Memory Management

1) Virtual Address Space

2)Address translating

3)Protection attribute

4) Alignment

5)Stack and Heap Memory Allocation

1)Virtual Address Space

Không gian địa chỉ ảo cho một tiến trình là tập hợp các địa chỉ bộ nhớ ảo mà nó có thể sử dụng. Không gian địa chỉ cho mỗi quá trình là riêng tư và không thể được truy cập bởi các quá trình khác trừ khi nó được chia sẻ.

Địa chỉ ảo không đại diện cho vị trí vật lý thực của một đối tượng trong bộ nhớ; thay vào đó, hệ thống duy trì một bảng trang cho mỗi quá trình, đây là một cấu trúc dữ liệu nội bộ được sử dụng để dịch các địa chỉ ảo thành địa chỉ vật lý tương ứng của chúng. Mỗi khi một luồng tham chiếu đến một địa chỉ, hệ thống sẽ dịch địa chỉ ảo thành địa chỉ thực.

Không gian địa chỉ ảo cho Windows 32-bit có kích thước 4 gigabyte (GB) và được chia thành hai phân vùng: một phân vùng để sử dụng cho tiến trình và phân vùng còn lại để hệ thống sử dụng

Dung lượng bộ nhớ vật lý tối đa mà Microsoft Windows hỗ trợ nằm trong khoảng từ 2 GB đến 24 TB, tùy thuộc vào phiên bản Windows. Không gian địa chỉ ảo của mỗi tiến trình có thể nhỏ hơn hoặc lớn hơn tổng bộ nhớ vật lý hiện có trên máy tính. Tập hợp con của không gian địa chỉ ảo của một tiến trình nằm trong bộ nhớ vật lý được gọi là tập làm việc. Nếu các luồng của một quy trình cố gắng sử dụng nhiều bộ nhớ vật lý hơn hiện có, hệ thống sẽ trang một số nội dung bộ nhớ vào đĩa. Tổng dung lượng địa chỉ ảo có sẵn cho một quá trình bị giới hạn bởi bộ nhớ vật lý và dung lượng trống trên đĩa có sẵn cho tệp hoán trang.

Bộ nhớ vật lý và không gian địa chỉ ảo của mỗi tiến trình được tổ chức thành các trang, đơn vị bộ nhớ, kích thước của chúng phụ thuộc vào máy tính chủ. Ví dụ: trên máy tính x86, kích thước trang chủ là 4 kilobyte.

Để tối đa hóa tính linh hoạt trong việc quản lý bộ nhớ, hệ thống có thể di chuyển các trang của bộ nhớ vật lý đến và từ một tệp hoán trang trên đĩa. Khi một trang được di chuyển trong bộ nhớ vật lý, hệ thống sẽ cập nhật bản đồ trang của các quá trình bị ảnh hưởng. Khi hệ thống cần dung lượng trong bộ nhớ vật lý, hệ thống sẽ di chuyển các trang ít được sử dụng gần đây nhất của bộ nhớ vật lý sang tệp hoán trang. Việc điều khiển bộ nhớ vật lý của hệ thống là hoàn toàn trong suốt đối với các ứng dụng, chỉ hoạt động trong không gian địa chỉ ảo của chúng.

**2)** **Address translating**

Dịch địa chỉ

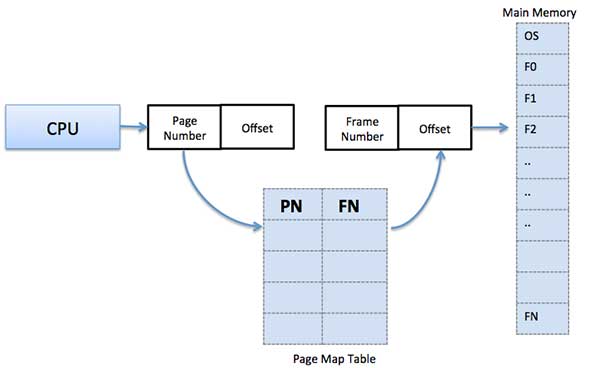
Địa chỉ trang được gọi là địa chỉ logic và được biểu diễn bằng số trang và phần bù.

