Improving many flavor QCD simulations using multiple GPUs

M. Hayakawa^a**, K.-I. Ishikawa**^{*b}†**, Y. Osaki**^{b}**, S. Takeda**^{c}**, S. Uno**^{a}**, N. Yamada**^{de}

^{a} *Department of Physics, Nagoya University, Nagoya 464-8602, Japan*

^{b} *Department of Physics, Hiroshima University, Higashi-Hiroshima 739-8526, Japan*

^{c} *School of Mathematics and Physics, College of Science and Engineering, Kanazawa University, Kakuma-machi, Kanazawa, Ishikawa 920-1192, Japan*

^{d} *KEK Theory Center, Institute of Particle and Nuclear Studies, High Energy Accelerator Research Organization (KEK), Tsukuba 305-0801, Japan*

^{e} *School of High Energy Accelerator Science, The Graduate University of Advanced Studies (Sokendai), Tsukuba 305-0801, Japan*

We accelerate many-flavor lattice QCD simulations using multiple GPUs. Multiple pseudo-fermion fields are introduced additively and independently for each flavor in the many-flavor HMC algorithm. Using the independence of each pseudo-fermion field and the blocking technique for the quark solver, we can assign the solver task to each GPU card. In this report we present the blocking technique for the many-flavor dynamical QCD simulations. We investigate the effect of the blocking and the acceleration with the multiple GPUs for the Schrödinger functional simulations with Wilson $SU(3)$ plaquette gauge action and $N_f = 10$ Wilson fermions. Five pseudo-fermion fields are introduced and the quark solver task is distributed in the ratio of 2:3 to two GPUs. We expect a 40% timing reduction from the single GPU case and have observed a 34% timing reduction in the test simulations.

---

*Speaker.

†*E-mail*: ishikawa@theo.phys.sci.hiroshima-u.ac.jp

1. Introduction

The Large Hadron Collider (LHC) experiment has begun to trap the tail of Higgs boson and to find the evidence of a theory beyond the standard model (SM). Motivated from the unnatural feature of elementary scalar Higgs field, many models beyond the SM have been proposed and studied. The technicolor (TC) model is one of them and describes the origin of electroweak symmetry breaking without introducing elementary scalar particles. The TC is a scaled-up version of QCD, but should have different features from the simple scaled-up QCD. The most promising TC models should have a slowly running (=walking) coupling and a large mass anomalous dimension. The non-perturbative feature of such models has been investigated using lattice technique over the last years [1]. Especially the gauge theories with many fermions, which realize the walking feature, are very attractive. Simulating the lattice gauge theory with many dynamical fermions is, however, a heavier task than that for QCD, since the computational cost is roughly proportional to the number of dynamical fermions. Improving the simulation algorithm with many dynamical flavors becomes more important.

In this paper we present some techniques to improve the HMC algorithm with many dynamical fermions using multiple GPUs. The application of GPU computing to lattice field theory has been introduced by [2] and has been studied extensively in recent years [3, 4, 5]. However the parallel GPU computations with field domain decomposition, which is usually employed for large-scale QCD simulations, are still very challenging because there is no efficient device for direct communication among GPUs. Here we restrict our attention to a single node computation with multiple GPU cards to accelerate the many flavor dynamical QCD simulations on a rather small lattice. We develop a blocked algorithm for the HMC algorithm with many dynamical fermions and test the algorithm for the Schrödinger functional (SF) simulations with the $SU(3)$ gauge theory with ten dynamical fermions (ten-flavor QCD). The performance is compared among the single GPU case, the dual GPU case, and the case without GPU. The next section describes the fermionic part of the HMC algorithm in a general form. We find a parallelism in the force computation of the molecular dynamics of the HMC algorithm. We present the CPU and the GPU implementations in section 3. The results are shown in section 4. The summary is given in the last section 5.

2. Many-flavor simulations

The fermion determinant in the HMC partition function is written by

$$\det[D]^{N_f},\tag{2.1}$$

where D is a lattice Dirac operator and N_f is the number of dynamical fermions. For simplicity we assume that N_f to be an even-number, $\det[D] = \det[D^\dagger]$ holds, and the mass degenerates. The determinant is evaluated by introducing pseudo-fermion fields in the HMC algorithm. There are several ways to introduce the pseudo-fermion fields, additively or multiplicatively. Here we introduce $N_f/2$ pseudo-fermion fields ϕ_i additively.

$$\det[D]^{N_f} = \int \prod_{i=1}^{N_f/2} \mathscr{D}\phi_i^\dagger \mathscr{D}\phi_i e^{-\sum_{i=1}^{N_f/2}|D^{-1}\phi_i|^2}.\tag{2.2}$$

