# Debugging Tools



New User Training June 16, 2020

Woo-Sun Yang User Engagement Group

## Debuggers

• Program errors:

• Program crashes, program hangs, program generates incorrect results, ...

• How to find and fix them?

	Using print statements	Using debuggers
Typical workflow	<ul> <li>Add print statements around suspicious or strategic locations in the source code</li> <li>Compile the code</li> <li>Run the program and examine the printed values to get a hint about what or where the problem may be</li> <li>If no hint is obtained, add different print statements</li> <li>Repeat</li> </ul>	<ul> <li>Start your program under a debugger</li> <li>Set breakpoints in your program</li> <li>Run</li> <li>When the program stops at the breakpoints, check variables</li> <li>Can add more breakpoints and continue</li> </ul>
Pro and con	<ul> <li>"Easy" – no need to learn about a debugging tool</li> <li>Difficult to guess where and what to print</li> <li>Time consuming and tedious <ul> <li>Rebuild the code each time the code is modified</li> <li>Will likely use multiple batch jobs – inefficient use of allocations</li> </ul> </li> <li>Not easy to understand what is wrong from the potentially long printed values (e.g., multi-dim arrays)</li> </ul>	<ul> <li>Compile only once (in general)</li> <li>Control program execution (stop, continue,)</li> <li>Tools and features available to aid to spot problem areas (e.g., visually check for abnormality in variable values by plotting them with the debugger's visualization tool)</li> </ul>

bringing ocience opiations to the world

## Parallel Debuggers Available on Cori

- Parallel debuggers with a graphical user interface (GUI)
  - DDT (Distributed Debugging Tool) part of the Arm Forge tool
  - TotalView
- Specialized debuggers
  - STAT (Stack Trace Analysis Tool) 0
    - Collect stack backtraces from all (MPI) tasks
  - ATP (Abnormal Termination Processing) 0
    - STAT invoked when an application fails
  - Valgrind 0
    - Suite of debugging and profiling tools
    - Best known for its detailed memory debugging tool, 'memcheck'
    - https://docs.nersc.gov/development/performance-debugging-tools/valgrind/
  - Intel Inspector 0
    - Threading and memory debugging



https://docs.nersc.gov/programming/performance-debugging-tools/inspector/





Science

#### **DDT and TotalView**

- GUI-based traditional parallel debuggers
- C, C++, Fortran codes with MPI, OpenMP, pthreads
- Licenses
  - DDT: up to 4,096 processes
  - TotalView: up to 512 processes
  - Shared among users and machines
- For info:
  - <u>https://developer.arm.com/docs/101136/latest/arm-forge</u>
  - o <u>https://docs.nersc.gov/development/performance-debugging-tools/ddt/</u>
  - <u>https://docs.roguewave.com/en/totalview/current/html/</u>
  - <u>https://docs.nersc.gov/development/performance-debugging-tools/totalview/</u>





#### How to Build and Run with DDT

 Compile with -g for debugging symbols and -00 for no optimization (Intel compiler)

```
$ ftn -g -00 -o jacobi_mpi jacobi_mpi.f90
```

• Start an interactive batch job and run DDT:

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

	🔀 Run		
Application:	/global/cscratch1/sd/wyang/debug	ging/jacobi_mpi	Details
Application:	/global/cscratch1/sd/wyang/debug	ging/jacobi_mpi	
Arguments:			•
🗖 std <u>i</u> n file:			<b>T</b>
Working Direc	tory:		
🔽 MPI: 16 рг	rocesses, SLURM (MPMD)		Details
Number of Pro	ocesses: 16 🛨		
🗖 Processes	per Node 📔 🚊		
Implementatio	on: SLURM (MPMD) C <u>h</u> ange		
srun argumer	nts 🗌		•
□ OpenMP			Details
CUDA			Details
Memory I	Debugging		Details
🗆 Submit to	o Queue	Configure	Parameters
Environment	t Variables: none		Details
Plugins: non	e		Details
Help	Options	<u><u>P</u>u</u>	n Cancel
	KELEY LAB	S. DEPARTMENT OF	Office of Science



#### If You Work Far Away From NERSC

- Running X11 GUIs over network: it responses painfully slowly due to intrinsically high latency and inefficient bandwidth between X11 client and server
- Two solutions
  - Use NoMachine (NX) to improve the speed
    - Works for X11 window applications
    - <u>https://docs.nersc.gov/connect/nx/</u>
  - Use Arm Forge remote client
    - Run on your desktop/laptop
    - Submit a debugging batch job on a NERSC machine and make the job connect to the client ("reverse connect")
    - Display results in real time
    - <u>https://developer.arm.com/tools-and-software/server-and-hpc/downloads/arm-forge</u> (for downloading remote clients)
    - https://docs.nersc.gov/programming/performance-debugging-tools/ddt/#reverse-connect-using-remote-client (for setup)







