

# Introduction to MPI

Masao Fujinaga

Academic Information and Communication Technology  
University of Alberta

## Connect to the linux cluster

- ssh -Y **ccid**@cluster.srv.ualberta.ca
- “-Y” : enables trusted X11 forwarding
- cd /scratch/**ccid**/
- cp -r /scratch/fujinaga/nov2010-mpi .
- cd nov2010-mpi
- ggv nov2010-mpi.pdf &

## Message Passing

- Parallel computation occurs through a number of processes, each with its own local data
- Sharing of data is achieved by message passing. i.e. by explicitly sending and receiving data between processes

## What is MPI?

- MPI
  - Specified by a committee of experts from research and industry
  - Standard message-passing specification for all the Massively Parallel Processor (MPP) vendors involved

- Fortran

```
INCLUDE 'mpif.h'
INTEGER error, rank,len
character*255 hostname
CALL MPI_Init(error)
CALL MPI_Comm_rank(MPI_COMM_WORLD, rank, error)
CALL MPI_Get_processor_name(hostname,len,error)
PRINT *, "Hello world from ",rank,hostname
CALL MPI_Finalize(error)
STOP
END
```

## A simple MPI program

- C

```
#include <stdio.h>
#include <mpi.h>
void main (int argc, char *argv[]) {
    int rank,len;
    char name[255];
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Get_processor_name(name,&len);
    printf("Hello world from %d %s\n", rank,name);
    MPI_Finalize();
}
```

## Different implementations of MPI

- Default

- MPICH2 over gigabit ethernet
- Portland compilers

- OpenMPI

- Infiniband
- gnu compilers

- MVAPICH2

- Infiniband
- gnu compilers

- Intel MPI

- Infiniband
- gnu compiler

## OpenMPI

- module load mpi/openmpi-1.2.5

- which mpif77

/usr/local/openmpi-1.2.5/bin/mpif77

- mpif77 -o hello hello.f -show

gfortran -I/usr/local/openmpi-1.2.5/include -pthread -o hello hello.f -L/usr/local/openmpi-1.2.5/lib -lmpi\_f77 -lmpi -lopen-rte -lopen-pal -ldl -Wl,--export-dynamic -lnsl -lutil -lm -ldl

- Change compiler

C: OMPI\_CC

C++: OMPI\_CXX

Fortran 77: OMPI\_F77

Fortran 90: OMPI\_FC

- setenv OMPI\_F77 pgf77

- mpif77 -o hello hello.f -show

pgf77 -I/usr/local/openmpi-1.2.5/include -pthread -o hello hello.f -L/usr/local/openmpi-1.2.5/lib -lmpi\_f77 -lmpi -lopen-rte -lopen-pal -ldl -Wl,--export-dynamic -lnsl -lutil -lm -ldl

## Running an MPI program

- module load mpi/openmpi-1.2.5

- mpif77 -o hello hello.f

- mpicc -o hello hello.c

- mpiexec -np 4 ./hello

Hello world from 0 cluster-login.nic.ualberta.ca

Hello world from 1 cluster-login.nic.ualberta.ca

Hello world from 2 cluster-login.nic.ualberta.ca

Hello world from 3 cluster-login.nic.ualberta.ca

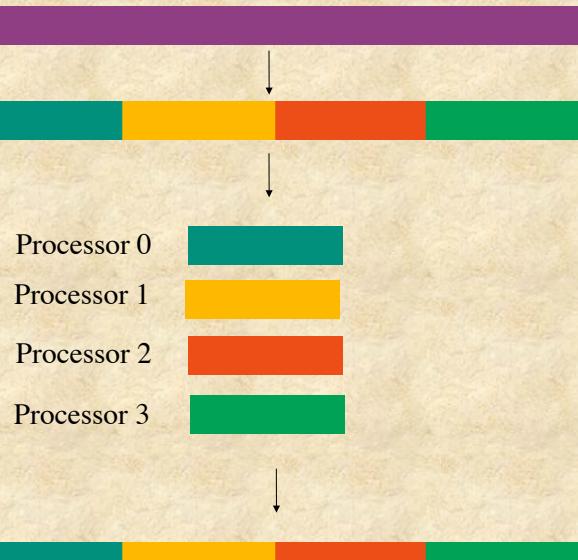
## Running MPI in batch

```
#!/bin/bash -l
#PBS -S /bin/bash
#PBS -l nodes=2:ppn=2
#PBS -l walltime=01:00:00
module load mpi/openmpi-1.2.5
cd $PBS_O_WORKDIR
mpieexec ./hello > out

qsub script.pbs
qstat -u ccid
cat out
```

## Serial program

```
do i = 1, n
    y(i) = x(i)**2.3
enddo
```



## Master-slave program

```
call MPI_Comm_rank(MPI_COMM_WORLD, rank, error)
call MPI_Comm_size(MPI_COMM_WORLD, size, error)

if(rank .eq. 0)then
    - master code
        send data to slaves
        calculate its share of results
        receive results from slaves
else
    - slave code
        receive data from master
        calculate results
        send results to master
endif
```

# MPI\_Send/MPI\_Recv

- MPI\_Send

`MPI_Send(buf, count, type, dest, tag, comm, ierr)`

- MPI\_Recv

`MPI_Recv(buf, count, type, source, tag, comm, status, ierr)`

Wildcards

`MPI_ANY_SOURCE, MPI_ANY_TAG`

Fortran

`status(MPI_SOURCE)`

`status(MPI_TAG)`

`status(MPI_ERROR)`

C

`status.MPI_SOURCE`

`status.MPI_TAG`

`status.MPI_ERROR`

MPI Data Type	Fortran Data Type
<code>MPI_INTEGER</code>	<code>integer</code>
<code>MPI_REAL</code>	<code>real</code>
<code>MPI_DOUBLE_PRECISION</code>	<code>double precision</code>
<code>MPI_COMPLEX</code>	<code>complex</code>
<code>MPI_CHARACTER</code>	<code>character(1)</code>
<code>MPI_LOGICAL</code>	<code>logical</code>
<code>MPI_BYTE</code>	<code>(none)</code>
<code>MPI_PACKED</code>	<code>(none)</code>

