In multiprogramming CPU Utilization dependence on the number of Process in MM

2 CPU UTILIZATION ESTIMATE MM SIZE and IO USE EFFECTS.

3 CPU UTILIZATION DEPENDENCE

4 CPU UTILIZATION USE OF 1-p^n / n TO COMPUTE TIME GIVEN TO SETS OF PROCESSES (SIMPLE EXAMPLE)

5 CPU UTILIZATION USE OF 1-p^n / n TO COMPUTE TIME GIVEN TO SETS OF PROCESSES EXAMPLE (BOOK)

Overview: Process-Hole MM assignment and Paging MM Assignment

6 PROCESS ASSIGNMENT OF MEMORY-OVERVIEW

So that only Process X has access to Process X’s MM even though Process X may move

7 RELOCATION AND PROTECTION HARDWARE

Effect of process removal on generation of Holes. Schemes for finding a Hole to put a Process in.

8 PROCESSES REMOVAL RESULTANT HOLES

9 PROCESSES AND HOLES INSERTION SCHEMES: FIRST, NEXT AND BEST FIT

Keeping track of locations of Processes and Holes

10 PROCESSES AND HOLES BIT MAP DATA STRUCTURE *BLOCK SIZE Analysis*

11 PROCESSES AND HOLES LINK LIST DATA STRUCTURE

12 PROCESSES AND HOLES BINARY TREE DATA STRUCTURE BUDDY ALGORITHM

More details on Buddy Algorithm

13 BUDDY ALGORITHM MARKING TREE NODES TRACES

14 PROCESSES AND HOLES Buddy Algorithm Example

There are 1/2 as many holes as Processes (on average). Why? Consequence on assigning algorithms

15 PROCESSES AND HOLES-THE 1/2 RULE

16 PROCESSES AND HOLES UTILIZATION ANALYSIS USING THE 1/2 RULE

CONTENTS

© M C Paull 2005
**Defining CPU UTILIZATION (Efficiency)**

\[
\text{Utilization Fraction} = \frac{\sum_{i,x} t^x_i}{T} = \left( \text{As} + \text{Bs} + \text{Cs} \right)
\]

\[n = \text{Number of Processes (degree of multiprogramming) that are in MM. The larger the Memory the larger } n \text{ can be. For User Processes In MM:}
\]

\[p = \text{(average) fraction of the time a User Processes Waits for I/O}^{*} \text{ (running in Multi-Programming mode) [= Probability it is waiting for I/O]}
\]

\[p^n = \text{(average) fraction of the time all } n \text{ User Processes are Waiting for I/O} = \text{Probability all } n \text{ are Waiting ]}
\]

\[1 - p^n = \text{(average) fraction of the time a single User Process is in CPU. [=Probability all are Waiting]}
\]

*Time Waiting for I/O to Completed \(>= \) Time in I/O. Time Waiting for I/O to Completed = Time in I/O iff all I/O requests can be handled in parallel*

**Estimating CPU UTILIZATION**

Utilization is affected by many scheduling factors, such as the size of Quanta, the change in Priority associated with I/O. In addition it is affected by the size of memory and the mix of Processes-and the percentage the time each requires IO.

**CPU UTILIZATION ESTIMATE MM SIZE and IO USE EFFECTS.**

© M C Paull 2007
<table>
<thead>
<tr>
<th>#Processes</th>
<th>States</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU idle-</td>
<td>All in I/O</td>
<td>p</td>
<td>p²</td>
<td>p³</td>
<td>p⁴</td>
</tr>
<tr>
<td>CPU busy</td>
<td>Not All in I/O</td>
<td>1-p</td>
<td>1-p²</td>
<td>1-p³</td>
<td>1-p⁴</td>
</tr>
<tr>
<td>CPU per Process</td>
<td></td>
<td>1-p/1</td>
<td>1-p²/2</td>
<td>1-p³/3</td>
<td>1-p⁴/4</td>
</tr>
</tbody>
</table>

If $p = 3/4$, $n = 2$ prob that Process in I/O

$p^n = 9/16$ prob that CPU idle-All in I/O

$1-p^n = 7/16$, prob that CPU busy-not all in I/O

$\frac{1-p^n}{2} = 7/32$ fraction of CPU given to each Process

Table of Ave Fraction of CPU Time Available/Process as a Function of Ave Fraction of I/O Wait ($p$) and of Number of Processes in MM ($n$)

Note: Assumes that if any are not in IO one is in CPU Of course those not blocked waiting for IO they may be blocked for other reasons. So the estimate of Utilization is likely to be lower than estimated this way.

