Systems and Technology Group

z/VM Support for ILMT and CPU Pooling

Romney White
System z Architecture and Technology

Agenda

- **System z Software Pricing**
- **Linux Guest Software Pricing Without CPU Pooling**
- **Linux on z and ILMT**
- **CPU Pooling**
  - Approach
  - Externals
- **New Interfaces**
- **Linux Guest Software Pricing with CPU Pooling**
- **Customer Use Cases**

System z software pricing methodology objectives

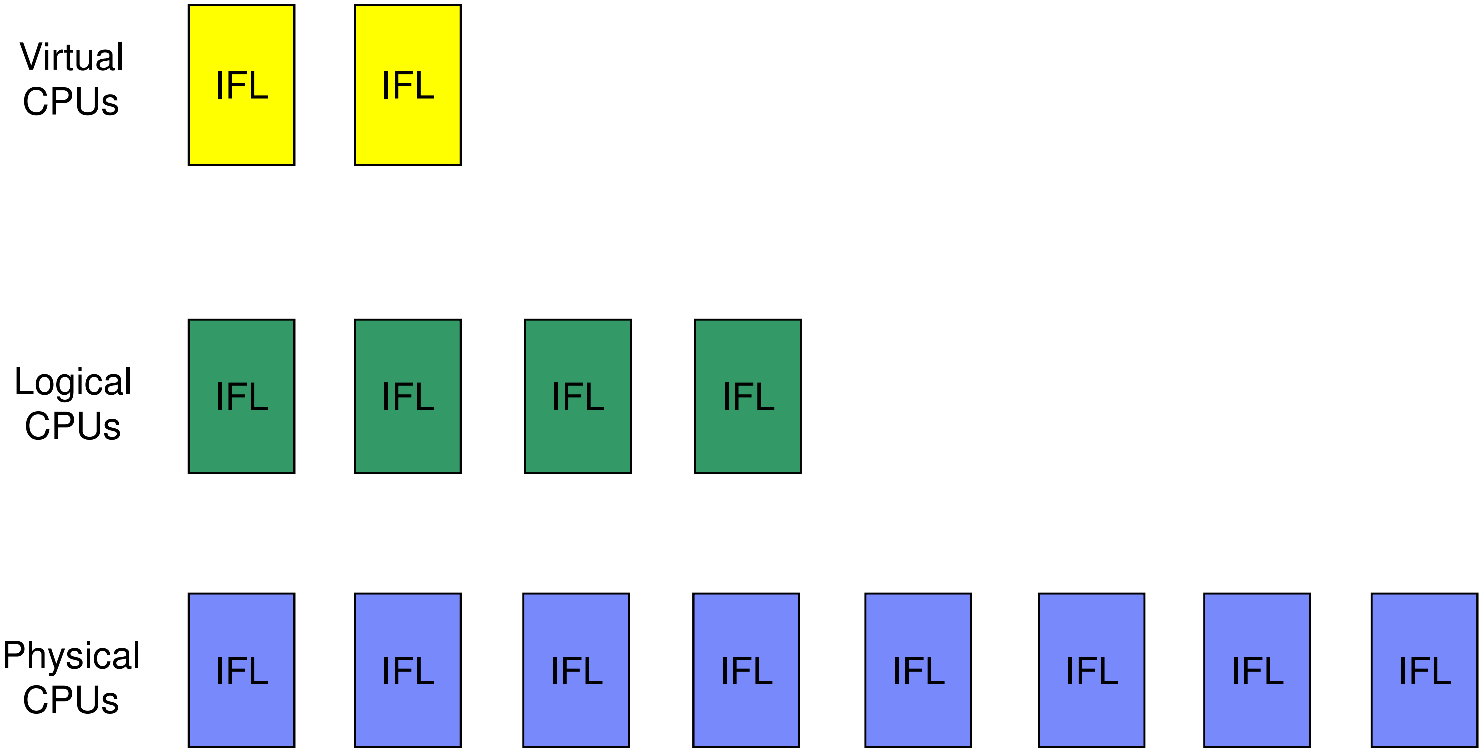
- **Price-to-value**
- **Flexibility to run software where it is most efficient**
- **Capability to predict software charges**
- **Help with cost of new applications**
- **Flexibility to pay for software based on workload requirements**

Pricing metrics for z/VM IPLA products

- **z/VM V5 and V6 and certain z/VM middleware products have pricing based on the number of engines**
  - *Engine-based Value Unit* pricing allows for a lower cost of incremental growth as additional engine-based licenses are purchased
- **Most IBM middleware for Linux is also priced based on the number of engines**
  - The number of engines is converted into *Processor Value Units* (PVUs) under the Passport Advantage® terms and conditions
- **z/VM 6.3 (with APARs) allows *CPU pooling***
  - *ILMT enhancements* available August 12, 2014 enable using ILMT with pooling

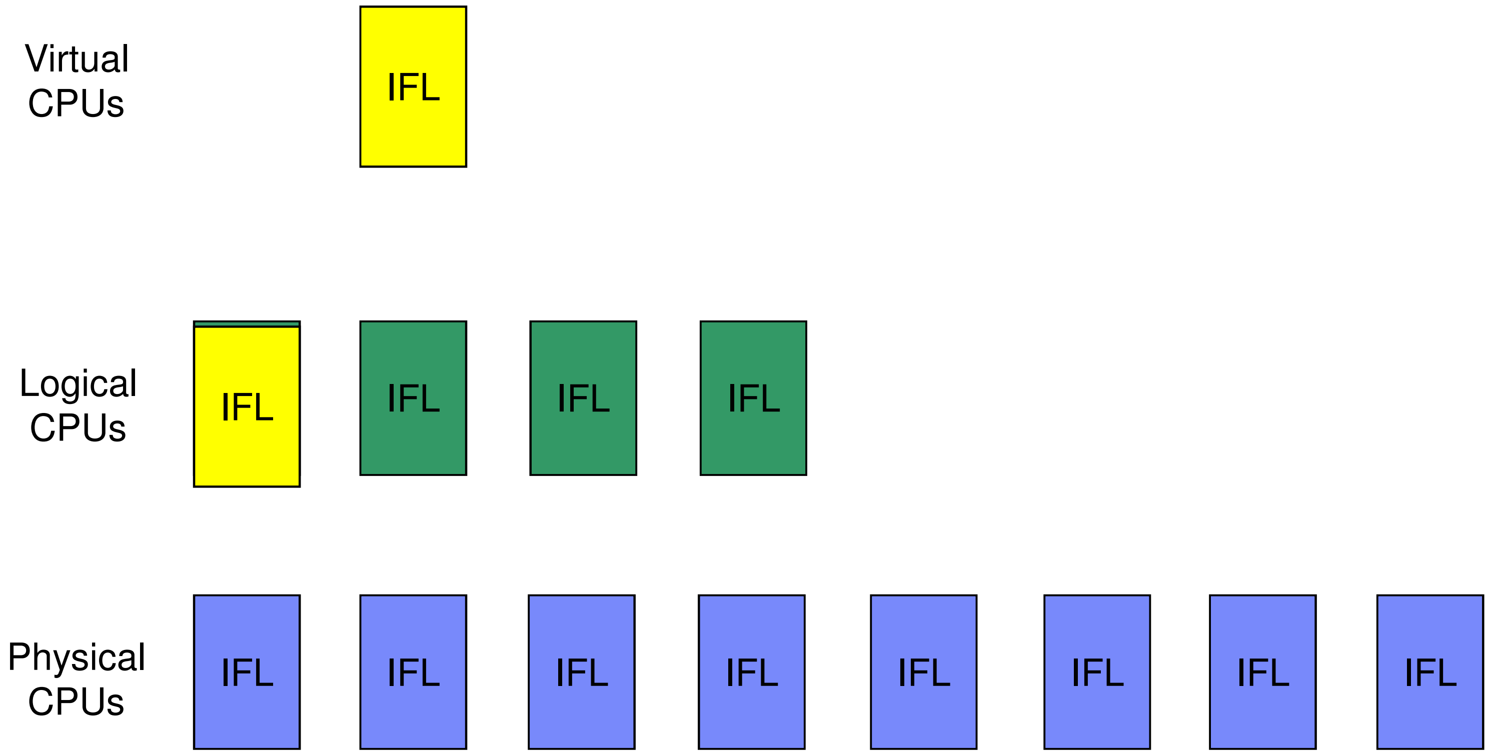
Linux Guest Software Pricing Without CPU Pooling

Pricing rule for products in z/VM guests: The lower of the sum of the virtual engines available to guests running a product or the engine capacity of the z/VM LPAR from which the guests obtain their resources.



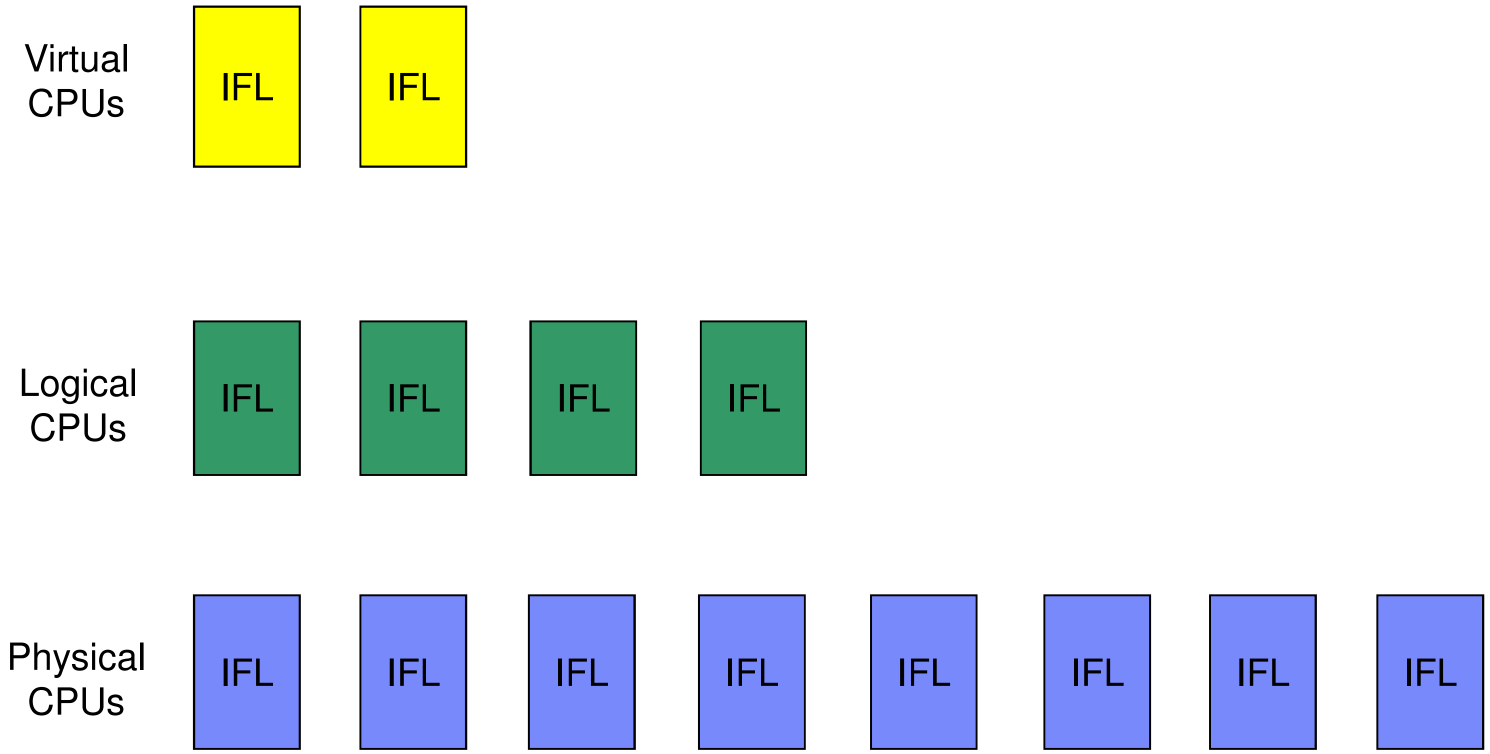
Linux Guest Software Pricing Without CPU Pooling

Pricing rule for products in z/VM guests: The lower of the sum of the virtual engines available to guests running a product or the engine capacity of the z/VM LPAR from which the guests obtain their resources.



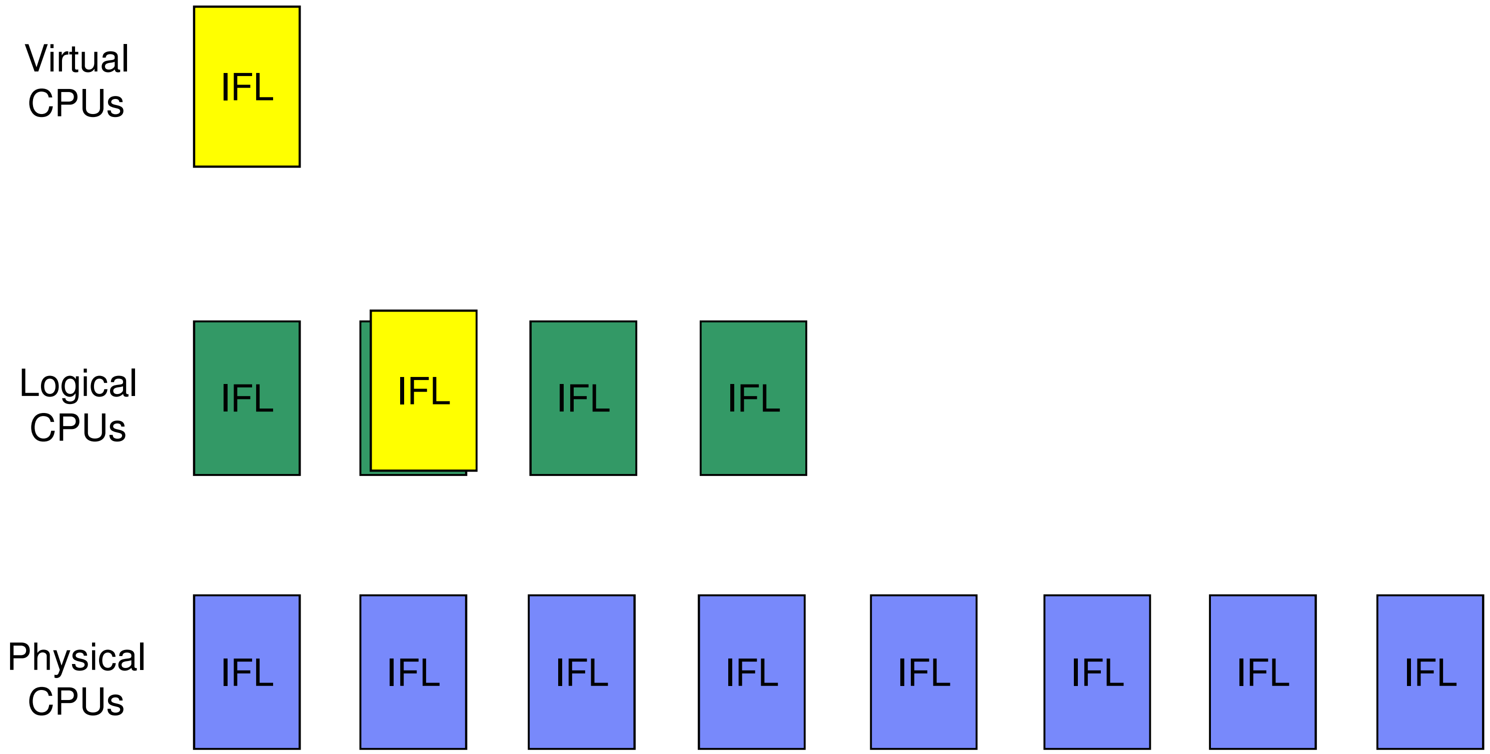
Linux Guest Software Pricing Without CPU Pooling

Pricing rule for products in z/VM guests: The lower of the sum of the virtual engines available to guests running a product or the engine capacity of the z/VM LPAR from which the guests obtain their resources.



