Active Memory Sharing

Nigel Griffiths
IBM Europe ATS

Active Memory Sharing (AMS) Sales Pitch

AMS allows the dynamic moving of memory between LPARs at a 4KB page level

- Reduces required memory
  = save you money
- Finds little used memory
  = save you time
- Moves memory to where its needed
  = increased flexibility

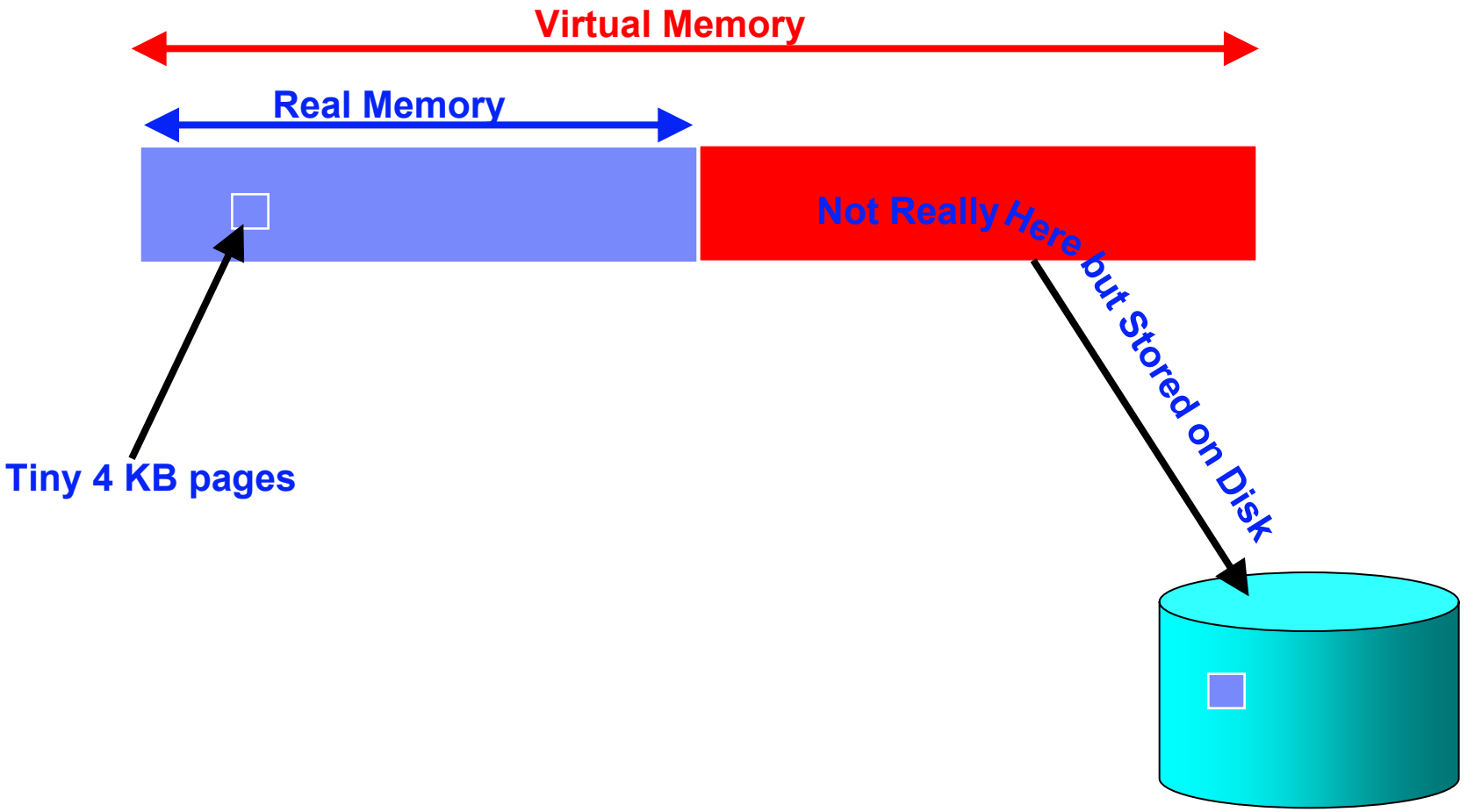
Active Memory Sharing (AMS) Warning

AMS is built on top of Virtual Memory & Paging

Mandatory:
You need to understand both

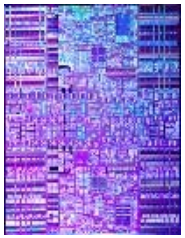
Covered in three slides!

So how does Virtual Memory work?



Why bother? Disk space costs 1000 times less

Paging in



1) Page fault
→invoke kernel

6) Restart
Instruction

Programs View

Code

Data

Data

Heap

Code

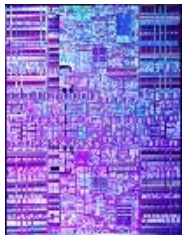
Data

Stack

Stack

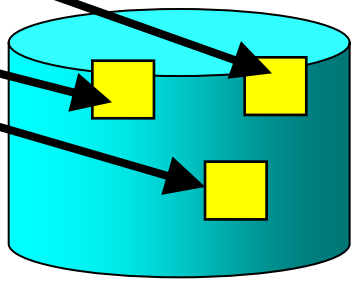
Memory Pages 4 KB

2) Check Address in the
Page Table



Page Tables entries
managed by AIX Kernel

Raw block copy on the
Paging space device



5) Fix page table

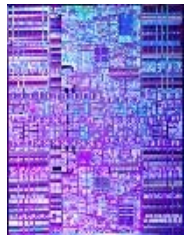
3) Find memory on free list
4) Move disk copy into
empty memory page

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

Want to know more? – watch the movie

Five Paging Golden Rules

Is Paging Good or Bad for Performance?

Bad but happens

How much can we cope with?

It depends

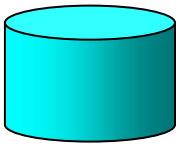
My Rule of Thumb:

10 pages
per second
per CPU
per paging disk

= ignorable noise level

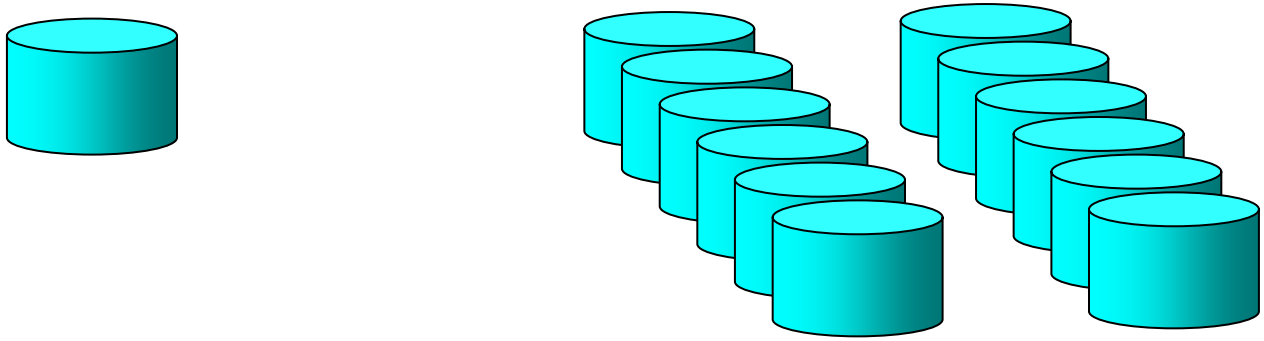
How to live with paging?

- Spread paging I/O across many disk spindles
  - 5 second glitch



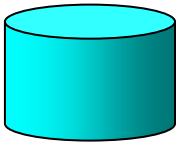
How to live with paging?

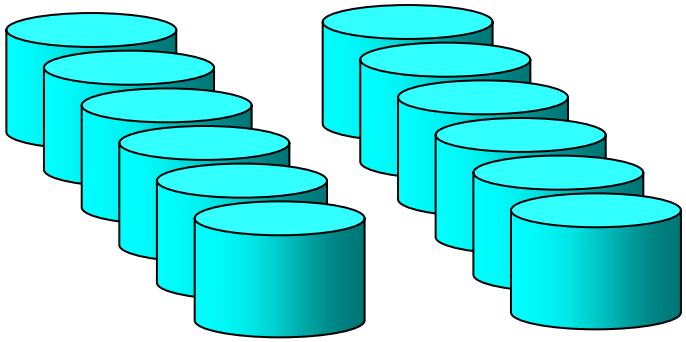
- Spread paging I/O across many disk spindles
  - 5 second glitch or 0.5 second glitch



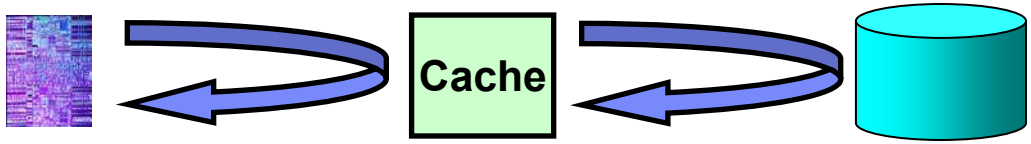
How to live with paging?

- Spread paging I/O across many disk spindles
  - 5 second glitch or 0.5 second glitch





- Disks with caches



- Solid State Disks

If you lose paging space, can you survive?

If you lose paging space, can you survive?

No you can’t

Paging Space 100%! - What happens next??

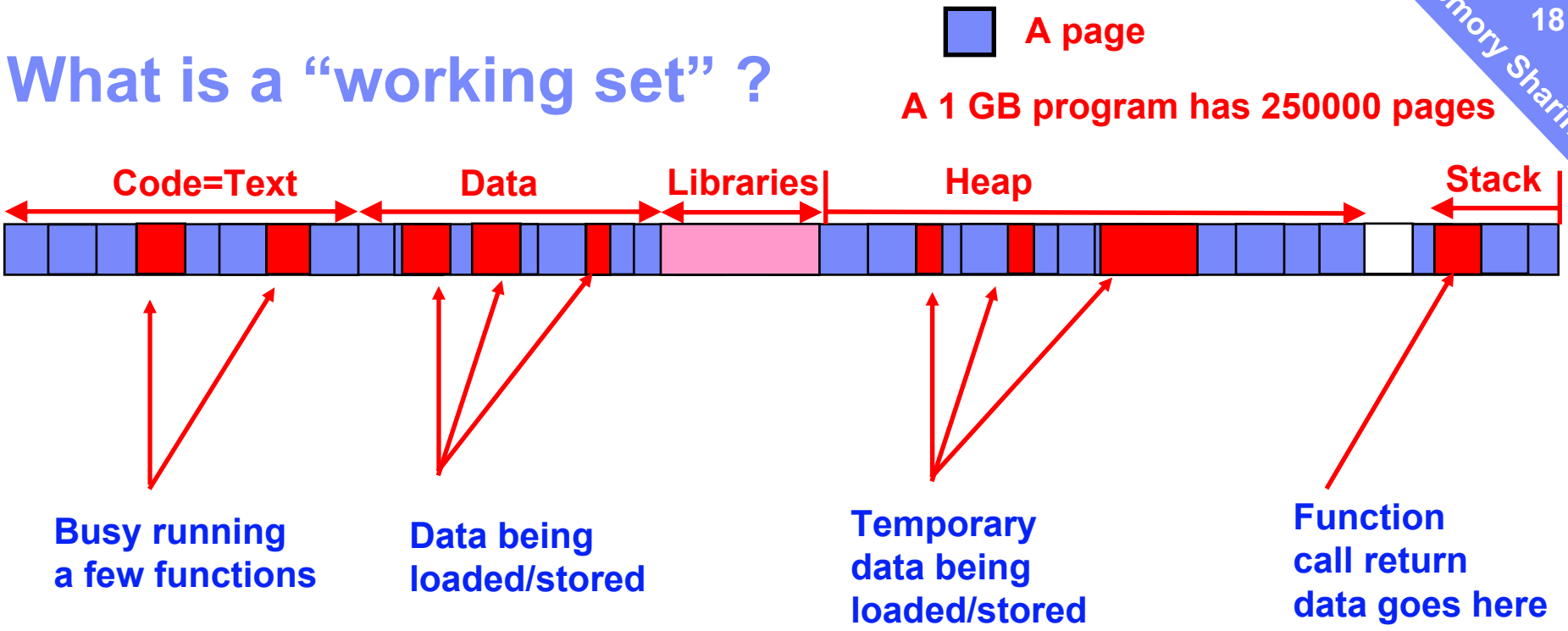
UNIX version 7 Manual entry:

"Absolute mayhem is guaranteed"

Five Paging Golden Rules

1. Don't do it! → hurts 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 → mayhem!

What is a “working set” ?

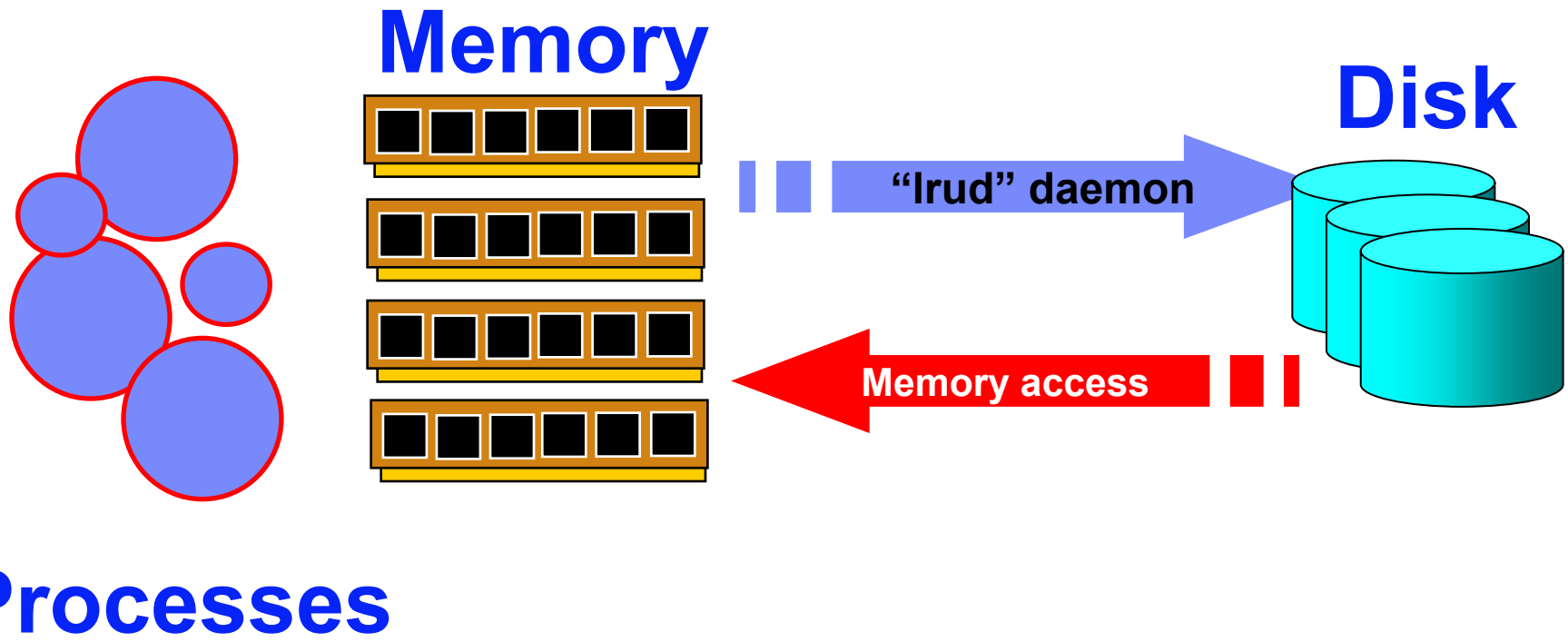


Working Set is the pages needed to run in the short term (seconds)

