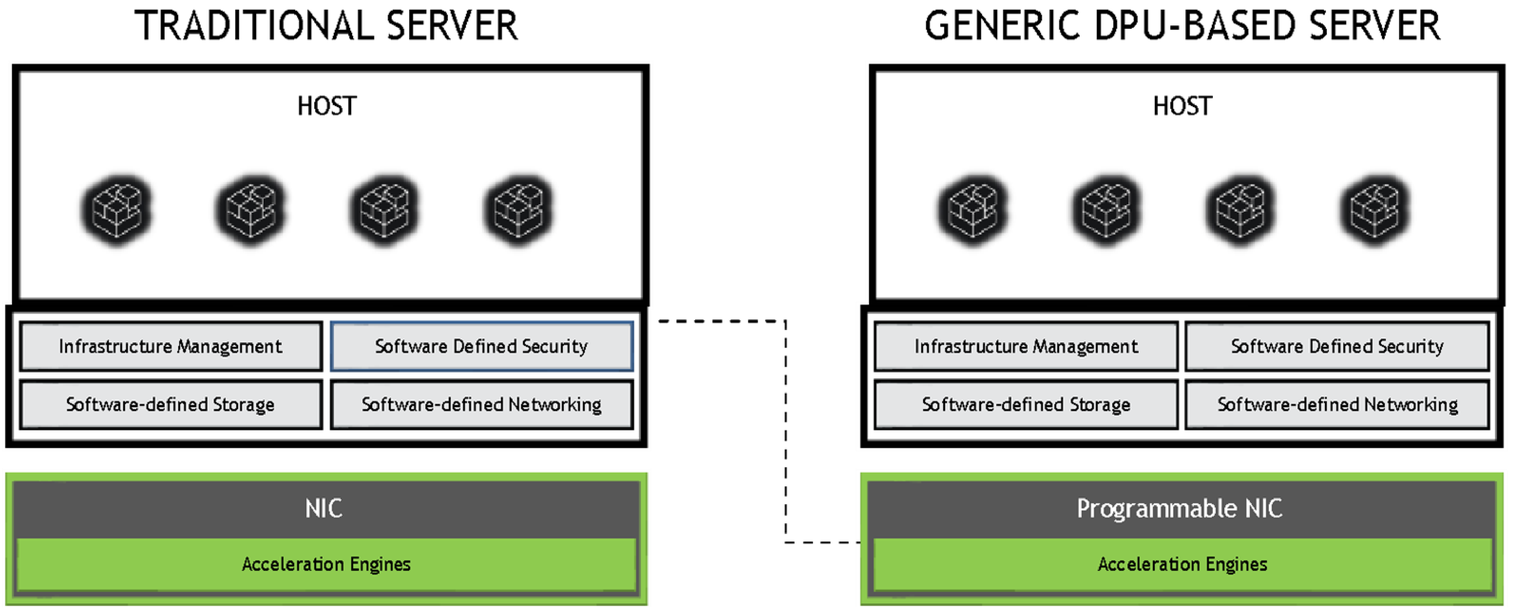
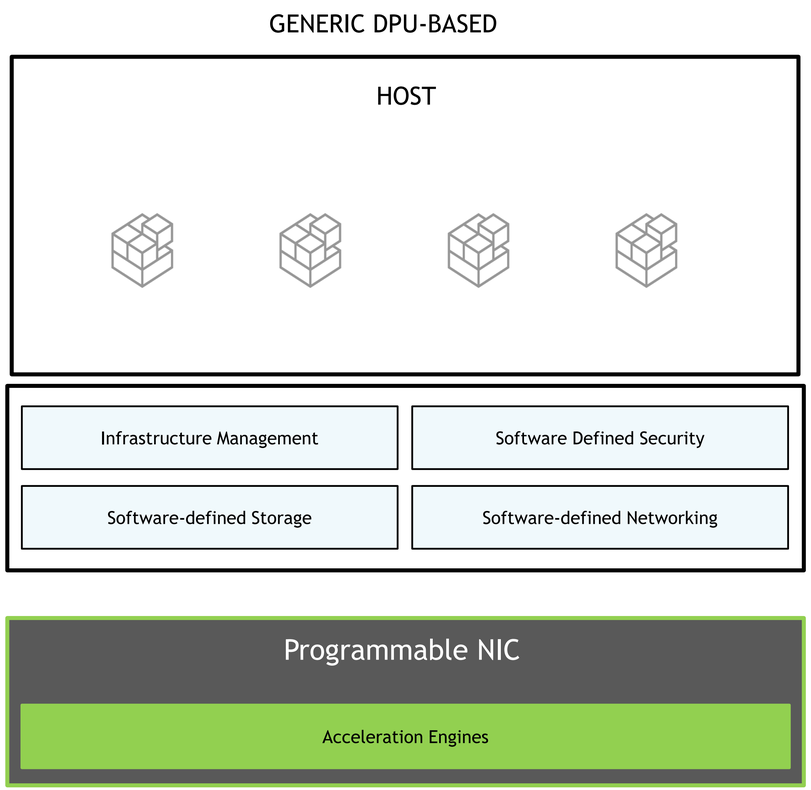
Cross-GVMI UMR Mkey Pool

