ANUGA CDAC

Team Members

Stephen Roberts
ANU

Ole Nielsen

Samir Shaikh
C-DAC, India

Samrit Maity
C-DAC, India

Shweta Das
C-DAC, India

Jayashri Pawar
C-DAC, India

Shashank Sharma
C-DAC, India

Rudi Prihandoko
ANU

Mentors

Bharat Kumar
NVIDIA

Max Rietmann

ANUGA

- About application
  - ANUGA is a Free & Open Source Software (FOSS) package capable of modelling the impact of hydrological disasters such as dam breaks, riverine flooding, storm-surge or tsunamis. ANUGA is based on the Shallow Water Wave Equation discretised to unstructured triangular meshes using a finite-volumes numerical scheme.

- Problem trying to solve
  - Identifying the performance bottleneck of the Legacy ANUGA application
  - Porting ANUGA to GPU architecture

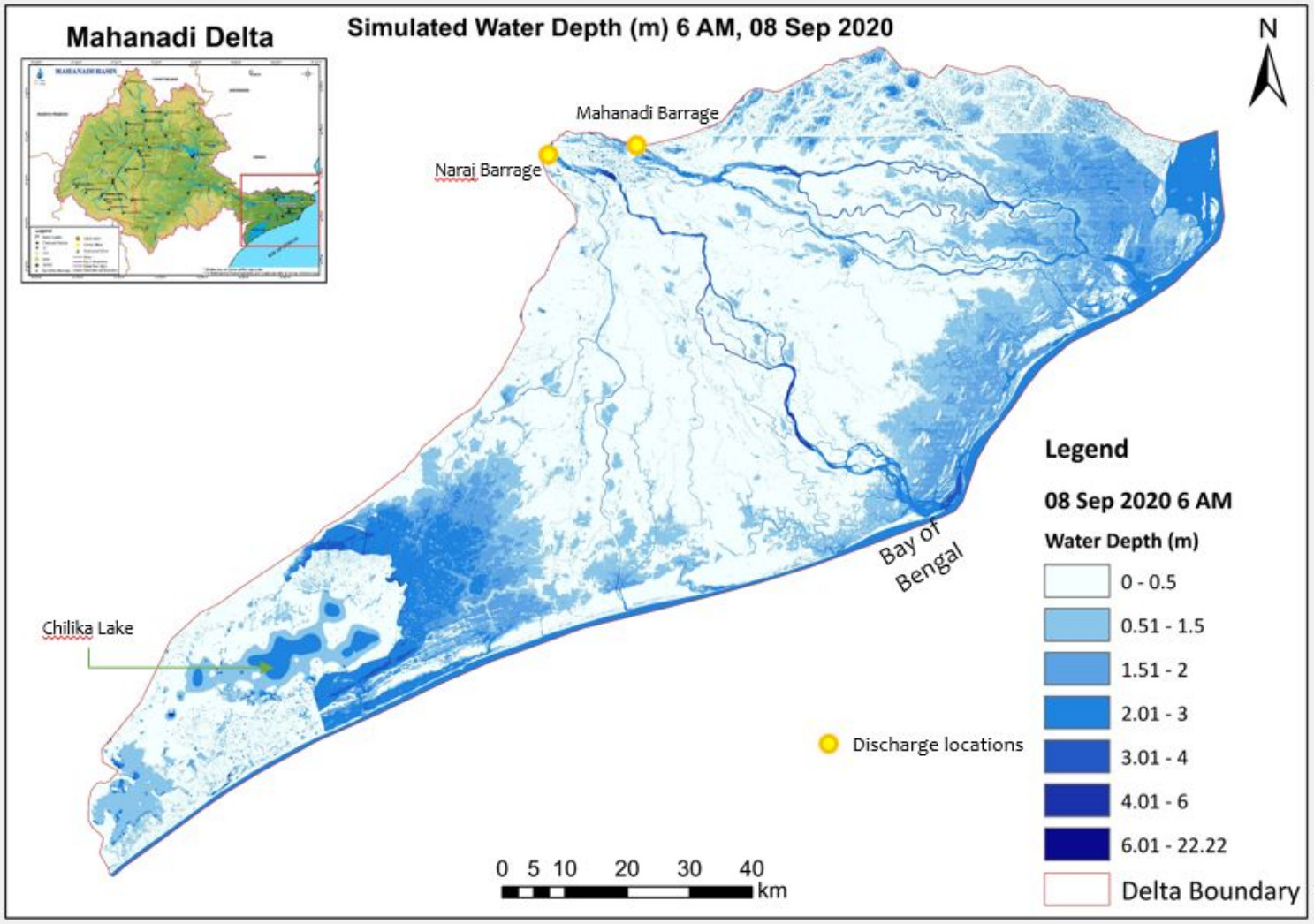
ANUGA

- Scientific driver for the chosen algorithm
  - The end users of ANUGA application needs flood simulation to be performed in less time
  - Theses simulation need to cover larger domain areas, such as full river basin
  - The application execution should take less time and fewer compute resources

- What parts are you focusing on?
  - We profiled the code with Nsight to identify the compute-intensive section of the algorithm.
  - Based on our finding, we worked on two major subroutines
    - Flux calculation subroutine
    - Extrapolation subroutine

ANUGA

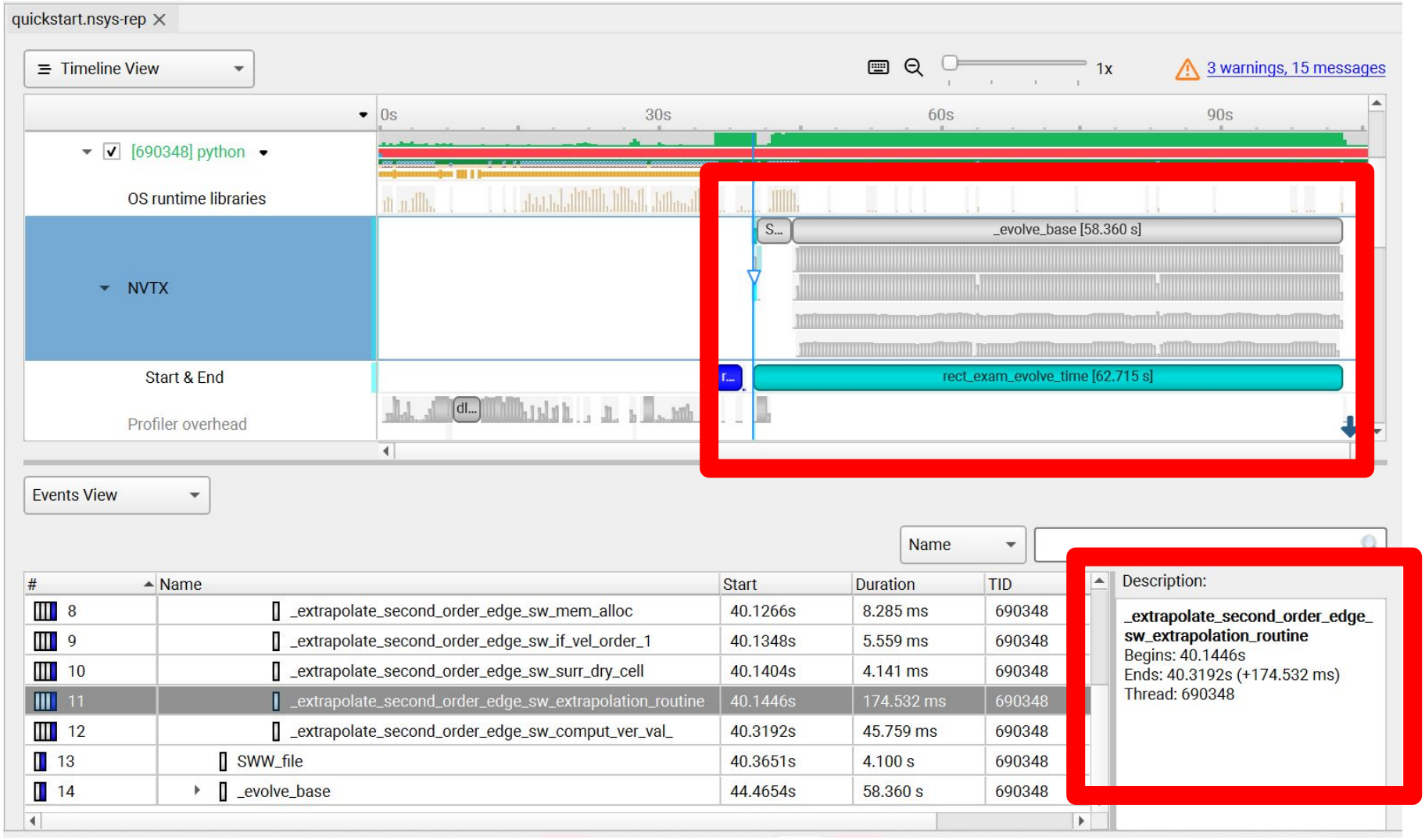
SIMULATION OF DELTA REGION
OF MAHANADI RIVER BASIN



