P2PCS – A Pure Peer-to-Peer Computing System for Large Scale Computation Problems

Jigyasu Dubey
Department of Information Technology
Shri Vaishnav Institute of Technology & Science
Indore, India
jigyasudube@yahoo.co.in

Vrinda Tokekar
Institute of Engineering & Technology
Devi Ahilya Vishwavidyalaya
Indore, India
vrindatokekar@yahoo.com

***Abstract*— Complex and large scale scientific computation problems require high computing machines to process data or jobs which are expensive in terms of money. One most successful and low cost mechanism for acquiring the necessary computation power for such type of application is the Peer-to-Peer computing paradigm, which makes use of the computational power of personal computers. The peer-to-peer (P2P) systems represent the applications that allow direct communication between peers and resource harvesting. In this paper we propose a generic Peer-to-Peer computing system (P2PCS) to process complex and large scale scientific computation problems. The system utilizes the CPU cycles of desktop PCs which are connected to the network to perform the computations. We are implementing this system in JAVA technology by using Sun's JXTA –JXSE 2.5 libraries.**

Keywords- P2P; JXTA; P2P Computing; Peer

I. INTRODUCTION

In the era of 1970s & 1980s, scientific calculations & complex mathematical calculations were only run on dedicated and expensive multiprocessors like CRAY computers. The parallel machines are very expensive to build and maintain. On the other hand, there is also a huge potential computing power located in the millions of computers connected to the Internet. From the last few years, the scientific computing community has been aimed towards lower computational costs [1]. The global computing systems such as Javelin++ [2] and Bayanihanare [3] are basically centralized systems. The centralized architecture leads to some problems in scalability and accessibility [4]. The latest evolution was the use of desktop/home PCs combined with a new computing paradigm: peer-to-peer (P2P). The P2P architectures and systems are characterized by direct access between peer computers, rather than through a centralized server. The P2P model is rising as a new distributed model because of its capability to harvest the computing and storage power of hosts connected to the Internet to make their underutilized resources available to others. The P2P paradigm reduces the requirement of costly infrastructure by allowing direct communication between peers and resource harvesting [5]. It also increases the scalability and reliability by eliminating the need of a centralized point (server). Initially the applications developed were basically dedicated to file sharing, but now P2P is seen as a possible computing model for distributed and parallel computing.

Experience has shown that not only idle CPU cycles are widely available throughout the Internet; but in addition, many users are willing to share cycles [6]. These resources can be used in a Peer-to-Peer (P2P) fashion which was shown on the example of SETI@home [7] project. Applications such as distributed.net and SETI use the idle CPU cycles of thousands of computers connected to the Internet in order to break encryption codes and find signs of intelligent life in outer space. The underlying foundation for this type of application is parallel processing – breaking a large problem into smaller pieces, distributing those pieces to an array of processors, and then combining the small solutions to solve the larger problems. Popular applications for harvesting idle cycles from ordinary users, such as SETI@home [8], require donors of cycles which are manually coordinated through a centralized web site.

In this paper, we investigate the possibility of using a pure peer-to-peer network to construct a computing system. We proposed a generic distributed computational system P2PCS capable of utilizing the idle CPU cycles of any Internet computer that the peer is installed on. P2PCS runs as a standalone Java application on any Java enabled computing platform. The system is implemented in Java using JXTA [9] libraries. The P2PCS supports only those computations which can be divided into small tasks and which are embarrassingly parallel in nature. Only those PCs on which P2PCS code is installed can participate. The P2PCS system consists of two components- Task Provider and Task Processor. The task provider peer provides the task and data for processing to the task processors. The task processor peers collect the task and data from task provider and process it and after processing send the results back to the task provider.

II. RELATED WORK

In last few years, a number of P2P networking systems have been developed for various applications like file sharing and distributed computing. Napster, Gnutella, FreeNet are the example of file sharing systems while SETI@home is the example of the P2P computing system which is used to process the astronomical data only. SETI@home is the most successful application with the large number of contributors because of their exciting application theme "search for an extraterrestrial intelligence". Initial SETI project used the special super computers for the analysis of bulk data received from telescope. In 1995, David Gedye proposed the concept of a virtual super computer formed by large number

978-0-7695-4587-5/11 $26.00 © 2011 IEEE
DOI 10.1109/CICN.2011.126

of Internet connected computers for the analysis of the large amount of data received from telescope. The idea of David Gedye is implemented as SETI@home project which was launched in May 1999 and running successfully. In this project bulk of data received from telescopes are distributed to many PCs for processing by a centralized server and after completion the results are sent back to the centralized server [8][10].The online file sharing system called Napster is another example of P2P network. It is an on line music file sharing system developed by Shawn Fanning. This system allows its users to upload and download MP3 files without any restriction. Napster maintains a directory of shared file at central location and to download a file peer issue queries to the directory server to find which peer hold the desired file. Napster's search mechanism is centralized but its file sharing mechanism is decentralized. The downloading of files is done directly between the peers. The Napster was launched in June 1999 and shutdown by the court order in July 2001 due to violation of copyright rules [11] [12].

