UCCS FOR OPENSHMEM

Presenters: Aaron Welch, Pavel Shamis, Steve Poole

Material: Aaron Welch, Pengfei Hao, Deepak Eachempati, Swaroop Pophale, Tony Curtis

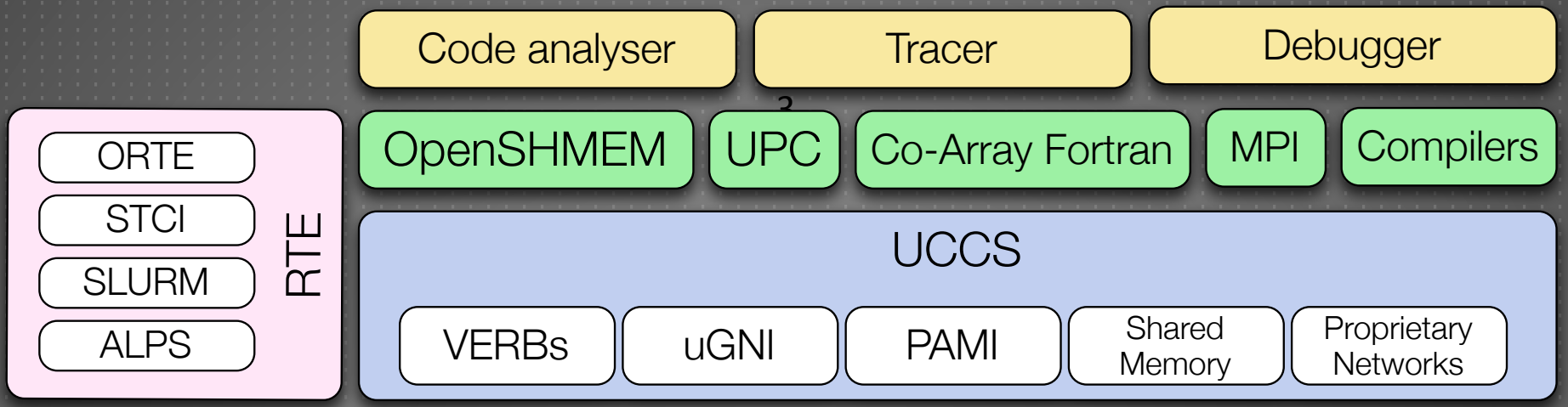
University of Houston, Texas

http://www.uh.edu/

UCCS – UNIVERSAL COMMON COMMUNICATION SUBSTRATE

UCCS OVERVIEW

- Abstract communication and networking implementation details
- Support multiple transport layers/programming models
- Provide single consistent interface
- Communication decoupled from run-time environment



MOTIVATION

- Efficient communication expected to become increasingly important
- What underlying technologies will prevail is unclear
- Provide general but low-level interface capable of supporting current or future models

USABILITY GOALS

- Increase code reuse, decrease complexity of network backend
- Support for multiple communication libraries
- Tight integration to foster support for languages and tools

