

# Performance of the 3D FFT on the 6D network torus QCDOC parallel supercomputer <sup>☆</sup>

Bin Fang <sup>a</sup>, Yuefan Deng <sup>a,\*</sup>, Glenn Martyna <sup>b</sup>

<sup>a</sup> Department of Applied Mathematics and Statistics, Stony Brook University, Stony Brook, NY 11794-3600, USA

<sup>b</sup> IBM Research, Physical Sciences Division, IBM T.J. Watson Laboratory, Yorktown Heights, NY 10598, USA

Received 24 May 2006; received in revised form 28 December 2006; accepted 28 December 2006

Available online 1 February 2007

## Abstract

QCDOC is a massively parallel supercomputer with tens of thousands of nodes distributed on a six-dimensional torus network. The 6D structure of the network provides the needed communication resources for many communication-intensive applications. In this paper, we present a parallel algorithm for three-dimensional Fast Fourier Transform and its implementation for a 4096-node QCDOC prototype. Two techniques have been used to increase its parallel performance: simultaneous multi-dimensional communication and communication-and-computation overlapping. Benchmarking experiments suggest that 3D FFTs of size  $128 \times 128 \times 128$  can scale well on such platforms up to 4096 nodes. Our performance results suggest stronger scalability on QCDOC than on IBM BlueGene/L supercomputer.

© 2007 Elsevier B.V. All rights reserved.

PACS: 02.70.Ns; 34.20.Gj; 31.15.Ar; 87.15.He

Keywords: Molecular dynamics simulation; Long-range interaction; Ab initio calculation; FFT; Blue Matter; Strong scalability; Parallel efficiency

## 1. Introduction

Investigation of biological, physical and chemical phenomena by means of computer simulation has been a wide scientific computing area since the end of last century. From the first computer simulation of the simplest system, hot-gas plasmas, reviewed in [1,2], to the current simulation of the small protein ‘trp cage’, consisting of 20-residue sequence [3], many significant improvements have been made both computational efficiency and accuracy. In recent years, the molecular dynamics simulations of more complex biological systems has become one of the more interesting and challenging research areas, attracting more and more researchers [4–7]. However, there is an inherent bottleneck in MD simulation, the calculation of the long-range interaction, which makes simulating large systems impractical because of its quadric computational complexity

when formulated in the most basic way [5,8]. In the past two decades, researchers have developed many methods to handle long range forces efficiently, an effort which involved both theoretical and algorithmic advances [9–11]. One of the most popular approaches involves the expression and computation of the dominant portion of the long force in reciprocal space to yield accurate and numerically efficient description of long range interactions with  $O(N \log N)$  computational complexity [6,7,10,11]. There are several commonly used reciprocal space based methods. The most commonly used methods in MD simulation include the Particle Mesh Ewald, Particle–Particle Ewald, and Smooth-particle mesh Ewald methods [4,12–17]. Besides their utility in increasing the efficiency of classic molecular dynamics, the reciprocal space based method has many other applications, such as in treating long range interactions in ab initio calculation in clusters [18] and even in non-three-dimensional spaces [19,20]. However, no matter what flavor of reciprocal space method is employed, the Fast Fourier transform [21,22] is the only means to access the reciprocal space in order  $O(N \log N)$ ; so, therefore, the 3D-Fast Fourier Transform (3D-FFT) plays an important role in the computation of the long-

<sup>☆</sup> This project is supported by BNL LDRD grant #36930, for molecular dynamics on ultrascale supercomputer.

\* Corresponding author. Tel.: +1 631 865 1008.

E-mail address: [Yuefan.Deng@StonyBrook.edu](mailto:Yuefan.Deng@StonyBrook.edu) (Y. Deng).

range interaction in a variety of important computer simulation methods.

Much effort has been put into improving the sequential efficiency of the 3D-FFT [22–26]. In this paper, we focus on improving the scalability and efficiency of the 3D-FFT run on a specific parallel supercomputer QCDOC, which has the ability to expand up to more than 10,000 nodes, with a six-dimensional torus network topology [27–29]. This is actually the most challenging computational phase implemented in MDoC [30], our new parallel MD simulation package, that is designed to simulate globular proteins in aqueous solution which involves treating systems containing between 10,000 and 100,000 atoms.

The most common parallelization scheme adopted to treat 3D-FFTs of size  $N \times N \times N$  is the so-called slab decomposition [31,32], in which a three-dimensional (3D) data array is decomposed along one single axis into “slabs”. For instance, to perform the computation on  $N$  nodes, each node will have partial data of size  $N \times N \times 1$ . The scalability of this method is limited by the number of data points along a single axis. In addition, the slab decomposition does not map naturally to torus architecture, such as QCDOC and IBM’s BlueGene/L supercomputer [33,34], with tens of thousands of nodes.

In this paper, we adopt an efficient method, implemented on BlueGene/L [35,36], to a new and much more complicated six-dimensional (6D) torus network architecture. With this decomposition, an  $N \times N \times N$  array is distributed on a 6D grid  $P_1 \times P_2 \times P_3 \times P_4 \times P_5 \times P_6$ , each node having a subset of data of size  $\frac{N}{P_1 \times P_2} \times \frac{N}{P_3 \times P_4} \times \frac{N}{P_5 \times P_6}$ . The main difference of our implementation from IBM’s volumetric method is that, instead of communicating along one-dimensional axis and within two-dimensional plane, we communicate within two-dimensional plane and within four-dimensional space, respectively.

