

## Programmability, Part 1: Exploring different approaches to programming for Cell/B.E. platforms

Discover the programming flexibility available for the Cell Broadband Engine

Skill Level: Introductory

[Anita Bateman \(ajbatema@us.ibm.com\)](mailto:ajbatema@us.ibm.com)  
Certified Senior IT Architect, Cell Solutions  
IBM

14 Oct 2008

The programming flexibility available for the Cell Broadband Engine™ is a hot topic in the multicore community. This article discusses leveraging your existing skills to program for Cell/B.E.™, offers three programming approaches for Cell/B.E. systems, and introduces the various tools, software, and hardware available for the platform.

This article describes the flexibility available with programming for the Cell Broadband Engine, jointly developed by IBM, Sony, and Toshiba. This is a hot item of discussion in the multicore community. This article assumes you already have basic knowledge of Cell/B.E. architecture and the demonstrated performance. If this is your first exposure to Cell/B.E., read the following articles before continuing with this programming discussion:

- ["Introduction to the Cell Broadband Engine"](#) (IBM, October 2005).
- ["Programming high-performance applications on the Cell/B.E. processor, Part 1: Intro to Linux® on PlayStation 3"](#) (developerWorks, January 2007).
- ["Programming high-performance applications on the Cell/B.E. processor, Part 2: Program PlayStation 3 SPEs"](#) (developerWorks, February 2007).
- ["Programming high-performance applications on the Cell/B.E. processor, Part 3: Meet the SPU"](#) (developerWorks, February 2007).

- ["Programming high-performance applications on the Cell/B.E. processor, Part 4: Program the SPU for performance"](#) (developerWorks, March 2007).
- ["A Programming Example: Large FFT on the CBE"](#) (IBM, October 2005).

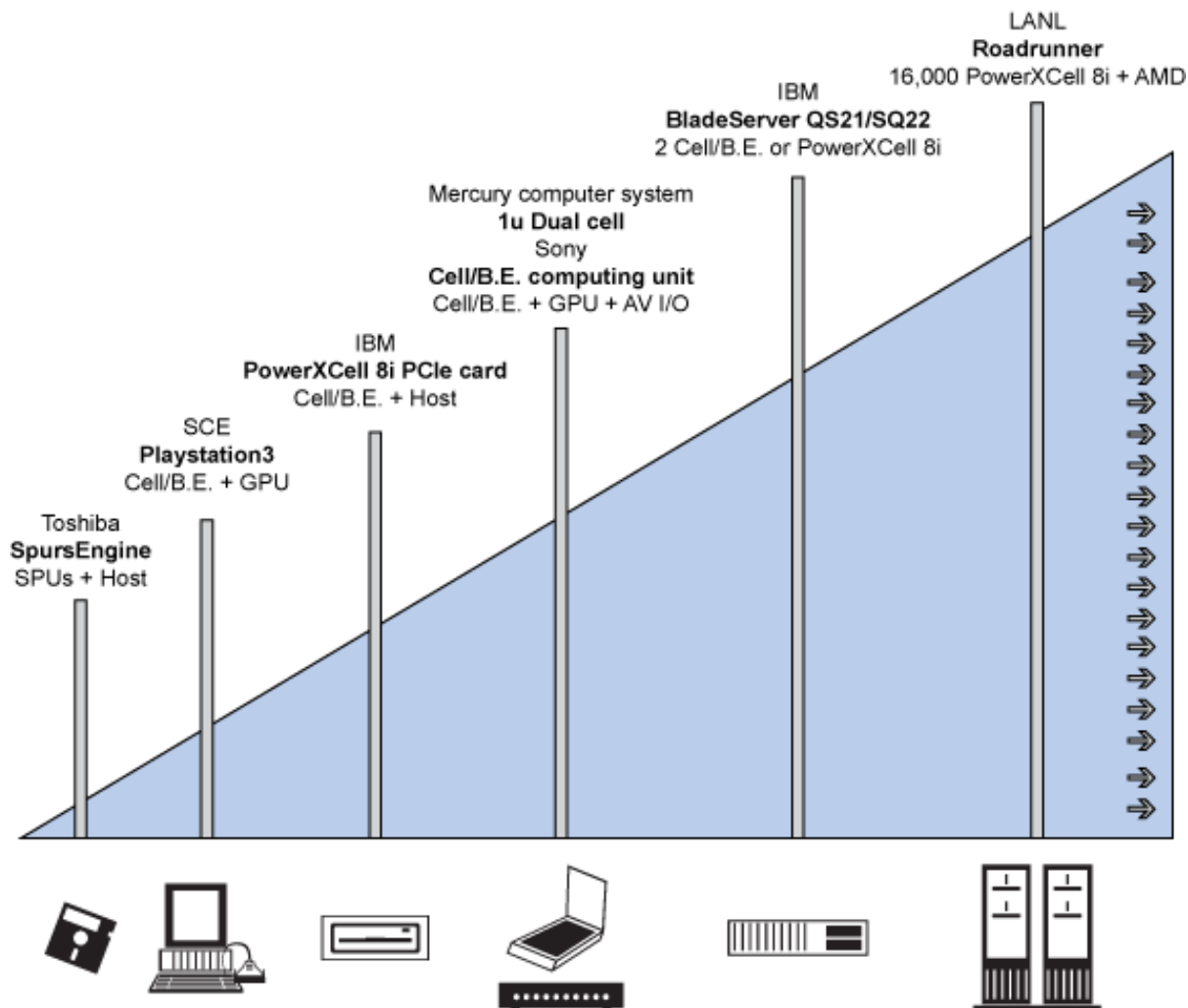
This article does the following:

- [Discusses leveraging your existing skills to program for Cell/B.E.](#)
- [Offers three programming approaches for Cell/B.E. systems](#)
- [Introduces the various tools, software, and hardware available](#)

## Reusing your existing skills

You can reuse your existing skills for Cell/B.E. development, including your programming skills, your IT administration skills, and your existing parallel programming skills. In addition, the Cell/B.E. hardware can plug into your existing infrastructure, easing the way for IT administration of your Cell/B.E. solution. Figure 1 shows the various cross-industry Cell/B.E. systems that run the same SPE code.

### **Figure 1. Cross-industry Cell/B.E. systems that run the same SPE code**



As you can see in Figure 1, there is a spectrum of Cell/B.E. systems that fully support Cell/B.E. code. The same SPE code runs across the spectrum from consumer products to business, enterprise, and high-performance computing systems. In addition, multiple vendors in the industry provide these systems, which demonstrates cross-industry support for Cell/B.E. technology.

Cell/B.E. can be programmed with standard programming languages, such as the C, C++, FORTRAN, and Java™ languages. You can use standard compilers and debuggers, such as the GNU toolchain and IBM XL C/C++/FORTRAN compilers, to support your development work. Common development tools and IDEs are also supported, including common code editors (vi, emacs), the Eclipse IDE, and partner development tools from Gedae and RapidMind.

Cell/B.E. gets standard Linux operating system support from software such as Red Hat, Fedora, and Terra Soft Solutions Yellow Dog Linux, as well as real-time operating system support from partners such as Wind River and Mentor Graphics.

Finally, you can use common libraries and frameworks, including IBM DaCS, ALF, DAV, FFT, BLAS, LAPACK, MASS, image-processing libraries, and Monte Carlo Random Number Generator. You can see a full listing of tools, software, and hardware available on Cell/B.E. at the end of this article.

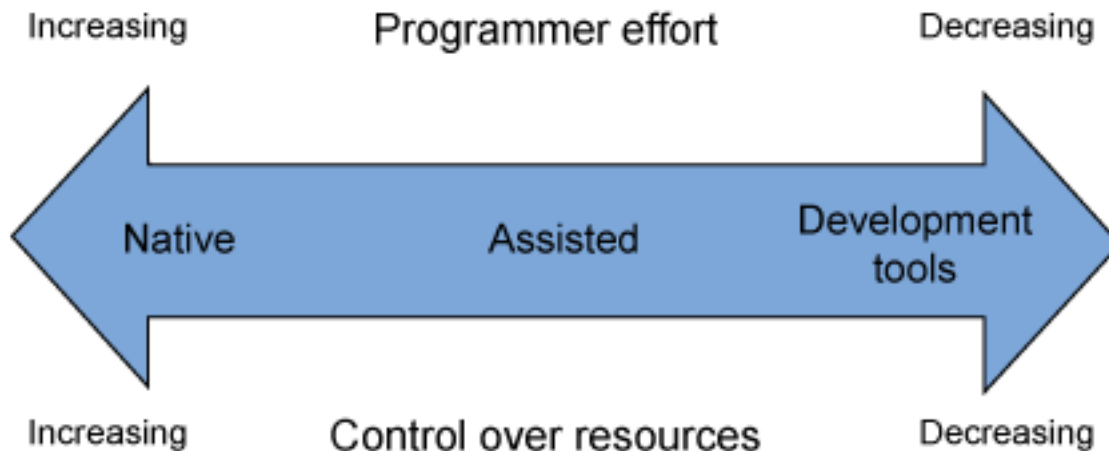
## Learning about programming approaches and how they are used

Now for the second point: What are the programming approaches for Cell/B.E., and how are they best used?

