ハンズオン
Lab 2

ハンズオンについて

- この講義は以下の教材をもとにしています

https://github.com/openhackathons-org/gpubootcamp/blob/58e1329572bebc508ba7489a9f9415d7e0592ab8/hpc/nways/nways_labs/nways_MD/English/Python/jupyter_notebook/cupy/cupy_guide.ipynb

- Lab 1
  - CuPy による GPU コンピューティングの講義
  - ハンズオン
    - Exercise 1-4 を実施
- **Lab 2**
  - **ハンズオンで使うコードの概要説明**
  - **Nsight Systems の概要説明**
  - ハンズオン
    - Lab Task を実施

The Serial Code

dcdreadhead and dcdreadframe

- 以下の Cell 1 を参照
- https://github.com/openhackathons-org/gpubootcamp/blob/58e1329572bebc508ba7489a9f9415d7e0592ab8/hpc/nways/nways_labs/nways_MD/English/Python/jupyter_notebook/cupy/serial_RDF.ipynb
- または、/work/EDU5/ユーザ名/hands-on/Lab2/nways_serial.py を参照
- dcdreadhead : DCDFile からフレーム数、原子数を読み込み
- dcdreadframe : MDAnalysis ライブラリを使って、10 (6720 atoms / frame) フレームを読み込み
- 上記 2 つの関数はホストで実行

The Serial Code

pair_gpu

```
def pair_gpu(d_x, d_y, d_z, d_g2, numatm, nconf, xbox, ybox, zbox, d_bin):
    box = min(xbox, ybox)
    box = min(box, zbox)

    _del = box / (2.0 * d_bin)
    cut = box * 0.5

    print("\n {} {}".format(nconf, numatm))

    for frame in range(nconf):
        print("\n {}".format(frame))
        for id1 in range(numatm):
            for id2 in range(numatm):
                dx = d_x[frame * numatm + id1] - d_x[frame * numatm + id2]
                dy = d_y[frame * numatm + id1] - d_y[frame * numatm + id2]
                dz = d_z[frame * numatm + id1] - d_z[frame * numatm + id2 ]

                dx = dx - xbox * (round(dx / xbox))
                dy = dy - ybox * (round(dy / ybox))
                dz = dz - zbox * (round(dz / zbox))

                r = math.sqrt(dx * dx + dy * dy + dz * dz)
                if r < cut :
                    ig2  = int((r/_del))
                    d_g2[ig2] = d_g2[ig2] + 1
```

10 x

6720 x

6720 x

Function that computes differences in xyz DCDframes

Operation

1. Subtraction
2. Division, multiplication, and subtraction

value update on distinct element of **d_g2** and may be incremental during the loops

The Serial Code

`pair_gpu`

NSIGHT SYSTEMS

System profiler

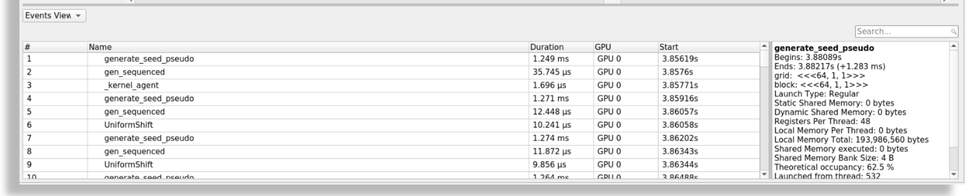
Key Features:

- System-wide application algorithm tuning
  - Multi-process tree support
- Locate optimization opportunities
  - Visualize millions of events on a very fast GUI timeline
  - Or gaps of unused CPU and GPU time
- Balance your workload across multiple CPUs and GPUs
  - CPU algorithms, utilization and thread state
    GPU streams, kernels, memory transfers, etc
- Command Line, Standalone, IDE Integration

