# Cell/B.E. blades: Building blocks for scalable, real-time, interactive, and digital media servers

A. K. Nanda
J. R. Moulic
R. E. Hanson
G. Goldrian
M. N. Day
B. D. D'Amora
S. Kesavarapu

*The Cell Broadband Engine™ (Cell/B.E.) processor, developed jointly by Sony, Toshiba, and IBM primarily for next-generation gaming consoles, packs a level of floating-point, vector, and integer streaming performance in one chip that is an order of magnitude greater than that of traditional commodity microprocessors. Cell/B.E. blades are server and supercomputer building blocks that use the Cell/B.E. processor, the high-volume IBM BladeCenter® server platform, high-speed commodity networks, and open-system software. In this paper we present the design of the Cell/B.E. blades and discuss several early application prototypes and results.*

## 1. Introduction

The rapid deployment of high-speed Internet and consumer broadband technologies, the creation and popularity of rich media content, and the proliferation of open-source infrastructure have enabled a new breed of streaming and interactive digital media applications that would not have been possible a few years ago. This new breed of applications encompasses a wide range of areas such as gaming, streaming media, medical imaging, video surveillance, three-dimensional and real-time rendering, collaborative engineering design, virtual worlds, military simulation, seismic computing, and financial modeling. These applications demand a large amount of numerically intensive and streaming-oriented computing power that is traditionally associated with expensive, high-end supercomputers.

Fortunately, the raw numeric computational power of high-volume processors, such as those designed for gaming consoles, has been growing at a much faster rate than that of traditional commodity processors and vector processors. For example, the Cell Broadband Engine[†] (Cell/B.E.) processor, designed by Sony, Toshiba, and IBM [1, 2], offers more than 200 Gflops of single-precision, floating-point performance in a single 3.2-GHz chip. This computing capability is an order of magnitude higher than what today's comparable desktop processors can achieve, such as the 32-bit Intel Architecture** (IA-32) or the IBM PowerPC Architecture* core. The availability of a supercomputing level of performance in

commodity processors intended for the entertainment markets presents a platform opportunity with an unprecedented level of cost performance for these numerically intensive applications.

Cell/B.E. blades were developed by IBM for high-performance, scale-out servers, and they are targeted toward a variety of highly interactive digital media, real-time, streaming, and supercomputing applications. The generic Cell/B.E. blade architecture takes advantage of four key high-volume and open-system technologies, namely

- The IBM BladeCenter* platform.
- The Cell/B.E. processor.
- Open-system software, including the Linux** operating system (OS).
- Commodity high-speed switches such as InfiniBand** and Ethernet.

As shown in **Figure 1**, Cell/B.E. blades fit into the IBM BladeCenter server platform [3], which also accommodates a variety of blade designs based on commodity IBM, Intel, and AMD processors. The BladeCenter platform allows modular, scalable clustering of these processor blades using high-speed commodity interconnection networks such as InfiniBand or Ethernet. Cell/B.E. blades add significant floating-point, vector, and integer stream processing power to the traditional processing capabilities of other blade types. Configuring a

**573**

A. K. NANDA ET AL.
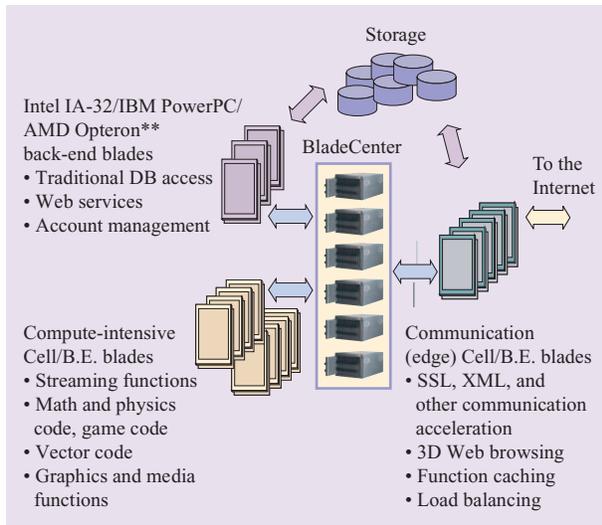
**Figure 1**

System architecture using Cell/B.E. blades. (DB: database; SSL: Secure Sockets Layer; XML: extensible markup language; IA-32: 32-bit Intel Architecture.)
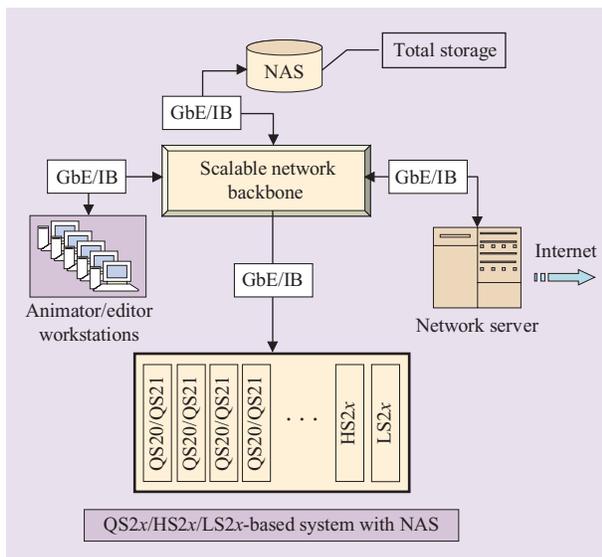


**Figure 2**

Example of a system configuration for digital content creation. (NAS: network-attached storage; GbE: Gigabit Ethernet; IB: InfiniBand; QS2$x$, HS2$x$, LS2$x$: IBM blade products based on AMD, Intel, and Cell/B.E. processors, respectively.)

