International Journal of Computer Applications (0975 – 8887)
Volume 38– No.8, January 2012

Reliable Clustering Model for Enhancing Processors Throughput in Distributed Computing System

Anurag Raii
Department of IT
College of Engineering Roorkee, Roorkee-247667, (U.K.)

Vikram Kapoor
Department of CS
O.I.M.T. Rishikesh (U.K.)

ABSTRACT

A distributed computing system is the system architecture that makes a collection of heterogeneous computers, workstations, or servers act and behave as a single computing system. In such a computing environment, users can uniformly access and name local or remote resources, and run processes from anywhere in the system, without being aware of which computers their processes are running on. Distributed computing systems have been studied extensively by researchers, and a great many claims and benefits have been made for using such systems. In fact, it is hard to rule out any desirable feature of a computing system that has not been claimed to be offered by a distributed system. However, the current advances in processing and networking technology and software tools make it feasible to achieve the diverse advantages like increased performance, Cost-effectiveness, Sharing of resources, increased extendibility. To meet such challenging computing requirements reliability plays an important role. In this paper authors are proposing a model for enhancing processors throughput in distributed computing system.

General Terms

Distributed Computing systems

Keywords

Distributed System, Task, Communication Cost, Clustering

1. INTRODUCTION

The spread of high-speed broadband networks in developed countries, the continual increase in computing power, and the growth of the Internet have changed the way in which society manages information and information services [1-2]. Geographically distributed resources, such as storage devices, data sources, and supercomputers, are interconnected and can be exploited by users around the world as single, unified resource. To a growing extent, repetitive or resource-intensive IT tasks can be outsourced to service providers, which execute the task and often provide the results at a lower cost. A new paradigm is emerging in which computing is offered as a utility by third parties whereby the user is billed only for consumption. This service-oriented approach from organizations offering a large portfolio of services can be scalable and flexible [3].

The idea of distributing resources within computer networks is not new. It dates back to remote job entry on mainframe computers and the initial use of data entry terminals. This was expanded first with minicomputers, then with personal computers (PCs) and two-tier client-server architecture [4]. While the PC offered more autonomy on the desktop, the trend is moving back to client-server architecture with additional tiers, but now the server is not in-house. Not only improvements in computer component technology but also in communication protocols paved the way for distributed computing. Networks based on Systems Network Architecture (SNA), created by IBM in 1974, and on ITU-T's X.25, approved in March 19761, enabled large-scale public and private data networks [5]. These were gradually replaced by more efficient or less complex protocols, notably TCP/IP. Broadband networks extend the geographical reach of distributed computing, as the client-server relationship can extend across borders and continents. A number of new paradigms and terms related to distribute computing have been introduced, promising to deliver IT as a service [6]. While experts disagree on the precise boundaries between these new computing models,

In general terms, a distributed system is "is a collection of independent computers that appears to its users as a single coherent system" (Andrew S. Tanenbaum). A second description of distributed systems by Leslie Lamport points out the importance of considering aspects such as reliability, fault tolerance and security when going distributed: "You know you have a distributed system when the crash of a computer you have never heard of stops you from getting any work done" [7]. Even without a clear definition for each of the distributed paradigms: clouds and grids have been hailed by some as a trillion dollar business opportunity.

The main goal of a distributed computing system is to connect users and IT resources in a transparent, open, cost-effective, reliable and scalable way. The resources that can be shared in grids, clouds and other distributed computing systems include: Physical resources, Computational power, Storage devices, and Communication capacity [8-10]. Virtual resources, which can be exchanged and are independent from its physical location; like virtual memory, Operating systems, Software and licenses, Tasks and applications and Services.

2. MAIN ASPECTS OF DISTRIBUTED COMPUTER SYSTEM (DCS)

Following are some important aspects of DCS that need special consideration

2.1 Cluster

Clustering is the use of multiple computers, typically PCs or workstations, multiple storage devices, and redundant interconnections, to form what appears to users as a single highly available system. Cluster computing can be used for load balancing as well as for high availability. It is used as a relatively low-cost form of parallel processing machine for scientific and other applications that lend themselves to parallel operations. Computer cluster technology puts clusters of systems together to provide better system reliability and performance. Cluster server systems connect a group of servers together in order to jointly provide processing service for the clients in the network [11]. Cluster operating systems divide the tasks amongst the available servers. Clusters of systems or workstations, on the other hand, connect a group of systems together to jointly share a critically demanding computational task. Theoretically, a cluster operating system should provide seamless optimization in every case [12]. At the present time, cluster server and workstation systems are mostly used in High Availability applications and in scientific applications such as numerical computations.

