Lattice QCD Performance

on Multi-core Linux Servers

Yang Suli*

Department of Physics, Peking University, Beijing, 100871

Abstract

At the moment, lattice quantum chromodynamics (lattice QCD) is the most well established non-perturbative approach to solving the theory of Quantum Chromodynamics. To get the best performance of Lattice QCD computing, it requires careful attention to balancing memory bandwidth, floating point throughput, and network performance.

In this paper, I will discuss my investigation of the effect to lattice QCD performance of different commodity processors. I will also discuss the effect of different lattice size, computing precision etc. Finally, I will give my suggestion on how to do lattice QCD computing effectively and economically.

Contents

*Email: yangsuli@gmail.com

1. Introduction to the benchmark suite

I use the BenchMZ suite made by Martin Luescher and Hartmut Wittig to evaluate the performance of lattice QCD computing. This suite is based on the DD-HMC lattice QCD simulation package, which implements the DD-HMC algorithm for lattice QCD with the Wilson plaquette action and a doublet of mass-degenerate O(a)-improved Wilson quarks [1].

1.1 DD-HMC algorithm

The DD-HMC algorithm combines domain decomposition ideas with the Hybrid-Monte-Carlo algorithm and the Sexton-Weingarten multiple-time integration scheme. In the relevant range of quark masses and lattice sizes, it is much faster than any algorithm used before. Moreover, it is well suited for parallel processing, which is a critical requirement if large lattices are to be simulated [1]. This algorithm is described in detail in [2] and [3].

1.2 DD-HMC lattice QCD simulation package

The DD-HMC lattice QCD simulation package which implements the DD-HMC algorithm is written by Martin Luescher, the same person who introduced this algorithm. So far this implementation has been extensively used in PC clusters, and we are using it in production runs.

The code is this package is highly optimized for the machines with current Intel and AMD processors, but will run correctly on any system that complies with the ISO C and the MPI 1.2 standards [1]. Many modules in the package include SSE (Streaming SIMD Extension) inline-assembly code which can be activated at compilation time, to profit from the modern x86 processors' extended capabilities of performing arithmetic operations on vectors of 4 single-precision or 2 double-precision floating-point numbers in just one or two machine circles.

In my investigation, the DD-HMC package version 1.0.1 is used. However, it should be easy to move to the latest version 1.2.1.

1.3 The BenchMZ suite

In the BenchMZ suite we focus on the computing time spent on various functions to benchmark a given system. The timing functions and short descriptions of these functions are listed in Table 1. These functions are chosen in the benchmark suite because they contain most of the time-critical parts of actual lattice QCD applications.

Typical values in production runs are used for the choice of parameter set. Two different lattice sizes are considered, 32x16x16x32 and 32x16x24x48; the corresponding block sizes are 8x8x8x8 and 8x8x12x12, respectively. The coupling parameter $\beta = 0.0$ and the hopping parameter $\kappa = 0.1$ are used.

Timing Function	Short Description
Qhat/Qhat_dble	Part of the application of the even-odd preconditioned Wilson-Dirac operator. Applies Qhat to the global single/double precision field and assigns the result.
Qhat_blk/Qhat_blk_dble	Part of the application of the even-odd preconditioned Wilson-Dirac operator. Applies Qhat to the single/double precision field on the block and assigns the result.
spinor_prod/ spinor_prod_dble	Computes the scalar product of the fields.
spinor_prod_re/ spinor_prod_re_dble	Computes the real part of the scalar product of the fields.
normalize/ normalize_dble	Replaces pk[] by pk[]/||pk|| and returns the norm ||pk||
mulc_spinor_ad/ mulc_spinor_ad_dble	Replaces pk[] by pk[]+z*pl[]
project/project_dble	Replaces ps_k by pk[]-(pl,pk)*pl[]
rotate/rotate_dble	Replaces p_k[] by sum_j p_j[]*v[n*j+k] where 0<=k, j<n and p_k=ppk[k]
norm_square/ norm_square_dble	Computes the square of the norm of the field.

