Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 10, October 2014, pg.442 – 450

RESEARCH ARTICLE

Optimizing the Performance of a Class of Distributed Systems over the Internet

P.SRINIVAS[1]
C.S.E & JNTUH
Srinivas.pulloori@gmail.com

K.Kishore Babu[2]
C.S.E & JNTUH
kishore.sa@gmail.com

B.Satish Kumar[3]
C.S.E & JNTUH
satish.nit@gmail.com

[1]M. Tech IV semester, Dept of CSE, Sree Chaitanya College of Engineering, Karimnagar
[2]Assistant Prof, Dept of CSE, Sree Chaitanya College of Engineering, Karimnagar
[3]Associate Prof, Dept of CSE, Sree Chaitanya College of Engineering, Karimnagar

Abstract— In this model we propose Algorithm to optimize the performance of the distributed systems over the internet. Their consist of large number of clients and servers. The servers consist of large number of clients for this the load is increases on the server to process the data. Here the goal of our proposed system is to assign the clients to the servers and reduce the load on server and minimize the cost of processing for that we define a heuristic algorithm. Optimizing the overall performance of such a system then can be formulated as a client-server assignment problem whose aim is to assign the clients to the servers in such a way to satisfy some prespecified requirements on the communication cost and load balancing.

1. Introduction

Distributed computing is a field of computer science that studies distributed systems. A distributed system is a software system in which components located on networked computers communicate and coordinate their actions by passing messages. The components interact with each other in order to achieve a common goal. There are many alternatives for the message passing mechanism, including RPC-like connectors and message queues. Three significant characteristics of distributed systems are: concurrency of components, lack of a global clock, and independent failure of components. An important goal and challenge of distributed systems is location transparency. Examples of distributed systems vary from SOA-based systems to massively multiplayer online games to peer-to-peer applications.

A computer program that runs in a distributed system is called a distributed program, and distributed programming is the process of writing such programs. Distributed computing also refers to the use of distributed systems to solve computational problems. In distributed computing, a problem is divided into many tasks, each of which is solved by one or more computers, which communicate with each other by message passing. The word *distributed* in terms such as "distributed system", "distributed programming", and "distributed algorithm" originally referred to computer networks where individual computers were physically distributed within some geographical area. The terms are nowadays used in a much wider sense, even referring to autonomous processes that run on the same physical computer and interact with each other by message passing. While there is no single definition of a distributed system, the following defining properties are commonly used.

- There are several autonomous computational entities, each of which has its own local memory.
- The entities communicate with each other by message passing.

In this article, the computational entities are called *computers* or *nodes*.

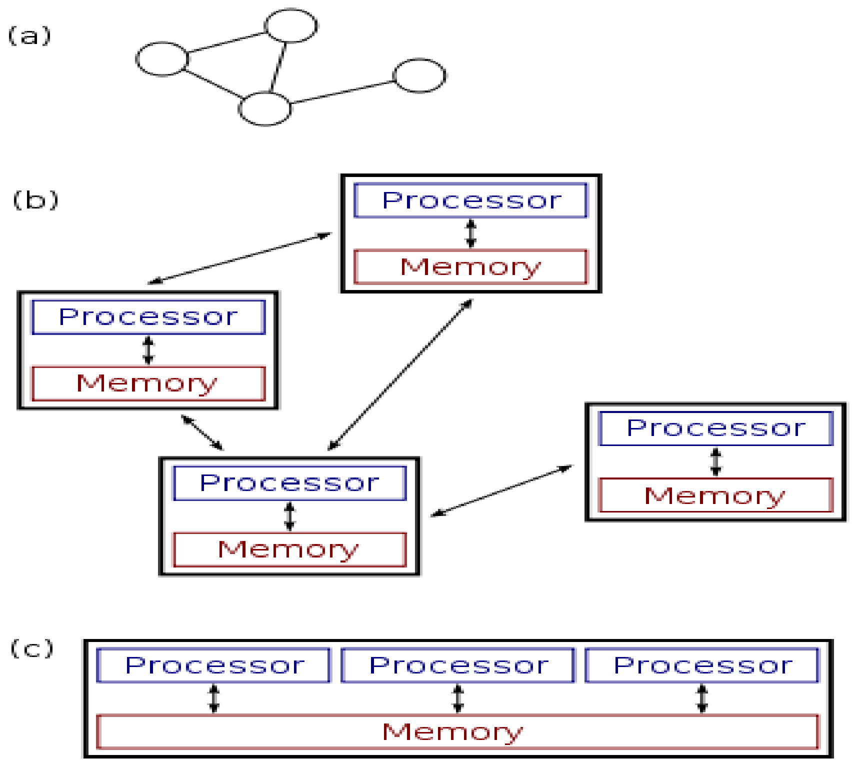
A distributed system may have a common goal, such as solving a large computational problem. Alternatively, each computer may have its own user with individual needs, and the purpose of the distributed system is to coordinate the use of shared resources or provide communication services to the users.

