

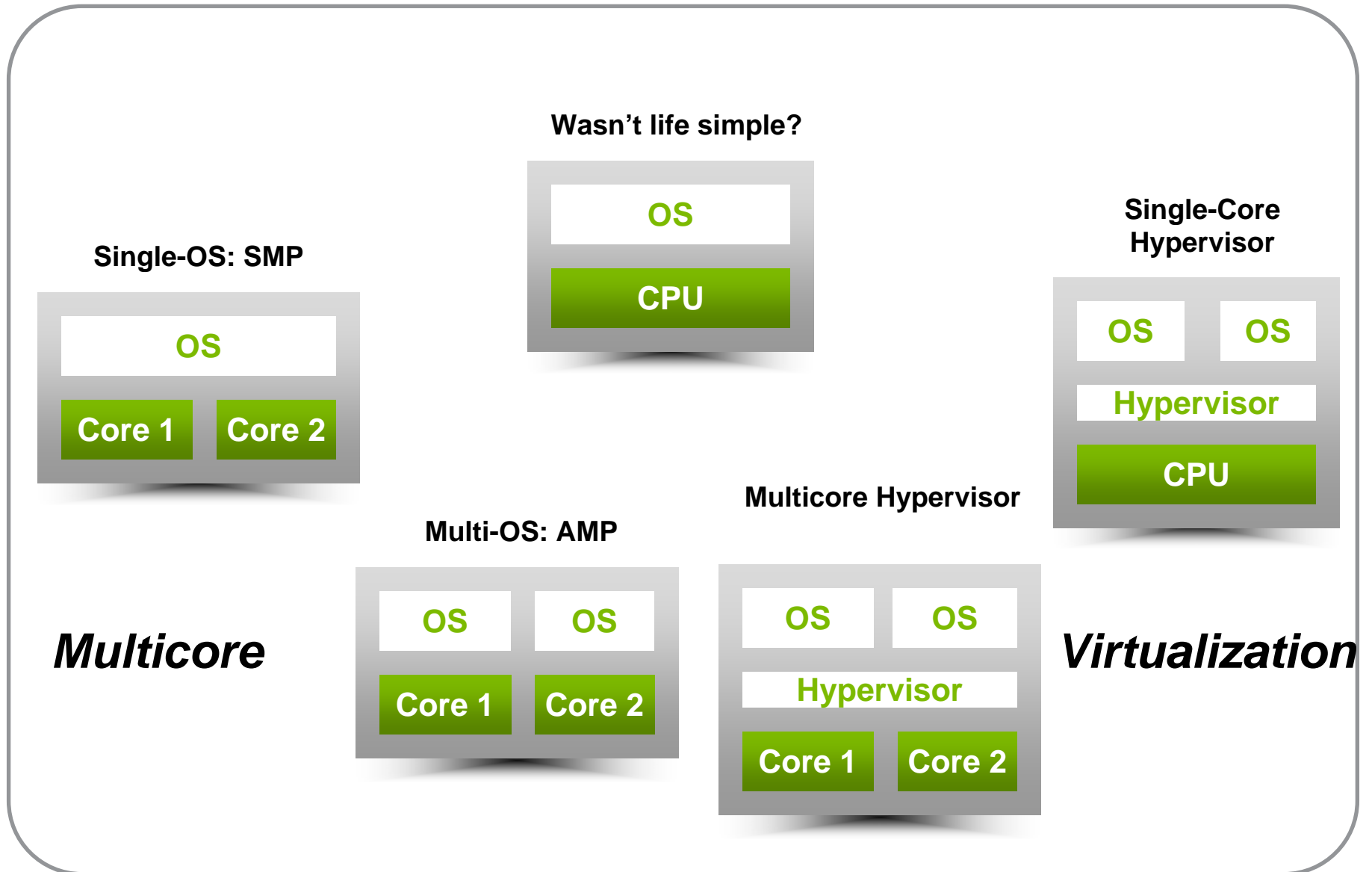
OS Multicore Enablement Wind River

Jegan Arthanari, Technical Account Manager
February 5, 2009

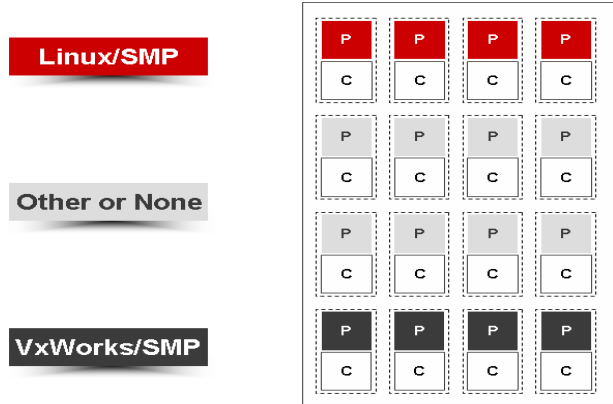
Agenda

- » Multicore software support
- » SMP
- » AMP
- » Hypervisor
- » Multicore Debug

