Hewlett Packard Enterprise

NonStop Technical Boot Camp 2023
TBC23-TB69 What is the NonStop Crown Jewel?
A closer look at the NonStop Kernel (NSK)

Bert van Es – Senior Instructor – NonStop Academy

September 2023

continuous
NonStop IT Solutions

Forward-looking statements

This is a rolling (up to three year) Roadmap and is subject to change without notice

This document contains forward looking statements regarding future operations, product development, product capabilities and availability dates. This information is subject to substantial uncertainties and is subject to change at any time without prior notification. Statements contained in this document concerning these matters only reflect Hewlett Packard Enterprise's predictions and / or expectations as of the date of this document and actual results and future plans of Hewlett Packard Enterprise may differ significantly as a result of, among other things, changes in product strategy resulting from technological, internal corporate, market and other changes. This is not a commitment to deliver any material, code or functionality and should not be relied upon in making purchasing decisions.

Agenda

Overview of L-series NonStop Operating Systems Architecture – Message based

How is the x86 Based processor used? Memory Management – Little and Big Endian

InfiniBand and Cluster I/O Module subsystems

Debugging and Run-Time architecture – TNS and TNS/X processes - Process control

Guardian and Open System Services file systems

Part 1

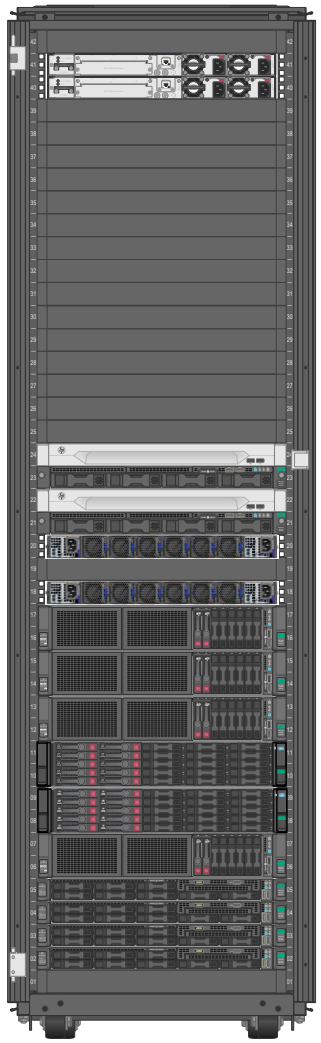
Overview of L-series NonStop Operating Systems Architecture

- Message based

Tandem Computers Design Goals: 1974

- Ability of the node to survive any single hardware or software failure
- Ability of a properly-coded application to survive any single failure
- Assured data integrity
  - When in doubt, fail fast
- Node and application scalability from 2 to 16 processors
- Application object code compatibility across releases
- Online repair of hardware resources
  - Including online reintegration of replacement hardware

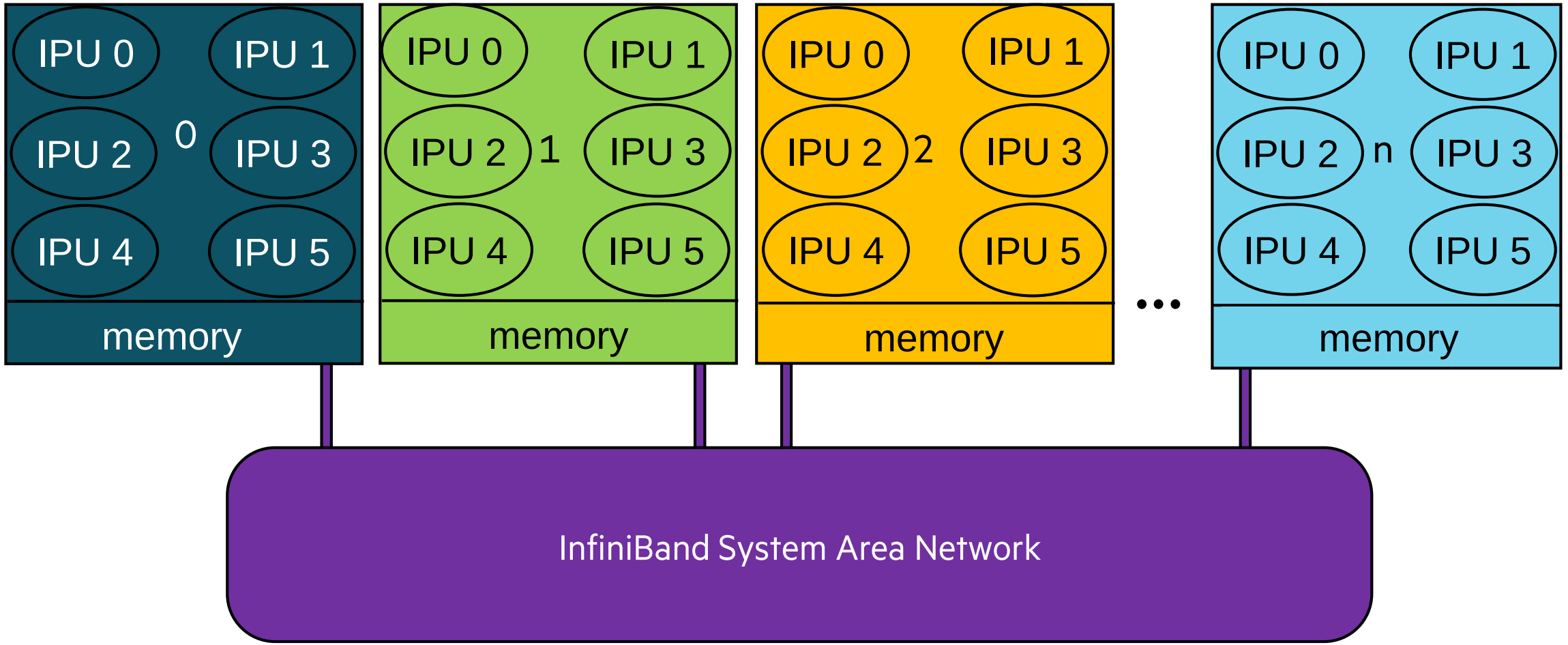
General Characteristics of NonStop Servers



A 4 CPU NS8

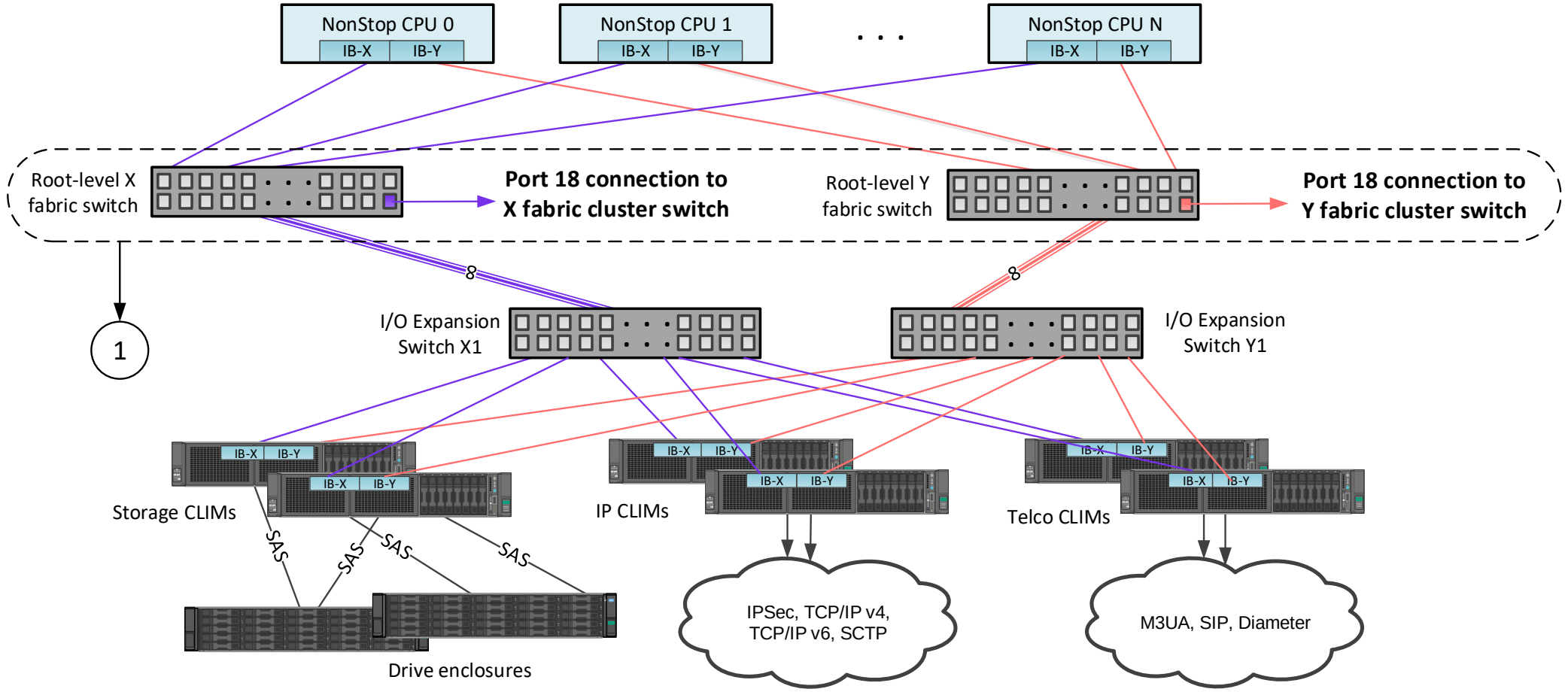
- Multiple independent logical processors.
- Hardware fault tolerance.
- Fault-tolerant operating system.
- High-performance SQL database.
- Hardware and software architectures that support:
  - Availability.
  - Data integrity
  - Performance and expandability/scalability
  - Open interface

HPE Integrity NonStop X architecture



NS8 Medium System Interconnect

InfiniBand layout



1

- NS7: 34-port FDR IB switches in c7000 enclosure
- NS8: rack-mount 40-port HDR IB switches

So why is the HPE NonStop OS or NSK the Crown Jewel?

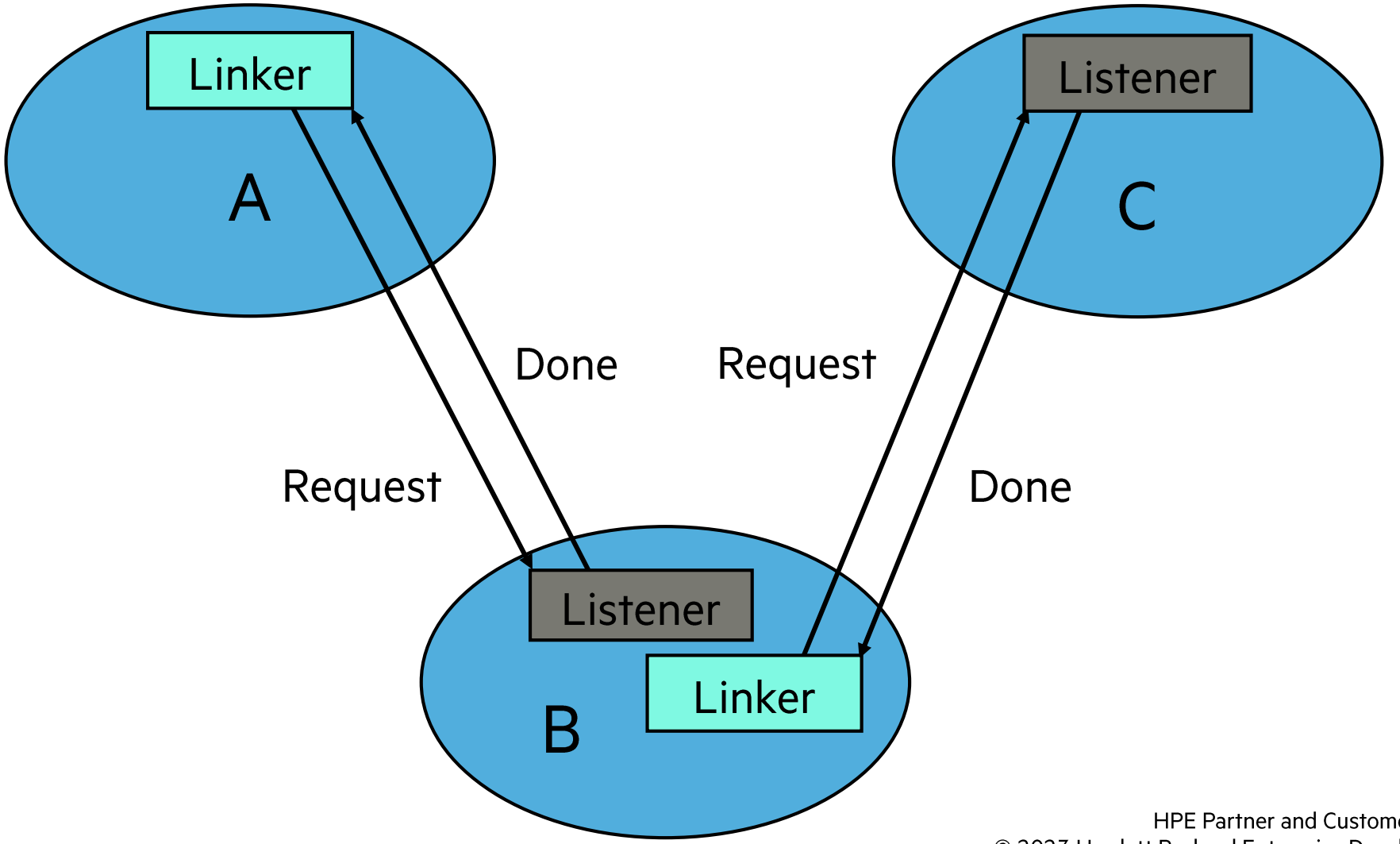
- NSK runs on standard available hardware, so the hardware is not special for NonStop.
- NSK is a message-based operating system giving you:
  - Processor fault tolerance, because each logical processor has its own copy of the operating system not sharing anything with another logical processor, and the message system offers them to communicate with each other and when one fails the other can take over the load, so the application stays up.
  - Software (process) fault tolerance, because the primary process running in one CPU can exchange messages with its backup process with enough information, that when the primary dies, the backup can continue to work where the primary stopped.
  - Scalability – you can start with a small 2-processor system and grow to a 16-processor system and with the same message system over Expand grow to a HPE NonStop network of 4080 processors without to change the application.

Interprocess Communication

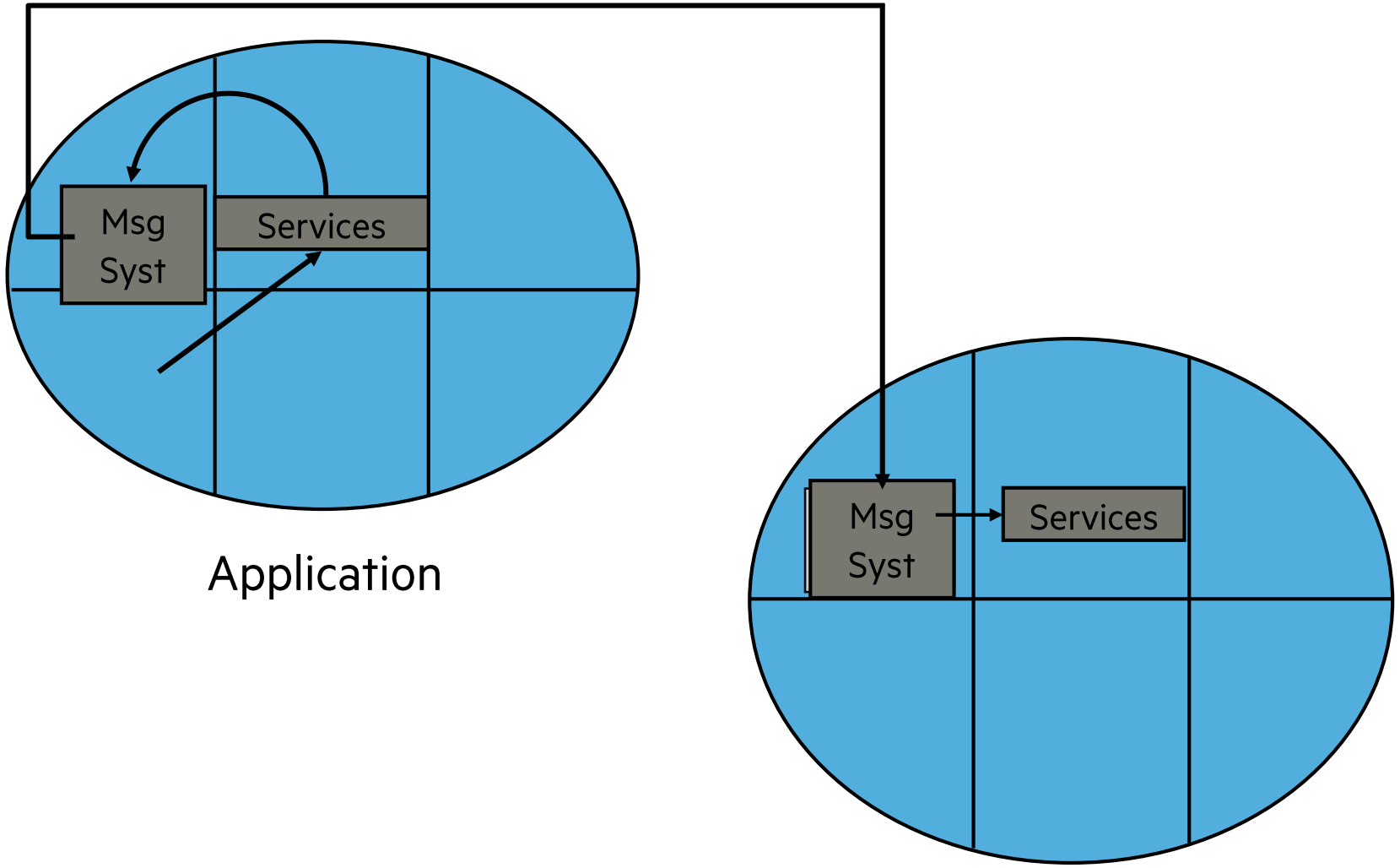
What the Message System Provides

- Geographic independence.
- Fast, priv interface.
- Transport.
- Detects and recovers from errors.
- Message interface is 2-way: Request/reply.
- Nowait interface: Client chooses when to wait for return of incoming message or reply message.
- Sessionless: Unlike the file system, no open or connection establishment necessary. Can always send a message.
- Fault tolerance.

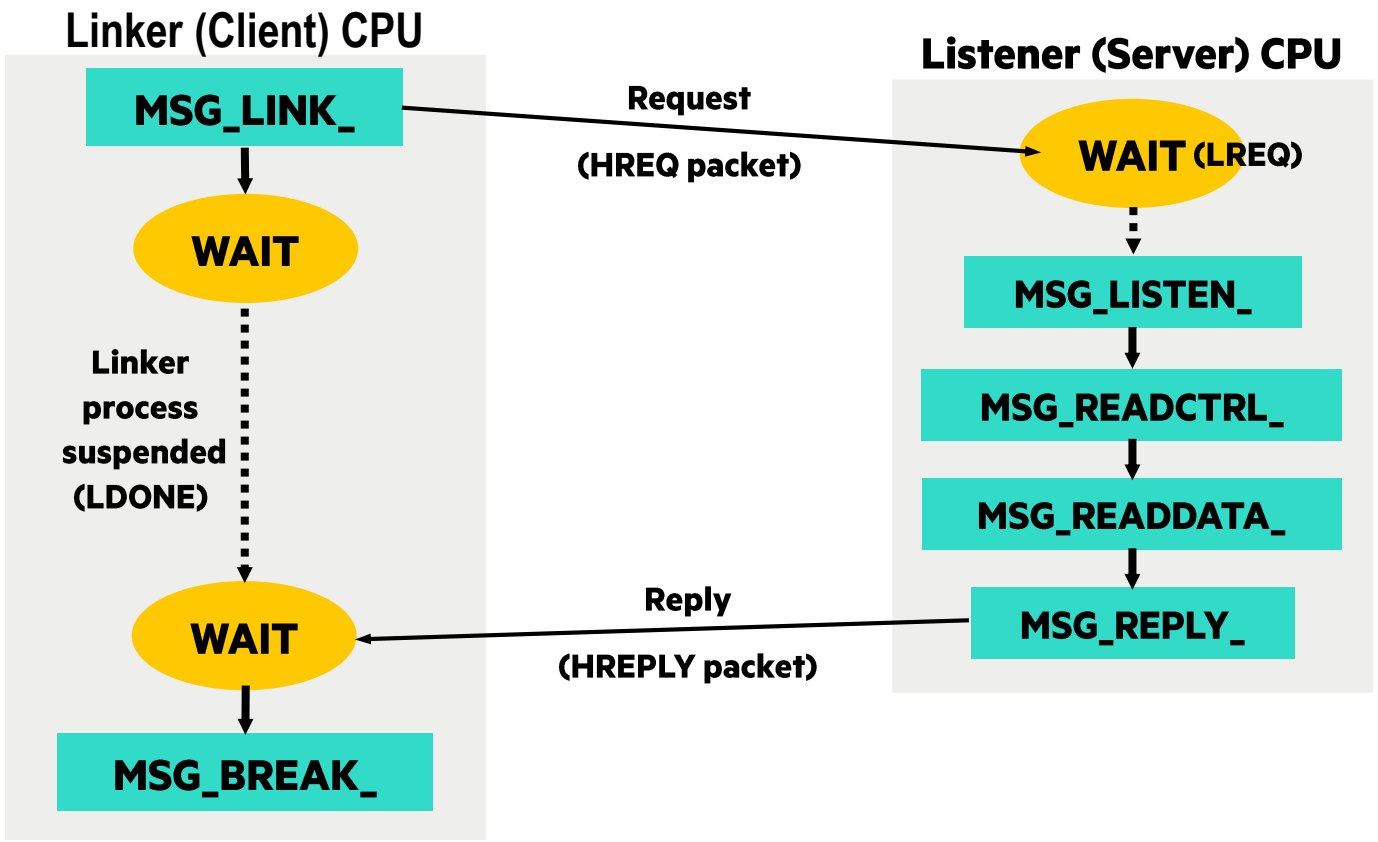
Interprocess Communication — Message System