Profiler Output: Before

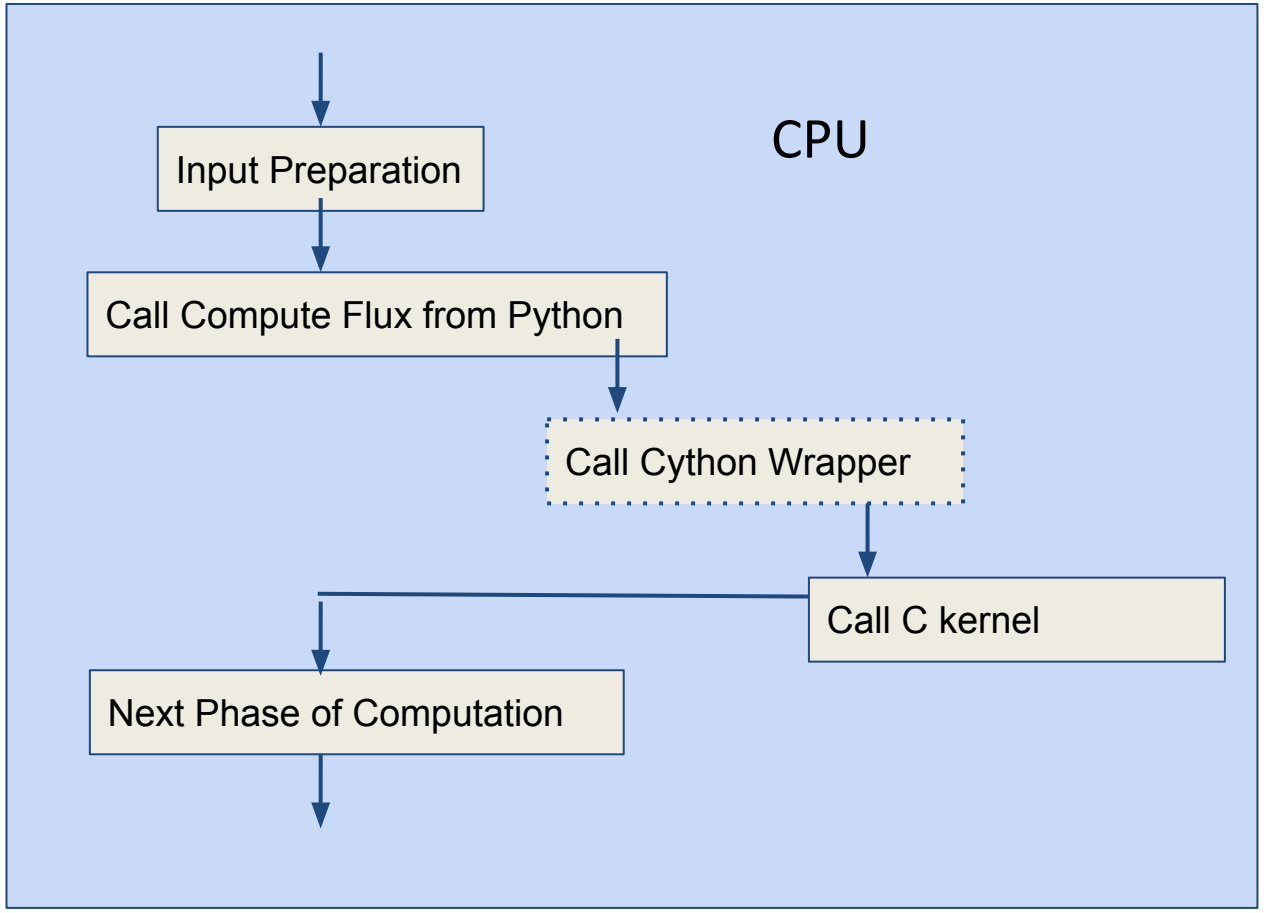
```
[2/4] [====26%                  ] quickstart.sqlite
[2/4] [=======39%               ] quickstart.sqlite
[2/4] [===========52%           ] quickstart.sqlite
[2/4] [===============65%       ] quickstart.sqlite
[2/4] [==================78%    ] quickstart.sqlite
[2/4] [======================91%] quickstart.sqlite
[2/4] [========================100%] quickstart.sqlite
[3/4] Executing 'nvtxsum' stats report

 Time (%)  Total Time (ns)  Instances    Avg (ns)       Med (ns)      Min (ns)     Max (ns)    StdDev (ns)   Style                                Range
 --------  ---------------  ---------  -------------  -------------  -----------  -----------  -----------  --------  ----------------------------------------------------
     18.8      62715084442          1  62715084442.0  62715084442.0  62715084442  62715084442          0.0  StartEnd  rect_exam_evolve_time
     10.7      58359902587          1  58359902587.0  58359902587.0  58359902587  58359902587          0.0  PushPop   _evolve_base
     10.7      33162878002        149    222569651.0    224400904.0    209315261    241548587    6022876.9  PushPop   extrapolate
     10.1      33153794047        149    222508684.9    224342342.0    209252052    241488398    6022411.7  PushPop   _extrapolate_second_order_edge_sw
     10.4      32201808812        144    223623672.3    225467237.0    210794551    242806792    5898630.2  PushPop   distribute_to_vertices_and_edges
      7.0      27915571461        149    187352828.6    189307128.0    173993815    200040136    5688568.9  PushPop   _extrapolate_second_order_edge_sw_extrapolation_routine
      7.0      21699368652        144    150690060.1    149798524.0    148400552    181464682    3697829.6  PushPop   compute_fluxes
      7.0      21687955697        144    150610803.5    149716801.5    148305646    181389463    3698170.0  PushPop   Compute_Fluxes_Central
      1.7       5174395514        288     17966651.1     17776623.5     17512919     23396679     838668.3  PushPop   update_conserved_quantities