Other typical properties of distributed systems include the following:

- The system has to tolerate failures in individual computers.
- The structure of the system (network topology, network latency, number of computers) is not known in advance, the system may consist of different kinds of computers and network links, and the system may change during the execution of a distributed program.
- Each computer has only a limited, incomplete view of the system. Each computer may know only one part of the input.

Distributed systems are groups of networked computers, which have the same goal for their work. The terms "concurrent computing", "parallel computing", and "distributed computing" have a lot of overlap, and no clear distinction exists between them. The same system may be characterized both as "parallel" and "distributed"; the processors in a typical distributed system run concurrently in parallel. Parallel computing may be seen as a particular tightly coupled form of distributed computing, and distributed computing may be seen as a loosely coupled form of parallel computing. Nevertheless, it is possible to roughly classify concurrent systems as "parallel" or "distributed" using the following criteria.

- In parallel computing, all processors may have access to a shared memory to exchange information between processors.
- In distributed computing, each processor has its own private memory (distributed memory). Information is exchanged by passing messages between the processors.



The figure on the right illustrates the difference between distributed and parallel systems.

Figure (a) is a schematic view of a typical distributed system; as usual, the system is represented as a network topology in which each node is a computer and each line connecting the nodes is a communication link.

Figure (b) shows the same distributed system in more detail: each computer has its own local memory, and information can be exchanged only by passing messages from one node to another by using the available communication links

Figure (c) shows a parallel system in which each processor has a direct access to a shared memory.

The situation is further complicated by the traditional uses of the terms parallel and distributed *algorithm* that do not quite match the above definitions of parallel and distributed *systems*; see the section Theoretical foundations below for more detailed discussion. Nevertheless, as a rule of thumb, high-performance parallel computation in a shared-memory multiprocessor uses parallel algorithms while the coordination of a large-scale distributed system uses distributed algorithms.

2. Literature Review

2.1 EXISTING SYSTEM

A typical distributed system consists of a mix of servers and clients. The servers are more computational and resource powerful than the clients. A classical example of such systems is e-mail. When a client A sends an e-mail to another client B, A does not send the e-mail directly to B. Instead, A sends its message to its e-mail server which has been previously assigned to handle all the e-mails to and from A. This server relays A’s e-mail to another server which has been previously assigned to handle e-mails for B. B then reads A’s e-mail by downloading the e-mail from its server. Importantly, the e-mail servers communicate with each other on behalf of their clients. The main advantage of this architecture is specialization, in the sense that the powerful dedicated e-mail servers release their clients from the responsibility associated with many tasks including processing and storing e-mails, and thus making e-mail applications more scalable. A more interesting scenario is the Instant Messaging System (IMS). An IMS allows real time text-based communication between two or more participants over the Internet. Each IMS client is associated with an IMS server which handles all the instant messages for its clients. Similar to e-mail servers, IMS servers relay instant messages to each other on behalf on their clients. In an IMS that uses the XMPP (Jabber) protocol such as Google Talk, clients can be assigned to servers independent of their organizations.

2.2 DISADVANTAGESOF EXISTING SYSTEM

We use multiple servers; we also need to balance the communication load among the servers for the following reasons:

- ❖ If one server is overloaded, we need to add another server to distribute the load, which is economically inefficient and usually increases the overall communication load
- ❖ As a heavily loaded server typically exhibits a low performance, we would like to avoid the situation.
- ❖ To minimize the amount of total communication load, assigning all clients to one server is optimal. However, it is impossible due to overloading and completely loses the load balance.

Simple load balancing does not usually take account of reducing the overall communication load.

2.3 PROPOSED SYSTEM

In the proposed system the primary contribution is a heuristic algorithm via relaxed convex optimization that takes a given communication pattern among the clients as an input, and produces an approximately optimal client-server assignment for a prespecified tradeoff between load balance and communication cost. We must strike a balance between reducing the overall communication load and increasing the load fairness among the servers, i.e., the load balance.

