OSPRI: An Optimized One-Sided Communication Runtime for Leadership-Class Machines

Jeff Hammond

Argonne Leadership Computing Facility

11 October 2012

Overview

- Motivating application: NWChem, which uses Global Arrays
- Target Hardware: Blue Gene/P and Cray Gemini
- Intellectual driver: seeking fixed-point in one-sided
- Adapt for new applications (FMM) and new hardware (BG/Q)

OSPRI (One-Sided PRImitives) attempts to build on 20+ years of community understanding of one-sided in SHMEM, ARMCI, MPI-2, etc.

This talk is about implementation details and performance, not API syntax and semantics.

PGAS in quantum chemistry

The key reason for the initial and sustained use of Global Arrays (GA) by NWChem is programmer productivity, such as:

- hides complexity of distributed data (lots of *n*-d arrays)
- convenience math routines
- simple dynamic load-balancing
- solves local memory limitations w/o disk

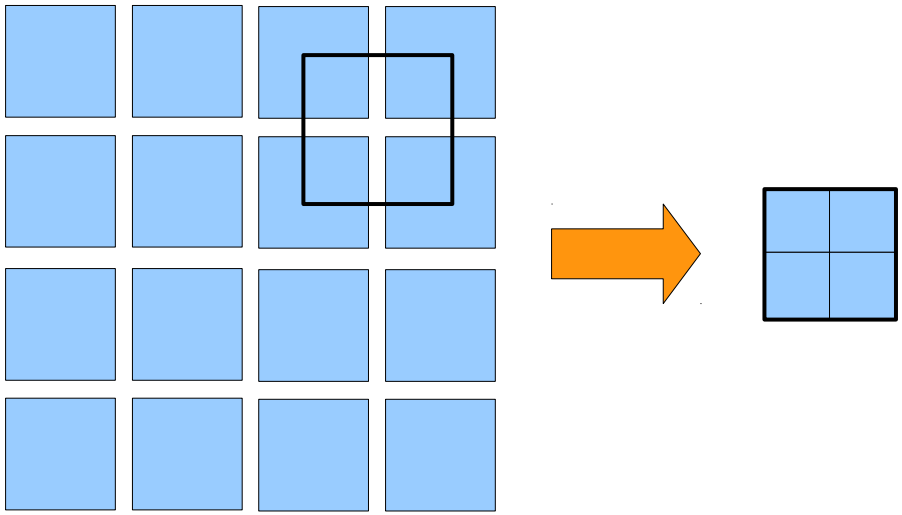
ARMCI emerged later as the communication runtime component within Global Arrays.

The NWChem project started before MPI was available.

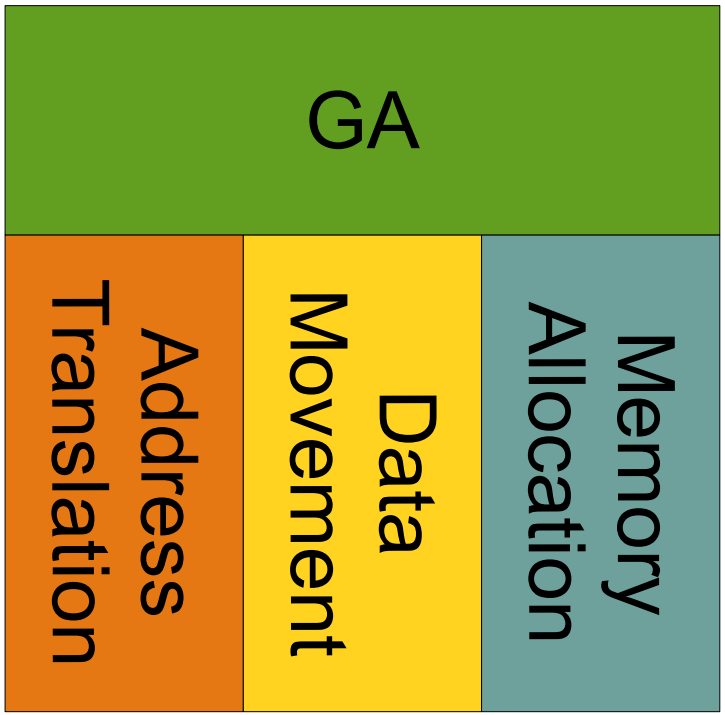
Global Arrays behavior

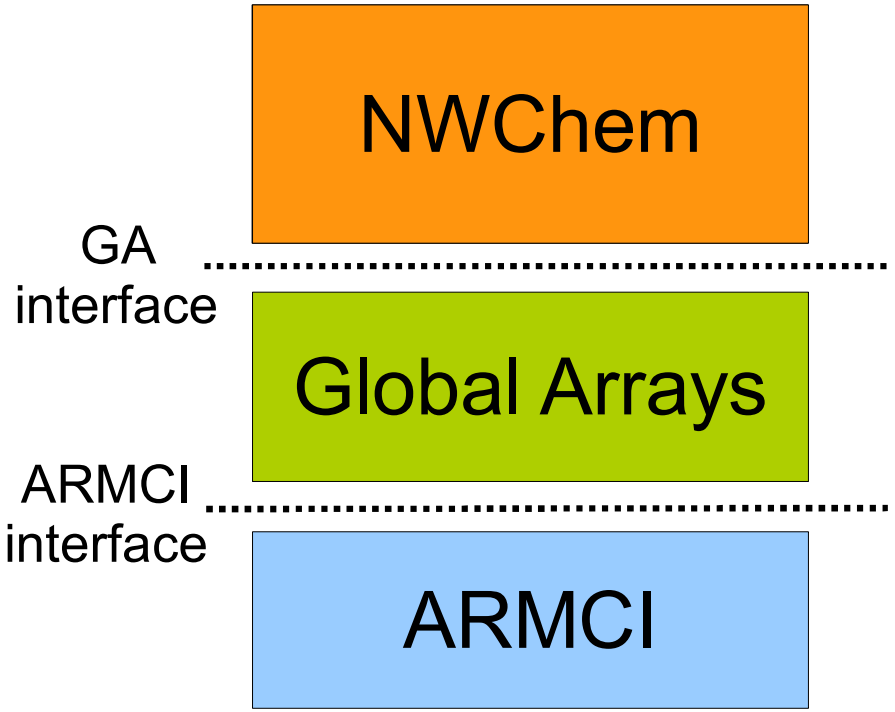
`GA_Get` arguments: handle, *global* indices, pointer to target buffer

1. translate global indices to rank plus local indices
2. issue remote GetS operations to each rank
3. data arrives at initiator from each target rank
4. local buffer assembled



Global Arrays components





MPI and parallel math libraries (e.g. ScaLAPACK) are largely orthogonal. All math routines are collective.

Key ARMCI functionality

One-sided communication:

`ARMCI_Put`, `ARMCI_Get`, `ARMCI_Acc(umulate)`

`ARMCI_PutS`, `ARMCI_GetS`, `ARMCI_AccS`

Remote atomics:

`ARMCI_Rmw` — scalar integer fetch-and-add and swap only

Synchronization:

`ARMCI_Fence` (1-to-1), `ARMCI_AllFence` (1-to-all)

Memory management:

`ARMCI_Malloc` (collective), `ARMCI_Free`,
`ARMCI_Malloc_local`, `ARMCI_Free_local` (registration)

Hardware Properties I

Leadership-class is a DOE term for "top 10"-type systems, which tend to be tightly integrated and custom, not COTS.

