Using OpenSHMEM with MVAPICH2-X

Antonio Gómez-Iglesias

Texas Advanced Computing Center

August 21, 2015

TACC

TACC

Out-of-core Methods

- Applications with large memory requirements $\rightarrow$ memory in normal nodes is not enough
- Offload data onto files
  - I/O is slow
  - Need for efficiently storing/retrieving data from disk
- Popular method in many applications

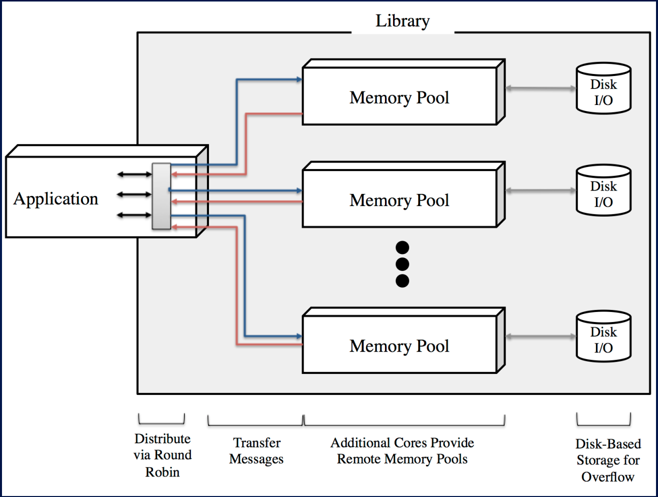
Problems of Out-of-core Methods

- I/O becomes a bottleneck at large scale
- Model not well suited for distributed file systems
  - Very high load on Lustre
  - Possible crashes on the file system

A Distributed Out-of-core Method

- Large clusters, with many nodes
- Each node has memory, even better than local disk
- Each node has a local disk $\rightarrow$ use it
- Offload data to nodes, not files

GRVY Out-of-core Model


Library
Application
Memory Pool
Memory Pool
Memory Pool
Disk I/O
Disk I/O
Disk I/O
Distribute via Round Robin
Transfer Messages
Additional Cores Provide Remote Memory Pools
Disk-Based Storage for Overflow

PGAS Models

- PGAS: Partitioned Global Address Space

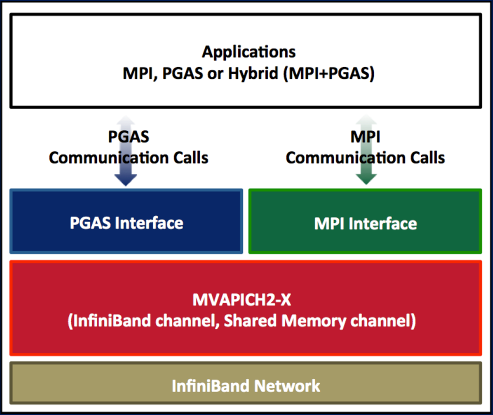
Different Approaches

Libraries	Languages
OpenSHMEM	UPC (Unified Parallel C)
Global Arrays	Co-Array Fortran
Chapel	X10

Why PGAS Models?

- Flexible and light weight way for one-sided communication
- Good fit for irregular communication patterns
- Combination with MPI
- Hybrid models are described as the practical way of programming exascale systems
- MVAPICH2-X allows applications to leverage the best of MPI and PGAS models

MVAPICH2-X


Applications
MPI, PGAS or Hybrid (MPI+PGAS)
PGAS
Communication Calls
MPI
Communication Calls
PGAS Interface
MPI Interface
MVAPICH2-X
(InfiniBand channel, Shared Memory channel)
InfiniBand Network

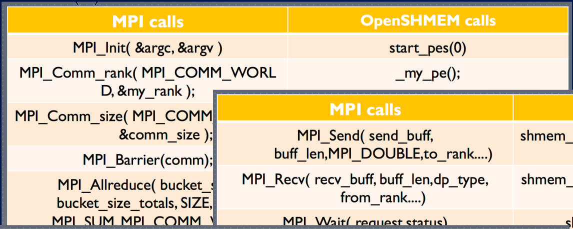
MVAPICH2-X Stampede

- Version 2.0 installed
- Available with
  - gcc/4.7.1
  - intel/14.0.1.106
- Two launchers:
  - tacc-upcrun
  - tacc-oshrun

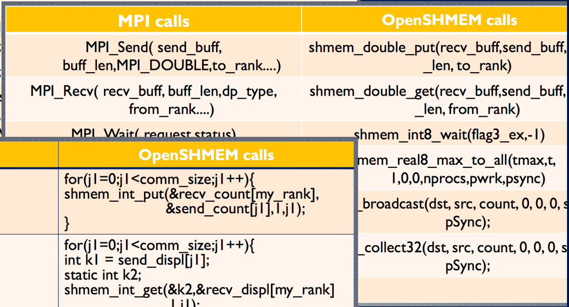
GRVY. MPI Implementation

- C++
- Master-slave model
- Point-to-point communication
- Collectives only at the beginning and end

