TRUNG TÂM GIẢI PHÁP VIỄN THÔNG VÀ CÔNG NGHỆ

THÔNG TIN VIETTEL

TÀI LIỆU HƯỚNG DẪN THEO DÕI VÀ CẢI TIẾN HIỆU NĂNG ỨNG DỤNG JAVA

Hà nội, 11/2015.

BẢNG GHI NHẬN THAY ĐỔI

*A – Tạo mới, M – Sửa đổi, D – Xóa bỏ

Ngày thay đổi	Vị trí thay đổi	A* M, D	Nguồn gốc	Phiên bản cũ	Mô tả thay đổi	Phiên bản mới
13/11/2015		M			Tạo mới	1.0

MỤC LỤC

1. GIỚI THIỆU

1.1. Mục đích và ý nghĩa của Tài liệu

Vấn đề về hiệu năng của ứng dụng do rất nhiều yếu tố quyết định, như: kiến trúc ứng dụng, băng thông đường truyền, năng lực xử lý của app server/database server... Tài liệu này được viết với mục đích giới thiệu về một số khái niệm liên quan đến hiệu năng ứng dụng; hướng dẫn người dùng sử dụng công cụ Java YourKit Profiler để theo dõi hiệu năng ứng dụng và phát hiện các lỗ hổng về hiệu năng như performance bottleneck, memory leak.

1.2. Phạm vi tài liệu

Phạm vi tài liệu chỉ đề cập đến các vấn đề liên quan đến hiệu năng ứng dụng Java (sử dụng JVM). Các vấn đề này xảy ra do chủ quan người lập trình gây ra. Các vấn đề liên quan bên ngoài ứng dụng không được đề cập tới.

Tài liệu này phục vụ cho các lập trình viên cần theo dõi, rà soát và tối ưu hóa hiệu năng ứng dụng Java.

1.3. Các thuật ngữ và từ viết tắt

Thuật ngữ	Định nghĩa	Ghi chú
Bottleneck	Nút thắt cổ chai	
GC	Garbage Collection	
JVM	Java Virtual Machine	

1.4. Cấu trúc Tài liệu

Tài liệu này gồm 3 phần, được bố trí như sau:

Phần 1 Giới thiệu: Giới thiệu về mục tiêu, phạm vi, đối tượng sử dụng tài liệu .....

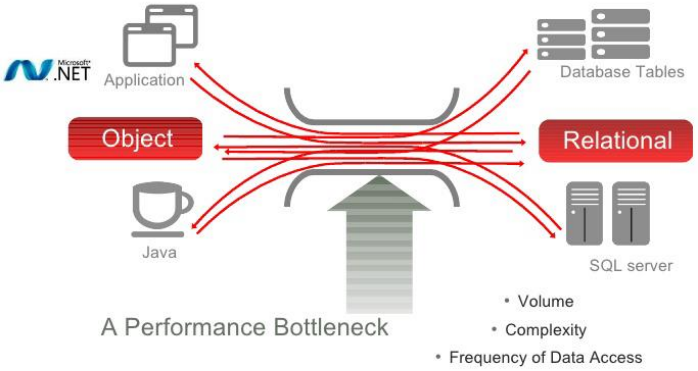
Phần 2 Một số hiện tượng liên quan đến hiệu năng ứng dụng: Giới thiệu sơ qua về một số hiện tượng liên quan đến hiệu năng ứng dụng thường gặp phải.

Phần 3 Theo dõi và xử lý các vấn đề hiệu năng thông qua công cụ YourKit Java Profiler

2. MỘT SỐ HIỆN TƯỢNG LIÊN QUAN ĐẾN HIỆU NĂNG ỨNG DỤNG

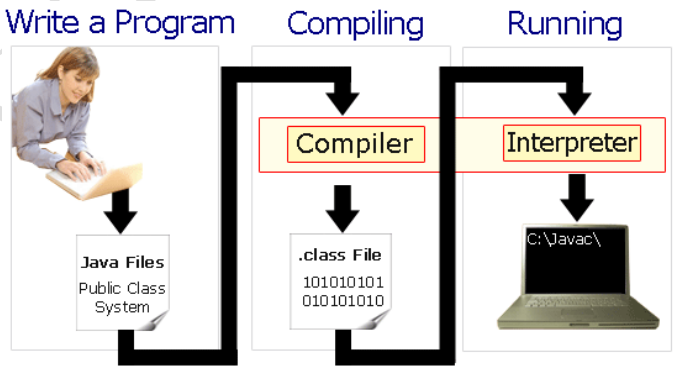
2.1. Hiện tượng Bottlenecks (thắt cổ chai)

Hiện tượng thắt cổ chai là hiện tượng mà bất kì một tài nguyên nào đó như: phần cứng, phần mềm hay mạng giới hạn khả năng thực thi của ứng dụng.

