

Introduction to NVIDIA Profilers on Summit

Tom Papatheodore Oak Ridge Leadership Computing Facility (OLCF)

Jeff Larkin NVIDIA

Oak Ridge National Laboratory - April 11, 2019

ORNL is managed by UT-Battelle, LLC for the US Department of Energy



Outline

- System Access & Local NVIDIA Toolkit Install
- Cloning Repository & Setting Up Environment
- A Simple Example: Vector Addition
- Jacobi Iteration
 - Serial
 - Single GPU
 - Single GPU (explicit data movement)
 - Multiple GPU (OpenMP + OpenACC)
- Redundant Matrix Multiply
 - Dealing with multiple MPI ranks
 - Basic annotation of CPU/GPU activities with NVTX
 - Unified Memory
 - Remote kernel Analysis



2



System Access & Local NVIDIA Toolkit Install

If you want to follow along with the handson portions of this tutorial, you will need

- to have access to a Summit-like system
- to have a local install of NVIDIA Toolkit (v10+)



Access to the Ascent Training System

If you do not already have access to Summit, you can use the Ascent training system for this tutorial. Please visit the following url for instructions on how to do so:

https://www.olcf.ornl.gov/for-users/system-user-guides/summit/summit-user-guide/#obtaining-access-to-ascent

For the Project ID field, please use GEN121

In-Person Attendees Only!

Once you have access, you can login as follows:

\$ ssh USERNAME@login1.ascent.olcf.ornl.gov (This will drop you into /ccsopen/home/USERNAME)



Local Installation of NVIDIA Toolkit (version 10+)

To ensure compatibility, please install NVIDIA Toolkit version 10+. Please visit the following url to download the toolkit:

https://developer.nvidia.com/cuda-downloads

Make sure to download the appropriate version for your local operating system.

NOTE: You do not need an NVIDIA GPU on your local machine to install the toolkit and use the profiler.





CAK RIDGE

6

Summit Node

(2) IBM Power9 + (6) NVIDIA Volta V100





7

NVLink2 (50 GB/s)

1 (900 GB/s)

Cloning Repository & Setting Up Environment



Log Into Ascent and Change Directory

From the command line:

\$ ssh USERNAME@login1.ascent.olcf.ornl.gov

(This will drop you into the directory /ccsopen/home/USERNAME)



Change to the following directory:

\$ cd /gpfs/wolf/gen121/scratch/USERNAME

On Summit, you should navigate to the corresponding Alpine/GPFS directory for your project (since you need read/write access from the compute nodes).

E.g. /gpfs/alpine/PROJID/scratch/USERNAME



Clone Repository and Set up Programming Environment

Once in the appropriate directory from step 2, clone the git repository:
 \$ git clone https://github.com/olcf/nvidia profilers.git

$\mathbf{\mathbf{9}}$ cd into directory:

\$ cd nvidia_profilers

5

Run script to set up environment for the tutorial:

\$ source environment_ascent.sh On Summit, source the environment_summit.sh file instead.

At this point, you prompt should look like this:

[USERNAME@login1: /gpfs/wolf/gen121/scratch/USERNAME/nvidia_profilers]\$



A Simple Example: Vector Addition



```
#include <stdio.h>
#define N 1048576
global void add vectors(int *a, int *b, int *c){
   int id = blockDim.x * blockIdx.x + threadIdx.x;
   if(id < N) c[id] = a[id] + b[id];
}
int main() {
    size t bytes = N*sizeof(int);
   int *A = (int*)malloc(bytes);
   int *B = (int*)malloc(bytes);
   int *C = (int*)malloc(bytes);
   int *d A, *d B, *d C;
    cudaMalloc(&d A, bytes);
   cudaMalloc(&d B, bytes);
    cudaMalloc(&d C, bytes);
    for(int i=0; i<N; i++) {</pre>
       A[i] = 1;
       B[i] = 2;
    }
    cudaMemcpy(d A, A, bytes, cudaMemcpyHostToDevice);
    cudaMemcpy(d B, B, bytes, cudaMemcpyHostToDevice);
    int thr per blk = 256;
    int blk in grid = ceil( float(N) / thr per blk );
    add_vectors<<< blk_in_grid, thr_per_blk >>>(d_A, d_B, d_C);
    cudaMemcpy(C, d_C, bytes, cudaMemcpyDeviceToHost);
    free(A);
    free(B);
    free(C);
    cudaFree(d A);
   cudaFree(d B);
    cudaFree(d C);
```

CUDA Vector Addition



return 0;

}

#include <stdio.h>
#define N 1048576

| <pre>globalvoid add_vectors(int *a, int *b, int *c){ int id = blockDim.x * blockIdx.x + threadIdx.x; if(id < N) c[id] = a[id] + b[id]; }</pre> | Vector addition kernel (GPU) |
|---|--|
| <pre>int main() { size_t bytes = N*sizeof(int);</pre> | |
| <pre>int *A = (int*)malloc(bytes); int *B = (int*)malloc(bytes); int *C = (int*)malloc(bytes);</pre> | Allocate memory on CPU |
| <pre>int *d_A, *d_B, *d_C; cudaMalloc(&d_A, bytes); cudaMalloc(&d_B, bytes); cudaMalloc(&d_C, bytes);</pre> | Allocate memory on GPU |
| <pre>for(int i=0; i<n; a[i]="1;" b[i]="2;" i++){="" pre="" }<=""></n;></pre> | Initialize arrays on CPU |
| <pre>cudaMemcpy(d_A, A, bytes, cudaMemcpyHostToDevice); cudaMemcpy(d_B, B, bytes, cudaMemcpyHostToDevice);</pre> | Copy data from CPU to GPU |
| <pre>int thr_per_blk = 256; int blk_in_grid = ceil(float(N) / thr_per_blk); add_vectors<<< blk_in_grid, thr_per_blk >>>(d_A, d_B, d_C);</pre> | Set configuration parameters and launch kernel |
| <pre>cudaMemcpy(C, d_C, bytes, cudaMemcpyDeviceToHost);</pre> | Copy data from GPU to CPU |
| <pre>free(A); free(B); free(C); cudaFree(d_A); cudaFree(d_B); cudaFree(d_C);</pre> | Free memory on CPU and GPU |

CUDA Vector Addition



return 0;

}

Vector Addition Example

- \$ cd vector_addition/cuda
- \$ make





Vector Addition Example (nvprof results – text only)

From vec_add_cuda.JOBID:

==174655== Profiling result:

| Туре | Time(%) | Time | Calls | Avg | Min | Max | Name |
|-----------------|---------|----------|-------|----------|----------|----------|--|
| GPU activities: | 56.25% | 463.36us | 2 | 231.68us | 229.66us | 233.70us | [CUDA memcpy HtoD] |
| | 41.59% | 342.56us | 1 | 342.56us | 342.56us | 342.56us | [CUDA memcpy DtoH] |
| | 2.16% | 17.824us | 1 | 17.824us | 17.824us | 17.824us | <pre>add_vectors(int*, int*, int*)</pre> |
| API calls: | 99.35% | 719.78ms | 3 | 239.93ms | 1.1351ms | 717.50ms | cudaMalloc |
| | 0.23% | 1.6399ms | 96 | 17.082us | 224ns | 670.19us | cuDeviceGetAttribute |
| | 0.17% | 1.2559ms | 3 | 418.64us | 399.77us | 454.40us | cudaFree |
| | 0.16% | 1.1646ms | 3 | 388.18us | 303.13us | 550.07us | cudaMemcpy |
| | 0.06% | 412.85us | 1 | 412.85us | 412.85us | 412.85us | cuDeviceTotalMem |
| | 0.03% | 182.11us | 1 | 182.11us | 182.11us | 182.11us | cuDeviceGetName |
| | 0.00% | 32.391us | 1 | 32.391us | 32.391us | 32.391us | cudaLaunchKernel |
| | 0.00% | 3.8960us | 1 | 3.8960us | 3.8960us | 3.8960us | cuDeviceGetPCIBusId |
| | 0.00% | 2.2920us | 3 | 764ns | 492ns | 1.1040us | cuDeviceGetCount |
| | 0.00% | 1.4090us | 2 | 704ns | 423ns | 986ns | cuDeviceGet |



Now, transfer the .nvvp file from Ascent to your local machine to view in NVIDIA Visual Profiler.

From your local system:

\$ scp USERNAME@login1.ascent.ccs.ornl.gov:/path/to/file/remote /path/to/desired/location/local



16

1 File->Import

| K NVIDIA Visual Profiler | File | View | Window | Help |
|--------------------------|----------|---------|-------------|------|
| • • • | * | New Se | ssion 8 | θN |
| * 🔒 🗟 | Op | en | 9 | fo B |
| | Clo | ne Sess | sion ជំន | βС |
| | | Save | 9 | #S |
| | | Save As | | |
| | Q | Save Al | 습 () | #S |
| | 2 | Import | . 8 | HE I |

Select
Import profile data generated by nvprof.
Select an import source:
Command-line Profiler
Nvprof

Select "Single Process" then "Next >"

Import Nvprof Data

Select "Nvprof" then "Next >"

Import Profile Data for Single Process

Select one nvprof profile file containing timeline data and zero or more addition nvprof profile files containing event and metric values.

| | Profile Files | Timeline Options | |
|---------------------|-----------------------------|-------------------------|--------------------|
| Connection: | Local | \$ | Manage connections |
| Timeline data file: | Enter nvprof profile file o | ontaining timeline data | Browse |



17

CAK RIDGE National Laboratory

Click "Browse" next to "Timeline data file" to locate the .nvvp file on your local system, then click "Finish"

To zoom in on a specific region, hold Ctrl + left-click and drag mouse (Cmd for Mac)





2.16% 17.824us

| | | | | | NVIDIA Visual P | rofiler | | | | | | |
|---|---|--|--|----------------|-----------------|-------------|-------------|-----------|------------|----------|--------|----------|
|) 🖫 🖳 🖳 🖏 🗣 🔍 - | ₹ E F K K 5 8 8 & . | | | | | | | | | | | |
| *vec_add_cuda.h49n16.120 | 095.nvvp ⊠ | | | | | | | | | | | |
| | .9 ms 805 ms | 805.1 ms | 805.2 ms | 805.3 ms | 805.4 ms | 805.5 ms | 805.6 ms 80 | 5.7 ms | 805.8 ms | 805.9 ms | 806 ms | 806.1 ms |
| Process "run" (174655) | | | | | | | | | | | | |
| Thread 288656 | | | | | | | | | | | | |
| Runtime API | cudaM | emcpy | | cuda | aMemcpy | | | | cudaMen | псру | | |
| Driver API | | | | | | | | | | | | |
| Profiling Overhead | | | | | | | | | | | | |
| [0] Tesla V100-SXM2-16GB | | | | | | | | | | | | |
| Context 1 (CUDA) | | | | | | | | | | | | |
| MemCpy (HtoD) | | Memcpy HtoD [sync | | | Memcpy HtoD [| syncj | | Marram Di | all formal | | | |
| Compute | | | | | | | | Memcpy Dt | OH [SYNC] | | | |
| cudaMemcpy cudaMemcpy // Set ext // tl // bi int thr_po int blk_in // Launch add_vectos // Copy da cudaMemcpy | <pre>v(d_A, A, bytes, cudak v(d_B, B, bytes, cudak ecution configuration nr_per_blk: number of lk_in_grid: number of er_blk = 256; n_grid = ceil(float() kernel rs<<< blk_in_grid, the ata_from device array v(C, d_C, bytes, cudak</pre> | <pre>lemcpyHostTo lemcpyHostTo parameters CUDA thread blocks in a t) / thr_per c_per_blk >> d_C to host lemcpyDevice</pre> | Device); Device); s per grid rid _bIk); >(d_A, d_B, array C ToHost); | block d_C); | | | | | | | | |
| | | | | | | | | | | | | |
| | Tvpe | Time(%) | Time | e Call | .s A | va Mi | n Max | Name | | | | |
| | GPIL activities. | 56 25% | 463 36110 | 9 | 2 231 68 | us 229 6611 | s 233 7011e | | memony | HtoD1 | | |
| | | 11 599 | 3/2 5611 | | 1 3/2 56 | 13 225.000 | a 3/2 5611a | ע עניס ן | memcny | D+0H] | | |
| | | ヨエ・リンつ | J74.J0U3 | 3 | _ JHZ.JU | UD JHZ.JUU | S JHZ.JUUS | ICUDA | | | | |

1 17.824us 17.824us 17.824us add_vectors(int*, int*, int*)





CAK RIDGE

20





21

Jacobi Iteration



Jacobi Iteration – Problem Description

Use Jacobi Iteration to solve 2D Poisson equation with periodic boundary conditions:

 $\Delta A(y,x) = e^{-10(x^*x + y^*y)}$



Execute a Jacobi Step on the Inner Points

 $A_{k+1}(iy,ix) = -0.25 * (rhs(iy,ix) - (A_k(iy,ix-1) + A_k(iy,ix+i) + A_k(iy-1,ix) + A_k(iy+1,ix)))$





Copy Values of Anew to A

```
for (int iy = 1; iy < NY-1; iy++)
{
    for( int ix = 1; ix < NX-1; ix++ )
    {
        A[iy][ix] = Anew[iy][ix];
    }
}</pre>
```



Apply Periodic Boundary Conditions

```
//Periodic boundary conditions
for( int ix = 1; ix < NX-1; ix++ )
{
       A[0][ix] = A[(NY-2)][ix];
        A[(NY-1)][ix] = A[1][ix];
}
for (int iy = 1; iy < NY-1; iy++)</pre>
{
       A[iy][0] = A[iy][(NX-2)];
        A[iy][(NX-1)] = A[iy][1];
}
```



Serial Version jacobi/1_serial



Serial Runtime

Compile the code

\$ make

pgcc -Minfo -fast -c poisson2d.c main:

- 54, Generated vector simd code for the loop FMA (fused multiply-add) instruction(s) generated
 65, Memory zero idiom, loop replaced by call to __c mzero8
- 84, FMA (fused multiply-add) instruction(s) generated
- 90, Generated vector simd code for the loop containing reductions
- 100, Memory copy idiom, loop replaced by call to $_c_mcopy8$
- 107, Loop not fused: dependence chain to sibling loop Generated vector simd code for the loop Residual loop unrolled 2 times (completely unrolled)
- 112, Loop not fused: function call before adjacent loop Loop unrolled 8 times

pgcc -Minfo -fast poisson2d.o -o run

Run the code (on single CPU core)

\$ bsub submit.lsf

Job <11536> is submitted to default queue <batch>.

\$ jobstat

| | | Running J | obs: 1 | (1 of 16 | nodes, | 6.25%) | |
|-----------------|----------|-----------|---------|----------|--------|----------|---------|
| JobId | Username | Project | Nodes | Remain | StartT | ime | JobName |
| 11536 | t4p | GEN117 | 1 | 8:48 | 02/24 | 10:03:14 | serial |
| | | Eligible | Jobs: 0 | | | | |
| Blocked Jobs: 0 | | | | | | | |

\$ less serial.11536

800, 0.249524 900, 0.249464

· · · . . .