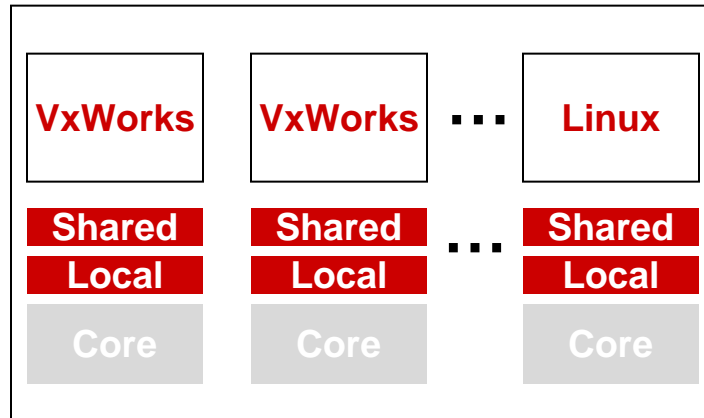
Multicore software support



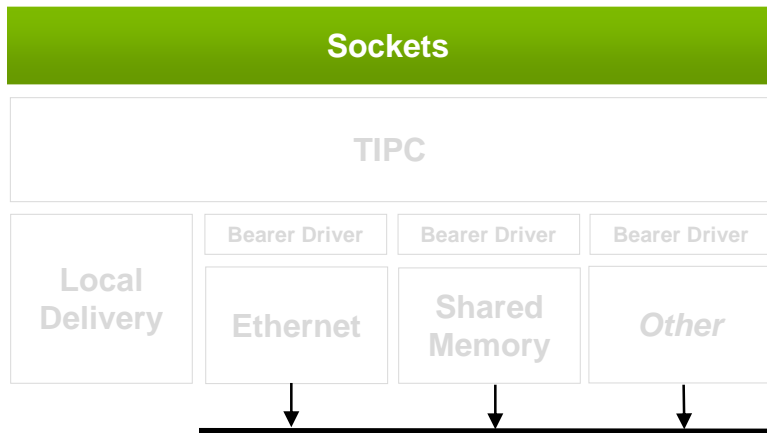
Multicore Run-Time Support



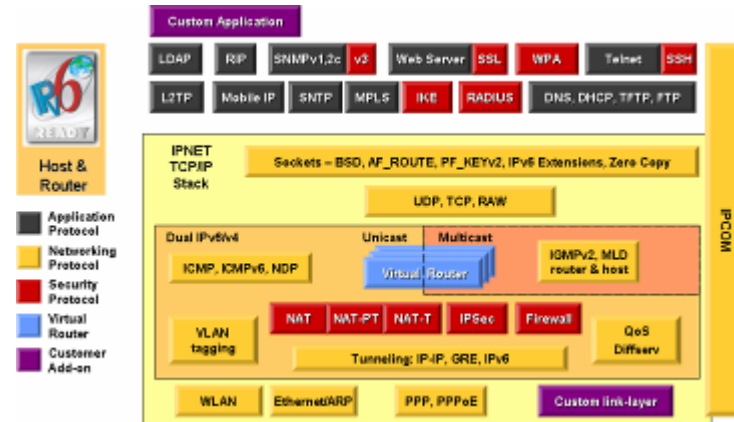
Operating Systems



Intercore Communication
MIPC



Intercluster Communication
TIPC



Multicore Enabled Networking

The Software Solution for Multicore

» Multiple Operating systems

- Real-time (such as VxWorks), safety/security certified (such as VxWorks MILS), general purpose (such as Linux)

» Choice of configurations

- SMP, AMP, SMP+AMP combinations, Hypervisor

» Multi-OS support

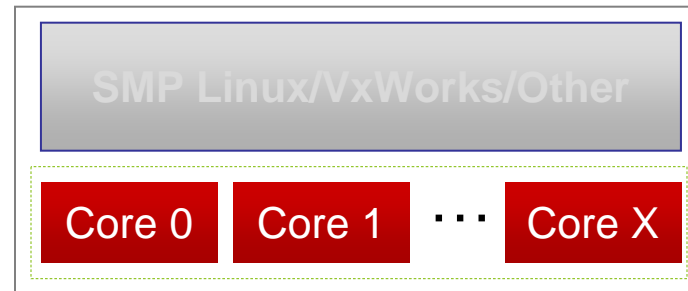
- Boot control, IPC, reliability and network offloading

