Early Experiences Evaluating the HPE/Cray Ecosystem for AMD GPUs

Verónica G. Melesse Vergara
Reuben D. Budiardja
Wayne Joubert
Oak Ridge National Laboratory

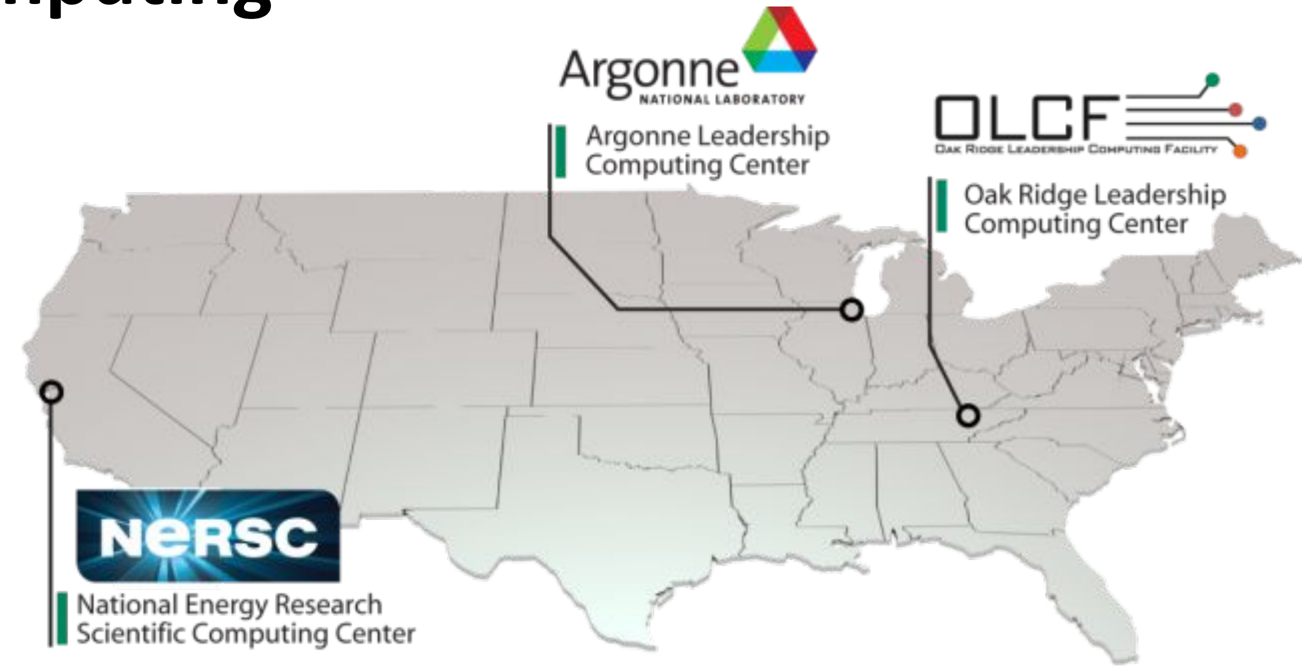
Cray User Group 2021
May 3, 2021
Virtual

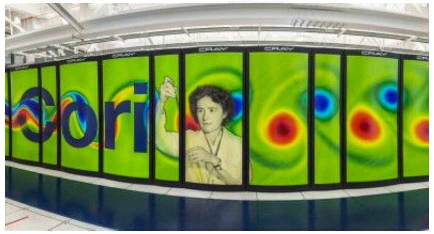
ORNL is managed by UT-Battelle, LLC for the US Department of Energy

Outline

- The Oak Ridge Leadership Computing Facility (OLCF)
- Background & Motivation
- Experimental Methods
  - Target Systems
  - Programming Models
- Results
- Lessons Learned
- Summary

The U.S. Department of Energy Office of Science and its role in computing


Argonne NATIONAL LABORATORY
Argonne Leadership Computing Center
OLCF OAK RIDGE LEADERSHIP COMPUTING FACILITY
Oak Ridge Leadership Computing Center
NERSC
National Energy Research Scientific Computing Center



