CIS 452 - Operating Systems Concepts

Nathan Bowman

Images taken from Silberschatz book

Memory Fragmentation

Contiguous memory allocation -- each process resides in its own single section of memory, and successive sections are contiguous

In a **variable-partition** scheme, the size of each partition can be different from the sizes of the others

OS keeps table tracking which memory is available

Region of available memory is called a **hole**

Allocating holes to processes is an example of **dynamic storage-allocation problem**

Solutions include **first fit**, **best fit**, and **worst fit**

First fit: allocate to first available hole of sufficient size

Best fit: allocate to smallest hole that fits process. Guaranteed to produce smallest leftover hole

Worst fit: allocate to largest hole. Guaranteed to produce largest leftover hole

Worst fit may seem like an odd idea, but the benefit is that it leaves a larger hole for the next process to fit

Best fit requires search through list of holes

First fit is very simple, but makes no guarantees about the leftover hole

Best way to compare is through simulation

Best fit and first fit are better than worst fit at using available memory

First fit is generally faster than best fit

Effective memory utilization must consider **fragmentation**

Fragmentation comes in two varieties: external and interal

External fragmentation exists when there are a large number of small holes

Even if there is enough unused memory for a process, the memory may be too split up (fragmented) to be useful

Fragmentation will be a problem regardless of which storage-allocation strategy is used

For first fit, if `N` blocks are allocated, `0.5 N` blocks will be lost to fragmentation on average

That is 1/3 of available memory!

This is known as the **50-percent rule**

One partial solution to external fragmentation is **compaction**

OS shuffles memory to move all free holes together into one large hole

Simplest way is to move all processes toward one end of memory

Compaction is not possible if relocation (adjusting memory addresses) is static

If relocation is dynamic, compaction requires simply moving the process and updating the base register

Even when possible, compaction can be expensive

We will see other solutions to external fragmentation later -- for now, we move on to internal fragmentation

To understand internal fragmentation, we need to change our scheme somewhat

If a process requires 1022 bytes and the available hole is 1024 bytes, it is not going to be worth it to create a new leftover hole of 2 bytes

To avoid this, physical memory is broken into fixed-size blocks and allocated in multiples of the block size

Downside of blocking this way is that process may not use entire block

Difference between actual storage of process and assigned size is **internal fragmentation**

Single process can have internal fragmentation of at most `block_size - 1`

Total internal fragmentation with `N` processes is at most `N*(block_size - 1)`