Debugging CUDA Accelerated MPI Codes

Chris Gottbrath
Principal Product Manager, Rogue Wave Software

Aug 16th, 2011
Agenda

• Rogue Wave Software
  — TotalView
  — MemoryScape
  — ReplayEngine
  — ThreadSpotter
• CUDA Debugging
  — Intro and Demo
• Memory Debugging
• Automated Debugging
• Technology Update
  — New Features and Capabilities
  — Scalability
• Conclusion
Rogue Wave Today

The largest independent provider of cross-platform software development tools and embedded components for the next generation of HPC applications.

- **Visual Numerics**: Leader in embeddable math and statistics algorithms and visualization software for data-intensive applications.
- **Acumem**: Leading provider of intelligent software technology which analyzes and optimizes computing performance in single and multi-core environments.
- **TotalView**: Industry-leading interactive analysis and debugging tools for the world’s most sophisticated software applications.
Rogue Wave Product Offerings

IMSL
SourcePro C++
PV-WAVE
PyIMSL
ThreadSpotter
TotalView
ReplayEngine
MemoryScape
What is TotalView?

- **Application Analysis and Debugging Tool: Code Confidently**
  - Debug and Analyze C/C++ and Fortran on Linux, Unix or Mac OS X
  - Laptops to supercomputers (BG, Cray)
  - Makes developing, maintaining and supporting critical apps easier and less risky

- **Major Features**
  - Easy to learn graphical user interface with data visualization
  - Parallel Debugging
    - MPI, Pthreads, OpenMP, GA, UPC
    - CUDA Support available
  - Includes a Remote Display Client **freeing users to work from anywhere**
  - Includes Memory Debugging with **MemoryScape**
  - Reverse Debugging available with **ReplayEngine**
  - Includes Batch Debugging with **TVScript and the CLI**
What Is MemoryScape?

- **Runtime Memory Analysis**: Eliminate Memory Errors
  - Detects memory leaks before they are a problem
  - Explore heap memory usage with powerful analytical tools
  - Use for validation as part of a quality software development process

- **Major Features**
  - Detects
    - Malloc API misuse
    - Memory leaks
    - Buffer overflows
  - Supports
    - C, C++, Fortran
    - Linux, Unix, and Mac OS X
    - MPI, pthreads, OMP, and remote apps
  - Low runtime overhead
  - Easy to use
    - Works with vendor libraries
    - No recompilation or instrumentation
  - Enables Collaboration
What Is ReplayEngine?

- **Reverse Debugging Tool:** Radically simplify your debugging
  - Captures and Deterministically Replays Execution
  - Eliminate the Restart Cycle and Hard-to-Reproduce Bugs
  - Step Back and Forward by Function, Line, or Instruction

- **Major Features**
  - Simple extension to TotalView
    - No recompilation or instrumentation
    - Explore data and state in the past just like in a live process
  - Supported on Linux x86 and x86-64
  - Supports MPI, Pthreads, and OpenMP
What is ThreadSpotter?

- **Runtime Cache Performance Optimization Tool:** Tune into the Multi-Core Era
  - Realize More of the Performance Offered by Multi/Many-Core Chips
  - Quickly Detects and Prioritizes Issues -- and then Provides Usable Advice!
    - Brings Cache Performance Into Reach for Every Developer
    - Makes Experienced Cache Optimizers Hyper-Efficient
- **Features**
  - Supports Linux x86/x86-64
  - Any compiled code
  - Runtime Analysis
    - Low overhead
  - Cache Modeling
    - Prioritizes Issues
    - Identifies Problem Lines of Code
  - Provides Advice
    - Explanations
    - Examples
    - Detailed statistics (if desired)
Programming for the GP-GPU

• CUDA
  – Function-like kernels are written for the calculations to be performed on the GPU
    • Data parallel style, one kernel per unit of work
  – Presents a hierarchical organization for thread contexts
    • 2D grid of blocks
    • 3D block of thread
  – Exposes memory hierarchy explicitly to the user
  – Includes routines for managing device memory and data movement to and from device memory using streams

