惊呆了，Spring Boot居然这么耗内存！

Java技术江湖 今天

点击上方“Java技术江湖”，选择“设为星标”

做积极的人，而不是积极废人

Spring Boot总体来说，搭建还是比较容易的，特别是Spring Cloud全家桶，简称亲民微服务，但在发展趋势中，容器化技术已经成熟，面对巨耗内存的Spring Boot，小公司表示用不起。如今，很多刚诞生的JAVA微服务框架大多主打“轻量级”，主要还是因为Spring Boot太重。

JAVA系微服务框架

No1-Spring Cloud

介绍

有Spring大靠山在，更新、稳定性、成熟度的问题根本不需要考虑。在JAVA系混的技术人员大约都听说过Spring的大名吧，所以不缺程序员……，而且这入手的难度十分低，完全可以省去一个架构师。

但是，你必然在服务器上付出：

- 至少一台“服务发现 ”的服务器；
- 可能有一个统一的网关Gateway；
- 可能需要一个用于“分布式配置管理”的配置中心；
- 可能进行“服务追踪”，知道我的请求从哪里来，到哪里去；
- 可能需要“集群监控”；
- 项目上线后发现，我们需要好多服务器，每次在集群中增加服务器时，都感觉心疼；

压测30秒

压测前的内存占用

```
Processes: 340 total, 2 running, 338 sleeping, 1715 threads          10:12:39
Load Avg: 2.87, 2.45, 1.99  CPU usage: 2.29% user, 1.81% sys, 95.88% idle
SharedLibs: 164M resident, 53M data, 34M linkedit.
MemRegions: 56640 total, 7188M resident, 179M private, 3249M shared.
PhysMem: 16G used (2653M wired), 228M unused.
VM: 1531G vsize, 1112M framework vsize, 0(0) swapins, 0(0) swapouts.
Networks: packets: 124631/37M in, 52794/14M out.
Disks: 164600/3838M read, 52539/937M written.

PID   COMMAND      %CPU TIME     #TH  #WQ  #POR MEM   PURG CMPR PGRP PPID
1526  java         0.1  00:10.58 43   1    124  304M  0B   0B   1526 1449
```

