COMP3100 PROJECT STAGE 2 REPORT – Performant resource allocating algorithm for distributed systems

BREYDEN LANDBERG **45357528**

Introduction

Stage 2 of the project seeks to improve upon the efficient cloud resource allocator pioneered in Stage 1, improving it significantly in at least one provided metric, potentially at the expense of others. The goal of Stage 2 is to design and implement a new scheduling algorithm that seeks to either minimise costs/turnaround time, or maximise resource utilisation. My algorithm aims to minimise turnaround time; I chose this because I am interested in high performance distributed systems, and sought to modify the Best-Fit algorithm to provide the best performance instead of reducing costs or maximising utilisation.

Problem definition

The three provided ds-client algorithms (First-Fit [FF], Best-Fit [BF], Worst-Fit [WF]) are all acceptable algorithms, but are limited in efficacy for distributed systems that contain more information from which scheduling decisions can be made.

My performance objective when developing my algorithm was to modify an existing baseline algorithm to significantly improve turn-around time with as little detriment to the other metrics as possible. I sought to modify and build upon the BF algorithm to achieve this goal.

The overarching idea of the modification was to handle waiting jobs differently from the BF algorithm. The BF algorithm searches for a server with no waiting jobs and defaults to the best fitting Active/Booting server upon not being able to find one. In my custom algorithm's implementation, I tried to overcome this inefficiency. I chose to overcome this inefficiency because I realised that there are more effective ways to schedule jobs based upon fitness and wait times than the baseline BF algorithm allowed for. More concrete details are in the algorithm section below.

I believe this was a good metric to focus on improving, as performant distributed systems will become more important as more and more data continues to be passed around, and wealthy companies invested in these sorts of technologies start allotting larger budgets to performance.

NOTE:

- Fitness value is the server's core count minus the job's core requirements. It is the integer variable fitnessValue in Client.java.
- Turn-around time is the time between a job's submission and completion.

The algorithm

The algorithm is simply a modified BF algorithm as follows:

Input: Job j, Servers s

Output: Server s* upon which Job j will be scheduled

Process: Search for Server s* in Servers s from index 0 to index n – 1 that satisfies the following conditions:

1. Has sufficient resources readily available for Job j.
   Every Server s in Servers s satisfies this condition, by nature of GETS Capable returning all capable servers.*
2. Has the smallest fitness value.
3. Has the fewest waiting jobs.

If no s* is found, set s* to the first server in Servers s and return it.

Where the algorithm diverges from the BF algorithm is how it accounts for a server's waiting jobs. The BF algorithm simply searches for a server that has **no** waiting jobs and the lowest fitness value. If none are found, simply the first Active/Booting server with the lowest fitness value is selected.

During the modified BF algorithm search however, s* is only set when either the current lowestFitnessValue is less than 0 or *both* the server's fitness value and waiting jobs are fewer than the current s* (initially set to servers.get(0)). From this search emerges the Server s* which has the lowest fitness value *and* fewest waiting jobs out of all the servers in Servers s.

Some performance is gained at the expense of other metrics, but overall, the algorithm definitively beats WF and ATL in all metrics and hovers around the same level as FF and BF in all metrics. Evidence for this is provided in the evaluation section below.

Example

Servers servers

Job j

set s* to first element of servers

set lowestFitnessValue to fitnessValue of s*

for each Server s in Servers servers

if lowestFitnessValue is less than 0 OR

if s.fitnessValue is less than lowestFitnessValue AND s has fewer waiting jobs than s*

s* = s

end conditional

end loop

return s*

Implementation details

The algorithm is implemented as method of Client.java, taking an ArrayList data structure of type Server. This ArrayList of servers is the collection of capable servers returned by the server after sending a GETS Capable command.

The algorithm searches by comparing fitness value integers and waiting jobs integers, returning a Server at the end.

All Client methods and fields are essentially taken from Stage 1 of this project.

Job and Server are both their own classes, with relevant fields and methods that correspond to the results of the commands received from the server; this code is the same as my implementation of Stage 1 of this project.

Evaluation

The modified BF algorithm performs well in turnaround time, with very slightly worse resource utilisation and very slightly improved total rental costs.

Using the second-to-final test script (the version before the final update on 29/05/2021) as follows:

```
breyden@breyden-VirtualBox:~/Documents/ds-sim/ds-sim/src/pre-compiled$ ./test_results "java Client" -o tt -n -c other/
```

shows the following results:

```
Turnaround time
Config                      |ATL        |FF         |BF         |WF         |Yours
config100-long-high.xml     |672786     |2428       |2450       |29714      |2431
config100-long-low.xml      |316359     |2458       |2458       |2613       |2458
config100-long-med.xml      |679829     |2356       |2362       |10244      |2357
config100-med-high.xml      |331382     |1184       |1198       |12882      |1182
config100-med-low.xml       |283701     |1205       |1205       |1245       |1205
config100-med-med.xml       |342754     |1153       |1154       |4387       |1154
config100-short-high.xml    |244404     |693        |670        |10424      |691
config100-short-low.xml     |224174     |673        |673        |746        |673
config100-short-med.xml     |256797     |645        |644        |5197       |645
config20-long-high.xml      |240984     |2852       |2820       |10768      |2787
config20-long-low.xml       |55746      |2493       |2494       |2523       |2494
config20-long-med.xml       |139467     |2491       |2485       |2803       |2412
config20-med-high.xml       |247673     |1393       |1254       |8743       |1441
config20-med-low.xml        |52096      |1209       |1209       |1230       |1210
config20-med-med.xml        |139670     |1205       |1205       |1829       |1176
config20-short-high.xml     |145298     |768        |736        |5403       |753
config20-short-low.xml      |49299      |665        |665        |704        |665
config20-short-med.xml      |151135     |649        |649        |878        |641
Average                     |254086.33  |1473.33    |1462.83    |6240.72    |1465.28
Normalised (FF)             |172.4568   |1.0000     |0.9929     |4.2358     |0.9945
Normalised (BF)             |173.6947   |1.0072     |1.0000     |4.2662     |1.0017
Normalised (WF)             |40.7143    |0.2361     |0.2344     |1.0000     |0.2348
Improvement: 52.10%
```

52.10% improvement in turn-around time, -2.25% improvement in resource utilisation, 2.92% improvement in total rental cost.

Using the most recently updated test_results script (30/05/2021) as follows:

```
breyden@breyden-VirtualBox:~/Documents/ds-sim/ds-sim/src/pre-compiled$ ./test_results_final "java Client" -o tt -n -c other/
```

Provides the following results (all averages):

	ATL	First-Fit	Best-Fit	Worst-Fit	Custom Algorithm
Turn-around time	254086.33	1473.33	1462.83	6240.72	1465.28
Resource utilisation	100.0	66.79	64.94	72.85	66.66
Total rental cost	256.05	417.90	414.42	443.03	412.69

Pros

Significantly improved turn-around time

Slightly improved rental costs

Improvement in distribution of job scheduling based on waiting jobs and fitness values

Cons

Slightly worsened resource utilisation

Conclusion

In conclusion, this custom scheduling algorithm based on the BF baseline algorithm demonstrates a significant increase in average turn-around time. I have found that the BF algorithm is a good foundation for designing algorithms optimised for turn-around time. In future, I would like to further optimise turn-around time with possibly even better resource utilisation results.

References

https://github.com/breydenlandbergstudent/comp-3100-project-stage2

https://github.com/distsys-MQ/ds-sim