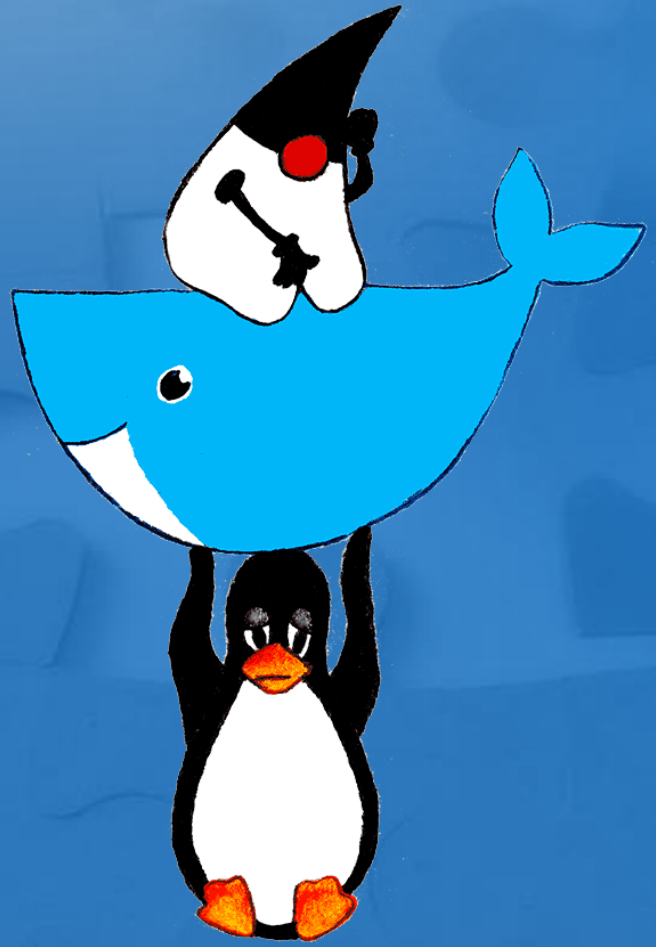


JAVA В КОНТЕЙНЕРЕ ОСОБЕННОСТИ ЭКСПЛУАТАЦИИ

Алексей Рагозин

Кто есть кто в виртуализации?

Гипервизор – среда выполнения машинного кода в эмулируемой среде, включая эмуляцию периферийных устройств.

Виртуальная машина – формальное описание взаимодействия прикладной программы со средой выполнения.

Что же такое Lunix контейнер?

docker, runc, podman, crun, lxc, …

Мечта всех времён

Write

Once

Run

Anywhere

Мечта всех времён

~~Write~~ Build

Once

Run

~~Anywhere~~

On any Linux

(with same arch and kernel version)

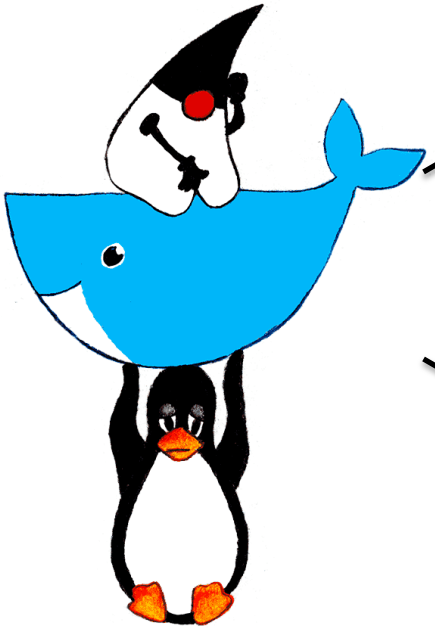
JVM в контейнере

Управлению памятью (heap)
Много-поточность
Кросс-платформенный API

JVM

Ядро Linux

Управлению памятью (virtual memory)
Вытесняющая многозадачность
Сеть, файлы и прочие API



Контейнеризация

cgroups – изоляция ресурсов

cgroups v1 или v2?