Our goal is to provide a highly scalable 3D FFT framework for QCDOC that can use any serial 1D FFT solver as a building block. Here, we use FFTW [24,25,32] as the sequential 1D FFT solver and QMP [37] as the communication protocols to assess the performance of our parallel scheme. The metric for efficiency used in this paper is the “total running-time to final solutions” for the problems. We conclude that although the efficiency is worse than IBM’s volumetric 3D-FFT run on BlueGene/L within the factor of hardware difference, the (strong) scalability for relatively small data sizes, such as  $32^3$ ,  $64^3$ , and  $128^3$ , is much better and the approach presented here actually can scale well up to 4096 nodes.

The rest of the paper is organized as follows. Section 2 describes the architecture of QCDOC relevant to our method. Section 3 presents the parallel algorithm we have developed. Section 4 provides the model for the performance analysis, followed by actual performance results and their comparison with BlueGene/L results.

## 2. QCDOC supercomputer

QCDOC (Quantum Chromodynamics On a Chip) [28,29,38] is a massively parallel computer developed by a group of researchers at Columbia University, the RIKEN BNL Research Center (RBRC), IBM’s Watson Research Laboratory, and the

UKQCD collaboration. QCDOC uses a six-dimensional, low-latency mesh network to connect processing nodes, each of which includes a single custom application-specific integrated circuit (ASIC) and an industry-standard DDR memory module. Each node has a peak speed of 1 Gflops. Larger systems can be built by integrating many such processing elements with prescribed resource requirements on a certain topology.

The main components of the QCDOC ASIC are:

- (1) 500 MHz, 32-bit PowerPC 440 processor core;
- (2) 64-bit, 1 Gflops floating-point unit;
- (3) 4 Mbytes embedded DRAM on-chip memory;
- (4) nearest-neighbor serial communications unit (SCU) with aggregate bandwidth of 12 Gbps in 12 independent bi-directions (for six-dimensional mesh);
- (5) other components such as Ethernet controller, JTAG, and interrupt controller.

With the ordinary inexpensive computing nodes, and the unique interconnection, QCDOC can deliver a price/performance of less than \$1 per sustained Mflops [29]. The most important characteristic of QCDOC system for parallel 3D-FFT is the six-dimensional torus network, in which every processor is directly connected to 12 other nearest neighboring processors with bidirectional communication channels.

## 3. 3D-FFT and its parallelization

Although there exist many mature parallel 3D-FFT methods [22–26], they are not efficient on QCDOC if they are routinely ported in. In this section, we first briefly review the basic 3D-Discrete Fourier Transform (DFT), layout the data mapping, followed by a description of the parallel 3D-FFT developed for and run on QCDOC.

### 3.1. Basic 3D-DFT

Let  $X(j_x, j_y, j_z) \in \mathbb{C}^{N_x \times N_y \times N_z}$  be a three-dimensional complex matrix. The 3D DFT of matrix  $X$ , also a 3D matrix  $Y(k_x, k_y, k_z)$ , is defined as

$$Y(k_x, k_y, k_z) = \sum_{j_z=0}^{N_z-1} \sum_{j_y=0}^{N_y-1} \sum_{j_x=0}^{N_x-1} X(j_x, j_y, j_z) W_{N_x}^{j_x k_x} W_{N_y}^{j_y k_y} W_{N_z}^{j_z k_z},$$

$$k_x = 0, \dots, N_x - 1, \quad k_y = 0, \dots, N_y - 1,$$

$$k_z = 0, \dots, N_z - 1,$$

where  $W_{N_\alpha} = e^{-2\pi i / N_\alpha}$ ,  $N_\alpha \in \mathbb{Z}$  and  $\alpha = x, y, z$  [21,22].

This 3D-DFT computation can be decomposed in three successive stages.

First, we compute the one-dimensional DFT of  $X(:, j_y, j_z)$  for all  $(j_y, j_z)$  pairs

$$X_1(k_x, j_y, j_z) = \sum_{j_x=0}^{N_x-1} X(j_x, j_y, j_z) W_{N_x}^{j_x k_x},$$

$$k_x = 0, \dots, N_x - 1, \quad j_y = 0, \dots, N_y - 1,$$

$$j_z = 0, \dots, N_z - 1,$$

by evaluating  $N_y \times N_z$  independent one-dimensional DFTs of size  $N_x$  along  $x$ -axis.

Second, we compute the one-dimensional DFT of  $X_1(k_x, :, j_z)$  for all  $(k_x, j_z)$  pairs

$$X_2(k_x, k_y, j_z) = \sum_{j_y=0}^{N_y-1} X_1(k_x, j_y, j_z) W_{N_y}^{j_y k_y},$$

$$k_x = 0, \dots, N_x - 1, \quad k_y = 0, \dots, N_y - 1,$$

$$j_z = 0, \dots, N_z - 1,$$

by evaluating  $N_x \times N_z$  independent one-dimensional DFTs of size  $N_y$  along  $y$ -axis.

