NCCS Brown Bag Series
TotalView on Discover:
Part 1

Parallel Debugging

Chongxun (Doris) Pan
doris.pan@nasa.gov
May 8, 2012
Agenda

• Overview
• Basic Navigation and Control
• Demo 1: Basic Navigation and Control
  ☑ Using an OpenMP Space weather application
• Debugging MPI Applications
• Demo 2: MPI Debugging:
  ☑ Using GEOS5-AGCM
TotalView 8.9.2-2 currently on Discover

• An interactive tool that lets you debug serial, multi-threaded and multi-processor programs with support for Fortran and C/C++

• The base TotalView debugger (TVD) solution now includes Replay Engine and CUDA debugging support for no additional fee

• Major features:
  - Parallel debugging: MPI, Pthreads, OpenMP, CUDA
  - Reverse debugging with ReplayEngine 2.1.0-2
  - Integrated Memory debugging with MemoryScape 3.2.2-2
  - Batch debugging with TVScript and the CLI
What is ReplayEngine?

- Reverse Debugging
  - Capture and deterministically replay execution
  - Eliminate restart cycle and hard-to-reproduce bugs
  - Step back and forward by function, line, or instruction

- Major features
  - No recompilation
  - Supports both MPI and OpenMP
  - Setenv TVD_REPLAY_TMPDIR to control the directory of saving the history information
What is MemoryScape?

- Runtime Memory Analysis to detect memory bugs
  - A memory bug is a mistake in heap memory usage
    - Leaking: failure to free memory
    - Dangling references: failure to clear pointers
    - Memory corruption
  - Use for validation as part of a quality software development process
- Major features
  - No recompilation
  - Supports both MPI and OpenMP
More Info

- Details on how to configure TV for your runs and use TV debugging serial, OpenMP, and MPI/OpenMP jobs can also be found in the NCCS Primer:
  
  http://www.nccs.nasa.gov/primer/computing.html#totalview

- Short TotalView video tutorials on interesting topics can be found at:
  
Agenda

• Overview
• **Basic Navigation and Control**
• Demo 1: Basic Navigation and Control
  ❖ Using an OpenMP Space weather application
• Debugging MPI Applications
• Demo 2: MPI Debugging:
  ❖ Using GEOS5-AGCM
Starting TotalView

- `-g` has to be used for debugging. Using `-g` automatically adds `-O0`
- Load module: `module load tool/tview-8.9.2-2`
- `setenv TVDSVRLAUNCHCMD ssh`
- Launch Totalview GUI via Command line
  - Normal: `totalview executable_name [-a executable_args]`
  - Attach to a running program: `totalview executable_name –pid PID# [-a executable_args]`
  - Attach to a core file: `totalview executable_name corefile_name [-a executable_args]`
Starting TotalView

discover25::/cpan2>xsub -I -V -l select=1:ncpus=12,walltime=30:00 -W group_list=k3001
Establishing X forwarding and submitting batch job...
qsub: waiting for job 1790833,borgpbs1 to start
qsub: job 1790833,borgpbs1 ready

borgc002::/cpan2>module purge
borgc002::/cpan2>module load comp/intel-12.1.0.233 tool/tview-8.9.2.2
borgc002::/cpan2>more $PBS_NODEFILE
borgc002
borgc002::/cpan2>cd /discover/nobackup/cpan2/tmp/K_src/
borgc002::/K_src>setenv OMP_NUM_THREADS 4
borgc002::/K_src>which totalview
/usr/local/toolworks/totalview.8.9.2-2/bin/totalview
borgc002::/K_src>totalview ./my
my_exec*  mysweep.inc*
borgc002::/K_src>totalview ./my_exec
Root Window

- State of all processes being debugged
- Process and Thread status
- Instant navigation access
- Sort and aggregate by status

Status Info

- T = stopped
- B = Breakpoint
- E = Error
- W = Watchpoint
- R = Running
- M = Mixed
- H = Held
Root Window

Hierarchical/Linear Toggle

Host name

Rank # (if MPI program)

TotalView Thread ID #

Expand - Collapse Toggle

Process Status

Action Point ID number

NASA Center for Climate Simulation
Process Window

Provides detailed state of one process, or a single thread within a process.