Table 1. List of the functions that are used to benchmarking the system and their short descriptions.

hostname	Processors	Memory(GB)	System	Compiler
hpbl1.ifh.de	Intel Xeon E5440 Quad Core * 2	16	Scientific Linux 4.6 64bit (kernel 2.6.9)	gcc 3.4.6
hpbl2.ifh.de	Intel Xeon E5440 Quad Core * 2	16	Scientific Linux 5.2 64bit (kernel 2.6.18)	gcc 4.1.2
hpbl3.ifh.de	AMD Opteron 2356 Quad Core * 2	16	Scientific Linux 4.6 64bit (kernel 2.6.9)	gcc 3.4.6
hpbl4.ifh.de	AMD Opteron 2356 Quad Core * 2	16	Scientific Linux 5.2 64bit (kernel 2.6.18)	gcc 4.1.2

Table 2. Test platforms

2. Introduction to the platforms for test

2.1 Platforms for test

The platforms used to performing the benchmarking are listed in Table 2. They are all x86_64 machines, with typical modern configurations. Multi-core architecture is the main focus of this investigation, because we want to know how the data exchanging between cores constrains the performance of lattice QCD. Scientific Linux Operating System is used because this is the most popular OS in high energy physics computing. For 64-bit executables in SL5 machines, OpenMPI 1.2.5 is used as MPI implementation, otherwise OpenMPI 1.2.3 is used.

2.2 Complier flags

For 32-bit executables, the complier flags used are "-ansi –pedantic –Wno-long-long –Wall –mcpu=i586 –malign-double –fno-force-mem –O –DSSE3 –DPM". For 64-bit executables, the complier flags used are "-std=c89 -pedantic -Wno-long-long -Wall –fomit -frame-pointer -O -DSSE3 –DPM". These flags are suggested by the author of the DD-HMC simulation package. Actually, several compiler options mentioned in gcc man pages are tried, but the effects are not very significant. What tend to really drive the performance are the prefetch distance (which is set by –DPM) and the SSE inline-assemble activation (which is set by –DSSEx).

3. Effect to lattice QCD performance of different processor architectures

High lattice QCD performance requires excellent single and double precision floating point performance, high memory bandwidth* and fast communications between nodes. The effect of network communications will not be investigated in this paper; and the other two factors both largely depend on the CPU architectures. Previous study shows that for older CPUs, memory bandwidth typically constrains the performance on single nodes [4].

3.1 Introduction to the compared CPU architectures

Two popular commodity processors, Intel Xeon E5440 and AMD Opteron 2356, which are believed to be representative to the modern Intel and AMD multi-core CPUs, are investigated.

Detailed technical information about these two processors can be found in the official websites of Intel and AMD companies [5-6]. I will concentrate only on the key aspects that affect the floating-point performances and memory bandwidth.

The main frequency of Intel E5440 is 2.83 GHz, while it is 2.30 GHz for AMD 2356, which basically means that E5440 runs faster than 2356.

*The term memory bandwidth used here not only refers to the access to main memory, but also to the data exchanging between cores and processors.

Both CPUs have integrated floating point unit and support SSE3 instruction set. AMD Opteron 2356 also provides 128-bit floating-point pipeline enhancements To improve float-point performance. Figure 1 gives the illustration of the comparison of floating-point performance on these two processors. One can find from the figures that generally AMD Opteron 2356 does better than Intel Xeon E5440 in floating-point performance despite its lower main frequency. However, this advantage is not obvious in low optimization codes that are typically used in high energy computing.

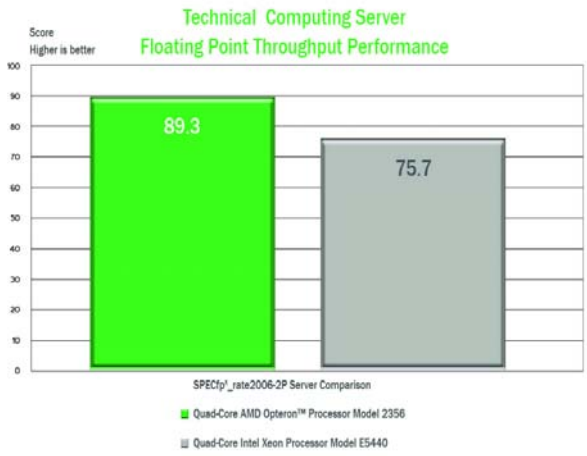

Fig. 1.a [7]

