TU Wien Informatics

TU Wien
Faculty of Informatics
Research Group for Parallel Computing

Parallel Algorithms
2022W
Assignment 2

January 11, 2023

2 Person Group 8
1: Ferdinand MOND, 11945397
2: David MÜLLER, 01620740

1 ParJudge Submission Information

ID	**#763**
Status	**Accepted**
Solved test cases	**10**
Score	**56.7**

2 Algorithm Description

2.1 Constructing a feasible schedule

Our algorithm is a First Come First Serve (FCFS) algorithm, which means it schedules the jobs in the order that they arrive. In our case, since we know that the jobs in the instance are in real time order, it is sufficient to loop through the total number of jobs and handle each in sequence. When a new iteration of the loop is started, time is moved forward to the release time of the current job. The list of running jobs is checked against the current time, and if any are known to have finished before the current time those are removed and the machines freed. Clearing this running list is the only time when the information about actual runtime that the scheduler is not supposed to know is used. When a new job is slated to be scheduled, the currently free amount of machines are compared to the required number for this job, and if there are at least as many available as required the job is scheduled at the current time. If there are not enough machines available to schedule the current job, the running list is emptied in the order of when the jobs actually finish until there are enough free machines available. Then the current job is scheduled at the current time, which is the time when enough machines became available.

2.2 Finding the best schedule

We tried to find a better method but were not able to implement one that full filed a valid schedule. We tried to use the EASY algorithm from the lecture slides but we failed to implement a vaild solution. The file is attached as well to the submission. The error was in

creating a situation where too many jobs were scheduled at one time, but even using debug tools and going through the scheduling manually we were unable to find the issue. This algorithm works by basically going FCFS until a situation arises where the recently arrived job cannot be scheduled due to a lack of available machines. If this happens, the algorithm finds a "shadow time" which is the earliest possible time that job could be scheduled with the current known schedule, which only takes into account the requested times so is usually quite far into the future. Additionally, a number of extra nodes is calculated, which is the number of nodes that are free at the shadow time when the job would be scheduled, so can be used freely to fill up space earlier. Now, if another job arrives before the original stuck job can be scheduled, it is determined if this new job can be currently scheduled with the available number of nodes, and if so, it is checked if this job would either finish in the requested time before the shadow time or fit into the extra nodes that are available anyways, and if so, it is scheduled. It should be noted here that in our current state the algorithm is only capable of backfilling one job if that is appropriate, but this could be changed if the actual core was producing a valid result. We will hand in the valid schedule algorithm as well as our attempt to get an improved schedule. Submission #800 was the last test we performed with our "improved" algorithm. The first schedule seems to be valid but the second one is not. We also did not have the time and energy to write our own machine counting script that would be able to tell us where the issue was, similar to what was provided in exercise 1.