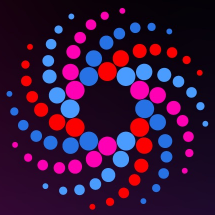


SC22
Dallas, TX | hpc accelerates.

Design and Performance Evaluation of UCX for Tofu-D Interconnect with OpenSHMEM-UCX on Fugaku

Yutaka Watanabe[1)2)], Mitsuhisa Sato [2)], Miwako Tsuji [2)], Hitoshi Murai [2)], Taisuke Boku [3)2)]

1) RIKEN Center for Computational Science

2) Graduate school of Science and Technology, University of Tsukuba

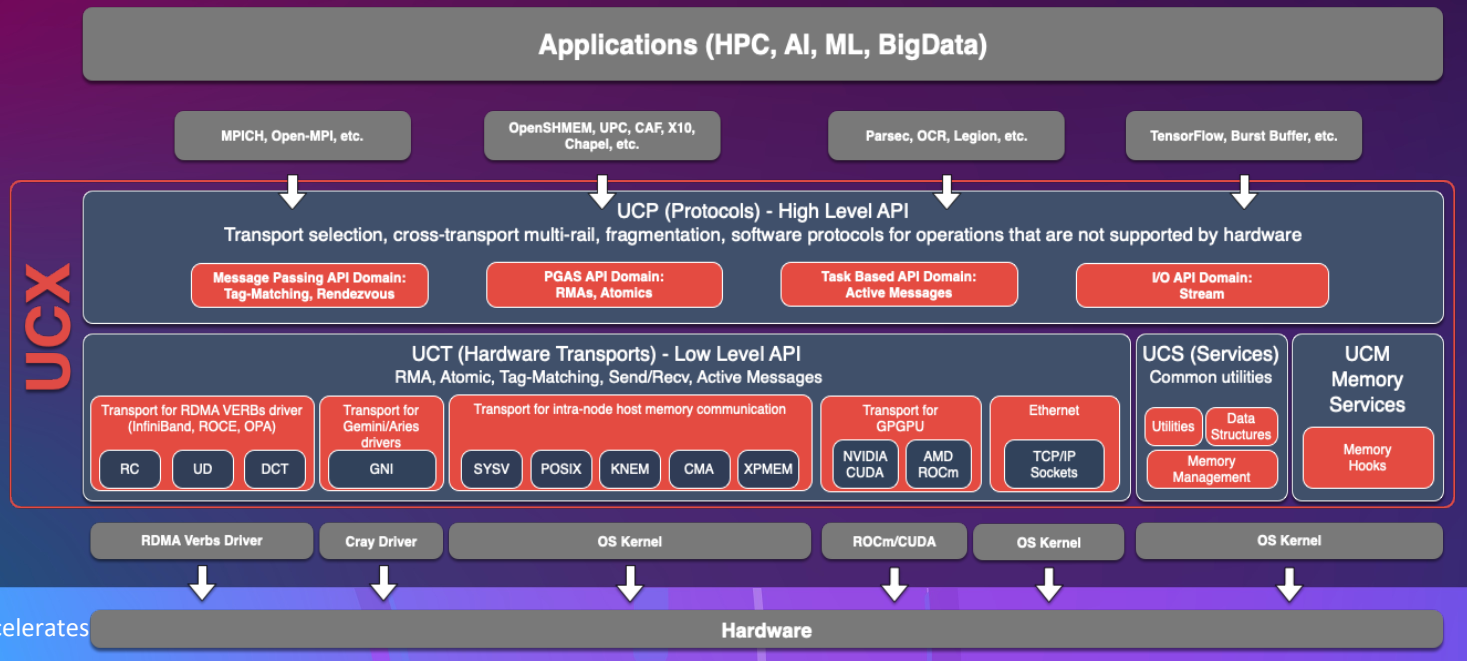
3) Center for Computational Sciences, University of Tsukuba

The 5th Annual Parallel Applications Workshop, Alternatives To MPI+X

Introduction and motivation

- The partitioned global address space (PGAS) model is recently getting attention as it is easy to describe remote data access
  - Mainly, one-sided communication is used in PGAS
  - One of the representative framework is OpenSHMEM
- UCX is a lower-level communication framework for modern, high-bandwidth, and low-latency networks
  - UCX is widely used in HPC; OpenMPI can use UCX as underlying network library, OpenSHMEM-UCX does so

UCX architecture[11, 12]



Introduction and motivation (Cont'd)

- UCX on Tofu-D for Fugaku supercomputer
  - Tofu-D is an interconnect network used in supercomputer Fugaku and FX1000 based systems
  - It enables lower, lightweight, and abstracted network library to users who want to use lower communication library than MPI but do not want to use library for Tofu-D (uTofu) directly. (e.g. for portability)
  - Users can take advantage Tofu-D's tightly coupled network and application-oriented topology with UCX

- RDMA based communication layers such as UCX are expected to perform better than MPI
  - We are interested in communication layer for multithreaded execution in manycore processors.
  - We found MPI has a performance problem with multithread communication, especially with many core processors.
  - We have a plan to implement a distributed task-based programming framework with abstracted network interface for portability

Tofu Interconnect D

- Tofu Interconnect D is a six dimensional mesh/torus network which is used on Fugaku and Fujitsu FX1000 based systems
  - The chip integrated to A64FX has 6 Tofu Network Interfaces which is an RDMA engine with 6.8GB/s capacity, 40.8GB/s aggregated bandwidth
  - Scalable Direct Network, designed for a large-scale MPP (over 150k nodes like Fugaku)

