A distributed system architecture for maximum utilization of idle resources and speedup for batch processing tasks.

Nikita Patil.
Computer Engg., Department, DCOER, Pune, India.

Tanvi Kanade.
Computer Engg. Department, DCOER, Pune, India.

Priyanka Sonone.
Computer Engg. Department, DCOER, Pune, India.

Priyanka Londhe.
Computer Engg. Department, DCOER, Pune, India.

Kavita Kimmatkar.
Assistant professor, Computer Engg. Department, DCOER, Pune, India.

Ashwini Mane.
Assistant professor, Computer Engg. Department, DCOER, Pune, India.

Dhiraj Motghare.
BE(CSE), MBA, TCS. Pune, India.

Abstract — Computers which are used in organizations for normal task, may have wastage of resources like processing power, memory (mainly primary memory). If computers are used as a single unit, even then some amount of resources are unutilized. To use these wasted resources distributed computing proves to be beneficial. The existing distributed systems can perform only homogeneous tasks efficiently and for performing these tasks, manual intervention is essential. In distributed systems tasks are executed at lowest priority. The proposed system can execute heterogeneous tasks and set priority to tasks. Therefore reduction in waiting time of tasks in queue is achieved. In the proposed system, batch processing is used so that without manual intervention tasks are executed. It also ranks all computers according to the configuration and status. Proposed system use SAC algorithm and Local Queue algorithm for distributing input files and load balancing respectively. Using all these features and algorithms, proposed system may utilize maximum idle resources, which increase execution speed and performance exponentially.

Keywords - Batch processing, Distributed computing, Horizontal scaling, Local Queue.

I. INTRODUCTION

Computers are used as single unit, then some wastage of resources takes place. To utilize these idle resources Distributed computing [6] is used. Distributed computing refers to the distributed systems. It is used to solve computational problems. In distributed computing, a problem is divided into many tasks, each task is solved by one or more computers, which communicate with each other by message passing. In proposed system for communication between two or more computers connected in LAN, RMI (Remote Method Invocation) is used. RMI is an API that provides a mechanism to create distributed applications in java. The RMI allows an object to invoke methods, addresses of computers on an object running in another JVM (Java Virtual Machine). The RMI provides remote communication between the applications using two objects stub and skeleton. To execute a series of programs or tasks on a computer without manual intervention, Batch processing [9] is used. For executing programs without manual intervention all input data is preselected through scripts or command-line parameters. A program takes a set of data files as input, processes the data and produces a set of output data files. This operating environment is termed as "Batch processing" because the input data is collected into batches of files and is processed in batches by the program. To connect multiple hardware or software entities, Horizontal scaling [10] is used. Hardware and software entities work as a single logical unit.

The objective of this paper is to develop a tightly coupled computer cluster system. This system consists of stand-alone computers in big organizations, to achieve a maximum consumption of the resources. Resources of the computers are like processing power, memory (mainly primary memory). Proposed system attempts to make use of such a computer system to accomplish various heterogeneous computer tasks, such as batch processing (image processing, file processing, sorting, searching, pattern matching, database operations such as indexing etc.). In this paper we propose to make our own batch processing scripting language. This batch processing scripting language will assist us in writing the distributed tasks. Proposed system aims to utilize the idle resources. It also aims to improve the performance given by an individual computer against the tightly coupled computer cluster system. The performance is expected to get faster by N times (where N is the number of tightly coupled computer clusters in the network system).

II. LITERATURE REVIEW

Distributed computing refers to the distributed system. Distributed system contains many computers connected to each other. In this system there are computers having different hardware and software configuration. Therefore it is necessary to study all hardware software configuration and their properties to increase execution speed and performance.

A. Data distribution algorithms

Data distribution algorithms are used to split all input files according to the number of computers connected in LAN. After processing, the computer sends output files to main computer. The main computer collects output files from all other computers and combines it in one folder. The proposed SAC algorithm is a modification of Map Reduce algorithm. SAC algorithm is used to distribute and collect input and output files respectively.

