June 2012 @Dubai
IBM Power Academy

IBM PowerVM
memory virtualization

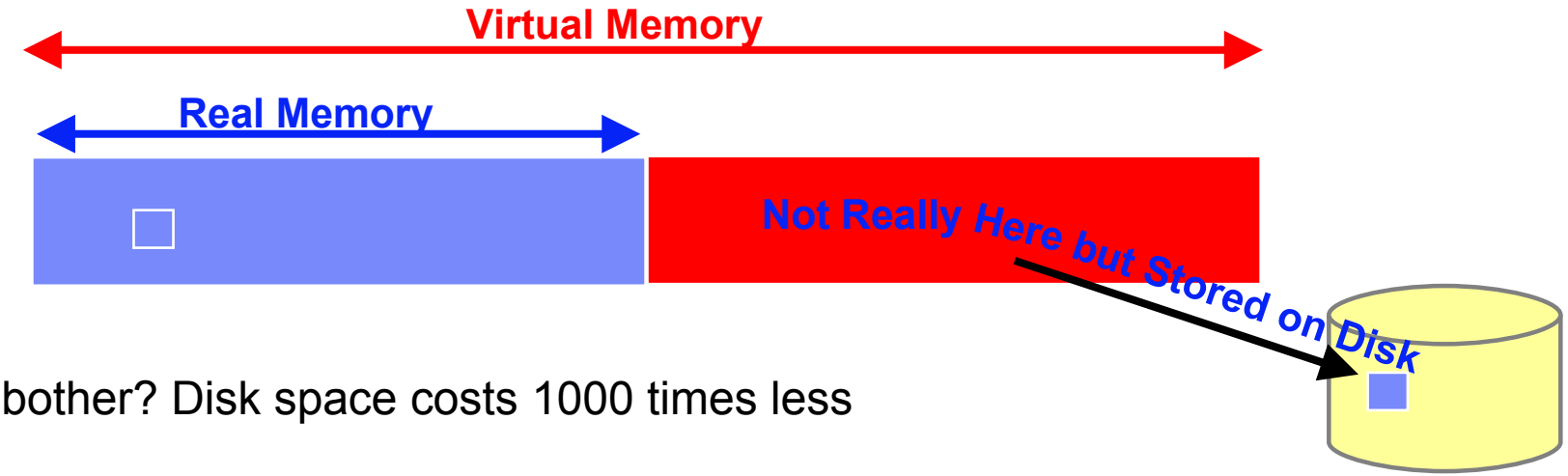
Luca Comparini
STG Lab Services Europe
IBM FR

June,13th 2012
@IBM Dubai

Agenda

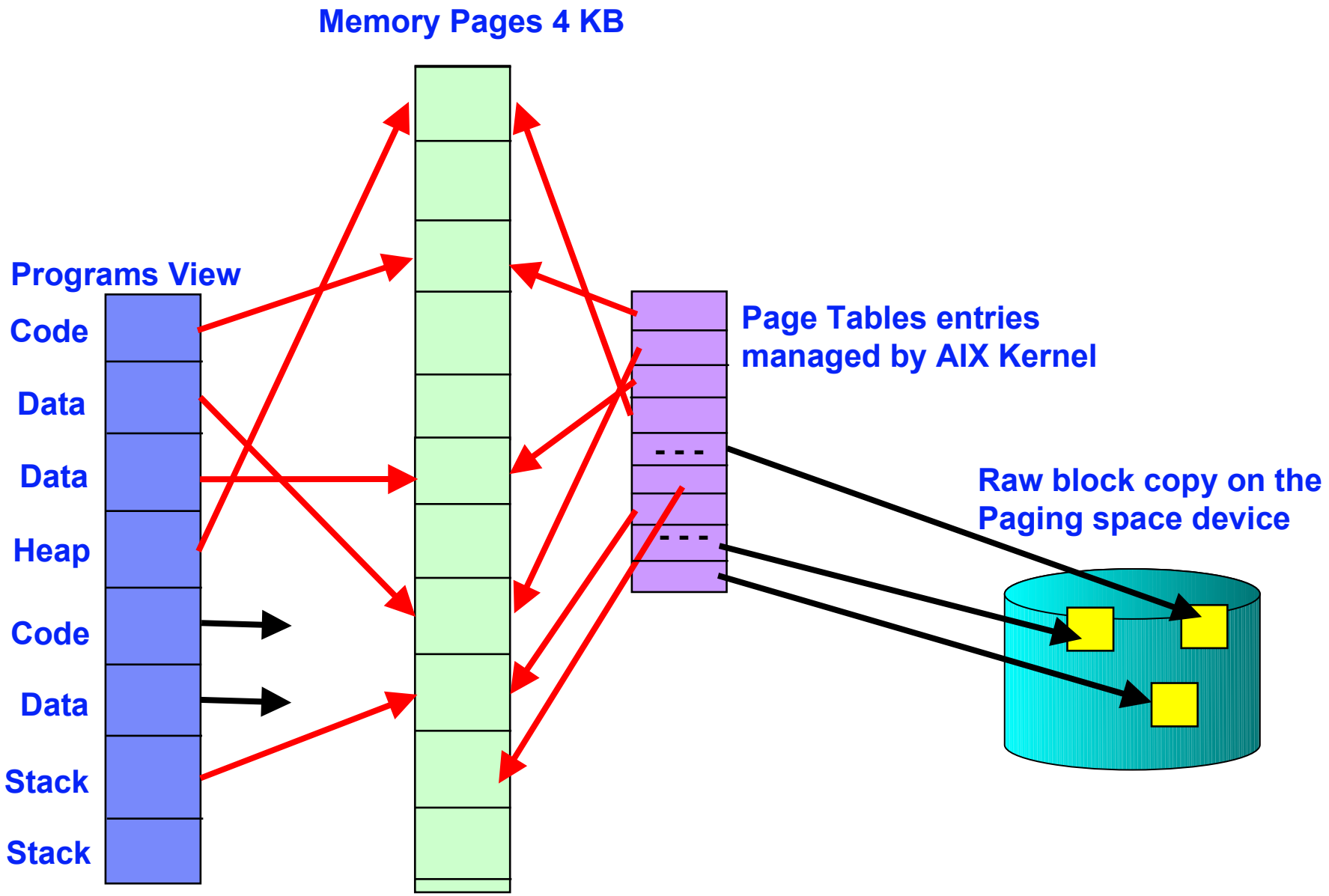
- How paging works
- Active Memory Sharing
- Active Memory Expansion
- Active Memory Deduplication

