International Journal of Computer Applications (0975 – 8887)
Volume 44– No.23, April 2012

Efficient Clustering Model for Utilization of Processor's Capacity in Distributed Computing System

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

Vikram Kapoor
Department of CS
Omkarananda Institute of Management & Technology
Rishikesh (U.K.)

ABSTRACT
Distributed Computing System (DCS) computing plays an important role in computing world where processing load is distributed for computational efficiencies. DCS are designed to facilitate the sharing of resources as well as to reduce communication costs, increase throughput, and decrease delay of services. DCS are motivated by the need for cost reduction in tasks execution. In such applications the quality of the output is proportional to the amount of real-time computations. To meet such challenging computing requirements at electrifying speeds efficient clustering strategies are required for proper utilization of Distributed System. In this paper, we proposed a model for efficient utilization of the processing units in distributed environment and calculated the through of individual processor as 0.339789, 0.371609 and 0.529287 which is far better in comparison to the non clustering model of Sig05.

General Terms
Distributed Computing systems

Keywords
Distributed system, Communication cost, clustering

1. INTRODUCTION
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 [1-4]. 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. Using workstations clusters for distributed computing has become popular with the increase of inexpensive, powerful workstations. Workstation clusters offer both a cost effective alternative to batch processing and an easy entry into parallel computing. Recently, there has been much interest in using inexpensive, powerful workstations to form workstation clusters. Workstation clusters offer many benefits over traditional central site computing. High performance workstation clusters can off-load jobs from saturated vector supercomputers, often providing comparable turn around time at a fraction of the cost [5]. If workstations within clusters have a high speed interconnect, they may also serve as an inexpensive parallel computer.

Cluster plays a vital role in the development of high speed computing environment. With the help of clustering one can get logical independence of hardware and software resources [6]. Clustering provides the real globalization of computing era. Process can execute on any cluster according to their requirements. Cluster has a large number of computing advantages that will make them contemporary computing technique. Some of them are:

Heterogeneous Processing Support
There are two types of cluster environments, homogeneous and heterogeneous. A homogeneous computing environment consists of a number of computers of the same architecture running the same operating system [7]. A heterogeneous computing environment consists of a number of computers with dissimilar architectures and different operating systems [8]. Many locations have a large number of different computers for varying resource and performance considerations.

Batch Processing Support
A popular use of clusters is off-loading batch jobs from saturated supercomputers [9] and to distribute it on different computers. Clusters can often provide better turn around time than supercomputers for small (in terms of memory and CPU requirements) batch jobs.

Parallel Processing Support
There is interest in moving to massively parallel processing machines via heterogeneous processing because of the heterogeneous environment's application to a larger set of problems. A cluster can serve as a parallel machine because workstations are inexpensive and easier to upgrade as separate pieces may be purchased to replace older models [10].

Interactive Execution Support
A cluster should provide users the option to execute interactive jobs on the cluster. The input, output, and error messages should all be optionally returned to the user's interactive machine.

Message Passing Support
Message passing is the ability to pass data between processes in a standardized method. This inter-process communication allows several processes to work on a single problem in parallel [11]. A large distributed application can be split across the many different platforms that exist in a heterogeneous environment.

Load Balancing Support
Load balancing refers to the distribution of the computational workload across a cluster so that each workstation in the cluster is doing an equivalent amount of work. On a network, some machines may be idle while others are struggling to process their workload.

Distributed computing systems have been studied extensively by researchers, and a great many claims and benefits have been made for using such systems [12]. 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 [13]. However, the current advances in processing and networking technology and software tools make it feasible to achieve the advantages such as increased performance, sharing of resources, increased extendibility, Increased reliability, availability, and fault tolerance, and cost-effectiveness. With the use of distributed computing is increased performance. The existence of multiple computers in a distributed system allows applications to be processed in parallel and thus improves application and system performance [14].