- 10-100K nodes, 200K-2M cores and growing
- stripped-down OS (e.g. Catamount, BG CNK)
- processor-network balance
- connectionless, reliable (at least at SW)
- NIC close to chip, powerful DMA

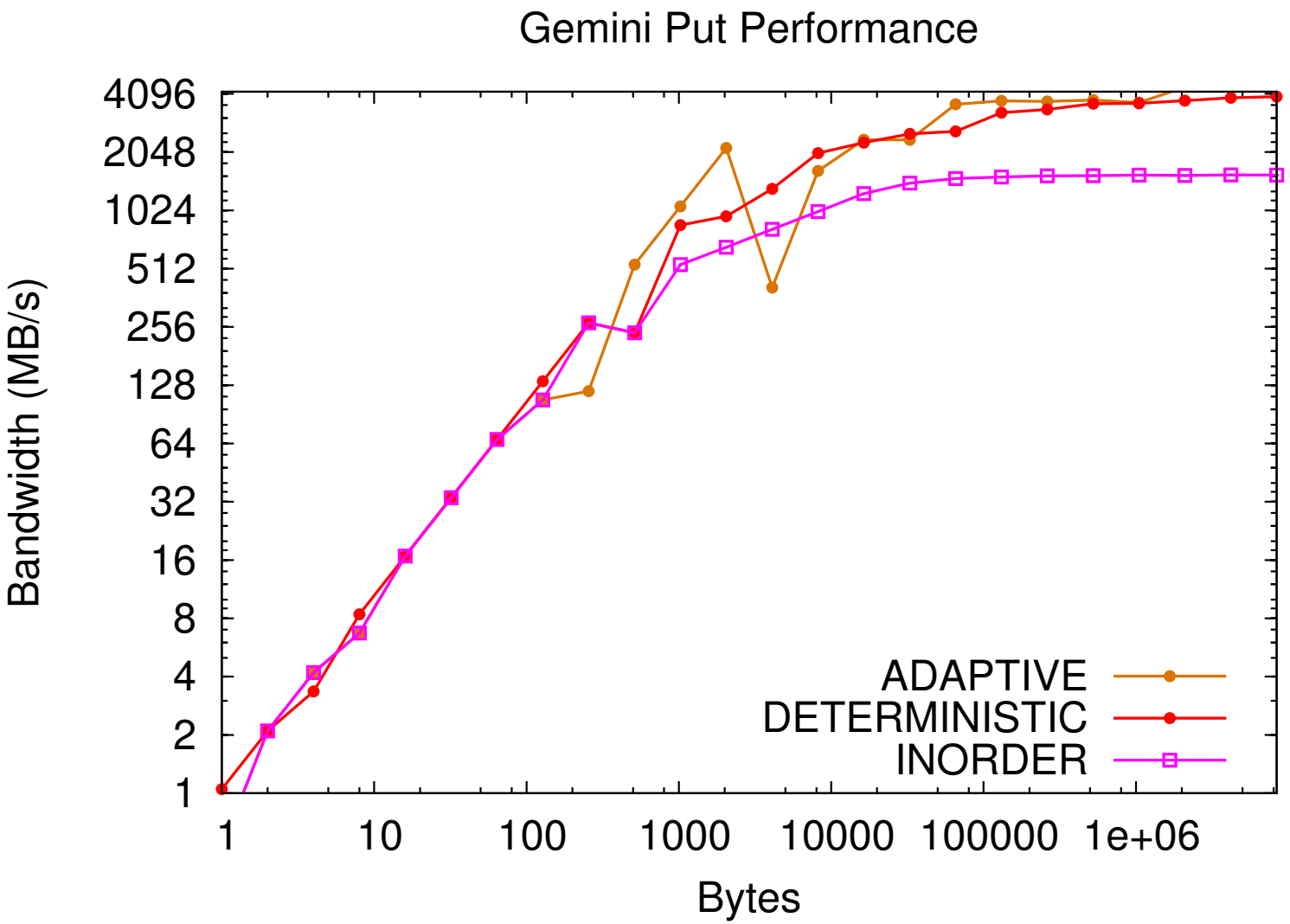
Our goal is to use hardware as much as possible and to make optimizations in software optional and tunable.

Hardware Properties II

Cray Gemini, Blue Gene/P, Blue Gene/Q and PERCS drove thinking about OSPRI design.

- network parallelism:
  e.g. BG/P and BG/Q can hit all links at once, BG/Q multi-context support.
- dynamic routing:
  e.g. PERCS and Gemini ordering is expensive
- slow CPUs:
  e.g. power-efficient BG cores are often the bottleneck
- buffer registration:
  e.g. trivial on BG/P, per-context on BG/Q, expensive on Gemini (and IB...)