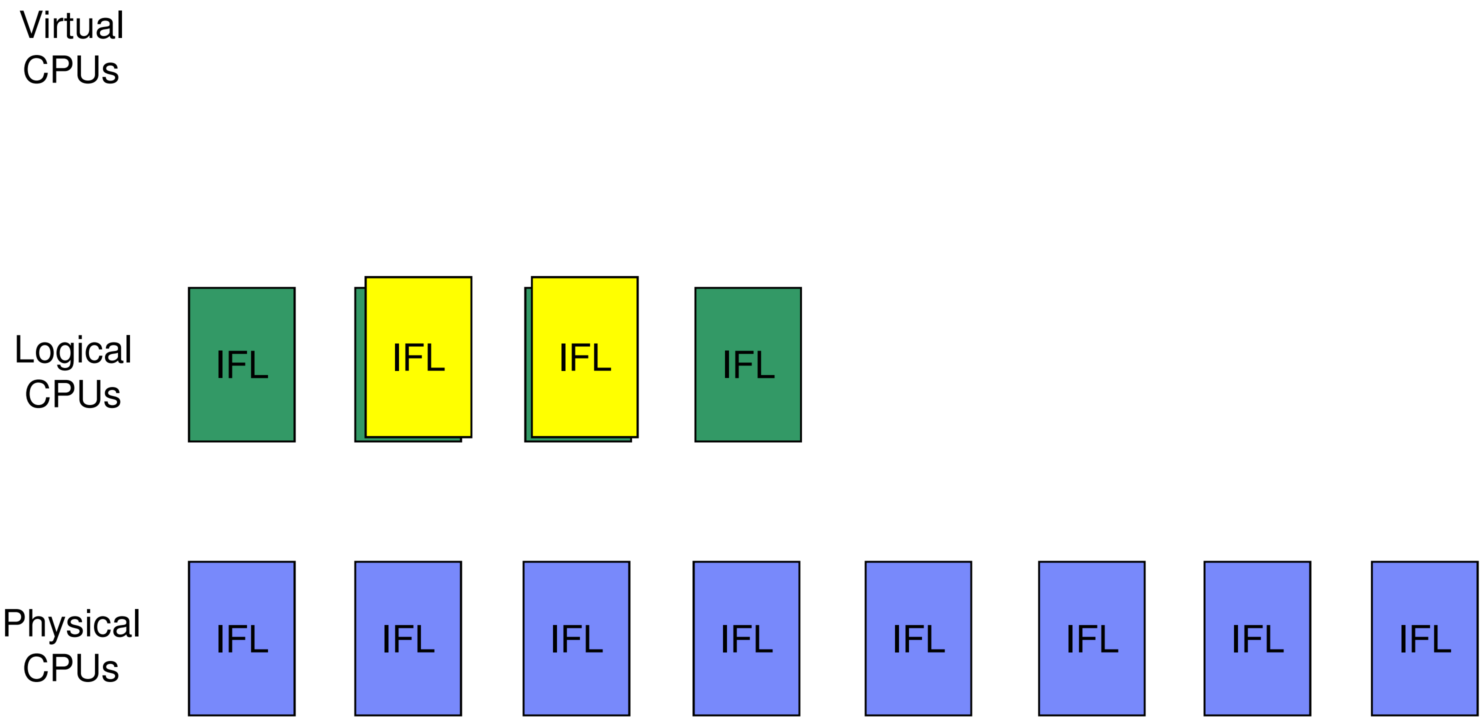
Linux Guest Software Pricing Without CPU Pooling

Pricing rule for products in z/VM guests: The lower of the sum of the virtual engines available to guests running a product or the engine capacity of the z/VM LPAR from which the guests obtain their resources.



Linux Guest Software Pricing Without CPU Pooling

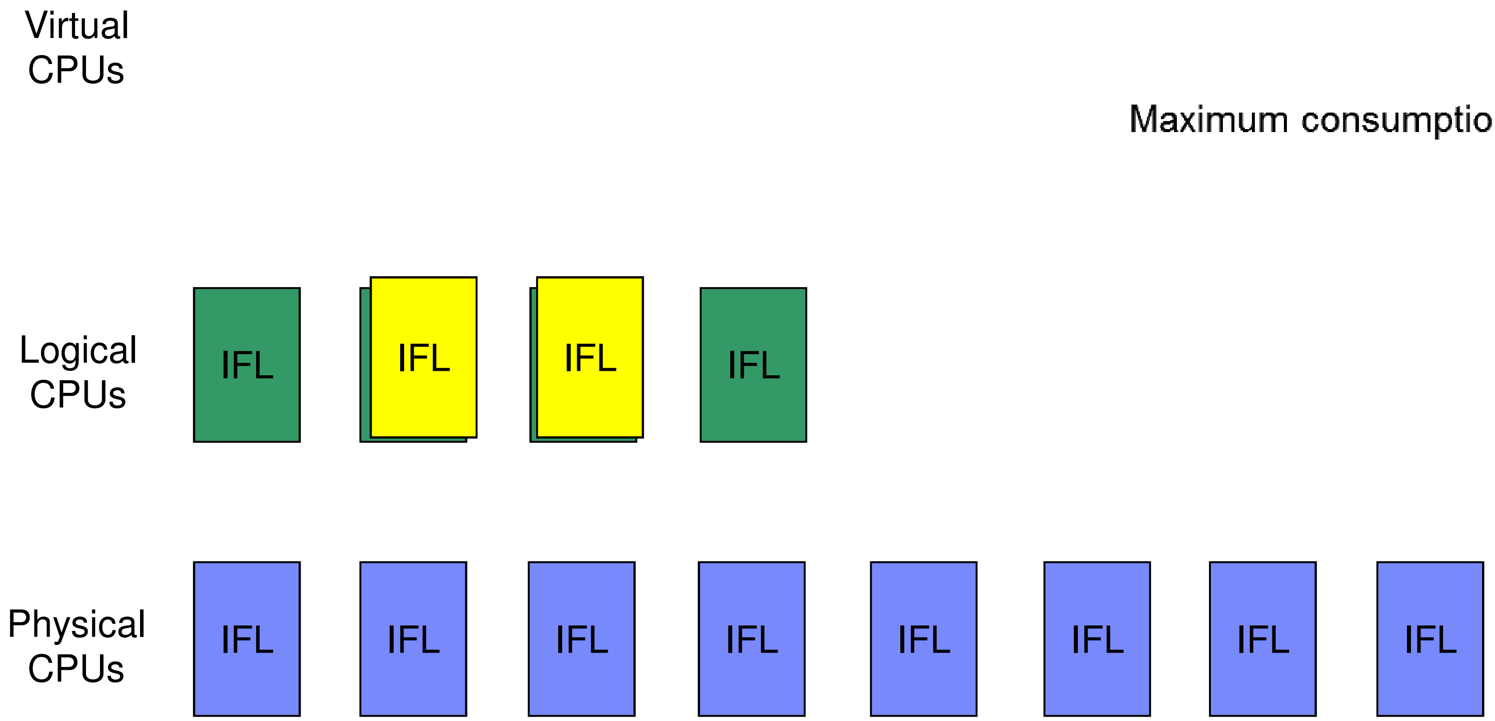
Pricing rule for products in z/VM guests: The lower of the sum of the virtual engines available to guests running a product or the engine capacity of the z/VM LPAR from which the guests obtain their resources.



Linux Guest Software Pricing Without CPU Pooling

Pricing rule for products in z/VM guests: The lower of the sum of the virtual engines available to guests running a product or the engine capacity of the z/VM LPAR from which the guests obtain their resources.

Maximum consumption: 2 IFLs



Linux on z and ILMT (IBM License Metric Tool)

- **Sub-capacity pricing required ILMT agent in each Linux guest**
  - Agent had to be installed and manually configured
  - Activation could cause excessive resource consumption
- **Strategic Solution**
  - New z/VM facility to obtain configuration information
  - Eliminate manual configuration and reduce resource use
  - Provide foundation for CPU Pooling
  - Linux access through library interface
  - SWG delivered ILMT 9.0.1 exploitation in August, 2014
- **Available June, 2014 – z/VM 6.3 APAR VM65419**

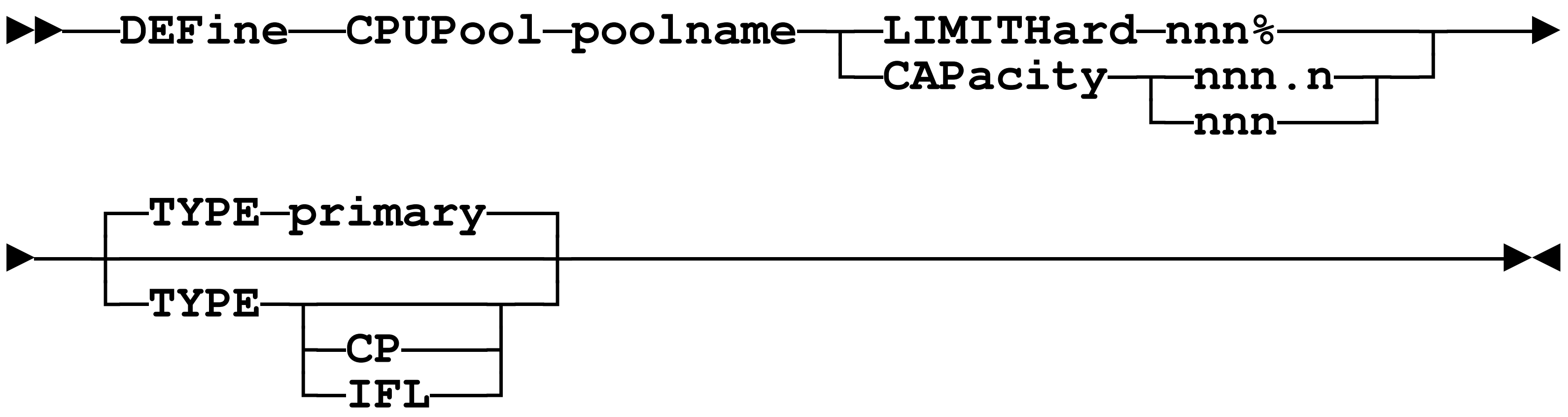
CPU Pooling Approach

- **Fine-grained CPU limiting for groups of virtual machines**
  - Allow client to define named CPU pools with associated capacity
    - Number of CPUs of particular type (CP, IFL)
    - Percentage of CPUs of particular type
  - Allow client to associate guests with CPU pools
  - Limit aggregate guest consumption to pool capacity
  - Include pool capacity information in ILMT interface
  - Provide new basis for software pricing (available August, 2014)
  - Over-commitment permitted
- **Available June, 2014 – z/VM 6.3 APAR VM65418**

Externals

- **DEFINE CPUPOOL**
- **SCHEDULE**
- **SET CPUPOOL**
- **QUERY CPUPOOL**
- **DELETE CPUPOOL**
- **Live Guest Relocation Implications**
- **New Monitor Records**
- **Existing Monitor Record Extensions**

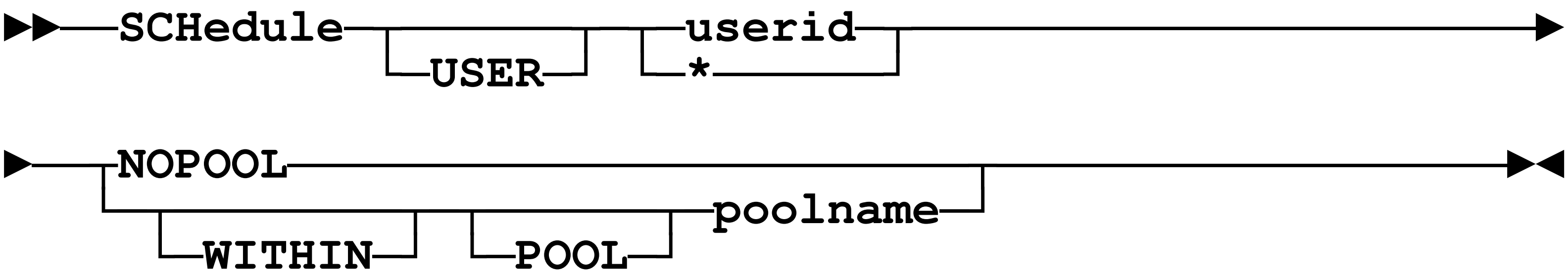
Define CPUPOOL



```
►►──DEFine──CPUPool─poolname─┬─LIMITHard─nnn%──────────┬──────►
                             └─CAPacity─┬─nnn.n─┬──────┘
                                        └─nnn───┘

   ┌─TYPE─primary──────┐
►──┼───────────────────┼──────────────────────────────────►◄
   └─TYPE──┬─CP───┬────┘
           └─IFL──┘
```

SCHEDULE



SCHedule

USER

userid

*

NOPOOL

WITHIN

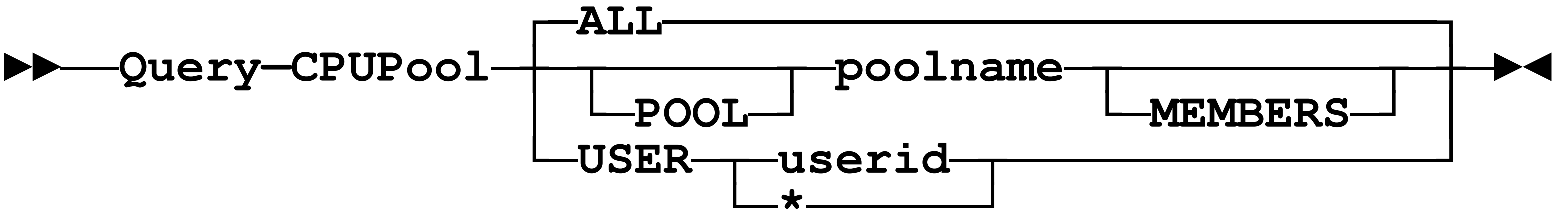
POOL

poolname

Set CPUPOOL

```
►►──Set──CPUPool──poolname──┬─LIMITHard─nnn%────────────┬──────────►◄
                            └─CAPacity──┬─nnn.n─┬───────┘
                                        └─nnn───┘
```

QUERY CPUPOOL



Query CPUPool ALL
POOL poolname MEMBERS
USER userid
*

Functions:

1. Display all pool definitions.
2. Display one pool definition and member names.
3. Display user’s pool name.

DELETE CPUPOOL

```
►►──Delete─CPUPool─poolname──────────────────────────────────►◄
```

Live Guest Relocation Implications

- **Guest in CPU pool requires identically named pool with same TYPE attribute on relocation target system**
  - Cannot be overridden by VMRELOCATE FORCE
  - If no limit required on target, remove guest from pool before relocation
  - If different pool required on target, create pool with same name on source and assign guest to it before relocation
  - Best practice is to use common pool names across cluster
- **Pool capacities independent and separately enforced on each member**

