Early Performance Evaluation of Lattice QCD on POWER+GPU Cluster

Jun Doi[1]
IBM Research – Tokyo
19-21 Nihonbashi Hakozaki-cho, Chuo-ku, Tokyo, Japan
E-mail: doichan@jp.ibm.com

As supercomputers are shifting from peta-scale to exa-scale, computers with accelerators such as GPUs, MICs and FPGAs have become one of the big trends of supercomputer because of their low energy consumption and high density. Now IBM's POWER processor has quite new power, Nvidia's Tesla GPU brings huge computational capability. It is important for us to understand how this new POWER+GPU environment brings power to the actual applications in the early stage. We implemented Wilson-Dirac operator and BiCGStab solver using CUDA7.0 on the POWER+GPU cluster and evaluated the performance.

The 33rd International Symposium on Lattice Field Theory
14 -18 July 2015
Kobe International Conference Center, Kobe, Japan

1 Speaker

1. Introduction

The IBM POWER8 processor is the first POWER processor that has Nvidia's Tesla GPU for the scientific simulations. In the OpenPOWER foundation [1], POWER processor and Tesla GPU will make new generation of supercomputing and 2 large scale supercomputers SUMMIT and SIERRA will be built with POWER processor and Tesla GPU in 2017. It is important to study performance of POWER+GPU system in the early stage and we select lattice QCD as one of the important application.

In this report we develop and measure lattice QCD program on IBM Power System S842L cluster. This system has 2 POWER8 processors per node and each POWER8 processor has 12 processors cores running at 3.02 GHz. Each processor core can exploit 8 hardware threads, thus we can run 96 threads per POWER8 processor. POWER8 processor core has 2 SIMD floating point units which calculates 2 FMA operations per cycle for double precision, and 4 FMAs for single precision. So the peak performance of the POWER8 processor which is installed on IBM Power System S842L is 289.92 GFlops for double precision and 579.84 GFlops for single precision. Table 1 shows specifications of our cluster used in this report.

Nvidia's Tesla GPUs can be attached to each socket via PCI express, and our cluster has Tesla K40 GPU per socket, so we have 2 Tesla K40 per node. Since 2 GPUs are attached to different socket we cannot offer peer-to-peer copy functions between 2 GPUs. POWER8 can handle both big and little endian, and for GPU attached model POWER8 runs under little endian mode without any overhead.

Table 1 POWER System used in this report

Model	IBM POWER System S824L
Processor	POWER8
Number of CPU cores	12 cores
Number of CPUs	2 sockets
Clock frequency	3.02 GHz
Hardware threads (SMT)	12 threads per core
Performance per socket	289.92 GFlops(double)/579.84 GFlops(Single)
Memory	512 GB per node
Memory bandwidth	192 GB/s
GPU	NVIDIA Tesla K40 per socket

Ubuntu Linux is running on each node of POWER8 cluster which supports little endian on POWER processor. The environment for development is as same as common x86 Linux cluster and most of applications can be executed with recompilation. Currently we can use IBM XL C/C++ Compiler and XL Fortran Compiler and third party compilers such as GNU compilers. Nvidia's CUDA toolkit started to support POWER Systems from version 7.0.

2. Optimization of lattice QCD on POWER+GPU System

2.1 Performance consideration of Wilson-Dirac operator

In this report, we optimized Wilson-Dirac operator which is commonly used in lattice QCD simulations. The Wilson-Dirac operator D is defined by

$$D(n) = \delta(n) - \kappa \cdot \sum_{\mu=1}^{4} \left\{ (1-\gamma_\mu) U_\mu(n) \delta(n+\hat{\mu}) + (1+\gamma_\mu) U^t_\mu(n-\hat{\mu}) \delta(n-\hat{\mu}) \right\}, \quad (1)$$

where $\delta(n)$ is a spinor containing four spins with three colors each defined by complex numbers, $U_\mu(n)$ is a gauge matrix containing 3 × 3 complex number elements, and γ_μ is a gamma matrix containing 4 × 4 matrix elements, defined in Equation 2. The μ is 1, 2, 3, or 4 corresponding to the X-, Y-, Z-, or T-dimensions, and $\pm\mu$ indicates which of the eight neighbors it refers to.