1) Map-Reduce algorithm

Map-Reduce[1] is a programming model and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster. To implement Map-Reduce with hadoop , Linux cluster is needed to setup the cluster system, which requires knowledge on hadoop. It also requires implementation of hadoop. Map-Reduce on hadoop has no GUI(Graphical User Interface). Map-Reduce uses DFS i.e. Distributed File System concept. Also Map-Reduce cannot assign tasks based on capabilities.

2) SAC algorithm

According to working of the proposed system some changes in map reduce are required. Therefore in this paper, we propose new SAC(Split And Collect) algorithm. It is used to spit all input files which are given by the user. After processing, when all the computers send their output files to main computer, then SAC collects all files in one folder. The main computer then gives this folder contains all processed files to user. SAC runs on Windows as well as Linux OS. It has a user friendly GUI and it only requires LAN network. The proposed system uses SAC for compressing technique to compress the data and saves time in transporting data to program. SAC can also assign tasks based on capabilities, so speed of execution is considerably increased.

B. Load balancing

Load balancing [7] becomes an essential factor when it is used for distributing the workloads across multiple computing resources. Computing resources means computers or clusters of computers in a distributed system. Many algorithms can be used for achieving the efficient distribution of workloads but particular algorithms perform well in particular conditions. The Round Robin Algorithm[7], Randomized Algorithm[4], Central Queue Algorithm[4], Central Manager Algorithm[4] , Threshold algorithm[4] , Threshold algorithm[4] , Local Queue algorithms can be used for this purpose. A comparative study reveals the compatibility level and efficiency of these algorithms in the proposed system.

1) Round Robin Algorithm

Round Robin algorithm [7] distributes jobs evenly to all slave processors. All jobs are assigned to slave processors based on Round Robin order. Round Robin order means processor assigning is performed in series till the last processor is not reached when last processor is reached it will be back to the first processor. Processors assigning is not consider allocations of other processor. Processor assigning is performed locally on each processor. Round Robin algorithm does not require inter-process communication is the main advantage of algorithm. In general, Round Robin is not expected to achieve good performance in general case.

2) Randomized Algorithm

Randomized algorithm [4] choose slave processors using random number. The slave processors are chosen randomly and random numbers generated based on a statistic distribution. For perticular special purpose application Randomized algorithm can attain the best performance among all load balancing algorithms .

3) Central Queue Algorithm

Central Queue Algorithm [4] works on the principle of dynamic distribution. It stores new activities and unfulfilled requests as a cyclic FIFO queue on the main host. Each new activity arriving at the queue

manager is inserted into the queue. Then, whenever a request for an activity is received by the queue manager, it sends it to the requester and removes the first activity from the queue. If there are no ready activities in the queue, the request is buffered, until a new activity is available. If there are unanswered requests in the queue and If a new activity arrives at the queue manager, then first unanswered request is removed from the queue and the new activity is assigned to it. When a processor load falls under the threshold, the local load manager sends a request for a new activity to the central load manager. The central load manager answers the request immediately if a ready activity is found in the process-request queue.

4) Central Manager Algorithm

Central Manager Algorithm [4] central processor will choose a slave processor to be assigned a job in each step. The slave processor having the least load is chosen as processor. The central processor is able to gather all slave processors load information, therefore the choosing based on this algorithm are possible to be performed. The load manager makes load balancing decisions based on the system load information, allowing the best decision when of the process created. High degree of inter-process communication could make the bottleneck state.

5) Threshold Algorithm

In Threshold algorithm [4] , the processes are assigned immediately upon creation to hosts. Hosts for new processes are selected locally without sending remote messages. Each processor keeps a private copy of the system's load. The load of a processor can characterize by one of the three levels: under loaded, medium and overloaded. Two threshold parameters ta_under and ta_upper can be used to describe these levels. Under loaded: load < ta_under , Medium : ta_under = load = ta_upper , and Overloaded: load > ta_upper. Initially, all the processors are considered to be under loaded. When the load state of a processor exceeds a load level limit, then it sends messages regarding the new load state to all remote processors, regularly updating them as to the actual load state of the entire system.

