Assignment #05

Nov, 13, 2017

JinYeong Wang
Parallel Computing Lab.
Mechanical Engineering
Hanyang University

Problem Definition

- Compute

$$\sum_{i=1}^{N} \frac{\sqrt{i}}{N} = \frac{\sqrt{1}}{N} + \frac{\sqrt{2}}{N} + \cdots + \frac{\sqrt{N}}{N}$$

- While

$$N = 10^n$$

- Using
  - Case 1: MPI_Reduce
  - Case 2: MPI_Scatter & MPI_Gather

Algorithm for Case 1. MPI_Reduce

- If rank == 0
  - Input n
  - Compute $N = 10^n$
- MPI_Bcast(N)
- Parallel Sum
- MPI_Reduce

```
int rank, size;
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &size);
int i, width, partialto;
double N, partialsum = 0, totalsum = 0;
time_t Start, End;
if(rank == 0)
{
    int power;
    fflush(stdin);
    printf("It Will Compute N = 10^(power), Input power\n");
    scanf("%d",&power);
    N = pow(10.0,(double)power);
    printf("N = 10^%d = %.1lf\n",power,N);
}
MPI_Bcast(&N,1,MPI_DOUBLE,0,MPI_COMM_WORLD);
width = (int)(N/size);
partialto = (rank+1)*width;
if(rank == size-1)
    partialto = (int)N;
```

Algorithm for Case 1. MPI_Reduce

```
MPI_Barrier(MPI_COMM_WORLD);
if(rank == 0)
    Start = clock();
for(i = rank*width+1 ; i <= partialto ; i++)
    partialsum += sqrt((double)i)/N;
MPI_Reduce(&partialsum,&totalsum,1,MPI_DOUBLE,MPI_SUM,0,MPI_COMM_WORLD);
MPI_Barrier(MPI_COMM_WORLD);
if(rank ==0)
{
    End = clock();
    double compute_time = ((double)(End-Start)/CLOCKS_PER_SEC)*1000;
    printf("Total sum = %lf and Time = %lf ms\n", totalsum, compute_time);
}
MPI_Finalize();
```

1. MPI_Barrier and Time check
2. Parallal sum and MPI_Reduce
3. MPI_Barrier and Time check
    1. Time scale is ms

Algorithm for Case 2. MPI_Scatter & MPI_Gather

- If rank == 0
  - Input n
  - Compute $N = 10^n$
  - Make Vector1 1 to N
- MPI_Bcast(N)
- Make Vector2 for MPI_Scatter
- If rank is size-1
  - Make Vector3 for MPI_Gather
- MPI_Scatter
  - vector1 to vector2
- Parallel Sum
- MPI_Gather
  - vector2 to vector3, root is size-1
- Sum vector3 for result

```
int rank, size;
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &size);
int i, width, partialto, partial_remain;
double N, totalsum = 0, partialsum = 0;
time_t Start, End;
double *vec = NULL;
double *totalvec = NULL;
double *partialvec = NULL;
if(rank == 0)
{
    int power;
    fflush(stdin);
    printf("It Will Compute N = 10^(power), Input power\n");
    scanf("%d",&power);
    N = pow(10.0,(double)power);
    printf("N = 10^%d = %.11f\n",power,N);
    int compute_size = size*((int)(N/size));
    vec = (double *)malloc(compute_size * sizeof(double));
    for(i = 0 ; i < compute_size ; i++)
        vec[i] = i + 1;
}
MPI_Bcast(&N,1,MPI_DOUBLE,0,MPI_COMM_WORLD);
width = (int)(N/size);
partialto = (rank+1)*width;
partialvec = (double*)malloc(width * sizeof(double));
if(rank == size-1)
{
    totalvec = (double *)malloc(size * sizeof(double));
    partial_remain = (int)N - partialto;
}
```

