Designing, Implementing, and Evaluating the Upcoming OpenSHMEM Teams API

David Ozog, MD. Wasi-ur- Rahman, Gerard Taylor, James Dinan

Parallel Applications Workshop, Alternatives to MPI+X (PAW-ATM)

November 17, 2019

Legal Disclaimers

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps. Intel provides these materials as-is, with no express or implied warranties.

This document contains information on products in the design phase of development which Intel may change at any time without notice. Do not finalize a design with this information. Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Intel® , Intel logo and Xeon® are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Other names and brands may be claimed as the property of others.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at [intel.com].

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

WARNING - This document contains technical data whose export is restricted by the Arms Export Control Act (Title 22, U.S.C., Sec 2751, etq.) or the Export Administration Act of 1979, as amended, Title 50, U.S.C., App. 2401 et seq. Violations of these export laws are subject to severe criminal penalties. Disseminate in accordance with provisions of DoD Directive 5230.25.

This document contains information on products in the design phase of development which Intel may change at any time without notice. Do not finalize a design with this information. Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Optimization Notice

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel.

Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors.

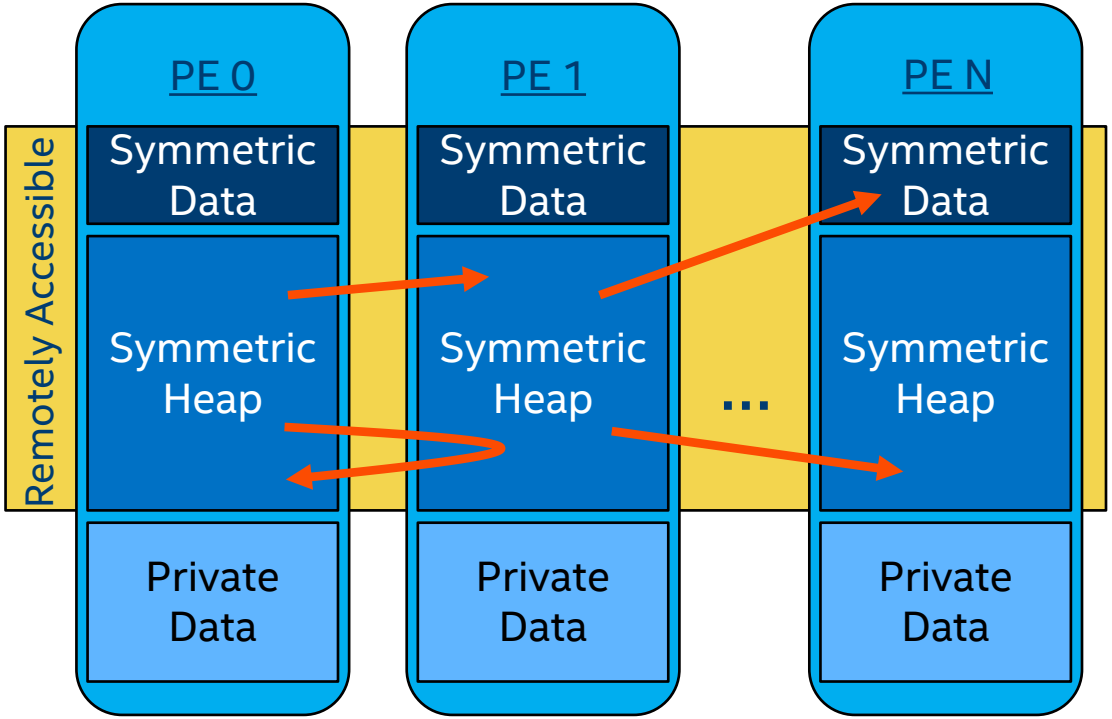
Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.

Test and System Configurations: Sandia OpenSHMEM running on Diamond cluster (Intel® Omni-Path 100 Series, Intel® Xeon Platinum 8170 (Skylake), and Cori, Intel®Xeon E5-2698 v3 (Haswell)

Performance results are based on testing as of Sept 19, 2019 and may not reflect all publicly available security updates. See configuration disclosure for details. No component or product can be absolutely secure.

