An Approach for Process Migration using Task Scheduling

Chandu Vaidya
M-Tech (2nd Year)
RKNEC Nagpur
WCE Sangali

Prof. M.B. Chandak
HOD RKNEC
Nagpur

ABSTRACT
Given paper contain proposed approached for task scheduling which is done on a group of computers. Analysis of process data part by dividing them into number of fixed part & merge into single set that as good as previous original data set. Parallelism an approach for doing jobs in amount of time i.e very fast. The paper contain dynamic approach for process migration using thread level paradigm. Creating a thread of process into number of task, that leads to reduce total execution time of process. An algorithm is used to calculate PCB for decision purpose to achieve load balancing. Fair share approach is considered to allocating task to every processor using preemption strategy. The MPI[4] [10]is used for process communication. This system has defined to reduce total execution time on onboard & between board times. Open knoppix & MOSIX platform(Middleware) are used to show the results. Prime number calculation code is used to show parallel architecture like SIMD computer. Cluster computing is way of resource managing & scheduling strategy.

General Terms
Cluster server, Middleware, Node, Resources.

Keywords
Cluster computing, MOSIX, MPI, load balancing, threads, Task load. Onboard-time, betweenbord time.

1. INTRADUCTION
The objective is to develop an algorithm for load sharing by inducing parallelism mechanism on a group of interconnected machines. This algorithm is useful only when the cost factor can be underestimated when compared to time. The algorithm developed should be smart enough to migrate thread to other node in the cluster [3]only when the time requirement for completion of process can be reduced by doing so. Traditionally, computer software has been written for serial computation. To solve a problem, an algorithm is constructed and implemented as a serial stream of instruction. These instructions are executed on a CPU on one computer. Only one instruction may execute at a time after that instruction is finished, the next is executed. If load increase or more load is given the time requirement for execution will be more. For reducing the execution time to get output concept of Parallel Computing arises. Parallel computing uses multiple processing elements simultaneously to solve a problem. This is accomplished by breaking the problem into independent parts so that each processing element can execute its part of the algorithm with the others. A computer cluster is a group of linked computers, working together closely thus in many respects forming a single computer. The components of a cluster are commonly, but not always, connected to each other through fast LAN. Clusters are usually deployed to improve performance and availability over that of a single computer, while typically being much more cost-effective than single computers of comparable speed or availability. Load balancing [1][2]is when multiple computers are linked together to share computational workload or function as a single virtual computer. Logically, from the user side, they are multiple machines, but function as a single virtual machine. Requests initiated from the user are managed by, and distributed among, all the standalone computers to form a cluster. This results in balanced computational work among different machines, improving the performance of the cluster systems. Scheduling refers to the way processes are assigned to run on the available CPUs. This assignment is carried out by software known as scheduler and dispatcher. Scheduler and dispatcher operate with the help of a software known as middleware's. Middleware is computer software that connects software components or people and their applications. The software consists of a set of services that allows multiple processes running on noe or more machines to interact. The middleware we are using is MPICH2. MPICH2 is an high performance and widely portable implementation of the Message Passing Interface standard. It efficiently support different computation and communication platforms including commodity clusters, SMPs, massively parallel systems and high-speed networks.

2. BACKGROUND
This system is refined from the concept of executing of tasks using single processor. Uni-processor system functioning includes preemptive scheduling scheme. We change this by using multiple processors to execute a particular task in proportional manner to reduce time to execute the task in relatively short time. The processors are connected with each other in a Cluster, such that it is viewed as a single coherent entity .Non-preemptive scheduling scheme is used for this project. This improves the performance of execution of tasks as compared to earlier type. This project uses fair scheduling approach for providing fair access to users.

This system is an example of a distributed system[8]. This project is a scheduling system that provides allocation of system resources of one or more processor sets among groups of processes. Each of the process groups is assigned a fixed number of shares, which is the number that is used to allocate system resources among processes of various process groups within a given processor set. The described fair share scheduler considers each processor set to be a separate virtual computer.