A single point of control for the process and other related processes.
Stack Trace and Stack Frame Panes

Process 1 (10893): fork_loopLinux (At Breakpoint 1)
Thread 1.1 (10893) (Stopped)

Stack Trace

<table>
<thead>
<tr>
<th>Language</th>
<th>Name</th>
<th>Frame Pointer</th>
<th>Local Variables</th>
<th>Register Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>C++</td>
<td>wait_a_while</td>
<td>FP=bffffeaa8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C++</td>
<td>snore</td>
<td>FP=bffffeae8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C++</td>
<td>forker</td>
<td>FP=bfffeb68</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C++</td>
<td>fork_wrapper</td>
<td>FP=bfffeb8d</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C++</td>
<td>main</td>
<td>FP=bfffe08</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>__libc_start_main</td>
<td>FP=bfffe48</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Function "wait_a_while":
- timeout: 0xbffffead0 -> (str Block "$b2":
  - result: 0x00000000 (0)

Registers for the frame:
- Rax: 0x00000000 (0)
- Rcx: 0x00000000 (0)
- Rdx: 0x00000000 (0)
- Rbx: 0x401d7dd4 (1075674580)
- Rsp: 0xbffffeaa (107374798)
Tabbed Area

**Action Points Tab**
- all currently defined action points

**Processes Tab**
- all current processes

**Threads Tab**
- all current threads, ID’s, Status
Stepping Commands

Based on PC location

PC: Program Counter

1. int sub2(int);  
2. int sub3(int);  
3. int main ()  
4. {  
5.   int j, k, i = 0;  
6.   j = sub1(i); k = sub3(j);  
7.   printf("The value of k is %d\n", k);  
8. }  
9. int main()  
10. {  
11.   int j, k, i = 0;  
12.   j = sub1(i); k = sub3(j);  
13.   printf("The value of k is %d\n", k);  
14. }  
15. int sub1(int x)  
16. {  
17.   x = sub2(0);  
18.   return (x++);  
19. }  
20. int sub2(int y)  
21. {  
22.   y = y - 10;  
23.   y++;  
24.   return (y+10);  
25. }  
26. int sub3(int z)  
27. {  
28.   return (z*z);  
29. }
Action Points

Breakpoints

Barrier Points

Conditional Breakpoints

Evaluation Points

Watchpoints
Setting Breakpoints

- Setting action points
  - Single-click line number
- Deleting action points
  - Single-click action point line
- Disabling action points
  - Single-click in Action Points Tab Pane
- Optional contextual menu access for all functions
- Action Points Tab
  - Lists all action points
  - Dive on an action point to focus it in source pane
- Action point properties
  - In Context menu
- Saving all action points
  - Action Point > Save All
Evalpoints allow you to add a code fragment in Fortran or C/C++. If the code fragment can make a decision whether to stop execution, it is called a conditional breakpoint.

Evalpoints can be used to test a fix, set values of variables, or visualize data automatically.
Evalpoints Test Fixes on the Fly

- Test small source code patches
- Call functions
- Set variables
- Test conditions
- C/C++ or Fortran
- Can’t use C++ constructors
- Use program variables
- Can’t modify variables or call functions with replay engine
Watchpoints

Watchpoints are set on a fixed memory region

Use Tools > Watchpoint from a Variable Window or
From source pane with contextual menu
When the contents of watched memory change, the watchpoint is triggered and TotalView stops the program.

A Watchpoint tracks a memory location -- it does not trace a variable!

Watchpoints can be conditional or unconditional
TV variables $newval and $oldval can be used in the conditional expression
Uses Hardware Watchpoints with various limitations based on architecture
### Items you dive on:

<table>
<thead>
<tr>
<th>Item</th>
<th>Information Displayed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process or thread</td>
<td>When you dive on a thread in the Root Window, TotalView finds or opens a Process Window for that process. If it doesn’t find a matching window, TotalView replaces the contents of an existing window and shows you the selected process.</td>
</tr>
<tr>
<td>Variable</td>
<td>The variable displays in a Variable Window.</td>
</tr>
<tr>
<td>Expression List Variable</td>
<td>Same as diving on a variable in the Source Pane: the variable displays in a Variable Window.</td>
</tr>
<tr>
<td>Routine in the Stack Trace Pane</td>
<td>The stack frame and source code for the routine appear in a Process Window.</td>
</tr>
<tr>
<td>Array element, structure element, or referenced memory area</td>
<td>The contents of the element or memory area replace the contents that were in the Variable Window. This is known as a nested dive.</td>
</tr>
<tr>
<td>Pointer</td>
<td>TotalView dereferences the pointer and shows the result in a separate Variable Window. Given the nature of pointers, you may need to cast the result into the logical data type.</td>
</tr>
</tbody>
</table>
Diving on Variables - Fortran Common Blocks
Fortran 90 Modules

Tools → Fortran Modules
The Variable Window

Window contents are updated automatically
Changed values are highlighted

- Click once on the value
- Cursor switches into edit more
- Esc key cancels editing
- Enter key commits a change
- Editing values changes the memory of the program
Groups of Variables -- Expression List Window

Add to the expression list using contextual menu with right-click on a variable, or by typing an expression directly in the window.

