Transparent Memory Compression in Linux

Seth Jennings
sjenning@linux.vnet.ibm.com
LinuxCon 2013
IBM, LTC

© 2006 IBM Corporation

Overview

- What is Transparent Memory Compression
- Swap is a four letter word
- What is zswap and how does it work
- Use cases and gotchas
- Other/Future work

© 2013 IBM Corporation

Transparent Memory Compression

- Kernel dynamically compresses process memory without process knowledge
- Achieved through the use of process virtual address space and demand paging
- The kernel can unmap pages from the process page table and compress them
- When a compressed page is accessed, the page fault handler uses the information in the Page Table Entry (PTE) to find the page in the compressed pool, decompress it, and link it back into the page table

© 2013 IBM Corporation

Swap is a Four Letter Word

- RAM sizing for a machine or virtual machine is usually done on a peak load basis
- Any workload that overcommits memory accessing a working set whose size exceed the RAM size becomes a I/O bound load via swapping
- We'll just talk about the anonymous memory overcommit situation for now
- Swap is a four letter word for most sysadmins

© 2013 IBM Corporation

Swap is a Four Letter Word

- Swapping pages out doesn't necessarily hurt performance
- The scanning and unmapping is typically done by kswapd, not the workload thread
- Cost is in the CPU overhead of scanning and unmapping and I/O to the swap device (aysnc)
- Swapping pages in is the problem
- The thread is halted on the page fault until the page can be read from the swap device (~10ms)

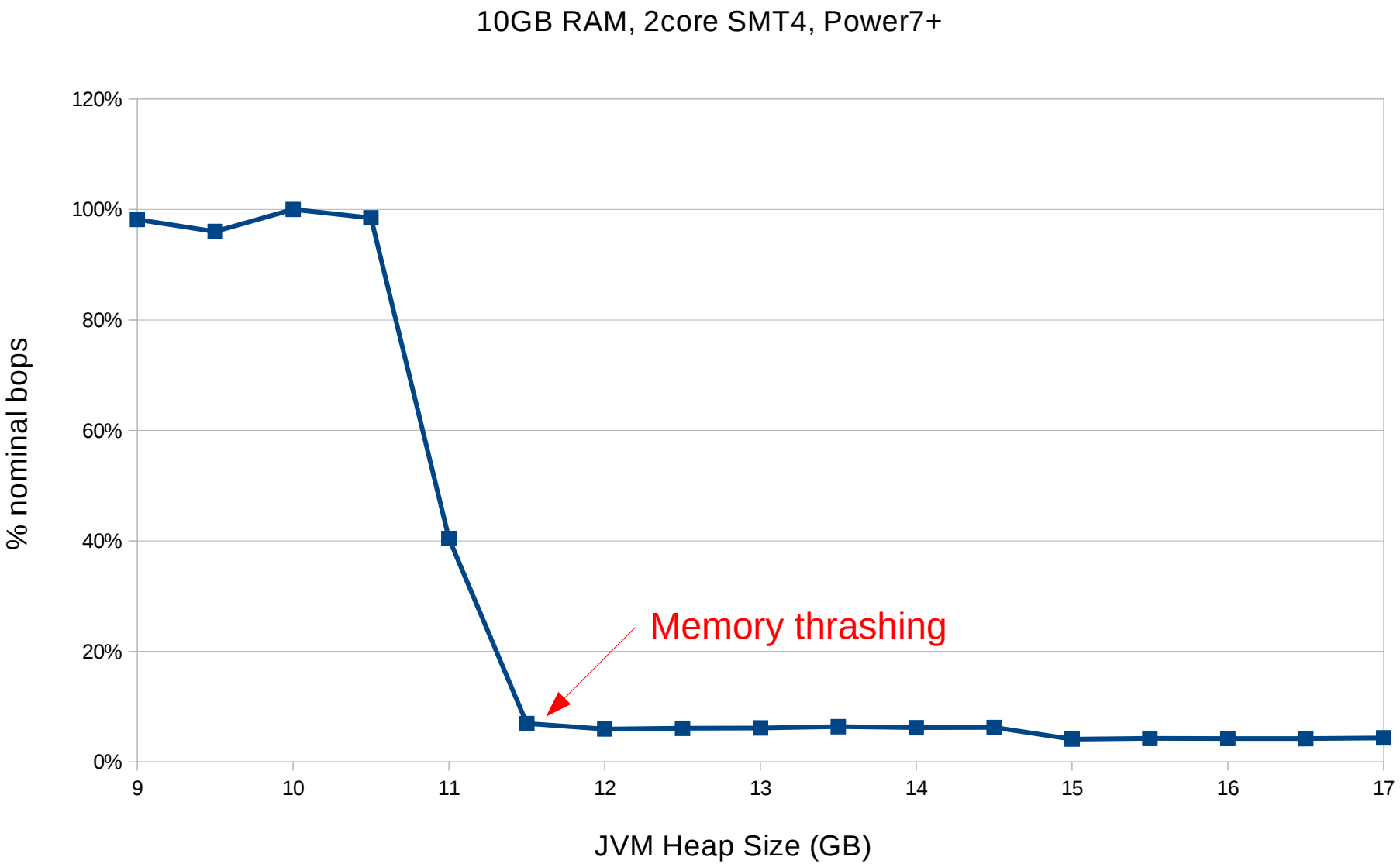
© 2013 IBM Corporation

Swap is a Four Letter Word

- When a CPU bound workload suddenly becomes I/O bound, saturating the swap device and crashing workload throughput and responsiveness

