International Journal of Scientific Research in Computer Science, Engineering and Information Technology

© 2017 IJSRCSEIT | Volume 2 | Issue 5 | ISSN : 2456-3307

A Survey for Different Classifications of Distributed Systems Scheduling Using Markov Chain Model

Shweta Jain[1], Saurabh Jain[2]

[1]Research Scholar, Faculty of Computer Science, Pacific Academy of Higher Education and Research University, Udaipur,Rajasthan, India

[2]Professor, Shri Vaishnav Institute of Computer Applications, Shri Vaishnav Vidyapeeth Vishwavidyalaya, Indore, Madhya Pradesh, India

ABSTRACT

With the establishment of internet with network technologies and the maturity in the computing industry, the distributed system has become popular and important with aspects of a challenging activity. The performance of computing system has enhanced extensively with the adding up of the concepts of multiprocessing and multi-computing. In distributed system, computing and developing of scheduling algorithm for vast spectrum of applications are a key problem and of greater interest of studying and scheduling for resource management. This paper presents a survey to classify various existing distributed system scheduling to apply the concept of Markov chain model to improve their throughput and performance.

Keywords : Coscheduling, Distributed system, Markov Chain, Scheduling, Stochastic, Transition Matrix

I. INTRODUCTION

A distributed system means different things to different people. A distributed system consists of a collection of loosely coupled diverse computing units which are geographically distributed and connected by a communications network. The key illustrations of distributed systems are web services and peer-to-peer systems. Over the last decade distributed systems have been emerging as popular computing platforms for computationally intensive applications with diverse computing needs to design and implement resource management systems with a variety of architectures and services.

Distributed system scheduling refers to the way processes are assigned to run on existing processors. But the problems lie in developing a scheduling algorithm to suit the various forms of applications, process of partitioning the application into tasks, encouraging coordinated communication among tasks, monitoring the synchronization among tasks, and mapping the tasks to the machines. It will also affect the system performance, then reduces the cost of performance, increases the efficiency of the system. In general, resource management is the use of available processors in the most efficient way possible. Scheduling the independent task is to reduce the computational time. Each independent task is scheduled to available suitable resources and runtime environment.

Scheduling algorithms play a key role in obtaining high performance in parallel and distributed systems. Nowadays, a wide variety of scheduling algorithms for distributed systems have been reported in the literature [1] [2] [3] [4] [5] [6] [7] [8] [9]. A probability-based stochastic model Markov chain is applied in order to determine the performance of these scheduling algorithms of distributed system. [11] Elaborated study of a variety of stochastic processes and their uses in the various applications. Some other useful contributions are for distributed system scheduling by [11] [12] [13] [14] [15].

II. SCHEDULING ALGORITHM MEASURES

Performance measures for distributed scheduling algorithms are as follows.

- System utilization refers percentage of time the resource is busy.
- Throughput refers the number of jobs processed in a given time period.

- Turnaround time refers as the sum of waiting time and execution time.
- Job slowdown refers as the ratio of the response time of a job to its actual execution time. It primarily occurs due to a long waiting time for execution of job.
- Economic-profit refers profit earned by resource according to usage.
- Make-span determines by subtracting start time of the first job from the actual finish time of the last job of particular application.
- Schedule Length Ratio divides the make-span by the expected time needed to execute the tasks present on the Longest Computation Path on the fastest resource of the application.
- Scheduling time depends on the time complexity and indicates how much time is taken by scheduling algorithm for making decision on assignment of jobs onto resources.
- Speedup indicates how faster an application runs on multiple resources as compared to running it on a single fastest resource.
- Flow time refers the sum of completion time of all the jobs.
- Economic Cost indicates total cost incurred for executing all jobs of the application.
- Fairness refers each process gets a fair share of the processors and resources.

