Porting Jacobi on Perlmutter using OpenMP and OpenACC

Lecture 11

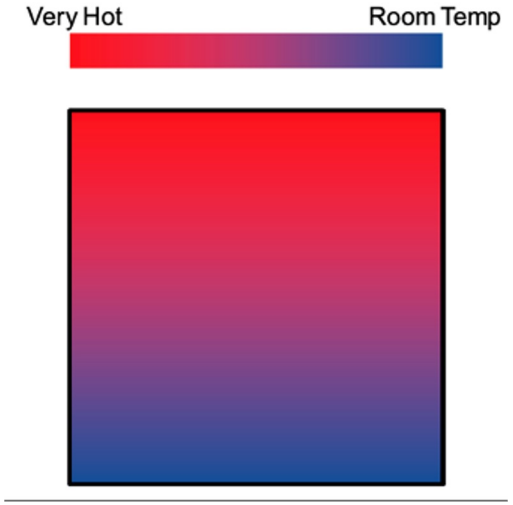
Sunita Chandrasekaran
Associate Professor, University of Delaware
PDC Summer School, Aug 2023
Materials also prepared by Dr. Felipe Cabarcas, Postdoctoral Fellow, UDEL

Table of content

- Laplace Serial code – example

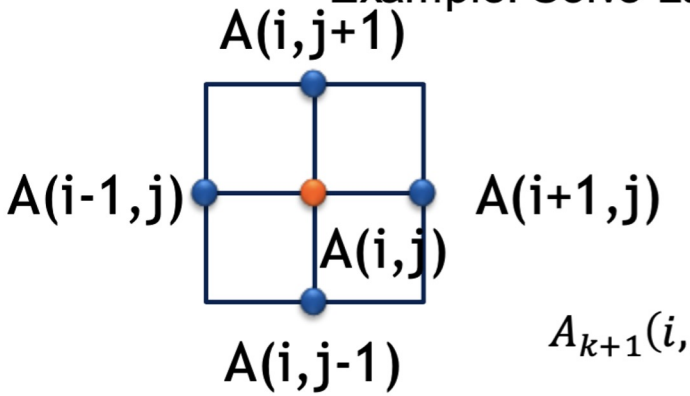
Laplace Heat Transfer

- A simple simulation of heat distributing across a metal plate
- Apply a consistent heat to the top of the plate
- Simulating the heat distribution across the plate



EXAMPLE: JACOBI ITERATION

- Iteratively converges to correct value (e.g. Temperature), by computing new values at each point from the average of neighboring points.
- Common, useful algorithm
- Example: Solve Laplace equation in 2D: $\nabla^2 f(x, y) = 0$



$$A_{k+1}(i,j) = \frac{A_k(i-1,j) + A_k(i+1,j) + A_k(i,j-1) + A_k(i,j+1)}{4}$$

```
while ( error > tol && iter < iter_max )
{
  error = 0.0;
```

Iterate until converged

Iterate across matrix elements

```
for( int j = 1; j < n-1; j++)
{
  for( int i = 1; i < m-1; i++ )
  {
    Anew[j][i] = 0.25 * ( A[j][i+1] + A[j][i-1] + A[j-1][i] + A[j+1][i]);
    error = fmax( error, fabs(Anew[j][i] - A[j][i]));
  }
}
```

Calculate new neighbors

Compute max error for convergence

```
for( int j = 1; j < n-1; j++)
{
  for( int i = 1; i < m-1; i++ )
  {
    A[j][i] = Anew[j][i];
  }
}
```

Swap input/output arrays

Profiling Sequential Code

Profile your code to obtain detailed information about how does the code runs:

- Total runtime
- runtime of routines
- Hardware counters

Identify portions that took longer to execute. These are the portions that you will want to parallelize.

LLVM

```
$ clang -Ofast -fopenmp -fno-inline -pg -o jacobi-serial jacobi.c
Jacobi relaxation Calculation: 4096 x 4096 mesh
    0, 0.250000
  100, 0.002397
  200, 0.001204
  300, 0.000804
  400, 0.000603
  500, 0.000483
  600, 0.000403
  700, 0.000345
  800, 0.000302
  900, 0.000269
 total: 25.557923 s
```

to use gprof add **-pg** to compile the application

Serial code with Nvidia nvc, performs similar to LLVM

NVC

```
$ nvc -O3  -o jacobi-serial jacobi.c
Jacobi relaxation Calculation: 4096 x 4096 mesh
    0, 0.250000
  100, 0.002397
  200, 0.001204
  300, 0.000804
  400, 0.000603
  500, 0.000483
  600, 0.000403
  700, 0.000345
  800, 0.000302
  900, 0.000269
 total: 23.364053 s
```

Table of content

- Laplace Serial code – example
- Parallelization using target parallel for

```
while ( error > tol && iter < iter_max )
{
  error = 0.0;
```

Parallelize first loop next OpenMP requires reduction clause

```
//#pragma acc parallel loop copy(A[:m*n],Anew[:m*n])
#pragma omp target parallel for map(tofrom: A[:m*n],Anew[:m*n]) \
reduction(max:error)
for( int j = 1; j < n-1; j++)
{
  for( int i = 1; i < m-1; i++ )
  {
    Anew[j][i] = 0.25 * ( A[j][i+1] + A[j][i-1] + A[j-1][i] + A[j+1][i]);
    error = fmax( error, fabs(Anew[j][i] - A[j][i]));
  }
}
```

```
//#pragma acc parallel loop copy(A[:m*n],Anew[:m*n])
#pragma omp target parallel for map(tofrom: A[:m*n],Anew[:m*n])
for( int j = 1; j < n-1; j++)
{
  for( int i = 1; i < m-1; i++ )
  {
    A[j][i] = Anew[j][i];
  }
}
```

Parallelize second loop

Build and run the code

- Using Perlmutter
- Module load nvhpc/23.1
- Target which architecture you want to use to compile and execute the code; for example
- Using OpenMP offloading
  - nvc -fast **-mp=gpu** -Minfo=all <source_code.c> -o <executable
  - **-mp=gpu**: denotes that the target gpu
  - -fast: an optimization flag that you can add to your compilation command
  - -Minfo=all: gives you information about what parts of the code were accelerated
- Check for
  - "Generating GPU kernel"
  - Proof that your code generated GPU code

