OpenSHMEM as an Effective Communication Layer for PGAS models

Naveen Namashivayam
Adviser: Dr. Barbara Chapman

University of Houston
{nravi, chapman}@cs.uh.edu

October 13, 2015

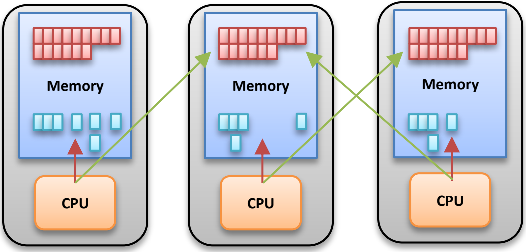
Introduction and Overview

- Almost 60 years later, Fortran still strong in HPC
  - NERSC estimates ½ the hours on their systems are used by Fortran codes
  - If it isn't broke, don't fix it. Economical costs also matters
  - Tested Libraries & Apps like LAPACK, BLAS, NWChem,....
- Evolution of Modern Fortran
- **Coarray Fortran (CAF) –** Parallel programming feature in Fortran 2008
- CAF is a part of **PGAS** Programming Model
- Important problem – Portability (system interface)
  - Not just in CAF, but it is too hard to miss in CAF
  - Possible Solution : Use a Common Communication Substrate
  - Or use **OpenSHMEM** as a transport layer for CAF

What ?	Why ?	How ?	What now?
Introduction to PGAS, CAF and OpenSHMEM	Using OpenSHMEM as transport layer for CAF	Implement CAF over OpenSHMEM	Experimental Analysis and Future Work

Partitioned Global Address Space (PGAS)

- A new class of languages and libraries
- Alternative to MPI for programming large-scale parallel systems
- Attempts to combine both the distributed and shared memory systems
- Memory – Logical shared but physically distributed across the system



- PGAS Languages – Unified Parallel C (UPC), Coarray Fortran (CAF), Chapel and X10, ….
- PGAS Library Interface – OpenSHMEM, Global Arrays (GA), RMA feature set from MPI-3.0, ….
- Apps either use PGAS alone (NWChem on Archer using GA) or use as a hybrid with other models (IFS on Titan use CAF+MPI)

Coarray Fortran (CAF) - Overview

- PGAS Language
- Integral part of Fortran 2008 standards
- As a language depends on compiler

- Properties of Fortran 2008 CAF standards
    - Symmetric memory object management
    - Remote Read and Write operations
    - Barrier and Point-Point synchronization
    - Critical section and locks
    - Atomic memory operations

UHCAF in OpenUH
(GASNet / ARMCI)

OpenCoarrays in GFortran
(MPI-3.0 / GASNet)

"Cray CAF" in CCE
(DMAPP)

"Intel CAF" in Intel Fortran
(MPI)

UHCAF in **OpenUH** by OSU
(UCR)

OpenSHMEM - Overview

- PGAS Library
- Culmination of unification effort among different SHMEM implementations

OpenSHMEM (GASNet)	Cray SHMEM	SGI SHMEM	OSU SHMEM (MVAPICH2-X)
OpenSHMEM (UCCS)	OSHMPI (over MPI-3.0)	Open MPI SHMEM	Portals SHMEM
Quadrics SHMEM	Mellanox Scalable SHMEM	Intel SHMEM (libfabrics)	other implementations

- Optimized implementations are light-weight and place priority on performance rather than portability
- SPMD-like style of programming
- Properties available in recent OpenSHMEM-1.2 specifications
    - Symmetric Data Object management
    - Remote Read and Write using Put and Get operations
    - Barrier and Point-Point synchronization
    - Atomic memory operations and
    - Collective reduction and broadcast operations

CAF and OpenSHMEM Example

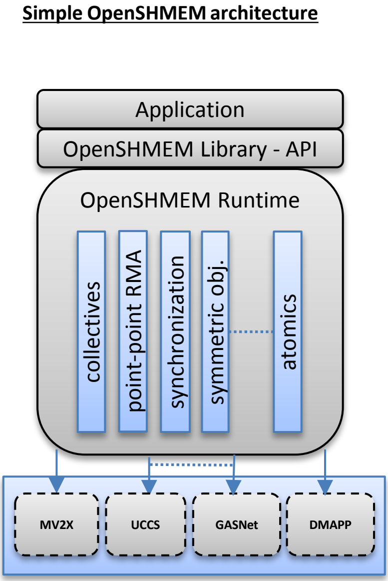
Example CAF program

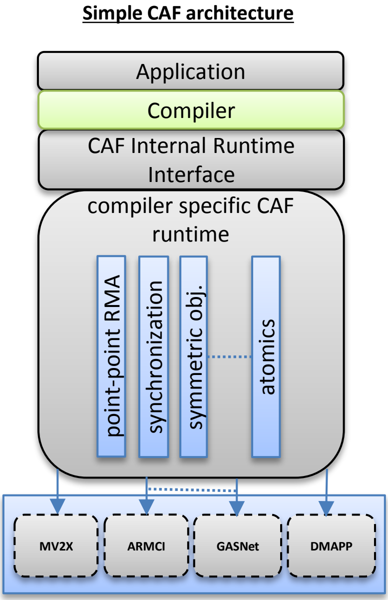
```
integer :: i
integer :: np, me

integer, allocatable :: coarray_x(:)[:]
integer, allocatable :: coarray_y(:)[:]

allocate ( coarray_x (4)[*] )
allocate ( coarray_y (4)[*] )

me   = this_image()
np   = num_images()

do i = 1, 4
      coarray_x (i) = me
      coarray_y (i) = 0
end do

coarray_y(2) = coarray_x(3)[4]

sync all
```