MPI Data Type	C Data Type
<code>MPI_INT</code>	<code>int</code>
<code>MPI_LONG</code>	<code>long</code>
<code>MPI_FLOAT</code>	<code>float</code>
<code>MPI_DOUBLE</code>	<code>double</code>
<code>MPI_UNSIGNED_CHAR</code>	<code>unsigned char</code>
<code>MPI_UNSIGNED_SHORT</code>	<code>unsigned short</code>
<code>MPI_UNSIGNED</code>	<code>unsigned int</code>
<code>MPI_UNSIGNED_LONG</code>	<code>unsigned long</code>

```

integer status(MPI_STATUS_SIZE)
call MPI_Comm_rank(MPI_COMM_WORLD, rank, error)
call MPI_Comm_size(MPI_COMM_WORLD, size, error)
npart = nmax/size
if(rank .eq. 0)then
    do iproc = 1, size-1
        index = iproc*npart+1
        call MPI_Send(x(index), npart, MPI_REAL, iproc, 1, MPI_COMM_WORLD,
                     error)
    enddo
    do 210 i = 1, npart
        y(i) = x(i)**2.3
    enddo
    do iproc = 1, size-1
        index = iproc*npart+1
        call MPI_Recv(y(index), npart, MPI_REAL, iproc, 2, MPI_COMM_WORLD,
                      status,error)
    enddo
else
    call MPI_Recv(x, npart, MPI_REAL, 0, 1, MPI_COMM_WORLD, status, error)
    do i = 1, npart
        y(i) = x(i)**2.3
    enddo
    call MPI_Send(y, npart, MPI_REAL, 0, 2, MPI_COMM_WORLD, error)
endif

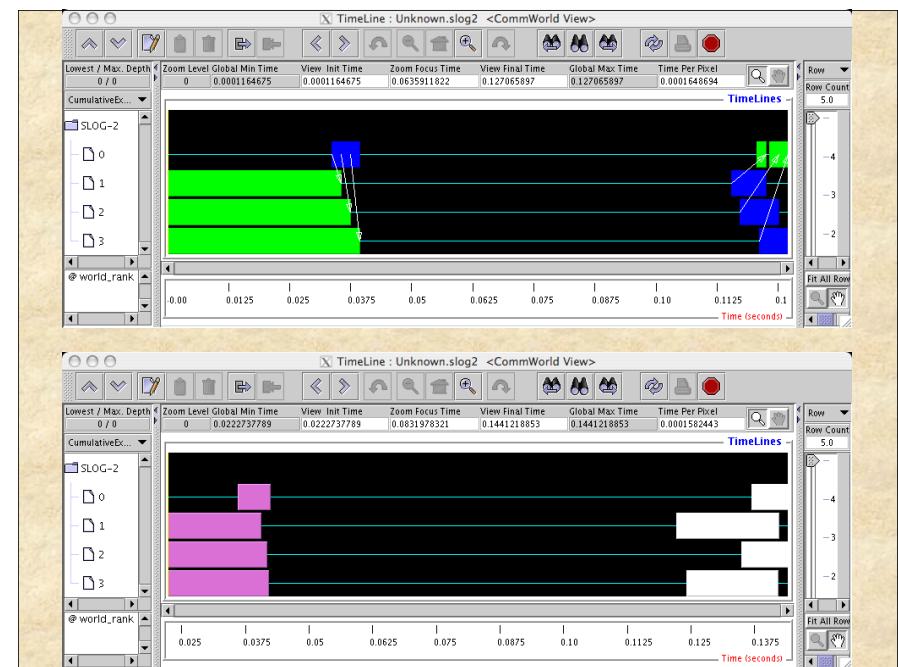
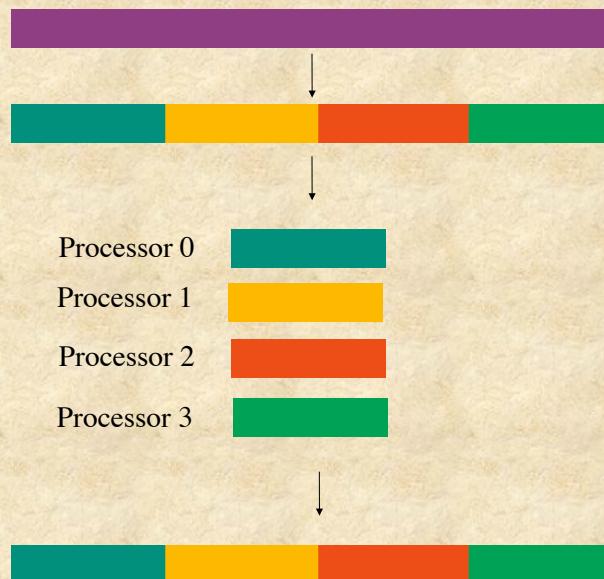
```

## Basic commands

- Include file
- MPI\_Init
- MPI\_Comm\_rank
- MPI\_Comm\_size
- MPI\_Send
- MPI\_Recv
- MPI\_Finalize
- To see list of routines, do  
man -k MPI
- To see details,  
man MPI\_Send

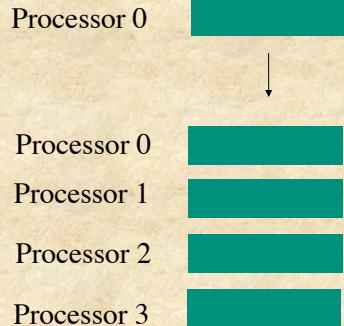
## A simpler way

```
call MPI_Scatter(x, npart, MPI_REAL, x, npart, MPI_REAL,
  0, MPI_COMM_WORLD, error)
do i = 1, npart
  y(i) = x(i)**2.3
enddo
call MPI_Gather(y, npart, MPI_REAL, y, npart, MPI_REAL,
  0, MPI_COMM_WORLD, error)
```



## Broadcast

```
MPI_Bcast(buffer, count, type, rank, comm, ierr)
```



## Exercise 1

- Modify exercise1.f or exercise1.c by adding MPI\_Scatter, MPI\_Bcast and MPI\_Gather
- The program will read from standard input, a1,a2,a3.
- For an array, x, it will calculate  
 $y=a1*x+a2*x+a3$
- Writes to standard output, x and y
- Do ‘man MPI\_Scatter’ etc. to get parameter list.