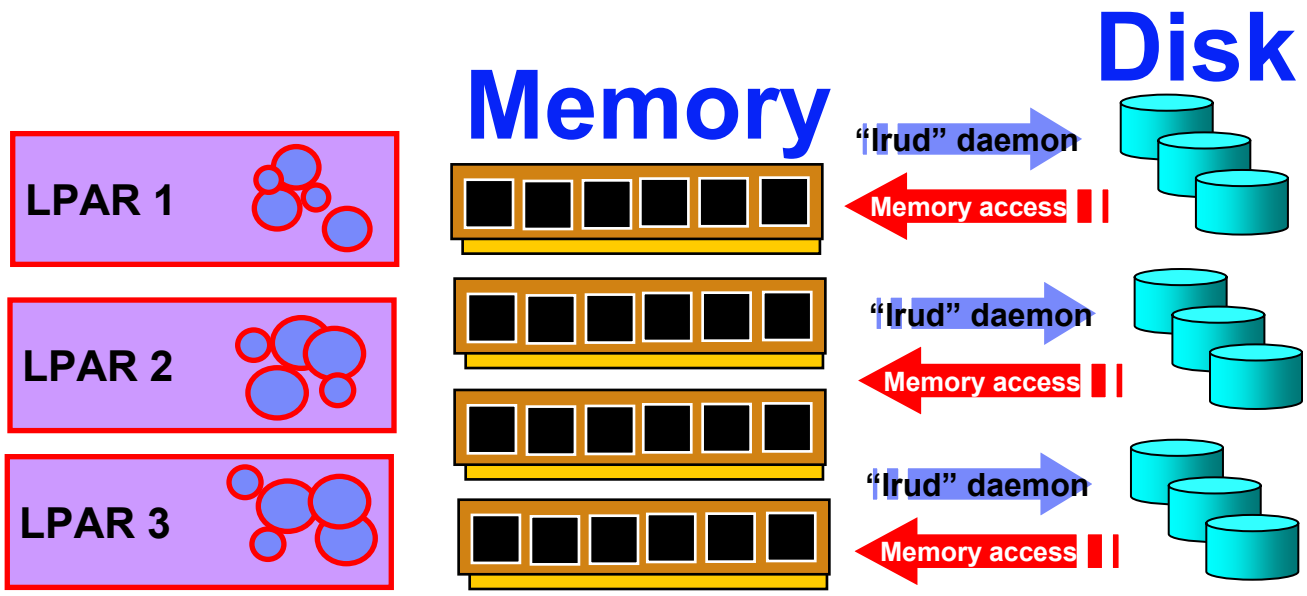
Also called Resident Set (resident in memory), see ps or nmon ResText & ResData

AMS acts on Working Sets but at whole LPAR level

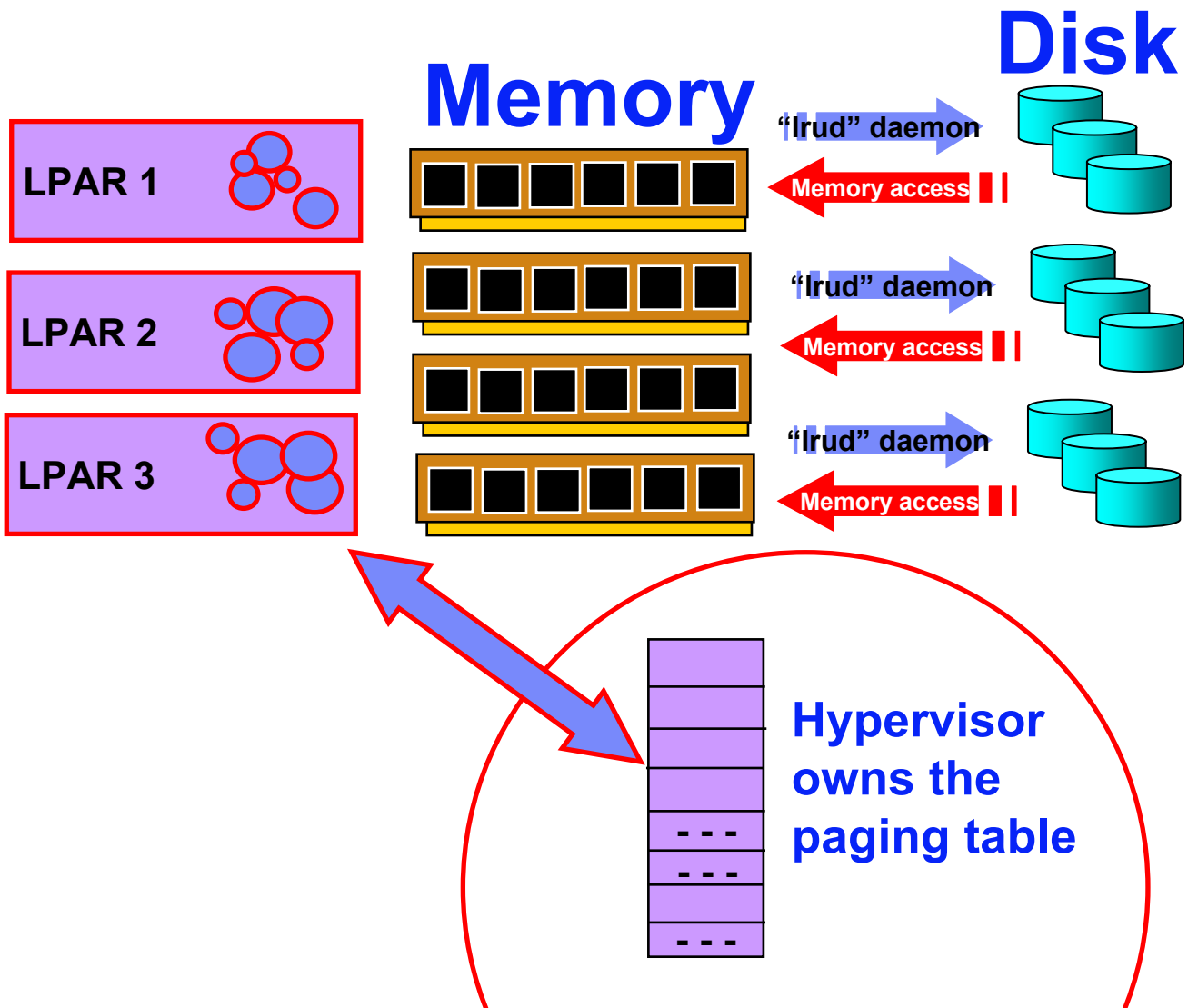
AIX Level Paging



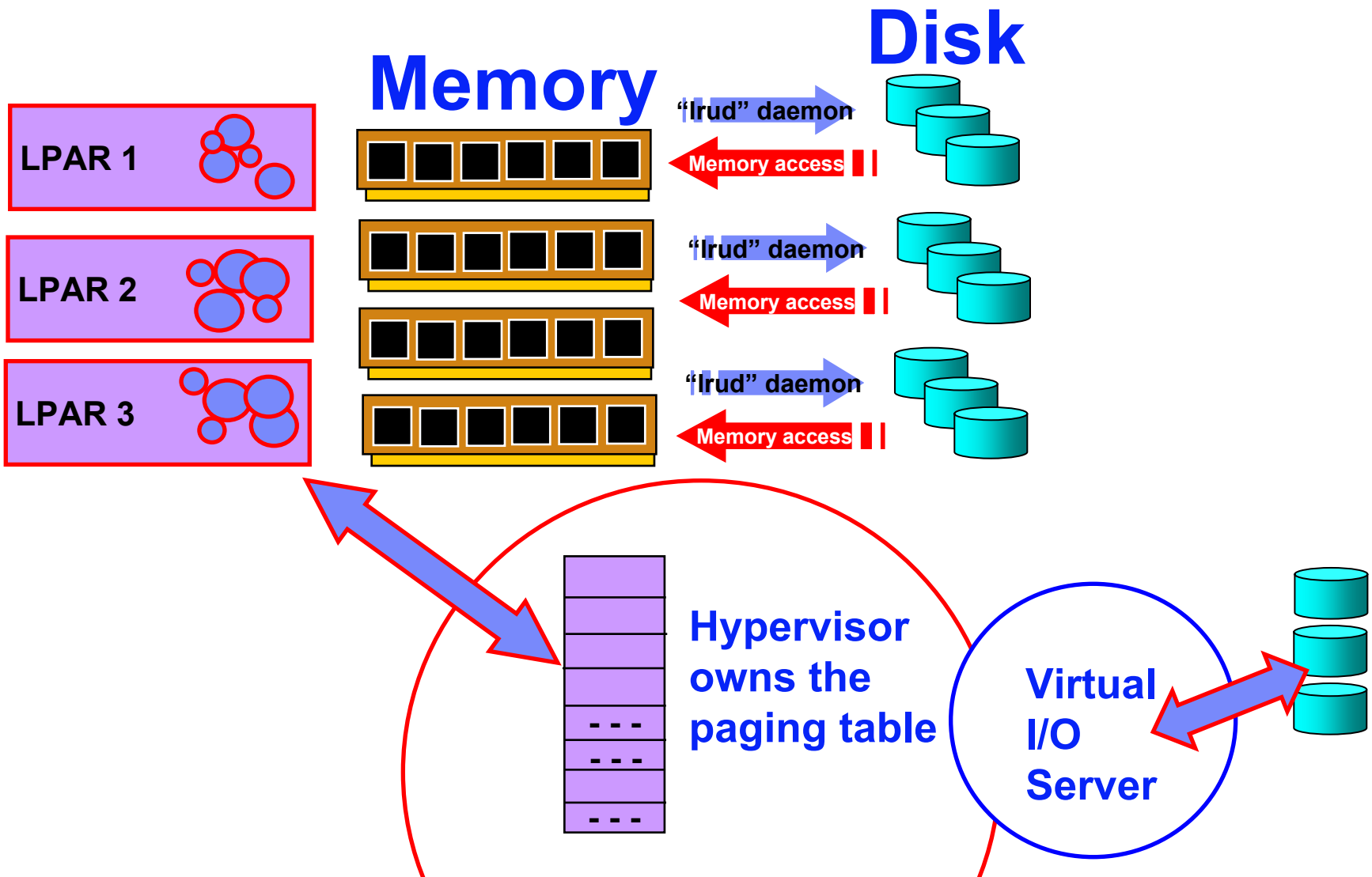
LPAR Level Paging = AMS



LPAR Level Paging = AMS



LPAR Level Paging = AMS

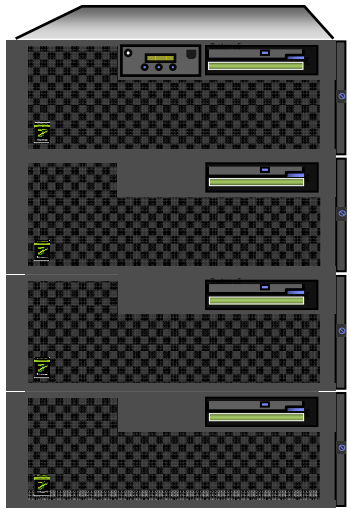


What is the problem?

Problem 1 – Where is the spare?

You have:
- 100 “standard template” LPARs

LPAR number 101
- Spare Shared CPU → YES
- Spare Memory → No



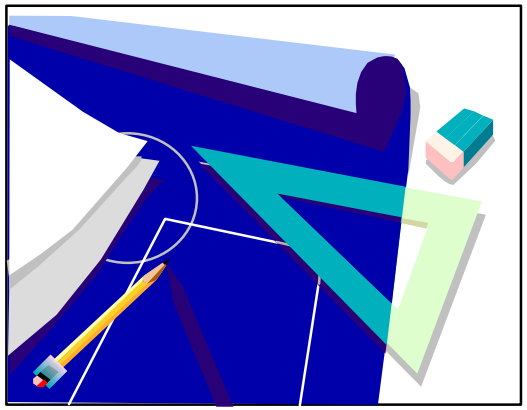
Which LPARs could give up some memory?
= Impossible to work out!

AMS can help

Problem 2 – Where to squeeze?

Solution design, each LPAR: How much memory?

- Guess “10 GB sounds about right”
- Policy “every app server gets 8 GB”
- App vendor recommend “12.5 GB”
- Add a bit for safety +10% ?



It is a Guessimate

Add it up and you get 280 GB

- Darn! That means the big DIMMs = €$£€$£€$£€$£
- If it was 256 GB = smaller DIMMs = cost down, speed up

AMS automatically balances RAM based on real use

Problem 3 – When to share and what?

