Studies in Bucket Exchange Communication with ISx

Jacob Hemstad*
University of Minnesota
hemst013@umn.edu
Ben Harshbarger
Cray Inc.
bharshbarg@cray.com
Ulf Hanebutte
Intel Corporation
ulf.r.hanebutte@intel.com
Brad Chamberlain
Cray Inc.
bradc@cray.com

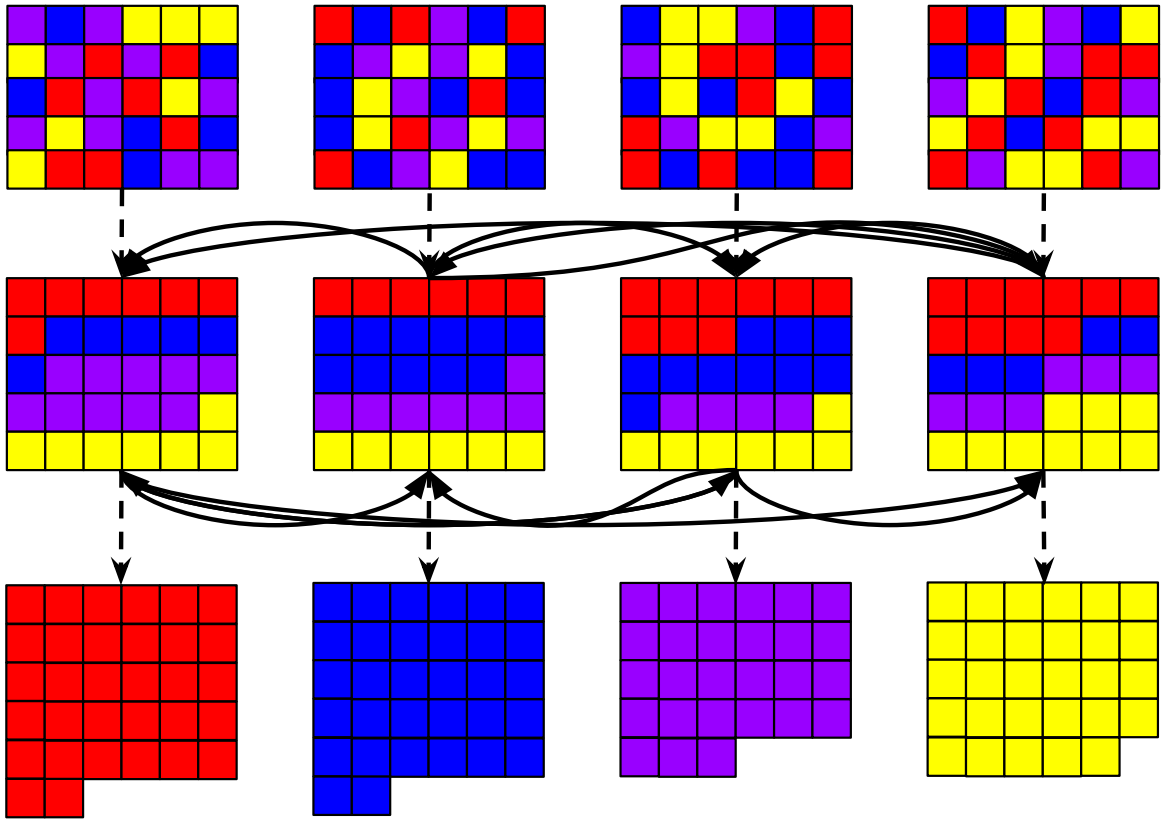
Bucket Sort

Random Key Distribution

Arrange into Local Buckets

Exchange Data

Local Sort



ISx - Bucket Sort Mini-Application

- Successor to the NAS Integer Sort application
  - Solves issues of ineffective load balancing, limited problem sizes, and inability to weak scale
- Uniform random key generation (work balance is guaranteed by default)
- Supports arbitrary problem sizes
- Strong and Weak Scaling
- Automatic solution verification
- Implementations
  - OpenSHMEM*
  - MPI 2-sided*
  - Chapel (available since version 1.13)

