Optimizing Hypre Communication with Node Aware Parallelism

Gerald Collom

Amanda Bienz

UNM Dept. of Computer Science

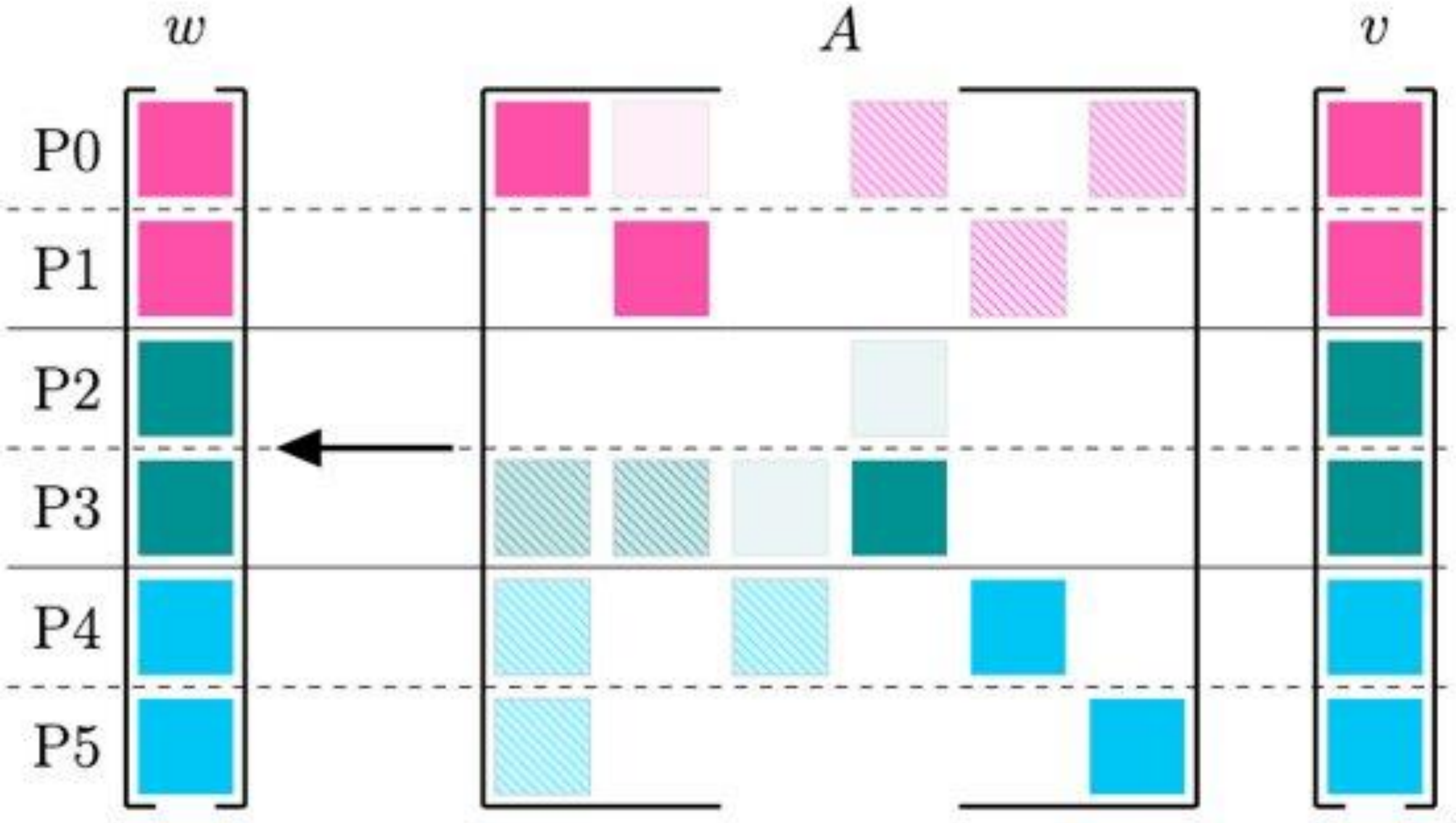
Ruipeng Li

Lawrence Livermore National Lab

Hypre Communication: Matrix Operations

- Hypre: Industry-leading multigrid solver for linear systems
- Communication in Hypre:
  - Parallel Sparse Matrix-Vector (SpMV) and Matrix-Matrix Multiplication
  - Irregular

