Extending Strided Communication Interface in OpenSHMEM

Naveen Namashivayam, Dounia Khaldi, Deepak Eachempati, Barbara Chapman

University of Houston

{nravi, dounia, dreachem, chapman}@cs.uh.edu

August 5, 2015

Introduction and Motivation

- Data transfers in OpenSHMEM:
  Contiguous vs Non-contiguous
- Utilization vs Implementations – how optimized are the strided API routines in different implementations?
- Support different stride orientations?
- Support for aggregate data types – e.g. array of structures?
- Existing proposals and extensions for strided routines

Strided Communication Routines

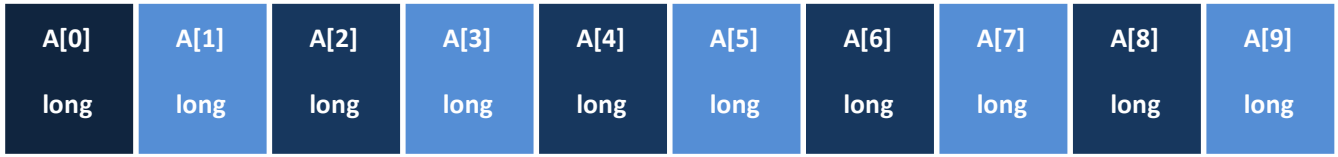
- **shmem_TYPE_iget** and **shmem_TYPE_iput** routines are used to remotely access strided data
- All RMA routines usually take source (**src_ptr**) and destination pointers (**dest_ptr**) along with the remote PE (**pe_id**)
- Strided routines in particular takes stride size (**src_stride, dest_stride**) and number of elements (**nelems**)
- Fixed to a particular data type (**short, int, long, double, float, long long, long double**)

```
shmem_double_put(dest_ptr, src_ptr, nelems, pe_id);

shmem_double_iput(dest_ptr, src_ptr, dest_stride, src_stride,
                  nelems, pe_id);
```

Assessment of Strided Routines

long A[10]



- Baseline algorithm – implement single strided access as multiple contiguous RMA accesses
- Alternative 1:
  Use network Scatter-Gather
- Alternative 2:
  Software approach - pack/unpack

```
void shmem_long_iput(
    long *dest_ptr,
    const long *src_ptr,
    ptrdiff_t dest_stride,
    ptrdiff_t src_stride,
    size_t nelems, int pe)
{
    for (int i = 1; i <= nelems; i++) {
        shmem_long_put(dest, src,
                nelems,
                pe_id);
        dest_ptr += dest_stride;
        src_stide += src_stride;
    }
}
```

Purpose of Assessment and Comparison Details

- Compared 6 different OpenSHMEM Implementations

1. Cray SHMEM in Cray MPT	2. SGI SHMEM in SGI MPT
3. UH SHMEM – GASNet	4. UH SHMEM – UCCS
5. OpenSHMEM in Open MPI	6. OpenSHMEM in MVAPICH2-X

