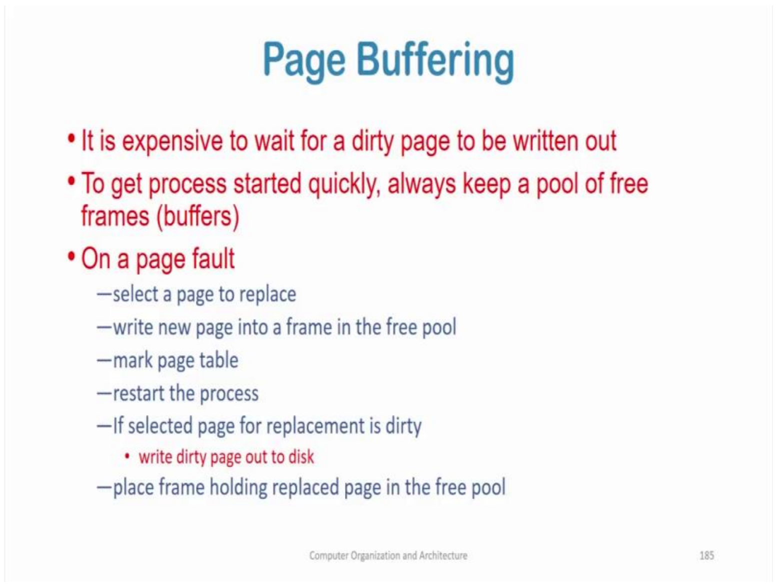
Computer Organization and Architecture: A Pedagogical Aspect
Prof. Jatindra Kr. Deka
Dr. Santosh Biswas
Dr. Arnab Sarkar
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Lecture – 11
Page Frame Allocation and Thrashing

(Refer Slide Time: 00:26)



Page Buffering

- It is expensive to wait for a dirty page to be written out
- To get process started quickly, always keep a pool of free frames (buffers)
- On a page fault
  - —select a page to replace
  - —write new page into a frame in the free pool
  - —mark page table
  - —restart the process
  - —If selected page for replacement is dirty
    - write dirty page out to disk
  - —place frame holding replaced page in the free pool

Computer Organization and Architecture 185

Welcome, in this lecture we will continue our discussion with paging. We have looking at schemes, to improve the performance of paging; in this we looked in the last lecture we looked at page replacement algorithms, where better page replacement algorithms improve the performance of paging. Then we looked at the scheme of page buffering; which is related to the issue that when during replacement when we have to write to a dirty page, that dirty page has to be first written to disk and the system has to wait for this dirty page to be written out to disk and then only the page that is required can be brought into the memory frame.

Now to avoid this waiting time, we keep a pool of free pages at any point in time so when we need to replace we as before we select a victim page, if that page is dirty we will write we will write it to the disk, but, but what we do is that instead of writing to this dirty, we select the victim page fine, but before writing this victim page to disk that victim page if it is dirty to disk what we do is, we select a page from the free pool and allocate this for replacement and therefore, the system does not have to wait for this frame to be for the victim page to be written to be to be written to disk first before replacement so whenever there is a replacement I choose a frame from the free frame pool and allocate this for replacement.

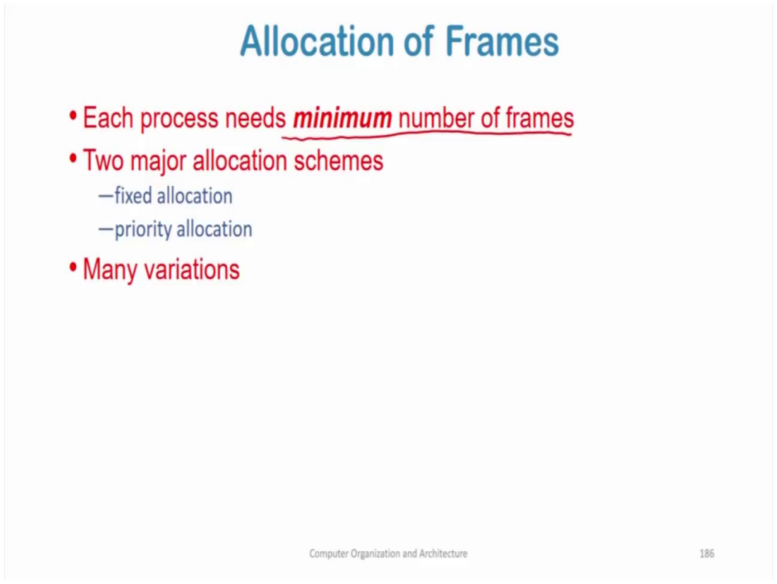
Now, after I have given this and the data has been written from the secondary storage into this frame in the free frame pool and the process has been restarted subsequent to service of the page-fault; then the I/O channel is free again and then what we do this victim page is written to the disk and then after this victim page is written to the disk, this page modified bit or the dirty bit for this page is again reset and then this frame is again added to the free frame pool.