Around the World

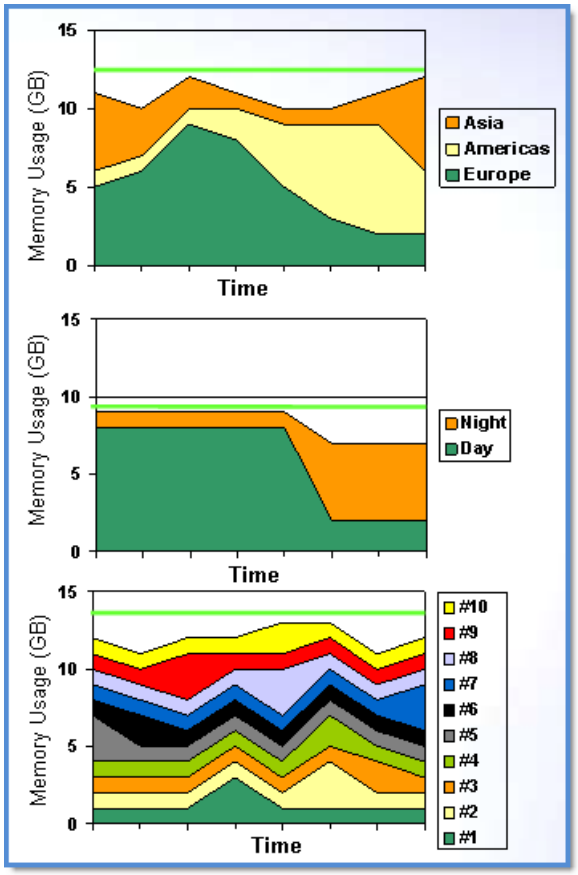
- Demands that peak at different times

Day and Night

- Day time web app & nightly batch

Infrequent use

- Many sporadic use partitions



Failover Ready Partition

- Like “Day and Night” but never actually happens 😐

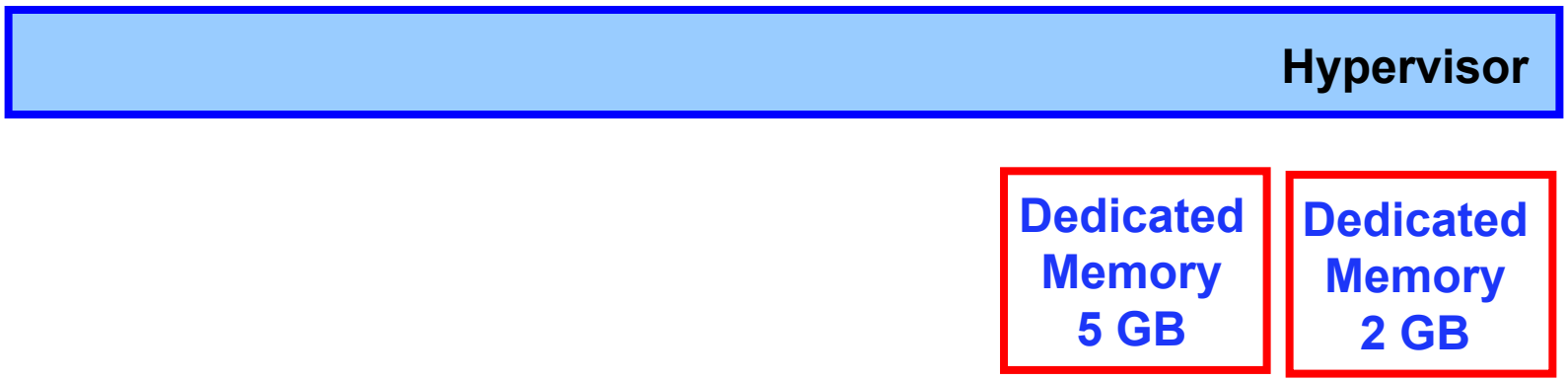
AMS pre-reqs?

* means new in May 2009

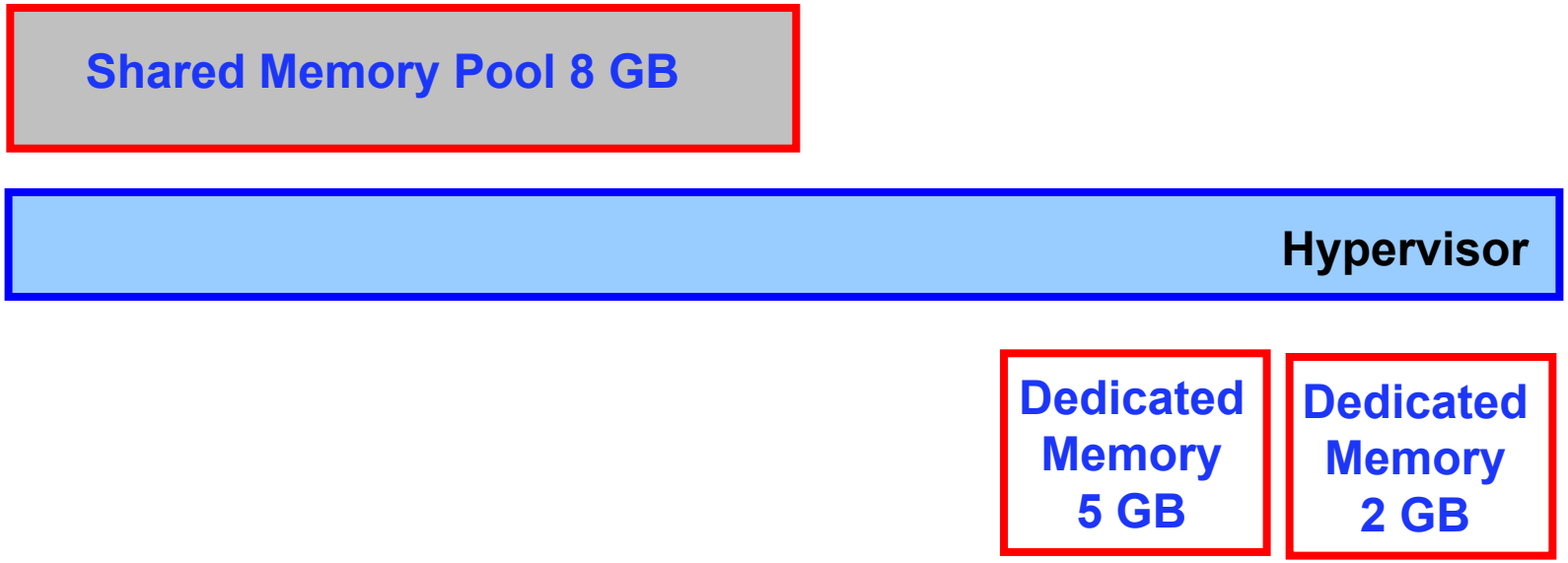
AMS Pre-Requisites

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)

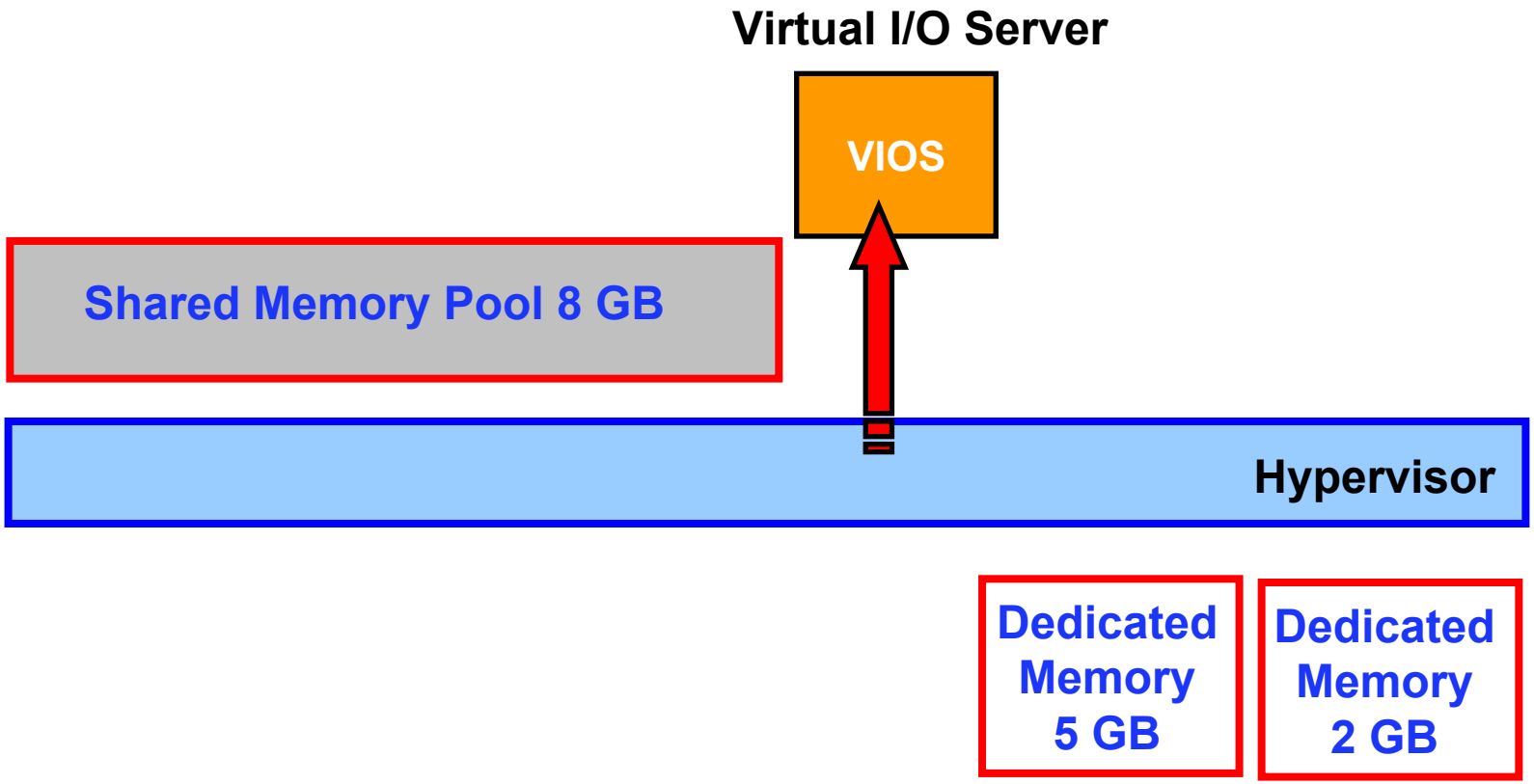
How is it set up?



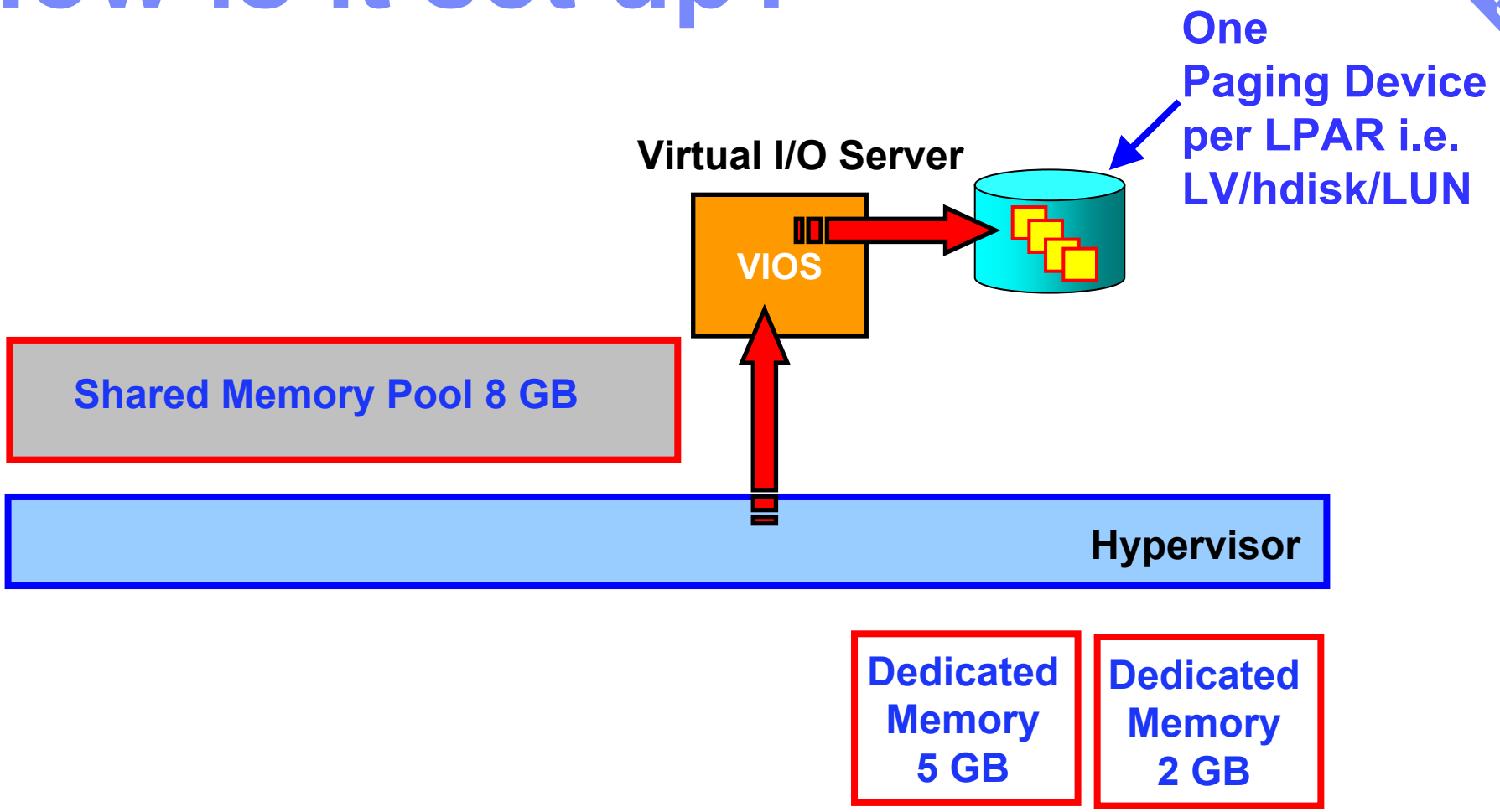
How is it set up?



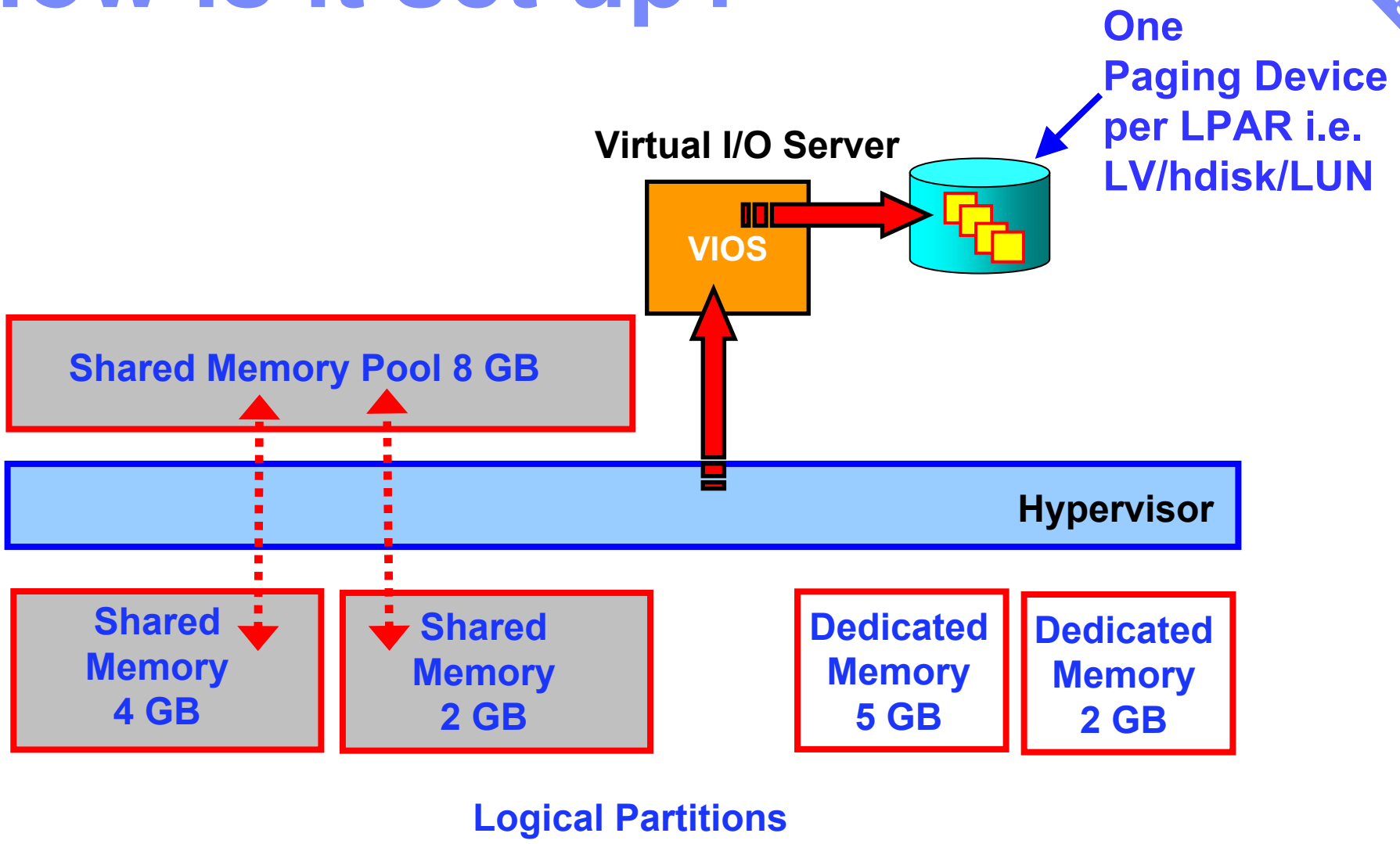
How is it set up?