» Tools to manage complexity

- Debug, profile, and optimize multicore and multi-OS systems

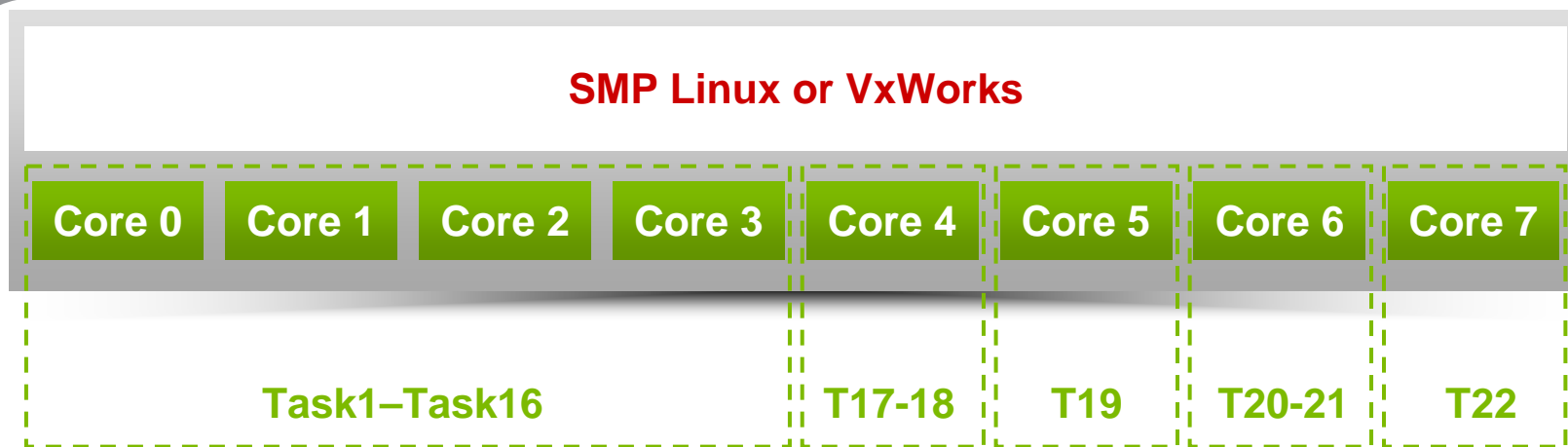
SMP

Symmetric Multiprocessing (SMP)



- » One operating system controls all cores as a resource
- » Abstracts hardware complexity from applications
- » Semaphores, global variables etc can be retained
- » Affinity feature—tasks, interrupts can be bound to a specific core
- » Load balancing provides good utilization
- » Performance depends on number of cores, OS, application

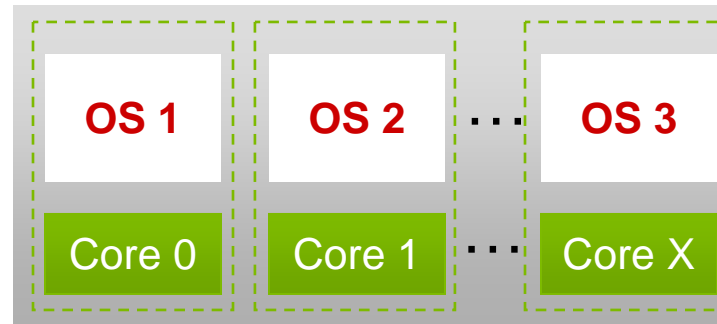
Consider “AMP Within SMP”



- » One operating system controls all cores
 - Simplifies booting, communication, sharing memory, and debugging
- » Affinity is used to bind tasks to specific cores
 - Localized data access and synchronization
- » Scales better than regular SMP worse than AMP
- » Load balancing achieved by SMP scheduler for subset of cores

