International Journal of Electrical Electronics & Computer Science Engineering
Special Issue - NCSCT-2018 | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
March, 2018 | Available Online at www.ijeecse.com

Distributed and Parallel Computing using JAVA

Manvendra Singh Rathore[1], Mr. Krutibash Nayak[2]
[1]UG Scholar, [2]Assistant Professor, Department of CSE, Poornima Institute of Engineering and Technology, Jaipur, India
[1]2014pietcsmanvendra@poornima.org, [2]kruti@poornima.org

Abstract: ***This research paper is the study to develop distributed and parallel computing system using cluster architecture and parallel execution capable architecture. Here we have studied the architecture implementing the Beowulf's cluster architecture in JAVA as well as discussed, the parallel computing and its algorithms. In this modern era of computing, time plays a crucial role. We value fast and optic results. Traditional methods using C, C++ or FORTAN programming language for developing Cluster based architecture could be complex and time consuming. This paper exhibits the development of distributed and parallel computing system using java and discussed the parallel computing algorithms and issues with the help of examples and graphs.***

Keywords: Distributed and Parallel Computing, JAVA API's, Parallel Computing Algorithm, Architecture.

I. INTRODUCTION

In this modern era, we can see the advancement in the field of computing and processors, speed and performance are getting better and better but it is uncertain that this trend will continue. We are increasing the number of transistors packing into the single chip but clock speed of processors already hit the wall due to which high demand of performance in the field of computing which cannot be fulfilled by single processor alone. Here comes the concept of parallel computing including multi-processor system, multi-core system, and multi-computer systems.

This paper is focused on creating an architecture and algorithms that allows high-performance computing to be possible using multicomputer system using Java. In other words, we are implementing Beowulf's cluster architecture for a distributed system using JAVA.

Features of Java make it a good choice for heterogeneous computing. Though Java has so many benefits for using it to develop distributed computing, it also has great compatibility for parallel computing. As we know single CPU computing is limited by its performance and availability of memory. Parallel computing allows one to solve problems that need more computation power and problems which need more time for executing the task.

Now using Distributed System and Parallel Computing, we can solve larger problems in a much faster way while maintaining the quality of the result as well as can be able to work on a number of cases at a single instance.

II. ARCHITECTURE OF DISTRIBUTED SYSTEM

The Distributed system architecture can be implemented by client- server architecture, which plays an essential role in developing the multi-tier architectures. In this architecture, information and knowledge processing is not limited to single machine rather it is distributed over multiple independent computers connected in the network.

Middleware is an infrastructure that works as a bridge between the development and executing of distributed applications. Its middleware can be developed using AXIS JAVA Web Services, J2EE etc. it supports and manages the components of a distributed system Examples are communication controllers, transaction processing, data convertors.

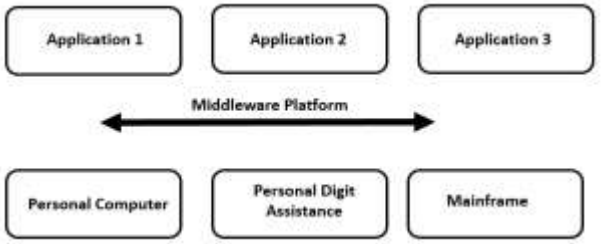

Fig. 1. Middleware as an Infrastructure for Distributed System

III. ARCHITECTURE OF CLUSTER

Here we have developed the architecture for cluster using JAVA. The system is based on Beowulf's cluster in which the main server is connected to the worker nodes through Ethernet switch. The architecture is based on divide and conquer strategy for performing computations. The design is made such that the task is first partitioned and mapped between the worker nodes by the server or Job Executor. Furthermore, along with the high computing aspect, it also comes with the package that includes fault tolerance and dynamic worker addition and removal.

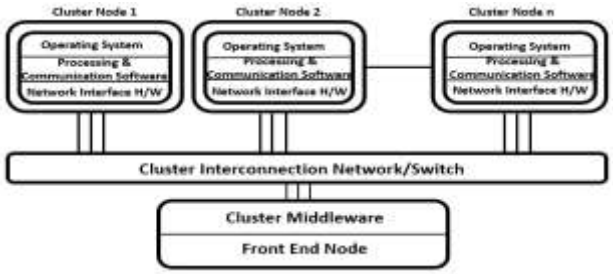

Fig. 2. Architecture of Cluster

IV. SYSTEM DESIGN AND ARCHITECTURE

The framework depends on Beowulf's group where the primary server is associated with the worker nodes through Ethernet switch. The plan is made with the end goal that the assignment is apportioned, what's more, the mapping between the worker nodes by the server and once the worker nodes finish the assignments, the outcome is

International Journal of Electrical Electronics & Computer Science Engineering
Special Issue - NCSCT-2018 | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
March, 2018 | Available Online at www.ijeecse.com

accessible to the server node which incorporates the outcome to give the client with the final output. The correspondence delay amid the booking of errand and recovering of result can be limited by either utilizing top-notch Ethernet switch or utilizing of routing algorithms or by utilizing some extraordinary means of communication like InfiniBand, Myrinet, QsNet II etc.

