Collaborative Framework for Distributed Computing in P2P Network

B. Swsssaminathan* and Sheila Anand**

Rajalakshmi Engineering College, Chennai, India
{Swamikb,sheila.anand}@gmail.com

Abstract. In this paper, we propose a Collaborative Framework for P2P to enable distribution processing of tasks or application among multiple peers. P2P networks provide a completely decentralized environment wherein each peer can provide services to other peers in the network. Most of the current work on P2P is in the area of data or content sharing. We propose a generic collaborative framework to facilitate distributed processing among peers by enabling provision for job or task scheduling, distributing of a task among multiple peers, messaging between peers, monitoring job execution, follow up and obtain the results of the completed tasks. Peer profiles are maintained to enable peers to discover other willing peers and explore their capacity and capability for distributed processing. Job profiles are proposed to maintain the details of submitted jobs, the job steps or granules and the order of execution of these job granules. A Job Manager component is present in each peer to schedule the jobs and follow up with the participating peers to obtain the result. We have tested the collaboration framework with an example and have presented the results and conclusion.

Keywords: P2P, Collaborative framework, Distributed processing, Distributed computing.

1 Introduction

In P2P system, each computer system or peer node acts as both client and server. As a server, it provides resources to be shared by other peers and as a client it uses or access resources of other peers. P2P networks enable peers to share data and content, communicate through message passing and also distribute the processing and computing. In a pure P2P network, the overlay links between peers are established arbitrarily. [5]. A new peer joining a network of peers can form its own links or copy the existing links of other peers by exchange of messages and route information. When a peer wishes to find a data or other resource, it normally floods the network with its query for the data. Peers having the resource will respond to the query. More than one peer can have the resource and respond to this query. The requesting peer chooses a peer, establishes a direct link with the peer and downloads the data. P2P

* Associate Professor, Research Scholar.
** Dean (Research), Computer Studies.

systems provide a completely decentralized environment wherein each system can provide services to each other systems in the network. The network of peers can thus be easily scaled up to any number and is more reliable. P2P can offer a robust solution that uses optimum bandwidth and network utilization. Some of the popular P2P networks are Napster, Gnutella and Kazaa which were primarily designed to share files and other data resources among multiple peers [15].

Distributed computing involves using many autonomous computer systems that come together to complete a task. One of the common applications of distributed computing is to process large amounts of data in a short span of time. An application or task can be broken up into many discrete and semi-independent parts which can be distributed for processing over multiple computer systems. With Internet as the communicating medium, peers can establish a flexible and reliable computational platform with enormous distributed computing potential. Distribution computing uses message passing whereby objects (code, data) can be exchanged between the computer systems that have come together to perform a large task or application. CORBA and DCOM example of distributed computing that provide a set of standards for object communication.

Many applications like BitTorrent and Limeware have been developed to facilitate sharing of data, music, video etc. There has been less focus on applications for distributed processing or computing. In this paper, we propose a Collaborative framework for P2P networks to enable distributed processing The objective of the framework is to facilitate distributed processing among peers by enabling provision for job or task scheduling, distributing of a task among multiple peers, messaging between peers, monitoring job execution, follow up and obtain the results of the completed tasks. The collaborative framework can also provide to establish the credibility and reliability of peers to enable peers to choose reliable partners for execution of jobs. The rest of the paper is organized as follows. Section 2 provides related work on this area and Section 3 describes the proposed P2P collaborative framework. In Section 4, we have applied the framework for a sample job and present the results. Section 5 concludes the paper.

2 Related Work

In this section, we briefly discuss research work relating to distributed computing. A Distributed computing system [10] consists of a set of independent computers connected by a computer network and operational with distributed system software. Jobs are split into independent small parts and processing of each of the job parts is done on individual PCs and the results are collected on a central server. The central server distributes the job parts among different PCs on the Internet and is also responsible for job coordination and completion. The clients take advantage of inactive periods to perform computation requested by the server.

There are two primary reasons for using distributed systems and distributed computing [1]. The nature of the application may need the use of a communication network to connect many numbers of computers. Data produced at one physical location may be required at another location and the communication network is used to make this possible. Secondly, it may be a case of using in principle a single computer to perform a specific job, but the use of a distributed system is advantageous from the viewpoint of time, efficiency and performance. For example, several low-end computers may together provide a more cost-efficient solution as compared to a single high-end computer.

