Why Another OpenSHMEM Implementation?

- Open source implementation
  - Vehicle for research
  - Platform for developing new OpenSHMEM extensions
- High-performance implementation
  - Achieve performance and scalability close to vendor implementation
- Implementation on Cray
  - Many scalable machines are Cray systems
OpenSHMEM-UCCS Architecture Diagram

RTE
- ORTE
- STCI
- SLURM
- ALPS/CRAY

Open SHMEM

UCCS
- VERBs
- uGNI
- Shared Memory
- Hardware Direct

Driver

Hardware
UCCS: Towards a Unified Network Abstraction for Parallel Programming Models, I/O, and Big Data

- UCCS is a low-level network API specification and implementation
- Driven by the requirements of
  - PGAS, Message Passing, and Hybrid Programming Models
  - I/O Systems
- Abstracts many network interfaces
  - InfiniBand, Gemini/Aries, Shared Memory
OpenSHMEM-UCCS Design: Uses UCCS and uGNI

• Leverages UCCS capabilities
  – Runtime system, component architecture,
  – Code reusability

• uGNI vs. DMAPP
  – uGNI enables development of programming models other than PGAS e.g. MPI
  – DMAPP design focuses on PGAS programming models
OpenSHMEM-UCCS : Operations Supported

• Memory Semantics
  – SHMEM PUT/GET
  – Small Message PUT/GET: Leverage uGNI’s FMA mechanism
  – Large Message PUT/GET: Leverage uGNI’s BTE mechanism

• Atomics
  – 64-bit only
    • `shmem_llong_fadd()`
    • `shmem_llong_finc()`
    • `shmem_llong_add()`
    • `shmem_llong_inc()`
    • `shmem_llong_cswap()`
shmem_get: Protocol to Handle Unaligned Data

The diagram illustrates the protocol for handling unaligned data in a shared memory environment between processes P-0 (remote) and P-1 (local). The processes communicate through buffers and perform operations such as `PostFma()` or `PostRdma()` to synchronize and transfer data. The `memcpy()` function is used to copy data between the buffers. The diagram shows the flow of data and operations over time.
Experimental Testbed

- Chester HPC system
  - 2 X 6-Core AMD Opteron(tm) Processor
  - 16GB RAM / node
  - Cray GEMINI interconnect

- Benchmark
  - OSU Micro Benchmarks v4.2

- Put/Get Message Size
  - Small, < 1024 bytes
  - Medium, > 1024, < 64K bytes
  - Large, > 64K bytes
OpenSHMEM-UCCS' `shmem_put` Performs Similar to Cray's SHMEM

![Graph showing latency vs. transfer size](image URL)
OpenSHMEM-UCCS’ `shmem_put/get` Performs Similar to Cray’s SHMEM

Latency (usec) vs Transfer Size (bytes)

- Cray/7.0.0
- UCCS/OSHMEM
Performance Results: Latency of shmem_get Operation

- Cray, shmem_get64()
- UCCS, shmem_getmem()
- Cray, shmem_getmem()

Latency (usec)
Transfer Size (bytes)

Cray, shmem_getmem()
UCCS, shmem_getmem()
Performance Results: Latency of AMO Operations

Cray - UCCS

Operations:
- llong_fadd
- llong_finc
- llong_add
- llong_inc
- llong_cswap
Summary

• OpenSHMEM prototype implementation using UCCS and uGNI libraries
  – Performance characteristics similar to Cray SHMEM with more optimizations required in certain code paths

• Demonstration of usability of uGNI interface for PGAS programming model

• Implement missing operations
  – 32bit, 64bit shmem_swap() atomic operations

• More evaluation
  – Application analysis
  – Scalability
  – Aries
  – Compare with similar implementations
Acknowledgements

This work was supported by the United States Department of Defense & used resources of the Extreme Scale Systems Center at Oak Ridge National Laboratory.
Results (Get)

Latency (usec)

Transfer Size (bytes)

128k  256k  512k  1m  2m  4m  8m

Cray, shmem_getmem()
UCCS, shmem_getmem()
Cray, shmem_get64()