What is virtual memory

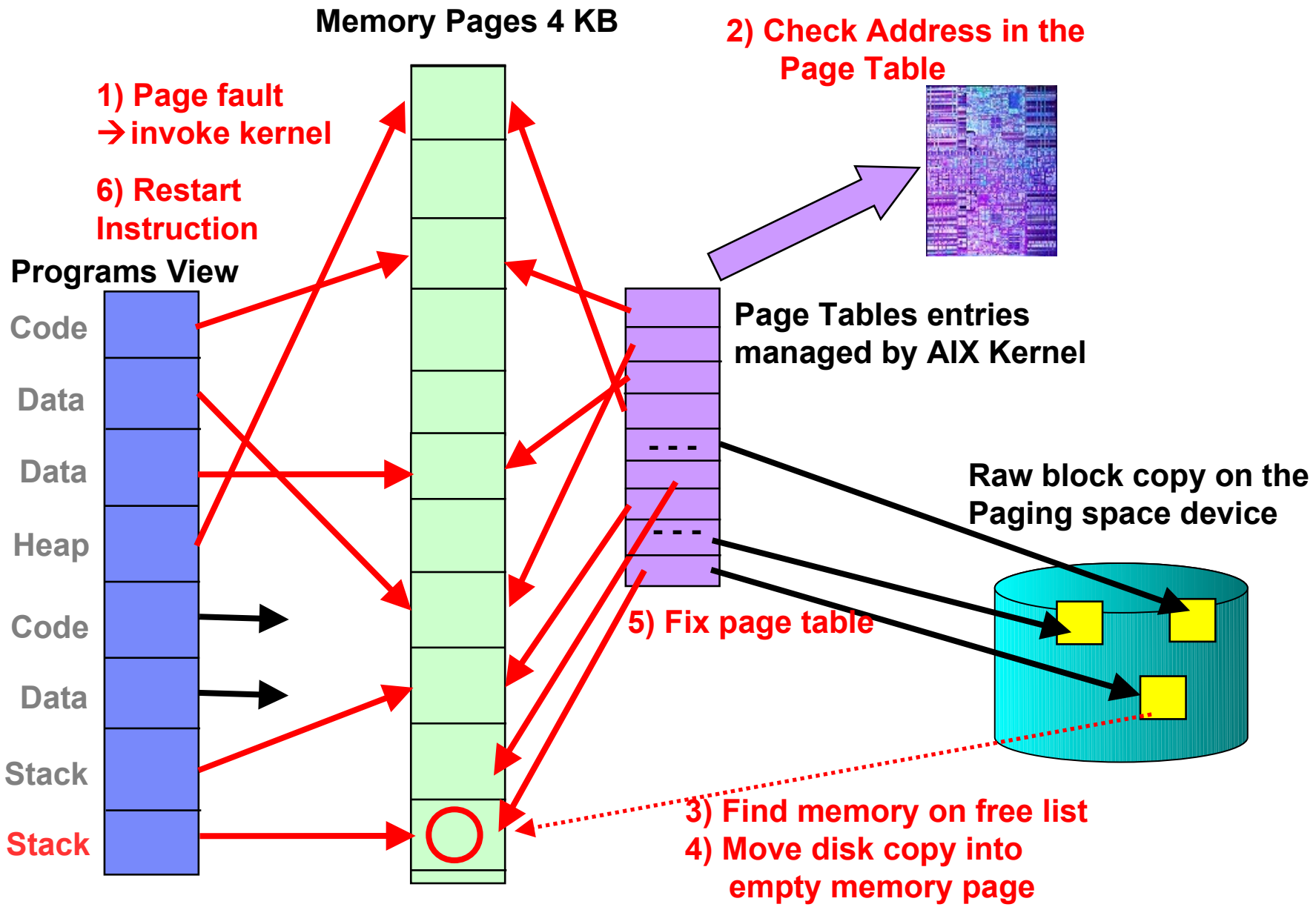


Why bother? Disk space costs 1000 times less

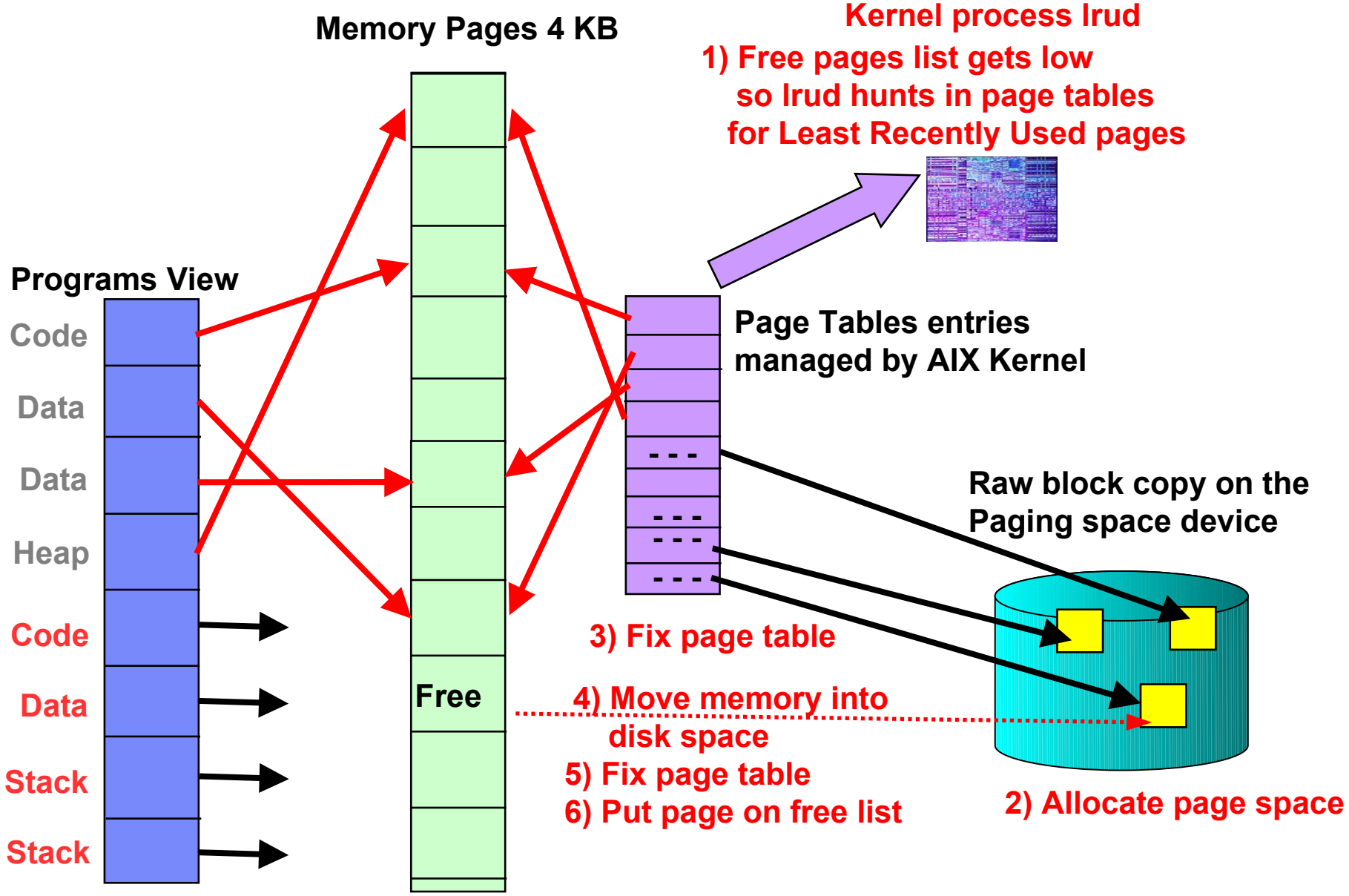
Paging


Memory Pages 4 KB
Programs View
Code
Data
Data
Heap
Code
Data
Stack
Stack
Page Tables entries
managed by AIX Kernel
Raw block copy on the
Paging space device

Paging in


Memory Pages 4 KB
2) Check Address in the Page Table
1) Page fault
→invoke kernel
6) Restart Instruction
Programs View
Code
Data
Data
Heap
Code
Data
Stack
Stack
Page Tables entries managed by AIX Kernel
Raw block copy on the Paging space device
5) Fix page table
3) Find memory on free list
4) Move disk copy into empty memory page

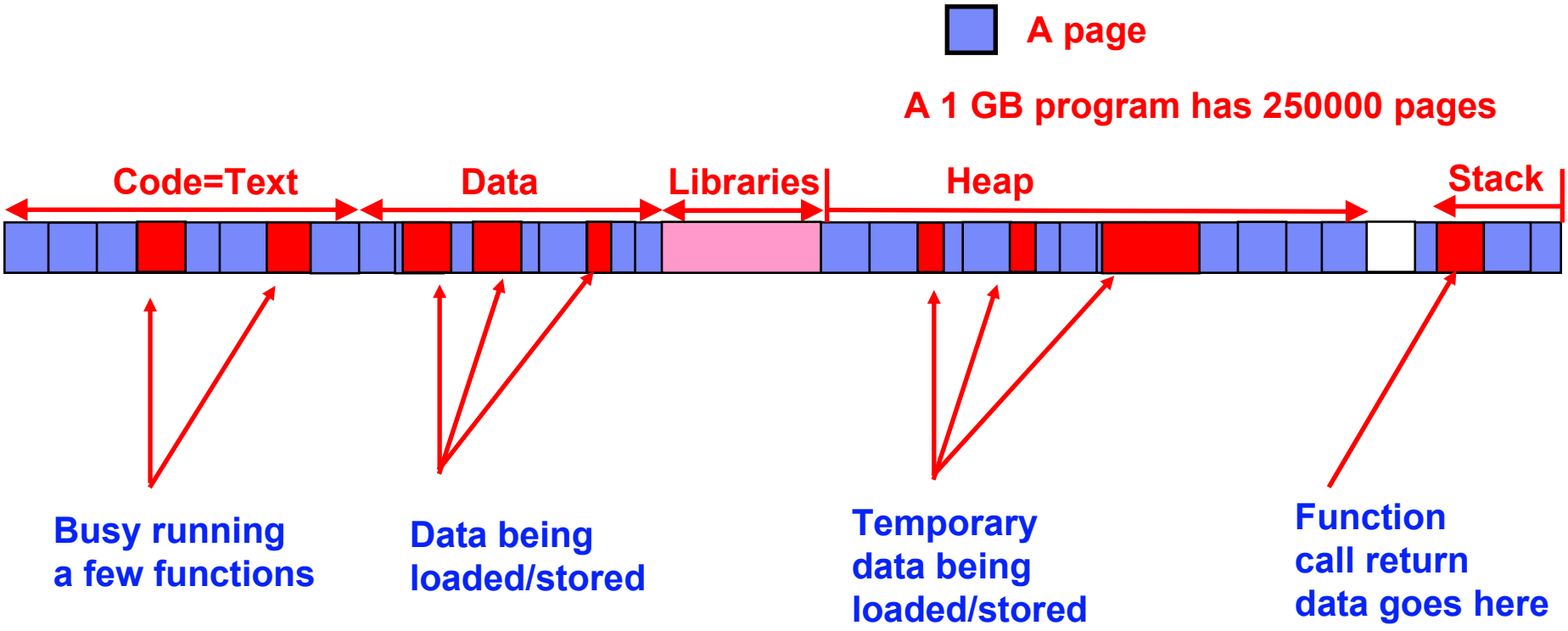
Paging out


Memory Pages 4 KB
Kernel process lrud
1) Free pages list gets low
so lrud hunts in page tables
for Least Recently Used pages
Programs View
Code
Data
Data
Heap
Code
Data
Stack
Stack
Free
Page Tables entries
managed by AIX Kernel
Raw block copy on the
Paging space device
3) Fix page table
4) Move memory into
disk space
5) Fix page table
6) Put page on free list
2) Allocate page space

Five Paging Golden rules

1. Don't do it! → hurst performance
2. Don't panic! → 10 pages/s per CPU = noise
3. Do it fast → use many disks
4. Always use Protection → mirror or RAID5
5. Never ever run out of paging space

What is a Working set



Working set is the pages needed to run in the short term (within seconds)