Example OpenSHMEM program

```
int i;
int np, me;

int *coarray_x, *coarray_y;

shmem_init ();
coarray_x = shmem_malloc (4 * sizeof (int));
coarray_y = shmem_malloc (4 * sizeof (int));

me    = shmem_my_pe() ;
np    = shmem_n_pes();

for ( i = 0; i <= 3; i++) {
     coarray_x [ i ] = me;
     coarray_y [ i ] = 0;
}

shmem_int_get(&coarray_y[1], &coarray_x[2], 1, 3);

shmem_barrier_all();
```

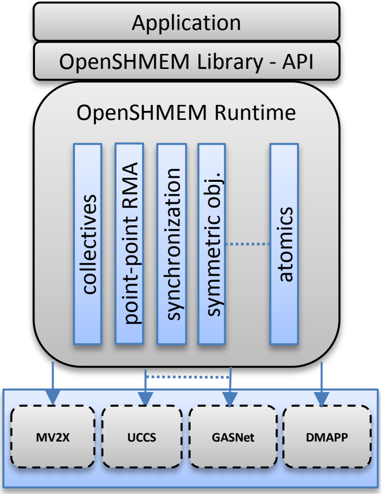
Communication Layer – System Interfaces



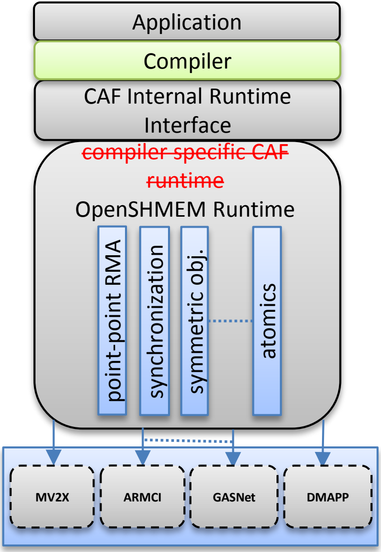


Communication Layer – System Interfaces

Simple OpenSHMEM architecture



Simple CAF architecture



Motivation – Functionality similarities

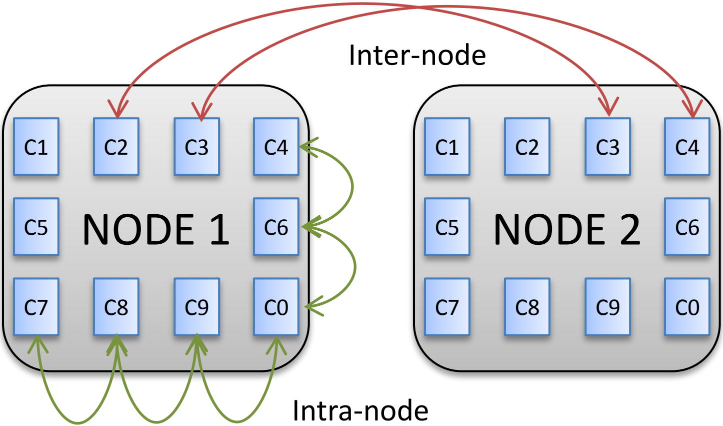
- Functionality and Features – Similar features as in CAF

Properties	CAF	OpenSHMEM
Symmetric data allocation	allocate	shmem_malloc()
Total image count	num_images()	shmem_n_pes()
Current image ID	this_image()	shmem_my_pe() + 1
Collectives – reduction*	co_***operator***	shmem_***operator***_to_all()
Collectives – broadcast*	co_broadcast	shmem_broadcast
Barrier Synchronization	sync_all	shmem_barrier_all()
Remote memory Put/Get	[]	shmem_putmem/getmem
Single dimensional strided put	[]	shmem_***TYPE***_iput/iget
Multi dimensional strided put	[]	X
Remote Locks	*lock*	X
Other Properties	……….	………

* Future revision of Fortran standards – Fortran 2015

Motivation – Performance Analysis

- Titan – 18, 688 Nodes – Current Cray machine in ORNL
- CORAL project - $525 million dollar systems
  - Summit – 3,400 Nodes – 2018 IBM machine for ORNL
  - Aurora – more than 50,000 Nodes – 2018 ANL Machine
- Analysis on Inter- and Intra-node Communication Costs



Inter-node – Test Overview

- Relative comparison with other PGAS libraries and PGAS runtimes
- Compared OpenSHMEM, MPI-3.0 and GASNet in 2 different systems
- PGAS Microbenchmark Test Suite from University of Houston

