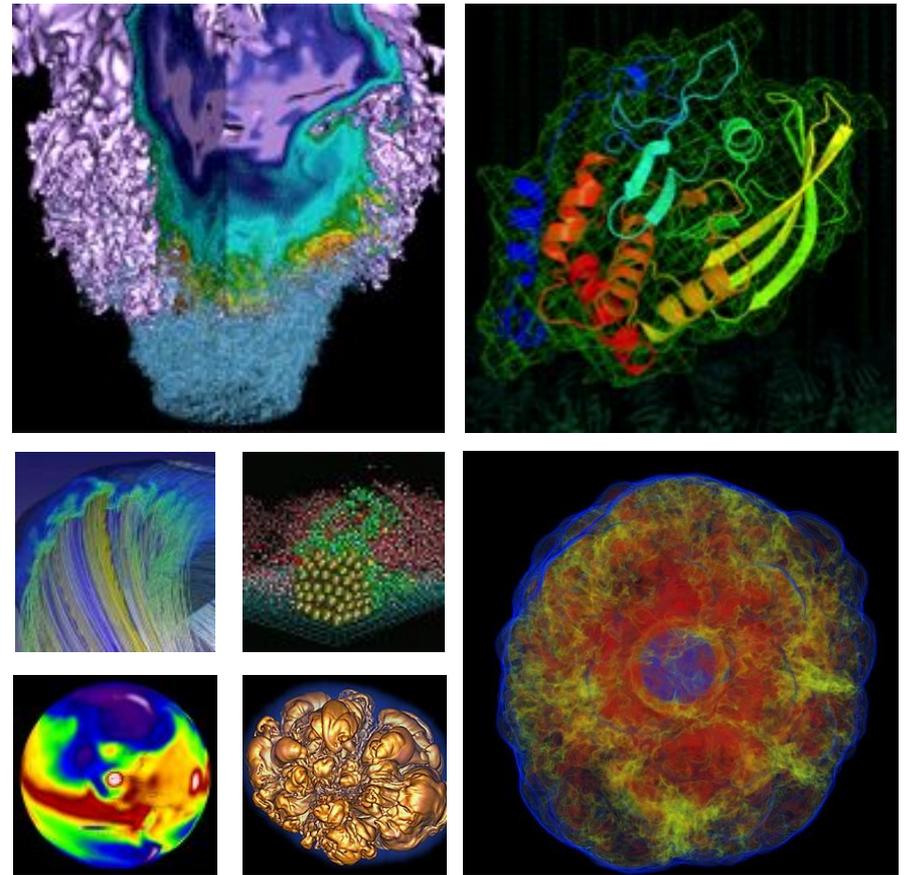


Debugging Tools New User Training



Woo-Sun Yang
User Engagement Group, NERSC

January 25, 2019

- **Program errors**
 - Program crashes
 - Program hangs
 - Wrong results
- **How to find and fix them?**
 - Print statements
 - Difficult to guess where to put them and what to print
 - Recompile whenever you change them
 - Tedious and exhausting, especially for parallel codes
 - Debuggers
 - Compile only once (generally)
 - Control execution of your program
 - Check variables
 - Identify where the code fails or hangs and why

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)
 - <https://docs.nersc.gov/development/performance-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**
- **C, C++, Fortran codes with MPI, OpenMP, pthreads**
- **Licenses**
 - DDT: up to 8192 MPI tasks on Cori and Edison
 - TotalView: up to 512 MPI tasks on Cori and Edison
 - Shared among users and machines
- **For info**
 - <https://developer.arm.com/products/software-development-tools/hpc/arm-forge>
 - <https://docs.nersc.gov/development/performance-debugging-tools/ddt/>
 - <https://www.roguewave.com/products-services/totalview>
 - <https://docs.nersc.gov/development/performance-debugging-tools/totalview/>