Message-System Customers

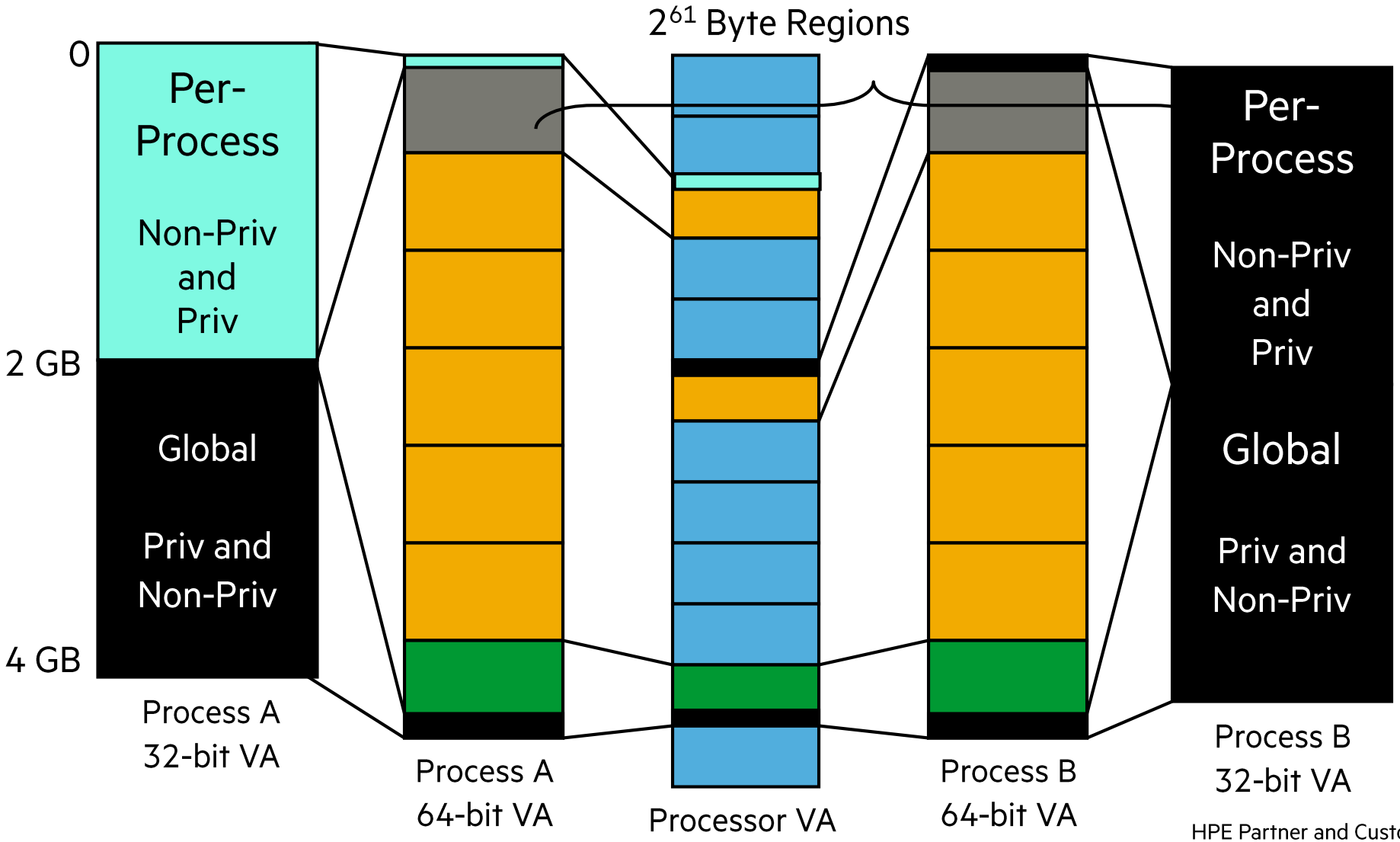


Message System Transfer Protocols

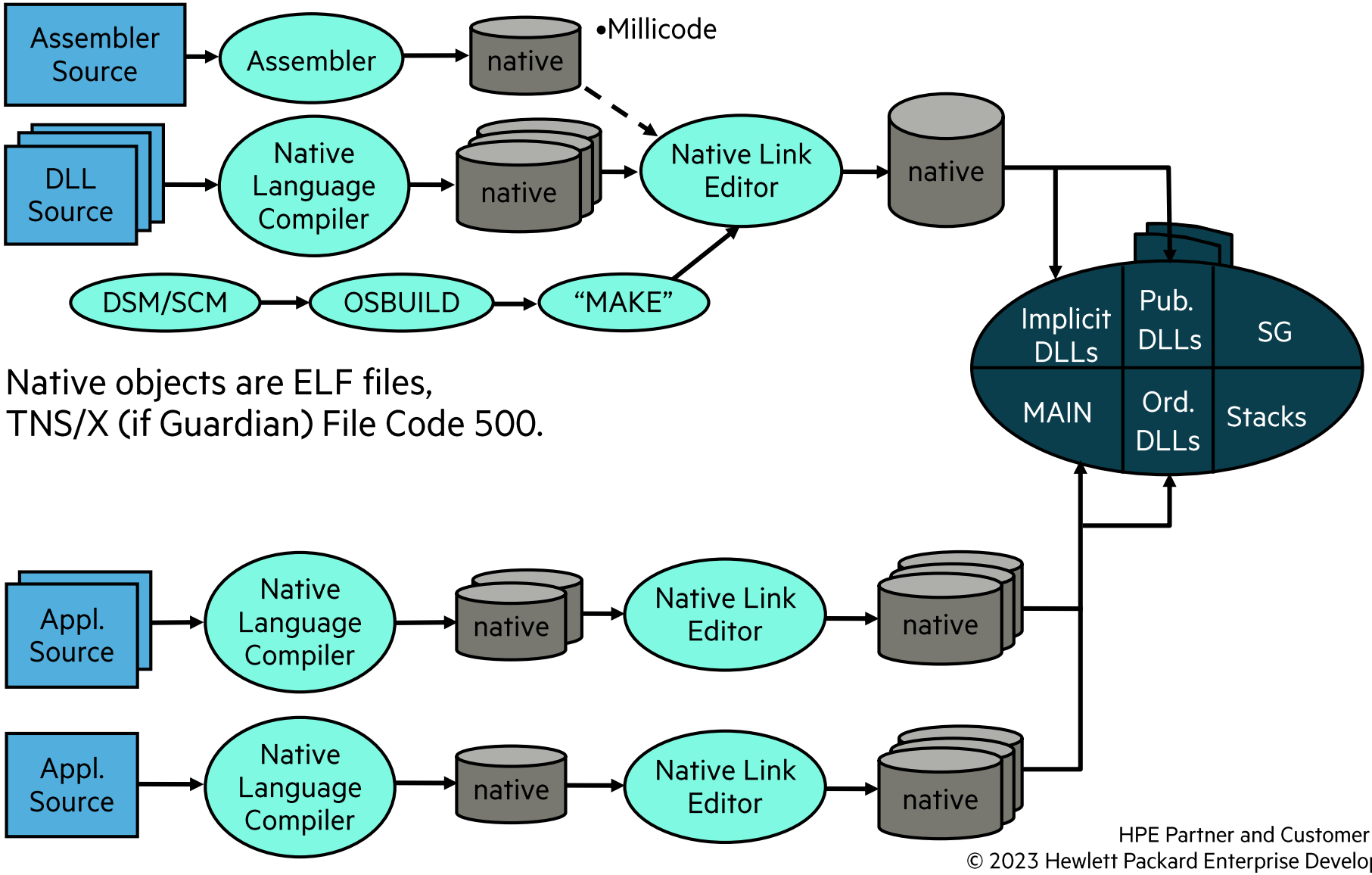


Code Generation, Address Space, Code Space, and Native Process Execution

Virtual Address Spaces

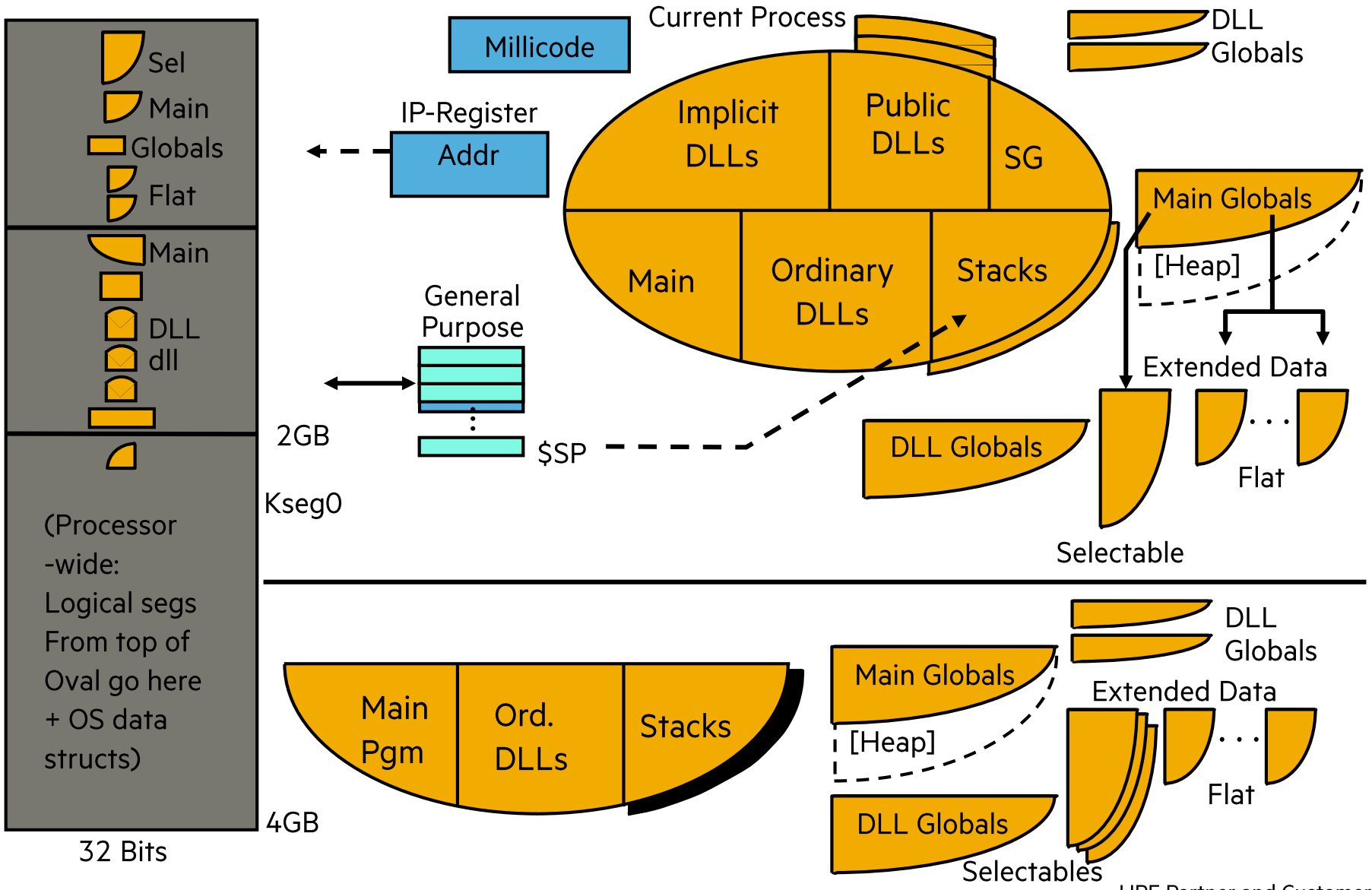


Native Code Generation

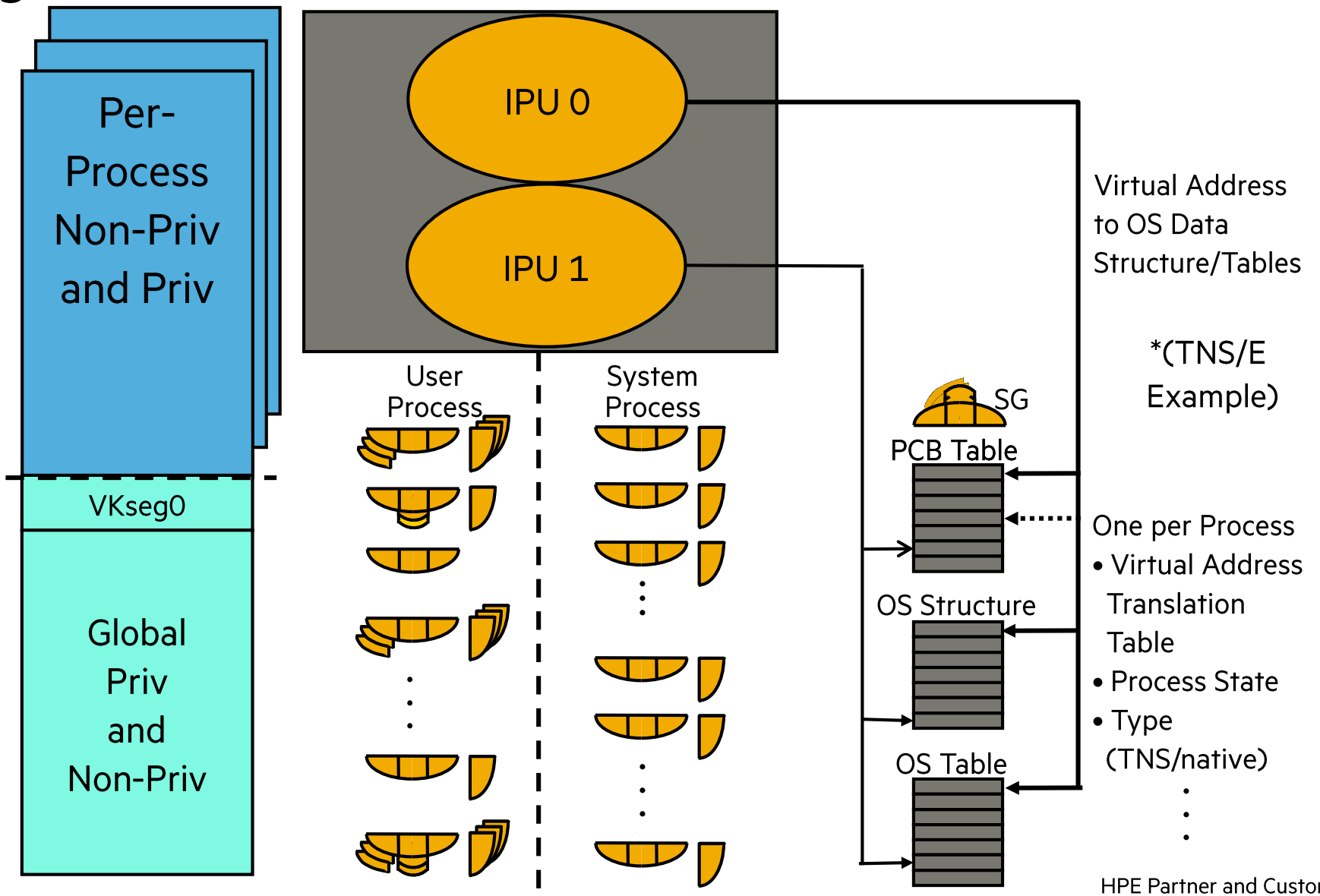


Native objects are ELF files,
TNS/X (if Guardian) File Code 500.

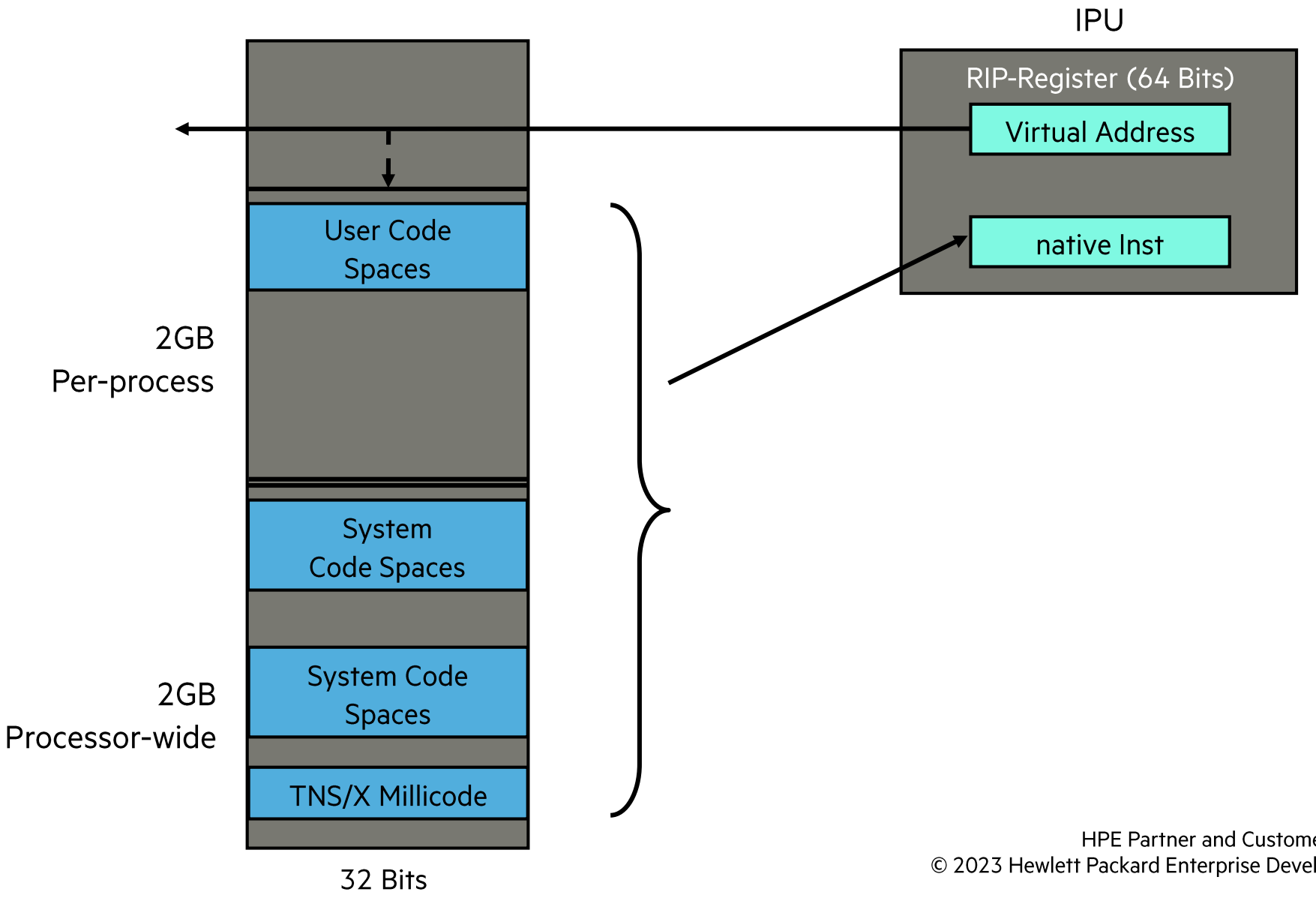
Native Process Execution Environment



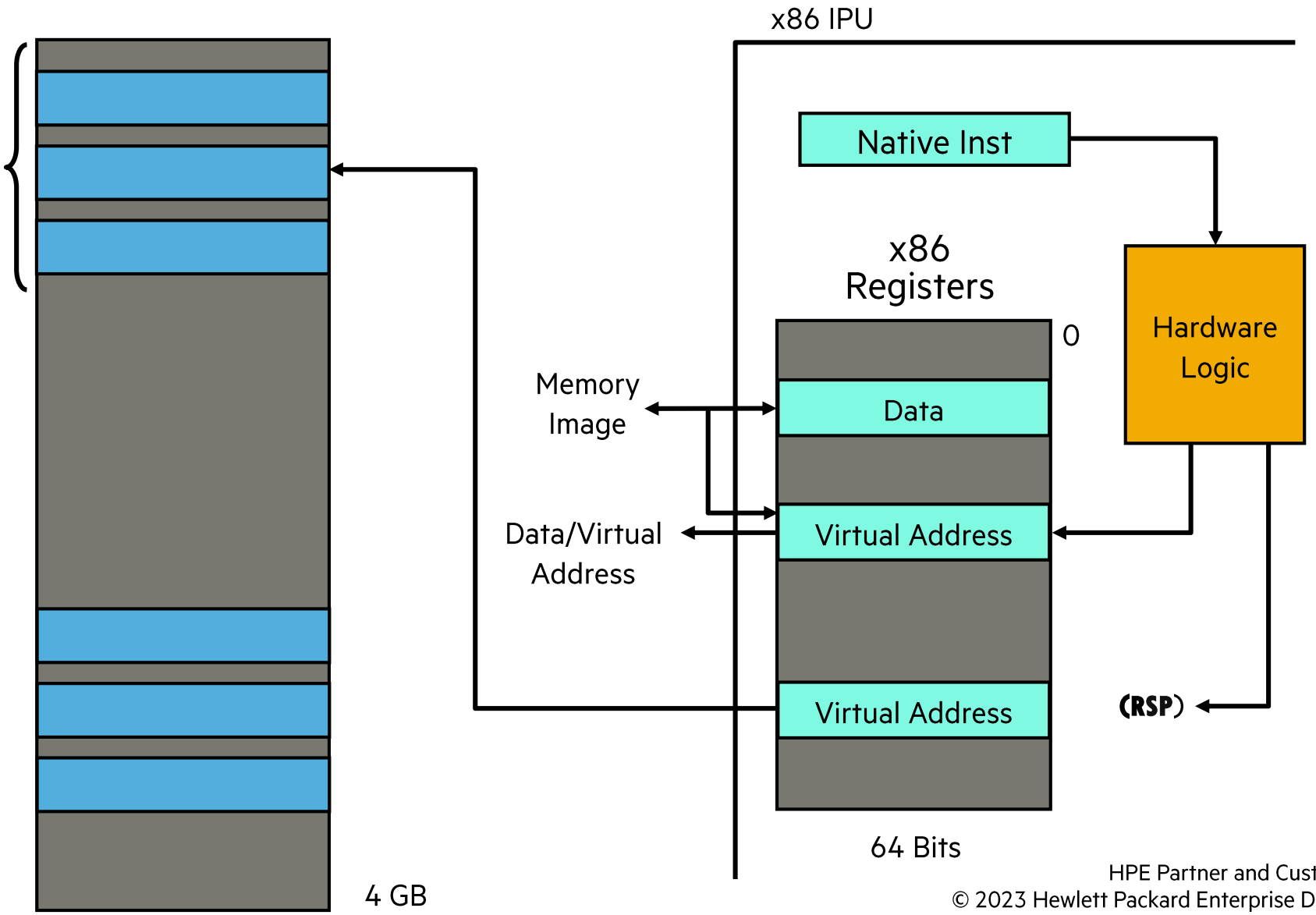
NonStop Logical Processor



Multiple Code Space Support

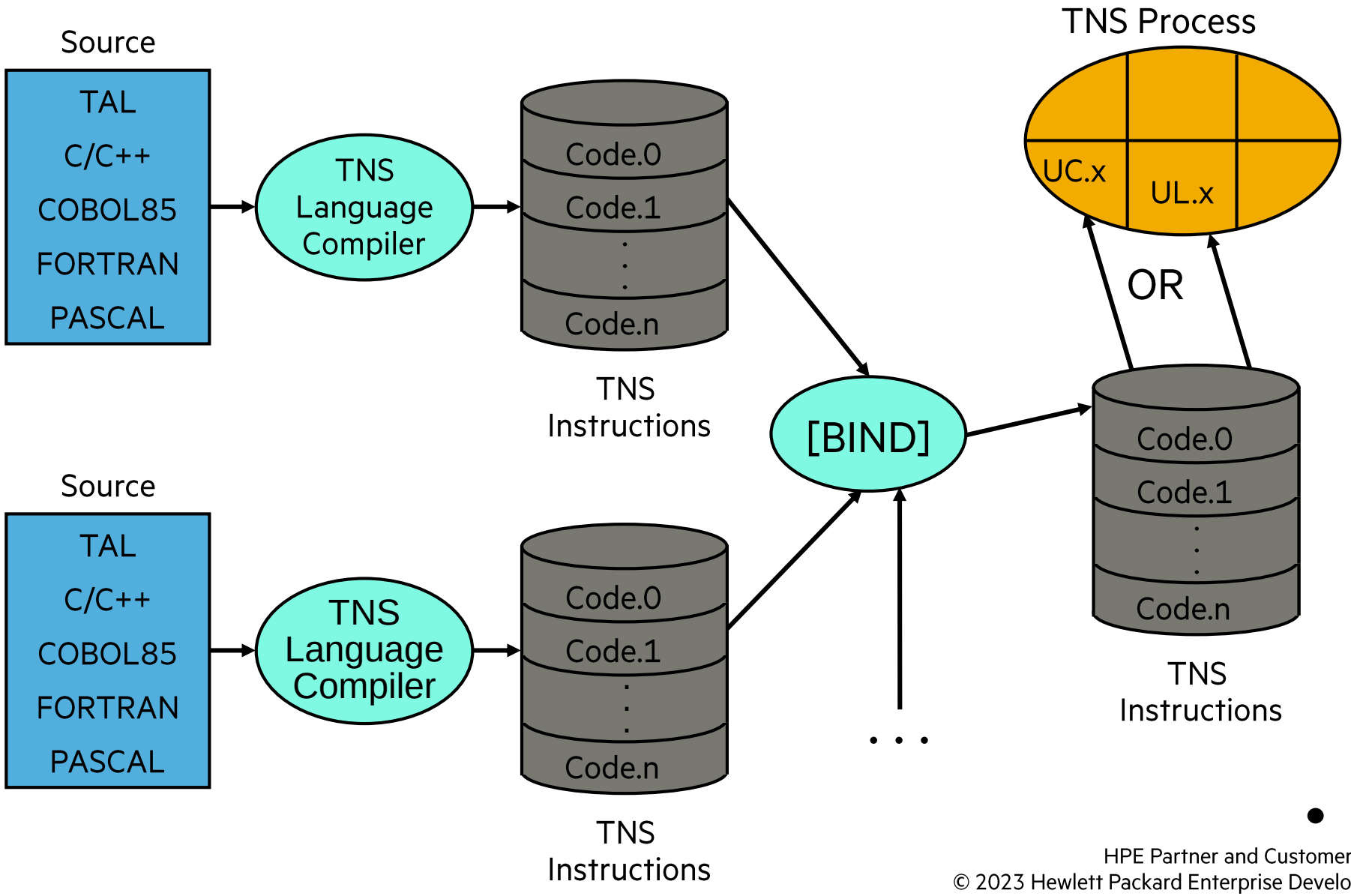


Native IPU — Data Access



Code Generation, Address Space, Code Space, and TNS Process Execution

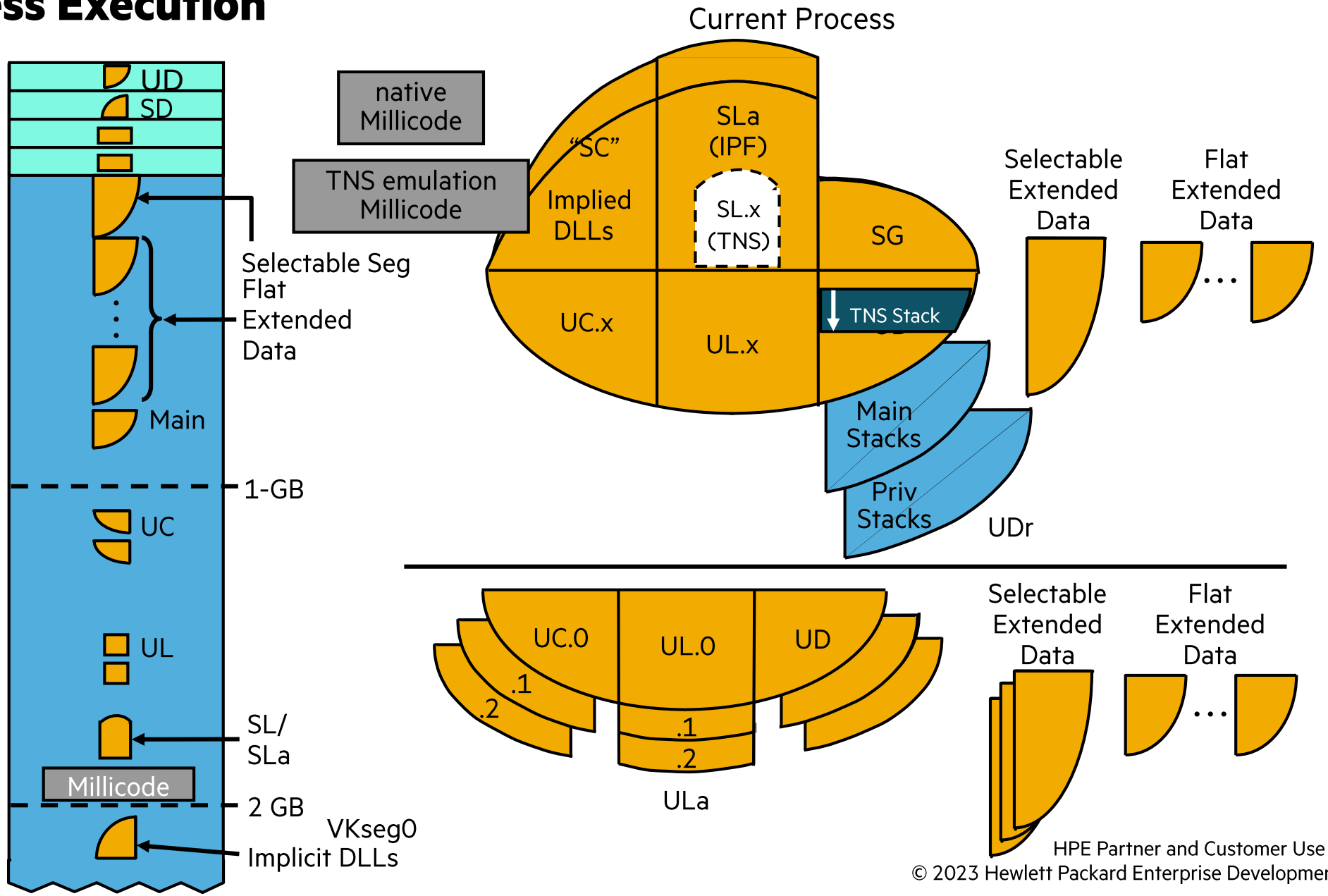
TNS Code Generation



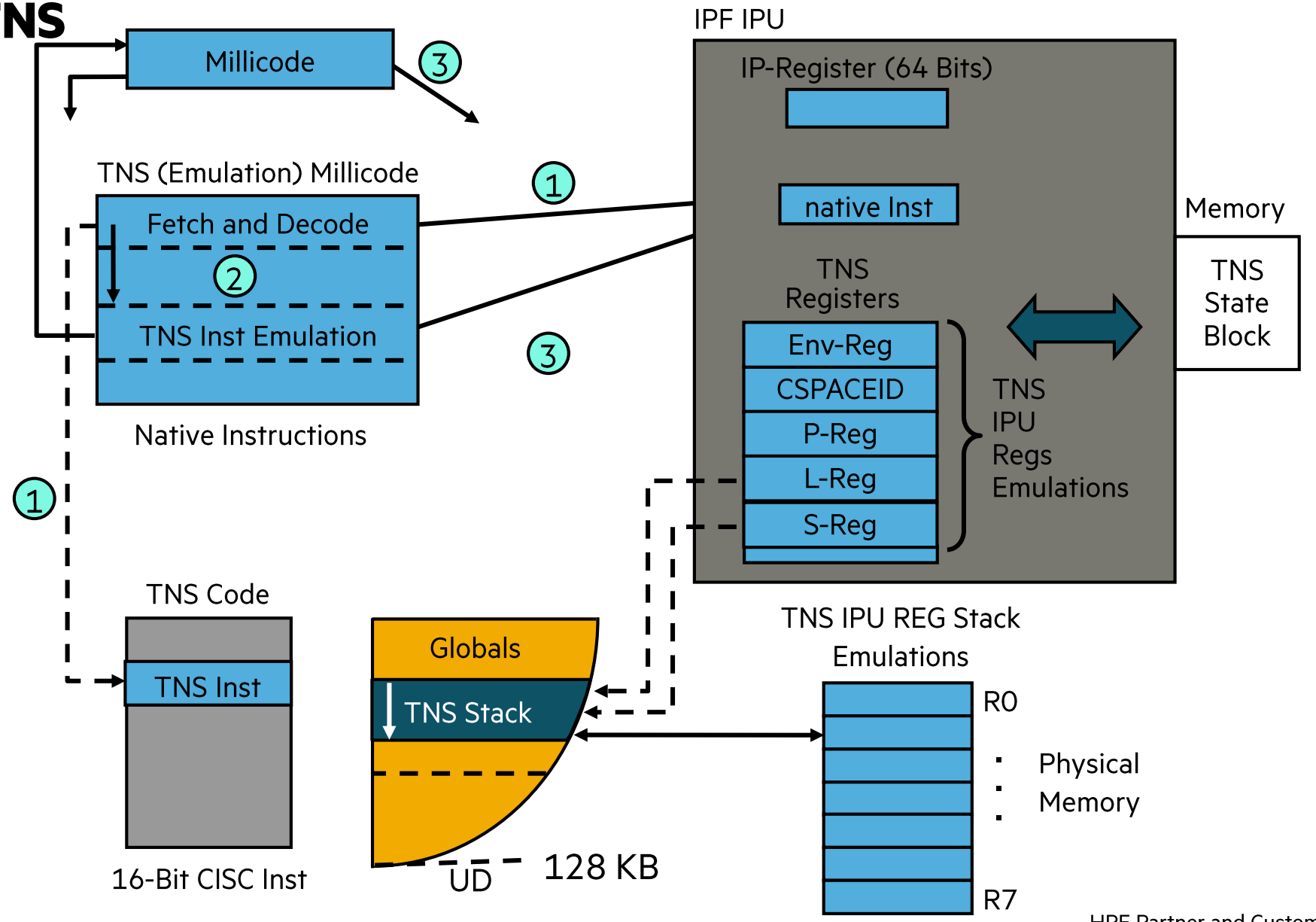
TNS Register Architecture

- ENV register – various status fields
  - 3 bits form the register pointer (which is call RP and frequently used like is was truly separate register)
- Register Stack of 8 16-bit registers, with RP pointing to the current top of stack. This top of stack register is used implicitly by most TNS instructions
- P register – the TNS instruction pointer
- S register – the current top of memory stack register (grows positive)
- L register – the base of the current memory stack frame

