Entering the Petaflop Era:
The Architecture and Performance of Roadrunner

Kevin J. Barker, Kei Davis, Adolfy Hoisie, Darren J. Kerbyson, Mike Lang, Scott Pakin, Jose C. Sancho

Performance and Architecture Lab (PAL)
Los Alamos National Laboratory
Los Alamos, USA
{kjbarker,kei,hoisie,djk,mlang,pakin,jcsancho}@lanl.gov

Abstract—**Roadrunner is a 1.38 Pflop/s-peak (double precision) hybrid-architecture supercomputer developed by LANL and IBM. It contains 12,240 IBM PowerXCell 8i processors and 12,240 AMD Opteron cores in 3,060 compute nodes. Roadrunner is the first supercomputer to run Linpack at a sustained speed in excess of 1 Pflop/s. In this paper we present a detailed architectural description of Roadrunner and a detailed performance analysis of the system. A case study of optimizing the MPI-based application Sweep3D to exploit Roadrunner's hybrid architecture is also included. The performance of Sweep3D is compared to that of the code on a previous implementation of the Cell Broadband Engine architecture—the Cell BE—and on multi-core processors. Using validated performance models combined with Roadrunner-specific microbenchmarks we identify performance issues in the early pre-delivery system and infer how well the final Roadrunner configuration will perform once the system software stack has matured.**

Keywords—Petascale computing, heterogeneous, accelerators, performance analysis, Roadrunner.

I. INTRODUCTION

Breaking the barrier of a petaflop has been a grand challenge in high-performance computing for the last several years. There are several projects aimed to deliver petaflop or multi-petaflop performance in the near future, such as the machines being developed in the DARPA High Productivity Computing Systems program [1]. The quest for systems in the petaflop regime and beyond is driven by important challenges in science that could be addressed with large-scale predictive simulations.

In this work we analyze the architecture and performance of Roadrunner, a novel large-scale system currently being installed at Los Alamos National Laboratory (LANL). It is the first system to achieve a sustained performance in excess of 1 Pflop/s on the LINPACK benchmark, achieving 1.026 Pflops/s in May 2008. Roadrunner is a heterogeneous system containing an equal number of conventional, general-purpose microprocessors and special-purpose accelerators. The system has a peak speed of 1.38 Pflops/s (double precision; 2.91 Pflop/s single precision). Each node in the system, a *triblade*, consists of three blades. One blade contains two dual-core AMD Opteron processors, and the other two blades each contain two PowerXCell 8i processors (previously referred to as the Cell extended Double-Precision, Cell-eDP), the latest implementation of the IBM Cell Broadband Engine architecture. A total of 3,060 triblades are interconnected using InfiniBand to form the complete Roadrunner system.

The combination of flexible microprocessors with high-performing accelerator processors results in an extremely powerful system. Within a triblade each Opteron core is associated with a single PowerXCell 8i processor containing eight Synergistic Processing Elements (SPEs) and one Power Processor Element (PPE). Roadrunner exposes a rich computational environment given its heterogeneity. It can be utilized in one of three main processing paradigms depending on the suitability of each application. An application can run unmodified using only the Opteron processors without acceleration by the PowerXCell 8i processors. Or the application can use both processor types, accelerating key performance hotspots of the code on the PowerXCell 8i without porting all of the code. Or the application can run on the PowerXCell 8i processors for all computational tasks and employ the Opterons only as support for internode communication, I/O, and visualization.

To illustrate the architecture and performance of this new hybrid system, we describe the porting of the well-known MPI-based scientific application Sweep3D, a kernel application that implements the main processing involved in deterministic particle transport computations. We show how this application performs on pre-production Roadrunner hardware at full scale, focusing on measurements obtained on the new PowerXCell 8i processor, and use validated performance models to determine how well it will perform on Roadrunner when given a mature software stack. The performance, due in particular to the early software configuration, is likely to improve substantially before production use at LANL.

The rest of this paper is organized as follows. In Section II we describe the Roadrunner architecture including its compute node and interconnection network. In Section III we provide an overview of how Roadrunner can and is being utilized from an application point of view. In Section IV we analyze the low-level performance characteristics including the PowerXCell 8i processor and the performance of communications both within and between compute nodes. Section V details the implementation of the Sweep3D application on Roadrunner. The performance of Sweep3D is described in Section VI, and finally we draw some conclusions in Section VII.

II. THE ARCHITECTURE OF ROADRUNNER

The design of Roadrunner represents a careful balance between the availability of existing components and the need for technological advancement. The goals of the design were to provide high computational performance within acceptable cost and power budgets. While employing heterogeneous accelerators, Roadrunner needs to support three main processing paradigms: using unmodified code in a conventional cluster environment, accelerating key performance hotspots of the code, or running all of a code's computational elements on accelerators.

The recent introduction of the Cell Broadband Engine (Cell BE) [2,3,4,5], as designed for use in the Sony Playstation 3, offered a significant advance in computational power over general-purpose CPUs. A single Cell BE has a peak performance of 217.6 Gflops/s from its nine processor-cores. However, this speed is limited to single-precision (SP) operations and drops to 21.0 Gflops/s for double-precision (DP) required for scientific simulations. A further limitation is the use of a low-performance PowerPC processor core, which typically achieves a quarter of the performance of a typical AMD Opteron core. Finally, the memory controller supports only Rambus XDR memory, limiting memory capacity to 2GB.

To overcome these limitations, IBM implemented a new Cell processor, the PowerXCell 8i, for use in Roadrunner. The PowerXCell 8i has a peak performance of 108.8 Gflops/s on double-precision operations, and supports DDR2 memory at 800MHz, allowing up to 32GB memory. The remaining shortcoming, the low-performance PowerPC processor core, is overcome by the incorporation of dual-core AMD Opteron 2210 HE processors with one Opteron core for each PowerXCell 8i processor. In effect this provides an accelerator to each Opteron core.

Approximately 95% of the peak performance of Roadrunner results from the PowerXCell 8i processors. In addition, the high processing efficiency that is possible for many applications gives rise to high *achievable* performance. In May 2008, Roadrunner was the first system to achieve over 1 Pflop/s sustained performance on LINPACK (see www.top500.org).

The PowerXCell 8i processors have low power consumption, making Roadrunner one of the most energy efficient supercomputers. This is documented by its placement on the top "Green" supercomputers list in June 2008 at position 3 (see www.green500.org), achieving 437 Mflops/W on LINPACK. The two systems above Roadrunner on this list are small-scale PowerXCell 8i systems, achieving 488 Mflops/W, that do not incorporate the less power-efficient Opterons.

We detail the architectural details of Roadrunner, bottom-up, starting with a description of its compute node.

