TOTALVIEW TECHNOLOGIES

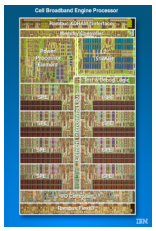
Debugging CUDA with TotalView

Lessons learned in porting TotalView to the Cell

Cell-Based Clusters

- LANL RoadRunner
- Three others on top 500 (LANL, IBM, U Warsaw)
- Barcelona Mare Nostrum cell prototype
- Often have mixed or hierarchical node architecture
  - Tri-blades

Cell Processor Architecture



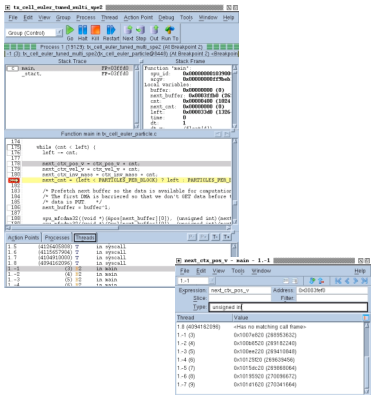
- Mixed architecture
  - One power processor
  - Eight synergistic processing units
- Hierarchical memory
  - System memory
  - Private SPU memory

Challenges Presented

- Architectural challenges
  - Multiple processor architectures at the same time
  - Multiple memory address spaces
- Programming challenges
  - Breaking program up into the right size (small) pieces
  - Writing the computational kernels
  - Manual data movement through the different levels of memory
  - Understanding memory usage patterns

Cell Port of TotalView

- Linux Cell support
  - Shows SPU threads
  - Asynchronous thread control
  - SPU and PPC threads have separate instruction sets
  - Each thread has its own address space
  - Correct addressing on PPU and SPU
  - Lays groundwork for debugging other hybrid computing architectures



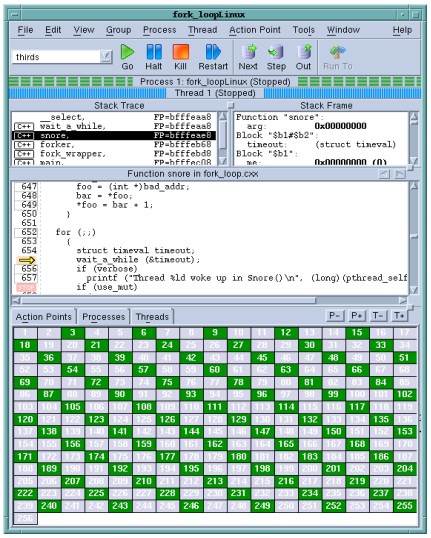
Cross and Heterogeneous Debugging

Host Platform \ Target Platform	Linux-x86	Linux-x86-64	Linux-PPC32	Linux-PPC64	Linux-Cell
Linux-x86	X		X	X	X
Linux-x86-64	X	X	X	X	X
Linux-PPC32					
Linux-PPC64			X	X	X
Linux-Cell			X	X	X

- Heterogeneous and cross debugging support
  - Mixture of Cell + X86-64 Linux targets in one session
  - Linux-Power32 and Power64 from a Linux-x86-64 host
  - Linux-x86 from a Linux-x86-64

TotalView Debugger

- C, C++, Fortran 77, Fortran90, UPC
  - Complex language features
- Wide compiler and platform support
  - Blue Gene/L and /P
  - Linux x86, x86-64, Power, Cell
  - Mac, Cray XT, and others
- Parallel debugging
  - MPI, pthreads, OpenMP, UPC
- Memory debugging capabilities
  - Integrated into the debugger
- Remote Display Client
- Graphical user interface
  - Simple things are easy
  - Advanced operations available
  - Visualization
- Scripting
  - CLI and TVScript



NVIDIA GPU accelerator

- Distinct processor architecture
  - Compared to host CPU
- *Many* more cores
  - Hundreds of streaming processors
  - Potentially 10k+ thread contexts
- Hierarchical memory with more layers
  - Local (thread)
  - Shared (block),
  - Global (GPU)
  - System (host)

Cuda and OpenCL

- Languages to support directly programming applications for GP-GPU
  - Cuda is NVIDIA hardware-specific
    - Compilers and tool chain available now
  - OpenCL is device-independent
    - Being defined

Cuda/NVIDIA Port of TotalView

- Currently in development
  - Building on many of the changes from the Cell port
- Planned characteristics
  - Full visibility of all device/kernel threads
    - Threads shown as part of the parent unix process
  - Correctly handle instruction set differences
  - Correctly handle the hierarchical memory
  - Provide thread control within architectural limitations
  - Provide allocation tracking of the device memory
    - Memory debugging

Cuda/NVIDIA Early Experience Program

- Way for users to participate in the development of TotalView for Cuda
  - Provide input into the design and development
  - Provide feedback on questions
    - How to group threads and provide status data without overwhelming the user
    - How to manage and control threads
    - How to display data from 10k + threads
  - Get early access to pre-release software before it is available to the public
  - NDA program
  - Sign up now
    - Contact Chris Gottbrath at TotalViewTech: Chris.Gottbrath@totalviewtech.com