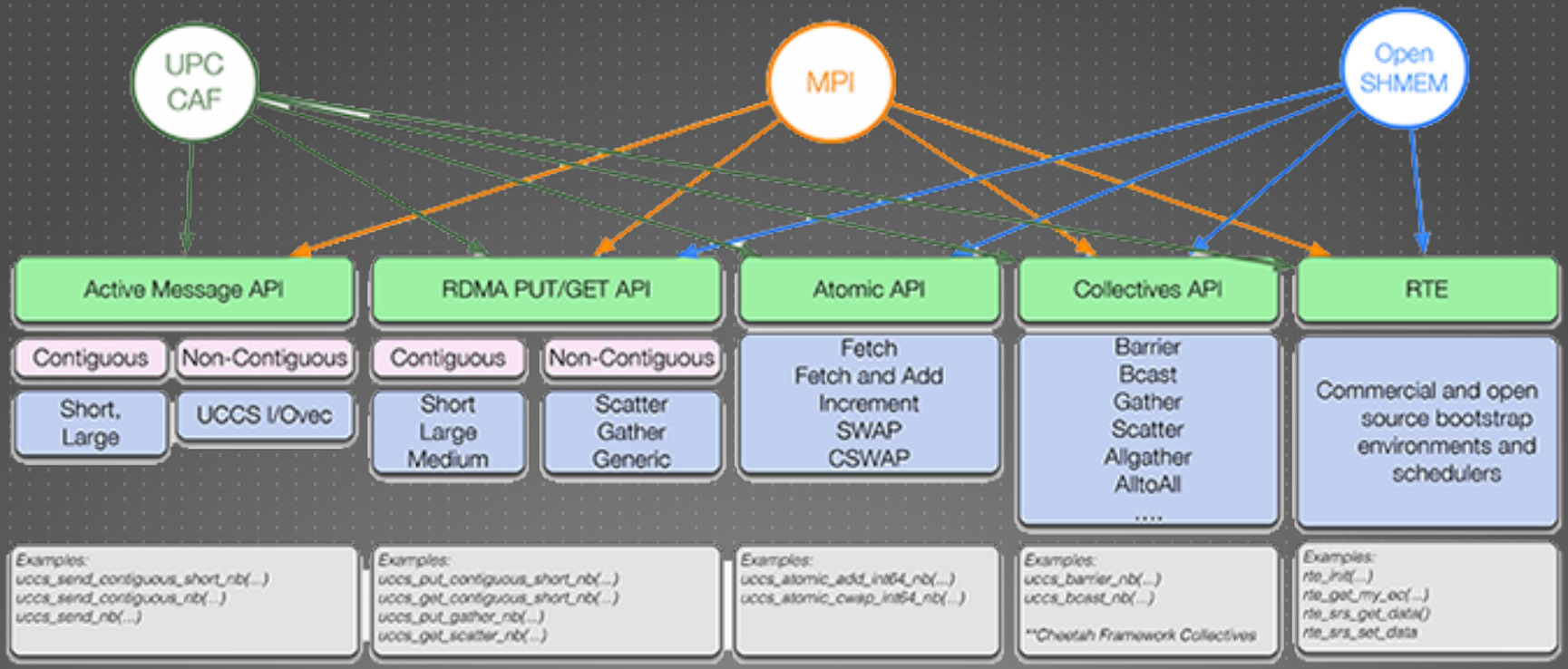
UCCS DESIGN

- Designed for low overhead, scalability, and resiliency
- Maintain minimal footprint with emphasis on reducing the critical path by operating very close to the hardware
- Three different sizes for puts and gets (short, medium, and large) using different methods for handling network requests
- Emphasis on non-blocking calls using request handles
- Support for atomic operations and low-level collectives

UCCS DESIGN

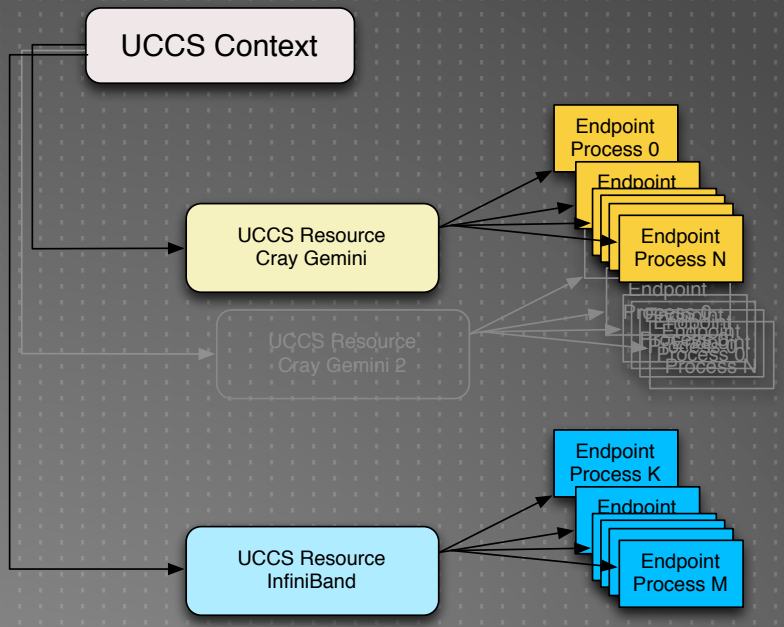
- Capabilities interface allows for querying of hardware support details
- Connectivity maps allow for heterogeneous network systems
- Dynamic memory registration for multiple remotely accessible regions
- Support for active messages

