RC23724 (W0509-093) September 27, 2005
Computer Science

IBM Research Report

Exploiting Workload Parallelism for Performance and Power Optimization

Valentina Salapura, Robert E. Walkup, Alan Gara
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Exploiting workload parallelism for performance and power optimization

ABSTRACT

To achieve application performance, future power-limited design constraints impose new rules on the scaling for applications. Where in the past, application performance was the domain of delivering peak performance and translating peak performance into actual application performance, the challenge for future supercomputing applications will depend on delivering the best performance for a given budget.

To determine the impact of application scaling parameters to deliver performance for a given power budget, we analyze three applications These applications include both strong and weak scaling applications, as well as a study of the effect of input sets on power/performance scaling characteristics of an application. Specifically, we study the NAMD, UMT2K and WRF codes using a Blue Gene/L system as the target platform. We find that even for strong scaling problems, Blue Gene/L systems can deliver superior performance scaling and deliver significant power/performance efficiency.

Application benchmark power/performance scaling for the voltage-invariant $energy \times delay^2$ power/performance metric demonstrates that choosing a power-efficient 700MHz embedded PowerPC processor core and relying on application parallelism was the right decision to build a powerful, and power/performance efficient system.

1. INTRODUCTION

To deliver increasing aggregate performance for supercomputing workloads, the main challenge for system designers today is addressing system power. This is the result of both the individual core power dissipation, and the resulting challenge to cool a rack, as well as the limits of pre-existing data centers housing these racks. Today, a system power is a concern.

Typical air-cooled data centers have a power and cooling capacity of about 400-1600 kW making solutions above this limit non-viable for most prospective system users. Thus for future power-constrained designs, the emphasis is on increasing aggregate performance while maintaining system power requirements. This challenge has been addressed by a number of recent designs, ranging from consumer applications embodied by Cell [3, 11], to supercomputing applications exemplified by the world's #1 supercomputer, the Blue Gene/L system installed at Lawrence Livermore National Laboratories [20, 22, 6].

The key decision in achieving the performance goals within the available design constraints is to optimize the system to exploit parallelism, not single node performance. This is of particular importance in view of results reported by [21], which formalizes the intuitive notion of limiting returns on investment in single node performance.

In this work, we analyze a collection of supercomputing applications ranging from life science workloads to weather modeling. We report power and performance measurements using the Blue Gene/L system for applications including: (1) NAMD, a molecular dynamics application [13, 19], (2) UMT2K, an unstructured mesh benchmark [4], and (3) the WRF numerical weather prediction system [23].

The remainder of the paper is structured as follows: sections 2 and 3 give an overview of the Blue Gene/L hardware and software systems on which these experiments were performed. Section 3 analyzes design constrains and describes power and performance efficiency metrics for modern systems. Section 4 analyzes power/performance application scaling for NAMD as an example of life science application and studies the impact of the problem size on energy efficiency. Section 5 discusses UMT2K and evaluates the impact of thread-level and data-level parallelism on workload performance characteristics. Section 6 discusses WRF as an example of weather modeling codes. We conclude in section 7.

2. SYSTEM OVERVIEW

To achieve high system performance, the Blue Gene/L design opts for parallelism instead of high frequency. Frequency and voltage scaling, and/or more aggressively pipelined microprocessors achieve the highest single-node performance, but the marginal cost of performance is extremely high, exceeding 2% increase in energy for 1% increase in performance [15]. Given that the heat dissipation capacity of an air-cooled rack is limited, the most energy-efficient approach to reach maximum performance is to increase parallelism using efficient low-frequency processors.

To ensure good scaling, the Blue Gene/L system provides five high performance networks. These have been optimized

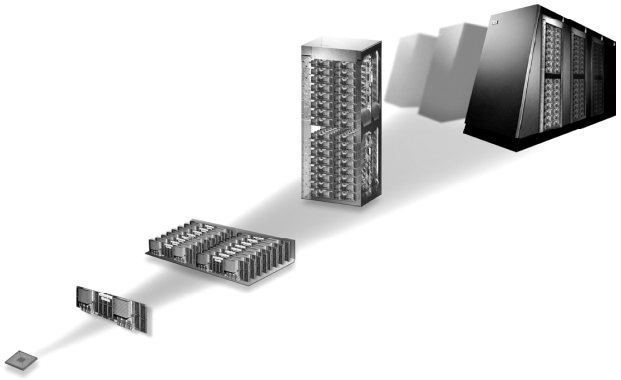

Figure 1: The Blue Gene/L concept leverages parallelism and advanced packaging to deliver superior power/performance.

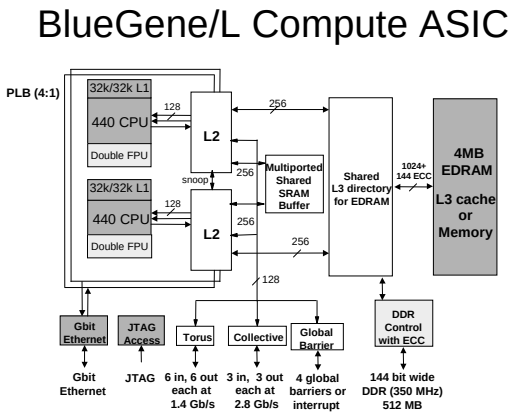

Figure 2: A single compute node ASIC integrates all node functions for the Blue Gene compute and I/O nodes.

and integrated into the system architecture. To reduce communication overhead, the network interfaces are located on the same chip as the processing units and implemented with an System-on-a-Chip design [6].

