Scalable Vector Collectives

Torsten Hoefler and Christian Siebert

On behalf of the Collectives working group

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Vector collectives

- Varying amounts of data from the processes
- Integral part of the standard since MPI-1.0:
  - MPI_GATHERV,
  - MPI_SCATTERV,
  - MPI_ALLGATHERV,
  - MPI_ALLTOALLV,
  - MPI_ALLTOALLW (added in MPI-2.0),
  - and MPI_REDUCE_SCATTER (no ‘v’ suffix)

Scalability Problems

- Need to specify counts and displacements for each process at each process
  - Memory need and time (p.p.) grows linearly in P

  ➔ Well-known scalability problem
- Problems are getting worse with ever increasing system sizes
- Memory needs will eventually prevent the use of current vector collectives (cf. “Exascale”)

Proposed Solution

- A straightforward solution is to distribute the global information amongst all processes
- Gatherv, Scatterv, Reduce_scatter, and Allgatherv have O(p) global information

  ➔ fixed variants need O(1) input (p.p.) ☺
- Unfortunately, Alltoall{v,w} need $O(p^2)$ input and can therefore not be be fixed… ☹

Are vector collectives used?

- Yes, even in libraries such as
  - ParMETIS (Parallel Graph Partitioning)
  - PBGL (Parallel Boost Graph Library)
  - PETSc (Parallel Algorithms to solve PDEs)
  - PSBLAS (Parallel Sparse Linear Algebra)
  - LibTopoMap (Topology Mapping Library)

➔ Several hundred applications use them!

Importance of Scalable Variants

- Vector collectives are used (irregular apps?)
- They work on today's machines
- It is primarily not about performance
- Current vector collectives will not work on future systems ≥ “Exascale”!
- Add-on: Scalable variants will perform more efficiently in sparse scenarios

Simple Proposal

- Four new distributed interfaces:
  - MPI_Gatherdv()
  - MPI_Scatterdv()
  - MPI_Allgatherdv()
  - MPI_Reduce_scatterdv()
- Each process specifies only the parameters for its local contribution

  e.g., int recvcounts[p] ➔ int recvcount

Proposal

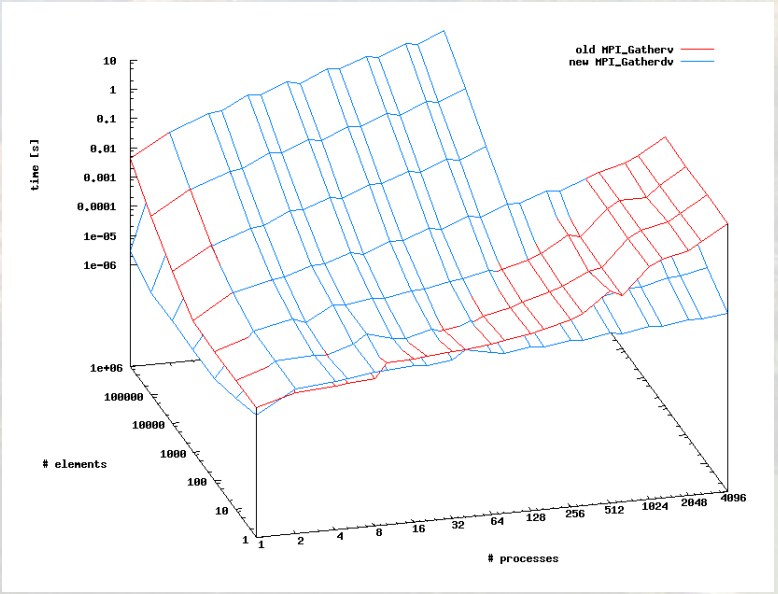
Ticket #264

("Scalable Variants of Vector Collectives")

See proposal document

Implementation

Implementation is done.



Measurements done on a Blue Gene/P

Optimized using “recursive doubling” ➔ very scalable

Solved Issues

Decision to omit displacements, because

- No consensus of how to specify consistently
- Not needed in any of the inspected libraries
- Slow-down on all DV-implementations

➔ Defined implicit displacements in **ascending** rank order

(e.g., elms of rank 0, elms of rank 1, elms of rank2)

Compatibility

If really needed, can be achieved as:

- MPI_Gatherdv(displacements)
- Evaluate elements in this order

  (or reorder using e.g., MPI_Sendrecv(COMM_SELF))

➔ Penalty only for those rare use cases

Summary

- Vector collectives are needed
- Current interface eventually stops working
- Distributing the arguments solves problems
- DV variants can be implemented efficiently
- Additional benefit for sparse scenarios

Straw-vote: yes: 13, no: 1, abstain: 1

Thanks!

Please review ticket #264!

Questions?