hp®

invent

hp partitioning continuum

hp processor sets (psets)

agenda

✓ hp partitioning continuum

psets

✓ overview

✓ features

✓ benefits

✓ configuration

✓ application binding

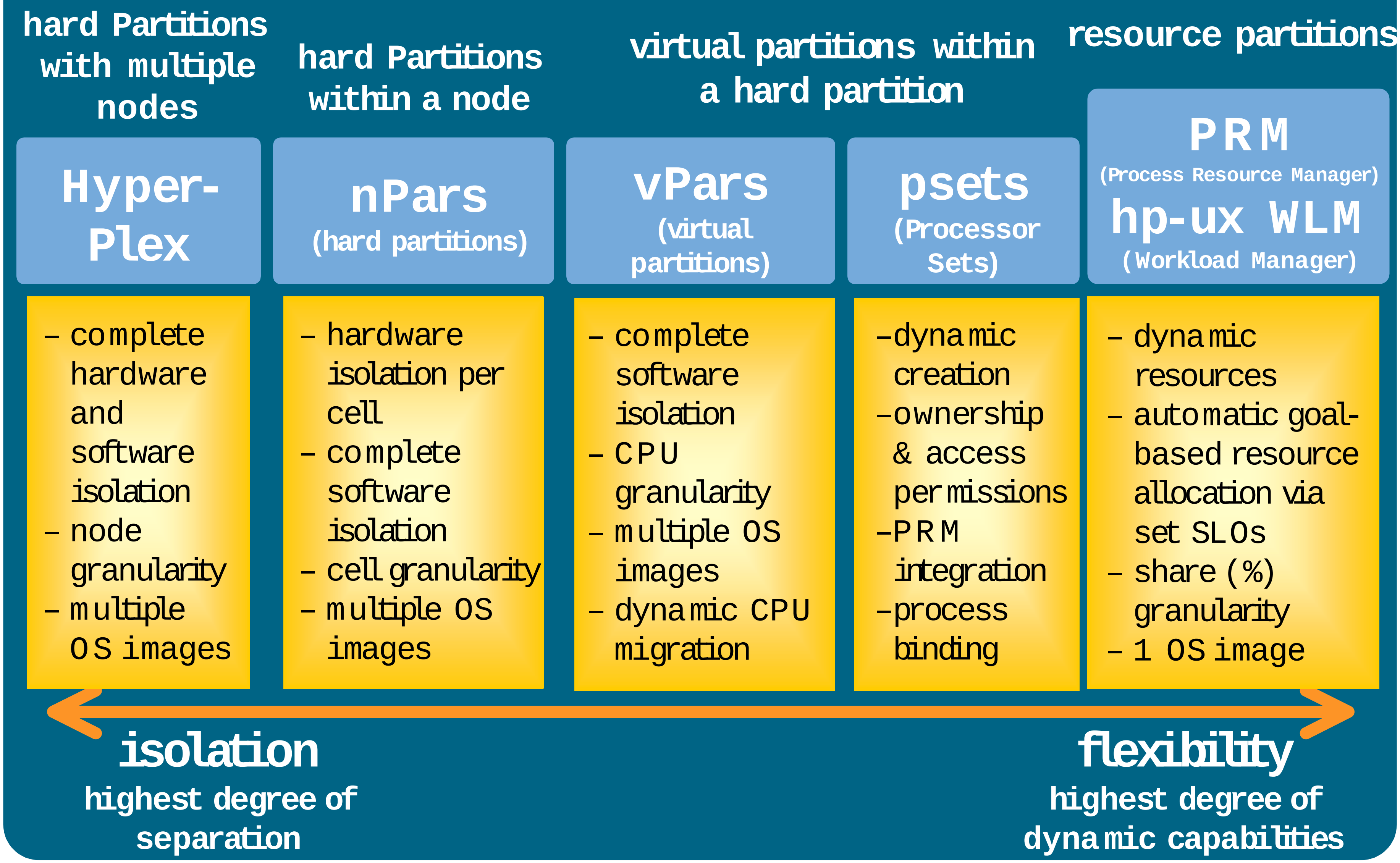
✓ access model & attributes

✓ user interface

✓ psets or vPars

hp partitioning continuum

technical positioning

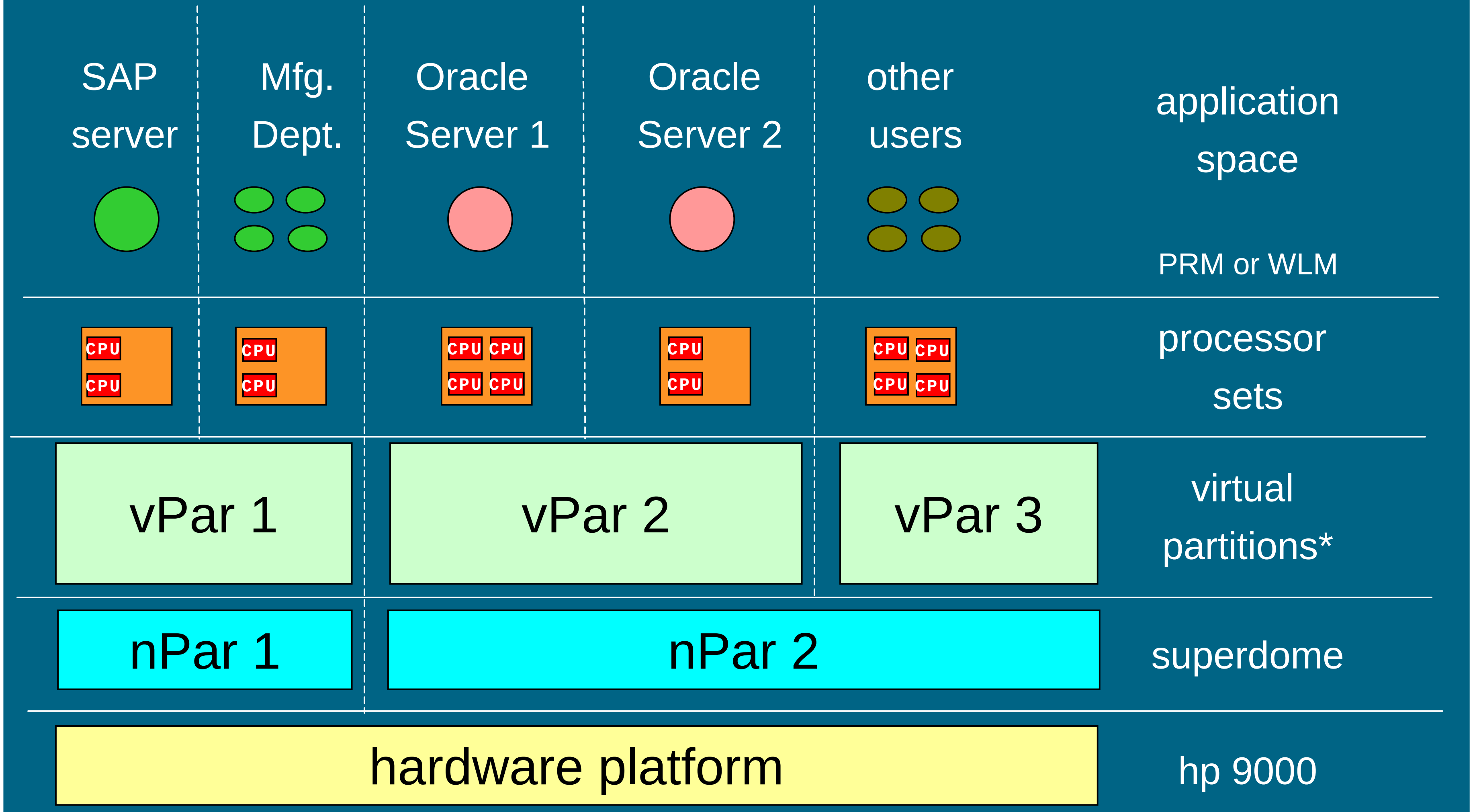


hard Partitions with multiple nodes	hard Partitions within a node	virtual partitions within a hard partition		resource partitions
Hyper-Plex	**nPars** (hard partitions)	**vPars** (virtual partitions)	**psets** (Processor Sets)	**PRM** (Process Resource Manager) **hp-ux WLM** (Workload Manager)
– complete hardware and software isolation – node granularity – multiple OS images	– hardware isolation per cell – complete software isolation – cell granularity – multiple OS images	– complete software isolation – CPU granularity – multiple OS images – dynamic CPU migration	–dynamic creation –ownership & access permissions –PRM integration –process binding	– dynamic resources – automatic goal-based resource allocation via set SLOs – share (%) granularity – 1 OS image

isolation
highest degree of separation

flexibility
highest degree of dynamic capabilities

psets within hp partitioning continuum



SAP server

Mfg. Dept.