A. The Roadrunner Compute Node

A Roadrunner compute node is built using a unique triblade configuration. One blade, an IBM LS21, contains two dual-core Opteron processors, and the remaining two blades, IBM QS22s, each contain two PowerXCell 8i processors. An expansion card, taking the space of a fourth blade, serves to interconnect the three compute blades. Each Opteron core and PowerXCell 8i within the triblade has 4 GB of DDR2 memory. The Opteron processors are clocked at 1.8 GHz, with each core able to issue two DP (double precision) floating-point operations per cycle, resulting in a peak of 14.4 Gflop/s per LS21 blade. Each core has a 64 KB L1 data cache, a 64 KB L1 instruction cache, and a 2 MB L2 cache. The PowerXCell 8i processors are clocked at 3.2 GHz, and contain one Power Processing Element (PPE), and eight Synergistic Processing Elements (SPEs). The PPE has a traditional cache-based memory hierarchy consisting of a 32 KB L1 data cache, a 32KB L1 instruction cache, and a 512 KB L2 cache. It can issue two DP floating-point operations per cycle. Each SPE contains a SIMD processing unit that can issue a total of 4 DP floating-point or 8 SP floating-point operations per cycle. Thus the peak performance per PowerXCell 8i is 108.8 DP Gflops/s of which 102.4 Gflop/s are from the eight SPEs. A key characteristic of the SPE is that it can directly address only 256 KB of memory; this high-speed memory, known as *local store*, takes the place of a conventional cache architecture. Main memory, shared with the PPE, can be accessed only via explicit direct memory access (DMA) transfers to or from local store.
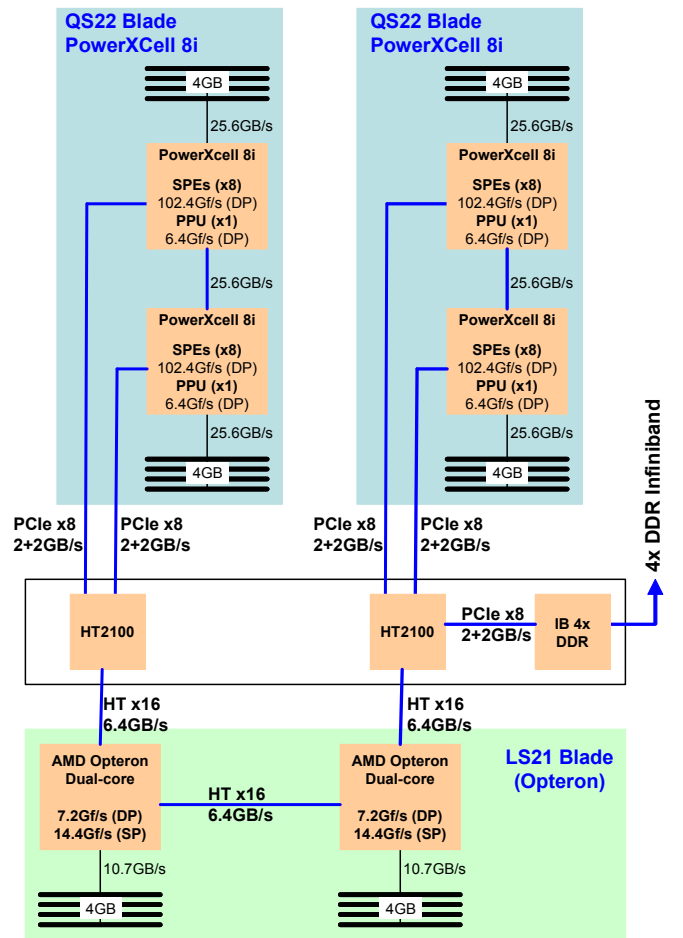

Figure 1. The structure of a Roadrunner compute node (triblade).

The PowerXCell 8i is a particular implementation of the IBM Cell Broadband Engine Architecture (CBEA). Relative to

the Cell BE that is utilized in the Sony PlayStation 3 game console, which has been extensively analyzed for scientific computation [6,7], the PowerXCell 8i has seven times the peak DP floating-point performance of the Cell BE. A performance comparison of these two processors is presented in Section IV.

Within the triblade, each of the PowerXCell 8i blades is connected to the Opteron blade via two PCIe x8 connections. The structure of this design is shown in Fig. 1. Each PowerXCell 8i blade has a direct connection to an Opteron processor. Indeed, as we describe later, each PowerXCell 8i processor is associated with a specific Opteron core such that each Opteron core communicates directly with one PowerXCell 8i processor in accelerated operation mode. The PCIe buses from the Cell blades are converted to HyperTransport for connection to the Opteron processors using two Broadcom HT2100 I/O controllers. The HT2100 has a single HyperTransport x16 port and three PCIe x8 ports. The third port on one of the HT2100 connects a Mellanox 4x DDR InfiniBand host channel adapter (HCA). The peak bandwidth between each PowerXCell 8i processor and its associated Opteron core is 2GB/s in each direction.

B. The Roadrunner Compute Unit

Each Roadrunner Compute Unit (CU) contains 180 compute nodes (triblades), 12 I/O nodes connected to a Panasas parallel file system (PFS), and an additional service node. Each CU contains a single Voltaire ISR 9288 4x DDR 288-port InfiniBand switch providing a peak bandwidth of 2GB/s per direction, per port. The switch contains a total of 36 24-port crossbars. The switch within each CU has the crossbars arranged in a two-level tree with 24 crossbars in a lower level, and 12 crossbars in the upper level. 22 of the lower level crossbars have 8 compute nodes attached, one crossbar has 4 compute nodes and 4 I/O nodes, and the last crossbar has 8 I/O nodes attached. In this way all nodes within a CU are interconnected in a full fat tree, utilizing 192 of the 288 available ports, yielding a stand-alone cluster with up to 96 up-links available to interconnect multiple CUs, as shown in the lower part of Fig. 2.

C. The Roadrunner System

The Roadrunner system consists of 17 CUs, for a total of 3,060 compute nodes. Eight Voltaire ISR 9288 switches are used to interconnect all 17 CUs in a 2:1 reduced fat tree, as shown in the upper part of Fig. 2. Note that each CU has 12 connections to each of the inter-CU switches.

The inter-CU switches are arranged as three levels of 12 crossbars. Each crossbar on the first level interconnects the first 12 CUs, and the last level interconnects the last 5 CUs, with the middle level allowing for communication between the two sets of CUs. The overall design allows for up to 24 CUs.

The interconnection topology reduces the average number of crossbar hops required between any nodes compared to a conventional fat-tree topology. A node is one hop away from the other seven on the same crossbar, three hops away from other nodes within the same CU, at most five hops away from any node on same side of the inter-CU switch, and at most seven hops away from nodes in CUs on the other side of the

inter-CU switch. Each switch-hop imposes approximately 220ns latency. A summary of the hop counts traversed for inter-node communications is given in Table I.

TABLE I.　SUMMARY OF THE DISTANCES BETWEEN NODE-0 (CU-1) AND ALL OTHER NODES (IN CROSSBAR HOPS) IN ROADRUNNER

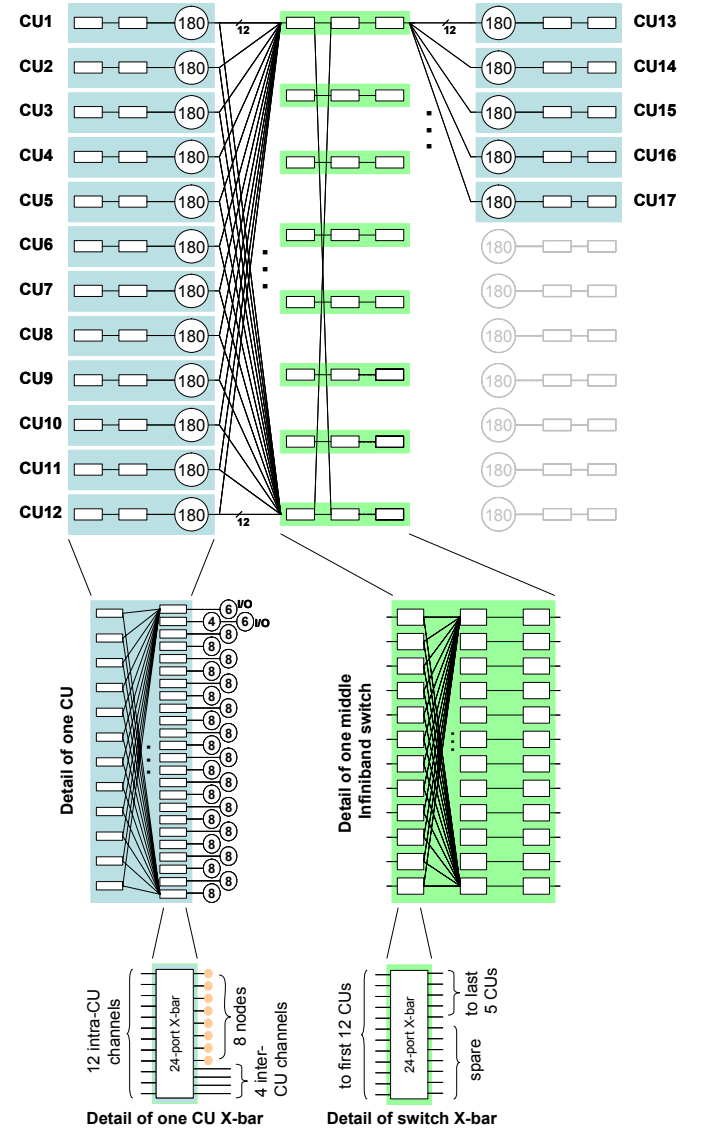| Destination node	No. of destinations	Hop count
Self	1	0
Within same crossbar	7	1
Within same CU	172	3
In CUs 2-12, same crossbar	88	3
In CUs 2-12, different crossbar	1892	5
In CUs 13-17, same crossbar	40	5
In CUs 13-17, different crossbar	860	7
Total	3060	5.38 (average)