The above mentioned advantages cannot be achieved easily because designing a general purpose distributed computing system is several orders of magnitude more difficult than designing centralized computing systems[15]. Designing a reliable general-purpose distributed system involves a large number of options and decisions such as the physical system configuration, communication network and computing platform characteristics, task scheduling and resource allocation policies and mechanisms, consistency control, concurrency control, and security, to name just a few. The difficulties can be attributed to many factors related to the lack of maturity in the distributed computing field, the asynchronous and independent behavior of the systems, and the geographic dispersion of the system resources [16].

The use of a communication network to interconnect the computers introduces another level of complexity. Distributed system designers not only have to master the design of the computing systems and system software and services, but also have to master the design of reliable communication networks, how to achieve synchronization and consistency, and how to handle faults in a system composed of geographically dispersed heterogeneous computers [17]. The number of resources involved in a system can vary from a few to hundreds, thousands, or even hundreds of thousands of computing and storage resources.

Despite these difficulties, there has been limited success in designing special-purpose distributed systems such as banking systems, online transaction systems, and point-of-sale systems. However, the design of a general purpose reliable distributed system that has the advantages of both centralized systems and networked systems.

2. MAIN ASPECTS OF DISTRIBUTED COMPUTER SYSTEM (DCS)

Following are some important aspects of DCS that need special consideration

2.1 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 [18]. 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 [19]. 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 [20].

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 [21]. 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.

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_n} of m executable tasks. The allocation of each task to n available processors such that an objective time functions is minimized subject to the certain resource limitations and constraints imposed by the application or environment. 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 nm exhaustive enumerations [22] 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 [21-22].

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= [t_1, t_2,…t_m].

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= {p_1,p_2,…p_n} interconnected by communication links, each of the n processor's in the system have their different execution rate (because of heterogeneous system). 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 [ ].

5. 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) Making utilization of each processor, and developing an algorithm for allocation of processor to the respective cluster to obtain minimum Cost.

6. 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.

Computational Algorithm

```
Declaration Section-

TASK_CLUSTER():  // form the clusters of tasks.
GENERATE_ECM(,):  // by multiplying the TS(,) and PSR(,)
TASK_MAPPING():  //Map the task clusters to the processors
using the Optimal Allocation Algorithm

End Declaration Section

Input:  m, n, TS (), PSR () and ITCCM (,)

TASK_CLUSTER()

Start Procedure

Begin
        Set: task cluster ←K
If K=m
        then
                No of clusters to be form =n and a cluster
may not contain more than K= [m/n] task
        endif

Initially each task is treated like a cluster $C_i=\{t_i\}$for i=1 to m.
Store these clusters in a linear array CLS={Ci, 1<=i <=m}.

Select the first tasks pair say (tr,ts) (say tr ε Cr and ts ε Cs )
from TS().

If the sum of number of tasks for clusters  Cr and Cs is less than
or equal to [m/n],
then
        fuse the clusters Cr with Cs
else

        select the next task pair from TS().
        Modify CLS={} by replacing the cluster Cr as Cr ←
Cr υ Cs={tr,rs} .and.
        Modify the TS() by deleting this tasks pair (tr,ts) also
        Modify TS() and ITCTM(,) as:
                (a): Modify the TS(,) by adding s-th row into
        r-th.
                (b): Reduce the communication time
        between tr and ts to zero.
                (c): Add the communication time Csj to Crj
        for all j.
                (d): Delete task ts from ITCTM (,).
        The above procedure is repeated until and unless we
        do not get number get number of tasks clusters equal
        to number of processors.
End
End Procedure TASK_CLUSTER

GENERATE_ECM(,)

Start Procedure

Begin

To determine the ECM(,) Initially we have taken the transpose
of the PSR(j)T and multiply with  TS(i) as:

                t1   Ts1        P1    p2    p3    .   .   pn
                t2   Ts2
ECM(I,j)=       t3   Ts3   X    Er1   Er2   er3   .   .   ern
                .    .
                .    .
                tm   tsm

End

End Procedure GENERATE_ECM

TASK_MAPPING()

Start Procedure

Begin

Apply the optimistic Algorithm to get the allocation.
Store the assignment in an linear array Tass(j)
(where j=1,2,….n).
Processor position are stored in a another linear array
Aalloc(j).
Get the value of TTASK (j) by adding the values of Aalloc(j) if a
task is assigned to a processor otherwise continue.

Evaluate the processor's wise Execution cost and Inter Task
Communication Cost.
```