How is it set up?



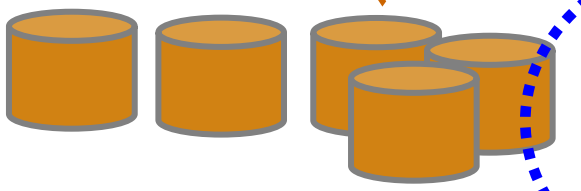
How is it set up?



POWER6 p520

CPU Four 4.2 GHz
RAM 16 GB
Disks Five 146 GB

VIOS rootvg
Virtual disks clientvg
AMS pagingvg

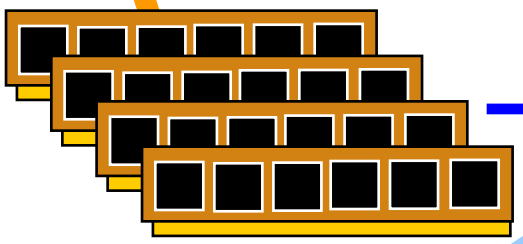


Logical Volumes:
ams1
ams2
....
ams6

Virtual I/O Server

all 32 GB

Client LPAR
Client LPAR



Shared Memory Pool

Client LPAR
Client LPAR
Client LPAR
Client LPAR
Client LPAR
Client LPAR
Client LPAR

If possible Demo Here

Machine Level - Memory Pool

Shared Memory Pool

- Bit like Shared Processor Pool
- Only one pool (no license issue here)

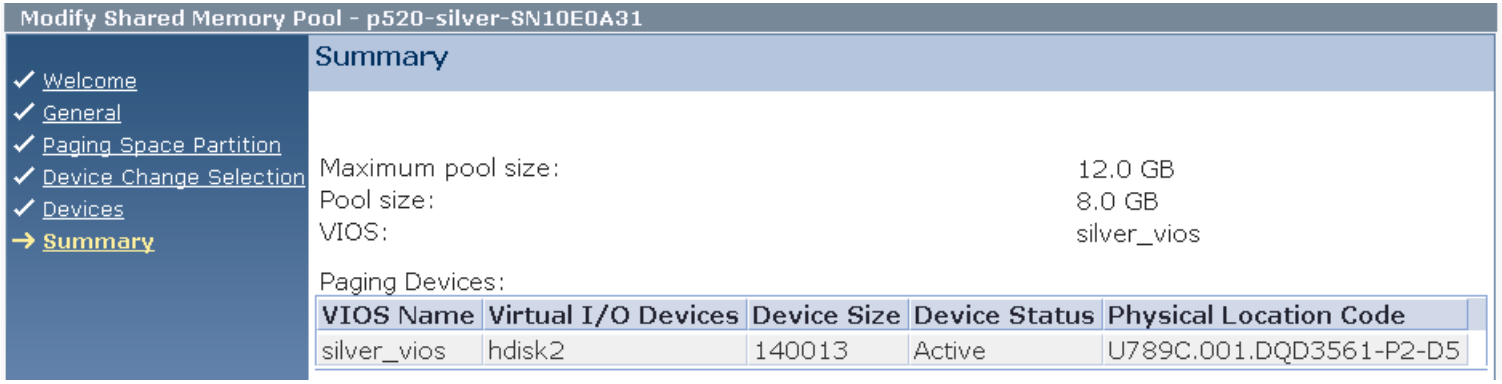
Creating the pool – HMC/IVM machine level

Specify

1. Pool size
2. Pool maximum size (sanity check for dynamic change)
3. VIOS to use for AMS paging
4. AMS paging spaces

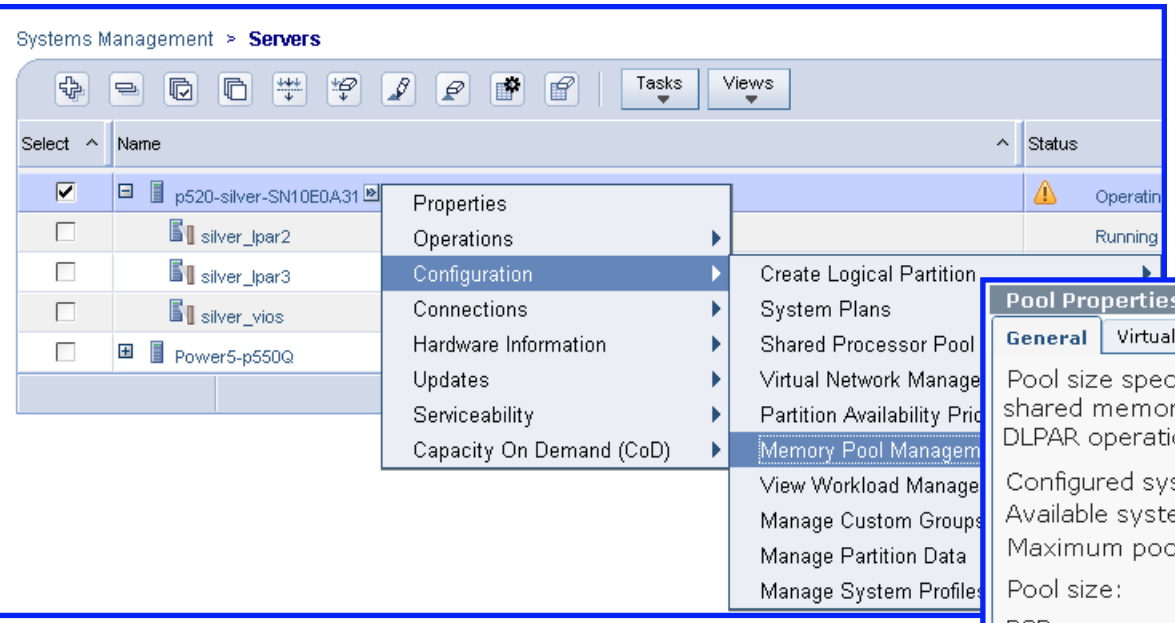
https://hmc8.aixncc.uk.ibm.com/preloginmonitor/index.jsp

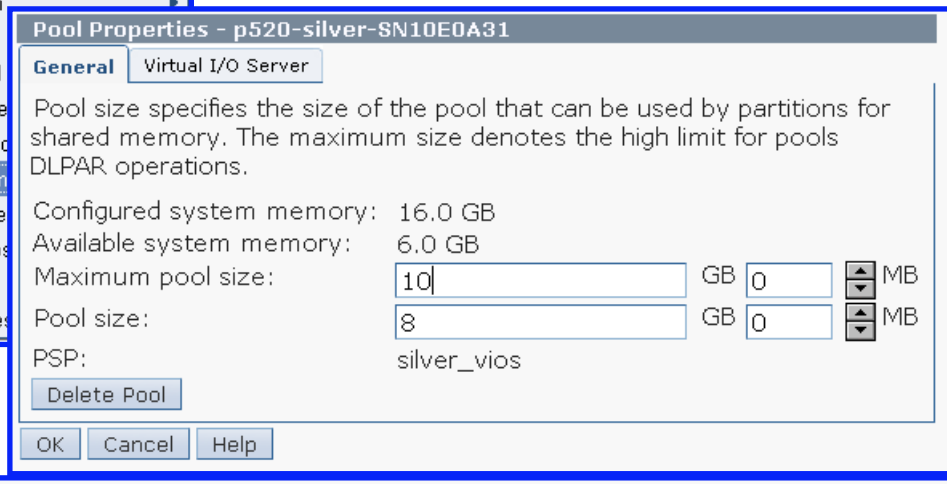
Creating a Memory Pool

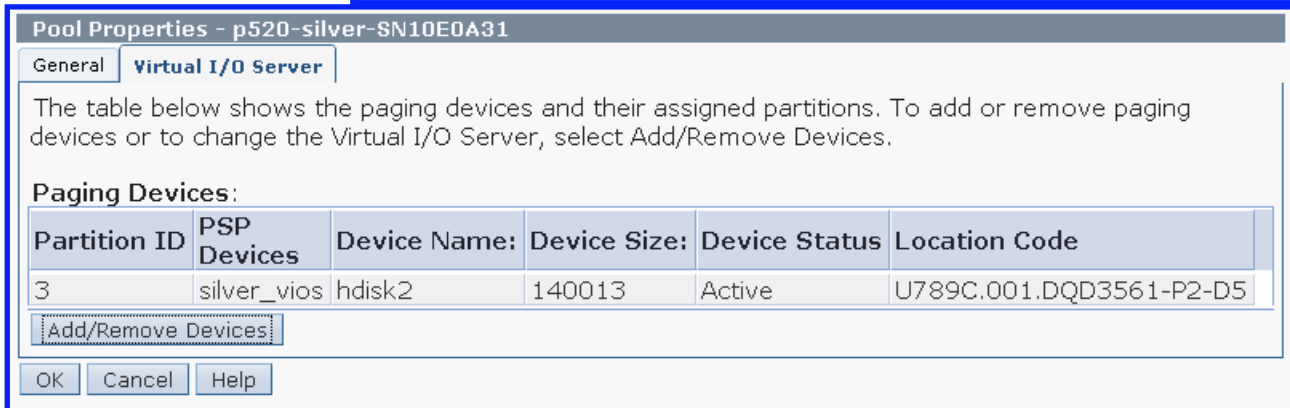


- Select the machine
  - Decide size of pool
  - Select VIOS for paging space
  - Select Device (whole hdisk or logical volume!)
- Create