如图，内存占用304M。

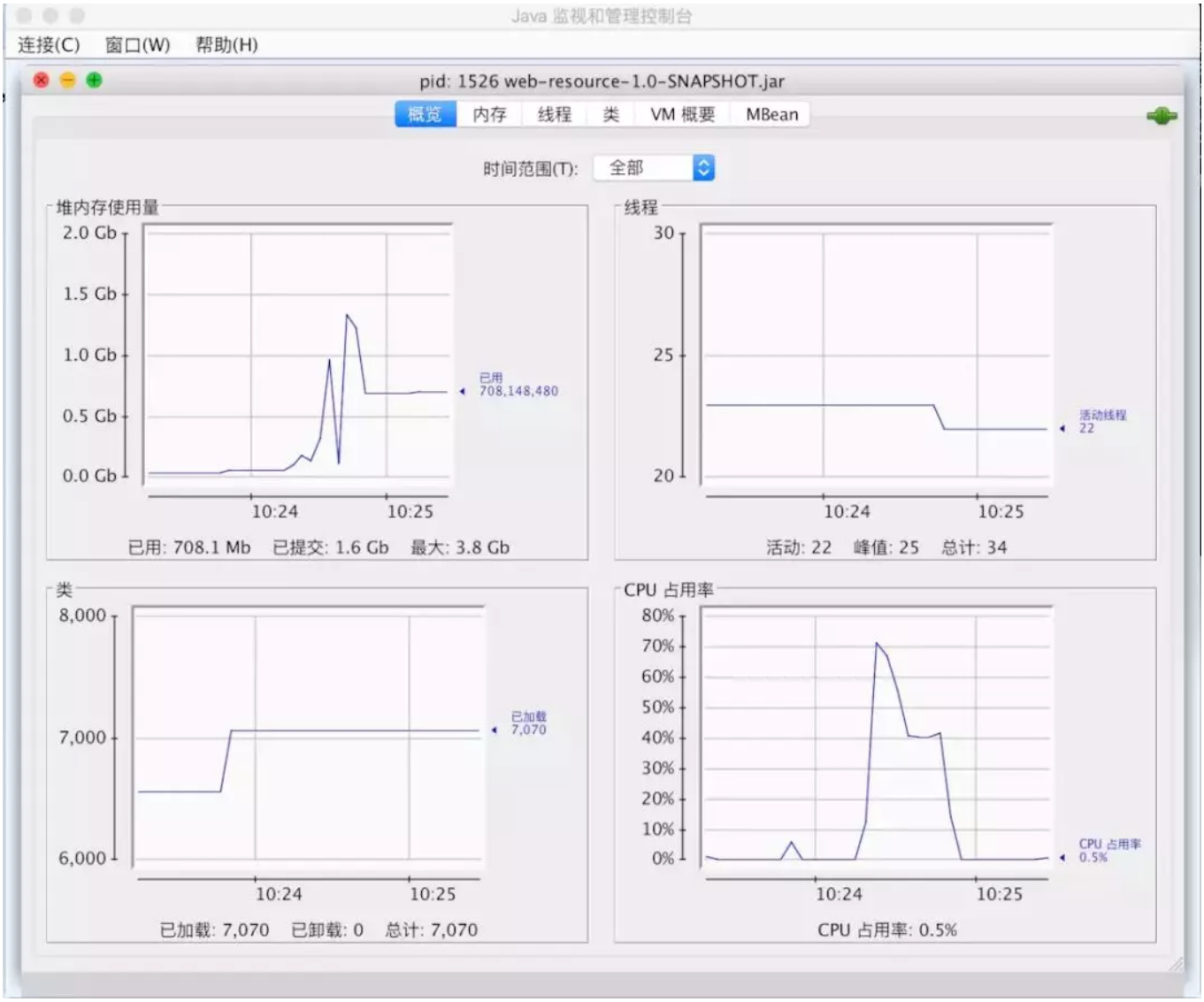
压测时的内存占用

```
Processes: 347 total, 8 running, 339 sleeping, 1917 threads          10:29:42
Load Avg: 2.43, 2.61, 2.38  CPU usage: 45.93% user, 40.90% sys, 13.15% idle
SharedLibs: 165M resident, 53M data, 33M linkedit.
MemRegions: 60070 total, 8183M resident, 184M private, 2083M shared.
PhysMem: 16G used (2721M wired), 356M unused.
VM: 1568G vsize, 1112M framework vsize, 0(0) swapins, 0(0) swapouts.
Networks: packets: 16407645/1573M in, 16310036/1547M out.
Disks: 172958/3944M read, 61862/1207M written.

PID   COMMAND      %CPU  TIME     #TH  #WQ  #POR MEM    PURG CMPR PGRP PPID
1526  java         321.0 02:50.33 41/4 1    120  1520M  0B   0B   1526 1449
```

如图，内存占用1520M（1.5G），CPU上升到321%

概览



总结

一个Spring Boot的简单应用，最少1G内存，一个业务点比较少的微服务编译后的JAR会大约50M；而Spring Cloud引入的组件会相对多一些，消耗的资源也会相对更多一些。

启动时间大约10秒左右: Started Application in 10.153 seconds (JVM running for 10.915)

JAVA系响应式编程的工具包Vert.x

介绍

背靠Eclipse的Eclipse Vert.x是一个用于在JVM上构建响应式应用程序的工具包。定位上与Spring Boot不冲突，甚至可以将Vert.x结合Spring Boot使用。众多Vert.x模块提供了大量微服务的组件，在很多人眼里是一种微服务架构的选择。