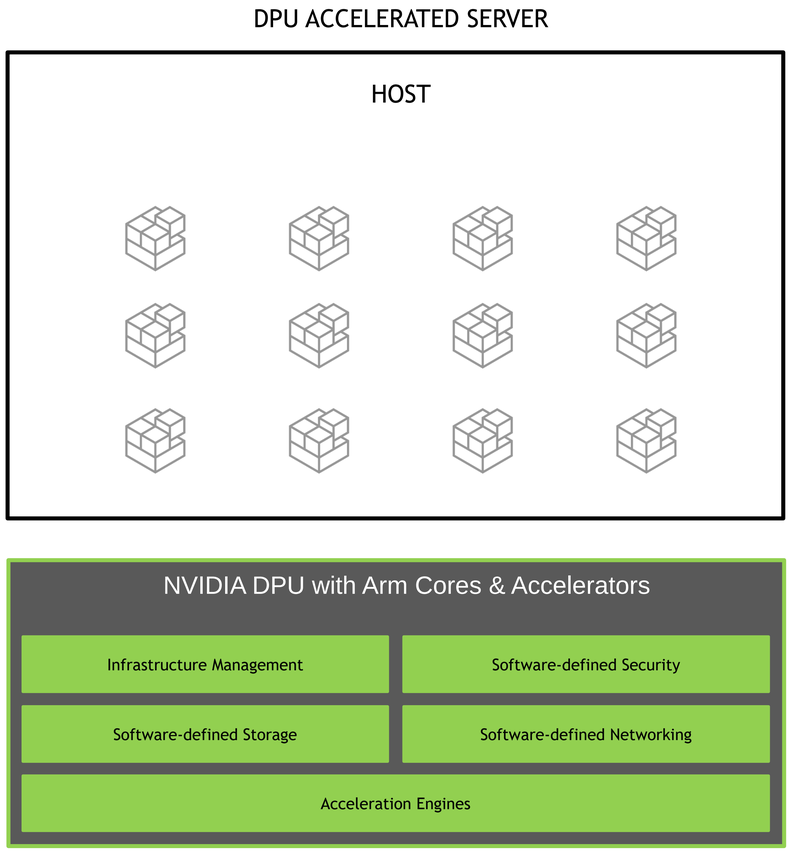
Yong Qin, Ph.D. | UCF 2023 | 12/6/2023

Data Processing Unit

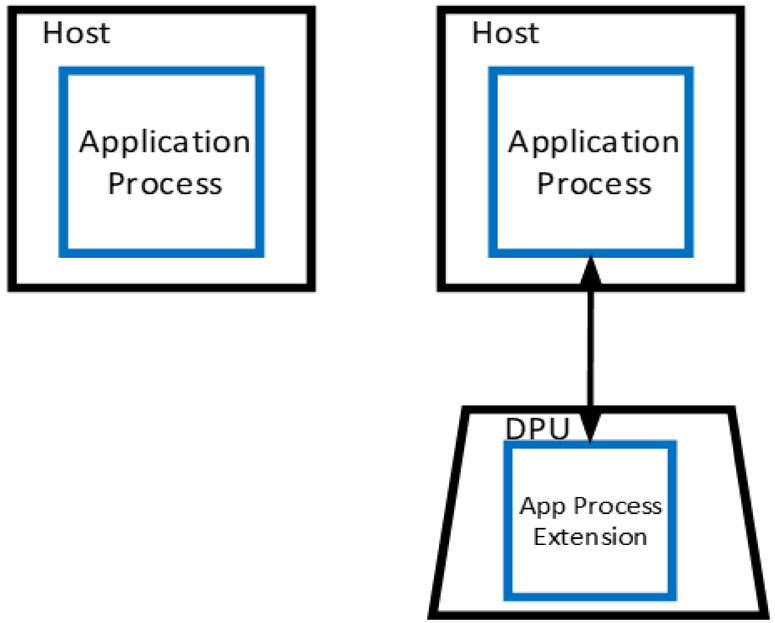


NVIDIA’s BlueField Data Processing Unit



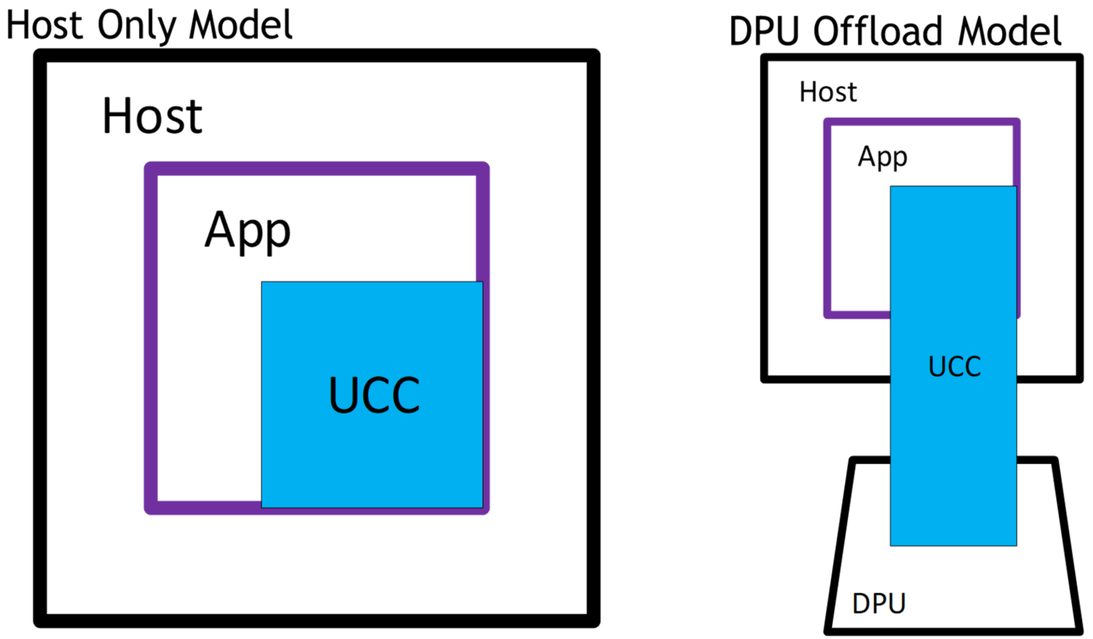


DPU Offloading Model

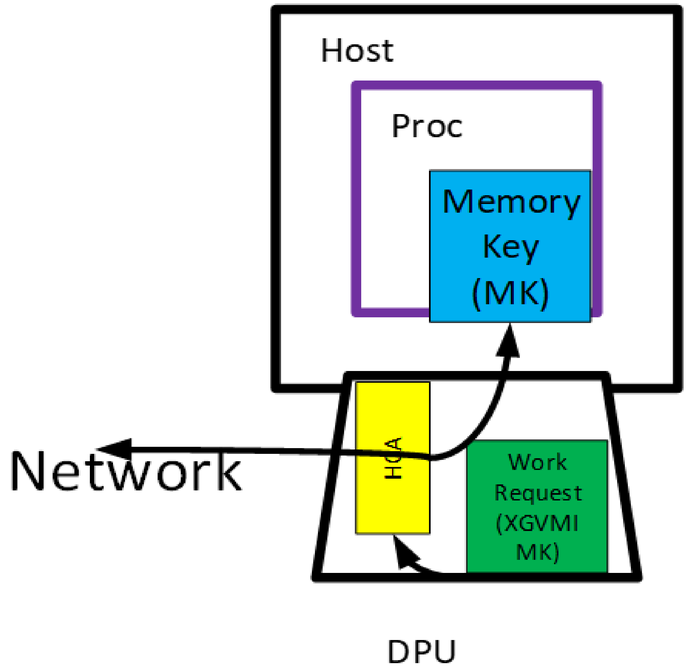


- Extension of application process – algorithms split between host and DPU
- Offload system services – storage, security, etc.
- Offload user applications – AI/ML, simulations, communications, etc.