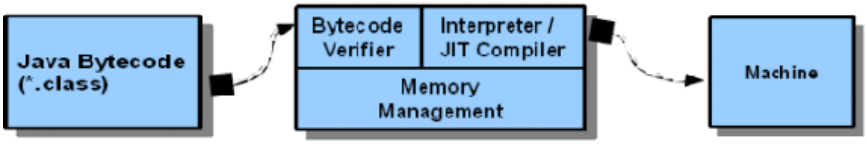


2.2. Hot spots (điểm nóng)

Để hiểu thế nào là một điểm hot spots, ta nắm lại cơ bản về quá trình thực thi một ứng dụng Java như sau:



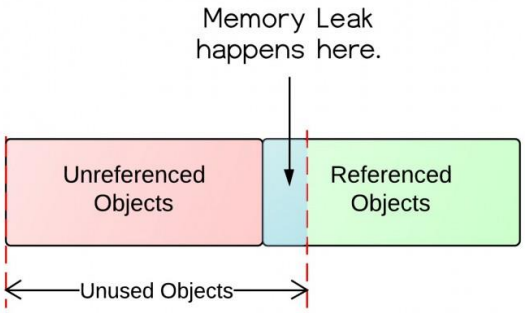
Trình thông dịch (interpreter) thường chạy với tốc độ khá chậm do phải chuyển đổi từng lệnh từ mã bytecode sang mã máy. Đối với những khối lệnh được thực thi lặp đi lặp lại nhiều lần, việc áp dụng thông dịch với chúng sẽ rất phí phạm thời gian. Chính vì vậy, JVM kết hợp sử dụng Interpreter với một dạng trình biên dịch "tức thời" khi chạy (Just-In-Time Compiler).



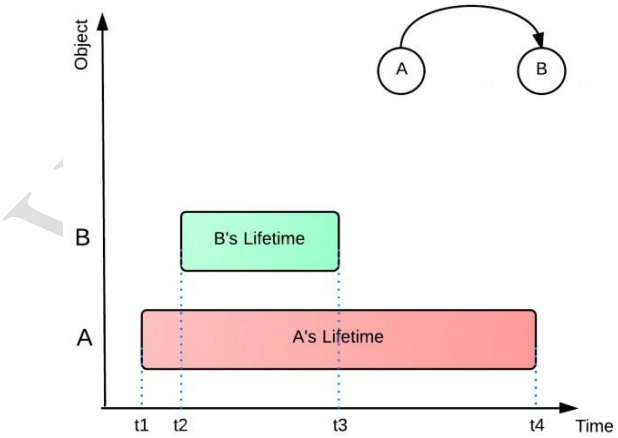
JVM sẽ phân tích để xác định các khối lệnh thực thi nhiều lần (**gọi các vùng mã lệnh này là các hotspot**) để chuyển cho JIT Compiler biên dịch thành mã máy rồi mới thực thi. Những lệnh chỉ thực thi một lần sẽ được phân công cho interpreter.

2.3. Memory leaks

Là những tình huống xảy ra khi có các đối tượng hết sử dụng, không còn tham chiếu trỏ đến nhưng vẫn tồn tại trong bộ nhớ mà không bị thu dọn rác (Garbage Collector) thu hồi.



Ví dụ sau sẽ nói rõ tác hại của memory leaks đối với một ứng dụng. Đối tượng A liên quan đến đối tượng B. Thời gian sống của A (t1-t4) lớn hơn thời gian sống của B (t2-t3). Khi B không sử dụng, A vẫn giữ liên kết đến nó. Trong trường hợp này, GC (Garbage Collector) không thể xóa B khỏi memory. Nếu có nhiều đối tượng như vậy trong chương trình sẽ làm tiêu thụ nhiều không gian bộ nhớ, ảnh hưởng đến hiệu năng của chương trình.



3. THEO DÕI VÀ XỬ LÝ CÁC VẤN ĐỀ HIỆU NĂNG THÔNG QUA CÔNG CỤ YOURKIT JAVA PROFILER

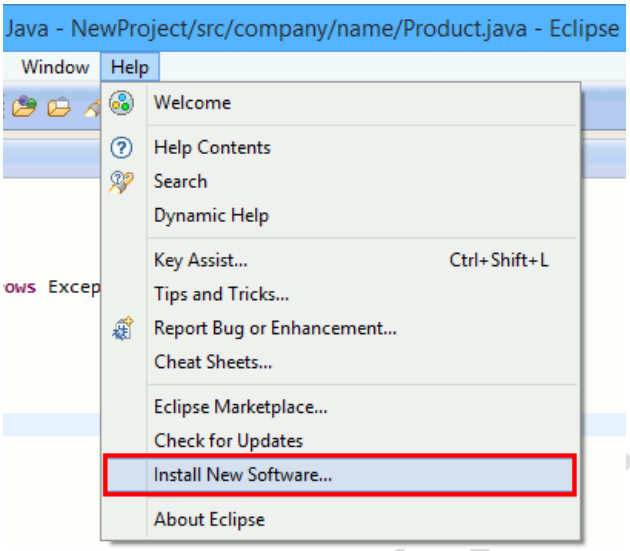
Công cụ Yourkit java profiler là công cụ để theo dõi, kiểm soát hiệu năng và phát hiện các vấn đề liên quan đến ứng dụng java như:

- Nghẽn cổ chai (bottlenecks)
- Tối ưu hóa bộ nhớ sử dụng
- Memory leaks
- ....

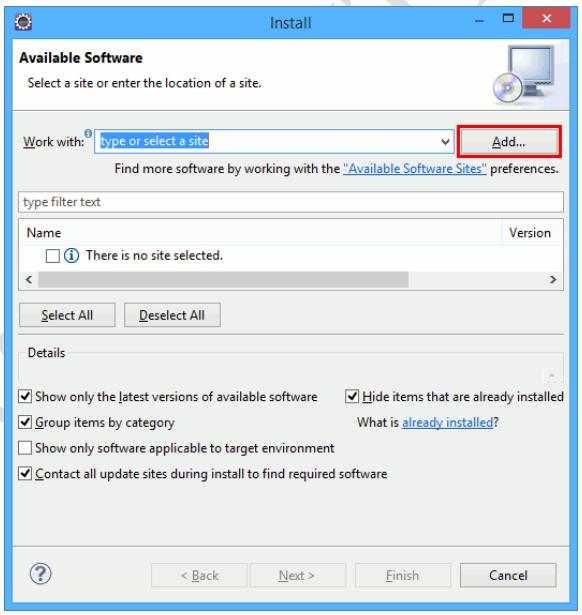
Tài liệu này sẽ tập trung vào hướng dẫn cho bạn cách sử dụng Yourkit để giải quyết vấn đề.