华为微服务框架Apache ServiceComb就是以Vert.x为底层框架实现的，在"基准测试网站TechEmpower"中，Vert.x的表现也十分亮眼。

压测30秒

压测前的内存占用

```
Processes: 319 total, 2 running, 317 sleeping, 1642 threads            12:11:55
Load Avg: 2.02, 1.87, 1.89  CPU usage: 3.15% user, 2.42% sys, 94.41% idle
SharedLibs: 199M resident, 58M data, 45M linkedit.
MemRegions: 69030 total, 7190M resident, 207M private, 3100M shared.
PhysMem: 15G used (2599M wired), 768M unused.
VM: 1447G vsize, 1111M framework vsize, 0(0) swapins, 0(0) swapouts.
Networks: packets: 282452/109M in, 84946/21M out.
Disks: 157962/4480M read, 126426/2246M written.

PID   COMMAND      %CPU TIME     #TH  #WQ  #POR MEM  PURG CMPR PGRP PPID
2656  java         0.1  00:01.23 25   1    88   65M  0B   0B   2656 2337
```

如图，内存占用65M。

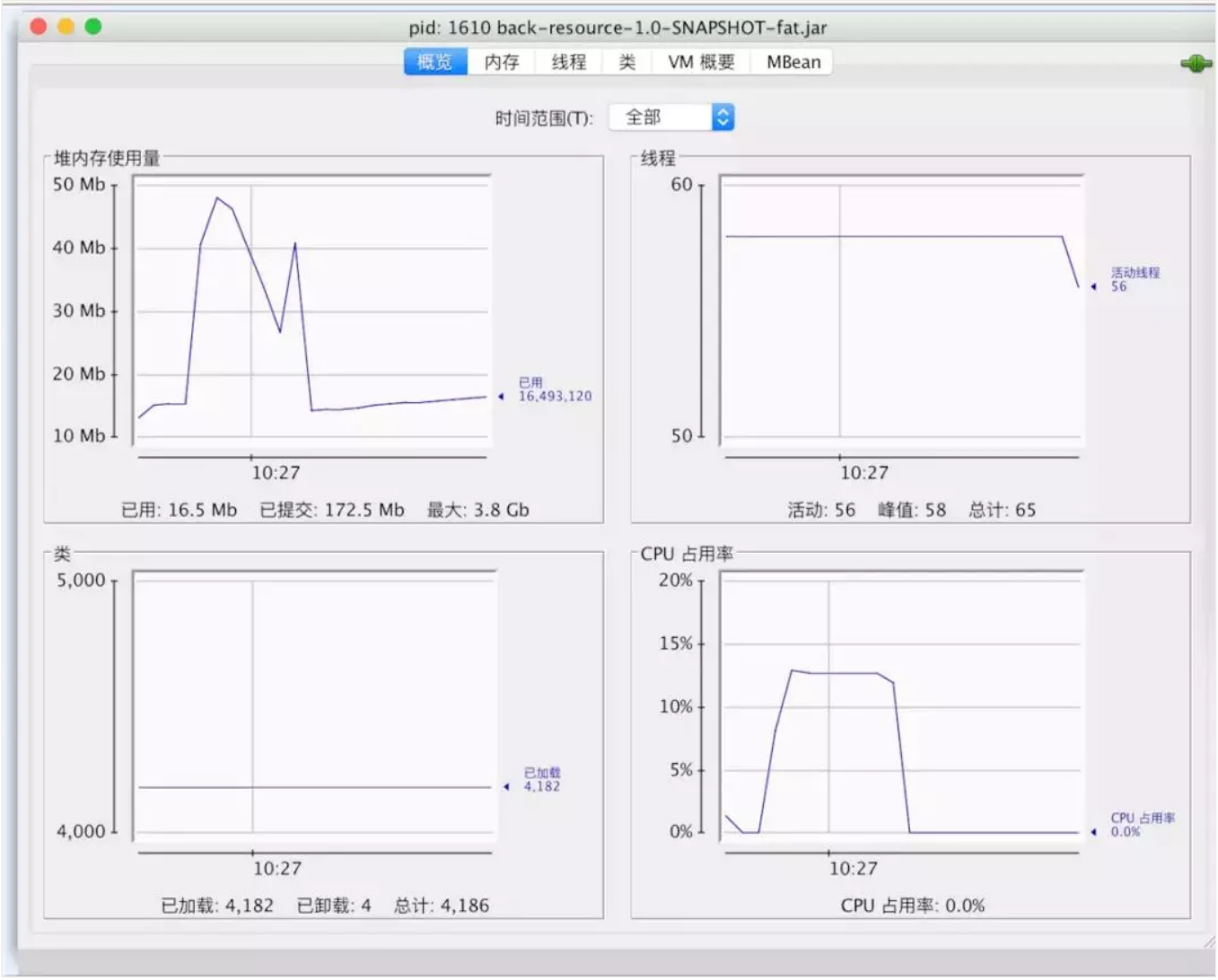
压测时的内存占用

```
Processes: 342 total, 3 running, 339 sleeping, 1682 threads            10:14:51
Load Avg: 2.05, 2.30, 1.99  CPU usage: 3.74% user, 3.14% sys, 93.10% idle
SharedLibs: 164M resident, 53M data, 33M linkedit.
MemRegions: 57424 total, 7319M resident, 180M private, 3223M shared.
PhysMem: 16G used (2683M wired), 29M unused.
VM: 1546G vsize, 1111M framework vsize, 0(0) swapins, 0(0) swapouts.
Networks: packets: 131426/38M in, 56136/14M out.
Disks: 165715/3854M read, 54070/1024M written.

PID   COMMAND      %CPU TIME     #TH  #WQ  #POR MEM    PURG CMPR PGRP PPID
1610  java         2.1  00:02.84 46   1    130+ 139M+ 0B   0B   1610 1396
```