UCCS DESIGN


UPC
CAF
MPI
Open
SHMEM
Active Message API
RDMA PUT/GET API
Atomic API
Collectives API
RTE
Contiguous
Non-Contiguous
Contiguous
Non-Contiguous
Short,
Large
UCCS I/Ovec
Short
Large
Medium
Scatter
Gather
Generic
Fetch
Fetch and Add
Increment
SWAP
CSWAP
Barrier
Bcast
Gather
Scatter
Allgather
AlltoAll
....
Commercial and open source bootstrap environments and schedulers
Examples:
uccs_send_contiguous_short_nb(...)
uccs_send_contiguous_nb(...)
uccs_send_nb(...)
Examples:
uccs_put_contiguous_short_nb(...)
uccs_get_contiguous_short_nb(...)
uccs_put_gather_nb(...)
uccs_get_scatter_nb(...)
Examples:
uccs_atomic_add_int64_nb(...)
uccs_atomic_cwap_int64_nb(...)
Examples:
uccs_barrier_nb(...)
uccs_bcast_nb(...)
**Cheetah Framework Collectives
Examples:
rte_init(...)
rte_get_my_ec(...)
rte_srs_get_data()
rte_srs_set_data

UCCS CONCEPTS

- Contexts provide communication scope and isolation
- Resources represent available communication channels for a network
- Endpoints are the destinations reachable over a particular resource



RTE DESIGN

- Handles bootstrapping and other dynamic aspects of the run-time environment
- Provides out-of-band support
- Abstracted under a single interface to support multiple RTE backends including ORTE, STCI, SLURM, and ALPS

MODULAR COMPONENT ARCHITECTURE

- Allows for multiple components to be plugged in or swapped out
- Interface allows for easy creation of custom components
- Licensing support extends to development of proprietary components

TRANSPORT LAYERS

- Multiple options including InfiniBand, uGNI, and support for intra-node communication
- Can be dynamically selected, mixed, and matched
- Allows support for hybrid network systems, multi-rail
- Integrates with capabilities for transport priority when multiple routes available

WHAT UCCS ALLOWS

- Library implementers can easily support a wide array of network technologies and configurations
- Consistent interface across multiple interconnects and communication libraries
- Hybrid development
- Heterogeneous systems

USER ENVIRONMENT

- Communication Library
- UCCS
- libRTE
- RTE backend (ORTE, STCI, etc)

EXAMPLE: SHMEM_INT_ADD()

```
shmem_int_add(void *target, void *value, size_t nbytes, int pe) {
    uccs_request_handle_t desc;
    uccs_status_t status;
    resource = select_highest_priority_resource(pe);
    dest = translate_target_to_remote_address(target);
    find_memory_registration_for_dest_on_pe(dest, pe, &rem_reg);
    uccs_atomic_add_int64_nb(resource, endpoint, dest, &rem_reg,
                             value, 255, &desc);
    uccs_wait(desc, &uccs_status);
}
```

EXAMPLE: SHMEM_INT_P()

```
void shmem_int_p(int *target, int value, int pe) {
    uccs_request_handle_t desc;
    dest = translate_target_to_remote_address(target);
    uccs_put_contiguous_short_nb(resource, endpoint, dest,
            &value, &remote_reg, sizeof(int), 0, &desc);
    uccs_wait(desc, &status);
}
```


16

EXAMPLE: SHMEM_INT_PUT()

```
void shmem_int_put(int *target, const int *source, size_t len, int pe) {
    uccs_request_handle_t desc;
    dest = translate_target_to_remote_address(target);
    if (len <= comms[pe].short_put_size)
        uccs_put_contiguous_short_nb(resource, endpoint, dest,
                                     source, remote_reg, len, 0, desc);
    else if (len <= comms[pe].medium_put_size)
        uccs_put_contiguous_medium_nb(resource, endpoint, dest, source,
                                      remote_reg, NULL, len, 0, desc);
    else
        uccs_put_contiguous_large_nb(resource, endpoint, dest, source,
                                     remote_reg, NULL, len, 0, desc);
    uccs_wait(desc, &status);
}
```

