Tugas : Latihan 4 MPI
Kelas : IF-41-09
Kelompok : 7
Anggota : Aqmarina Alifah Ismahyati / 1301174058 (Tugas 3 dan Tugas 4)
: Alvinda Julian Trismadi / 1301174147 (Tugas 1)
: Jodi Kusuma / 1301174265 (Tugas 2 dan Tugas 5)

Screenshot Hasil Program

1. Tugas 1 : point-to-point communication
    a. Source Code

01.mpi_p2p_terbesar.py

```python
# import mpi4py
from mpi4py import MPI

# buat COMM
comm = MPI.COMM_WORLD

# dapatkan rank proses
rank = comm.Get_rank()

# dapatkan total proses berjalan
size = comm.Get_size()

# jika saya rank terbesar maka saya akan mengirimkan pesan ke proses yang mempunyai rank 0 s.d rank terbesar-1
if rank == size-1:
    data = "pesan"
    for i in range(size-1):
        comm.send(data,dest=i)
        print('rank', rank, 'mengirim', data, 'ke rank', i)

# jika saya bukan rank terbesar maka saya akan menerima pesan yang berasal dari proses dengan rank terbesar
else:
    data = comm.recv(source = size - 1)
    print('rank', rank, 'menerima', data, 'dari rank', size-1)
```

01.mpi_p2p_terkecil.py

```python
#Import mpi4py
from mpi4py import MPI

# buat COMM
comm = MPI.COMM_WORLD

# dapatkan rank proses
rank = comm.Get_rank()

# dapatkan total proses berjalan
size = comm.Get_size()

# jika saya rank ke 0 maka saya akan mengirimkan pesan ke proses yang mempunyai rank 1 s.d size
if rank == 0:
    for i in range(1, size):
        sendMessage = ("pesan")
        comm.send(sendMessage, dest=i)
        print('rank', rank, 'mengirim', sendMessage, 'ke rank', i)

# jika saya bukan rank 0 maka saya menerima pesan yang berasal dari proses dengan rank 0
else:
    recvMessage = comm.recv(source=0)
    print('rank', rank, 'menerima', recvMessage, 'dari rank',0)
```

b. Output

```
PS C:\Users\Alvinda Julian\Downloads\sisparter_mpi> mpiexec.exe -n 4 python 01.mpi_p2p_terbesar.py
rank 0 menerima pesan dari rank 3
rank 3 mengirim pesan ke rank 0
rank 3 mengirim pesan ke rank 1
rank 3 mengirim pesan ke rank 2
rank 1 menerima pesan dari rank 3
rank 2 menerima pesan dari rank 3
PS C:\Users\Alvinda Julian\Downloads\sisparter_mpi>
```

```
PS C:\Users\Alvinda Julian\Downloads\sisparter_mpi> mpiexec.exe -n 4 python 01.mpi_p2p_terkecil.py
rank 1 menerima pesan dari rank 0
rank 0 mengirim pesan ke rank 1
rank 0 mengirim pesan ke rank 2
rank 0 mengirim pesan ke rank 3
rank 3 menerima pesan dari rank 0
rank 2 menerima pesan dari rank 0
PS C:\Users\Alvinda Julian\Downloads\sisparter_mpi>
```

2. Tugas 2 : broadcast communication
   a. Source Code

c: > Users > Nekozawa > Desktop > 02.bcast_mpi.py > ...

```python
# import mpi4py
from mpi4py import MPI

# buat COMM
COMM = MPI.COMM_WORLD

# dapatkan rank proses
rank = COMM.Get_rank()

# dapatkan total proses berjalan
total = COMM.Get_size()

# jika saya rank 0 maka saya akan melakukan broadscast
if rank == 0 :
    broadscast = {'A' : ('AWAS VIRUS CORONA !!!'),'B' : ('#DirumahAja')}

# jika saya bukan rank 0 maka saya menerima pesan
else :
    broadscast = None

broadscast = COMM.bcast(broadscast, root=0)
print('Rank',rank, broadscast)
```

   b. Output

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
PS C:\Users\Nekozawa\Desktop> mpiexec -n 5 python .\02.bcast_mpi.py
Rank 0 {'A': 'AWAS VIRUS CORONA !!!', 'B': '#DirumahAja'}
Rank 4 {'A': 'AWAS VIRUS CORONA !!!', 'B': '#DirumahAja'}
Rank 2 {'A': 'AWAS VIRUS CORONA !!!', 'B': '#DirumahAja'}
Rank 1 {'A': 'AWAS VIRUS CORONA !!!', 'B': '#DirumahAja'}
Rank 3 {'A': 'AWAS VIRUS CORONA !!!', 'B': '#DirumahAja'}
PS C:\Users\Nekozawa\Desktop>
```

3. Tugas 3 : penjumlahan MPI teknik p2p
    a. Source Code