2.4 ADVANTAGES OF PROPOSED SYSTEM

The advantages of the proposed system have follows:

- ✓ The two groups have different number of servers, a server within a group with fewer servers will likely to have a higher load than a server in the group with more servers. This reduces the load balance.
- ✓ We describe a number of emerging applications that have the potential to benefit from the client-server assignment problem.

3. IMPLEMENTATION

3.1.1 Client-Server Assignment

The client-server assignment for distributed virtual environment (DVE) systems exhibits a similar set of issues: balancing the workload and reducing the communication between the servers. The DVE systems allow multiple users working on different client computers to interact in a shared virtual world.

3.1.2 Communication Load

The communication load is the total load on all servers. The total communication load equals to1 plus the amount of the inter server communication (Fc), where the amount of the inter server communication can be expressed as the extra load caused by distributing the servers; if all clients are assigned to a single server, the total communication load is 1. The two clients are assigned to different servers; the amount of communication load is 1X {the size of the messages} for each server and 2X {the size of the messages} in total. Thus, to calculate the communication load, two different types of message passing need to be considered:

1. Message passing through a single server, i.e., intra server communication.
2. Message passing through two servers, i.e., inters server communication

3.1.3 Load Balance

Load balance should be a metric that represents the degree of load variations among different servers. Some popular metrics are variance, entropy, and Gini coefficient. The Gini coefficient is used often in economics to measure the inequality of income distribution in a society. In this paper, we consider the Gini coefficient as the load balance metric as it empirically captures the requirements of load balance on the servers better than other metrics. Specifically, for large M, the Gini coefficient is more sensitive to a slight change in the load balance than the entropy and variance.

3.1.4 Relaxed Convex Optimization

The main idea of relaxed convex optimization is to solve the special case with the number of servers M ¼ 2 via relaxed convex optimization. The main idea is to split the servers into two groups sequentially. Optimal client-server assignment for M servers by splitting M servers into two groups. If M is even, then it makes sense to split M servers into two equal groups with M=2 servers in each group. In the ideal case, optimizing the load balance between these two groups will result in individual servers in these two groups having identical communication loads. On the other hand, when making the number of servers in these groups is not same, two groups having total identical communication loads. However, since the two groups have different number of servers, a server within a group with fewer servers will likely to have higher loads than a server in the group with more servers. This reduces the load balance. Therefore, when M is not even, it is necessary to modify the objective at each step, depending on how splitting is done, so as to maintain the similar load at individual servers. Intuitively, the modified objective should reflect the number of servers in each group

System Configuration:-

H/W System Configuration:-

• System	: Pentium IV 2.4 GHz.
• Hard Disk	: 40 GB.
• Floppy Drive	: 1.44 Mb.
• Monitor	: 15 VGA Colour.
• Mouse	: Logitech.
• Ram	: 512 Mb.

S/W System Configuration:-

• Operating system	: - Windows XP.
• Coding Language	: JAVA/J2EE
• Data Base	: MYSQL

4. Problem Definition

An ideal distributed system is completely decentralized, and that every node is given equal responsibility and no node is more computational or resource powerful than any other. However, for many real-world applications, such a system often has a low performance due to a significant cost of coordinating the nodes in a completely distributed manner. In practice, a typical distributed system consists of a mix of servers and clients. The servers are more computational and resource powerful than the clients. A classical example of such systems is e-mail. When a client A sends an e-mail to another client B, A does not send the e-mail directly to B. Instead, A sends its message to its e-mail server which has been previously assigned to handle all the e-mails to and from A. This server relays A’s e-mail to another server which has been previously assigned to handle e-mails for B.B then reads A’s e-mail by downloading the e-mail from its server.

E-mail systems assign clients based primarily on the organizations that the clients belong to. Two employees working for the same company are likely to have their e-mail accounts assigned to the same e-mail server. Thus, the client-server assignment is trivial. A more interesting scenario is the Instant Messaging System (IMS). An IMS allows real time text-based communication between two or more participants over the Internet. Each IMS client is associated with an IMS server which handles all the instant messages for its clients. Similar to e-mail servers, IMS servers relay instant messages to each other on behalf on their clients. In an IMS that uses the XMPP protocol such as Google Talk, clients can be assigned to servers independent of their organizations. Furthermore, the client-server assignment can be made dynamic as deemed suitable, and thus making this problem much more interesting.

