A Survey on Scheduling Algorithms for Parallel and Distributed Systems

Rinki Tyagi and Santosh Kumar Gupta

Abstract Task scheduling plays a vital role in distributed computing. It enhances the performance of the system as it minimizes the overall execution time and reduces overhead problems like communication delay by allocating suitable task to appropriate processor. Different scheduling techniques are discussed in this paper which are employed for task scheduling. Taxonomy of hierarchical classification is discussed for concurrent system and further several task scheduling algorithms are described on the basis of dependency and approach used such as static or dynamic for low throughput and efficient performance.

Keywords Task scheduling · Distributed computing · Real-Time scheduling

1 Introduction

Due to advancement of hardware and software technologies, development of parallel and distributed system for database, real-time applications are also increased. But it leads to a problem of scheduling different tasks on various processing elements in such a way that performance metrics, such as execution time, resource utilization, throughput, and response time, should be satisfied [10]. Scheduling is a type of resource and task allocation problem [11]. Scheduling can be defined as "A set of tasks T can be executed on P processors by some optimization criteria C" [11]. The goal of scheduling is to allocate different tasks to processors with the aim of increasing execution speed, reducing the runtime of tasks, minimizing communication delay, communication cost, and priority problem. In distributed scheduling, whole task is divided into sub-tasks and assigned to

R. Tyagi (✉) · S. K. Gupta
Department of Computer Science and Engineering, Krishna Institute of Engineering and Technology, Ghaziabad 201206, India
e-mail: rinkitg61@gmail.com

S. K. Gupta
e-mail: santoshg25@gmail.com

A. Mishra et al. (eds.), *Silicon Photonics & High Performance Computing*, Advances in Intelligent Systems and Computing 718,
https://doi.org/10.1007/978-981-10-7656-5_7

several processors, thus they execute more quickly as compared to single processor. While scheduling tasks, prespecified precedence is obeyed among different tasks [12]. The good scheduler should be General Purpose: it is applicable for all types of applications with different attributes to execute them in an efficient way; Efficient: it enhances the overall performance of systems and reduced overhead problems; Fair: It should be fair. For example, when there are many tasks to execute, scheduler should maintain load through load balance; Transparent: result should not be affected by local or remote site executions; and Dynamic: A good scheduler should respond to local changes and it should fully exploit all resources available to it.

Before describing the approaches, techniques, and algorithms that are used for task scheduling, the role of scheduler and how scheduling of tasks is to be done on different workstations or processors is discussed. There are two inter-dependent steps of task scheduling: one is allocating the tasks to processors (space sharing) and another is to schedule it with time (time sharing). When job is assigned to a system, the complete information is noted by the system, i.e., CPU load, memory size, etc. Meanwhile, system also maintains a status table of different workstations over which job is to be scheduled. After this, job is divided into several components and assigned to different workstations on the basis of the status table information. Synchronization process is needed when tasks are executing on different workstations. Besides this, mechanism of process migration is also introduced during task scheduling. For example, if any processor is highly loaded, the job migrates lightly loaded processor to improve the overall performance of system [12].

2 Related Work

The concept of "Grid Computing" in distributed system is used to perform users tasks online at any place and at any time [1]. But it leads to a problem of uncertainty in scheduling overhead and response time during continuous task arrival and their execution process. To overcome this problem, MDP (Markov Decision Process) is introduced where it allocates the task arrival and execution pattern without uncertainty. In [2], author argued the most general problem of process distribution. To solve this, a modified version of AO* algorithm using statistical data as a heuristic function is used to find those processors which can execute the processes in most efficient way. In [3], author has proposed a general technique to design and implement priority-based resource scheduling in flow graph-based DSPS with priority metadata. Experimental results show the effectiveness of this approach. In [4], author discussed the problem of task scheduling for multi-core CPU and to solve this multistep, scheduling algorithm is proposed. A clustering algorithm that is based on SCAN to find clusters in a network in order to find parallelism is used to decrease the computation of scheduling. In [5], an adaptive distributed scheduling algorithm is introduced for multi-place parallel computation. A combination of

