ISSN: 2393-8528

IJICSE
Open Access

Contents lists available at www.ijicse.in

International Journal of Innovative Computer Science & Engineering

Volume 2 Issue 3; July-August-2015; Page No.05-11

A proposed work on randomized hydrodynamic algorithm for load balancing

Rashi saxena

Computer Science & Engineering Department, JVWU, Jaipur, India

ARTICLE INFO

Received: 18 May 2015
Accepted 28 June 2015

Corresponding Author:

Rashi saxena

Computer Science & Engineering Department, JVWU, Jaipur, India

Email: rashi.saxena@yahoo.co.in

ABSTRACT

Load Balancing is a method of distributing workload across many servers in a network. Typical datacenter implementations rely on large, powerful (and expensive) computing hardware and network infrastructure, which are subject to the usual risks associated with any physical device, including hardware failure, power and/or network interruptions, and resource limitations in times of high demand. Load balancing in the cloud differs from classical thinking on load-balancing architecture and implementation by using commodity servers to perform the load balancing. This provides for new opportunities and economies-of-scale, as well as presenting its own unique set of challenges. Load balancing is used to make sure that none of your existing resources are idle while others are being utilized. To balance load distribution, you can migrate the load from the source nodes (which have surplus workload) to the comparatively lightly loaded destination nodes. When load balancing is applied during runtime, it is called dynamic load balancing — this can be realized both in a direct or iterative manner according to the execution node selection: In the iterative methods, the final destination nodes are determined through several iteration steps. In the direct methods, the final destination node is selected in one step. Another kind of Load Balancing method can be used i.e. the Randomized Hydrodynamic Load Balancing method, a hybrid method that takes advantage of both direct and iterative methods.

Keywords: *About five key words in alphabetical order, separated by comma*

1. INTRODUCTION

1.1 Cloud Computing

Cloud computing has become very popular in recent years as it offers greater flexibility and availability of computing resources at very low cost. The major concern for agencies and organizations considering moving the applications to public cloud computing environments is the emergence of cloud computing facilities to have far-reaching effects on the systems and networks of the organizations. Many of the features that make cloud computing attractive, however, can also be at odds with traditional security models and controls.

As with any emerging information technology area, cloud computing should be approached carefully with due consideration to the sensitivity of data. Planning helps to ensure that the computing environment is as secure as possible and is in compliance with all relevant organizational policies and that data privacy is maintained. It also helps to ensure that the agency derives full benefit from information technology spending. The security objectives of an organization are a key factor for decisions about outsourcing information technology services and, in particular, for decisions about transitioning organizational data, applications, and other resources to a public cloud computing environment.

To maximize effectiveness and minimize costs, security and privacy must be considered from the initial planning stage at the start of the systems development life cycle. Attempting to address security after implementation and deployment is not only much more difficult and expensive, but also more risky. Cloud providers are generally not aware of a specific organization's security and privacy needs. Adjustments to the cloud computing environment may be warranted

to meet an organization's requirements. Organizations should require that any selected public cloud computing solution is configured, deployed, and managed to meet their security and privacy requirements.

Cloud computing technologies can be implemented in a wide variety of architectures, under different service and deployment models, and can coexist with other technologies and software design approaches. The security and privacy challenges cloud computing presents, however, are formidable, especially for public clouds whose infrastructure and computational resources are owned by an outside party that sells those services to the general public.

1.2 Cloud Service Models

Cloud service delivery is divided into three models. The three service models are:

1.2.1 Cloud Software as a service (SaaS)

The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through a thin client interface such as a web browser. The consumer does not manage the underlying cloud infrastructure.

1.2.2 Cloud Platform as a Service (Peas)

The capability provided to the consumer is to deploy onto the cloud infrastructure consumer created or acquired applications created using programming languages and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure, but has control over the deployed applications and possibly application hosting environment configurations.

1.2.3 Cloud Infrastructure as a Service (Iasi)

The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components.

Features and parts of Iasi include:

- Utility computing service and billing model.
- Automation of administrative tasks.
- Dynamic scaling.
- Desktop virtualization.
- Policy-based services.

1.3Virtualization

It is a very useful concept in context of cloud systems. Virtualization means"something which isn't real", but gives all the facilities of a real. It is the software implementation of a computer which will execute different programs like a real machine. Virtualization is related to cloud, because using virtualization an end user can use different services of a cloud. The remote datacenter will provide different services in a full or partial virtualized manner.