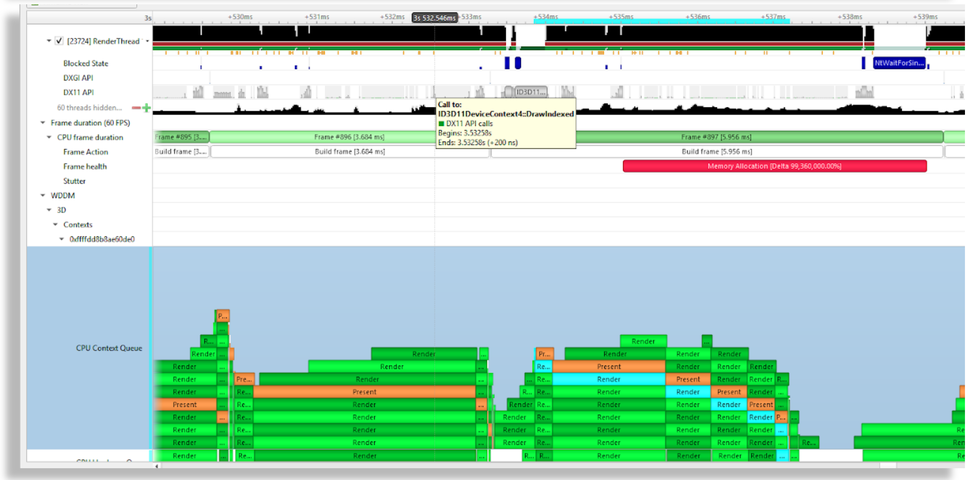
OS: Linux (x86, Power, Arm SBSA, Tegra), Windows, MacOSX (host)

