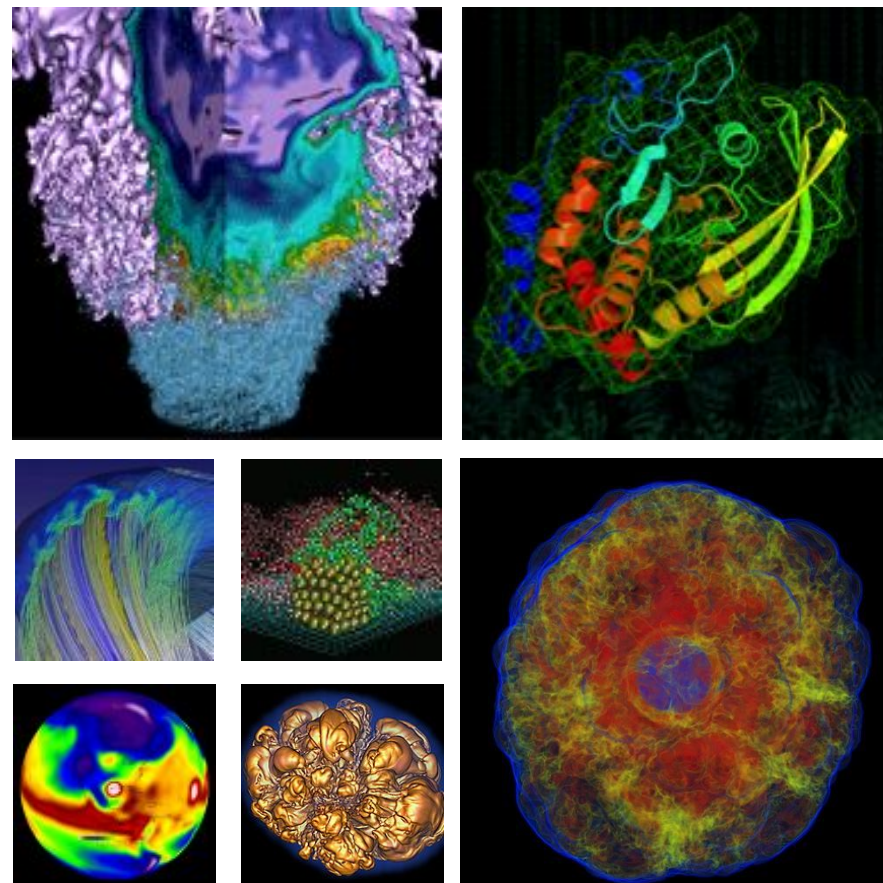


Debugging Tools

New User Training



Woo-Sun Yang
User Engagement Group, NERSC

June 21, 2019

- **Program errors**
 - Program crashes
 - Program hangs
 - Wrong results

- **How to find and fix them?**
 - Print statements
 - Difficult to guess where and what to print
 - Recompile and submit jobs whenever you change them
 - Tedious, exhausting and time-consuming
 - Hard to extract info from output about the error, especially for parallel codes
 - Debuggers
 - Compile only once (generally)
 - Control execution of your program
 - Check variables; visualize and get stats

Parallel debuggers on Cori



- **Parallel debuggers with a graphical user interface**
 - DDT (Distributed Debugging Tool)
 - TotalView
- **Specialized debuggers**
 - 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 tool, 'memcheck'
 - <https://docs.nersc.gov/development/performance-debugging-tools/valgrind/>
 - Intel Inspector
 - Threading and memory debugging
 - <https://docs.nersc.gov/programming/performance-debugging-tools/inspector/>

