**BÀI TẬP MEMORY MANAGEMENT**

**1. Quản lý bộ nhớ**

**a. Tại sao có sự phân biệt giữa paging và demand paging** ?

* Paging (phân trang):
  + Là chia bộ nhớ thành các khối nhỏ cố định (trang) để quản lý.
  + Các chương trình được chia thành các trang có kích thước tương ứng.
  + Toàn bộ chương trình thường được tải vào bộ nhớ trước khi thực thi.
* Demand Paging (phân trang theo nhu cầu):
  + Không tải toàn bộ chương trình vào bộ nhớ ngay từ đầu.
  + Chỉ tải những trang được truy cập vào bộ nhớ, giúp tiết kiệm bộ nhớ và tăng hiệu suất.

Khác nhau : Paging tải toàn bộ chương trình vào bộ nhớ, trong khi demand paging chỉ tải khi cần.

**b. Điều gì tốt cho demand paging**

Demand paging hoạt động hiệu quả khi :

* Chương trình có tính cục bộ cao : Nếu chương trình thường xuyên truy cập cùng một nhóm trang trong thời gian dài, hệ thống chỉ cần giữ một số trang đó trong bộ nhớ.
* Hệ thống có bộ nhớ ảo : Demand paging dùng bộ nhớ ảo để mở rộng không gian địa chỉ, giúp chạy được các chương trình lớn hơn bộ nhớ thực.
* Cơ chế thay thế trang hiệu quả: Giúp giảm số lần lỗi trang và cải thiện hiệu suất.

**c. Tên 1 bộ cấp phát trang**

Một trong những bộ cấp phát trang phổ biến là Buddy System Allocator

**d. Khi nào lỗi trang xảy ra**

* Tiến trình yêu cầu truy cập vào một trang nhưng trang đó không có trong bộ nhớ RAM.
* Hệ điều hành phải tải trang từ bộ nhớ ảo vào bộ nhớ chính (RAM).

**e. Điều gì xảy ra khi lỗi trang xảy ra**

Hệ thống sẽ thực hiện:

1. Kiểm tra bảng trang : Xác định trang yêu cầu có trong bộ nhớ hay không.
2. Nếu trang chưa có trong RAM:
   * Tìm một khung trang trống trong RAM.
   * Nếu không có khung trống, hệ thống phải thay thế trang.
3. Nạp trang từ ổ đĩa vào RAM.
4. Cập nhật bảng trang để ghi nhận vị trí mới của trang.
5. Tiếp tục thực thi tiến trình.

**f. Thuật giải thay thế trang làm gì**

Thuật toán thay thế trang chọn trang nào trong RAM để thay thế khi cần nạp trang mới.

Mục tiêu : Giảm số lần lỗi trang để tăng hiệu suất hệ thống.

**2. Answer following questions:**

**a. Define external and internal fragments. Consider the following memory management schemes: fixed-size partitions, variable-size partitions, and paging. Which schemes have external fragments, and which schemes have internal fragments? Why?**

Định nghĩa mãnh bộ nhớ ( fragmentation ) :

Phân mảnh bên ngoài : Xảy ra khi có nhiều khoảng trống nhỏ phân tán trong bộ nhớ nhưng không có khoảng nào đủ lớn để chứa tiến trình cần cấp phát.

Phân mảnh bên trong : Xảy ra khi một khối bộ nhớ được cấp phát lớn hơn nhu cầu của tiến trình, phần dư thừa không thể sử dụng.

|  |  |  |  |
| --- | --- | --- | --- |
| Phương pháp quản lý bộ nhớ | Phân mảnh bên ngoài | Phân mảnh bên trong | Giải thích |
| Phân vùng cố định | Không | có | Kích thước vùng cố định có thể lớn hơn nhu cầu tiến trình, gây lãng phí. |
| Phân vùng động | có | Không | Vùng nhớ có kích thước tùy chỉnh, nhưng có thể tạo ra nhiều khoảng trống nhỏ. |
| Phân trang | không | có | Bộ nhớ được chia thành các trang cố định, một số trang không sử dụng hết. |

**b. Given memory holes (i.e., unused memory blocks) of 100K, 500K, 200K, 300K and 600K (in address order) as shown below, how would each of the first-fit, next-fit, best fit and worst-fit algorithms allocate memory requests for 290K, 420K, 110K and 350K (in this order)? The shaded areas are used/allocated regions and are not available. Write your answer into the following diagrams. You should clearly write down the size of each memory block and indicate its status**

Bộ nhớ trống đang có : 100K, 500K, 200K, 300K, 600K

Yêu cầu cấp phát : 290K, 420K, 110K, 350K

1. First-Fit (Tìm khoảng trống đầu tiên đủ lớn)

|  |  |  |
| --- | --- | --- |
| Yêu cầu cấp phát | Bộ nhớ trống cấp phát và còn lại | Bộ nhớ sau cấp phát |
| 290k | 500k 🡪 còn lại 210k | 100K, (290K), 210K, 200K, 300K, 600K |
| 420k | 600k 🡪còn lại 180k | 100K, (290K), 210K, 200K, 300K, (420K), 180K |
| 110k | 210K ( còn dư của 500k ) 🡪 còn 100K | 100K, (290K), (110K), 100K, 200K, 300K, (420K), 180K |
| 350k | Không còn bộ nhớ nào đủ lớn để cấp phát | Không thể cấp phát |

