DPDK

DATA PLANE DEVELOPMENT KIT

Enhanced Memory Management

DPDK Summit - San Jose – 2017

Challenges

DPDK

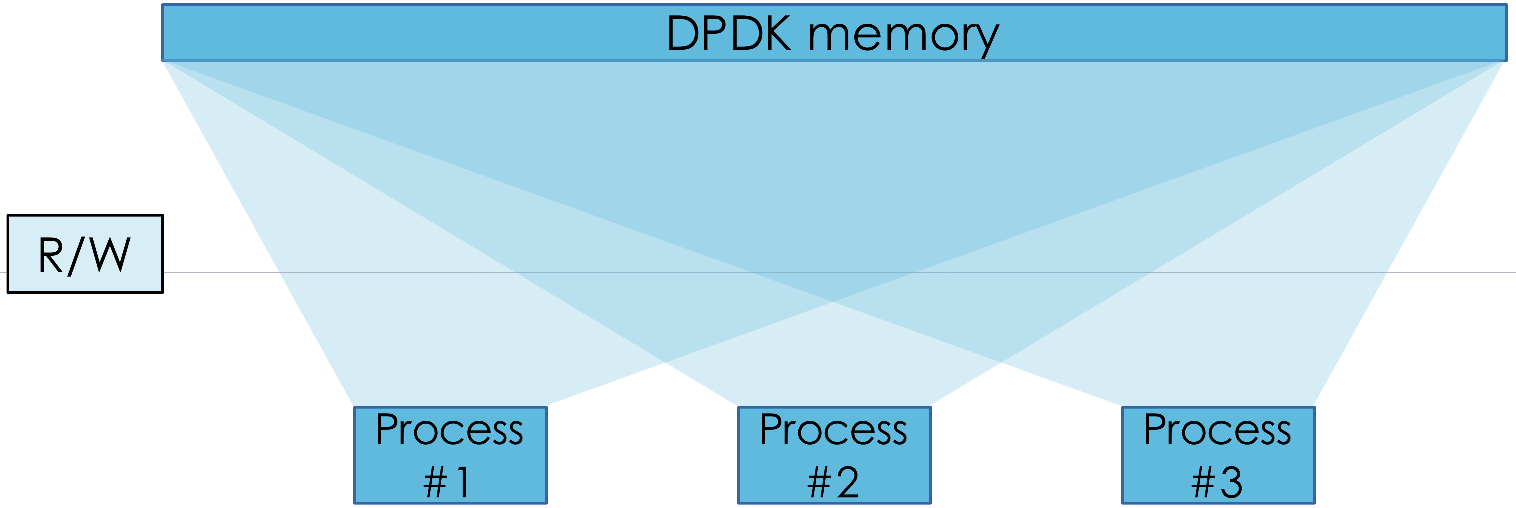
- The world is changing
- Adapt to varying application requirements
  - Performance, Security, Footprint, Robustness?
  - Native, Containers, VMs, Unikernels?
  - x86, ARM, PowerPC, … ?
  - Linux, BSD, Windows, … ?
- Abstract complexities of the environment from applications
  - Different environments with variable ways to allocate/attach to memory
  - Same valid for other resources such as network interfaces, HW accelerators etc.

Current Limitations

- Static hugepage memory allocation at DPDK initialization time
  - Dynamic allocation is not possible
- Memory initialization takes a long time (collecting, mapping, zeroing)
- DPDK relies on physical memory information (not always available)
  - Physically contiguous segments (allocation failure if no physically contiguous mem)
  - PMDs rely on physical memory for DMA
  - No virtually contiguous memory support
- No support for memory mapped files, shared memory segments, ivshmem, ramfs etc.