#### Arm Forge Remote Client Settings

• • •	📟 Remote Launch Settings	
Connection Name:	cori	
Host Name:	wyang@cori.nersc.gov wyang@cmom02.nersc.gov	
	How do I connect via a gateway (multi-hop)?	
Remote Installation Directory:	/global/common/sw/cray/cnl6/haswell/allinea-forge/default	
Remote Script	/global/common/sw/cray/cnl6/haswell/allinea-forge/remote-init	
	Always look for source files locally	
KeepAlive Packets:	Enable	
Interval:	30 seconds	0
Uncheck —	Proxy through login node	
for MFA		Test Remote Launch
Help		OK Cancel

• See

https://docs.nersc.gov/development/performance-debugging-tools/ddt/ .....

7





Science

BERKELE)

Bringing Science Solutions to the World

#### **DDT Window**



Bringing Science Solutions to the World

Office of

Science



#### 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





#### **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 Multi-Dimensional Array Viewer threads) Array Expression: uNorth[\$i] Evaluate

Cancel

Auto-update

Align Stack Frames

Distributed Array Dimensions: None 3 How do I view distributed arrays?

Staggered Array What does this do?

- MDA (Multi-Dimensional Array) Viewer
  - Visualization

Statistics     Quick sanity check, fo     ★ Multi-Dimensional Array Viewer	r ex., after halo exchange	Range of \$i From: 0 C To: 49 C Display: Rows
Array Expression:       uNorth[\$1]         Distributed Array Dimensions:       None C         How do I view distributed arrays2         Staggered Array What does this do2	Evaluate Cancel Nign Star Auto-upda	Only show if: See Examples Data Table Statistics
Range of \$i         From:       0         To:       49         Display:       Rows         Only show if:       See Examples         Only show if:       Data Table         Statistics       Data Table         Statistics       1         0.0131118195       2         0.00322203719       3         0.00322203719       5         0.00524146073       5         0.0054347625       6         0.01915098444       8         0.10462027       0.0915098444		Pri 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: 0.305235 Variance: 0.3052491 nan: 0 -nan: 0 inf: 0 -inf: 0 -inf: 0 -junt: 0 -jun
Help	Help	Help Close

#### **TotalView**

- Start a batch job interactively and run your code with TotalView
  - \$ salloc -N 1 -C knl -t 30:00 -q debug \$ module load totalview \$ export OMP\_NUM\_THREADS=4 \$ totalview srun -a \ -n 8 -c 32 --cpu-bind=cores ./jacobi mpiomp
- Click 'OK' in the 'Startup Parameters srun' window
- Click 'Go' in the main window



PARTMENT OF

ERC

No

Office of

Science

Click 'Yes' to the question 'Process srun is a parallel job.
 Do you want to stop the job now?'



### TotalView (cont'd)

#### Root window

Eile Edit View Tools			
Process State	Procs	Threads	Members
Running	1	1	р1
Ė- <unknown address=""></unknown>	1	8	p1.1-8
-1.1	1	1	p1.1
-1.2	1	1	p1.2
-1.3	1	1	p1.3
-1.4	1	1	p1.4
-1.5	1	1	p1.5
-1.6	1	1	p1.6
-1.7	1	1	p1.7
L_1.8	1	1	p1.8
i Breakpoint	8	8	0-7
Ėinit	1	1	0.2
L-2.2	1	1	0.2
🗄 jacobi_mpiomp	8	8	0-7.1
-2.1	1	1	0.1
-3.1	1	1	1.1
-4.1	1	1	2.1
-5.1	1	1	3.1
-6.1	1	1	4.1
-7.1	1	1	5.1
-8.1	1	1	6.1
-9.1	1	1	7.1

State of MPI tasks and threads; members denoted roughly as 'rank.thread'



