ICS143B Project2 Main Memory Manager - Project Report

Anbang Xu(35086995)

1. Introduction

Design/implement a main memory manager for variable size partition, including allocating memory and deallocating memory. And using simulation to compare different allocation strategies. Here I use First-Fit and Best-Fit strategies.

2. High Level overview

(1). The user invokes driver, a. **generates streams of requests and releases** using parameters, b. repeatedly invokes request/release function, c. **gather statistics** in files for each request

(2). The driver repeats for **different parameters** and **different allocation strategies**

(3). Analyze, plot and describe results

What to vary?

- the total initial size
- distribution of request size
- steps of simulation
- allocation strategy selection

What to measure?

- average memory utilization - Occupied Block Size / Total Memory Block Size
- average search time - # holes examined / total # holes

(4). Algorithm for different allocation methods

a. first-fit:

```
public int firstFit(int requestSize) {
        numHoleExamined = 0;
        int curHole = firstHole;
        while (curHole >= 0) {
                numHoleExamined++;
                if (requestSize < getBlockSize(curHole))
                        return curHole;
                curHole = getNextHole(curHole);
        }
        return -1;
}
```

b. best-fit:

```
public int bestFit(int requestSize) {
        numHoleExamined = 0;
        int curHole = firstHole;
        int minDiff = Integer.MAX_VALUE;
        int returnHole = -1;

        while (curHole >= 0) {
                int blockSize = getBlockSize(curHole);
                numHoleExamined++;
                if (blockSize >= requestSize && (blockSize - requestSize) < minDiff) {
                        minDiff = blockSize - requestSize;
                        returnHole = curHole;
```

```
                }
                curHole = getNextHole(curHole);
            }
            return returnHole >= 0 ? returnHole : -1;
    }
```

3. Hypothesis

(1). Expected results before experiments. In this project, I use two strategies - First-fit and Best-fit.

a. **First-fit memory allocation:** first partition fitting the requirements
- Expect: Leads to fast allocation of memory space

b. **Best-fit memory allocation:** smallest partition fitting the requirements
- Expect: Results in least wasted space

(2). How to select a and d. a is the mean of request size and b is the deviation of request size. Assume the initial maximum memory size is MM(word-addressable). And I use MM = 2000 in the experiment.

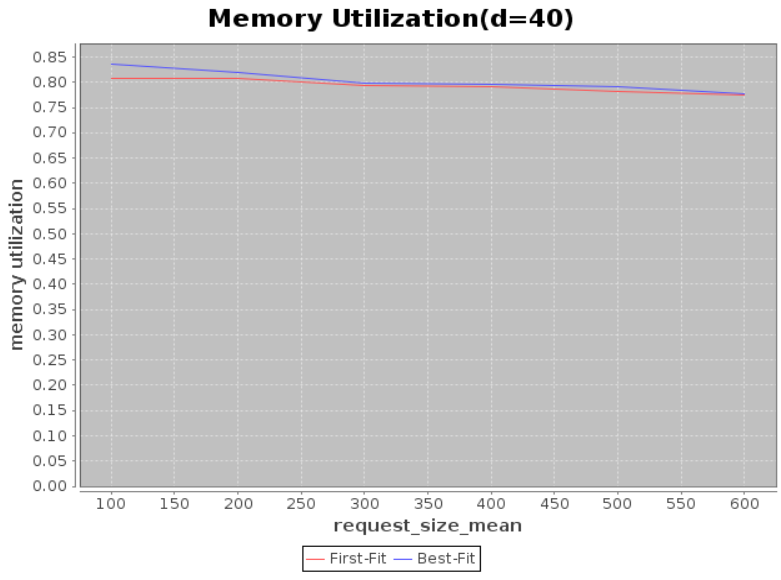
a. choose a:
- Analysis: in one hand, request “too big - more than half of total memory size” block leads low utilization due to large holes. On the other hand, request “too small” block also leads low utilizations due to tags
- Thus, I pick [100, 0.3 * MM] as the range of the mean of request size and split them into 6 slots in the experiment - [100, 200, 300, 400, 500, 600]

b. choose b :
- Analysis: 1. large d makes curve too flat - all request sizes are almost equally likely; 2. if d is bigger than 0.2 * MM, it’s still highly variable; 3. it’s better to pick up the d smaller than 0.1 * MM and it could help that most requests are close to a
- Thus, I pick [0.02, 1] as the range of the percentage of d in term of the total memory size. I split them into 5 slots in the experiment - [0.02, 0.04, 0.06, 0.08, 0.1] * MM