How to build and run with DDT

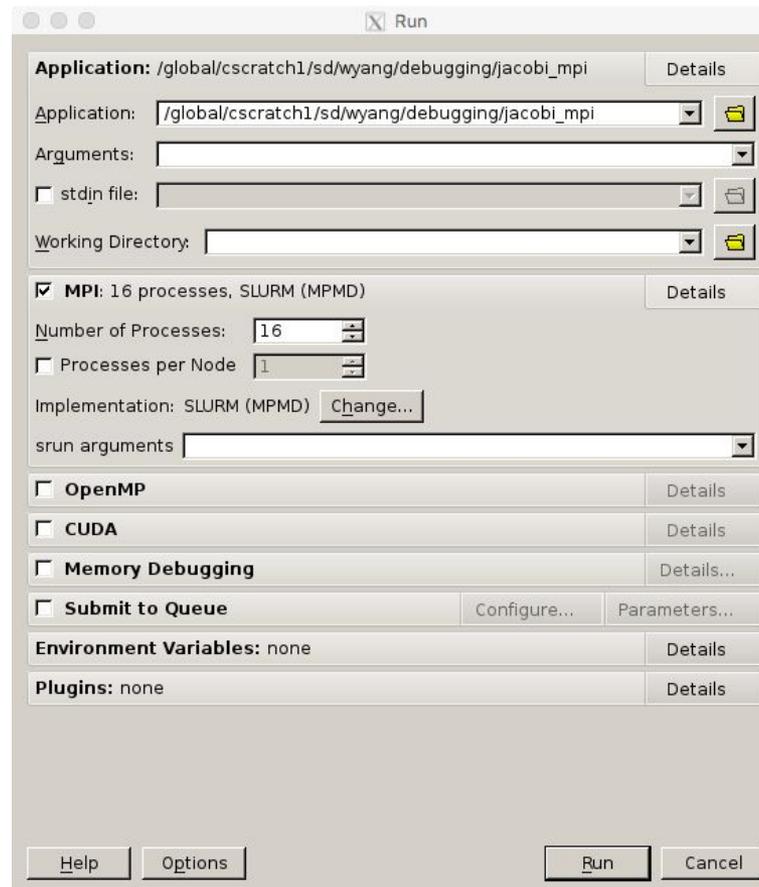


```
$ ftn -g -O0 -o jacobi_mpi jacobi_mpi.f90
```

-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 X11 window GUI application over network: painfully slow response**
- **Two solutions**
 - Use NX (NoMachine) to improve the speed
 - Works for X window applications
 - <https://docs.nersc.gov/connect/nx/>
 - 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**”)
 - Displays results in real time
 - <https://docs.nersc.gov/development/performance-debugging-tools/ddt/#reverse-connect-using-remote-client> (for setup)
 - <https://developer.arm.com/products/software-development-tools/hpc/downloads/download-arm-forge> (for downloading remote clients)

Arm Forge remote client settings



- Uncheck the 'Proxy through login node' box -- for MFA authentication

The screenshot shows a 'Remote Launch Settings' dialog box with the following fields and options:

- 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
- Proxy through login node

Buttons: Help, Test Remote Launch, OK, Cancel

DDT window



For navigation

File Edit View Control Tools Window Help

Current Group: All Focus on current: Group Process Thread Step Threads Together

Processing entity to control

Project Files

Search (Ctrl+K)

Application Code

- /
- Sources
 - jacobi_mpi.f90
 - compute_diff
 - get_indices
 - init_fields
 - jacobi_mpi
 - read_params
 - set_bc
- External Code

```
180 subroutine set_bc(u,n,js,je)
181
182 implicit none
183 include 'mpif.h'
184 integer n, js, je
185 real u(0:n,js-1:je+1)
186 integer i, j, joff, np, myid
187 real h
188 integer ierr
189
190 call mpi_comm_size(mpi_comm_world,np,ierr)
191 call mpi_comm_rank(mpi_comm_world,myid,ierr)
192
193 joff = myid * ((n + 1) / np)
194
195 h = 1.0 / n
196
197 if (myid == 0) then
198 do i=0,n
199 u(i,js) = (i * h)**2
200 enddo
201 endif
```

Locals Current Line(s) Current Stack

Current Line(s)

Name	Value
joff	416475
myid	0
n	28999
np	16

Sparklines

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

Input/Output Breakpoints Watchpoints Stacks Tracepoints Tracepoint Output Logbook

Stacks

Processes	Threads	Function
16	16	jacobi_mpi (jacobi_mpi.f90:45)
16	16	init_fields (jacobi_mpi.f90:174)
16	16	set_bc (jacobi_mpi.f90:193)
2	2	nem_gni_error_handler