3.1. Cài đặt và tích hợp với Eclipse

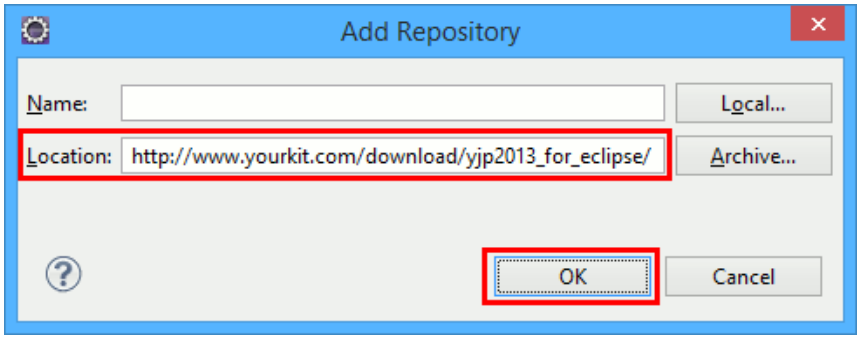
- Download bộ cài và tiến hành cài đặt. Bộ cài tại đây.
- Thực hiện các bước sau để tích hợp với IDE Eclipse.
  Vào menu Help | Install New Software...



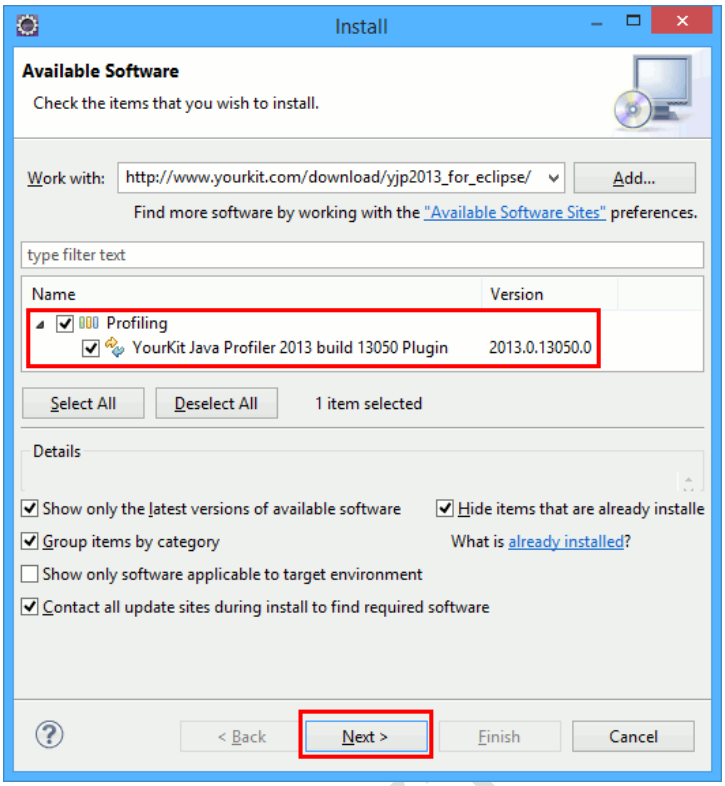
Click "Add":



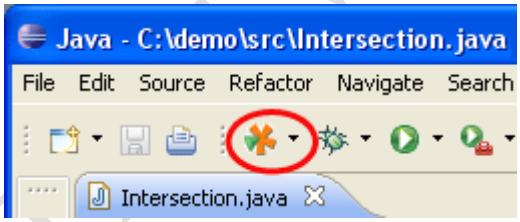
Copy URL sau vào location: http://www.yourkit.com/download/yjp2015_for_eclipse/



Lựa chọn YourKit Java Profiler plugin và click "Next":



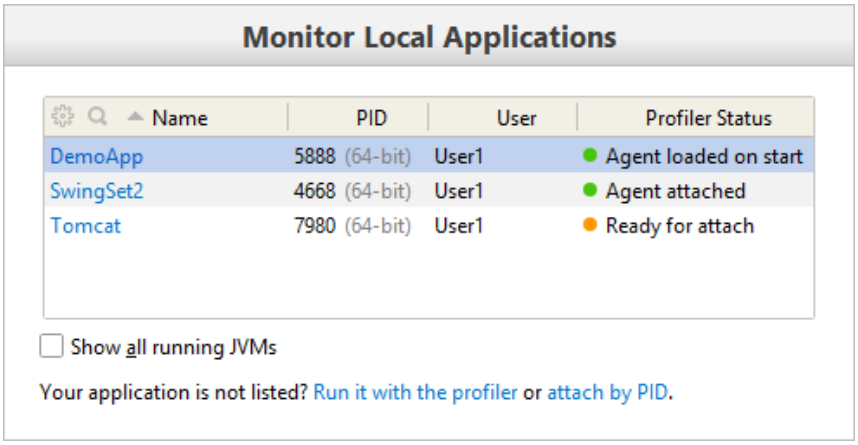
Sau khi kết thúc cài đặt thành công. Plugin YourKit sẽ hiển thị trên toolbar.