Vậy cấp phát được 3 tiến trình : 290k, 420k, 110k, 350k

2. Next-Fit (Tiếp tục tìm từ vị trí cuối cùng)

|  |  |  |
| --- | --- | --- |
| Yêu cầu cấp phát | Bộ nhớ trống cấp phát và còn lại | Bộ nhớ sau cấp phát |
| 290k | 500k 🡪 còn lại 210k | 100K, (290K), 210K, 200K, 300K, 600K |
| 420k | 600k 🡪còn lại 180k | 100K, (290K), 210K, 200K, 300K, (420K), 180K |
| 110k | 210K ( còn dư của 500k ) 🡪 còn 100K | 100K, (290K), (110K), 100K, 200K, 300K, (420K), 180K |
| 350k | Không còn bộ nhớ nào đủ lớn để cấp phát | Không thể cấp phát |

Vậy cấp phát được 3 tiến trình : 290k, 420k, 110k, 350k

3. Best-Fit (Chọn khối nhỏ nhất có thể chứa tiến trình)

|  |  |  |
| --- | --- | --- |
| Yêu cầu cấp phát | Bộ nhớ trống cấp phát và còn lại | Bộ nhớ sau cấp phát |
| 290k | 300k 🡪 còn lại 10k | 100K, 500K, 200K, (290K), 10K, 600K |
| 420k | 500k 🡪còn lại 80k | 100K, (420K), 80K, 200K, (290K), 10K, 600K |
| 110k | 200K 🡪 còn 90K | 100K, (420K), 80K, (110K), 90K, (290K), 10K, 600K |
| 350k | 600K → còn 250K | 100K, (420K), 80K, (110K), 90K, (290K), 10K, (350K), 250K |

Vậy cấp phát thành công tất cả tiến trình.

4. Worst-Fit (Chọn khối lớn nhất có thể chứa tiến trình)

|  |  |  |
| --- | --- | --- |
| Yêu cầu cấp phát | Bộ nhớ trống cấp phát và còn lại | Bộ nhớ sau cấp phát |
| 290k | 600k 🡪 còn lại 310k | 100K, 500K, 200K, 300K, (290K), 310K |
| 420k | 500k 🡪còn lại 80k | 100K, (420K), 80K, 200K, 300K, (290K), 310K |
| 110k | 310K → còn 200K | 100K, (420K), 80K, 200K, 300K, (290K), (110K), 200K |
| 350k | Không còn bộ nhớ nào đủ lớn | Không thể cấp phát |

Vậy chỉ cấp phát được 3 tiến trình : 290k, 420k, 110k

**3. Giả sử trong quá trình quản lý bộ nhớ ảo dạng phân đoạn, HĐH duy trì Segment Table:**

|  |  |  |
| --- | --- | --- |
| Segment | Base | Limit |
| 0 | 300 | 700 |
| 1 | 1200 | 500 |
| 2 | 2000 | 600 |

**Hãy tính địa chỉ vật lý cho mỗi địa chỉ logic sau: (1,200), (1,0), (0,700), (2,0), (2,600)**

**Công thức tính địa chỉ vật lý :**

**Địa chỉ vật lý = Base + d**

**Nếu d < limit của phân đoạn 🡪 địa chỉ hợp lệ.**

**Nếu d >= limit của phân đoạn 🡪 lỗi segmentation fault.**

**Địa chỉ logic : ( s, d )**

s : số hiệu phân đoạn segment

d : độ dịch ( so sánh với limit )

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Địa chỉ logic | Base | Limit | Hợp lệ | Địa chỉ vật lý |
| ( 1, 200 ) | 1200 | 700 | Hợp lệ ( 1 có trong segment, 200 < 500 ) | Base + d = 1200 + 200 = 1400 |
| ( 1, 0 ) | 1200 | 700 | Hợp lệ ( 1 có trong segment, 0 < 500 ) | Base + d = 1200 + 0 = 1200 |
| ( 0, 700 ) | 300 | 700 | Không hợp lệ ( 0 có trong segment, 700 = = 700 ) | Lỗi |
| (2, 0 ) | 2000 | 600 | Hợp lệ ( 2 có trong segment, 0 < 600 ) | Base + d = 2000 + 0 = 2000 |
| (2, 600 ) | 2000 | 600 | Không hợp lệ ( 2 có trong segment, 600 = = 600 ) | Lỗi |

**4. Giả sử có bảng đoạn sau :**

|  |  |  |
| --- | --- | --- |
| Segment | Base | Limit |
| 0 | 219 | 600 |
| 1 | 2300 | 14 |
| 2 | 90 | 100 |
| 3 | 1327 | 580 |
| 4 | 1952 | 96 |