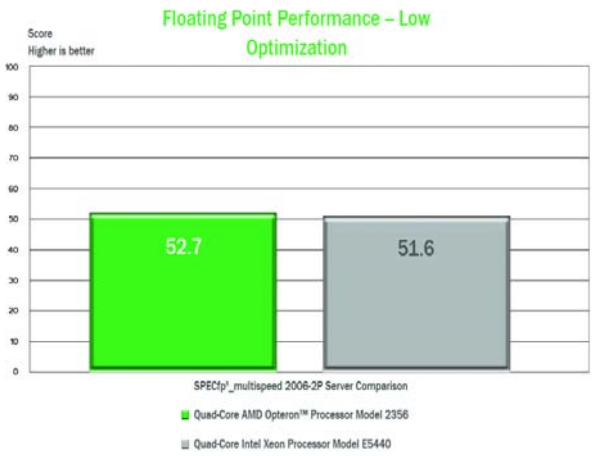

Fig. 1.b

Figure 1: Illustration of the floating-point throughput performance comparison of Intel Xeon E5440 and AMD Opteron 2356 2P servers. Fig. 1.a is the SPECfp*_rate results with high optimization; while Fig. 1.b is the SPECfp*_multispeed results with typical optimization used in high energy computing [8].

Concerning memory bandwidth, AMD has made some important changes in the processor architectures of Opteron series. Each Opteron has an integrated memory controller and a separate memory bus attached to the controller. HyperTransport™ Technology Links are used to link between processors. A given processor can address both local memory, and memory attached to the other processor. However, NUMA-aware kernels such as the Linux 2.6.x series must be used to gain the best performance on AMD processors [4]. In the contrast, Intel Xeon E5400 series still use the traditional front-size bus architectures, with E5440 having a 1333 MHz FSB. Figure 2 gives the comparison of memory bandwidth of these two processors. One can find from the figure that the memory bandwidth of Intel Xeon E5440 processor system only approaches 45%-55% of the bandwidth of AMD Opteron 2356 processor system.

3.2 Benchmark results

The Benchmark results used to evaluate performances of different processors are shown in Figure 3; performances are given in Mflops (higher is better), whose value

is inversely proportional to time spent per lattice point. Lattice QCD applications with single and double precision, smaller and bigger lattice, as well as 32-bit and 64-bit executable format, are runned in hpbl1, hpbl2 (Intel-based machines), and hpbl3, hpbl4 (AMD-based machines).

Performances vary from different timing functions, but one can find that generally AMD-based machines have a higher Mflops value, which is in accordance with the fact that AMD Opteron2356 beats Intel Xeon E5440 in both floating-point performance and memory bandwidth despite its lower main frequency. Moreover, one will observe that for bigger lattice size or higher (double) precision, the