Now this basic scheme was an extension to this basic scheme is that, what we do is that whenever I/O channel is free I find out a dirty page and I write it down into the disk so I all I maintain a queue of all the dirty pages that are currently there in memory and whenever the I/O channel is free, I slowly in my in the in the when the I/O channel is free I write it down into the disk and add these pages into the free frame pool. Another small extension to this scheme is that so I whenever I have written a page a dirty page so the pre in the previous scheme we saw, that I took a dirty page and I wrote it on to the disk in my free time and then I added it to the free frame pool.

Now, I have added this to the free frame pool, I have added this frame into the free frame pool, but when I have written I have not destroyed the contents of this frame; I have written the contents of the frame into the secondary storage and I have reset the dirty bit fine and I have also added this page in page frame into the free frame pool, but this does not mean that I have destroyed the contents of this page.

So therefore, if it so happens that my processor needs a page, which is there in the free frame pool and the then instead of going to the storage I can directly take that page from the free frame pool itself; again this issue what we are trying to say is that I have a page for which its dirty bit is on therefore, according to the earlier scheme in when the I/O channel is free, I/O processor is free I have written this page into the into the free frame pool sorry into the disk, and then I have added this page into the free frame pool, but I addition of this page frame into the free frame pool does not destroy the contents of this page so by chance in sub during subsequent accesses the processor needs this page, this page of is there in the free frame pool; I directly take it from the free frame pool, I don’t have to go to the secondary memory and therefore, I save a page-fault so whenever I have to access after I will see the free frame pool first, if the page is there in the free frame pool I have to I will I will take it from there itself instead of going to the disk.

(Refer Slide Time: 05:30)



Allocation of Frames

- Each process needs ***minimum*** number of frames
- Two major allocation schemes
  - —fixed allocation
  - —priority allocation
- Many variations

Computer Organization and Architecture 186

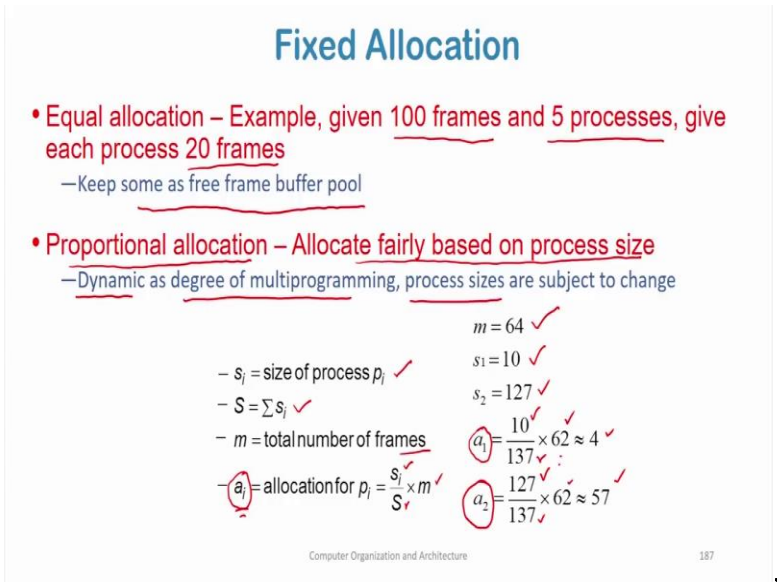
Then we look at schemes for allocation of frames. Till now we have been looking at schemes where I have the entire set of frames in the in the memory and any page can be replaced by any process so I have a global set of frames and the replacement algorithm will search from this global set of frames to find a frame for replacement and give me.

Now, this may hit the performance of processes; so to avoid that what we do is that we often allocate a minimum number of frames to each process so each process needs a minimum number of frames. Now why does it need for example, to execute an instruction it will need one page from in which the instruction is there; it will also need data, it may need to read data, it may need to write data so whenever it needs to read data, it will it will have to read from data from the pages in memory which contain that data it so these pages corresponding to this process have to be kept in main memory; page which contains the instruction needs to be kept in memory.

The pages which are there which contains the data needs to be kept in memory and the pages where the data needs to be written to needs to be kept in memory so therefore, there is a minimum set of active pages with which a process is at any time is working on. This set of active pages it must keep in memory so therefore, each process at any given point in time will require a minimum number of frames to be allocated to it, because these frames will contain the pages, active pages that this process needs at any time. If these minimum number of frames is not given to the process; the process will frequently page-fault ok so to avoid that each process must be allocated a minimum number of frames to ensure a minimum level of performance.

