Concurrent Systems

3D4 ⟷ CS2016

Operating Systems

Andrew Butterfield
ORI.G39, Andrew.Butterfield@scss.tcd.ie

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

with thanks to Mike Brady

Resident Set Management

- How many pages can each process have resident in memory?

Resident Set Management – Factors

- If we allocate less memory to each process, we can fit more processes into memory, thus increasing the probability that there will be at least one process ready to be executed
- If each process has only a small number of pages resident in memory, we will have more page faults
- If we increase the number of resident pages for a process beyond some threshold, the performance increase will become negligible (locality of reference).

Resident Set Management – Policy Types

- Fixed allocation – every process has a fixed number of pages in memory
- Variable allocation – allocation varies over the lifetime of the process

Variable Allocation

- Processes suffering from many page faults can be allocated more page frames
- Processes with few page faults can have their allocation reduced
- This appears to be more powerful than fixed allocation
  - With fixed allocation, we are stuck with the decision made at load-time – the ideal resident set size may vary depending on input to the program
  - And how to we calculate the working set size in the first place
  - However, it is also more expensive to implement

Resident set management

- Variable allocation raises another question …
  - Should the set of candidate pages for replacement be restricted to the pages already allocated to the process that caused the page fault?
  - Two choices:
    - Variable allocation – global scope
    - Variable allocation – local scope

Variable Allocation – Global Scope

- The operating system maintains a list of free page frames
- When a fault occurs, a free frame is added to the resident set of the process
- When there are no free pages available, the page selected for replacement can belong to any process
- This will result in a reduction in the working set size for the process whose page was replaced – might be unfair, there is no way to discriminate between processes

Variable Allocation – Local Scope

- When a process is loaded, it is allocated some number of page frames based, for example, on application type
- When a page fault occurs, a page is selected for replacement from the resident set of the process that caused the fault
- Occasionally, the operating system will re-evaluate the number of page frames allocated to each process

Resident set management

	Local Replacement	**Global Replacement**
Fixed Allocation	● Number of frames allocated to process is fixed ● Page to be replaced is chosen from among pages allocated to the faulting process	
Variable Allocation	● Number of frames allocated to process will vary according to working set requirements ● Page is chosen from among pages allocated to faulting process	● Page to be replaced is chosen from among all available page frames in main memory ● This causes the size of the resident set to vary