SETI@home (Search for Extra Terrestrial Intelligence) is an Internet-based volunteer computing system which employs the Berkeley Open Infrastructure for Network Computing (BOINC) platform [13]. Originally, it was designed to search for radio signals emitted in the outer space and thus search for any signs of extra-terrestrial beings. Later, it has branched out into many activities such as parallel and distributed computing. The aim of SETI@home is to demonstrate viability of public-participation distributed scientific computing, and to increase public knowledge and awareness of SETI and radio astronomy. The original design of SETI was centralized with one server and data was distributed to clients using proxy servers. The clients process the job given to them and submit the results to server. As all interactions are with the server, there is a problem of load handling.

Distributed object systems are based on object-oriented programs where objects consisting of code and data are distributed across a network. Remote Method Invocation (RMI) and Common Object Request Broker Architecture (CORBA) are instances of such systems. Client-specific optimization is required for efficient performance thus making such systems difficult to maintain. RMI passes the objects as parameters to remote methods. Remote Procedure Call (RPC) is a dominant technique for constructing distributed, client-server based applications [7]. It is based on extending the notion of conventional or local procedure calls to making remote calls without the need to know the details of network interfaces. RPC tries to isolate the data communication layer from the application layer. RPC makes the client/server model of computing more powerful and easier to program, but the remoteness is not truly transparent to the client.

We have applied the concepts of distributed computing to P2P networks to develop a collaborative framework for distributed processing among many peers.

3 Collaborative Framework for P2P Networssk

In this paper, we propose a P2P collaborative framework to distribute and share the processing load among multiple peers. We propose a generic framework with the use of profiles to stores the details of peers who are willing to collaborate with other peers for job execution. Job profiles are proposed to maintain the details of submitted jobs, the job steps and the order of execution of these job steps.

Peers could be of types: initiator and participating peers. The initiator peer submits jobs to the participating peers which in turn work on the job steps or granules and send the results back to the initiator. The initiator peer collects the response of all the job steps from the participating peer and aggregates the results. The initiator peers can also provide feedback on job execution of the participating peers; thereby providing a method to evaluate the credibility and reliability of peers for collaboration on execution of jobs. The proposed collaborative framework for distributed computing in P2P network is given in Figure 1.

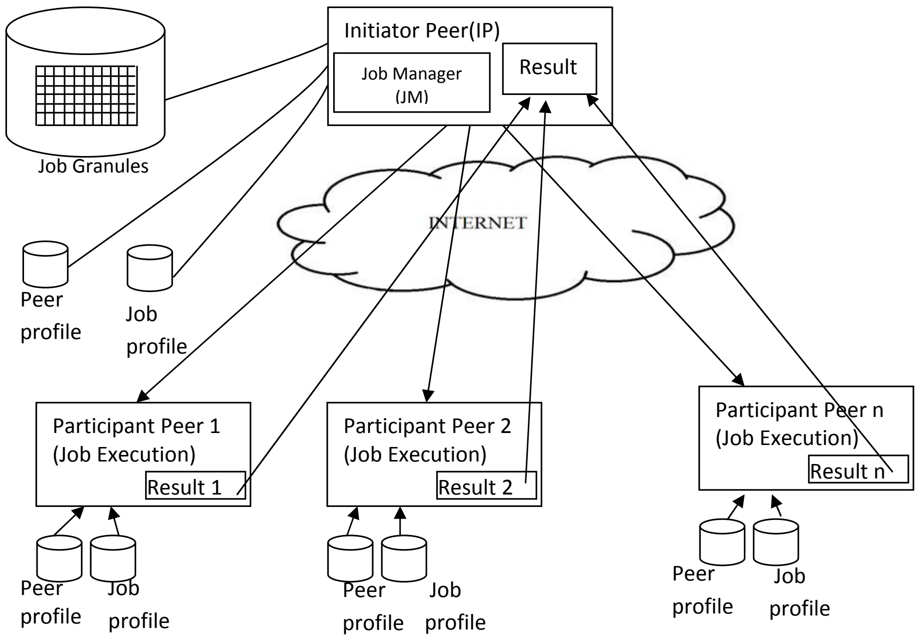


Fig. 1. Collaborative Framework For P2P Network

Peers, who are willing to collaborate with other peers for job execution, advertise their services. Such peer details are maintained as peer profile by each individual peer. The peer profile contains information on the processing capabilities of the peer node; such as CPU specification, memory available for processing, secondary storage space available for temporary storage of data of intermediate steps and final result, operating system, software installed etc. When a job has to be executed, the initiating peer will select the peers based on peer profile and their capacity to execute the job.

