Explicit RMA and Merged Requests

Swen Boehm
August, 2 2016

Outline

- Motivation
- Proposed API
- Use Cases
- Implementation
- Benchmark results
- Conclusion

Motivation

- Current non-blocking operations need to be finished using shmem_quiet, shmem_barrier or shmem_barrier_all
  - Will finish **all** outstanding operations
- Improve control over outstanding RMA operations by introducing explicit handles
  - Only finish RMA operations that are needed to continue computation
- Provide Interface to group related RMA operation

Proposed API – Explicit requests

- shmem_TYPE_put_nbe (TYPE *target, const TYPE *source, size t nelems, int pe, shmem request handle t **handle);
- shmem_putSIZE_nbe (TYPE *target, const TYPE *source, size t nelems, int pe, shmem request handle t **handle);
- shmem_TYPE_get_nbe (TYPE *target, const TYPE *source, size t nelems, int pe, shmem request handle t **handle);
- shmem_getSIZE_nbe (TYPE *target, const TYPE *source, size t nelems, int pe, shmem request handle t **handle);

Proposed API – Merged requests

- shmem_TYPE_put_nbe multiple(TYPE *target, const TYPE *source, size t nelems, int pe, shmem request handle t **handle);
- shmem_TYPE_get_nbe multiple(TYPE *target, const TYPE *source, size t nelems, int pe, shmem request handle t **handle);

Proposed API – Requests completion

- void shmem_test_req(shmem request handle t *handle);
  - Test if operation is complete
- void shmem_wait_req(shmem request handle t *handle);
  - Wait for operation to complete

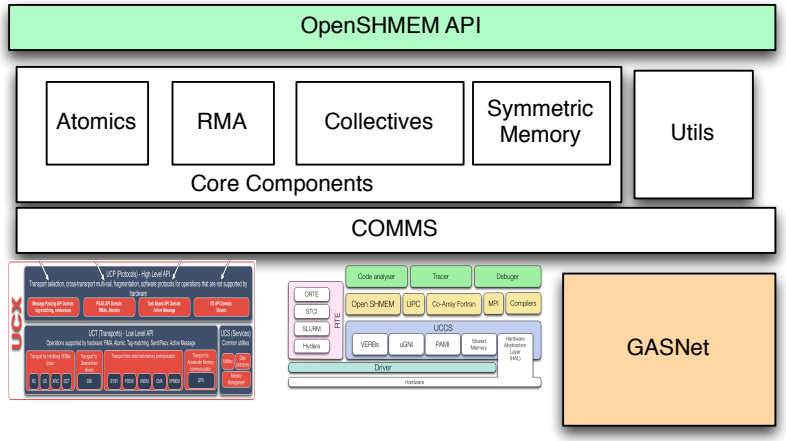
Use Cases

- Define Patterns
  - Merge related operations and provide overlap with computation
  - combine communication phase in stencil operation
- merged requests can provide the means for customized asynchronous collectives
  - i.e. custom broadcast from any PE
  - Remove requirement for active-set
  - Provide overlap for collectives not updating the same symmetric object

Use Cases cont.

- Combine RMA operations of a thread into merged request
  - allows concurrency between non-related RMA operations issued by the same or different thread