## Reduction

```
sum = 0
do i = 1, nmax
    sum = sum + x(i)
enddo
```

The diagram consists of two vertically aligned rectangular boxes. The top box contains a sequence of pseudocode: 'sum = 0', 'do i = 1, nmax', ' sum = sum + x(i)', and 'enddo'. The bottom box contains a similar sequence: 'sum = 0', 'subsum = 0', 'do i = 1, npart', ' subsum = subsum + x(i)', and 'enddo'. Below this, there is a call to the MPI\_Reduce function: 'call MPI\_Reduce(subsum, sum, 1, MPI\_INTEGER, MPI\_SUM, 0, MPI\_COMM\_WORLD, ierr)'. An arrow points from the bottom of the top box down to the top of the bottom box, indicating a transformation or comparison between the two code snippets.

```
sum = 0
subsum = 0
do i = 1, npart
    subsum = subsum + x(i)
enddo
call MPI_Reduce(subsum, sum, 1, MPI_INTEGER,
    MPI_SUM, 0, MPI_COMM_WORLD, ierr)
```

## Exercise 2

- Modify exercise2.f or exercise2.c to do the sum in parallel
- Use MPI\_Scatter and MPI\_Reduce

## Performance

- For best performance, minimize communication.
  - Minimize the amount of data transferred and the number of calls to message passing routines.
- Next best thing: Minimize communication time relative to computation time.
- Or overlap communication with calculation
- Avoid synchronization steps.
- Make sure that all processes are busy (load balancing)

## Benchmarking

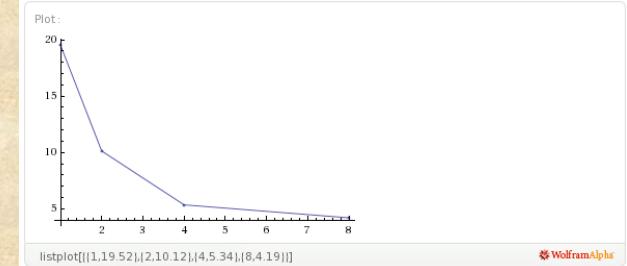
- For any parallel program, it is important to know the parallel efficiency of the program
- time the real time it takes to run the program using various number of processors

## Exercise 3

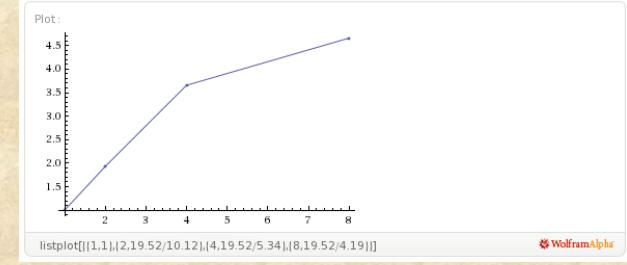
- compile sample1long.f (or .c)  
`mpif77 -o sample1long sample1long.f`
- submit script3.pbs. This will run the program four times, changing -n to 1, 2, 4, and 8
- look at the real time ('grep real out.time')
- plot using [www.wolframalpha.com](http://www.wolframalpha.com)
- plot time  
`listplot[{{1,t1},{2,t2},{4,t4},{8,t8}}]`
- plot speedup  
`listplot[{{1,1},{2,t1/t2},{4,t1/t4},{8,t1/t8}}]`

## Benchmark results

Time



Speedup



## Performance analysis with OpenMPI/MPE

- Set up environment

```
module load mpi/openmpi-1.2.5
```

- Compile program

```
mpefc -o sample1 sample1.f -mpilog
```

```
mpecc -o sample1 sample1.c -mpilog
```

- Run program

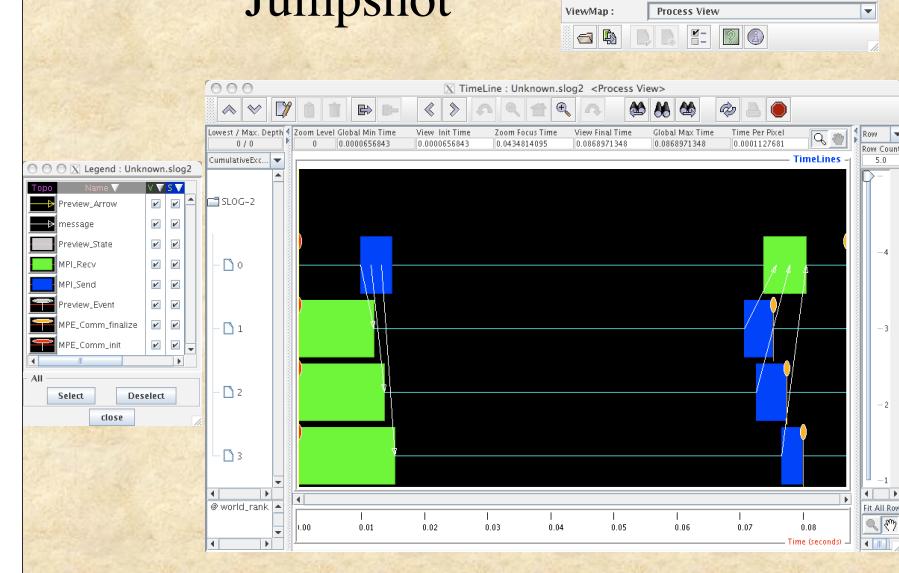
```
mpiexec -np 4 ./sample1
```

- View results

```
clog2TOslog2 Unknown.clog2
```

```
jumpshot Unknown.slog2
```

