CIS 452 - Operating Systems Concepts

Nathan Bowman

Images taken from Silberschatz book

Thrashing

We briefly discussed how frames can be allocated to processes

- equal vs proportional allocation
- local vs global replacement

What happens if a poor allocation is provided?

Process without enough memory will page fault frequently

Any time new page must be brought in, it will kick out old page that is going to be used soon

When kicked-out page is requested, page fault will occur and another useful page will be paged out

Creates situation where process spends more time page faulting than performing useful work

When process spends excessive amount of time page faulting, we call this **thrashing**

With global replacement, situation can cascade and become much worse than one slow process

Recall: OS tries to keep degree of multiprogramming high

When OS notices CPU utilization is low, it increases degree of multiprogramming

As new process is brought in, process requires frames of memory to run

Thrashing

Consider the following scenario:

Current system memory usage is highly effective -- each process has enough frames to avoid thrashing, but few excess frames (because of high degree of multiprogramming)

One process moves to new, memory-intensive phase of its program, requiring sharp increase in number of frames

With global replacement, frames will be allocated to process and taken from other processes

Thrashing

Because system was near capacity, taking frames from other processes leads them to page fault

These page faults lead to frames being taken from more processes, which then page fault

Processes end up queuing and waiting for paging hardware, not making effective use of CPU

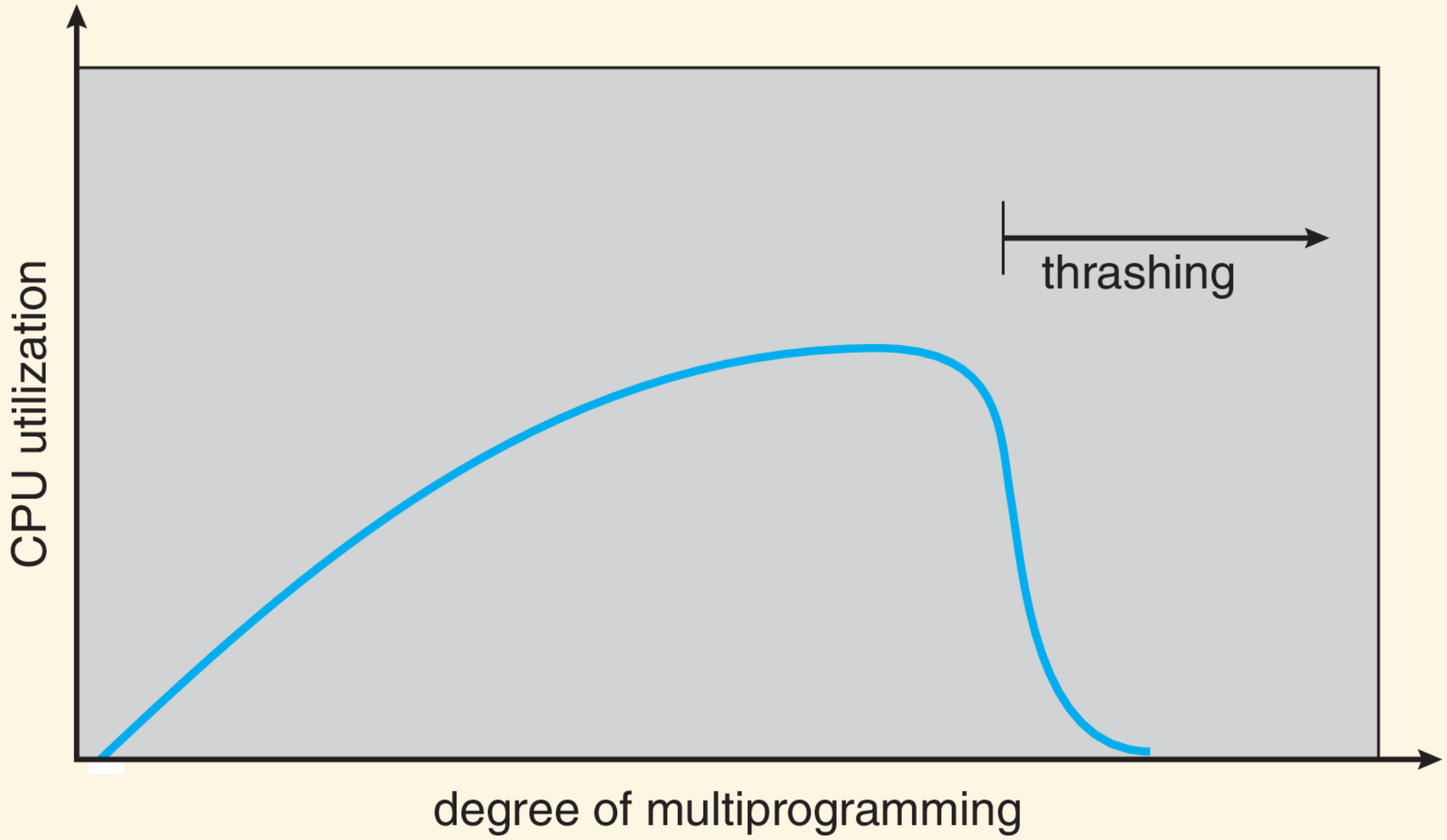
OS notices low utilization and *increases* degree of multiprogramming

Thrashing

Bad situation is now worse -- new processes require memory, more page faults will occur

CPU utilization decreases further, leading OS to increase degree of multiprogramming yet again

Behavior continues in cycle of increasing page faults that sharply decreases amount of useful work performed by system


CPU utilization
thrashing
degree of multiprogramming

Correct response to thrashing is to decrease degree of multiprogramming

When process is swapped out, corresponding frames can be reallocated among other processes, eventually decreasing number of page faults

How to prevent thrashing from cascading in this way?

Simple way is to use local replacement instead of global

If process does not take frames from another process, one process thrashing will not cause other processes to do so

Downside: process that is thrashing will make heavy use of paging hardware

This slows down access to paging hardware for other processes, resulting in increased page fault times

Even though only one process is thrashing, performance of other processes still deteriorates, though not to same degree

Also, prefer to use global replacement when possible to make better use of available frames

We will discuss other ways of handling this issue that try to avoid these downsides