      1.3       4100338933          1   4100338933.0   4100338933.0   4100338933   4100338933          0.0  PushPop   SWW_file
      1.2       3797005952        149     25483261.4     24904708.0     24728355     45758794    2379508.4  PushPop   _extrapolate_second_order_edge_sw_comput_ver_val_
      1.0       2947345897          1   2947345897.0   2947345897.0   2947345897   2947345897          0.0  StartEnd  rect_example_creat_time
      0.2        652362427        149      4378271.3      4308527.0      4268130      7389729     362159.3  PushPop   _extrapolate_second_order_edge_sw_if_vel_order_1
      0.2        535748454          4    133937113.5    134562508.5    121602384    145021053    9940621.9  PushPop   store_timestep
      0.1        451937600        149      3033138.3      3007161.0      2903171      4141079     149381.8  PushPop   _extrapolate_second_order_edge_sw_surr_dry_cell
      0.1        325426090        149      2184067.7      2084617.0      2078135      8285004     633222.6  PushPop   _extrapolate_second_order_edge_sw_mem_alloc
      0.1        236989633        144      1645761.3      1607629.0      1585220      2536357     153780.8  PushPop   update_boundary
      0.1        194956946        149      1308435.9      1263787.0      1227534      6960216     467606.4  PushPop   protect_against_infinities
      0.1        194257909        149      1303744.4      1259515.0      1224134      6950903     467233.8  PushPop   protect_new
      0.0        141477230        144       982480.8       951819.5       932935      2300597     192734.1  PushPop   compute_forcing_terms
      0.0          1485506        144        10316.0        9809.5         8543        29234       2154.4  PushPop   update_timestep
      0.0           106186        144          737.4         586.0          451         5105        676.6  PushPop   compute_flux_update_frequency
      0.0            13705          1        13705.0       13705.0        13705        13705          0.0  StartEnd  rectangular_exam_domain_distr

[4/4] Executing 'osrtsum' stats report
```