remote spawns and remote work steals is used to reduce the overhead problems in scheduling and helps to maintain load balance with maximal affinity. In [6], the author addresses the problem of load balancing optimization. As the load is distributed randomly to all processors in the distributed system without any fixed affinity, the goal is to minimize the computation time while distributing the load with the limited communication delay. For each load transfer, sending and receiving sites are maintained to obtain optimal delivery and load transmission. In [7], author proposed long-term CPU load prediction method, namely process search method also called runtime prediction-based method which predicts long-term CPU load more accurately than the conventional methods. A prediction module selection using neural network is proposed that selects an appropriate prediction method according to the change in the state of CPU load and shows improvement in prediction accuracy. In [8], author addressed the problem of producing the optimal schedule using genetic algorithm which minimizes peak load and communication cost and maximizes load balancing and average CPU utilization. In [9], author introduced an integrated BOA approach to overcome the efficiency problem while solving the NP scheduling problems. In [35], a taxonomy of load sharing is introduced that includes source initiative and server initiative approaches for evaluation of performance.

3 Issues in Multiprocessor Scheduling

There are many issues during scheduling for multiprocessor system [11]. First is distinction between Policy and Mechanisms. Mechanism is the ability to do any action while Policy enables to do with that mechanism. For example, automobiles have the ability to travel with the speed of 160 km/h (mechanism), while legal speed is set below 160 km/h (Policy). Second, distinguish between distributed and parallel system. If the communication between symmetric multiprocessors is through shared memory, it is parallel but if communication among network of workstations, it is distributed. Next is the distinction between types of scheduling. One is local scheduling and another is global. Global scheduling is done before local scheduling [11] although migration changes the global mapping when task moves to a new processor. During migration, the system stops the tasks, saves its state, and transfers that state to new processor and then restarts the task. Due to migration, several overheads occur. Local scheduling decides which an appropriate set of tasks at a processor executes next on that processor. *Then*, there are two choices at that time of task allocation. Either several processors are assigned to a single job or several tasks are assigned to a single processor. Few processors assigned to a single job for execution; then, it is called as *space sharing* and it is a function of global scheduling. *Time sharing* is a function of local scheduling. In this, several tasks are assigned to a single processor for its execution.

4 Scheduling Techniques

In Fig. 1, scheduling techniques are categorized into two types: local and co-scheduling. Local scheduling consists of predictive that easily adopts new architectures and proportional sharing that executes at a uniform rate. Three co-schedulings are discussed in which gang co-scheduling is simple than other two [12].

4.1 Local Scheduling

In local scheduling, an individual site schedules the processes assigned to it to improve the overall performance [12]. Local scheduler requires global information for maximizing the performance of system. Many new scheduling techniques are developed such as proportional sharing schedule approach and predictive scheduling. A module that has advance reservation capability is discussed that possessed local scheduling [13]. Local scheduling is used in opportunistic routing [14], in which wireless topology is broken down into several sub-graphs and end-to-end transmission of different forwarders is done that proved more efficient in wireless network compared to traditional routing.