6) Local Queue

Main feature of Local Queue algorithm [6] is dynamic process migration support. The basic idea of the Local Queue algorithm is static allocation of all new processes with process migration initiated by a host when its load falls under threshold limit, is a user-defined parameter of the algorithm. The parameter defines the minimal number of ready processes the load manager attempts to provide on each processor. Initially, new processes created on the main computer are allocated on all under loaded computers. The number of parallel activities created by the first parallel construct on the main computer is usually sufficient for allocation on all remote computers. Then all the processes created on the main computer and all other computers are allocated locally. When the computer gets under loaded, the local load manager attempts to get several processes from remote computers. It randomly sends requests with the number of local ready processes to remote load managers. When a load manager receives such a request, it compares the local number of ready processes with the received number. If the former is greater than the latter, then some of the running processes are transferred to the requester and an affirmative confirmation with the number of processes transferred is returned.

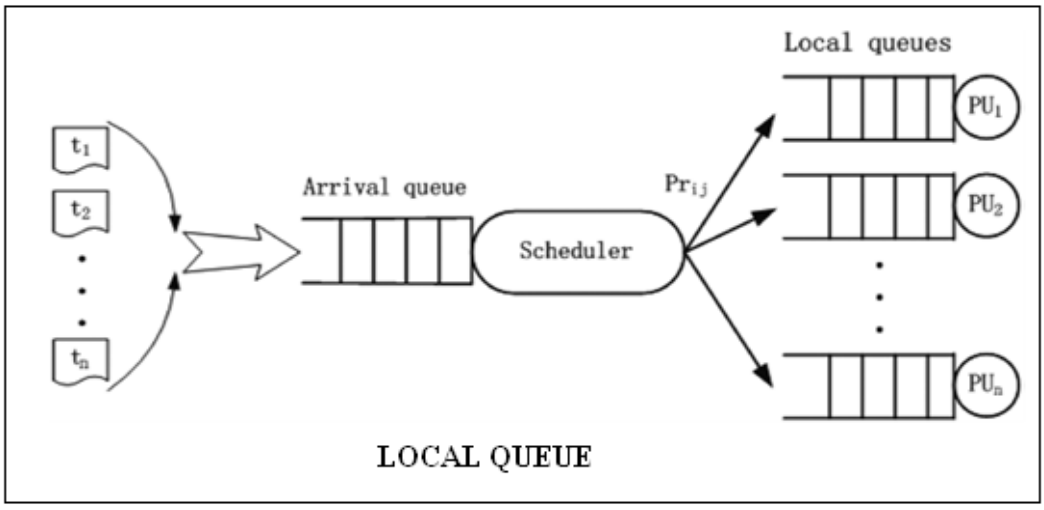

Fig 1. Local Queue working

The comparative study of various quality parameters of the above load balancing algorithms reveals their compatibility in the proposed system.

Table 1. Comparative study of algorithms.

Parameters	**Round robin**	**Random**	**Central queue**	**Central manager**	**Threshold**	**Local Queue**
Reliability	Less	Less	More	Less	Less	More
Adaptability	Less	Less	More	Less	Less	More
Overload rejection	No	No	Yes	No	No	Yes
Predictability	More	More	Less	More	More	Less
Fault tolerant	No	No	Yes	Yes	No	Yes
Forecasting accuracy	More	More	Less	More	More	Less
Stability	Large	Large	Small	Large	Large	Small
Centralized / Decentralized	D	D	C	C	D	D
Dynamic / static	S	S	Dy	S	S	Dy
Cooperative	No	No	Yes	Yes	Yes	Yes
Process migration	No	No	No	No	No	Yes
Resource utilization	Less	Less	Less	Less	Less	More
Non preemptive / Preemptive	Np	Np	Np and p	Np	Np	Np and p
Throughput	Low	Low	High	Low	Low	High
Waiting time	More	More	Less	More	More	Less
Turnaround time	Less	Less	More	Less	Less	More

Performance of Distributed System Load Balancing Algorithms is checked using Comparative Analysis of algorithms. Comparison is done based on Qualitative Parameters [8]. According to the proposed system, Local Queue algorithm is found to be the best for load balancing.