New Monitor Records

- **Domain 1 Record 28 – CPU Pool Configuration**
  - Sample configuration record
  - Shows pool definition informatuion
- **Domain 1 Record 29 – CPU Pool Definition**
  - Event record for DEFINE/SET/DELETE CPUPOOL
  - Shows pool definition information
- **Domain 4 Record 13 – CPU Pool Change**
  - Event record when user’s pool relationship changes
- **Domain 5 Record 19 – CPU Pool Utilization**
  - Sample record for each CPU Pool
  - Includes
    - Pool definition
    - CPU consumption
    - Interval timestamp

Existing Monitor Record Extensions

- **Pool name added to Scheduler domain event records 13 and 14 (Limit List Add/Drop)**
- **Pool name in Monitor domain sample record 15 (Logged On User)**
- **Pool name in User domain sample record 3 (User Activity)**

New z/VM Interfaces

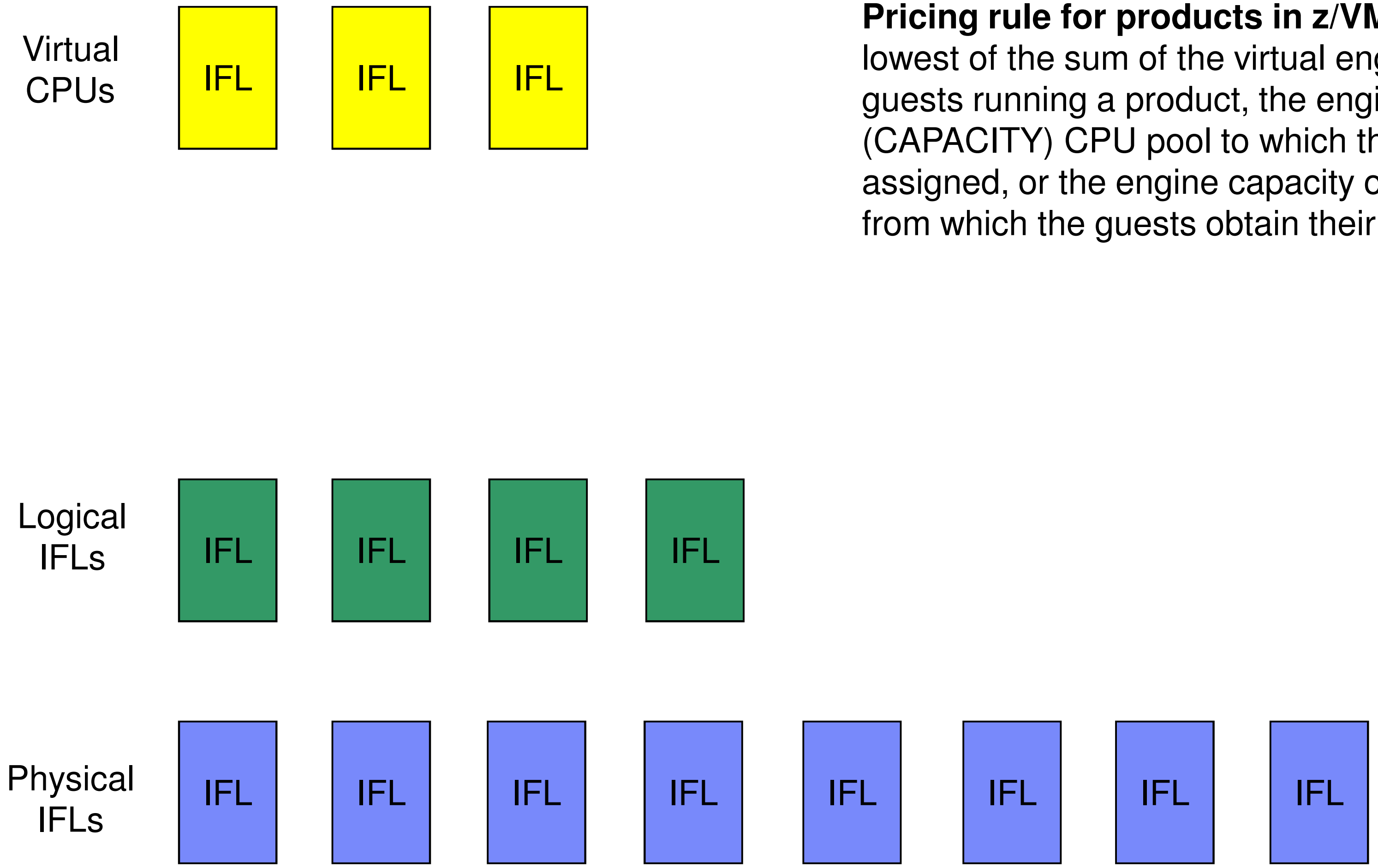
- **New problem state STHYI (STore HYpervisor Information) instruction**
  - RRE format
  - Opcode B256
  - R_1 contains function code in bits 48–63
  - R_1+1 is ignored
  - R_2 contains logical address of 4K output buffer
  - R_2+1 contains return code
- **Associated STFLE facility bit**
- **Supported by z/VM 6.3; tolerated by z/VM 6.2 (reports “function not supported”)**

New Linux Interface

- **Employs STHYI instruction**
- **License suitable for delivering binary-only products**
- **Exploited by ILMT**

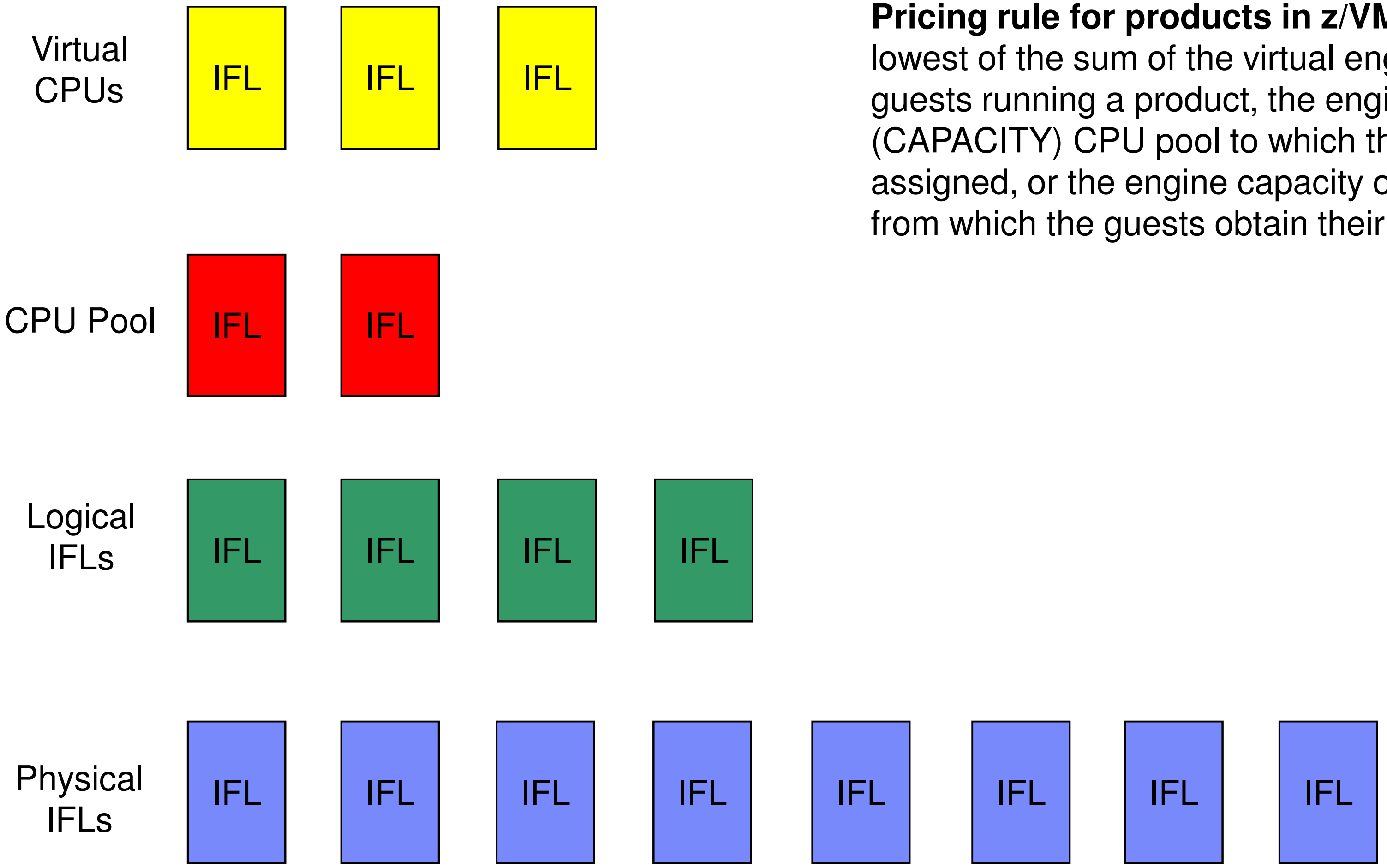
```
#include "query_capacity.h"
[...]
void *handle;
handle = qc_open_configuration();
levels = qc_get_container_levels(handle);
printf (“CPC type %s\n”,
          qc_get_string_attribute(handle, "type", 0));
```

Linux Guest Software Pricing With CPU Pooling



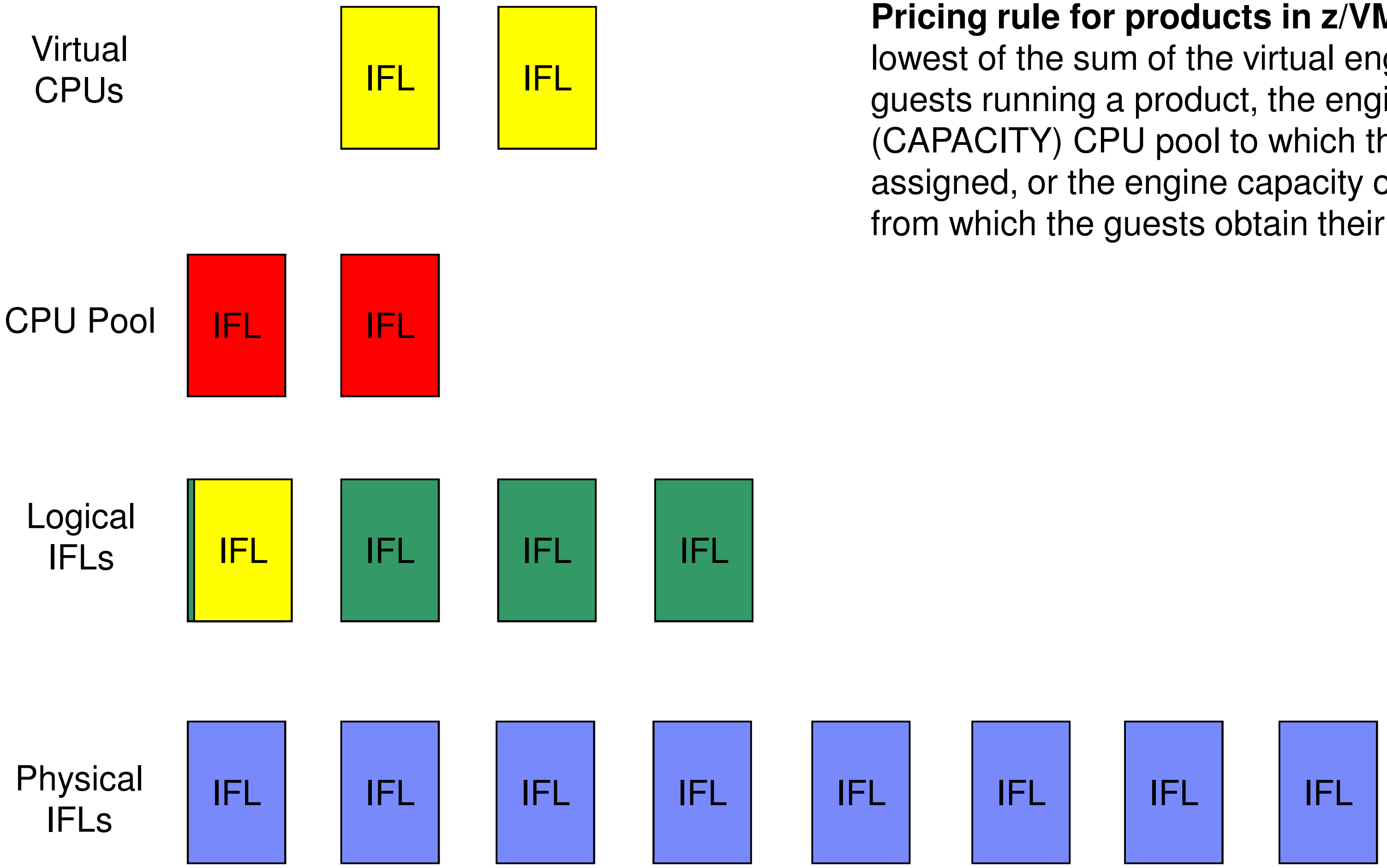
Pricing rule for products in z/VM guests: The lowest of the sum of the virtual engines available to guests running a product, the engine capacity of the (CAPACITY) CPU pool to which the guests are assigned, or the engine capacity of the z/VM LPAR from which the guests obtain their resources.

Linux Guest Software Pricing With CPU Pooling



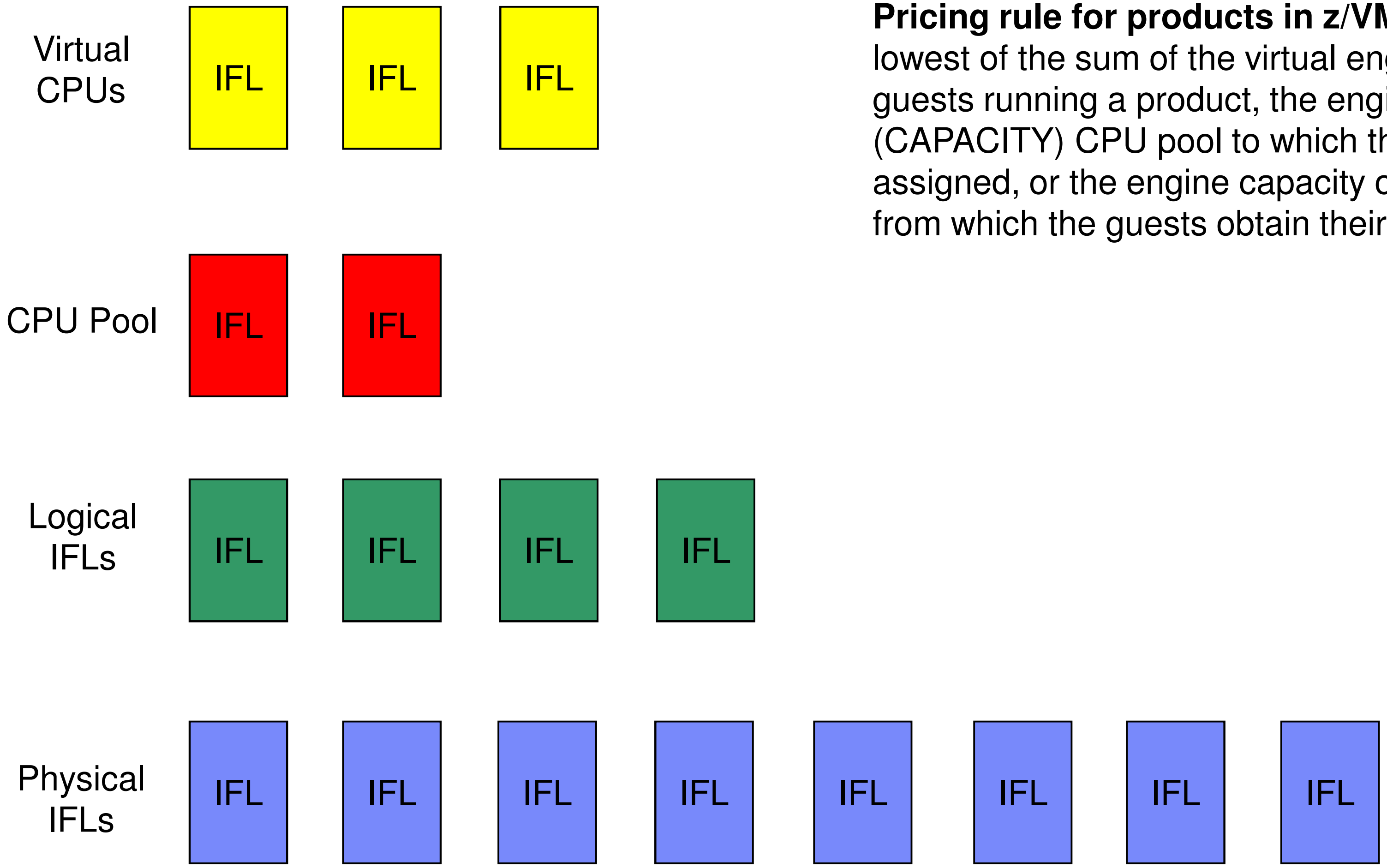
Pricing rule for products in z/VM guests: The lowest of the sum of the virtual engines available to guests running a product, the engine capacity of the (CAPACITY) CPU pool to which the guests are assigned, or the engine capacity of the z/VM LPAR from which the guests obtain their resources.

