PGAS Models using an MPI Runtime: Design Alternatives and Performance Evaluation

Jeff Daily, Abhinav Vishnu, Bruce Palmer, Hubertus van Dam

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

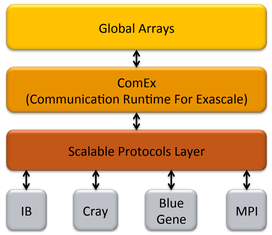
What is the best way to design a partitioned global address space (PGAS) communication subsystem given that MPI is our only choice?

Contributions:

- An in-depth analysis of design alternatives for a PGAS communication subsystem using MPI. We present a total of four design alternatives: MPI-RMA (RMA), MPI Two-Sided (TS), MPI Two-Sided with Multi-threading (MT), and MPI Two-Sided with Dynamic Process Management (DPM).
- A novel approach which uses a combination of two-sided semantics and an automatic, user-transparent split of MPI communicators to act as asynchronous progress ranks (PR) for designing scalable and fast communication protocols.
- Implementation of TS, MT, and PR approaches and their integration with ComEx - the communication runtime for Global Arrays. We perform an in-depth evaluation on a spectrum of software including communication benchmarks, application kernels, and a full application, NWChem[1].

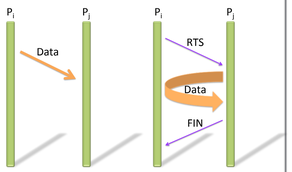
Communications Runtime for Exascale (ComEx):

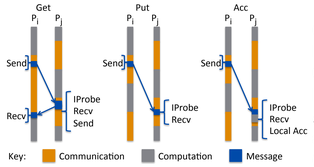
Software Ecosystem of Global Arrays and ComEx. Native implementations are available for Cray, IBM, and IB systems, but not for Kcomputer and Tianhe-1A.



Design: Exploration Space

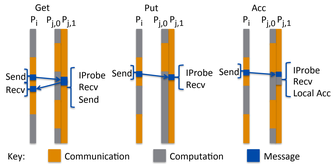
Typical Communication Protocols in MPI. The left side shows eager protocol, and the right side shows the read based protocol with RTS (request to send) and FIN (finish) messages. Variants such as Receiver initiated [2] and Write based [3] protocols exist in literature and practice.

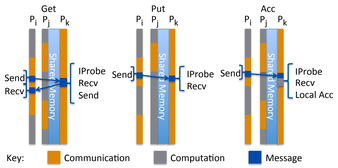




One-Sided Communication Protocols using Two- Sided Communication Protocols in MPI. Protocols for Get, Put and Accumulate are on the left, middle, and right, respectively. (Not Shown: MPI-RMA.)

One-Sided Communication Protocols using Two-Sided Communication Protocols in MPI with Multi-threading (MT). Protocols for Get, Put and Accumulate are on the left, middle, and right, respectively. Process P_i initiates a request to process P_j which is handled asynchronously by thread $P_{j,1}$. (Not Shown: Dynamic Process Management.)





One-Sided Communication Protocols using Two-sided Communication Protocols in MPI with Progress Ranks. Protocols for Get, Put, and Accumulate are on the left, middle, and right, respectively. Process P_i initiates a request to the progress rank P_k for the RMA targeting P_j . P_j and P_k reside within the same shared memory domain.

Solution: Progress Ranks for Shared Memory Domains

- User-transparent split of the world communicator
- One master rank per shared memory domain
- For performance reasons, uses two-sided semantics
- Asynchronous progress by master rank

Algorithm ComEx

Input: source address s, target address d, message size m, target process r

Procedure PUT(s,d,m,r)
- $r1 \leftarrow$ TranslateRankstoPR(r);
- **if** $m < \delta$ **then**
  - $buf \leftarrow$ InlineDataWithHeader(m);
  - Send$(buf \ldots r1)$;
- **else**
  - $buf \leftarrow$ PrepareHeader(m);
  - Send$(buf \ldots r1)$;
  - Send$(d \ldots s)$;
- **end**

Procedure GET(s,d,m,r)
- $r1 \leftarrow$ TranslateRankstoPR(r);
- $handle \leftarrow$ Irecv$(d \ldots r1)$;
- $buf \leftarrow$ PrepareHeader(m);
- Send$(buf \ldots r1)$;
- Wait$(handle)$;

Procedure ACC(s,d,m,r)
- $r1 \leftarrow$ TranslateRankstoPR(r);
- **if** $m < \delta$ **then**
  - $buf \leftarrow$ InlineDataWithHeader(m);
  - Send$(buf \ldots r1)$;
- **else**
  - $buf \leftarrow$ PrepareHeader(m);
  - Send$(buf \ldots r1)$;
  - Send$(d \ldots s)$;
- **end**

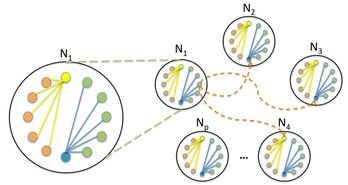
Procedure FADD(s,d,m,r)
- $r1 \leftarrow$ TranslateRankstoPR(r);
- $buf \leftarrow$ InlineDataWithHeader(m);
- Send$(buf \ldots r1)$;
- Recv$(d \ldots r1)$;