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
 - TotalView: up to 512 MPI tasks on Cori
 - Shared among users and machines
- **For info**
 - <https://developer.arm.com/tools-and-software/server-and-hpc/arm-architecture-tools/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
```

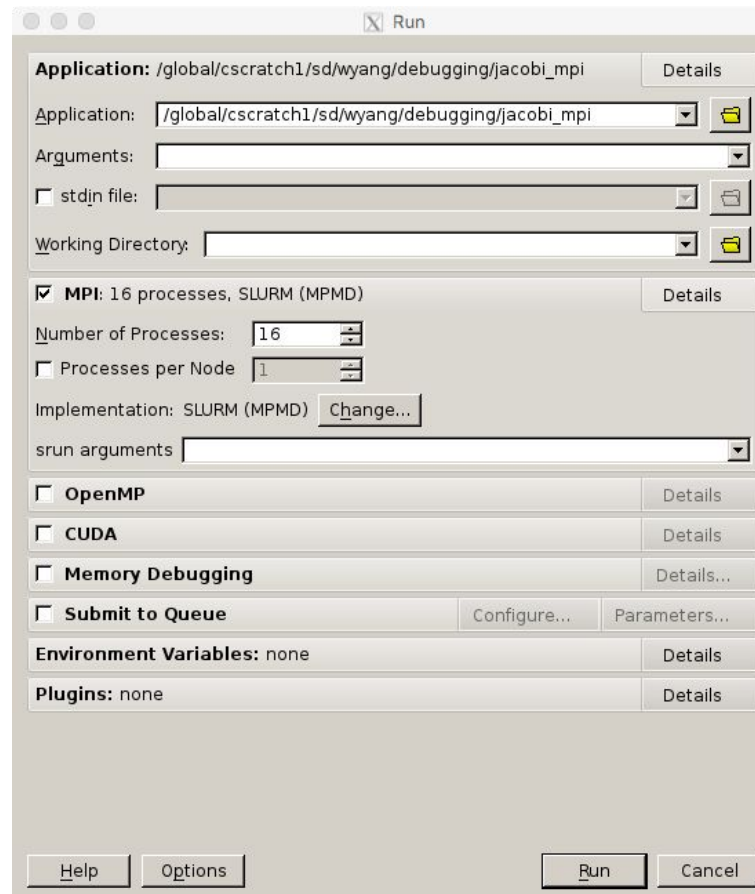
-g for debugging symbols;
-O0 for the Intel compiler

```
$ 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



- Running X11 GUIs over network: responses painfully slow due to intrinsically high latency and inefficient bandwidth between X11 client and server
- Two solutions
 - Use NX (NoMachine) to improve the speed
 - Works for X11 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://developer.arm.com/tools-and-software/server-and-hpc/arm-architecture-tools/downloads/download-arm-forge> (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