Cluster computing[9][10] (or the use of computational Clusters) is the application of several computers to a single problem at the same time usually to a scientific or technical problem that requires a great number of computer processing cycles or access to large amounts of data.A Cluster can provide significant processing power for users with extraordinary needs. Animation software, for instance, which is used by students in the arts, architecture, and other departments, eats up vast amounts of processor capacity.

Description:

The main function of client/user is to submit the process in the process pool related with a processor. The processes in the process pool are waiting for the execution. From these processes the higher priority process is selected by using the appropriate scheduler and is given to the Cluster server.

The process division is a function that divides the process into the pieces or threads.

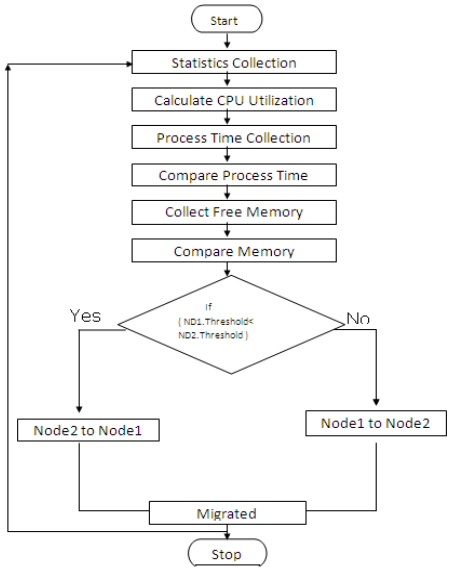

Fig-Simple model.

Thread distribution distributes these threads proportionally among the several nodes in the Cluster network[10]. Thread execution is a function that executes each thread independently on different nodes. The threads are executed using Fair-Share Scheduling. It allocates equal CPU time for each node. While executing threads, the resources required for the execution of that thread on the node, the load on the node and the complexity of each threads are taken into account. Each node may require same or different resources for the execution of the thread. These resources must be provided to each node. Above simple model(fig) show the general idea regarding project. Finally the output from each node is combined and the final output is given to the Cluster server.

3. EXPLANATIONS RELATED TO STRUCTURE

The object model of the given system includes different classes with the data objects having attributes of particular attribute values. These classes contain the functions which are dedicated to perform the appropriate tasks. Also the processes submitted to the system are moving in a specific manner undergoing some events. Processes are going through different states as shown in state diagram.

Above description about the Cluster system is discussed in following modules:

1. Cluster server: This class derived from Cluster setup as a client or a user supplies a set of process to the Cluster server the main objective of Cluster server[9] is to select an appropriate process for execution purpose. The basic idea is that every process is concerned with its priority, so each time a process with highest priority is selected by Cluster server from the process pool related it. To check the priority of the processes, process priority operation is used and a select process function grabs a particular process for execution.

2.Middleware: The main objective of middleware is to communicate with and manipulate heterogeneous hardware and data sets. It brings co-ordination between Cluster nodes and their resources. Middleware has well defined standards as its attributes. These standards have control over security and communication purpose of the Cluster system.

3.Resources: Resources are including the attributes as database and support files. They help in execution of the tasks given to the nodes. Each node completely relies on a set of resources such that a particular node is concerned with execution of some specific tasks only. The manage operation has a record of each node and its related resources.

4.Node: Node class actually executes the threads given after distribution in Cluster. Server or Cluster node is also subclasses of node. Thus execution of thread is operation of node class. Node has associated with scheduler and scheduler uses Fair-Share Scheduling scheme which allocates proportional CPU usage to nodes.

4. TECHNOLOGY PREFER

MPI:-MPI[10] (Message Passing Interface) is the middleware. Middleware is computer software that connects software components or people and their applications. The software consist of a set of services that allows multiple processes running on one or more machines to interact. The middleware we are using is MPICH2.