BladeCenter chassis with a mix of different types of processor blades, storage, and interconnect networks allows the server hardware and software platform to be optimized for a particular target application, based on its need for different types of computation and communication. For example, a Web server environment could include a few Cell/B.E. blades to help accelerate security functions while the remainder of the work would need a large number of traditional blades. On the other hand, a gaming server, a war simulation server, or a video surveillance server could have hundreds or thousands of Cell/B.E. blades to perform media-, physics-, and image-intensive functions and only a few traditional blades to provide support functions such as user account management and database access. **Figure 2** shows a specific example of a parallel "render farm" for digital content creation that could be used in a movie, game, or broadcasting production studio.

In this paper we present the hardware and software architecture and provide a few application examples using the first generation of Cell/B.E. blades. Section 2 describes the Cell/B.E. blade hardware. Section 3 describes the Cell/B.E. blade software and various programming models. Several application examples are presented in Section 4, and the challenges and opportunities of this new platform are provided in Section 5. Finally, Section 6 concludes the paper.

## 2. Cell/B.E. blade hardware

### Cell/B.E. processor
The Cell/B.E. processor is primarily targeted at digital media and entertainment devices such as gaming consoles and high-definition televisions. The processor incorporates a traditional 64-bit PowerPC Architecture core, extended through tight integration of multiple, cooperative offload vector processors, or synergistic processor elements (SPEs). Each SPE consists of a 128-bit-wide vector, single-instruction multiple-data (SIMD) synergistic execution unit (SXU). Each SPE is fed from a dedicated local store (LS) and is linked to an on-chip coherent interconnection bus, called the element interconnect bus (EIB), for communication and cooperation with other on-chip elements or off-chip resources. The first-generation Cell/B.E. processor combines a dual-threaded, dual-issue 64-bit PowerPC Architecture–compliant PowerPC processor element (PPE) with eight SPEs. This unique on-chip vector multiprocessor design, along with the very high memory and input/output (I/O) bandwidths of the Cell/B.E. processor, results in floating-point and integer streaming performance that is an order of magnitude greater than that of any other high-volume processor currently on the market.

The PPE is optimized for design frequency and power efficiency, by utilizing short pipeline depths, avoiding long wiring runs, and limiting communication delays. Hence, the design of the PPE is more simplified than that

of the more recent four-issue out-of-order processors. The SPE implements a separate instruction set architecture (floating point and integer) that is optimized for power and performance on compute-intensive and media applications with 128-bit-wide vector, SIMD SXUs. The SPE operates on an LS memory (256 KB) that stores instructions and data that are transferred between this local memory and system memory by asynchronous, coherent direct memory access (DMA) commands that are executed by the memory flow control unit included in each SPE. The LS organization introduces an additional level of memory hierarchy beyond the registers that provide local storage of data in most processor architectures. This provides a mechanism to combat the "memory wall" [2], by allowing many memory transactions to be performed simultaneously without requiring the deep speculation that drives high degrees of inefficiency on other processors.

### IBM BladeCenter high-volume server platform

The first-generation IBM BladeCenter [3] is a modular chassis that is capable of housing 14 current standard blades; first-generation Cell/B.E. blades occupy two slots, so a chassis can accommodate up to seven Cell/B.E. blades. The BladeCenter chassis allows individual blades to be physically mounted in one standard 19-inch equipment rack and to share system resources such as switching or interconnect, storage, networking or communication fabrics, system management, power distribution, and cooling of air-moving devices.

Blade servers are a relatively new technology that has captured industry focus because of their modular design, which can reduce cost with more efficient use of valuable floor space, and their simplified management, which can help speed up such tasks as deploying, reprovisioning, updating, and troubleshooting hundreds of blade servers. The standardization of the blade form factor in the BladeCenter platform makes it an ideal candidate for flexible, adaptable system configurations in which various types of processor, storage, and interconnect or communication blades can be plugged into a single chassis.

### Cell/B.E. blade

A first-generation Cell/B.E. blade is shown in **Figure 3**. The two Cell/B.E. processors in a blade are configured as a two-way, symmetric multiprocessor (SMP); each processor runs with a clock frequency of 3.2 GHz and can deliver peak single-precision, floating-point performance of 217.6 Gflops. Memory is directly connected to each of the two Cell/B.E. processor chips via high-speed Rambus XDR** random access memory (RAM) interfaces. Each Cell/B.E. chip is also connected to a

**Figure 3**
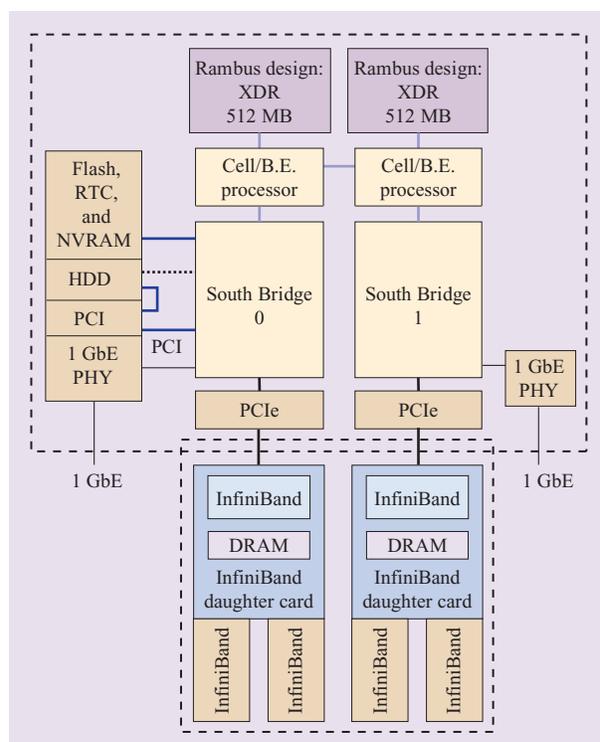
First-generation Cell/B.E. blade.



**Figure 4**

Cell/B.E. blade functional block diagram. (RTC: real-time clock; NVRAM: nonvolatile RAM; HDD: hard disk drive; GbE: Gigabit Ethernet; PHY: circuit providing bridge between digital and modulated parts of the interface; PCIe: PCI Express; DRAM: dynamic random access memory.)

separate South Bridge[1] chip that provides I/O functionality. The bandwidth of each XDR RAM interface is 25.6 GB/s, the bandwidth of the interface

---

[1]Also known as the *I/O controller hub* (ICH), the South Bridge chip implements the "slower" capabilities of the motherboard.

