International Journal of Innovative Research in Engineering
Volume 3, Issue 2 (March-April 2022), PP: 164-167.
www.theijire.com **ISSN No: 2582-8746**

Resource Management Using Parallel Computing

Shruti Rajesh Shastrakar[1], Shivani Satish Bhange[2], Smital Ravindra Telang[3], Divya Waman Raut[4], Diksha Rupesh Mendhe[5], Surbhi Khare[6]
[1,2,3,4,5]*Information technology/ Priyadarshini college of Engineering, RTM Nagpur University, India.*
[6] *Assistant Professor, Information technology/ Priyadarshini college of Engineering, RTM Nagpur University, India.*

How to cite this paper:
Shruti Rajesh Shastrakar[1], Shivani Satish Bhange[2], Smital Ravindra Telang[3], Divya Waman Raut[4], Diksha Rupesh Mendhe[5], "Resource Management Using Parallel Computing", IJIRE-V3I02-164-167.

Abstract: *In this paper, we purpose to efficient algorithms refers to as rate-based load balancing via virtual routing (RLBVR) and queue-based load balancing via virtual routing (QLBVR), which belongs to the above RAP and QRAP Policies. Our focus is analyze and understand the behavior of these algorithm in terms of their load balancing ability under varying load conditions (light, moderate, or high) and the minimization of the main response time of job.*

Key Word: *Algorithm, Balancing, Load*

I. INTRODUCTION

Today a growing number of companies have to process a huge amount of data in a cost-efficient manner. The vast amount of data they have to deal with everyday has made traditional database solutions prohibitively expensive. Problems like processing crawled documents or regenerating a web index are split into several independent subtask, distributed among the available nodes, and computed in parallel. In order to simplify the development of distributed applications on top of such architectures of many of these companies have also built customized data processing framework. They can be classified by terms like high throughput computing (HTC) or many task computing (MTC) depending on the amount of data and the number of tasks involved in the computation. For companies that only have to process large amount of data occasionally running their own data center is obviously not an option. Instead, cloud computing has merged as a promising approach to rent a large IT infrastructure on a short -term pay-per-usage basis.

II. PROBLEM DEFINITION

The goal of a cloud-based architecture is to provide some form of elasticity, the ability to expand and contract capacity on-demand. The implication is that at some point additional instances of that application will be needed in order for the architecture to scale and meet demand. That means there needs to be some mechanism in place to balance request between two or more instances of that application. A load balancer provides the means by which instances of applications can be provisioned and de-provisioned automatically without requiring changes to the network or its configuration. It automatically handles its distribution decisions based in the capacity available at the same time a request is made.

