“Best fit plus” Job scheduler Cost-efficient

Aydin Sumer (45437009)

Introduction:

The goal of this project is to design and implement a new scheduling algorithm that minimises the total server rental cost.

It should also have a better average in all three-performance metrics (avg turnaround time, avg resource utilisation and total rental cost) than one or more of three baseline algorithms, FF, BF and WF.

Problem definition:

When trying to optimize the cost performance metric another one will become worse off. But since we want to keep all three-performance metrics in the range of the three base line algorithms we can only make slight improvements.

To reduce the cost, high-capacity servers that would have wasted cores will not be used and queuing of jobs to the minimum required server can elevate costs while not majorly affecting turnaround time or resource utilisation.

Algorithm description:

This algorithm works in a priority order.

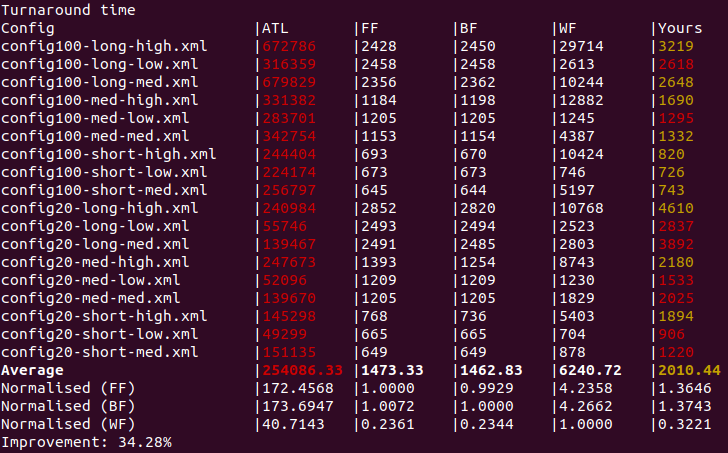
1. An available server perfectly fits the requirements to do the job.
2. An available server has the lowest fit factor (left over cores) of less than half the core required to do the work.
3. A capable server perfectly fits the requirements to do the job.
4. The first capable server regardless of status that does not have 2 or more waiting jobs.
5. The first capable server regardless of status and waiting jobs.

The goal of this algorithm is to only use large servers if they will be able to do another job i.e., parallel processing. Otherwise send the jobs to be queued to a capable one while load balancing them among the capable servers. This will allow it to queue up jobs to save cost while keeping relatively the same performance as best fit in most case scenarios.

Implementation:

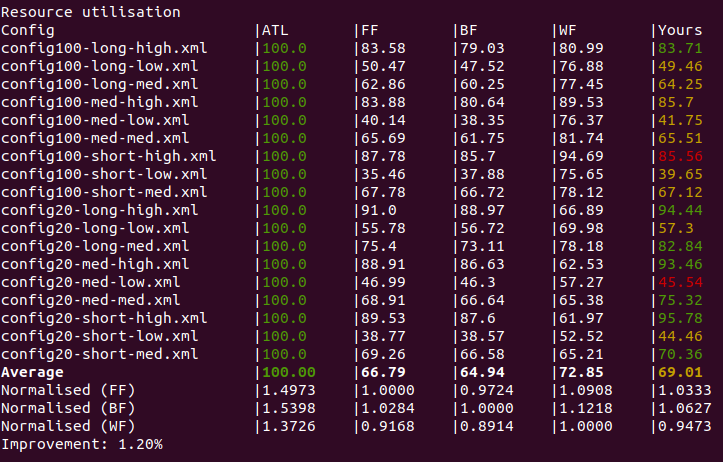
This algorithm uses data structures to hold each server’s data in a string array form if it fits the criteria to be selected. This way no extra servers are hold onto. Each job is also stored in as string array. Fit factor is an integer calculated from the server’s available core count – the necessary cores.

Evaluation:

Turnaround time:

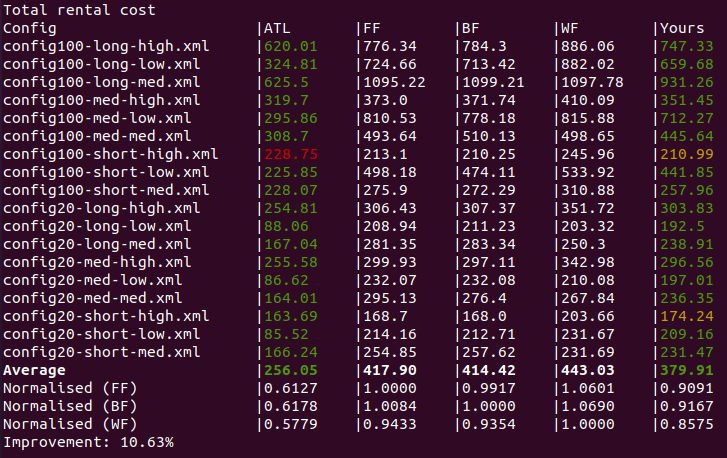
With the goal to be cost effective, turnaround time was expected to be worse than FF or BF but less than worst fit in cases where it could find lots of available slots to fit in parrel processing.

As seen in the turnaround comparison table to the right.

Resource utilization:

On average this algorithm has similar or better utilization to FF and BF. This is due to using servers that can accommodate parallel jobs or queuing jobs to best fit servers.

As seen in the resource utilization comparison table to the right.

Total rental cost:

With this algorithm using the same servers more often or trying to parallel processing jobs, less expensive servers are also used to do the work. As seen in the chart to the right, it is about 10% more cost efficient than the baseline algorithms.

Conclusion:

In conclusion this new algorithm on average is better than all 3 base algorithms when it comes to cost at the expense of turnaround time. If scheduled jobs have varied core count requirements, there are jobs that can be parallel processed and will be taking the turnaround time hit. This is much more preferred as it will be lower cost and sometimes also higher utilization than the other base line algorithms. This algorithm can be adjusted when jobs and server sizes are known, to maximise its effectiveness.

References:

https://github.com/a758/BFA