

# HandsOnGPUProgramming\_Solution

November 8, 2019

## 1 Hands-On GPU Programming

*Supercomputing 2019 Tutorial “Application Porting and Optimization on GPU-Accelerated POWER Architectures”, November 18th 2019*

---

### 1.1 Solutions

**This contains the output for the solutions.**

The solutions are described in the solution section. The directory links to the solution source files should work though. For the *html* and *pdf* versions please navigate to the corresponding directory to find the solution profiles and sources.

#### 1.1.1 GPU Programming

- Section ?? Accelerate a CPU Jacobi solver with OpenACC relying on Unified Memory for data movement using `-ta=tesla:managed`
- Section ?? Fix memory access pattern of OpenACC accelerated Jacobi Solver

#### 1.1.2 Multi-GPU with MPI

- Section ?? Use MPI to make OpenACC accelerated Jacobi Solver scale to multiple GPUs
- Section ?? Hide MPI communication time by overlapping communication and computation in a MPI+OpenACC multi GPU Jacobi Solver

#### 1.1.3 Multi-GPU with NVSHMEM (*Advanced – C only*)

- Section ?? Use NVSHMEM instead of MPI
- Section ?? Put NVSHMEM calls on stream to hide API calls and GPU/CPU synchronization

#### 1.1.4 Survey

- Section 3 Please remember to take the survey !

## 1.2 Setup

Please **select your language choice (C or FORTRAN)** below by making sure your choice is uncommented and comment out the other language. Then execute the cell by hitting **Shift+Enter**!

```
[1]: # select language here
LANGUAGE='C'
#LANGUAGE='FORTRAN'

## You should not touch the remaining code in the cell
import os.path
import pandas

try: rootdir
except NameError: rootdir = None
if(not rootdir):
    rootdir=%pwd
basedir=os.path.join(rootdir,LANGUAGE)
basedirC=os.path.join(rootdir,'C')

print ("You selected {} for the exercises.".format(LANGUAGE))

def checkdir(dir):
    d=%pwd
    assert(d.endswith(dir) or d.endswith(dir+'p') or d.endswith(dir+'m')),_
    ↪ "Please make sure to cd to the right directory first."

def cleanall():
    # clean up everything -- use with care
    for t in range(4):
        d='%s/task%i'%(basedir,t)
        %cd $d
        !make clean

#cleanall()
```

You selected C for the exercises.

```
[2]: %cd $basedir/task0
```

```
/autofs/nccsopen-svm1_home/mathiasw/sc19-tutorial-
openpower/4-GPU/HandsOn/Solution/C/task0
```

## 2 Solutions

Below are suggested solutions. This is only a short description of the solution, but the `poisson2d.solution.(c|F03)` files linked below have the full source code. If you want to run / profile the solutions feel free to duplicate the cells for the tasks and change the Section ?? to the

\*.solution ones.

Section ??

---

## 2.1 Solution 0:

```
#pragma acc parallel loop
for (int ix = ix_start; ix < ix_end; ix++)
{
    #pragma acc loop
    for( int iy = iy_start; iy < iy_end; iy++ )
    {
        Anew[iy*nx+ix] = -0.25 * (rhs[iy*nx+ix] - ( A[iy*nx+ix+1] + A[iy*nx+ix-1]
                                                    + A[(iy-1)*nx+ix] + A[(iy+1)*nx+ix] ));
        error = fmaxr( error, fabsr(Anew[iy*nx+ix]-A[iy*nx+ix]));
    }
}
```

Code

- [C Version](#)
- [Fortran Version](#)

**Compiling, Running and Profiling** You can compile, run and profile the solution with the next cells. **After** the profiling finished the output file `poisson2d.solution.pgprof` can be downloaded from here: [C Version](#) / [Fortran Version](#).

```
[3]: %cd $basedir/task0
```

```
/autofs/nccsopen-svm1_home/mathiasw/sc19-tutorial-
openpower/4-GPU/HandsOn/Solution/C/task0
```

```
[4]: checkdir('task0')
      !make poisson2d.solution
```

```
pgcc -c -DUSE_DOUBLE -Minfo=accel -fast -acc -ta=tesla:cc70,managed
poisson2d_serial.c -o poisson2d_serial.o
pgcc -DUSE_DOUBLE -Minfo=accel -fast -acc -ta=tesla:cc70,managed
poisson2d.solution.c poisson2d_serial.o -o poisson2d.solution
poisson2d.solution.c:
main:
    66, Generating Tesla code
    67, #pragma acc loop gang /* blockIdx.x */
    68, #pragma acc loop vector(128) /* threadIdx.x */
    66, Generating implicit copyout(A[:])
    68, Loop is parallelizable
    88, Generating Tesla code
    89, #pragma acc loop gang /* blockIdx.x */
```

```

    90, #pragma acc loop vector(128) /* threadIdx.x */
    94, Generating implicit reduction(max:error)
88, Generating implicit copyin(A[:,rhs[:]])
    Generating implicit copyout(Anew[:])
90, Loop is parallelizable
98, Generating Tesla code
    99, #pragma acc loop gang /* blockIdx.x */
    100, #pragma acc loop vector(128) /* threadIdx.x */
98, Generating implicit copyin(Anew[:])
    Generating implicit copyout(A[:])
100, Loop is parallelizable
106, Generating Tesla code
    107, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
106, Generating implicit copyin(A[:])
    Generating implicit copyout(A[nx*(ny-1)+1:2046])
111, Generating Tesla code
    112, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
111, Generating implicit copy(A[:])

```

```

[5]: checkdir('task0')
    !make run.solution

```

```

bsub -W 60 -nnodes 1 -Is -P TRN003 jsrun -n 1 -c 1 -g ALL_GPUS
./poisson2d.solution
Job <25189> is submitted to default queue <batch>.
<<Waiting for dispatch ...>>
<<Starting on login1>>
Jacobi relaxation Calculation: 2048 x 2048 mesh
Calculate reference solution and time serial CPU execution.
    0, 0.249999
    100, 0.249760
    200, 0.249522
    300, 0.249285
    400, 0.249048
GPU execution.
    0, 0.249999
    100, 0.249760
    200, 0.249522
    300, 0.249285
    400, 0.249048
2048x2048: 1 CPU:   5.4684 s, 1 GPU:   0.1884 s, speedup:   29.02

```

```

[6]: checkdir('task0')
    !make profile.solution

```

```

bsub -W 60 -nnodes 1 -Is -P TRN003 jsrun -n 1 -c 1 -g ALL_GPUS pgprof -f --cpu-
profiling off --openmp-profiling off -o
/gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.pgprof

```

```

./poisson2d.solution 10
Job <25190> is submitted to default queue <batch>.
<<Waiting for dispatch ...>>
<<Starting on login1>>
==91820== PGPROF is profiling process 91820, command: ./poisson2d.solution 10
==91820== Generated result file:
/gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.pgprof
Jacobi relaxation Calculation: 2048 x 2048 mesh
Calculate reference solution and time serial CPU execution.
    0, 0.249999
GPU execution.
    0, 0.249999
2048x2048: 1 CPU:   0.1230 s, 1 GPU:   0.0189 s, speedup:    6.51
mv /gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.pgprof .
Section ??

```

---

## 2.2 Solution 1:

Swap the ix and iy loops to make sure that ix is the fastest running index

```

#pragma acc parallel loop
for (int iy = iy_start; iy < iy_end; iy++)
{
    for( int ix = ix_start; ix < ix_end; ix++ )
    {
        Anew[iy*nx+ix] = -0.25 * (rhs[iy*nx+ix] - ( A[iy*nx+ix+1] + A[iy*nx+ix-1]
                                                    + A[(iy-1)*nx+ix] + A[(iy+1)*nx+ix] ));
        error = fmaxr( error, fabsr(Anew[iy*nx+ix]-A[iy*nx+ix]));
    }
}

```

### Code

- [C Version](#)
- [Fortran Version](#)

**Compiling, Running and Profiling** You can compile, run and profile the solution with the next cells. **After** the profiling finished the output file `poisson2d.solution.pgprof` can be downloaded from here: [C Version](#) / [Fortran Version](#).

```
[7]: %cd $basedir/task1
```

```

/autofs/nccsopen-svm1_home/mathiasw/sc19-tutorial-
openpower/4-GPU/HandsOn/Solution/C/task1

