TỔNG QUAN VỀ JVM: HOTSPOT GARBAGE COLLECTOR

by **pnhung177** — on Java , Java Performance , JVM , HotSpot VM — **12/07/2014**

Bài viết này giới thiệu về HotSpot Garbage Collector, cũng như giải thích cơ chế

Acegik

Trang bìa Danh sách Phân loại Giới thiệu Feed

TỔNG QUAN VỀ JVM: HOTSPOT GARBAGE COLLECTOR

by **pnhung177** — on Java , Java Performance , JVM , HotSpot VM — **12/07/2014**

Bài viết này giới thiệu về HotSpot Garbage Collector, cũng như giải thích cơ chế

Vùng nhớ do GC quản lý theo mô hình **Generational Garbage Collection** được chia làm 3 phần:

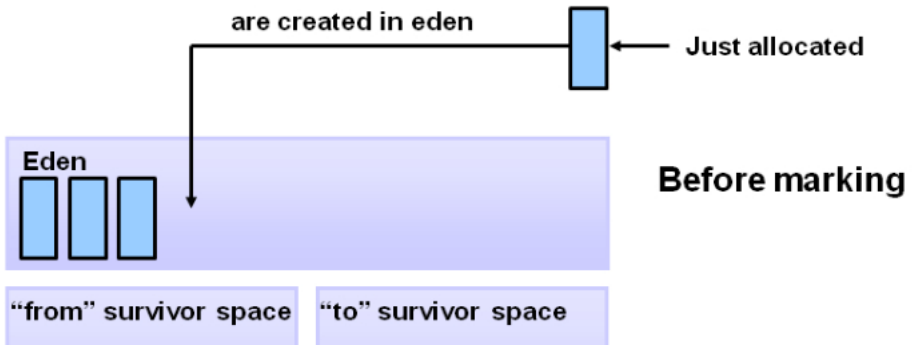
- **Young Generation**: được dùng để chứa các đối tượng mới được chương trình tạo. Vùng nhớ này được chia nhỏ thành 3 vùng con: vùng Eden và 2 vùng nhớ Survivor có tên là S0 và S1. Vùng nhớ Eden là nơi chứa các đối tượng vừa được khởi tạo. Các vùng nhớ Survivor được dùng để tính tuổi cho các đối tượng mới được chuyển qua từ vùng Eden.
- **Old Generation** hay còn gọi là **Tenured Generation**: được dùng để chứa các đối tượng tồn tại lâu dài. Các đối tượng trong vùng **Young Generation** sau khi đã trải qua một số lần thu dọn rác nhất định, nếu vẫn tồn tại thì sẽ được chuyển qua vùng **Old Generation**.
- **Permanent Generation**: được máy ảo sử dụng để chứa dữ liệu cần thiết để mô tả đối tượng. Ví dụ, đối tượng mô tả các lớp đối tượng (class) và phương thức (method) được lưu trữ trong vùng Permanent Generation.

Các vùng nhớ hoạt động như thế nào?

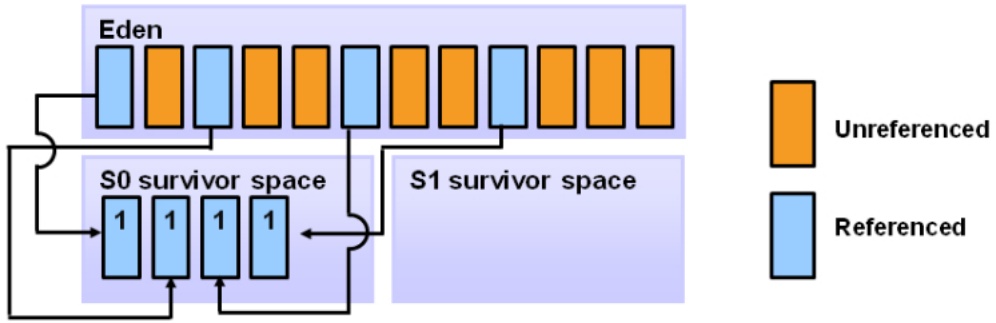
Vùng nhớ Young Generation

Vùng nhớ Young Generation được dùng để chứa những đối tượng vừa được chương trình tạo ra hoặc đã tồn tại trong chương trình một thời gian ngắn. Vùng nhớ này giãn nở nhanh chóng và khi đã đầy sẽ được *quá trình thu dọn nhỏ* (**minor garbage collection**) dọn dẹp các đối tượng không dùng nữa và chuyển những đối tượng tồn tại lâu dài sang vùng Old Generation. Cơ chế này được giải thích chi tiết như sau:

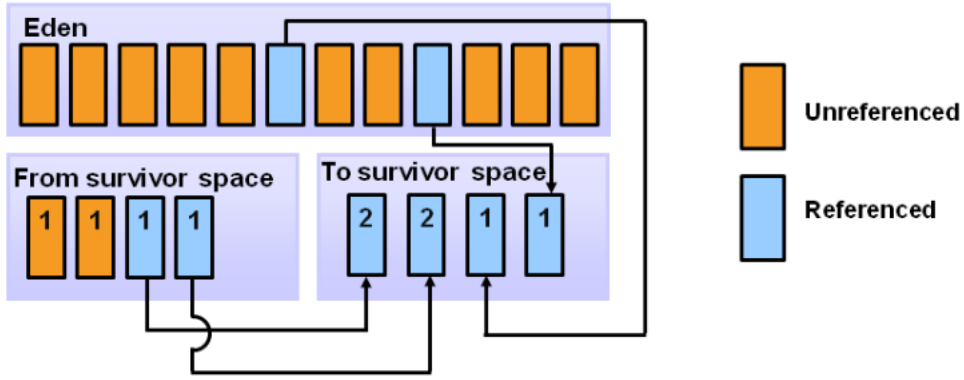
1. Khi một đối tượng được tạo ra, nó được cấp phát bộ nhớ trong vùng Eden. Hai vùng nhớ Survivor S0 và S1 khi mới bắt đầu đều trống rỗng. Vùng nhớ Eden tiếp tục nhận các đối tượng mới tạo cho đến khi đầy.



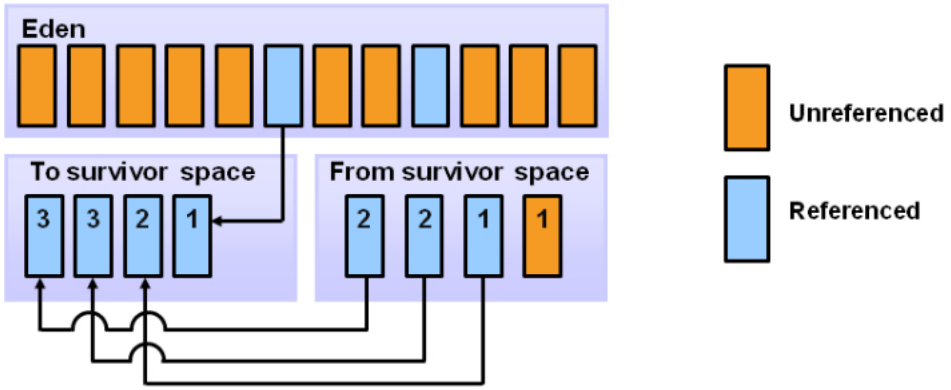
2. Khi Eden đầy, *quá trình thu dọn nhỏ* (**minor garbage collection**) được kích hoạt. Quá trình này quét qua toàn bộ vùng Eden, những đối tượng còn tham chiếu (Referenced object) sẽ được chuyển qua vùng Survivor thứ nhất (vùng S0), đồng thời được đánh dấu tuổi bằng 1. Những đối tượng hết tham chiếu (Unreferenced object) sẽ loại bỏ khi vùng Eden bị xóa vào cuối quá trình.



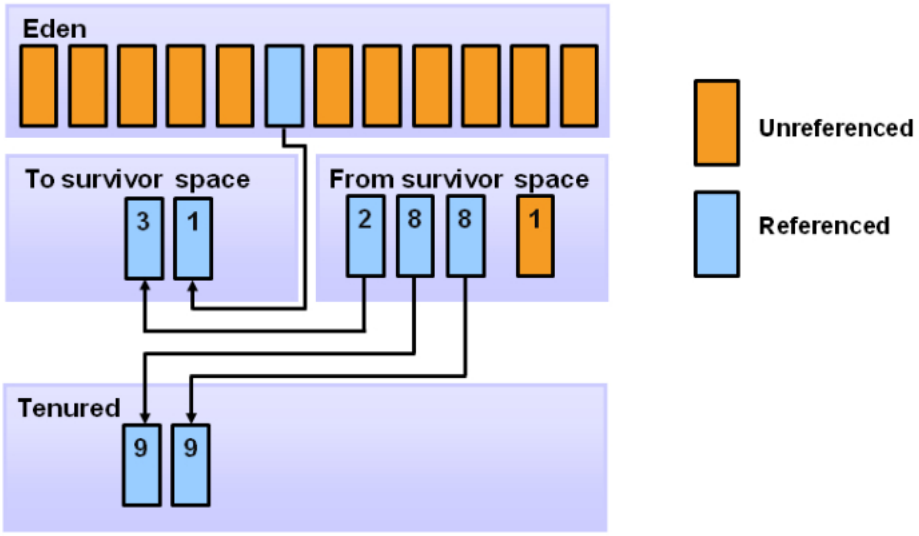
3. Với vùng Eden đã được dọn sạch ở bước trước, quá trình cấp phát bộ nhớ lại tiếp tục cho đến khi vùng này đầy. Quá trình thu dọn rác thứ cấp được kích hoạt trở lại. Trong lần này, các đối tượng "còn tham chiếu" trong Eden được chuyển sang vùng Servivor thứ hai (vùng S1). Sau đó, các đối tượng trong vùng S0 được chuyển từ Eden qua trong lần dọn trước sẽ được quét, những đối tượng "còn tham chiếu" sẽ được tăng số tuổi lên một đơn vị và được chuyển qua vùng Survivor S1. Cuối quá trình, vùng Eden và vùng S0 chỉ còn lại các đối tượng "hết tham chiếu" và sẽ bị xóa sạch.



