



First application of lattice QCD to Pezy-SC processor

Tatsumi Aoyama¹, Ken-Ichi Ishikawa², Yasuyuki Kimura³, Hideo Matsufuru⁴,
Atsushi Sato⁵, Tomohiro Suzuki⁵, and Sunao Torii³

¹ Kobayashi-Maskawa Institute for the Origin of Particles and the Universe (KMI),

Nagoya University, Japan. aoym@kmi.nagoya-u.ac.jp

² Department of Physics, Hiroshima University, Japan.

ishikawa@theo.phys.sci.hiroshima-u.ac.jp

³ ExaScaler Inc., Japan.

yasuyuki@exascaler.co.jp, torii@exascaler.co.jp,

⁴ Computing Research Center, High Energy Accelerator Research Organization (KEK), Japan,
and Graduate University for Advanced Studies (Sokendai), Japan.

hideo.matsufuru@kek.jp

⁵ PEZY Computing K.K., Japan.

atsushi@pezy.co.jp, tomo_suzuki@pezy.co.jp

Abstract

Pezy-SC processor is a novel new architecture developed by Pezy Computing K. K. that has achieved large computational power with low electric power consumption. It works as an accelerator device similarly to GPGPUs. A programming environment that resembles OpenCL is provided. Using a hybrid parallel system “Suiren” installed at KEK, we port and tune a simulation code of lattice QCD, which is computational elementary particle physics based on Monte Carlo method. We offload an iterative solver of a linear equation for a fermion matrix, which is in general the most time consuming part of the lattice QCD simulations. On single and multiple Pezy-SC devices, the sustained performance is measured for the matrix multiplications and a BiCGStab solver. We examine how the data layout affects the performance. The results demonstrate that the Pezy-SC processors provide a feasible environment to perform numerical lattice QCD simulations.

Keywords: Pezy-SC, Lattice gauge theory, Accelerator, OpenCL

1 Introduction

Pezy-SC processor is a novel new architecture developed by Pezy Computing K. K. that has achieved large computational power with low electric power consumption. Combined with a liquid cooling equipment, it composes “ZettaScaler” system which won the top three ranks of the Green 500 list in June 2015. The Pezy-SC processor is the second generation of the Pezy processor series, and intends to be used in high performance computing. It has been drawing much attention in many areas of research applications. Since a standard usage of the Pezy-SC processor is as an arithmetic accelerators, similarly to GPGPUs, programming techniques developed for the latter are expected to be applicable. A programming environment compatible to OpenCL is provided. Applications to scientific researches have recently been started in several fields. Among them, this paper concerns the application of Pezy-SC processor to lattice QCD simulations in computational elementary particle physics.

QCD (quantum chromodynamics) describes the strong interaction among quarks that are ingredients of protons and neutrons in atomic nuclei. Since the QCD interaction becomes destructively strong at low energy, in other words at long distance, many characteristic features of QCD cannot be calculated analytically based on the perturbation theory which relies on the smallness of the coupling parameter of the interaction. The lattice QCD is a quantum field theory on discretized Euclidean spacetime (lattice). It enables numerical calculation of quantum expectation values by evaluating the path integral applying a Monte Carlo method [9, 4]. The lattice QCD simulations are in many cases only the way to extract precision theoretical calculation of QCD. With recent large computational power, the lattice QCD has been producing more and more precise results, and furthermore being extensively applied to other field theories, *e.g.* in search for a theory beyond the standard model of elementary particle physics.

In this paper, we develop and tune a lattice QCD simulation code on Pezy-SC processors using hybrid parallel system “Suiren” installed at High Energy Accelerator Research Organization (KEK). This paper is organized as follows. Section 2 summarizes the architecture and programming scheme of the Pezy-SC processor. Section 3 describes a hot spot of the numerical simulation of lattice QCD that is to be executed on the device. In Section 4, we implement the lattice QCD code on Pezy-SC processors and test its performance. Section 5 is devoted to our conclusion and outlook.