Introduction to OpenSHMEM

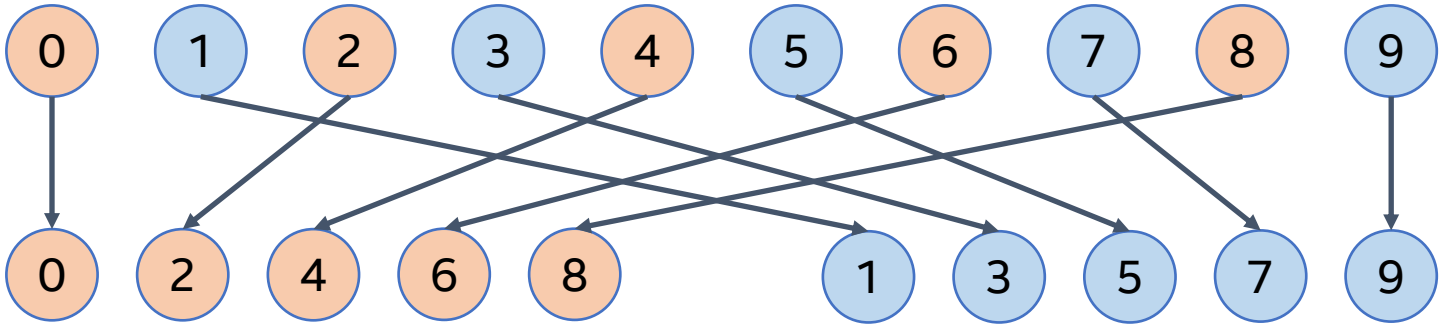
- Open standard PGAS model
- Emphasizes one-sided operations (put/get, atomics)
- Symmetric memory exposed for remote access
- Collective Operations:
  - Barrier, broadcast, collect, reductions, all-to-all, etc.