(sample1.clog2 and sample.slog2 for c)



## MPI\_Wtime()

- Returns elapsed (wall) time on the calling processor

- Time in seconds since an arbitrary time in the past

```
real*8 time
```

```
time = MPI_Wtime()
```

```
Calculate...
```

```
write(*,*)' elapsed time =' ,MPI_Wtime()-time
```

---

```
double time;
time = MPI_Wtime();
Calculate ...
printf("elapsded time = %f\n",MPI_Wtime()-time);
```

## Blocking and Completion

- MPI\_Send and MPI\_Recv block the calling process. i.e. they do not return until the communication operation is complete
- MPI\_Recv is complete when the message is copied to the output variable.
- MPI\_Send is complete when the message has been passed off to MPI

# Deadlock

- When two or more blocked processes are waiting for each other and cannot make progress.

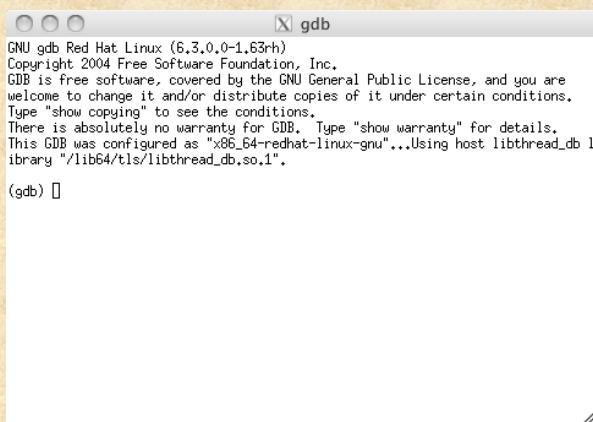
```
if( rank .eq. 0)then
    call MPI_Recv(x, nmax, MPI_REAL, 1, 1, MPI_COMM_WORLD, status, ierr)
    call MPI_Send(y, nmax, MPI_REAL, 1, 2, MPI_COMM_WORLD, ierr)
else if ( rank .eq. 1)then
    call MPI_Recv(y, nmax, MPI_REAL, 0, 2, MPI_COMM_WORLD, status, ierr)
    call MPI_Send(x, nmax, MPI_REAL, 0, 1, MPI_COMM_WORLD, ierr)
endif
```

## Debugging deadlock - 1

- mpif77 -o deadlock0 deadlock0.f -g
- mpiexec -n 2 ./deadlock0
- ctrl-c to exit program

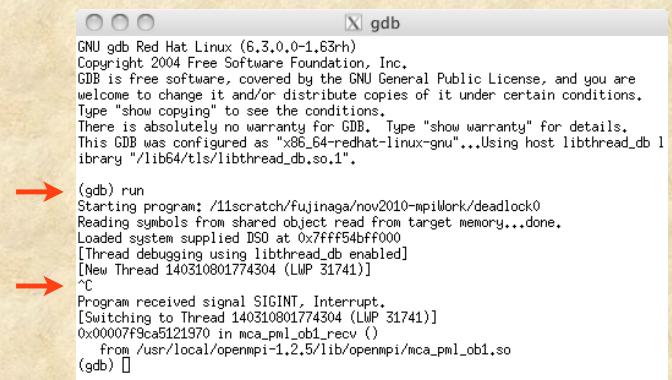
## Debugging deadlock - 2

- mpiexec -n 2 xterm -e gdb ./deadlock0



## Debugging deadlock - 3

- In each window:
  - type 'run' to start program
  - ctrl-c to halt program



## Debugging deadlock - 4

- type ‘where’

```
gdb
Reading symbols from shared object read from target memory...done.
Loaded system supplied DSO at 0x7ffff1ff000
[Thread debugging using libthread_db enabled]
[New Thread 139856254928608 (LWP 31691)]
^C
Program received signal SIGINT, Interrupt.
[Switching to Thread 139856254928608 (LWP 31691)]
0x00007f32cfe7222d in mca_bml_r2_progress ()
    from /usr/local/openmpi-1.2.5/lib/openmpi/mca_bml_r2.so
(gdb) where
#0 0x00007f32cfe7222d in mca_bml_r2_progress ()
    from /usr/local/openmpi-1.2.5/lib/openmpi/mca_bml_r2.so
#1 0x00007f32d2960b2a in opal_progress ()
    from /usr/local/openmpi-1.2.5/lib/openpal-pal.so
#2 0x00007f32cff7a975 in mca_pml_ob1_recv ()
    from /usr/local/openmpi-1.2.5/lib/openmpi/mca_pml_ob1.so
#3 0x00007f32d2c678e8 in PMPI_Recv ()
    from /usr/local/openmpi-1.2.5/lib/libmpi.so.0
#4 0x00007f32d2dc853b in pmapi_recv_()
    from /usr/local/openmpi-1.2.5/lib/libmpi_f77.so.0
#5 0x0000000000400d84 in MAIN_( ) at deadlock0.f:26
#6 0x0000000000400ddc in main (argc=Variable "argc" is not available,
) at ../../libgftran/fmain.c:18
(gdb) 
```