**575**
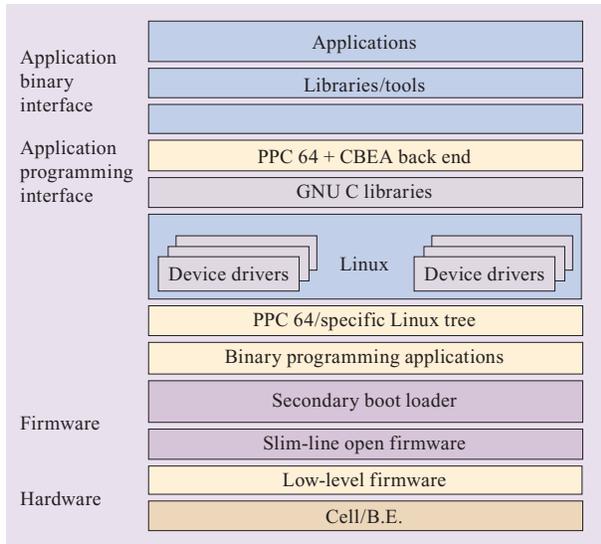
**Figure 5**

Cell/B.E. blade software stack. (CBEA: Cell Broadband Engine Architecture; PPC 64: 64-bit PowerPC.)
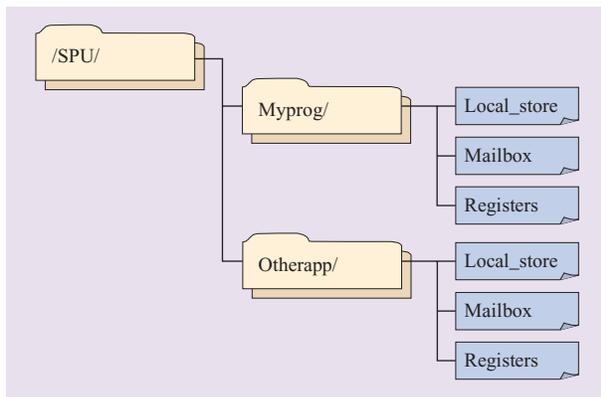


**Figure 6**

File abstraction of Cell/B.E. programming model. (SPU: synergistic processor unit.)

between the Cell/B.E. processors is 20 GB/s in each direction, and the effective bandwidth of the South Bridge I/O interface from each Cell/B.E. chip is ∼1.0 GB/s in each direction, as shown in **Figure 4**.

The South Bridge 0 chip provides a PCI Express[**,2] (PCIe) channel, a PCI bus interface, an integrated drive electronics (IDE) channel used for an onboard hard disk, a Gigabit Ethernet[3] (GbE) media access controller

(MAC), Universal Serial Bus (USB) 2.0 ports, a universal asynchronous receiver/transmitter (UART), and an external bus controller. Attached to the external bus are a Flash erasable, programmable read-only memory (EPROM) device (8 MB), 1 MB of battery-backed nonvolatile RAM (NVRAM), and a battery-backed real-time clock (RTC). On the South Bridge 1 chip, only the PCIe channel and the GbE MAC are used. Optionally, up to two InfiniBand daughter cards can be connected via the PCIe ports. The Cell/B.E. blade also includes an IDE hard disk (which provides a boot mechanism for the OS).

## 3. Cell/B.E. blade software

### Software stack

The Cell/B.E. processor was designed with the appropriate architectural characteristics for multi-OS support, including the real-time game OS. However, the system software foundation for the Cell/B.E. blade is the PowerPC 64-bit Linux SMP–enabled kernel, with appropriate extensions to utilize the SPEs. Patches incorporating the extensions are integrated with a standard Fedora[**] core[4] build, and they provide the application programmer with a familiar GNU C library (GLibC) application programming interface to the dual-processor, two-way simultaneous multithreading (SMT) hardware features and OS utilities. The software stack, as shown in **Figure 5**, has a full GNU compiler collection[5] (GCC) development environment that includes compilers and debuggers for the PPE and the SPE, a linker/loader, and low-level profiling support. A set of Cell/B.E. processor–optimized reference libraries is provided for application developers to use as templates for creating application-specific libraries. Also incorporated in the Linux OS build are implementation-specific device drivers for I/O support, including GbE, USB, Serial console, PCI, PCIe, interrupt controllers, and other blade-level hardware for testing, performance monitoring, reliability, availability, and serviceability features that are exposed to the IBM BladeCenter maintenance console. The OS interfaces with the blade hardware via low-level firmware and slim-line open firmware (SLOF).

### Exploiting the SPEs

There are several models for exploiting the SPEs. The SPEs can be abstracted as Linux OS devices and manipulated by the programmer via standard file I/O mechanisms. **Figure 6** shows a high-level representation of the file abstraction Cell/B.E. processor programming model. In this model the programmer communicates with

---

[2]*PCI Express* is a computer system bus/expansion card interface format that replaces the earlier PCI format.
[3]*Gigabit Ethernet* is a technology for transmitting Ethernet packets at a rate of 1 Gb/s.