namespaces – изоляция сети, файловой системы и пр.

overlayFS – магия образов

В докладе

Память

ЦПУ

Файловая система

Диагностика JVM в условиях контейнеризации

Память

Память Linux

- “Плоское” адресное пространство процесса
- Память организована в страницы (4k)

  (Есть ещё “huge” страницы, но это другая история)
- Может быть включён файл подкачки (swap)

Память Linux процесса

С точки зрения процесса, диапазон адресов может быть

- Reserved (SEGFAULT при попытке доступа) -
- Committed (доступ не приведён к ошибке)
  - Страница ещё не привязана (выделение при первом обращении)
  - ✓ Страница привязана
  - Страница в вытеснена в swap (загрузка при обращении)
  - Страница в copy on write режиме (копирование при попытки записи)
- File mapped

Память в контейнере Linux

Всё то же самое + **cgroups**

- Лимиты по памяти, CPU и другим ресурсам

Лимиты на память контейнера

- Resident memory (Включая отображаемые в память файлы)
- Resident + Swap memory

Память JVM

Куча (Heap) – основная часть

Стеки потоков – растут с числом потоков

NIO Buffers

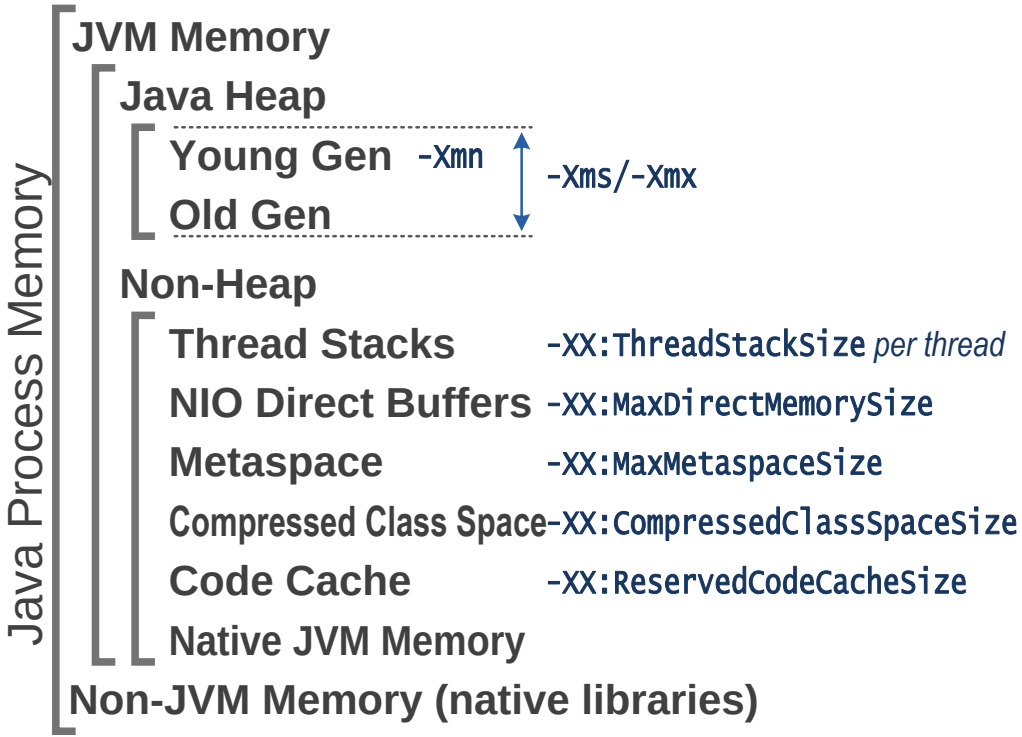
Память “нативных” библиотек

} Специфика приложения

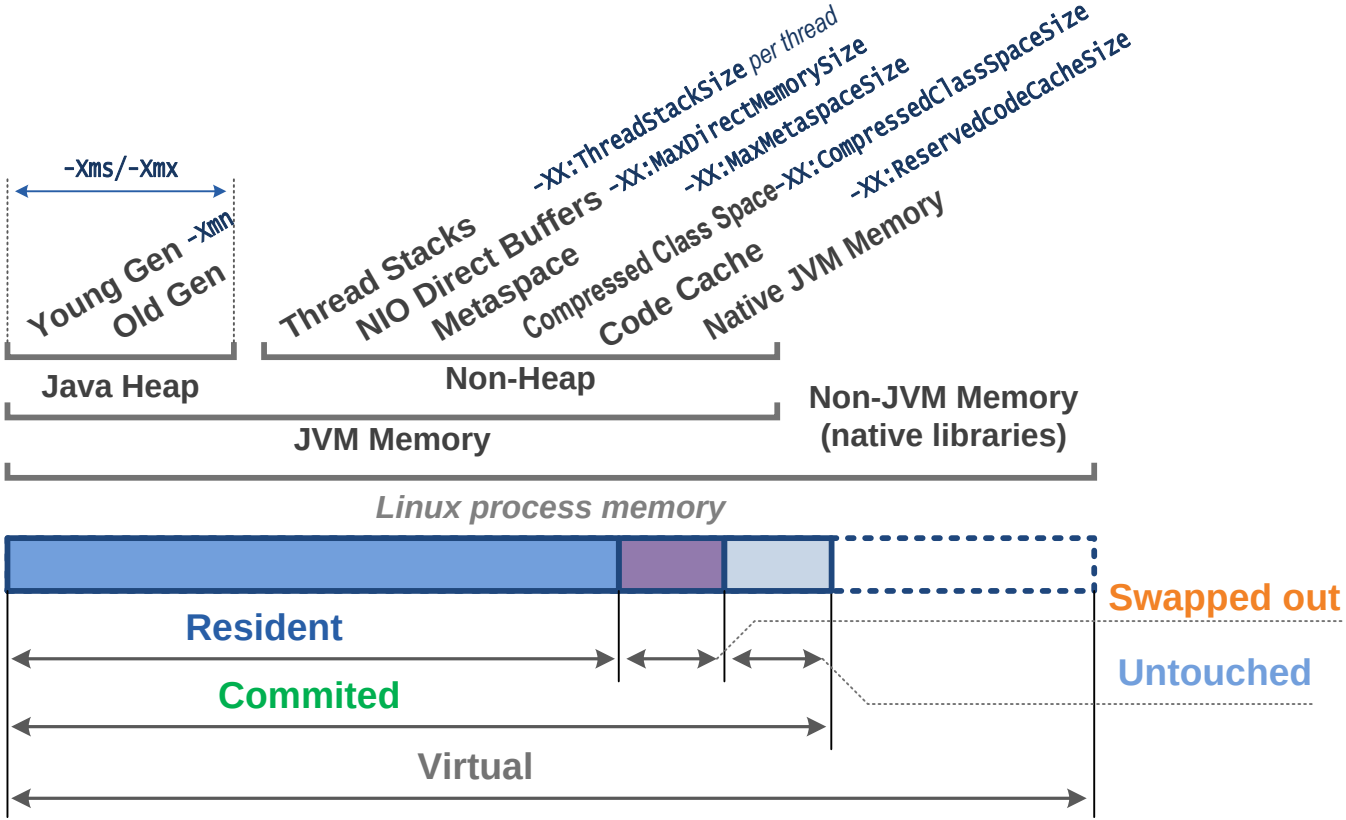
Метаданные и JIT – проблема маленьких JVM