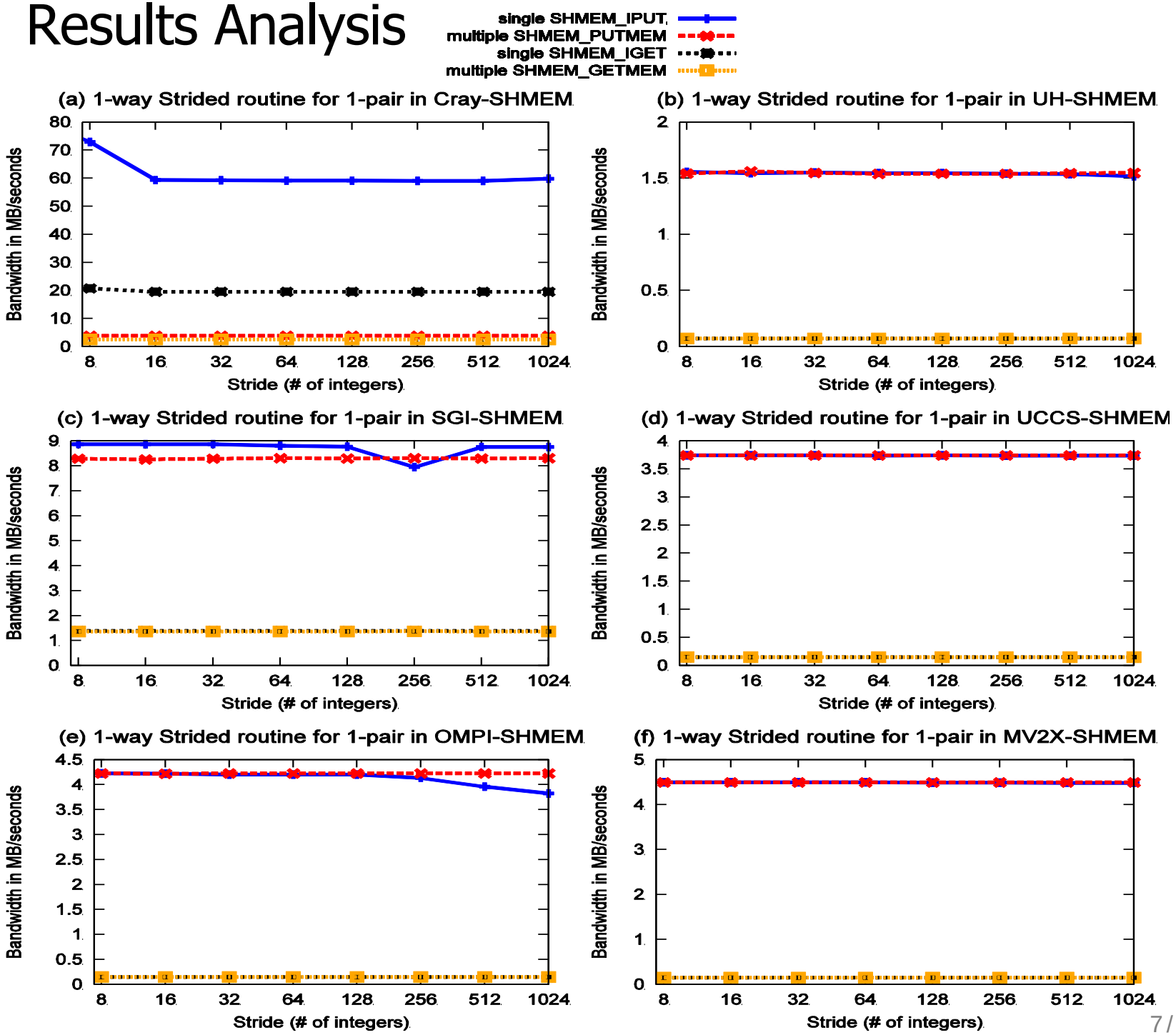
- Purpose of the assessment
  - Not to compare different implementations
  - But to compare strided routines in different implementations against the baseline algorithm

- Compared **shmem_TYPE_iput** and **shmem_TYPE_iget** with multiple **shmem_TYPE_put** and **shmem_TYPE_get** calls

Experimental Setup

Name	Nodes	Cores per Node	Processor Type	Interconnect
Cray XC 30	64	32	Intel Xeon E5 Sandy Bridge	Dragonfly Interconnect with Aries
SGI Altix	12	12	Intel Xeon X5660	Mellanox ConnectX-2 QDR HCA 1 port
Whale	81	8	AMD Opteron	4xInfiniBand DDR 2012 switch

Results Analysis



Different Stride Orientations – 1

- Contiguous data of same basic data type (yes, with TYPE_put/get)

- Element-wise strided data of same basic data type (yes, with TYPE_iput/iget)

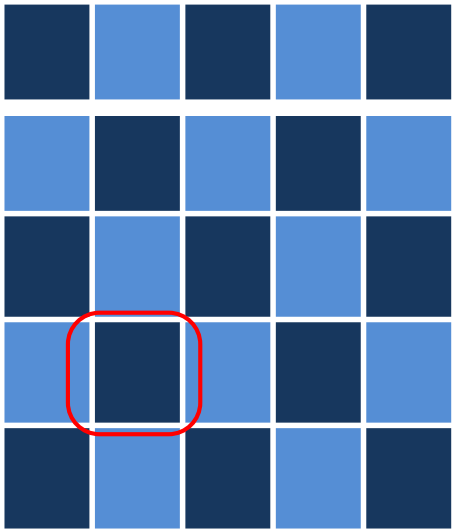
- Block-wise strided data of same basic data type