Algorithm for Case 2. MPI_Scatter & MPI_Gather

```
MPI_Barrier(MPI_COMM_WORLD);
if(rank == size-1)
    Start = clock();
MPI_Scatter(vec, width, MPI_DOUBLE, partialvec, width, MPI_DOUBLE, 0, MPI_COMM_WORLD);
for(i = 0 ; i < width ; i++)
    partialsum += sqrt(partialvec[i])/N;
MPI_Gather(&partialsum, 1, MPI_DOUBLE, totalvec, 1, MPI_DOUBLE, size-1, MPI_COMM_WORLD);
if(rank == size-1)
{
    for(i=0;i<size;i++)
        totalsum+=totalvec[i];
    if(partial_remain!=0)
        for(i = partial_remain;i>0;i--)
            totalsum += sqrt((double)(i+partialto))/N;
}
MPI_Barrier(MPI_COMM_WORLD);
if(rank == size-1)
{
    End = clock();
    double compute_time = ((double)(End-Start)/CLOCKS_PER_SEC)*1000;
    printf("Total sum = %lf and Time = %lf ms\n", totalsum, compute_time);
    free(totalvec);
}
if(rank == 0)
    free(vec);
free(partialvec);
MPI_Finalize();
```

Validation Result

Case 1. MPI_Reduce

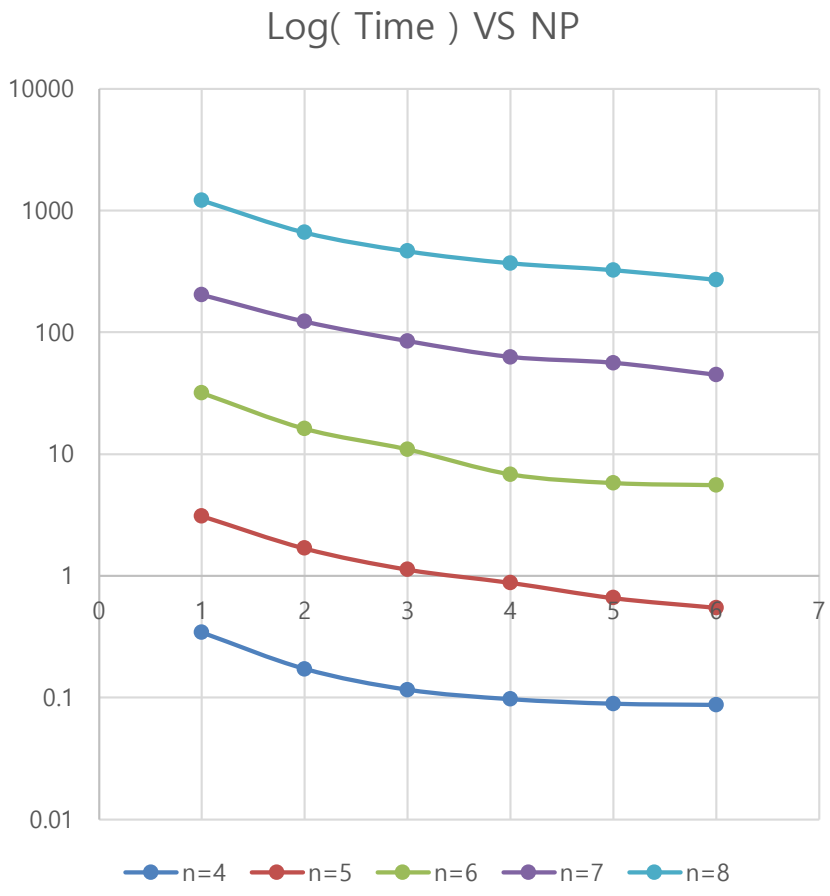
```
root@GAIA:/workspace/MPIJOBS/WJY/mpi04_sum# mpirun -np 1 01.o
It Will Compute N = 10^(power), Input power
4
N = 10^4 = 10000.0
Total sum = 66.671646 and Time = 0.343000 ms
root@GAIA:/workspace/MPIJOBS/WJY/mpi04_sum# mpirun -np 3 01.o
It Will Compute N = 10^(power), Input power
4
N = 10^4 = 10000.0
Total sum = 66.671646 and Time = 0.118000 ms
root@GAIA:/workspace/MPIJOBS/WJY/mpi04_sum# mpirun -np 6 01.o
It Will Compute N = 10^(power), Input power
4
N = 10^4 = 10000.0
Total sum = 66.671646 and Time = 0.103000 ms
root@GAIA:/workspace/MPIJOBS/WJY/mpi04_sum# mpirun -np 1 01.o
It Will Compute N = 10^(power), Input power
5
N = 10^5 = 100000.0
Total sum = 210.820090 and Time = 3.370000 ms
root@GAIA:/workspace/MPIJOBS/WJY/mpi04_sum# mpirun -np 3 01.o
It Will Compute N = 10^(power), Input power
5
N = 10^5 = 100000.0
Total sum = 210.820090 and Time = 1.137000 ms
root@GAIA:/workspace/MPIJOBS/WJY/mpi04_sum# mpirun -np 6 01.o
It Will Compute N = 10^(power), Input power
5
N = 10^5 = 100000.0
Total sum = 210.820090 and Time = 0.208000 ms
```