Накладные расходы JVM

Память JVM в деталях

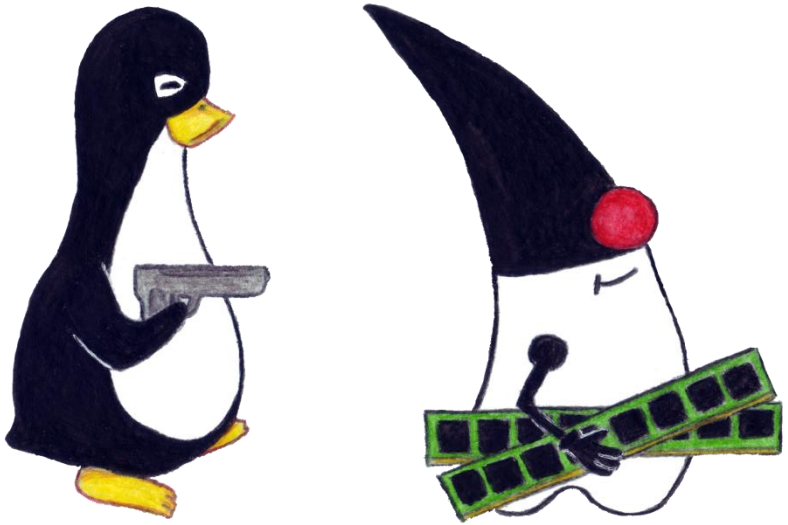


Память JVM/Linux



Out of Memory in Linux

Недостаток ресурсов памяти решается OOMKiller`ом!



JVM в контейнере

JVM детектирует контейнер и лимиты*

Размер кучи устанавливается по лимиту на память

`-XX:MaxRAMPercentage` (по-умолчанию 25%)

Максимальное число GC потоков выставляется по лимиту CPU

`-XX:ParallelGCThreads`

* - cgroups v2 поддерживаются начиная с Java 17

Опция -XshowSettings

Команда

`java -XshowSettings:system -version`

Показывает информацию о лимитах доступную JVM

```
Operating System Metrics:
    Provider: cgroupv1
    Effective CPU Count: 2
    CPU Period: 100000us
    CPU Quota: 150000us
    CPU Shares: -1
    List of Processors, 4 total:
    0 1 2 3
    List of Effective Processors, 4 total:
    0 1 2 3
    List of Memory Nodes, 1 total:
    0
    List of Available Memory Nodes, 1 total:
    0
    Memory Limit: 1.00G
       Memory Soft Limit: Unlimited
       Memory & Swap Limit: 1.50G

openjdk version "17" 2021-09-14 LTS
OpenJDK Runtime Environment (build 17+35-LTS)
OpenJDK 64-Bit Server VM (build 17+35-LTS,
mixed mode)
```

Сайзинг JVM для контейнера

Типовое Java приложение

не требует настроек.

Сайзинг JVM для контейнера

Типовое Java приложение

не требует настроек.

(Можно выставить `-XX:MaxRAMPercentage` побольше)

Нужна ли вам память вне хипа?

Использует ли ваше приложение диск?

- Да – оставляйте запас для кэша
- ✓ Нет – можно оставить настройки по-умолчанию

Swap или Resident memory

Управляйте лимитом на swap

Ставьте лимит на swap чуть больше чем лимит на память

- При недостатке памяти приложение начнёт страдать, но
- Вы увидите, что проблема с память
- Если это был временный фактор у приложения есть шанс прийти в норму
- Альтернатива – перезагрузка контейнера OMM killer

ЦПУ

ЦПУ лимиты контейнера

CPU Limit – верхняя граница использования ЦПУ

CPU Shares – вес контейнера для планировщика в условиях недостатка ЦПУ

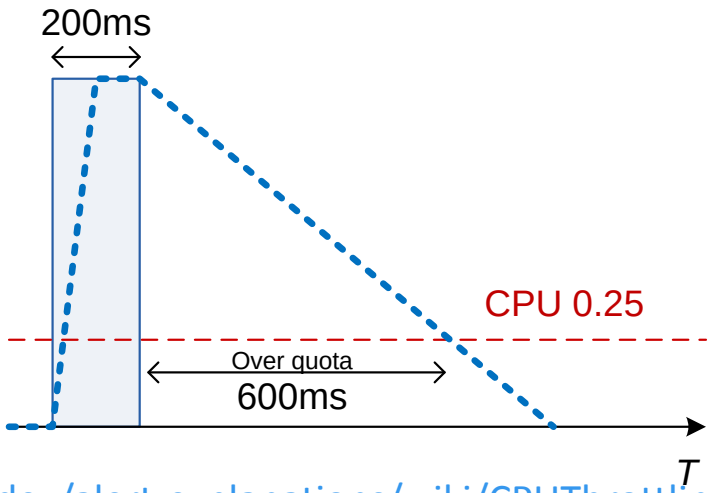
CPU Set – фиксированные CPU для выполнения