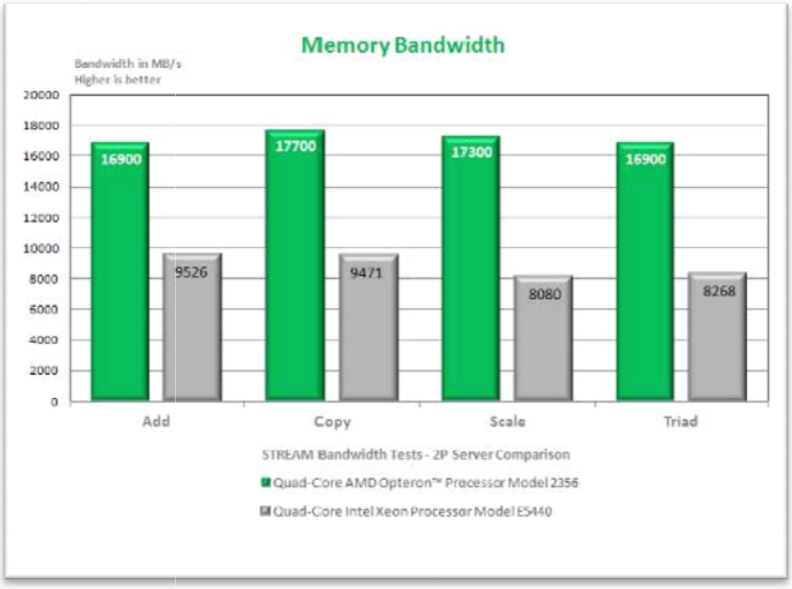

Figure 2: Illustration of the memory bandwidth comparison of Intel Xeon E5440 and AMD Opteron 2356 2P servers. The results are obtained using STREAM benchmark suite [7].