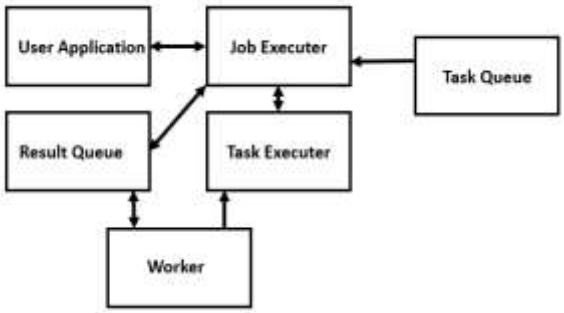

Fig. 3. Software Architecture

When the worker tries to connect with Work Server, it gives the specific Id and connected through the socket. Then it run in a different thread in the main Server. The Job Executor passes the task to the different nodes connected in the same network. The task executor helps to allocate the work and starts the computation of the task. Each node runs multiple worker session parallelly for an increasing the performance and resource utilization. When worker complete the task, it transfers the result to the Executor and the Executor enter the result to a Result Queue. After all threads are closed i.e. All worker completes the work, the Job Executor reads the various values from the result queue and provides the final result to the User. During the working of the Worker, if they crash or fault arises, the Executor detects it and throws exception to the Executor and again writes the task to the Task Queue overcoming the Fault.

The overall working of the cluster is shown in the figure. The user assigns the input to the Job Executor The program forwards small chunks of tasks as per instruction to different nodes such that there is no problem in result or calculation process. The nodes perform task assigned to them. The results are returned and integrated to provide the user with the final result or solution.

V. PARALLEL COMPUTING

Parallel computing or parallelization is a type of computation in which many calculations and computations are carried out simultaneously, operating on the concepts of large problems can often be partitioned into smaller ones, which are then solved parallelly or concurrently. In other words, if a CPU intensive problem or tasks can be divided into smaller, independent tasks, then those tasks can be allocated to different processors.

Regarding multi-threading and concurrency, Java is very helpfull. It has had features for multi-threading and could manipulate execution of threads using a low-level approach with the help of interrupt, join. Moreover, the notify and wait methods that all objects inherit could also be helpful.

VI. FOSTER'S DESIGN METHODOLOGY

It is not easy to design a parallel program from scratch without some logical methodology. It is far better to use a proven methodology that is general enough and that can be followed easily. Otherwise you will not learn from the mistakes of others. In 1995, Ian Foster proposed such a methodology, which has come to be called Foster's design methodology. It is a four-stage design process whose input is the problem statement, as shown in Figure 4. The four stages, with brief descriptions, are.

Partitioning: The process of dividing the computation and the data into pieces.

Communication: The process of determining how tasks will communicate with each other, distinguishing between local communication and global communication.

Agglomeration: The process of grouping tasks into larger tasks to improve performance or simplify programming.

Mapping: The process of assigning tasks to physical processors.

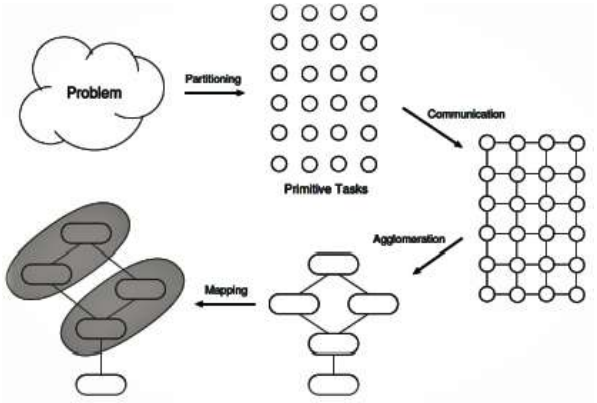

Fig. 4. Foster's Parallel Algorithm Design Methodology

A. Partitioning:

The purpose of partitioning is to discover as much parallelism as possible. This remaining stage, partitioning, is the only chance to do this; the remaining stages typically reduce the amount of parallelism, so the goal in this stage is to and all of it. There are two potential sources of parallelism: data and computation, leading to two complementary methods of extracting parallelism.

B. Communication:

When the entire computation is one sequential program, all of the data is available to all parts of the program. When that computation is divided up into independent tasks that may execute in separate processors, some of the data needed by a task may reside in its local memory, but some of it may reside in that of other tasks. As a result, these tasks may need to exchange data with one another.

International Journal of Electrical Electronics & Computer Science Engineering
Special Issue - NCSCT-2018 | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
March, 2018 | Available Online at www.ijeecse.com

This information flow is specified in the communication stage of the design.