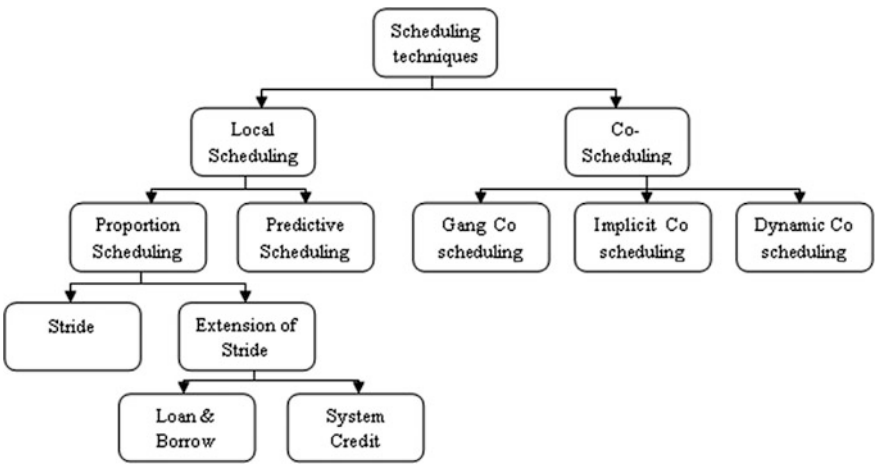

Fig. 1 Scheduling techniques

4.1.1 Proportional Sharing Schedule

To solve the problem incurred during traditional priority-based schedulers which take long time to allocate processors, a scheduler that allocates jobs to all processors in a fair manner is required [12]. In this, resource consumption is proportional to all jobs that are allocated. In [15], a notion of "Pessimism" is introduced in proportional sharing for improving performance as well as overcome error problem and meet the deadlines of various real-time applications. Through proportional sharing, scheduling is done at a uniform rate [16]. It is also beneficial for both real and non-real-time processing.

Stride Scheduling

It is an example of proportional sharing that shows how allocation of jobs and consumption of resources is done in fair manner to a single processor when many users have to execute their tasks. In this scheduling, users hold a number of tickets that are in a proportion of resources of competing users and have a time interval called stride which is inversely proportional to allocation of tickets that helps to decide how fast it comes in a usable state [12]. Also, a pass is associated with each user and the user with minimum pass is scheduled in that time interval and incremented by job stride. Its evaluation can be done by two ways: one is by using simulations and another is by implementing prototypes for Linux Kernel [17].

Extension to Stride Scheduling

Stride scheduling is only used for CPU-bound jobs, not for interactive and I/O intensive jobs [12]. In I/O intensive jobs, there is a need to improve response time and throughput rather than focusing on resources for competing users. For this, extension of stride scheduling is used. It uses two approaches: one is loan and borrow and another is system credits. Loan and borrow: In this approach, many exhausted tickets are distributed among the users. And, if any user wants to exit from system for a while, then another user borrows its ticket; otherwise, it would be an inactive ticket. System credits: It is an approximated approach and does not have any type of overhead and is easy to implement [18].

4.1.2 Predictive Scheduling

Predictive scheduling provides adaptively, intelligence and proactive type of scheduling to the system so that it can perform well in any type of environment. It can be easily embedded in new type of architectures. It is divided into three major components: H-Cell, S-Cell, and allocator [12]. In [19], a scheduling problem is investigated for minimizing completion time through random machine breakdowns.

Through predictive scheduling, preferences of users are also considered [20] and it is done through fuzzy techniques that not only use imprecise information but also views of users are to be considered.

4.2 Co-scheduling

Co-scheduling introduced in [21] is used for scheduling the interactive activities such that all jobs executed simultaneously on their workstations. In [22], flexible co-scheduling is introduced that addresses the problem of internal and external fragmentations. In this, scheduling is based on synchronization among processors and also on load balancing requirements. Co-scheduling was used for proper utilization of resources and covers all problems identified in multi-core system [23]. Several key issues are discussed in co-scheduling algorithm for clusters [39]. For this, co-ordinate scheduling in time-sharing clusters is done through a genetic framework.

4.2.1 Gang Co-scheduling

In gang co-scheduling, a job is referred as gang and its members are referred as gang members. They are allocated to a class whose one processor is assigned by one gang member and executed in parallel. A local scheduler exists in gang co-scheduling which has its own policies. When timestamps finish, it precepts all job members and assigns another job to that class [12]. The main disadvantage of this scheduling is its centralized control. It creates bottleneck when load is heavy. In [24], gang co-scheduling technique is combined with backfilling for addressing the problem of space sharing in scheduling. In [37], flexible co-scheduling is done through gang co-scheduling by reducing fragmentation problem and improves efficiency by providing the processor to each job through preprocess-based classification.

