A MULTI-SOCKET FERRARI FOR NFV
GOALS

GOING VIRTUALIZED
ENVIRONMENT-INDEPENDENCY
FLEXIBLE PLATFORM
BACKWARD COMPATIBILITY
HIGH AND DETERMINISTIC PERFORMANCE
HIGH AVAILABILITY
ROBUSTNESS
INTACT DPDK COMMON LIBRARIES
GOALS
GOING VIRTUALIZED

THAT'S LIFE :)

DPDK Summit 2014 | Public | © Ericsson AB 2014 | September 8 | Page 3
GOALS
ENVIRONMENT-INDEPENDENCY

RUNNING IN CLOUD
HYPERVERSOR-AGNOSTIC SOLUTION
RUNNING ALSO NATIVE OR BARE METAL
REUSING EXISTING BLADES
FLEXIBLE YET SIMPLE CONFIGURATION
SUPPORT OF MULTIPLE APPLICATION MODELS
SCALABILITY
PERFORMANCE TUNING

(ADOPTING TO THE ENVIRONMENT WITHOUT HARDCODING)
GOALS
BACKWARD COMPATIBILITY

EASY MIGRATION OF EXISTING APPLICATIONS
GOALS
HIGH AND DETERMINISTIC PERFORMANCE

HIGH THROUGHPUT
LOW PACKET DELAY VARIATION
SAME PERFORMANCE AFTER VM INSTANTIATIONS
EQUAL PER-INSTANCE PERFORMANCE
EQUALLY DISTRIBUTED RESOURCES
GOALS
HIGH AVAILABILITY

TELCO REQUIREMENT
FAST RECOVERY
AVOID SINGLE POINT OF FAILURE
REDUNDANCY
GOALS
ROBUSTNESS

PROTECT AGAINST ACCIDENTAL MEMORY WRITES
MULTI-PROCESS SUPPORT
INCREASED DEBUGGABILITY
GOALS
INTACT DPDK COMMON LIBRARIES

EASY INTEGRATION OF NEW DPDK RELEASES
OUR TYPICAL APPLICATIONS HAVE LARGE MEMORY FOOTPRINTS AND SIGNIFICANT AMOUNT OF RANDOM READS/WRITES SUPPORTING GIANT VIRTUAL MACHINES NUMA-AWARENESS IN XEN HVM
IN REALITY, WE PRESENT A VIRTUAL WORLD TO DPDK
SOLUTION
OUR NEW DPDK EAL

SPLIT EAL INTO PUBLISHER AND CONSUMER
DECOUPLED ENVIRONMENT DETECTION
TOPOLOGY PUBLISHING
(CPU, MEMORY, DEVICE TOPOLOGY)
MEMORY CONFIGURATION PUBLISHING
DEVICE PUBLISHING
(REGISTERED DPDK PORTS)
SELECTIVE MEMORY MAPPING
MEMORY ACCESS CONTROL
(AVOID MAPPING EVERYTHING AS R/W)
VIRTUAL ADDRESS SPACE ALLOCATOR
PRIMARY ROLE PASSING
(DISTRIBUTED RESOURCE MANAGEMENT)
DEVICE (RE-)INITIALIZATION
SOLUTION
IMPROVED MEMORY MANAGEMENT

SIMPLE MEMORY MANAGEMENT API FOR APPS (NAMED MEMORY PARTITION POOLS)
FLEXIBLE CONFIGURATION / RE-CONFIGURATION (WITHOUT CODE CHANGE)
MAKING DPDK NUMA AWARE IN A NUMA FLAT OS
MORE GRANULAR WAY OF PLACING OBJECTS (CONTROL TLB ENTRY USAGE => ZERO TLB MISS)
SOLUTION
IMPROVED MEMORY MANAGEMENT CONT’D

