A Review of Dynamic Scheduling Algorithms for Homogeneous and Heterogeneous Systems

Mahfooz Alam, Asif Khan and Ankur K. Varshney

Abstract The dynamic scheduling algorithms are widely used to evaluate the performance of homogeneous and heterogeneous systems in terms of QoS parameters such as scheduling length, execution time, load imbalance factor and many more. Over the time, many dynamic scheduling policies were introduced which are designed to achieve their goal such as efficient utilization of process elements, minimization of resources idleness, or determining the total execution time. In this paper, we analyzed different aspects in dynamic scheduling algorithm and numerous issues in various levels of the homogeneous and heterogeneous systems.

Keywords Parallel processing · Multiprocessor system · Static and dynamic scheduling · Heterogeneous and homogeneous systems

1 Introduction

Scheduling is a process of comparable tasks to the resources at specific times. A scheduling algorithm (SA) is used to find out a schedule for a set of task on the bases of task's deadline and recourse requirements specified. The prospective speedup of applications has motivated the extensive use of multiprocessors in current years [1–5]. In multiprocessor, scheduling algorithms developed for uniprocessor can be applied if we consider each core of the multiprocessor as an

M. Alam (✉)
Department of Computer Science, Al-Barkaat College of Graduate Studies, Aligarh, India
e-mail: mahfoozalam.amu@gmail.com

A. Khan
University of Electronic Science and Technology of China, Chengdu, China
e-mail: asif05amu@gmail.com

A. K. Varshney
Institute of Technology & Management, Aligarh, India
e-mail: ankur.varshn@gmail.com

S. K. Muttoo (ed.), *System and Architecture*, Advances in Intelligent Systems and Computing 732, https://doi.org/10.1007/978-981-10-8533-8_8

isolated core, which is a uniprocessor [6–8]. However, in multiprocessor scheduling the complexity of varying the execution of different tasks on multiple cores does not interfere with each other and also determining which tasks should be given to a certain core increases the complexity greatly as compared to uniprocessor scheduling. The two main approaches for SA on multiprocessors are global scheduling and partitioning scheduling [9–12]. But the point of emphasis of scheduling is always to reduce the execution time, schedule length (or makespan), and maximization of speedup. Besides this, it is clear that the multiprocessor scheduling suffers from NP-complete problem in its many variants excluding some interpreted conditions. The SA can be categorized as *off-line* (static or deterministic) SA and *on-line* (dynamic or nondeterministic) SA.

In off-line scheduling, all scheduling decisions are taken before the system starts and the scheduler has an exact knowledge of the all task properties and behaviors are therefore needed. During runtime, the tasks are executed in a predetermined order. The static task scheduling problem is known as NP-complete [2]; that is, task list is not updated with new ordering at runtime. There are two kinds of static scheduling algorithms (SSA): one is Heuristic Based (HB) and another is Guided Random Search-Based (GRSB). HB algorithm can be further subdivided into three categories: List Scheduling, Cluster Scheduling, and Task Duplication-Based Scheduling (TDBS) algorithms.

An on-line scheduling algorithm takes its scheduling decisions during the operation of the application. In other words, on-line scheduling tasks can be reallocated to other processors during runtime [3]. On-line scheduling is supple and very faster than the static scheduling algorithms. The key point of on-line scheduling is to map tasks in parallel on the multiprocessor and arrange their execution so that a minimal makespan is given in the bound of task priority necessities. Dynamic scheduling algorithm can be further subdivided into three

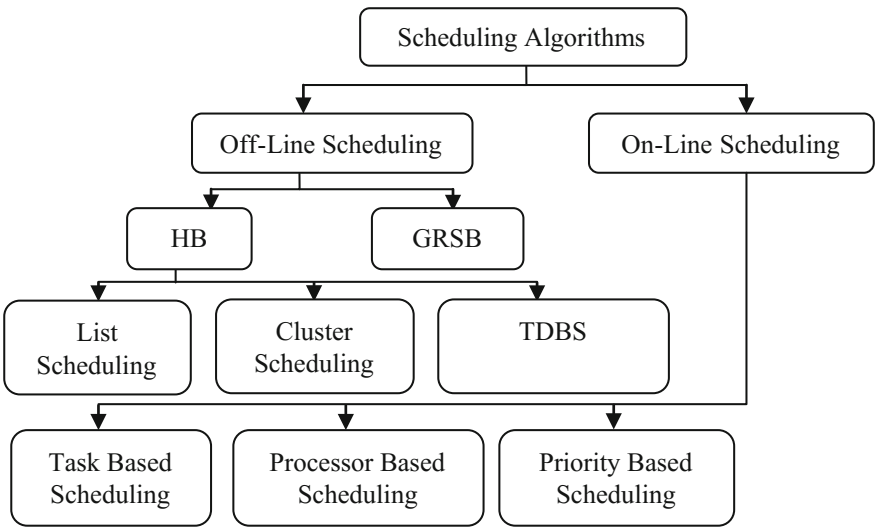

Fig. 1 Taxonomy of scheduling algorithms

kinds: Task-Based Scheduling (TBS) Algorithms, Processor-Based Scheduling (PBS) Algorithms, and Priority-Based Scheduling (PRBS) Algorithms (Fig. 1).