4.2.2 Implicit Co-scheduling

Implicit co-scheduling is a type of time-sharing communication process. It has a local scheduler that schedules the processes individually. It makes individual decisions while executing the job members rather following centralized policy and deals with the problem of gang scheduling [12]. It uses communication and synchronization within an application. It schedules both fine- and coarse-grained parallel application [25].

4.2.3 Dynamic Co-scheduling

Dynamic co-scheduling is used to make decisions on the arrival of messages. A schedule is made when a message is arrived to any process and no need for explicit identification to specify the process that needs co-scheduling. Dynamic co-scheduling reduces response time up to 20% over implicit [26] and is more robust and effective.

5 Taxonomy of Scheduling Algorithms

In Fig. 2, hierarchal classification of different scheduling algorithms is given. This classification can be used for categorizing different types of strategies that are used for allocating tasks to different processors and also for categorizing resource management system especially process management system.

Local versus Global: Local scheduling decides the appropriate set of tasks to execute next on the processor. Global scheduling is done before local scheduling and is used to allocate the tasks within the systems [27]. A new concurrency control criterion is proposed for local and flexible transactions execution through global scheduling in heterogeneous distributed environment [36].

Static versus Dynamic: Static algorithms are used for scheduling when information available at compile time. On the other hand, dynamic algorithm takes all these factors into account during execution time [27].

Optimal versus Sub-optimal: When information about the states of each processor and the types of resource needs is known, the algorithm is called as optimal otherwise sub-optimal.

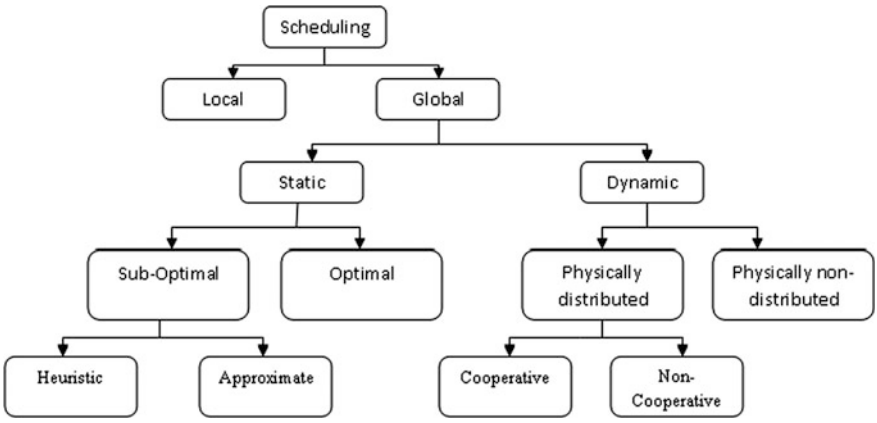

Fig. 2 Taxonomy of scheduling algorithms

Distributed versus Non-distributed: Physically non-distributed algorithms are centralized. A single processor decides task allocation. On the other hand, in physically distributed algorithm, all processors decide task placement [11].

Approximate versus Heuristic: Heuristic algorithms use various guidelines for scheduling such as allocate jobs to a processor with heavy inter-task communication, while approximate algorithms use same method that is used by optimal solutions but within a accepted range [27].

Cooperative versus Non-cooperative: In non-cooperative algorithms, each processor is independent of making choices from other processors for scheduling while in cooperative, all processors co-ordinate with each other to achieve a goal [28].

6 Task Scheduling Algorithms

While performing task scheduling, the performance of algorithms is affected by choosing various strategies. Different task scheduling algorithms are shown in Fig. 3.