Figure 2.　An overview of the roadrunner system showing the interconnection of 17CUs, and expanded views of: a single CU Infiniband switch; an inter-CU Infiniband switch; a single CU Infiniband crossbar with 8 nodes attached, and a single inter-CU crossbar.

The physical layout of Roadrunner consists of 16 compute racks per CU and an additional 4 racks for the inter-CU switches. Each rack holds either 12 triblades (4 triblades per BladeCenter chassis), two Voltaire switches, or a combination of both.

D. Roadrunner System Characteristics

A summary of the characteristics of Roadrunner is listed in Table II for its compute node, CU, and the overall system. The overall system peak performance is taken to be the sum of the peak performance of all Cell processors (PPE and SPE) and all Opteron processors contained within the compute nodes. The peak performance for both DP and SP floating-point operations is listed. A breakdown of the flops/s (DP) and the memory capacity of a single node are shown in Fig. 3. This clearly illustrates that the majority of the floating point performance derives from the SPEs on the PowerXCell 8i, and that the main memory capacity is equally split between the Opteron blade and the PowerXCell 8i blades. It is also interesting to note that the on-chip memory is similar between the four PowerXCell 8i processors and the four Opteron cores.

TABLE II.   PERFORMANCE CHARACTERISTICS OF ROADRUNNER

System		
CU count	17	
Node count	3,060	
Peak Performance (DP)	1.38 Pflops/s	
(SP)	2.91 Pflops/s	
Connected Unit (CU)		
Node count	180	
Peak performance / CU (DP)	80.9 Tflops	
(SP)	171.1 Tflops	
Compute Node (triblade)	**1x Opteron blade**	**2x Cell blades**
Processor count	2	4
Processor-core count	4	4 PPEs, 32 SPEs
Clock Speed	1.8 GHz	3.2 GHz
Peak-performance/node (DP)	14.4 Gflops/s	435.2 Gflops/s
(SP)	28.8 Gflops.s	921.6 Gflops/s
Memory per processor	4 GB	4 GB
	(667MHz DDR2)	(800MHz DDR2)

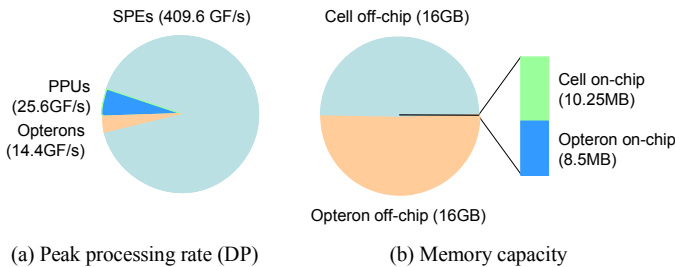(a) Peak processing rate (DP)   (b) Memory capacity

Figure 3.   Processing and memory capacities of a Roadrunner node

III.   ROADRUNNER MODES OF USE

Roadrunner was designed to support a spectrum of usage models. Existing codes can immediately take advantage of Roadrunner's conventional CPUs and network, ignoring the accelerators and treating the system as an ordinary cluster. (Without accelerators, Roadrunner would appear at approximately position 50 on the June 2008 Top 500 list, http://www.top500.org/list/2008/06/100.) Users can then identify performance-critical sections of code and modify those sections to run on the Cell blades. The SPaSM molecular-dynamics code is an example of an early Roadrunner application that followed that approach [8]. In some cases, such as the VPIC particle-in-cell code [9], all computation is performed on the Cell blades; the Opterons are used almost exclusively to relay messages across nodes on behalf of the Cells. Finally, IBM's Roadrunner version of the LINPACK benchmark [10] not only uses both the Opterons and the Cells for computation but uses both at the same time—in contrast to the simpler, RPC-style offload of performance-critical functions to the accelerators.

Because Roadrunner—and Roadrunner's form of hybrid computing—is still in its infancy, there are virtually no high-level languages or application programming interfaces (APIs) that support it. (IBM's ALF library [11] does support hybrid execution within a node but not across nodes.) Consequently, all of the initial applications that have targeted hybrid execution on Roadrunner utilize low-level communication mechanisms: MFC I/O [12] for intra-socket communication among SPEs (over the Element Interconnect Bus) and between the SPEs and the PPE, DaCS [13] for communication within a node between a PPE and an Opteron, and MPI [14] for internode communication.

An implication of Roadrunner's deep communication hierarchy—Element Interconnect Bus (EIB), PCI Express, HyperTransport, InfiniBand—is that the performance of a hybrid application is critically dependent upon the application's ability to exploit spatial and temporal locality. That is, a high-performance Roadrunner program should be able to do most of its work on the SPEs and directly from local store, occasionally transferring data between local store and the Cell blade's memory, even less frequently transferring data between Cell memory and Opteron memory, and only rarely transferring data over the network.

In porting codes to Roadrunner, two hybrid programming models have been explored. We call the first approach the *accelerator model*. In the accelerator model, the basic structure of the application is no different from that running on a conventional architecture: most of the work is performed locally, and communication is used primarily for boundary exchanges or other relatively infrequent operations. On Roadrunner, the local work is pushed down to the Cell, and the SPE programs run—for long stretches of time—out of Cell memory. We call the second Roadrunner programming approach the *SPE-centric model*. The SPE-centric model is essentially the inverse of the accelerator model: instead of each Opteron having a unique MPI rank and pushing compute-intensive work down to the SPEs, each SPE has a unique MPI rank and pushes non-compute-intensive work (including communication over the InfiniBand network) up to an Opteron. An advantage of the SPE-centric model is that it facilitates intra-Cell SPE-to-SPE communication over the high-bandwidth EIB links. A disadvantage is that achieving good performance still requires that attention be paid to intranode versus internode communication even though the model provides the illusion of a flat communication space. As in the accelerator model, the

bandwidth to Cell memory is the primary performance limiter. Section V presents a case study of a Roadrunner application built to the SPE-centric model.

IV. LOW-LEVEL PERFORMANCE MEASUREMENTS

In this section we look at the measured performance characteristics of Roadrunner. We examine two subsystems: memory and communication. Also we compare the performance of the PowerXCell 8i and the original Cell BE.

A. IBM PowerXCell 8i processor

The PowerXCell 8i processor is designed to improve some capabilities of the previous Cell processor (Cell BE), namely double-precision floating point performance and memory capacity in the blades. To increase the memory capacity, the memory controller in the Cell processor was modified to support DDR2 memory. This change enables the PowerXCell 8i to support up to 32GB of memory in a blade. In the previous Cell BE, only Rambus XDR memories were supported, limiting the memory capacity to 2GB per blade. In addition, with DDR2 800MHz memories, the performance is quite similar to that of the previous XDR memories, providing 25.6GB/s memory bandwidth to each Cell.

