# 第2回若手勉強会

## パフォマンスチューニングについて

「オンラインが遅い」、「バッチが遅い」などレスポンスが悪い状態、バッチの実行が終わらず業務に問題がでる状態の原因を解析し、解決する方法

## ボトルネックとなる箇所について

大きく分けて以下のとおりとなる

### Hardware

- Disk
- Network
- CPU
- Memory

### Middleware

- AP
- DB
- OS

### Application

- Logic
- アルゴリズムの計算量
- APIの使用方法
- 言語のお作法
- コーディングミス

## 計測する

主に以下のフローですすめることになる。

1. 負荷をかける
2. 計測する
3. ボトルネック箇所特定
4. 頑張って早くする。

## 負荷をかける

### 想定データの準備

負荷テストを実施する。 テスト方法としては想定件数×N倍で試す。 DBにデータを生成する。 データを増幅させる（SQL, プログラム）

想定時間ないで本当に完了するかを試す。 この時点ではまだプロファイルは実施しない。

## 計測する

負荷をかけてみて問題があるようであればデータ件数を少なくし、CPUの使用量、メモリの使用量などを確認する。

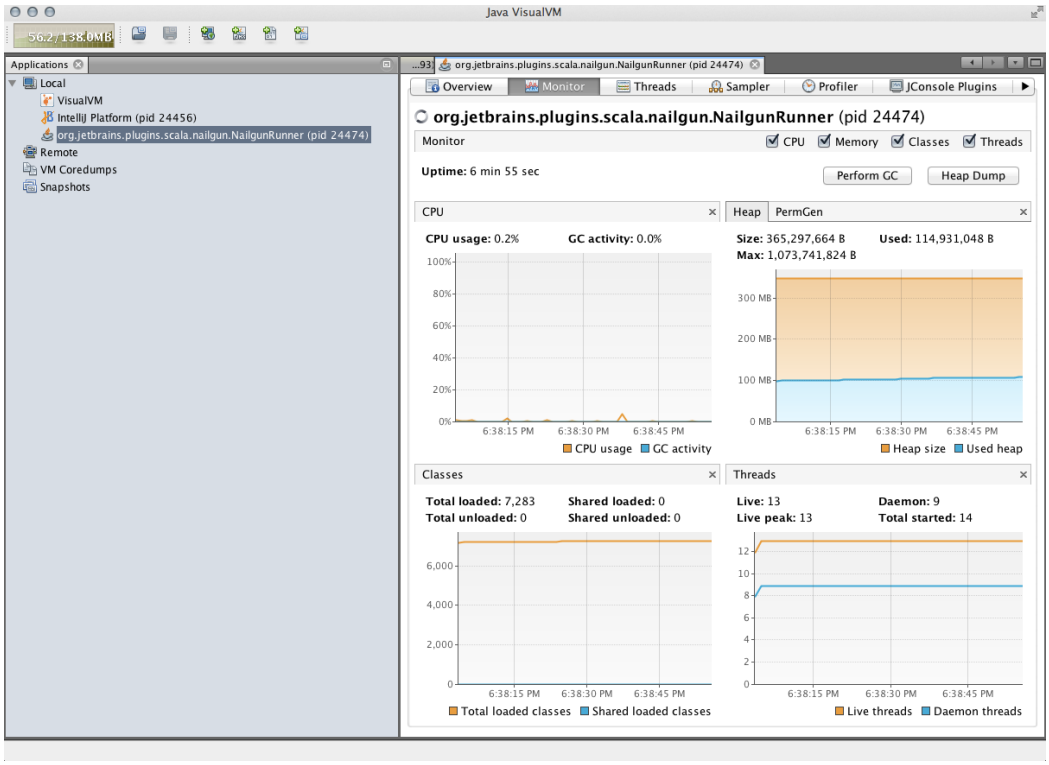
Javaのプログラムの計測する、ボトルネック箇所特定をするツールである、JDK標準のVisualVMを使用することでかなり多くの情報を取得することができる。