Oracle Server 1

Oracle Server 2

other users

application space

PRM or WLM

processor sets

vPar 1

vPar 2

vPar 3

virtual partitions*

nPar 1

nPar 2

superdome

hardware platform

hp 9000

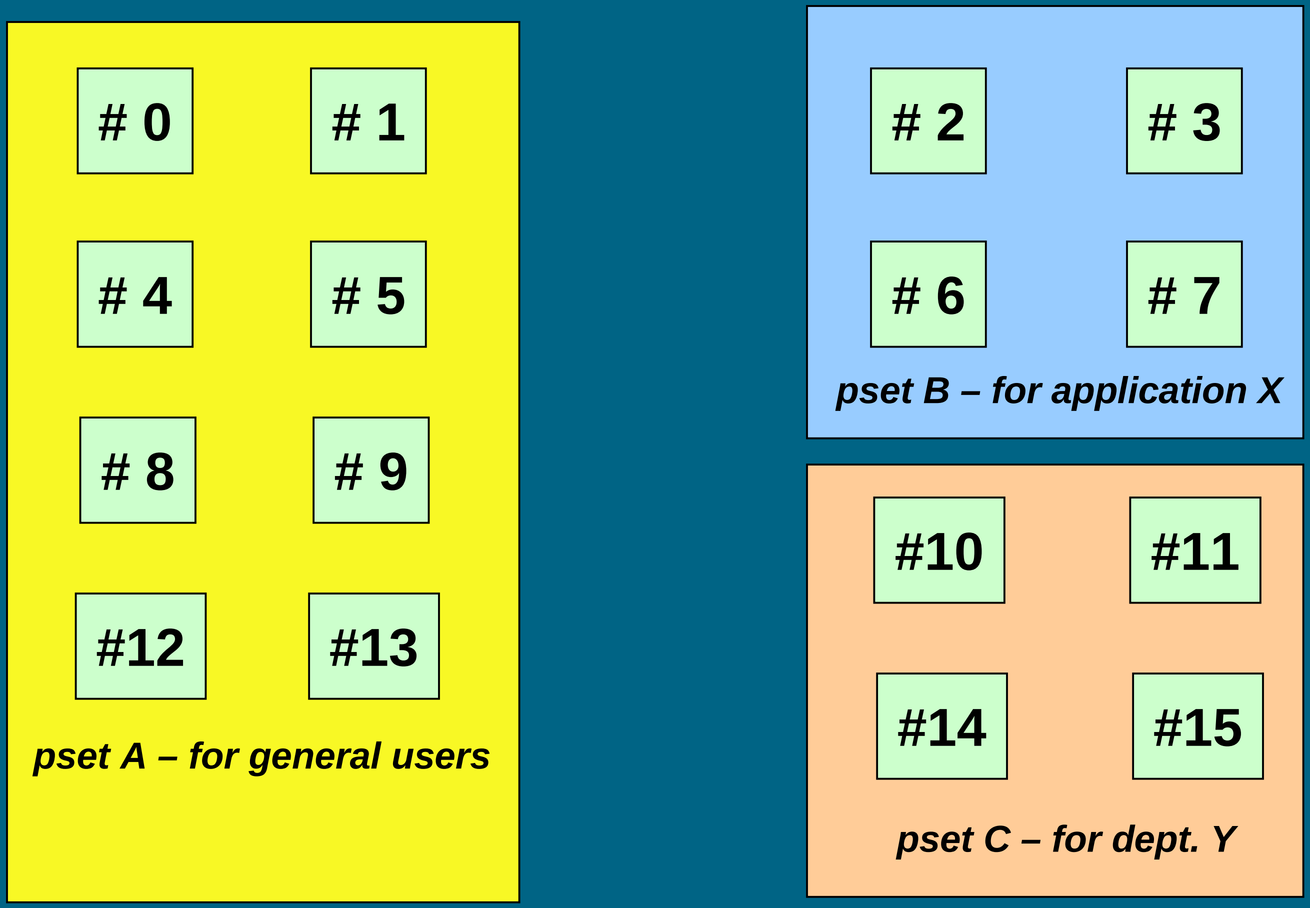
* vPars available only on L, N Class and Superdome