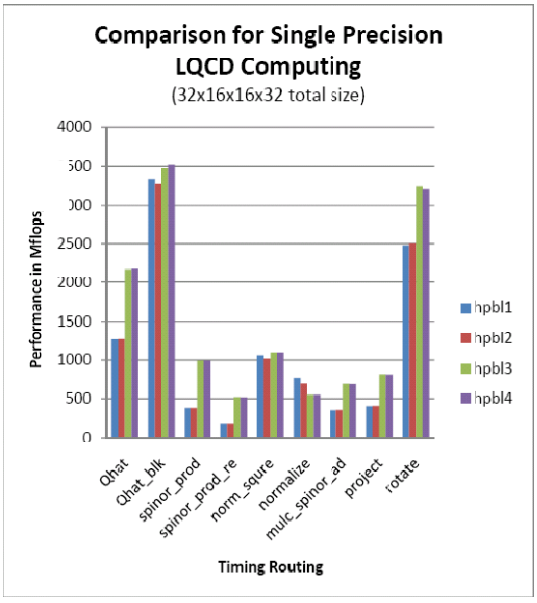

Fig 3.a

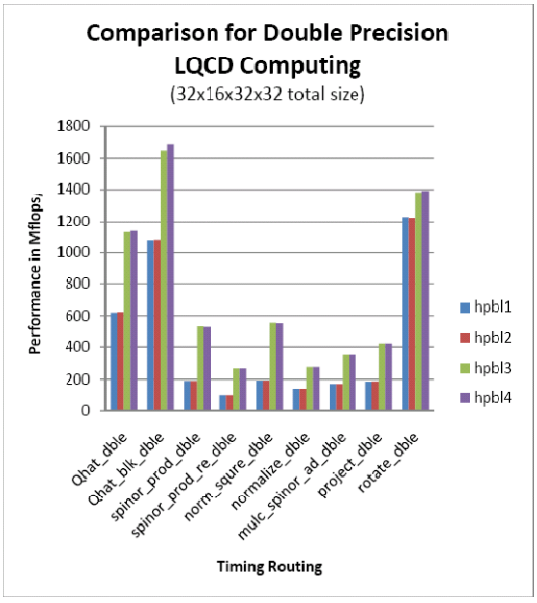

Fig 3.b

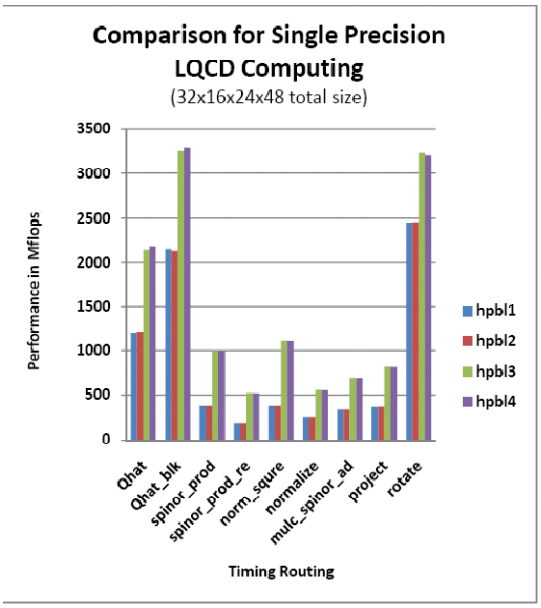

Fig 3.c

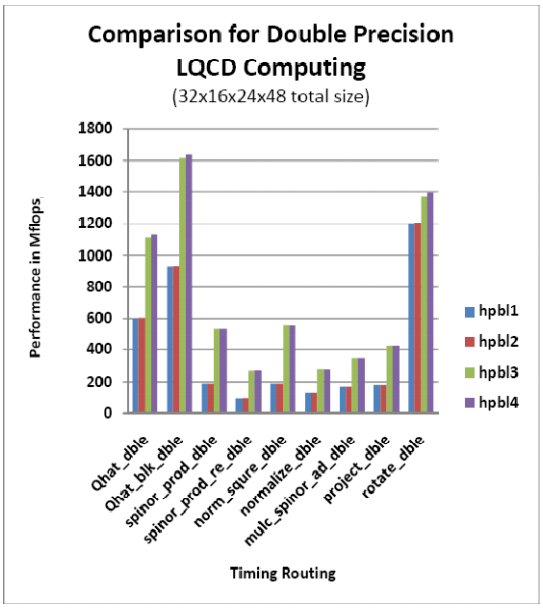

Fig 3.d

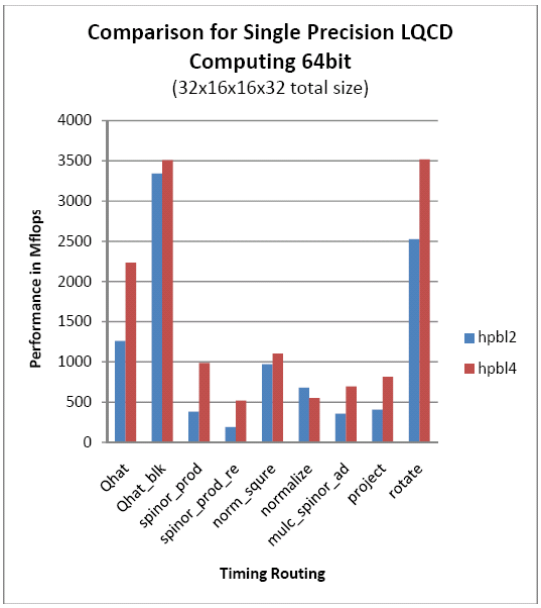

Fig 3.e

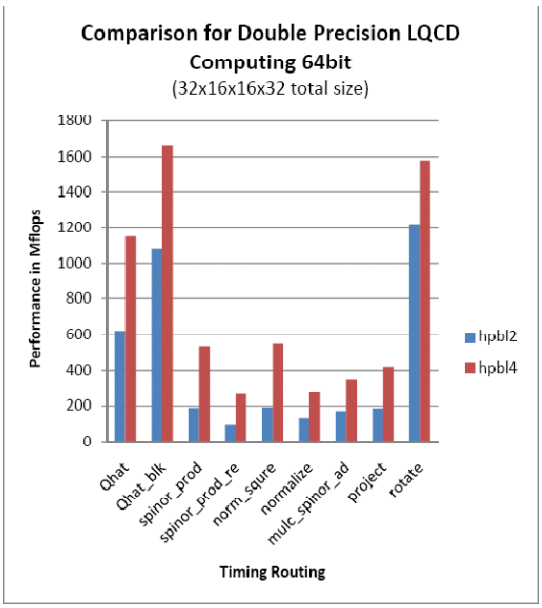

Fig 3.f

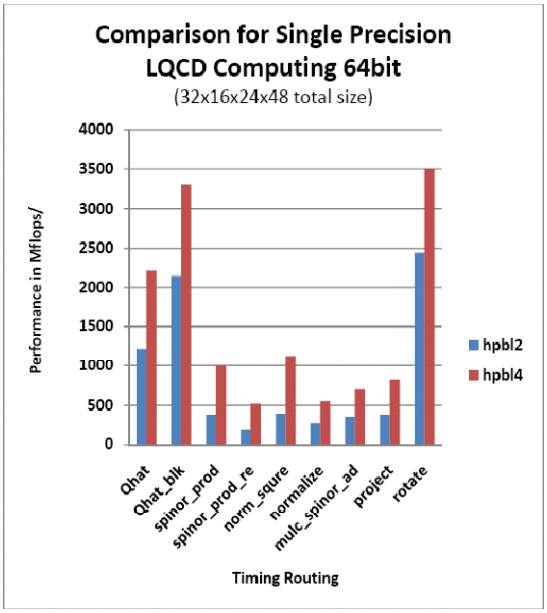

Fig. 3.g

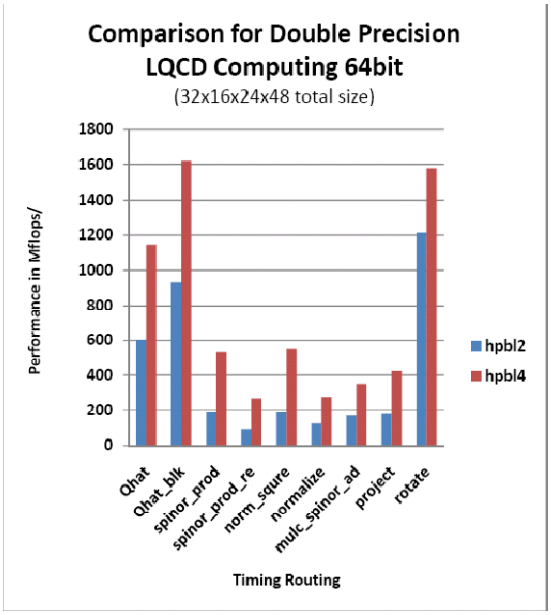

Fig 3.h

Figure 3: Benchmark results for processor architecture comparison.

Intel-based machine performances have the tendency of approaching 40%-50% of AMD-base machine performances. This tendency can be seen more clearly when one looks at the Qhat_blk and norm_squre timing functions. This observation can be explained by that memory bandwidth is still the constraining factor in lattice QCD computing, so when the lattice extends into main memory, the lattice QCD performance basically reflects the memory bandwidth performance of the particular machine, which is 40%-50% of AMD for Intel-based machines.

4. Effect to lattice QCD performance of different lattice size and computing precision

The Benchmark results used to evaluate performances of different lattice size and computing precision are shown in Figure 4. The lattice QCD benchmark suite is runned in four different machines (hpbl1-4); and for each machine, the suite is runned with single and double precision, and with 32x16x16x32 and 32x16x24x48 lattice, respectively.

Comparing the results of different lattice size runs, one can find that in single precision computing, there may be some performance decline for larger lattice; but in double precision runs, the Mflops performance value for different lattice size tends to become the same, i.e. in high precision computing the time spent on each lattice point does not increase when lattice size increases.

Comparing the result of different precision runs, one can find some analogous property. In small lattice runs, the double precision performance may only reach 1/5, or even 1/7 of the single precision performance, e.g. in the normalize or norm_squre functions. However, in large lattice runs, almost all the benchmarking functions

