Program Three Report

Wenqing Jiang(wxj150330)

1. Problem statement:
2. Implement Floyd’s Algorithm. using MPI, OpenMP, and OpenACC in combinations
3. Compare the results and get the speedup value.
4. Approach to solution:
5. Language using is C and OpenACC,MPI, OpenMP.
6. Tools used:
7. I use PuTTY to connect to bridges.psc.edu.
8. I use Emacs to edit code.
9. C compiler pgcc and mpicc are used.
10. Design:
11. Connect to bridges.
12. Implement Floyd’s Algorithm. using MPI, OpenMP, and OpenACC in combinations
13. Design and implement test data generation code to be used with the algorithm.
14. Calculate and compare the performance.
15. Solution description:

1. Generate Matrix:

a. Build and Execution

1. pgcc -o genMatrix genMatrix\_floyd.c
2. Usage: genMatrix <n> <outfile>

B. Code:

for(i = 0; i < n; ++i)

for(j = 0; j < n; ++j)

{

if(i==j)

A[i][j] = 0;

else if ((i-j==1)||(j-i==1)||(i==0&& (((j+1)%(n/8))==0))||(j==0&& (((i+1)%(n/8))==0)) )

A[i][j] = 1;

else

A[i][j] = INFTY;

}

1. APSP Serial code:
2. Build and Execution
3. pgcc -o floyd\_serial.out floyd\_serial.c
4. floyd\_serial.out matrix\_2000



b. Code:

void read\_matrix (argv[1], (void \*) &a, (void \*) &storage, &m, &n);

void print\_matrix(int\*\*, int, int);

for (k = 0; k < n; k++)

for (i = 0; i < n; i++)

for (j = 0; j < n; j++)

a[i][j] = MIN(a[i][j], a[i][k] + a[k][j]);

c. Result:

Average performace is less than 1 Gflops.

1. OpenMP implementation:

a. Build and Execution

1. pgcc -mp -o floyd\_omp.out floyd\_omp.c
2. interact –n 28
3. export OMP\_NUM\_THREADS=28
4. floyd\_omp.out matrix\_4000



b. Code:

#pragma omp parallel for private(k,i,j)//Thread this loop

c. Result:

|  |  |
| --- | --- |
| size | Gflops |
| 1000 | 11.6 |
| 2000 | 17.88 |
| 4000 | 18.14 |
| 8000 | 17.66 |

1. OpenACC implementation:
2. Build and Execution
3. pgcc -acc -Minfo=accel -o floyd\_acc.out floyd\_acc.c
4. interact -p GPU-shared --ntasks-per-node=7 --gres=gpu:k80:1
5. floyd\_acc.out matrix\_1000



1. Code:

int \*\*restrict a ; /\* Doubly-subscripted array \*/

for (k = 0; k < n; k++){

#pragma acc parallel loop

for (i = 0; i < n; i++){

for (j = 0; j < n; j++){

a[i][j] = MIN(a[i][j], a[i][k] + a[k][j]);

}

}

}

1. Result:

|  |  |
| --- | --- |
| size | Gflops |
| 800 | 0.34 |
| 1000 | 0.41 |
| 2000 | 0.57 |

1. MPI implementation:
2. Build and Execution
3. module load pgi
4. module load mpi/pgi\_openmpi
5. mpicc floyd\_mpi.c MyMPI.c -o floyd\_mpi.out
6. mpirun -np 28 -mca mpi\_cuda\_support 0 floyd\_mpi.out matrix\_1000



1. Code:

MPI\_Bcast (tmp, n, MPI\_TYPE, root, MPI\_COMM\_WORLD);

for (i = 0; i < BLOCK\_SIZE(id, p, n); i++)

for (j = 0; j < n; j++)

a[i][j] = MIN(a[i][j], a[i][k] + tmp[j]);

1. Result:

|  |  |
| --- | --- |
| size | Gflops |
| 1000 | 19.48 |
| 2000 | 21.09 |
| 4000 | 18.15 |
| 8000 | 17.18 |

1. MPI+OpenMP implementation:
2. Build and Execution

interact -n 28

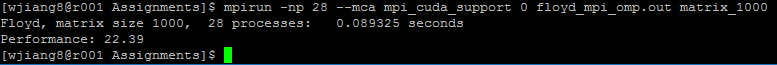
module load pgi

module load mpi/pgi\_openmpi

mpicc -mp -o floyd\_mpi\_omp.out floyd\_mpi\_omp.c MyMPI.c

export OMP\_NUM\_THREADS=28

mpirun -np 28 --mca mpi\_cuda\_support 0 floyd\_mpi\_omp.out matrix\_1000



1. Code:

#pragma omp for schedule(auto)

for (i = 0; i < BLOCK\_SIZE(id, p, n); i++)

for (j = 0; j < n; j++)

a[i][j] = MIN(a[i][j], a[i][k] + tmp[j]);

1. Result:

|  |  |
| --- | --- |
| size | Gflops |
| 1000 | 22.39 |
| 2000 | 23.8 |
| 4000 | 24.46 |
| 8000 | 21.85 |

1. MPI+OpenACC implementation:
2. Build and Execution
3. module load pgi
4. module load mpi/pgi\_openmpi
5. mpicc -acc -o floyd\_mpi\_acc.out floyd\_mpi\_omp.c MyMPI.c
6. mpirun -np 28 --mca mpi\_cuda\_support 0 floyd\_mpi\_acc.out matrix\_1000



1. Code

#pragma acc parallel loop

for (i = 0; i < BLOCK\_SIZE(id, p, n); i++)

for (j = 0; j < n; j++)

a[i][j] = MIN(a[i][j], a[i][k] + tmp[j]);

1. Result

|  |  |
| --- | --- |
| size | Gflops |
| 1000 | 29.42 |
| 2000 | 34.01 |
| 4000 | 36.46 |
| 8000 | 24.4 |

1. Conclusion:

For 2000x2000 Matrix:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | serial | OpenMP | OpenACC | MPI | MPI+OpenMP | MPI+OpenACC |
| Gflops | 0.94 | 17.88 | 0.57 | 21.09 | 23.80 | 34.01 |
| speedup | 1.00 | 19.02 | 0.61 | 22.44 | 25.32 | 36.18 |