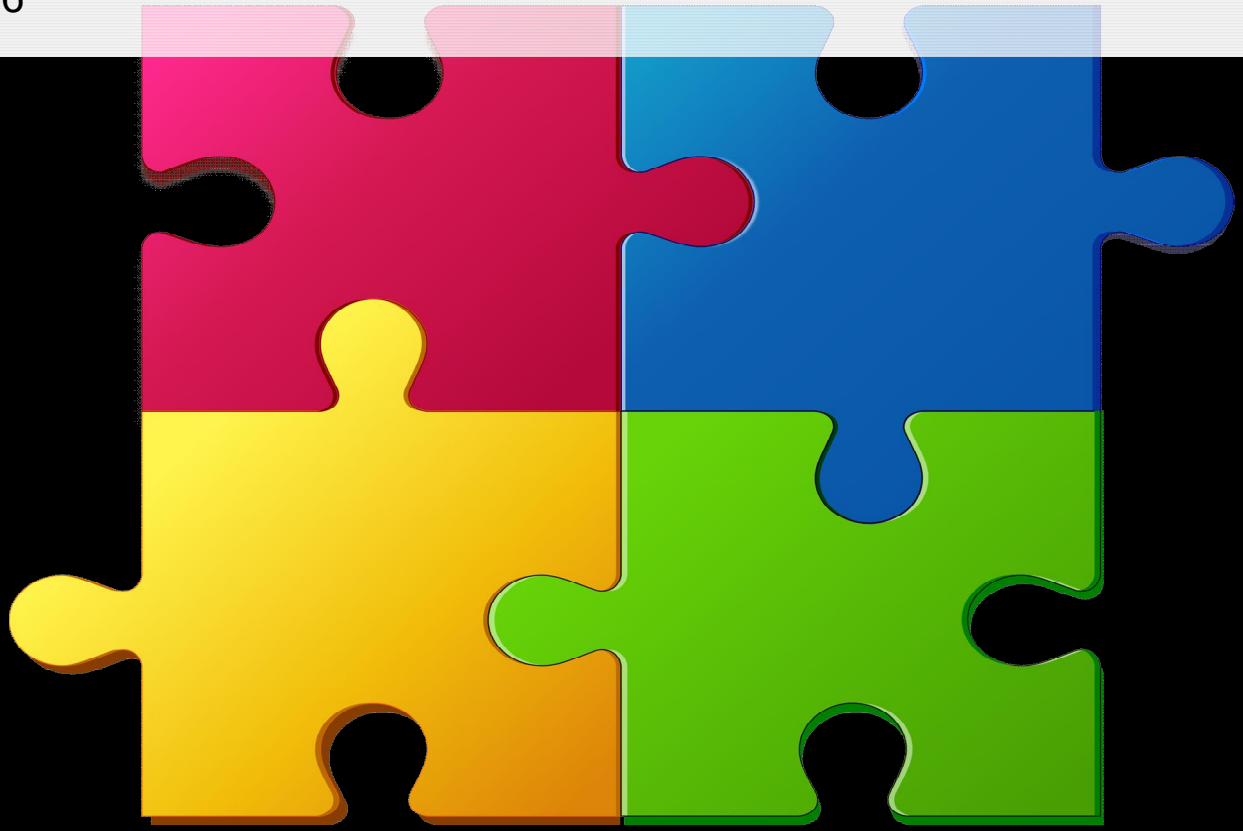