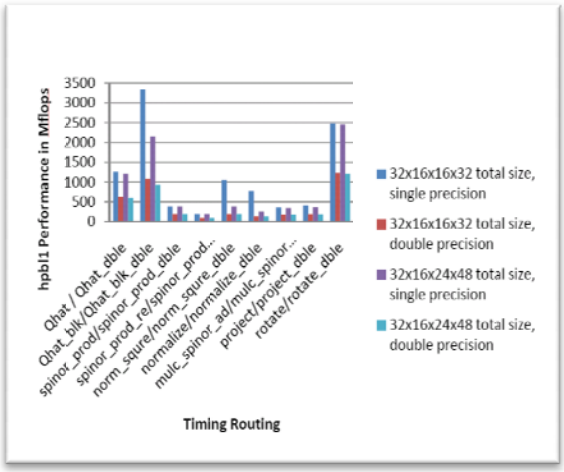

Fig. 4.a

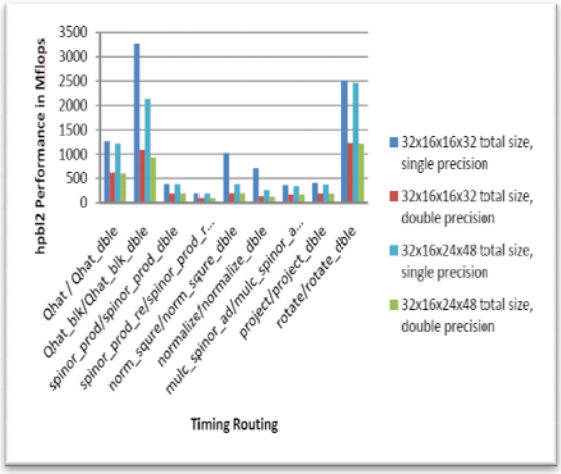

Fig. 4.b

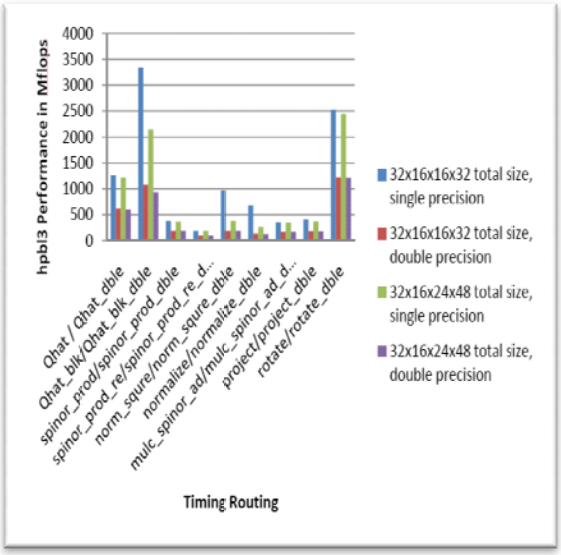

Fig. 4.c

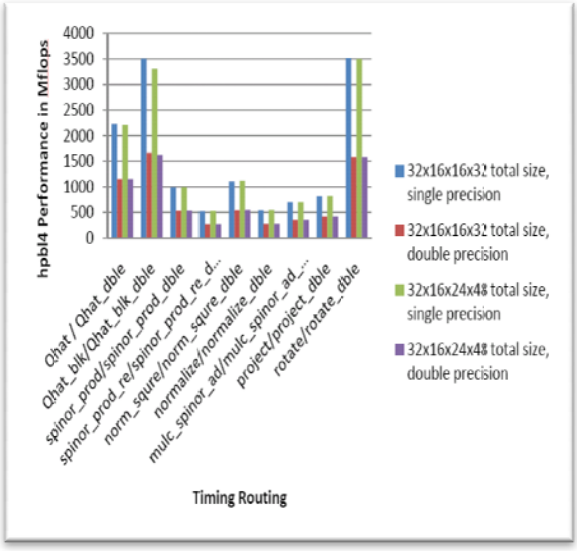

Fig. 4.d

Figure 4: Benchmark results for lattice size and computing precision comparison

report a 2:1 performance ratio for single vs. double precision runs.

Both observations may result from the fact that memory bandwidth typically constrains the lattice QCD performance. When larger lattice size or higher computing precision is specified, the lattice extends to the main memory, so the nonlinear effect of the algorithm is disguised by the linear memory access time increasing. As a result, time spent on per lattice point remains linearly. However, more statistics is still needed to confirm this explanation.

5. Effect to lattice QCD performance of 32bit and 64bit applications

The benchmark results used to evaluate performance of 32bit and 64 bit applications are showed in Figure 5. The 32-bit and 64-bit lattice QCD benchmark suites are runned in two machines with different architectures.