Jerome Verbeke, Neelakanth Nadgir et al. in [13] presented a decentralized P2P computing framework for large-scale computation problems named as JNGI. In this framework the computational resources are divided into groups according to their functionality. They proposed three peer groups: the monitor group, the worker group, and the task dispatcher group. The design of framework limits communication to small peer groups that enables the framework to scale to a very large number of peers. Jean-Baptiste et al. in [14] added new types of groups called similarity groups into the JNGI project. These new groups were formed on the basis of two criteria, qualitative (structural) or quantitative (performance). The qualitative criteria included OS type or JVM version where as quantitative criteria included physical characteristics such as CPU speed, bandwidth, and RAM size. However peer grouping based on geographic location criteria needs to be considered to improve the reliability. It is observed that most of the research work in the area of P2P computing is based on hybrid architecture, very few of them are considering the pure P2P architecture. Most of the load balancing algorithms and security mechanisms so far developed consider a centralized system for indexing purpose. However do not consider the decentralized nature of pure P2P network systems.

It is observes that the focus of research work in the area of P2P computing is to develop communication protocols and frameworks to accomplish data sharing and data exchange. Most of the protocols and frameworks have been developed to support the data sharing in the P2P file sharing systems. There are very few mechanisms available that use the computing power of the remote systems for computing purposes.

III. PEER-TO-PEER COMPUTING SYSTEM

The Peer-to-Peer computing system (P2PCS) utilizes the processing power of idle Desktop PCs presented at the edge of the Internet as shown in fig. 1. These Desktop PCs are known as peers. The architecture of P2PCS is shown in Fig. 2. The P2PCS system consists of two components- Task Provider and Task Processor. A peer in P2PCS at a time can be a task provider or a task processor, but not both. A user on a peer provides the job in form of computation code with the data which is to be processed. This peer becomes as a task provider in the system and other peers as task processors. The task which is provided by the user for processing must be able to split in subtasks.

