The optimize load balancing in cluster computing.

.

Mr. Sunil Kumar Pandey, Prof. Rajesh Tiwari.

Computer Science and Engineering Department, shri sankaracharya college Of Engineering & Technology Bhilai. Csvtu Bhilai

ABSTRACT: **In this paper we introduce the efficient load balancing algorithm which is helpful to distribute the packet of data to the group of slave processor for achieving the highest possible execution speed and proper utilization of system processor. We propose the load balancing algorithm in the cluster environment means group of slave processor make the cluster or communicator and master processor distribute work to the particular communicator . communicator have unique id which give identification to this. cluster computing provide various computing resource to facilitate execution of large scale task so select the proper node for execute a task able to enhance the performance of large scale cluster computing environment .The performance of the processor is calculated by the system throughput and response time . In our proposed algorithm one communicator do one type of work .It strike a good balancing throughput and output order of data unit and process Synchronization. this type of computing is very important in Engineering and scientific field where different different job come in the main processor.**

Index : **communicator , distributed system , tree bisection , message passing routine , task partition real time scheduling.**

I. INTRODUCTION

Load balancing is a technique to spread work .between many computer process , disks or other resource in order to get optimal resource utilization and decrease computing time. load balancer can be used to increase the capacity of server beyond that a single server. Our modal is master slave communication message passing routine master processor divide the problem domain in the fix number of processor that are to be executed in parallel there is some fix slave processor make the group and form communicator . communicator run parallel y and execute the program . they have certain unique id which is use to differentiate between them . It can be possible that some processor complete their task earlier than other and ideal for some time due to difference of capacity handle of processor .load balancing is

useful where the amount of work is not known prior to execution . there are two category of load balancing proper load balancing and improper load balancing . in the improper load balancing we obtain the non linier structure means they cannot finish their work in same time . their head of contact line is zigzag up and down if we take the graph between load and execution time. perfect load balancing oppose to improper load balancing where line of contact head of processor is smoothly and perpendicular of time axis. Perfect load balancing is sometimes rare so we try to obtain the graph just smoothly. Cluster computing load balancing has been an active research area and therefore many different assumption and terminology are independent suggested .the amount of time the communicator spend to finish or complete the work called workload time of the communicator. The master_ slave communication modal for load balancing used both in centralised and distributed platform. Load could be storage , bandwidth etc. many researcher work on load balancing for many year target to obtain the load balance scheame with overhead as low as possible. We purpose the algorithm which work in dynamic environment to use the dynamic approach . in this algorithm we will have implement multiple job to multiple communicator. The mapping between communicator and job is one to one at the same time in starting but if any communicator finish their job earliar then it can couple to another communicator dynamically .in this way response time reduce and performance increases . our algorithm is FCPDA (fully centralised and partially distributed algorithm).

II. SYSTEM OVERVIEW

In this section we outline basic platform and system architecture where our algorithm work properly without created any problem and complexity. We have already discuss in the introduction part that that our modal work for cluster computing environment .cluster is same as communicator which is group of processor. How the processor will group depend upon master processor. It is 3 category (1) cost of processing base , (2) response time base,(3) distance of processor base. All information about this statically known by the master processor. Master take decision what type it grouping the processor. we use the message passing interface routine to solve problem. There are 3 way by which message passing routine occur. (1)designing the special purpose programming (2) extending the syntax and reserved words of an existing sequential high level language to handle message passing.(3) using an existing sequential high level and providing a library of external procedure for message passing. OCCAM is common message passing parallel programming which was designed to be used the unique message passing processor called transputer and also translate to serial code to parallel code . message passing library calls that platform. where direct process to process communication through IPC. In this view it is necessary to say explicit what process are to be executed when to pass message between concurrent process. In the form of message passing system we have to need

1. A method of creation separate process for execution for different computer .
2. A function of sending and receiving message.

Message passing environment occur in two modal.

1. **Mpmd modal :** it is most general programming modal in which completely separate and different program is written for each processor. however it is normally to sufficient to have just two different program . one processor execute the master program and another ideal processor execute slave programs. slave and master have unique process id may be used to customize the execution for example to specify the destination of generated message.
2. **Spmd modal:** In this modal the different program integrated into one program unit. within the program are control statement that will select different parts for each process .after the source program is constructed with the required control statement to separate the action of each processor ,the program is compiled into executable code for each processor. if different type of processor exist then the source code has to be compiled into each processor type, and correct type must be loaded for execution by each processor .

   for dynamic process creation two distinct program may written , a master program and a slave program separately compiled and ready to execution. our algorithm work in the both MPMD and SPMD modal. It works in centralise dynamic load balancing. because here cluster directly contact to master processor.