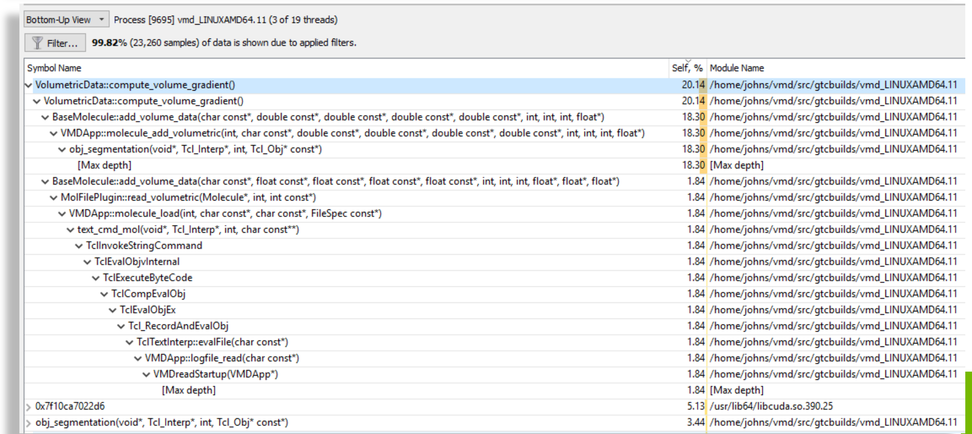
GPUs: Pascal+

Docs/product: https://developer.nvidia.com/nsight-systems

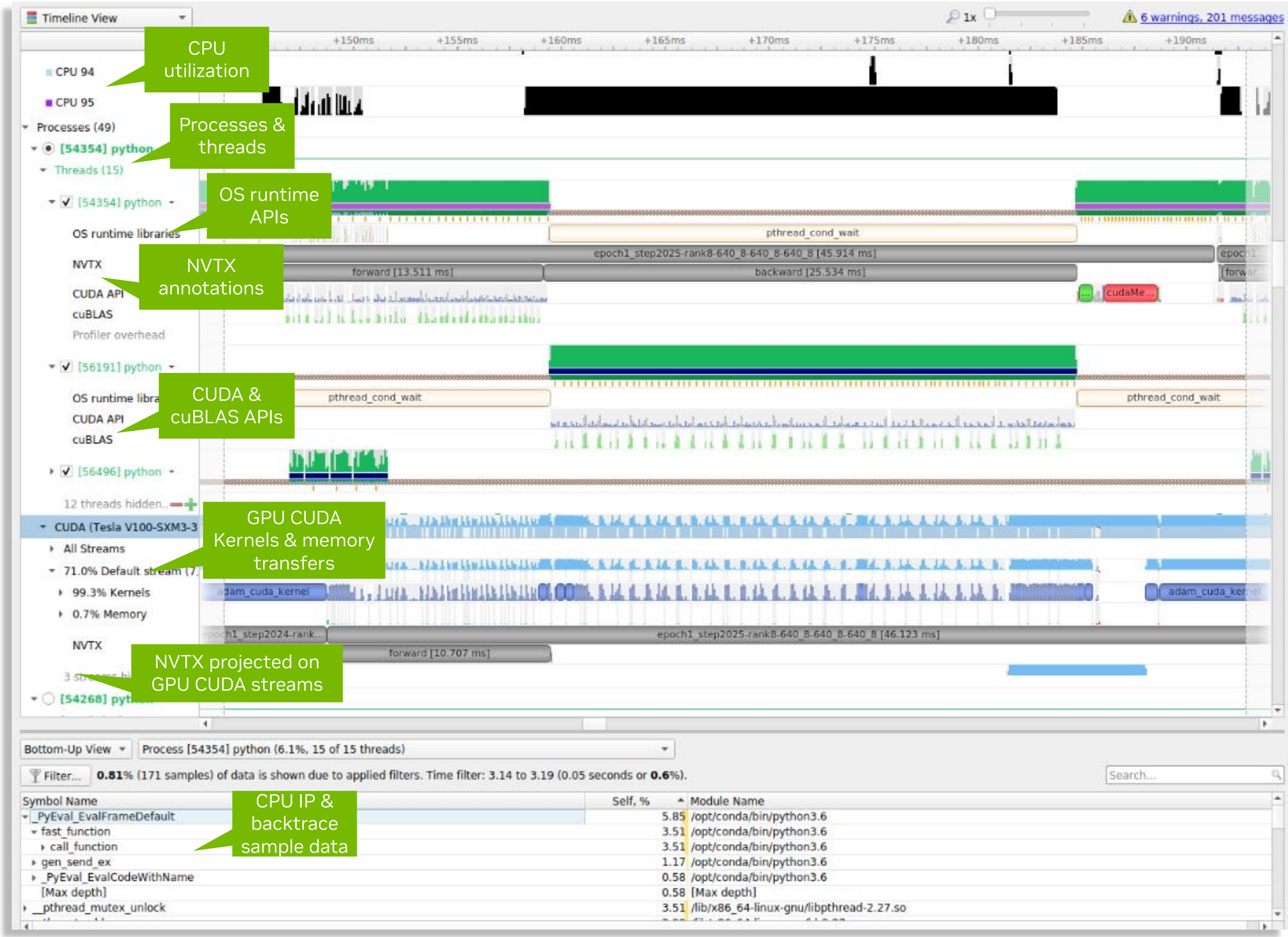


#	Name	Duration	GPU	Start
1	generate_seed_pseudo	1.249 ms	GPU 0	3.85619s
2	gen_sequenced	35.745 µs	GPU 0	3.8576s
3	_kernel_agent	1.696 µs	GPU 0	3.85771s
4	generate_seed_pseudo	1.271 ms	GPU 0	3.85916s
5	gen_sequenced	12.448 µs	GPU 0	3.86057s
6	UniformShift	10.241 µs	GPU 0	3.86058s
7	generate_seed_pseudo	1.274 ms	GPU 0	3.86202s
8	gen_sequenced	11.872 µs	GPU 0	3.86343s
9	UniformShift	9.856 µs	GPU 0	3.86344s





Symbol Name	Self, %	Module Name
VolumetricData::compute_volume_gradient()	20.14	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
VolumetricData::compute_volume_gradient()	20.14	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
BaseMolecule::add_volume_data(char const*, double const*, double const*, double const*, double const*, int, int, int, float*)	18.30	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
VMDApp::molecule_add_volumetric(int, char const*, double const*, double const*, double const*, double const*, int, int, int, float*)	18.30	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
obj_segmentation(void*, Tcl_Interp*, int, Tcl_Obj* const*)	18.30	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
[Max depth]	18.30	[Max depth]
BaseMolecule::add_volume_data(char const*, float const*, float const*, float const*, float const*, int, int, int, float*, float*, float*)	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
MolFilePlugin::read_volumetric(Molecule*, int, int const*)	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
VMDApp::molecule_load(int, char const*, char const*, FileSpec const*)	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
text_cmd_mol(void*, Tcl_Interp*, int, char const**)	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
TclInvokeStringCommand	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
TclEvalObjvInternal	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
TclExecuteByteCode	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
TclCompEvalObj	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
TclEvalObjEx	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
Tcl_RecordAndEvalObj	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
TclTextInterp::evalFile(char const*)	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
VMDApp::logfile_read(char const*)	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
VMDreadStartup(VMDApp*)	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
[Max depth]	1.84	[Max depth]
0x7f10ca7022d6	5.13	/usr/lib64/libcuda.so.390.25
obj_segmentation(void*, Tcl_Interp*, int, Tcl_Obj* const*)	3.44	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11