Due to the overarching goal of reducing complexity in the first Cell BE processor, double-precision operations were not fully pipelined, yielding a poor performance compared with single precision operations. Specifically, at a clock rate of 3.2 GHz the aggregate SPE peak performance on the Cell BE is 204.8 Gflops/s SP but only 14.6 Gflops/s DP. The redesigned floating point unit in the PowerXCell 8i processor achieves a peak performance of 102.4 Gflops/s DP.

To provide detailed performance analysis of the new double precision unit, we developed several microbenchmarks that measure three characteristics of all instruction types: (i) latency – from entering to exiting the instruction pipeline, (ii) local stall – the minimum number of cycles that must elapse between two issues to the same execution unit, and (iii) global stall – the number of cycles the processor stalls before any more instructions can be issued. The microbenchmarks are coded in assembly and thus were not subject to compiler optimizations.

The latencies (in cycles) for each instruction group are shown in Fig. 4. The only difference in performance between the Cell BE and the PowerXCell 8i is observed on the FPD (Floating-Point Double) instruction group. The latency of FPD instructions is decreased from 13 cycles on the Cell BE to 9 cycles on the PowerXCell 8i.

The repetition distance, the number of cycles between consecutive uses (i.e., the sum of local and global stalls for each instruction group) is shown in Fig. 5. Note that a value of one corresponds to execution units that are fully pipelined. The only execution unit not fully pipelined in the Cell BE was the FPD unit, which in the new PowerXCell 8i processor is fully pipelined. This modification gives the SPEs in the PowerXCell 8i the expected peak performance of 102.4 Gflops/s at 3.2GHz.
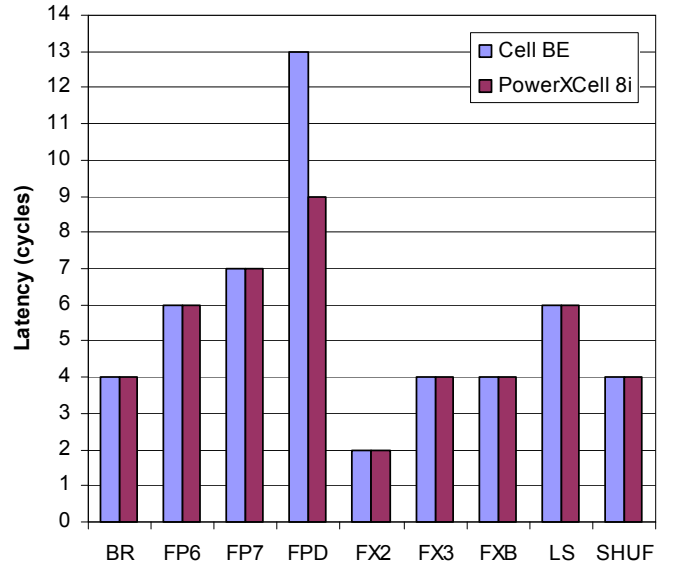

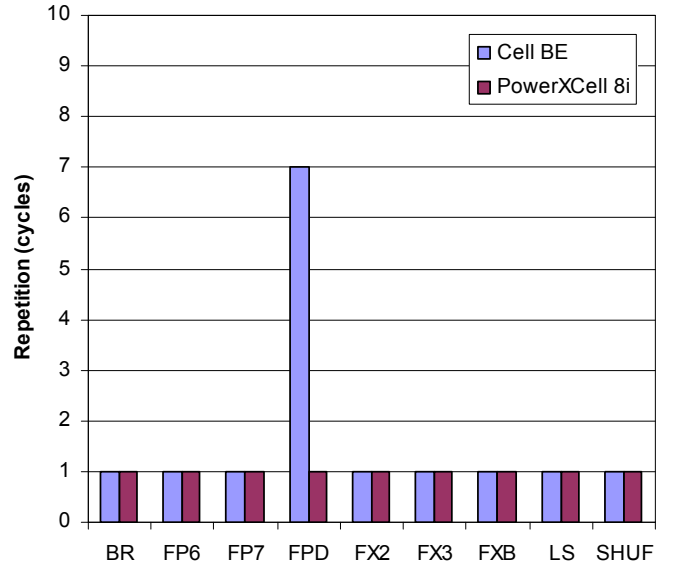
Figure 4.   Latency of each execution group

Figure 5.   Repetition distance of each execution group

The PowerXCell 8i's impact on several scientific applications of interest to LANL (VPIC, SPaSM, and Milagro) has recently been evaluated [15]. The PowerXCell 8i increases the performance of both SPaSM and Milagro by a factor of 1.5x. VPIC doesn't show significant improvements on this new processor as its calculations use single precision floating-point operations. As we show in Section VI a further application, Sweep3D, achieves a factor of almost 2x on the PowerXCell 8i.

B. Memory subsystem performance

Due to the complex memory hierarchy and heterogeneity of Roadrunner, exploiting locality is very important to achieving good performance. Knowledge of the memory performance at

each level gives insight into optimizations for this system. Streams [16] is used to quantify the performance of each of Roadrunner's three processors' memory systems: the SPE's local store and access to main memory, the PPE's access to main memory, and the Opterons conventional memory system. Each SPE dispatches one 128-bit load with a load latency of 6 cycles [3]; pipelined, this gives a maximum bandwidth of 51.2 GB/s. The maximum bandwidth for both the SPE and the PPE to main memory is 25.6 GB/s. As shown in Fig. 1, access is provided by the memory controller interface via the EIB. The eight SPEs and the PPE share access to the memory controller through the EIB which runs at 96 bytes/cycle. The Opteron has a maximum bandwidth of 10.7 GB/s per socket to main memory, again as shown in Fig. 1. Streams results are shown in Table III. The results reported are from the Streams TRIAD, and are for a single SPE or by socket on the Opteron and the PPE. For the SPE we report results from accessing the local store as this is the only directly addressable memory. The measured results show that the PPE is a bottleneck and is best used for control functions such as setting up memory regions for the SPEs to use. The latency to memory was measured using *memtime*, a microbenchmark designed to access one word per cache-line. Each word that is read is used to determine the address of the next word. By altering the total size of the data, the latency to each level of the memory hierarchy can be obtained. The measured latency to main memory is also listed in Table III.

TABLE III.   MEASURED MEMORY PERFORMANCE OF ROADRUNNER PROCESSORS

	Stream Triad (GB/s)	Latency (memtime) (ns)
Opteron	5.41	30.5
PowerXCell 8i (PPE)	0.89	23.4
PowerXCell 8i (SPE)	29.28	9.4

C. Communication subsystem performance

An important performance aspect of the triblade and overall system design is the achievable communication performance. Here we analyze the communication performance within the triblade (intranode) and of the entire system (internode). On a Roadrunner triblade, only the Opteron blade is connected to the InfiniBand network. Cell blades can only communicate with each other and with other triblades indirectly by transferring data over he PCIe bus to an Opteron and having the Opteron forward the data over InfiniBand (then back over PCIe to a target Cell). We use the DaCS communication library [17] for Cell-Opteron and Opteron-Cell communication and MPI (specifically, Open MPI [18]) for communication between Opterons in different triblades.

A set of three communication ping-pong tests were developed to determine the achievable latency and bandwidth of each component of a Cell-to-Cell data transfer. One test measures the DaCS/PCIe latency between a Cell and an Opteron, one test measures the MPI/InfiniBand latency between Opterons in different triblades, and one test measures the latency of a complete Cell-Opteron-Opteron-Cell message path. Computing the difference among these latencies shows the breakdown of the latency of a zero-byte message as it travels from a Cell to its corresponding Opteron, over the

network to another Opteron, and back down to a Cell. Fig. 6 shows that the major communication cost resides in the communication between the Cell and the Opteron; the current implementation of DaCS and the corresponding PCIe driver has a higher latency than that of Infiniband. The communication between the Opterons is comparatively fast, probably due to the relative maturity of the Open MPI and InfiniBand software stacks. Overall, the latency of a message between a Cell and another Cell located in a different node is 8.78μs as measured with the current software.
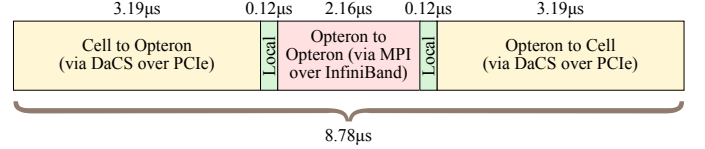

Figure 6.   Breakdown of the latency of a zero-byte message as it travels from a Cell to another Cell located in a different node.