Now, allocation schemes are typically of two types fixed allocation and priority based allocation and there are many variations of this.

(Refer Slide Time: 07:52)



Fixed Allocation

- Equal allocation – Example, given 100 frames and 5 processes, give each process 20 frames
  - Keep some as free frame buffer pool
- Proportional allocation – Allocate fairly based on process size
  - Dynamic as degree of multiprogramming, process sizes are subject to change

  - s_i = size of process p_i
  - $S = \sum s_i$
  - m = total number of frames
  - a_i = allocation for $p_i = \frac{s_i}{S} \times m$

$m = 64$

$s_1 = 10$

$s_2 = 127$

$a_1 = \frac{10}{137} \times 62 \approx 4$

$a_2 = \frac{127}{137} \times 62 \approx 57$

Computer Organization and Architecture 187

In a fixed allocation scheme what happens? Suppose this is basically equal allocation for example, if you are given 100 page frames, if you are given 100 page frames and we have 5 processes, we give 20 frames to each process so just you know the number of processes and you know the number of frames; number of frames divided by number of processes and therefore, that many frames is allocated to each process so in addition to this you can keep some free frame buffer pool for each process. Now against as against this equal fixed allocation, we also have Proportional allocation so what does proportional allocation do? It tries to fairly allocate frames to each process based on process size.

So what we are trying to do? We are trying to find out how many frames should be allocated to each process; the first one looked at fixed allocation, where all processes gets depending on the degree of multi programming depending on how many processes can co-exist together, co-execute together in the processor and so that many processes need to be kept in main memory when I know this; when I know that therefore, the degree of multi programming I divide the frames equally among these processes, irrespective of the characteristics of each process ok.

So, therefore if the process is big a small process which requires a small number of pages at a given time and a big process let us say which requires a large number of frames at a given time, we will get the same, same number of pages so the small process will do very nicely and the large process may be maybe doing bad in performance because, the large process has an insufficient number of frames allocated to it whereas, the small process has surplus frames allocated to it so the small process does very well, but the large process does not as well because the number of frames allocated to it is not as it requires.

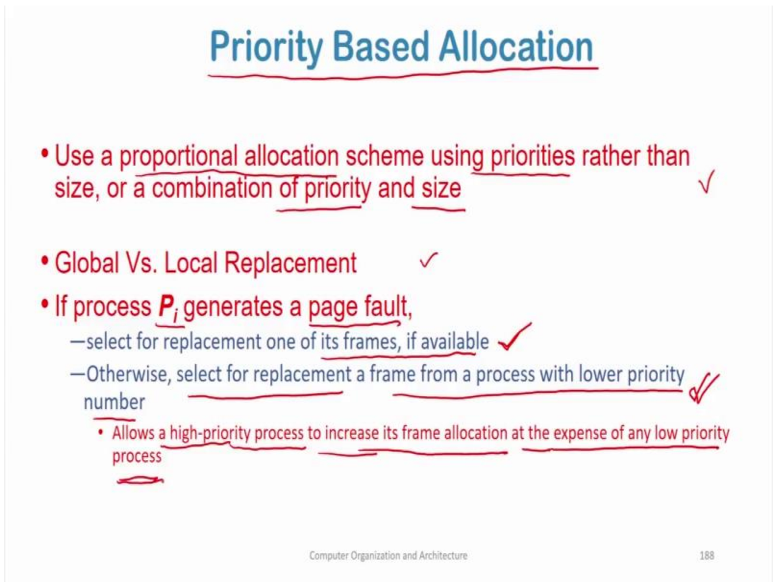
Now, to overcome this problem, we looked at proportional allocation; what do we do in proportional allocation? We allocate the frames to processes based on the process size ok. This is dynamic this the so the number this is dynamic as the degree of multi programming process sizes are subject to change. The size of a process; that means, the size of a process in main memory at a given time, will vary in time; the degree of multi program that is the that is the number of processes that are co-executing together in the system at a given time will also vary; therefore, the number of frames allocated to a process will also vary in proportional allocation.

For example, let us say m is the total number of frames that are there in memory at a given time. Let s_i be the size of process P_i and S is the summation of the size of individual processes; so the size of process in terms of the number of pages it requires; m is the total number of frames and how does this allocation, how many how many frames with will process will process P_i get, it will get a_i number of frames, how is a_i calculated? It is calculated as small s_i the number of frames it requires, S divided by, S the total demand for $s \times m$ the total number of frames available in memory $\frac{s_i}{S} \times m$.

So, let us say if we have two processes the first one requires 10 pages, the second one is a big process requires 127 pages so S becomes 137 so for the first process the number of frames that the first process P_1 will get will be a_1 and how is a_1 calculated? $\frac{10}{137} \times 62 = 4$ ok why 62? Because say let us say we have allocated 2 frames for the operating system and the rest of the frames are allocated to user processes let us say.