```

```

[8]: checkdir('task1')
!make poisson2d.solution

```

```
pgcc -c -DUSE_DOUBLE -Minfo=accel -fast -acc -ta=tesla:cc70,managed,lineinfo
poisson2d_serial.c -o poisson2d_serial.o
pgcc -DUSE_DOUBLE -Minfo=accel -fast -acc -ta=tesla:cc70,managed,lineinfo
poisson2d.solution.c poisson2d_serial.o -o poisson2d.solution
poisson2d.solution.c:
```

main:

```
66, Generating Tesla code
67, #pragma acc loop gang /* blockIdx.x */
68, #pragma acc loop vector(128) /* threadIdx.x */
66, Generating implicit copyout(A[:])
68, Loop is parallelizable
88, Generating Tesla code
89, #pragma acc loop gang /* blockIdx.x */
90, #pragma acc loop vector(128) /* threadIdx.x */
94, Generating implicit reduction(max:error)
88, Generating implicit copyin(A[:],rhs[:])
Generating implicit copyout(Anew[:])
90, Loop is parallelizable
98, Generating Tesla code
99, #pragma acc loop gang /* blockIdx.x */
100, #pragma acc loop vector(128) /* threadIdx.x */
98, Generating implicit copyin(Anew[:])
Generating implicit copyout(A[:])
100, Loop is parallelizable
106, Generating Tesla code
107, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
106, Generating implicit copyin(A[:])
Generating implicit copyout(A[nx*(ny-1)+1:2046])
111, Generating Tesla code
112, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
111, Generating implicit copy(A[:])
```

```
[9]: checkdir('task1')
!make run.solution
```

```
bsub -W 60 -nnodes 1 -Is -P TRN003 jsrun -n 1 -c 1 -g ALL_GPUS
./poisson2d.solution
Job <25191> is submitted to default queue <batch>.
<<Waiting for dispatch ...>>
<<Starting on login1>>
Jacobi relaxation Calculation: 2048 x 2048 mesh
Calculate reference solution and time serial CPU execution.
0, 0.249999
100, 0.249760
200, 0.249522
300, 0.249285
400, 0.249048
GPU execution.
```

```

    0, 0.249999
   100, 0.249760
   200, 0.249522
   300, 0.249285
   400, 0.249048
2048x2048: 1 CPU:   5.4691 s, 1 GPU:   0.1866 s, speedup:   29.31

```

```

[10]: checkdir('task1')
      !make profile.solution

```

```

bsub -W 60 -nnodes 1 -Is -P TRN003 jsrun -n 1 -c 1 -g ALL_GPUS pgprof -f --cpu-
profiling off --openmp-profiling off -o
/gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.timeline.pgprof
./poisson2d.solution 3
Job <25192> is submitted to default queue <batch>.
<<Waiting for dispatch ...>>
<<Starting on login1>>
==92054== PGPROF is profiling process 92054, command: ./poisson2d.solution 3
==92054== Generated result file:
/gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.timeline.pgprof
Jacobi relaxation Calculation: 2048 x 2048 mesh
Calculate reference solution and time serial CPU execution.
    0, 0.249999
GPU execution.
    0, 0.249999
2048x2048: 1 CPU:   0.0465 s, 1 GPU:   0.0154 s, speedup:   3.01
bsub -W 60 -nnodes 1 -Is -P TRN003 jsrun -n 1 -c 1 -g ALL_GPUS pgprof -f --cpu-
profiling off --openmp-profiling off --analysis-metrics -o
/gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.metrics.pgprof
./poisson2d.solution 3
Job <25193> is submitted to default queue <batch>.
<<Waiting for dispatch ...>>
<<Starting on login1>>
==71647== PGPROF is profiling process 71647, command: ./poisson2d.solution 3
==71647== Some kernel(s) will be replayed on device 0 in order to collect all
events/metrics.
==71647== Generated result file:
/gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.metrics.pgprof
Jacobi relaxation Calculation: 2048 x 2048 mesh
Calculate reference solution and time serial CPU execution.
    0, 0.249999
GPU execution.
    0, 0.249999
2048x2048: 1 CPU:   0.0476 s, 1 GPU:  12.4561 s, speedup:   0.00
bsub -W 60 -nnodes 1 -Is -P TRN003 jsrun -n 1 -c 1 -g ALL_GPUS pgprof -f --cpu-
profiling off --openmp-profiling off --metrics gld_efficiency,gst_efficiency -o
/gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.efficiency.pgprof
./poisson2d.solution 3

```

```

Job <25194> is submitted to default queue <batch>.
<<Waiting for dispatch ...>>
<<Starting on login1>>
==92292== PGPROF is profiling process 92292, command: ./poisson2d.solution 3
==92292== Some kernel(s) will be replayed on device 0 in order to collect all
events/metrics.
==92292== Generated result file:
/gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.efficiency.pgprof
Jacobi relaxation Calculation: 2048 x 2048 mesh
Calculate reference solution and time serial CPU execution.
    0, 0.249999
GPU execution.
    0, 0.249999
2048x2048: 1 CPU:  0.0487 s, 1 GPU:  0.6897 s, speedup:      0.07
pgprof --csv -i
/gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.efficiency.pgprof 2>&1 |
grep -v "=====" > poisson2d.solution.efficiency.csv
mv /gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.*.pgprof .
tar -cvzf pgprof.poisson2d.Task1.solution.tar.gz  poisson2d.solution.*.pgprof
poisson2d.solution.efficiency.pgprof
poisson2d.solution.metrics.pgprof
poisson2d.solution.timeline.pgprof

For the Global Memory Load/Store Efficiency the make profile command also generated a CSV
file that you can import and view with the cell below.
If you purely work in a terminal you can view the same output by running pgprof -i
poisson2d.efficiency.solution.pgprof.

```

```

[11]: data_frame_solution = pandas.read_csv('poisson2d.solution.efficiency.csv',
      ↪sep=',')
data_frame_solution

```

```

[11]:

```

	Device	Kernel	Invocations	Metric Name \
0	Tesla V100-SXM2-16GB (0)	main_98_gpu	3	gld_efficiency
1	Tesla V100-SXM2-16GB (0)	main_98_gpu	3	gst_efficiency
2	Tesla V100-SXM2-16GB (0)	main_106_gpu	3	gld_efficiency
3	Tesla V100-SXM2-16GB (0)	main_106_gpu	3	gst_efficiency
4	Tesla V100-SXM2-16GB (0)	main_94_gpu__red	3	gld_efficiency
5	Tesla V100-SXM2-16GB (0)	main_94_gpu__red	3	gst_efficiency
6	Tesla V100-SXM2-16GB (0)	main_66_gpu	1	gld_efficiency
7	Tesla V100-SXM2-16GB (0)	main_66_gpu	1	gst_efficiency
8	Tesla V100-SXM2-16GB (0)	main_88_gpu	3	gld_efficiency
9	Tesla V100-SXM2-16GB (0)	main_88_gpu	3	gst_efficiency
10	Tesla V100-SXM2-16GB (0)	main_111_gpu	3	gld_efficiency
11	Tesla V100-SXM2-16GB (0)	main_111_gpu	3	gst_efficiency

	Metric Description	Min	Max	Avg
0	Global Memory Load Efficiency	90.868353%	90.896134%	90.881874%



1	Global Memory Store Efficiency	88.956522%	88.956522%	88.956522%
2	Global Memory Load Efficiency	94.722222%	94.722222%	94.722222%
3	Global Memory Store Efficiency	88.956522%	88.956522%	88.956522%
4	Global Memory Load Efficiency	99.756335%	99.756335%	99.756335%
5	Global Memory Store Efficiency	25.000000%	25.000000%	25.000000%
6	Global Memory Load Efficiency	0.000000%	0.000000%	0.000000%
7	Global Memory Store Efficiency	100.000000%	100.000000%	100.000000%
8	Global Memory Load Efficiency	91.834032%	91.855433%	91.843628%
9	Global Memory Store Efficiency	88.845486%	88.845486%	88.845486%
10	Global Memory Load Efficiency	25.000000%	25.000000%	25.000000%
11	Global Memory Store Efficiency	25.000000%	25.000000%	25.000000%

Section ??

