

# The BlueGene/L Supercomputer and Quantum ChromoDynamics

Pavlos Vranas, vranasp@us.ibm.com,  
Gyan Bhanot, gyan@us.ibm.com,  
Matthias Blumrich, blumrich@us.ibm.com,  
Dong Chen, chendong@us.ibm.com  
Alan Gara, alangara@us.ibm.com  
Philip Heidelberger, philiph@us.ibm.com  
Valentina Salapura, salapura@us.ibm.com  
James C. Sexton, sem.comxtonjc@us.ibm.com

IBM T.J. Watson Research Laboratory  
Yorktown Heights, New York

## Abstract:

We describe our methods for performing quantum chromodynamics (QCD) simulations that sustain up to 20% of the peak performance on BlueGene supercomputers. We present our methods, scaling properties, and first cutting edge results relevant to QCD. We show how this enables unprecedented computational scale that brings lattice QCD to the next generation of calculations. We present our QCD simulation that achieved 12.2 Teraflops sustained performance with perfect speedup to 32K CPU cores. Among other things, these calculations are critical for cosmology, for the heavy ion experiments at RHIC-BNL, and for the upcoming experiments at CERN-Geneva. Furthermore, we demonstrate how QCD dramatically exposes memory and network latencies inherent in any computer system and propose that QCD should be used as a new, powerful HPC benchmark. Our sustained performance demonstrates the excellent properties of the BlueGene/L system.

## 1) Introduction

Quantum Chromo Dynamics (QCD), chroma in Greek means color, is the theory of the strong nuclear force that binds the constituents of sub-nuclear matter, quarks and gluons, together to form stable nuclei and therefore more than 90% of the visible matter of the Universe. Being able to calculate quantities in QCD is of paramount importance for many reasons. Cosmological models depend on the mechanism that transformed the primordial soup of quarks and gluons to nuclei. QCD predicts that a phase transition occurs between a quark gluon plasma and stable nuclear matter at a temperature of about 170 MeV (about 2 trillion degrees Celsius). This transition is thought to have occurred at about  $10^{-5}$  sec (10 millionths of a second) after the Big Bang. Currently, an attempt to recreate the reverse of this process is in progress at the Brookhaven National Laboratory's RHIC accelerator. This experiment, as well as the next generation heavy ion collision experiments at RHIC and at CERN-Geneva, need theoretical input from first principles QCD to be able to interpret the wealth of

results that have been and will be generated. Finally, at CERN, the LHC accelerator will shortly begin to probe energies beyond our current understanding. To be able to do this the colliding beams are made out of nuclear particles. Therefore, in order to interpret the results of the collisions one will have to be able to accurately separate the QCD contributions from the new high energy physics processes.

In our work we use the methods of Lattice Gauge Theory to perform numerical simulations of Lattice QCD (LQCD). Every time there is a serious increase in computational speed coming from a massively parallel supercomputer, LQCD breaks new ground in the understanding of sub-nuclear matter. However, enormous amounts of computational speed are not enough to enable the next generation of LQCD calculations. To do so, one must also implement new scientific methods and algorithms that are hardware independent as well as methods that enable sustained speeds that achieve a considerable fraction of peak in the fastest supercomputers.

In this work we show how we programmed QCD for the dual-core BlueGene/L (BG/L) supercomputer and achieved a performance of 20% of peak. We present the weak and strong scaling properties of our implementation and demonstrate that Blue Gene/L is a computer of choice for LQCD. Indeed an unprecedented amount of Blue Gene/L Teraflops for LQCD have already been deployed around the world using our LQCD kernel code directly or as a starting point for further improvement and experimentation.

Furthermore, we will explain the cutting edge scientific methods that we used, and in particular the Domain Wall Fermion (DWF) method. This method actually uses a fifth space time dimension to bring the most notorious systematic errors present in LQCD due to the fermion doubling problem under control. A glimpse on our first results is presented.

Also, we show that QCD is a rather unique application because it has such a substantial dependence on memory and network latencies. In this sense, it provides a powerful tool for evaluating computer systems from the latency standpoint.

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.  
SC2006 November 2006, Tampa, Florida, USA  
0-7695-2700-0/06 \$20.00 ©2006 IEEE

---

We propose that the sustained performance of the QCD kernel should be considered as a standard benchmark for high-performance computing. We expect QCD to be particularly valuable for evaluating systems based on chip multiprocessors, where latency will dominate as more and more processors compete for limited memory and network resources.