The Initiating Peer (IP) which would like to submit a job for distributed processing would first have to break the job into steps or modules which we call as job granules. These job granules are to be executed by the participating peers. Job profile maintains details of each submitted job, the job granules and the order of execution of these job granules. Details of which of the job granules can be executed in parallel and those that have to be executed in sequence and in which order are maintained.

Peers who wish to submit a job would first determine from the peer profile, the peers suitable for execution of the concerned job. The initiating peer will then send a request message to the peers to check their availability and willingness to do the job. Willing peers will respond with an affirmative message while other peers would send back a decline message. No response from peers within a specified time would also be treated as decline message. Based on the response from the peers, a set of peers are selected by the IP as the Participating Peers (PP) for the job execution. A Job Manager (JM) module is present in each peer to schedule the job granules and follow up with the respective peers to obtain the result. JM will send the job granule to the PPs for execution. The participating peers will execute the job granule and send back a successfully completed message and the result. In case of failure of job execution,

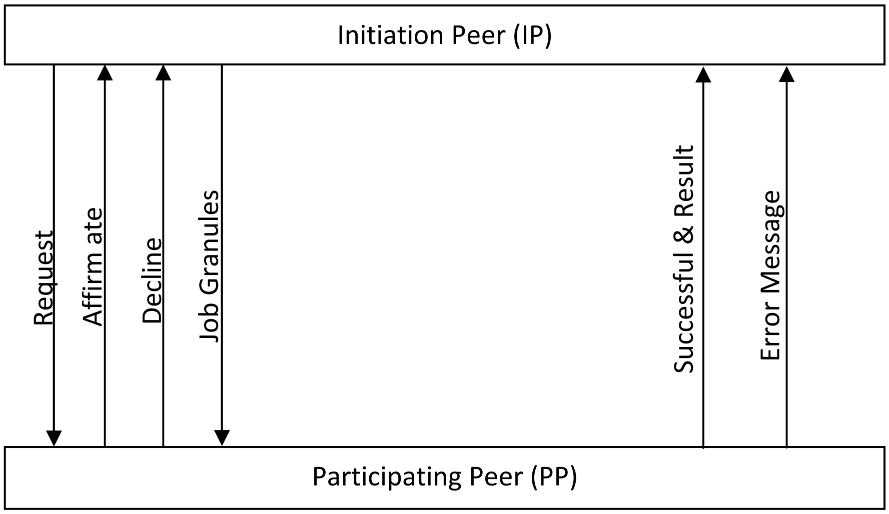


Fig. 2. Interaction among Peers for Job Execution

then an error message would be sent by the PP to the JM. The interactions between IP and PP for job execution are depicted in Figure 2.

JM would ensure that job execution follows the steps and order specified in the job profile. JM would wait for a specified time to obtain the result. If the successful or error message is not obtained within this specified time, then JM would look for another peer to execute the job granule. In case of execution errors, depending on the error type, JM would decide to send the same job granule to another peer. JM would abandon the job in case of serious or critical errors. JM would also abandon the job if required peers are not available for job execution, or there is a delay in obtaining the result. A threshold limit is set for the job execution and if the job is not completed within the threshold, then JM would abandon the job. IP would then examine why the job execution was abandoned. In case of errors in job processing, IP would first need to make necessary corrections before re-submitting the job. If the job was abandoned because of other reasons like network failure, peer non-availability etc, IP could wait and re-submit the job after an interval.

The collaborative framework can also provide feedback mechanisms to establish the credibility and reliability of peers. This would assist peers in choosing reliable partners for execution of jobs

4 Implementation and Results

We have implemented the proposed P2P collaborative framework using JXTA and Windows operating system. We have used JXTA services for peer registration, peer discovery, peer messaging and content communication and management. The nature and details of the job are kept partially hidden from the PP as IP sends only the data and execution module as class file.