There are two types of communication among the tasks, local and global. Local communication is when a task needs values from a small number of other tasks. Global communication is when a great many tasks must contribute data to perform a computation

C. Agglomeration:

In the first two stages of the design process, the computation is partitioned to maximize parallelism, and communication between tasks is introduced so that tasks have the data they need. The resulting algorithm is still an abstraction, because it is not designed to execute on any particular parallel computer. It may also be very inefficient, especially if there are many more tasks than processors on the target computer. This is because:

- Creating a task (process) typically uses overhead, and scheduling tasks on the processor uses overhead.
- The communication between tasks on the same processor adds artificial overhead, because if these tasks were combined into a single task, that communication would not exist.

D. Mapping:

Mapping, the final stage of Foster's methodology, is the procedure of assigning each task to a processor. Of course, this mapping problem does not arise on uniprocessors or on shared-memory computers whose operating systems provide automatic task scheduling. Therefore, we assume here that the target architecture is a distributed-memory parallel computer, i.e., a private memory multicomputer.

VII. DOMAIN DECOMPOSITION

Domain decomposition is a type of parallel algorithm in which, we first divide data into smaller sets and then we decide how to implement computations with data. Typically, our main focus is on the largest and most frequently used data structure. Strategy involved: Block allocation Suppose there exist n numbers of elements and p numbers of processors, we can allocate n/p numbers of blocks.

First element controlled by node i= i∗ (n p)

Last element controlled by process i= [(i+1)(n p)]-1

Processes controlling particular array element j is

[p(j+1)−1 n]

VIII. RESULT AND ANALYSIS

The tests were conducted on 1-16 nodes [5]. They tested the cluster in different scenarios and environments where nodes were connected at a time with specific number of cores utilized. Every computer was connected on the basis of star topology with network switches. Ethernet cables were used for the connection to the network. IP address of each computer was unique and assigned by the network switch. Number of experiments were performed using number of nodes, threads and task size. Different constraints affect the performance of the system like network connectivity strength, no. of cores used, size of chunk depending on resource availability etc. Maximum numbers of threads allocated to single node was limited to 4 as the desktop would only have 4 cores while enabling Hyper-Threading mode. Same version of JVM were used for the experiment. Following results were obtained while experimenting with different nodes and different threads on single node.

A. Windows vs Linux:

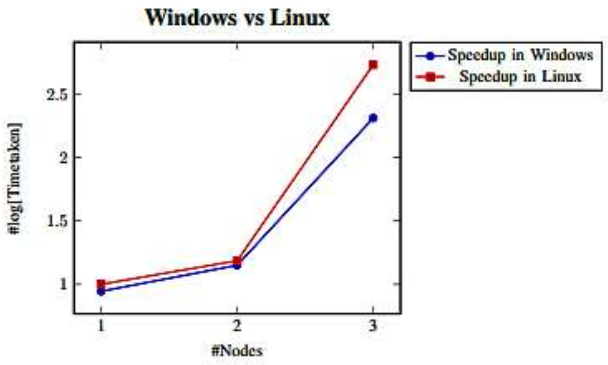

Fig. 5. Windows vs Linux

As you can see they have done the comparison of performance for the designed architecture between windows and Linux Operating System (OS) was made on same node with 4 threads on both machine. For Windows, we used Windows 10 and for Linux, Ubuntu 14.04 was used. From the following Figure 5 plotted between no. of nodes and speedup achieved, as observed Linux has more speedup than windows in the basis of execution time. Linux is always preferable choice over windows for higher performance and we can see the result. We can customize the Linux kernel as required for our designed architecture to achieve higher performance. But doing same with windows is difficult.

IX. CONCLUSION

We can conclude that using low-level API for creating purposed cluster architecture and Foster's Design Methodology may be complex in case of job scheduling and automatic resource management. When it comes on distributed system for cluster architecture, Java could outrun other languages in case of higher level of abstraction. Java can be a good choice for developing and implementation of distributed computing due to its features like portability, Graphics User Interface development, inbuilt multi-threading and network library. Heterogeneity can be easily handled with Java. Using Java or not for distributed and parallel computing is choice of the user. If the requirement includes portability, high

availability, easier software development, GUI or security Java will be the best option for that type of applications. With integration of NoSQL based Database technology, this architecture can be used to process huge datasets and will help the system to be scalable. This architecture can be made more modular for integrating and developing other applications making it more useful.

ACKNOWLEDGMENT

I am heartily thankful to Mr. Krutibash Nayak for giving me proper guidance in the field of distributed and parallel computing.

X. REFERENCES

[1] Beowulf Clustering frequently questions. www.beowulf.org/ overview.faq.html.

[2] Cluster computing: High-performance, high-availability, and highthroughput processing on a network of computers. http://www.cloudbus. org/papers/ic cluster.pdf.

[3] Distributed Shared Memory distributed global address space(dgas). https://en.wikipedia.org/wiki/Distributed shared memory.

[4] Mark A. Baker paper on “Parallel and Distributed Computing with Java”.

[5] Subarna Shakya, paper “Distributed High Performance Computing using JAVA”.

[6] Ting-Wei Hou, Fuh-Gwo, paper “Distributed and Parallel Execution of Java programs On a DSM System”.