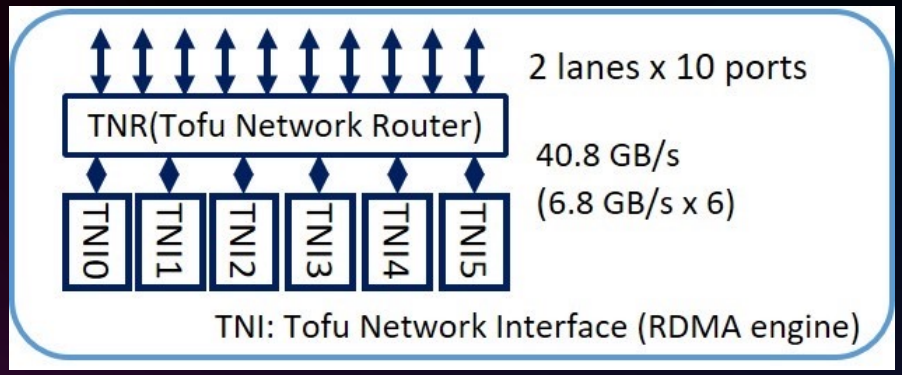
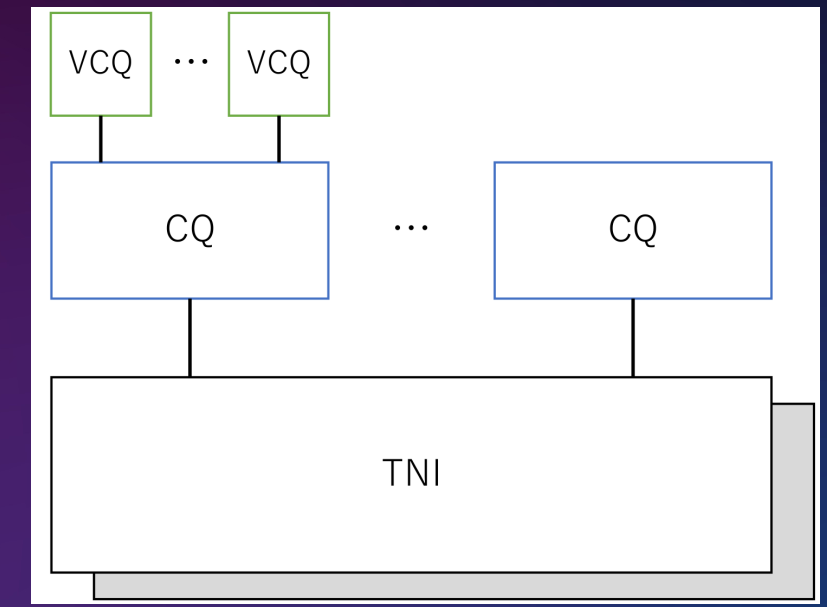


Diagram of Tofu-D[14]

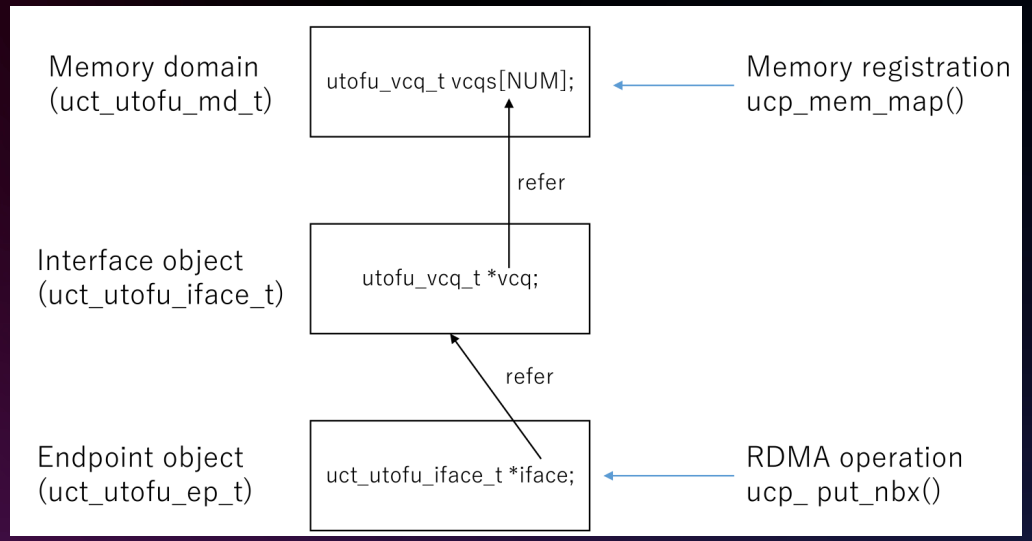
- The queue structure of Tofu is like right-below figure
  - hardware control queues (CQ) are on each TNI
  - Virtualized Control Queues (VCQ) are on each CQ



- With uTofu library (to use Tofu interconnect D from userland), those TNIs need to be explicitly managed by user
  - User creates VCQ on specific TNI
  - Memory registrations or RDMA operations are done thorough VCQ

Design and implementation: Mapping uTofu to UCT layer

- We have mapped uTofu library to UCT like above figure
  - VCQ is created on memory domain as all memory registration is done through memory domain
  - A interface object has a reference to VCQs on memory domain
  - A end point object has a reference to interface as all RDMA operation is done through endpoint object



- uTofu needs special address “steering address” for RDMA operation, not virtual address
  - Mapping remote’s steering address as rkey
    => exchange rkey with remote means
    exchange remote buffer address
  - register and de-register for local buffer address to get src steering address

```
ucs_status_t ucp_put_nbi(
    ucp_ep_h ep,
    const void *buffer,
    size_t length,
    uint64_t remote_addr,
    ucp_rkey_h rkey
)
```

```
int utofu_put(
    utofu_vcq_hdl_t vcq_hdl,
    utofu_vcq_id_t rmt_vcq_id,
    utofu_stadd_t lcl_stadd,
    utofu_stadd_t rmt_stadd,
    size_t length,
    uint64_t edata,
    unsigned long int flags,
    void *cbdata
)
```

Design and implementation: Scheduling Tofu Network Interfaces (TNIs)

- The scheduling of TNIs may be a design issue to improve performance
  - As Tofu Interconnect D has six TNIs and needs to be managed explicitly with uTofu library

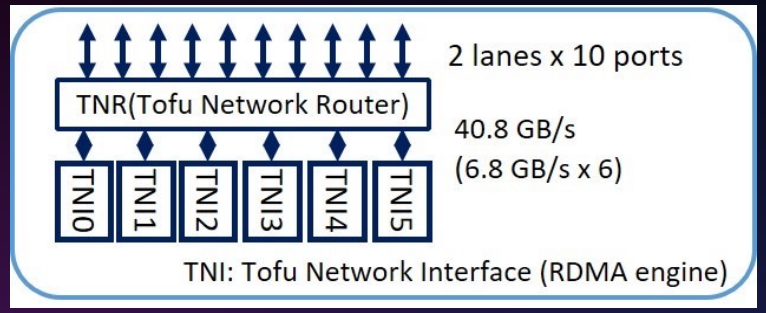


Diagram of Tofu-D[14]