Two types of virtualization are found in case of clouds as given in:

- Full virtualization
- Para virtualization

1.3.1 Full Virtualization

In case of full virtualization a complete installation of one machine is done on another machine. It will result in a virtual machine which will have all the software that is present in the actual server. Here the remote datacenter delivers the services in a fully virtualized manner. Full virtualization has been successful for several purposes as pointed out in:

a. Sharing a computer system among multiple users

b. Isolating users from each other and from the control program

c. Emulating hardware on another machine

1.3.2 Para virtualization

In Para virtualization, the hardware allows multiple operating systems to run on single machine by efficient use of system resources such as memory and processor. E.g. VMware software. Here all the services are not fully available, rather the services are provided partially. Para virtualization has the following advantages as given in :

- **Disaster recovery:** In the event of a system failure, guest instances are moved to another hardware until the machine is repaired or replaced.
- **Migration:** As the hardware can be replaced easily, hence migrating or moving the different parts of a new machine is faster and easier.
- **Capacity management**: In a virtualized environment, it is easier and faster to add more hard drive capacity and processing power. As the system parts or hardware can be moved or replaced or repaired easily, capacity management is simple and easier.

1.4 Cloud Components

A Cloud system consists of 3 major components such as clients, datacenter, and distributed servers. Each element has a definite purpose and plays a specific role.

1.4.1 Clients

End users interact with the clients to manage information related to the cloud. Clients generally fall into three categories:

- **Mobile:** Windows Mobile Smartphone, Smartphone, like a Blackberry, or an iPhone.
- **Thin:** They don't do any computation work. They only display the information. Servers do all the works for them. Thin clients don't have any internal memory.
- **Thick:** These use different browsers like IE or Mozilla Firefox or Google Chrome to connect to the

Internet cloud. Now-a-days thin clients are more popular as compared to other clients because of their low price, security, low consumption of power, less noise, easily replaceable and repairable etc.

1.4.2 Datacenter

Datacenter is nothing but a collection of servers hosting different applications. A end user connects to the datacenter to subscribe different applications. A datacenter may exist at a large distance from the clients. Now-a-days a concept called virtualization is used to install software that allows multiple instances of virtual server applications.

1.4.3 Distributed Servers

Distributed servers are the parts of a cloud which are present throughout the Internet hosting different applications. But while using the application from the cloud, the user will feel that he is using this application from its own machine.

1.5 Load balancing in cloud computing

Load Balancing is a method to distribute workload across one or more servers, network interfaces, hard drives, or other computing resources. Typical datacenter implementations rely on large, powerful (and expensive) computing hardware and network infrastructure, which are subject to the usual risks associated with any physical device, including hardware failure, power and/or network interruptions, and resource limitations in times of high demand. Load balancing in the cloud differs from classical thinking on load-balancing architecture and implementation by using commodity servers to perform the load balancing. This provides for new opportunities and economies-of-scale, as well as presenting its own unique set of challenges. Load balancing is used to make sure that none of your existing resources are idle while others are being utilized. To balance load distribution, you can migrate the load from the *source nodes* (which have surplus workload) to the comparatively lightly loaded *destination nodes*. When you apply load balancing during runtime, it is called *dynamic load balancing* — this can be realized both in a direct or iterative manner according to the execution node selection:

• In the iterative methods, the final destination node is determined through several iteration steps.

• In the direct methods, the final destination node is selected in one step.

Another kind of Load Balancing method can be used i.e. the Randomized Hydrodynamic Load Balancing method, a hybrid method that takes advantage of both direct and iterative methods.

1.5.1 Goals of Load balancing

The goals of load balancing are:

1. To improve the performance substantially.
2. To have a backup plan in case the system fails even partially.
3. To maintain the system stability.
4. To accommodate future modification in the system.

1.5.2 Types of Load balancing algorithms

Depending on who initiated the process, load balancing algorithms can be of three categories as given in :

- **Sender Initiated**: If the load balancing algorithm is initialized by the sender.
- **Receiver Initiated**: If the load balancing algorithm is initiated by the receiver.
- **Symmetric**: It is the combination of both sender initiated and receiver initiated. Depending on the current state of the system, load balancing algorithms can be divided into 2 categories as given in :

Static: It does not depend on the current state of the system. Prior knowledge of the system is needed.

