FusionReactor Webinar: Troubleshooting JVM Memory Problems with FusionReactor

INTERGRAL information solutions

fusion reactor™

Introductions

Charlie Arehart

Independent Consultant, CArehart.org

(Focused on server troubleshooting)

Agenda

- Foreword
- Some common misconceptions about memory/memory problems
- Key components related to memory for java servers
- What if JVM memory spaces fill?
- How FusionReactor can help prove, disprove, or diagnose a memory problem
- If heap use IS high, what may be the cause?
- What if FR can't help explain high heap?
- How FusionReactor will help with heap problems even more in the future

Foreword

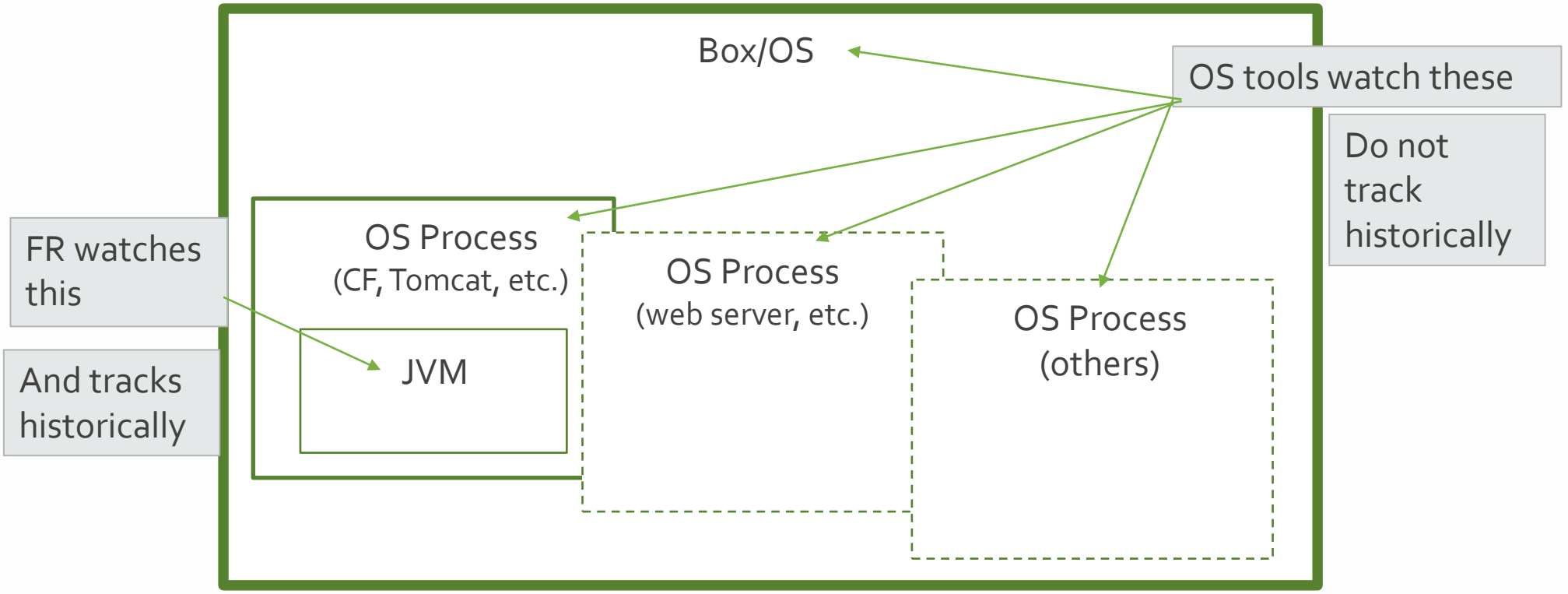
- Audience: presumed to already be using FR
  - But perhaps not using it to its fullest extent, especially regarding memory issues
- Concepts apply generally to any Java/CFML server that FR can monitor
- Preso is being recorded, so you will be able to revisit details

Some common misconceptions about memory

- Why "memory" problems aren't always what they seem
  - Is a seeming memory problem really a cause or an effect?
- Why a memory problem is typically NOT a "memory leak"
- Why mem problems may have nothing to do with JVM tuning or GC algorithms
- Why common tips and tools (and myths) often fail to help

Key components related to memory for Java servers

- When looking at a "memory problem", we need to determine/consider where the problem really is/may be
  - One can be misled based on where/how they look…



Memory within the JVM (not in-geek-depth)

- Heap: what it is, how it's used:
  - How long remain in the heap, even when no longer in use
  - What is garbage collection, minor and major
    - No need to babysit forcing GCs; why is the button there?
  - How heap is divided into generations:
    - Depends on GC algorithm. Traditionally eden space, survivor space, old gen
- Non-heap JVM memory areas:
  - Metaspace/permgen
  - Codecache
  - Compressed class files
- Thread space
  - Xss (does not set max thread space but rather space per thread)

What if JVM memory spaces fill?

- Limits are set in the JVM via arguments, for max/min respectively (defaults vary)
  - Xmx/xms (these set the heap size)
  - XX:Maxmetaspacesize/metaspacesize (Java 8)
  - XX:Maxpermgen/permgen (Java 7 and earlier)
  - XX:ReservedCodeCacheSize/InitialCodeCacheSize
  - XX:CompressedClassSpaceSize
- Errors which can happen: some passing by and some crashing the JVM
  - Where to find them: console logs, jvm logs

How FusionReactor can help prove, disprove, or diagnose a memory problem

- How FR quickly, easily shows heap use (use, allocated, max)
- How to drill in on heap use over time (since instance startup)
- How FR monitors all JVM memory spaces (and over time)
- How FR monitors garbage collections (number, duration, over time)
- How and when you might force a GC
- How FR logs all these over time (30 days by default), mem spaces and GCs
- Ways FR can be misinterpreted to suggest a memory problem

If heap use IS high, what may be the cause?