Timeline View
1x
6 warnings, 201 messages
+150ms
+155ms
+160ms
+165ms
+170ms
+175ms
+180ms
+185ms
+190ms
CPU utilization
CPU 94
CPU 95
Processes & threads
Processes (49)
[54354] python
Threads (15)
OS runtime APIs
[54354] python
OS runtime libraries
pthread_cond_wait
NVTX annotations
NVTX
epoch1_step2025-rank8-640_8-640_8-640_8 [45.914 ms]
forward [13.511 ms]
backward [25.534 ms]
CUDA API
cudaMe...
cuBLAS
Profiler overhead
[56191] python
CUDA & cuBLAS APIs
OS runtime libra
pthread_cond_wait
pthread_cond_wait
CUDA API
cuBLAS
[56496] python
12 threads hidden...
GPU CUDA Kernels & memory transfers
CUDA (Tesla V100-SXM3-3
All Streams
71.0% Default stream (7
99.3% Kernels
0.7% Memory
adam_cuda_kernel
adam_cuda_kernel
epoch1_step2024-rank...
epoch1_step2025-rank8-640_8-640_8-640_8 [46.123 ms]
NVTX
forward [10.707 ms]
NVTX projected on GPU CUDA streams
[54268] python
Bottom-Up View
Process [54354] python (6.1%, 15 of 15 threads)
Filter...
0.81% (171 samples) of data is shown due to applied filters. Time filter: 3.14 to 3.19 (0.05 seconds or 0.6%).
Search...
CPU IP & backtrace sample data
Symbol Name
Self, %
Module Name
_PyEval_EvalFrameDefault 5.85 /opt/conda/bin/python3.6
fast_function 3.51 /opt/conda/bin/python3.6
call_function 3.51 /opt/conda/bin/python3.6
gen_send_ex 1.17 /opt/conda/bin/python3.6
_PyEval_EvalCodeWithName 0.58 /opt/conda/bin/python3.6
[Max depth] 0.58 [Max depth]
__pthread_mutex_unlock 3.51 /lib/x86_64-linux-gnu/libpthread-2.27.so

Nsight Systems 101

Quick start

- Nsight Systems CLI によるプロファイリング

```
$ nsys profile [options] <application> [application-arguments]
```

  - `-t <parameters>` : トレースする API を指定。デフォルトは、`cuda, opengl nvtx, osrt`
  - `--stats <true|false>` : `true` でプログラム実行時の統計情報を標準出力に表示
  - `-o <filename>` : 出力ファイル名を指定
  - `--force-overwrite <true|false>` : `true` で出力ファイルの上書きを許可。デフォルトは `false`

  など... 詳細は、`nsys --help or nsys [specific command] --help` で確認可能

  - `<filename>.nsys-rep` が出力される
- ローカルに `<filename>.nsys-rep` を転送し、Nsight Systems UI で可視化

Nsight Systems user guide:
https://docs.nvidia.com/nsight-systems/UserGuide/index.html

Nsight Systems 101

Example

- 実行コマンド例

```
$ nsys profile  --stats true --force-overwrite true -o my_report python solution.py
```