Dynamic: Decisions on load balancing are based on current state of the system. No prior knowledge is needed. So it is better than static approach. Here we will discuss on various dynamic load balancing algorithms for the clouds of different sizes.

2. Proposed work

Cloud computing is designed to provide on demand resources or services over the Internet, usually at the scale and with the reliability level of a data center. Map Reduce is a programming model designed for processing large volumes of data in parallel by dividing the work into a set of independent tasks. It is a style of parallel programming that is supported by some capacity-on- demand-style clouds such as Google's Big Table, Hadoop, and Sector.

In this paper, a load-balancing algorithm that follows the approach of the Randomized Hydrodynamic Load Balancing technique (more on that in the following sections) is used. Virtualization is used to reduce the actual number of physical servers and cost; more importantly, virtualization is used to achieve efficient CPU utilization of a physical machine.

To get the most from this paper, you should have a general idea of cloud computing concepts, the Randomized Hydrodynamic Load Balancing technique, and the Hardtop Map Reduce programming model. A basic understanding of parallel programming will help and any programming knowledge on Java™ or other object-oriented languages will be a good support tool.

For this experiment, the MapReduce algorithm was implemented on a system using:

• Hadoop 0.20.1.

• Eclipse IDE 3.0 or above (or Rational Application Developer 7.1).

• Ubuntu 8.2 or above.

Before diving into the Map Reduce algorithm, we'll set up the basics of the cloud architecture, load balancing, Map Reduce, and parallel programming — enough at least for this article.

2.1 Cloud architecture: The basics

Figure 1 shows a detailed picture of the complete system, platforms, software, and how they are used to achieve the goal set in this paper.

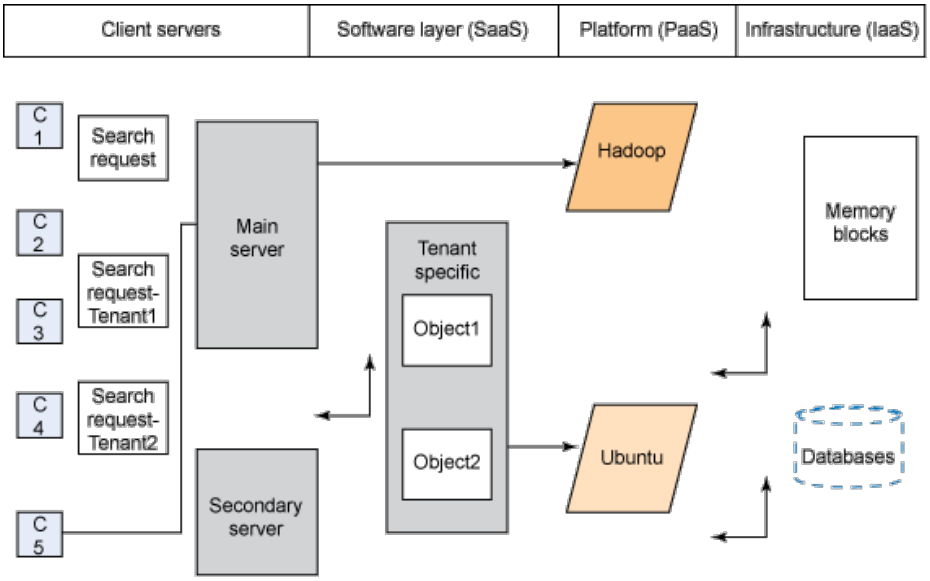

Figure 1: The cloud architecture

You can see Ubuntu 9.04 and 8.2 is used for the operating systems; Hadoop 0.20.1, Eclipse 3.3.1, and Sun Java 6 for the platforms; the Java language for programming; and HTML, JSP, and XML as the scripting languages.

This cloud architecture has both mastered and slave nodes. In this implementation, a main server is maintained that gets client requests and handles them depending on the type of request. The master node is present in main server and the slave nodes in secondary server.

Search requests are forwarded to the Name Node of Hadoop, present in main server as you can see in Figure 2. The Hadoop Name Node then takes care of the searching and indexing operation by initiating a large number of Map and Reduce processes. Once the Map Reduce operation for a particular search key is completed, the Name Node returns the output value to the server and in turn to the client.

