A QUDA-branch to compute disconnected diagrams in GPUs

Alejandro Vaquero

Computational-based Science and Technology Research Center (CaSToRC) at The Cyprus Institute

In collaboration with:

Constantia Alexandrou, CaSToRC and University of Cyprus
Giannis Koutsou, CaSToRC at The Cyprus Institute

Kyriacos Hadjiyiannakou, University of Cyprus

Alexei Strelchenko, Scientific Computing Division, FermiLab

August 2^{nd}, 2013

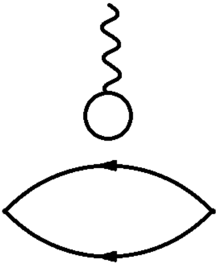
The Cyprus Institute

CaSToRC

Outline

- **Disconnected diagrams: the computational challenge**
- **Software details**
  - The Truncated Solver Method (TSM)
  - The One-End Trick for twisted mass fermions
  - Spin, color and time dilution
  - The Hopping Parameter Expansion (HPE)
  - Integration in QUDA
- **Conclusions and future plans**

Disconnected diagrams: the computational challenge



$L(x) = \mathrm{Tr}\left[\Gamma M^{-1}(x;x)\right]$

- **For the disconnected we need the all-to-all propagators**
  - Must calculate inverse of the fermionic matrix
  - Size $N \times N$ with $N \sim 10^6 - 10^8$
- **Stochastic techniques**

$$M\left|s_j\right\rangle = \left|\eta_j\right\rangle$$

$$M_E^{-1} := \frac{1}{N}\sum_{j=1}^{N}\left|s_j\right\rangle\left\langle\eta_j\right| \approx M^{-1}$$

- **Error decresases as $1/\sqrt{N}$**

Implementation of the different methods: The TSM and the one-end trick

- **The Truncated Solver Method** — Bali, Collins, Schäffer 2007
  - We truncate the solver in $M\left|s_j\right\rangle = \left|\eta_j\right\rangle$
  - Cheap and biased prediction
  - We correct the bias *stochastically*

$$M_E^{-1} := \frac{1}{N_{HP}} \sum_{j=1}^{N_{HP}} \left(\left|s_j\right\rangle \left\langle\eta_j\right|_{HP} - \left|s_j\right\rangle \left\langle\eta_j\right|_{LP} \right) + \frac{1}{N_{LP}} \sum_{j=N_{HP}+1}^{N_{HP}+N_{LP}} \left|s_j\right\rangle \left\langle\eta_j\right|_{LP}$$

  - Efficiency depends on quark mass
- **The One-End Trick** — Foster, Michael 1998; McNeile, Michael 2006
  - For twisted mass fermions, $\sum X \left(M_u^{-1} - M_d^{-1}\right) = -2i\mu \sum_r \left\langle s^\dagger \gamma_5 X s \right\rangle_r$
  - Also for the sum, $\sum X \left(M_u^{-1} + M_d^{-1}\right) = 2 \sum_r \left\langle s^\dagger \gamma_5 X \gamma_5 D_W s \right\rangle_r$

Implementation of the different methods: Dilution and HPE

- **Regarding dilution, only time-dilution implemented so far** Bernardson et al. 1993
- **Regarding the HPE: For twisted mass fermions,** Foster, McNeile, Michael 1999

$$M_u^{-1} = B - BHB + (BH)^2 B - (BH)^3 B + (BH)^4 M_u^{-1}$$

$$B = (1 + i2\kappa\mu a\gamma_5)^{-1} \qquad H = 2\kappa D$$

- **Both methods can be combined without a noticeable impact in performance ($+0.15$ s)**

Integration of TSM in QUDA

- **GPU+QUDA are perfect candidates for evaluating TSM inversions**
  - QUDA provides mixed precision solvers
  - A single GPU in a HP double/single inversion $\approx$ 100GFlops
  - A single GPU in a LP double/half inversion $\approx$ 180GFlops
- **Most inversions are LP (24 HP vs 524 LP in our computations)**
- **We implemented methods to perform HP and LP inversions**
- **Unfeasible to store 500+ propagators, only store contractions**
- **We developed a clever storage procedure for contractions**

Storage

- **Naive contraction storing**
  - 160MB per source (text) $\times \approx$500LP $=$ 80GB per conf ($32^3 \times 64$)
- **Switch to binary**
  - Reduces storage requirements in 70%, not enough yet
- **Power-of-two storing method**
  - Storing several contractions together in a power-of-two fashion reduces the storage to a few hundred MB per conf
  - Storage requirements decrease from N to $\log_2 N$

Prop.C001.Bin, Prop.C002.Bin, Prop.C004.Bin, Prop.C008.Bin, Prop.C016.Bin, Prop.C032.Bin...

- **Immediate reconstruction**
  - 25 sources $=$ Prop.C001.Bin $+$ Prop.C008.Bin $+$ Prop.C016.Bin

Contractions in QUDA

- **The contraction kernel is a big scalar product**
- **The GPUs are an excellent platform to perform contractions**
  - A contraction involves $O(V)$ parallel products to be performed
  - A single fermi GPU can handle thousands of parallel threads
  - Each thread calculates the trace in color of each point
  - Up to $\approx$ 300GFlops per GPU in double precision, peak 500GFlops ($\approx$ 600GFlops in single precision, peak 1TFlop)
- **Our output is the contraction per point**
  - Advantage: Direct transfer to cuFFT for Fourier transform
  - Disadvantage: Large memory requirements
    Time-dilution $\longrightarrow$ OK
    One-End trick $\longrightarrow$ KO