2 Programming on Pezy-SC processor

We first summarize the architecture of the Pezy processor used in this work. Table 1 summarizes basic specification of the Pezy-SC architecture installed at KEK as the Suiren system.

Pezy-SC is a MIMD many-core processor which possesses 1024 cores. Each core has one instruction decoder and executes 8 threads in units of 4 threads simultaneously. The cores (PEs) are classified into three layers: village (4PEs, L1 cache), city (4 villages, L2 cache), prefecture (16 cities, L3 cache), and 4 prefectures compose one Pezy-SC device. Since the cache coherency is not automated, a program needs to flush data explicitly at each cache level. The numbers of register files for floating-point-numbers per each thread are 32 for single or 16 for double precision.

Each node of the Suiren system contains 8 Pezy-SC processors as well as 2 Intel Xeon E5-2620v2 processors as the host processor. The inter-node network is provided by Infiniband FDRx4 dual port. Novel feature of the Suiren system is its cooling system. The ExaScaler’s submersion liquid cooling system with fluorinert accomplishes highly efficient cooling with low electric power consumption. The Suiren system was ranked the second at the Green 500 in

Core cycle	733 MHz
Number of cores	1024PE
Peak performance	double: 1.5 TFlops, single: 3 TFlops
Device memory	DDR4 32 GB, 1333 MHz
Device memory B/W	153.6 GB/s
Host-device network	PCIe Gen2 \times 8 lanes
Cache	L1: 1 MB, L2: 4 MB, L3: 8 MB

Table 1: Basic specification of Pezy-SC processor installed at KEK (Suiren system).

November 2014 and the third in June 2015. The top and the second ranked systems in June 2015 were the improved version of the system, “ZettaScaler-1.4”, named “Shoubu” at RIKEN and “Suiren-Blue” at KEK.

For the current Pezy-SC architecture, programming scheme is offloading of hot spots of the code to Pezy devices. This is a familiar scheme to the accelerator devices such as GPGPUs and Xeon-Phi, and large part of the wisdom acquired for them also applies to the Pezy-SC processor. Pezy-SC provides a subset of OpenCL as a programming environment (“PZCL”). While PZCL provides most of OpenCL’s run-time API functions, there are several restrictions:

- Online compiling is not available. The device code must be compiled before a program runs.
- Directives and built-in functions of OpenCL C language are not fully supported.
- Several original API functions are added to control the synchronization, cache flushing, and getting IDs of PE and thread, and so on.
- No API to control the local shared memory is available; thus the `local` qualifier for variables is not available.

Although porting of OpenCL codes seems rather straightforward, optimization may not be so depending on applications. It is thus essential to demonstrate the feasibility with a practical implementation and to accumulate implementation techniques. Since the Pezy-SC architecture and PZCL environment are rapidly developing, feedback from the application developers would be useful.

3 Lattice QCD simulations

For the formulation of lattice QCD and a principle of the numerical simulations, there are many textbooks and reviews [9, 4]. Thus we concentrate on solving a linear equation for a fermion matrix, which is in many cases the most time consuming part in the numerical simulations.

A lattice QCD action consists of fermion (quark) fields and a gauge (gluon) field. The latter mediates interaction between quarks and are represented by link variables, $U_\mu(x) \in SU(N_c)$ ($N_c = 3$), where $x = (x_1, x_2, x_3, x_4)$ stands for a lattice site and $\mu=1-4$ is a spacetime direction. In numerical simulations the size of a lattice is finite, $x_\mu = 1, 2, \dots, L_\mu$. The fermion field is represented as a complex vector on lattice sites, which carries 3 components of color and 4 components of spinor. The fermion action is a bilinear of fermion field ψ , $S_F = \sum_{x,y} \bar{\psi}(x) D[U](x,y) \psi(y)$, where $D[U]$ is a fermion operator (matrix).