DPDK memory

process1

process2

DPDK memory

DPDK socket 0

DPDK socket 1

DPDK socket 2

memzones allocated/used by process1

memzones allocated/used by process2
SOLUTION

IMPROVED MEMORY MANAGEMENT CONT’D

PARTITIONING MEMORY
(MULTIPLE DPDK NUMA SOCKETS)
FRAGMENTATION SUPPORT
PER-LCORE PRIVATE MEMORY
(DEDICATED DPDK SOCKET ID)
WHILE LEAVING DPDK COMMON LIBRARIES INTACT
KEEP TRACK OF AVAILABLE RESOURCES: LCORE, CPU, MEMORY, PORT, QUEUE, DEVICE ON-DEMAND RESOURCES
RECLAIM RESOURCES
PUBLISHING CONFIGURATION
MEMZONES (MEMDOMAINS FOR APP)
NAMED PACKET POOLS
NAMED VIRTUAL PORTS / QUEUES
The diagram illustrates the relationship between different components:

- **lcores**: Cores with and without Hyper-Threading (HT).
- **CPUs**: CPUs with and without Hyper-Threading (HT).
- **Memory**: NUMA (Non-Uniform Memory Access) and default memory allocation.
- **Ports**: Ports for network communication.
- **Queues**: Queues for data processing.
- **Socket**: Devices connected to the system's socket.

The diagram shows that CPUs and memory are distributed across multiple sockets, with some CPUs having exclusive access to certain memory regions. The NUMA regions are indicated with different colors, with the default region highlighted.

The diagram is a visual representation of how data and resources are managed in a complex system, highlighting the importance of efficient allocation and distribution for optimal performance.
RESULTS
(SAMPLE TEST APP BRIEF)
RESULTS
(DUAL SOCKET - NUMA ALIGNMENT)

Performance

53% improvement

Mixed-NUMA memory

Socket0

Socket1

NUMA-aligned memory

Socket0

Socket1
RESULTS
(SINGLE SOCKET - ZERO TLB MISS)

Fragmented allocation

Performance

Unfragmented allocation

16% improvement

memzones allocated/used by process1
memzones allocated/used by process2
RESULTS
(NUMA ALIGNMENT PLUS ZERO TLB MISS)

Mixed-NUMA memory

Performance

NUMA-aligned memory

Socket0Socket1

Socket0Socket1

77% improvement
CONCLUSION

BE(A)WARE OF THE NUMA
HUGEPAGE BACKING (HYPERVERSOR CONTROL)
TLB MISSES ARE COSTLY IN VM
PROPER MEMORY ALIGNMENT IS VITAL
IDEAS

IMPROVE MEMZONE ALLOCATOR
FURTHER IMPROVE MULTI-PROCESS SUPPORT
PER-LCORE RTE_MALLOC AREA
DPDK DOMAINS
GENERIC RESOURCE MANAGER FOR DPDK
ADDITIONAL CONFIGURATION LAYER TO EAL
ADD CACHE-QOS TO MEMDOMAINS
BACKUP SLIDES
CONFIGURATION EXAMPLE

CPUALIAS

cpualias all {
    cpumask = "0-31"  # all available CPU
}
cpualias foreground {
    cpumask = "2-31:2"  # even cpus excluding cpu0
}
# Per NUMA node shared memory.
# App instances tied to the same NUMA
# are sharing the same memory partition.

memdomain App_NUMA_Shared {
  type = numa
  cpualias = "all"
  alloc_memzone = true
  size {
    is_per_numa = true
    huge_2M = 0
    huge_1G = 1G
  }
}
CONFIGURATION EXAMPLE
EXCLUSIVE TYPE

# -----------------------------------
# Per App instance private memory.
# Every instance has its own memory
# partition.
# -----------------------------------

memdomain App_Thread_Local {
    type = excl
    cpualias = "foreground"
    alloc_memzone = true
    size {
        is_per_cpu = true
        huge_2M = 0
        huge_1G = 1G
    }
}