[4]*Fedora core* (developed by Red Hat originally for Linux) is a general-purpose OS containing only free and open-source software.
[5]*GNU compiler collection* is a set of free programming language compilers produced by the GNU Project.

the SPEs and their components via standard file I/O commands such as open/close, read/write, and I/O control (*ioctl*).

Programmers can also exploit the SPEs through task-based abstraction in user-space libraries. SPE programs are managed as either single or grouped thread primitives of the on-chip PowerPC core (i.e., the PPE). For example, library functions are provided to create, destroy, and execute SPE threads. Additional library support exists for synchronizing access to system memory areas shared by both PPE and SPE programs. Primitives are provided for "mailbox," DMA, and event management functions. GNU tool chain elements include the GNU debugger (gdb) for the PPE and SPEs, as well as OProfile.[6] Separate PPE (standard GCC from ppc64[7] Linux OS distribution) and SPE compilers are included in development tools, along with an interface description language (IDL) compiler to define interface between the PPE main program and tasks or threads targeted for off-load execution on one or more of the SPEs. **Figure 7** shows a typical build process for Cell/B.E. processor executables using this model.
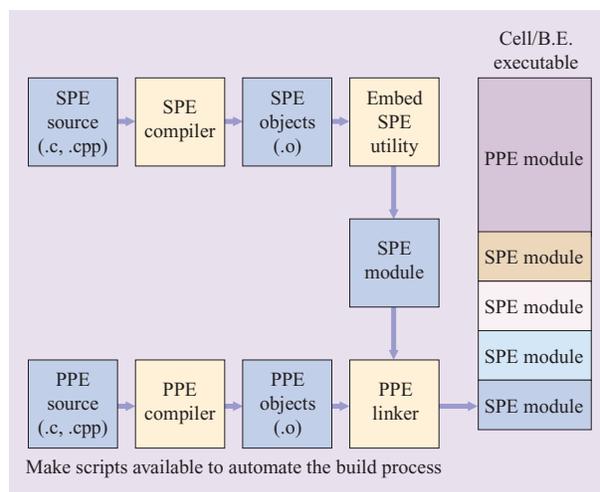
### Cell/B.E. blade programming models

#### *Single-source optimizing compiler with OpenMP*
In addition to the open-source GCC compiler, a "single-source" compiler prototype based on the IBM optimizing compiler technology is also available to programmers for ease of use and additional productivity and performance [4]. This optimizing compiler is a parallelizing, "SIMD-izing" compiler that generates code from a single C or Fortran input source file, using multiple binaries to target the PPE and SPE units. The goal is to generate highly optimized code for the multiple levels of parallelism while providing an abstraction of the underlying architectural intricacies, thus allowing the user to develop applications for a parallel architecture with a single shared memory image.

At the core of the IBM Cell/B.E. compilation strategy is the technique for abstracting the small local memories of the SPE. An SPE can directly access its LS only, requiring a DMA transfer whenever it reads or writes locations in the shared system memory. This imposes a nontrivial burden on the programmer, especially for large programs accessing significant amounts of data. The compiler-controlled software cache, memory hierarchy optimizations, and code partitioning techniques assume all data resides in shared system memory and enable automatic transfer of code and data while preserving coherence across all of the local SPE memories and system memory. This infrastructure provides the
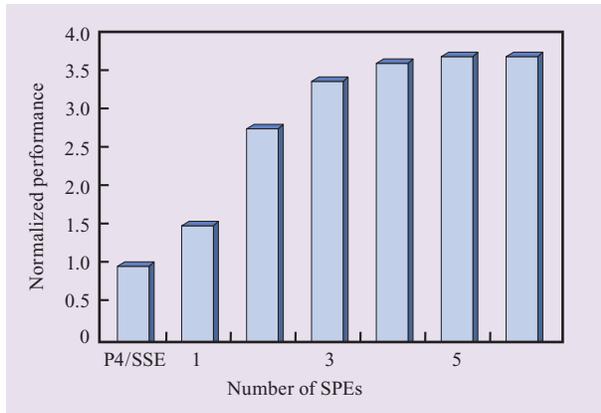
underpinning for enabling parallelism across the SPEs using Open Multiprocessing (OpenMP**) pragmas.[8]

#### *Message-passing interface (MPI) microtasks*
Another mechanism to hide the LS management chores from a Cell/B.E. processor programmer is provided through an MPI "microtasking" prototype. This mechanism allows a programmer to parallelize the application program into a collection of microtasks, each of which can fit in the LS. The microtasks provide programmers with two advantages. First, they can use the MPI, a widely used standard interface that can hide the details of communication hardware from programmers. Second, the mechanism optimizes the execution of microtasks by converting message-passing programs into ones for a streaming model, in which computation kernels and messages between them stream through SPEs. The Cell/B.E. processor can execute programs in this model very efficiently because of its high-speed on-chip DMA mechanism. For this conversion, the microtask system employs a preprocessor that optimizes the scheduling of computations and communications by exploiting explicit communications in the MPI programming model.

The preprocessor first divides each microtask into a set of basic tasks, which represent a unit of communications that can proceed without interacting with other basic tasks in the middle of the execution. The processor then creates a precedence graph of basic tasks and statically schedules them. A novel static scheduler exploits the high-speed on-chip communication mechanism on the Cell/B.E.

---

[6]*OProfile* is a systemwide profiler for Linux.
[7]*Ppc64* is an identifier frequently used when compiling source code to target architecture for applications optimized for 64-bit PowerPC processors.

[8]OpenMP pragmas make up a compiler directive, communicating pragmatic information.