$$\gamma_1 = \begin{pmatrix} 0 & 0 & 0 & -i \\ 0 & 0 & -i & 0 \\ 0 & i & 0 & 0 \\ i & 0 & 0 & 0 \end{pmatrix} \gamma_2 = \begin{pmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix} \gamma_3 = \begin{pmatrix} 0 & 0 & -i & 0 \\ 0 & 0 & 0 & i \\ i & 0 & 0 & 0 \\ 0 & -i & 0 & 0 \end{pmatrix} \gamma_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \quad (2)$$

In Equation 1, we multiply the 3 × 3 matrix and four spins with three colors for eight neighboring lattice sites and reduce them by multiplying $-\kappa$ to the spins of the focused lattice site. Actually, we can halve the matrix multiplications by using symmetry in the gamma matrices. Equation 3 shows an example for the X+ direction ($\mu=+1$), where we can see two pairs of the same matrix and spin multiplications (h_1, h_2):

$$(1-\gamma_1) U_1(n) \delta(n+\hat{1}, m) = \begin{pmatrix} U_1(n)\cdot(s_1 + i\cdot s_4) \\ U_1(n)\cdot(s_2 + i\cdot s_3) \\ -i\cdot U_1(n)\cdot(s_2 + i\cdot s_3) \\ -i\cdot U_1(n)\cdot(s_1 + i\cdot s_4) \end{pmatrix} = \begin{pmatrix} U_1(n)\cdot h_1 \\ U_1(n)\cdot h_2 \\ -i\cdot U_1(n)\cdot h_2 \\ -i\cdot U_1(n)\cdot h_1 \end{pmatrix} \quad (3)$$

This technique is also used for data compression to exchange boundaries between neighboring processes. We refer to the two compressed spins as a half-spinor.

The Wilson-Dirac operator for each direction is calculated in the following three steps:

(1) Half-spinor construction: 12 flops
(2) Multiplication of gauge matrix and half-spinor: 132 flops
(3) Spinor reduction: 48 flops (24 flops for T-dimension)

The flop counts of the Wilson-Dirac kernel per lattice site are calculated as 1,488 by adding the flop counts for all three of these steps. While calculating 1,488 floating operations, eight gauge matrices and nine spinors are loaded and one spinor is stored. The Wilson-Dirac kernel requires 2.06 bytes/flop in double precision and 1.03 bytes/flop in single precision, so its performance is limited by the memory bandwidth. By referring a roofline model, we can estimate the maximum performance of the Wilson-Dirac operator on single Tesla K40 as 139.8 GFlop/s for double precision and 279.6 GFlop/s for single precision.

2.2 Data structure consideration

There are two data forms to store structures: array of structure (AoS) and structure of arrays (SoA) shown in Figure 1. In our previous work, we stored spinors and gauge matrices in the AoS form for optimized Blue Gene supercomputer implementations [2][3] because this form can efficiently exploit cache memory to handle continuous data elements in the structures. So we apply AoS form to calculate Wilson-Dirac operator on the POWER8 processor However, the SoA form is better suited for GPU computing, and many applications running on GPUs apply this form to store data structures because it can supply continuous elements in arrays to the streaming multiprocessors. The SoA form is also used in a common optimization technique called coalescing widely used in many GPU applications. Therefore, in this work we use the SoA form to store spinors and gauge matrices to optimize our Wilson-Dirac kernel on the Kepler GPU. We pack all elements in a 1-dimensional array with an X, Y, Z, T order.