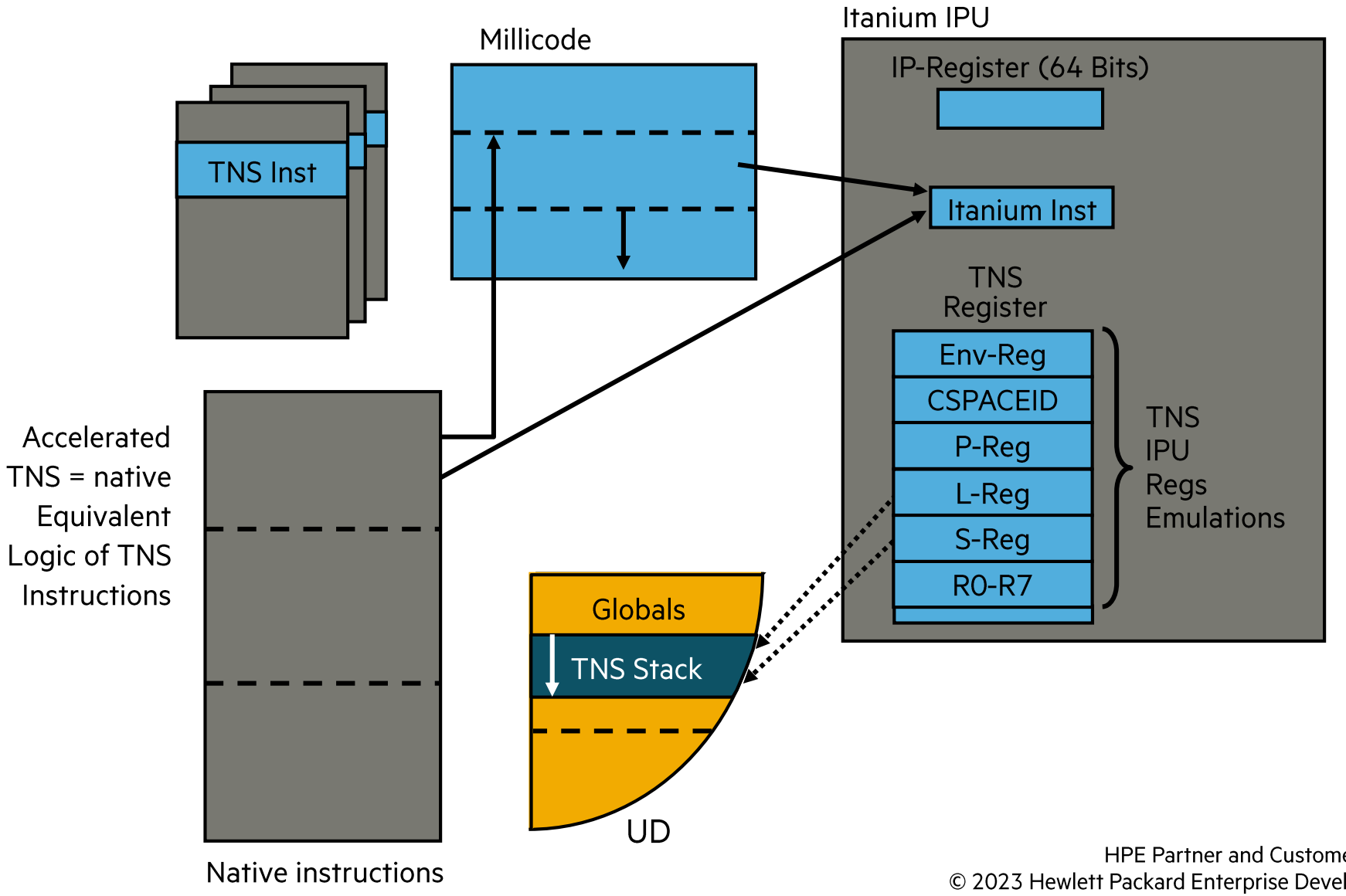
TNS Process Execution



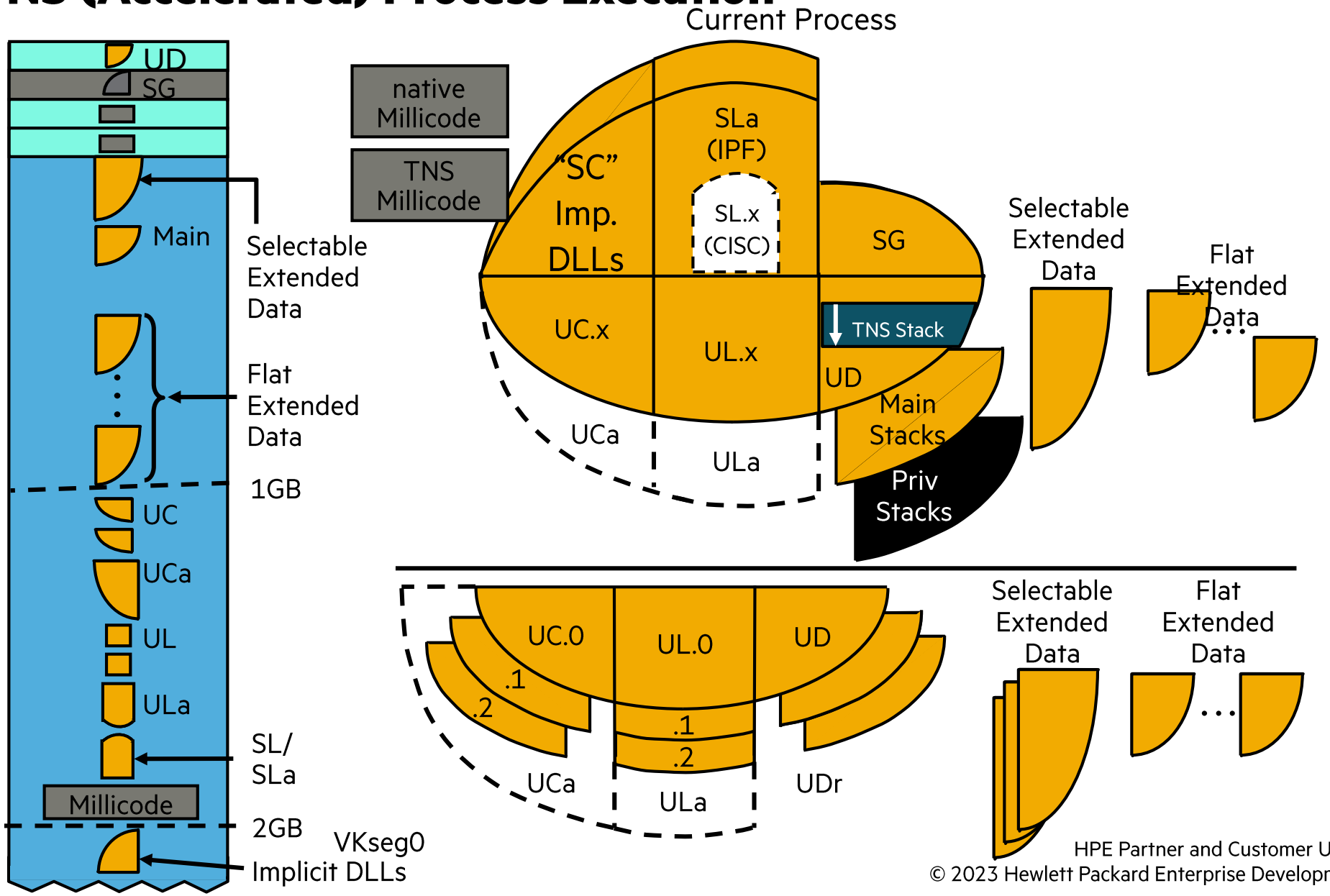
Executing TNS



Accelerated TNS Execution



NonStop TNS (Accelerated) Process Execution



Part 2

How is the x86 Based processor used?

- Memory Management
- Little and Big Endian

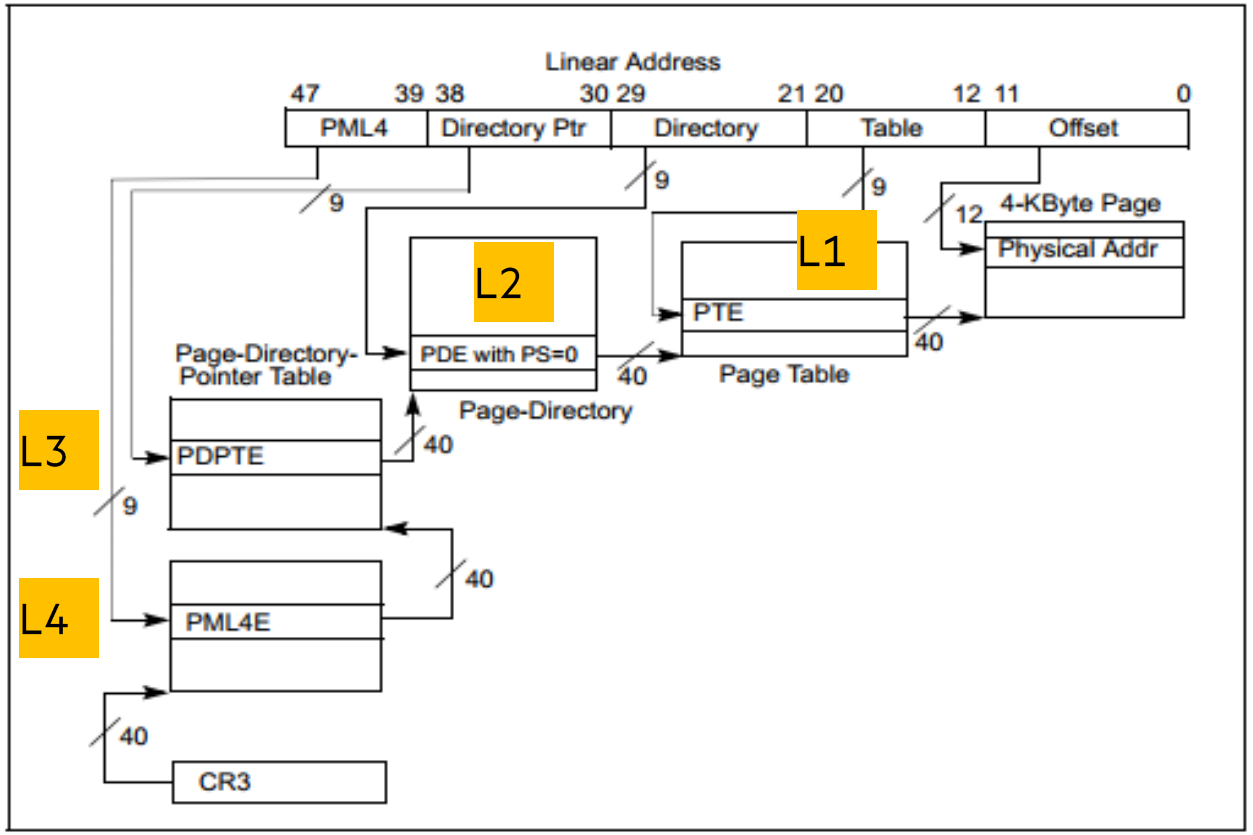
Hardware - XPF

Translation Hardware

- x86 is a traditional CISC architecture:
  - The HW does a page table lookup to access the physical page and the TLB caches the result. It may also cache the page tables themselves.
  - Page tables are shared among all IPUs
- The page tables are shared among all IPUs of the CPU.
  - So, XPF uses the GS (or %gs) register for that purpose.
- The system always runs in 64-bit mode and uses four level page tables.

X86-64 Page tables, 4KB pages

- A canonical address is x86-64 is where the 52-48 bit virtual address is sign extended.
- To make a 16KB page 4 4KB PTEs are updated as a group (interrupts masked). The dirty and other bits when checked are the orring of the four entries.
- The TLB is loaded by the HW when a lookup is done.
- Each of the four tables (L4, L3, L2, and L1) have 512 entries.



Linear-Address Translation to a 4-KByte Page using 4-Level Paging

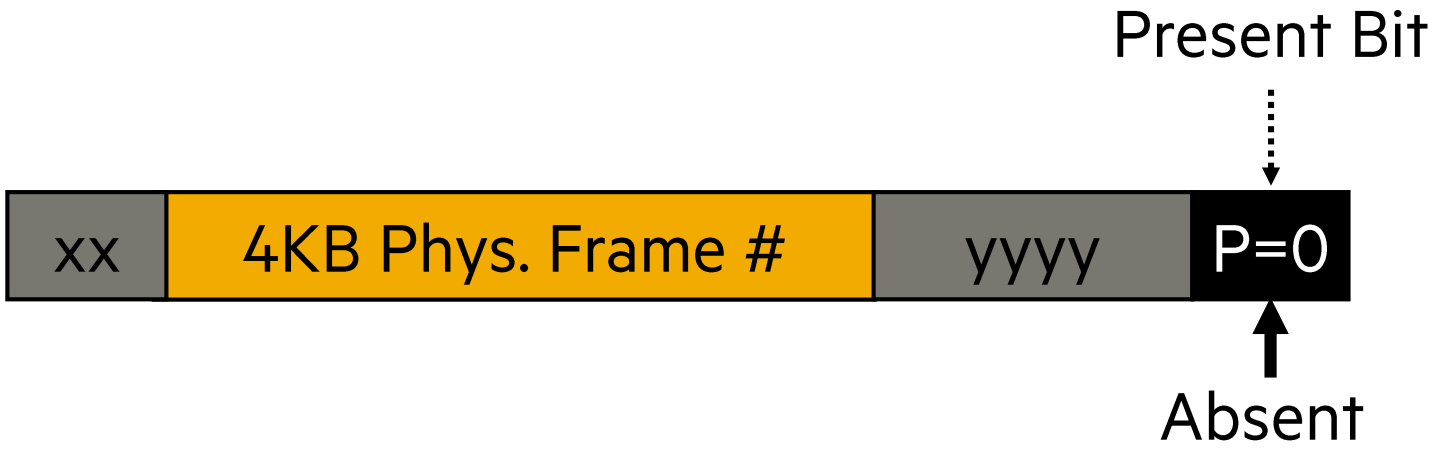
Page Faults

Lesson 8

XPF: Software Interrupt Vector Code

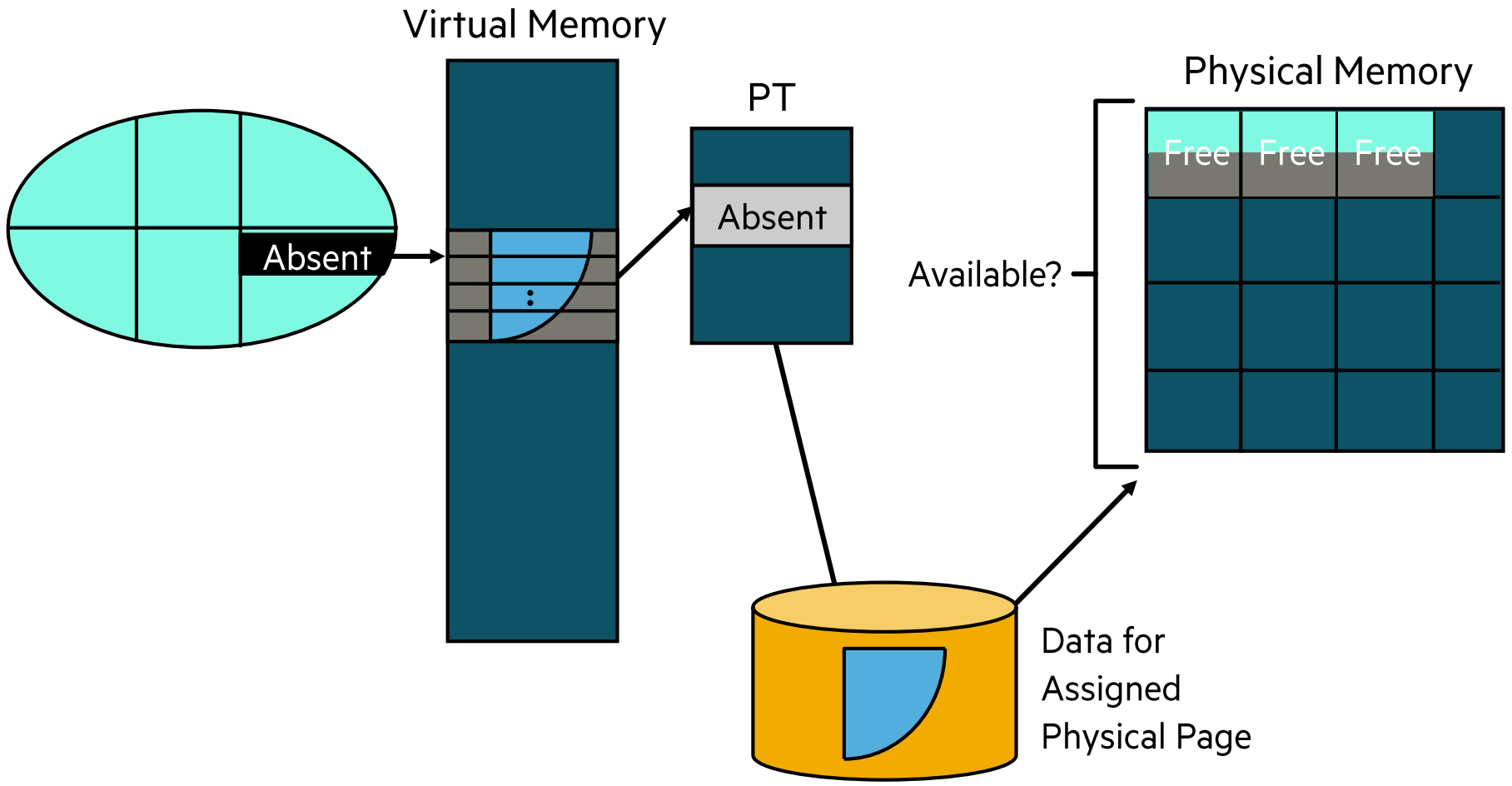
- A TLB miss causes a HW page table search which finds an invalid page (for this current kind of access).

XPF: L1 (Page Table) Entry — Absent Page

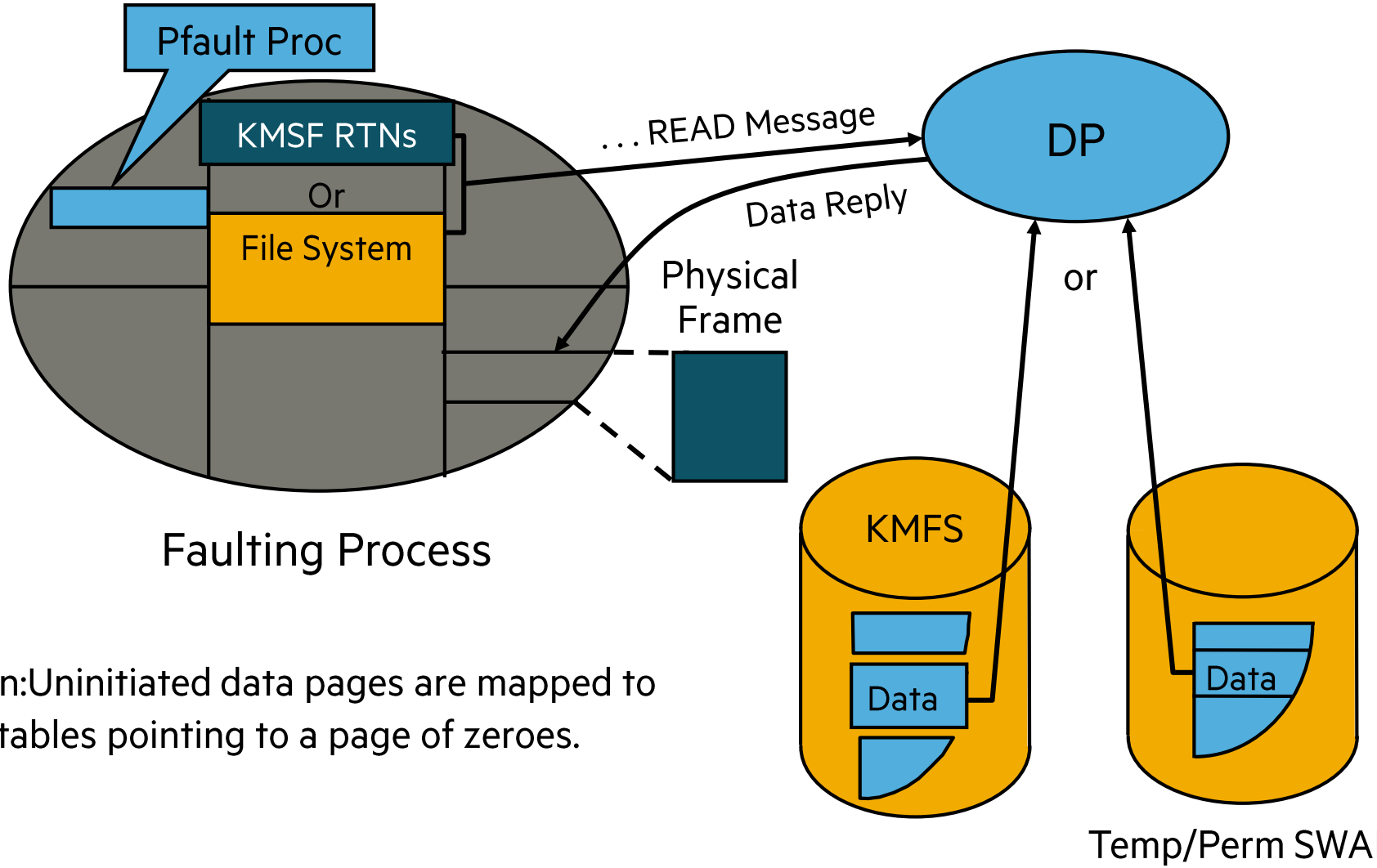


- Xx: flags denoting execute disable, and 4-bit protection key
- Yyy: flags denoting read/write, user/supervisor,write through,cache disble,accessed, dirty, global

Absent — Page Fault

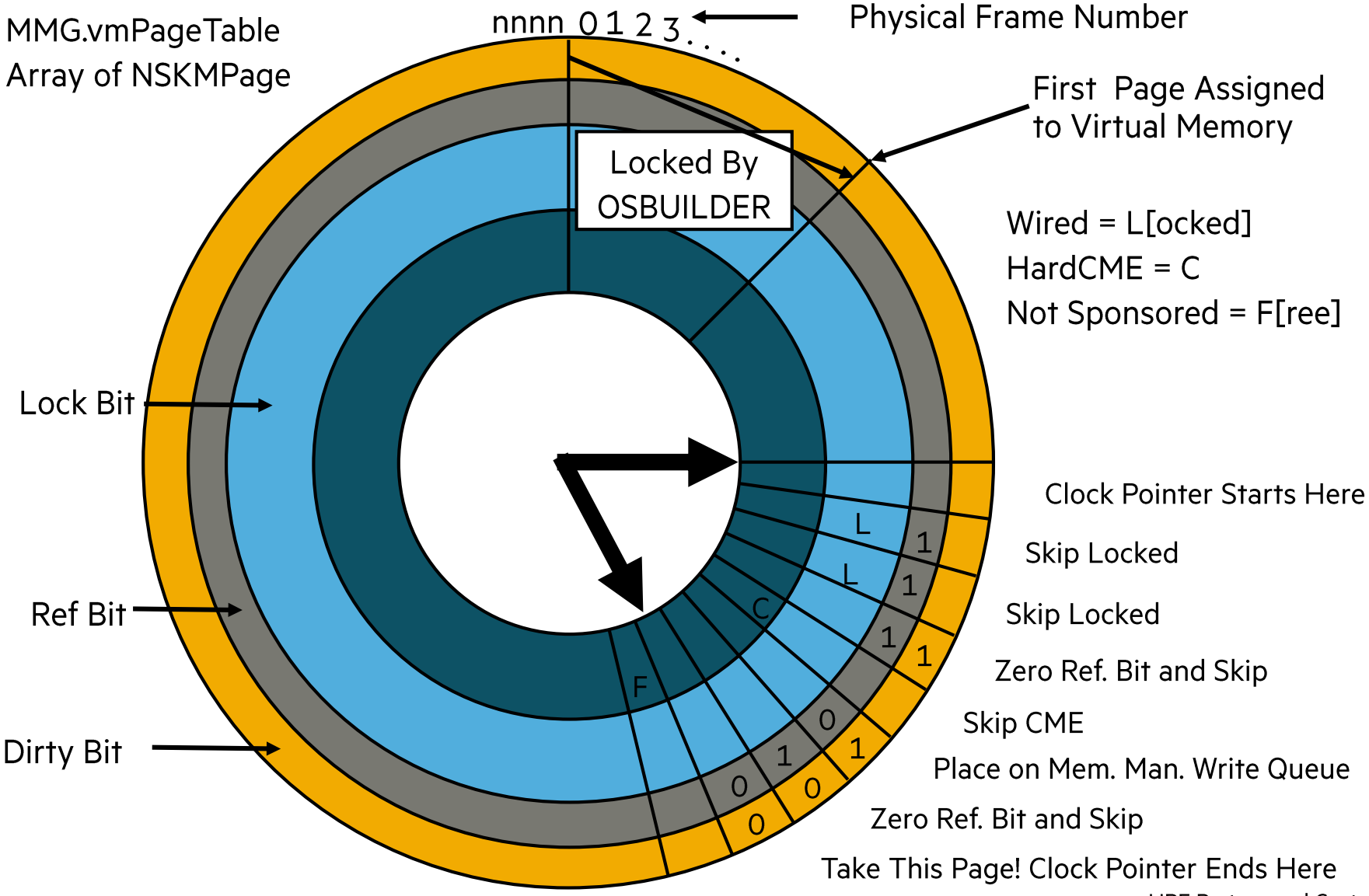


Pfault Proc Initializes the Fault-In Page in Faulting Process Context



Not shown:Uninitiated data pages are mapped to mapping tables pointing to a page of zeroes.

Frame Selection — Clock Algorithm



Memory Manager Process Role Reduced

- Page faults are handled in the context of the faulting process.
- The functions of the memory manager process have been drastically reduced. The memory manager now:
  - During initialization, common opens implicit DLLs.
  - Replenishes the supply of pages available for allocation under mutex.
  - Cleans pages.
- Every 10 seconds, runs the clock to:
  - Age pages (turn off reference bits).
  - Add dirty pages to the dirty page queue.
- Woken up on INTR when there are dirty pages.
  - Writes them out to disk and puts them on a “stealable” list.
  - Can wake up a process waiting for a page.

Endianism - Big vs Little endian

Big vs Little Endian

- Big Endian means that for a given datum the high order byte is stored first in memory (lowest address) and the low order byte is stored last (highest address).
- Little Endian means that for a given datum the low order byte is stored first in memory (lowest address) and the high order byte is stored last (highest address).
- So, for the number x01020304:

address	0000	0001	0002	0003
Big	01	02	03	04
Little	04	03	02	01

- TNS, MIPS, and Itanium are Big endian.
- X86-64 is Little endian

Making XPF Big Endian

- NonStop runs x86 in “Big Endian by doing the following:
  - All program (source level) variables are written to memory as big endian datums.
  - For data that is little endian in memory but must be presented to a user/program the data is changed to big endian format.
- Some parts of the system actually run in little-endian mode (i.e., it sees real little-endian data). This include the TNS emulator and millicode.
- So, if you look at raw memory of a process there is a mix of little- and big-endian items.
- Some things in little endian format:
  - Page table data.
  - Addresses pushed onto the memory stack by a call or interrupt.
  - Register values in a jump buffer.

What parts of the system know about Little endian?