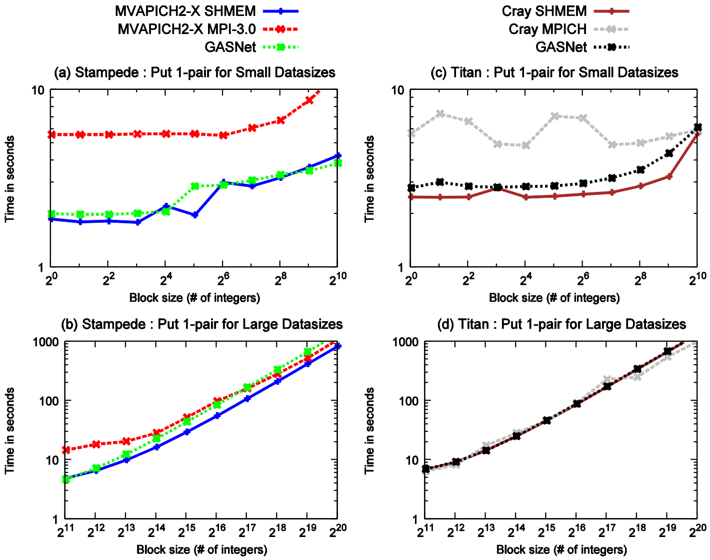
- System – 1: Stampede (TACC) – Infiniband Cluster

Nodes (cores)	OpenSHMEM	MPI-3.0 (RMA)	GASNet (conduit)
6400 (16)	MVAPICH2-X SHMEM	MVAPICH2-X	GASNet 1.24.0 (IB)

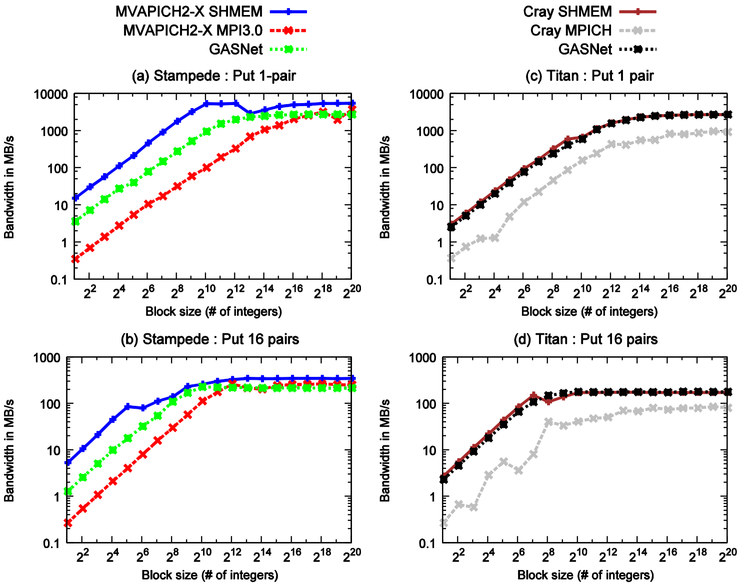
- System – 2: Titan (Cray XK7, ORNL) – Gemini system

Nodes (cores)	OpenSHMEM	MPI-3.0 (RMA)	GASNet (conduit)
18688 (16)	Cray SHMEM in Cray MPT	Cray MPICH in Cray MPT	GASNet 1.24.0 (Gemini)

Inter-node – Comparing Latency

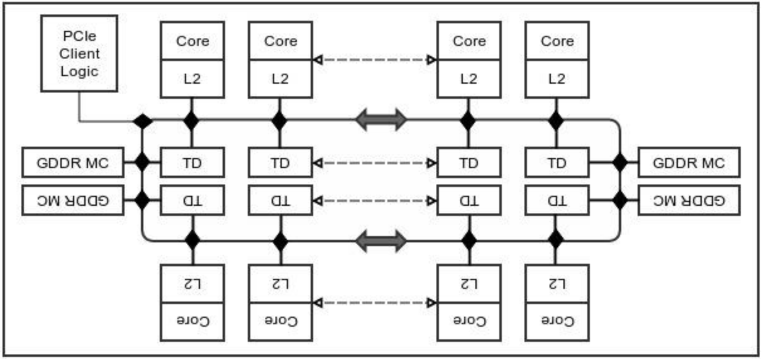


Inter-node – Comparing Bandwidth



Intra-node – Test Overview

- Intel Xeon Phi – KNC : Architectural Overview



- 61 * 4 = 244 cores
- Ring Interconnect
- Bidirectional
- x86 compatibility
- L1(32KB), L2(512KB)
- 8 Memory Controllers
- 16 GDDR5 memory
- PCI Express System

Intra-node – RMA Put-Get to Local Load/Store

- *shmem_ptr:* address of a data object on a specific PE
- No function call → enhancing compiler optimizations
- Enables optimizations such as vectorization

```
shmem_int_put(target, source, i, pe);
```

→

```
int *ptr = (int *)shmem_ptr(target,pe);
for (m = 0; m < i; m+=1)
  ptr[m] = source[m];
```

