LANDAU GAUGE FIXING ON GPUS AND STRING TENSION*

Nuno Cardoso, Pedro Bicudo

CFTP, Departamento de Física, Instituto Superior Técnico
Universidade Técnica de Lisboa
Av. Rovisco Pais, 1049-001 Lisbon, Portugal

Paulo J. Silva, Orlando Oliveira

CFC, Departamento de Física, Faculdade de Ciências e Tecnologia
Universidade de Coimbra
3004-516 Coimbra, Portugal

(Received September 3, 2012)

We explore the performance of CUDA in performing Landau gauge fixing in Lattice QCD, using the steepest descent method with Fourier acceleration. The code performance was tested in a Tesla C2070, Fermi architecture. We also present a study of the string tension at finite temperature in the confined phase. The string tension is extracted from the colour averaged free energy and from the colour singlet using Landau gauge fixing.

DOI:10.5506/APhysPolBSupp.5.1135
PACS numbers: 11.15.Ha, 12.38.Gc

1. Introduction

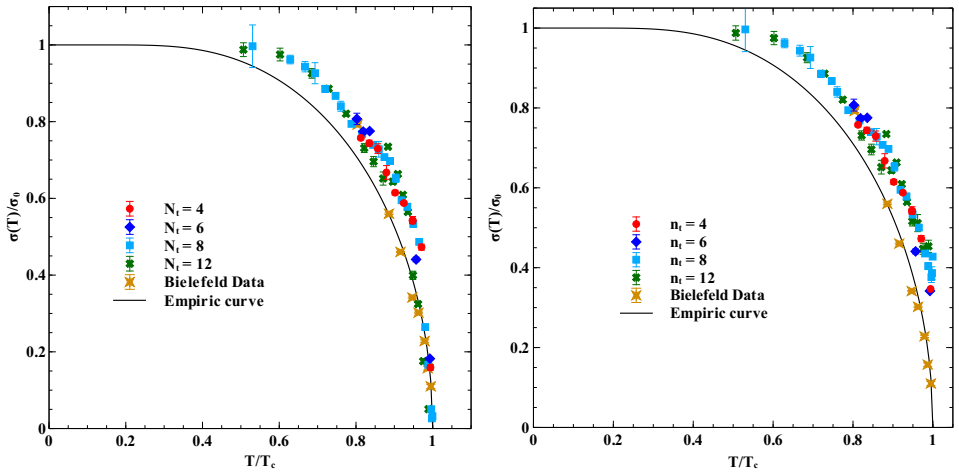
The string tension $\sigma(T)$ is a relevant order parameter to study confinement. While above the deconfinement temperature T_c the simplest order parameter is the Polyakov loop, below T_c the expectation value of a single Polyakov loop vanishes. Below T_c, to study the decrease of confinement with T a new order parameter must be used, and we use the string tension.

The existing lattice QCD results for the string tension critical curve have been computed by the Bielefeld group [1] and Cardoso *et al.* [2], Fig. 1, who have studied in detail the heavy quark potential, from the colour average free

* Presented by N. Cardoso at the Workshop "Excited QCD 2012", Peniche, Portugal, May 6–12, 2012.

energy, at finite temperature in the confined phase. Their results confirmed a first order deconfinement phase transition, as expected for SU(3). Bicudo [3] compared the string tension points obtained by Bielefeld group [1] with different order parameter curves and found empirically that the ferromagnetic critical curve is the one closer to the Bielefeld data, Fig. 1.

In Fig. 1, we show the results for the string tension at finite temperature, for more details see [2]. Near the phase transition, it is necessary to analyse the histogram of the Polyakov loop history, if a double peak structure is found in the histogram then there are configurations in the wrong phase. In Fig. 1 (a) and Fig. 1 (b), we show the results with and without those contaminated configurations, respectively.



(a) (b)

Fig. 1. String tension, σ/σ_0, as a function of temperature for lattices $48^3 \times N_t = 4, 6, 8, 12$, see [2], combined with the results from the Bielefeld group, [1]. The black line corresponds to the ferromagnet magnetization $M/M_{\rm sat}$ critical curve, [3]. (a) string tension with contaminated configurations near the phase transition and (b) string tension with removed contaminated configurations.

