Journal of Data Mining and Knowledge Discovery
ISSN: 2229–6662 & ISSN: 2229–6670, Volume 3, Issue 2, 2012, pp.-79-82.
Available online at http://www. bioinfo. in/contents. php?id=42

PARELLEL PROCESSING USING DISTRIBUTED COMPUTING SYSTEM FOR DATA CONVERSION

NIMODIYA K.H.*, KELANI N.A., KULLARWAR S.S. AND REDDY S.M.

Department of Computer Science & Engineering, J.D.I.E.T, Yavatmal, MS, India.
*Corresponding Author: Email- komalnimodiya.7@gmail.com

Received: March 15, 2012; Accepted: April 12, 2012

Abstract- Distributed computing deals with hardware and software systems containing more than one processing element or storage element, concurrent processes, or multiple programs, running under a loosely or tightly controlled administration. In distributed computing a program is divide into parts that run simultaneously on multiple computers communicating over a network. Distributed computing is a form of parallel computing, but parallel computing is most commonly used to describe program parts running concurrently on multiple processors in the same computer. Both types of processing require dividing a program into parts that can run simultaneously, but distributed programs often must deal with assorted environments, network links of varying latencies, and unpredictable failures in the network or the computers. There are many different types of distributed computing systems and many challenges to overcome in successfully designing one. The main goal of this paper is to connect users and resources in a transparent, open, and scalable way. Ideally this arrangement is drastically more fault tolerant and more powerful than many combinations of stand-alone computer systems.
Keywords- Distributed computing, parallel processing, data conversion, distributed programming.

Citation: Nimodiya K.H., et al. (2012) Parellel Processing using Distributed Computing System for Data Conversion. Journal of Data Mining and Knowledge Discovery, ISSN: 2229–6662 & ISSN: 2229–6670, Volume 3, Issue 2, pp.-79-82.

Introduction

The application of several processors to a single task is an older idea with a relatively large literature. The advent of very large-scale integrated technology has made testing the idea realistic, and the fact that single processor systems are impending their maximum performance level has made it crucial. We shall show, however, that victorious use of parallel processing imposes rigorous performance necessities on algorithms, software, and architecture. The so-called asynchronous systems which use a few tightly coupled high-speed processors are a natural development from high-speed single-processor systems.In fact, systems with two to four processors will soon be available (for example, the Cray X-MP, the Cray-2, and the Control Data System 2XX).
To estimate the speedup of a tightly coupled system on a single application, we use a model of parallel computation introduced by Ware. We define a as the fraction of work in the application that can be processed in parallel. Then we make a simplifying assumption of a two-state machine; that is, at any instant either all p processors are operating or only one processor is operating. Consider the condition user having 10000 document to process and each having large data in this case project need to employee the system which will transfer processing over the network system and save output on the server or main system. So the implemented system will aimed at parallel processing of provided task.
In Distributed Computing approach, it is followed to assign a job to a processor if it is idle. The focus is now on how to optimize resources to decrease the energy consumption by volumes of computing equipments to deal with green and sustainability issues [10]. Various hardware and software architectures are used for distributed computing. At a lower level, it is necessary to intercon-

nect multiple CPUs with some sort of network, regardless of whether that network is printed onto a circuit board or made up of loosely-coupled devices and cables. At a higher level, it is necessary to interconnect processes running on those CPUs with some sort of communication system.

Literature survey

Various algorithm and models have been proposed, mostly heuristic in nature, as the optimal solution often requires future knowledge and is computationally intensive. The most widely approach for studying DLB algorithms is analytic modeling and simulation. For analytic modeling, the computer system is modeled as a queuing network with job arrivals and their resource consumptions following certain probabilistic patterns. Queuing network solution techniques are used to compute performance measures [2, 9, 8, 7] Due to limitations of the solution techniques, simulation is often resorted to for approximate solutions [5, 4]. Some of the-source-initiated DLB algorithms are by Eager [8, 7, 6].

Task partitioning is proposed by Deelman et al [11]. It partitions a workflow into multiple sub-workflows which are executed sequentially. Rather than mapping the entire workflow on Grids, allocates resources to tasks in one sub-workflow at a time.

Distributed Programming Architectures

Distributed programming typically falls into one of several basic architectures or categories: Client-server, 3-tier architecture, N-tier architecture, Distributed objects, loose coupling, or tight coupling [1].

- **Client-server**

Smart client code contacts the server for data, then formats and displays it to the user. Input at the client is committed back to the server when it represents a permanent change.

- **3-tier architecture**

Three tier systems move the client intelligence to a middle tier so that stateless clients can be used. This simplifies application deployment. Most web applications are 3-Tier.