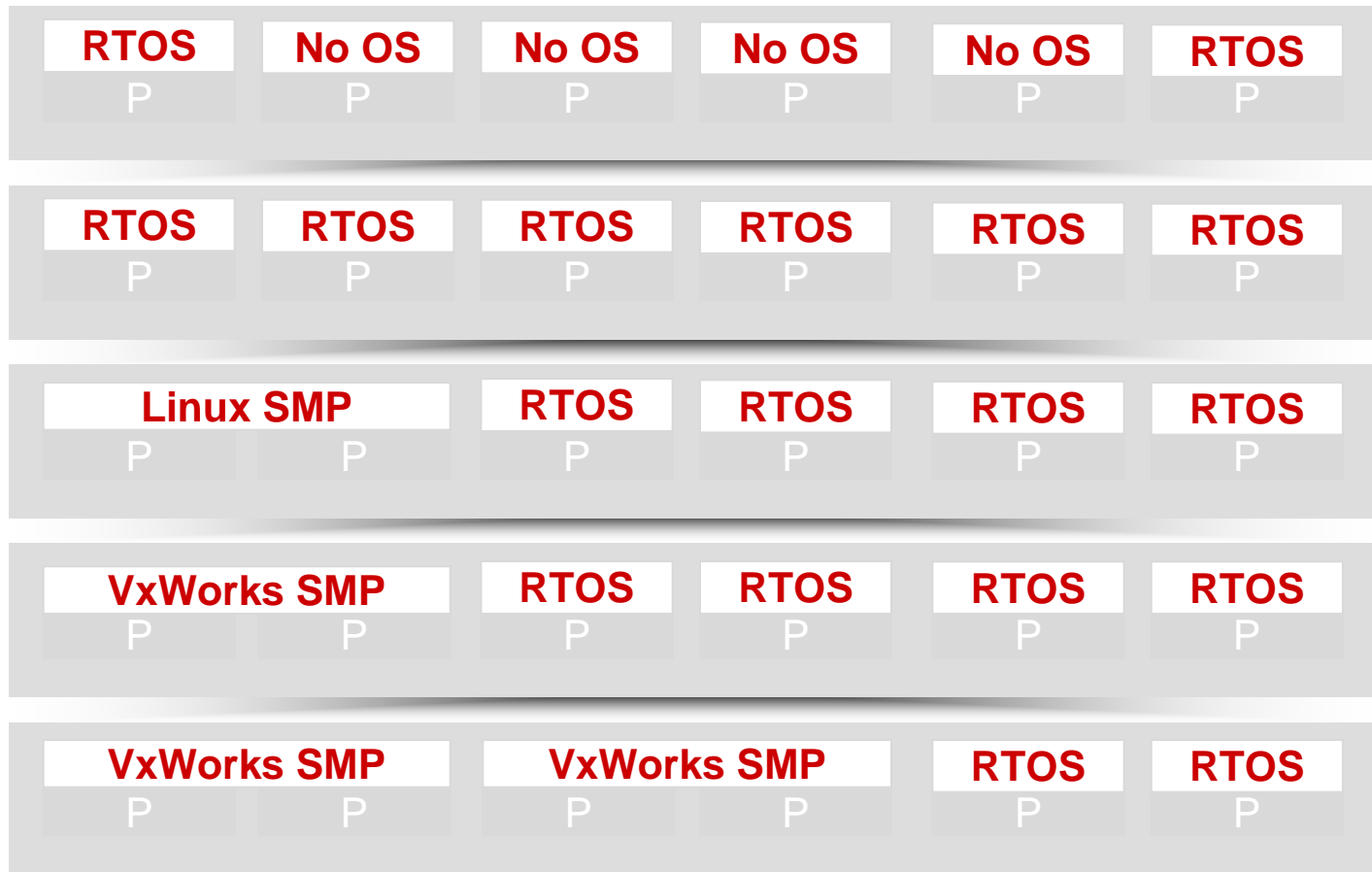
AMP

Asymmetric Multiprocessing (AMP)



- » Homogenous AMP (identical cores) / heterogeneous (different cores)
- » Each core has dedicated local memory plus shared memory
- » Each one has an independent OS or control software
- » Connected for communication and synchronization
- » Performance scales well to large number of cores
- » Utilization may suffer

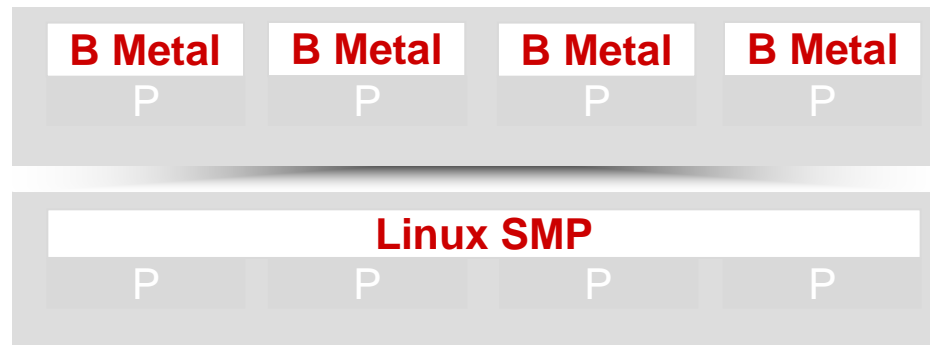
Limitless AMP Configurations



AMP configurations can be created in endless combinations of cores, boards, and operating systems.

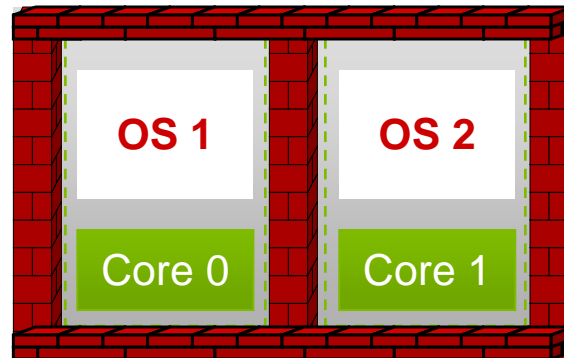
“Bare Metal”

- » When using cores for dedicated tasks and no OS needed
 - E.g. crypto offload, fast path forwarding, etc.
- » “Bare Metal” is not really bare metal
 - Need HW abstraction, IPC, debug support, scalability, ...
- » Consider using a small OS in AMP mode
 - Common APIs, Tools, etc.