DPU-Offloaded Unified Collective Communication (UCC)

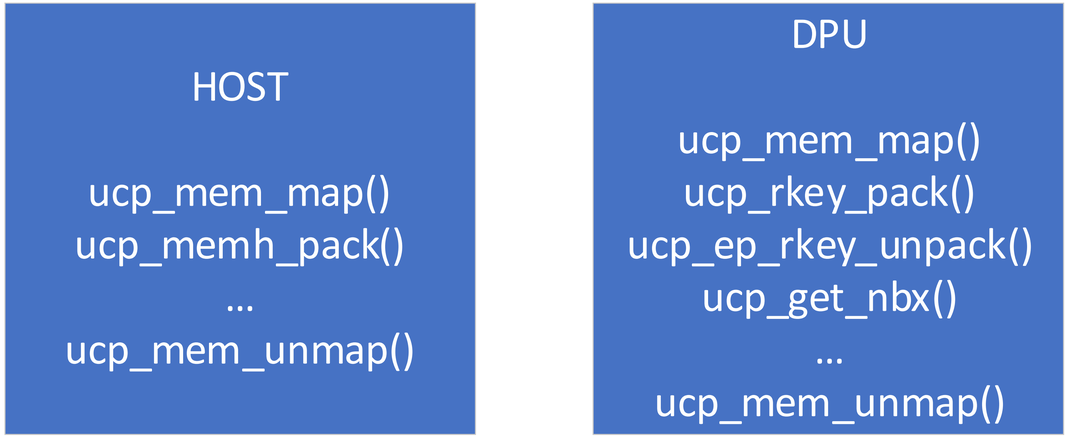


Cross-GVMI Memory Key (Mkey)



- Guest Virtual Machine ID (GVMI) or vHCA ID
- Special memory keys that allow DPU based work requests to reference host side memory
- DPU can initiate data transfer on behalf of the host w/o host involvement
  - No need to move data to the DPU before DPU can send it
  - Post receive work requests for host-resident memory

UCC Workflow with Cross-GVMI Mkey