• Programming challenges
  – Coordinating CPU code + device code
  – Understanding what is going on in each kernel
    • Exceptions
  – Understanding memory usage
  – Understanding performance characteristics
TotalView for CUDA

- **Characteristics**
  - Debugging of application running on the GPU device (not in an emulator)
  - Full visibility of both Linux threads and GPU device threads
  - Fully represent the hierarchical memory
  - Thread and Block Coordinates
  - Device thread control
  - Handles CUDA function inlining
  - Reports memory access errors
  - Multi-Device Support
  - Can be used with MPI

- **Supports CUDA 4.0 (in beta)**
Memory Debugging

- **Heap Memory**
  - User is responsible for managing
  - C: Malloc / Free
  - C++: New / Delete
  - F90: Allocate / Deallocate

- **Buffer Overrun / Array Bounds Violations**
- **Memory Leaks**
- **Memory Optimization**
Heap Array Bounds Violations

- **Writing Outside of Allocation**
  - Can result in random errors
  - Dangling pointer
  - Array index error (off by one)

- **Guard Blocks**
  - Lightweight (few bytes per allocation)
  - Fast
  - Notification on demand
  - Notification after free

- **RedZones**
  - Heavier (page per allocation)
  - Fast
  - Notification at point of error
Leak Detection

- Based on Conservative Garbage Collection
- Can be performed at any point in runtime
  - Helps localize leaks in time
- Multiple Reports
  - Backtrace Report
  - Source Code Structure
  - Graphically Memory Location
Memory Optimization

- Prevent OOM errors
- Mem Usage
  - Per process
  - Per library
  - Per function
- Compare
  - Between
    - Processes
    - Points in Time
    - Datasets
    - Runs
- Track
  - Automate reporting
Automatic Debugging

- **Non-Interactive Batch Debugging**
  - Work in the “main” batch queue
  - Don’t have to baby-sit job waiting on it to run
  - Can script to perform checks that would be tedious to do by hand
  - Verification can be part of automated processes (nightly build and test)

- **Automatic Transformation of Data**
  - Simplify interactive (and scripted) debugging
  - Perform validation/sanity checking of large datasets
  - Comparative debugging
  - Allows you to focus on troubleshooting your program
TVScript Overview

- Gives you non-interactive access to TotalView’s capabilities
- Useful for
  - Debugging in batch environments
  - Watching for intermittent faults
  - Parametric studies
  - Automated testing and validation
- TVScript is a script (not a scripting language)
  - It runs your program to completion and performs debugger actions on it as you request
  - Results are written to an output file
  - No GUI
  - No interactive command line prompt
TVScript Syntax

- **tvscript syntax:**
  - tvscript [ options ] [ filename ] [ -a program_args ]

- **Options express (“event”, ”action”) pairs**
  - **Typical events**
    - Action_point
    - Any_memory_event
    - Guard_corruption
    - error
  - **Typical actions**
    - Display_backtrace [-level level-num] [num_levels] [options]
    - List_leaks
    - Save_memory
    - Print [-slice {slice_exp} {variable | exp}]

- **Example**
  - `create_actionpoint "#85=>print foreign_addr"`
C++View

- C++View is a simple way for you to define type transformations
  - Simplify complex data
  - Aggregate and summarize
  - Check validity
- Transforms
  - Type-based
  - Compose-able
  - Automatically visible
- Code
  - C++
  - Easy to write
  - Resides in target
  - Only called by TotalView
C++View Interface

• Only two functions:

  int TV_ttf_display_type ( const T * )

  int TV_ttf_add_row (  
      const char * field_name,  
      const char * field_type,  
      const char * address)
Scalability In TotalView Today

- **A Long History of Leadership**
  - Have worked with customers such as LLNL, LANL, Sandia and others on scalability improvements for many years