There are several fermion matrix formulations. Each formulation has its own advantages and disadvantages, and combined with a variety of improvement. As a common feature, a

fermion operator is represented with local interactions, so that the fermion matrix is sparse. As a representative example, we here consider the Wilson fermion operator,

$$D_W(x, y) = \delta_{x, y} - \kappa \sum_{\mu=1}^d [(1 - \gamma_\mu)U_\mu(x)\delta_{x+\hat{\mu}, y} + (1 + \gamma_\mu)U_\mu^\dagger(x - \hat{\mu})\delta_{x-\hat{\mu}, y}] \quad (1)$$

where x, y are lattice sites, $d = 4$ is the space-time dimension, $\hat{\mu}$ is a unit vector along the μ -th axis. γ_μ is 4×4 matrix acting on the spinor degree of freedom. κ is a parameter related to the fermion mass m through the relation $\kappa = 1/2(4 + m)$. Thus D_W is a $4N_c L_x L_y L_z L_t$ dimensional complex matrix.

The Wilson fermion has several improved variants so as to decrease the lattice artifacts. The form of Eq. (1) results in an $O(a)$ lattice artifact in numerical results, where a is the lattice spacing. A popular improvement of the Wilson fermion action is to add a local term that cancel the $O(a)$ effect. This so-called clover fermion matrix is widely used in numerical studies, and later we briefly note the results for this action.

Since the numerical simulation of lattice QCD is based on the Monte Carlo method, one needs to generate ensemble of the gauge field $\{U_\mu(x)\}$ by modifying it by certain update algorithm. The standard algorithm requires to solve a linear equation for the fermion matrix, $D[U]v = b$, at every step of updating the gauge field. Thus large part of simulation time is consumed in the linear equation solver. Since the fermion operator is a large sparse matrix, iterative solvers, such as the Conjugate Gradient (CG) and BiCGStab algorithms, are employed. This is the part which we offload to the Pezy-SC processors.

4 Implementation and performance

In this paper, we perform benchmark tests of the BiCGStab solver for the Wilson fermion matrix in Eq. (1) that is offloaded to the Pezy-SC processors. The implementation is similar to that for GPGPUs and for the offload model of Xeon-Phi processors. There has been a decade of applications of lattice QCD to such accelerators [7, 3] including OpenCL-based applications [1, 5, 6, 10].

Data layout We start with measuring the performance within a single device. In the case of GPUs, coalesced memory access is a key ingredient in optimization. In the case of the Pezy-SC processors, the PEs are organized in hierarchical layers sharing caches, and thus the memory access pattern may affect the performance. We investigate this point with the following four types of data layouts.

Site-packed is a so-called array-of-structure type, in which data are organized so that the internal degrees of freedom (color and spin) are packed in the innermost indices.

Site-major is a so-called structure-of-array type, in which data are represented by a set of vectors of volume length.

PE-thread mixed is a layout in which data are arranged in units of 4 threads, followed by the internal degrees of freedom (color and half-spinor), and the remaining site and spinor indices.

PE-major is a layout in which data are arranged in units of N_{VLEN} PEs, followed by the internal degrees of freedom (color and half-spinor), and the site index indicated by the threads and the remaining PEs.