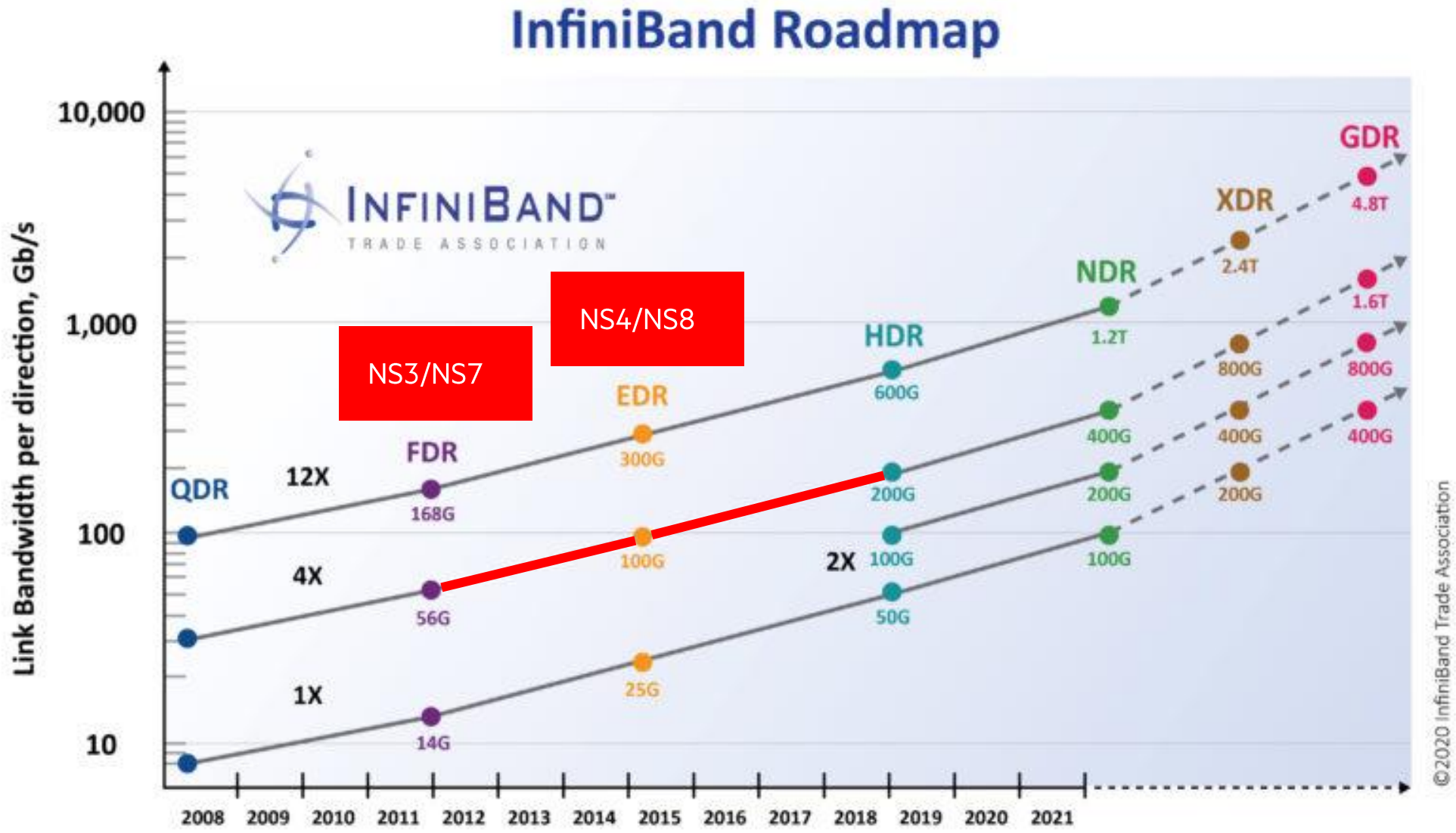
- The OS part that deals with page tables, and other structures known to the x86-hardware.
- The debuggers which know the difference between program variables and “hidden” data.
- The TNS emulator.
- All native compilers.

Part 3

- InfiniBand
- Cluster I/O Module subsystems

InfiniBand

Fourteen Data Rate InfiniBand (4X)

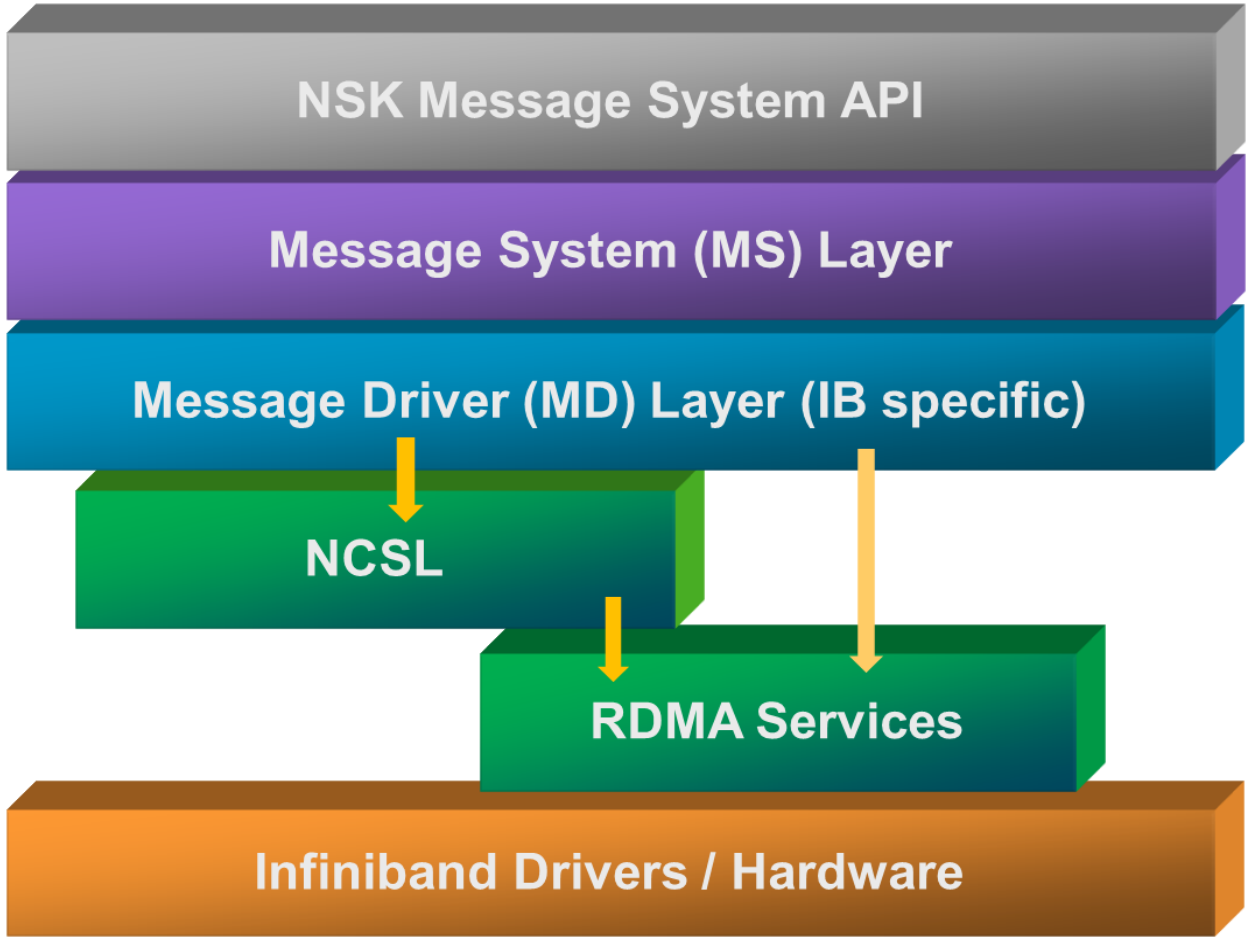


ServerNet vs InfiniBand Message System Differences

- There is no SvNet address space in IB.
- Normal sends are transferred from the buffer on the linker's CPU to a buffer which is on the "receive" part of the Queue Pair on the Listener's CPU. The actual memory address is not known to the linker.
- Large transfers are done because the linker of listener has called a MSG function
  - listener MSG_READDATA_ - a pull of data from the linker
  - linker MSG_REPLY_ - a push of data to the linker from the listener
- These are controlled by the "memory region" mapped on the source and destination CPUs
  - Which is what prevents rogue memory transfers.
- On SvNet the fabric choice is fully up to the MS but on IB the NCSL layer handles fabric failover. The MS has some input into the choice and in fact causes both fabrics to be used for some high priority traffic.
- On SvNet we allow up to 4 packets to be inflight at the same time, with IB that number goes up to 8. Moreover since we have multiple connections the actual number is 24.

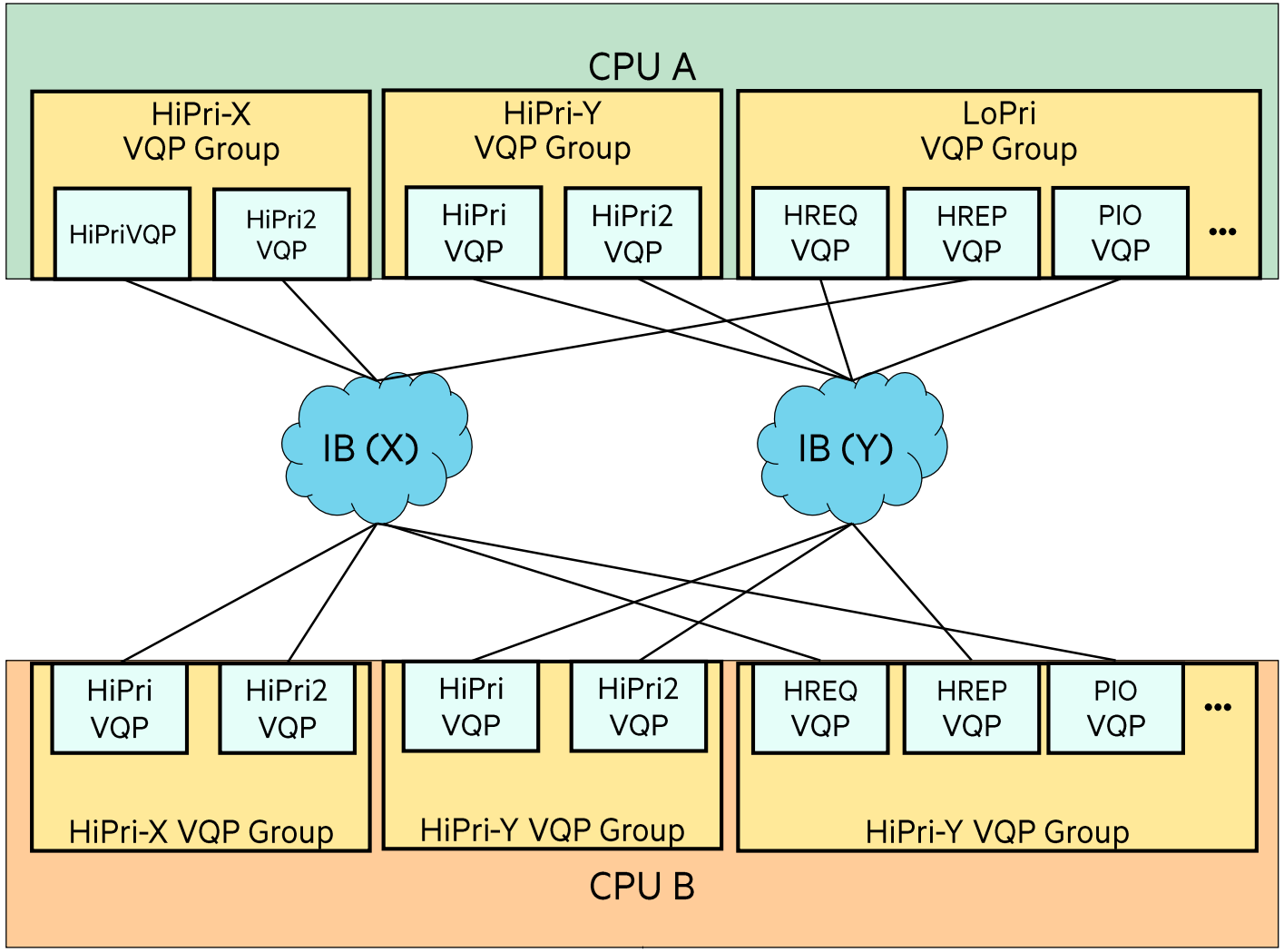
XPF Message System layers

On InfiniBand the Message System has a similar set of layers, with the Message Driver layer changed to support IB and talk to the NCSL API and RDMA Services below it instead of the TNet Services:



IB Connections vs SNet Connnectionless

- Using SvNet each message includes the address of the target CPU.
- However, IB is a connected protocol, so each CPU is connected to every other CPU. (A la sockets)
  - Moreover, there are multiple connections to allow the different classes of messages to move independently.

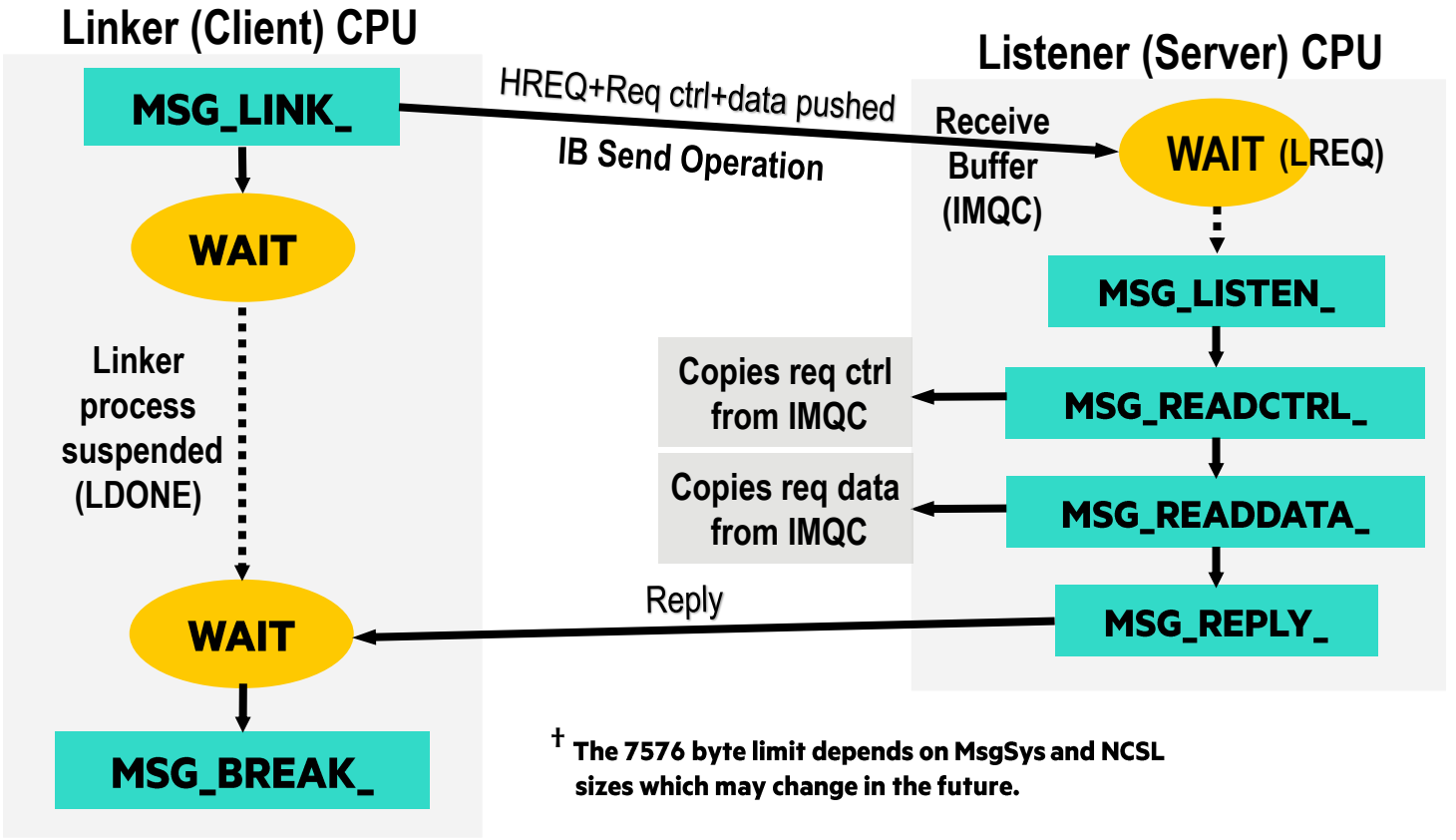


Message System Transfer Protocols on IB

Sending request : ctrl + data size ≤ 7576† bytes

(All the data is pushed along with the HREQ packet in an IB Send operation)

- The generic MS link operation is mapped to an IB send
- The MS reply is also linked to and IB send.

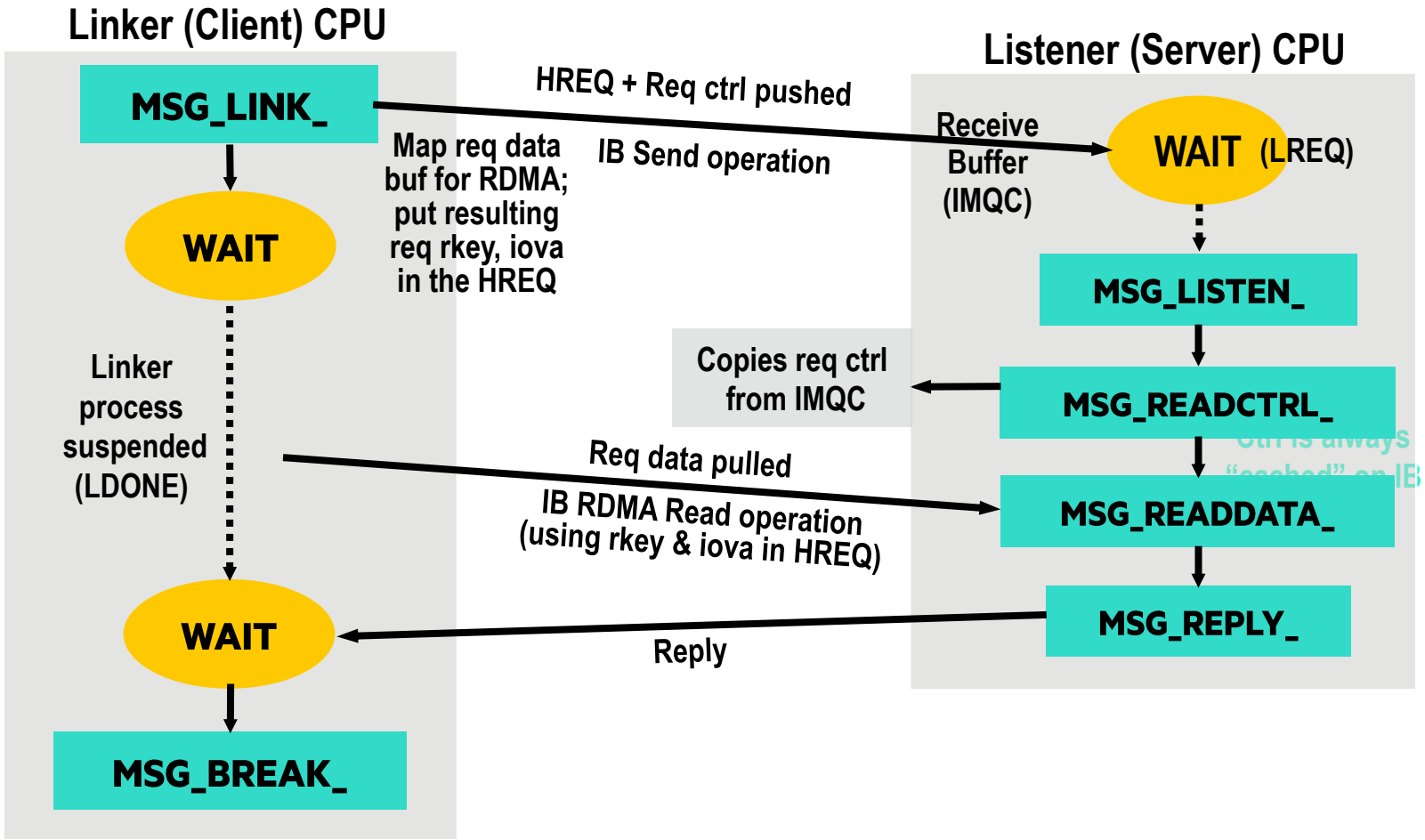


† The 7576 byte limit depends on MsgSys and NCSL sizes which may change in the future.

Message System Transfer Protocols on IB

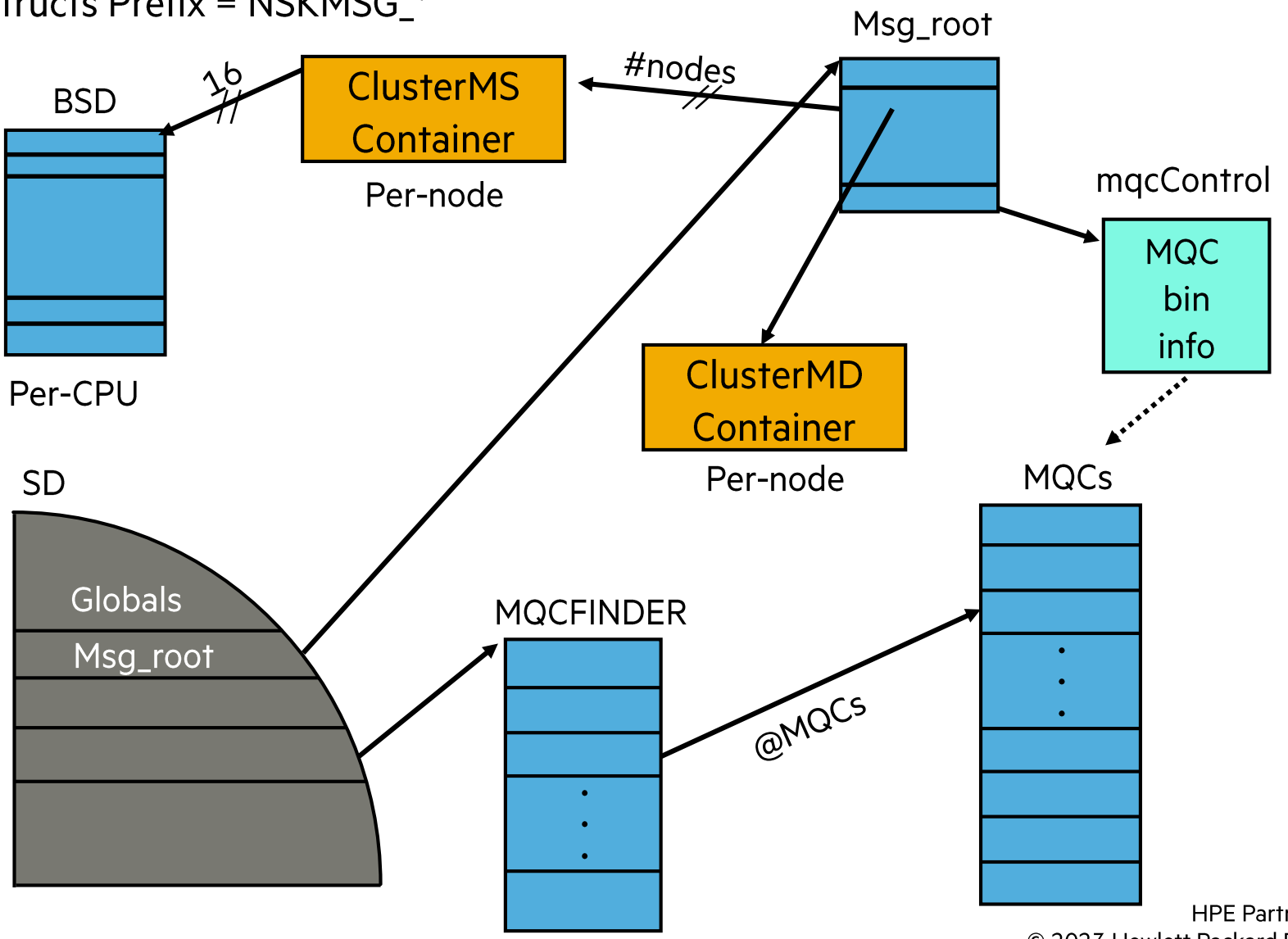
Sending request : ctrl + data size > 7576 bytes

- The req ctrl is pushed along with the HREQ packet in an IB Send operation.
- The link data is obtained with a IB RDMA read operation.



Message-System Data Structures

Data Structs Prefix = NSKMSG_*



General NonStop I/O

- I/O transfer:
  - I/O as remote memory transfer.
  - NonStop L-series: InfiniBand uses IB connections and its memory mappings.

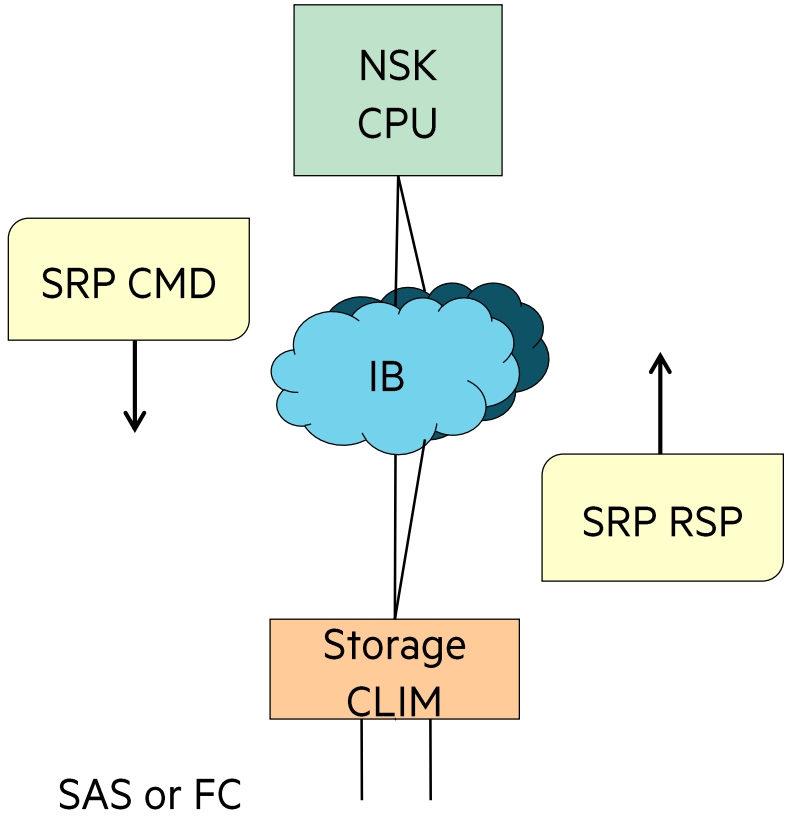
Storage I/O

- High level view of storage I/O
- Life of an I/O operation
  - Initiation
  - Data transfer
  - Completion
- Starting a device

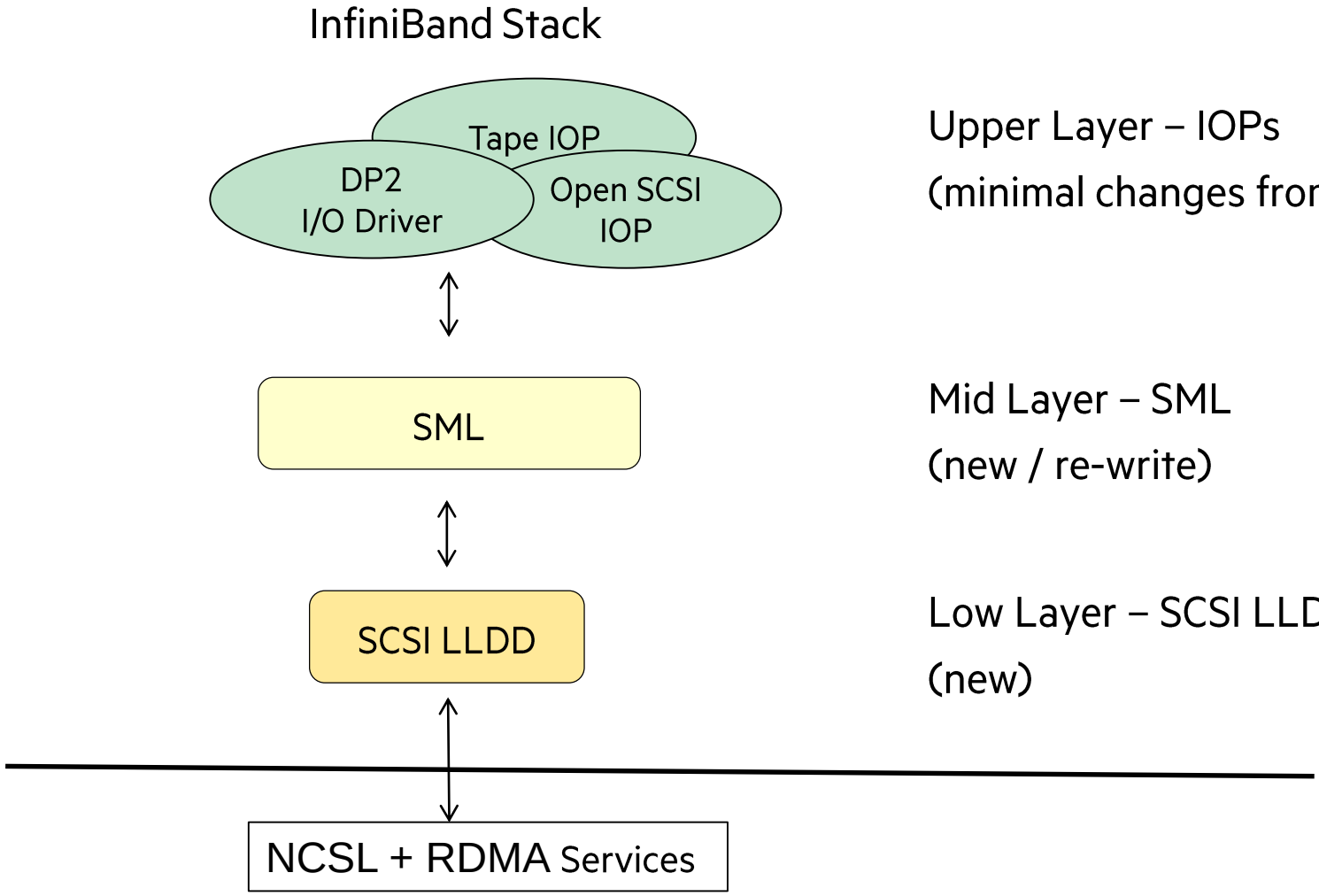
IB based Storage Subsystem

Storage Protocol

InfiniBand: SCSI RDMA Protocol (SRP)