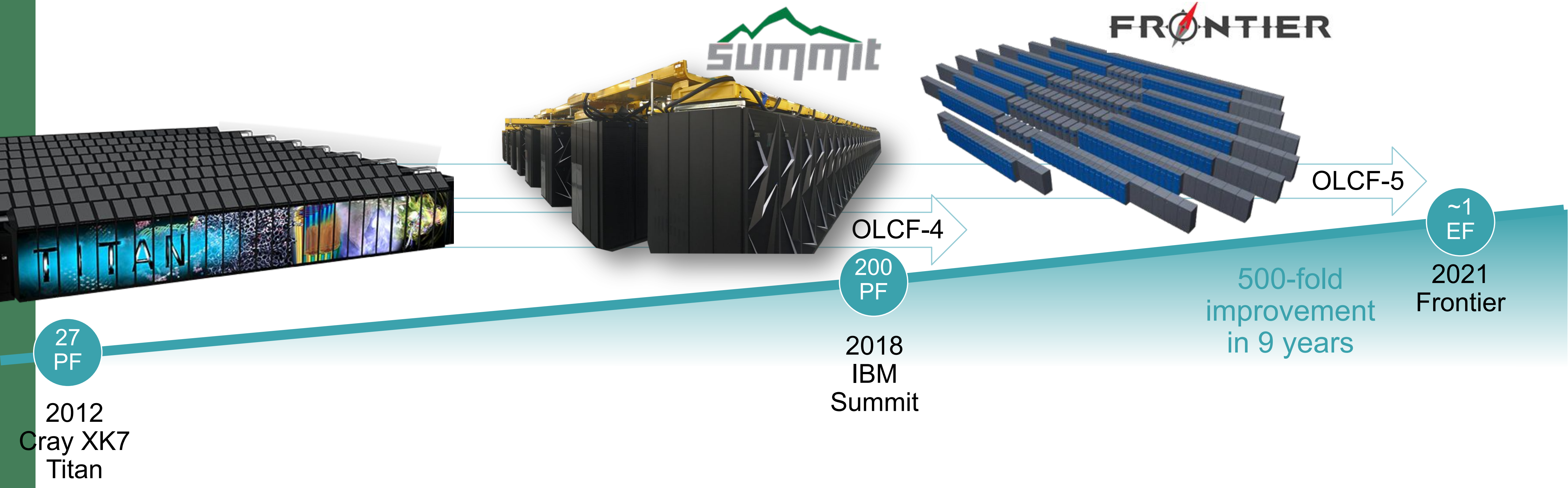
NERSC
Cori is 30 PF

ALCF
Theta is 12 PF

OLCF
Summit is 200 PF

- DOE is leader in open High-Performance Computing
- Provide the world’s most powerful computational tools for open science
- Access is free to researchers who publish
- Boost US competitiveness
- Attract the best and brightest researchers