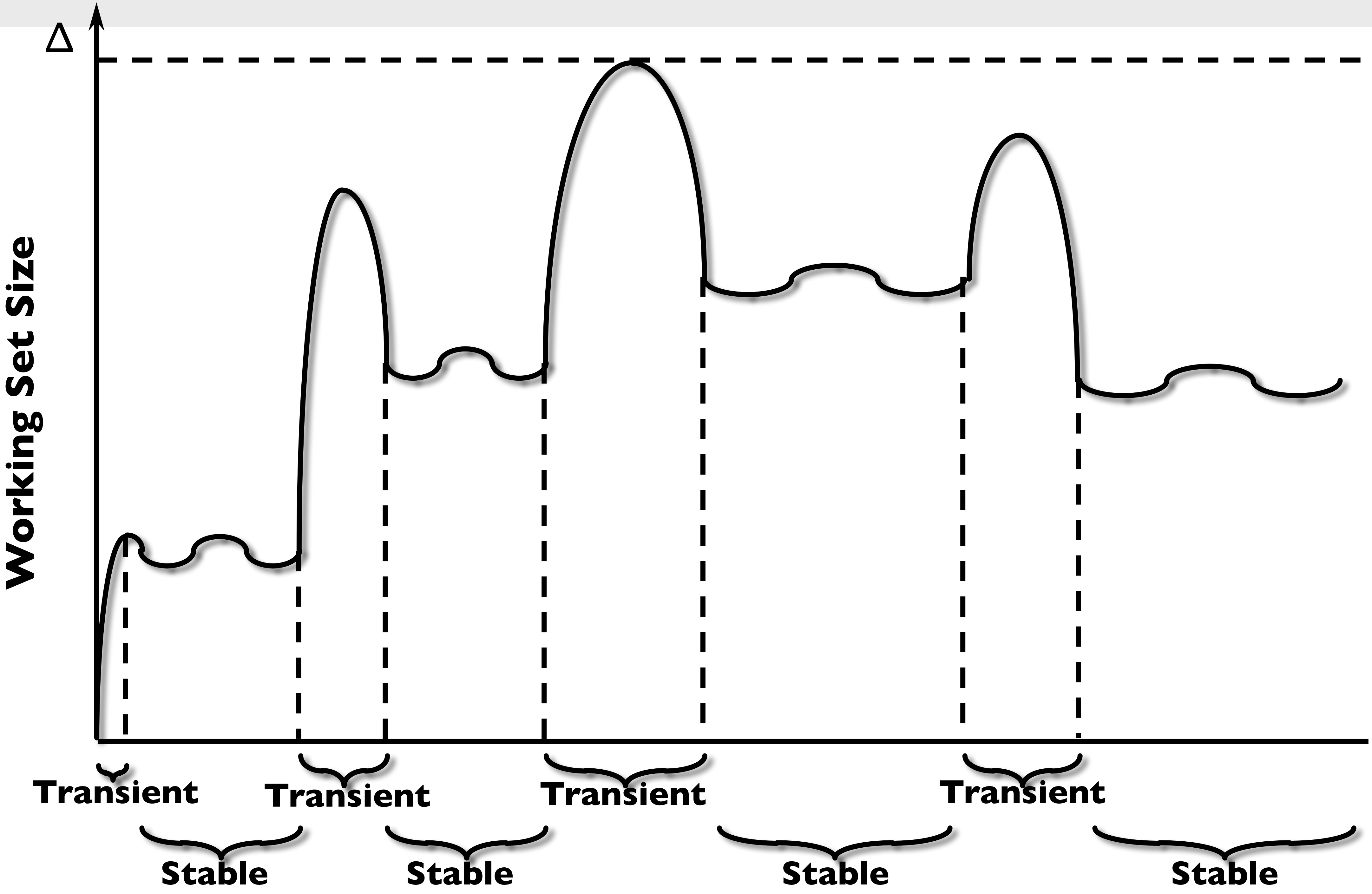
The Working Set

- The *Working Set* idea is that processes should have their working set resident in physical memory to be able to function efficiently.
  - If too few pages allocated (i.e. less than the working set), a process will spend lots of time page-faulting.
  - If too many pages are allocated, (i.e. more than the working set), other processes might be deprived of the physical memory they need.

The Working Set

- The Working Set of a process is the list of pages it has referenced in the last Δ references, proposed by Denning.
- Thus, the Working Set is a "windowed" snapshot of a process' paging referencing.
- If there isn't enough memory for the working sets of all the processes, some processes are temporarily removed from the system.
- Only difficulty is keeping track of the working set.

Resident set management



Resident set management

- How do we decide on a resident set size for a process?
  - We could monitor the working set of each process and periodically remove from the resident set those pages not in the working set

Problems

- Problems
  - Past doesn't predict future – size and membership of working set will change over time
  - Measurement of working set is impractical
  - Optimal value for Δ is unknown

Solution

- We can approximately detect a change in working set size by observing the page fault rate
- If the page fault rate is below some minimum threshold, we can release allocated page frames for use by other processes
- If the page fault rate is above some maximum threshold, we can allocate additional page frames to the process

Page Cleaning

- A Page Cleaning Policy determines when a modified page should be written to secondary memory
- Two approaches
  - Demand cleaning
  - Precleaning

Page Cleaning

- Demand cleaning
  - Pages are written to secondary memory only when they are selected for replacement
- Precleaning
  - Write pages periodically, before they need to be replaced
  - Pages can be written in batches to increase efficiency

Page Cleaning

- Neither approach is ideal
  - If pages are precleaned long before they are replaced, there is a high probability they will have been modified again
  - Demand cleaning results in a longer delay when replacing pages