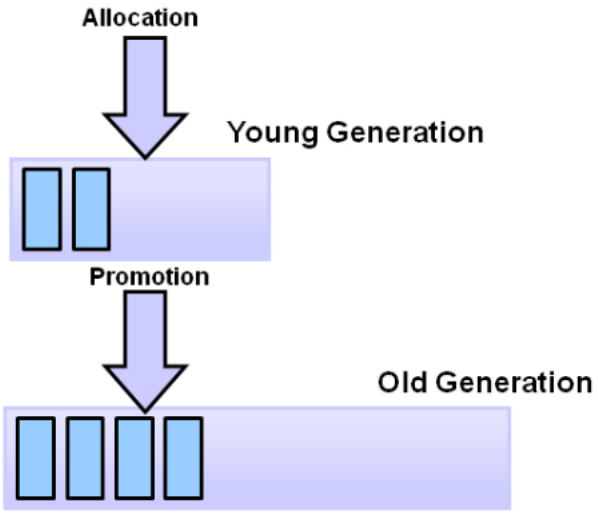
4. Quá trình cấp phát bộ nhớ lại được lặp lại. Tuy nhiên, trong bước này, vai trò của S0 và S1 được hoán đổi với nhau: Khi Eden đầy, các đối tượng "còn tham chiếu" được chuyển từ Eden và S1 sang S0 và được tăng số tuổi. Cuối quá trình vùng Eden và S1 được dọn sạch.



5. Sau mỗi một lần thực hiện *quá trình thu dọn nhỏ*, nếu đối tượng nào có tuổi đạt được ngưỡng (chẳng hạn trong hình minh họa là 8), nó sẽ được thăng cấp (promote) từ vùng Young Generation lên vùng Old Generation (còn gọi là Tenured Generation).



6. Quá trình này được thực hiện lặp đi lặp lại liên tục. Chỉ những đối tượng tồn tại lâu dài (có tuổi đạt ngưỡng) thì mới được thăng cấp từ "Young Generation" qua vùng "Old Generation".



Vùng nhớ Old Generation

Như đã đề cập ở phần trên, vùng nhớ Old Generation được dùng để lưu trữ các đối tượng tồn tại lâu dài được chuyển qua từ vùng nhớ Young Generation. Vùng nhớ Old Generation thường lớn hơn nhưng giãn nở chậm hơn so với Young Generation. Khi vùng nhớ này đầy, *quá trình thu dọn lớn* (**major garbage collection**) được kích hoạt để quét qua tất cả các đối tượng và loại bỏ các đối tượng "hết tham chiếu". Quá trình thu dọn chính còn được gọi là *quá trình thu dọn toàn bộ* (**full garbage collection**).

Thông thường quá trình thu dọn lớn sẽ mất thời gian hơn rất nhiều (so với *quá trình thu dọn nhỏ*). Quá trình này cũng thuộc loại sự kiện "Stop the World": ngưng tất cả mọi thứ trong khi nó vận hành. Do đó, đối với những ứng dụng có độ đáp ứng cao (responsive application), quá trình thu dọn lớn phải được hạn chế tối đa.

Vùng nhớ Permanent Generation

Mặc dù tên gọi nằm trong nhóm "generation", nhưng vùng nhớ này không nằm trong cơ cấu hoạt động dịch chuyển và thăng cấp đối tượng (tức là, các đối tượng do người lập trình tạo ra sẽ không bao giờ được chuyển từ vùng "Old generation" sang vùng "Permanent generation"). Thay vào đó, vùng nhớ này chỉ được sử dụng bởi HotSpot VM để lưu trữ **metadata**, chẳng hạn như các cấu trúc dữ liệu lớp đối tượng, interned strings, v.v. Ngoài ra, các lớp đối tượng thư viện và các phương thức của JavaSE cũng được chứa ở đây.

Các lớp đối tượng có thể có thể được gỡ bỏ nếu JVM nhận thấy không cần dùng chúng nữa. Việc thu dọn cũng được thực hiện trong một *quá trình thu dọn toàn bộ* (**full garbage collection**).

Author

PHẠM NGỌC HÙNG

pnhung177@drupalex.net

... when the sun comes up, you better be running!

COMMENTS

ALSO ON

TỔNG QUAN Về IOC TRONG SPRING FRAMEWORK | ACEGIK'S BLOG

2 comments • 5 years ago

La Kiến Vinh

HOWTO NEST LIQUID TEMPLATE VARIABLES INSIDE YAML FRONT MATTER BLOCK | ...

4 comments • 4 years ago

Dwouglas Mhagnum

THEO DõI HIệU NăNG CủA JVM | ACEGIK'S BLOG

1 comment • 5 years ago

Nguyen Ngoc Tu

TỔNG QUAN Về DEPENDENCY INJECTION TRONG ANGULARJS (P1) | ACEGIK'S BLOG

2 comments • 5 years ago

pnhung177

https://www.facebook.com/da...