The Blue Gene/L computer is a scalable system consisting of up to 65,536 nodes using IBM CMOS CU-11 technology. Each node is built around a single System-on-Chip based on the PowerPC 440 processor core and 9 or 18 SDRAM-DDR memory chips. The Blue Gene/L compute node contains a prefetching L2 cache and 4MB high bandwidth embedded DRAM used as on-chip L3-cache shared between the two processors. This high density/low component count SoC-based design approach is important to reach an optimum cost/performance point.

Blue Gene/L also leverages the software infrastructure of the industry-standard PowerPC architecture. The aim was to differentiate where big gains were possible, and use standard components everywhere else. The standard components include industry standard networks from the Blue ASIC CoreConnect library, the PowerPC 440 processor core, embedded DRAM from the IBM 0.13μ CU-11 process technology, and IBM XL compiler technology.

The PowerPC 440 microprocessor is a high-performance, out-of-order industry-standard PowerPC processor originally targeted at high-end embedded systems. The processor supports 2-way super-scalar instruction execution with a seven stage pipelined micro-architecture. The processor core includes highly associative first level instruction and data caches with a capacity of 32KB each. Optimized components include the "Double Hummer" SIMD floating point unit, based on a standard PowerPC floating point unit but replicating key functionality, collective and torus networks, and an optimized memory system with software-managed coherence between cores.

System packaging is an integrated aspect of the Blue Gene/L design. In this design, a single rack consists of two midplanes. A midplane is populated with 512 processing nodes, with 5.6GFlop per compute node. In addition to compute

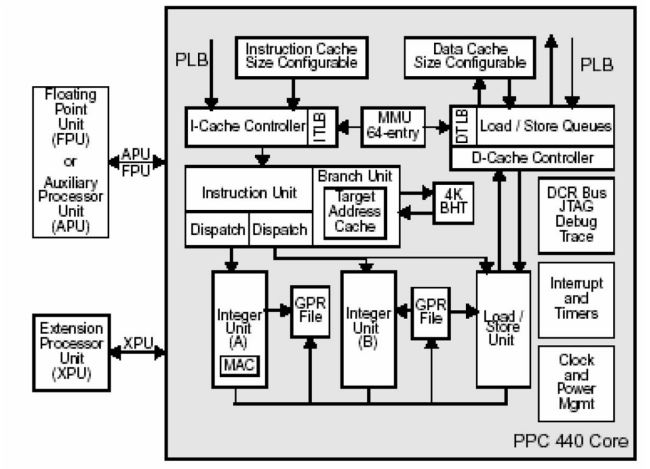

Figure 3: The PowerPC440.

nodes, a midplane is also populated with several I/O nodes. These I/O nodes are implemented using the same ASIC SoC as the compute node, but configured to handle file I/O and host communication.

Blue Gene/L addresses the communication requirements to achieve good application performance scaling by providing 5 dedicated communication networks: the torus network, collective network, barrier network, Ethernet, and IEEE1149.1 (JTAG). The networks are described in more detail in [7].

An important element of the Blue Gene/L system is support for multiple concurrent users. This "multi-user" mode is accomplished through logical partitioning (LPAR) of the machine, which allows each user to have a dedicated set of nodes for their application, including dedicated network resources. This partitioning is accomplished through the use of the link chips.

All of Blue Gene/L's networks pass through link chips when they cross midplane boundaries. The link chip is used

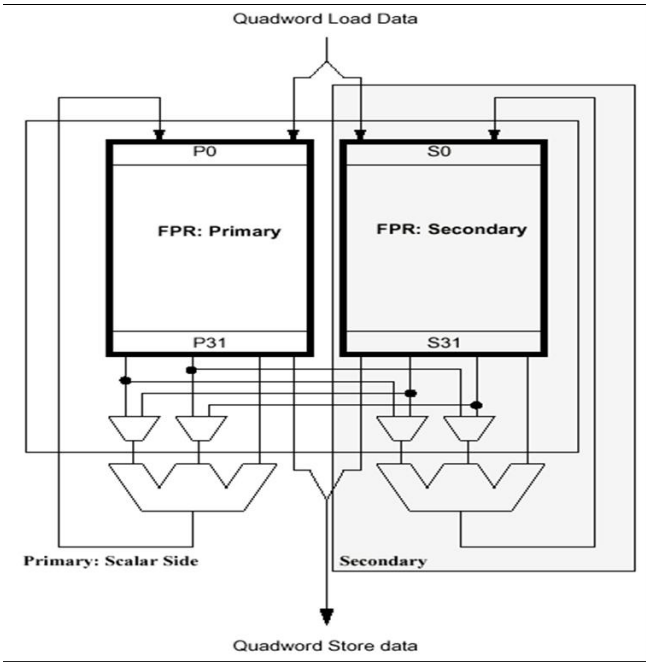

Figure 4: The Double Hummer dual-floating point SIMD architecture extends the traditional PowerPC architecture to deliver 4 floating point operations per cycle with a single instruction.