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

Uncheck for MFA → ☐ Proxy through login node

Test Remote Launch

Help OK Cancel

- See

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

DDT window



For navigation

Processing entity to control

Sparklines

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

Breakpoints, watchpoints and tracepoints

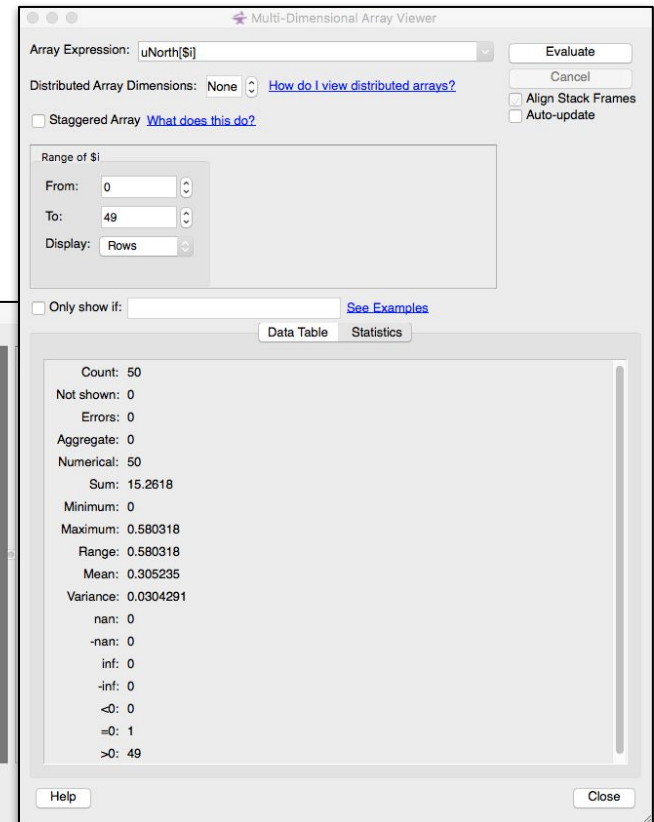
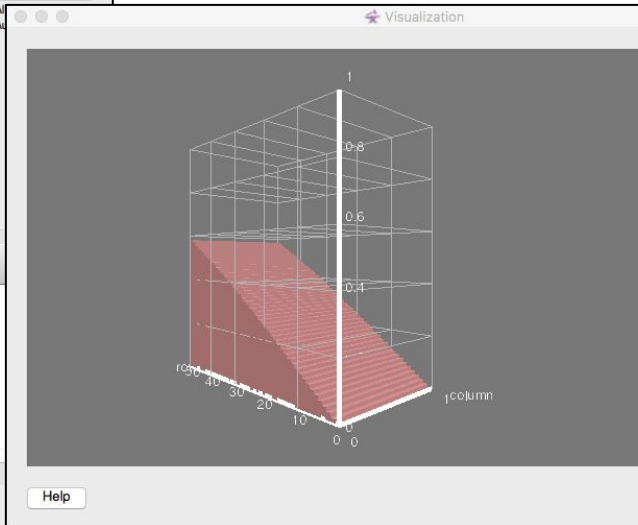
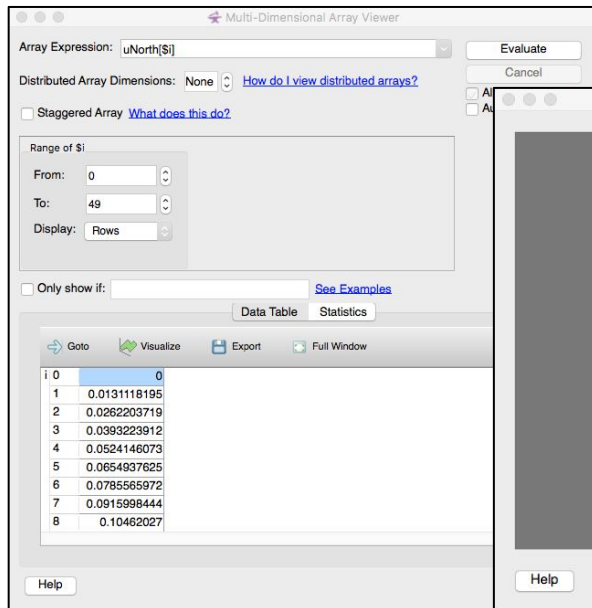


- **Breakpoint**
 - Stops execution when a selected line (breakpoint) is reached
 - Double click on a line to create one; there are other ways, too
- **Watchpoints for variables or expressions**
 - Stops when a variable or an expression changes its value
- **Tracepoints**
 - When reached, prints what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