Elapsed Time (s): 94.9856

Jacobi relaxation Calculation: 4096 x 4096 mesh 0, 0.250000 100, 0.249940 200, 0.249880 300, 0.249821 400, 0.249761 500, 0.249702 600, 0.249642 700, 0.249583

(Enter q to quit/exit less)



Single GPU Version jacobi/2_single_gpu



Difference From Serial Version

 Added OpenACC pragmas to inform compiler where to offload work to GPU

#pragma acc kernels

 Added (optional) serial version to compare with timing and results of GPU version

```
// Set to 1 to run serial test, otherwise 0
int serial_test = 0;
```



Runtime of Single GPU Version

Compile the code

\$ make

| pgcc | -acc | -Minf | fo=acc | -ta= | =tesla:cc | 70 -: | fast | -c | poiss | on2c | d.d | С |
|-------|------|-------|--------|------|-----------|-------|------|----|-------|------|-----|---|
| main: | : | | | | | | | | | | | |
| | | - | | | | | | - | | | | |

| 1 | <pre>L1/, Generating implicit copyin(A[:][:],rhs[1:4094][1:4094])</pre> |
|------|---|
| | Generating implicit copyout(Anew[1:4094][1:4094]) |
| 1 | 18, Loop is parallelizable |
| 1 | 20, Loop is parallelizable |
| | Accelerator kernel generated |
| | Generating Tesla code |
| | 118, #pragma acc loop gang, vector(4) /* blockIdx.y threadIdx.y */ |
| | 120, #pragma acc loop gang, vector(32) /* blockIdx.x threadIdx.x */ |
| | 124, Generating implicit reduction(max:error) |
| 1 | 28, Generating implicit copyin(Anew[1:4094][1:4094]) |
| | Generating implicit copyout(A[1:4094][1:4094]) |
| 1 | .29, Loop is parallelizable |
| 1 | .31, Loop is parallelizable |
| | Accelerator kernel generated |
| | Generating Tesla code |
| | 129, #pragma acc loop gang, vector(4) /* blockIdx.y threadIdx.y */ |
| | 131, #pragma acc loop gang, vector(32) /* blockIdx.x threadIdx.x */ |
| 1 | <pre>.38, Generating implicit copy(A[:][1:4094])</pre> |
| 1 | .39, Loop is parallelizable |
| | Accelerator kernel generated |
| | Generating Tesla code |
| | 139, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x * |
| 1 | .44, Generating implicit copy(A[1:4094][:]) |
| 1 | .45, Loop is parallelizable |
| | Accelerator kernel generated |
| | Generating Tesla code |
| | 145, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x * |
| pgcc | -acc -Minfo=acc -ta=tesla:cc70 -fast poisson2d.o -o run |
| | |

Run the code (on single GPU)

\$ bsub submit.lsf

/

\$ less single_gpu.JOBID

Jacobi relaxation Calculation: 4096 x 4096 mesh 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 Elapsed Time (s) - Parallel:



31

Runtime of Single GPU Version

Compile the code

\$ make

| pgcc | -acc | -Minfo=acc | -ta=tesla:cc70 | -fast | -c | poisson2d.c |
|-------|------|------------|----------------|-------|----|-------------|
| main: | | | | | | |

117, Generating implicit copyin(A[:][:],rhs[1:4094][1:4094]) Generating implicit copyout (Anew [1:4094] [1:4094]) 118, Loop is parallelizable 120, Loop is parallelizable Accelerator kernel generated Generating Tesla code 118, #pragma acc loop gang, vector(4) /* blockIdx.y threadIdx.y */ 120, #pragma acc loop gang, vector(32) /* blockIdx.x threadIdx.x */ 124, Generating implicit reduction(max:error) 128, Generating implicit copyin(Anew[1:4094][1:4094]) Generating implicit copyout (A[1:4094][1:4094]) 129, Loop is parallelizable 131, Loop is parallelizable Accelerator kernel generated Generating Tesla code 129, #pragma acc loop gang, vector(4) /* blockIdx.y threadIdx.y */ 131, #pragma acc loop gang, vector(32) /* blockIdx.x threadIdx.x */ 138, Generating implicit copy(A[:][1:4094]) 139, Loop is parallelizable Accelerator kernel generated Generating Tesla code 139, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */ 144, Generating implicit copy(A[1:4094][:]) 145, Loop is parallelizable Accelerator kernel generated Generating Tesla code 145, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */ pqcc -acc -Minfo=acc -ta=tesla:cc70 -fast poisson2d.o -o run

Run the code (on single GPU)

\$ bsub submit.lsf

\$ less single_gpu.JOBID

Jacobi relaxation Calculation: 4096 x 4096 mesh 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 Elapsed Time (s) - Parallel: 127.2326

Why are we slower than serial version??

How can we answer such questions?



32

Using NVIDIA's NVProf Profiler, we see...

\$ bsub submit.lsf (jsrun --smpiargs="none" -n1 -c1 -g1 -a1 nvprof -s -o single_gpu.%h.\${LSB_JOBID}.nvvp ./run)

\$ less single gpu.JOBID

==56446== NVPROF is profiling process 56446, command: ./run ==56446== Profiling application: ./run

Jacobi relaxation Calculation: 4096 x 4096 mesh 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 Elapsed Time (s) - Parallel: 130.9012

==56446== Profiling result: Type Time(%)

GPU activities:

| Time(% |) Time | Calls | Avg | Min | Max | Name |
|--------|------------|-------|----------|----------|----------|--------------------|
| 53.55 | % 14.4180s | 41000 | 351.66us | 1.3110us | 382.72us | [CUDA memcpy HtoD] |
| 42.84 | % 11.5335s | 33000 | 349.50us | 1.7590us | 362.53us | [CUDA memcpy DtoH] |
| 2.01 | § 541.55ms | 1000 | 541.55us | 539.61us | 546.01us | main_120_gpu |
| 1.38 | % 372.18ms | 1000 | 372.18us | 369.47us | 376.64us | main_131_gpu |
| 0.19 | % 49.816ms | 1000 | 49.815us | 48.448us | 51.231us | main_124_gpured |
| 0.02 | % 6.1174ms | 1000 | 6.1170us | 5.7270us | 6.9760us | main_145_gpu |
| 0.01 | % 2.1649ms | 1000 | 2.1640us | 1.8880us | 2.8480us | main_139_gpu |

Do we really need all these data transfers?

Let's look at visual output (and compiler output) to see what's going on...



Transfer .nvvp file from Ascent/Summit to local system

From your local system:

\$ scp USERNAME@login1.ascent.ccs.ornl.gov:/path/to/file/remote /path/to/desired/location/local



Using NVIDIA's Visual Profiler, we see...



35

National Laboratory

Where are arrays actually needed?

CAK RIDGE

National Laboratory

36



jacobi/2_single_gpu
Single GPU Version with Data Regions jacobi/3_single_gpu_data



Difference From Initial GPU Version

Added a data region around while loop

```
#pragma acc data ...
{
   while loop
}
```

• Still have (optional) serial version to compare with timing and results of GPU version

```
// Set to 1 to run serial test, otherwise 0
int serial_test = 0;
```



Runtime of Single GPU Version with Data Directives

Compile the code

\$ make

- pgcc -acc -Minfo=acc -ta=tesla:cc70 -fast -c poisson2d.c
 main:
 - 112, Generating copyin(rhs[:][:])
 Generating create(Anew[:][:])
 Generating copy(A[:][:])
 - 121, Loop is parallelizable
 - 123, Loop is parallelizable Accelerator kernel generated
 - Generating Tesla code
 - 121, #pragma acc loop gang, vector(4) /* blockIdx.y threadIdx.y */
 - 123, #pragma acc loop gang, vector(32) /* blockIdx.x threadIdx.x */
 - 127, Generating implicit reduction(max:error)
 - 132, Loop is parallelizable
 - 134, Loop is parallelizable Accelerator kernel generated
 - Generating Tesla code
 - 132, #pragma acc loop gang, vector(4) /* blockIdx.y threadIdx.y */
 - 134, #pragma acc loop gang, vector(32) /* blockIdx.x threadIdx.x */
 - 142, Loop is parallelizable
 - Accelerator kernel generated
 - Generating Tesla code
 - 142, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
 - 148, Loop is parallelizable
 - Accelerator kernel generated
 - Generating Tesla code
- 148, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */ pgcc -acc -Minfo=acc -ta=tesla:cc70 -fast poisson2d.o -o run

Run the code (on single GPU)

\$ bsub submit.lsf

\$ less single_gpu_data.JOBID

Jacobi relaxation Calculation: 4096 x 4096 mesh 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
- Elapsed Time (s) Parallel:



39

Using NVIDIA's NVProf Profiler, we see...

\$ bsub submit.lsf (jsrun --smpiargs="none" -n1 -c1 -g1 -a1 nvprof -s -o single_gpu_data.%h.\${LSB_JOBID}.nvvp ./run)

\$ less single_gpu_data.JOBID

==139388== NVPROF is profiling process 139388, command: ./run ==139388== Profiling application: ./run





Using NVIDIA's Visual Profiler, we see...



We have eliminated the unnecessary data transfers.



41

jacobi/3_single_gpu_data

Multiple GPU Version (OpenMP + OpenACC) jacobi/4_multiple_gpu_openmp



• Each OpenMP thread calculates its own loop bounds for its portion of the domain and uses its own GPU.

OpenMP Thread $0 \Rightarrow$ GPU 0

OpenMP Thread $1 \Rightarrow$ GPU 1

OpenMP Thread $2 \Rightarrow$ GPU 2

OpenMP Thread $3 \Rightarrow$ GPU 3



jacobi/4_multiple_gpu_openmp

#pragma omp parallel default(shared) firstprivate(num_threads, thread_num){}



```
#pragma omp master
{
    // Set rhs
    for (int iy = 1; iy < NY-1; iy++)
    {
        for (int ix = 1; ix < NX-1; ix++ )
        {
            const double x = -1.0 + (2.0*ix/(NX-1));
            const double y = -1.0 + (2.0*iy/(NY-1));
            rhs[iy][ix] = exp(-10.0*(x*x + y*y));
        }
    }
    /* pragma omp master */
    COAK RIDGE
    National Laboratory
</pre>
```

44

Only the master thread needs to set value of rhs

```
jacobi/4_multiple_gpu_openmp
```

Thread 0's copy of its rows of A (on GPU 0) CPU copy of A

Thread 1's copy of its rows of A (on GPU 1)

jacobi/4 multiple gpu openmp



```
#pragma acc kernels
   for (int iy = iy_start; iy < iy_end; iy++)</pre>
                                                                  After GPUs update their values
   {
      for( int ix = ix start; ix < ix end; ix++ )</pre>
                                                                     of A, the CPU copy is no
      {
                                                                          longer correct
         Anew[iy][ix] = -0.25 * (rhs[iy][ix] - (A[iy][ix+1] + A[iy][ix-1]))
                                       + A[iy-1][ix] + A[iy+1][ix] ));
         error = fmax( error, fabs(Anew[iy][ix]-A[iy][ix]));
 Thread 0's copy of its rows of A
                                       CPU copy of A
                                                                Thread 1's copy of its rows of A
                                                                         (on GPU 1)
          (on GPU 0)
                                    CAK RIDGE
```

jacobi/4 multiple gpu openmp

46 **WAK KIDGE** National Laboratory

Recall that boundary conditions must be updated for A matrix as a whole

- But each GPU only has its rows of A
- So some data must be passed back to CPU

CPU copy of A





#pragma acc update self(A[iy_start:1][0:NX], A[(iy_end-1):1][0:NX])

Each thread updates the "shared" CPU copy of A with its "2nd-to-top" row and "2nd-to-bottom" row



| es | <pre>if(0 == {</pre> | (iy_start-1)) |) | | | |
|-----------------------|----------------------|---------------------------|---------|---------------------|---------------|----|
| op/Botto 3oundarie | for { | (int ix = 1; A[0][ix] | ix = | < NX-1; A[(NY-2) | ix++][ix] |); |
| <u> </u> | } | | | | | |

if((NY-1) == (iy end))Boundaries { for (int ix = 1; ix < NX-1; ix++) Side A[(NY-1)][ix] = A[1][ix];

Only the threads with $(0 == (iy_start-1))$ and $((NY-1) == (iy_end))$ perform the boundary updates

Thread 0's copy of its rows of A (on GPU 0) CPU copy of A

Thread 1's copy of its rows of A (on GPU 1)





49

#pragma acc update device(A[(iy_start-1):1][0:NX], A[iy_end:1][0:NX])

Each thread updates its "top" row and "bottom" row from the new values of the CPU copy of A



Runtime of Multi-GPU Version (with Data Directives)

Compile the code

\$ make

AK RIDGE

National Laboratory

51

| pgcc -acc -Minfo=acc -ta=tesla:cc70 -mp -fast -c poisson2d.c poisson2d_serial: main: | \$ si |
|--|----------|
| 103, Generating implicit copyout(A[:][:],A ref[:][:]) | |
| 104, Loop is parallelizable | |
| 106, Loop is parallelizable | |
| Accelerator kernel generated | |
| Generating Tesla code | |
| 104, #pragma acc loop gang, vector(4) /* blockIdx.y threadIdx.y */ | |
| 106, #pragma acc loop gang, vector(32) /* blockIdx.x threadIdx.x */ | |
| 167, Generating copyin(rhs[iv_start:iv_end-iv_start][:]) | |
| Generating create (Anew[iv start:iv end-iv start][:]) | |
| Generating copy(A[iy start-1:iy end-iy start+2][:]) | |
| 181, Loop is parallelizable | Pa |
| 183, Loop is parallelizable | |
| Accelerator kernel generated | |
| Generating Tesla code | |
| 181, #pragma acc loop gang, vector(4) /* blockIdx.y threadIdx.y */ | |
| 183, #pragma acc loop gang, vector(32) /* blockIdx.x threadIdx.x */ | |
| 187, Generating implicit reduction(max:error) | |
| 200, Loop is parallelizable | |
| 202, Loop is parallelizable | |
| Accelerator kernel generated | |
| Generating Tesla code | |
| 200, #pragma acc loop gang, vector(4) /* blockIdx.y threadIdx.y */ | E |
| 202, #pragma acc loop gang, vector(32) /* blockIdx.x threadIdx.x */ | |
| <pre>211, Generating update self(A[iy_start][:],A[iy_end-1][:])</pre> | |
| <pre>230, Generating update device(A[iy_start-1][:],A[iy_end][:])</pre> | |
| 231, Loop is parallelizable | |
| Accelerator kernel generated | |
| Generating Tesla code | |
| 231, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */ | 1 |
| pqcc -acc -Minfo=acc -ta=tesla:cc70 -mp -fast poisson2d.o -o run | |

Run the code (on 2 GPUs)

\$ bsub submit2.1sf

less multi gpu 20mp ingle-GPU 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 arallel 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

lapsed Time (s) - Serial: 1.0990, Parallel: 0.6692, Speedup: 1.6424

jacobi/4_multiple_gpu_openmp

Using NVIDIA's Visual Profiler, we see...