III. GENERAL CLASSIFICATION OF DISTRIBUTED SYSTEM SCHEDULING

This classification of scheduling is based on different criteria, such as static vs. dynamic environment, centralized vs. distributed etc. Different distributed computing scheduling is classified as follows in the figure:

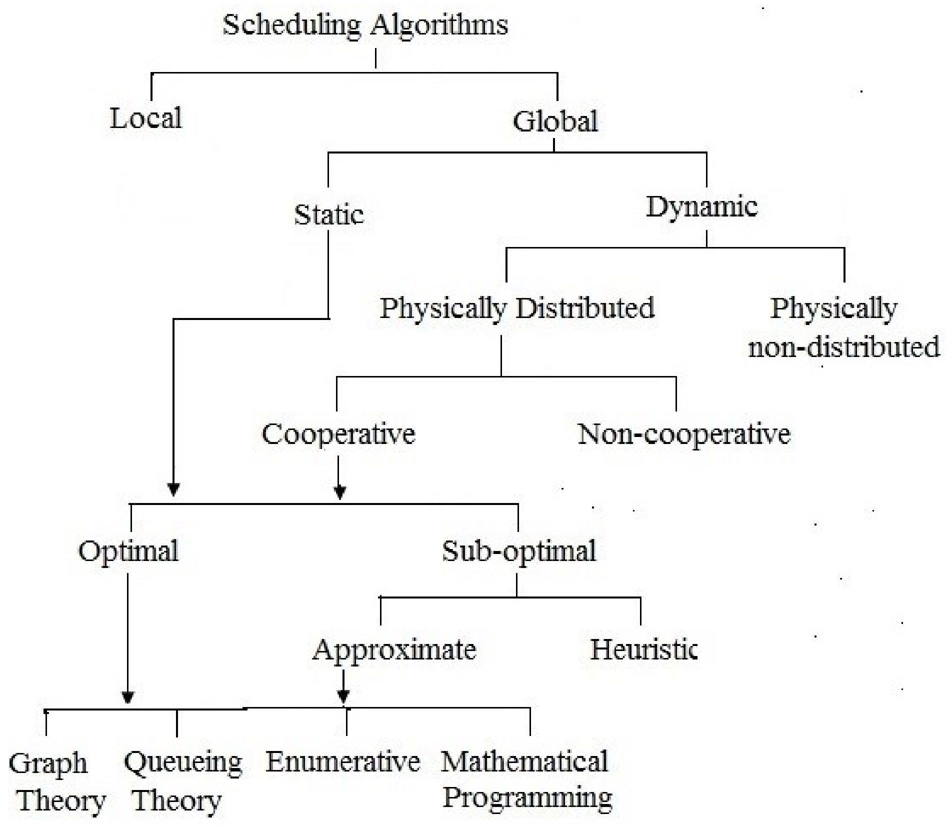

Fig : Distributed System Scheduling Classification

There are two levels of scheduling in a system: global scheduling and local scheduling [10].

- **Local vs. Global scheduling**

At the highest, level, there is local and global scheduling. Local scheduling is involved with the assignment of processor time of a single processor to processes. Global scheduling is the problem of deciding where to execute a process. The job of local scheduling is left to the operating system of the processor to which the process is ultimately allocated. This gives processors increased independence while reducing the responsibility and consequently overhead of the global scheduling.

- **Static vs. Dynamic scheduling**

The next level in the hierarchy scheduling is a choice between static and dynamic scheduling which is based on the time at which the scheduling decisions are made. Static scheduling refers assigning processes to processors at compile time. Dynamic scheduling means that processes are assigned to a processor when they begin execution (run time), and that they may be reassigned while they are running. A static scheduler makes decisions based only on information concerning the processes (expected execution time, I/O characteristics, etc.) and the static system (processor power, network configuration, etc.), while a dynamic scheduler takes into account the current state of the system (workload, queue lengths, etc.).

In the dynamic scheduling problem, priori knowledge is required about the resource needs of a process. It is also unidentified in what environment the process will execute during its lifetime. In the static case, a decision is made for a process image before it is ever executed, while in the dynamic case no decision is made until the process begins its life in the dynamic environment of the system. The principal advantage of static scheduling is its simplicity, because system state information need not be maintained. It is also effective when the workload can be sufficiently well described before making a decision. However, it fails to adjust to fluctuations in the system load.