- We have implemented two type methods for the use of TNIs
  - 1) TNI-RR: select specific TNI per remote node; the TNI will be selected by round-robin manner
  - 2) TNI-LU: using the least-used TNI for each communication

- TNI-RR will be better in terms of low overhead, but may result in load-imbalance between TNIs
- TNI-LU will be better in terms of load balancing, but requires extra overhead to find the least-used TNI

Mixed use of UCX and MPI

- OpenSHMEM-UCX relies on scholl library for collective operations
  - scholl implements collective operations with UCX
- uTofu has hardware-assisted operations like barriers and reduction, but not be able to utilize from UCX
  - This could affect performance on Fugaku
- The solution is to make use of MPI with UCX on OpenSHMEM-UCX, as MPI and uTofu can be mixed in a single program
  - Currently, global barrier of OpenSHMEM-UCX is modified to use MPI_Barrier for Fugaku

Performance evaluation

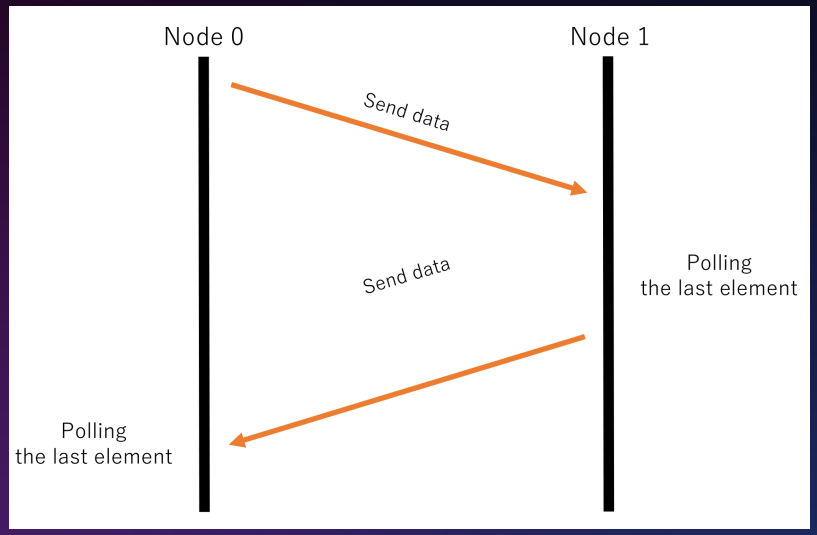
- We conducted performance evaluation on supercomputer Fugaku

- We have evaluated with three kind of benchmnarks
  - Simple pingpong benchmark with Fujitsu MPI, UCX, and uTofu
  - OpenSHMEM implentation of OSU Micro Benchmark with OpenSHMEM-UCX and OSHMPI
  - Oak Ridge OpenSHMEM Benchmarks with OpenSHMEM-UCX and OSHMPI

Fugaku specification and used softwares	
CPU	Fujitsu Arm A64FX Processor 48 compute cores + 2or4 assistant cores at 2.0GHz
Interconnect	Tofu Interconnect D
OS	Red Hat Enterprise Linux 8.5
Linux Kernel	4.18.0-348.20.1.el8 5.aarch6
MPI	Fujitsu Technical Computing Suite 4.8.0
Base UCX	Commit ID 5f357ae0215868a89bc09a05c5392c710eb6e7da
OSHMPI	Commit ID a51874c829e66d83486974b9f7ce076f3687934
Base OpenSHMEM-UCX	Commit ID a51874c829e66d83486974b9f7ce076f3687934

Pingpong benchmark on Fugaku: uTofu, UCX, and MPI RDMA

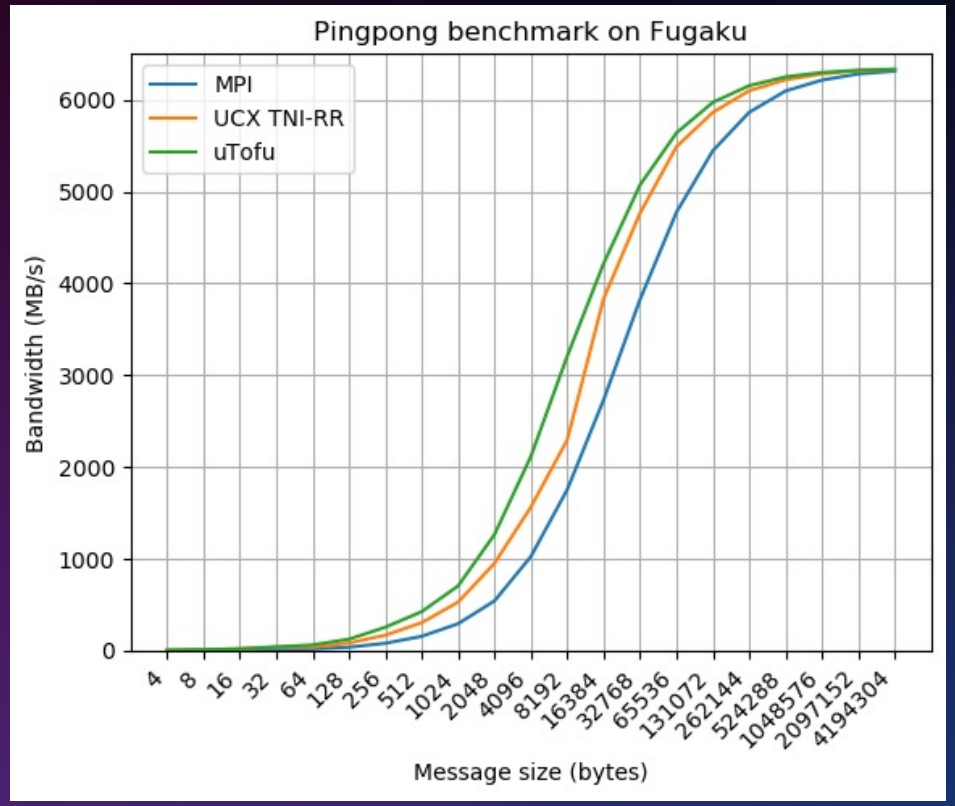
- We have implement a pingpong benchmark to evaluate and compare the RDMA write performance of Fujitsu MPI, UCX, and uTofu on Fugaku
  - For Fujitsu MPI, we used window lock based mechanism