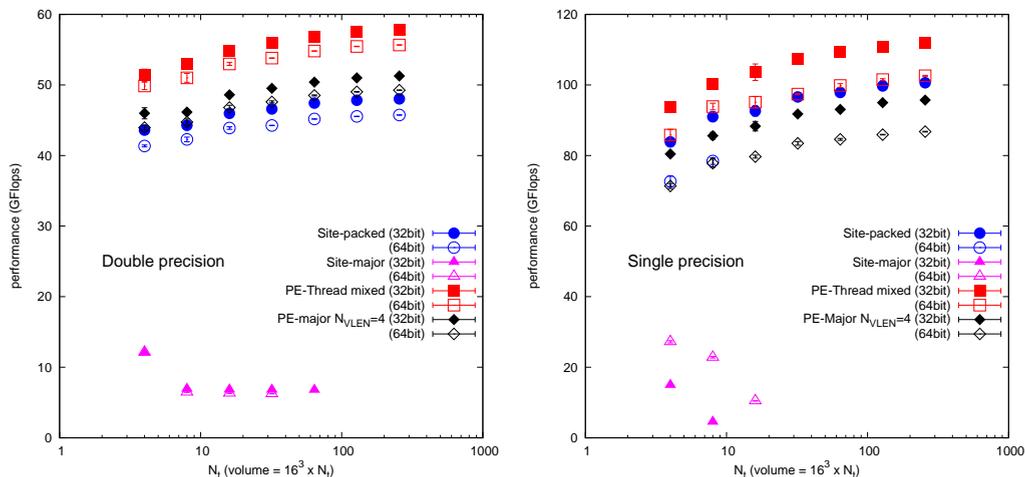


Figure 1: Sustained performance of the Wilson matrix multiplication on $16^3 \times N_t$ lattices with respect to the temporal lattice size N_t for the double (left panel) and single (right) precision floating-point-numbers. Filled and open symbols represent 32-bit and 64-bit addressing modes of the kernel, respectively.

The lattice sites in Cartesian coordinates are serialized into a single index that is mapped to thread and PE indices as $i_{\text{th}} + N_{\text{th}} \cdot i_{\text{PE}}$, where the number of threads per PE is $N_{\text{th}} = 8$, except for the PE-major case in which they are indexed as $(i_{\text{PE}} \bmod N_{\text{VLEN}}) + N_{\text{VLEN}} \cdot (i_{\text{th}} + N_{\text{th}} \cdot (i_{\text{PE}}/N_{\text{VLEN}}))$.

To reduce memory transfer, it is frequently adopted that the third column of $SU(3)$ matrix in the gauge field is reconstructed on the fly from the other two columns, exploiting the relation $v_3 = (v_1 \times v_2)^*$ where $U = (v_1, v_2, v_3) \in SU(3)$. We have applied this procedure and found so far there is no sizable difference in the performance. Therefore, in the following tests this technique is not applied.

Figure 1 shows the performance of the Wilson matrix multiplication on a single Pezy-SC device. The lattice size is taken $16^3 \times N_t$ with varying the temporal lattice extent N_t . The left and right panels show the performance with the double and single precision arithmetics, respectively. In both cases, the PE-thread mixed data layout surpasses the others. Pezy-SC has two addressing modes with 32-bit and 64-bit. The former achieves better performance in general, and this tendency is also seen in our data. The site-packed layout retain 20% less performance than the PE-thread mixed, while the site-major case results in unacceptable performance. We also examined the dependence on the vector length N_{VLEN} of the PE-major type, to find that $N_{\text{VLEN}} = 4$ achieves the best among $N_{\text{VLEN}} = 4, 8, 16$. The plot shows only the $N_{\text{VLEN}} = 4$ case. It may reflect that $N_{\text{VLEN}} = 4$ fits in the size of “village” of the Pezy-SC architecture.

As the Wilson matrix is a large sparse matrix, the memory access is considered as a bottleneck, if the arithmetic operations on each thread is efficiently scheduled. This speculation is demonstrated from the fact that the performance of the single-precision arithmetics almost doubles that of the double-precision arithmetics, and the performance is determined by the bandwidth of the global memory.