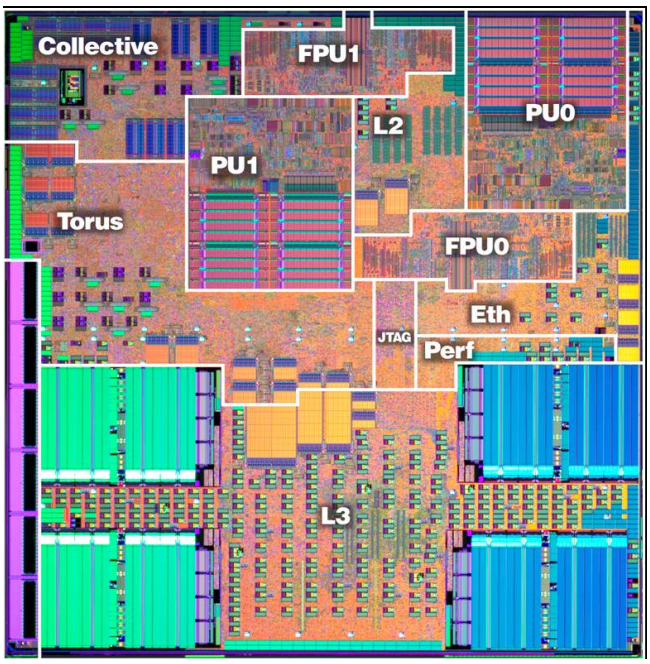

Figure 5: The Blue Gene/L compute node chip integrates two PowerPC 440 processors with a SIMD FP2 unit, dense EDRAM L3 on-chip cache and I/O capabilities to drive several high-performance communication networks.

to re-drive signals, to preserve the high speed signal characteristics over the cabling across midplanes. The link chip can also redirect signals between its different ports. This redirection capability enables partitioning of a single Blue Gene/L system into multiple, logically separate systems.

The Blue Gene/L compute node contains two processors based on the PowerPC 440 core with a SIMD floating point unit, achieving a peak performance of 2.8 GFlops per core. A SIMD approach was advantageous because it allows parallel execution of both floating-point and load/store instructions, thus reducing the need for power-intensive fetching and issuing of instructions. Figure 4 shows the design of the FP2 core. The Double Hummer uses two copies of the architecturally defined PowerPC floating-point register file. Both register files (primary and secondary) are independently addressable; in addition, they can be jointly accessed by SIMD instructions.

The primary register file is used in the execution of the pre-existing PowerPC floating-point instructions as well as the SIMD instructions, while the secondary register file is reserved for use by the new instructions. Along with the two register files, there are also primary and secondary pairs of datapaths, each consisting of a computational datapath and a load/store datapath.

The final aspect of low power design in Blue Gene/L was the System-on-a-Chip approach. By leveraging SoC integration to reduce component count, many high-power off-chip I/O signals driven across the signal pins and PCBs are eliminated. As described previously in [6], the Blue Gene ASICs were built with an optimized ASIC design flow incorporating guided placement and bitstacking, but no custom circuit work.

3. BLUE GENE SYSTEM SOFTWARE

As can be expected from a system the scale of Blue Gene/L, the software stack poses a set of interesting challenges. The basic programming model for Blue Gene/L is the MPI message passing interface for communication between nodes. Two configurations are possible: (1) "communication coprocessor mode" where one processor is dedicated to computation and the second can offload some of the communication work, and (2) "virtual node mode" where two independent MPI processes are placed on each node, so that both processors can be used for computation, while memory is partitioned between the two processes.

Each compute node has a micro kernel that can handle all functions necessary for high performance real time execution. The kernel provides an interface to the hardware for interrupts, timers, and error handling which are executed with supervisor privilege. To allow for fast communication and synchronization during execution of an application, access to the memory-mapped torus network interface is provided to the user address space. Thus, MPI messages are passed to other nodes without incurring the cost of a context switch from user to supervisor mode.

The Blue Gene/L computer node kernel does not implement a paging system to support virtual memory, reflecting the large number of nodes and threads provided in the system. Given the fact that nodes do not have private disk or other secondary storage devices, paging would be required over the I/O networks which would be prohibitive in a system of this size.

Instead, all threads use the same address space, mapping

PowerPC effective addresses (virtual addresses) directly to real (physical) addresses. The TLB is statically allocated at system startup and implements a flat 256MB effective to real address translation. Similarly, software threads map directly to hardware threads.

To allow multiple users to use the Blue Gene/L system concurrently, partitioning of the system is implemented by reprogramming the link chips. Within each Blue Gene/L partition, the operating system supports single user running single program application. This approach protects machine resources – such as memory or communication channels – from accidental corruption so they can be used reliably for error detection, debugging, and performance monitoring [1].

Ensuring reliable operation is another system function. In a large system of such as Blue Gene/L with up to 65536 nodes operating on computations for extended time periods, node failures are to be expected. Even with low MTTF rates, the compounding effect of large system image and long run times will make node failures a reality to be dealt with. Instead of expensive hardware recovery mechanism, reliability is a system function achieved by the system software layer through the implementation of a checkpointing system.

External access to a Blue Gene/L system occurs via the I/O nodes which provide an offload engine for I/O and interface traffic. I/O nodes do not participate in the the torus network, and in the MPI communication protocol. Instead, they run a standard Linux operating system kernel with appropriate service extensions to communicate with the compute nodes.

A host computer is required for compiling, diagnostics, and result analysis The host computer is also responsible for file system input/output and program loading, which is accomplished via message passing. The choice of host will depend on the class of applications and their bandwidth and performance requirements.

Since the processor at the core of the Blue Gene/L system is the industry standard PowerPC architecture, the familiar compiler and tool infrastructure available for the PowerPC family can be used to program the Blue Gene/L system. The XL compiler family has also been extended to support generating code which exploits the high performance dual floating point SIMD unit available in each core.

System bringup and testing is performed with the BGL ADE (Blue Gene/L Advanced Diagnostic Environment), an operating system which was designed expressly with the purpose to exploit and access all of Blue Gene/L's capabilities [8]. The BGL ADE system can deconfigure portions of a chip so as not to trigger hardware components which are suspected of being defective, and allow isolated testing of all system components.

