Wrong Way: Successes, Failures, and Lessons Learned from Using the “Wrong” Programming Approach for Summit

Philip C. Roth
Oak Ridge Leadership Computing Facility
Oak Ridge National Laboratory

September 2020

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

Mid-2020 DOE Landscape

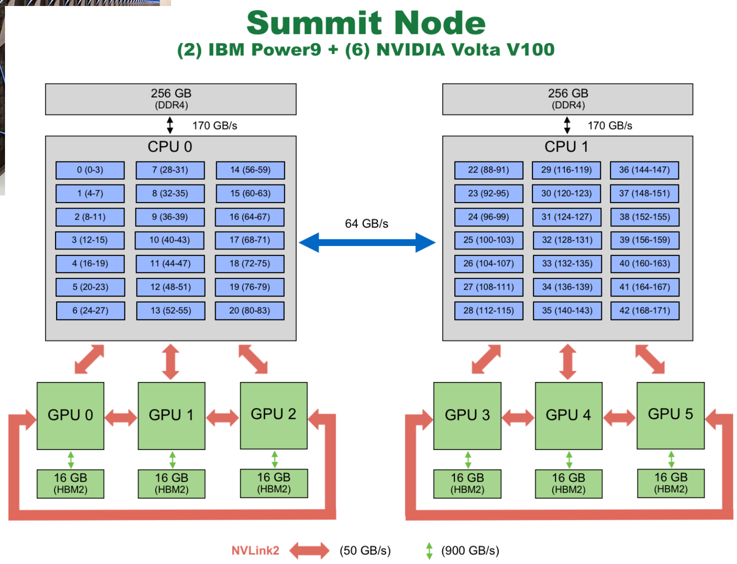
- Department of Energy (DOE) computing centers exhibit variety
  - Accelerators vs. no accelerators
    - Variety of accelerator vendors
    - Variety in CPU/GPU ratios, connectivity
  - Interconnect type, topology, capabilities
  - On-node NVM vs. near-node NVM vs. no NVM
- DOE Centers epitomize need for ***Performance, Portability, and Productivity***

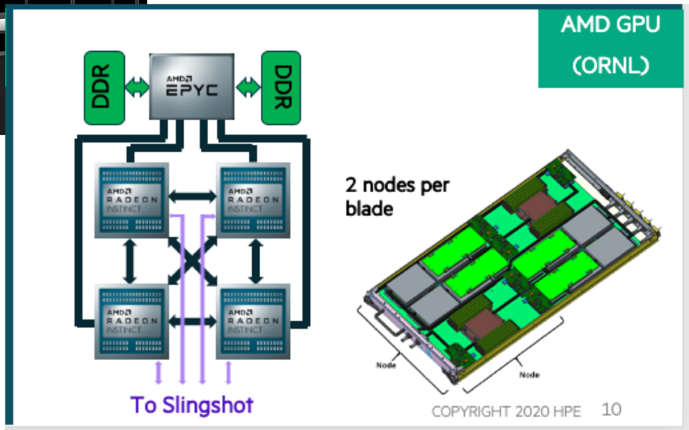
The DOE Landscape: OLCF

- Oak Ridge Leadership Computing Facility (OLCF) currently fields Summit
  - Each node contains six NVIDIA V100 GPUs and two POWER9 CPUs

- OLCF will soon deploy Frontier
  - Each node will contain four AMD Radeon Instinct GPUs and one AMD EPYC CPU

- ***<u>90%+ of OLCF systems' computational capability comes from GPUs</u>***
  - Other systems have/will have similar characteristics
  - ***<u>I will focus on GPUs in this talk</u>***





“Right Way” vs. “Wrong Way”