и прочее опции планировщика

- CPU Limit может быть не целым
- Лимит ЦПУ для контейнера ограничивает GC потоки –> что ведёт к увеличение GC пауз

Нюансы малых лимитов ЦПУ

При лимитах меньше 1 CPU приложение может выработать ресурсы ЦПУ в “долг” (особенно если хост не нагружен), в следствии чего контейнер будет лишён ЦПУ до тех пор, пока среднее использование ЦПУ не уложиться в квоту.



https://github.com/robusta-dev/alert-explanations/wiki/CPUThrottlingHigh-(Prometheus-Alert)

Файловая система

Контейнер и файловая система

“Linux контейнер – chroot на стероидах.”

Корневая системы контейнера – overlayfs или аналог

- Как правило есть лимит на размер
- Идеосинхрозии posix семантики (например переименование файлов)

Используйте “точки монтирования” для рабочих директорий и /tmp

- Более эффективное IO
- Совместное использование между контейнерами
- Простой доступ с хоста (например к дампам кучи)

Временные файлы JVM

`-Djava.io.tmpdir=`... - позволяет использовать альтернативный путь для временных файлов

`-XX:PerfDataSaveFile=`... - альтернативный путь для hsperfdata файла

Диагностика JVM

Работа с локальным контейнером

Типовые операции доступны из коробки

- Дамп потоков – `jcmd <pid> Thread.print`
- Дамп памяти – `jcmd <pid> GC.heap_dump <dumpfile>` *
- Профилирование – JDK Flight Recorder с использованием `jcmd` *

Графические инструменты Visual VM, Mission Control и т.п.
требуют JMX для полноценной работы!

* - Путь к дампу указывается в файловой системе контейнера!

Особенности JMX

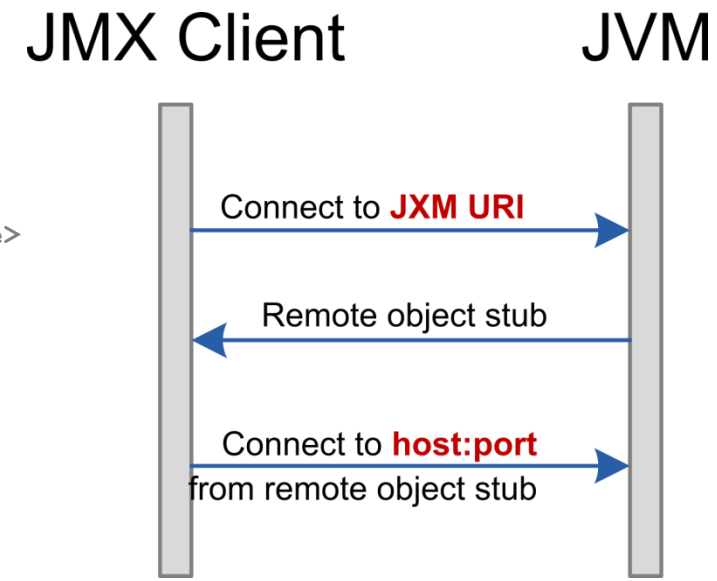
JVM стандартный протокол мониторинга/диагностики JVM.

Включается опциями

```
-Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.authenticate=<true/false>
-Dcom.sun.management.jmxremote.ssl=<true/false>
-Dcom.sun.management.jmxremote.port=<port>
-Dcom.sun.management.jmxremote.rmi.port=<port>
-Djava.rmi.server.hostname=<hostname>
```

Порт должен быть проброшен из контейнера

Адрес RMI хоста должен совпадать с внешним адресом контейнера



Что делать с JMX?

