Dynamic Mempool:
One of the Final Steps to Make DPDK Cloud-Native

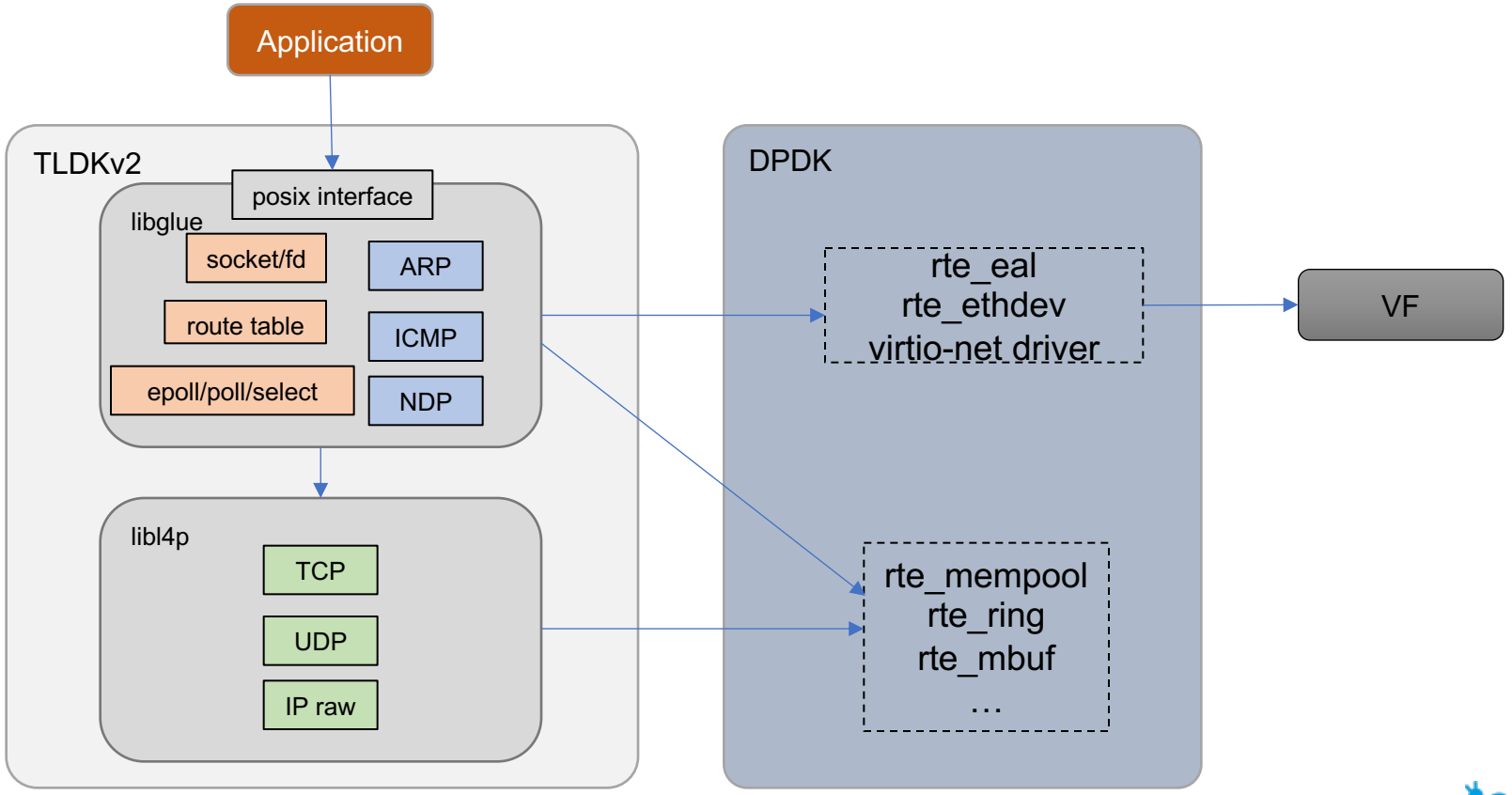
Jielong Zhou

jielong.zjl@antgroup.com

蚂蚁集团
ANT GROUP
www.antgroup.com

Our scenario: TLDKv2 in cloud-native environment

- TLDKv2 (https://github.com/FDio/tldk/tree/dev-next-socket)
  - As the net stack for application kernel, run cloud-native APPs