#### **Process window**

File Edit, View Goog nav	igation	Tebus Tools Window
Group (Control)	ill Restart Next Step D	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Thread 1	Rank 7: srunkjacobi_mp. (0x2aaaaab32b80/58223)	iomp).7 (At Breakpoint 2) : jacobi_mpiomp (At Breakpoint 2)
Stack Trace	no percentino Feli	Stack Frame
<pre>[190] Jacobi_mpiomp, main, libc_start_main, start,</pre>	FP=7fffffff5540 FP=7fffffff5600 FP=7fffffff5608	<pre>%rip: 0x2000a723 (jacobi_mpiomp+0xeb3) %fs_base: 0x2aaaaab32b80 (46912496675712) %gs_base: 0x000000000 (0) %temp: 0xffffffffffffffffffffffffffffffffffff</pre>
	Fl	oating point registers:
		ara+0. 0
		%st1: 0
		%st2: 0 %st3: 0
Fri	[[M] nction jacobi mpione in	tacobi spices £90
62 0.25 * ( u(: 63 + u(i,j+1) + 1	i+1,j) + u(i-1,j) & u(i,j-1) &	
V-1		
65         unew(i, j) = omega           66         enddo           67         enddo           68         !\$omp end parallel do           69         call set_bc(unew, n, js, j)           71         call set_bc(unew, n, js, j)           72         ! Compute the difference           73	* utmp + (1 omega) * je) between unew and u.	To see the value of a variable, right-click on
65     unew(i,j) = omega       66     enddo       67     enddo       68     !\$omp end parallel do       69     70       71     call set_bc(unew, n, js, ;       71     71       72     ! Compute the difference       73     call compute_diff(u, unit)	* utmp + (1 omega) * je) between unew and u. Dive	To see the value of a variable, right-click on
65     unew(i,j) = omega       66     enddo       67     enddo       68     !Somp end parallel do       69     70       71     call set_bc(unew, n, js, ')       72     ! Compute the difference       73     call compute_diff(u,ung       75     if (myid == 0) print *	* utmp + (1 omega) * je) between unew and u. Dive	To see the value of a variable, right-click on a variable to "dive" on
65     unew(i, j) = omega       66     enddo       67     enddo       68     !Somp end parallel do       69     70       71     call set_bc (unew, n, js, jr)       72     ! Compute the difference       73     call compute_diff (u, ug)       75     if (myid == 0) print *       76     if (myid == 0) print the	* utmp + (1 omega) * je) between unew and u. Dive Add to Expression List	To see the value of a variable, right-click on a variable to "dive" on it or just hover mouse
65       unew(i, j) = omega         66       enddo         67       enddo         68       !\$omp end parallel do         69       70         71       call set_bc(unew, n, js, j)         72       ! Compute the difference         73       call compute_diff(u, units)         75       if (myid == 0) print *         76       if (myid == 0) print *         78       ! Make the new value the	* utmp + (1 omega) * je) between unew and u. Dive Add to Expression List Across Processes	To see the value of a variable, right-click on a variable to "dive" on it or just hover mouse
65     unew(i, j) = omega       66     enddo       67     enddo       68     !\$omp end parallel do       70     call set_bc (unew, n, js, j)       71     72       72     ! Compute the difference       73     call compute_diff(u,ug)       76     if (myid == 0) print *       77     78       1     Make the new value the       79     30       81     ! Somp parallel do	* utmp + (1 omega) * je) between unew and u. Dive Add to Expression List Across Processes Across Threads	To see the value of a variable, right-click on a variable to "dive" on it or just hover mouse over it
65         unew(i, j) = omega           66         enddo           67         enddo           68         !\$omp end parallel do           69         call set_bc (unew, n, js, j)           70         call set_bc (unew, n, js, j)           71         72           72         Compute the difference           73         call compute_diff(u,ug)           76         if (myid == 0) print *           77         ?6           78         ! Make the new value the           79         isomp parallel do           81         do j=js-1, je+1           92         u(;,j) = unew(:,j)	* utmp + (1 omega) * je) between unew and u. Dive Add to Expression List Across Processes Across Threads Set Breakpoint	To see the value of a variable, right-click on a variable to "dive" on it or just hover mouse over it