And then how many frames will the second process get? The second process will get $\frac{127}{137} \times 62$ which is approximately equal to 57 so here we are allocating frames based on the sizes proportionally we are allocating frames proportional to proportionally to processes based on the size of the processes in terms of the pages it requires.

(Refer Slide Time: 12:31)



Priority Based Allocation

- Use a proportional allocation scheme using priorities rather than size, or a combination of priority and size ✓
- Global Vs. Local Replacement ✓
- If process P_i generates a page fault,
  - select for replacement one of its frames, if available ✓
  - Otherwise, select for replacement a frame from a process with lower priority number ✓
    - Allows a high-priority process to increase its frame allocation at the expense of any low priority process

Computer Organization and Architecture 188

Now, the use of a priority based allocation scheme using now it may so happen; now when we are using this priority based allocation scheme, we are ignoring priorities of processes. Now we may so want that we will allocate a higher number of frames to a high priority process while making low priority process suffer, because if for a we want higher performance for the high priority processes with respect to the low priority processes. Now to take care of this, we can use to allow this such priority based allocation; what we can do is we can use a proportional scheme using priorities; that means, instead of instead of process sizes instead of the sizes of processes in terms of the number of pages it requires; we can proportionally allocate frames to processes based on the priority of the processes. So if you proportionally allocate based on priorities instead of the sizes of the processes we are we are doing a priority based allocation; however, if we are only doing a priority based allocation and looking at nothing else we are completely ignoring the size of the processes.

So, we can also do a priority based proportional allocation; based on a combination of priority and size, instead of going aggressively for only priorities. So a combination of priority proportional allocation of frames to processes based on a combination of priority and size may be will balances both priority as well as the requirements of the processes. It takes care of both the priority of the processes and the requirements of the processes in number of frames; it tries to look at both ends and do a better allocation ok. So, this is all with priority based allocation; the other issue to look is doing a global versus local allocation of frames.

Now, in a local frame allocation we said that we have allocated till now, we were the discussion that we have been doing deals with this; how many frames will I allocate to each frames? Let us say I have allocated us a certain number of frames to each process currently under execution; now how will the replacement work in this scenario? Now there are two there are two replacement schemes, there are two ways to look at replacement after we have allocated frames to each process. One is local replacement in which page replacement will only occur within pages which are allocated within frames, which are allocated to a given process. So let us say I have a process P_1 and I know that I have allocated a set of 10 frames to this process. Now during replacement I will only consider these 10 frames when I am doing a replacement.

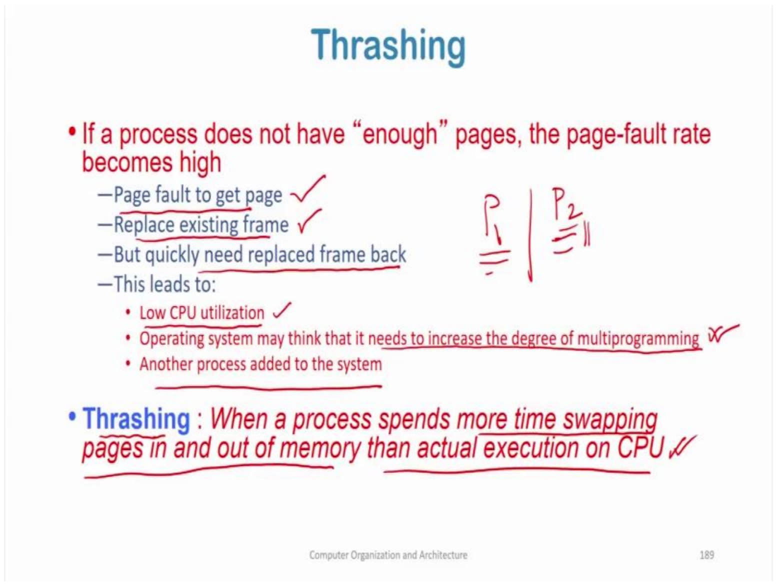
I will not consider for allocation frames allocated to other processes for this process. This is a completely local replacement strategy; that means, when replacing I will only consider pages of these frames of this process for replacement, against global allocation what will wait what will I do in global replacement? So during replacement I can look at the entire set of processes that I have and choose any frame for replacement for this process. So let us say in global replacement I have a process P_1 and then what happened this P_1 has done a page-fault and I require to replace a page, while during replacement I will not see whether I will not replace this page into a frame of P_1 I may not; I will consider all frames in my entire memory and I can choose any frame for replacing this page, and then this frame will be will come into will come into P_1.