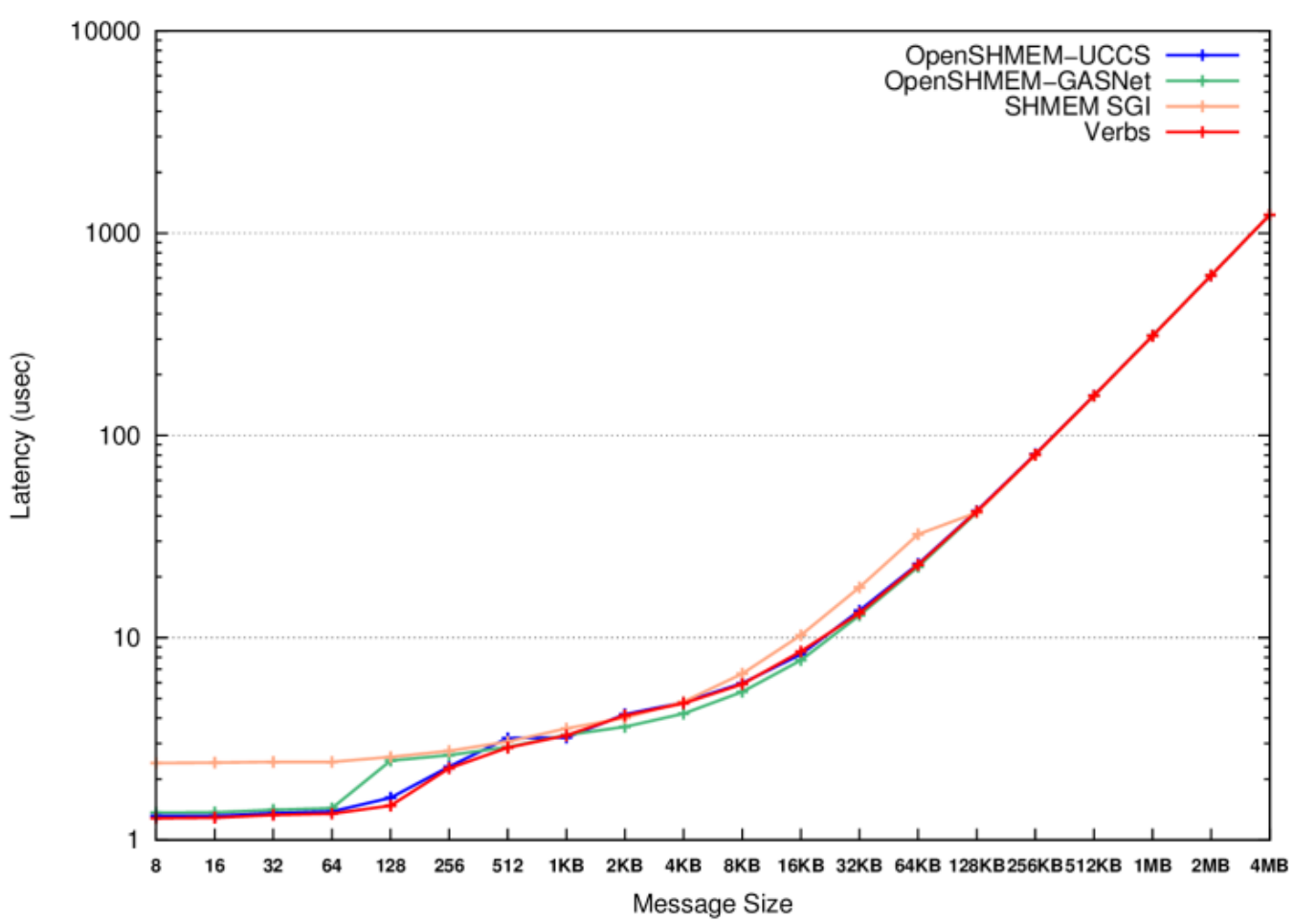
TESTING ENVIRONMENT

- OpenSHMEM reference implementation 1.0e
- GASNet v1.20.2
- Pre-production version of UCCS based on v0.2 of UCCS specification
- SGI Altix XE1300 system with 12 nodes with two Intel Xeon X5660 CPUs
- InfiniBand interconnect using Mellanox ConnectX-2 QDR HCA
- SGI MPT v2.03