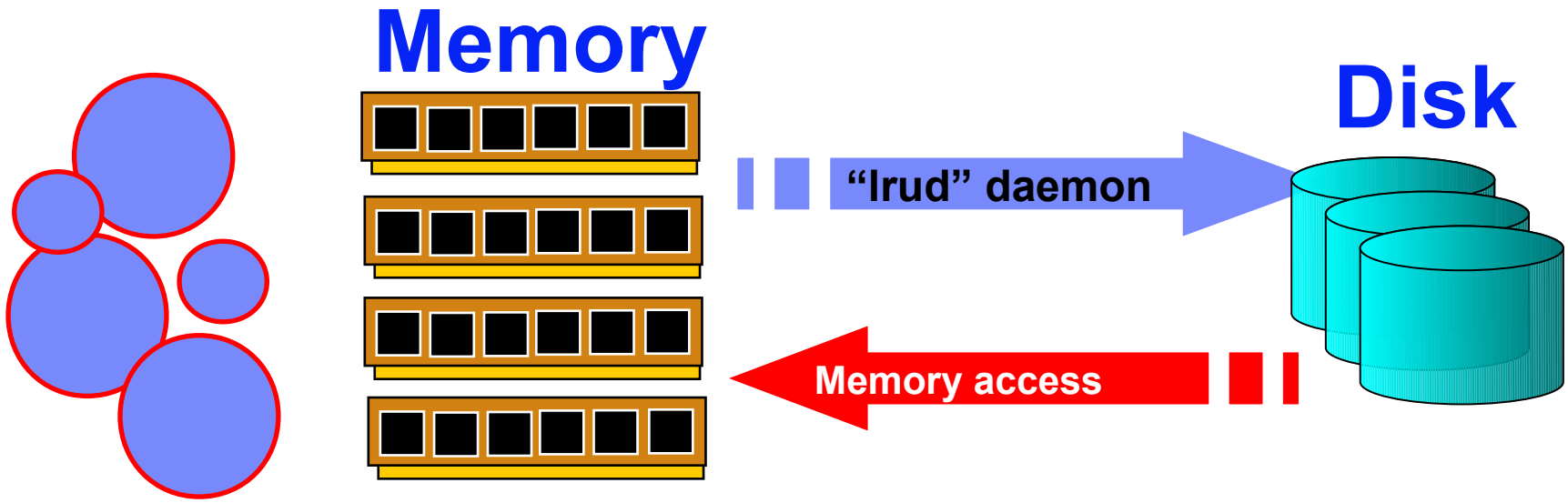
Also called resident set (resident in memory):
→ see ps or nmon ResText & ResData

Active Memory Sharing works on Working Sets but at a whole LPAR level

AIX level paging

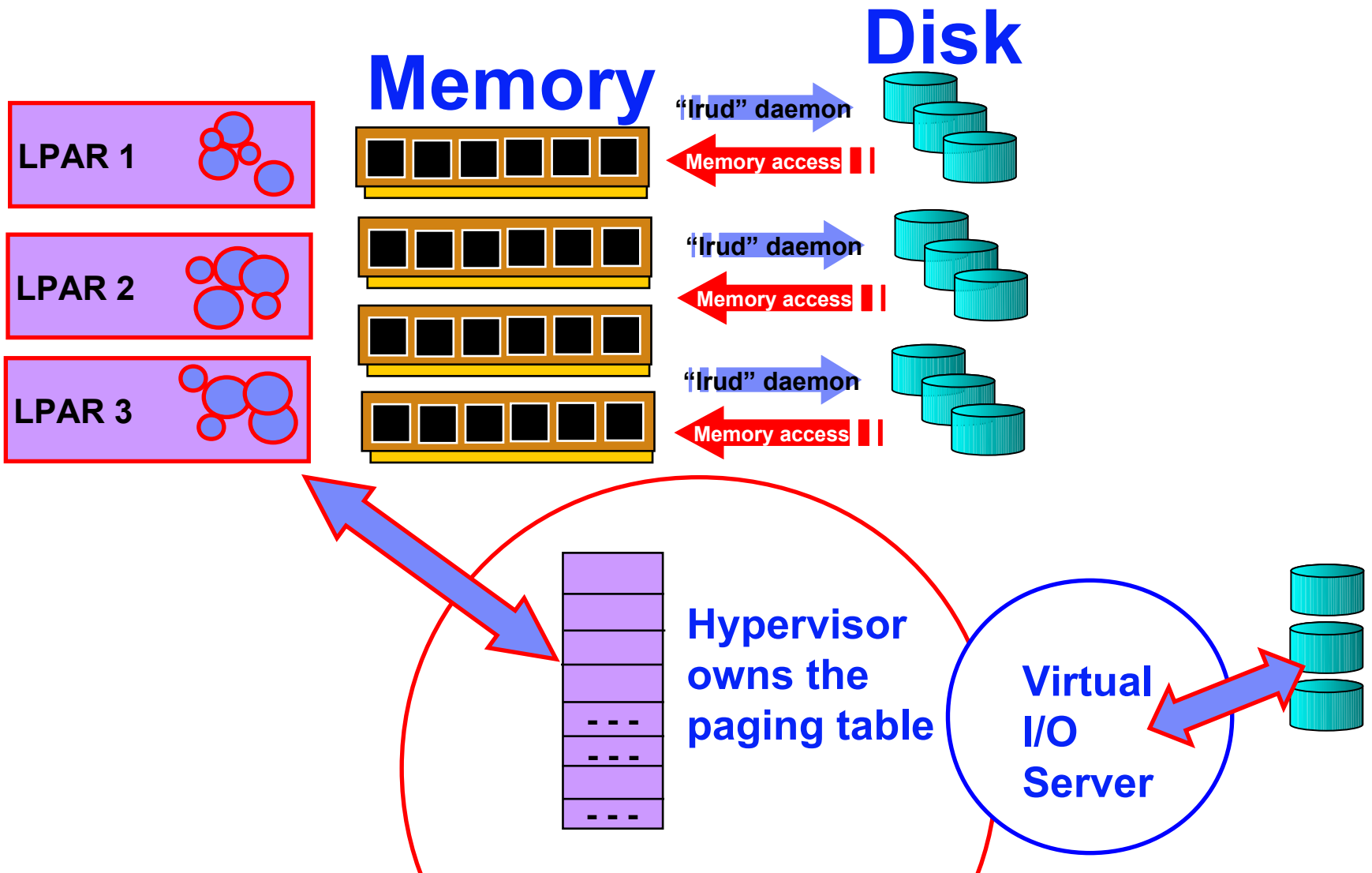
Page in = after a page fault, get raw disk block into memory

Page out = lrud daemon frees page space



Active Memory Sharing

Active Memory Sharing = LPAR level paging



When AMS can help?

1. You have 100 “standard template” LPARs, you need to create LPAR101:
   you have spare shared CPU
   you don't have spare memory

2. You may want to squeeze (i.e.) 280GB of RAM (requiring big expensive dimms) into (i.e.) 256GB (small dimms, cheaper and faster).

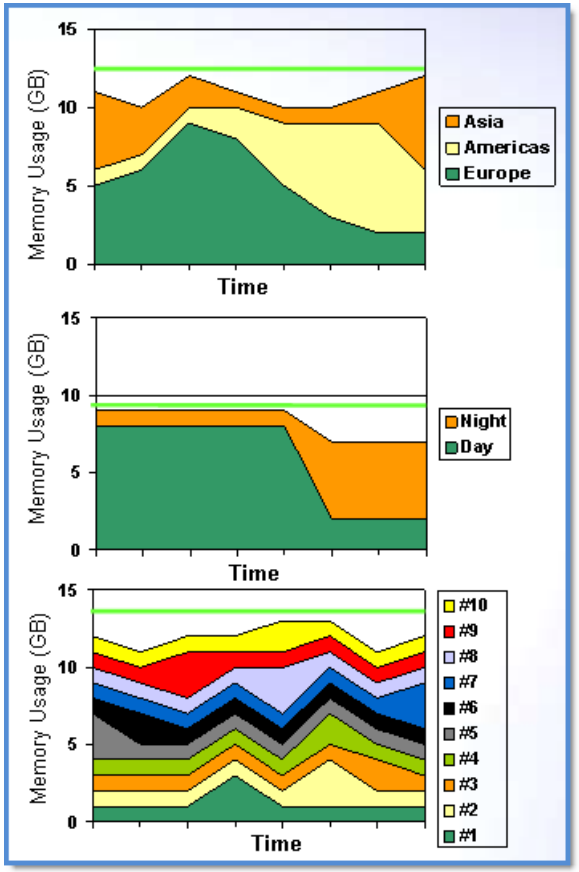
3. Share memory with LPAR:

- around the world (peak at different times)

- day and night (day time web app, night time batch)

- infrequent use

- failover ready partition
(like day and night, but never actually happens)



AMS prerequisites

1. POWER6 only
2. Firmware 342*
3. HMC 7.3.4 sp2*
4. VIOS 2.1.1*
5. AIX 6.1 TL03* → No AIX 5.3 support
6. PowerVM - Enterprise Edition
   - Extra VET activation code for installed machines
7. No 16 MB pages (used by some HPC codes)
8. Shared CPU LPAR only
9. Shared I/O i.e. Pure Virtual I/O LPARs
10. Also supported → SLES 11, (RHEL 6 later) & IBM i 6.1 (plus PTF)

AMS – How to set it up?