OpenMP Thread 0 (GPU 0)

| | | | | | | NIV | | | | | | | | | |
|----------------------------|------------|-----------|--------------|----------------|------------|-----------|-------------|----------|------------|-----------------|------------|-----------|------------|----------|-----------|
| | | K S P 🙏 | • | | | 144 | | | | | | | | | |
| t •multiple_pu_2omp.nvvp ∺ | 3 | | | | | | | | | | | | | | |
| | 2197.65 ms | 2197.7 ms | 2197.75 ms | 2197.8 ms | 2197.85 ms | 2197.9 ms | 2197.95 ms | 2198 ms | 2198.05 ms | 2198.1 ms | 2198.15 ms | 2198.2 ms | 2198.25 ms | 2198.3 r | ns 2198.3 |
| + Process "run" (14531) | 1 | 1 | | 1 | 1 | 1 | | 1 | | 1 | | 1 | | | |
| [0] Tesla V100-SXM2-16GB | | | | | | | | | | | | | | | |
| Context 1 (CUDA) | | | | | Or | ne itera | tion of th | ne while | e loon | | | | | | |
| L 🍸 MemCpy (HtoD) | | | | | 01 | | | | oloop | | | | | | |
| ⊢ 🍸 MemCpy (DtoH) | | | | | | | | | | | | | | | |
| + Compute | | | poisson2d_se | rial_1F1L70_18 | 3_gpu | - | poisson2d | | poisson | 2d_serial_1F1L7 | 0_202_gpu | | | | |
| Streams | | | | | | | | | | | | | | | _ |
| Stream 25 | | _ | poisson2d_se | rial_1F1L70_18 | 3_gpu | - | poisson2d | | poisson | 2d_serial_1F1L7 | 0_202_gpu | | | | |
| [1] Tesla V100-SXM2-16GB | | | | | | | | | | | | | | | |
| Context 2 (CUDA) | | | | | | | | | | | | | | | |
| └ 🐺 MemCpy (HtoD) | | | | | | | | | | | | | | | |
| – 🕤 MemCpy (DtoH) | | | | | | | I | | | | | - I | | | |
| 🛨 Compute | | | poisson2d_se | rial_1F1L70_18 | 3_gpu | | poisson2d_s | | poisson | 2d_serial_1F1L7 | '0_202_gpu | | | | |
| Streams | | | | | | | | | | | | | | | |
| ⊢ Stream 24 | | | poisson2d_se | rial_1F1L70_18 | 3_gpu | | poisson2d_s | | poisson | 2d_serial_1F1L7 | '0_202_gpu | | | | |
| | <u> </u> | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

OpenMP Thread 1 (GPU 1)



jacobi/4_multiple_gpu_openmp

Multiple MPI Ranks



Redundant Matrix Multiply

Each MPI rank is mapped to a GPU and performs the same steps (hence, redundant):

- Fill 2 NxN matrices with random numbers
- Perform a matrix multiply on CPU
- Perform a matrix multiply on GPU (loop_count times)
- Check for consistency between CPU and GPU results

Each MPI rank prints

- Its rank ID
- The hardware thread, GPU, and compute node it ran on
- Its total runtime and time spent computing on GPU



Multiple MPI Ranks

redundant MM



Multiple MPI Ranks

Compile the code

\$ make

Run the code

\$ bsub submit.lsf

From submit.lsf

jsrun -n1 -c42 -g6 -a2 -bpacked:7 nvprof -o mat_mul.\${LSB_JOBID}.%h.%q{OMPI_COMM_WORLD_RANK}.nvvp ./redundant_mm 2048 100 | sort

%q{OMPI COMM WORLD RANK} (Replace with MPI Rank)

\$ cat mat mul.12233

•••

==127243== Generated result file: /gpfs/wolf/stf007/scratch/t4p/nvidia_profilers/redundant_MM/mat_mul.12233.h49116.1.nvvp ==127242== Generated result file: /gpfs/wolf/stf007/scratch/t4p/nvidia_profilers/redundant_MM/mat_mul.12233.h49n16.0.nvvp

(N = 2048) Max Total Time: 3.524076 Max GPU Time: 0.308476 Rank 000, HWThread 008, GPU 0, Node h49n16 - Total Time: 3.520249 GPU Time: 0.308134 Rank 001, HWThread 054, GPU 1, Node h49n16 - Total Time: 3.524076 GPU Time: 0.308476



. . .

2 Select "Nvprof" then "Next >"

1 File->Import

| KVIDIA Visual Profiler | File View Wind | dow Help | | 00 | | | | | |
|---|---|---------------------------|---|--|--|--|--|--|--|
| | Provident Session Open Clone Session | 第N 第O 압米C | Select Import profile data generated by nvprof. | Nvprof Import | | | | | |
| | 📓 Save | жs | Select an import source: | Sing Mult | | | | | |
| | Save All | 企業S | Command-line Profiler | | | | | | |
| | 👌 Import | % I | Nvprof | | | | | | |
| | | | Import Nvprof Data | | | | | | |
| 4 | | Import Prof Select nvp | ile Data for Multiple Processes prof profile files containing timeline data for multiple | processes | | | | | |
| Click "Browse" next to "Timeline locate the .nvvp files on your lo | e data file" to ocal system, | | Profile Files Timeline Options | | | | | | |
| | files | Connection: Local | | | | | | | |
| | | The nvpro | f profile files: | | | | | | |

3 Select "Multiple Process" then "Next >"

Nvprof profile files
Import profile data for a

Multiple processes

Single process

Manage connections...

Browse...

57





Multiple MPI Ranks

| <u>Run the code</u> | From submit.lsf |
|--------------------------|---|
| \$ bsub submit_named.lsf | jsrun -n1 -c42 -g6 -a2 -bpacked:7 \ nvprof -s -o mat mul.\${LSB JOBID}.%h.%q{OMPI COMM WORLD_RANK}.nvvp \ context-name "MPI Rank %q{OMPI_COMM_WORLD_RANK}" \ process-name "MPI Rank %q{OMPI_COMM_WORLD_RANK}" ./redundant_mm 2048 100 sort |
| | Name the Process and CUDA Context |

\$ cat mat_mul.12240

•••

==144939== Generated result file: /gpfs/wolf/stf007/scratch/t4p/nvidia_profilers/redundant_MM/mat_mul.12240.h49n16.0.nvvp ==144938== Generated result file: /gpfs/wolf/stf007/scratch/t4p/nvidia profilers/redundant MM/mat_mul.12240.h49n16.1.nvvp

(N = 2048) Max Total Time: 3.634345 Max GPU Time: 0.311632

Rank 000, HWThread 024, GPU 0, Node h49n16 - Total Time: 3.634345 GPU Time: 0.311632 Rank 001, HWThread 053, GPU 1, Node h49n16 - Total Time: 3.622655 GPU Time: 0.310216

•••





visual profiler sections



Multiple MPI Ranks (annotating with NVTX)

redundant MM nvtx





And added the following NVIDIA Tools Extension library to the Makefile: -InvToolsExt

CAK RIDGE

62

Multiple MPI Ranks

Compile the code

\$ make

Run the code

\$ bsub submit.lsf

Same process as previous version of the code



| *mat_mul_nvtx.12243.h49n16.0.nvvp 🛙 | | | | | | | | | | | | | |
|---|-----|--------|---------------------|-------------|-----|-------------------------|-------|---------|------|--------|-----------------------|---------------------|-----|
| | 0 s | 0.25 s | 0.5 s | 0.75 s | 1 s | 1.25 s | 1.5 s | 1.75 s | 2 s | 2.25 s | 2.5 s | 2.75 s | 3 s |
| - Process "MPI Rank 0" (1483 | | | | | | | | | | | | | |
| Thread 310816 | | | | | | | | | | | | | |
| Runtime API | | | cudaMallo | 0C | | | | | | | cudaFree | cudaEventSynchroniz | e |
| Driver API | | | | | | | | | | | | | |
| + Markers and Ranges | | | Allocate arrays (Cl | PU & GPU) | | Initialize Arrays (CPU) | | CPU DGE | MM | CL | JBLAS Initialization | GPU DGEMM (loop_cou | |
| Profiling Overhead | | | | | | | | | | | | | |
| - Process "MPI Rank 1" (1483 | | | | | | | | | | | | | |
| Thread 310816 | | | | | | | | | | | | | |
| Runtime API | | | cudaM | alloc | | | | | | | cudaFree | cudaEventSynchro | ize |
| Driver API | | | | | | | | | | | | | |
| + Markers and Ranges | | | Allocate arrays | (CPU & GPU) | | Initialize Arrays (CPU) | | CPU D | GEMM | | CUBLAS Initialization | GPU DGEMM (loop_c | bun |
| Profiling Overhead | | | | | | | | | | | | | |
| [0] Tesla V100-SXM2-16GB | | | | | | | | | | | | | |
| Context MPI Rank 0 (CUDA) | | | | | | | | | | | | | |
| - 🍸 MemCpy (HtoD) | | | | | | | | | | | | | |
| – 🍸 MemCpy (DtoH) | | | | | | | | | | | | | |
| + Compute | | | | | | | | | | | | | |
| + Streams | | | | | | | | | | | | | |
| [1] Tesla V100-SXM2-16GB | | | | | | | | | | | | | |
| Context MPI Rank 1 (CUDA) | | | | | | | | | | | | | |
| MemCpy (HtoD) | | | | | | | | | | | | | |
| - 🍸 MemCpy (DtoH) | | | | | | | | | | | | | |
| + Compute | | | | | | | | | | | | | |
| + Streams | | | | | | | | | | | | | |
| | | | | | | | | | | | | | |

Now we have a better (and fuller) mapping to what is happening in our code.



Multiple MPI Ranks (Unified Memory)

redundant_MM_UM



| Redundant Matrix Multiply – Visual Profiler + UM + NVTX | | | | | | | | | |
|---|--|--|--|--|--|--|--|--|--|
| /*Allocate memory for arrays on CPU and GPU | /* Transfer data from CPU to GPU | | | | | | | | |
| RANGE_PUSH("Allocate CPU and UM arrays", CLR_YELLOW); | // No explictit data transfer required for arrays allocated with cudaMallocManaged | | | | | | | | |
| <pre>// Allocate memory for C_cpu on CPU double *C_cpu = (double*)malloc(N*N*sizeof(double));</pre> | | | | | | | | | |
| <pre>// Allocate memory for A, B, C for use on both CPU and GPU double *A, *B, *C; cudaErrorCheck(cudaMallocManaged(&A, N*N*sizeof(double))); cudaErrorCheck(cudaMallocManaged(&B, N*N*sizeof(double))); cudaErrorCheck(cudaMallocManaged(&C, N*N*sizeof(double)));</pre> | <pre>/* Transfer data from GPU to CPU</pre> | | | | | | | | |

RANGE_POP;

Then use the common pointers on both CPU and GPU



Multiple MPI Ranks

Compile the code

\$ make

Run the code

\$ bsub submit.lsf

Same process as previous version of the code









When data is needed on GPU (for the first GPU DGEMM), GPU page faults trigger data migration from CPU to GPU. When data is needed on CPU (to compare CPU/GPU results), CPU page faults trigger data migration from GPU to CPU.





The time for the 1st GPU DGEMM is increased due to page faults and data migration, while subsequent calls are not since data is already on the GPU



| | 3.13 s | 3.14 s | 3.15 s | 3.16 s | 3.17 s | 3.18 s | 3.19 s | 3.2 s | 3.21 s | 3.22 s |
|------------------------------------|---------------------------------|-------------------------------------|--|-------------|------------------|--------------------------|---|---|-------------------|-----------------|
| 🛨 Process "MPI Rank 1" (67517) | | | | | | | | | | |
| Process "MPI Rank 0" (67516) | | | | | | | | | | |
| Thread 310560 | | | | | | | | | | |
| Runtime API | | | | | | | | | | |
| L Driver API | | | | | | | | | | |
| Markers and Ranges | | | | | | | | | | |
| Profiling Overhead | Instrume | | | | | | | | , | |
| Unified Memory | | | | | | | | | | |
| - T CPU Page Faults | | | | | | | | | | |
| [0] Tesla V100-SXM2-16GB | | | | | | | | | | |
| Unified Memory | | | | | | | | | | |
| - 🍸 Data Migration (DtoH) | | | | | | | | | | |
| - T GPU Page Faults | | 1 | GPU Pa | e Faults | | | GPU Page Fault | ts | | |
| - 🍸 Data Migration (HtoD) | Groups of | | Data Migra | tion (HtoD) | 1 | | Data Migration (H | toD) | | |
| Context MPI Rank 0 (CUDA) | | | | | | | | | | |
| - T MemCpy (HtoD) | page launs | | | | | | | | | |
| - Compute | for a given | | | volt | a dgemm_64x64_nn | | volta volta vol | ta volta volta volta | volta volta volta | a volta volta v |
| └ \ 100.0% volta_dg | time period | / | | volt | a dgemm 64x64 nn | | volta volta vol | ta volta volta | volta volta volta | a volta volta v |
| - Streams | | | | | | | | | _ | |
| └ Stream 7 | | | | volt | a_dgemm_64x64_nn | | volta volta vol | ta volta volta volta | volta volta volta | a volta volta v |
| + [1] Tesla V100-SXM2-16GB | | | | | | | | | _ | |
| | Properties X | | | | | Properties 🕅 | | | 8 | |
| | | | | | | Troperices to | | | | |
| | GPU Page Fault groups | r this timeline. In this mode the t | timeline is solit into equal width segments an | 4 | | Data Migration (HtoD) | ad for this timeline. In this mode the | timeline is split into equal width segments | and | |
| | only aggregated data values for | or each time segment are shown | n. | - | | only aggregated data val | ues for each time segment are show | vn. | | |
| | Start | | 3.139 s (3,138,706,395 ns | | | Start | | 3.139 s (3,138,706,395 | ns) | |
| | Duration | | 36.077 ms (36,077,085 ns) | | | Duration | | 3.175 s (3,174,783,480 36.077 ms (36,077,085 | ns) | |
| | Virtual Address Range | | 0x200080000000 - 0x20 | | | Size | | 65.012 MB | , | |
| | GPU Page Faults | | 14022 | | | Throughput | | | | |
| | Duration of GPU page fault | ults | 26.821 ms | | | Min Max | | 18.124 GB/s 41.98 GB/s | | |
| | Process | | 67516 | | | Virtual Address Rang | je | 0x20008000000 - 0x2 | 0 | |
| | | | | | | Duration of HtoD dat | a migrations | 2.305 ms | | |
| | | | | | | Process | | 67516 | | |
| | The time taken to resolve GP | U page faults within the segmen | nt | | | The time taken for data | migrations from host to device with | in the segment | | |
| | 0 - 10 % [0 - 3 | 3.608 ms] | | | | 0 - 10 % | [0 - 3.608 ms] | | | |
| | 10 - 20 % [3.60 | 08 ms - 7.215 ms] | | | | 10 - 20 % | [3.608 ms - 7.215 ms] | | | |
| | 20 - 30 % [7.21 | 15 ms - 10.823 ms] | | | | 20 - 30 % | [7.215 ms - 10.823 ms] | | | |
| | 30 - 40 % [10.8 | 323 ms - 14.431 ms] | | | | 30 - 40 % | [10.823 ms - 14.431 ms] | | | |
| | 40 - 50 % [14.4 | 431 ms - 18.039 ms] | | | | 40 - 50 % | [14.431 ms - 18.039 ms] | | | |
| | 50 - 60 % [18.0 | 039 ms - 21.646 ms] | | | | 50 - 60 % | [18.039 ms - 21.646 ms] | | | |
| | 60 - 70 % [21.6 | 546 ms - 25.254 ms] | | | | 60 - 70 % | [21.646 ms - 25.254 ms] | | | |
| | | 254 ms - 28.862 ms] | | | | 70 - 80 % | [25.254 ms - 28.862 ms] | | | |
| 71 National Laborat | 90 - 100 % [28.8 | 2 469 ms] | | | | 90 - 100 % | [20.002 ms - 32.409 ms] [> 32 469 ms] | | | |
| | 00-100% [23 | | | | | 00 100 % | | | | |

Kernel Analysis


Kernel Analysis – Gathering Details Remotely

1. Gather a timeline for a **short** run.

\$ jsrun --smpiargs="none" -n1 -c1 -g1 -a1 nvprof -fo single gpu data.timeline100.nvprof ./run

2. Gather matching "analysis metrics" (Runtime will explode due to each kernel being replayed multiple times.

\$ jsrun --smpiargs="none" -n1 -c1 -g1 -a1 nvprof --analysis-metrics -fo single gpu data.metrics100.nvprof ./run

If you cannot shorten your run any longer, it's possible to use the --kernels option to only replay some kernels, but guided analysis may not work as well.