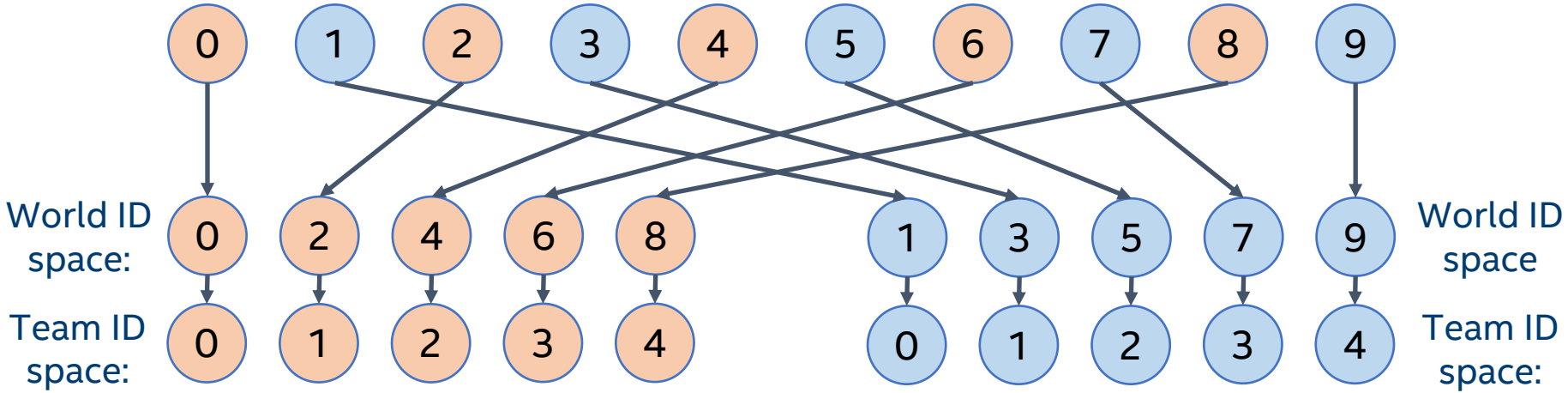
Teams - Motivation

Teams - Motivation

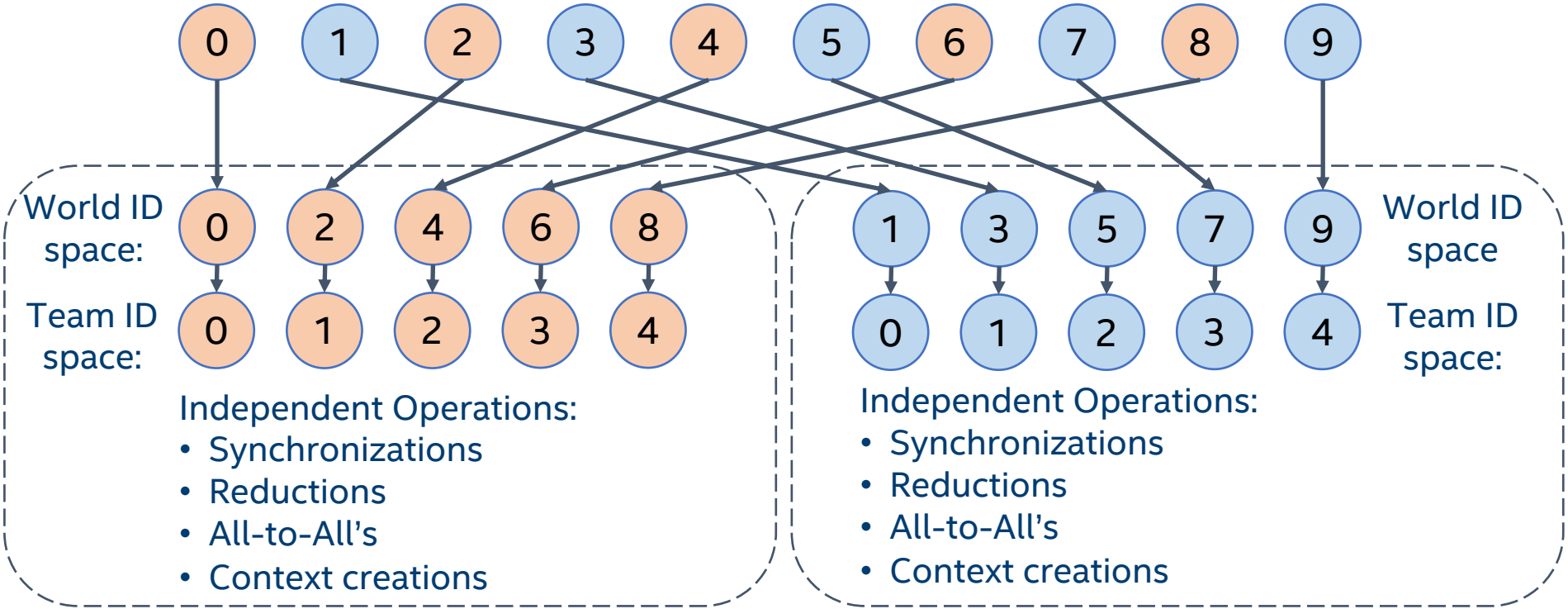
Teams - Motivation



Teams - Motivation

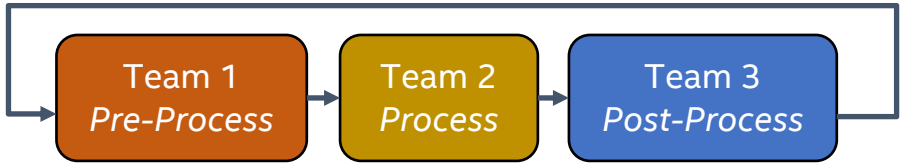


Teams - Motivation

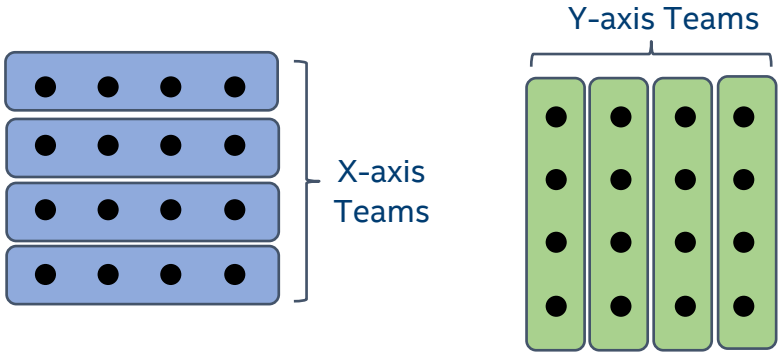


Teams – Some Motivating Examples

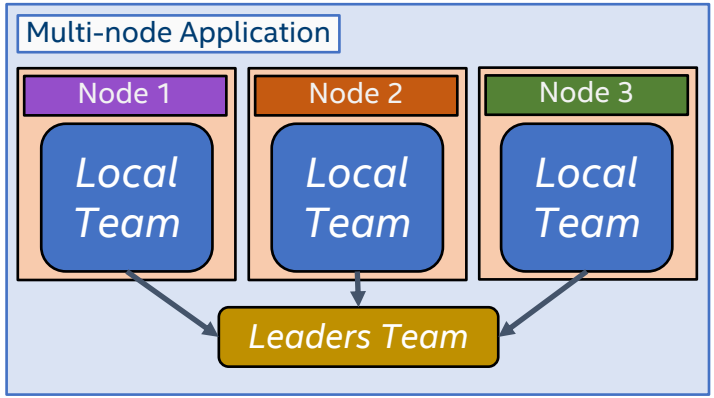
Pipelining / Rings



Useful Topologies (like Cartesian)



Shared Memory Teams



3rd Party Library management

- Parallel HDF5, ADIOS, PETSc, etc.

OpenSHMEM Collectives APIs

Current Version 1.4:

- Active set based: if (*start*, *log_stride*, *size*) == (3, 1, 4), then the active PEs are {3, 5, 7, 9}.
- User supplies a “pSync” array: used for explicit synchronization control.

```
void shmem_broadcast32(void *dest, const void *source, size_t nelems,
int PE_root, int PE_start, int logPE_stride, int PE_size, long *pSync);
```