https://github.com/hypre-space/hypre/



Hypre Code Example: Persistent Communication for SpMV

In file: src/parcsr_mv/par_csr_communication.c

In method: hypre_ParCSRPersistentCommHandleCreate

```c
          for (i = 0; i < num_recvs; ++i)
          {
             HYPRE_Int ip = hypre_ParCSRCommPkgRecvProc(comm_pkg, i);
             HYPRE_Int vec_start = hypre_ParCSRCommPkgRecvVecStart(comm_pkg, i);
             HYPRE_Int vec_len = hypre_ParCSRCommPkgRecvVecStart(comm_pkg, i + 1) - vec_start;
             hypre_MPI_Recv_init( (HYPRE_Complex *)recv_buff + vec_start, vec_len, HYPRE_MPI_COMPLEX,
                                  ip, 0, comm, requests + i );
          }
```

This loop initializes each receive of vec_len data starting at vec_start into recv_buff. The method hypre_MPI_Recv_init is a simple wrapper for MPI_Recv_init.

Neighborhood Collectives

Create graph from communication pattern:

```
int MPI_Dist_graph_create_adjacent(MPI_Comm comm_old, int indegree, const int sources[],
     const int sourceweights[], int outdegree, const int destinations[], const int destweights[],
        MPI_Info info, int reorder, MPI_Comm *comm_dist_graph)
```

Do single exchange based on graph:

```
int MPI_Neighbor_alltoallv(const void *sendbuf, const int sendcounts[],
     const int sdispls[], MPI_Datatype sendtype,
     void *recvbuf, const int recvcounts[],
     const int rdispls[], MPI_Datatype recvtype, MPI_Comm comm)
```

Persistent alternative:

```
int MPI_Neighbor_alltoallv_init(const void *sendbuf, const int sendcounts[],
     const int sdispls[], MPI_Datatype sendtype,
     void *recvbuf, const int recvcounts[],
     const int rdispls[], MPI_Datatype recvtype, MPI_Comm comm,
     MPI_Info info, MPI_Request *request)
```

Why?: replace several separate send/recv calls and provide communication metadata to MPI so MPI can optimize communication itself

Implementing Neighborhood Collectives in Hypre

When communication is initialized, create communication graph:

```
MPIX_Dist_graph_create_adjacent( comm, num_recvs, hypre_ParCSRCommPkgRecvProcs(comm_pkg),
                                 MPI_UNWEIGHTED, num_sends, hypre_ParCSRCommPkgSendProcs(comm_pkg),
                                 MPI_UNWEIGHTED, MPI_INFO_NULL, 0, &neighbor_comm);
```

Instead of calling Recv_init/Send_init in a loop:

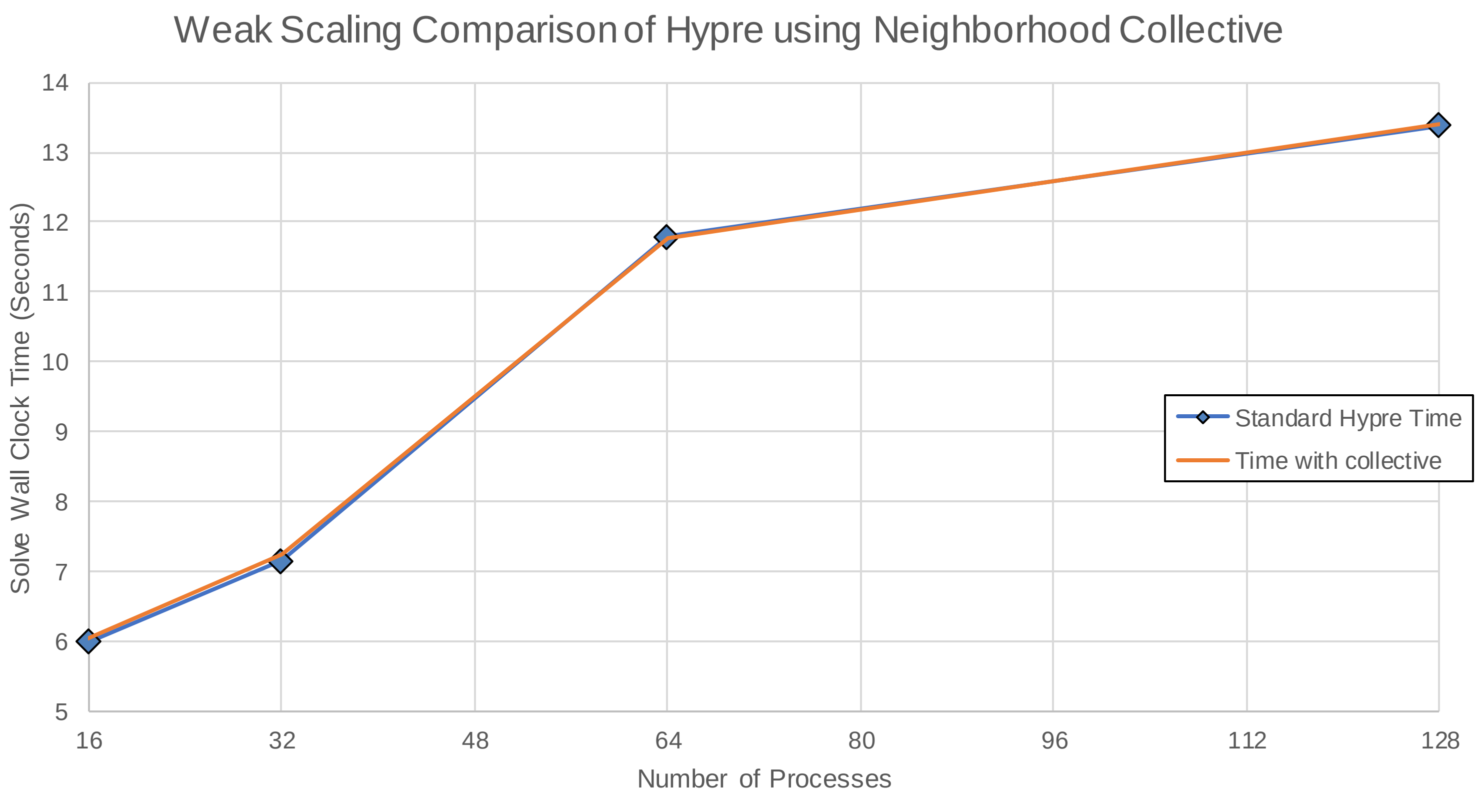
— store the size for each send/recv message

— After getting all send/recv sizes call neighborhood collective **once**

```
recv_sizes[i] = (hypre_ParCSRCommPkgRecvVecStart(comm_pkg, i + 1) -
                 hypre_ParCSRCommPkgRecvVecStart(comm_pkg, i));
```

```
MPIX_Neighbor_alltoallv_init( (HYPRE_Complex *)send_buff, send_sizes,
                              hypre_ParCSRCommPkgSendMapStarts(comm_pkg), HYPRE_MPI_COMPLEX,
                              (HYPRE_Complex *)recv_buff, recv_sizes,
                              hypre_ParCSRCommPkgRecvVecStarts(comm_pkg),
                              HYPRE_MPI_COMPLEX, neighbor_comm,
                              0, &Xrequest);
```

