Porting GASNet to Portals: Partitioned Global Address Space (PGAS) Language Support for the Cray XT

Dan Bonachea

Paul Hargrove, Michael Welcome, Katherine Yelick

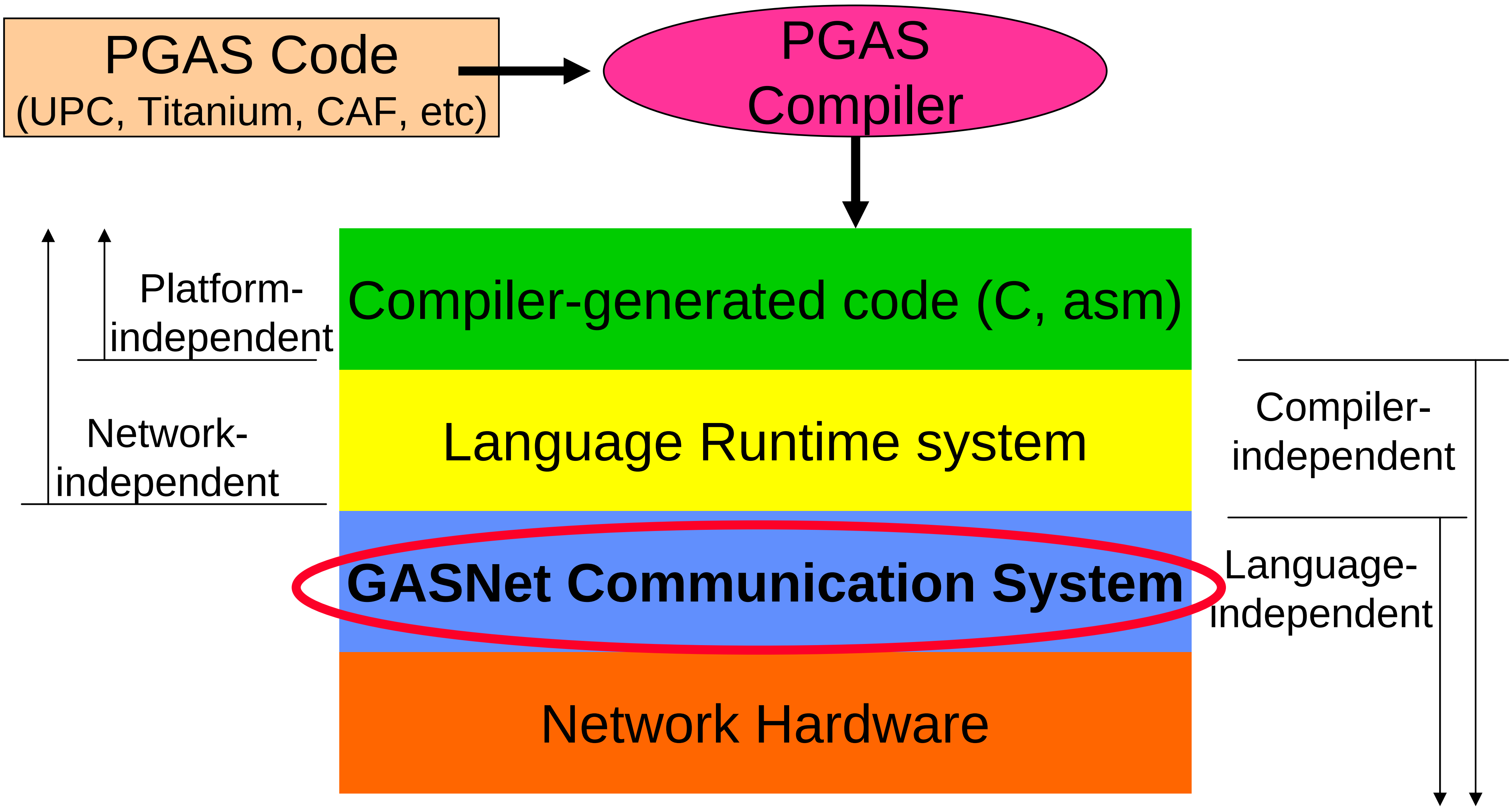
Cray User Group (CUG) 2009

http://gasnet.cs.berkeley.edu
http://upc.lbl.gov

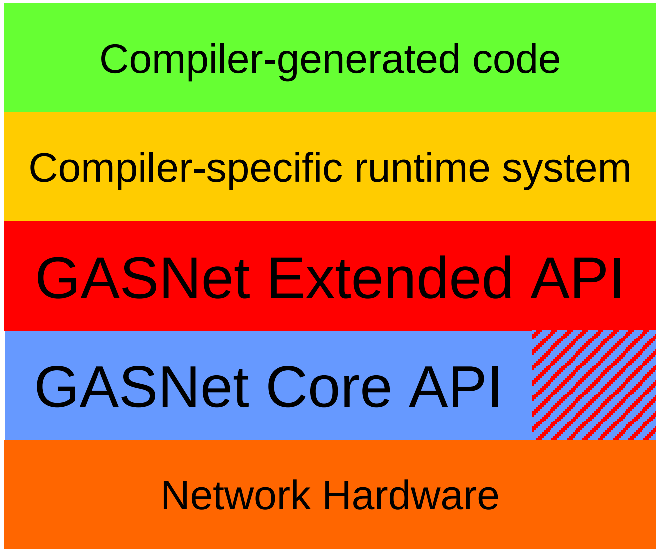
What is GASNet?

