UCX GPU SUPPORT

Yossi Itigin, Nov 30 2020

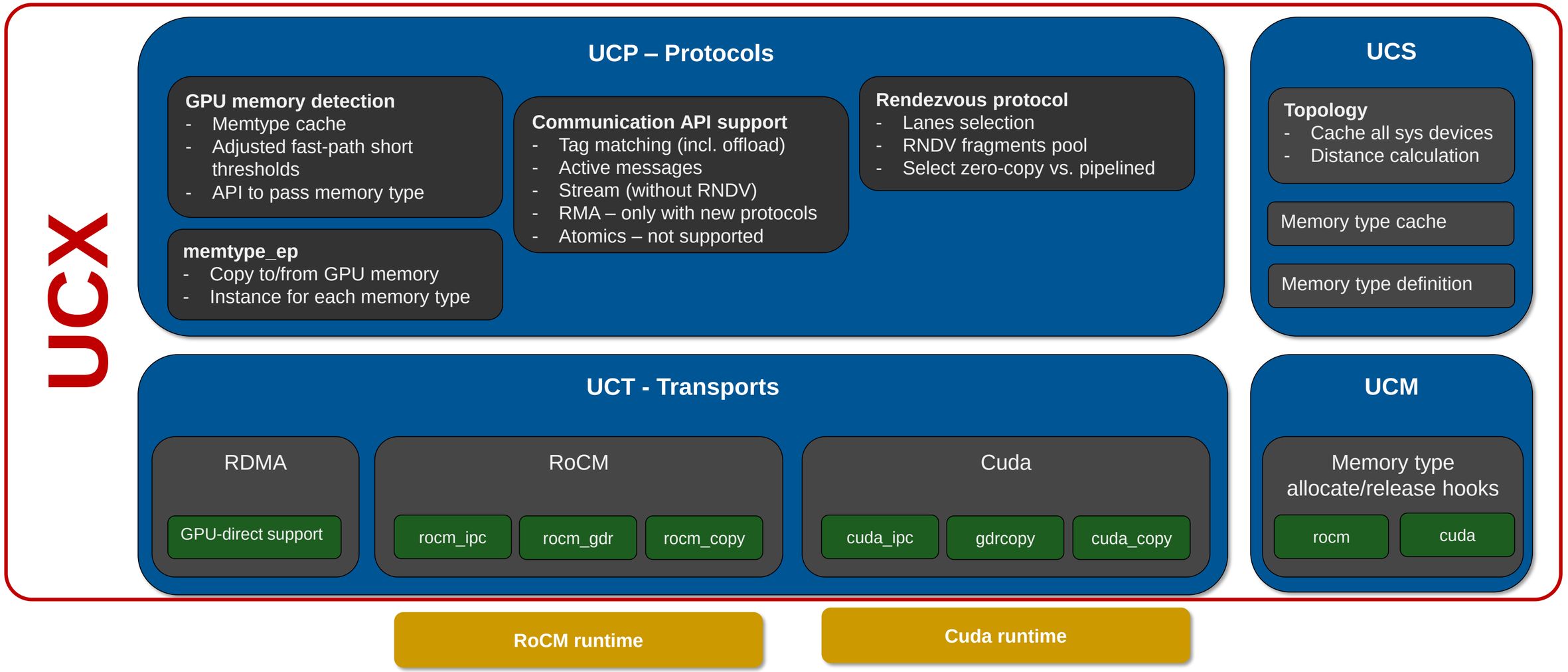
UCX GPU SUPPORT STATUS

High level goal:

Provide out-of-box support and optimal performance for GPU memory communications

- Supported GPU types: RoCM, Cuda
- Most protocols support GPU memory
- Rendezvous protocol as zero-copy and pipelined (2-stage, 3-stage)
- Memory type cache for short messages

UCX GPU SUPPORT MAP



UCX GPU SUPPORT GAPS

- Memory hooks and static link
- Out of box performance:
  - Topology detection
  - Protocol and thresholds selection
  - GPU buffers for rendezvous pipelined protocol
  - GPU stream handle in API
  - Communication/computation overlap

GPU MEMORY HOOKS

Current status:

- In some cases (e.g static link) the UCM hooks are not called
- As a result, memory type cache works incorrectly and can lead to segfault / data corruption
- This is the most common issue reported by GPU users

Proposed solution:

- Disable memtype cache by default (will hurt micro-benchmarks)
- Validate buffer ID in memory registration cache
    - How can we do this for RoCM?
- Better memory hooks?