© 2013 IBM Corporation

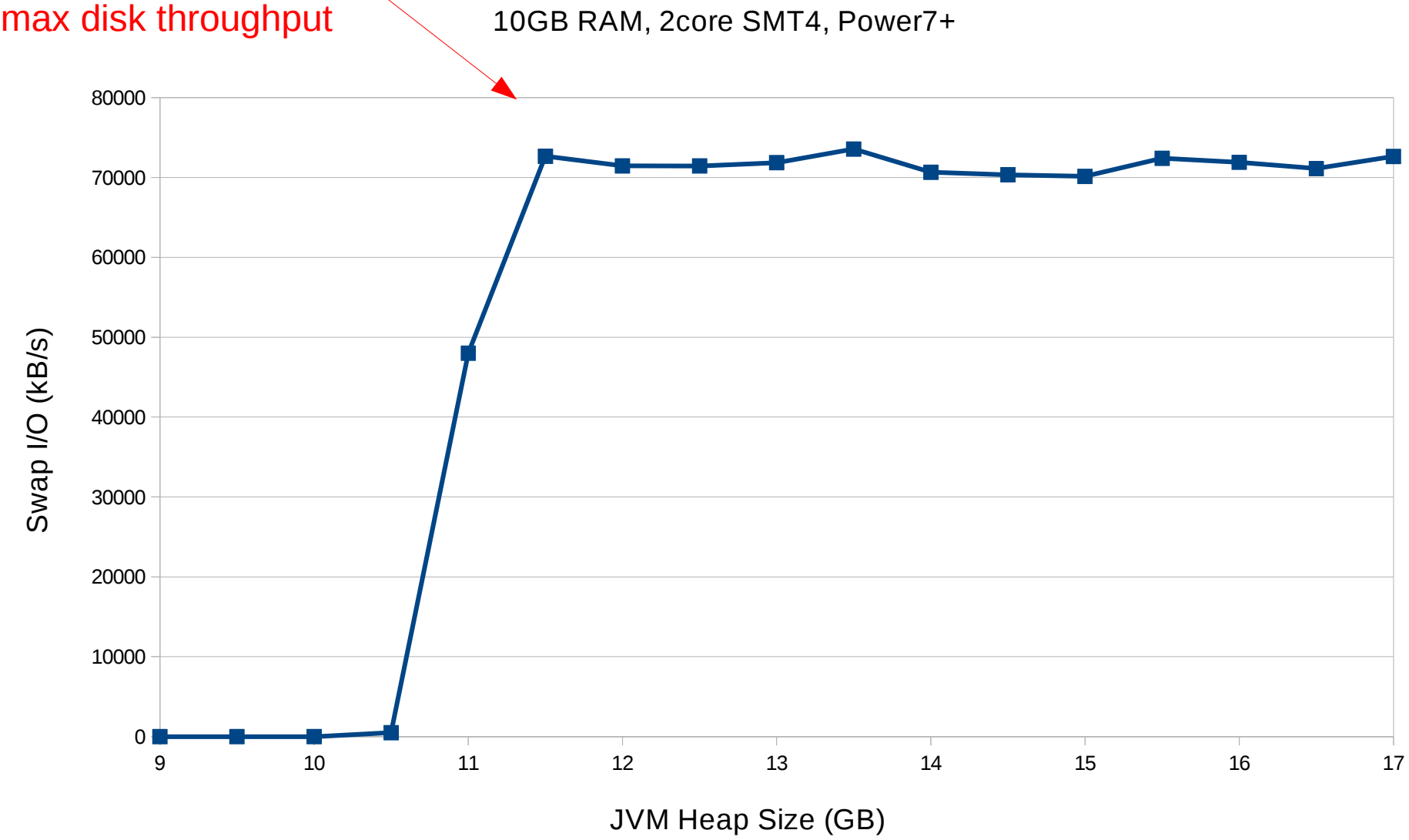
SPECjbb Performance



© 2013 IBM Corporation

System throttles and reaches equilibrium at max disk throughput

Swap I/O



© 2013 IBM Corporation

Swap is Four Letter Word

- In the memory thrashing case, some would prefer to run their systems swap-less and have their workload fall victim to the Out Of Memory (OOM) killer rather than take the non-deterministic latency and performance degradation that I/O introduces
- We need a way to smooth out this I/O storm and performance cliff as memory demand meets memory capacity
- Zswap!

