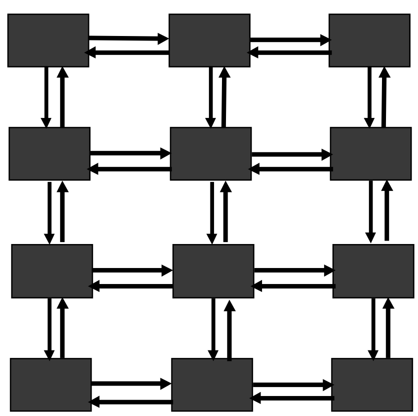
Unexpected Hot Spots

- Arises even in common grid exchange patterns
- Message passing illustrates problems present even in shared memory
  - » Blocking operations may cause unavoidable stalls

1

Mesh Exchange

- Exchange data on a mesh



2

Sample Code

- Do i=1,n_neighbors

```
Do i=1,n_neighbors
  Call MPI_Send(edge, len, MPI_REAL, nbr(i), tag,
                comm, ierr)
Enddo
Do i=1,n_neighbors
  Call MPI_Recv(edge,len,MPI_REAL,nbr(i),tag,
                comm,status,ierr)
Enddo
```

3

Deadlocks!

- All of the sends may block, waiting for a matching receive (will for large enough messages)
- The variation of

```
if (has down nbr)
   Call MPI_Send( … down … )
if (has up nbr)
   Call MPI_Recv( … up … )
…
```

sequentializes (all except the bottom process blocks)

4

Sequentialization

Start Send	Start Send	Start Send	Start Send	Start Send	Start Send	Send	Recv
					Send	Recv	
				Send	Recv		
			Send	Recv			
		Send	Recv				
	Send	Recv					
Send	Recv						

5

Fix 1: Use Irecv

- Do i=1,n_neighbors
  Call MPI_Irecv(edge,len,MPI_REAL,nbr(i),tag,
  comm,requests(i),ierr)
  Enddo
  Do i=1,n_neighbors
  Call MPI_Send(edge, len, MPI_REAL, nbr(i), tag,
  comm, ierr)
  Enddo
  Call MPI_Waitall(n_neighbors, requests, statuses, ierr)
- Does not perform well in practice. Why?

6

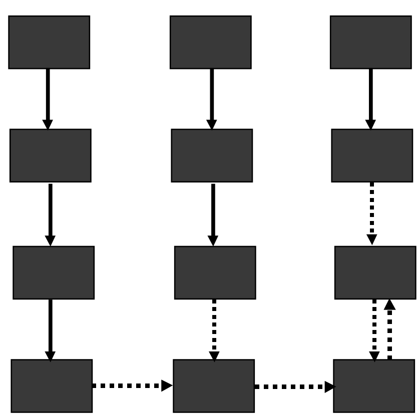
Timing Model

- Sends interleave
- Sends block (data larger than buffering will allow)
- Sends control timing
- Receives do not interfere with Sends
- Exchange can be done in 4 steps (down, right, up, left)