Pingpong benchmark on Fugaku: uTofu, UCX, and MPI RDMA

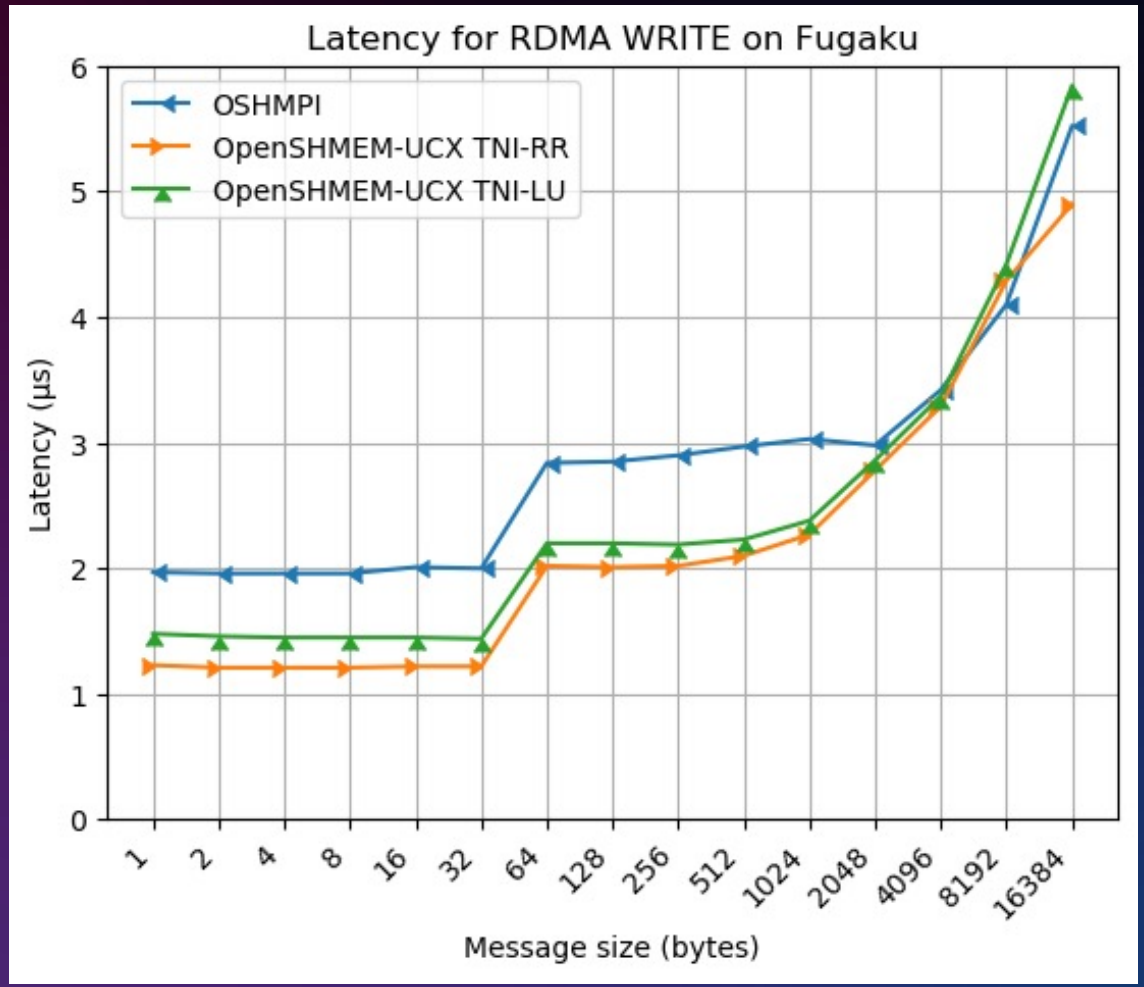
- uTofu shows the best performance, followed by UCX (TNI-RR) and MPI RDMA respectively.

- One of the reason for the difference between uTofu and UCX is caused by memory registration cost in uTofu layer on UCX
  - Source buffer is registered and de-registered every time (except up to 32byte message)
  - Improving memory management is not done yet



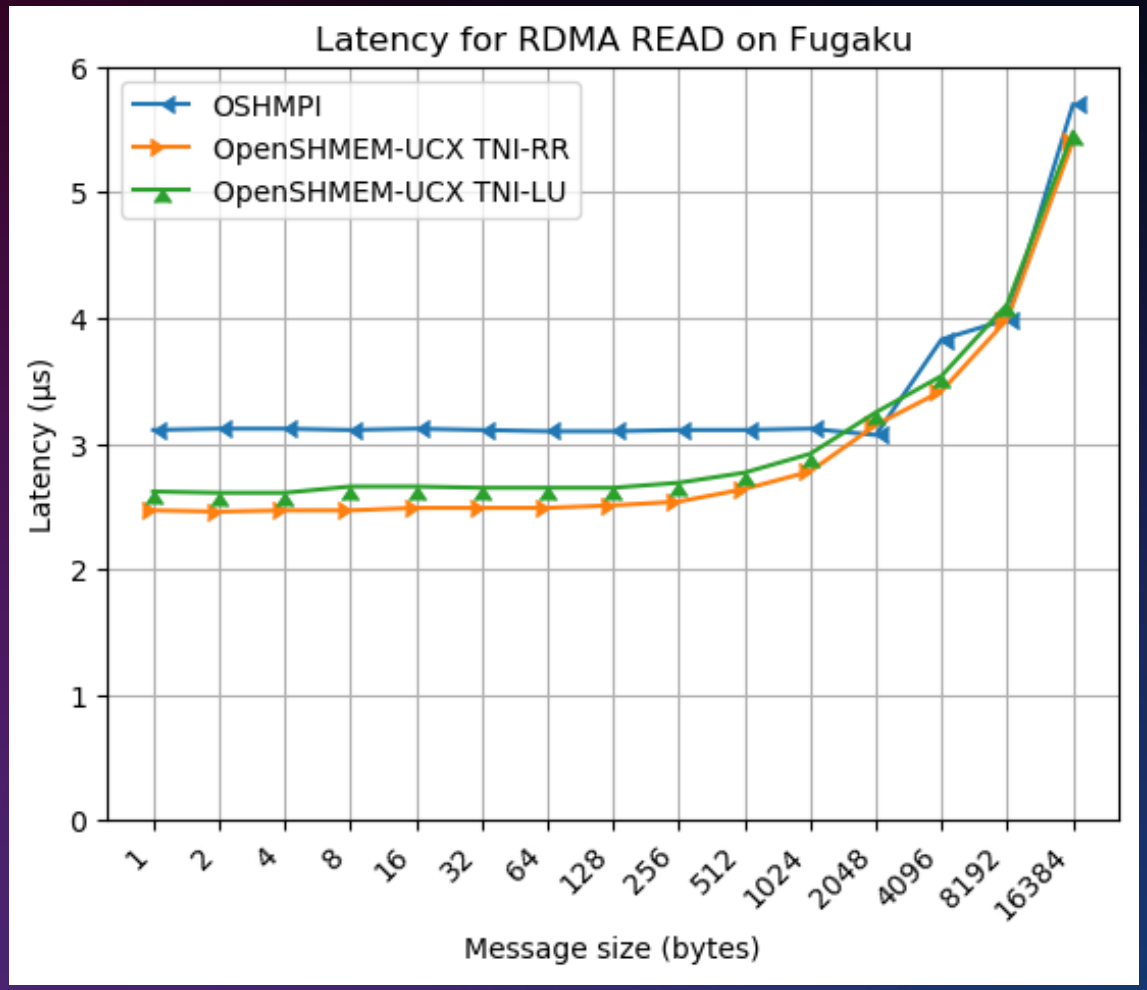
OSU Micro Benchmarks on Fugaku: OSHMPI and OpenSHMEM-UCX with RDMA WRITE