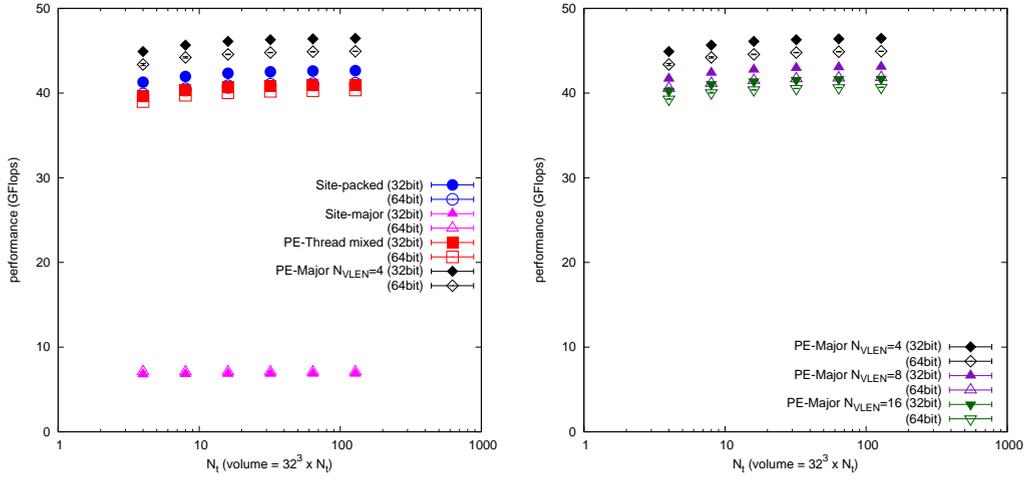


Figure 2: Sustained performance of the Wilson matrix multiplication on $32^3 \times N_t$ lattices with respect to the temporal lattice size N_t for double precision floating-point numbers. The left and right panels compare four types of data layouts and several values of N_{VLEN} for the PE-major layout, respectively. Filled and open symbols represent 32-bit and 64-bit addressing modes of the kernel, respectively.

The byte-per-flop ratio of the Wilson matrix multiplication in Eq. (1) is obtained as follows. The number of the floating-point operations in Eq. (1) amounts to 1392 per each lattice site.¹ Here, we exploited the property that $(1 \pm \gamma_\mu)$ is a projector so that the Dirac degree of freedom can be halved in the multiplication of the matrix U_μ , and that the components of γ_μ are ± 1 or $\pm i$. Assuming no cache effect, on each lattice site, nine complex vectors of $4N_c$ components and eight complex matrix of $N_c \times N_c$ are loaded from the memory, and one complex vector of $4N_c$ components is stored. Namely, for double precision, the amount of data to be transferred is 384×8 bytes. The byte-per-flop ratio therefore turns to be 2.2 and 1.1 for double and single precision, respectively. A sustained bandwidth between the device memory and the core is estimated by $\text{axpy} (\mathbf{y}=\mathbf{a}*\mathbf{x}+\mathbf{y})$ operations on a single Pezy-SC device, and it results in about 50GB/s for the data sizes used in these measurements. Thus, the expected performance derived from the bandwidth and the byte-per-flop ratio is about 23 and 45 GFlops for double and single precision, respectively. As each complex vector is also accessed from neighboring lattice sites, cache may have some effect. The results in Fig. 1 indicates that the obtained performance is rather consistent with the expectation, and that the existence of cache improves the efficiency by a factor 2–2.5. In the following, we concentrate on the case with the double precision arithmetics.

Figure 2 shows the performance on a single Pezy-SC device on a different lattice size in which we take $32^3 \times N_t$. The left panel shows the performance of the double-precision multiplication with varying the temporal extent N_t . On these lattice sizes, the PE-major layout exhibits the best performance, while the PE-thread mixed layout runs rather poorly. This implies that the preferred layout depends on the lattice size, *i.e.* the stride of the data of the neighboring site in the four-dimensional lattice accessed from each thread, which may affect the cache efficiency. Indeed, the single-precision case behaves differently in which the PE-thread mixed achieves the best, and the site-packed layout follows. The right panel shows N_{VLEN} dependence of the

¹We assume the so-called Dirac representation of γ_μ matrices.

