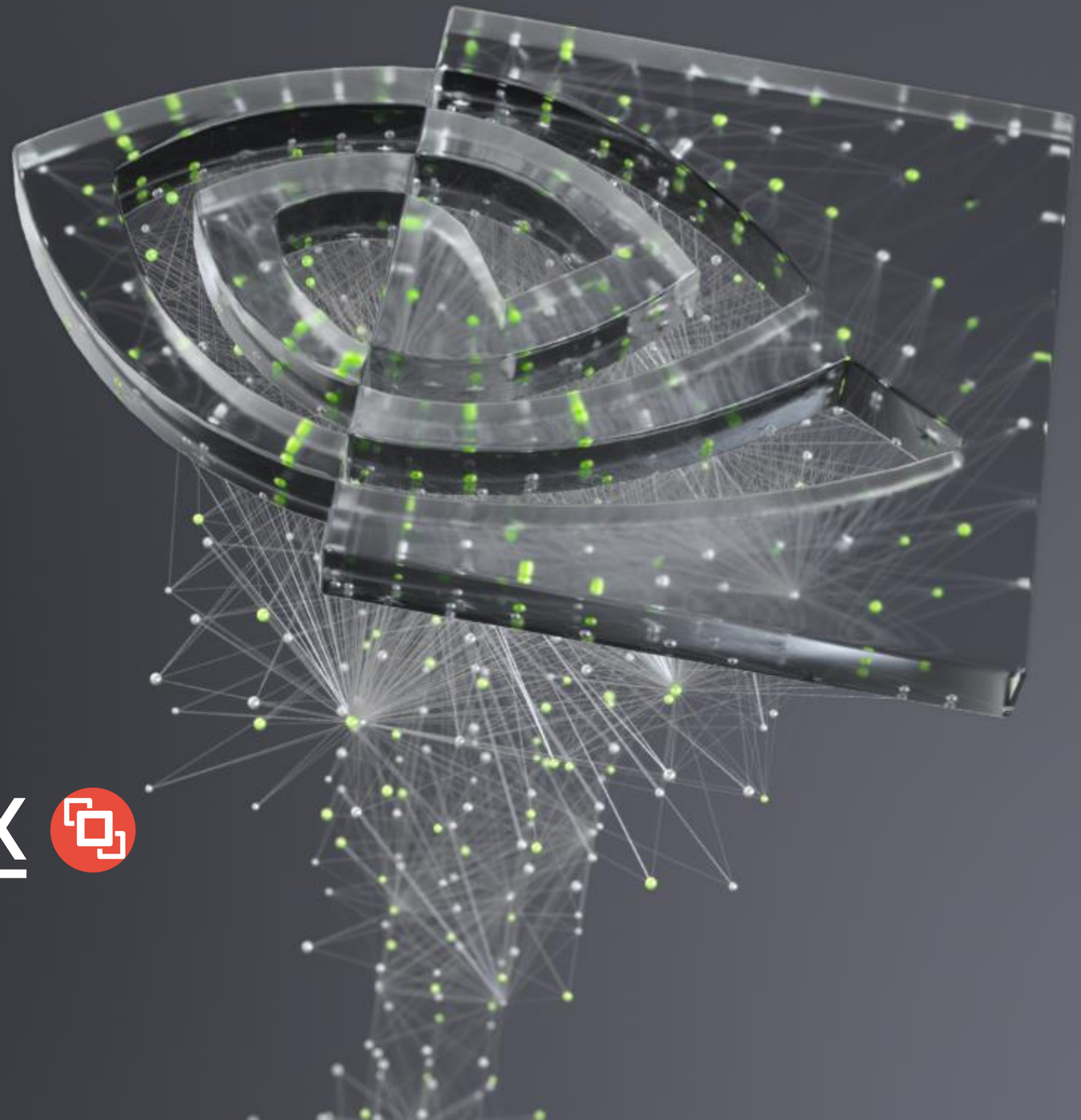


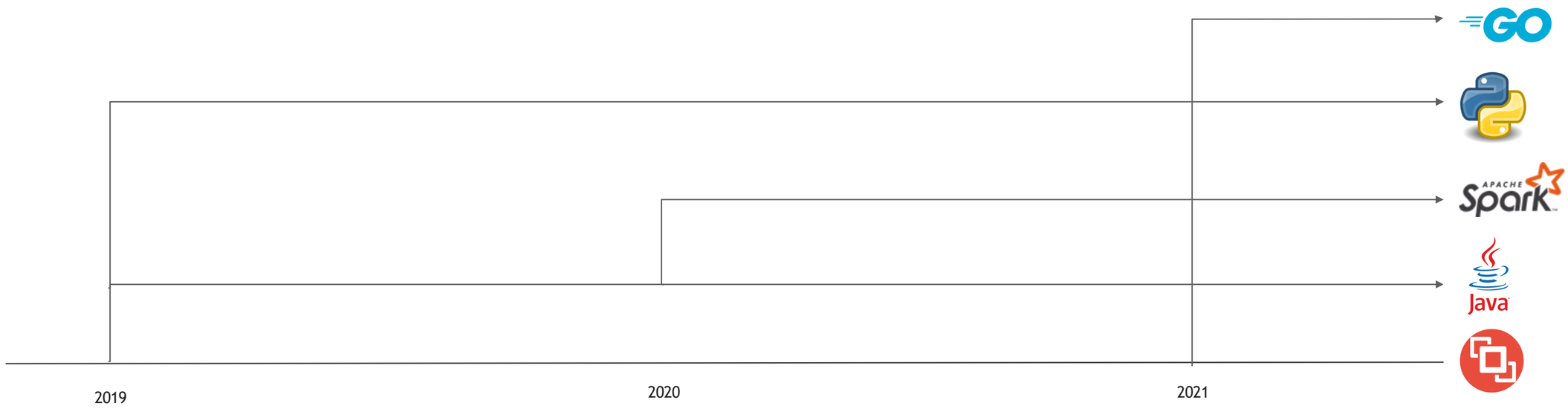
NVIDIA

GO BINDINGS FOR UCX

Peter Rudenko (prudenko@nvidia.com)

UCX BINDINGS

Bringing the power of UCX to other languages and frameworks



A language for high-performance networking and multiprocessing.

- Go was designed at Google in 2007 to improve programming productivity in an era of multicore, networked machines and large codebases. [Wiki]
- Statically linked binaries by default; therefore, all Go binaries include the Go runtime.
- Built-in concurrency and a robust standard library
- Growing ecosystem of developers, communities, and tools:

Connecting 2 worlds.

- Go is a garbage collected language, and the garbage collector needs to know the location of every pointer to Go memory.
- Go code may pass a Go pointer to C provided the Go memory to which it points. The C code must not store any Go pointers in Go memory, even temporarily.
- Because of the pointer passing rules Go code can not pass a Go function value directly to C. Instead, it is necessary to use an indirection:
    - Register go callback in map and get it's id.
    - Pass id to C ucp structs as user_data argument.
    - On callback completion pass user_data to Go static callback
    - Call user's Go callback by it's id.

```
//export ucxgo_completeGoSendRequest
func ucxgo_completeGoSendRequest(request  unsafe.Pointer, status C.ucs_status_t, callbackId unsafe.Pointer)
{
        if callback, found := deregister(uint64(uintptr(callbackId))); found {
                callback.(UcpSendCallback)(&UcpRequest{
                        request: request,
                        Status:  UcsStatus(status),
                }, UcsStatus(status))
        }
}
```

API Example

1. **Init Ucp context:**

```
context, err := NewUcpContext((&UcpParams{}).EnableWakeup().EnableAM())
```

2. Register memory:

```
memory,err := context.MemMap((&UcpMmapParams{}).SetMemoryType(UCS_MEMORY_TYPE_HOST).Allocate().SetLength(1))
```

3. Init ucp worker

```
worker, err := context.NewWorker((&UcpWorkerParams{}).SetThreadMode(UCS_THREAD_MODE_MULTI).WakeupTX())
```