**577**

**Figure 8**

P4/SSE vs. SPE integration code performance, normalized to P4/SSE. (SSE: streaming SIMD extensions; SIMD: single-instruction multiple-data; P4: Intel Pentium 4.)

processor. Namely, the preprocessor identifies a set of basic tasks that can be gang-scheduled and can directly communicate among each other without going through the off-chip system memory. Since the number of microtasks is typically larger than the number of SPEs, a context switch may occur during the execution of a microtask. Initial experimental results have shown that the runtime overhead resulting from context switches is reasonably small.

## 4. Early experience with Cell/B.E. blade applications

Several applications and application kernels have been ported to Cell/B.E. blades and reported in the literature [5–10]. In this paper we describe three specific examples.

### Online game prototype using Cell/B.E. blades as a remote server

This prototype is an example of an online game application running on a Cell/B.E. processor–based server. The game prototype was designed with an emphasis on physics-based modeling of rigid-body dynamics in collaboration with Episode, Inc.[9] The premise of this project is that next-generation online games will rely heavily on server-based physical simulations to add a dimension of behavioral realism not available in the current generation of game platforms. The game prototype leverages the SPEs to perform a hybrid integration calculation required to compute rigid-body displacements in a multibody game scene.

*Game play*

The story line for the game involves mechanical robots attacking a city inhabited by humans who are forced to defend themselves by using a variety of weapons to disable the robots. A robot can be destroyed by directing weapons fire at vulnerability points located near its arm, leg, or joints, for example. Alternatively, humans can destroy static structures causing a robot to lose balance and fall or be hit by falling debris.

Making the articulation of robot and human joints result in realistic-looking movement depended on solving several technical challenges. First, collision detection was needed to determine when moving bodies were intersecting with other moving bodies as well as static bodies such as walkways, buildings, and terrain. Collision detection was implemented with a two-phase approach using a broad phase to quickly eliminate bodies that could not physically collide with each other during any given frame update and a narrow phase to specifically determine whether pairs of bodies would intersect with each given frame update. The entire process is integer and floating-point intensive. A large robot with many articulating joints is described in [7]. Each pair of joints in a robot is represented in a database of collision bodies. Bodies that are static are referred to as "sleeping," whereas moving bodies are referred to as "awake."

Client–server synchronization was another important challenge. A round-trip communication between client and server could not be tolerated for a short player response time; because of network latency, it would be necessary to compensate the client for the response from the player.

*Game code performance*

We profiled the server-side components to determine qualitative and quantitative performance differences between a 3.2-GHz Intel Pentium** 4 processor system and a 2.4-GHz Cell/B.E. processor–based system.

One point to note is that performance profiling of the Cell/B.E. processor was done on early hardware that was not running at potential maximum clock speed and that had only six (as opposed to eight) functioning synergistic processor units (SPUs).

We used the "Ben25" benchmark [7] to measure integration performance. It is a synthetic benchmark that contains 25 "ben" robots, and it is designed to stress integration.

**Figure 8** shows the relative performance of Cell/B.E. blades compared with a higher speed Pentium 4 processor with streaming SIMD extensions (SSEs). A performance advantage of 1.5 times is obtained for one SPE when compared with a higher speed Pentium 4 with SSEs and with early hardware. In a system simulating millions of objects in a game or virtual world environment, the

[9]Montreal, Quebec, Canada.

improved performance and cost savings from utilizing the Cell/B.E. blades could be very significant.

### Video surveillance server prototype

The IBM smart surveillance system [5] relies on a number of visual analysis technologies to detect moving objects in video and track those objects, making it a primary target application for the Cell/B.E. blades. The smart surveillance engine processes incoming video from a camera and stores the viewable video index in the activity database, as shown in **Figure 9**.

The video surveillance middleware provides support for querying the database for various patterns, which makes it useful for security personnel to either monitor the cameras in real time or to retrieve the video offline. The smart surveillance engine consists of two main algorithms: the background subtraction (BGS) algorithm to detect objects and the object tracker to track the detected objects. In addition, the camera streams are stored in a database using H.264[10] encoding, which is a good match for the stream processing capability of the Cell/B.E. processor. The BGS module combines evidence from differences in color, texture, and motion. Consisting of compute-intensive image processing algorithms, the BGS can benefit from media processing elements of various processors. The primary challenge in porting the BGS code to the Cell/B.E. processor was getting it to run on multiple SPEs, because the code does not fit into the LS of one SPE. After looking at the profile data and identifying data-flow patterns, the code was partitioned into four modules, with each module assigned to run on a dedicated SPE. When processing a single camera input, all of these SPEs run in sequence, with results flowing from one SPE to the other. However, when processing input from multiple cameras, these four SPEs run in parallel, with each SPE processing a different camera input and at a different stage of processing. Because the data file is too large to store in the LS of the SPE, it is stored in system memory, and the SPE brings in the data as needed before processing it and transfers the results back to system memory.

