Workshop async-profiler

Victor Gallet MiXiT 2025

Disclaimers

Victor Gallet - Tech Lead chez Bpifrance

- Je ne suis pas un expert de la JVM !
- Je ne suis pas un expert de async-profiler

Objectifs de ce workshop

- utiliser async-profiler
- Utiliser le meilleur outil de performance testing : k6
- Apprenez à profiler votre application Java
- Apprenez à lire et interprétez un Flamegraph
- N'ayez plus peur des problèmes de performance!

Resources

- async-profiler
- jvmperf
- Coloring Flame Graphs: Code Hues by Brendan Gregg
- A Guide to async-profiler by Anshul Bansal and Eric Martin
- 🎬 USENIX ATC '17: Visualizing Performance with Flame Graphs by Brendan Gregg
- 🎬 Taming performance issues into the wild: a practical guide to JVM profiling by Francesco Nigro, Mario Fusco
- [Java][Profiling] Async-profiler - manual by use cases by Krzysztof Ślusarski
- [Java][Profiling][Memory leak] Finding heap memory leaks with Async-profiler by Krzysztof Ślusarski
- Java Safepoint and Async Profiling by Seetha Wenner
- 🇫🇷 Traquer une fuite mémoire : cas d'étude avec Hibernate 5, ne tombez pas dans le IN ! by Ling-Chun SO
- 🎬 🇫🇷 Sous le capot d'une application JVM - Java Flight Recorder / Java Mission Control
- by Guillaume Darmont
- 🇫🇷 Performance et diagnostic - Méthodologie et outils by Vladislav Pernin

Avant de démarrer

Récupérez le projet ici :

https://gitlab.com/victor.gallet/was

async-profiler késako ?

- low overhead sampling profiler
- se base sur une API non standard de Hotspot JVM (issue AsyncGetCallTrace replacement)
- ne souffre pas du Safepoint biais problem
- permet de profiler plusieurs types d'événements :
  - CPU
  - Memory Allocation
  - Wall
  - Lock
  - Java Method Profiling

Utiliser async-profiler

- En process standalone

  > asprof -d <duration> -e <event> -f /tmp/flamegraph.html <pid>

- En tant qu'agent

  > java -agentpath:/path/to/libasyncProfiler.so=start,event=cpu,file=profile.html ...

Utiliser async-profiler

- Dans un container

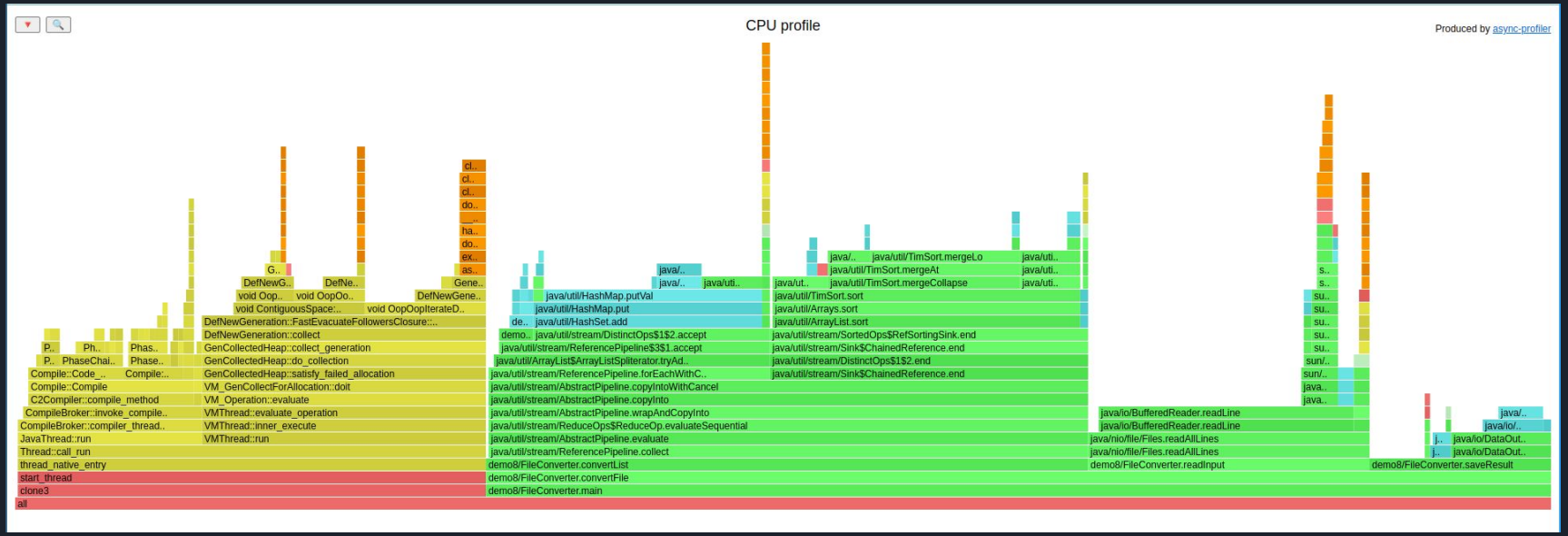
  > wget https://github.com/async-profiler/async-profiler/releases/download/v3.0/async-profiler-3.0-linux-x64.tar.gz

  > tar -xzf async-profiler-3.0-linux-x64.tar.gz

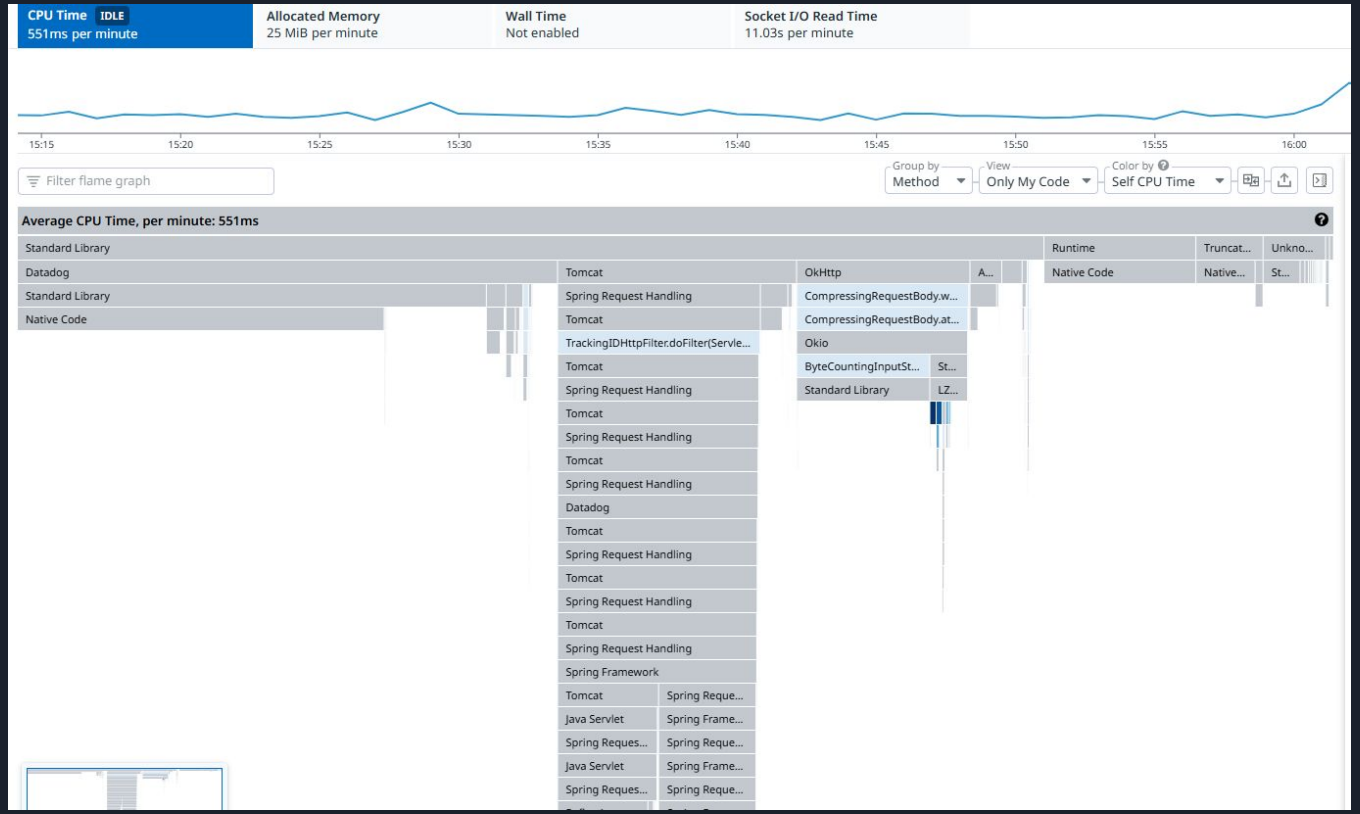
  > asprof -d <duration> -e <event> -f /tmp/flamegraph.html <pid>

https://github.com/async-profiler/async-profiler/blob/master/docs/ProfilingInContainer.md

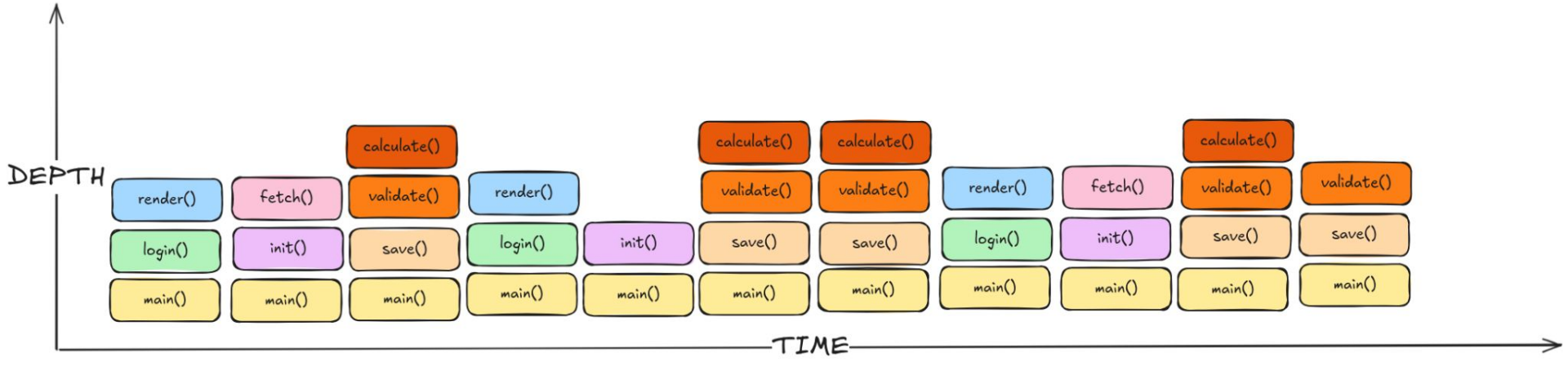
Interpréter un Flamegraph



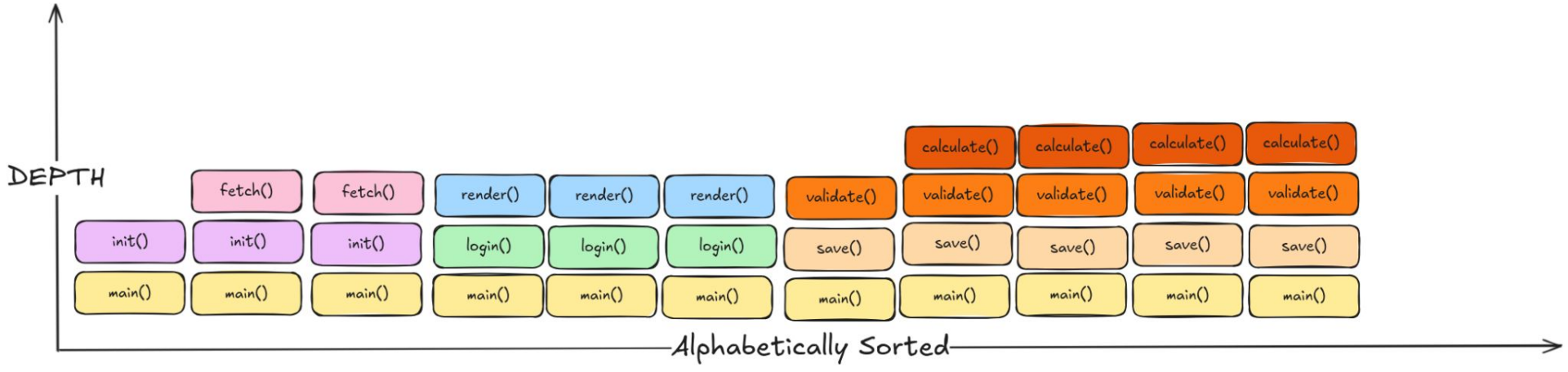
Interpréter un Flamegraph



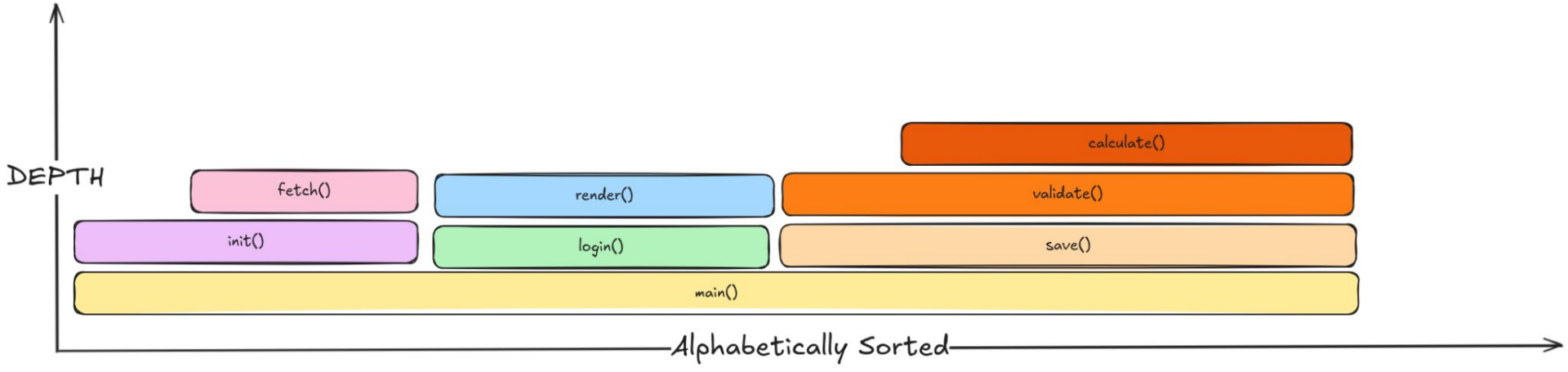
Construction d’un Flamegraph



Tri par ordre alphabétique



Agrégation



la méthode calculate() est un bon candidat à l'optimisation car elle est plus "large" sur l'axe du temps et de la profondeur dans la stack.

Les méthodes render() et fetch() peuvent être des candidats de second choix.

Code Couleur d’un Flamegraph

Kernel	__x64_sys_futex
	do_syscall_64
	entry_SYSCALL_64_after_hwframe
Native	__lll_lock_wake
	pthread_mutex_unlock
C++	SharedRuntime::handle_ic_miss_helper_internal
	SharedRuntime::handle_ic_miss_helper
	SharedRuntime::handle_wrong_method_ic_miss
Native	ic_miss_stub
C1 compiled	java/util/ComparableTimSort.countRunAndMakeAscending
Interpreted	java/util/ComparableTimSort.sort
	java/util/Arrays.sort
Java inlined	java/util/Arrays.sort
Java compiled	java/util/ArrayList.sort
Match found	java/util/Collections.sort
	nonapi/io/github/classgraph/utils/CollectionUtils.sortIfNotEmpty

- System (User native)
- Java
- C++

Durant ce workshop ?

- Nous allons profilez une application Java avec des exemples plus ou moins factices
- Merci de vous arrêter à chaque étape pour que nous prenions le temps d'expliquer
- Bien utiliser k6 avant chaque session de profiling !

Feedback