4. Results

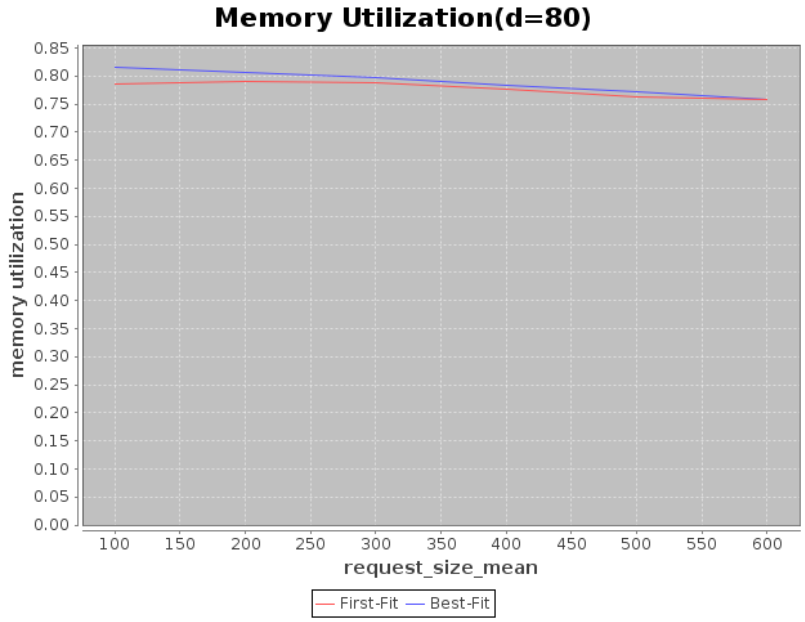
a. Memory Utilization(# used blocked size / total memory size)

1). d = 40:



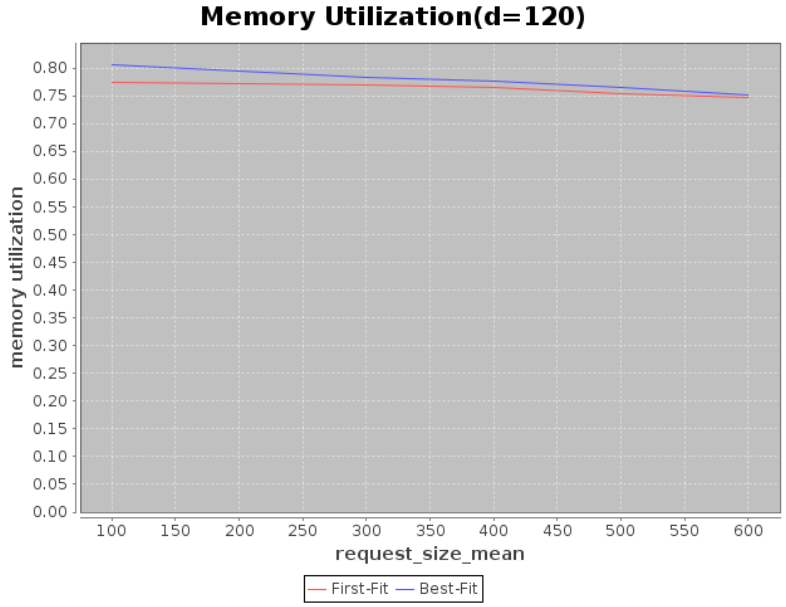
Analysis: When d = 40, the memory utilization of first-fit strategy and best-fit strategy both decrease as “a” increase. For the best-fit strategy, the memory utilization decreases faster when “a” varies from 100 to 300 than when “a” varies from 300 to 600. And in terms of the memory utilization, the best-fit strategy is better than the first-fit strategy in any “a”.

2). d = 80:



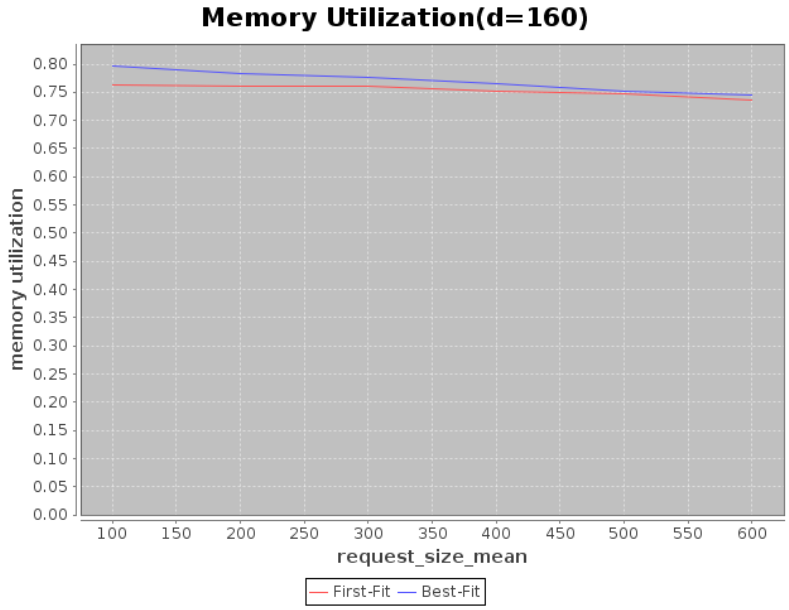
Analysis: When d = 80, the memory utilization of the first-fit strategy keeps almost the same(maybe a little bit increase) when “a” varies from 100 to 300 but decreases when “a” varies from 300 to 600. The memory utilization of the best-fit strategy decreases uniformly as “a” increases. And in terms of the memory utilization, the best-fit strategy is better than the first-fit strategy in any “a”.

3). d = 120:



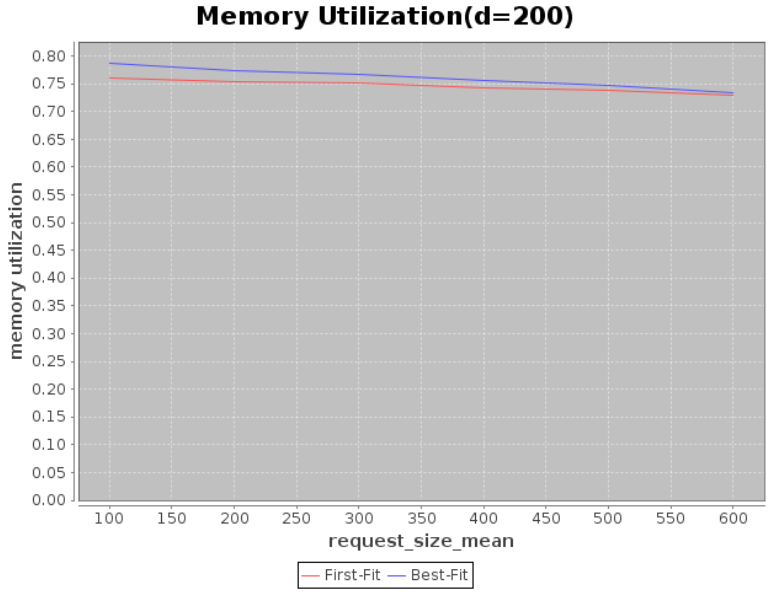
Analysis: When d = 120, the memory utilization of the first-fit strategy keeps almost the same when “a” varies from 100 to 400 but decreases when “a” varies from 400 to 600. The memory utilization of the best-fit strategy decreases uniformly as “a” increases. And in terms of the memory utilization, the best-fit strategy is better than the first-fit strategy in any “a”.

4). d = 160:



Analysis: When d = 160, the memory utilization of the first-fit strategy keeps almost the same when “a” varies from 100 to 300 but decreases when “a” varies from 300 to 600. The memory utilization of the best-fit strategy decreases uniformly as “a” increases. And in terms of the memory utilization, the best-fit strategy is better than the first-fit strategy in any “a”.

5). d = 200:



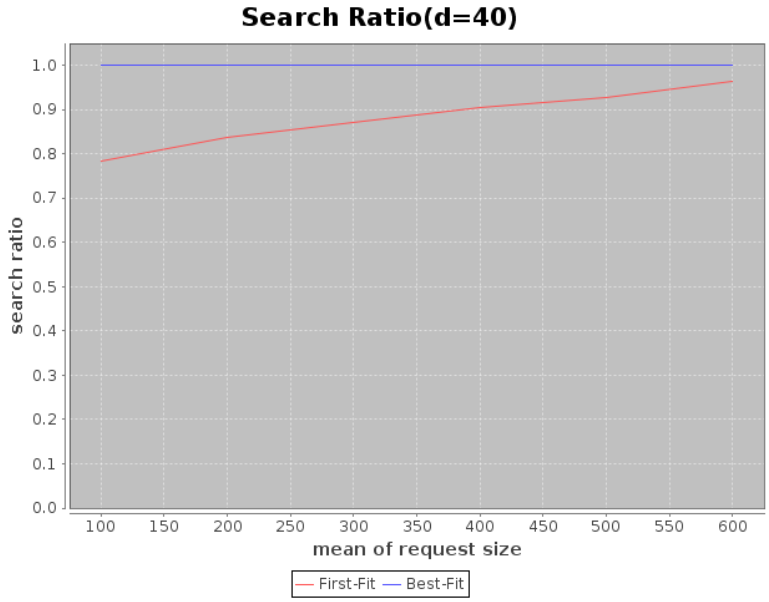
Analysis: When d = 200, the memory utilization of first-fit strategy and best-fit strategy both decrease as “a” increase. And in terms of the memory utilization, the best-fit strategy is better than the first-fit strategy in any “a”.