Kernel Details – Import into Visual Profiler

1 File->Import

| Ś | NVIDIA Visual Profiler | File | View | Window | v Hel | lp |
|-------|-------------------------------|----------|---------|--------|-------|----|
| | | * | New Se | ssion | ЖN | |
| 📫 🖪 K | | Op | en | | жo | |
| | | Clo | ne Sess | ion 🕆 | жC | |
| | | | Save | | жs | |
| | | | Save As | | | |
| | | Q | Save Al | Û | жs | |
| | | è | Import | | ۴I | |
| | | _ | | | | |

Select

Import profile data generated by nvprof.

Select an import source:

Command-line Profiler

Nvprof

Select "Single Process"

Select "Nvprof" then "Next >"

| 💺 Import Nvprof D | ata | | | | × |
|--|---|---------|--------|-------------|-----------|
| Import Profile D | ata for Single Process | | | | |
| Select one nvprof p event and metric v | ofile file containing timeline data and zero or more addition nvprof profile files containing lues. | | | | |
| Profile Files Time | ne Options | | | | |
| Connection: | Local ~ | Manag | ge coi | nnectio | ns |
| Timeline data file: Event/Metric data f | C:\Users\jlarkin\OneDrive - NVIDIA Corporation\2019\Profilers Tutorial\single_gpu_data.tin | neline1 | 00.nv | Brow | se |
| C:\Users\jlarkin\O | eDrive - NVIDIA Corporation\2019\Profilers Tutorial\single_gpu_data.metrics100.nvprof | | | Brow Rem | se ove |
| | | | | | |

4

Click "Browse" next to "Timeline data file" to locate the .nvprof file on your local system, then do the same for "Event/Metric data files," then click "Finish"

CAK RIDGE National Laboratory

Visual Profiler Import – Common Warning





| File View Window Run Help Image: Section of the | 0.4 s | 0.5 s | 0.6 s | 0.7 s | 0.8 s | 0.9 s | | 1 s |
|---|---|------------|-------|-------|--|-------|-----|------------|
| Image: | 0.4 s | 0,5 s | 0.6 s | 0.7 s | s 0.8 s 0.9 s cuDevicePrimaryCtxR □ | | 1 s | |
| ▼single_gpu_data.timeline100.nvprof ∅ 0.1 s 0.2 s 0.3 s ■ Process "run" (176968) ■ <td>0.4 s etain</td> <td>0.5 s</td> <td>0.6 s</td> <td>0.7 s</td> <td>0.8 s</td> <td>0.9 s</td> <td></td> <td>1 s</td> | 0.4 s etain | 0.5 s | 0.6 s | 0.7 s | 0.8 s | 0.9 s | | 1 s |
| 0 s 0.1 s 0.2 s 0.3 s Process "run" (176968) - - Thread 294448 - - OpenACC - - Driver API cuDevicePrimaryCtxRet Profiling Overhead - [0] Tesla V100-SXM2-16GB - [] Context 1 (CUDA) - [] Y MemCpy (HtoD) - | 0.4 s | 0.5 s | 0.6 s | 0.7 s | 0.8 s | 0.9 s | | 1 <u>s</u> |
| Process "run" (176968) Thread 294448 OpenACC Driver API cuDevicePrimaryCtxRe Profiling Overhead [0] Tesla V100-SXM2-16GB Context 1 (CUDA) \ ™ MemCpy (HtoD) | 2vorfiler - ovr Byn Heip - attrinetine100.0xppcf 32 - 0 s 0.1 s 0.2 s 0.3 s 0.4 s 0.5 s 0.6 s 0.7 s 0.8 s 0.9 s 16 - <td></td> <td>^</td> | | ^ | | | | | |
| Thread 294448 CopenACC Driver API Profiling Overhead [0] Tesla V100-SXM2-16GB Context 1 (CUDA) U Y MemCpy (HtoD) | WOULA Visual Profile | | | | | | | |
| └ OpenACC └ Driver API CuDevicePrimaryCtxRe └ Profiling Overhead ○ [0] Tesla V100-SXM2-16GB ○ Context 1 (CUDA) └ \ MemCpy (HtoD) | MDDL Visual Profiler - We Window Bun Heip - Single, gou, data timeline100.mprof SL - None 0 Process Touri (17696) 0.1 s 0.2 s 0.3 s 0.4 s 0.5 s 0.6 s 0.7 s 0.9 s | | | | | | | |
| L Driver API cuDevicePrimaryCtxR L Profiling Overhead Image: Context 1 (CUDA) Image: Context 1 (CUDA) Image: Context 1 (CUDA) Image: Context 1 (CUDA) Image: Context 1 (CUDA) | etain | | | | | | | |
| Profiling Overhead [0] Tesla V100-SXM2-16GB Context 1 (CUDA) | | | | | cuDevicePrimaryCtxF | 0.9 s | | |
| [0] Tesla V100-SXM2-16GB Context 1 (CUDA) \[\frac{1}{2} MemCpy (HtoD) \] | | | | | | | | |
| Context 1 (CUDA) \[\screwtytem="context-align: center;" MemCpy (HtoD) | | | | | | | | |
| L 🍸 MemCpy (HtoD) | | | | | | | | |
| | | | | | | | | |
| L 🍸 MemCpy (DtoH) | | | | | | | | _ |
| 🗔 Analysis 🕱 💼 GPU Details (Summary) 🎫 CPU Details 🧊 OpenACC Details 📷 OpenMP Details 🗐 | Console | 🖩 Settings | | | Properties 🛛 | | | |
| 🔚 🗄 🔂 🔤 Export PDF Report Results | | | | | | | | |
| 1. CUDA Application Analysis | - - - - 0.1 s 0.2 s 0.4 s 0.5 s 0.6 s 0.7 s 0.4 s 0.9 s cuDexicePrimaryCbRetain - - - - - - vD Details OpenACC Details © OpenAMP Details © Console Settings - - - Results - - - - - - - - v - < | e properti | 25 | | | | | |
| Start with a high- level overview of application performance. | | | | | | | | |

76

| 💺 NVIDIA Visual Profiler | | | | | | | | | _ | | × |
|---|-------------------|--|-----------------------------|----------------------|--------------------------|--------------------------|-------|------------------------------|-------------------|---------------|-----|
| <u>File V</u> iew <u>W</u> indow <u>R</u> un <u>H</u> elp | | | | | | | | | | | |
| | 🗓 🖏 🔍 🗸 🛛 🤅 | Ð, Q, Ð, I 🗉 I F 🖡 | 、 🔣 🚉 🖳 🛛 | Å • | | | | | | | |
| single_gpu_data.timeline100.nvprof | x | | | | | | | | | r | - 8 |
| 0 s | 0.1 | s 0.2 s | 0.3 s | 0.4 s | 0.5 s | 0.6 s | 0.7 s | 0.8 s | 0.9 s | | 1 s |
| Process "run" (176968) | | | | | | | | | | | ^ |
| Thread 294448 | | | | | | | | | | | |
| └ OpenACC | | | | | | | | | | | |
| L Driver API | | | cuDevicePrimary | /CtxRetain | | | | cuDevicePrimaryCtxR | | | |
| Profiling Overhead | | | | | | | | | | | |
| [0] Tesla V100-SXM2-16GB | | | | | | | | | | | |
| Context 1 (CUDA) | | | | | | | | | | | |
| 🗆 🍸 MemCpy (HtoD) | | | | | | | | | | | |
| L 🍸 MemCpy (DtoH) | | | | | | | | | | | |
| | | | 1 | | | | | | | ee propertie | |
| Most applications | ary) 🔠 CPU Det | ails 📺 OpenACC Details esults | 🗑 OpenMP Deta | ils 🖳 Console 🗌 | 🖬 Settings | \ | | Properties 🛛 | | | |
| will see these. | | A Low Memcpy/Kernel | Overlap [0 ns / 8.9 | 3188 ms = 0% 1 | | | ^ | Select or highlight a single | e interval to see | propertie | es |
| 1. CODA Application Analysis | | The percentage of time w | hen memcpy is beir | ng performed in pa | rallel with kernel is lo | ow. | | | | | |
| 2. Check Overall GPU Usage | | A Law Kara I Carawa | | | | | | | | | |
| The analysis results on the right indicate | potential | Low Kernel Concurre The perceptage of time will | ncy [0 ns / 97.2022 | ms = 0% j | aarallel is low | | | | | | |
| of the GPU's available compute and data | movement | The percentage of time w | nen two kernels are | being executed in | parallel is low. | | _ | | | | |
| capabilities. You should examine the infor provided with each result to determine if | mation vou can | Low Memcpy Throug | hput [6.775 MB/s a | ivg, for memcpys a | ccounting for 3.5% | of all memcpy time] | | Also co | ommon | . mc | VĽ |
| make changes to your application to incre | ease GPU | The memory copies are no | ot fully using the av | ailable host to devi | ce bandwidth. | | | indicat | | e of | . / |
| | | 💧 Low Memcpy Overla | p [0 ns / 3.0515 ms | = 0%] | | | | unninn | ed me | mon | 18 |
| 🖳 Examine Individual Kerne | ls · | The percentage of time w | hen two memory co | pies are being perf | ormed in parallel is | low. | | | | n Or) Nata | γœ |
| You can also examine the performance of individual ke | meis to expose | 💧 Low Compute Utiliza | tion [97.2522 ms / 8 | 877.80852 ms = 11.1 | 1%] | | | Synchro | Shoos (| Jaia | |
| adational optimization opportantices. | · · | The multiprocessors of on | e or more GPUs are | mostly idle. | | | | copies | | | |
| May indicate | | i Compute Utilization | | | | | _ | | | | |
| insufficient • | | The device timeline shows | an estimate of the | amount of the tota | I compute capacity | being used by the kernel | ls ex | | | | |
| amount of work | | | | | 1 | , ., | | | | | |
| GITTOUTI OF WORK. | | 1 NVLink Analysis | | - In the INDUCT | | CDU CDU- A-L | | | | | |
| | < | i ne tollowing NVLink top | biogy diagram show | vs logical NVLink co | onnections between | GPUs and CPUs. A logic | | | | | |
| | | | | | | | | | | | |



| Visual Profiler | | | | | | | | | | - | |
|--|-----------------------------|------------------------|--|------------------------------|-----------------------|-------------------|---------------------|--------------|---------------------|----------------------|-----------|
| File view window Kun Help | | - (() - (| | K 5 P . | • | | | | | | |
| single gpu data.timeline100.nvpr | of X | 1440 | | | | | | | | | |
| • ····· |) s | 0.1 s | 0.2 s | 0.3 s | 0.4 s | 0.5 s | 0.6 s | 0.7 s | 0.8 s | 0.9 s | |
| Process "run" (176968) | | I | | I | I | | I | I | I | | |
| Thread 294448 | | | | | | | | | | | |
| └ OpenACC | | | | | | | | | | | |
| L Driver API | | | | cuDevicePrimaryCt | xRetain | | | | cuDevicePrima | aryCtxR | |
| Profiling Overhead | | | | | | | | | | | |
| [0] Tesla V100-SXM2-16GB | | | | | | | | | | | |
| Context 1 (CUDA) | | | | | | | | | | | |
| 🗆 🍸 MemCpy (HtoD) | | | | | | | | | | | |
| 🗆 🍸 MemCpy (DtoH) | | | | | | | | | | | |
| | | | | | 1 | 1 | | | • | 1 | |
| 🔚 Analysis 🛛 🧱 GPU Details (Sum | nmary) 🔠 CPL | J Details 🛛 📺 🕻 | OpenACC Details | 🏢 OpenMP Details | 📮 Console 🗔 Se | ttings | | ∖ □ □ | Properties 🛛 | | |
| 🔚 🗄 🗘 🔛 🛄 Export PDF | Report | Results | | | | | | | | | |
| 1. CUDA Application Analysis | | 💧 Low N | Memcpy/Kernel 0 | verlap [0 ns / 8.931 | 88 ms = 0%] | | | | Select or highlight | a single interval to | see prope |
| 2. Check Overall GPU Usage | | The perce | ntage of time whe | n memcpy is being | performed in paralle | with kernel is l | ow. | | | | |
| The analysis results on the right indicat | te potential | 🗧 💧 Low 🛛 | Cernel Concurren | cy [0 ns / 97.2522 m | s = 0%] | | | | | | |
| problems in how your application is ta | king advantage | The perce | ntage of time whe | n two kernels are be | ing executed in para | lel is low. | | | | | |
| capabilities. You should examine the in | formation | 💧 Low N | Memcpy Through | out [6.775 MB/s avg | , for memcpys acco | unting for 3.5% | of all memcpy time | •] | | | |
| provided with each result to determine make changes to your application to in | e if you can Icrease GPU | The mem | ory copies are not | fully using the availa | ble host to device b | andwidth. | | | | | |
| utilization. | | A low M | Memcov Overlap | 0 ns / 3.0515 ms = (|)% 1 | | | | | | |
| 🖳 Examine Individual Ker | nels | The perce | ntage of time whe | n two memory copi | es are being perform | ed in parallel is | low. | | | | |
| You can also examine the performance of individua | Hernels to expose | Alow(| - Compute I Itilizati | n [07 2522 ms / 877 | 200952 ms = 11.1% 1 | | | | | | |
| additional optimization opportunities. | | The multi | processors of one | or more GPUs are m | ostly idle. | | | | | | |
| | | i Comp | oute Utilization | | | | | | | | |
| | | The devic | e timeline shows a | n estimate of the an | nount of the total co | mpute capacity | being used by the l | kernels ex | | | |
| | | | | | | | | | | | |
| | | i NVI in | nk Analysis | | | | | | | | |
| | | i NVLin | n k Analysis ving NVLink topol | ogy diagram shows l | ogical NVLink conne | ections between | GPUs and CPUs. A | logical N' 🗸 | | | |

Next zoom in on individual kernel optimizations.

