Debugging with Totalview

Martin Čuma
Center for High Performance Computing
University of Utah
mcuma@chpc.utah.edu
Overview

- Totalview introduction.
- Basic operation.
- Serial debugging.
- Parallel debugging.
  - OpenMP
  - MPI
- Memory debugging.
- Some other useful information
Totalview in a nutshell

• source and machine level debugger
• command line and graphic interface
• serial and parallel debugging support
• supports remote debugging
• runs on variety of platforms
Totalview windows

Process 1003847: mpirun-trapp_r (At Breakpoint 4)
Thread 1003847:1: mpirun-trapp_r (At Breakpoint 4)

Function 'main'

```
int main()
{
    MPI_Send((void *)&local_a, 1, MPI_INT, 0, tag, MPI_COMM_WORLD);
    tag = 0;
    MPI_Send((void *)&local_b, 1, MPI_INT, 0, tag, MPI_COMM_WORLD);
    tag = 1;
    MPI_Send((void *)&local_b, 1, MPI_INT, 0, tag, MPI_COMM_WORLD);
    tag = 2;
}
```

Thread (1)

```
1/1003847:14 in main
```
Totalview basic operations

- Data examination
  - view data in the variable windows
  - change the values of variables
  - modify display of the variables
  - visualize data
- Action points
  - breakpoints and barriers (static or conditional)
  - watchpoints
  - evaluation of expressions
Multiprocess debugging

- Automatic attachment of child processes
- Create process groups
- Share breakpoints among processes
- Process barrier breakpoints
- Process group single-stepping
- “Laminate” variables
- Display MPI message queue state
1. Compile binary with debugging information
   - flag `-g`
     ```
     g77 -g test.f -o test
     ```
   - if use fork() or execve(), link ...
     ```
     g77 -g -L/.../totalview/linux-x86-64/lib -ldbfork
     test.f -o test
     ```

2. Run Totalview
   - TV + executable
     ```
     totalview executable
     ```
   - TV + core file
     ```
     totalview executable core_file
     ```
How to use Totalview

- run TV and attach the executable
  - start TV
  - menu New Program Window
  - fill in executable file name
- run TV and attach running program
  - start TV
  - menu New Program Window
  - pullout Attach to an Existing Process
  - choose process ID and fill in executable file name

3. Totalview operation
- left mouse button - select
- right mouse button - menu
- left mouse button double click - dive
Example 1 – Serial code

• CUSP – DFT plane wave atomic simulation
  ▪ atomic nuclei interact classically
  ▪ electrons described by wavefunctions - plane waves
  ▪ interaction between w.f.'s and nuclei define the system potential and kinetic energy
  ▪ iterate electronic and nuclear energy to self consistency to achieve ground state

• Implementation
  ▪ 10+ Fortran source code files
  ▪ simple Makefile w/ flags -g -byteswapio
Process view window

- Stack trace – procedure hierarchy
- Stack frame – variables display
- Source code – code + process navigation
- Threads list – in case of multithreaded application
- Action points – list of breakpoints, barriers,…
Running the debugger

- Menu Go/Halt/Next/Step/Hold or shortcuts
- Possible actions (thread, process/group):
  - go (g/G)
  - halt (h/H)
  - step (source line) (s/S)
  - step (instruction) (i/I)
  - next (source line) (n/N)
  - next (instruction) (x/X)
  - run (to selection) (r/R)
  - return (out of function) (o/O)
Action points

- Breakpoints and barriers
  - toggle location with left mouse (shift for barrier)
  - right-click – Properties for options
- Evaluation points
  - set conditional breakpoints
  - conditionally patch out code
- Watchpoints
  - watch for change in a memory location
• Variable view
  - dive (right mouse) on any variable
  - change data type
  - select an array slice, e.g. (3:3,:)
  - filter array values, e.g. .ne. 0
• Variable visualization
• menu Visualize – only up to 2D arrays
Example 2 – OpenMP code

• Compilation
  ▪ Arches:
    pathf90 -openmp test.f -g -o test.exe
    pgf90 -mp test.f -g -o test.exe
    ifort -openmp test.f -g -o test.exe
    gcc4 -fopenmp test.c -g -o test.exe

• Running
  ▪ set OMP_NUM_THREADS
• Example – saxpy routine
  ▪ simple vector-scalar product
OpenMP specific debugging