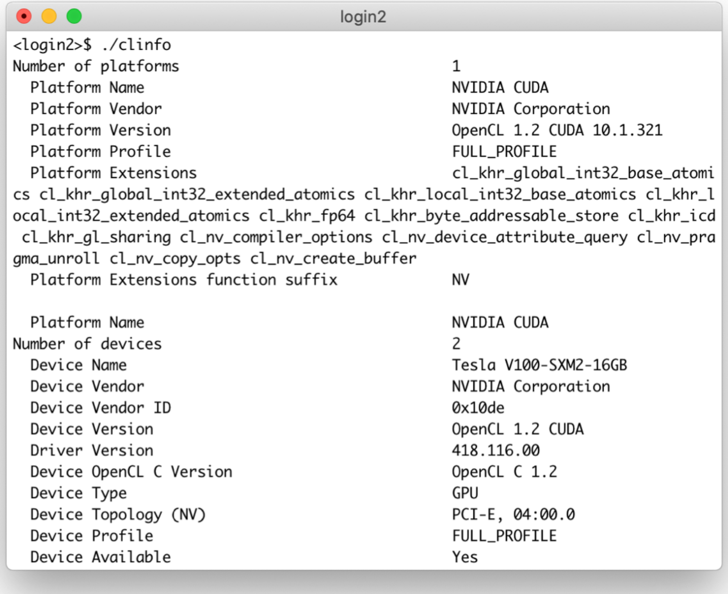
- GPU type suggests “right” or “natural” approach
  - Summit:
    - CUDA
    - OpenACC
    - OpenMP offload
  - Frontier
    - HIP
    - OpenMP offload
  - Also portability libraries (e.g., Kokkos, RAJA) with these backends
- ***Sometimes fun to consider what is possible, especially when it is “natural” on some other interesting system(s)***
  - Open source options
  - Functionality rather than performance
- ***Definitely NOT a criticism about vendor(s) choices!***

OLCF: Some Possible “Wrong Ways”

- What “wrong ways” are theoretically possible on Summit?
  - OpenCL
  - HIP
  - SYCL
  - DPC++
- And on Frontier?
  - OpenCL
  - SYCL
  - DPC++

Summit: OpenCL

- OpenCL: Khronos standard, C-based, **very** mature – and might even be performance portable
- CUDA installation includes some OpenCL-related files
  - Installable Client Driver (ICD) with config file
  - OpenCL loader library (libOpenCL.so) – but it is for X86_64
  - No OpenCL headers
  - ***To reiterate: I am not criticizing NVIDIA for not supporting OpenCL on POWER9***
- Two possible “wrong ways”
  - For both: Download Khronos headers, build Khronos ICD loader library
  - Option 1: use NVIDIA ICD
    - Platform/device queries and data transfer OK, can’t do OpenCL JIT compile
  - Option 2: use Portable Computing Language (POCL) open source OpenCL implementation