Typical DPDK application VS Cloud-native applications

Typical DPDK applications

- High bandwidth, high throughput
- Single (few) instance(s) on a server
- Long lifecycle
- Reserved hugepages
- Initialize resources and preallocate memory at startup
- **Care little about initialization time and memory cost**

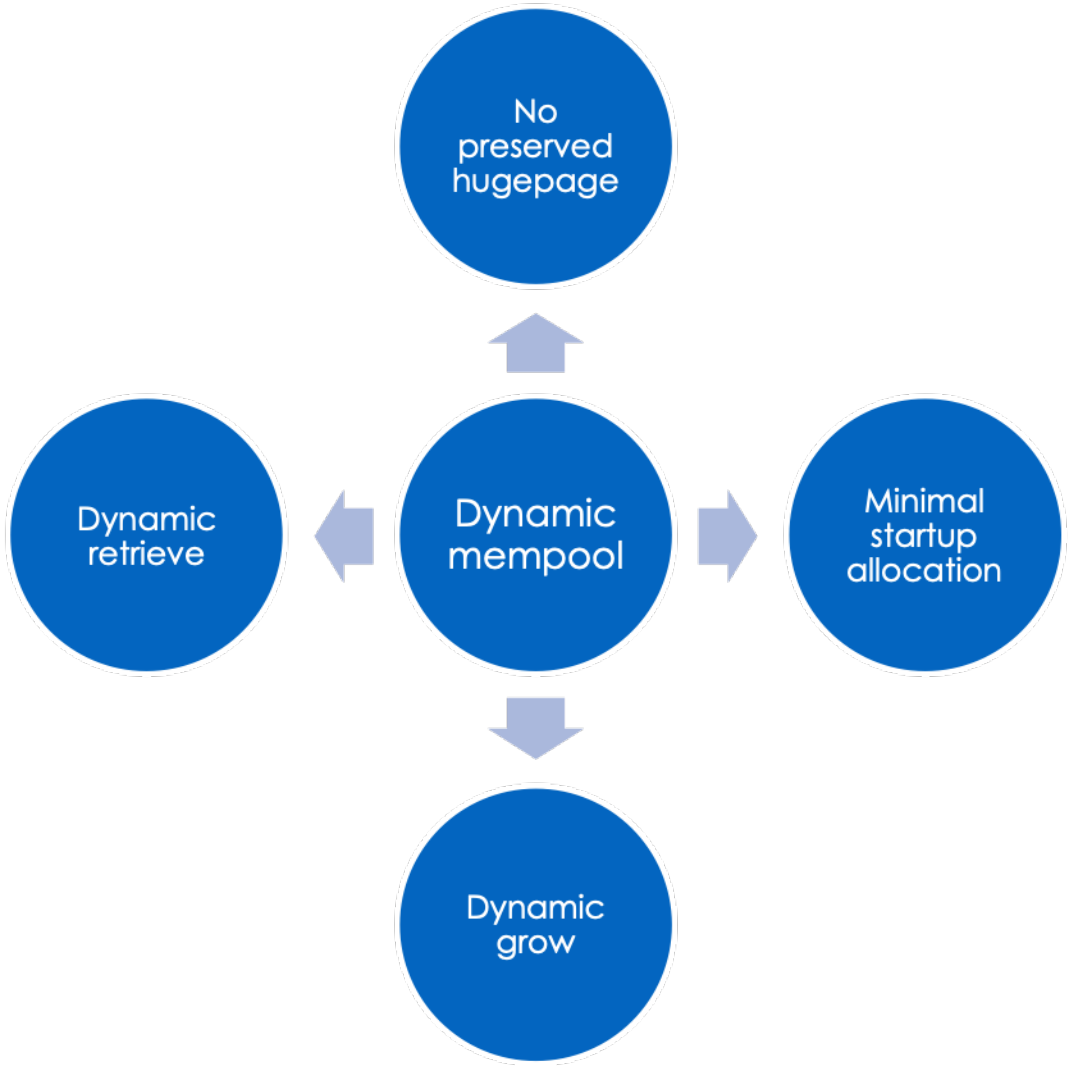
Cloud-native applications

- Dynamic load and throughput
- Many instances on a single server
- Short lifecycle
- Elastic resource consumption shared among applications
- **Sensitive to startup speed**
- **Sensitive to memory cost**

