OAK RIDGE National Laboratory

Experiences with the Heterogeneous-compute Interface for Portability (HIP) on OLCF Summit

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

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

U.S. DEPARTMENT OF ENERGY

Heterogeneous-compute Interface for Portability (HIP)

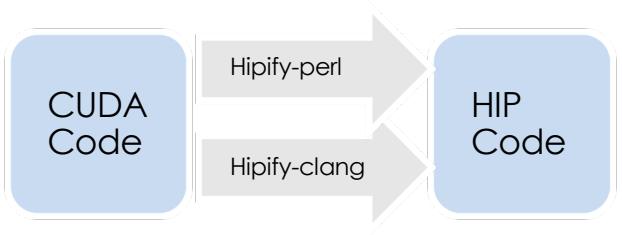
- Support for software that can run on AMD or NVIDIA GPUs
  - C++ API implemented in header-only library
  - Language for writing GPU kernels in C++ with some C++11 features
  - Tools for converting CUDA code to HIP code
- Part of AMD's Radeon Open Compute platform (ROCm)
- Sometimes viewed as one-time approach for porting CUDA software to run on AMD GPUs…
- …but can also be viewed as portability layer for simultaneously targeting both types of GPUs

Importance to OLCF

- OLCF's two most recent systems (Titan, Summit) include NVIDIA GPUs
  - User base, staff has large investment in CUDA software and knowledge
  - CUDA and associated NVIDIA libraries only supported on NVIDIA GPUs
- OLCF's next system (Frontier) will include AMD GPUs
- Summit is best platform for preparing codes for Frontier, especially at scale

- Portability tools will be important to OLCF and its users
  - Enable "Day 1 Success" of Frontier
  - Support overlapping OLCF system lifetimes
  - Support developers/users who also target non-OLCF systems

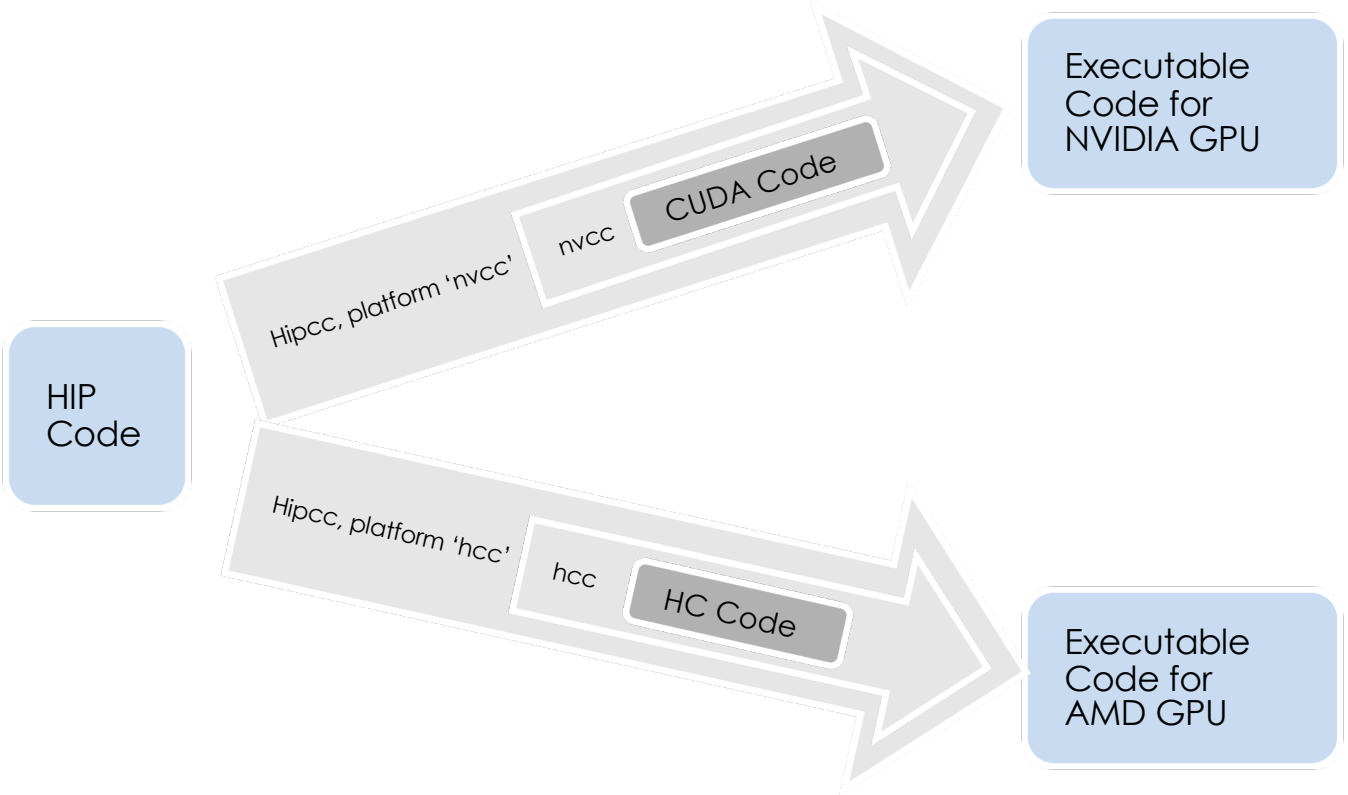
Converting CUDA Code to Executable Via HIP