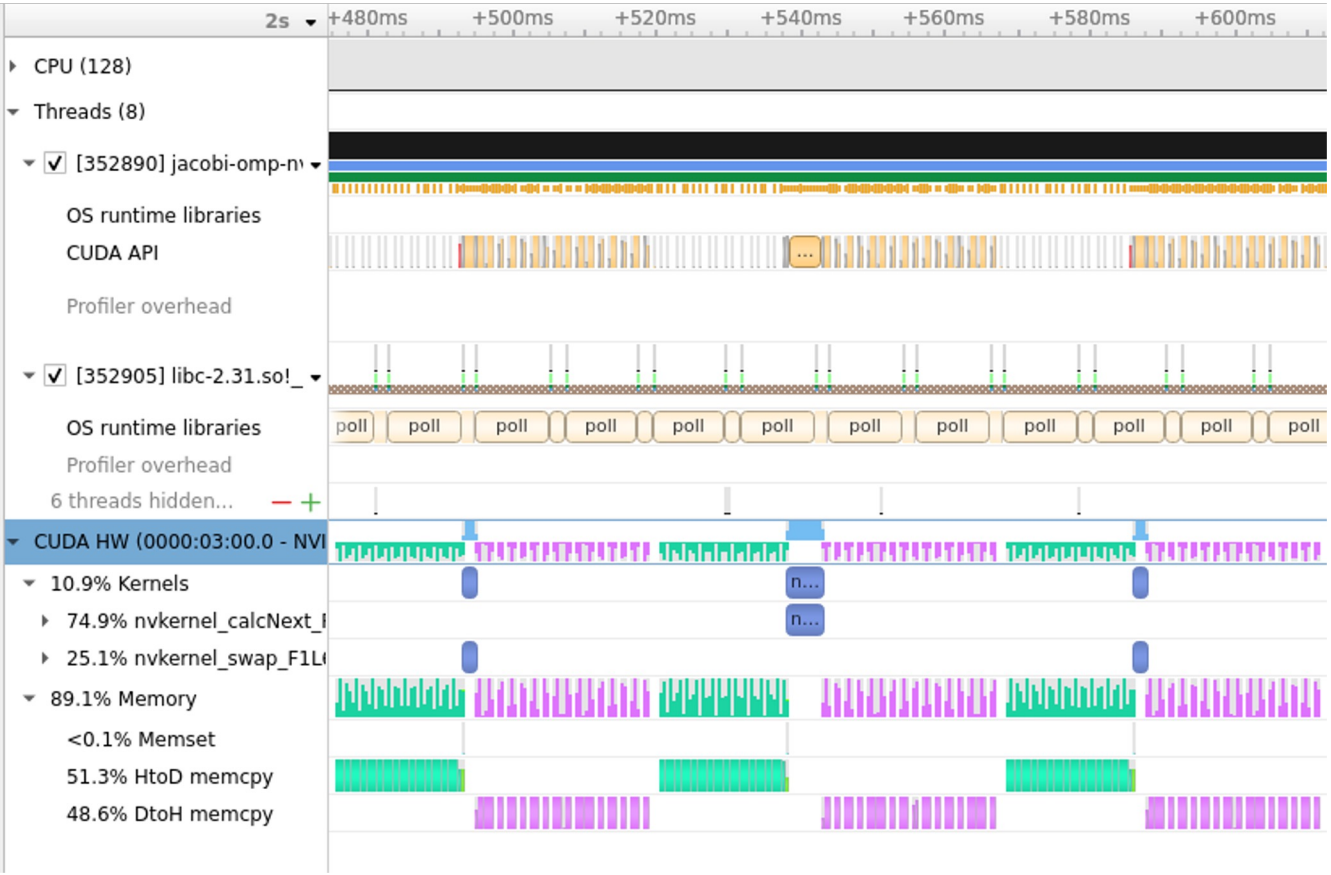
NVC

```
$ nvc -fast -mp=gpu -Minfo=all -o jacobi-omp-nvc-loop jacobi.c
initialize:
     41, Generated vector simd code for the loop
calcNext:
     52, #omp target parallel do
         52, Generating "nvkernel_calcNext_F1L52_2" GPU kernel
             Generating reduction(max:error)
             Loop parallelized across threads(128), schedule(static)
     52, Generating map(tofrom:A[:n*m],Anew[:n*m],error)
         Loop not vectorized/parallelized: not countable
     54, Generated vector simd code for the loop containing
reductions
     60, Loop not vectorized/parallelized: not countable
swap:
     65, #omp target parallel do
         65, Generating "nvkernel_swap_F1L65_5" GPU kernel
         68, Loop parallelized across threads(128), schedule(static)
     65, Generating map(tofrom:Anew[:n*m],A[:n*m])
     70, Memory copy idiom, loop replaced by call to __c_mcopy8
main:
    113, initialize inlined, size=10 (inline) file jacobi.c (37)
          41, Loop not fused: function call before adjacent loop
              Generated vector simd code for the loop
    121, Loop not vectorized/parallelized: potential early exits
    136, deallocate inlined, size=2 (inline) file jacobi.c (78)
```

```
Jacobi relaxation
Calculation: 4096 x 4096
mesh
    0, 0.250000
  100, 0.002397
  200, 0.001204
  300, 0.000804
  400, 0.000603
  500, 0.000483
  600, 0.000403
  700, 0.000345
  800, 0.000302
  900, 0.000269
 total: 89.513495 s
```

Using OpenMP offloading -
Accelerated code using parallel and no data clauses takes 89.51 on GPUs
about 4 times slower than serial

Using Nsight System




```
nvkernel_calcNext_F1L52_2
Begins: 2.63115s
Ends: 2.63503s (+3.885 ms)
grid:  <<<108, 1, 1>>>
block: <<<128, 1, 1>>>
Launch Type: Regular
Static Shared Memory: 16 bytes
Dynamic Shared Memory: 1,656 bytes
Registers Per Thread: 54
Local Memory Per Thread: 0 bytes
Local Memory Total: 127,401,984 bytes
Shared Memory executed: 65,536 bytes
Shared Memory Bank Size: 4 B
Theoretical occupancy: 56.25 %
Launched from thread: 352890
Latency: ←7.512 µs
Correlation ID: 716
Stream: Stream 16
```

Using LLVM OpenMP Offloading and NVIDIA A100

LLVM

```
$ clang -Ofast -fopenmp --offload-arch=native -g -Rpass=openmp-opt -Rpass-missed=openmp-opt -
Rpass-analysis=openmp-opt -o jacobi-omp-llvm-loop jacobi.c
remark: Found thread data sharing on the GPU. Expect degraded performance due to data
globalization. [OMP112] [-Rpass-missed=openmp-opt]
```

```
Jacobi relaxation
Calculation: 4096 x 4096
mesh
    0, 0.250000
  100, 0.002397
  200, 0.001204
  300, 0.000804
  400, 0.000603
  500, 0.000483
  600, 0.000403
  700, 0.000345
  800, 0.000302
  900, 0.000269
 total: 242.194770 s
```

