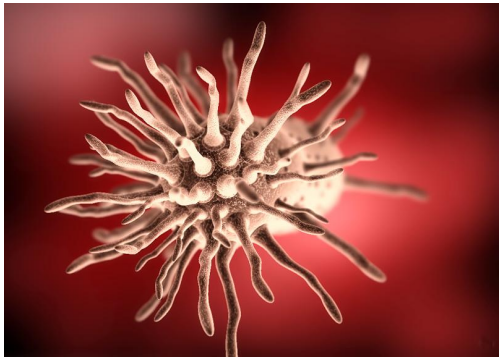
Bioinformatics: A Case Study

NERSC GPUs for Science Day

Bryce Foster

2019-07-02

DOE Joint Genome Institute



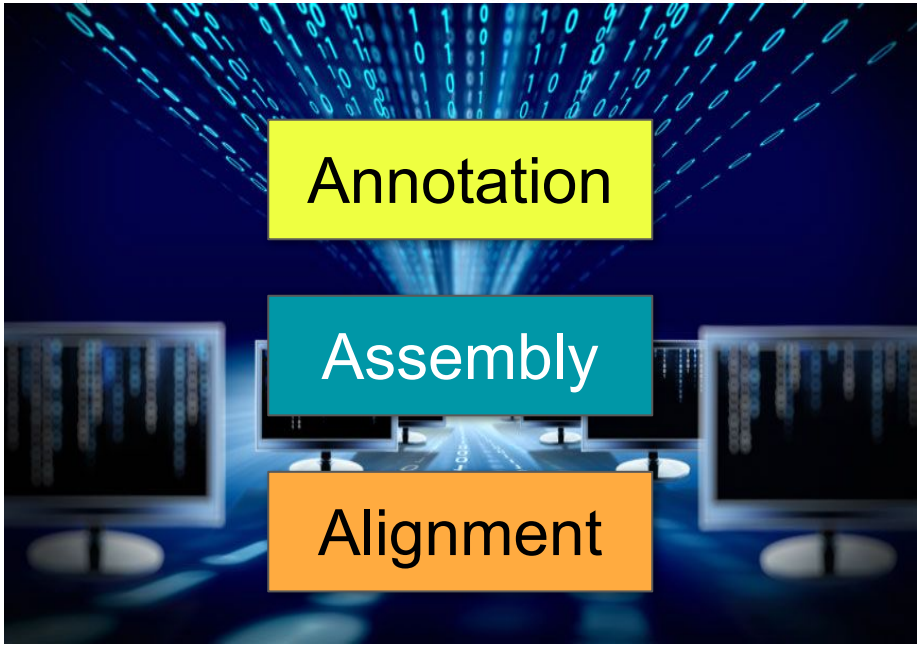
JGI
JOINT GENOME INSTITUTE

ATCGCTCGACTATCGACGAT
GTCACGATGTGCACGATC
TCGACTGATCGATGTGCAT
CATACGTAGCCGTACTCGA
TATGCGACTGCACTATACG
ATCGTAGCTGACTGCACTA
ACTGCACTATGCTACTGCA
AGCCATGCG TATGC CAT
GACTCGTCATCTGACTCG
TACGTAGCCGTACTCGACA

NERSC

Bioinformatics Analysis Software

ATCGCTCGACTATCGACGAT
GTCACGATGTGCACGATC
TCGACTGATCGATGTGCAT



- **Typical bioinformatics algorithm**
  - Read large text file from disk
    - Megabytes to 100 gigabytes+
  - Process data on CPUs and in memory
    - Take advantage of multi-CPUs
  - Write smaller files back to disk

