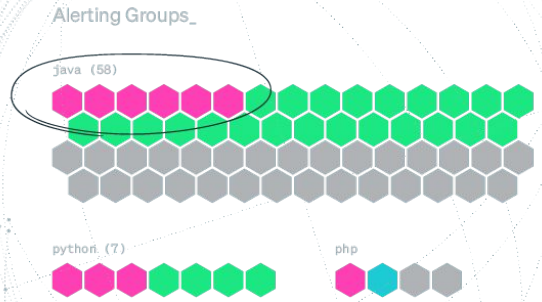
new relic

New Relic 中の人が教える Java 入門編

Kenya Takagi
Solutions Consultant



New Relic
技術統括コンサルティング部
ソリューションコンサルタント

高木 憲弥

兵庫県尼崎市出身　広島県在住

経歴:

・日系SIer

金融顧客にて要件定義・設計・開発・運用

フロント・バックエンド開発とインフラを担当

・Java関連 技術領域

Java7　Java11(OpenJDK 11) Spring Framework 4
トラブルシューティングが得意
jstat叩いて調査、heapdump分析

なぜ中の人が解説？

元Javaエンジニア　かつ　中の人　の目線で解説します

・現場でこんな問題あるよね、に対し

・**New Relic**でこうやってアプローチできる

Java固有の問題　×　New Relic

new relic.