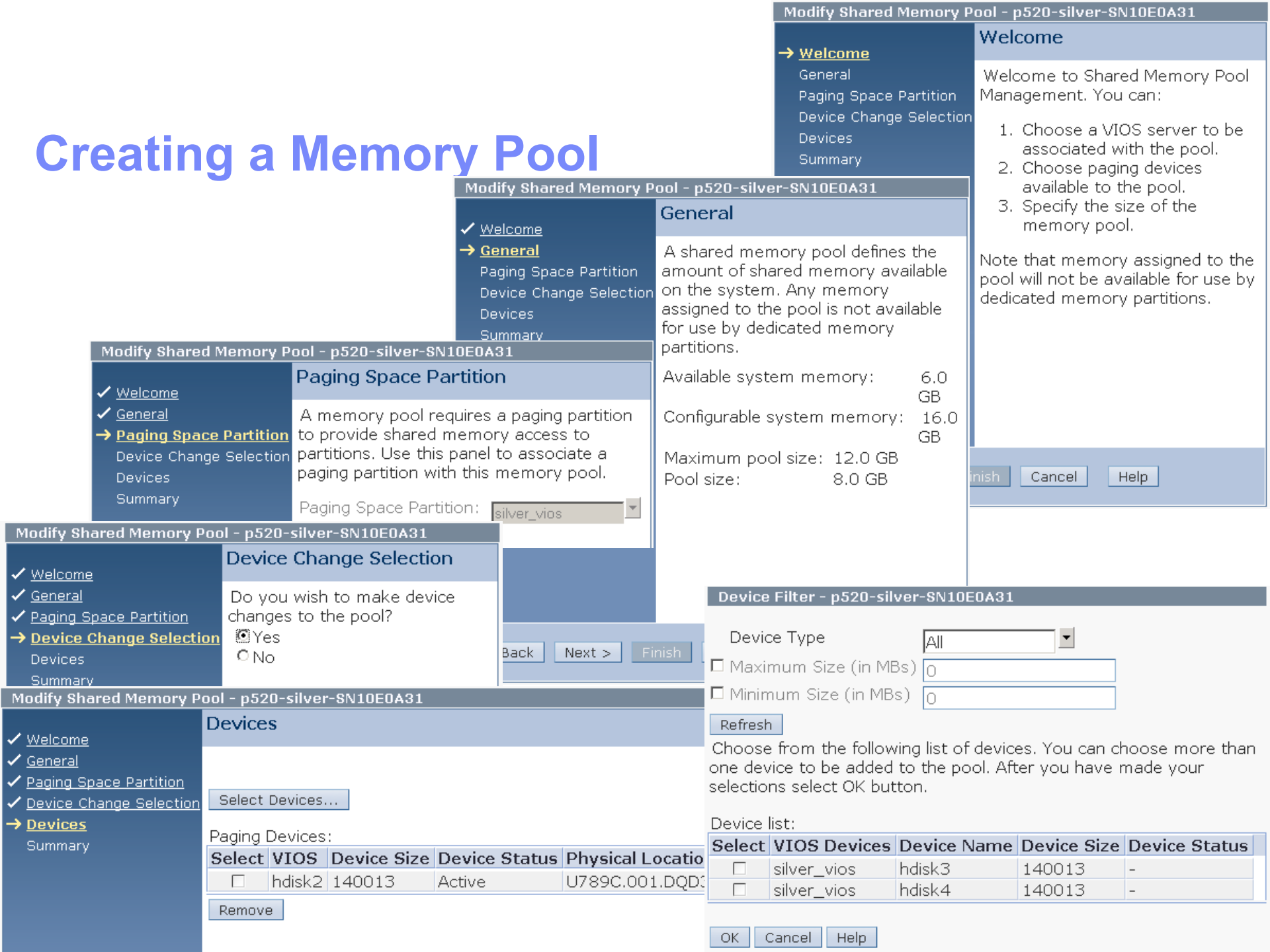
Creating a Memory Pool







Creating a Memory Pool



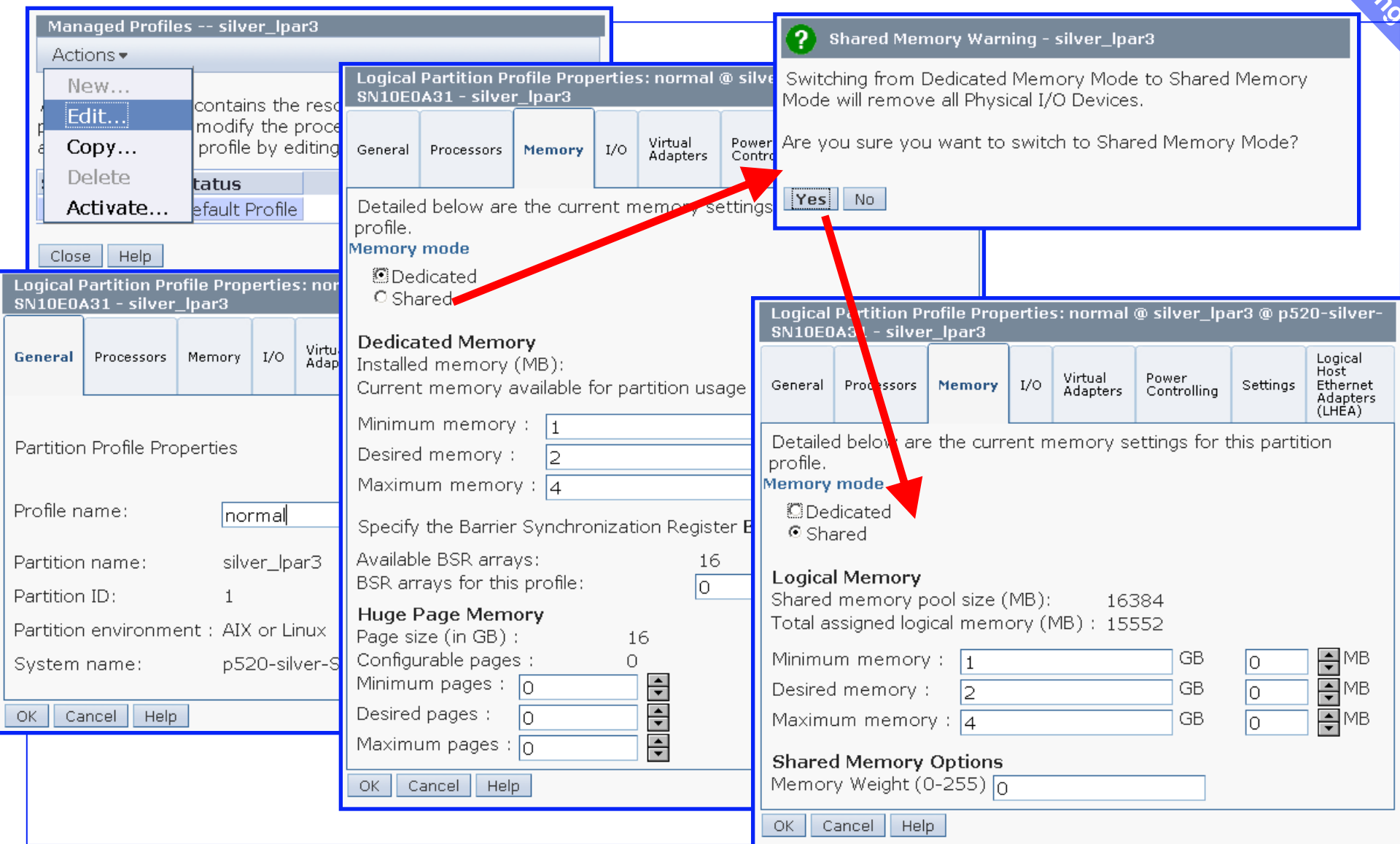
LPAR level: Allocate Shared Memory

- LPARs are either
  - Dedicate Memory (old style)
  - Shared Memory from the Memory Pool

} NOT BOTH

- LPAR Switch Shared ←→ Dedicated Memory
  Cold LPAR Reboot (not restart)
- Shared Memory min, desired, max are
  Logical Memory (not Physical)
- LPAR Dynamic Memory changes are Logical too

Creating LPAR with Shared Memory



Each LPAR uses 1 AMS paging space = Easier for PM

Systems Management > **Servers**

Tasks ▾ Views ▾

Select	Name	Status	ID	Available Processing Units	Processing Units	Processor	Available Memory (GB)	Memory (GB)
☑	p520-silver-SN10E0A31	Operating		2			9.75	
☐	silver_lpar3	Running	3		0.5	1		2
☐	silver_lpar4	Running	4		0.5	1		2
☐	silver_lpar5	Not Activated	5		0	0		0
☐	silver_vios	Running	2		0.5	1		1
☐	sliver_lpar2	Running	1		0.5	1		2

In use

Pool Properties - p520-silver-SN10E0A31

General | **Virtual I/O Server**

The table below shows the paging devices and their assigned partitions. To add or remove paging devices or to change the Virtual I/O Server, select Add/Remove Devices.

Paging Devices:

Partition ID	PSP Devices	Device Name:	Device Size:	Device Status	Location Code
4	silver_vios	ams1	32768	Active	
3	silver_vios	ams2	32768	Active	
1	silver_vios	ams3	32768	Active	
	silver_vios	ams4	32768	Inactive	

Add/Remove Devices

OK | Cancel | Help

Named VIOS

Logical Volume on the VIOS

32 GB

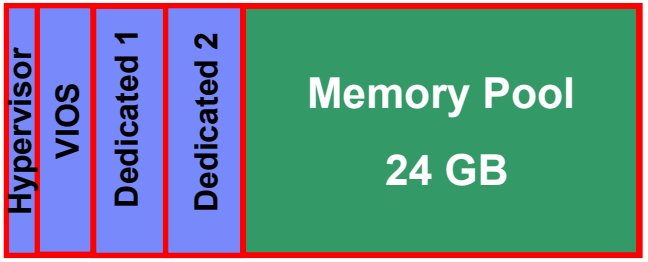
In use

Only 1 more AMS LPAR can be started

Work through an example …

Configuration Example

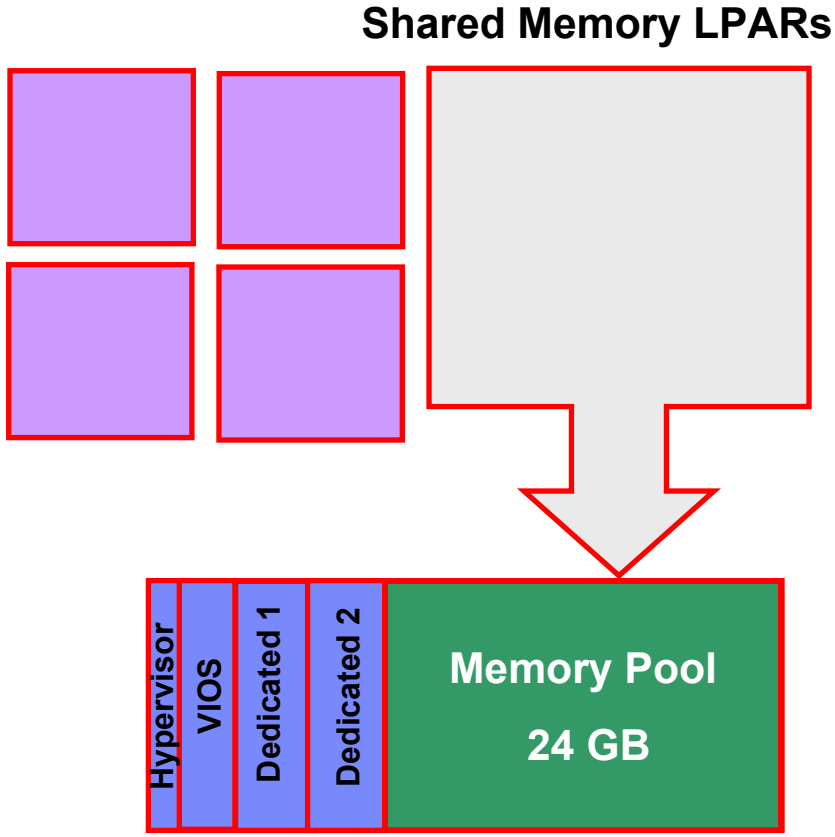
- Example: 32 GB Machine
  - Hypervisor 2 GB
  - 2 x 2 GB dedicated LPAR 4 GB
  - VIOS LPAR 2 GB
  - **Memory Pool 24 GB**



Configuration Example

- Shared Memory LPARs
- Create 4 x 8 GB memory pool LPARs
  - 32 GB of Logical Memory
  - 24 GB of Physical Memory

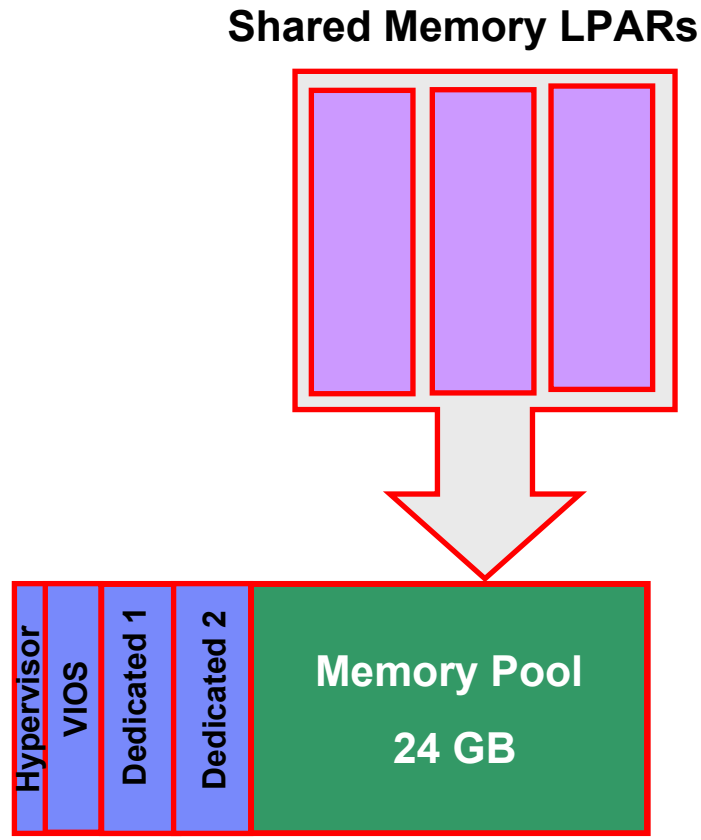
i.e. does not fit



Configuration Example

Situation:

1. If 3 LPARs started = it fits

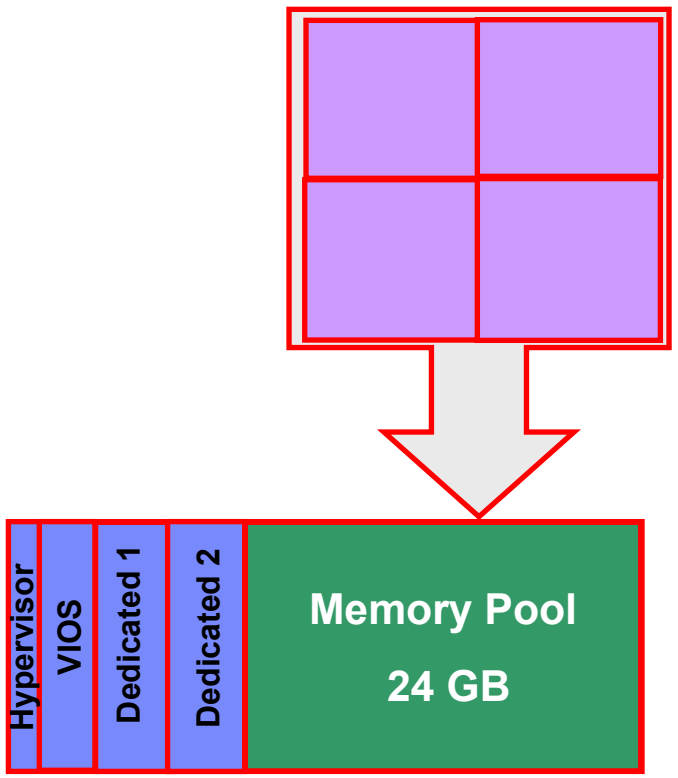


Configuration Example

Start fourth LPAR

Situation :

1. If 3 LPARs started = it fits
2. If Resident Size ~ 24 GB → it works

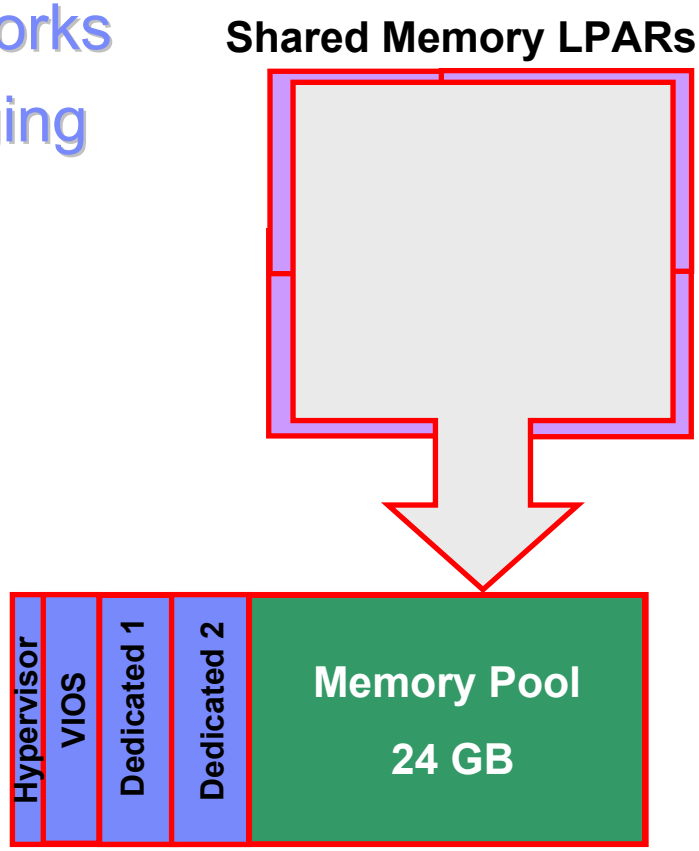


Configuration Example

LPAR demand more memory

Situation :

1. If 3 LPARs started = it fits
2. If Resident Size ~ 24 GB → it works
3. If Resident Size > 24 GB → paging

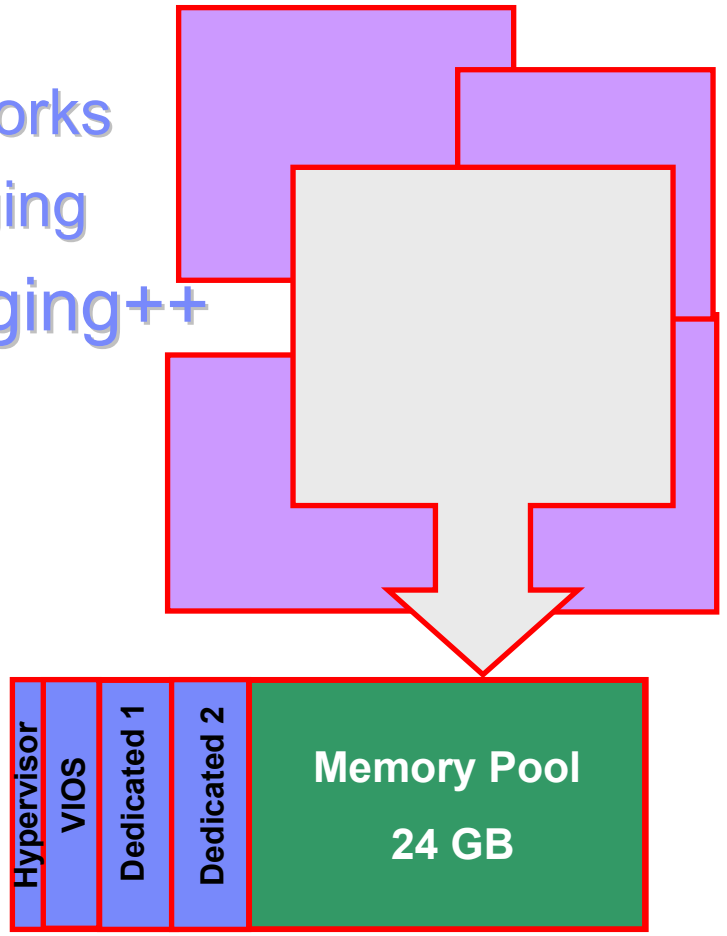


Configuration Example

LPAR demand lots more memory

Situation :

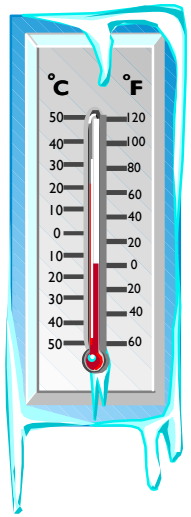
1. If 3 LPARs started = it fits
2. If Resident Size ~ 24 GB → it works
3. If Resident Size > 24 GB → paging
4. If Resident Size >> 24 GB → paging++



How does AMS deal with the four situations?

