Cell BE Software Aspects

Roland Seiffert
IBM Linux Technology Center

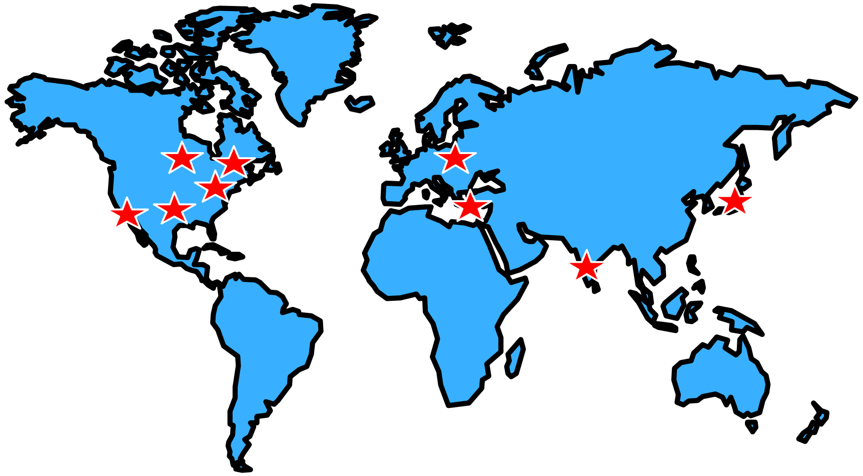
Agenda

1. Cell BE overview – from a SW perspective
2. Linux for Cell BE
3. Programming for Cell BE – an example

Cell BE – Overview

Cell BE History

- IBM, SCEI/Sony, Toshiba Alliance formed in 2000
- S/T/I Design Center opened in March 2001 in Austin, TX
- Hardware designed in parallel with software, Linux
- **Linux used for bringup / test throughout dev't cycle**
- February 7, 2005: First external technical disclosures on Cell BE
- **April 26, 2005: First Linux patches for Cell BE disclosed**
- May 2005: IBM Cell BE-blade prototype running Linux demonstrated at E3
- August 25, 2005: Release of technical documentation
- **September 2005: Linux kernel 2.6.13 enables Cell BE platform**
- November 9, 2005: SDK 1.0 Released
- **March 20, 2006: Linux kernel 2.6.16 released with official support for Cell BE**
- November 11, 2006: Playstation 3 availability with Linux support