- **Thousands of 3rd party tools**

Java

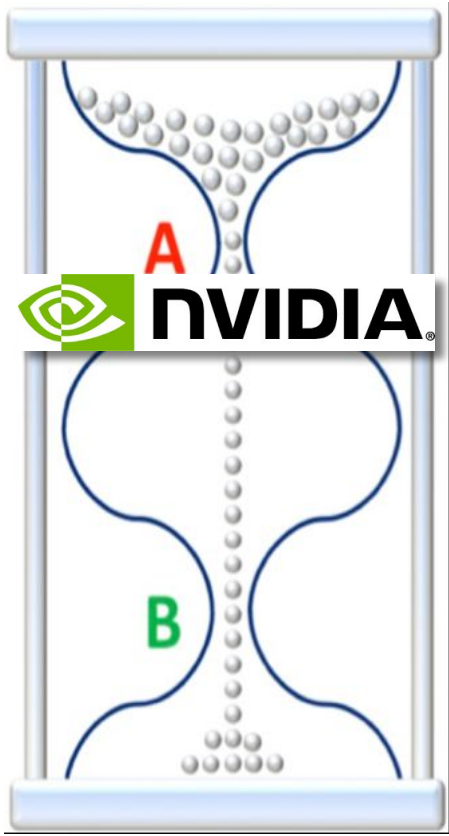
C++

python

PL

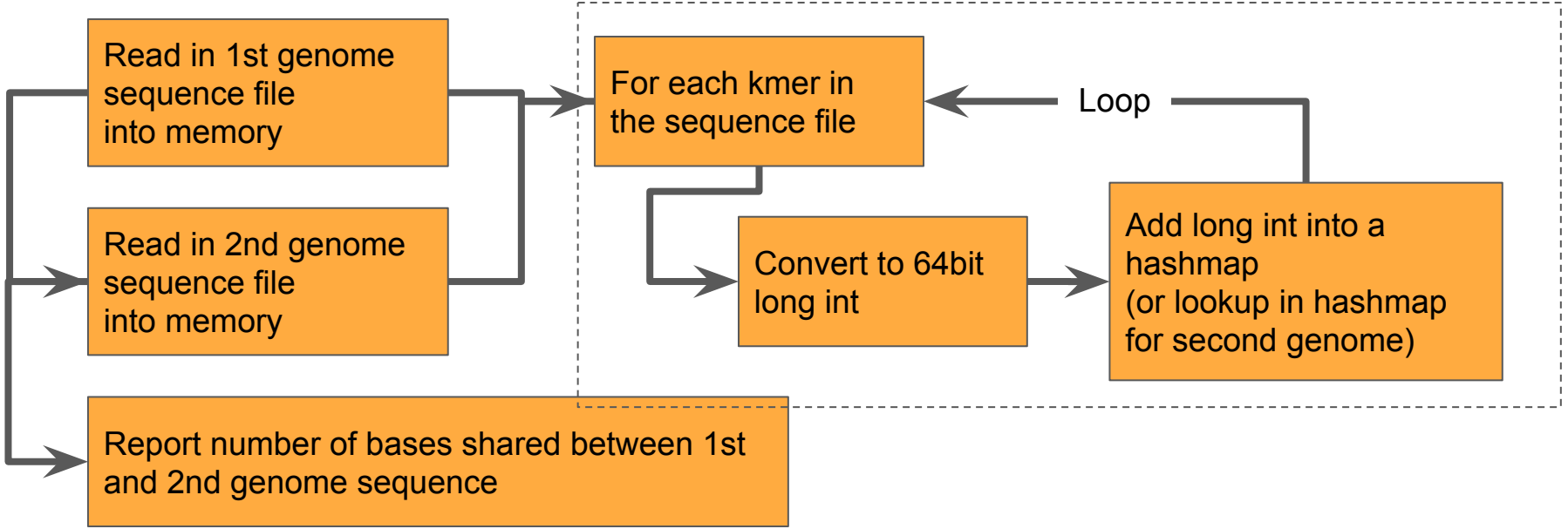
Our Strategy

- Convert an algorithm from BBTools (Java) into C++
- Compile code with GNU C++ compiler
- Get code to compile with NVidia's compiler
- Check the answers
- Profile code
  - Where are the bottlenecks
  - Is it taking advantage of Nvidia hardware?
- Add OpenACC pragma statements (NVidia lib)
- Compare accelerated versus non-accelerated runtimes



