

HW07 - Global Sum

CS5500

Chelsi Gupta

October 13, 2018

1 Implementation Variations

I tried 4 different variations for sending the global_sum(of numbers at all processes) to all the processes in the cluster. These are:

1.1 global_sum_ring1

In this variant I generated a random number at each process and then circulated those numbers by passing the numbers from rank to $(\text{rank}+1)\% \text{size}$ one by one in a ring fashion. So each process(except 0) will receive data from its rank - 1, add it to its global sum and then pass the global sum to rank + 1. 0 starts by sending the data and at last when 0 receives the global sum then it sends it to all the other processes.

Compile and Run: 'mpirun -oversubscribe -np 8 python global_sum_ring1.py'

Time: The maximum time taken for one amongst the 8 processes was 0.083 seconds.

1.1.1 Code

```
import time
start_time = time.time()
from mpi4py import MPI
import math
import random
comm = MPI.COMM_WORLD

size = comm.Get_size()
rank = comm.Get_rank()
num = random.randint(1,5)
global_sum = num

if rank!=0:
    rdata = comm.recv(source=rank-1)
    global_sum += rdata
```

```

comm.send(global_sum, dest=(rank+1)%size)

if rank==0:
    rdata = comm.recv(source=size-1)
    for i in range(1, size, 1):
        comm.send(rdata, dest=i)

if rank != 0:
    rdata = comm.recv(source=0)

print('I am process', rank, ' had:', num, " global sum:", rdata, ' time:', "%s seconds" % (tim

```

1.1.2 Output

```

chelsi@chelsi-HP-Z220-CMT-Workstation: ~/Parallel_Computing/hw07$ mpirun --oversubscribe -np 8 python global_sum_ring1.py
I am process 1 had: 9 global sum: 50 time: 0.08716511726379395 seconds
I am process 2 had: 4 global sum: 50 time: 0.08026528358459473 seconds
chelsi@chelsi-HP-Z220-CMT-Workstation:~/Parallel_Computing/hw07$ mpirun --oversubscribe -np 8 python global_sum_ring1.py
I am process 6 had: 1 global sum: 37 time: 0.0613253116607666 seconds
I am process 7 had: 2 global sum: 37 time: 0.05303525924682617 seconds
I am process 0 had: 5 global sum: 37 time: 0.07448863983154297 seconds
I am process 1 had: 3 global sum: 37 time: 0.07445573806762695 seconds
I am process 2 had: 9 global sum: 37 time: 0.08318853378295898 seconds
I am process 3 had: 8 global sum: 37 time: 0.0769810676574707 seconds
I am process 4 had: 8 global sum: 37 time: 0.06328415870666504 seconds
I am process 5 had: 1 global sum: 37 time: 0.06255292892456055 seconds
chelsi@chelsi-HP-Z220-CMT-Workstation:~/Parallel_Computing/hw07$ 

```

Figure 1: Ring 1

1.2 global_sum_ring2

In this variant I generated a random number at each process and then circulated those numbers by passing the numbers from rank to $(rank+1)\%size$ in a ring fashion. So each process(except 0) will receive data from its rank - 1, add it to its global sum and then pass the previously received data to rank + 1,the process goes on and after 'size -1' steps like these each process has the global sum. Here each process has to maintain two data with them one is the global sum and other is the data received. All the processes send at once here and do not have to wait for receiving data from their previous process.

Compile and Run: 'mpirun -oversubscribe -np 8 python global_sum_ring2.py'

Time: The maximum time taken for one amongst the 8 processes was 0.085 seconds.

1.2.1 Code

```
import time
start_time = time.time()
from mpi4py import MPI
import math
import random
comm = MPI.COMM_WORLD

size = comm.Get_size()
rank = comm.Get_rank()
num = random.randint(1,5)
global_sum = num
rdata = num

for i in range(1,size,1):
    comm.send(rdata,dest=(rank+1)%size)
    if rank != 0:
        rdata = comm.recv(source=rank-1)
    else:
        rdata = comm.recv(source=size-1)
    global_sum += rdata

print ( 'I_am_process_',rank, '_had:',num,"_global_sum:",global_sum, '_time:', "%s_seconds" %
```

1.2.2 Output

```
chelsi@chelsi-HP-Z220-CMT-Workstation: ~/Parallel_Computing/hw07

I am process 5 had: 1 global sum: 37 time: 0.06255292892456055 seconds

chelsi@chelsi-HP-Z220-CMT-Workstation:~/Parallel_Computing/hw07$ mpirun --oversubscribe -np 8 python global_sum_ring2.py
I am process 0 had: 4 global sum: 42 time: 0.08067750930786133 seconds
I am process 1 had: 3 global sum: 42 time: 0.07773613929748535 seconds
I am process 2 had: 1 global sum: 42 time: 0.08576107025146484 seconds
I am process 3 had: 9 global sum: 42 time: 0.07449173927307129 seconds
I am process 6 had: 9 global sum: 42 time: 0.06785893440246582 seconds
I am process 7 had: 7 global sum: 42 time: 0.057119131088256836 seconds

I am process 4 had: 4 global sum: 42 time: 0.07450032234191895 seconds
I am process 5 had: 5 global sum: 42 time: 0.07067108154296875 seconds

chelsi@chelsi-HP-Z220-CMT-Workstation:~/Parallel_Computing/hw07$
```

Figure 2: Ring 2

1.3 global_sum_cube1

In this variant I generated a random number at each process and made the processes to exchange data through cube exchange. Cube function is used to determine which process we have to exchange the data with. After determining the destination, we check if the rank of dest is greater than the current rank then the dest process sends the data and the current process receives it and vice-versa. So in this way at the end process 0 will have the global sum and we follow the reverse process by sending the data to the process which has a greater rank then the current process.