## Debugging deadlock - 5

- type ‘frame 5’ ( or whichever number corresponds to the user code)
- type ‘info local’

```
gdb
from /usr/local/openmpi-1.2.5/lib/libopen-pal.so.0
#2 0x00007f32cff7a975 in mca_pml_ob1_recv ()
    from /usr/local/openmpi-1.2.5/lib/openmpi/mca_pml_ob1.so
#3 0x00007f32d2c678e8 in PMPI_Recv ()
    from /usr/local/openmpi-1.2.5/lib/libmpi.so.0
#4 0x00007f32d2dc853b in pmapi_recv_()
    from /usr/local/openmpi-1.2.5/lib/libmpi_f77.so.0
#5 0x0000000000400d84 in MAIN_( ) at deadlock0.f:26
#6 0x0000000000400ddc in main (argc=Variable "argc" is not available,
) at ../../libgftran/fmain.c:18
(gdb) frame 5
#5 0x0000000000400d84 in MAIN_( ) at deadlock0.f:26
26 . mpi_comm_world,status,error
Current language: auto; currently fortran
(gdb) info local
error = 0
i = 10001
rank = 1
status = (1, 0, 521689840, 32767, 0)
size = 2
y = (2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46
    48
    50
    52
    54
    56
    58
    60
    62
    64
    66
    68
    70
    72
    74
    76
    78
    80
    82
    84
    86
    88
    90
    92
    94
    96
    98
    100
    102
    104
    106
    108
    110
    112
    114
    116
    118
    120
    122
    124
    126
    128
    130
    132
    134
    136
    138
    140
    142
    144
    146
    148
    150
    152
    154
    156
    158
    160
    162
    164
    166
    168
    170
    172
    174
    176
    178
    180
    182
    184
    186
    188
    190
    192
    194
    196
    198
    200
    202
    204
    206
    208
    210
    212
    214
    216
    218
    220
    222
    224
    226
    228
    230
    232
    234
    236
    238
    240
    242
    244
    246
    248
    250
    252
    254
    256
    258
    260
    262
    264
    266
    268
    270
    272
    274
    276
    278
    280
    282
    284
    286
    288
    290
    292
    294
    296
    298
    300
    302
    304
    306
    308
    310
    312
    314
    316
    318
    320
    322
    324
    326
    328
    330
    332
    334
    336
    338
    340
    342
    344
    346
    348
    350
    352
    354
    356
    358
    360
    362
    364
    366
    368
    370
    372
    374
    376
    378
    380
    382
    384
    386
    388
    390
    392
    394
    396
    398
    400
    402
    404
    406
    408
    410
    412
    414
    416
    418
    420
    422
    424
    426
    428
    430
    432
    434
    436
    438
    440
    442
    444
    446
    448
    450
    452
    454
    456
    458
    460
    462
    464
    466
    468
    470
    472
    474
    476
    478
    480
    482
    484
    486
    488
    490
    492
    494
    496
    498
    500
    502
    504
    506
    508
    510
    512
    514
    516
    518
    520
    522
    524
    526
    528
    530
    532
    534
    536
    538
    540
    542
    544
    546
    548
    550
    552
    554
    556
    558
    560
    562
    564
    566
    568
    570
    572
    574
    576
    578
    580
    582
    584
    586
    588
    590
    592
    594
    596
    598
    600
    602
    604
    606
    608
    610
    612
    614
    616
    618
    620
    622
    624
    626
    628
    630
    632
    634
    636
    638
    640
    642
    644
    646
    648
    650
    652
    654
    656
    658
    660
    662
    664
    666
    668
    670
    672
    674
    676
    678
    680
    682
    684
    686
    688
    690
    692
    694
    696
    698
    700
    702
    704
    706
    708
    710
    712
    714
    716
    718
    720
    722
    724
    726
    728
    730
    732
    734
    736
    738
    740
    742
    744
    746
    748
    750
    752
    754
    756
    758
    760
    762
    764
    766
    768
    770
    772
    774
    776
    778
    780
    782
    784
    786
    788
    790
    792
    794
    796
    798
    800
    802
    804
    806
    808
    810
    812
    814
    816
    818
    820
    822
    824
    826
    828
    830
    832
    834
    836
    838
    840
    842
    844
    846
    848
    850
    852
    854
    856
    858
    860
    862
    864
    866
    868
    870
    872
    874
    876
    878
    880
    882
    884
    886
    888
    890
    892
    894
    896
    898
    900
    902
    904
    906
    908
    910
    912
    914
    916
    918
    920
    922
    924
    926
    928
    930
    932
    934
    936
    938
    940
    942
    944
    946
    948
    950
    952
    954
    956
    958
    960
    962
    964
    966
    968
    970
    972
    974
    976
    978
    980
    982
    984
    986
    988
    990
    992
    994
    996
    998
    1000
    1002
    1004
    1006
    1008
    1010
    1012
    1014
    1016
    1018
    1020
    1022
    1024
    1026
    1028
    1030
    1032
    1034
    1036
    1038
    1040
    1042
    1044
    1046
    1048
    1050
    1052
    1054
    1056
    1058
    1060
    1062
    1064
    1066
    1068
    1070
    1072
    1074
    1076
    1078
    1080
    1082
    1084
    1086
    1088
    1090
    1092
    1094
    1096
    1098
    1100
    1102
    1104
    1106
    1108
    1110
    1112
    1114
    1116
    1118
    1120
    1122
    1124
    1126
    1128
    1130
    1132
    1134
    1136
    1138
    1140
    1142
    1144
    1146
    1148
    1150
    1152
    1154
    1156
    1158
    1160
    1162
    1164
    1166
    1168
    1170
    1172
    1174
    1176
    1178
    1180
    1182
    1184
    1186
    1188
    1190
    1192
    1194
    1196
    1198
    1200
    1202
    1204
    1206
    1208
    1210
    1212
    1214
    1216
    1218
    1220
    1222
    1224
    1226
    1228
    1230
    1232
    1234
    1236
    1238
    1240
    1242
    1244
    1246
    1248
    1250
    1252
    1254
    1256
    1258
    1260
    1262
    1264
    1266
    1268
    1270
    1272
    1274
    1276
    1278
    1280
    1282
    1284
    1286
    1288
    1290
    1292
    1294
    1296
    1298
    1300
    1302
    1304
    1306
    1308
    1310
    1312
    1314
    1316
    1318
    1320
    1322
    1324
    1326
    1328
    1330
    1332
    1334
    1336
    1338
    1340
    1342
    1344
    1346
    1348
    1350
    1352
    1354
    1356
    1358
    1360
    1362
    1364
    1366
    1368
    1370
    1372
    1374
    1376
    1378
    1380
    1382
    1384
    1386
    1388
    1390
    1392
    1394
    1396
    1398
    1400
    1402
    1404
    1406
    1408
    1410
    1412
    1414
    1416
    1418
    1420
    1422
    1424
    1426
    1428
    1430
    1432
    1434
    1436
    1438
    1440
    1442
    1444
    1446
    1448
    1450
    1452
    1454
    1456
    1458
    1460
    1462
    1464
    1466
    1468
    1470
    1472
    1474
    1476
    1478
    1480
    1482
    1484
    1486
    1488
    1490
    1492
    1494
    1496
    1498
    1500
    1502
    1504
    1506
    1508
    1510
    1512
    1514
    1516
    1518