Preliminary algorithm-level performance estimates of some of the surveillance kernels optimized for the Cell/B.E. processor indicate a significant performance improvement compared with traditional desktop processors (**Table 1**). By utilizing the media processing capabilities of the Cell/B.E. processor, these image processing algorithms are able to process more camera inputs than a comparable general-purpose desktop processor. As a result, with the Cell/B.E. processor–based blades, the number of servers required in a large surveillance system with hundreds or thousands of
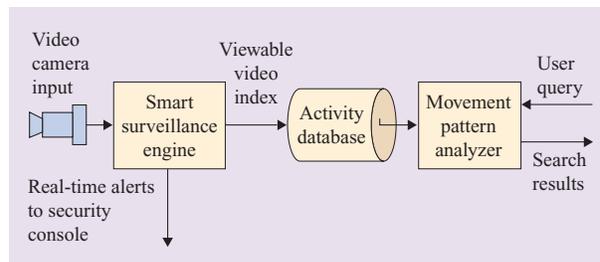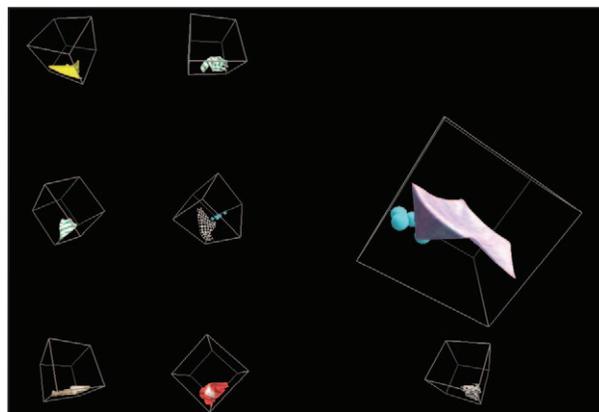


**Figure 9**

IBM smart surveillance engine.



**Figure 10**

Multiple cloth simulations using Cell/B.E. blade.

cameras can be significantly reduced, which in turn decreases the total cost of ownership.

### Cloth simulation prototype on Cell/B.E blades

#### Description

The goal for developing this application prototype (with our partners Alias Systems[11]) was to test the premise that the SPE of the Cell/B.E. processor could provide the necessary acceleration for real-time, interactive simulation of a soft-body material such as cloth. This type of simulation is very compute intensive, requiring many calculations per second to be performed on a large set of data. **Figure 10** shows eight simultaneous simulations, with each being represented visually by a cube with a piece of cloth folded or constrained at one or more positions. The folding and falling behavior of the pieces of cloth is in response to gravitational and collision

---

[10]*H.264* is a compression algorithm in MPEG-4 (also known as Advanced Video Coding [AVC]).

**579**
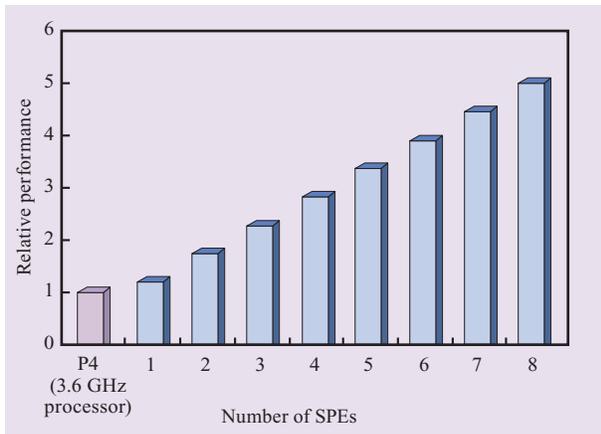
**Figure 11**

Cloth simulation performance. (VMX is a *floating point* and integer *SIMD instruction set*.)

forces simulated by the user as they interact with the enclosing cube. The cloth simulation computations are performed on one or more Cell/B.E blade prototypes running the Linux OS and the results of those computations are sent to an Apple Power Mac** G5 client for display.

### Implementation on the Cell/B.E. processor
Porting to the Cell/B.E. processor required efficient use of the nine processing cores and ten user threads of the Cell/B.E. processor. First, we sought to establish an effective data flow between the PPE and SPE by implementing a single simulation on one SPE and transferring the output data via a socket to the client system, the Power Mac G5. Once this was accomplished, in order to support additional simulations, the original single-threaded code needed to be parallelized over the eight SPEs. The primary challenge was getting as much code executing on an SPE thread as possible. This resulted in moving a significant amount of scalar code to the SPE including branch code, which is typically not optimal on an SPE but, with appropriate use of branch

hints, can be made to perform well. The remaining code executing on the PPE was partitioned as two execution threads, utilizing the SMT hardware support. One thread was tasked with communicating with the client renderer, accepting user events and transferring simulation output vertices to the client. The other thread was used for controlling SPE threads and converting user input into force vectors, which could be transferred via DMA to the appropriate SPE thread.

### Performance results
The cloth simulation leverages the scalable parallel processing capabilities of the Cell/B.E. processor by simultaneously running eight instances of the simulation per Cell/B.E. processor—one on each of the SPEs. Initial performance on a prototype 2.4-GHz Cell/B.E. processor is approximately five times as many simulation frames per second as a 3.6-GHz Pentium 4 class processor, as shown in **Figure 11**. At this early stage, not all potential vectorizable code has been optimized for the Cell/B.E. processor. The prototype executables were built using IBM XL C compilers for the PPE and the SPE.

When multiple Cell/B.E. processors are available, as in the Cell/B.E. blade prototype, the cloth solver can take advantage of the additional modular processing power by utilizing multiple Cell/B.E. blades networked in a BladeCenter chassis. We have simulated up to 32 pieces of cloth simultaneously on a two-blade prototype system. The performance of cloth simulation is heavily dependent on large matrix operations; hence, there is an opportunity to utilize the SPE SIMD units when computing vertex displacements as a result of changing force and gravitational vectors.

## 5. Challenges and opportunities for Cell/B.E. processor–based systems
The Cell/B.E. processor and systems using the Cell/B.E. processor bring a new architecture to the computing field. Although the PowerPC core in the Cell/B.E. processor provides continuity and compatibility with legacy PowerPC-based application code, the Cell/B.E. processor introduces a new flexible, hybrid computing model through the SPEs, which integrate elements of parallel,

**Table 1** Performance of various surveillance engine modules on a Cell/B.E. processor compared with a general-purpose processor at the same clock speed

| Module | % of execution time that can be vectorized | Potential increase in performance (vs. general-purpose processor at the same clock speed) |
| --- | --- | --- |
| Smart surveillance engine with background subtraction algorithm | 70% | 8× to 16× |
| Smart surveillance engine with object tracking | 45% | 8× to 16× |
| H.264 encoder | 55% | 8× to 16× |

**580**

pipelined, and vector computing. These features bring new challenges and new opportunities.

