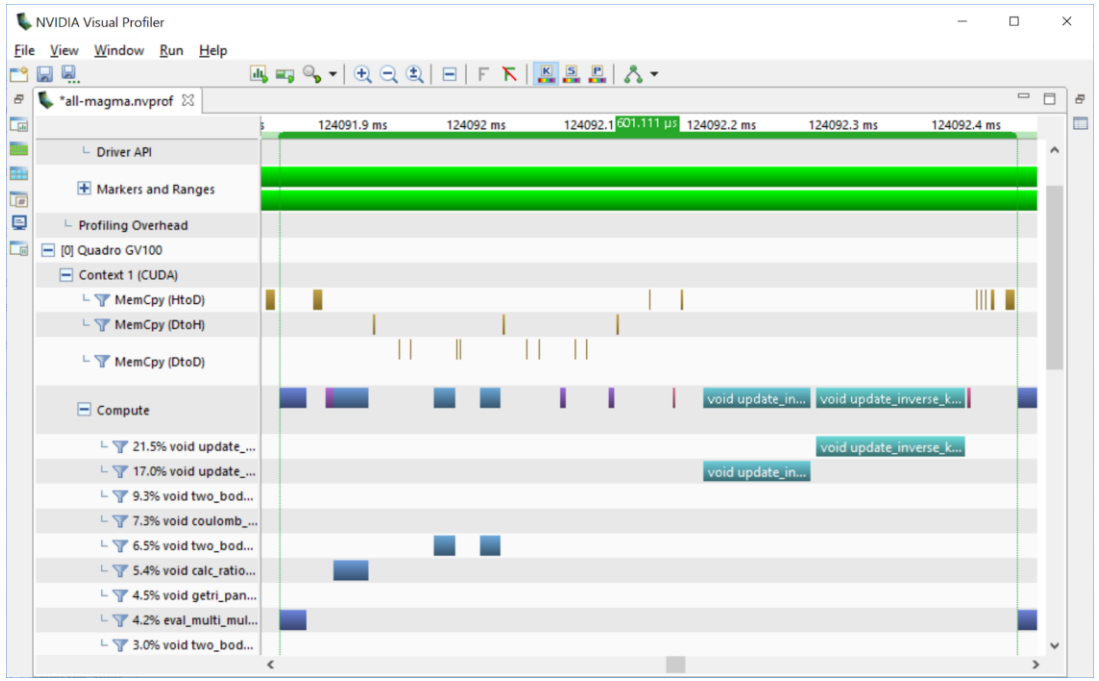
Project Introduction

- Simulation broken into 3 phases, Variadic Monte Carlo (VMC) with and without drift & Diffusion Monte Carlo (DMC)
- Each phase has similar structure with lots of very small kernels and launch latencies.
- Goal: Move a portion of these to CUDA graphs to reduce launch latencies & increase concurrency



CUDA 10.0 Graph Capture Pattern

```
cudaGraphExec_t gexec = NULL;
char graph_error[1024];
cudaError_t cuda_error;
for (int isub = 0; isub < nSubSteps; isub++)
  for (int iat = 0; iat < nat; ++iat)     {
    cudaGraph_t graph = NULL;
    cuda_error = cudaStreamBeginCapture(gpu::kernelStream, cudaStreamCaptureModeRelaxed);
    Psi.ratio(W, curr_iat, ratios_d, newG_d, newL_d);
    cuda_error = cudaStreamEndCapture(gpu::kernelStream, &graph)
    cudaGraphNode_t error_node;
    if ( (gexec == NULL) || (updateResult != cudaGraphExecUpdateSuccess) ) {
      cuda_error = cudaGetLastError(); // Clear last error
      if ( gexec != NULL ) cudaGraphExecDestroy(gexec);
      cudaGraphNode_t error_node;
      cuda_error = cudaGraphInstantiate(&gexec,graph,&error_node,graph_error,1024);
    }
    cuda_error = cudaGraphLaunch(gexec,gpu::memoryStream);
    cuda_error = cudaEventRecord(gpu::ratioSyncDiracEvent, gpu::memoryStream);
  }
}
```

Capture *every* step.

Destroy previous executor

Instantiate *every* step

Launch *every* step.

* Extensive error checking removed for space.