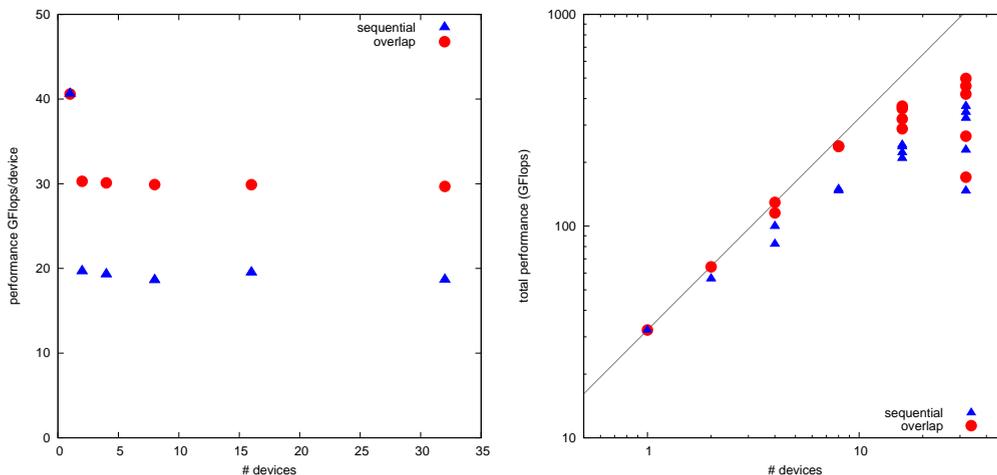


Figure 3: Sustained performance of the Wilson matrix multiplications for multiple Pezy-SC devices. The left panel is a weak scaling plot on $32^3 \times 8n$ lattices where n is the number of devices. The right panel shows strong scaling behavior on a $32^3 \times 64$ lattice. Each point corresponds to a different way to divide the lattice.

PE-major layout. Varying the vector length results in about 10% of the performance deviation, for both single and double precision cases. In the following analyses, we adopt the PE-major data layout with $N_{\text{VLEN}} = 4$.

Multiple device Multiple devices are used to the calculation of a single lattice in which the space-time lattice is divided into a grid, each piece of which is assigned to a device. In this setup, the data transfer between the device processors via the host system introduces another bottleneck. Our code is parallelized with MPI and each MPI process handles one Pezy-SC processor. To quantify the performance of data transfer and the operations inside device individually, we start with the code that executes them sequentially.

Figure 3 shows the sustained performance of the Wilson matrix multiplication on multiple Pezy-SC devices. The results of the sequential version of the code are represented with the blue triangle symbols. The left panel displays the weak scaling behavior on $32^3 \times 8n$ lattices, where n is the number of devices. Since the lattice is divided only in the t -direction, the communication between the MPI ranks occurs with the fixed size of transferred data. As each node accommodates 8 Pezy-SC devices, for $n \leq 8$ all communications are on-node, while for $n > 8$ cases the inter-node communications take part in. The plot shows the plateau in the performance, indicating that the inter-node communication does not decrease the performance. While the result shows good weak scaling behavior, the performance decreases to about 40 % of the case without communication.

The right panel of Fig. 3 shows the strong scaling behavior on a $32^3 \times 64$ lattice. In this case the performance depends not only on the number of the devices but also strongly on how the lattice is divided. It may rely on the buffer sizes for communication between MPI ranks and the number of communication times. In the figure, the vertically scattered points at the same number of devices represent how the performance depends on the way to divide the lattice. For the examples of 32 devices, the best performance is obtained with $(1 \times 1 \times 4 \times 8)$ division which is almost three times faster than $(2 \times 2 \times 2 \times 4)$ division.