III. RELATED WORK AND ALGORITHM

1. **Line structure algorithm**: line structure algorithm is suggested by Wilson(1995) which is particular implemented to processor connect it in the line structure , but the technique could be extended for other various interconnection structure as mess network , Ring network , Hub network etc. line structure means pipeline structure . we consider the method here to show the possibility of a specific interconnection network . The basic idea to create a queue of tasks with individual processor accusing location in queue. The master processor feeds the task at one end and task are shifted down to queue. When slave processor Pi detect a task at its input from the Queue and process is idle it take the task from the queue. then the tasks to left shuffle down the queue so space held by task is filled . in this structure it is possible that one processor also depend to the another processor. But in this way all processor are busy in all time and system utilization increase. Queue is designed for priority of task base means high priority process come.
2. **Central queue algorithm**: It is dynamic data distribution algorithm. . where the main processor maintain the queue where new job is stored. When the new task come then the queue manager distribute it's task to the requester slave processor. if no job is ready then the request is buffered until new activity occur. When the load fall under certain limit of processor then local load manager in the slave processor send request to the master processor which is further handle by central manager.
3. **Adaptive load sharing algorithm** : No. Of load sharing algorithm exist but due to complexity and overhead we unaware the real implementation of this. this algorithm minimize out of order rate as well as reduce the probability of units associated to the same flow being treated by different processor . this algorithm use i parallel computer as well as broadcast network. According to this algorithm data unit mapped to any server according to function.

   F(x) = y. Which is defined as

   Pj G(x,y) = max (G(x,k)) where k is processor node 1< k<= P.

   Where x is identification vector of unit use to identify a certain flow the unit belong , j is the server node to which data will be mapped . G(x,y) is psedofunction.
4. **Global graph partition algorithm** : It is the dynamic load balancing algorithm which is very popular and powerful . this algorithm use the fact that a problem is a big domain . the master processor divide the domain into sub domain. Object are as a graph vertices and interconnection between object is denoted by edge. E . suppose master processor divide the task or domain in k way means the domain is divided D_1 , D_2 , D_3, D_4............$D_{k.}$ all sub domain are independent to each other means D1∩D2∩D3∩D4.................................∩Dk =Φ and when we integrate these sub domain then we obtain the exact domain . the dividing the problem in optimise way are NP complete problem so heuristic strategy used.
5. **Local queue algorithm** : This algorithm support dynamic approach or migration .main aim of this algorithm is statically allocation of all new process migration by the main processor when load fall down some certain limit. The parameter defined the fix number of ready task the master processor attempts to distribute all slave processor . when processor have load below the threshold load then local load manager target to get many task to remotely processor. It randomly send request with number local task to remote load manager , after this load manager compare to local id of ready process to received id if later is smaller than former then some task which are running are pre-empted to requester and affective configure with no. Of task transfer is returned .

6. **Receiver Initiated load balancing** : In this load balancing scheme it is possible for a task to migrate multiple times until the proper load balancing not occur . here slave can also be communicated between them .so light weighted slave processor broadcast to its neighbour slave processor . after receiving the data each neighbour slave processor compare it's local queue length to requested queue if it is smaller than the its local then neighbour processor replies to single job.

IV. METHODOLOGY

we have already introduce that our algorithm work in cluster computing environment . It adopt the modal of message passing interface master slave architecture . It is applicable to multiple job multi cluster platform. . at first the task manager request to the master processor to know their status. If master processor ready to receive the task then it response to task manager as well as flag status is equal to 1 otherwise send 0. If task manager obtain status =1 then it send the data packet to the master processor master processor have limited number of communicator so that it could be handle without any complexity .master processor have some queue which is equal length and every queue length is equal to the page size of main memory of the task manager and the number of queue equal to number of communicator around master processor plus 3.if the task manager supply task is less than or equal to master communicator queue then they are stored in queue . if the size of packet more than queue length then master processor response to task manager and to fragment this data packet less than or equal to page size of memory. then after task manager send task to master. If the flag status bit is equal to 0 means master processor not ready to receive the packet i this condition task manager check permanently to know the status of the master processor. master processor set the counter cont which is initially equal to no. Of communicator when one type of job come then it is store in the queue then counter decrement by one this loop continuously executed until counter equal to zero then if extra job come then it is rejected by the master processor. when all the task store in the respective queue then master processor started it's operation the queue where the job store called communicator queue ,there is second queue which is smaller than communicator queue is called Task queue where p slot exist p is the number of communicator exist in processing every slot have fix number of sample data of every queue. The sample task of first slot broadcast to all slave processor processor each slave processor computer the job and send to the master processor which is store in response queue Q[k]. Response queue is have P^2 slot how many time the data spend in particular operation or job store in these queue. Same procedure apply for another job so in this way a 2 dimensional matrix form between number of communicator and number of different different tasks. In the dynamic approach the master obtain the time for execution to proceed different different task to different different communicator. So according these extra information the master processor apply the proposed algorithm . In this algorithm one communicator do one type of work in same time .if one communicator have done it's work early then it may couple to another work and they do their work simultaneously if both of them complete it's work then they couple with third processor in this load balancing occur. in this way proper load balancing occur and processor will always busy so proper response time decrease and utilization and throughput increase. This algorithm use some property of line structure of dynamic load balancing strategy.