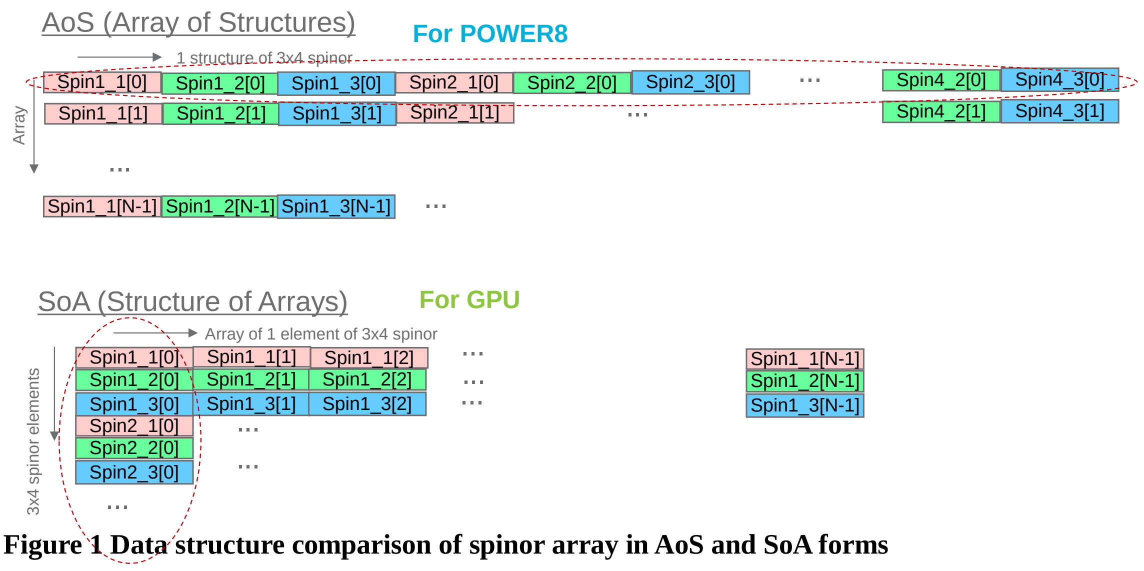

AoS (Array of Structures)
For POWER8
1 structure of 3x4 spinor
Array
Spin1_1[0] Spin1_2[0] Spin1_3[0] Spin2_1[0] Spin2_2[0] Spin2_3[0] … Spin4_2[0] Spin4_3[0]
Spin1_1[1] Spin1_2[1] Spin1_3[1] Spin2_1[1] … Spin4_2[1] Spin4_3[1]
Spin1_1[N-1] Spin1_2[N-1] Spin1_3[N-1] …
SoA (Structure of Arrays)
For GPU
Array of 1 element of 3x4 spinor
3x4 spinor elements
Spin1_1[0] Spin1_1[1] Spin1_1[2]
Spin1_2[0] Spin1_2[1] Spin1_2[2]
Spin1_3[0] Spin1_3[1] Spin1_3[2]
Spin2_1[0]
Spin2_2[0]
Spin2_3[0]
Spin1_1[N-1]
Spin1_2[N-1]
Spin1_3[N-1]

Figure 1 Data structure comparison of spinor array in AoS and SoA forms

2.3 Offloading Wilson-Dirac operator on GPU

We limit the maximum problem size so that we can store all the necessary data on the memory of GPU. So we previously transfer all the gauge matrix and spinor elements on the GPU memory, thus we only have to transfer data between GPU and host when we exchange boundary half-spinor elements between nodes or between GPUs.

We allocate one lattice site per one thread of streaming multiprocessor of GPU and we set the number of thread per grid to multiple of 32 (= warp size) to maximize thread performance. We used a least common multiple of 32 and size of X dimension.

Since we have 2 GPUs per node, however we do not have peer-to-peer capability on our system, we exchange data between 2 GPUs via host's memory. We exploit 1 process per node and control 2 GPUs by switching by cudaSetDevice function.

2.4 Other optimization on GPU

Because the performance of the Wilson-Dirac kernel is limited by the memory bandwidth, the performance improves when we increase memory access speed or decrease the total amount of memory access itself. Since the Wilson-Dirac kernel uses an SU(3) gauge matrix, which has special unitarity, 3×3 complex elements can be parameterized with 3×2 complex elements (12 real numbers). Using these parameterizations enables us to decrease memory access for the gauge matrices, but additional calculations must be executed to reconstruct the gauge matrix elements. Since GPU has strong calculation capability, we get better performance by the additional calculation than to load full gauge matrix.

