11th Charm++ workshop

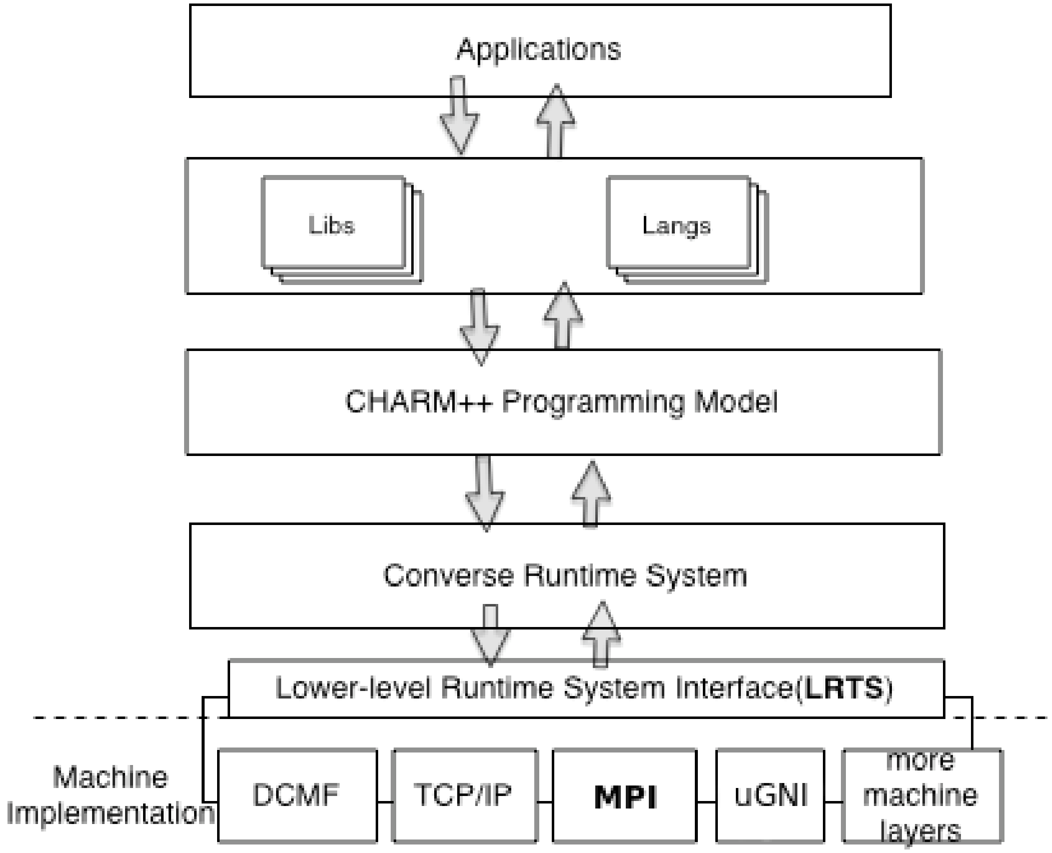
Optimizing Charm++ over MPI

Ralf Gunter,
David Goodell, James Dinan, Pavan Balaji
April 15, 2013

Programming Models and Runtime Systems Group
Mathematics and Computer Science Division
Argonne National Laboratory

rgunter@mcs.anl.gov

The Charm++ stack



(Sun et al., IPDPS '12)

- Runtime goodies sit on top of **LRTS**, an abstraction of the underlying network API.
  - `LrtsSendFunc`
  - `LrtsAdvanceCommunication`
  - Choice of native API (uGNI, DCMF, etc) and MPI.

Why use MPI as the network engine

- Vendor-tuned MPI implementation from day 0.
  - Continued development over machine's life-time.
- Prioritizing development.
  - Charm's distinguishing features sit above this level.
- Reduce resource usage redundancy in MPI interoperability.

Why not use MPI as the network engine

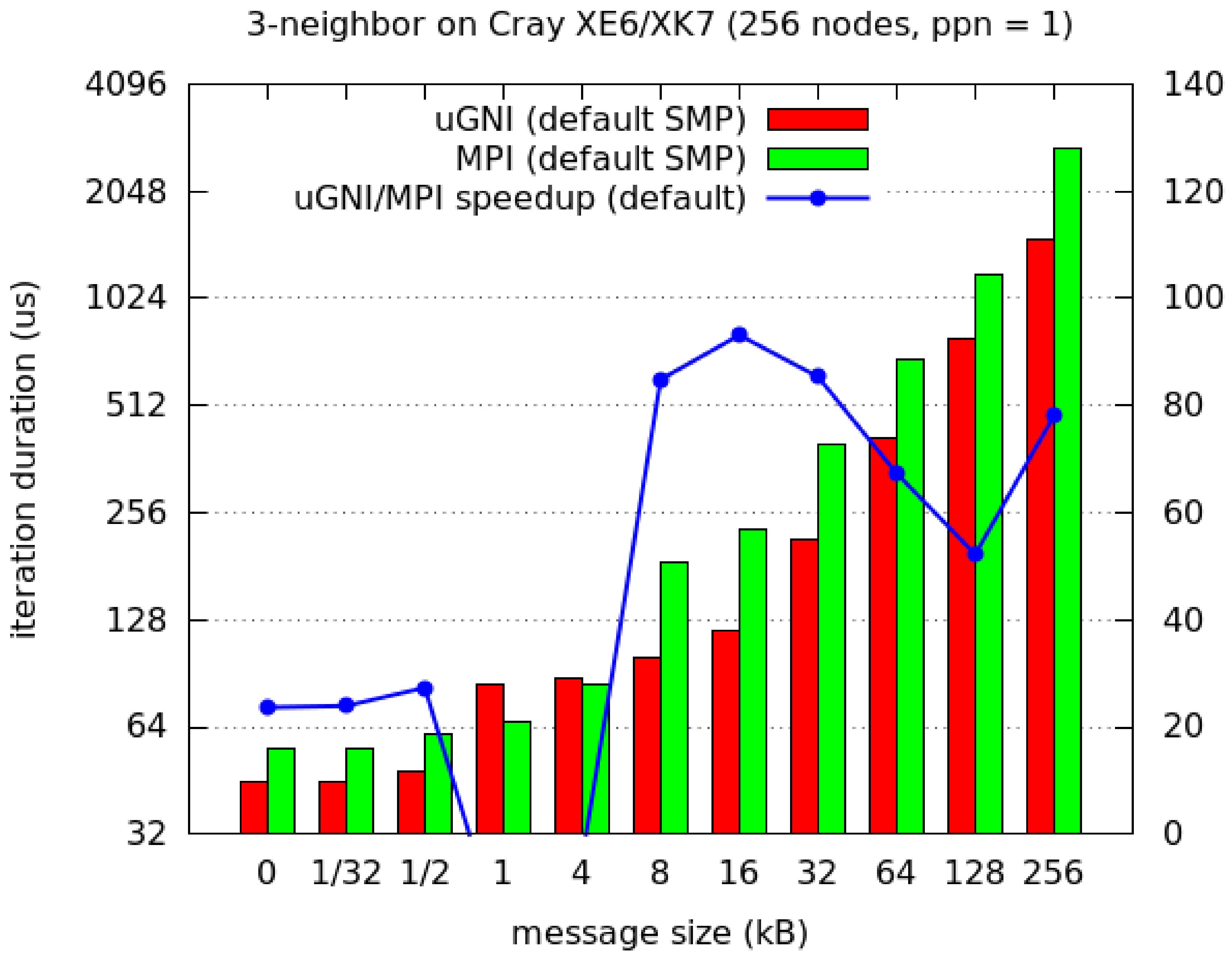
- Unoptimized default machine layer implementation.
  - In non-SMP, communication will stall computation on the rank.
    - Many chares are mapped to the same MPI rank.
  - In SMP, incoming messages are serialized.
- Charm++'s semantics don't play well with MPI's.