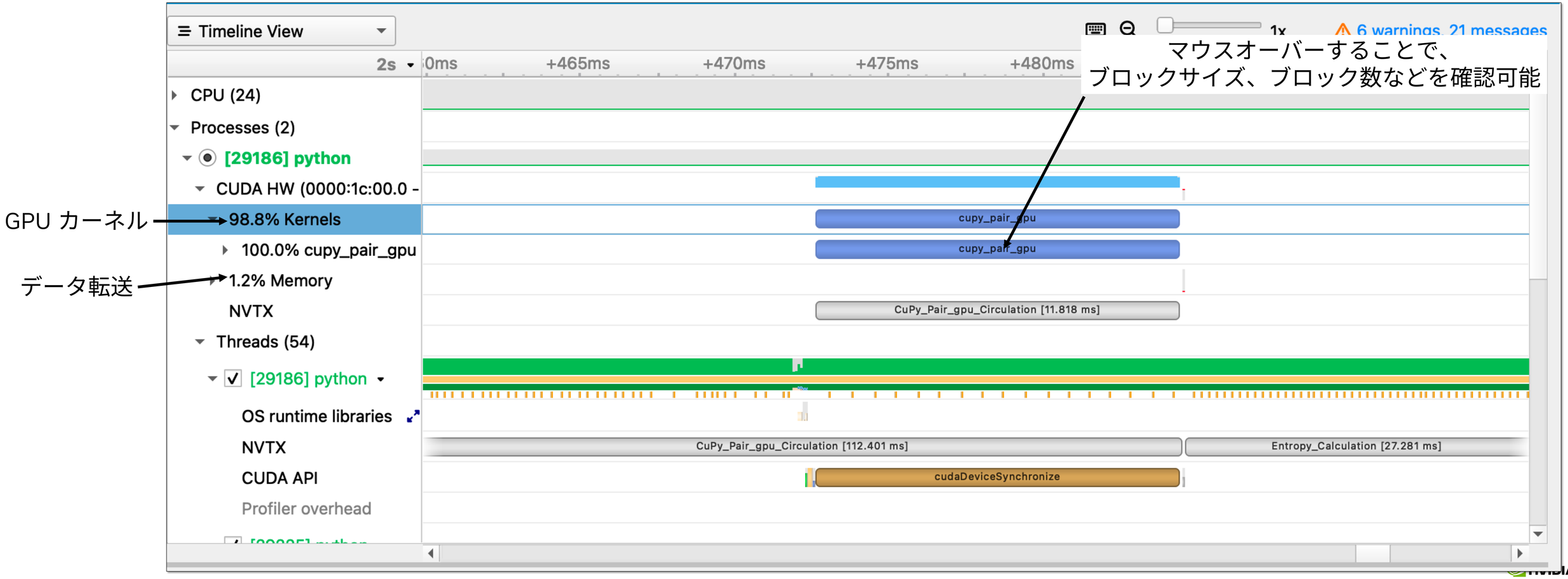
- 統計情報が標準出力に出力される

```
NVTX Range Statistics:

 Time (%)  Total Time (ns)  Instances     Avg (ns)         Med (ns)        Min (ns)       Max (ns)     StdDev (ns)   Style            Range
 --------  ---------------  ---------  ---------------  ---------------  -------------  -------------  -----------  -------  -------------------------
     87.9    1,015,559,769          1  1,015,559,769.0  1,015,559,769.0  1,015,559,769  1,015,559,769          0.0  PushPop  Read_File
      9.7      112,401,039          1    112,401,039.0    112,401,039.0    112,401,039    112,401,039          0.0  PushPop  CuPy_Pair_gpu_Circulation
      2.4       27,280,924          1     27,280,924.0     27,280,924.0     27,280,924     27,280,924          0.0  PushPop  Entropy_Calculation

...

[5/8] Executing 'cudaapisum' stats report

CUDA API Statistics:

 Time (%)  Total Time (ns)  Num Calls    Avg (ns)      Med (ns)     Min (ns)    Max (ns)     StdDev (ns)              Name
 --------  ---------------  ---------  ------------  ------------  ----------  ----------  -------------  ----------------------------
     94.3      219,106,094          4  54,776,523.5       8,807.5       6,978  219,081,501  109,536,651.7  cudaMalloc
      5.1       11,788,845          1  11,788,845.0  11,788,845.0  11,788,845   11,788,845            0.0  cudaDeviceSynchronize
      0.4          958,943          2     479,471.5     479,471.5       6,294      952,649      669,174.0  cudaHostAlloc
      0.0          106,326          1     106,326.0     106,326.0     106,326      106,326            0.0  cuModuleLoadData
      0.0           67,470          1      67,470.0      67,470.0      67,470       67,470            0.0  cuModuleUnload
...

[6/8] Executing 'gpukernsum' stats report

CUDA Kernel Statistics:

 Time (%)  Total Time (ns)  Instances    Avg (ns)      Med (ns)     Min (ns)    Max (ns)   StdDev (ns)      Name
 --------  ---------------  ---------  ------------  ------------  ----------  ----------  -----------  ------------
    100.0       11,817,865          1  11,817,865.0  11,817,865.0  11,817,865  11,817,865          0.0  cupy_pair_gpu

[7/8] Executing 'gpumemtimesum' stats report

CUDA Memory Operation Statistics (by time):

 Time (%)  Total Time (ns)  Count  Avg (ns)  Med (ns)  Min (ns)  Max (ns)  StdDev (ns)      Operation
 --------  ---------------  -----  --------  --------  --------  --------  -----------  ------------------
     98.1          143,295      4  35,823.8  46,207.5     3,232    47,648     21,738.5  [CUDA memcpy HtoD]
      1.9            2,784      1   2,784.0   2,784.0     2,784     2,784          0.0  [CUDA memcpy DtoH]

[8/8] Executing 'gpumemsizesum' stats report

CUDA Memory Operation Statistics (by size):

 Total (MB)  Count  Avg (MB)  Med (MB)  Min (MB)  Max (MB)  StdDev (MB)      Operation
 ----------  -----  --------  --------  --------  --------  -----------  ------------------
      1.629      4     0.407     0.538     0.016     0.538        0.261  [CUDA memcpy HtoD]
      0.016      1     0.016     0.016     0.016     0.016        0.000  [CUDA memcpy DtoH]
```

