OpenSHMEM Sets and Groups: An Approach to Worksharing and Memory Management

Ferrol Aderholdt, ORNL

Swaroop Pophale, ORNL

Manjunath Gorentla Venkata, ORNL

Neena Imam, ORNL

Current Look at Collectives in OpenSHMEM

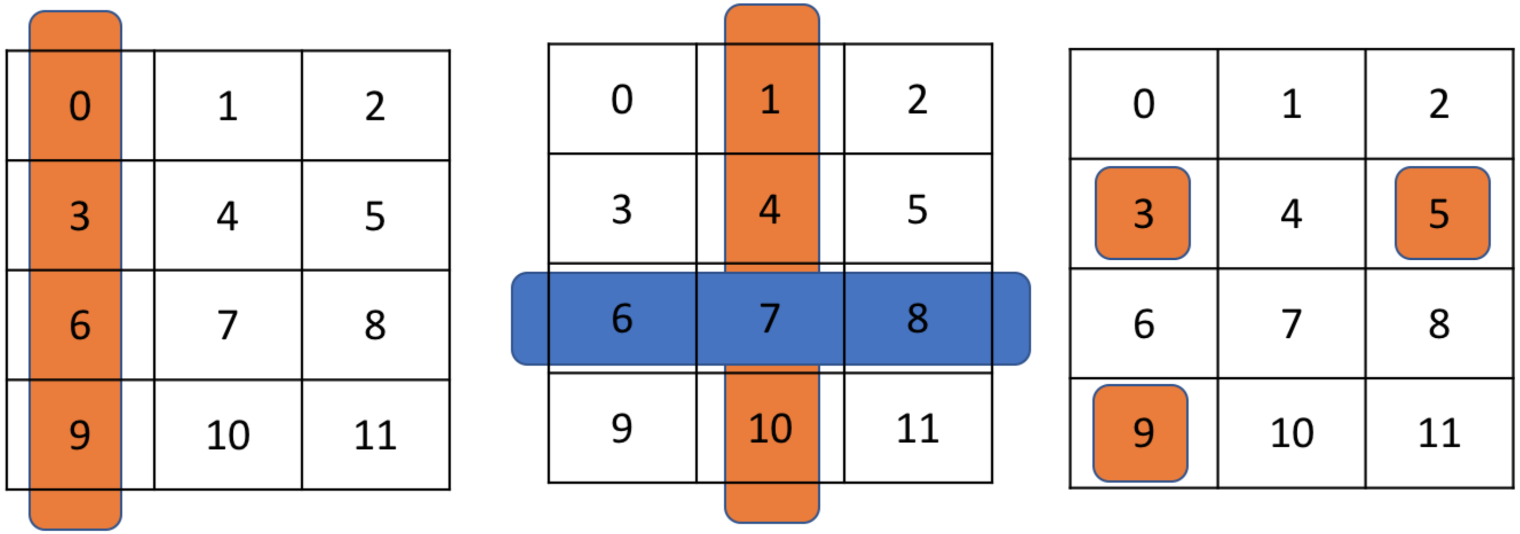
- Collective operations defined over active sets
  - Temporary grouping of PEs
  - Triple of information: starting PE, log base 2 stride, and size
  - Requires Users allocate and initialize their own resources for collectives
    - Synchronization (pSync) and scratchpad (pWrk) buffers
- Need to re-evaluate this concept for future systems
  - **Goal**: flexible, persistent groupings of PEs that have library managed resources

Possible solution?

- Teams
  - Grouping of PEs
  - Created using with strided, 2D, 3D, and color-based split
    - Collective operation over PEs of the parent team
  - Fulfills the persistent grouping of Pes from our goals
  - Did not have library managed resources
    - Users required to allocate and initialize resources
- Our solution: **Sets and Groups**
  - Decoupling of definition (i.e., Sets) and creation (i.e., Groups)
    - Creation collective postponed until needed, and only over Pes in the Group
  - Allocates and initializes resources on Users behalf
  - **Fulfills all of our goals**

Sets

- A group of PEs
  - Indexed based on positive integers monotonically increasing from zero
    - PEs not in the set have a negative index
- Opaque to users
- Created based on:
  - Strides and ranges
  - Modified with unions, intersections, and differences
- Enables:
  - Strided Sets
  - Multiple Strided Sets
  - Disjoint Sets



Sets

- Set creation is a local operation; requires no communication between PEs
  - Local operation: All PEs will calculate the same Set given valid initial Sets
    - Strides and ranges are simple calculations; Unions and Intersections result in the same set
  - Based on a parent Set
- Library provided sets: SHMEM_SET_WORLD, SHMEM_SET_EMPTY
- Set utility interfaces give the user the ability to:
  - Query the size of the set
  - Get the calling PE's index
  - Translate indices between Sets

Sets API (summary)

- Strided creation:
  - int **shmem_create_set_strided**(shmem_set_t * parent_set,

```
int shmem_create_set_strided(shmem_set_t * parent_set,
                             int index_start,
                             int index_stride,
                             int size,
                             shmem_set_t ** new_set);
```

- Range creation:

```
int shmem_create_set_range(shmem_set_t * parent_set,
                           int low_index,
                           int high_index,
                           shmem_set_t ** new_set);
```

Sets API (summary)

- Union/Intersection/Difference
  - ```
    int shmem_set_union(shmem_set_t * set1,
                        shmem_set_t * set2,
                        shmem_set_t ** new_set);
    ```
  - Intersection and Difference interfaces are similar…
    ```

Groups

- A grouping of a valid Set with the resources required for collective operations
  - Resources may include: pSync/pWrk buffers, hardware collective resources, etc.
- Opaque to the user
- Group creation is a collective operation
  - Collective over the Set
  - Ordering provided by the library
  - Blocking operation

Groups API (summary)

- Creation operations
  - Create Group from Set
  - Create Group using a color-based split
  - Duplicate valid Group
- Query operations
  - Determine the size of the group
  - Retrieve the Set the Group is based on

Example of the Difference Between Active Sets and Sets/Groups