- **N-tier architecture**

N-Tier refers typically to web applications which further forward their requests to other enterprise services. This type of application is the one most responsible for the success of application servers.

- **Tightly coupled (clustered)**

refers typically to a cluster of machines that closely work together, running a shared process in parallel. The task is subdivided in parts that are made individually by each one and then put back together to make the final result.

- **Peer-to-peer**

an architecture where there is no special machine or machines that provide a service or manage the network resources. Instead all responsibilities are uniformly divided among all machines, known as peers. Peers can serve both as clients and servers.

- **Space based**

refers to an infrastructure that creates the illusion (virtualization) of one single address-space. Data are transparently replicated according to application needs. Decoupling in time, space and reference is achieved.

Another basic aspect of distributed computing architecture is the method of communicating and coordinating work among concurrent processes. Through various message passing protocols, processes may communicate directly with one another, typically in a master/slave relationship. Alternatively, a "database-centric" architecture can enable distributed computing to be done without any form of direct inter-process communication, by utilizing a shared database.

Working Modules

Figure 1. Shows the basic block diagram of the project. Actual task processing needs a series of steps to be performed. These series of processes are simultaneously executed on different client machines. Basically here we are distributing the no. of files on to the network through the shared database.

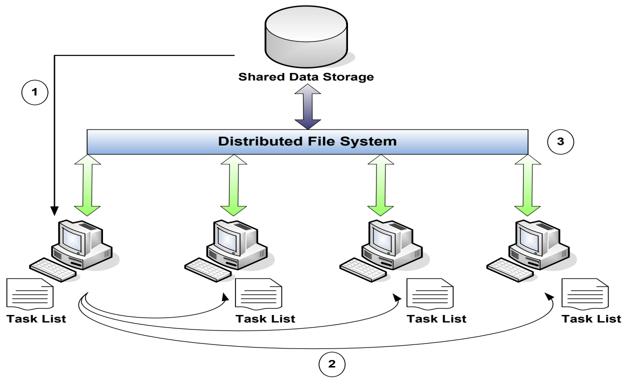


Fig. 1- System Architecture

- **Master / Slave System**

Parallel processing system built using server / client technology. Where master server act as a process manager system. In this system whenever network initialize or the first system user started in the network will check for active server in the network if no server running found then it will become host or master server and other were become slave system. This feature can be dynamic or static which means user can disable or enable this feature.

- **Task (Conversion)**

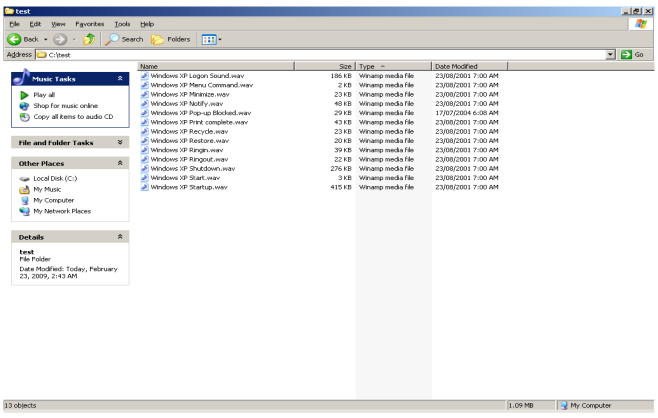
This is the main input to the parallel processing system. First user need to define the task to perform. In this system it will consider a task of batch file format conversion. Data can be passed to the client to process or it can be placed on shared data storage location from where all clients will fetch data to process.

( ) Single File Conversion ( ) Multiple File Conversion [✓ Batch File Conversion]

Fig. 2- File Conversion Methods

- **Task Assigning**

Once the user provides input task on master server then master will analyze the task and divide it in to proper task and create its task list for client. Once all clients get the task list to process it will start processing. As implemented system will work on shared data storage it will reduce network processing and network traffic by removing data transfer processing.