**Tính địa chỉ vật lý cho mỗi địa chỉ logic sau:**

**a. (0, 430) b. (1,010) c. (2,500) d. (3,400) e. (4,112)**

**Công thức tính địa chỉ vật lý :**

**Địa chỉ vật lý = Base + d**

**Nếu d < limit của phân đoạn 🡪 địa chỉ hợp lệ.**

**Nếu d >= limit của phân đoạn 🡪 lỗi segmentation fault.**

**Địa chỉ logic : ( s, d )**

s : số hiệu phân đoạn segment

d : độ dịch ( so sánh với limit ) ( offset )

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Địa chỉ logic | Base | Limit | Hợp lệ | Địa chỉ vật lý |
| ( 0, 430 ) | 219 | 600 | Hợp lệ ( 0 có trong segment, 430 < 600 ) | Base + d = 219 + 430 = 649 |
| ( 1, 010 ) | 2300 | 14 | Hợp lệ ( 1 có trong segment, 10 < 14 ) | Base + d = 2300 + 10 = 2310 |
| ( 2, 500 ) | 90 | 100 | Không hợp lệ ( 2 có trong segment, 500 > 100 ) | Lỗi |
| (3 , 400 ) | 1327 | 580 | Hợp lệ ( 3 có trong segment, 400 < 580 ) | Base + d = 1327 + 400 = 1727 |
| (4, 112 ) | 1952 | 96 | Không hợp lệ ( 4 có trong segment, 112 > 96) | Lỗi |

**5. Vùng địa chỉ logic (vùng nhớ ảo) của 1 tiến trình được chia thành các trang A, B, C, D, E được ánh xạ sang vùng địa chỉ vật lý theo bảng sau:**

|  |  |  |
| --- | --- | --- |
| Stt | Trang logic | Khung trang số |
| 1 | A | 3 |
| 2 | B |  |
| 3 | C | 5 |
| 4 | D | 1 |
| 5 | E |  |

**Hãy minh họa bằng hình vẽ các thành phần sau:**

* **Vùng địa chỉ logic**
* **Bảng trang (sử dụng bit “true – false”)**
* **Vùng địa chỉ vật lý**
* **Nội dung đĩa cứng lưu các trang**

Vùng địa chỉ logic :

|  |  |
| --- | --- |
| Page A | Đã tải vào Ram – frame 3 |
| Page B | Chưa có trong Ram 🡪 lưu trên đĩa cứng |
| Page C | Đã tải vào Ram – frame 5 |
| Page D | Đã tải vào Ram – frame 1 |
| Page E | Chưa có trong Ram 🡪 lưu trên đĩa cứng |

Bảng trang ( sử dụng bit “true – false”)

|  |  |  |
| --- | --- | --- |
| Trang | Frame | Bit |
| A | 3 | true |
| B |  | false |
| C | 5 | true |
| D | 1 | true |
| E |  | false |

Vùng địa chỉ vật lý

|  |  |
| --- | --- |
| Frame 0 | Trống |
| Frame 1 | Page D |
| Frame 2 | Trống |
| Frame 3 | Page A |
| Frame 4 | Trống |
| Frame 5 | Page C |
| Frame 6 | Trống |
| Frame 7 | Trống |

Nội dung đĩa cứng lưu các trang :

|  |  |
| --- | --- |
| Trang | Tình trạng |
| B | Chưa tải vào Ram |
| E | Chưa tải vào Ram |

1. **Dung lượng vùng nhớ là 3000K. Địa chỉ bắt đầu cấp phát là 0K. Cấp phát bộ nhớ bằng kỹ thuật phân vùng động với giải thuật Worst-Fit. Xây dựng danh sách quản lý cấp phát tại từng thời điểm tương ứng với chuỗi cấp phát sau :**

**A(300K)→ B(400K)→ C(550K)→ thu hồi B→ D(400K)→ thu hồi A→ E(300K)→ F(350K)→ thu hồi D→ G(50K)**

Thuật toán cấp phát: Worst-Fit (chọn vùng trống lớn nhất để cấp phát)

|  |  |  |  |
| --- | --- | --- | --- |
| Bước | Mô tả | Vùng trống lớn nhất | Kết quả |
| 1 |  | 3000k |  |
| 2 | Cấp phát A ( 300k ) | 3000k 🡪 còn 2700k | A(300K) | 2700K Free |
| 3 | Cấp phát B ( 400k ) | 2700k 🡪 còn 2300k | A(300K) | B(400K) | 2300K Free |
| 4 | Cấp phát C ( 550k ) | 2300k 🡪 còn 1750k | A(300K) | B(400K) | C(550K) | 1750K Free |
| 5 | Thu hồi B : xoá khỏi bộ nhớ | 1750k 🡪 thêm 400k free | A(300K) | 400K Free | C(550K) | 1750K Free |
| 6 | Cấp phát |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |