SRE 讀書會
Linux 性能優化實戰

58 | 答疑(六) : 容器冷啟動如何性能分析？

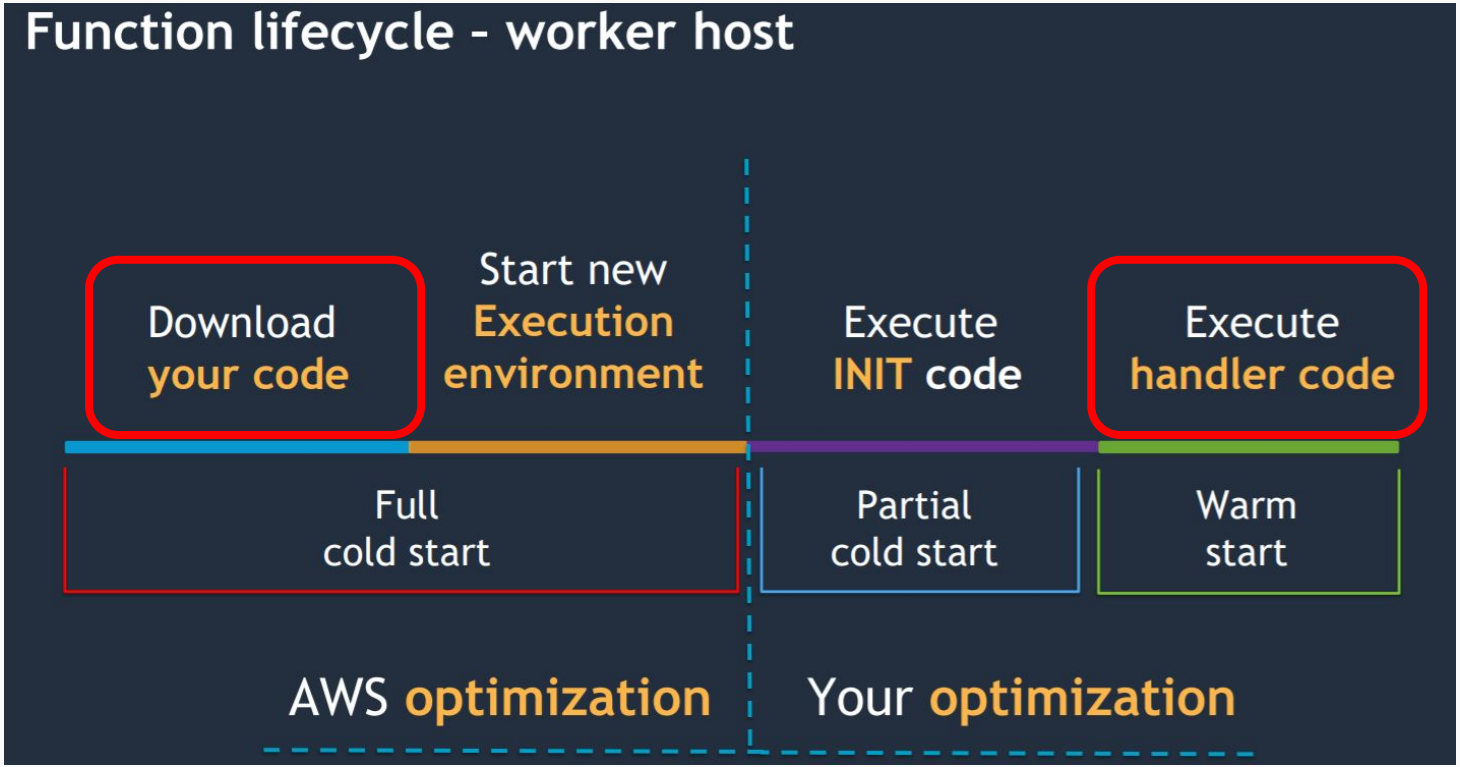
2020/12/24 Hans Hsu

問題 1: 容器冷啟動性能分析

問題 1：容器冷啟動性能分析

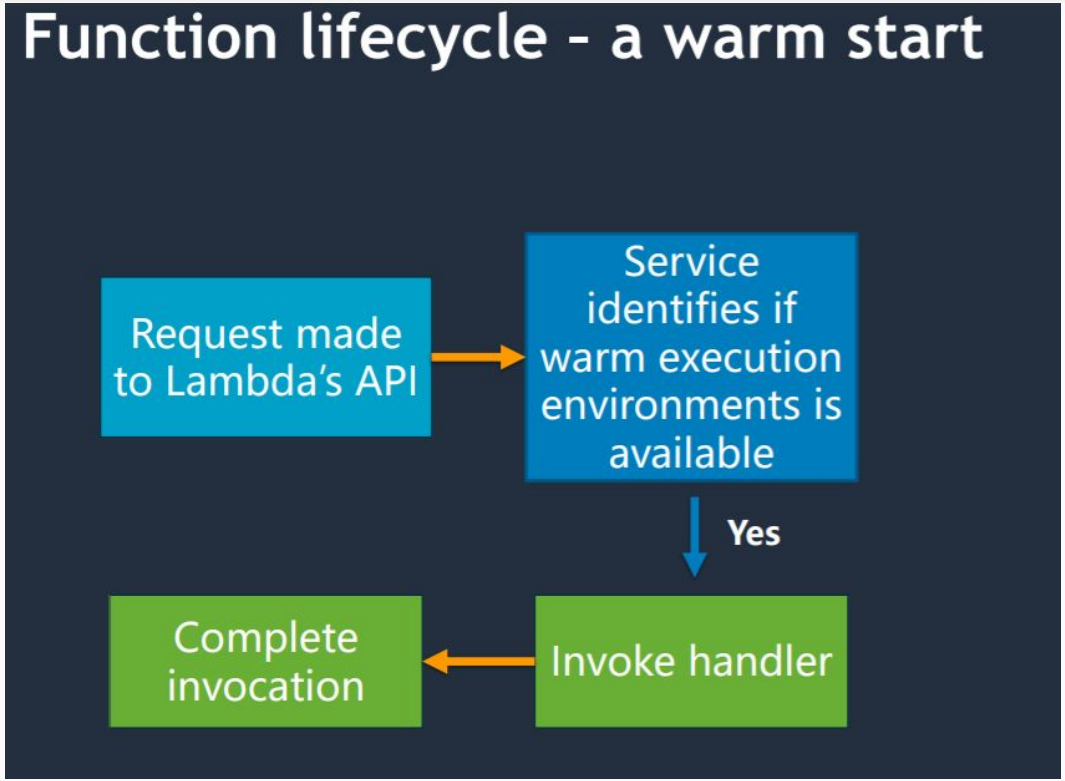
- 事件觸發
- 資源調度
- 鏡像拉取
- 網絡配置
- 啟動應用

以 AWS Lambda 為例



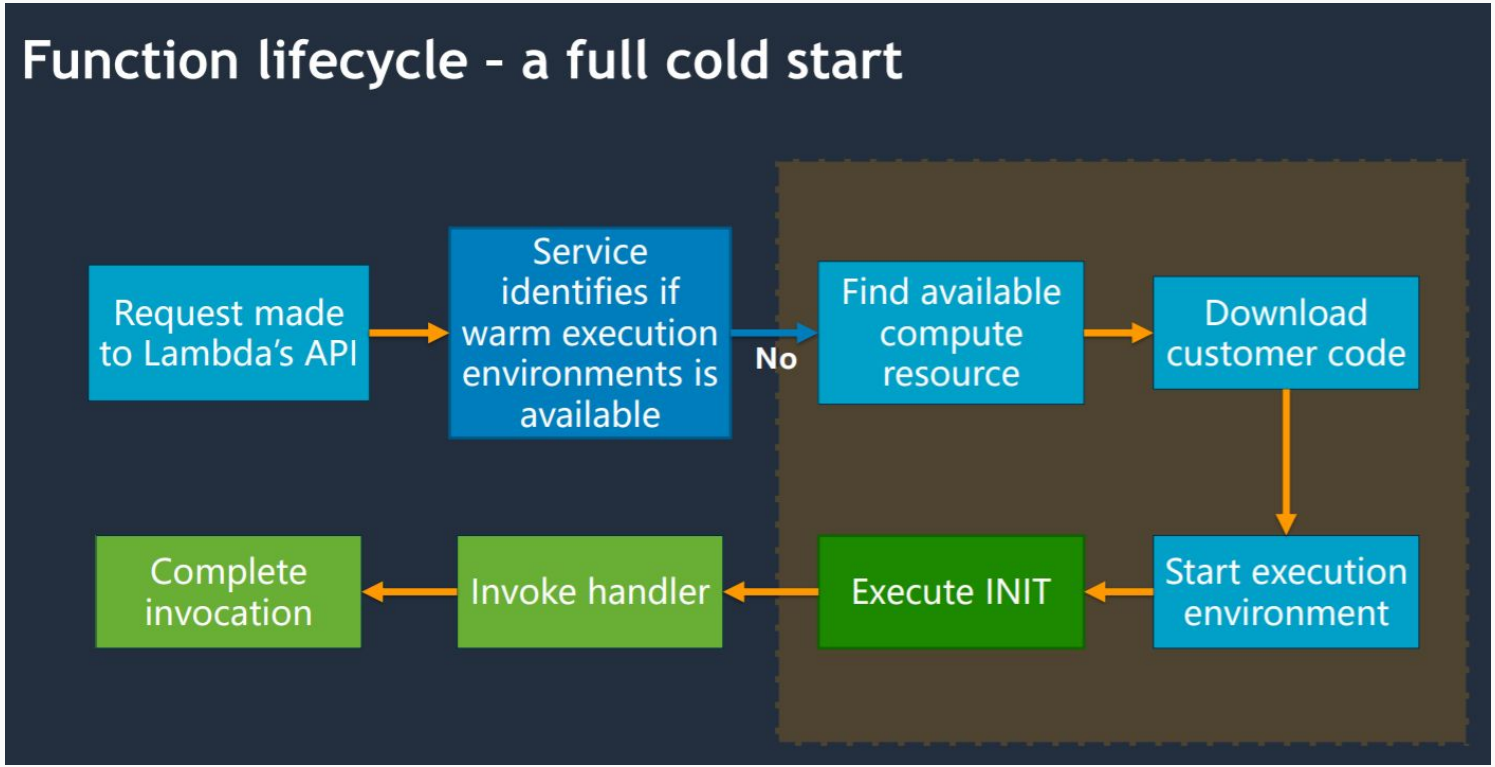
source: [Optimizing Lambda performance for your serverless applications]

以 AWS Lambda 為例 - Warm start



source: [Optimizing Lambda performance for your serverless applications]

以 AWS Lambda 為例 - Full cold start



source: [Optimizing Lambda performance for your serverless applications]

以 AWS Lambda 為例 - 如何調整程式碼加快 Lambda 執行速度 (Node.js)

Optimized dependency usage (Node.js SDK & X-Ray)

```
// const AWS = require('aws-sdk’)
const DynamoDB = require('aws-sdk/clients/dynamodb’) // 125ms faster

// const AWSXRay = require('aws-xray-sdk’)
const AWSXRay = require('aws-xray-sdk-core’)  // 5ms faster

// const AWS = AWSXRay.captureAWS(require('aws-sdk’))
const dynamodb = new DynamoDB.DocumentClient()
AWSXRay.captureAWSClient(dynamodb.service)  // 140ms faster
```

source: [Optimizing Lambda performance for your serverless applications]

以 AWS Lambda 為例 - 如何調整程式碼加快 Lambda 執行速度 (Python)

Lazy initialization example (Python & boto3)

```
import boto3

S3_client = None
ddb_client = None

def get_objects(event, context):
    if not s3_client:
        s3_client = boto3.client("s3")
    # business logic

def get_items(event, context):
    if not ddb_client:
        ddb_client = boto3.client("dynamodb")
    # business logic
```

source: Optimizing Lambda performance for your serverless applications

以 AWS Lambda 為例 - Cold Start

Cold starts - Execution environment

The facts:

- <1% of production workloads
- Varies from <100ms to >1s

Be aware...

- You cannot target warm environments
- Pinging functions to keep them warm is limited

Cold starts occur when...

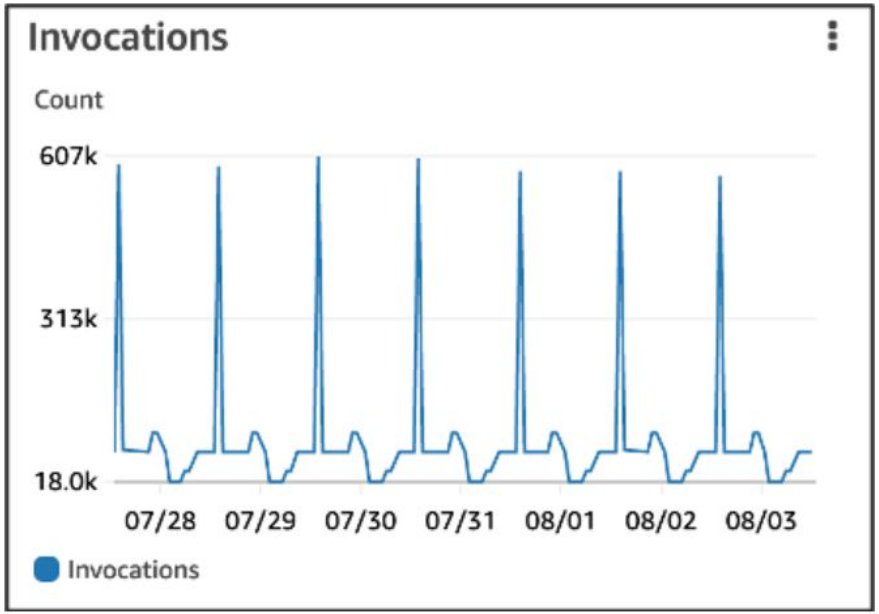
- Environment is reaped
- Failure in underlying resources
- Rebalancing across Azs
- Updating code/config flushes
- Scaling up

source: Optimizing Lambda performance for your serverless applications

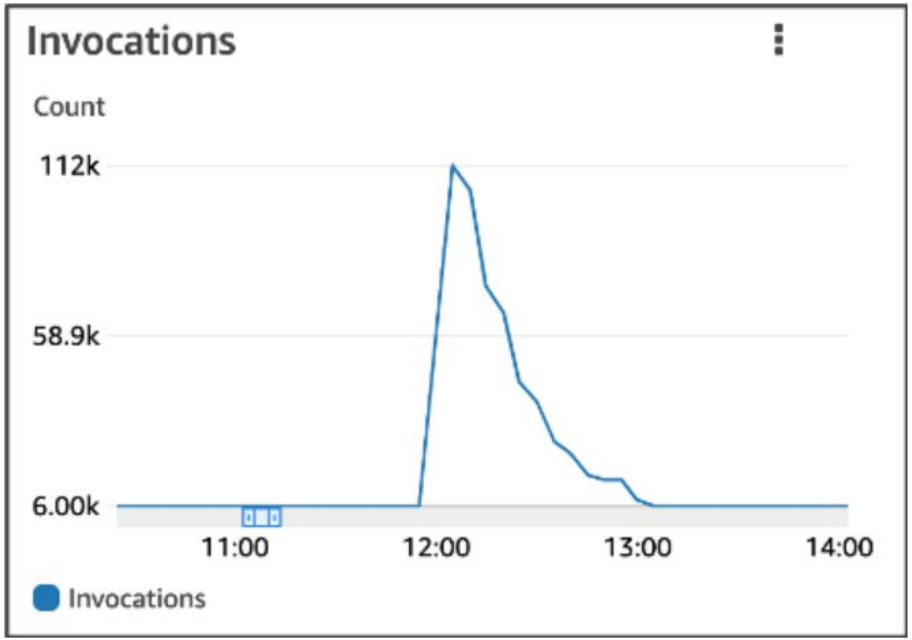
延伸討論容器冷啟動如何解決

淺談一下對應 AWS Lambda cold start 的解決方案

應用場景 - 每天中午開始促銷活動



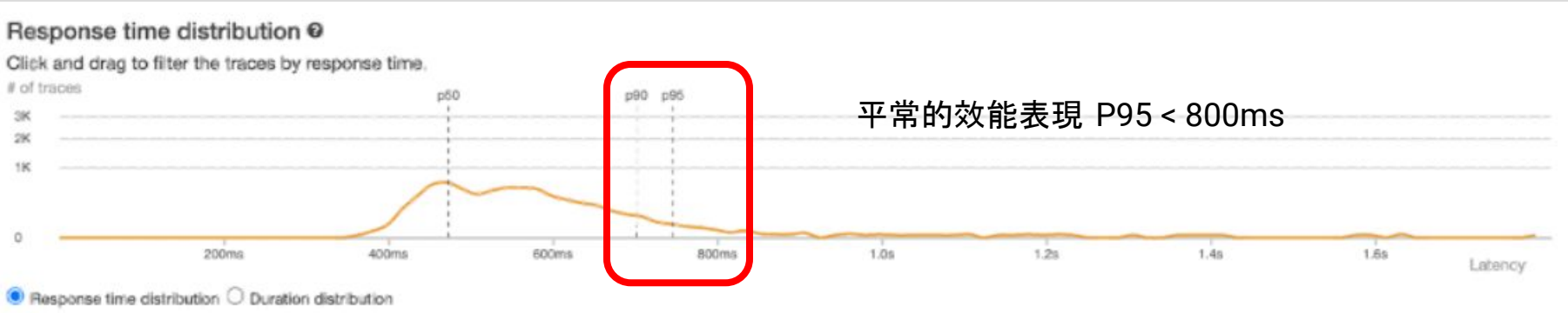
Reoccurring invocations



Peak invocations

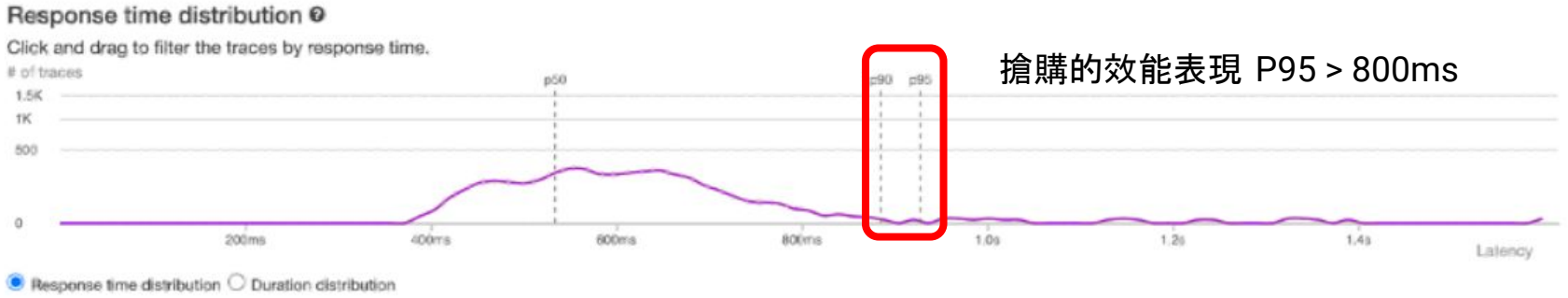
source: [Optimizing Lambda performance for your serverless applications]

統計數據



平常的效能表現 P95 < 800ms

Normal performance distribution

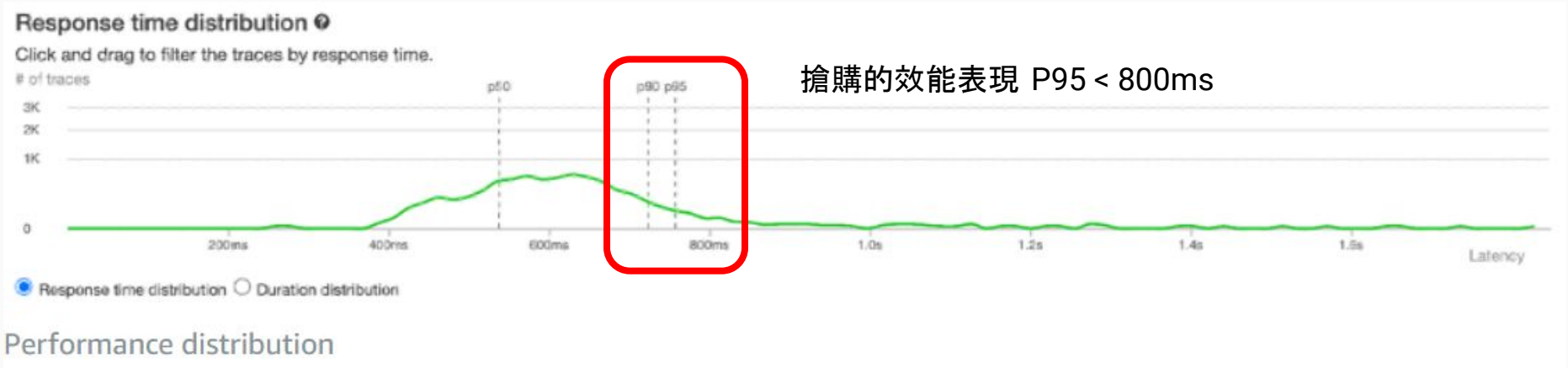


搶購的效能表現 P95 > 800ms

Peak performance distribution

source: [Optimizing Lambda performance for your serverless applications]

利用 Application Auto Scaling 排程增加 Lambda 的 Provisioned Concurrency 數量



搶購的效能表現 P95 < 800ms

source: [Optimizing Lambda performance for your serverless applications]