Algorithm 1 MD force computation in sequential version.	**Algorithm 2** MD force computation in blocked version.
1: Pseudo-fermion fields $\{\phi_i\}$ are given.	1: Fields $\Phi = (\phi_1, \phi_2, \cdots, \phi_{N_f/2})$ are given.
2: **for** $i = 1, \cdots, N_f/2$ **do**	2: Solve $DX = \Phi, \rightarrow X = D^{-1}\Phi$.
3: Solve $Dx = \phi_i, \rightarrow x = D^{-1}\phi_i$.	3: Solve $D^\dagger Y = X, \rightarrow Y = D^{\dagger^{-1}}X$.
4: Solve $D^\dagger y = x, \rightarrow y = D^{\dagger^{-1}}x$.	4: **for** $i = 1, \cdots, N_f/2$ **do**
5: Accumulate $F_\mu = F_\mu + f[x, y]$.	5: Accumulate $F_\mu = F_\mu + f[x_i, y_i]$.
6: **end for**	6: **end for**

In this form, $N_f/2$ ϕ_i fields become independent each others. The exponent, together with the gauge action and gauge kinetic term, constructs the effective action of the HMC algorithm as usual. We need to compute the molecular dynamics (MD) force contribution from this effective action (2.2). The general form of the MD force is written as

$$F_\mu(n) = \sum_{i=1}^{N_f/2} F_{\mu,i}(n), \quad F_{\mu,i}(n) = f[D^{-1}\phi_i, D^{\dagger^{-1}}D^{-1}\phi_i], \tag{2.3}$$

where $f[x, y]$ is a function of x and y derived from the derivative of the fermion action $|D^{-1}\phi_i|^2$ with respect to the gauge field. By the additive introduction of the pseudo-fermion fields, the force contribution $F_{\mu,i}$ can be computed independently. We make use of this coarse-grained parallelism to employ multiple GPUs. However it is difficult to assign the task computing fully i-th MD force $F_{\mu,i}$ to a single GPU because the MD force computation contains several steps and the GPU could handle rather simple task to achieve its high efficiency. We extract the solver part from the force computation and parallelize the solver part with multiple GPUs using a blocking technique. In the next section we describe the details of extraction of the solver part and the blocking technique.

3. Implementation

3.1 CPU implementation

In this subsection we describe the blocking technique and the blocked solver algorithm employed for the CPU side computation in detail. Alg. 1 shows the compute step of the MD force in the original (non-blocked) version, where the force is sequentially computed and accumulated. This form is not suitable for the parallel execution of the solver. We reorganize the flavor do-loop in the blocked form as shown in Alg. 2. where the working block vectors, $X = (x_1, x_2, \cdots, x_{N_f/2})$, $Y = (y_1, y_2, \cdots, y_{N_f/2})$ are introduced. The linear equations are organized in the blocked form at the 2nd (3rd) line of Alg. 2.

To solve the linear equations the iterative solvers, such as CG, BiCGStab, GMRES *etc.*, are usually employed. The blocked form of the linear equations could have a benefit from sharing the Krylov subspace among $Dx_i = b_i$ and various blocked iterative solvers have been proposed and explored. We implemented the Global BiCGStab (Gl-BiCGStab) [6] and the Blocked BiCGStab (Bl-BiCGStab) [7] for the CPU side double-precision solver.

Our test problem is $N_f = 10$ QCD with Wilson fermions. We employ the site even/odd preconditioning. The blocked HMC algorithms are compared to the original version which uses the

BiCGStab solver sequentially. On top of the blocking modification we can accelerate the blocked solver using multiple GPUs.

3.2 GPU implementation

We employ GPU cards produced by Nvidia. The details of the GPU architecture is described in [8]. We write the GPU codes in the CUDA language. To achieve high efficiency for the GPU computation the task assigned to the GPU should be as simple as possible and should contain high parallelism. The GPU computation has been applied to the quark solver and high efficiency has been achieved in the case of a single GPU computation using the mixed-precision (or flexible) preconditioner technique [4]. Based on these success we would like to assign one linear equation $Dx = b$ to one GPU even if we have many GPUs.