Cray Gemini Put Bandwidth



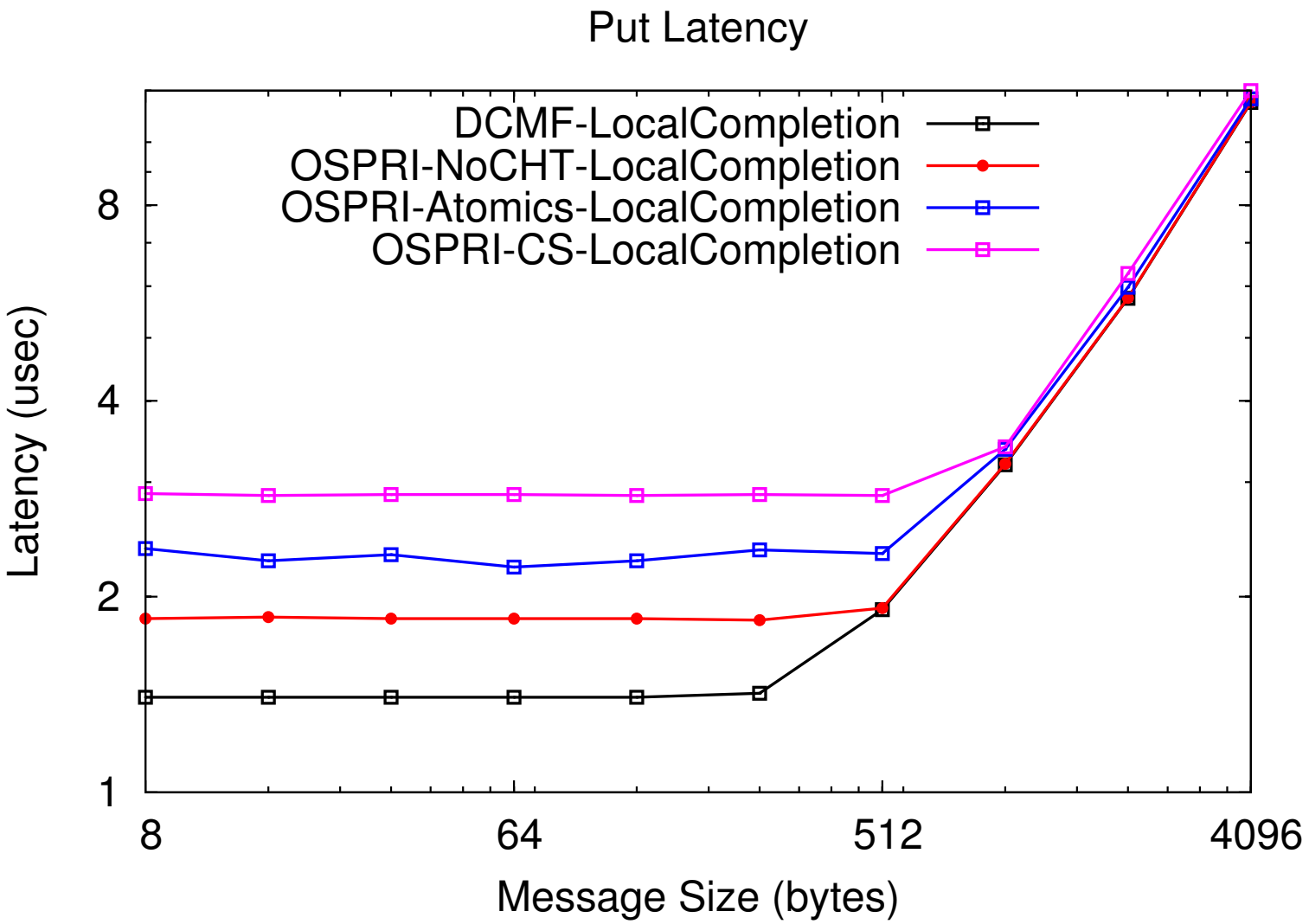
Blue Gene/P details

There was no documentation on DCMF performance behavior so we had to ask IBM and then measure (trust, but verify).

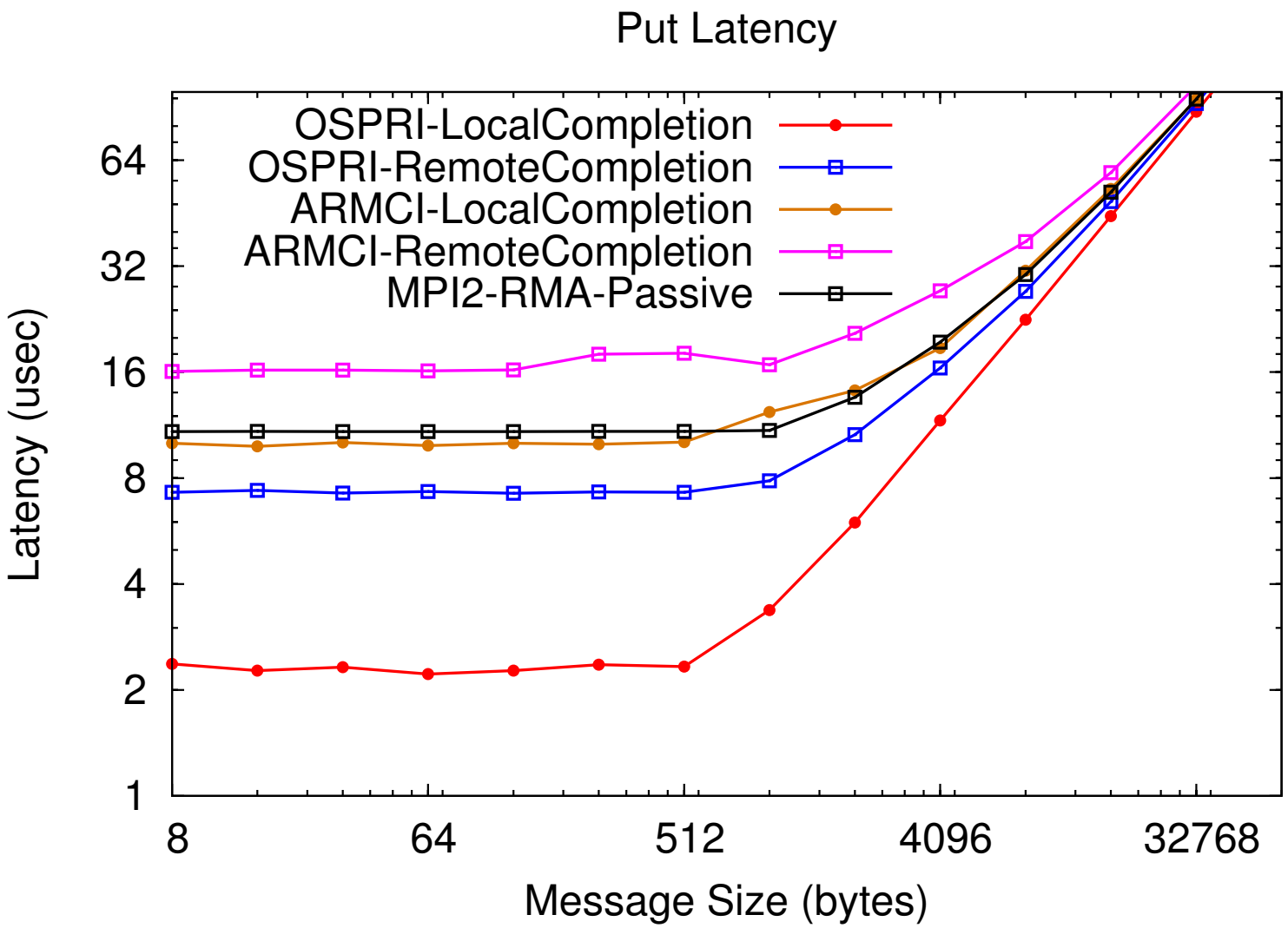
- DCMF provides RDMA Put and Get as well AMs (Send)
- `memcpy` slower than DMA for messages larger than L1
- no performance from network parallelism (but channels work)
- dynamic routing not beneficial (designed for all-to-all)
- contention is a huge problem (not solvable in OSPRI)
- interrupts are useful, but expensive (blow out L1)

Performance Results

Put latency I



Put latency II



Ordering semantics I

Standard data hazards (WAW, WAR, RAW) insufficient for one-sided.

- In general, we have both RDMA and non-RDMA communication (e.g. DCMF Put v. Send).
- For RDMA, packet fifo is the end, AM to CPU then memory.
- Ordering packets is fine for RDMA in practice.
- Same operation may use multiple protocols: Eager v. Rendezvous or Direct v. Packed.
- Local access is another "protocol" to handle (if used).

{Put,Get,Acc,Rmw}After{Put,Get,Acc,Rmw} data hazards with one-sided (Also: {Contig,Strided}After{Contig,Strided}).

Ordering semantics II

We define the following:

- **Strict Ordering** (ARMCI location consistency): all blocking operations happen in-order.
- **Partial Ordering** (what GA requires): blocking operations of a given type happen in-order.
- **No Ordering**: User has to manage all ordering with Fence.