Linux Guest Software Pricing With CPU Pooling



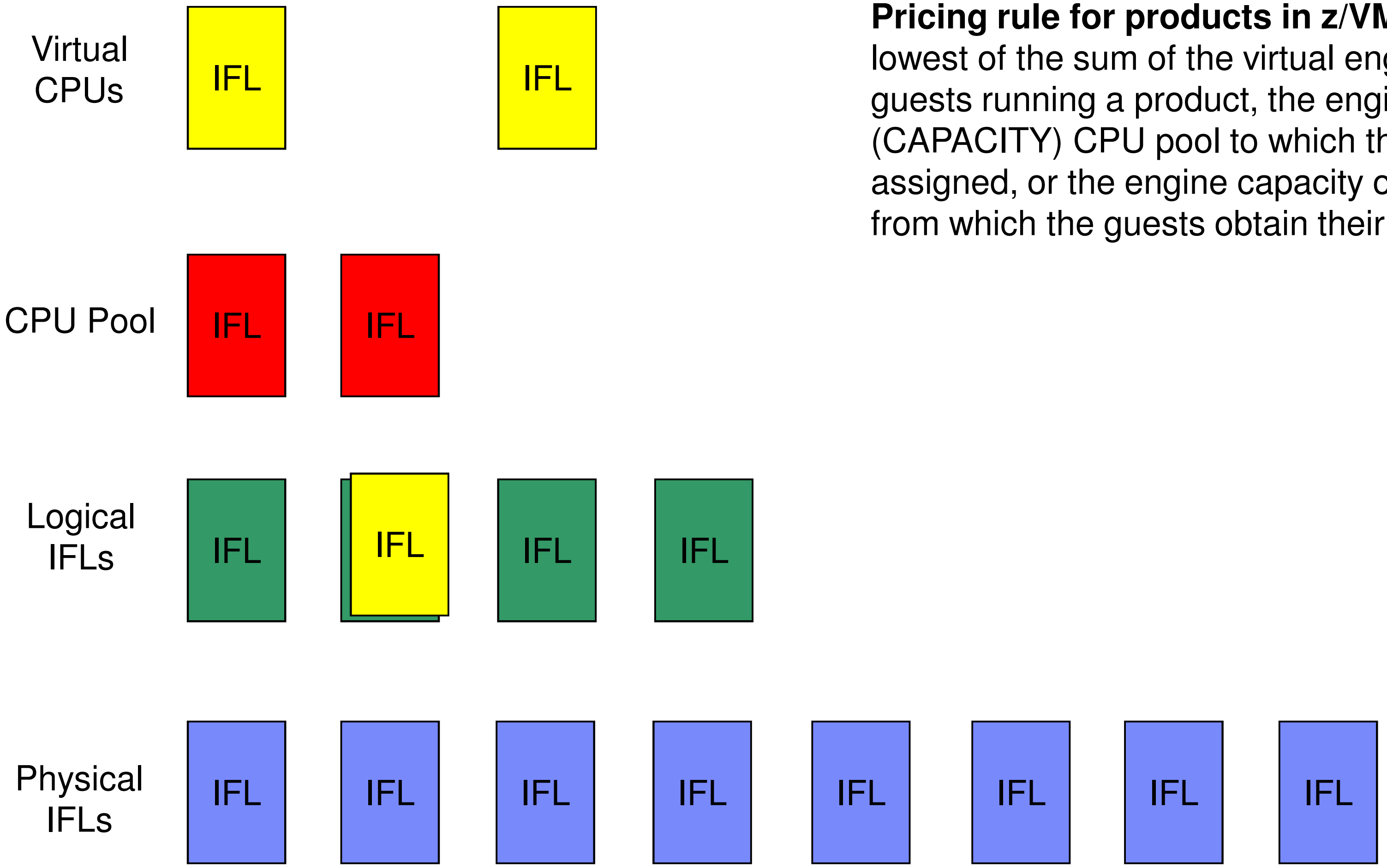
Pricing rule for products in z/VM guests: The lowest of the sum of the virtual engines available to guests running a product, the engine capacity of the (CAPACITY) CPU pool to which the guests are assigned, or the engine capacity of the z/VM LPAR from which the guests obtain their resources.

Linux Guest Software Pricing With CPU Pooling



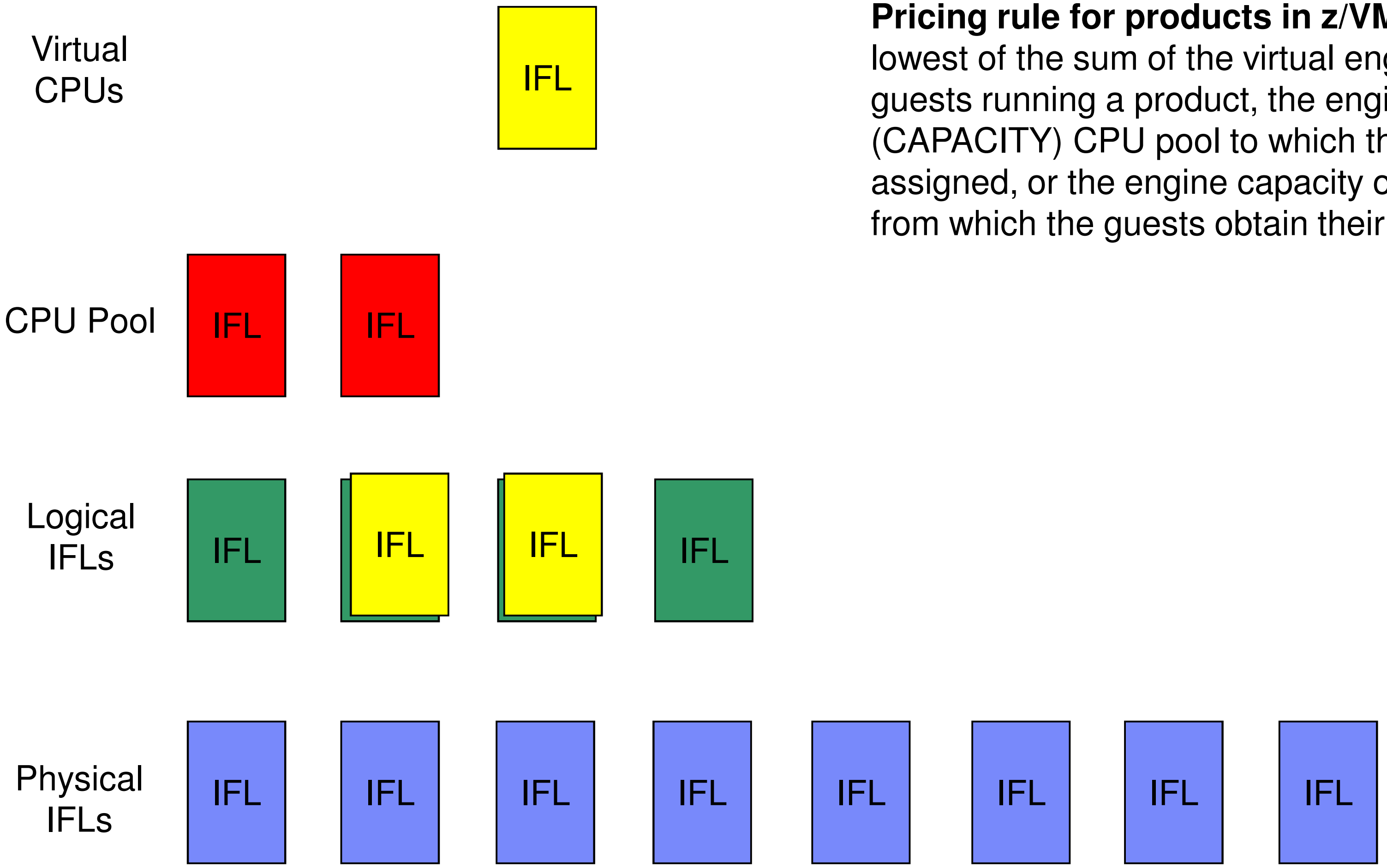
Pricing rule for products in z/VM guests: The lowest of the sum of the virtual engines available to guests running a product, the engine capacity of the (CAPACITY) CPU pool to which the guests are assigned, or the engine capacity of the z/VM LPAR from which the guests obtain their resources.

Linux Guest Software Pricing With CPU Pooling



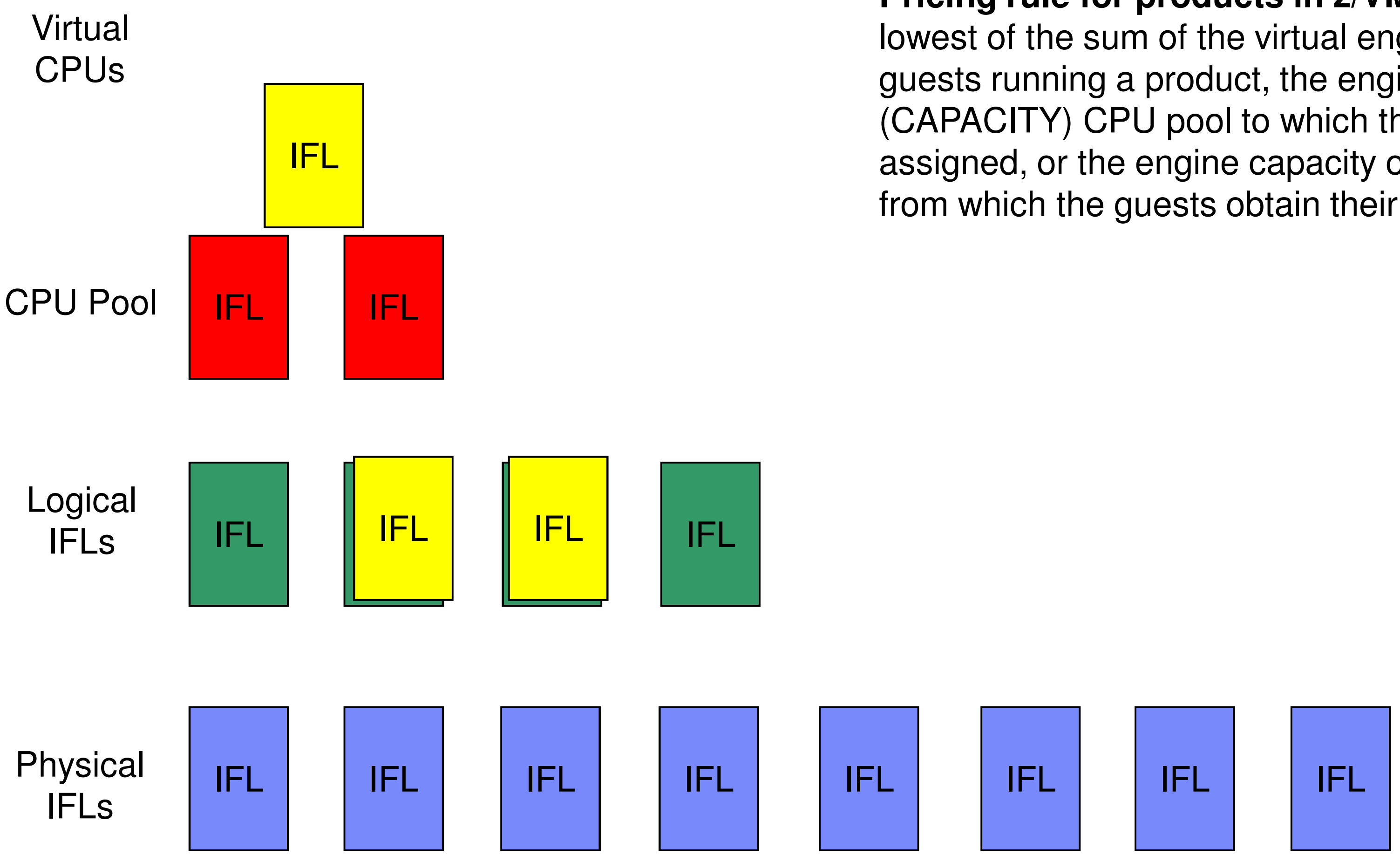
Pricing rule for products in z/VM guests: The lowest of the sum of the virtual engines available to guests running a product, the engine capacity of the (CAPACITY) CPU pool to which the guests are assigned, or the engine capacity of the z/VM LPAR from which the guests obtain their resources.