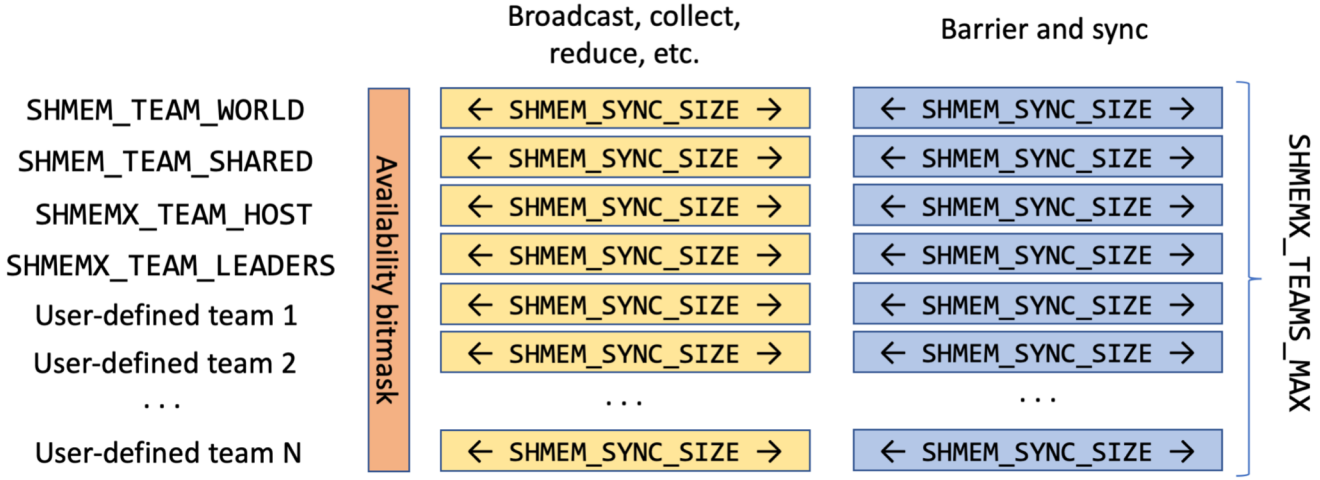
OpenSHMEM Teams proposal (version 1.5):

- The “active set” is specific to a team – defined during team creation.
- No more pSync exposure – managed by the implementation.

```
int shmem_broadcast(shmem_team_t team, TYPE *dest, const TYPE *source,
size_t nelems, int PE_root);
```

pSync Allocation and Management

- pSync/pWrk allocation done at initialization.
- A bitmask dictates which teams utilizes a particular pSync/pWrk region.
- Team creation bottleneck becomes an AND reduction across the bitmask.
- Can have any number of psyncs for back-to-back collectives)



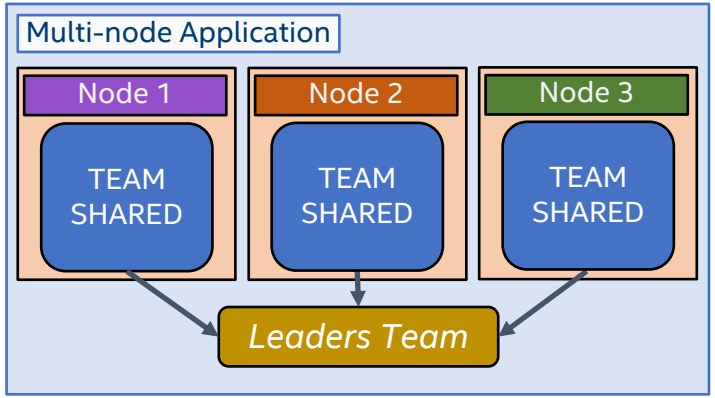
Team creation via `team_split_strided`

```
int shmem_team_split_strided(shmem_team_t parent_team,
                             int start, int stride, int size,
                             shmem_team_config_t *config, long config_mask,
                             shmem_team_t *new_team);
```

- Always collective on the parent team.
- If a PE is outside the (child) active set, call an internal barrier.
- If PE is inside the (child) active set, agree on a pSync via an AND reduction.
- Set the determined pSync reservation bit to 1, update team pool pointer.
- Finish parent barrier.

Teams Sharing Memory

- Iterate over all PE numbers and check `shmem_ptr()`
  - `shmem_ptr()` returns NULL if PE's address is inaccessible.
- For each PE, check that the stride stays *consistent*.
  - Certain teams not representable via a triplet, like {0, 1, 2, 4}.
  - If inconsistent, we give up and set `SHMEM_TEAM_SHARED` to the self PE.
- Various alternative teams possible:
  - PEs sharing a hostname
  - PEs sharing a NUMA domain
  - PEs within a topological fabric group