b. Search Ratio(# holes examined / # total holes)

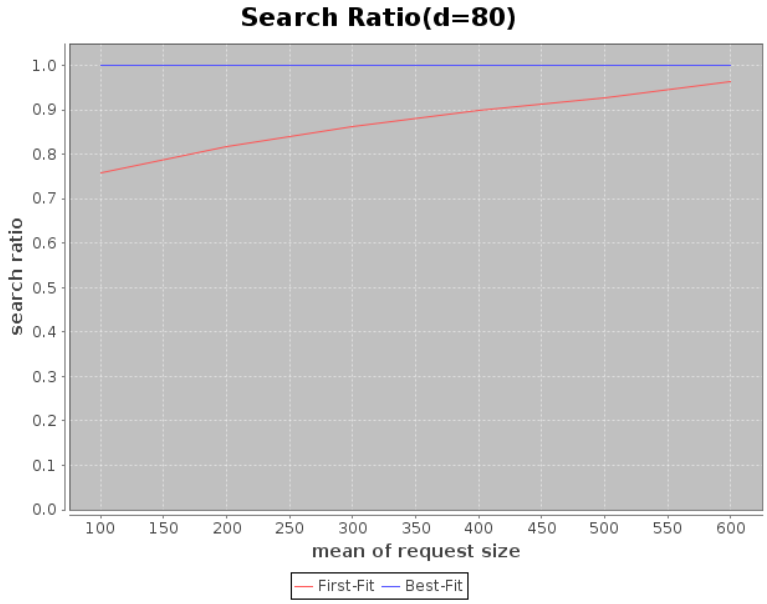
The analyses of all the graphs are the same, so I put this analysis on the top.

Analysis: no matter how “d” changes, the search ratio of the best-fit strategy is 1.0 in any “a”. This is due to my implementation. While the search ratio of the first-fit strategy increases as “a” increases.

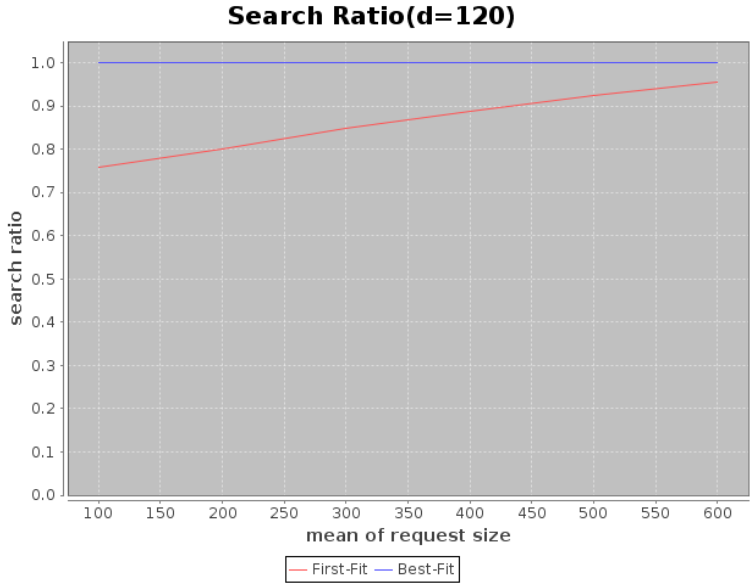
1). d = 40:



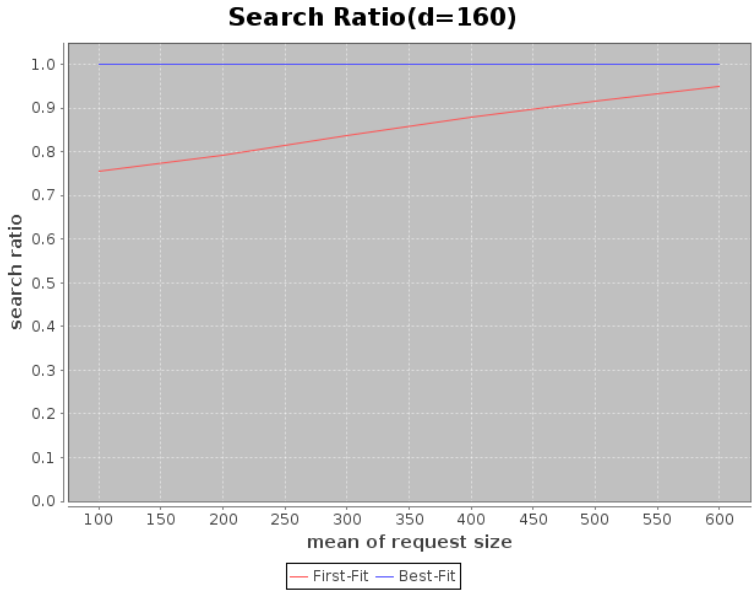
2). d = 80:



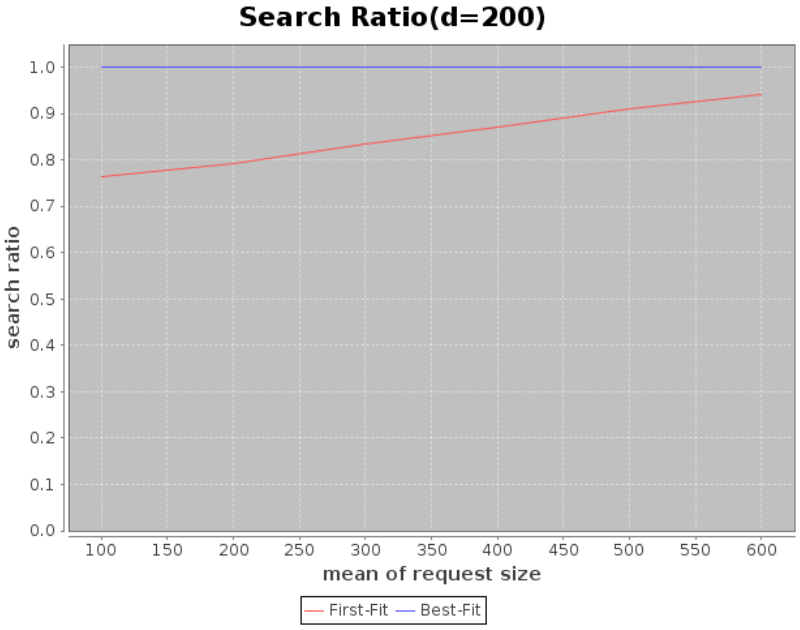
3). d = 120:


Search Ratio(d=120)
search ratio
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
100
150
200
250
300
350
400
450
500
550
600
mean of request size
First-Fit
Best-Fit

4). d = 160:


Search Ratio(d=160)
search ratio
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
100
150
200
250
300
350
400
450
500
550
600
mean of request size
First-Fit
Best-Fit

5). d= 200:



5. Conclusion

a. Were the results what you expected?

The results are within my expectation.

b. Which "a" value is best? Is it best for memory usage or search time or both? Is it best for both allocation methods?

From the graph above, we can get these conclusions.

1). For the first-fit strategy, the memory usage will decrease as "a" increases. In the mean time, the search ratio will increase as "a" increases, which means that the search time will decrease as "a" increases. Thus, in terms of both memory usage and search time, the best "a" value for first-fit strategy is the smallest request size - 100.

2). For the best-fit strategy, the memory usage almost keeps the same when "a" varies from 100 to 300, but it will decrease when "a" keeps increasing. While the search time always keeps the same, because my implementation of this strategy is to search through all the holes and figure out the best-fit available memory block. Thus, the best "a" value for best-fit strategy can be a range, which is [100, 300].

3). Roughly speaking, a = 100 is best for both of allocation methods.

b. Which "d" value is best? Is it best for memory usage or search time or both? Is it best for both allocation methods?

From the graph above, we can get these conclusions.

1). For the first-fit strategy, the memory usage will decrease as "d" increases. The search ratio is highest when "d" is the smallest one - 40, but it decreases down to 0.75 and keeps the same as "d" varies from 80 to 200. Thus, the best "d" value for both memory usage and search time is 40.

2). For the best-fit strategy, the memory usage will decrease as "d" increases. But compared with first-fit strategy, it decreases more slowly. Regarding the search time, it will keep the same due to my

implementation. Thus, the best “d” value for the best-fit strategy is 40 regarding the memory utilization and every “d” is the best regarding the search time.

3). d = 40 is best for both of allocation methods.

c. Which allocation method is best? Is it best for memory usage or search time or both?
As I expected, the first-fit strategy is better in “search time”, while the best-fit strategy is “memory usage”.

6. Code

See my code attachment in EEE.