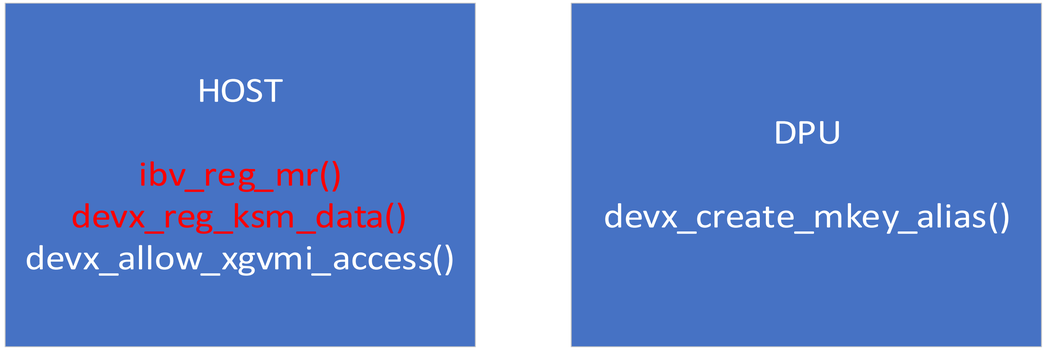
- Host map user buffer via ucp_mem_map()
- Host pack memh via ucp_memh_pack() (register target mkey if necessary) then send to DPU via HOST_ARRIVE_EVENT
- DPU map xgvmi user buffer via ucp_mem_map() (register alias mkey if necessary)
- DPU pack rkey via ucp_rkey_pack() then send to remote DPU via DPU_RTS_EVENT
- Remote DPU unpack rkey via ucp_ep_rkey_unpack() then issue RDMA_READ (ucp_get_nbx())
- At completion both end unmap user buffer via ucp_mem_unmap()

