Generalized MPI Collectives – proposal to MPI Forum

Hans-Joachim Plum

Parallel Applications Engineer
Intel Corporation

March 2008

´v-operations´ with unknown message sizes

- In Gatherv / Scatterv / Allgatherv / Alltoallv,
  often the receive counts are not known a priori; user needs to do a preparing gather operation (inconvenient, time consuming)

- Example: sparse matrix operations

  Assume as sparse matrix in compact row storage, distributed by rows to processes. A row is attached to a mesh point. Every process knows what mesh points of other processes it touches through the matrix, but not which other processes touch what points of its part

  => Alltoallv required, with mutually unknown elementary portion sizes

  => must be done by Alltoall + Alltoallv in current setting

´v-operations´ with unknown message sizes

- Introduce 2 new MPI flags for unknown receive sizes / displacements

  MPI_SIZE_UNKNOWN
  entered in the receive counts array[0] which is then INOUT

  MPI_DISPL_CONSECUTIVE
  entered in the displacements array[0] which is then INOUT;
  important when sizes are unknown: place receive portions consecutively in rank order

- Additional argument RECV_MAX_COUNT for an upper bound of the #elements to be received overall = size (at least) of the recv buffer

- Probably sensible to introduce new interfaces for these extensions (more user friendly, standard case easier to tune if kept separate)

´v-operations´ with unknown message sizes

E.g. Gatherv

```
// all processes:

rcnt[0] = MPI_SIZE_UNKNOWN;
rdispl[0] = MPI_DISPL_CONSECUTIVE;

// could be released to “only root enters this” but more convenient to
// implement if entered globally;
// note: non-roots can use single int for rcnt, rdispl, no arrays needed
// note: all other entries of rcnt/rdispl are irrelevant; on return,
//       rcnt[], rdispl[] will have the actual values
// note: all combinations known / unknown rcnt with
// definite / consecutive rdispl allowed

// Root has to provide enough overall buffer space
RECV_MAX_COUNT = <certainly enough to hold all messages>

MPI_XGatherv(SENDBUF, SEND_COUNT, SEND_TYPE,
             RECVBUF, RECV_MAX_COUNT, rcnt, rdispl, RECV_TYPE,
             root, COMM)
```

Implementation concept; draft prototype timings

Prototype implementation with a non trivial pattern (other than “all non roots just send”) done; compared against what has to be done momentarily if MPI_SIZE_UNKNOWN (MPI_Gather + MPI_Gatherv), with the analogous implementation (same pattern) used for the standard case

All cases use 16 processes on 16 different nodes, IB, non uniform message sizes, root=rank 1 (not 0);
Timings in us, gain = gain XGatherv vs. Gather+Gatherv

	KNOWN SIZES	MPI_SIZE_UNKNOWN / MPI_DISPL_CONSECUTIVE		
Avrg size	MPI_Gatherv std	MPI_Gather + MPI_Gatherv std.	MPI_XGatherv	gain %
41	6,8	10,1	6,9	31,7
83	7,2	10,6	7,4	30,2
169	8,2	11,6	8,4	27,6
340	9,0	12,7	9,3	26,8
682	14,4	17,9	14,5	19,0
1366	20,0	22,7	20,3	10,6
2733	28,6	31,8	29,8	6,3
5469	49,1	50,8	49,2	3,1