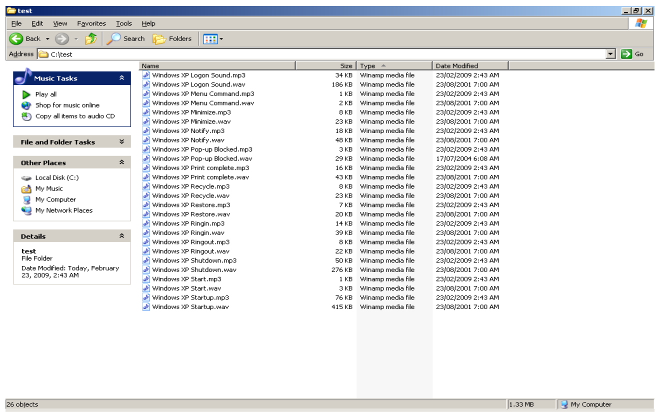


Name	Size	Type	Date Modified
Windows XP Logon Sound.wav	186 KB	Winamp media file	23/08/2001 7:00 AM
Windows XP Menu Command.wav	2 KB	Winamp media file	23/08/2001 7:00 AM
Windows XP Minimize.wav	23 KB	Winamp media file	23/08/2001 7:00 AM
Windows XP Notify.wav	48 KB	Winamp media file	23/08/2001 7:00 AM
Windows XP Pop-up Blocked.wav	29 KB	Winamp media file	17/07/2004 6:08 AM
Windows XP Print complete.wav	43 KB	Winamp media file	23/08/2001 7:00 AM
Windows XP Recycle.wav	23 KB	Winamp media file	23/08/2001 7:00 AM
Windows XP Restore.wav	20 KB	Winamp media file	23/08/2001 7:00 AM
Windows XP Ringin.wav	39 KB	Winamp media file	23/08/2001 7:00 AM
Windows XP Ringout.wav	22 KB	Winamp media file	23/08/2001 7:00 AM
Windows XP Shutdown.wav	276 KB	Winamp media file	23/08/2001 7:00 AM
Windows XP Start.wav	3 KB	Winamp media file	23/08/2001 7:00 AM
Windows XP Startup.wav	415 KB	Winamp media file	23/08/2001 7:00 AM

Fig. 3- Shared Data Storage

- **Parallel Processing**

After master distributes the task over the distributed network all clients will process simultaneously and send acknowledgement to the master server. Master server will also process the task and at the same time will check for the client processing status and monitor it on the screen.

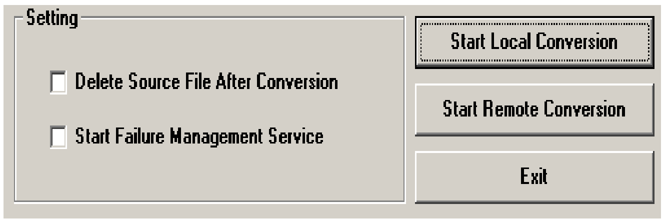


Name	Size	Type	Date Modified
Windows XP Logon Sound.mp3	34 KB	Winamp media file	23/02/2009 2:43 AM
Windows XP Logon Sound.wav	186 KB	Winamp media file	23/08/2001 7:00 AM
Windows XP Menu Command.mp3	1 KB	Winamp media file	23/02/2009 2:43 AM
Windows XP Menu Command.wav	2 KB	Winamp media file	23/08/2001 7:00 AM
Windows XP Minimize.mp3	8 KB	Winamp media file	23/02/2009 2:43 AM
Windows XP Minimize.wav	23 KB	Winamp media file	23/08/2001 7:00 AM
Windows XP Notify.mp3	18 KB	Winamp media file	23/02/2009 2:43 AM
Windows XP Notify.wav	48 KB	Winamp media file	23/08/2001 7:00 AM
Windows XP Pop-up Blocked.mp3	3 KB	Winamp media file	23/02/2009 2:43 AM
Windows XP Pop-up Blocked.wav	29 KB	Winamp media file	17/07/2004 6:08 AM
Windows XP Print complete.mp3	16 KB	Winamp media file	23/02/2009 2:43 AM
Windows XP Print complete.wav	43 KB	Winamp media file	23/08/2001 7:00 AM
Windows XP Recycle.mp3	8 KB	Winamp media file	23/02/2009 2:43 AM
Windows XP Recycle.wav	23 KB	Winamp media file	23/08/2001 7:00 AM
Windows XP Restore.mp3	7 KB	Winamp media file	23/02/2009 2:43 AM
Windows XP Restore.wav	20 KB	Winamp media file	23/08/2001 7:00 AM
Windows XP Ringin.mp3	14 KB	Winamp media file	23/02/2009 2:43 AM
Windows XP Ringin.wav	39 KB	Winamp media file	23/08/2001 7:00 AM
Windows XP Ringout.mp3	8 KB	Winamp media file	23/02/2009 2:43 AM
Windows XP Ringout.wav	22 KB	Winamp media file	23/08/2001 7:00 AM
Windows XP Shutdown.mp3	50 KB	Winamp media file	23/02/2009 2:43 AM
Windows XP Shutdown.wav	276 KB	Winamp media file	23/08/2001 7:00 AM
Windows XP Start.mp3	1 KB	Winamp media file	23/02/2009 2:43 AM
Windows XP Start.wav	3 KB	Winamp media file	23/08/2001 7:00 AM
Windows XP Startup.mp3	76 KB	Winamp media file	23/02/2009 2:43 AM
Windows XP Startup.wav	415 KB	Winamp media file	23/08/2001 7:00 AM

Fig. 4- Data Storage After Conversion

- **Process Failure Detection System**

The system will also manage failure of the client system at run time. Consider a condition if any of the client get failed due to any reason the remaining task should be processed by other system in the network. Master server will take care of this. It will continuously get acknowledgment from client time to time after each task completion. Once any client's connection get closed server will check work remain by specific client and then again divide this task and pass it to other client and client will process it.