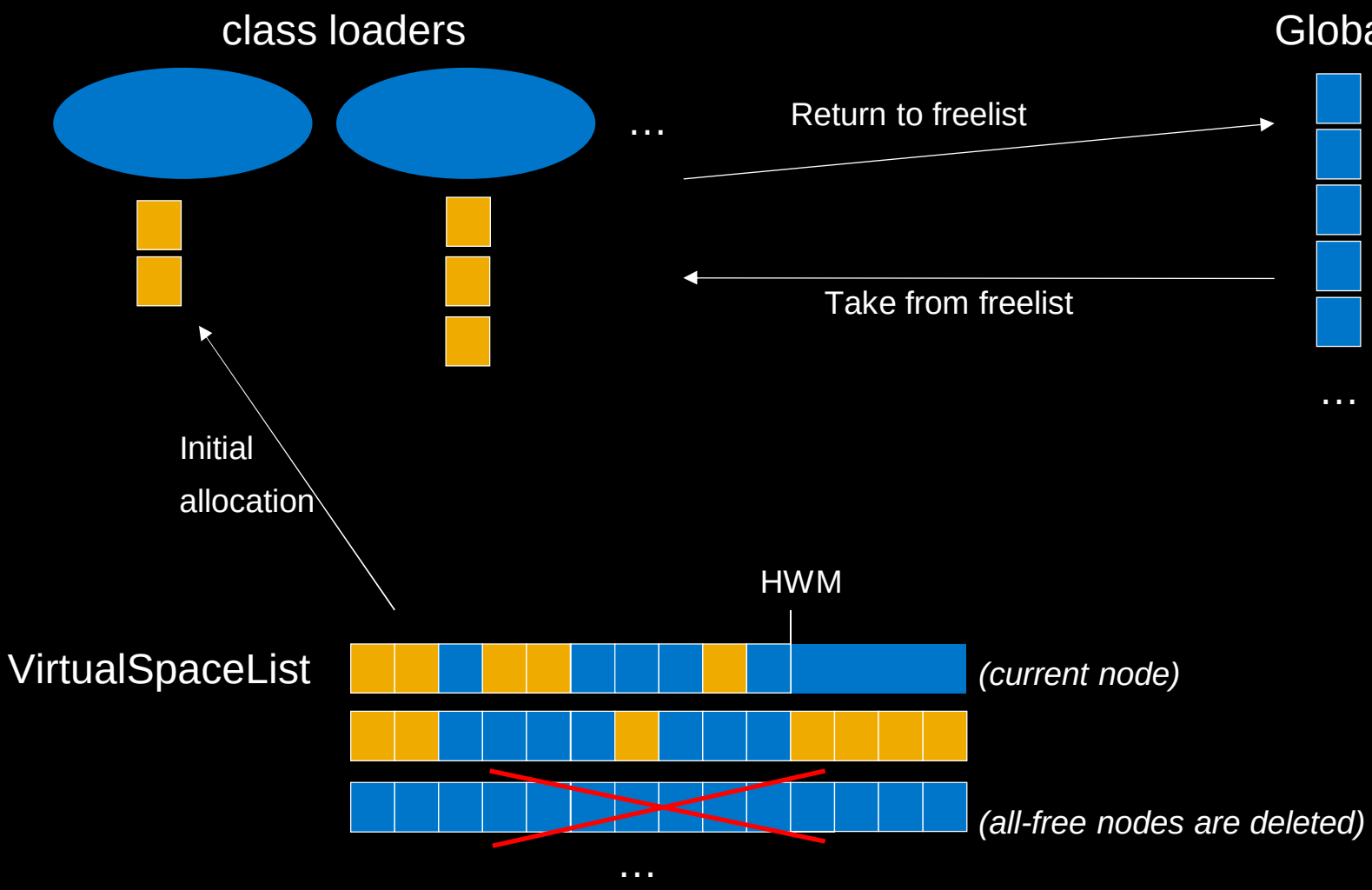
Public