Compile and Run: 'mpirun -oversubscribe -np 8 python global_sum_cube1.py'

Time: The maximum time taken for one amongst the 8 processes was 0.126 seconds.

1.3.1 Code

```
import time
```

```

start_time = time.time()
from mpi4py import MPI
import math
import random
comm = MPI.COMM_WORLD

size = comm.Get_size()
rank = comm.Get_rank()
num = random.randint(1,5)
rdata = 0

def cube(i):
    size = comm.Get_size()
    rank = comm.Get_rank()
    mask = 1
    mask = mask << i
    dest = rank ^ mask
    #dest is the process we want to exchange the data with
    return dest

global_sum = num
k = int(math.log2(size))

for i in range(0,k,1):
    dest = cube(i)
    if dest < rank:
        comm.send(global_sum, dest=dest)
    else:
        rdata = comm.recv(source=dest)
        global_sum += rdata

for i in range(k-1,-1,-1):
    dest = cube(i)
    if dest > rank:
        comm.send(global_sum, dest=dest)
    else:
        rdata = comm.recv(source=dest)
        global_sum = rdata

print ( 'I_am_process_',rank, '_had:',num,"_global_sum:",global_sum, '_time:', "%s_seconds" %

```

1.3.2 Output

```
chelsi@chelsi-HP-Z220-CMT-Workstation: ~/Parallel_Computing/hw07

I am process 0 had: 7 global sum: 40 time: 0.08192110061645508 seconds

chelsi@chelsi-HP-Z220-CMT-Workstation:~/Parallel_Computing/hw07$ mpirun --oversubscribe -np 8 python global_sum_cube1.py
I am process 0 had: 5 global sum: 43 time: 0.11990737915039062 seconds
I am process 1 had: 10 global sum: 43 time: 0.12631630897521973 seconds
I am process 4 had: 8 global sum: 43 time: 0.10987067222595215 seconds
I am process 5 had: 7 global sum: 43 time: 0.097381591796875 seconds
I am process 6 had: 7 global sum: 43 time: 0.0788576602935791 seconds
I am process 7 had: 3 global sum: 43 time: 0.07482790946960449 seconds
I am process 2 had: 1 global sum: 43 time: 0.1149749755859375 seconds
I am process 3 had: 2 global sum: 43 time: 0.11459541320800781 seconds
chelsi@chelsi-HP-Z220-CMT-Workstation:~/Parallel_Computing/hw07$
```

Figure 3: Cube 1

1.4 global_sum_cube1

In this variant I generated a random number at each process and made the processes to exchange data through cube exchange. Cube function is used to determine which process we have to exchange the data with. Through cube exchange each process will send and receive data from its cube exchanger. The maximum cube exchange we do here is $\log(\text{size}-1)$. After the $\log(\text{size}-1)$ th cube exchange each process will have the global sum.

Compile and Run: 'mpirun -oversubscribe -np 8 python global_sum_cube2.py'

Time: The maximum time taken for one amongst the 8 processes was 0.087 seconds.

1.4.1 Code

```
import time
start_time = time.time()
from mpi4py import MPI
import math
```

```

import random
comm = MPI.COMM_WORLD

size = comm.Get_size()
rank = comm.Get_rank()
num = random.randint(1,5)

def cube(i, sendData):
    size = comm.Get_size()
    rank = comm.Get_rank()
    mask = 1
    mask = mask << i
    dest = rank ^ mask
    #dest is the process we want to exchange the data with
    comm.send(sendData, dest=dest)
    recvData = comm.recv(source=dest)
    return recvData

global_sum = num

k = int(math.log2(size))
for i in range(0,k,1):
    rdata = cube(i,global_sum)
    global_sum += rdata

print ( 'I am process ',rank, ' had: ',num," global sum: ",global_sum, ' time: ', "%s seconds" %

```

1.4.2 Output

```
chelsi@chelsi-HP-Z220-CMT-Workstation: ~/Parallel_Computing/hw07
chelsi@chelsi-HP-Z220-CMT-Workstation:~/Parallel_Computing/hw07$ mpirun --oversu
bscribe -np 8 python global_sum_cube2.py
I am process 3 had: 10 global sum: 50 time: 0.07598352432250977 seconds

I am process 4 had: 2 global sum: 50 time: 0.06237435340881348 seconds
I am process 5 had: 1 global sum: 50 time: 0.061614036560058594 seconds

I am process 6 had: 8 global sum: 50 time: 0.061005592346191406 seconds
I am process 7 had: 7 global sum: 50 time: 0.062375545501708984 seconds

I am process 0 had: 9 global sum: 50 time: 0.08550429344177246 seconds
I am process 1 had: 9 global sum: 50 time: 0.08716511726379395 seconds
I am process 2 had: 4 global sum: 50 time: 0.08026528358459473 seconds

chelsi@chelsi-HP-Z220-CMT-Workstation:~/Parallel_Computing/hw07$
```

Figure 4: Cube 2