© 2013 IBM Corporation

Crash Course in Memory Reclaim

© 2006 IBM Corporation

Page Frame Management

- All memory is managed in units called “page frames”. This is the basic unit of memory on which architecture memory hardware operates
- Each page frame is managed in the kernel with a page structure (struct page)
- Allocated page frames are maintained on two lists: an “active” list and an “inactive” list

© 2013 IBM Corporation

Page Frame Reclaim

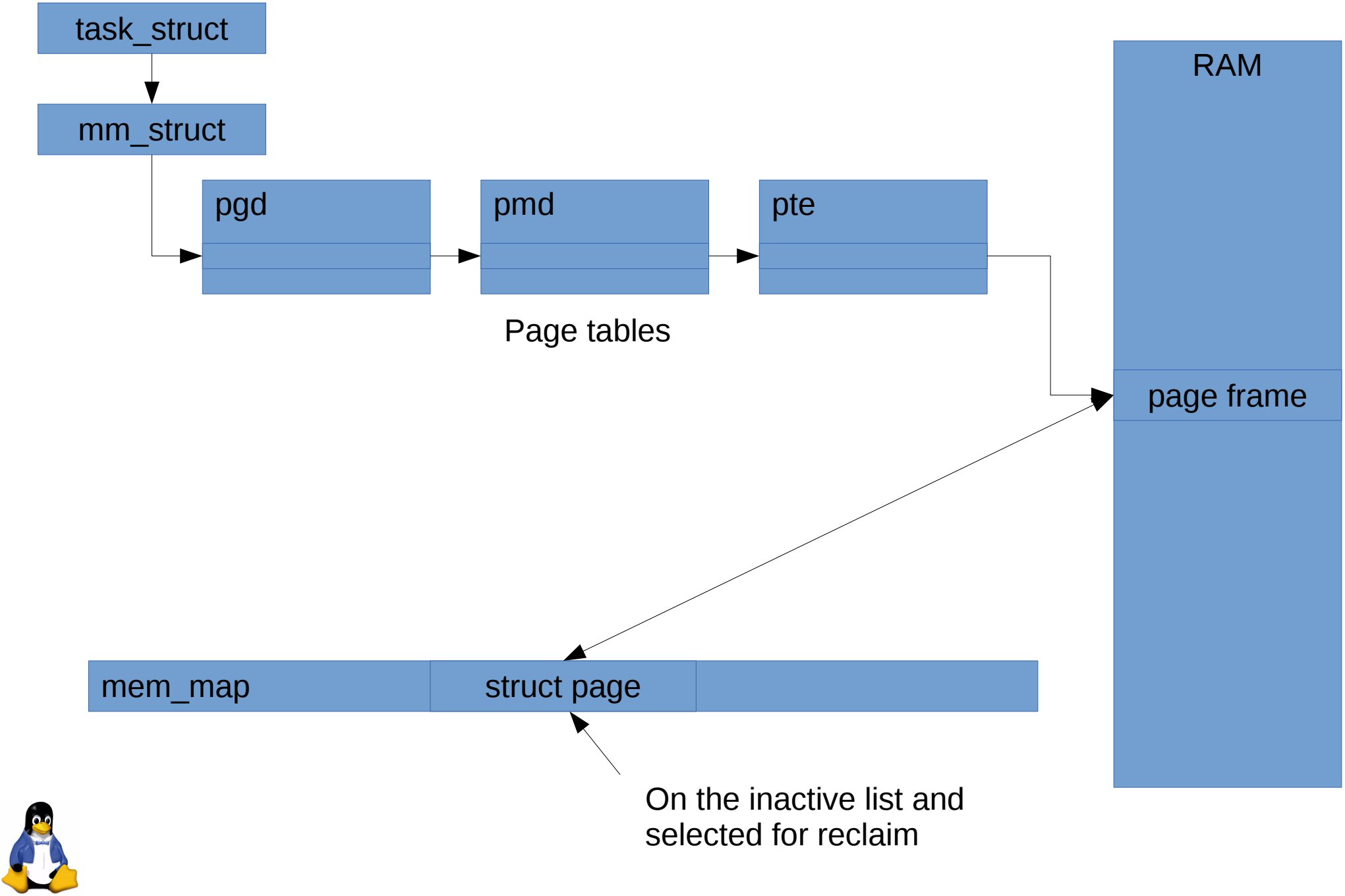
- When the system is low on free page frames, the memory manager begins to search the inactive list for page frames it can reclaim
- Typical page types:
  - Clean page cache, no I/O required (cheap reclaim)
  - Dirty page cache, filesystem I/O needed to clean
  - Anonymous user pages, swap I/O needed to store