- Both OpenSHMEM-UCX shows lower latency up to 16384 than OSHMPI except with 8192 byte message
  - TNI-RR is about 1.5x faster than OSHMPI with up to 32 byte message
- TNI-RR shows lower latency than TNI-LU
  - The additional cost of calculating which TNI is the most least is affecting the performance
- The latency increase after 32 byte is because of different utofu
  - Up to 32 byte message uses more efficient utofu_put_piggyback function, which reduces the cost of local memory access



OSU Micro Benchmarks on Fugaku: OSHMPI and OpenSHMEM-UCX with RDMA READ

- Both OpenSHMEM-UCX shows lower latency up to 16384 byte message than OSHMPI except 2048 byte message
  - TNI-RR mode is about 1.28x faster than OSHMPI with 1 byte message
- TNI-RR shows lower latency than TNI-LU same as WRITE
  - The additional cost of calculating which TNI is the most least is affecting the performance



OSU Micro Benchmarks on Fugaku: OSHMPI and OpenSHMEM-U

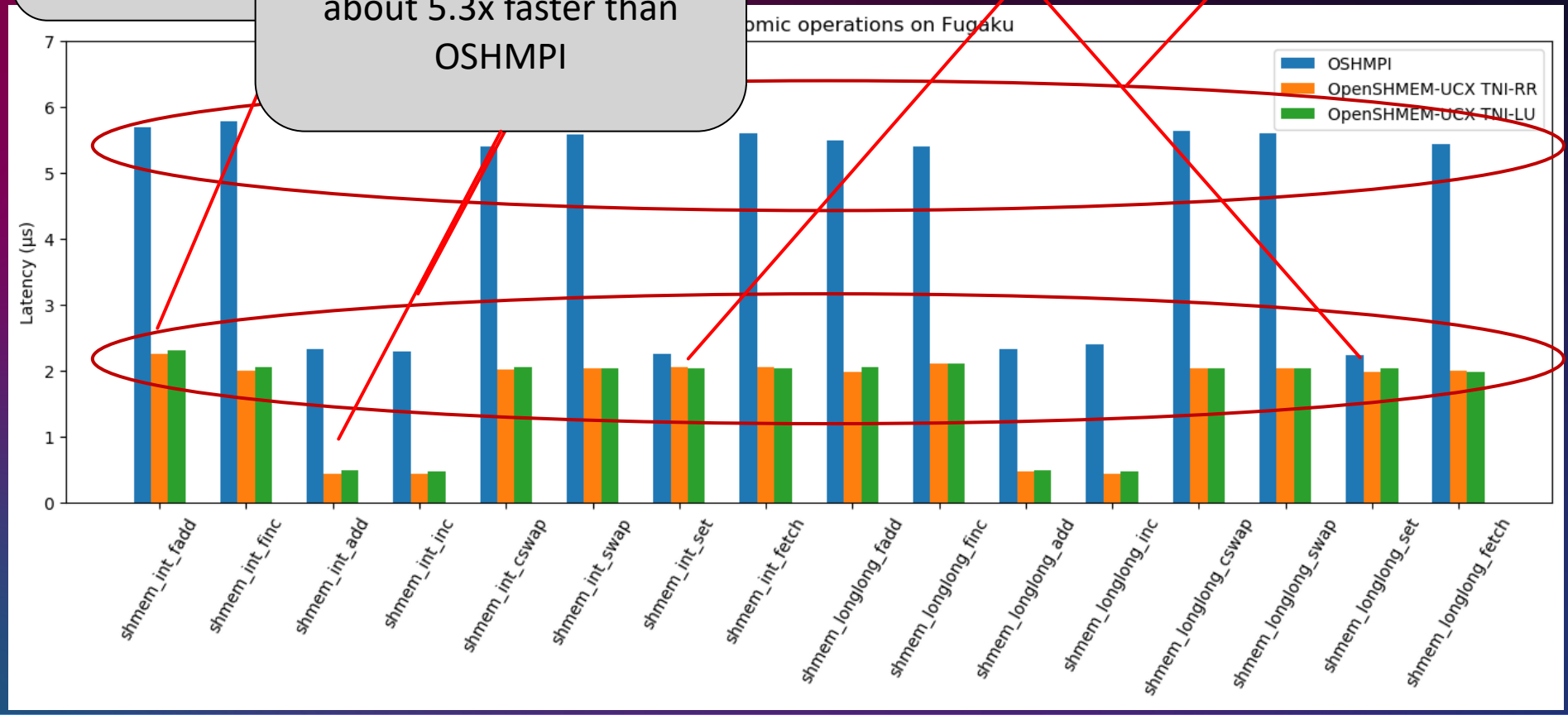
For 4byt
OpenSHMEI
about 2.5x fas

For 4byte add, OpenSHMEM-UCX is about 5.3x faster than OSHMPI

For 4 or 8 byte set, OSHMPI is slightly slower than both OpenSHMEM-UCX

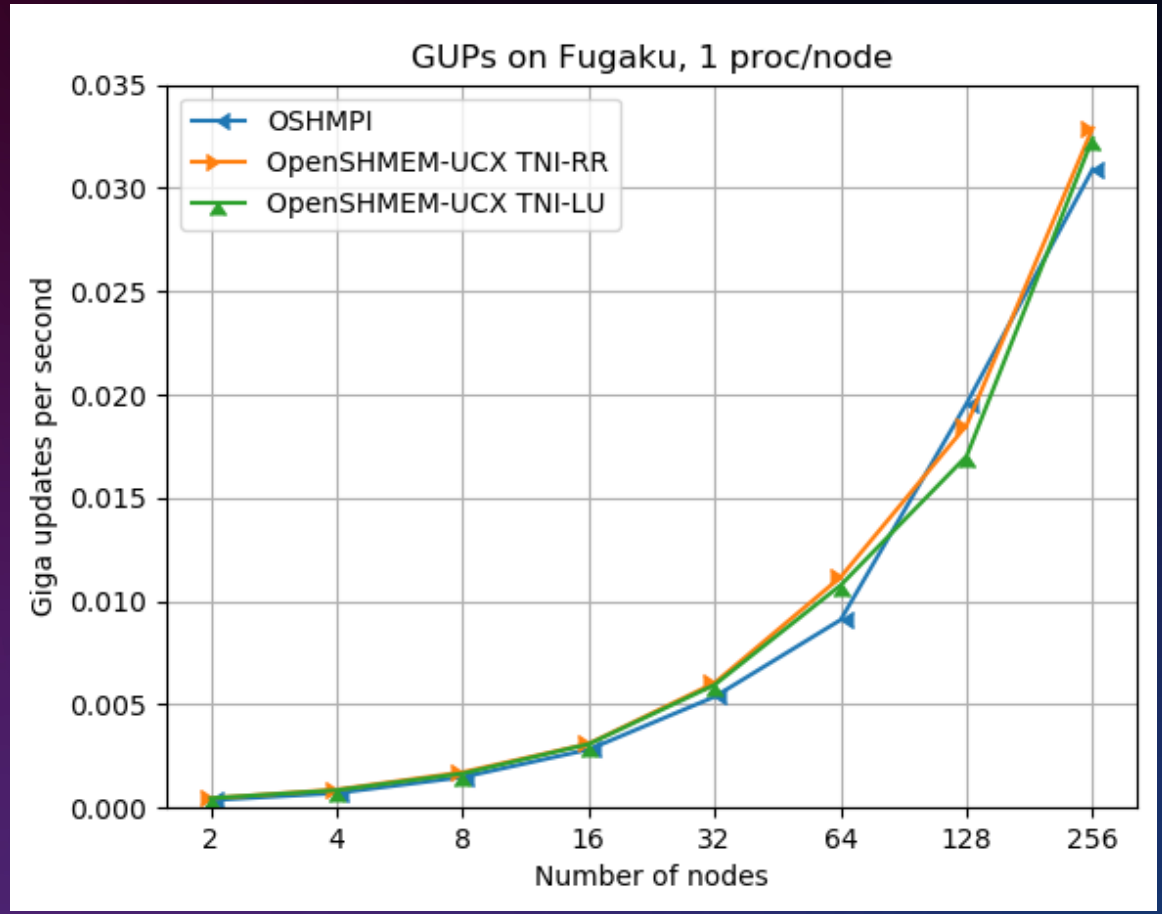
Some atomic operation fo OpenSHMEM-UCX shows
alr nction are used
fa add{4,8}
dd{4, 8}
wap{4, 8}

s shows
y than
M-UCX



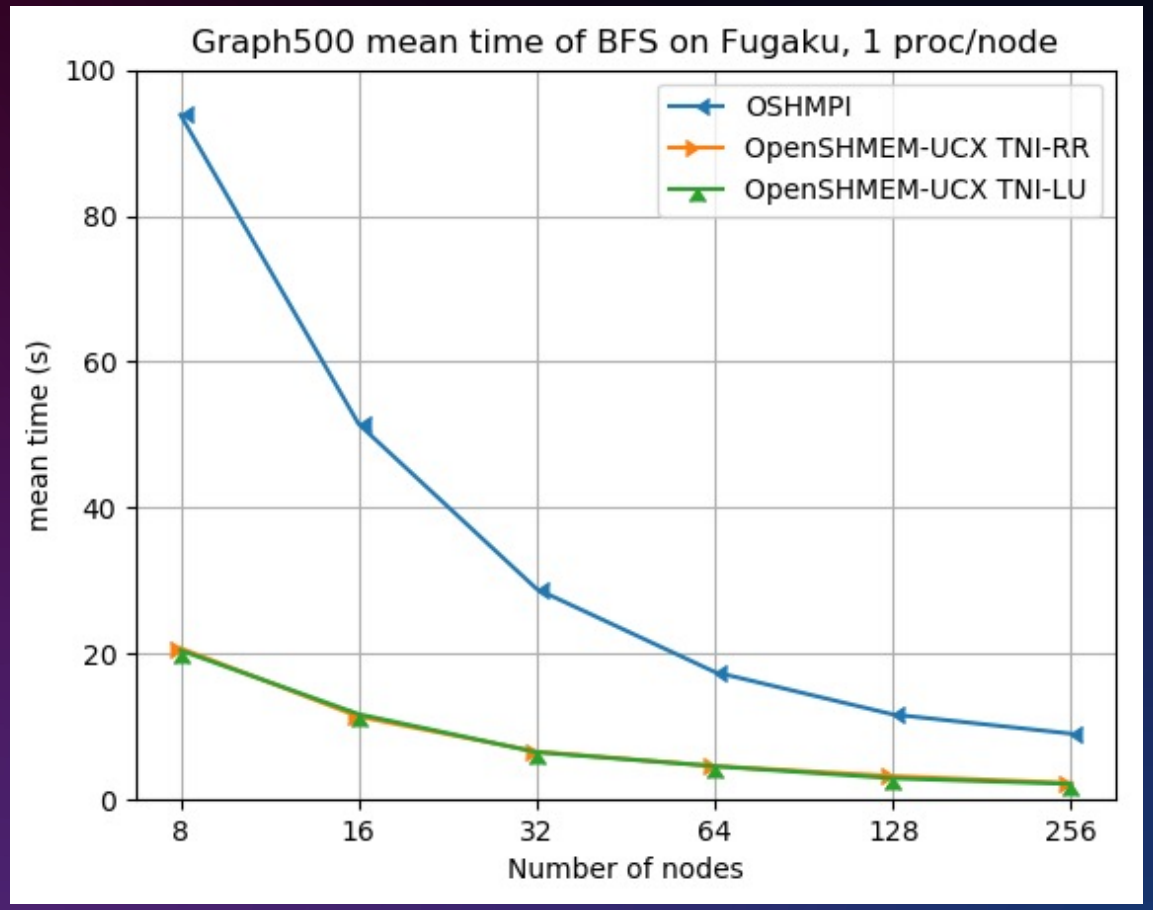
Oak Ridge OpenSHMEM benchmarks on Fugaku: OSHMPI and OpenSHMEM-UCX with GUPs