Performance Cost of Neighborhood Collective

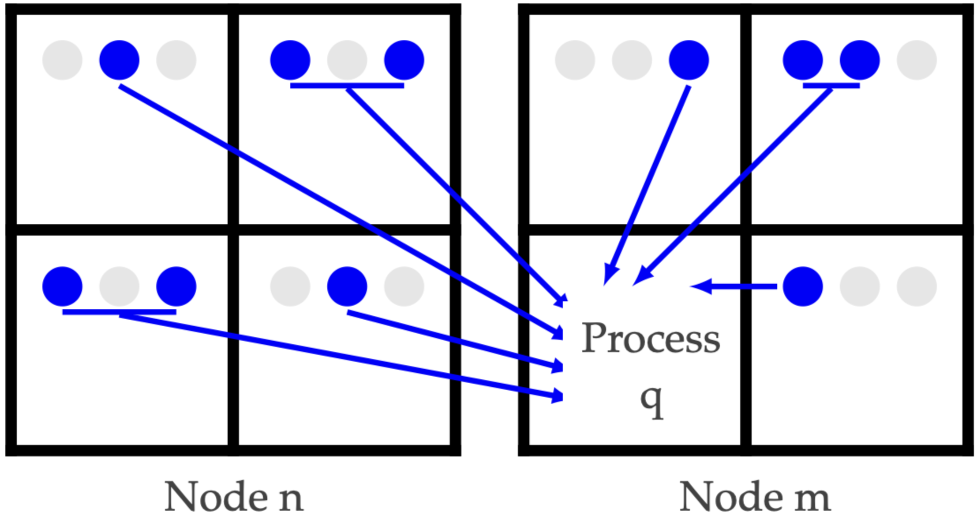


Data generated on Lassen with problem size of 100^3 per process using IJ driver. The neighborhood collective adds a ~1% overhead that vanishes by 64 processes.

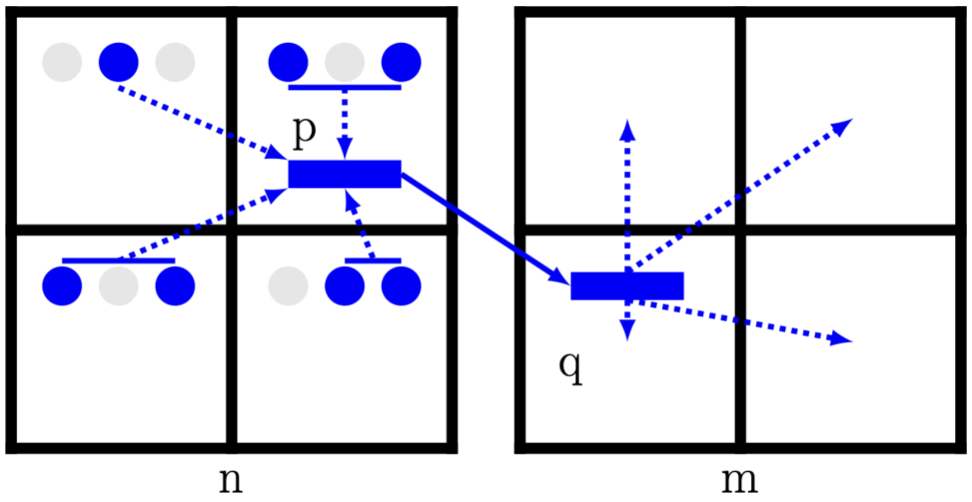
Takeaway: Neighborhood collective costs little overhead to Hypre but allows for optimization behind MPI

Node-Aware Optimization

Standard MPI: send data directly to process regardless of which node it is running on



Node-aware MPI: aggregate on-node before sending across nodes



Node-Aware Parallel SpMV[1]:

—Reduces number and size of costly internode messages

Implementation in Hypre (WIP):

Used library that provides optimized Neighbor_alltoallv created by Amanda Bienz, included when building Hypre

Extended interface requires additional changes to Hypre currently being debugged

Future and Related Work

- Persistent and Partitioned MPI in Comb:
  - —Partitioned MPI
  - —Comb: Regular halo exchange communication benchmark
  - —Initial performance comparison against Comb with standard MPI suggested no significant overhead to persistent MPI in Comb
  - —WIP: Partitioned MPI working in Comb for single thread case, debugging implementation of partitioned MPI + OpenMP
  - —Next steps: Performance analysis of partitioned MPI in Comb and GPU-triggered partitioned MPI, remove sync with CPU for communication
- Performance Comparison with middle-ground Neighbor_alltoallv optimization which has an interface identical to standard MPI
  - — Does not require additional code changes, only switching to neighborhood collective
  - — cannot benefit from the full node-aware SpMV optimization
- Inverse Neighbor_alltoallv interface, an operation required by AMG process. Currently creating an additional inverted graph

1. Amanda Bienz, William D. Gropp, & Luke N. Olson (2016). TAPSpMV: Topology-Aware Parallel Sparse Matrix Vector Multiplication. *CoRR, abs/1612.08060.*