- **Optimal vs. Sub-optimal**

Because all information regarding the state of the system and the resource needs of a process is known, an optimal assignment can be made based on some criterion function. Commonly use these following optimization measures are

1. Minimizing of the total process completion time.
2. Maximizing utilization of resources in the system.
3. Maximizing system throughput.

Because of the size of a distributed system (large number of processes, processors, and other resources that involve some restrictions) static scheduling is a complex computational problem. Thus getting optimal solutions can be very expensive and not feasible in a reasonable time period in many cases but sub-optimal solutions may be better in those cases. Dividing sub-optimal solutions to the scheduling problem is into two general categories.

- **Approximate vs. Heuristic**

The approximate approach uses the same formal computational model for the algorithm, but instead of searching the entire solution space for an optimal solution, it is to be satisfied when finding a good one. These solutions are categorized as sub-optimal-approximate. The problem is how to find out that a solution is good enough. The factors which determine whether this approach is worthy:

1. Availability of a function to evaluate a solution.
2. The time required to evaluate a solution.
3. The ability to judge the value of an optimal solution according to some measures.
4. Availability of a method for intelligently trimming the solution space.

The second category is based on heuristic search strategies which represent the solutions to the static scheduling problem which require the most reasonable amount of time and other system resources to perform their function. The heuristic schedulers use special parameters which affect the system and much simpler to calculate. This decreases the overhead involved in passing information between processors while dropping the interference among processes which may run without synchronization with one another.

- **Optimal vs. Sub-optimal approximate techniques**

Regardless of whether a static solution is optimal or sub-optimal-approximate, there are four basic categories of task allocation algorithms which can be used to arrive at an assignment of processes to processors.

1. Enumeration (solution space and search).
2. Graph theory.
3. Mathematical programming.

4. Queuing theory.

- **Distributed vs. Non-distributed**

The next concern (under dynamic solutions) considers whether the status information of all processors and execution environments is to be collected at one location or physically non-distributed, or whether the decision-making process is to be physically distributed among the processors that utilize information stored in many places.

The most important feature of making decisions centrally (non-distributed) is simplicity but such systems suffer from gathering information and maintaining a proper system state in one central location and leading to a large time overhead, since there are transfer delays and messages lost, low reliability of systems leads to failure of the node or falling down of the entire system due to load scheduling. A globally distributed load-scheduling algorithm does not suffer from the above drawbacks that are avoided collecting status information at a single location so that the schedulers can respond rapidly to dynamic changes in the system state. The scheduler decides that if one computer fails, others can continue their jobs.

- **Cooperative vs. Non-cooperative**

In distributed dynamic global scheduling, there are mechanisms cooperative which involve cooperation between the distributed components and mechanisms non-cooperative in which the individual processors make decisions independently without interferences to other processors. Degrees of independence of each processor determine how its own resources should be used. Each processor's local operating system is concerned with making decisions with other processors in the system in order to achieve some global goal, instead of making decisions based on the way in which the decision may also affect its local performance. Distributed system have already many schedulers and processors, each one is responsible for performing certain activity in scheduling processes with cooperation of procedures, rules and current system users and status.

- **Lower sub-branches of the dynamic branch**

In the static scheduling, the classification hierarchy has attained a point to consider optimal, sub-optimal-approximate, and sub-optimal-heuristic solutions. Optimal solutions may be optimal locally or globally. When optimal solutions are infeasible, it leads to use sub-optimal solutions. Both hierarchical options are based on mathematical programming, queuing theory, graph theory, and enumeration (solution space and search).

IV. ANOTHER COMMON CLASSIFICATION OF DISTRIBUTED SYSTEM SCHEDULING

- **First come First Served (FCFS)**

It is an abstracted way of organizing and allocating of resources to jobs over time, it serves as a principle of a queue processing or demands' servicing by ordering that means what comes in first is allocated first, what comes in next waits until the first is finished.