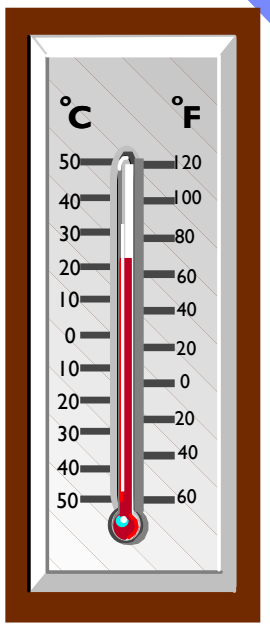
AMS Algorithm 1 – It all fits

- Local paging AIX level
- Not an issue



Relaxed Mode

AMS Algorithm 2 - If it nearly fits?



Hypervisor asks AIX images for help

→ once a second

1. AIX then frees memory, if necessary paging out
2. Loans pages to Hypervisor
3. Hypervisor gives pages to high demand LPAR

Co-operative Mode

AMS Algorithm 2 - If it nearly fits?

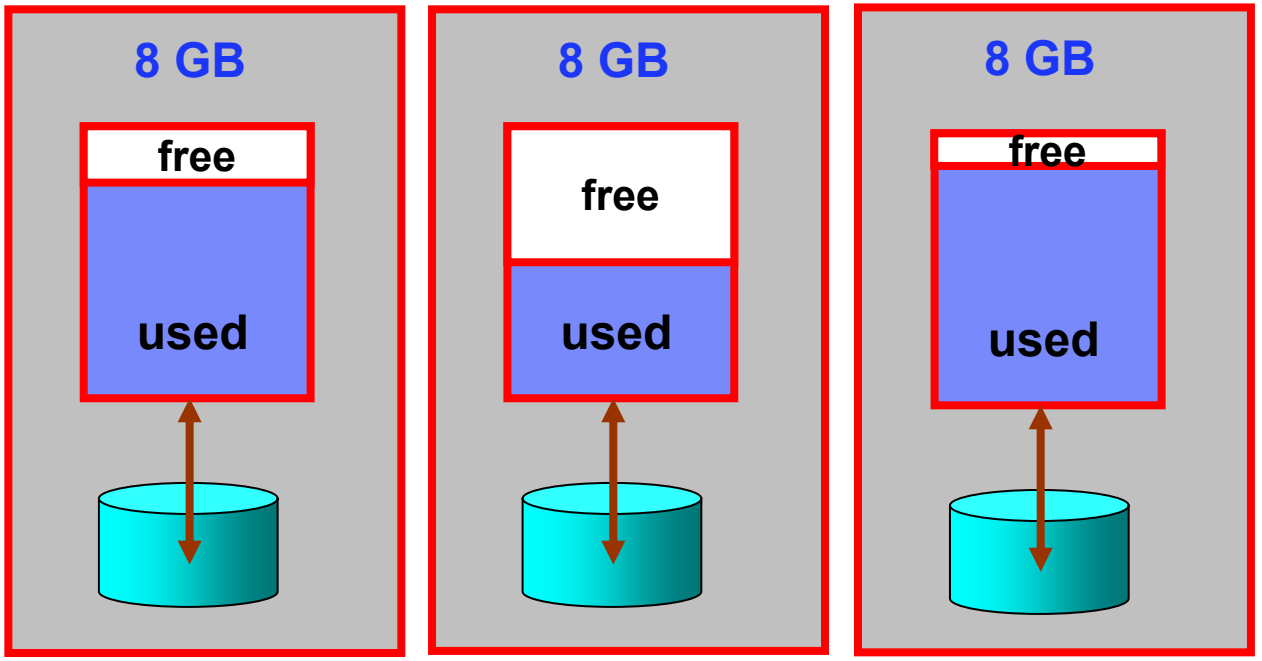
AIX level AMS Tuning on how aggressive:
none, File system cache, programs too

```
# vmo -L ams_loan_policy
NAME                      CUR    DEF    BOOT   MIN    MAX    UNIT           TYPE
------------------------------------------------------------------------------
ams_loan_policy           n/a    1      1      0      2      numeric          D
------------------------------------------------------------------------------
```