CAK RIDGE National Laboratory

| | 💺 NVIDIA Visual Profiler | | | | | | | | | | _ | ΟX |
|---|--|--|--|--|--|--------------------------------|---|-------------------|----------------------|-----------------------|------------------------|------------|
| | <u>File View Window Run H</u> elp | | | | | | | | | | | |
| | | 🕕 🛶 🔍 🗸 | \odot \odot \odot | 🖃 F 🔭 | K 📮 🔜 📩 🕶 | | | | | | | |
| | 💺 *single_gpu_data.timeline100.nvp | rof 🖾 | | | | | | | | | | - 8 |
| | | 0 s 0.1 | 1 s | 0.2 s | 0.3 s | 0.4 s | 0.5 s | 0.6 s | 0.7 s | 0.8 s | 0.9 s | 1 s |
| | Process "run" (176968) | | | | | | | | | | | ^ |
| | Thread 294448 | | | | | | | | | | | |
| | └ OpenACC | | | | | | | | | | | |
| | L Driver API | | | | cuDevicePrimaryCtxRet | ain | | | | cuDevicePrimary | CtxR | |
| | Profiling Overhead | | | | | | | | | | | |
| | [0] Tesla V100-SXM2-16GB | | | | | | | | | | | |
| | Context 1 (CUDA) | | | | | | | | | | | |
| | 🗕 🍸 MemCpy (HtoD) | | | | | | | | <u> </u> | | | |
| | └ 🍸 MemCpy (DtoH) | | | | | | | | | | | ~ |
| | 🗔 Analysis 🔀 📴 GPU Details (Sur | nmary) 🔠 CPU De | etails 🕞 Oper | nACC Details [| 🖉 OpenMP Details 📃 | Console | Settings | | \ | Properties 🛛 | | |
| | E 🗘 🚺 Export PDF | Report | Results | | | | | | | | | |
| | 1. CUDA Application Analysis | | i Kernel O | ptimization Prio | orities | | | | | Select or highlight a | single interval to see | properties |
| This table ranks | 2. Performance-Critical Kernels | | Optimization compared to | y kernels are orde of higher ranked lower ranked ker | ered by optimization in d kernels (those that ap rnels. | portance ba pear first in t | sed on execution tim he list) is more likely | to improve perfor | ccupancy. mance | | | |
| the kernels by bang for buck, click the top one. Click here to deep dive on the selected kernel. | The results on the right show your application will be running the distingthered by potential for performing over the sense of the sens | plication s rmance s with the entry from the s to discover | Rank Desc 100 [100 66 [100 37 [100 5 [100 2 [100 | cription D kernel instance D kernel instance D kernel instance D kernel instance | s] main_123_gpu s] main_134_gpu s] main_127_gpu_red s] main_148_gpu s] main_142_gpu | | | | | | | |
| OAK RIDGE | or one new results at right to highlight the individua the analysis applies. | a kernels for Which | | | | | | | | | | |

Visual Profiler – Guided Analysis – Bandwidth Bound

| The state of the s | | | | | | | | | | — | |
|--|---|---|--|---|---|--|---|--|--|-----------------------------------|--------------------------------|
| File View Window Run He | WIDIA Visual Profiler Yiew Window Run Help Image: gpu_data.timeline100.nvprof 0 s 0.1 s 0.2 s 0.3 s occess 'run' (176968) Thread 294448 OpenACC Driver API Profiling Overhead Tesla V100-SXM2-16GB Context 1 (CUDA) Image: GPU Details (Summary) < | | | | | | | | | | |
| | //DIA Visual Profiler (iew Window Run Help (iew Window Run Help (igle_gpu_data.timeline100.nyprof 22 (igle_gpu_data.tigligle_gpu_data.timeline100.nyprof 22 | % - | | | | | | | | | |
| *single_gpu_data.timeline100. | nvprof 🖾 | | | | - | | | | | | |
| | VIOLA Visual Profiler Yiew Window Bun Help Image: gpu_data.timeline100.nyprof | 0.5 s | 0.6 s | 0.7 s | 0.8 s | 0.9 s | | | | | |
| Process "run" (176968) | A Visual Profiler | · | · · · · · | • | · · · · · · | • | | | | | |
| Thread 294448 | IDIA Visual Profiler jew Window Run Help iew Window Run Help igle_gpu_data.timeline100.nvprof ⊠ istriction istriction <t< td=""><td></td><td></td><td></td><td></td><td></td><td></td><td></td></t<> | | | | | | | | | | |
| └ OpenACC | | | | | | | | | | | |
| L Driver API | IA Visual Profiler w Window Bun Help | | | | | | | | | | |
| Profiling Overhead | IA Visual Profiler w Window Run Help | | | | | | | | | | |
| [0] Tesla V100-SXM2-16GB | DIA Visual Profiler ew Window Run Help | | | | | | | | | | |
| Context 1 (CUDA) | | | | | | | | | | | |
| 🗕 🍸 MemCpy (HtoD) | | | | | | | | | | | |
| 🗕 🍸 MemCpy (DtoH) | | | | | | | | | | | |
| | (C) | | 0 ACC 0 1 1 | | | . e | | | | | ~] |
| | (Summary) | CPU Details La | Results | L# OpeniviP Detai | | settings | | | N | | ···· ~ [|
| | | лт. | i Kernel Perfo | rmance is Bound I | Ry Memory Bandy | vidth | | | | · · · | |
| 1. CUDA Application Analysis | | | 1 Kemerreno | initialice is bound i | by memory bandy | | | | | | |
| | WIDIA Visual Profiler Yiew Window Bun Help Image: gpu_data.timeline100.nvprof IX Image: gpu_data.timelintimage: g | the kernel's compu | ite utilization is sign | ificantly lower that | n its memory utilizatio | on. These | Select o | r hiahlial | | | |
| 2. Performance-Critical Kernels | MDIA Visual Profiler Yiew Window Run Help Image: gpu_data.timeline100.nvprof S3 0 s 0.1 s 0.2 s 0.3 s occess "run" (176968) Thread 294448 OpenACC Driver API Profiling Overhead I testa V100-SXM2-1668 Context 1 (CUDA) YmmenCpy (HtoD) YmmenCpy (DtoH) Thread 294448 I testa V100-SXM2-1668 Context 1 (CUDA) YmmenCpy (DtoH) YmmenCpy (DtoH) Image: gpu is most likely limited by memory Intre 1 pin analyzing an individual kernel is to mismic if the performance of the kernel is to mismic if the performance of the kernel is to mismic if the performance of the kernel is to mismic if the performance of the kernel is to mismic if the performance of the kernel is to mismic if the performance of the kernel is to mismic if the performance of the kernel is to mismic if the performance of the kernel is to mismic if the performance of the kernel is to mismic if the performance of the kernel is to mismic if the performance of the kernel is to mismic if the performance of the kernel is to mismic if the performance of the kernel is to mismic if the performance of the kernel is to mismic if the performance of the kernel is to mismic if the performance of the kernel is bounded by tubulation, memory bandwidth analysis Imperform Compute Analysis Imperform Anenory Bandwidth analysis t | the kernel's compu formance of the k | ite utilization is sign ernel is most likely b | ificantly lower that being limited by the | n its memory utilizatio e memory system. Foi | on. These r this kernel the | Select or single in | r highligl Iterval to | | | |
| 2. Performance-Critical Kernels 3. Compute, Bandwidth, or Late | WIDIA Visual Profiler Yiew Window Run Help Image: gpu_data.timeline100.nvprof 0 s 0.1 s 0 s 0.1 s 0 s 0.1 s 0 s 0.1 s 0 penACC Driver API Profiling Overhead 1) Tesla V100-SXM2-16GB 2 Context 1 (CUDA) Y MemCpy (HtoD) Y MemCpy (DtoH) Y MemCpy (D | the kernel's compu formance of the k n is the bandwidth | ite utilization is sign ernel is most likely b of the Device memo | ificantly lower that being limited by the ory. | n its memory utilizatio e memory system. Foi | on. These r this kernel the | Select o single in properti | r highligł iterval to es | | | |
| 2. Performance-Critical Kernels 3. Compute, Bandwidth, or Late The first step in analyzing an indiv | ncy Bound idual kernel is t | to ^ | For device "Tesla" utilization levels in limiting factor in | V100-SXM2-16GB" ndicate that the per the memory systen | the kernel's compu formance of the k n is the bandwidth | ite utilization is sign ernel is most likely b of the Device memo | ificantly lower tha being limited by the ory. | n its memory utilizatic e memory system. For | on. These r this kernel the | Select o single in properti | r highligi iterval to es |
| 2. Performance-Critical Kernels 3. Compute, Bandwidth, or Late The first step in analyzing an indiv determine if the performance of th computation, memory bandwidth. | Viola visual Profiler View Window Run Help Image: gpu_data.timeline100.nvprof Image: gpu_data. | the kernel's compu rformance of the k n is the bandwidth | ite utilization is sign ernel is most likely b of the Device memo | ificantly lower that being limited by the ory. | n its memory utilizatio e memory system. Foi | on. These r this kernel the | Select o single in properti | r highligl iterval to es | | | |
| 2. Performance-Critical Kernels 3. Compute, Bandwidth, or Later The first step in analyzing an individetermine if the performance of the computation, memory bandwidth, latency. The results at right indica kernel imain 123 gnut is most like | e_gpu_data.timeline100.nvprof \(\lambda\) e_gpu_data.timeline100.nvprof \(\lambda\) sss "run" (176968) read 294448 OpenACC Driver API ofiling Overhead dila V100-SXM2-16GB rntext 1 (CUDA) Y MemCpy (HtoD) Y MemCpy (HtoD) Y MemCpy (DtoH) Sis \(\lambda\) imance-Critical Kernels ute, Bandwidth, or Latency Bound step in analyzing an individual kernel is to eif the performance of the kernel is bounded by tion, memory bandwidth, or instruction/memory The results at right indicate that the performance of nain_123_gpu" is most likely limited by memory th. | For device "Tesla utilization levels in limiting factor in 100% 90% | V100-SXM2-16GB" ndicate that the per the memory systen | the kernel's compu rformance of the k n is the bandwidth | rte utilization is sign ernel is most likely b of the Device memo | ificantly lower that being limited by the ory. | n its memory utilizatio e memory system. Foi | on. These this kernel the | Select o single in properti | r highligl iterval to es | |
| 2. Performance-Critical Kernels 3. Compute, Bandwidth, or Late The first step in analyzing an indiv determine if the performance of th computation, memory bandwidth, latency. The results at right indica kernel "main_123_gpu" is most like bandwidth. | ncy Bound idual kernel is t e kernel is bou or instruction/i te that the perf ely limited by me | to Inded by Imemory formance of emory | For device "Tesla" utilization levels in limiting factor in 100% 90% 80% | V100-SXM2-16GB" ndicate that the per the memory systen | the kernel's compu formance of the k n is the bandwidth | rte utilization is sign ernel is most likely b of the Device memo | ificantly lower that being limited by the ory. | n its memory utilizatio e memory system. Foi | on. These this kernel the | Select o single in properti | r highligi iterval to es |
| 2. Performance-Critical Kernels 3. Compute, Bandwidth, or Later The first step in analyzing an individermine if the performance of the computation, memory bandwidth, latency. The results at right indicates kernel "main_123_gpu" is most liker bandwidth. | ncy Bound idual kernel is t be kernel is bou or instruction/ te that the perf ly limited by me | to Aunded by memory formance of emory | For device "Tesla utilization levels in limiting factor in 90% 80% 70% | V100-SXM2-16GB" ndicate that the per the memory systen | the kernel's compu rformance of the k n is the bandwidth | rte utilization is sign ernel is most likely b of the Device memo | ificantly lower that being limited by the ory. | n its memory utilizatio e memory system. Foi | on. These r this kernel the | Select o single in properti | r highligl iterval to es |
| 2. Performance-Critical Kernels 3. Compute, Bandwidth, or Later The first step in analyzing an individe determine if the performance of the computation, memory bandwidth, latency. The results at right indica kernel "main_123_gpu" is most liker bandwidth. | ncy Bound idual kernel is t ne kernel is bou or instruction/ te that the perf ly limited by me indwidth Analy | to unded by formance of emory rsis | For device "Tesla" utilization levels in limiting factor in 90% 80% 70% 50% 60% | V100-SXM2-16GB" ndicate that the per the memory systen | the kernel's compu rformance of the k n is the bandwidth | rte utilization is sign ernel is most likely b of the Device memo | ificantly lower that being limited by the ory. | n its memory utilizatio e memory system. For | on. These this kernel the goperations | Select o single in properti | r highligl iterval to es |
| 2. Performance-Critical Kernels 3. Compute, Bandwidth, or Late The first step in analyzing an indiv determine if the performance of th computation, memory bandwidth, latency. The results at right indica kernel "main_123_gpu" is most like bandwidth. | ncy Bound idual kernel is t e kernel is bou or instruction/i te that the perf ely limited by me indwidth Analy: rthis kernel is memo th analysis to determ | to inded by formance of emory /sis ory bandwidth mine how it is | For device "Tesla" utilization levels in limiting factor in 90% 80% 70% 60% 50% | V100-SXM2-16GB" ndicate that the per the memory systen | the kernel's compu formance of the k n is the bandwidth | rte utilization is sign ernel is most likely b of the Device memo | ificantly lower that being limited by the ory. | n its memory utilizatio e memory system. For Memor Control | on. These r this kernel the g operations -flow operations tic operations | Select o single in properti | r highligl iterval to es |
| 2. Performance-Critical Kernels 3. Compute, Bandwidth, or Later The first step in analyzing an individermine if the performance of the computation, memory bandwidth, latency. The results at right indica kernel "main_123_gpu" is most like bandwidth. Perform Memory Band The most likely bottleneck to performance to so you should first perform memory bandwid limiting performance. | DIA Visual Profiler ew Window Run Help | For device "Tesla" utilization levels in limiting factor in 90% 80% 70% 60% 50% 40% | V100-SXM2-16GB" ndicate that the per the memory systen | the kernel's compu rformance of the k n is the bandwidth | rte utilization is sign ernel is most likely b of the Device memo | ificantly lower that being limited by the ory. | n its memory utilizatio e memory system. For Memor Control Arithme Memor | n. These r this kernel the g operations -flow operations etic operations g (Device) | Select o single in properti | r highligl terval to es | |
| 2. Performance-Critical Kernels 3. Compute, Bandwidth, or Late The first step in analyzing an indiv determine if the performance of th computation, memory bandwidth, latency. The results at right indica kernel "main_123_gpu" is most like bandwidth. | /IDIA Visual Profiler /jew Window Run Help igle_gpu_data.timeline100.nvprof % 0's 0.1 s 0.2 s 0.3 s cess 'run' (176968) Thread 294448 OpenACC Driver API Profiling Overhead Tesla V100-SXM2-16GB Context 1 (CUDA) 'w MemCpy (HtoD) 'w MemCpy (DtoH) 'w Memcpy is most likely limited by memory with. 'f he performance of the kernel is bounded by trainon memory bandwidth, or instruction/memory y. The results at right indicate that the performance of memory bandwidth, or instruction/memory y. The results at right indicate that the performance of memory bandwidth analysis to determine how it is perform memory bandwidth analysis to determine how it is perform memory bandwidth analysis to determine how it is perform memory latency are likely not the perimary and control to perimary and conthis bend but to perimary and control be per | the kernel's compu formance of the k n is the bandwidth | rte utilization is sign ernel is most likely b of the Device memo | ificantly lower that being limited by the ory. | n its memory utilizatio e memory system. For Memor Control Arithme Memor | on. These r this kernel the g operations -flow operations tic operations g (Device) | Select o single in properti | r highligl terval to es | | | |
| 2. Performance-Critical Kernels 3. Compute, Bandwidth, or Late The first step in analyzing an indiv determine if the performance of th computation, memory bandwidth, latency. The results at right indica kernel "main_123_gpu" is most like bandwidth. Perform Memory Ba The most likely bottleneck to performance to so you should first perform memory bandwid limiting performance. Perform Comp | VIDIA Visual Profiler View Window Run Help | the kernel's compu formance of the k n is the bandwidth | rte utilization is sign ernel is most likely b of the Device memo | ificantly lower that being limited by the | n its memory utilizatio e memory system. For Memor Control Arithme Memor | on. These r this kernel the y operations -flow operations tic operations y (Device) | Select o single in properti | r highligl iterval to es | | | |
| 2. Performance-Critical Kernels 3. Compute, Bandwidth, or Late The first step in analyzing an indiv determine if the performance of th computation, memory bandwidth, latency. The results at right indica kernel "main_123_gpu" is most like bandwidth. Perform Memory Ba The most likely bottleneck to performance to so you should first perform memory bandwid imiting performance. Perform Comp Compute and instruction and memory latence | WDIA Visual Profiler Yiew Window Run Help Image: gpu_data.timeline100.nvprof Ingle_gpu_data.timeline100.nvprof Image: gpu_data.timeline100.nvprof | V100-SXM2-16GB" ndicate that the per the memory system | the kernel's compu formance of the k n is the bandwidth | rte utilization is sign ernel is most likely b of the Device memo | ificantly lower that being limited by the ory. | n its memory utilizatio e memory system. For Memor Control Arithme Memor | on. These r this kernel the g operations -flow operations etic operations g (Device) | Select o single in properti | r highligl terval to es | | |

