Lattice QCD GPU Inverters on ROCm Platform

Yujiang Bi, Zhaofeng Li, Ming Gong, Peng Sun, Yi Xiao, Yibo Yang

Institute of High Energy Physics, CAS

2019.11.07 Adelaide, Australia

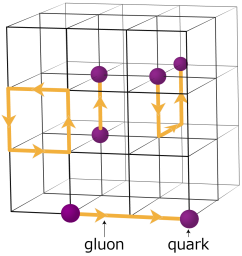


Outline

A Glance of Lattice QCD

Quantum Chromodynamics (QCD)

- A theory describing interactions between quarks and gluons.
- Asymptotic freedom, quark confinement and chiral symmetry spontaneous breaking.
- Coupling constant $\alpha_s \to \infty$ as Energy E $\to$ 0.
- Non-perturbative method required!



Lattice QCD

- A Non-perturbative QCD calculation framework.
- Continuous space-time to discrete space-time.
- Monte Carlo simulating calculation.
- Huge complex matrices: $4 \times 3 \times 3 \times T \times L^3 \approx 9G$, T=L=64.
- Massive computation resource required $\to$ CPU + GPU heterogeneous computing.

GPU Accelaration Framework

QUDA

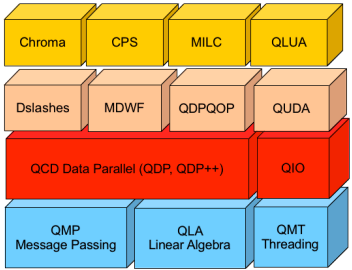
- A widely used GPU framework for LQCD.
- Supports MPI + OpenMP + CUDA.
- Works combined with existing programs like Chroma.
- Ver 1.0 released in Summer.

CHROMA

- A Common framework for LQCD.
- Combined with QUDA to using GPU.

GWU-Code

- By χQCD Collaborations for Overlap & Clover fermions.
- Supporting MPI + OpenMP + CUDA.



Lattice QCD with AMD GPU?

Current Frameworks

- OpenCL/OpenACC
  - CL2QCD
- ROCm/HIP
  - ???

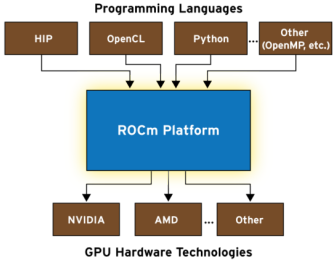
Lattice QCD with AMD GPU?

Current Frameworks

- OpenCL/OpenACC
  - CL2QCD
- ROCm/HIP
  - ???

What is ROCm?

- Open-sourced: Radeon Open Compute Platform.
- Born at 2016: under rapid development.
- Hyper Scale: Supports AMD & NVIDIA GPU.
- Language: **HIP**, **HCC** and OpenCL.
- Almost the Same APIs and structures as CUDA.
- Possible to port current CUDA programs.



Porting QUDA to ROCm?

An Alternative Option besides Nvidia

- Playing Lattice QCD on AMD Exscale Computers.
- Cheaper price and acceptable performance.

Easy to Port CUDA Programs !?

- Automatic porting tools: hipify-perl, hipify-cmakefile …
- Official porting guide - HIP Porting Guide
- Ported programs: Caffee, Tensorflow, PyTorch …

A New Framework Based on ROCm ?

- Time-consuming and may unnecessary.
- Keep the same interface with Chroma.

Our Goal?

Play with Lattice QCD on ROCm platform

1. Porting QUDA (and GWU-Code?) to ROCm with HIP.
2. Correctness validation on ROCm.
3. Optimizing Performance on ROCm.
4. Building QUDA Using HIP on both AMD/NVIDIA GPU?

Porting Procedure

Experience

- HIP Porting Guide and HIP Programming Guide.
- Automatic porting tools: hipify-perl, hipify-clang …
- Ported cases: Caffee, Tensorflow, PyTorch …

Porting Procedure

- Porting source codes using hipify tools.
- Replacing the rest CUDA APIs and structure.
- Adding missing libs such as **AMDDeviceLibs**.

Dependent Libs

CUDA	cublas	cub	thrust	cufft	curand	cusparse	Eigen
ROCm/HIP	hipblas	rocprim	hipthrust	rocfft	hiprand	hipsparse	Eigen

Porting Obstacles

ROCm/HIP

- A young platform and under fast development.
- New problems may raise with new ROCm/HIP.
- Un-implemented APIs, types and structures: cublasCgetriBatched.
- PTX and inline assembly, using C or GAN codes.
- Thrust not self-consistent.

