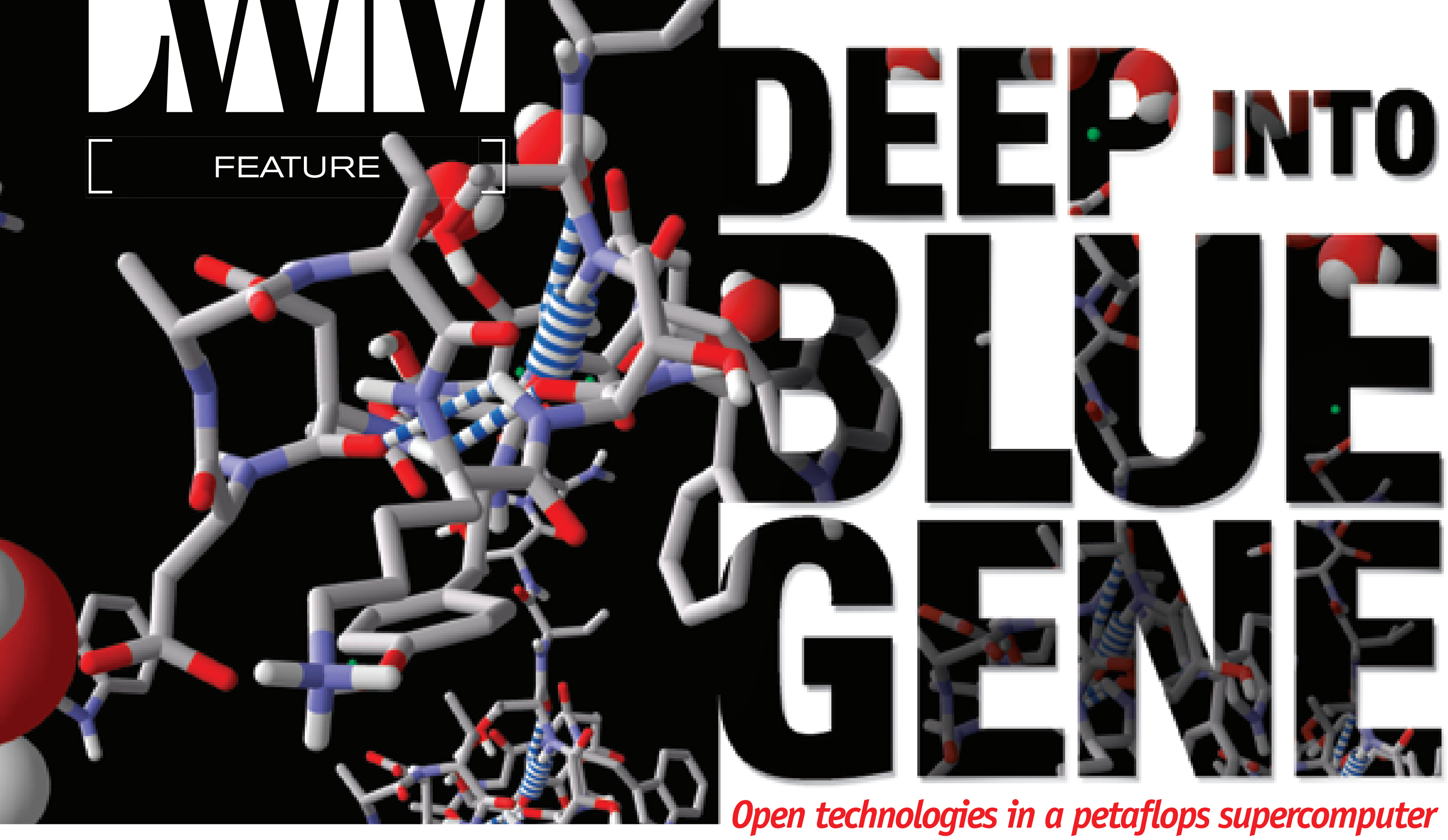
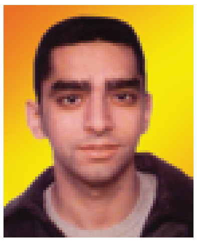


DEEP INTO BLUE GENE

Open technologies in a petaflops supercomputer



BY JOY KHORIATY

Blue Gene/L (BG/L) is a massively parallel supercomputer with low cost-to-performance ratios for speed, power, cooling, and floor space. It's designed and built by IBM in partnership with the Lawrence Livermore National Laboratory (LLNL) for the U.S. Department of Energy (DOE).