CUDA 10.2 Graph Update Pattern

```
cudaGraphExec_t gexec = NULL;
char graph_error[1024];
cudaError_t cuda_error;
for (int isub = 0; isub < nSubSteps; isub++)
  for (int iat = 0; iat < nat; ++iat)     {
    cudaGraph_t graph = NULL;
    cuda_error = cudaStreamBeginCapture(gpu::kernelStream, cudaStreamCaptureModeRelaxed);
    Psi.ratio(W, curr_iat, ratios_d, newG_d, newL_d);
    cuda_error = cudaStreamEndCapture(gpu::kernelStream, &graph);
    cudaGraphNode_t error_node;
    cudaGraphExecUpdateResult updateResult;
    if ( gexec != NULL ) cuda_error = cudaGraphExecUpdate(gexec,graph,&error_node,&updateResult);
    if ( (gexec == NULL) || (updateResult != cudaGraphExecUpdateSuccess) ) {
      cuda_error = cudaGetLastError(); // Clear last error
      if ( gexec != NULL ) cudaGraphExecDestroy(gexec);
      cudaGraphNode_t error_node;
      cuda_error = cudaGraphInstantiate(&gexec,graph,&error_node,graph_error,1024);
    }
    cuda_error = cudaGraphLaunch(gexec,gpu::memoryStream);
    cuda_error = cudaEventRecord(gpu::ratioSyncDiracEvent, gpu::memoryStream);
  }
}
```

Capture *every* step.

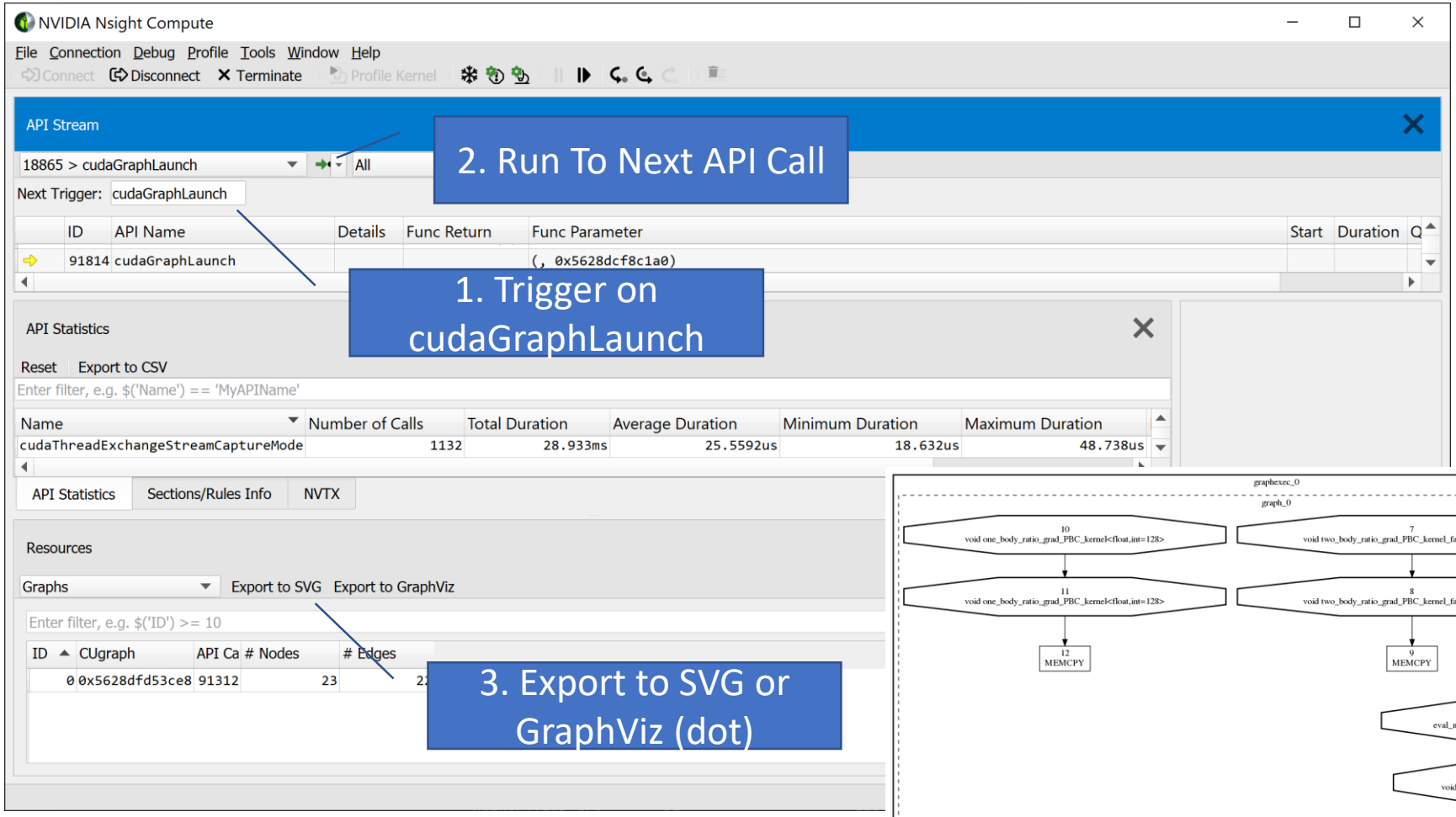
Attempt to update & instantiate as necessary