To accelerate the blocked solver using multiple GPUs we modify the Gl-BiCGStab to have the mixed-precision preconditioning functionality according to the description in the appendix of [9]. We control the multiple GPUs using OpenMP threading on top of the CUDA environment. We have implicitly assumed that the number of pseudo-fermion fields equals to the number of GPUs until now. If this is not the case, some GPUs may solve several linear equations sequentially or may solve nothing, which results in a load imbalance among GPUs. By this modification the single-precision BiCGStab solvers on the GPUs are called in embarrassingly parallel in the preconditioning part of the Gl-BiCGStab. We follow the tuning techniques described in [4] for the GPU solver.

Our test problem is $N_f = 10$ QCD and five pseudo-fermion fields are introduced. The task imbalance occurs since we have only two GPU cards. We distribute the task in the ratio of $2 : 3$ to the two GPU cards, the one GPU solves the two equations sequentially and the other solves the three sequentially. We also test the blocked algorithm with a single GPU where the five equations are solved sequentially on the single GPU card. The speed up (reduction of timing) from the single GPU case to the dual GPU case is expected to be $3/5 = 0.6$. The single-precision coefficient matrix D on the GPUs is also even/odd preconditioned as in the CPU side.

3.3 CPU SSE acceleration

The GPU implementation is highly optimized and the acceleration with GPU from the case without GPU is almost obvious from the previous studies. It is fair for comparison to introduce a more tuned solver for the CPU case. To get the best performance for the CPU case we also implemented the single-precision solver using the SSE intrinsics in the C++ language and employed a more aggressive preconditioner for the Wilson fermions. To use the multi-cores of a CPU efficiently we employed the locally-lexicographical site ordered SSOR (ll-SSOR) preconditioner [10]. To reduce the memory bandwidth requirement the 3rd column of the $SU(3)$ matrices is dropped from the memory and is reconstructed on the fly. The SSE intrinsics are used entirely in the single-precision solver. Instead of the single-precision GPU solver, the single-precision solver with the SSE and the ll-SSOR preconditioner is sequentially called in the preconditioning part of the double-precision Gl-BiCGStab for each pseudo-fermion.

4. Results

We test our algorithm on a PC box which has a single CPU and two GPU cards. The CPU is

Intel's Core i7 920 (4 cores) running at 2.67 GHz, and the GPUs are two Nvidia's GeForce GTX 285 (240 cores) cards. The OS is CentOS 5.2 (Linux). Intel Fortran is used for the HMC algorithm and C++ is partly used for the SSE acceleration. The GPU code is written with the CUDA 2.3. The whole do-loops for lattice site are parallelized with OpenMP.

We test the algorithm described above on the SF setup with the $SU(3)$ plaquette gauge and $N_f = 10$ Wilson quarks action on a 16^4 lattice. The action parameters are $\beta = 4.52$ and $\kappa = 0.15805$ which gives a rather strong coupling. To compare the solver residual history and the timing on the same basis, we rerun the HMC algorithms (with/without various improvements) starting from the same thermalized configuration.

Table 1 shows the test list of the combination of the solver algorithm, the number of GPUs, and the SSE. The timing results are summarized in Table 2. A slight improvement in the solver timing is observed in the case B owing to the localisation of the data access and the better use of data cache by the blocking. The residual history for the case B (Gl-BiCGStab) is very similar to that for the case A and no improvement on the iteration count is observed. The case C (Fig. 3) uses the Bl-BiCGStab which shares the Krylov subspace effectively and we observed a 18% reduction for the total iteration count from the case A (Fig. 1). This behavior agrees with [7]. The timing is also reduced by 25%.

Figure 4 shows the result from the case D1. The case D2 also has the same history. The history shows the residual for the single-precision GPU solver called within the Gl-BiCGStab solver. The GPU solver is called three times in this case to achieve the double-precision solutions. The timing is reduced by 87% for the case D1 (single GPU case) and by 92% for the case D2 (dual GPU case) from the original timing as shown in Tab. 2. A factor of ten speed up has been observed in the literature and we also obtain the similar result on the timing for the single GPU case. The speed up from the single GPU case to the dual GPU case is 62%, which is close to the ideal speed up of 3/5 = 60%. The embarrassingly parallel execution of the GPU solver as a preconditioner works well.

As described in the previous section, we also implemented our best code for the CPU solver to make a fair comparison among the CPU and GPU computations. Figure 5 shows the residual history of the SSE single-precision-ll-SSOR preconditioned CPU solver. The iteration count does not match with the previous figures in terms of the floating point number operation since four-iterations are already included in the SSOR preconditioner. The timing is also given in Tab. 2 and we observed a 69% timing reduction. A naive implementation of double-precision solvers is not recommended for the Intel architecture. One must try various improvement techniques from the algorithmic and architectural point of view.