### Monitorタブ

monitorタブでは以下の情報を見ることができる。

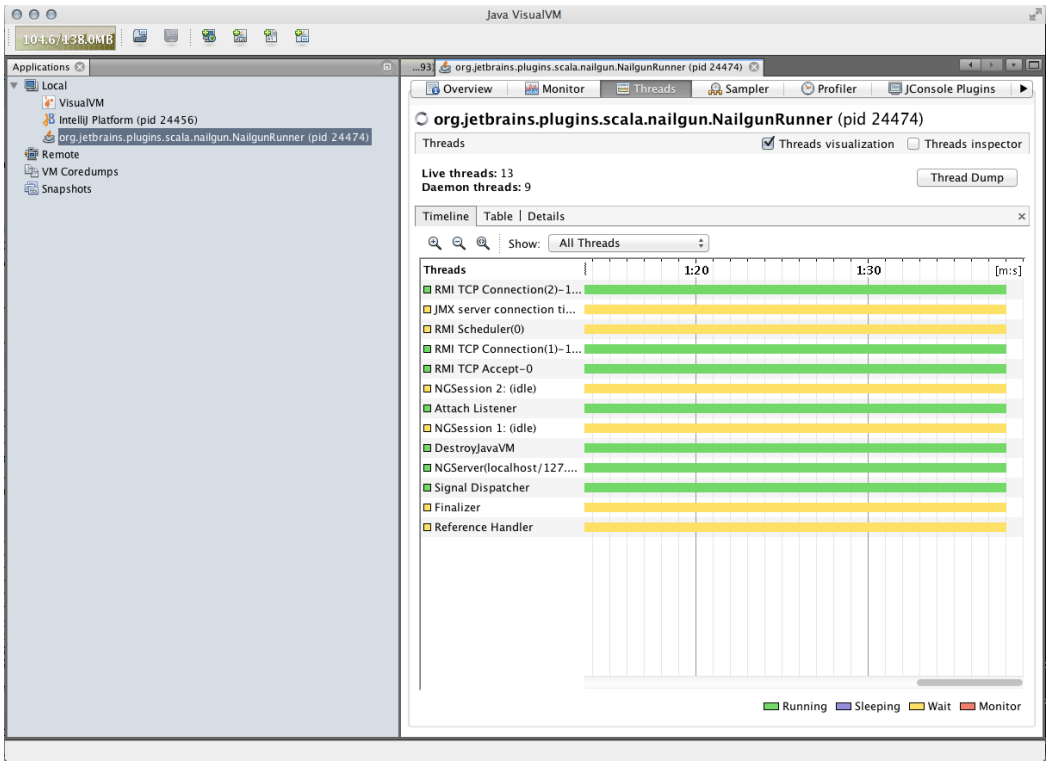
- CPU使用率
- メモリー使用量
- クラスロード数
- スレッド数

メモリーリークの有無などを確認できる。 右上部のチェックボックスは、[Perform GC]はアプリケーションに対してGCの実行を促すボタン、 [Heap Dump]はアプリケーションのヒープダンプを取るボタン。