---

CPU UTILIZATION DEPENDENCE

© M C Paull 2005
BASIC

\[
\frac{\text{Fraction of CPU for each Proc}}{\text{CPU for eachProc}} = \frac{1-p^n}{n}
\]

\[
T_{\text{In CPU for eachProc}} = \frac{T_{\text{Run}}}{1-p^n/n} - T_{\text{Run}} = \frac{T_{\text{In CPU/for eachProc}}}{n/1-p^n}
\]

GIVEN:

<table>
<thead>
<tr>
<th>Jobs</th>
<th>Time of Arrival</th>
<th>CPU is Needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>J2</td>
<td>10</td>
<td>6</td>
</tr>
</tbody>
</table>

and Probability in I/O = \( p = .5 \)

PROBLEM: HOW LONG BEFORE J1 GETS ALL ITS CPU TIME?

Only First Job Present

J1 runs alone for 10 time units

1 JOB, J1, RUNNING

\[
\text{Run Time: } T_R = 10
\]

\[
T_C = 5, \quad T_I/O = 5
\]

In run time \( T_R = 10 \) with a only J1 running

J1 gets

\[
T_C = T_R \times \frac{(1-p)}{1} = 10 \times .5 = 5 \text{ in CPU (and } T_I/O = 5 \text{ in I/O)}
\]

Second Job Arrives

J2 joins J1 for 10 time units.

2 JOBS, J1 AND J2, RUNNING

\[
\text{Run Time: } T_R = 10
\]

\[
T_C = 3.75, \quad T_I/O = 5
\]

In run time \( T_R = 10 \) with J1 and J2 both running

J1 and J2 each get

\[
T_C = T_R \times \frac{(1-p^2/2)}{2} = 10 \times .375 = 3.75 \text{ in CPU (and } T_I/O = 1.25 \text{ in I/O)}
\]

CONTINUE

J1 needs a total of 10 time units in CPU to complete CPU use. How long till J1 completes?

Now J1 has 5 + 3.75 = 8.75 needs 1.25 = to complete its CPU time, (TotC = 10)

Need: 1.25 CPU. Run Time = ?

\[
T_C = T_R \times F_{C/P} (= (1-p^2/2))
\]

\[
\frac{T_C}{F_{C/P}} = T_R = 3.33
\]

\[
1.25/3.33 = T_R = 3.33
\]

CONTINUE

CPU UTILIZATION USE OF 1-\( p^n \)/n TO COMPUTE TIME GIVEN TO SETS OF PROCESSES(SIMPLE EXAMPLE)

© M C Paull 2005
CPU UTILIZATION USE OF $1-p^n/n$ TO COMPUTE TIME GIVEN TO SETS OF PROCESSES
EXAMPLE (BOOK)

**If 3 jobs are present. How long ($T$) must the jobs be present so that each gets a CPU time of 1.25?**

The fraction of the time devoted to the CPU with 3 jobs present is $1 - p = .875$.

This is shared equally amongst 3 jobs so each gets .291 of the time.

Therefore $.291T = 1.25$. So $T = 4.3$. After this there is a job requiring time 1 to completion.

Since there are 2 jobs present we have $.375T = 1$, so $T = 2.67$. etc.
**SINGLE PROGRAM**

- **RAM (RWM)**
- **ROM or RAM (top or bottom)**

**PARTITIONS**

- **First Fit**
- **By Size**

- **J, J, J, J**

**Best Fit**

- **1 Entire Progs**
- **2 Protection**
- **3 1 Process/Partition (At Most)**
- **4 Possible Relocation**

**Pooling Resources**

- Making Resource Available
- To More than 1 Claimant
- Here for growth in Data or Stack