As shown in Figure 2, the approaches to Cell/B.E. programming are customizable and fall into the following three approaches:

- *Native* programming
- Assisted programming
- Development tools programming

**Figure 2. Work and control tradeoffs**



One thing to note about these programming approaches: As you move from the native approach through the assisted programming approach to the development tools programming approach, you cover another spectrum. At the native end, you have more granular control over Cell/B.E. resources and you do a lot more of the work. At the other end, you have the least amount of resource control, but the workload is much less.

The three different approaches are not mutually exclusive. As you try them out, you start to see that they overlap in functionality. The next sections describe the advantages of each approach, their limitations, and in which programming situations you might find each approach useful. The productivity that can be achieved with

each approach is based on the skill level of the individual programmer. The last section provides a list of tools that IBM, third-party vendors, and open source groups provide that are available for each approach.

## **Native programming**

The native programming approach involves working with hardware resources, intrinsics, and data movement to specifically control your application. This flexibility is not available in other programming frameworks and multicore systems. This flexibility is included in the Cell/B.E. architecture in order to enable you to decide how to allocate and use resources such as memory within your application.

### **The advantages**

The native programming approach provides the following advantages:

- You generally get the best, most highly tuned performance possible.
- You make the best use of the native resources.

### **The limitations**

The native programming approach also comes with some limiting factors, including:

- You will do the most amount of your own coding using this approach.
- You might have a learning curve because you need to learn about the native resources that you will use.

### **The situation**

When would you use the native programming approach?

- For embedded hardware applications
- For real-time operating system uses
- When performance is the most important measure and you cannot achieve your targets with the other approaches
- Any situation in which resources, power, space, or cost is at a premium

## **Assisted programming**

Not all programmers want to work directly with the hardware resources, so the assisted programming approach is available to use libraries and frameworks for your Cell/B.E. programming.

## The advantages

The assisted programming approach provides the following advantages:

- Reduced development time over the native method
- You still have the opportunity to use some of the native resources

## The limitations

The assisted programming approach also comes with some limiting factors, including:

- Performance gains generally won't be as great as with the native approach
- You are confined to the limitations of the frameworks and libraries you choose

## The situation

When would you use the assisted programming approach?

- For the vast majority of just about any application you can think of
- This is the typical approach to programming to which most people are accustomed

## Programming with development tools

As the highest-level approach, the development tool programming approach provides the most abstraction from the hardware resources. This approach includes working with development tools and environments.

## The advantages

The development tool programming approach provides the following advantages:

- Can significantly reduce the development time required
- Platform independence might be easier to achieve because many tools already incorporate it

## The limitations

The development tools programming approach also comes with some limiting factors, including:

- You might see even fewer performance gains than with the other approaches
- The CASE tools predetermine the debugging capabilities and platform support choices available to you