```
/*Only odd PEs*/
if (my_pe % 2) {
    /*Some Work*/
    shmem_broadcast64(dest,src,4,0,1,1,(npes/2),pSync3);
    /*Some Work*/
    shmem_barrier(1,1,(npes/2),pSync1);
}
/*All PEs except PE 0*/
if (my_pe > 0) {
    /*Some Work*/
    shmem_broadcast64(dest,src,4,0,1,0,npes-1,pSync4);
    /*Some Work*/
    shmem_barrier(1,0,npes-1,pSync2);
}
shmem_alltoall64(dest,source,4,0,0,npes,pSync5);
```

- Performing 5 collectives over 3 active sets
  - Active Set of odd PEs
  - Active Set of PEs greater than 0
  - Active Set of all the World PEs

Example of the Difference Between Active Sets and Sets/Groups

```
/* Required - Allocating symmetric synchronization arrays */
long * pSync1 = (long *) shmem_malloc(sizeof(long) *
SHMEM_BARRIER_SYNC_SIZE);
long * pSync2 = (long *) shmem_malloc(sizeof(long) *
SHMEM_BARRIER_SYNC_SIZE);
long * pSync3 = (long *) shmem_malloc(sizeof(long) *
SHMEM_BCAST_SYNC_SIZE);
long * pSync4 = (long *) shmem_malloc(sizeof(long) *
SHMEM_BCAST_SYNC_SIZE);
long * pSync5 = (long *) shmem_malloc(sizeof(long) *
SHMEM_ALLTOALL_SYNC_SIZE);
...
/* Required - Initializing symmetric synchronization arrays */
for (int i = 0; i < SHMEM_BARRIER_SYNC_SIZE; i++) {
    pSync1[i] = SHMEM_SYNC_VALUE;
    pSync2[i] = SHMEM_SYNC_VALUE;
}
for (int j = 0; j < SHMEM_BCAST_SYNC_SIZE; j++) {
    pSync3[j] = SHMEM_SYNC_VALUE;
    pSync4[j] = SHMEM_SYNC_VALUE;
}
for (int k = 0; k < SHMEM_ALLTOALL_SYNC_SIZE; k++) {
    pSync5[k] = SHMEM_SYNC_VALUE;
}
```

Allocate pSync arrays

Initialize pSync arrays

Example of the Difference Between Active Sets and Sets/Groups

```
/*Only odd PEs*/
if(my_pe % 2){
    /*Some Work*/
    shmem_group_broadcast64(dest, src, 4, 0, group_odd);
    /*Some Work*/
    shmem_group_barrier(group_odd);
}
/*All PEs except PE 0*/
if(my_pe > 0){
    /*Some Work*/
    shmem_group_broadcast64(dest, src, 4, 0, group_gz);
    /*Some Work*/
    shmem_group_barrier(group_gz);
}
shmem_group_alltoall64(dest, source, 4, group_world);
```

- Becomes 5 collectives with 3 Groups
- Does not really look like much changed…

Example of the Difference Between Active Sets and Sets/Groups

```
shmem_set_t * set_odd, * set_gz;
shmem_group_t * group_odd, * group_gz, * group_world;

shmem_create_set_strided(SHMEM_SET_WORLD, 1, 2, npes / 2, &set_odd);
shmem_create_set_range(SHMEM_SET_WORLD, 1, npes - 1, &set_gz);