IPF and XPF Storage Software Stack Comparison

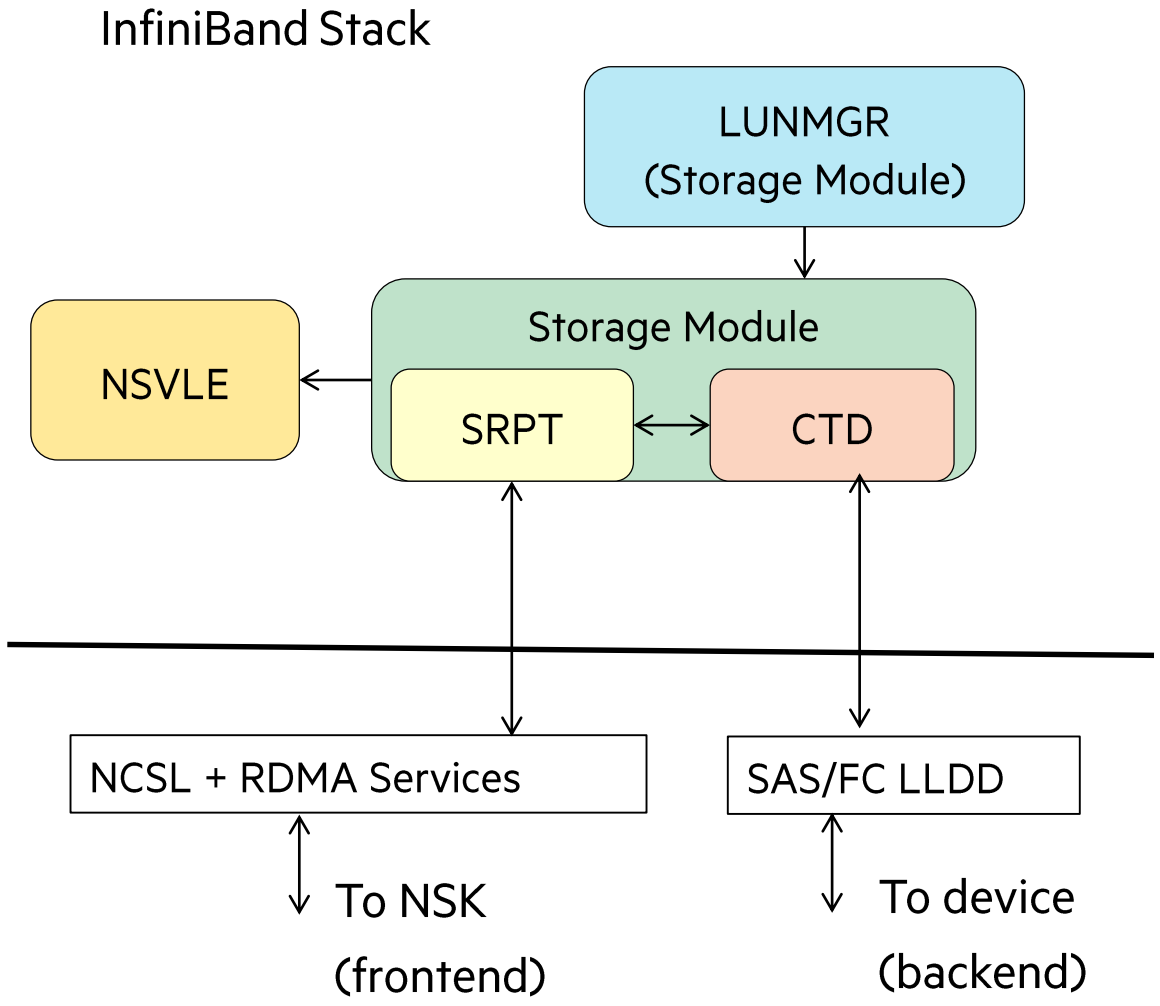


Upper Layer – IOPs
(minimal changes from SNet)

Mid Layer – SML
(new / re-write)

Low Layer – SCSI LLDD
(new)

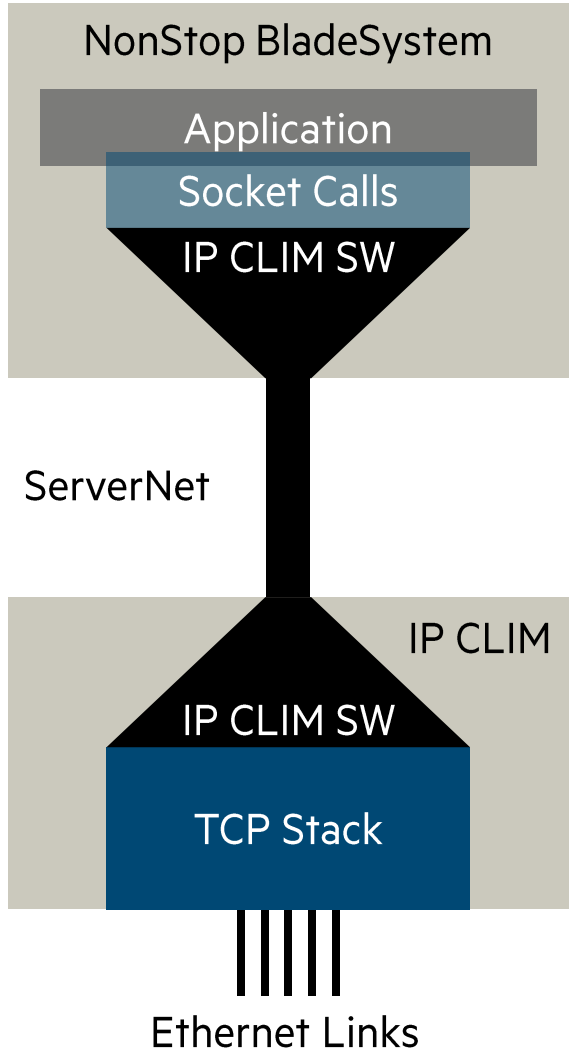
Storage CLIM - CLIM Software Stack



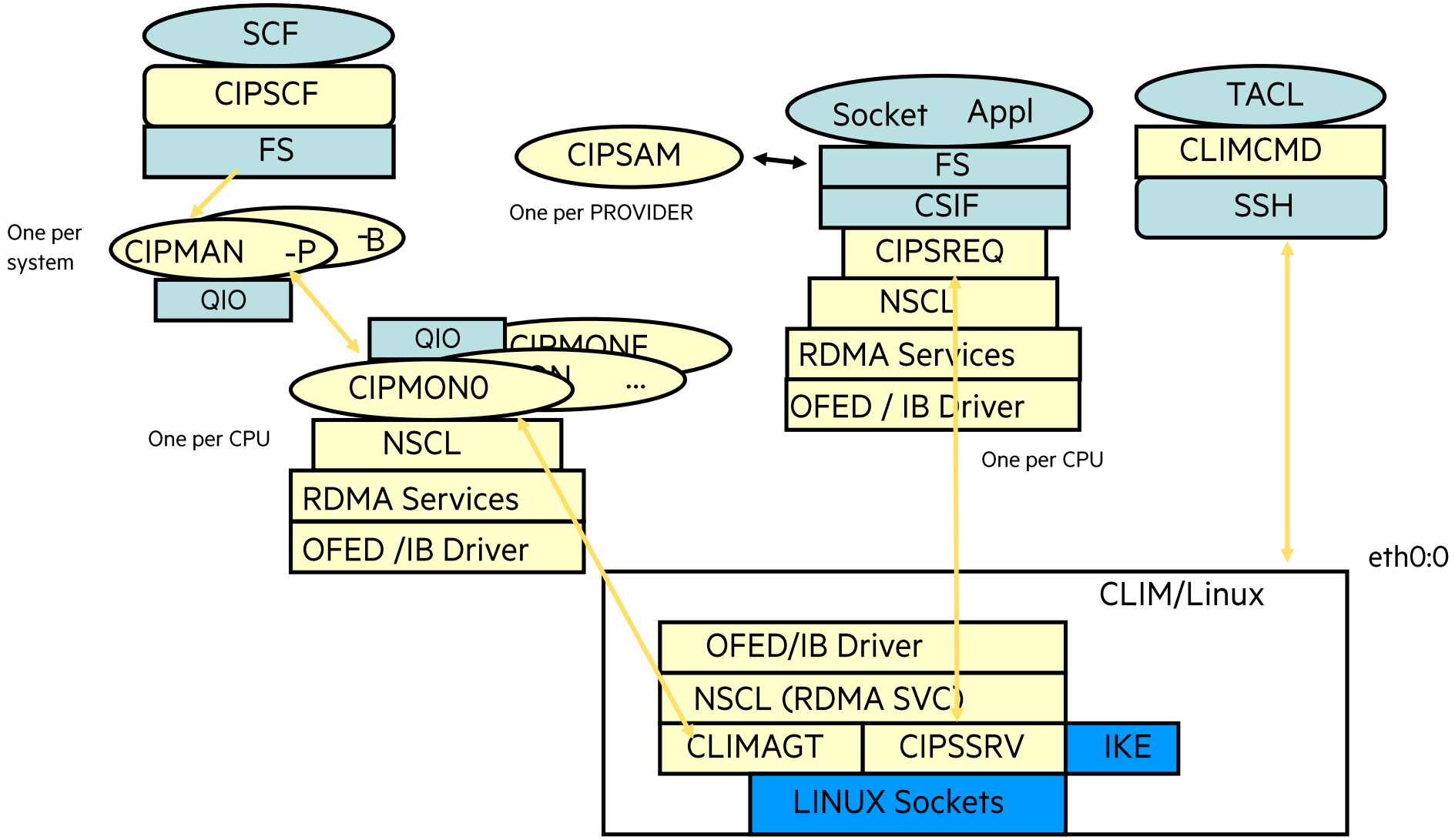
IB based TCP/IP Subsystem

Theory of Operation - IP CLIM

- Cluster I/O Protocol (CIP) Subsystem
  - Provides a configuration and management interface for I/O
- Uses a Linux server as a front end
  - Moves network stacks from the NonStop host to a Linux server
- Leverages the Open-Source Linux environment
- Provides support for IPsec, SCTP, IPv6
- Dual Infiniband fabrics connections
- Provides Gigabit and 10 Gigabit Ethernet interfaces
- The CLIM provides the physical interface to the network or storage devices



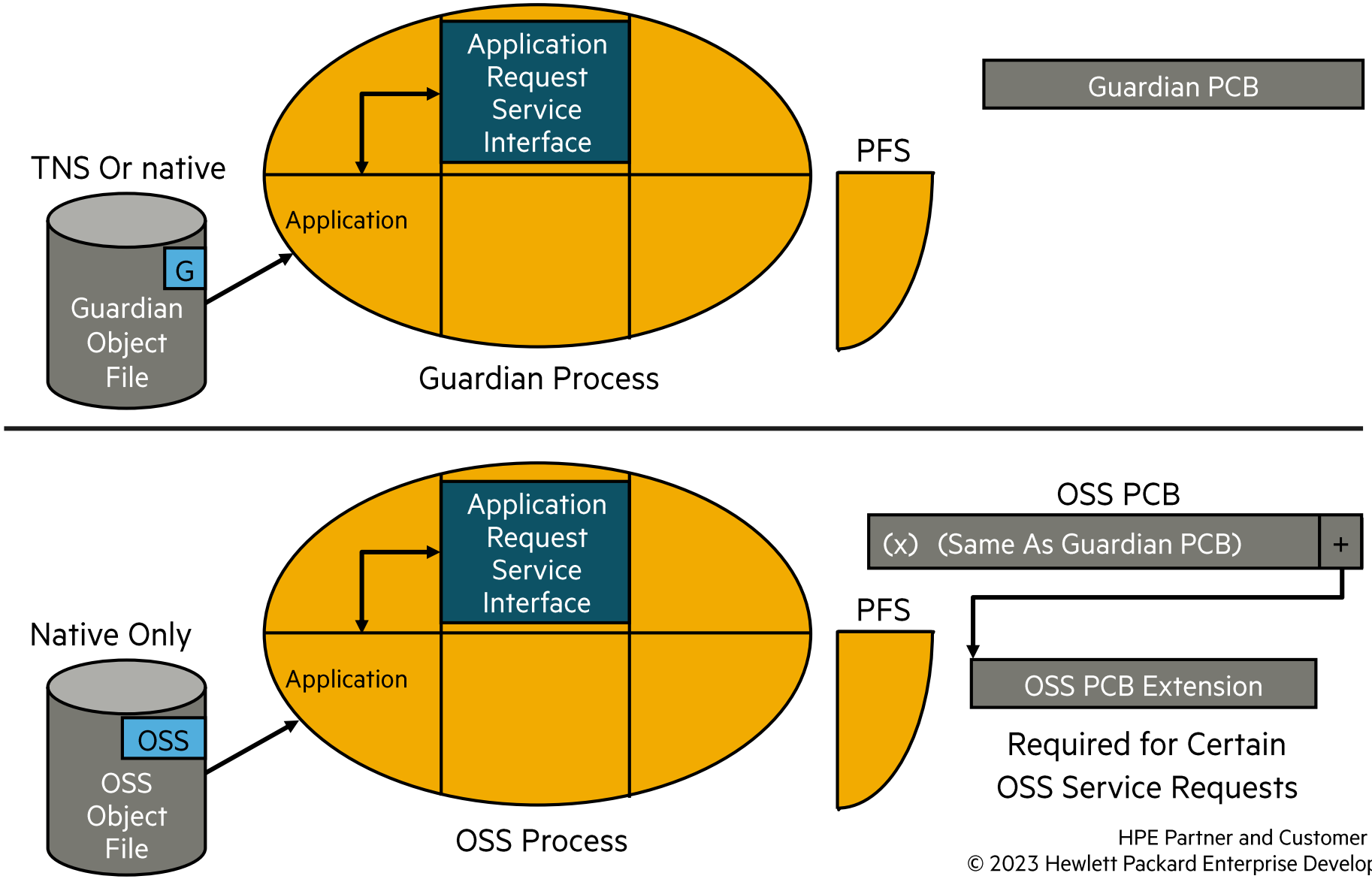
CIP Software Architecture Overview (IP CLIM (XPF version))



Part 4

- Debugging
- Run-Time architecture
  - TNS Prtocesses
  - TNS/X processes
- Process control
  - Guardian
  - OSS

Guardian Versus OSS Processes

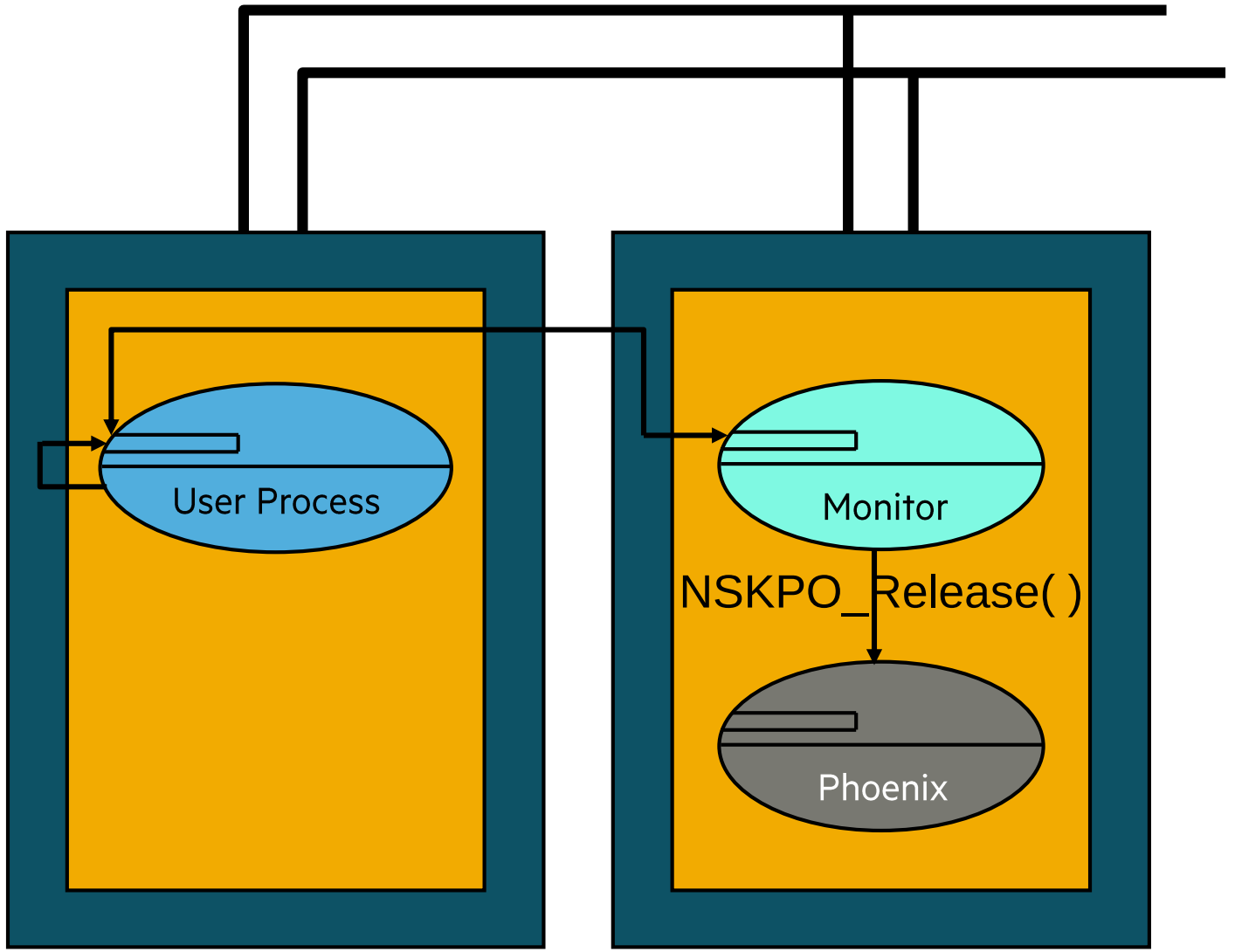


Guardian Process Control

Process Creation

- Guardian environment:
  - NEWPROCESS
    - Uses PID for the process identifier.
    - PID cannot identify process numbers > 255.
  - PROCESS_CREATE_
    - Uses PHANDLE (process handle) to identify processes.
  - PROCESS_LAUNCH_
    - Can override native process defaults.
    - Uses an extensible struct to pass parameters.
  - From a TACL prompt, the RUN command uses PROCESS_LAUNCH_.

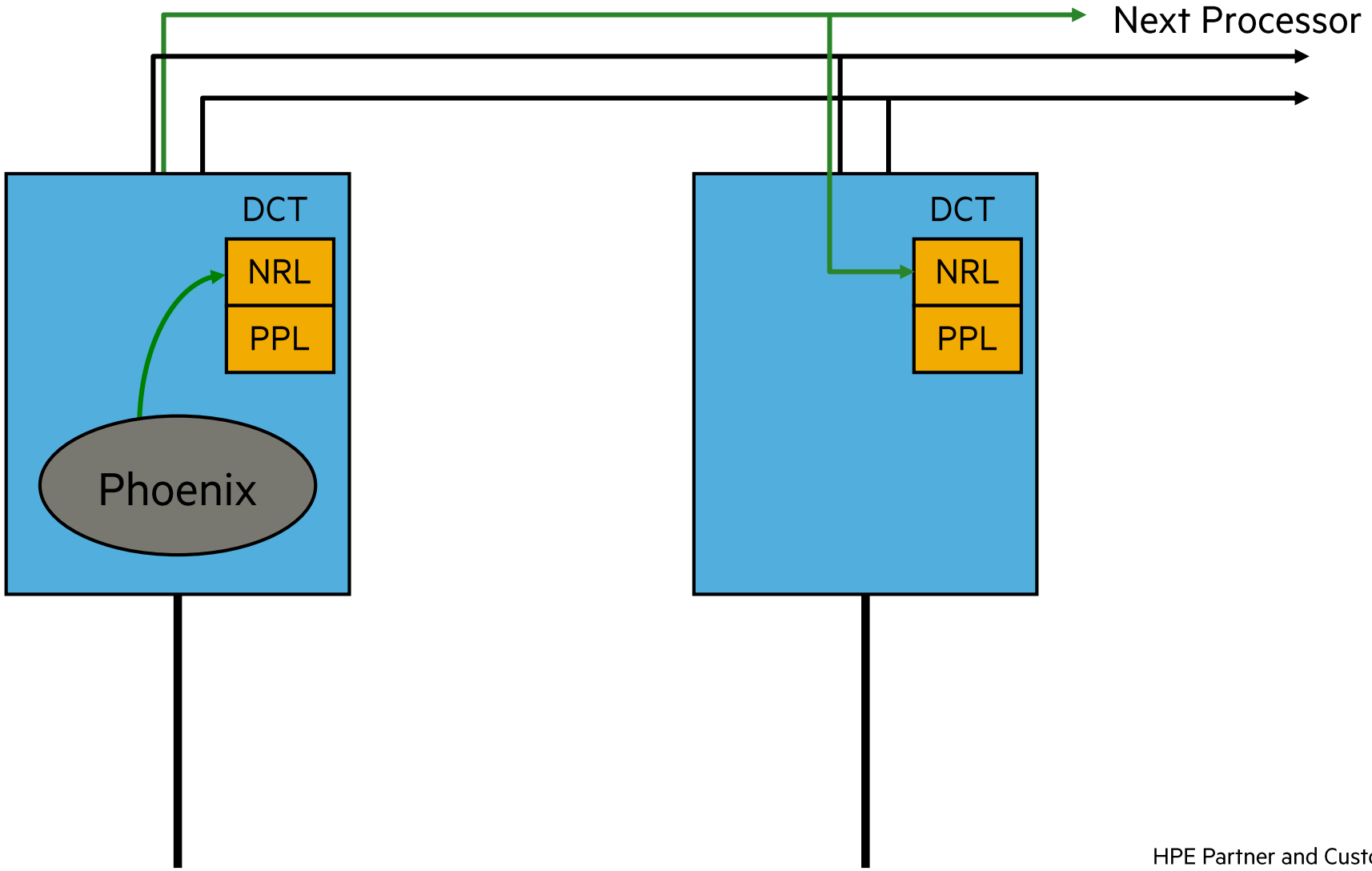
Process Creation — Guardian Environment



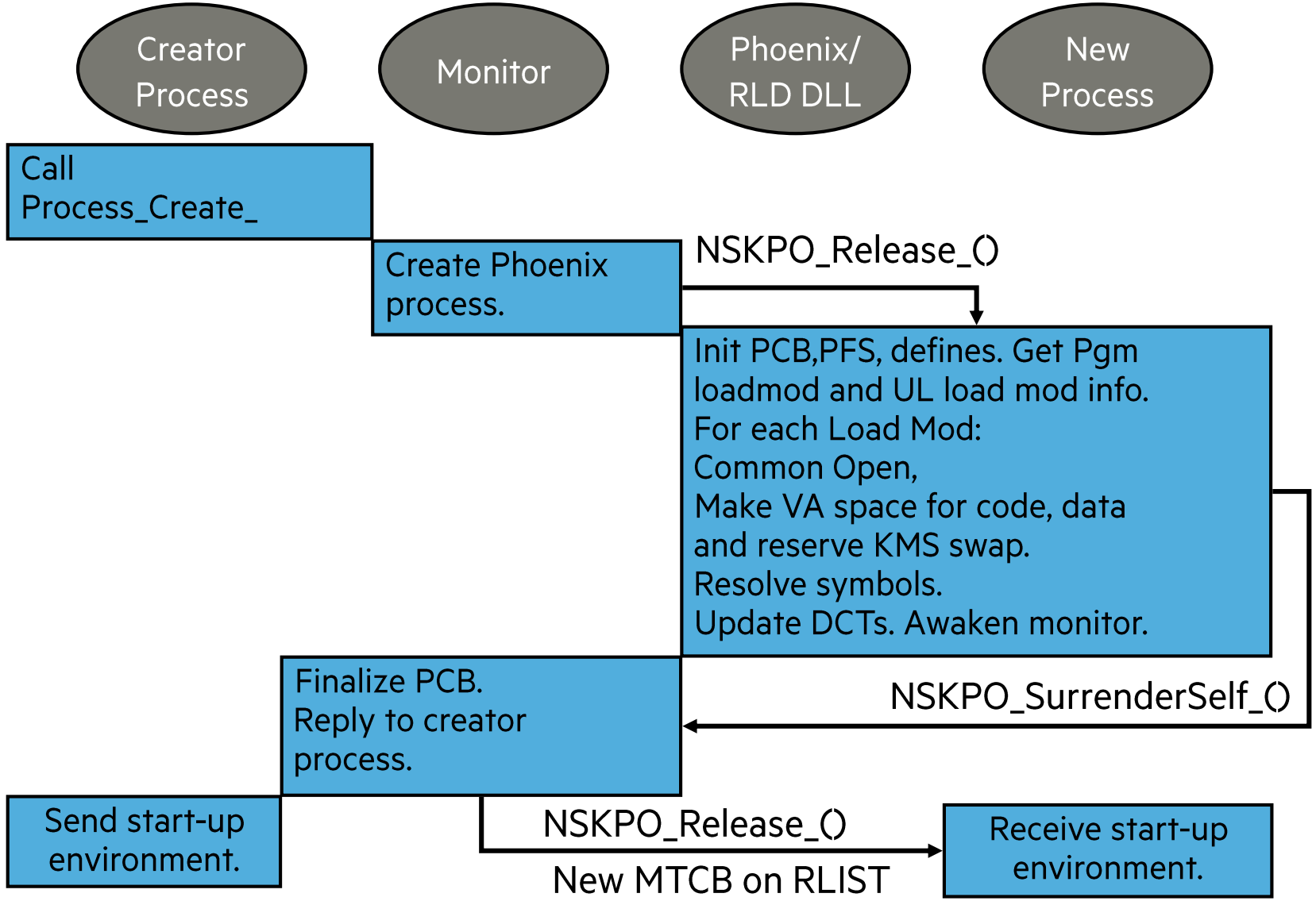
RLD Symbol Resolution

- For each load module:
  - Determines what symbols each load module defines.
  - Determines what symbols each load module needs resolved.
  - Resolves each symbol from the loadlist.
  - Fills in the GOT and PLT (for IPF) tables with the values of the symbols.