The Oak Ridge Leadership Computing Facility



27 PF

2012
Cray XK7
Titan

OLCF-4

200 PF

2018
IBM
Summit

OLCF-5

~1 EF

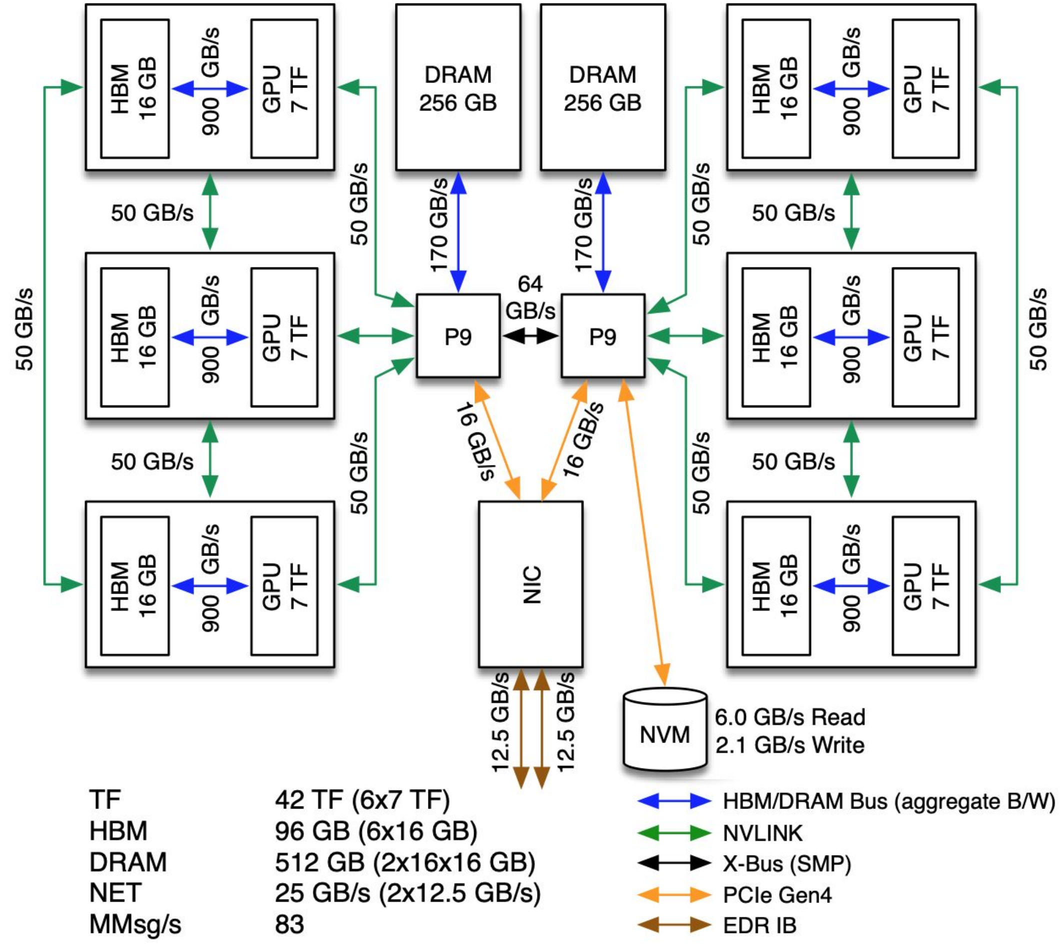
2021
Frontier

500-fold improvement in 9 years

- Transitioning from Titan to Summit was fairly straightforward as they both use NVIDIA GPUs
- Transitioning from Summit to Frontier will require porting efforts from application teams
  - Understanding the maturity of the tools available and learning from porting experiences will be key for the OLCF user community