3.2. Kết nối với ứng dụng cần profiled

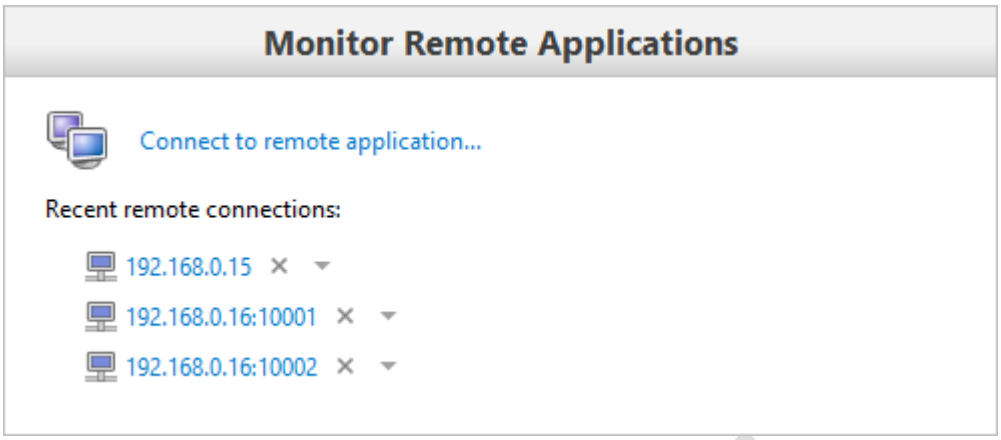
3.2.1. Kết nối với các ứng dụng nội bộ

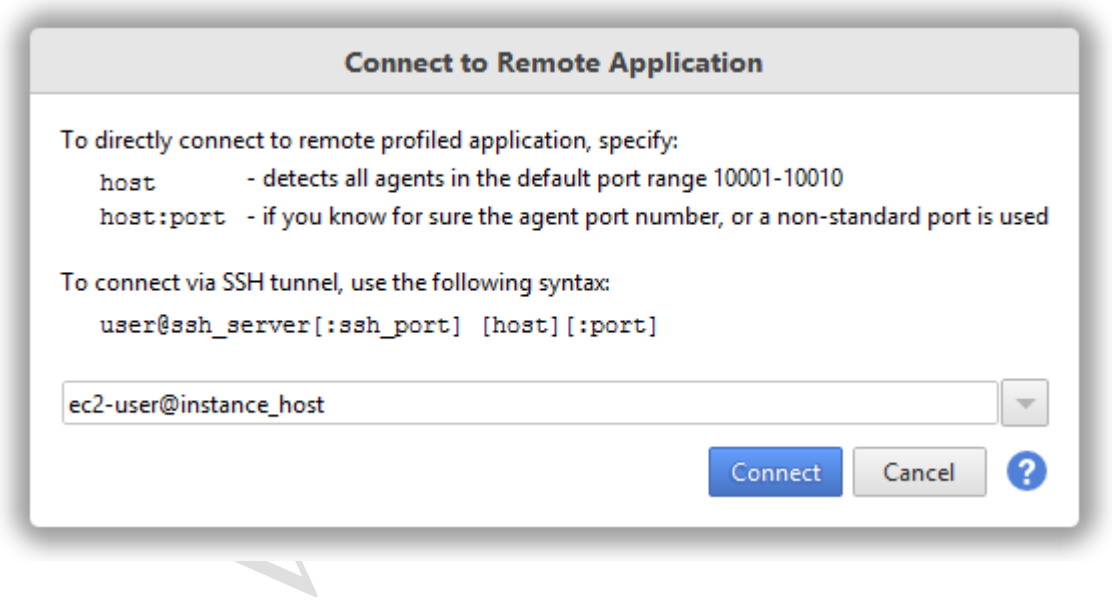
Các ứng dụng nội bộ là các ứng dụng chạy cùng trên máy tính sử dụng YourKit. Yourkit sẽ tiến hành detect tự động tất cả các ứng dụng đang chạy trên máy tính đó.



3.2.2. Kết nối với các ứng dụng từ xa

Để tiến hành kiểm tra hiệu năng của ứng dụng từ xa, sử dụng tính năng "Monitor Remote Applications" để kết nối đến ứng dụng cần kiểm tra.





3.3. Xử lý các vấn đề với YourKit:

3.3.1. Phát hiện các nút thắt cổ chai (Bottlenecks)

Giả sử bạn có một "task nghi vấn" trong ứng dụng cần kiểm tra vấn đề liên quan đến nút cổ chai, tiến hành các bước sau:

- Chạy ứng dụng với Yourkit profiler.
- Kết nối với ứng dụng.
- Tiến hành "Start CPU profiling" ngay trước khi thực hiện "task" nghi vấn được thực hiện.
- Chờ đến khi "task nghi vấn" thực hiện xong hoặc đủ lâu.
- Capture CPU snapshot
- Mở snapshot. Sử dụng CPU view phân tích các tham số CPU Time (Thời gian gọi ứng dụng) và Invocation count (số lần thực thi) để phát hiện nguyên nhân.