4. Init listener

```
listenerParams := addr, _ := net.ResolveTCPAddr("tcp", fmt.Sprintf("0.0.0.0:%v", perfTestParams.port))
(&UcpListenerParams{}).SetSocketAddress(addr).SetConnectionHandler(func(connRequest *UcpConnectionRequest) { fmt.Printf("Got connection from %v”,
connRequest.Query(UCP_CONN_REQUEST_ATTR_FIELD_CLIENT_ID, UCP_CONN_REQUEST_ATTR_FIELD_CLIENT_ADDR))})
listener, err = worker.NewListener(listenerParams)
```

5. Create an endpoint

```
(&UcpEpParams{}).SetUcpAddress(workerAddress).SetPeerErrorHandling()
.SetErrorHandler(func(ep *UcpEp, status UcsStatus) {sender.t.Logf("Error handler called with status %d", status)})
```

6. Send data

```
ep.SendAmNonBlocking(amId, headerMem, headerLength, dataMem, dataLen, UCP_AM_SEND_FLAG_RNDV|UCP_AM_SEND_FLAG_REPLY, nil)
```

Standalone benchmark

```
$./src/tools/perf/ucx_perftest -t tag_bw -s 125000000 -c 1 jazz14
+--------------+--------------+------------------------------+---------------------+-----------------------+
|              |              |       overhead (usec)        |   bandwidth (MB/s)  |  message rate (msg/s) |
+--------------+--------------+----------+---------+---------+----------+----------+-----------+-----------+
|    Stage     | # iterations | 50.0%ile | average | overall |  average |  overall |  average  |  overall  |
+--------------+--------------+----------+---------+---------+----------+----------+-----------+-----------+
[thread 0]              223      0.469  4952.762  4952.762    24069.25   24069.25         202         202
[thread 0]              439      0.468  5089.412  5019.998    23423.00   23746.88         196         199
[thread 0]              655      0.468  5089.408  5042.887    23423.02   23639.10         196         198
```