shmem_create_group_from_set(set_odd, &group_odd);
shmem_create_group_from_set(set_gz, &group_gz);
shmem_create_group_from_set(SHMEM_SET_WORLD, &group_world);
```

- More has changed here
  - User no longer needs to allocate and initialize the buffers used for the pSync arrays
  - User creates 2 Sets and 3 Groups
    - Allocation and Initialization handled by library

Sets Implementation

- Implementation focuses on minimizing memory usage
  - Stores Set as a stride or list of strides depending on mathematical relationship between Pes
    - E.g., A Set with Pes {0, 1, 2, 3} is stored as a strided Set with a stride of 1; A set with Pes {0, 2, 3, 4} is stored as a list of strides (2)

- Each modification of a set (i.e., Union, Intersection, Difference) will attempt to collapse the Set to a simpler representation
  - The union of sets {0, 2, 3, 4} and {0, 1, 2, 3, 4} results in a single, strided set {0, 1, 2, 3, 4}

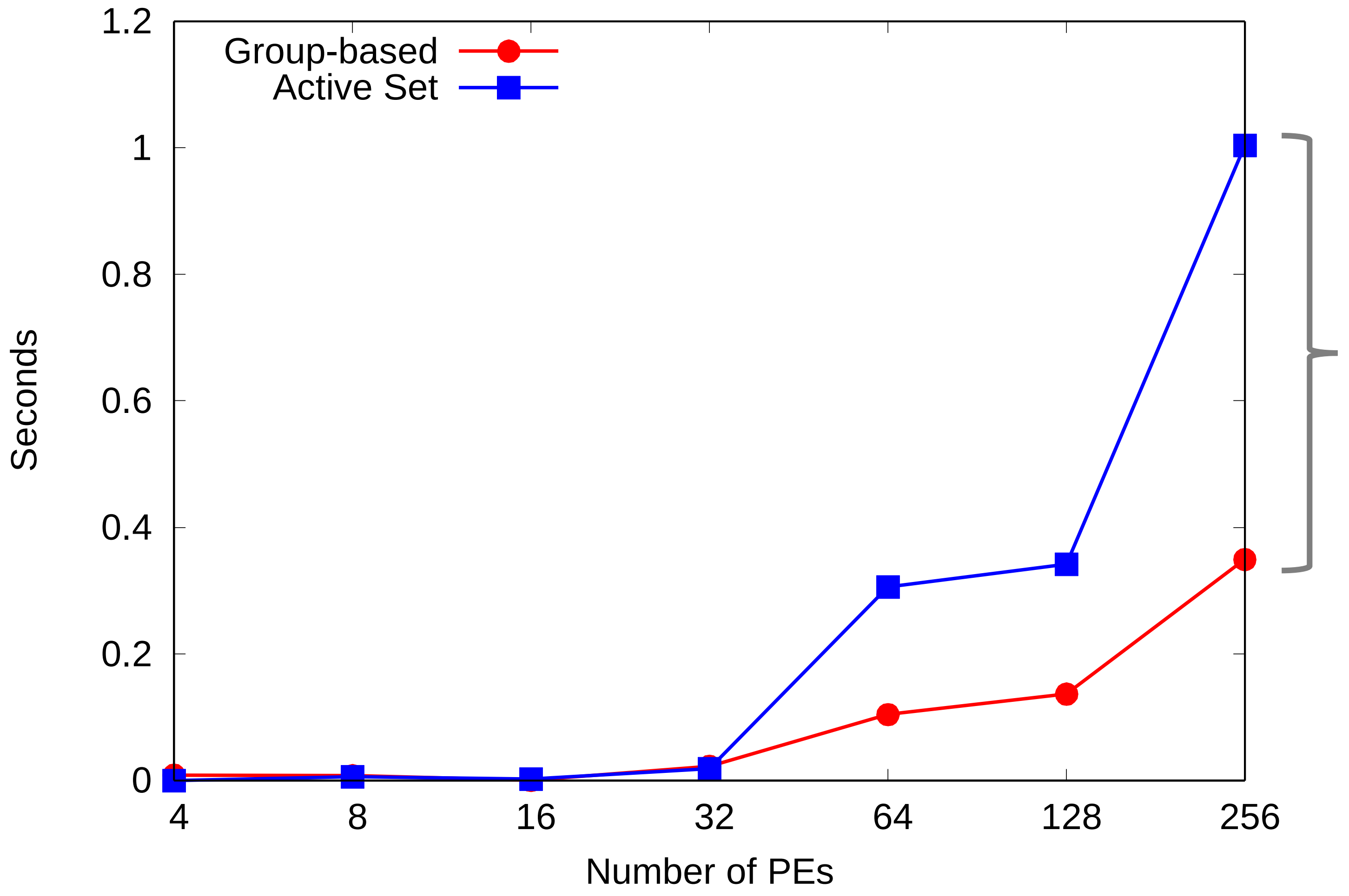
Groups Implementation

- Extended collectives interfaces to support Groups
  - Barrier, Reductions, Collect, etc.
    - As a proof of concept and for the sake of comparison, implemented algorithms are identical