- **Round-robin (RR)**

It is a simplest and easy to implement scheduling algorithm based on time sharing among jobs in equal slice or quantum focusing on fairness between jobs. It works in circular queue without priority but it has a starvation problem. The advantage of RR is that no job has to wait for another one to be completed as FCFS and others but it is not a suitable for those jobs that are largely varies in their size and requirements.

- **Priority-based**

It is a preemptive based scheduling algorithm. Each process is assigned a certain level of priority that corresponds to the relative importance of the event that it services. The processor is normally allocated to the highest-priority process among those that are ready to execute. Higher-priority processes usually preempt execution of the lower-priority processes. As a job is waiting, raise its priority so eventually it will have the maximum priority which is a solution to the problem of starvation.

- **Local Scheduling**

In a distributed system, local scheduling means how an individual workstation should schedule those processes assigned to it in order to maximize the overall performance. In a distributed system, the local scheduler may need global information from other workstations to achieve the optimal overall performance of the entire system. There are two local scheduling techniques. Proportional-sharing scheduling and predictive scheduling.\

❖ **Proportional-sharing scheduling**

Basically priority-based schedulers are difficult to understand and give more processing time to users with many jobs, which lead to unfairness among users. So that Proportional-sharing scheduling is easy to implement and can solve this problem of allocating resources to users fairly over time in which the resource consumption rights of each active process are proportional to the relative shares that it is allocated.

There are two types:

1. Lottery scheduling

Lottery scheduling is a probabilistic scheduling algorithm for processes in an operating system which provides proportional-sharing of resources management. Processes are each assigned some number of lottery tickets, and the scheduler draws a random ticket to select the next process to be executed. The distribution of tickets need not be uniform; granting a process more tickets provides it a relative higher chance of selection. It solves the starvation problem in which every job gets at least one ticket. Processor time is proportional to the number of tickets given to each job. It considers that there could be a large number of tickets distributed among a large pool of threads. To have an array of tickets with each ticket corresponding to a thread may be highly inefficient [17].

2. Stride scheduling

It is a deterministic scheduling technique that efficiently supports flexible resource management abstractions like lottery scheduling. Compared to other scheduling, stride scheduling achieves significantly improved accuracy over relative throughput rates, with significantly lower response time variability. It implements proportional-share control over processor time and other resources which is designed for networks [18]. Fairness can also be guaranteed in stride scheduling.

❖ **Predictive Scheduling**

Predictive scheduling which is adaptive to the processor load and resource distribution of the distributed system. It provides intelligence, adaptivity and proactivity and adapts to new architectures, algorithms, methods and environmental changes automatically that are embedded into the system providing guarantees of service. Predictive scheduling is very effective in performance and reliability enhancement, even with the simplest methods, but at the cost of design complexity and management overhead.

Other scheduling policies are used either when a process blocks or at the end of a time slice, which reduces the performance and substantial drop of time during scheduling. Predictive scheduling solves this problem by predicting scheduling decision is necessary or predicting the parameters needed that are not known in advance. Based on the collected static information (machine type, CPU power, etc.) and dynamic information (memory free space, CPU load, etc.), predictive scheduling tries to make an educated guess about the future behavior, such as CPU idle time slot, which can be used to make scheduling decisions in advance. Predicting the future performance based on past information is a common strategy and it can achieve a satisfactory performance in practical work.