Why use MPI as the network engine

- Vendor-tuned MPI implementation from day 0.
  - Continued development over machine's life-time.
- Prioritizing development.
  - Charm's distinguishing features sit above this level.
- Reduce resource usage redundancy in MPI interoperability.

Why not use MPI as the network engine

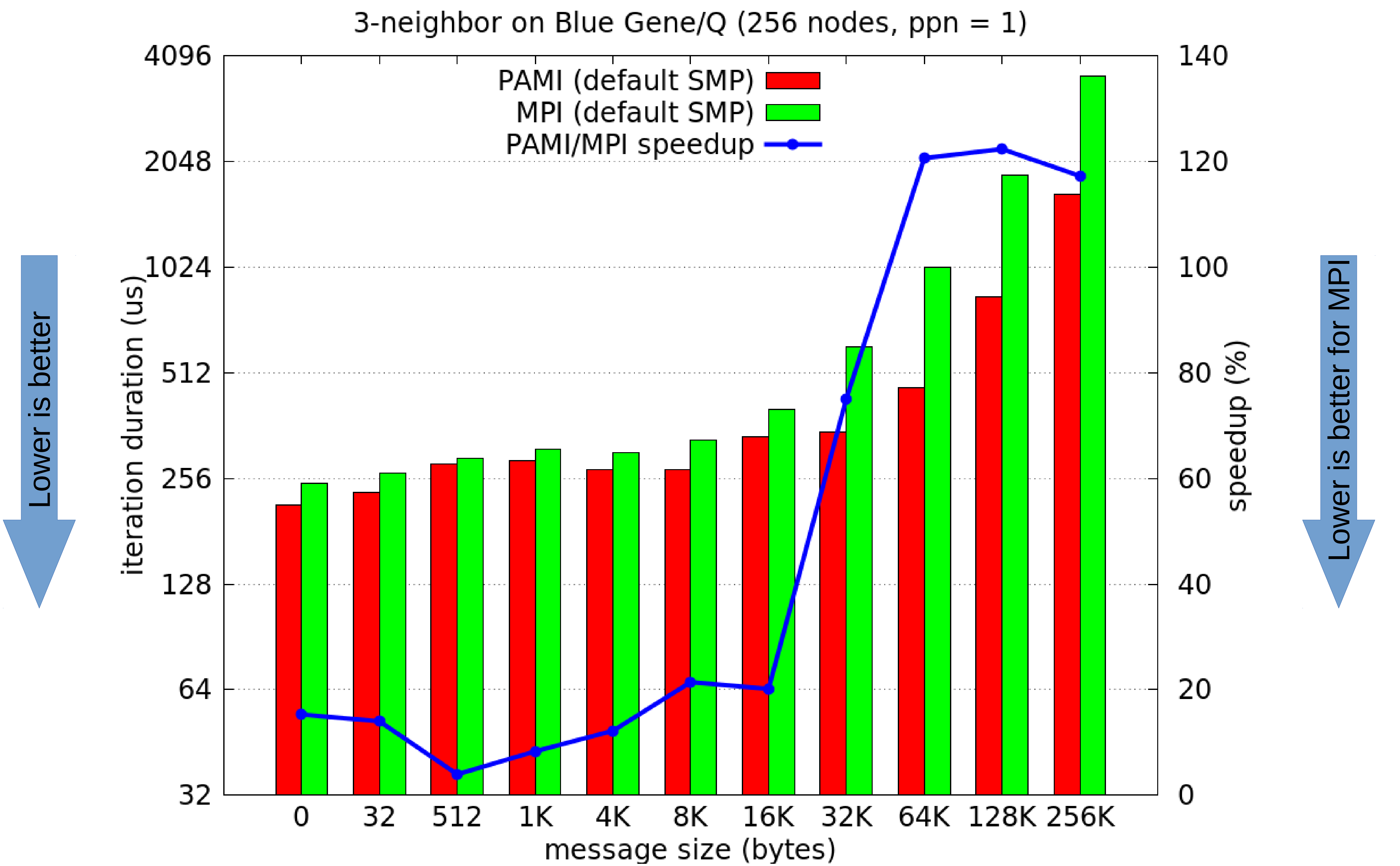
Lower is better



Lower is better for MPI

R. Gunter, D. Goodell, J. Dinan, P. Balaji

Why not use MPI as the network engine



The inadequacy of MPI matching for Charm++

- Native APIs have no concept of source/tag/datatype matching
  - Neither does Charm, but MPI doesn't know it (if using `Send/Recv`)
  - One-sided semantics avoid matching.
    - Can write directly to desired user buffer.
    - Same for rendezvous-based two-sided MPI, but with a receiver synchronization trade-off.
    - Most importantly, it can happen with little to no receiver-side cooperation.