Contractions in QUDA

- **The contraction kernels give results for a general Γ structure**
- **Remember**
  Stochastic source $\rightarrow |\eta_j\rangle$, **Inverted source** $\rightarrow |s_j\rangle$
  $M|s_j\rangle = |\eta_j\rangle$
- **Exterior product in Dirac space** $|s_j\rangle\langle\eta_j|_{\mu\nu}$, $|s_j\rangle\langle s_j|_{\mu\nu}$

$$\mu\nu = 03 \rightarrow \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \mu\nu = 12 \rightarrow \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\mu\nu = 21 \rightarrow \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \mu\nu = 30 \rightarrow \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \ldots$$

- **So we can reconstruct any general gamma structure (scalar, vector and tensor operators)**

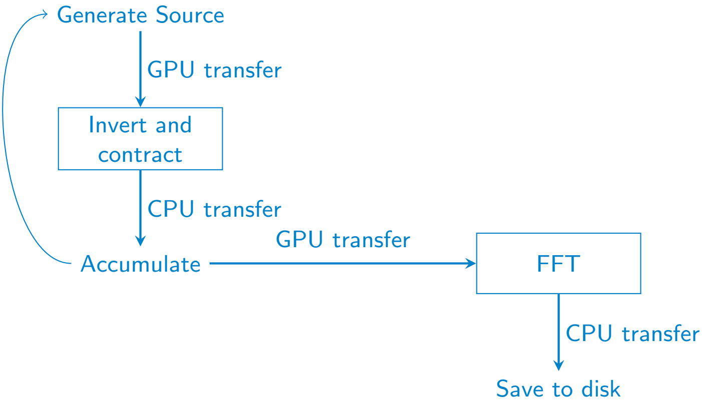
Integration of disconnected code in QUDA and limitations

- **We also developed a covariant derivative operator, to allow for one-derivative insertions**
  - Unfortunately is not optimal yet, and contraction time raises to several seconds
  - Also, can't calculate one-derivative insertions with time-dilution
- **Our code is compatible with the multiGPU implementation of QUDA through MPI**
  - But at this moment only splitting on the time direction is supported
  - QUDA suffers from a large performance impact when splitting on X, Y or Z
  - No QDP/QMP support included yet

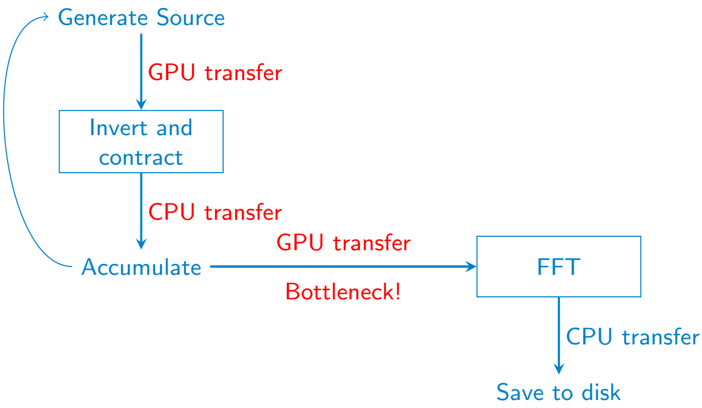
Integration of disconnected code in QUDA and limitations

- **We wrote interfaces for all these methods**
  - INPUT: Random source
  - OUTPUT: Contraction for all momenta and a general Γ insertion
- **CPU code to generate Z_4 noise vectors with RANLUX (gsl) is also included**
- **The Fourier Transform is performed on GPUs with cuFFT**
  - FFT time negligible! The momenta we get is limited by storage and IO time
  - Supports multiGPU only when splitting on the T direction

Integration of disconnected code in QUDA and limitations



Integration of disconnected code in QUDA and limitations



Solving bottlenecks

- **GPUs rely on fast memory, memory transfers GPU/CPU degrade performance**
- **Must reduce memory transfers to/from host as much as possible!!**
  - Generate source on GPU

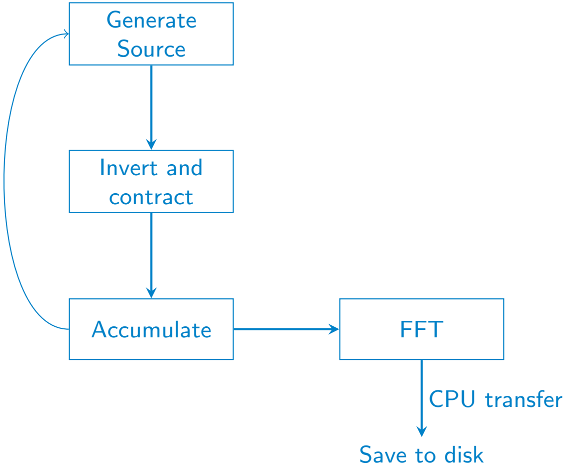
    Piece of cake
  - Accumulate on GPU

    Ok with time-dilution

    Ok with only local one-end trick

    Memory constraints with one-derivative and one-end trick

Solving bottlenecks



Conclusions

- **Accelerators are suitable for computing disconnected diagrams**
- **The use of GPUs displaces the main problem of disconnected computation**
  - The stochastic nature of our estimation of the inverse is not the main problem any more
- **A library makes very easy the GPU implementation**
- **The disconnected diagrams are becoming accessible**

Future plans

- Implement source generation in GPUs by using cuRAND
- Optimize contractions and the covariant derivative operator
- Solve bottleneck on storage/accumulation of contractions
- Allow splitting in the x, y, z direction
- Allow dilution in color/spin
- Allow any regularization supported in QUDA
- Allow QMP/QDP for multiGPU
- Look for integration with master branch

Current branch

- The current branch of the disconnected package can be found at

https://github.com/lattice/quda/tree/discLoop

This presentation has been funded by the Research Promotion Fundation under project ΠΡΟΣΕΛΚΥΣΗ/ΝΕΟΣ/0609/16