



# **SNS COLLEGE OF TECHNOLOGY**



**Coimbatore-35.**

**An Autonomous Institution**

**Accredited by NBA – AICTE and Accredited by NAAC – UGC with ‘A+’ Grade  
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai**

**COURSE NAME : 19CSB201 – OPERATING SYSTEMS**

**II YEAR/ IV SEMESTER**

**UNIT – III Storage Management**

**Topic: Virtual Memory : Allocation of Frames & Thrashing**

Mr.N.Selvakumar

Assistant Professor

Department of Computer Science and Engineering



# Allocation of Frames

- Each process needs ***minimum*** number of frames
- Example: IBM 370 – 6 pages to handle SS MOVE instruction:
  - instruction is 6 bytes, might span 2 pages
  - 2 pages to handle *from*
  - 2 pages to handle *to*
- ***Maximum*** of course is total frames in the system
- Two major allocation schemes
  - fixed allocation
  - priority allocation
- Many variations



# Fixed Allocation

- Equal allocation – For example, if there are 100 frames (after allocating frames for the OS) and 5 processes, give each process 20 frames
  - Keep some as free frame buffer pool
- Proportional allocation – Allocate according to the size of process
  - Dynamic as degree of multiprogramming, process sizes change

–  $s_i$  = size of process  $p_i$

–  $S = \sum s_i$

–  $m$  = total number of frames

–  $a_i$  = allocation for  $p_i = \frac{s_i}{S} \times m$

$$m = 64$$

$$s_1 = 10$$

$$s_2 = 127$$

$$a_1 = \frac{10}{137} \cdot 62 \gg 4$$

$$a_2 = \frac{127}{137} \cdot 62 \gg 57$$



# Priority Allocation

- Use a proportional allocation scheme using priorities rather than size
- If process  $P_i$  generates a page fault,
  - select for replacement one of its frames
  - select for replacement a frame from a process with lower priority number



# Global vs. Local Allocation

- **Global replacement** – process selects a replacement frame from the set of all frames; one process can take a frame from another
  - But then process execution time can vary greatly
  - But greater throughput so more common
- **Local replacement** – each process selects from only its own set of allocated frames
  - More consistent per-process performance
  - But possibly underutilized memory



# Non-Uniform Memory Access

- So far all memory accessed equally
- Many systems are **NUMA** – speed of access to memory varies
  - Consider system boards containing CPUs and memory, interconnected over a system bus
- Optimal performance comes from allocating memory “close to” the CPU on which the thread is scheduled
  - And modifying the scheduler to schedule the thread on the same system board when possible
  - Solved by Solaris by creating **lgroups**
    - Structure to track CPU / Memory low latency groups
    - Used my schedule and pager
    - When possible schedule all threads of a process and allocate all memory for that process within the lgroup