Explicit RMA Implementation using UCX as Communicaton Layer



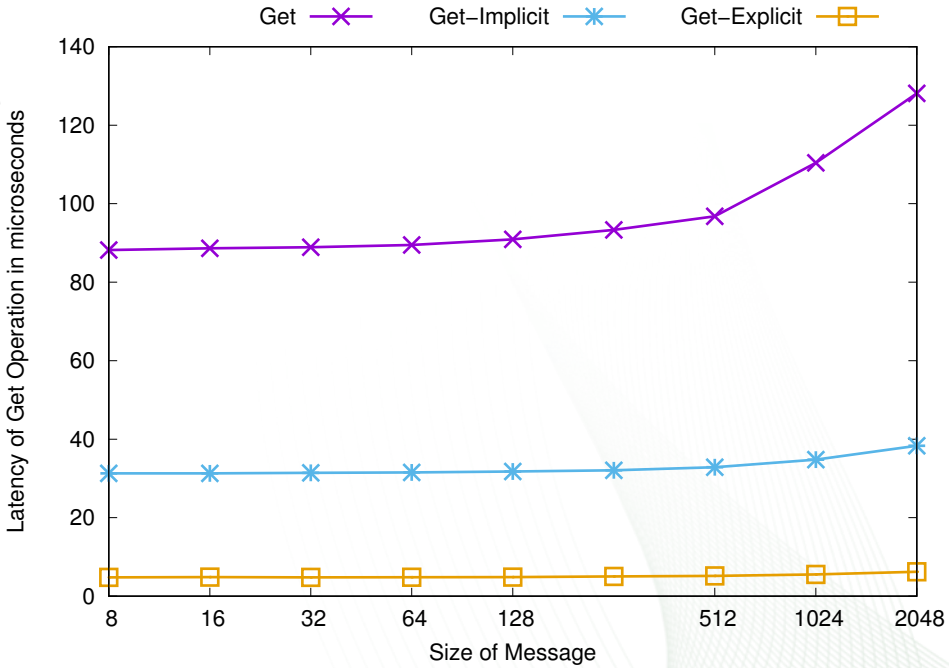
- Implemented in the OpenSHMEM reference Implementation
  - Reference implementation defines the new interface as SHMEM extension
  - Implementation in UCX networking layer

Benchmarks

- Ported OSU benchmarks to support implicit & explicit RMA operations
  - Micro benchmarks used to show that explicit RMA operations do not decrease performance
- SSCA 1 benchmark ported to explicit RMA operations
  - Synthetic Application Benchmark
  - Performance improvements of 49-72%

Benchmarks - get-many latency

- Implemented get_many (based on OSU get)
  - Benchmark uses get operations get data from multiple nodes
  - Non-blocking operations outperform blocking get
  - Explicit non-blocking operation has advantage over implicit operation

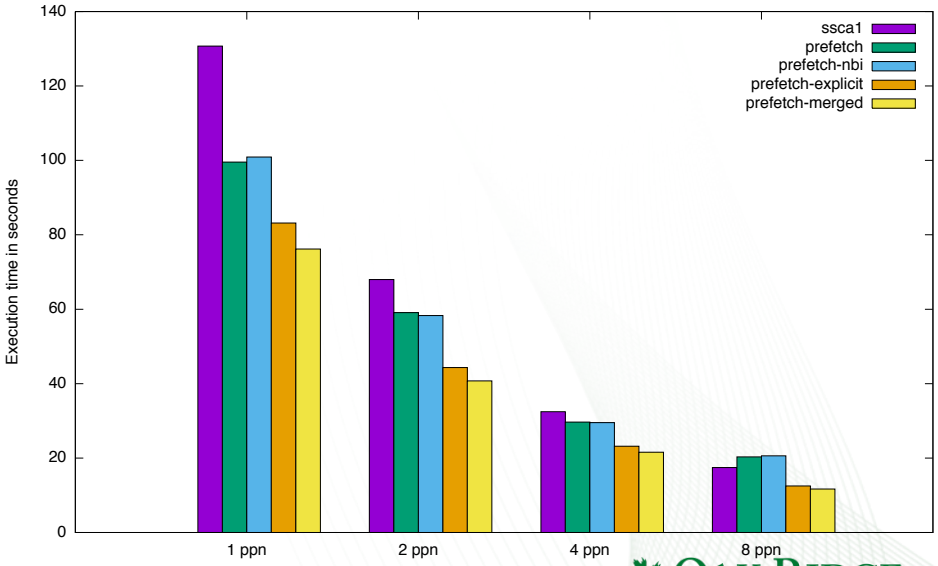


Benchmarks - SSCA 1

- Bioinformatics benchmark from DARPA High Productivity Computing Systems program
- Smith- Waterman local sequence alignment algorithm
- Improvements focus on Kernel 1