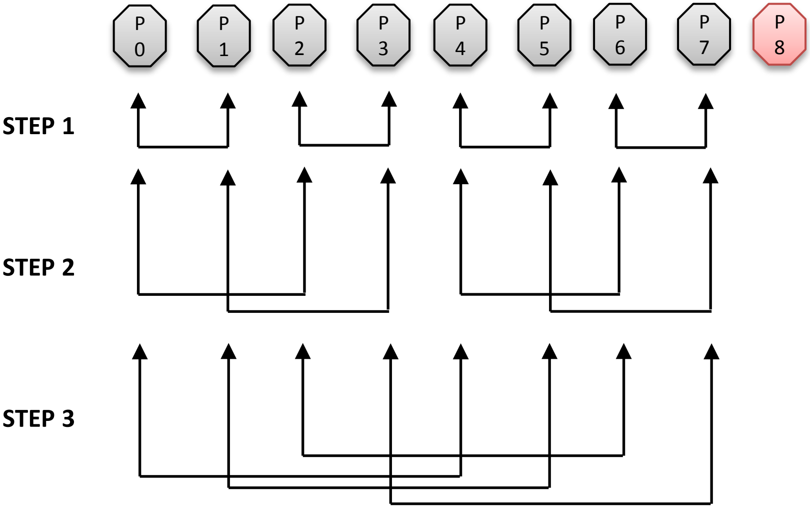
- Using load/store operations to optimize SHMEM collectives: Barrier, Broadcast, Collect and Reduce
- Implementation of different reduction algorithms:
  - Flat Tree (FT)
  - Recursive Doubling (RD)
  - Rabenseifner: Reduce Scatter All Gather (RSAG)*
- Performance comparison with Intel MPI and MVAPICH
  - PGAS-Microbenchmarks from University of Houston^

* Rajeev Thakur, Rolf Rabenseifner, and William Gropp. Optimization of Collective Communication Operations in MPICH, 2005

^ https://github.com/uhhpctools/pgas-microbench

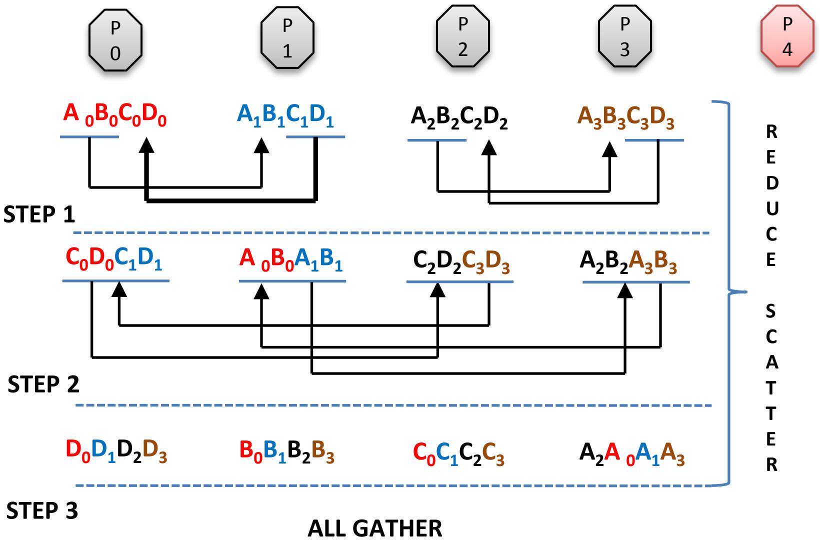
Recursive Doubling (RD)

Array = { 1, 2 .... 100 }



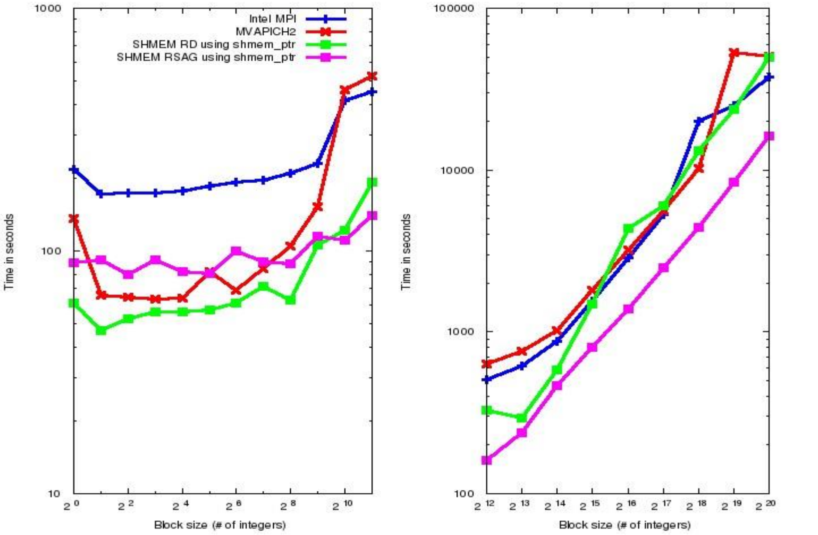
Reduced Scatter All Gather (RSAG)

Array = { 1, 2 .... 100 } → **A = { 1, .. 25 } B = { 26, .. 50 }**
C = { 51, .. 75 } D = { 76, .. 100 }

