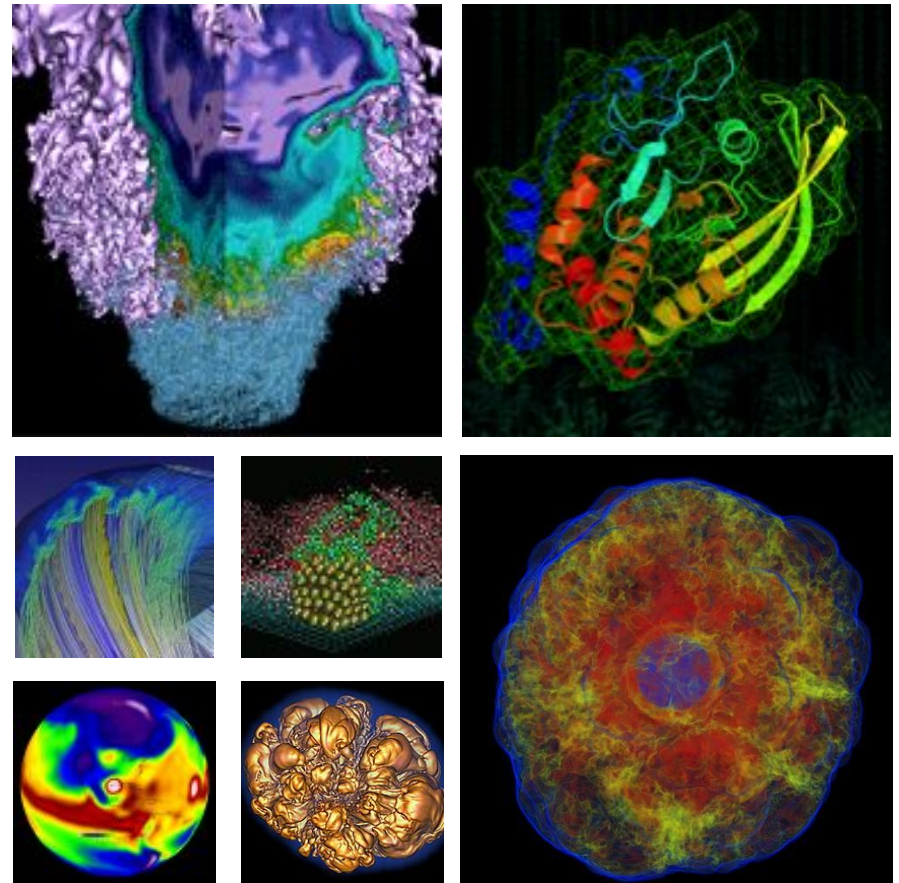


Debugging on Cori and Edison

New User Training
2018



Woo-Sun Yang
User Engagement Group, NERSC

March 21, 2018

Debuggers



- **Fix coding errors for**
 - Wrong results
 - Program crash
 - Program hang
- **How to find them?**
 - Place print statements in what you think strategic locations
 - Difficult to know where the code fails and whether variables have incorrect values
 - Recompile whenever you make a change
 - Tedious and exhausting
 - Using debuggers for your detective work
 - Compile only once (generally)
 - Control execution pace of your program
 - Examine values using debugger's tools
 - Visualization and statistics
 - Can identify where the code fails or hangs

Parallel debuggers on Cori and Edison



- **Parallel debuggers with a graphical user interface**
 - DDT (Distributed Debugging Tool)
 - TotalView
- **Specialized debuggers on Cori and Edison**
 - STAT (Stack Trace Analysis Tool)
 - Collect stack backtraces from all (MPI) tasks
 - ATP (Abnormal Termination Processing)
 - Collect stack backtraces from all (MPI) tasks when an application fails
- **Valgrind**
 - Suite of debugging and profiling tools
 - Best known for its detailed memory debugging (memcheck)
 - <http://www.nersc.gov/users/software/performance-and-debugging-tools/valgrind/>
- **Intel Inspector**
 - Thread and memory debugging
 - <http://www.nersc.gov/users/software/performance-and-debugging-tools/inspector/>
- **Cray debuggers for comparative debugging**
 - CCDB
 - lgdb

Parallel debuggers on Cori and Edison



- **Parallel debuggers with a graphical user interface**

- DDT (Distributed Debugging Tool)
- TotalView

← Today

- **Specialized debuggers on Cori and Edison**

- STAT (Stack Trace Analysis Tool)
 - Collect stack backtraces from all (MPI) tasks
- ATP (Abnormal Termination Processing)
 - Collect stack backtraces from all (MPI) tasks when an application fails

Some Other
Tools: Training
on April 24

- **Valgrind**

- Suite of debugging and profiling tools
- Best known for its detailed memory debugging (memcheck)
- <http://www.nersc.gov/users/software/performance-and-debugging-tools/valgrind/>