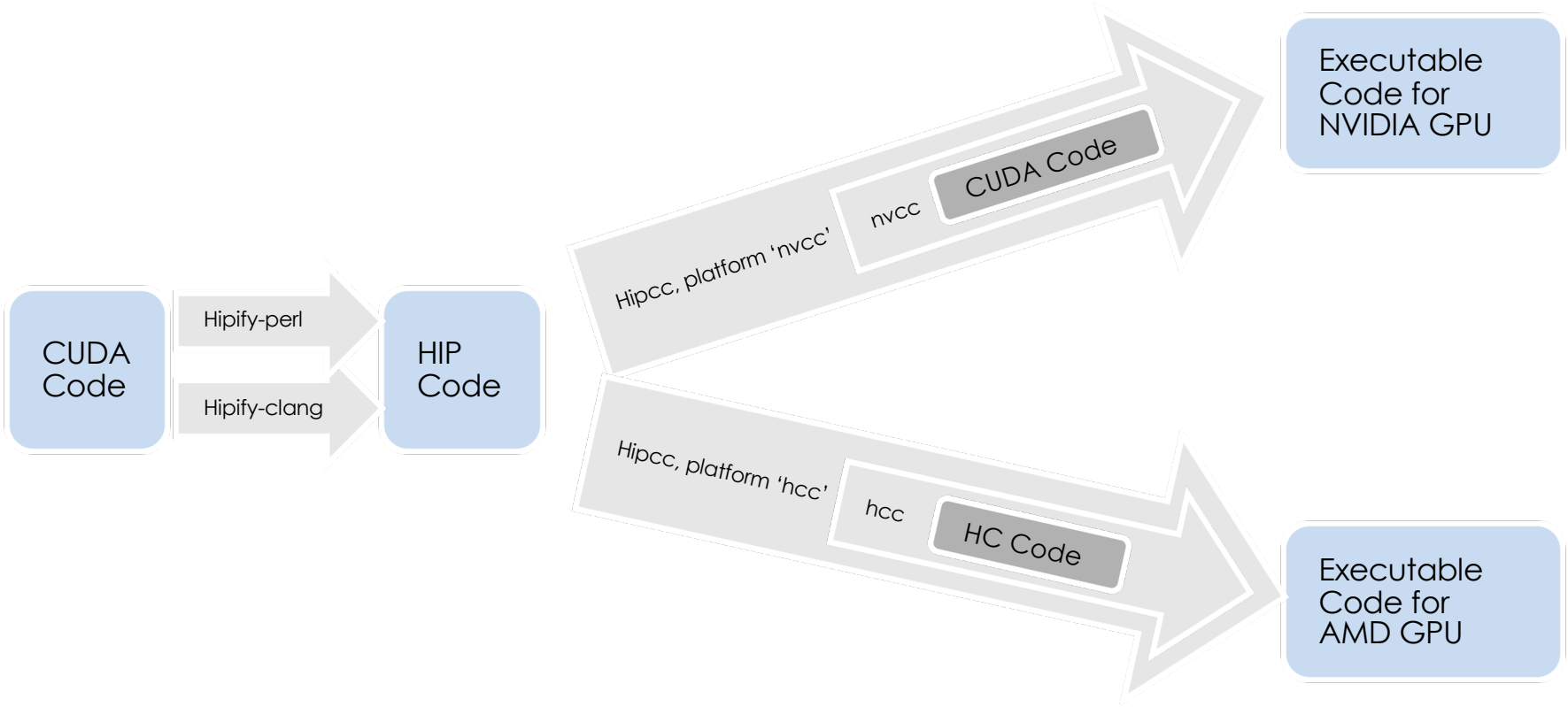
- HIP includes two tools to help convert CUDA code (kernels and API calls) to HIP
  - Hipify-perl: text-based search and replace
  - Hipify-clang: source-to-source translator that uses clang compiler front-end