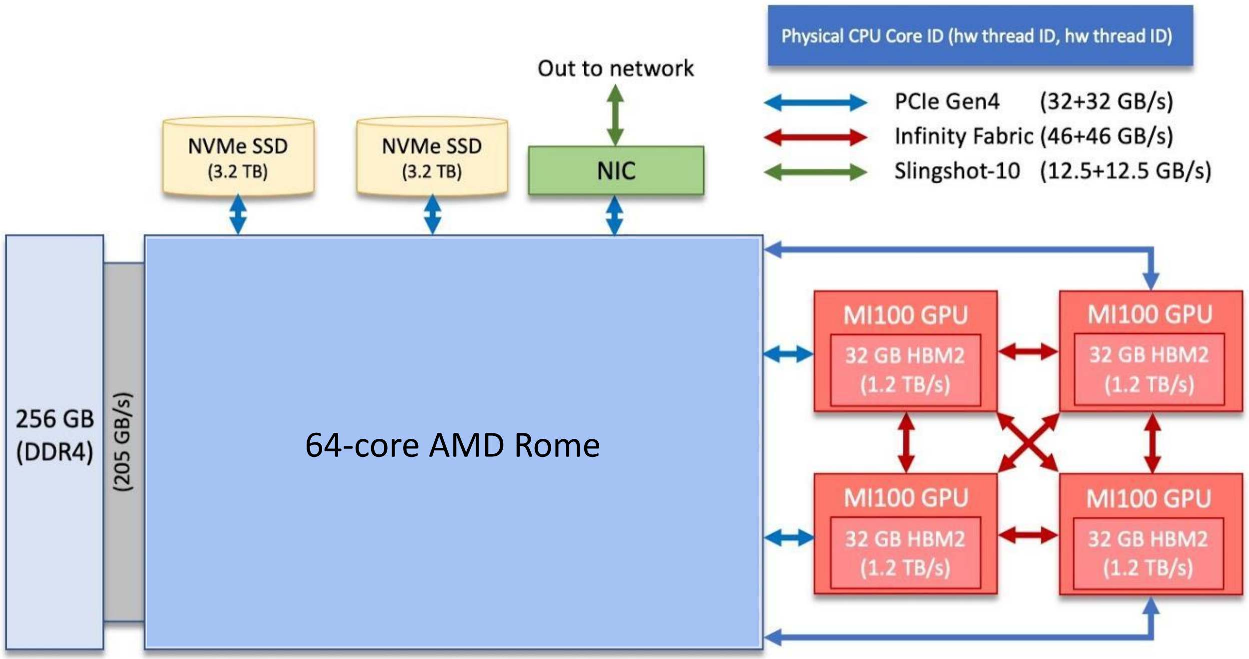
Systems

Summit



TF	42 TF (6x7 TF)
HBM	96 GB (6x16 GB)
DRAM	512 GB (2x16x16 GB)
NET	25 GB/s (2x12.5 GB/s)
MMsg/s	83

Spock



GenASiS (**Gen**eral **A**strophysics **Si**mulation **S**ystem)

- Use GenASiS `Basics` for this work: a simplified version of divergence solvers without the sophistication of multi-patches meshes and other physics modules (self-gravity, radiations, nuclear EoS, …)

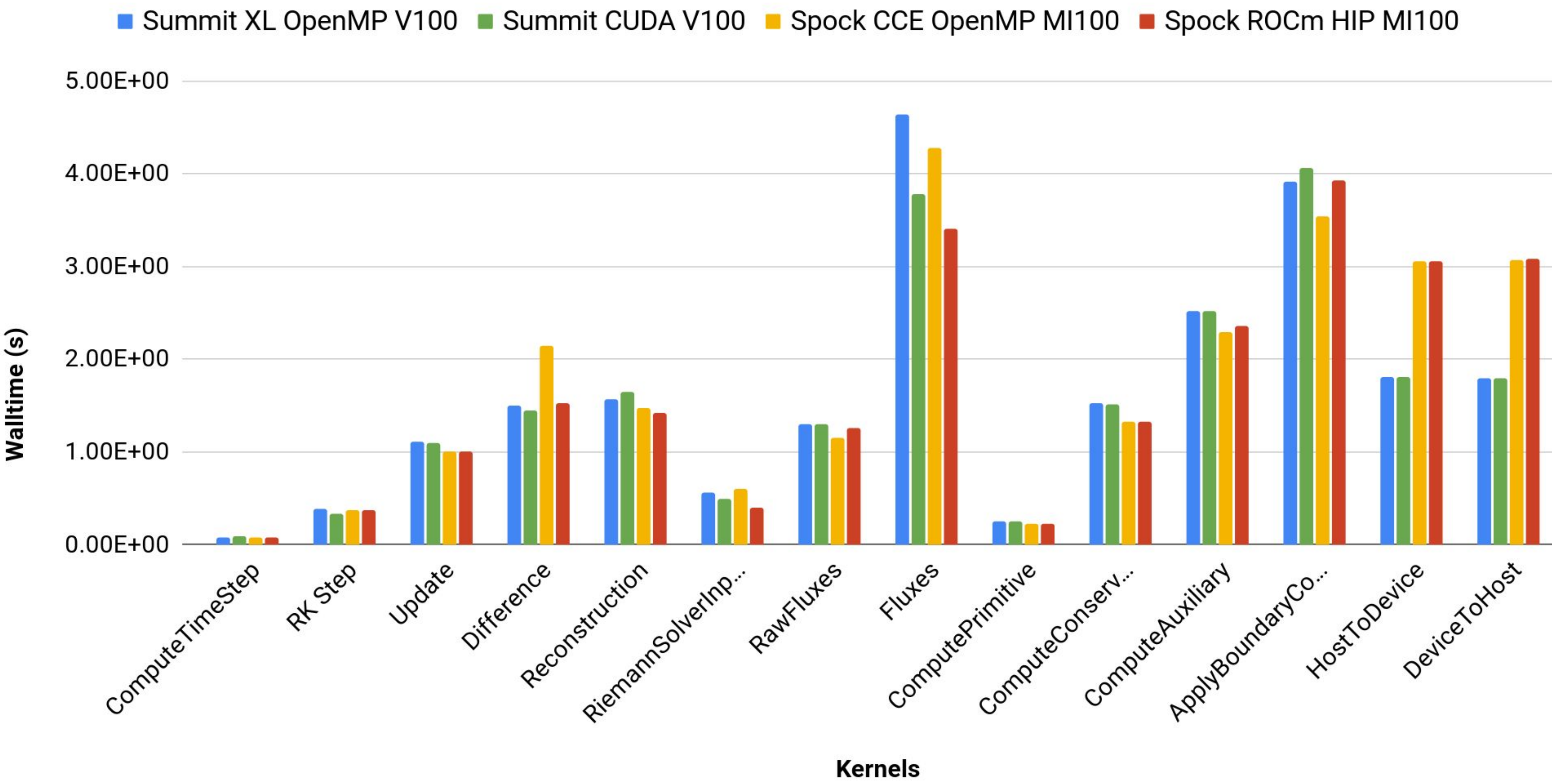
- GenASiS Basics: OpenMP offload and CUDA versions for performance testing