## Threadタブ

現在動いているスレッド、停止しているスレッドなどが確認でき、 スレッドのデッドロックなどを検出することが可能。

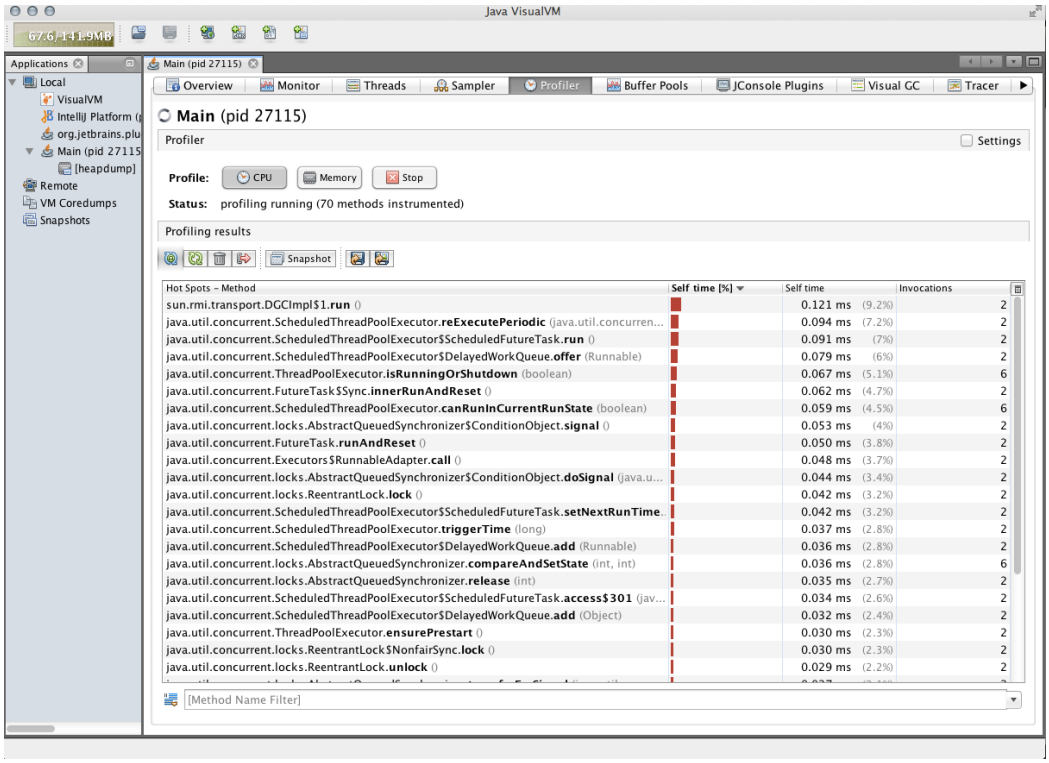



## プロファイルタブ

VisualVM では2種類のプロファイルが可能。

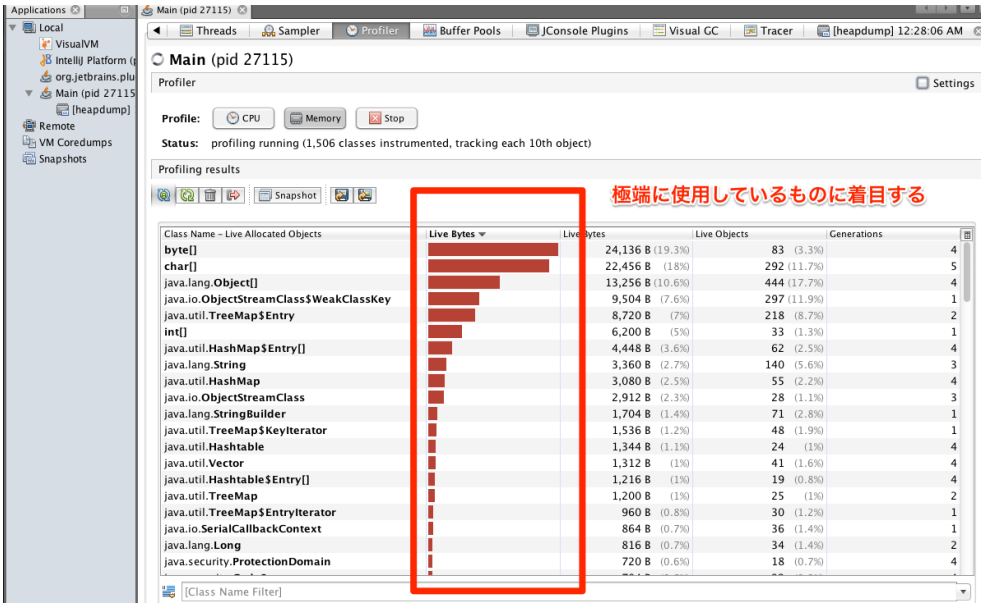
### CPUプロファイル

- CPUプロファイル: メソッドの実行に要した時間やメソッドコールの回数などを解析すること可能




## Memoryプロファイル

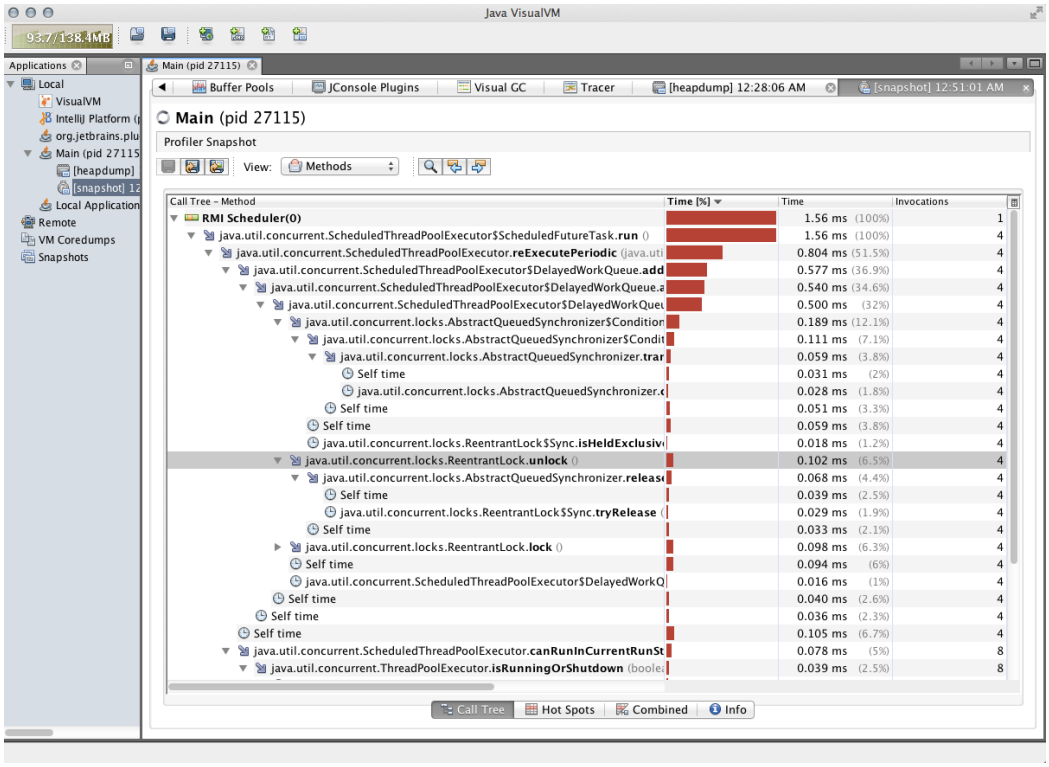