Địa chỉ lôgic = Số trang + độ lệch trang

Địa chỉ khung được gọi là địa chỉ vật lý và được biểu diễn bằng số khung và độ lệch.

Địa chỉ vật lý = Số khung + độ lệch trang

Một cấu trúc dữ liệu được gọi là bảng sơ đồ trang được sử dụng để theo dõi mối quan hệ giữa một trang của quy trình với một khung trong bộ nhớ vật lý.



Khi hệ thống phân bổ khung cho bất kỳ trang nào, nó sẽ dịch địa chỉ logic này thành địa chỉ vật lý và tạo mục nhập vào bảng trang để sử dụng trong suốt quá trình thực thi chương trình.

Khi một quy trình được thực thi, các trang tương ứng của nó sẽ được tải vào bất kỳ khung bộ nhớ khả dụng nào. Giả sử bạn có một chương trình 8Kb nhưng bộ nhớ của bạn chỉ có thể chứa 5Kb tại một thời điểm nhất định, khi đó khái niệm phân trang sẽ xuất hiện. Khi máy tính hết RAM, hệ điều hành (OS) sẽ di chuyển các trang không mong muốn hoặc không mong muốn của bộ nhớ sang bộ nhớ phụ để giải phóng RAM cho các tiến trình khác và đưa chúng trở lại khi chương trình cần.

Quá trình này tiếp tục diễn ra trong toàn bộ quá trình thực thi chương trình trong đó HĐH tiếp tục xóa các trang không hoạt động khỏi bộ nhớ chính và ghi chúng vào bộ nhớ phụ và đưa chúng trở lại khi chương trình yêu cầu.

**3)Protection attribute**

|  |  |
| --- | --- |
| Protection Attribute | Description |
| PAGE\_NOACCESS | Cố gắng đọc từ, ghi vào hoặc thực thi mã trong trang này để tăng quyền truy cập |
| PAGE\_READONLY | Cố gắng ghi hoặc thực thi mã trong trang này nâng cao quyền truy cập |
| PAGE\_READWRITE | Cố gắng thực thi mã trong trang này dẫn đến vi phạm quyền truy cập. |
| PAGE\_EXECUTE | Cố gắng đọc hoặc ghi bộ nhớ trong trang này làm tăng vi phạm quyền truy cập. |
| PAGE\_EXECUTE\_READ | Cố gắng ghi vào bộ nhớ trong trang này làm tăng vi phạm quyền truy cập. |
| PAGE\_EXECUTE\_READWRITE | Bạn không thể làm gì với trang này để tăng vi phạm quyền truy cập. |
| PAGE\_WRITECOPY | Nỗ lực thực thi mã trong trang này dẫn đến vi phạm quyền truy cập. Cố gắng viết thư cho  bộ nhớ trong trang này khiến hệ thống phân trang tệp). |
| PAGE\_EXECUTE\_WRITECOPY | ạn không thể làm gì với khu vực này để gia tăng vi phạm quyền truy cập. Nỗ lực để  ghi vào bộ nhớ trong trang này khiến hệ thống đặt quá trình riêng tư của nó  bản sao của trang (được hỗ trợ bởi tệp hoán trang) |

Một số ứng dụng phần mềm độc hại ghi mã vào các vùng bộ nhớ dành cho dữ liệu (chẳng hạn như ngăn xếp của chuỗi) và sau đó

ứng dụng thực thi mã độc. Tính năng Ngăn chặn Thực thi Dữ liệu (DEP) của Windows cung cấp khả năng bảo vệ chống lạikiểu tấn công phần mềm độc hại này. Với DEP được bật, hệ điều hành chỉ sử dụng các biện pháp bảo vệ PAGE\_EXECUTE\_ \* trêncác vùng bộ nhớ được dùng để thực thi mã; các biện pháp bảo vệ khác (thườnglàPAGE\_READWRITE) được sử dụng cho

các vùng bộ nhớ dự định có dữ liệu trong đó (chẳng hạn như ngăn xếp luồng và đống ứng dụng).