Hipcc Compiler Driver Script "Magic"

- Hipcc uses HIP_PLATFORM to choose whether to compile with nvcc or hcc
- When compiling with nvcc, HIP headers include inline function implementations of HIP functions that call CUDA API
- When compiling with hcc, HIP headers include inline functions that call HC API



Converting CUDA Code to Executable Via HIP



Libraries

- NVIDIA provides a rich collection of scientific libraries accelerated for NVIDIA GPUs (e.g., cuBLAS, cuFFT, cuSPARSE)
- AMD provides similar collection of libraries for ROCm (rocBLAS, rocFFT, rocSPARSE)

- HIP "marshalling libraries" (e.g., hipBLAS, hipSPARSE) provide portability layer
  - Distributed separately from HIP
  - Slight API differences from base libraries, e.g.:
    - Handle parameter added to BLAS functions
    - DGEMM's alpha passed as pointer to double, not scalar
  - Hipify tools do not automatically convert these calls…
  - …but they do warn about them, at least for cuBLAS

Evaluation Methodology

- Built HIP and hipBLAS on Summit and installed into home directory
- Produced HIP versions of the benchmark programs from the Scalable HeterOgeneous Computing (SHOC) suite
  - Benchmark programs of a range of complexity ("level 0" to "level 2")
  - CUDA and OpenCL implementations
  - Serial versions of all benchmarks, embarrassingly parallel versions of some, truly parallel versions of a few
  - Experimental, incomplete implementations for OpenACC and Intel MIC
- Ran CUDA and HIP versions (serial only) on Summit
  - Verified functionality
  - Measured performance
- Compared amount of modification needed to produce HIP version from CUDA version

Why Build Own HIP and hipBLAS?

- Matt Belhorn (OLCF) announced a HIP installation in early May
  - Included much that would not be used in CUDA->HIP->executable conversion targeting NVIDIA GPUs
    - Patched version of LLVM with OpenMP offloading for NVIDIA GPUs
    - ROCm libraries
  - Some pieces missing: hipBLAS
  - Some undesirable requirements: gcc 4.8.5
- Note: I may very well be wrong about what was built and why
  - Relationship between hcc and hip-clang is not clear
  - hcc is based on LLVM, appears to be able to compile HIP code for AMD GPUs
  - Separate hip-clang repository seems to be mothballed