```python
# import mpi4py
from mpi4py import MPI

# import library random untuk generate angka integer secara random
import random

# buat COMM
comm = MPI.COMM_WORLD

# dapatkan rank proses
rank = comm.Get_rank()

# dapatkan total proses berjalan
size = comm.Get_size()

# generate angka integer secara random untuk setiap proses
angka = random.randint(1,100)

# jika saya adalah proses dengan rank 0 maka:
# saya menerima nilai dari proses 1 s.d proses dengan rank terbesar
# menjumlah semua nilai yang didapat (termasuk nilai proses saya)
if rank == 0:
    sum = 0
    for i in range (1,size):
        nilai = comm.recv(source = i, tag = 1)
        print(nilai)
        sum = sum + nilai['send']
    print("Total = ", sum)
# jika bukan proses dengan rank 0, saya akan mengirimkan nilai proses saya ke proses dengan rank=0
else:
    nilai = {'rank' : rank, 'dest' : 0, 'send' : angka}
    comm.send(nilai,dest=0,tag=1)
```

    b. Output

```
(base) C:\Users\Aqmarina>mpiexec -n 5 python 03.sum_p2p.py
{'rank': 1, 'dest': 0, 'send': 30}
{'rank': 2, 'dest': 0, 'send': 28}
{'rank': 3, 'dest': 0, 'send': 18}
{'rank': 4, 'dest': 0, 'send': 58}
Total =  134
```

4. Tugas 4 : penjumlahan MPI teknik reduce
    a. Source Code

```python
# import mpi4py
from mpi4py import MPI

# import library random untuk generate angka integer secara random
import random

# buat COMM
comm = MPI.COMM_WORLD

# dapatkan rank proses
rank = comm.Get_rank()

# dapatkan total proses berjalan
size = comm.Get_size()

# generate angka integer secara random untuk setiap proses
angka = random.randint(1,100)

# lakukam penjumlahan dengan teknik reduce, root reduce adalah proses dengan rank 0
jumlah = comm.allreduce(angka, op = MPI.SUM)

# jika saya proses dengan rank 0 maka saya akan menampilkan hasilnya
if rank== 0:
    print('Sum =', jumlah)
```

    b. Output

```
(base) C:\Users\Aqmarina>mpiexec -n 5 python 04.sum_reduce.py
Sum = 290
```

5. Tugas 5 : menghitung nilai pi
    a. Source Code (Pararel)

c: > Users > Nekozawa > Desktop > 05.mpi_pi.py > local_loop

```python
# import mpi4py
from mpi4py import MPI

# buat fungsi dekomposisi bernama local_loop
# local_loop akan menghitung setiap bagiannya
# gunakan 4/(1+x^2), perhatikan batas awal dan akhir untuk dekomposisi
# misalkan size = 4 maka proses 0 menghitung 0-25, proses 1 menghitung 26-50, dst
def local_loop(num_steps,begin,end):
    step = 1.0/num_steps
    sum = 0
    # 4/(1+x^2)
    for i in range(begin,end):
        a = (i+0.5)*step
        sum += 4.0/(1.0+a**2)
    print (sum)
    return sum

# fungsi Pi
def Pi(num_steps):

    # buat COMM
    comm = MPI.COMM_WORLD

    # dapatkan rank proses
    rank = comm.Get_rank()

    # dapatkan total proses berjalan
    total = comm.Get_size()

    # buat variabel baru yang merupakan num_steps/total proses
    var = num_steps/total

    # cari local_sum
    # local_sum merupakan hasil dari memanggil fungsi local_loop
    local_sum = local_loop(num_steps, int(rank*var), int((rank+1)*var))

    # lakukan penjumlahan dari local_sum proses-proses yang ada ke proses 0
    # bisa digunakan reduce atau p2p sum
    sum = comm.allreduce(local_sum, op=MPI.SUM)