Third, we compute the one-dimensional DFT of  $X_2(k_x, k_y, :)$  for all  $(k_x, k_y)$  pairs

$$Y(k_x, k_y, k_z) = \sum_{j_z=0}^{N_z-1} X_2(k_x, k_y, j_z) W_{N_z}^{j_z k_z},$$

$$k_x = 0, \dots, N_x - 1, \quad k_y = 0, \dots, N_y - 1,$$

$$k_z = 0, \dots, N_z - 1$$

by evaluating  $N_x \times N_y$  independent one-dimensional DFTs of size  $N_z$  along  $z$ -axis.

### 3.2. The parallel algorithm

The QCDOC network topology is a truncated 6D torus that can be expressed as  $p_x \times 2 \times p_y \times 2 \times p_z \times 2$ , where  $p_i = 2^{c_i}$  represents the number of nodes in each of the three regular dimensions and  $c_i$  can be any positive integer. The other three dimensions are truncated, with two nodes in each of them. Along every regular dimension, there is a bidirectional link between each pair of nearest neighbors; while along every truncated dimension, there are two bidirectional links connecting the two nearest nodes, which can do the data communication independently and simultaneously. Technically, we manipulate QCDOC as if every processor was directly connected to 12 nearest neighboring processors with bidirectional communication channels. Let  $P_x = p_x \times 2$ ,  $P_y = p_y \times 2$ , and  $P_z = p_z \times 2$ ; i.e.  $P_i$  is the number of nodes in two-dimensional mesh consisting of one regular dimension and one truncated dimension. Without loss of generality, we assume the data matrix is of size  $N = N_x \times N_y \times N_z$ .

Initially, a node with the coordinate  $(a, b, c, d, e, f)$ , where  $a = 0, \dots, p_x - 1$ ;  $b = 0, 1$ ;  $c = 0, \dots, p_y - 1$ ;  $d = 0, 1$ ;  $e = 0, \dots, p_z - 1$ ;  $f = 0, 1$ , stores a sub-block of data  $n_x \times n_y \times n_z$ , where  $n_t = \lceil N_t / P_t \rceil$  with  $t \in \{x, y, z\}$ . We define submatrix

$$X_{i,j,k} \equiv X((i-1)n_x : in_x, (j-1)n_y : jn_y, (k-1)n_z : kn_z),$$

$$i = b \cdot p_x + a, \quad j = d \cdot p_y + c, \quad k = f \cdot p_z + e \quad (1)$$

and store this submatrix in this node  $(a, b, c, d, e, f)$ , with the following constraints on the sizes of these sub-matrices:

$$n_x \cdot n_y = \alpha \times P_z, \quad n_x \cdot n_z = \beta \times P_y, \quad n_y \cdot n_z = \gamma \times P_x,$$

where  $\alpha$ ,  $\beta$ , and  $\gamma$  are integers.

As mentioned earlier in Section 3.1, 3D FFT is computed dimension by dimension.

First, along the  $z$ -axis, we compute  $N_x \times N_y$  independent 1D FFTs of size  $N_z$  each. Before the actual 1D transform, we rearrange the scattered data along the  $z$ -axis to let each of the  $N_x \times N_y$  1D FFTs be stored locally. To do this, an all-to-all personalized communication along the  $z$ -axis must be performed. Serial 1D-FFT computations are carried out in each node independently, followed by the same communication as above to transfer the data back to the original distribution described by Eq. (1). Thus, we have completed the 3D-FFT computation along one dimension and obtained the intermediate array  $X_1$ , as in Section 3.1. Next, along the  $y$ -axis,  $N_z \times N_x$  sets of independent 1D FFTs with the size of  $N_y$  are performed. Finally, we perform similar operations along the  $x$ -axis. All of them consist of the same three steps, i.e. forward transpose (FT), serial computation (SC), and backward transpose (BT), as solving  $z$ -directional FFTs. Among these three steps, communication is needed for the transposes.

In our implementation, the most straightforward scheme is to link the three steps as shown in Part A of Fig. 1, resulting in utilization of communication channels in only two out of six dimensions concurrently. There are at least two improvements that can be made to increase the efficiency.

Since the forward transpose along one axis and the backward transpose along the next axis are applied within different 2D torus plane, we can combine them to reduce the communication as shown in Part B of Fig. 1. That is, the backward transpose along the  $z$  physical axis can be done on the  $ab$  torus plane simultaneously with, and independently upon, its succeeding forward transpose along the  $y$  physical axis done on the  $cd$  torus plane, exploiting the independent bidirectional communication channels in four dimensions concurrently. The same manipulation is applied to the backward transpose along the  $y$ -axis and its succeeding forward transpose along the  $x$ -axis. In this way, instead of six, we actually count only four times of communication, which has already been taken into consideration in the model next section.

Furthermore, within each one of the three-dimensional FFT computations, instead of doing all local 1D-FFTs sequentially with the communication channels idle, followed by an intensive communication with all CPUs idle, we chop the local data into pieces by means of giving another input parameter, indicating how big the chop is. By doing this, we perform the sequential 1D FFT computation on one piece, while simultaneously transferring the next piece. This communication-computation overlapping, shown in Part C of Fig. 1, accrues from the unblocking communication scheme of QMP [37]. Some remarks here are necessary. First, the parameter we choose to indicate the size of the chopped data is dependent upon the hardware, including the speed of CPU, the communication latency and the channel bandwidth. In this paper, all the efficiencies shown in the next section are obtained on QCDOC by choosing optimal parameters empirically. Second, in Part C of Fig. 1, we only depict the communication part, which is the bottleneck in parallel 3D FFT computation, with darker shaded areas mostly representing the time spent on unavoidable memory access and