C. Rank all computers

In this paper a new policy for ranking the computers is proposed. The computers are ranked according to their present memory sizes and hardware and software configurations. Then according to the rank of computers the jobs are distributed accordingly. Thus time required for processing is reduced and performance is also expected to increase[2].

D. Heterogeneous Distribution

This paper attempts to develop a system which can process heterogeneous tasks [3]. Consider for example that the task of processing a large amount of images is to be performed by the proposed system. These images are of heterogeneous type i.e. the images to be processed are bitmap, JPEG and GIF which are combined together. The proposed system is expected to process all the types of images without any loss of efficiency.

E. Priority

The proposed system is expected to set priorities to tasks. The priorities are set in such a way that the amount of waiting time of the tasks in the queue is minimal[5]. The proposed system may therefore finish the given tasks within less amount of time.

III. METHODOLOGY

The proposed system consists of a number of computers which are connected in LAN. The computer to which an input is given by the user is treated as main computer. This computer acts as distributor and is responsible for distributing the input among the other connected computers.

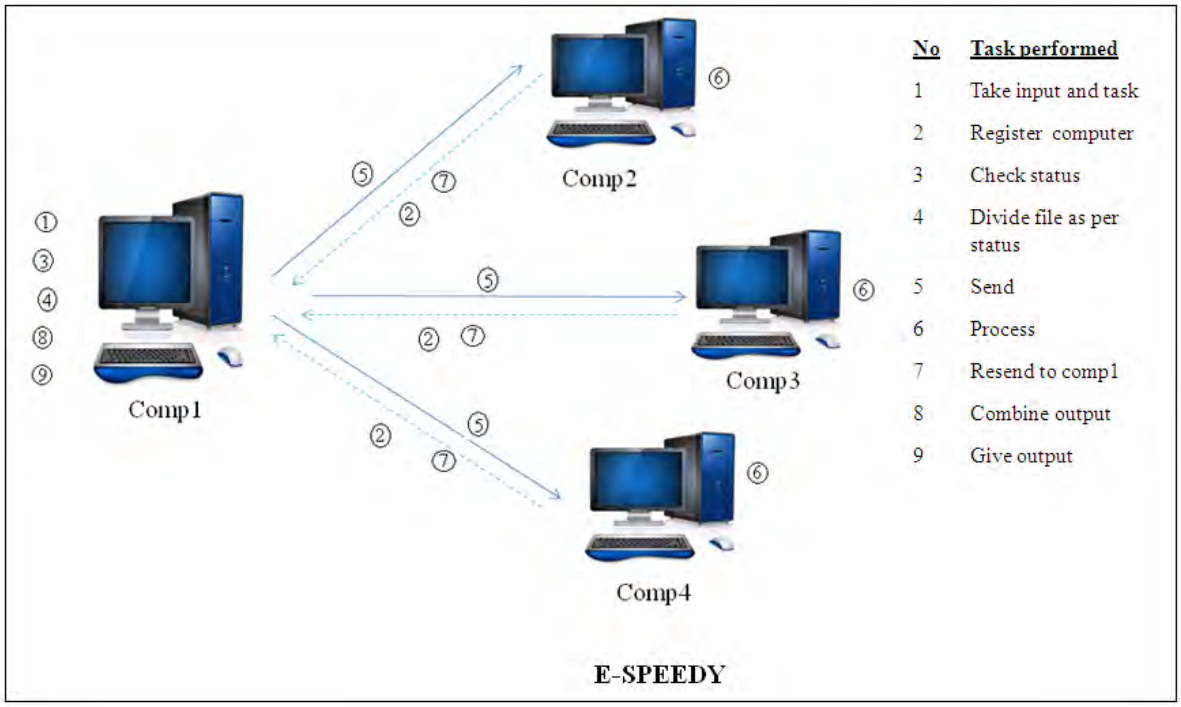

Fig 2. E-SPEEDY

In the above proposed system, the user gives the input files and task file to any one computer which is connected in LAN. The task file contains the Java code of the functions that should be performed on the input files. Any task that can be coded in Java is expected to be performed by the proposed system. The computer to which user gives input acts as the main computer. It is the responsibility of the main computer to distribute the given input to other computers in the network. In this system the main computer communicates with other computers using RMI. It sends messages to all other computers to ask them whether they are available to perform the task. If they are available they need to register themselves to the main computer. Then main computer stores the host name of all computers which register themselves in one array. It also stores the status of all computers. Using SAC algorithm all input files are split according registered status of computers. Then a zip file for each computer which contains input file and task file is created. In this system Local Queue algorithm distributes these zip files to each computer. All computers unzip the received zip file. The computers then process all input files according to the task file. After processing, a zip file containing all processed files i.e. output files is created .When all processing is completed the computer sends signal to main computer which indicates that all processing is finished. When signal reaches to main computer it takes zip file of output files and unzips it. When all computers send their zip files, then main computer collects all output files in one folder and gives this folder to user .If any computer stops working due to any reason then for disaster recovery that computer sends an error message to main computer. The main computer then resends those input files to some other computer for processing.