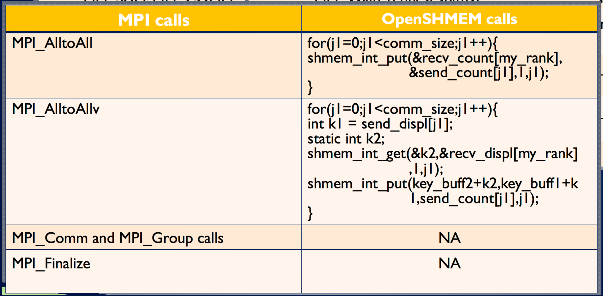
From MPI to OpenSHMEM



MPI calls	OpenSHMEM calls
MPI_Init(&argc, &argv)	start_pes(0)
MPI_Comm_rank(MPI_COMM_WORL D, &my_rank);	_my_pe();
MPI_Comm_size(MPI_COMM &comm_size);	
MPI_Barrier(comm);	
MPI_Allreduce(bucket_s bucket_size_totals, SIZE, MPI_SUM, MPI_COMM_W	



MPI calls	OpenSHMEM calls
MPI_Send(send_buff, buff_len,MPI_DOUBLE,to_rank....)	shmem_double_put(recv_buff,send_buff, _len, to_rank)
MPI_Recv(recv_buff, buff_len,dp_type, from_rank....)	shmem_double_get(recv_buff,send_buff, _len, from_rank)
MPI_Wait(request,status)	shmem_int8_wait(flag3_ex,-1)
	mem_real8_max_to_all(tmax,t, 1,0,0,nprocs,pwrk,psync)
	_broadcast(dst, src, count, 0, 0, 0, s pSync);
	_collect32(dst, src, count, 0, 0, 0, s pSync);



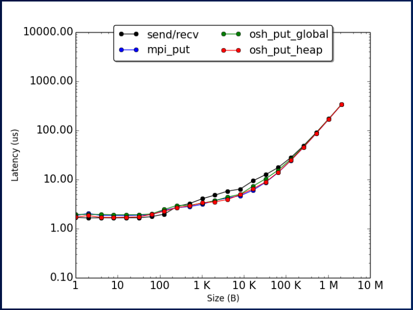
MPI calls	OpenSHMEM calls
MPI_AlltoAll	for(j1=0;j1<comm_size;j1++){ shmem_int_put(&recv_count[my_rank], &send_count[j1],1,j1); }
MPI_AlltoAllv	for(j1=0;j1<comm_size;j1++){ int k1 = send_displ[j1]; static int k2; shmem_int_get(&k2,&recv_displ[my_rank] ,1,j1); shmem_int_put(key_buff2+k2,key_buff1+k1,send_count[j1],j1); }
MPI_Comm and MPI_Group calls	NA
MPI_Finalize	NA

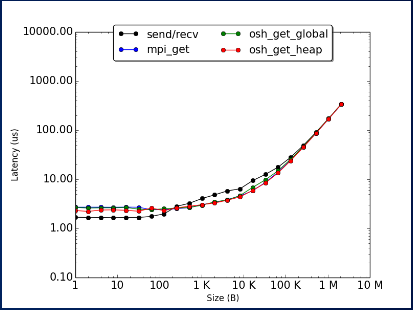
OpenSHMEM Tutorial. PGAS 2012.

Benchmark

- We developed a benchmark that:
  - Does not perform calculations
  - Only measures communication
  - Two parts:
    - Generate data in a master process and store it using the library (store)
    - After all the data has been stored, retrieve it (retrieve)

Latency





Put

Get

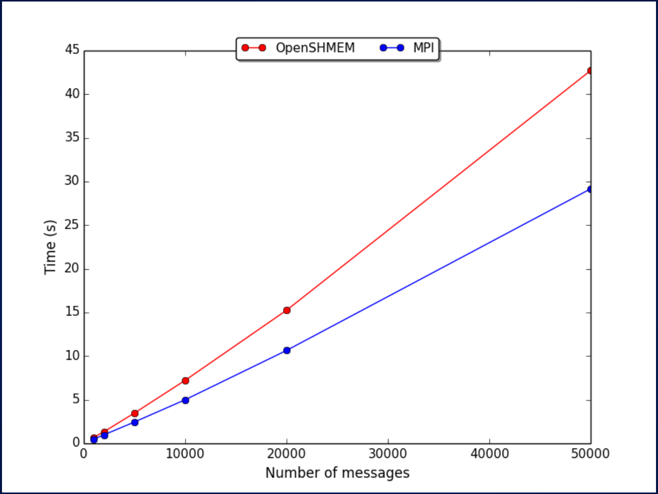
OpenSHMEM Synchronization

- Use a *header*:
  - Fixed structure that contains the operation, the origin and the destination
- Data: shared structure that contains the actual data