- Implementation of Groups as proof of concept
  - Creation operations linear in nature
    - Can be improved with respect to performance

Evaluation of the Example

- Testbed: 16 node Turing cluster at ORNL
  - Each node: two Intel Xeon processors with 20 logical cores, 128 GB of RAM and a ConnectX-4 InfiniBand interconnect
- Evaluation: Measure the overall execution time
  - Gives us an initial understanding of the performance implications of Sets and Groups

Evaluation of the Example

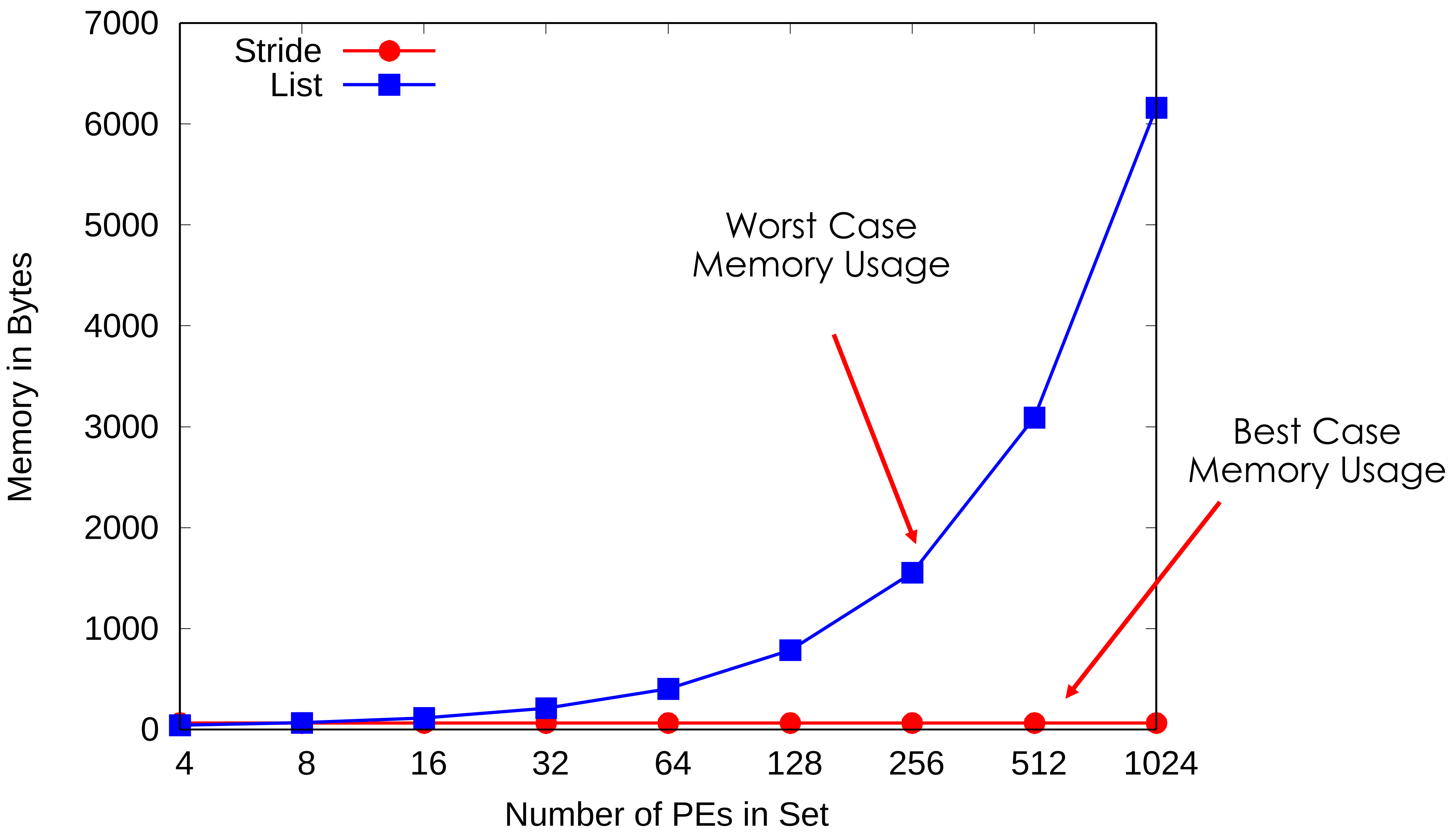


1/3 of the execution time

Experimental Evaluation

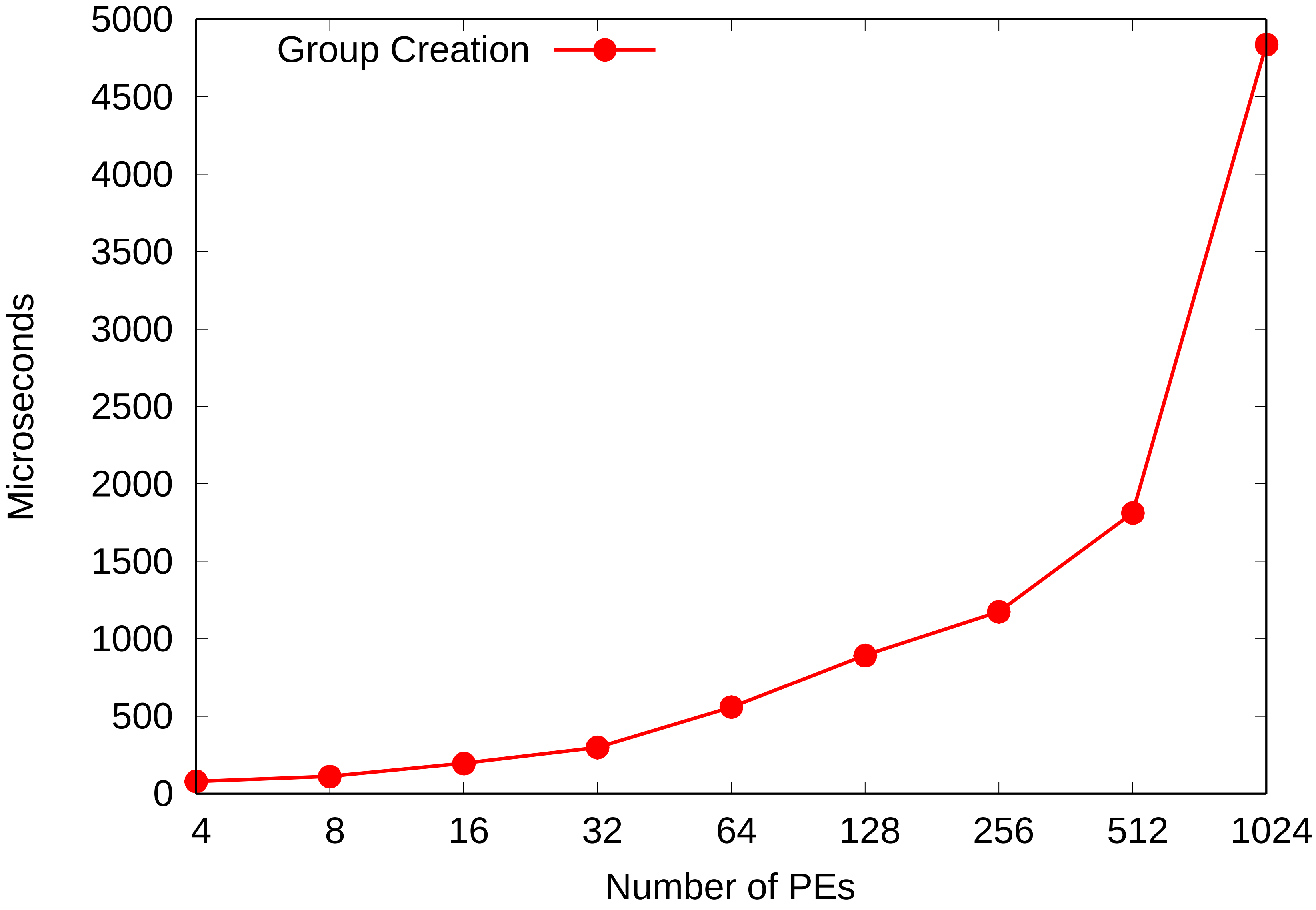
- Goals of evaluation
  1. Evaluate Sets memory usage requirements
  2. Overhead of collectives using Groups
  3. Demonstration of Sets/Groups with All-Pairs Shortest Path with a synthetic dataset
- Testbeds at OLCF:
  - Eos (1 & 2)
    - Each node: two 8-core Intel Xeon processors, 64 GB of memory, and Aries interconnect
  - Titan (3)
    - Each node: one 16-core AMD Opteron processors, 32 GB of memory, and the Gemini interconnect

Evaluation: Sets Memory Usage based on representation