The paper is organized as follows: In section 2 we review the overall BlueGene/L architecture. In section 3 we give a brief overview of the LQCD algorithm paying special attention to the most crucial part of the algorithm and its most recent implementation, the five-dimensional Domain Wall Fermions. We identify the relatively small LQCD kernel piece (Dslash) that consumes more than 90% of the cycles and explain how Dslash depends crucially on memory and network latencies making it a powerful HPC benchmark. In section 4 we describe how we optimize D-slash for BlueGene/L for the demanding dual-core mode (virtual node mode) and how our methods naturally enable the embedding of LQCD on the multi-core chips that are beginning to emerge as a dominant technology. This allows LQCD to maintain its record of perfect weak-scalability. In section 5 we present our most recent simulation sustaining 12.2 TFLOPS and our historic 2004 simulation that was the first to cross the 1 Teraflops landmark. In section 6 we give our performance results and power measurements that demonstrate the power/performance efficiency of BlueGene/L for QCD. In section 7 we present our simulation of the LQCD phase transition using DWF. We drive our conclusions in section 8.

## 2) An Overview of the BlueGene/L Hardware

In this section we give a brief overview of the BlueGene/L hardware. A full issue of the IBM Journal for R&D has been devoted to BG/L and the reader is referred there for more details [**IBM BG/L 2005**].

The chip has two embedded IBM 440 CPU cores, each paired with a double floating point unit (DFPU) capable of two multiply/add instructions per cycle. Therefore the chip is capable of a peak of 8 floating point instructions per cycle. At the operation speed of 700 MHz the chip delivers 5.6 GFlops of peak speed.

The DFPU has two separate register files (primary and secondary). Each has 32, 64-bit floating point registers. There are floating-point instructions that allow load/store and manipulation of all registers. These instructions are an extension to the PowerPC Book E instruction set. The DFPU is ideal for complex arithmetic. If a DFPU register is loaded and then read within the next 4 cycles, a bubble is introduced into the pipeline. Therefore for optimal code, the registers should be used with some care in order to avoid pipeline bubbles.

The on-chip memory hierarchy is as follows: The first-level caches within the 440 cores are not kept coherent by the hardware, but the remainder of the memory system is. Each core has separate 32 Kbyte instruction and data caches. The second-level (L2) caches are very small and serve primarily as sophisticated, multi-stream sequential prefetchers for data (instruction fetches go around L2). The third-level (L3) cache is 4-way set-associative and is constructed with a fast SRAM tag store and a large, 4 Mbyte embedded DRAM data store. Finally, a DDR controller connects the L3 to external DRAM, which can be 512 Mbytes or 1 Gbyte in size. Because of the L2 prefetcher, very high bandwidth can be achieved for code that is organized to maximize the length of sequential streams.

Five network controllers are built in on the chip. The main network (torus) connects the nodes in a 3-dimensional torus grid with nearest neighbor high speed interconnects. The hardware implements a sophisticated dynamic virtual cut-through network. Hardware packets can be sent from any node to any other node without CPU intervention. Furthermore, the nodes are also connected via a global collective network that can efficiently perform global operations as well as file I/O. Also, the nodes are connected via separate interrupt and control networks. The system is completely symmetric with respect to the two cores.

To create the BlueGene/L system, two node chips are assembled on a compute-card, and 16 compute-cards are plugged into a single node-card. Then 32 node cards are plugged into two vertical midplane cards to create a 1024-node rack, which is housed in a refrigerator-sized cabinet. Many racks can be connected with cables to form large systems. Also, a single rack can be configured as an 8x8x16 torus or as two 8x8x8 torus grids without re-cabling.

## 3) The Lattice QCD Algorithm

The LQCD simulation overall algorithmic steps are well established [**Creutz 1986; Montvay and Munster 1993**]. Here we describe the algorithm briefly at a general level. The full LQCD code is tens of thousands lines long. Standard packages exist that can be straight-forwardly compiled for BlueGene/L. Here we use the Columbia University Physics System code version [**CPS**].

In LQCD, more than 90% of the cycles are consumed by a small kernel routine which is about 1000 lines long. This kernel goes by the name D-slash. The reported LQCD performance has historically been given as the performance of D-slash. Obviously, it is extremely worthwhile to highly optimize D-slash. Typically compilers do a poor job. In most cases this kernel is hand-coded in assembly for the particular hardware. Here we describe our hand-coded implementation of the D-slash kernel for BG/L.

LQCD uses a discretization of the 4-dimensional space time to a grid of points connected with links. Typically, a hypercubic grid is used with lattice spacing denoted by “a” (the distance between two neighboring lattice points). The quarks (q) live on the lattice points while the gauge fields (U) live on the links that connect the points and are responsible for “carrying” the interaction between the quarks. Cast in this form, the theory becomes very similar to a statistical physics system and can be simulated in very similar ways. In what follows, a quark field (q) denotes the collection of the numbers needed to describe a quark at all lattice points. Similarly for the gauge field (U).