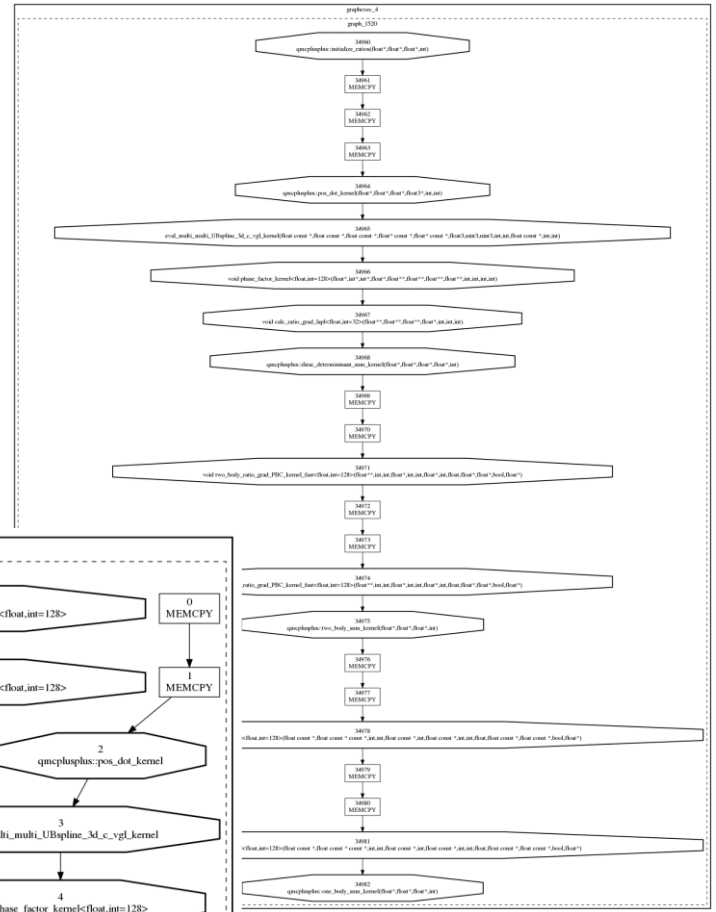
Instantiate only when necessary