2.2 Tasks

A task is a sequential program, which performs some predefined action and possibly communicates with other tasks in a system. Some tasks often have priorities relative to other tasks in a system [13-15]. Other common words for tasks are threads and processes. Tasks can be preemptive or non preemptive and are defined to take different states/modes: ready, executing, waiting, blocked or dormant. A task will experience state changes during its execution time [16]. Three blocks usually construct a task: a control block, a program code and a data area. When a task is ready to execute, it is set to active (ready state). The task with the highest priority among the ones ready will then begin its execution. There exist different kinds of tasks depending on what action they implement. Since actions handle events and events have different structures, tasks can be periodic, aperiodic or sporadic. Periodic means that tasks are activated at a repeatedly periodic interval. Aperiodic means that tasks can occur at any time and there are no known arrival patterns between the occasions. Sporadic tasks can also occur at any time but there is a known minimum time between the arrivals [17].

Moreover in a DCS the ability to meet task deadlines largely depends on the underlying task allocation and hence we need a pre-runtime task allocation algorithm that takes into consideration the real-time constraints [18]. Since the end-to-end system response time of distributed applications is affected significantly by inter-task communication, one must account for the effect of delays and precedence constraints imposed by inter-task communication when task allocation decisions are made.

2.3 Tasks Allocation Problem

Consider a set $P = \{p_1, p_2, p_3, ..., p_n\}$ of n processors interconnected by communication links and a set $T = \{t_1, t_2, t_3, ..., t_m\}$ of m executable tasks. The allocation of each task to n available processors such that objective times function is minimized subject to the certain resource limitations and constraints imposed by the application or environment [19]. In a DCS, a program is portioned into small tasks and distributed among several processors to minimize the overall system time. Several challenges have been posed by this mode of processing which can be classified mainly into two broad categories. One class belongs to the hardware oriented issues of building such systems more and more effective while the other class aims at designing efficient algorithms to make the best use of the technology in hand. The task allocation problem in DCS belongs to the later class.

Assigning m tasks to n processors requires n^m exhaustive enumerations. [20] showed that the problem of finding an optimal allocation from amongst all possible assignments is exponentially complex. An efficient task allocation policy should avoid excessive Inter-Processor Communication (IPC) and exploit the specific efficiencies of the processors and in case of a system having similar processors, the tasks or modules should be distributed as evenly as possible. The IPC is the bottleneck in providing linear speed-up with the increase in the number of processors [21].

2.4 Assumptions and Definitions

In our underline model some assumption should be taken for the batter utilization of the resources. Number of task is more than the number of processor's. Total m task are arranged in a list T=[t1,t2,…tm].The size of different task of the list are arranged in the task size matrix TS[]. We have Inter Task Communication Time Matrix ITCTM [,]. It holds the communication time between the Inter Task Communication CTS []. In our distributed computing system we have n processor's, P={p1,p2,…pn} interconnected by communication links, each of the n processor's in the system have their different execution rate (because of heterogeneous system) [22]. The processing efficiency of individual processor is given in the form of PER [] (Process Execution Rate). With the help of ECM algorithm k cluster are created from m task over the n processor's and store it in the list CLS [].

3. PROBLEM STATEMENT

Let us given a distributed computing system consist of a set of n processor's, P=[p1,p2…pn] interconnected by communication links. These links serves the purpose of transferring messages between the processors, and a set of m task T=[t1,t2,..tn] that constitute a communicated program. These constitute a communicated program. These tasks are collectively responsible for attaining the desired goal.

The processing efficiency of individual processor is given in the form of matrix ECM[,] of order m X n and ITCTM[,] is taken in the form of a symmetric matrix CCM[,] or order m X m. The proposed model relies upon:

(i) Developing the methods fro clustering m task.
(ii) Reduction of ITCTM
(iii) Formulating the Cost Function to measure ECM
(iv) Calculating reliability of individual processor, and enhancing the throughput with prefect reliability.

3. THE PROPOSED METHOD

A task is allocated to a processor in such a way that extensive Inter Task Communication is avoided and the capabilities of the processor's suit to the execution requirement of the task. The proposed allocation policy involves clustering of task to the heterogeneous multiprocessor's environment. Initially we concentrate on the task selection for strategy cluster. With the help of ECM algorithm different cluster are created with respect to the number of processor's in the system. Now these clusters are allocated to the different processing units. This allocation is governed by the optimistic allocation strategy. That includes both communication cost and execution cost. The following are the steps for solution (algorithm):

Step1:

(a) Input the size of Processor's Execution Rate PSR(,) matrix and size of Task Matrix TS(,) respectively m and n.

(b) Input size of each Task into TS(,) matrix and Processor's Execution Rate of each processors into PER(,) matrix .

(c) Input inter task communication cost of each task into ITCCM(,) matrix.