3.3.2. Phát hiện memory leaks và tối ưng hóa bộ nhớ

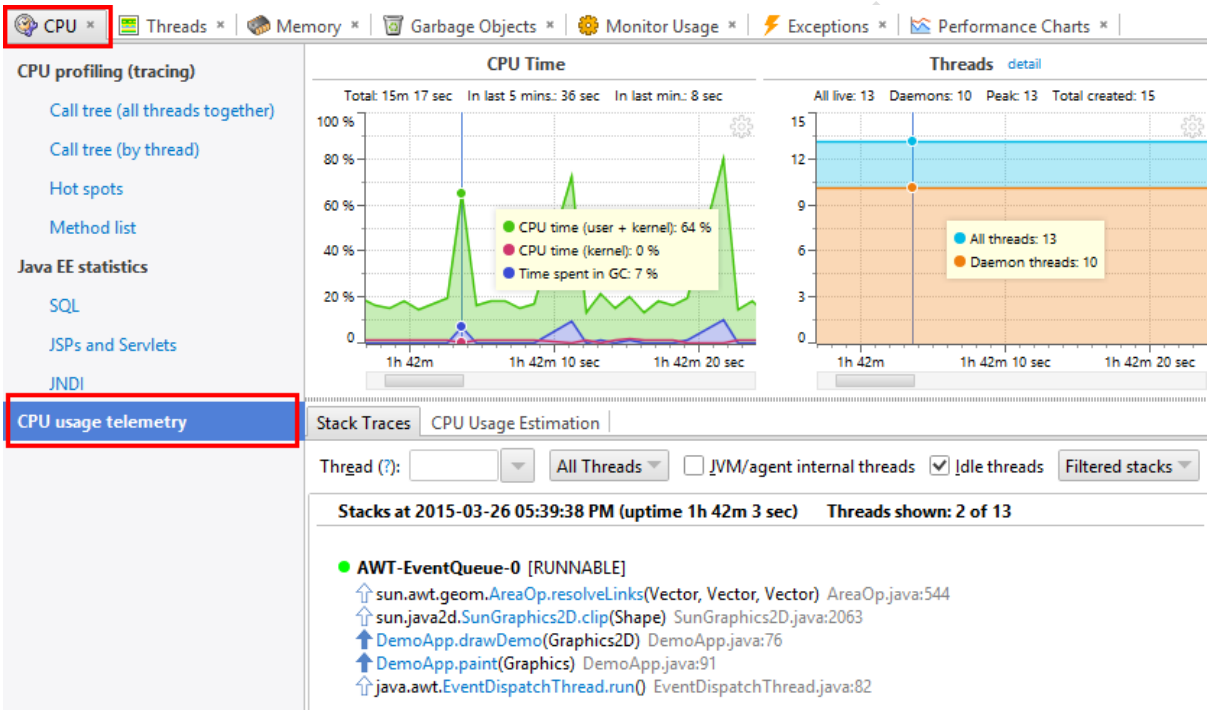
Để phát hiện ứng dụng chiếm dụng nhiều bộ nhớ, thực hiện các bước sau:

- Chạy ứng dụng với Yourkit profiler
- Kết nối với ứng dụng.
- Tiến hành capture memory snapshot khi thấy memory tăng lên (sử dụng Telemetry).
- Mở snapshot. Sử dụng memory view phân tích các tham số: số lượng đối tượng, shallow size, retain size để xác định nguyên nhân và đưa ra cách thức xử lý.

3.4. Theo dõi CPU

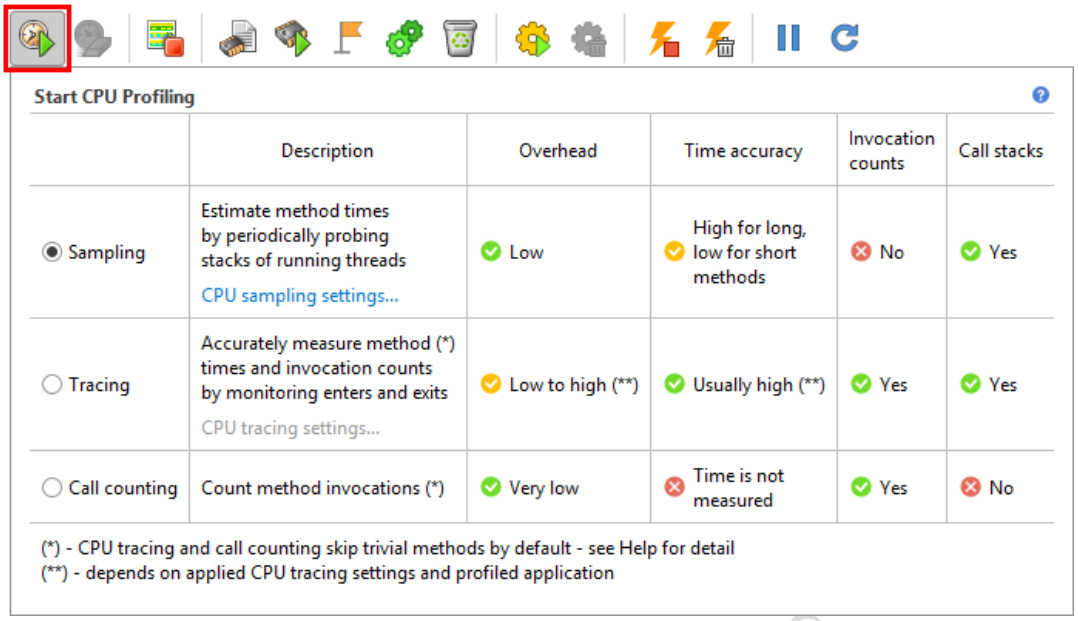
3.4.1. CPU usage telemetry

Khi đã kết nối đến ứng dụng cần theo dõi, để theo dõi CPU sử dụng màn hình "CPU usage telemetry". Màn hình này sẽ cho chúng ta biết ứng dụng sử dụng CPU theo thời gian và tất cả các threads ứng dụng chạy.