TOPOLOGY DETECTION

Current status:

- UCT RDMA transport can report NIC locality
- UCT GPU transport can detect memory locality
- UCS topology can provide basic latency/bw estimation based on sysfs tree distance

Missing parts:

- UCS topology accurate distance (including PCIe switch presence), support external topology file for VM
- UCP to estimate performance per operation based on buffer’s locality
    - As intermediate solution, select NIC closest to active GPU – WIP

PROTOCOL/THRESHOLD SELECTION

Current status:

- Thresholds are based mostly on Host memory behavior
- Rendezvous zero copy is always preferred over pipelined

Missing parts:

- Select rendezvous protocol based on topology
- Extend UCX_RNDV_SCHEME config to select all possible options
- Select eager/rndv cutoff based on protocol (e.g for pipelined the threshold should probably be higher than zcopy)
- UCT to report performance estimations per {operation,memory-type}
    - For example, copy-from GPU can be slower than copy-to CPU

GPU STREAM HANDLE IN API

Current status:

- cuda_copy transport uses internal streams
- Requires user application to synchronize with running kernel

Missing parts:

- Allow passing external stream (from application) to be used for memory copy
- Application is responsible to avoid deadlock

Open questions:

- Can we define an opaque “stream” object in UCP API?
- Does the same problem exist for RoCM?

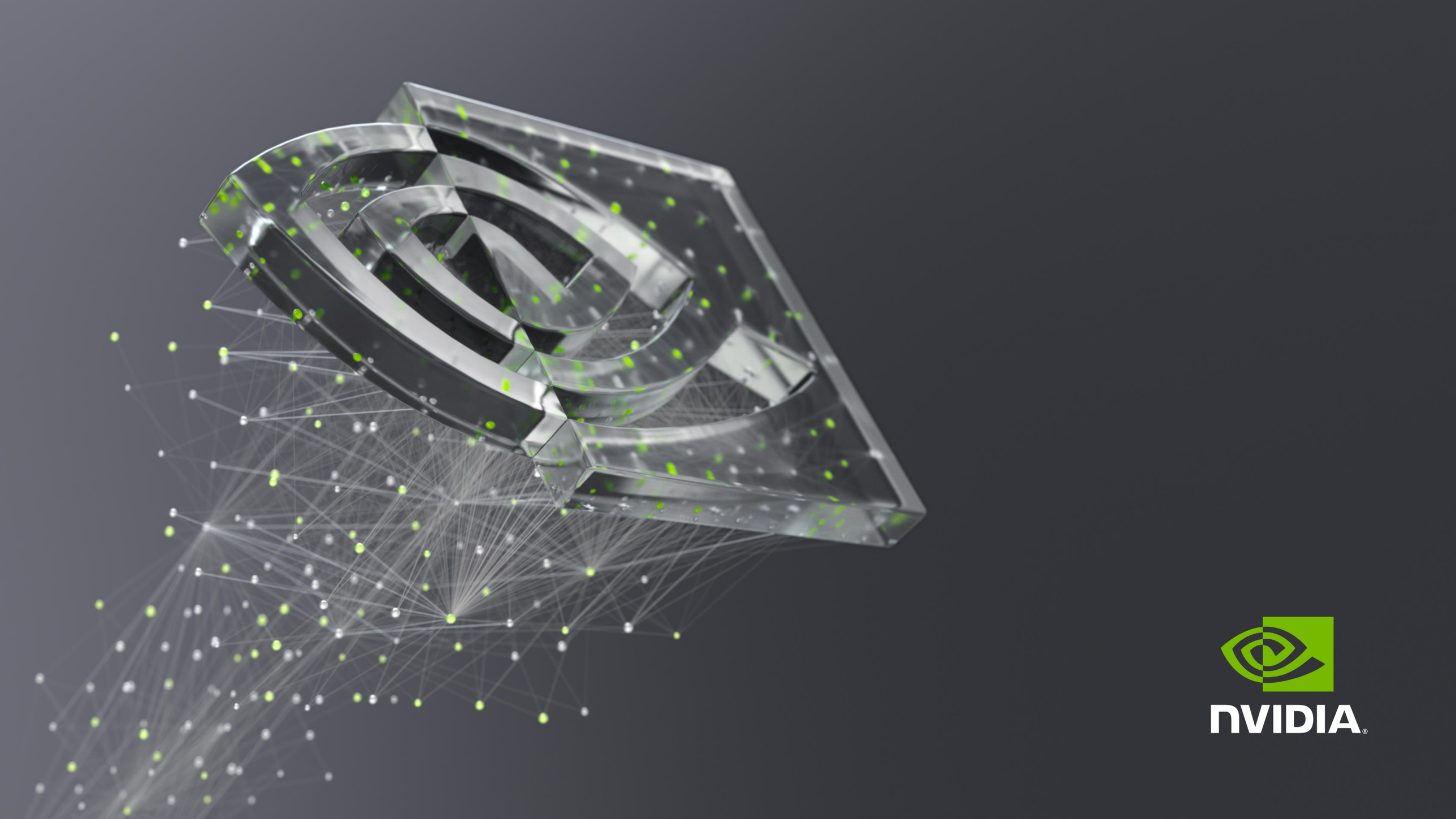
COMMUNICATION/COMPUTATION OVERLAP

Current status:

- Rendezvous pipelined protocol requires calls to ucp_progress() to work
- Does not allow overlap with compute and/or requires progress thread

Possible solution:

- Arm “wakeup” notifications from UCT, to progress on the async thread
  - When GPU copy is completed
  - When RDMA_READ/WRITE completes
  - When RTR/ATP messages arrive
- Thread safety: either require worker with MT support enabled, or create separate UCT objects


NVIDIA