4. POWER/PERFORMANCE EFFICIENCY

Power/performance efficiency was a prime design constraint to arrive at a high computing density system that would fit in the form factor of air-cooled racks in a standard machine room. Here, we analyze the power/performance efficiency of the final Blue Gene/L system and compare design choices in the design of systems, and how they influence power efficiency. While peak numbers are an eye-catching metric, delivered power/performance on applications is the relevant metric. Thus, our analysis is based on a detailed analysis of workloads to understand how well the Blue Gene/L system delivered on its promise of power/performance efficiency.

Power is a critical parameter as the densities that we are aiming for are more than a factor of 10 beyond where we could go with nodes based on traditional uni-processors. In addition, there are serious cost and reliability issues associated with high power density designs.

Several metrics have been proposed for characterizing energy efficiency. The most common of these metrics is MIPS / Watt. This metric corresponds to energy per operation, i.e., it 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, energy per operation metric does not reflect the benefit of reduced execution time with the increasing number of nodes. Thus, Gonzalez and Horowitz argue for the use of energy-delay product as a metric, which corresponds to paying 1% energy for an increase of 1% in performance [10, 9]. Introducing parallelism reduces execution time, without *ideally* increasing the power consumption per node, thus keeping energy consumption for a problem constant with dropping execution time.

Martin *et al.* propose the $energy \times delay^2$ product as an efficiency metric for VLSI computation [15, 16, 18]. This metric puts even more emphasis on performance thus favoring speedup via parallelism at a constant energy budget. This metric is considered to be superior to other metrics such as energy or energy-delay because it is reflects a "better" design point regardless of voltage – i.e., this metric remains constant as a system is voltage scaled to higher or lower performance. This metric is useful in considering tradeoffs between higher-frequency, higher voltage design points, and more power efficient lower frequency lower power cores.

In contrast, under the energy-delay metric, design optimality changes under the assumption of scaling to a different voltage. From another perspective, when voltage scaling is an option, the highest cost which can be justified is 2% energy for 1% performance – if the cost becomes higher than this, voltage scaling is always more profitable.[1]

Based on the observation that for many large-scale scientific problems, multi-processor scaling gives much better return on hardware resources than scaling a single processor, it is advantageous to address such problem classes with a system-level approach. Large scale scientific problems typically offer multi-processor efficiency by exploiting thread-level parallelism in the 60+% percent range, far exceeding the improvements to be achieved by a microprocessor-centric optimization approach.

Figure 6 shows the normalized energy, energy-delay and $energy \times delay^2$ metrics for idealized scaling of performance and power on a system offering thread-level parallelism. The scaling of this idealized metric is usually closely tracked by the reported LINPACK benchmark results.[2] While LIN-

[1]The energy, energy-delay and $energy \times delay^2$ metrics are closely related to the MIPS^n / W ratings, where $energy$ = 1 / (MIPS/W), $energy \times delay$ = 1 / (MIPS^2/W) and $energy \times delay^2$ = 1 / (MIPS^3 / W).

[2]Based on the Top 500 submissions for Blue Gene/L, LIN-

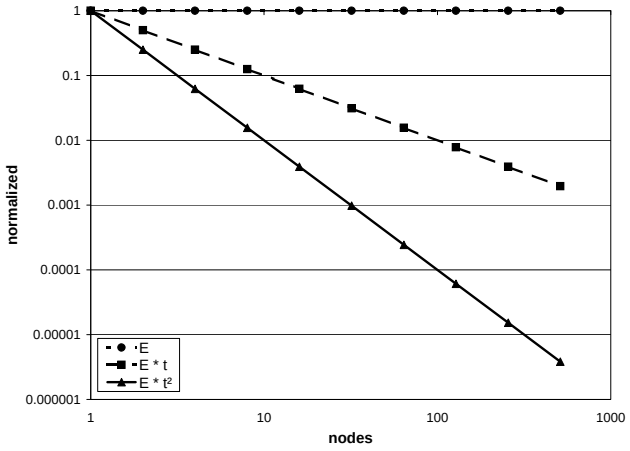

Figure 6: Ideal power/performance scaling across a range of Blue Gene/L partition sizes using logarithmic scale.

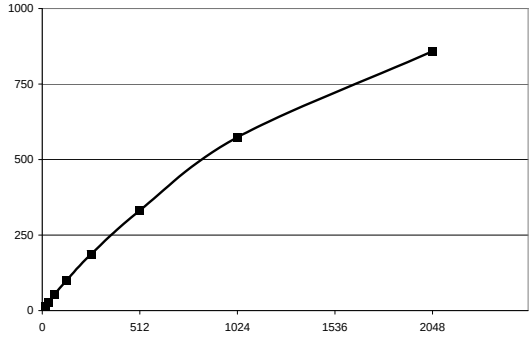

Figure 7: Normalized NAMD performance scaling across a range of Blue Gene/L partition sizes.

PACK efficiency may seem overly optimistic for actual applications, we will show below that applications also exhibit significant scaling efficiency.

In this and the following charts, each curve has been self-normalized to allow all three metrics to be represented in a single figure. In keeping with the interpretation of these numbers, a smaller energy-delay product is better, representing either less energy at the same performance, or more performance at the same energy, or both.

In the idealized benchmark, the energy per operation (curve labeled E) remains constant across all configuration, as performance per processor and power per processor remain unaffected by the increasing number of nodes. Introducing parallelism reduces execution time, without *ideally* increasing the power consumption per node, thus keeping energy consumption for a problem constant with dropping execution time. This is reflected by the improvement of of the peak performance energy-delay curve (labeled $\mathrm{E} \times \mathrm{t}$).