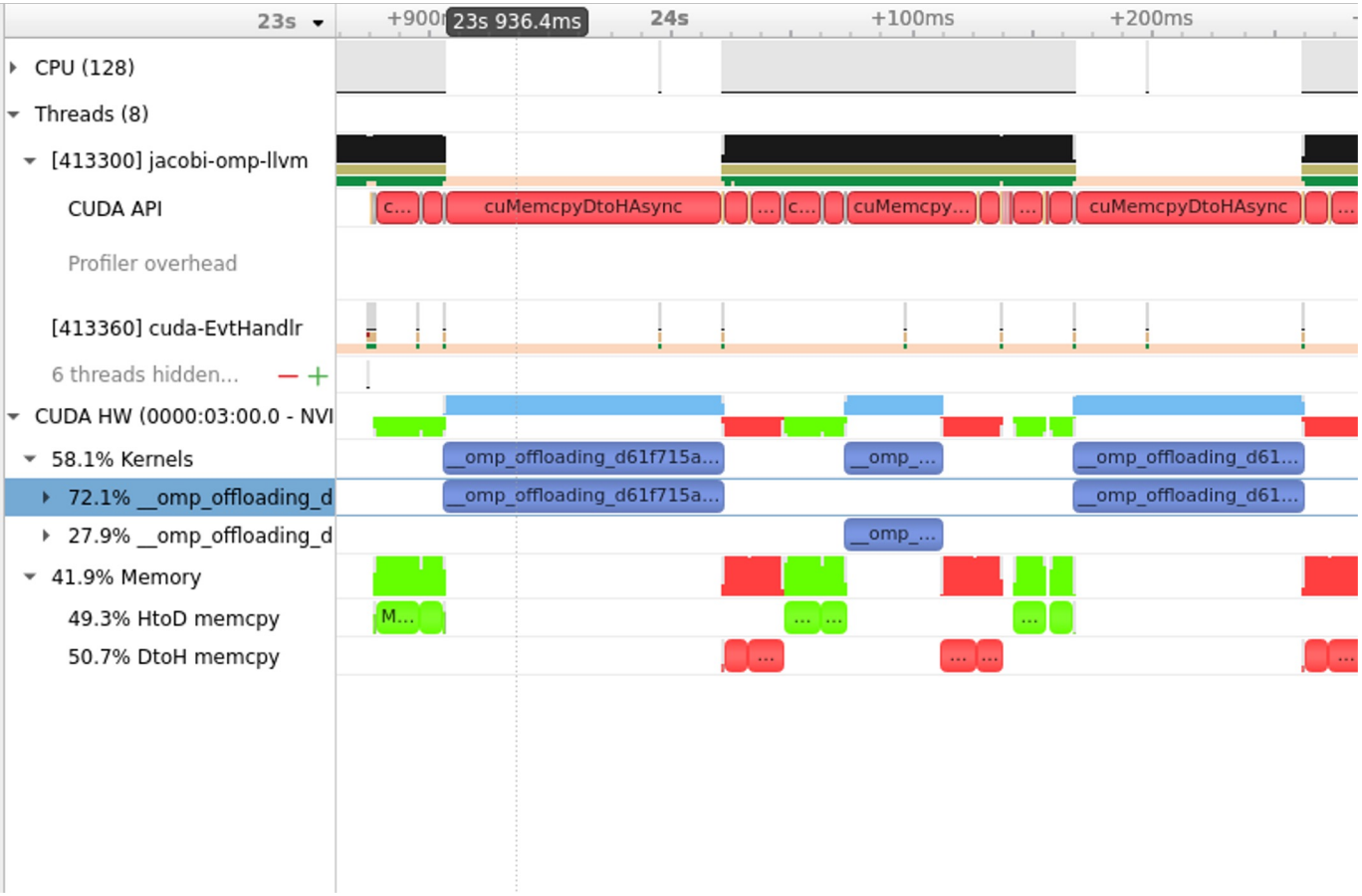
https://openmp.llvm.org//remarks/OptimizationRemarks.html

https://openmp.llvm.org//remarks/OMP112.html#omp112

This missed remark indicates that a globalized value was found on the target device that was not either replaced with stack memory by OMP110 or shared memory by OMP111. Globalization that has not been removed will need to be handled by the runtime and will significantly impact performance.
…

Using OpenMP offloading:
Accelerated code using parallel and no data clauses takes 242.19 on GPUs **about 10 times slower than serial**

Using Nsight System




```
__omp_offloading_d61f715a_4f00042e_calc
Next_l51
Begins: 24.6334s
Ends: 24.7279s (+94.461 ms)
grid:  <<<1, 1, 1>>>
block: <<<128, 1, 1>>>
Launch Type: Regular
Static Shared Memory: 1,712 bytes
Dynamic Shared Memory: 0 bytes
Registers Per Thread: 48
Local Memory Per Thread: 0 bytes
Local Memory Total: 244,187,136 bytes
Shared Memory executed: 65,536 bytes
Shared Memory Bank Size: 4 B
Theoretical occupancy: 62.5 %
Launched from thread: 413300
Latency: ←42.323 µs
Correlation ID: 282
Stream: Stream 16
```

Improving first openMP version

- The LLVM OpenMP offloading version is really slow
- Adding **teams distribute**, improves it significantly

```
while ( error > tol && iter < iter_max )
{
  error = 0.0;
```

Parallelize first loop adding teams distribute
OpenMP requires reduction clause

```
//#pragma acc parallel loop copy(A[:m*n],Anew[:m*n])
#pragma omp target teams distribute parallel for \
 map(tofrom: A[:m*n],Anew[:m*n]) reduction(max:error)
for( int j = 1; j < n-1; j++)
{
  for( int i = 1; i < m-1; i++ )
  {
    Anew[j][i] = 0.25 * ( A[j][i+1] + A[j][i-1] + A[j-1][i] + A[j+1][i]);
    error = fmax( error, fabs(Anew[j][i] - A[j][i]));
  }
}
```

```
//#pragma acc parallel loop copy(A[:m*n],Anew[:m*n])
#pragma omp target teams distribute parallel for \
 map(tofrom: A[:m*n],Anew[:m*n])
for( int j = 1; j < n-1; j++)
{
  for( int i = 1; i < m-1; i++ )
  {
    A[j][i] = Anew[j][i];
  }
}
```

Parallelize second loop

NVC

```
$ nvc -fast -mp=gpu -Minfo=all -o jacobi-omp-nvc-loop jacobi.c
initialize:
     41, Generated vector simd code for the loop
calcNext:
     52, #omp target teams distribute parallel for
         52, Generating "nvkernel_calcNext_F1L52_2" GPU kernel
             Loop parallelized across teams and threads(128),
schedule(static)
             Generating reduction(max:error)
     52, Generating map(tofrom:A[:n*m],Anew[:n*m],error)
         Loop not vectorized/parallelized: not countable
     54, Generated vector simd code for the loop containing
reductions
     60, Loop not vectorized/parallelized: not countable
swap:
     65, #omp target teams distribute parallel for
         65, Generating "nvkernel_swap_F1L65_6" GPU kernel
         68, Loop parallelized across teams and threads(128),
schedule(static)
     65, Generating map(tofrom:Anew[:n*m],A[:n*m])
     70, Memory copy idiom, loop replaced by call to __c_mcopy8
main:
    113, initialize inlined, size=10 (inline) file jacobi.c (37)
          41, Loop not fused: function call before adjacent loop
              Generated vector simd code for the loop
    121, Loop not vectorized/parallelized: potential early exits
    136, deallocate inlined, size=2 (inline) file jacobi.c (78)
```

```
Jacobi relaxation
Calculation: 4096 x 4096
mesh
    0, 0.250000
  100, 0.002397
  200, 0.001204
  300, 0.000804
  400, 0.000603
  500, 0.000483
  600, 0.000403
  700, 0.000345
  800, 0.000302
  900, 0.000269
 total: 89.992197 s
```

Using OpenMP Offloading:
Accelerated code using parallel and no data clauses takes 89.51 on GPUs **about 4 times slower than serial**