- **TotalView Architecture**
  - No Hard Limit
  - Multi-Platform (Cray, IBM BG, Linux Clusters, etc..)
  - Efficient Use of Cluster Resources
    - Extremely light weight debug agents
    - Minimal memory footprint (efficient shared data structures)
    - Each agent can control many processes and threads
  - Challenging User Applications
    - More space on the compute nodes for user application code

- **Customer Experiences**
  - TotalView is regularly used to debug scales of up to 10k processes
  - TotalView is also used on >10k processes
Research and Development

• **Current Focus Areas**
  – Transition TotalView from a flat 1:N communication to a tree
  – Scalable presentation of state and data
  – Usability at scale
  – Application driven tuning: Optimization focused on real-world applications and workloads
    • Across various machines

• **Goals**
  – Provide performance at >100,000 tasks to be debugged
  – Setting the stage for the millions of tasks we expect to see at exascale

• **Several Concurrent Projects**
  – FastOS project with Bart Miller and Mike Brim of University of Wisconsin
    • TBON-FS Group File Operations
    • Academic research based on MRNet & Dyninst components
  – LLNL Petascale Parallel Debugger Scalability contract
    • MRNet - product R & D
    • Multi-platform: BlueGene/Q, Cray XT/XE/XK, Linux Cluster
    • Preliminary results
      – First user observable improvements are in start up time
      – 5x improvement in at-scale start up performance on Cray
      – 20x improvement in at-scale start up performance on a “vanilla” linux cluster.
  – LLNL IDDA Dynamic Application contract
    • Focusing on a class of tool-breaking applications
    • Thousands of DLLs and Huge Symbol Table Size
Peta and Exascale Scalability

- R&D work is planned to roll into the product releases 2012 and 2013
  - Multi-platform Application Based Optimization
    - Cray XT/XE/XK, Blue Gene/Q, Linux clusters
    - Scientific applications including especially dynamic apps
    - GPU accelerated cluster scalability
  - Tree-Based Overlay Network
    - Broadcast of Operations
    - Aggregation of Events and Data
  - UI Layer
    - New GUI Framework
    - Co-Design of Advanced Displays for Debugging at Scale
    - Simplified Discovery of Relevant Information Through Aggregation

- These changes set the stage for exascale debugging
  - Multi-platform
  - Highly real-world optimized
  - Tree based
  - Low resource usage
  - Support for computational accelerator technology
  - Highly flexible architecture with an exclusive focus on HPC
Recent Changes

- **TV 8.9 series**
  - Powerful parallel debugging
  - Support for CUDA 3.0 - 4.0 (in beta)
  - New Views: Multi-dimensional Array & Parallel Backtrace
  - C++View and TVScript for Automatic Debugging
  - Easy and Secure Remote Graphical Display
  - Updated platform support

- **ReplayEngine 2.0 series**
  - Deterministic Replay Radically Transforms Debugging
  - Brings Reverse Debugging to HPC Clusters

- **MemoryScape 3.2 series**
  - Memory Leaks and Array Bounds Checking for HPC
  - Red Zones for Instant Array Bounds Checking

- **ThreadSpotter 2011**
  - Memory Cache Optimization Made Easy
Summary

• Rogue Wave
  HPC tools, components and libraries
  Parallel Programming is Hard, We Make it Easier

• Debugging with the TotalView Family of Products
  – Advanced, Scalable, Graphical, Easy to Use
  – MPI Debugging
  – CUDA Debugging
  – Memory Debugging
  – Automated Debugging
  – Deterministic Reverse Debugging

• Optimization with ThreadSpotter
  – Programmer Friendly Analysis of Cache and Memory Use
Thanks!

- Contact me
  - Chris.Gottbrath@RogueWave.com

- or my colleagues
  - Ian.Dillan@RogueWave.com
  - Ed.Hinkel@RogueWave.com

- or for more information
  
  Check out: www.roguewave.com
  
  Email: TVSupport@roguewave.com