The unidirectional and bidirectional performance for Cell-to-Cell communication is shown in Fig. 7 for both intranode and internode communication. The bidirectional bandwidth is taken to be the sum of bandwidths in both directions and is compared to twice the unidirectional bandwidth. All curves in Fig. 7 depict the worst-performing pair when all Cell-Opteron pairs are in use. In the intranode (PPE-Opteron) case, the bidirectional bandwidth is 64% of the double-unidirectional bandwidth (1,295 MB/s vs. 2,017 MB/s). In the internode (PPE-Opteron-Opteron-PPE) case, the bidirectional bandwidth is 70% of the double-unidirectional bandwidth (375 MB/s vs. 536 MB/s).

The unidirectional bandwidth between the Opteron cores in two nodes is shown in Fig. 8. The unidirectional bandwidth varies depending on the cores that are actually communicating, due to proximity of the Infiniband HCA being closer to one pair of Opteron cores as described in Section II. Significantly better bandwidth is obtained when cores 1 and 3 communicate (1,478 MB/s) than when cores 0 and 2 communicate (1,087 MB/s). Cores 1 and 3 (and their memory) are closer to the HCA than cores 0 and 2.
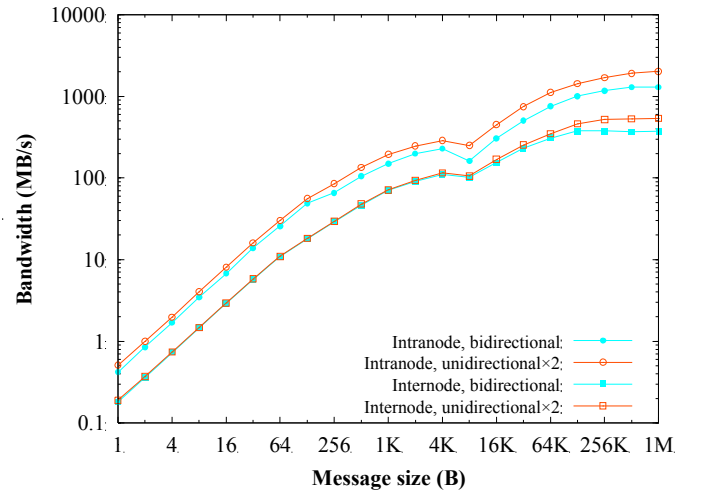

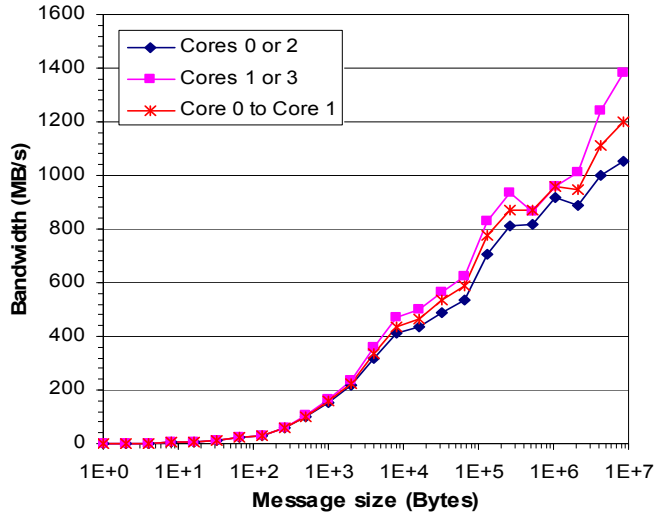Figure 7.   Intra- and internode bandwidth (PPE to PPE)

Figure 8.   Internode unidirectional bandwidth (Opteron to Opteron)

Figure 10. Zero-byte latency from MPI rank 0 to other nodes

A comparison between the current DaCS PCIe performance to that observed on the Infiniband is shown in Fig. 9. This is an interesting comparison because both communication transfers are over an 8x PCIe bus. In fact, the test is slightly biased in favor of DaCS because the MPI-over-InfiniBand ping-pong data represents network crossing in addition to *two* PCIe crossings—one on the sending node and one on the receiving node. Comparing the DaCS performance to the InfiniBand performance on the left-hand *y*-axis reveals that the InfiniBand achieves significantly more bandwidth than current DaCS. The ratio of the two curves is plotted using the right-hand *y*-axis. Although the ratio approaches 1 for large message sizes, at smaller messages in the range 0 to 20KB, DaCS achieves less than half the bandwidth of InfiniBand. This performance should improve as the DaCS software matures.
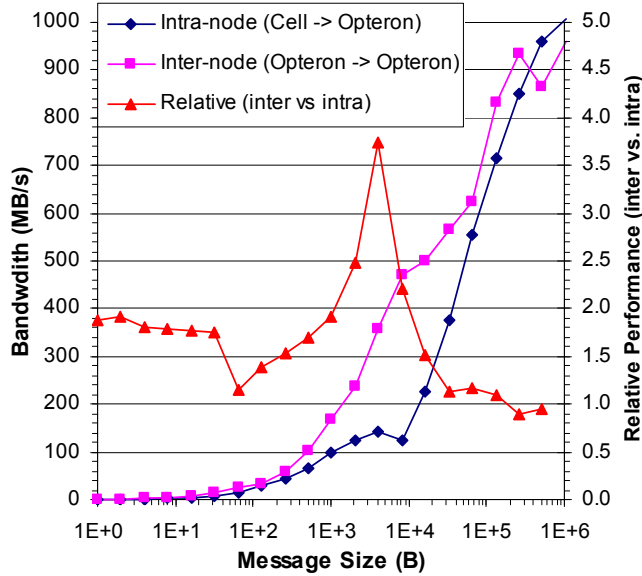

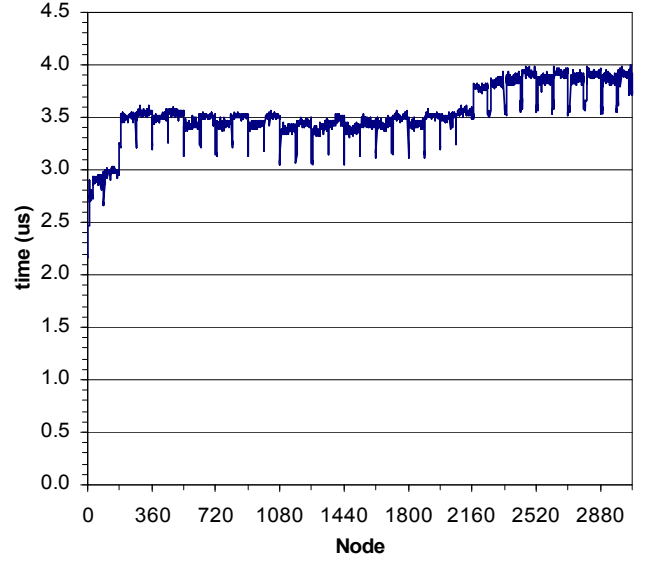Figure 9.   InfiniBand vs. DaCS PCIe performance

Fig. 10 shows the latency from MPI rank 0 to each of the other 3059 nodes for a zero-byte message. Rank 0 communicates to each of the other nodes in sequence with no network contention. The switch hierarchy as described in Section II-B is observed from the message latency. The first seven neighbor nodes are connected to the same crossbar as node zero and observe the minimum latency of 2.5µs. The performance plateau around 3µs reflects the latency to the other nodes in the same CU. The next network level increases the latency to approximately 3.5µs to the 12 CUs that are 5 hops away. Here we can observe periodic lower latencies, every 90 nodes, that are due to the unique wiring between CUs as described in Section II-B. The final plateau shows the latency of the extra switch traversals (7 hops) to reach the last 5 CUs, here we see latency of just under 4µs. A similar test with a 1MB message from MPI rank 0 to all other nodes reports an average bandwidth to the nodes of 980 MB/s under default OpenMPI parameters and 1.6GB/s when memory buffers are pinned.