- **GASNet is:**
  - **A high-performance, one-sided communication layer**
  - **Portable abstraction layer for the network**
    - **Runs on most architectures of interest to HPC**
    - **Native ports to a wide variety of low-level network APIs**
    - **Can run over portable network interfaces (MPI, UDP)**
  - **Designed as compilation target for PGAS languages**
    - **UPC, Co-array Fortran, Titanium, Chapel,...**
    - **Targeted by 7 separate parallel compiler efforts and counting**
      - **Berkeley UPC, GCC UPC, Cray XT UPC**
      - **Rice CAF, Cray XT CAF, Berkeley Titanium, Cray Chapel**
      - **Numerous prototyping efforts**

PGAS Compiler System Stack



GASNet Design Overview: System Architecture



- Two-Level architecture is mechanism for portability

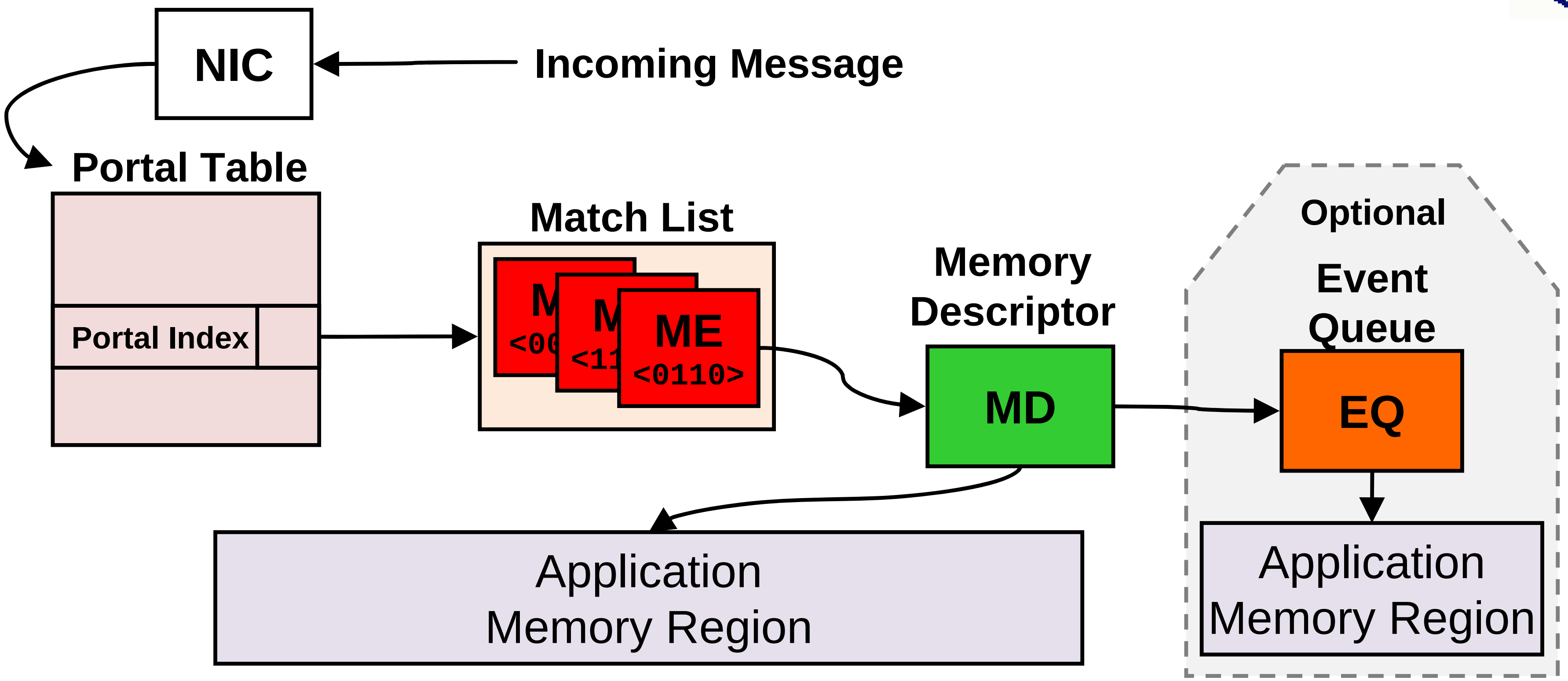
- GASNet Core API
  - Most basic required primitives, narrow and general
  - Implemented directly on each network
  - Based on Active Messages lightweight RPC paradigm

- GASNet Extended API
  - Wider interface that includes higher-level operations
    - puts and gets w/ flexible sync, split-phase barriers, collective operations, etc
  - Have reference implementation of the extended API in terms of the core API
  - Directly implement selected subset of interface for performance
    - leverage hardware support for higher-level operations