The third metric $\mathrm{E} \times t^2$ puts more emphasis on performance than the E and $\mathrm{E} \times \mathrm{t}$ metrics, thus favoring speedup via parallelism at a constant energy budget even more. The $energy \times delay^2$ curve (labeled $E \times t^2$) reflects a constant energy-delay metric under the assumption of voltage scaling – i.e., this metric remains constant as a system is voltage scaled to higher or lower performance. This metric is useful in considering tradeoffs between higher-frequency, higher voltage design points, and more power efficient lower frequency lower power cores. Thus, according to this metric a 100 node system offers a nominal four orders of magnitude better power/performance efficiency than voltage scaling a single core. Evidently, voltage scaling cannot cover such a range, but relative figures on the curve offer insights into tradeoffs in system design. For example, the peak performance of a 128 node system could also be obtained by a voltage scaled 100 node system at a loss of power/performance efficiency of $\mathrm{et2}_{128}/\mathrm{et2}_{100}$, where $\mathrm{et2}_i$ indicates the value of the $\mathrm{E} \times \mathrm{t}^2$ metric for a system with i nodes.

PACK shows about 80% efficiency (Rmax/Rpeak of DD2 hardware).

While we have discussed the use of these metrics for idealized performance scaling for thread-level parallelism, these observations will be most useful when applied to actual benchmark performance and power data to evaluate the power/performance efficiency of a massively parallel system such as Blue Gene/L.

While many large problems can be arbitrarily parallelized to allow the problem to match the size of the system on which computation is performed – such as LINPACK – many applications are fixed size problems requiring constant amount of computing independently of the size of the system. This is referred to as strong scaling. Strong scaling problems give a more conservative performance evaluation, as they characterize what can be gained from a parallel system on many real problems. Application performance results are also more realistic in that they include multiprocessor overhead, such as communication overhead, synchronization, program sections which cannot be parallelized, etc.

5. NAMD

As a representative of typical life-science applications which are an important application area for the Blue Gene/L project, we use NAMD, a molecular dynamic simulation system [13, 19]. The NAMD code is available as open source. To compare NAMD performance across a wide range of parallel systems, the NAMD distribution includes a benchmark problem which serves as the basis of the results reported here. The benchmark problem – referred to as ApoA1, for apoprotein A1 – models one high density lipoprotein particle (apoprotein A1) found in the bloodstream. The setup consists of the apoprotein A1 molecule solvated in water and has a fixed problem size of 92224 atoms of lipid, protein and water calculated in 500 steps.

In this work, we use NAMD version 2.6b1 [14] which is tuned to exploit the underlying high-performance hardware primitives of the Blue Gen/L systems.

Figure 7 shows the scaling of the NAMD molecular dynamic code on the Blue Gene/L system and plots the normalized performance against the number of nodes.

System performance scales well across a range of configu-

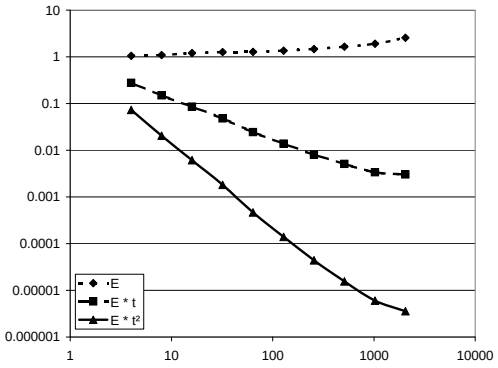

Figure 8: Normalized power / performance scaling across a range of Blue Gene/L partition sizes for the optimized NAMD version 2.6b1 and for the Apoa1 benchmark using logarithmic scale.

rations with an increasing number of nodes. Detailed analysis with a number of installed systems shows the extremely advantageous scaling behavior of the Blue Gene/L system. The increase in number of processors translates directly into increased system performance with only a small impact of multiprocessor overhead.

Energy and energy-delay metrics for NAMD is shown in Figure 8. The energy-delay metric shows a significant improvement based on the overall performance gain and shows three orders of magnitude improvement when scaling from a 1 node system to a 2048 node configuration.

The power/performance efficiency advantage is even more pronounced for the $energy \times delay^2$ metric showing more than five orders of magnitude efficiency gain of a 1024 node Blue Gene/L system over the base 1 node configuration.

Put another way, according to this metric, parallel NAMD execution on a 1024 node Blue Gene/L system is nearly five orders of magnitude more efficient than voltage and frequency scaling of the microprocessor in a 1 node configuration. While ranging across this performance differential is not possible with voltage scaling, the curve shows possible tradeoff points. The impact of choosing a higher performance microprocessor in a node can be determined from $E \times t^2$ curve.

Exploiting the metric invariance under voltage scaling, we can choose a configuration with n nodes and voltage scale it until it reaches the performance of a configuration with m nodes under the idealized assumption that all system components can be voltage scaled. While the curve does not tell us *how much* we would have to scale, we can determine the outcome in terms of loss in power/performance efficiency for compute-intensive problems with the ratio $et2_n/et2_m$, where $et2_i$ indicates the value of the $E \times t^2$ metric for a system with i nodes. Evidently, a Blue Gene/L system consists of computation, communication, memory, storage, and I/O components which will all have distinct power and performance characteristics in response to voltage scaling. Thus, programs which derive their performance and power characteristics from these other subsystems need to be analyzed in accordance with the scaling rules for those domains.

While voltage scaling may not allow to span the performance differential between any two configurations of n and m nodes, other tools are at the microarchitect's disposal. However, many of these techniques offer even worse % energy for % performance tradeoffs than voltage scaling.

A study on efficiency of pursuing a microprocessor-centric approach to achieve performance post-dates the Blue Gene effort – reported by Bose *et al.* [5, 12] - and confirms this decision. Bose *et al.* study the efficiency of microarchitecture changes and in particular increasing pipeline depth to achieve higher clock frequency. The results point to a very limited potential for power/performance efficiency improvement with modestly deep pipelines, and significant power/performance efficiency degradation beyond that point.