3.4.2. Start CPU profiling

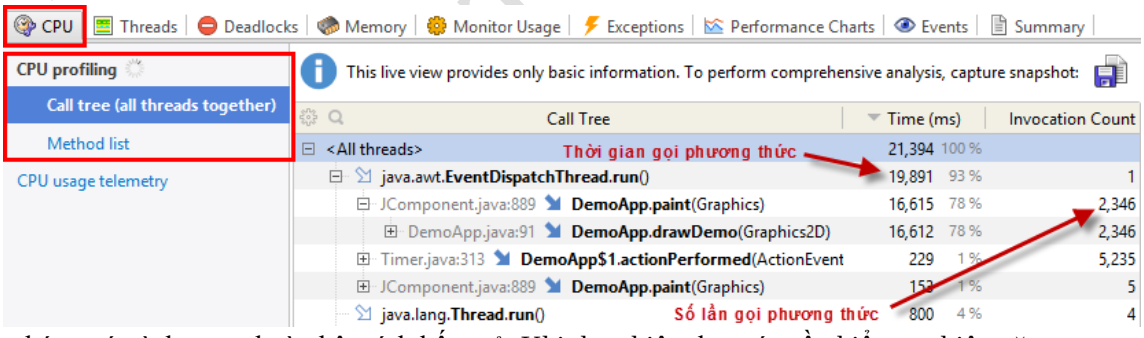
Để có được kết quả đo, tiến hành "Start CPU profiling" như hình dưới:



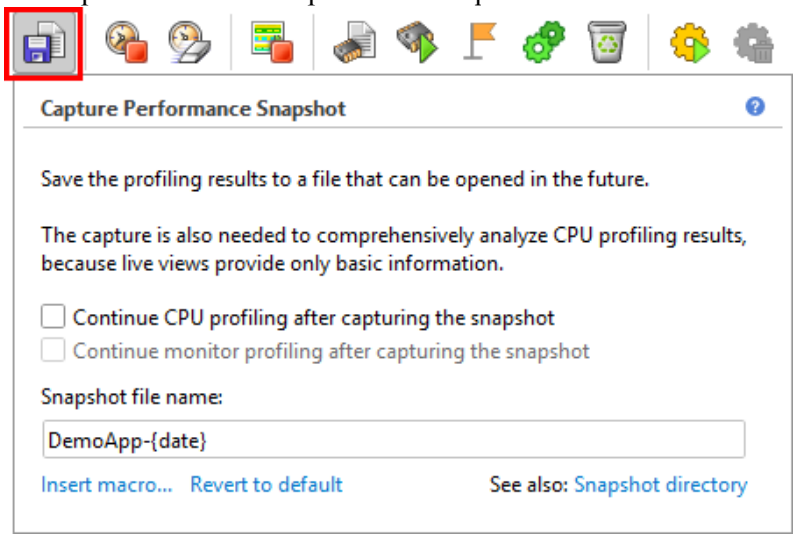
Có 3 chế độ profiling:

- Sampling: Không quan tâm đến các method được gọi, chỉ quan tâm đến CPU time. Sử dụng trong trường hợp muốn phát hiện các điểm "Performance bottlenecks".
- Tracing: Cung cấp thêm thông tin về số lượng method được gọi.
- Call counting:

Kết quả realtime (**Live results**): Khi CPU profiling được bật, kết quả ngay lập tức được hiển thị trong tab "Call tree" và "Method list" như hình dưới. Với chế độ CPU tracing thì cả thông tin method time và invocation count đều được hiển thị, còn với chế độ CPU sampling chỉ thông tin về CPU time được hiển thị.



Kết thúc quá trình record và phân tích kết quả: Khi thực hiện thao tác cần kiểm tra hiệu năng xong, tiến hành lưu lại bản snapshort và mở ra để phân tích kết quả.



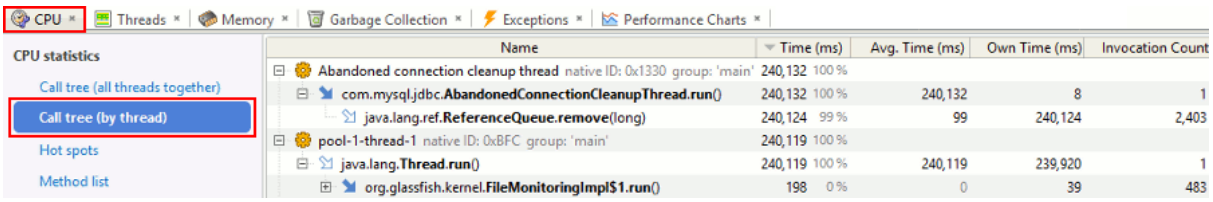
3.4.3. CPU view

CPU view thể hiện chi tiết việc tiêu thụ CPU của ứng dụng (sử dụng kết quả record trên). Chúng ta sẽ dựa trên 2 thông số đo đạc chính: Time (Thời gian gọi ứng dụng) và Invocation count (số lần thực thi) để dự đoán và xác định nguyên nhân xảy ra bottleneck.