V. PROPOSED ALGORITHM

```
1. The task manager send the request $r_m$ to the master processor.
2. If ( status=1 and communicator queue Q[i] = empty and busy = 0 or 1 )
   {
   Master processor ready to receive the packet $D_i$ . where $D_i$. Is type of job allotted to communicator queue i.
   }
   Else
   {
   {
   master processor send the response status =0 to task manager means it can not receive the data packet
   }
3. If step 2 true then task manager send the task Di to the master processor .
4. If (size of data packet > size of communicator queue Q[i]
   {
   Master reject that packet and response fragment bit flag =1 , Q[i] =x means this packet is larger than the size of communicator queue and send it's queue size to this .
   }
   Else
   {
   Packet store in the communicator queue Q[i].
   }
5. master processor set the counter cont which is initialised to P where P is number of communicator around it.
6. While(recievetask ==true )
   {
   Counter = counter -1;
   }until counter =0;
7. If (counter ==0)
   {
   All communicator queue have been filled .
   }
8. If ( counter ==0 and receivetask == 1)
   {
   Packet are rejected by master processor and suggest to wait some time.
   }
9. If (step 7 ==true)
   {
   Copy data of every communicator queue to distributed queue T[i] , i= P in respective slot S[i]. Where i =1 to P(number of communicator).
```

```
    }
10. Counter = P;
    While( counter<>0)
    {
    Distributed queue broadcast slot S[i] of sample data to all processor.
    Counter =counter – 1;
    }
    If( counter ==0 )
    {
    All slot of T[i] to all processor and wait for some times for response
    }
11. Every processor Pi response to master processor with correct result .
12. Master processor again set the counter and initialised to 0 .
    While(R[i] == true )// R[i] means result obtain i ‘th number of communicator to master processor.
    {
    Counter = counter + 1;
    $(\text{Absolute Time})_i$ = (Total time spent for processing the task between master and communicator – propagation delay for task)$_{i..}$ ;
    Store the time in response queue [j][i];
    }
13. Repeat step 12 for other type of task until i = total number of communicator.
14. In this way 2 dimensional matrix a[i][j] formed between number of task to number of job .matrix is is N to N .means N job and N communicator.
15. While (N> 0)
    {
    For (i=0;i≤N ;i++)
    {
    For(j=0; j≤N ; j++)
    {
    Float $L = (a_{ij} + a_{i(j+1)} + a_{i(J+3)} + a_{i(J+4)}) + a_{i(j+5)}.........a_{i(i+n)/}N$;
    Minus L to every element in the particular row i.
    }
    }
    N=N-1;
    }
16. Repeat step 15 until at least one of element in every row not equal to zero.
17. If (step 16 == success)
    {
    Make the two group G[i] where 1≤ i≤ N of every row element one for positive and zero and one for negative element.
    }
    // selection lowest time in particular row//

18. For ( i=1; i<= N ; i++)
    {
    For (j= 1; j<= N ; j++)
    {
    Search the very large negative element a[i][j] in $G_{i2.}$
    Means negative group of element of row i.
    If ( i= 1)
    {
    Select the highest negative element a[i][j].
    }
    Else
    {
    Check whether a[i[j] is equal to a[m][j] .
    If (true)
    {
    Search these second highest element in $G_{i2.}$ Until a[i][j] is not equal to a[m][j].
    }
    }
    }
19. In this way the master have known which job it should give to which communicator to apply this algorithm.
20. Sort these element .
21. First whole job given to that processor which is lowest element after sorting .then 50 % task given to second lowest element 25 % task given to third lowest processor in this way data is distributed.
22. A communicator which give lowest response time to master processor and get ideal so master distribute it the remaining job of second processor. After complete they response to master then both of them obtain 50 % task of remaining task of third communicator which they divide 25 – 25% and finish. In this way processor complete their task and busy all time until whole job would not complete.
```

VI. RESULT AND ANALYSIS

Suppose that there are 3 communicator and 3 type of job come in the master processor and store in respective queue Q[1],Q[2],Q[3]. type of job are sorting and obtain adding of n element and counting the number. Suppose 5 sample data of every job are broadcast to all of processor included master processor how much time a communicator take is given below.