Khi CPU cố gắng thực thi mã trong một trang không có bảo vệ PAGE\_EXECUTE\_ \*, CPU sẽ ném

một ngoại lệ vi phạm quyền truy cập.hậm chí cònđược bảo vệ nhiều hơn. Khi bạn xây dựng mã của mình bằng công tắc trình liên kết / SAFESEH, các trình xử lý ngoại lệ được đăng ký trong một

bảng đặc biệt trong tệp hình ảnh. Bằng cách đó, khi một trình xử lý ngoại lệ sắp thực thi, hệ điều hành sẽ kiểm tra

trình xử lý này có được đăng ký trong bảng hay không trước khi cho phép nó chạy.

Quyền truy cập Copy-on-Write

Tất cả các thuộc tính bảo vệ được liệt kê trong bảng trước phải khá dễ hiểu ngoại trừ hai thuộc tính cuối cùng:

PAGE\_WRITECOPY và PAGE\_EXECUTE\_WRITECOPY. Các thuộc tính này tồn tại để tiết kiệm việc sử dụng RAM và tệp hoán trang

không gian. Windows hỗ trợ một cơ chế cho phép hai hoặc nhiều quy trình chia sẻ một khối lưu trữ duy nhất. Vì vậy, nếu 10

các phiên bản Notepad đang chạy, tất cả các phiên bản đều chia sẻ mã của ứng dụng và các trang dữ liệu. Chia sẻ tất cả các phiên bản

các trang lưu trữ giống nhau cải thiện đáng kể hiệu suất của hệ thống — nhưng điều này đòi hỏi tất cả các trường hợp phải xem xét

bộ nhớ ở chế độ chỉ đọc hoặc chỉ thực thi. Nếu một chuỗi trong một trường hợp được ghi vào bộ nhớ sửa đổi nó, bộ lưu trữ dưới dạng

được nhìn thấy bởi các trường hợp khác cũng sẽ được sửa đổi, gây ra sự hỗn loạn hoàn toàn.

Để ngăn chặn sự hỗn loạn này, hệ điều hành chỉ định tính năng bảo vệ sao chép-ghi cho khối lưu trữ được chia sẻ. Khi nào

một mô-đun .exe hoặc .dll được ánh xạ vào một không gian địa chỉ, hệ thống sẽ tính toán có bao nhiêu trang có thể ghi được. (Thường xuyên,

các trang chứa mã được đánh dấu là PAGE\_EXECUTE\_READ trong khi các trang chứa dữ liệu được đánh dấu

PAGE\_READWRITE.) Sau đó, hệ thống phân bổ bộ nhớ từ tệp hoán trang để chứa các trang có thể ghi này. Đây

lưu trữ tệp hoán trang không được sử dụng trừ khi các trang có thể ghi của mô-đun thực sự được ghi vào.

Khi một luồng trong một quy trình cố gắng ghi vào một khối được chia sẻ, hệ thống sẽ can thiệp và thực hiện các bước sau:

1. Hệ thống tìm thấy một trang bộ nhớ trống trong RAM. Lưu ý rằng trang miễn phí này sẽ được hỗ trợ bởi một trong các trang

được phân bổ trong tệp hoán trang khi mô-đun được ánh xạ lần đầu tiên vào không gian địa chỉ của quy trình. Vì

hệ thống đã phân bổ tất cả không gian tệp hoán trang có thể được yêu cầu khi mô-đun được ánh xạ lần đầu tiên, bước này

không thể không thành công.