Calculate RT, MSR and TRP

End

End Procedure TASK_MAPPING

7. RESULTS AND DISCUSSIONS

The present paper deals with a simple yet efficient mathematical and computational algorithm for clustering of task for evaluation of performance of the DCS. A simple procedure has been developed to determine the following

i) Clustering of task.
ii) Processing rate of different CPU's.
iii) Communication delay.
iv)Throughput of the processors.

Figure 1 shows the throughput of DCS with the set of seven tasks and three processor having the processing cost, throughput and service rate as according to the Table 1.
With the help of ECM algorithm we calculate the throughput of individual processor as 0.339789, 0.371609 and 0.529287 which is far better in comparison to the non clustering models. Overall cost of the existing system is reduced which makes it efficient. Also the service rate of the existing system will constant during the high load of execution.

Table 1

Processor	P_Cost	Throughput	Service Rate
p1	0.047	0.339789331	0.169894665
p2	0.051	0.371609067	0.185804534
p3	0.044	0.529287227	0.176429076

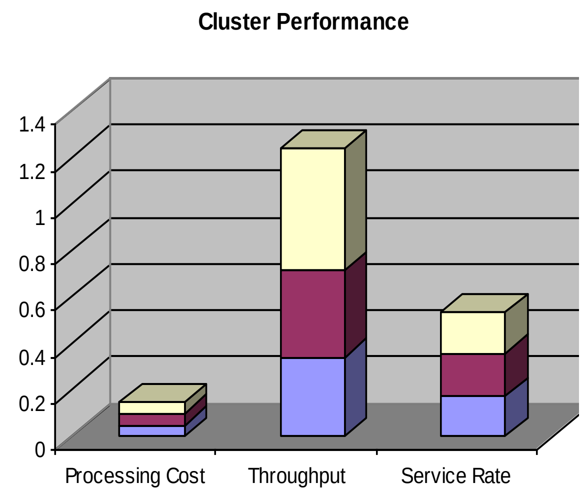

Figure 1: Process Cost ,Throughput and Service Rate of Efficient Clustering System

8. CONCLUSIONS

In this paper, we presented computational algorithm for clustering of task for evaluation of performance of the DCS. The 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.

9. 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.

[23] Tushar Deepak Chandra, Vassos Hadzilacos , Sam Toueg 1996. The weakest failure detector for solving consensus. In the Journal of ACM (JACM), Volume 43 Issue 4

[24] Florina M. Ciorba, Timothy Hansen, Srishti Srivastava, Ioana Banicescu, Anthony A. Maciejewski, and Howard Jay Siegel, "A Combined Dual-stage Framework for Robust Scheduling of Scientific Applications in Heterogeneous Environments with Uncertain Availability," *21st Heterogeneity in Computing Workshop (HCW 2012)*.

[25] Mourad Elhadef and Amiya Nayak, Comparison-Based System-Level Fault Diagnosis: A Neural Network Approach IEEE transactions on parallel and distributed systems, vol. 23, no. 6, june 2012

[26] Jay Smith, Edwin K. P. Chong, Anthony A.Maciejewski, and Howard Jay Siegel, "Overlay Network Resource Allocation Using a Decentralized Market-Based Approach," *Future Generation Computer Systems*, Vol. 28, No. 1,pp. 24-35, Jan. 2012.