Mempool: one of main issues

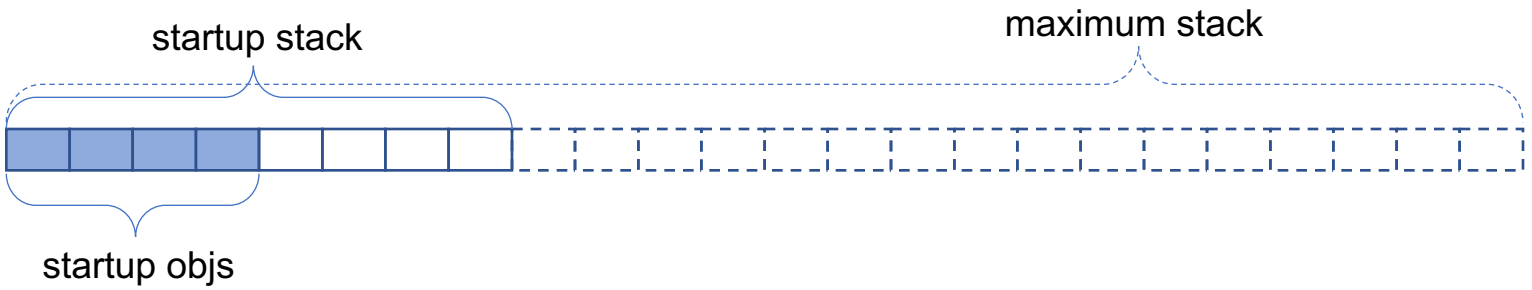
- Cloud-Native: Elastic memory (expand and shrink)
- Mempool is a fundamental part of DPDK
  - Dynamic memory mode only makes it dynamic for memseg
  - Libs/Drivers: rte_mbuf, rte_ethdev, …
  - Hard to replace it with new libraries
- Varying # of mbufs used over time
  - TLDKv2: mbufs stored in send/recv buffer or rx/tx queues

Solution: dynamic mempool



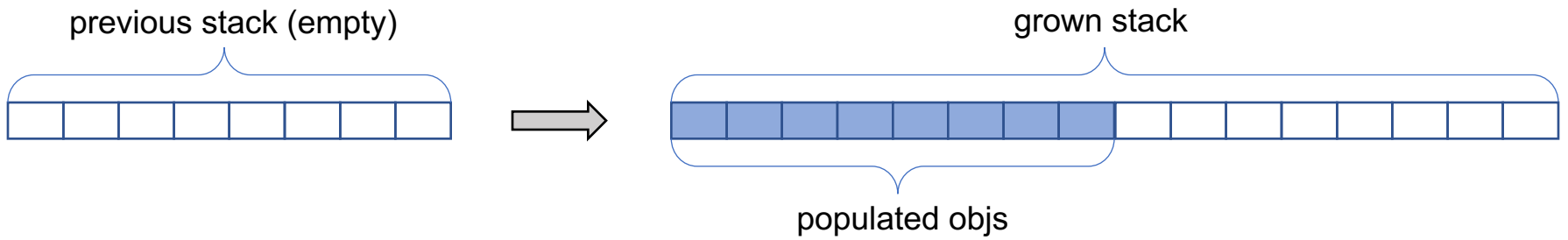
Dynamic mempool: initialization

- Set resource limitation and dynamic allocation count
- Allocate management structures (ring/stack) with minimal size
- Allocate minimal amount of objects in mempool



