

Hybrid Programming

John Urbanic

Parallel Computing Specialist
Pittsburgh Supercomputing Center

Hybrid OpenACC Programming (Fast & Wrong)

```
#pragma acc data copy(Temperature_last), create(Temperature)
while ( dt_global > MAX_TEMP_ERROR && iteration <= max_iterations ) {

    #pragma acc kernels
    for(i = 1; i <= ROWS; i++) {
        for(j = 1; j <= COLUMNS; j++) {
            Temperature[i][j] = 0.25 * (Temperature_last[i+1][j] + Temperature_last[i-1][j] +
                                         Temperature_last[i][j+1] + Temperature_last[i][j-1]);
        }
    }

    if(my_PE_num != npes-1){
        MPI_Send(&Temperature[ROWS][1], COLUMNS, MPI_DOUBLE, my_PE_num+1, DOWN, MPI_COMM_WORLD);
    }

    if(my_PE_num != 0){
        MPI_Recv(&Temperature_last[0][1], COLUMNS, MPI_DOUBLE, my_PE_num-1, DOWN, MPI_COMM_WORLD, &status);
    }

    if(my_PE_num != 0){
        MPI_Send(&Temperature[1][1], COLUMNS, MPI_DOUBLE, my_PE_num-1, UP, MPI_COMM_WORLD);
    }

    if(my_PE_num != npes-1){
        MPI_Recv(&Temperature_last[ROWS+1][1], COLUMNS, MPI_DOUBLE, my_PE_num+1, UP, MPI_COMM_WORLD, &status);
    }

    dt = 0.0;

    #pragma acc kernels
    for(i = 1; i <= ROWS; i++){
        for(j = 1; j <= COLUMNS; j++){
            dt = fmax( fabs(Temperature[i][j]-Temperature_last[i][j]), dt);
            Temperature_last[i][j] = Temperature[i][j];
        }
    }

    MPI_Reduce(&dt, &dt_global, 1, MPI_DOUBLE, MPI_MAX, 0, MPI_COMM_WORLD);
    MPI_Bcast(&dt_global, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);

    if((iteration % 100) == 0) {
        if (my_PE_num == npes-1){
            #pragma acc update host(Temperature)
            track_progress(iteration);
        }
    }

    iteration++;
}
```

MPI
routines
using
host
data

0.9s

Hybrid OpenACC Programming (Slow and Right)

```
#pragma acc data copy(Temperature_last), create(Temperature)
while ( dt_global > MAX_TEMP_ERROR && iteration <= max_iterations ) {

    #pragma acc kernels
    for(i = 1; i <= ROWS; i++) {
        for(j = 1; j <= COLUMNS; j++) {
            Temperature[i][j] = 0.25 * (Temperature_last[i+1][j] + Temperature_last[i-1][j] +
            Temperature_last[i][j+1] + Temperature_last[i][j-1]);
        }
    }

    #pragma acc update host(Temperature, Temperature_last)

    if(my_PE_num != npes-1){
        MPI_Send(&Temperature[ROWS][1], COLUMNS, MPI_DOUBLE, my_PE_num+1, DOWN, MPI_COMM_WORLD);
    }

    if(my_PE_num != 0){
        MPI_Recv(&Temperature_last[0][1], COLUMNS, MPI_DOUBLE, my_PE_num-1, DOWN, MPI_COMM_WORLD, &status);
    }

    if(my_PE_num != 0){
        MPI_Send(&Temperature[1][1], COLUMNS, MPI_DOUBLE, my_PE_num-1, UP, MPI_COMM_WORLD);
    }

    if(my_PE_num != npes-1){
        MPI_Recv(&Temperature_last[ROWS+1][1], COLUMNS, MPI_DOUBLE, my_PE_num+1, UP, MPI_COMM_WORLD, &status);
    }

    #pragma acc update device(Temperature, Temperature_last)

    dt = 0.0;

    #pragma acc kernels
    for(i = 1; i <= ROWS; i++){
        for(j = 1; j <= COLUMNS; j++){
            dt = fmax( fabs(Temperature[i][j]-Temperature_last[i][j]), dt);
            Temperature_last[i][j] = Temperature[i][j];
        }
    }

    MPI_Reduce(&dt, &dt_global, 1, MPI_DOUBLE, MPI_MAX, 0, MPI_COMM_WORLD);
    MPI_Bcast(&dt_global, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);

    if((iteration % 100) == 0) {
        if (my_PE_num == npes-1){
            #pragma acc update host(Temperature)
            track_progress(iteration);
        }
    }

    iteration++;
}
```