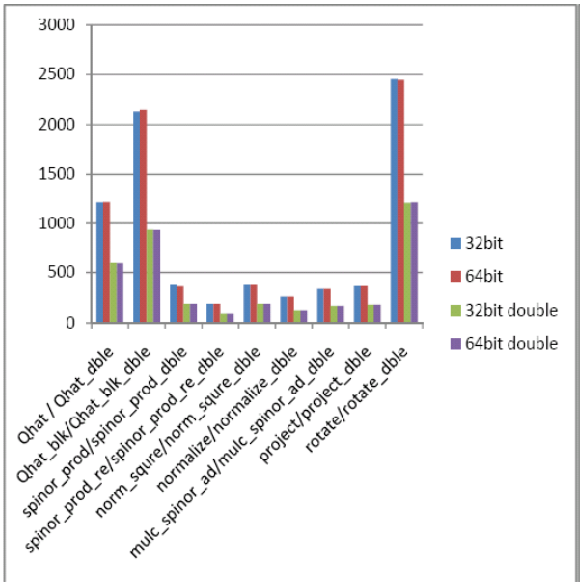

Fig 5.a Benchmark results on Intel-based machine hpbl2

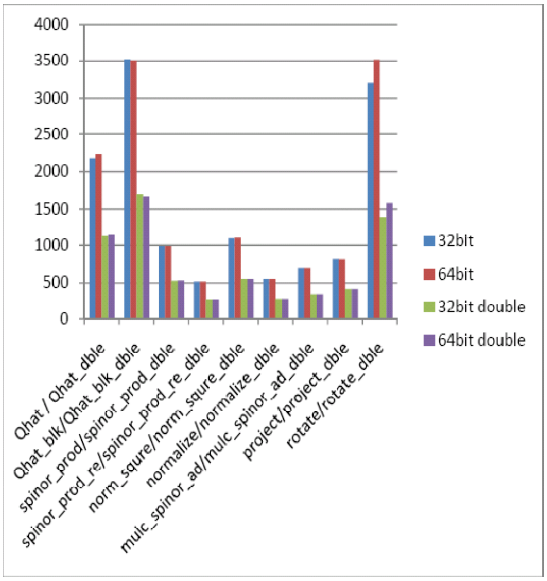

Fig 5.b Benchmark results on AMD-based machine hpbl4

Figure 5: Benchmark results for ELF format comparison

One can see that there is no considerable performance difference between 32-bit and 64-bit lattice QCD applications. The reason for this phenomenon still needs some investigation. But the guess is that it has something to do with the fact that the prefetch distance is 128 bit for both 32-bit and 64-bit executables.

6. Conclusions

Memory bandwidth is still the constraining factor of lattice QCD computing even for fairly new machines. So processors with special technology to speed up the memory access have evident advantage in lattice QCD computing. As a respective case, AMD Opteron 2356 with integrated memory controller and HyperTransport channels gets a 120% higher score than Intel Xeon 2345 with traditional front size bus architecture in lattice QCD benchmarking. Besides, the increase of lattice size or computing precision has a roughly linear effect on computing time. .

So for people who want to perform effective and economic lattice QCD computing, memory bandwidth should be a key factor to focus on; AMD Opteron systems with enhanced memory access design may probably be a good choice. Also, since there is no dramatic performance decline for larger lattice and higher computing precision;

this would be quite acceptable when there is need.

Acknowledgment

This research is sponsored by DESY summer student program. I want to thank my supervisor Peter Wegner for his guidance. I want to thank Mr. Martin Luescher and Mr. Hartmut Wittig for their kindness and patience when answering my rather naive questions. I would also like to thank Mr. Goetz Waschk and Mr. Stephan Wiesand for their kind help to my work.

References

[1] M. Luescher, README file of the DD-HMC package.
[2] M. Luescher, Comput. Phys. Commun. 165 (2005) 199
[3] M. Luescher, Comput. Phys. Commun. 156 (2004) 209
[4] D. Holmgren et. al, FERMILAB-CONF-04-484-CD
[5]http://www.intel.com/products/processor/xeon5000/documentation.htm
[6]http://www.amd.com/us-en/Processors/ProductInformation/0,,30_118_8796,00.html
[7]http://www.amd.com/us-en/Processors/ProductInformation/0,,30_118_8796_8800,00.html
[8]https://hepix.caspur.it/processors/dokuwiki/doku.php?id=benchmarks:introduction