In this paper, we will study the string tension extracted from the singlet energy

$$e^{-F_{\rm singlet}(r,T)/T+C} = \tfrac{1}{3}\left\langle \mathrm{Tr}\left(W(\vec{r})\,W^{\dagger}(0)\right)\right\rangle , \qquad (1)$$

where $W(.)$ is the temporal Wilson line. However, this is not gauge invariant and, therefore, we must fix the gauge. Here, we will use the Landau gauge fixing using the steepest descent method with Fourier acceleration, [4, 5]. In order to improve the signal-to-noise-ratio, we apply the multi-hit method before the Landau gauge fixing.

In Sec. 2 we discuss the CPU and GPU implementation of the Landau gauge fixing algorithm. In Sec. 3, we show the benchmark results and the string tension results in the Landau gauge extracted from the singlet energy. In Sec. 4, we conclude.

2. Parallel implementation of the Landau gauge fixing

2.1. CPU implementation

The MPI parallel version was implemented in C++, using the machinery provided by the Chroma library [6]. The Chroma library is built on top of QDP++, a library which provides a data-parallel programming environment suitable for Lattice QCD. The use of QDP++ allows the same piece of code to run both in serial and parallel modes. For the Fast Fourier transforms, the code uses PFFT, a parallel FFT library written by Pippig [7]. Note that in order to optimize the interface with the PFFT routines, we have compiled QDP++ and Chroma using a lexicographic layout.

2.2. GPU implementation

For the parallel implementation of Landau gauge fixing on GPUs [8], we used version 4.1 of CUDA. For the 4D dimensional lattice, we address one thread per lattice site; using 3D thread blocks, we only need to reconstruct the other lattice index inside the kernel. Although CUDA supports up to 3D thread blocks [9], the same does not happen for the grid, which can be up to 2D or 3D depending on the architecture and CUDA version. Nevertheless, the code is implemented with 3D thread blocks and for the grid, we adapted the code for each situation. We use the GPU constant memory to put most of the constants needed by the GPU, like the number of points in the lattice, using `cudaMemcpyToSymbol()`. To store the lattice array in global memory, we use a SOA type array as described in [10]. The main reason to do this is due to the implementation of the FFT algorithm. The FFT is applied for all elements of $\Delta(x)$ matrix separately. Using the SOA type array, the FFT can be applied directly to the elements without the need of copying data or data reordering. For the Fourier transforms, the code uses CUDA CUFFT [11]. Since there is no direct support for 4D FFTs, we will use 2D plus 2D FFTs with `cufftPlanMany()`.

In order to reduce memory traffic, we can use the unitarity of SU(3) matrices and store only the first two rows (12 real numbers) and reconstruct the third row on the fly when needed, instead of storing it.

3. Results

In this section, we show the benchmarks for the steepest descent Fourier accelerated code for Landau gauge fixing in lattice QCD and the results for the string tension in the Landau gauge extracted from the singlet energy.

The benchmark runs using the MPI implementation were performed on the Centaurus cluster at the University of Coimbra. The Centaurus has 8 cores per node, each node has 24 GB of RAM, with 2 intel Xeon E5620@2.4 GHz (quad core) and has a DDR Infiniband interconnecting the various nodes. The benchmark runs on GPU were performed at Instituto Superior Técnico on a NVIDIA Tesla C2070, Table I, and using version 4.1 of CUDA.

TABLE I

NVIDIA's graphics card specifications used in this work.

NVIDIA Tesla C2070			
Number of GPUs	1	Global memory	6 GB
CUDA Capability	2.0	Mem. bandwidth (ECC off)	148 GB/s
Multiprocessors (MP)	14	Shared mem. (per SM) KB	48 or 16
Cores per MP	32	L1 cache (per SM) KB	16 or 48
Total # of cores	448	L2 cache (chip wide)	768KB
Clock rate	1.15 GHz	ECC support	yes

3.1. Performance results

In Fig. 2, we show the GPU performance, in GFlops, of the algorithm using a 12 parameter reconstruction and the full (18 number) representation in single and double precision. The GPU memory access using the L1 and L2 cache and the texture memory was compared. The best performance is achieved using texture memory and the 12 real number parametrization.