Evaluate

Name	Value
(n+1) / np	1500

To evaluate expressions

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

Breakpoints, watchpoints and tracepoints



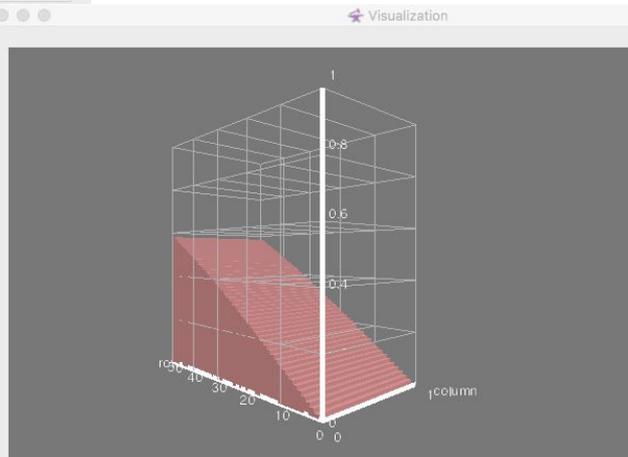
- **Breakpoint**
 - Stops execution when a selected line (breakpoint) is reached
 - Double click on a line to create one; there are other ways, too
- **Watchpoints for variables or expressions**
 - Stops when a variable or an expression changes its value
- **Tracepoints**
 - When reached, prints what 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 threads)
- MDA (Multi-dimensional Array) Viewer
 - Visualization
 - Statistics

Row	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

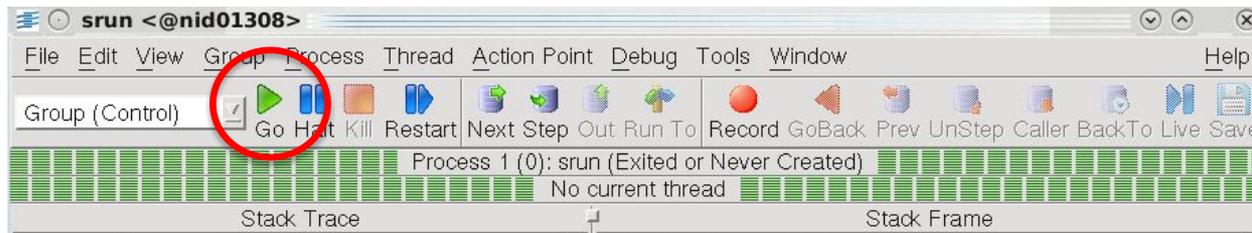
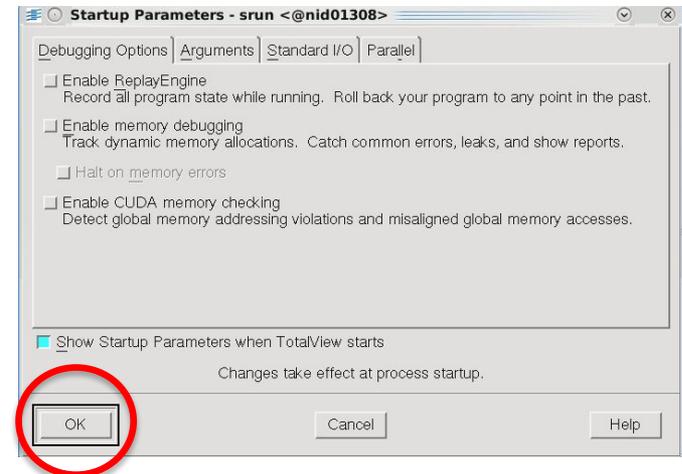
TotalView



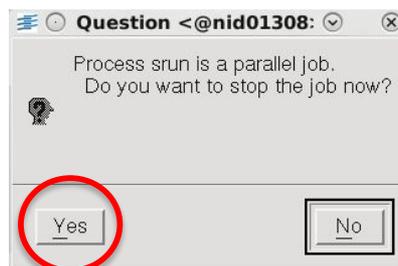
```
$ salloc -N 1 -t 30:00 -q debug
$ module load totalview
$ export OMP_NUM_THREADS=6
$ totalview srun -a -n 4 ./jacobi_mpiomp
```

Then,

- Click OK in the 'Startup Parameters - srun' window
- Click 'Go' button in the main window



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



TotalView (cont'd)