GASNet Design Progression on XT

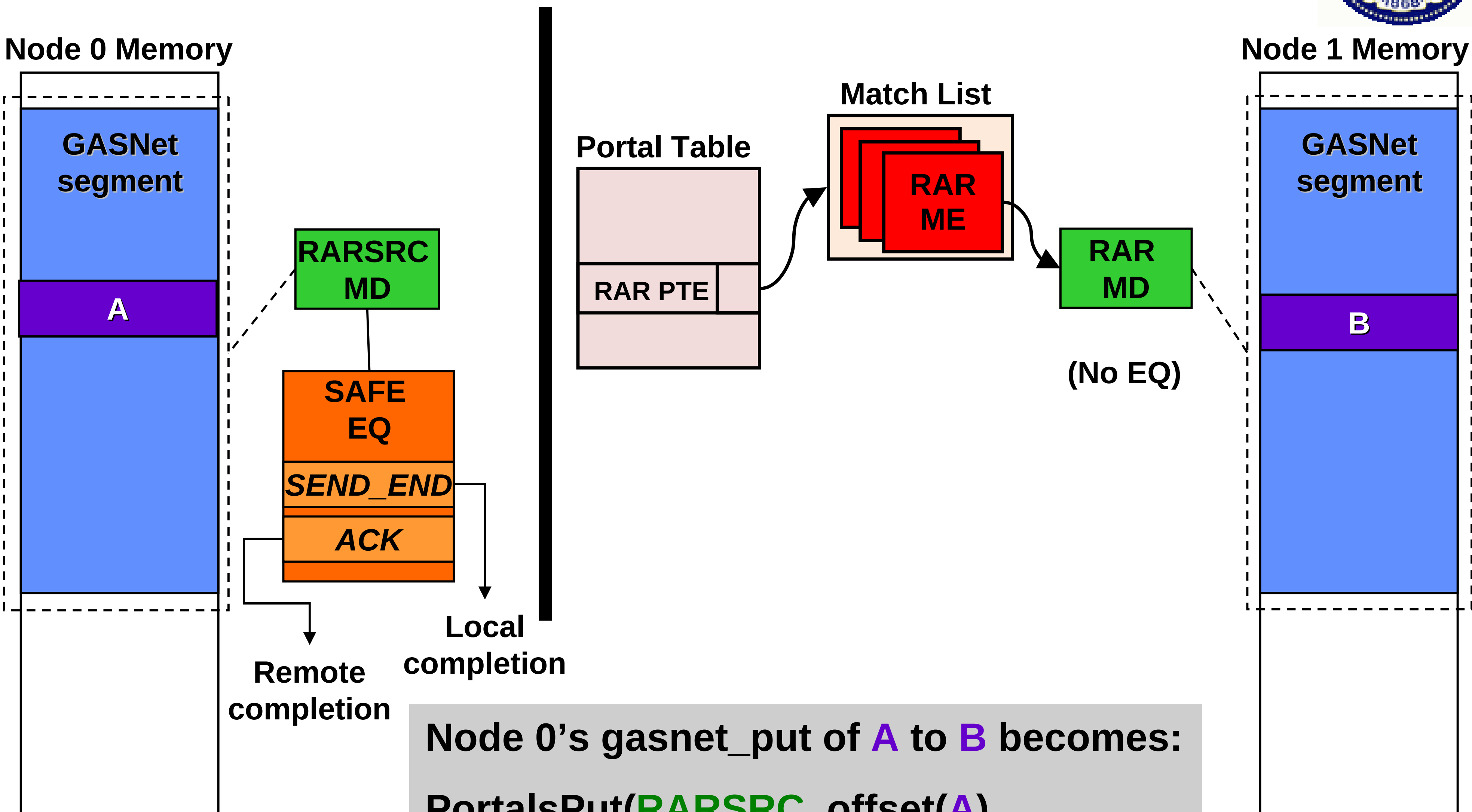
- Pure MPI: mpi-conduit
  - Fully portable implementation of GASNet over MPI-1
  - “Runs everywhere, optimally nowhere”
- Portals/MPI Hybrid
  - Replaced Extended API (put/get) with Portals calls
  - Zero-copy RDMA transfers using SeaStar support
- Pure Portals: portals-conduit
  - Native Core API (AM) implementation over Portals
  - Eliminated reliance on MPI
- Firehose integration
  - Reduce memory registration overheads

Portals Message Processing



- Lowest-level software interface to the XT network is Portals
  - All data movement via Put/Get btwn pre-registered memory regions
  - Provides sophisticated recv-side processing of all incoming messages
- Designed to allow NIC offload of MPI message matching
  - Provides (more than) sufficient generality for our purposes