    # jika saya proses dengan rank 0  maka tampilkan hasilnya
    if rank == 0:
        pi = sum / num_steps
        print('pi : ',pi)

# panggil fungsi utama
if __name__ == '__main__':
    Pi(10000)
```

    b. Output (Pararel)

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

PS C:\Users\Nekozawa\Desktop> mpiexec -n 5 python .\05.mpi_pi.py
6396.524927233077
4426.288845374425
5372.857677220325
7324.432694244498
7895.822400158931
pi :  3.141592654423125
PS C:\Users\Nekozawa\Desktop>
```

c. Source Code (Serial)

c: > Users > Nekozawa > Desktop > 05.mpi_pi.py > ...

```python
# import mpi4py
from mpi4py import MPI
from serial_pi import Pi

# buat fungsi dekomposisi bernama local_loop
# local_loop akan menghitung setiap bagiannya
# gunakan 4/(1+x^2), perhatikan batas awal dan akhir untuk dekomposisi
# misalkan size = 4 maka proses 0 menghitung 0-25, proses 1 menghitung 26-50, dst
def local_loop(num_steps,begin,end):
    step = 1.0/num_steps
    sum = 0
    # 4/(1+x^2)
    for i in range(begin,end):
        a = (i+0.5)*step
        sum += 4.0/(1.0+a**2)
    print (sum)
    return sum

# fungsi Pi
print(Pi)

# panggil fungsi utama
if __name__ == '__main__':
    Pi(10000)
```

c: > Users > Nekozawa > Desktop > serial_pi.py > ...

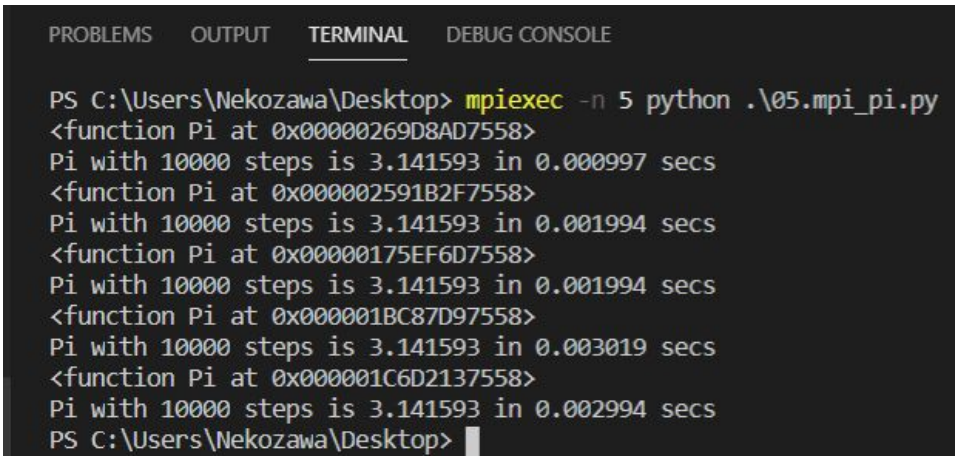
```python
import time

def Pi(num_steps):
    start = time.time()
    step = 1.0/num_steps
    sum = 0
    for i in range(num_steps):
        x= (i+0.5)*step
        sum = sum + 4.0/(1.0+x*x)
    pi = step * sum
    end = time.time()
    print ("Pi with %d steps is %f in %f secs" %(num_steps, pi, end-start))

if __name__ == '__main__':
    Pi(100000)

```

d. Output (Serial)


PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
PS C:\Users\Nekozawa\Desktop> mpiexec -n 5 python .\05.mpi_pi.py
<function Pi at 0x00000269D8AD7558>
Pi with 10000 steps is 3.141593 in 0.000997 secs
<function Pi at 0x000002591B2F7558>
Pi with 10000 steps is 3.141593 in 0.001994 secs
<function Pi at 0x00000175EF6D7558>
Pi with 10000 steps is 3.141593 in 0.001994 secs
<function Pi at 0x000001BC87D97558>
Pi with 10000 steps is 3.141593 in 0.003019 secs
<function Pi at 0x000001C6D2137558>
Pi with 10000 steps is 3.141593 in 0.002994 secs
PS C:\Users\Nekozawa\Desktop>