Charm++ with UCX
A focus on inter-GPU communication

Jaemin Choi
University of Illinois Urbana-Champaign

Dec 2, 2020
UCF Virtual Workshop 2020
Charm++ Messaging API
Message-Driven Execution

void foo(int a, vector<double> b, vector<double> c)

thisProxy[dest].foo(my_a, my_b, my_c);

Source object

my_a
my_b
my_c

Source node’s host memory

Destination object
Charm++ Messaging API

Parameter Marshalling

thisProxy[dest].foo(my_a, my_b, my_c);

void foo(int a, vector<double> b, vector<double> c)

Source object

Message

Destination object

Source node’s host memory

Marshalled payload

Metadata

Contains info for message-driven execution, including dest object ID, registered method ID, ...

Copy
Charm++ Messaging API
Parameter Unmarshalling

- Message is unmarshalled (unpacked) on the receiver
- Metadata is used to invoke the correct method on the destination object, e.g. `foo()`
- Destination object can access the received data (`a`, `b`, `c`)
Charm++ GPU Messaging
Built on Zero Copy EM Post API\textsuperscript{1} & UCX

```c
void foo(int a, nocopydevice double* b, nocopydevice double* c)
```

```
thisProxy[dest].foo(my_a, my_b, my_c);
```

Source object

Destination object

Source node's GPU memory

\textsuperscript{1} https://charm.readthedocs.io/en/latest/charm++/manual.html#zero-copy-entry-method-post-api
Charm++ GPU Messaging
Parameter Marshalling & UCX Send

void foo(int a, nocopydevice double* b, nocopydevice double* c)

thisProxy[dest].foo(my_a, my_b, my_c);

Source object

Message

Source node's GPU memory

my_a

my_b

my_c

Metadata

ucp_tag_send_nb

6/15
Charm++ GPU Messaging
Parameter Unmarshalling & UCXRecv

```
void foo(int a, nocopydevice double* b, nocopydevice double* c);
```

```
thisProxy[dest].foo(my_a, my_b, my_c);
```

Source object

```
Message
```

Call

Unpack

Destination node’s GPU memory

```
ucp_tag_recv_nb
```

```
ucp_tag_recv_nb
```

Destination object

```
a
```

```
b
```

```
c
```
Preliminary Performance
Evaluation Platform/Configurations

- OLCF Summit
- OSU latency benchmark with GPU
  source/dest buffers (MPI, Charm++ versions)
- IBM Spectrum MPI v10.3.1.2
- UCX v1.9.0
- Inter-process (single-node)
- Inter-node
Preliminary Performance
Inter-process OSU Latency Benchmark

![Graph showing latency for different message sizes and implementations.](image-url)
Preliminary Performance

Inter-process OSU Latency Benchmark

Profiled with nvprof

- CUDA-aware MPI
- Charm-Host
- Charm-UCX

D2H/H2D memcpy

CUDA IPC

Used when message size > UCX_RNDV_THRESH
Hangs when UCX_RNDV_THRESH < 64
Preliminary Performance
Inter-node OSU Latency Benchmark

![Graph showing latency benchmark results for different message sizes and communication libraries.]

- CUDA-aware MPI
- Charm-Host
- Charm-UCX

Latency (us)

Message size (1, 2, 4, 8, 16, 32, 64, 128, 256, 512 KB, 1 MB, 2 MB, 4 MB)
Preliminary Performance
Inter-node OSU Latency Benchmark

Profiled with nvprof

- D2H/H2D memcpy, pipelined copies with size >= 1MB
- 2x more memcpy (bug?)

GPU activities: 2 cudaMemcpy calls (GPUDirect RDMA?)

Host-staged

Message size
Conclusion
Charm++ with UCX

• Relatively easy experience to integrate GPU-enabled UCX in Charm++
  • Single implementation for intra-node & inter-node

• Similar designs possible with other task-based runtime systems
  (esp. those that require metadata in host memory)

• Need to investigate UCX behavior and improve performance

• Plan to be included in next major Charm++ release
Thank you!

Please feel free to contact me at jaemin@acm.org.
Appendix
UCX/Charm++ Build Commands on OLCF Summit

• Build libcheck ([https://github.com/libcheck/check](https://github.com/libcheck/check))
  • Modify [libcheck_install_path]/lib64/pkgconfig/check.pc to use lib64 instead of lib

• Build GDRCopy ([https://github.com/NVIDIA/gdrcopy](https://github.com/NVIDIA/gdrcopy))
  • git checkout 2.1
  • mkdir install
  • PKG_CONFIG_PATH=[libcheck_install_path]/lib64/pkgconfig make PREFIX=[gdrcopy_path]/install CUDA=[cuda_install_path] all install

• Build UCX ([https://github.com/openucx/ucx](https://github.com/openucx/ucx))
  • git checkout v1.9.0
  • ./autogen.sh
  • mkdir build & & mkdir install & & cd build
  • ../contrib/configure-release --prefix=[ucx_path]/install --with-cuda=[cuda_path] --with-gdrcopy=[gdrcopy_install_path]
  • make -j & & make install

• Build PMIx ([https://github.com/openpmix/openpmix](https://github.com/openpmix/openpmix))

• Build Charm++ ([https://github.com/UIUC-PPL/charm](https://github.com/UIUC-PPL/charm))
  • ./buildold charm++ ucx-linux-ppc64le smp cuda ompipmix -j -g --with-production -basedir=[pmix_install_path] --basedir=[ucx_install_path]
  • export LD_LIBRARY_PATH=[ucx_install_path]/lib:$LD_LIBRARY_PATH