The CPU MPI implementation shows a good linear scaling against the number of computing nodes, Fig. 3. We test the GPU (using 12 real parametrization, texture memory and ECC off), and the CPU performance for a 32^4 lattice volume and for $\beta = 5.8, 6.0, 6.2$ in double precision. In order to have the same performance as the GPU, the CPU MPI implementation requires 32 computing codes, *i.e.*, 256 cores.

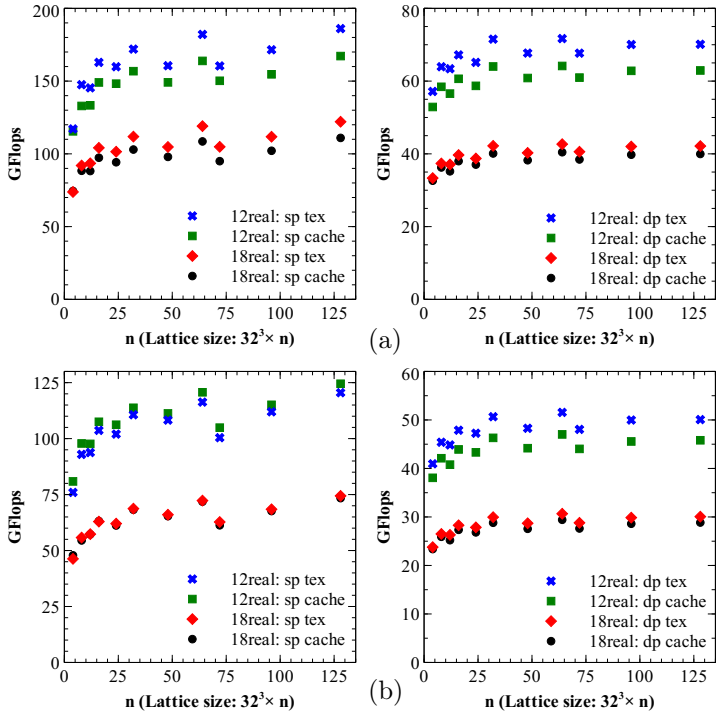


Fig. 2. Performance in GFlops. (a) with ECC Off and (b) with ECC On. sp: single precision, dp: double precision, 18real: full SU(3) matrix, 12real: 12 real parametrization, tex: using texture memory and cache: using L1 and L2 cache memory.

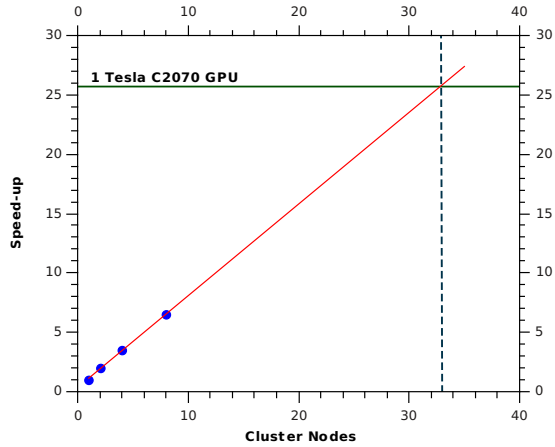
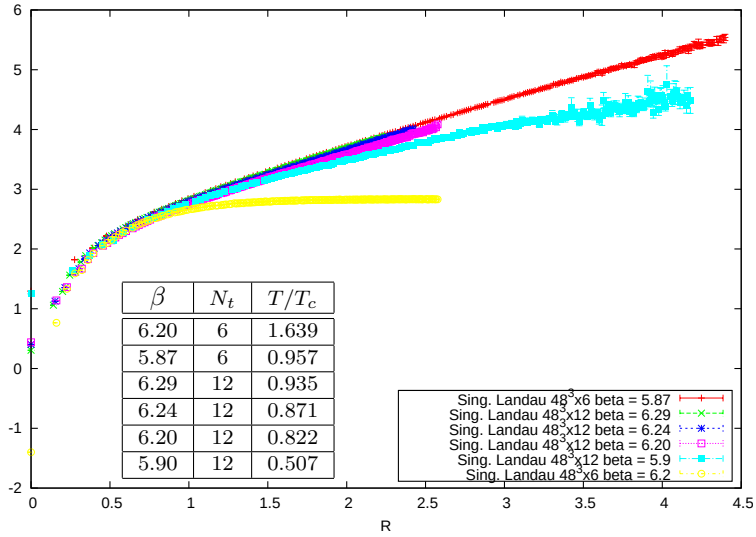