Cross-GVMI Mkey Registration in UCX (Previous/BlueField-2)

- Host register user buffer via umem registration (no ~~ibv_reg_mr~~, cost varies with buffer size, could be >110ms for 1GB)
  - Significant performance bottleneck for large user buffers
- Host create mkey with the registered user buffer (fixed cost ~160us)
- Host modify mkey to allow xgvmi access (cost varies with buffer size, in the order of 100~200us)
- DPU create mkey alias for the host mkey (fixed cost ~200us)

cost (ns)	umem			
	16 KB	1 MB	64 MB	1 GB
host umem registration	165,139	320,991	5,833,211	114,294,081
host mkey creation	158,753	159,132	158,678	160,000
host mkey export	94,778	101,124	107,185	185,765
host mkey destroy	137,371	137,560	136,885	138,362
host umem deregistration	74,027	132,564	386,428	2,473,056
dpu mkey registration	176,168	176,198	176,236	196,198
dpu mkey deregistration	149,525	149,585	149,687	159,722

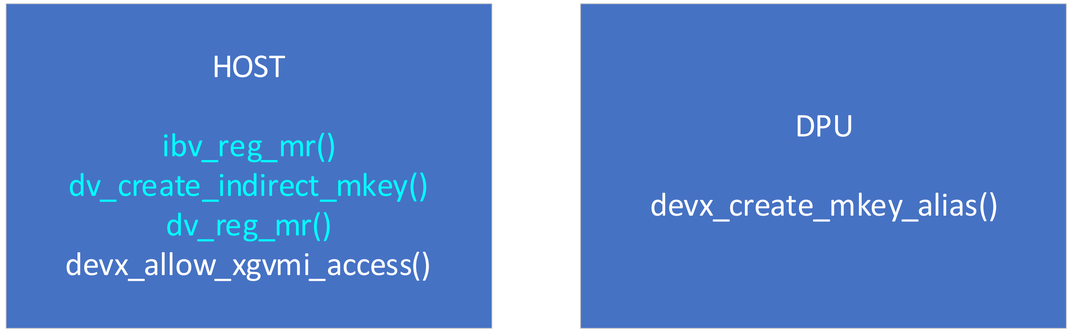
Cross-GVMI Mkey Registration in UCX (Current/BlueField-3)



- Host register user buffer via ibv_reg_mr and KSM
  - Use direct mkey to create an indirect mkey
- Host modify indirect mkey to allow xgvmi access (cost varies with buffer size, in the order of 100~200us)
- DPU create mkey alias for the host mkey and GVMI (fixed cost ~200us)

cost (ns)	indirection			
	16 KB	1 MB	64 MB	1 GB
~~host umem registration~~	~~165,139~~	~~320,991~~	~~5,833,211~~	~~114,294,081~~
host memory registration	10,125	22,296	232,007	5,823,605
host mkey creation	158,753	159,132	158,678	160,000
host mkey export	94,778	101,124	107,185	185,765
host mkey destroy	137,371	137,560	136,885	138,362
~~host umem deregistration~~	~~74,027~~	~~132,564~~	~~386,428~~	~~2,473,056~~
host memory deregistration	8,230	14,073	28,635	2,585,700
dpu mkey registration	176,168	176,198	176,236	196,198
dpu mkey deregistration	149,525	149,585	149,687	159,722

Cross-GVMI Mkey Registration in UCX (UMR/BlueField-3)