The theory of QCD dictates the energy function. A U field consists of four 3x3 complex matrices on every site of the lattice. If the number of lattice sites in the 4-dimensional volume is V then a U field needs  $3 \times 3 \times 2 \times 4 \times V = 72 \times V$  floating point numbers to be described. Typically V is distributed over the parallel supercomputer’s grid of nodes. The gauge field, also called a “configuration”, is “check-pointed” with some frequency since it is a minimal set of numbers from which the simulation can be re-started.

The algorithm “evolves” the U and q fields so that the energy function remains constant. This is done using molecular-dynamic techniques. The evolution is composed of a series of small update steps for the U and q fields. In each update step the D-slash operator is inverted. Once the fields are evolved a wealth of quantities of interest are calculated based on the field values (for example one can calculate the temperature, the masses of particles, the energy etc.). Once the measurements are done another evolution can be started. Typically hundreds to thousands of evolution/measurement steps are performed so that large statistics of the measured quantities is collected. The data is then analyzed “off-line” with standard statistical methods.

Since D-slash is inverted for each update step it is used an enormous number of times. However, it is used in an operational way so that one does not have to store D-slash. One only needs a routine that performs the action of D-slash on an input column vector Y. Furthermore D-slash is a sparse matrix. It only connects nearest neighbor sites. This simplifies the communication patterns.

### 3.1) Lattice Fermions and the 5-Dimensional Domain Wall Method

The way the quark fields are represented on the lattice is complex. The particular implementation is reflected in the specific form of D-slash used in the inner-most loop.

When a continuum theory is regularized on a 4-dimensional lattice the original number of fermion species is increased by  $2^4$ , with a net chirality of zero. This is the well-known fermion doubling problem [Nielsen and Ninomiya 1981]. For vector theories like QCD, this problem has been traditionally treated at the expense of

exact chiral symmetry by using Wilson [Wilson 1975] or Kogut-Susskind [Kogut and Susskind 1975] fermions. Chiral symmetry is broken for any non-zero lattice spacing ‘a’, but is recovered together with Lorentz symmetry as ‘a’ is decreased to zero resulting in the correct target theory in the continuum. However, since any numerical simulation is done at non-zero ‘a’, one must be able to simulate at small enough ‘a’ so that the effect of the breaking does not obscure the underlying physics. As mentioned in the introduction, the problem is that the computational cost of decreasing ‘a’ is large (decreasing ‘a’ by a factor of 2 requires  $2^8$  to  $2^{10}$  more computations).

In recent years an alternative lattice fermion method, domain wall fermions (DWF), has been developed and used in many QCD simulations. The method was introduced in [Kaplan 1992]. The first numerical simulations using DWF were done in [Vranas 1998]. Domain wall fermions are defined by extending space-time to five dimensions. A non-zero five dimensional mass  $m_0$  is present. The size of the fifth dimension is  $L_5$  and free boundary conditions for the fermions are implemented. As a result, the two chiral components of the Dirac fermion are separated. One chirality is bound exponentially on one wall and the other on the opposite wall. For any lattice spacing the two chiralities mix only by an amount that decreases exponentially as  $L_5$  tends to infinity. At  $L_5 = \text{infinity}$  chiral symmetry is exact even at non-zero ‘a’. In this way the continuum ‘a’ vanishing to zero, limit has been separated from the chiral  $L_5$  tending to infinity limit. Furthermore, the computing cost increases only linearly with  $L_5$ . Therefore, this method provides a new practical “knob” on the amount of chiral symmetry breaking due to the lattice discretization. Our simulations presented here are the first ones to use DWF on large size lattices on a general purpose supercomputer.

The DWF implementation of D-slash uses  $L_5$  applications of a simpler 4-dimensional D-slash operator called the Wilson D-slash kernel. Furthermore, in order to improve the convergence properties of the inverter routine, a particular form of that operator called even/odd preconditioned Wilson D-slash is used. In this paper, we will further refer to it as D-slash.

### 3.2) The Kernel of LQCD as a Powerful Latency Benchmark

The operational form of the Wilson D-slash is given below. This embodies the quark kinematics and the quark interactions with the gluons. For the purposes of this paper the details of these equations are not important (for more details please see [Creutz 1986; Montvay and Munster 1993]).

$$D(x,y) = \frac{1}{2} \sum_{\mu=1, \dots, 4} [U_{\mu}(x) (1 + \gamma_{\mu}) \delta(x+\mu,y) + U_{\mu}^{\dagger}(x-\mu) (1 - \gamma_{\mu}) \delta(x-\mu,y) ]$$