DPDK Today – Memory Initialization

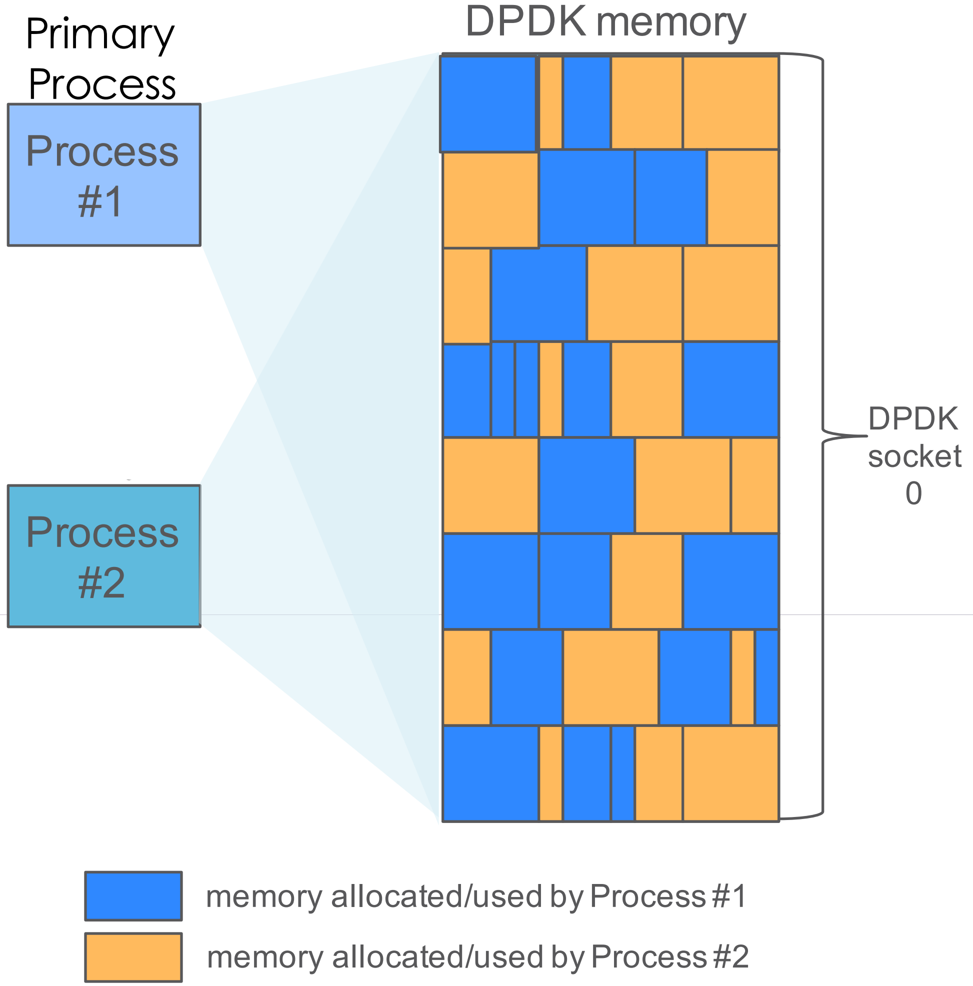
- DPDK grabs all required hugepages and fills up its heap during initialization
- Everything is mapped with R/W permissions in all processes



