Java Performance Enhancements

Jim Johnston

IBM Z

IBM

Disclaimer

Any reference to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.

Agenda

MMAP Private Allocator for Java

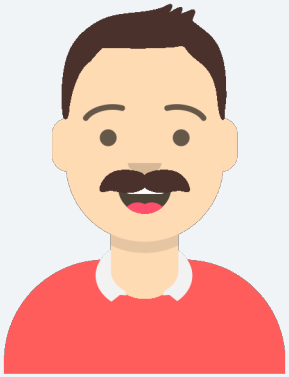
1Q 2021 Refresh (Java 8 SR6 FP25)

Future Java Performance Enhancements

Problem Statement

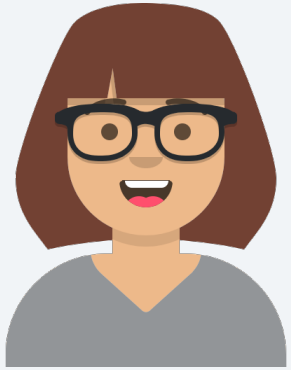
How can we improve memory management for Java on z/TPF?

Users



Brian
Business Executive

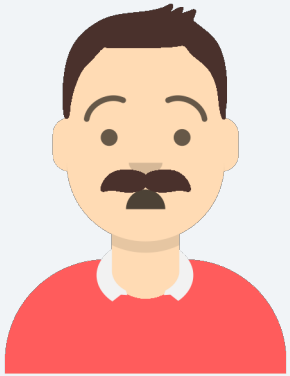
"I like having Java on z/TPF because we can leverage existing technologies like Kafka to integrate with other distributed technologies creating new business opportunities."



Anna
Application Architect

"I'm generally excited about Java on z/TPF because it adds another programming language option going forward and facilitates modernizing some of our z/TPF applications."

Pain Points



Brian
Business Executive

- Brian likes how Java is being used to leverage high business value applications like Kafka, but he is concerned about the potential IT costs associated with the increase in memory requirements.

Pain Points

Anna
Application Architect

- Anna likes how Java adds more alternatives to application development languages, but she is frustrated when it comes to getting insight into memory usage so that she can validate her memory configurations.

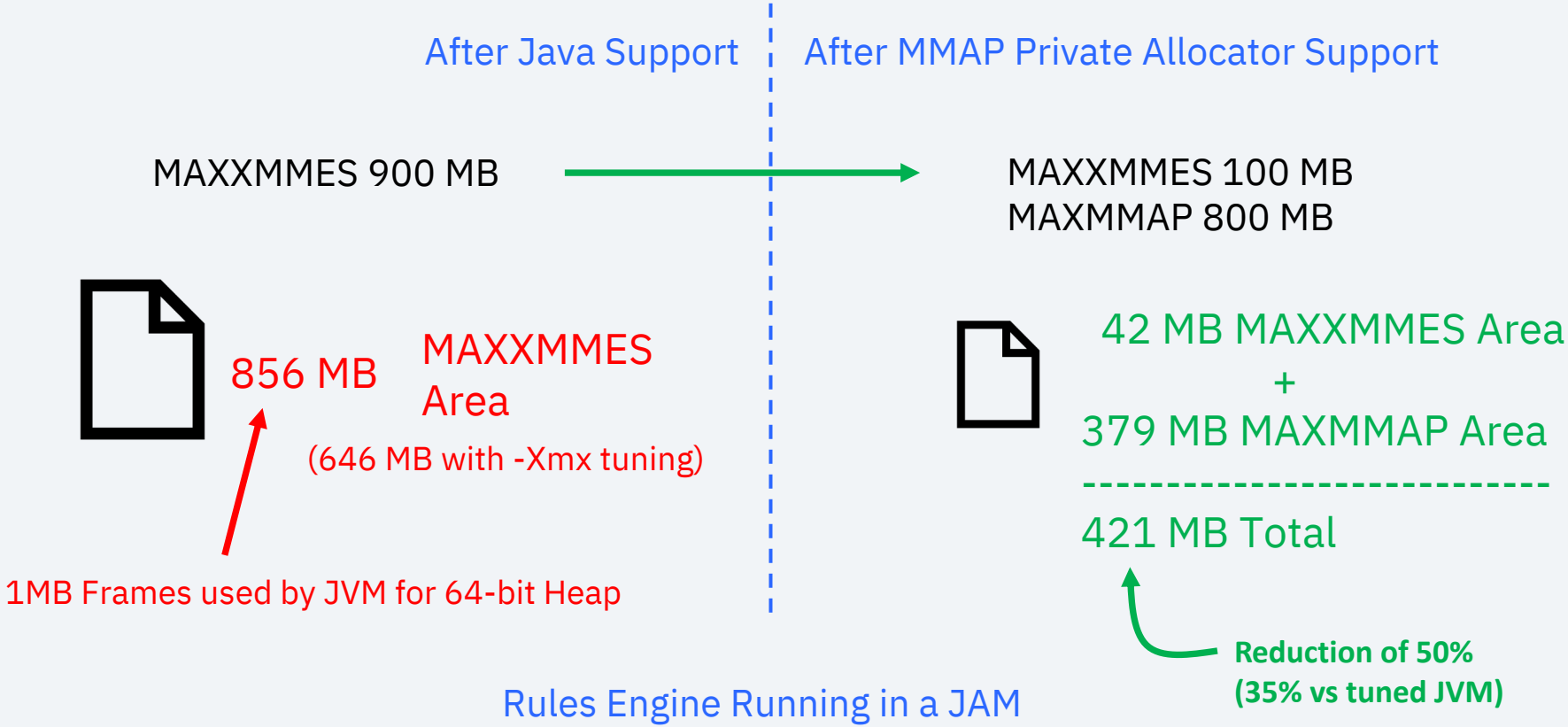
Support for MMAP Private Allocator for Java