In the 2nd column of Tab. 2 we tabulate the timing for a single trajectory of the HMC algorithm. The timing is reduced by 34% from the Case D1 to the Case D2, and the number deviates a little more from the ideal number 40% according to Amdahl's law.

5. Summary

In this paper, we have shown the blocking technique for the many-flavor dynamical QCD simulations. The blocking can suitably distribute the solver task independently to many GPU cards attached to a single PC box. We have implemented two types of the blocked quark solvers and have applied the mixed-precision technique with the single-precision facility of GPU and CPU. We have

| Case		Solver	# of GPUs	SSE+ll-SSOR
A	Non-Blocked(Original)	BiCGStab	0	None
B	Blocked	Gl-BiCGStab	0	None
C	Blocked	Bl-BiCGStab	0	None
D1	Blocked	Gl-BiCGStab/BiCGStab	1	None
D2	Blocked	Gl-BiCGStab/BiCGStab	2	None
E	Blocked	Gl-BiCGStab/BiCGStab	0	Yes

Table 1: List for test case. A, B, C and E use only CPU. D1 and D2 use GPU(s). D1, D2, and E use the single-precision BiCGStab solver with the mixed-precision technique.
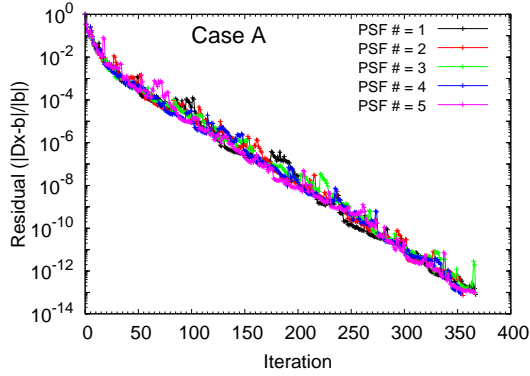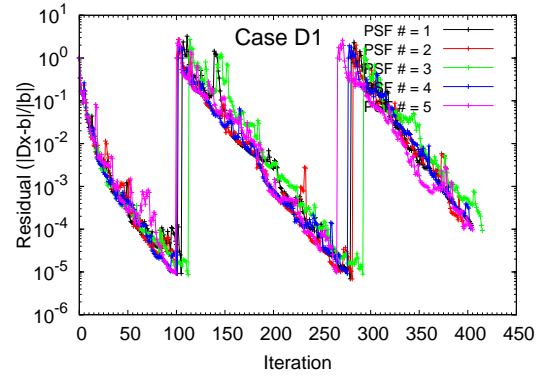

Figure 1: A sample of the residual history in the case A. The five histories are overlaid for the five linear equations.
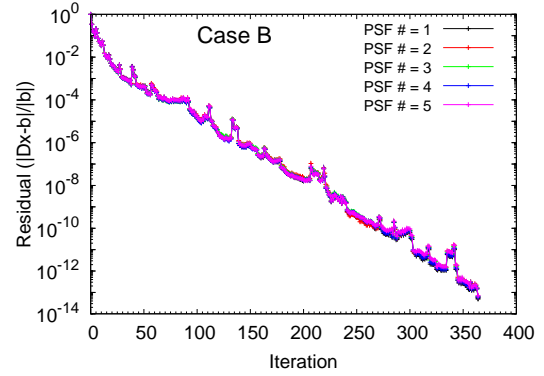
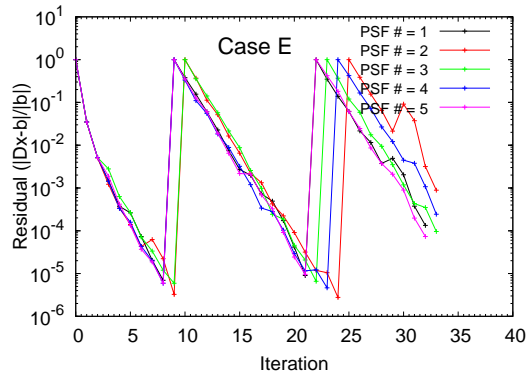**Figure 2:** Same as Fig. 1, but for the case B.
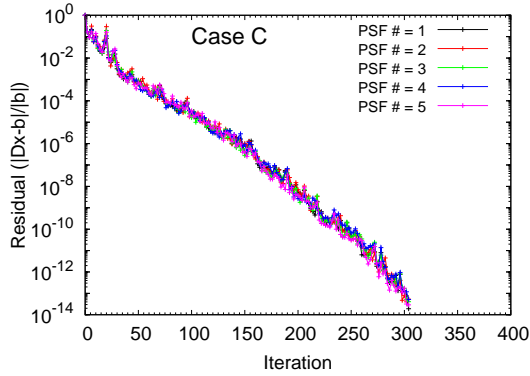
Figure 3: Same as Fig. 1 but for the case C.