- Measuring GUPS (Giga Updates Per Second) following the RandomAccess test from the HPC Challenge
  - Evaluated with 2 to 256 nodes
- Both OpenSHMEM-UCX shows better performance than OSHMPI except with 128 nodes
  - TNI-RR is about 1.09x better than OSHMPI with 256 nodes
- TNI-RR is better than TNI-LU, similar to previous evaluation



Oak Ridge OpenSHMEM benchmarks on Fugaku: OSHMPI and OpenSHMEM-UCX with Graph500 mean time

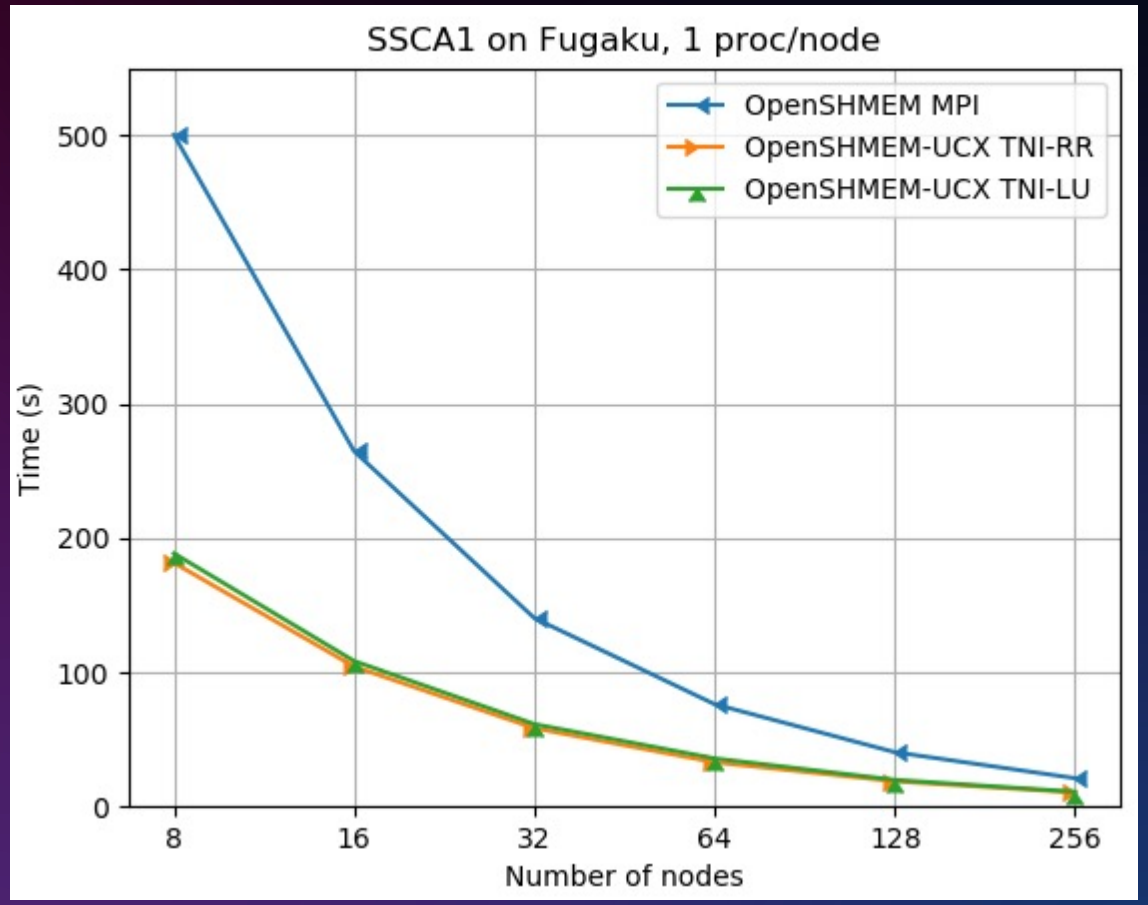
- Problem setup
  - SCALE: 20, Edge factor:16
  - Evaluated with 8 to 256 nodes
- Both OpenSHMEM-UCX method are faster than OSHMPI
  - OpenSHMEM-UCX TNI-RR is about 3.9x faster than OSHMPI with 256nodes, more difference than observed with OMB
- Almost same performance between TNI-RR mode and TNI-LU mode with OpenSHMEM-UCX



Oak Ridge OpenSHMEM benchmarks on Fugaku: OSHMPI and OpenSHMEM-UCX with SSCA1

- SSCA1 is a sequence alignment algorithms
- Problem setup
  - SCALE: 27
  - Evaluated with 8 to 256 nodes