## 2.3 Solution 2:

Set the GPU used by the rank using `#pragma acc set device_num`

*//Initialize MPI and determine rank and size*

```
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &size);
```

```
#pragma acc set device_num( rank )
```

```
real* restrict const A      = (real*) malloc(nx*ny*sizeof(real));
real* restrict const Aref   = (real*) malloc(nx*ny*sizeof(real));
real* restrict const Anew   = (real*) malloc(nx*ny*sizeof(real));
real* restrict const rhs    = (real*) malloc(nx*ny*sizeof(real));
```

Apply domain decomposition

```
// Ensure correctness if ny%size != 0
int chunk_size = ceil( (1.0*ny)/size );
```

```
int iy_start = rank * chunk_size;
int iy_end   = iy_start + chunk_size;
```

*// Do not process boundaries*

```
iy_start = max( iy_start, 1 );
iy_end = min( iy_end, ny - 1 );
```

Exchange data

*//Periodic boundary conditions*

```
int top      = (rank == 0) ? (size-1) : rank-1;
int bottom = (rank == (size-1)) ? 0 : rank+1;
#pragma acc host_data use_device( A )
{
```

```

double start_mpi = MPI_Wtime();
//1. Sent row iy_start (first modified row) to top receive lower boundary (iy_end) from bo
MPI_Sendrecv( A+iy_start*nx+ix_start, (ix_end-ix_start), MPI_REAL_TYPE, top, 0,
              A+iy_end*nx+ix_start, (ix_end-ix_start), MPI_REAL_TYPE, bottom, 0,
              MPI_COMM_WORLD, MPI_STATUS_IGNORE );

//2. Sent row (iy_end-1) (last modified row) to bottom receive upper boundary (iy_start-1)
MPI_Sendrecv( A+(iy_end-1)*nx+ix_start, (ix_end-ix_start), MPI_REAL_TYPE, bottom, 0,
              A+(iy_start-1)*nx+ix_start, (ix_end-ix_start), MPI_REAL_TYPE, top, 0,
              MPI_COMM_WORLD, MPI_STATUS_IGNORE );
mpi_time += MPI_Wtime() - start_mpi;
}

```

## Code

- [C Version](#)
- [Fortran Version](#)

**Compiling, Running and Profiling** You can compile, run and profile the solution with the next cells. You can profile the code by executing the next cell. **After** the profiling completed download the tarball containing the profiles (`pgprof.Task2.solution.poisson2d.tar.gz`) with the File Browser. Then you can import them into pgprof / nvvp using the *Import* option in the *File* menu. Remember to use the *Multiple processes* option in the assistant.

```
[12]: %cd $basedir/task2
```

```

/autofs/nccsopen-svm1_home/mathiasw/sc19-tutorial-
openpower/4-GPU/HandsOn/Solution/C/task2

```

```
[13]: checkdir('task2')
!make poisson2d.solution
```

```

mpicc -c -DUSE_DOUBLE -Minfo=accel -fast -acc -ta=tesla:cc70,pinned
poisson2d_serial.c -o poisson2d_serial.o
poisson2d_serial:
    36, Generating present(Anew[:],rhs[:],Aref[:])
    39, Generating update device(rhs[:ny*nx],Aref[:ny*nx])
    42, Generating Tesla code
    43, #pragma acc loop gang /* blockIdx.x */
    44, #pragma acc loop vector(128) /* threadIdx.x */
    49, Generating implicit reduction(max:error)
    44, Loop is parallelizable
    53, Generating Tesla code
    54, #pragma acc loop gang /* blockIdx.x */
    55, #pragma acc loop vector(128) /* threadIdx.x */
    55, Loop is parallelizable
    61, Generating Tesla code
    62, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */

```

```

        66, Generating Tesla code
        67, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
        78, Generating update self(Aref[:ny*nx])
mpicc -DUSE_DOUBLE -Minfo=accel -fast -acc -ta=tesla:cc70,pinned
poisson2d.solution.c poisson2d_serial.o -o poisson2d.solution
poisson2d.solution.c:
main:
    71, Generating enter data
create(Aref[:ny*nx],rhs[:ny*nx],A[:ny*nx],Anew[:ny*nx])
    87, Generating present(Aref[:],A[:])
        Generating Tesla code
        88, #pragma acc loop gang /* blockIdx.x */
        89, #pragma acc loop vector(128) /* threadIdx.x */
    89, Loop is parallelizable
    140, Generating update device(A[nx*(iy_start-1):nx*((iy_end-
iy_start)+2)],rhs[nx*iy_start:nx*(iy_end-iy_start)])
    143, Generating present(A[:],rhs[:],Anew[:])
        Generating Tesla code
        144, #pragma acc loop gang /* blockIdx.x */
        145, #pragma acc loop vector(128) /* threadIdx.x */
        149, Generating implicit reduction(max:error)
    145, Loop is parallelizable
    157, Generating present(Anew[:],A[:])
        Generating Tesla code
        158, #pragma acc loop gang /* blockIdx.x */
        159, #pragma acc loop vector(128) /* threadIdx.x */
    159, Loop is parallelizable
    184, Generating present(A[:])
        Generating Tesla code
        185, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
    195, Generating update self(A[nx*(iy_start-1):nx*((iy_end-iy_start)+2)])
    213, Generating exit data delete(rhs[:1],Aref[:1],A[:1],Anew[:1])

```

```

[14]: checkdir('task2')
      !NP=2 make run.solution

```

```

bsub -W 60 -nnodes 1 -Is -P TRN003 jsrun -n 1 -c 1 -g ALL_GPUS -a 2 -c ALL_CPUS
-d cyclic -b packed:7 --smpiargs "-gpu" ./poisson2d.solution
Job <25195> is submitted to default queue <batch>.
<<Waiting for dispatch ...>>
<<Starting on login1>>
Jacobi relaxation Calculation: 4096 x 4096 mesh
Calculate reference solution and time serial execution.
    0, 0.250000
   100, 0.249940
   200, 0.249880
   300, 0.249821
   400, 0.249761

```

```

500, 0.249702
600, 0.249642
700, 0.249583
800, 0.249524
900, 0.249464
Parallel execution.
  0, 0.250000
100, 0.249940
200, 0.249880
300, 0.249821
400, 0.249761
500, 0.249702
600, 0.249642
700, 0.249583
800, 0.249524
900, 0.249464
Num GPUs: 2.
4096x4096: 1 GPU:   1.3165 s, 2 GPUs:   0.7221 s, speedup:   1.82, efficiency:
91.17%
MPI time:   0.0422 s, inter GPU BW:   2.89 GiB/s

```

```

[15]: checkdir('task2')
      !NP=2 make profile.solution

```

```

bsub -W 60 -nnodes 1 -Is -P TRN003 jsrun -n 1 -c 1 -g ALL_GPUS -a 2 -c ALL_CPUS
-d cyclic -b packed:7 --smpiargs "-gpu" pgprof -f --cpu-profiling off --openmp-
profiling off --annotate-mpi openmpi -o /gpfs/wolf/trn003/scratch/mathiasw//pois
son2d.solution.Task2.NP2.%q{OMPI_COMM_WORLD_RANK}.pgprof ./poisson2d.solution 10
Job <25196> is submitted to default queue <batch>.
<<Waiting for dispatch ...>>
<<Starting on login1>>
==92521== PGPROF is profiling process 92521, command: ./poisson2d.solution 10
==92520== PGPROF is profiling process 92520, command: ./poisson2d.solution 10
==92520== Generated result file:
/gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.Task2.NP2.1.pgprof
Jacobi relaxation Calculation: 4096 x 4096 mesh
Calculate reference solution and time serial execution.
  0, 0.250000
Parallel execution.
  0, 0.250000
Num GPUs: 2.
4096x4096: 1 GPU:   0.0224 s, 2 GPUs:   0.0130 s, speedup:   1.73, efficiency:
86.37%
MPI time:   0.0007 s, inter GPU BW:   1.75 GiB/s
==92521== Generated result file:
/gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.Task2.NP2.0.pgprof
mv /gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.Task2.NP2.?.pgprof .
tar -cvzf pgprof.poisson2d.Task2.solution.tar.gz

```

```
poisson2d.solution.Task2.NP2.?.pgprof
poisson2d.solution.Task2.NP2.0.pgprof
poisson2d.solution.Task2.NP2.1.pgprof
```

**Scaling** You can do a simple scaling run for up to all 6 GPUs in the node by executing the next cell.

```
[16]: checkdir('task2')
!NP=1 make run.solution | grep speedup > scale.out
!NP=2 make run.solution | grep speedup >> scale.out
!NP=4 make run.solution | grep speedup >> scale.out
!NP=6 make run.solution | grep speedup >> scale.out
data_frameS2 = pandas.read_csv('scale.out', delim_whitespace=True, header=None)

!rm scale.out

data_frameS2b=data_frameS2.iloc[:,[5,7,10,12]].copy()
data_frameS2b.rename(columns={5:'GPUs', 7: 'time [s]', 10:'speedup', 12:
    ↳'efficiency'})
```

```
<<Waiting for dispatch ...>>
<<Starting on login1>>
<<Waiting for dispatch ...>>
<<Starting on login1>>
<<Waiting for dispatch ...>>
<<Starting on login1>>
<<Waiting for dispatch ...>>
<<Starting on login1>>
```

```
[16]:   GPUs  time [s] speedup efficiency
      0     1    1.4201   0.93,    92.67%
      1     2    0.7157   1.83,    91.44%
      2     4    0.4301   3.08,    76.91%
      3     6    0.3037   4.32,    71.94%
```

Section ??