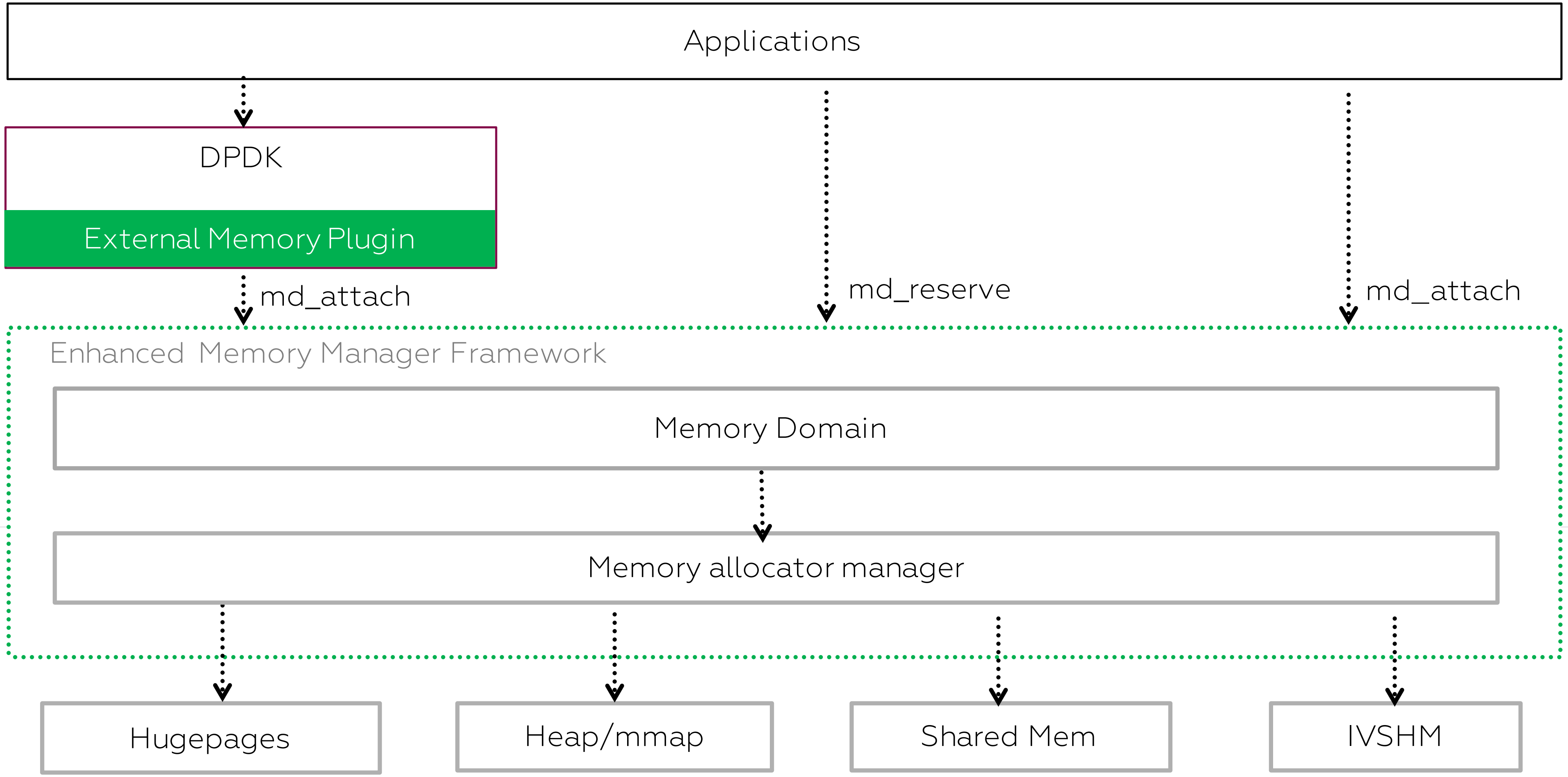
DPDK Today – Multi-process Sequence

1) Primary Proc #1 and Proc #2 started
2) Proc #1 inits memory, Proc #2 is waiting
3) Primary Proc #1 releases lock,
   Proc #2 attaches to DPDK memory
4) Both processes start allocating memory
   at the same time (lock contention)

No Isolation!
No Memory Protection!
No fine grain control of object placement!
(except NUMA and page sizes)


Primary Process
Process #1
Process #2
DPDK memory
DPDK socket 0
memory allocated/used by Process #1
memory allocated/used by Process #2