The goal is to optimize all of these and then allow the user to ask for what they need. OSPRI won't penalize user more than hardware requires if SO used.

User can't experiment if they don't have quality implementation of multiple options in the same runtime (UPC strict v. relaxed good).

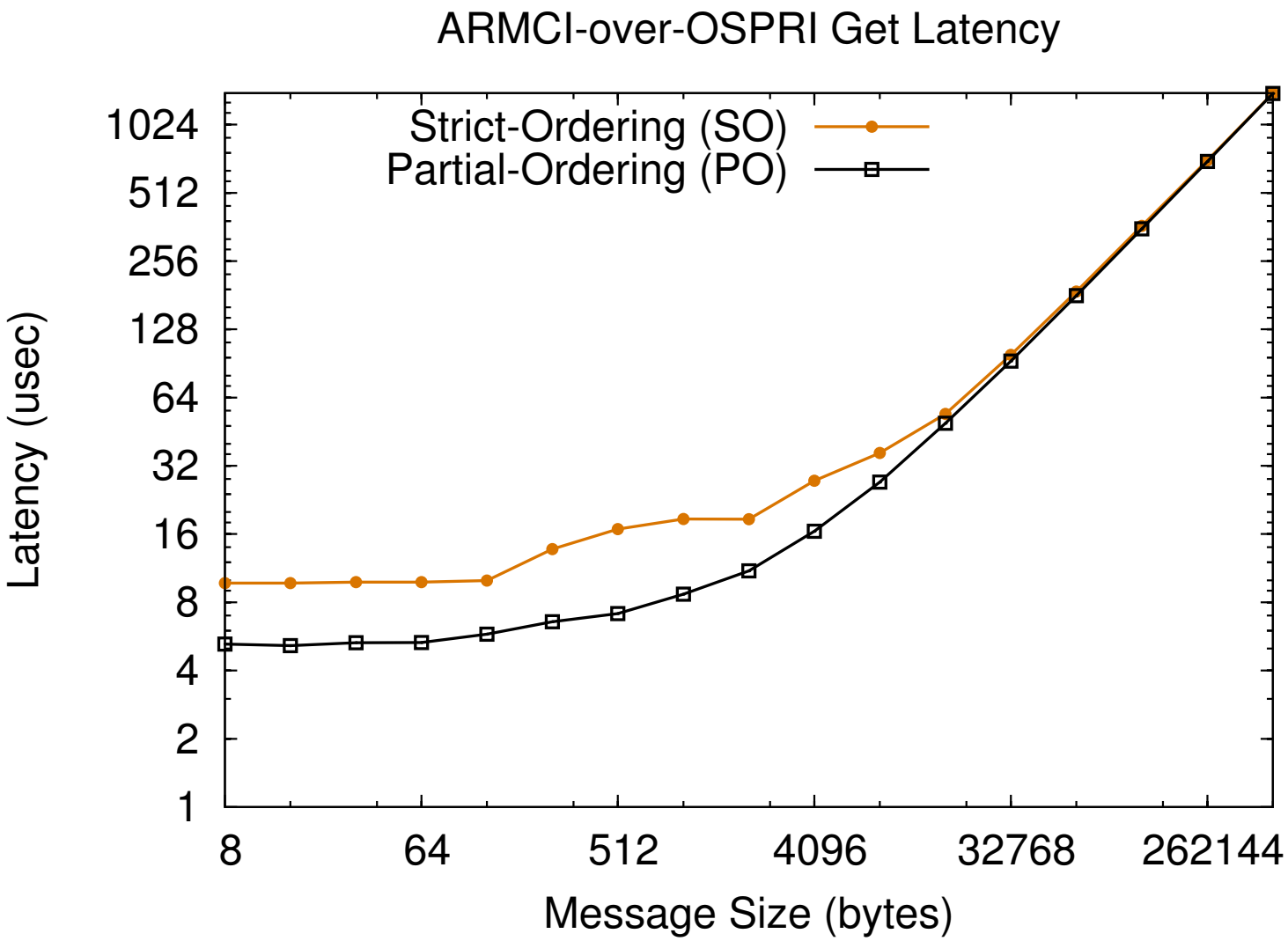
Ordering semantics III

Motivation from implementations:

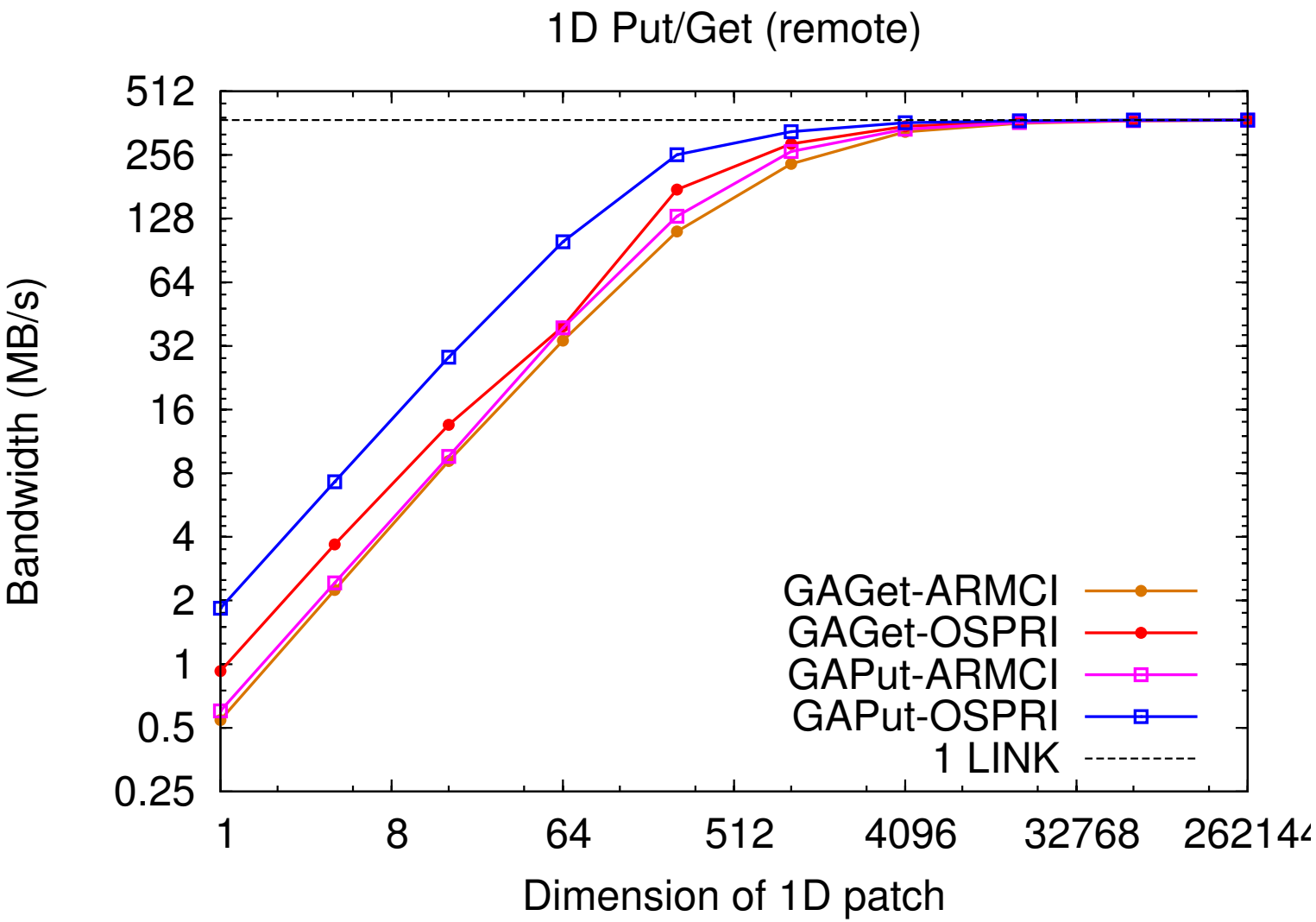
- SO requires AMFence or end-to-end completion of Acc on BG and lock-test on Gemini (assuming LGCPU).
- PO allows all-RDMA for Put and Get on BGP, BGQ and Gemini.
- Multi-protocol (Direct vs. Packed) is local check on BG because we know about outstanding Puts.
- Commutative-associative accumulate operations are not difficult to handle in PO.
- NO allows more network parallelism than PO.