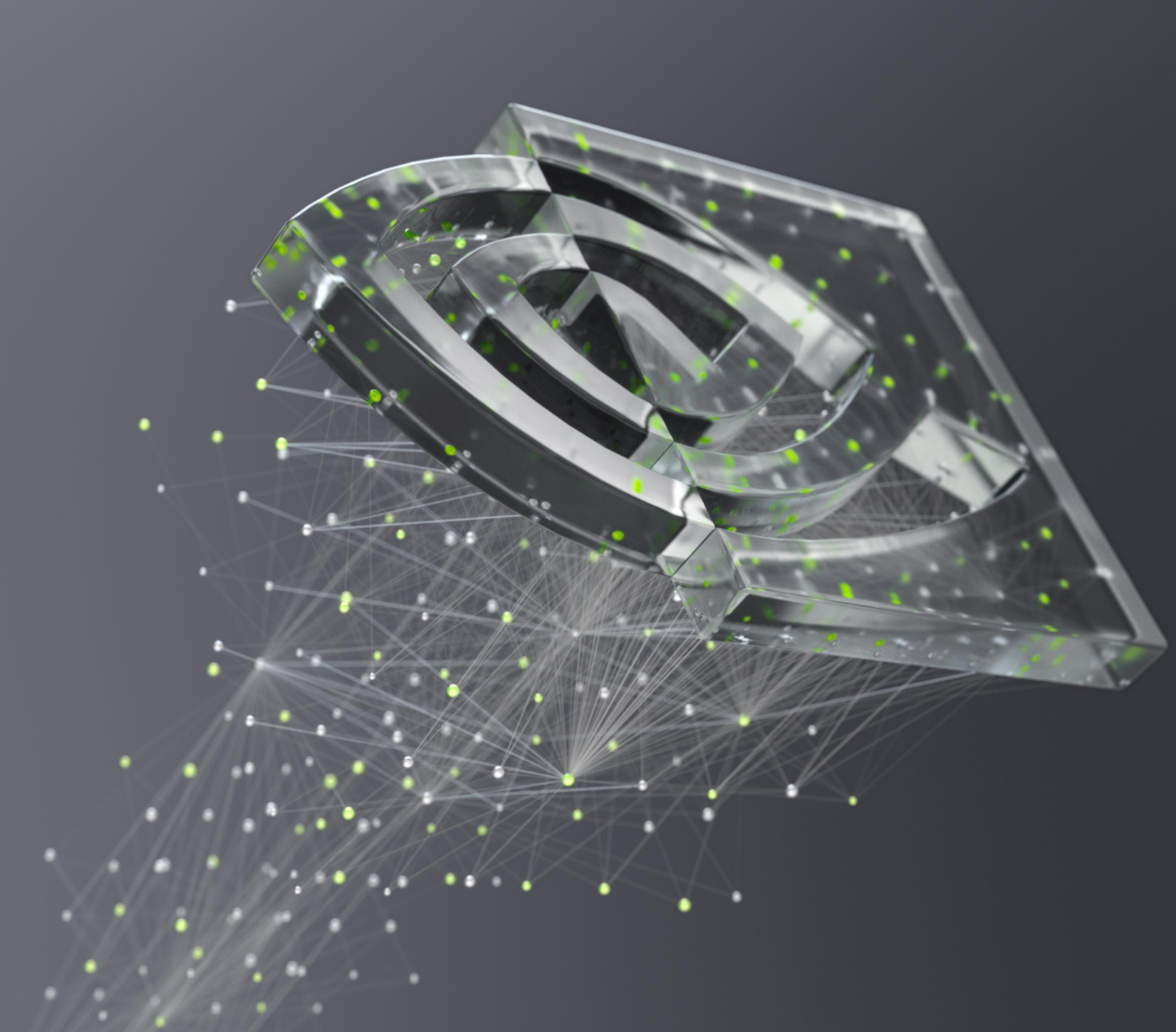
```
cd /hpc/mtr_scrap/users/peterr/devel/ucx-go/bindings/go/src/examples/perftest ;\
LD_LIBRARY_PATH=/hpc/mtr_scrap/users/peterr/devel/ucx-go/src/ucp/.libs:/hpc/mtr_scrap/users/peterr/devel/ucx-go/src/ucs/.libs:/hpc/mtr_scrap/users/peterr/devel/ucx-go/src/ucm/.libs:/hpc/mtr_
scrap/users/peterr/devel/ucx-go/src/uct/.libs: /hpc/mtr_scrap/users/peterr/devel/ucx-go/bindings/go/.libs/tmp/goperftest -s=125000000 -i=jazz14
|--------------------|--------------------|--------------------|--------------------|
|              Thread|           Iteration|             Latency |     Bandwidth (Gb/s)|
|--------------------|--------------------|--------------------|--------------------|
|                1/1|           100/1000|          5.143365ms|          194.425245|
|                1/1|           200/1000|          5.154557ms|          194.003093|
|                1/1|           300/1000|          5.155456ms|          193.969263|
|                1/1|           400/1000|          5.145187ms|          194.356396|
|                1/1|           500/1000|          5.130215ms|          194.923605|
|                1/1|           600/1000|           5.14066ms|          194.527551|
|                1/1|           700/1000|          5.137829ms|          194.634738|
|                1/1|           800/1000|          5.116431ms|          195.448742|
|                1/1|           900/1000|          5.174381ms|          193.259831|
|--------------------|--------------------|--------------------|--------------------|
Number of iterations: 1000, number of threads: 1, message size: 125000000, memory type: host, average latency (ms): 5.234, average bandwidth (Gb/s): 191.049
```

Conclusions

- Go UCX bindings provides zero overhead language friendly API for high performance networking.
- CUDA and HOST memory type supported.
- Part of UCX CI pipelines
- Beta in UCX v.1.12 release: UCP am + tag API

Next steps

- One sided API
- IOV operations



Thanks,
QA

NVIDIA