Volume 1,Issue-1,December-2011

International Journal of Advanced Research in Computer Science and Software Engineering

A Monthly Scientific Journal of Computer Science

Research paper

Available online at www.ijarcsse.com

An intelligent algorithm for data conversion in distributed computing environment using parallel processing.

Ravindra Gupta[1], Anupam R Chaube[2], Dr Shailendra singh.[3]

,ravindra_p84@rediffmail.com, anupam_chaube@yahoo.co.in ssingh@nittterbpl.ac.in

Abstract

Data conversion is one of the major task in computerization and the data mining process. Thousands of private and the governmental organizations are trying to convert the entire paper document to soft documents. In another example many old format wave files need to be keep safe in very low space and there are thousands of historical wave file need to be convert in to lower size file format. Considering such conditions there is a need to employee the system which will transfer processing over the network system and save output on the server or main system, we are proposing a parallel computing model for the distributed computing platforms. This model ensures easy distribution of the software components, files and resources along the participating computers. We are using a concept in the model which creates slave objects dynamically to fulfill the master/slave parallel computing pattern. When compared with the other similar models results show that our model is not only a feasible model for distributed environment but also an efficient approach of data conversion in distributed parallel computing environment.

INTRODUCTION

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 split up 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 simultaneously 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 heterogeneous environments, network links of varying latencies, and unpredictable failures in the network or the computers. 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.[9]There are many different types of distributed computing systems and many challenges to overcome in successfully designing one. The main goal of a distributed computing system 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.

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 [1] [8] [7] [6] Due to limitations of the solution techniques, simulation is often resorted to for approximate solutions [5] [4]. Some of thesource-initiated DLB algorithms are by Eager [7] [6][5].

Task partitioning is proposed by Deelman et al [10]. 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.

Architecture

Various hardware and software architectures are used for distributed computing. At a lower level, it is necessary to interconnect 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. 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.

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.

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. Leslie Lamport famously quipped that: "A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable."[1]

Troubleshooting and diagnosing problems in a distributed system can also become more difficult, because the analysis may require connecting to remote nodes or inspecting communication between nodes.

Many types of computation are not well suited for distributed environments, typically owing to the amount of network communication or synchronization that would be required between nodes. If bandwidth, latency, or communication requirements are too significant, then the benefits of distributed computing may be negated and the performance may be worse than a non-distributed environment.

Methodology

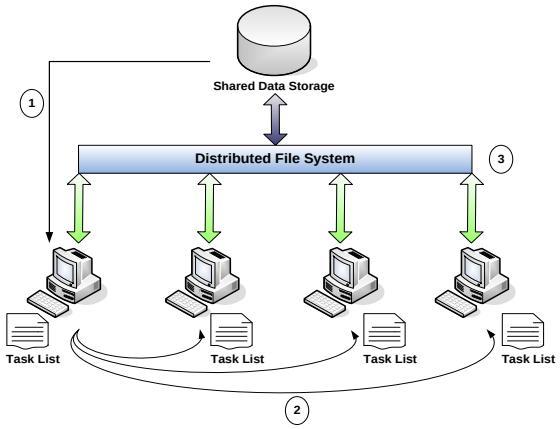

Figure 1 : System Architecture

Figure 1.0 shows the basic block diagram of the project. Actual task processing needs a series of steps to be performed.

Master / Slave System: Parallel processing system built using server / client technology. Where master server act as a process manager system. In proposed 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 proposed 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.

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 proposed system will work on shared data storage it will reduce network processing and network traffic by removing data transfer processing.

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.

Process Failure Detection System: Proposed 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.

Distributed File System: System sends data to client for processing but it will increase system overhead. So in proposed model data to be processed will kept on shared storage and accessed using distributed file system so that network protocol processing can be reduced.

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.

Model Implemented

Step 1.Format Conversion:

- *Task for proposed system is*
  - *Wav files to MP3 format conversion*
  - *Batch file Conversion Simultaneously on available number of client*
  - *Each client will process same number of files*
- *Distribution of files to process*
- *Not distribution of part of files*
- *Using O.S. Provided API Functions*

Step 2. Network Connectivity:

server and client connected to each other using some logical link called as "SOCKET"

- *Using Functions like*
  - *Listen*
  - *Connect*
  - *Accept*
  - *SendData, GetData*

Step 3. Process Distribution:

- *User provides input task on master server Master will analyze the task and divide it.*
- *Create its task list for client.*
- *Clients get the task list to process.*
- *All work on shared data storage.*
- *Reduce network processing and network traffic by removing data transfer processing.*

Step 4. Process Failure Management:

- *Managing failure of the client system at run time.*
- *Continuously get acknowledgment from*

client after each task completion.

- *Once any client's connection get Status Check work retain by client*
- *Again divide this task and pass it to other client and client will process it.*

Conclusion

Implementation of proposed parallel processing distributed computing Model can reduce overheads and it makes the proper utilization of multiple systems rather than implementing supercomputing processor proposed system can use normal lower configuration PC system to complete the task and even input task is not dependent on the single system so it reduces the risk of failure.

Future Scope

As the system is based on master slave terminology we can extend to dynamic role to every system. By the time of failure any client system can become master system and fulfill the user requirement and handle rest of the process which can be called as backup server or backup maser system.

References

[1] Y.Wang and R. Morris, "Load balancing in distributed systems," IEEE Trans. Computing. C-34, no. 3, pp. 204-217, Mar. 1985.

[2] H.S. Stone, "Critical Load Factors in Two-Processor Distributed Systems," IEEE Trans. Software Eng., vol. 4, no. 3, May 1978.

[3] H.S. Stone. "Multiprocessor scheduling with the aid of network flow algorithms". IEEE Trans of Software Engineering, SE-3(1):95--93, January 1977.

[4] Miron Livny, Myron Melman, "Load balancing in homogeneous broadcast distributed systems", Proceedings of the Computer Network Performance Symposium, p.47-55, April 13-14, 1982, College Park,Maryland, United States

[5] C.H.Hsu and J.W.Liu "Dynamic Load Balancing Algorithms in Homogeneous Distributed System," Proceedings of The 6th International Conference on Distributed Computing Systems, May,1986, pp. 216-223.

[6] Derek L. Eager, Edward D. Lazowska , John Zahorjan, "Adaptive load sharing in homogeneous distributed systems", IEEE Transactions on Software Engineering, v.12 n.5, p.662-675, May 1986.

[7] D L Eager, E D Lazowska , J Zahorjan, "A comparison of receiver-initiated and sender-initiated adaptive load sharing", PerformanceEvaluation, v.6 n.1, p.53-68, March 1986.

[8] Y.C. Chow and W. Kohler, "Models for Dynamic Load Balancing in a Heterogeneous Mu1tiiple Processor System," IEEE Transactions on Computers, Vol. C-28, pp. 334-361, May 1979.

[9] E Joshi "Improving Performance of Algorithms in Distributed Computing with Perspective of Green Information Technology"2010 International Journal of Computer Applications (0975 - 8887) Volume 1 – No. 18.

[10]. E. Deelman et al., Pegasus: Mapping scientific workflows onto the grid, In European Across Grids Conference, pp. 11-20, 2004.