- **Intel Inspector**

- Thread and memory debugging
- <http://www.nersc.gov/users/software/performance-and-debugging-tools/inspector/>

- **Cray debuggers for comparative debugging**

- CCDB
- lgdb

DDT and TotalView



- **GUI-based traditional parallel debuggers**
- **Works for C, C++, Fortran programs with MPI, OpenMP, pthreads**
 - DDT supports CAF (Coarray Fortran) and UPC (Unified Parallel C), too
- **Licenses**
 - DDT: up to 4096 MPI tasks on Cori (Haswell and KNL) and Edison
 - TotalView: up to 512 MPI tasks on Cori (Haswell) and Edison
 - Licenses shared among users and machines
- **For info**
 - <https://www.arm.com/products/development-tools/hpc-tools/cross-platform/forg>
 - <http://www.nersc.gov/users/software/debugging-and-profiling/ddt/>
 - <https://www.roguewave.com/products-services/totalview>
 - <http://www.nersc.gov/users/software/debugging-and-profiling/totalview/>

How to build and run with DDT

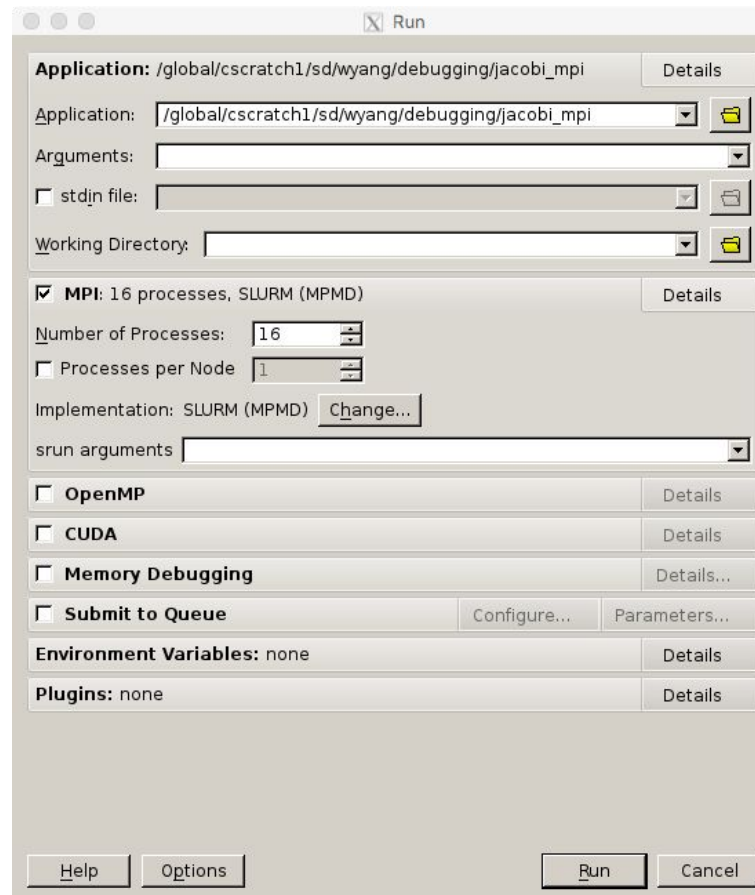


```
$ ftn -g -O0 -o jacobi_mpi jacobi_mpi.f90
```

-g for debugging symbols;
-O0 for the Intel compiler

```
$ salloc -N 1 -t 30:00 -q debug -C knl  
$ module load allinea-forge  
$ ddt ./jacobi_mpi
```

Start an interactive batch session
Load the allinea-forge module to use DDT
Start DDT



If you work far away from NERSC



- Remote X window application (GUI) over network: slow response
- Two solutions
 - Use NX to improve the speed
 - Works for X window applications
 - <https://www.nersc.gov/users/network-connections/using-nx/> (general)
 - http://portal.nersc.gov/project/mpccc/nx/NX_Tutorial/Start_Over.html (installation and quick user guide)
 - Use Arm Forge remote client
 - Runs on your desktop/laptop
 - Submit a debugging batch job from a NERSC machine and make the client **reverse connect** to the job
 - Displays results in real time
 - No license file required on your local desktop/laptop
 - <https://www.nersc.gov/users/software/performance-and-debugging-tools/ddt#toc-anchor-5> (**setup**)
 - <https://developer.arm.com/products/software-development-tools/hpc/downloads/download-arm-forge> (for downloading remote clients)

DDT window



For navigation

Processing entity to control

Sparklines to quickly show variation over MPI tasks

To check the value of a variable, right-click on a variable or check the pane on the right