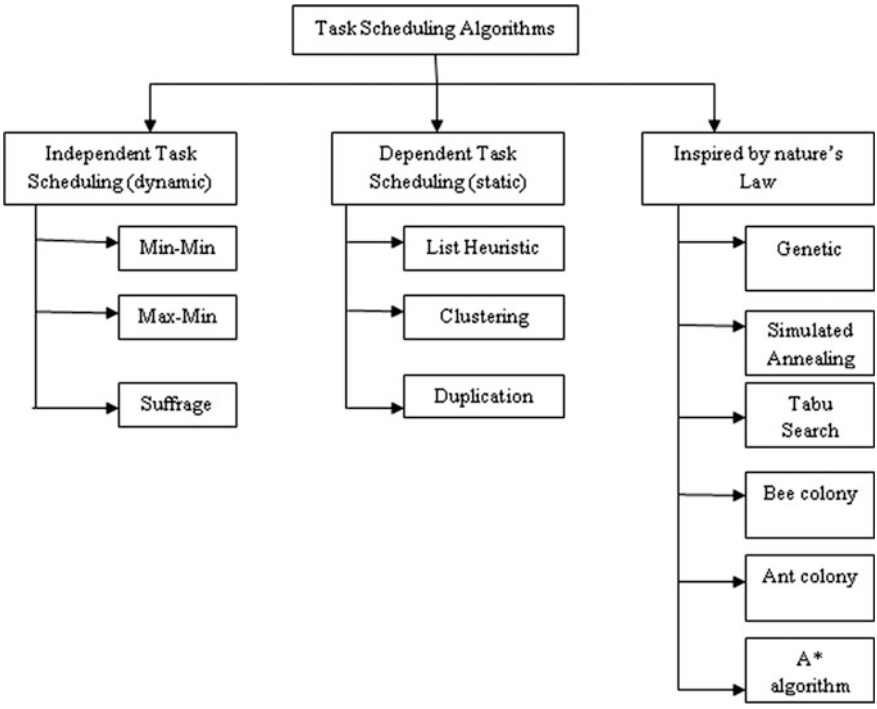

Fig. 3 Task scheduling algorithm

6.1 Independent Task Scheduling

This is used for dynamic scheduling. Some of the heuristic-based static algorithms also use this for execution of cost estimates [28]. *Max-Min* is somehow similar to *Min-Min* algorithm. In this, a set of M tasks of low completion time are selected from a set U of unmapped tasks. Then, the task of high execution time from M set is assigned to a processor. The newly mapped task is removed from set U and this process repeats until all the tasks of set U are not mapped. *Min-Min* is a type of scheduling which is applied for a set of tasks without dependencies onto a heterogeneous system. If there is a set of S tasks, then *Min-Min* selects a task with minimum execution time and it would be scheduled on that processor which has minimum completion time. It is one of the fastest algorithm and easy to implement. *Suffrage* is based on suffrage value and is calculated as the difference between its first best completion time and its second best completion time.

6.2 Dependent Task Scheduling

In [28], DAG (Direct Acyclic Graphs) is used to denote task precedence graph where node represents graphs and edge represents precedence order. The main objective of dependent task scheduling is to minimize the make-span time. But some problems are NP-complete which does not produce optimal results. It has three types of algorithms.

(i) *List Heuristic Algorithm*: It is based on the idea that some priorities are assigned to tasks and then repeatedly executes the following two steps until whole graph is not scheduled. First, remove the first node from list of scheduling. Then, assign the node to a processor that allows the earliest start time. Table 1 shows different algorithms for list heuristic.