---

## 2.4 Solution 3:

Update the boundaries first.

```
#pragma acc parallel loop present(A,Anew)
for( int ix = ix_start; ix < ix_end; ix++ )
{
    A[(iy_start)*nx+ix] = Anew[(iy_start)*nx+ix];
    A[(iy_end-1)*nx+ix] = Anew[(iy_end-1)*nx+ix];
}
```

Start the interior loop asynchronously so it can overlap with the MPI communication and wait at the end for the completion.

```
#pragma acc parallel loop present(A,Anew) async
for (int iy = iy_start+1; iy < iy_end-1; iy++)
{
    for( int ix = ix_start; ix < ix_end; ix++ )
    {
        A[iy*nx+ix] = Anew[iy*nx+ix];
    }
}

//Periodic boundary conditions
int top    = (rank == 0) ? (size-1) : rank-1;
int bottom = (rank == (size-1)) ? 0 : rank+1;
#pragma acc host_data use_device( A )
{
    double start_mpi = MPI_Wtime();
    //1. Sent row iy_start (first modified row) to top receive lower boundary (iy_end) from bo
    MPI_Sendrecv( A+iy_start*nx+ix_start, (ix_end-ix_start), MPI_REAL_TYPE, top    , 0,
                  A+iy_end*nx+ix_start,    (ix_end-ix_start), MPI_REAL_TYPE, bottom, 0,
                  MPI_COMM_WORLD, MPI_STATUS_IGNORE );

    //2. Sent row (iy_end-1) (last modified row) to bottom receive upper boundary (iy_start-1)
    MPI_Sendrecv( A+(iy_end-1)*nx+ix_start, (ix_end-ix_start), MPI_REAL_TYPE, bottom, 0,
                  A+(iy_start-1)*nx+ix_start, (ix_end-ix_start), MPI_REAL_TYPE, top    , 0,
                  MPI_COMM_WORLD, MPI_STATUS_IGNORE );
    mpi_time += MPI_Wtime() - start_mpi;
}
#pragma acc wait
```

## Code

- [C Version](#)
- [Fortran Version](#)

**Compiling, Running and Profiling** You can compile, run and profile the solution with the next cells. You can profile the code by executing the next cell. **After** the profiling completed download the tarball containing the profiles (pgprof.Task2.solution.poisson2d.tar.gz) with the File Browser. Then you can import them into pgprof / nvvp using the *Import* option in the *File* menu. Remember to use the *Multiple processes* option in the assistant.

```
[17]: %cd $basedir/task3
```

```
/autofs/nccsopen-svm1_home/mathiasw/sc19-tutorial-  
openpower/4-GPU/HandsOn/Solution/C/task3
```

```
[18]: checkdir('task3')
      !make poisson2d.solution
```

```
mpicc -c -DUSE_DOUBLE -Minfo=accel -fast -acc -ta=tesla:cc70,pinned
poisson2d_serial.c -o poisson2d_serial.o
poisson2d_serial:
    36, Generating present(Anew[:,rhs:],Aref[:])
    39, Generating update device(rhs[:ny*nx],Aref[:ny*nx])
    42, Generating Tesla code
        43, #pragma acc loop gang /* blockIdx.x */
        44, #pragma acc loop vector(128) /* threadIdx.x */
        49, Generating implicit reduction(max:error)
    44, Loop is parallelizable
    53, Generating Tesla code
        54, #pragma acc loop gang /* blockIdx.x */
        55, #pragma acc loop vector(128) /* threadIdx.x */
    55, Loop is parallelizable
    61, Generating Tesla code
        62, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
    66, Generating Tesla code
        67, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
    78, Generating update self(Aref[:ny*nx])
mpicc -DUSE_DOUBLE -Minfo=accel -fast -acc -ta=tesla:cc70,pinned
poisson2d.solution.c poisson2d_serial.o -o poisson2d.solution
poisson2d.solution.c:
main:
    71, Generating enter data
create(rhs[:ny*nx],Aref[:ny*nx],A[:ny*nx],Anew[:ny*nx])
    87, Generating present(Aref[:,A[:])
        Generating Tesla code
        88, #pragma acc loop gang /* blockIdx.x */
        89, #pragma acc loop vector(128) /* threadIdx.x */
    89, Loop is parallelizable
    140, Generating update device(A[nx*(iy_start-1):nx*((iy_end-
iy_start)+2)],rhs[nx*iy_start:nx*(iy_end-iy_start)])
    143, Generating present(A[:,rhs:],Anew[:])
        Generating Tesla code
        144, #pragma acc loop gang /* blockIdx.x */
        145, #pragma acc loop vector(128) /* threadIdx.x */
        149, Generating implicit reduction(max:error)
    145, Loop is parallelizable
    157, Generating present(Anew[:,A[:])
        Generating Tesla code
        158, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
    163, Generating present(Anew[:,A[:])
        Generating Tesla code
        164, #pragma acc loop gang /* blockIdx.x */
```

```

165, #pragma acc loop vector(128) /* threadIdx.x */
165, Loop is parallelizable
191, Generating present(A[:])
    Generating Tesla code
192, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
202, Generating update self(A[nx*(iy_start-1):nx*((iy_end-iy_start)+2)])
220, Generating exit data delete(rhs[:1],Aref[:1],A[:1],Anew[:1])

```

```

[19]: checkdir('task3')
      !NP=2 make run.solution

```

```

bsub -W 60 -nnodes 1 -Is -P TRN003 jsrun -n 1 -c 1 -g ALL_GPUS -a 2 -c ALL_CPUS
-d cyclic -b packed:7 --smpiargs "-gpu" ./poisson2d.solution
Job <25201> is submitted to default queue <batch>.

```

```
<<Waiting for dispatch ...>>
```

```
<<Starting on login1>>
```

```
Jacobi relaxation Calculation: 4096 x 4096 mesh
```

```
Calculate reference solution and time serial execution.
```

```

0, 0.250000
100, 0.249940
200, 0.249880
300, 0.249821
400, 0.249761
500, 0.249702
600, 0.249642
700, 0.249583
800, 0.249524
900, 0.249464

```

```
Parallel execution.
```

```

0, 0.250000
100, 0.249940
200, 0.249880
300, 0.249821
400, 0.249761
500, 0.249702
600, 0.249642
700, 0.249583
800, 0.249524
900, 0.249464

```

```
Num GPUs: 2.
```

```
4096x4096: 1 GPU: 1.3175 s, 2 GPUs: 0.6962 s, speedup: 1.89, efficiency:
94.62%
```

```
MPI time: 0.0583 s, inter GPU BW: 2.09 GiB/s
```

```

[20]: checkdir('task3')
      !NP=2 make profile.solution

```

```
bsub -W 60 -nnodes 1 -Is -P TRN003 jsrun -n 1 -c 1 -g ALL_GPUS -a 2 -c ALL_CPUS
```



```

-d cyclic -b packed:7 --smpiargs "-gpu" pgprof -f --cpu-profiling off --openmp-
profiling off --annotate-mpi openmpi -o /gpfs/wolf/trn003/scratch/mathiasw//pois
son2d.solution.Task3.NP2.%q{OMPI_COMM_WORLD_RANK}.pgprof ./poisson2d.solution 10
Job <25202> is submitted to default queue <batch>.
<<Waiting for dispatch ...>>
<<Starting on login1>>
==93249== PGPROF is profiling process 93249, command: ./poisson2d.solution 10
==93248== PGPROF is profiling process 93248, command: ./poisson2d.solution 10
==93249== Generated result file:
/gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.Task3.NP2.1.pgprof
Jacobi relaxation Calculation: 4096 x 4096 mesh
Calculate reference solution and time serial execution.
    0, 0.250000
Parallel execution.
    0, 0.250000
Num GPUs: 2.
4096x4096: 1 GPU:    0.0262 s, 2 GPUs:    0.0127 s, speedup:        2.06, efficiency:
103.02%
MPI time:    0.0009 s, inter GPU BW:        1.39 GiB/s
==93248== Generated result file:
/gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.Task3.NP2.0.pgprof
mv /gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.Task3.NP2.?.pgprof .
tar -cvzf pgprof.poisson2d.Task3.solution.tar.gz
poisson2d.solution.Task3.NP2.?.pgprof
poisson2d.solution.Task3.NP2.0.pgprof
poisson2d.solution.Task3.NP2.1.pgprof

```

**Scaling** You can do a simple scaling run for up to all 6 GPUs in the node by executing the next cell.

```

[21]: checkdir('task3')
!NP=1 make run.solution | grep speedup > scale.out
!NP=2 make run.solution | grep speedup >> scale.out
!NP=4 make run.solution | grep speedup >> scale.out
!NP=6 make run.solution | grep speedup >> scale.out
data_frameS3 = pandas.read_csv('scale.out', delim_whitespace=True, header=None)

!rm scale.out

data_frameS3b=data_frameS3.iloc[:,[5,7,10,12]].copy()
data_frameS3b.rename(columns={5:'GPUs', 7: 'time [s]', 10:'speedup', 12:
↪'efficiency'})