In the next section we take these results into consideration in porting and optimizing the Sweep3D application to the Roadrunner architecture.

V.   APPLICATION CASE STUDY: SWEEP3D

The components of Roadrunner each exhibit high performance as observed by the analysis of the processing characteristics of the PowerXCell 8i and the intranode and internode communication characteristics. In the following two subsections we describe a particular application, Sweep3D, and its optimization for Roadrunner. Sweep3D is a challenging application on large-scale systems, in that it exhibits parallelism at different levels of granularity and because typically it does not achieve high single-core efficiency [19]. Its processing characteristics are representative of production applications of interest to LANL.

A. Overview of Sweep3D

Sweep3D solves a single-group time-independent discrete ordinates (S_N) neutron-transport problem. The input data is specified in a 3-D Cartesian geometry with dimensions *I*, *J*, and *K*. Sweep3D is commonly run in weak-scaling mode with each process computing on the same number of grid-points regardless of the number of processes used. The global data grid of size $(I \times n) \times (J \times m) \times K$ is decomposed in two dimensions across a logical 2-D processor array of size $n \times m$. The unit of work in Sweep3D is a block of the *K* dimension which is split into *K/MK* blocks, where *MK* is the blocking factor. At most one block is computed on a processor in any one time-step. Blocking is used to achieve high parallel efficiency rather than to maximize cache utilization [19].

The underlying algorithm in Sweep3D corresponds to a wavefront. The computation consists of a succession of wavefronts in which each processor performs a computation on the data it owns, updates its block's boundaries with data from its upstream neighbors, and sends updated boundaries to its downstream neighbors. The algorithm initiates wavefronts in each one of the corners of the eight octants of the 3-D problem space. A subgrid is computed for a number of angular values once its upstream boundary information is available, producing boundaries that are passed downstream. Communication using MPI is used to transfer boundary results. The progression of the sweep calculation in 1-D, 2-D, and 3-D are shown in Fig. 11 for four steps. The inflows to, and outflows from, a single element are also shown for a particular wavefront direction.
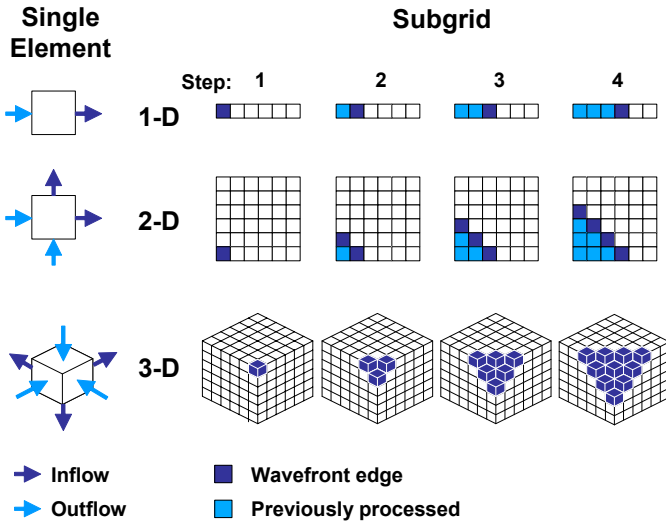

Figure 11. Wavefront propagation in Sweep3D.

B. Implementation of Sweep3D on the Cell

Sweep3D was one of the first scientific applications ported to the Cell architecture [20], wherein the investigators reported good performance using a master/worker paradigm. In this early implementation the unit of work was a single "pencil" of work in the *I* dimension. However, the approach required a significant number of DMAs to transfer data volumes repeatedly to SPE workers. Consequently, the performance was bounded by the available memory bandwidth, because the volume was large relative to the local store of the SPEs.

Rather than using a master/worker arrangement, our implementation retains the structure of the original MPI version of Sweep3D. That is, each SPE has its own MPI rank and processes a static allocation of an $I \times J \times K$ sub-grid. This SPE-centric approach allows balancing and overlapping of the computation of a block in the *K* dimension with the communication of the surfaces, thereby maximizing the ratio of computation to DMA activity. In other words, our version communicates surfaces, while the previous algorithm required communicating entire volumes. Furthermore, much of the communication in our implementation occurs on the high-bandwidth Element Interconnect Bus (EIB) that interconnects the SPEs. The PPEs in this programming model are used only to forward messages and to handle some of the mundane operations that are not available on the SPE, such as main-memory allocation. To implement Sweep3D in this way the Cell Messaging Layer was used (see Section V-C).

Other optimizations were implemented to take advantage of dual issue and the SIMD instructions on the SPEs. The number of angles within an octant was fixed at six and the inner loop nest was re-ordered so that the angle loop was innermost. This allowed the processing of two of the six angles at a time using SIMD instructions. This inner loop was then unrolled three times. In this way all six angles are efficiently computed in a single iteration. The SPE is a dual-issue processor but only if the correct instruction mix is available for the odd and even pipelines [21,22]. Taking this into account, instructions were interleaved for these two pipes by rearranging non-dependent code and unrolling and adding temporary variables so that more instructions were available to fill the two pipes. Also, the order of the instructions was carefully chosen to hide as much of their latencies as possible. Another issue to be addressed in programming the Cell is the small size of the local store. The size of the computational work block is $I \times J \times (K/MK)$ in Sweep3D and the blocking parameter, *MK*, must be carefully chosen so that the block fits into the local store. Given that the subgrid resides in the main memory, fetching and storing each block requires the SPEs to DMA to/from the main memory.

C. Sweep3D communication support

Communication in Sweep3D is handled by the Cell Messaging Layer[1] (CML) [23], a message-passing library for the Cell that implements a subset of MPI [14]. The key abstraction provided by CML is that the cluster appears to be a sea of interconnected SPEs. Each SPE in the entire cluster has a unique MPI rank, and any SPE can communicate with any other SPE regardless of whether the SPEs are in the same socket, the same blade, the same node, or different nodes. CML was in fact designed in concert with our Sweep3D implementation and supports all of the MPI functions needed by Sweep3D including point-to-point messaging, barriers, broadcasts, and data reductions. The advantage of CML's SPE-centric model is that MPI codes can be ported quickly to the Cell, then incrementally modified to stage data in and out of main memory and to exploit the SPE's vector units.

In CML, the PPE and Opteron are subservient to the SPE and are used primarily for shuttling messages to SPEs in other blades. However, CML does provide a convenient remote procedure call (RPC) mechanism that enables a SPE to invoke a function on the PPE, and the PPE to invoke a function on the Opteron, and receive the result. Our Sweep3D implementation uses RPC functions to invoke `malloc()` on the PPE in order to allocate main-memory buffers that can be used for holding intermediate results. Roadrunner does not expose the parallel filesystem to the PPEs, so our Sweep3D invokes an RPC function on the Opteron to read and return the input file.

The CML implementation is structured for high-speed communication. Messages sent between SPEs in the same socket, or cache-coherent sockets within a blade (as in a stock QS21 blade [24,25], not the case in Roadrunner), can proceed entirely over the high-speed Element Interconnect Bus (EIB) with no PPE involvement. Within a socket, CML peak performance has been measured as 0.272μs latency for a zero-byte message and 22.4GB/s for a large (128KB) message. Communication between SPEs in different sockets involves DMAs to the PPE, which transfers the data to a target PPE using MPI; the target SPE then DMAs the message from its PPE. On Roadrunner, since PPEs are not directly connected, each PPE forwards all SPE MPI requests over the PCIe bus to its corresponding Opteron using DaCS, and the Opteron performs the MPI operations on behalf of the PPE.