QUDA

- BIG: 300,000 lines and heavily depends on CUDA.
- Depends on Eigen partly Supporting HIP.
- Long time to compiling on ROCm.
- Default argument type omitted in templates …

Since 2018.10

Porting Obstacles

- Porting tools don’t cover all cuda terms.
- Different argument order for some APIs.
- Some functions not implemented in HIP: **rsqrt()**.
- Some problems related to clang:
  - **__float128** not supported.
  - **1/complex<double>** leads to error.
  - **std::__sso_string** not found.
  - …
- Undefined references in libquda.so.

Porting Obstacles

Thrust Self-consistent

```
In file included from /opt/rocm/include/thrust/system/detail/generic/temporary_buffer.h:50:
/opt/rocm/include/thrust/system/detail/generic/temporary_buffer.inl:38:42: error: no template named 'malloc'
      in namespace 'thrust'; did you mean simply 'malloc'?
  thrust::pointer<T,DerivedPolicy> ptr = thrust::malloc<T>(exec, n);
                                         ^~~~~~~~
```

QUDA - template argument type omitting

```
// definition with full template type argments
template <int mu, int nu, typename Float, typename Arg>
__device__ __forceinline__ void computeFmunuCore(Arg &arg, int idx, int parity)
// calling with some argments omitted
case 0: computeFmunuCore<1,0,Float>(arg, x_cb, parity); break;
case 1: computeFmunuCore<2,0,Float>(arg, x_cb, parity); break;
```

Porting Status

QUDA Porting Status

- Porting QUDA completed:
  - Eigen: missing device or host functions added.
  - texture: limited supported and deprecated.
- Compiling completed
  - compiling time: from 4 hours to 1 hour.
  - hacked compiling progress.
  - shared/static libs and built-in test cases.

The first step in a long march!

Basic Tests

Platform

- Hardware: AMD Radeon VII.
- System: Ubuntu.
- Compiler: hipcc, clang 9.0

Basic Tests

- SU3 test: basic SU3 matrix operations
- smearing: smearing operation on gauge fields
- dslash_wilson: one of core operations for Lattice fermions.

Problems

1. Works on small lattice no larger than 32^4.
2. Passing argument in device/global functions.

QUDA Performance

- Validated: compared by results from CPU and GPU.
- Not enough benchmark cases so far.
- Performance depends on lattice size.
- dslash_wilson:
  1. $\approx$ 500 GFlops: 32^4.
  2. $\approx$ 300 GFlops: $32^2 \times 16^2$.
  3. $\approx$ 200 GFlops: 20^4.
- Low performance compared to CUDA ($\approx$ 900 GFlops: 32^4).

How far We Reached

Summary

- Porting and compiling completed.
- Basic test passed: su3_test, smear, dslash_wilson
- dslash_wilson performance :
  - 500 GFlops for 32^4.
  - 300 GFlops for $16^2 \times 32^2$

Problems

- Low QUDA performance.
- Thrust not Self-consistent.
- Necessary but un-implemented APIs.
- Keep pace with the QUDA upstream (?).

Where We Are Going

Outlook

- Test on larger lattice: $32^3 \times 64$, 64^4 ...
- Test combined with Chroma.
- Optimizing QUDA Performance.
- Un-implemented APIs like cublas APIs:
  - Waiting for HIP to implement.
  - Perform LU decomposition on CPU.
  - Find alternative solution: Magama.

Vision

- Generating large lattice for production.
- Compiling QUDA on ROCm and CUDA with the same codes.
- Pushing HIP version to the official repo.

Where We Are Going

Outlook

- Test on larger lattice: $32^3 \times 64$, 64^4 ...
- Test combined with Chroma.
- Optimizing QUDA Performance.
- Un-implemented APIs like cublas APIs:
  - Waiting for HIP to implement.
  - Perform LU decomposition on CPU.
  - Find alternative solution: Magama.

Vision

- Generating large lattice for production.
- Compiling QUDA on ROCm and CUDA with the same codes.
- Pushing HIP version to the official repo.

A HIP-backend branch of QUDA since 2019 Sep.

Any Question?

Thanks!