2 Homogeneous and Heterogeneous Systems

A multiprocessor system is either homogeneous (HM) or heterogeneous (HT). Heterogeneous and homogeneous systems are explained as under.

2.1 Heterogeneous System

Heterogeneous systems (HTS) consist of different processors that are capable of performing different tasks. HTS become the latest thing in both client and cloud. Scheduling a parallel program is critical step in inefficiently harnessing the complexity power of a HTS. Heterogeneity in parallel systems introduces an extra degree of complexity to the scheduling difficulty that is variable speed of processors. The difficulty of the problem rise when task scheduling is to be done in heterogeneous nature, where the computational nodes in the system may not be equal and different amount of time to execute the same task. Heterogeneity can be proposed in two types, namely functional-level and performance-level heterogeneity [4, 5]. All tasks may not be running on all functional units. One of the key concepts of HTS is schedule problem. To achieve high performance, the efficient scheduling of the tasks of a function is needed.

2.2 Homogeneous System

Homogeneous systems (HMS) consist of the processors identical in terms of their functionality [2]. The characteristics of homogeneous multiprocessor system are replication effect, memory dominant, less performance, data parallelism, shared memory and dynamic task mapping. HMS is all exactly the identical: cache sizes, equivalent frequencies, functions, etc., while every core in a HTS may have a unlike function, memory model, frequency, etc. So, it is easier to develop software for HMS. However, in homogeneous processor, system typically requires additional registers for "special instructions" such as Single Instruction Multiple Data likes MMX, SSE, while a HTS can implement unlike types of hardware for unlike instructions/uses.

This paper is classified as follows: Sect. 2 presents HMS and HTS, and Sect. 3 reviews the dynamic scheduling algorithms. Section 4 presents comparison of algorithm with factors, and Sect. 5 presents conclusions and suggests possible avenues for future research.

3 Review of Dynamic Scheduling Algorithms (DSA)

In dynamic scheduling the decision about how to schedule tasks priorities which are either assigned dynamically or statically? Many research works investigated the DSA problem in different aspects and the problems faced in various levels of the HMS and HTS. Then important contributions of DSA in HMS and HTS are discussed in this section.

3.1 The Earliest Time First (ETF) Algorithm

This algorithm is alike to the Modified Critical Path Algorithm; it considers fixed node priorities and assumes only a limited number of processors. Yet, a node with superior priority may not automatically obtain scheduled earlier than the nodes with inferior priority [13]. After the computation of earliest begin times (EBT) for all the prepared nodes at each step, ETF algorithm selects the one with the lesser begin time, which is calculated by investigating the begin time of the node on all processors thoroughly. The ETF is stated below.

1. At every node, static b-level is calculated.
2. In the starting, the group of prepared nodes contains only the admission nodes. Repeat.
3. At every processor, for every node in the prepared group, the EBT is calculated.
4. Finally, joint the new prepared nodes to the prepared node group. Till all nodes are scheduled.

3.2 Dynamic Level Scheduling (DLS) Algorithm

This algorithm is alike to the Mobility Directed Algorithm; its uses as node priorities dynamically through allocating an attribute called dynamic level (DL) to unscheduled nodes at every scheduling step. The stepwise description of the DLS is given below.

1. At every node, static b-level is calculated.
2. In the beginning, the group of prepared contains only the admission nodes. Repeat.
3. The EBT is calculated at each processor for every node.
4. Pick the node processor couple that gives the biggest DL. All nodes are scheduled to the equivalent processor.
5. Finally, joint the new prepared nodes to the prepared group. Till all nodes are scheduled.

3.3 The Earliest Deadline First (EDF) Algorithm

EDF is a DSA that helps real-time operating systems in placing processes in a precedence line. When a scheduling incident arises the processor searches the line for finding out the process nearest to its limit. This process succeeds to be scheduled for execution. EDF is the finest scheduling algorithm on preemptive uniprocessor, in the following way: if a group of free jobs, each having its own arrival time, execution necessity and a limit, can be programmed in a manner that ensures all the jobs complete by their limit, the EDF will schedule this group of jobs so they all complete by their limit. With scheduling periodic processes that have limits equal to their stages, the utilization limit of EDF is 100%.