65       unew(i, j) = omega         66       enddo         67       enddo         68       !\$omp end parallel do         69       call set_bc (unew, n, js, j)         71       72         72       ! Compute the difference         73       call compute_diff(u,ugg         75       if (myid == 0) print *         77       76         78       ! Make the new value the         79       do j=js-1, je+1         80       isomp parallel do         81       some parallel do         82       !Somp parallel do         84       !Somp ond parallel do	* utmp + (1 omega) * je) between unew and u. Dive Add to Expression List Across Processes Across Threads Set Breakpoint Set Barrier	To see the value of a variable, right-click on a variable to "dive" on it or just hover mouse over it
65       unew(i, j) = omega         66       enddo         67       enddo         68       !\$omp end parallel do         69       call set_bc (unew, n, js, j)         71       72         72       ! Compute the difference         73       call compute_diff(u,umerce)         75       if (myid == 0) print *         77       ?8         80       expanded do         81       !Somp parallel do         81       somp parallel do         81       somp parallel do         82       u(i,j) = unew(:,j)         83       end do         84       !\$omp end parallel do	* utmp + (1 omega) * je) between unew and u. Dive Add to Expression List Across Processes Across Threads Set Breakpoint Set Barrier Create Watchpoint	To see the value of a variable, right-click on a variable to "dive" on it or just hover mouse over it
65       unew(i, j) = omega         66       enddo         67       enddo         68       !\$omp end parallel do         69       call set_bc (unew, n, js, jange)         71       call compute the difference         72       ! Compute the difference         73       call compute_diff(u,ume)         75       if (myid == 0) print *         77       ??         80       !\$omp parallel do         81       !\$omp parallel do         81       !\$omp end parallel do         82       !\$omp end parallel do         84       !\$omp end parallel do         53       end do         84       !\$omp end parallel do	* utmp + (1 omega) * je) between unew and u. Dive Add to Expression List Across Processes Across Threads Set Breakpoint Set Barrier Create Watchpoint Enable	To see the value of a variable, right-click on a variable to "dive" on it or just hover mouse over it
65       unew(i, j) = omega         66       enddo         67       enddo         68       !\$omp end parallel do         69       call set_bc(unew, n, js,         70       call set_bc(unew, n, js,         71       call compute the difference         73       call compute_diff(u.um         75       if (myid == 0) print *         77       rs       ! Make the new value the         79       end do       do         80       !\$omp parallel do       do         81       !Somp parallel do       disj=j=-1, j=+1         82       u(:,j) = unew(:,j)       end do         84       !\$omp end parallel do       disj=u(i,j) = unew(:,j)         83       end do       send do         84       !\$omp end parallel do       disj=u(i,j) = unew(:,j)         84       !\$ome one jarallel do       disj=u(i,j) = unew(:,j)         84       !\$ome one jarallel do       disj=u(i,j) = unew(:,j)         85       !       'iiiiiii	* utmp + (1 omega) * je) between unew and u. Dive Add to Expression List Across Processes Across Threads Set Breakpoint Set Barrier Create Watchpoint Enable Disable	To see the value of a variable, right-click on a variable to "dive" on it or just hover mouse over it
65       unew(i, j) = omega         66       enddo         67       enddo         68       !Somp end parallel do         69       call set_bc(unew, n, js,         70       call set_bc(unew, n, js,         71       call compute the difference         73       call compute_diff(u.une         75       if (myid == 0) print *         77       76       if (myid == 0) print *         77       81       Make the new value the         79       80       !Somp parallel do         80       isomp parallel do       do         81       !Somp end parallel do       send do         84       !Somp end parallel do       send do         84       !Somp end parallel do       send do         82	* utmp + (1 omega) * je) between unew and u. Dive Add to Expression List Across Processes Across Threads Set Breakpoint Set Barrier Create Watchpoint Enable Disable Delete	To see the value of a variable, right-click on a variable to "dive" on eratic it or just hover mouse over it jacobi_mpiomp+0xeb3 For selecting
65 66 67 68 68 69 70 69 70 60 71 72 72 72 73 72 73 73 74 75 75 75 76 75 76 76 76 77 76 77 78 78 78 78 78 78 78 78 78 78 78 78	* utmp + (1 omega) * je) between unew and u. Dive Add to Expression List Across Processes Across Threads Set Breakpoint Set Barrier Create Watchpoint Enable Disable Delete Properties	To see the value of a variable, right-click on a variable to "dive" on eratic it or just hover mouse over it jacobi_mpiomp+0xeb3 For selecting