Dynamic mempool: populate

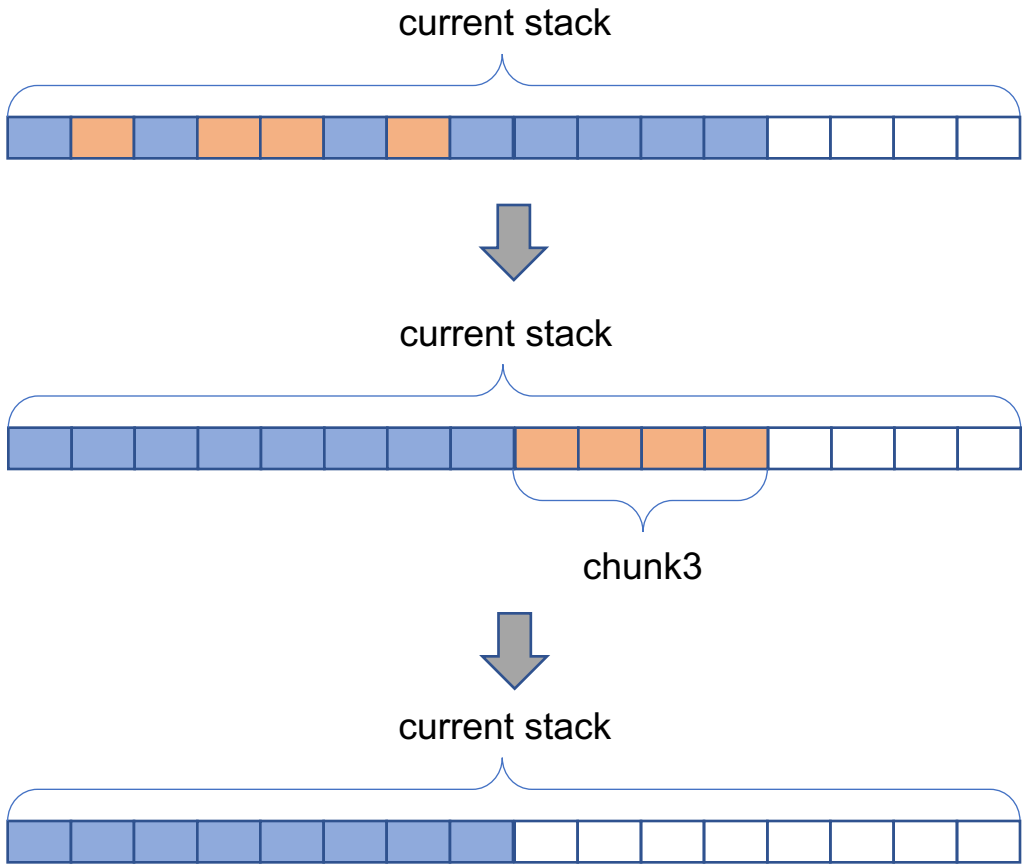
- Triggered during dequeue operations when there are not enough objects left in mempool
- Populate a specified amount of new objects and insert them into management ring/stack
- Allocate bigger management structure to hold increased objects if necessary
- Based on DPDK dynamic memory management on memseg



Dynamic mempool: shrink

- Triggered when amount of free objects is more than threshold (e.g. > 75%)
- Check if any memchunk is totally unused
  - Record number of total objects and free objects in each memchunk
  - Memchunk is totally unused if total_num equals free_num
- Release unused memchunk
  - Remove objects of target memchunk from ring/stack, and free them
  - Compact left objects in ring/stack
  - Remove memchunk from chunk list of mempool
  - Free memchunk and related memzone

