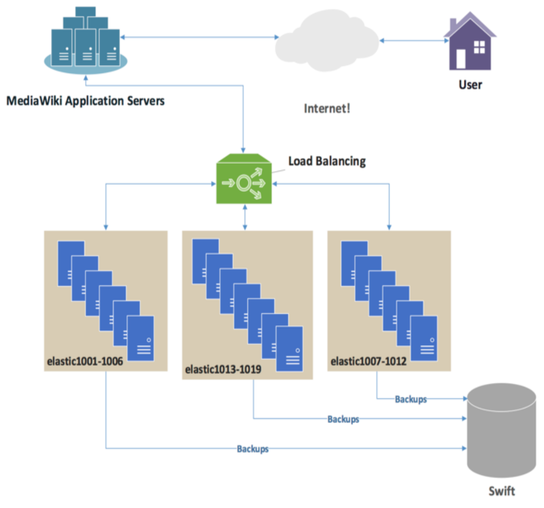


Figure2: Userrequests to the Wikimedia Elasticsearch server cluster are routed via load balancing

Tables:
Internet-based services:**One of the most commonly used applications of load balancing is to provide a single Internet service from multiple *servers*, sometimes known as a *server farm*. Commonly load-balanced systems include popular *web sites*, large *Internet Relay Chat* networks, high-bandwidth *File Transfer Protocol* sites, *Network News Transfer Protocol* (NNTP) servers, *Domain Name System* (DNS) servers, and databases.**

Round-robin DNS: ***An alternate method of load balancing, which does not require a dedicated software or hardware node, is called round robin DNS. In this technique, multiple IP addresses are associated with a single domain name; clients are given IP in round robin fashion. IP is assigned to clients for a time quantum.***

DNS delegation : ***Another more effective technique for load-balancing using DNS is to delegate www.example.org as a sub-domain whose zone is served by each of the same servers that are serving the web site. This technique works particularly well where individual servers are spread geographically on the Internet. For example:***

```
one.example.org A 192.0.2.1
two.example.org A 203.0.113.2
www.example.org NS one.example.org
www.example.org NS two.example.org
```

However, the zone file for www.example.org on each server is different such that each server resolves its own IP Address as the A-record.[2] On server *one* the zone file for www.example.org reports:

```
@ in a 192.0.2.1
```

On server *two* the same zone file contains:
This way, when a server is down, its DNS will not respond and the web service does not receive any traffic. If the line to one server is congested, the unreliability of DNS ensures less HTTP traffic reaches that server. Furthermore, the quickest DNS response to the resolver is nearly always the one from the network's closest server, ensuring geo-sensitive load-balancing. A short TTL on the A-record helps to ensure traffic is quickly diverted when a server goes down. Consideration must be given the possibility that this technique may cause individual clients to switch between individual servers in mid-session.

Client-side random load balancing
Another approach to load balancing is to deliver a list of server IPs to the client, and then to have client randomly select the IP from the list on each connection. This essentially relies on all clients generating similar loads, and the Law of Large Numbers to achieve a reasonably flat load distribution across servers. It has been claimed that client-side random load balancing tends to provide better load distribution than round-robin DNS; this has been attributed to caching issues with round-robin DNS, that in case of large DNS caching servers, tend to skew the distribution for round-robin DNS, while client-side random selection remains unaffected regardless of DNS caching.

With this approach, the method of delivery of list of IPs to the client can vary, and may be implemented as a DNS list (delivered to all the clients without any round-robin), or via hardcoding it to the list. If a "smart client" is used, detecting that randomly selected server is down and connecting randomly again, it also provides fault tolerance.

Server-side load balancers
For Internet services, server-side load balancer is usually a software program that is listening on the port where external clients connect to access services. The load balancer forwards requests to one of the "backend" servers, which usually replies to the load balancer. This allows the load balancer to reply to the client without the client ever knowing about the internal separation of functions. It also prevents clients from contacting back-end servers directly, which may have security benefits by hiding the structure of the internal network and preventing attacks on the kernel's network stack or unrelated services running on other ports.

Some load balancers provide a mechanism for doing something special in the event that all backend servers are unavailable. This might include forwarding to a backup load balancer, or displaying a message regarding the outage.

It is also important that the load balancer itself does not become a single point of failure. Usually load balancers are implemented in high-availability pairs which may also replicate session persistence data if required by the specific application.