2. Hệ thống sao chép nội dung của trang đang cố gắng sửa đổi sang trang miễn phí được tìm thấy ở bước 1. Điều này

trang miễn phí sẽ được chỉ định bảo vệ PAGE\_READWRITE hoặc PAGE\_EXECUTE\_READWRITE. Bản gốc

bảo vệ và dữ liệu của trang không thay đổi gì cả.

3. Sau đó, hệ thống cập nhật các bảng trang của quy trình để địa chỉ ảo được truy cập hiện chuyển thànhtrang mới của RAM.

Sau khi hệ thống thực hiện các bước này, quá trình có thể truy cập phiên bản riêng tư của trang lưu trữ này. Trong

Ngoài ra, bạn không nên vượt qua PAGE\_WRITECOPY hoặc PAGE\_EXECUTE\_WRITECOPY khi đặt trước

không gian địa chỉ hoặc cam kết lưu trữ vật lý bằng cách sử dụng chức năng VirtualAlloc. Làm như vậy sẽ khiến cuộc gọi đến

VirtualAlloc không thành công; gọi GetLastError trả về ERROR\_ INVALID\_PARAMETER. Hai thuộc tính này được sử dụng bởi

hệ điều hành khi nó ánh xạ hình ảnh tệp .exe và DLL.

Cờ thuộc tính bảo vệ quyền truy cập đặc biệt

Ngoài các thuộc tính bảo vệ đã được thảo luận, có ba cờ thuộc tính bảo vệ: PAGE\_NOCACHE,

PAGE\_WRITECOMBINE và PAGE\_GUARD. Bạn sử dụng ba cờ này theo cách bitwise HOẶC đặt chúng bằng bất kỳ biện pháp bảo vệ nàocác thuộc tính ngoại trừ PAGE\_NOACCESS.

Cờ đầu tiên trong số các cờ thuộc tính bảo vệ này, PAGE\_NOCACHE, vô hiệu hóa bộ nhớ đệm của các trang đã cam kết. Cờ này khôngđược khuyến nghị để sử dụng chung — nó tồn tại chủ yếu dành cho các nhà phát triển trình điều khiển thiết bị phần cứng, những người cần thao tác với bộ nhớbộ đệm.

Cờ thuộc tính bảo vệ thứ hai, PAGE\_WRITECOMBINE, cũng dành cho các nhà phát triển trình điều khiển thiết bị. Nó cho phép ghi nhiều lần vàomột thiết bị duy nhất được kết hợp với nhau để cải thiện hiệu suất.

Cờ thuộc tính bảo vệ cuối cùng, PAGE\_GUARD, cho phép ứng dụng nhận được thông báo (thông qua một ngoại lệ) khibyte trên một trang đã được ghi vào. Có một số cách sử dụng thông minh cho lá cờ này. Windows sử dụng cờ này khi nó tạongăn xếp của luồng

4) Alignment

## **Memory access granualarity**

Khi lập trình thường thấy bộ nhớ là một chuỗi các ô nhớ liên tiếp nhau, mỗi ô nhớ tương đương với 1 byte.

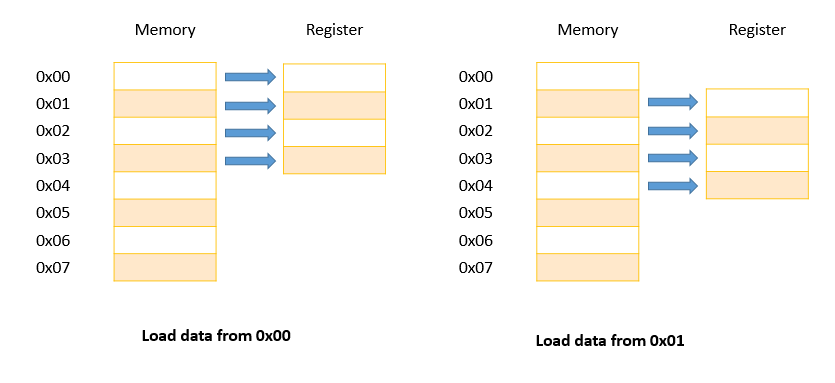
Tuy nhiên, đối với hệ thống máy tính, việc đọc và ghi với từng ô nhớ như vậy tốn nhiều thời gian dẫn tới việc giảm hiệu suất. Vì vậy, tại mỗi thời điểm, processor truy cập bộ nhớ với 2-, 4-, 8-, 16- và thậm chí 32- byte tùy vào từng hệ thống. Kích thước khi một process truy cập bộ nhớ được gọi là bộ nhớ truy cập chi tiết (Memory access granularity).