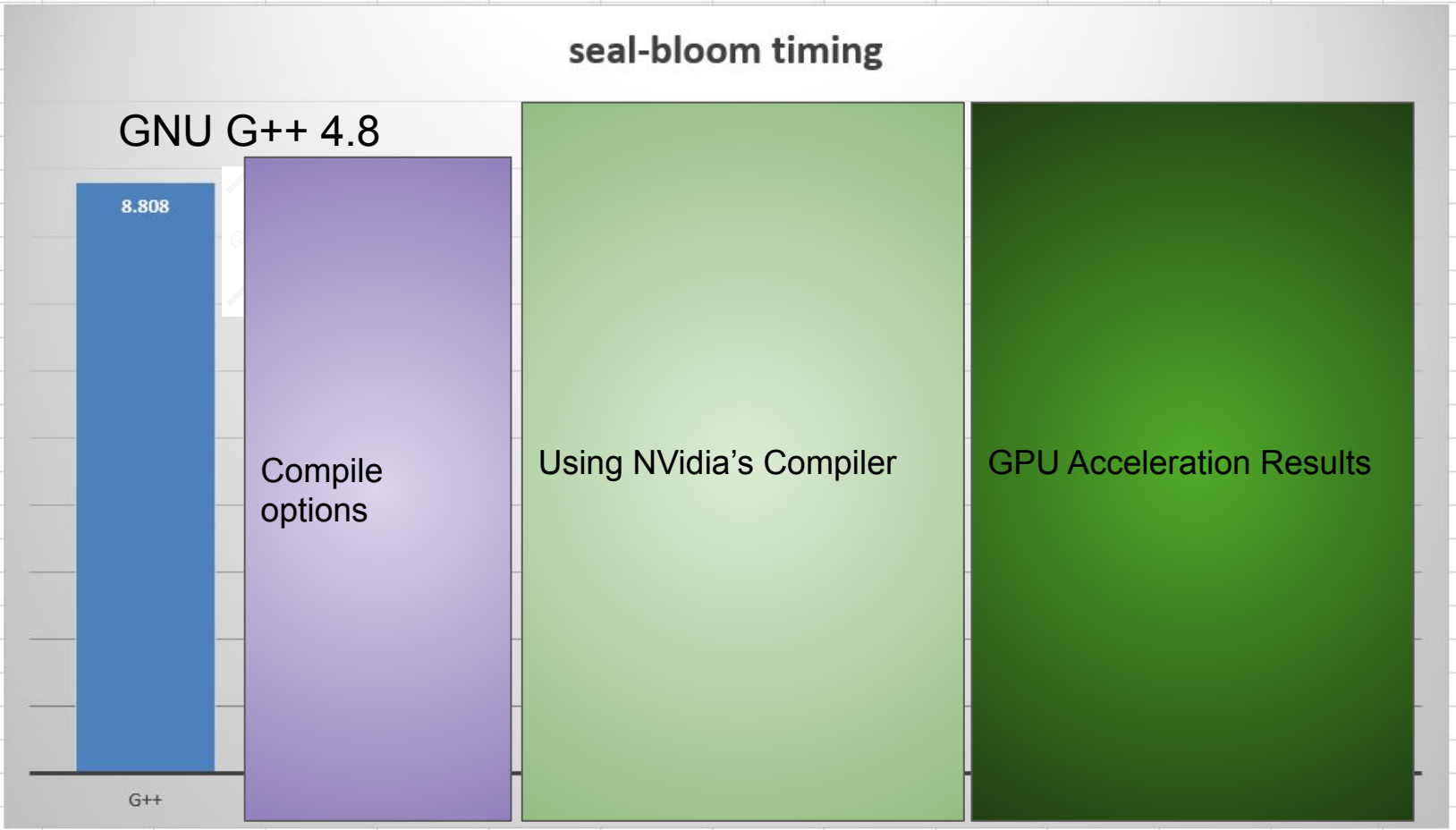
Our Code: Seal (C++)

Seal is an alignment algorithm to compare the similarity between 2 genomic sequences