Step2: Formation of Cluster of task.

```
(a) Set Cluster :=k
    If k=m then
        Number_of_cluster := n and
        K = $\left\lceil \frac{m}{n} \right\rceil$ task
    End if
(b) $C_i$ =[$t_i$] , each task is like a cluster
    Repeat step 2.b by m times.
(c) Store all cluster in a linear array
    CLS = { $C_i$ , 1<= i <=m}
(d) Select first task pair ($t_r$, $t_s$) : $t_{r \in} C_r$ and $t_{s \in Cs}$
    from Task Size matrix TS(,).
```

Step 3:

```
    If the sum of number of task for cluster Cr and Cs is
    less than or equal
    to $\left\lceil \frac{m}{n} \right\rceil$ then use the cluster $C_i$ with $C_s$
  Else
    (a) Select the next task pair from TS(,) matrix
    (b) Update cluster array CLS ={ } by repeating the
        cluster $C_r \leftarrow C_r \cup C_s$    { $t_r$ , $t_s$ }
    (c) Update the task size matrix TS() by deleting this
        task pair ($t_r$ , $t_s$ ) also
    (d) Update task size matrix TS() and Inter
        communication cost time   matrix ICCTM(,)
    (i)     Update TS(,) by adding s^th row into r^th
    (ii)    Reduce the communication time between
            $T_r$ and $T_s$ to zero
    (iii)   Add the communication time $C_{sj}$ to $C_{rj}$ for
            all j
    (iv)    Delete task Ts from inter task cost time
            matrix ITCTM(,)
    (e) The above procedure is repeated until and
        unless we do not get number if task cluster
        equal to number of processors.
    End if
```

Step 4: Generate execution cost matrix ECM (,)

(a) Find the transpose of $PSR(J)^T$ and multiply with TS(i) as

ECM (I,j) =

Step5: Apply the optimize algorithm to get allocation, and store the assignment

in a linear array $T_{ass}(j)$. Also processor position are stored in a another linear array Allocate (j)

Get the value of Alloc (j) if a task and inter task communication cost from the equation.

Calculate TR, MS and TRP by using the equation

Step 6: We have the execution cost matrix ECM from where we get the best assignment with help of Hungarian methods.

Step 7: Now we have to found the reliability of the system in our assumption let the processors will fail after the 1000 iteration. So we get the failure by ECM/1000.

Step 8: Recursively use step 1 to 9th to calculate reliability.

Step 9: We get this by reliability probabilistic model.

Step 10: Total execution cost EC.

We can get it with the help of adding all the assigned clusters as in step eight.

Step 11: Total Communication Cost.

Step 12: Total Execution reliability [TER], Total Communication Reliability [TCR] and Total Processor Reliability [TPR] is calculated by multiplying all the assigned reliability terms.

4. RESULTS AND DISCUSSIONS

The present paper deals with a simple yet efficient computational algorithm for reliable clustering of task. Developed algorithm results the overall reliability of the **DCS.** A simple procedure has been developed to determine the following

i) Best Clustering Domain.

ii) Throughput of the processors.

iii) Total Execution Reliability.

iv) Total Communication Reliability.

v) Overall System Reliability.

Figure 1 shows the reliability of Distributed Computer System with the set of seven task and three processor having execution rate 0.248, 0.218 and 0.234 respectively with the help of RECM algorithm we calculate the reliability of individual processor as 0.99411, 0.99433 and 0.99462 which is far batter in comparison to the non clustering model of Sig05 also the throughput of the existing system will enhance during the high load of execution.

5. CONCLUSIONS

It is conclude that algorithm is general and can accommodate a large number of task to be clustered on any number of processor's to check the generality of our algorithm several sets of input data are considered and is found that the algorithm is suitable for arbitrary number of processor's with the random program structure and workable in all the cases.

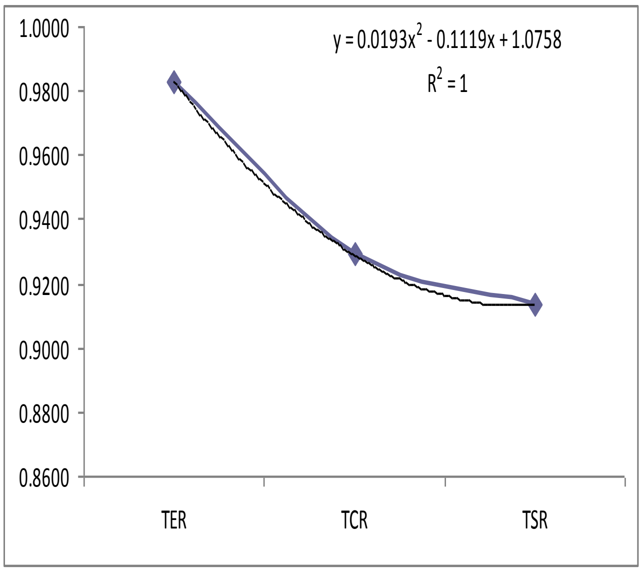

Figure 1: Reliability of distributed computer system

6. REFERENCES

[1] A. Tom P. and Ram Murthy, C. S. 1997. An improved algorithm for module allocation in distributed computing Systems. Journal of Parallel and Distributed Computing Systems, Vol. 42, pp. 82-90.