**PROCESSES & HOLES**

- **Relocation & Protection**

- **J6**
  - **J1**
  - **J2**
  - **J3**
  - **J4**
  - **J5**

- **J2**

- **J7**
  - **J3**
  - **J4**
  - **J5**

**Page Table**

- **MM**

**PAGE OFFSET**

- **address**

**VIRTUAL MEMORY**

- **VM**

**Page Table**

- **MM**

**PAGE OFFSET**

- **address**

**PAGING**

- **Process Assignment of Memory Overview**

© M C Paull 2005
RELOCATION AND PROTECTION HARDWARE
PROCESSES REMOVAL RESULTANT HOLES

© M C Paull 2005
FiTTING PROCESSES IN HOLES: FIRST, NEXT AND BEST FIT, SCHEMES An Example
© M C Paull 2005
\[ B = \frac{mp}{b} \]

\[ \Rightarrow \]

\[ \text{bits in MM Procs} / (\text{bits / block}) = \text{blocks in MM} = \text{bits in Bitmap} \]

\[ \ldots \]

\[ \overline{000} \overline{11} \overline{00} \overline{11} \overline{111} \overline{0111} \ldots \overline{0111} \overline{100} \]

\[ \text{Overhead} = U + B = \frac{Np}{2} + \frac{mp}{b} \]

\[ \minimizing: \quad d(U+B)/db = \frac{Np}{2} - \frac{mp}{b^2} \]

\[ b_{\text{optimal}} = \sqrt{\frac{2mp}{Np}} \]

\[ \ldots \]

\[ \overline{000} \overline{11} \overline{00} \overline{11} \overline{111} \overline{0111} \ldots \overline{0111} \overline{100} \]

\[ \overline{0} \overline{1} \overline{11} \overline{1} \overline{0} \overline{1} \overline{1} \overline{1} \ldots \overline{0111} \overline{100} \]

\[ \ldots \]

\[ \overline{000} \overline{11} \overline{00} \overline{11} \overline{111} \overline{0111} \ldots \overline{0111} \overline{100} \]

\[ \ldots \]

\[ \overline{000} \overline{11} \overline{00} \overline{11} \overline{111} \overline{0111} \ldots \overline{0111} \overline{100} \]

\[ \ldots \]

\[ \overline{000} \overline{11} \overline{00} \overline{11} \overline{111} \overline{0111} \ldots \overline{0111} \overline{100} \]

\[ \ldots \]

\[ \overline{000} \overline{11} \overline{00} \overline{11} \overline{111} \overline{0111} \ldots \overline{0111} \overline{100} \]
1. Reconstructing HOLES when PROCESS is Removed
2. Finding a HOLE to Accommodate a PROCESS
3. OVERHEAD

Can Speed Up Search For Hole to Accommodate a Process by having an additional linked list connecting Holes linked list of Holes 2 More Pointers/Cell

Processes and Holes Link List Data Structure

© M C Paull 2005
The Buddy Algorithm for placing a PROCESS in a HOLE is similar to the following procedure:

To enter a PROCESS, P, of size \( N \), round \( N \) up to the next highest power of 2 giving \( P' \). Suppose that the size of \( P' \) is \( 2^n \). Now examine the memory. Check for a HOLE of size \( 2^n \) starting at location 0, if not, check for such a HOLE starting at location \( 2^n \), if not check location \( 2^{n+1} \), etc. Use Leftmost Depth First Search.

In general check for a HOLE of size \( 2^n \) starting at location 0 then \( 2^{n+k} \) with \( k = 0, 1, 2, \ldots \), until such a hole is first found. P is placed there. The Buddy algorithm is based on a tree data-structure whose use decreases the amount of search for a hole once the level at which a hole is sought has been determined. The algorithm avoids many of the already occupied locations at that level, by marking tree nodes of occupied locations and others to help direct the search to an available hole. We first show a simple version of such an algorithm.

**DATA STRUCTURE FOR KEEPING TRACK OF AVAILABLE BLOCKS OF SIZE \( 2^n \)**

The diagram illustrates how the memory is divided into intervals and how the algorithm searches for an available hole. Each level of the tree represents a different size of hole, with the tree nodes marking the availability of each block within that size.
Example:
Here we give a simple algorithm based on the tree data structure for keeping track of Holes and Processes. The algorithm maintains information (MARKS) at the nodes of the tree to shorten the search for a hole which will accommodate a given Process. In an MM contains $2^5$ bytes in this example. So if the size of process $P_s=12$. The next highest power of 2 is $16=2^4$. Go, by leftmost depth first search to level $2^4$. Does the leftmost node at that level indicate there is space starting at location 0? If so place $P$ there and mark the tree node accordingly. If it is not there look at the node corresponding to location $16=2^4$, In general start at $0$ then $2^{4+k}, k=0,1,2,...$, until such a hole is first found. $P$ is placed there. If no node at level $2^4$ is found $P$ cannot be placed in MM. The information that is maintained at each node is given by the following node “Markings”