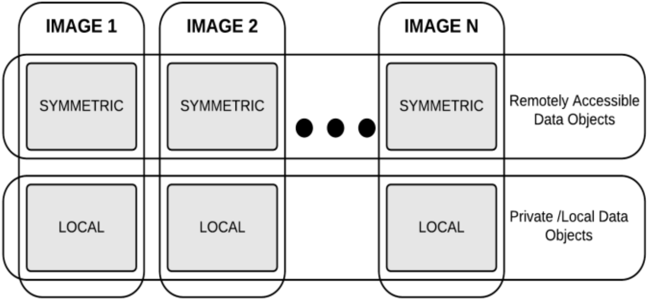


SHMEM Pointer – Performance Comparison

Various algorithms using load/store vs. Intel MPI and MVAPICH2 - 64 PEs



Implementing UHCAF over OpenSHMEM
Symmetric Data Allocation



- How to make the data remotely accessible?
  - CAF – **save** or **allocatable**
  - OpenSHMEM – **static/global** variables or use **shmalloc()**
- Symmetric Heap : **allocate** and **deallocate** -> **shmalloc()** and **shfree()**
  - A single memory segment allocation with **shmalloc()**
  - Or, use **shmalloc()** for each **allocate** statement

Implementing UHCAF over OpenSHMEM

Remote Memory Accesses

- Remote Memory Access (RMA) in OpenSHMEM is done by **shmem_putmem()** and **shmem_getmem()**
- Similar properties in CAF, but not exactly matching
- Same location and from the same image
    - CAF ensures it is ordered
    - OpenSHMEM allows out-of-order with respect to other remote access

1. Local Completion

coarray_y(:)[2] = coarray_x(:)
coarray_x(:) = 0

2. Remote Completion

coarray_a(:)[2] = coarray_b(:)
coarray_c(:) = coarray_a(:)[2]

- For RMA Put : **shmem_putmem()** followed by **shmem_quiet()**
- For RMA Get : **shmem_getmem()** preceded by **shmem_quiet()**

Implementing UHCAF over OpenSHMEM
Strided Data Transfer – Single Dimensional Arrays

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

A[10] =	A [0]	A [1]	A [2]	A [3]	A [4]	A [5]	A [6]	A [7]	A [8]	A [9]	✓

- Block-wise strided data of same basic data type

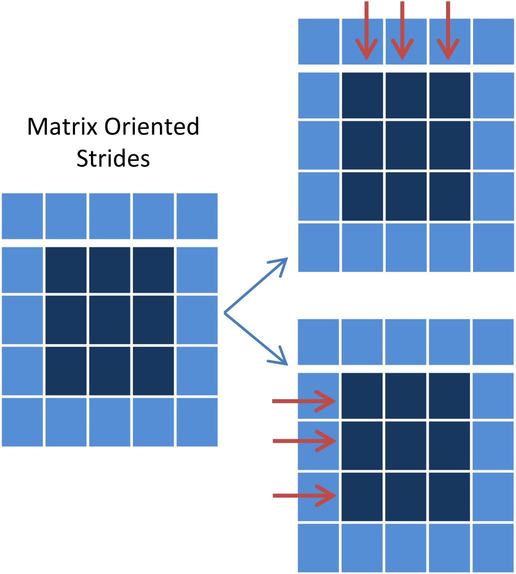
A[10] =	A [0]	A [1]	A [2]	A [3]	A [4]	A [5]	A [6]	A [7]	A [8]	A [9]	✗

Why is strided data transfer important?
Example from my experience – DFT iterations in NWChem, ScaLAPACK

- Highly scalable parallel application
- Run on approximately 25,000 cores – 800 nodes – minimum count
- Most time spent on these kind of Block-wise strided data transfer

Implementing UHCAF over OpenSHMEM
Strided Data Transfer – Multidimensional Arrays

Matrix Oriented Strides



<u>Algorithm 1</u>
Naïve Algorithm
shmem_putmem()
shmem_getmem()
All possible stride size

<u>Algorithm 2</u>
2-DIM Algorithm
shmem_TYPE_iput()
shmem_TYPE_iget()
Not all possibilities

- Size is even
- Size if odd (no 1 byte transfer in SHMEM)

Performance Evaluations

- University of Houston – CAF Test Suite
  - PGAS Microbenchmark Test Suite*
  - Distributed Hash Table Benchmark and^
  - Himeno Benchmark^

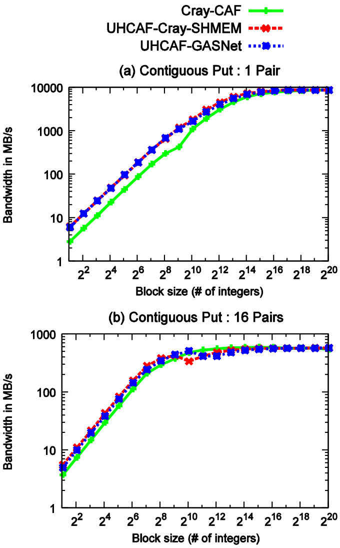
System	Stampede	Cray XC30
Interconnect	InfiniBand	Aries
Native CAF Implementation	-	Cray Compiler (Cray CAF)
UHCAF over GASNet (conduit)	GASNet-1.24.0 (IB)	GASNet-1.24.0 (Aries)
UHCAF over OpenSHMEM	MVAPICH2-X	Cray SHMEM

