Persistent Operations

- Many operations are repetitive
- Same parameters used many time
  - » Do i=1,n
    call MPI_Isend(…)
    call MPI_Irecv(…)
    call MPI_Waitall(…)
- MPI provides *persistent* operations to make this more efficient
  - » Reduce error checking of args (needed only once)
  - » Implementation may be able to make special provision for repetitive operation (though none do to date)
  - » All persistent operations are nonblocking

1

Using Persistent Operations

- Replace
  MPI_Isend(buf, count, datatype, tag, dest, comm, &request)
  with
  MPI_Send_init(buf, count, datatype, tag, dest, comm, &request)
  MPI_Start(request)
- MPI_Irecv with MPI_Recv_init, MPI_Irsend with MPI_Rsend_init, etc.
- Wait/test requests just like other nonblocking requests
- Free requests when done with MPI_Request_free

2

Example: Sparse Matrix-Vector Product

- Many interative methods require matrix-vector products
- Same operation (with same arguments) performed many times (vector updated in place)
- Divide sparse matrix into groups of rows by process: e.g., rows 1-10 on process 0, 11-20 on process 1. Use same division for vector.
- To perform matrix-vector product, get elements of vector on different processes with Irecv/Isend/Waitall

3

Changing MPI Nonblocking to MPI Persistent

- Do i=1,n
```
   ! Exchange vector information
   Call MPI_Isend( … )
   Call MPI_Irecv( … )
   Call MPI_Waitall( … )
```
- Replace with
```
Call MPI_Send_init( … )
Call MPI_Recv_init( … )
Do i=1,n
   Call MPI_Startall( 2, requests, ierr )
   Call MPI_Waitall( … )
EndDo
Call MPI_Request_free( request(1), ierr )
Call MPI_Request_free( request(2), ierr )
```

4