Figure 4: Same as Fig. 1, but for the case D1. The case D2 also has the same history.

tested the blocked HMC algorithm for the Schrödinger functional simulations with the plaquette gauge and ten-flavor Wilson quarks. We found the almost ideal speed up for the solver part using dual GPUs. This algorithm has been being used to estimate the walking behavior of the coupling in the SF scheme partly in the strong coupling region and to search a near conformal theory for the TC model [11].

A part of the program development and the numerical simulations have been done on the IN-SAM (Institute for Numerical Simulations and Applied Mathematics) GPU cluster at Hiroshima

Figure 5: Same as Fig. 1, but for the case E.

Case	Averaged solver timing in a trajectory [sec]	HMC timing for a trajectory [sec]
A	30.9(1.0)	4097
B	25.1(0.2)	3346
C	23.1(0.1)	3094
D1	4.10(0.02)	565
D2	2.56(0.01)	374
E	9.53(0.05)	1280

Table 2: Timing comparison. The averaged solver timing is for five linear equations.

References

[1] L. Del Debbio, Lattice 2010 plenary, in these proceedings; R. Sekhar Chivukula, Lattice 2010 plenary, in these proceedings; E. Pallante, PoS **LAT2009** (2009) 015 [arXiv:0912.5188 [hep-lat]]; G. T. Fleming, PoS **LATTICE2008** (2008) 021 [arXiv:0812.2035 [hep-lat]]; and references therein.

[2] G. I. Egri, Z. Fodor, C. Hoelbling, S. D. Katz, D. Nogradi and K. K. Szabo, Comput. Phys. Commun. **177** (2007) 631 [arXiv:hep-lat/0611022].

[3] R. Babich, *et al.*, Lattice 2010, in these proceedings; S. Gottlieb and G. Shi, *ibid*; H.-J. Kim and W. Lee, *ibid*; Y.-Y. Mao, T.-W. Chiu, and K. Ogawa, *ibid*; B. Walk, *ibid*; Y. Osaki and K.-I. Ishikawa, *ibid*.

[4] M. A. Clark, PoS **LAT2009** (2009) 003 [arXiv:0912.2268 [hep-lat]];

[5] M. A. Clark, R. Babich, K. Barros, R. C. Brower and C. Rebbi, Comput. Phys. Commun. **181** (2010) 1517 [arXiv:0911.3191 [hep-lat]]; K. Barros, R. Babich, R. Brower, M. A. Clark and C. Rebbi, PoS **LATTICE2008** (2008) 045 [arXiv:0810.5365 [hep-lat]]; V. Anselmi, G. Conti and F. Di Renzo, PoS **LATTICE2008** (2008) 024 [arXiv:0811.2111 [hep-lat]]; K.-I. Ishikawa, PoS **LATTICE2008** (2008) 013 [arXiv:0811.1661 [hep-lat]].

[6] K. Jbilou, H. Sadok, and A. Tinzefte, Elect. Trans. Numer. Anal. **20**, 119 (2005).

[7] H. Tadano, T. Sakurai, and Y. Kuramashi, JSIAM Lett. **1**, 44 (2009); T. Sakurai, H. Tadano and Y. Kuramashi, Comput. Phys. Commun. **181** (2010) 113 [arXiv:0903.4936 [hep-lat]].

[8] Nvidia CUDA ZONE, [http://www.nvidia.com/object/cuda_home_new.html].

[9] S. Aoki *et al.* [PACS-CS Collaboration], Phys. Rev. D **79** (2009) 034503 [arXiv:0807.1661 [hep-lat]].

[10] N. Eicker, W. Bietenholz, A. Frommer, T. Lippert, B. Medeke and K. Schilling, Nucl. Phys. Proc. Suppl. **73** (1999) 850 [arXiv:hep-lat/9809038]; S. Fischer, A. Frommer, U. Glassner, T. Lippert, G. Ritzenhofer and K. Schilling, Comput. Phys. Commun. **98** (1996) 20 [arXiv:hep-lat/9602019].

[11] N. Yamada, M. Hayakawa, K.-I. Ishikawa, Y. Osaki, S. Takeda and S. Uno, arXiv:1003.3288 [hep-lat]; PoS **LAT2009** (2009) 066 [arXiv:0910.4218 [hep-lat]].