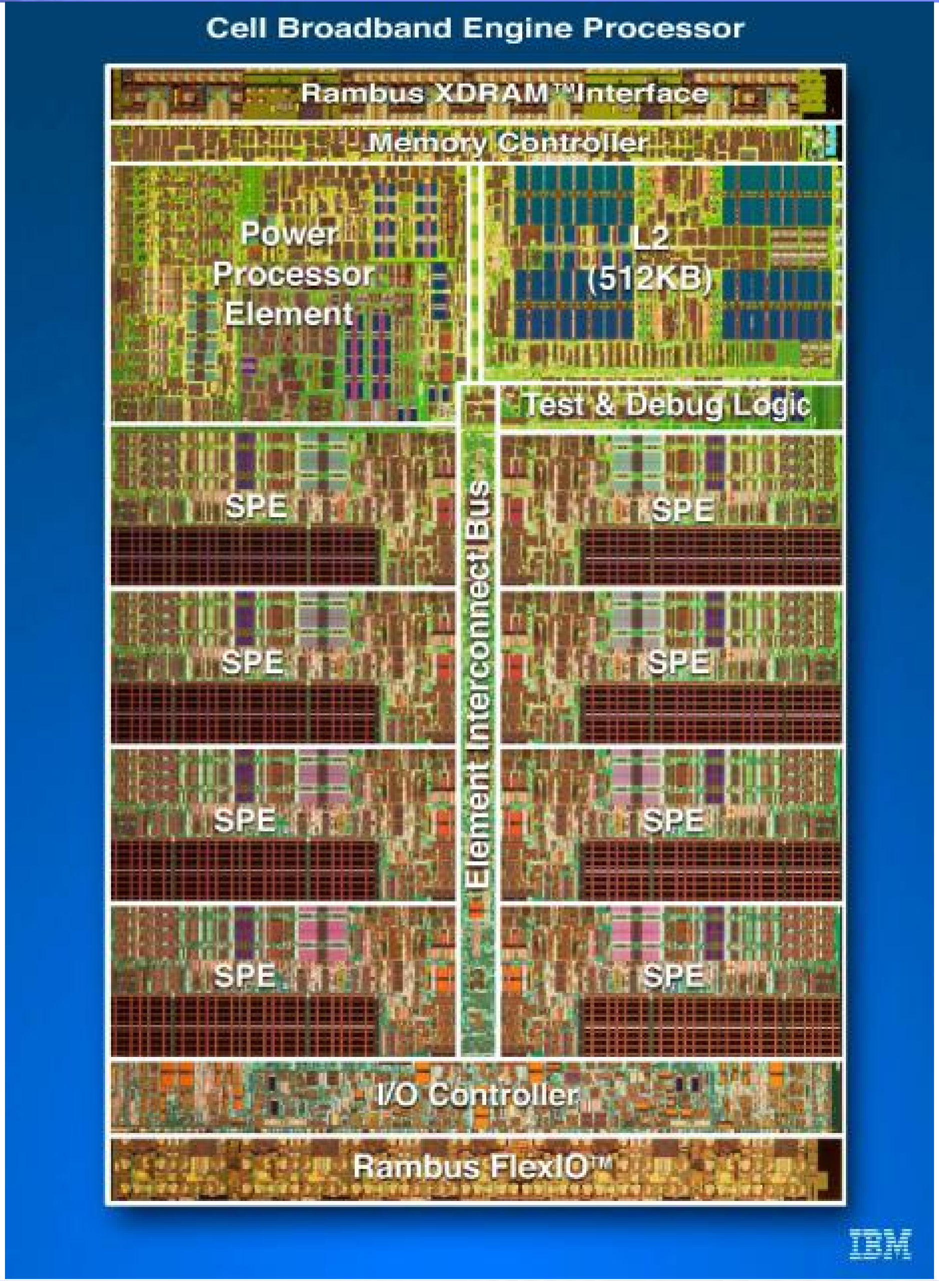


Cell BE Processor

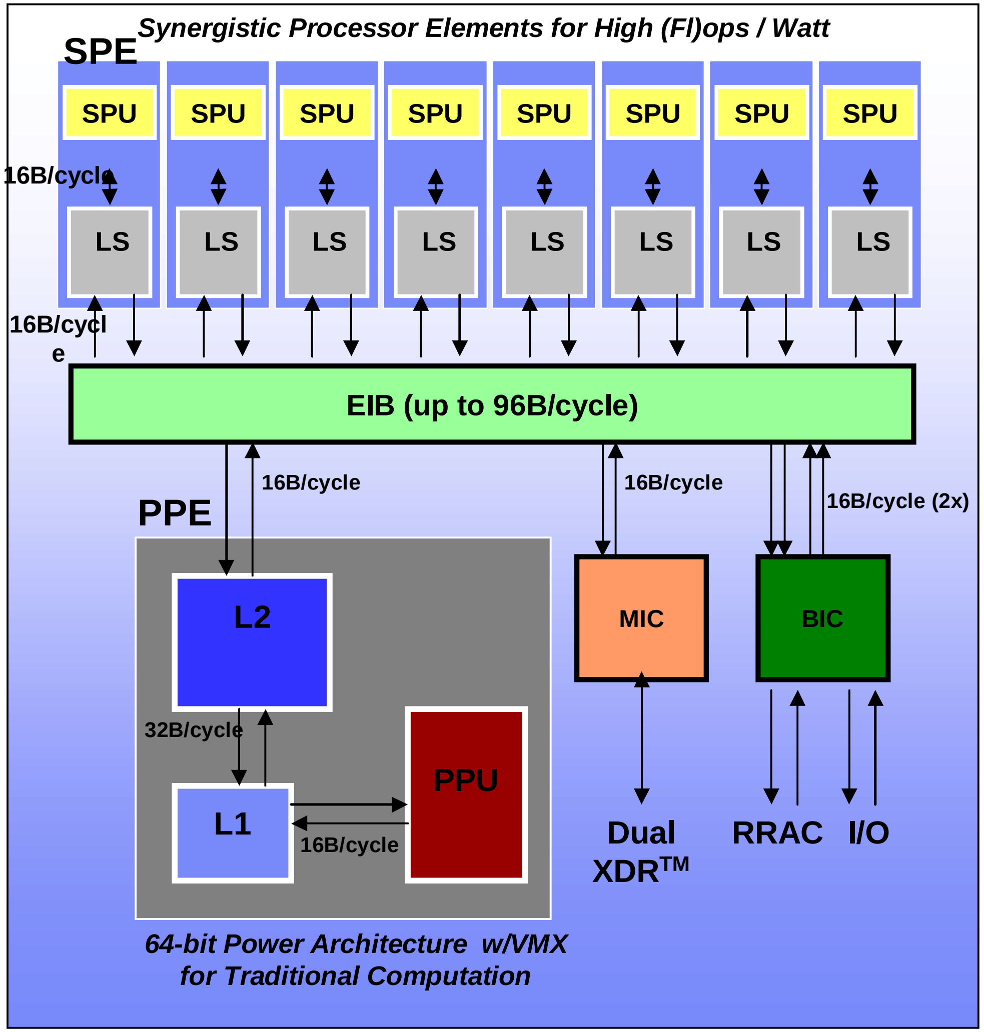
- ~241M transistors
- ~235mm2
- Top frequency in lab >4GHz
- 9 cores, 10 threads
- > 256 GFlops (SP) @4GHz
- > 26 GFlops (DP) @4GHz
- Up to 25.6GB/s memory B/W
- Up to 75 GB/s I/O B/W

Heterogeneous multicore architecture

- 1 Power Processor Element
  - Control tasks
- 8 Synergistic Processor Elements
  - Compute-/Data-intensive tasks