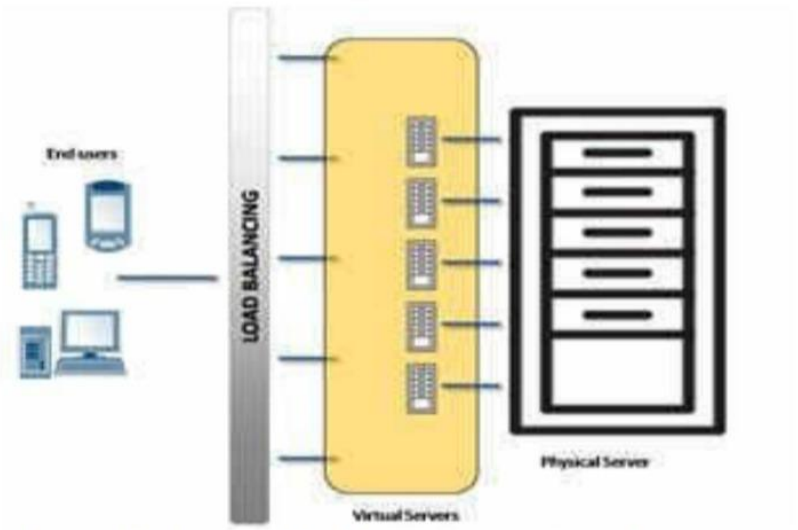


Figure 1.1: A Typical Load Balancer used in Cloud Computing

III. METHODOLOGY

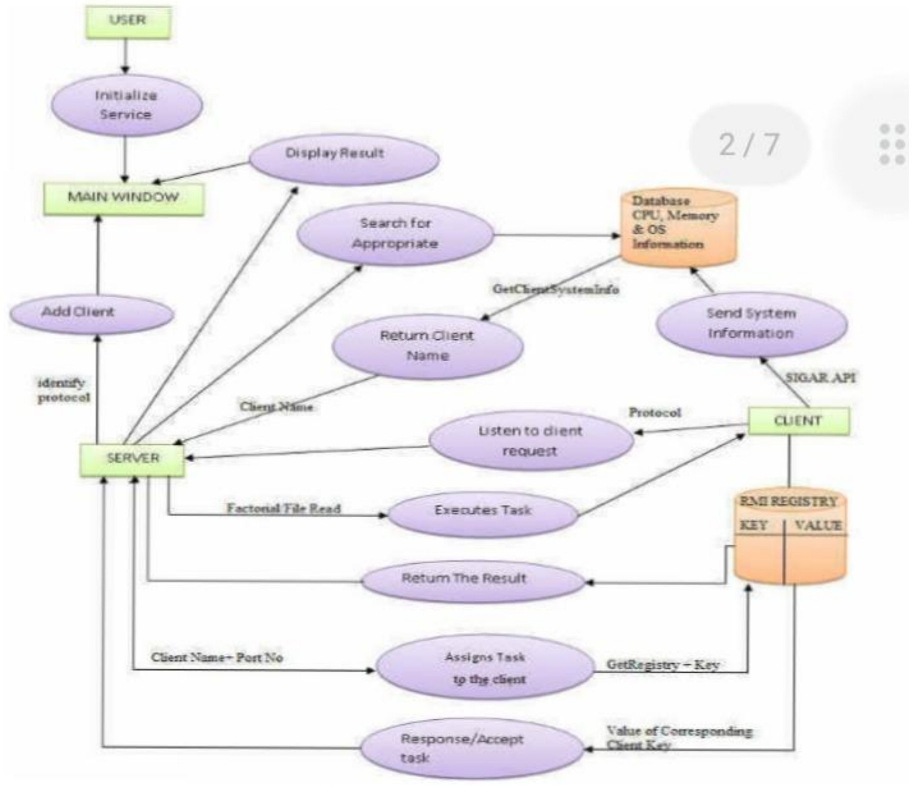


Figure: 1.2 the Data Flow Diagram of the System.

First present a general system model in the design of the algorithms. For convenience, we use “node” and “processors “interchangeably in the rest of this paper. we consider a generic parallel/distributed system shown in fig. the system consists of n heterogenous nodes, which represents host computers having different processing capabilities, interconnected by an underlying arbitrary communication network. Here, we use N to denote the set of nodes, i.e.; n=[N], and E to denote a set whose elements aur unordered pairs of distinct elements of N. each unordered pair e = (i, j) in E is called an edge. For each edge (i, j), we define two ordered pair, (i, j) and (j, i), which are called links, and we define two ordered pairs, (i, j) and (j, i), which are called links, and we denote L as the set of links. A node I is said to be a neighboring node of j if I is directly connected to j by an edge. For a node j, let denote a set of neighboring nodes of node j. We assume that jobs arrive at node i (i, n) according to an erotic process then, such as in homogenous poison process with intensity functions.