Shared Memory Pool

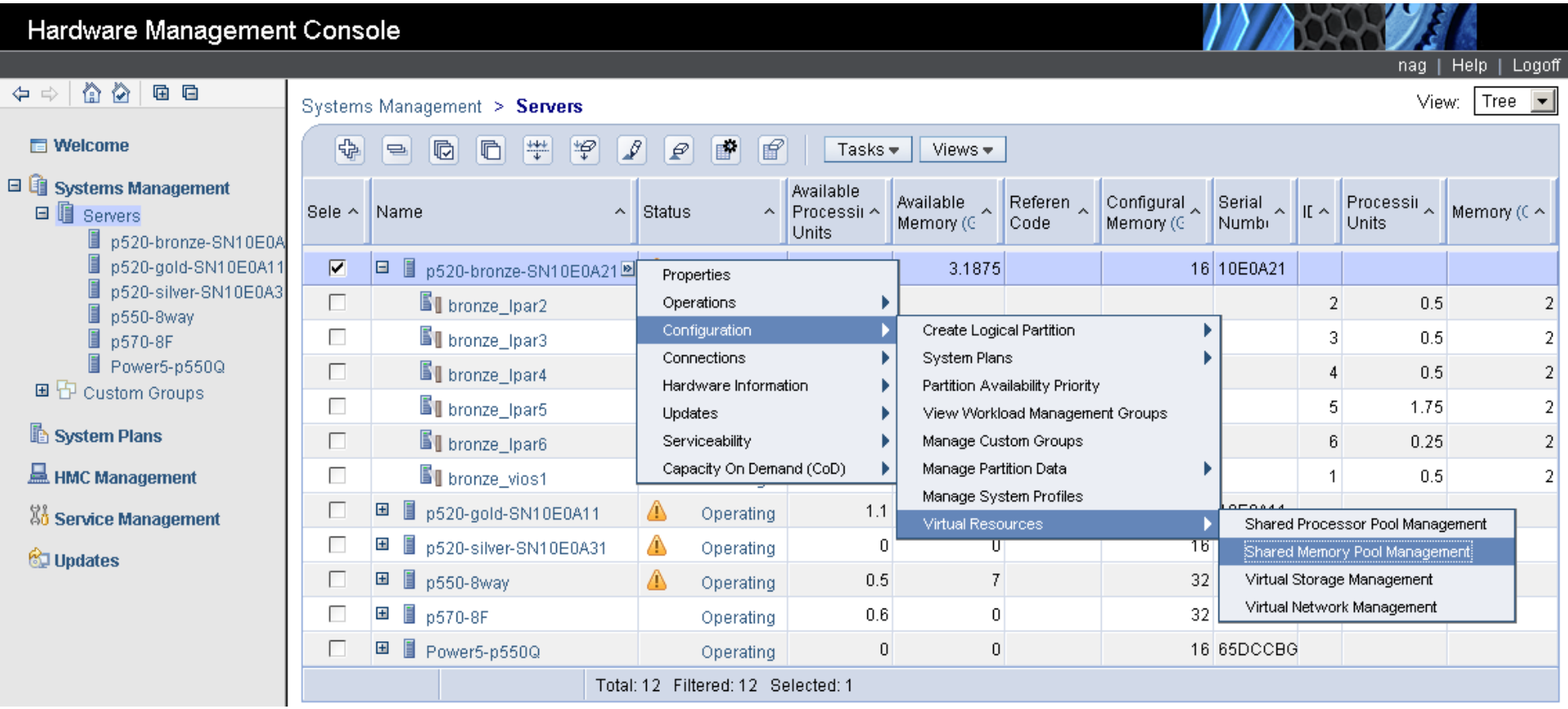
- Only one pool

1. Decide which VIOS to use
2. On VIOS: create AMS paging devices

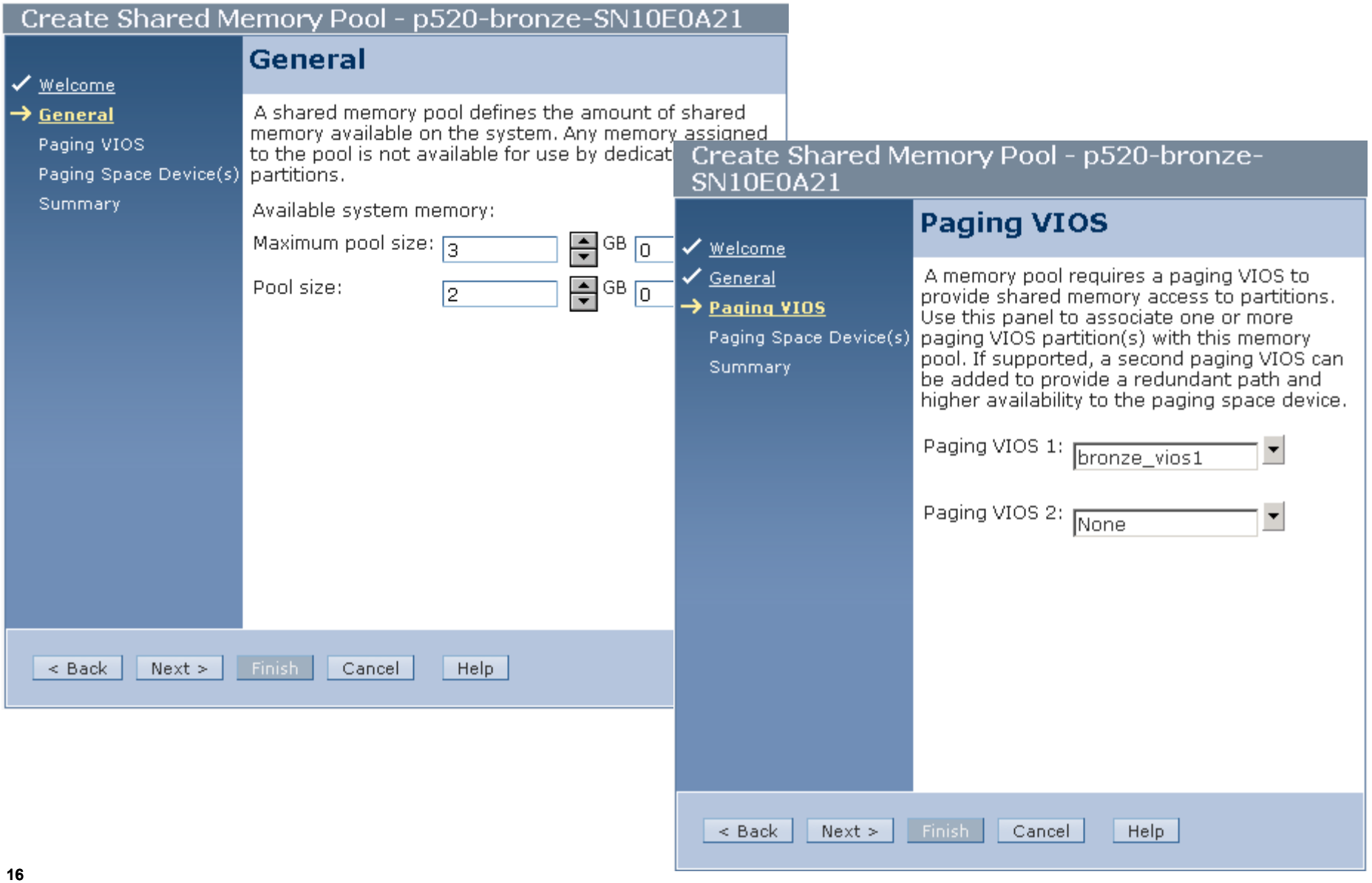
On HMC (or IVM) Creating the pool

3. Pool size
4. Pool maximum size (sanity check for dynamic change)
5. VIOS to use for AMS paging
6. Select AMS paging spaces

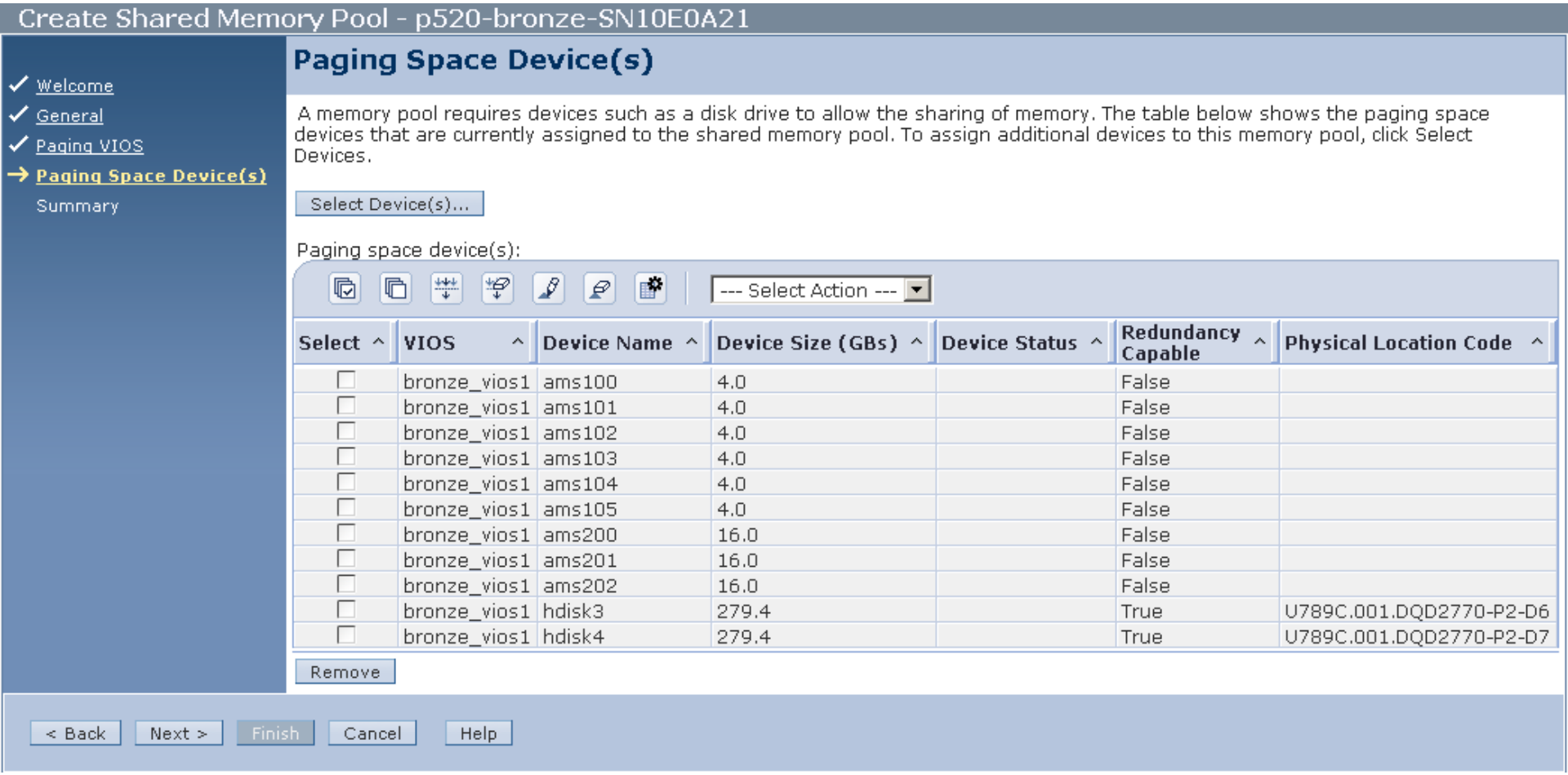
AMS – How to set it up? - Machine Level – Memory Pool



