Debugging with DDT at NERSC

David Lecomber
CTO Allinea Software
Contents

- Overview of Allinea and DDT
- Using Allinea DDT to fix bugs at NERSC
- Hands on
Allinea Software

- HPC tools company since 2001

- Core products
  - **DDT** - Debugger for MPI, threaded/OpenMP and scalar applications
  - **OPT** - Optimizing and profiling tool for MPI and non-MPI applications
  - **DDTLite** - Plugin for Microsoft Visual Studio 2008
    - Adds parallel and multi-threaded components to user interface
    - Real parallel debugging for Windows!
    - Released September 22nd 2008
High Profile Clients (extract)

• Government
  – CEA, NERSC, IDRIS, BSC, ONERA, AWE, RAL, HLRS, CASPUR, CINECA, ORNL, NERSC, LLNL, Ifremer, Proudman, Plymouth Marine, BGS

• Universities
  – Sharcnet, Jülich, North West Grid, Vanderbilt, Penn State, TACC, IPGP, ETHZ, HLRS, LRZ, Dresden, Karlsruhe, ICHEC, Bristol, Nottingham, Glasgow, Edinburgh, Oxford, Tokyo, USATU, Arizona, UCL

• Aerospace research
  – DLR, EADS CCR, CIRA, MBDA, BAE Systems

• Commercial research
  – Synopsys, Airbus, Fujitsu (Japan & UK), CGGVeritas, Total, IFP, OHM, AVL, MTEM, Konica-Minolta
Other Allinea products: DDTLite

- **DDTLite for Visual Studio**
  - Simplifying development on the Microsoft® platform
  - Bringing features from DDT into Visual Studio®
  - Take total control of all processes
  - Makes Visual Studio® a true parallel debugger
  - Released last week!
Debugging with DDTLite

- Group control
  - Step or play multiple processes
- Compare data over processes
- See all the process stacks
- C/C++/C#, Intel or PGI Fortran
Using DDT
Using DDT to fix bugs - fast!

• Overview of the features
• How to get started with DDT
• Memory Debugging with DDT
Parallel Software is Complicated

• **Multithreaded, multiprocess code**
  – The usual issues: bugs, speed
  – ... now add communication, synchronization, race conditions, deadlock, scalability

• **Hardware / Software capability misaligned**
  – Significant gap between user requirements and application capabilities
  – Emphasis is now moved to software development
  – How do we address this issue?
Some problems DDT helps with

• For when you have a bug – if your application
  – Crashes all the time
    • Click play and run until it crashes – DDT shows the point of segfault, abort or exit – before it has quit
  – Crashed yesterday
    • Load up the core file and look at where it crashed
  – Crashes sometimes
    • Possibly a memory error – turn on memory debugging and force the problem
  – And many other scenarios – not just crashes

• It helps when maintaining and developing code
  – When you need to know how it works to add a feature
    • Load DDT and just watch how your program runs
DDT - Distributed Debugging Tool

• A mature, powerful and highly intuitive tool
  – Traditional focus has been HPC
  – Developing new capabilities.....multicore, GP-GPU

• Cross-platform support
  – Linux, Solaris, AIX, Super-UX
  – GNU, Absoft, IBM, Intel, PGI, PathScale, Sun compilers
  – x86, x86-64, ia64, PowerPC, Cell, UltraSparc, NEC
  – Support for all major scheduling systems
  – Across all MPIs
  – OpenMP and Threads
The basic features

• Automatically locates source files
• Step, play, pause
• Breakpoints and data watchpoints
• Fortran 90, 95 and 2003
  – Modules, allocatable data, pointers and derived types
• C, C++ support
  – Structs, classes, pointers, STL, namespaces, virtual functions and templates
  – Catch exceptions at throw or at catch point
Features for multithreading

• Perform actions individually or collectively
• Breakpoints for all or per thread
• See all threads at a glance
  – Parallel stack view shows threads and groups common stacks
Features for parallel codes