Leveling the field

- Analyzed implementation inefficiencies and semantic mismatches.
  1. MPI implementation issues
     1. MPI's unexpected message queue ✗
  2. Charm++ over MPI implementation issues
     1. MPI Progress frequency ✗
     2. Using MPI `Send`/`Recv` vs. MPI one-sided ✓
  3. Semantics mismatches
     1. MPI tuning for expected vs. unexpected messages ✓

1) Length of MPI's unexpected message queue

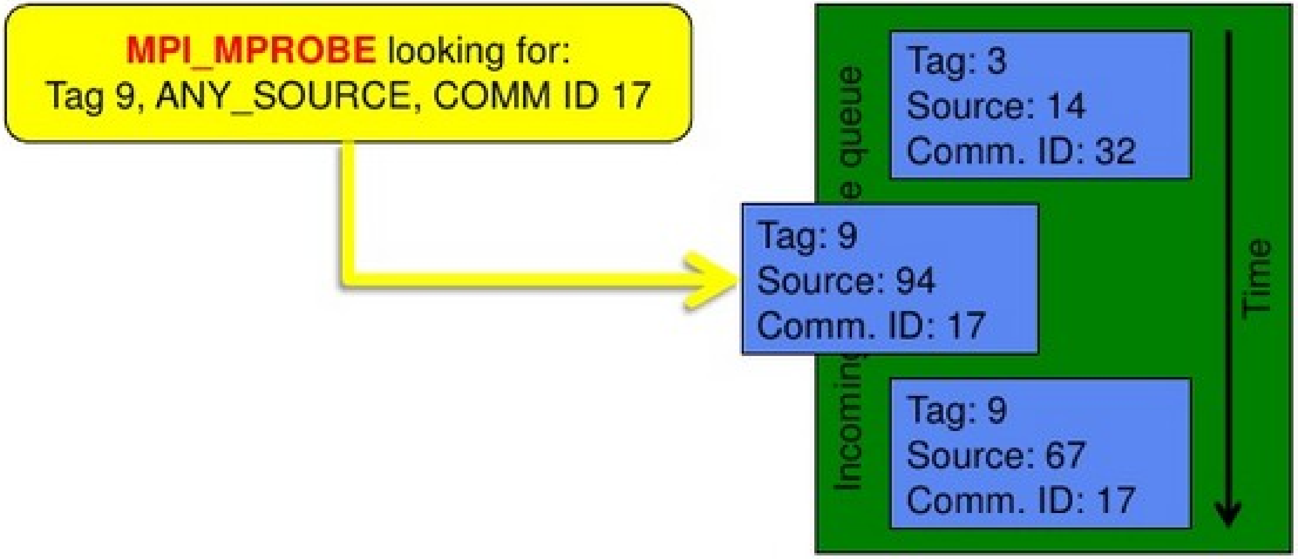
- Unexpected messages (no matching `Recv`) have a twofold cost.
  - `memcpy` from temp to user buffer.
  - Unnecessary message queue searches.
  - Part of why there's an eager and a rendezvous protocol.
- Tested using `MPI_T`, a new MPI-3 interface for performance profiling and tuning.
  - Internal counter keeps track of queue length.
  - Refer to section 14.3 of the standard.

1) Length of MPI's unexpected message queue ✗

- Arguably has no significant impact on performance.
  - Default uses `MPI_ANY_TAG` and `MPI_ANY_SOURCE`, meaning `MPI_Recv` only looks at the head.
  - No need for dynamic tag shuffling (another option in the machine layer).
  - Only affects eager messages.
    - Bulk of rendezvous messages is handled as if expected.

1) Mprobe/Mrecv instead of Iprobe/Recv.

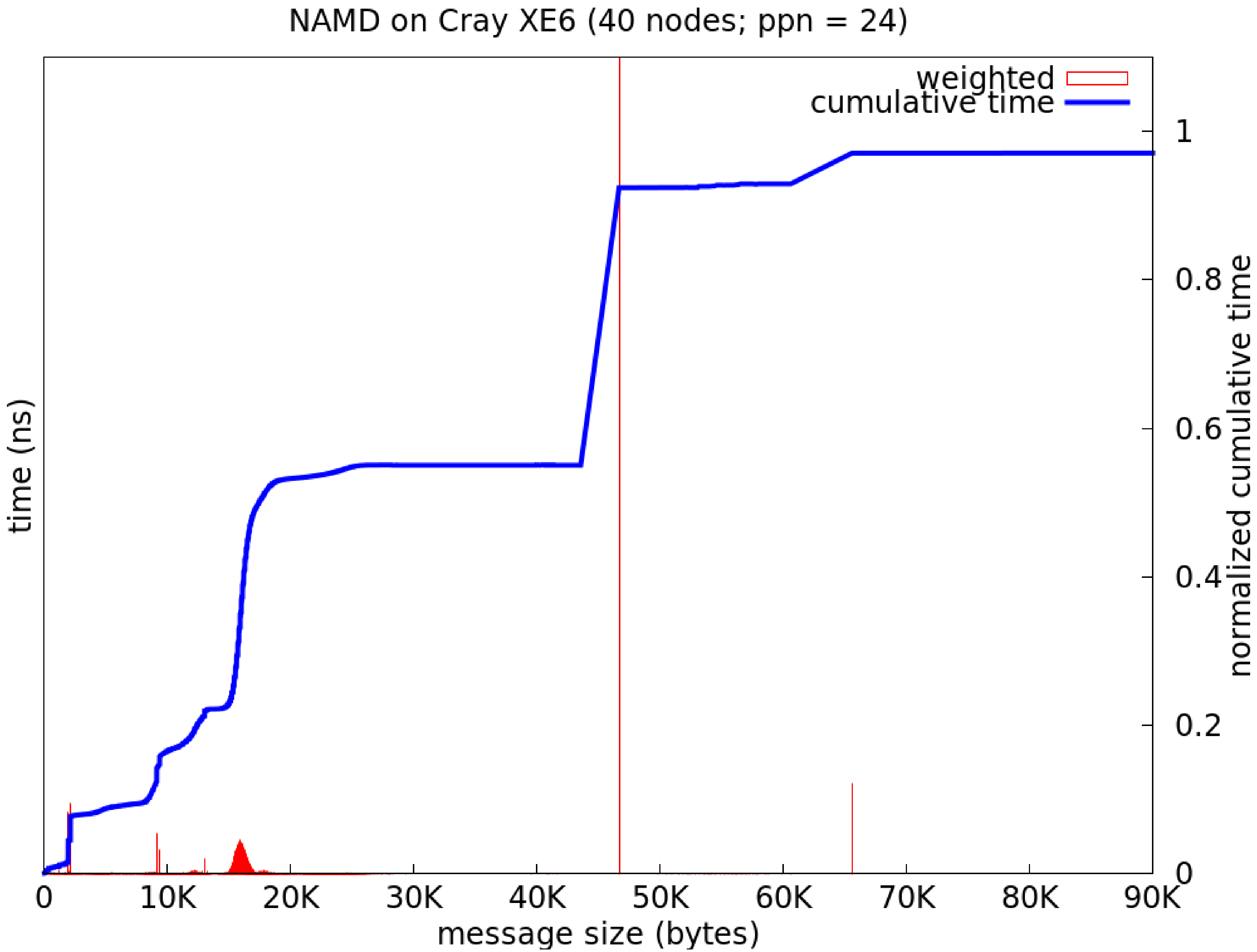
- In schemes with multiple tags, `MPI_Iprobe` + `MPI_Recv` walks the queue twice.
- `MPI_Mprobe` instead deletes entry from queue and outputs a handle to it, used by `MPI_Mrecv`.
- No advantage with double wildcard matching.
- Reduced critical section may help performance with multiple commthreads.