A[3][3]= [{3,6,5},{4,7,1},{5,8,9}]

Means first communicator take 3 unit, 6 unit and 5 unit of time to solve the respective problem. In the same way communicator 2 finish the 3 task in 4,7,1 unit time respectively and communicator 3 takes 5, 8 ,9 time unit to complete these task. We take the average of first row obtain (14/3) is equal to 4.66 we minus this to every element in the row obtain ={ -1.66 , 1.33,0.67} we divide these element into two parts G11={-1.66} and G12={1.33,0.67}. Same operation for row 2 element average of these element= (12/3)=4 .so we minus 4 into every element of row 2 element obtain A{2,3}={0,3,-3} we put these element in to two group G21={-3},G22={0,3}.and at last same operation for row 3 and average the element = (22/3)=7.33.now subtract this element to every element in row 3 obtain {-2.33,0.67,1.67} we take two group G31 ={-2.33},G32={0.67 ,1.67}.now more negative element in row 1 is {-1.66} so select this a11 means first number of job sorting allot to communicator or cluster 1.same more negative element in row 2 is which is {-3} which is first element in row 1 means job 1 distribute to job but it is already decide to cluster 1 so we search the minimum element in G21 which is 0 so second job allot to cluster 2.and last one job remaining which is

distribute to cluster 3. We take these selected element and sort and obtain {0,-1.67 ,-3} to decide the optimize distribution here we send the 100 % job to third cluster then we send 50 % job to second cluster and 25 % task given to first cluster. suppose that first job is sorting of 100 data second job is add the 100 number and third job is obtain the count of 100 number. We know that the time complexity of sorting is o(n^2) in worst case , time complexity of adding the number is 0(n^2) and counting the number is 0(n). So in the example the algorithm decide that first communicator obtain the first job which is sorting of 100 number then the time to spent to sort 100 number by communicator = (time unit spent to processing the particular job time complexty of operation of sample data) time compexty of operation of given data.
So time = (3 divide 25)multiply 10000 = 1200 time unit. Same way the time spent by communicator 2 to complete the task the third task is (1÷5)× ×100 = 20 time unit .same way the task 2 completed by communicator is = (8÷25)× 10000 = 3200 time unit. we sort these time unit and obtain = {3200 ,1200 ,20} so whole job 3 allot to communicator 2. Then 50 data of first job provide to cluster 1. And 50 data of job 2 provided to cluster 3.
When cluster 1 complete it's task then remaining two processor are busy. After complete the third job cluster 2 combine with first cluster and both complete the first job in (3÷25) ×2500 = 300 time unit and communicator 2 complete the same work with (4÷25) ×2500 = 400 time unit means 420 time unit from initially. Then after they combine with cluster 3 and complete the job 2 and divide remain 50 data between them . complete 25 sort data by cluster 1 = (6÷25) ×625 = 150 and same data for cluster 2 = (7÷25) × 625= 175 . complete of all job in max(420+150,420+175,800) = 800 time unit . if we not use algorithm then we obtain 3200 time unit and lot of cluster processor ideal after they complete their job.

VII. CONCLUSION

The aim of this paper to show how we can load balance in multiple task multiple cluster environment. This algorithm solve these type of problem perfectly . It works in dynamic environment so master know the exact information about the processing capacity of communicator or cluster priory to use this algorithm . master also take decision that which job it should to provide to which cluster and also what is the order of data distribution to tasks. We design our algorithm in that platform when number of cluster and number of job are equal. if extra job come beyond the capacity of master processor then they are rejected but This problem can also be handle in two way first we have to increase the number of communicator queue but increase the number of queue incurs the cost of processor as well as can reduce the performance due to increase the response time and throughput. Second way is that we combine the Round Robin algorithm with our propose algorithm and if extra job come then Round Robin algorithm applies. However our algorithm works in very much in the real time application where deadline of job are already specified we have to complete the lot of job in fix time unit.