- **Coscheduling**

Meanwhile, coordinated scheduling of parallel jobs across the nodes of a multiprocessor (coscheduling) is also essential in a distributed system. In 1982, Outsterhout introduced the term “coscheduling” in which the process working set must be coscheduled or scheduled for execution simultaneously for the parallel program to make progress on distinct processors. It can produce benefits in both system and individual job efficiency. Without coordinated scheduling, the processor thrashing may lead to high communication latencies and consequently degraded overall performance. So the success of coscheduling becomes a more important factor in deciding the performance. Coscheduling is classified into three types:

1. Explicit coscheduling or gang scheduling

It requires all processing to actually take place at the same time, and is typically implemented by global scheduling across all processors. The strategy bypasses the busy-waiting problem by scheduling all processes at the same time. It works well for parallel jobs having a lot of inter-process communications. However it is a centralized scheduling strategy with a single scheduler making decisions for all jobs and all workstations. This centralized nature can easily become the bottleneck when the load is heavy. Although this scheduler can achieve high system efficiency on regular parallel applications which is difficult in selecting alternate jobs to run when processes block, requiring simultaneous multi-context switches across the nodes. To achieve good performance requires long scheduling quanta, which can interfere with interactive response, making them a less attractive choice for use in a distributed system. These limitations motivate the integrated approaches.

2. Local Coscheduling

It allows individual processors to schedule the processing independently. It's ease of construct but the performance of communicating jobs degrades significantly because scheduling is not coordinated across processors.

3. Dynamic or implicit Coscheduling

It is a form of coscheduling where individual processors can still schedule processing independently, but they make scheduling decisions in cooperation with other processors. It is for dynamically coordinating the time-sharing of communicating processes across distributed systems. However, implementations of explicit coscheduling often suffer from multiple failure points, high context-switch overheads, and poor interaction with client-server, interactive, and I/O-intensive jobs. But implicit coscheduling minimizes the communication waiting time of parallel processes by identifying the processes through gathering and analyzing implicit runtime information or communication events. Unfortunately it does not provide guarantee the performance of local and parallel jobs when increasing the number of parallel jobs compete from each other.

V. MARKOV CHAIN MODEL

This is a probabilistic-based stochastic model which will be applied to classify different kinds of distributed system scheduling schemes which are based on the following two concepts:

- Transition of the next state depends on the current state.
- A suitable estimation of the transition probabilities between states.

The stochastic process $\{X_n, n=0, 1, 2\ldots\}$ is called Markov chain, if, for j, k,
$j_1,\ldots j_{n-1}$ € N (or any subset of I), and
$P[X_n = k / X_{n-1} = j, X_{n-2} = j_1, \ldots, X_0 = j_{n-1}] = P[X_n = k / X_{n-1} = j] = p_{jk}$
The transition probabilities p_{jk} satisfy

$$\sum_k p_{jk} = 1$$

$p_{jk} \geq 0$ for all j.
These probabilities may be written in the matrix form

$$P = X^{(n-1)} \updownarrow \begin{array}{c|cccc} \multicolumn{5}{c}{\longleftarrow X^{(n)} \longrightarrow} \\ \hline & p_{11} & p_{12} & p_{13} & \cdots \\ & p_{21} & p_{22} & p_{23} & \cdots \\ & p_{31} & p_{32} & p_{33} & \cdots \\ & \cdots & \cdots & \cdots & \cdots \\ \hline \end{array}$$

P referred as the transition probability matrix of the Markov chain. The P is a stochastic matrix, i.e. a square matrix with non–negative elements and unit row sums.

VI. CONCLUSION

This paper covers different types of possible classification available in literature. A good scheduling approach requires a good balance between achieving fairness across users optimizing throughput and other system measures in a distributed system. The system must often choose between balancing fairness and balancing load when placing jobs among different processors as well as reducing the scheduling overhead by overlapping scheduling decisions with other operations. Distributed systems have different scheduling algorithms but all algorithms have their own deficiency and restrictions. In this research survey paper proposes to study probabilistic-based stochastic Markov chain model to improve the performance of different scheduling schemes. This research will be proposed new approaches for scheduling algorithm which helps to improve the efficiency of systems using model-based study in which assuming the random movement of schedulers over various processes with their processors and the various queues. The proposed work will be useful for the distributed system designers by incorporating some new schemes in currently available scheduling algorithms. It is also possible that some algorithms can be merged to get the optimized performance of distributed system where the performance of different scheduling schemes will be analyzed under probability based models. Simulation study will support this proposed discussion for setting up approaches and trend. After surveying and reviewing the available literature that explored some important issues and challenges associated with scheduling algorithms. This research work will be helpful to measures scheduler's performance in distributed systems.