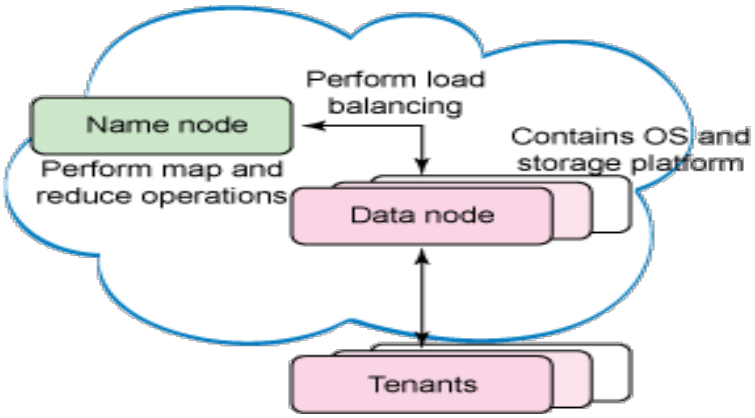

Figure 2: Map and Reduce functions do searching and indexing

If the request is for a particular software, authentication steps are done based on the customer tenant ID, payment dues, eligibility to use a particular software, and the lease period for the software. The server then serves this request and allows the user to consume a selected software combination.

The multi tenancy feature of SaaS is provided here, in which a single instance of the software serves a number of tenants. For every tenant specific request there will be a thin line of isolation generated based on the tenant id. These requests are served by a single instance. When a tenant specific client request wants to search files or consume any other multi-tenant

software the offerings use Hadoop on the provisioned operating system instance (PaaS). Also, in order to store his data -- perhaps a database or files-- in the cloud, the client will have to take some memory space from the data center (IaaS). All these moves are transparent to the end user.

2.2 Randomized Hydrodynamic Load Balancing: The basics

Load balancing is used to make sure that none of your existing resources are idle while others are being utilized. To balance load distribution, you can migrate the load from the *source nodes* (which have surplus workload) to the comparatively lightly loaded *destination nodes*.

When you apply load balancing during runtime, it is called *dynamic load balancing*— this can be realized both in a direct or iterative manner according to the execution node selection:

• In the iterative methods, the final destination node is determined through several iteration steps.

• In the direct methods, the final destination node is selected in one step.

For this paper, the Randomized Hydrodynamic Load Balancing method is used, a hybrid method that takes advantage of both direct and iterative methods.

2.3 Map Reduce: The basics

Map Reduce programs are designed to compute large volumes of data in a parallel fashion. This requires dividing the workload across a large number of machines. Hadoop provides a systematic way to implement this programming paradigm

The computation takes a set of input key/value pairs and produces a set of output key/value pairs. The computation involves two basic operations: Map and Reduce.

The Map operation, written by the user, takes an input pair and produces a set of intermediate key/ value pairs. The Map Reduce library groups together all intermediate values associated with the same intermediate Key #1 and passes them to the Reduce function.

The Reduce function, also written by the user, accepts an intermediate Key #1 and a set of values for that key. It merges together these values to form a possibly smaller set of values. Typically just an output value of 0 or 1 is produced per Reduce invocation. The intermediate values are supplied to the user's Reduce function via an integrator (an object that allows a programmer to traverse through all the elements of a collection regardless of its specific implementation). This allows you to handle lists of values that are too large to fit in memory.

Take the example of Word Count Problem. This will count the number of occurrences of each word in a large collection of documents. The Mapper and Reducer function will look like Listing 1.

Listing 1. Map and Reduce in a WordCount problem

```
mapper (filename, file-contents):
for each word in file-contents:
emit (word, 1)
reducer (word, values):
sum = 0
for each value in values:
sum = sum + value emit (word, sum)
```

The Map function emits each word plus an associated count of occurrences. The Reduce function sums together all counts emitted for a particular word. This basic functionality, when built over clusters, can easily turn into a high-speed parallel processing system.

Performing computation on large volumes of data has been done before, usually in a distributed setting. What makes Hadoop unique is its simplified programming model — which allows the user to quickly write and test distributed systems — and it's efficient, automatic distribution of data and work across machines and in turn utilizing the underlying parallelism of the CPU cores.

Let's try to make things a little clearer. As discussed earlier, in a Hadoop cluster you have the following nodes:

• The Name Node (the cloud master).