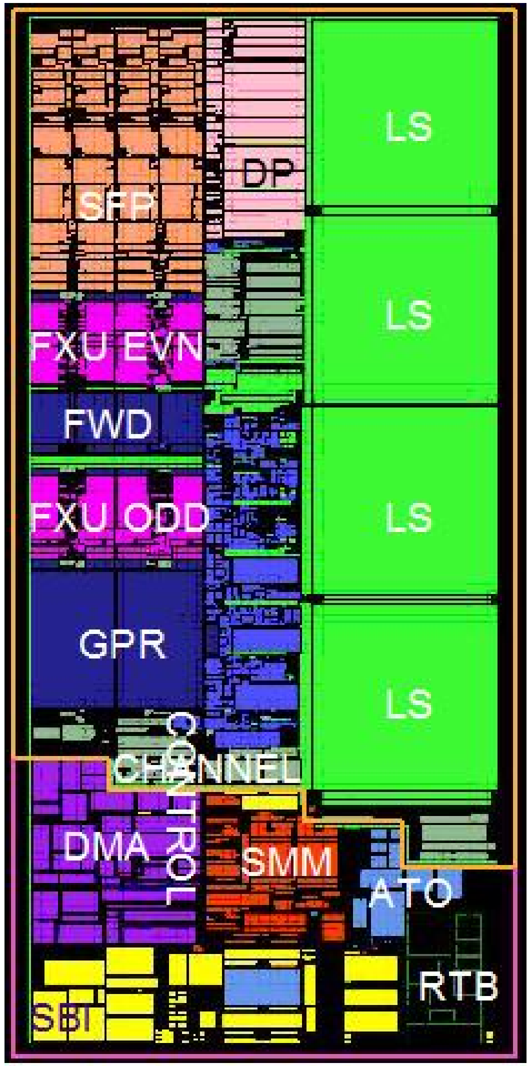


Cell BE - A Multi-Core System-on-Chip



- Power Processor Element
  - Control tasks
- Synergistic Processor Element
  - Data-intensive tasks
- Memory Interface Controller
  - Rambus XDR memory
- Bus Interface Controller
  - Rambus FlexIO
- Element Interconnect Bus
  - Data movement

SPE Highlights



- User-mode (application) architecture
  - No translation/protection within SPU
- 256 KB local store
  - Combined I & D (not a cache!)
- SIMD dataflow
  - Graphics SP-Float
  - IEEE DP-Float
  - Rich set of integer operations
- Unified Register File
  - 128 entry x 128 bit
  - No register renaming
- Direct Program Control
  - DMA, list DMA
  - Branch hint

The Cell BE Processor and Architecture is a Breakthrough Architectural Innovation in Chip Design

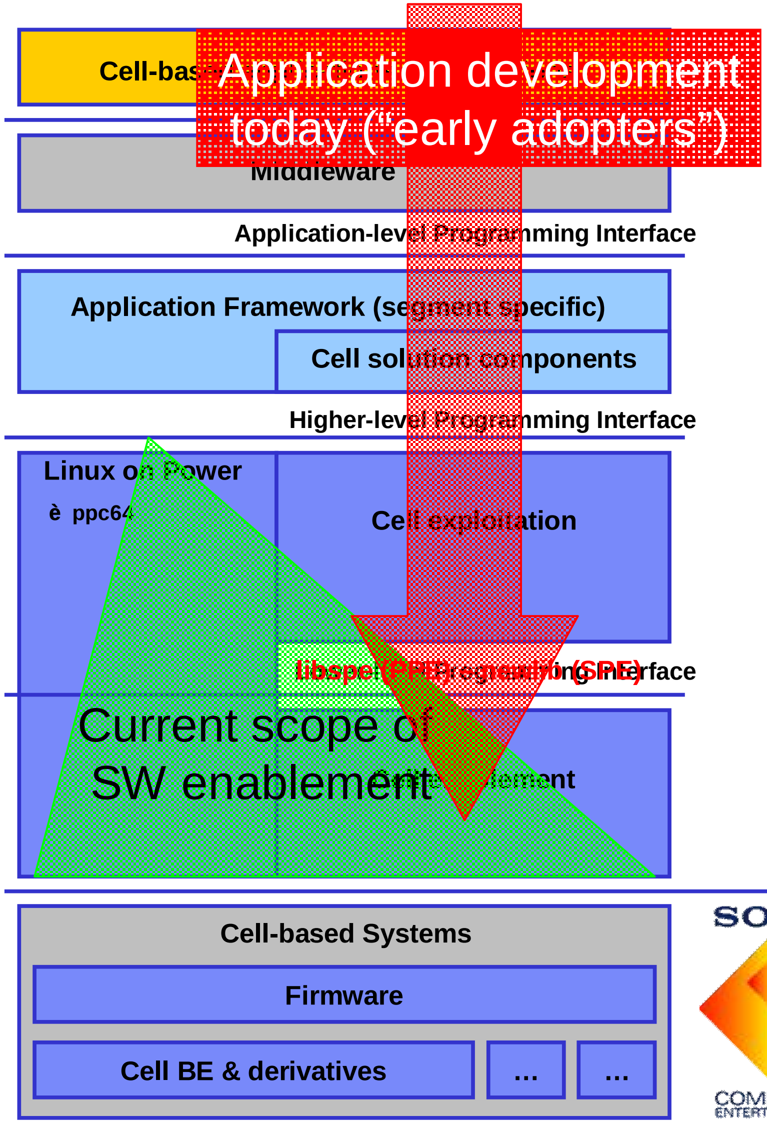
OK. Now what?