VI. PERFORMANCE OF SWEEP3D

The performance of Sweep3D was first measured on a single PowerXCell 8i processor. The input file specified a 5×5×400 sub-grid size per SPE in weak-scaling mode with a blocking factor of 20 K-planes (*MK*=20) with the number of angles fixed at 6. Fig. 12 shows the iteration time for a single SPE of the PowerXCell 8i and all 8 SPEs in a single processor. For comparison, the figure also shows the performance of the original version of Sweep3D running on dual-core and quad-core AMD Opterons as well as a quad-core Intel Tigerton. It can be seen that the implementation of Sweep3D on a single SPE of the PowerXCell 8i achieves a runtime comparable to a single core of the Intel and AMD processors. The performance of the full socket (8 SPEs) is twice that of the quad-core processors and almost 5 times that of a dual-core Opteron.

A performance comparison of the Sweep3D implementation described here to that previously reported by other researchers [20] is given in Table IV. The problem compared is the 50x50x50 subgrid, with *MK* blocking of 10, and the number of angles (*MMI*) set to be 6. The table shows that the performance is significantly higher for the implementation reported here and that the PowerXCell 8i's improved double-precision floating-point capability can also substantially improve the performance delivered to applications. For Sweep3D the improvement is a factor of 1.9x.

TABLE IV.   PERFORMANCE COMPARISON OF SWEEP3D
IMPLEMENTATIONS FOR THE CELL.

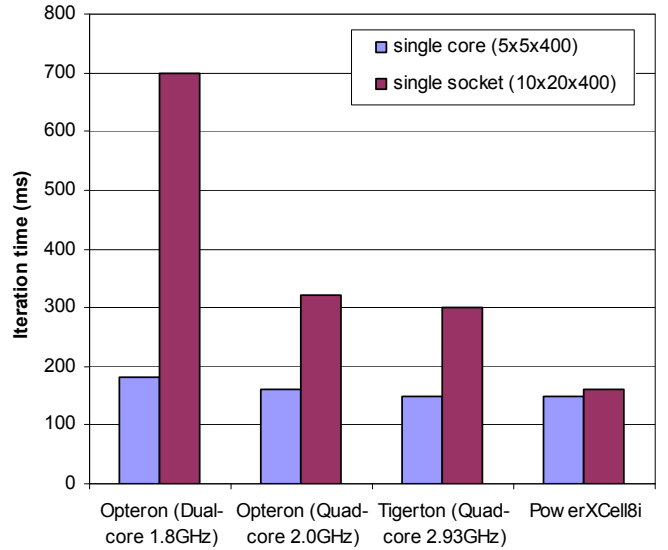| `	previous Sweep3D	our Sweep3D
CBE	1.3 s	0.37 s
PowerXCell 8i	N/A	0.19 s

Figure 12. Performance comparison of Sweep3D

A. Performance at Scale

In this section we examine the performance of Sweep3D at large-scale on the pre-production Roadrunner system. Using a subgrid size of 5x5x400 per SPE, the achieved performance from a non-accelerated code (Opteron only), and from the accelerated code that used the PowerXCell 8i processors is shown in Fig. 13. The time for one iteration is shown when using between 1 and 3,060 nodes of Roadrunner (the full 17CU system). It can be seen that the measured times on the PowerXCell 8i processors are substantially lower than that on the Opterons.

The use of DaCS as the underlying transport mechanism for CML introduces some overheads in terms of increased latency and decreased bandwidth across the PCIe bus between the PowerXCell 8i and the Opteron. The peak PCIe performance was measured using a small microbenchmark showing that the achievable peak bandwidth is 1.6GB/s (unidirectional) and with a minimum latency of 2μs. This is substantially better than that seen for the achieved DaCS latency as shown in Fig. 6, and achieved unidirectional bandwidth shown in Fig. 7.

Using a performance model of Sweep3D [19], which has been validated on most large-scale systems over the last decade, we have predicted what the best achievable performance should be. This is also shown in Fig. 13 utilizing the peak PCIe performance characteristics. The peak PCIe performance will not be realized in practice as overheads induced by flow control and multiple buffering when dealing with multiple communications will be required for transmission correctness. However, as shown in Fig. 13, the performance of the current implementation is close to the best achievable at small scale, and could be improved by almost a factor of two at large scale. We expect that some of this performance improvement will be realized before Roadrunner becomes a production machine in late 2008.
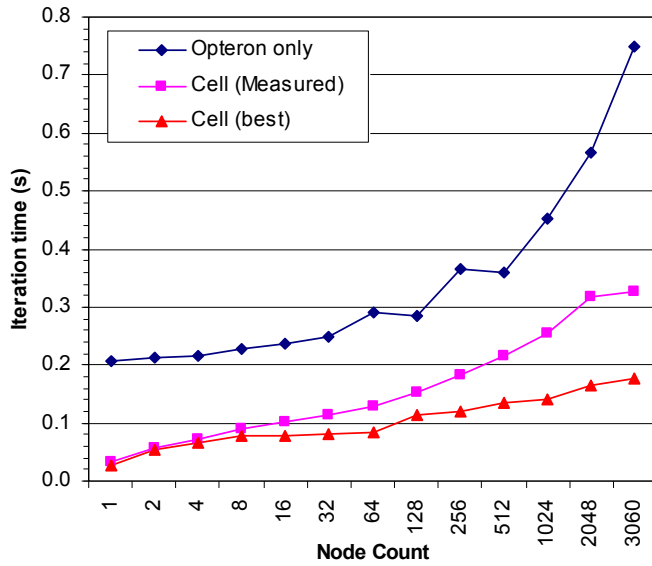
Figure 13. Performance of accelerated and non-accelerated Sweep3D on Roadrunner.

The relative performance between the accelerated, and non-accelerated version of Sweep3D on Roadrunner is shown in Fig. 14. Both the current measured improvement and that modeled using the peak PCIe performance is shown in Fig. 14. It can be seen that currently almost a factor of two higher performance is achieved when using the accelerators in Roadrunner. The performance improvement may be as high as 4x at large-scale if the peak PCIe performance were to be realized.
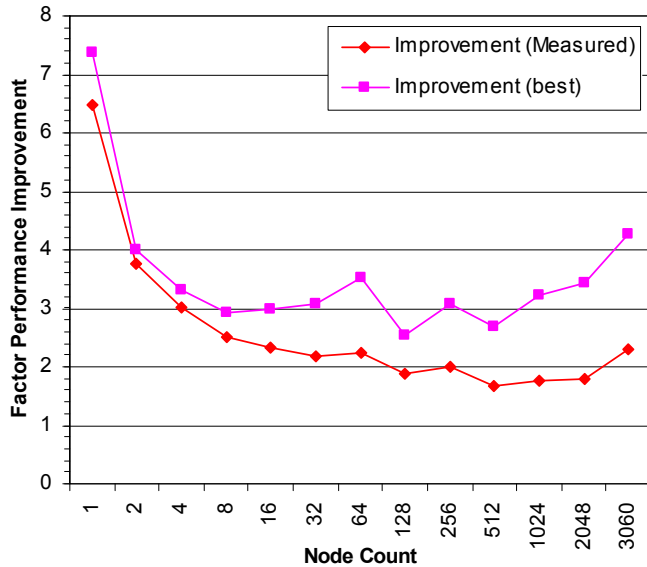

Figure 14. Performance improvements of Sweep3D between the use of the PowerXCell 8i and the Opteron processors in Roadrunner.