Root window

Process State	Procs	Threads	Members
Running	1	1	p1
<unknown address>	1	4	p1.1-4
1.1	1	1	p1.1
1.2	1	1	p1.2
1.3	1	1	p1.3
1.4	1	1	p1.4
Breakpoint	4	4	0-3
jacobi_mpiomp	4	4	0-3.1
2.1	1	1	0.1
3.1	1	1	1.1
4.1	1	1	2.1
5.1	1	1	3.1
sched_yield	4	20	0-3.2-6
2.2	1	1	0.2
2.3	1	1	0.3
2.4	1	1	0.4
2.5	1	1	0.5
2.6	1	1	0.6
3.2	1	1	1.2
3.3	1	1	1.3
3.4	1	1	1.4
3.5	1	1	1.5
3.6	1	1	1.6
4.2	1	1	2.2

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

Process window

For navigation

Group (Control) [Go] [Halt] [Kill] [Restart] [Next Step] [Out] [Run To] [Record] [GoBack] [Prev] [UnStep] [Caller] [BackTo] [Live] [Save]

Rank 0: run jacobi_mpiomp>.0 (At Breakpoint 1) [M]

Thread 1 (12365696): jacobi_mpiomp (At Breakpoint 1)

Stack Trace

f90	jacobi_mpiomp,	FP=7fffffff65f0
	main,	FP=7fffffff6690
c	__libc_start_main,	FP=7fffffff6750
	_start,	FP=7fffffff6760

Function "jacobi_mpiomp":
No arguments.
Local variables:
ierr: 0 (0x00000000)
nbr_up: 1 (0x00000001)
nbr_down: -1 (0xffffffff)
jel: 5999 (0x0000176f)
js1: 1 (0x00000001)
je: 5999 (0x0000176f)
is: 0 (0x00000000)

Function jacobi_mpiomp in jacobi_mpiomp.f90

```

65      unew(i,j) = omega * utmp + (1. - omega) * u(i,j)
66      enddo
67      enddo
68      !$omp end parallel do
69
70      call set_bc(unew,n,js,je)
71
72      ! Compute the difference between unew and u.
73
74      call compute_diff(u,unew,n,is,ie,diffnorm)
75
76      if (myid == 0) print *,
77
78      ! Make the new value the c
79
80      !$omp parallel do
81      do j=js-1,je+1
82      u(:,j) = unew(:,j)
83      end do
84      !$omp end parallel do
    
```

Context Menu:

- Dive
- Add to Expression List
- Across Processes
- Across Threads
- Set Breakpoint
- Set Barrier
- Create Watchpoint
- Enable
- Disable
- Delete
- Properties

Action Points | Threads

1 jacobi_mpiomp.f90#74 ja

Breakpoints, etc.

To see the value of a variable, right-click on a variable to "dive" on it or just hover mouse over it

For selecting MPI task and thread

STAT (Stack Trace Analysis Tool)

The NERSC logo is a dark blue rectangle with the word "NERSC" in white, bold, sans-serif font. The letters are slightly shadowed, giving it a 3D appearance as if it's floating above a surface.

- **Gathers stack backtraces (sequence of function calls leading up to the current function) for 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 and UPC, OpenMP**

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

Debug a hanging application with STAT



- If your code hangs in a consistent manner, you can use STAT to see whether some MPI ranks got stuck.

```
$ ftn -g -o jacobi_mpi jacobi_mpi.f90
$ salloc -N 2 -t 30:00 -q debug -C knl
```

with usual optimization flags, if any

```
...
$ srun -n 4 ... ./jacobi_mpi &
```

```
[1] 158190
```

```
$ module load stat
```

```
$ stat-cl -i 158190
```

```
...
Attaching to application...
```

-i to get source line numbers

```
Attached!
```

```
Application already paused... ignoring request to pause
```

STAT samples stack backtraces a few times

```
Sampling traces...
```

```
Traces sampled!
```

```
...
Resuming the application...
```

```
Resumed!
```

```
Merging traces...
```

```
Traces merged!
```

```
Detaching from application...
```

```
Detached!
```