What is it good for?
How can I get it to do that?
Am I on my own?

The Cell BE Processor and Architecture is a Breakthrough Architectural Innovation in Chip Design

- ***SW Challenge #1*****:** The breakthrough capabilities of Cell BE must be made fully available to application developers
  - Standardized (low-level) APIs, libraries, compilers, debuggers, …
- ***SW Challenge #2*****:** New programming models and corresponding high-level APIs are required to allow for easy exploitation of the Cell BE capabilities
  - Open community collaboration in Research & Development to drive Cell BE exploitation – build new Cell-focused communities
- ***SW Challenge #3*****:** Keep existing Linux environment standard, while enabling breakthrough exploitation by applications
  - Leverage the existing Linux and Linux on POWER ecosystems as a base for the Cell BE operating environment
- ***SW Challenge #4*****:** Rapidly enable new communities and end-to-end solutions based on Cell BE systems
  - Interweave the existing and new communities into a “Cell-society” that embraces both standards and envelope-pushing
  - (Initial) focus on application segments with well-understood, high “Cell affinity” to create a success story and further enable community expansion

Cell BE Software Platform


Cell-bas
Application development today (“early adopters”)
Middleware
Application-level Programming Interface
Application Framework (segment specific)
Cell solution components
Higher-level Programming Interface
Linux on Power
è ppc64
Cell exploitation
libspe (PPE) newlib (SPE)
ing Interface
Current scope of SW enablement
Cell enablement
Cell-based Systems
Firmware
Cell BE & derivatives
…
…

Commercial and Open Source Exploitation of Cell

Provide standard application platforms for Cell

- Middleware and frameworks provide architecture-specific components and hide Cell –specifics from application developer

Make Cell easier to program

- Hide complexity in critical libraries
- Compiler support for standard tasks, e.g., overlays, global data access, SW-managed cache, auto vectorization, auto parallelization, …
- Smart tooling

Provide access to full Cell capabilities

- Reflects the “exotic platform” and is hard to program
- Challenging to exploit, e.g., SPE’s Limited local memory (256 KB) – need to DMA data and code fragments back and forth; Multi-level parallelism – 8 SPEs, 128-bit wide; SIMD units in each SPE; …