```

```

<<Waiting for dispatch ...>>
<<Starting on login1>>
<<Waiting for dispatch ...>>
<<Starting on login1>>
<<Waiting for dispatch ...>>

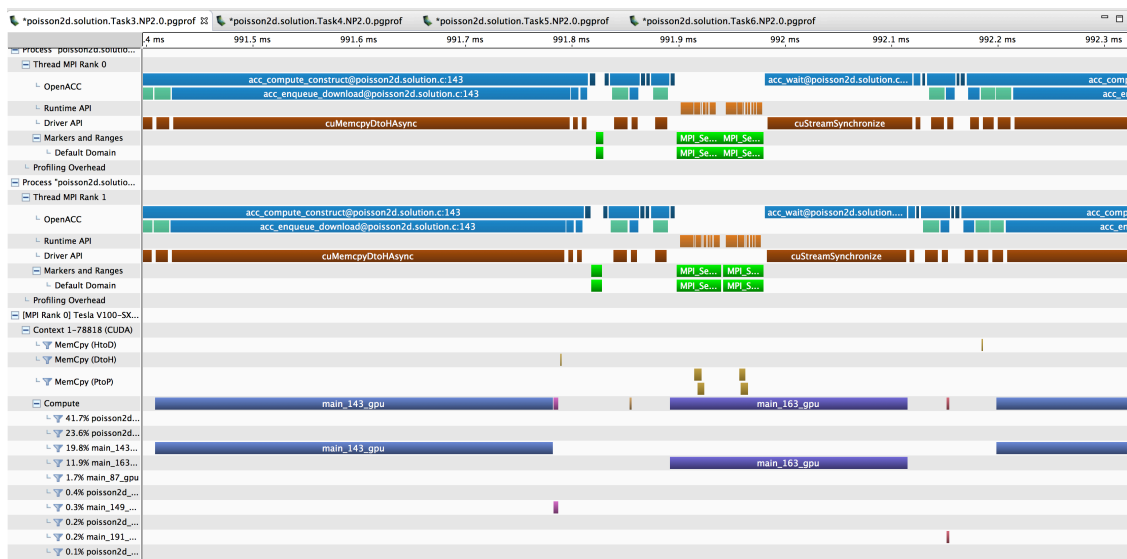
```

```
<<Starting on login1>>
<<Waiting for dispatch ...>>
<<Starting on login1>>
```

[21]:

	GPUs	time [s]	speedup	efficiency
0	1	1.3935	0.94,	93.86%
1	2	0.6910	1.89,	94.52%
2	4	0.3920	3.37,	84.13%
3	6	0.2841	4.58,	76.29%

The overlap of compute and communication can be seen in the profiler, e.g. as shown below.



Section ??

## 2.5 Solution 4:

Include NVSHMEM headers

```
#include <nvshmem.h>
#include <nvshmemx.h>
```

and initialize NVSHMEM with MPI

```
MPI_Comm mpi_comm = MPI_COMM_WORLD;
nvshmemx_init_attr_t attr;
attr.mpi_comm = &mpi_comm;
nvshmemx_init_attr(NVSHMEMX_INIT_WITH_MPI_COMM, &attr);
```

Allocate device memory and map it to the host allocation for OpenACC

```
real *d_A = (real *)nvshmem_malloc(nx * ny * sizeof(real));
map(A, d_A, nx * ny * sizeof(real));
```

Calculate the right locations on the remote GPUs and communicate data

```
// Periodic boundary conditions
int top = (rank == 0) ? (size - 1) : rank - 1;
int bottom = (rank == (size - 1)) ? 0 : rank + 1;
int iy_start_top = top * chunk_size;
int iy_end_top = iy_start_top + chunk_size;

// Do not process boundaries
iy_start_top = max(iy_start_top, 1);
iy_end_top = min(iy_end_top, ny - 1);

int iy_start_bottom = bottom * chunk_size;
int iy_end_bottom = iy_start_bottom + chunk_size;

// Do not process boundaries
iy_start_bottom = max(iy_start_bottom, 1);
iy_end_bottom = min(iy_end_bottom, ny - 1);

// Halo exchange
#pragma acc host_data use_device(A)
{
    double start_mpi = MPI_Wtime();
    nvshmem_double_put((double *) (A + iy_end_top * nx + ix_start),
                      (double *) (A + iy_start * nx + ix_start), (ix_end - ix_start), top);
    nvshmem_double_put((double *) (A + (iy_start_bottom - 1) * nx + ix_start),
                      (double *) (A + (iy_end - 1) * nx + ix_start), (ix_end - ix_start),
                      bottom);
    nvshmem_barrier_all();
    mpi_time += MPI_Wtime() - start_mpi;
}
```

Finally, remember to deallocate:

```
nvshmem_free(d_A);
```

## Code

- [C Version](#)

**Compiling, Running and Profiling** You can compile, run and profile the solution with the next cells. You can profile the code by executing the next cell. **After** the profiling completed download the tarball containing the profiles (`pgprof.Task4.solution.poisson2d.tar.gz`) with the File Browser. Then you can import them into pgprof / nvvp using the *Import* option in the *File* menu. Remember to use the *Multiple processes* option in the assistant.

```
[22]: %cd $basedir/task4
```

```
/autofs/nccsopen-svm1_home/mathiasw/sc19-tutorial-
openpower/4-GPU/HandsOn/Solution/C/task4
```

```
[23]: checkdir('task4')
      !make poisson2d.solution
```

```
mpicxx -c -DUSE_DOUBLE -Minfo=accel -fast -acc -ta=tesla:cc70,pinned
poisson2d_serial.c -o poisson2d_serial.o
poisson2d_serial(int, int, double, double *, double *, int, int, const double
*):
    37, Generating present(Anew[:,rhs[:,Aref[:]])
    39, Generating update device(rhs[:ny*nx],Aref[:ny*nx])
    40, Generating Tesla code
        43, #pragma acc loop gang /* blockIdx.x */
        44, #pragma acc loop vector(128) /* threadIdx.x */
        49, Generating implicit reduction(max:error)
    44, Loop is parallelizable
    51, Generating Tesla code
        54, #pragma acc loop gang /* blockIdx.x */
        55, #pragma acc loop vector(128) /* threadIdx.x */
    55, Loop is parallelizable
    58, Generating Tesla code
        62, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
    65, Generating Tesla code
        67, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
    77, Generating update self(Aref[:ny*nx])
mpicxx -DUSE_DOUBLE -Minfo=accel -fast -acc -ta=tesla:cc70,pinned
-I/gpfs/wolf/trn003/world-shared/software/nvshmem//include poisson2d.solution.c
poisson2d_serial.o -o poisson2d.solution -L/gpfs/wolf/trn003/world-
shared/software/nvshmem//lib -lnvshmem -Mcuda -lcuda -lrt
poisson2d.solution.c:
main:
    90, Generating enter data
create(Aref[:ny*nx],rhs[:ny*nx],A[:ny*nx],Anew[:ny*nx])
    101, Generating present(Aref[:,A[:]])
        Generating Tesla code
        105, #pragma acc loop gang /* blockIdx.x */
        106, #pragma acc loop vector(128) /* threadIdx.x */
    106, Loop is parallelizable
    162, Generating update device(A[nx*(iy_start-1):nx*((iy_end-
iy_start)+2)],rhs[nx*iy_start:nx*(iy_end-iy_start)])
    163, Generating present(A[:,rhs[:,Anew[:]])
        Generating Tesla code
        166, #pragma acc loop gang /* blockIdx.x */
        167, #pragma acc loop vector(128) /* threadIdx.x */
        171, Generating implicit reduction(max:error)
    167, Loop is parallelizable
    177, Generating present(Anew[:,A[:]])
        Generating Tesla code
        180, #pragma acc loop gang /* blockIdx.x */
```

```

181, #pragma acc loop vector(128) /* threadIdx.x */
181, Loop is parallelizable
214, Generating present(A[:])
    Generating Tesla code
217, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
227, Generating update self(A[nx*(iy_start-1):nx*((iy_end-iy_start)+2)])
246, Generating exit data delete(rhs[:1],Aref[:1],A[:1],Anew[:1])

```

```

[24]: checkdir('task4')
      !NP=2 make run.solution

```

```

bsub -W 60 -nnodes 1 -Is -P TRN003 jsrun -n 1 -c 1 -g ALL_GPUS -a 2 -c ALL_CPUS
-d cyclic -b packed:7 --smpiargs "-gpu" ./poisson2d.solution
Job <25207> is submitted to default queue <batch>.
<<Waiting for dispatch ...>>
<<Starting on login1>>
WARN: IB HCA and GPU are not connected to a PCIe switch so IB performance can be
limited depending on the CPU generation
Jacobi relaxation Calculation: 4096 x 4096 mesh
Calculate reference solution and time serial execution.
  0, 0.250000
 100, 0.249940
 200, 0.249880
 300, 0.249821
 400, 0.249761
 500, 0.249702
 600, 0.249642
 700, 0.249583
 800, 0.249524
 900, 0.249464
Parallel execution.
  0, 0.250000
 100, 0.249940
 200, 0.249880
 300, 0.249821
 400, 0.249761
 500, 0.249702
 600, 0.249642
 700, 0.249583
 800, 0.249524
 900, 0.249464
Num GPUs: 2.
4096x4096: 1 GPU:   1.3171 s, 2 GPUs:   0.7377 s, speedup:   1.79, efficiency:
89.27%
MPI time:   0.0686 s, inter GPU BW:   1.78 GiB/s