Profiler Output: Before



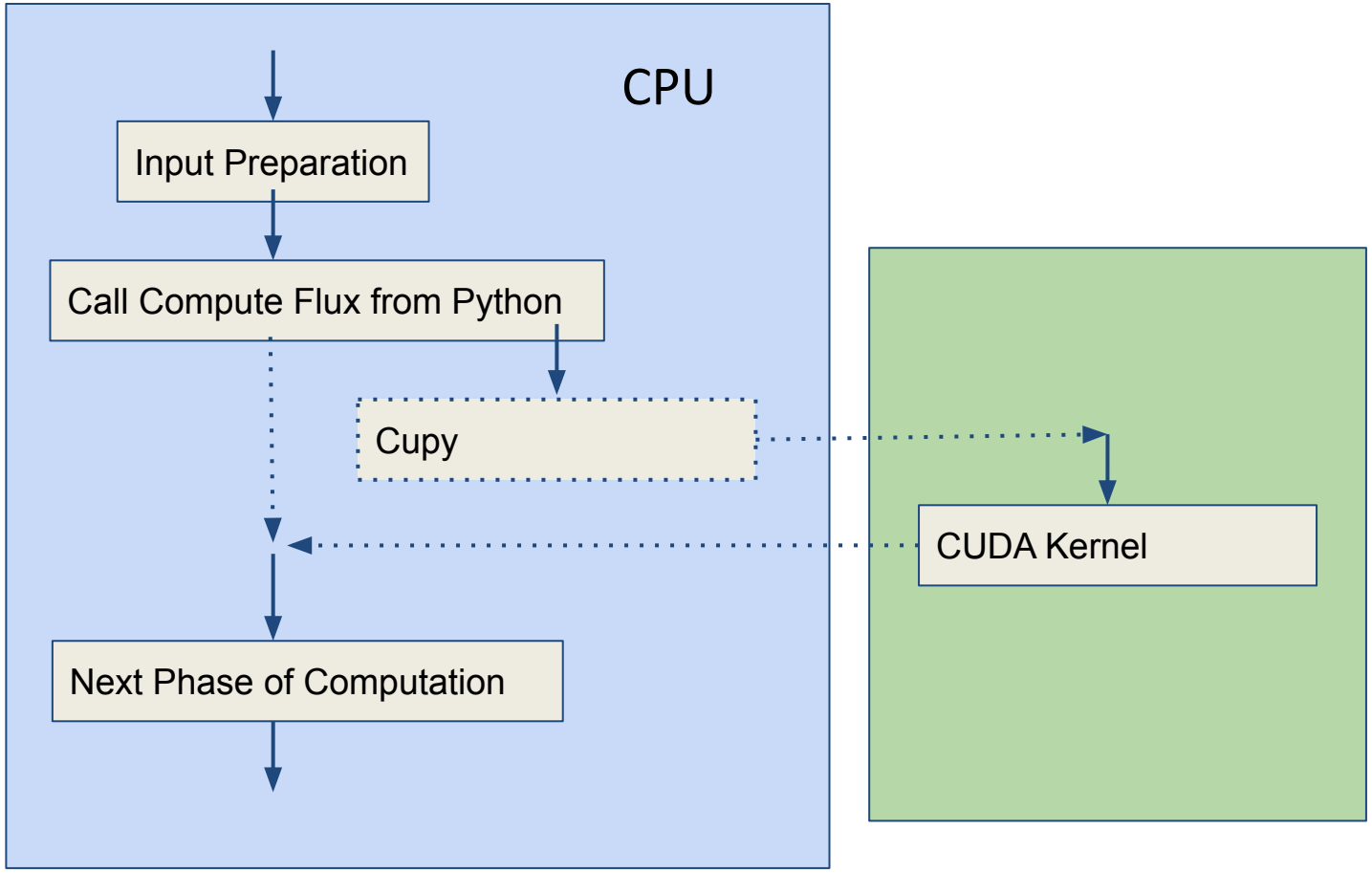
ANUGA Structure

Legacy Structure



ANUGA Structure

Proposed Compute Pattern



Evolution and Strategy

- What was your initial strategy?
  - Attempted parallel strategy for porting the compute-intensive part using both CuPy and OpenACC
- How did this strategy change?
  - We realized that due to time constraints, it would be more feasible to target a single key kernel and port it to both CuPY and OpenACC