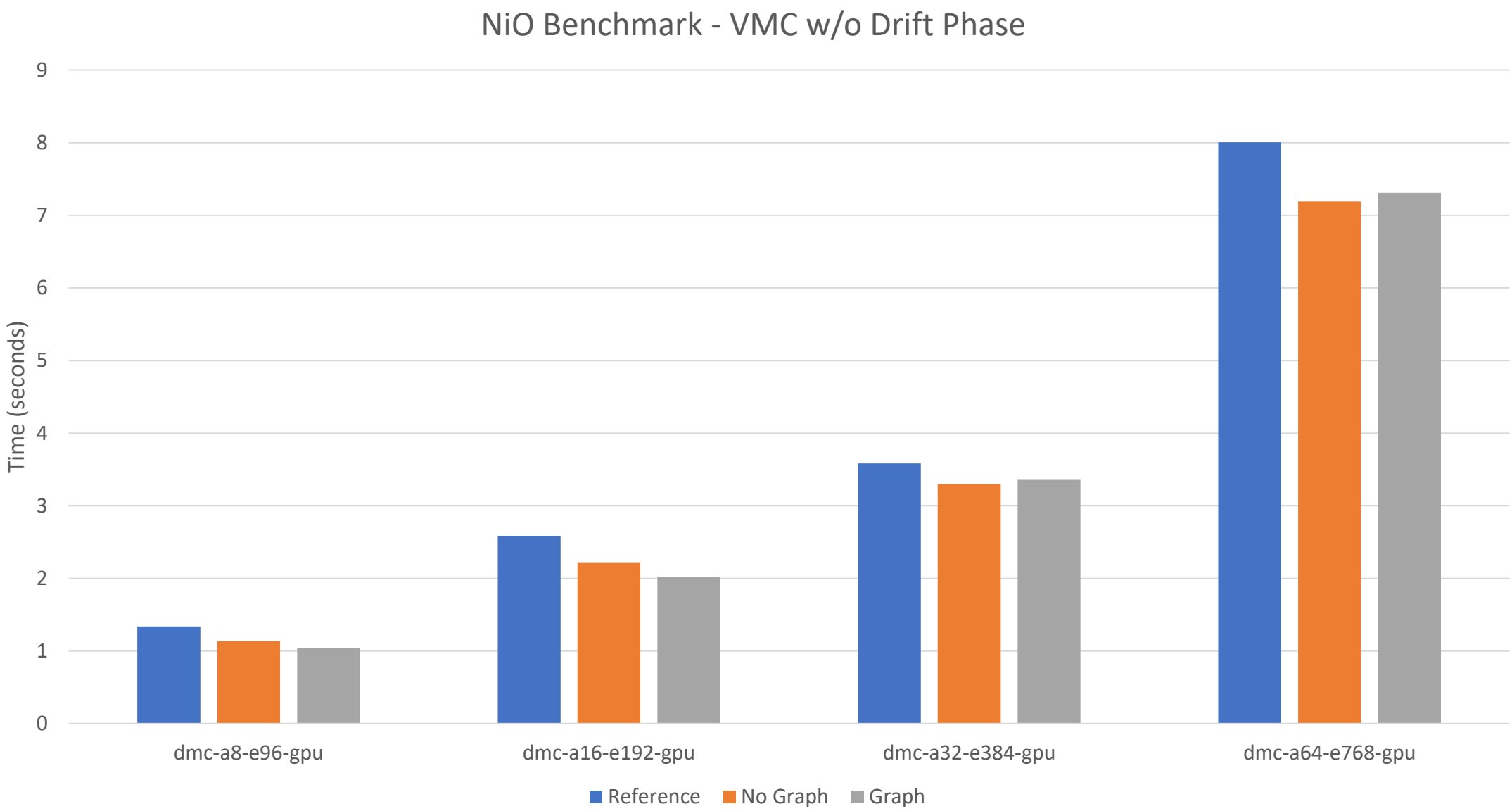
Launch *every* step.

