Evaluation of OpenMP for Portable CPU and GPU Programming with GridMini

Meifeng Lin[1,*], Peter Boyle[1,2], Lingda Li[1], Vivek Kale[1]
[1] Brookhaven National Laboratory [2]University of Edinburgh
[*]Presenter. mlin@bnl.gov

Introduction

- Lattice QCD is a computational framework to simulate the strong interactions between quarks and gluons, and provides essential theoretical input to nuclear and high energy physics research.
- Computationally expensive. Need as many resources as we can get.
- Multi-prong approach to exascale performance portability
  - HIP, SyCL, Kokkos are all being investigated
- **OpenMP as another possible path to performance portability**

Grid and GridMini

- **Grid [1]** is a modern C++ library for lattice QCD
- Arranges the data layout as if the lattice is divided into virtual "sub-lattices", with one SIMD lane for each sublattice.
- Same data layout can be mapped to GPU architectures
- Extensive use of templates for high-level abstraction
- Custom expression template engine for performance
- **Header file with macros to encapsulate architecture-dependent implementations**

```
#ifdef GRID_NVCC
#define accelerator          __host__ __device__
#define accelerator_inline __host__ __device__ inline
#define accelerator_for (…) { //CUDA kernel}
#else
#define strong_inline      __attribute__((always_inline)) inline
#define accelerator
#define accelerator_inline strong_inline
#define accelerator_for(…)  thread_for(…) //for loop with
#pragma omp parallel for
```

- **Custom AlignedAlloctor for dynamic memory allocation on different architecture**

```
#ifdef GRID_NVCC
    if ( ptr == (_Tp *) NULL ) auto err =
cudaMallocManaged((void **)&ptr,bytes);
#else
  #ifdef HAVE_MM_MALLOC_H
    if ( ptr == (_Tp *) NULL ) ptr = (_Tp *)
_mm_malloc(bytes,GRID_ALLOC_ALIGN);
  #else
    …
```

- **GridMini [2]:** A substantially reduced version of Grid for experimentation with different programming models.
- Retains same Grid structure: data structures/types, data layout, aligned allocators, macros, …
- Only keeps the high-level components necessary for the benchmarks.

[1] https://github.com/paboyle/Grid
[2] https://github.com/meifeng/GridMini

OpenMP Offloading w/ UVM

- SU(3)×SU(3) benchmark: STREAM-like memory bandwidth test
- Important as LQCD is bandwidth bound.

```
double start=usecond();
for(int64_t i=0;i<Nloop;i++){
    z=x*y; //x,y,z are all arrays of 3x3 matrices
}
double stop=usecond();
double time=(stop-start)/Nloop*1000.0;

double bytes=3*vol*Nc*Nc*sizeof(Complex);
double flops=Nc*Nc*(6+8+8)*vol;
double bandwidth=bytes/time; //GB/s
double Gflops=flops/time;    //0.9 flops/byte SP
```

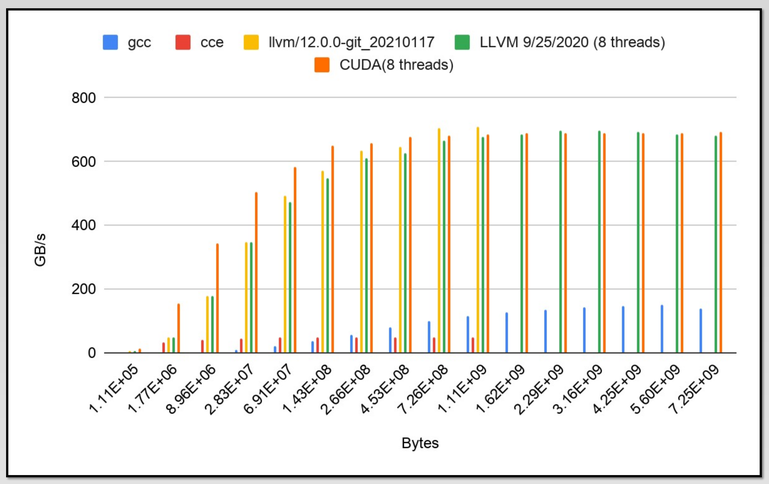
- To enable OpenMP offloading requires **two considerations**
  - **New macros** for OpenMP target offloading

```
#elif defined (OMPTARGET)
#define accelerator_inline strong_inline
#define accelerator_for(iterator,num,nsimd, ... )  \
{                                                  \
    _Pragma("omp target teams distribute parallel for") \
     naked_for(iterator, num, { __VA_ARGS__ }); \
}
```