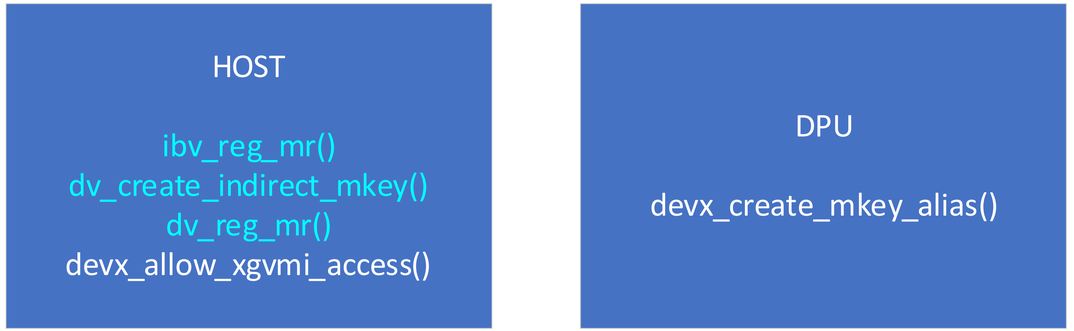


- Host register user buffer via ibv_reg_mr and User-Mode Memory Registration
- Host create indirect (UMR) mkey (fixed cost ~220us)
- Host register direct mkey to indirect mkey (fixed cost ~1us)
- Host modify indirect mkey to allow xgvmi access (cost varies with buffer size, in the order of 100~200us)
- DPU create mkey alias for the host mkey and GVMI (fixed cost ~200us)

cost (ns)	umr			
	16 KB	1 MB	64 MB	1 GB
host memory registration	10,125	22,296	232,007	5,823,605
host umr mkey creation	216,900	217,703	217,908	218,510
host umr mkey registration	1,025	1,097	1,080	1,022
host umr mkey export	138,253	147,796	147,373	201,466
host umr mkey invalidation	973	1,008	1,030	1,018
host umr mkey destroy	150,892	151,362	151,745	151,781
host memory deregistration	8,230	14,073	28,635	2,585,700
dpu mkey registration	177,270	174,919	176,509	193,689
dpu mkey deregistration	150,699	148,548	149,765	157,555

Cross-GVMI Mkey Registration in UCX (UMR Mkey Pool/BlueField-3)

Credit to: Gil Bloch@NVIDIA



- Host register user buffer via ibv_reg_mr and User-Mode Memory Registration
- ~~Host create indirect (UMR) mkey (fixed cost ~220us)~~
- Host register direct mkey to indirect mkey (fixed cost ~1us)
- ~~Host modify indirect mkey to allow xgvmi access (cost varies with buffer size, in the order of 100~200us)~~
- ~~DPU create mkey alias for the host mkey and GVMI (fixed cost ~200us)~~
- Applications use lots of temporary buffers
  - Save 800~900us per Mkey – important for P2P operations

cost (ns)	umr			
	16 KB	1 MB	64 MB	1 GB
host memory registration	10,125	22,296	232,007	5,823,605
~~host umr mkey creation~~	~~216,900~~	~~217,703~~	~~217,908~~	~~218,510~~
host umr mkey registration	1,025	1,097	1,080	1,022
~~host umr mkey export~~	~~138,253~~	~~147,796~~	~~147,373~~	~~201,466~~
host umr mkey invalidation	973	1,008	1,030	1,018
~~host umr mkey destroy~~	~~150,892~~	~~151,362~~	~~151,745~~	~~151,781~~
host memory deregistration	8,230	14,073	28,635	2,585,700
~~dpu mkey registration~~	~~177,270~~	~~174,919~~	~~176,509~~	~~193,689~~
~~dpu mkey deregistration~~	~~150,699~~	~~148,548~~	~~149,765~~	~~157,555~~

Cross-GVMI UMR Mkey Pool Architecture

- UMR dedicated QP and CQ
- UMR Mkey pool (on the host) and hash map (on the DPU)

```
/* Data structure to hold the UMR MR (on the host) item in the mkey pool */
typedef struct {
    ucs_list_link_t     super;
    struct mlx5dv_mkey  *umr_mkey;
} uct_ib_mlx5dv_indirect_mr_t;

/* Data structure to hold the UMR mkey alias (on the DPU) item in the hash map */
typedef struct {
    struct mlx5dv_devx_obj  *cross_mr;
    uint32_t                lkey;
} uct_ib_mlx5dv_indirect_alias_t;

/* Hash map of indirect mkey (from the host) to mkey alias (on the DPU) */
/* Note the hash key here is: gvmi_id << 32 | mkey (both uint32_t) */
KHASH_MAP_INIT_INT64(indirect_mkey_map, uct_ib_mlx5dv_indirect_alias_t);
```

