Learning of Scheduling Algorithm with Maximum Compatible Activity or Minimum Makespan

Vaishali K. Patel
PG Student
Department of Computer Engineering,
Alpha college of Engineering,
Khatraj, Gujarat, India.
hir.vish125@gmail.com

Mitula H. Pandya
Assistant Professor
Department of Computer Engineering,
Alpha college of Engineering,
Khatraj, Gujarat, India.
mitula.pandya.88@gmail.com

***Abstract*—For managing process of cpu there are many types of scheduling algorithm. Each scheduling algorithm has its own merits and demerits. A resource is allocated to client depends on mechanism of scheduling algorithms. Scheduling decisions try to minimize average response time,turn around time and waiting time for process. This paper proposes various scheduling algorithm.**

***Index Terms*— Activity selection problem, Job shop scheduling problem, Round robin scheduling, Compatible activity, Makespan**

I. INTRODUCTION

Scheduling is the process of assigning an order to the process according to which it will be executed on CPU. There are different types of scheduling algorithms. In this paper we will give consideration to activity selection problem, job shop scheduling problem and round robin scheduling.

Activity selection problem can be defined as: given set of activities. Each activity has its own start time and finish time denoted by s and f respectively. The main goal of activity selection problem is to select maximum number of compatible activities with each other[8]. Two activities are said to be compatible (non-conflicting) if starting time of second activity is greater than or equal to finish time of first activity.

Job shop scheduling problem can be described as following: Given n jobs and m machines[9]. Processing time of each job on each machine is also given. Each job will be processed on each machine. The main goal of job shop scheduling is to select the order of execution of process on each machine in such a way that makespan becomes minimum. Makespan is total time unit required to complete given sequence of process.

In round robin scheduling predefined time slices are assigned to each process in circular manner. Round robin scheduling handles all the activity without any priority.

The concept of selecting next process for execution is called scheduling decision. During each Scheduling decision context switch occurs. Context switch means one process currently running process stop its execution and put back to the end of ready queue and another process will be dispatched.

To choose which algorithm should use in particular situation we must consider the properties of different algorithm. Many criteria have been defined for comparing scheduling algorithms are as follows:

1. CPU utilization: keep the CPU busy for maximum time.
2. Throughput: Number of processes that are completed in particular time unit.
3. Turnaround Time: The time interval between submission of a process and competition of a process.
4. Waiting time: Total time spend waiting in the queue.
5. Response time: time interval between submission of a process and first response from the system.

II. BASIC CONCEPTS

A. *Activity Selection Problem*

Activity selection problem is a mathematical optimization problem. It is concern with selection of non-conflicting or compatible activities to perform in a predefined time period. There are different types of algorithm to solve optimization problem. Two methods of them are: greedy method and dynamic programming method. Here we will mainly discuss on different greedy strategies for optimization problem.

i	0	1	2	3	4	5	6	7	8	9	10
s_i	13	9	29	3	26	13	1	17	10	1	13
f_i	15	24	30	7	28	20	6	27	29	13	15

Fig.1. activities with start time and finish time[1]

Activities can be selected based on following criteria[1]:

1.Random
2. Index of activity in increasing order
3. Index of activity in decreasing order
4.Starting time of activity in increasing order
5.Starting time of activity in decreasing order
6.Finish time of activity in increasing order
7.Finish time of activity in decreasing order
8.Time period of activity in increasing order
9. Time period of activity in decreasing order

Strategy	Activities selected	# activities
Random	[9, 4, 5, 2]	4
Index, increasing	[0, 2, 3, 4]	4
Index, decreasing	[9, 7, 2, 0]	4
Start time, increasing	[6, 1, 4, 2]	4
Start time, decreasing	[2, 4, 5, 3]	4
Finish time, increasing	[6, 0, 7, 2]	4
Finish time, decrease.	[2, 8, 3]	3
Duration, increasing	[2, 0, 4, 3]	4
Duration, decreasing	[8, 6, 2]	3

Fig.2. results of applying greedy strategy[1]

Generally we use finish time of activity in increasing order as greedy strategy. Two activities 1 and 2 are said to compatible or non-conflicting if start time of 2 is greater than or equal to finish time of 1.