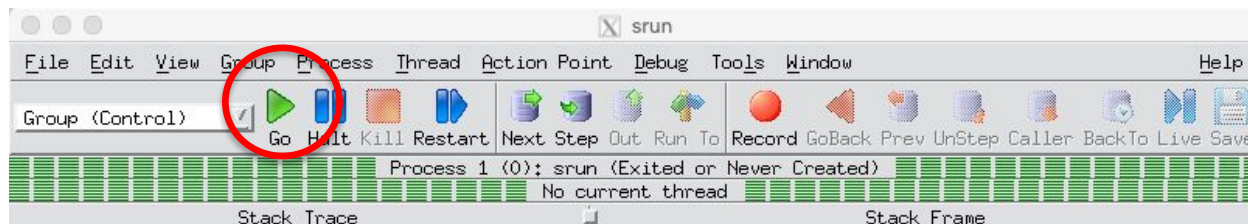
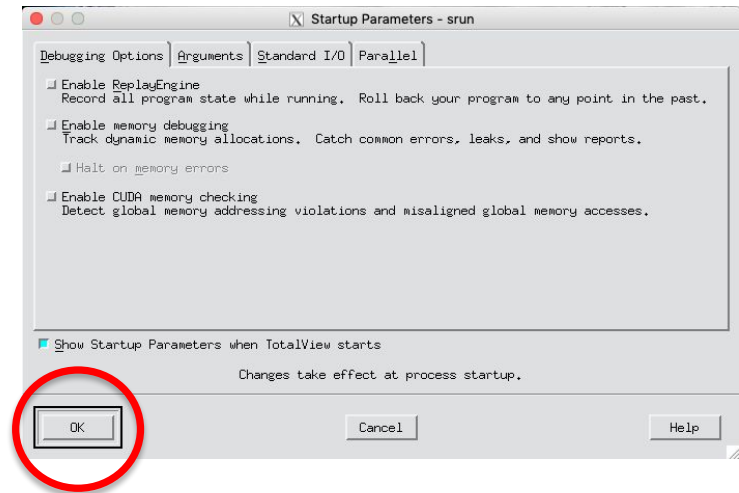
TotalView



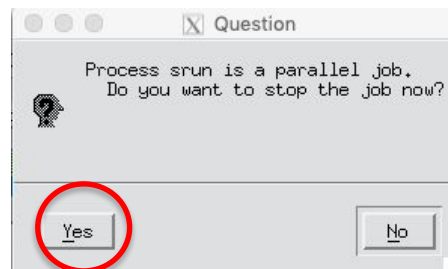
```
$ salloc -N 1 -C knl -t 30:00 -q debug
$ module load totalview
$ export OMP_NUM_THREADS=4
$ totalview srun -a \
    -n 8 -c 32 --cpu_bind=cores ./jacobi_mpiomp
```

Then,

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



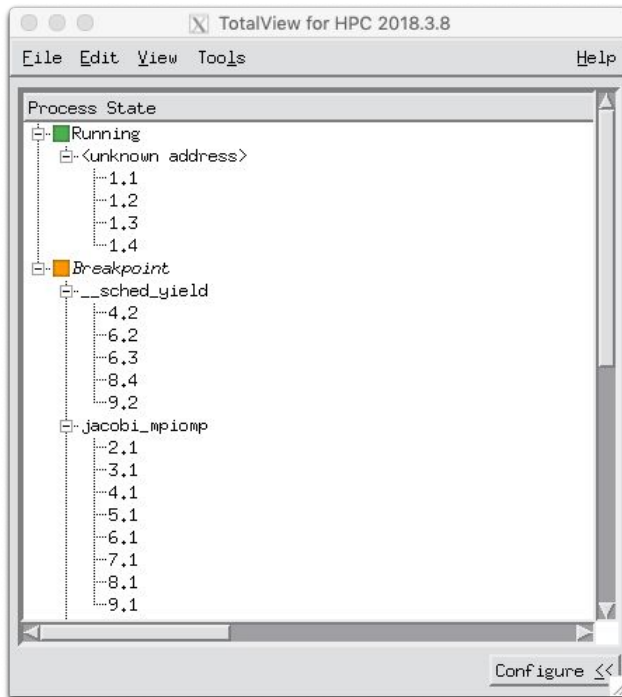
- Click 'Yes' to the question 'Process srun is a parallel job. Do you want to stop the job now?'