Metaspace Chunk Coalescation in the SAP JVM

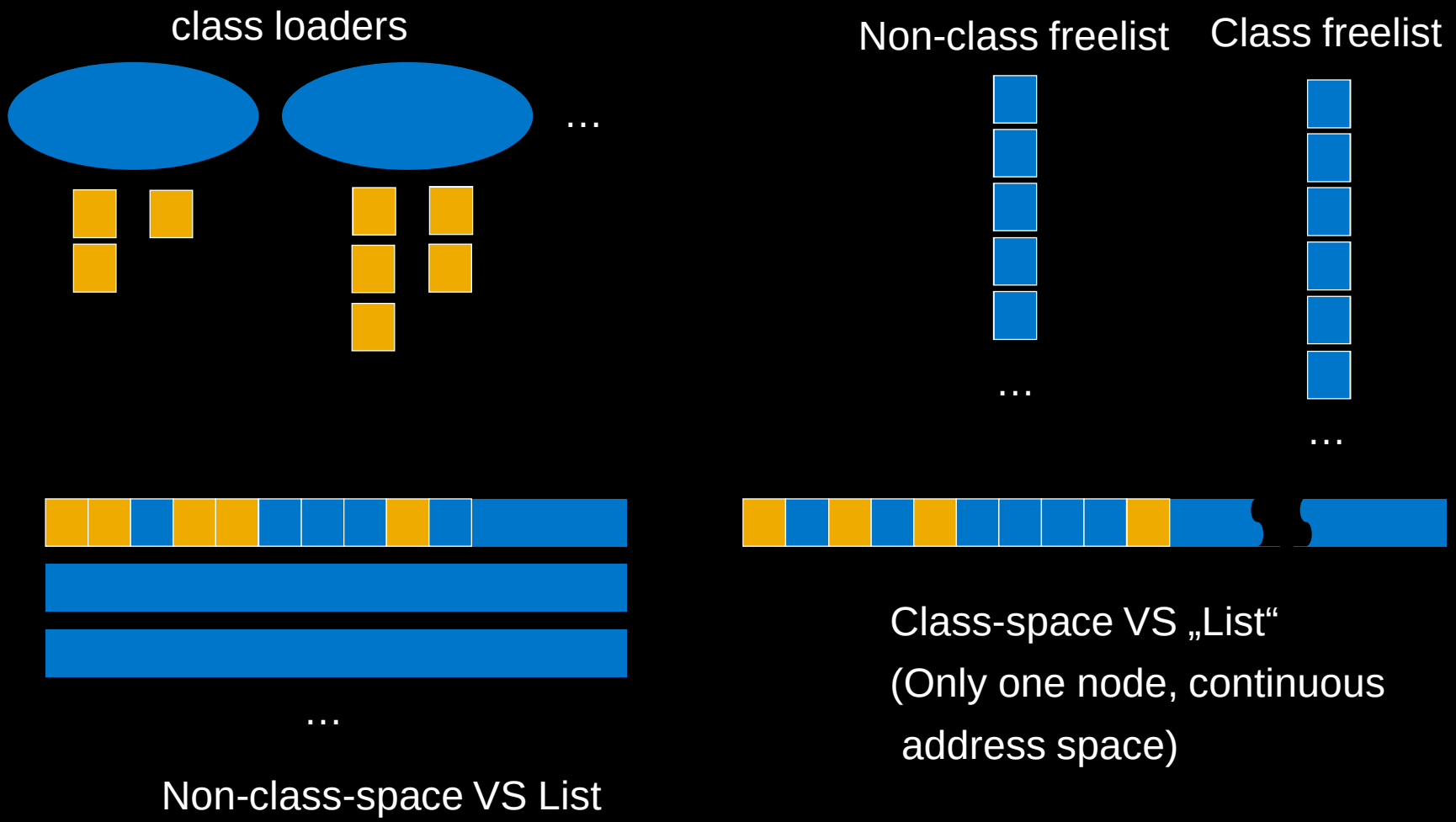
Thomas Stüfe, SAP
Sep 28, 2016



Metaspace Chunk Allocation



… x2 with compressed class space



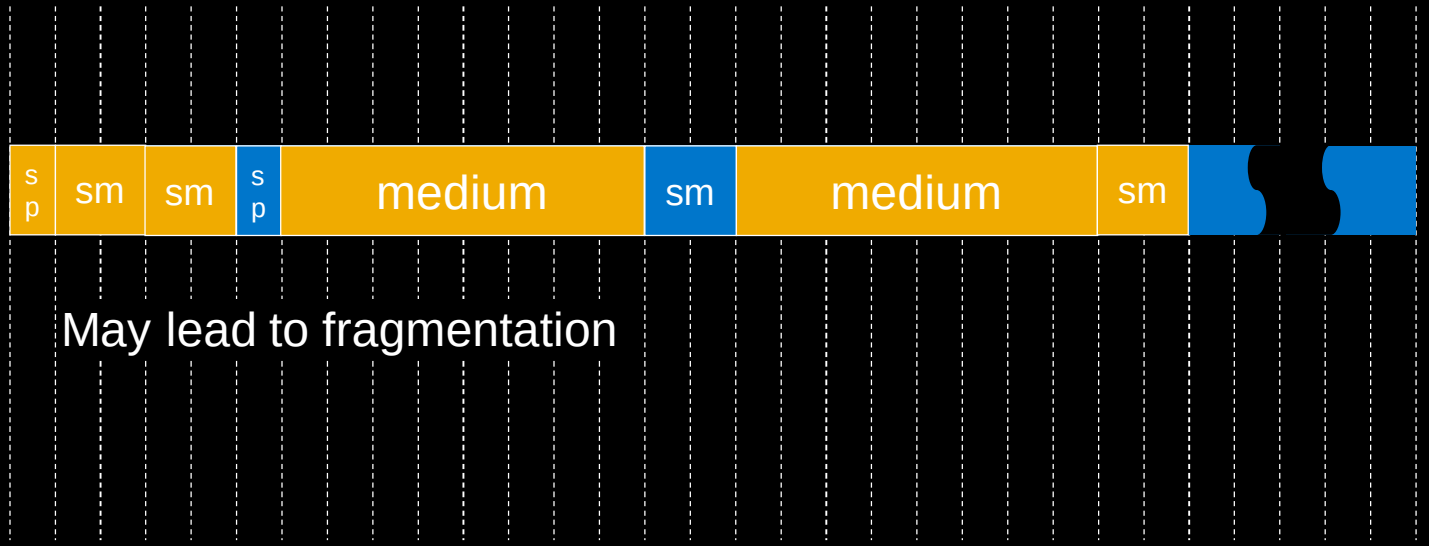
Metaspace chunk come in four sizes

- “Specialized” (128 words), “Small” (256/512 words), “Medium” (4K/8K words), “Humongous” (large, variable sized)

- A class loader first gets 4 small chunks – only after using them up, allocator switches to medium chunks.