The SU(3) gauge matrix can be reconstructed from any of two rows or two columns picked from the original 3×3 complex elements. Therefore, we only load the first two rows from the 3×3 complex matrix and construct elements of the third row on the fly by computation. When we define the SU(3) gauge matrix by Equation 4, the elements of the third row are calculated by Equation 5.

$$\begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ c_0 & c_1 & c_2 \end{bmatrix} \quad (4)$$

$$C = (A \times B)^* \quad (5)$$

To reconstruct a gauge matrix, we need an additional 42 flops.

2.5 Parallelization of Wilson-Dirac operator

To parallelize the Wilson-Dirac kernel, we simply apply array decomposition to divide the lattice into small domains for each distributed memory space. In the Wilson-Dirac kernel, each lattice site refers to eight neighboring lattice sites, so boundary sites on the decomposed lattice need to be copied to neighboring memory spaces. Since we have 2 GPUs per node we divide lattice sites into 2 small domains for each GPU on the node. We divide lattice in Y, Z and T dimensions, and we do not divide in X dimension because the X dimension is the innermost of the array. If we divide in X dimension, the boundary data collection of the innermost of the array will not be sequential access that causes performance loss.

To exchange boundary between nodes or between GPUs we have to transfer data between GPU and host, then we can exchange boundary via MPI. To hide data transfer latency, we overlap data transfer and calculation by using cudaMemcpyAsync and CUDA streams shown in Figure 2. We exploit (2*dimension +1) CUDA streams to prepare and transfer half-spinor arrays for each direction.