© 2013 IBM Corporation

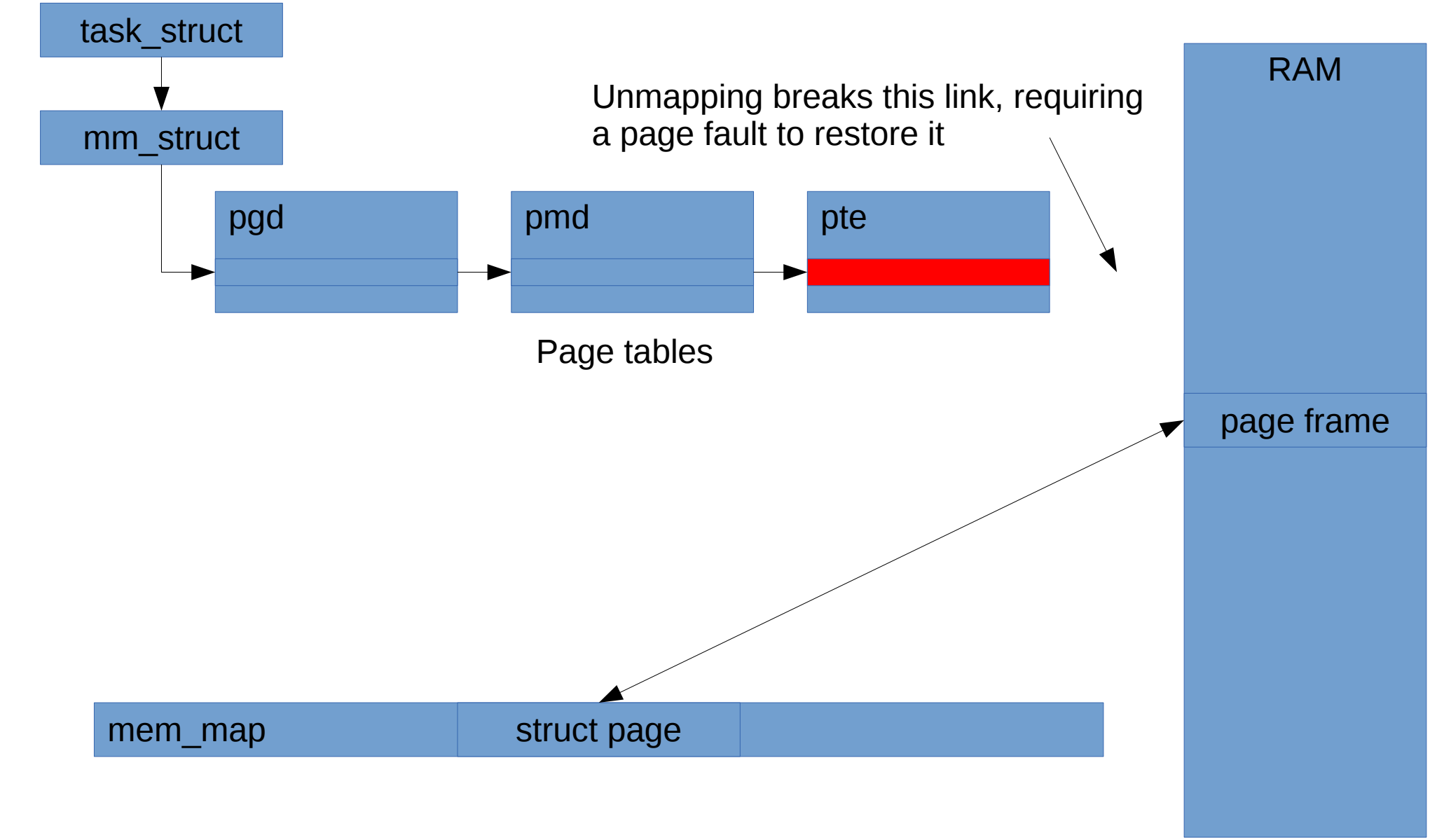
Anonymous Page Reclaim

- The memory manager can reclaim (most) user space anonymous memory through a process called “memory unmapping”
- This process consists of finding all the PTEs that reference the page and replacing those entries with a “swap entry”
- The swap entry contains information about where to find the swapped out page and is used by the page fault handler to repopulate the page if the user process ever accesses it again