Chunk Allocation (now)

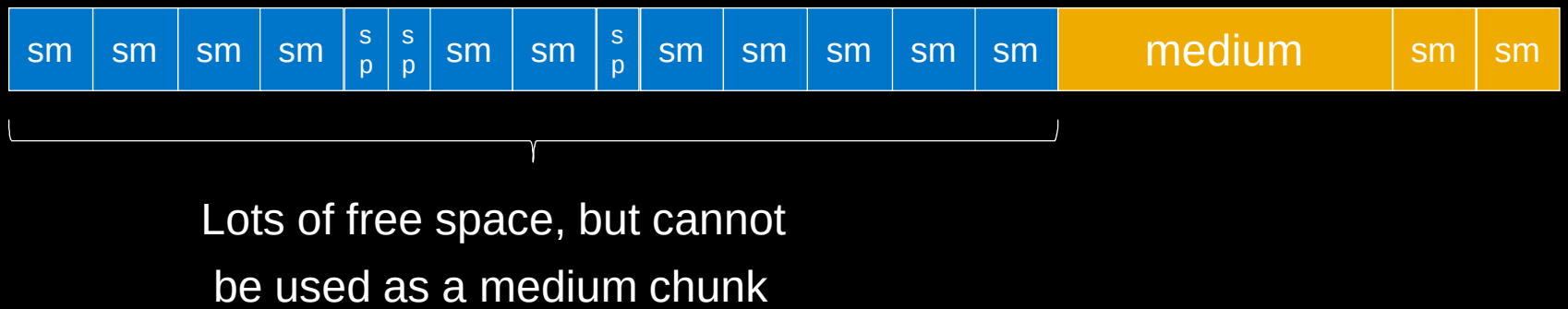
- Chunks cannot be moved
- *Chunks cannot be merged or split*
- Chunks are placed in order of allocation
- Chunks are allocated aligned to smallest chunk size



The Problem

Chunk cannot be changed in size, cannot be split or merged

- We may get an OOM *even though* the majority of the metaspace may be free: the metaspace may be already chopped up into chunks of the wrong size.