7

Mesh Exchange - Step 1

- Exchange data on a mesh

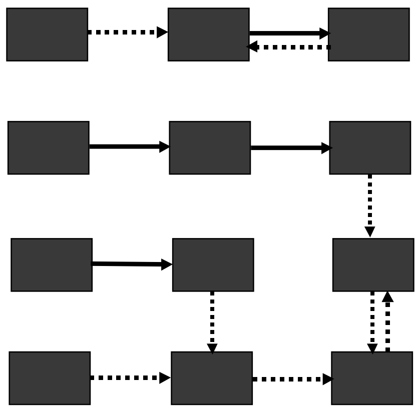


8

4

Mesh Exchange - Step 2

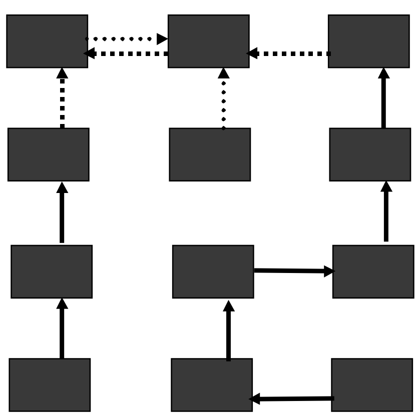
- Exchange data on a mesh



9

Mesh Exchange - Step 3

- Exchange data on a mesh

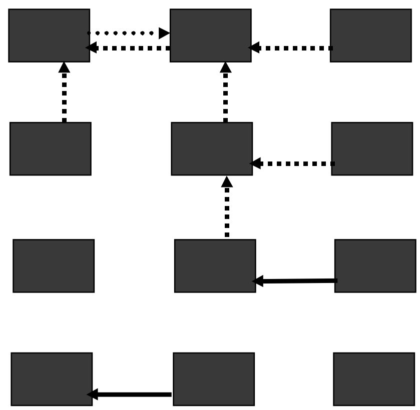


10

5

Mesh Exchange - Step 4

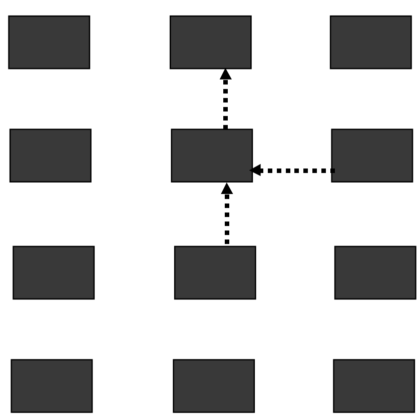
- Exchange data on a mesh



11

Mesh Exchange - Step 5

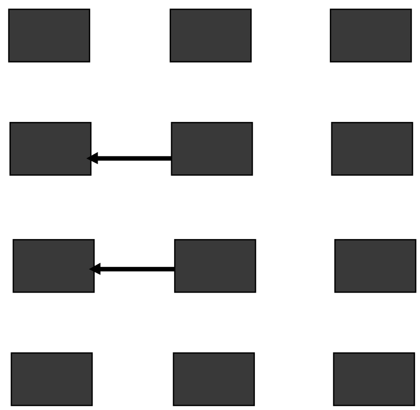
- Exchange data on a mesh



12

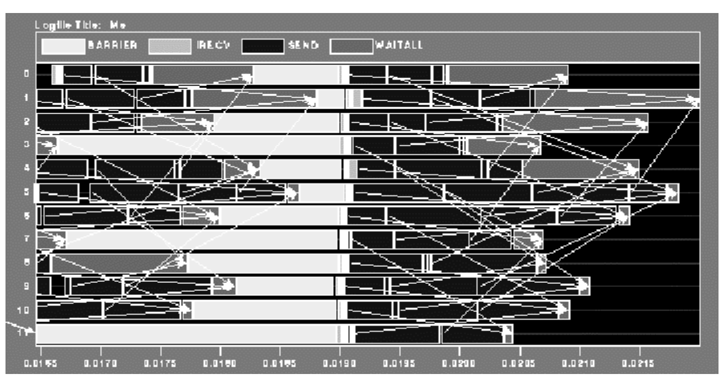
Mesh Exchange - Step 6

- Exchange data on a mesh



13

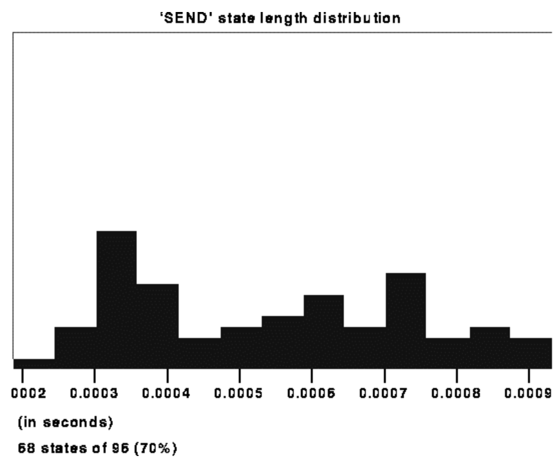
Timeline from IBM SP



- Note that process 1 finishes last, as predicted

14

Distribution of Sends



Why Six Steps?