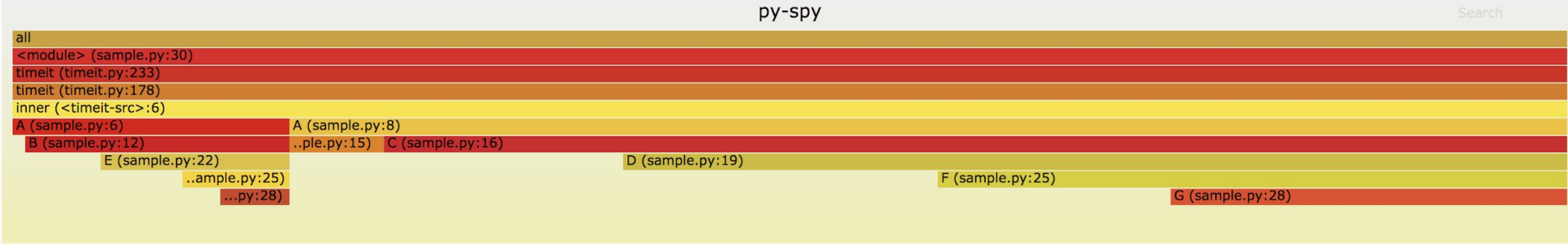
問題 2：CPU 火焰圖和內存火焰圖有什麼不同？

問題 2：CPU 火焰圖和記憶體火焰圖有什麼不同？

- 對 CPU 火焰圖來說，採集的數據主要是消耗 CPU 的函數
- 而對記憶體火焰圖來說，採集的數據主要是內存分配、釋放、換頁等記憶體管理函數

Python 範例 (把東西都 Dump 出來)

```
sudo py-spy record -o profile.svg -- python sample.py
sudo py-spy top -- python sample.py
```

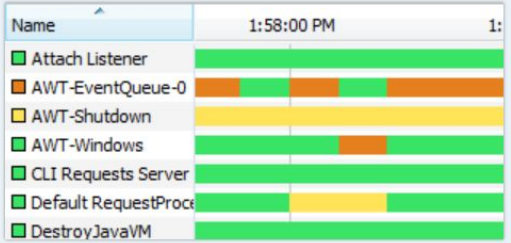


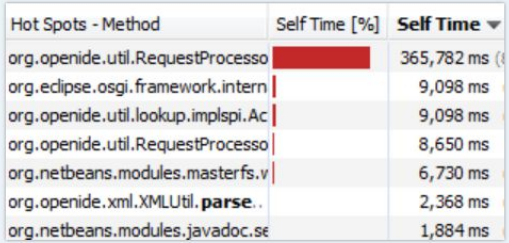
問題 3：perf probe 失敗怎麼辦？

閒聊一下之前開發的時候用什麼工具追蹤

Visualize Process Threads

All threads running in a Java process are displayed in a timeline together with aggregated Running, Sleeping, Wait, Park and Monitor times.





Profile Performance And Memory Usage

VisualVM provides basic profiling capabilities for analyzing application performance and memory management. Both sampling and instrumentation profilers are available.

source: https://visualvm.github.io/plugins.html

問題 4：RED 法監控微服務應用

問題 4：RED 法監控微服務應用

- USE 法
  - 使用率
  - 飽和度
  - 錯誤數
- RED 法
  - 請求數（Rate）
  - 錯誤數（Errors）
  - 響應時間（Duration）
- USE 方法適用於系統資源的監控
- RED 方法適用於微服務應用的監控

淺談一下監控定義的指標

- USE 法更接進工程端 vs RED 法更接近業務端
  - CPU 使用率 100% vs 使用者: 我覺得很慢 (延遲變高)
  - 記憶體 OOM vs 使用者: 我沒辦法用購物車下單 (HTTP Error)

問題 5:深入內核的方法

問題 5:深入內核的方法

- 你怎麼不直接去看 Linux 的 code(誤

BPF Compiler Collection (BCC) 覺得有用的工具 - 找 TCP 連線

./tcpconnect

PID	COMM	IP	SADDR	DADDR	DPORT
1479	telnet	4	127.0.0.1	127.0.0.1	23
1469	curl	4	10.201.219.236	54.245.105.25	80
1469	curl	4	10.201.219.236	54.67.101.145	80
1991	telnet	6	::1	::1	23

BPF Compiler Collection (BCC) 覺得有用的工具 - 找 OOM Kill

./oomkill

Tracing oom_kill_process()... Ctrl-C to end.

21:03:39 Triggered by PID 3297 ("ntpd"), OOM kill of PID 22516 ("perl"), 3850642 pages, loadavg: 0.99 0.39 0.30 3/282 22724

21:03:48 Triggered by PID 22517 ("perl"), OOM kill of PID 22517 ("perl"), 3850642 pages, loadavg: 0.99 0.41 0.30 2/282 22932

Linux 性能優化實戰 58 問題與討論

- 各位怎麼解決 FAAS 冷啟動?
- 各位常用的程式語言都用什麼工具追蹤效能?