2) MPI progress engine frequency ✗

- In Charm, failed `Iprobe` calls drive MPI's progress engine.
  - Pointless spinning around if are no incoming messages.
- Tried reducing calling frequency to 1/16-1/32th of the default rate.
  - Reduces unexpected queue length.
  - Little to no benefit.
    - Network may need it to kickstart communication.

3) Eager/rendezvous threshold ✓



3) Eager/rendezvous threshold ✓

- **Builds on idea of asynchrony.**
  - Rendezvous needs active participation from receiver.
- Forces use of preregistered temp buffers on some machines.
- Environment vars aren't the appropriate granularity.
  - Implemented per-communicator threshold on MPICH.
    - Specified using info hints (section 6.4.4).

    - Each library may tune their communicator differently.
    - Particularly useful with hybrid MPI/charm apps.
    - Available starting from MPICH 3.0.4.

4) Send/Recv vs one-sided machine layer ✓

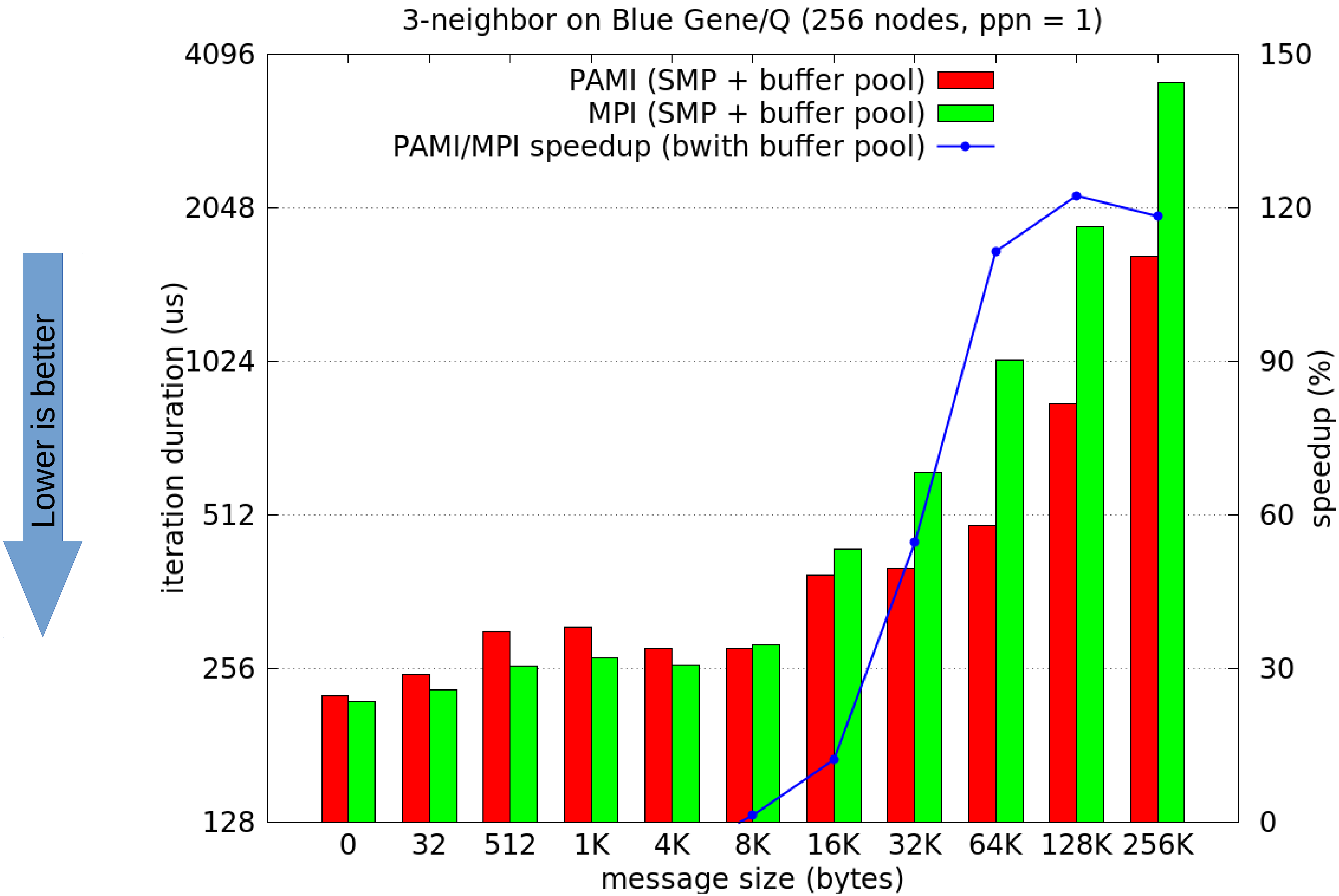
- **Implemented machine layer** using MPI-3 RMA to generalize what native layers do.
  - Dynamic windows (attaching buffers non-collectively);
  - Multi-target locks (`MPI_Win_lock_all`);
  - Request-based RMA Get (`MPI_Rget`).
  - Based on “control message” scheme.
    - Sends small messages directly; larger ones happen via MPI-level RMA.
  - Handles multiple incoming messages concurrently.
  - Can't be tested yet for performance.
    - IBM and Cray MPICH don't currently support MPI-3.