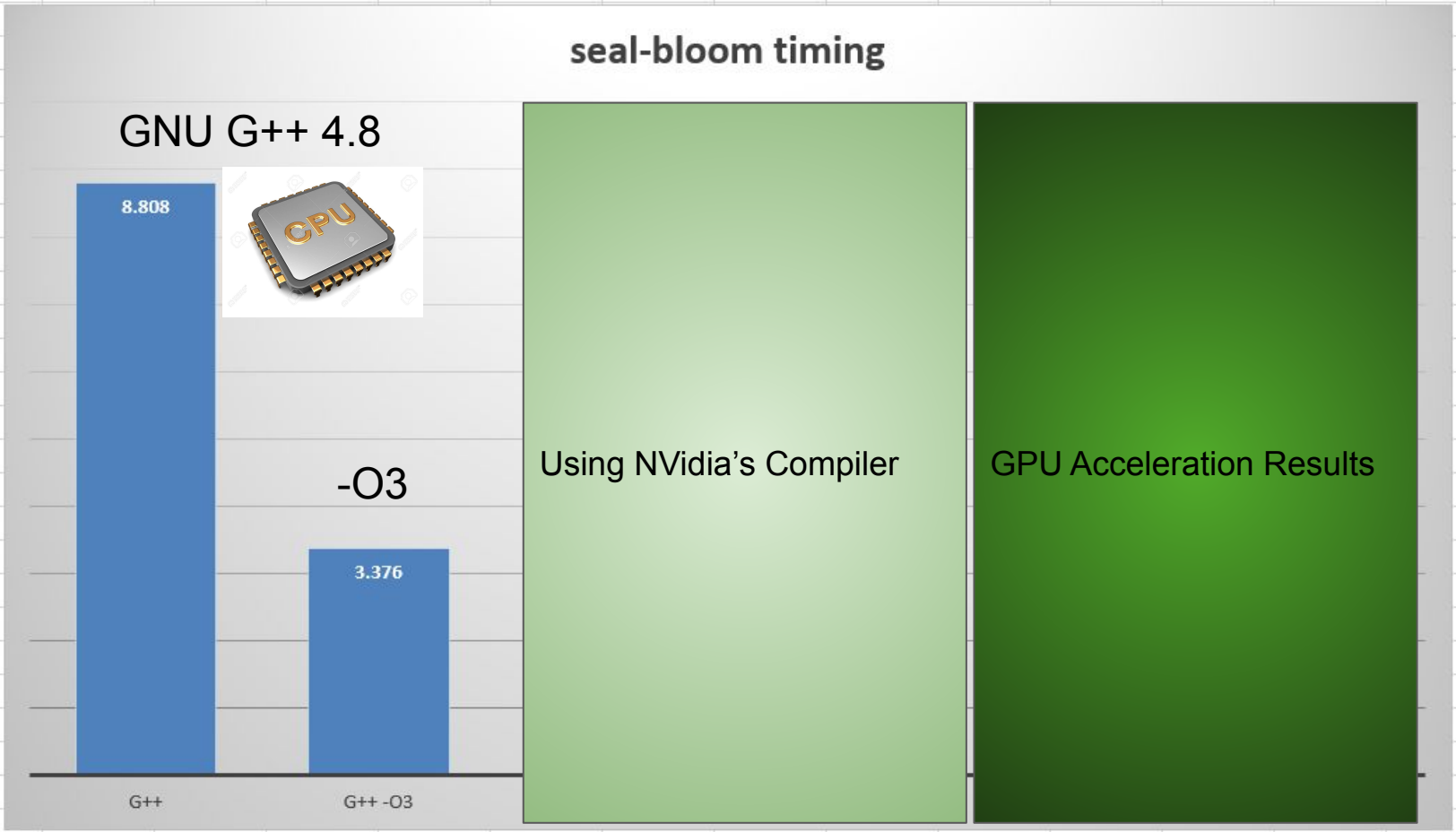



```
Sequence = “...GTTGCATGCAATCCGCGCGCAAGAACTGGTTCTGGGGCAACGCAGGTCTATCTGTC…”
kmer = "CCGCGCGCAAGAACTGGTTCTGGGGCAACGC";
Binary = 0101100110011001000010000001111010111011110101010010000011001
Long Int = 1614050117235155993
```