© 2013 IBM Corporation

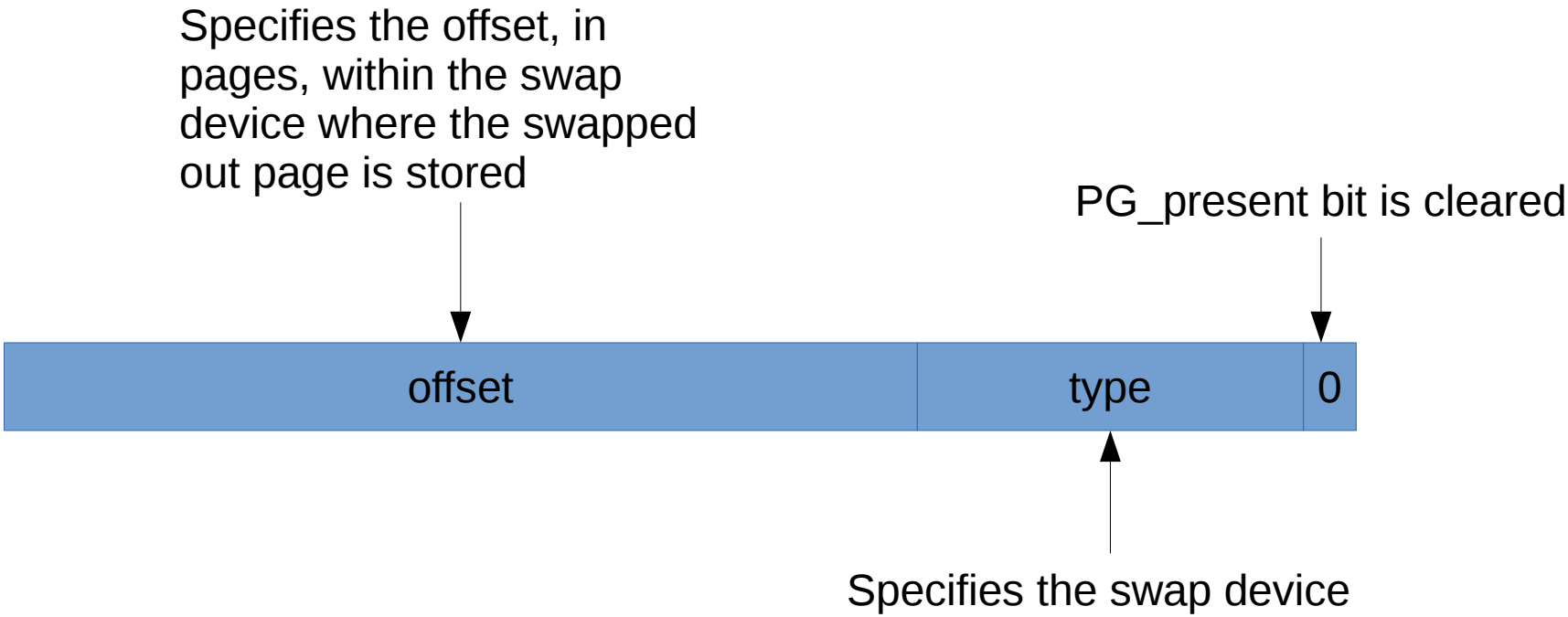


© 2013 IBM Corporation



© 2013 IBM Corporation

Swap Entry



Note that the (type, offset) value is a unique identifier for a swapped out page within the system

© 2013 IBM Corporation

Zswap

Transparent Memory Compression for Swap Pages

© 2006 IBM Corporation

Zswap

- Zswap is feature that hooks into the read and write sides of the swap code and acts as a compressed cache for pages go to and from the swap device
- As a page is being written to the swap device, zswap hooks into the write side (swap_writepage()) via the frontswap API [1] and tries to compress and store the page in a dynamically sized compressed memory pool
- If zswap is successful, the page is clean and ready to be freed without writing the page to the swap device

[1] http://lwn.net/Articles/386090/

© 2013 IBM Corporation

Zswap

- When a page fault occurs on a swap entry, zswap hooks into the read side (swap_readpage()) and attempts to find the swap page, uniquely identified by the swap entry, in the compressed cache
- If the entry is found, the page is decompressed from the cache and linked into the process page tables
- Code at mm/zswap.c

© 2013 IBM Corporation

Zbud

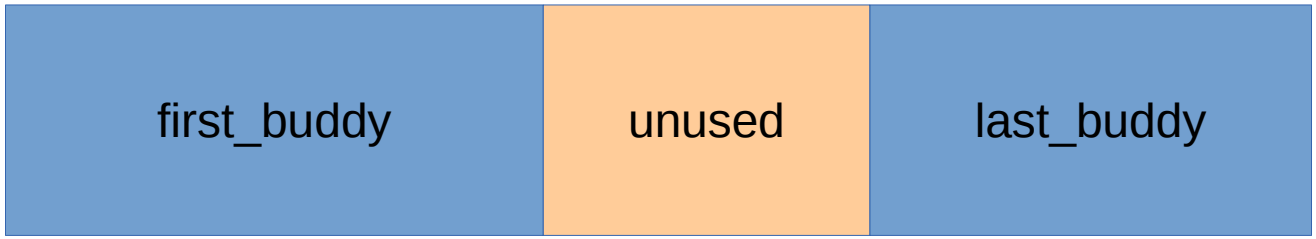
- Zbud is the name of the allocator (more like a bin packer) for the compressed memory pool
- Compressed page data is paired, hence the “bud”, and store in a page frame together
- Only two compressed pages per page frame
  - Caps effective compression at 50% BUT
  - Enables a more deterministic reclaim time
- Hoping to enabled zsmalloc soon
- Code at mm/zbud.c

© 2013 IBM Corporation

Zbud

Page Frame in the pool

Last compressed page data
Is page aligned at the end



First compressed page data
Is page aligned at beginning

© 2013 IBM Corporation

Enabling Zswap

- Zswap must be enabled at boot time with a kernel parameter
  - zswap.enabled=1
- An optional kernel parameter can set the compressor to any compressor enabled in the kernel cryptoraphic API (lzo is the default)
  - zswap.compressor=deflate