- Contiguous data of structure data type (yes, with putmem/getmem)

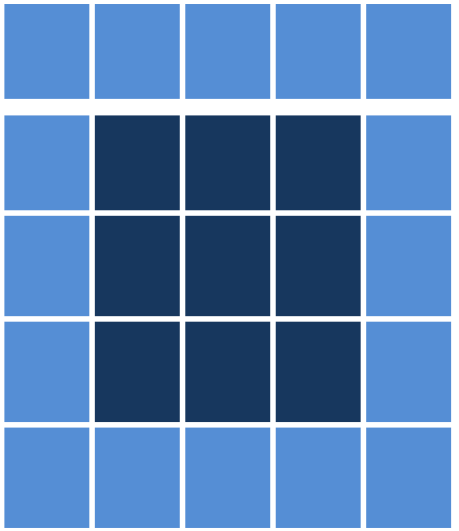
- Strided data of structure data type – strides should be blocks

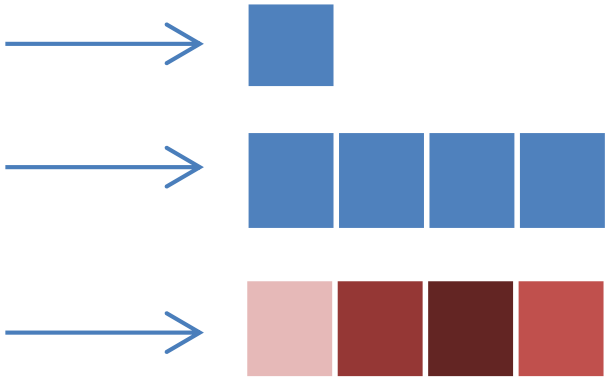
Different Stride Orientations – 2

(a) Checker Board Orientation



(b) Matrix Orientation





- Single element
- Group of elements of same data type
- Group of elements of different data types

Proposal for Generic Strided Routines

- Generic strided routine – support aggregate data types (e.g. blocks of basic-type elements, structure types)
- Similar to **shmem_putmem** and **shmem_getmem**
- **src_stride, dest_stride** and **blksize** specified in bytes
- **nblks** being the number of blocks

```
void shmem_iputmem (void  *dest_ptr, const void  *src_ptr,
                    ptrdiff_t dest_stride, ptrdiff_t src_stride,
                    size_t blksize, size_t nblks, int pe_id);

void shmem_igetmem (void  *dest_ptr, const void  *src_ptr ,
                    ptrdiff_t dest_stride , ptrdiff_t src_stride,
                    size_t blksize, size_t nblks, int pe_id);
```

Implementation for shmem_iputmem

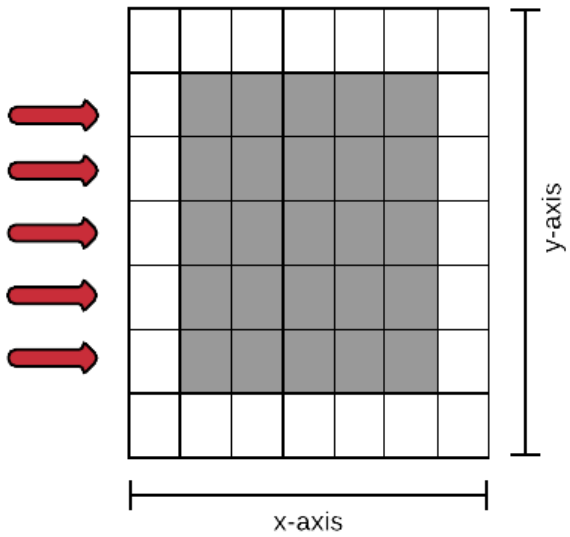
- Different Algorithms, Hardware and Software Techniques
- Implement on top of existing OpenSHMEM routines
- **Algorithm 1*** – Use **shmem_putmem**
- **Algorithm 2*** – Use combinatons of **shmem_iput128, shmem_iput64,** and **shmem_iput32**
- **Algorithm 3** – Uses a predicate table to select between Algorithm 1 and Algorithm 2

* use corresponding OpenSHMEM **get** routines for shmem_igetmem

Algorithm 1: Using shmem_putmem


```
for ( i = 1, i ≤ nelems, i++) do
    shmem putmem(dest_ptr, src_ptr, blksize, pe_id)
    dest_ptr += dest_stride
    src_ptr  += src_stride
end for
```