As the first step, the peers advertise giving details of their processing capability. The peer details are updated in the Peer Profile which is maintained in XML format and a sample peer profile is given below. The Peer profile has been described using XML tag for Peer Name, Peer ID, Hardware details like Processor, processor details, Memory, etc. Tags have also been provided for installed software and versions. A typical peer profile is given below.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPEjxta:Profile>
<jxta:Profile type="jxta: Peer Profile" xmlns:jxta="http://jxta.org">
    <PID>urn:jxta:uuid-
596162616461626l4A78746150325033CB4691E17A5E49DB89F0A6CADB9C60
2E03
    </PID>
    <Name>
            peer1
    </Name>
    <Desc>
            Profile of the peer
    </Desc>
    <HWArch>
            X86
    </HWArch>
    <NoCores>
            2
    </NoCores>
    <RAM>
            3 GB
    </RAM>
    <OS>
            Windows
    <OS>
    <OSVer>
            XP
    </OSVer>
</jxta:Profile>
```

The profile can be extended to provide complete and generic details of the peers. The IP uses the peer profile to select PP for execution of jobs.

The Job Profile is also maintained in XML format and we give below a sample format. A job profile has the following specifications about a job, Job ID, Step ID, File Name, Class Name of executable module. The Dependency tag is used to indicate job granules which have to be completed the prior to execution of the concern job granules. For example GranID 2 can be executed only after completion of GranID 1.