The  $U_\mu$  are  $3 \times 3$  complex matrices. The  $\gamma_\mu$  are  $4 \times 4$  complex spinor matrices.  $\delta(x,y) = 1$  if  $x=y$  and 0 otherwise. The operation

$$\Psi(x) = D(x,y) \Psi(y)$$

is discussed where the fermion field  $\Psi(x)$  is defined on every site of an  $L_x, L_y, L_z, L_t$  lattice and is a complex vector with a color index that takes 3 values and a spin index that takes 4 values (24 floating point numbers). Because the matrices  $(1 + \gamma_\mu)/2$ ,  $(1 - \gamma_\mu)/2$  are projection operators from the 4 component spinors to 2 component spinors one can perform that projection, do the calculations and communications using 2 component spinors (12 floating point numbers) and then reconstruct the final 4 component spinor. That method is called spin projection. Both the gauge and fermion fields are stored in memory with the real/imaginary, spin, color indices in sequence. All calculations are in double precision. In a simplified form the steps are:

**Loop over the site “x” of the left hand side of the above equation{**

- 1) Load  $\Psi$  = load 24 doubles (sequential).
- 2) Spin project  $\Psi$  into  $\Phi$ . This involves only additions and can therefore only perform at most at 50% of peak since the machine has fused multiply/add hardware. Total operations are 16.
- 3) If  $\Phi$  is not on this node do a nearest neighbor communication of 12 doubles.
- 4) Store  $\Phi$  (12 sequential stores).
- 5) Load  $U$  = load 18 doubles (sequential).
- 6) Wait for the communication to complete.
- 7) Multiply the matrix  $U$  with the column vector  $\Phi$ . Total operations are 72.
- 8) Repeat steps 1-7 a total of 8 times (one for each term in  $D$ ). Notice that in each repetition the  $\Psi$  fields are not sequential. The  $U$  fields are sequential for the first set of 4 terms and for the second set of 4 terms but are not sequential between the two sets.
- 9) Load each temporary  $\Phi$  (12 double load).
- 10) From  $\Phi$  reconstruct the 24-double component fields and sum them up to form the resulting  $\Psi(x)$ .

**}**

Since this is a 4-dimensional problem, the fields are not sequential between the loop iterations. Furthermore, we use the even/odd preconditioning approach, since this approach has been shown to improve inverter times by more than a factor of two. Even/odd preconditioning means that D-slash acts on all even sites (sites with the sum of their coordinates being an even number) first and then acts on all odd sites. It is applied twice on half the lattice each time.

It is obvious from the above that memory accesses are sequential only in very small chunks. Similarly, communication accesses are for very small data chunks.

Both memory accesses and communications are interspersed with calculations that can not be rearranged. The application is very “bursty”.

This type of computation is therefore extremely sensitive to both memory and communication latencies. Given that LQCD simulations are on the scientific frontier, this application property makes LQCD an ideal benchmark for evaluating computer systems. This is particularly important in the modern multi-core chips where memory and network resources are shared by many CPUs. A well designed machine should not only achieve high bandwidths, but also latencies that are small enough so that they are compensated by the number and computational power of the CPUs.

We propose that the performance of the QCD even/odd preconditioned Wilson D-slash operator on a  $4 \times 4 \times 4 \times 4$  lattice per CPU serves as an HPC benchmark. This is a powerful metric based on science that examines a crucial property of computer systems.

#### **4) Optimizing the High Performance LQCD Kernel for BlueGene/L**

Recently, the increase of computational speed comes by exploiting the CMOS scaling that allows placement of multiple CPU cores on a single chip, as opposed to the traditional approach of increasing the frequency and performance of a single core. It is therefore important to be able to still do QCD simulations that scale perfectly with the number of cores per chip as well as the more traditional scaling of larger and larger number of chips.

The best way to do so is the one that is natural for the science involved: one simply extends the lattice grid to exploit all CPU cores available. However, this has to be done in a way that the four dimensional grid is maintained as a grid of CPUs. In the BlueGene/L, there is a 3-dimensional cubic network with the topology of a torus connecting all chips. On each chip there are two CPUs. QCD can be embedded by using the three-dimensional network to embed three of the lattice dimensions, while the lattice points of the fourth dimension, time, can be spread out among the two CPUs. The two CPUs of a chip communicate with each other by exchanging messages via their common memory. In order to communicate along the three dimensional space, they both use the torus network. This configuration is akin to one dimensional fiber bundle on a three dimensional space.

As already mentioned, more than 90% of the computations in a LQCD simulation are done in a single kernel routine D-slash. It is therefore paramount to highly optimize this kernel. For BG/L, we took special care in order to take full advantage of the underlying hardware. This is a demanding task, since BlueGene/L is a general purpose supercomputer, not designed with QCD in mind. Also, BlueGene/L

achieves its peak speed by a double floating point multiply-add unit, making the memory and instruction staging particularly sensitive.