VII. REFERENCES

[1]. C. Arpaci-Dusseau, "Implicit Coscheduling: Coordinated Scheduling with Implicit Information in Distributed Systems," ACM Trans. on Computer Systems, Vol.19, No. 3, pp.283-331, Aug. 2001.

[2]. P. Barcaccia, M.A. Bonuccelli, and M. Di Ianni, "Complexity of Minimum Length Scheduling for Precedence Constrained messages in Distributed Systems," IEEE Trans. Parallel and Distributed Systems, Vol.11, No.10, pp. 1090–1102, Oct. 2000.

[3]. Lampson, M. Abadi, M. Burrows, and E. Wobber, "Authentication in distributed systems: Theory and practice," ACM Trans. Computer Systems, Vol.10, No.4, pp 265-310, Nov. 1992.

[4]. V. M. Lo, "Heuristic Algorithms for Task Assignment in Distributed Systems," IEEE Trans. Computers, Vol. 37, No.11, pp. 1384-1397, 1988.

[5]. F. Petrini and W.-C. Feng, "Scheduling with Global Information in Distributed Systems," Proc. 20th Int'l Conf. Distributed Computing Systems, pp. 225 – 232, April 2000.

[6]. S. Srinivasn and N. K. Jha, "Safty and Reliability Driven Task Allocation in Distributed Systems," IEEE Trans. Parallel and Distributed Systems, Vol.10, No.3, pp. 238-251, Mar. 1999.

[7]. H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and Low-complexity Task Scheduling for Heterogeneous Computing," IEEE Trans. Parallel and Distributed Sys., Vol.13, No.3, Mar. 2002.

[8]. International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-3, July 2012, Overview of Scheduling Tasks in Distributed Computing Systems, O. M. Elzeki, M. Z. Rashad, M. A. Elsoud

[9]. Ramya S Gowda, "Qualitative Study on the efficiency of Load balancing algorithms in Cloud Environment," IOSR Journal of Computer Engineering (IOSR-JCE), Vol. 16, No. 6, Ver. VIII, pp 09-12, Nov – Dec. 2014.

[10]. Thomas l. Casavant and jon g. Kuhl, "A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems, IEEE Transactions On Software Engineering, Vol. 14, No. 2, pp 141-154,February 1988.

[11]. Medhi, J. Stochastic processes, Ed. 4, Wiley Limited (Fourth Reprint), New Delhi. 1991.

[12]. Silberschatz, A., Galvin, P. and Gagne, G. Operating System Concepts, International Student Version, Ed.8, India, John Wiley and Sons, Inc. 2010.

[13]. A. S. Tanenbaum and M. V. Steen, Distributed Systems: Principles and Paradigms, Low price edition, Pearson Prentice Hall.

[14]. Chapin, Steven J. and Weissman, Jon B, "Distributed and Multiprocessor Scheduling" (2002). Electrical Engineering and Computer Science. Paper 40.

[15]. X. Evers, Thesis on "A literature study on scheduling in Distributed system", submitted on Department of Mathematics and computing operating system and distributed systems group Delft University of Technology, The Netherlands, Oct.1992

[16]. Dongning Liang, Pei-Jung Ho, Bao Liu, "Scheduling in Distributed Systems".

[17]. Waldspurger, C.A., Weihl, W.E. (1994). Lottery Scheduling: Flexible Proportional-Share Resource Management. First USENIX Symposium on Operating System Design and Implementation.

[18]. Waldspurger, C.A., Weihl, W.E. (1995). Stride Scheduling: Deterministic Proportional-Share Resource Management. Technical Report MIT/LCS/TM-528, Massachusetts Institute of Technology, MIT Laboratory for Computer Science.