Case 2. MPI_Scatter & MPI_Gather

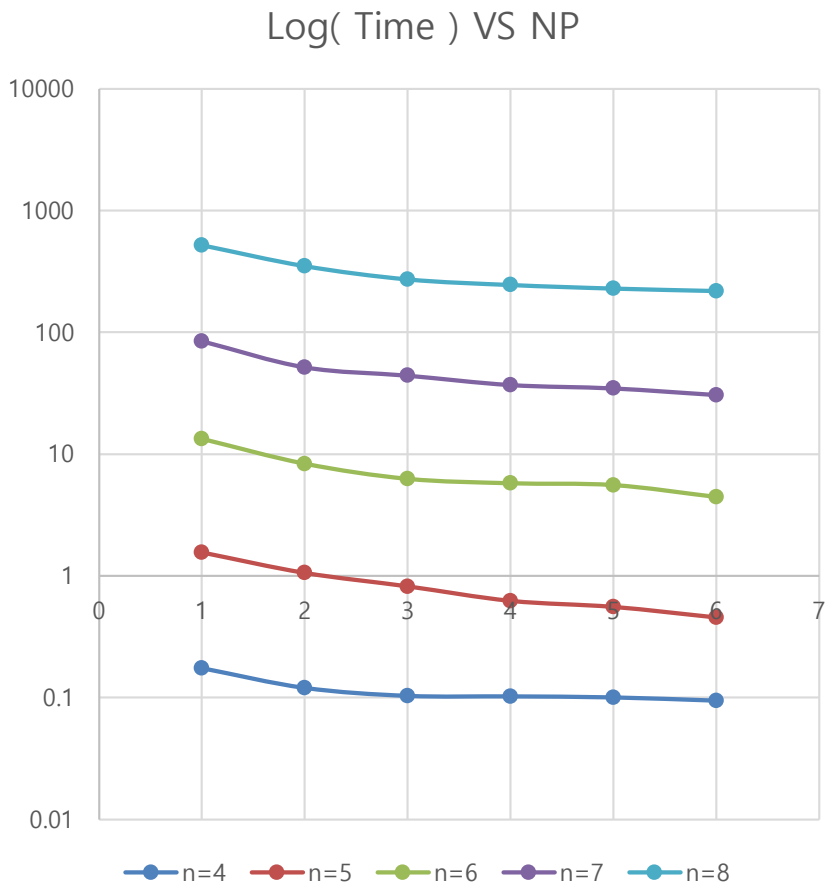
```
root@GAIA:/workspace/MPIJOBS/WJY/mpi04_sum# mpirun -np 1 02.o
It Will Compute N = 10^(power), Input power
4
N = 10^4 = 10000.0
Total sum = 66.671646 and Time = 0.153000 ms
root@GAIA:/workspace/MPIJOBS/WJY/mpi04_sum# mpirun -np 3 02.o
It Will Compute N = 10^(power), Input power
4
N = 10^4 = 10000.0
Total sum = 66.671646 and Time = 0.113000 ms
root@GAIA:/workspace/MPIJOBS/WJY/mpi04_sum# mpirun -np 6 02.o
It Will Compute N = 10^(power), Input power
4
N = 10^4 = 10000.0
Total sum = 66.671646 and Time = 0.084000 ms
root@GAIA:/workspace/MPIJOBS/WJY/mpi04_sum# mpirun -np 1 02.o
It Will Compute N = 10^(power), Input power
5
N = 10^5 = 100000.0
Total sum = 210.820090 and Time = 1.575000 ms
root@GAIA:/workspace/MPIJOBS/WJY/mpi04_sum# mpirun -np 3 02.o
It Will Compute N = 10^(power), Input power
5
N = 10^5 = 100000.0
Total sum = 210.820090 and Time = 0.782000 ms
root@GAIA:/workspace/MPIJOBS/WJY/mpi04_sum# mpirun -np 6 02.o
It Will Compute N = 10^(power), Input power
5
N = 10^5 = 100000.0
Total sum = 210.820090 and Time = 0.445000 ms
```