The QCD even/odd preconditioned Wilson D-slash kernel was hand-written in inline assembly. We are listing the kernel optimizations we employed to exploit BlueGene/L strength:

- 1) All floating point arithmetic uses double multiply/add instructions. We have paired all multiplications with additions in sets of two. The complex arithmetic of QCD makes this natural.
- 2) All floating point computations are carefully arranged to avoid pipeline conflicts. In particular, we do not use a register after it is loaded for at least 4 cycles by careful register management.
- 3) Memory storage ordering is chosen to maximize the size of consecutive accesses and minimize pointer arithmetic. We store the U and q fields so that they can be accessed sequentially with respect to the real/imaginary, color, and spin indices.
- 4) Quad Load/stores (quad = two double) are carefully arranged to take advantage of the cache hierarchy and the CPUs ability to issue up to 3 outstanding loads.
- 5) We overlap computations almost fully with load/stores. A single core performance is limited by memory access latency to L3 cache. By carefully overlapping load/store instructions and computation, we minimize or eliminate the CPU waiting time.
- 6) Global sums are done via fast torus or collective network routines. These sums are done for every inverter iteration. For large machines the ability to do them on the collective network is crucial in maintaining perfect weak scaling.
- 7) We have implemented an effective nearest-neighbor communication layer, which interacts directly with the torus network hardware to do the data transfers. We poll directly the torus hardware, and we precalculate the hardware headers and write the header and data directly into the torus hardware. Furthermore, we overlap the local memory copy between the cores with the latency associated with the arrival of the first packet in every iteration.

Because in BG/L communications do not overlap with computations or memory accesses one has to be particularly careful with the communication layer. As it is shown in table 1, minimizing the communications layer latency is crucial for small local lattices. These small lattices are the most interesting since they determine the strong scalability limit and have the fastest time to solution on a large system. The latencies inherent in other communication layers, such as for example MPI, are too large and seriously limit the achievable performance. Only

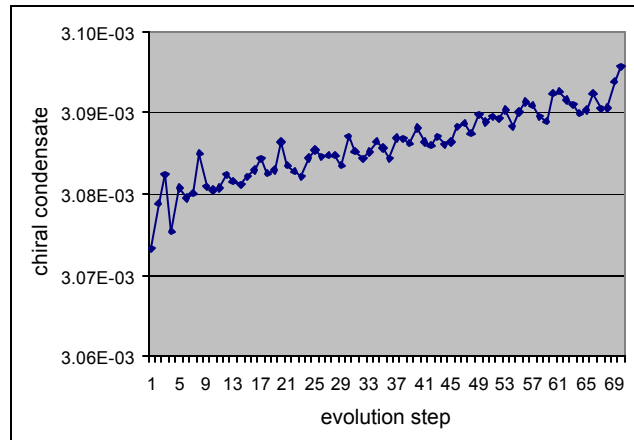
for lattices as large as  $8 \times 8 \times 8 \times 8$  per core MPI latency does not affect performance significantly.

For small local lattice size, most of the data can be located in BlueGene/L L1 cache and thus achieve fast access time, but small lattice size requires more relative communication since the surface to volume ratio is larger. On the other extreme, for large local lattice size data access latency is higher, since data are located in the slower L3 cache, but less communications is needed since the surface to volume ratio is smaller. These effects are evident in the strong scaling table 1.

Even/odd preconditioning for the D-slash operator improves inversion times by a factor larger than two, but it increases the amount of communication required compared to the non-preconditioned operator since it has a larger surface to volume ratio. This decreases the sustained performance but significantly improves the overall execution time.

### 5) Making History (Twice)

An historic moment for lattice QCD was reached when our code was the first ever to achieve 1 sustained Teraflops in June, 2004. Another historic moment was reached in April of this year (2006) when we simulated QCD with a 12.2 Teraflops sustained performance. The simulation was done on the 16-rack system at the IBM Watson Research Laboratory in Yorktown Heights, New York.



**Figure 1.** The chiral condensate from our 12.2 Teraflops simulation. The lattices is  $64^3 \times 16$ , the fifth dimension has 8 lattice points. There are two quarks with bare mass = 0.04. The gauge action is Iwasaki with inverse squared gauge coupling  $\beta=2.1$ .

The recent simulation was for QCD at high temperatures using Domain Wall Fermions. The total problem size was an unprecedented  $64^3 \times 16$  four-dimensional lattice (the lattice size per CPU was  $2 \times 2 \times 2 \times 8$ ) with a fifth dimension of 8 lattice sites. Two quark flavors were used with bare

mass of 0.04. The gauge action was Iwasaki with inverse squared gauge coupling  $\beta=2.1$ . The size of this simulation has long been a dream for QCD scientists.

In figure 1 we show a sample of our results. The chiral condensate (measuring the clustering of quarks) (y axis) is plotted as the simulation evolves (x axis) towards thermalization. This is the very first glimpse into QCD physics on such large lattices. This simulation took about one hour on the 16-rack BG/L system and it is only a sample. In order to obtain results for physical observables such as temperature and particle masses one would have to simulate for a few months. In the previous generation of 1 Teraflops supercomputers this would have required several years of simulation.