### **The situation**

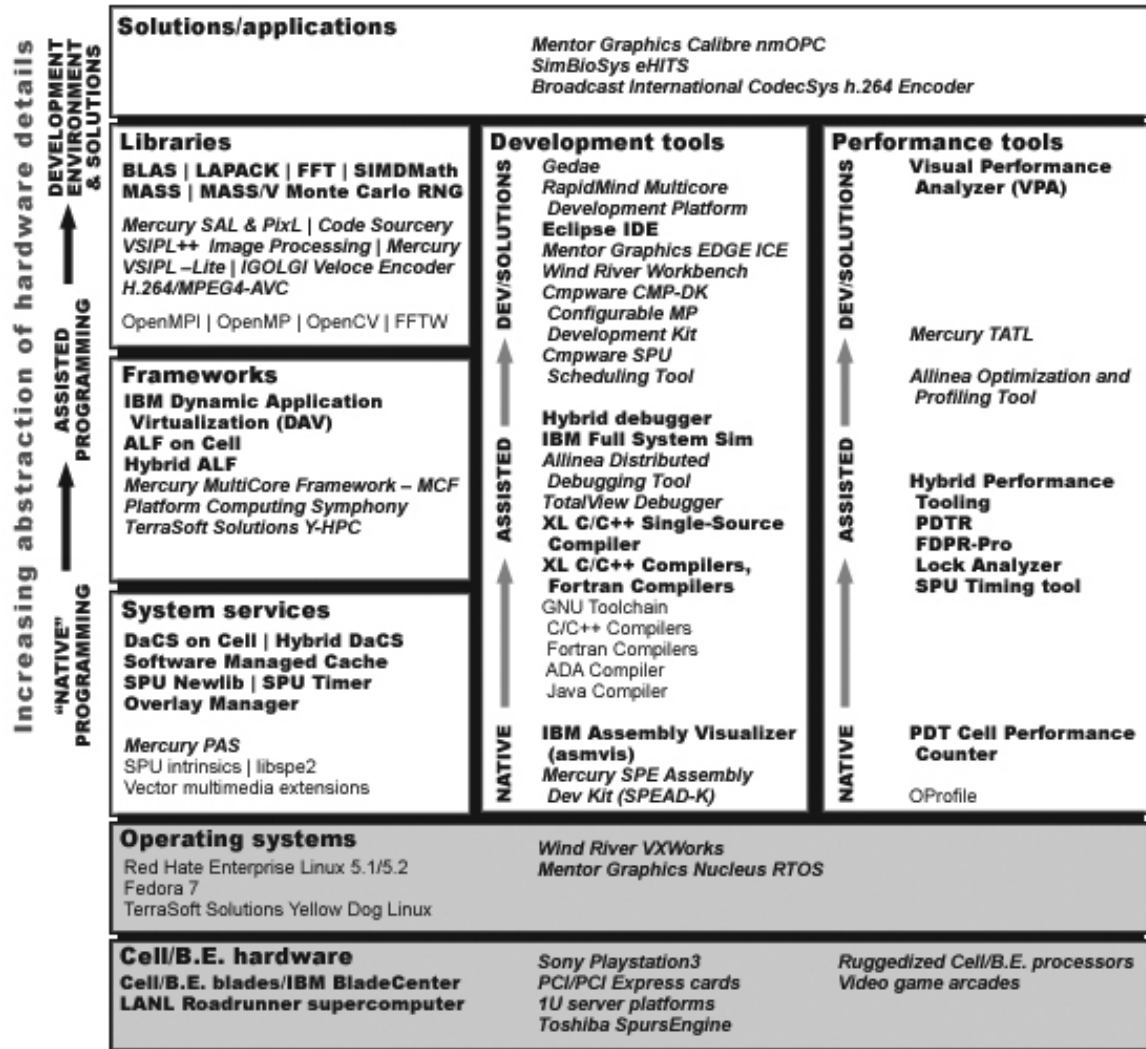
When would you use the development tools programming approach?

- For applications that are to be simultaneously deployed across multiple platforms and hardware architectures
- If you are more accustomed to working with higher-level tools than with working directly with the hardware resources

## **Determining the software and tools to support the approaches**

Figure 3 shows various software, hardware, and tools available for Cell/B.E. across the three programming approaches.

### **Figure 3. Cell/B.E. software and tools**



- **IBM offerings in heavy bold**
- ***Third-party offerings in bold italic (IBM may not offer support)***
- **Open source offerings in regular text (IBM may not offer support)**

(The most current version of this chart is available as a PDF in the [Downloads](#) section of this article.)

The IBM offerings are denoted with heavy bold text. Third-party offerings are denoted with bold italic text. Open source offerings are denoted with regular text. At the bottom of the chart, the foundation layers include the hardware and the operating systems. Cell/B.E. hardware is mature, developed, and diverse in the form factors that are available. Many of the operating systems Cell/B.E. excels on are mature, stable, well-entrenched platforms.

Above the foundation layers, the chart indicates software and tools available for **Native programming**, then **Assisted programming**, then **Development environments or application solutions** (spectrum on the left side of the chart). This spectrum is repeated within the **Development tools** and **Performance tools** boxes. The tools near the bottom of these boxes are more Native-oriented, and the tools progress through the Assisted stage into the Development environment tools.