  - Optionally can add `num_teams` and `thread_limit` to the omp target pragma.
  - **Memory management:** CUDA UVM for simplicity

```
#if defindded (GRID_NVCC) || defined (OMPTARGET_MANAGED)
    if ( ptr == (_Tp *) NULL ) auto err =
cudaMallocManaged((void **)&ptr,bytes);
```

- **Results on Cori-GPU with NVIDIA V100**
  - CUDA implementation as comparison (compiled with nvcc, cuda 11)
  - LLVM/Clang compiler two versions tested (mainline 9/25/2020 and 01/17/2021)
  - **gcc:** gcc/10-devel-omp_20201218
  - **cce:** Cray cce/11.0.1



OpenMP Offloading w/ Map

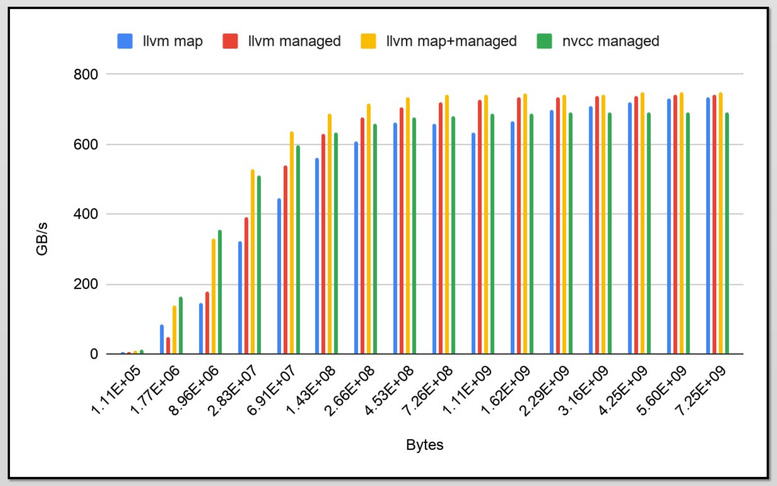
- Using `cudaMallocManaged` is not portable to other GPUs.
- Best to replace it with OpenMP pragmas or APIs for portability.
- Use **`declare mapper`** to simplify data mapping of complex objects

```
#pragma omp declare mapper(decltype(xv) x)
map(x._odata[0:x.size()]) map(x)
```

- Use **`omp target enter/exit data`** for data movement

```
#pragma omp target enter data map(alloc:zv) map(to:xv)
map(to:yv)
```

- So far only works with mainline LLVM (last tested with llvm/12.0.0-git_20210117): `_odata` is still a composite object.
- **Results on Cori-GPU with NVIDIA V100**
  - **llvm map**: compiled with LLVM with the above data mapping
  - **llvm managed**: use cudaMallocManaged without explicit data mapping
  - **llvm map+managed**: use cudaMallocManaged with data mapping
  - **nvcc managed:** CUDA version with cudaMallocManaged compiled with nvcc



Acknowledgments

- This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.
- Part of this work was performed during the OpenMP Hackathons organized by SOLLVE in 2020 and 2021. M.L. thanks the mentors at the hackathons for their help and advice, in particular Johannes Doerfert (ANL) and Rahulkumar Gayatri (LBNL).