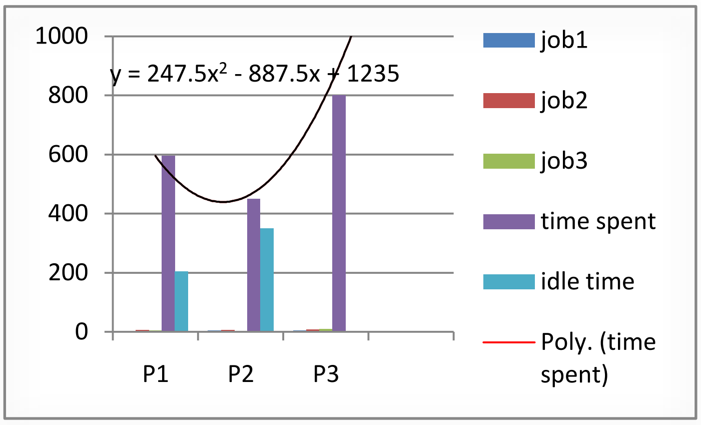

Figure 1. Optimize load balance in cluster computing

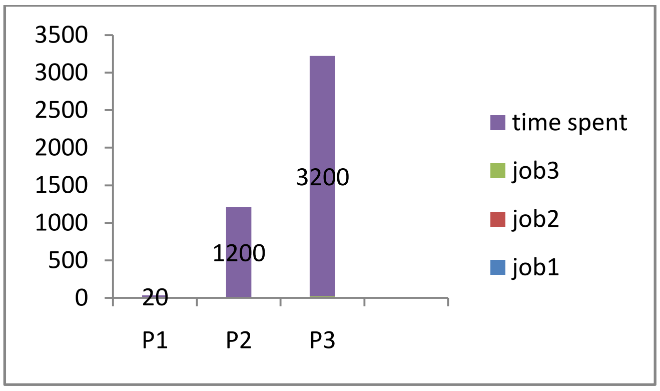

Figure 2. Imperfect load balance in cluster system.

VIII. ACKNOLEDGEMENT

Working on this research paper has been great learning experience for me. It will be my pleasure to acknowledgement , utmost cooperation & valuable suggestion from time to time given by other staff members of Department ,& also I would like to thank all person who have directly or indirectly helped me in the making the paper.

My greatest gratitude is reserved for my Project guide Prof. Rajesh Tiwari . He will always have the honor of introducing me to the world of research. I am thankfulness for his benevolence & time to time help ,valuable

suggestion ,constructive criticism & active interest in successfully making of this paper.

IX. REFRENCE

[1] Bharadwaj, V., Ghose, D., Robertazzi, T.G.: Divisible Load Theory: A New Paradigm for Load Scheduling in Distributed Systems. Cluster Comput. 6, 7--17 (2003).
[2] Cierniak, M., Zaki, M.J., Li, W.: Compile-Time Scheduling Algorithms for Heterogeneous Network of Workstations. Computer J. 40, 356--372 (1997).
[3] Lastovetsky, A., Reddy, R.: Distributed Data Partitioning for Heterogeneous Processors Based on Partial Estimation of their Functional Performance Models. In: HeteroPar'2009. LNCS, vol. 6043, pp. 91--101. Springer (2010).
[4] Legrand, A., Renard, H., Robert, Y., Vivien, F.: Mapping and load-balancing iterative computations. IEEE T. Parall. Distr. 15, 546--558 (2004).
[5] Grama, Gupta, Karypis, Kumar. Introduction to Parallel Computing, Addision –Wesley, 2nd Edition, 2003.
[6] Rajkumar Buyya et.al. High Performance Cluster Computing Vol 1, Prentice Hall PTR, 1999 .
[7] G. R. Andrews, D. P. Dobkin, and P. J. Downey, "Distributed allocation.
[8] Dr. A. k ghosh," in ACM SIGACT-SIGOPS Symp. Principles of Distributed Computing, Aug. 1982, pp. 73-83.
[9] S. Malik, "Dynamic Load Balancing in a Network of Workstation", 95.515 search Report, 19 November, 2000.
[10] Zhong Xu, Rong Huang, "Performance Study of Load Balancing Algorithms in Distributed Web Server Systems", CS213 Parallel and Distributed Processing Project Report.
[11] Berger, M. J., and Bokhari, S. A partitioning strategy for non-uniform problems on multiprocessors. IEEE Trans. Comput. C-26 (1987), 570-580.
[12] G. Cybenko, Dynamic load balancing for distributed-memory multiprocessors, J. Parallel Distrib. Comput. 7 (1989), 279–301..
[13] Cariño, R.L., Banicescu, I.: Dynamic load balancing with adaptive factoring methods in scientific applications. J. Supercomput. 44, 41--63 (2008).