Using Nsight System


```
nvkernel_calcNext_F1L52_2
Begins: 1.17145s
Ends: 1.17577s (+4.317 ms)
grid:  <<<32, 1, 1>>>
block: <<<128, 1, 1>>>
Launch Type: Regular
Static Shared Memory: 8 bytes
Dynamic Shared Memory: 0 bytes
Registers Per Thread: 56
Local Memory Per Thread: 0 bytes
Local Memory Total: 127,401,984 bytes
Shared Memory executed: 32,768 bytes
Shared Memory Bank Size: 4 B
Theoretical occupancy: 56.25 %
Launched from thread: 624918
Latency: ←10.322 µs
Correlation ID: 706
Stream: Stream 16
```

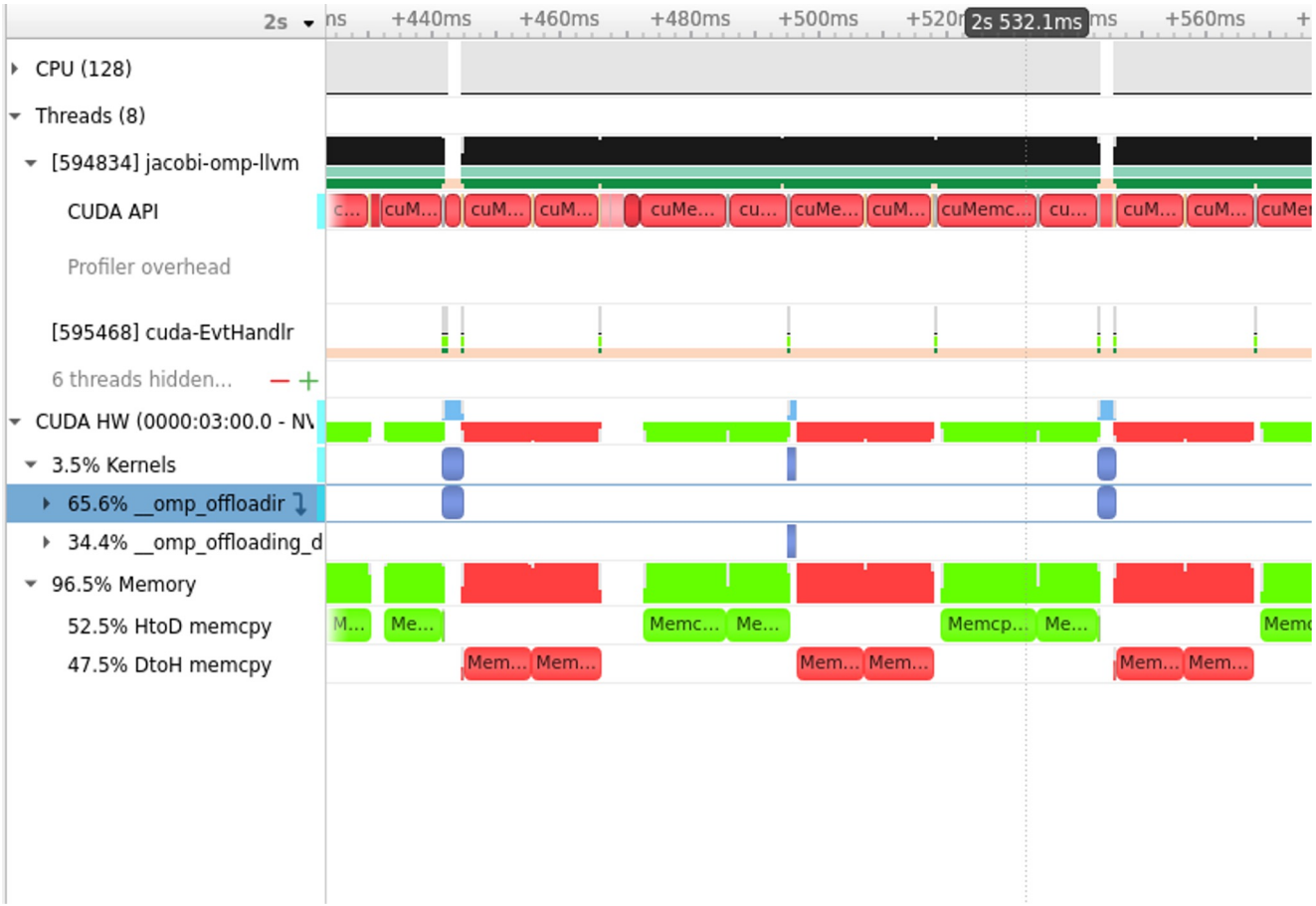
Using LLVM OpenMP Offloading and NVIDIA A100

```
LLVM
$ clang -Ofast -fopenmp --offload-arch=native -g -Rpass=openmp-opt -Rpass-missed=openmp-opt -
Rpass-analysis=openmp-opt -o jacobi-omp-llvm-loop jacobi.c
remark: Found thread data sharing on the GPU. Expect degraded performance due to data
globalization. [OMP112] [-Rpass-missed=openmp-opt]
```

```
Jacobi relaxation
Calculation: 4096 x 4096
mesh
    0, 0.250000
  100, 0.002397
  200, 0.001204
  300, 0.000804
  400, 0.000603
  500, 0.000483
  600, 0.000403
  700, 0.000345
  800, 0.000302
  900, 0.000269
 total: 100.463713 s
```

Using OpenMP offloading:
Accelerated code using parallel and no data clauses takes 242.19 on GPUs **about 10 times slower than serial**

Using Nsight System




```
__omp_offloading_d61f715a_4f000429_calcNext_l51
Begins: 2.54371s
Ends: 2.54584s (+2.132 ms)
grid:  <<<128, 1, 1>>>
block: <<<32, 1, 1>>>
Launch Type: Regular
Static Shared Memory: 1,720 bytes
Dynamic Shared Memory: 0 bytes
Registers Per Thread: 64
Local Memory Per Thread: 0 bytes
Local Memory Total: 244,187,136 bytes
Shared Memory executed: 102,400 bytes
Shared Memory Bank Size: 4 B
Theoretical occupancy: 50 %
Launched from thread: 594834
Latency: ←41.585 µs
Correlation ID: 227
Stream: Stream 16
```

What was missing in the previous code?

Our next goal is to add data clauses to our code

Table of content

- Laplace Serial code – example
- Parallelization using target parallel for
- Parallelization with target parallel and data constructs

```
//#pragma acc data copy(A[:n*m]) create(Anew[:n*m])
#pragma omp target data map(to:A[:m*n],Anew[:m*n])
while ( error > tol && iter < iter_max )
{
   error = 0.0;
```

Create data on the GPUs

```
//#pragma acc parallel loop reduction(max:error) copy(A[:m*n],Anew[:m*n])
#pragma omp target teams distribute parallel for map(tofrom:
A[:m*n],Anew[:m*n]) reduction(max:error)
for( int j = 1; j < n-1; j++)
{
  for( int i = 1; i < m-1; i++ )
  {
     Anew[j][i] = 0.25 * ( A[j][i+1] + A[j][i-1] + A[j-1][i] + A[j+1][i]);
     error = fmax( error, fabs(Anew[j][i] - A[j][i]));
   }
}
```