```
<?xml version="1.0" encoding="UTF-8"?>
    <!DOCTYPEjxta:JobProfile>
    <jxta:JobProfile type="jxta: Peer Profile" xmlns:jxta="http://jxta.org">
        <JobID>
                    1
        </JobID>
        <GranID>
                    1
        </GranID>
        <FileName>
                    Input.txt
        </FileName>
        <ClassName>
                    ExecuteClass2
        </ClassName>
        <Dependency>
                    0
        </Dependency>
        < GranID>
                    2
        </ GranID >
        <FileName>
                    Input.txt
        </FileName>
        <ClassName>
                    ExecuteClass2
        </ClassName>
        <Dependency>
                    1
        </Dependency>
    </jxta:JobProfile>
```

We explain the job submission and execution process using a standard deviation calculation. A Job Manager (JM) module is present in each peer to schedule the job granules and follow up with the participating peers to obtain the result. We have calculated the time taken by a single peer to execute the job. A threshold limit is set for the complete job execution which is twice the time taken by a single peer to complete the job. We have assumed the same time as threshold limit for each job step as well.

Standard deviation σ is calculated using the formula:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{i=N} (x_i - \bar{x})^2}{N}}$$

Where $$\bar{x} = \frac{\sum_{i-1}^{i=N} x_i}{N}$$

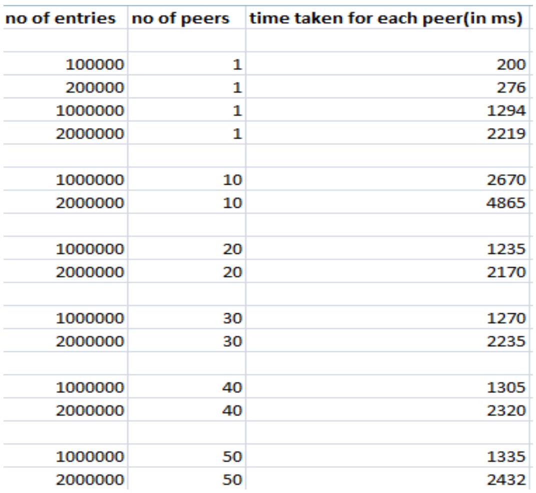
x is a list of numbers and $\bar{x}$ is the average of the numbers. x_i is the ith number in the list. N is the count of the list of numbers.

The execution of the equation among multiple PP was tested using a range of 10 – 20 lakh numbers. Since the list of numbers is huge, it is a time consuming process when only one peer is doing the calculation. The job is divided into granules and represented as steps below:

Step1: The list of numbers N are divided into sets S of lakh of numbers
Step2: Each set is sent to a different PP for summing up. The count of the numbers in the set can is determined in parallel.
Step3: The total of all the individual sums is determined from the results of the previous step and average is calculated.
Step4: $\sum_{i=1}^{i=n}(x_i - \bar{x})^2$ is calculated in parallel for each of the set of numbers S by PPs
Step5: Final step is to calculate the σ

We have compared the performance for calculating standard deviation using a single peer and multiple peers as shown in Table 1. There was no significant difference in time when the dataset was small, that is, 1 or 2 lakh of data entries. Likewise, when the number of peers (1 - 10) was small, there was an increase in the computation time because of the communication delay. Also when the number of peers was large, there was no significant difference in the total processing time. For a dataset of 10 or 20 lakhs, we obtained optimum result with 20 peers.

Table 1. Dataset for Standard Deviation



no of entries	no of peers	time taken for each peer(in ms)
100000	1	200
200000	1	276
1000000	1	1294
2000000	1	2219
1000000	10	2670
2000000	10	4865
1000000	20	1235
2000000	20	2170
1000000	30	1270
2000000	30	2235
1000000	40	1305
2000000	40	2320
1000000	50	1335
2000000	50	2432

The results are graphically depicted in Figure 3.

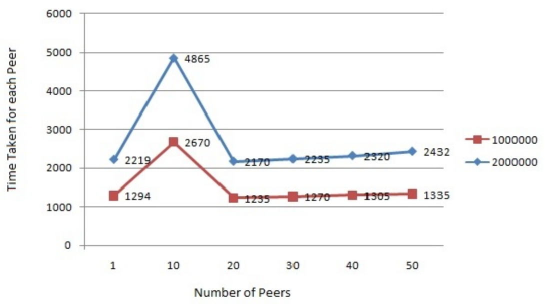


Fig. 3. Graph for the dataset

It can be concluded that for large jobs and datasets, it is significantly faster with multiple peers sharing the processing load. In analyzing the performance benefits of the proposed system, it is clear that as the number of peers in a P2P environment increases, the time taken to process the job modules reduces.

5 Conclusion and Future Work

The proposed collaborative framework for P2P provides a scalable and reliable means for processing large jobs. The proposed framework is a generic one where peer capability and job details are described using peer profile and job profiles. In the given example, execution modules in the form of class files are sent to the participating peers so that the nature of the job is kept hidden from the peers. Likewise, the peers do not have access to the full dataset thereby providing a certain level of data security. As future work, we propose to use well known P2P data distribution protocols and applications like FTP, Bit Torrent, etc. We also aim to develop an efficient scheduler for job execution and management.

References

[1] Berry, K.: Distributed and Grid Computing via the Browser Computing Research. Villanova University, Villanova (2009), `http://www.csc.villanova.edu/~tway/courses/csc3990/f2009/csrs2009/Kevin_Berry_Grid_Computing_CSRS_2009.pdf`

[2] Costa, F., Silva, L., Kelley, I., Fedak, G.: Optimizing Data Distribution layer of BOINC with Bit Torrent. In: IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008, pp. 1–8 (2008)

[3] Boldrin, F., Taddia, C., Mazzini, G.: Distributed Computing Through Web Browser. In: IEEE 66th Vehicular Technology Conference, VTC-2007, September 30-October 3, pp. 2020–2024. Univ. of Ferrara, Ferrara (2007)

[4] Ho, K., Wu, J., Sum, J.: On the Session Lifetime Distribution of Gnutella. Providence University, Sha-Lu (2007) `http://www.citeseerx.ist.psu.edu/`

[5] Awan, A., Ferreira, R.A., Jagannathan, S., Grama, A.: Unstructured Peer-to-Peer Networks for Sharing Processor Cycles, vol. 1. Purdue University, USA (2005)

[6] Petrie, D.: A Framework for Session Initiation Protocol User Agent Profile Delivery. PingtelCorp (October 24, 2004)

[7] Tilevich, E., Cook, W.R., Jiao, Y.: Explicit Batching for Distributed Objects, Computer Science Department, Virginia Tech

[8] Milojicic, D.S., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rollins, S., Xu, Z.: Peer-to-Peer Computing (July 3, 2003), `http://www.hpl.hp.com/techreports/2002/HPL-2002-57R1.html`

[9] Brookshier, D., Govoni, D., Krishnan, N., Soto, J.C.: JXTA: Java^{TM} P2P Programming. Sams Publishing (March 22, 2002)

[10] Preliminary Research on Grid-based Remote Sensing Image distributed Processing-2007 IFIP International Conference on Network and Parallel Computing

[11] JXTA Java Standard Edition V2.5: Programmer's Guide

[12] Client-Server Model, `http://www.cs.umbc.edu/~mgrass2/cmsc645/`

[13] SETI@home, `http://www.setiathome.berkeley.edu/`

[14] Gnutella, `http://www.gnutella.org`

[15] JXTA Protocol, `https://jxta.dev.java.net/`

[16] RPC and RMI, `http://www.careerride.com/RMI-advantages-and-disadvantages-of-RPC.aspx`

[17] ClassLoader, `http://download.oracle.com/javase/1.4.2/docs/api/java/lang/ClassLoader.html`