Nsight Systems 101

Example

- *.nsys-rep を手元の端末に転送
- 手元の端末で Nsight Systems を起動し、File->Open で *nsys-rep ファイルを選択



ハンズオンについて

- この講義は以下の教材をもとにしています

https://github.com/openhackathons-org/gpubootcamp/blob/58e1329572bebc508ba7489a9f9415d7e0592ab8/hpc/nways/nways_labs/nways_MD/English/Python/jupyter_notebook/cupy/cupy_guide.ipynb

- Lab 1
  - CuPy による GPU コンピューティングの講義
  - ハンズオン
    - Exercise 1-4 を実施
- **Lab 2**
  - ハンズオンで使うコードの概要説明
  - Nsight Systems の概要説明
  - **ハンズオン**
    - **Lab Task を実施**

ハンズオン

- Lab Task
  - `nways_serial.py` を CPU 実行し時間計測 (標準出力の `kernel compute time` に注目)
  - `nways_serial.py` を CuPy Raw Kernel を使って GPU 対応、実行、時間計測
  - 正しく GPU 化できたかを確認しながら進める (標準出力の s2 と s2bound が以下の値であれば OK)

```
s2 value is -2.43191
s2bond value is -3.87014
```

  - 必要に応じて、前記講義資料で復習
  - (時間があれば、Nsight Systems を使ったプロファイリングにもチャレンジしてみましょう)

回答例

```
raw_kernel = cp.RawKernel(r'''
extern "C" __global__
void cupy_pair_gpu(const double* d_x, const double* d_y, const double* d_z,
             unsigned long long int *d_g2, int numatm, int nconf,
             double xbox, double ybox, double zbox, int d_bin)
{
    double r, cut, dx, dy, dz;
    int ig2;
    double box;
    box = min(xbox, ybox);
    box = min(box, zbox);

    double del = box / (2.0 * d_bin);
    cut = box * 0.5;

    int id1 = blockIdx.y * blockDim.y + threadIdx.y;
    int id2 = blockIdx.x * blockDim.x + threadIdx.x;

    if (id1 >= numatm || id2 >= numatm) return;
    if (id1 > id2) return;

    for (int frame = 0; frame < nconf; ++frame) {
        dx = d_x[frame * numatm + id1] - d_x[frame * numatm + id2];
        dy = d_y[frame * numatm + id1] - d_y[frame * numatm + id2];
        dz = d_z[frame * numatm + id1] - d_z[frame * numatm + id2];

        dx = dx - xbox * (round(dx / xbox));
        dy = dy - ybox * (round(dy / ybox));
        dz = dz - zbox * (round(dz / zbox));

        r = sqrtf(dx * dx + dy * dy + dz * dz);
        if (r < cut) {
            ig2 = (int)(r / del);
            atomicAdd(&d_g2[ig2], 2);
        }
    }
}
''', 'cupy_pair_gpu')
```