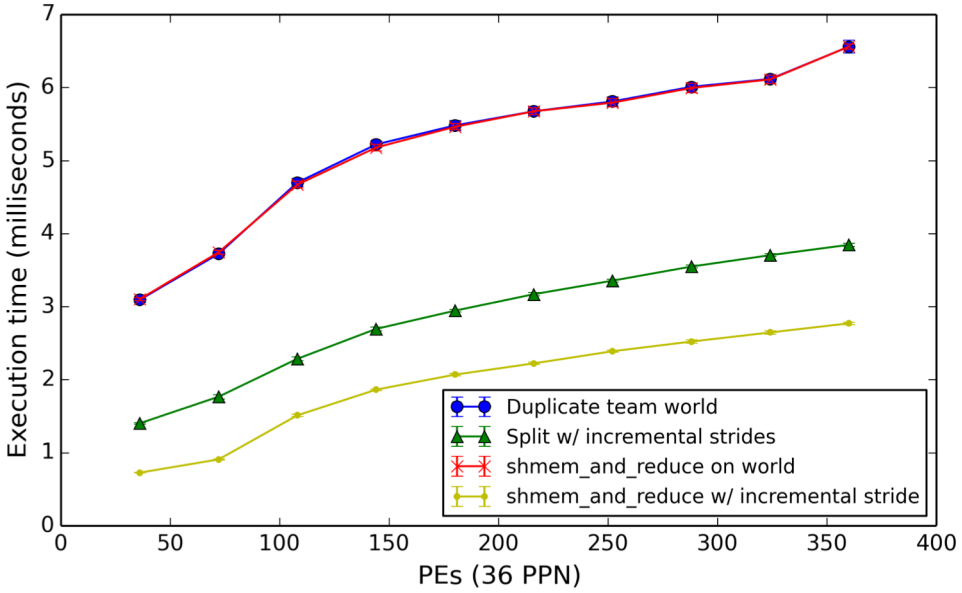


Performance Experiments

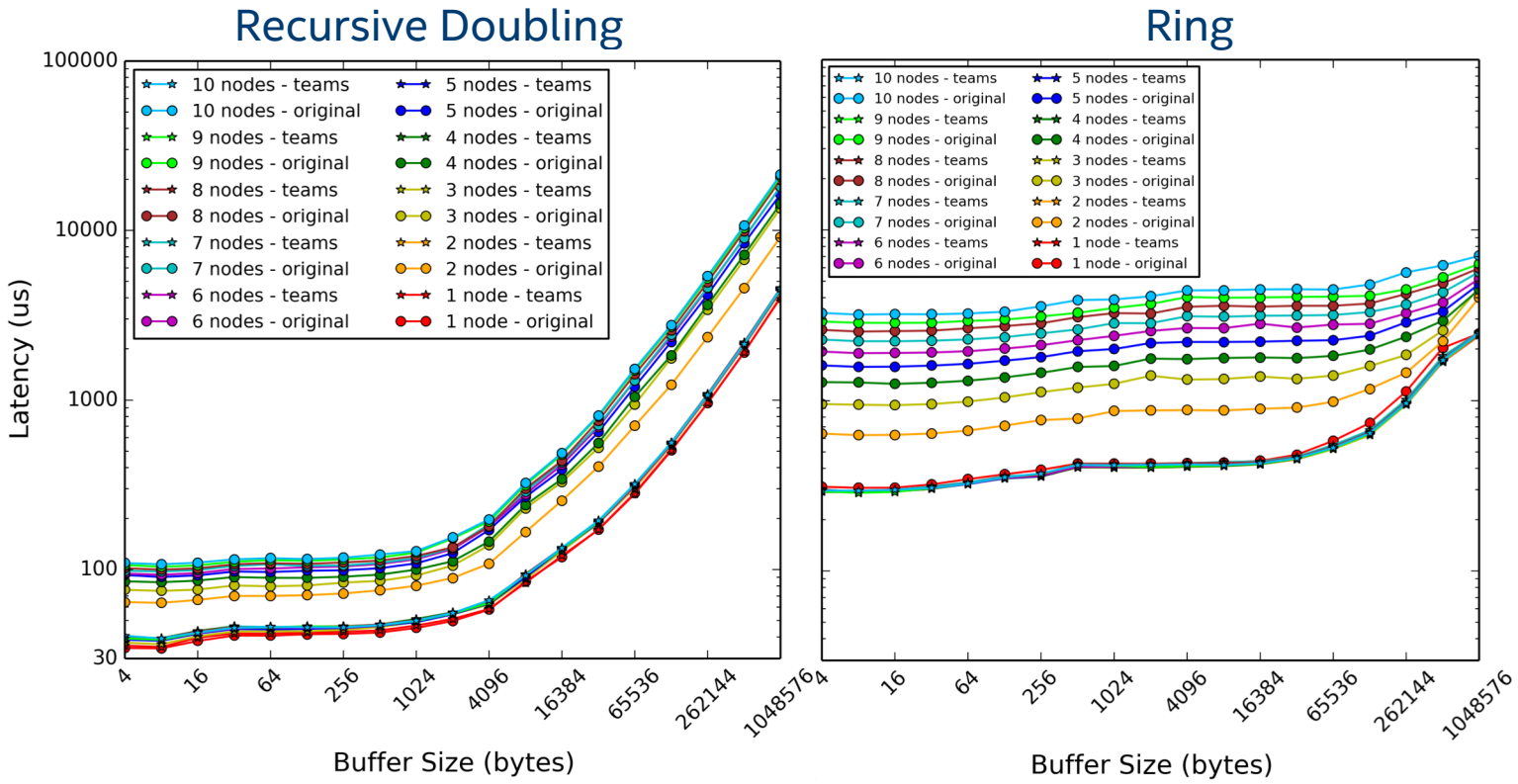
- Cori at NERSC:
  - Cray XC40, Intel®Xeon E5-2698 v3 (Haswell), 64 Hyper-Threaded cores per node, Aries fabric with dragonfly, GNU GCC version 8.2.0, libfabric version 1.8.x with the GNI provider.
- Diamond at Intel:
  - Intel® Xeon® Platinum 8170 (Skylake), 104 Hyper-Threaded cores per node, Intel® Omni-Path 100 series fabric with fat-tree, GNU GCC 4.8.5, libfabric version 1.7.0 with the PSM2 provider.
- OSU sum reduction microbenchmark
  - Reduce across SHMEM_TEAM_SHARED, then across SHMEM_TEAM_LEADERS.
- Ring
  - Reduce-Scatter followed by all-gather
  - Bandwidth optimized
- Recursive Doubling
  - Series of pairwise transfers between PE's with powers of 2
  - Latency optimized