```

```
[25]: checkdir('task4')
      !NP=2 make profile.solution
```

```
bsub -W 60 -nnodes 1 -Is -P TRN003 jsrun -n 1 -c 1 -g ALL_GPUS -a 2 -c ALL_CPUS
-d cyclic -b packed:7 --smpiargs "-gpu" pgprof -f --cpu-profiling off --openmp-
profiling off --annotate-mpi openmpi -o /gpfs/wolf/trn003/scratch/mathiasw//pois
son2d.solution.Task4.NP2.%q{OMPI_COMM_WORLD_RANK}.pgprof ./poisson2d.solution 10
Job <25208> is submitted to default queue <batch>.
<<Waiting for dispatch ...>>
<<Starting on login1>>
==93971== PGPROF is profiling process 93971, command: ./poisson2d.solution 10
==93970== PGPROF is profiling process 93970, command: ./poisson2d.solution 10
==93971== Generated result file:
/gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.Task4.NP2.0.pgprof
WARN: IB HCA and GPU are not connected to a PCIe switch so IB performance can be
limited depending on the CPU generation
Jacobi relaxation Calculation: 4096 x 4096 mesh
Calculate reference solution and time serial execution.
    0, 0.250000
Parallel execution.
    0, 0.250000
Num GPUs: 2.
4096x4096: 1 GPU:    0.0225 s, 2 GPUs:    0.0132 s, speedup:    1.71, efficiency:
85.34%
MPI time:    0.0010 s, inter GPU BW:    1.24 GiB/s
==93970== Generated result file:
/gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.Task4.NP2.1.pgprof
mv /gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.Task4.NP2.?.pgprof .
tar -cvzf pgprof.poisson2d.Task4.solution.tar.gz
poisson2d.solution.Task4.NP2.?.pgprof
poisson2d.solution.Task4.NP2.0.pgprof
poisson2d.solution.Task4.NP2.1.pgprof
```

**Scaling** You can do a simple scaling run for up to all 6 GPUs in the node by executing the next cell.

```
[26]: checkdir('task4')
      !NP=1 make run.solution | grep speedup > scale.out
      !NP=2 make run.solution | grep speedup >> scale.out
      !NP=4 make run.solution | grep speedup >> scale.out
      !NP=6 make run.solution | grep speedup >> scale.out
      data_frameS4 = pandas.read_csv('scale.out', delim_whitespace=True, header=None)

      !rm scale.out

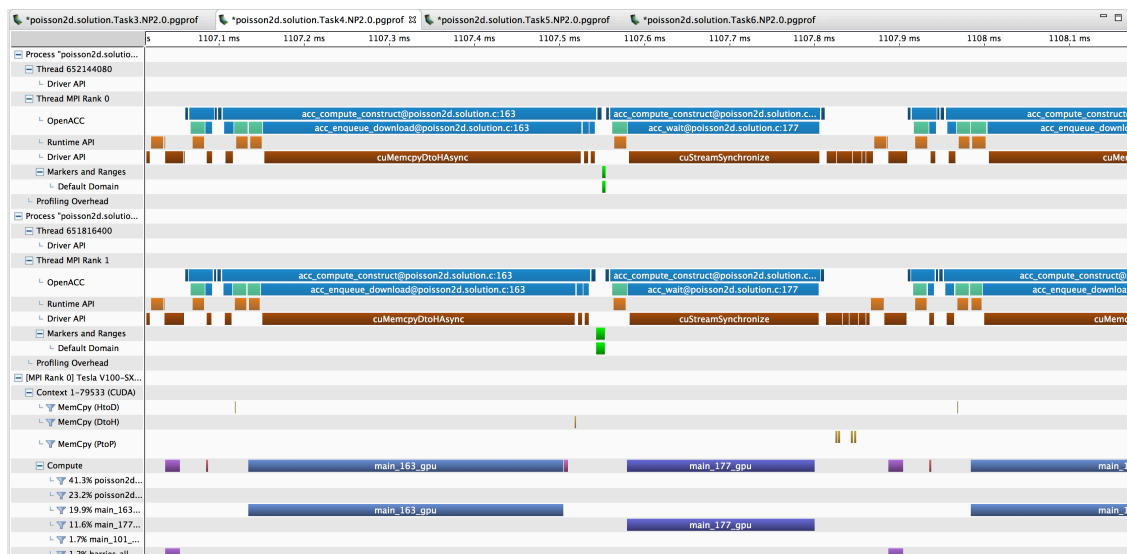
      data_frameS4b=data_frameS4.iloc[:, [5,7,10,12]].copy()
```

```
data_frameS4b.rename(columns={5:'GPUs', 7: 'time [s]', 10:'speedup', 12:
    ↪'efficiency'})
```

```
<<Waiting for dispatch ...>>
<<Starting on login1>>
<<Waiting for dispatch ...>>
<<Starting on login1>>
<<Waiting for dispatch ...>>
<<Starting on login1>>
<<Waiting for dispatch ...>>
<<Starting on login1>>
```

```
[26]:   GPUs  time [s] speedup efficiency
      0    1   1.3685   0.96,   96.08%
      1    2   0.7472   1.78,   88.90%
      2    4   0.4605   2.85,   71.27%
      3    6   0.3612   3.60,   60.05%
```

The communication using NVSHMEM and the barrier executed as a kernel on the device can be seen in the profiler, e.g. as shown below.



Section ??

## 2.6 Solution 5:

Basically all kernels in the `while` loop can use the `async` keyword. Please take a look in the solution source code. They will all use the OpenACC default `async` queue.

To also place the halo exchange in the queue use:

```
#pragma acc host_data use_device(A)
{
    nvshmemx_double_put_on_stream(
        (double *) (A + iy_end_top * nx + ix_start),
        (double *) (A + iy_start * nx + ix_start), (ix_end - ix_start), top,
        (cudaStream_t) acc_get_cuda_stream(acc_get_default_async()));
    nvshmemx_double_put_on_stream(
        (double *) (A + (iy_start_bottom - 1) * nx + ix_start),
        (double *) (A + (iy_end - 1) * nx + ix_start), (ix_end - ix_start), bottom,
        (cudaStream_t) acc_get_cuda_stream(acc_get_default_async()));
}
nvshmemx_barrier_all_on_stream((cudaStream_t) acc_get_cuda_stream(acc_get_default_async()));
```

Finally when copying out data make sure to wait on all device computation first:

```
#pragma acc update self(A [(iy_start - 1) * nx : ((iy_end - iy_start) + 2) * nx]) wait
```

## Code

- [C Version](#)

**Compiling, Running and Profiling** You can compile, run and profile the solution with the next cells. You can profile the code by executing the next cell. **After** the profiling completed download the tarball containing the profiles (pgprof.Task5.solution.poisson2d.tar.gz) with the File Browser. Then you can import them into pgprof / nvvp using the *Import* option in the *File* menu. Remember to use the *Multiple processes* option in the assistant.

```
[27]: %cd $basedir/task5
```

```
/autofs/nccsopen-svm1_home/mathiasw/sc19-tutorial-
openpower/4-GPU/HandsOn/Solution/C/task5
```

```
[28]: checkdir('task5')
!make poisson2d.solution
```

```
mpicxx -c -DUSE_DOUBLE -Minfo=accel -fast -acc -ta=tesla:cc70,pinned
poisson2d_serial.c -o poisson2d_serial.o
poisson2d_serial(int, int, double, double *, double *, int, int, const double
*):
```

```
37, Generating present(Anew[:,rhs[:],Aref[:])
39, Generating update device(rhs[:ny*nx],Aref[:ny*nx])
40, Generating Tesla code
43, #pragma acc loop gang /* blockIdx.x */
44, #pragma acc loop vector(128) /* threadIdx.x */
49, Generating implicit reduction(max:error)
44, Loop is parallelizable
51, Generating Tesla code
54, #pragma acc loop gang /* blockIdx.x */
55, #pragma acc loop vector(128) /* threadIdx.x */
```



```

55, Loop is parallelizable
58, Generating Tesla code
62, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
65, Generating Tesla code
67, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
77, Generating update self(Aref[:ny*nx])
mpicxx -DUSE_DOUBLE -Minfo=accel -fast -acc -ta=tesla:cc70,pinned
-I/ccsopen/home/mathiasw/nvshmem-master/build/include poisson2d.solution.c
poisson2d_serial.o -o poisson2d.solution -L/ccsopen/home/mathiasw/nvshmem-
master/build/lib -lnvshmem -Mcuda -lcuda -lrt
poisson2d.solution.c:
main:
90, Generating enter data
create(Aref[:ny*nx],rhs[:ny*nx],A[:ny*nx],Anew[:ny*nx])
101, Generating present(Aref[:],A[:])
Generating Tesla code
105, #pragma acc loop gang /* blockIdx.x */
106, #pragma acc loop vector(128) /* threadIdx.x */
106, Loop is parallelizable
137, Generating update device(A[nx*(iy_start-1):nx*((iy_end-
iy_start)+2)],rhs[nx*iy_start:nx*(iy_end-iy_start)])
138, Generating present(A[:],rhs[:],Anew[:])
Generating Tesla code
141, #pragma acc loop gang /* blockIdx.x */
142, #pragma acc loop vector(128) /* threadIdx.x */
146, Generating implicit reduction(max:error)
142, Loop is parallelizable
152, Generating present(Anew[:],A[:])
Generating Tesla code
155, #pragma acc loop gang /* blockIdx.x */
156, #pragma acc loop vector(128) /* threadIdx.x */
156, Loop is parallelizable
190, Generating present(A[:])
Generating Tesla code
193, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
203, Generating update self(A[nx*(iy_start-1):nx*((iy_end-iy_start)+2)])
221, Generating exit data delete(rhs[:1],Aref[:1],A[:1],Anew[:1])