Benchmarks - SSCA 1

- SSCA #1
  - ssca1 and prefetch are unmodified
- Modified Benchmark in multiple steps
  - prefetch-nbi
    - Add put_nbi add the end of the inner loop
  - prefetch-explicit
    - Replace implicit operations with explicit operations
  - prefetch merged
    - Use merged requests



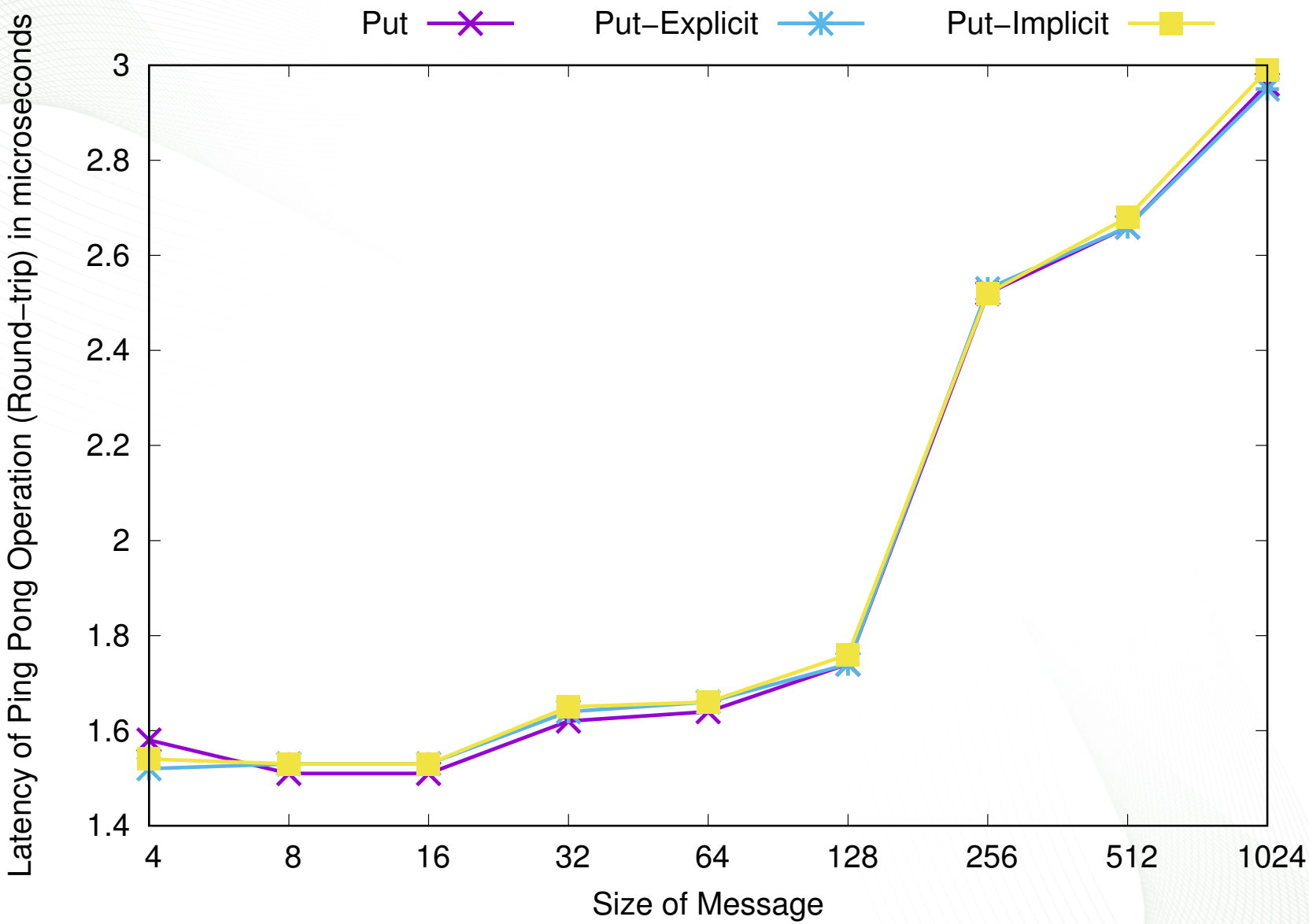
Conclusion

- Familiar interface
- Better control over outstanding RMA operations
- Increased performance for some communication patterns