© 2013 IBM Corporation

Enabling Zswap

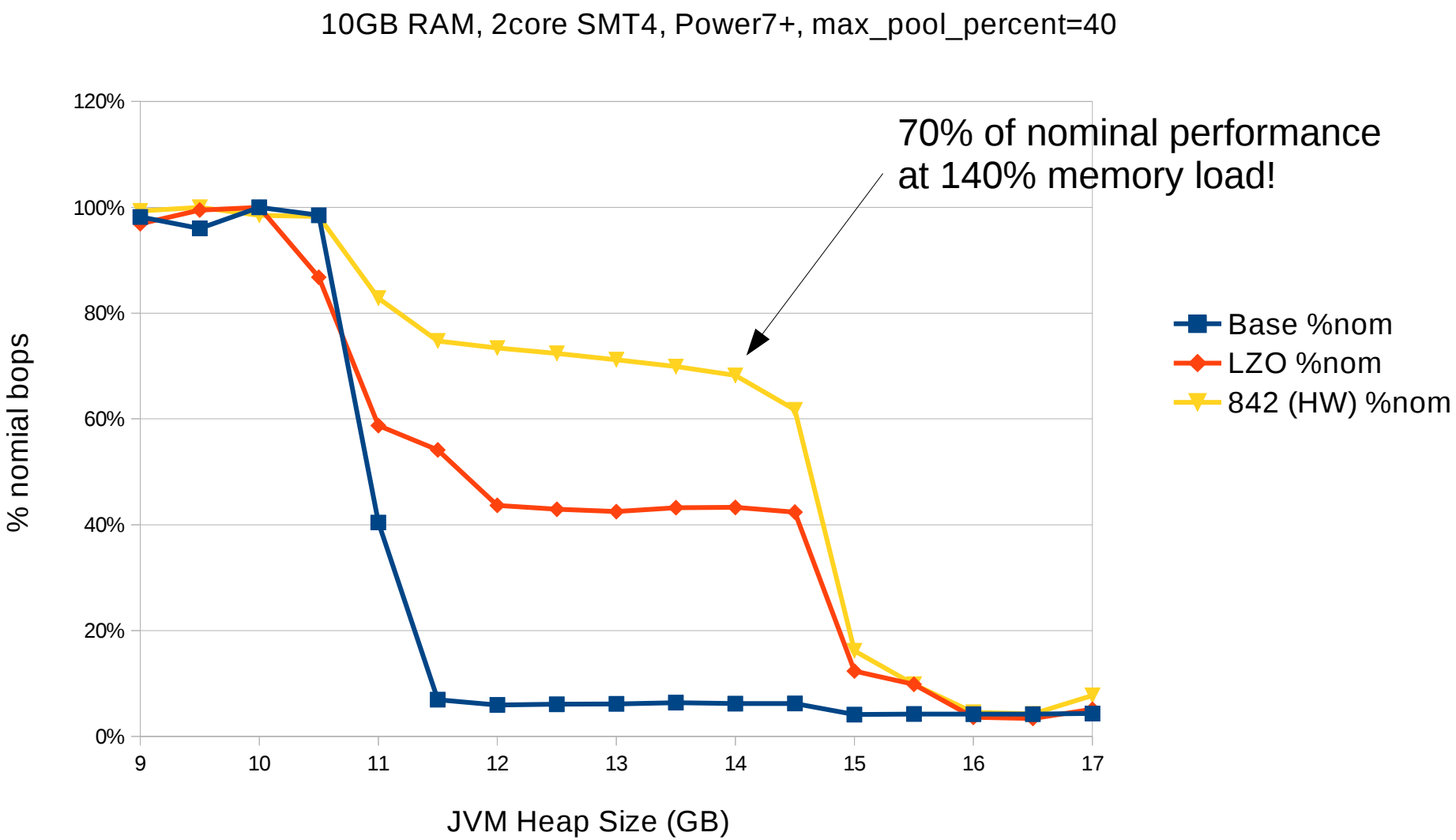
- There is only one tunable, max_pool_percent, that specifies the maximum percentage of RAM that can be used for compressed pool
- /sys/modules/zswap/parameters
- Statistics on zswap activity are available in debugfs
  - /sys/kernel/debug/zswap
  - Frontswap stats can also be relevant

© 2013 IBM Corporation

Enough talk, let's do this!