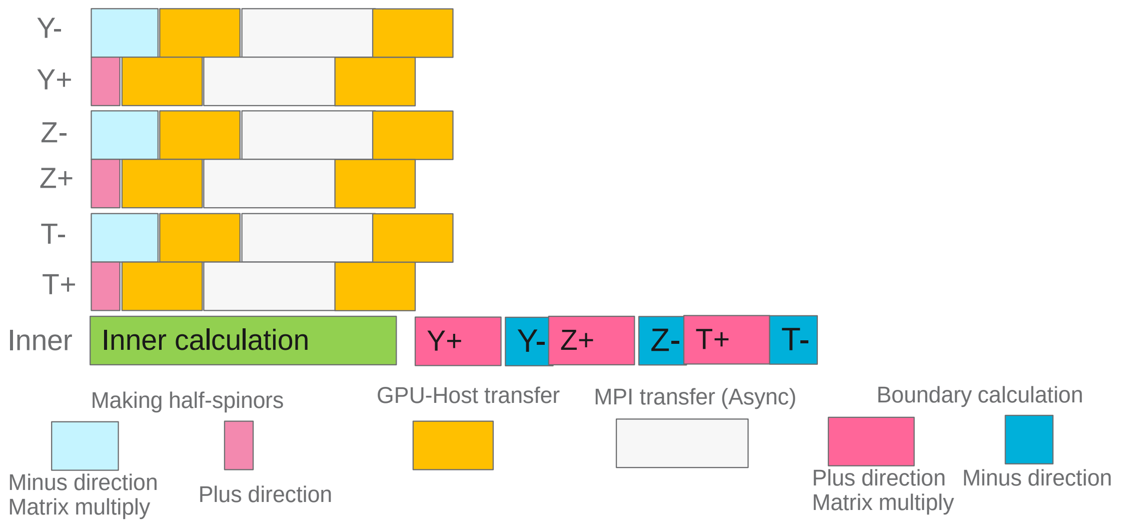


Figure 2 Overlapping data transfer and calculation by using CUDA streams

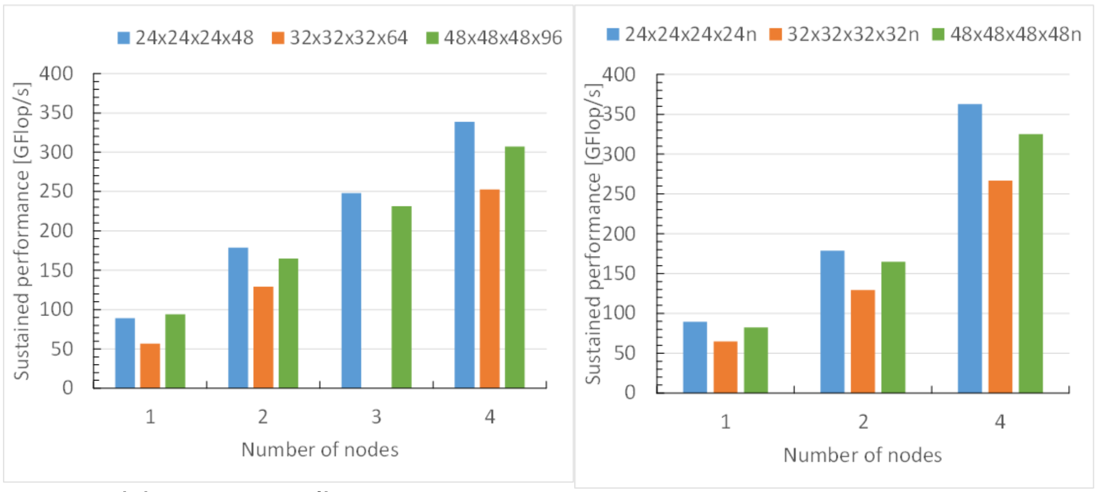
3. Performance evaluation

We implemented BiCGStab solver which uses Wilson-Dirac operator for matrix multiplication and we implemented with even-odd preconditioning. We used 4 nodes of POWER8 nodes which has 2 Tesla K40 GPUs and nodes are connected via Infiniband FDR. For the comparison we also implemented another version for POWER8 by using OpenMP for in node thread parallelization.

We used CUDA Toolkit version 7.0. And we used GNU compiler and OpenMPI version 1.8.4 for both GPU version and POWER8 version.

3.1 Performance of Wilson-Dirac operator on POWER8

Figure 3 shows the sustained performance of Wilson-Dirac operator exploited in the BiCGStab solver in double precision. We observe good scaling both for strong and weak scaling test. Since POWER8 processor has 12 cores, the performance is relatively well when the lattice size is multiple of 12.