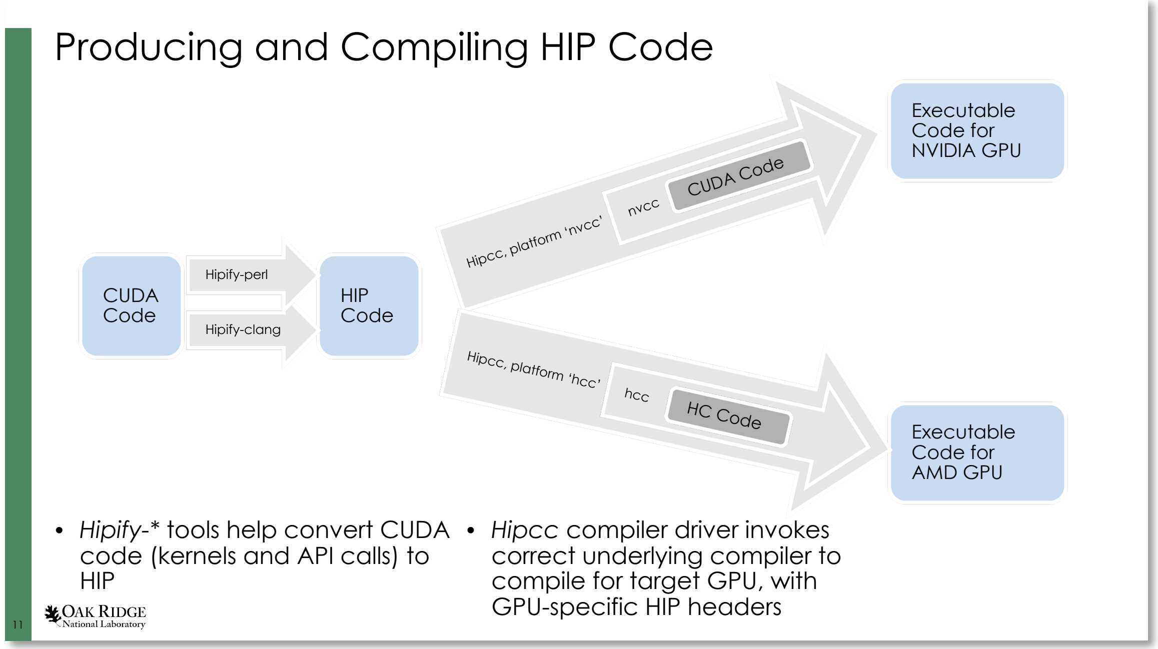
```
<login2>$ ./clinfo
Number of platforms                               1
  Platform Name                                   NVIDIA CUDA
  Platform Vendor                                 NVIDIA Corporation
  Platform Version                                OpenCL 1.2 CUDA 10.1.321
  Platform Profile                                FULL_PROFILE
  Platform Extensions                             cl_khr_global_int32_base_atomi
cs cl_khr_global_int32_extended_atomics cl_khr_local_int32_base_atomics cl_khr_l
ocal_int32_extended_atomics cl_khr_fp64 cl_khr_byte_addressable_store cl_khr_icd
 cl_khr_gl_sharing cl_nv_compiler_options cl_nv_device_attribute_query cl_nv_pra
gma_unroll cl_nv_copy_opts cl_nv_create_buffer
  Platform Extensions function suffix             NV

  Platform Name                                   NVIDIA CUDA
Number of devices                                 2
  Device Name                                     Tesla V100-SXM2-16GB
  Device Vendor                                   NVIDIA Corporation
  Device Vendor ID                                0x10de
  Device Version                                  OpenCL 1.2 CUDA
  Driver Version                                  418.116.00
  Device OpenCL C Version                         OpenCL C 1.2
  Device Type                                     GPU
  Device Topology (NV)                            PCI-E, 04:00.0
  Device Profile                                  FULL_PROFILE
  Device Available                                Yes
```