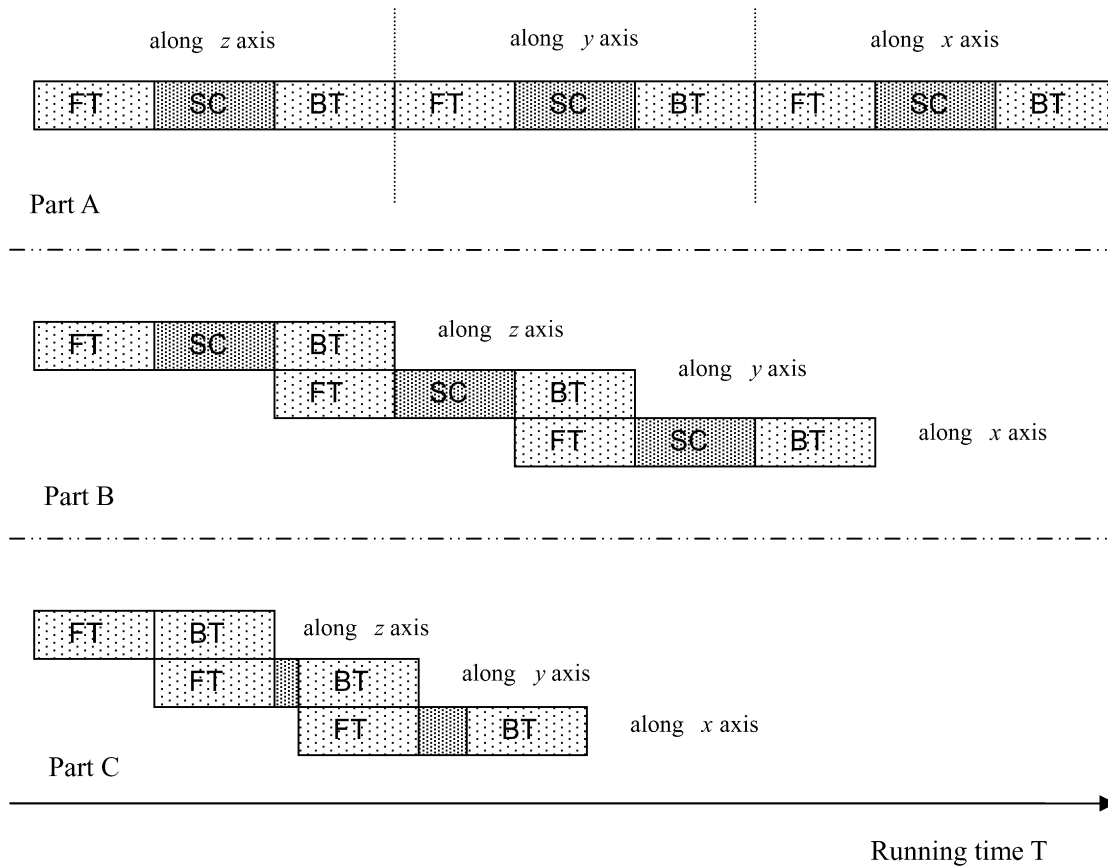


Fig. 1. This image shows different steps in 3D FFT computation. Both of the communication steps, forward transpose (FT) and backward transpose (BT), are shown in shallow shaded area; the sequential computation part (SC) is shown in dark.

little non-overlapped sequential 1D FFT computation. Most of the sequential computation time (darker shaded area) appearing in Part A and B has been overlapped with communication.

#### 4. Performance analysis

In this section, both the theoretical model and the actual performance are presented. We also analyze the deviation of actual performance from the model.

##### 4.1. The model

QCDOC inter-node network's store-and-forward routing determines our communication model [39]:

$$t_{\text{comm}} = t_s + (mt_w + t_h)l$$

where  $t_s$  is the start-up time,  $t_w$  the per-word transferring time,  $t_h$  the per-hop time,  $m$  the number of words transferred and  $l$  the number of hops. The fact that QMP supports non-blocking communication allows us the convenience of building communication handlers before actual data transfer, justifying negligence of start-up time. Moreover, because the memory-to-memory transfer time for nearest neighbors is around 0.6  $\mu\text{s}$ , the total per-hop time in the 3D-FFT implementation is so small that it can be ignored. Additionally, we may assume the actual link transferring speed is approximately 90% of its theoretical

peak value. With these assumptions, we simplify the communication model as  $t_{\text{comm}} = mt_w \times l/0.9$ .

Because the parallel 3D-FFT is communication-intensive, we only include the communication time in our model. However, for the computation part, we may adopt the idealized bound [35] given by IBM. Although the theoretical complexity of 1D FFT on a problem of size  $N$  is  $5N \log_2 N$ , data dependencies force a fused multiply-add (FMA) machine to use eight cycles when a fused multiply-add is issued every cycle, resulting in a more accurate bound of  $8N \log_2 N$  clock cycles.

##### 4.2. Performance and its analysis

We have benchmarked our parallel 3D-FFT algorithm on three QCDOC prototypes. The first, with fewer than 16 nodes running at 360 MHz, resides at Columbia University; the second, with up to 64 nodes running at 420 MHz, resides at Brookhaven National Laboratory (BNL); the third, up to 4096 nodes running at 400 MHz, also resides at BNL. Among all three prototypes, the 4-node and 8-node systems are configured as two-dimensional tori, while the 16-node and 32-node systems are configured as four-dimensional tori. All others are configured as truncated six-dimensional tori.