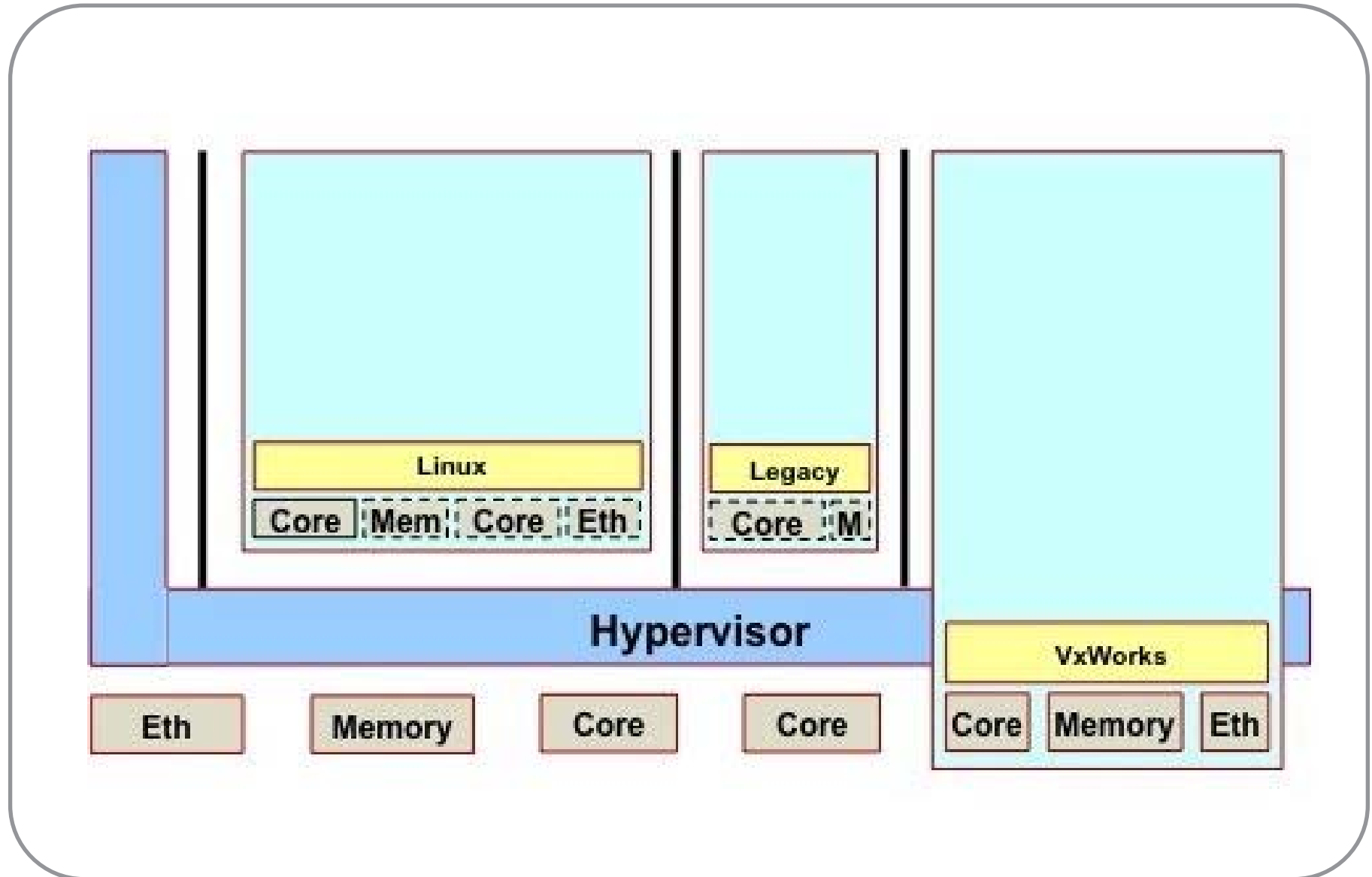
Hypervisor

Using Hypervisor for Protection/Safety/Security



- » Each node has its own separate OS/executive
- » Neither OS can bring down the system
- » Implement heart-beating
- » Restart failed nodes

Hypervisor Architecture Example



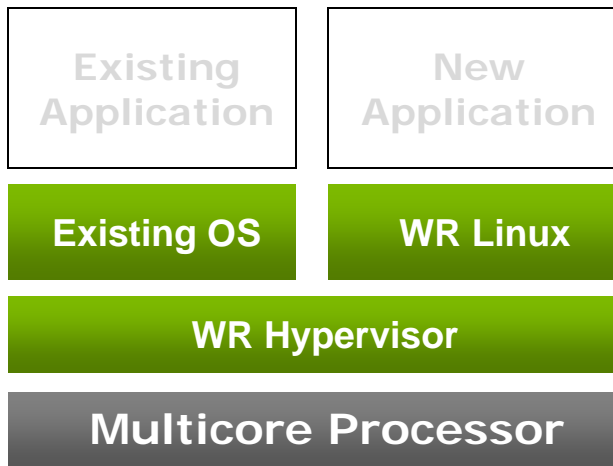
Use of Hypervisors

1. Enables protection between OSs in a Multi-OS system by reducing fault propagation paths between them
2. Enables OS to independently restart in a Multi-OS situation (health monitoring)
3. Enables independence from processor variants
4. Enables systems containing more OSs (and OS types) than cores, including Multi-OS system design on a uniprocessor system

Use of Hypervisors

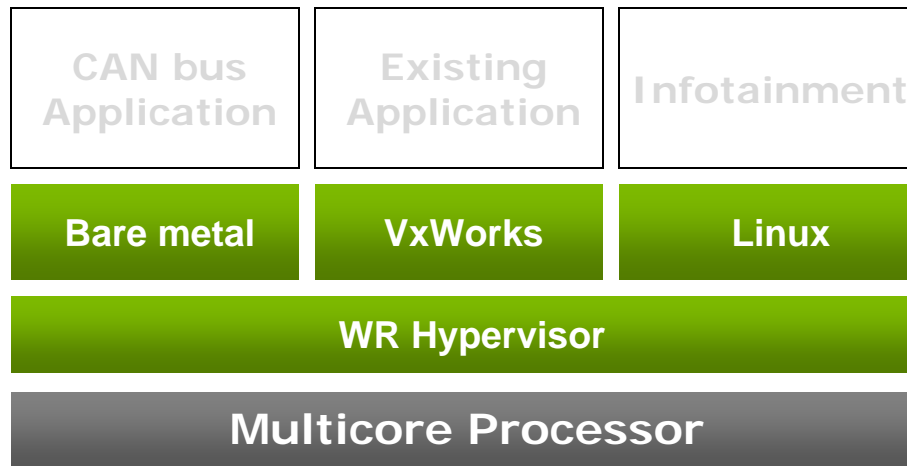
5. Enables system architecture to be partitioned into manageable units (operating systems) tailored for each task (bare metal, Linux, RTOS)
6. Enables fast system start/restart on uniprocessors by deferring slow-initializing OSs on startup
7. Hypervisor enables device sharing