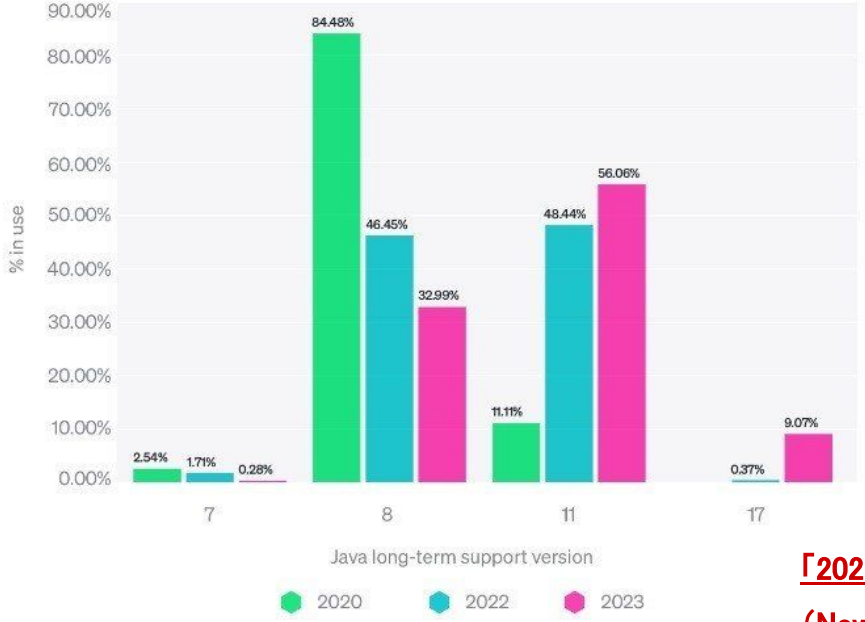
なぜJava？

・New RelicのAPMの中でも、Javaは導入APM数No.1

・エンタープライズ企業をはじめ、幅広い企業で採用されており
　大規模システムにおける導入率が高い

（余談）

New Relicで最も使われているバージョンはJava 11



「2023 State of the Java Ecosystem」（New Relic）

new relic

Agenda

01	Java固有のトラブル
02	現場で起きたJavaのトラブル例で実践New Relic
03	New Relic活用のメリット

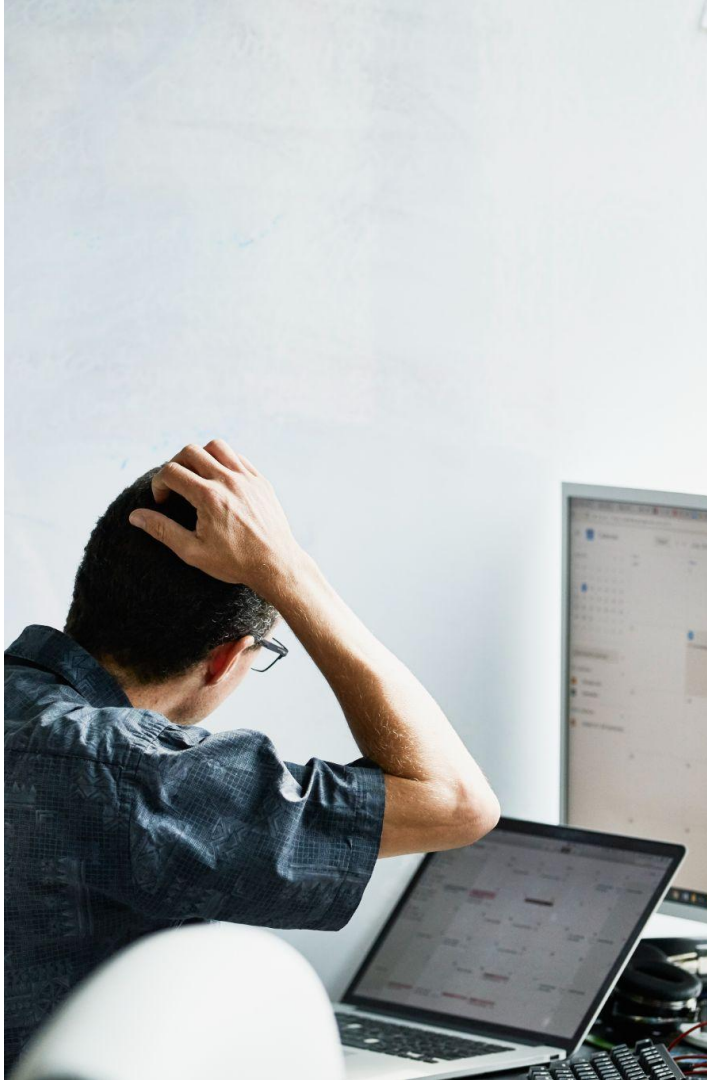
対象者とゴール

対象者

- Java APMを導入したものの、もっと活用を進めてみたい方

ゴール

- Java仮想マシン（JVM）の問題かどうか切り分けができるようになる



01. Java固有のトラブル

Java固有のトラブル

Javaはメモリを「いい感じ」に管理してくれる

- ガベージコレクション（GC)
- メモリが細分化されている
  - Heap、Metaspace、C-Heap
    - New（Eden、Survivor）
    - OLD

→ 任せっきりでもある程度使えてしまう反面・・・

Java固有のトラブル

・いつGCが発生したのか見えない

・GCの頻発によるCPUへの負荷

・Out Of Memoryがどこの領域が原因か見えない

・大量のスレッド生成によるC-Heap溢れ

→ 見えない部分（JVMの領域）を可視化して
他の事象と関連付けて判断できるようにすることが重要

02. 現場で起きたJavaのトラブル例で実践New Relic

new relic.

現場でこんな問題ありました

1. 不定期にCPU使用率が急騰。ユーザーから遅いと問い合わせ

2. 新機能リリース後に、Out Of Memoryが発生

3. メモリリークが発生して、定期的に再起動

現場でこんな問題ありました

1. 不定期にCPU使用率が急騰。ユーザーから遅いと問い合わせ

2. 新機能リリース後に、Out Of Memoryが発生

3. メモリリークが発生して、定期的に再起動

JVMの分析（JVMs)

まずは事象の切り分け

→ New Relicで全体を俯瞰して調査し、切り分けを行う

- アプリのスループット、処理時間
- JVMのGC CPU時間、各メモリ状況
- Infraのリソース状況

→ GCが発生していれば

パフォーマンスと相関がないかを確認

現場でこんな問題ありました

1. 不定期にCPU使用率が急騰。ユーザーから遅いと問い合わせ

2. 新機能リリース後に、Out Of Memoryが発生

3. メモリリークが発生して、定期的に再起動

JVMの分析（JVMs)