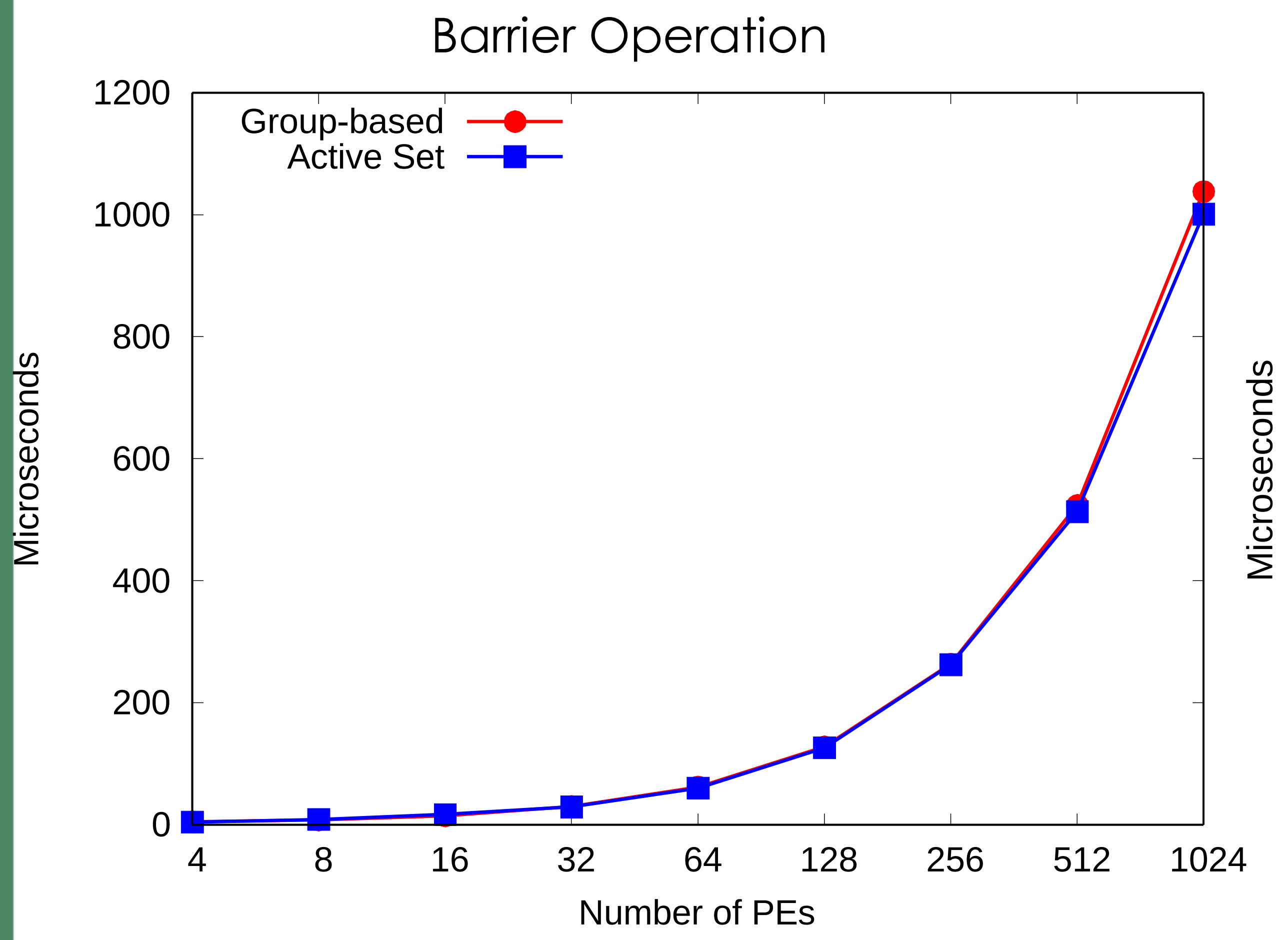


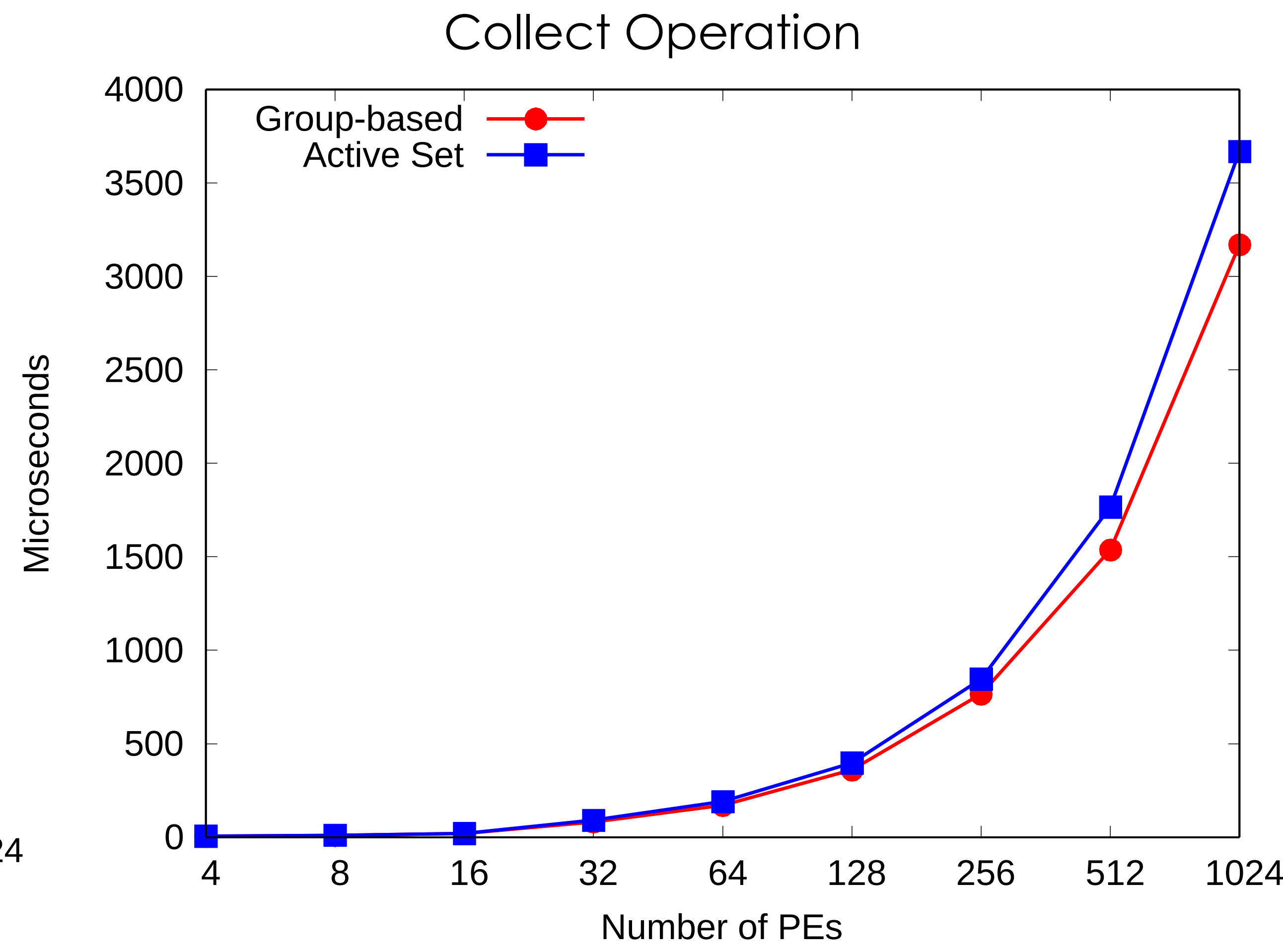
OAK RIDGE National Laboratory

Evaluation: Group Creation Overhead



Evaluation: Micro-benchmarks of Collective Operations





Use Case: All-Pairs Shortest Path

- Briefly Review All-Pairs Shortest Path (APSP)
  - Fundamental Graph Problem
  - Goal: Find the shortest path between each pair of vertices in a graph
  - Results can be used to find the Betweeness Centrality in a graph
  - Multiple methods of solving this problem:
    1. Iterate over sources using SSSP while filling in a graph
    2. Use dynamic programming (i.e., Floyd-Warshall)

Use Case: All-Pairs Shortest Path

- Briefly Review All-Pairs Shortest Path (APSP)
  - Fundamental Graph Problem
  - Goal: Find the shortest path between each pair of vertices in a graph
  - Results can be used to find the Betweeness Centrality in a graph
  - Multiple methods of solving this problem:
    1. **Iterate over sources using SSSP while filling in a graph**
       - **SSSP easily parallelizable (e.g., Bellman-Ford)**
       - **Can further parallelize APSP**
    2. Use dynamic programming (i.e., Floyd-Warshall)