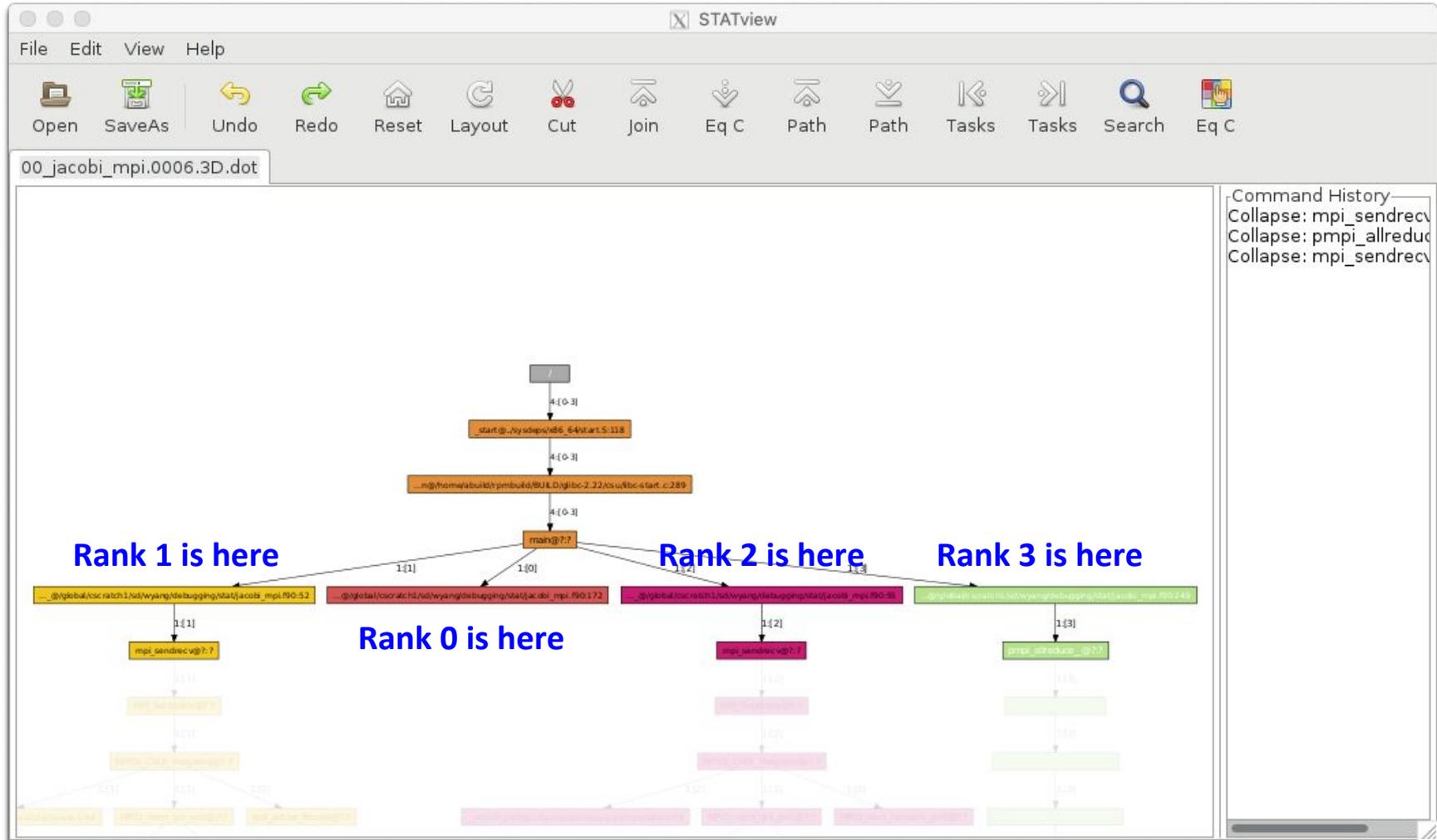
```
Results written to /global/cscratch1/sd/wyang/debugging/stat/stat_results/jacobi_mpi.0006
```

```
$ ls -l stat_results/jacobi_mpi.0006/*.dot
```

```
-rw-r--r-- 1 wyang wyang 4855 Nov  6 00:58 stat_results/jacobi_mpi.0006/00_jacobi_mpi.0006.3D.dot
```

```
$ STATview stat_results/jacobi_mpi.0006/00_jacobi_mpi.0006.3D.dot
```

Debug a hanging application with STAT (Cont'd)



Cray ATP (Abnormal Termination Processing)



- **ATP gathers stack backtraces from all processes *when an application fails***
 - Invokes STAT underneath
 - Output in atpMergedBT.dot and atpMergedBT_line.dot (which shows source code line numbers), which are to be viewed with STATview
- **The atp module is loaded on Cori and Edison by default, but ATP is not enabled; to enable:**

```
export ATP_ENABLED=1      # sh/bash/ksh
setenv ATP_ENABLED 1      # csh/tcsh
```
- **For more info**
 - ‘intro_atp’ man page
 - https://docs.nersc.gov/development/performance-debugging-tools/stat_atp/



**National Energy Research Scientific Computing
Center**