Update
data
entering
and
leaving
MPI
section

9.3 s

```
#pragma acc data copy(Temperature_last), create(Temperature)
while ( dt_global > MAX_TEMP_ERROR && iteration <= max_iterations ) {

    #pragma acc kernels
    for(i = 1; i <= ROWS; i++) {
        for(j = 1; j <= COLUMNS; j++) {
            Temperature[i][j] = 0.25 * (Temperature_last[i+1][j] + Temperature_last[i-1][j] +
            Temperature_last[i][j+1] + Temperature_last[i][j-1]);
        }
    }

    #pragma acc update host(Temperature[1:1][1:COLUMNS], Temperature[ROWS:1][1:COLUMNS])
```

```
    if(my_PE_num != npes-1){
        MPI_Send(&Temperature[ROWS][1], COLUMNS, MPI_DOUBLE, my_PE_num+1, DOWN, MPI_COMM_WORLD);
    }

    if(my_PE_num != 0){
        MPI_Recv(&Temperature_last[0][1], COLUMNS, MPI_DOUBLE, my_PE_num-1, DOWN, MPI_COMM_WORLD, &status);
    }

    if(my_PE_num != 0){
        MPI_Send(&Temperature[1][1], COLUMNS, MPI_DOUBLE, my_PE_num-1, UP, MPI_COMM_WORLD);
    }

    if(my_PE_num != npes-1){
        MPI_Recv(&Temperature_last[ROWS+1][1], COLUMNS, MPI_DOUBLE, my_PE_num+1, UP, MPI_COMM_WORLD, &status);
    }
}
```

```
#pragma acc update device(Temperature_last[0:1][1:COLUMNS], Temperature_last[ROWS+1:1][1:COLUMNS])
```

```
dt = 0.0;
```

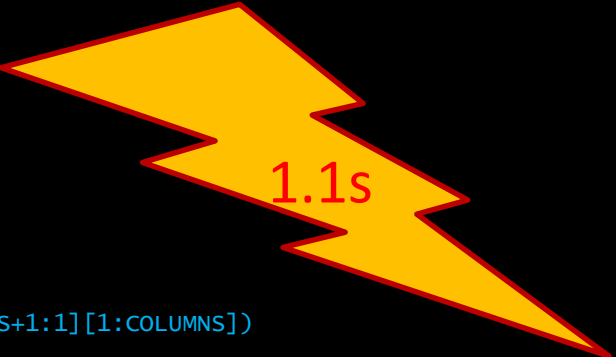
```
#pragma acc kernels
for(i = 1; i <= ROWS; i++){
    for(j = 1; j <= COLUMNS; j++){
        dt = fmax( fabs(Temperature[i][j]-Temperature_last[i][j]), dt);
        Temperature_last[i][j] = Temperature[i][j];
    }
}
```

```
MPI_Reduce(&dt, &dt_global, 1, MPI_DOUBLE, MPI_MAX, 0, MPI_COMM_WORLD);
MPI_Bcast(&dt_global, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
```

```
if((iteration % 100) == 0) {
    if (my_PE_num == npes-1){
        #pragma acc update host(Temperature)
        track_progress(iteration);
    }
}
```

```
iteration++;
```

```
}
```



1.1s

Mix and Match

- PGI Compile:

```
mpicc -acc laplace_hybrid.c  
mpf90 -acc laplace_hybrid.f90  
mpicc -mp -acc laplace_hybrid.c  
etc...
```

- Running:

```
interact ?  
-n 4  
-N1 -n4  
-p GPU -N1 -n4  
-p GPU -N4 -n4  
-N1 -n28  
-N4 -n112  
etc...
```

- Intel bonus detail:

```
export I_MPI_PIN_DOMAIN=omp      (or you may not actually get multiple cores!)  
Details at https://software.intel.com/en-us/articles/hybrid-applications-intelmpi-openmp
```

Once again...