SONY COMPUTER ENTERTAINMENT ®

MERCURY Computer Systems, Inc. Challenges Drive Innovation

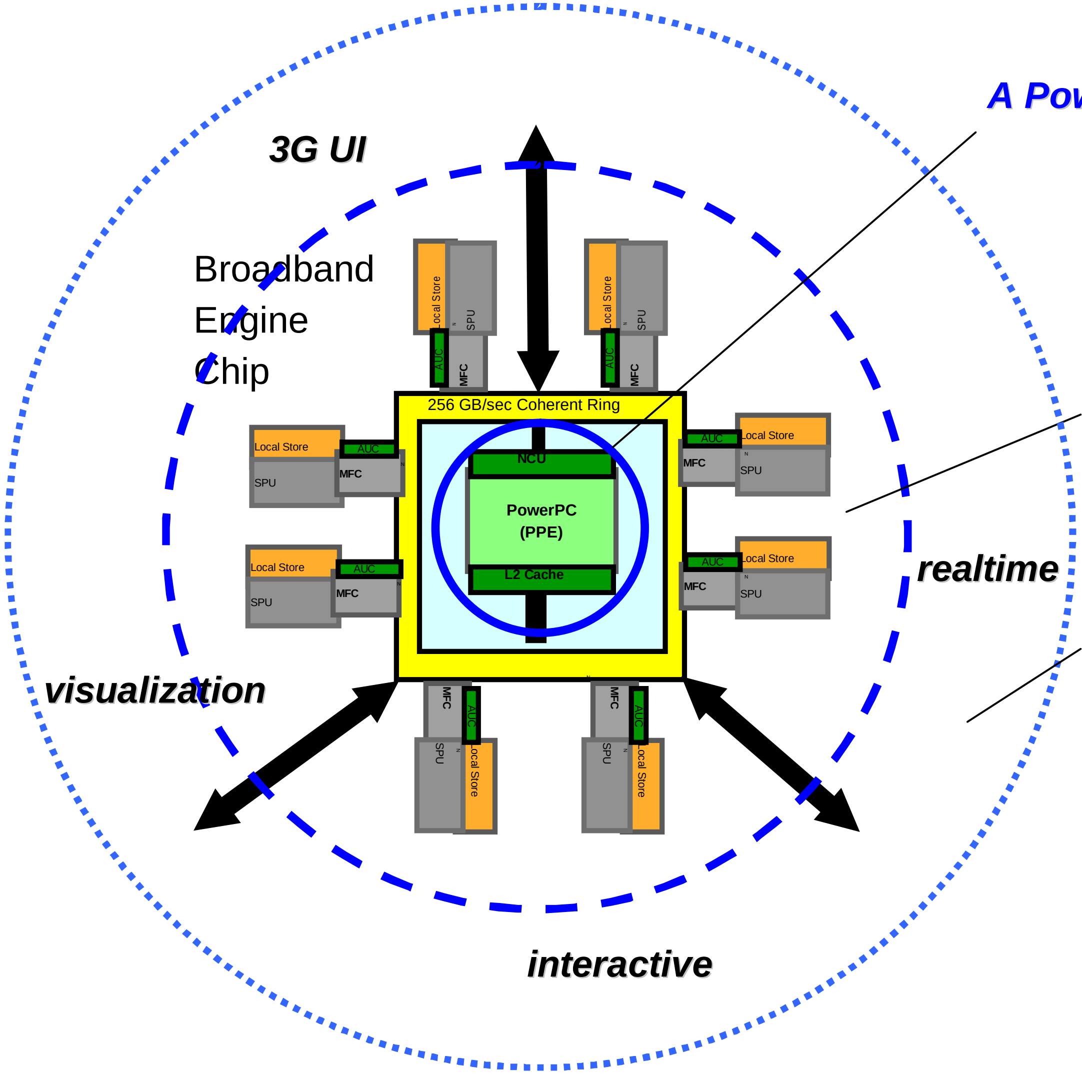
TOSHIBA

Others may appear soon…

IBM ®

Linux for Cell BE

Cell BE from a Software Perspective



A PowerPC processor

- ***known architecture***
- ***known programming model***
- ***known SW stack/tool chain***
- ***supported by Linux on Power ecosystem***

with

- ***breakthrough new capabilities***
- ***radically new application structure***
- ***new programming models***
- ***accessible via support in Linux***

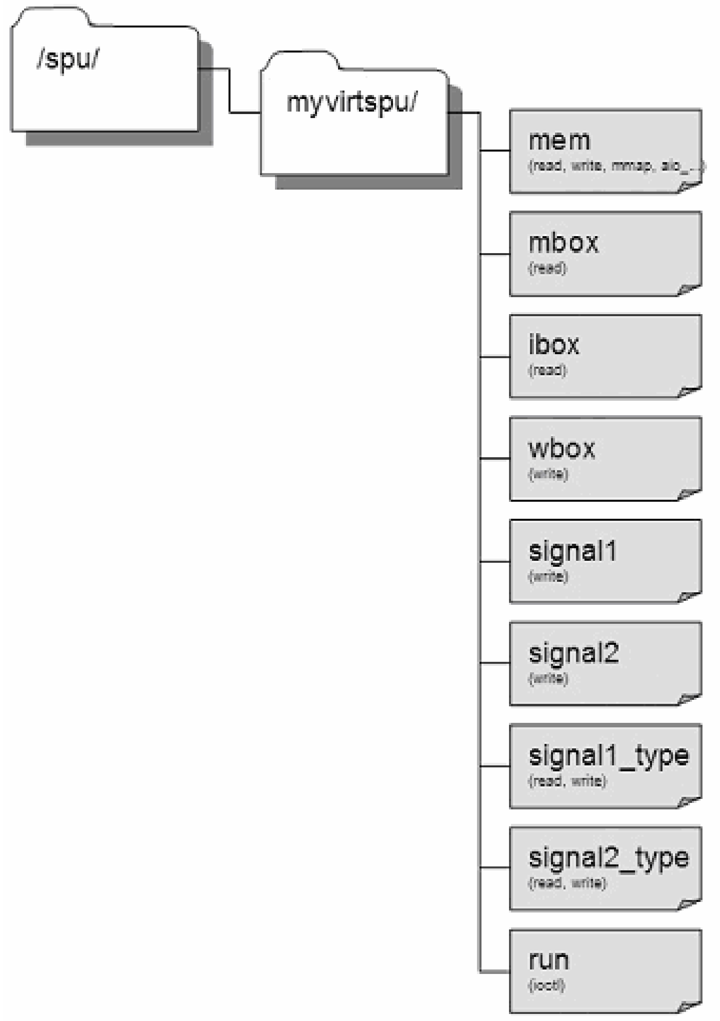
Everything available for PowerPC just works… That's a great start!

True exploitation of the Cell BE performance potential may still be a significant challenge.

Leveraging Linux on Power

- PPE is 100% compliant with the PowerPC Architecture
  - Only minimal effort needed to make Linux on Power run on CBE
  - All Linux on Power applications run on CBE
  - BUT: just on PPE – no automatic usage of SPEs
- PowerPC architecture in Linux allows for „platforms“ (pSeries, iSeries, PowerMac, …) to encapsulate specialities of the various systems based on the Power architecture
  - Share all of the Linux common code and most of the Power architecture-specific code, but can be easily extended w/o interference with existing platforms
  - Added new „Cell“ platform to provide CBE special code, e.g. /spufs to enable usage of SPEs – and made commitment to maintain it, i.e., provide a skilled maintainer
  - Active, focussed, open-minded and highly skilled PowerPC Linux community
- Rapid upstream acceptance of CBE patches
  - First patches integrated in 2.6.13
  - „Cell“ officially supported platform in 2.6.16
  - Distros pick up Cell easily and rapidly, e.g. Fedora Core 5 has Cell support
- Linux OS very stable from the beginning – even though CBE is a radically different chip design

Integrating SPEs: the /spufs Virtual Filesystem



q Virtual File Systems are an established method in Linux to provide access to new HW features in a standardized way w/o introducing new, achitecture-specific system calls

q A (virtual) SPE context is represented by a directory with all user-accessible SPE components being represented as files in the directory.