```

```

[29]: checkdir('task5')
      !NP=2 make run.solution

```

```

bsub -W 60 -nnodes 1 -Is -P TRN003 jsrun -n 1 -c 1 -g ALL_GPUS -a 2 -c ALL_CPUS
-d cyclic -b packed:7 --smpiargs "-gpu" ./poisson2d.solution
Job <25213> is submitted to default queue <batch>.
<<Waiting for dispatch ...>>
<<Starting on login1>>
WARN: IB HCA and GPU are not connected to a PCIe switch so IB performance can be
limited depending on the CPU generation

```

Jacobi relaxation Calculation: 4096 x 4096 mesh  
Calculate reference solution and time serial execution.

0, 0.250000  
100, 0.249940  
200, 0.249880  
300, 0.249821  
400, 0.249761  
500, 0.249702  
600, 0.249642  
700, 0.249583  
800, 0.249524  
900, 0.249464

Parallel execution.

0, 0.250000  
100, 0.249940  
200, 0.249880  
300, 0.249821  
400, 0.249761  
500, 0.249702  
600, 0.249642  
700, 0.249583  
800, 0.249524  
900, 0.249464

Num GPUs: 2.

4096x4096: 1 GPU: 1.3176 s, 2 GPUs: 0.6777 s, speedup: 1.94, efficiency:  
97.22%

```
[30]: checkdir('task5')  
!NP=2 make profile.solution
```

```
bsub -W 60 -nnodes 1 -Is -P TRN003 jsrun -n 1 -c 1 -g ALL_GPUS -a 2 -c ALL_CPUS  
-d cyclic -b packed:7 --smpiargs "-gpu" pgprof -f --cpu-profiling off --openmp-  
profiling off --annotate-mpi openmpi -o /gpfs/wolf/trn003/scratch/mathiasw//pois-  
son2d.solution.Task5.NP2.%q{OMPI_COMM_WORLD_RANK}.pgprof ./poisson2d.solution 10  
Job <25214> is submitted to default queue <batch>.
```

```
<<Waiting for dispatch ...>>
```

```
<<Starting on login1>>
```

```
==94705== PGPROF is profiling process 94705, command: ./poisson2d.solution 10
```

```
==94707== PGPROF is profiling process 94707, command: ./poisson2d.solution 10
```

```
==94707== Generated result file:
```

```
/gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.Task5.NP2.1.pgprof
```

```
WARN: IB HCA and GPU are not connected to a PCIe switch so IB performance can be  
limited depending on the CPU generation
```

Jacobi relaxation Calculation: 4096 x 4096 mesh

Calculate reference solution and time serial execution.

0, 0.250000

Parallel execution.

0, 0.250000

```

Num GPUs: 2.
4096x4096: 1 GPU: 0.0225 s, 2 GPUs: 0.0117 s, speedup: 1.92, efficiency:
96.05%
==94705== Generated result file:
/gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.Task5.NP2.0.pgprof
mv /gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.Task5.NP2.?.pgprof .
tar -cvzf pgprof.poisson2d.Task5.solution.tar.gz
poisson2d.solution.Task5.NP2.?.pgprof
poisson2d.solution.Task5.NP2.0.pgprof
poisson2d.solution.Task5.NP2.1.pgprof

```

**Scaling** You can do a simple scaling run for up to all 6 GPUs in the node by executing the next cell.

```

[31]: checkdir('task5')
!NP=1 make run.solution | grep speedup > scale.out
!NP=2 make run.solution | grep speedup >> scale.out
!NP=4 make run.solution | grep speedup >> scale.out
!NP=6 make run.solution | grep speedup >> scale.out
data_frameS5 = pandas.read_csv('scale.out', delim_whitespace=True, header=None)

!rm scale.out

data_frameS5b=data_frameS5.iloc[:,[5,7,10,12]].copy()
data_frameS5b.rename(columns={5:'GPUs', 7: 'time [s]', 10:'speedup', 12:
↪'efficiency'})

```

```

<<Waiting for dispatch ...>>
<<Starting on login1>>
<<Waiting for dispatch ...>>
<<Starting on login1>>
<<Waiting for dispatch ...>>
<<Starting on login1>>
<<Waiting for dispatch ...>>
<<Starting on login1>>

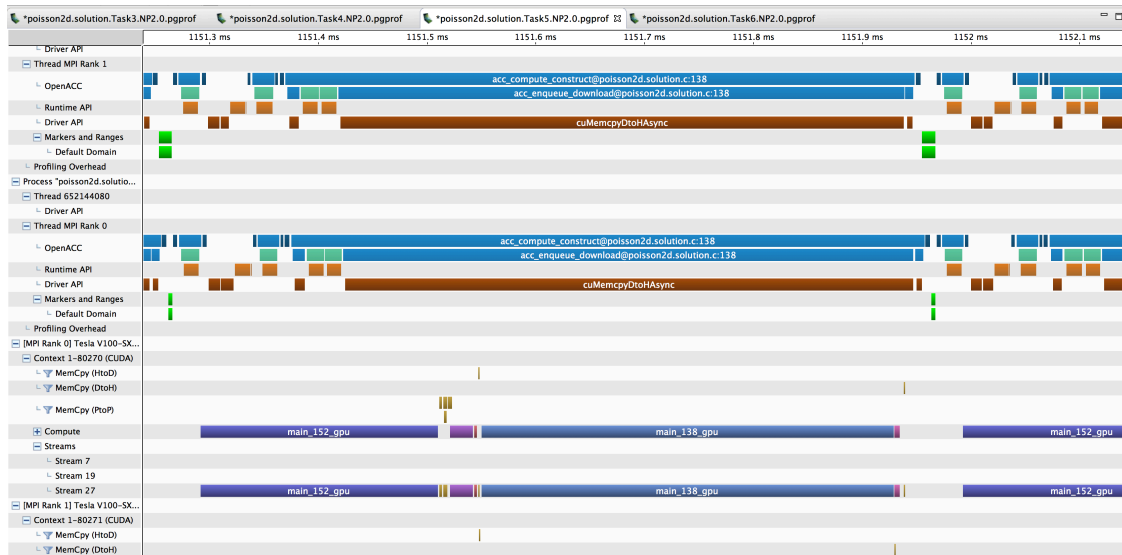
```

```

[31]:   GPUs  time [s] speedup efficiency
0      1    1.2915   1.02,   101.63%
1      2    0.6742   1.96,    98.08%
2      4    0.3801   3.47,    86.66%
3      6    0.2733   4.80,    80.04%

```

The asynchronous execution and execution in the same stream can be seen in the profiler, e.g. as shown below.



Section ??