- Dummy UMR Mkeys upfront (or lazy UMR Mkey creation at first use)
- Reuse RCACHE (both host and DPU)
  - MR found in RCACHE – reuse
  - MR not found in RCACHE
    - Host: retrieve one UMR Mkey from Mkey pool / create a new UMR Mkey
    - DPU: find the alias from hash map / create a new alias

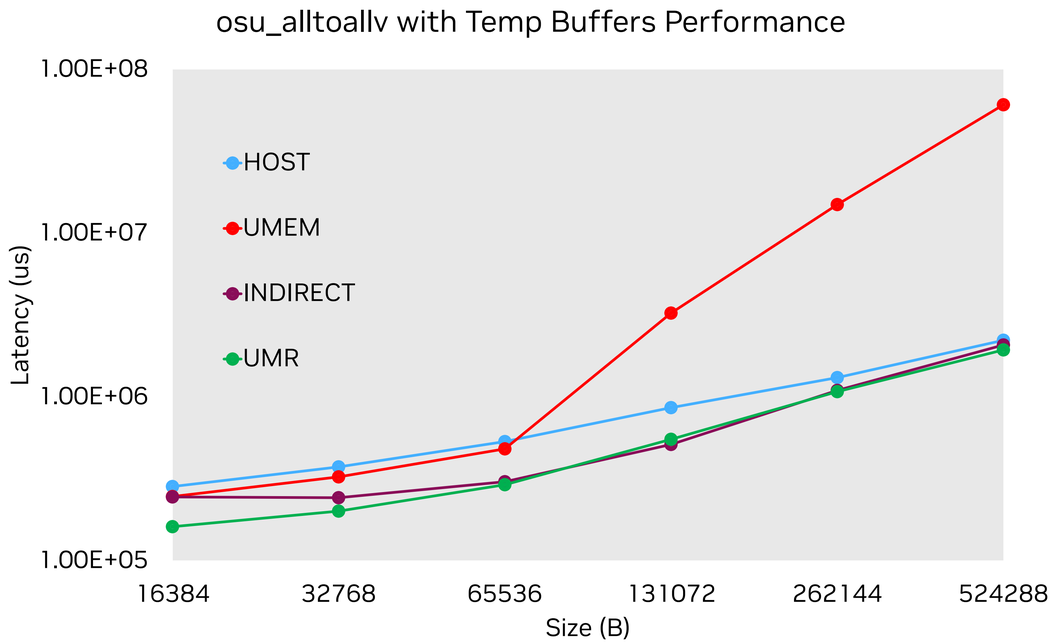
Zero-based Virtual Address Region

- UMR Mkeys can only be used in zero-based mode
- Incompatible with existing UCX Mkey management mechanism
- POC
  - Expand uct_ib_mem_t
  - Pack base address and flags for memh and rkey
  - Build WQE with proper offset using base address
  - Caveat - overlapped memory regions
- How to deal with?
  - POC – instructions to use XGVMI Mkey in RDMA operations
    - Quick to implement but POC only (not necessarily handle every corner cases)
  - New memory type
    - Longer time to implement, fully compatible with existing UCX infrastructure, but may incur overheads to register same memory region with different memory types
  - New Mkey management mechanism
    - A new architecture from bottom up, future proof, support for different transports, but touches everything in UCP & UCT
  - (No need to deal with if UMR Mkeys can be used with normal VA)

```
typedef struct uct_ib_mem {
    uint64_t                address;
    uint32_t                lkey;
    uint32_t                exported_lkey;
    uint32_t                rkey;
    uint32_t                atomic_rkey;
    uint32_t                indirect_rkey;
    uint32_t                flags;
} uct_ib_mem_t;
```

Performance Evaluation / Summary

- Modified osu_alltoallv with temporary buffers for each iteration
- 32 Nodes 32 PPN, BlueField-3, 4 SPs, 4 Sends and 4 Receives
- 34% better @16KB than current (indirect Mkey), 17% better @32KB
- Performance determined by “registration cost/communication cost”
- No overprovisioning of UMR Mkeys



XGVMI UMR Mkey Pool is a highly desired feature for HPC!