q Usage of SPE components by using file I/O operations, e.g.,

   q access to local store using read/write or mmap and direct access

   q mailbox communication to/from SPE using read/write or mmap and direct access

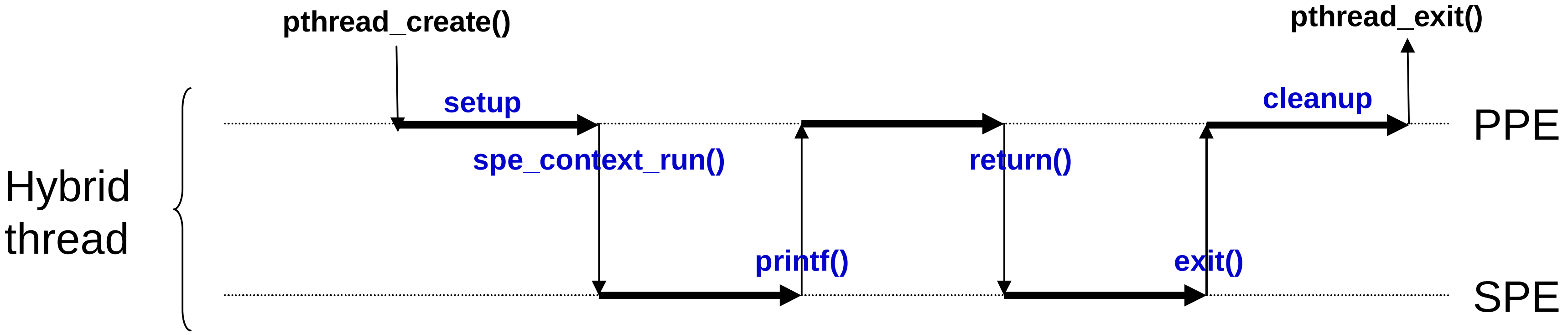
q A specific SPE scheduler puts virtual SPE contexts onto physical SPEs for actual execution according to priority and scheduling policy.

libSPE2 - Hybrid Threads

In SDK 2.1 IBM will move to libspe2 and we will deprecate libspe1

- PPE provides the “infrastructure”
  - OS kernel, device drivers,…
  - Overall application logic (“orchestration”)
- SPE provides the “compute power”
  - Accelerators – for application and/or OS functions

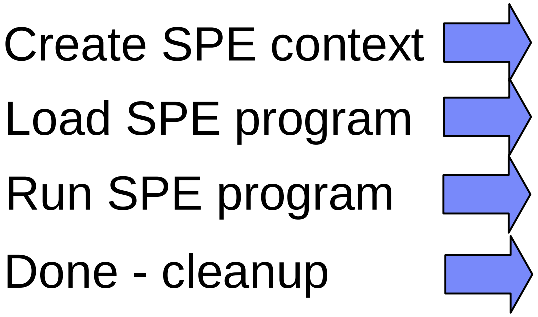
- Key design: the “hybrid thread”
  - A (regular) OS thread started on the PPE that may use one (or more) SPEs
  - Execution flip-flops between PPE and SPE as needed, e.g., setup code on PPE, then computation on SPE, execution of library/system calls on PPE, more computation on SPE, …



Notes:

1) **Multi-threaded (parallel) applications use multiple hybrid threads to use multiple SPEs**
2) **A single hybrid thread may manage multiple SPE contexts – but only one can be running at any given point in time**

Example: Run the simple SPE program “hello”


Create SPE context

Load SPE program

Run SPE program

Done - cleanup

```
#include <stdlib.h>
#include <libspe2.h>
int main()
{
   spe_context_ptr_t spe;
   unsigned int createflags = 0;
   unsigned int runflags = 0;
   unsigned int entry = SPE_DEFAULT_ENTRY;
   void * argp = NULL;
   void * envp = NULL;
   spe_program_handle_t * program;
   spe = spe_context_create(createflags, NULL);
   program = spe_image_open("hello");
   spe_program_load(spe, program);
   spe_context_run(spe, &entry, runflags, argp, envp, NULL);
   spe_image_close(program);
   spe_context_destroy(spe);
}
```

“3C Common Linux”