Стратегии настройки JMX

Настройка JMX на публичный IP
необходимо знать IP

Настройка IP на туннельный доступ
необходимо уметь доступ к туннелю
прямой доступ к JMX работать не будет

Радикальные решения

Jolokia – JVM over HTTP
https://jolokia.org/

SJK JMX proxy
https://github.com/aragozin/jvm-tools/
(позволяет включать JMX без рестарта, через консоль контейнера)

https://blog.ragozin.info/2023/09/curse-of-jmx.html

Диагностика JMX

SJK – https://github.com/aragozin/jvm-tools

```
sjk.jar mxping -s <hostname:port> …
```

Детальная диагностика JMX хендшейка с распечаткой адресов

```
> java -jar sjk.jar mxping -s 127.0.0.1:34000
SJK is running on: OpenJDK 64-Bit Server VM 25.275-b01 (BellSoft)
Java home: C:\Java\jdk1.8.0_275+1_bellsoft_x64\jre
Try to connect via TLS
Establishing connection to 127.0.0.1:34000
Failed to connect using TLS: java.rmi.ConnectIOException: error during JRMP connection establishment;
nested exception is:
          javax.net.ssl.SSLHandshakeException: Remote host terminated the handshake
Try to use plain socket
Establishing connection to 127.0.0.1:34000
Establishing connection to 192.168.100.1:34000
Remote VM: OpenJDK 64-Bit Server VM 25.275-b01 (BellSoft)
```

Стоит ли JMX этой боли?

Мониторинг

- Не стоит
- Prometheus JMX Exporter, Sprinf Micrometer и п.р.

Диагностика/профилирование

- Да, если вы хотите использовать VisualVM, MisssionControl и п.р.

Заключение

Заключение

Позитив

- Современная JVM неплохо адаптировалась к реалиям контейнеров
- `-XX:MaxRAMPercentage` – удобный способ сайзинга кучи относительно лимитов контейнера
- `jcmd` позволяет работать с локально запущенными контейнерами

Негатив

- JMX и диагностика – очень сложна в настройке
- Новые сценарии отказов (превышение лимитов)
- Когнитивный барьер: нужно учить cgroup и очередные диалекты yaml

Ссылки

Доклад JVM и Linux - особенности эксплуатации
https://habr.com/en/companies/oleg-bunin/articles/358520/

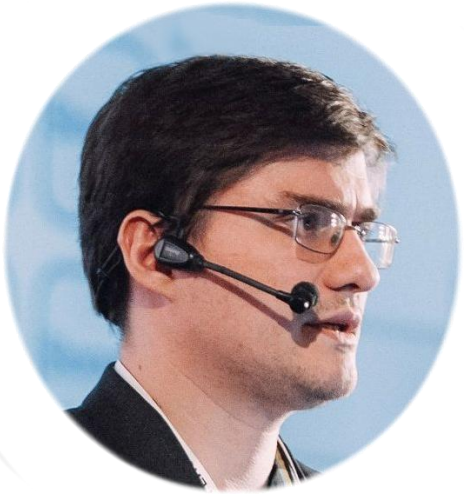
Поисковик по опциям JVM
https://chriswhocodes.com/

Гайд по JMX в контейнере
https://blog.ragozin.info/2023/09/curse-of-jmx.html

Статься по тротлингу ЦПУ
https://github.com/robusta-dev/alert-explanations/wiki/CPUThrottlingHigh-(Prometheus-Alert)

Jolokia – JMX over HTTP
https://jolokia.org/

Моя статья “JVM in Linux containers, surviving the isolation”
https://bell-sw.com/announcements/2020/10/28/JVM-in-Linux-containers-surviving-the-isolation/



Спасибо!

Алексей Рагозин

Email: alexey.ragozin@gmail.com
Блог: blog.ragozin.info
Github: https://github.com/aragozin
Митапы: https://aragozin.timepad.ru