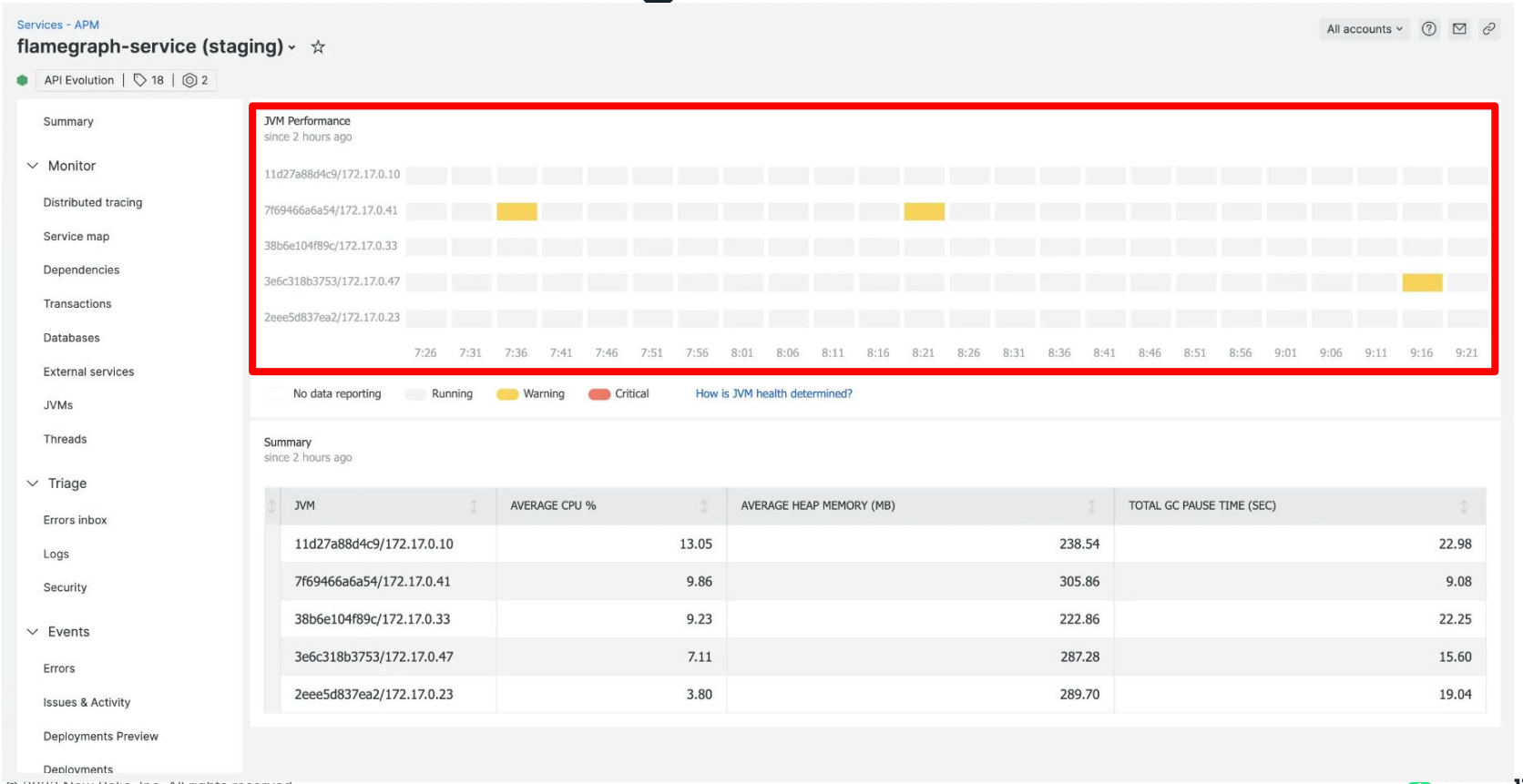
JVMsで原因の切り分け

・heapか、metaspaceか、c-heapか

・GCに時間がかかりすぎているのか

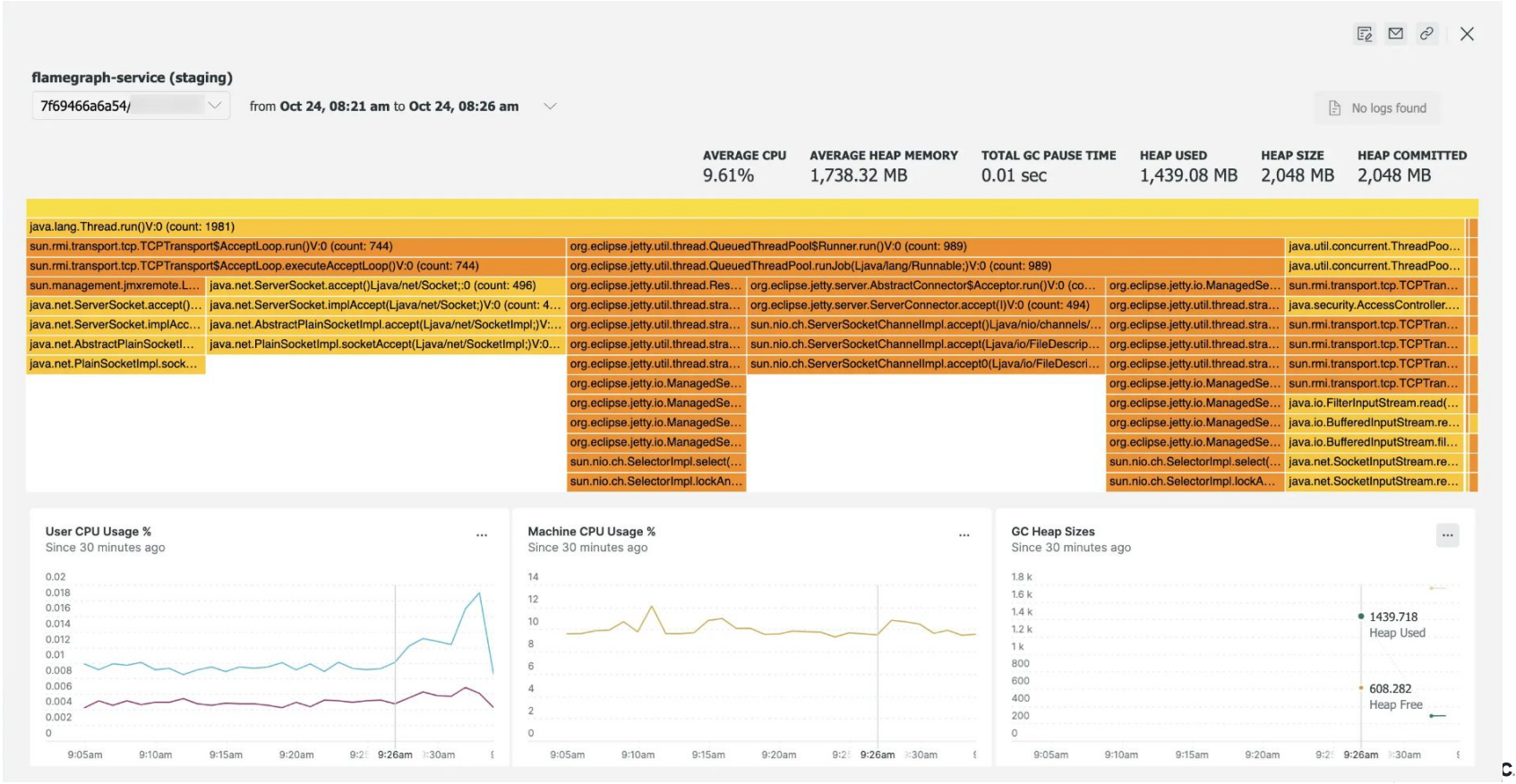
などの要因から切り分けしていく

JVMの分析（Java Flight Recorder：JFR）



new relic

JVMの分析（FlameGraph）



JVMの分析

JVMsとJFRの使い分け

普段：JVMsでパフォーマンス分析

問題発生時：JFRを有効化してトラブルシューティング
（オーバーヘッドあり）

現場でこんな問題ありました

1. 不定期にCPU使用率が急騰。ユーザーから遅いと問い合わせ

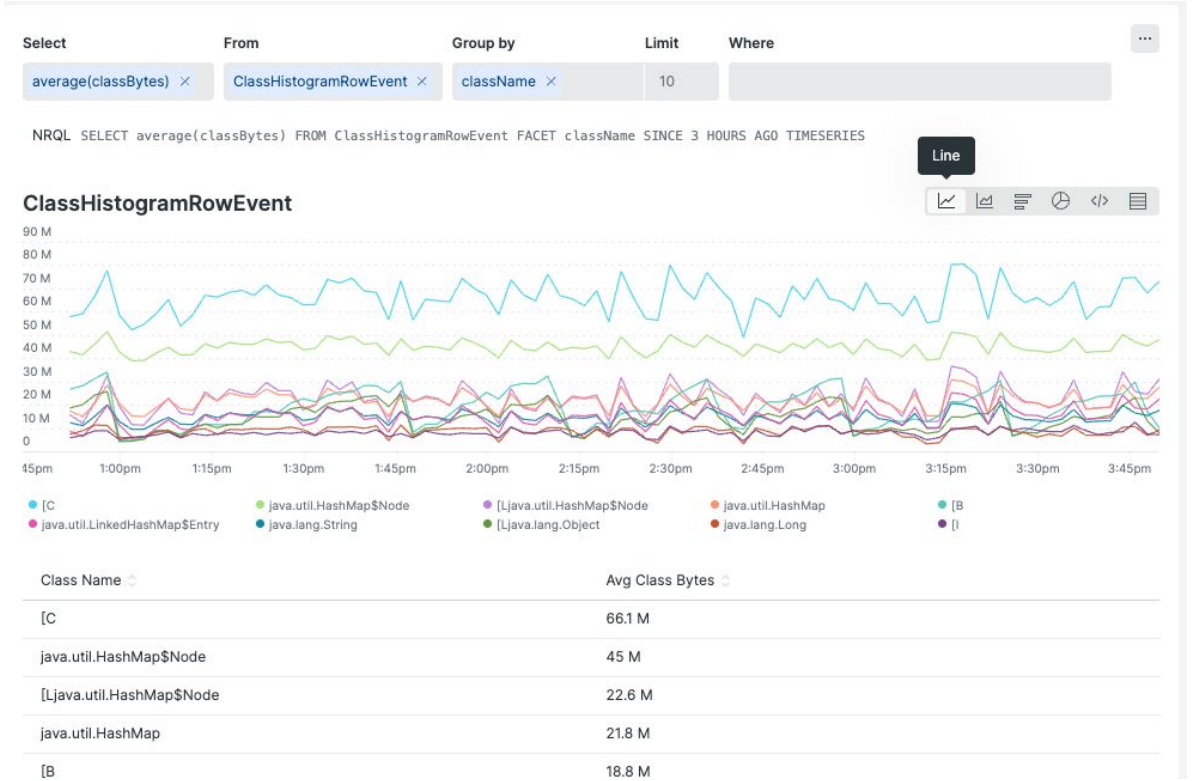