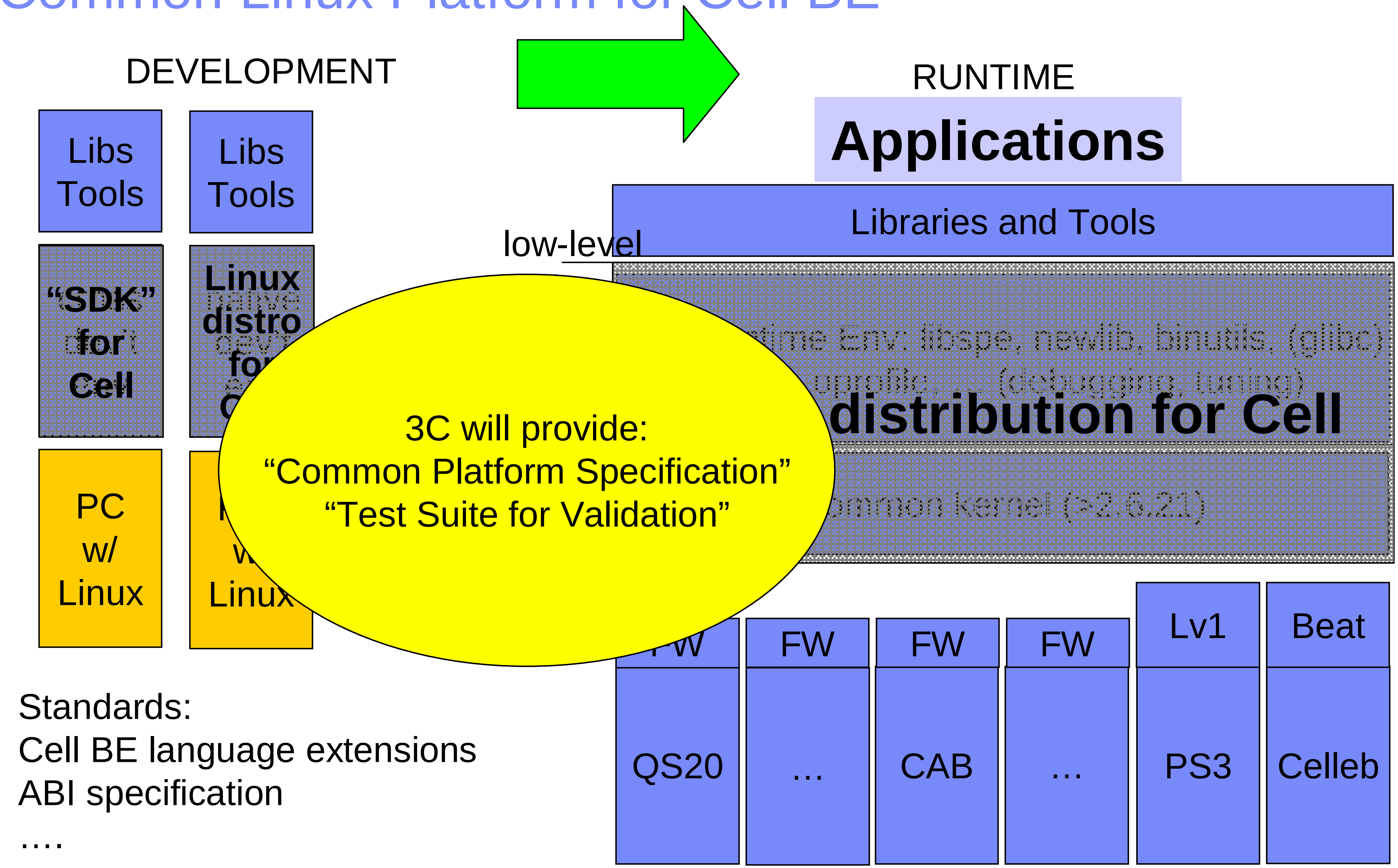
S/T/I Cell BE Open Source Cooperation

- Sony/SCE, Toshiba and IBM share a common vision on Cell BE Architecture and its potential in many application areas
- S/T/I agreed to cooperate closely to enable an active ecosystem for a broad usage of Cell BE systems

- The goal is to provide a single “Common Linux” for all systems using the Cell BE processor – including IBM’s QS20, Mercury’s Cell-based Blade, Toshiba’s Reference Set, SCE’s PlayStation 3, and all others to come.
- The commitment is to develop this platform as part of the Open Source communities and achieve mainstream integration of the new Cell BE platform.

- This is *not* an exclusive club. We…
  - …work with and within the existing communities wherever possible
  - …actively encourage participation by others in these efforts

Common Linux Platform for Cell BE



DEVELOPMENT

Libs Tools

Libs Tools

"SDK" for Cell

Linux distro for C

PC w/ Linux

w/ Linux

RUNTIME

Applications

Libraries and Tools

low-level

time Env: libspe, newlib, binutils, (glibc)

oprofile, ... (debugging, tuning)

distribution for Cell

ommon kernel (>2.6.21)

FW | FW | FW | FW | Lv1 | Beat

QS20 | … | CAB | … | PS3 | Celleb

3C will provide:
"Common Platform Specification"
"Test Suite for Validation"

Standards:
Cell BE language extensions
ABI specification
….