So P_1 has done a page-fault, P_1 contains a set of 10 frames. Now in global replacement I am not considering that I will only use one of these 10 page frames for replacement. I will consider all frames that are in memory at a given time and I can choose any frame for replacing this page and after this frame will now be added to the frame pool of P_1 ok. So this will happen in global replacement; in right in completely global replacement and against completely local replacement.

A third strategy is this; a priority based third strategy is this. Let us say if a process P_i generates a page-fault; select for replacement one of its frames is available, so this is local replacement. So you try to do a local replacement if you have a free frame in your if you have a free frame, in the pool of frames that are allocated to P_i, use that frame for replacement. If you do not have a free frame; what do you do? Otherwise you try to select for replacement of frame from a process with lower priority number. So this process P_i has a priority or importance either you try to get frame, local replacement from local replacement from a frame of this process or you choose a frame from a process with lower priority; what does this allow? It allows a high priority process to increase its frame allocation at the expense of low priority processes.

Now when you, when you allow this semi global allocation what you basically do is that the number of frames after you do this replacement; so basically you are stealing a frame from a lower priority process after replacement. And after replacement this stolen frame will be added to the frame list of P_i; so the number of frames which are allocated to the lower priority process reduces and the number of frames allocated to P_i increases. So therefore, using this scheme this scheme allows a high priority process say P_i here to increase its frame allocation at the expense of low priority processes.

(Refer Slide Time: 19:14)



Thrashing

- If a process does not have "enough" pages, the page-fault rate becomes high
  - —Page fault to get page ✓
  - —Replace existing frame ✓
  - —But quickly need replaced frame back
  - —This leads to:
    - Low CPU utilization ✓
    - Operating system may think that it needs to increase the degree of multiprogramming
    - Another process added to the system
- **Thrashing** : *When a process spends more time swapping pages in and out of memory than actual execution on CPU*

Computer Organization and Architecture 189

Now, we come to the next issue which is thrashing; now we are saying that as we have discussed each process requires a minimum number of active pages, when it is executing at a given time. This is required for the instructions so the pages that it requires for the instruction that it is executing the data it requires for these instructions and the data it requires to write to.

So these active pages is at least required for the at a given time; so these active pages are required by the process and this process the number of active pages that will be required by the process at a given time is a characteristic of the process itself; what how it is using data and how it is using, how it is executing data, how it is executing instructions at a given time; that will determine the number of pages that it requires.

Now we said that we have allocated a certain number of frames to each process; now if a process does not have enough pages ok, does not have enough pages then page-fault rate becomes high; what do we mean by does not have enough pages? So in the frames that are allocated to it so sufficient frames are not allocated to this process, for its active pages to be in memory; that is why we are saying that the process does not have enough pages in memory. So if it does not have active pages if it cannot keep all its active pages in memory then the process will frequently page-fault. A page that it requires is not there in main memory and therefore, it has to be brought from secondary memory.

Now, we need to do a page-fault to get this page; now for that because we have to replace this an existing page so we need to do a page-fault to get this page; then we need to replace an existing frame, because it does not have enough frames in memory all its frames all its pages that are there in its frames are in active use, it needs to replace an existing frame, but because all these pages are in active use; it quickly needs to replace the frame back. It is replacing it is replacing a page which is in active use; to get the page that it to get the page that it immediately needs, to get a page that is not there in main memory and it immediately needs. It is replacing a page which is also in active use.

Now because this page in active use very quickly in future that page which it has replaced and put in secondary memory will be required again and this will lead to low CPU utilization, because the process will spend will be spending more time in serving the servicing the page-faults. This will lead to low CPU utilization the operating system and a problem, another problem is that especially for early operating systems this was a grave problem; that the operating system may be thinking that it needs to increase the degree of multiprogramming why?.

Because the CPU utilization has reduced why has the CPU utilization reduced? Because processes are doing page-fault servicing more than it is executing and why is that so? Even if let us say we are using a global replacement algorithm. One page does not have enough pages in memory and therefore, it is it is continuously doing a page-fault and when which is doing a page-fault it may be encroaching into frames allocated to other processes, reducing the number of frames allocated to those processes; moreover what happens is that the I/O processor remains busy servicing page-faults; so when other processes it even if other processes have enough frames in memory; whenever it requires a page from the secondary storage it has to wait because the other process has completely consumed I/O processor.

