Introduction to Parallel Programming with MPI

Dr. Alireza Ghasemi, Dr. Georg Hager

Erlangen National High Performance Computing Center

Nonblocking Collectives

Nonblocking Collectives in MPI

Similar to blocking collectives: nonblocking collective calls including all ranks of a communicator
All ranks must call the function!

- Nonblocking variants (since MPI 3.0):
  buffer can be used after completion (`MPI_Wait*`/`MPI_Test*`)
- Local: not synchronization
- Multiple outstanding collective communications on same communicator
- Cannot interfere with point-to-point communication
  - Completely separate modes of operation!
- Cannot interfere with blocking collective communication
  - Such interference was allowed in point-to-point communication

Collectives in MPI

- Rules for all collectives
  - Data type matching
  - No tags
  - Count must be exact, i.e., there is only one message length, buffer must be large enough
- Types:
  - Synchronization (barrier)
  - Data movement (broadcast, scatter, gather, all to all)
  - Collective computation (reduction, scan)
  - Combinations of data movement and computation (reduction + broadcast)
- General assumption: MPI does a better job at collectives than you trying to emulate them with point-to-point calls

Barrier

- Nonblocking synchronization

  ```
  MPI_Ibarrier(MPI_Comm comm,MPI_Request *request)
  ```

- Must be followed by an `MPI_Wait`
- Calling process enters the barrier, no synchronization happens
- Synchronization may happen asynchronously
- Overlapping synchronization with work: reducing idle time
- Comarison:

  1) `MPI_Ibarrier` $\longrightarrow$ Work $\longrightarrow$ `MPI_Wait`

  2) Work $\longrightarrow$ `MPI_Barrier`

  idle times before and after Work differ on each process and their sum!

Collectives: Blocking vs. Nonblocking

- Broadcast:
  - `MPI_Bcast(buf,count,datatype,root,comm);`
  - `MPI_Ibcast(buf,count,datatype,root,comm,request);`
- Scatter:
  - `MPI_Scatter(sendbuf,sendcount,sendtype,recvbuf,recvcount, recvtype,root,comm);`
  - `MPI_Iscatter(sendbuf,sendcount,sendtype,recvbuf,recvcount, recvtype,root,comm,request);`
- Gather:
  - `MPI_Gather(sendbuf,sendcount,sendtype,recvbuf,recvcount, recvtype,root,comm);`
  - `MPI_Igather(sendbuf,sendcount,sendtype,recvbuf,recvcount, recvtype,root,comm,request);`

Collectives: Blocking vs. Nonblocking

- Similarly many blocking collective calls have nonblocking analogues:
  - `MPI_Iallgather`, `MPI_Ialltoall`, `MPI_Ireduce`,…

Remarks for `MPI_Ireduce`:

- Both send and receive buffers can be used only after completion!
- Similar to nonblocking point-to-point calls, `MPI_Wait*` or `MPI_Test*` must be used to halt or examine for the completion a request, respectively
- `root` (if available) and `comm` must be the same on all processes
- Type signature of send and receive variables must match

Nonblocking reduction on all ranks

```
MPI_Iallreduce(sendbuf,recvbuf,count,datatype,op,
                   comm,request);
```

- No **`root`**
- **`sendbuf,recvbuf`** can be reued after completion: requires **`MPI_Wait`** or **`MPI_Test`**
- **`Recvbuf`** is significant on all processes

Quiz:

1) Nonblocking collective are useful when there exist multiple collective calls on the same communicator so overlapping different collective calls.
   a) Correct b) Incorrect

2) Nonblocking collectives can interfere with blocking collectives.
   a) Correct b) Incorrect