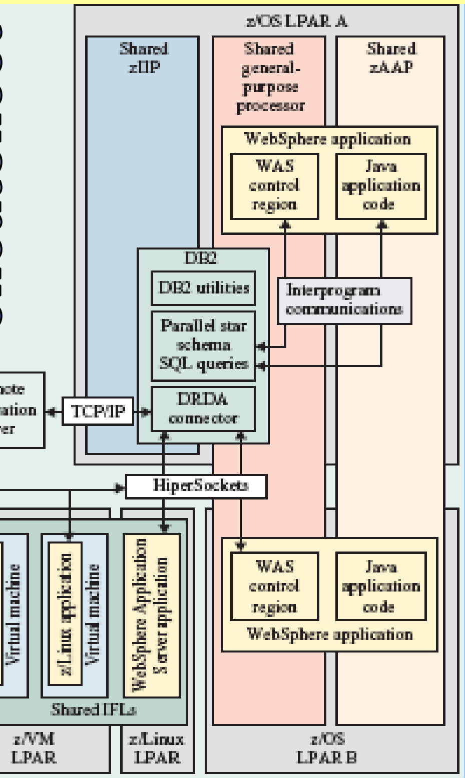
Controlling execution of zIIPs and zAAPs

CheatSheet



•The hypervisor creates and maintains separate physical processor pools for each processor type—CPs, zAAPs, zIIPs, Integrated Facility for Linux processors (IFLs), and Integrated Coupling Facility processors (ICFs) and uses the proprietary z/Architecture start interpretive execution (**SIE**) virtualization technology provided by System z to define, initiate, and control the execution of zAAP and zIIP logical processors on their corresponding physical CPs.
- This architecture provides a logicall processor state description that is loaded into the physical zAAP and zIIP instruction processing controls when the SIE instruction is executed by the hypervisor..
- In turn, the SIE instruction activates the physical processor on behalf of its associated LPAR.
- Once activated, zAAP and zIIP logical processors typically remain active on behalf of their associated partitions until certain OS conditions are encountered i.e. OS WAIT STATE or an IO Interrupt..

* PR/SM LPAR Hypervisor controls the execution of LPARs.
- This includes the assignment of processors, memory and I/O
- Each partition must have at least one CP and may have one or more zAAPs and zIIPS up to a max of 54 total processors.
- The number of zAAPs or zIIPs individually can not exceed the number of CPs for a given system configuration. (i.e. 5 installed CPs = max of 5 zIIPs and/or max of 5)

* zAAPs and zIIPs may be defined as either dedicated or shared just as with CPs assigned to a z/OS LPAR.
- Regardless of processor type, dedicated and shared processors may not be concurrently assigned to the same LPAR.
- Dedicated zIIPs, and CPs are used exclusively by the z/OS operating system where they are assigned to an LPAR.
- More typical, as with shraed CPs, shared zAAPs and zIIPs are available for use by *all* LPARs for which shared zAAPs and zIIPs are defined.
- Each zIIP and zAAP logicall processor comprises processor hardware, millicode, and LPAR firmware controls which collectively represt the physical processor.
- Just as with CPs, each zAAP and zIIP logical processor contains a complete set of physical processor controls and associated operating states necessary to execute the logical processor on the hypervisor- selected physical processor.

EX: if a zAAP is configured to two LPARs, each partition has a separate zAAP logical processor configured to it. In turn, the z/OS control program provides independent zAAP and zIIP logical processor controls in order to dispatch the appropriate programming tasks and SRB programs to their associated zAAP and zIIP logical processors.

* To control the percentage of zAAP and zIIP processing capacity for each LPAR that shares zAAPs and zIIPs, the LPAR-activation-profile provides individual zAAP and zIIP processing weights, like those provided for CPs; that is, CPs, zAAPs, and zIIPs each have separate initial ,minimum, and maximum processing weights assigned to them for each sharing partition.
- The hypervisor maintains separate physical CP, zAAP, and zIIP processor pools and uses these weight specifications to control the percentage of physical processor capacity from each processor pool for each sharing partition.
- However, unlike CPs, zAAPs and zIIPs are not subject to the z/OS WLM soft-capping function.
- While the z/OS WLM monitors and provides usage statistics for both zAAPs and zIIPs, it does not dynamically adjust zAAP and zIIP shared processor weights, as is possible for shared CPs.

• **SYS1.PARMLIB(IEAOPT*xx*)** controls the zAAP behavior
- IFACROSSOVER & IFAHONORPRIORITY(*)

<u>NOTE:</u> *There is a change under zOS REL.8 intended to allow more zAAP eligible work to run on zAAP processors while still remaining responsive to zAAP demand.*
- IFAHONORPRIORITY is *now* independent from IFACROSSOVER

* A zAAP is a new resource type that is WLM'd is aware of
- Contributes using and delay samples
- Contributes service times
- zAAP utilization is reported by RMF (SPE OA05731)

* Up to z/OS 1.7 zAAPs are managed by WLM as extension of CPs
- Java work executing on zAAPs inherits the dispatch priority from its execution on regular CPs
- Execution is accounted for in execution velocity and goal achievement (PI)

* Beginning with z/OS 1.8 work on zAAP is managed independently
- Not included in routing decisions
- zAAP service not included in defined capacity computations and resource group management
- zAAPs are not varied by IRD Vary CPU Management
- WLM/SRM support of zIIPs is relatively equivalent to the zAAP monitoring support with the zAAP processing enhancements (available via distinct FMIDs)
- zIIP work managed as an extension of CP work
- zIIP work will flow over to general purpose CPs

<u>NOTE:</u> : No zIIP external controls like zAAP (no IEAOPT*xx* settings))
- Not included in routing decisions
- zIIP service not included in defined capacity computations and resource group management
- zIIPs are not varied by IRD Vary CPU Management.

* Assigning zAAP-eligible programs to their physical zAAP processors is accomplished by a four-phase process: **Phase 1** is the process by which the Java program is made eligible for execution on zAAP processors. **Phase 2** is the process by which the z/OS dispatcher, operating on a CP, suspends execution of the zAAP-eligible programs. **Phase 3** is the process by which the System z9 PR/SM selects and dispatches the zAAP logical processor to a corresponding zAAP physical processor. **Phase 4** is the process by which the z/OS dispatcher, operating on a zAAP, selects and dispatches the zAAP-eligible programs on their respective zAAP logical processors operating on physical zAAP processors.

<u>NOTE:</u> *The JVM that controls and executes Java programs uses an internal secure "switch" programming interface to request the z/OS control program to make it eligible for execution on a zAAP. This z/OS switching service is available only to the appropriate IBM JVMs. Attempts by other programs to use this interface will not result in a switching action by z/OS.*

* zIIPs execute those z/OS programs that are structured to operate unde control of z/OS-preemptable enclave service request blocks (SRBs).

<u>Note:</u> An enclave is a z/OS construct that allows a unit of work or tranx, such as a distributed relational database architecture (DRDA) request for DB2 to be assigned a goal by the z/OS WLM on the basis of rules.. The execution threads, such as SRBs, that perform the transaction are assigned priority and are activel managed by WLM to achieve the assigned goal.

* The process of assigning and executing zIIP-eligible SRB programs is similar to the four-phase process used for zAAP-eligible programs, as described in the preceding subsection. However, unlike zAAPs, which execute only Java programs, zIIPs can be exploited to execute any programming function that is structured to operate under control of enclave SRB, that is, zIIPs are not restricted to specific programming languages, but rather to a specific type of z/OS-dispatchable unit.

(*) Certain zIIP circumstances apply

JK zCPO/Ref: SJ