CGGVeritas

Accelerators Experience

Jean-Yves BLANC

Guillaume THOMAS-COLLIGNON

CGGVERITAS

Agenda

- Our environment
- FPGA on Cray XD-1
- IBM Cell broadband engine
- GPGPU with nVidia CUDA

CGGVERITAS

Our environment

Software

- Hundreds of codes (modules) used in production
- A dozen codes are very CPU intensive
- Fortran & C, single precision, MPI / OpenMP

Hardware

- > 20,000 CPUs (cores) in Houston
- Intel Woodcrest, PowerPC 970 (2 x dual core)
- Apps typically run on 128 cores

CGGVERITAS

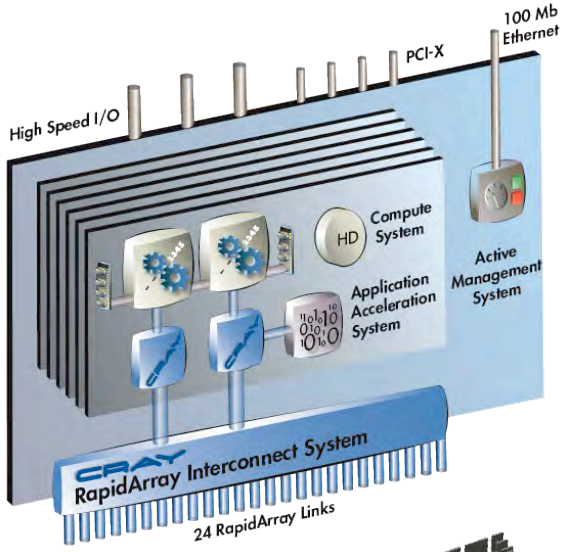
FPGA / Cray XD1

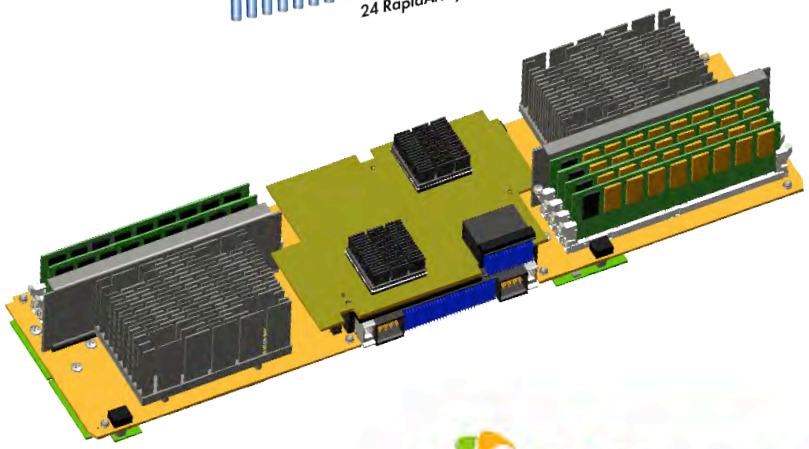
Tightly coupled cluster architecture

6 SMP nodes per chassis :

- 2 Opteron CPUs
- 1 Xiling Virtex II Pro FPGA

Proprietary high performance interconnect (up to 12 chassis)