Team Creation Latency Experiment

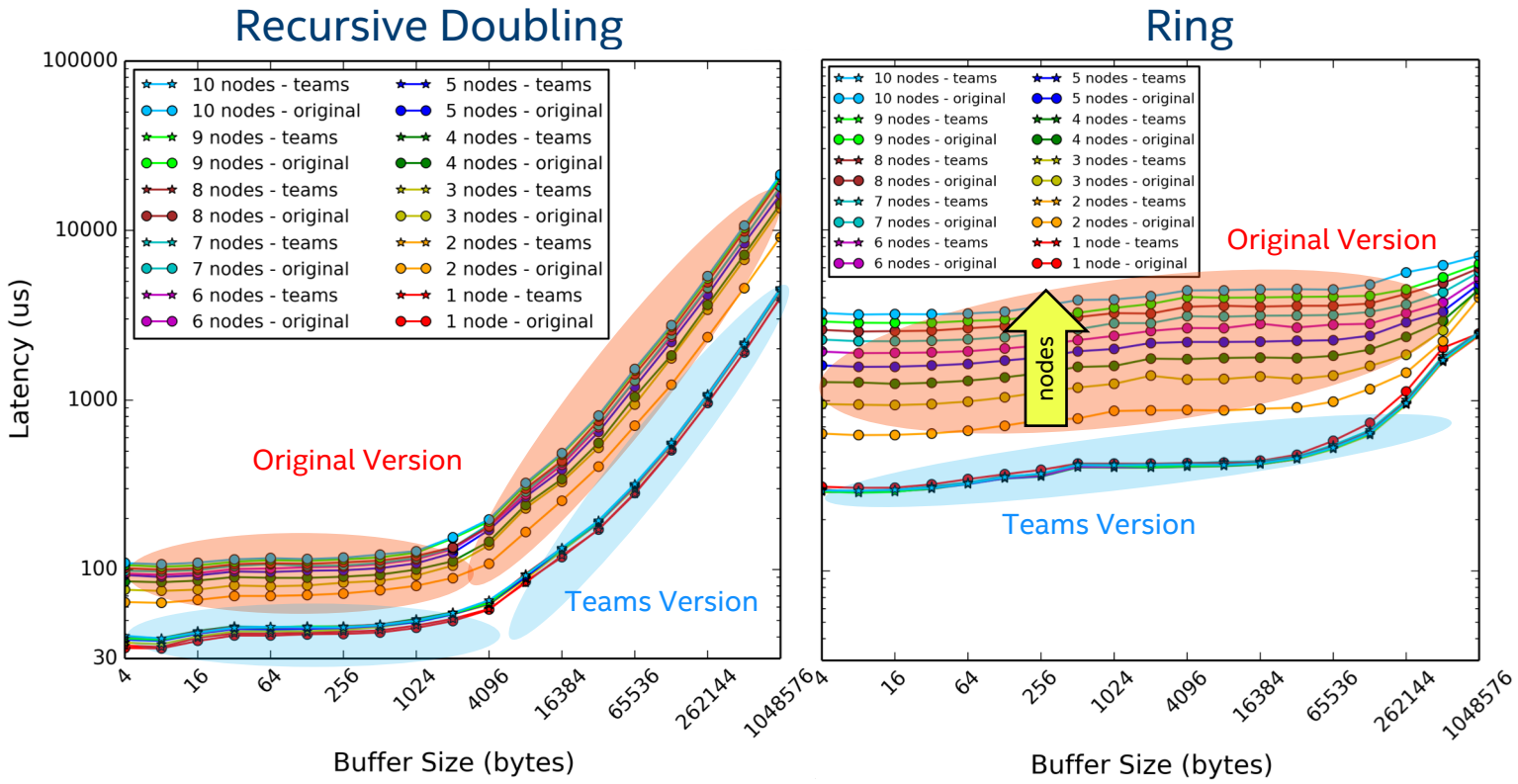
- 64 team creations (max supported) per iteration
- 1000 iterations – average time measured
- Blue – `TEAM_WORLD` "duplicate" split
- Red – *reduction* across `TEAM_WORLD`
- Green – `TEAM_WORLD` "halve" split
  - 1st split = {0, 1, 2, 3, …, N} (size=N)
  - 2nd split = {0, 2, 4, …, ~N}. (size=N/2)
  - 3rd split = {0, 4, 8, …, ~N} (size=N/4) etc.
- Yellow – *reduction* across the "halved" teams