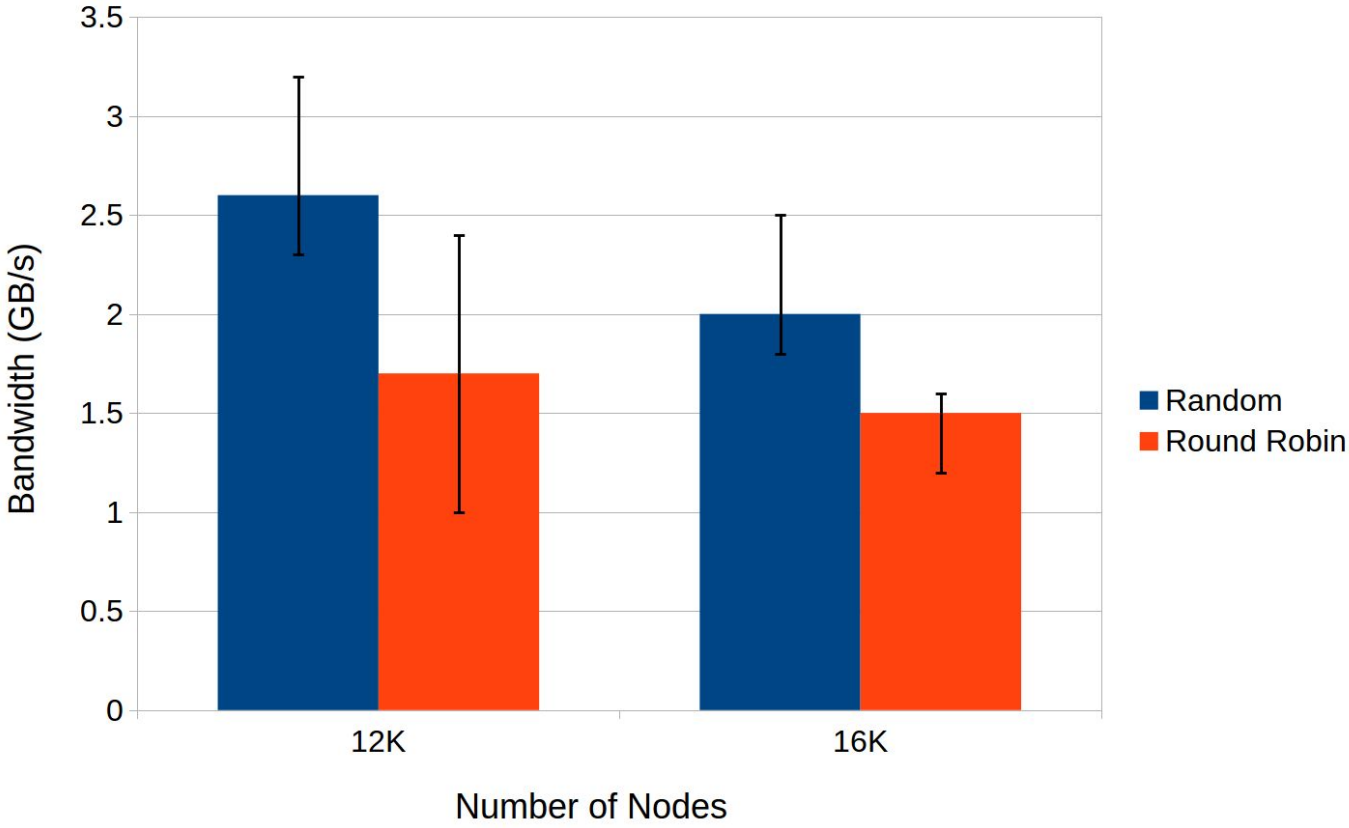
*https://github.com/ParRes/ISx

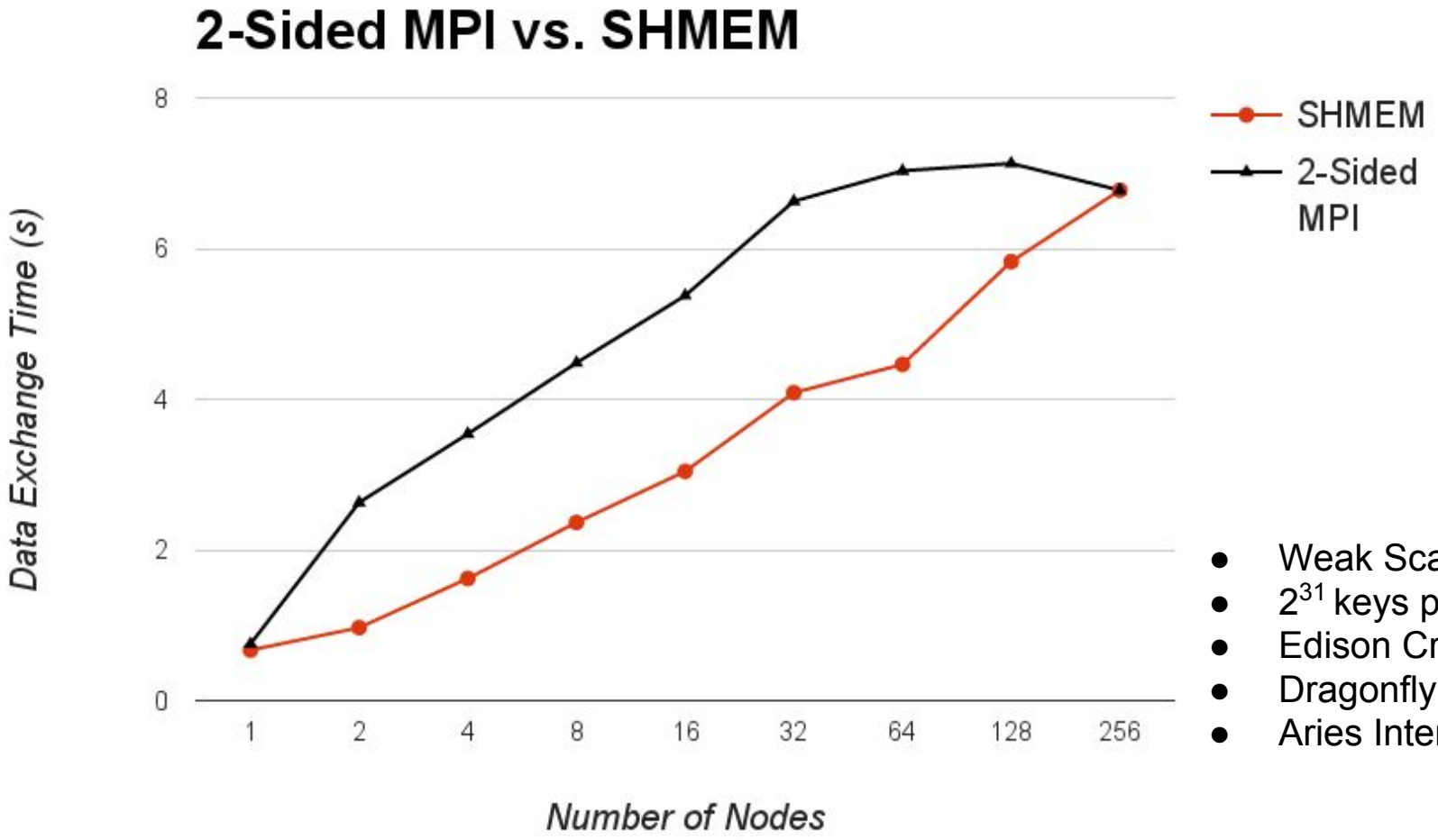
SHMEM Communication Strategy Comparison

Random