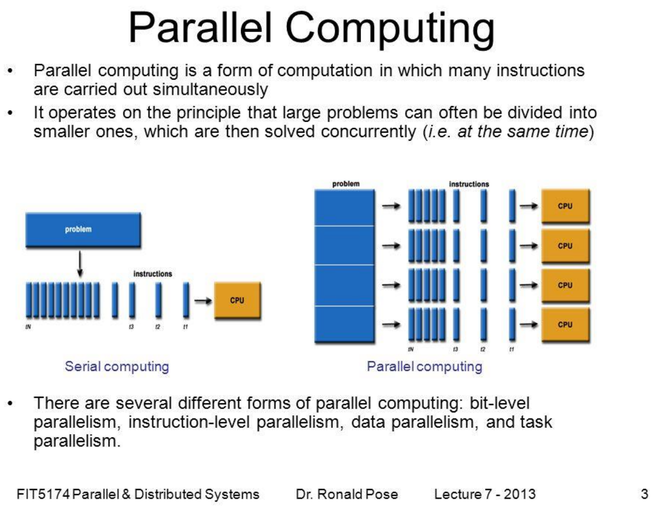
Scheduling algorithms
Numerous scheduling algorithms, also called load-balancing methods, are used by load balancers to determine which back-end server to send a request to.[6] Simple algorithms include random choice or round robin. More sophisticated load balancers may take additional factors into account, such as a server's reported load, least response times, up/down status (determined by a monitoring poll of some kind), number of active connections, geographic location, capabilities, or how much traffic it has recently been assigned.

Load balancer features
The fundamental feature of a load balancer is to be able to distribute incoming requests over a number of backend servers in the cluster according to a scheduling algorithm. Most of the following features are vendor specific:

Asymmetric load: A ratio can be manually assigned to cause some backend servers to get a greater share of the workload than others. This is sometimes used as a crude way to account for some servers having more capacity than others and may not always work as desired.

Priority activation: When the number of available servers drops below a certain number, or load gets too high, standby servers can be brought online.

Figures and tables



Parallel Computing

- Parallel computing is a form of computation in which many instructions are carried out simultaneously
- It operates on the principle that large problems can often be divided into smaller ones, which are then solved concurrently (*i.e. at the same time*)

problem
instructions
CPU
tN
t3
t2
t1
Serial computing
problem
instructions
CPU
CPU
CPU
CPU
Parallel computing

- There are several different forms of parallel computing: bit-level parallelism, instruction-level parallelism, data parallelism, and task parallelism.

FIT5174 Parallel & Distributed Systems Dr. Ronald Pose Lecture 7 - 2013 3

Table 1:Difference between

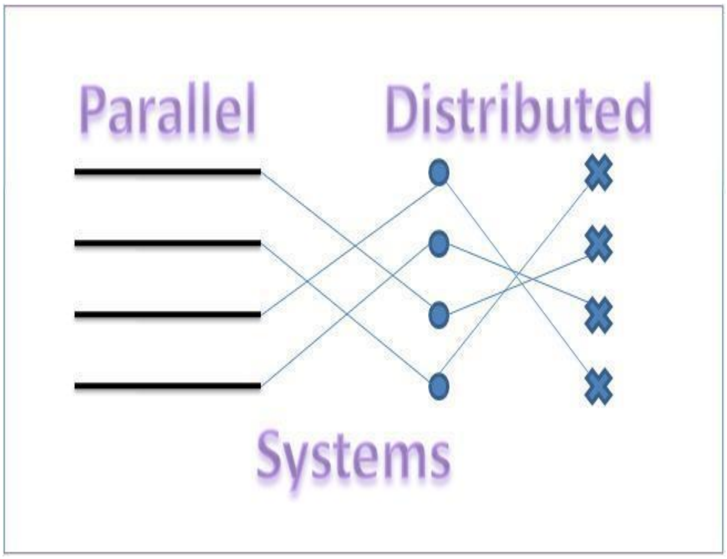


Figure 4:Showing the difference between Paralell and Distributed systems

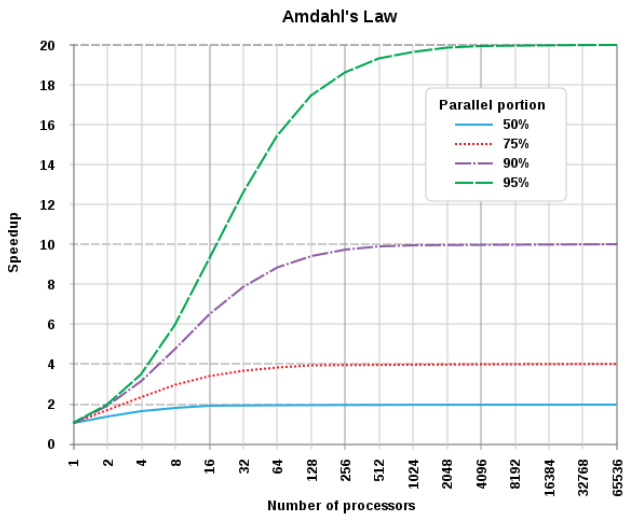


Figure 5:Graph Respresentation

II. CONCLUSION

Using multiple components with load balancing instead of a single component may increase reliability and availability through redundancy. Load balancing usually involves dedicated software or hardware, such as a multilayer switch or a Domain Name System server process. Priority activation: When the number of available servers drops below a certain number, or load gets too high, standby servers can be brought online

III. REFERENCES

1. Andrews, Gregory R. (2000), Foundations of Multithreaded, Parallel, and Distributed Programming, Addison–Wesley,ISBN 0-201-35752-6.
2. Gottlieb, Allan; Almasi, George S. (1989). Highly parallel computing. Redwood City, Calif.: Benjamin/Cummings. ISBN 0-8053-0177-1.