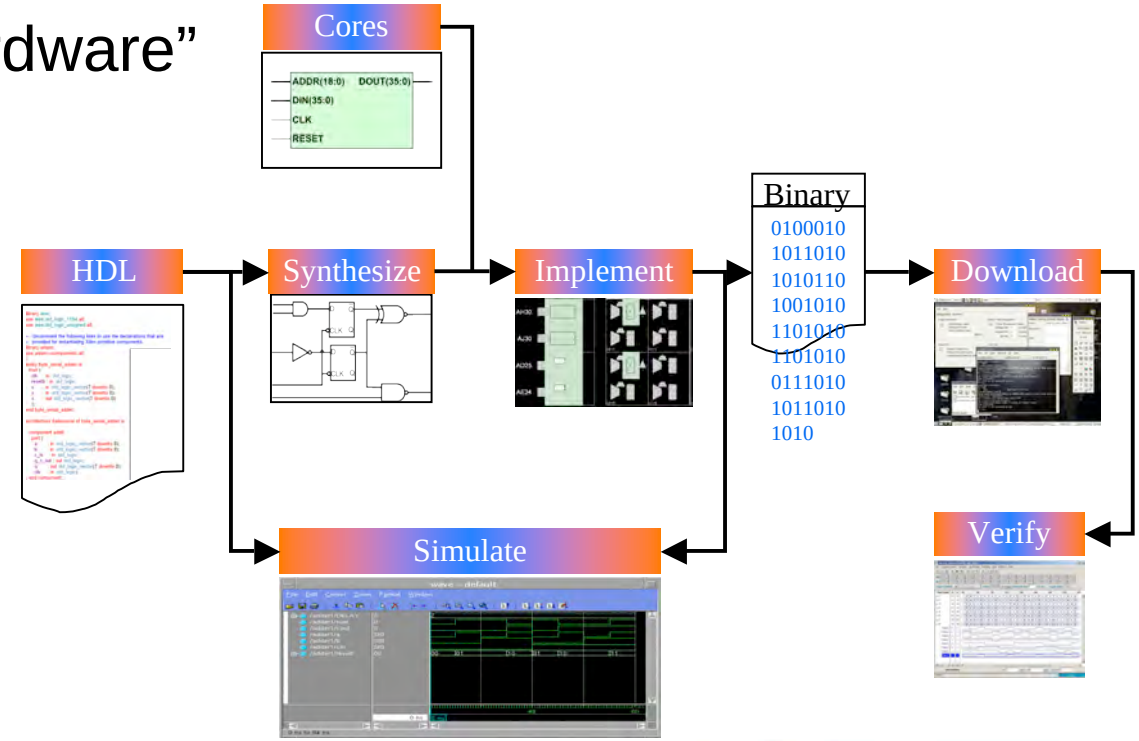


CGGVERITAS

FPGA Implementation

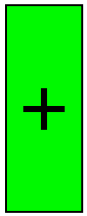
2D Convolution

- 32-bit IEEE floating point arithmetic library (ENS Lyon)
- Xilinx Software stack
- “Design your own hardware”