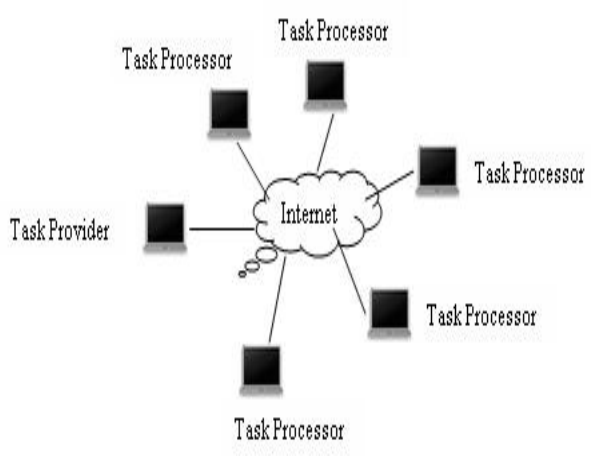

Figure 1. P2PCS Model

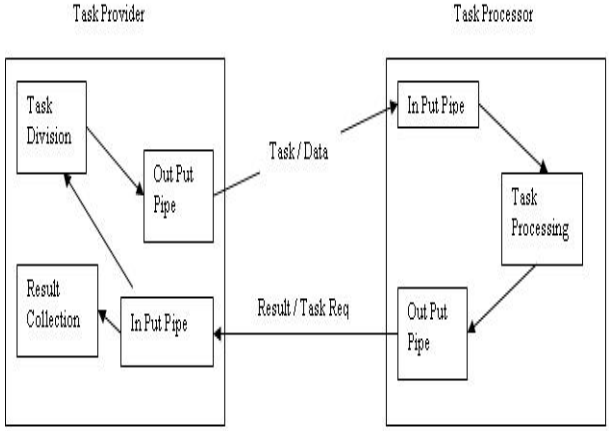

Figure 2. P2PCS Architecture

The task provider peer is responsible for splitting the task in to number of small tasks and distributes these subtasks to the task processor peers in the system for processing. The task provider is also responsible for integrating the results which it receives from the different task processors after the completion of task. The task processor peers receive the computation code and data from the task provider and return the results back to the task provider after processing the task. The proposed system will work on following algorithm:

```
Algorithm:
1.  Start
2.  Launch the peer into JXTA network;
3.  If (task is available for processing)
4.          Divide the task into multiple subtasks;
5.          Create input pipe advertisement;
6.          Build input pipe by using the pipe advertisement.
7.          Broadcast a pipe advertisement to announce the task availability;
8.          while(subtask is available for processing)
9.                  Wait for response;
10.                 if (response == result)
11.                         Store the result;
12.                 end if
13.                 else
14.                         Create output pipe;
15.                         Send the sub task and data to the peer which responds via output pipe;
16.                 end else
17.         end while
18.         if(no. of results = no. of subtasks)
19.                 Integrate all the results of subtasks;
20.         end if
21.         Go to step 3;
22. end if
23. else
24.         create input pipe;
25.         Wait for pipe advertisement which carry the message of task availability on other peer;
26.         Receive the pipe advertisement;
27.         If(advertisement contains task availability announcement)
28.                 Create output pipe;
29.                 while (sub tasks are available for processing on task provider)
30.                         Send message with the type element value  "task" to the task provider;
31.                         Receive a sub task and data;
32.                         Call appropriate method to process the sub task and data;
33.                         send the result back with the type element value "result" ;
34.                 end while
35.         end if
36.         Go to step 25;
37. end else
38. Go to step 3;
39. End
```

A. Task Provider

As the task provider program start on a peer it launches the peer into JXTA network. After that it builds and publishes an input pipe through pipe advertisement on JXTA network. From this input pipe the task provider receives the task requests from the task processors, and results. The pipe advertisement contains a message that task is available for processing. In response of this input pipe advertisement the task provider expects to receive a JXTA message with the type element defined. If the type element has a value of "results", the message will also contain an element called results. If the type element has value other than result, the task provider will assume that the remote peer is looking for new task. In this case, the message from the task processor peer should contain an element called pipe that contain an input pipe advertisement of the task processor. The task provider sends the task and data to the task processor through this pipe. The sub task which is to be processed is in form of serialized object. Upon reception of any result, the Task Provider integrates it with the previous results sent by other task processor (if any) thus when the last task processor sends the result, Task Provider forms the complete result.

B. Task Processor

In the P2PCS the task processor has two important functionalities 1) to receive and process the sub task and 2) to return results back to the task provider peer. Initially the task processor peer discovers the pipe advertisement published by task provider, requests the task through an output pipe and then connects with. The task processor will send a message with a type element having a value of task. The message also includes a pipe advertisement which task processor has created to receive the task and data from the task provider. When the task processor program is run on a peer it launches the peer into JXTA network and it will attempt to find and connect to the pipe advertised by the task provider peer. After receiving the pipe advertisement from task provider the task processor peer sends a message to the task provider with a type element other then result and a pipe advertisement of its input pipe. The task processor peer uses this pipe to receive the task and data from the task provider peer. The task is received by the listener of the task processor's input pipe. The object received in the message element will call the processing code to process the data which is attached in message. When processing is done the task processor peer sends a message back to the task provider with the type element result and other element hold the actual results.

IV. P2PCS IMPLEMENTATION

The implementation of proposed system is done in JAVA by using JXTA-JXSE 2.5 libraries. The JXTA libraries provide fundamental infrastructure to implement a virtual network of peers over an existing physical network. In the proposed system, for the communication between peers we use JXTA's unicast pipe. At each peer our program creates two pipes, an input pipe and an output pipe. The input pipe is used to receive the messages from other peers while output pipe is used to send messages to other peers. We used a homogeneous collection of 6 machines to carry out the experiments. Each machine has equipped with Intel® Core™ 2 CPU 4300@1.80 GHz 900 MHz processor, 2 GB RAM and Windows Vista™ Ultimate 32 bit operating system. All these machines are connected via 100 Mbps switch using CAT-5 Ethernet cable.

We develop an application in JAVA to find out Summation of all natural numbers from 0 to given fixed number. This application is embarrassingly parallel in nature. In our setup one machine is act as task provider and other five machines used as task processors. In the first experiment we calculate the sum of natural number between 0 and 200000. This range is specified by the user on command line at task provider machine. On one machine task provider program is installed and it divides above task into small tasks

(divide the range 0-200000 in equal sub ranges 0-1000, 1001-2000, 2001-3000, ….. up to 200000) and provides these small tasks to the task processor program running on other five machines for processing. The time is measured from the moment when range of numbers given to task provider to the moment all the results received from the task processors. The same task, summation of natural numbers between 0 and 200000, is executed on a single machine with same specification and time is measured. On single machine mention task is completed in 72 seconds while when same task is executed on five machines it takes 39 seconds to complete.