## **Data Alignment**

Xét các kết quả của các bộ nhớ truy cập chi tiết (Memory access granularity) khác nhau để có thể hiểu rõ cơ chế. Ở đây sử dụng hệ thống 32 bit - x86, nên không gian mỗi lần register load dữ liệu sẽ là 4 bytes.

### **Single-byte memory access granularity**

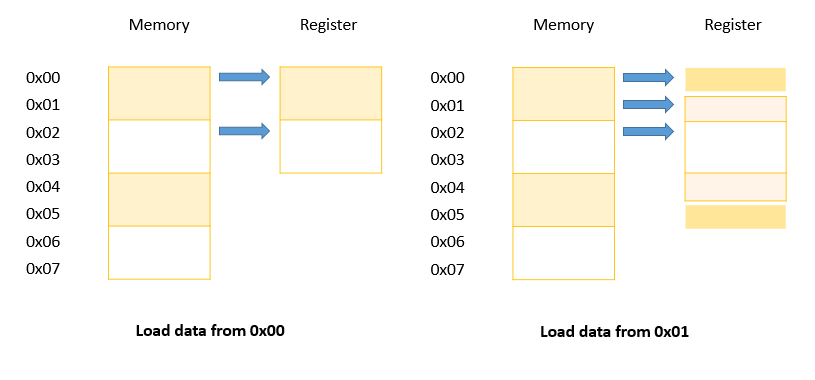
Với 1 byte, register sẽ chia làm 4 phần, hiện tại các processor không sử dụng theo cơ chế này, tuy nhiên đưa ra để so sánh với cơ chế bên dưới.



Khi đọc, ghi 4 byte từ địa chỉ 0x00 hay 0x01 đều không có sự khác biệt do truy cập ở kích thước là 1 byte.

### **Double-byte memory access granularity**

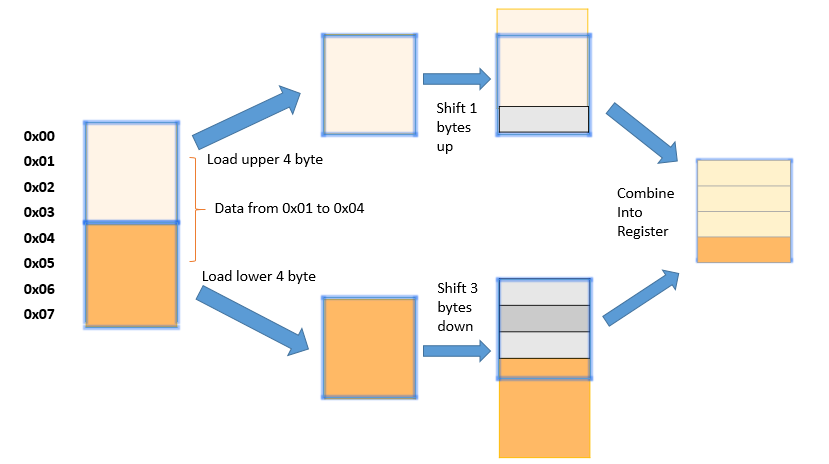
Với 2 bytes, register được chia làm ít nhất là 2 phần. Với việc tăng gấp đôi bộ nhớ truy cập chi tiết, nghĩa là công việc giảm 1 nửa, thấy được lợi ích của mô hình này khi đọc ghi dữ liệu ở địa chỉ 0x00.