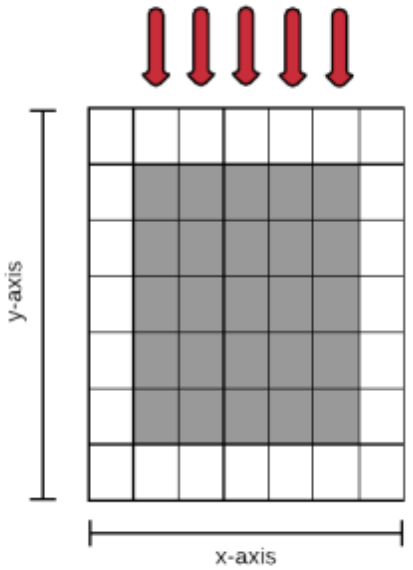
- Implementing using multiple **shmem_putmem**
- Call **shmem_putmem** along the y-axis
- Number of **shmem_putmem** will be equal to the number of elements (**nelems**)
- More number of SHMEM calls
- Normal implementation – support all possible **src_stride**, **dest_stride** and **blksize**



Implementation – Algorithm 2

```
if (check whether shmem_iput32 can be used)
  if (check whether shmem_iput128 can be used)
    call shmem_iput128 for each 16 byte chunk
    update dest_ptr and src_ptr
  if (check whether shmem_iput64 can be used)
    call shmem_iput64 for each remaining 8 byte chunk
    update dest_ptr and src_ptr
  if (check whether shmem_iput32 can be used)
    call shmem_iput32 for each remaining 4 byte chunk
else
  use ALGORITHM 1
```

- Implementing using multiple **shmem_iput128, shmem_iput64,** and/or **shmem_iput32** calls
- Call ***iput*** routines along the x-axis
- Shift between different types
- Suitable for small data sizes
- Does not support all the values for src_stride, dest_stride and blksize



Performance Analysis – Algorithm 1 and 2

	Stride size – Block size (bytes)					
Block Size (bytes)	**4**	**16**	**64**	**256**	**1024**	**4096**
16	iput	iput	iput	iput	iput	iput
64	iput	iput	iput	iput	iput	iput
128	putmem	iput	iput	iput	iput	iput
256	putmem	putmem	iput	iput	iput	iput
512	putmem	putmem	putmem	putmem	putmem	putmem
1024	putmem	putmem	putmem	putmem	putmem	putmem
4096	putmem	putmem	putmem	putmem	putmem	putmem
16384	putmem	putmem	putmem	putmem	putmem	putmem