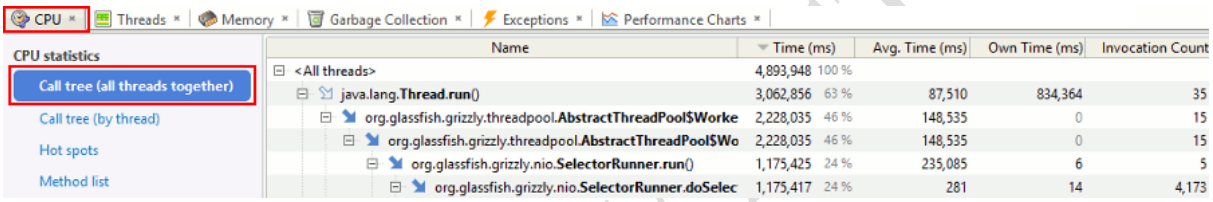
CPU view bao gồm những sections sau:

Call tree:

Hiển thị cây top-down từ trên xuống cho mỗi thread

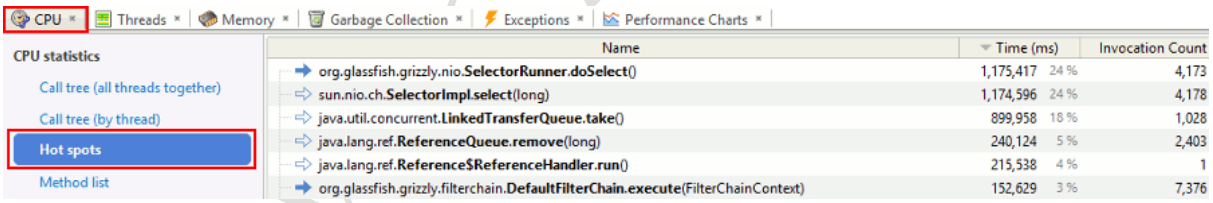


Hoặc cây các hàm được gọi gom chung các thread (all threads together)

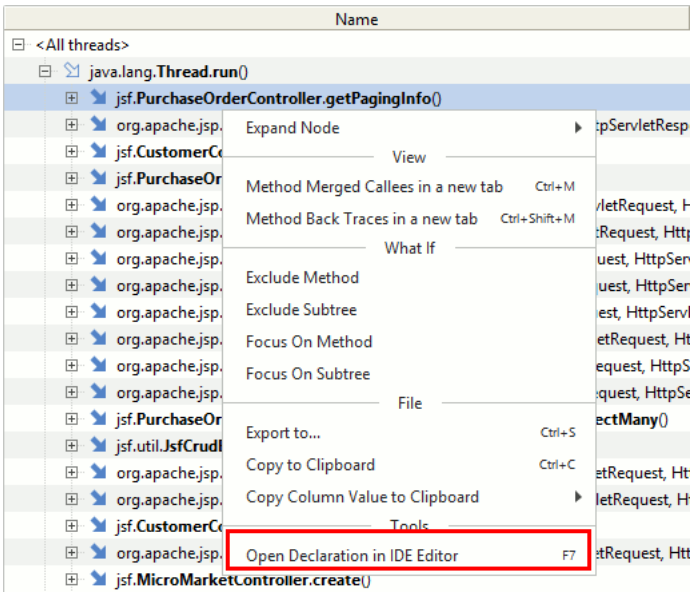


Hot spots:

Hiển thị các điểm hotpots (methods được gọi nhiều lần)



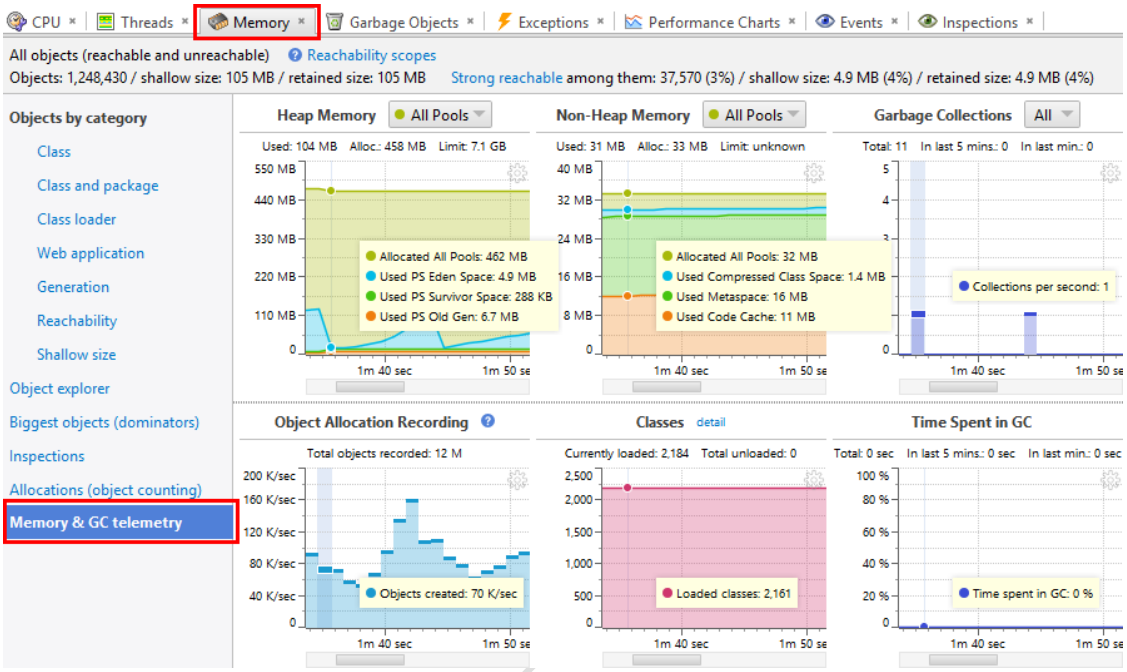
Để hiển thị chi tiết các method trên IDE, chọn phương thức đó và nhấn F7.