- Ordering of Sends introduces delays when there is contention at the receiver
- Takes roughly twice as long as it should
- Bandwidth is being wasted
- Same thing would happen if using memcpy and shared memory

8

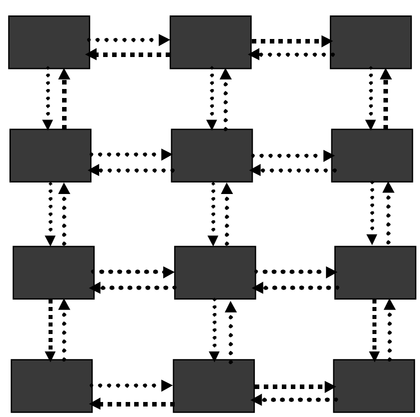
Fix 2: Use Isend and Irecv

- ```
  Do i=1,n_neighbors
    Call MPI_Irecv(edge,len,MPI_REAL,nbr(i),tag,
                   comm,request(i),ierr)
  Enddo
  Do i=1,n_neighbors
    Call MPI_Isend(edge, len, MPI_REAL, nbr(i), tag,
                   comm, request(n_neighbors+i), ierr)
  Enddo
  Call MPI_Waitall(2*n_neighbors, request, statuses,
                   ierr)
  ```
- (We'll see later how to do even better than this)

17

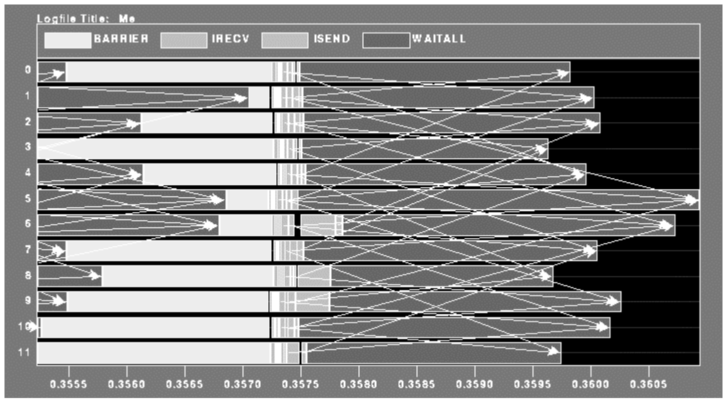
Mesh Exchange - Steps 1-4

- Four interleaved steps



18
  ```

Timeline from IBM SP



Note processes 5 and 6 are the only interior processors; these perform more communication than the other processors

19

Lesson: Defer Synchronization

- Send-receive accomplishes two things:
  - » Data transfer
  - » Synchronization
- In many cases, there is more synchronization than required
- Use nonblocking operations and MPI_Waitall to defer synchronization

20

MPI-2 Solution

- MPI-2 introduces one-sided operations
  - » Put, Get, Accumulate
  - » Separate data transfer from synchronization
- These are all nonblocking (blocking implies some synchronization)

One-sided Code

- Do i=1,n_neighbors

```
    Call MPI_Get(edge,len,MPI_REAL,nbr(i),
              edgedisp(i),len,MPI_REAL,win,ierr)
Enddo
Call MPI_Win_fence( 0, win, ierr )
```

- MPI_Put may be preferable on some platforms
- Can avoid global synchronization (MPI_Win_fence) with MPI_Win_start/post/complete/wait
- Use MPI_Accumulate to move and add

Exercise: Deferred Synchronization

- Write code that has each processor send to all of the other processors. Use MPI_Irecv and MPI_Send. Compare:
  - » All processors send to process 0, then process 1, etc., in that order
  - » Each process sends to process (myrank+1), then (myrank+2), etc.
  - » Compare with the MPI routine MPI_Alltoall
- If you have access to a shared-memory system, try the same thing using direct shared-memory copies (memcpy).