    1520
    1522
    1524
    1526
    1528
    1530
    1532
    1534
    1536
    1538
    1540
    1542
    1544
    1546
    1548
    1550
    1552
    1554
    1556
    1558
    1560
    1562
    1564
    1566
    1568
    1570
    1572
    1574
    1576
    1578
    1580
    1582
    1584
    1586
    1588
    1590
    1592
    1594
    1596
    1598
    1600
    1602
    1604
    1606
    1608
    1610
    1612
    1614
    1616
    1618
    1620
    1622
    1624
    1626
    1628
    1630
    1632
    1634
    1636
    1638
    1640
    1642
    1644
    1646
    1648
    1650
    1652
    1654
    1656
    1658
    1660
    1662
    1664
    1666
    1668
    1670
    1672
    1674
    1676
    1678
    1680
    1682
    1684
    1686
    1688
    1690
    1692
    1694
    1696
    1698
    1700
    1702
    1704
    1706
    1708
    1710
    1712
    1714
    1716
    1718
    1720
    1722
    1724
    1726
    1728
    1730
    1732
    1734
    1736
    1738
    1740
    1742
    1744
    1746
    1748
    1750
    1752
    1754
    1756
    1758
    1760
    1762
    1764
    1766
    1768
    1770
    1772
    1774
    1776
    1778
    1780
    1782
    1784
    1786
    1788
    1790
    1792
    1794
    1796
    1798
    1800
    1802
    1804
    1806
    1808
    1810
    1812
    1814
    1816
    1818
    1820
    1822
    1824
    1826
    1828
    1830
    1832
    1834
    1836
    1838
    1840
    1842
    1844
    1846
    1848
    1850
    1852
    1854
    1856
    1858
    1860
    1862
    1864
    1866
    1868
    1870
    1872
    1874
    1876
    1878
    1880
    1882
    1884
    1886
    1888
    1890
    1892
    1894
    1896
    1898
    1900
    1902
    1904
    1906
    1908
    1910
    1912
    1914
    1916
    1918
    1920
    1922
    1924
    1926
    1928
    1930
    1932
    1934
    1936
    1938
    1940
    1942
    1944
    1946
    1948
    1950
    1952
    1954
    1956
    1958
    1960
    1962
    1964
    1966
    1968
    1970
    1972
    1974
    1976
    1978
    1980
    1982
    1984
    1986
    1988
    1990
    1992
    1994
    1996
    1998
    2000
    2002
    2004
    2006
    2008
    2010
    2012
    2014
    2016
    2018
    2020
    2022
    2024
    2026
    2028
    2030
    2032
    2034
    2036
    2038
    2040
    2042
    2044
    2046
    2048
    2050
    2052
    2054
    2056
    2058
    2060
    2062
    2064
    2066
    2068
    2070
    2072
    2074
    2076
    2078
    2080
    2082
    2084
    2086
    2088
    2090
    2092
    2094
    2096
    2098
    2100
    2102
    2104
    2106
    2108
    2110
    2112
    2114
    2116
    2118
    2120
    2122
    2124
    2126
    2128
    2130
    2132
    2134
    2136
    2138
    2140
    2142
    2144
    2146
    2148
    2150
    2152
    2154
    2156
    2158
    2160
    2162
    2164
    2166
    2168
    2170
    2172
    2174
    2176
    2178
    2180
    2182
    2184
    2186
    2188
    2190
    2192
    2194
    2196
    2198
    2200
    2202
    2204
    2206
    2208
    2210
    2212
    2214
    2216
    2218
    2220
    2222
    2224
    2226
    2228
    2230
    2232
    2234
    2236
    2238
    2240
    2242
    2244
    2246
    2248
    2250
    2252
    2254
    2256
    2258
    2260
    2262
    2264
    2266
    2268
    2270
    2272
    2274
    2276
    2278
    2280
    2282
    2284
    2286
    2288
    2290
    2292
    2294
    2296
    2298
    2300
    2302
    2304
    2306
    2308
    2310
    2312
    2314
    2316
    2318
    2320
    2322
    2324
    2326
    2328
    2330
    2332
    2334
    2336
    2338
    2340
    2342
    2344
    2346
    2348
    2350
    2352
    2354
    2356
    2358
    2360
    2362
    2364
    2366
    2368
    2370
    2372
    2374
    2376
    2378
    2380
    2382
    2384
    2386
    2388
    2390
    2392
    2394
    2396
    2398
    2400
    2402
    2404
    2406
    2408
    2410
    2412
    2414
    2416
    2418
    2420
    2422
    2424
    2426
    2428
    2430
    2432
    2434
    2436
    2438
    2440
    2442
    2444
    2446
    2448
    2450
    2452
    2454
    2456
    2458
    2460
    2462
    2464
    2466
    2468
    2470
    2472
    2474
    2476
    2478
    2480
    2482
    2484
    2486
    2488
    2490
    2492
    2494
    2496
    2498
    2500
    2502
    2504
    2506
    2508
    2510
    2512
    2514
    2516
    2518
    2520
    2522
    2524
    2526
    2528
    2530
    2532
    2534
    2536
    2538
    2540
    2542
    2544
    2546
    2548
    2550
    2552
    2554
    2556
    2558
    2560
    2562
    2564
    2566
    2568
    2570
    2572
    2574
    2576
    2578
    2580
    2582
    2584
    2586
    2588
    2590
    2592
    2594
    2596
    2598
    2600
    2602
    2604
    2606
    2608
    2610
    2612
    2614
    2616
    2618
    2620
    2622
    2624
    2626
    2628
    2630
    2632
    2634
    2636
    2638
    2640
    2642
    2644
    2646
    2648
    2650
    2652
    2654
    2656
    2658
    2660
    2662
    2664
    2666
    2668
    2670
    2672
    2674
    2676
    2678
    2680
    2682
    2684
    2686
    2688
    2690
    2692
    2694
    2696
    2698
    2700
    2702
    2704
    2706
    2708
    2710
    2712
    2714
    2716
    2718
    2720
    2722
    2724
    2726
    2728
    2730
    2732
    2734
    2736
    2738
    2740
    2742
    2744
    2746
    2748
    2750
    2752
    2754
    2756
    2758
    2760
    2762
    2764
    2766
    2768
    2770
    2772
    2774
    2776
    2778
    2780
    2782
    2784
    2786
    2788
    2790
    2792
    2794
    2796
    2798
    2800
    2802
    2804
    2806
    2808
    2810
    2812
    2814
    2816
    2818
    2820
    2822
    2824
    2826
    2828
    2830
    2832
    2834
    2836
    2838
    2840
    2842
    2844
    2846
    2848
    2850
    2852
    2854
    2856
    2858
    2860
    2862
    2864
    2866
    2868
    2870
    2872
    2874
    2876
    2878
    2880
    2882
    2884
    2886
    2888
    2890
    2892
    2894
    2896
    2898
    2900
    2902
    2904
    2906
    2908
    2910
    2912
    2914
    2916
    2918
    2920
    2922
    2924
    2926
    2928
    2930
    2932
    2934
    2936
    2938
    2940
    2942
    2944
    2946
    2948
    2950
    2952
    2954
    2956
    2958
    2960
    2962
    2964
    2966
    2968
    2970
    2972
    2974
    2976
    2978
    2980
    2982
    2984
    2986
    2988
    2990
    2992
    2994
    2996
    2998
    3000
    3002
    3004
    3006
    3008
    3010
    3012
    3014
    3016
    3018
    3020
    3022
    3024
    3026
    3028
    3030
    3032
    3034
    3036
    3038
    3040
    3042
    3044
    3046
    3048
    3050
    3052
    3054
    3056
    3058
    3060
    3062
    3064
    3066
    3068
    3070
    3072
    3074
    3076
    3078
    3080
    
```