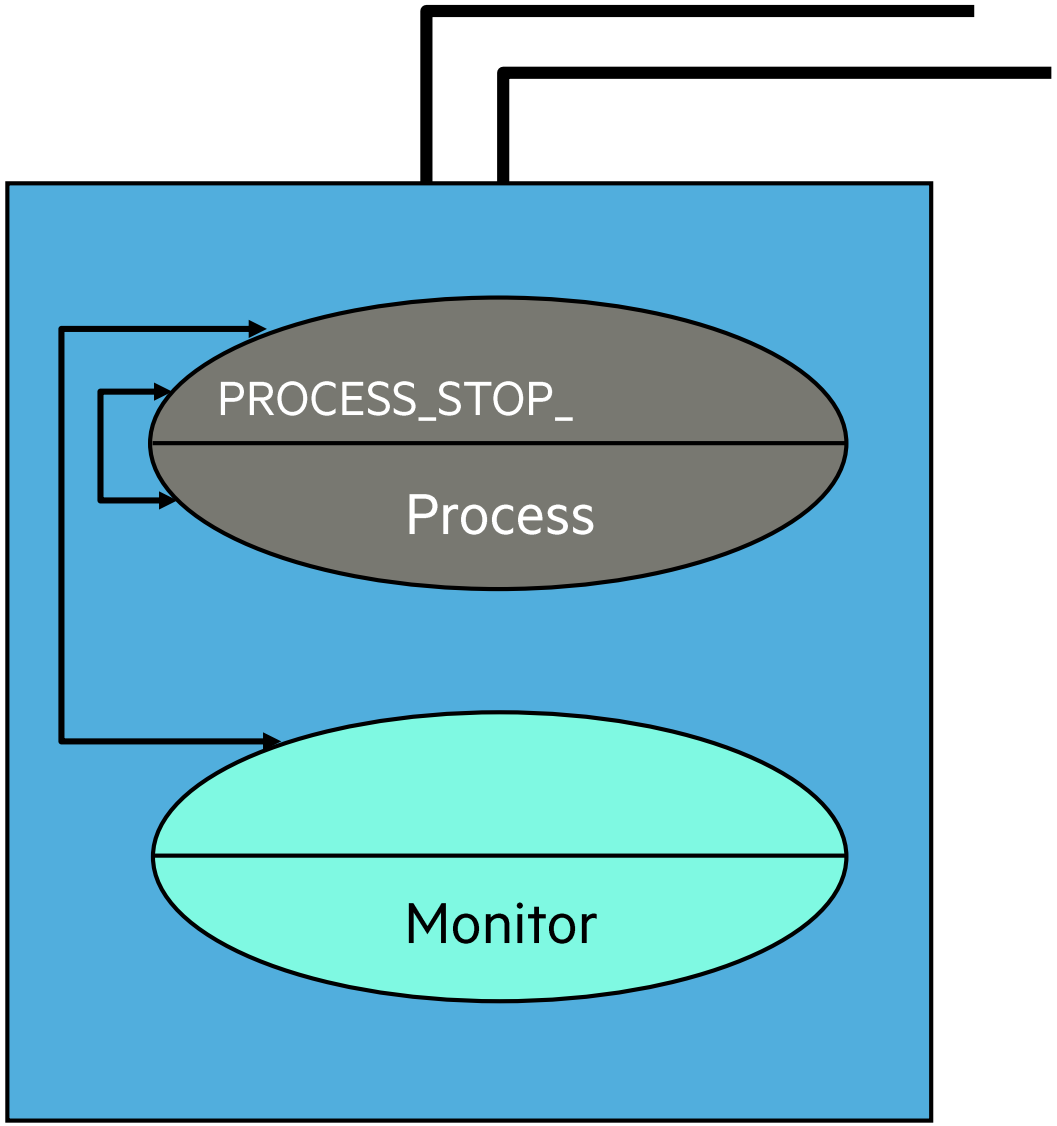
Phoenix Handling of Named Processes



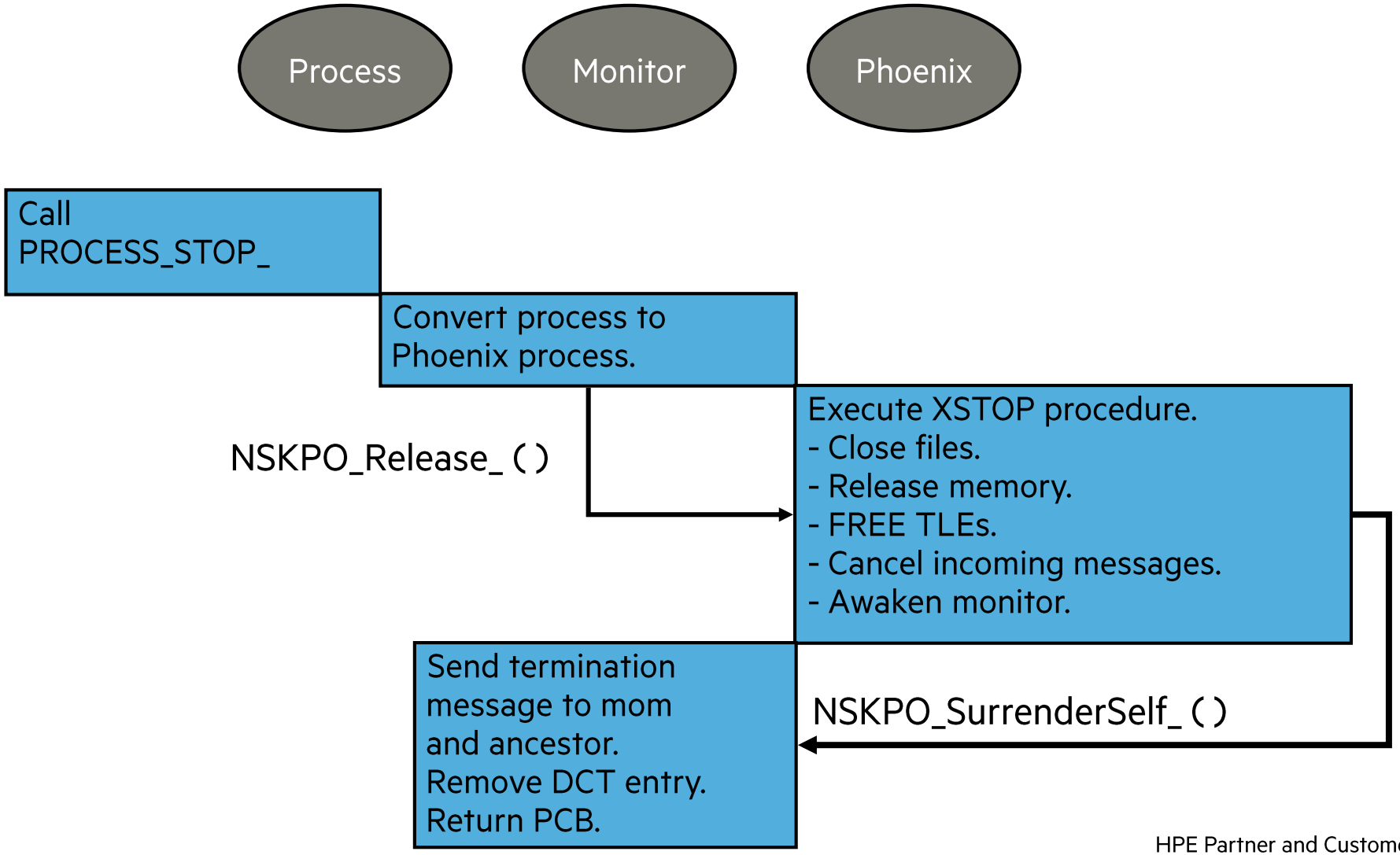
Guardian Process Creation — Summary



Process Termination —PROCESS_STOP_ Procedure



Guardian Termination — Summary



Monitor Functions

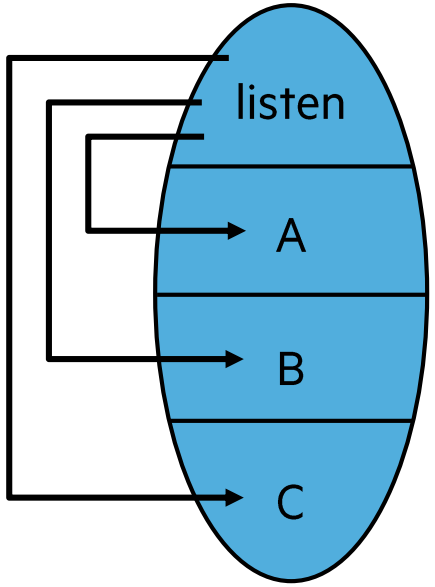
- Starts and stops processes.
  - STM and others give Monitor the process to terminate after fatal traps.
  - Dynamic system configuration.
- Performs process control.
- Returns information.
- Maintains the system time-of-day.

OSS Process Control

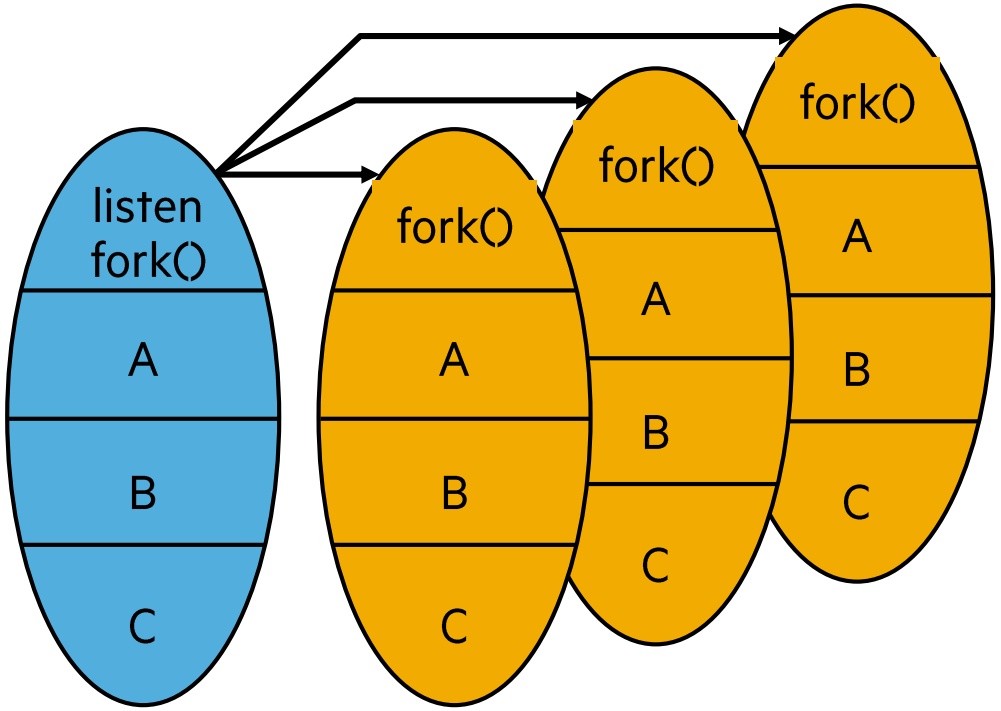
OSS Process Creation

- OSS environment.
  - fork().
  - exec*().
  - Shell run.
  - tdm_spawn().
  - tdm_*() takes additional parameters similar to PROCESS_LAUNCH_.
  - PROCESS_SPAWN_ - can be used from a Guardian process.
    - It is used by the OSH program.

Process Creation — OSS

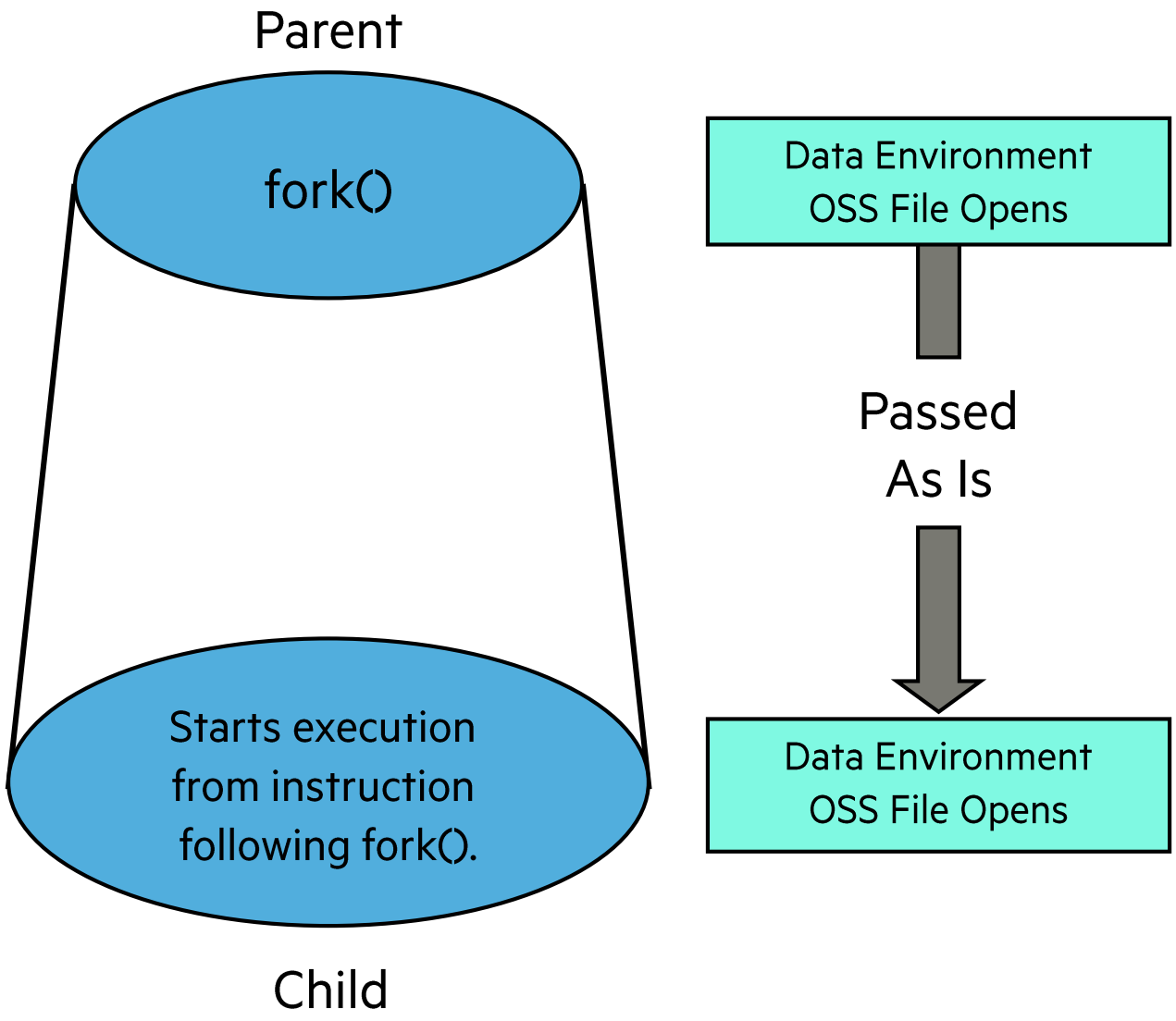


1 — Single, monolithic program, multithreaded, difficult to write and to test.

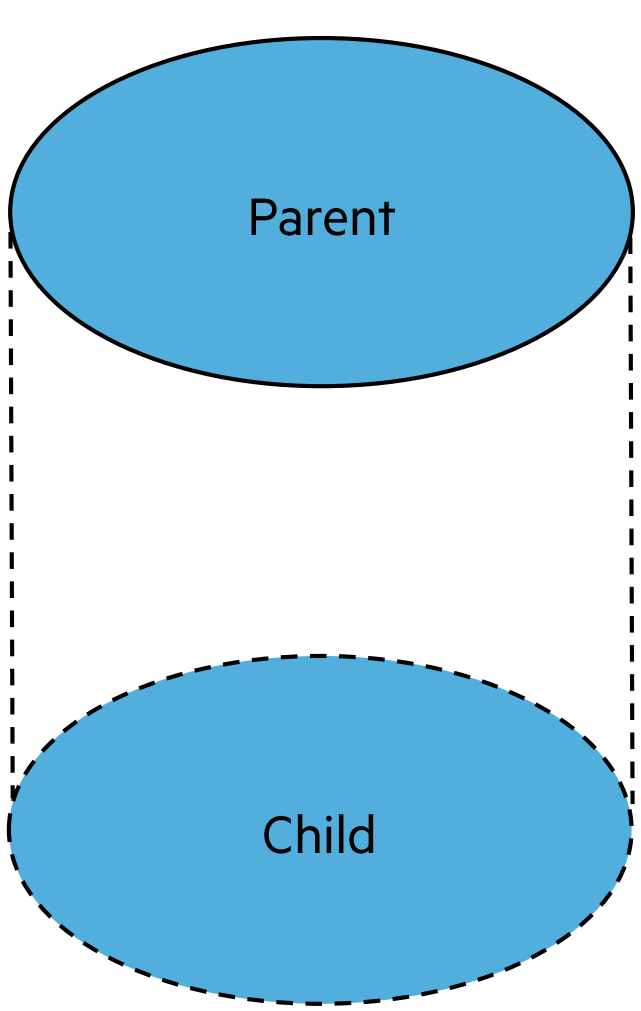


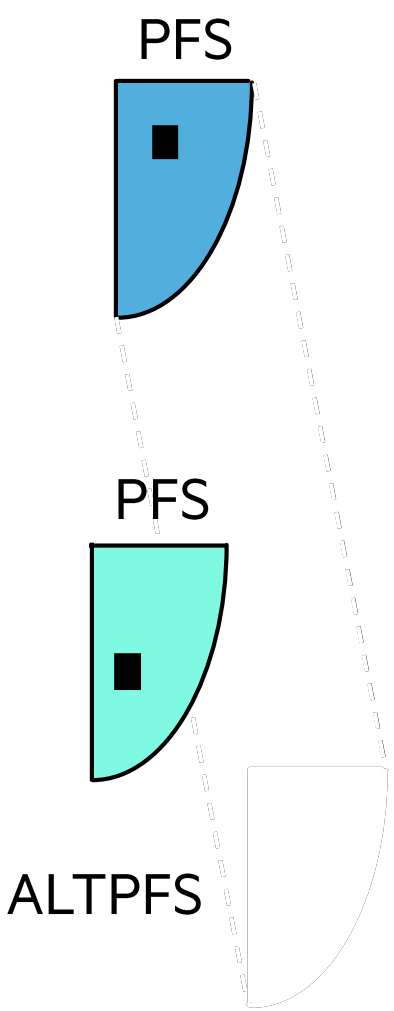
2 — One program forks itself, keeping its opens, and each forked copy shares the code. Simpler to write and test because not multithreaded.

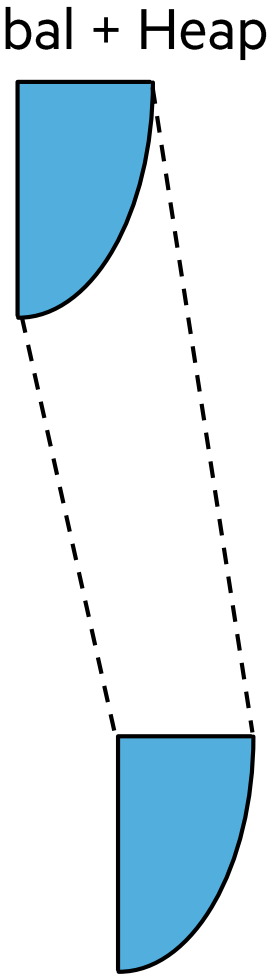
Process Creation — OSS fork()



The Copy Environments

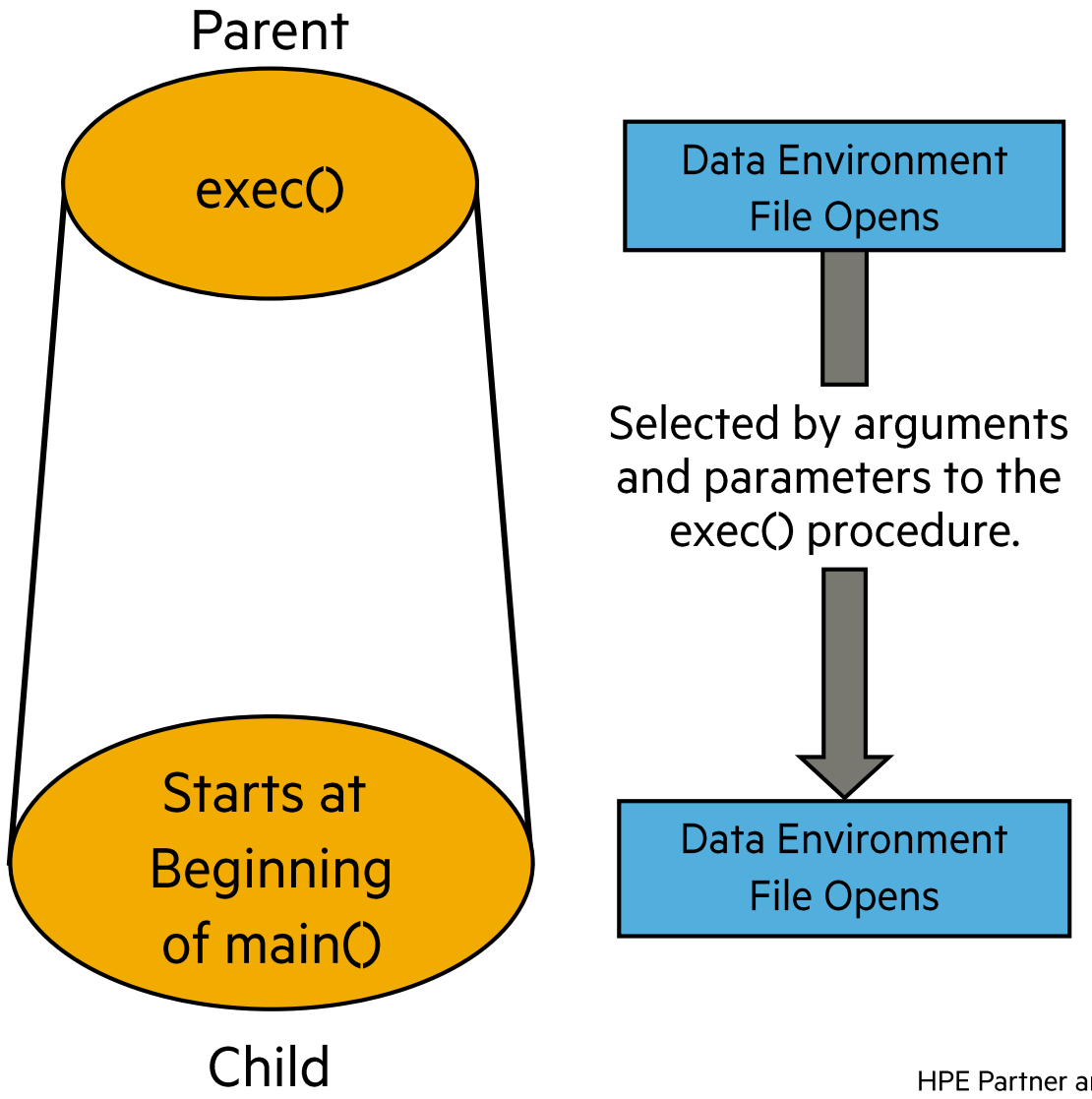




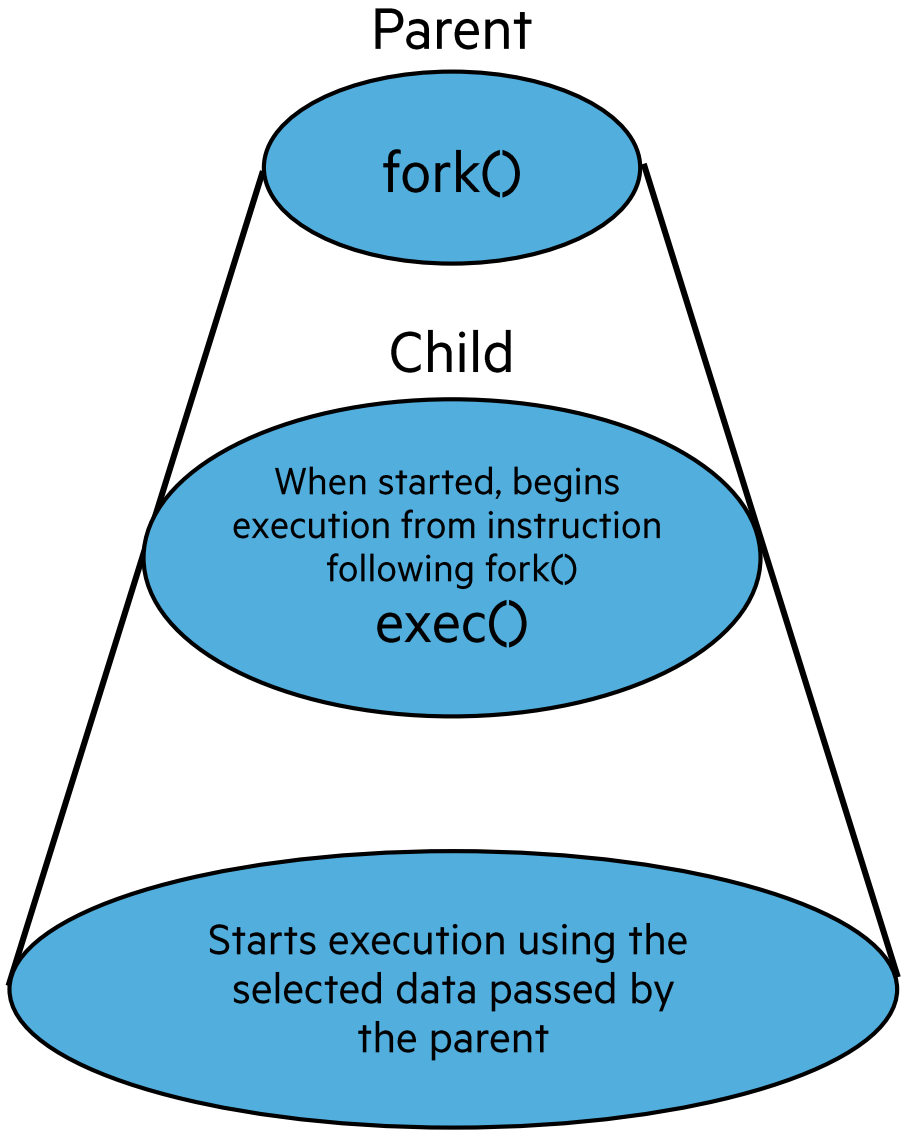


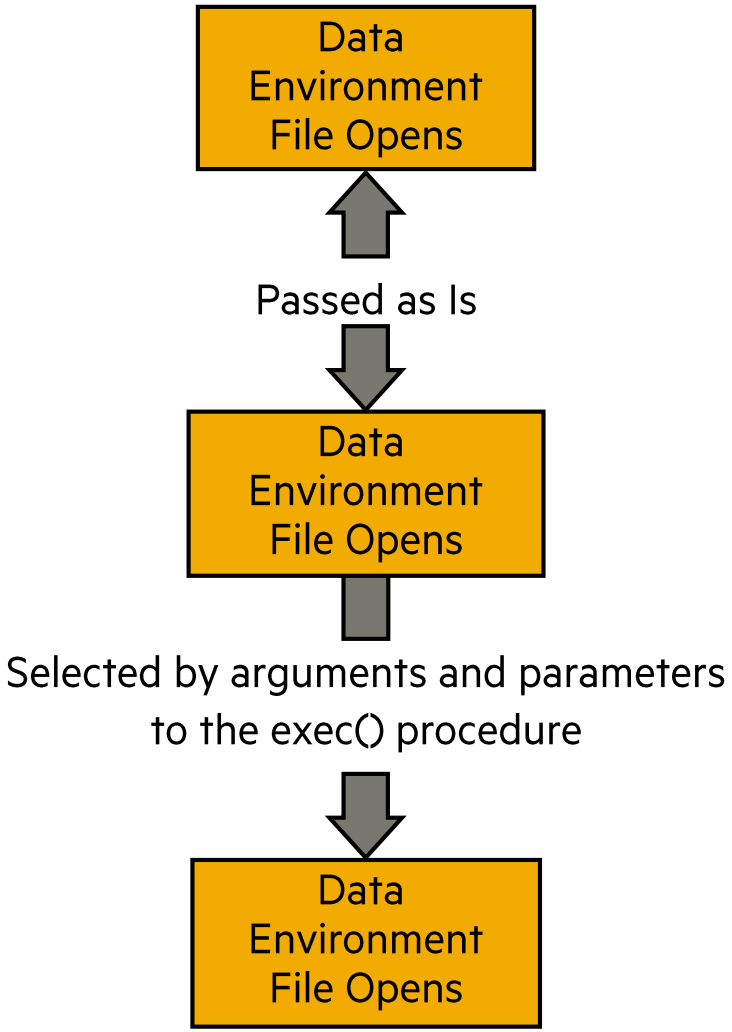
Process Creation — OSS exec()

Following the start of the child process, the parent goes away.