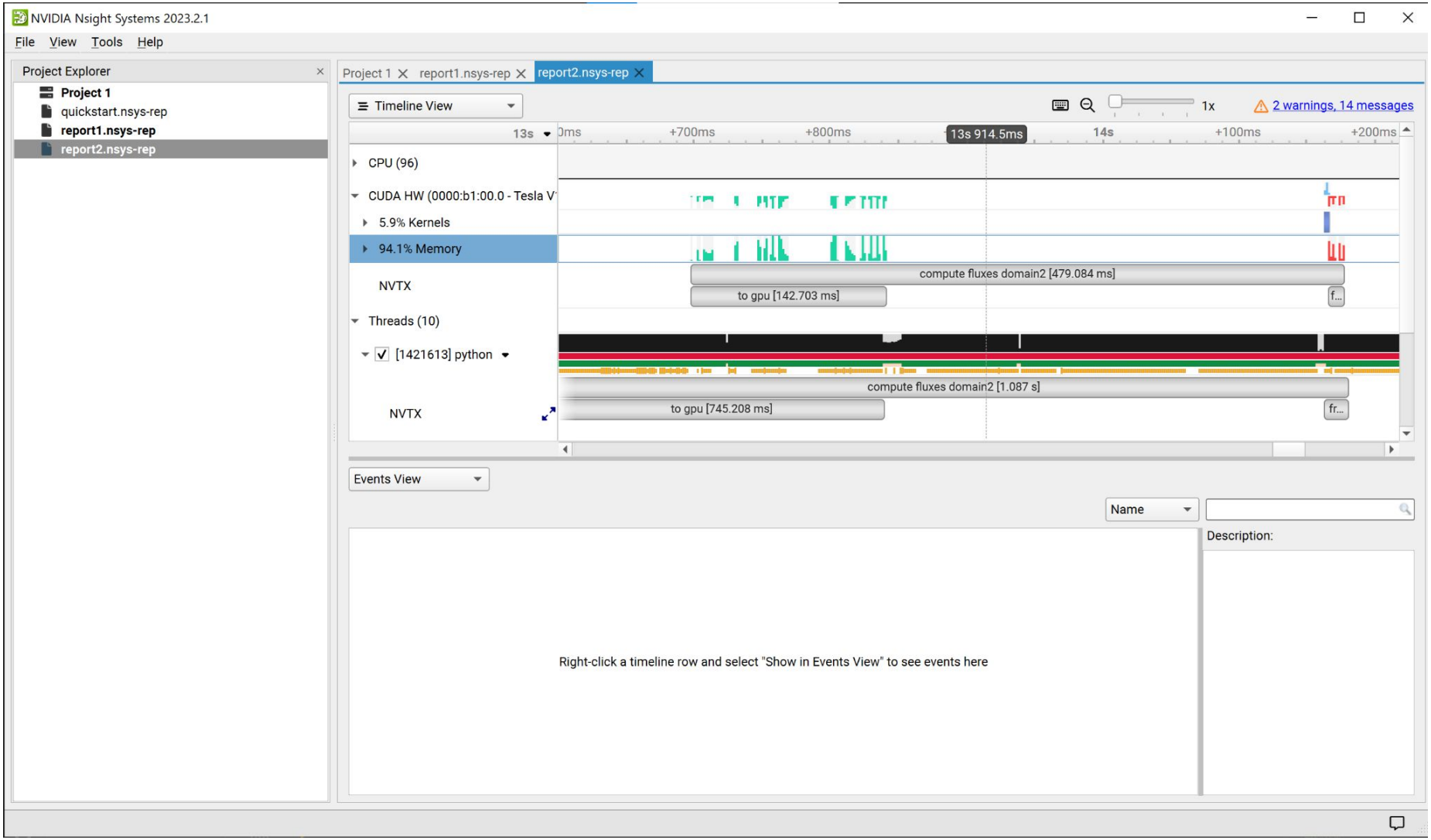
```
import cupy as cp

nvtxRangePush('to gpu')

gpu_local_timestep        = cp.array(local_timestep)          #InOut
gpu_boundary_flux_sum     = cp.array(boundary_flux_sum )      #InOut
gpu_max_speed             = cp.array(max_speed)               #InOut
gpu_stage_explicit_update = cp.array(stage_explicit_update)   #InOut
gpu_xmom_explicit_update  = cp.array(xmom_explicit_update)    #InOut
gpu_ymom_explicit_update  = cp.array(ymom_explicit_update)    #InOut


kernel( (NO_OF_BLOCKS, 0, 0), \
                          (THREADS_PER_BLOCK, 0, 0), \
                           ( \
                             gpu_local_timestep, \
                             gpu_boundary_flux_sum, \
                             gpu_max_speed, \
                             gpu_stage_explicit_update,\
                             gpu_xmom_explicit_update,\
                             gpu_ymom_explicit_update,\
                             gpu_stage_centroid_values,\
                             gpu_stage_edge_values,\
                             gpu_xmom_edge_values, \
                             gpu_ymom_edge_values,\
```