## Deadlock - solution 2

```
if( rank .eq. 0)then
    call MPI_Recv(x, nmax, MPI_REAL, 1, 1, MPI_COMM_WORLD, status, ierr)
    call MPI_Send(y, nmax, MPI_REAL, 1, 2, MPI_COMM_WORLD, ierr)
else if ( rank .eq. 1)then
    call MPI_Send(x, nmax, MPI_REAL, 0, 1, MPI_COMM_WORLD, ierr)
    call MPI_Recv(y, nmax, MPI_REAL, 0, 2, MPI_COMM_WORLD, status, ierr)
endif
```

## More debugging

- Setting breakpoints  
eg. break deadlock2.f:17
- Check for deadlock and unbalanced send/receive.
- Make use of tags to make sure that the correct message is received.
- Write statements to make sure that the contents of messages are correct.
- Add MPI\_Barrier to synchronize processes.

## Nonblocking Sends and Receives

- Separate send (or receive) into initiation and completion
- Initiation is nonblocking thus allowing other instructions to be processed
- Completion stage can either be a blocking wait or a nonblocking test

## MPI\_ISEND/MPI\_IRecv

- Similar to MPI\_Send/MPI\_Recv except for an addition of a request handle and the lack of a status in MPI\_Irecv

`MPI_Send(buf, count, type, dest, tag, comm, ierr)`

`MPI_Isend(buf, count, type, dest, tag, comm, req, ierr)`

`MPI_Recv(buf, count, type, source, tag, comm, status, ierr)`

`MPI_Irecv(buf, count, type, source, tag, comm, req, ierr)`

## Completion waiting and testing

- Completion waiting blocks until the initiated process is completed

`MPI_Wait(req, status, ierr)`

- Completion testing returns immediately with flag set to true if the process is complete

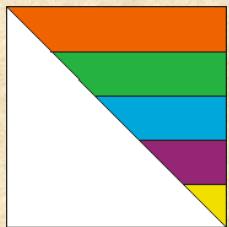
`MPI_Test(req, flag, status, ierr)`

## Load balancing

- Make sure that all the processes are busy.
- Make sure that each process has the same amount of work.
- sample - calculating distances

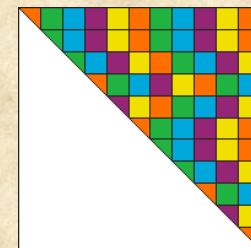
```
do j = 1, n-1  
    do i = j+1, n  
        dist(i,j) = sqrt((x(i)-x(j))**2)  
    enddo  
enddo
```