hp processor sets (psets) overview

- represents a group of processors in the system
- represents a scheduling allocation domain
- provides a mechanism for CPU resource management
- provides CPU resource isolation for applications and users
- does not provide fault isolation
- available free for any HP-UX 11.11 licensed system
- available Q401

the system may be configured into more than one processor set

hp processor sets
example pset configuration



0 # 1

4 # 5

8 # 9

#12 #13

pset A – for general users

2 # 3

6 # 7

pset B – for application X

#10 #11

#14 #15

pset C – for dept. Y

hp processor sets (psets) features

- dynamic creation, deletion, and reconfiguration of psets
- dynamic migration of threads and processes across psets
- ownership and access permissions for psets
- attributes to control psets behavior under different conditions
- processors are assigned to one pset at a time
- processes and threads have binding to one pset at a time
- system default pset for default users
- integration with PRM and gang scheduler

hp processor sets (psets) benefits

- server consolidation
  - *assign dedicated set of processors to a set of applications to take advantage of locality and to prevent interference between applications*
  - *processor resource partitioning among different departments or user groups in an organization*
- integration with HP process resource manager (PRM) and work load manager (WLM)
- dedicated processor resources for a job in batch processing
- special needs can be met
  - *isolation of processors can help support real-time applications*
- hardware and platform independent
  - *11i customer can use psets on all existing multi-processor hardware*

hp processor sets (psets) configuration

- dynamic creation of new processor sets
- dynamic deletion of existing processor sets
- dynamic reassignment of a processor from one pset to another
- any processor (except processor 0) can be reassigned across processor sets
- need appropriate privileges

hp processor sets (psets)
ownership and access permissions

- superuser and privileged group users may perform all pset operations
- every processor set has an owner
- users are divided into owner, group and others (similar to file system)
- there are READ, WRITE and EXEC permissions for users of each category
- user needs READ access to query processor set attributes
- user needs WRITE access to change processor set configuration & attributes
- user needs EXEC access to run applications in a processor set

hp processor sets (psets)
application binding to pset

- every thread and process has binding to a pset
- pset binding determines which processors a thread may execute on
- pset binding of a thread or process can be changed dynamically with appropriate privileges
- thread or process may further bind to a specific processor or a locality domain within a pset to exploit locality
- all processes with same user id or in same process group can be migrated to another pset with a single request

hp processor sets (psets)
application binding to pset (2)

- all threads of a process need not be bound to same pset
- migration of a process to another processor set will result in migration of all its threads.
- a child process inherits its pset binding from its parent process on creation.
- new threads in a multithreaded process inherit their pset binding from the creator thread.

hp processor sets (psets)
system default pset

- the default pset is created at system initialization time
- all processors, by default, are assigned to the Default pset
- processor 0 is always assigned to the Default pset, and cannot be reassigned to another processor set
- all other processors can be reassigned in and out of the Default pset
- the default pset has default values for all attributes, and they cannot be changed
- the default pset is always available to all applications and users in the system
- superuser owns the default pset

hp processor sets (psets)
attributes for better management

- ownership and access permissions
- attempt to remove last processor from a processor set
    - migrate workload to default pset
    - fail the request
- attempt to destroy a busy or populated processor set
    - migrate workload & processors to default pset
    - fail the request
- attempt to migrate application to an empty processor set
    - fail the request
- processor availability to handle I/O interrupts
    - available by default
    - redistribute interrupts to other processors in system

hp processor sets (psets)
user interfaces

pset_ create()	create a new pset
pset_ destroy()	destroy a processor set
pset_ assign()	reassign a processor from one pset to another
pset_ bind()	migrate a thread or a process from one pset to another
pset_ setattr()	change pset attributes
pset_ getattr()	query pset attributes
pset_ ctl()	query pset configuration
psrset	command line interface

hp processor sets (psets)
pset scheduler impact

- psets define new scheduling allocation domains
- posix realtime scheduler works on pset boundary
- PRM fair share scheduler works in default pset (at least in first release)
- gang scheduler works in default pset (at least in first release)
- load balancer works on pset boundary
- no load balancing by system across psets
- system (kernel) daemons are free to run anywhere in the system

hp processor sets (psets)
why psets when vPars are available?