3.5. Theo dõi Memory

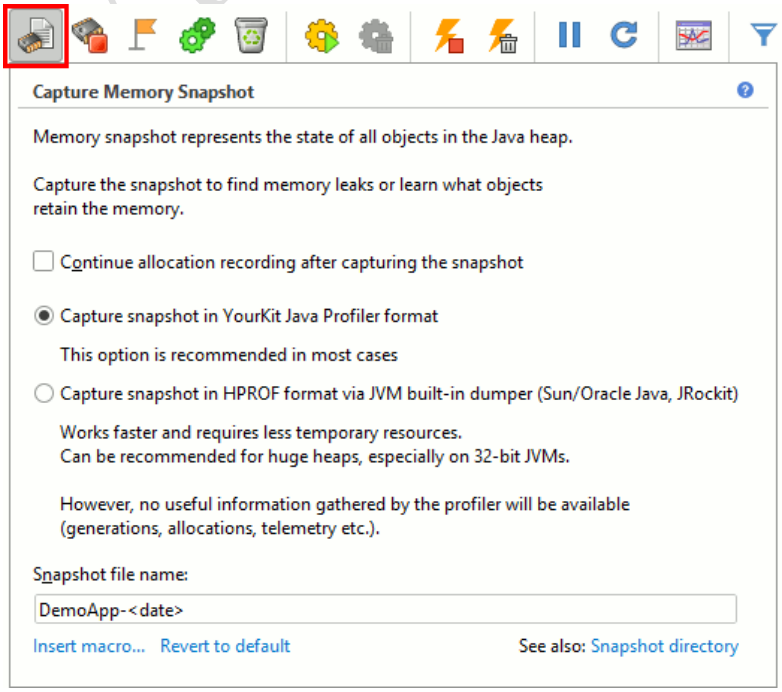
3.5.1. Memory telemetry

Màn hình này cho phép theo dõi việc tiêu thụ bộ nhớ thực tế của ứng dụng thông qua các biểu đồ về Heap memory, Non-heap memory, Class, Garbage Collection và Time spent in GC.

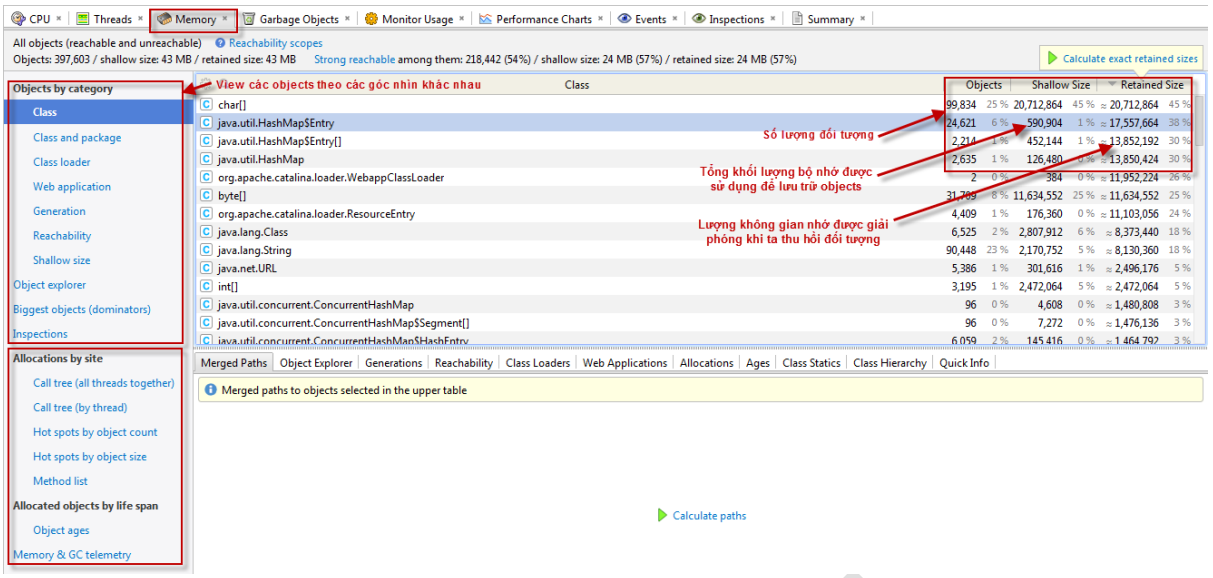


3.5.2. Memmory views

Để xem đầy đủ các thông tin về bộ nhớ, bạn phải lưu lại một snapshot của bộ nhớ tại một thời điểm nào đó. Bản snapshot này chứa đầy đủ thông tin về các class được tải, các objects tồn tại và tất cả các objects tham chiếu.



Memory view hiển thị như sau:



Thông qua góc nhìn này chúng ta có thể phát hiện ra nguyên nhân gây ra các vấn đề liên quan đến bộ nhớ.

Màn hình view "**Biggest objects**" hiển thị tất cả các objects chiếm giữ hầu hết bộ nhớ. Các đối tượng này hiển thị dưới dạng cây dominator. Thông qua góc nhìn này chúng ta có thể phát hiện các đối tượng gây ra hiện tượng "**Memory leaks**"