Parallelize and max *reduction*

```
//#pragma acc parallel loop copy(A[:m*n],Anew[:m*n])
#pragma omp target teams distribute parallel for
map(tofrom: A[:m*n],Anew[:m*n])
for( int j = 1; j < n-1; j++)
{
  for( int i = 1; i < m-1; i++ )
  {
    A[j][i] = Anew[j][i];
  }
}
```

Parallelize second loop

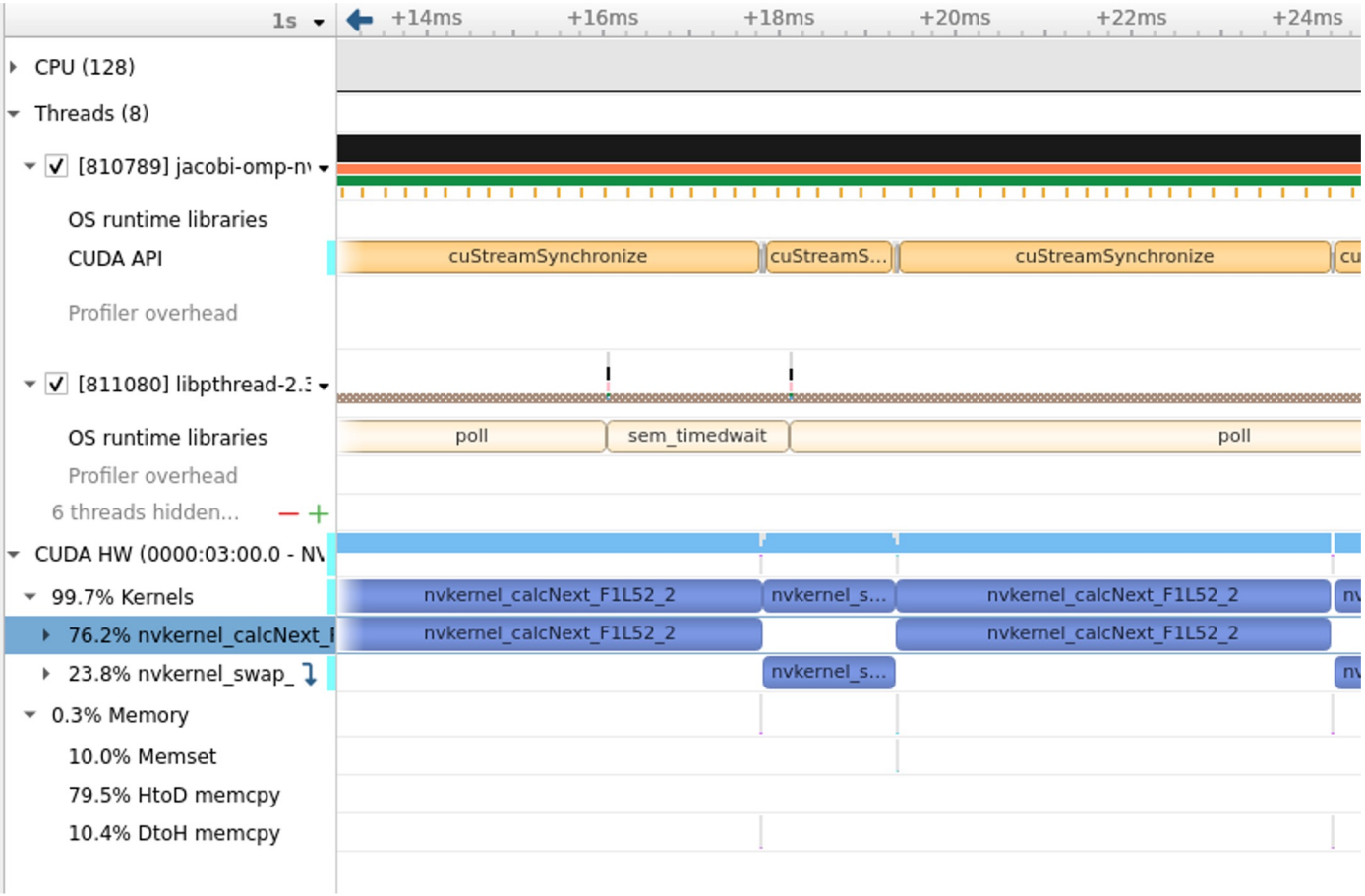
```
$ nvc -fast -mp=gpu -Minfo=all -o jacobi-omp-nvc-copy jacobi.c
initialize:
     41, Generated vector simd code for the loop
calcNext:
     52, #omp target teams distribute parallel for
         52, Generating "nvkernel_calcNext_F1L52_2" GPU kernel
             Loop parallelized across teams and threads(128),
schedule(static)
             Generating reduction(max:error)
     52, Generating map(tofrom:A[:n*m],Anew[:n*m],error)
         Loop not vectorized/parallelized: not countable
     54, Generated vector simd code for the loop containing
reductions
     60, Loop not vectorized/parallelized: not countable
swap:
     65, #omp target teams distribute parallel for
         65, Generating "nvkernel_swap_F1L65_6" GPU kernel
         68, Loop parallelized across teams and threads(128),
schedule(static)
     65, Generating map(tofrom:Anew[:n*m],A[:n*m])
     70, Memory copy idiom, loop replaced by call to __c_mcopy8
main:
    113, initialize inlined, size=10 (inline) file jacobi.c (37)
          41, Loop not fused: function call before adjacent loop
              Generated vector simd code for the loop
    122, Generating map(to:Anew[:m*n],A[:m*n])
         Loop not vectorized/parallelized: potential early exits
    137, deallocate inlined, size=2 (inline) file jacobi.c (78)
```

```
Jacobi relaxation
Calculation: 4096 x 4096
mesh
    0, 0.250000
  100, 0.002397
  200, 0.001204
  300, 0.000804
  400, 0.000603
  500, 0.000483
  600, 0.000403
  700, 0.000345
  800, 0.000302
  900, 0.000269
 total: 6.215937 s
```

Using OpenMP offloading:
Accelerated code using parallel and data clauses taking 6.21s on GPUs using OpenMP offloading

OpenACC version 1.58s on GPUs

Using Nsight System




```
nvkernel_calcNext_F1L52_2
Begins: 1.01936s
Ends: 1.02428s (+4.916 ms)
grid:  <<<32, 1, 1>>>
block: <<<128, 1, 1>>>
Launch Type: Regular
Static Shared Memory: 8 bytes
Dynamic Shared Memory: 0 bytes
Registers Per Thread: 56
Local Memory Per Thread: 0 bytes
Local Memory Total: 127,401,984 bytes
Shared Memory executed: 32,768 bytes
Shared Memory Bank Size: 4 B
Theoretical occupancy: 56.25 %
Launched from thread: 810789
Latency: ←6.734 µs
Correlation ID: 180
Stream: Stream 16
```

