Parallel Optimization for HPC
Hands-on Lab

Todd Evans
rtevans@tacc.utexas.edu

June 22, 2016

Setup

1. Login to Stampede

   ```
   ssh username@stampede.tacc.utexas.edu
   ```

2. Untar the files

   ```
   cd
   tar xvf ~train00/SS_ASC2016_par_lab.tar
   ```

3. Change directories

   ```
   cd SS_ASC2016_par_lab
   ls
   ```

A Simple 2D Problem

- No particular physical process
- Similar structure to many codes

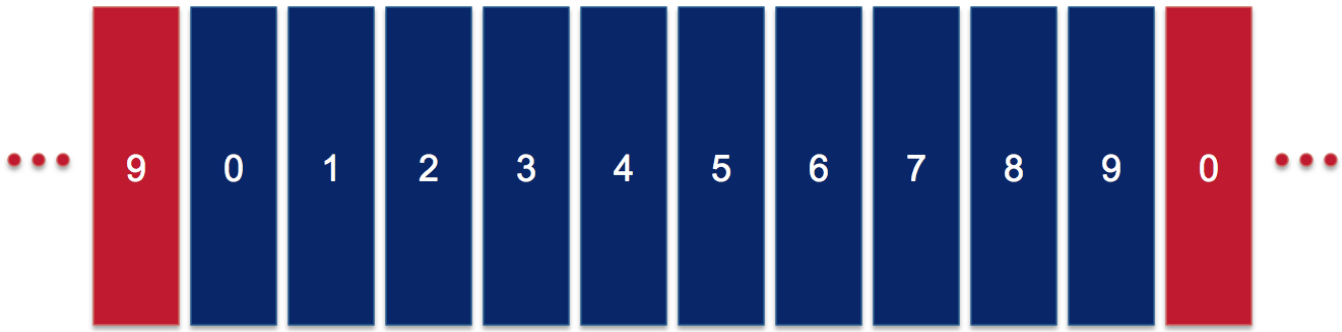
1. Calculate the derivative of f
2. Update f
3. Update neighbor boundary values
4. Start again

$$f'(x, y, t) = \frac{df(x, y, t)}{dx} = \frac{f(x + \triangle x, y, t) - f(x - \triangle x, y, t}{2 \triangle x}$$

$$f(x, y, t + 1) = f(x, y, t) + \epsilon f'(x, y, t)$$

The Partitioning Scheme

- We will use a 1D partitioning scheme
- Lines 35-48 (C) and 38-50 (F90) define the 1D topology
- Periodic boundary conditions are embedded in the topology
- We can use *left* and *right* for the MPI exchange



Data Exchange Optimization

- Focus on the main loop on line 69
- Several ways to optimize the data exchange
- Think back to the concepts presented and find at least one way to improve the execution time
- Make any changes you need to improve the current performance (keep a copy of the original!)

Getting Started

1. Choose the C or Fortran version
2. Make a copy that you'll modify

```
cp ./exchange_1d.c ./exchange_opt.c
cp ./exchange_1d.f90 ./exchange_opt.f90
```

3. Compile the current version of the code

```
mpicc ./exchange_1d.c -o original
mpif90 ./exchange_1d.f90 -o original
```

4. Start an interactive session in Stampede

```
idev
```

5. Run the code using 10 cores and record the timings

```
ibrun -np 10 ./original
```

6. Try to improve that time by modifying exchange_opt.c or exchange_opt.f90. (Hint: Is it more efficient to send many small messages are a few large messages?)

Proposed Solution

Original Code

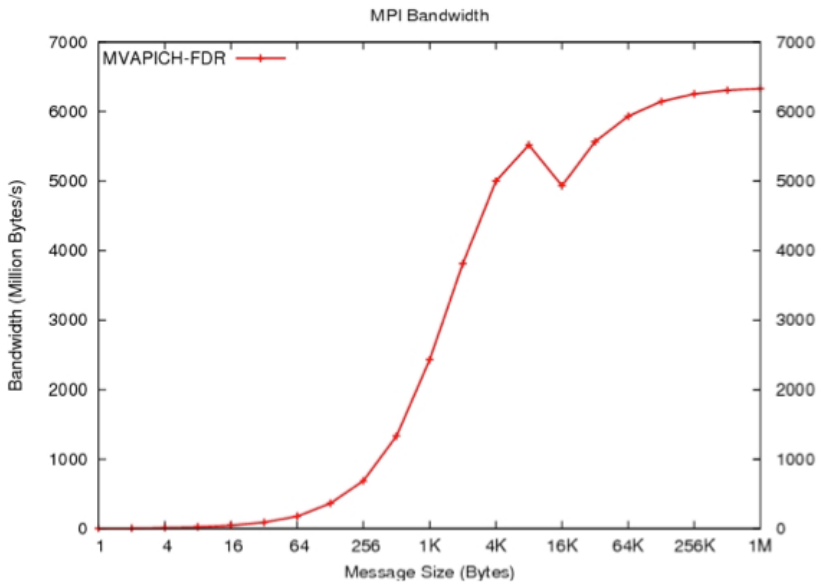
Send to right, receive from left

```
1 for( j = 0; j < NY; j++ ){
2     sendBuf[0] = f[NX][j];
3     MPI_Irecv( recvBuf, 1, MPI_DOUBLE, left,  ...);
4     MPI_Send( sendBuf, 1, MPI_DOUBLE, right, ...);
5     MPI_Wait( &request, &status );
6     f[0][j] = recvBuf[0];
7 }
```