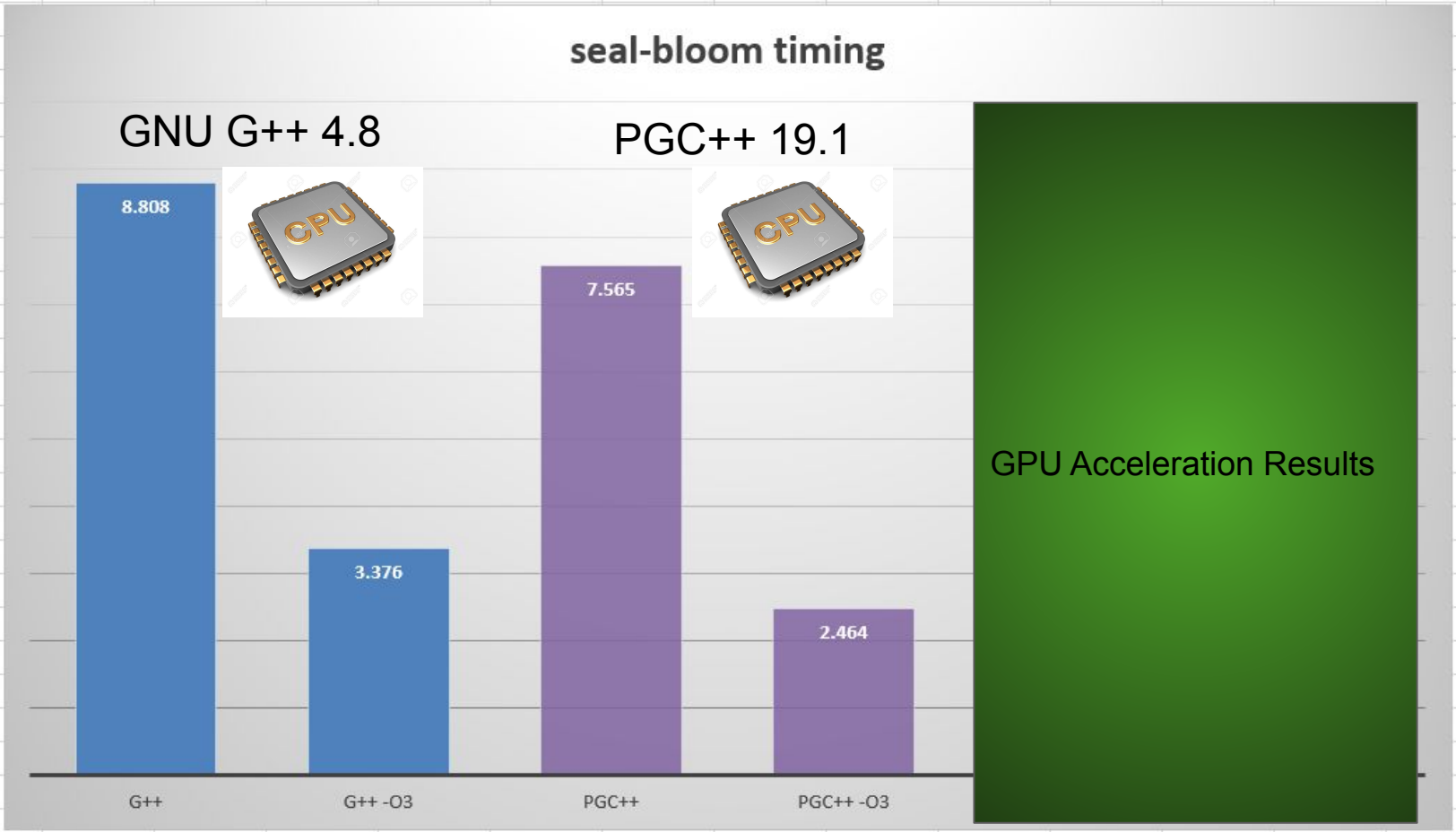
Our Results for Seal



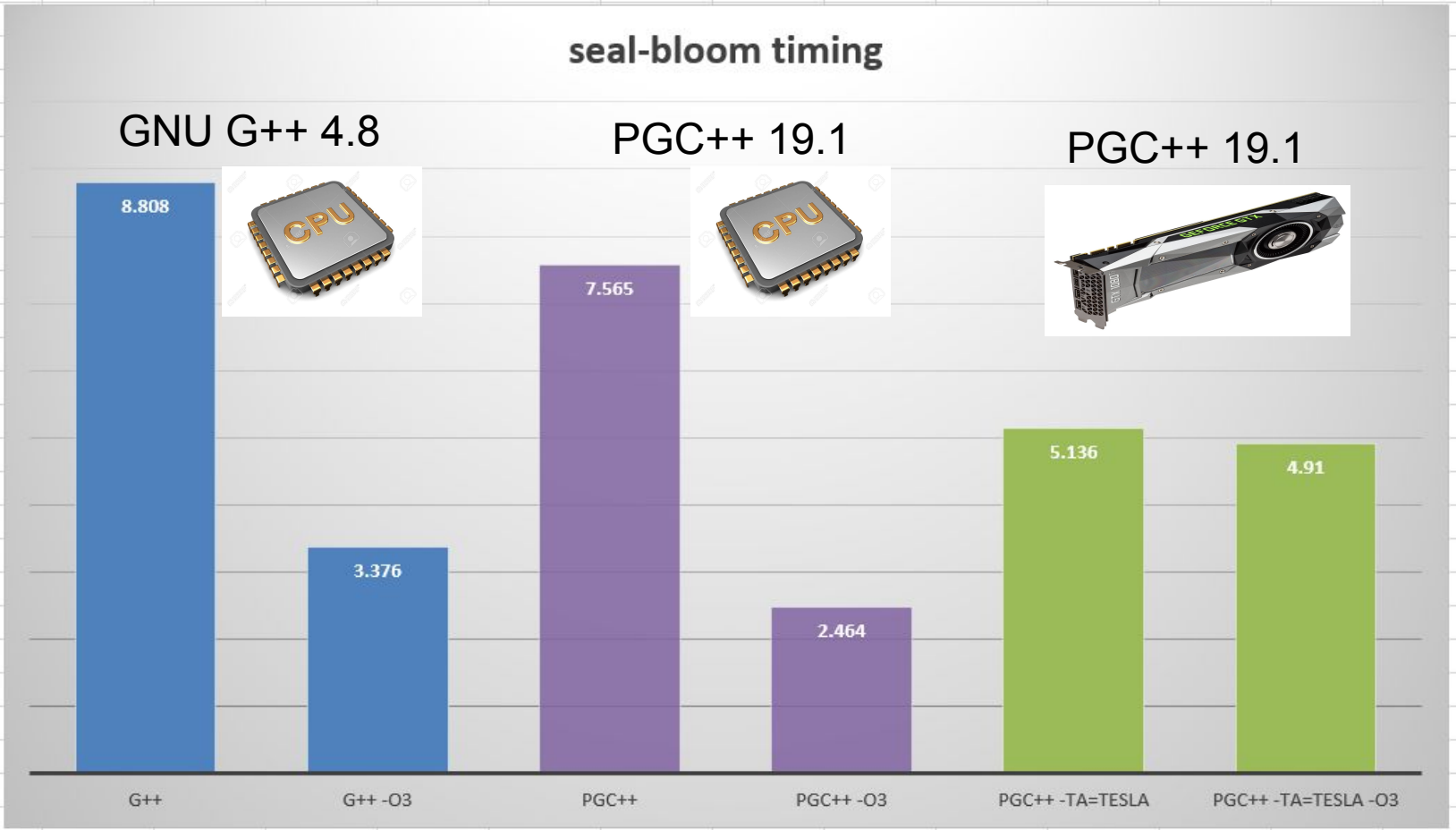
Our Results for Seal

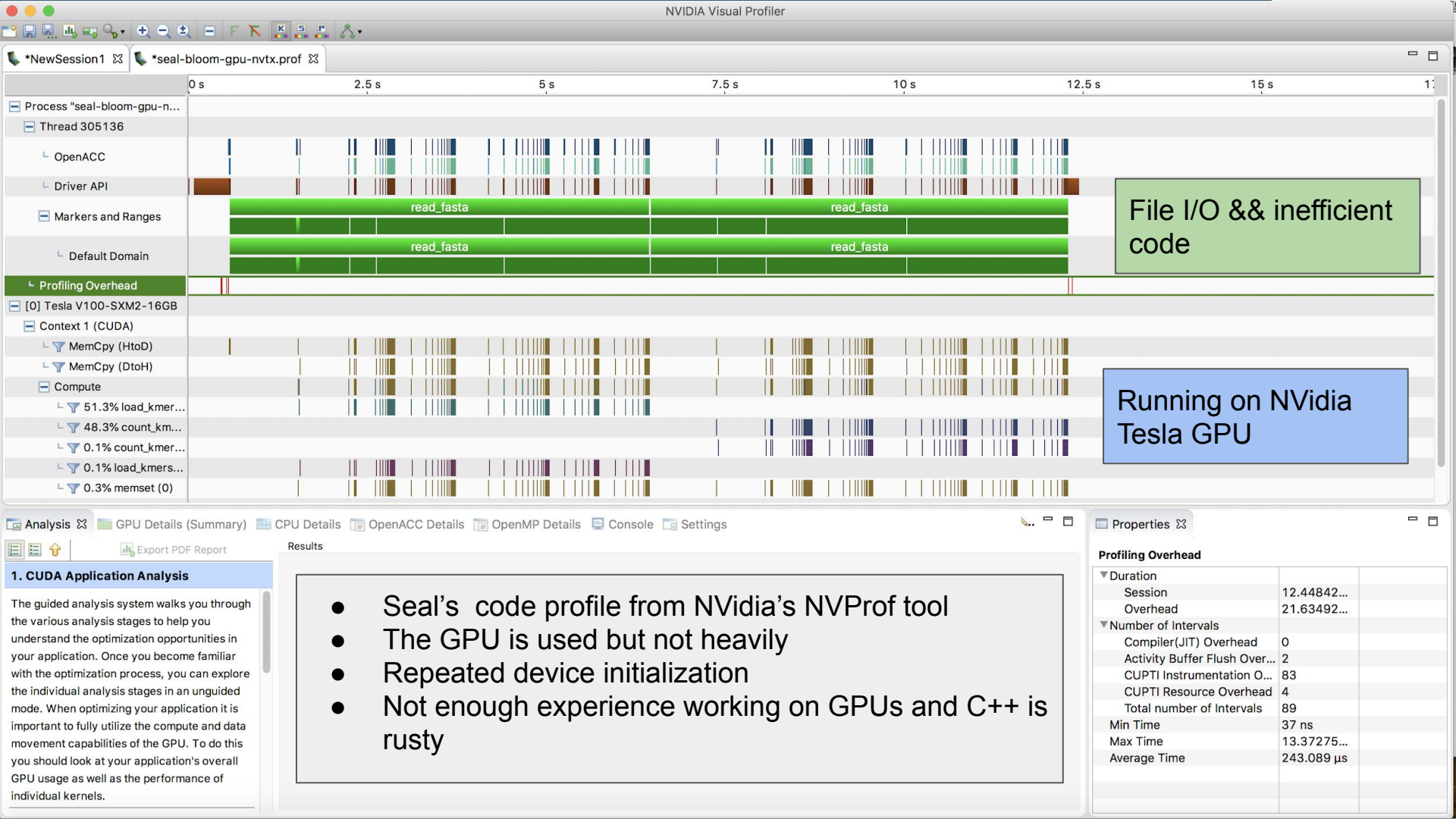


Our Results for Seal



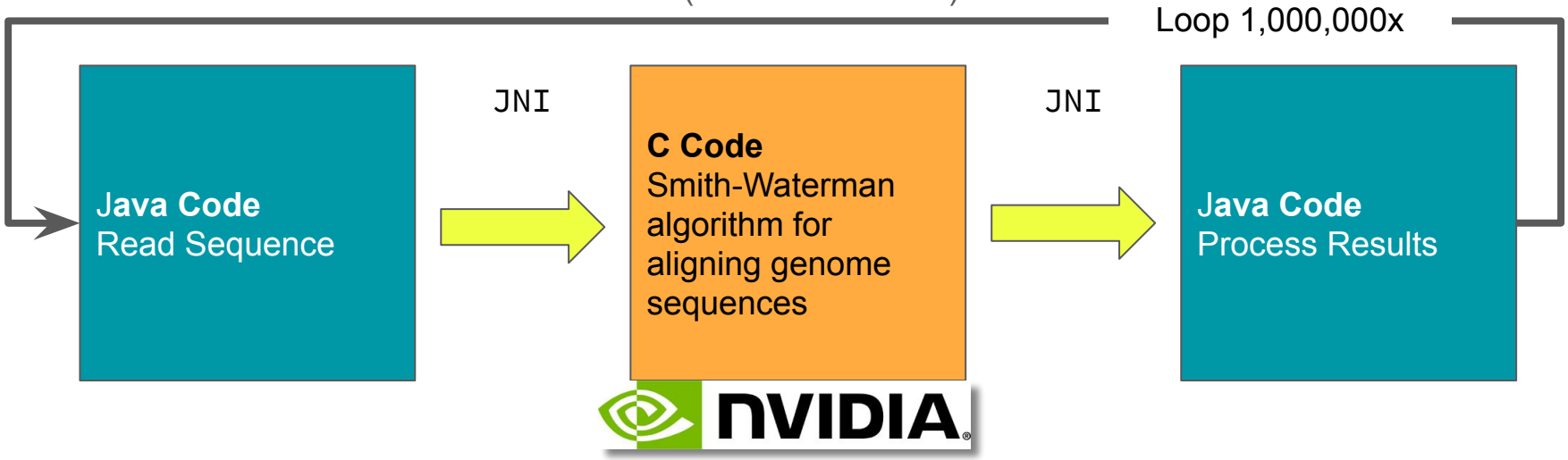
Our Results for Seal




NVIDIA Visual Profiler
*NewSession1
*seal-bloom-gpu-nvtx.prof
0 s
2.5 s
5 s
7.5 s
10 s
12.5 s
15 s
Process "seal-bloom-gpu-n...
Thread 305136
OpenACC
Driver API
Markers and Ranges
read_fasta
read_fasta
Default Domain
read_fasta
read_fasta
Profiling Overhead
[0] Tesla V100-SXM2-16GB
Context 1 (CUDA)
MemCpy (HtoD)
MemCpy (DtoH)
Compute
51.3% load_kmer...
48.3% count_km...
0.1% count_kmer...
0.1% load_kmers...
0.3% memset (0)
File I/O && inefficient code