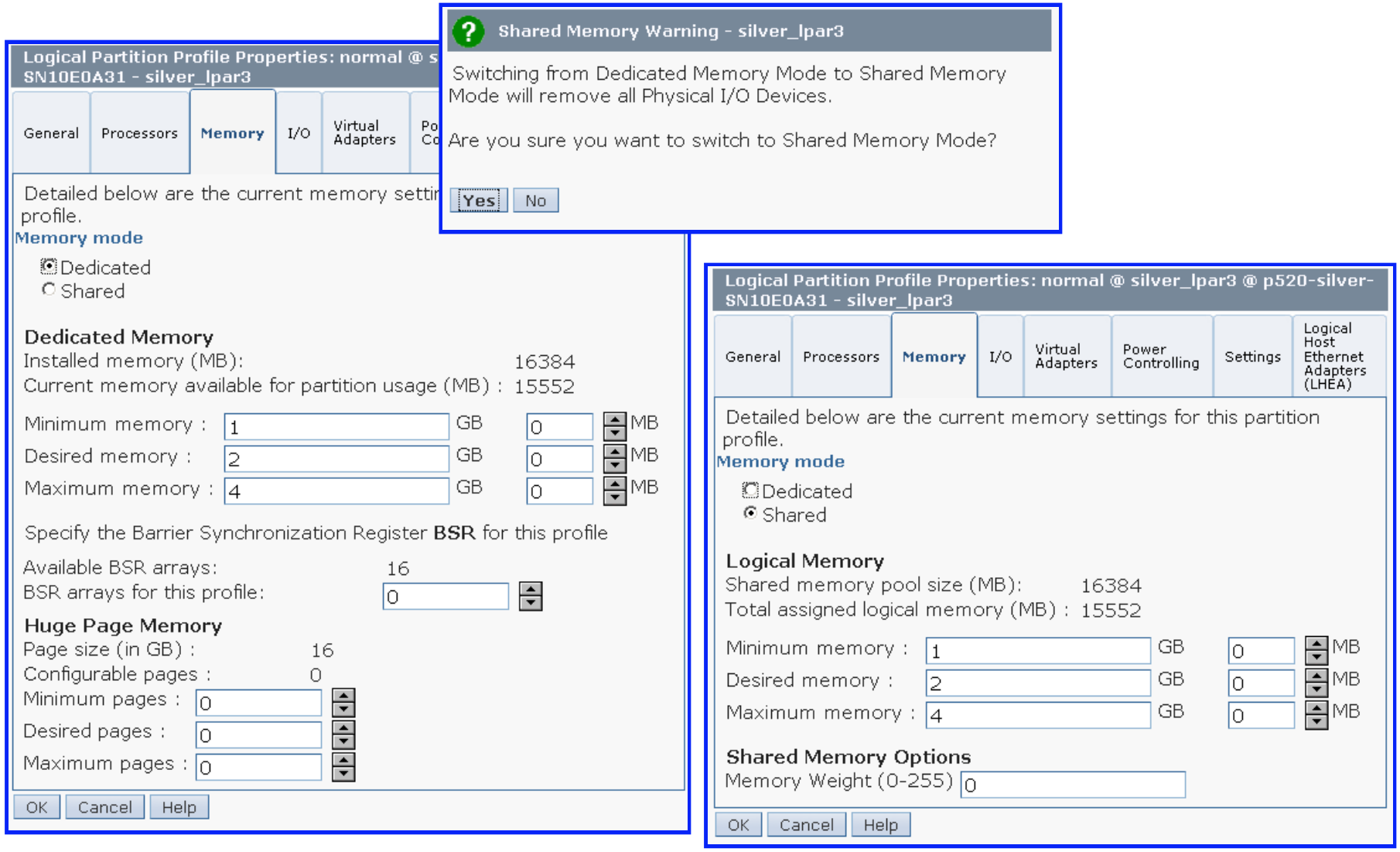
AMS – How to set it up? - Machine Level – Memory Pool



AMS – How to set it up? - Machine Level – Memory Pool



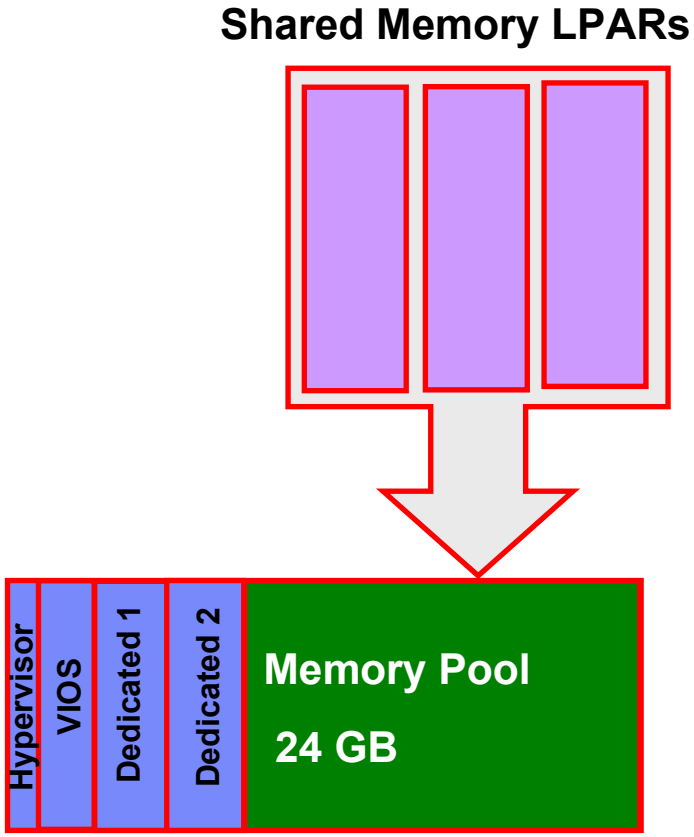
AMS – How to set it up? - LPAR Level



Active Memory Sharing – Use Cases

3 LPARs x 8GB → it all fits (local paging at AIX level)

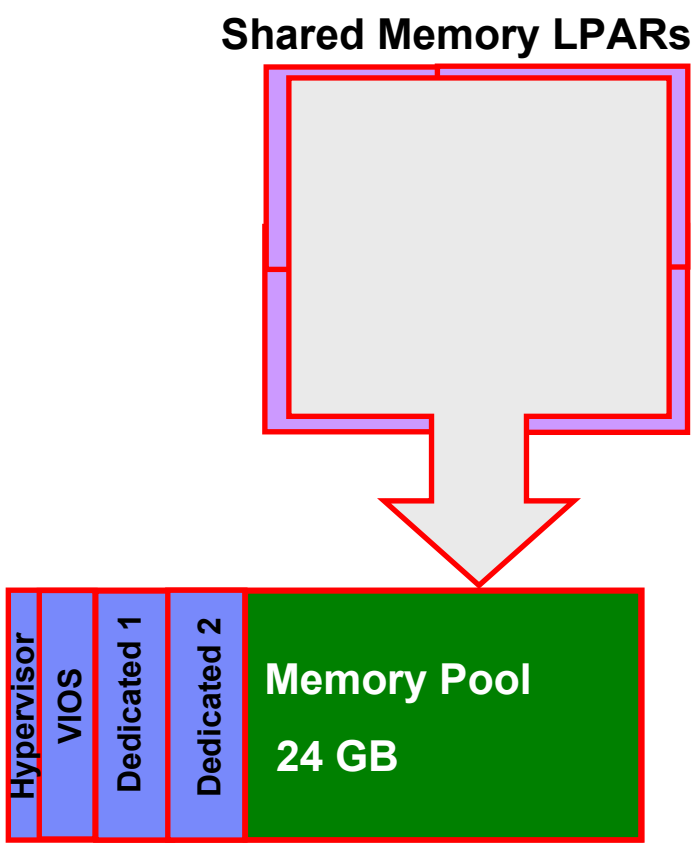
→ AMS in relaxed mode (it does nothing)



Active Memory Sharing – Use Cases: if it nearly fits?

If Resident size ~ 24GB: it works → cooperative mode:

hypervisor asks AIX LPARs for help once a second.