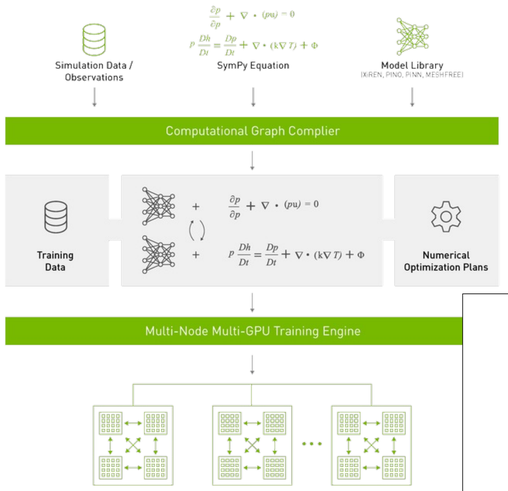
NVIDIA Modulus

NVIDIA Modulus

Platform for developing Physics ML surrogate model

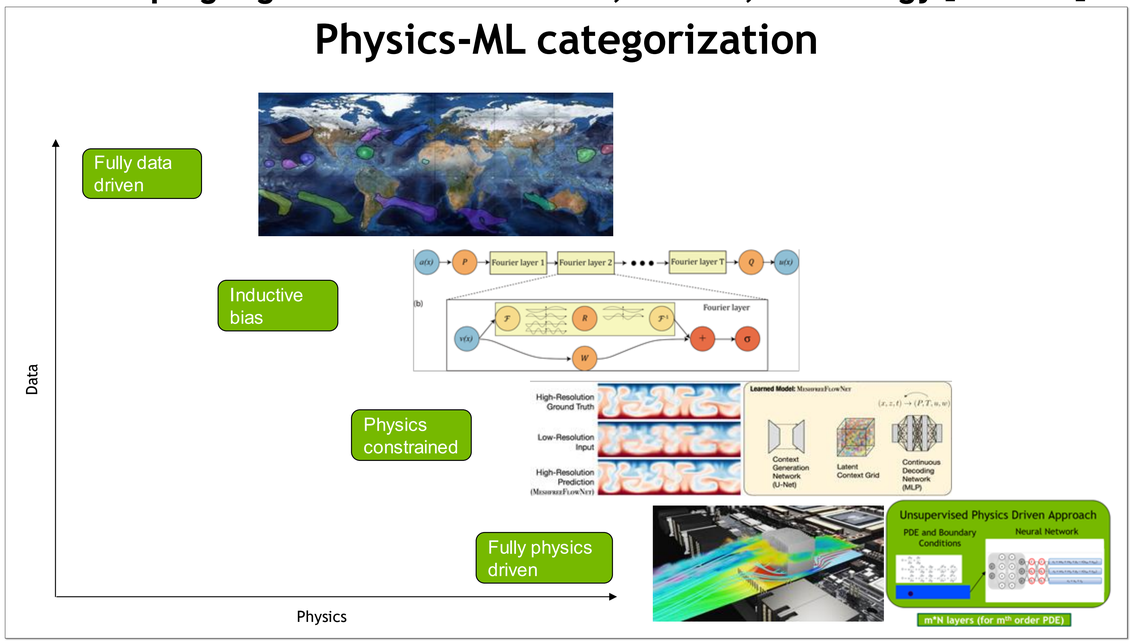
- A training and inference pipeline - using Physics (governing equations) and Data (simulation/observations)
- Customizable and scalable platform
- Higher level of abstraction for domain experts
- Build generalized AI surrogates for parameterized domain
- Near real-time high-fidelity simulation

Modulus EULA – It's free.



https://developer.nvidia.com/modulus#

Developing digital twins for weather, climate, and energy [S41823]




NVIDIA