Tuy nhiên khi đến địa chỉ 0x01, do không thuộc vào ranh giới truy cập (địa chỉ ô nhớ chia hết cho 2), nên processor phải làm thêm một số công việc để xử lý là truy cập thêm bộ nhớ, dẫn tới làm chậm lại các hoạt động. Dữ liệu ô nhớ không tương thích như vậy được gọi là "Unaligned memory access".

### **Quad-byte memory access granularity**

Tương tự như Double-byte memory access granularity, nhưng hãy xét cách mà processor xử lý "Unaligned memory access".



Trường hợp này, muốn lấy được vùng nhớ data, processor sẽ phải lấy 4 byte phái trên và để ô nhớ dịch lên 1 byte và lấy 4 byte phải dưới và dịch lên trên 3 byte, sau đó ghép lại sẽ thu được vùng nhớ data.

Rất nhiều công việc phải thực hiện, như vậy làm giảm rất nhiều hiệu suất của hệ thống.

Vậy nên cần tổ chức thực hiện cơ chế nào đó để không còn Unaligned memory access thì hiệu suất của hệ thống sẽ tăng lên rất nhiều. Đó là lý do tại sao cần Data Alignment.

5) Stack and Heap Memory Allocation

Phân bổ ngăn xếp: Việc phân bổ xảy ra trên các khối bộ nhớ liền kề. Chúng tôi gọi nó là cấp phát bộ nhớ ngăn xếp vì việc cấp phát xảy ra trong ngăn xếp lệnh gọi hàm. Kích thước của bộ nhớ được cấp phát đã được trình biên dịch biết trước và bất cứ khi nào một hàm được gọi, các biến của nó sẽ nhận được bộ nhớ được cấp phát trên ngăn xếp. Và bất cứ khi nào cuộc gọi hàm kết thúc, bộ nhớ cho các biến sẽ bị hủy cấp phát. Tất cả điều này xảy ra bằng cách sử dụng một số quy trình được xác định trước trong trình biên dịch. Một lập trình viên không phải lo lắng về việc cấp phát bộ nhớ và hủy cấp phát các biến ngăn xếp. Loại cấp phát bộ nhớ này còn được gọi là Cấp phát bộ nhớ tạm thời bởi vì ngay sau khi phương thức kết thúc việc thực thi, tất cả dữ liệu thuộc về phương thức đó sẽ tự động thoát ra khỏi ngăn xếp. Có nghĩa là, bất kỳ giá trị nào được lưu trữ trong lược đồ bộ nhớ ngăn xếp đều có thể truy cập được miễn là phương thức chưa hoàn thành việc thực thi và hiện ở trạng thái đang chạy.

Những điểm chính:

Đó là một lược đồ cấp phát bộ nhớ tạm thời trong đó các thành viên dữ liệu chỉ có thể truy cập được nếu phương thức () chứa chúng hiện đang chạy.

Nó tự động cấp phát hoặc hủy cấp phát bộ nhớ ngay sau khi phương thức tương ứng hoàn thành việc thực thi.

Chúng tôi nhận được lỗi Java tương ứng. lang. StackOverFlowError bởi JVM, Nếu bộ nhớ ngăn xếp được lấp đầy hoàn toàn.

Phân bổ bộ nhớ ngăn xếp được coi là an toàn hơn so với cấp phát bộ nhớ heap vì dữ liệu được lưu trữ chỉ có thể được truy cập bởi luồng chủ sở hữu.

Cấp phát và khử cấp phát bộ nhớ nhanh hơn so với cấp phát bộ nhớ Heap.