Our implementation is based on QMP message-passing interface and FFTW 3.1 library as its 1D FFT solver. The running

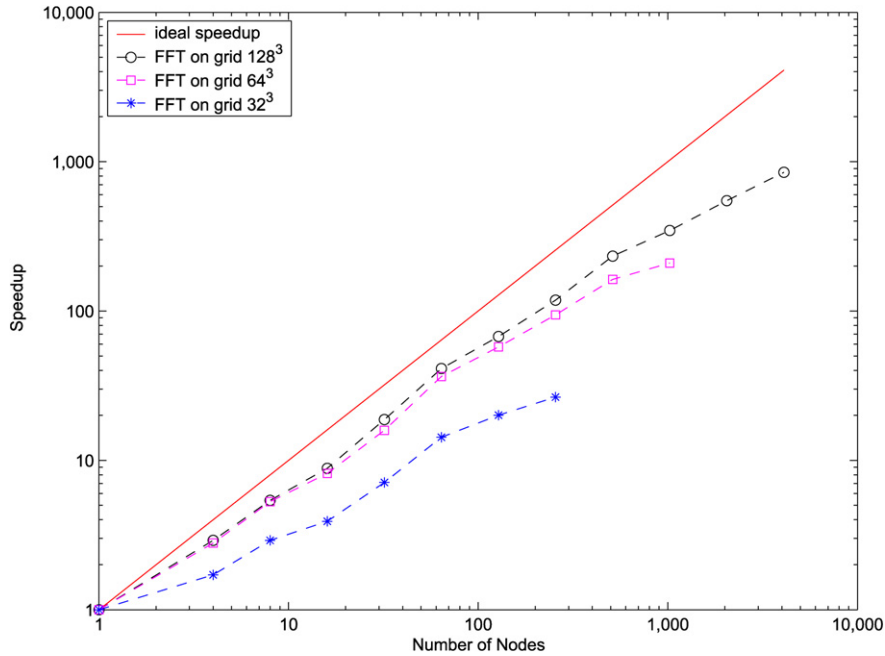


Fig. 2. Speed-up for 3D-FFT on grids  $32^3$ ,  $64^3$  and  $128^3$  for up to 4096 nodes.

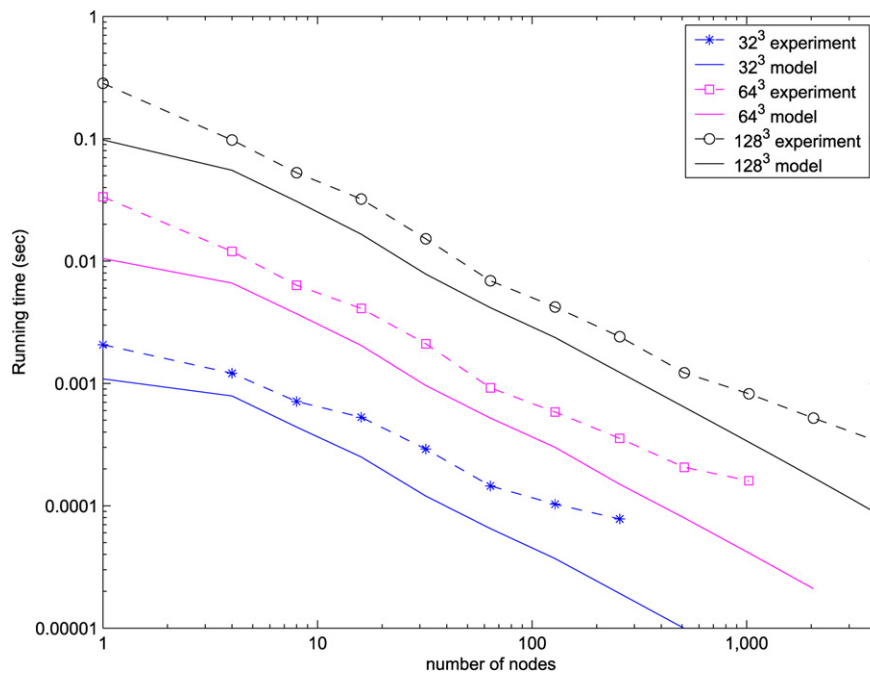


Fig. 3. The experiment and model performances, measured by the total running time in solving  $32^3$ ,  $64^3$  and  $128^3$  3D-FFTs on different systems, vs. number of nodes.

time is collected from the experiments of varying node numbers, with fixed problem sizes—strong scaling experiments.

Fig. 2 shows the speedups of three sets of experiments for three problem sizes:  $32^3$ ,  $64^3$  and  $128^3$ .