Using the latest hardware and software technologies, Blue Gene designers have set the grounds for a system that should smoothly scale to several petaflops. However, BG/L brings forth new challenges such as hardware failure management, efficient programming, and scalability of code.

From Quantum Chromodynamics to Protein Folding

The world of physics is a driving force in high performance computing. Applied physicists rely on supercomputers to simulate the evolution of dynamic systems that make up our universe. BG/L's story starts with a specific branch of physics called quantum chromodynamics (QCD), which deals with modeling the behavior of particles on subnuclear scales.

In 1998, a group of physicists working on QCD problems at the University of Columbia realized that clusters such as generic high-end servers interconnected by a fast switch were not efficient for QCD. So they designed and built a machine called QCDSP, which won them the Gordon Bell prize for the most effective supercomputer. QCDSP reached 600 GFlop/s (floating point operations per second) using 12,000 processors based on the Texas Instrument DSP C31 chip, with 2MB DRAM.

Their achievement led to the creation of a newer system named QCDOC (QCD On-a-Chip) in collaboration with IBM research in 2002. QCDOC is a 20 TFlop/s system based on 20,000 IBM processors, with 4MB EDRAM and external DDR/SDR SDRAM.

These massively parallel QCD machines consisted of thousands of low-end processors interconnected to form a multidimensional torus with nearest neighbor connections and global functions.

The successful work IBM research did with the University of Columbia on the QCD supercomputers influenced the design philosophy of the Blue Gene project, which IBM announced in 1999 as a multi-year plan to build a petaflop machine for calculations in the area of life sciences.

ABOUT THE AUTHOR

Joy Khoriaty is studying for his MSc in high performance computing at the Edinburgh Parallel Computing Centre (EPCC), at the University of Edinburgh, Scotland.
joy@elventails.com

The BG/L Hardware

BG/L's hardware consists of 65,536 compute nodes, 1,024 I/O nodes, and five different networks. The densely packed system grows from two nodes per compute card to 16 compute cards per node board to 16 node boards per 512-node midplane, and to two midplanes in a 1024 rack (see Figure 1).

Each node or ASIC (Application-Specific Integrated Circuit) is based on the system-on-a-chip (SoC) technology, which integrates processors, memory, and communications logic onto the same chip.

The ASIC includes two 32-bit PowerPC 440 processing cores, each with two 64-bit FPUs (Floating-Point Units) (see Figure 2). Each core has a private 32KB instruction and 32KB data L1 cache, a 2KB L2 cache, and a shared 4MB L3 EDRAM cache.

The PowerPC 440 embedded microprocessor is not L1 cache coherent and therefore does not offer symmetric multiprocessing (SMP) support, but the L2 caches are coherent and act as a prefetch buffer for the L3 cache.

At a target speed of 700MHz, the peak performance of a node is 2.8 GFlop/s using one processing core, and 5.6 GFlop/s when both cores and FPUs in a chip are used. This gives the entire system a target peak performance of 180/360 TFLOPS.

The nodes consume low power by using these simple embedded microprocessors that have low clock cycles, and also require less power, cooling, and space.

Compute Nodes

The 65,536 compute nodes strictly handle computations: the image of the program to be run is loaded by a lightweight kernel into the node's memory, and full resource use is allocated to that single computational run.

In a normal operation, one processing core per node is used for computation while the other handles messaging; however, both cores can be used for computation if there is no need for a dedicated communication processor in the application.

The compute node operating system is a simple, single-user, and lightweight kernel. It provides a single, static, virtual address space to one running process and a user-level runtime library that provides access to the networks.

Because of its single-process nature, the kernel does not implement context switching nor demand paging, but it provides large pages and SMP support. This approach results in the application process using the system resources as fully as possible.

According to Dr. Manish Gupta, senior manager of Emerging System Software at IBM Research, "the Compute Node Kernel provides a subset of Linux services, but preserves the Linux interface for the services that it does support. By design, the kernel does not support services (that many HPC applications can live without) related to dynamic process creation/management and demand paging. There are several services, such as file I/O and sockets, which the Compute Node Kernel supports by function shipping those requests to the I/O node running Linux. This organization helps achieve scalability to tens of thousands of nodes, while preserving much of the familiar Linux functionality."

I/O Nodes

Each of the 1,024 I/O nodes manages communications for a group of 64 compute nodes.

They provide access to the filesystem for the application running on the compute nodes, as well as socket connections to processes in other systems.

When a compute process on a compute node performs an I/O operation such as a read/write to a file, that operation is forwarded over the network to the managing I/O node. That I/O node issues the operation on the filesystem and returns the result to the compute node.

The I/O node's hardware is the same ASIC as for the compute nodes, with added external memory and a gigabit Ethernet connection.

These nodes run an embedded Linux operating system and support the execution of multiple processes. They also perform process authentication and authorization, job accounting, and debugging.

Link Nodes

Signals that cross the midplane boundaries go through the link nodes, which provide two functions:

- They boost the signal, improving its shape and amplitude.
- They can also redirect the signal to different ports, effectively partitioning the BG/L system into logically separate systems. Spare midplanes can then be set up for backup or to run separate jobs independently.

Network

BG/L has five communication networks (see Figure 3):

1. A 64x32x32 three-dimensional torus for nearest-neighbor calculations on grids. This is where point-to-point message passing takes place between compute nodes (compute nodes only/serial communication).
2. A global tree network for broadcast and reduction operations (all nodes/serial communication).
3. A barrier network for synchronization (compute nodes only).
4. Gigabit Ethernet to the JTAG (Joint Test Action Group) network for machine monitoring and control (all nodes).
5. Gigabit Ethernet for a connection to other systems such as external front-end nodes where jobs can be submitted from (I/O nodes only).

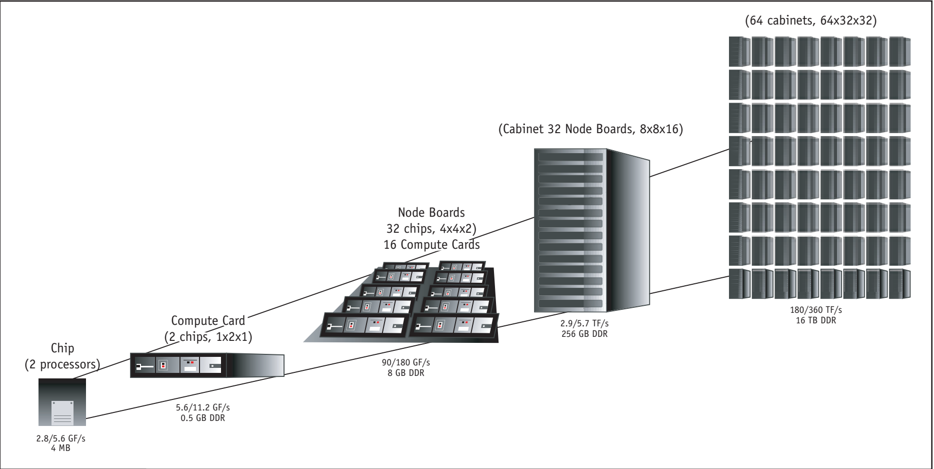


FIGURE 1 BLUE GENE/L SYSTEM PACKAGING

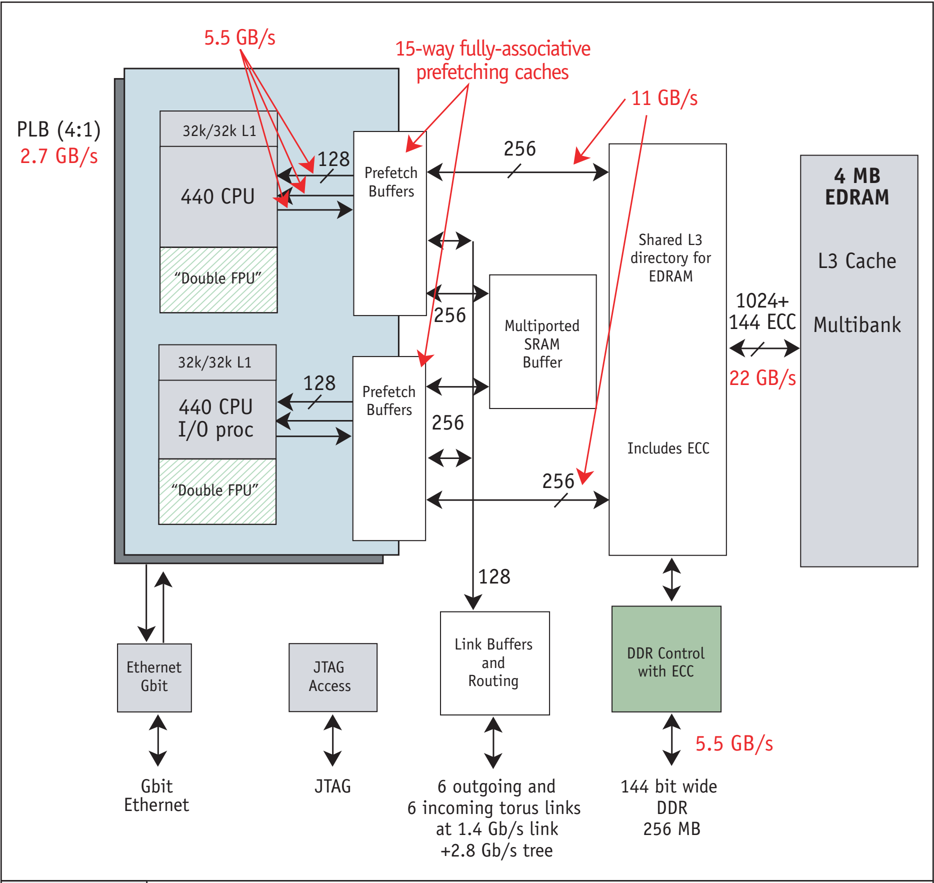


FIGURE 2 BLUE GENE/L NODE DIAGRAM

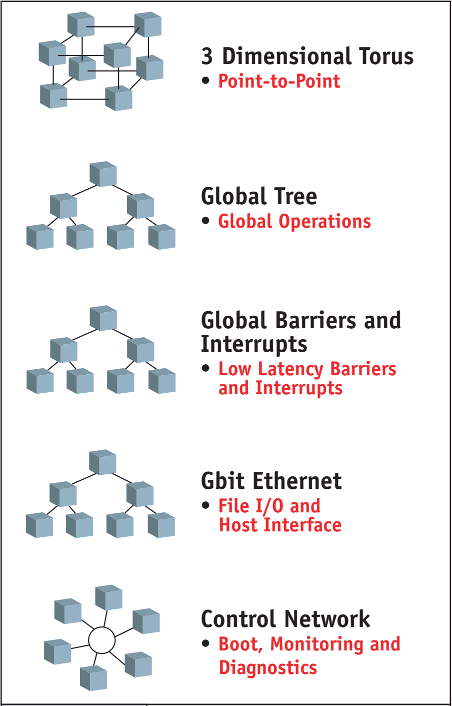

3 Dimensional Torus
• Point-to-Point
Global Tree
• Global Operations
Global Barriers and Interrupts
• Low Latency Barriers and Interrupts
Gbit Ethernet
• File I/O and Host Interface
Control Network
• Boot, Monitoring and Diagnostics

FIGURE 3 BLUE GENE/L NETWORKS

Programming BG/L

There are several approaches to programming BG/L. They include the familiar parallel programming model of message passing using the MPI library in C, C++, or FORTRAN programs, as well as more recent global address space programming models such as Co-Array FORTRAN (CAF) and Unified Parallel C (UPC).

Also, mathematical libraries are being updated to take advantage of the BG/L architecture.

The Message Passing Interface (MPI)

The main parallel programming model for BG/L is message passing using MPI. The message passing model consists of compute nodes sharing data by sending messages to each other over the network and is widely used in today's supercomputers.

Message passing on BG/L is supported through an implementation of the MPICH2 message-passing library that efficiently maps onto the torus and tree networks.

MPI is a familiar programming model for most researchers who write their programs in C, C++, or FORTRAN and provides a straightforward method for migrating existing application code to BG/L.

Co-Array FORTRAN (CAF) and Unified Parallel C (UPC)

CAF and UPC are explicitly parallel, global address space languages that incorporate the Single Program Multiple Data (SPMD) model into Fortran 90 and C, respectively.

For example, UPC adds the "forall" construct to the C language for distributing a for loop computation across nodes. This is in contrast to message passing where the programmer must specify what data is sent to which node.

Mathematical Libraries

Mathematical libraries are also being updated to take advantage of the BG/L architecture, such as exploiting the second core for compute-intensive kernels. Some of them include:

- The Engineering Scientific Subroutine Library (ESSL), an IBM implementation of the Linear Algebra Package (LAPACK)
- The Mathematical Acceleration Subsystem (MASS), an optimized mathematics library for reciprocals and square roots, in single and double precision mode
- Fast Fourier Transforms (FFT) and 3D-FFT libraries, which are also being optimized for effective use of the BG/L FPUs

Higher-level application frameworks are also being developed, like the Blue Matter framework for biomolecular simulations that separates the complexities of parallel programming from the underlying science.

Running the Programs

BG/L runs one program at a time; the program image is loaded and run on the tens of thousands of compute nodes that are available.

There are two approaches to running programs on the compute nodes:

- ***The coprocessor model:*** Each compute node runs an instance of the program

(65,536 instances) with two threads on noncoherent shared memory and 256MB of memory.
- ***The virtual node model:*** Both cores of each compute node separately load and run single threaded images of the application program (131,072 instances) while sharing the node's memory (128MB).

What Scientific Applications Will BG/L Run?

BG/L is not a general purpose machine. Its design is optimized to solve grid-based problems that are challenging in terms of the number of nearest neighbor communications that take place, as well as the amount of computational power required.

A good number of these interesting and computationally challenging problems can be found in high-energy physics, molecular dynamics, and astrophysics.

Protein folding is one of these key problems in the area of life sciences. Work on BG/L will help us understand and hopefully cure diseases such as Alzheimer's, cystic fibrosis, and Mad Cow by looking at the functions that folded/misfolded proteins play in their development.

Also on top of the list are simulations of the aging U.S. nuclear stockpile, which would mainly reduce the impractical need of testing nuclear weapons and ensure the safety and reliability of the nuclear supply.

Applications that BG/L does not optimally deal with include problems where little or no communication takes place between the compute nodes, and where distributed computing over the grid and farming can be used instead.

An Open System

Open technologies play a major role in the BG/L supercomputer as it relies at the software level on key open source software. Also, as the system was being built, BG/L simulations were being regularly run on Linux machines.

Linux, GNU, and MPI

Supercomputers are moving away from closed source operating systems to Linux and open source software.

This is the case with BG/L where the I/O and external front-end nodes run Linux, and the compute nodes run a kernel that is inspired by Linux. The choice to use Linux comes mainly from the familiarity of the scientific community with the OS, its libraries, and its interfaces.

This eases the task of programmers getting used to a new system, and brings the rich and widespread open source model of software development to scientists and programmers of BG/L.

As indicated by Dr. Gupta, "We wanted to provide a familiar development environment to the users so that they could benefit from the enormous computational power of BG/L without having to learn a new way of working with supercomputers. Using an open source environment also makes it easier to reach out to the community and foster collaborations."

Other fundamental open source software components include the BSD-licensed implementation of MPI, MPICH2, which is the main model of programming BG/L.

Open Hardware

IBM has made the PowerPC architec-ture an open standard. It has allowed third-party manufacturers to build the chips and has made the design tools available.

According to Dr. Alan Gara, chief architect of Blue Gene, IBM Research, "The Blue Gene/L machine was built on the premise that the key to reaching the highest levels of performance is to use 'open' hardware technologies along with commodities in a highly integrated, low-power system."

Open hardware is proving beneficial in many ways, where it "allows customers to customize their hardware in a manner best suited to their needs such as was done for the Blue Gene/L supercomputer."