R. Gunter, D. Goodell, J. Dinan, P. Balaji

Current workarounds using MPI-2

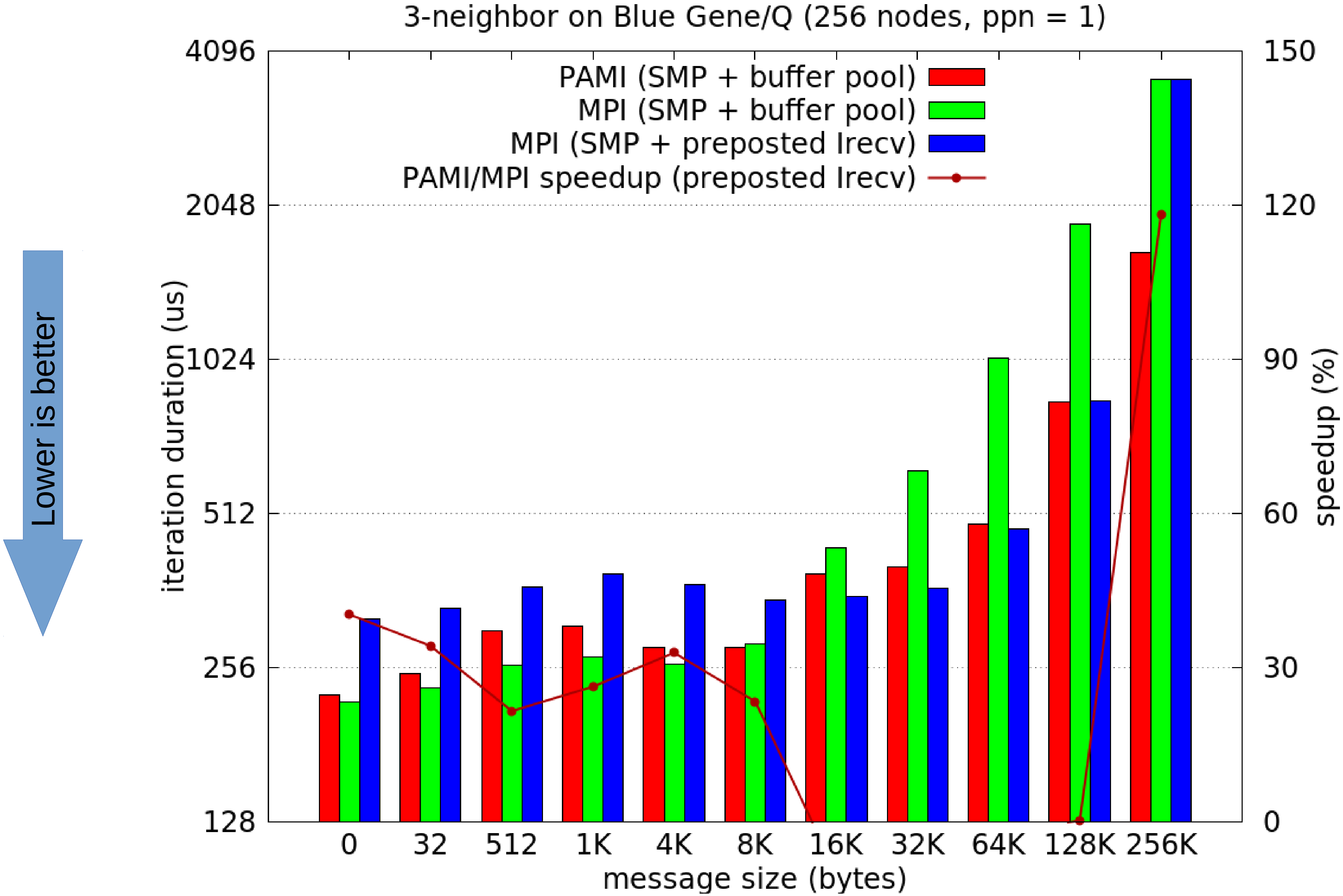
- **Blue Gene/Q**: use the **pamilrts buffer pool** and **preposted MPI_Irecvs** (toggle `MPI_POST_RECV` on machine.c to 1).
  - Interconnect seems to be more independent from software for RDMA
    - Preposting `MPI_Irecv` help it handle multiple incoming messages.
- **Cray XE6 (and InfiniBand clusters)**: increase **eager threshold** to a reasonably large size.
  - Cray's eager (E1) and rendezvous (R0) protocols differ mostly in their usage of preregistered buffers.