My HIP and hipBLAS Builds

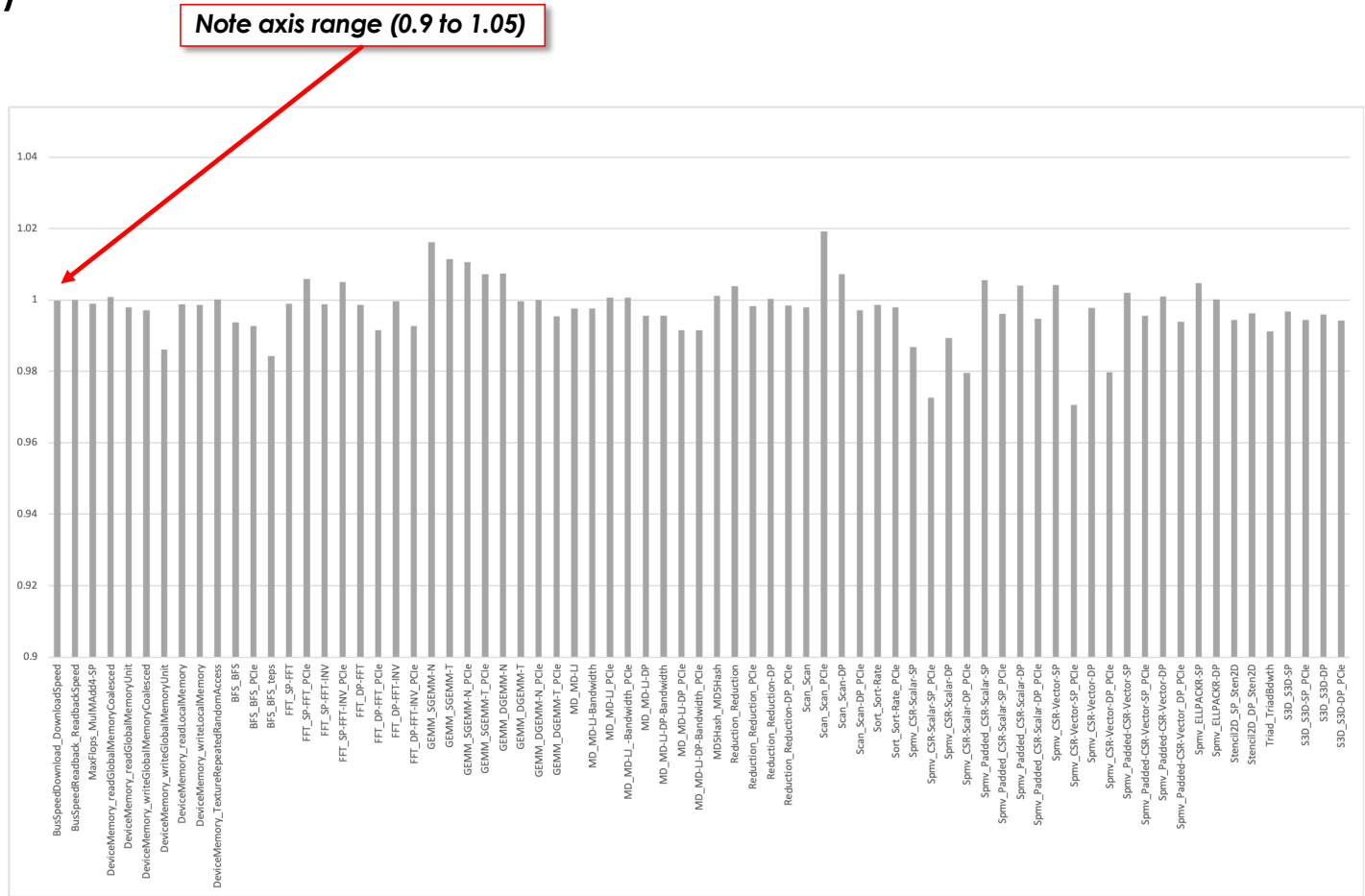
- LLVM 8.0.0, Clang 8.0.0, OpenMP 8.0.0
  - OpenMP probably not needed for targeting NVIDIA GPU via HIP
- HIP commit 418b89b, no local modifications needed
  - Built with clang 8.0.0, enabled hipify-clang
- hipBLAS commit 5d5b375d, a few local modifications needed
  - Work around inconsistencies in CMake files between HIP and hipBLAS distributions
  - Short-circuit dependency on hcc compiler, unused when targeting NVIDIA GPU

Performance (I)

- Compared performance of HIP versions against that of CUDA versions
- HIP results normalized to CUDA performance
  - No guarantee that SHOC CUDA implementation is highly optimized or best implementation for current generation GPUs in Summit
- Results considered for most metrics reported by SHOC driver
  - Includes measurements that both include and exclude data transfer costs for most benchmarks
  - QTC benchmark excluded by SHOC driver
  - Neither CUDA nor HIP version of NeuralNet benchmark ran on Summit

Performance (II)

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



OAK RIDGE National Laboratory

Performance (III)