- Reorder, delete, add
- Sort the expressions
- Edit expressions in place
- Dive to get more info

- Updated automatically
- Expression-based
- Simple values/expressions
- View just the values you want to monitor
Slicing and Filtering Arrays

Slice notion is ([start:end:stride], [start:end:stride],..)
Visualizing Arrays

- Visualize array data using Tools > Visualize from the Variable Window
- Large arrays can be sliced down to a reasonable size first
- Visualize is a standalone program
- Data can be piped out to other visualization tools

- Visualize allows to spin, zoom, etc.
- Data is not updated with Variable Window; You must revvisualize
- $visualize()$ is a directive in the expression system, and can be used in evaluation point expressions.
Variables Across Processes

- TotalView allows you to look at the value of a variable in all MPI processes
  - Right Click on the variable
  - Select the View > View Across
- TotalView creates an array indexed by process
- You can filter and visualize
- Use for viewing distributed arrays as well.
Basic Process Control

Groups

• Control Group
  – All the processes created or attached together

• Share Group
  – All the processes that share the same image

• Workers Group
  – All the worker threads within a control group.
  These threads can be in >=1 share groups

• Lockstep Group
  – All threads at the same PC

• Call Graph Group
  – All processes going through the same node in
    the call graph

• User Defined Group
  – Process group defined in Custom Groups dialog

NASA Center for Climate Simulation
Agenda

- Overview
- Basic Navigation and Control
  - Demo 1: Basic Navigation and Control
    - Using an OpenMP Space weather application
- Debugging MPI Applications
- Demo 2: MPI Debugging:
  - Using GEOS5-AGCM
Agenda

• Overview
• Basic Navigation and Control
• Demo 1: Basic Navigation and Control
  ❖ Using an OpenMP Space weather application
• Debugging MPI Applications
• Demo 2: MPI Debugging:
  ❖ Using GEOS5-AGCM
Tools ➔ Call Graphs

- **Quick view of program state**
  - Each call stack is a path
  - Functions are nodes
  - Numbers indicate which threads have a function on their call stack

- **Construct process groups**

- **Look for outliers**

Dive on a node in the call graph to create a Call Graph group.
Tools → Message Queue Graph

- Hangs & Deadlocks
- Pending Messages
  - Receives
  - Sends
  - Unexpected
- Inspect
  - Individual entries
- Patterns
Tips for debugging MPI jobs

• **Reduce N**
  - **Problem:** Each process added requires overhead
  - **Strategy:** Reduce the number of processes TotalView is attached to
    - Simply reducing N is best, however data or algorithm may require large N
  - **Technique:** subset attach mechanism

• **Focus Effort**
  - **Problem:** Some debugger operations are much more intensive than others, and when multiplied by N this could be significant
  - **Strategy:** Reduce the interaction between the debugger and the processes
  - **Technique:** Use TotalView’s process control features to
    - Avoid single stepping - Usually causing Totalview hanging
    - Focus on one or a small set of processes
TotalView does not need to be attached to the entire job

- You can be attached to different subsets at different times through the run
- You can attach to a subset, run till you see trouble and then 'fan out' to look at more processes if necessary.
- This greatly reduces overhead
- It also requires a smaller license if you have a TotalView Team license.
Tips for debugging MPI jobs (Cont’d)

• Setting breakpoint behavior
• Synchronizing processes
  ❖ Barrier and the process hold/release feature work together to control the executions
• Determining which processes/threads are executing
• Viewing variables across processes/threads
• Restarting from within Totalview
Agenda

• Overview
• Basic Navigation and Control
• Demo 1: Basic Navigation and Control
  ❖ Using an OpenMP Space weather application
• Debugging MPI Applications
• Demo 2: MPI Debugging:
  ❖ Using GEOS5-AGCM
Coming Up -- TotalView Part 2

• **Replay Engine**
  - Record and deterministic replay
  - Use breakpoints, watchpoints, and some conditional breakpoints when running in replay mode

• **Memory Debugging**
  - Why are memory bugs hard to detect?
  - How do your program’s data reside in memory?
  - Heap graphic view
  - Leak and dangling pointer detection
  - Memory corruption report
  - Memory usage statistics