If user disables progress in AMs, need all-RDMA implementation anyways.

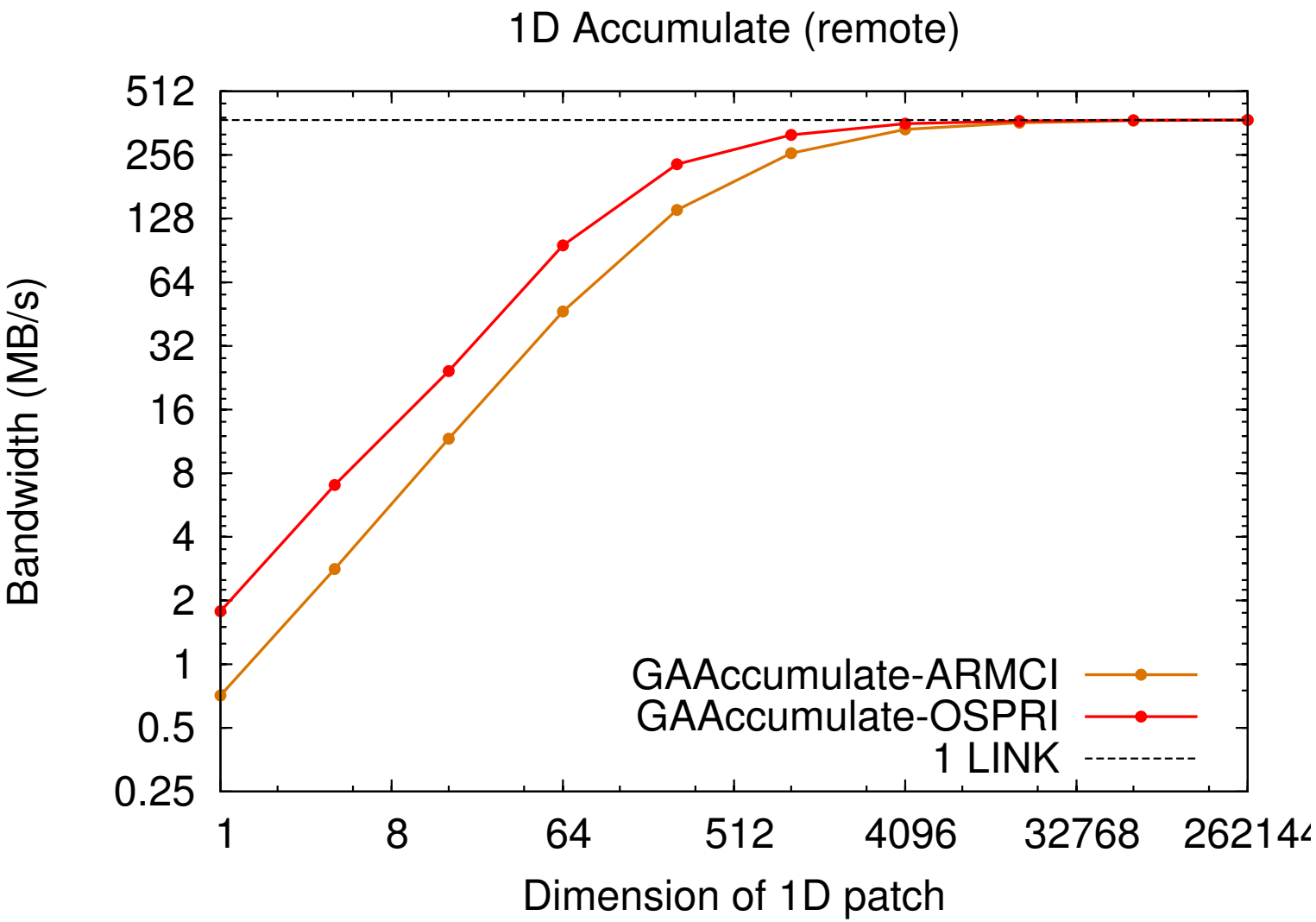
Effect of ordering semantics



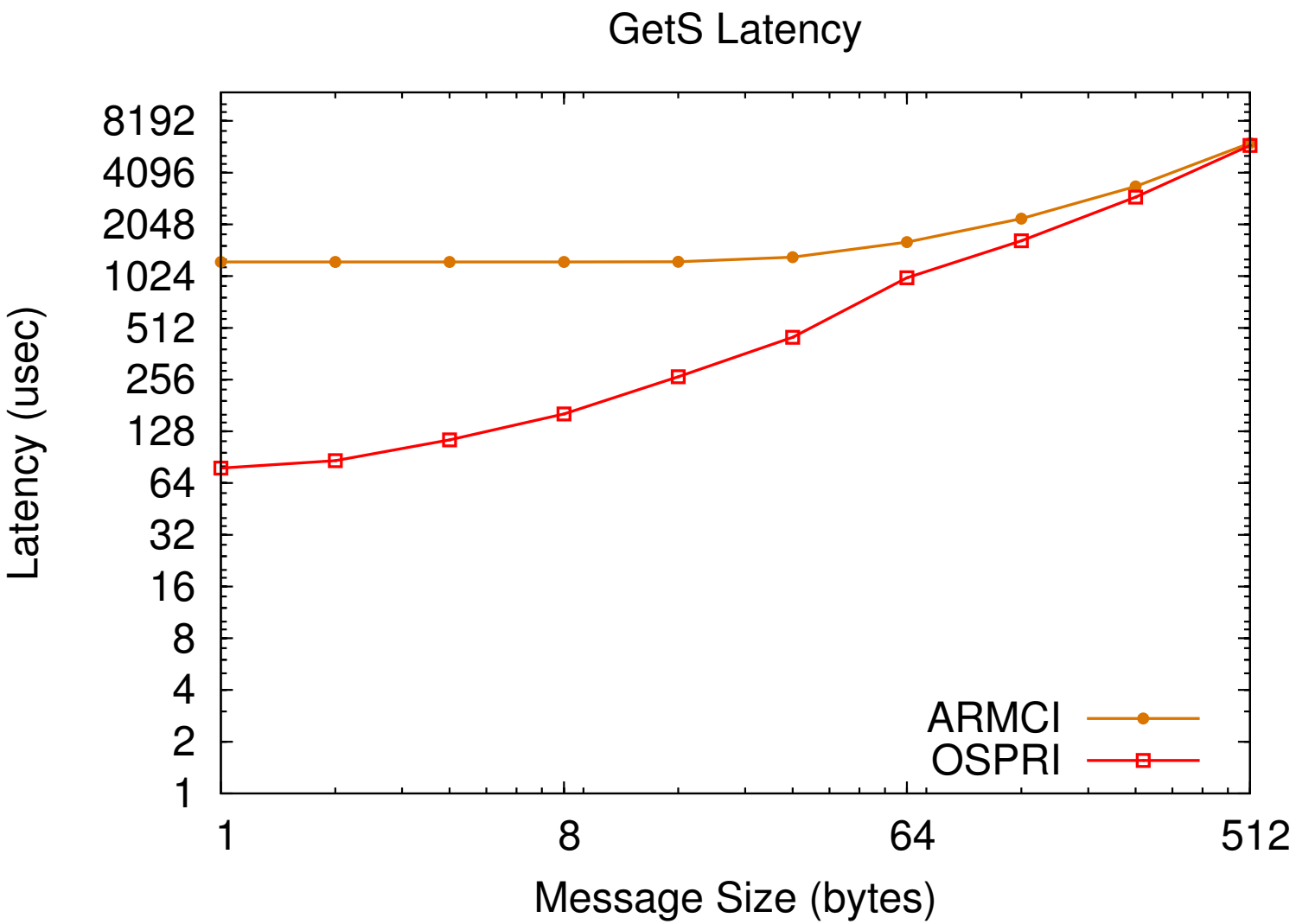