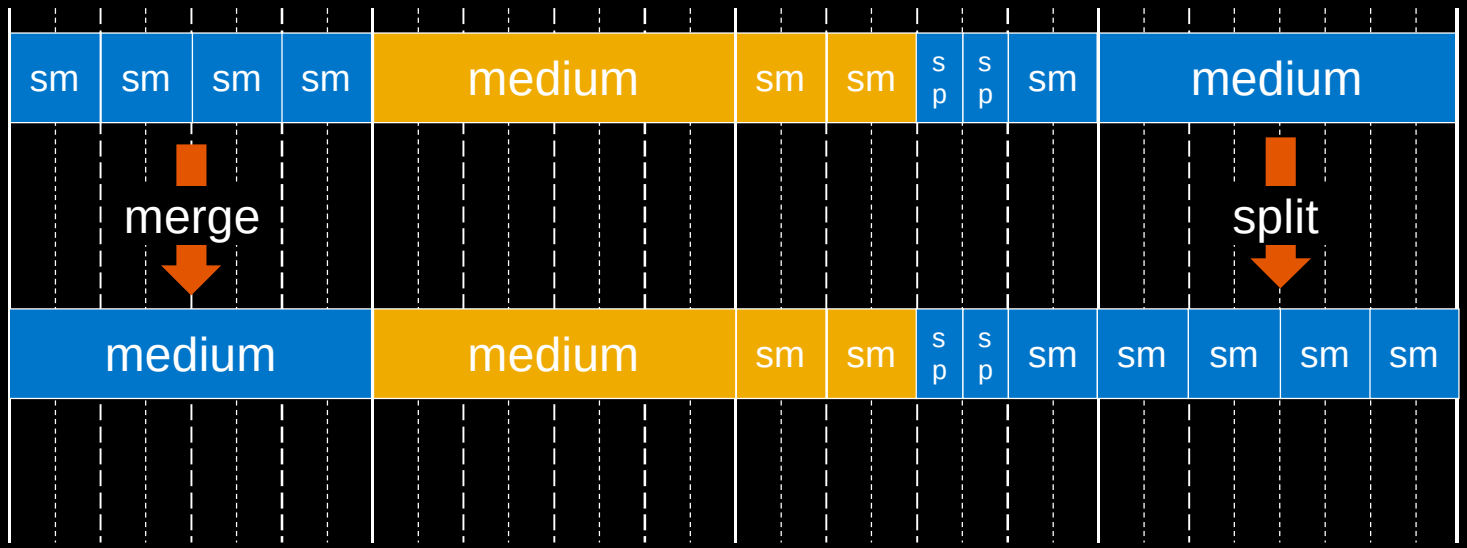
The Problem (contd.)

Example: Lots of class loaders, each loading only a few small classes. A lot of small chunks are created. Even if class loaders are unloaded, the small chunks remain – free, but unavailable to form a medium chunk.

Effectively, once allocated chunks are “locked in” into their size.

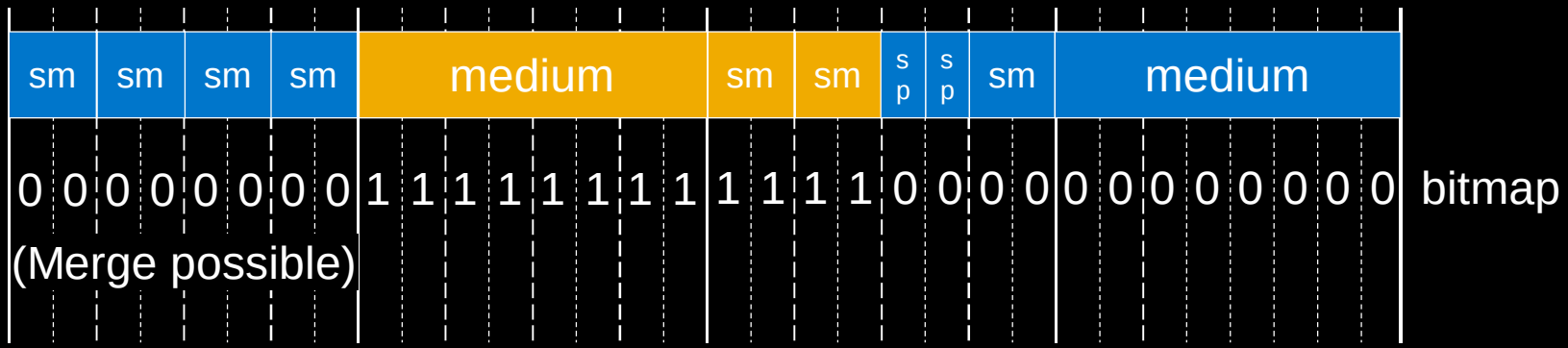
Chunk Allocation (with coalescation patch)

