Cray Debugging Support Tools

Luiz DeRose
Programming Environments Director
Cray Inc.
ldr@cray.com

The Next Generation of Debuggers on Cray Systems

- Peta-scale systems with hundreds of thousands of threads need a new debugging paradigm
  - Innovative techniques for productivity and scalability
    - Scalable Solutions based on MRNet
      - STAT - Stack Trace Analysis Tool
      - ATP - Abnormal Termination Processing
    - Comparative debugging
      - A **data-centric paradigm** instead of the traditional control-centric paradigm
    - Fast Track Debugging
      - Debugging optimized applications
  - Support for traditional debugging mechanism
    - TotalView, DDT, and gdb

MRNet - Multicast Reduction Network

- Tree based software overlay network
- Provides efficient multicast and reduction communications for parallel and distributed tools
- Uses a tree of processes between the tool's front-end and back-ends to improve group communication performance
  - Internal processes are used to distribute important tool activities
    - Reduce data analysis time
    - Keep tool front-end loads manageable

MRNet Based Debugging Support Tools

- STAT - Stack Trace Analysis Tool
  - Scalable generation of a single, merged, stack backtrace tree
    - 128K processes analyzed in 2.7 seconds, using MRNet
    - A comprehensible view of the entire application
    - Focuses and directs debugger subset attach opportunities
- ATP - Abnormal Termination Processing
  - Scalable analysis of a sick application, delivering a STAT tree and a minimal, comprehensive, core file set.
    - Minimal impact on application run
    - Automated, transparent collection of data
    - Ability to hold failing application for close inspection

ATP: The Problem Being Solved

- Applications on Cray systems use hundreds of thousands of processes
- On a crash one, many, or all of them might trap
- No one wants that many core files
- No one wants that many stack backtraces
- They are too slow and too big
- They are too much to comprehend

ATP Description

- System of light weight back-end monitor processes on compute nodes
- Coupled together with MRNet
- Leap into action on any application process trapping
- STAT like analysis provides merged stack backtrace tree
- Leaf nodes of tree define a modest set of processes to core dump
- Or, a set of processes to attach to with a debugger

STAT (Stack Trace Analysis Tool)

- Lawrence Livermore and University of Wisconsin
- Scalable collection of stack backtraces
- Fast, scalable, and compact

ATP Components

- Application process signal handler
  - triggers analysis
- Back-end monitor
  - collects backtraces via StackwalkerAPI
- Front-end controller
  - coordinates analysis via MRNet

ATP Requirements

- Minimum jitter
- Scalability
- Robustness
- Small footprint
- Limited core file dumping
- On by default

Limiting Core File Dumping

- ATP must overtly request dumping
- RLIMIT used to block accidental cascade of dumps
- core_pattern enhancement for "just in time" control of naming

Signal Handler Robustness

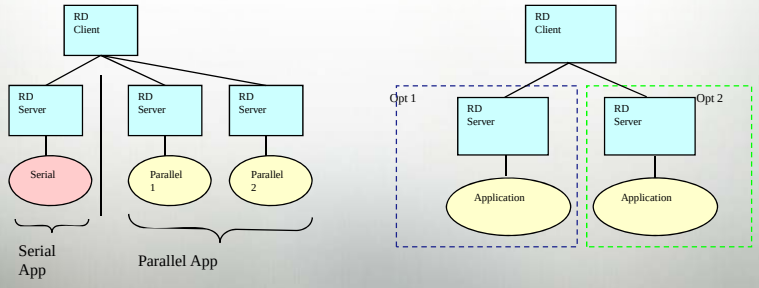
- Contrasted against ptrace
- Pre-allocated alternate stack
- State kept in read only memory segments

Additional features

- E-mail of failure status
- Stack backtrace of "first" failure to stderr???
- List of signaled processes and their signal
- Checkpoint/restartable

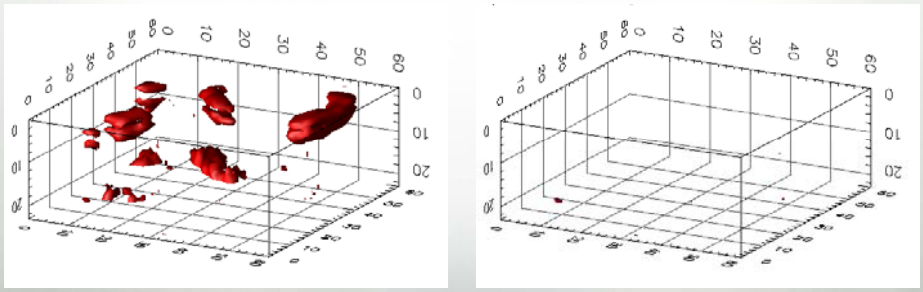
Comparative Debugging

- Compare working application to failing application
  - Simultaneous execution of both
  - Comparison of computed results at key moments
  - Focus on data – not state and internal operations
  - Narrow down problem without massive thread study
- Comparative debugging scenarios
  - Serial converted to parallel
  - Small scaling versus large scaling
  - One optimization level versus another
  - One programming language converted to another



Comparative Debugging Example

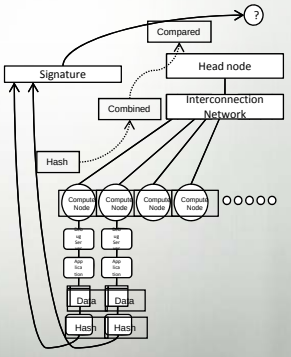
- MM5 Weather model
- Serial versus parallel
- Difference >0.1%
- Narrowed to missing term in equation
- **… term found and added in**



Courtesy of David Abramson from GuardSoft

Comparative Debugging at Scale

- Existing implementation of Guard ported to Cray XT Systems
- Modifying Guard core to improve scalability
  - Add process sets as primitive data types
  - Implement set operations for setting breakpoints, etc
  - considering MRNET tree for gathering breakpoint events
  - Removing reliance on socket connections to debug servers
  - Simplified original data decomposition specification language to match current high level languages
- Experimenting with alternative comparison techniques
  - Computing signatures on decomposed data structures rather than always importing data for comparison



Fast Track Debugging

- Debug and non-debug versions of each subroutine appear in the executable
- User sets breakpoints or other debug constructs
- Routines automatically presented using the debug version of the routine
- Rest of program executes using optimized versions of the routines

Fast Track Debugging

- Available in September 2009 Programming Environment Release
  - LGDB 1.2 (released in the Cray Debugger Support Tools 1.0)
  - LGDB is the launcher of the GNU debugger
  - Access by loading xt-lgdb/1.2
  - Compile application with CCE **–G fast** option
  - Launch application with lgdb (*see lgdb man page*)
  - Lgdb will give directions on how to attach gdb to your application processes

THE SUPERCOMPUTER COMPANY

Cray Debugging Support Tools

Questions / Comments
Thank You!