

CoE Meeting; December 10, 2014

Tools Discussion

Mahesh Rajan & Dennis Ding
SNL

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.



Tools installed and in use at SNL

- vTune
 - Most used by code developers; More usage as we go to hybrid codes; not optimal for MPI and at scale
- CrayPat
 - We have lot of experience with it
 - Stresses our SIERRA apps; for large scale runs/trace
- Allinea DDT & Map
- OpenSpeedShop
 - Dynamic instrumentation; easy to use as no mods to the executable
- HPCToolkit
 - Worked with Prof Mellor Crummy to improve use with OpenMP; Nice features not in other tools.
- mpiP & memP
 - Easy to use and low overhead; used frequently;
- Totalview
 - Most used debugger; DDT is getting more exposure
- Intel's new Vector Advisor (looking to participate in the Alpha program);
 - Looks like a powerful tool ; complements Cray's Reveal



Requirements

- Ease of use; users often discouraged by steep learning curve
- The real challenge: go from generating large amount of data to insights for tuning
- Need training with vectorization tools
 - Reveal
 - Vector Advisor
- Working with Intel local team to set up compilers & tools training to the larger SNL code development teams in early 2015
- SIERRA/SD teams feedback
 - Tools for memory utilization; need thread based memory counter
 - Tools to help identify non-thread-safe code
- It's been a while since we have had a training on Cray's Program development environment – Need to set up soon!
 - Cray's PrgEnv
 - CrayPat and Apprentice
 - Reveal

Some SNL user experience with Intel Inspector

Program FDEM calculates a 3D finite-difference solution of the six coupled first-order electromagnetic partial differential equations (the "EH system") appropriate for isotropic electromagnetic media. Numerical solution methodology is explicit, time-domain, finite-differencing with $O(2,N)$ FD operators on staggered spatial and temporal grids.

❖ Program: FDEM

- Version: 2.1 - memory efficient
- Primary Developer: D.F. Aldridge
- Secondary Developer: K.A. Schramm
- Last Revision Date: 13 March 2014

❖ User threading of FDEM Fortran code with OpenMP

❖ After ~50 Cycles and -O3 NaNs occurred

❖ Intel inspector identified several race conditions and uninitialized variables

❖ It's a nice tool for use in the porting process

❖ Easy to show the user how to use it

❖ This is very important because user won't use complicated tools

Inspector used to locate threading errors in FEDM

The screenshot displays the Intel Inspector XE 2013 interface. The main window is titled "Locate Deadlocks and Data Races" and shows a list of 18 data race problems (P1 through P18) detected in the module `fdem_v2.13.exe`. The problems are categorized as "Data race" and "New".

ID	Type	Sources	Modules	State
P1	Data race	fdem_v2.13.f	fdem_v2.13.exe	New
P2	Data race	fdem_v2.13.f	fdem_v2.13.exe	New
P3	Data race	fdem_v2.13.f	fdem_v2.13.exe	New
P4	Data race	fdem_v2.13.f	fdem_v2.13.exe	New
P5	Data race	fdem_v2.13.f	fdem_v2.13.exe	New
P6	Data race	fdem_v2.13.f	fdem_v2.13.exe	New
P7	Data race	fdem_v2.13.f	fdem_v2.13.exe	New
P8	Data race	fdem_v2.13.f	fdem_v2.13.exe	New
P9	Data race	fdem_v2.13.f	fdem_v2.13.exe	New
P10	Data race	fdem_v2.13.f	fdem_v2.13.exe	New
P11	Data race	fdem_v2.13.f	fdem_v2.13.exe	New
P12	Data race	fdem_v2.13.f	fdem_v2.13.exe	New
P13	Data race	fdem_v2.13.f	fdem_v2.13.exe	New
P14	Data race	fdem_v2.13.f	fdem_v2.13.exe	New
P15	Data race	fdem_v2.13.f	fdem_v2.13.exe	New
P16	Data race	fdem_v2.13.f	fdem_v2.13.exe	New
P17	Data race	fdem_v2.13.f	fdem_v2.13.exe	New
P18	Data race	fdem_v2.13.f	fdem_v2.13.exe	New

The right-hand pane shows filters for the problems, including Severity (Error), Type (Data race), Source (fdem_v2.13.f), Module (fdem_v2.13.exe), State (New), Suppressed (Not suppressed), and Investigated (Not investigated).

The bottom pane shows the "Code Locations: Data race" for the selected problem. It displays the source code snippet for the function `pml_elec_ompparallel@11335` in the module `fdem_v2.13.exe`. The code snippet shows a loop that iterates over `km1` and `j`, with a write operation to `gee_ave, term1, temp`.

```
White fdem_v2.13.f:11343 pml_elec_$omp$parallel@11335 fdem_v2.13.exe
11341 $omp6      gee_ave, term1, temp
11342      do k=kstart(1), kstop(1)
11343          km1=k-1
11344          do j=jstart(1), jminPMLstop
11345              jml=j-1
```