S/T/I Joint Open Source Activities & Status

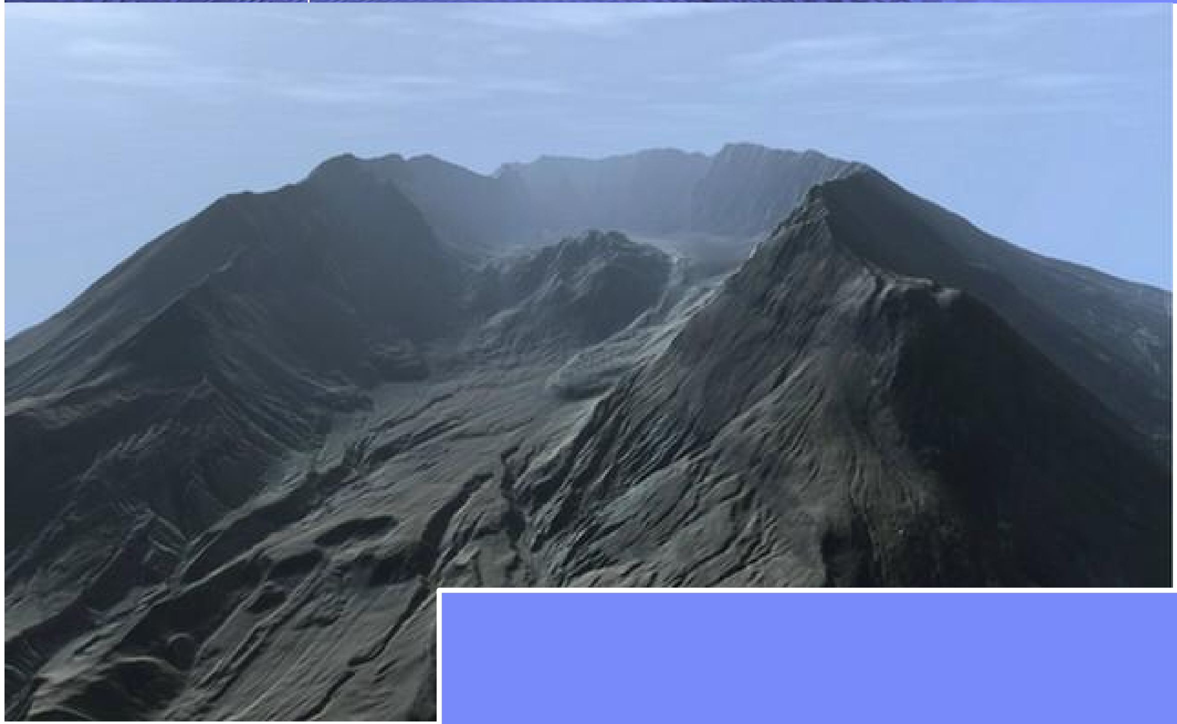
- Linux kernel – new platform in ppc64
  - 2.6.16 – first official Cell BE support, SPUFS; 2.6.16+patches – initial PS3 support; 2.6.18+patches – current QS20 support
  - Plans: 2.6.20 – initial PS3 / TRS support, QS20 support; 2.6.21 – full PS3 / TRS support
  - Major contributors: IBM, SCE/Sony, Toshiba
- Binutils – SPU support, for PPU added "-mcell" to ppc
  - Part of 2.18
  - Major contributors: SCE/Sony, IBM
- GCC – SPU support, PPU optimizations
  - Currently available as 4.1.1-based package – not in mainline yet, but donated to FSF
  - Plans: 4.3
  - Major contributors: SCE/Sony, IBM
- GDB – SPU debugger, combined PPU/SPU debugger
  - gdb 6.6 has SPU debugger; combined debugger currently available as part of IBM SDK 2.0
  - Major contributors: IBM, Toshiba
- Newlib – C base library for SPE
  - Part of 1.15
  - Major contributors: IBM, SCE/Sony
- Libspe – a SPE runtime management library
  - Available on kernel.org in ~arnd and discussed/developed on cbe-oss-dev
  - Plan to move to public repository
  - Major contributors: IBM, SCE/Sony
- SIMD math lib – a version of libm optimized for Cell BE SPUs using SIMD
  - Currently 2 versions (IBM, SCE/Sony) available as part of IBM SDK 2.0 and from Sony/SCE
  - Plan to unify and merge
  - Major contributors: SCE/Sony, IBM

TRS = Toshiba Reference Set (a CBE dev't system from Toshiba)

IBM Cell BE SDK 2.0 - Overview

- A complete Cell BE development environment that contains binaries and source code that are available for downloading from both IBM alphaWorks and Barcelona Supercomputing Center's Web site.
- IBM alphaWorks contains IBM-authored material, including
  - Library and Samples Source Code
  - IBM XL C/C++ Alpha Edition Compilers for Cell Broadband Engine Processor
  - IBM Full-System Simulator for the Cell Broadband Engine Processor
  - Eclipse-based Integrated Development Environment.
- Barcelona Supercomputing Center's Web site contains open-source projects that have been modified for Cell BE Processor, including
  - GNU GCC compilers for PPU and SPU, Linux Kernel 2.6.18, SPE Library support, NUMA support, and a system root image for the Full System Simulator.

- IBM Cell BE SDK Version 2.0 contains a number of significant enhancements over previous versions and completely replaces those versions. These enhancements include the following:
  - Linux kernel upgraded to 2.6.18
  - GNU GCC tools upgraded to Version 4.1.1 and XL C/C++ compiler to Version 0.8.1
  - Support added for a combined Power Processing Unit (PPU) and Synergistic Processing Unit (SPU) debugger
  - Addition of programming model frameworks, including SPU code overlays, an accelerator framework for offloading work to SPUs, and software managed cache
  - Addition of SIMD Math library for PPU and SPU; revamping of libm library for SPU; addition of MASS and MASS/V libraries for PPU
  - Simulator support for performance modeling of memory subsystem components and interactions
  - addition of Cell BE-specific, post-link code optimization tool
  - addition of Eclipse Integrated Development Environment (IDE) support for building, compiling, and debugging Cell BE applications. The IDE uses the underlying SDK tools, including compilers, debugger, and system simulator.



Questions?