MPICH2:- MPICH2 is an high performance and widely portable implementation of the Message Passing Interface standard. It efficiently support different computation and communication platforms including commodity clusters, SMPs, massively parallel systems and high-speed networks.

POSIX:-POSIX is a "Portable Operating System Interface for Unix" API. It efficiently support different computation and communication platforms including commodity clusters, SMPs, massively parallel systems and high-speed networks.

OpenMOSIX:-OpenMOSIX was a free cluster management system that provided single-system image (SSI) capabilities, e.g. automatic work distribution among nodes. It allowed program processes (not threads) to migrate to machines in the node's network that would be able to run that process faster (process migration). It was particularly useful for running parallel and intensive input/output (I/O) applications.

***1.mosps* :-** shows MOSIX and related processes.

***2. migrate* : -** to migrate a process to the given computer.

Syntax:-"migrate {pid} {hostname or IP-address or node-number}"

3.moskillall : - to kill all your MOSIX processes (with the SIGTERM signal).

Syntax:-moskillall [-{signum} | -{symbolic_signal}] -G[{class}] [-J{jobid}]

Controlling running processes (migrate)

If we *transfer* the state of a process from one machine to another, we have *migrated* the process. Process migration is most interesting in systems where the involved processors do not share main memory, as otherwise the state transfer is trivial. A typical environment where process migration *is* interesting is autonomous computers connected by a network.

you can manually migrate the processes of all users send them away bring them back to home move them to other nodes freeze or unfreeze (continue) them, overriding the MOSIX[7] system decisions as well as the placement preferences of users. Even though as the Super-User you can technically do so, you should never kill (signal) guest processes. Instead, if you find guest processes that you don't want running on one of your nodes, you can use "migrate" to send them away (to their home-node or to any other node).

A checkpoint/restart facility is one which will

allow a process to save its state to a checkpoint this checkpoint will later be subjected to a restart procedure which will resume execution of the check pointed process
at the point at which the checkpoint was made.
Such a facility is referred to as a "Checkpoint/Restart" mechanism; such mechanisms have been available in
operating systems since the 1960s.
Syntax : migrate {{pid}|-j{jobID}} {node-number|IP-address|host}

5. IMPLEMENTATION

Implementation purpose some steps are considered throughout the project that are as follows .

BASIC STEPS

1.Clustering.
2.Stasatical collection.
3.Setting Threshold.
4.Thread Creation
5.Making Decision
6.Process migration.
7.Process Execution.
8.Collecting Back & Merge.
9.Analysis.
10. Deployment on Kernel code.

Algorithms used

Algorithm schedule process
Input: none
Output: none
{
while (no process picked to execute)
{
for (every process on run queue) pick highest priority process that is loaded in memory;
if (no process eligible to execute)
idle the machine;
/* interrupt takes machine out of idle state */
}
remove chosen process from run queue;
switch context to that of chosen process, resume its execution;
}

Example of SIMD Application:

Prime No. Generation Using MPI :

Algorithm:

1] Start.
2] Input no. of tasks.
3] Calculate Rank of each Task.
4] i) calculate Stride distance using ,
stride=2 * tasks;
ii) Fork the tasks & assign start to each Task using,
start = rank * 2 + 1;
5]for each task,
i]calculate largest prime no.
ii]store it in slarge, & increment prime count;
6]synchronize outputs of all task & calculate largest prime no.
7]finalize output & display time as,
i) total time.
ii) communication time.
I]On board communication time.
II]Between board communication time.
iii) execution time.
8] Stop.