Dynamic mempool: shrink

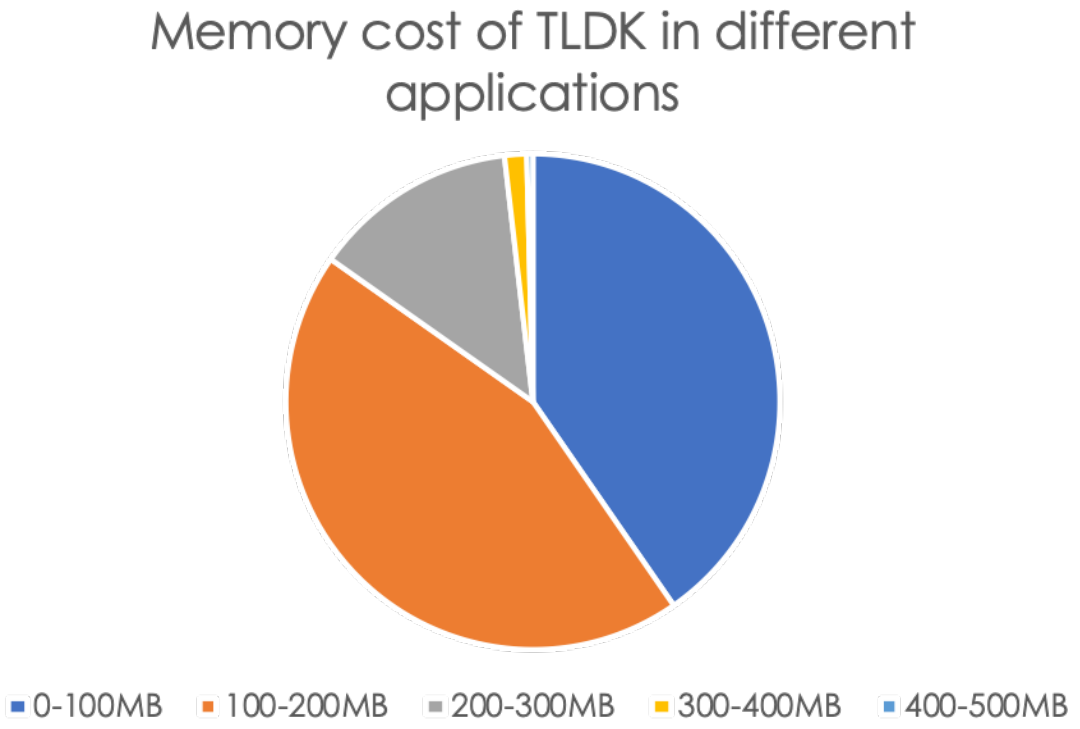


chunk1: total=4, free=3
chunk2: total=4, free=3
chunk3: total=4, free=4
chunk4: total=4, free=2

chunk1: total=4, free=3
chunk2: total=4, free=3
chunk4: total=4, free=2

Dynamic mempool

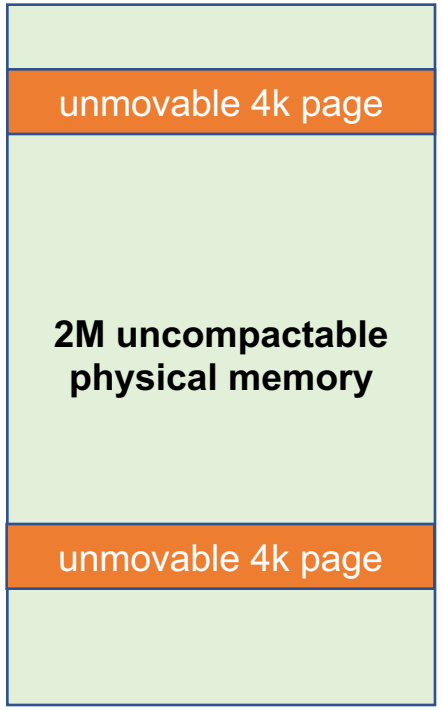
- Startup memory: **from >1GB to <100MB**
- Startup time: **from >1s to <100ms**
- Elastic memory



Hugepage or not?

- Non-hugepage leads to acceptable performance degradation

- **What's worse**, used with pass-through devices, lead to large amount of non-compactable memory fragments
  - VFIO is necessary to avoid DMA attack; however it requires pinning memory which is unmovable and affects memory compaction
  - In worst case, **100MB** of 4k pages may spread in **50GB** physical memory, causing no hugepages could be allocated in this address space.