Linux Guest Software Pricing With CPU Pooling



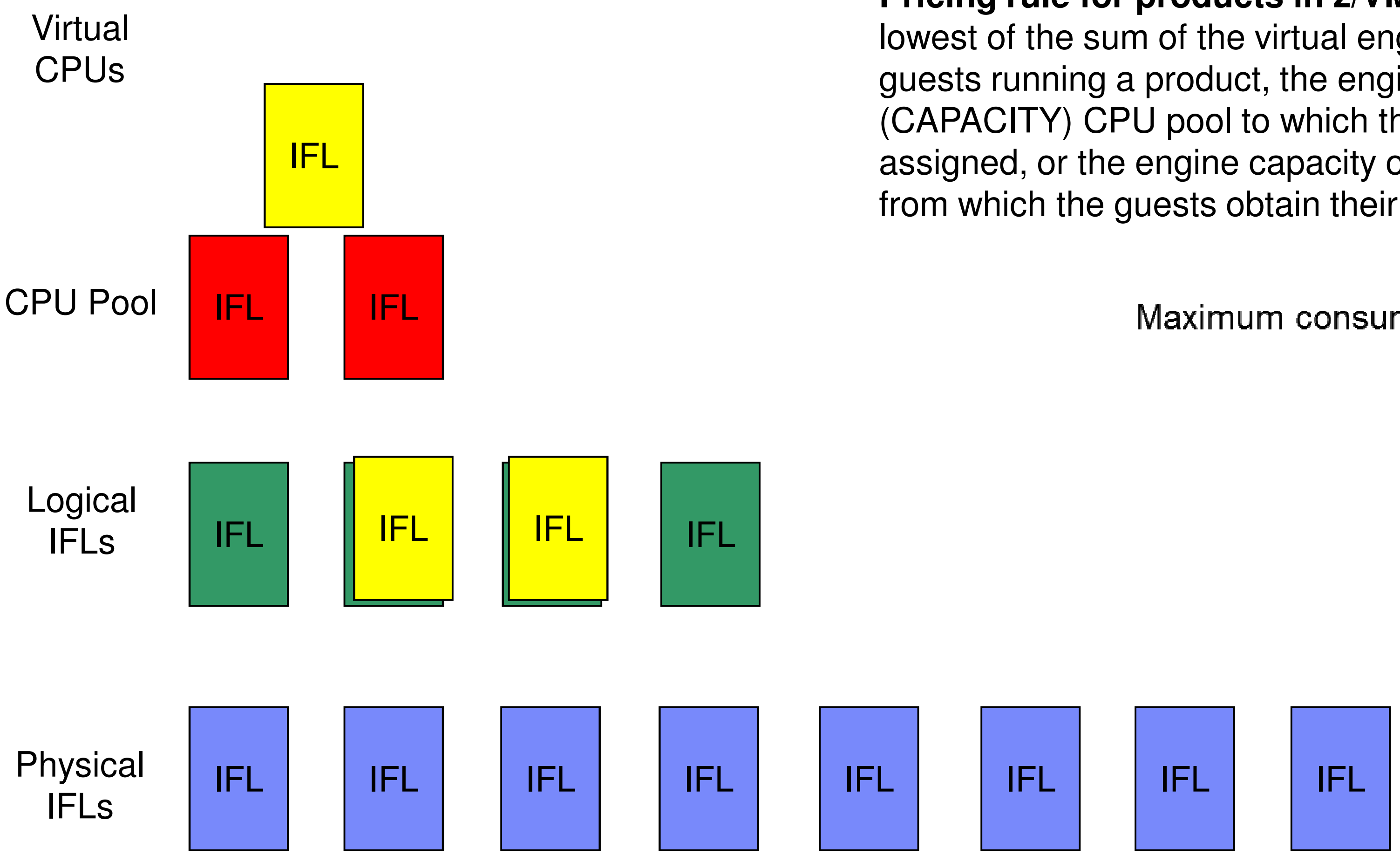
Pricing rule for products in z/VM guests: The lowest of the sum of the virtual engines available to guests running a product, the engine capacity of the (CAPACITY) CPU pool to which the guests are assigned, or the engine capacity of the z/VM LPAR from which the guests obtain their resources.

Linux Guest Software Pricing With CPU Pooling



Pricing rule for products in z/VM guests: The lowest of the sum of the virtual engines available to guests running a product, the engine capacity of the (CAPACITY) CPU pool to which the guests are assigned, or the engine capacity of the z/VM LPAR from which the guests obtain their resources.

Linux Guest Software Pricing With CPU Pooling



Pricing rule for products in z/VM guests: The lowest of the sum of the virtual engines available to guests running a product, the engine capacity of the (CAPACITY) CPU pool to which the guests are assigned, or the engine capacity of the z/VM LPAR from which the guests obtain their resources.

Maximum consumption: 2 IFLs

Use cases for CPU Pooling

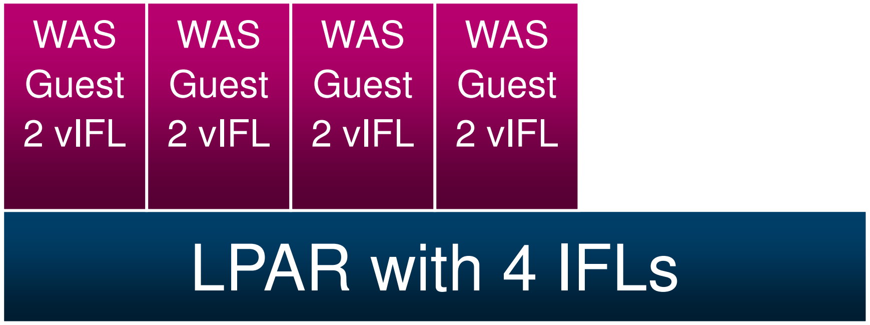
- Department budgeting
  - Assign each department’s guests to CPU pool with contracted capacity
- Grow workloads without affecting the budget
  - Add New Workload
  - Add Capacity
  - Combine LPARs
  - Handle fractional workload requirements
- Prevent resource over-consumption
  - Limit aggressive workloads

Enforce Resource Budget

- **Agree to provide specific amount of resource to group (e.g., department)**
- **Create CPU pool for group with agreed capacity**
- **Assign guests in group to pool**
- **Limits group resource consumption (and associated charges)**

Add New Workload Without CPU Pooling

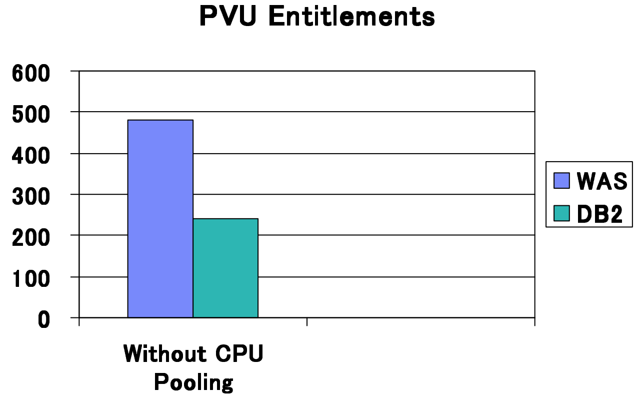
- **4 WAS production guests**
  - **Requires 4-engine WAS entitlement**

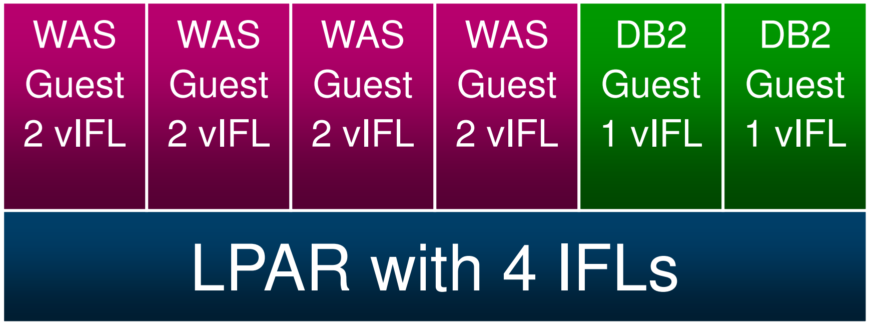


Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Add New Workload Without CPU Pooling

- 4 WAS production guests
  - Requires 4-engine WAS entitlement
- Add 2 DB2 production guests
  - Requires 2-engine DB2 entitlement



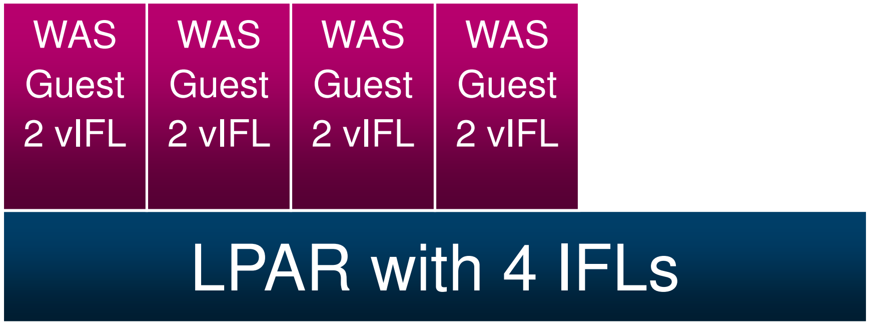

WAS Guest 2 vIFL | WAS Guest 2 vIFL | WAS Guest 2 vIFL | WAS Guest 2 vIFL | DB2 Guest 1 vIFL | DB2 Guest 1 vIFL

LPAR with 4 IFLs

Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Add New Workload With CPU Pooling

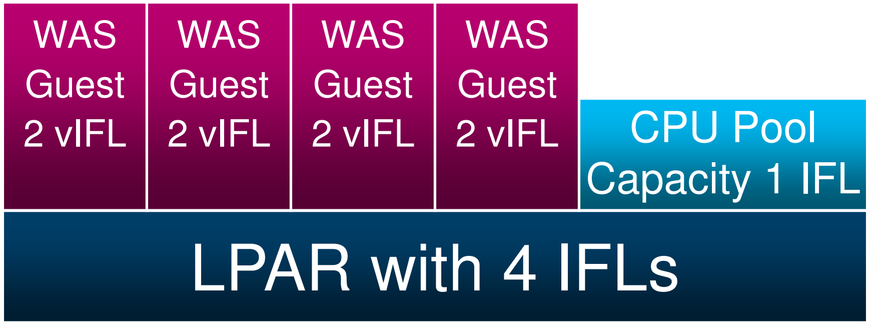
- **4 WAS production guests**
  - **Requires 4-engine WAS entitlement**



Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Add New Workload With CPU Pooling

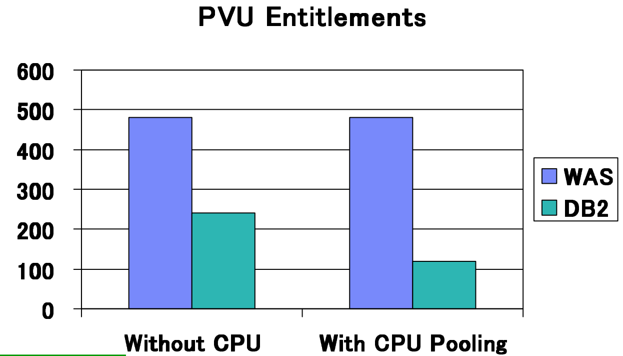
- **4 WAS production guests**
  - **Requires 4-engine WAS entitlement**
- **Create a 1-IFL pool**

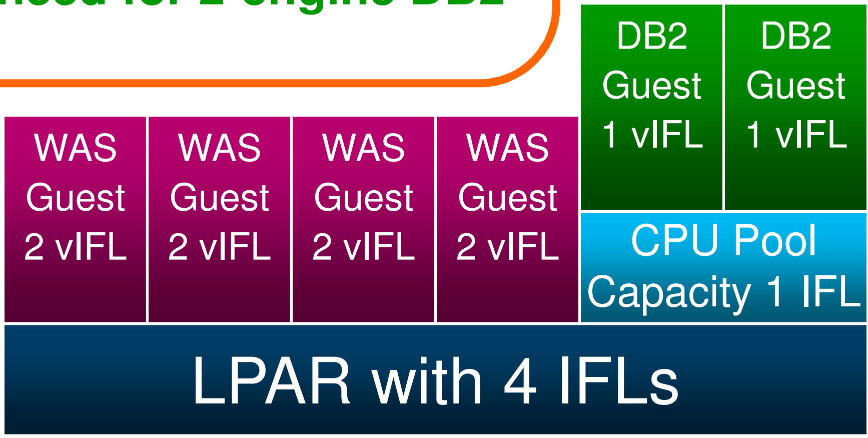


Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Add New Workload With CPU Pooling

- **4 WAS production guests**
  - **Requires 4-engine WAS entitlement**
- **Create a 1-IFL pool**
- **Put the 2 DB2 production guests in pool**
  - **Requires 1-engine DB2 entitlement (avoiding the need for 2-engine DB2 entitlement)**

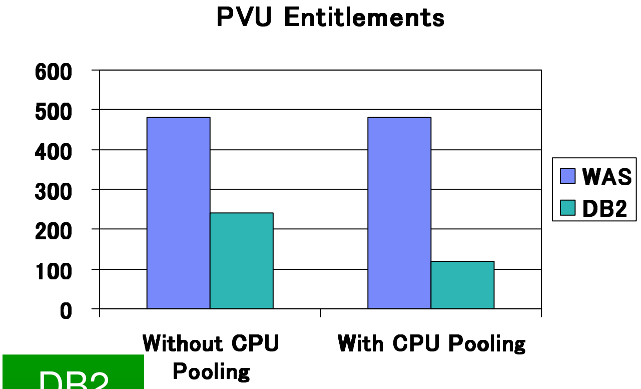


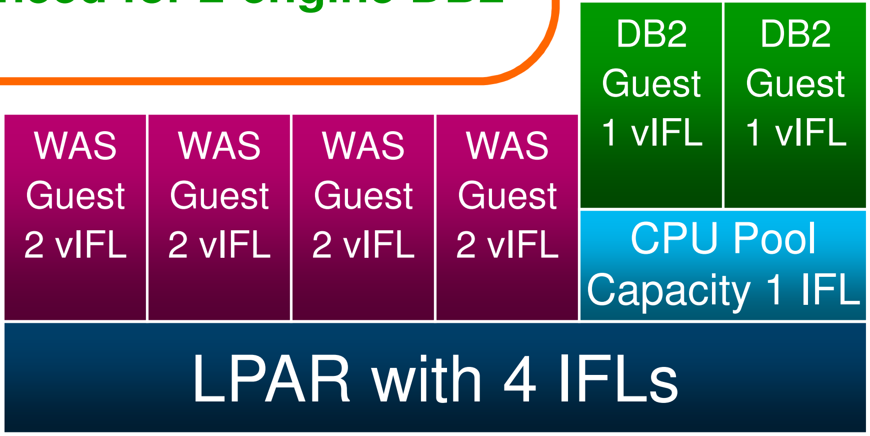


Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Add New Workload With CPU Pooling

- 4 WAS production guests
  - Requires 4-engine WAS entitlement
- Create a 1-IFL pool
- Put the 2 DB2 production guests in pool
  - Requires 1-engine DB2 entitlement (avoiding the need for 2-engine DB2 entitlement)




DB2 Guest 1 vIFL

DB2 Guest 1 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

CPU Pool Capacity 1 IFL

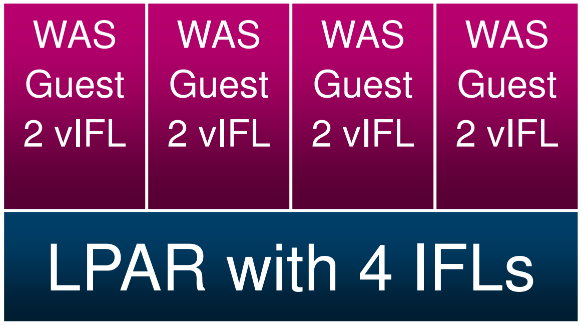
LPAR with 4 IFLs

- Allows new workloads to be added cost effectively
- Encourages additional workload consolidation after initial success

Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Add Capacity Without CPU Pooling