- OpenMP Porting is largely straightforward, except for
  - Different mapping of directives to threads, need `simd` in CCE
    !$OMP target teams distribute simd
  - Not yet implemented default mapping rule for reduction variable in CCE, need explicit mapping
  - Uncovered several Fortran and OpenMP bugs

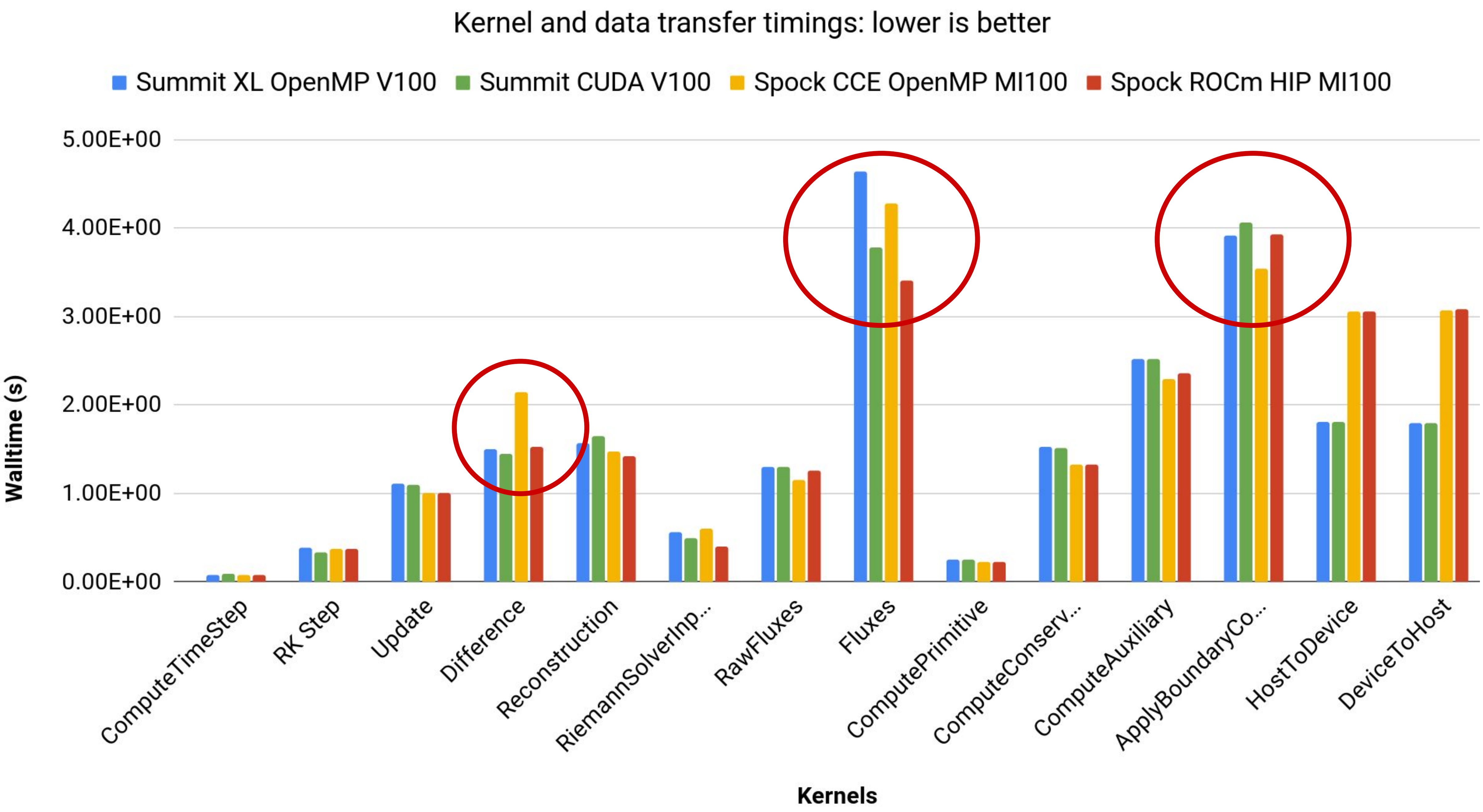
- CUDA (V100) to HIP (MI100) porting:
  - No Unified Memory support used to move array indices and offset