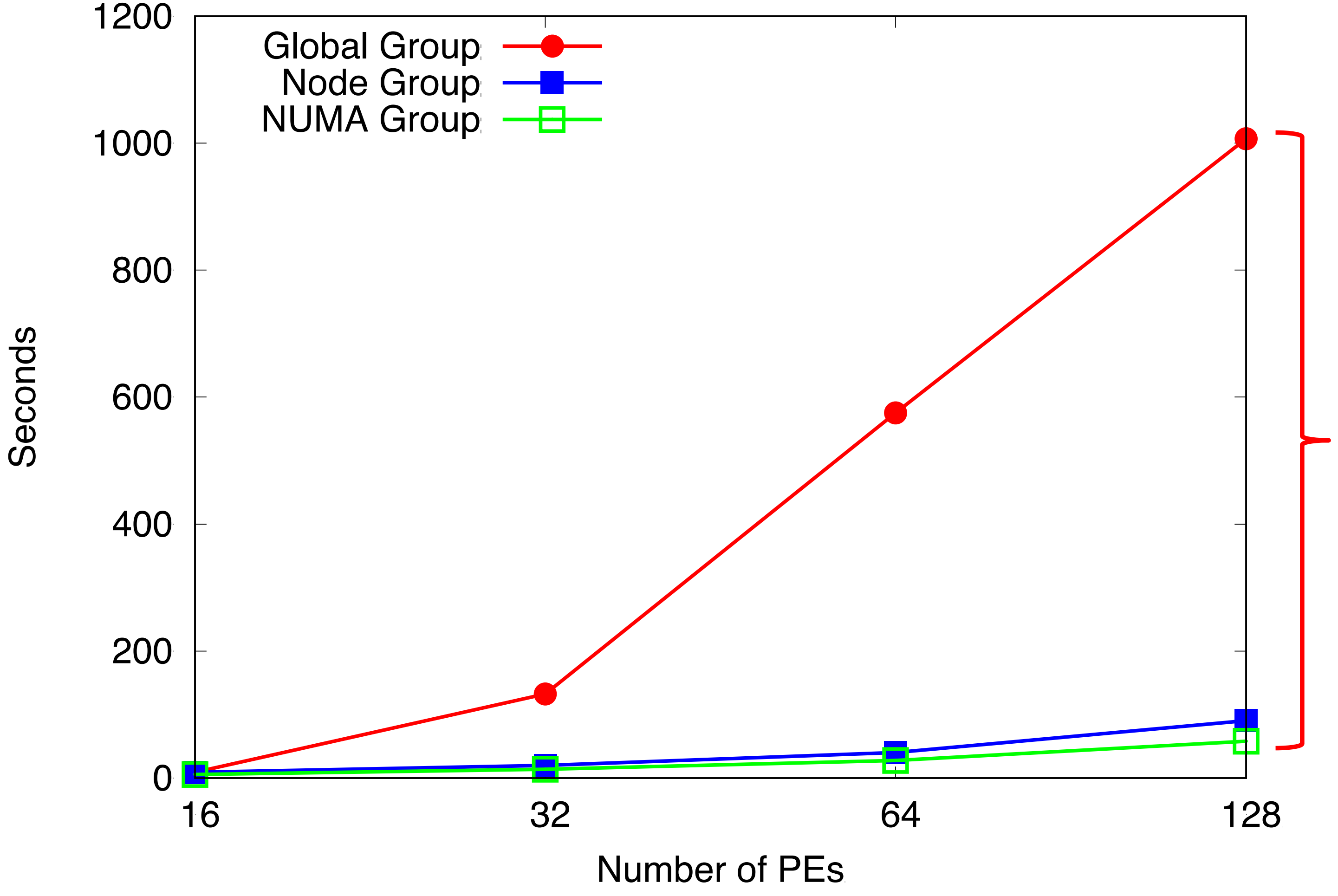
Parallelizing APSP with OpenSHMEM

- We parallelized SSSP (i.e., Bellman-Ford/Dijkstra’s) with OpenSHMEM
- For APSP, we simply:
  - Partition the Pes into Sets/Groups where each Set will have a roughly uniform amount of source vertices
  - Each Group has its own copy of the Graph
  - Each Group can then:
    - Iterates over their sources of the graphs performing the parallel SSSP with OpenSHMEM on the source
    - The resulting distance/predecessor array is stored on stable storage to build the distance matrix
- Work distribution can be simplified:
  - Work queue with Groups taking a source to work on when completing their own

Experimental Evaluation

- Dataset
  - Recursive-Matrix (R-MAT) Graphs
    - Synthetic, scale-free Graphs
    - Parameters for generation
      - A = 0.57, b = 0.19, c = 0.19, and d = 0.05
      - Average vertex degree of 16
    - Using similar scale to Graph500 graphs
      - i.e., scale=10 means 2^{10} vertices
    - Similar to a social network graph
- Evaluation
  - Weak scaling with an initial scale of 10

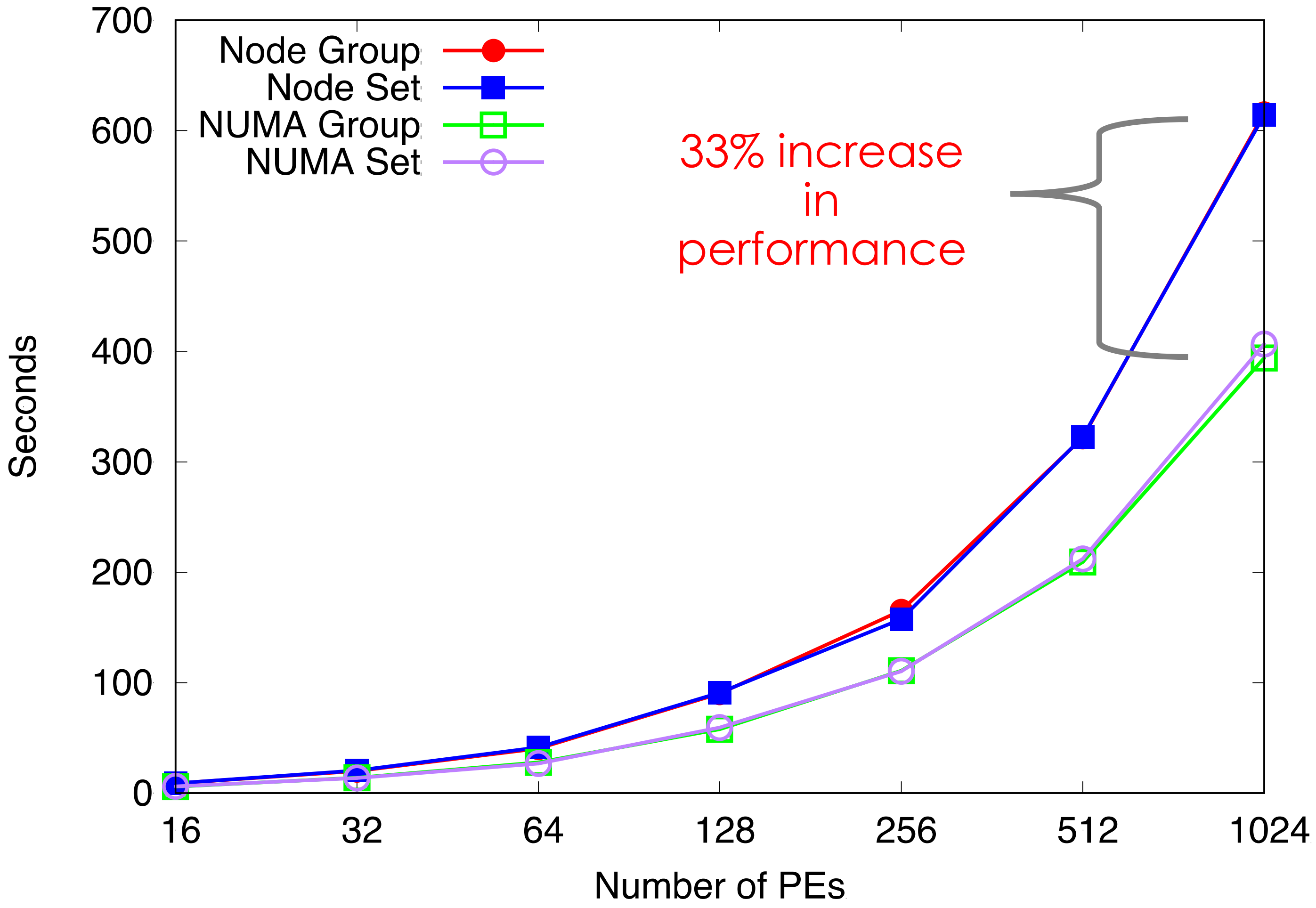