3.4 Online Scheduling of Dynamic Task Graph (OSDTG)

This algorithm stated that the OSDTG is more practical compared to conventional schemes and task graphs are subject to variation at execution time and interprocessor communications (IC) of static number (SN) of channels. Broadcast and point-to-point communication is applied in online scheduling. The key point of this algorithm is to decrease scheduling length [6].

3.5 Dynamic Load Balancing Using Task-Transfer Probabilities (DLBTTP)

This algorithm is based on the recent position of the system load; task move probabilities were calculated for every node in the system. These probabilities, represented by P_{nm}, are between a node n and each other node m. The computation and adjustment of the task move probabilities later than the move of every recently arrived task are at done at the end of every periodic interval [14].

3.6 Dynamic Task Scheduling (DTS) Algorithm

This algorithm has been proposed with a lower time complexity for homogeneous environment. This algorithm is based on DAG model, and the plan makes the entire parallel task complete at the feasible initial time; that is, response time (RT) of this parallel task is smallest [7].

3.7 DLS Algorithm with Genetic Operators (DLSAGO)

Dynamic Level Scheduling Genetic Algorithm (DLSGA) proposes to overcome the excess time for schedule. DLSGA uses a quantity known as dynamic level, which is the dissimilarity between the maximum total of calculated costs from task to a way out task, and the earliest begin run time on processor [8]. No tasks are scheduled between two earlier scheduled tasks by DLSGA. To reproduce offspring, the most favored and the fittest are selected.

3.8 Dynamic Task Graph Scheduling with Fault-Tolerant (FTDTGS)

An approach has been proposed to fault-tolerant execution of dynamic task graphs scheduled using work stealing. The work-stealing-based algorithm is applied when the data and metadata corresponding with a task get corrupted; then a task graph is mounted to enable recovery [10]. The algorithm was shown to be asymptotically optimal for graphs whose degree can be bound by a stable. In the lack of faults, the fault-tolerant edition was shown to not incur significant overheads compared to the non-fault-tolerant version.

3.9 DTS with Load Balancing (DTSLB)

DTSLB proposes to schedule a heterogeneous tasks dynamically on to corresponding processors in a distributed setup and load balancing which is a main problem in task scheduling is also measured. The nature of the tasks is free and non-preemptive. To determine the efficiency of the scheduling algorithm, a number of different tests have been performed [11]. The unique approache of this algorithm is solving the DTS using Hybrid Particle Swarm Optimization (HPSO).

3.10 Dynamic Load Balancing Using Genetic Algorithms (DLBGA)

DLBGA have been proposed for a static number of tasks; each having a task number and a length is arbitrarily generated and located in a task group from which tasks are allocated to processors [15]. As load balancing is performed by the centralized Genetic Algorithm (GA)-based method, begin with initializing, a population of probable solutions [16]. By using the sliding window technique it is achieved.

3.11 Parallel Genetic Algorithms for Heterogeneous (PGAH)

This PGAH engages a central scheduler, which has the processor lists and the task queue. The processors are heterogeneous in distributed system. The available network resources vary over time between the processors in distributed system. The possibility of every processor can vary over time. In distributed system the task which are indivisible, independent of all other tasks, arrive randomly and can be processed by any processor [17].

3.12 Global Scheduling for Mixed-Critically (GSMC)

In the context of multiprocessor mixed-criticality system, the fixed-priority algorithms can be applied globally [18]. The global Fixed-priority (FP) scheduling of mixed-criticality task sets (MCTS) are on HM multiprocessors. In this paper, there are two main ways in which an algorithm must be selected. Particularly, a priority assignment strategy to individual tasks and a feasibility testing given a specific priority assignment must be selected.

3.13 The Response Time Analysis of Global Fixed-Priority (RTAGFP)

This algorithm states a new analysis that improves over current state-of-the-art (SOA) response time analysis (RTA) by reducing its pessimism. Finally, how to improve the new response time analysis method by empirical tests with arbitrarily generated task sets [19, 20]. The RTA with Limited Carry-in (RTA-LC) [21] is the most accurate algorithm for RTA of global FP scheduling on multiprocessors. In this paper, first propose a new formula to bound the workload of carry-in jobs.

