OpenSHMEM-UCX: Evaluation of UCX for implementing OpenSHMEM Programming Model
Languages R&D
Matthew Baker, Ferrol Aderholdt, Manjunath Gorentla Venkata, Pavel Shamis (ARM)

August 3rd, 2016
UCX

- Unified Communication layer X
- Framework for communication
- Portable performance
- Scalability
Why UCX is good for PGAS

• UCX developed in close relation with OpenSHMEM team

• Performance Oriented
  – UCX a light weight layer over hardware
  – OpenSHMEM should be a lightweight layer over UCX

• Portable
  – Targeting upper layer UCX protocol allows easy portability with low level transports
UCX layers

• UCT
  – Supports specific hardware
  – Currently supported: Cray UGNI, Mellanox IB, shared memory
  – Abstractions
    • Memory domain (md)
      – Memory registration
    • Interface (iface)
      – A particular interface to a network device
    • End Point (ep)
      – A particular point to point communication object
  • Worker
    – ensures that underlying network hardware makes progress
UCX layers

- UCP
  - Provides protocol abstractions.
  - Abstracts away common high level tasks
    - Selects best mechanism for transfers
    - Wiring up end points
    - Selecting transports
    - Tagged messages
  - Implements missing features
    - 32 bit atomics on Gemini
- Workers
  - Provides application contexts, one application can have multiple progress threads
UCX layers

• UCS
  – Common platform services
    • Local atomics
    • Thread safety
  – Data structures
  – Debug and logging output

• UCM
  – Memory management
  – Manages registration caches
UCX on Cray hardware

• UCT on Cray has 3 different transports
  – FMA and BTE APIs in RDMA transport
  – SMSG short messages in SMSG transport
  – Datagrams in UDT transport

• Each exposes different capabilities in UCT
  – RDMA exposes RMA bcopy/zcopy
  – SMSG exposes active messages
  – UDT exposes active messages on interfaces
OpenSHMEM on UCX on Cray

• OpenSHMEM only requires the RDMA transport.
  – UCX will, on its own, only initialize RDMA and skip SMSG

• Nothing special required, other than passing in correct feature flags
OpenSHMEM on UCX

- OpenSHMEM becomes thin on UCX
- Atomics and put/gets done directly through UCP.
- Collective operations and handled by OpenSHMEM
- OpenSHMEM handles memory, but UCX has hardware specific knowledge
  - OpenSHMEM registers memory, receives keys
  - UCX worries about how to register memory with driver
Benchmark results
Test System

- Titan
  - 18,688 nodes
  - 16 core AMD cpu per node
  - 32 GB of memory per node
  - Nvidia K20X accelerator
Modified OSU benchmark

• Ping pong test
  – Bouncing puts between two nodes
Ping pong test (lower is better)
OSU message rate

• Message rate benchmark from OSU benchmark suite

• Graphed results for 8 byte puts between two nodes, going from 1 PE per node to 16
Message Rate

The graph shows the message rate for two different systems: UCX and Cray. The message rate is measured in (log) Messages/s as the number of processing elements (PEs) per node increases from 0 to 16.

- UCX has a higher message rate compared to Cray, especially as the number of PEs per node increases.
- The message rate for both systems increases exponentially with the number of PEs per node.
SHOMS tests

• SHOMS is a OpenSHMEM microbenchmark suite
  – Tests cover entire OpenSHMEM API

• Test results
  – One origin PE sending to 4095 other PEs
  – Latency from 4B to 1MB
  – Bandwidth from 4B to 1MB
SHOMS latency test

![Graph showing latency test results for UCX put, Cray put, UCX get, and Cray get. The x-axis represents data sizes in bytes (4B, 128B, 4KB, 128KB), and the y-axis represents (log) micro seconds.]
SHOMS bandwidth

The graph shows the bandwidth performance of UCX put and Cray put, UCX get, and Cray get. The x-axis represents the data size in bytes (4B, 128B, 4KB, 128KB), and the y-axis represents the logarithm of bytes per second (log Bytes/seconds). The performance varies with different data sizes and methods, indicating the efficiency of each method at different scales.
SSCA1 Smith-Waterman

- Genetic local alignment benchmark
- Large parallel inner loop with many short puts and gets
  - 7 gets and 5 puts
- Extremely sensitive to message rates
SSCA1 up to 2048 PEs

![Graph showing performance comparison between UCX and Cray](chart.png)
SSCA1 16k PEs
Conclusions

- Implementing OpenSHMEM over UCX is lightweight
- UCX provides useful abstractions for high performance and high throughput
  - Workers, interfaces, endpoints arbiters
- Evaluation of microbenchmarks and kernels prove usefulness
  - Short put and get message rates
  - Still work to be done (improved management of memory registrations)
Acknowledgements

This work was supported by the United States Department of Defense (DoD) and used resources of the Computational Research and Development Programs at Oak Ridge National Laboratory.