GA Put/Get — 1D remote



GA Acc — 1D remote

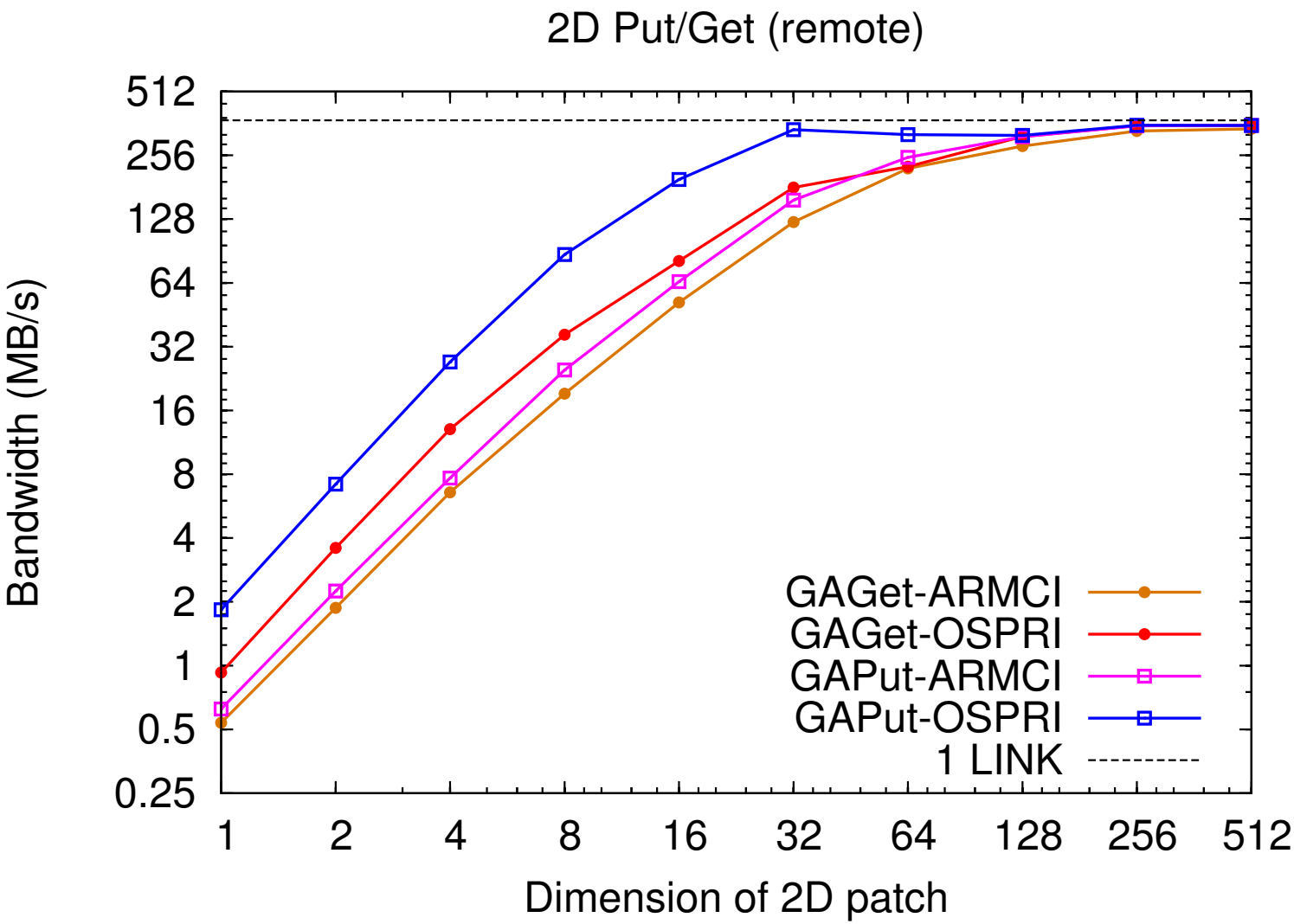


Importance of packing

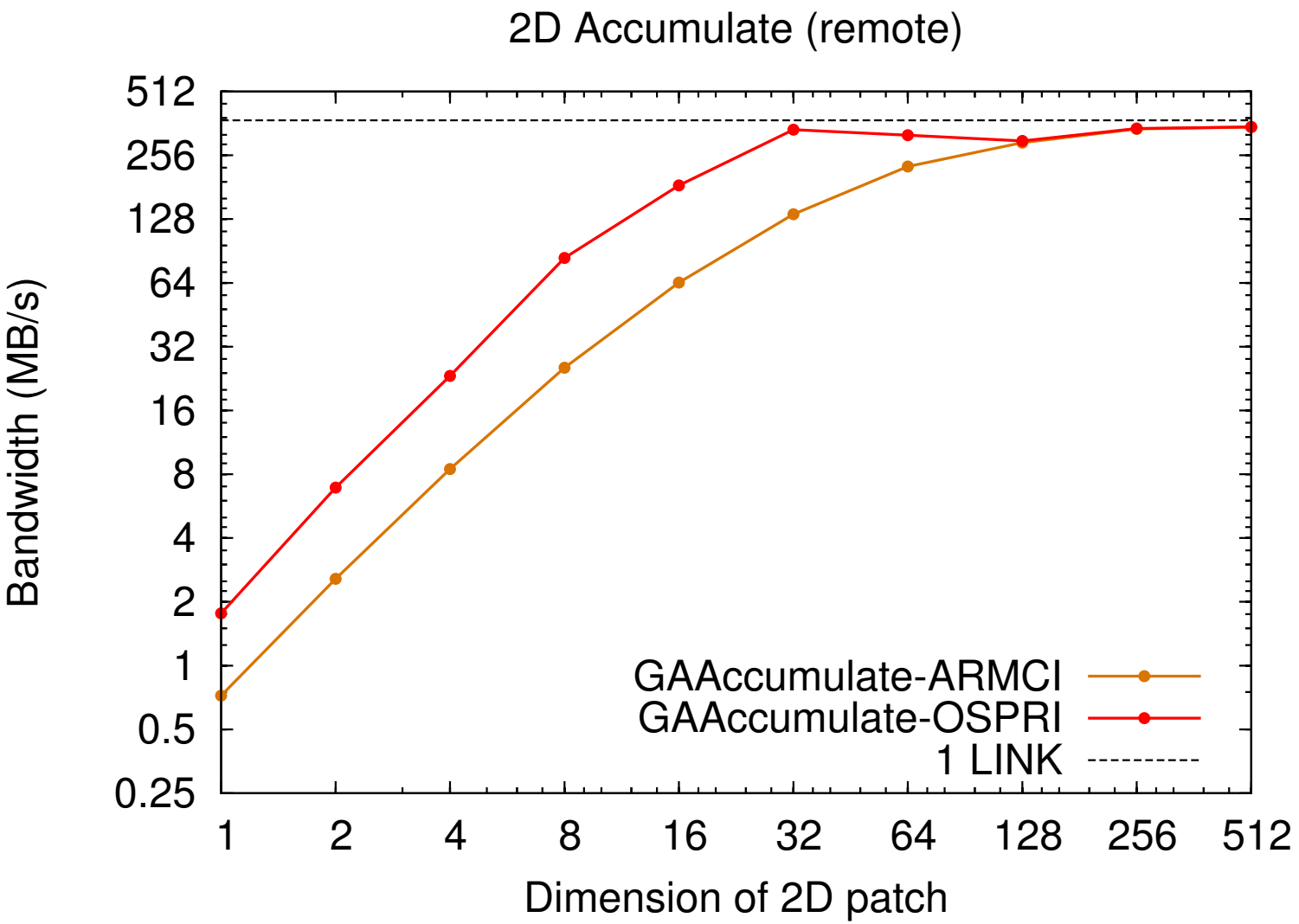


(1024 chunks of message size)