CLInfo with NVIDIA ICD

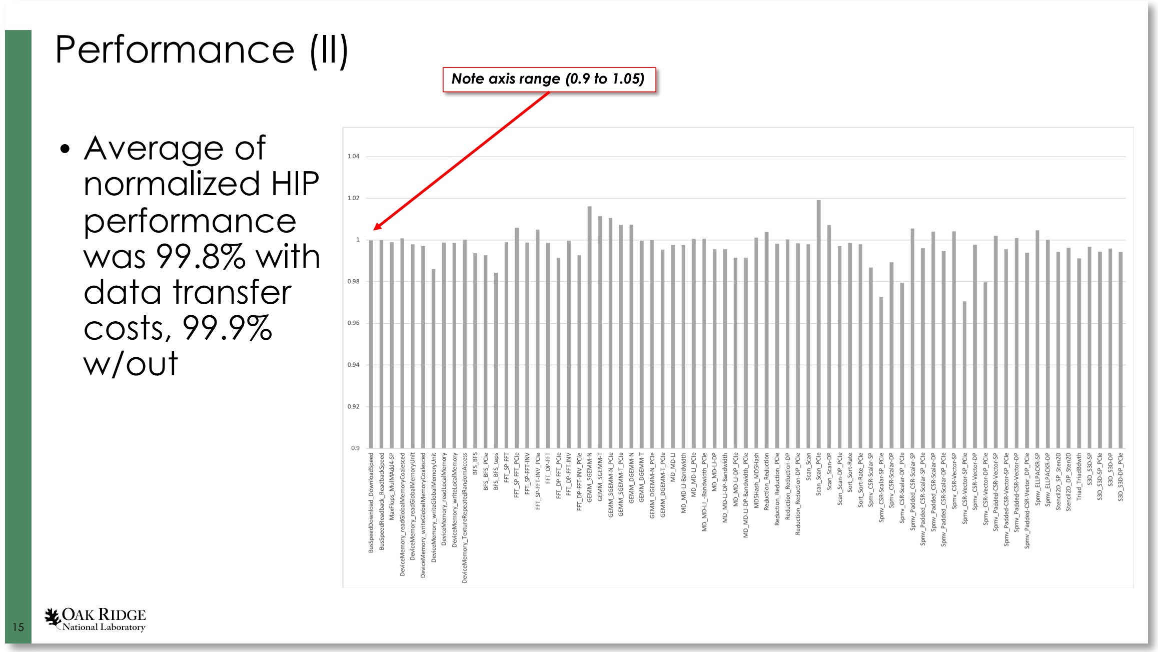
Summit: HIP

- Heterogeneous-compute Interface for Portability (HIP)
- Not really a “wrong way” on Summit
    - HIP designed as portability layer with AMD ROCm and NVIDIA CUDA backends
- OLCF provides a module for HIP but not (yet) any of the hip* libraries
    - HIP can be installed by user as header-only library
    - HIP libraries can be built for CUDA backend and installed by user



Producing and Compiling HIP Code

- *Hipify-** tools help convert CUDA code (kernels and API calls) to HIP
- *Hipcc* compiler driver invokes correct underlying compiler to compile for target GPU, with GPU-specific HIP headers



Performance (II)

- Average of normalized HIP performance was 99.8% with data transfer costs, 99.9% w/out

Note axis range (0.9 to 1.05)

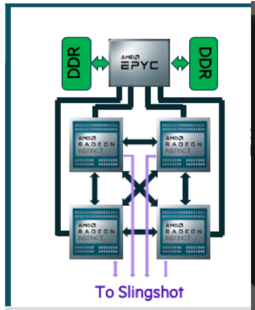
Summit: SYCL

- SYCL: Khronos standard, C++-based, OpenCL’s spiritual successor
- Some options for this “wrong way”:
  - hipSYCL: a SYCL 1.2 implementation built on HIP
    - CUDA for GPU, OpenMP for CPU
    - Have demonstrated this running on Summit with simple examples, e.g., matrix aX+Y
  - Tried using CodePlay’s Community Edition to compile kernels to PTX code on spare x86_64 system, transferring to Summit, and using them via POCL – not successful

DPC++

- Intel’s oneAPI C++-based programming approach
  - Several useful extensions to SYCL 1.2 (some appearing in SYCL 2020)
- A “wrong way” for Summit:
  - Intel LLVM staging repository includes DPC++ compiler sources
  - Found small number of build problems, e.g., reliance on CPUID instruction that isn’t supported on POWER9
- Others have reported some success in working around for other non-x86_64 platforms, so may be possible soon on Summit