GenASiS Basics: RiemannProblem 3D, 256^3 per GPU, 1 process, 50 cycles

Kernel and data transfer timings: lower is better



GenASiS Basics: RiemannProblem 3D, 256^3 per GPU, 1 process, 50 cycles



Minisweep

Overview

- Minisweep is an Sn radiation transport miniapp corresponding to the Denovo radiation transport code (part of Exnihilo package)
- written in C, supports OpenMP 3.1, CUDA, OpenACC, OpenMP offload, now HIP

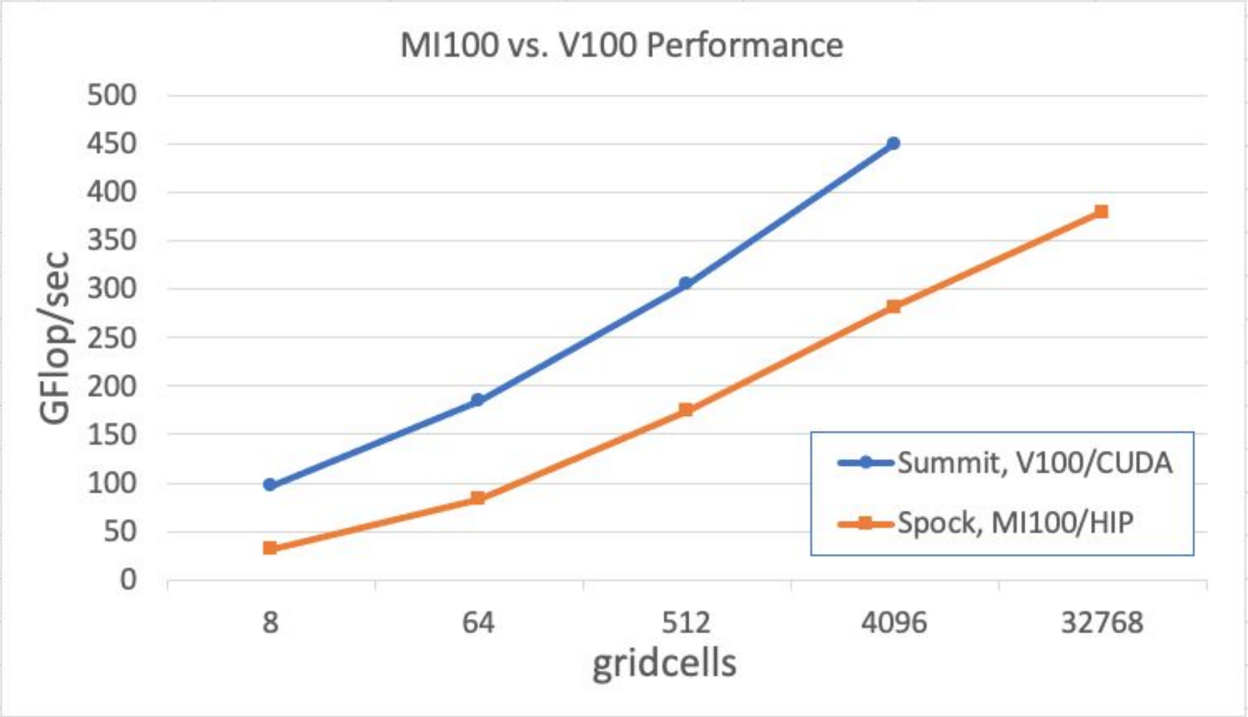
Porting experience:

- code already had CUDA constructs (mostly) in single file, using #ifdefs
- easy to manually port to HIP since API mostly isomorphic to CUDA
- a few differences, like `__CUDA_ARCH__` vs. `__HIP_DEVICE_COMPILE__`, kernel launch syntax
- used early version of `HIPLOCALConfig.cmake`, made adjustments to `CMakeLists.txt`
- overall straightforward experience

Minisweep

Performance Results

- preliminary findings
- run on 1 rank, 1 GPU, different grid sizes
- typical memory-bound performance, ~ 5% of FP peak
- MI100 and V100 qualitatively similar
- larger MI100 memory allows larger problem size
- performance better for larger problems -- amortized overheads
- slower Spock performance may be due to various reasons, possibly PCIe rate



(*) https://github.com/wdj/comet

Sparkler: Porting Experience

Overview

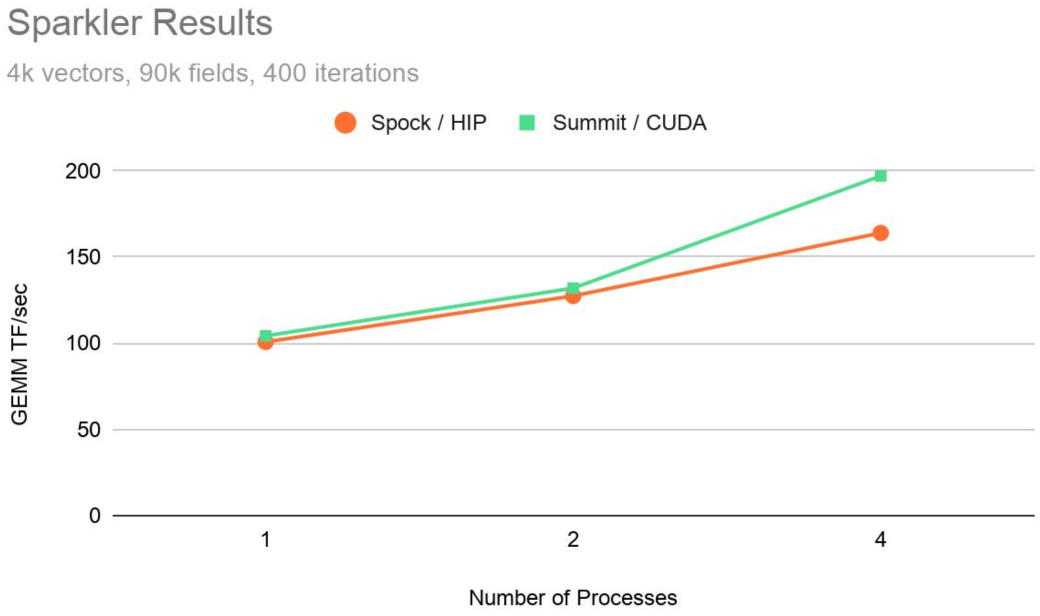
- Mini-application for the CoMet* computational genomics code
- Dense matrix-matrix multiplication for small integer elements