## 2.7 Solution 6: TODO

### Code

- [C Version](#)

**Compiling, Running and Profiling** You can compile, run and profile the solution with the next cells. You can profile the code by executing the next cell. **After** the profiling completed download the tarball containing the profiles (`pgprof.Task6.solution.poisson2d.tar.gz`) with the File Browser. Then you can import them into pgprof / nvvp using the *Import* option in the *File* menu. Remember to use the *Multiple processes* option in the assistant.

```
[32]: %cd $basedir/task6
```

```
/autofs/nccsopen-svm1_home/mathiasw/sc19-tutorial-  
openpower/4-GPU/HandsOn/Solution/C/task6
```

```
[33]: checkdir('task6')  
!make poisson2d.solution
```

```
mpicxx -c -DUSE_DOUBLE -Minfo=accel -fast -acc -ta=tesla:cc70,pinned  
poisson2d_serial.c -o poisson2d_serial.o  
poisson2d_serial(int, int, double, double *, double *, int, int, const double  
*):  
    37, Generating present(Anew[:,rhs:],Aref[:])  
    39, Generating update device(rhs[:ny:nx],Aref[:ny:nx])  
    40, Generating Tesla code  
        43, #pragma acc loop gang /* blockIdx.x */
```

```

44, #pragma acc loop vector(128) /* threadIdx.x */
49, Generating implicit reduction(max:error)
44, Loop is parallelizable
51, Generating Tesla code
54, #pragma acc loop gang /* blockIdx.x */
55, #pragma acc loop vector(128) /* threadIdx.x */
55, Loop is parallelizable
58, Generating Tesla code
62, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
65, Generating Tesla code
67, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
77, Generating update self(Aref[:ny*nx])
mpicxx -DUSE_DOUBLE -Minfo=accel -fast -acc -ta=tesla:cc70,pinned
-I/ccsopen/home/mathiasw/nvshmem-master/build/include poisson2d.solution.c
poisson2d_serial.o -o poisson2d.solution -L/ccsopen/home/mathiasw/nvshmem-
master/build/lib -lnvshmem -Mcuda -lcuda -lrt
poisson2d.solution.c:
main:
95, Generating enter data
create(Aref[:ny*nx],rhs[:ny*nx],A[:ny*nx],Anew[:ny*nx])
106, Generating present(Aref[:],A[:])
Generating Tesla code
110, #pragma acc loop gang /* blockIdx.x */
111, #pragma acc loop vector(128) /* threadIdx.x */
111, Loop is parallelizable
159, Generating update device(rhs[nx*iy_start:nx*(iy_end-
iy_start)],A[nx*(iy_start-1):nx*((iy_end-iy_start)+2)])
160, Generating present(A[:],rhs[:],Anew[:])
Generating Tesla code
165, #pragma acc loop gang /* blockIdx.x */
166, #pragma acc loop vector(128) /* threadIdx.x */
170, Generating implicit reduction(max:error)
166, Loop is parallelizable
176, Generating present(Anew[:],A[:])
Generating Tesla code
179, #pragma acc loop gang /* blockIdx.x */
181, #pragma acc loop vector(128) /* threadIdx.x */
181, Loop is parallelizable
192, Generating present(A[:])
Generating Tesla code
195, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
205, Generating update self(A[nx*(iy_start-1):nx*((iy_end-iy_start)+2)])
224, Generating exit data delete(rhs[:1],Aref[:1],A[:1],Anew[:1])

```

```

[34]: checkdir('task6')
      !NP=2 make run.solution

```

```
bsub -W 60 -nnodes 1 -Is -P TRN003 jsrun -n 1 -c 1 -g ALL_GPUS -a 2 -c ALL_CPUS
```

```

-d cyclic -b packed:7 --smpiargs "-gpu" ./poisson2d.solution
Job <25219> is submitted to default queue <batch>.
<<Waiting for dispatch ...>>
<<Starting on login1>>
WARN: IB HCA and GPU are not connected to a PCIe switch so IB performance can be
limited depending on the CPU generation
Jacobi relaxation Calculation: 4096 x 4096 mesh
Calculate reference solution and time serial execution.
    0, 0.250000
   100, 0.249940
   200, 0.249880
   300, 0.249821
   400, 0.249761
   500, 0.249702
   600, 0.249642
   700, 0.249583
   800, 0.249524
   900, 0.249464
Parallel execution.
    0, 0.250000
   100, 0.249940
   200, 0.249880
   300, 0.249821
   400, 0.249761
   500, 0.249702
   600, 0.249642
   700, 0.249583
   800, 0.249524
   900, 0.249464
Num GPUs: 2.
4096x4096: 1 GPU:   1.3157 s, 2 GPUs:   0.6533 s, speedup:   2.01, efficiency:
100.70%
MPI time:   0.0000 s, inter GPU BW:   inf GiB/s

```

```

[35]: checkdir('task6')
      !NP=2 make profile.solution

```

```

bsub -W 60 -nnodes 1 -Is -P TRN003 jsrun -n 1 -c 1 -g ALL_GPUS -a 2 -c ALL_CPUS
-d cyclic -b packed:7 --smpiargs "-gpu" pgprof -f --cpu-profiling off --openmp-
profiling off --annotate-mpi openmpi -o /gpfs/wolf/trn003/scratch/mathiasw//pois
son2d.solution.Task6.NP2.%q{OMPI_COMM_WORLD_RANK}.pgprof ./poisson2d.solution 10
Job <25220> is submitted to default queue <batch>.
<<Waiting for dispatch ...>>
<<Starting on login1>>
==95445== PGPROF is profiling process 95445, command: ./poisson2d.solution 10
==95446== PGPROF is profiling process 95446, command: ./poisson2d.solution 10
==95445== Generated result file:
/gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.Task6.NP2.1.pgprof

```

WARN: IB HCA and GPU are not connected to a PCIe switch so IB performance can be limited depending on the CPU generation

Jacobi relaxation Calculation: 4096 x 4096 mesh

Calculate reference solution and time serial execution.

0, 0.250000

Parallel execution.

0, 0.250000

Num GPUs: 2.

4096x4096: 1 GPU: 0.0225 s, 2 GPUs: 0.0116 s, speedup: 1.94, efficiency: 96.85%

MPI time: 0.0000 s, inter GPU BW: inf GiB/s

==95446== Generated result file:

/gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.Task6.NP2.0.pgprof

mv /gpfs/wolf/trn003/scratch/mathiasw//poisson2d.solution.Task6.NP2.?.pgprof .

tar -cvzf pgprof.poisson2d.Task6.solution.tar.gz

poisson2d.solution.Task6.NP2.?.pgprof

poisson2d.solution.Task6.NP2.0.pgprof

poisson2d.solution.Task6.NP2.1.pgprof

**Scaling** You can do a simple scaling run for up to all 6 GPUs in the node by executing the next cell.

```
[36]: checkdir('task6')
!NP=1 make run.solution | grep speedup > scale.out
!NP=2 make run.solution | grep speedup >> scale.out
!NP=4 make run.solution | grep speedup >> scale.out
!NP=6 make run.solution | grep speedup >> scale.out
data_frameS5 = pandas.read_csv('scale.out', delim_whitespace=True, header=None)

!rm scale.out

data_frameS5b=data_frameS5.iloc[:,[5,7,10,12]].copy()
data_frameS5b.rename(columns={5: 'GPUs', 7: 'time [s]', 10: 'speedup', 12:
↪ 'efficiency'})
```

<<Waiting for dispatch ...>>

<<Starting on login1>>

<<Waiting for dispatch ...>>

<<Starting on login1>>

<<Waiting for dispatch ...>>

<<Starting on login1>>

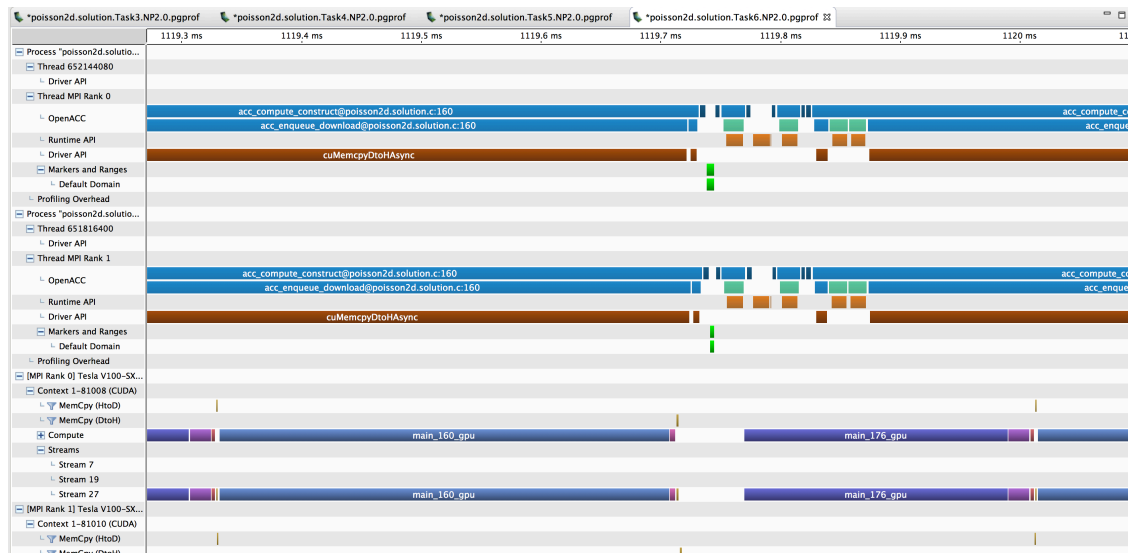
<<Waiting for dispatch ...>>

<<Starting on login1>>

```
[36]:   GPUs  time [s] speedup efficiency
0      1    1.2869    1.02,   102.05%
1      2    0.6574    1.99,    99.26%
2      4    0.3670    3.59,    89.71%
```

3      6      0.2450      5.37,      89.42%

The missing of device copies can be seen in the profiler, e.g. as shown below.



Section ??

---



---

### 3 Survey

Please remember to take some time and fill out the survey <http://bit.ly/sc19-eval>.