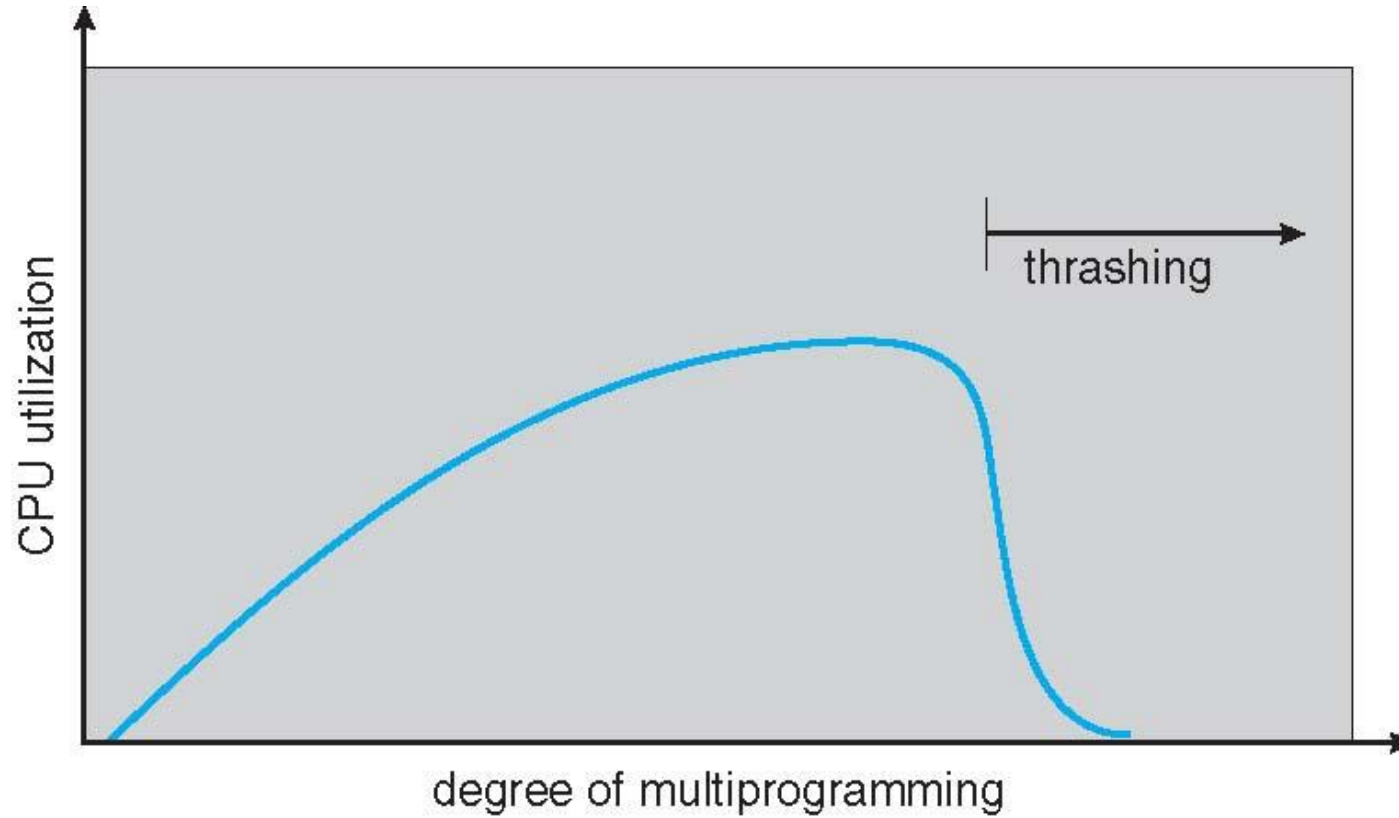
# Thrashing



- If a process does not have “enough” pages, the page-fault rate is very high
  - Page fault to get page
  - Replace existing frame
  - But quickly need replaced frame back
  - This leads to:
    - Low CPU utilization
    - Operating system thinking that it needs to increase the degree of multiprogramming
    - Another process added to the system
- **Thrashing**  $\equiv$  a process is busy swapping pages in and out