## 6) Performance

We have run our code on various local lattice sizes on a single node (two CPUs in virtual-node mode) and we present our results for strong scaling in Table 1. These represent strong scaling since the local problem size changes which is equivalent with keeping the global size fixed while changing the number of CPUs. The numbers are the percentage of peak that we sustain. One can see the performance with and without communications for various local lattice sizes. We also present the Conjugate Gradient inverter results for our kernel.

Lattice Size on a CPU	$2^4$	$4 \times 2^3$	$4^4$	$8 \times 4^3$	$8^2 \times 4^2$	$16 \times 4^3$
D-slash, No Comms	31.5	28.2	25.9	27.1	27.1	27.8
D-slash, With Comms	12.6	15.4	15.6	19.5	19.7	20.3
CG Inverter	13.1	15.3	15.4	18.7	18.8	19.0

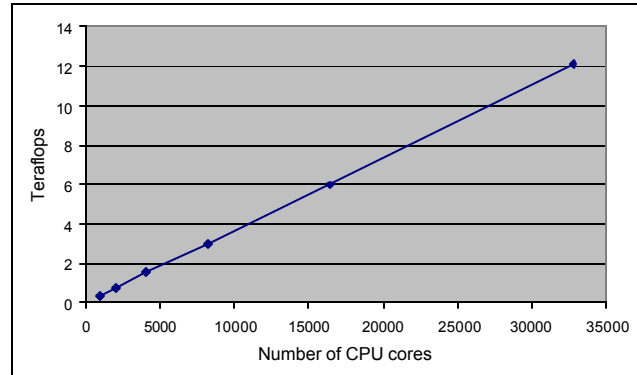
**Table 1.** Percentage of peak performance for strong scaling.

QCD is typically used as a weak scaling application, where the problem size is increased to hold the work per node constant. In keeping with this usage, we measured weak scaling.

Figure 2 shows self-normalized performance of QCD across a range of BG/L partition sizes starting at 1024 CPU cores. Perfect scaling corresponds to linear increase of total GFlops. The speedup relative to the 1024 CPU system is nearly perfect i.e. the 32K CPU system runs 32 times faster than the 1K CPU system.

In order to put the above performance results into context we must note that when we compiled an “out of the box” c-language code for D-slash we achieved performance which

was less than 4% of peak. Furthermore, the reader must also note that because there are more “additions” than multiplications in the D-slash kernel one can achieve at most about 75 % of peak in a fused multiply-add CPU such as the one used in BG/L. Even under perfect conditions our performance numbers could not have exceeded 75% of peak. This is an inherent limitation.



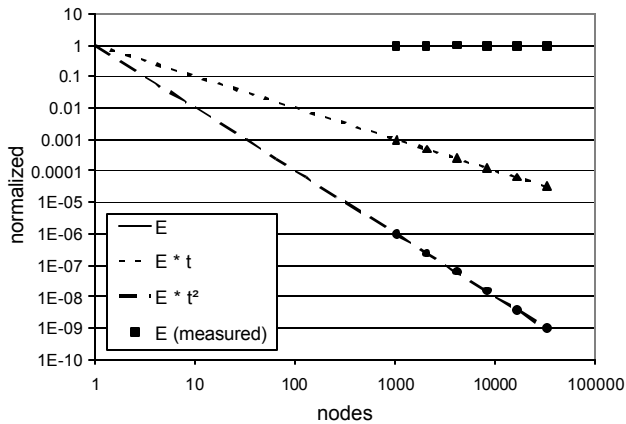
**Figure 2.** Weak scaling speedup of QCD on BG/L.

We have also analyzed the power/performance efficiency of BG/L for the QCD application. While peak performance numbers are an eye-catching metric, delivered power/performance on a scientific application is far more relevant. Thus, we analyzed workloads in detail to understand how well the BG/L system delivered on power/performance efficiency for QCD. More on BG/L power/performance analysis for a range of scientific applications can be found in [Salapura et al. 2005]. The most commonly used metric for characterizing energy efficiency is MIPS / Watt, corresponding to energy per operation. This metric does not assume that there is any benefit in speeding up computation, or cost for reducing its speed. Increasing the number of nodes, the energy per operation remains constant as performance per processor and power per processor remain unaffected.

However, the energy per operation metric does not reflect the benefit of reduced execution time. Gonzalez and Horowitz [Gonzalez et al. 1997] argue for the use of energy-delay product as a metric. This metric recognizes designs with reduced execution time for the same energy budget, where execution time reduction is achieved by a variety of techniques, such as better circuit design, architecture, or exploitation of parallelism. Executing an application on multiple nodes reduces execution time, without ideally increasing the power consumption per node, thus keeping energy consumption for a problem constant with reduced execution time.