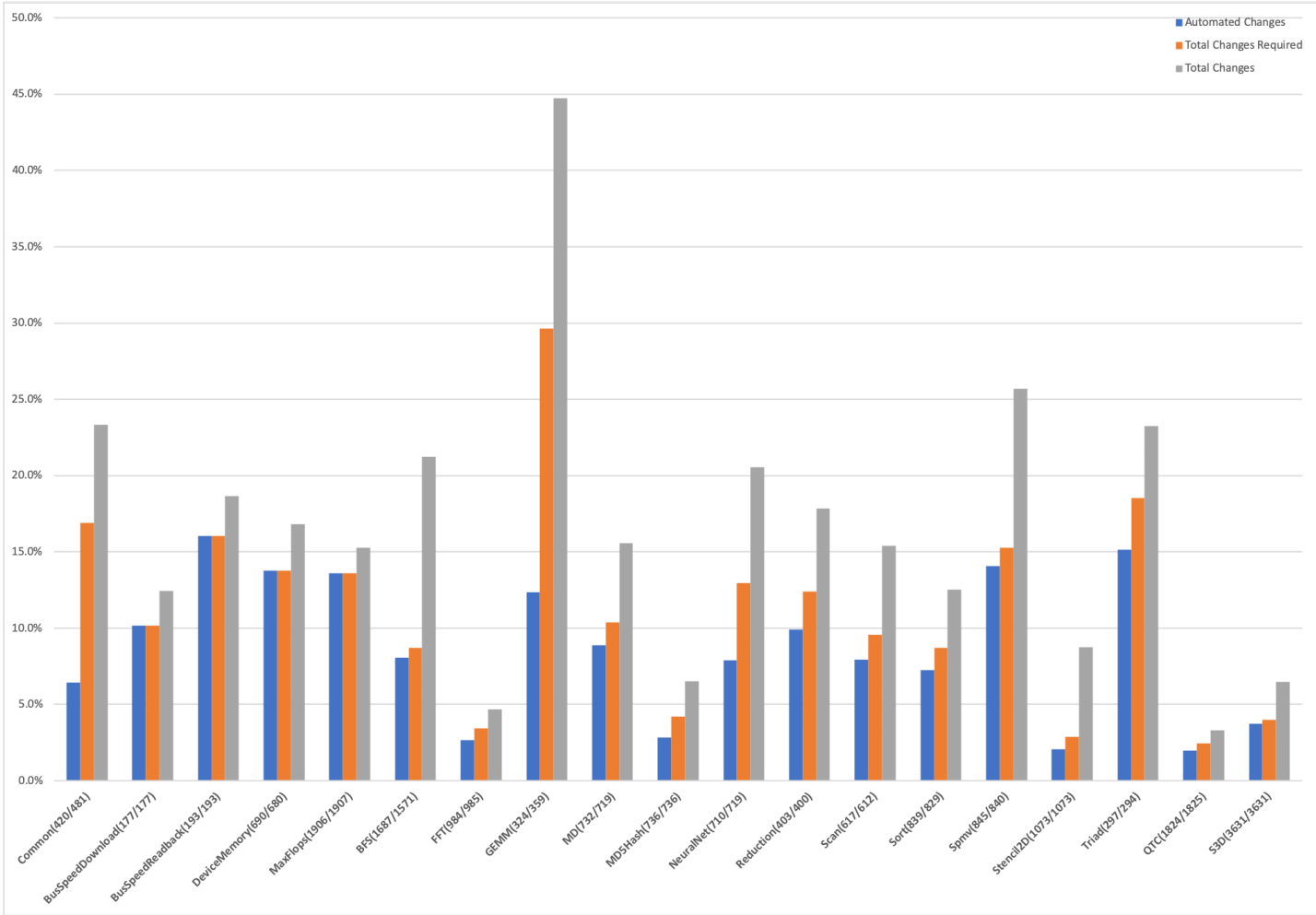
- QTC was only benchmark whose HIP version performance was substantially different from CUDA version
  - With data transfer: 43.7% slower
  - Without data transfer: 45.1% slower
- CUDA version uses texture memory. Determines amount using maxTexture2D member of device properties structure. HIP version of that structure does not have that member, so HIP version sets amount of texture memory available to 0.
- Addressing discrepancy is left to future work

Performance (IV)

- Have recently built and run the HIP-ified versions on AMD MI60 GPU
- Also measured performance of OpenCL versions on same system
- Not showing relative performance here due to
  - A few HIP runtime errors seen (MD, Spmv), not yet diagnosed
  - Spot-check of relative performance shows some wide discrepancies causing concern about OpenCL version implementation quality
  - Outdated ROCm version

Code Modifications

- % of lines of CUDA version
  - Automatically changed (blue)
  - Plus required manual changes (orange)
  - Plus desired manual changes (gray)
- Labeled as "component name (CUDA LOC/HIP LOC)"