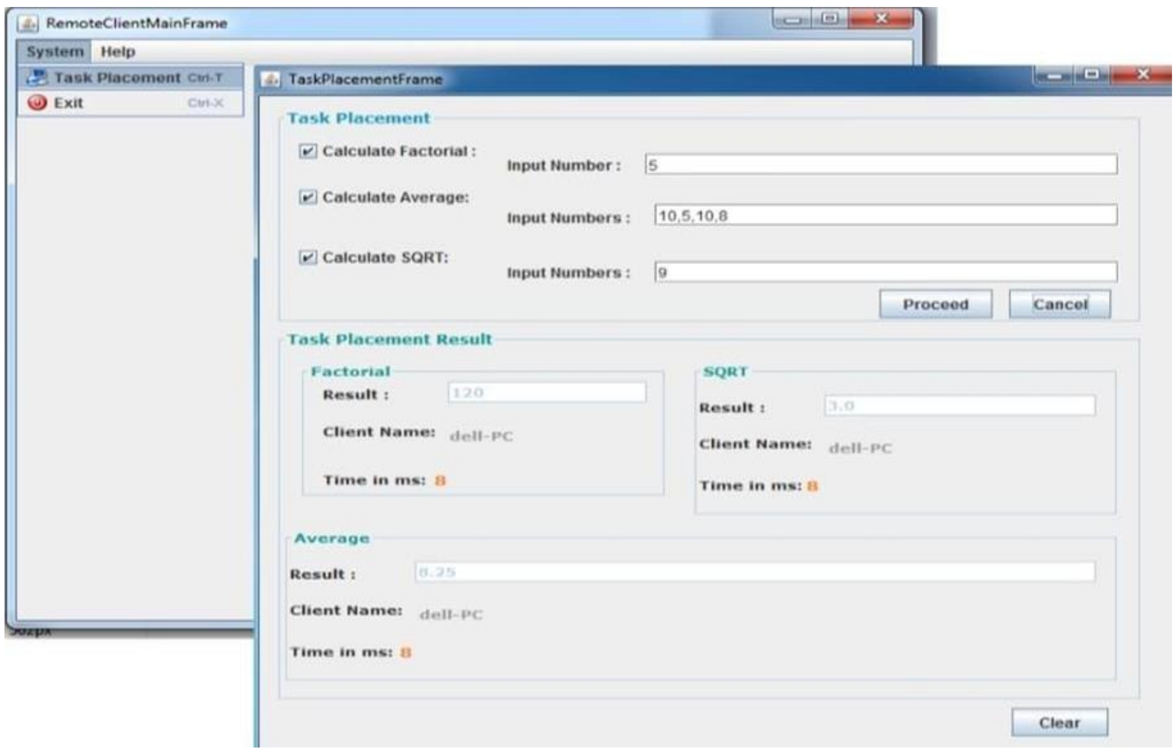
IV. SOFTWARE REQUIREMENT

- Backend: My SQL version 5.1.36
- Frontend: java 1.6(jdk 1.6.0_04)
- Neat beans: 8.1 IDE
- Hardware requirement: Port Fast ethernet switch (DLINK)
- LAN Wires

V. ADVANTAGES

- The main reason for peer-to- peer system is to execute code efficiently, since parallel programing save time, allowing the execution of application in shorter wall-clock time.
- Peer-to-peer programing goes beyond the limits imposed by sequential computing, which is often constrained by physical and practical factor. That limits the ability to construct faster sequential computer. For example, the speed of sequential computer depends on how fast data moves through its hardware.
- Resources are managed by each system share resources. Other reason for developing parallel program involved managing resource better. For example, since a single computer has a limits memory resources, ushing several computers overcome this problem for large problem.
- A feature is provisioning of compute resource on demand future trend indicate that every improvement in multiprocessor computer architecture, faster network and distributed system allow for obtaining continual gains when using parallel programming. Parallel programming is does the base of future computing system.

VI. EXPERIMENTAL RESULTS AND DISCUSSION


RemoteClientMainFrame
System Help
Task Placement Ctrl-T
Exit Ctrl-X

TaskPlacementFrame

Task Placement
- [✓] Calculate Factorial : Input Number : 5
- [✓] Calculate Average: Input Numbers : 10,5,10,8
- [✓] Calculate SQRT: Input Numbers : 9

Proceed Cancel

Task Placement Result

Factorial
Result : 120
Client Name: dell-PC
Time in ms: 8

SQRT
Result : 3.0
Client Name: dell-PC
Time in ms: 8

Average
Result : 8.25
Client Name: dell-PC
Time in ms: 8

Clear

- The figure show that the task is placed to the client who was list loaded and gives client-name and total time require to execution along with the result.

- The figure shows that the task is placed to the client which was list loaded and give the client -name and total time required to the execution along with the result.

VII. CONCLUSION

- Our system model and experimental study can be directly extended to a large size network, such as multidimensional hyper cube networks, to test their performance. In our near future work, we intend to divide the job in the system into s3eeverel and assign each class of job its own priority.

Reference

1. *D.Battr'e, S.Ewen, F.Hueske, O.Kao, V.Markl, and D.Warnek. Nephele/PACTs: Programming model and execution framework for web-scale analytical processing in SoCC'10: proceedings to the ACM Symposium on cloud computing2010, pages119-130, New York, NYK USA, 2010.ACM.*
2. *Lv Ying, Luo Zhiqiang, JieYupie, Ji Shijle, He chunjang, Chang Nacchio , Yu Zhihong.Parallel scheduling strategy for multitime scale plane security check based on unified computing resources. (2018)*
3. *Hanan M.Shupur ,subhi R .M .zeebaree ,Addulraheem jamil Ahmed,rizgar R.Zebari, Omar M.ahmed ,Bareen ShamsAdleen Tahir, Mohammed A.M.Sadeeq.A state of art survey for concurrent computation and clustering the parallel computing for distributed system. (2019)*
4. *D.warneke and O.Kao. Nephele: Efficient Parallel data processing in the cloud. In MTAGS'09: Proceedings of the 2nd workshop on many tasks computing on grids and super computers, pages 1-10, New York, NY, USA, 2009.ACM.*
5. *I.Raicu, I.Foster, and Y.Zaho. Many task computing for grid and supercomputers in many -task computing on grid and super computer, 2008.MTAGS2008.Workshop on pages 1-11 november.2008.*