Module 4: Critical Thinking

Brady Chin

Colorado State University Global

CSC507-2: Foundations of Operating Systems

Dr. Joseph Issa

February 9th, 2025

Critical Thinking

In operating systems, the characteristic that is more important is memory allocation, where every process is granted its resources to execute. Two of the many classic algorithms in memory allocation are First-Fit and Best-Fit. Memory's allocation on processes is the example of their similarity in view of a common aim, albeit different tools with their respective advantages and disadvantages in order to attain it. The first fit just assigns the first available block that is large enough to hold a process, considering speed over efficiency. On the contrary, First-Best tries not to waste space. Placement of a process is deposited in the smallest block that could be used for fitting it and hence reduced external fragmentation is expected to be brought about. The comparison will look into studies as concerned with workings in both algorithms with regard to benefits and appropriate circumstances for use.

How the Algorithm Works

The First-Fit algorithm works by making a memory allocation in which it scans the blocks of memories from the beginning for the first block that is large enough to accommodate this process (geeksforgeeks.org, 2023, May 31). Once a suitable block is found, the process is allocated, and the size of the block is updated. That is a fast approach because the algorithm stops the search once it finds a block that fits, hence being efficient in terms of time complexity. The disadvantage is that over time, this leads to external fragmentation-where memory blocks are left with small unused spaces, too small to fit other processes yet too large to be discarded. This is a problem of First-Fit because it searches for each process not the best block but the first available one.

First-Fit vs Best-Fit

The Best-Fit algorithm tries to minimize wasted space by placing a process in the smallest block that is able to accommodate it (PrepInsta Staff, 2025). This would prevent large chunks of memory from being left over unused since, with such an approach, the process gets

fitted into a block that would be closest to its size. However, Best-Fit does have to scan all the available blocks to find its best fit; hence, it is slower compared to First-Fit. Also, while it minimizes large gaps, it can create many small, unusable gaps within blocks, leading to internal fragmentation (University of Illinois Staff, 2020). This can be a problem if over a period of time many small processes are allocated because these small gaps cannot be utilized effectively.

When Best-Fit is Preferable

Best-Fit is suitable when memory is critical, in particular, in processes of roughly equal sizes and in cases when the objective is to minimize the number of unused large chunks. It is also good in small-memory systems, where every single bit may be of importance. However, systems that require a fast process or those that vary greatly in their process size can be better fitted with First-Fit because its process of allocation is faster. First-Fit does lead to some fragmentation, but it provides a good trade-off between the speed of allocation and simplicity in an environment where processes arrive and leave regularly.

Conclusion

It would, therefore, follow that both First-Fit and Best-Fit find their applications in memory management, each applicable under different system requirements. First-Fit works best in environments where speed is of the essence with the sizes of processes being all over the place since it minimizes allocation time at the possible expense of fragmentation. On the other hand, Best-Fit is a good fit for systems that want memory efficiency-especially when working with processes that are roughly comparable in size, and/or there should not be large holes. However, its execution is generally slower, it may result in internal fragmentations. The understanding of the trade-offs of each will help system designers make informed choices among various memory allocation strategies depending on the particular demands of the environment.

Code

Figure 1 shows the successful execution of the code demonstrating the First-Fit algorithm.

Figure 1: First-Fit algorithm example

first-fit.py

```python
memory_blocks = [500, 200, 400, 300, 700]
process_sizes = [213, 341, 900, 586, 152]

for i, process in enumerate(process_sizes):
    assigned = False
    for j, block in enumerate(memory_blocks):
        if process <= block:
            memory_blocks[j] -= process
            print(f'Process {process} assigned to block {j+1}. New size of block {j+1} is {memory_blocks[j]}')
            assigned = True
            break
    if not assigned:
        print(f'Process {process} can not be assigned. ')

print(f'\nFinal block sizes: {memory_blocks}')
```

Run first-fit

```
/usr/local/bin/python3.12 /Users/bradychin/Library/Mobile Documents/com~apple~CloudDocs/University/CSU Global/csu-gl
Process 213 assigned to block 1. New size of block 1 is 287
Process 341 assigned to block 3. New size of block 3 is 59
Process 900 can not be assigned.
Process 586 assigned to block 5. New size of block 5 is 114
Process 152 assigned to block 1. New size of block 1 is 135

Final block sizes: [135, 200, 59, 300, 114]
```

References

geeksforgeeks.org (2023, May 31) *First-Fit Allocation in Operating Systems.* GeeksForGeeks. https://www.geeksforgeeks.org/first-fit-allocation-in-operating-systems/

PrepInsta Staff (2025), *Best-Fit Algorithm in Operating System.* PrepInsta. https://prepinsta.com/operating-systems/page-replacement-algorithms/best-fit/

University of Illinois Staff (2020), *Memory Allocation.* Grainger College of Engineering, University of Illinois. https://courses.grainger.illinois.edu/cs240/fa2020/notes/heapMemoryAllocation.html