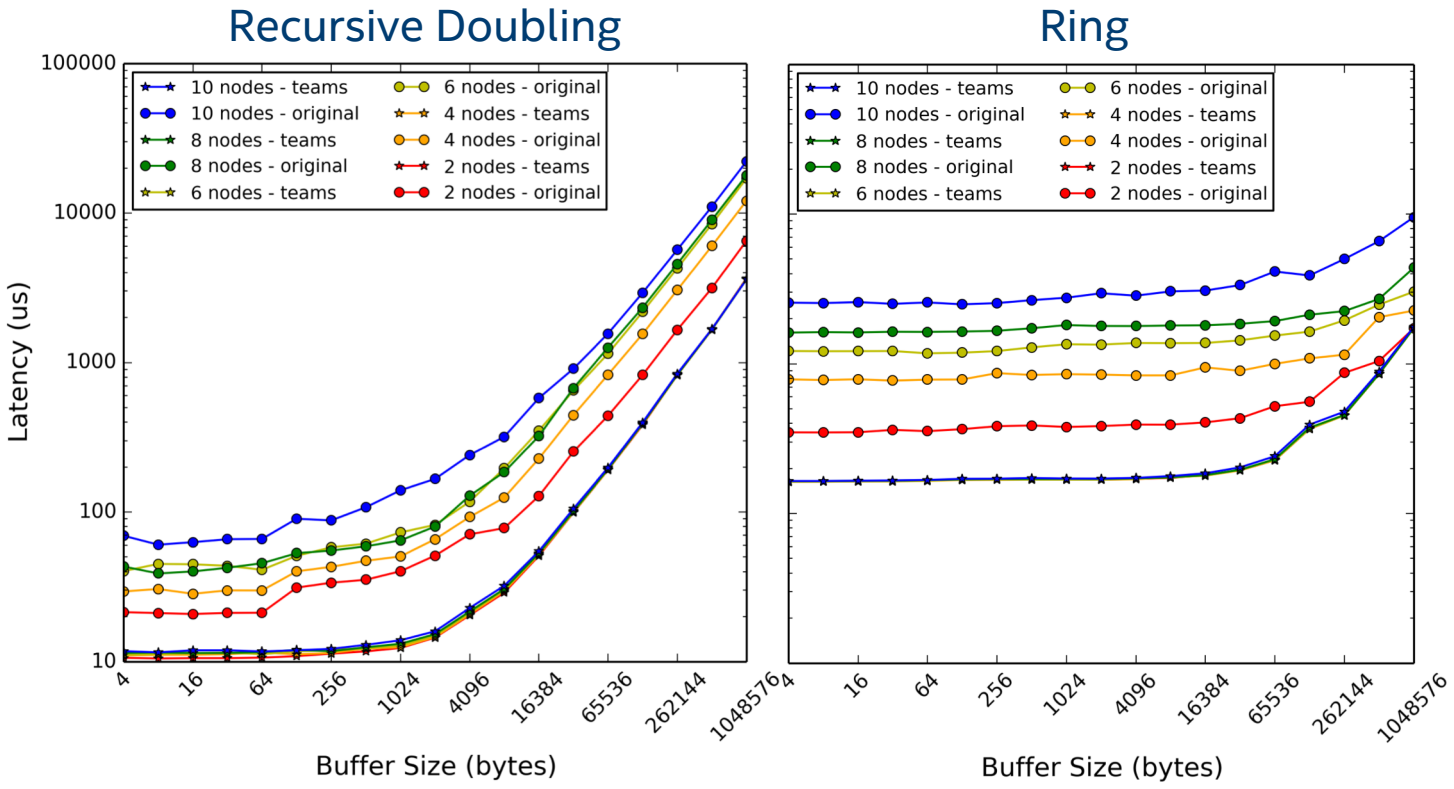
Diamond Sum-Reduction Measurements



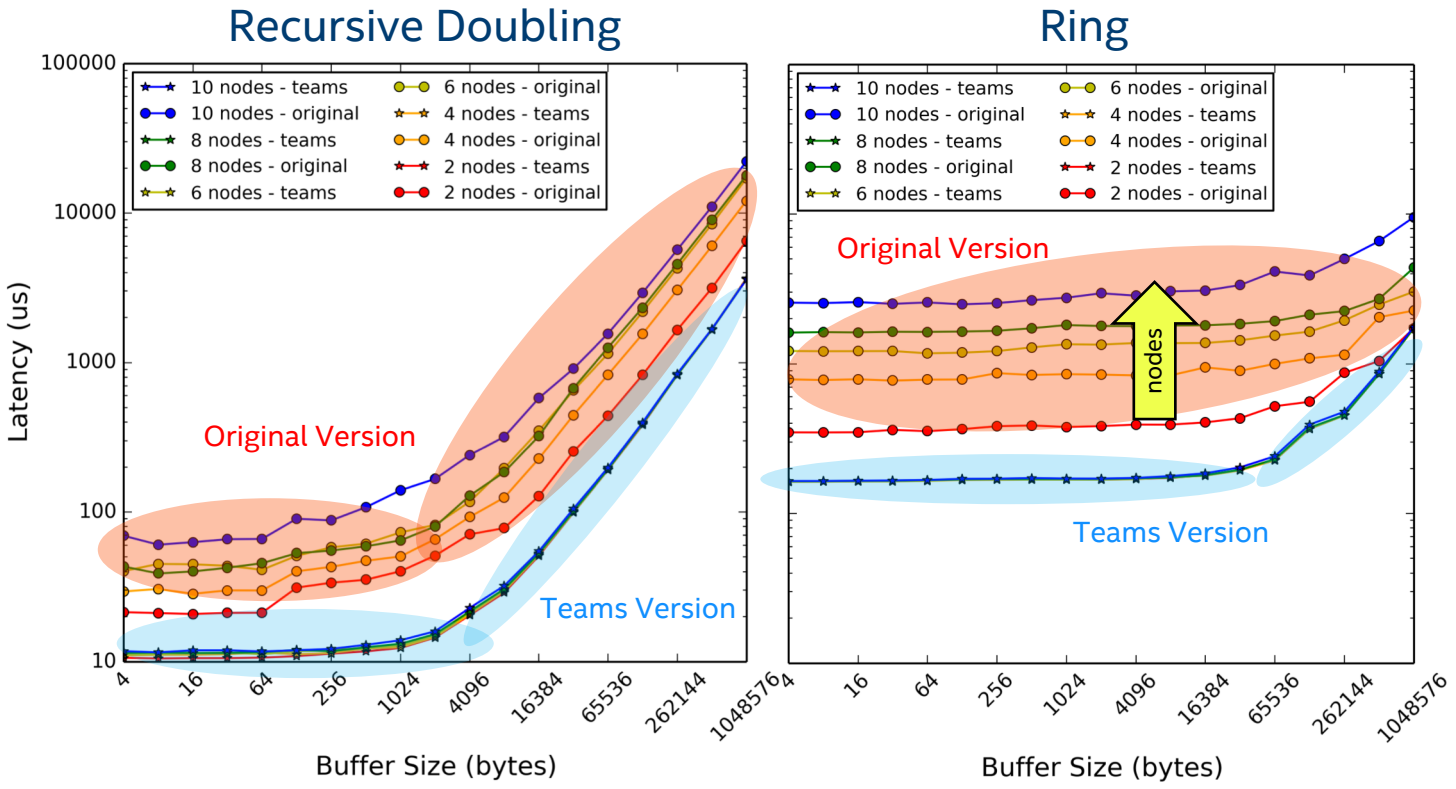
Diamond Sum-Reduction Measurements



Cori Sum-Reduction Measurements



Cori Sum-Reduction Measurements

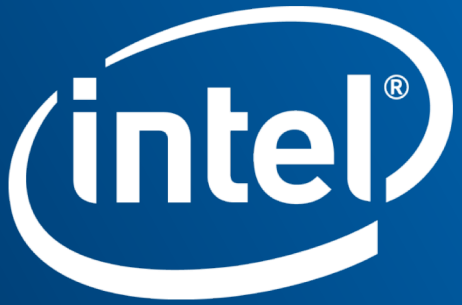


Conclusion

- The OpenSHMEM Teams API is impactful for applications: easier decomposition, better locality, simpler collectives usage.
- The latest API is implementable and shows good performance characteristics.
- From the experiments:
  - Team creation is dominated by a reduction.
  - Both bandwidth-optimized (ring) and latency optimized (rec. dbl.) algorithms benefit from team-based shared memory reduction.
- This code is under review within the SOS topic/teams branch:
  - https://github.com/Sandia-OpenSHMEM/SOS/pull/886
- Status:
  - Recently ratified by the specification committee – will be released in v1.5.

Future Work

- Support non-triplet (set-based) teams?
  - More general support for `SHMEM_TEAM_SHARED`.
  - May enable topological optimizations.
- Various teams-related extensions:
  - Memory Spaces
    - Teams-based memory allocation
    - Different traits for different memories
  - Atomicity domains
    - Isolate atomic operations to within a team (or team hierarchy)


intel®