5. ALGORITHEM

Algorithm 1.

```
1) We start with picking up one arbitrary xi and set it 0.
2) Afterwards, we solve (28) (29) by convex optimization.
3) However, in most cases, other elements of x will still
remain non-binary. Therefore, we choose xc whose
value is closest to 0 or 1, then set it 0 or 1 whichever
xc is closer to.
4) Repeat 2) - 3) until no more non-binary element exists
in x.
```

Thus far, we have described the basic idea of our relaxed convex optimization approach for the two-server scenario. We can achieve the nearly optimal client-server assignment for M servers by splitting M servers into two groups and recursively splitting within each group as shown in Fig. 4. How to split M servers is the central question. If M is even, then it makes sense to split M servers into two equal groups with M/2 servers in each group. In the ideal case, optimizing the load balance between these two groups will result in individual servers in these two groups having identical server loads. On the other hand, when making the number of servers in these groups is not same, optimizing the objectives in (28) or (29) will result in two groups having total identical server loads. However, since the two groups have different number of servers, a server within a group with fewer servers
will likely to have a higher load than a server in the group with more servers. This reduces the load balance. Therefore, when M is not even, it is necessary to modify the objective at each step, depending on how splitting is done, so as to maintain the similar load at individual servers. Intuitively, the modified objective should reflect the number of servers in each group.
Claim 4.1: When splitting M servers into two groups consisting of m and M−m servers in each group, the load balance metrics in (26) and (27) should be replaced by:

Due to lack of space, the justification for this claim is omitted. The full justification can be found in [11]. We would like to mention that the modified load balance metrics allocate a higher load to groups with more servers, and is intuitively plausible. Importantly, both metrics in Claim 4.1 are convex functions, therefore we can employ the Algorithm 1 embedded in the Algorithm 2 below for finding an approximate solution.
Algorithm 2.
1) Split the number of servers into two groups with m = dM
2 e and M − m = bM
2 c servers in each group.
2) Run Algorithm 1 with modified load balance metrics stated in Claim 4.1.
3) Repeat steps 1 and 2 for each of the two groups, until the number of servers in each group equal to 1.

6. Conclusions

In this paper, we present a mathematical model and an algorithmic solution to the client-server assignment problem for optimizing the performance of a class of distributed systems over the Internet. We show that in general, finding the optimal client-server assignment for some pre-specified requirements on total load and load balancing is NP-hard, and propose a heuristic via relaxed convex optimization for finding the approximate solution to the client-server assignment problem. Our simulation results indicate that the proposed algorithm almost always finds the optimal solution. Furthermore, the proposed algorithm outperforms other heuristics, including the popular Normalized Cuts algorithm.

References

1. H Nishida, T Nguyen - Parallel and Distributed Systems, IEEE …, 2013 - ieeexplore.ieee.org
2. S Balakrishna, L Ding - Dept. Comp. Sc. & Sys., Univ. of … - uwtdev1.tacoma.uw.edu
3. DNB Ta, S Zhou - Parallel and Distributed Processing …, 2006 - ieeexplore.ieee.org
4. RD Bharati, I Naidu, A Kiran, K Khune, C Vyas - ijettjournal.org
5. A Dubey, S Sharma - ijcttjournal.org
6. M Shiga, I Takigawa, H Mamitsuka - Pattern Recognition, 2011 – Elsevier
7. X Geng, X Fan, J Bian, X Li, Z Zheng - Proceedings of the 21st …, 2012 - dl.acm.org
8. MP Brinda, MK Yazhini, MN Radhika - ijaert.org
9. Y Deng, RWH Lau - Proceedings of the 17th ACM Symposium on Virtual …, 2010 - dl.acm.org
10. W Xu, Y Gong - Proceedings of the 27th annual international ACM …, 2004 - dl.acm.org

Authors Biography:

First Author: P.SRINIVAS

M.Tech in CSE from JNTUH,
Sree Chaitanya College Of Engineering,Karimnagar

Second Author : K.Kishore Babu

M.Tech in CSE from JNTUH,
Assistent Prof, Dept of CSE,
Sree Chaitanya College Of Engineering,Karimnagar

Third Author :B.Satish Kumar

M.Tech in CSE from JNTUH,
Assoc Prof, Dept of CSE,
Sree Chaitanya College Of Engineering,Karimnagar