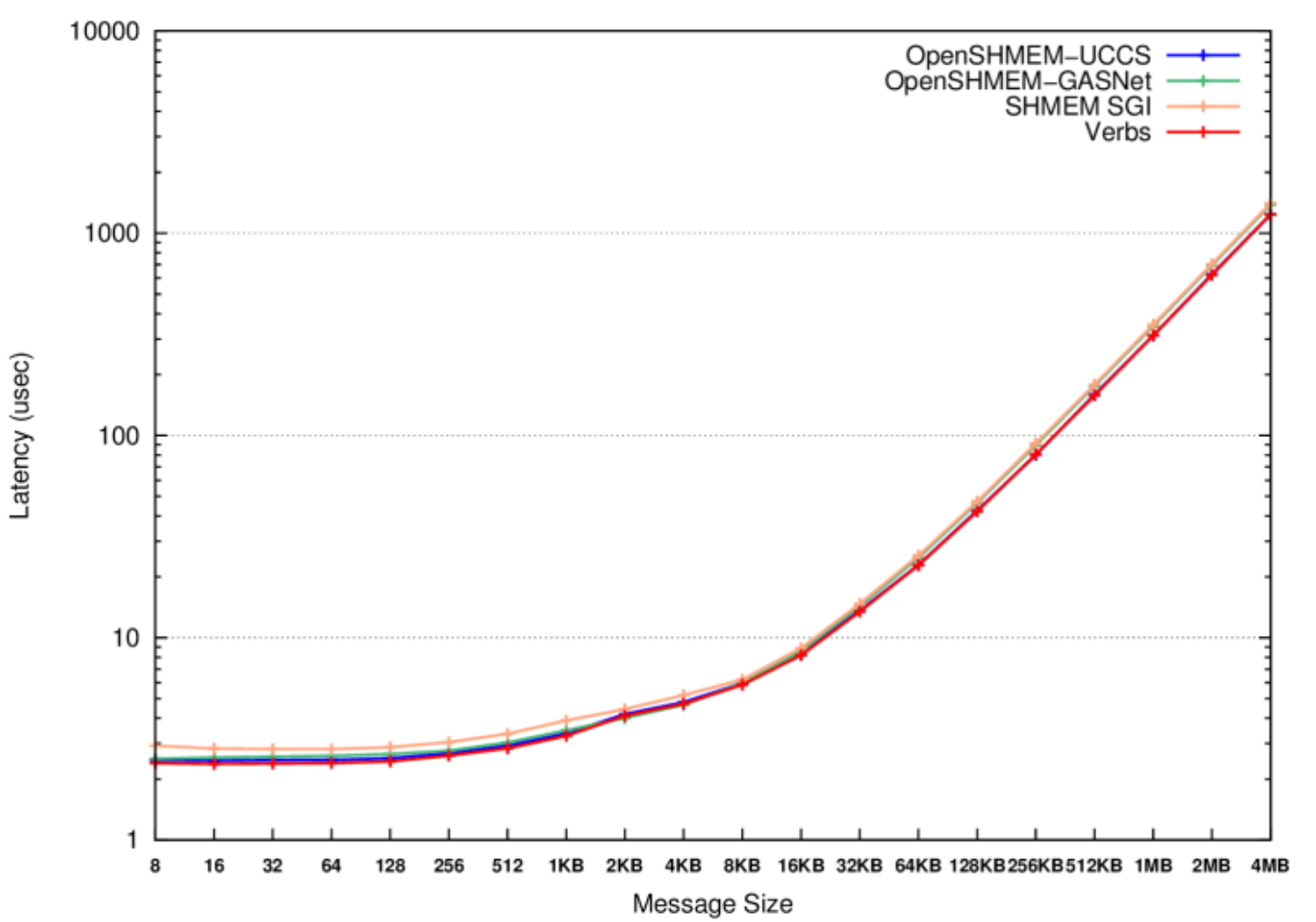
TESTING ENVIRONMENT

- **Results obtained from "Designing a High Performance OpenSHMEM Implementation Using Universal Common Communication Substrate as a Communication Middleware", First OpenSHMEM Workshop**