One more important thing to note in this chart: Not only does IBM support the Cell/B.E. ecosystem, there is a robust and diverse (and growing!) third-party and open source, commercial- and community-based, community around the Cell Broadband Engine Architecture™.

## Conclusion

This article covered how to leverage your existing skills for Cell/B.E. programming, discussed the three programming approaches, and illustrated the various tools available for Cell/B.E. So, how can you get started with Cell/B.E. programming now? Here is a quick checklist to get started with Cell/B.E. programming.

### Getting started with Cell/B.E. programming

1. Visit the Cell Broadband Engine resource center on IBM developerWorks to [download the latest IBM SDK for Multicore Acceleration](#).
2. Install the IBM SDK for Multicore Acceleration
  - on your Linux workstation using the [SDK 3.0 Installation Guide](#).
  - on your Windows® workstation using [VMWare instructions from Georgia Tech](#).
3. Start programming for Cell/B.E. systems using the [Cell/B.E. IDE tutorial](#) and the [IBM Full System Simulator](#).
4. Explore and submit questions to the [Cell/B.E. discussion forum](#) on developerWorks. The forum is heavily trafficked, and it is the fastest way to get expert answers.
5. Purchase the Cell/B.E. hardware offering that fits your needs. [Contact IBM about Cell Broadband Engine](#). You can even get started with a Sony Playstation 3 for your development.

### Acknowledgments

Thank you to the IBM/Cell/B.E. team that helped in formulating this article and

reviewing it.

## Downloads

| Description                                    | Name               | Size | Download method      |
|--|--------------------|------|----------------------|
| PDF version of the tools chart, easier to read | sdktools091809.pdf | 94KB | <a href="#">HTTP</a> |

[Information about download methods](#)

# Resources

## Learn

- Use an [RSS feed](#) to request notification for the upcoming articles in this series. (Find out more about [RSS feeds of developerWorks content](#).)
- Visit the [Cell Broadband Engine Education and Training wiki](#).
- Learn more about Cell/B.E. programming from the developerWorks series:
  - ["Programming high-performance applications on the Cell/B.E. processor"](#)
  - ["PS3 fab-to-lab"](#)
  - ["The little broadband engine that could"](#)
- Refer to the [Cell Broadband Engine documentation](#) section of the IBM Semiconductor Solutions Technical Library for a wealth of downloadable manuals, specifications, and more.
- Sign up for the [developerWorks newsletter](#) and get the latest developer news and Cell/B.E. happenings delivered to your inbox each week. Check *Power Architecture*® when you sign up to receive Cell/B.E. news in your newsletter.

## Get products and technologies

- Get your copy of the [IBM SDK for Multicore Acceleration 3.0](#) or browse through the [library of Cell/B.E. documentation](#).
- Find all Cell/B.E.-related articles, discussion forums, downloads, and more at the IBM developerWorks [Cell Broadband Engine resource center](#): your definitive resource for all things Cell/B.E.
- Contact IBM about [custom Cell/B.E.-based or custom-processor based solutions](#).
- Explore additional cutting edge Cell/B.E tools at [IBM alphaWorks](#).

## Discuss

- [Participate in the discussion forum for this content](#).
- Check out the [Cell Broadband Engine Architecture forum](#) to get your technical questions about the processor answered. Juicy problems and answers from the forums are rounded up periodically and highlighted in the ["Forum watch" blog series](#).
- Go to the [Cell Broadband Engine/Power Architecture blog](#) for *news*, downloads, instructional resources, and event notifications for Cell/B.E. and other Power Architecture-related technologies. You can find the popular ["Forum watch"](#) blog

series (Q&A roundup), the "FixIt" technology updates, and the [Infobomb](#) quick-read technology introductions.

## About the author

Anita Bateman

Anita Bateman is a Senior IT Architect with the IBM Corporation. She has been with IBM doing software development and architecture since 1998. She is a certified architect with both IBM and The Open Group, and she has filed and published several patents. She holds an M.S. in computer science from the University of Texas at Austin and a B.S. in computer science from Hope College in Holland, Michigan. She is currently a Cell Broadband Engine solutions architect, working with partners and customers to adopt the IBM multi-core technology and to improve Cell/B.E. programmability.

## Trademarks

IBM and developerWorks are trademarks of IBM Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Cell Broadband Engine is a trademark of Sony Computer Entertainment Inc.

Other company, product, or service names may be trademarks or service marks of others.