- **4 WAS production guests**
  - **Requires 4-engine WAS entitlement**



WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

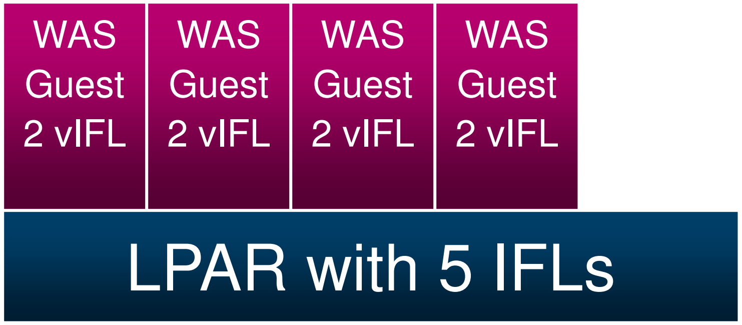
WAS Guest 2 vIFL

LPAR with 4 IFLs

Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Add Capacity Without CPU Pooling

- **4 WAS production guests**
  - **Requires 4-engine WAS entitlement**
- **Add another IFL to the LPAR**
  - **Requires increase to 5-engine WAS entitlement**



WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

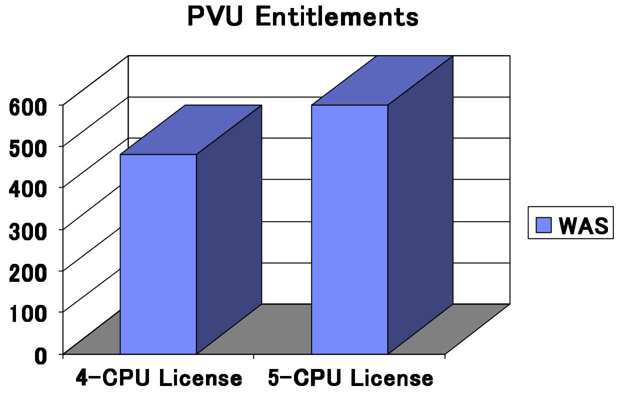
WAS Guest 2 vIFL

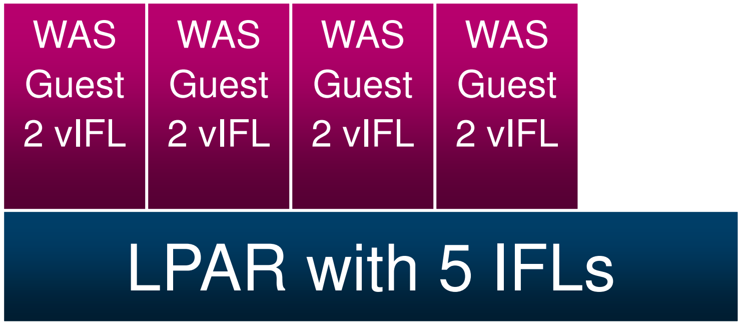
LPAR with 5 IFLs

Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Add Capacity Without CPU Pooling

- **4 WAS production guests**
  - **Requires 4-engine WAS entitlement**
- **Add another IFL to the LPAR**
  - **Requires increase to 5-engine WAS entitlement**





Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Add Capacity With CPU Pooling

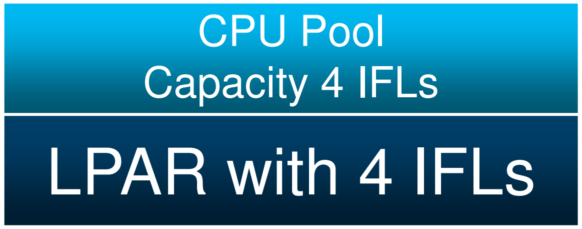
- **LPAR with 4 IFLs**

LPAR with 4 IFLs

Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Add Capacity With CPU Pooling

- **LPAR with 4 IFLs**
- **Set up CPU Pooling for 4 IFLs**



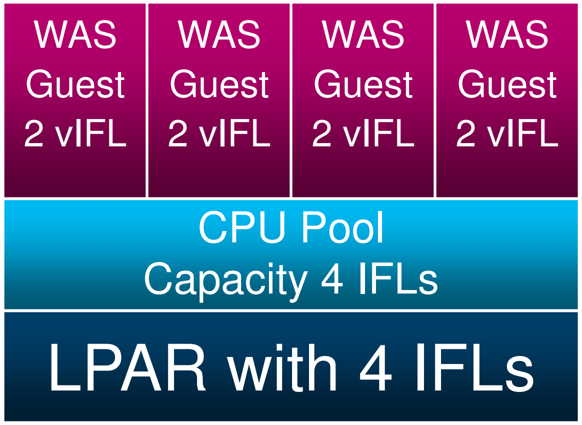
CPU Pool
Capacity 4 IFLs

LPAR with 4 IFLs

Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Add Capacity With CPU Pooling

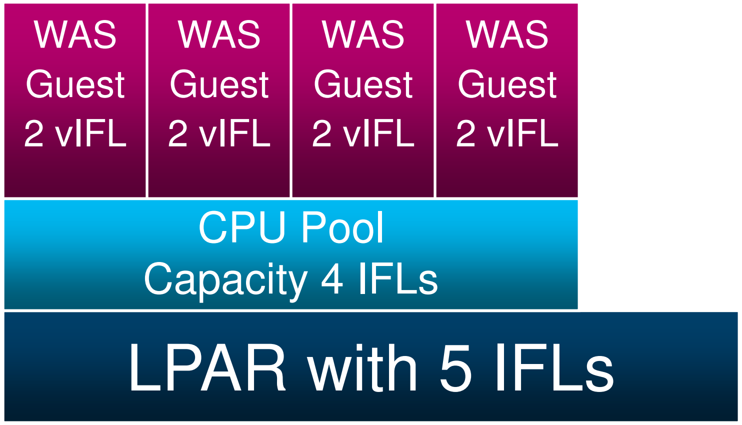
- **LPAR with 4 IFLs**
- **Set up CPU Pooling for 4 IFLs**
  - **4 WAS production guests require 4-engine WAS entitlement**



Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Add Capacity With CPU Pooling

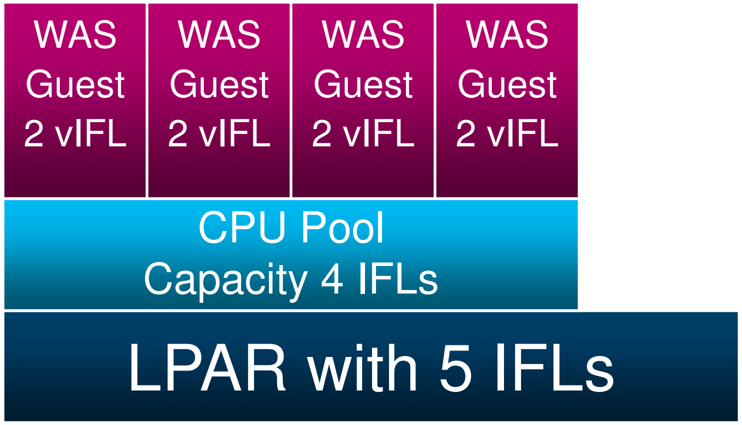
- **LPAR with 4 IFLs**
- **Set up CPU Pooling for 4 IFLs**
  - **4 WAS production guests require 4-engine WAS entitlement**
- **Add another IFL to the LPAR**

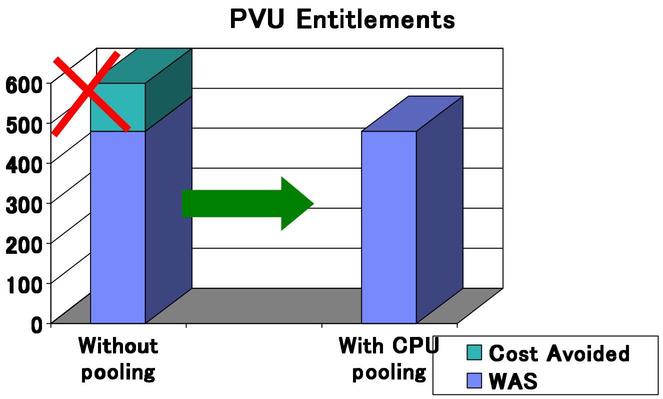


Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Add Capacity With CPU Pooling

- **LPAR with 4 IFLs**
- **Set up CPU Pooling for 4 IFLs**
  - **4 WAS production guests require 4-engine WAS entitlement**
- **Add another IFL to the LPAR**
- **Avoids an incremental WAS entitlement license – allows capacity to be added without increasing software license charges**

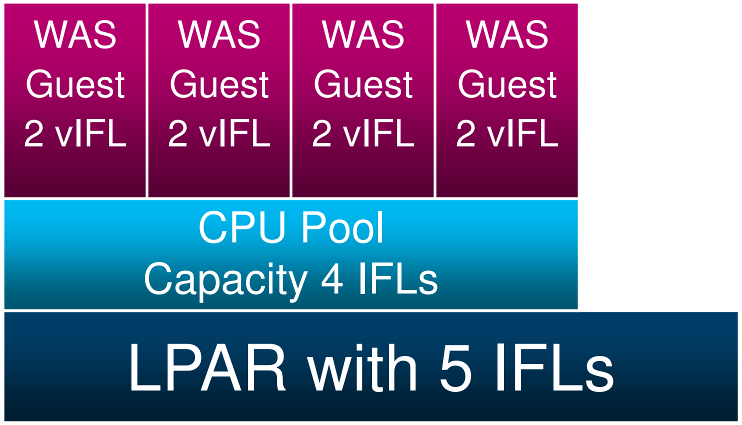




Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Add Capacity With CPU Pooling

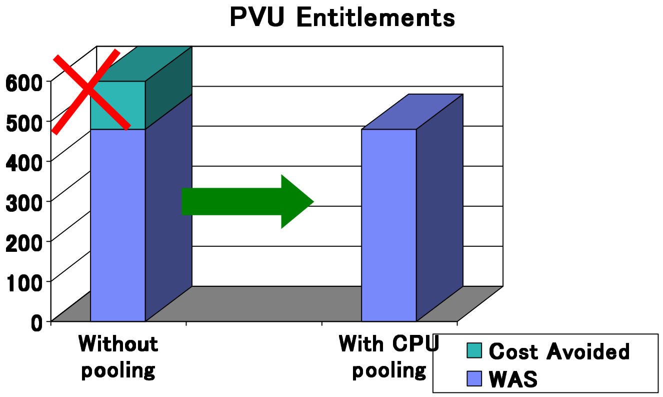
- **LPAR with 4 IFLs**
- **Set up CPU Pooling for 4 IFLs**
  - **4 WAS production guests require 4-engine WAS entitlement**
- **Add another IFL to the LPAR**
- **Avoids an incremental WAS entitlement license – allows capacity to be added without increasing software license charges**
- **Encourages adding capacity for other workloads (e.g., open source applications)**



WAS Guest 2 vIFL | WAS Guest 2 vIFL | WAS Guest 2 vIFL | WAS Guest 2 vIFL

CPU Pool Capacity 4 IFLs

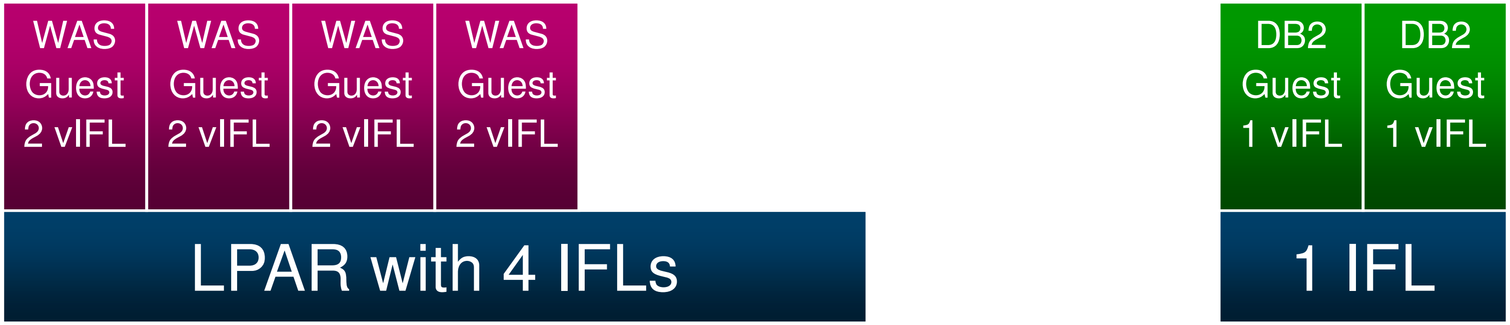
LPAR with 5 IFLs



Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Combine LPARs Without CPU Pooling

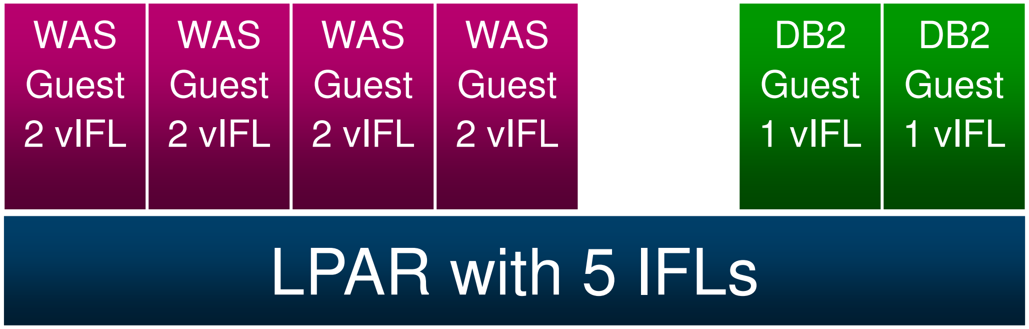
- **LPAR with 4 IFLs and 4 WAS production guests**
  - **Requires 4-engine WAS entitlement**
- **LPAR with 1 IFL and 2 DB2 production guests**
  - **Requires 1-engine DB2 entitlement**



Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Combine LPARs Without CPU Pooling

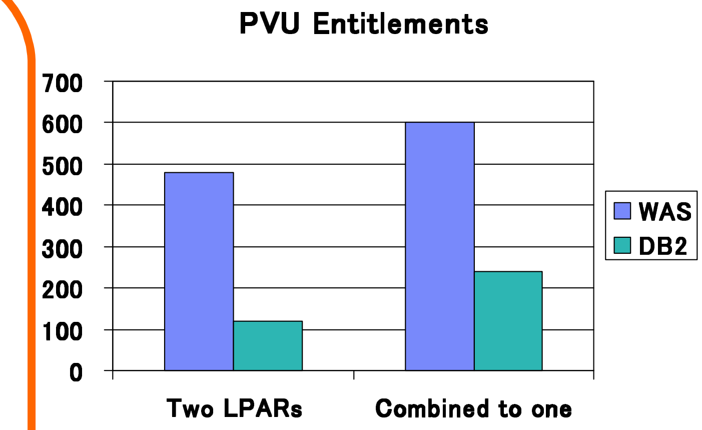
- LPAR with 4 IFLs and 4 WAS production guests
  - Requires 4-engine WAS entitlement
- LPAR with 1 IFL and 2 DB2 production guests
  - Requires 1-engine DB2 entitlement
- LPARs merge to one LPAR with 5 IFLs