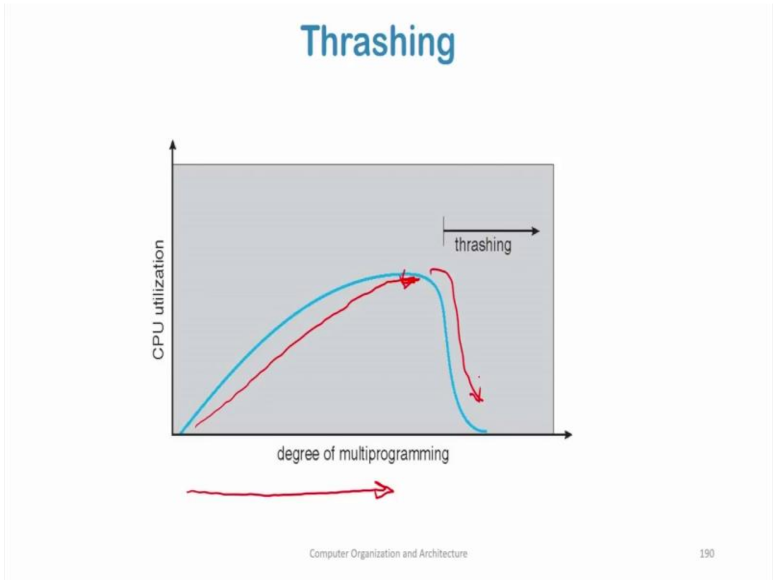
Suppose I have a process P_1 which is doing which is continuously doing page-faults because it does not have enough frames in memory and I have process P_2 which for which it has enough frames and to keep its active pages. This one P_2 has enough frames to P_2 has enough frames to keep all its active pages in memory P_1 does not have enough frames to keep all its all its active pages in memory P_1 does not have. Now P_1 is continuously doing a page-fault and in doing so it is keeping the I/O processor busy. Now when P_2 whenever it requires one page and it does one page-fault it will it will find the I/O processor busy because, P_1 is has completely consumed the I/O processor and therefore, the performance of P_2 is also going to reduce.

Now in this way the whole CPU utilization the execution in the CPU is going to reduce because everybody is waiting for some of its pages to be brought from secondary memory. All processes in memory are waiting for some of its pages to be brought from secondary memory because I/O processor is busy. Now if the operating system fails to understand this situation, when it sees the CPU utilization low; it may think that it does not have enough processes in main memory to execute in the CPU. So it will push in another process into the main memory by increasing the degree of multi programming; when one more process gets into the main memory and it tries to co-execute it will also encroach into frames of allocated to other processes. It will take away frames allocated to other processes and the performance of the system will further degrade.

So another process when added to the system will further degrade the total performance of this system and CPU utilization will dip further, because of two reasons because the new process will again ask for pages from secondary memory and also other processes have for other processes that the number of frames allocated to them have reduced because the new process have to be given a minimum number of frames. Due to both these reasons the CPU utilization is going to reduce further and therefore, the CPU utilization will finally, plunge.

So, what is the formal definition of thrashing? When a process spends more time swapping pages in and out of memory than actual execution on the CPU, we say that the system is undergoing thrashing or the process is undergoing thrashing; when a process spends more time in swapping pages in and out than actual execution on the CPU then we say that the process is thrashing. The system will also undergo a complete if this process is thrashing for a long time and the thrashing is severe it will also degrade the performance of the entire system as we discussed.

(Refer Slide Time: 26:40)



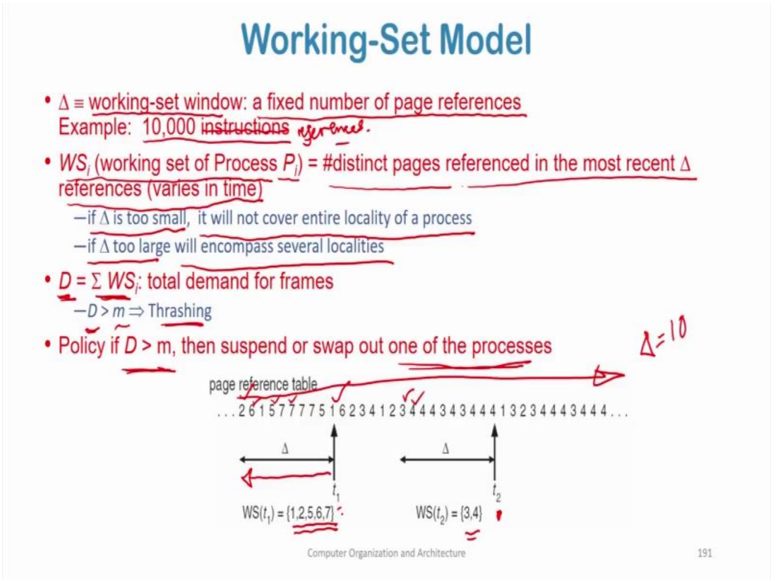
Thrashing

Now, so therefore, when we increase CPU utilization or we increase the degree of multi programming up to a certain extent; we are we increase CPU utilization; then after that more the processes will start spending more time in paging than in execution, because there are so many processes have come are co-existing in the main memory; not enough frames could be allocated to keep the active pages of these processes in main memory. So when we are increasing the degree of multi programming; the number of processes that are co-executing on the CPU are increasing and all these processes will take a share of the frames allocated in main memory.