Using LLVM OpenMP Offloading and NVIDIA A100

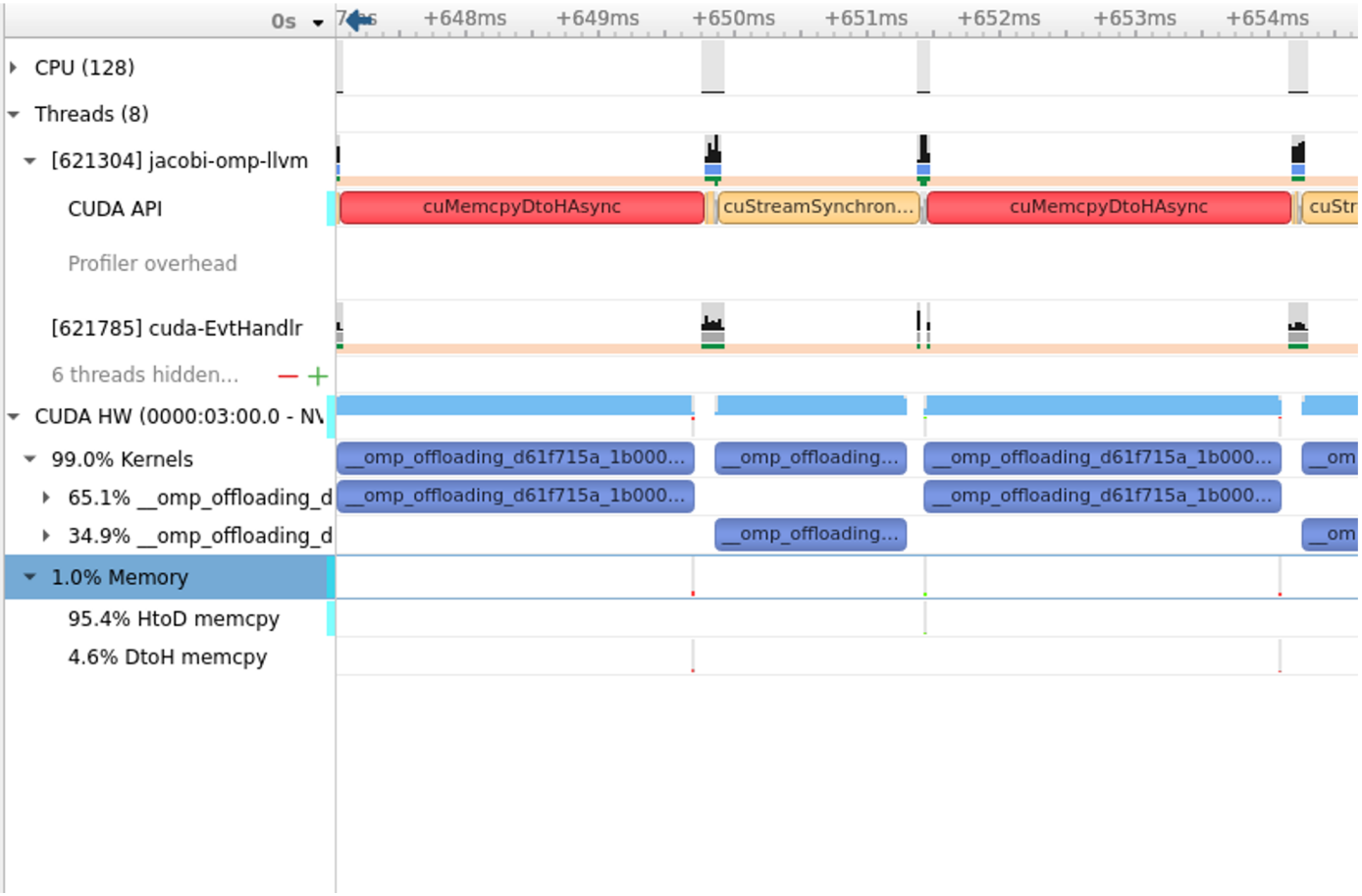
```
$ clang -Ofast -fopenmp --offload-arch=native -g -o jacobi-
omp-llvm-copy jacobi.c
```

```
Jacobi relaxation
Calculation: 4096 x
4096 mesh
    0, 0.250000
  100, 0.002397
  200, 0.001204
  300, 0.000804
  400, 0.000603
  500, 0.000483
  600, 0.000403
  700, 0.000345
  800, 0.000302
  900, 0.000269
 total: 3.666144 s
```

Using OpenMP offloading:
Accelerated code using parallel and data clauses taking 3.66s on GPUs

OpenACC version 1.58s on GPUs

Using Nsight System




```
__omp_offloading_d61f715a_1b000019_calc
Next_l51
Begins: 0.655767s
Ends: 0.658423s (+2.657 ms)
grid:  <<<128, 1, 1>>>
block: <<<32, 1, 1>>>
Launch Type: Regular
Static Shared Memory: 1,720 bytes
Dynamic Shared Memory: 0 bytes
Registers Per Thread: 64
Local Memory Per Thread: 0 bytes
Local Memory Total: 244,187,136 bytes
Shared Memory executed: 102,400 bytes
Shared Memory Bank Size: 4 B
Theoretical occupancy: 50 %
Launched from thread: 621304
Latency: ←7.381 µs
Correlation ID: 165
Stream: Stream 16
```

OpenMP collapse

- In order to improve the code, and obtain more parallelism, the collapse() clause

```
//#pragma acc data copy(A[:n*m]) create(Anew[:n*m])
#pragma omp target data map(to:A[:m*n],Anew[:m*n])
while ( error > tol && iter < iter_max )
{
   error = 0.0;
```

Create data on the GPUs

```
//#pragma acc parallel loop reduction(max:error) copy(A[:m*n],Anew[:m*n])
#pragma omp target teams distribute parallel for map(tofrom:
A[:m*n],Anew[:m*n]) reduction(max:error) collapse(2)
for( int j = 1; j < n-1; j++)
{
  for( int i = 1; i < m-1; i++ )
  {
     Anew[j][i] = 0.25 * ( A[j][i+1] + A[j][i-1] + A[j-1][i] + A[j+1][i]);
     error = fmax( error, fabs(Anew[j][i] - A[j][i]));
   }
}
```

Parallelize and max *reduction*

```
//#pragma acc parallel loop copy(A[:m*n],Anew[:m*n])
#pragma omp target teams distribute parallel for
map(tofrom: A[:m*n],Anew[:m*n]) collapse(2)
for( int j = 1; j < n-1; j++)
{
  for( int i = 1; i < m-1; i++ )
  {
    A[j][i] = Anew[j][i];
  }
}
```