Case Study: Asset Bridge



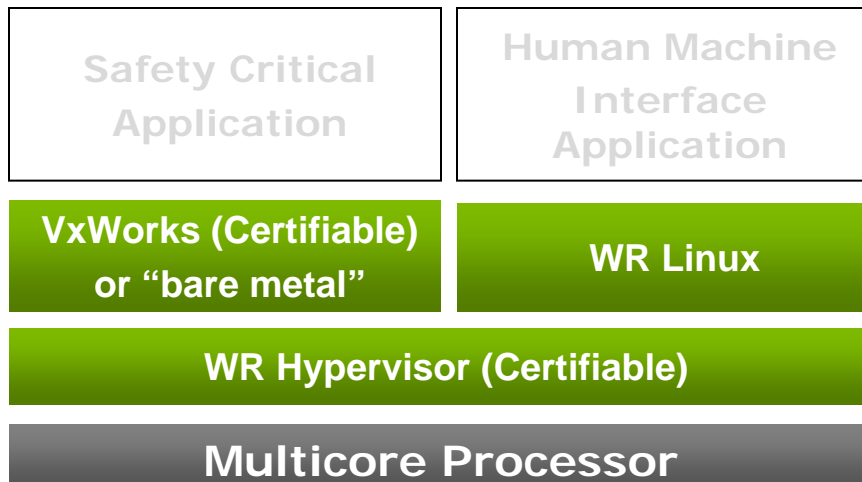
- Leverage and evolve existing assets
- Innovate and leverage new environment
- Networking, Industrial, Automotive, A&D, etc.

Case Study: Consolidation/Reliability



- Increase reliability through multiple independent instances
- Reduce hardware
- Network equipment, Machinery, Cars etc.

