It's HIP to be Open

Convert your CUDA Code to C++ Using AMD's New HIP Tool

The world of HPC as it exists today consists of many GPU-accelerated applications that use the proprietary CUDA language and infrastructure. The problem with using proprietary software is that it is almost always tightly controlled and its source code is most often kept secret, and with CUDA, the hardware options are limited to one vendor.

AMD, a strong proponent of open source and open standards, has created a new tool that will allow developers to convert CUDA code to common C++. The resulting C++ code can run through either CUDA NVCC or AMD HCC compilers. This new Heterogeneous-compute Interface for Portability, or HIP, is a tool that provides customers with more choice in hardware and development tools.

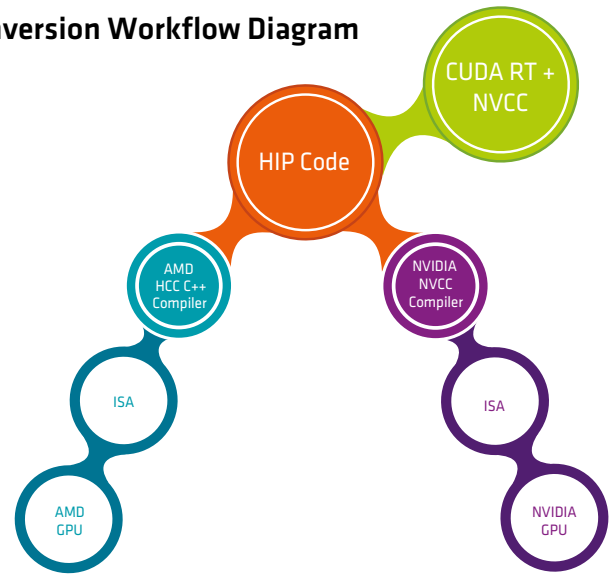
How does HIP's portability compare to OpenCL™?

Both AMD and our competitors support OpenCL™ 1.2 on their devices, and this can be used to write portable code.

HIP offers several benefits over OpenCL:

- Developers can code in C++, and mix host and device C++ code in their source files. HIP C++ code can use templates, lamdbas, classes, etc.
- HIP API is less verbose than OpenCL, and C++ is familiar to CUDA developers.
- Because both CUDA and HIP are C++ languages, porting from CUDA to HIP is significantly easier than porting from CUDA to OpenCL.
- HIP uses the state-of-the-art development tools on each platform: on competitor GPUs, HIP code is compiled with NVCC and can use nSight profiler and debugger.
- HIP provides pointers and host-side pointer-arithmetic.
- HIP provides device-level control over memory allocation and placement.
- HIP offers an offline compilation model.

Code Conversion Workflow Diagram



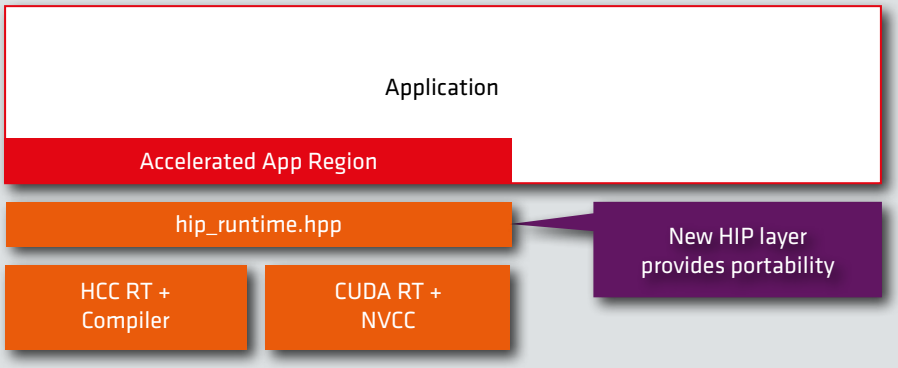
Implementation

- Header maps hip* calls to CUDA RT or HSA RT
- Strong subset of CUDA RT functionality, focused on most commonly used functions like memory management, events and streams
- Layers on HCC runtime and C++ support
- Can utilize #ifdef for complicated cases and/or performance tuning

Supported APIs

HIP provides:

- **Devices** (hipSetDevice(), hipGetDevice(), hipGetDeviceProperties())
- **Memory management** (hipMalloc(), hipMemcpy(), hipMemcpyAsync(), hipMemSet(), hipFree(), ...)
- **Streams** (hipStreamCreate(), hipStreamWaitEvent(), hipStreamSynchronize(), hipStreamDestroy(),...)
- **Events** (hipEventRecord(), hipEventElapsedTime(), etc)
- **Kernel launching** (hipLaunchKernel is standard C/C++ function replacing <<< >>>)
- **CUDA-style kernel indexing** (hipBlockIdx, hipThreadIdx, ...)
- **Device-side math builtins** (200+ functions covering entire CUDA math library)
- **Error reporting** (hipGetLastError(), hipGetErrorString(), ...)



Example Code Conversion

Cuda Code

```
cudaMalloc((void **) &m_cuda, Size * Size * sizeof(float));
cudaMemcpy(m_cuda, m, Size * Size * sizeof(float),cudaMemcpyHostToDevice );
gpu_kernel<<<dimGridXY,dimBlockXY>>>(m_cuda,a_cuda,b_cuda,Size,Size-t,t);
cudaThreadSynchronize();
cudaMemcpy(m, m_cuda, Size * Size * sizeof(float),cudaMemcpyDeviceToHost );
cudaFree(m_cuda);

__global__ void gpu_kernel(float *m_cuda, float *a_cuda, float *b_cuda,int Size,
int j1, int t)
{
            …
}
```

HIP-ify

```
hipMalloc((void **) &m_cuda, Size * Size * sizeof(float));
hipMemcpy(m_cuda, m, Size * Size * sizeof(float),hipMemcpyHostToDevice );
hipLaunchKernel(gpu_kernel, dim3(dimGridXY), dim3(dimBlockXY), 0, 0,
                                m_cuda,a_cuda,b_cuda,Size,Size-t,t);
hipDeviceSynchronize();
hipMemcpy(m, m_cuda, Size * Size * sizeof(float),hipMemcpyDeviceToHost );
hipFree(m_cuda);

__global__ void HIP_FUNCTION(gpu_kernel, float *m_cuda, float *a_cuda, float *b_
cuda,int Size, int j1, int t)
{
      …
}
HIP_FUNCTION_END
```

Learn more about AMD and the Heterogeneous computing at http://developer.amd.com/heterogeneous-computing/