2. 新機能リリース後に、Out Of Memoryが発生

3. メモリリークが発生して、定期的に再起動

Class Histogram

オブジェクトとメモリ量の
傾向がわかるので
原因の特定に繋がる

しかし・・・



new relic

Class Histogram

メモリリーク調査は難易度が高く

Class Histogramだけでは問題の特定に至らぬ場合も多い

（外部ライブラリの問題や相性による問題など）

→ **HeapDump吐いて**

外部ツールも使って詳細にプロファイリングするなど

更なる調査も必要となる

現場でこんな問題ありました

1. 不定期にCPU使用率が急騰。ユーザーから遅いと問い合わせ

   →JVMsでGCとパフォーマンスの相関を確認

2. 新機能リリース後に、Out Of Memoryが発生

   →JVMs、JFRで原因調査

3. メモリリークが発生して、定期的に再起動

   →Class Histogramで原因調査

04. New Relic活用のメリット

new relic

New Relic活用のメリット

・Java APMの導入だけで、簡単にJVMの可視化が可能

・従来のJVM調査ではできなかった
　各レイヤーを時系列揃えて一気通貫で見ることができる
　　（OS・アプリ・フロント・JVMs）

New Relic活用のメリット

従来のJVM調査は、職人の勘所を頼りに順番に切り分け

→ 他原因との相関が判断しづらく、時間もかかる

New RelicのJVM調査は、全体を一気通貫に見て素早く特定

→ 素早い次の一手を打てるようになる

→ メモリ増強する際にも New Relicを共通言語として活用できる

Thank you.

Kenya Takagi