So why because each of these processes will be allocated a certain number of frames; now therefore, when the number of processes increase the number of frames allocated to each process reduces, when the number of frames reduces after a certain time, after a certain increase in the degree of multi programming, there will not be enough frames for each process to keep all its active pages. Now for a given process when it when it is not able to keep all its active pages in memory; when the number of frames allocated to a process is not is not sufficient to keep all its active pages in memory then the process will start thrashing.

When the process starts thrashing and when more and more degree of multi programming increases, more and more processes starts thrashing and this will finally, bring down the entire performance of the system; which means that the CPU utilization will reduce. CPU utilization meaning the percentage of time, that the CPU is spending in executing instructions. So that will increase up to a certain extent, when the degree of multiprogramming is increased why? Because up to this case I am keeping more processes and therefore, I am keeping more processes in memory and I am utilizing the CPU to a higher extent, because more number of processes are there. So even if some other processes are being suspended for memory I/O etcetera whatever; some other processes will be there for utilizing the CPU and the CPU will be efficiently sharing itself among multiplexing itself among all these processes, but after a certain extent the number of frames allocated to each of these processes decreases and then thrashing starts and the CPU utilization plunges.

(Refer Slide Time: 29:25)



Working-Set Model

- $\Delta \equiv$ working-set window: a fixed number of page references
  Example: 10,000 ~~instructions~~ references.
- WS_i (working set of Process P_i) = #distinct pages referenced in the most recent Δ references (varies in time)
  - if Δ is too small, it will not cover entire locality of a process
  - if Δ too large will encompass several localities
- $D = \Sigma\, WS_i$: total demand for frames
  - $D > m \Rightarrow$ Thrashing
- Policy if $D > m$, then suspend or swap out one of the processes

$\Delta = 10$

Computer Organization and Architecture 191

Now, you cannot say that using any such model, you cannot say that a fixed number a certain fixed number of frames is sufficient for a given process; why because the active pages that the that that a particular process requires is going to vary; why because depending on the locality of reference for a process will vary depending on what it is doing at a given time.

Now, for example, let us say at a given time it is continuously within a loop and is executing a certain set of instructions and a certain set of data it is operating on a matrix. So it requires a certain number of pages for that matrix and a certain number of pages for the instructions within that loop. So let us say that at a given time a process is executing a loop; so this loop contains a set of instructions and let us say in this loop it is executing on a matrix. So if I am able to keep enough frames, enough pages it if I have enough frames for this process to keep the pages of this matrix that are actively being used and enough pages for the instructions in the loop then I will not suffer a lot of page-faults; however, if I am not able to keep the if the number of frames are not sufficient to keep the data corresponding to the matrix or the data corresponding to the instructions; in the main memory pages if I do not have that many sufficient number of frames, then there is going to be a lot of page-faults need possibly leading to thrashing.

Now this when now if the if now the program comes out of this loop and does something else the requirements of its active pages corresponding to data as well as for instructions is going to change. So therefore, over time a programs characteristics characteristic in terms of the active pages that is required is going to change and this is captured by something called the working set model. So for the working set model we first define a working set window.

A working set window is a fixed number of page references in the past; let us say you look at 1000 page reference, 10000 page references in the past. So, Δ equals to 10000 instructions in the past ok. So WS_i is a working set of process i; Let us say WS_i is the working set of process I. Its defined as the number of distinct pages referenced in the most recent Δ references. So let us say these are not 10000 instructions, but 10000 references.

So, in the last 10000 references how many distinct pages were referenced, this will be called the working set of the process. For example, let us say that this is the sequence of references for a given process. Now if the working set Δ equals to 10; then at time t_1 I am going to look at the last 10 in 10 references and I want to find out how many distinct page pages were referenced in the last 10 page references. For this case the number of distinct pages that were referenced in the last 10 references are 1, 2, 6, 5 and 7. So 1, 2, 5, 6, 7 are the distinct pages referenced in the last 10 references.