Blue Gene/L allocates partitions in sizes of 32 nodes or larger, making this the smallest useful configuration size in a production system. Configuration sizes below 32 nodes are reported only to study application scaling.

6. UMT2K

UMT2K is an ASCI Purple benchmark, which solves a photon transport problem on an unstructured mesh [4]. This application is written in Fortran-90 using MPI and optionally OpenMP. We use an MPI-only implementation, because there is no support for OpenMP on Blue Gene/L. (And no plans to support it, given the complexity created by the non-coherent L1 caches.) The unstructured mesh is statically partitioned using the Metis library [17]. The UMT2K test case used in this work was modified from the RFP2 benchmark problem, following the benchmark instructions to keep the amount of work per task approximately constant.

UMT2K 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 will report weak scaling results. [3] Figure 9 shows performance scaling of UMT2K across a range of Blue Gene/L partition sizes. Perfect scaling would correspond to constant steps per second executed on a node, or a completely flat line in Figure 9.

Figure 10 shows energy and energy delay products for this application. The dots on the figure 10 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 4098 compute nodes. Each compute node consist of two processors operating in the co-processor mode, where one processor handles communication tasks and the another one calculations.

While concentrating on microprocessor performance is not the central optimization point, microprocessor performance should not be neglected. A variety of processor design choices can be made which allow to generate higher-performing code. Examples of such optimizations are making available more registers to hide memory latency, and to exploit data parallelism when available.

To this effect, the Blue Gene/L system implements the

[3]Evidently, UMT2K can also be used as a *strong scaling* application by fixing the problem size. We posit that the observed lower performance for lower node counts will translate for longer runtimes for large problem sizes in a strong scaling scenario.

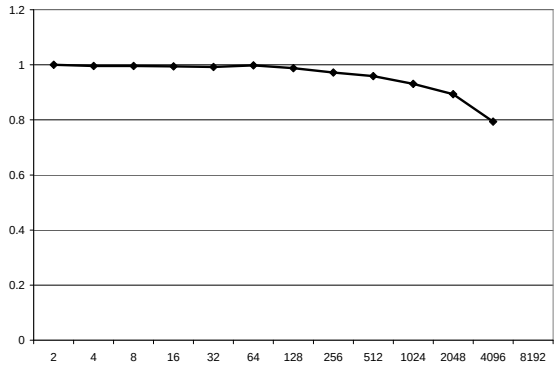

Figure 9: Normalized UMT2K performance scaling across a range of Blue Gene/L partition sizes.

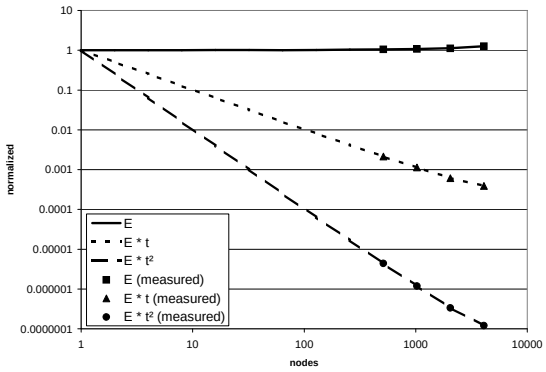

Figure 10: UMT2K application power/performance scaling across a range of Blue Gene/L partition sizes using logarithmic scale.

"Double Hummer" dual floating point unit, a SIMD architecture offering four parallel double precision operations per issued instructions (each SIMD instruction can issue a dual merged multiply-add operation). By tuning code to achieve better blocking factors by exploiting the ability to store 64 double precision values in architected floating point registers, and exploiting parallelism, efficiency can be improved with only a modest cost in power and area. Again, parallelism (in the form of data parallelism) offers high leverage for power performance optimization (as can be seen from the small area dedicated to the FP units in the floorplan).

UMT2K is a good example for the exploitation of both thread-level and data-level parallelism in Blue Gene/L. In UMT2K, there can be a significant spread in the amount of computational work per task, and this load imbalance affects the scalability of the application. Initial profiling [2] demonstrated that the elapsed time for UMT2K was dominated by a single computational routine consisting of a sequence of dependent division operations.

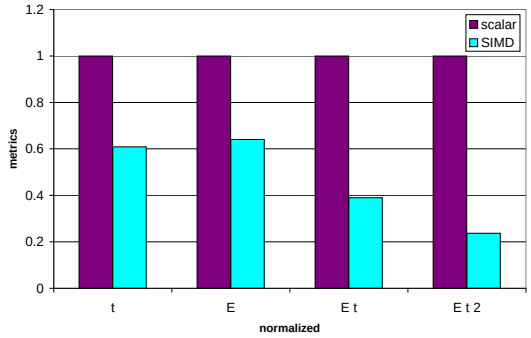

Figure 11: Normalized metrics for scalar and SIMDizied UMT2K runs for the 1024 Blue Gene/L partition size.

By splitting loops into independent vectorizable units, the IBM XL compiler was able to generate efficient data parallel code to exploit the data parallelism of the double FPU for computing reciprocals, resulting in a 50-60% overall performance boost from the double-FPU for this application.