Fig. 3. Strong scaling CPU tested for a 32^4 lattice volume and comparison with the GPU for the best performance, 12 real number parametrization, ECC Off and using texture memory in double precision.

3.2. String tension in the Landau gauge

Fig. 4 shows the results for the singlet energy in the Landau gauge. The singlet energy is fitted with the potential $a_0 - a_1/r + a_2 r$, where a_1 is fixed to the Lüscher Coulomb $\pi/12$ term and a_2 provides $\sigma(T)/\sigma_0$. For $T > T_c$, the string tension is zero and in agreement with the colour average free energy. However, for $T < T_c$, the string tension, $\sigma(T)/\sigma_0$, show a temperature independent behaviour, *i.e.*, $\sigma \sim 0.7\sigma_0$. The string tension extracted from the colour singlet in Landau gauge is constant and temperature independent in the confined phase.



β	N_t	T/T_c
6.20	6	1.639
5.87	6	0.957
6.29	12	0.935
6.24	12	0.871
6.20	12	0.822
5.90	12	0.507

Fig. 4. Singlet energy as function of the distance in units of the zero temperature string tension.

4. Conclusions

From all the runs using a C2070 Tesla GPU, peak performance was measured as 186/71 GFlops for single/double precision. From the performance point of view, a run on a single GPU delivers the same performance as the CPU code when running on 32 nodes (256 cores), if one assumes a linear speed-up behavior.

In the Landau gauge, the string tension, at finite temperature, extracted from the singlet energy shows a constant behaviour in the confined phase.

Our GPU code can be downloaded from the Portuguese Lattice QCD collaboration homepage [12].

This work was partly funded by the FCT contracts POCI/FP/81933/2007, CERN/FP/83582/2008, PTDC/FIS/100968/2008, CERN/FP/109327/2009, CERN/FP/116383/2010 and CERN/FP/123612/2011. Nuno Cardoso and Paulo Silva are supported by FCT under the contracts SFRH/BD/44416/2008 and SFRH/BPD/40998/2007, respectively. We would like to thank NVIDIA Corporation for the hardware donation used in this work via Academic Partnership program.

REFERENCES

[1] O. Kaczmarek, F. Karsch, E. Laermann, M. Lutgemeier, *Phys. Rev.* **D62**, 034021 (2000) [arXiv:hep-lat/9908010].

[2] N. Cardoso, P. Bicudo, *Phys. Rev.* **D85**, 077501 (2012) [arXiv:1111.1317 [hep-lat]].

[3] P. Bicudo, *Phys. Rev.* **D82**, 034507 (2010) [arXiv:1003.0936 [hep-lat]].

[4] C. Davies *et al.*, *Phys. Rev.* **D37**, 1581 (1988).

[5] O. Oliveira, P. Silva, *Comput. Phys. Commun.* **158**, 73 (2004) [arXiv:hep-lat/0309184].

[6] R.G. Edwards, B. Joo, *Nucl. Phys. Proc. Suppl.* **140**, 832 (2005).

[7] M. Pippig, PFFT — An extension of FFTW to Massively Parallel Architectures, Chemnitz University of Technology, 09107 Chemnitz, Germany, Preprint 06.

[8] N. Cardoso, P.J. Silva, P. Bicudo, O. Oliveira, arXiv:1206.0675 [hep-lat].

[9] NVIDIA, NVIDIA CUDA C Programming Guide, version 4.1, Edition 2011.

[10] N. Cardoso, P. Bicudo, arXiv:1112.4533 [hep-lat].

[11] NVIDIA, CUDA Toolkit 4.1 CUFFT Library, 2012.

[12] Portuguese Lattice QCD Collaboration, http://nemea.ist.utl.pt/~ptqcd