# Thrashing (Cont.)





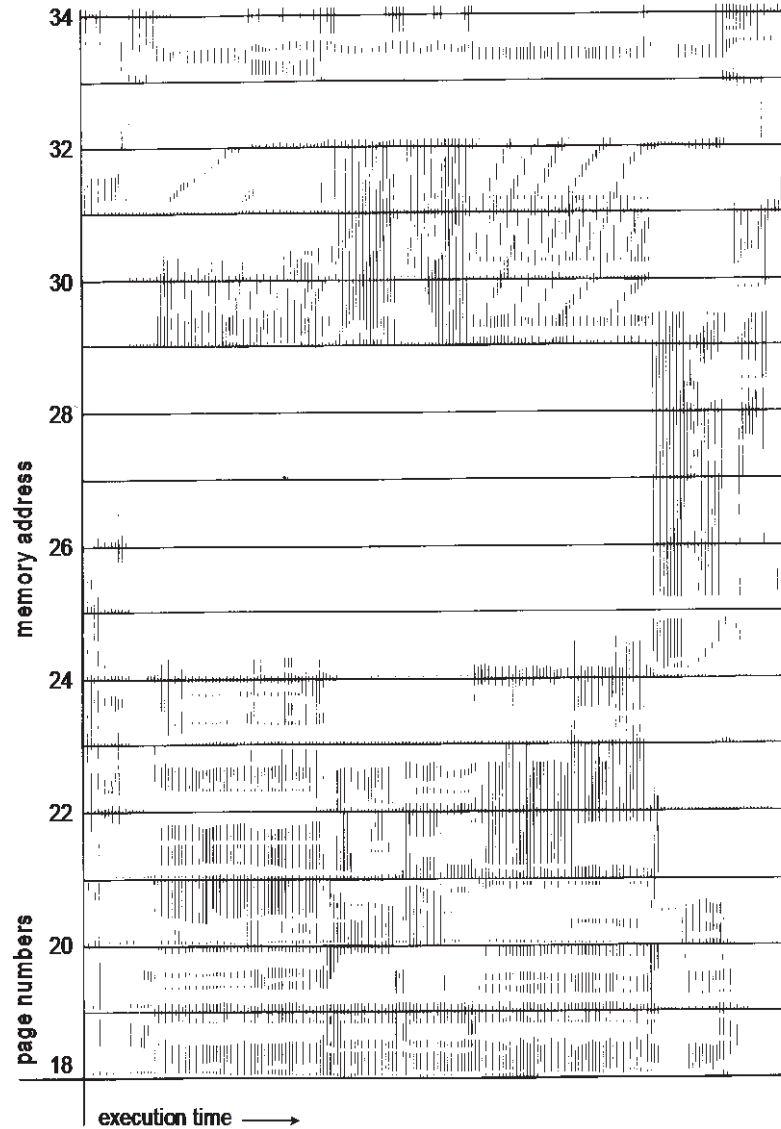


# Demand Paging and Thrashing

- Why does demand paging work?  
**Locality model**
  - Process migrates from one locality to another
  - Localities may overlap
- Why does thrashing occur?  
 $\Sigma$  size of locality > total memory size
  - Limit effects by using local or priority page replacement



# Locality In A Memory-Reference Pattern



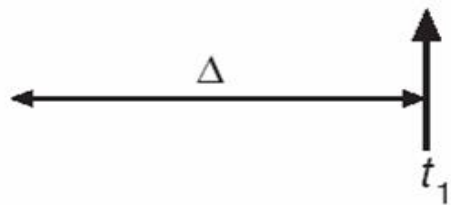


# Working-Set Model

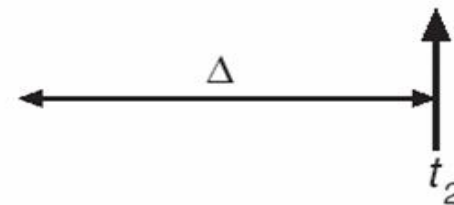
- $\Delta \equiv$  working-set window  $\equiv$  a fixed number of page references  
Example: 10,000 instructions
- $WSS_i$  (working set of Process  $P_i$ ) =  
total number of pages referenced in the most recent  $\Delta$  (varies in time)
  - if  $\Delta$  too small will not encompass entire locality
  - if  $\Delta$  too large will encompass several localities
  - if  $\Delta = \infty \Rightarrow$  will encompass entire program
- $D = \sum WSS_i \equiv$  total demand frames
  - Approximation of locality
- if  $D > m \Rightarrow$  Thrashing
- Policy if  $D > m$ , then **suspend or swap out one of the processes**

page reference table

. . . 2 6 1 5 7 7 7 5 1 6 2 3 4 1 2 3 4 4 4 3 4 3 4 4 4 4 1 3 2 3 4 4 4 3 4 4 4 . . .