Exploiting data parallelism is advantageous for improving energy efficiency as it offers multiple benefits which improve both power and performance product terms:

- marginal increase in CPU power (<10%) reduces runtime by half.
- reducing runtime reduces energy dissipation (which is power per time unit multiplied by time), leading to significantly better energy efficiency as the computation time is sped up.
- system components other than the CPU are exercised for a shorter period

This is illustrated in Figure 11. Figure 11 shows execution time, energy, energy-delay and energy delay squared power/performance metrics for UMT2K runs on a 1024 partition size on Blue Gene/L. The numbers are normalized to scalar run. Exploiting data parallelism by efficiently employing SIMD architecture reduces execution time to 60%. Although power consumption for exercising double-FPU increased for approximately 6%, the overall consumed energy is only 62% of the energy consumed for the scalar run due to significantly reduced execution time. Power/performance efficiency is even more obvious using energy-delay and energy-delay squared metrics for two approaches, showing 61% and 76% improvement.

Thus exploiting data parallelism is a particularly advantageous optimization, as it not only improves time, but actually energy consumption as well, whereas power/performance optimization involves a tradeoff between improving performance or energy consumption. Exploiting data parallelism leads to nearly an order of magnitude energy efficiency improvement for the $energy \times delay^2$ product.

7. WRF

WRF stands for Weather Research and Forecasting, and is a framework for numerical weather prediction [23]. It is used by a number of sites including the National Weather Service, and is under active development in a collaboration involving several centers including NCAR (National Center for Atmospheric Research) and the NOAA (National Oceanographic and Atmospheric Administration) Labs. The WRF forecast applications are specifically designed for parallel execution based on the MPI programming model, but WRF can also be used on single-processor systems as well as shared-memory machines.

For its calculations, WRF uses domain decomposition. In a typical situation, a three-dimensional finite-difference grid is partitioned in latitude and longitude, with each MPI process owning a block that includes all vertical layers in the block. The communication mainly involves boundary exchange with the MPI processes that "own" neighboring blocks using point-to-point communication routines, but there are also important collective operations, such as gathering the results to save "history" files.

The ratio of communication to computation is determined by the number of grid points in the physical domain and how the grid is partitioned among MPI processes. Roughly speaking, computation is related to the volume of the local domain, and communication is related to the boundary area of the local domain. In addition to boundary exchange, there are also gather operations that tend to be more expensive as the number of processors increases.

The performance data shown in Figure 12 is generated with the WRF benchmark problem, which uses a 425x300x34 grid. The benchmark problem is constructed to report performance of the forecast application without doing I/O. Perfect scaling would correspond to linear increase of total GFlops. The parallel efficiency on Blue Gene/L is a little larger than 50% at 2048 processors, i.e. you get about a 1000x speedup with 2000 processors.

Blue Gene/L is an interesting architecture to consider for WRF. With the typical numerical grids that are used today, weather forecast applications can scale fairly well out to several thousand processors. On Blue Gene/L it would be necessary to implement a parallel I/O strategy to keep the I/O and computation balanced.

Figure 13 analyzes energy and energy-delay metrics for WRF for a range of Blue Gene/L partition sizes, where all three curves are self-normalized. Based on the scaling behavior of WRF shown in figure 12, the overall energy consumption shows an increase as the problem scales to a bigger system. As WRF is strong scaling application, multiprocessor overhead due to non-parallel program sections, communication and synchronization overhead is visible in increase of energy required with the increase of partition sizes.

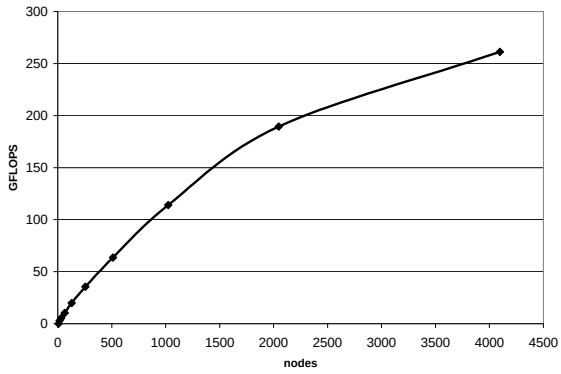

Figure 12: WRF performance scaling across a range of Blue Gene/L partition sizes.

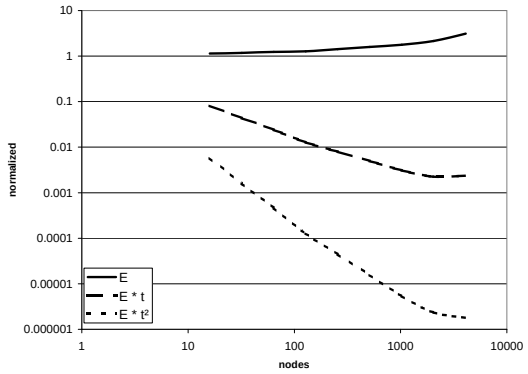

Figure 13: WRF application power/performance scaling across a range of Blue Gene/L partition sizes.

8. CONCLUSIONS

The Blue Gene/L system leverages parallelism to achieve high performance under power-constrained conditions. In Blue Gene/L systems, we exploit data- *and* thread-level parallelism with a massively parallel system using a data-parallel floating point unit as its compute engine.

We have given an overview of the Blue Gene/L architecture, and analyzed performance and power/performance characteristics of the Blue Gene/L system under a variety of conditions.

To derive actual application performance we have analyzed the scaling of several significant supercomputing applications – ranging from life science workloads to climate modeling – on a Blue Gene/L system. Specifically, our results are based on empirical studies for NAMD molecular dynamic code, the UMT2K unstructured mesh benchmark, and the WRF numerical weather prediction application.

We have also analyzed performance and power/performance characteristics using energy and energy-delay metrics. For the voltage-scaling invariant energy $\times$ delay^2 metric, we show that exploiting thread-level application scaling with lower power cores offers significantly better power/performance characteristics than using higher frequency cores with high power consumption. Additionally, our experiments show that exploiting data parallelism is advantageous for improving energy efficiency.