AIX then frees memory, if necessary paging out

Loans pages to hypervisor

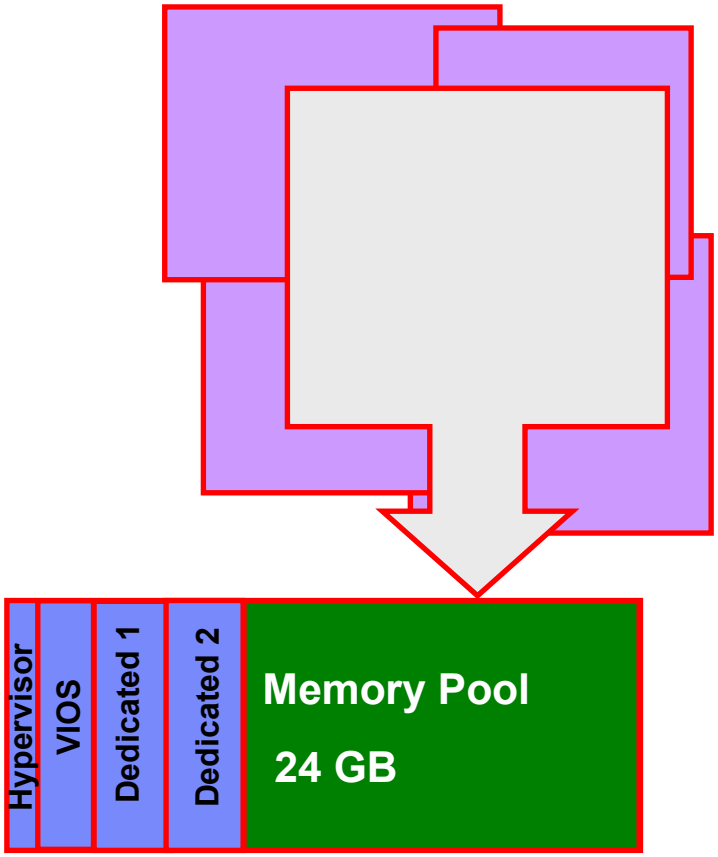
Hypervisor give pages to high demanding LPAR

AIX level AMS tuning on how aggressive

Active Memory Sharing – Use Cases

If Resident size > 24GB: paging

If Resident size >> 24GB: paging++



LPARS refuse to loan more memory

Hypervisor gets aggressive:

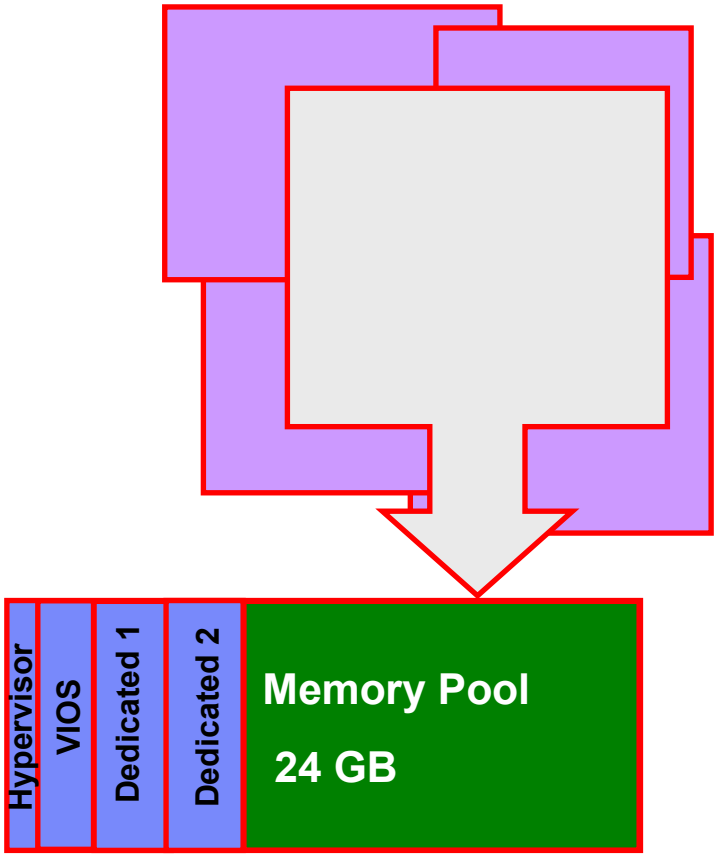
- steal some pages
  - it can see the page tablesµ
  - it avoids critical memory pages
  - use Least Recently Used pages
- asks VIOS to page out LPAR mem
- once the memory is free
- gives pages to high demanding LPAR

LPARs are not aware of this happening

Active Memory Sharing – Use Cases

If Resident size > 24GB: paging

If Resident size >> 24GB: paging++



Now LPAR accesses a page that is not present:

- causes page fault

- Hypervisor hands interrupt to the LPARs to handle

- Checks: if it is an Hypervisor paged page

- if yes: it recovers the page and restart the instruction

- if no: it passes the page fault onto AIX to handle as normal

Active Memory Sharing – The ugly but obvious

High, sustained memory residency requirements

- High Performance – HPC
- RDBMS wth fixed size disk block cache
    - doesn't page but uses 95% of memory

Where paging is “not an option” anyway

- Real time
- Response time or predictable sensitive

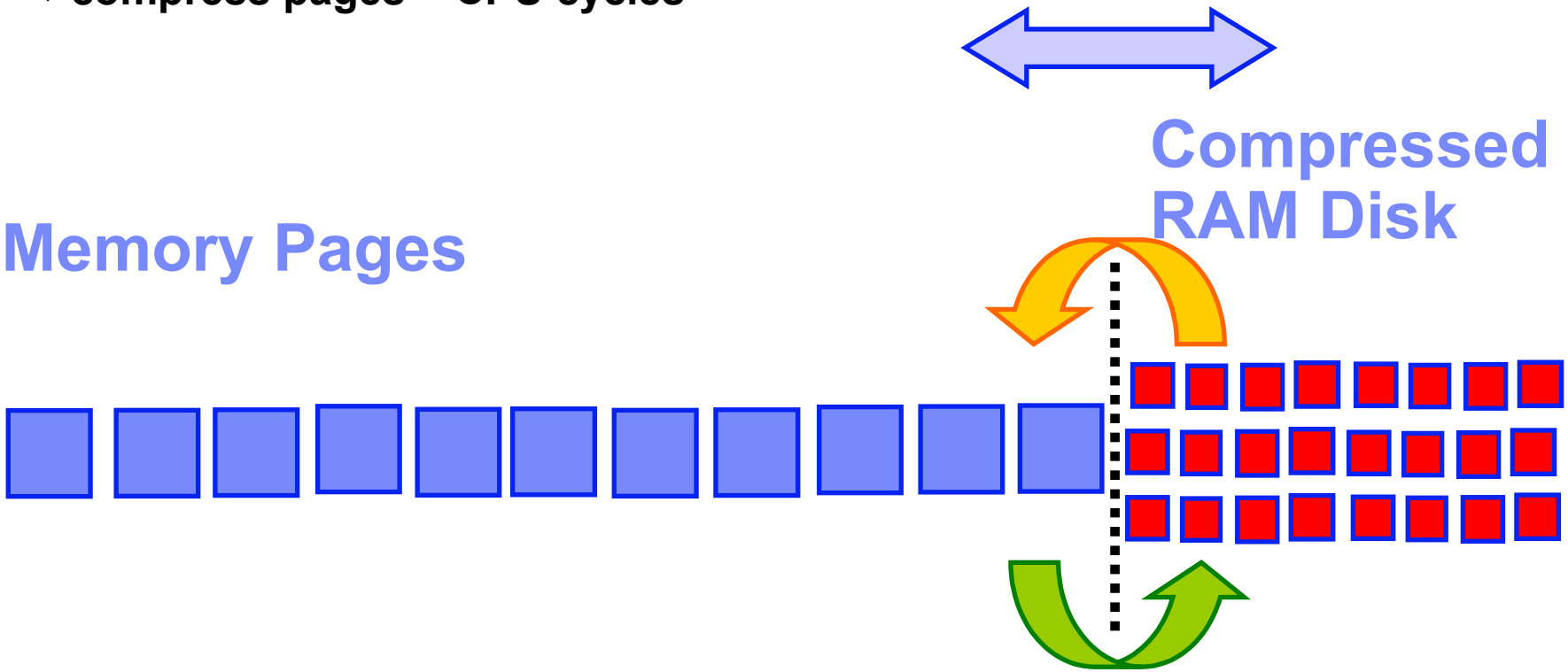
Active Memory Expansion

Active Memory Expansion – conceptual model

Conceptually, split memory in real memory (blue) and a RAM disk like part (red)

- Use the red part as a very fast paging space
- While paging, shrink the memory pages so many more pages fit
- Dynamically adjusted depending on compression rate & target

→ **compress pages = CPU cycles**



Active Memory Expansion – bad compression targets

- AIX Kernel
  - Not a AME target
- Filesystem cache, code or memory mapped files
  - Best to page out to filesystems
  - Performance tools → “numperm”
- Pinned Memory
  - Pinned = never page out (AME is like paging)
  - Performance tools → “pinned pages”
- So what can AME compress?