Martin et al. propose the energy-delay<sup>2</sup> product as an efficiency metric for VLSI computation [Martin et al. 2001]. This metric reflects a better design point regardless of voltage -- this metric remains constant as a system is

voltage scaled to higher or lower performance. Typically, a system can be voltage scaled up to achieve higher performance while increasing power consumed. Thus, this metric is useful in considering tradeoffs between higher-frequency, higher voltage cores, and lower-frequency lower power cores to determine which system is more power efficient. The energy (denoted  $E$ ), energy-delay (denoted  $E*t$ ) and energy-delay<sup>2</sup> (denoted  $E*t^2$ ) metrics are closely related to the FLOPS<sup>n</sup>/W ratings, where energy  $E = 1 / (\text{FLOPS}/W)$ , energy-delay  $E*t = 1 / (\text{FLOPS}^2/W)$  and energy-delay<sup>2</sup>  $E*t^2 = 1 / (\text{FLOPS}^3/W)$ .



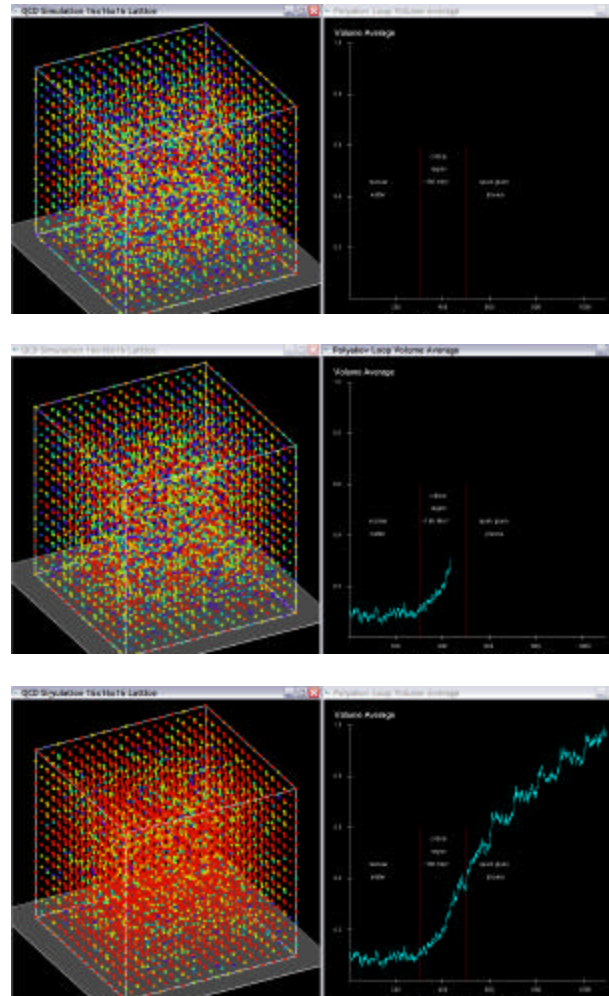
**Figure 3.** QCD power/performance scaling across a range of BlueGene/L partition sizes.

Figure 3 analyzes energy and energy-delay metrics for QCD for a range of Blue Gene/L partition sizes, where all three curves are self-normalized. Based on the perfect weak scaling behavior of QCD shown in figure 2, the overall energy consumption shows no increase as the problem scales to a bigger system. The dots in figure 3 represent the energy-delay metrics calculated using the performance results and based on the actual power measurements obtained for the partition sizes ranging from 512 to 16384 compute nodes (32768 CPU cores). The energy curve is flat indicating perfect energy scaling for the QCD application across the range of BlueGene/L partition sizes – unlike other applications which does not show perfect scaling due to multiprocessor overhead and communication pattern.

### 7) QCD at Very High Temperature

As mentioned in the introduction, QCD is the theory of the strong nuclear force. This force is mediated by the exchange of elementary bosonic particles called gluons that act on elementary fermionic particles called quarks. The force becomes very small as the quarks get close to each other (for example inside a proton). This property is called asymptotic freedom. However, when quarks try to separate, the force becomes extremely strong and increases linearly with the separation distance (confinement). Consequently, the quarks can never escape outside the nuclear particle.