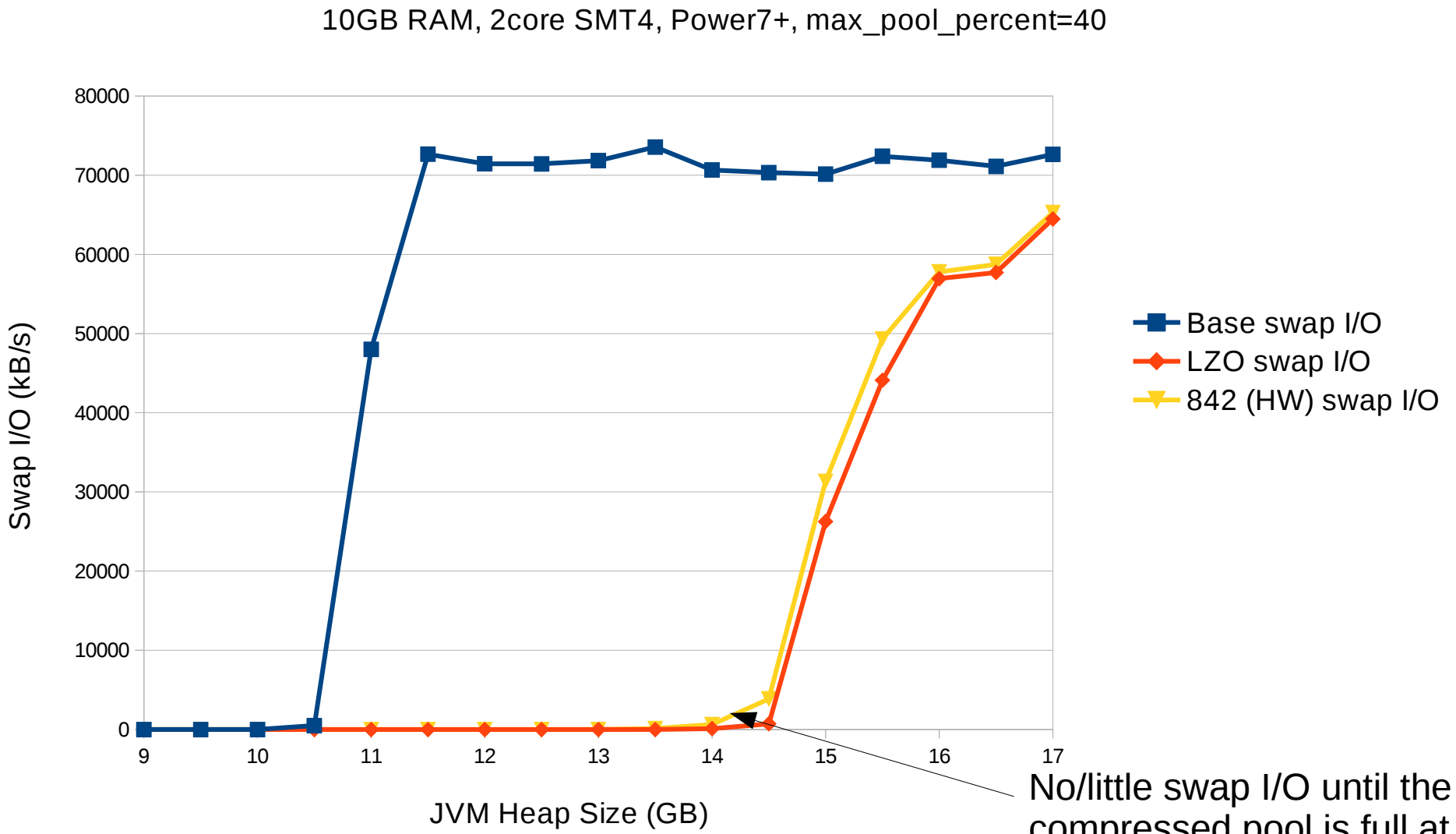
© 2006 IBM Corporation

SPECjbb Performance



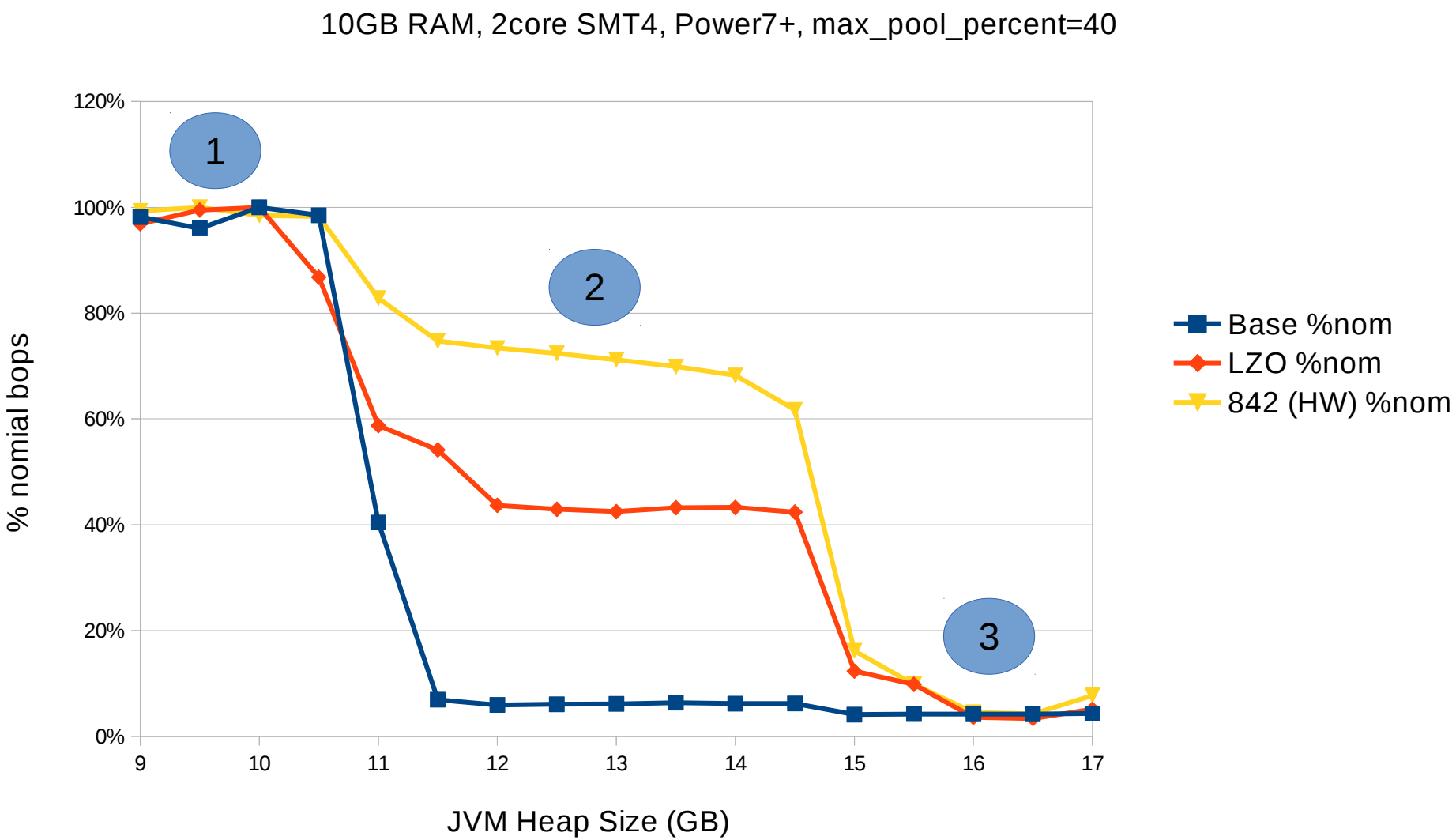
© 2013 IBM Corporation

Swap I/O



© 2013 IBM Corporation

SPECjbb Performance



© 2013 IBM Corporation

Case 1

- Memory is not overcommitted
- No/little swap I/O is occurring
- Zswap is idle

© 2013 IBM Corporation

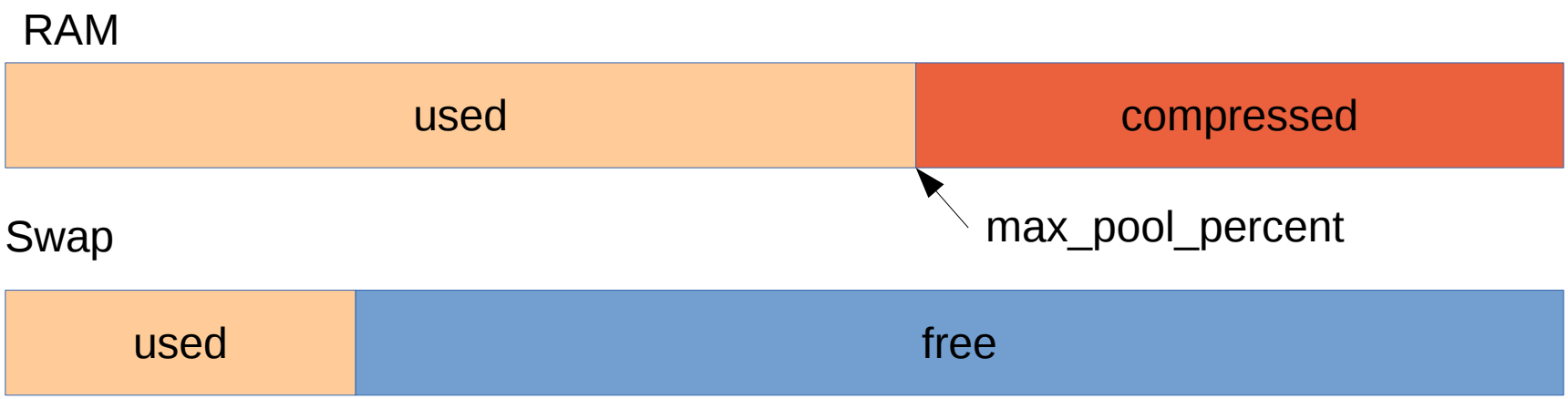
Case 2

- Memory is overcommitted
- Below nominal performance due to memory scanning, unmapping, compression, and decompression
- No/little swap I/O is occurring

© 2013 IBM Corporation

Case 3

- Memory is overcommitted and compressed pool is full
- Zswap begins decompressing the oldest pages and writing them to the swap device
- Converges on non-zswap performance and swap I/O



© 2013 IBM Corporation

Policy Design

- Pool allocation
  - Static, pre-allocated
  - Dynamic, grows under pressure
- Pool sizing
  - Managed by some heuristic in the MM
  - Controlled by the user via a tunable
- Pool overflow action
  - Write oldest compressed pages to the swap device
  - Reject cache store; falls to swap device (inverse LRU)

© 2013 IBM Corporation

Uses cases

- IaaS user
  - Pay for less RAM
- IaaS provider
  - Higher guest density
- Fixed memory systems

© 2013 IBM Corporation

Gotchas

- Zswap is a cache layer on top of the swap device, not the swap device itself
- The cache can not hold more pages than can fit in the swap device

© 2013 IBM Corporation

Other/Future Work in Compressed Memory

© 2006 IBM Corporation

zram

- Zram is a driver currently in the drivers staging tree that acts as a compressed ram disk that can be used directly as a swap device