GASNet Put in Portals-conduit

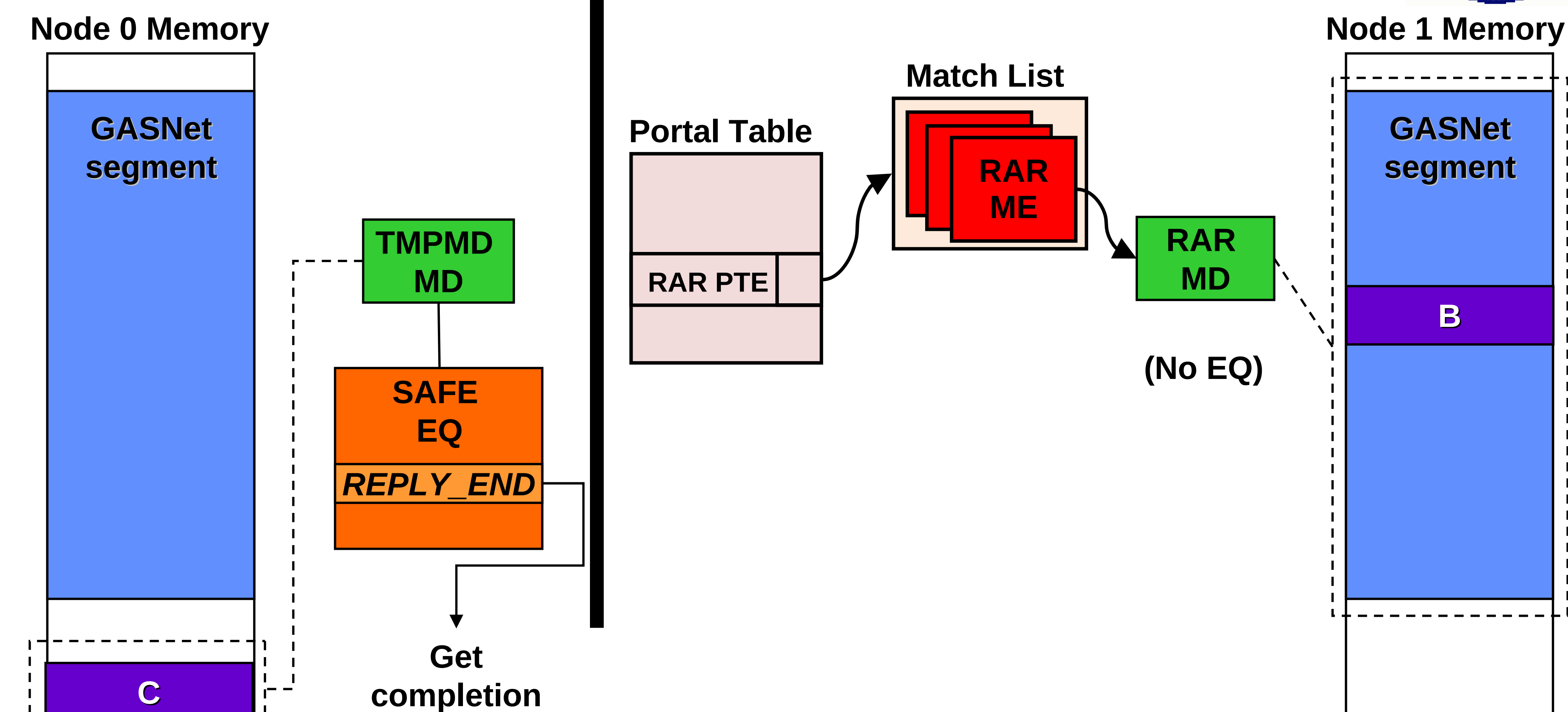


Node 0's gasnet_put of A to B becomes:

PortalsPut(RARSRC, offset(A),
RARME | op_id, offset(B))

Operation identifier smuggled thru ignored match bits

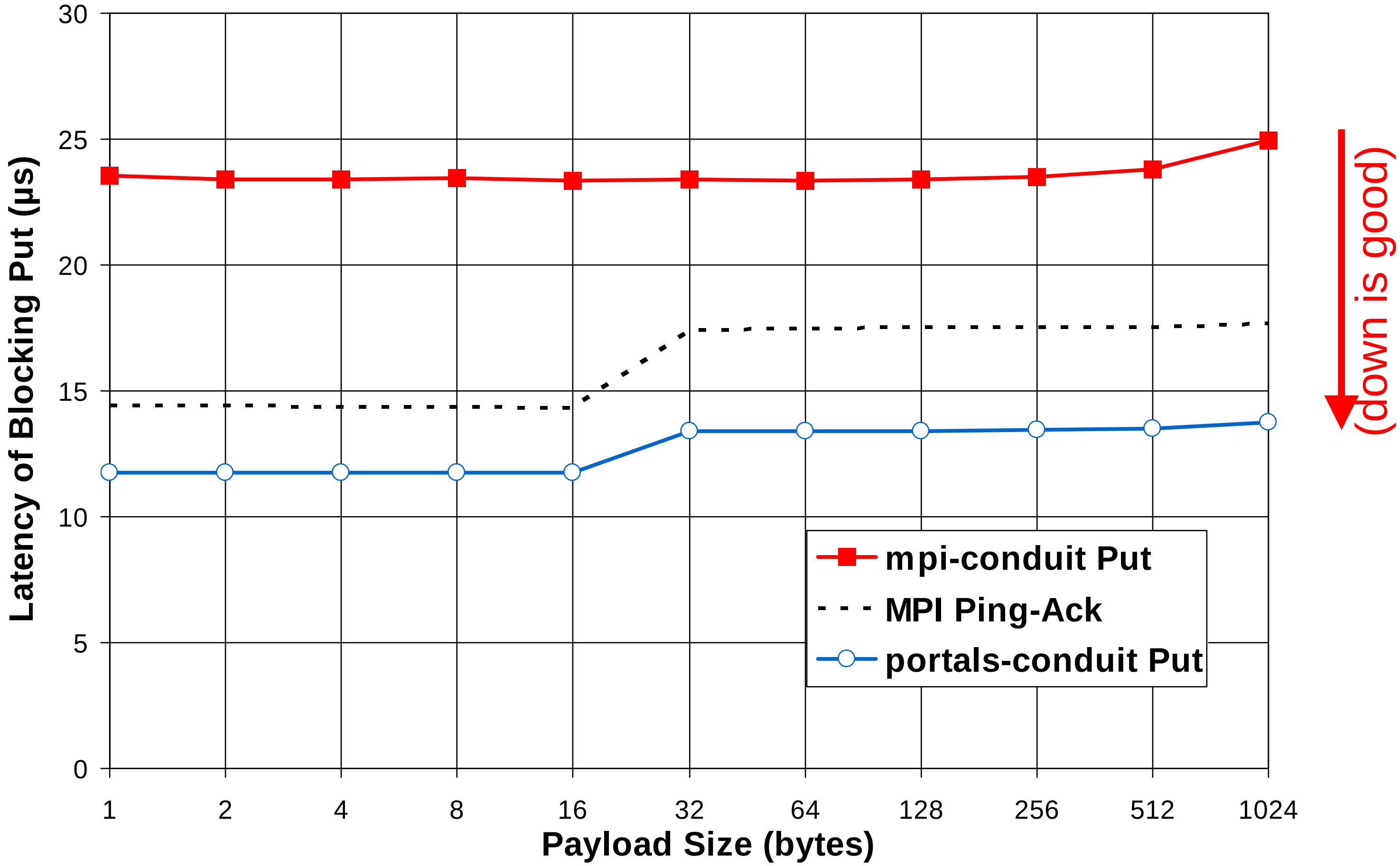
GASNet Get in Portals-conduit



Node 0’s gasnet_get of B to C becomes:

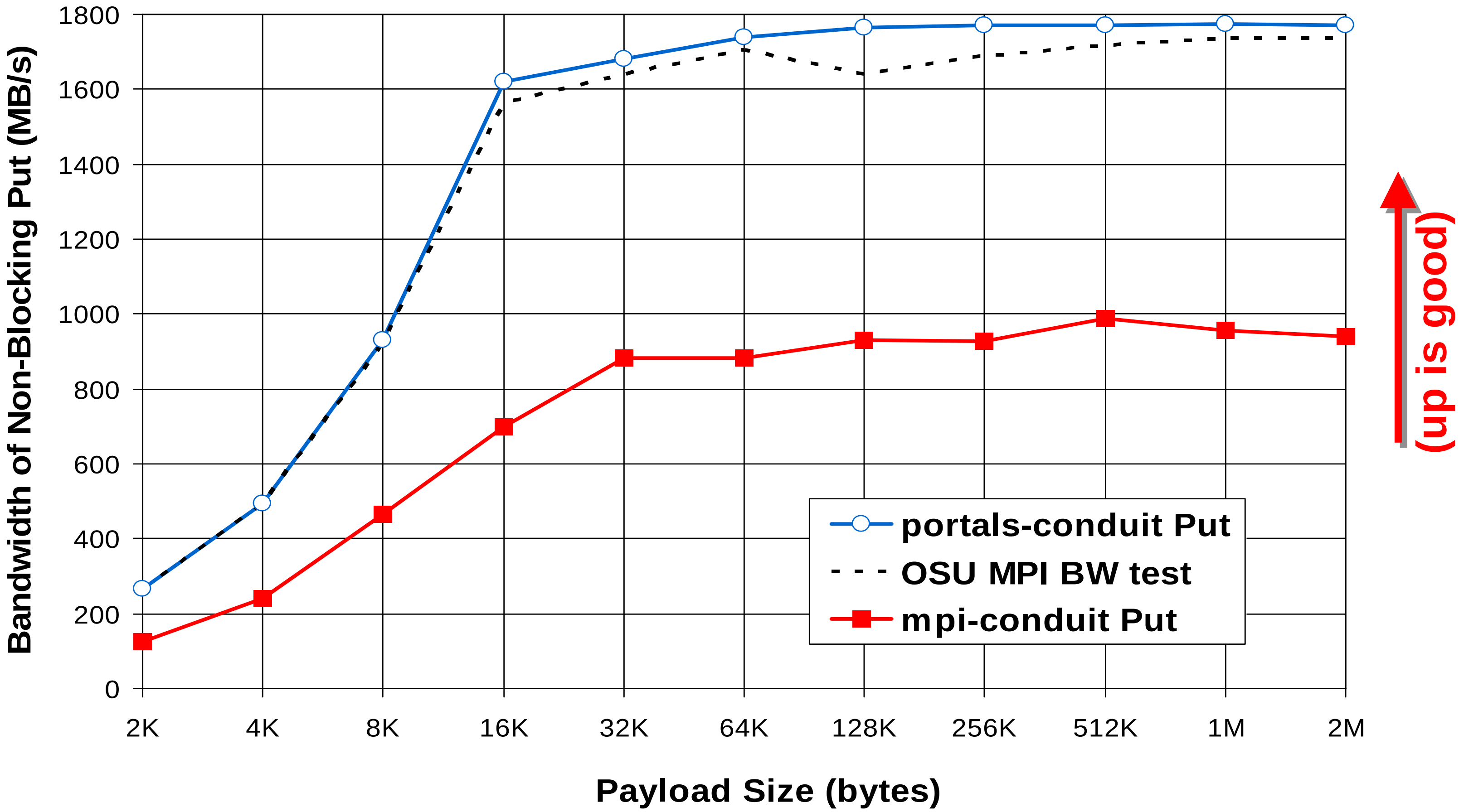
PortalsGet(TMPMD, 0,
RARME | op_id, offset(B))

Performance: Small Put Latency



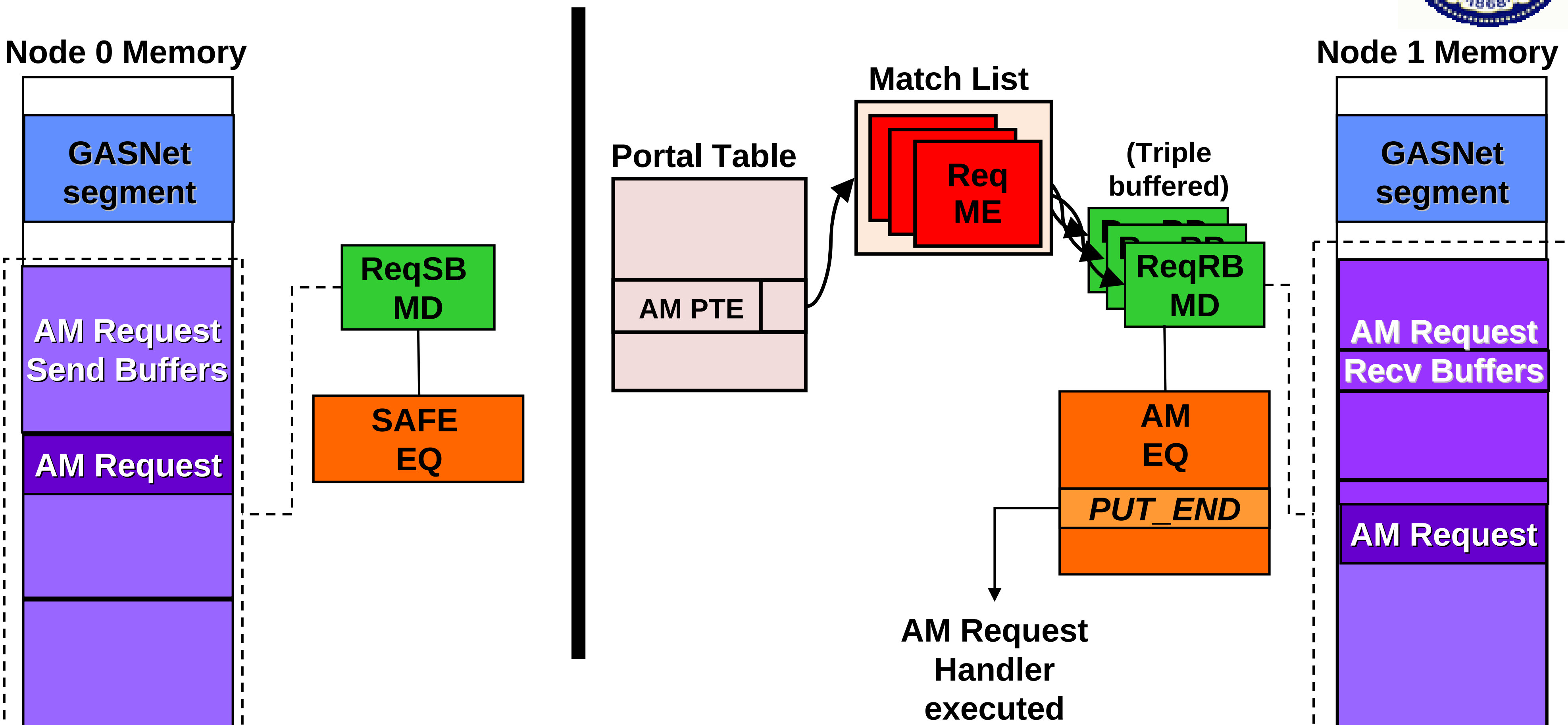
- All performance results taken on 2 nodes of Franklin, quad-core XT4 @ NERSC
- Portals-conduit outperforms GASNet-over-MPI by about 2x
  - Semantically-induced costs of implementing put/get over message passing
  - Leverages Portals-level acknowledgement for remote completion
- Outperforms a raw MPI ping/pong by eliminating software overheads