$$WS(t_1) = \{1, 2, 5, 6, 7\}$$



$$WS(t_2) = \{3, 4\}$$



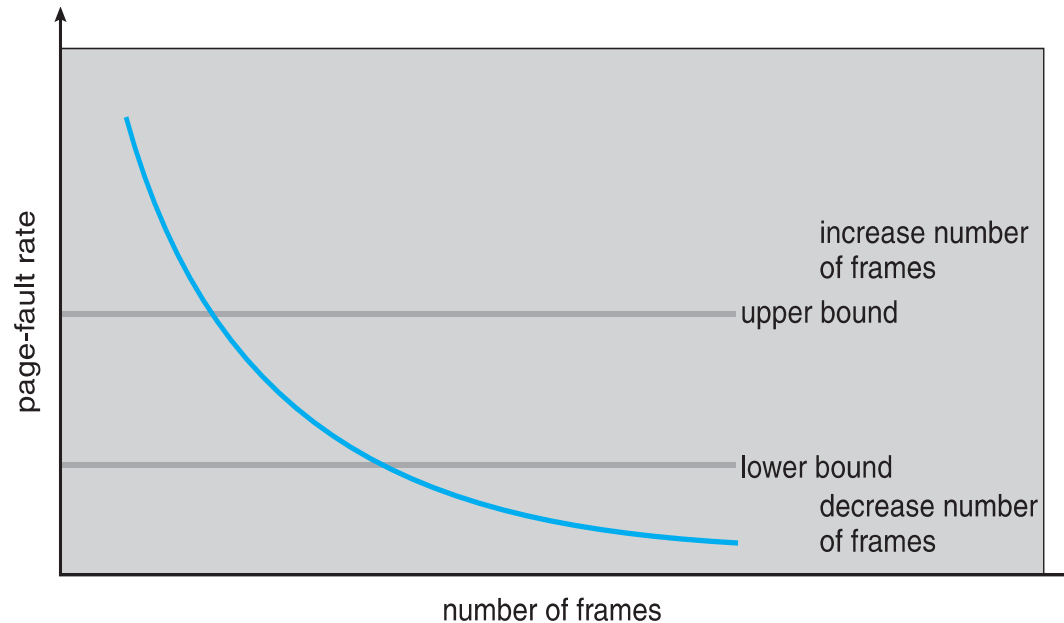
# Keeping Track of the Working Set

- Approximate with interval timer + a reference bit
- Example:  $\Delta = 10,000$ 
  - Timer interrupts after every 5000 time units
  - Keep in memory 2 bits for each page
  - Whenever a timer interrupts copy and sets the values of all reference bits to 0
  - If one of the bits in memory = 1  $\Rightarrow$  page in working set
- Why is this not completely accurate?
- Improvement = 10 bits and interrupt every 1000 time units



# Page-Fault Frequency

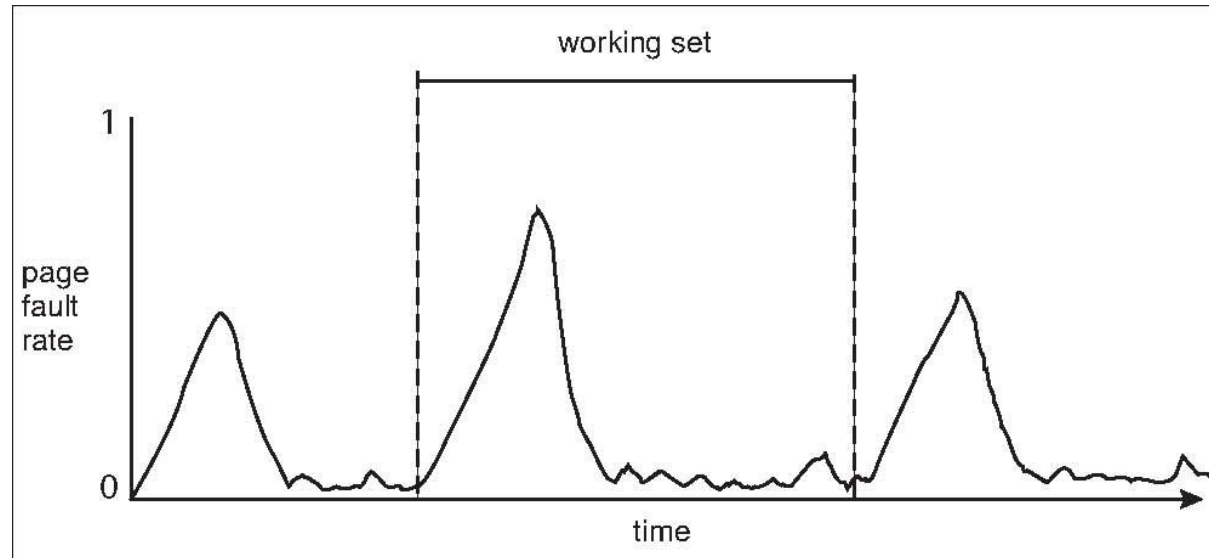
- More direct approach than WSS
- Establish “acceptable” **page-fault frequency (PFF)** rate and use local replacement policy
  - If actual rate too low, process loses frame
  - If actual rate too high, process gains frame





# Working Sets and Page Fault Rates

- n Direct relationship between working set of a process and its page-fault rate
- n Working set changes over time
- n Peaks and valleys over time





# REFERENCES

## TEXT BOOKS:

- T1 Silberschatz, Galvin, and Gagne, “Operating System Concepts”, Ninth Edition, Wiley India Pvt Ltd, 2009.)
- T2. Andrew S. Tanenbaum, “Modern Operating Systems”, Fourth Edition, Pearson Education, 2010

## REFERENCES:

- R1 Gary Nutt, “Operating Systems”, Third Edition, Pearson Education, 2004.
- R2 Harvey M. Deitel, “Operating Systems”, Third Edition, Pearson Education, 2004.
- R3 Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, “Operating System Concepts”, 9th Edition, John Wiley and Sons Inc., 2012.
- R4. William Stallings, “Operating Systems – Internals and Design Principles”, 7th Edition, Prentice Hall, 2011