Nearest-neighbors results



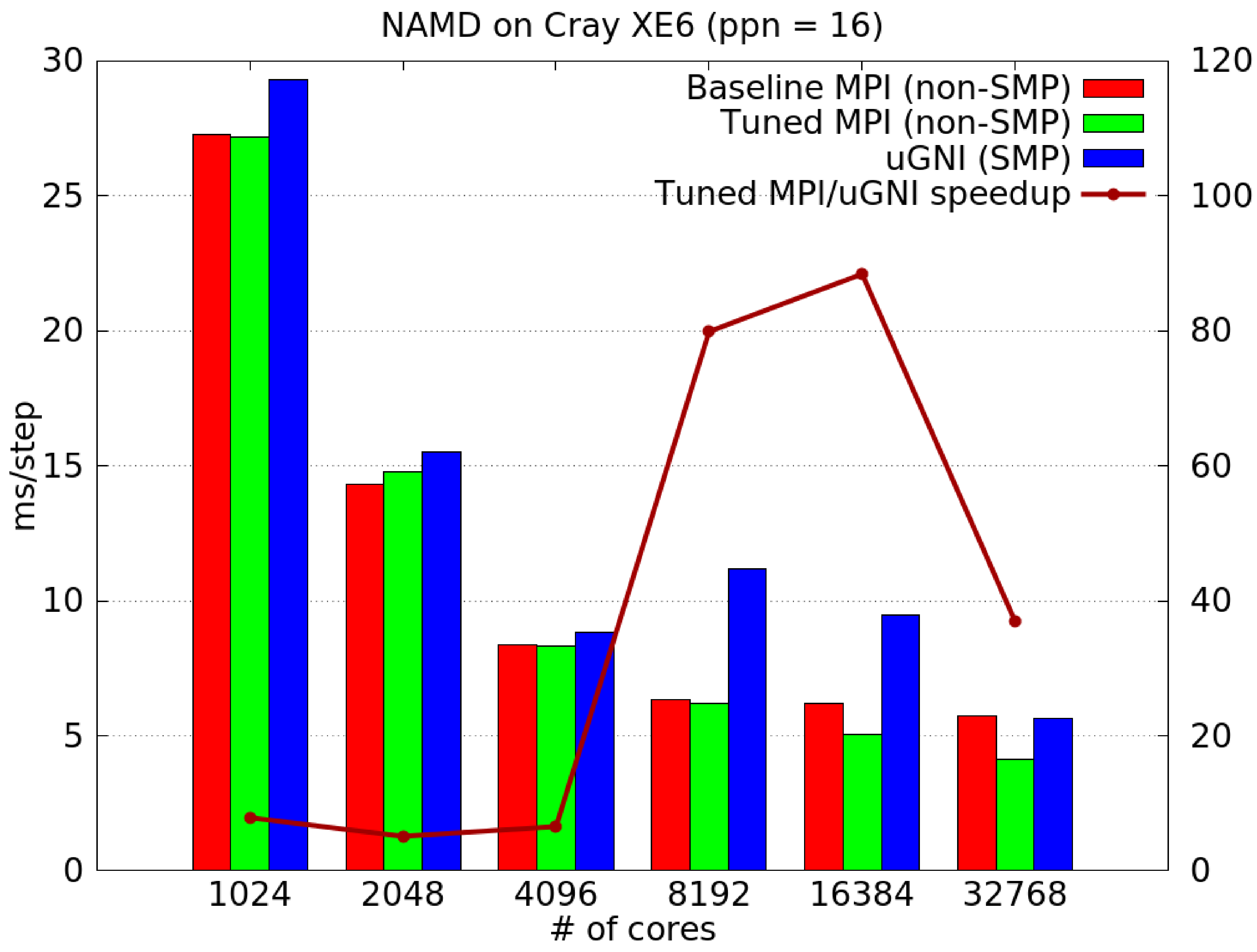
R. Gunter, D. Goodell, J. Dinan, P. Balaji

Nearest-neighbors results



R. Gunter, D. Goodell, J. Dinan, P. Balaji

Nearest-neighbors results

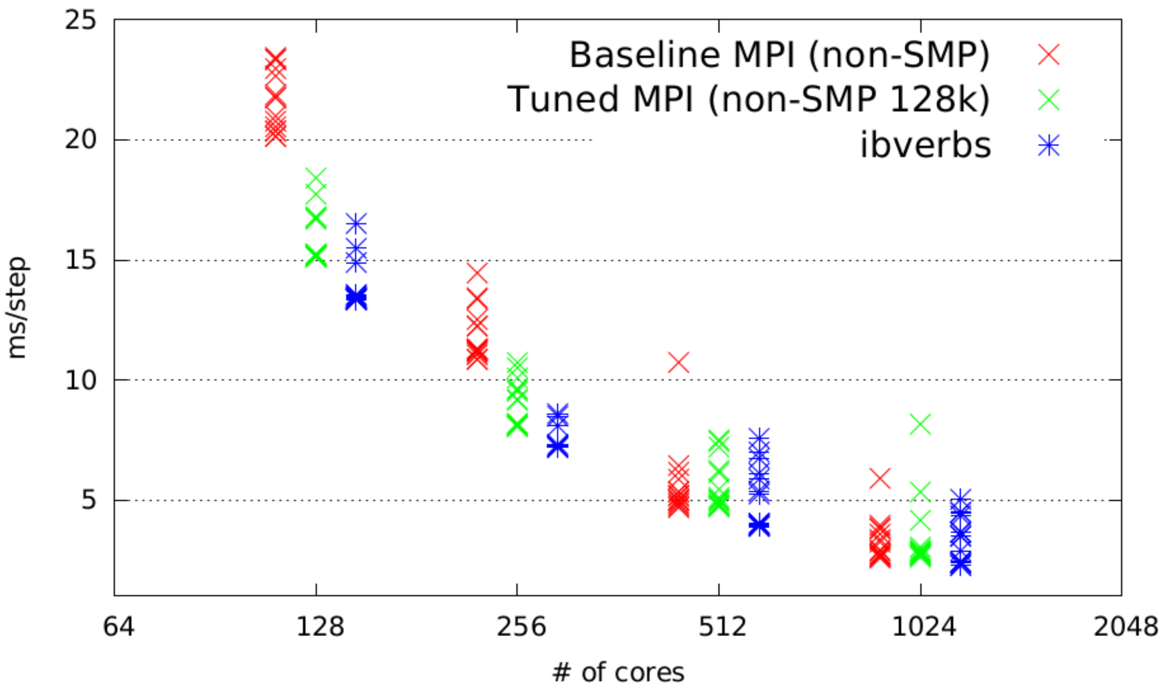


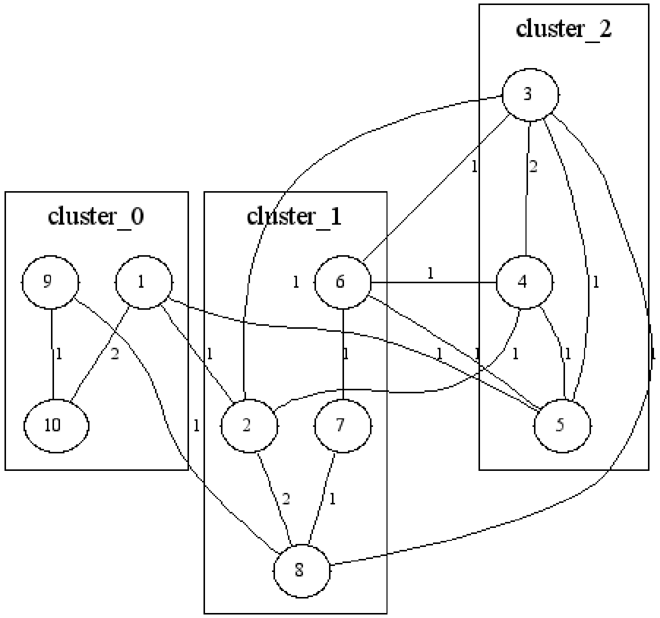
Lower is better

Higher is better for MPI

Future work.

