POSH: Paris OpenSHMEM

Camille Coti
coti@lipn.fr

OpenSHMEM

OpenSHMEM standard [1]:

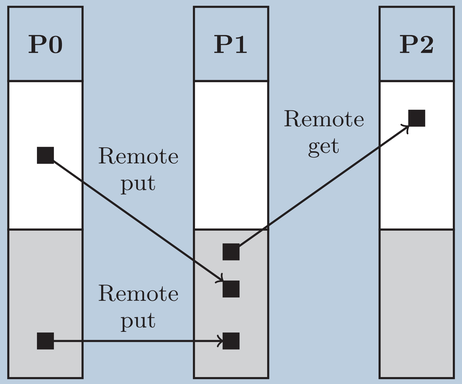
- One-sided communications
- RDMA-fashion

Shared, public memory areas are *symmetric*

One-sided communication

Remote memory access, one-sided communications [3]

- put, get operations
  - Write data in another process's symmetric heap
  - Read data from another process's symmetric heap
- collective communications



Communication engine

Segment of *shared memory*

- Based on `Boost.Interprocess`
- Using the POSIX `shm` API

Optimized memcpy

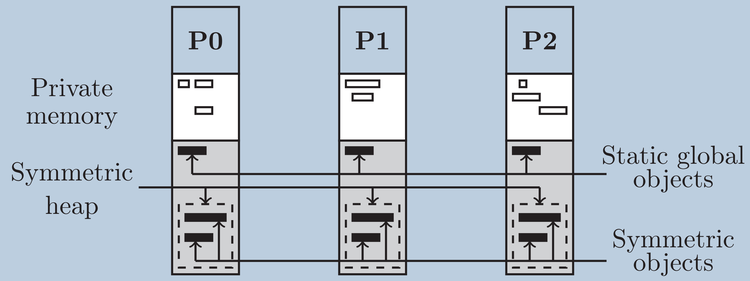
- Peer-to-peer communication = copy from/in other processes' symmetric heap
- Optimized `memcpy` → better communication performance
- SSE, MMX implementations are available

Memory Model

Each process has a *symmetric heap*

- Memory allocations must be performed in a symmetric way
- Data is placed everywhere at the same relative address (see eq. 1)

Static global variables are also located in the symmetric heap.



Template implementation

Each communication operation is declined for each datatype:

- `shmem_char_put`, `shmem_short_put`, `shmem_int_put`, `shmem_long_put`...
- Same action but on a different datatype

→ Can be generated by the compiler!

```
template<class T> void shmem_template_put( T*, const T*, size_t, int );


void shmem_char_put( char *target, const char *source, size_t nelems, int pe
){
  shmem_template_put( target, source, nelems, pe );
}
void shmem_short_put( short *target, const short *source, size_t nelems, int pe ){
  shmem_template_put( target, source, nelems, pe );
}
```

Same as if the programmer had written different functions, easier to write and maintain.

Properties

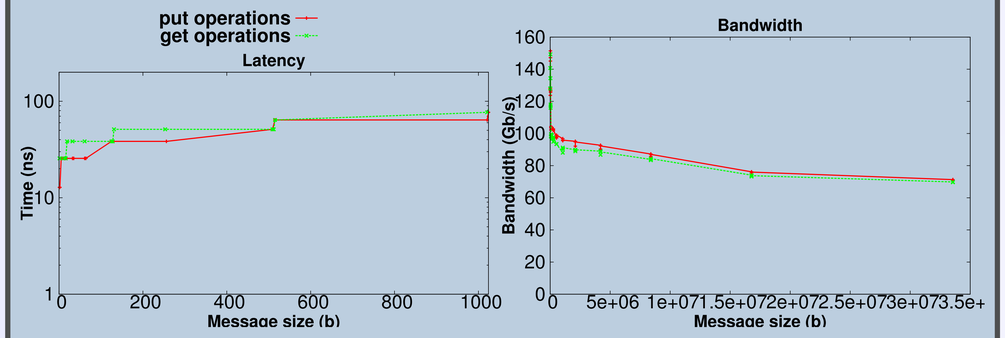
Fact 1 *If all the processing elements are running on the same architecture, the offset between the beginning of a symmetric heap and a symmetric object which is contained by this heap is the same on each processing element.*

Address of a remote variable:

$$addr_{remote} = heap_{remote} + (addr_{local} - heap_{local}) \quad (1)$$

Lemma 1 *Non-symmetric, temporary memory allocations in the heap of a subset of the processing elements that are performed during collective operations do not break the symmetry of the heaps outside of the concerned collective operation.*

Performance: latency and throughput



Performance

Point-to-point communication performance

	SHMEM latency (ns)					SHMEM bandwidth (Gb/s)			
	Best copy		`memcpy`			Best copy		`memcpy`	
	get	put	get	put		get	put	get	put
Caire	38.40	38.40	38.40	38.40	Caire	18.36	18.38	18.36	18.38
Jaune	1741.85	1665.90	1667.90	1663.90	Jaune	17.62	17.55	10.52	10.59
Magi10	38.40	38.40	38.40	38.40	Magi10	20.46	20.16	20.46	20.16
Maximum	38.40	38.40	38.40	38.40	Maximum	74.09	76.15	68.51	69.28
Pastel	1830.40	1689.60	1830.40	1689.60	Pastel	26.07	25.50	26.07	25.50

Comparison with Berkeley UPC :

	UPC latency (ns)			UPC bandwidth (Gb/s)	
	get	put		get	put
Caire	39.40	37.55	Caire	18.03	18.45
Jaune	1623.90	1623.90	Jaune	9.95	10.63
Magi10	73.80	54.90	Magi10	18.64	16.33
Maximum	26.75	25.00	Maximum	67.45	68.86
Pastel	2025.10	1689.95	Pastel	23.52	25.06

Get POSH

How to get POSH:

- Web page: `http://lipn.univ-paris13.fr/~coti/POSH/`
- On GitHub: `https://github.com/coti/POSH`

References

[1] High Performance Computing Tools group at the University of Houston and Oak Ridge National Laboratory Extreme Scale Systems Center: *OpenSHMEM application programming interface, version 1.0 final*. `http://www.openshmem.org`, January 2012.

[2] C. Coti: *POSH: Paris OpenSHMEM: A High-Performance OpenSHMEM Implementation for Shared Memory Systems*, in CoRR abs/1403.7791 [cs.DC], March 2014.

[3] F. Butelle, C. Coti: *A Model for Coherent Distributed Memory For Race Condition Detection*, in Proceedings of the 13th Workshop on Advances in Parallel and Distributed Computational Models (APDCM'11), Anchorage, Ak, May 2011.