To evaluate expressions

Parallel stack frame view is helpful in quickly finding out where each process is executing

```
subroutine set_bc(u,n,js,je)
  implicit none
  include 'mpif.h'
  integer n, js, je
  real u(0:n,js-1:je+1)
  integer i, j, joff, np, myid
  real h
  integer ierr

  call mpi_comm_size(mpi_comm_world,np,ierr)
  call mpi_comm_rank(mpi_comm_world,myid,ierr)

  joff = myid * ((n + 1) / np) ! j-index offset

  h = 1.0 / n

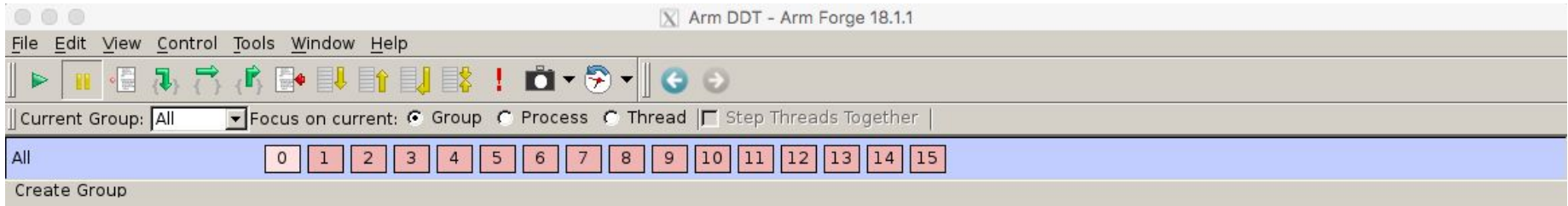
  if (myid == 0) then
    do i=0,n
      u(i,js) = (i * h)**2
    enddo
  endif

  if (myid == np - 1) then
    do i=0,n
      u(i,je) = (i * h)**2 + 1.
    enddo
  endif
end
```

Variable Name	Value
joff	6265
myid	0
n	23999
np	16

Processes	Function
16	jacobi_mpi (jacobi_mpi.f90:45)
16	init_fields (jacobi_mpi.f90:174)
16	set_bc (jacobi_mpi.f90:193)

Navigating in your program



- **Play/Continue**
- **Pause**
- **Add Breakpoint**
- **Step Into**
 - To next line; if it's a function call, enter the function
- **Step Over**
 - To next line in the current stack frame even if it's a function call
- **Step Out**
 - Return to the caller function
- **Run To Line**

Breakpoints, watchpoints and tracepoints



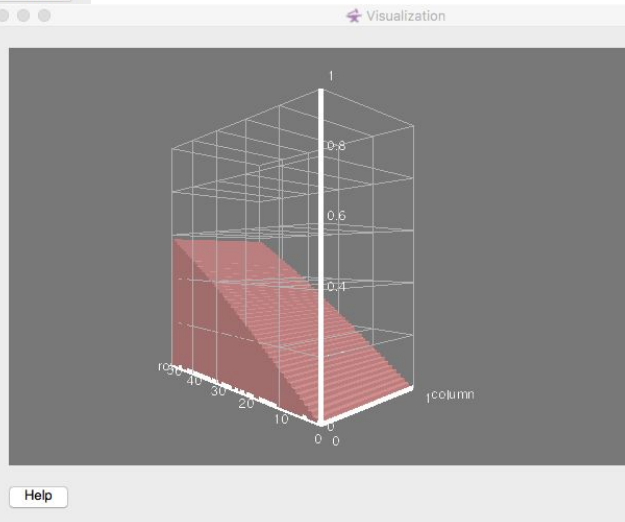
- **Breakpoint**
 - Stops execution when a selected line (breakpoint) is reached
 - Double click on a line to create one; there are other ways, too
- **Watchpoints for variables or expressions**
 - Stops when a variable or an expression changes its value
- **Tracepoints**
 - When reached, prints what lines of codes is being executed and the listed variables
- **Can add a condition for an action point**
 - Useful inside a loop
- **Can be made active or inactive**

Many ways to check variables



- Right click on a variable for a quick summary
- Variable pane
- Evaluate pane
- Display variable values over processes (Compare across processes) or threads (Compare across threads)
- MDA (Multi-dimensional Array) Viewer
 - Visualization
 - Statistics

Index	Value
0	0
1	0.0131118195
2	0.0262203719
3	0.0393223912
4	0.0524146073
5	0.0654937625
6	0.0785565972
7	0.0915998444
8	0.10462027



Statistic	Value
Count	50
Not shown	0
Errors	0
Aggregate	0
Numerical	50
Sum	15.2618
Minimum	0
Maximum	0.580318
Range	0.580318
Mean	0.305235
Variance	0.0304291
nan	0
-nan	0
inf	0
-inf	0
<0	0
=0	1
>0	49



National Energy Research Scientific Computing Center