- Fully integrate one-sided machine layer with charm.
- No convincing explanation yet for **ibverbs**/MVAPICH difference.
- **Hybrid benchmark** for per-communicator eager/rendezvous threshold on Cray.





R. Gunter, D. Goodell, J. Dinan, P. Balaji

Conclusions

- There's more to MPI slowdown than just “overhead”.
  - Mismatch of **MPI** with **Charm semantics** is a better story.
- Specific MPI-2 techniques per machine.
  - May not be portable, like eager/rendezvous threshold for Cray XE6 vs preposted `Irecv` for Blue Gene/Q.
- `Send`/`Recv` **machine layer** should be replaced with **one-sided** version once MPI-3 is broadly available.

R. Gunter, D. Goodell, J. Dinan, P. Balaji

Programming Models and Runtime Systems Group

Group Lead

- Pavan Balaji (scientist)

Current Staff Members

- James S. Dinan (postdoc)
- Antonio Pena (postdoc)
- Wesley Bland (postdoc)
- David J. Goodell (developer)
- Ralf Gunter (research associate)
- Yuqing Xiong (visiting researcher)

Upcoming Staff Members

- Huiwei Lu (postdoc)
- Yan Li (visiting postdoc)

Past Staff Members

- Darius T. Buntinas (developer)

Current and Past Students

- Xiuxia Zhang (Ph.D.)
- Chaoran Yang (Ph.D.)
- Min Si (Ph.D.)
- Huiwei Lu (Ph.D.)
- Yan Li (Ph.D.)
- David Ozog (Ph.D.)
- Palden Lama (Ph.D.)
- Xin Zhao (Ph.D.)
- Ziaul Haque Olive (Ph.D.)
- Md. Humayun Arafat (Ph.D.)
- Qingpeng Niu (Ph.D.)
- Li Rao (M.S.)
- Lukasz Wesolowski (Ph.D.)
- Feng Ji (Ph.D.)
- John Jenkins (Ph.D.)
- Ashwin Aji (Ph.D.)
- Shucai Xiao (Ph.D.)
- Sreeram Potluri (Ph.D.)
- Piotr Fidkowski (Ph.D.)
- James S. Dinan (Ph.D.)
- Gopalakrishnan Santhanaraman (Ph.D.)
- Ping Lai (Ph.D.)
- Rajesh Sudarsan (Ph.D.)
- Thomas Scogland (Ph.D.)
- Ganesh Narayanaswamy (M.S.)

Advisory Staff

- Rusty Lusk (retired)
- Marc Snir (director)
- Rajeev Thakur (deputy director)

External Collaborators (partial)

- Ahmad Afsahi, Queen's, Canada
- Andrew Chien, U. Chicago
- Wu-chun Feng, Virginia Tech
- William Gropp, UIUC
- Jue Hong, SIAT, Shenzhen
- Yutaka Ishikawa, U. Tokyo, Japan
- Laxmikant Kale, UIUC
- Guangming Tan, ICT, Beijing
- Yanjie Wei, SIAT, Shenzhen
- Qing Yi, UC Colorado Springs
- Yunquan Zhang, ISCAS, Beijing
- Xiaobo Zhou, UC Colorado Springs