- vPars are supported only on new platforms (L, N class and Superdome)
- each vPar executes as a separate hp-ux instance, and feels like a separate system. applications in different vPars need to interact through networking with each other.
- psets provide only processor partitioning without the memory and I/O partitioning which is what some applications and users need or care for.
- psets are excellent light-weight alternative when user cares only about processor resource partitioning.
- psets are more flexible and light-weight in dynamic reconfiguration.
- an application can be migrated from one pset to another dynamically, which is not allowed with vPars.
- psets provide applications with single system image (SSI)
- vPars require system management as they are separate systems.
- psets are tightly integrated with PRM

hp processor sets (psets)

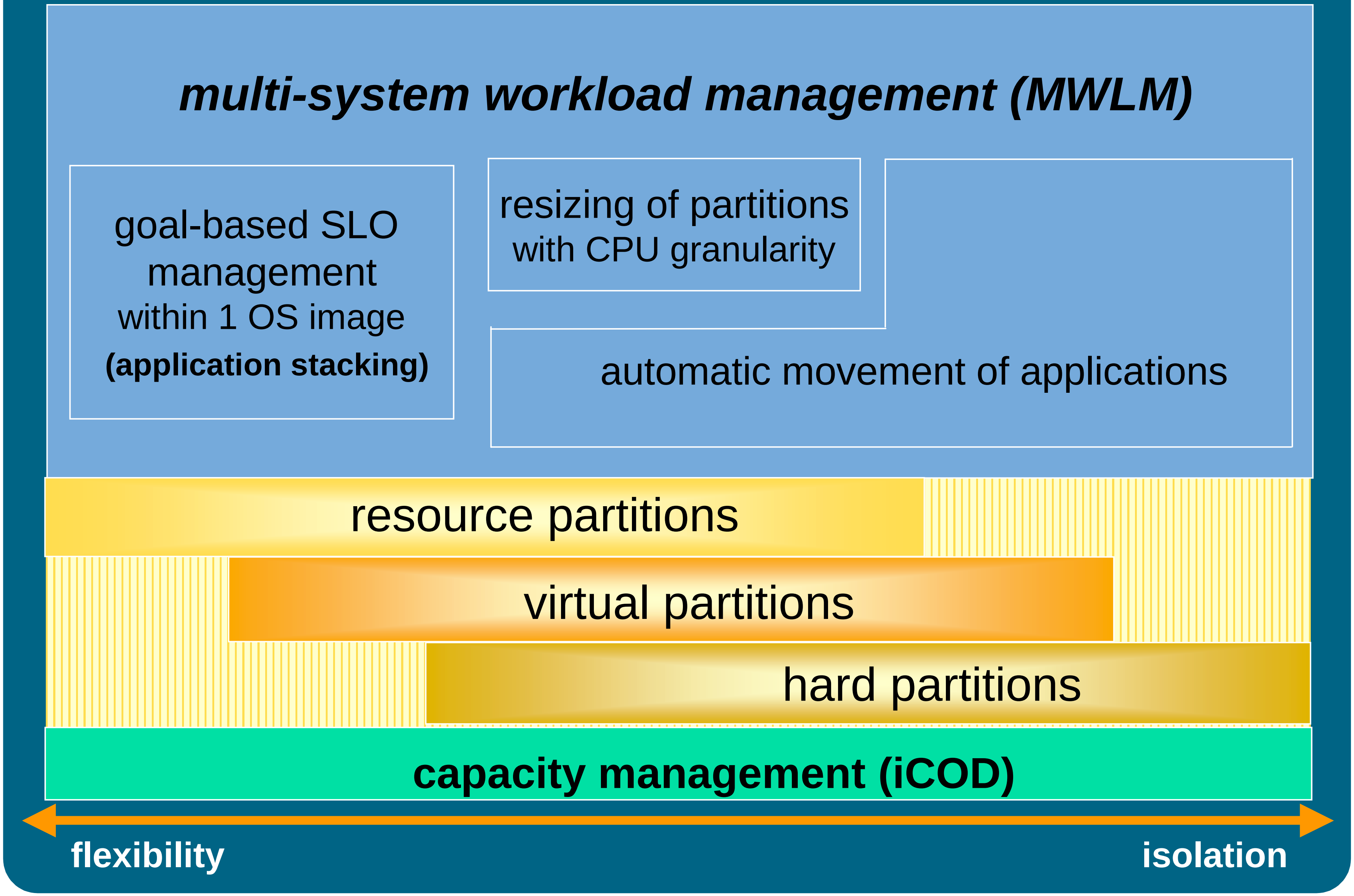
further information

- http://www.hp.com/go/hpux
- Information on other hpux features is available at this site