* https://github.com/uhhpctools/pgas-microbench.git
^ https://github.com/uhhpctools/caf-testsuite.git

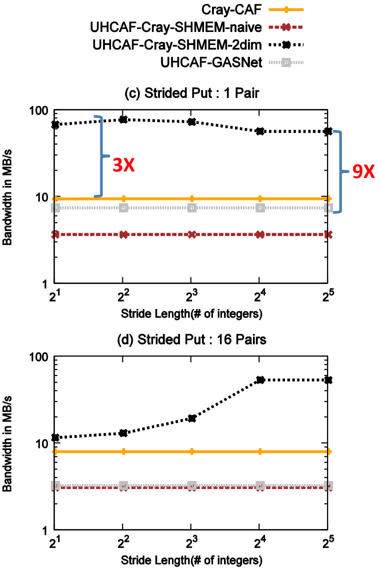
Experimental Analysis – Contiguous and Strided

- PGAS Microbenchmark - Cray XC30 Aries System

Put Bandwidth



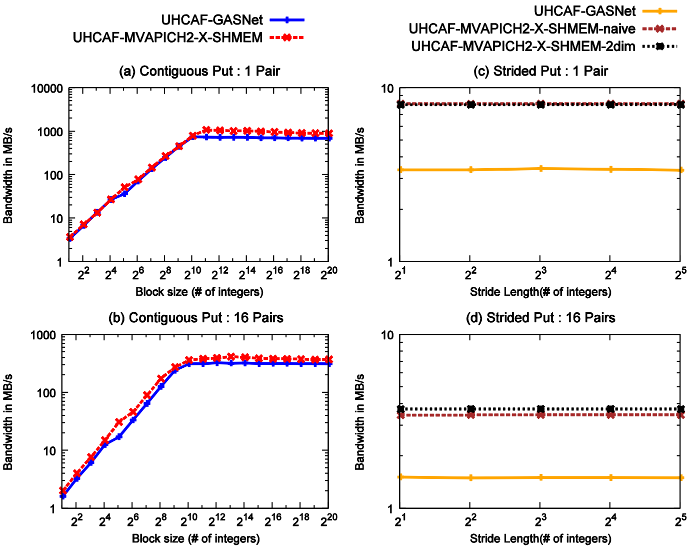
2-DIM Strided Put Bandwidth



Experimental Analysis – Contiguous and Strided

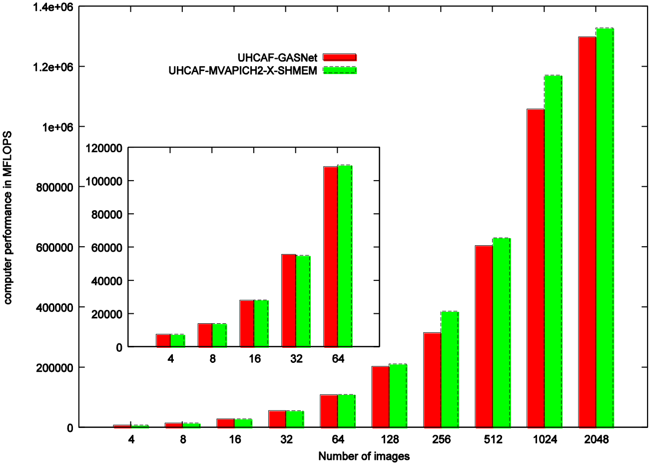
- PGAS Microbenchmark - Stampede Infiniband Cluster

Put Bandwidth **2-DIM Strided Put Bandwidth**



Experimental Analysis – Himeno Benchmark

- 2-Dimensional data decomposition and strided data communication
- Advantage in using Naïve strided algorithm (**shmem_putmem**)
- Average 6% improvements with UHCAF(MV2-X SHMEM) and maximum 22% improvements



Research Contributions

- PGAS Conference – 2014 (**Best Paper**)

Native Mode-Based Optimizations of Remote Memory Accesses in OpenSHMEM for Intel Xeon Phi

Naveen Namashivayam, Sayan Ghosh, Dounia Khaldi, Deepak Eachempati and Barbara Chapman

- IEEE Cluster Conference – 2015

OpenSHMEM as a Portable Communication Layer for PGAS Models: A Case Study with Coarray Fortran

Naveen Namashivayam, Deepak Eachempati, Dounia Khaldi and Barbara Chapman

- OpenSHMEM Workshop – 2015

Extending the Strided Communication Interface in OpenSHMEM

Naveen Namashivayam, Dounia Khaldi, Deepak Eachempati and Barbara Chapman

- OpenSHMEM Workshop – 2015

Proposing OpenSHMEM Extensions Towards a Future for Hybrid Programming and Heterogeneous Computing