OAK RIDGE National Laboratory

Comments on Code Modifications

- Types of "required" manual changes
  - cuBLAS -> hipblas (GEMM and NeuralNet)
    - Library initialization/finalization
    - Function signature changes
  - Headers:
    - Remove #include "cuda.h"
    - Change #include "hip/hip_runtime_api.h" to "hip/runtime.h"
  - CMakeLists to avoid conflicts with CUDA versions
    - Only 'required' because project supports both CUDA and HIP versions
- Types of "desired" manual changes
  - Change use of CUDA (in all capitalizations) to HIP in variable names, class names, macros, file names
- Caveats:
  - Counting is error prone
  - Sometimes same line is changed for multiple reasons

Hipify-perl

- Hipify-perl worked better than expected to convert SHOC CUDA code to HIP
  - Perhaps not surprising:
    - SHOC development stopped years ago so does not use recent CUDA features
    - SHOC's use of CUDA is mostly basic functionality
- Warns about use of cuBLAS, but no automatic conversion to hipBLAS calls
  - SHOC benchmarks using cuBLAS were written for version 1 API

Hipify-clang (I)

- Did not try to use hipify-clang to convert all of SHOC
- Can build hipify-clang using stock LLVM/Clang
  - I used LLVM and Clang 8.0.0, with OpenMP
  - Since only used for conversion to HIP code, doesn't need offload capabilities, probably not even OpenMP at all
- CUDA version specific
  - Currently gives "Unsupported version" for CUDA 10.1
  - Seems to work with CUDA 9.2
  - Did not have this problem with hipify-perl (it does simple textual substitution)

Hipify-clang (II)

- Must be able to find headers
  - Not necessarily easy to produce all command lines for code bases with complex build requirements
  - Has silent support for finding headers in ./include (clang itself does not)
- Does not consider preprocessor macros (or maybe some but not all?)
  - E.g., in #ifdef/else/endif, changes code in both #if and #else branches regardless of value of controlling macro
- Unlike hipify-perl, does *some* conversion of cuBLAS calls
  - Did not handle the cuBLAS v1 API use
    - Did not recognize cublasInit or cublasShutdown calls, but did warn about them
    - Changed cublasDgemm -> hipblasDgemm but did not change argument types (cublas char vs. hipblasOperation_t, alpha param as pointer instead of scalar)

Other Thoughts on Hipifying Existing Code

- Hipifying SHOC revealed lots of user code where development team used CUDA-specific naming scheme
  - Header names (cudacommon.h)
  - Variable names (start_cuda_event, stop_cuda_event)
  - Macros for checking CUDA call results (CUDA_SAFE_CALL, CHECK_CUDA_ERROR)
- Developers may want to convert to HIP-based names if using HIP as portability layer, or more generic names
- What to do with #ifdefs controlled by CUDA version number?
  - One approach: portability interface allowing to test for features rather than version numbers of underlying APIs

Bonus: SYCL on Summit

- SYCL is a single-source C++-based programming model targeting systems with heterogeneous architecture
- Most (known) SYCL implementations are built on OpenCL…
  - …but NVIDIA's distribution for POWER does not support OpenCL
- hipSYCL: a SYCL implementation built on HIP
  - Distribution includes HIP plus hipCPU implementation that targets CPU only
- Very recently (last week) demonstrated hipSYCL toolchain working on Summit
  - LLVM/Clang 8.0.1 with CUDA language support and NVPTX backend
  - Demonstrated functionality with simple example (aM+N where a scalar, M, N 2D matrices)
    - Minor modifications needed (use 'id' instead of 'item' data type for kernel index space)
  - Work in progress to add SYCL versions of SHOC programs

Summary

- Recently used "hipify" tools to produce HIP versions of SHOC benchmarks from CUDA versions
  - Percentage of lines changed varied by benchmark, from <5% to approximately 45%; worst case involved heavy use of cuBLAS
- Measured performance of both versions when running on NVIDIA GPUs in Summit
  - Performance of HIP versions overall was very similar to CUDA versions
- Next steps
  - Resolve problems/concerns with HIP/OpenCL comparison
  - Complete SYCL versions and compare performance/lines of code with HIP and OpenCL versions on both Summit (with hipSYCL) and AMD MI60 system (probably with both hipSYCL and CodePlay implementations)
- For more information: rothpc@ornl.gov