### STAT (Stack Trace Analysis Tool)

- Gathers stack backtraces (sequence of function calls leading up to the current function) from all (MPI) processes
  - Merge them into a single file (\*.dot)
  - Results displayed as a single call tree for all processes
  - Can be useful for debugging a hanging application
  - With the info learned from STAT, can investigate further with DDT or TotalView
- Works for MPI, CAF, UPC, and OpenMP





## STAT (Stack Trace Analysis Tool) (cont'd)

- STAT commands (after loading the 'stat' module)
  - stat-cl: invokes STAT to gather stack backtraces
  - STATview: A GUI to view the results
  - STATGUI: a GUI to run STAT or view results
- For more info:
  - 'intro\_stat', 'STAT', 'STATview' and 'STATGUI' man pages
  - /opt/cray/pe/stat/default/doc/stat\_userguide.pdf
  - -<u>https://docs.nersc.gov/development/performance-debugging-too</u>
     <u>ls/stat\_atp/</u>





#### Debug a Hanging App with STAT

• If your code hangs consistently, use STAT to examine whether MPI processes get stuck.

```
With usual optimization flags, if any
$ ftn -q -o jacobi mpi jacobi mpi.f90
$ salloc -N 1 -t 30:00 -g debug -C knl
$ srun -n 4 -c 64 --cpu-bind=cores ./jacobi mpi &
[1] 135543
S module load stat
                                                      -i to get source line numbers
$ stat-cl -i 135543
                                                      STAT samples stack backtraces 10 times
. . .
Attaching to application...
Attached!
Application already paused... ignoring request to pause
Sampling traces...
Traces sampled!
. . .
Resuming the application ...
Resumed!
Merging traces...
Traces merged!
Detaching from application...
Detached!
```

Results written to /global/cscratch1/sd/wyang/debugging/stat/stat\_results/jacobi\_mpi.0003

\$ ls -l stat\_results/jacobi\_mpi.0003/\*.dot -rw-rw---- 1 wyang wyang 5201 Jun 7 14:55 stat\_results/jacobi\_mpi.0003/00\_jacobi\_mpi.0003.3D.dot \$ STATview stat\_results/jacobi\_mpi.0003/00\_jacobi\_mpi.0003.3D.dot

15





Office of

Science

#### Debug a Hanging App with STAT (cont'd)



Office of Science

# ATP (Abnormal Termination Processing)

- ATP invokes STAT wgeb the application fails
  - Output in atpMergedBT.dot and atpMergedBT\_line.dot (showing source line numbers), which are to be viewed with STATview
- The atp module is loaded on Cori by default, but ATP is <u>not</u> enabled; to enable:
  - \$ export ATP\_ENABLED=1 # sh/bash/ksh
  - % setenv ATP\_ENABLED 1 # csh/tcsh
- For more info
  - 'intro\_atp' man page
  - <u>https://docs.nersc.gov/development/performance-debugging-tools/stat\_</u>





## Debug a Hanging App with ATP

#### Submit a hanging job with ATP enabled

```
$ ftn -g -o jacobi_mpi jacobi_mpi.f90
$ cat runit
#!/bin/bash
#SBATCH -N 1
#SBATCH -C knl
...
export ATP_ENABLED=1
export FOR_IGNORE_EXCEPTIONS=true
srun -n 4 -c 64 --cpu-bind=cores ./jacobi_mpi
```

```
# Enable ATP# Code built with Intel fortran compiler
```

#### \$ sbatch runit

Submitted batch job 31445729





# Debug a Hanging App with ATP (cont'd)

From a login node, ssh to a MOM node and cancel the srun job

\$ ssh cmom02

\$ sacct -j 31445729							
JobID JobName	Partition	Account	AllocCPUS	State	ExitCode		
31445729 runit	debug_knl	nstaff	272	RUNNING	0:0		
31445729.0 jacobi_mpi		nstaff	256	RUNNING	0:0		
31445729.1 cti_dlaun+		nstaff	1	RUNNING	0:0		
\$ scancel -s ABRT <b>31445729.0</b>							
\$ logout							

#### • Dot files are generated; view them with STATview

\$ ls -l \*.dot
-rw-rw---- 1 wyang wyang 1287 Jun 7 15:31 atpMergedBT.dot
-rw-rw---- 1 wyang wyang 1837 Jun 7 15:31 atpMergedBT\_line.dot
\$ module load stat

\$ STATview atpMergedBT\_line.dot





#### Debug a Hanging App with ATP (cont'd)



Office of Science

## Arm Tools Tutorial on July 16, 2020!

- <sup>1</sup>⁄<sub>2</sub>-day tutorial for Arm tools
  - Arm Forge
    - DDT debugger
    - MAP performance profiling
  - Performance Reports: performance summary
- Beginning/Intermediate level
- Will teach how to profile Python apps, too
- By Arm engineer
- Info and registration:

https://www.nersc.gov/users/training/events/arm-debugging-and-profiling-t

21

NERSC pols-tutorial-june-25-2020/







Thank You and Welcome to NERSC!