Parallelize second loop

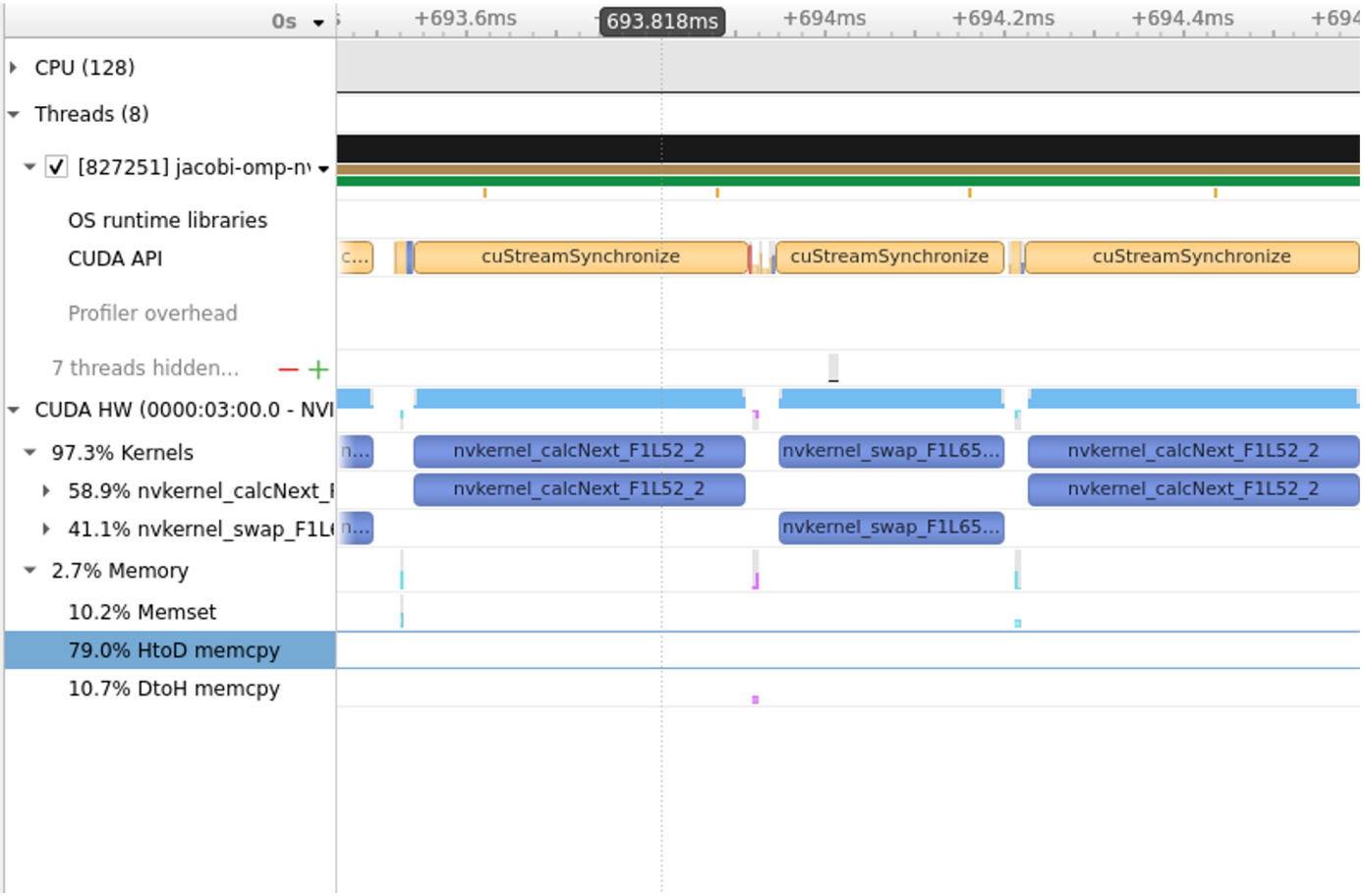
```
$ nvc -fast -mp=gpu -Minfo=all -o jacobi-omp-nvc-copy-collapse
jacobi.c
initialize:
     41, Generated vector simd code for the loop
calcNext:
     52, #omp target teams distribute parallel for
         52, Generating "nvkernel_calcNext_F1L52_2" GPU kernel
             Loop parallelized across teams and threads(128),
schedule(static)
             Generating reduction(max:error)
     52, Generating map(tofrom:A[:n*m],Anew[:n*m],error)
     54, Loop not vectorized/parallelized: not countable
     60, Loop not vectorized/parallelized: not countable
swap:
     65, #omp target teams distribute parallel for
         65, Generating "nvkernel_swap_F1L65_6" GPU kernel
         68, Loop parallelized across teams and threads(128),
schedule(static)
     65, Generating map(tofrom:Anew[:n*m],A[:n*m])
     70, Loop not vectorized/parallelized: not countable
main:
    113, initialize inlined, size=10 (inline) file jacobi.c (37)
          41, Loop not fused: function call before adjacent loop
              Generated vector simd code for the loop
    122, Generating map(to:Anew[:m*n],A[:m*n])
         Loop not vectorized/parallelized: potential early exits
    137, deallocate inlined, size=2 (inline) file jacobi.c (78)
```

```
Jacobi relaxation
Calculation: 4096 x 4096
mesh
    0, 0.250000
  100, 0.002397
  200, 0.001204
  300, 0.000804
  400, 0.000603
  500, 0.000483
  600, 0.000403
  700, 0.000345
  800, 0.000302
  900, 0.000269
 total: 1.604023 s
```

Using OpenMP offloading:
Accelerated code using parallel and data clauses taking 1.6s on GPUs

OpenACC version 1.58s on GPUs

Using Nsight System




```
nvkernel_calcNext_F1L52_2
Begins: 0.69491s
Ends: 0.695277s (+366.846 µs)
grid:  <<<130945, 1, 1>>>
block: <<<128, 1, 1>>>
Launch Type: Regular
Static Shared Memory: 8 bytes
Dynamic Shared Memory: 0 bytes
Registers Per Thread: 46
Local Memory Per Thread: 0 bytes
Local Memory Total: 127,401,984 bytes
Shared Memory executed: 32,768 bytes
Shared Memory Bank Size: 4 B
Theoretical occupancy: 62.5 %
Launched from thread: 827251
Latency: ←6.564 µs
Correlation ID: 167
Stream: Stream 16
```