Notice; not only can considerable space be unuseable because Processes do not generally have sizes which are a power of two, but the placement choice for a Processes which can fit in a number of holes can affect the loss of space in the MM. Processes of size $2^k$ equal in number to the number of size $2^{k+1}$ partitions in the MM could be located 1 per $2^{k+1}$ partition. Then there would be no room for a Process of size $2^{k+1}$. Whereas if these Processes were located with 2 in each of the leading $2^{k+1}$ partitions there would be plenty of room for Processes of size $2^{k+1}$. So generally it is advantageous to fill a partially filled partition when 1/2 of it is available rather start an empty partition. So one ought to place a Process at the lowest address at which it can be accommodated. Leftmost depth-first search would achieve this. Backup for this search could be minimized if if the algorithm chosen gives the highest level available by both the left child, and the right child. Such an algorithm is developed now.
Because it is generally best to use siblings of already occupied node it is probably best to pursue the paths whose highest available level is lowest, but still \( \geq \) level sought.

**BUDDY ALGORITHM MARKING TREE NODES TRACES**

© 2005 M C Paull
After A Time the Number of Holes in Memory is 1/2 the Number of Processes

There are $n$ processes. There are $n+1$ positions between successive processes each of which could be occupied by a hole or nil (nothing). The average layout of MM can then be represented as:

<table>
<thead>
<tr>
<th>(*) Hole or $\lambda$ #</th>
<th>1 2 3 ... $n$  $n+1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hole or $\lambda$</td>
<td>* 1 * 1 * ... * 1 *</td>
</tr>
<tr>
<td>(1) Process #</td>
<td>1 2 ... $n$</td>
</tr>
</tbody>
</table>

Each * is either a hole (0) or unoccupied ($\lambda$). Considering all possible combinations of assignments to the *'s. In the average assignment 1/2 of the *'s are 0. So there are $n$ 1's (processes) in each and an average of $(n+1)/2$ 0's (holes) per combination. Therefore:

Average number of processes = $P(n) = n$
Average number of holes = $H(n) = (n+1)/2$
So the ratio of Holes to Processes is $H(n) / P(n) = (n+1) / 2n = (1 + 1/n) / 2$

= 1/2 quickly as $n$ increases

Example $n = 2$

Dependence Of Utilization On The Ratio of Process to Hole Size

Obviously, if the ratio of hole size to process size decreases, and the number of holes is the same as the number of processes, the fraction of the memory occupied by holes decreases, that is Utilization improves (less of the memory is devoted to Holes). In fact if the number of holes bares any fixed relation to the number of processes decreasing the ratio of hole to process size will increase the Utilization. In fact once the size relation is fixed Utilization dependence on ratio of sizes is completely determined. Since a fixed relation between the number of Holes and Processes (Number of Holes = 1/2 Processes) is known, how the Utilization will increase with the decrease in the ratio of Hole to Process size is determined.

Process size is uneffected by whether we use the Best First or First Fit algorithms, but Hole size tends to be smaller with Best Fit than First Fit. (This assumes that the 1/2 rule holds equally well independent of which Best or First Fit is used) The details are on the next page.)
Best First always places a new entering Process so as to leave the minimum size hole (Note an exact fit is always Optimal.) This gives good Utilization, but tends to leave (many) small Holes which become unuseable. Neither First or Best Fit is uniformly optimal.

\[ U = \frac{P}{(P + H)} \]

\[ U = \frac{2}{(2 + R)} \]

\[ R = \frac{s_h}{s_p} \]

\[ f_P = \frac{P}{(P + H)} \]

\[ f_P = \frac{2}{2 + 1} = .66 \]

\[ s_h \text{ is the average size of a Hole in pages} \]

\[ s_p \text{ is the average size of a Process in pages} \]

\[ n \text{ is the number of Processes in MM} \]

\[ U \text{ fraction of the MM in Processes} = \text{Utilization} \]

\[ U = \frac{s_p n}{(s_p n + s_h (n/2))} \]

\[ = 1 / (1 + s_h / 2s_p) \]

\[ = 1 / (1 + R/2) \text{ - where } R = s_h / s_p \text{ Ratio of average Hole to Process size} \]

\[ U = \frac{2}{2 + R} \]

\[ = .66 \]

Using the 1/2 rule some idea about utilization’s dependence on the ratio average hole to average process size

**PROCESSES AND HOLES UTILIZATION ANALYSIS USING THE 1/2 RULE**

© M C Paull 2005