• The Data Nodes (or the slaves).

Nodes in the cluster have preloaded local input files. When the Map Reduce process is started, the Name Node uses the Job Tracker process to assign tasks which have to be carried out by Data Nodes, through the Task Tracker processes. Several Map processes will run in each Data Node and the intermediate results will be given to the combiner process which generates, for instance, the count of words in file of one machine as(in case of a Word Count problem). Values are shuffled and sent to reduce processes which generate the final output for the problem of interest.

2.4 How load balancing is used

Load balancing is helpful in spreading the load equally across the free nodes when a node is loaded above its threshold level. Though load balancing is not so significant in execution of a Map Reduce algorithm, it becomes essential when handling large files for processing and when hardware resources use is critical. As a highlight, it enhances hardware utilization in resource- critical situations with a slight improvement in performance.

A module was implemented to balance the disk space usage on a Hadoop Distributed File System cluster when some data nodes became full or when new empty nodes joined the cluster. The balancer (Class Balancer tool) was started with a threshold value; this parameter is a fraction between 0 and 100 percent with a default value of 10 percent. This sets the target for whether the cluster is balanced; the smaller the threshold value, the more balanced a cluster will be. Also, the longer it takes

to run the balancer. (Note: A threshold value can be so small that you cannot balance the state of the cluster because applications may be writing and deleting files concurrently.)

A cluster is considered balanced if for each data node, the ratio of used space at the node to the total capacity of node (known as the *utilization of the node*) differs from the the ratio of used space at the cluster to the total capacity of the cluster (*utilization of the cluster*) by no more than the threshold value.

The module moves blocks from the data nodes that are being utilized a lot to the poorly used ones in an iterative fashion; in each iteration a node moves or receives no more than the threshold fraction of its capacity and each iteration runs no more than 20 minutes.

In this implementation, nodes are classified as *highly-utilized*, average-*utilized*, and *under-utilized*. Depending upon the utilization rating of each node, load was transferred between nodes and the cluster was balanced. The module worked like this:

• First, it acquires neighborhood details:

1. When the load increases in a Data Node to the threshold level, it sends a request to the Name Node.
2. The Name Node had information about the load levels of the specific Data Node's nearest Neighbours.
3. Loads are compared by the Name Node and then the details about the free-est neighbour nodes are sent to the specific Data Node.

• Next, the Data Nodes go to work:

1. Each DataNode compares its own load amount with the sum of the load amount of its nearest neighbours.
2. If a Data Node's load level is greater than the sum of its neighbours, then load-destination nodes (direct neighbours AND other nodes) will be chosen at random.
3. Load requests are then sent to the destination nodes

• Last, the request is received:

1. Buffers are maintained at every node to received load requests.
2. A message passing interface (MPI) manages this buffer.
3. A main thread will listen to the buffered queue and will service the requests it receives.
4. The nodes enter the load-balancing-execution phase.

2.5 Evaluating the performance

Different sets of input files were provided, each of different size, and executed the Map Reduce tasks in both single- and two-node clusters. The corresponding times of execution were measured and we came to the conclusion that running Map Reduce in clusters is by far the more efficient for a large volume of input file.

The graphs in Figure 3 illustrate our performance results from running on various nodes.

