German Research School
for Simulation Sciences

Scalable Irregular Collectives

Christian Siebert

16th September 2010

JÜLICH FORSCHUNGSZENTRUM

RWTH AACHEN UNIVERSITY

German Research School
for Simulation Sciences

Known Problem

MPI_GATHERV(sendbuf, sendcount, sendtype, recvbuf, recvcounts, displs,
recvtype, root, comm)

...

IN	recvcounts	non-negative integer array (of length group size) containing the number of elements that are received from each process (significant only at root)
IN	displs	integer array (of length group size). Entry i specifies the displacement relative to recvbuf at which to place the incoming data from process i (significant only at root)

...

(MPI standard version 2.2 page 141)

Known Problem

Gatherv example *:

100 million processes * 2 integer arrays * 4 byte per integer
= tremendous 800 MB just for parameters!

- memory bottleneck will come
  ($\rightarrow$ listen to talks of Bill G. & Jack D.)
- performance penalty is already there
  ($\rightarrow$ Balaji et al.: *MPI on a Million Processors*)

Conclusion

The MPI interface for irregular collectives is not scalable!

* (similar for Scatterv, Allgatherv, Alltoallv, . . .)

Solution needed!

New irregular collective functions for MPI-3.0?

Solution needed!

New irregular collective functions for MPI-3.0?

MPI alphabet for (prefix and) postfix:
. . . v (irregular), w (alltoallw), **x** (unused), . . .

Mnemonic trick?

The e**X**tremely scalable interface for irregular collectives.

Idea

$O(p)$ global pieces of information stored per process:

rank 0: MPI_Gatherv(. . . , counts=[1,2,3,4], displs=[3,4,0,6], . . .)
rank 1: MPI_Gatherv(. . . , counts=[1,2,3,4], displs=[3,4,0,6], . . .)
rank 2: MPI_Gatherv(. . . , counts=[1,2,3,4], displs=[3,4,0,6], . . .)
rank 3: MPI_Gatherv(. . . , counts=[1,2,3,4], displs=[3,4,0,6], . . .)

Note: Arrays are significant only at root.

Idea

$O(p)$ global pieces of information stored per process:

rank 0: MPI_Gatherv(..., counts=[1,2,3,4], displs=[3,4,0,6], ...)
rank 1: MPI_Gatherv(..., counts=[1,2,3,4], displs=[3,4,0,6], ...)
rank 2: MPI_Gatherv(..., counts=[1,2,3,4], displs=[3,4,0,6], ...)
rank 3: MPI_Gatherv(..., counts=[1,2,3,4], displs=[3,4,0,6], ...)

Note: Arrays are significant only at root.

Better: Distribute the non-scalable parameter arrays!

Idea

$O(p)$ global pieces of information stored per process:

rank 0: MPI_Gatherv(. . . , counts=[1,2,3,4], displs=[3,4,0,6], . . .)
rank 1: MPI_Gatherv(. . . , counts=[1,2,3,4], displs=[3,4,0,6], . . .)
rank 2: MPI_Gatherv(. . . , counts=[1,2,3,4], displs=[3,4,0,6], . . .)
rank 3: MPI_Gatherv(. . . , counts=[1,2,3,4], displs=[3,4,0,6], . . .)

Note: Arrays are significant only at root.

Better: Distribute the non-scalable parameter arrays!

rank 0: MPI_Gatherx(. . . , count=1, displ=3, . . .)
rank 1: MPI_Gatherx(. . . , count=2, displ=4, . . .)
rank 2: MPI_Gatherx(. . . , count=3, displ=0, . . .)
rank 3: MPI_Gatherx(. . . , count=4, displ=6, . . .)

Same amount of global information, but only $O(1)$ per process!

Trivial Implementation

MPI_Scatterx(sb, sendcount, displ, st, rb, rc, rt, root, comm):

```
MPI_Comm_size(comm, &p);
sendcounts = malloc(sizeof(int)*p);
displs = malloc(sizeof(int)*p);
MPI_Allgather(&sendcount, 1, MPI_INT, sendcounts, 1, MPI_INT, comm);
MPI_Allgather(&displ, 1, MPI_INT, displs, 1, MPI_INT, comm);
MPI_Scatterv(sb, sendcounts, displs, st, rb, rc, rt, root, comm);
free(displs);
free(sendcounts);
```

In words

Simply reconstruct the vectors and call the corresponding v-collective.

Optimizing x-Collectives

Much better implementations possible!

- Only $O(\log p)$ parameter handling with reductions!
  (e.g., Allreduce, Exscan, . . .)
- Good algorithms only need $O(1)$ entries p.p.!
  (e.g. recursive doubling, linear pipeline, . . .)

German Research School
for Simulation Sciences

Optimizing x-Collectives

Much better implementations possible!

- Only $O(\log p)$ parameter handling with reductions!
  (e.g., Allreduce, Exscan, ...)
- Good algorithms only need $O(1)$ entries p.p.!
  (e.g. recursive doubling, linear pipeline, ...)

Proposal summary

Enables scalable interface for Gatherv, Scatterv, Allgatherv, and Reduce_Scatterv with $O(1)$ space per process.

Optimizing x-Collectives

Much better implementations possible!

- Only $O(\log p)$ parameter handling with reductions!
  (e.g., Allreduce, Exscan, ...)
- Good algorithms only need $O(1)$ entries p.p.!
  (e.g. recursive doubling, linear pipeline, ...)

Proposal summary

Enables scalable interface for Gatherv, Scatterv, Allgatherv, and Reduce_Scatterv with $O(1)$ space per process.

Unfortunately, not applicable to Alltoallv or Alltoallw.
(Because of $O(p^2)$ global information)

German Research School
for Simulation Sciences

Thank you.

JÜLICH FORSCHUNGSZENTRUM RWTH AACHEN UNIVERSITY