Fig. 3 shows the actual running time compared with our models. In the  $32^3$  case, all original data can fit into the fast EDRAM, the sequential performance on one node is much better than that for  $64^3$  and  $128^3$  cases, for which the data have to be stored in slower DDR memory. So we notice the deviation of

the experiment performance from the model, for the  $32^3$  case, is smaller than that for the other two cases. The performance for larger systems deviates from the model due to memory hierarchy effect, i.e. data prefetching from the two types of memory at different speeds. Also, the store-and-forward routing of the system makes intra-memory data transfer inevitable. The existence of non-overlapping of communication and either memory access or sequential computation, as shown in Part C of Fig. 1, causes additional deviation from our model. Table 1 lists the

Table 1  
The modeled communication time, the measured non-overlapped 3D-FFT computation time and memory hierarchy time are shown, for problem of size  $128^3$  on 64 and 512-node systems, in addition to their actual running times

	$T_{\text{model}}$ ( $10^{-3}$ sec)	$T_{\text{non-overlapped FFT}}$ ( $10^{-3}$ sec)	$T_{\text{memory hierarchy}}$ ( $10^{-3}$ sec)	$T_{\text{actual performance}}$ ( $10^{-3}$ sec)
64 nodes	41.6	7.2	15.9	69.0
512 nodes	6.4	0.6	2.2	12.2

Table 2  
QCDOC and BlueGene/L hardware parameters are listed. These parameters are all relevant to the parallel 3D FFT computation

	Dimensionality	Processor clock speed	Bandwidth per link	Message passing interface	Communication routing
QCDOC	6-dimensional torus $p_x \times 2 \times p_y \times 2 \times p_z \times 2$	400 MHz	1 bit/clock cycle	QMP	Store-and-forward routing
Blue Gene/L	3-dimensional torus $b_x \times b_y \times b_z$	700 MHz	2 bits/clock cycle	1. MPI 2. Active packet	Cut-through routing

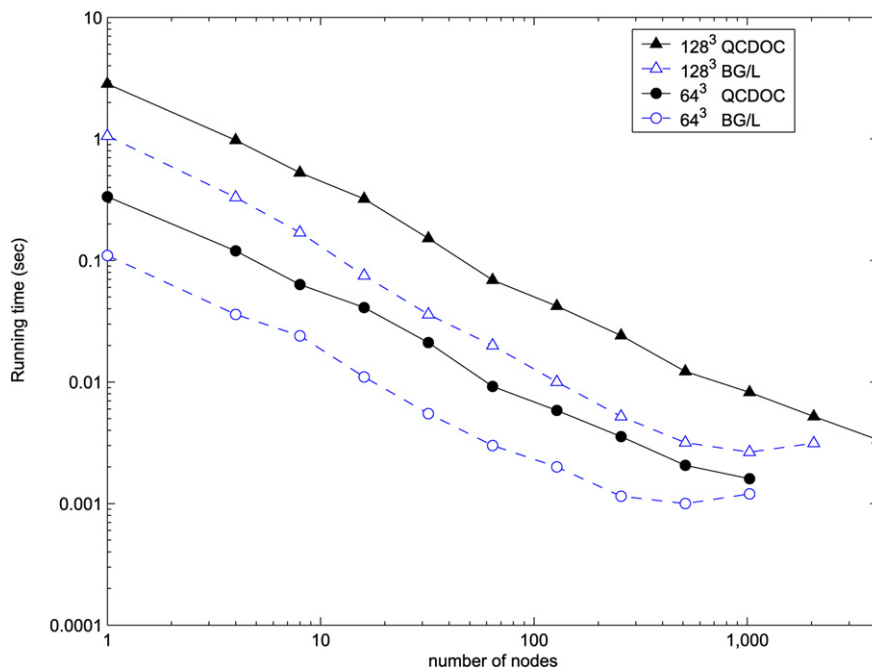


Fig. 4. The 3D FFT performance comparison between QCDOC and BlueGene/L, for problems of size  $64^3$  and  $128^3$ .

time attributed to each of these two parts and the modeled communication time for a problem of size  $128^3$  using 64 and 512-node systems.

For asymptotic analysis of scalability, the bandwidth-only model is inadequate because of the increasing hardware and software overheads for larger systems. For example, with more than 512 nodes, the software and hardware overheads in communication become significant and prevent the running-performance from increasing, as evident in Table 1. On the 64-node system, the running time for the non-overlapped memory access and the 3D-FFT computation is more significant in the actual performance than it is on the 512-node system. The software and hardware overheads for the 512-node system can no longer be neglected. Even with the increased overhead, our

algorithm can scale well up to 4096 nodes for problems of size  $128^3$ , as shown in Fig. 3.

#### 4.3. Comparison with volumetric 3D FFT on BlueGene/L

In [35,36], Eleftheriou et al. presented the volumetric 3D FFT method and its performance on BlueGene/L [33,34]. Here, we compare the performance obtained on QCDOC with that on BlueGene/L. The performance on QCDOC is obtained from BNL's 512-node system running at 400 MHz, while the performance on BlueGene/L is quoted from [35].

Table 2 lists hardware parameters for QCDOC and BlueGene/L, which are relevant to parallel 3D-FFT computation. Fig. 4 represents, in solving 3D FFTs of size  $64^3$  and  $128^3$ , the comparison of the actual performances of our FFT run on

QCDOC, and Eleftheriou et al.'s run on BlueGene/L. The efficiency deviation is within the factor of hardware difference, according to Table 2. For bigger machines of more than 1000 nodes, the better scalability of our method on QCDOC is apparent.