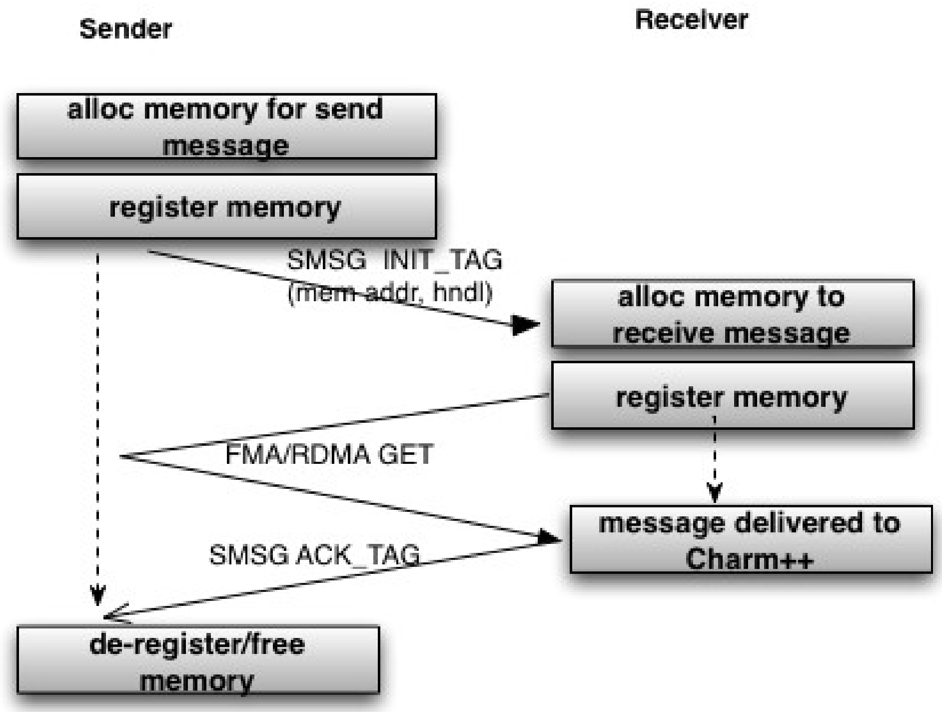
Acknowledgments

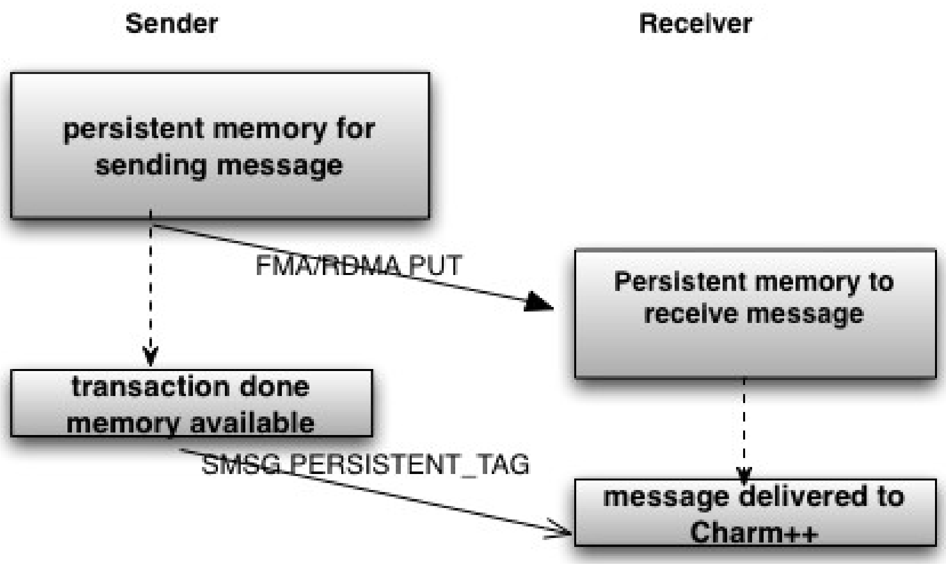
Funding Grant Providers

Infrastructure Providers

3) Send/Recv vs one-sided machine layer

- One-sided communication better suits charm's asynchrony.
  - `Send`/`Recv` puts too much burden on receiver.
  - All native machine layers take advantage of this.

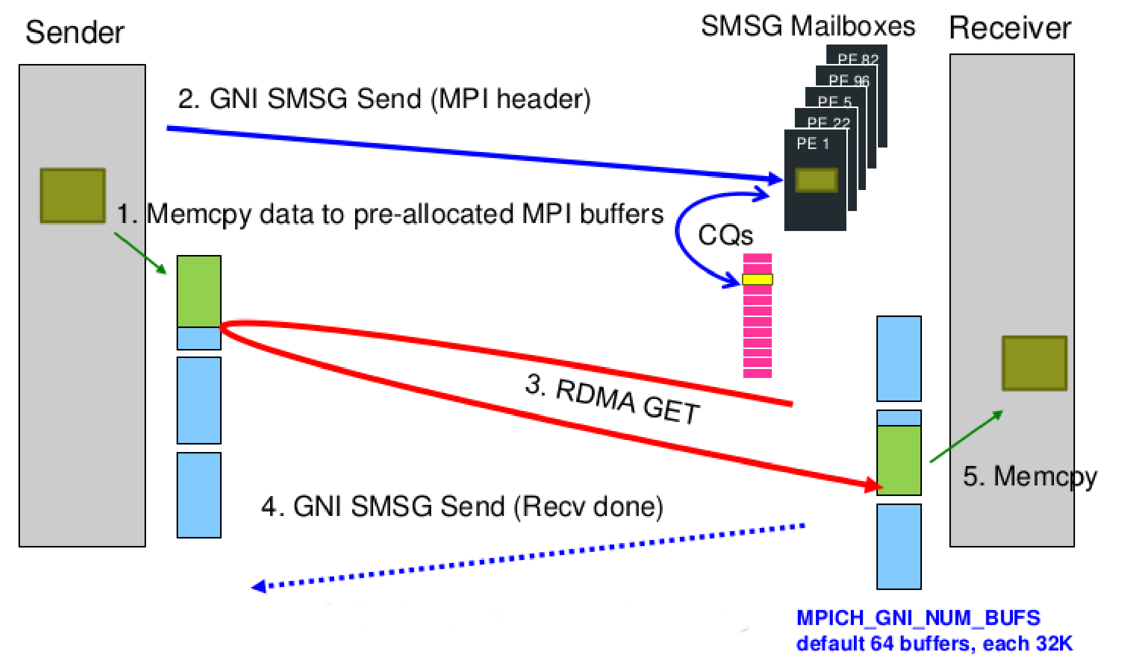


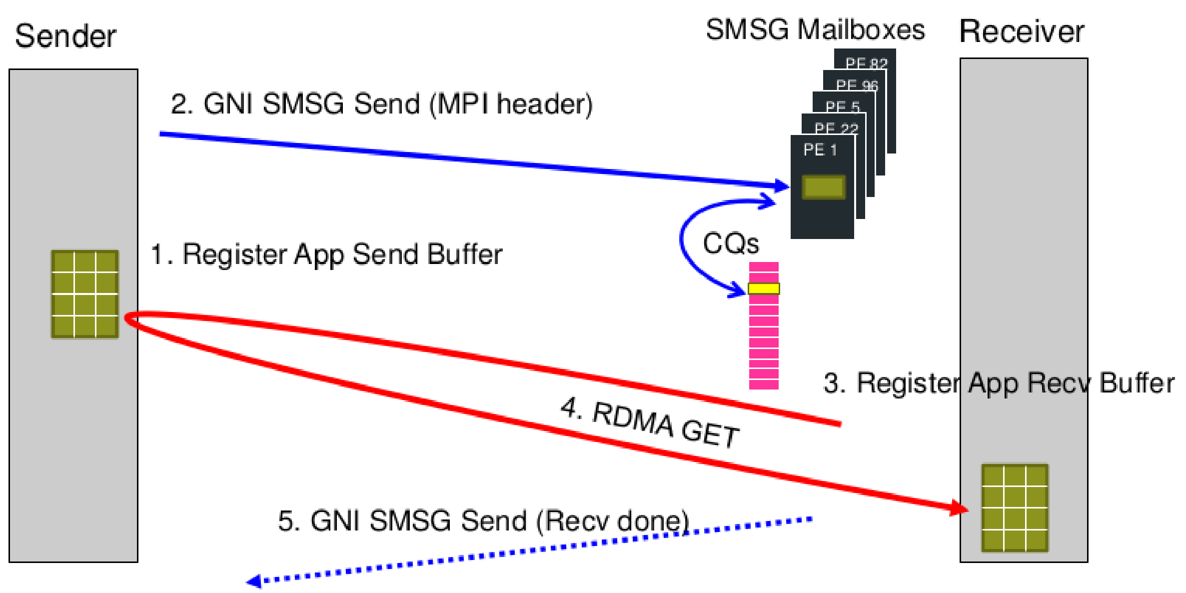


(Sun et al., IPDPS '12)

3) Send/Recv vs one-sided machine layer

- Vendor-supplied MPI implementations already do this internally.
- Two-sided matching semantics are just inappropriate.
  - "Tuned" for expected messages.
  - Blue Gene/Q suffers from serialization because of `Send`/`Recv`.





(Cray Inc., PRACE '12)