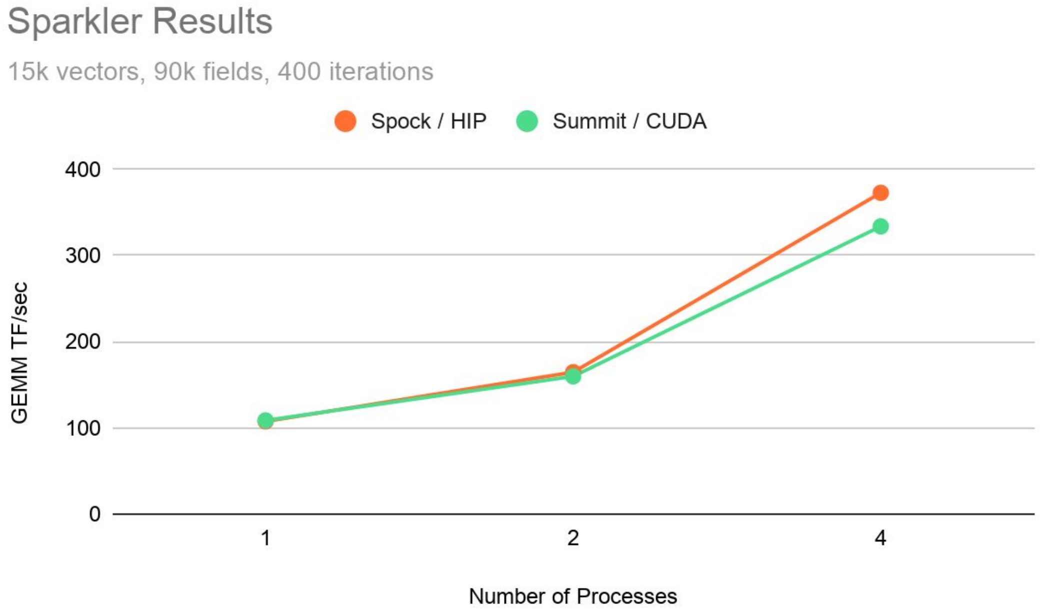
Porting experience to HIP:

- Code already supported CUDA
- Fairly straightforward though some options not directly translatable
- Started with `hipify-perl` script provided in ROCm 4.1.0
- Exact translation for the following were not available:
  - `CUBLAS_GEMM_ALGO4_TENSOR_OP` -> `HIPBLAS_GEMM_DEFAULT`
  - `CUBLAS_TENSOR_OP_MATH` -> removed
  - `cublasSetMathMode` -> removed
- HIP build uses ROCm 4.1.0 and PrgEnv-cray 8.0.0 (default on Spock)
- CUDA build uses CUDA 11.0.3, GCC 9.1.0, and ESSL 6.3.0 (default on Peak)

Sparkler: Results

- Experiments on a single node of each system
- Comparable performance for 1 and 2 devices
- Initially observed performance degradation for 4 GPU case
  - Utilizing SLURM GPU binding capabilities partially addresses the issue
- Better performance on Spock for larger problems -- amortized overheads





Conclusions

- The porting process from NVIDIA to AMD GPU platforms is fairly straightforward
  - Functionality can be obtained by both manual or script-aided ports from CUDA to HIP
  - OpenMP offloading requires minimal changes particularly with mapping
- Obtaining comparable performance “out of the box” is possible for specific cases even in the controlled environment of these mini-apps
- Additional tuning and potential code changes needed depending on the use case
- Further investigation being done to understand performance degradation of specific kernels of GenASiS, minisweep, and small Sparkler problems

Thank you! Questions?