This box will estimate the performance limiter of your kernel

Click here to dive deeper on that performance limiter

CAK RIDGE

Visual Profiler – Guided Analysis – Bandwidth Bound

| rile view window Kun He | ew <u>Window Run Help</u> | | | | | | | | | | |
|--|---|-------------------|--|--|--|---|--|--|--|------------|--------------|
| | Image: Second | | | | Δ.T | | | | | | |
| *single_gpu_data.timeline100.n | Image: Second system Image: Second system Image: Second system Ima | | | | | | | | | | |
| | Intervention Intervention Interventinter Intervention <tr< td=""><td>0.2 s</td><td>0.3 s</td><td>0.4 s</td><td>0.5 s</td><td>0.6 s</td><td>0.7 s</td><td>0.8 s</td><td>0.9 s</td><td></td></tr<> | | 0.2 s | 0.3 s | 0.4 s | 0.5 s | 0.6 s | 0.7 s | 0.8 s | 0.9 s | |
| Process "run" (176968) | Igle_gpu_data.timeline100.nvprof ⊠ 0 s 0 s 0 s 0 s 0 s 0 s 0 s 0 s | | | | | | | | | | |
| Thread 294448 | | | | | | | | | | | |
| └ OpenACC | | | | | | | | | | | |
| 🔚 Analysis 🛛 🔤 GPU Details (S | cess "run" (176968) Thread 294448 OpenACC alysis 🕸 📷 GPU Details (Summary) ① ① ①A Application Analysis formance-Critical Kernels npute, Bandwidth, or Latency Bound | | # OpenACC Details | 📻 OpenMP Deta | ils 📃 Console 🗔 | Settings | | | <u>\.</u> | D Prop |) X |
| E E 🔂 🛄 Exp | port PDF Repor | rt | Results | | | | | | | | |
| 1 CUDA Application Analysis | | | 💧 Global Mem | ory Alignment and | d Access Pattern | | | | | ^ | |
| | | | Memory bandwi | dth is used most ef | ficiently when each | global memory loa | ad and store has pro | per alignment and | access pattern. The ar | Select or | high |
| 2. Performance-Critical Kernels | | | per assembly ins | truction. | | | | | | single int | terval or |
| 3. Compute, Bandwidth, or Later | ncy Bound | | Optimization: Sel | lect each entry belo or each load or stor | w to open the sourc e improve the alian | e code to a global lo ment and access no | oad or store within t attern of the memor | he kernel with an ir v access. | nefficient alignment or | propertie | |
| 4. Memory Bandwidth | | | access partern r | | e unprore ine dugin | ineni ana access pa | internet, the memory | decessi | | | |
| Memory bandwidth limits the perfo | ormance of a ke | ernel when | ✓ Line / File p | oisson2d.c - \gpfs\ | wolf\gen110\scrate | h\j2k\nvidia profile | ers\jacobi\3 single | qpu data | | | |
| one or more memories in the GPU o | annot provide | data at the | 126 G | lobal Load L2 Trans | sactions/Access = 9 | , Ideal Transactions | /Access = 8 [47121 | 94 L2 transactions f | or 524032 total executi | | |
| the kernel is limited by the bandwi | dth available to | the device | 126 G | lobal Load L2 Trans | sactions/Access = 9 | , Ideal Transactions | /Access = 8 [47121 | 94 L2 transactions f | or 524032 total executi | | |
| memory. | | | 126 G | lobal Load L2 Trans | sactions/Access = 9 | , Ideal Transactions | /Access = 8 [47121 | 94 L2 transactions f | or 524032 total executi | | |
| JL Rerun A | nalysis | | 126 G | lobal Store L2 Trans | sactions/Access = 9 | , Ideal Transactions | /Access = 8 [47121 | 94 L2 transactions f | or 524032 total executi | | |
| If you modify the kernel you need to rerun you | rapplication to upda | ato thic a makeic | 120 G | lobal Load L2 Trans | sactions/Access = 9 sactions/Access = 9 | Ideal Transactions | /Access = 8 [4/121 /Δccess = 8 [47121 | 94 L2 transactions f 94 L2 transactions f | or 524032 total executi or 524032 total executi | | |
| in you moonly the kenter you need to renarryou | | ite ina analysia. | | | actions, Access = 5 | , lacar mansaccions, | 140000 - 014100 | | or service total executi | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | GPU Utilizat | tion is Limited By I | Memory Bandwidt | n al haadh is haaraal ƙasa | | | in The table also also | | |
| | | | utilization of eac is potentially lim | ble shows the mem h memory type rela ited by the bandwid | ative to the maximu ative to the maximu dth available from o | d by this kernel for i m throughput supp one or more of the r | the various types of ported by the memo memories on the de | r memory on the de ory. The results sho evice. | w that the kernel's per | | |
| | | | Optimization: Try | the following opti | mizations for the me | mory with high bar | ndwidth utilization. | | | | |
| | | | Shared 12 Car | d Memory - If possii che - Alian and blo | ble use 64-bit acces ck kernel data to m | ses to shared memo | ry and 8-byte bank iiciency | mode to achieved i | 2x throughput. | | |
| | | | Unifie | d Cache - Reallocat | te texture data to sh | ared or global men | nory. Resolve alignn | nent and access pat | tern issues for global | | |
| | | | loads and stores. | | -1 | | | -4 | - | | |
| | | | Device | e memory - Kesolve | augnment and acc | ess pattern issues fo | or global loads and | stores. | | | |

This is the final set of suggestions for this kernel.

CAK RIDGE National Laboratory

Visual Profiler – Guided Analysis – Latency Bound



Low Compute & Memory utilization points to being latency bound.

Now a latency analysis is suggested

Visual Profiler – Guided Analysis – Latency Bound

| 🕵 *single gnu data timeline100 r | r Sun Help intelloOn.ppt 02 intelloOn.ppt 02 | | | | | | | | | | |
|--|--|---|---|---|--|--|--|--|---|----------------|------|
| Single_gpa_adatationerroom | 0 s | 0.1 s | 0.2 s | 0.3 s | 0.4 s | 0.5 s | 0.6 s | 0.7 s | 0.8 s | 0.9 s | |
| Process "run" (176968) | | | | · · · · | I | · · · | I | | | | |
| Thread 294448 | | | | | | | | | | | |
| └ OpenACC | | | | | | | | | | | |
| L Driver API | | | | | | | | | | | |
| Profiling Overhead | | | | | | | | | | | |
| [0] Tesla V100-SXM2-16GB | | | | | | | | | | | |
| Context 1 (CUDA) | | | | | | | | | | | |
| - Y MemCpy (HtoD) | | | | | | | | | | | |
| 1. CUDA Application Analysis | port PDF Repo | rt | Besults Grid Size Too | o Small To Hide Co | mpute And Memo | ry Latency | a latency Typically | the kernel arid rize m | wrt he large enough | Select or bigh | ыі |
| 2. Performance-Critical Kernels | | | to fill the GPU wi | th multiple "waves" | of blocks to hide me | n theoretical occup | ancy, device "Tesla | V100-SXM2-16GB" c | an simultaneously | single interva | al t |
| 3. Compute, Bandwidth, or Late | ncy Bound | | execute 8 blocks latency. If the ker | on each of the 80 S rnel is executing co | Ms, so the kernel n ncurrently with oth | nay need to execute er kernels then few | e a multiple of 640 b er blocks will be req | locks to hide the con juired because the ke | npute and memory rnel is sharing the | properties | |
| 4. Instruction and Memory Later | Profile dow Bun Help atatimeline100.nvprof S2 0 s 0.1 s 0.2 s 0.3 s 0.4 s 0.5 s 176960 446 176960 446 176960 446 176960 446 176960 446 176960 446 176960 446 176960 446 1769600 1769600 1769600 1769600 1769600 1769600 1769600 1769600 1769600 1769600 1769600 1769600 | | | 2 | | | | | | | |
| Instruction and memory latency lim kernel when the GPU does not hav busy. The results at right indicate t have enough work because the ke enough blocks. | it the performa re enough work hat the GPU do rnel does not e | nce of a A k to keep bes not xecute | Optimization: Inc | rease the number o | † blocks executed b | y the kernel. | | | More | | |
| 🖳 Examine Oc | cupancy | | | | | | | | | | |
| Occupancy is a measure of how many warps to relative to the maximum number of warps su occupancy provides an upper bound while ach | A Visual Profiler v Window Run Help | e on the GPU, Theoretical dicates the lay not be eccrupancy | | | | | | | | | |
| kernel's actual occupancy. For this kernel, exai useful until you modify the kernel to execute r analysis assumes there are enough blocks to f | nore blocks because fill the GPU. | occupancy | | | | | | | | | |

The kernel doesn't do enough work for the GPU.