- COULD be about some one or a few long-running requests using lots of memory, sure
- But usually it's about some objects that are living far longer than expected (not a "leak")
- In the case of CFML (and to similar degree in java web apps), common causes:
  - Sessions (they live beyond life of request, until timeout), may be high in number (see FR)
  - Shared variable scopes (application, server), which live generally until restart (why not see FR)
  - Query caching (CF Admin setting, App setting, and controlled by coding. See FR)
  - Template caching (CF Admin setting. See FR)
  - ehcache caching of pages, objects, etc. (since CF9)
  - ORM caching
  - VFS (virtual file system, since CF9)
  - Change in CF10: above objects are cached per-app vs across all apps
  - Impact of CF Enterprise Server Monitor, if "memory tracking" enabled
- So what about true "leaks", caused by bugs, etc.? …

What if FR can't help explain high heap?

- Some problems, especially true "leaks", can be hard to identify with FR alone
- In this case, the next step is heap analysis
  - In my experience, this is a last resort, especially for CFML developers
  - Often hard to connect the dots to the CFML creating the object, but not impossible
  - Of course, for pure java developers, heap analysis can be far more useful
- Many tools (free and commercial) can provide this analysis
  - VisualVM included with JDK is adequate. Others include Eclipse MAT, YourKit, JProfiler
  - Involves taking a heap dump (may be gigs in size but usually pretty quick)
    - Tracks details of every object in memory (and connections among them)
    - Can be done manually or via jvm arg creating it on outofmemory condition
  - Next step is to analyze heap by such characteristics as:
    - Largest classes by size or count/percentage
    - Filtering by name
    - Finding GC roots
- Beyond the scope of this webinar topic to elaborate

But FusionReactor will help with this in the future

- Next release, FR 7 (due later this year), WILL include heap profiling tool
  - No need to install JDK tools
  - No need to worry about whether JVM/app was started as service, or by what user
  - No need to configure RMI or open ports, etc.
- Will be provided as interface feature within FR web UI
  - Just like FR6 Ultimate added step debugging without any IDE
- Will enable viewing, analysis of heap dump, similar to traditional JVM tools
  - Including counts, sizes, filtering on class names, finding GC roots, etc
- We're planning a webinar on this after it becomes available
- Like debugger, memory profiling will be an FR Ultimate feature
  - So many great features now to justify considering that:
    - Memory profiler
    - Request profiler
    - Step debugger
    - And more

What about JVM tuning? GC algorithm choices?

- Finally, you may notice I have not talked about JVM tuning
  - The args to control ratios among heap generations, etc.
  - Features to disable explicit GC
  - Choosing different GC algorithms
- In my experience (10 years of troubleshooting), these are not the solution
  - They generally arise from people in a panic, googling for any possible answer
  - Often resources found are from people without good tools to diagnose issues
  - And they recommend these jvm tweaks, like darts thrown at balloons
- I'm not saying you never need to tweak such args or change GC algo
  - Just saying it's not ever been the solution to GC problems I've found
  - The approaches discussed here have worked, nearly always

Perhaps simplest solution

- Consider increasing your heap size!
  - Your box may have gigs of available memory, while your app server remains constrained
  - You may be suffering outofmemory errors needlessly
- Sure, it may be that you'll just "delay the inevitable", need to raise again and again
  - But very often there could be a heap size where your app runs comfortably for days/weeks
- Beware also that you may THINK the heap is set to more than it really is
  - Recall how FR can SHOW you what the heap max is, in its UI
  - Make sure it's showing to be (near) the amount you expect
- Finally, note that sometimes one REGION of the heap may fill, unexpectedly
  - There are JVM args to help tweak that
  - But again sometimes the right solution is find the CAUSE for them filling, rather than tweak

Conclusion

- Memory problems are not often what they seem
  - What may seem a memory "leak" may just be long-lived objects
  - Most memory problems are an effect: challenge is to find the root cause
- Solving most memory problems is not about JVM tuning or GC algo choices
  - Instead the challenge is to find the memory space in trouble
  - Then find what's causing that to fill
  - Maybe consider raising it
- FusionReactor provides several tools, in UI and logs, to help
  - Tracks heap use, and spaces within heap
  - And memory spaces outside of heap
  - And garbage collection
- And next release will include heap profiling tool, adding another vital tool

Other upcoming webinars

- More on analyzing FR logs with Excel – TBA
- Troubleshooting and Identifying Issues using FusionReactor 6 - Part 2 – TBA
- Registration: www.fusion-reactor.com/webinars
  - Recordings of past webinars also offered there

Other FR resources

- **FR web site: fusion-reactor.com**
  - Downloads
  - Docs, videos, technotes, forums, and much more
- **Email: sales@fusion-reactor.com, support@fusion-reactor.com**
- **Phone: (978) 496-9990 (sales)**
- **Consulting assistance: cfconsultant.com**
- We welcome your feedback on these or the other webinars, or any you would like to see

Questions & Answers