B. Job Shop scheduling

In job shop scheduling problem there are **n**(number of jobs) jobs to be processed on **m**(number of machine) machines with the aim of minimizing certain objective function.

1) *Job Shop Scheduling Using Genetic Algorithm*

In search process genetic algorithm will generate a new solution using genetic operators such as selection, crossover and mutation. In hill-climbing, search process will stop when next solution is not better than current solution. This criterion leads the solution to the problem of local optima. While, genetic algorithm starts its search space in population and maintain number of population in iteration[2].

In job shop scheduling problem critical path is longest path taken from the first operation processed until the last operation leaves the workplace. Critical operations are all the operations on critical path. The critical operations on same machine are called critical block(CB). The distance between two schedule is known as disjunctive graph(DG) distance. DG distance between two schedules s1 and s2 can be measured by calculating number of differences in the processing order of operation on each machine.

Schedule can be represented by set of permutations of jobs on each machine. This representation is called job sequence matrix. In this job sequence matrix rows will represent machine and columns will represent order of jobs on that machine. In job shop scheduling problem objective function which is to be minimized is makespan.

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 1 & 3 \end{bmatrix}$$

Fig.3 job sequence matrix[2]

In this algorithm DG distance between two randomly selected schedules is calculated. If this distance is less than some predefined value then mutation will be performed[2]. Otherwise crossover is performed on parents to generate ne child.

2) *job Shop Scheduling With Load Balancing*

Load balancing is used to distribute workload among multiple computing resources. There are many advantages of load balancing: resource use optimization, maximum throughput, minimum response time and avoid overload of any one of resource. The load balancing algorithm allocates job on identical machines with aim to reduce job total throughput time and to improve total resource utilization.

In load balancing load is transferred from heavily loaded machine to lightly loaded machine so that no machines are idle[3]. In this problem n number of jobs have to scheduled on m number of machines. Machines are grouped into working center. Working center consists of identical machines(which have same characteristics) and non-identical machines(don't have same characteristics).

There may be some situation when it is not possible to meet due date for all the jobs. Hence to deliver at least some part of jobs schedule can decide to split the job into lots. There are three phases in job shop scheduling problem:

Phase-1: lots determination
Phase-2: Machine allocation
Phase-3: Sequencing of each machine

Here focus is given only to machine allocation phase. Different steps of machine allocation phase are:

a). *Initialization***:** there are two cases
1)Number of lots of job is less than or equal to number of machines

2) Number of lots of job is greater than number of machines

*b). Chromosome Representation:*There are two layers in representation:

Layer-1: contains job identification number and the number of lots the job is split into.
Layer-2: consists of 2 parts
i)Identification of lot
ii) Machine the lot is allocated to

C. Round Robin Scheduling

All processes should be scheduled fairly. To provide fairness in schedule round robin algorithm employs time sharing. This algorithm gives each job a time slot. The job will be interrupted if it is not completed by its time slot. The job is resumed next time when a time slot is assigned to that process.

For example, if the time slot is 50 milliseconds, and *job1* wants a total time of 300ms to complete, the round-

robin scheduler will interrupt the job after 50 ms and give other jobs their time on the CPU. Once the other jobs have had their equal share (50 ms each), *job1* will get another allocation of CPU time and the cycle will repeat. This process is repeated until the job finishes and needs no more time on the CPU.

In this paper we will discuss about two different ideas about round robin scheduling algorithm: 1) An Improved Round Robin Scheduling Algorithm 2) Burst Round Robin as a Proportional-Share Scheduling Algorithm.

1). *An Improved Round Robin Scheduling Algorithm*

In round robin scheduling algorithm time is split into equal time slices.In contemporary round robin scheduling algorithm scheduler maintains a queue for processes which are ready to execute. Scheduler also have list of blocked processes and swapped out processes. At the end of ready queue the process code block of newly created process is added and process code block of terminating process is removed. The scheduler will select the process code block which is at the head in ready queue. When the running process finishes its time slice this process will be moved from end of ready queue.