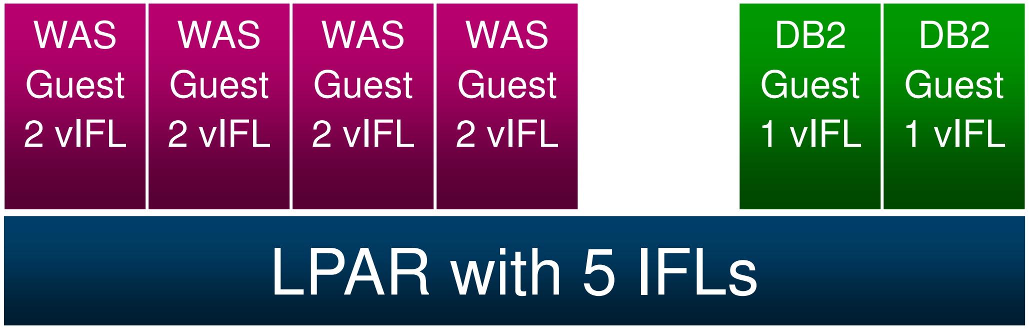


Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Combine LPARs Without CPU Pooling

- **LPAR with 4 IFLs and 4 WAS production guests**
  - **Requires 4-engine WAS entitlement**
- **LPAR with 1 IFL and 2 DB2 production guests**
  - **Requires 1-engine DB2 entitlement**
- **LPARs merge to one LPAR with 5 IFLs**
  - **Requires increase to 5-engine WAS entitlement**
  - **Requires increase to 2-engine DB2 entitlement**





WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

DB2 Guest 1 vIFL

DB2 Guest 1 vIFL

LPAR with 5 IFLs

Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Combine LPARs With CPU Pooling

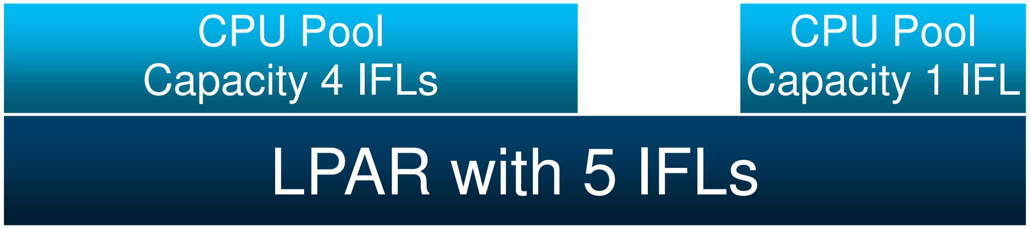
- **LPAR with 5 IFLs**

LPAR with 5 IFLs

Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Combine LPARs With CPU Pooling

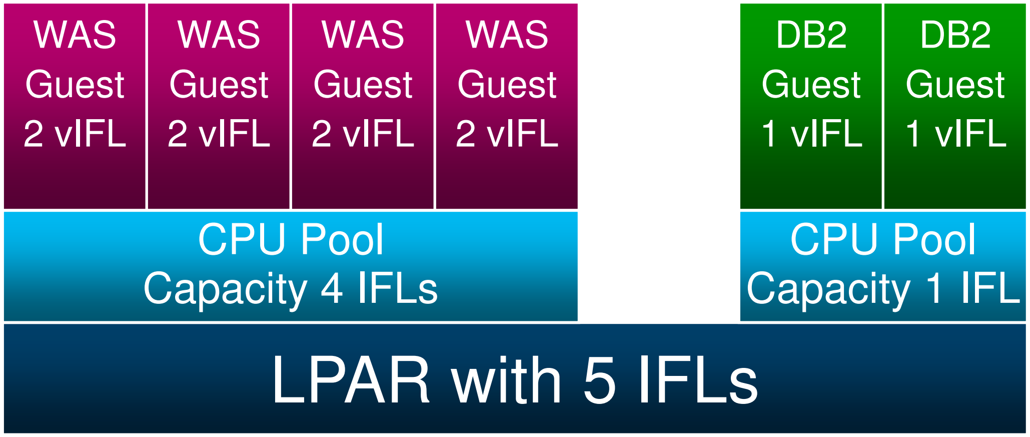
- **LPAR with 5 IFLs**
- **Create 2 Pools – one with 4 IFLs and one with 1 IFL**



Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Combine LPARs With CPU Pooling

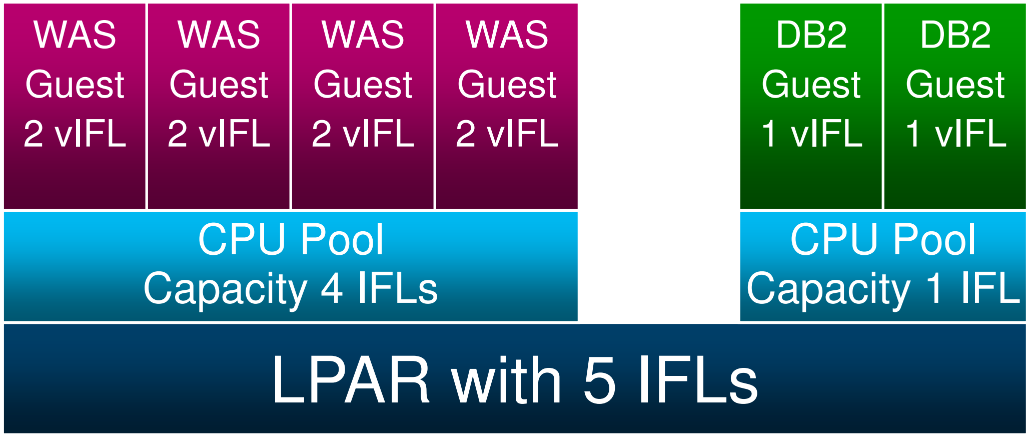
- **LPAR with 5 IFLs**
- **Create 2 Pools – one with 4 IFLs and one with 1 IFL**
- **Place the four WAS guests in the 4-IFL pool and the two DB2 guests in the 1-IFL pool**

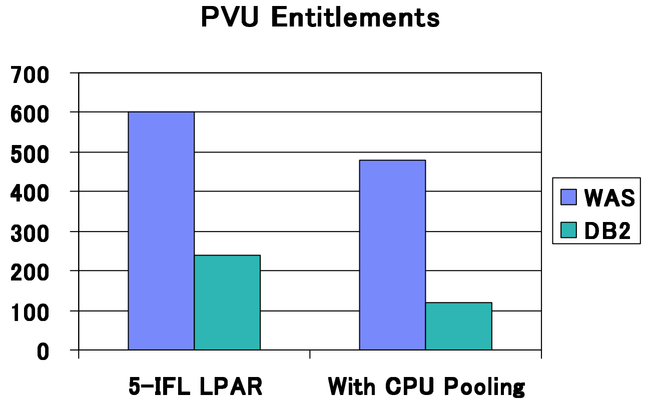


Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Combine LPARs With CPU Pooling

- **LPAR with 5 IFLs**
- **Create 2 Pools – one with 4 IFLs and one with 1 IFL**
- **Place the four WAS guests in the 4-IFL pool and the two DB2 guests in the 1-IFL pool**
  - **Requires 4-engine WAS entitlement**
  - **Requires 1-engine DB2 entitlement**

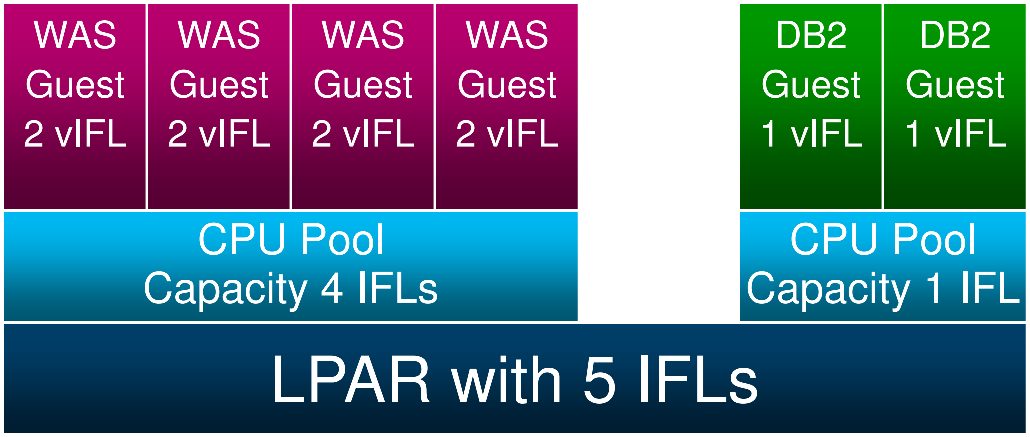




Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Combine LPARs With CPU Pooling

- **LPAR with 5 IFLs**
- **Create 2 Pools – one with 4 IFLs and one with 1 IFL**
- **Place the four WAS guests in the 4-IFL pool and the two DB2 guests in the 1-IFL pool**
  - **Requires 4-engine WAS entitlement**
  - **Requires 1-engine DB2 entitlement**



WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

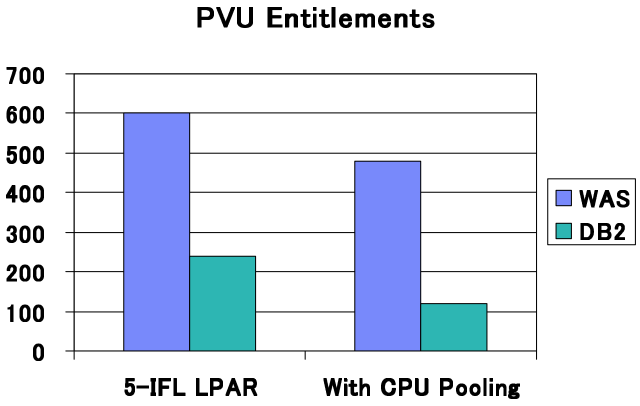
DB2 Guest 1 vIFL

DB2 Guest 1 vIFL

CPU Pool Capacity 4 IFLs

CPU Pool Capacity 1 IFL

LPAR with 5 IFLs

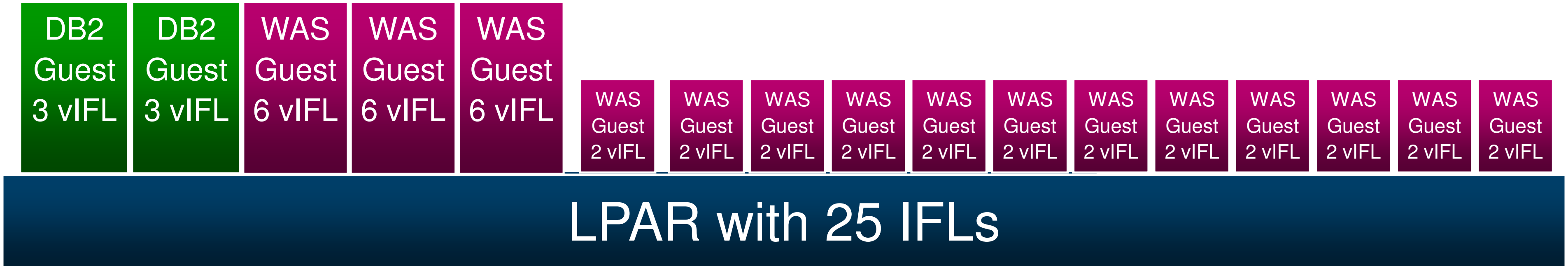


- **Avoids increase in software license requirements (and costs)**
- **Reduces z/VM system management and maintenance workload**
- **Consolidates resources (memory, paging, network) for greater efficiency**

Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Large system with guests needing fractional IFL capacity

- **LPAR with 25 IFLs**
- **2 DB2 production guests**
  - **Requires 6-engine DB2 entitlement**
- **3 WAS production guests and 12 small WAS test guests**
  - **Requires 25-engine WAS entitlement**



DB2 Guest 3 vIFL

DB2 Guest 3 vIFL

WAS Guest 6 vIFL

WAS Guest 6 vIFL

WAS Guest 6 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

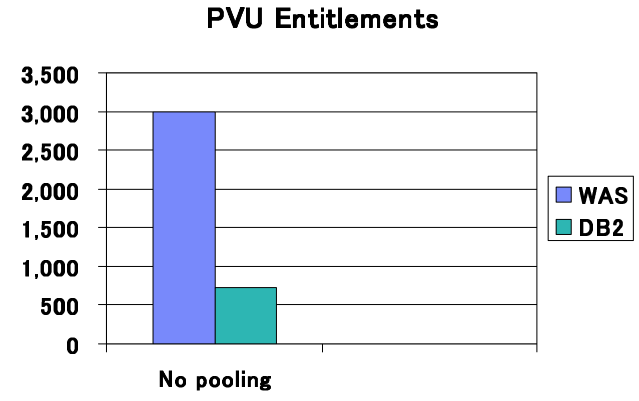
WAS Guest 2 vIFL

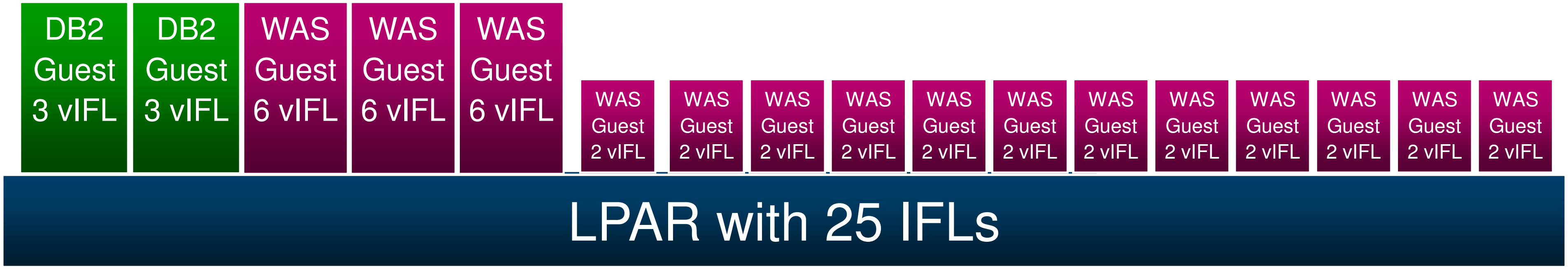
LPAR with 25 IFLs

Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Large system with guests needing fractional IFL capacity

- **LPAR with 25 IFLs**
- **2 DB2 production guests**
  - **Requires 6-engine DB2 entitlement**
- **3 WAS production guests and 12 small WAS test guests**
  - **Requires 25-engine WAS entitlement**




DB2 Guest 3 vIFL

DB2 Guest 3 vIFL

WAS Guest 6 vIFL

WAS Guest 6 vIFL

WAS Guest 6 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

WAS Guest 2 vIFL

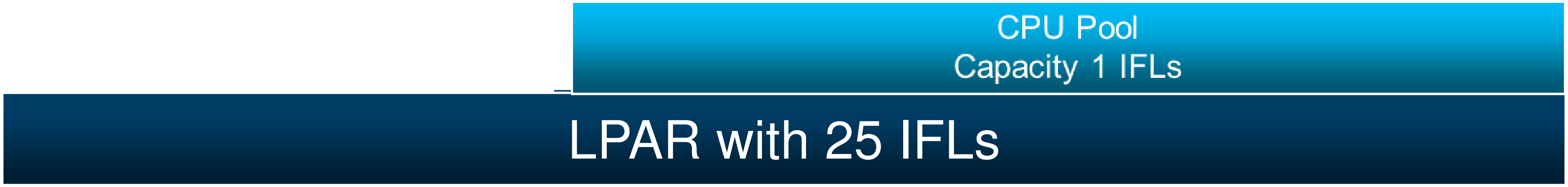
WAS Guest 2 vIFL

LPAR with 25 IFLs

Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Assign fractional capacity virtual machines to small CPU pool