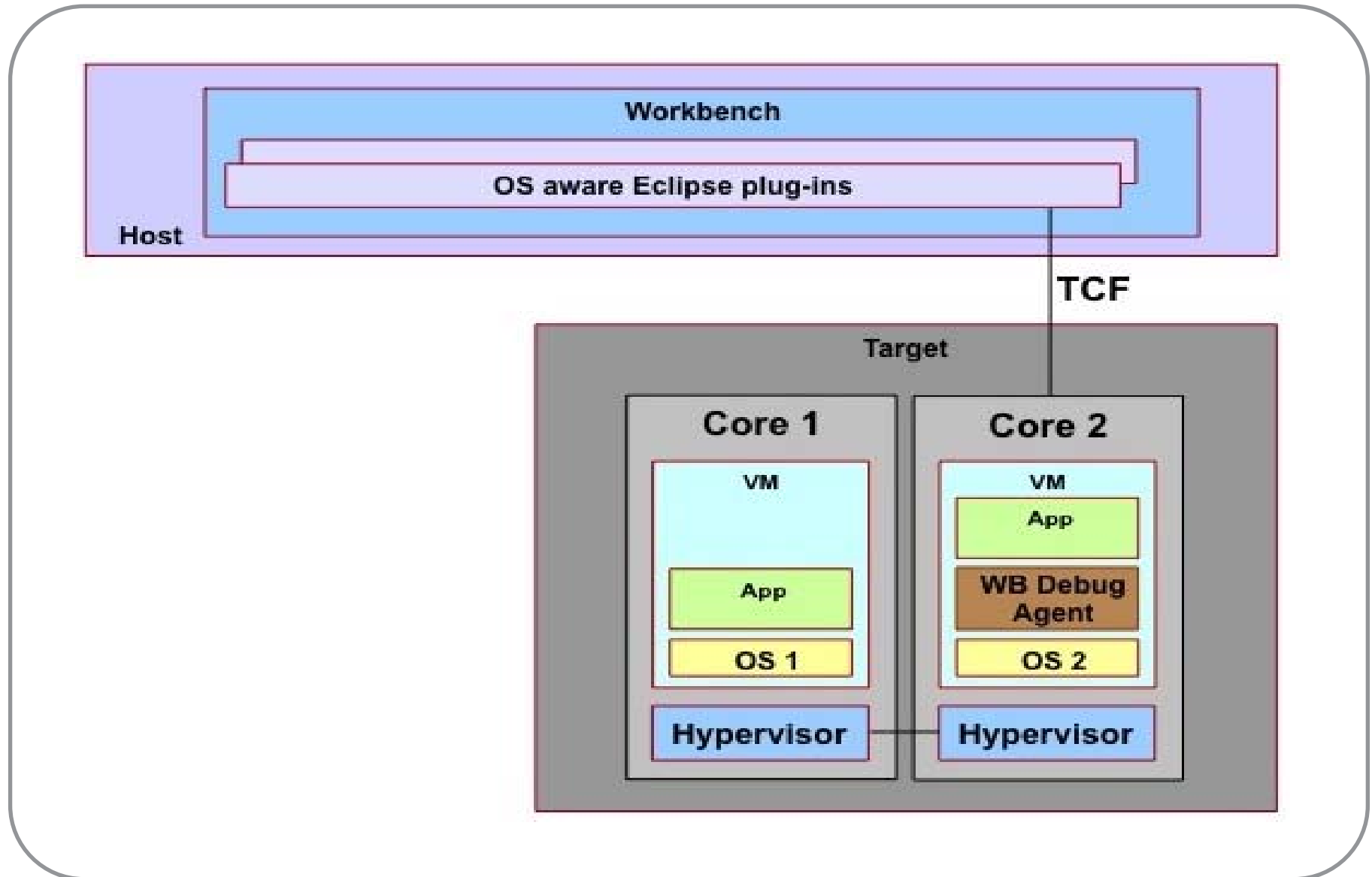
Case Study: Safety



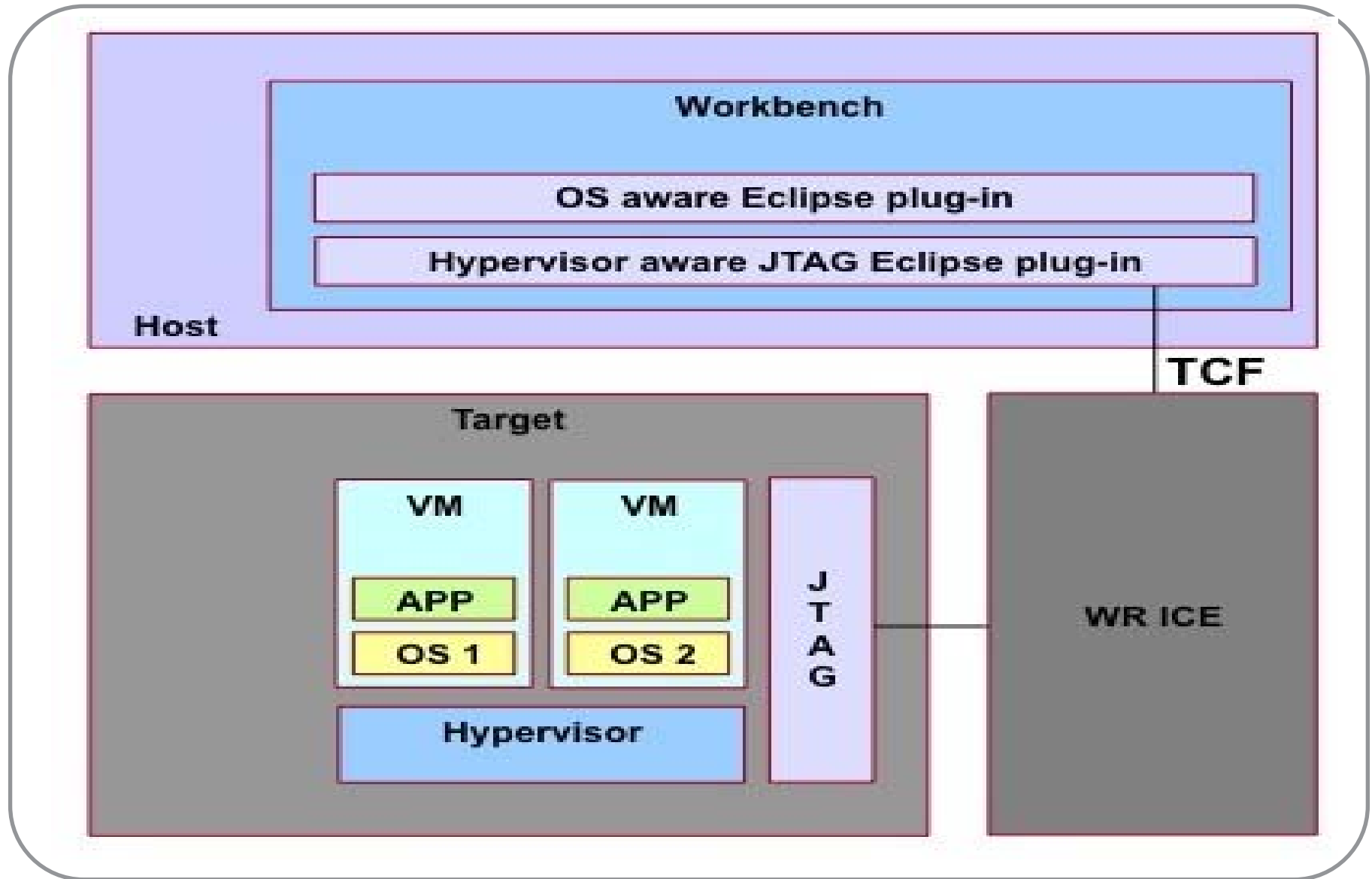
- Preserve certification efforts.
- Innovate and in new environment
- Reduce hardware
- Machinery, Medical, Transport