The event handler performs actions as following:

- When process makes an input or output request process code block is removed from ready queue to blocked list.
- When process swapped out its process code block is removed from ready queue to swapped out list.
- When I/O operation awaited by a process finishes its execution its PCB is removed from blocked to the end of the ready queue.
- When process is swapped in its process code block is removed from list to the end of the ready queue.

In improved round robin algorithm first time all processes are allocated to CPU as like present round robin algorithm. After that scheduler will select the next process whose time requirement is shortest from waiting queue. This procedure is repeated until all processes complete their execution.

2). Burst Round Robin as a Proportional-Share Scheduling Algorithm

In this algorithm idea of low scheduling overhead of round robin algorithm and favor of shortest jobs is combined. This idea does not completely eliminate unfairness, but tries to reduce unfairness of process allocation. This idea presents weight adjustments for the processes that are blocked for input or output operations and lose some CPU time. This algorithm propose the idea that processes that are close to complete their completion will get more chances to complete their execution and remove from ready queue. This will reduce the number of processes in the ready queue by completing shorter jobs faster.

A modified version of round robin algorithm is weighted round robin algorithm. In weighted round robin each process P has allocated weight. This weight decides the share of CPU time of that process. If a time quantum is specified to be 20 time units and if we have three processes A1,A2,A3. The weights assign to each process are 14,8 and 18 respectively. The algorithm will assign 70% of time quantum to process A1, 40% of time quantum to process A2 and 90% of time quantum to process A3.

To achieve proportional fairness to all the process is the main goal of proportional share scheduling algorithm. Providing fairness is important in a dynamic environment in which processes get blocked from using CPU. The process that lost CPU time during it was blocked should get more shares of CPU time.

Higher the process weights, higher the time quanta. Shorter jobs will be given more time. Hence they will be removed from ready queue as soon as possible. In the proposed weight technique process weight is inversely proportional to its CPU burst time and can be given as follows:

$$Weight_i \propto \frac{1}{Burst_i}$$

When a process is blocked for input or output operations it will be moved to a waiting queue. After finishing its work the process will move back to ready queue. To achieve proportional fairness the weights of blocked process will be updated according to formula as given below:

$$New\_Weight_i = Old\_Weight_i \frac{process\_Waiting_i}{Avg\_processes\_Waitinig}$$

III. CONCLUSION

Here different scheduling algorithms are considered. Each algorithm has its own advantages and disadvantages. Based on our working situation we can use different scheduling algorithm.

REFERENCES

[1] *ITiCSE*,Active Learning of Greedy Algorithms by Means of Interactive Experimentation, J. Angel Velázquez-Iturbide, Antonio Perez-Carrasco

[2] 2006-IMT-GT conference on mathematics, science and application, A Job-Shop Scheduling Problem (JSSP) Using Genetic Algorithm (GA), Mahanim Omar, Adam Baharum, Yahya Abu Hasan

[3] A Genetic Algorithm for Job Shop Scheduling with Load Balancing, SanjaPetrovic and Carole Fayad

[4] International Journal on Computer Science and Issues,A Genetic Algorithm Approach for Solving Flexible Job Shop Scheduling Problem, Sayedmohammadreza Vaghefinezhadand Kuan Yew Wong

[5] IEEE,An Improved Genetic Algorithm for Solving Flexible Job shop Scheduling Problem, ZHOU Wei, BU Yan-ping, ZHOU Ye-qing
[6] International Journal on Computer Science and Engineering,An Improved Round Robin Scheduling Algorithm for CPU scheduling, Rakesh Kumar Yadav, Abhishek K Mishra, NavinPrakash and Himanshu Sharma
[7] International Journal on Computer Science and Engineering,Burst Round Robin as a Proportional-Share Scheduling Algorithm, TarekHelmy and AbdelkaderDekdouk
[8] http://en.wikipedia.org/wiki/Activity_selection_problem
[9] http://en.wikipedia.org/wiki/Load_balancing_%28computing%29
[10] Introduction to Algorithm, Thomas H. Coreman