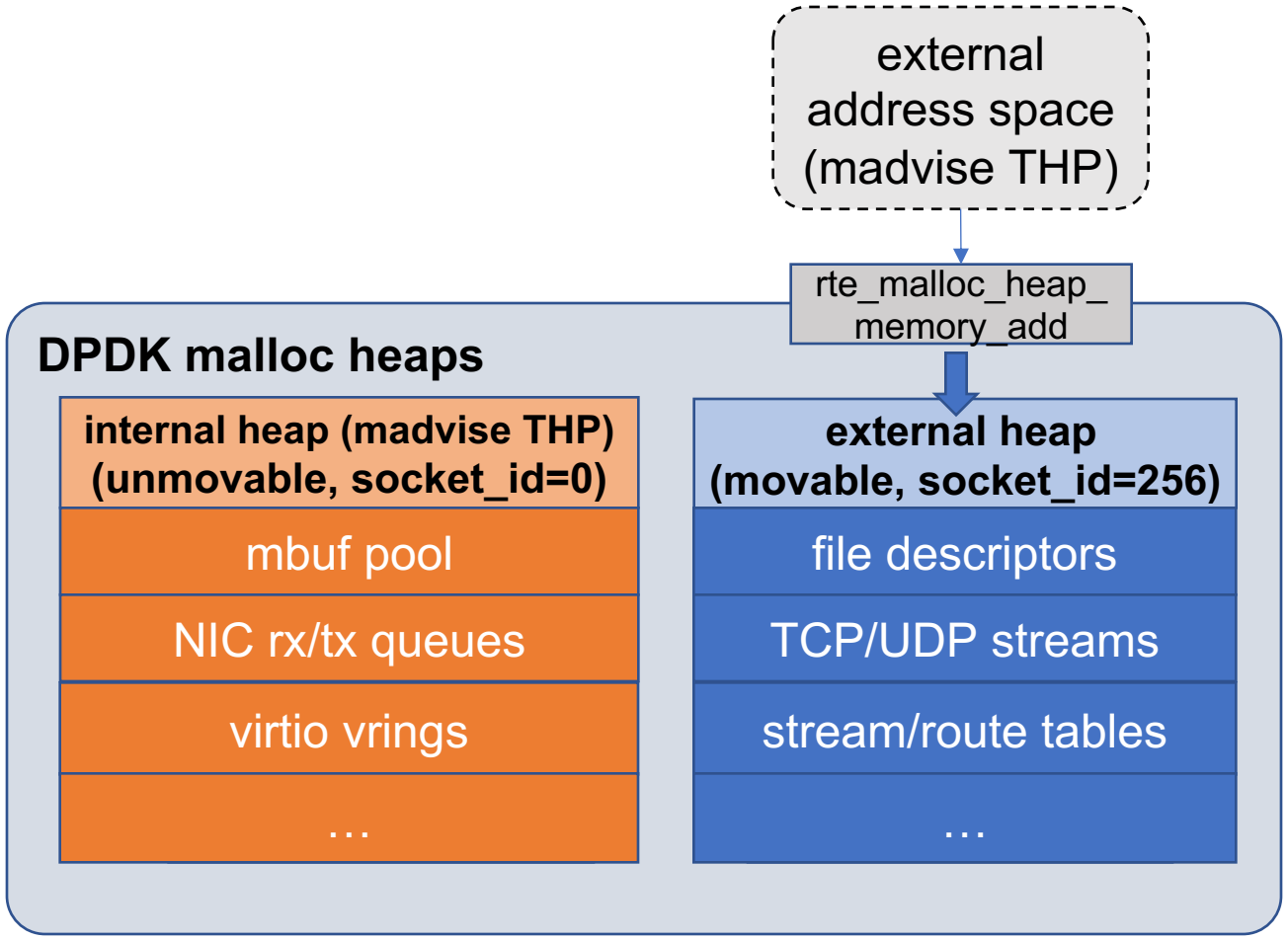
Solution: Transparent hugepages

- Use transparent hugepages in kernel
  - set /sys/kernel/mm/transparent_hugepage/enabled

- When DPDK allocate memory dynamically, call madvise to suggest kernel allocating hugepages for the allocated virtual address

Reduce unmovable memory by external memory

- Allocate most resources in external heap memory
- Reduce unmovable memory to about **1/10**.



Future work: more effective shrink

- Current shink method sometimes inefficient
  - No memchunk is totally free while there are many free objects
- Our idea: Add a mechanism to prevent memchunks nearly empty to be used.

Summary

- Cloud-native environment requires DPDK to be more lightweight and elastic
- Dynamic mempool: one solution to support elastic memory usage and quick startup
- TLDKv2 has been adopted in dozens of key applications of Ant group. And DPDK is proved could be a solid part for cloud-native network applications.

THANK YOU

蚂蚁集团
ANT GROUP

www.antgroup.com