- TV automatically attaches all threads
- put breakpoint to OpenMP parallel section to debug threads
- variable lamination - show values from all threads in one window – does not always work
- barrier points – shift-left click
- ambiguous action points – select all
Example 3 – MPI/OpenMP code

• Job submitter code
  - one MPI process – master – distributes work
  - other MPI processes – slaves – do the work

• Implementation
  - master reads job database and sends work information to slaves
  - master runs two threads via OpenMP, one distributes the work, the other runs the work
  - slave processes the information and runs the work (serially via fork() and execve() functions)
Example 3 scheme

Master reads database with available jobs, and sends work requests to available workers
Example 3 – MPI code – Arches

- MPICH2 on interactive nodes
  - good enough for most parallel problems
  - source
    /uudfs/arches/sys/pkg/mpich2/std_pgi/etc/mpich2.csh
  - mpicc submit.o -g -o submit

- Running
  - Start MPD daemon on the interactive node
    mpdboot -n 1
  - if have problems, create ~/.mpd.conf file
  - Run:
    open Totalview (totalview), then in New Program Window enter:
    - program name
    - MPICH2 for parallel system
    - number of processors
Example 3 – MPI code – Arches

- Running on compute nodes
  - use larger number of processors or other MPI than MPICH2 is needed
  - `pgcc -g -mp -I/uufs/arches/sys/pkg/mpich/std/include submit.c`
  - `pgcc submit.o -L/uufs/arches/sys/pkg/mpich/std/lib -lmpich -o submit`

- Running on compute nodes (large # of processors)
  - contact CHPC to get ssh access to the nodes
  - interactive PBS:
    - `qsub -I -l nodes=2,walltime=2:00:00 hostname -i`
  - ssh to node name obtained above
Example 3 – MPI code – Arches

- Running on interactive nodes
  - run:
    
    `/uufs/arches/sys/pkg/mpich/std/bin/mpirun -tv -np 2 -machinefile $PBS_NODEFILE ./submit`
  - modify path for the tdsvr – Totalview server:
    
    Menu File – Properties – Launch strings

    `/uufs/arches/sys/pkg/totalview/std/bin/tdvsrvr`
  - Set a breakpoint after MPI_Init and select Go
  - Select No or Yes at the next dialog, does not matter since we already have a breakpoint set
  - Debug and enjoy
MPI specific debugging

- Process synchronization – program groups
- Barrier points
- Message queue state graph and display
Memory debugging

- display memory status
- paint allocated and deallocated blocks
- find memory leaks
- identify dangling pointers
- enable with Tools > Memory Debugger > Enable memory debugging checkbox
Some useful resources

- TotalviewTech webpage
  http://www.totalviewtech.com
- Location of Totalview
  Arches:/uufs/arches/sys/pkg/totalview/std
  Some group desktops: inquire at CHPC
- Documentation
  http://www.totalviewtech.com/Support/docs/index.html
  http://www.chpc.utah.edu/software/docs/par_devel.html
  http://www.chpc.utah.edu/software/docs/totalview.html
  http://www.chpc.utah.edu/short_courses/Totalview
Totalview
Student Edition

- Free for students
- Limited to one computer, 4 processes
- To sign up, e-mail mcuma@chpc.utah.edu:
  - name
  - e-mail
  - university ID
  - anticipated year of graduation
- More details
  http://www.totalviewtech.com/academia/totalview_express.html
Code checkers

• compilers check for syntax errors
  – some compiler flags help too (-C)
• memory checking tools - many errors are due to bad memory management
  – valgrind – easy to use
  – purify – harder to use
Thread checking

• Good for finding OpenMP errors
  – race conditions
  – privatization
• Intel thread checker (ITC)
  – OpenMP
  – pthreads
Using ITC

• Source ITC environment
  
  source
  /uufs/arches/sys/pkg/itc/std/tcheck/bin/32e/tcheckvars.csh

• Compile with -tcheck -g
  ifort -openmp -tcheck -g trap.f

• Run tcheck
  tcheck_cl ./a.out

• More info
Future Presentations

- Mathematical libraries at the CHPC
- Chemistry packages (Anita Orendt)
- Using Gaussian (Anita Orendt)
- Performance analysis with TAU/Vampir
- Introduction to OpenMP
- MPI-IO
- Mixed MPI/OpenMP programming Intermediate MPI ???