Support uses new 64-bit heap area which utilizes physical memory better

- Reduces total 1 MB frames used by Java applications
- Reduction in memory requirements to support Java
- Reduction in Java system dump size
- Provides insight into Java application memory utilization through Javacore diagnostic.

PJ46404
(April 2021)

Value – Minimizes Memory Used by Java Process



Value – New 64-bit Heap Limit Configuration for Java

Before Java Support

550 ECBs defined: 500 Transactional Workload ECBs, 10 Processes (5 thread ECBs each)
Transactional work uses at most 10 MB of 64-bit heap so XMMES is set to 10.
Long running threaded processes (like shared SSL) need 50 MB of heap so MAXXMMES set to 50
1MB frames needed for 64-bit ECB heap = (500*10) + (10*50) = **5500**

Add in Java Support

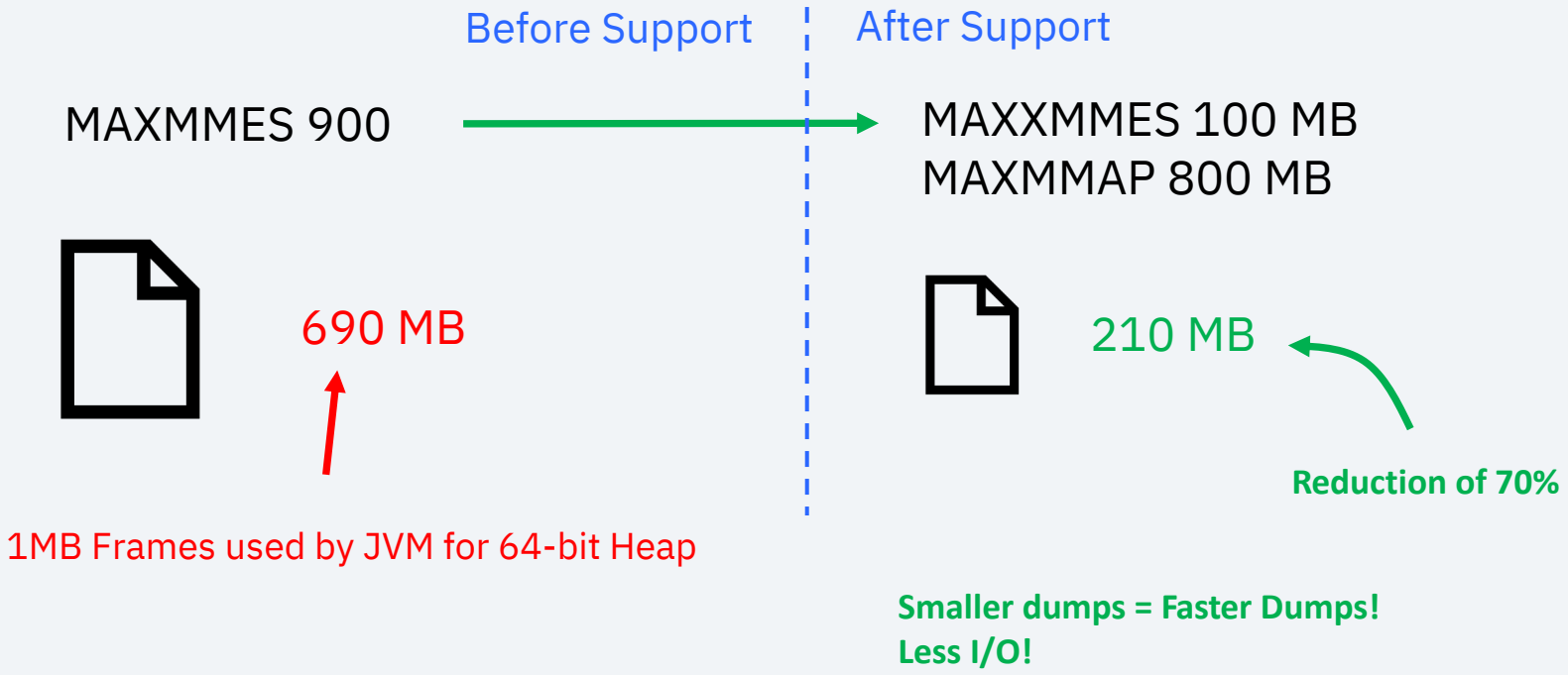
Add 3 JVMs requiring 900 MB each and 3 JVMs requiring 200 MB each for 6 total JVMs.
Must set MAXXMMES to 900 now
1MB frames needed for ECB heap = (500*10) + (10*900) + (6*900) = **19,400**