IV. CONCLUSION

There are many existing distributed computing systems which aim at CPU utilization and speedup. Most of the existing systems need manual intervention and can process only homogeneous tasks. Therefore the efficiency and performance of those systems is limited to a certain extent. The proposed system aims at maximum utilization of resources by introducing features like ranking, priority setup and heterogeneous task processing for computers. The proposed system is expected to give exponential increase in speed for batch processing tasks and also maximum utilization of idle resources.

V. REFERENCES

[1] Gregory Tauer , Rakesh Nagi , "A map-reduce lagrangian heuristic for multidimensional assignments problems with decomposable costs",Parallel Computing 39(2013),Page(s):653-668.

[2] Jianzhe Tai ,"Load balancing for cluster systems under heavy-tailed and temporal dependent workloads", Simulation Modelling Practice and Theory 44 (2014),Page(s):63–77.

[3] Pushpendra Kumar Chandra, Bibhudatta Sahoo , "Dynamic Load Distribution Algorithm Performance in Heterogeneous Distributed System for I/O- intensive Task", IEEE Region 10 Conference 19-21 (Nov.2008), Page(s):1-5.

[4] Sandeep Sharma , Sarbjit Singh , Meenakshi Sharma , "Performance Analysis Of Load Balancing Algorithms",World Acadamy Of Science,Engineering and Technology 14 (2008),Page(s):1-16.

[5] Dhinesh Bhau L.D. , P. Venkata krishna , "Honey bee behavior load balancing of tasks in cloud computing environments ",Applied soft computing 13(2013), Page(s):2292-2303.

[6] Zhao Tong , Zheng Xiao , Kenli Li , Keqin Li , "Proactive scheduling in distributed computing-A reinforcement learning approach" ,J. Parallel Distrib. Comput. 74 (2014),Page(s): 2662–2672.

[7] Ajay Tiwari , Priyesh Kanungo , "Dynamic Load Balancing Algorithm for Scalable Heterogeneous Web Server Cluster with Content Awareness",IEEE (2010),Page(s):143-148.

[8] Amit Chhabra , Gurvinder Singh , " Qualitative Parametric Comparison of Load Balancing Algorithms in Distributed Computing Environment", IEEE (2006),Page(s):58-61.

[9] Arnaud Malapert, Christelle Gueret, Louis-Martin Rousseau, "A constraint programming approach for a batch processing problem with non-identical job sizes", European journal of operational research (2012),page(s):13.

[10] Ka-Man Wong,Lai-Man Po,Kwok-Wai Cheung,Chi-Wang Ting, Ka-Ho Ng, Xuyuan Xu, "Horizontal Scaling and Shearing-Based Disparity-Compensated Prediction for Stereo Video Coding"IEEE(2012),page(s):14.