## 5. Conclusions

We present our approach for mapping 3D data arrays to QCDOC supercomputer with 6D tori network for efficient 3D-FFT performance. The communication-and-computation overlapping strategy, coupled with the data distribution and the added communication channels, gives us very favorable performance compared with IBM BlueGene/L. On BlueGene/L, the parallel 3D-FFT scales linearly up to 1024 nodes and well up to about 2048 nodes, while on QCDOC the performance for problems with the same size scales beyond 4096 nodes, which is the largest system we have available. We speculate this scalability will extend beyond tens of thousands of nodes on QCDOC based on performance modeling. This suggests much broader application space for QCDOC than was originally designed for QCD and that future architectures consider the use of a 6D torus.

Additionally, the model and experimental performances and their discrepancies are given. We observe that there are two major reasons causing these discrepancies from the bandwidth-based model. One is the memory hierarchy effect, including the time for storing and forwarding data; the other is the non-overlapped sequential 1D-FFT computation.

Just as Volumetric 3D-FFT has been applied in Blue Matter [40] on BlueGene/L as the core part to give the accurate evaluation of the long-range interaction for complex biological systems, we anticipate our parallel 3D-FFT framework could also be used to enhance our parallel MD software, MDoc, which runs well on QCDOC. We expect that much better scalability could be achieved in classic MD simulation for two reasons. First, MD needs real to complex FFT computation, which can save half of the communication of its complex to complex counterpart. Second, interpolation error makes the implementation of a spherical cutoff in the 3D-FFT possible which reduces more communication yet further [41].

## References

- [1] O. Buneman, D.A. Dunn, Computer experiments in plasma physics, *Sci. J. (U.K.)* 2 (1966) 34.
- [2] A.H. Taub, S. Fernbach, *Computers and their Role in the Physical Sciences*, Gordon and Breach, New York, 1970.
- [3] C. Simmerling, B. Strockbine, A.E. Roitberg, All-atom structure prediction and folding simulations of a stable protein, *J. Am. Chem. Soc.* 124 (2002) 11258.
- [4] K. Kholmurodov, W. Smith, K. Yasuoka, T. Darden, T. Ebisuzaki, A smooth-particle mesh Ewald method for DL\_POLY molecular dynamics simulation package on the Fujitsu VPP700, *J. Comput. Chem.* 21 (2000) 1187.
- [5] J. Norberg, L. Nilsson, On the truncation of long-range electrostatic interactions in DNA, *Biophys. J.* 79 (2000) 1537.
- [6] C. Sagui, T.A. Darden, Molecular dynamics simulations of biomolecules: Long-range electrostatic effects, *Annu. Rev. Biophys. Biom.* 28 (1999) 155.
- [7] D.M. York, W.T. Yang, H. Lee, T. Darden, L.G. Pedersen, Toward the accurate modeling of DNA—the importance of long-range electrostatics, *J. Am. Chem. Soc.* 117 (1995) 5001.
- [8] T.A. Darden, A. Toukmaji, L.G. Pedersen, Long-range electrostatic effects in biomolecular simulations, *J. Chim. Phys. Pcb.* 94 (1997) 1346.
- [9] A.R. Leach, *Molecular Modelling: Principles and Applications*, Prentice-Hall, Harlow, England; Reading, MA, 2001.
- [10] R.W. Hockney, J.W. Eastwood, *Computer Simulation using Particles*, McGraw-Hill International Book Co., New York, 1981.
- [11] M.P. Allen, D.J. Tildesley, *Computer Simulation of Liquids*, Clarendon Press, Oxford University Press, Oxford [England], 1989.
- [12] T. Darden, D. York, L. Pedersen, Particle Mesh Ewald—an  $N \log(N)$  method for Ewald sums in large systems, *J. Chem. Phys.* 98 (1993) 10089.
- [13] M. Deserno, C. Holm, How to mesh up Ewald sums. I. A theoretical and numerical comparison of various particle mesh routines, *Journal of Chemical Physics* 109 (1998) 7678.
- [14] M. Deserno, C. Holm, How to mesh up Ewald sums. II. An accurate error estimate for the particle-particle-particle-mesh algorithm, *Journal of Chemical Physics* 109 (1998) 7694.
- [15] U. Essmann, L. Perera, M.L. Berkowitz, T. Darden, H. Lee, L.G. Pedersen, A smooth particle mesh Ewald method, *Journal of Chemical Physics* 103 (1995) 8577.
- [16] B.A. Luty, M.E. Davis, I.G. Tironi, W.F. Vangunsteren, A comparison of particle-particle, particle-mesh and Ewald methods for calculating electrostatic interactions in periodic molecular-systems, *Molecular Simulation* 14 (1994) 11.
- [17] E.L. Pollock, J. Glosli, Comments on P(3)M, FMM, and the Ewald method for large periodic Coulombic systems, *Computer Physics Communications* 95 (1996) 93.
- [18] G.J. Martyna, M.E. Tuckerman, A reciprocal space based method for treating long range interactions in ab initio and force-field-based calculations in clusters, *Journal of Chemical Physics* 110 (1999) 2810.
- [19] P. Minary, J.A. Morrone, D.A. Yarne, M.E. Tuckerman, G.J. Martyna, Long range interactions on wires: A reciprocal space based formalism, *Journal of Chemical Physics* 121 (2004) 11949.
- [20] P. Minary, M.E. Tuckerman, K.A. Pihakari, G.J. Martyna, A new reciprocal space based treatment of long range interactions on surfaces, *Journal of Chemical Physics* 116 (2002) 5351.
- [21] R.N. Bracewell, *The Fourier Transform and its Applications*, McGraw-Hill, New York, 1986.
- [22] E.O. Brigham, *The Fast Fourier Transform*, Prentice-Hall, Englewood Cliffs, NJ, 1974.
- [23] A. Edelman, P. Mccorquodale, S. Toledo, The future fast Fourier transform? *SIAM Journal on Scientific Computing* 20 (1999) 1094.
- [24] M. Frigo, S.G. Johnson, The design and implementation of FFTW3, *Proceedings of the IEEE* 93 (2005) 216.
- [25] M. Frigo, S.G. Johnson, FFTW: An adaptive software architecture for the FFT, in: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 3, 1998, p. 1381.
- [26] C. Van Loan and Society for Industrial and Applied Mathematics, *Computational Frameworks for the Fast Fourier Transform*, SIAM, Philadelphia, 1992.
- [27] P. Boyle, D. Chen, N. Christ, M. Clark, S. Cohen, C. Cristian, Z. Dong, A. Gara, B. Joo, C. Jung, C. Kim, L. Levkova, X. Liao, G. Liu, S. Li, H. Lin, R. Mawhinney, S. Ohta, K. Petrov, T. Wettig, A. Yamaguchi, The QCDOC project, *Nuclear Physics B* 140 (Proc. Suppl.) (2005) 169.
- [28] P.A. Boyle, D. Chen, N.H. Christ, M.A. Clark, S.D. Cohen, C. Cristian, Z. Dong, A. Gara, B. Joo, C. Jung, C. Kim, L.A. Levkova, X. Liao, G. Liu, R.D. Mawhinney, S. Ohta, K. Petrov, T. Wettig, A. Yamaguchi, Overview of the QCDSF and QCDOC computers, *IBM Journal of Research and Development* 49 (2005) 351.
- [29] D. Chen, N.H. Christ, C. Cristian, Z. Dong, A. Gara, K. Garg, B. Joo, C. Kim, L. Levkova, X. Liao, R.D. Mawhinney, S. Ohta, T. Wettig, QCDOC: A 10-teraflops scale computer for lattice QCD, *Nuclear Physics B* 94 (Proc. Suppl.) (2001) 825.