- Every rank sends to other ranks in random order

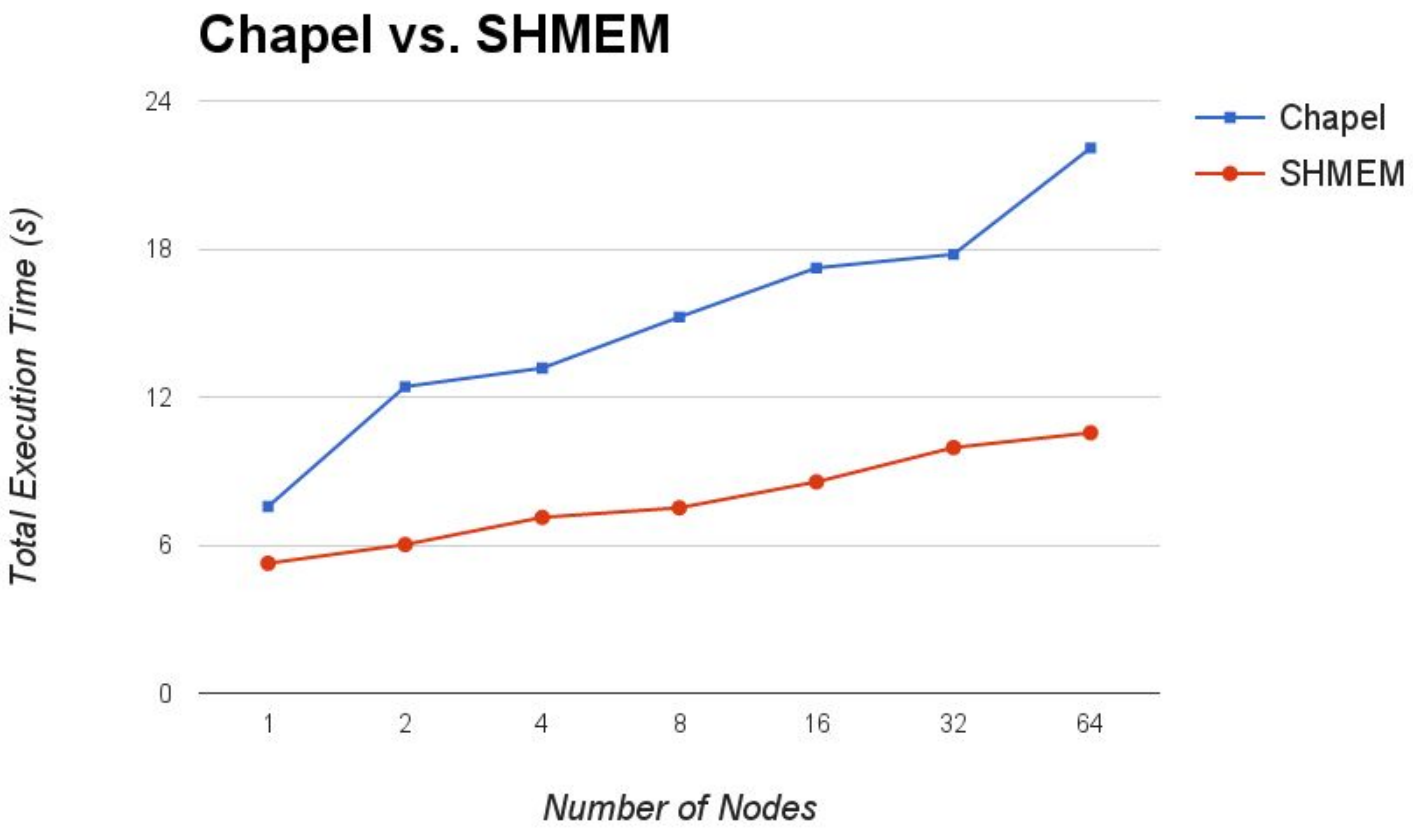
Round Robin
- Rank *i* sends to *i+1, i+2,* etc. with wrap around