(ii) *Clustering Heuristic Algorithm*: Clustering is an efficient process for both parallel and distributed systems. It is used to minimize communication delay

Table 1 List heuristic algorithms

Type	Description
HEFT	It is Heterogeneous Earliest Finish Time algorithm. At each step, it selects a task that has higher upward rank and with maximum distance between current and exist nodes
Chaining	It does not allow duplication of tasks and distribute the tasks among all processors
HLFET	Highest Level First with Estimated Times algorithm is one of the simplest scheduling algorithms. It calculates b-level (bottom level), and then makes a ready list in descending order (contains only entry nodes). It schedules first node of ready list to a processor with earliest execution
FCP	It is Fast Critical Path algorithm and tries to reduce the complexity of algorithm

Table 2 Clustering heuristic algorithms

Task clustering phase	DSC: It is Dominant Sequence clustering algorithm. The scheduled DAG has a critical path called Dominant Sequence and used to distinguish it from scheduled clustered DAG critical path
Post-Clustering Phase	LB: Load Balancing is used to define the overall workload as sum of execution time of all tasks. It merges a pair of clusters C1 and C2 such that C1 is the cluster that has minimum workload and C2 is a cluster that minimum workload with communication edge to all clusters CTM: It is Communication Traffic Minimization and it is defined by using two terms C1 and C2. These are the sums of communication time of edges from C1 to C2 and from C2 to C1 RAND: It is random algorithm and make random pairs of clusters. First, assign a cluster with highest communication traffic. Then, it selects an unassigned cluster of highest communication traffic through assigned cluster. Then repeat step(b) until all processors are assigned

by clustering the same level of tasks into one cluster and assign resources to a single cluster. Table 2 shows that first phase is task clustering phase in which clusters are formed by portioning the tasks and same levels of tasks are clusters into one group. Second phase is post-clustering phase which is used to refine the cluster to get final task to resource map [28].

(iii) *Duplication Heuristic Algorithm*: It is basically used for homogenous system. For this, various duplication-based scheduling algorithms are introduced. The main logic behind this is to fully utilize the idle resources by duplicated tasks.

6.3 Scheduling Inspired by Nature's Law

Genetic algorithm, simulated annealing, Tabu search, etc., are scheduling methods that are inspired by nature's law. They all are used to solve standardized scheduling problems. A long runtime problem will occur when they are used in practical application. Then, optimization algorithms are used to overcome the problem of long runtime [28]. Genetic algorithm optimizes the total flow time and makes span in task scheduling and is more robust. A simulator package of grid is used for large problems and to evaluate performance of GA [29]. Tabu search is also an optimization algorithm that gives optimal or close to optimal results for different types of scheduling [30]. Tabu search is used for bi-objective optimization problems [31]. Simulated annealing is to solve optimization problems for separating continuous and batch chromatographic systems under various conditions such as gradient and isocratic [32]. It states that if temperature is lowered sufficiently, then the solid reached to optimal state, where temperature is completion time and change in

temperature is task mapping. And if temperature is increased, then it accepts that "worse" state because it provides a way to escape from local search [28]. Ant colony optimization is best suited for TSP problems. Ants act as an agent and finds the best solution by parallel search. Many ants generate many cities and their corresponding paths, and best suited path (shortest path between source and destination) is selected [33]. Bee colony optimization is best suited for combinatorial optimization problems. When independent tasks are scheduled in grid environment, then it is a NP-hard problem. Many approaches are used for this but bee colony is more efficient as it reduces the finish time and delays during execution time [34]. In [38], different applications are discussed for BCO. A* algorithm is a type of best search algorithm for tree search. It initially starts with null solution and then expanded through partial solution for complete solution by allocating different tasks to suitable processors [28].

7 Analysis Work

Table 3 shows the different scheduling techniques, task scheduling algorithms, and approach that can be used for scheduling tasks.