Performance: Large Put Bandwidth



- Portals-conduit exposes the full zero-copy RDMA bandwidth of the SeaStar
  - Meets or exceeds achievable bandwidth of a raw MPI flood test
  - Mpi-conduit bandwidth suffers due to 2-copy of the payload

GASNet AM Request in Portals-conduit

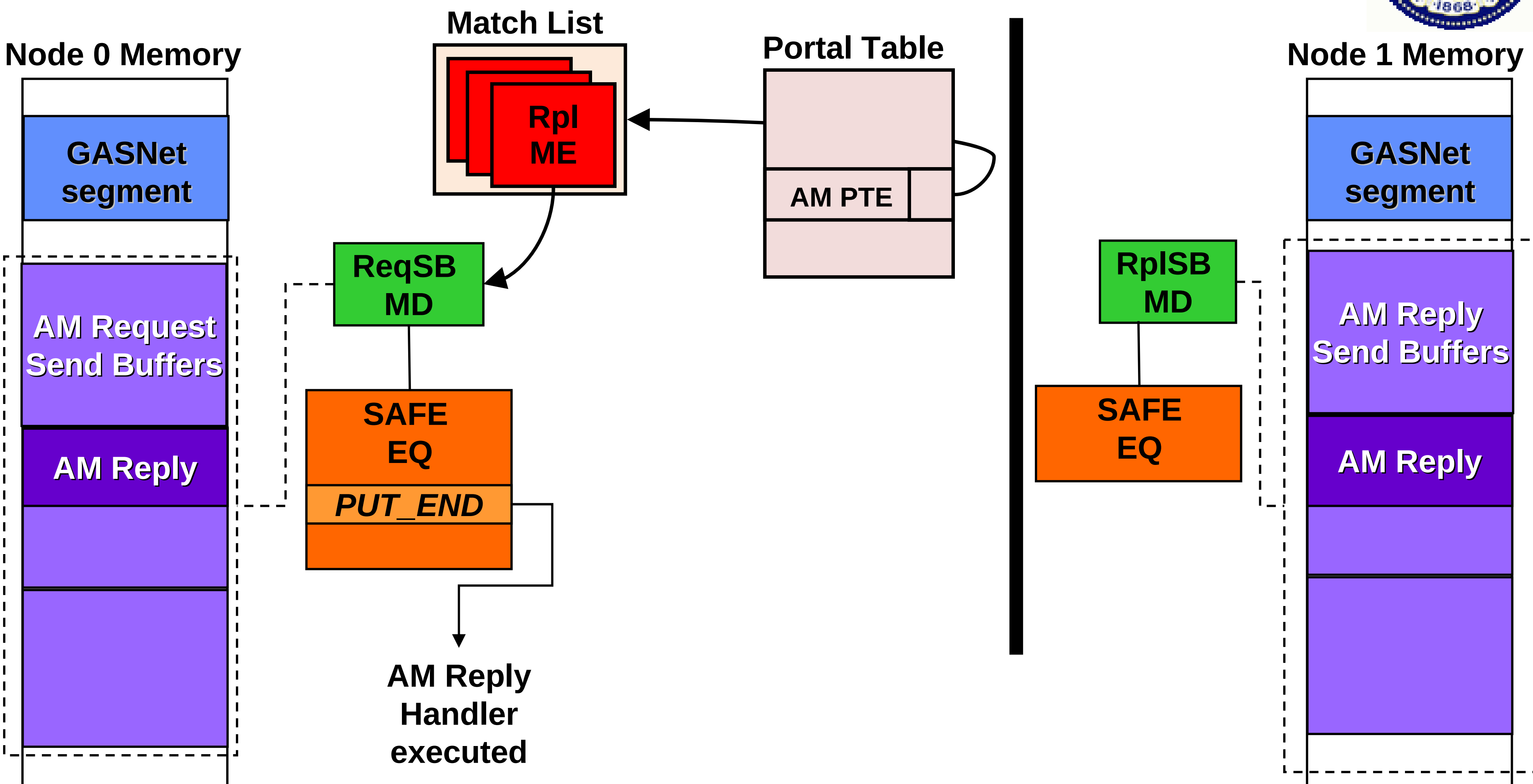


Node 0's gasnet_AMRequestMedium becomes:

PortalsPut(ReqSB_MD, offset(sendbuffer),
Req_ME | op_id | <AM metadata>, 0)