Results
- Edison Cray XC30
- Aries Interconnect w/ Dragonfly Topology
- Average of 40 runs across 2 unique job placements
- Average with min/max bounds





- Weak Scaling
- 2^{31} keys per node
- Edison Cray XC30
- Dragonfly Network
- Aries Interconnect

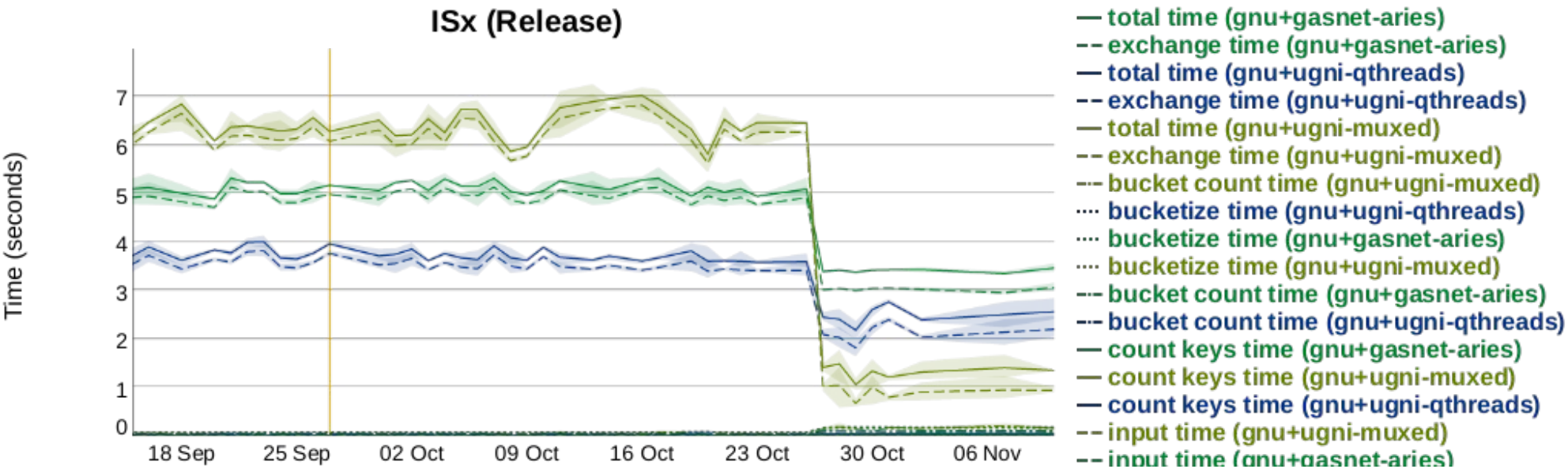


Array Slicing:

```
destination[dest_id][offset..#size]
=
source[offset..#size];
```

- Weak Scaling
- 2^{27} keys per node
- Cray XC30 internal to Cray

Chapel Performance Today



Manually disabling reference counting no longer needed

Wrapping Up & Looking Forward

- Studies to date:
  - Random vs. Round Robin
    - Better performance with random data exchange on Edison
  - SHMEM vs. 2-sided MPI
    - Better performance with SHMEM up to network saturation
  - SHMEM vs. Chapel
    - Optimizations to Chapel reduce sources of overhead and bring performance closer to SHMEM
- Future:
  - Threaded implementation
  - One-sided MPI
  - Integrate performance counter metrics and communication model
  - Specialized