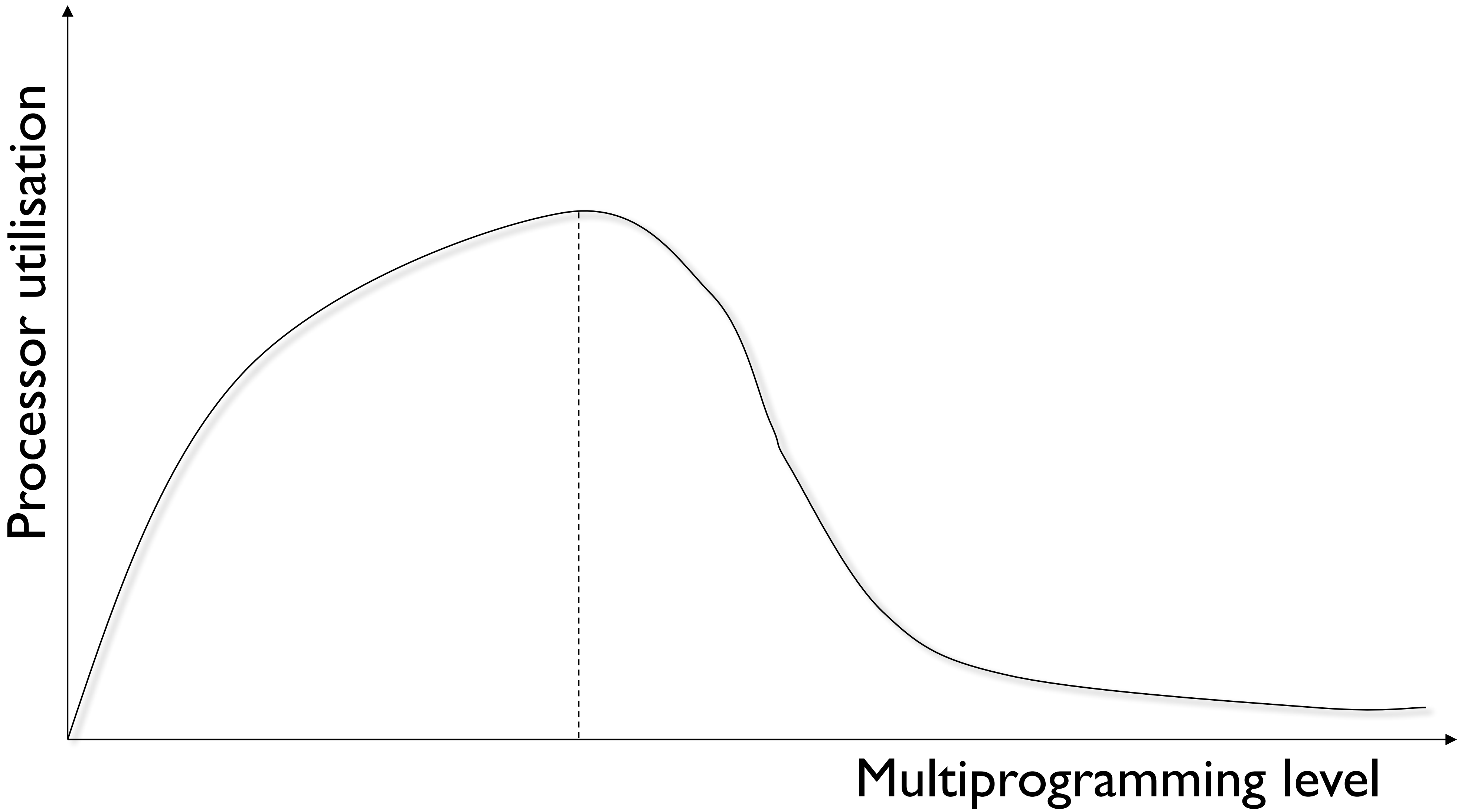
Page Buffering

- A better solution for both page replacement and cleaning
- When a page is deallocated, place it on either an unmodified list or a modified list
- Page on the modified list can periodically be written out to secondary memory and moved to the unmodified list
- Pages on the unmodified list can be reclaimed if they are referenced again before they are allocated to another process

Page Buffering

- Notes
  - “Moving” a page to a modified or unmodified list does not result in copying of data.
  - Instead, the PTE for the page is removed from the page table and placed on one of the lists
  - This approach can be combined with FIFO replacement to improve the performance of FIFO replacement while remaining efficient to implement

Load control



Load Control

- We can use a few different policies
  - Only allow processes whose resident set is sufficiently large to execute
  - Research has shown that when the mean time between faults is equal to the mean time required to process a fault, processor utilisation will be maximised

Load Control – Suspending Processes

- To reduce the degree of multiprogramming, we need to suspend (swap out) one or more resident processes – but which ones do we swap?
  - Lowest priority process
  - Faulting process
  - Last process activated
  - Process with smallest resident set size
  - Largest process