3.14 New Response Time Bounds for Fixed Priority (RTBFP)

This algorithm improves analysis precision response time for sporadic tasks on multiprocessor systems where the limits of tasks are within their periods and task systems with random limits, allowing tasks to have limits beyond the end of their periods [22]. In this paper, there are two main contribution folds such as: (1) The analysis precision has been significantly improved against previous work, and (2) To my knowledge, this is the first work to rectify the RTA problem of arbitrary-deadline systems on multiprocessors.

Table 1 Existing dynamic scheduling algorithm

Algorithms	Objectives	Merits	System types	Conclusion/ future enhancements
ETF	The initiate time reduced of a node at every step	Processor selection phase goes on side by side	HM	Heterogeneous environment
DLS	To select highest static stage and lesser time to schedule	Pair matching of nodes is performed, and highest priority node is found	HM	Heterogeneous environment
EDF	Job priority is inversely proportional to its deadline	No need to define priorities	HT	Future deadlines can be calculated
OSDTG	To minimize the makespan	IC of SN of channels	HT	Off-line
DLBTTP	Determine transferring tasks between each two nodes in the system	Well RT with a feasible amount of communication overheads	Distributed	IC and transfer delays
DTS	Lower time complexity	High speedup-dependent tasks	HM	Add fault-tolerant
DLSAGO	Overcome the spend much time and space for search	The most favored and the fittest are selected	HT	Homogeneous system
FTDTGS	To minimize overheads in the absence of faults with recovery costs	Handling recovery correctly and efficiently for dynamic task graph	HT	Recovery faults without global coordination
DTSLB	Schedule the tasks in a HTS, free and non-preemptive	Solving the DTS using PSO with fixed inertia	HT	Bounded and pre-emptive
DLBGA	Less run time, higher processor utilization, and good load balance	Threshold strategies, information switching criteria, and IC	HT	Large number of very effective tasks
PGAH	Reduce parallelization of the fitness evaluation	More scalable and extends its practicability	HT	Spends less time
GSMC	Timing controls to progress the schedulability of MCTS	Priority task policy and global schedulability tests	HM	Offers robust performance
RTAGFP	Improve the response time	Iterative analysis procedure and improve the efficiency	HM	Lead worst-case RT and good SOA

(continued)

Table 1 (continued)

Algorithms	Objectives	Merits	System types	Conclusion/ future enhancements
RTBFP	Improve the analysis precision response time	Improve the RTA problem	HT	Deal with platforms and task systems
LBSCMCS	Priority-based scheduling	Better quality to the space-time partitioning approach to scheduling	HM	Study of the schedulability properties of OCBP

3.15 Load-Based Schedulability Analysis of Certificate Mixed-Criticality System (LBSCMCS)

It is apriority-based algorithm for scheduling such mixed-criticality systems on preemptive uniprocessor platforms. This paper shows that this algorithm is strictly better to prior algorithms that have been used for scheduling mixed-criticality systems needing certification [23]. The conventional real-time scheduling theory does not address the certification consideration which give rise to fundamental new resource allocation and scheduling challenges (Table 1).

4 Comparison of Dynamic Scheduling Algorithms on HMS and HTS

A Comparison of aspects influencing the homogeneous and heterogeneous environment has been done and different algorithms are discussed. Out of which, ETF and DLS were applied on homogeneous setting while OSDTG and DLBGA were applied on heterogeneous setting, although their objective were same, i.e., to minimize the execution time and makespan. DTS algorithm may be adding fault-tolerance, whereas DLSGA can be homogeneous in future. FTDTGS is useful for recovery from faults without global coordination, while DTSLB is pre-emptive and dependent in nature, yet both use HTS. The goal of RTAGFP and RTBFP improves the response time but cannot be applied on same system. The objectives, merits, and future enhancement of the algorithm have already been discussed. This review is mainly focused on the parameters such as makespan, running time, resource utilization load balancing, speedup, efficiency, and high performance.

5 Conclusion and Future Work

The paper gives a short review of DSA for HMS and HTS. In this study, we found that dynamic scheduling algorithm is important to scheduling. It helps in answering questions such as: how to minimize the execution time and makespan, how to load balance on each processor, how to manage selection of task, and how to improve performance utilization and efficiency of system. It is also analyzed that load balancing issue in HMS was found easier compare to HTS in essence of load capacity. The appliance of the dynamic scheduling algorithm is a rapidly developing research area.

Hence, on the basis of this study in future will propose a novel dynamic scheduling algorithm for homogeneous as well as heterogeneous system to better performance.