Enhanced Memory Manager Framework



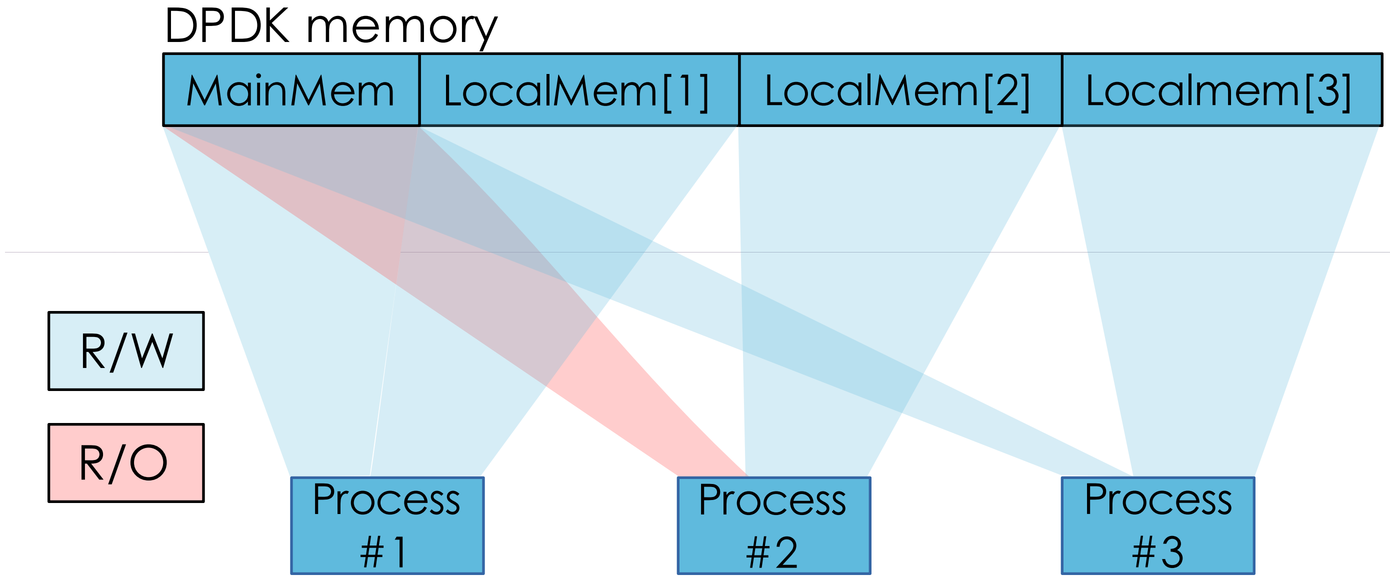
Proposal

- Both DPDK and the enhanced memory manager are used by applications
  - Maintain backward compatibility
  - Applications could directly use memdomain APIs to get required memory on demand
- Extend DPDK to support external memory manager plugins
  - DPDK would be a user of memory domain
- Enhanced memory manager is a framework
  - support many types of memory allocators, such as hugepages, mmap/heap, shared memory, or inter-vm shared memory

Named Memory Partitions - Feature

- DPDK/Apps are attaching to named partitions pools (memdomains)
- Multiple processes can attach to partitions with distinct access rights
- Processes/threads can attach to named partitions on-demand