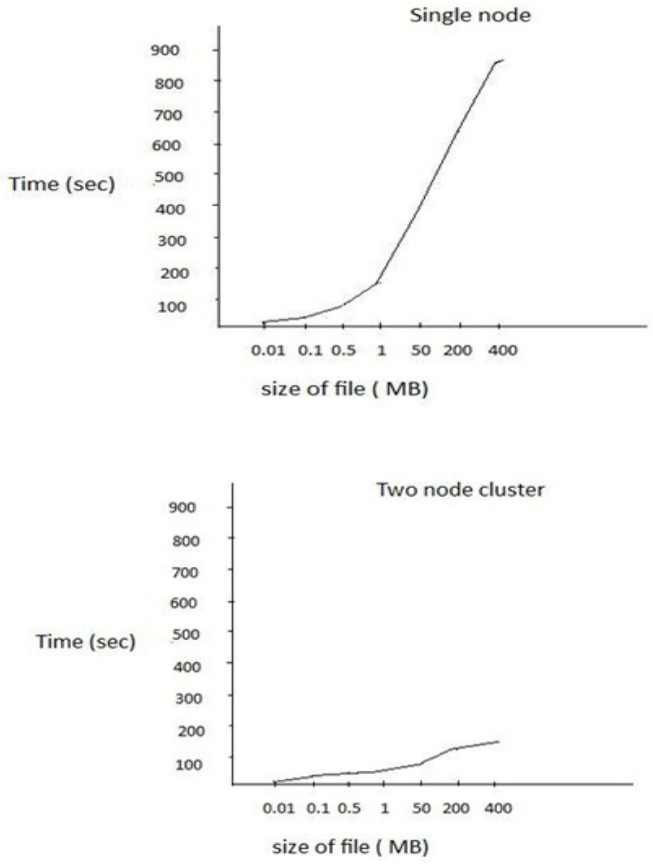

Figure 3: Map Reduce load balancing works more efficiently in clusters

3. Conclusion

Our experiment with Hadoop Map Reduce and load balancing lead to two inescapable **conclusions:**

• In a cloud environment, the Map Reduce structure increases the efficiency of throughput for large data sets. In contrast, you wouldn't necessarily see such an increase in throughput in a non-cloud system.

• When the data set is small, Map Reduce and load balancing do not affect an appreciable increase in throughput in a cloud system.

Therefore, consider a combination of Map Reduce-style parallel processing and load balancing when planning to process a large amount of data on your cloud system.

References

1. Chengzhong XU, Burkhard MONIEN, Reinhard LÜNING, Francis C. M. LAU "Nearest neighbor algorithms for load balancing in parallel Computers"
2. Chang-Zhang XU, Francis C.M. LAU "Iterative dynamic load balancing in multicomputers", Journal of Operational Research Society, Vol. 45 No. 7, July 1994, pp. 786–796

3. Serkan AKSOY, Hüseyin A. SERİM "Technical report on normal mode method - Normal mod yötemi teknik analiz raporu", unpublished.

4. G. CYBENKO "Load balancing for ditributed memory multiprocessors", Journal of Parallel and Distributed Computing, Vol. 7, 1989, pp. 279–307

5. F.C.H. LIN, R.M. KELLER "The gradient model load balancing method", IEEE Transactions on Software Engineering, Vol. 13, 1987, pp. 32–38

6. W. SHU, L.V. KALE "A dynamic scheduling strategy for the chare kernel systems", In Proceedings of Supercomputing, 1989, pp. 389–398

7. Marta BELTRÀN, Jose BOSQUE "Information policies for load balancing on heterogeneous systems", IEEE International Symposium on Cluster Computing and the Grid, 2005, pp. 970 – 979

8. Mark H. WILLEBEEK-LEMAIR, Anthony P. REEVES "Strategies for dynamic load balancing on highly parallel computers", IEEE Transactions on parallel and distributed systems, Vol. 4 No. 9, 1993, pp. 979 – 993

9. Malcolm YOKE, Hean LOW "Dynamic load-balancing for BSP time warp", 35th Annual Simulation Symposium, 2002

10. Deyu QI, Weiwei LIN "TGrid: A next grid environment", First International Multi-Symposiums on Computer and Computational Sciences, 2006

11. Wei JIE, Wentong CAI, Stephen J. Turner "Dynamic load-balancing using prediction in a parallel object-oriented system", IEEE, 2001

12. Wei JIE, Wentong CAI, Stephen J. Turner "Dynamic load-balancing in a data parallel object-oriented system", IEEE, 2001

13. Giuseppe Di FATTA, Michael R. BERTHOLD "Dynamic load balancing for the distributed mining of molecular structures", IEEE Transactions on Parallel and Distributed Systems, Vol. 17 No. 8, August 2006

14. Jacques M. BAHI, Sylvain CONTASSOT, Raphael COUTURIER "Dynamic load balancing and efficient load estimators for asynchronous iterative algorithms", IEEE Transactions on Parallel and Distributed Systems, Vol. 16 No. 4, April 2006

15. Chi-Chung HUI, Samuel T. CHANSON "A hydrodynamic approach to heterogeneous dynamic load balancing in a network of computers", IEEE International Conference on Parallel Processing, 1996

16. Chi-Chung HUI, Samuel T. CHANSON "Hydrodynamic load balancing", IEEE Transactions on Parallel and Distributed Systems, Vol. 10 No. 11, November 2006.