- [30] P. Rissland, Y. Deng, Structure and Performance of molecular dynamics package MDoC on QCDOC, Parallel Computing, in press. Available online January 2007.
- [31] P.D. Haynes, M. Cote, Parallel fast Fourier transforms for electronic structure calculations, *Computer Physics Communications* 130 (2000) 130.
- [32] M. Frigo, S.G. Johnson, The Fastest Fourier transform in the west, Technical Report, 1997.
- [33] A. Gara, M.A. Blumrich, D. Chen, G.L.T. Chiu, P. Coteus, M.E. Giampapa, R.A. Haring, P. Heidelberger, D. Hoenicke, G.V. Kopsay, T.A. Liebsch, M. Ohmacht, B.D. Steinmacher-Burow, T. Takken, P. Vranas, Overview of the Blue Gene/L system architecture, *IBM Journal of Research and Development* 49 (2005) 195.
- [34] N.R. Adiga, M.A. Blumrich, D. Chen, P. Coteus, A. Gara, M.E. Giampapa, P. Heidelberger, S. Singh, B.D. Steinmacher-Burow, T. Takken, M. Tsao, P. Vranas, Blue Gene/L torus interconnection network, *IBM Journal of Research and Development* 49 (2005) 265.
- [35] M. Eleftheriou, B.G. Fitch, A. Rayshubskiy, T.J.C. Ward, R.S. Germain, Scalable framework for 3D FFTs on the Blue Gene/L supercomputer: Implementation and early performance measurements, *IBM Journal of Research and Development* 49 (2005) 457.
- [36] M. Eleftheriou, J.E. Moreira, B.G. Fitch, R.S. Germain, A volumetric FFT for BlueGene/L, *High Performance Computing—HIPC 2003* 2913 (2003) 194.
- [37] QMP, LQCD Message Passing API, version 2.0, 2004.
- [38] P.A. Boyle, D. Chen, N.H. Christ, M. Clark, S.D. Cohen, C. Cristian, Z. Dong, A. Gara, B. Joo, C. Jung, C. Kim, L. Levkova, X. Liao, S. Li, H. Lin, G. Liu, R.D. Mawhinney, S. Ohta, K. Petrov, T. Wettig, A. Yamaguchi, The status of user software on QCDOC, *Nuclear Physics B* 140 (Proc. Suppl.) (2005) 829.
- [39] V. Kumar, *Introduction to Parallel Computing: Design and Analysis of Algorithms*, Benjamin/Cummings Pub. Co., Redwood City, CA, 1994.
- [40] B.G. Fitch, R.S. Germain, M. Mendell, J. Pitera, M. Pitman, A. Rayshubskiy, Y. Sham, F. Suits, W. Swope, T.J.C. Ward, Y. Zhestkov, R. Zhou, Blue Matter, an application framework for molecular simulation on Blue Gene, *Journal of Parallel and Distributed Computing* 63 (2003) 759.
- [41] B. Fang, Y. Deng, G.J. Martyna, A fine grained parallel smooth particle mesh Ewald algorithm for biophysical simulation studies: Application to the 6D torus QCDOC supercomputer, *Comput. Phys. Comm.* (2007), in press.