Active Memory Expansion – good compression targets

Mostly private pages within programs

- Data
- Heap
- Stack
- Not the code

Data that compresses well

- Data only used on program initialisation
- Pages allocated but unused = full of zeros/blanks
- Pages with lots of repeat data like database records

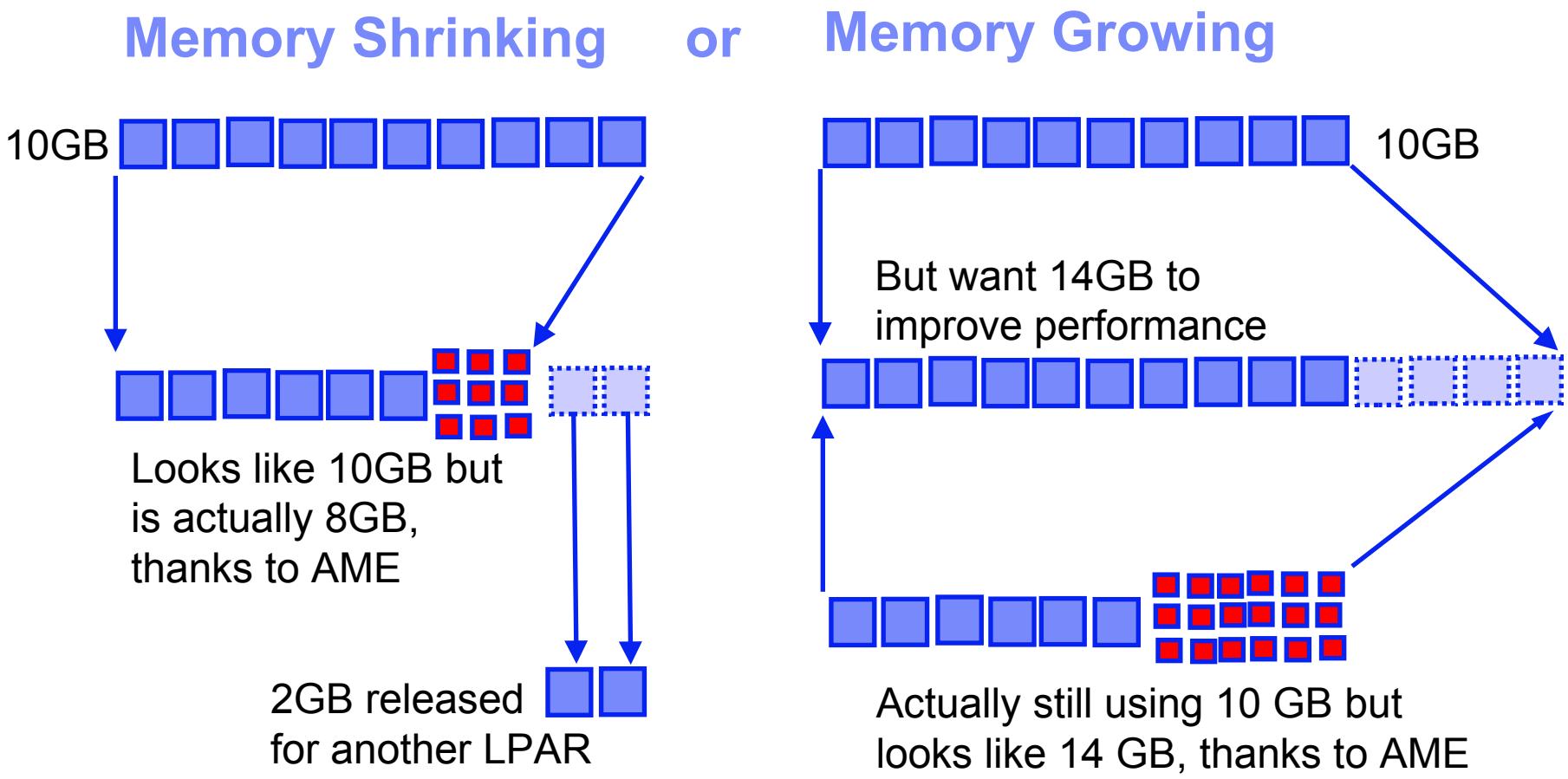
Access Pattern

- Some **hot** pages, some **warm**, some **freezing**
- All pages equally used (HPC) – not so good

Active Memory Expansion – planning for AME

- An AIX command: **amepat**
  - Active Memory Expansion Performance Analysis Tool
- Scans actual memory use
  - Determines compression ratio & CPU requirement
- With AME on or AME off
  - AIX 6.1 TL04 SP2+ also works on POWER4/5/6/7

Active Memory Expansion – planning for AME



Active Memory Expansion – amepat example

```
-> REMOVED CONFIG DETAILS ABOVE HERE
. . .
AME Statistics:                              Current
---------------                            ----------------
AME CPU Usage (Phy. Proc Units)                 0.02 [  1%]
Compressed Memory (MB)                            65 [  4%]
Compression Ratio                               2.04

Active Memory Expansion Modeled Statistics:
-------------------------------------------
Modeled Expanded Memory Size    :    1.50 GB
Average Compression Ratio       :    2.04

Expansion     Modeled True      Modeled                 CPU Usage
Factor        Memory Size       Memory Gain             Estimate
---------     -------------     ------------------      -----------
     1.00          1.50 GB          0.00 KB [  0%]     0.00 [  0%]
     1.09          1.38 GB        128.00 MB [  9%]     0.00 [  0%]
     1.20          1.25 GB        256.00 MB [ 20%]     0.00 [  0%]
     1.33          1.12 GB        384.00 MB [ 33%]     0.13 [  3%]
     1.50          1.00 GB        512.00 MB [ 50%]     0.28 [  7%]

Active Memory Expansion Recommendation:
---------------------------------------
The recommended AME configuration for this workload is to configure
the LPAR with a memory size of 1.00 GB and to configure a memory
expansion factor of 1.50.  This will result in a memory gain of 50%.
With this configuration, the estimated CPU usage due to AME is
approximately 0.28 physical processors, and the estimated overall peak
CPU resource required for the LPAR is 0.85 physical processors.
```

AME thinks 0.28 CPU for 0.5 GB RAM is a good trade-off = last combination

Active Memory Expansion – pre-requisits

POWER7 based machine

AIX 6.1 TL04 SP2+

Also note:

- Transparent to all applications
- Not IVM - Activation key via the HMC
  - But configured at LPAR level
- AME will switch off AIX 64KB page support
  - Can be enabled but tests showed it was slower

Active Memory Sharing vs Active Memory Expansion – comparison

Active Memory Expansion

- Jan 2010
- AIX6 TL4+ on POWER7
- Not Linux nor IBM i
- Machine Activation (LPP)
  - 60 day trial
- Pure Virtual LPAR
- Internal to single LPARs
- Assume some CPU capacity can be used for compression
- Simple to setup in LPAR
- Use amepat to predetermine the compression factor
- Use topas to monitor

Active Memory Sharing

- May 2009
- AIX6 TL3+ & POWER6
- Also Linux & IBM i 6.1
- PowerVM Enterprise
- Pure Virtual LPAR
- Cooperating group of LPARs
- Assumes spare RAM capacity
- Pages flow between LPARs at a few MB/s
- More complex to setup on VIOS & LPARs
- Use topas –C to monitor

Can we use both AMS and AME?

Should work fine but ..

Difficulty, when we start paging to work out why?

1. AIX paging to/from paging space
2. AIX paging to/from file system
3. AMS paging to/from paging space to loan memory
4. AMS remote paging to/from VIOS
5. AME paging to/from compressed pages

Use both but only if you have an IQ of 150+ ☺

- Recommend using one until you are 100% OK with it

Active Memory Deduplication

Active Memory Deduplication

Active Memory™ Deduplication detects and removes duplicate memory pages to optimize memory usage in Active Memory Sharing configurations.

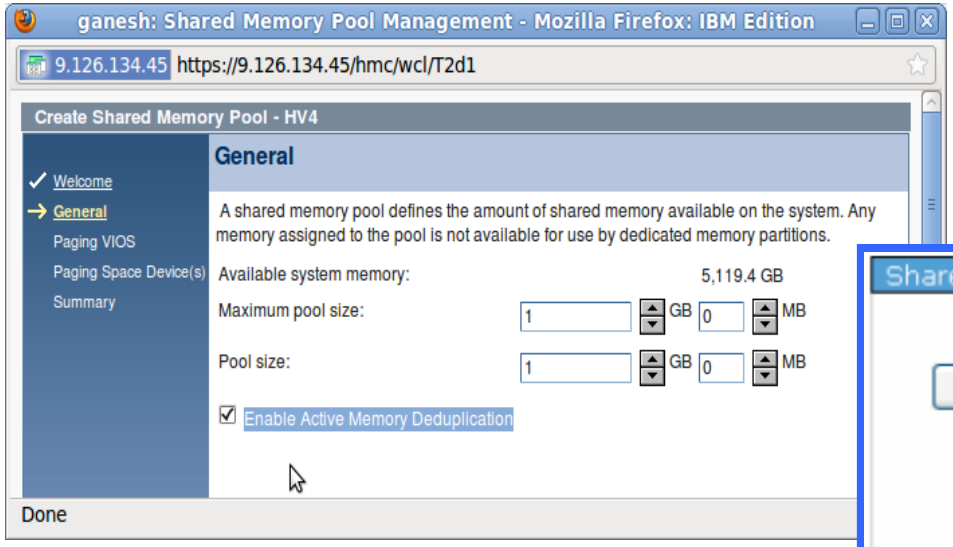
1. The function is performed by the Hypervisor
2. Already involved with Active Memory Sharing Pool
3. Hypervisor entered
   - Handles the Interrupts
   - Operating System makes hypervisor call for services
   - Operating Systems runs out of work, so yields the CPU(s)
4. Finding duplicates is not a high priority task
5. Hypervisor uses non-busy VIOS CPU cycles