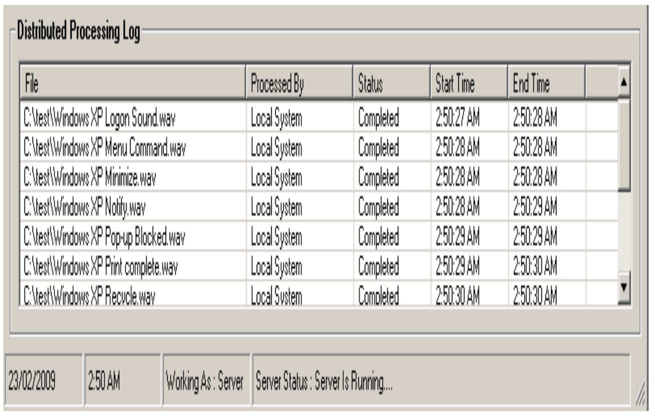
☐ Delete Source File After Conversion

☐ Start Failure Management Service

Fig. 5- Failure Management

- **Distributed File System**

System send data to client for processing but it will increase system overhead. So in this project data to processed will kept on shared storage and accessed using distributed file system so that network protocol processing can be reduced.



File	Processed By	Status	Start Time	End Time
C:\test\Windows XP Logon Sound.wav	Local System	Completed	2:50:27 AM	2:50:28 AM
C:\test\Windows XP Menu Command.wav	Local System	Completed	2:50:28 AM	2:50:28 AM
C:\test\Windows XP Minimize.wav	Local System	Completed	2:50:28 AM	2:50:28 AM
C:\test\Windows XP Notify.wav	Local System	Completed	2:50:28 AM	2:50:29 AM
C:\test\Windows XP Pop-up Blocked.wav	Local System	Completed	2:50:29 AM	2:50:29 AM
C:\test\Windows XP Print complete.wav	Local System	Completed	2:50:29 AM	2:50:30 AM
C:\test\Windows XP Recycle.wav	Local System	Completed	2:50:30 AM	2:50:30 AM

23/02/2009 | 2:50 AM | Working As : Server | Server Status : Server Is Running...

Fig. 6- Distributed Processing Log

First work is to start the server on specific port in listen mode now server is ready to get the request from the client. Now client can make request to the server by providing server address and the server port detail. Client send connect request.

Server will get the connection request same as we get the ring on mobile for connection. After this server can accept or reject the connection and is server accept the request, a connection link get established between server and client this link is called as SOCKET Now server can send the data to the client by using SendData function and client get the data arrival ACK.

After this client can read the data using function GetData Same thing happened with server and communication goes on At last any one of both can close the connection.

Conclusion

The implemented system is best for batch or mass execution.This parallel processing distributed computing Model can reduce overheads and it makes the proper utilization of multiple system rather than implementing supercomputing processor. This system reduces the risk of failure as it can use normal lower configuration PC system to complete the task and even input task is not dependent on the single system. But If not planned properly, a distributed system can decrease the overall reliability of computations if the unavailability of a node can cause disruption of the other nodes.

References

[1] Gupta R., Chaube A.R., Singh S. (2011) *International Journal of Advanced Research in Computer Science and Software Engineering*,.

[2] Wang Y. and Morris R. (1985) *IEEE Trans. Computing*, 34(3), 204-217.

[3] Stone H.S. (1978) *IEEE Trans. Software Eng.*, 4(3).

[4] Stone H.S. (1977) *IEEE Trans of Software Engineering*, 3(1), 95-93.

[5] Miron Livny, Myron Melman (1982) *The Computer Network Performance Symposium*, 47-55.

[6] Hsu C.H. and Liu J.W. (1986) *The 6th International Conference on Distributed Computing Systems*, 216-223.
[7] Derek L. Eager, Edward D. Lazowska, John Zahorjan (1986) *IEEE Transactions on Software Engineering*, 12(5), 662-675.
[8] Eager D.L., Lazowska E.D. and Zahorjan J. (1986) *Performance Evaluation*, 6(1), 53-68.
[9] Chow Y.C. and Kohler W. (1979) *IEEE Transactions on Computers*, 28, 334-361.
[10] Joshi E. *International Journal of Computer Applications,* 1(18), 0975 - 8887.
[11] Deelman E. et al. (2004) *European Across Grids Conference*, 11-20.