Bộ nhớ ngăn xếp có ít không gian lưu trữ hơn so với bộ nhớ Heap

Heap Allocation: Bộ nhớ được cấp phát trong quá trình thực thi các lệnh do người lập trình viết. Lưu ý rằng tên heap không liên quan gì đến cấu trúc dữ liệu heap. Nó được gọi là heap vì nó là một đống không gian bộ nhớ có sẵn cho các lập trình viên để cấp phát và khử cấp phát. Mỗi khi chúng ta tạo một đối tượng, nó luôn tạo trong Heap-space và thông tin tham chiếu đến các đối tượng này luôn được lưu trữ trong Stack-memory. Phân bổ bộ nhớ Heap không an toàn như phân bổ bộ nhớ Stack vì dữ liệu được lưu trữ trong không gian này có thể truy cập hoặc hiển thị cho tất cả các chuỗi. Nếu một lập trình viên không xử lý tốt bộ nhớ này, thì chương trình có thể bị rò rỉ bộ nhớ.

Việc cấp phát bộ nhớ Heap được chia thành ba loại: - Ba loại này giúp chúng ta sắp xếp thứ tự ưu tiên dữ liệu (Đối tượng) sẽ được lưu trữ trong bộ nhớ Heap hoặc trong bộ sưu tập Rác.

Young Generation - Đó là phần bộ nhớ nơi tất cả dữ liệu (đối tượng) mới được tạo ra để phân bổ không gian và bất cứ khi nào bộ nhớ này được lấp đầy hoàn toàn thì phần còn lại của dữ liệu sẽ được lưu trữ trong bộ sưu tập Rác.

Old or Tenured Generation - Đây là phần của bộ nhớ Heap chứa các đối tượng dữ liệu cũ hơn không được sử dụng thường xuyên hoặc không được sử dụng.

Permanent Generation - Đây là phần của bộ nhớ Heap chứa siêu dữ liệu của JVM cho các lớp thời gian chạy và các phương thức ứng dụng.

Những điểm chính:

Sơ đồ cấp phát bộ nhớ này khác với sơ đồ phân bổ không gian ngăn xếp, ở đây không cung cấp tính năng tự động phân bổ bộ nhớ. Chúng ta cần sử dụng bộ thu gom rác để loại bỏ các đối tượng cũ không sử dụng để sử dụng bộ nhớ một cách hiệu quả.

Thời gian xử lý (Thời gian truy cập) của bộ nhớ này khá chậm so với bộ nhớ ngăn xếp.

Bộ nhớ Heap cũng không an toàn theo luồng như Bộ nhớ ngăn xếp vì dữ liệu được lưu trữ trong bộ nhớ Heap được hiển thị cho tất cả các luồng.

Kích thước của bộ nhớ Heap lớn hơn khá nhiều so với bộ nhớ ngăn xếp.

Sự khác biệt chính giữa phân bổ ngăn xếp và đống

Trong một ngăn xếp, việc cấp phát và hủy cấp phát được trình biên dịch tự động thực hiện trong khi trong heap, nó cần được lập trình viên thực hiện theo cách thủ công.

Xử lý khung Heap tốn kém hơn việc xử lý khung ngăn xếp.

Vấn đề thiếu bộ nhớ có nhiều khả năng xảy ra trong ngăn xếp trong khi vấn đề chính trong bộ nhớ heap là phân mảnh.

Truy cập khung xếp chồng dễ dàng hơn khung heap vì ngăn xếp có một vùng bộ nhớ nhỏ và thân thiện với bộ nhớ cache, nhưng trong trường hợp khung heap được phân tán khắp bộ nhớ, do đó, nó gây ra nhiều bộ nhớ cache hơn.

Một ngăn xếp không linh hoạt, không thể thay đổi kích thước bộ nhớ được phân bổ trong khi một ngăn xếp linh hoạt và bộ nhớ được phân bổ có thể được thay đổi.

Thời gian truy cập của đống mất nhiều hơn một ngăn xếp.