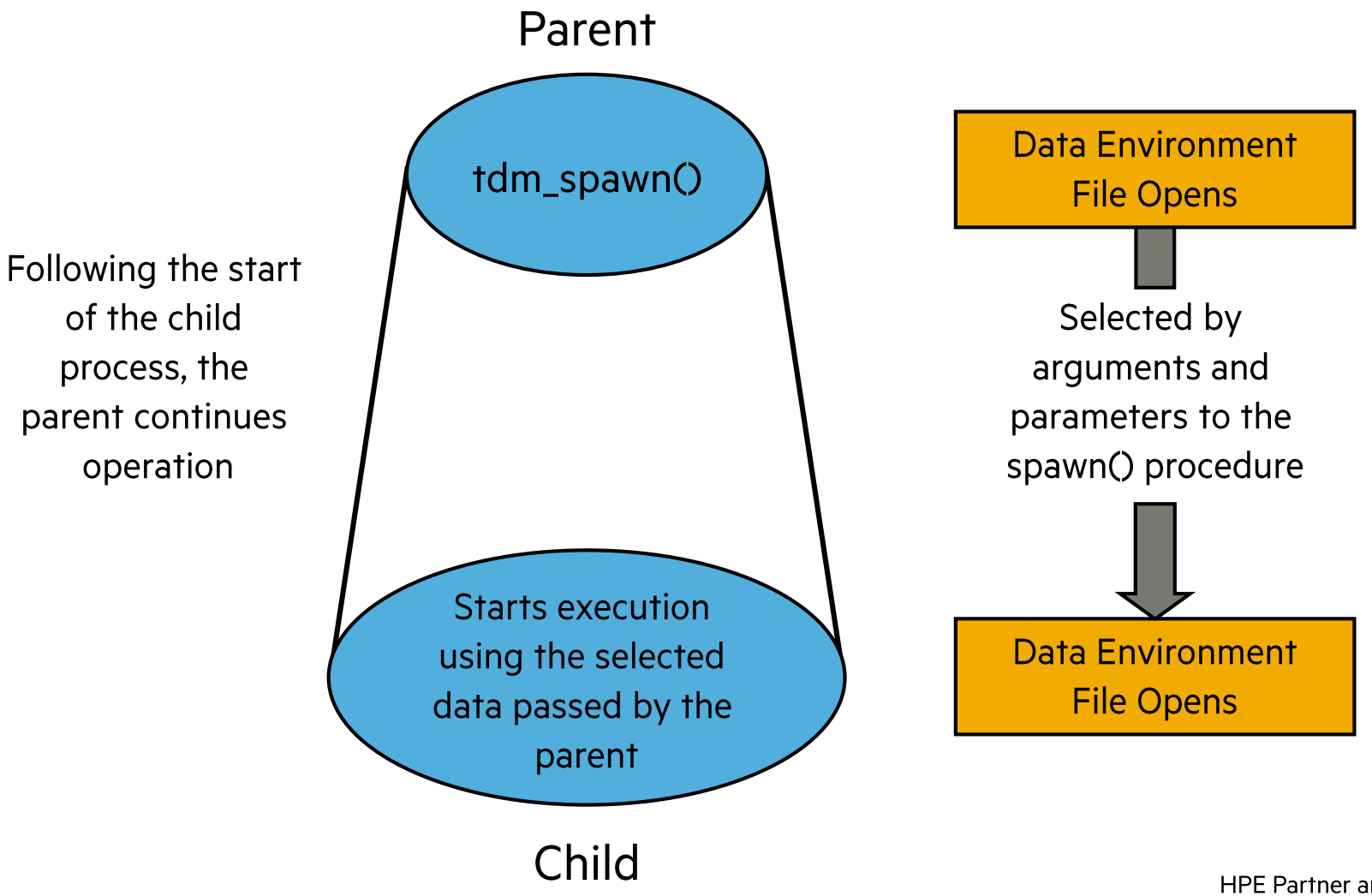


Process Creation — OSS fork(), exec()



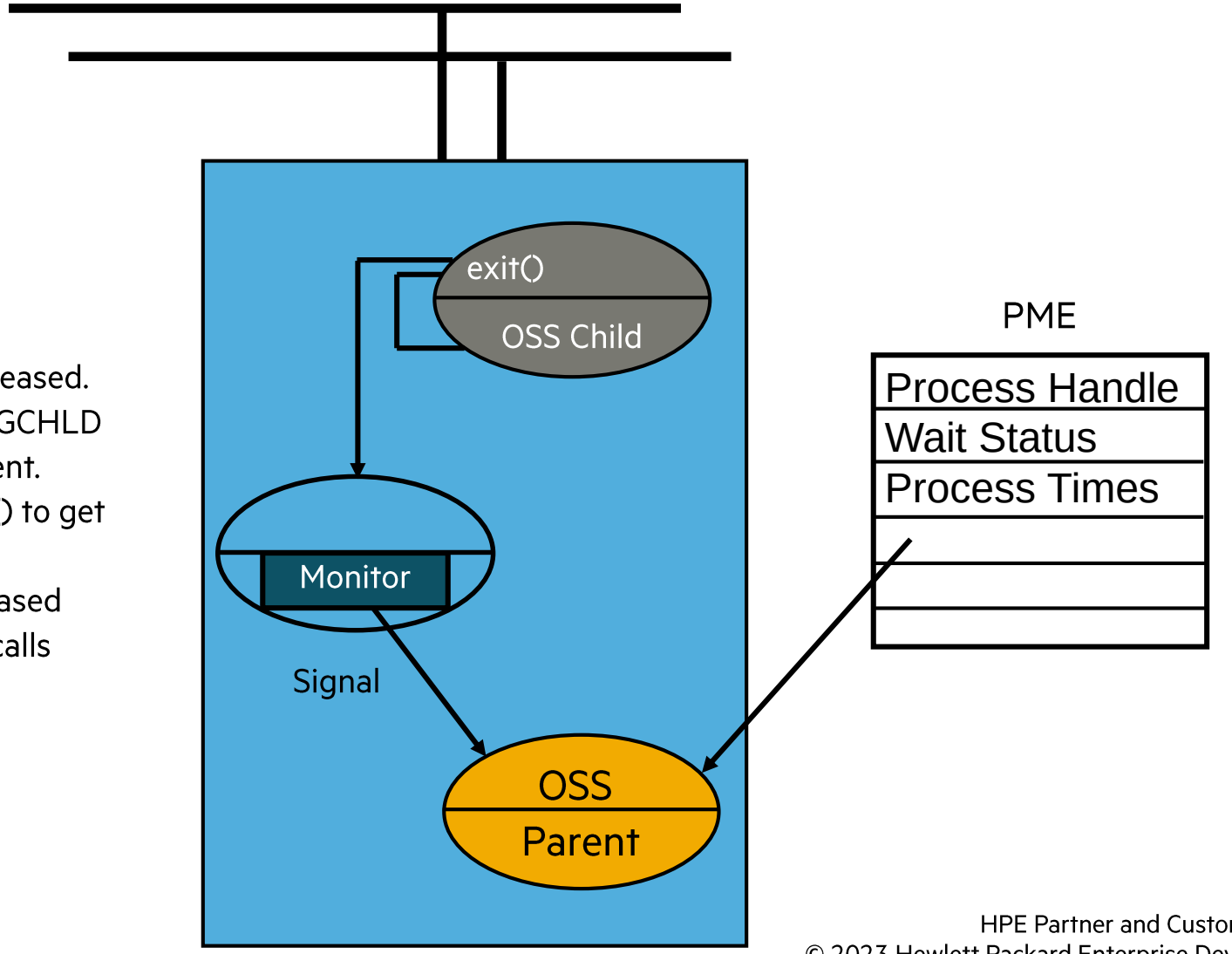


Process Creation — OSS tdm_spawn()



OSS Process Termination

1. Child calls exit().
2. Resources are released.
3. Monitor sends SIGCHLD signal to the parent.
4. Parent calls wait() to get the status.
5. Child PME is released after the parent calls wait().



Debuggers

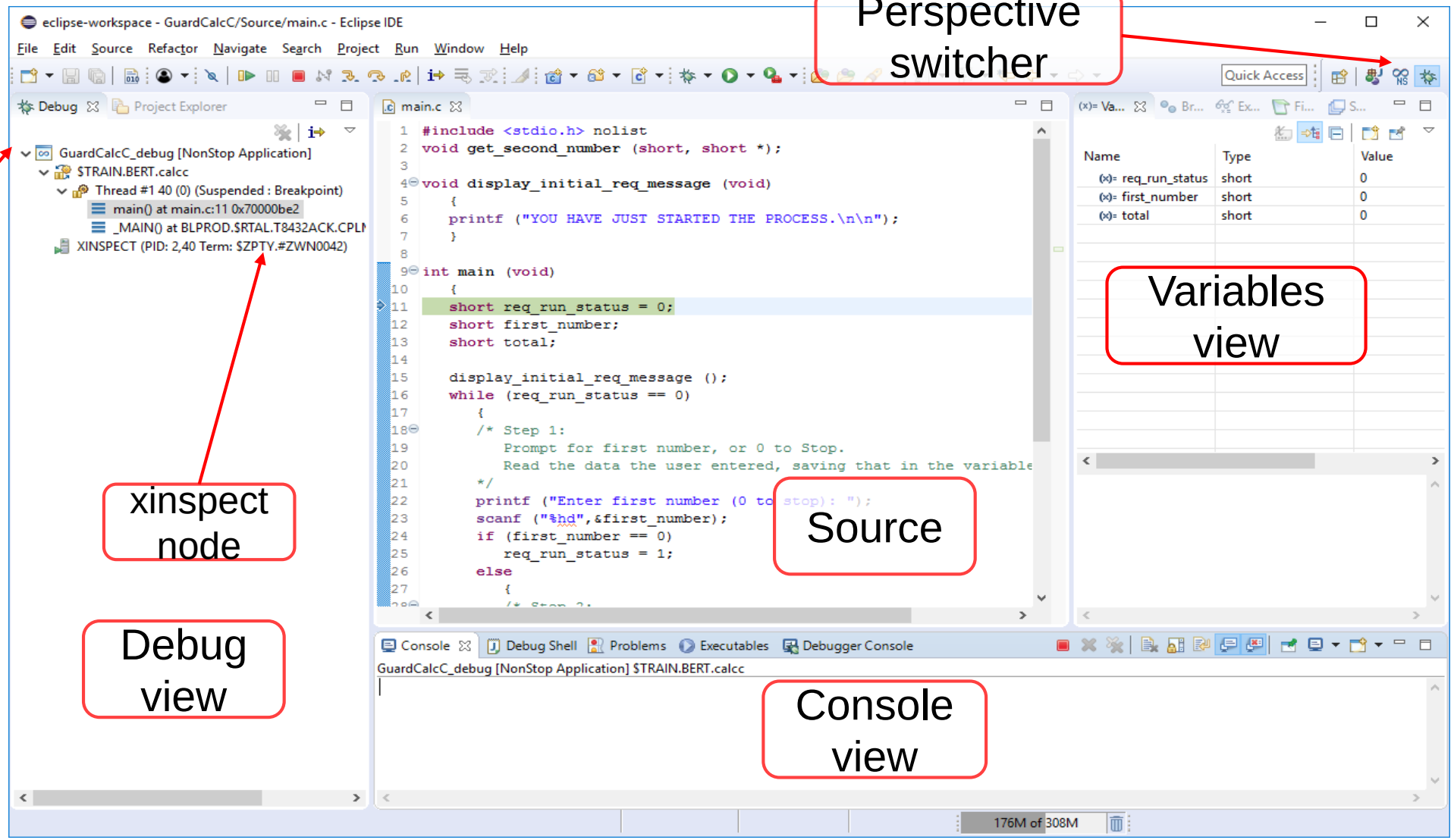
What are the L-Series Debuggers? (1 of 2)

- xGarth:
  - The system-level debugging tool for remote system debugging, processor dump analysis, halted processor analysis, process snapshot file analysis.
    - TNS/X Garth is available on the TNS/X platform for dump analysis.
  - Gnu gdb
  - Not further described in this course
- xInspect (Native Inspect):
  - xGarth but running from the TNS/X command line.

What are the L-Series Debuggers? (2 of 2)

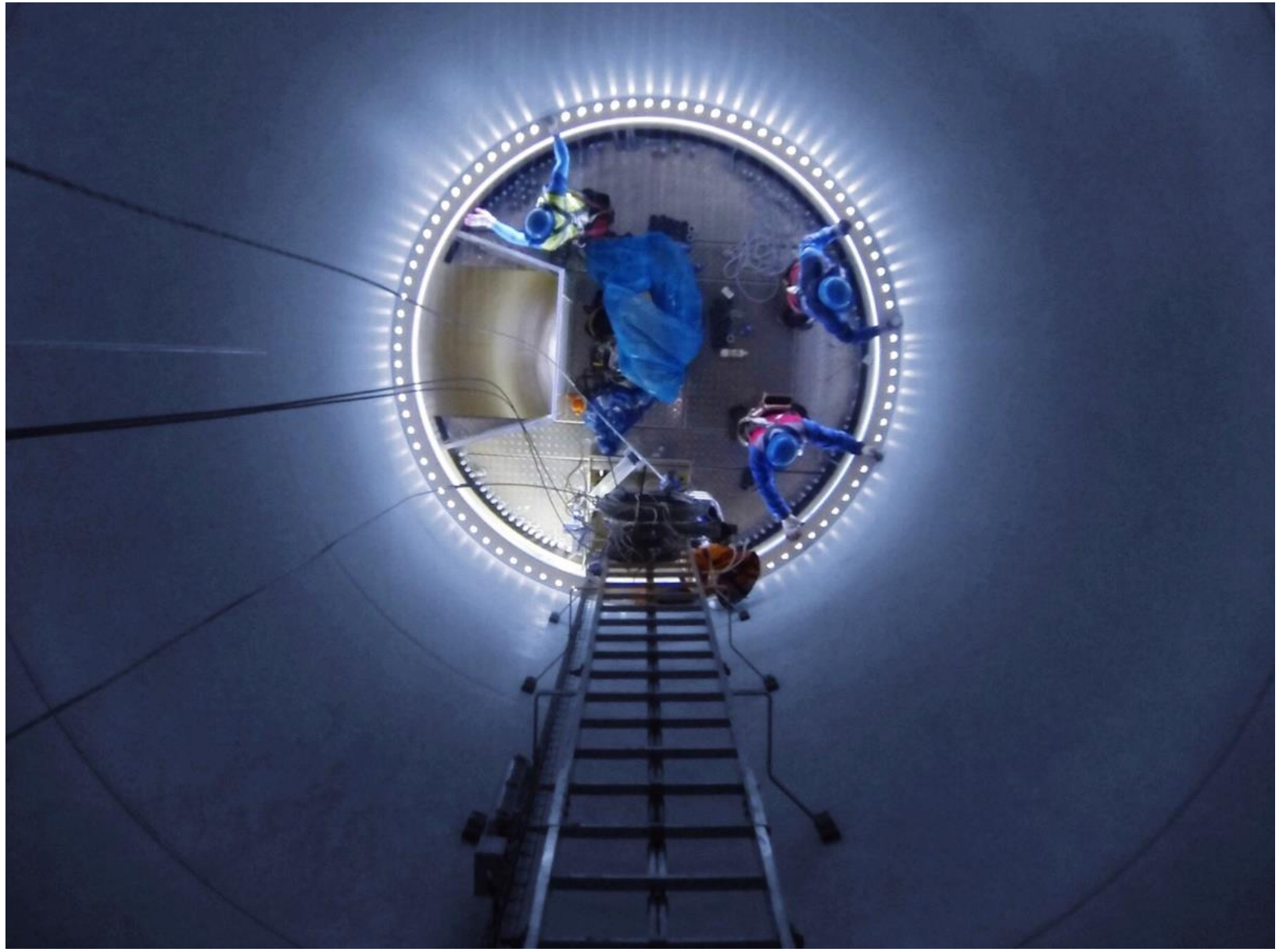
- NSDEE (Eclipse) version 13:
  - Supports Native TNS/E and TNS/X debugging (no TNS)
  - Part of Eclipse development environment
  - PC hosted GUI
- Inspect:
  - TNS debugging only.
- TNS Visual Debugger (TNSVDBG):
  - TNS debugging only
  - Basically, Visual Inspect but restricted to TNS only debugging on TNS/X systems

Debug Perspective



Part 5

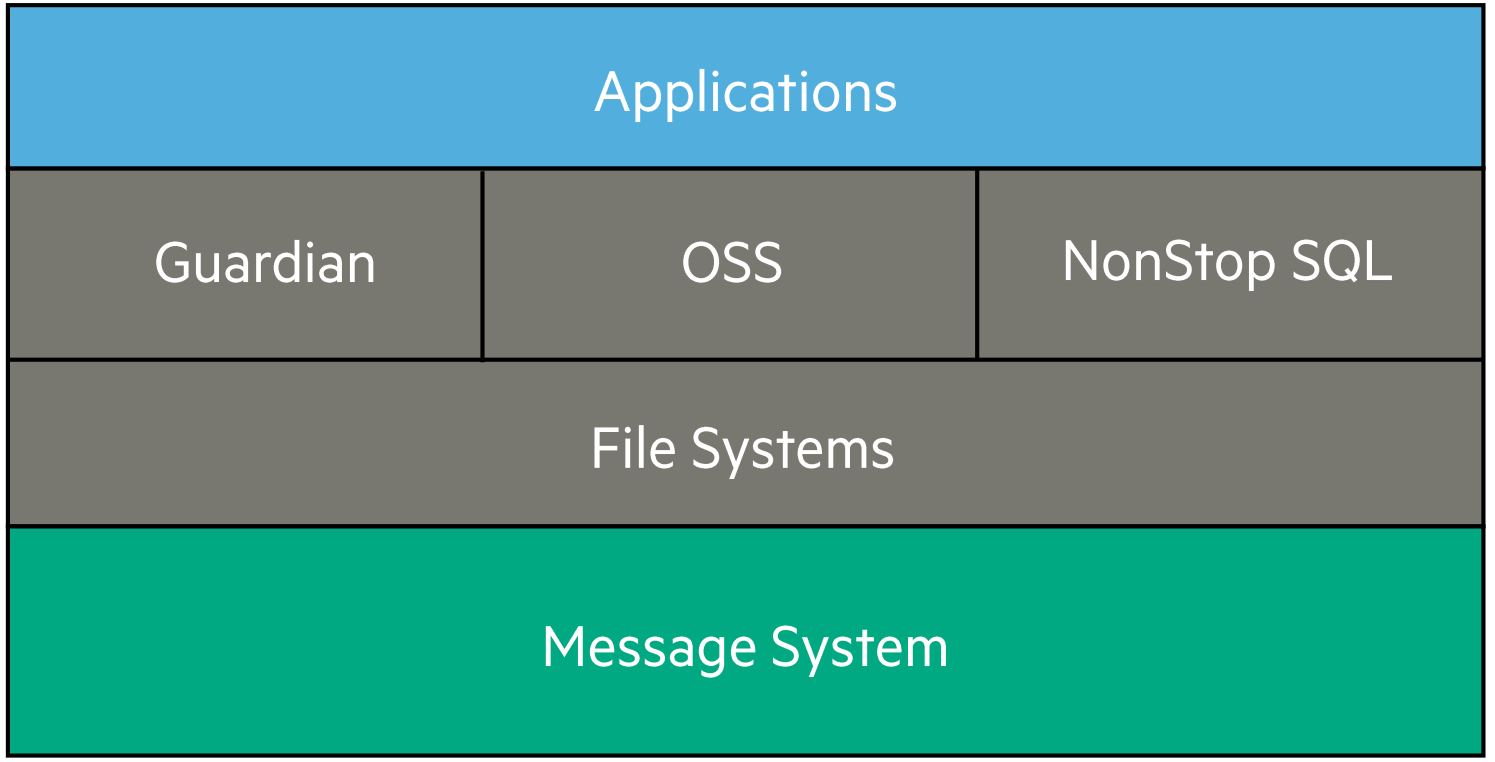
- Guardian file system
- Open System Services file system



Functions of the File System

- Callable interface to message system.
- Logical file name to process handle (phandle) resolution.
- Passes security information to the IOP or file-system layer of the application server.
- Device independence: Everything is treated as a file.
- Keeps track of outstanding messages and associated buffers.
- Some fault tolerance support.
- (For disk) Partition file and alternate-key support.
- What the file system does not do: IOPs and TMF software, though not part of the file system, provide functions that are often associated with a file system.

File Systems

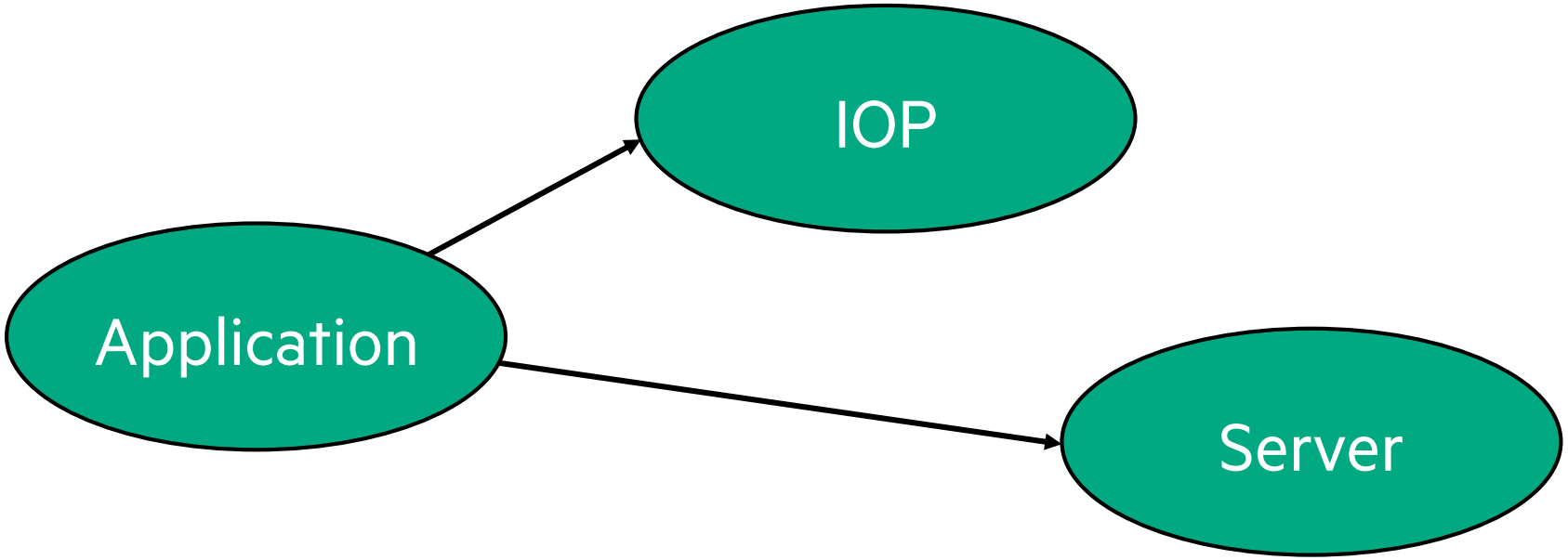


Control Structures for Guardian File System

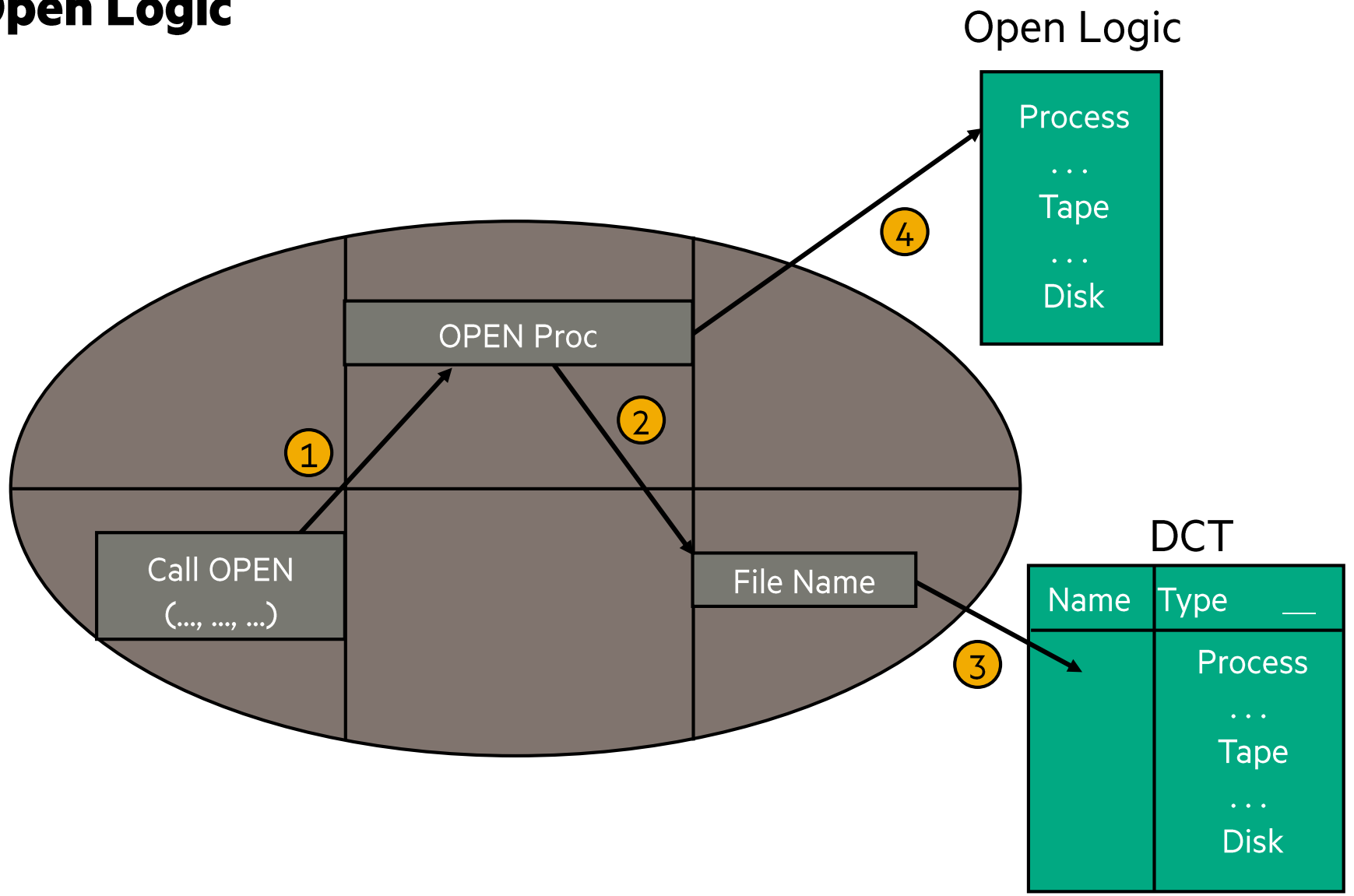
- Access control block (ACB) — Opener controlled.
- Destination control table (DCT) — Kernel-reserved segments.
- File table.
- IOP has open control block (OCB) corresponding to ACB.
- Disk process has a file control block (FCB) to represent each open disk file.