Running on NVidia Tesla GPU
Analysis
GPU Details (Summary)
CPU Details
OpenACC Details
OpenMP Details
Console
Settings
Export PDF Report
1. CUDA Application Analysis
The guided analysis system walks you through the various analysis stages to help you understand the optimization opportunities in your application. Once you become familiar with the optimization process, you can explore the individual analysis stages in an unguided mode. When optimizing your application it is important to fully utilize the compute and data movement capabilities of the GPU. To do this you should look at your application's overall GPU usage as well as the performance of individual kernels.
Results
Seal's code profile from NVidia's NVProf tool
The GPU is used but not heavily
Repeated device initialization
Not enough experience working on GPUs and C++ is rusty
Properties
Profiling Overhead
Duration
Session 12.44842...
Overhead 21.63492...
Number of Intervals
Compiler(JIT) Overhead 0
Activity Buffer Flush Over... 2
CUPTI Instrumentation O... 83
CUPTI Resource Overhead 4
Total number of Intervals 89
Min Time 37 ns
Max Time 13.37275...
Average Time 243.089 µs

JNI Optimization

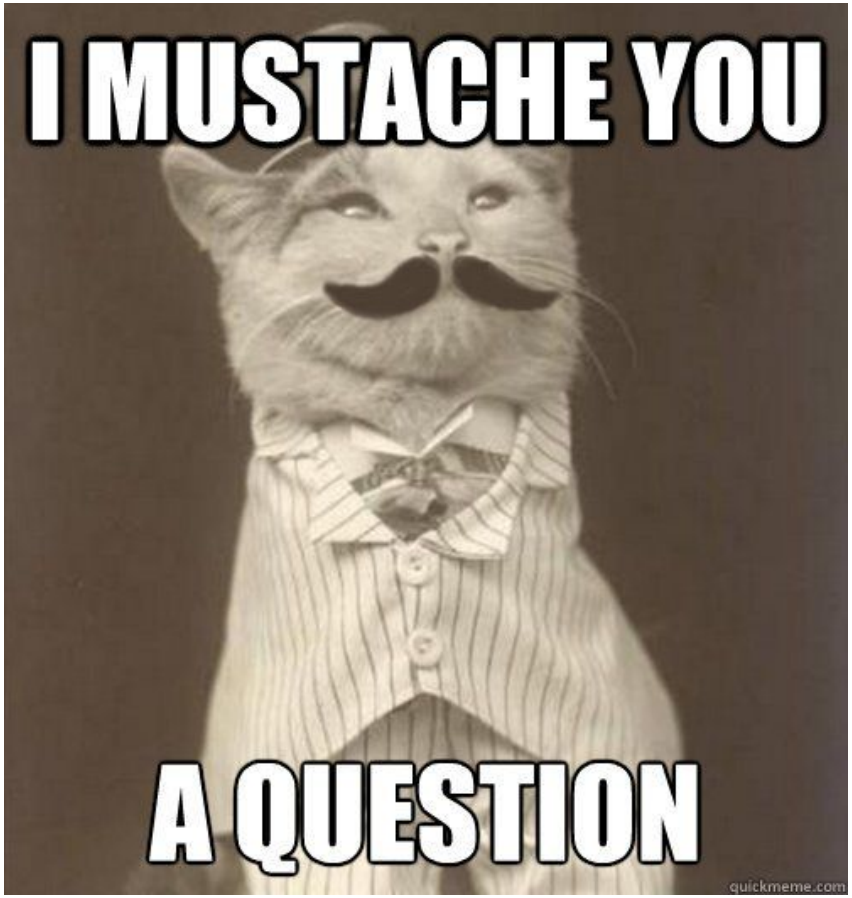
- **BBTools is Java code but there is C code linked to the Java code using JNI (Java Native Interface)**
- **Added OpenACC pragma statements**
  - Had to refactor C code to be thread-friendly
  - Did not successfully accelerate code
    - got wrong answers
    - Slower than CPU version (GPU initialization)



Discoveries

- Slow code
  - `“String Foo = foo + bar;” vs “String foo.append(bar);”`
  - `“String foobar.toupper();”`
- Reviewing and profiling code found some easy to fix CPU optimizations
- The GNU C++ compiler does not optimize by default (-O3)
- GPUs do not support strings
- Could not get a GPU enabled hashmap class working
- Difficult to compile 3rd party code
- Had to re-architect code to be able to take advantage of GPU acceleration

Questions?



- **Comments from JGI's NVidia Hackathon teams**
  - The NVidia hackathon was valuable
  - Refreshed software engineering skills
  - Got out of our comfort zone
  - Learned about GPU programming and GPU technologies