The foremost challenge for the Cell/B.E. platform is to hide the new architectural features from the programmer by using appropriate compiler and parallel programming tools for ease of use and programmer productivity, and for extracting performance from this highly parallel chip architecture. Some of the efforts already underway are described in Section 3. However, there is a long way to go toward establishing a robust software ecosystem around the Cell/B.E. processor before it can establish itself as a widely used platform for a broad class of applications. The programming model challenge also brings the opportunity to seek innovative solutions to these difficult tasks to architecture, compiler, and parallel programming researchers.

The high volume of gaming consoles gives the Cell/B.E. processor competitive commodity pricing; combined with an order of magnitude greater streaming, floating-point, and vector performance compared with other commodity processors, this makes the Cell/B.E. processor a very attractive platform. The Cell/B.E. blades extend this volume economy and superior performance of the Cell/B.E. processors into the area of affordable, scalable servers and supercomputers using the high-volume BladeCenter server platform and high-speed commodity switches. Tying Cell/B.E. blades to Linux and open-source software brings the additional advantage of mass participation in building the ecosystem. These factors highlight the opportunities around the Cell/B.E. blades platform for government laboratories in the areas including, but not restricted to, nuclear simulation, security, video surveillance, and war simulation servers. For the entertainment industry, it brings opportunities in areas such as gaming, virtual worlds, broadcasting, and movie creation. The communication industry would find a good match with this platform in digital media processing, encryption/decryption, compression/decompression, and other compute-intensive applications. Like other examples, the life sciences industry would find Cell/B.E. blades useful in drug simulation and medical imaging, the finance industry in financial modeling, the petroleum industry in seismic modeling, and the design industry in three-dimensional collaborative design.

## 6. Summary
In this paper, we present the hardware and software design of Cell/B.E. blades, the building blocks for affordable, scalable servers and supercomputers for streaming, real-time, interactive, digital media, and numerically intensive applications. We also discuss the various programming models that a programmer can use to efficiently utilize the architectural resources of the Cell/B.E. processor. Three application prototypes on Cell/B.E. blades are discussed. The results show a significant performance advantage of the Cell/B.E. blades

over blades using IA-32 processors. We also discuss the challenges and opportunities for Cell/B.E. processor–based systems.

## References
1. B. Flachs, S. Asano, S. H. Dhong, H. P. Hofstee, G. Gervais, R. Kim, T. Le, et al., "The Microarchitecture of the Streaming Processor for a Cell Processor," *Proceedings of the IEEE International Solid-State Circuits Symposium*, February 2005, pp. 184–185.
2. J. A. Kahle, M. N. Day, H. P. Hofstee, C. R. Johns, T. R. Maeurer, and D. Shippy, "Introduction to the Cell Multiprocessor," *IBM J. Res. & Dev.* **49**, No. 4/5, 589–604 (2005).
3. IBM Corporation, IBM BladeCenter; see *http://www.ibm.com/servers/eserver/bladecenter/*.
4. IBM Cell compiler project home page; see *http://domino.research.ibm.com/comm/research_projects.nsf/pages/cellcompiler.index.html*.
5. A. Hampapur, L. M. Brown, J. Connell, M. Lu, H. Merkl, S. Pankanti, A. W. Senior, C.-F. Shu, and Y. L. Tian, "Multi-scale Tracking for Smart Video Surveillance," *IEEE Trans. Signal Processing* **22**, No. 2, 38–51 (March 2005).
6. RIDMS-1: First Workshop on Real Time and Interactive Digital Media Supercomputing; see *http://domino.research.ibm.com/comm/research_people.nsf/pages/ashwini.RIDMS1.html*.
7. RIDMS-2: Second Workshop on Real Time and Interactive Digital Media Supercomputing; see *http://domino.research.ibm.com/comm/research_projects.nsf/pages/ridms2.index.html*.
8. B. D'Amora, K. Magerlein, A. Binstock, A. Nanda, and B. Yee, "High-Performance Server Systems and the Next Generation of Online Games," *IBM Syst. J.* **45**, No. 1 (January 2006).
9. Y. Liu, H. Jones, S. Vaidya, M. Perrone, B. Tydlitát, and A. K. Nanda, "Speech Recognition Systems on the Cell Broadband Engine Processor," *IBM J. Res. & Dev.* **51**, No. 5, 583–591 (2007, this issue).
10. T. Chen, R. Raghavan, J. N. Dale, and E. Iwata, "Cell Broadband Engine Architecture and its First Implementation—A Performance View," *IBM J. Res. & Dev.* **51**, No. 5, 559–572 (2007, this issue).

**581**

**Ashwini K. Nanda**   *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598 (ashwini@us.ibm.com).* As the Chief Architect of Quasar/Cell/B.E. systems, Dr. Ashwini Nanda established and managed the Quasar systems team in the IBM Systems and Technology group. He played a lead role in establishing the Cell/Quasar systems technology, product roadmap (including QS20, the first Cell/B.E. blade product), and business to focus on the emerging compute-intensive, streaming, real-time, and interactive applications. Prior to that, Dr. Nanda led research and prototyping of Cell/B.E. processor–based systems and their application at the IBM Thomas J. Watson Research Center, in Yorktown Heights, New York. Earlier at IBM Research, he established and managed the Scalable Server Architecture group for several years. His key research contributions include MemorIES (Memory Instrumentation and Emulation System) and High Throughput Coherence Controllers. Dr. Nanda was co-General Chair of the International Symposium on High Performance Computer Architecture (HPCA-7), he served on the editorial board of IEEE Transactions on Parallel and Distributed Systems, and co-edited a special issue of the IEEE Computer magazine. He holds ten U.S. patents and has published more than 40 papers on computer systems architecture, design, and performance.