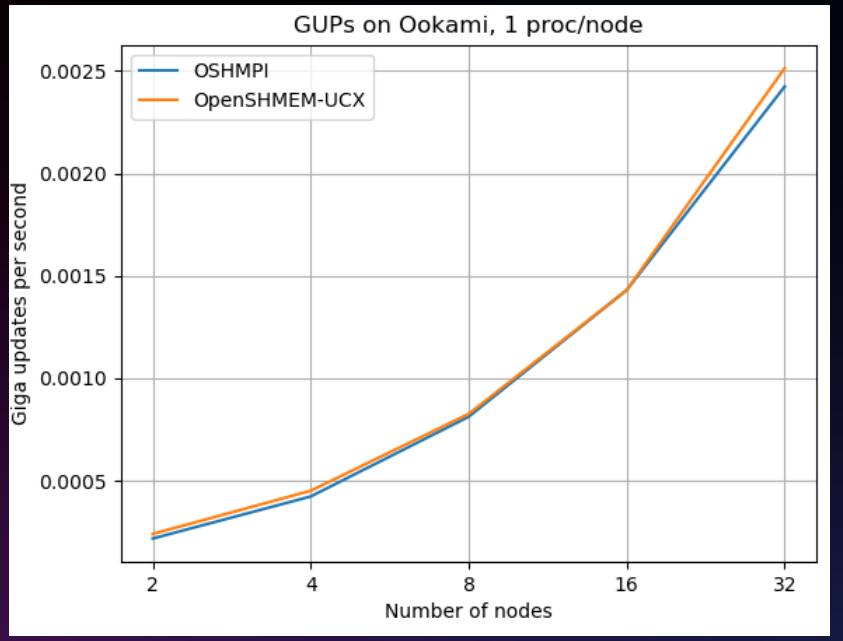
- Both OpenSHMEM-UCX method are faster than OSHMPI
  - OpenSHMEM-UCX TNI-RR is about 2x faster than OSHMPI with 256nodes, more difference than observed with OMB but similar to Graph500

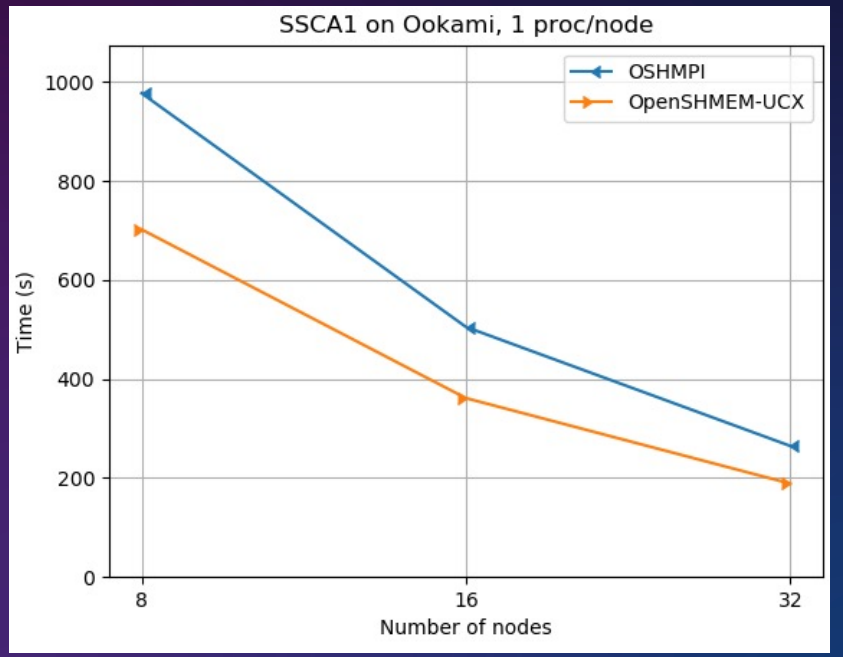


Discussion

- The large difference observed in Graph500 and SSCA1 application
  - We have evaluated on Ookami cluster
  - Unfortunately, Graph500 didn't run correctly

Ookami specification and used softwares	
CPU	Fujitsu Arm A64FX Processor 48 compute cores at 1.8GHz
Interconnect	**InfiniBand HDR100 PCIe Gen3 x16**
OS	Rocky Linux 8.4
Linux Kernel	4.18.0-305.25.1.el8 4.aarch64
MPI	OpenMPI 4.1.4
UCX	1.11.2
OSHMPI	Commit ID 62e1ec0267cd3be05e764de2ee54dbfe40311132
OpenSHMEM-UCX	Commit ID a51874c829e66d83486974b9f7ce076f3687934





- With GUPs benchmarks, as similar to Fugaku, OpenSHMEM-UCX shows better performance than OSHMPI
- With SSCA1 benchmarks and from 8 to 32 nodes, as similar to Fugaku, larger difference is observed
  - This would mean the reason of this difference is not simply due to between UCX and MPI, but also between OpenSHMEM-UCX and OSHMPI
  - While the root cause of this is not identified yet, users can benefit more OpenSHMEM-UCX on than OSHMPI on Fugaku

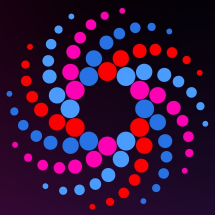
Discussion:
UCX and Tofu Interconnect D

- For collective operations with multiple nodes
  - Tofu-D has hardware assisted operations like barrier and reduction
  - UCX would not be able to utilize them because of the architecture

- One of the solution for this is to mix MPI and UCX as Fujitsu MPI and uTofu can be used in a single program
  - This requires additional cost to users, but users would take advantages of UCX's lightweight RDMA operations
  - UCC project might be the alternative solution?

Conclusion and future work

- We have designed and implemented uTofu layer for UCX
  - UCX on Fugaku shows better performance than MPI with RDMA based benchmarks, and is expected as more lightweight and portable network library.
  - Scheduling TNIs for Tofu-D improves performance slightly.
  - UCX can be used as a efficient RDMA lower communication layer for Fugaku supercomputer than MPI in portable way.
  - Our design of uTofu layer for UCX would be helpful for users who try to port UCX for any kind of network interface
- For future works
  - Improving memory management like fewer registration/de-registration cost which could reduce the latency and close to uTofu raw performance
  - Evaluation and Optimization of our UCX on Tofu-D for thread-to-thread communication in distributed task-based programming framework for parallel manycore processors system such as Fugaku



SC22
Dallas, TX | hpc accelerates.

Design and Performance Evaluation of UCX for Tofu-D Interconnect with OpenSHMEM-UCX on Fugaku

Yutaka Watanabe[1)2)], Mitsuhisa Sato [2)], Miwako Tsuji [2)], Hitoshi Murai [2)], Taisuke Boku [3)2)]

1) RIKEN Center for Computational Science

2) Graduate school of Science and Technology, University of Tsukuba

3) Center for Computational Sciences, University of Tsukuba

The 5th Annual Parallel Applications Workshop, Alternatives To MPI+X