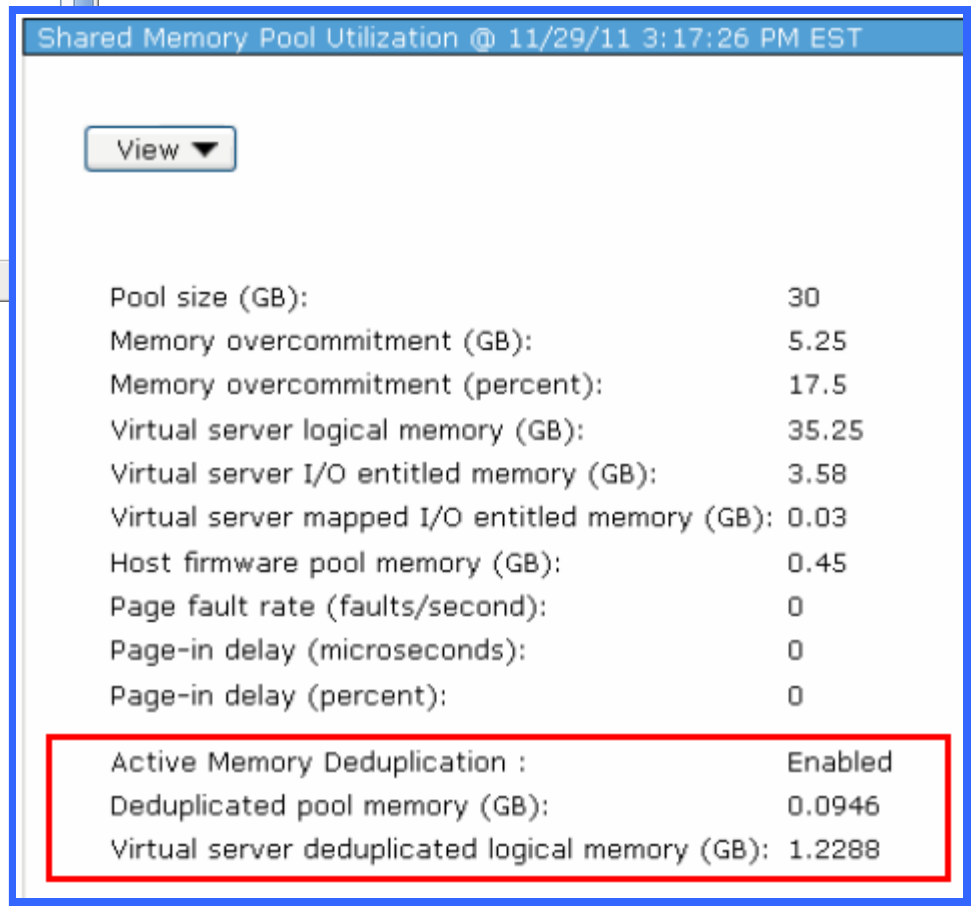
Active Memory Deduplication

To find/remove duplicates, the Hypervisor:

1. Pages are lightly examine to create a “finger print”
2. This is compared with a table of finger prints
3. If no match → add new finger print to in-memory table
4. If matches → the full page is checked
5. If a duplicate change the virtual memory
    a) Both page-table entries refer to a single master page
    b) The other page is put on the free list

Active Memory Deduplication





Active Memory Deduplication

What happens on a page write attempt

1. Master pages are set to read-only
2. The page write generates a memory exception interrupt
3. If a real read-only page
   - Generate process crash signal – this is not allowed
4. If a read-write page
   a) Find a free page
   b) Make a copy of the master page to the new one
   c) Change page-table to refer to the new copy
   d) Change new copy to read-write
   e) Exit the interrupt & the process reties the write and it works

Active Memory Deduplication – memoru page targets

- Good
  - Zero filled memory (perfect!)
    - All heap memory is zero filled to start with
  - Partly used pages (the rest is zeros)
    - Database disk blocks
  - Common read-only program code & static data
    - Operating systems code
    - Applications
  - Anything used by Java ☺

- Bad = memory pages very likely to be unique
  - Every VM running 100% different applications
  - HPC and every VM handling different data models
  - In memory images/movies editing - JPEG, GIF, TIF, MPEG
  - Encrypted data

Active Memory Deduplication

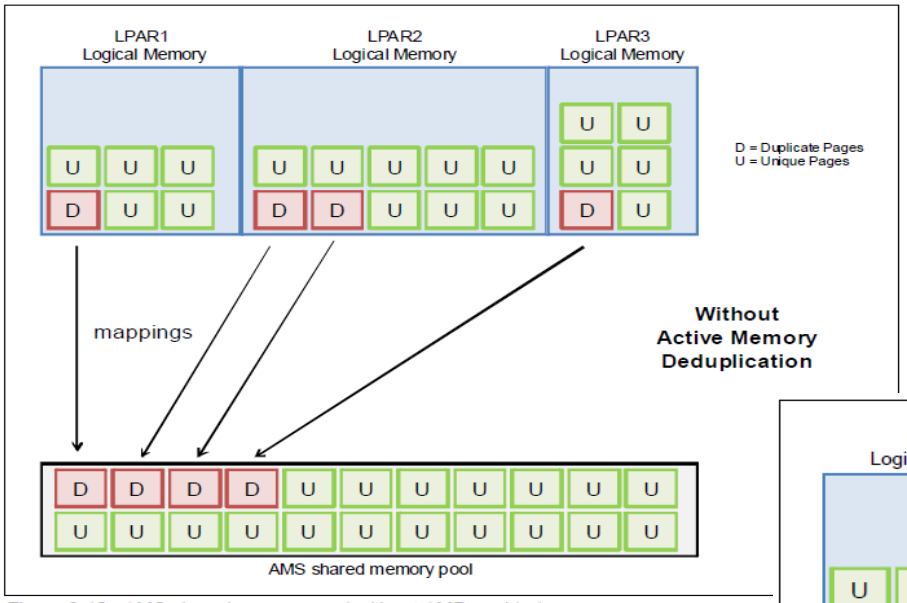


Figure 3-13 AMS shared memory pool without AMD enabled

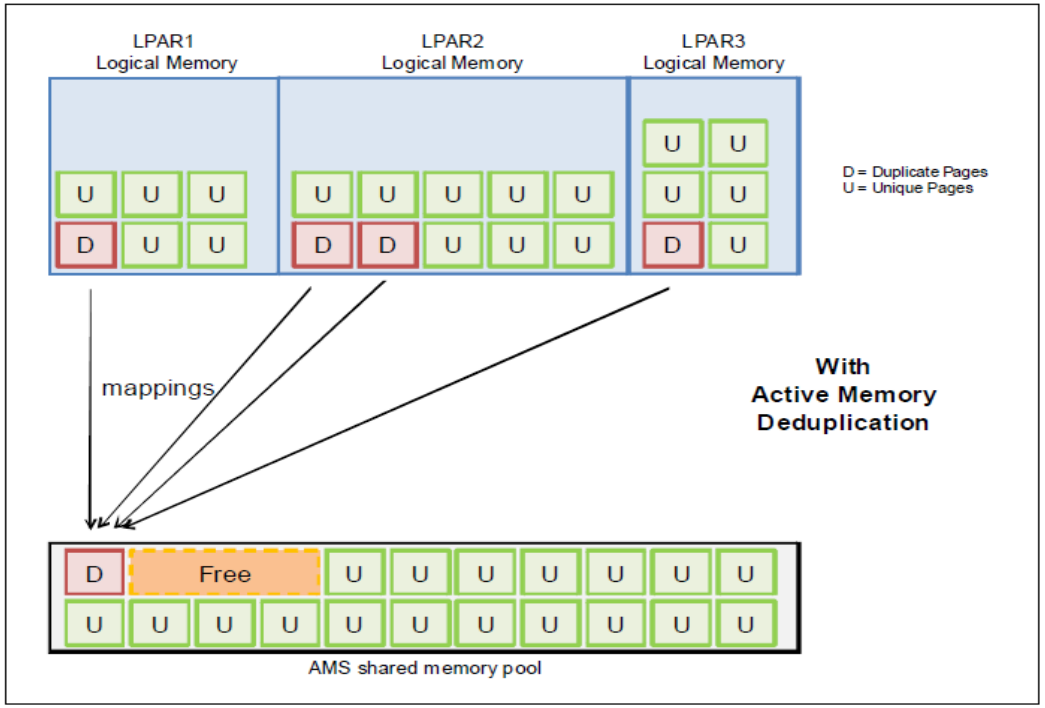


Figure 3-14 Identical memory pages mapped to a single physical memory page with Active Memory Duplication enabled

Active Memory Deduplication – pre requisites

1. **POWER7** only
2. PowerVM **Enterprise** Edition
   - HMC → Server → Capabilities: “AMS Capable”=true
   - Suspect there is also a “Deduplication Capable” too
3. System **Firmware** level **740**
   - HMC → Update panel “EC Number”=01A*740
   - Power7xx C models only introduced in Oct 2011 only
4. **HMC** level **7.7.4**
   - Matches the system firmware
5. **Operating Systems**
   - AIX Version 6: AIX 6.1 TL7, or later
   - AIX Version 7: AIX 7.1 TL1 SP1, or later
   - IBM i: 7.14 or 7.2, or later
   - SLES 11 SP2, or later and RHEL 6.2, or later
6. **Virtual I/O Server 2.1.1.10** (FP21) or later
   - Use VIOS ioslevel command
   - AMD uses VIOS CPU cycles via the Hypervisor code but not VIOS/AIX code = so no dependency

Nigel suggests: latest VIOS 2.2.1.3 = FP25 Oct 2011 or at least 2.2.something

Active Memory Deduplication – pre requisites

7. **AMS** virtual machine requirements
   - Deduplication is ONLY for Active Memory Sharing virtual machines (LPARs), so AMS pre-reqs apply

   - Shared CPU only (no dedicated CPUs)
   - Shared I/O only (no dedicated adapters)
   - No 16 MB pages (used by some HPC codes)

   - LPAR needs restarting in AMS mode
   - Only one pool = single set of co-operating VMs

Questions?

CREDITS to

Nigel Griffiths
IBM Power ATS EMEA
IBM UK

Luca Comparini
STG Lab Services Europe
IBM FR

THANKS