Compare the result

Case 1. MPI_Reduce

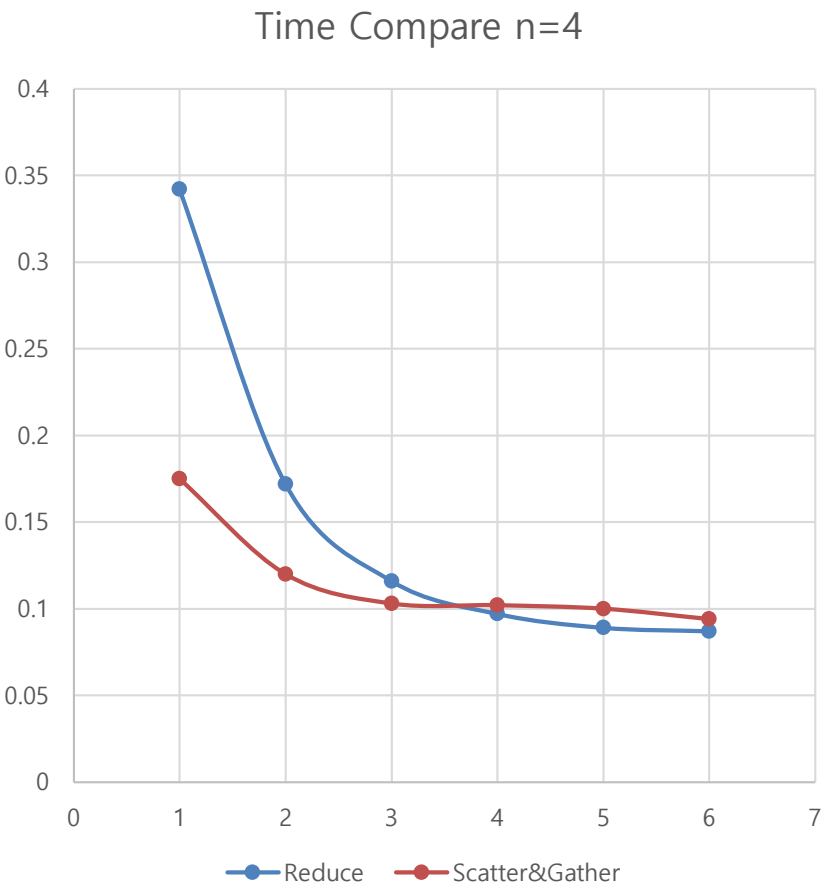


Case 2. MPI_Scatter & MPI_Gather



Compare the result

N = 10^4



N = 10^8