Table 3 Analysis work

Techniques	Description	Approach	Algorithms	Advantage
Proportional share	Resource consumption is proportional for all jobs	Static	List heuristic	Does not allow duplication of tasks
			Clustering heuristic	Minimize communication delay
Predictive	Adaptive, intelligent, and proactive techniques that perform well in any type of environment		Duplication	Fully utilized the idle resources
Gang co-scheduling	Dynamically scalable and utilized time sharing and space sharing simultaneously	Dynamic	Min-Min	Fastest algorithm and easy to implement
Implicit co-scheduling	A local scheduler that schedules the processes individually rather centralized policy		Max-Min	Removes the penalties occurred through longer execution time
Dynamic co-scheduling	Used to make the decisions on the arrival of messages		Suffrage	Finds minimum completion time

8 Conclusion

In distributed environment, task scheduling plays a vital role. To improve, the performance of system scheduling of tasks can be done in an efficient manner on several processors. A good scheduler must be efficient, dynamic, transparent, general purpose, and fair. Different scheduling techniques are employed for both parallel and distributed systems that are used for proper utilization of resources and for improving system performance. Two scheduling techniques, i.e., local and co-scheduling, are described in which co-scheduling technique is better as it is used for the concurrent execution of the parallel systems and we conclude that gang co-scheduling is the simplest one among three. A hierarchal classification of scheduling algorithms is done and on this basis task scheduling algorithms are described. Further, a comparative analysis for different scheduling techniques has been done.

9 Future Work

Further, more comprehensive survey can be made on real-time scheduling and scheduling inspired by nature's law and these can be compared with other scheduling algorithms.

References

1. Zhao T, Zheng X (2014) Proactive scheduling in distributed computing—A reinforcement learning approach. J Parallel Distributed Computing, pp 2662–2672
2. Gyire T (1995) A distributed process scheduling algorithm based on statistical heuristic search. IEEE International Conference
3. Bellavista P (2014) Priority-based resource scheduling in distributed stream processing systems for big data applications. Utility and Cloud Computing (UCC). In: IEEE/ACM 7th International Conference on IEEE
4. Yamazaki H, Konishi K, Shin S, Sauada, K (2013) Multistep scheduling algorithm for parallel and distributed processing in heterogeneous systems with communication costs. Mathematical Problems in Engineering
5. Narang A, Srivastava A, Shyamasundar RK (2013) High performance adaptive distributed scheduling algorithm. In: Parallel and distributed processing symposium workshops & PhD Forum (IPDPSW), 27th international IEEE
6. Haddad E (1994) Real-time optimization of distributed load balancing. Proceedings of the second workshop on IEEE
7. Sugaya Y, Tatsum H, Kobayashi M, Aso H (2008) Long-Term CPU Load Prediction System for Scheduling of Distributed Processes and its Implementation. Advanced Information Networking and Applications, 22nd International Conference on. IEEE
8. Wang PC, Korfhage W (1995) Process scheduling using genetic algorithms. Parallel and distributed processing, proceedings seventh ieee symposium on IEEE