Algorithm ComEx PR Progress

Input: source address s, target address d, message size m, target process r

Procedure PROGRESS()
- **while** *running* **do**
  - $flag \leftarrow$ Iprobe();
  - **if** *flag* **then**
    - $header \leftarrow$ Recv();
    - **switch** $header.messageType$ **do**
      - **case** *PUT*
        - **if** IsDataInline$(header)$ **then**
          - CopyInlineData$(header)$;
        - **else**
          - Recv$(header.d \ldots header.r1)$;
        - **end**
        - break;
      - **end**
      - **case** *GET*
        - Send$(header.s \ldots header.r1)$;
        - break;
      - **end**
      - **case** *ACC*
        - LocalAcc$(header.d)$;
        - break;
      - **end**
      - **case** *FADD*
        - $counter \leftarrow$ LocalFAdd$(header.d)$;
        - Send$(counter \ldots header.r1)$;
      - **end**
    - **endsw**
  - **end**
- **end**

Translation of communication operations in the PR approach. Left: A typical node with with two PR ranks (blue and yellow circles). Blue PR is responsible for performing RMA operations on the memory hosted by green user-level processes. Right: A user-level process communicates with PR ranks on other nodes for RMA requests. On-node requests are performed using shared memory.



Evaluation of Approach

Motivating Applications

Algorithm Self-Consistent Field

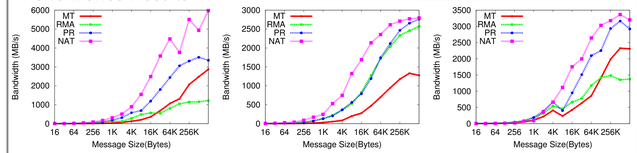
Procedure SCF(m,n)
- **Input**: my rank m, total tasks n
- **Data**: current task ID t, data indices i, data d
- $t \leftarrow m$;
- **while** $t < n$ **do**
  - $i \leftarrow$ CalculateIndices(t);
  - $d \leftarrow$ Get(i);
  - FockMatrix(t,i,d);
  - Accumulate(t,i,d);
  - $t \leftarrow$ NextTask();
- **end**

Algorithm Triangle Counting

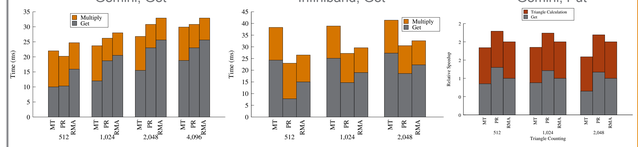
Procedure TC(p,m,v,e)
- **Input**: job size p, my rank m, graph G=(v,e)
- **Data**: vertex ID t
- $t \leftarrow m * v/p$;
- **while** $t < (m+1) * v/p$ **do**
  - $n \leftarrow$ NeighborList(t);
  - **for** $v_n \in n$ **do**
    - $nn \leftarrow$ NeighborList(v_n);
    - CalculateIfTriangleExists(t, v_n, nn);
  - **end**
- **end**

Procedure NeighborList(v)
- **Input**: vertex v
- $f \leftarrow$ Get(v);
- **return** f

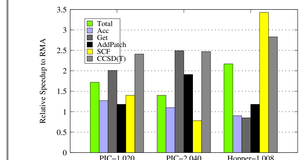
Performance Results



Gemini, Get | Infiniband, Get | Gemini, Put



Gemini, SpMV | Infiniband, SpMV | Infiniband, TC



NWChem CCSD(T) results for PR relative to RMA. MT did not finish execution for these runs.

- PR approach consistently outperforms other approaches
- 2.17x speed-up on 1008 processors over other MPI designs

For more information on the science you see here, please contact:

Jeff Daily
Pacific Northwest National Laboratory
P.O. Box 999, MS-IN: K7-90
Richland, WA 99352
(509) 372-6548
jeff.daily@pnnl.gov

References

[1] M. Valiev, E.J. Bylaska, N. Govind, K. Kowalski, T.P. Straatsma, H.J.J. Van Dam, D. Wang, J. Nieplocha, E. Apra, T.L. Windus, and W.A. de Jong. Nwchem: A comprehensive and scalable open-source solution for large scale molecular simulations. *Computer Physics Communications*, 181(9):1477 – 1489, 2010.

[2] Scott Pakin. Receiver-initiated message passing over rdma networks. In IPDPS, pages 1–12, 2008.

[3] Sayantan Sur, Hyun-Wook Jin, Lei Chai, and Dhabaleswar K. Panda. Rdma read based rendezvous protocol for mpi over infiniband: design alternatives and benefits. In PPOPP, pages 32–39, 2006.

Acknowledgments

This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE- AC02-05CH11231.

U.S. DEPARTMENT OF ENERGY

www.pnnl.gov