Kernel		33.875 GFlops	
	bulk	8.442 msec	
	pack	0.717 msec	
	unpack	1.428 msec	
Data transfer			
	Device to Host	2.039 msec	2.943 GB/s
	Host to Host	3.475 msec	1.727 GB/s
	Host to Device	2.213 msec	2.712 GB/s
Other		~ 1msec	
Total		19.241 msec	18.638 GFlops

Table 2: Profile of the Wilson matrix multiplication.

Table 2 shows a typical profile of the elapsed times consumed in the steps of the Wilson matrix multiplication on the $32^3 \times 64$ lattice with $(1 \times 1 \times 1 \times 8)$ division. The data transfer and the arithmetic operations on the bulk can be carried out simultaneously. The itemized values of the consumed time indicates that it is possible to overlap the communication with the arithmetic operations on the device.

Overlapping communications In Fig. 3, the red circle symbols represent the performance of the Wilson matrix multiplication with overlapping the communication and the arithmetic operations. The left panel, the weak scaling plot, indicates that the sustained performance is almost independent of the number of devices. While the performance increases by 50 % compared to the sequential case, it is still much less than that of the single device result, indicating the performance loss due to multiple kernel calls and other overheads. The right panel of Fig. 3 exhibits that the strong scaling is also improved. The straight line represents the perfect strong scaling, that is, the performance is proportional to the number of devices n . As seen in Table 2, the elapsed time for the communication and the calculation on the bulk are almost comparable at $n = 8$. The performance on the further division ($n > 8$) should be limited by the communication, as implied by the departure from the perfect scaling line.

Linear algebra and the solver algorithm Figure 4 shows the sustained performance of the BiCGStab solver for the Wilson matrix on a $32^3 \times 64$ lattice with respect to the number of devices. The performance of the linear algebraic operations is also constrained by the bandwidth between the device memory and PE, and the byte-per-flop ratio is rather severer than the Wilson matrix multiplication. We measured the performance of the kernels of both 32-bit and 64-bit addressing modes. On single device, the 32-bit kernel fails for an unclear reason, which is now under investigation. At each number of devices, there are several points that correspond to different ways of dividing the lattice. The performance of the solver exhibits good strong scaling up to $n = 8$. It implies that the linear algebraic operations shows better strong scaling behavior.

Further applications As mentioned in Sec. 3, the clover fermion is an improved version of the Wilson fermion matrix in Eq. (1) by adding the so-called clover term that eliminates the leading order lattice artifact. Since the clover term is local to lattice site, it increases intra-device operations without affecting the data transfer between the device and the host. This leads to the increase in the performance of the matrix multiplication. We also apply the

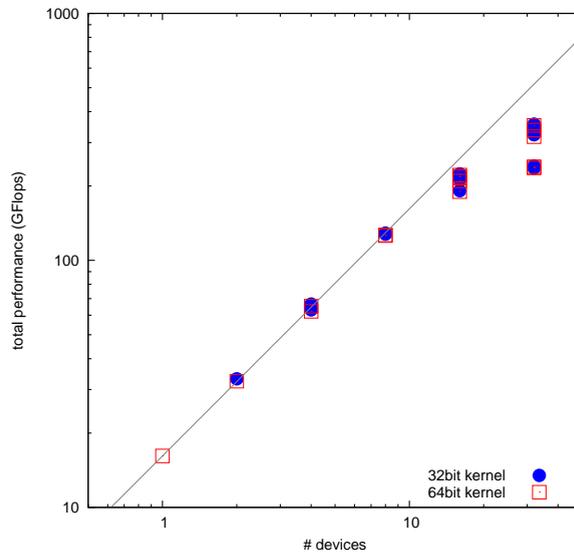


Figure 4: Sustained performance of the BiCGStab solver for the Wilson matrix on a $32^3 \times 64$ lattice with respect to the number of devices. Filled and open symbols represent 32-bit and 64-bit addressing modes of the kernel, respectively.

mixed-precision solver algorithm so that the arithmetic operations on the devices are performed in single-precision. As the result, for $32^3 \times 64$ lattice with 8 devices, the performance of the fermion matrix multiplications and the whole solver algorithm are 37.2 GFlops and 20.0 GFlops per device, respectively. Up to 64 devices, the performance shows good weak scaling behavior.