9. Qiang L, Xiao TY (2006) Cooperated Bayesian algorithm for distributed scheduling problem. Frontiers Electr Electron Eng China, pp 251–254
10. Shirazi BA, Hurson AR, Kavi KM (1995) Introduction to scheduling and load balancing in parallel and distributed system. Wiley-IEEE computer society press
11. Chapin SJ, Weissman JB (1996) Distriibuted and Multiprocessor scheduling. published ln: ACM computing survey(CSUR), 28:233–235
12. Dongning L, Ho PJ, Liu B (2000) Scheduling in distributed system
13. Nakada H, Kishimoto M, Kudoh, T, Tanaka Y, Sekiguchi S, Takefusa A (2006) Design and implementation of a local scheduling system with advance reservation for co-allocation on the grid. In: Computer and information technology, sixth IEEE international conference
14. Li Y, Liu YA, Li L, Luo P (2009) Local scheduling scheme for opportunistic routing. In: Wireless networking conference IEEE, pp 1–6, (2009)
15. Regehr J.: Some guidelines for proportional share CPU scheduling in general purpose operating system. In: Work in progress of the 22nd IEEE Real -time system symposium (RTSS) (2001)
16. Stoica I, Wahab HA, Jeffay K, Baruan SK, Gehrke JE, Plaxton CG (1996) A proportional share resources allocation algorithm for real time, time shared systems. IEEE, pp 288–299
17. Gu W, Carl A, Weihl WE (1995) Stride scheduling: deterministic proportional share resource management. Massachuesetts Institute of Tech, laboratory for computer science
18. Koshy R (2014) Scheduling in distributed system: a survey and future perspective. Int J Adv Technol Eng Sci
19. Xing Z, Zhijon C, Yugeng X (2007) The applications of predictive scheduling algorithm for single machine problem. In: Control conference IEEE, 810–814
20. Sauer J, Chua TJ (2014) Fuzzy predictive and reactive scheduling in soft computing for business intelligence. Springer Berlin Heidelberg, pp 281–297
21. Gupta A, Taucker A, Urushibaras S (1995) The impact of OS scheduling policies and synchronisation methods on performance of parallel applications. In: SIGMETRICS perform evaluation review
22. Frachtenberg E, Feitelson DG, Petrini F, Fernandez J (2005) Adaptive parallel job scheduling with flexible coscheduling. IEEE, pp 1066–1077
23. Schonhers JH, Lutz B, Richling J (2012) Non-Intrusive co-scheduling for general purpose operating system. Springer Berlin Heidelberg, pp 66–77
24. Zhang Y, Franke H, Moreira JE, Sivasubramaniam A (2000) Improving parallel job scheduling by combining Gang scheduling and backfilling techniques. IEEE, pp 133–142
25. Anglano C (2000) A Comparative evaluation of implicit coscheduling strategies for network of workstations. IEEE, pp 221–228, 1 Aug–4 Aug
26. Sobalvarro PG, Scott P, Weihl EW, Andrew AC (1998) Dynamic coscheduling on workstations clusters in Job Scheduling Strategies for Parallel Processing. Springer Berlin Heidelberg, pp 231–256
27. Casavant TL, Kuhl JG (1988) A taxonomy of scheduling in general-purpose distributed computing systems. Software Engineering, IEEE Transactions on 14(2):141–154
28. Shahsavari M, et al (2004) Task scheduling policies in general distributed systems: a survey and possibilities
29. Carretero J, Xhafa F (2006) Use of genetic algorithm for scheduling jobs in large scale grid application. ISSN 1392–8619 UKIO Technologinis IR Econominis Vystymas, pp 11–17
30. Glover F (1990) Tabu search: a tutorial. pp 74–94
31. Xhafa F, Carretero J, Dorronsoro B, Alba E (2012) A tabu search algorithm for scheduling independent jobs in computational grids. Comput Inform 28:237–250
32. Kaczmarski K, Antos D (2006) Use of simulated annealing for optimization of chromatographic separations. Acta Chromatographica 17
33. Darquennes D (2005) Implementation and Applications of Ant Colony Algorithms. Facultées Universitaires Notre-Dame de la Paix, Namur Institute Informatique

34. Mousavinasab Z, Entezarii ME, Movaghar A (2011) A bee colony task scheduling algorithm in computational grids. Digital Information Processing and Communications, Springer, Berlin Heidelberg
35. Wang Y Load sharing in distributed system. IEEE
36. Zhang A, Noidine M, Bhargava B (2001) Global Scheduling for flexible transactions in heterogeneous distributed database systems. 13(3):439–450
37. Frachtenberg E, Feitelson DG, Petrini F, Fernandez I (2003) Adaptive Parallel job scheduling with flexible co-scheduling. Parallel and Distributed processing, 10 pp, IEEE
38. Karwan KS, Choudhary S, Sharma K Applications of artificial bee colony optimization techniques. pp 1660–1664, IEEE, (2015)
39. Agarwal S, Yoo AB, Choi GS, Nagar S (2003) Coordinated co-scheduling in time sharing through a genetic framework. pp 84–91, IEEE, (2003)