References

1. Kwok, Y.K., Ahmad, I.: Static scheduling algorithms for allocating directed task graphs to multiprocessors. ACM Comput. Surv. **31**(4), 406–471 (1999)
2. Singh, K., Alam, M., Sharma, S.K.: A survey of static scheduling algorithm for distributed computing system. Int. J. Comput. Appl. **129**(2), 25–30 (2015)
3. Singh, M.K., Tiwari, R.: A survey on scheduling of parallel program in heterogeneous system. Int. J. Advanced Research in Computer Engineering & Technology. **1**(8), 357 (2012)
4. He, Y., Liu, J., Sun, H.: Scheduling functionally heterogeneous systems with utilization balancing. In: IEEE International Parallel and Distributed Processing Symposium, pp. 1187–1198 (2011)
5. Alam, M., Varshney, A.K.: A comparative study of interconnection network. Int. J. Comput. Appl. **127**(4), 37–43 (2015)
6. Choudhury, P.: Online scheduling of dynamic task graphs with communication and contention for multiprocessors. IEEE Trans. Parallel Distrib. Syst. **23**(1), 126–133 (2012)
7. Amalarethinam, D.I.G., Joyce Mary, G.J.: A new DAG based dynamic task scheduling algorithm (DYTAS) for multiprocessor systems. Int. J. Comput. Appl. **19**(8), 24–28 (2011)
8. Kaur, P., Kaur, A.: Implementation of Dynamic Level Scheduling Algorithm Using Genetic Operators. Int. J. of Appl. or Innovation in Eng. & Manag. **2**(7), 2319–4847 (2013)
9. Khan, Z.A., Siddiqui, J., Samad, A.: Linear crossed cube (LCQ): a new interconnection network topology for massively parallel system. Int. J. Comput. Netw. Inf. Secur. **7**(3), 18–25 (2015)
10. Kurt, M.C., Krishnamoorthy, S., Agrawal, K., Agrawal, G.: Fault-tolerant dynamic task graph scheduling. In: SC14: International Conference for High Performance Computing, Networking, Storage and Analysis (2014)
11. Visalakshi, P., Sivanandam, S.N.: Dynamic Task Scheduling with Load Balancing using Hybrid Practical Swarm Optimization. Int. J. Open Problems Compt. Math. **2**(3), 475–488 (2009)
12. Khan, Z.A., Siddiqui, J., Samad, A.: A novel multiprocessor architecture for massively parallel system. Int. Conf. Parallel Distrib. Grid Comput. 466–471 (2015)
13. Kwok, Y.K., Ahmad, I.: Static scheduling algorithms for allocating directed task graphs to multiprocessors. ACM Comput. Surv. (CSUR) **31**(4) (1999)

14. Evans, D.J., Butt, W.U.N.: Dynamic load balancing using task-transfer probabilities. Parallel Comput. Elsevier North-Holland **19**, 897–916 (1993)
15. Zomaya, A.Y., Hwei, Y.: Observations on using genetic algorithms for dynamic load-balancing. Parallel Distrib. Syst. IEEE **12**(9) (2001)
16. Munetomo, M., Takai, Y., Sato, Y.: A genetic approach to dynamic load-balancing in a distributed computing system. In: Proceedings of First International Conference on Evolutionary Computation, IEEE World Congress Computational Intelligence, vol. 1, pp. 418–421 (1994)
17. Pico, C.A.G., Wainwright, R.L.: Dynamic scheduling of computer tasks using genetic algorithms. In: Proceedings of First IEEE Conference Evolutionary Computation, IEEE World Congress Computational Intelligence, vol. 2, pp. 829–833 (1994)
18. Kelly, O.R., Aydin, H.: Fixed—priority global scheduling for mixed-critically real-time system. Int. J. Embedded Syst. **6**(2/3) (2014)
19. Sun, Y., Lipariy, G., Guanzx, N., Yix, W.: Improving the Response Time Analysis of Global Fixed-Priority Multiprocessor Scheduling. Embedded and Real-Time IEEE Xplore (2014)
20. Davis, R.I., Burns, A.: Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. Real-Time Syst. **47**(1), 1–40 (2011)
21. Guan, N., Stigge, M., Yi, W., Yu, G.: New response time bounds for fixed priority multiprocessor scheduling. In: 30th IEEE on Real-Time Systems Symposium, 2009, RTSS 2009, pp. 387–397. IEEE (2009)
22. Guan, N., Stigge, M., Yi, W., Yu, G.: New Response Time Bounds for Fixed Priority Multiprocessor Scheduling. In Real-Time Systems Symposium, RTSS 2009. 387–397 IEEE, (2009)
23. Li, H., Baruah, S.: Load-based schedulability analysis of certifiable mixed-criticality systems. In: Proceedings of the 10th ACM International Conference on Embedded Software (2010)