9. ACKNOWLEDGMENTS

This work has benefited from the cooperation of many individuals in IBM Research (Yorktown Heights, NY), IBM Engineering & Technology Services (Rochester, MN), and IBM Microelectronics (Burlington, VT and Raleigh, NC).

The authors would like to thank Michael Gschwind for many insightful discussions about power/performance optimization and the limitations of uniprocessor scaling and his encouragement which helped us formulate these ideas.

The Blue Gene/L project has been supported and partially funded by the Lawrence Livermore National Laboratories on behalf of the United States Department of Energy, under Lawrence Livermore National Laboratories Subcontract No. B517552.

NAMD was developed by the Theoretical and Computational Biophysics Group in the Beckman Institute for Advanced Science and Technology at the University of Illinois at Urbana-Champaign.

10. REFERENCES

[1] G. Almasi, C. Cascaval, J. Castanos, D. Lieber, and J. Moreira. Developing system software for BlueGene. Technical report, IBM TJ Watson Research Center, 2001.

[2] G. Almasi, S. Chatterjee, A. Gara, J. Gunnels, M. Gupta, A. Henning, J. Moreira, B. Walkup, A. Curioni, C. Archer, L. Bachega, B. Chan, B. Curtis, S. Brunett, G. Chukapalli, R. Harkness, and W. Pfeiffer. Unlocking the performance of the BlueGene/L supercomputer. In *Proceedings of SC 2004*, Pittsburgh, PA, November 2004.

[3] E. Altman, P. Capek, M. Gschwind, P. Hofstee, J. Kahle, R. Nair, S. Sathaye, J. Wellman, M. Suzuoki, and T. Yamazaki. Symmetric multi-processing system with attached processing units being able to access a shared memory without being structurally configured with an address translation mechanism. US patent US6779049, 2004.

[4] ASCI purple benchmark page. http://www.llnl.gov/asci/purple/benchmarks/-limited/code_list.html.

[5] P. Bose, D. Brooks, P. Emma, M. Gschwind, V. Srinivasan, P. Strenski, and V. Zyuban. Integrated analysis of power and performance for pipelined microprocessors. IBM Research Report RC22913, IBM TJ Watson Research Center, Yorktown Heights, NY, April 2003.

[6] A. Bright, M. Ellavsky, A. Gara, R. Haring, G. Kopcsay, R. Lembach, J. Marcella, M. Ohmacht, and V. Salapura. Creating the BlueGene/L supercomputer from low power SoC ASICs. In *Internationcal Solid State Circuits Conference*. IEEE, February 2005.

[7] A. Gara et al. An overview of the BlueGene/L system architecture. *IBM Journal of Research and Development*, 49(2), 2005.

[8] M. Giampapa, R. Bellofatto, M. Blumrich, D. Chen, A. Gara, P. Heidelberger, D. Hoenicke, G. Kopcsay, B. Nathanson, B. Steinmacher-Burow, M. Ohmacht, V. Salapura, and P. Vranas. BlueGene/L advanced diagnostics environment. *IBM Journal of Research and Development*, 49(2), 2005.

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

[10] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid State Circuits*, 31(9):1277–1284, September 1996.

[11] M. Gschwind, P. Hofstee, B. Flachs, M. Hopkins, Y. Watanabe, and T. Yamazaki. A novel SIMD architecture for the CELL heterogeneous chip-multiprocessor. In *Hot Chips 17*, Palo Alto, CA, August 2005.

[12] M. Gschwind, V. Zyuban, P. Bose, V. Srinivasan, P. Strenski, and P. Emma. The impact of power dissipation limits on microarchitectural choices. In *2003 Austin Conference on Energy-Efficient Design*, IBM Austin Research Laboratory, Austin, TX, February 2003.

[13] L. Kalé et al. NAMD2: Greater scalability for parallel molecular dynamics. *Journal of Computational Physics*, 151:283–312, 1999.

[14] S. Kumar. NAMD version 2.6b1 optimized for Blue Gene/L. personal communication, August 2005.

[15] A. Martin. Towards an energy complexity of computation. *Information Processing Letters*, 77:181–187, 2001.

[16] A. Martin, M. Nyström, and P. Pénzes. ET^2: a metric for time and energy efficiency of computation. In R. Melhem and R. Graybill, editors, *Power-Aware Computing*. Kluwer Academic Publishers, 2001.

[17] Metis home page. http://www-users.cs.umn.edu/ karypis/metis/index.html.

[18] P. Pénzes and A. Martin. Energy-delay efficiency of VLSI computations. In *Proc. of the 12th Grand Lakes Symposium on VLSI*, pages 104–107, April 2002.

[19] J. C. Phillips, G. Zheng, S. Kumar, and L. V. Kalé. NAMD: Biomolecular simulation on thousands of processors. In *Proceedings of SC 2002*, Baltimore, Maryland, September 2002.

[20] V. Salapura et al. Power and performance optimization at the system level. In *Proceedings of Computing Frontiers 2005*, Ischia, Italy, May 2005.

[21] V. Srinivasan, D. Brooks, M. Gschwind, P. Bose, V. Zyuban, P. Strenski, and P. Emma. Optimizing pipelines for power and performance. In ACM/IEEE, editor, *Proceedings of the 35th Annual International Symposium on Microarchitecture*, pages 333–344, Istanbul, Turkey, November 2002.

[22] The BlueGene Team. An overview of the BlueGene/L supercomputer. In *Proceedings of SC 2002*, Baltimore, Maryland, November 2002.

[23] The weather research and forecasting model page. http://www.wrf-model.org.