- **LPAR with 25 IFLs**
- **Set up a 1-IFL pool**



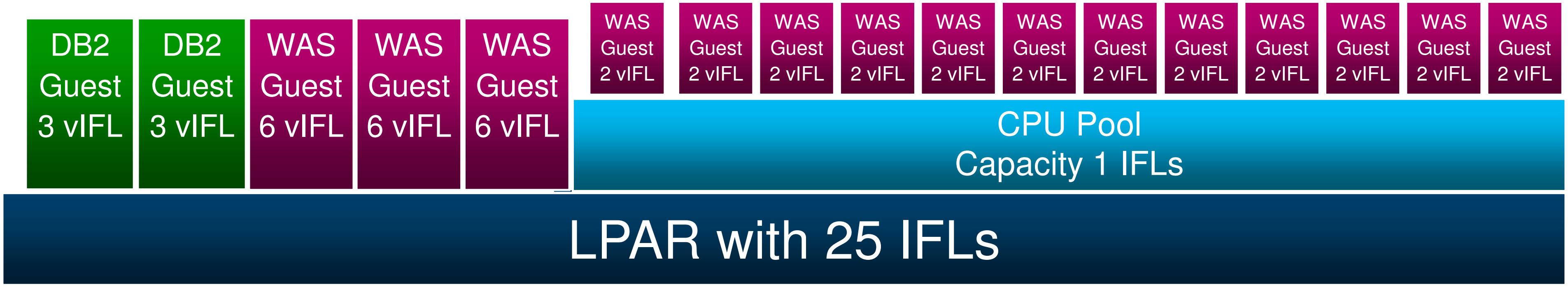
CPU Pool
Capacity 1 IFLs

LPAR with 25 IFLs

Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Assign fractional capacity virtual machines to small CPU pool

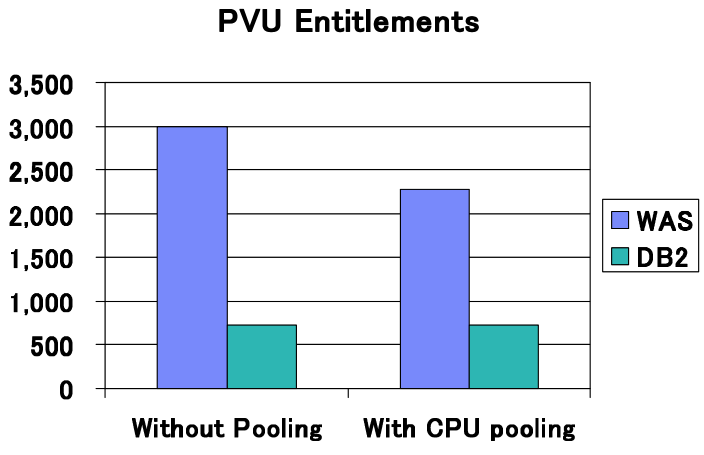
- **LPAR with 25 IFLs**
- **Set up a 1-IFL pool**
- **2 DB2 production guests**
- **3 WAS production guests and 12 small WAS test guests in IFL pool**

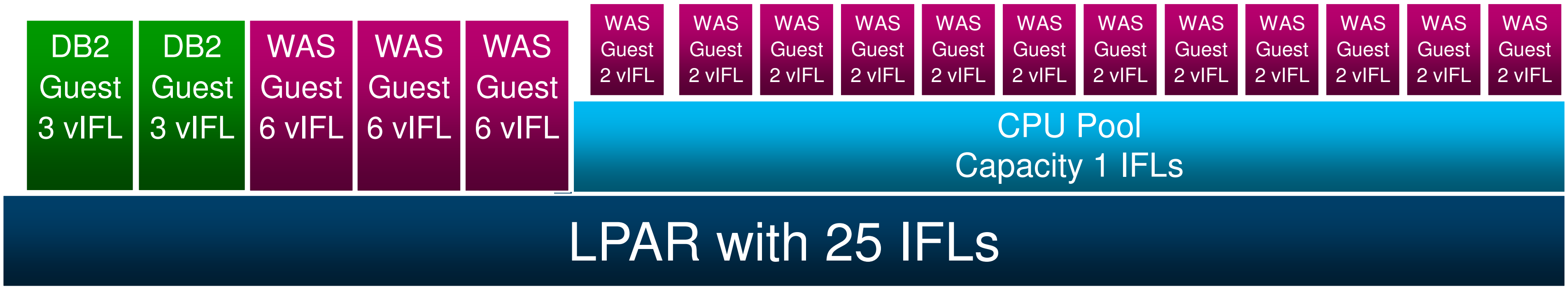


Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Assign fractional capacity virtual machines to small CPU pool

- **LPAR with 25 IFLs**
- **Set up a 1-IFL pool**
- **2 DB2 production guests**
  - **Requires 6-engine DB2 entitlement**
- **3 WAS production guests and 12 small WAS test guests in IFL pool**
  - **Requires 19-engine WAS entitlement**




DB2 Guest 3 vIFL

DB2 Guest 3 vIFL

WAS Guest 6 vIFL

WAS Guest 6 vIFL

WAS Guest 6 vIFL

WAS Guest 2 vIFL | WAS Guest 2 vIFL | WAS Guest 2 vIFL | WAS Guest 2 vIFL | WAS Guest 2 vIFL | WAS Guest 2 vIFL | WAS Guest 2 vIFL | WAS Guest 2 vIFL | WAS Guest 2 vIFL | WAS Guest 2 vIFL | WAS Guest 2 vIFL | WAS Guest 2 vIFL

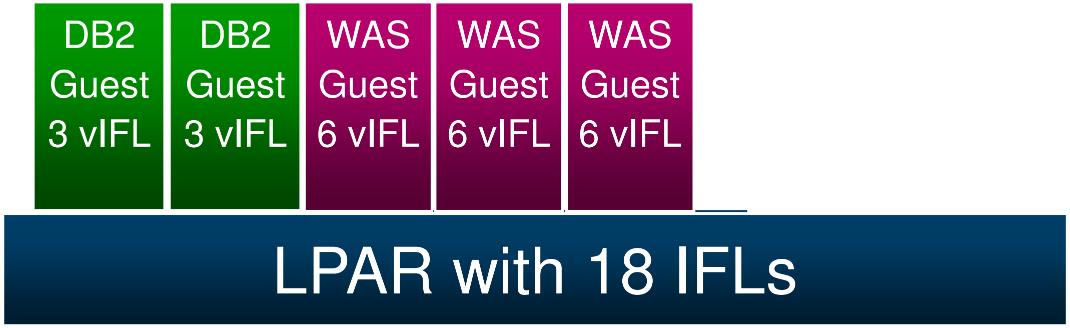
CPU Pool
Capacity 1 IFLs

LPAR with 25 IFLs

Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Contain workloads that take too many resources

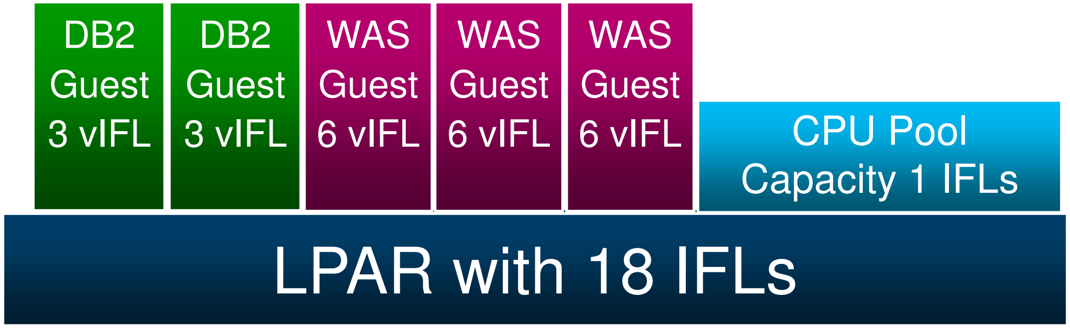
- **LPAR with 18 IFLs**
- **2 DB2 production guests and 3 WAS production guests are sharing the 18 IFLs**
- **Month-end processing or nightly backup uses any available capacity – could take from production guests**



Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Contain workloads that take too many resources

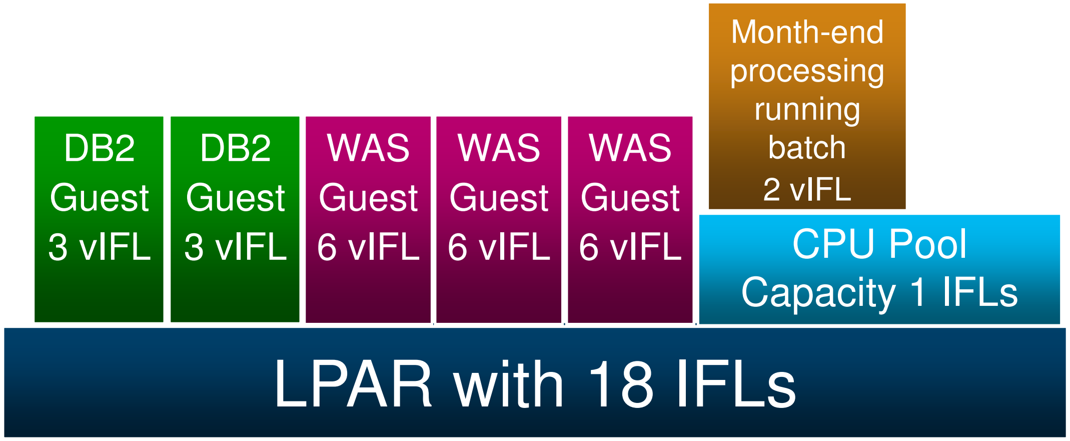
- **LPAR with 18 IFLs**
- **2 DB2 production guests and 3 WAS production guests are sharing the 18 IFLs**
- **Month-end processing or nightly backup uses any available capacity – could take from production guests**
- **Set up a 1-IFL CPU pool for running these tasks**



Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Contain workloads that take too many resources

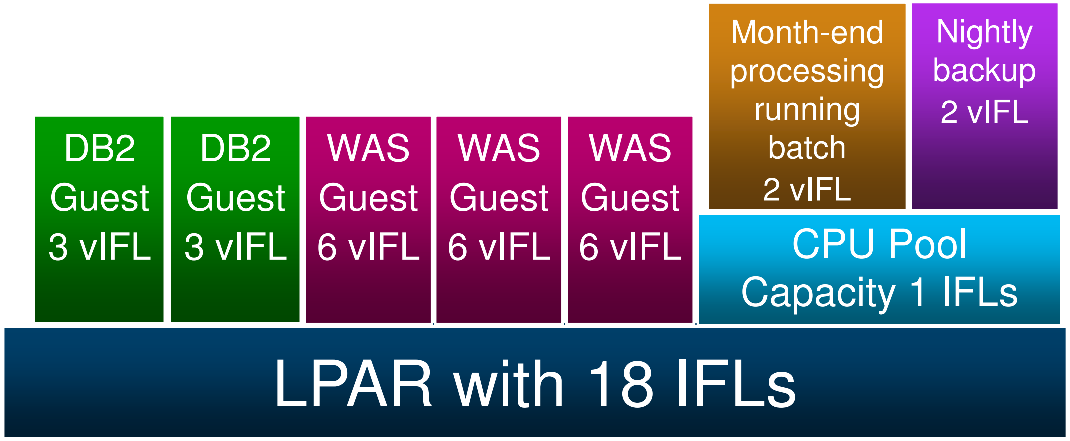
- **LPAR with 18 IFLs**
- **2 DB2 production guests and 3 WAS production guests are sharing the 18 IFLs**
- **Month-end processing or nightly backup uses any available capacity – could take from production guests**
- **Set up a 1-IFL CPU pool for running these tasks**



Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Contain workloads that take too many resources

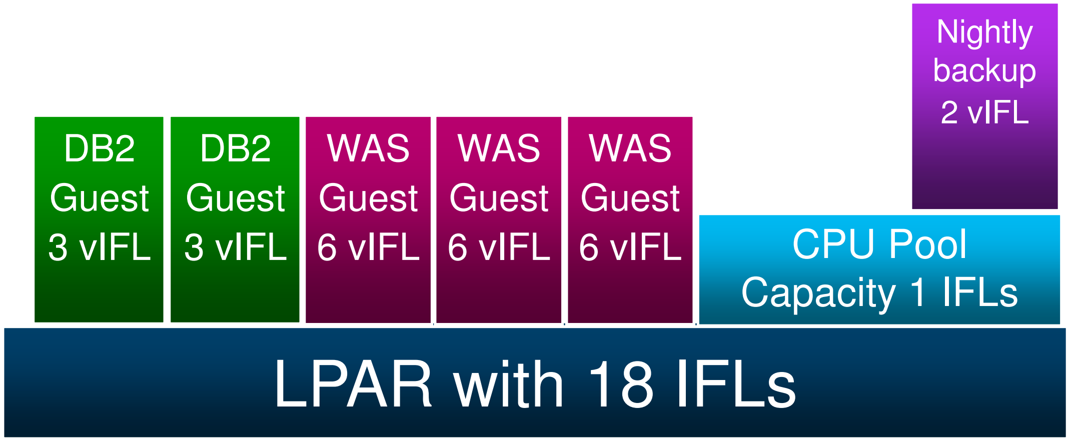
- **LPAR with 18 IFLs**
- **2 DB2 production guests and 3 WAS production guests are sharing the 18 IFLs**
- **Month-end processing or nightly backup uses any available capacity – could take from production guests**
- **Set up a 1-IFL CPU pool for running these tasks**



Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Contain workloads that take too many resources

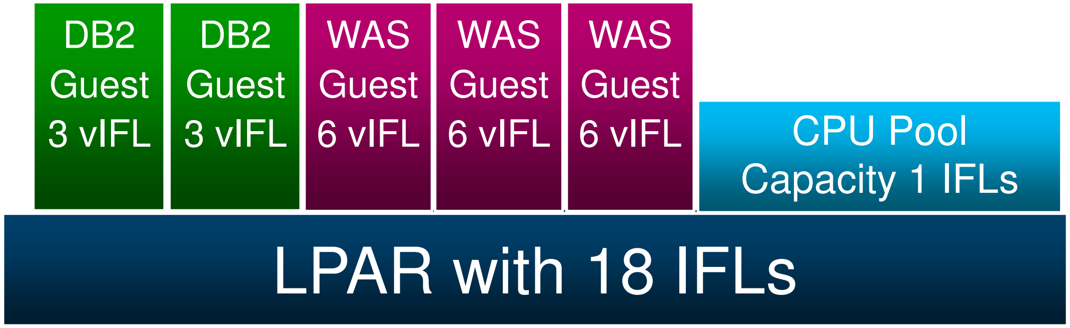
- **LPAR with 18 IFLs**
- **2 DB2 production guests and 3 WAS production guests are sharing the 18 IFLs**
- **Month-end processing or nightly backup uses any available capacity – could take from production guests**
- **Set up a 1-IFL CPU pool for running these tasks**



Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Contain workloads that take too many resources

- **LPAR with 18 IFLs**
- **2 DB2 production guests and 3 WAS production guests are sharing the 18 IFLs**
- **Month-end processing or nightly backup uses any available capacity – could take from production guests**
- **Set up a 1-IFL CPU pool for running these tasks**



Note: All PVU Entitlement examples based on zEC12 (120 PVU per IFL) – will look proportionally the same on zBC12 (100 PVU per IFL)

Questions?