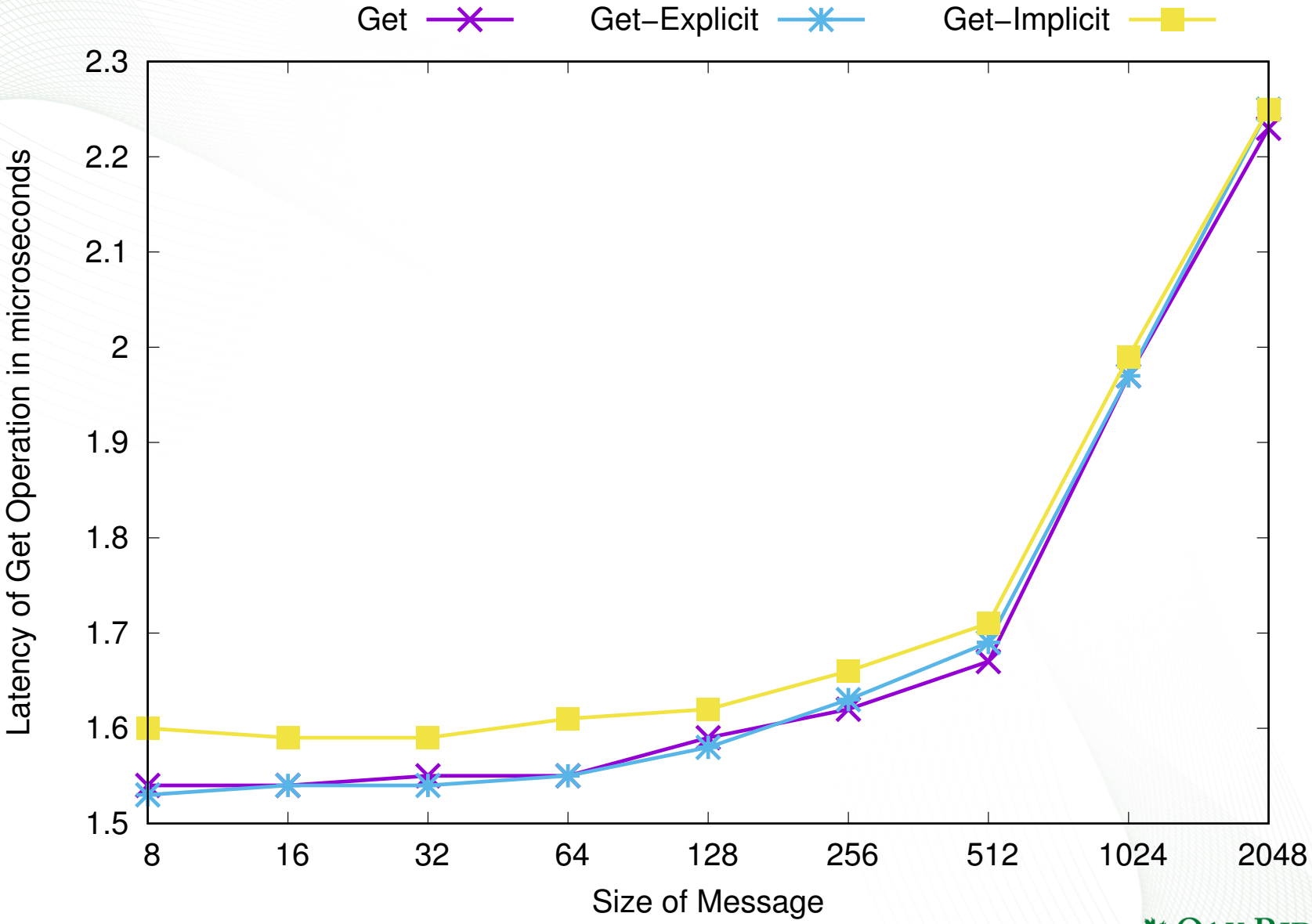
Questions?