VII.   CONCLUSIONS

In this work we have introduced the architecture of a large-scale hybrid system being deployed at LANL in 2008. Roadrunner contains both conventional micro-processors alongside IBM PowerXCell 8i accelerator processors. We have presented the basic system characteristics of Roadrunner and of the new PowerXCell 8i chip. Performance characteristics of the PowerXCell 8i show a significant performance improvement over the earlier implementation of the Cell Broadband Engine (Cell BE) by a factor of 7x on double-precision floating point operations. We also have illustrated the high communication performance that is available within a compute-node as well as between compute nodes.

Though Roadrunner is the first system to achieve over a sustained petaflop on the LINPACK benchmark, the real benefit of the system results from achieved application performance. In this work we have shown that an optimized version of a demanding application, Sweep3D, achieves significant speedup on the PowerXCell 8i compared to current state of the art multi-core processors from AMD and Intel, as well as the older Cell BE processor. The implementation of Sweep3D is also compared to a previous implementation on the Cell BE, indicating a speedup of 3x. Finally performance results of Sweep3D at full scale of Roadrunner running on all 3,060 compute nodes utilizing all 97,920 SPEs were presented. These results show a 2x speedup over the base system, which is running early versions of the communication software. We then show through the use of highly accurate performance modeling the expected performance gains of the full Roadrunner system once the software layers are optimized to more closely match the achievable hardware performance. For small scale jobs the expected performance advantage is 10x, and for large-scale jobs the performance advantage is 5x. This clearly illustrates the performance potential of Roadrunner and advantage that a hybrid accelerator design can have in terms of performance.

REFERENCES

[1] HPCS, High Productivity Computing Systems initiative in DARPA, Available from http://www.darpa.mil/ipto/programs/hpcs

[2] Michael Gschwind, IBM, H. Peter Hofstee, Brian Flachs, Martin Hopkins, Yukio Watanabe, Takeshi Yamazaki, "Synergistic Processing in Cell's Multicore Architecture", IEEE Micro, 26(22):10-24, March 2006.

[3] J. A. Kahle, M. N. Day, H. P. Hofstee, C. R. Johns, T. R. Maeurer, and D. Shippy, Introduction to the Cell Multiprocessor. IBM Journal of Research and Development, 49(4):pp.589—604, 2005.

[4] Catherine Crawford, Paul Henning, Michael Kistler, Cornell Wright, Accelerating Computing with the Cell Broadband Engine Processor. ACM Computing Frontiers 2008, Ischia, Italy.

[5] Michael Gschwind, The Cell Broadband Engine: Exploiting Multiple Levels of Parallelism in a Chip Multiprocessor. International Journal of Parallel Programming, 35(3):pp. 233—262, 2007.

[6] Samuel Williams, John Shalf, Leonid Oliker, Shoaib Kamil, Parry Husbands, and Katherine Yelick. The Potential of the Cell Processor for Scientific Computing. In proc. ACM Int. Conf. on Computing Frontiers, 2006, Ischia, Italy.

[7] Samuel Williams, John Shalf, Leonid Oliker, Shoaib Kamil, Parry Husbands, and Katherine Yelick, Scientific Computing Kernels on the Cell Processor, Int. J. of Parallel Programming, 35(3):263–298.

[8] Sriram Swaminarayan, Kai Kadau, and Timothy C. Germann. 350-450 Tflops Molecular Dynamics Simulations on the Roadrunner General-purpose Heterogeneous Supercomputer. ACM Gordon Bell Prize finalist, n proc. of the ACM/IEEE SC2008 Conference, Austin, Texas, November 2008.

[9] Brian J. Albright, Benjamin K. Bergen, Lin Yin, Kevin J. Barker, and Darren J. Kerbyson. 0.365 Pflop/s Trillion-particle Particle-in-cell Modeling of Laser Plasma Interactions on Roadrunner. ACM Gordon Bell Prize finalist, in proc. of the ACM/IEEE SC2008 Conference, Austin, Texas, November 2008.

[10] Jack J. Dongarra, Piotr Luszczek, and Antoine Petitet. The LINPACK benchmark: Past, present and future. Concurrency and Computation: Practice and Experience, 15(9):803–820, August 2003.

[11] International Business Machines Corporation. Accelerated Library Framework for Hybrid–x86: Programmer's Guide and API Reference. Technical document SC33-8406-00. IBM SDK for Multicore Acceleration version 3, release 0. 2007. Available from http://tinyurl.com/5cyfc9.

[12] International Business Machines Corporation. C/C++ Language Extensions for Cell Broadband Architecture, Version 2.5. February 27, 2008. Available from http://tinyurl.com/5stuga.

[13] International Business Machines Corporation. Data Communication and Synchronization Library for Hybrid–x86: Programmer's Guide and API Reference. Technical document SC33-8408-00. IBM SDK for Multicore Acceleration version 3, release 0. 2007. Available from http://tinyurl.com/6kn98k.

[14] Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker, and Jack Dongarra. MPI: The Complete Reference, volume 1, The MPI Core. The MIT Press, Cambridge, Massachusetts, 2nd edition, September 1998. 1st edition is available from http://www.netlib.org/utk/papers/mpi-book/mpi-book.ps.

[15] John A. Turner, Roadrunner Applications Team: Cell and Hybrid Results to Date. Los Alamos National Laboratory presentation. Available from http://www.lanl.gov/orgs/hpc/roadrunner/rrinfo/RR%20webPDFs/Turner_Apps_v6_LA-UR.pdf

[16] John McCalpin. "Memory bandwidth and machine balance in current high performance computers", in IEEE Comp. Soc. Tech. committee on Computer Architecture (TCCA) Newsletter, pages 19-25, Dec. 1995.

[17] International Business Machines Corporation. Data Communication and Sychronization for Hybrid-x86 Programmer's Guide and API Reference, version 3.0. October 19, 2007.

[18] Richard L. Graham, Galen M. Shipman, Brian W. Barrett, Ralph H. Castain, George Bosilca, and Andrew Lumsdaine. Open MPI: A high-performance, heterogeneous MPI. In *Fifth International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks (HeteroPar'06)*, pp. 1–9, Barcelona, Spain, September 2006.

[19] Adolfy Hoisie, Olaf M. Lubek, Harvey J. Wasserman, Performance and Scalabilty Analysis of Teraflop-Scale Parallel Architectures Using Multidimensional Wavefront Applications. *Int. J. High Performance Computing Applications*, 14 (4): 330--347, November 2000. Available from http://www.c3.lanl.gov/pal/publications/papers/terraflop00:parallel.pdf

[20] F. Petrini, G. Fossum, J. Fernandez, A. L. Varbanescu, N. Kistler, M. Perrone, Multicore Surprises: Lessons Learned from Optimizing Sweep3D on the Cell Broadband Engine, in proc. Int. Parallel and Distributed Processing Symposium, Long Beach, California, 2007.

[21] A. Eichenberger et.al., Optimizing Compiler for the Cell Processor, PACT 2005, September 2005.

[22] Kevin Krewell. Cell moves into the limelight. *Microprocessor Report*, pp. 1–9, February 14, 2005.

[23] Scott Pakin. *Receiver-initiated Message Passing over RDMA Networks*. In Proceedings of the 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS 2008), Miami, Florida, April 2008. Available from http://www.c3.lanl.gov/PAL/publications/papers/Pakin2008:cellmsg.pdf.

[24] Thomas Chen, Ram Raghavan, Jason N. Dale, and Eiji Iwata. Cell Broadband Engine Architecture and its first implementation—a performance view. *IBM Journal of Research and Development*, 51(5):559–572, September 2007. Available from http://www.research.ibm.com/journal/rd/515/chen.pdf.

[25] Ashwini K. Nanda, J. Randal Moulic, Robert E. Hanson, Gottfried Goldrian, Michael N. Day, Bruce D. D'Amora, and Sreeni Kesavarapu. Cell/B.E. blades: Building blocks for scalable, real-time, interactive, and digital media servers. *IBM Journal of Research and Development*, 51(5):573–582, September 2007. Available from http://www.research.ibm.com/journal/rd/515/nanda.pdf.