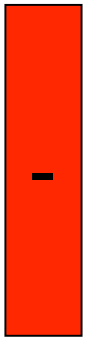
CGGVERITAS

FPGA Pros & Cons



Architecture, density

Just an accelerator on a regular system



Complex to develop, hard to maintain

Limited bandwidth

Efficiency

CGGVERITAS

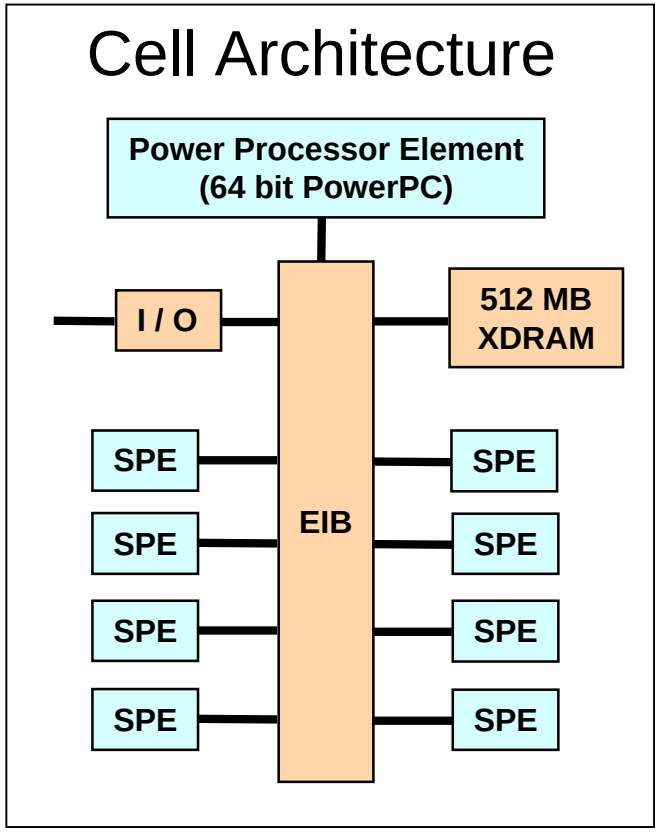
IBM Cell Broadband Engine

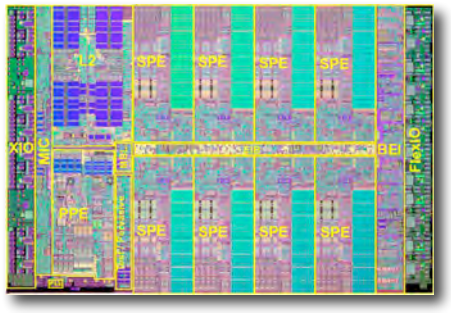
IBM QS 20 Blade

- 2 x 3.2 GHz Cell B.E.
- 205 Gflops per B.E.
- 1GB XDRAM

Cell floating point engine

- VMX-like SIMD instructions
- 256kB local store (Instructions + Data)
- EIB circular interconnect (204GB/s)
- 128 128-bit registers





CGGVERITAS

IBM Cell Porting process

Port code to Linux on Power

Rearrange the code to make use of the SPEs
- Function offloading
- Data streaming (DMA lists)

Write the SPE code
- Vectorize the scalar code
- Port from VMX / SSE intrisics to SPU intrinsics

Optimize for multiple SPEs

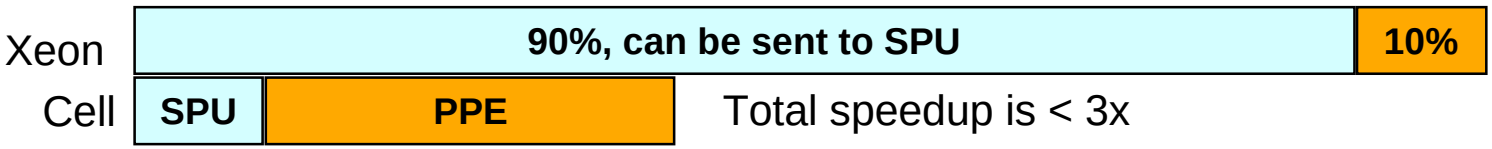
⇒The optimized code is not human-readable

CGGVERITAS

Performance on Cell

Compute kernels on SPUs : up to 20x faster

Code that runs on the PPE is ~3x slower



\+

High volume processor (PS3)

Performance on codes that fit

Memory bandwidth (25GB/s)

\-

Not a general purpose CPU

Complicated code

1 GB of memory was not enough

CGGVERITAS

GPGPU : NVIDIA/CUDA

- Volume technology with aggressive roadmap
- nVidia significantly ahead
- Tight partnership with nVidia

CUDA

- Data parallel SIMD
- C language with easy to learn extensions
- Write the code for 1 thread, spawn thousands

CGGVERITAS

nVidia Hardware

G80 GPU family

1.5 GB memory, Bandwidth = 80GB/s

8 multiprocessors x 16 scalar processing units = 128 CPUs

16 KB shared memory per multiprocessor

8192 registers per multiprocessor

~ 300 Gflops peak

PCI Express bus = 3.2 GB/s

CGGVERITAS

CUDA performance strategies

PCI Express is a bottleneck

- Works only for compute intensive applications.
- Leave the data in the GPU, run several kernels