–continued on page 46

"Supercomputers are moving away from closed source operating systems to Linux and open source software"

–continued from page 37

Customers can "achieve custom solutions for complex problems at a fraction of the investment and cost."

Also, "By using the 'Open' Power processor cores, the design team is able to focus on other aspects of the design, which can deliver performance. For Blue Gene/L this focus was put on the network support, which is important for a supercomputer."

Simulating BG/L on Linux Clusters

BGLsim is a parallel application that runs on Linux clusters to simulate the BG/L system.

It executes the full BG/L hardware instruction set and accurately models all the system memory and communication devices.

The simulator was used to develop and test the system software, including the compute and I/O node kernels, device drivers, MPI, compilers, Math libraries, and benchmarking suites.

This allowed for the rapid development and testing of the BG/L software even before the hardware arrived, which greatly contributed to accelerating the BG/L development process.

Challenges Ahead

The BG/L project presents challenges to the designers and programmers of the system at several levels.

One of them is handling the failure rate of the hardware components: having tens of thousands of processors dictates that the mean time between failures (MTBF) is relatively high. Checkpointing is being used as a solution at the software level. It consists of saving the state of the system at different times during a program run, so that only part of the computation can be restarted in case of failure.

Writing parallel code is still not simple and straightforward enough for most scientists and programmers, but global address space languages might make that job easier.

Getting the code to easily map well onto the hardware and scale up efficiently to such a large number of processors may also present several difficulties.

Onwards to Petaflops Computing

So far the BG/L system has been successful in achieving its design goals in terms of low-power processors, high-density packaging, high scalability in architecture, and encouraging performance results. The complete system should be ready in early 2005.

Crucial challenges in performance, scalability, and reliability need to be met so that Blue Gene/P, the petaflops generation machine that scales on the BG/L architecture with newer processors and more memory, becomes a reality toward the end of 2006.

Meanwhile, open source technologies, with Linux at their forefront, have proved themselves vital in the step toward building petaflops supercomputers.

Blue Gene/L Specifications

Peak Computational Rate	360 TFLOPS (in symmetric mode) 180 TFLOPS (in communications co-processor mode)
Aggregate Memory	16 TB
Aggregate Global Disk	400 TB
Delivered I/O Bandwidth to Applications	40 GB/s
External Networking	1,024 x 1 Gb/s Ethernet
Number of Nodes(processors)	65,536 (131,072)
Memory per Node	256 MB / 512 MB
Microprocessor Technology	Dual PowerPC 440 (700MHz)
Power Required for Computer and Cooling	2 MW
Heat Generated	4,500,000 BTU/hour
Cables in the Machine	5,000
Aggregate Cable Length	12 miles (19.3 km)

References

- *QCDSP/QCDOC:* http://phys.columbia.edu/~cqft/
- *IBM Blue Gene:* www.research.ibm.com/bluegene/
- *The MPICH and MPICH2:* www-unix.mcs.anl.gov/mpi/mpich
- *CO-Array FORTRAN:* www.co-array.org
- *UPC Community:* http://upc.gwu.edu/
- *Open Power Architecture:* www.power.org

Thanks to Adam Emery, Manish Gupta, and Alan Gara from IBM Research for their feedback and support.

LINUXWORLD MAGAZINE WWW.LINUXWORLD.COM

Leading Systems in the Top 500 Supercomputers List

November 2004 (http://www.top500.org)

RANK	SYSTEM NAME	MANUFACTURER/MODEL/PROCESSOR/INTERCONNECT	NUMBER OF PROCESSORS	PERFORMANCE (TFLOP/s)
1	BlueGene/L beta-system	IBM, BlueGene/L DD2 beta-system, PowerPC440, 0.7 GHz	32768	70.72
2	Columbia	SGI Altix, Intel Itanium 2, 1.5 GHz, Infiniband	10160	51.87
3	Earth-Simulator	NEC, SX6, Vector Processor, 1.0 GHz, Non-blocking Crossbar Switch	5120	35.86
4	MareNostrum	IBM, eServer BladeCenter JS20, PowerPC970, 2.2 GHz, Myrinet	3564	20.53
5	Thunder	California Digital Corporation, Intel Itanium 2 Tiger4, 1.4 GHz, Quadrics	4096	19.94