Send to left, receive from right

```
1 for( j = 0; j < NY; j++ ){
2     sendBuf[0] = f[1][j];
3     MPI_Irecv( recvBuf, 1, MPI_DOUBLE, right, ...);
4     MPI_Send(  sendBuf, 1, MPI_DOUBLE, left, ...);
5     MPI_Wait( &request, &status );
6     f[NX+1][j] = recvBuf[0];
7 }
```

- One message for each data item to exchange in each direction
- Message size is 8 Bytes



Optimized Code v1

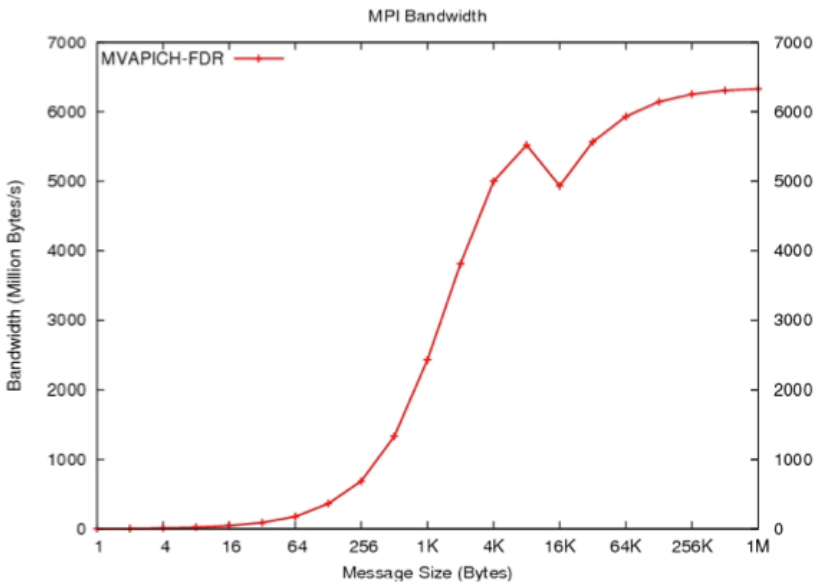
Send to right, receive from left

```
1  for( j = 0; j < NY; j++ ) sendBuf[j] = f[NX][j];
2
3  MPI_Irecv(recvBuf,NY,MPI_DOUBLE,left,...);
4  MPI_Send(sendBuf,NY,MPI_DOUBLE,right,...);
5  MPI_Wait(&request, &status );
6
7  for( j = 0; j < NY; j++ ) f[0][j] = recvBuf[j];
```

Send to left, receive from right

```
1  for( j = 0; j < NY; j++ ) sendBuf[j] = f[1][j];
2
3  MPI_Irecv(recvBuf,NY,MPI_DOUBLE,right,...);
4  MPI_Send(sendBuf,NY,MPI_DOUBLE,left,...);
5  MPI_Wait(&request, &status );
6
7  for(j=0;j<NY;j++) f[NX+1][j]=recvBuf[j];
```

- Pack data to be sent to the right
- Single exchange with packed data
- Unpack data received from left
- Repeat for the left to right exchange
- Message size of 4KB

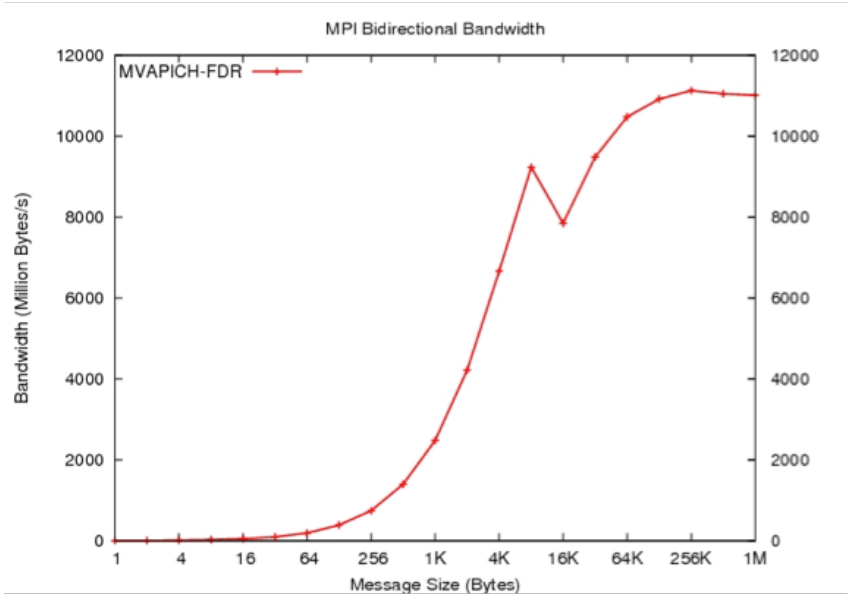


Optimized Code v2

- Pack data to send to both left and right
- Non-blocking data exchange
- Unpack data
- Uses bi-directional capability of IB

Bi-directional

```
1  for( j = 0; j < NY; j++ ){
2     sendBufRight[j] = f[NX][j];
3     sendBufLeft[j] = f[1][j];
4  }
5
6  MPI_Irecv(recvBufLeft, NY, MPI_DOUBLE,
          left,...);
7  MPI_Irecv(recvBufRight,NY, MPI_DOUBLE,
          right,...);
8  MPI_Isend(sendBufRight,NY, MPI_DOUBLE,
          right,...);
9  MPI_Isend(sendBufLeft, NY, MPI_DOUBLE,
          left,...);
10 MPI_Waitall( 4, request, status );
11
12 for( j = 0; j < NY; j++ ){
13    f[0][j] = recvBufLeft[j];
14    f[NX+1][j] = recvBufRight[j];
15 }
```



License