David Knaak and **Naveen Namashivayam**

Conclusion and Future Work

- Implemented UHCAF over OpenSHMEM and evaluated performance
    - At times, in a Cray System UHCAF over Cray SHMEM is better than Cray CAF itself (based on Nov' release)
- Should all CAF implementations use OpenSHMEM as transport layer ?
    - On CORAL timeline, it is a better option compared to UCX or libfabrics
    - Availability in many systems, system specific optimizations
    - OpenSHMEM implementations are generally not portable, but Apps that use OpenSHMEM are highly portable
- Using OpenSHMEM for other PGAS Languages and Libraries
    - Global Arrays, UPC
- OpenSHMEM specifications committee – accepting new extensions
    - Unavailable features like Active Messages and Atomic Memory Operations can be added
- Future Work
    - Perform tests at large scale
    - Optimize Symmetric memory heap maintenance
    - Optimize using native OpenSHMEM features like **shmem_ptr**

Acknowledgements

Adviser : Prof. Barbara Chapman
Thesis Committee Members: Prof. Edgar Gabriel, Mikhail Sekachev
Project Mentor : Deepak Eachempati, Dounia Khaldi, Tony Curtis
HPC Tools : Sunita Chandrasekaran, Siddhartha Jana, Sayan Ghosh, Shiyao Ge, Pengfei Hao and HPC Tools
Cray : Mark, David, Bill, Bob, Krishna, Dan, Steve, Kim, Nick, Joe and all my PE-MPT team members
TOTAL : Henri Calandra, Maxime Hugues

TOTAL for allowing us to use their Cray XC30 system, and their interest in OpenUH CAF project

Oak Ridge Leadership Computing Facility for the CrayXK7 – Titan.

Texas Advanced Computing Center for their Infiniband Cluster Stampede.

OpenSHMEM as an Effective Communication Layer for PGAS models

Naveen Namashivayam
Adviser: Dr. Barbara Chapman

University of Houston
{nravi, chapman}@cs.uh.edu

October 13, 2015

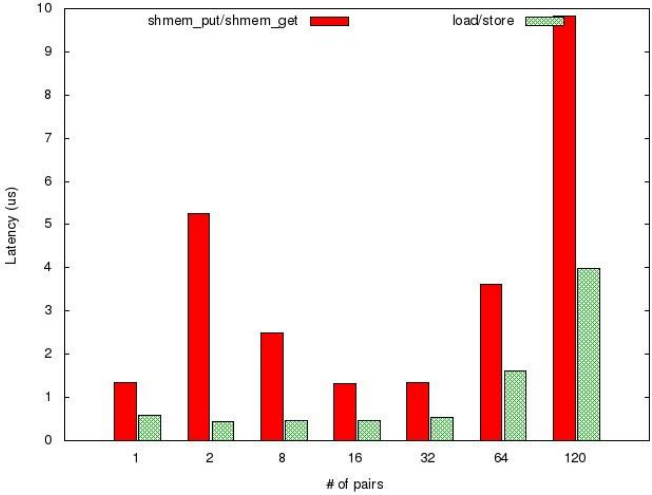
UNIVERSITY of HOUSTON | COMPUTER SCIENCE

Masters Thesis Defense

Backup Slides

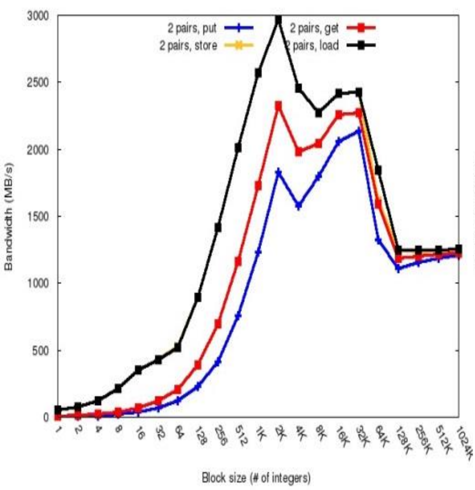
put/get → load/store - Latency

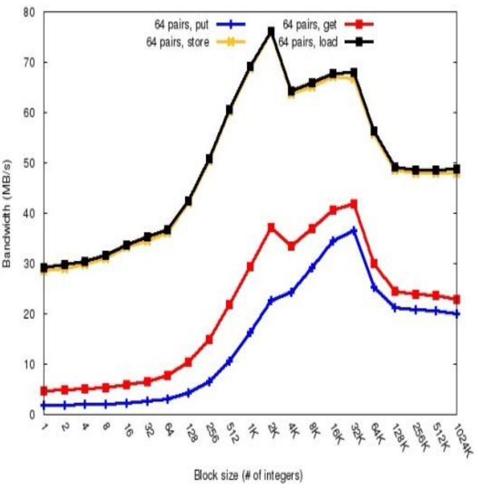
- Latency comparison
- ***shmem_put/shmem_get*** vs. load/store
- PGAS-Microbenchmarks from University of Houston