Using LLVM OpenMP Offloading and NVIDIA A100

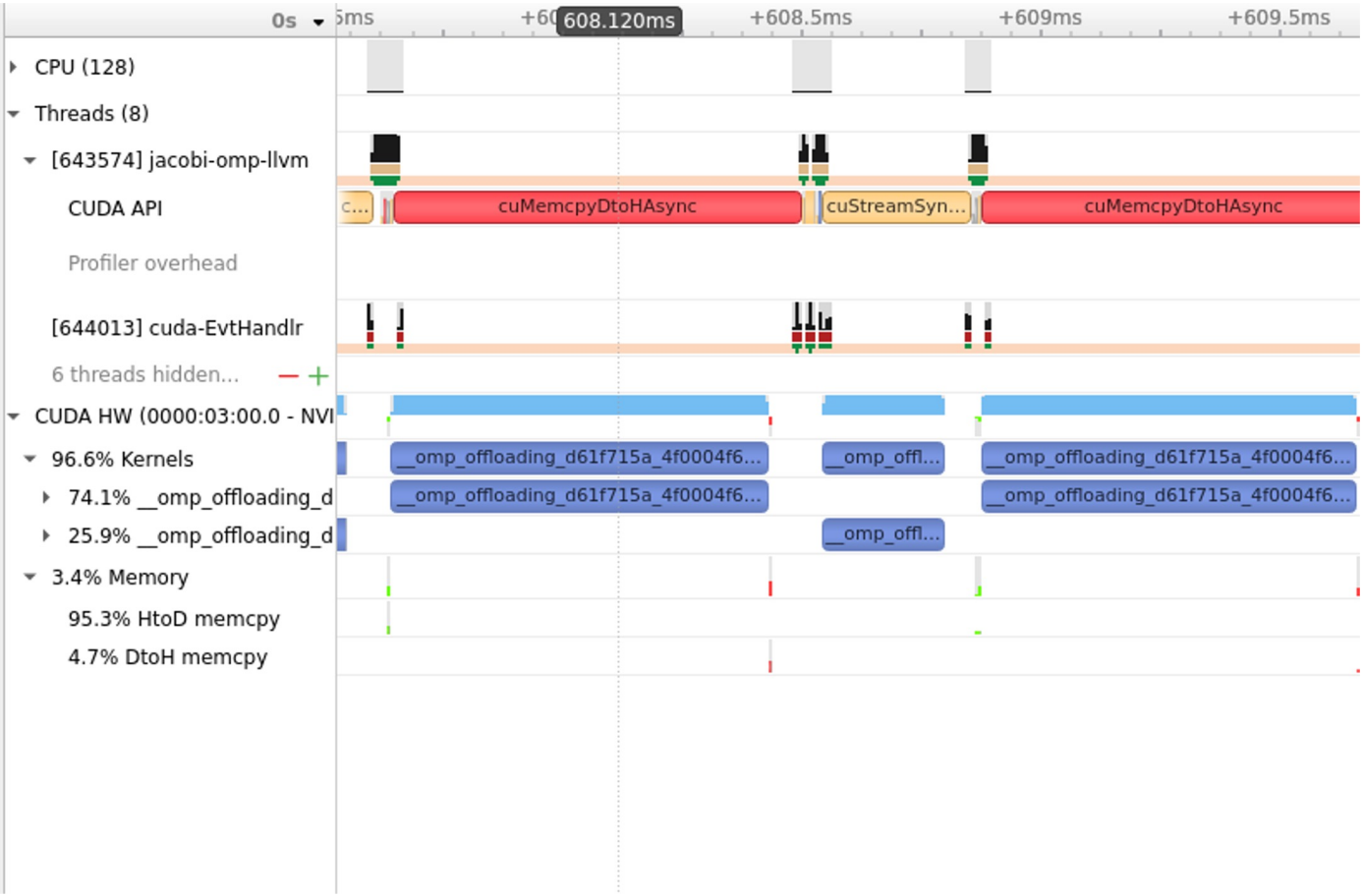
```
$ clang -Ofast -fopenmp --offload-arch=native -g -o jacobi-omp-llvm-copy-collapse jacobi.c
```

```
Jacobi relaxation
Calculation: 4096 x
4096 mesh
    0, 0.250000
  100, 0.002397
  200, 0.001204
  300, 0.000804
  400, 0.000603
  500, 0.000483
  600, 0.000403
  700, 0.000345
  800, 0.000302
  900, 0.000269
 total: 1.222854 s
```

Using OpenMP offloading:
Accelerated code using parallel and data clauses taking 1.22s on GPUs

OpenACC version 1.58s on GPUs

Using Nsight System




```
__omp_offloading_d61f715a_4f0004f6_calc
Next_l51
Begins: 0.610096s
Ends: 0.610878s (+781.915 µs)
grid:  <<<3200, 1, 1>>>
block: <<<128, 1, 1>>>
Launch Type: Regular
Static Shared Memory: 1,720 bytes
Dynamic Shared Memory: 0 bytes
Registers Per Thread: 64
Local Memory Per Thread: 0 bytes
Local Memory Total: 244,187,136 bytes
Shared Memory executed: 65,536 bytes
Shared Memory Bank Size: 4 B
Theoretical occupancy: 50 %
Launched from thread: 643574
Latency: ←7.498 µs
Correlation ID: 165
Stream: Stream 16
```

Table of content

- Laplace Serial code – example
- Parallelization using target parallel for
- Parallelization with target parallel and data constructs
- Parallelization using offload to host =CPU

Use multicore flag

```
$ nvc -fast -mp=multicore -Minfo=all -o jacobi-omp-nvc-host jacobi.c
initialize:
     41, Generated vector simd code for the loop
calcNext:
     52, #omp target teams distribute parallel for
         52, Loop parallelized across teams and threads,
schedule(static)
             Generating reduction(max:error)
     54, Loop not vectorized: unknown
         Loop not vectorized: may not be beneficial for target
     60, Loop not vectorized/parallelized: not countable
swap:
     65, #omp target teams distribute parallel for
         68, Loop parallelized across teams and threads,
schedule(static)
     70, Loop not vectorized: unknown
         Loop not vectorized: may not be beneficial for target
main:
    113, initialize inlined, size=10 (inline) file jacobi.c (37)
          41, Loop not fused: function call before adjacent loop
              Generated vector simd code for the loop
    122, Loop not vectorized/parallelized: potential early exits
    137, deallocate inlined, size=2 (inline) file jacobi.c (78)
```

Set cores

```
$ OMP_NUM_THREADS=128
./jacobi-omp-nvc-host
Calculation: 4096 x 4096
mesh
    0, 0.250000
  100, 0.002397
  200, 0.001204
  300, 0.000804
  400, 0.000603
  500, 0.000483
  600, 0.000403
  700, 0.000345
  800, 0.000302
  900, 0.000269
 total: 2.269870 s
```

Parallelized code using parallel construct took 2.269870 s on 64 core CPU

AMD EPYC 7763 64-Core Processor

Using LLVM OpenMP Offloading and NVIDIA A100

Use multicore flag

Set cores

```
$ clang -Ofast -g -fopenmp --target=x86_64-pc-linux-gnu -o jacobi-omp-llvm-host
```

```
$ OMP_NUM_THREADS=128 jacobi-omp-llvm-host jacobi.c
Jacobi relaxation Calculation: 4096 x 4096 mesh
    0, 0.250000
  100, 0.002397
  200, 0.001204
  300, 0.000804
  400, 0.000603
  500, 0.000483
  600, 0.000403
  700, 0.000345
  800, 0.000302
  900, 0.000269
 total: 2.277123 s
```

Using OpenMP offloading:
Parallelized code using parallel construct took 0.852s on 64 core CPU
AMD EPYC 7763 64-Core Processor

Increasing the size of the mesh size

4096 x 4096 GRID

OpenACC

16384 x 16384 GRID

Accelerated code using parallel and data clauses take **1.589625s** on GPUs

Parallelized code using parallel construct took **0.852060s** on 64 core multicore CPUs

Accelerated code using parallel and data clauses take **9.582822 s** on GPUs

Parallelized code using parallel construct took **356.332373 s** on 64 core multicore CPUs

4096 x 4096 GRID

OpenMP

16384 x 16384 GRID

NVC **1.604023 s**, and LLVM **1.222854s** on GPUs

NVC **2.269870s** and LLVM **2.277123s** on 64 core multicore CPUs

NVC **8.966994 s** and LLVM **12.720154** on GPUs

NVC **372.458698 s** and LLVM **393.499142 s** on 64 core multicore CPUs

	OpenMP		OpenACC
	nvc	**llvm**	**nvc**
Serial	23.364053s	25.557923	23.364053s
parallel for	89.513495	242.194770	84.040213
Teams distribute	89.992197	100.463713	
copy teams parallel for	6.215937	3.666144	1.589625
copy teams parallel for collapse	1.604023	1.222854	
multicore	2.269870	2.277123	0.852060