- Chunks are aligned at chunk-size boundaries (except humongous)
- Chunks can be merged and split



Are-neighbors-free check

- Before merging chunks, we need to check if neighboring chunks are free
- We use a bitmap to store in-use information for each smallest-chunk-sized range



Chunks are aligned to chunk size, and medium chunks are x32/x64 smallest-chunk-size => it is very cheap to check for a potential medium chunk sized merger: just a 32/64bit load & compare with zero

Chunk Merge

- Chunk merge happens proactively, when a chunk is returned to the free list
  - For the medium-sized-chunk range the freed chunk is part of do:
    - Check if all neighbors are free (via bitmap)
    - Yes: Remove neighbors from freelist, form new medium sized chunk, add it to freelist

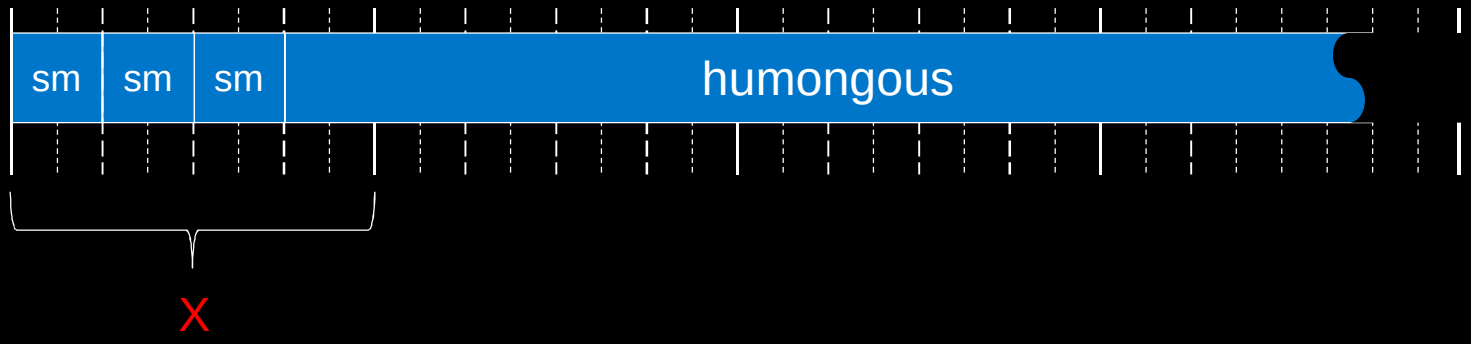
- Repeat this for a potential specialized-chunks to small-chunk merger

Chunk Split

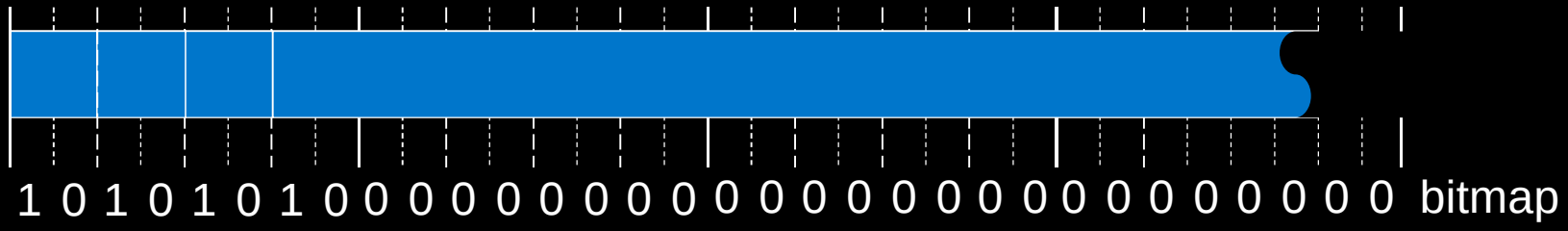
- Chunk split happens when needed: we have only large chunks, but need a small chunk
  - Remove large chunk from freelist
  - Split it into n smaller ones.
  - Return n-1 chunks to freelist
  - Return one chunk to caller