Weak Scaling Performance of Sets/Groups on R-MAT Graphs (initial scale=10)



Global Group is the same as World

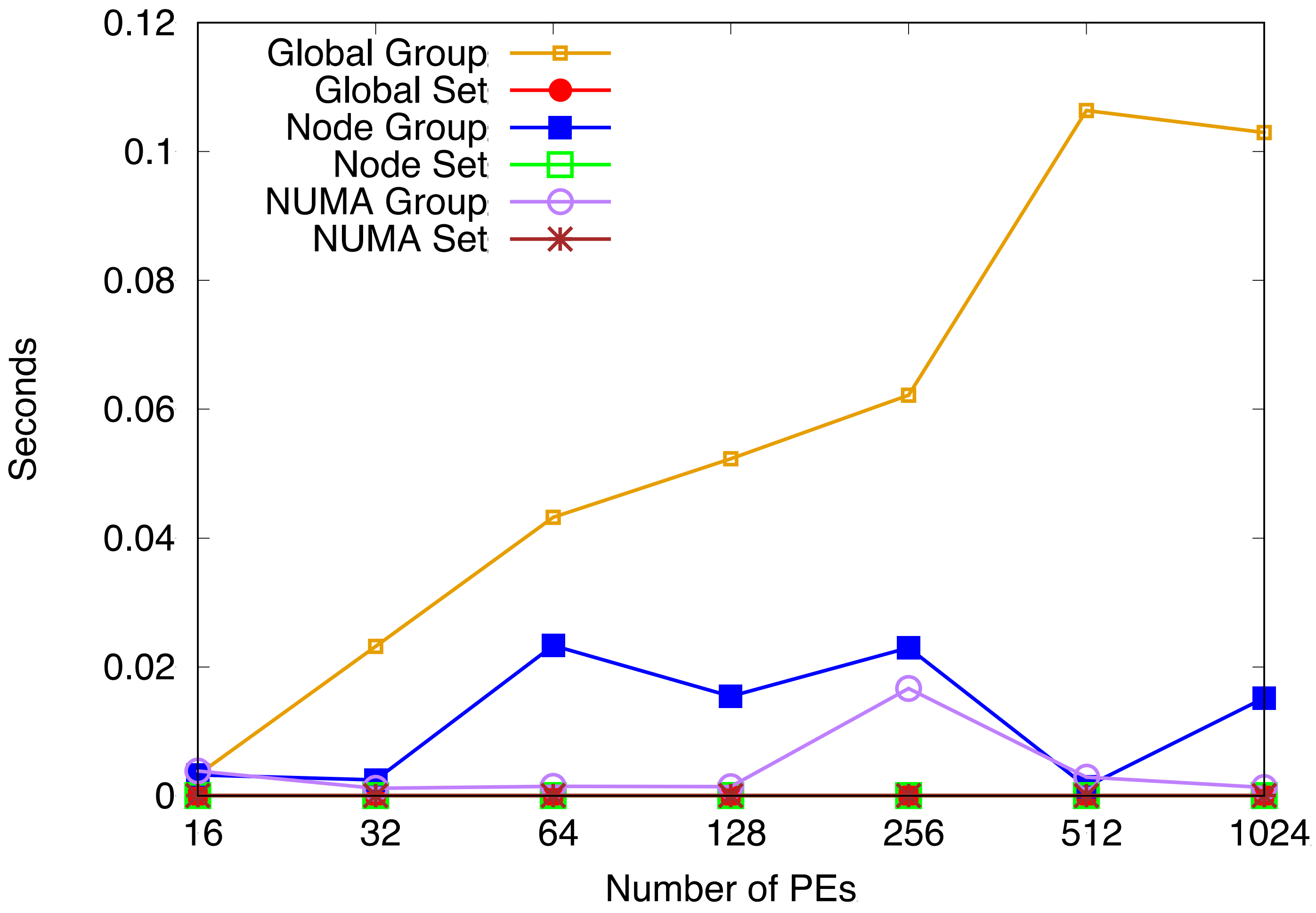
94% difference in performance

Weak Scaling Performance of Sets/Groups on R-MAT Graphs (cont.)



- Sets are using pSync/pWrk arrays allocated using shmem_malloc()
- Groups/Sets have similar performance
- Obvious observation:
  - Grouping of PEs on a single NUMA is best performing, but not always possible

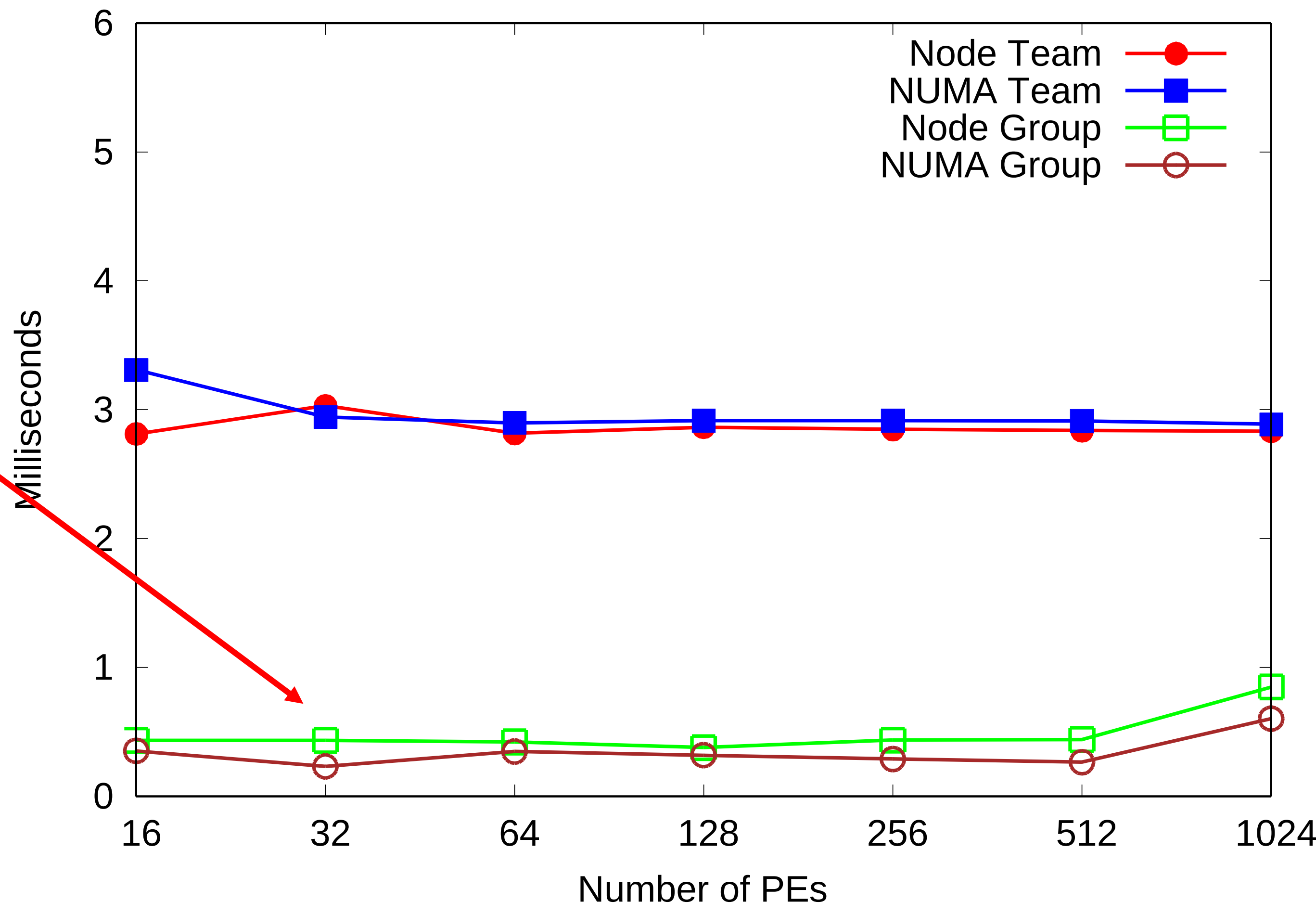
Set/Group Creation Overhead



- Important note: Group creation is expensive
  - It may be more performant to leverage Sets if you know you are going to have global allocations regardless of logical PE groupings
  - Also, if we were to performance asynchronous SSSP as demonstrated in previous paper, no need for Groups
    - All operations are Puts/Gets without collectives

Comparison of Sets/Groups with Cray SHMEM

- Set + Group creation compared with Cray SHMEM Team creation
  - 92% decrease in latency
  - Increased performance due to Group creation being local to its Set
    - Node and NUMA group creation takes advantage of shared memory



Conclusion

- Active Sets provide a useful, but temporary and limiting abstraction for collective operations
- Discussed the Sets/Groups abstractions as a possible replacement for Active Sets with library managed resources
- Demonstrated
  - Flexibility and utility of Sets and Groups
  - Negligible performance overhead related to Groups implementations
  - Multiple-levels of parallelism with All-Pairs Shortest Path and Sets/Groups

Conclusion

- Active Sets provide a useful, but temporary and limiting abstraction for collective operations
- Discussed the Sets/Groups abstractions as a possible replacement for Active Sets with library managed resources
- Demonstrated
  - Flexibility and utility of Sets and Groups
  - Negligible performance overhead related to Groups implementations
  - Multiple-levels of parallelism with All-Pairs Shortest Path and Sets/Groups
- Hopefully jump started the conversation regarding Teams after lunch!

Acknowledgements

This work was supported by the United States Department of Defense (DoD) and used resources of the Computational Research and Development Programs at Oak Ridge National Laboratory.

Questions?