Frontier: OpenCL, SYCL, and DPC++

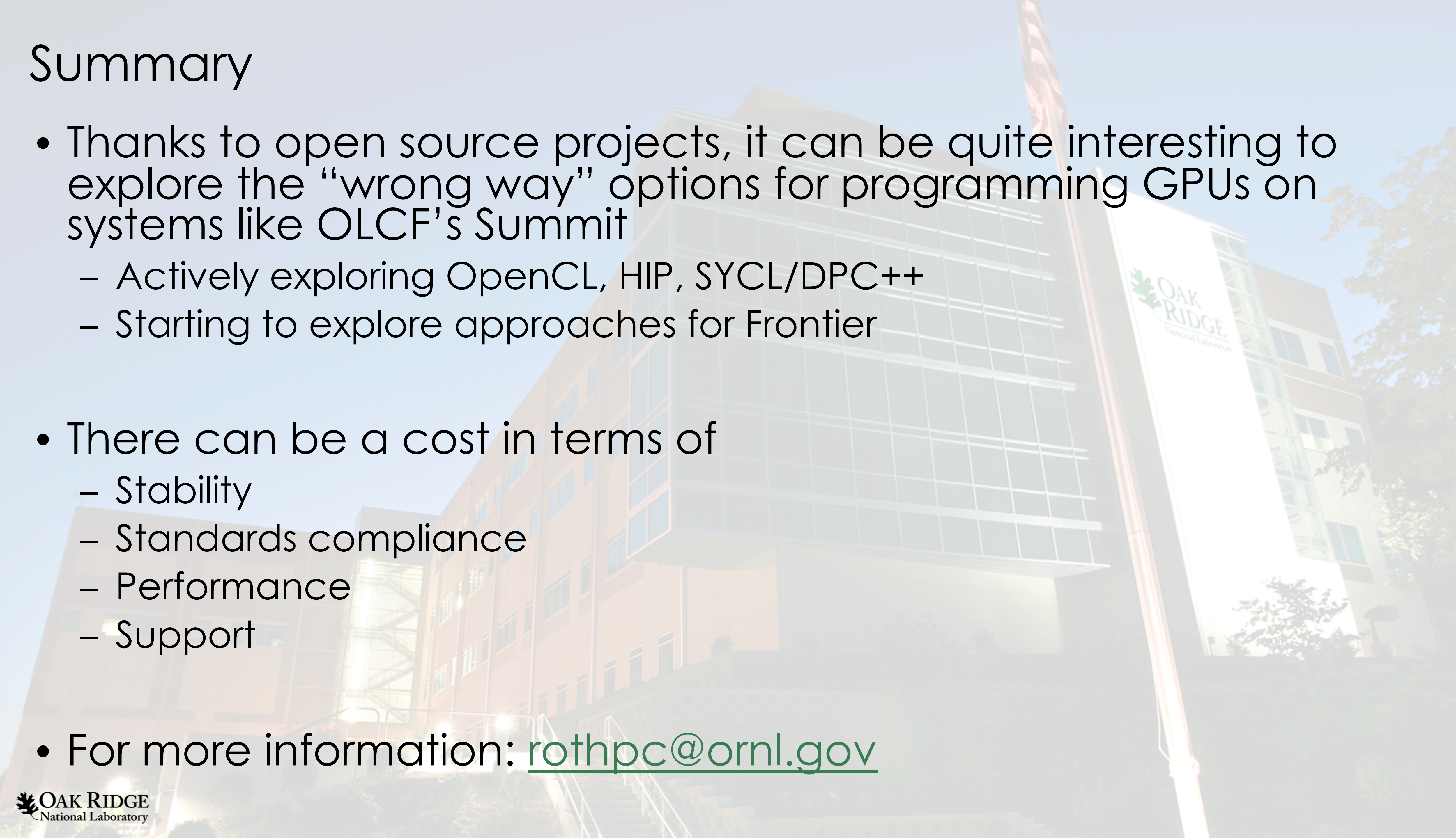


- Have less experience trying these “wrong way” approaches on pre-Frontier systems so far
- AMD has traditionally supported OpenCL
  - But SPIR/SPIR-V support varies by product line - not supported on MI25/MI60
  - Options: POCL, “manual” conversion of SPIR-V to AMDGCN
- SYCL and DPC++
  - CodePlay’s community edition
    - Earlier versions had some undocumented support for AMDGCN, missing from more recent versions
  - Intel LLVM repository
    - CPUID not an issue here
    - Still reliant on SPIR-V tools/translator to convert to AMDGCN?

Acknowledgements

- This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.
- This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Summary



- Thanks to open source projects, it can be quite interesting to explore the “wrong way” options for programming GPUs on systems like OLCF’s Summit
  - Actively exploring OpenCL, HIP, SYCL/DPC++
  - Starting to explore approaches for Frontier

- There can be a cost in terms of
  - Stability
  - Standards compliance
  - Performance
  - Support

- For more information: rothpc@ornl.gov