For Demonstration purposed we select on task that is finding largest prime no. up to 25000000 as well as total number of prime numbers;

Time taken by Standalone Computer for above problem = **750.87 sec** we want to reduce this huge execution time by using our approached
Between board communication time is nearly constant for any number of tasks. Because system bus speed between board is nearly constant at any time.
On-board communication time varies with the no. of tasks.

obt α n
Where,
obt = onboard communication time.
n = number of tasks

. Total communication time (tcm) varies with no. of tasks.

tcm = obt + bbt
Total Execution time is calculated as :
Total time = communication time + Execution time
= (Onboard time + Between Board time) + Execution time
Consider,

Tob=On Board time
N=no. of tasks
Texe= execution time
Tob α N
Texe α 1/N

Our simple model contain the basic architecture of the project that indicate how the flow of project goes. We conclude that implementation of task scheduling which lead to fair share process allocation and load balancing as well as the total execution time. We have been able to collect the information of all nodes in Cluster environment. We have been able to perform load balancing by considering available resources such as free memory and CPU utilization for migratable processes. This increases the performance of the Cluster by decreasing the execution time for the processes. For testing purpose we have select Prime number generation program using MPI programming it's. good enough to success this approached. Deployment of our approached on kernel code so that we will in the position to developed one component module that will beneficial to someone.

6. ACKNOWLEDGMENTS

Success is the manifestation of diligence, inspiration, motivation and innovation. I attribute my success in this venture to my seminar guide Prof. M. B. Chandak, (HOD) who showed the guiding light at every stage of my seminar preparation.
I indebted to Dr.N.V Thakur, M-Tech Coordinate of the Department Computer Science & Engineering, who has provided facilities and the infrastructure to work at an extended ends.
Last but not the least I am also thankful to all the faculty members for helping directly or indirectly to accomplish the seminar work. I would like to thank My Mind for not letting me down at the time of crisis and showing me the silver lining in the dark clouds.

7. REFERENCES

[1] M. Willekk-Lemair and A.P. Reeves, Strategies for dynamic load-balancing on highly parallel computers, IEEE Transaction on Parallel and Distribured Systems, (4)9, September 1993, Pages 979-993.

[2] M. Wu and W. Sbu, A load balancing algorithm for n-cube, Proceedings of rhe 1996 Inremarwnal Conference on Parallel Processing, IEEE Computer Society, 1996, Pages 148-155.

[3] H. Shan, J.P. Singh, L. Oliker and R. Biswas, "Messge passing and shared address space parallelism on an SMP cluster," Parallel Computing, vol 29, 2003, pp. 167-186.

[4] W. Pan, L. Chan, J. Zhang, Y. Li, L. Wan and F. Xia, "Research on MPI+OpenMP hybrid programming paradigm based on SMP cluster," Application Research of Computers, vol. 26, 2009, pp. 4492–4594

[5] M.Cosnard, E. Jeannot, and L. Rougeot, Low MemoryCostDynamicSchedulingofLargeCoarseGrainTaskGraphs,Proc.Int'lParallelProcessingSymp./Symp.ParallelandDistributedProces-sing(IPPS/SPDP), Mar.1998

[6] Rewini,H.H.Ali,andT.G.Lewis,TaskSchedulingin MultiprocessingSystems, Computer, pp.27-37,Dec.1995.

[7] Oren LA'ADAN Amnon BARAK andAmnonSHILOH.ScalableclustercomputingwithMOSIXforLINUX.InProc.LinuxExpo'99,pages95–100,May1999.

[8] N. Islam and A. Prodromidis and M. S. Squillante, "Dynamic Partitioning in Different Distributed-Memory Environments", Proceedings of the 2nd Workshop on Job Scheduling Strategies for Parallel Processing, pages 155-170,April 1996.

[9] Huajie Zhang School of Math, Physics and Information Engineering College of Zhejiang Normal University, Jinhua 321004, China" On Load Balancing Model for Cluster Computers", IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.10, October 2008.

[10] Amith R. Mamidala Rahul Kumar Debraj De D. K. Panda Department of Computer Science and Engineering" MPI Collectives on Modern Multicore Clusters: Performance Optimizationsand Communication Characteristics", Eighth IEEE International Symposium on Cluster Computing and the Grid.