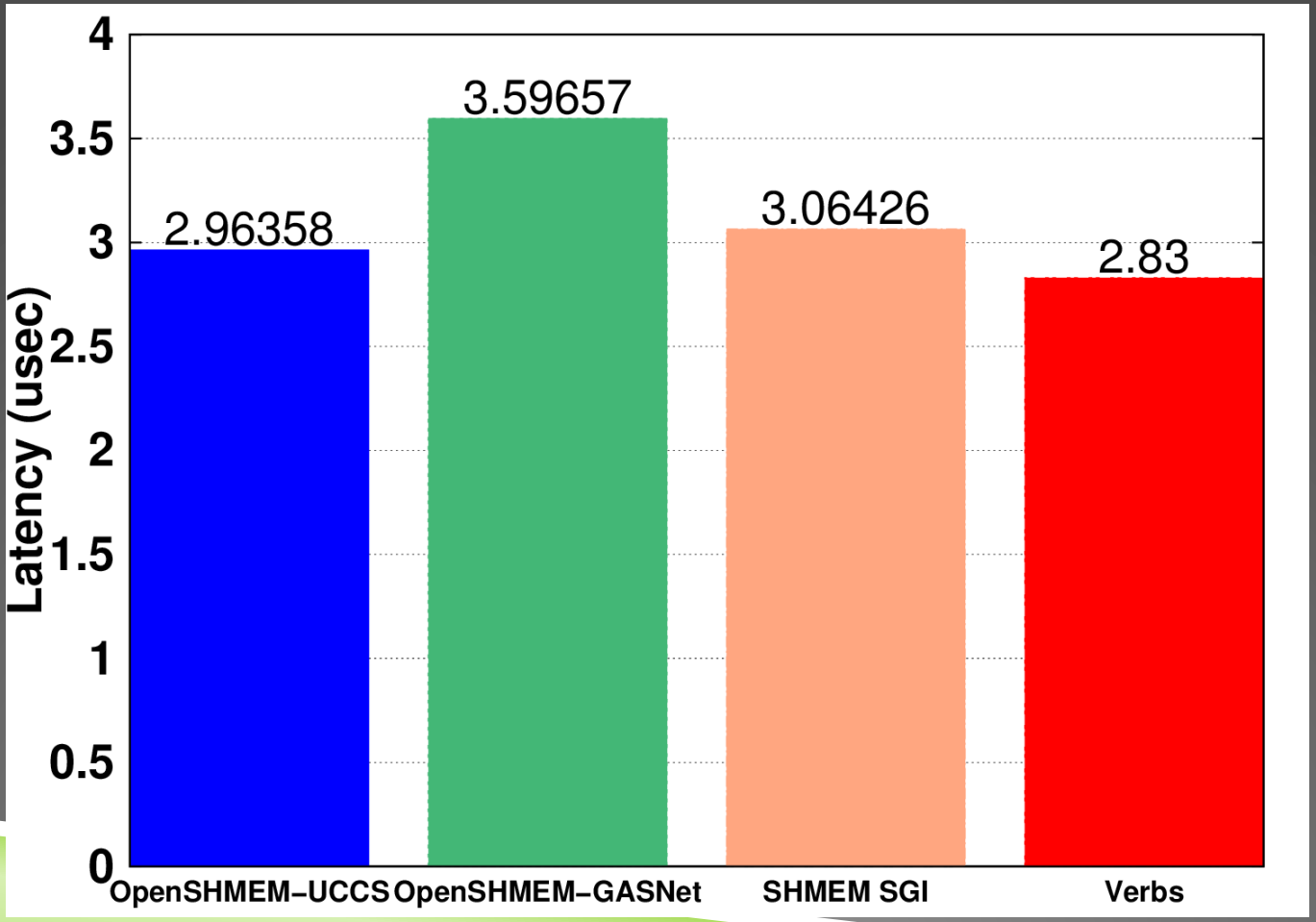
PUT LATENCY


OpenSHMEM–UCCS
OpenSHMEM–GASNet
SHMEM SGI
Verbs
Latency (usec)
10000
1000
100
10
1
8
16
32
64
128
256
512
1KB
2KB
4KB
8KB
16KB
32KB
64KB
128KB
256KB
512KB
1MB
2MB
4MB
Message Size

GET LATENCY


OpenSHMEM–UCCS
OpenSHMEM–GASNet
SHMEM SGI
Verbs
Latency (usec)
10000
1000
100
10
1
8
16
32
64
128
256
512
1KB
2KB
4KB
8KB
16KB
32KB
64KB
128KB
256KB
512KB
1MB
2MB
4MB
Message Size

LONG LONG FETCH-AND-ADD LATENCY



DEVELOPMENT STATUS

- Supported interconnects: IB, uGNI/Cray
- http://uccs.github.io/uccs/
- mailto:uccs-info@ornl.gov

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.