Multicore Debugging

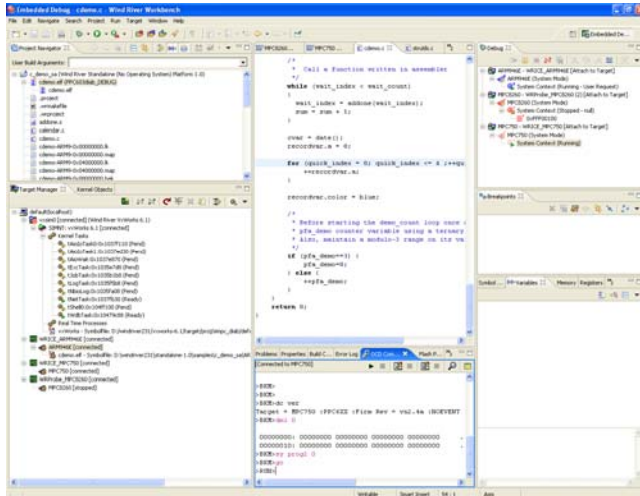
Software Based Multicore Debug



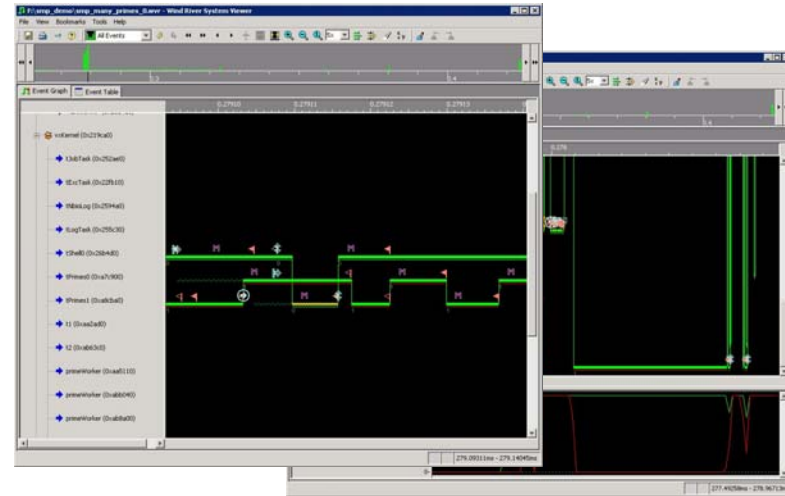
Software Based Multicore Debug



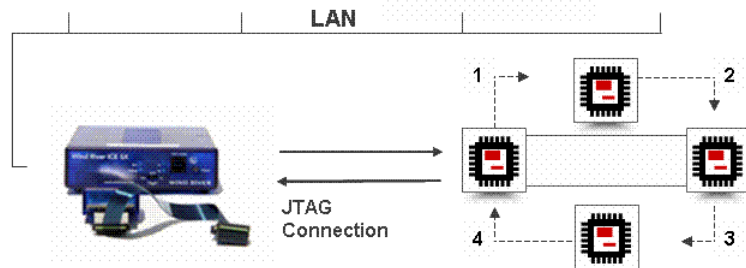
Multicore Debug and Analysis



**Debug Multiple Contexts
Workbench Debugger**



**View Asynchronous Events
System Viewer**



**Multicore On-Chip Debugging
WR ICE**

The screenshot shows the ProfileScope hierarchical profiling tool. The table below displays the performance metrics for various system components.

Function...	Current Indirect %	Current Direct %	Cumulative Indirect %	Cumulative Direct %	Current Usage Bar
0x40000000 (0x40000000)	51.72	0.0	43.30	0.0	
libc_start_main (lib/libc-2.2.5.so)	52.13	0.0	46.5	0.0	
main (usr/bin/bzip2)	52.13	0.0	46.5	0.0	
compress (usr/bin/bzip2)	52.13	0.0	46.5	0.0	
testStream (usr/bin/bzip2)	52.13	0.0	46.5	0.0	
bz2_bzwrite (usr/lib/bz2.so.1.0.2)	51.94	0.0	46.29	0.0	
bz2_compress (usr/lib/bz2.so.1.0.2)	51.74	0.0	46.29	0.0	
handle_compress (usr/lib/bz2.so.1.0.2)	50.70	0.0	43.96	0.0	
bz2_blockSort (usr/lib/bz2.so.1.0.2)	49.8	0.0	36.06	0.0	
mainSort (usr/lib/bz2.so.1.0.2)	43.8	0.0	35.99	0.0	
mainSort1 (usr/lib/bz2.so.1.0.2)	33.14	0.0	20.13	0.0	
mainSort2 (usr/lib/bz2.so.1.0.2)	27.13	0.0	13.55	0.0	
mainSort3 (usr/lib/bz2.so.1.0.2)	24.42	0.0	10.89	0.0	
generateMTRvalues (usr/lib/bz2.so.1.0.2)	4.65	0.0	6.48	0.0	
makeFlags_3 (usr/lib/bz2.so.1.0.2)	2.93	0.0	1.44	0.0	
bz2_hCreateBlockTables (usr/lib/bz2...	0.0	0.0	0.11	0.0	
handle_compress (usr/lib/bz2.so.1.0.2)	0.97	0.0	2.14	0.0	
add_pair_to_block (usr/lib/bz2.so.1.0.2)	0.0	0.0	0.04	0.0	
copy_output_block (usr/lib/bz2.so.1.0.2)	0.0	0.0	0.16	0.0	
write (lib/libc-2.2.5.so)	0.19	0.0	0.11	0.0	
_IO_file_xsputn@@GLIBC_2.1 (lib/libc-2.2.5.so)	0.19	0.0	0.11	0.0	
_IO_file_sprintf@@GLIBC_2.1 (lib/libc-2.2.5...	0.19	0.0	0.11	0.0	
_IO_file_underflow@@GLIBC_2.1 (lib/libc-2...	0.19	0.0	0.11	0.0	
new_do_write (lib/libc-2.2.5.so)	0.19	0.0	0.11	0.0	
_IO_file_open (lib/libc-2.2.5.so)	0.19	0.0	0.11	0.0	
_IO_write (lib/libc-2.2.5.so)	0.19	0.0	0.11	0.0	
system_call (vmlinux)	0.19	0.0	0.11	0.0	
put_write (vmlinux)	0.19	0.0	0.11	0.0	

**Hierarchical Profiling
ProfileScope**