0 = no loans
1 = filesystem cache only (default)
2 = also loan program memory

Co-operative Mode

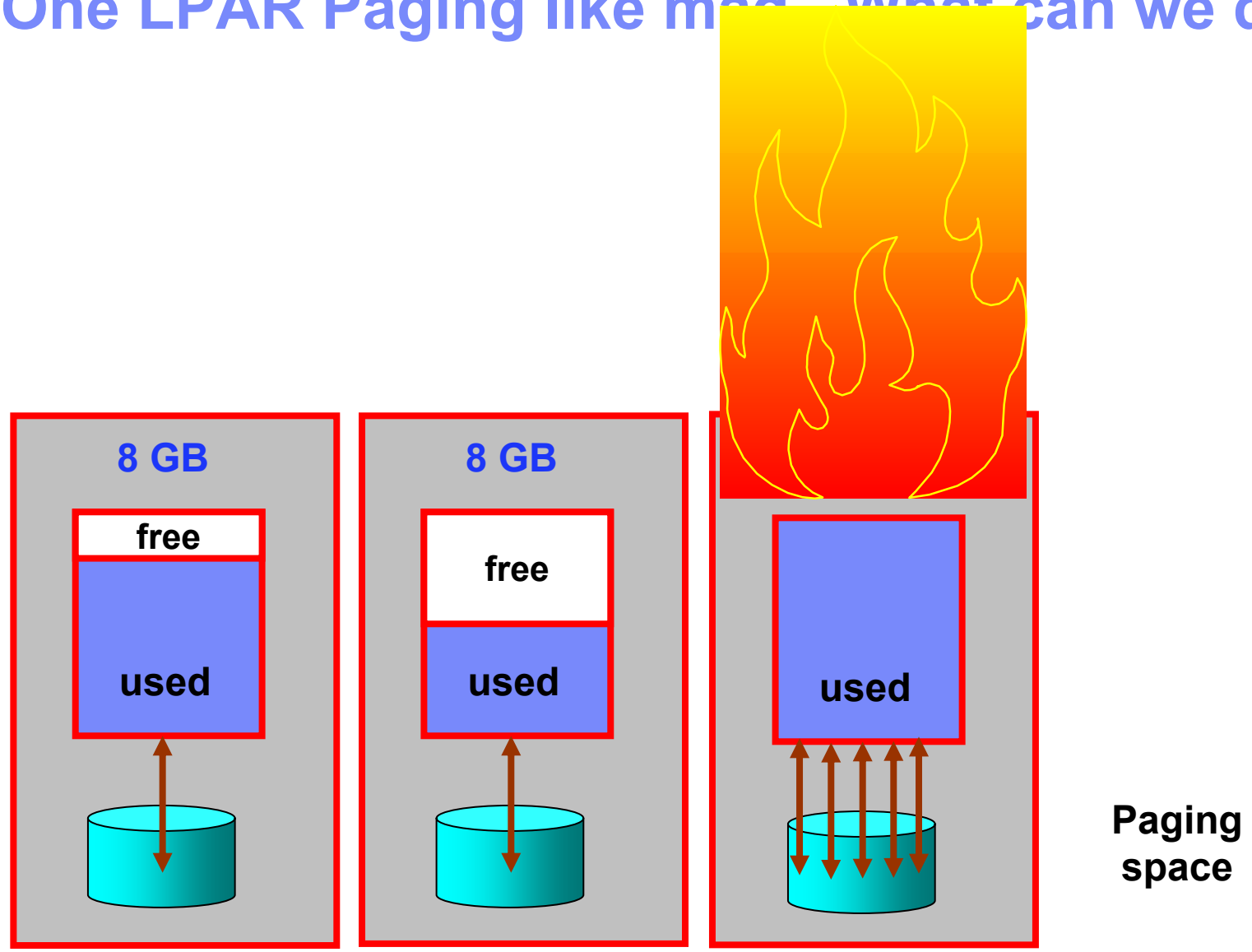
Starting with It All Fits = sweet



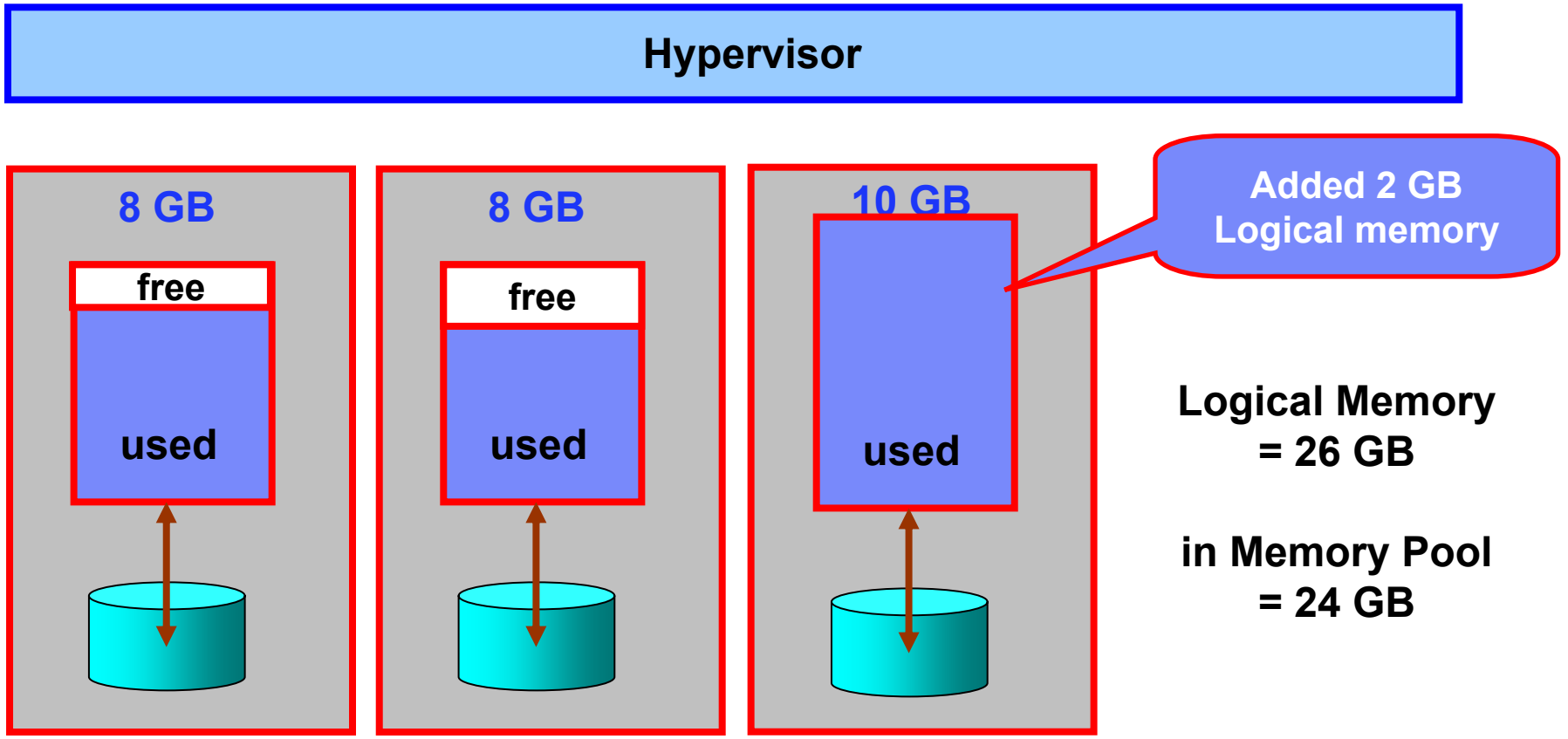
Memory pool
= 24 GB

Paging
space

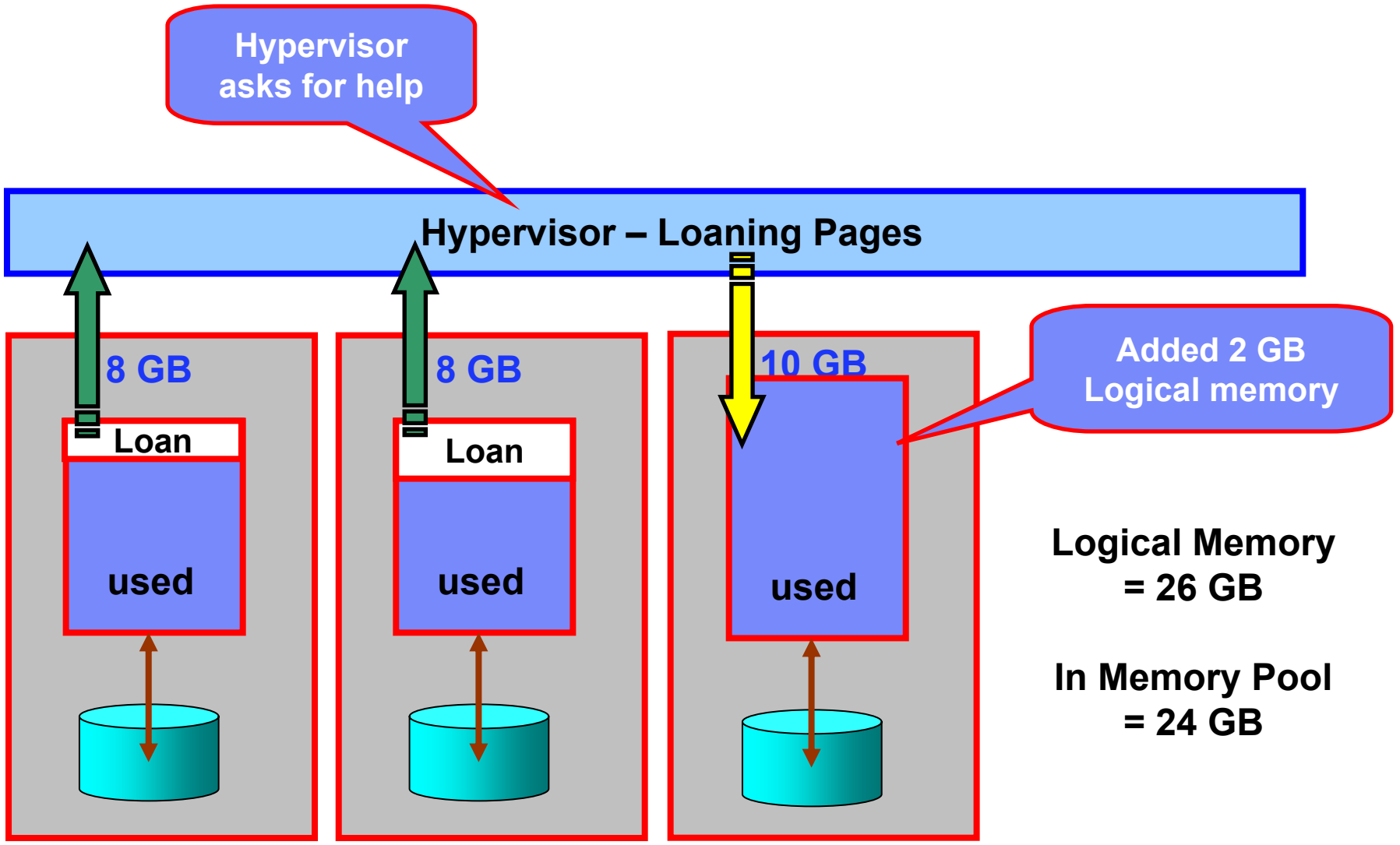
One LPAR Paging like mad. What can we do?



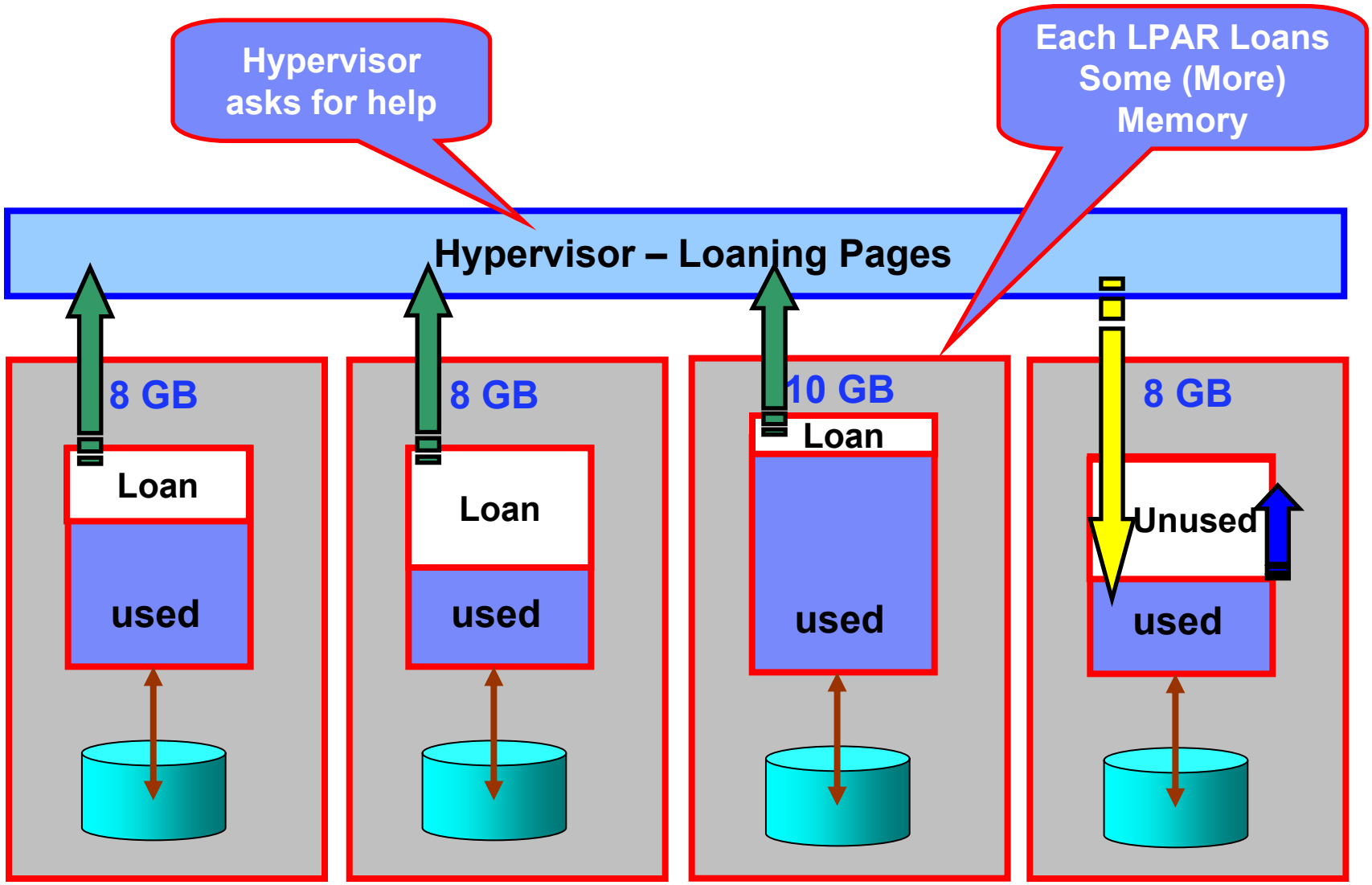
Other LPARs Co-Operating – Loaning "spare"



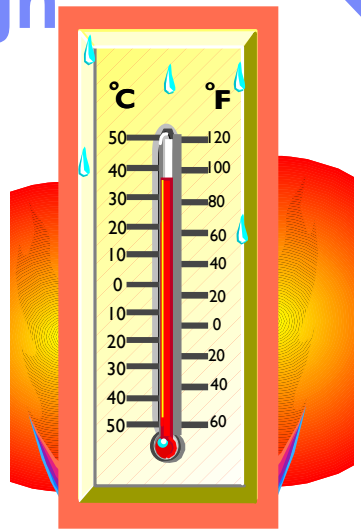
Other LPARs Co-Operating – Loaning "spare"



Further Co-Operating



AMS Algorithm 3 – Loans are not enough



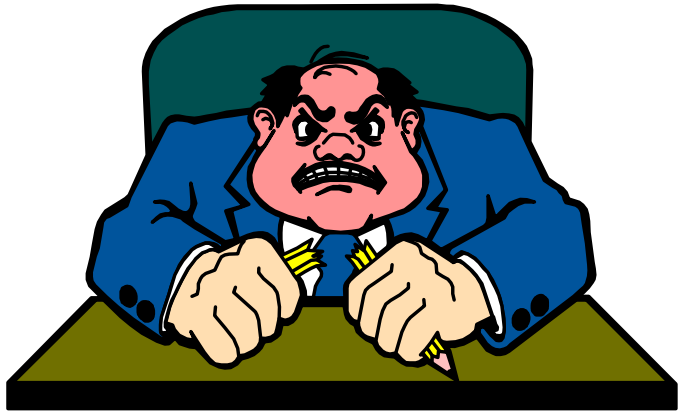
LPARs refuse to loan more memory

Hypervisor gets aggressive

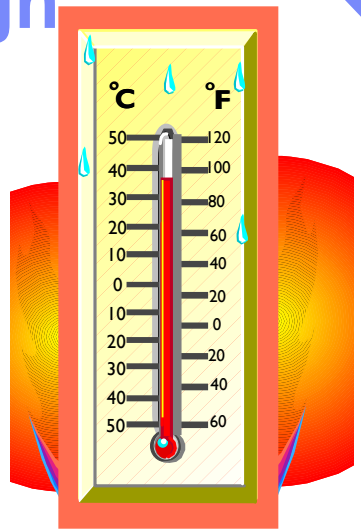
1. Steals some pages
   - It can see the page tables
   - It avoids critical memory pages
   - Use Least Recently Used page table data
2. Asks VIOS to page out LPAR memory
3. Once the memory page is free
4. Gives pages to high demand LPAR

LPARs are not aware of this happening

Aggressive Mode

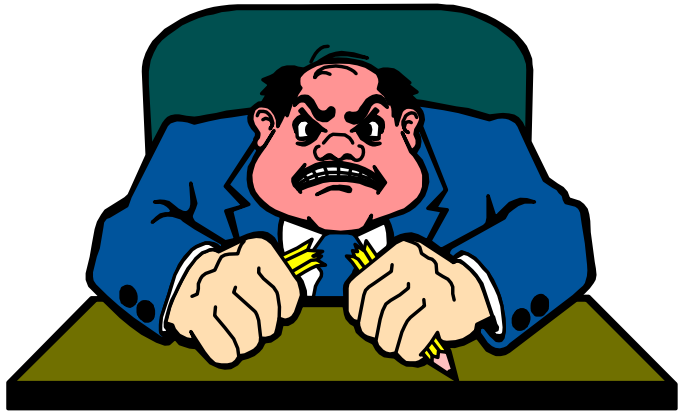


AMS Algorithm 3 – Loans are not enough



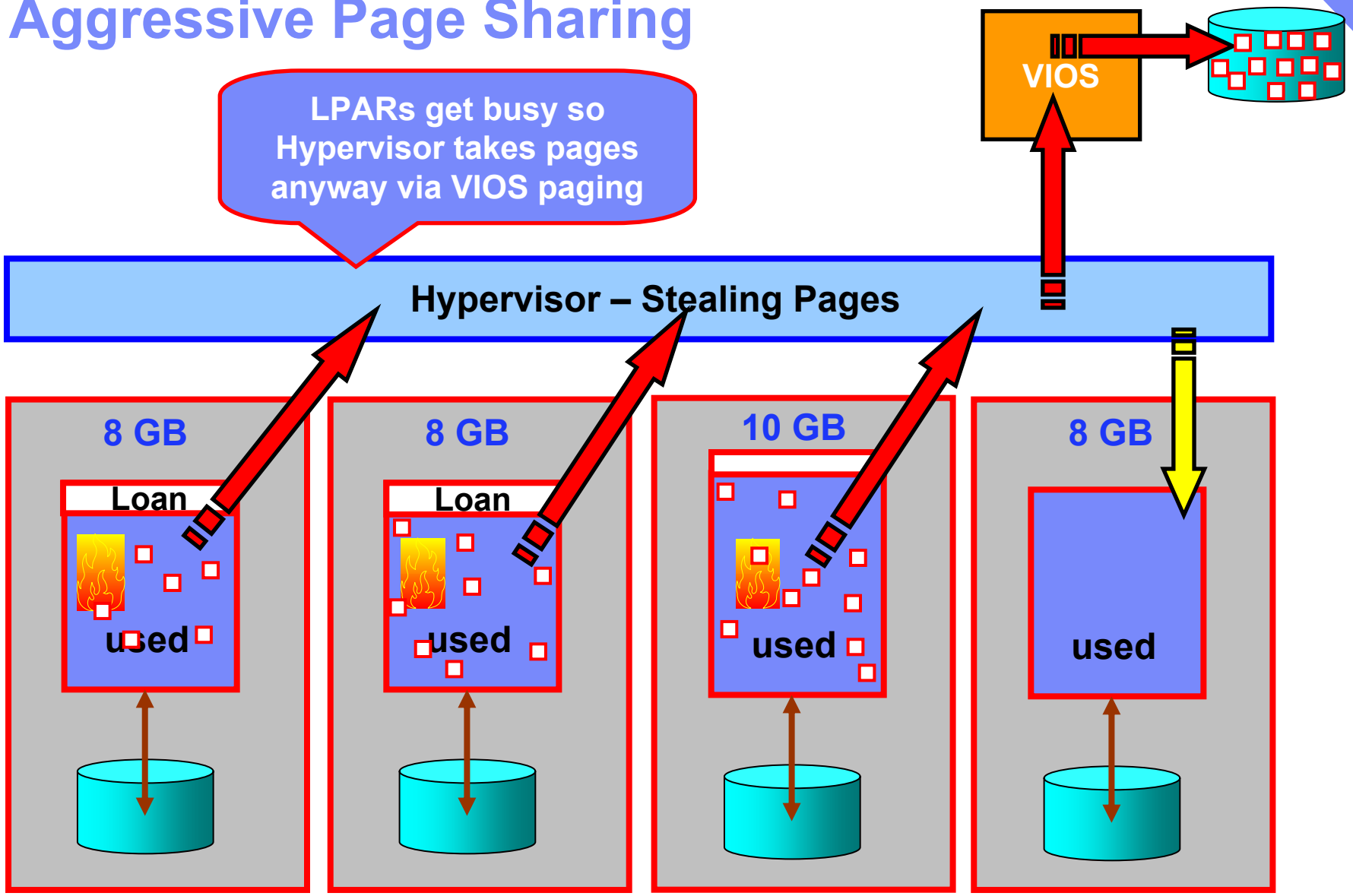
Now LPAR accesses a page that is not present

1. Causes page fault
2. Normally, Hypervisor hands interrupts to the LPARs to handle
3. Checks: if it's a Hypervisor paged pages
4. If yes, it recovers the page and restarts the instruction
5. If no, it passes the page fault onto AIX to handle as normal