ReqRB has a Locally-managed offset

GASNet AM Reply in Portals-conduit



Node 1's gasnet_AMReplyMedium becomes:

PortalsPut(RplSB_MD, offset(sendbuffer),
Rpl_ME | op_id | <AM metadata>, request_offset)

Portals-conduit Data Structures

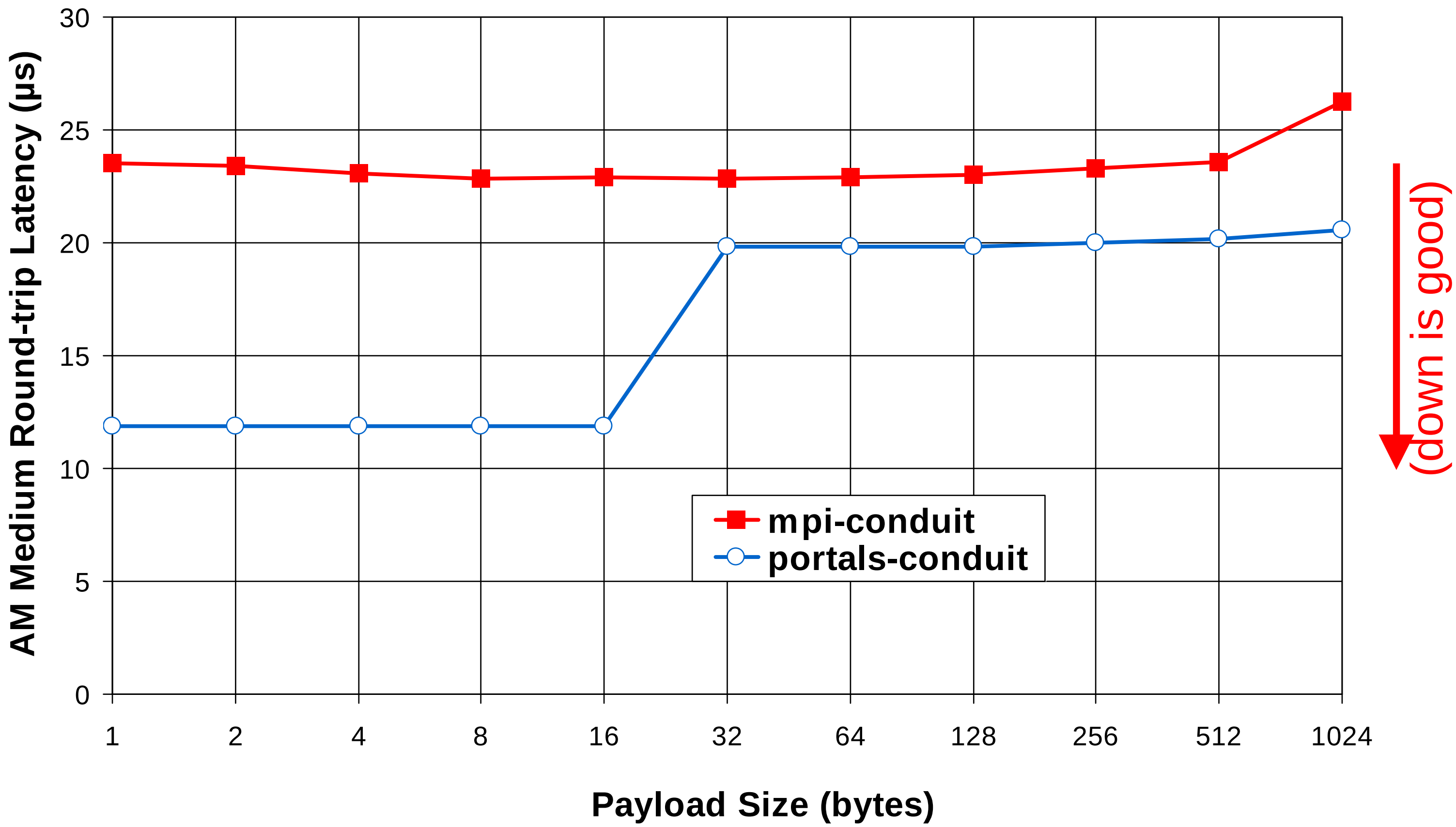
MD	PTE	Match Bits	Ops Allowed	Offset Mgt.	Event Queue	Description
RAR	RAR	0x0	PUT/GET	REMOTE	NONE	Remote segment: dst of Put, src of Get
RARAM	RAR	0x1	PUT	REMOTE	AM_EQ	Remote segment: dst of RequestLong payload
RARSRC	RAR	0x2	PUT	REMOTE	SAFE_EQ	Remote segment: dst of ReplyLong payload Local segment: src of Put/Long payload, dst of Get
ReqRB	AM	0x3	PUT	LOCAL	AM_EQ	Dest of AM Request Header (double-buffered)
ReqSB	AM	0x4	PUT	REMOTE	SAFE_EQ	Bounce buffers for out-of-segment Put/Long/Get, AM Request Header src, AM Reply Header dst
RplSB	none	none	N/A	N/A	SAFE_EQ	Src of AM Reply Header
TMPMD	none	none	N/A	N/A	SAFE_EQ	Large out-of-segment local addressing: Src of Put/AM Long payload, dest of Get

- RAR PTE: covers GASNet segment with 3 MD’s with diff EQs
- AM PTE: Active Message buffers
  - 3 MD’s: Request Send/Reply Recv, Request Recv, and Reply Send
  - EQ separation for deadlock-free AM
- TMPMD’s created dynamically for transfers with out-of-segment local side