put/get → load/store - Bandwidth

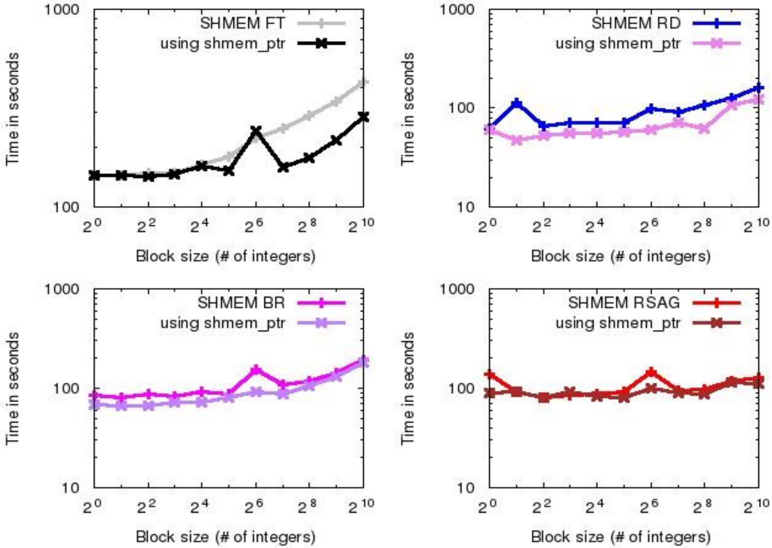
- Bandwidth comparison
- *Blocksize* is represented in number of integers
- Results for 2 pairs and 64 pairs of PEs





put/get → load/store - Latency

- ***shmem_put/get*** vs. load/store – small data sizes - 64 PEs



NAS Parallel Benchmarks

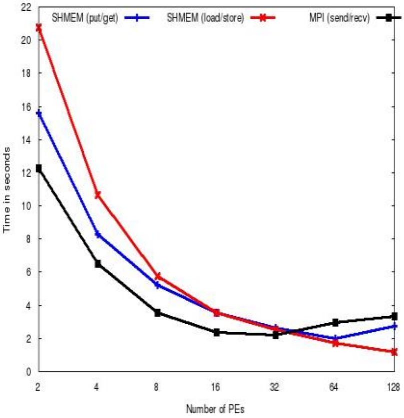
- MPI, CLASS C
- NAS SHMEM benchmarks*

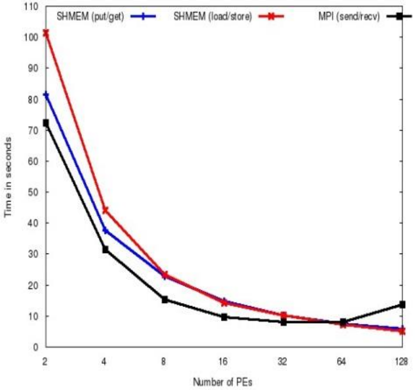
Benchmark	Reduction (%)	Remote Access (%)	SHMEM* Ver. Available
MG	0.1	19.6	YES
BT	0	15.5	YES
EP	1.6	0	YES
SP	**0**	**44.1**	**YES**
IS	**12.4**	**11.7**	**YES**
CG	0	33.2	NO
FT	0.8	31.2	NO
DT	0	10	NO
LU	0.1	14.8	NO

* https://github.com/openshmem-org/openshmem-npbs

IS NAS Benchmark

- Improved reduction algorithms: up to 22% compared to MVAPICH and 60% compared to IMPI
- Improved communications: decrease in latency by up to 60% and increase in bandwidth by up to 12x
- Use RD for small message sizes and RSAG for large message sizes.





Implementing Strided Data Transfer

Algorithm 1

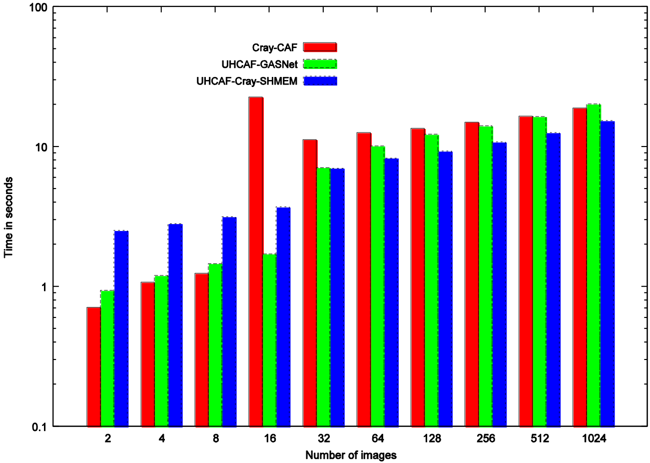
```
for ( i = 1, i ≤ nelems, i++) do
  shmem putmem(dest_ptr, src_ptr, blksize, pe_id)
  dest_ptr += dest_stride
  src_ptr  += src_stride
end for
```

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
```

Experimental Analysis – Distributed Hash Table

- Titan Supercomputer, Distributed Hash Table Benchmark
- Latency measurement
- UHCAF(Cray SHMEM) 28% faster than Cray CAF and 18% faster than UHCAF(GASNet)



Experimental Analysis – Locks

- PGAS Microbenchmark Test Suite - Titan (OLCF, ORNL)
- Latency measurement for Locking and Unlocking Operation
- UHCAF(Cray SHMEM) is 22% faster than Cray CAF and 10% faster than UHCAF(GASNet)