如图，内存占139M，CPU占2.1%，给人的感觉似乎并没有进行压测。

概览



总结

Vert.x单个服务打包完成后大约7M左右的JAR，不依赖Tomcat、Jetty之类的容器，直接在JVM上跑。

Vert.x消耗的资源很低，感觉一个1核2G的服务器已经能够部署许多个Vert.x服务。除去编码方面的问题，真心符合小项目和小模块。git市场上已经出现了基于Vert.x实现的开源网关- VX-API-Gateway帮助文档

https://duhua.gitee.io/vx-api-gateway-doc/

对多语言支持，很适合小型项目快速上线。

启动时间不到1秒：Started Vert.x in 0.274 seconds (JVM running for 0.274)

JAVA系其他微服务框架

SparkJava

- jar比较小，大约10M
- 占内存小，大约30~60MB；
- 性能还可以，与Spring Boot相仿；

Micronaut

- Grails团队新宠；
- 可以用 Java、Groovy 和 Kotlin 编写的基于微服务的应用程序；
- 相比Spring Boot已经比较全面；
- 性能较优，编码方式与Spring Boot比较类似；
- 启动时间和内存消耗方面比其他框架更高效；
- 多语言；
- 依赖注入；
- 内置多种云本地功能；
- 很新，刚发布1.0.0

Javalin

- 上手极为容易；
- 灵活，可以兼容同步和异步两种编程思路；
- JAR小，4～5M；
- 多语言；
- 有KOA的影子；
- 只有大约2000行源代码，源代码足够简单，可以理解和修复；
- 符合当今趋势；
- 多语言；
- 嵌入式服务器Jetty；

Quarkus

- 启动快；
- JAR小，大约10M；
- 文档很少；

来源：http://t.cn/Ai8xQ3lT

THE END

推荐阅读：

MyBatis中的$和#，用不好，准备走人!

点个“在看”，转发朋友圈，都是对我最好的支持!