**J. Randal Moulic**   *IBM Systems and Technology Group, Enterprise Systems Development, 2455 South Road, Poughkeepsie, New York 12601 (rmoulic@us.ibm.com).* Dr. Moulic is an IEEE Fellow and Research Staff Member at the IBM Thomas J. Watson Research Center. In 34 years of research and development work at IBM, he has participated in the development of many high-performance and personal computing systems, including the RS/6000*/Scalable Parallel supercomputers, IntelliStation*, and Cell/B.E. processor graphics workstations, ThinkPad** "TransNote" laptop, and QS20 Cell/B.E. processor server. He founded, organized, and directed the Deep Blue* computer chess project, initiating the first exhibition match events with World Champions Gary Kasparov and Anotoly Karpov. He has authored many technical papers, holds many patents, and is an Adjunct Professor at Columbia and Polytechnic Universities.

**Robert E. Hanson**   *IBM Systems and Technology Group, Enterprise Systems Development, 2455 South Road, Poughkeepsie, New York 12601 (rejhan@us.ibm.com).* In the past five years, Mr. Hanson has established and led the cross-IBM team to develop Cell/B.E. processor–based systems. He began this work as an emerging technology opportunity within IBM Research and subsequently moved to the Systems and Technology Group (STG) as the leader of the Quasar Design Center to set up the new product effort. This work has since resulted in the release of the original QS20 Cell/B.E. blade and two versions of the Cell/B.E. Software Development Kit, with several generations of the Cell/B.E. systems product in the pipeline to address real-time and digital media applications, as well as broader markets including financial services, medical imaging, aerospace and defense, high-performance computing, and several others. Prior to driving the Cell/B.E. processor–based system work, Mr. Hanson had made extensive contributions to the IBM systems business and technology as the Director of Microprocessor Development. This includes the first IBM CMOS-based zSeries* systems in the 1990s, which brought significant performance enhancements and surpassed the performance of previous-generation bipolar machines by 2.5 times, as well as the first IBM microprocessor capable of running at 1 GHz.

**Gottfried Goldrian**   *IBM Systems and Technology Group, Development Laboratory, Schoenaicherstrasse 220, D-71032 Boeblingen, Germany (Goldrian@de.ibm.com).* Mr. Goldrian received his diploma in electrical engineering from the Polytechnikum in Munich, Germany, in 1964. He worked for Siemens before he joined IBM in 1967. His first job at IBM was in the development of a digital recorder for computer problem analysis. Since then, he has worked on many different development projects in Boeblingen and San Jose, California. He was the lead designer and architect in the development of printer electronics and later in the development of zSeries I/O attachments. Mr. Goldrian is now a Distinguished Engineer and the lead system architect for all Cell/B.E. processor–based projects in Boeblingen.

**Michael N. Day**   *IBM Systems and Technology Group, STI Design Center, 11400 Burnet Road, Austin, Texas 78758 (mnday@us.ibm.com).* Mr. Day graduated with a B.S. degree in electrical engineering–computer science from the University of Texas at Austin in 1977. He joined IBM that same year as an engineer designing and implementing hardware and software for a large multi-user timesharing office product system including workstation controllers, full page displays, speech digitization, and filing systems. He then became the lead firmware and software architect for the first battery-powered IBM laptop with advanced power-management features. In 1987, he became a kernel subsystem architect on the IBM Unix OS project called AIX*. Mr. Day was elected to the IBM Academy of Technology in 1992, and he went on to become chief architect of AIX V4, delivering SMP support and kernel-based threads. He was one of the first engineers to be appointed IBM Distinguished Engineer in 1997. A year later, he led a real-time broadband video streaming project, introducing the MediaStreamer* product based on AIX. He then went on to drive the design and implementation of AIX on IA-64, and then moved to the STI project in 2001 as Chief System Software Architect, defining the programming features of the Cell/B.E. processor, enabling Linux and software tool chains to support various programming models for the Cell/B.E. processor. He also leads a team of programmers developing application libraries, test suites, workloads, and demonstration programs for the Cell/B.E. processor.

**Bruce D. D'Amora**   *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (damora@us.ibm.com).* Mr. D'Amora is a Senior Technical Staff Member and Digital Media Solutions Architect in the Emerging Systems Software group at the IBM T. J. Watson Research Center. He is currently focusing on the design of Cell/B.E.-based platforms to accelerate applications used for creating digital animation and visual effects. Mr. D'Amora has presented numerous talks over the last several years focusing on the uses and programmability of the Cell/B.E. for accelerating the creation of digital content. He was previously the Chief Software Architect for the 3D graphics development group at IBM Austin, where he led the OpenGL development efforts from 1991 to 2000. He holds B.S. degrees in microbiology and applied mathematics from the University of Colorado, as well as an M.S. degree in computer science from the National Technological University.

**Sreeni Kesavarapu**   *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (sulu@us.ibm.com).* Mr. Kesavarapu is a Senior Software Engineer currently leading the development of the digital video surveillance solution on the Cell/B.E. and working as a Lead Consultant in the Cell/B.E. ecosystem and solutions development in the Systems and Technology Group. He has previously contributed to a number of products in mobile and digital media systems, including MPEG and DVD playback software for various ThinkPads and the development of ThinkPad Transnote from concept to product. He received an M.S. degree in computer science from Polytechnic University and a B.Tech. degree in computer science and engineering from Andhra University.

**582**