V. CONCLUSION

The proposed Peer-to-Peer computing system achieves high processing capability very economically as compare to other approaches like super computers, grid computing, and cluster computing. It may be very useful for the scientific community in important scientific projects and applications such as astronomical measurements, multimedia streaming, bio-medical studies, academic research, weather forecast and data mining applications where very large volumes of jobs or data are available for processing. These applications require high computing machines to process data or jobs which are expensive in terms of money. The P2PCS system act like a virtual super computer with varying processing power by utilizing the ideal CPU cycles of the desktop PCs connected to the internet. It is a generic distributed computing system. Anyone can use this system for processing the task and data by changing the task processing code.

The existing global computing systems are centralized which leads to some problems in scalability and accessibility. The proposed system has decentralized architecture. It uses the JXTA protocol for communication and collaboration between the peers. In this system the peer can be a task processor or task provider. When a peer has the large task to process it will become as task provider and when peer's CPU goes ideal it will become a task processor.

REFERENCES

[1] Pawel Jurczyk, Maciej Golenia, Maciej Malawski, Dawid Kurzyniec, Marian Bubak ; and Vaidy S. Sunderam, "P2P Computing System with Remote Method Invocation over JXTA", 6th International Parallel Processing and Applied Mathematics 2005, pp. 667-674

[2] M. O. Neary, S. P. Brydon, P. Kmiec, S. Rollins, P. Capello, "Javelin++: Scalability Issues in Global Computing," Proceedings of the ACM Java Grande 1999 Conference, June 12-14, 1999, San Francisco, California.

[3] Luis F. G. Sarmenta, "Volunteer Computing", Ph.D. Thesis, MIT Department of Electrical Engineering and Computer Science, March 2001.

[4] N.A.Al-Dmour and W.J.Teahan, "ParCop:A Decentralized Peer-to-Peer Computing System", In proceedings of the Third International Workshop on Parallel and Distributed Computing, 2004, pp. 162 – 168,

[5] Jigyasu Dubey, Dr. (Mrs.) Vrinda Tokekar, Anand Rajavat, "A Study of P2P Computing Networks", in proceedings of ICCET' 10, pp 623-627, 13-14 Nov. 2010, Jodhpur, India.

[6] Virginia Lo, Daniel Zappala, Dayi Zhou, Yuhong Liu, and Shanyu Zhao, "Cluster Computing on the Fly: P2P Scheduling of Idle Cycles in the Internet", In Proc. 3rd International Workshop on Peer-to-Peer System (IPTPS 2004). San Diego,Feb. 2004.

[7] SETI@home: Search for extraterrestrial intelligence at home (2003) http://setiathome.ssl.berkeley.edu/.

[8] D. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. SETI@home: An Experiment in public-resource computing. Communicationsof the ACM, 45:56–61, 2002.

[9] JXTA Java ™ Standard Edition v2.5: Programmers Guide September 10 th , 2007

[10] Korpela, E., Werthimer, D., Anderson, D., Cobb, J., and Lebofsky, "SETI@home: Massively Distributed Computing for SETI," In Journal on Computing in Science and Engineering, Volume 3, Issue 1, pp. 78 – 83, 2001.

[11] S. Androutsellis-Theotokis and D. Spinellis, "A Survey of Peer-to-Peer File Sharing Technologies," White Paper, Electronic Trading Research Unit (ELTRUN), Athens University of Economics and Business, 2002.

[12] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," In Proceedings of ACM/SPIE Multimedia Computing and Networking (MMCN '02), 2002.

[13] Jerome Verbeke, Neelakanth Nadgir, Greg Ruetsch, Ilya Sharapov, "Framework for Peer-to-Peer Distributed Computing in a Heterogeneous, Decentralized Environment," In Proceedings of the 3rd International Workshop on Grid Computing, pp.1–12, Year 2002.

[14] Jean-Baptiste Ernst-Desmulier, Julien Bourgeois and Francois Spies, Jerome Verbeke, "Adding New Features In A Peer-to-Peer Distributed Computing Framework," In Proceedings of the 13th Euromicro Conference on Parallel, Distributed and Network-Based Processing (Euromicro-PDP'05), pp.34 – 41, 2005.