5 Conclusion and outlook

In this paper, we demonstrated practical applicability of Pezy-SC processors to numerical simulations of lattice QCD. For the Wilson fermion matrix, we measured the performance of processors and examined how the data access patterns affect the performance. Compared to GPGPUs, one drawback of the current Pezy-SC processors is less bandwidth between the device global memory and the cores. This causes less sustained performance than the currently available GPGPUs. When the memory bandwidth is improved in future version of the Pezy-SC architecture, our result shows that Pezy-SC processors are promising devices for lattice QCD simulations as well as the similar types of applications.

For multiple-device setup which is indispensable for productive researches, the bottleneck of performance turns to be the data transfer between the host and the device, and between host MPI ranks. The former would be improved when the PCIe Gen3 becomes available. To reduce the data transfer between MPI ranks, the domain decomposition [8, 11] that localizes the communication may improve the solver algorithms. We are planning to extend our implementation to other types of fermion matrix formulations and linear algorithms.

Acknowledgments

We thank Motoaki Saito, Kei Ishii, Shuntaro Tsunoda (PEZY Computing) and Tadashi Ishikawa (KEK) for their continuous support. The simulation codes used in this work are partially based on CCS QCD benchmark and Bridge++ Lattice QCD code set [2, 12]. The code development and numerical measurements were performed on Suiren system at High Energy Accelerator Research Organization under the support of its Large-scale Simulation Program (14/15-T01, 15/16-T01). This project is partially supported by Joint Institute for Computational Fundamental Science and HPCI Strategic Program Field 5 ‘The origin of matter and the universe’. This work is supported in part by JSPS Grant-in-Aid for Scientific Research (No. 25400284).

References

- [1] M. Bach, V. Lindenstruth, O. Philipsen, and C. Pinke. Lattice QCD based on OpenCL. *Comput. Phys. Commun.*, 184:2042–2052, 2013.
- [2] Bridge++ project. <http://bridge.kek.jp/Lattice-code/>. [online], 2012–2015.
- [3] M.A. Clark. QCD on GPUs: cost effective supercomputing. *PoS, LAT2009:003*, 2009.
- [4] T. DeGrand and C. DeTar. *Lattice Methods for Quantum Chromodynamics*. World Scientific Pub., 2006.
- [5] V. Demchik and N. Kolomojets. QCDGPU: open-source package for Monte Carlo lattice simulations on OpenCL-compatible multi-GPU systems. *arXiv:1310.7087 [hep-lat]*, 2013.
- [6] M. Di Pierro. QCL: OpenCL meta programming for lattice QCD. *PoS, LATTICE2013:043*, 2014.
- [7] G. I. Egri, Z. Fodor, C. Hoelbling, S. D. Katz, D. Nogradi, and K. K. Szabo. Lattice QCD as a video game. *Comput. Phys. Commun.*, 177:631–639, 2007.
- [8] M. Luscher. Solution of the Dirac equation in lattice QCD using a domain decomposition method. *Comput. Phys. Commun.*, 156:209–220, 2004.
- [9] I. Montvay and G. Münster. *Quantum Fields on a Lattice*. Cambridge Univ. Press, 1994.
- [10] S. Motoki et al. Development of Lattice QCD Simulation Code Set on Accelerators. *Procedia Computer Science*, 29:1701, 2014.
- [11] Y. Osaki and K.-I. Ishikawa. Domain Decomposition method on GPU cluster. *PoS, LATTICE2010:036*, 2010.
- [12] S. Ueda et al. Development of an object oriented lattice QCD code ‘Bridge++’. *J. Phys. Conf. Ser.*, 523:012046, 2014.