OpenMP as GPU Kernel Language
Progress Report

Johannes Doerfert <jdoerfert@llnl.gov>
Tom Scogland <scogland1@llnl.gov>

Prepared by LLNL under Contract DE-AC52-07NA27344.
int i = blockDim.x;
if (i >= N) return;
int j = threadIdx.x;
if (j >= M) return;
body1(i, j);
__syncthreads();
body2(i, j);
int i = blockIdx.x;
if (i >= N) return;
int j = threadIdx.x;
if (j >= M) return;
body1(i, j);
__syncthreads();
body2(i, j);

#pragma omp target teams distribute
for (int i = ...) {
    #pragma omp parallel for
    for (int j = ...) {
        body1(i, j);
        #pragma omp parallel for
        for (int j = ...) {
            body2(i, j);
        }
    }
}
int i = blockIdx.x;
if (i >= N) return;
int j = threadIdx.x;
if (j >= M) return;
body1(i, j);
__syncthreads();
body2(i, j);
}

#pragma omp target loop
for (int i = ...) {
    #pragma omp loop
    for (int j = ...) 
        body1(i, j);
    #pragma omp loop
    for (int j = ...) 
        body2(i, j);
}
OpenMP Kernel Language
_LIBOMPX_NAMESPACE_BEGIN

extern "C" void *ompx_malloc(size_t size) {
    if (size <= 0)
        return nullptr;
    return omp_target_alloc(size, omp_get_default_device());
}

void *ompx::malloc(size_t size) { return ompx_malloc(size); }

_LIBOMPX_NAMESPACE_END
/** AMDGCN Implementation **/

```c
#pragma omp begin declare variant match(device = {arch(amdgcn)})

uint32_t ompx_get_thread_num(int Dim = 0) {
    switch(Dim) {
        case 0:
            return __builtin_amdgcn_workitem_id_x();
        case 1:
            return __builtin_amdgcn_workitem_id_y();
        case 2:
            return __builtin_amdgcn_workitem_id_z();
        default:
            break;
    }
    __builtin_unreachable();
}

...  // This is the end of the function definition.
```

#pragma omp end declare variant
kern<<<nbblocks, nthreads, shmem>>>(a1, a2)

#pragma omp target teams num_teams(nbblocks) thread_limit(nthreads) \ ompx_cgroup_dyn_mem(shmem) ompx_kernel
kern(a1, a2)
__device__ void foo();

void foo();
#pragma omp declare target device_type(nohost) to(foo)
__device__ void foo();
CUDA via LLVM/OpenMP
LLVM/OpenMP as Target Independent Runtime Layer

Figure 8: Execution times of XSBench.

Breaking the Vendor Lock — Performance Portable Programming Through OpenMP as Target Independent Runtime Layer (PACT’22)
LLVM/OpenMP as Target Independent Runtime Layer

Breaking the Vendor Lock – Performance Portable Programming Through OpenMP as Target Independent Runtime Layer (PACT’22)

Figure 8: Execution times of XSBench.

Figure 10: Execution times of Lulesh.
Questions?
OpenMP as Intermediate Layer
LLVM/OpenMP Target Offloading

- OpenMP offload code compilation for CPUs, virtual GPU (VGPU), AMD and NVIDIA GPUs
LLVM/OpenMP Target Offloading

- OpenMP offload code compilation for CPUs, virtual GPU (VGPU), AMD and NVIDIA GPUs
- Intel GPU support is WIP
LLVM/OpenMP Target Offloading + Math Runtimes

- OpenMP offload code compilation for CPUs, virtual GPU (VGPU), AMD and NVIDIA GPUs
- Intel GPU support is WIP
- Target independent math library (libm.a) for all supported architectures. Defines \( \sin(...) \), etc.
LLVM/OpenMP Target Offloading + CUDA Device Compilation

- OpenMP offload code compilation for CPUs, virtual GPU (VGPU), AMD and NVIDIA GPUs
- Intel GPU support is WIP
- Target independent math library (libm.a) for all supported architectures. Defines $\sin(...)$, etc.
- CUDA device code interoperability with OpenMP target. Link in CUDA device runtimes e.g., Thrust.
• OpenMP offload code compilation for CPUs, virtual GPU (VGPU), AMD and NVIDIA GPUs
• Intel GPU support is WIP
• Target independent math library (libm.a) for all supported architectures. Defines $\sin(...)$, etc.
• CUDA device code interoperability with OpenMP target. Link in CUDA device runtimes e.g., Thrust.
• Define CUDA API and builtins through OpenMP runtime functions. Allow to retarget CUDA codes.
LLVM/OpenMP as Target Independent Runtime Layer (WIP)

- OpenMP offload code compilation for CPUs, virtual GPU (VGPU), AMD and NVIDIA GPUs
- Intel GPU support is WIP
- Target independent math library (libm.a) for all supported architectures. Defines \( \sin(...) \), etc.
- CUDA device code interoperability with OpenMP target. Link in CUDA device runtimes e.g., Thrust.
- Define CUDA API and builtins through OpenMP runtime functions. Allow to retarget CUDA codes.
- HIP, SYCL, and other languages can be added as needed. Full interoperability and portability.
LLVM/OpenMP as Target Independent Runtime Layer (WIP)

- OpenMP offload code compilation for CPUs, virtual GPU (VGPU), AMD and NVIDIA GPUs
- Intel GPU support is WIP
- Target independent math library (libm.a) for all supported architectures. Defines $\sin(\ldots)$, etc.
- CUDA device code interoperability with OpenMP target. Link in CUDA device runtimes e.g., Thrust.
- Define CUDA API and builtins through OpenMP runtime functions. Allow to retarget CUDA codes.
- HIP, SYCL, and other languages can be added as needed. Full interoperability and portability.
- Overall WIP but proof-of-concept is ready and under review (PACT) right now. Parts have been upstreamed (incl. Driver) or are prepared to be.
LLVM/OpenMP as Target Independent Runtime Layer (WIP)

Host GDB running the SU3 bench CUDA code via the OpenMP layer on the virtual GPU.