Memory access constraints to get performance

- Like managing the CPU cache yourself

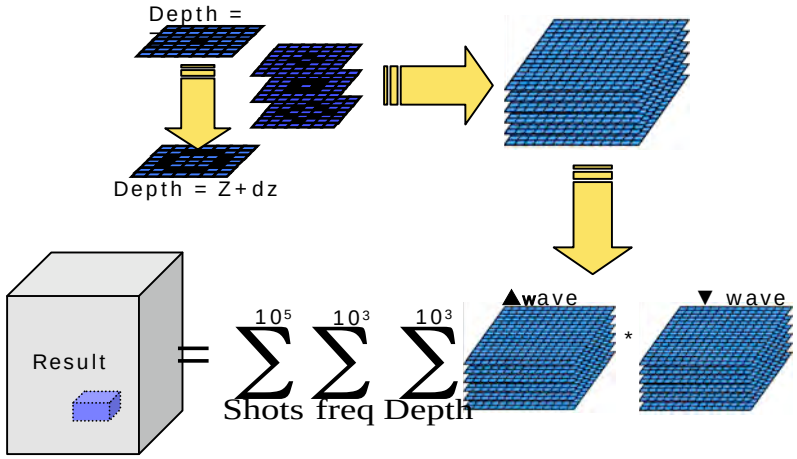
Unique memory features

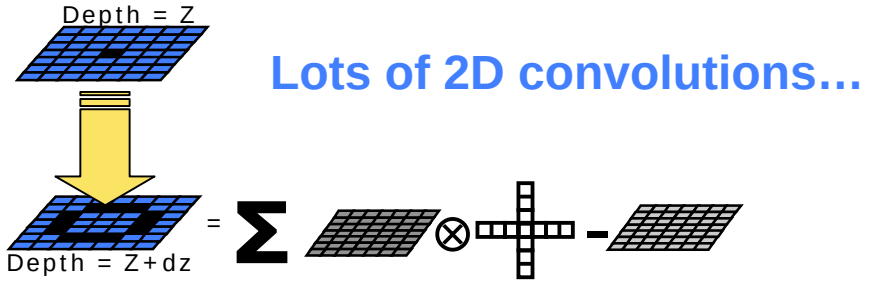
- 2D Texture memory, float indices, boundary conditions

CUDA perf library (BLAS, FFTs)

CGGVERITAS

Wave Equation Example

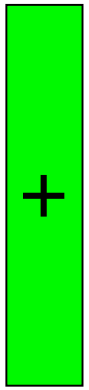




- Deep re-engineering of the code was needed
- Several GPU kernels applied to the data
- Over 15x performance can be achieved
- Performance depends on the dataset

CGGVERITAS

nVidia CUDA Pros & Cons

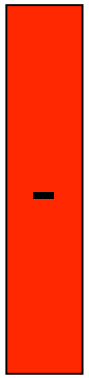


Performance

Easy to learn, code still looks like C

Just an add-on to regular systems

Fast evolution of CUDA



Tricky to get full performance

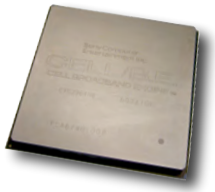
Only for very intensive computations

Limited amount of memory

Limited debugging & profiling

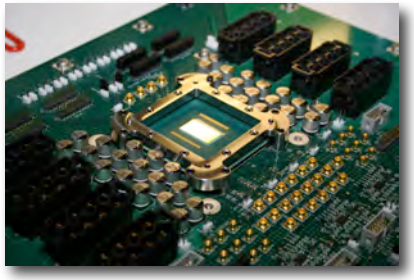
CGGVERITAS

Conclusions



Programming issues

- Specific programming models
- Algorithms must be modified
- Must understand the hardware
- Too many codes to port



Performance aspects

- Not always rewarding
- CUDA is our best bet so far
- CPUs are getting quicker too !
- Future : more parallelism

CGGVERITAS