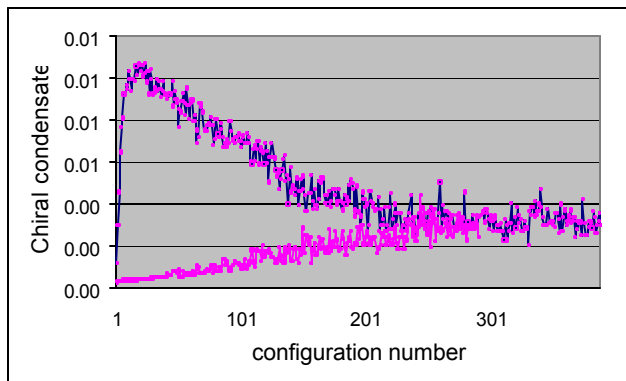
This property is responsible for the stability of nuclear matter, which constitutes most of the matter in the visible universe. When nuclear matter is heated to high temperatures (about 160-170 MeV = about 2 trillion degrees Celsius) it undergoes a phase transition (it melts) into a quark-gluon plasma. Currently, an attempt to recreate this condition is underway in accelerator laboratories. The reverse process is thought to have occurred in the early universe when the fireball from the big bang, containing a quark-gluon plasma, cooled-off and formed the stable nuclear matter and the cosmos as we know it.



**Figure 4.** Heating up nuclear matter above 2 trillion degrees Celsius. The temperature increases from top to bottom.

In Figure 4 we show a few frames of a movie clip that we made by simulating QCD and slowly changing the temperature. The frames are from below and above the transition from nuclear matter to the quark-gluon plasma. The frames show 2-flavor dynamical QCD on a 16x16x16x4 lattice with the DWF 5th dimension set to 24 sites. The pion mass in the simulation is about 400 MeV. The transition occurs around 164 MeV. On the left we

show the values of the order parameter (Polyakov loop) on the three-dimensional space. The color of each lattice point is the value of the Polyakov loop, which can fluctuate between -3 (blue) and 3 (red). Think of it as a spin system. The graph on the right shows the volume average of the Polyakov loop. This value is related to the single quark free energy. In the confined phase (low temperature) there are no free quarks and the value is low (not zero because of screening). In the quark-gluon plasma phase (high temperature), quarks can exist alone and the value is large. The vertical red lines indicate the transition region.



**Figure 5.** Thermalization simulation near the critical coupling of QCD using Domain Wall Fermions with  $L_s=32$  at  $\beta=2.0$ .

In Figure 5 we show a thermalization evolution for a  $16 \times 16 \times 16 \times 8$  lattice with  $L_s=32$  and pions with mass  $250 \pm 50$  MeV. This is the first simulation ever with a pion multiplet with such low masses.

## 8) Conclusions

We have optimized lattice QCD for the BlueGene/L supercomputer and efficiently exploited hardware features to achieve sustained speeds up to 20% of peak. We presented our methods, scaling properties, and first cutting edge results relevant to QCD. We showed how this enables unprecedented computational scale that brings QCD to the next generation of calculations. We presented our simulation that achieved 12.2 Teraflops sustained performance with perfect speedup to 32K CPU cores. QCD calculations are critical for cosmology, for the heavy ion experiments at RHIC-BNL, and for the upcoming experiments at CERN-Geneva, among others. Furthermore, we demonstrated how QCD dramatically exposes memory and network latencies inherent in any computer system and we proposed that QCD should be used as a new powerful HPC benchmark. Our sustained performance demonstrates the excellent properties of the BlueGene/L system

## 9) Acknowledgments

We are grateful to Dr. George Chiu of IBM Research for his input, encouragement, support, and guidance. We would also like to thank IBM Research and the IBM BlueGene/L team for their support.

## References

IBM Journal of R&D 2005. Vol 49, Number 2/3, March/May 2005.

Creutz M. 1986. Quarks Gluons and lattices, Cambridge Monographs in Mathematical Physics.

Monvay I. and Münster G. 1993. Quantum Fields on a Lattice, Cambridge Monographs in Mathematical Physics.

The Columbia Physics System.  
<http://www.epcc.ed.ac.uk/~ukqcd/cps/>

Nielsen H. B., Ninomiya M. 1981. The no-go theorem for chiral lattice fermions. Nucl. Phys. B185, 20.

Wilson K.G. 1975. New Phenomena in Subnuclear Physics. ed. A Zichichi (Plenum Press, New York), Part A, 69.

Kogut J., Susskind L. 1975. Staggered lattice fermions. Phys. Rev. D11, 395.

Kaplan D.B. 1992. Five dimensional lattice fermions. Phys. Lett. B288, 342.

Vranas P.M. 1998. Chiral Symmetry Restoration in the Schwinger Model with Domain Wall Fermions. Phys. Rev. D57, 1415.

Salapura V., Walkup R. and Gara A. 2005. Exploiting Workload Parallelism for Performance and Power Optimization. IBM Research report RC23724, September.

Gonzalez R., Gordon B., and Horowitz M. 1997. Supply and threshold voltage scaling for low power CMOS. IEEE Journal of Solid State Circuits, 32(8):1210-1216, August.

Martin A., Nystroem M., and Penzes P. 2001. ET2: a metric for time and energy efficiency of computation. Power-Aware Computing. Kluwer Academic Publishers.