- Memoryプロファイル: オブジェクトの個数やサイズ、また世代(GCを経てきた回数)などを解析することが可能




## スナップショット

### CPUプロファイルのスナップショット

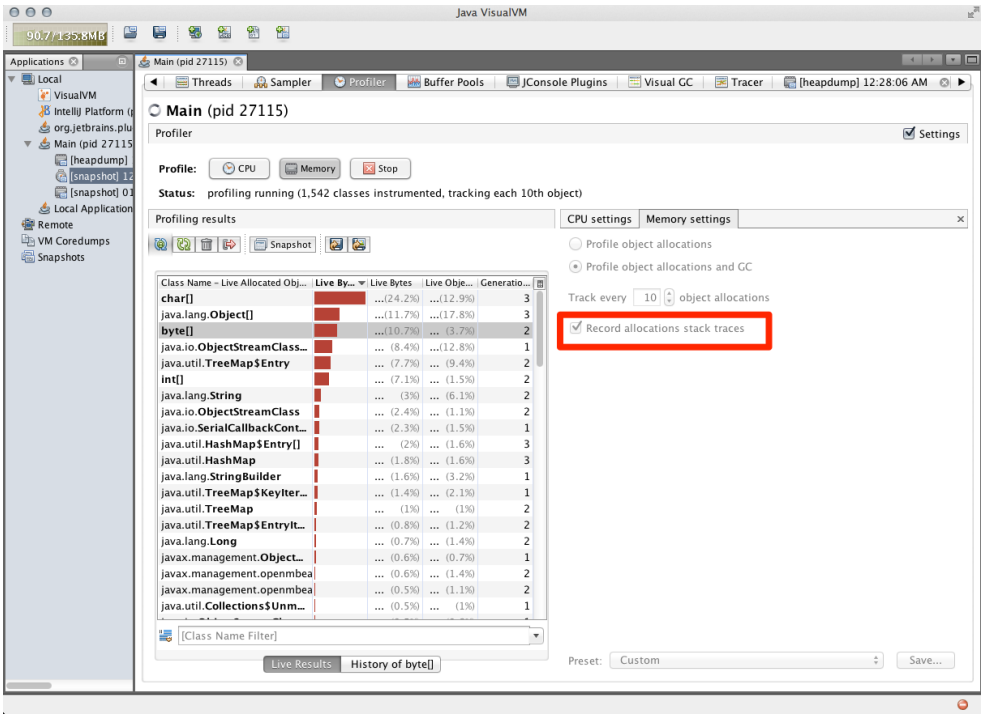
CPUプロファイルではメソッドごとの実行時間やコール回数を参照することが可能だが、あるメソッドがどのメソッドからコールされているかなどの情報を得ることができない。このような情報は、スナップショットをしようすることで可能。 プロファイルタブの「Snapshot」ボタンから取得