CAK RIDGE

Visual Profiler – Guided Analysis – Latency Bound

| | , =_ ©, + <u>+</u> (= | | K 5 P 👌 - | | | | | | | |
|---|--|---|--|--|--|---|--|--|---------------|---|
| *single_gpu_data.timeline100.nv | /prof 🚺 *vec_add_cud | a.timeline.nvprof 🖾 | | | | | | | | |
| | 0 s 0.05 s | 0.1 s | 0.15 s 0. | 2 s (| 0.25 s | 0.3 s | 0.35 s | 0.4 s | 0. | 45 s |
| Process "run" (2129) | | • | | | | | • | · | | |
| Thread 288400 | | | | | | | | | | |
| - Runtime API | | | | | | | | | | |
| L Driver API | | | | | | | | | | |
| Profiling Overhead | | | | | | | | | 1 | |
| [0] Tesla V100-SXM2-16GB | | | | | | | | | | |
| Context 1 (CUDA) | | | | | | | | | | |
| └ ▼ MemCpv (HtoD) | | | | | | | | | | |
| MemCov (DtoH) | | | | | | | | | | |
| | | | | | | | | | | |
| 🗔 Analysis 🛛 🧱 GPU Details (S | ummary) 🔠 CPU Details [| 🕡 OpenACC Details [| 🖥 OpenMP Details 🛛 🖳 C | onsole 🔚 Settin | igs | | | ' | | Prop |
| 📃 🗄 🗘 🛛 🛄 Exp | ort PDF Report | Results | | | | | | | | |
| | | i Osamana la N | | | | | | | · ^ | |
| 1. UUUA Application Analysis | | 1 Occupancy is in | lot Limiting Kernel Ferr | ormance | | | | | | add_vecto |
| | | The kernel's block s | ize, register usage, and sł | ormance nared memory usa | age allow it to fu | lly utilize all wa | rps on the GPU. | More. | | add_vecto |
| 2. Performance-Critical Kernels | | The kernel's block s | ize, register usage, and sł Achieved | nared memory usa | age allow it to fu | Ily utilize all wa Grid Size: [40 | rps on the GPU. 096,1,1] (4096 block | More. s)Block Size: [256, | | add_vecto Queue Submi |
| 2. Performance-Critical Kernels 3. Compute, Bandwidth, or Latene | cy Bound | The kernel's block s Variable Occupancy Per SM | ize, register usage, and sh | nared memory usa | age allow it to fu Device Limit | Ily utilize all wa Grid Size: [40 | rps on the GPU. 096,1,1] (4096 block | <u>More.</u> ss)Block Size: [256, | <u></u> 1 | add_vecto Queue Submi Start |
| CODA Application Analysis A Performance-Critical Kernels Compute, Bandwidth, or Latence Instruction and Memory Latence | cy Bound | The kernel's block s Variable Occupancy Per SM | ize, register usage, and sh | nared memory usa | age allow it to fu Device Limit | Ily utilize all wa Grid Size: [40 | rps on the GPU. 096,1,1] (4096 block | More. s)Block Size: [256, | <u>"</u> 1 | add_vecto Queue Submi Start End |
| CODA Application Analysis 2. Performance-Critical Kernels 3. Compute, Bandwidth, or Latence 4. Instruction and Memory Latence | cy Bound cy | The kernel's block s Variable Occupancy Per SM Active Blocks | ize, register usage, and si Achieved | arred memory usa | age allow it to fu Device Limit 32 | Ily utilize all wa Grid Size: [40 | rps on the GPU. 096,1,1] (4096 block | More. s)Block Size: [256, 24 28 32 | <u>.</u> 1 | Add_vector Queue Submi Start End Durati |
| CODA Application Analysis 2. Performance-Critical Kernels 3. Compute, Bandwidth, or Latence 4. Instruction and Memory Latence Instruction and memory latency limit kernel when the or U does not have | cy Bound Cy t the performance of a ^ enough work to keep | The kernel's block s Variable Occupancy Per SM Active Blocks Active Warps | ize, register usage, and sk Achieved | arred memory usa | age allow it to fu Device Limit 32 64 | Ily utilize all wa | rps on the GPU. 096,1,1] (4096 block 8 12 16 20 | More. ss)Block Size: [256, 24 28 32 | <u></u> 1 | Add_vector Queue Submi Start End Duratii Stream Grid Si |
| CODA Application Analysis 2. Performance-Critical Kernels 3. Compute, Bandwidth, or Latend 4. Instruction and Memory Latend Instruction and memory latency limit kernel when the orth does not have busy. The performance of latency lim be improved by increasing occurany | cy Bound cy t the performance of a e enough work to keep aited kernels can often ov. Oromency is a | The kernel's block s Variable Occupancy Per SM Active Blocks Active Warps | ize, register usage, and sh Achieved | Theoretical 8 64 2010 | age allow it to fu Device Limit 32 64 | Ily utilize all wa Grid Size: [40 0 4 0 9 | rps on the GPU. 096,1,1] (4096 block 8 12 16 20 18 27 36 | More. ss)Block Size: [256, 24 28 32 45 54 664 | | add_vecto Queue Submi Start End Durati Strean Grid Si Block |
| CODA Application Analysis C. Performance-Critical Kernels Compute, Bandwidth, or Latence Instruction and Memory Latence Instruction and memory latency limit kernel when the ore does not have busy. The performance of latency lim be improved by increasing occupance measure of how many warps the ker | cy Bound cy t the performance of a enough work to keep eited kernels can often cy. Occopency is a rnel has active on the | The kernel's block s Variable Occupancy Per SM Active Blocks Active Warps Active Threads | ize, register usage, and si Achieved | Theoretical 8 64 2048 | age allow it to fu Device Limit 32 64 2048 | Ily utilize all wa Grid Size: [40 0 4 0 9 0 9 | rps on the GPU. 096,1,1] (4096 block 8 12 16 20 18 27 36 512 1024 | More. (s)Block Size: [256, 24 28 32 45 54 664 1536 2048 | <u>.</u> 1 | add_vecto Queue Submi Start End Durati Strean Grid Si Block |
| CODA Application Analysis Compute, Bandwidth, or Latend Compute, Bandwidth, or Latend Instruction and Memory Latend Instruction and memory latency limit kernel when the or b does not have busy. The performance of latency lim be improved by increasing occupancy measure of how many warps the ker GPU, relative to the maximum numb the GPU. Theoretical occupancy prov | cy Bound cy t the performance of a : enough work to keep bited kernels can often cy. Occupency is a rnel has active on the er of warps supported by vides an upper bound | The kernel's block s Variable Occupancy Per SM Active Blocks Active Warps Active Threads Occupancy | Sterner Period ize, register usage, and sh Achieved 53.8 84.1% | ared memory usa Theoretical 8 64 2048 100% | age allow it to fu Device Limit 32 64 2048 100% | Ily utilize all war Grid Size: [40 0 4 0 9 0 4 | rps on the GPU. 096,1,1] (4096 block 8 12 16 20 18 27 36 512 1024 | More. ss)Block Size: [256, 24 28 32 45 54 664 1536 2048 | | add_vecto Queue Submi Start End Durati Stream Grid Si Block Regist Shareo |
| CODA Application Analysis 2. Performance-Critical Kernels 3. Compute, Bandwidth, or Latence 4. Instruction and Memory Latence Instruction and memory latency limit kernel when the OFU does not have busy. The performance of latency lim be improved by increasing occupancy measure of how many warps the ker GPU, relative to the maximum numb the GPU. Theoretical occupancy prov while achieved occupancy indicates orcupancy | cy Bound Cy t the performance of a e enough work to keep oited kernels can often cy. Occopency is a rinel has active on the er of warps supported by vides an upper bound the kernel's actual | The kernel's block s Variable Occupancy Per SM Active Blocks Active Warps Active Threads Occupancy | 53.8 84.1% | Theoretical 8 64 2048 100% | age allow it to fu Device Limit 32 64 2048 100% | Ily utilize all war Grid Size: [40 0 4 0 9 0 5 0% | rps on the GPU. 096,1,1] (4096 block 8 12 16 20 18 27 36 512 1024 25% 50% | More. ss)Block Size: [256, 24 28 32 45 54 664 1536 2048 75% 100% | | add_vecto Queue Submi Start End Durati Strean Grid Si Block Regist Shareo Launci |
| CODA Application Analysis C. Performance-Critical Kernels S. Compute, Bandwidth, or Latence A. Instruction and Memory Latence Instruction and memory latency limit kernel when the GPU does not have busy. The performance of latency lim be improved by increasing occupancy measure of how many warps the ker GPU, relative to the maximum numb the GPU. Theoretical occupancy prov while achieved occupancy indicates occupancy. | cy Bound Cy t the performance of a e enough work to keep aited kernels can often cy. Occupency is a rnel has active on the er of warps supported by vides an upper bound the kernel's actual | The kernel's block s Variable Occupancy Per SM Active Blocks Active Warps Active Threads Occupancy Warps | 53.8 84.1% | ared memory usa Theoretical 8 64 2048 100% | age allow it to fu Device Limit 32 64 2048 100% | Ily utilize all wa Grid Size: [40 0 4 0 9 0 9 | rps on the GPU. 096,1,1] (4096 block 8 12 16 20 18 27 36 512 1024 25% 50% | More. ss)Block Size: [256, 24 28 32 45 54 664 1536 2048 75% 100% | | add_vector Queue Submi Start End Durati Strean Grid Si Block Regist Sharec Launci |
| CODA Application Analysis C. Performance-Critical Kernels G. Compute, Bandwidth, or Latence A. Instruction and Memory Latency limit kernel when the one does not have busy. The performance of latency limit kernel when the one does not have busy. The performance of latency limit be improved by increasing occupancy measure of how many warps the ker GPU, relative to the maximum numb the GPU. Theoretical occupancy prov while achieved occupancy indicates occupancy. Examine Occu | cy Bound cy t the performance of a enough work to keep aited kernels can often cy. Occopency is a rnel has active on the er of warps supported by vides an upper bound the kernel's actual | The kernel's block s Variable Occupancy Per SM Active Blocks Active Warps Active Threads Occupancy Warps Threads/Block | 53.8 84.1% | ared memory usa Theoretical 8 64 2048 100% | age allow it to fu Device Limit 32 64 2048 100% | Ily utilize all wa Grid Size: [40 0 4 0 9 0 9 0 5 0% | rps on the GPU. 096,1,1] (4096 block 8 12 16 20 18 27 36 512 1024 25% 50% | More. s)Block Size: [256, 24 28 32 45 54 664 1536 2048 75% 100% 768 1024 | | add_vecto Queue Submi Start End Durati Stream Grid Si Block Regist Shareo Launci ✓ Occup Ac |
| CODA Application Analysis C. Performance-Critical Kernels Compute, Bandwidth, or Latend Compute, Bandwidth, or Latend Instruction and Memory Latency limit Kernel when the or & does not have busy. The performance of latency limit kernel when the or & does not have busy. The performance of latency limit kernel when the or & does not have busy. The performance of latency limit kernel when the or & does not have busy. The performance of latency limit kernel when the or & does not have busy. The performance of latency limit kernel when the or & does not have busy. The performance of latency limit kernel when the or & does not have busy. The performance of latency limit kernel when the or & does not have busy. The performance of latency limit kernel when the or & does not have busy. The performance of latency limit kernel when the or & does not have busy. The performance of latency limit kernel when the or & does not have busy. The performance of latency limit kernel when the or & does not have busy. The performance of latency limit kernel when the or & does not have busy. The performance of latency limit kernel when the or & does not have busy. The performance of latency limit kernel when the or & does not have busy. The performance of latency limit kernel when the or & does not have busy. The performance of latency limit kernel when the or & does not have busy. The performance of latency limit kernel when the or & does not have busy. The performance of latency limit kernel when the or & does not have limit the or & does not have busy. The performance of latency limit the or & does coupancy indicates coupancy | cy Bound Cy t the performance of a enough work to keep aited kernels can often cy. Occupency is a rnel has active on the er of warps supported by vides an upper bound the kernel's actual upancy e kernel has active on the GPU, | The kernel's block s Variable Occupancy Per SM Active Blocks Active Warps Active Threads Occupancy Warps Threads/Block Warps/Block | Standing Kerner Period | ared memory usa Theoretical 8 64 2048 100% 256 8 | age allow it to fu Device Limit 32 64 2048 100% 1024 32 | Ily utilize all war Grid Size: [40 0 4 0 9 0 9 0 9 0 9 0 9 | rps on the GPU. 096,1,1] (4096 block 8 12 16 20 18 27 36 512 1024 25% 50% | More. (s)Block Size: [256, 24 28 32 45 54 654 1536 2048 75% 100% 75% 100% | | add_vector Queue Submi Start End Duratii Stream Grid Si Block Regist Shareo Launci ✓ Occup Ac Th ✓ Shareo |
| CODA Application Analysis Control Application Analysis Compute, Bandwidth, or Latence S. Compute, Bandwidth, or Latence A. Instruction and Memory Latence Instruction and memory latency limit kernel when the GPU does not have busy. The performance of latency limit kernel when the GPU does not have busy. The performance of latency limit kernel when the GPU does not have busy. The performance of latency limit kernel when the GPU does not have busy. The performance of latency limit kernel when the GPU does not have busy. The performance of latency limit kernel when the GPU does not have busy. The performance of latency limit kernel when the GPU does not have busy. The performance of latency limit kernel when the GPU does not have busy. The performance of latency limit kernel when the GPU does not have busy. The performance of latency limit kernel when the GPU does not have busy. The performance of latency limit kernel when the GPU does not have busy. The performance of latency limit kernel when the GPU does not have busy. The performance of latency limit kernel when the GPU does not have busy. The performance of latency limit kernel when the GPU does not have busy. The performance of latency limit kernel when the GPU does not have busy. The performance of latency limit kernel when the GPU does not have busy. The performance of latency limit kernel when the GPU does not have busy. The performance of latency limit kernel when the GPU does not have busy. The performance of latency limit kernel when the GPU does not have busy. The performance of latency limit kernel when the GPU does not have busy. The performance of latency limit kernel when the GPU does not have busy. The performance of latency limit kernel when the does not have busy. The performance of latency limit kernel when the maximum number of warps support cupancy provides an upper bound while achieve busy. The performance of latency limit kernel when the maximum number of warps support cupancy limit kernel when the maximum number of | cy Bound Cy t the performance of a e enough work to keep aited kernels can often cy. Occopency is a rnel has active on the er of warps supported by vides an upper bound the kernel's actual upancy e kernel has active on the GPU, poorted by the GPU. Theoretical wed occupancy indicates the | The kernel's block s Variable Occupancy Per SM Active Blocks Active Warps Active Threads Occupancy Warps Threads/Block Warps/Block | 53.8 84.1% | ared memory usa hared memory usa 8 64 2048 100% | age allow it to fu Device Limit 32 64 2048 100% | Ily utilize all wat Grid Size: [40 0 4 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 0 9 | rps on the GPU. 096,1,1] (4096 block 8 12 16 20 18 27 36 512 1024 25% 50% 256 512 8 12 16 20 | More. ss)Block Size: [256, 24 28 32 45 54 664 1536 2048 75% 100% 768 1024 24 28 32 | | add_vector Queue Submi Start End Duratii Stream Grid Si Block Regist Shared Launci V Occup Ac Th V Shared Shared |
| CODA Application Analysis C. Performance-Critical Kernels G. Compute, Bandwidth, or Latence A. Instruction and Memory Latence Instruction and memory latency limit kernel when the orth does not have busy. The performance of latency lim be improved by increasing occupancy measure of how many warps the ker GPU, relative to the maximum numbe the GPU. Theoretical occupancy indicates occupancy. Examine Occu Occupancy is a measure of how many warps th relative to the maximum number of warps sup occupancy provides an upper bound while achieved kernel's actual occupancy. | cy Bound Cy t the performance of a e enough work to keep aited kernels can often cy. Occupency is a rnel has active on the er of warps supported by vides an upper bound the kernel's actual upancy e kernel has active on the GPU, ported by the GPU. Theoretical eved occupancy indicates the | The kernel's block s Variable Occupancy Per SM Active Blocks Active Warps Active Threads Occupancy Warps Threads/Block Warps/Block Block Limit | 53.8 84.1% | ared memory usa Theoretical 8 64 2048 100% 256 8 8 8 | age allow it to fu Device Limit 32 64 2048 100% 1024 32 32 | Ily utilize all wa Grid Size: [40 0 4 0 4 0 9 0 9 0 6 0 4 0 4 0 4 0 4 0 4 0 4 0 4 0 4 | rps on the GPU. 096,1,1] (4096 block 8 12 16 20 18 27 36 512 1024 25% 50% 256 512 8 12 16 20 8 12 16 20 8 12 16 20 | More. ss)Block Size: [256, 24 28 32 45 54 664 1536 2048 75% 100% 758 1024 24 28 32 24 28 32 | | add_vector Queue Submi Start End Durati Strean Grid Si Block Regist Shareo Launci V Occup Ac Th Shareo Sh Sh Sh |

In other cases an occupancy analysis may be performed.