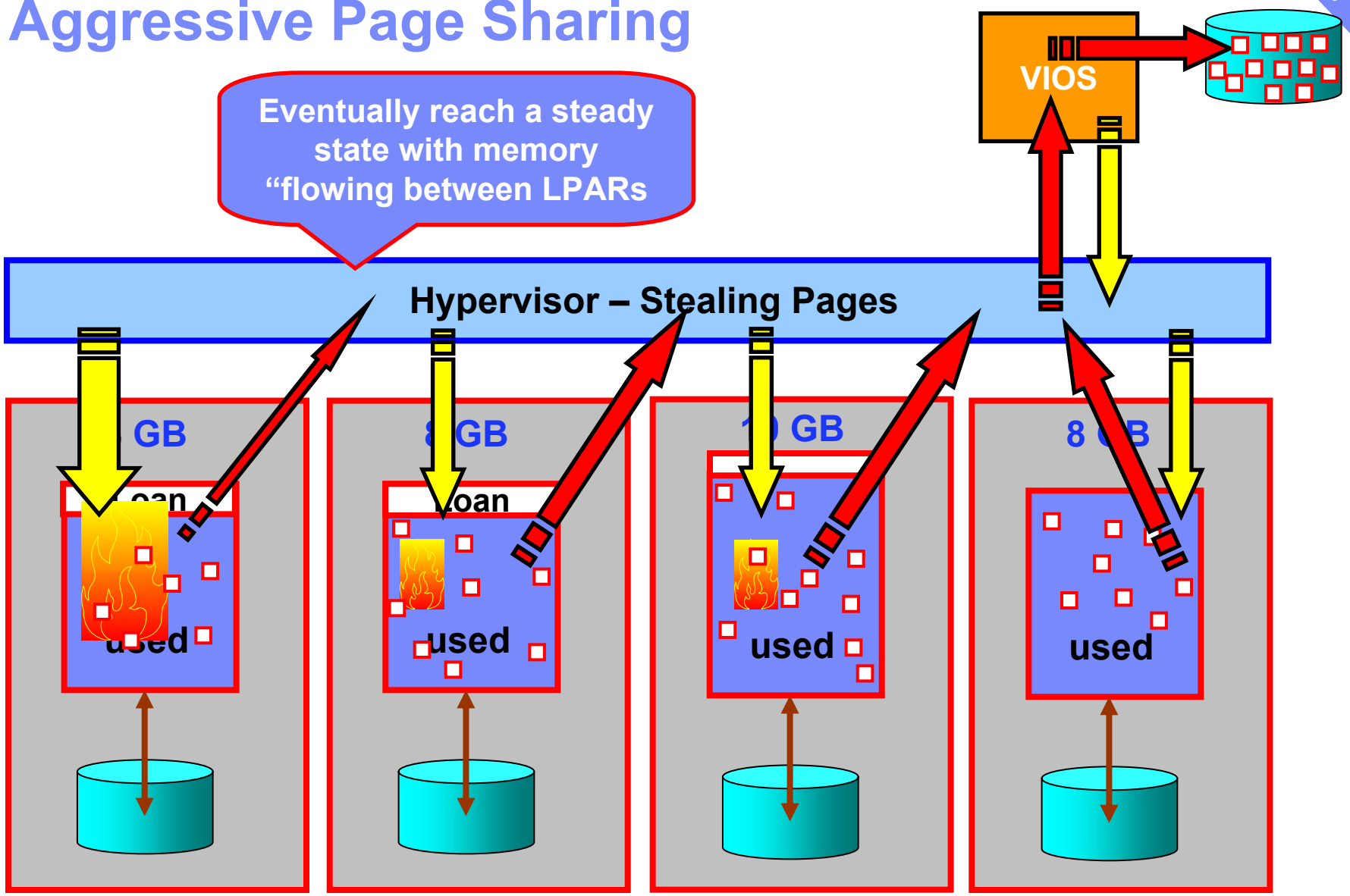


Aggressive Mode

Aggressive Page Sharing



Aggressive Page Sharing



AMS Algorithm 4 - Page Thrashing between LPARs

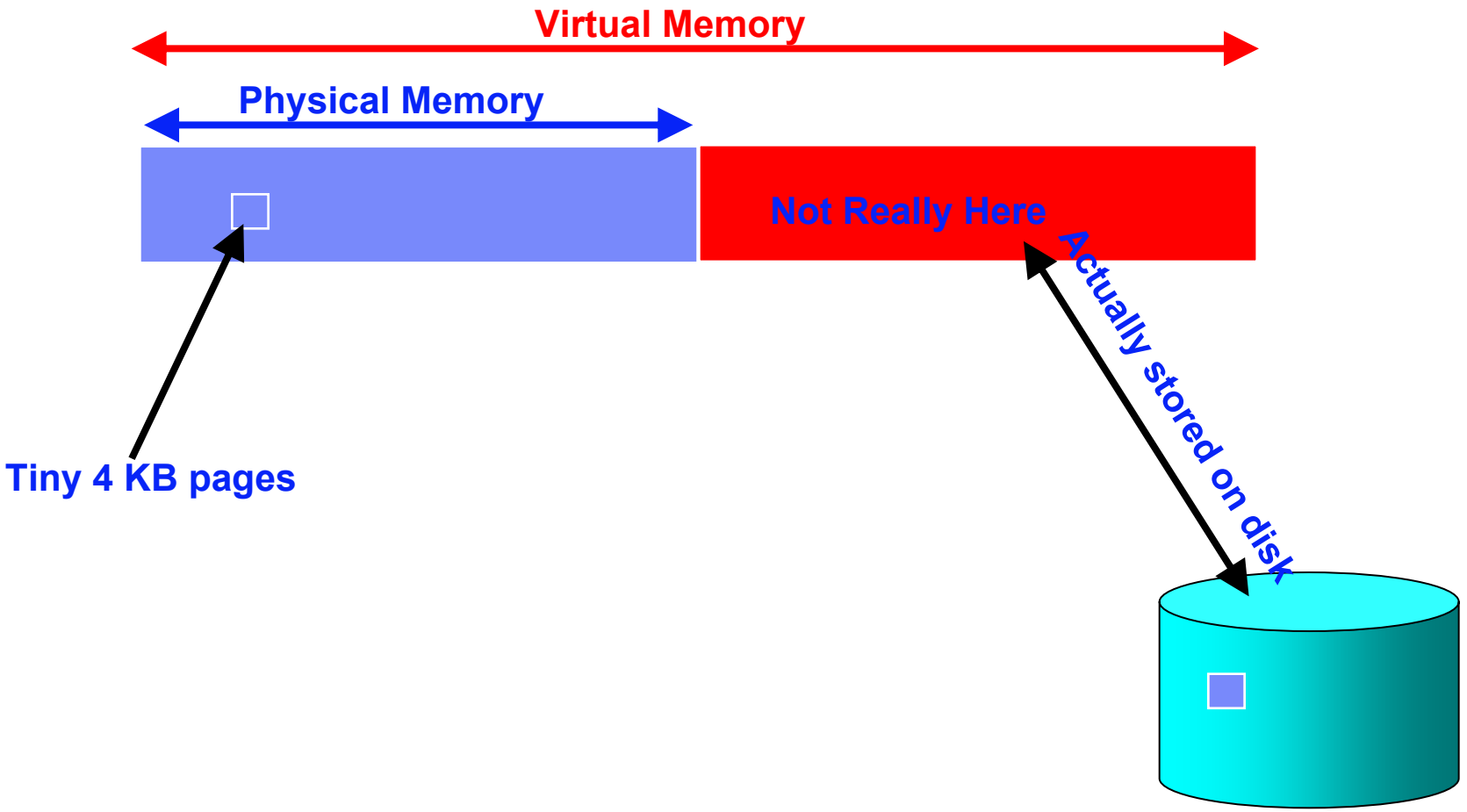
Alternative strategies to reduce this are

- Live with it – Spread Paging Space across more disks
- Add memory to the shared pool
  - If necessary, remove it from Dedicated memory LPARs
- Reduce memory requirements
  - Tune down application settings
- Power down an LPAR!
- Partition Mobility to other machine
- Buy more memory/CUoD

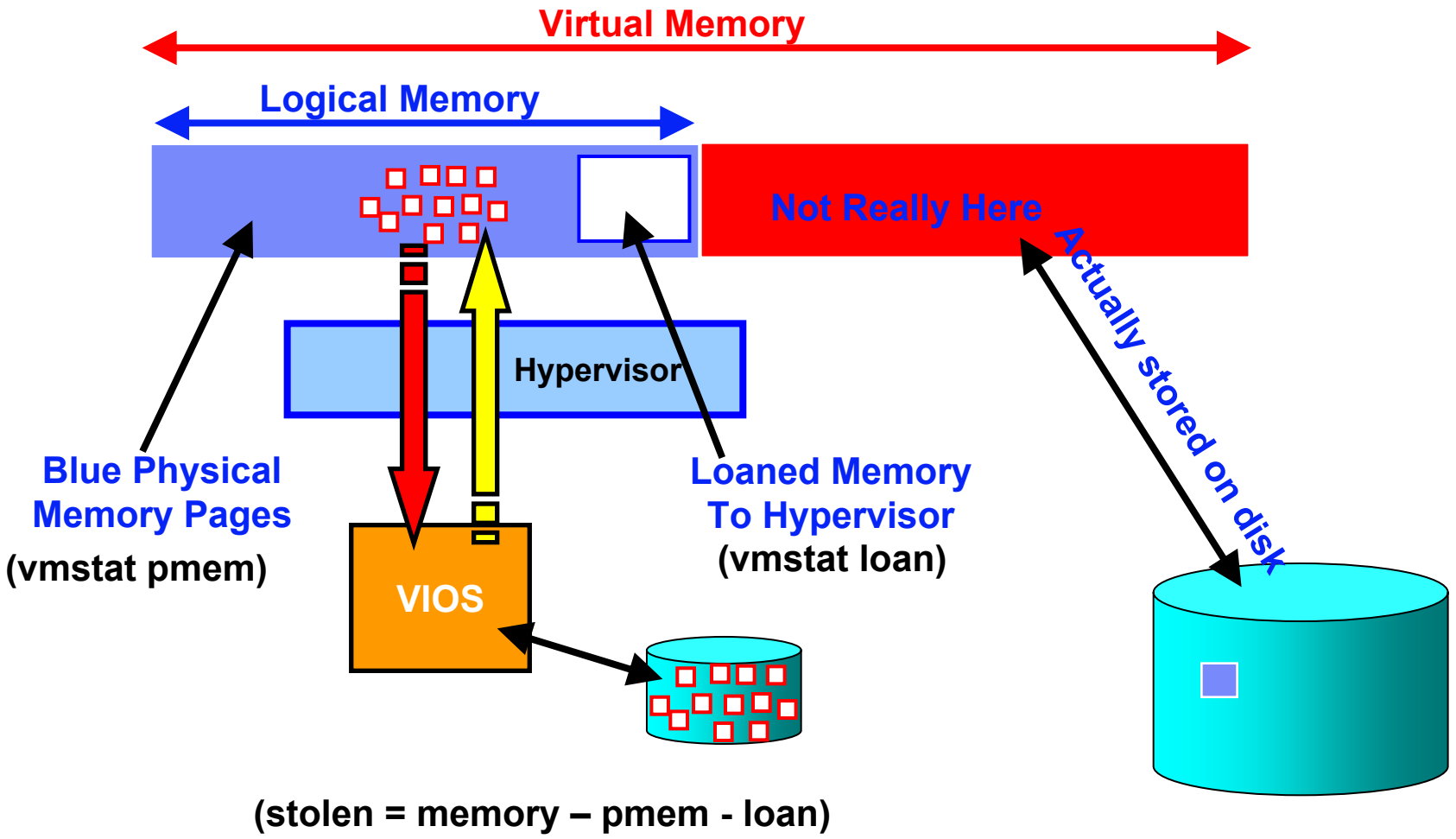
The Ugly but Obvious

- High, sustained memory residency requirements
  - High performance - HPC
  - RDBMS with 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

Classic Virtual Memory (LPAR)



Active Shared Virtual Memory (LPAR)



LPAR: vmstat -h

Logical Memory

Memory Pool Size

```
# vmstat -h 10

System configuration: lcpu=2 mem=2048MB ent=0.50 mmode=shared mpsz=4.00GB

kthr     memory              page              faults        cpu             hypv-page
----- ----------- ------------------------ ------------ ----------------------- ----------------------
r  b   avm   fre  re  pi  po  fr  sr  cy  in   sy  cs  us sy id wa   pc    ec  hpi  hpit  pmem  loan
0  0 190419 173073   0   0   0   0   0   0   2  149 159   0  1 99  0 0.01   1.6    0     0  1.20  0.80
0  0 190419 173073   0   0   0   0   0   0   2   24 152   0  0 99  0 0.00   0.8    0     0  1.20  0.80
0  0 190419 173073   0   0   0   0   0   0   1   19 166   0  0 99  0 0.00   0.8    0     0  1.20  0.80
0  0 207225 189696   0   0   0   0   0   0   6  334 196  35  2 64  0 0.18  36.9   25    53  1.33  0.67
0  0 207227 189694   0   0   0   0   0   0   2   39 170  50  1 50  0 0.25  50.8    7    15  1.33  0.67
0  0 207227 189694   0   0   0   0   0   0   5   20 164  50  1 50  0 0.25  50.8    0     0  1.33  0.67
.  .  .
.  .  .
```

Logical Memory Statistics

Hypervisor Page-ins Faults/s

Time waiting for hypervisor page-ins (in milliseconds)

Physical Memory → pmem

Loaned Memory

If ams_loan_policy=0 (off) this will be zero

CEC: topas –C (hit "g" for the extra top info)

```
Topas CEC Monitor                Interval:  10              Wed Dec  3 10:15:06 2008
Partition Info     Memory (GB)         Processor           Virtual Pools :      0
Monitored   :  4   Monitored   : 8.0   Monitored   :2.0     Avail Pool Proc:     3.7
UnMonitored:  -   UnMonitored:   -     UnMonitored:  -      Shr Physical Busy:   0.28
Shared      :  4   Available   :   -     Available   :  -     Ded Physical Busy:   0.00
Uncapped    :  4   UnAllocated:   -     UnAllocated:  -      Donated Phys. CPUs   0.00
Capped      :  0   Consumed    : 6.5     Shared      :  2     Stolen Phys. CPUs :  0.00
Dedicated   :  0                         Dedicated   :  0     Hypervisor
Donating    :  0                         Donated     :  0     Virt. Context Switch: 976
                                         Pool Size   :  4     Phantom Interrupts  :   1

Host          OS  M  Mem InU Lp  Us Sy Wa Id   PhysB  Vcsw Ent  %EntC PhI  pmem
------------------------------------shared-----------------------------------
silver_vios1 A61 U  2.0 1.9  8  08  1  0 90   0.01   873  0.40   2.6   0     -
silver_lpar2 A61 UM 2.0 1.4  2  49  1  0 49   0.26   244  0.50  51.3   1  1.30
silver_lpar3 A61 UM 2.0 1.6  2   0  0  0 99   0.01   294  0.50   1.4   0  0.78
silver_lpar4 A61 UM 2.0 1.7  2   0  0  0 99   0.01   220  0.50   1.3   0  0.87
silver_lpar5 A61 UM 2.0 1.7  2   0  0  0 99   0.01   218  0.50   1.1   0  1.01
```

Logical Memory → Mem

Inu→InUse LPAR Working Set

Shared memory LPAR
Physical Memory → pmem

Dedicated
Memory
LPAR

PVI11 Active Memory Sharing Session

60 Hands-On Movies AIX6, POWER6, PowerVM and other cool stuff!

Includes four on AMS with more to come

Google™ → **AIX movie**

nag@uk.ibm.com