hp®

invent

hp partitioning continuum



multi-system workload management (MWLM)

goal-based SLO management within 1 OS image **(application stacking)**

resizing of partitions with CPU granularity

automatic movement of applications

resource partitions

virtual partitions

hard partitions

capacity management (iCOD)

flexibility

isolation

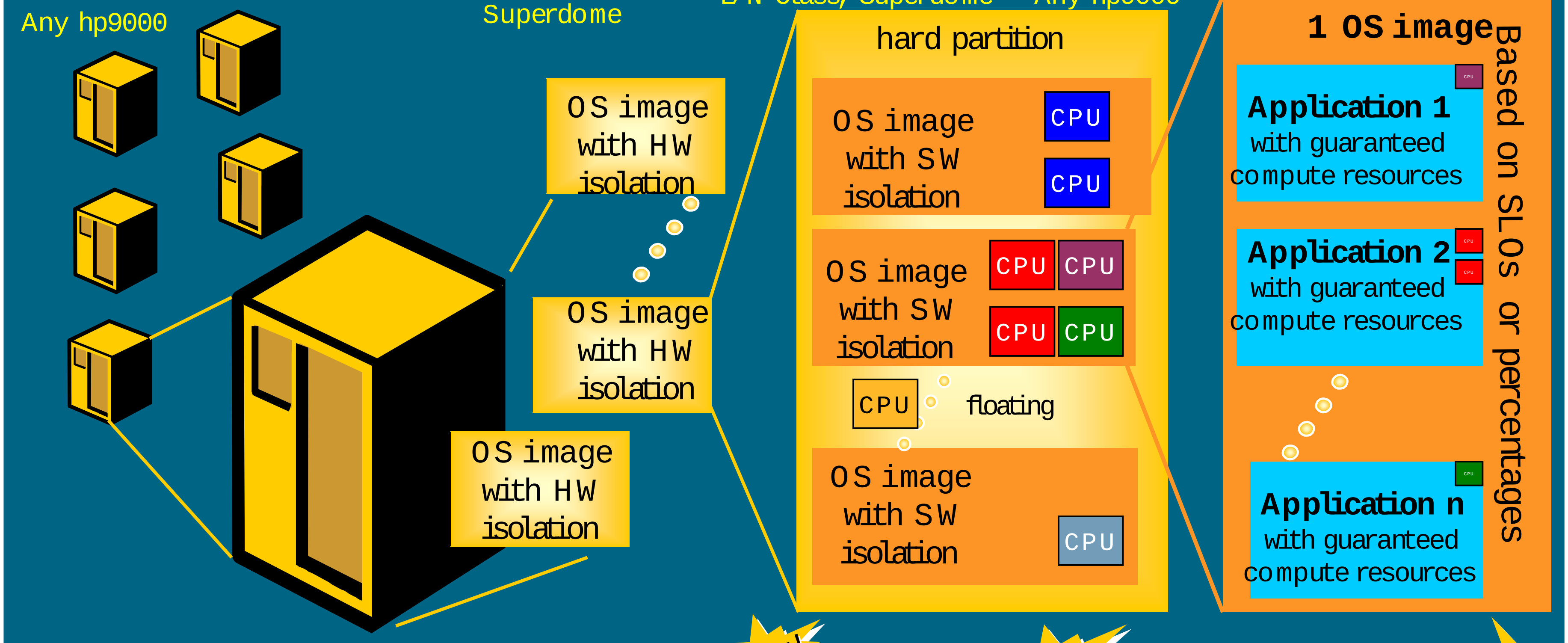
hp partitioning continuum

hard partitions with multiple nodes
(1 OS image per node)
Any hp9000

hard partitions within a node
(multiple OS images with SW/HW isolation)
Superdome

virtual partitions w/in a hard partition
(multiple OS images SW isolation)
L/N-Class, Superdome Any hp9000

resource partitions within an OS image
(resource allocation by app)
Any hp9000



OS image with HW isolation

OS image with HW isolation

OS image with HW isolation

hard partition

OS image with SW isolation CPU CPU

OS image with SW isolation CPU CPU CPU CPU

CPU floating

OS image with SW isolation CPU

1 OS image

Application 1 with guaranteed compute resources

Application 2 with guaranteed compute resources

Application n with guaranteed compute resources

Based on SLOs or percentages

HyperPlex

nPartitions

New! **Virtual partitions**

New! **psets** (Processor Sets)