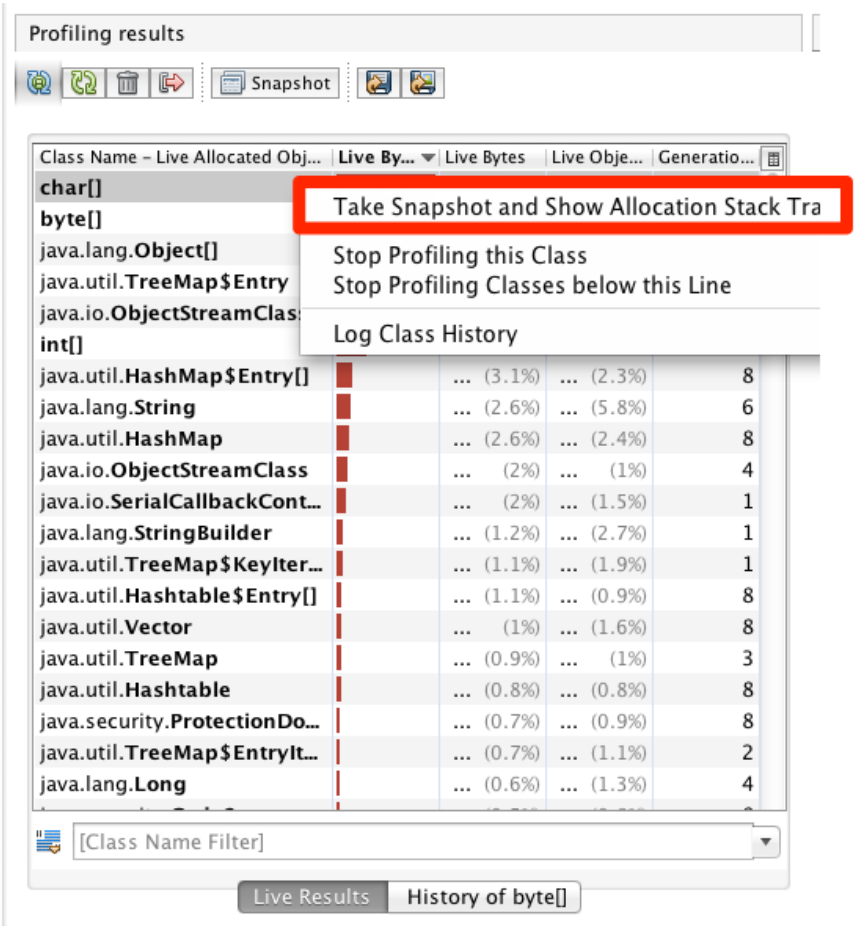

### Memory プロファイルのスナップショット

CPUプロファイルのスナップショットと同様に、設定をするとどのメソッドでオブジェクトが生成されたかを確認することが可能。

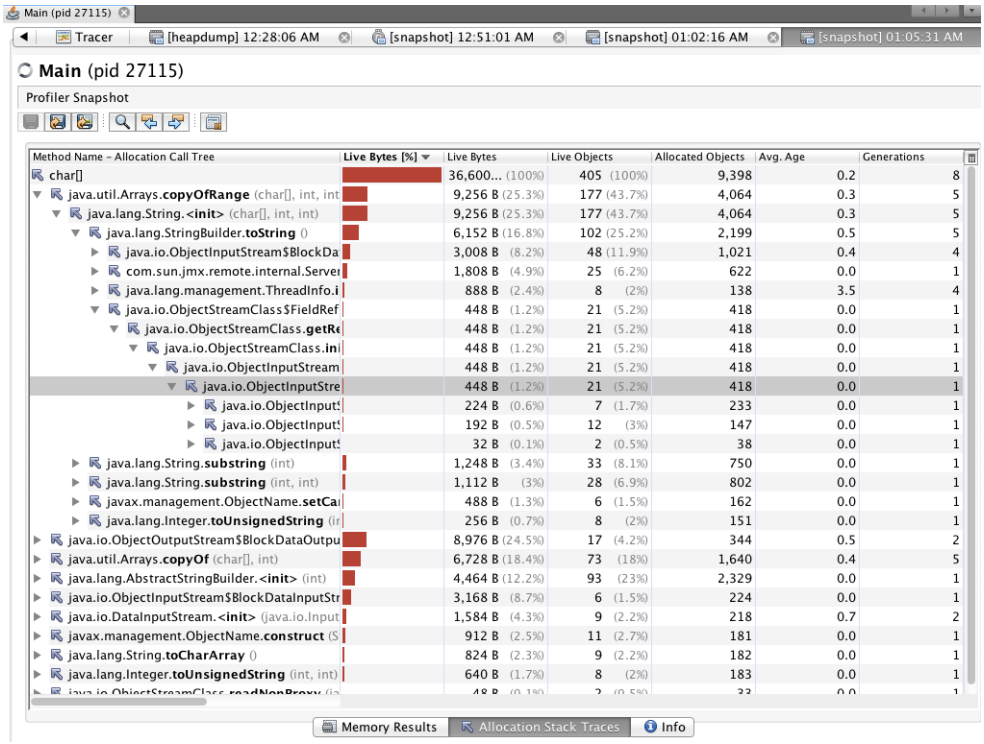
- 設定: 「Record allocation stack traces」にチェックをつける


Java VisualVM
Main (pid 27115)
Profiler
Profile:
CPU
Memory
Stop
Status: profiling running (1,542 classes instrumented, tracking each 10th object)
Profiling results
Snapshot
CPU settings
Memory settings
Profile object allocations
Profile object allocations and GC
Track every 10 object allocations
Record allocations stack traces
[Class Name Filter]
Live Results
History of byte[]
Preset: Custom
Save...

- 表示方法


Profiling results
Snapshot
Take Snapshot and Show Allocation Stack Tra
Stop Profiling this Class
Stop Profiling Classes below this Line
Log Class History
[Class Name Filter]
Live Results
History of byte[]

- call-tree が見れる




## チューニングする

### 実装に依存するもの

- 無駄にインスタンスを生成していないか。（メモリ使用量からわかる）
- 現実的ではないアルゴリズムで実装していないか (CPU 使用率、メソッドの呼び出し回数などからわかる)
    - 一般的に業務では O(nlogn)以下のオーダのアルゴリズムを使用する。

- 使用している言語上、やってはいけないことをやっていないか。
- ログ出力などで異常に時間がかかってないか。

### その他ミドル

- DBの設定
- SQLの実行計画

### HW

- Disk書き込み
- N/W
- etc

# まとめ

- どんなところにボトルネックが発生するか認識しておく
- アルゴリズムを知る
- 使用している言語を知る
- 推測するな、計測せよ
- ツールの使い方、データの見方を知る

# 参考資料など

## Java

- http://www.slideshare.net/tattyamm/visualvm
- http://visualvm.java.net/ja/gettingstarted.html
- http://docs.oracle.com/javase/jp/7/technotes/guides/visualvm/intro.html
- http://visualvm.java.net/