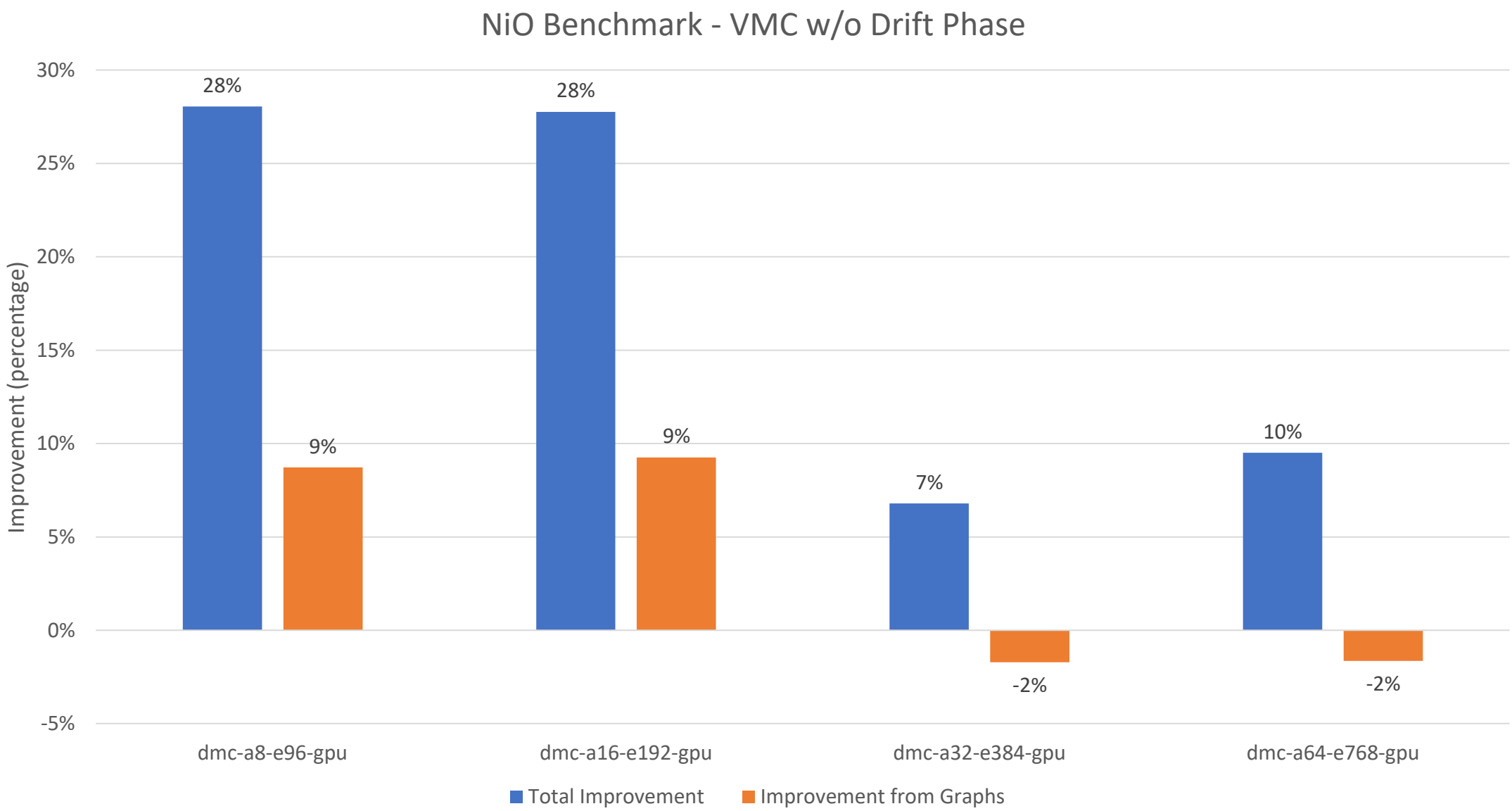
* Extensive error checking removed for space.

Nsight Compute Graph Visualization






NiO Benchmark - VMC w/o Drift Phase
Time (seconds)
0
1
2
3
4
5
6
7
8
9
dmc-a8-e96-gpu
dmc-a16-e192-gpu
dmc-a32-e384-gpu
dmc-a64-e768-gpu
Reference
No Graph
Graph


NiO Benchmark - VMC w/o Drift Phase
Improvement (percentage)
30%
25%
20%
15%
10%
5%
0%
-5%
28%
9%
28%
9%
7%
-2%
10%
-2%
dmc-a8-e96-gpu
dmc-a16-e192-gpu
dmc-a32-e384-gpu
dmc-a64-e768-gpu
Total Improvement
Improvement from Graphs

Lessons I’ve Learned (So Far)

- Graph Update makes CUDA Graphs *significantly* simpler
  - Didn’t necessitate moving as much from the CPU for successful capture.
  - Only instantiates when a structural change has occurred
  - NOTE: I couldn’t always tell what changed via Nsight graph
- Application had WAY more reliance on default stream than I expected
  - Avoid the default stream, expose as much concurrency as possible.
- Error check OFTEN
  - Errors during capture crop up much later, cudaGetLastError is your friend
  - Not returning cudaSuccess shouldn’t always cause an abort