- Differs from zswap in that zram is the swap device rather than a caching layer on top of the swap device
- Zram is preferred by embedded application that do not have an actual swap device
- Minchan Kim is heading up the mainlining effort

© 2013 IBM Corporation

zcache

- Zcache has been many things over the years in the driver staging tree
- Has now been dropped from staging
- Bob Lui has stripped it down to only the page cache compression elements and is trying to get it accepted directly
  - Would be the first non-Xen user of cleancache API

© 2013 IBM Corporation

zcache

- Page cache compression has a unique challenge over swap compression
- A swap cache store always avoids a write where as a file cache store may avoid a (re)read
- Better heuristics and feedback mechanisms are needed to determine whether or not compressing a particular page cache page is worth it
- We don't want to be compressing page cache page just to end up throwing them away

© 2013 IBM Corporation

Future

- This is a new field for memory management with lots of use cases, mechanisms, and best policies yet to be determined
- Zswap is just a first step
- Move to “page addressable memory” concept model
  - Swap
  - High memory
- Managed and reclaimed with other memory types

© 2013 IBM Corporation

Summary

- Zswap is in mainline for v3.11
  - Off by default
  - Experimental until properly vetted on a variety of systems and workloads
- Allows for RAM sizings closer to the average working set than the peak or increased workload size
- Compressed memory acts like a safety net that won't complete trash your workload and swamp your SAN if you overcommit a little
  - Higher guest density in IaaS setups

© 2013 IBM Corporation

Questions?

© 2006 IBM Corporation