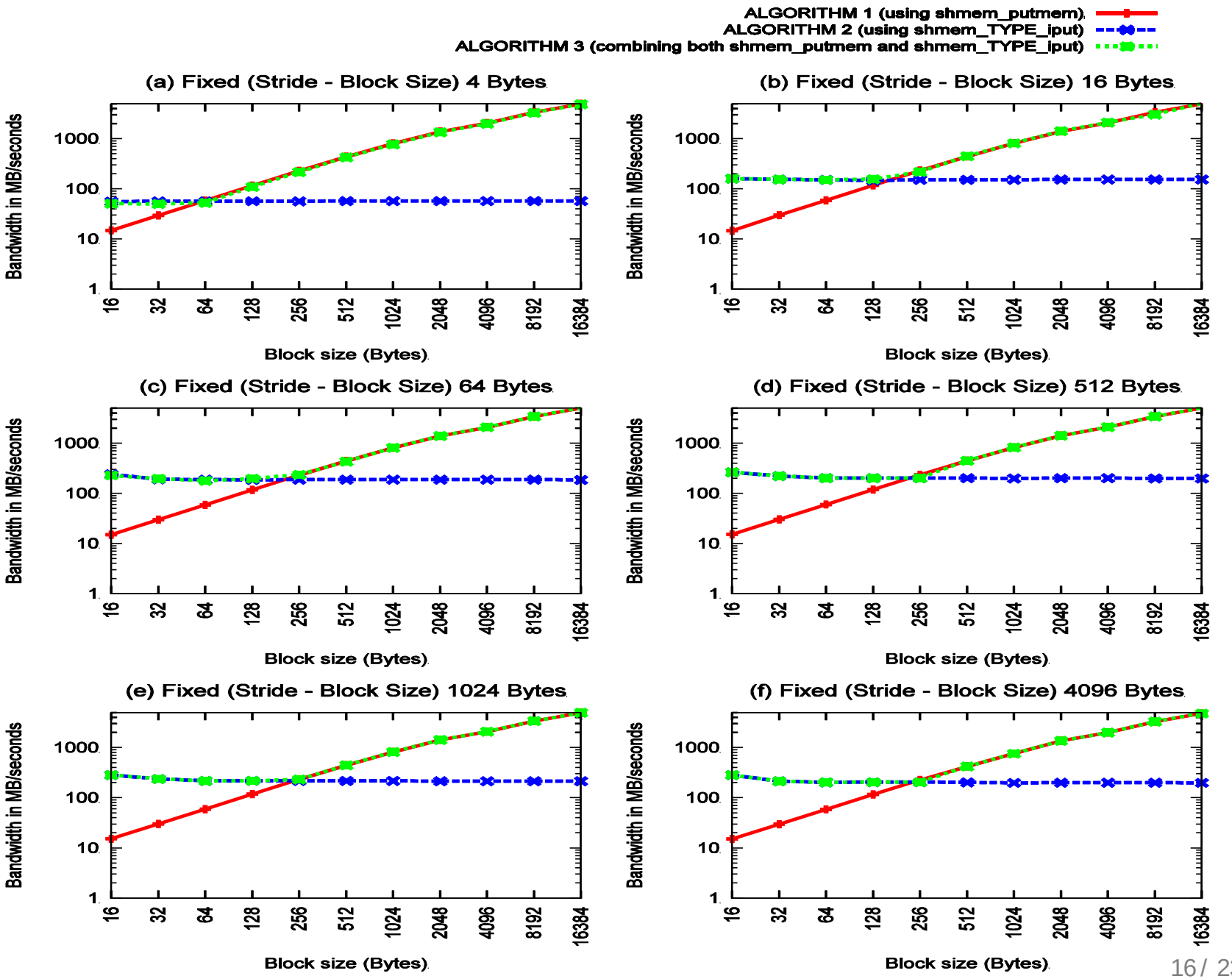
- Implemented **shmem_iputmem** using Algorithm 1 (**shmem_putmem**) and Algorithm 2 (**shmem_iputX)**
- Compared both the implementations for different stride and block size
- Total array size is kept fixed at 64 MiB
- Algorithm 1 performs well for large block size and Algorithm 2 for small block sizes

Tuned Baseline Algorithm - Analysis

- Based on previous analysis, designed a tuned baseline Algorithm using both Algorithm 1 and Algorithm 2
- Tuned Algorithm – will select either Algorithm 1 or Algorithm 2 based on the block size
- For small block sizes will select Algorithm 2 and for large block sizes will select Algorithm 1