After MMAP Private Support

Change MAXXMMES from 900 down to 100 (lowest value recommended for Java)
Define new MAXMMAP value to be 800
1MB frames needed for ECB heap = (500*10) + (10*100) + (6*(800+100)) = **11,400**

41% reduction

Value – Smaller System Dumps for Java



Value – Insight into Memory Utilization

New section in javacore for diagnostics.
"zfile kill –s SIGIQUIT <JVM PID>" to capture javacore.
Non-disruptive, JVM continues running.

```
==========================================
Native memory stats for z/TPF
==========================================
Total highwater mark 64-bit heap 1MB Frames:     421
Total highwater mark 31-bit heap 1MB Frames:     6
Highwater mark 64-bit MMAP heap 1MB Frames:      379
Current process 64-bit MMAP heap limit:          900
MMAP Region Virtual Address start:       0x00000003C4C00000
MMAP Region highwater mark Virtual Address end:       0x00000003F0C00000
Highwater mark 64-bit MAXXMMES heap 1MB Frames:  42
Total GC heap 1MB Frames:        118
Total JIT Code Cache heap 1MB Frames:      15
```

EMPS Determination

MAXMMAP Determination

Actual MAXMMAP setting at startup

MAXXMMES Determination

Garbage Collector (-Xmx) Determination

Recap

- Support provides memory utilization improvements & some configuration relief
- Delivered with PJ46404 (April 2021)
- Pre-requisite JRE Runtime 8.0-6.20 (PJ46358, Jan 2021)
- Javacore enhancement Pre-requisite JRE Runtime 8.0-6.25 (PJ46432, April 2021)

1Q 2021 Refresh (Java 8 SR6 FP25)

Performance Improvements

88%

Reduction in JIT CPU Consumption

50%

Reduction in High Water Mark Response Time

4%

Increase in Peak Message Rate

Rules Engine Running in a JAM

1Q 2021 Refresh (Java 8 SR6 FP25)

New User Exit - UJVM

JVM Startup User Exit

Allows for custom hooks at the very beginning of JVM startup. For example, custom data collection hooks or custom debugging hooks.

Thread Startup User Exit

Allows for customizing thread behavior based on JVM defined components. For example, marking certain threads as low priority or thread scoped custom data collection hooks.

Dump Agent User Exit

Allows for customizing actions to take after a dump agent runs. For example, using FTP to offload a system core from z/TPF.

Recap - 1Q 2021 Refresh (Java 8 SR6 FP25)

- Provides performance optimizations
- New User exits provided
- MMAP Insight Javacore summary enabled
- Refresh associated with PJ46432 (April 2021)
- Required updates to offline *offldr* utility are included

Don’t forget to load JRE shared objects using updated offldr!

Possible Future Performance Enhancement Work

- MMAP Private Configurable per Process Support
- MMAP Shared Allocator for Java (Class Cache Recoverable Storage)
- Further JIT and GC Performance Improvements (utilize fenced i-streams, cached JIT compilations)
- Java Hardware Encryption Support

If you want to be a Sponsor User for any of the above contact jjohnst@us.ibm.com

Possible Future – MAXMMAP Configurable per Process

After MMAP Private Support

3 JVMs requiring 900 MB each and 3 JVMs requiring 200 MB each for 6 total JVMs.
Change MAXXMMES from 900 down to 100 (lowest value recommended for Java)
Define new MAXMMAP value to be 800
1MB frames needed for ECB heap = (500*10) + (10*100) + (6*(800+100)) = 11,400

After Future MMAP configurable per Process Support

Now can differentiate 3 JVMs requiring 200 MB each.
1MB frames needed for ECB heap = (500*10) + (10*100) + (3*900)+(3*200) = **9,300**

18% additional reduction

Thank you

Virtual TUG Q&A

Question	Answer
Is the MMAP memory for java in a new physical memory allocated from cctin? or a new use of 1M frames physical memory ?	The mmap area is mapped above the general 64-bit heap area, it pulls from the same 1 MB frames as the general 64-bit heap area.