Locks

- Easy implementation with *shmem_set_lock* & *shmem_clear_lock*
- Set a lock when writing shared data, clear it once the data has been written
- Only *shmem_put* used

Locks. Results

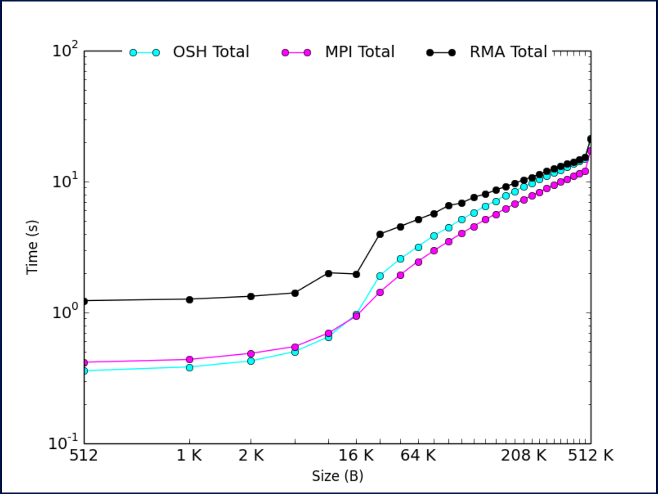


OpenSHMEM Synchronization

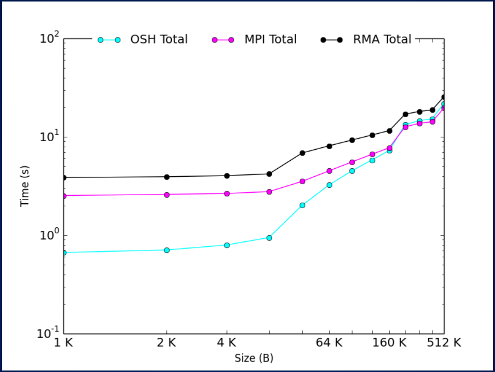
Active Polling

- Instead of using locks, the processes synchronize using *shmem_wait*
- Faster implementation than locks
- Larger change in the code

Active Polling. 128 processes

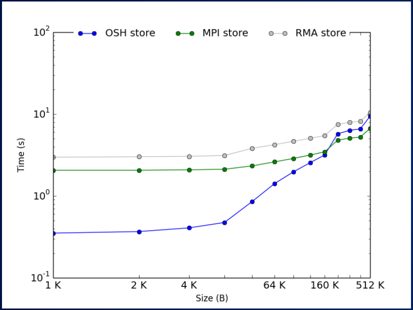


Active Polling. 2048 processes

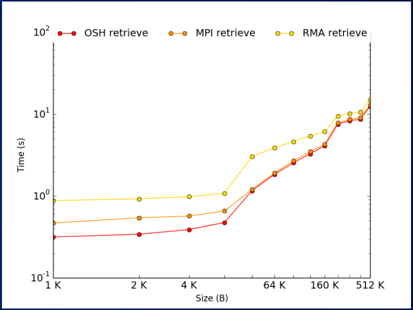


Total

Active Polling. 2048 processes

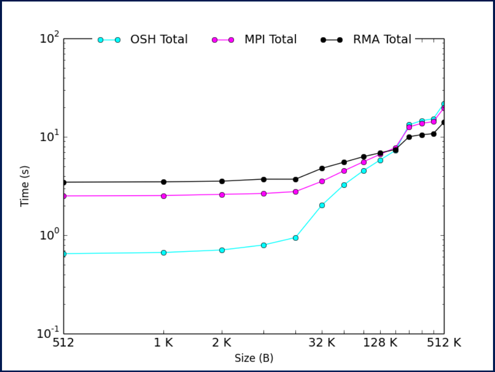


Store



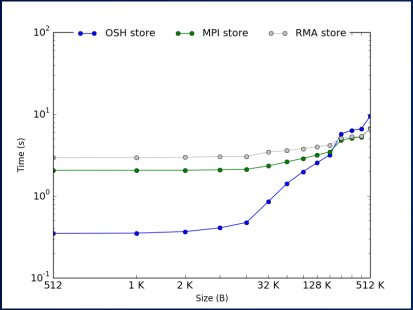
Retrieve

Active Polling. 4096 processes

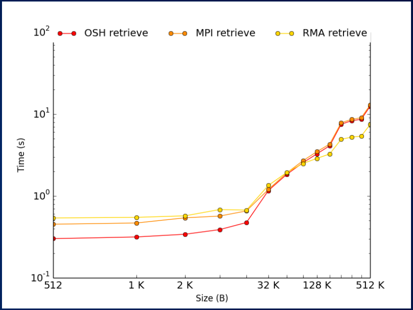


Total

Active Polling. 4096 processes



Store



Retrieve

- OpenSHMEM is a good option for implementing an out-of-core library
- *Easy* porting
- Important to choose the best synchronization model
- MVAPICH2-X makes very easy to use PGAS