## Load balancing continued



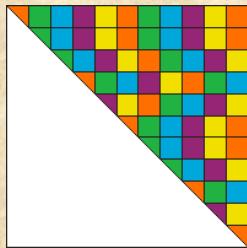
## Dynamic scheduling

- Small chunks of work are given to each process
- As each process finishes the chunk of work, it gets the next chunk of work



## Master - part 1

```
irank=1
do j=1,n
do i=j,n
if (irank .lt. size) then
  send data to irank
  irank = irank +1
else
  receive result from ifree
  send data to ifree
endif
enddo
enddo
```



## Master - part 2

```
do irank = 1, size-1
  receive the remaining results
  send termination signal to processes
enddo
```

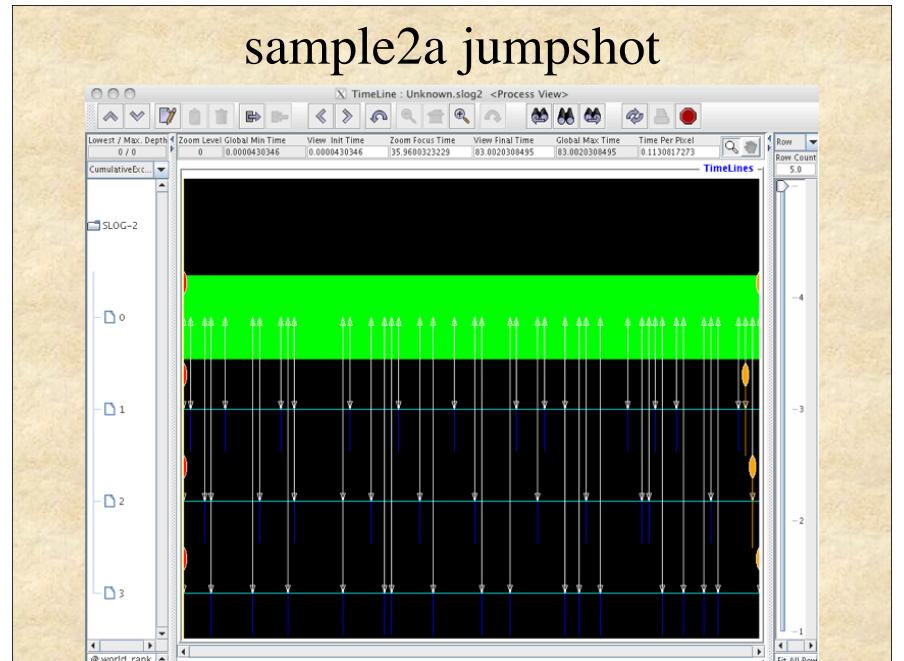
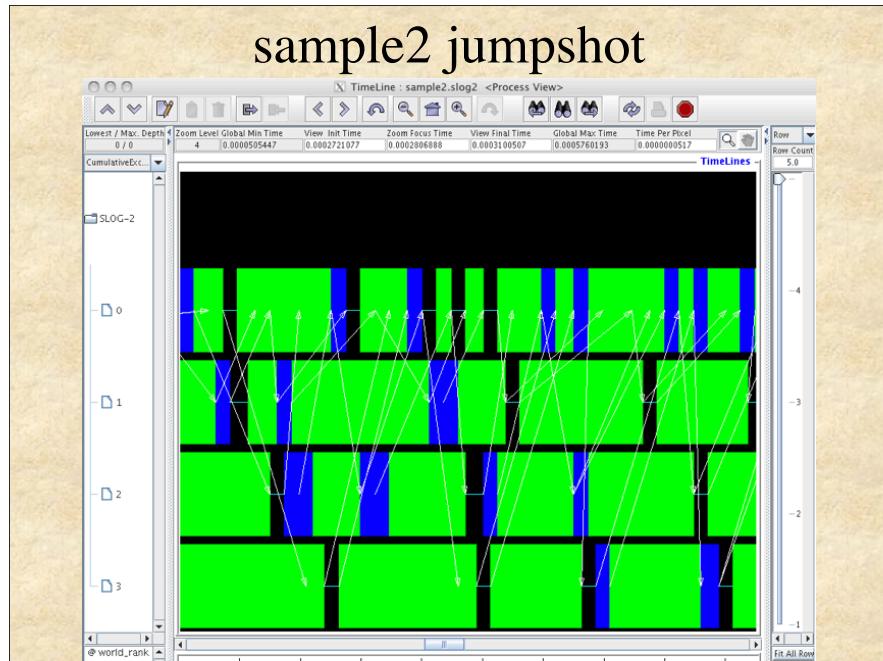
## Slave

```
do
receive data
if( termination signal ) exit
calculate
send results
enddo
```

## Sample2

```
mpefc -o sample2 sample2.f -mpilog
mpiexec -n 4 ./sample2
clog2TOslog2 Unknown.clog2
jumpshot Unknown.slog2
```

```
mpecc -o sample2 sample2.c -mpilog
mpiexec -n 4 ./sample2
clog2TOslog2 sample2.clog2
jumpshot sample2.slog2
```



- Books
    - Using MPI: Portable Parallel Programming with the Message Passing Interface
      - William Gropp, Ewing Lusk, and Anthony Skjellum
    - Parallel Programming with MPI
      - Peter Pacheco
  - Websites
    - MPI: The Complete Reference
      - <http://www.netlib.org/utk/papers/mpi-book/mpi-book.html>
    - Introduction to MPI
      - <http://ci-tutor.ncsa.uiuc.edu/browse.php>
    - MPI and MPE routines
      - <http://www-unix.mcs.anl.gov/mpi/www>
  - [research.support@ualberta.ca](mailto:research.support@ualberta.ca)