Results and Final Profile

- What were you able to accomplish?
  - Refactoring, Refactoring and Refactoring
  - Able to integrate CuPy into top kernel for flux calculation subroutine
  - Kernel for extrapolate subroutine is ready to be integrated with ANUGA
  - Did you achieve speed up?
    - Computer kernel execution time - 135 Milisec on CPU, 2.3 Milisec on GPU (more than 60x speed up)
    - Datatransfer back and forth to GPU should be looked into.

- What did you learn?
  - Application profiling
    - how to applying nvtx marker
    - how analyse output on Nsight tool and identify performance bottleneck section of code.
    - Plugging Cupy with existing Python based application.
    - CUDA architecture know-how

Results and Final Profile



What problems have you encountered?

- Problems with legacy app structure
  - Legacy code having complex structure
  - Python - Cython - C function call
- Tool lack of features:
  - Better documentation for integrating “profiling markers” into complex code structure (Python -> Cython -> C).
  - There is no AtomicMin (AtomicMax) for double in CUDA programming
- System setup:
  - Suggesting to have important packages like CuPy and GDAL as modules or recipes in conda environment on GADI System

Wishlist

- What do you wish existed to make your life easier?
  - Event - Thank you all for perfect execution of event.
  - Tools
    - Need power supply (and/or extension) that can accommodate participants, especially availability of “international power adapter”
  - Systems
    - Having a login node that could do computation for small scale testing
  - Standard practice or guideline for the process of converting C code to CUDA code
  - A detailed tutorial that covers the process of profiling a scientific Python application using NVIDIA tools on an HPC (High-Performance Computing) facilitation

Was it worth it?

- Was this worth it? Yes, of course

- Will you continue development?
  - Certainly. We will continue porting exercise till goal is achieved and end users requirements are met.
  - Working on OpenACC
  - Try to optimize further

- What sustained resources/support will be critical for your work after the event?
  - Wish to continue working with mentors
  - Prolonged access to GADI system with GPU Resource

Team's achievements during this Hackathon

Refactoring, Refactoring, Refactoring. But where?

- We performed profiling of the ANUGA code and identified certain parts that take up a significant amount of time and require optimization.
- Delve deeper into the profiling, with NVIDIA Nsight System and examined the innermost loop of the program.
- We had to prioritize which part was the most critical and could be ported to CuPy and OpenACC within the given time frame of the hackathon.
- Successfully ported the flux calculation subroutine within the main loop of ANUGA to be compatible with CuPy and OpenACC.

PROMOTING YOUR WORK: AVAILABLE OPPORTUNITIES

- **Papers and Talks:** Please acknowledge the Open Hackathons program and OpenACC Organization in any planned or upcoming papers, presentations, or talks.

  "This work was completed in part at the NCI Open Hackathon, part of the Open Hackathons program. The authors would like to acknowledge OpenACC-Standard.org for their support."
- **Social Media Support:** Please feel free to promote your participation across your social media channels. Tag **@OpenACCorg** and **#OpenHackathons** and we are happy to amplify.
- **Blogs and Technical Write-ups**: Create a blog post or technical article that highlights the work being done and results achieved.
- **Quotes and Testimonials:** Highlight your quote or feedback on our channels (i.e. social, website, etc.).

***Please reach out to Izumi Barker (ibarker@nvidia.com) to discuss marketing options and opportunities.**