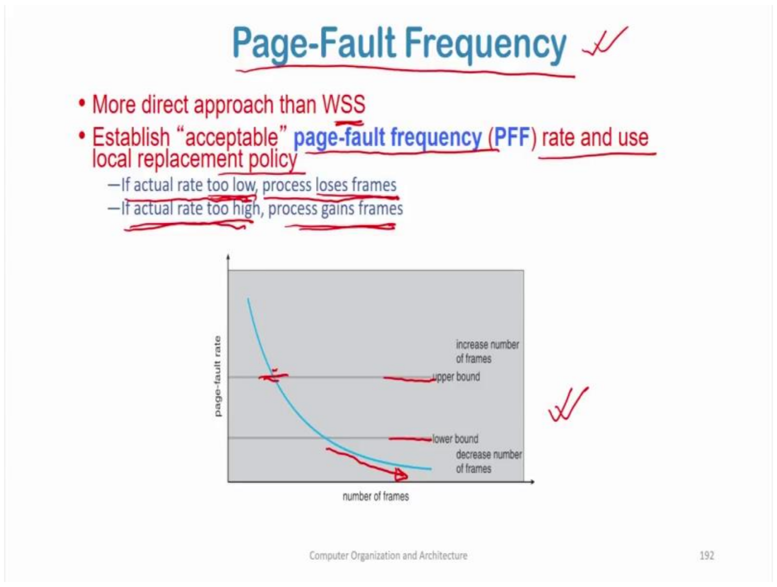
So I will say that the working set of this processor process is 5; which is a measure of the number of active pages that it requires at this point in time; given a working set window Δ of 10. Now let us say after a certain amount of time at time t_2 this is what happened at time t1. Let us say now at time t_2 what has happened the characteristic of the program has changed. Now the program is doing something else and it is continuously referencing pages only 3 and 4. So in the last ten page references at time t_2; what is happening is that it has the unique or the distinct pages that it has referenced is only 3 and 4. So the working set at time t_2 is 3 and 4; while the working set at time tt_1 is 1, 2, 5, 6, 7. So the demand for pages varies depending on time and is defined and is roughly defined by the working set at a given time.

Now, if this working set data working set window is too small; it will not cover the entire locality of a process what do we mean by the locality of a process by the locality of a process we will mean at a given time; what is does what is it doing and what are the active pages it is referencing, that will define the locality of the process. Now if the Δ is if this value of Δ is very small it will not cover the entire locality of the process. So let us say even if Δ says that it requires the working set size is 5 and if we then give 5 frames for it, it will still encounter a lot of faults because the it has not cove Δ is so small; that it has not covered its entire locality and the other hand if Δ is too large it will encounter multiple localities and then I will I will uselessly allocate more frames to this process than it actually requires.

May be at the expense of other processes ok because other processes will possibly not get enough frames may not get enough frames. So if the total demand for frames will demand D will be given by the summation of the working sets of all processes and we can say that the system will be thrashing if the total demand is greater than m which is the total number of frames available in my main memory. If the total demand for frames which is basically roughly defined by the working set of the processes as we said. So if the total demand which is the summation of the working sets of all processes is more than the total number of frames available in main memory; the system is going to experience thrashing.

So, what do you do when D when the total demand is greater than the number of processes that you have; you suspend or swap out one of the processes. So you go on suspending processes until you have sufficient number of frames for each process, possibly you will suspend the low priority processes to start with to allow high priority processes to have better performance.

(Refer Slide Time: 36:30)



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 frames
  - If actual rate too high, process gains frames

page-fault rate
increase number of frames
upper bound
lower bound
decrease number of frames
number of frames

Computer Organization and Architecture 192

Now working set size has been a very successful model, but a more direct approach is to directly work on the basis of how many know what is the page-fault frequency that a particular process is currently experiencing. Working set size is a fine model, but a more direct approach than working set size is to measure the page-fault frequency at a given time and to establish to at a given time to for each process to have an acceptable page-fault frequency for each process. For example, the page-fault frequency for a low priority process the acceptable page-fault frequency for a low priority process may be higher than the acceptable page-fault frequency for a high priority process; however, for each process we can set an acceptable page-fault frequency rate ok.

So establish acceptable page-fault frequency rate and use local replacement policy. If the actual rate is too low the process loses frames, we deduct frames from this process. So if the if the if the page-fault frequency for a process is lower than a given threshold; that means, what it does not require frames allocated to it. Then we can take frames from this process and give it to somebody else.

So if the actual page-fault frequency rate is very low, the process loses frames. We take frames from this process and give it to other processes. If the actual page-fault rate frequency is very high then the process gains frames. If the actual page-fault frequency is very high it means that this process possibly does not have enough frames to keep all its active pages therefore, it needs more frames for it. Now it can be given from other processes to it.

So, this figure shows us that for a process I have an acceptable upper bound and I have and a lower bound. If the page-fault frequency decreases than this lower bound; what we do is that we decrease the number of frames allocated to this process when the when the when the number of when the page-fault frequency for this process is higher than this bound; that means, it that that means, the process is experiencing more page-fault than it is supposed to; what we should do is? We should increase the number of frames allocated to this process so that it can keep more active pages in it and this is how it controls page-fault frequency; with this we come to the end of this lecture.