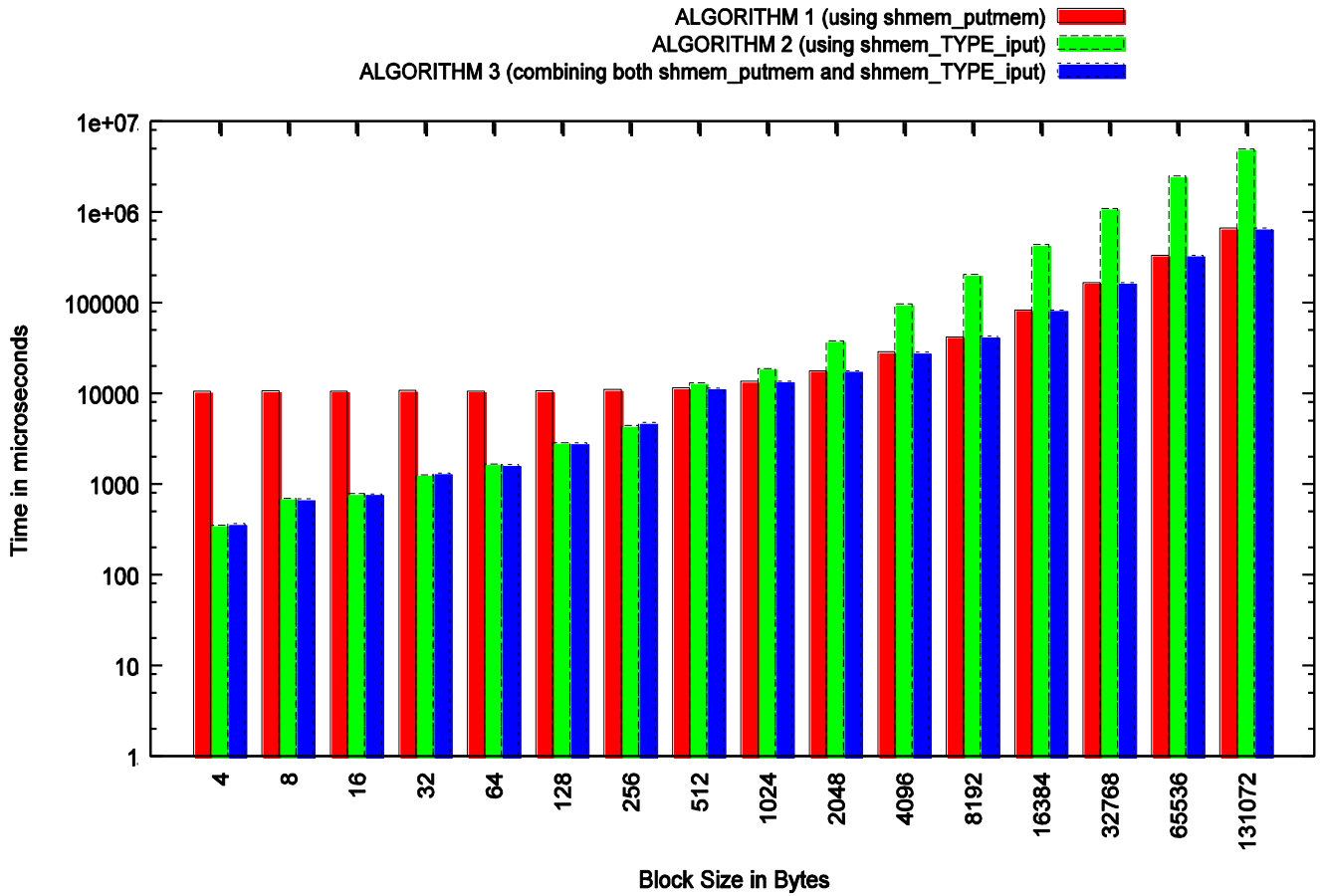
```
bool α = predicate_lookup_table(dest_stride, src_stride, blksize);
if (α == 0) then
    call Algorithm_1(using shmem_putmem)
else
    call Algorithm_2(using shmem_TYPE_iput)
end if
```

Tuned Baseline Algorithm - Implementation



2D Halo Exchange Benchmark

- Data decomposition across single dimension
- Performance of three Algorithms
- Fixed Data Size of 128 MiB
- Cray SHMEM 7.2.2



Related Work

- complementary to **shmem_TYPE_aput** and **shmem_TYPE_aget** * proposed at OUG last year
  - would be identical for TYPE="byte"
  - may be adapted as **shmem_aputmem** and **shmem_agetmem** within that framework

- Other PGAS models – Coarray Fortran (CAF)
- Strided support in CAF – generic and support data transfers in blocks
- Derived Data Types in MPI

*For reference, see:

http://nic.uoregon.edu/pgas14/oug_submissions/oug2014_submission_4.pdf
http://www.csm.ornl.gov/OpenSHMEM2014/documents/talk2_hammond_OUG14.pdf

Conclusion and Future Work

- Block-wise strided data transfer should be supported by OpenSHMEM
- Support for aggregate data types like array of structures
- Performance results on basic benchmarks and the usage of these block-wise strides in applications are discussed; early evaluations shows 64% improvements on using **predicate_lookup_table**
- Exploring use of additional parameters for predicate lookup table to optimize selection logic
- Exploring other options for optimizations, rather than using the existing APIs

Acknowledgements

- TOTAL
- HPC Tools, University of Houston
- Oak Ridge Leadership Computing Facility (OLCF) at the Oak Ridge National Laboratory
- Redmine: **Extension #58**

Extending Strided Communication Interface in OpenSHMEM

Naveen Namashivayam, Dounia Khaldi, Deepak Eachempati, Barbara Chapman

University of Houston

{nravi, dounia, dreachem, chapman}@cs.uh.edu

August 5, 2015