[2] Kafil, M. and Ahmad, I. 1997. Optimal task assignment in heterogeneous computing systems, In Proceeding of Sixth Heterogeneous Computing Workshop, pp. 135-146.

[3] Peng, D. T., Shin, K.G. and Abdel, Zoher, T.F. 1997. Assignment scheduling communication periodic tasks in distributed real time system. IEEE Transactions on Software Engineering, SE-13, pp. 745- 757.

[4] Chu, W.W. Holloway, L.J., Lan, M.T., and Kfe, K. 1980. Task allocation in distributed data processing. IEEE Concurrency, pp.57-69.

[5] Richard, P.Y., Edward, M., Lee, Y.S., and Tsuchiya, M. 1982. A task allocation model for distributed computing systems. IEEE Transactions on Computers, Vol.C-31, pp. 41- 46.

[6] Shen, C. C., and Tasi, W.H. 1985. A graph matching approach to optimal task assignment in distributed computing systems using a minimax criterion. IEEE Transactions on Computers, Vol. C- 34, pp. 197-203.

[7] Stone, H.S. 1978. Critical load factors in two- processor distributed system. IEEE Transactions on Software Engrg. Vol. 4, pp. 254- 258.

[8] Muhammad, I.A., Dhodhi, K., and Ghafoor, A. 1995. Task assignment in distributed computing systems. IEEE Concurrency, pp.49-53.

[9] Lee, C.H., Lee, D. and Kim, M. 1997. Optimal task assignment in linear array networks. IEEE Transactions on Computers, Vol.41, No. 7, pp.877-880.

[10] Shatz, S.L., Wang, J.P., and Goto, M. 1992. Task allocation for maximizing reliability of distributed computer systems. IEEE Transactions on Computers, Vol.41, 9, pp.

[11] Kartik, S., and Ram Murthy, C.S. 1997. Task allocation algorithms for maximizing reliability of distributed computing system. IEEE Transactions on computers, Vol.46, No. 6, pp. 719-724.

[12] Chen, D.J., Chen, R.S., Hol, W.C., Ku, K.L. 1995. A heuristic algorithm for the reliability- oriented file assignment in a distributed computing system. Computers Math. Applic., Vol. 29, No.10, pp. 85- 104.

[13] Yin, P.Y., Yu, S.S., Wang, P.P., Wang, Y.T. 2007. Task allocation for maximizing reliability of a distributed system using hybrid particle swarm optimization. The Journal of Systems and Software, Vol. 80, pp. 724-735.

[14] Srinivasan, S., and Jha, N.K. 1999. Safety and reliability driven task allocation in distributed systems. IEEE transactions on Parallel and Distributed Systems, Vol.10. No. 3, pp.238-251.

[15] Vidayarthi, D.P., and Tripathi, A.K. 2001. Maximizing reliability of distributed computing system with task allocation using simple genetic algorithm. Journal of System Architecture, Vol. 47. pp. 549-559.

[16] Kng, Q.M., He, H., Song, H. M., Deng, R. 2010. Task allocation for maximizing reliability of distributed computing system using honeybee mating optimization. The Journal of Systems and software, Vol.83, No. 2.pp.

[17] Woo, S.H., Yang, S. B., Kim, S.D., and Han, T.D. 1997. Task scheduling in distributed computing systems with a genetic algorithm. Doi.0- 8186- 7901- 8/97 10.000, IEEE p.p. 301-305.

[18] Lu, H. 1996. Load balanced task allocation in locally distributed computer sciences. Technical report# 633.

[19] Elsadek, A.A., and Wells, B. E. 1999. A heuristic model for task allocation in heterogeneous distributed computing systems. International journal of computers and there applications, Vol.6, No.1, March 1999. pp. 1-35.

[20] Lo, V.M. 1988. Heuristic algorithms for task assignment in distributed systems. IEEE Transactions on computers, Vol.37. No. 11, pp. 1384- 1397.

[21] Kfe, K. 1982. Heuristic models of task assignment scheduling in distributed systems. Computer, Vol. 15, pp. 50- 56.

[22] Ellis, H., Sahni, S. and Rajsekaram, S. 2005. Fundamentals of computers algorithm. Galgotiya publication Pvt Ltd.