Backup – Benchmarks

OSU put

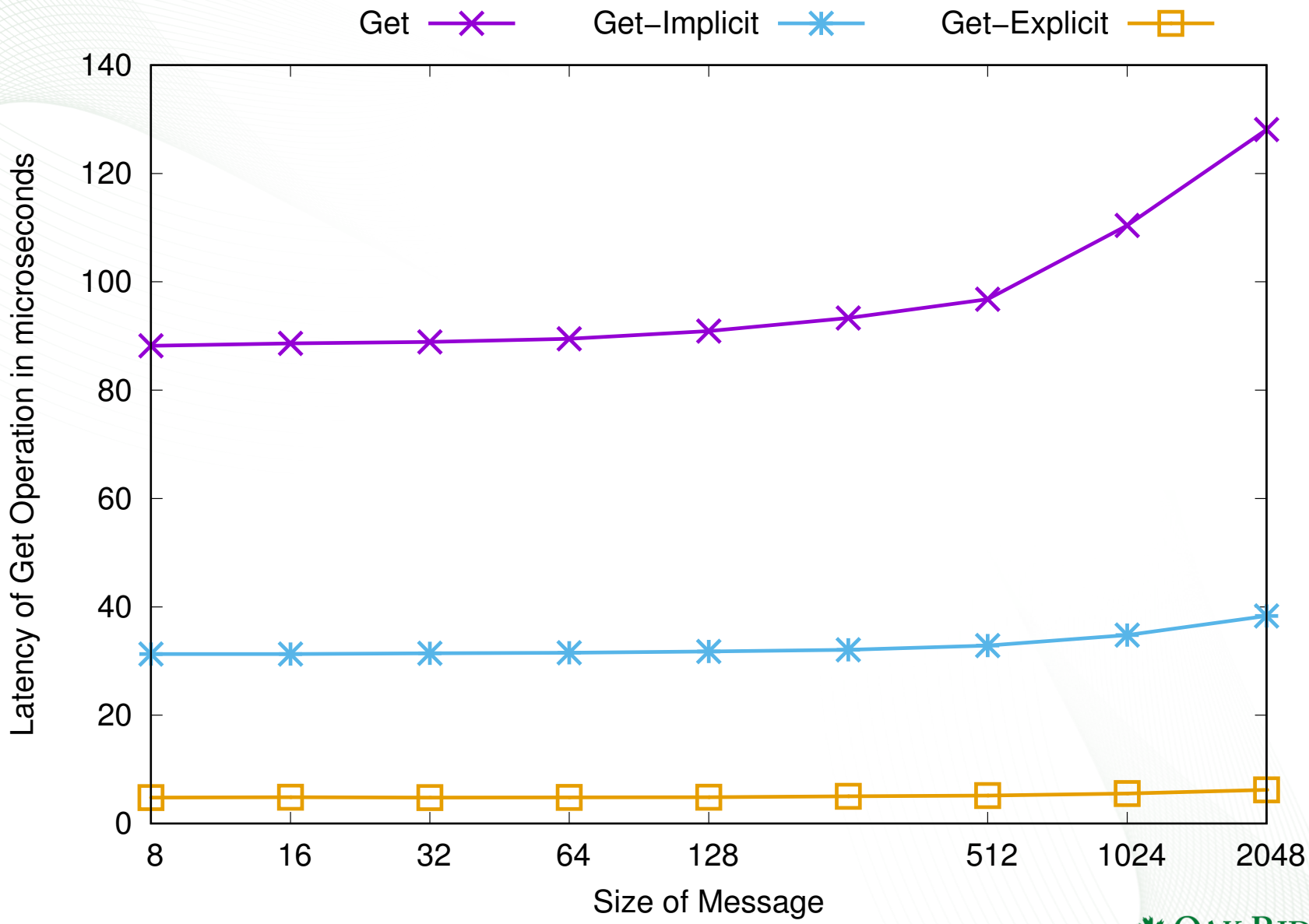


OSU get latency

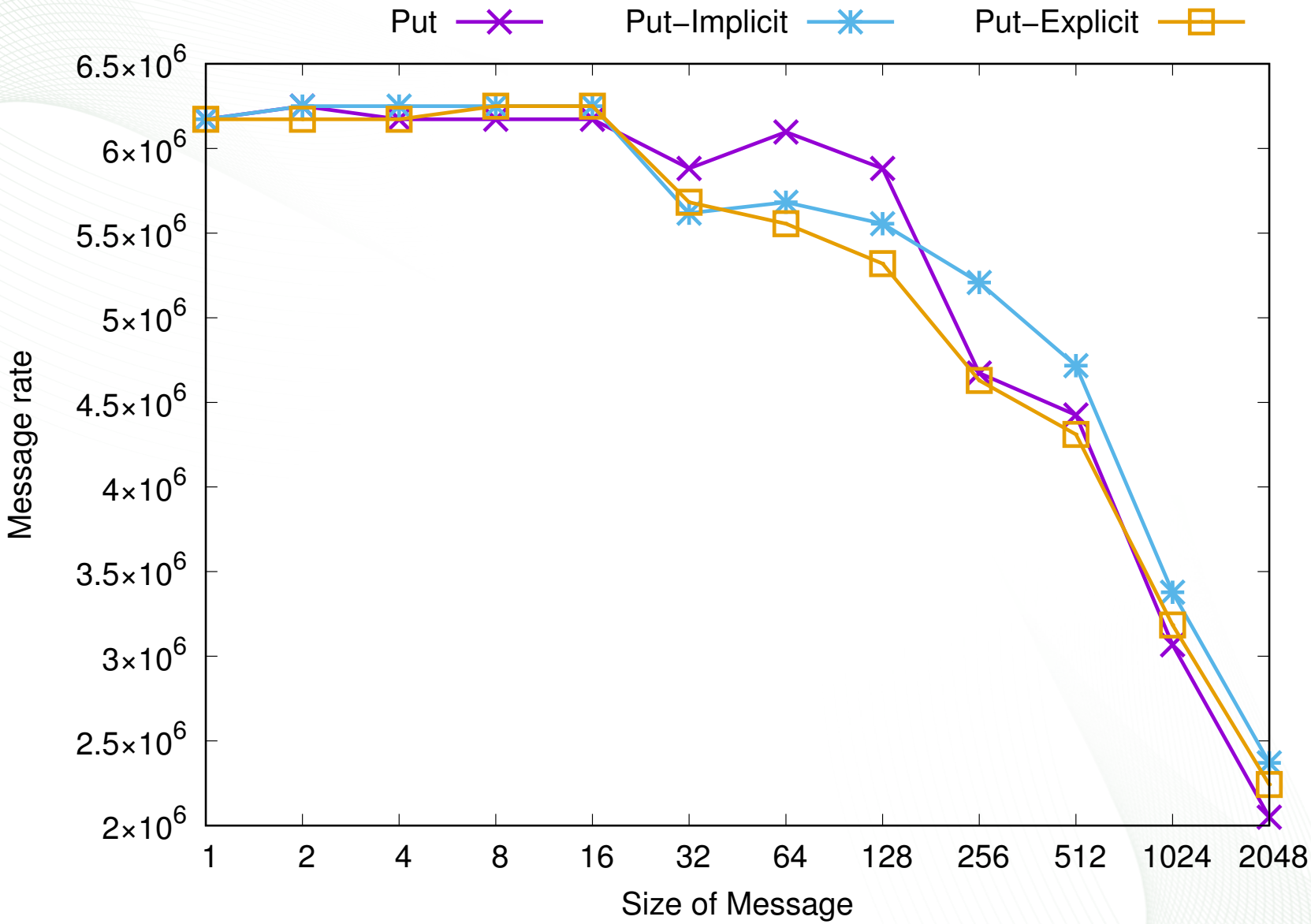


OAK RIDGE National Laboratory

OSU get-many latency



OSU put message rate



OAK RIDGE National Laboratory

SSCA 1