Humongous chunks are special…

Humongous chunks may straddle merge boundaries – merge not possible even if all chunks are free..



Hence, to find prospective merging boundaries, we need a cheap way to find out if at a given point a chunk starts. We use a second bitmap for that (1 for „chunk starts here“)

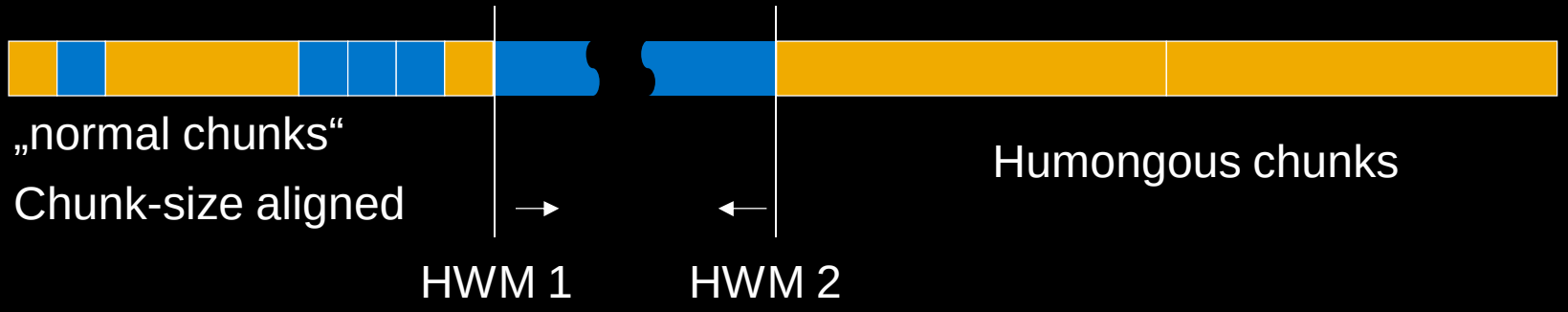


… (contd.)

Humongous chunks are not aligned to their chunk size (impossible to do) but still share the same space with „normal“ chunks.

They need a lot of special treatment.

Alternative: Lets humongous chunk live at the end of a virtual space node? (We did not do this in the SAP JVM, too many changes, but maybe possible in the OpenJDK?)



Success

Example program: loads many small classes in many small classloaders, unloads them, then loads large classes.

Ran with CompressedClassSpaceSize=10M

No patch: OOM (class space) after loading ~ 1000 large classes, only 40% of class metaspace used, 60% of chunks in freelist

With patch: OOM (class space) after loading ~ 3000 large classes, class metaspace mostly used, no chunks in freelist at OOM

Using Metaspace Coalescation in the SAP JVM

Switches:

- **-XX:[+-]CoalesceMetaspace**
- **-XX:VerifyMetaspaceEveryNth=<0…n>: Debug switch - run a verification every n allocation requests. 0 = deactivated (default), n>0 = every nth request**
- **-XX:[+-]PrintMetaspaceStatisticOnOutOfMemoryError: print a metaspace statistic if an OOME (metaspace related) happens to stderr.**
- **-XX:[+-]PrintMetaspaceMapOnOutOfMemoryError: print an ascii-art metaspace map if an OOME (metaspace related) happens to stderr.**

Contribute to OpenJDK?

Draft for JEP: https://bugs.openjdk.java.net/browse/JDK-8166690

Currently waiting for input from community.

Thank you

Contact information:

thomas.stuefe@sap.com