Open Request Processing Between Requester and Server

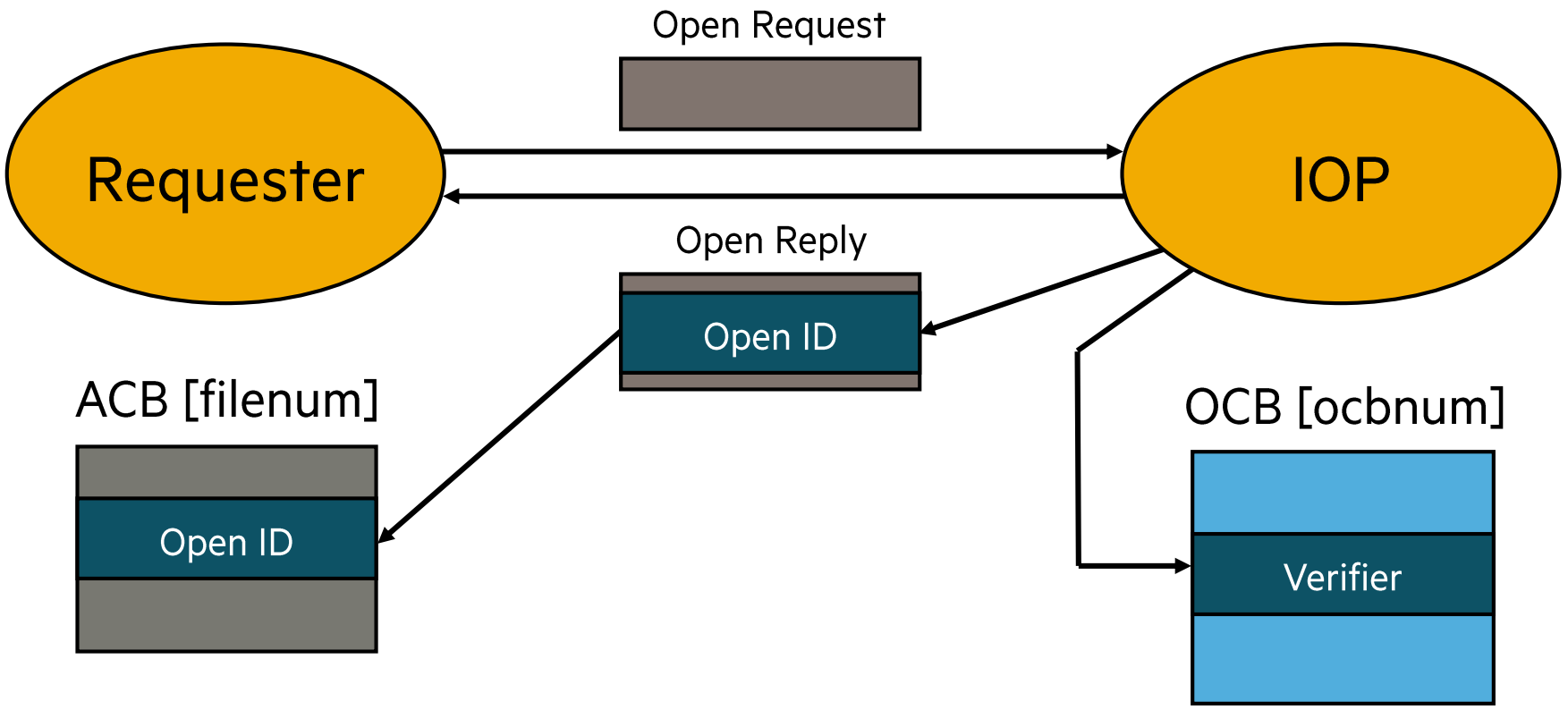
Application Might Send OPEN	Case 1 to IOP	Case 2 Or To Application



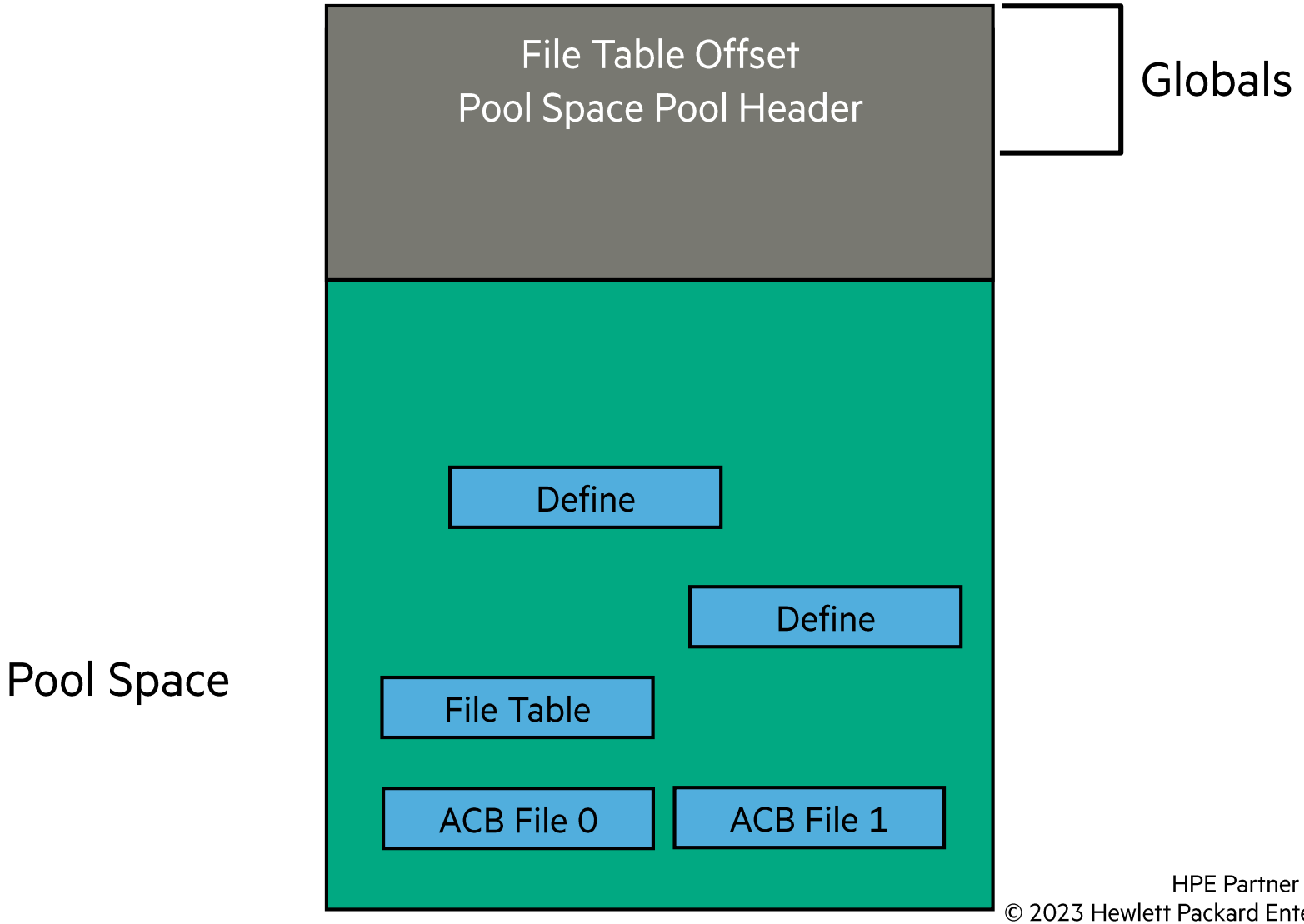
Requester Open Logic



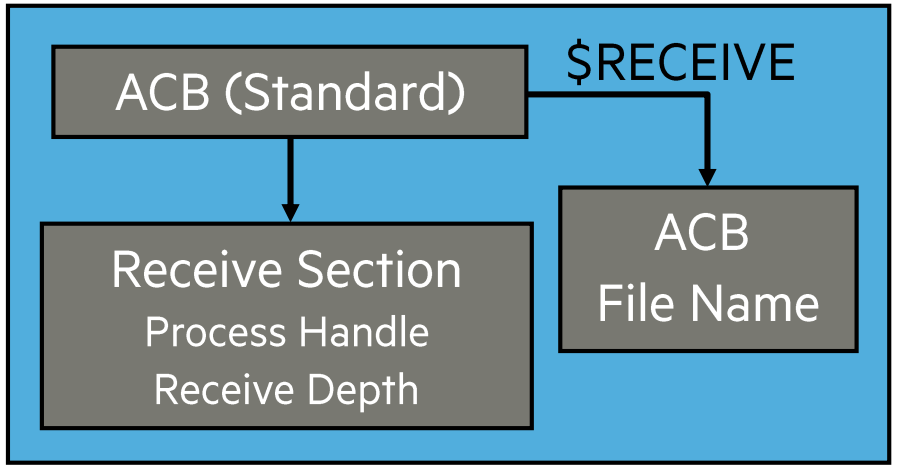
Open Request Processing By the IOP

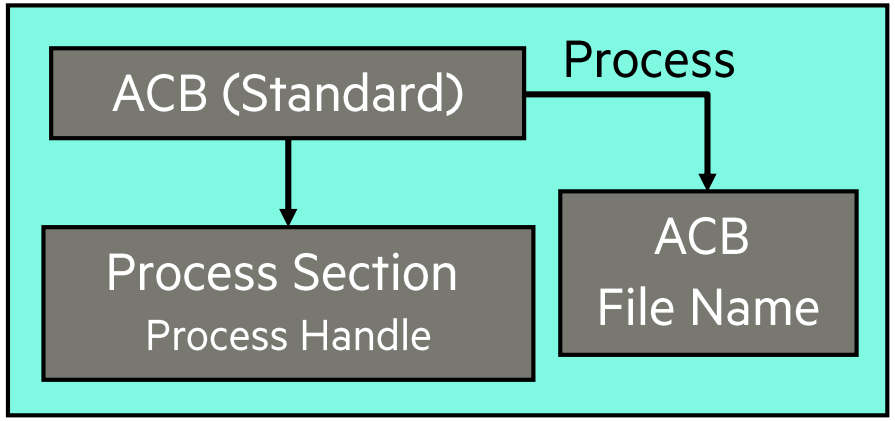


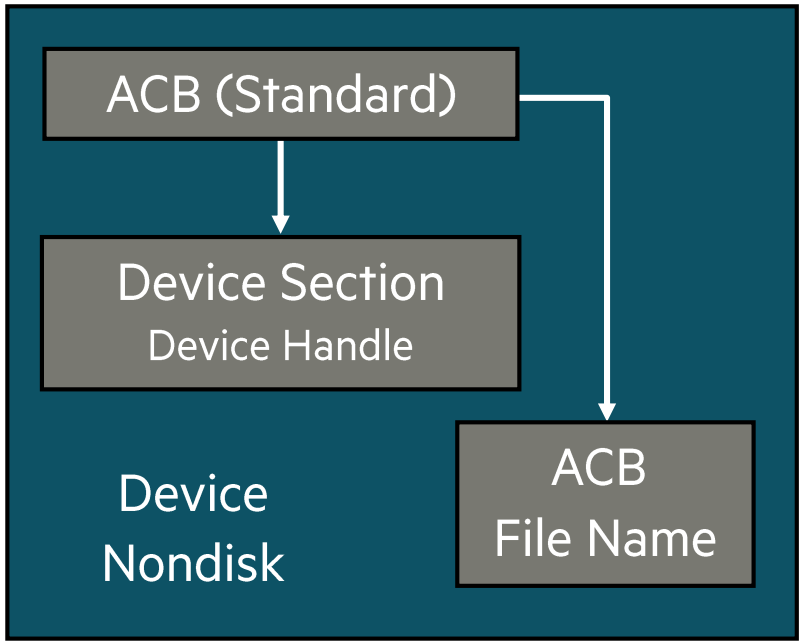
Process-File Segment (PFS)



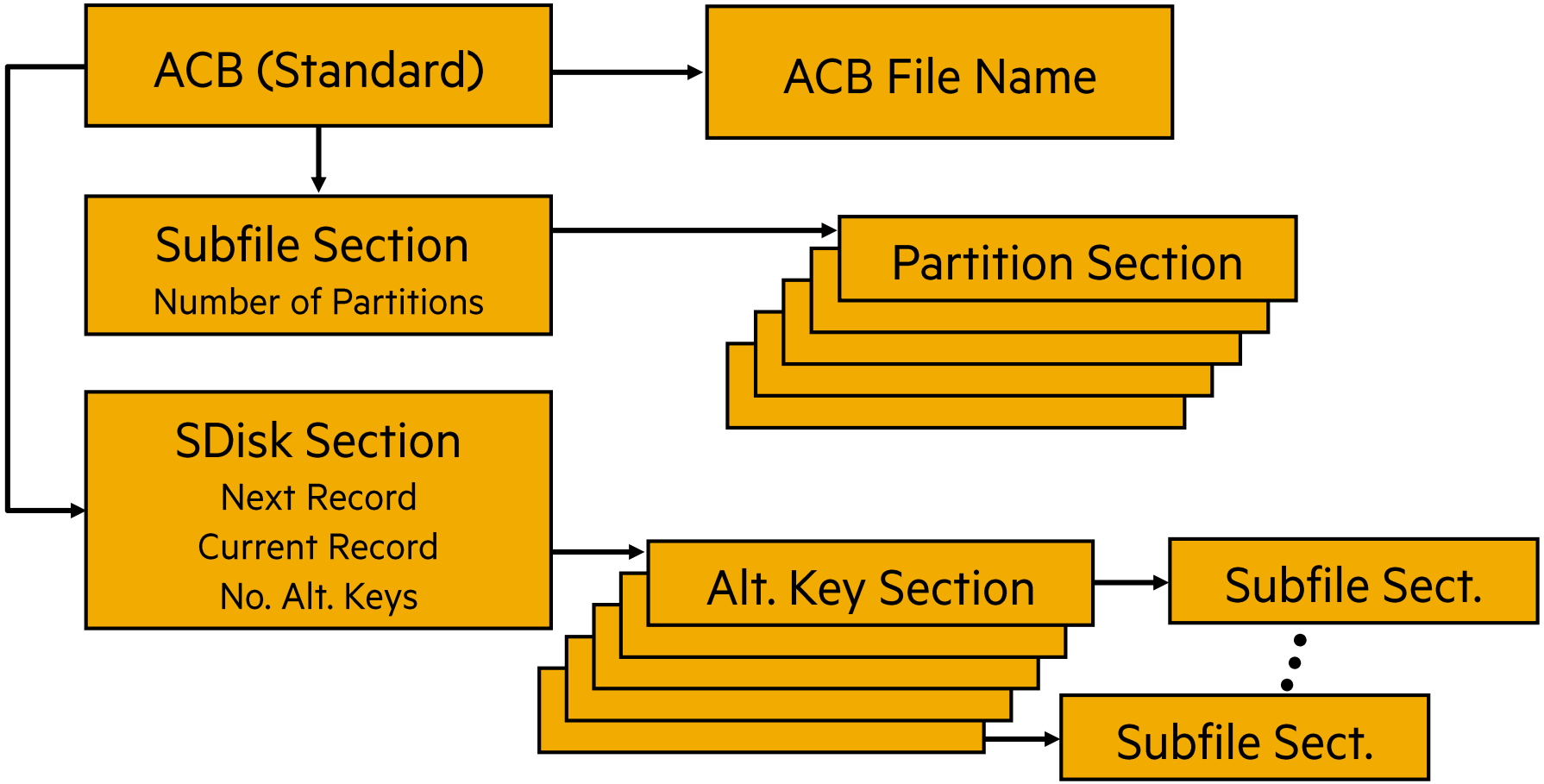
ACBs, Guardian File System







Structured Disk-File ACB, Guardian File System

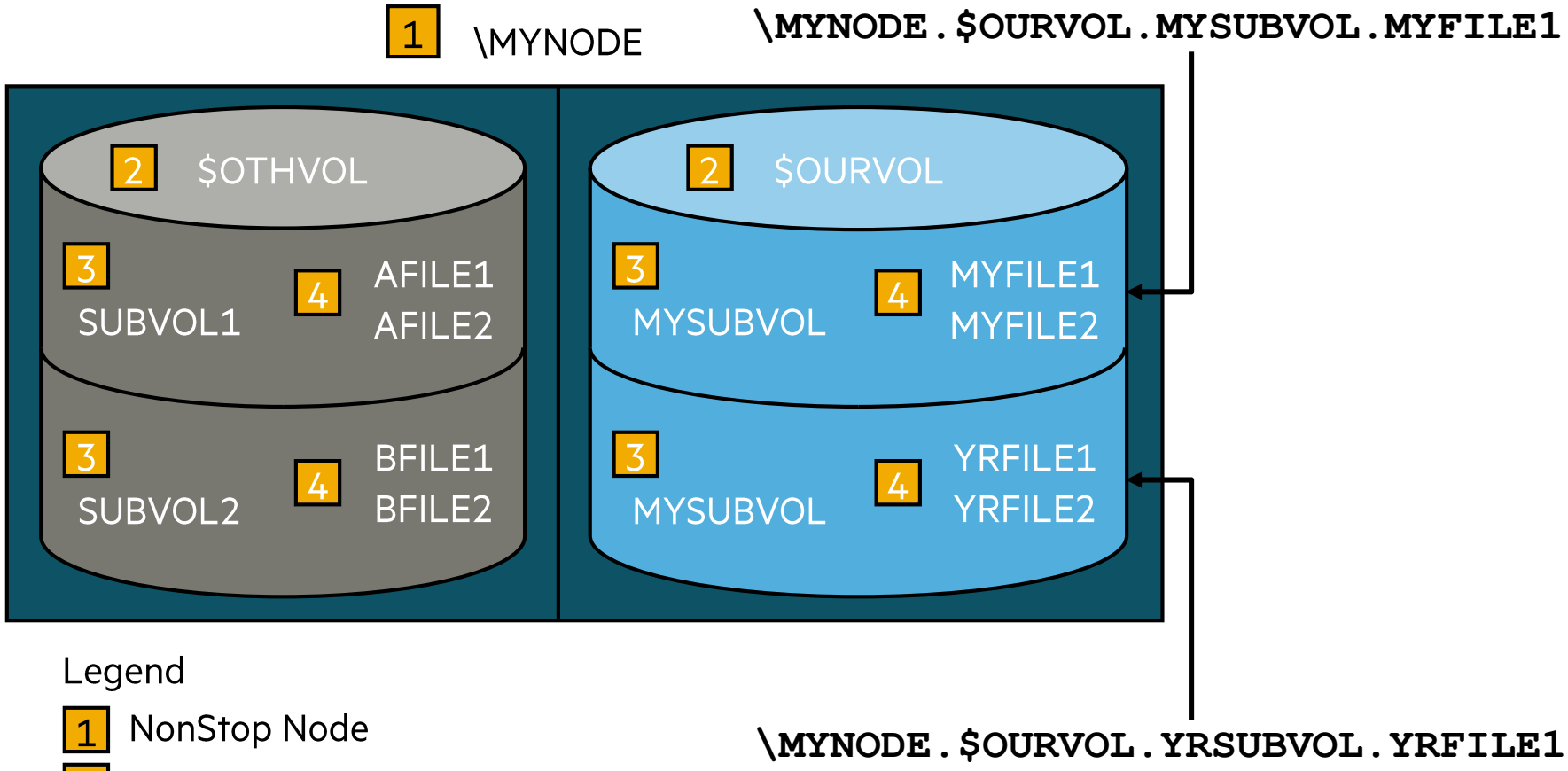


OSS File System

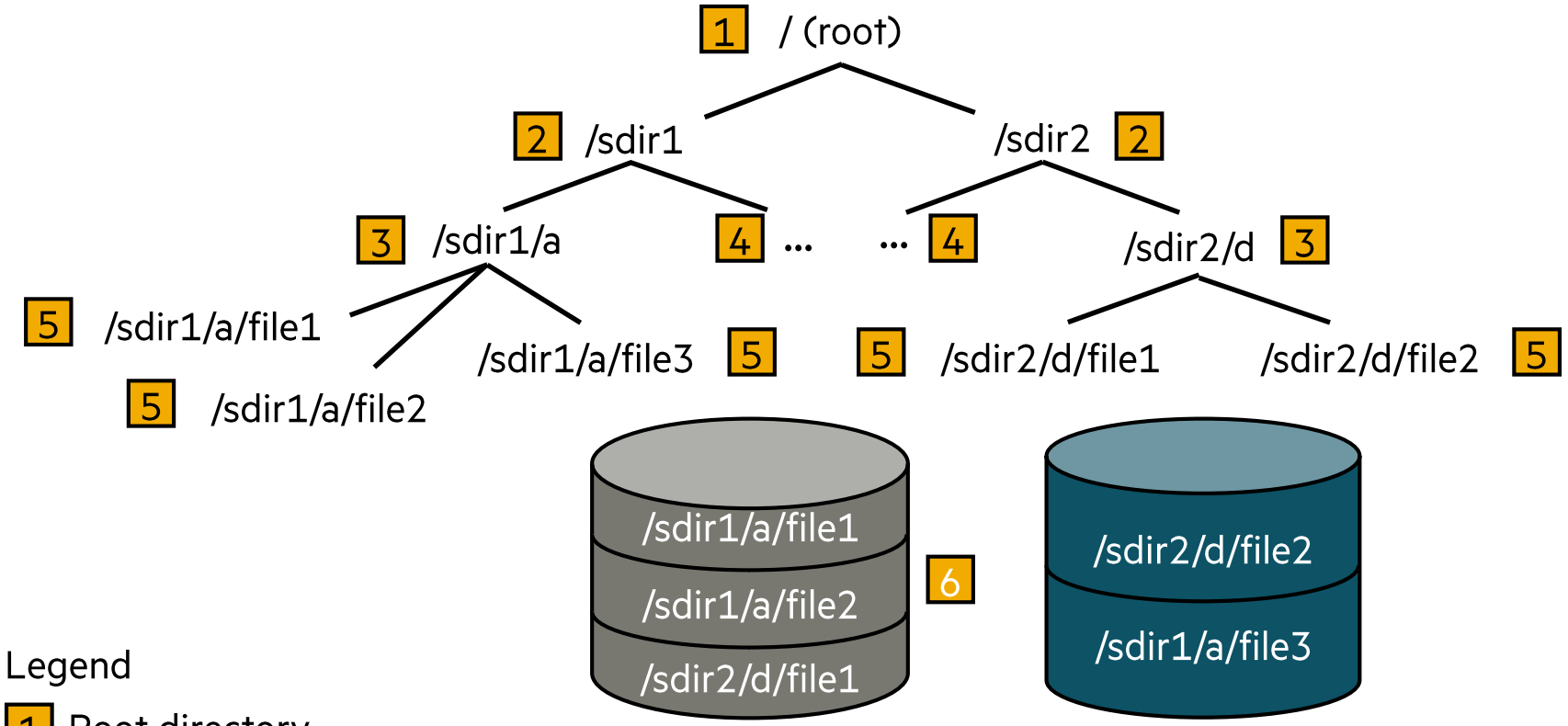
- OSS file system refers to an entire collection of files.
- File system includes a hierarchical structure of directories, subdirectories, and files.
- File system has a single root.

Guardian Disk File-Name Hierarchy Has Few Levels

- Guardian Disk File Organization



OSS File Organization



Legend

1 Root directory

2 Directories in the root directory (subdirectories)

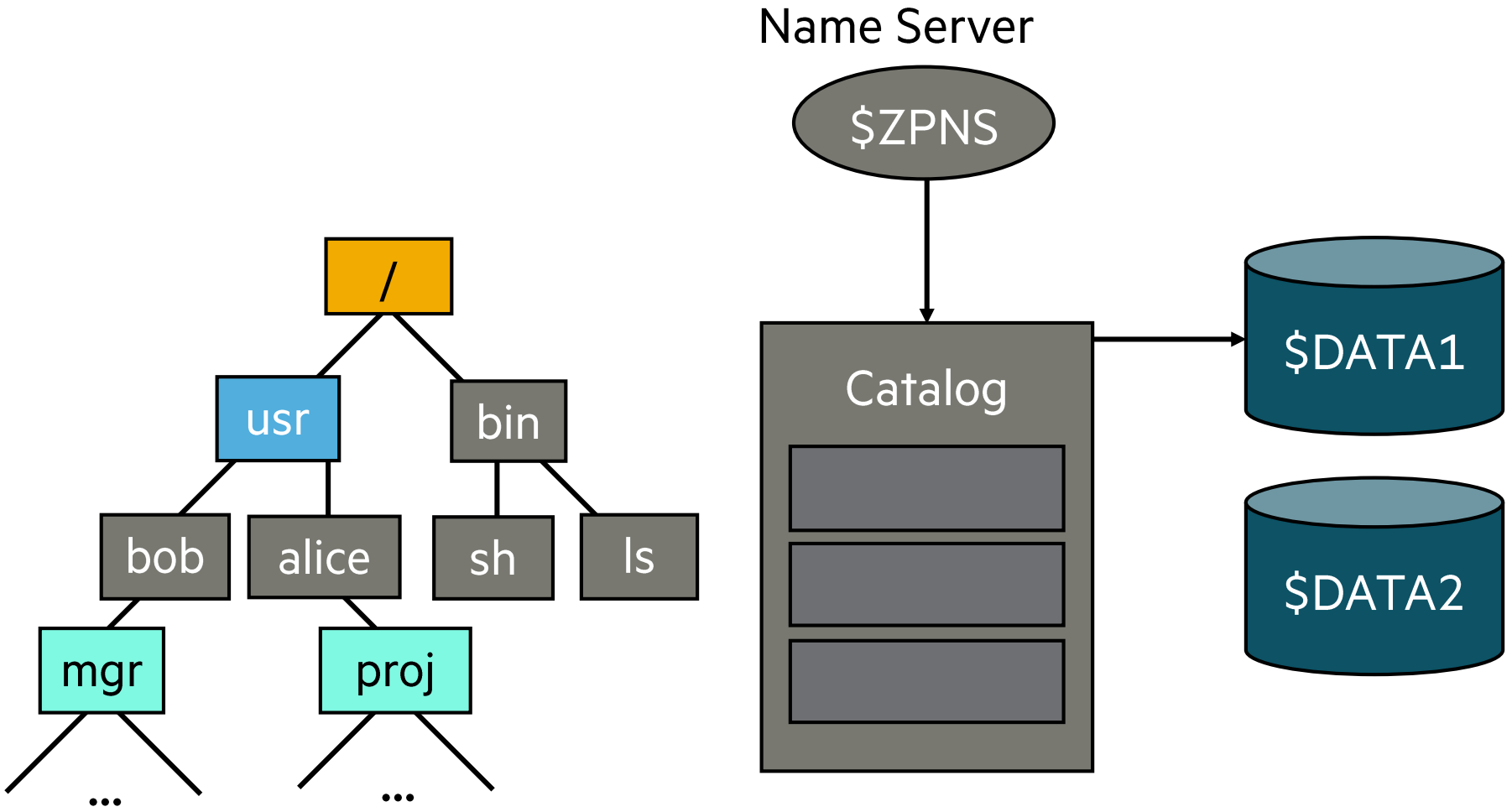
3 Directories in a subdirectory (also called subdirectories)

4 Any number of directories can be a subdirectory

5 Files

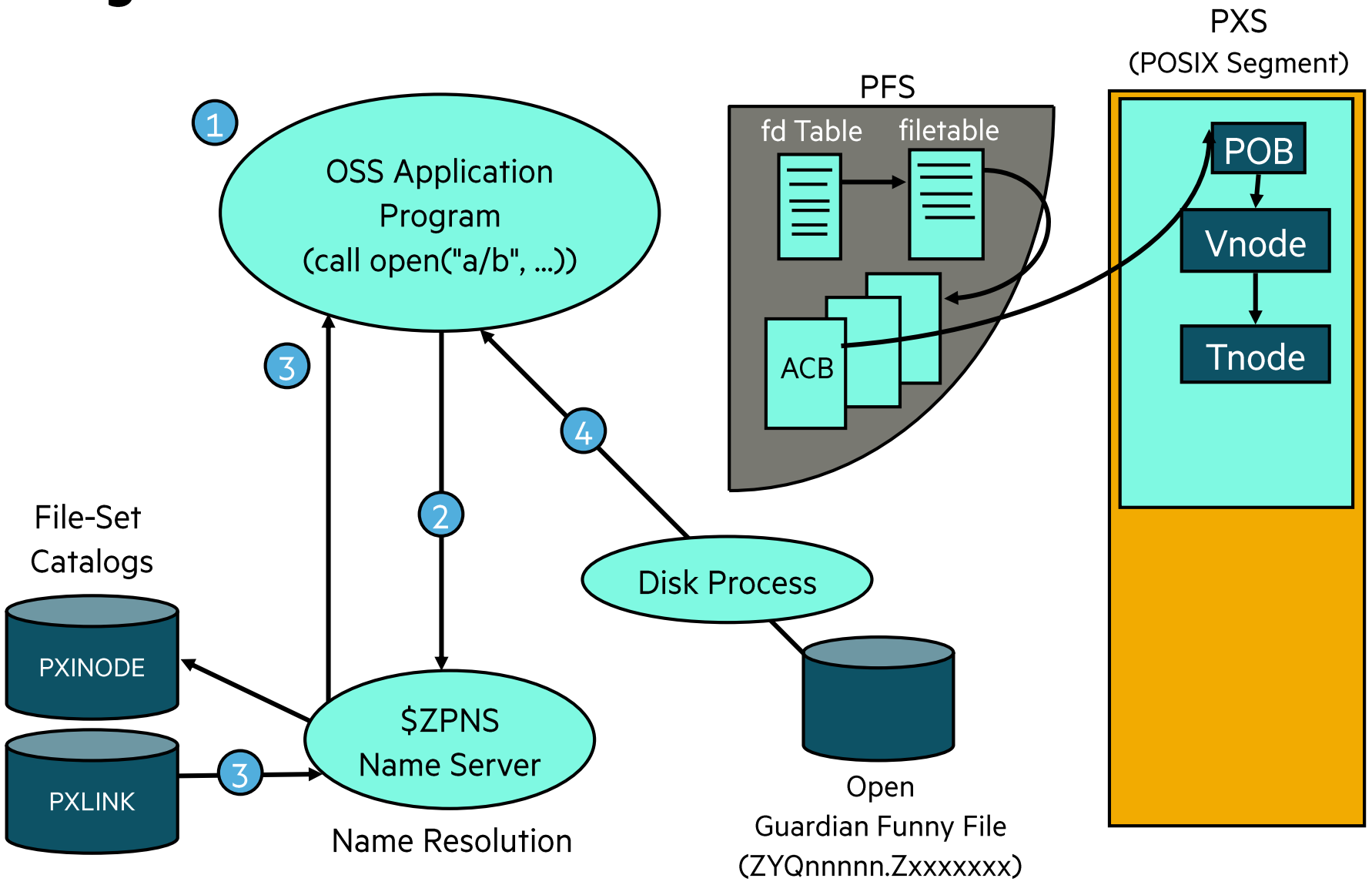
6 Physical disk volumes

OSS Pathname Mapping



`/usr/alice/proj/schedule` → `$DATA1.ZYQ00003.Z000002A`

OSS Open — Regular



Would you like to know more?

- This slide deck is a heavy compressed version of the 5 day “HPE Integrity NonStop Operating System Architecture U8609S” course.
- If you would like to know more, you should enrol for the U8609S course.
- The U8609S course is scheduled in the EMEA (CET) and USA (CT) time zone and delivered in English.
- Check out our schedule on:

www.nonstop-academy.com

or the HPE website:

http://hpe.com/ww/learnnonstop

NonStop Partnership– It's a Beautiful Thing!

Thank you for attending this talk
TBC23-TB69 What is the NonStop Crown Jewel?

Bert van Es – bert.van.es@continuous.nl