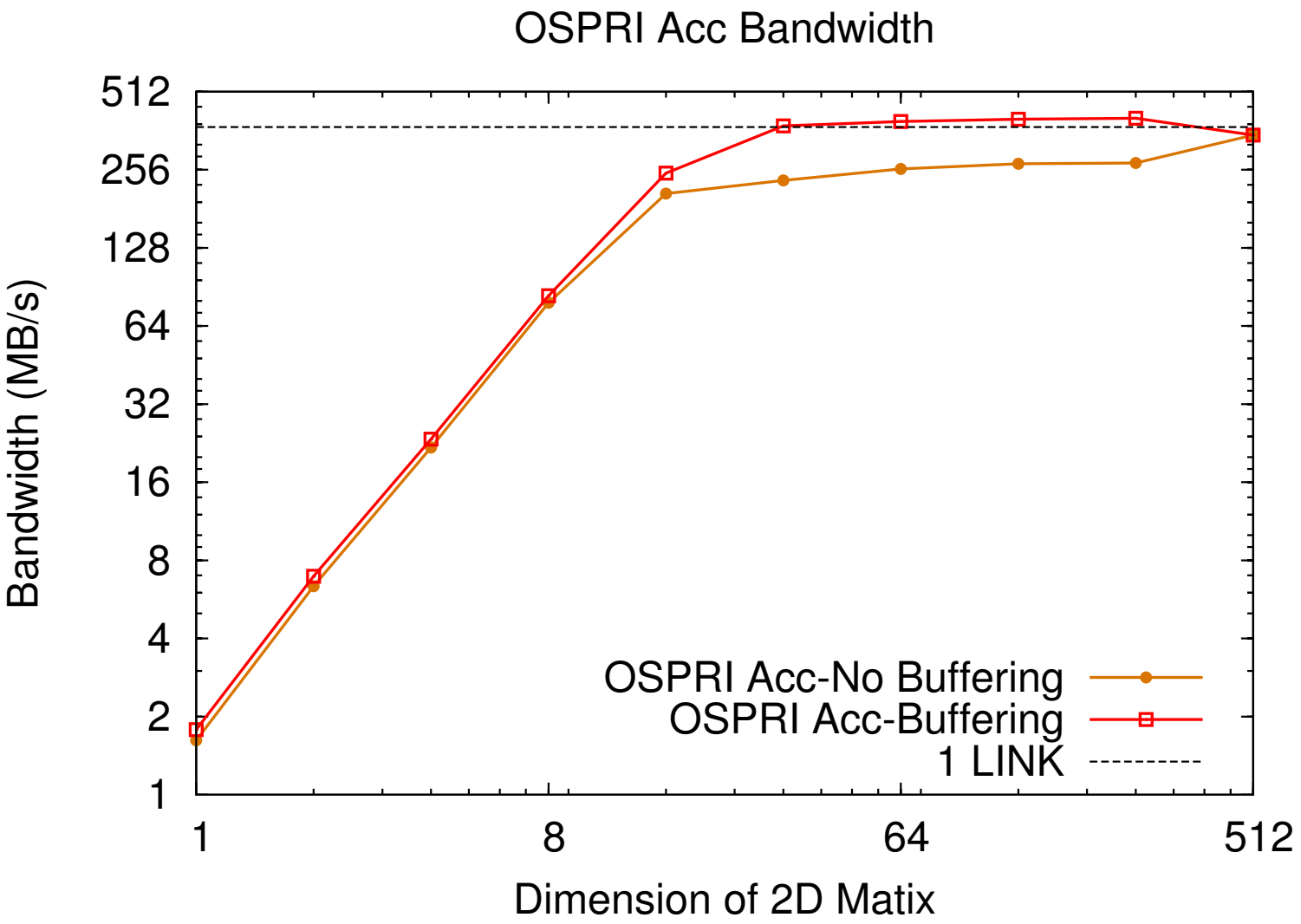
GA Put/Get — 2D remote



GA Acc — 2D remote



Offloaded 2D Accumulate



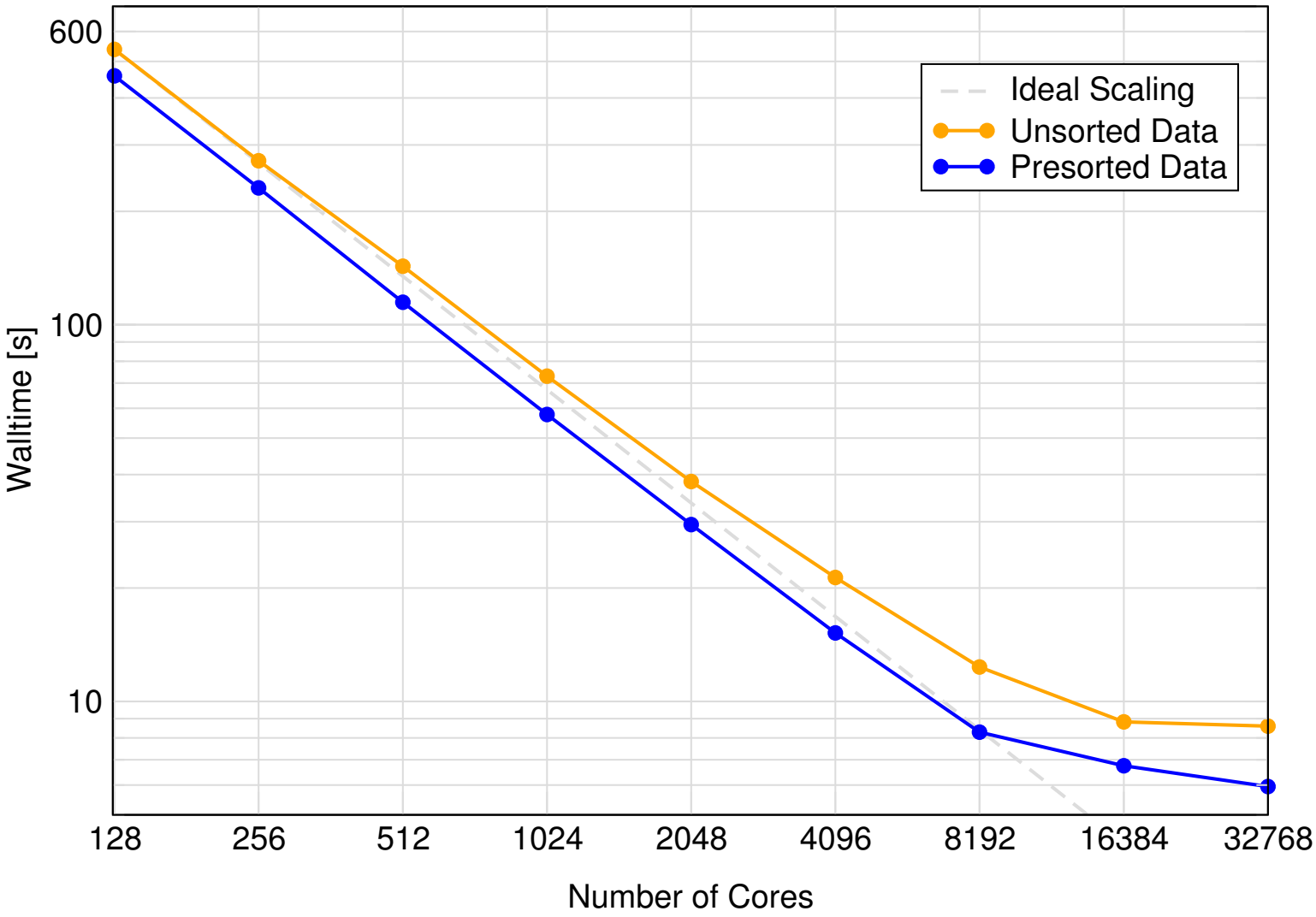
Other performance details

- Rmw is identical to Acc. because we remote complete both (Acc. flow-control problems on BGP); achieves the max of what DCMF can do (no HW atomics on BG).
- Replace $O(N^2)$ registration with Allgather (huge impact on FMM code).
- Fence and AllFence are cheap (RDMA flushes RDMA, AM flushes both) and scalable (also fixed in ARMCI).
- Optimize local access, which GA (esp. NWChem) uses extensively, but not POSIX shared memory due to DMA performance and consistency issues (how to lock a node?).

ScaFaCoS Application Performance

- ScaFaCoS is an N-body solver that uses the Fast Multipole Method.
- Implemented from the beginning using one-sided, first with ARMCI and now with OSPRI-lite.
- Ivo targeting trillions of particles on Blue Gene/P, wants all the cores and all the memory.
- Reduced set of calls - Malloc+Free, Put+Fence, Notify+Wait (or Acc+spin) - so we disable remote agency.
- ARMCI on BG/P stopped scaling/working at 1024 nodes (same for NWChem).

ScaFaCoS Scaling



ScaFaCoS Application Performance

Trillion-particle FMM performance on Jugene with OSPRI.

		Time (s)	
Partition	Particles	Unsorted	Presorted
32768x1	1030607060301	3285	2203
73728x4	2010394559061	2288	530
73728x4	3011561968121	3812	715

Billion-particle FMM performance on Hopper with OSPRI.

		Time (s)	
Partition	Particles	ARMCI-MPI	OSPRI-DMAPP
168x24	1073741824	22.57	8.32

All other Hopper runs failed in NIC...

Comparison of one-sided runtimes