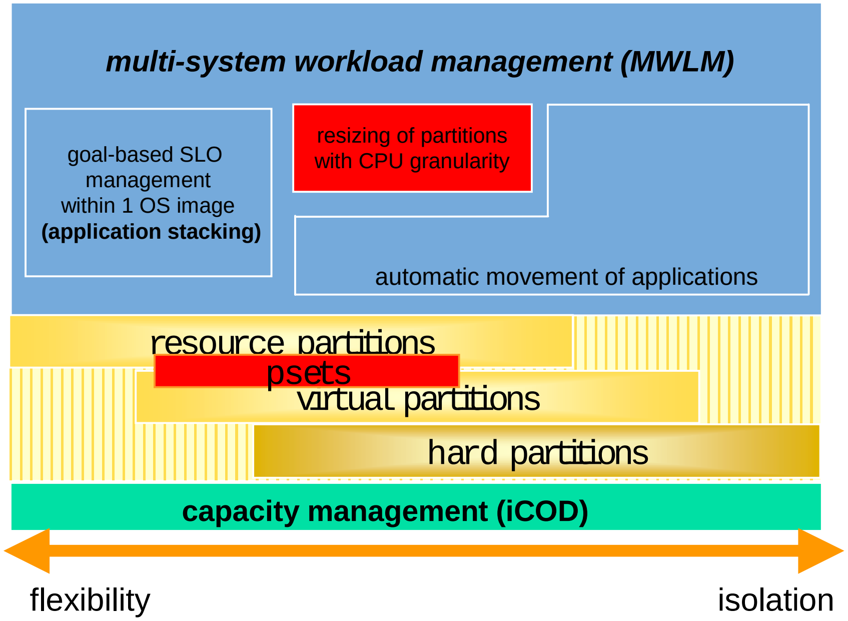
New Release **PRM and hp-ux WLM**

+Isolation

+Flexibility

hp partitioning continuum

technology overview



multi-system workload management (MWLM)

goal-based SLO management within 1 OS image **(application stacking)**

resizing of partitions with CPU granularity

automatic movement of applications

resource partitions

psets

virtual partitions

hard partitions

capacity management (iCOD)

flexibility

isolation

hp processor sets

(psets)

pset operation permissions

<table>
<tr><th>pset operations</th><th>Superuser</th><th>PRIV_pset</th><th>other users with pset permissions</th><th>All Users</th></tr>
<tr><td><em>create pset</em></td><td><em>YES</em></td><td><em>YES</em></td><td><em>N/A (pset is yet to be created)</em></td><td><em>YES</em></td></tr>
<tr><td><em>destroy pset</em></td><td><em>YES</em></td><td><em>YES</em></td><td><em>YES with WRITE permission</em></td><td><em>NO</em></td></tr>
<tr><td><em>reasign a processor to another pset</em></td><td><em>YES</em></td><td><em>YES</em></td><td>
<table>
<tr><th><em>source pset</em></th><th><em>target pset</em></th><th><em>permissions needed</em></th></tr>
<tr><td>default</td><td>non default</td><td>N/A (cannot have WRITE permission in default pset)</td></tr>
<tr><td>non default</td><td>non default</td><td>YES with WRITE permission in both psets</td></tr>
<tr><td>non default</td><td>default</td><td>YES with WRITE permission in source pset</td></tr>
</table>
</td><td><em>NO</em></td></tr>
<tr><td><em>bind threads</em></td><td><em>YES</em></td><td><em>YES</em></td><td><em>YES with EXEC permission</em></td><td><em>NO</em></td></tr>
<tr><td><em>set pset attributes (except owner, group, iointr, and access permission)</em></td><td><em>YES</em></td><td><em>YES</em></td><td><em>YES with WRITE permission</em></td><td><em>NO</em></td></tr>
<tr><td><em>set owner, group, access permissions</em></td><td><em>YES</em></td><td><em>YES</em></td><td><em>only the pset owner</em></td><td><em>NO</em></td></tr>
<tr><td><em>enable/disable iointr attribute</em></td><td><em>YES</em></td><td><em>YES</em></td><td><em>NO</em></td><td><em>NO</em></td></tr>
<tr><td><em>pset_ctr</em></td><td><em>YES</em></td><td><em>YES</em></td><td><em>YES</em></td><td><em>YES</em></td></tr>
</table>