Portals-conduit Flow Control

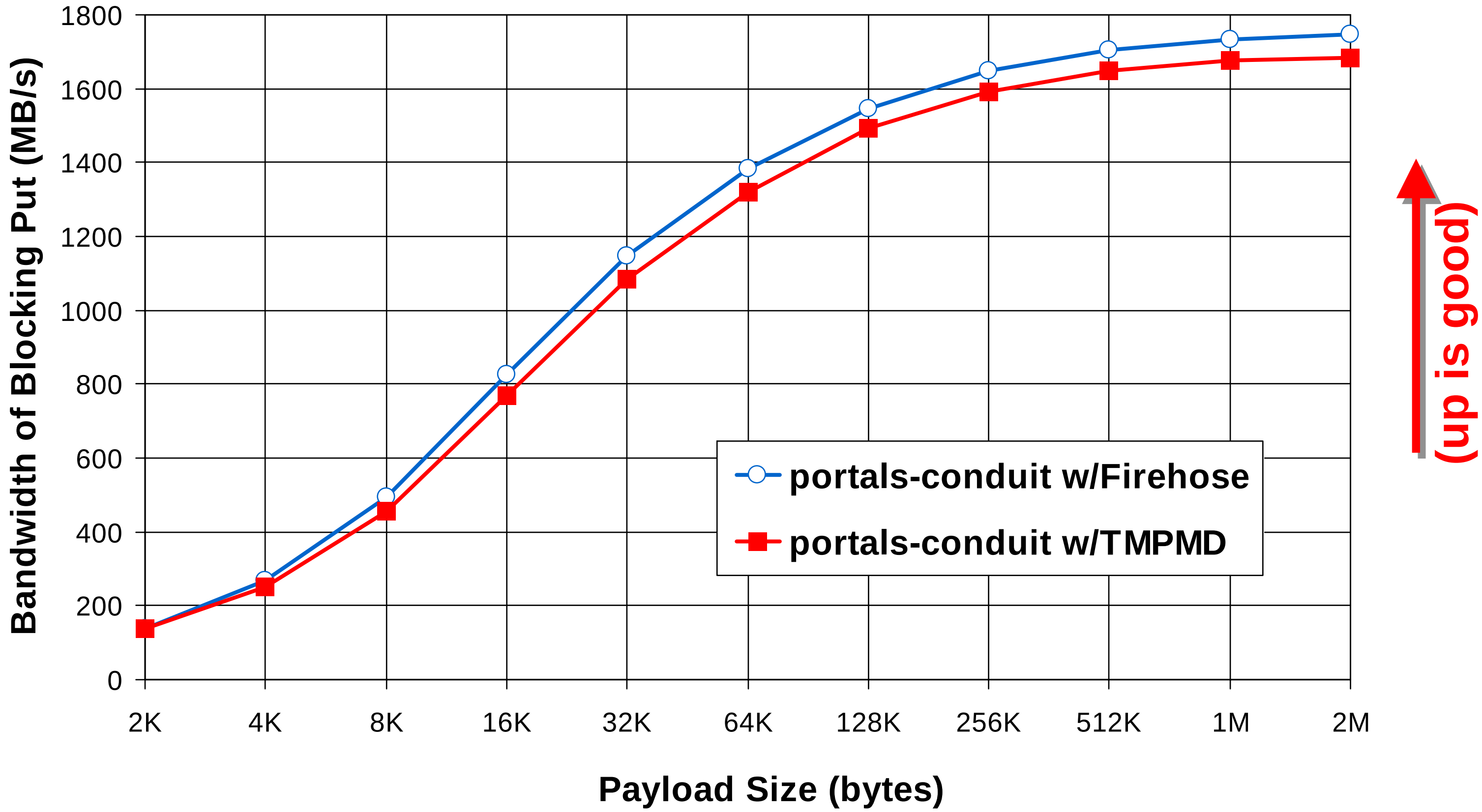
- Most significant challenge in the AM implementation
  - Prevent overflowing recv buffers at the target
  - Prevent overflowing EQ space at either end
- Local-side resources managed using send tokens
  - Request injection acquires EQ and buffer space for send and Reply recv
  - Still need to prevent overflows at remote (target) end
- Initial approach: **Statically Partition** recv resources between peers
  - Reserve worst-case space at target for each sender to get full B/W
  - Initiator-managed, per-target credit system
    - Requests consume credits (based on payload sz), Replies return them
  - Downside: Non-scalable buffer memory utilization
- Final approach: **Dynamic credit redistribution**
  - Reserve space for each receiver to get full B/W
  - Each peer starts with minimal credits, rest banked at the target
  - Target loans additional credits to “chatty” peers, and revokes from “quiet” ones

Performance: Active Message Latency



- Shows the benefit of implementing AM natively
- Portals-conduit AM's outperform mpi-conduit
  - Less per-message metadata, big advantage under 1 packet
  - Beyond one packet, less software overheads w/o MPI

Performance: Out-of-segment Put Bandwidth (Firehose)



- **Blocking** put test (no overlap), exaggerates software overheads
- TMPMD pays synchronous MD create/destroy every transfer
  - Incurs a pinning cost linear in the page count (on CNL)
- Firehose exploits spatial/temporal locality to reuse local MDs
  - LRU algorithm with region coalescing – quickly discovers the working set
  - Provides 4% to 8% bandwidth improvement

Conclusions

- Portals-conduit delivers good GASNet performance on Cray XT
  - Outperforms generic GASNet-over-MPI by about 2x
  - Microbenchmark performance competitive with raw MPI
  - Solid comm. foundation for many PGAS compilers
- Future Work
  - Expand Firehose integration to include remote memory
- Acknowledgements:

  - Thanks to all at Cray who helped in our efforts!
  - Office of Science DOE Contracts DE-AC02-05CH11231, DE-FC03-01ER25509
  - NERSC, DOE Contract DE-AC02-05CH11231
  - ORNL, DOE Contract DE-AC05-00OR22725
  - NSF TeraGrid & PSC System Access

For more information:

http://gasnet.cs.berkeley.edu
http://upc.lbl.gov