Feature	Progress	Accum.	NonBlock.	NonContig.	Atomics
OSPRI	Yes	Yes	Yes	Yes	Yes
ARMCI	Yes	Yes	Maybe	Yes	Yes
MPI-3	Maybe	Yes	Yes	Yes	Yes
SHMEM	Yes	No	Yes	Partial	Yes
MPI-2	Maybe	Yes	Yes & No	Yes	No
GASNet	No	*No*	Yes	*No*	*No*

Obviously, GASNet can do anything with active-messages, but these need polling for progress, which is totally reasonable for a compilation target.

The arguments for OSPRI over ARMCI or MPI-3 are primarily performance and programmability, not features.

Where is this going?

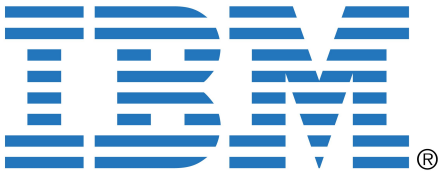
OSPRI for BG/P is not going to be used except by ScaFaCoS...

- Rewrite from the ground up, missing some optimizations, and release for PAMI and DMAPP by the end of the year (?).
- Reference implementation using MPI-RMA (MPI-2 then MPI-3) in 2013, possibly POSIX shm (for SGI?).
- Implement OpenSHMEM and GA-lite (basic features) on top of OSPRI in 2013.
- Infiniband work only if funded to do so.
- Less interested in NWChem; more interested in new applications (and PGAS languages).

I implemented every feature Ivo requested because I wanted a user other than myself. I will be very happy to work with interested parties on features and/or other ports.

Acknowledgments

Co-authors: Jim Dinan, Pavan Balaji, Ivo Kabadshow (FMM), Sreeram Potluri (wrote most of the code) and Vinod Tipparaju.



Michael Blocksome, Brian Smith and Sameer Kumar, for explaining DCMF.

George Almasi for discussions of APGAS.

Howard Pritchard, for explaining DMAPP.

Sriram Krishnamoorthy, for explaining ARMCI.

The Argonne Leadership Computing Facility gave me the freedom to work on this project.