CAK RIDGE National Laboratory

Visual Profiler – Guided Analysis – Compute Bound

| WNDA Visual Profile WNDA Visual Profile Withdow Bunk Left Withdow Bunk Left Process TMR Bank 0* (100 T) Process TMR Bank 0* (100 T) Process TMR Bank 0* (100 T) Withdow Bank 0* (100 T) Process TMR Bank 0* (100 T) Withdow Bank 0* (100 T) Process TMR Bank 0* (100 T) Withdow Bank 0* (100 T) Process TMR Bank 0* (100 T) Withdow Bank 0* (100 T) Withdow Bank 0* (100 T) Withdow Bank 0* (100 T) Process TMR Bank 0* (100 T) Withdow Bank 0* (100 T) Withdow Bank 0* (100 T) Withdow Bank 0* (100 T) Process TMR Bank 0* (100 T) Withdow Bank 0* (100 T) Withdow Bank 0* (100 T) Withdow Bank 0* (100 T) Process TMR Bank 0* (100 T) Withdow Bank 1* (100 T) Wi | | | | | | | | | | | | |
|--|---|--|---|------------------------------------|-------------------------------------|--|--|------------------------------------|----------------------------|---------------------|--|--------------|
| NUDUA Visual Prefie NUDUA Visual Prefie Numerical Prefix | | | | | | | | | | | | |
| WNDUA Visual Profile With visual Prof | | | | | | | | | | | | |
| | 0 s 0.25 s | 0.5 s | 0.75 s | 1 s | 1.25 s | 1.5 s | 1.75 s | 2 s | 2.25 s | 2.5 s | 2.75 s | |
| Process "MPI Rank 0" (180175) | | | | | | | | | | | | |
| Thread 310560 | | | | | | | | | | | | |
| Runtime API | | | | | | | | | | | | |
| L Driver API | | | | | | | | | | | | |
| Markers and Ranges | | | | | | | | | | | | |
| L Default Domain | | | | | | | | | | | | |
| Profiling Overhead | | | | | | | | | | | | |
| [0] Tesla V100-SXM2-16GB | | | | | | | | | | | | |
| Context MPI Rank 0 (CUDA) | | | | | | | | | | | | |
| NVDA Visual Profiler If visual Pr | \ | Prop | .ε | | | | | | | | | |
| E 🗄 🗘 🛛 🛄 Expor | t PDF Report | Results | | | | | | | | | | |
| MUDIA Visual Portier Work Mudow Ban Help If Yew Window Ban Help Trat.mol.mot.timeline.0urpert 22 Precess MR Rank 0" (100173) | | | | | | | | | | | | |
| 1. CODA Application Analysis | | 1 Kerne | | e is bound by | compute | | | | | - | | |
| 2. Performance-Critical Kernels | | For device | e "Tesla V100-S) cate that the pe | KM2-16GB" the erformance of | e kernel's memo the kernel is mo | ory utilization is ost likely being | significantly low limited by comp | ver than its con utation on the | npute utilization. SMs. | . These utilization | Select or h single inte | nigh erva |
| 2. Performance-Critical Kernels 3. Compute, Bandwidth, or Latence | Round | For device levels indi | e "Tesla V100-S) cate that the pe | KM2-16GB" the erformance of | e kernel's memo the kernel is mo | ory utilization is ost likely being | significantly low limited by comp | ver than its con utation on the | npute utilization. SMs. | . These utilization | Select or h single inte properties | nigh erva |
| 2. Performance-Critical Kernels 3. Compute, Bandwidth, or Latency The first step in analyzing an individua | Round | For device levels indi | e "Tesla V100-S) cate that the pe | XM2-16GB" the erformance of | e kernel's memo the kernel is mo | ory utilization is ost likely being | significantly low limited by comp | ver than its con utation on the | npute utilization. SMs. | . These utilization | Select or h single inte properties | nigh rva |
| Soute NVIDA Visual Profile Window Run Bip Imagind intectimation of the service Process MR Pank 01 (160178) 9 0.25 5 0.5 5 0.75 1 3 1.25 1.5 1.75 2 5 2.25 2 Process MR Pank 01 (160178) 9 0.25 5 0.5 5 0.75 1 3 1.25 5 1.5 1.75 2 5 2.25 2 2 Process MR Pank 01 (160178) 9 0.25 5 0.5 9 0.75 1 3 1.25 5 1.5 1.75 2 5 2.25 5 2 Process MR Pank 01 (160178) 9 0.25 5 0.5 9 0.75 1 3 1.25 5 1.5 1.75 2 5 2.25 5 2 Process MR Pank 01 (160178) 9 0.25 5 0.5 9 0.75 1 3 1.25 5 1.5 1.75 2 5 2.25 5 2 Process MR Pank 01 (160178) 9 0.25 5 0.5 9 0.75 1 3 1.25 5 1.5 1.75 1.75 2 5 2.25 5 2 9 0.5 0.5 0.5 0.75 1 3 1.25 5 1.5 1.75 1.75 2 5 2.25 5 2 9 0.5 0.5 0.5 0.75 1.5 1.75 1.75 2 5 2.25 5 2 9 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 | . These utilization | Select or h single inte properties | nigh erva | | | | | | | | | |
| NVIDIA Visual Profiler File View Window Run Help **mat_mul_nvtx.timeline.0.nvprof S 0's 0.25's 0.5's 0.75's 1's Process *MPI Rank 0' (180175) • Thread 310560 • Runtime API • Oriver API • Default Domain • Default Domain • Profiling Overhead • Default Domain • Default Domain • Profiling Overhead • Default Rank 0 (CUDA) • Matters and Ranges • Default CUDA • Matters and Ranges • Default Domain • Profiling Overhead • Oriver API • Matters and Ranges • Default Domain • Profiling Overhead • Default CUDA • Matters and Ranges • Default Domain • Profiling Overhead • OpenACC Details @ OpenAPC Details @ • OpenAPC Details @ Op | e kernel's memo the kernel is mo | ory utilization is ost likely being | significantly low limited by comp | ver than its con utation on the | npute utilization. SMs. | . These utilization | Select or h single inte properties | nigh erva | | | | |
| CODA Application Analysis Compute, Bandwidth, or Latency The first step in analyzing an individua determine if the performance of the ke computation, memory bandwidth, or i latency. The results at right indicate th kernel "volta_dgemm_64x64_nn" is mos compute. | Round I kernel is to rrnel is bounded by nstruction/memory at the performance of it likely limited by | For device levels indi | "Tesla V100-S) cate that the pe | KM2-16GB" the | e kernel's memo | ny utilization is ost likely being | significantly low limited by comp | ver than its con utation on the | npute utilization. SMs. | . These utilization | Select or h single inte properties | nigh erva |
| CODA Application Analysis Compute, Bandwidth, or Latency The first step in analyzing an individua determine if the performance of the ke computation, memory bandwidth, or i latency. The results at right indicate th kernel "volta_dgemm_64x64_nn" is more compute. | Pound I kernel is to ernel is bounded by natruction/memory hat the performance of it likely limited by | For device levels indi | 2 "Tesla V100-S) cate that the pe | KM2-16GB" the | e kernel's memo | ny utilization is ist likely being | significantly low limited by comp | ver than its con utation on the | npute utilization. SMs. | . These utilization | Select or h single inte properties | nigh erva |
| CODA Application Analysis Constant | Bound I kernel is to rrnel is bounded by nstruction/memory at the performance of t likely limited by Analysis | For device levels indi | "Tesla V100-S) cate that the pe 90% 80% 70% 60% 50% | KM2-16GB" the | e kernel's memo the kernel is mo | ory utilization is ost likely being | significantly low limited by comp | ver than its con utation on the | npute utilization. SMs. | . These utilization | Select or h single inte properties | nigh erva |
| CODA Application Analysis Constant Constant Analysis Constant Analysis Constant Analysis | Round I kernel is to rnel is bounded by nstruction/memory lat the performance of it likely limited by Analysis kernel is compute so you a how it is limiting | For device levels indi | "Tesla V100-S) cate that the pe 00% 90% 80% 70% 60% 50% 40% | KM2-16GB" the | e kernel's memo the kernel is mo | ny utilization is ist likely being | significantly low limited by comp | ver than its con utation on the | npute utilization. SMs. | . These utilization | Select or h single inte properties | nigh erva |
| CODA Application Analysis Concern and a second analysis Compute, Bandwidth, or Latency The first step in analyzing an individual determine if the performance of the key computation, memory bandwidth, or i latency. The results at right indicate the kernel "volta_dgemm_64x64_nn" is most compute. Perform Compute The most likely bottleneck to performance for this should first perform compute analysis to determine performance. | Pound I kernel is to rmel is bounded by nstruction/memory at the performance of it likely limited by Analysis emel is compute so you e how it is limiting | For device levels indi | * Tesla V100-S) cate that the period 90% 80% 70% 60% 50% 40% 30% | KM2-16GB" the | e kernel's memo the kernel is mo | ny utilization is ist likely being | significantly low limited by comp | ver than its con utation on the | npute utilization. SMs. | . These utilization | Select or h single inte properties | nigh erva |
| CODA Application Analysis Constant | Round I kernel is to rrnel is bounded by nstruction/memory hat the performance of the tikely limited by Analysis eemel is compute so you how it is limiting | For device levels indi | 2 "Tesla V100-S) cate that the per- 90% 80% 70% 60% 50% 40% 30% 20% | KM2-16GB" the | e kernel's memo the kernel is mo | ny utilization is | significantly low limited by comp | ver than its con utation on the | npute utilization. SMs. | . These utilization | Select or h single inte properties | nigh rrva |
| CODE Application Analysis Construct A second analysis Compute, Bandwidth, or Latency The first step in analyzing an individual determine if the performance of the key computation, memory bandwidth, or i latency. The results at right indicate the kernel "volta_dgemm_64x64_nn" is more compute. Perform Compute The most likely bottleneck to performance for this should first perform compute analysis to determine performance. Perform Latency A Perform Memory Bandwidth | VIDIA Visual Profiler View Window Run Help Imat_mul_nvtx.timeline.0.nvprof Imat_mul_nvtx.timprof Imat_mul_nvtx.timeline.0.nvprof <td>For device levels indi</td> <td>"Tesla V100-S) cate that the person of the second secon</td> <td>KM2-16GB" the</td> <td>e kernel's memo the kernel is mo</td> <td>ny utilization is ost likely being</td> <td>significantly low limited by comp</td> <td>ver than its con utation on the</td> <td>npute utilization. SMs.</td> <td>These utilization</td> <td>Select or h single inte properties</td> <td>nigh erva</td> | For device levels indi | "Tesla V100-S) cate that the person of the second secon | KM2-16GB" the | e kernel's memo the kernel is mo | ny utilization is ost likely being | significantly low limited by comp | ver than its con utation on the | npute utilization. SMs. | These utilization | Select or h single inte properties | nigh erva |
| NUDDA Visual Profiler I W Monda Visual Profiler I Mark Control Profiler I Process MPI Rank Of (10173) I Process MPI Rank Of (10174) | | | | | | | | | | | | |

High compute utilization indicates the kernel is compute bound.

Now a compute analysis is suggested

CAK RIDGE

Visual Profiler – Guided Analysis – Compute Bound

| | = | u 🔍 🖌 🖯 | Q 🔍 🖃 | F 🔨 📕 | 📮 📇 📩 | • | | | | | | | | | |
|--|---|---------------------------------|------------------------|--|------------------------------------|--------------------------------------|------------------------------------|--------------------------------------|--|-----------------------------------|-------------------------------|-----|------------------------------|---------------------|---|
| *mat_mul_nvtx.timeline.0.n | nvprof ⊠ | | | | | | | | | | | | | - | 7 |
| | 0 s | 0.25 s | 0.5 s | 0.75 s | 1 s | 1.25 s | 1.5 s | 1.75 s | 2 s | 2.25 s | 2.5 s | 2.7 | 5 s | 3 s | |
| Process "MPI Rank 0" (18017 | 75) | | | | | | | | | | | | | | |
| Thread 310560 | | | | | | | | | | | | | | | |
| Runtime API | | | | | | | | | | | | | | | _ |
| 🔚 Analysis 🖾 📑 GPU Deta | ails (Summary) | E CPU Details | s 🗇 OpenACC | C Details 🕅 🗐 Op | enMP Details | 🚽 Console 🗖 | Settings | | | | \ [_] | | Prop | ΞĽ. | _ |
| | Export PDF Re | eport | Results | | | | | | | | | | | | |
| 1. CUDA Application Analysis | ; | | 💧 GPU | Utilization Is L | imited By Fund | tion Unit Usag | | | | | | | | | |
| 2. Performance-Critical Kern | IA Visual Profiler w Window Run Help mul_nvtx.timeline.0.nvprof o s 0.2 s *MPI Rank 0° (180175) ead 310560 Runtime API sis | | Different is over-u | t types of instruc used by the instru | tions are execu uctions execute | ted on different d by the kernel. | tunction units The following | within each SM. results show that | . Performance car t the kernel's perf | n be limited if formance is po | a function unit stentially | | Select or hi single inter | ghlight val to s | 1 |
| 3. Compute, Bandwidth, or L | atency Bound | | limited b | by overuse of the | e following fun | ction units: Dou | ble. | stant menors | | | | | properties | | |
| 4 C | | | _ | Texture - Load | and store inst | ructions for loca | snared and cor I, global, and t | exture memory. | | | | | | | |
| 4. Compute Resources | | | | Half - Half-pr | ecision floating | -point arithmet | c instructions. | tic instructions | | | | | | | |
| GPU compute resources limit the those resources are insufficient | he performance t or poorly utili: | of a kernel whe zed. Compute | n | Double - Dou | ble-precision fl | oating-point ari | hmetic instruc | tions. | | | | | | | |
| resources are used most efficie overuse a function unit. The re- | ently when instr | uctions do not | | Special - Spec | ial arithmetic in | nstructions such | as sin, cos, po | pc, etc. Ic | | | | | | | |
| compute performance may be l | limited by overu | ise of a function | | Control-riow | - Direct and inc | meet branches, | jumps, and ca | 15, | | | | | | | |
| | | | | | | | | | | | | | | | |
| L Show Kernel Prom | ie - Instruction | Execution | | | | | | | | | | | | | |
| The kernel profile shows the execution of threads for each source and assembly lin | ount, inactive thread | s, and predicated | | | Llink | | | | | | | | | | |
| can pinpoint portions of your kernel that a | are making inefficier | nt use of compute | | | riign | | | | | | | | | | |
| | | | | evel | | | | | | | | | | | |
| ाम् Ren | un Analysis | | | u L a | | | | | | | | | | | |
| If you modify the kernel you need to reru | in your application to | update this a nalysis. | | zati | IMIED | | | | | | | | | | |
| | | | | 3 | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | Low | | _ | | _ | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |





"Poor Man's" Guided Analysis

- Sometimes you can get enough information from a simple nvprof run to get you started.
- Utilization will be shown as a scale from 1 (Low) to 10 (Max)

| \$ jsrun -n1 -c1 -a | g1 -a1 nvprof -m dram_utilization,12_v | tilization,double_precision_fu_utilization,ac | chieved_occup | ancy ./redun | dant_mm | |
|---|--|---|--|----------------------|----------|--|
| 2048 100 | | | | | | |
| ==13250== NVPROF is profiling process 13250, command: ./redundant_mm 2048 100 | | | Ideally, something will be "High" or "Max". If everything is "Low", | | | |
| ==13250== Some kernel(s) will be replayed on device 0 in order to collect all events/metrics. | | | | | | |
| ==13250== Profiling application: ./redundant_mm 2048 100 | | | | | | |
| (N = 2048) Max Total Time: 10.532436 Max GPU Time: 8.349185 | | | | and check occupancy. | | |
| Rank 000, HWThread 002, GPU 0, Node h49n16 - Total Time: 10.532436 GPU Time: 8.349185 🛛 🛛 🔾 | | | | | | |
| ==13250== Profili | ng result: | | | 1 | | |
| ==13250== Metric : | result: | | | | | |
| Invocations | Metric Name | Metric Description | Min | Max | Avg | |
| Device "Tesla V100 | 0-SXM2-16GB (0)" | | | L L | | |
| Kernel: volta | _dgemm_64x64_nn | | | | | |
| 100 | dram_utilization | Device Memory Utilization | Low (1) | Low (2) | Low (1) | |
| 100 | 12_utilization | L2 Cache Utilization | Low (2) | Low (2) | Low (2) | |
| 100 | double_precision_fu_utilization | Double-Precision Function Unit Utilization | Max (10) | Max (10) | Max (10) | |
| 100 | achieved occupancy | Achieved Occupancy | 0.114002 | 0.120720 | 0.118229 | |



Summary

- How to generate text and visual output using the NVIDIA profilers
- The workflow for using the NVIDIA profilers on Summit
 - Generate visual output remotely
 - Scp visual output to local machine
 - Explore using NVIDIA Visual Profiler on local machine
- A simple example of how the text+visual profiles might be used when porting an application to run on GPUs
- How to profile multiple MPI ranks (when not too many!)
- How to insert simple annotations into visual profiles using NVTX
- How to interpret Unified Memory results in the visual profiler
- How to perform remote kernel analysis