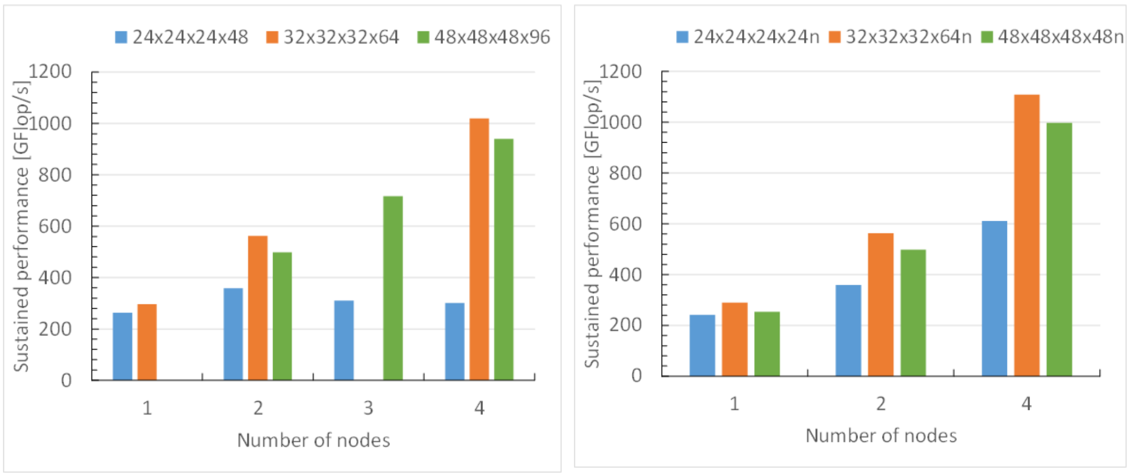


(a) Strong scaling (b) Weak scaling

Figure 3 Sustained performance of Wilson-Dirac operator using POWER8 cores

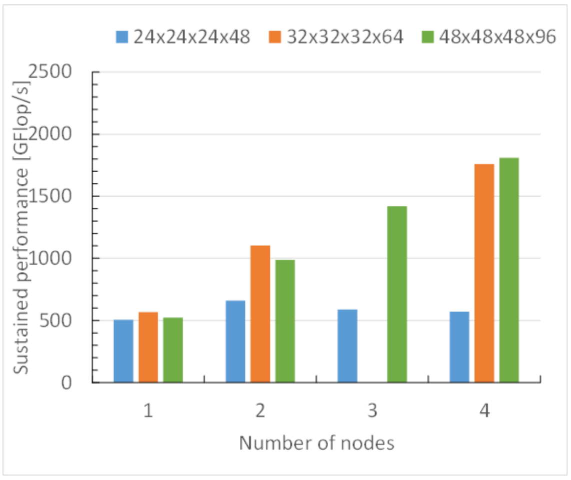
3.2 Performance of Wilson-Dirac operator on Tesla K40 controlled by POWER8

We run both double and single precision using 2 GPUs per POWER8 node. Figure 4 shows the sustained performance for double precision and Figure 5 shows that of single precision. We also observe good scalability for both strong and weak scaling test cases. Comparing to POWER8 cases, the performance of 32x32x32x64 is relatively better in GPU execution because warp size in streaming multiprocessor is 32.

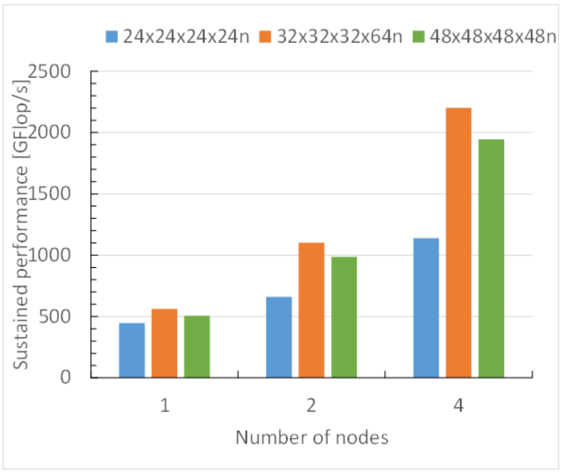


(a) Strong scaling (b) Weak scaling

Figure 4 Sustained performance of Wilson-Dirac operator using Tesla K40 GPU (double precision)



(a) Strong scaling



(b) Weak scaling

Figure 5 Sustained performance of Wilson-Dirac operator using Tesla K40 GPU (single precision)

4. Summary

We implement Wilson-Dirac operator on the IBM POWER System which has Tesla GPUs. We observe good scalability and we achieve 1 TFlop/s in double precision and 2 TFlop/s in single precision using 8 Tesla K40 GPUs. However by comparing the performance of POWER8 and Tesla K40, the performance of K40 is about only 3 times better than that of POWER8 in double precision because the performance of the Wilson-Dirac operator is bounded by the memory bandwidth. We are thinking that the performance of POWER8 itself cannot be wasted by this result. We will run the lattice QCD simulation in combination with both POWER processors and GPUs in the future. Currently GPUs are attached to POWER8 via PCI express, but in the future faster interconnect, NVLINK will be used to connect POWER processor and GPUs. We will study how NVLINK takes advantage in lattice QCD simulation.

References

[1] http://openpowerfoundation.org/

[2] J. Doi, Performance evaluation and tuning of lattice QCD on the next generation Blue Gene, PoS 25th International Symposium on Lattice Field Theory, Lattice2007, 2007.

[3] J. Doi, Peta-scale Lattice Quantum Chromodynamics on a Blue Gene/Q supercomputer, 2012 International Conference for High Performance Computing, Networking, Storage and Analysis (SC12), 2012.