- Support for every MPI
- Control processes individually or by groups
- Visualize message queues
Viewing concurrent data

- Cross process/thread comparison
Viewing vast data

- Visualize multidimensional data
  - From 2D viewer to new multidimensional viewer
  - 3D OpenGL array viewer (stereo !)
Scaling up

- **Increased thread and processor counts are coming**
  - Desktop multicore and GPUs - multithreading by default
  - 1,000+ core clusters everywhere

- **DDT allows control of 1,000s of processes**
  - An MPI application or a custom network of processes
  - Highly threaded codes GPUs, and (very-) multi-core ready
  - Or many single core Monte Carlo applications

**DDT 2.4 Measured Timings**
Dual Core 2.0 GHz 1GB RAM frontend

![DDT 2.4 Measured Timings Graph]

- Startup
- Step Over
Finding rogues quickly

- **Parallel Variable View**
  - Find processes or threads with different data

- **Parallel Stack View**
  - Identifies classes of process and thread behaviour easily
  - Allows rapid grouping of processes

- **Integrated with process groups**
Using groups for process control

- At lower parallelism
  - See state of each process
  - Drag and drop to manage groups

- At 100s or 1000s of processes
  - Summary view gives rapid overview and same level of control
How to get started at NERSC

• Compile your code – with “-g” option
• Bring up the DDT GUI - “ddt”
• Select your program
• Click submit! DDT submits the job for you.

• Is it that easy?

• Not quite:
  – Temporarily complicated until MPT update to 3.1
    • module unload xt-mpt/3.0.2
    • module load xt-mpt/2.0.53b
    • Recompile your code
Considerations for NERSC

• **Interactive sessions?**
  – Change “Session/Options/Job Submission” so that DDT does not submit for you.
    • “qsub -l ....” and wait, then launch DDT in the shell
  – Why would I do that?
    • Quickly and repeatedly try something: run 10 times in a 10 minute session instead of waiting in the queue 10 times
  – Why wouldn't I do that?
    • Easier not to – is a trade off – it's convenient if your program runs for a while

• **Using the queue differently with DDT**
  – Can change the default queue at click of button
    • Click “change” on launch page default is “debug”
  – Can use DDT up to 1024 processes
    • “regular” queue, not “debug” queue
    • But remember your budgeted hours....!!
  – Change the walltime in the same manner
Interlude!

DDT demo
A Tour of Memory Debugging
Focus on Memory Debugging

- DDT's memory debugging works with the Heap
  - Memory allocated via allocate (F90), malloc (C), or new (C++)
Memory Debugging

Check your overall memory usage
Leaks

- Common C coding error, increasingly in F90
  - When memory is not released after use
    - Kills job if repeated too often, eg. calls to leaky function
  - First identify if there is a problem
  - Then find it..

```c
void fun()
{
  Car *p;
  p = new Car();
  /*
  ... do something
  */
  return;
}
```
Checking for leaks

Check your current memory usage and where memory was allocated

Locate where memory was allocated in depth
Memory Errors

• **Dangling Pointers**
  - Pointer variables pointing to memory after it has been discarded
  - Pointing to discarded memory now reissued to another role

• **We can prevent both**

```cpp
Car *p, *q;
p = new Car();
q = p;
/*
... do something
*/
delete p;
/*
... something else
*/
q->drive();
```
Beyond Bounds Reading

- **“Guard Pages”**
  - Instant protection against read/write outside an array

- **DDT uses OS to lock the page**
  - Above or below for protection (but not both)
  - Or lock more pages.. for multi-dimensional arrays

- **.. and can still allow you to continue!**

- **“Fence post” checking**
  - Periodic protection against write outside an array – at both ends
Memory Debugging

Stop immediately on reads beyond array end and common errors
Using memory debugging

- **Easy to use**
  - Tick check box to enable and use the default, or choose your own settings
  - Do not tick “Preload” - we use link explicitly on Cray

- **Relink**
  - Use “-Bstaticddt” compile option at NERSC
Q & A: Open Floor

• Any questions?