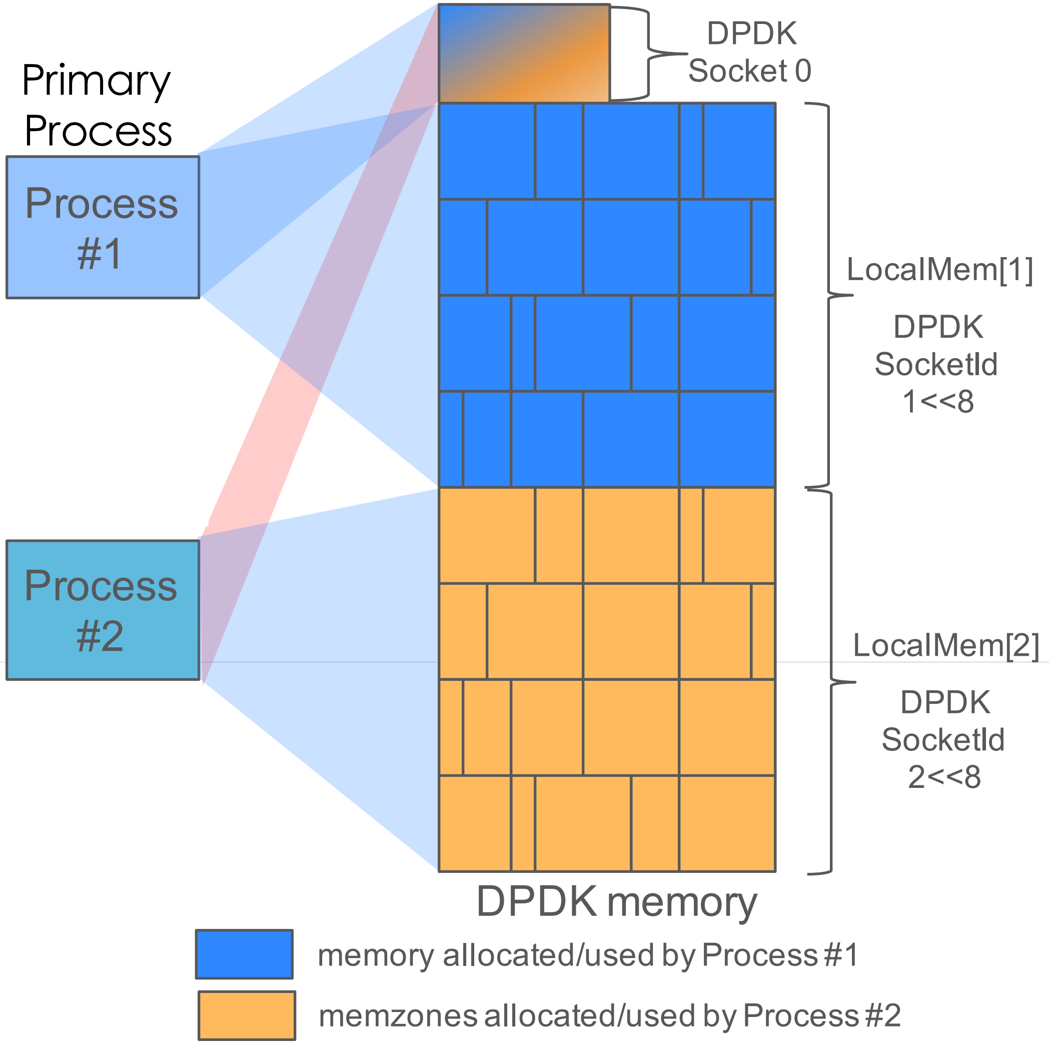
```
memdomain MainMem {
    type = numa
    cpualias = "all"
    policy = static
    size {
        huge_4K = 128MB
        huge_2M = 256MB
    }
}
```



Named Memory Partitions - Sequence

1) Primary Proc #1 and Proc #2 started
2) Primary Proc #1 inits "MainMem" partition
3) Proc #2 awakes, attaches to "MainMem"
4) Proc #1 and #2 attaches to "LocalMem" (allocates/maps partition) – no zero on req.
5) Both processes start allocating memory within their own partition (no contention)

Access Control!
Placement Control!
Isolation!
Memory Protection!



🟦 memory allocated/used by Process #1
🟧 memzones allocated/used by Process #2

Named Memory Partitions - Possibilities

- Flexible environment specific configuration
  - Adapt to different architectures, e.g. no page-size hard-coding in application
  - Adopting configuration to required policy, e.g. Performance, Security, Footprint
  - Performance tuning accelerated by publishing different configurations
- Access control
  - External resource manager managing the resources of applications running in containers or VMs
- Memory classification
  - Creating fast/medium/slow partitions on x86 (zero TLB miss, 2M huge, 4K)
- Physically contiguous memory partition only for DMA

Named Memory Partitions - Possibilities

- Virtual address space control:
  - Short pointer support by requesting specific virtual address range
  - On demand static or dynamic virtual address assignment
- Transparent NUMA awareness
  - Each process requests local memory partition which is created based on the location of that instance
- Scale up/down
  - Allocate/Free resources on-demand as processes start/stop
- Support of different types of shared memory techniques
  - Named partition for Inter-container or Inter-VM shared memory (global namespace)
- etc.

Previous Work

DPDK Summit 2014

László Vadkerti

Ericsson Lead Software Developer

András Kovács

Ericsson Lead Software Developer

Multi-Socket Ferrari for NFV

DPDK Userspace 2015

GENERIC RESOURCE MANAGER

ANDRÁS KOVÁCS (ANDRAS.KOVACS@ERICSSON.COM)
LÁSZLÓ VADKERTI (LASZLO.VADKERTI@ERICSSON.COM)

A manager we would like :)

Questions?

Laszlo Vadkerti
laszlo.vadkerti@ericsson.com

Jiangtao Zhang
tom.zhang@ericsson.com

THANK YOU!