TotalView (cont'd)

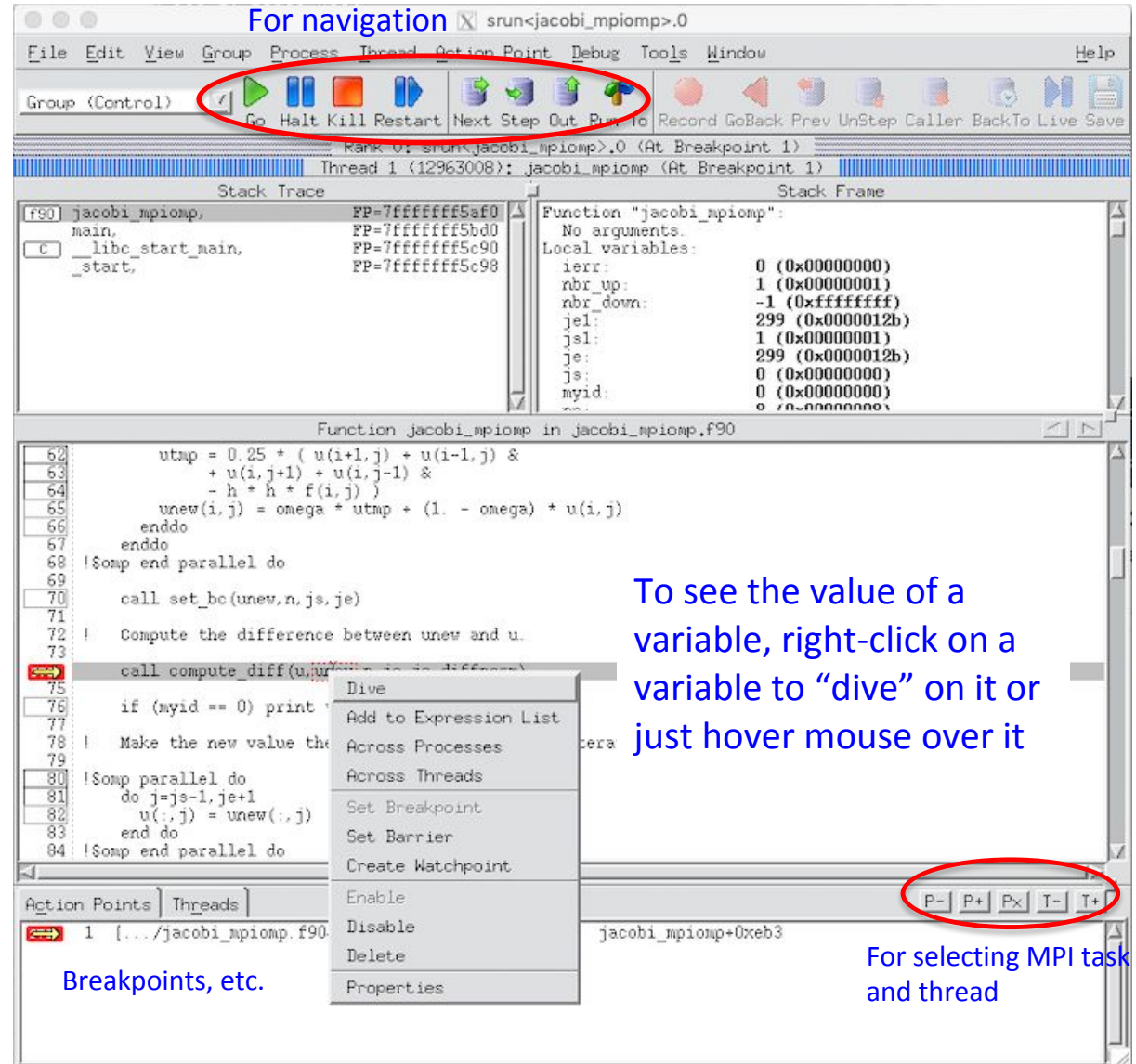


Root window



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

Process window



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 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



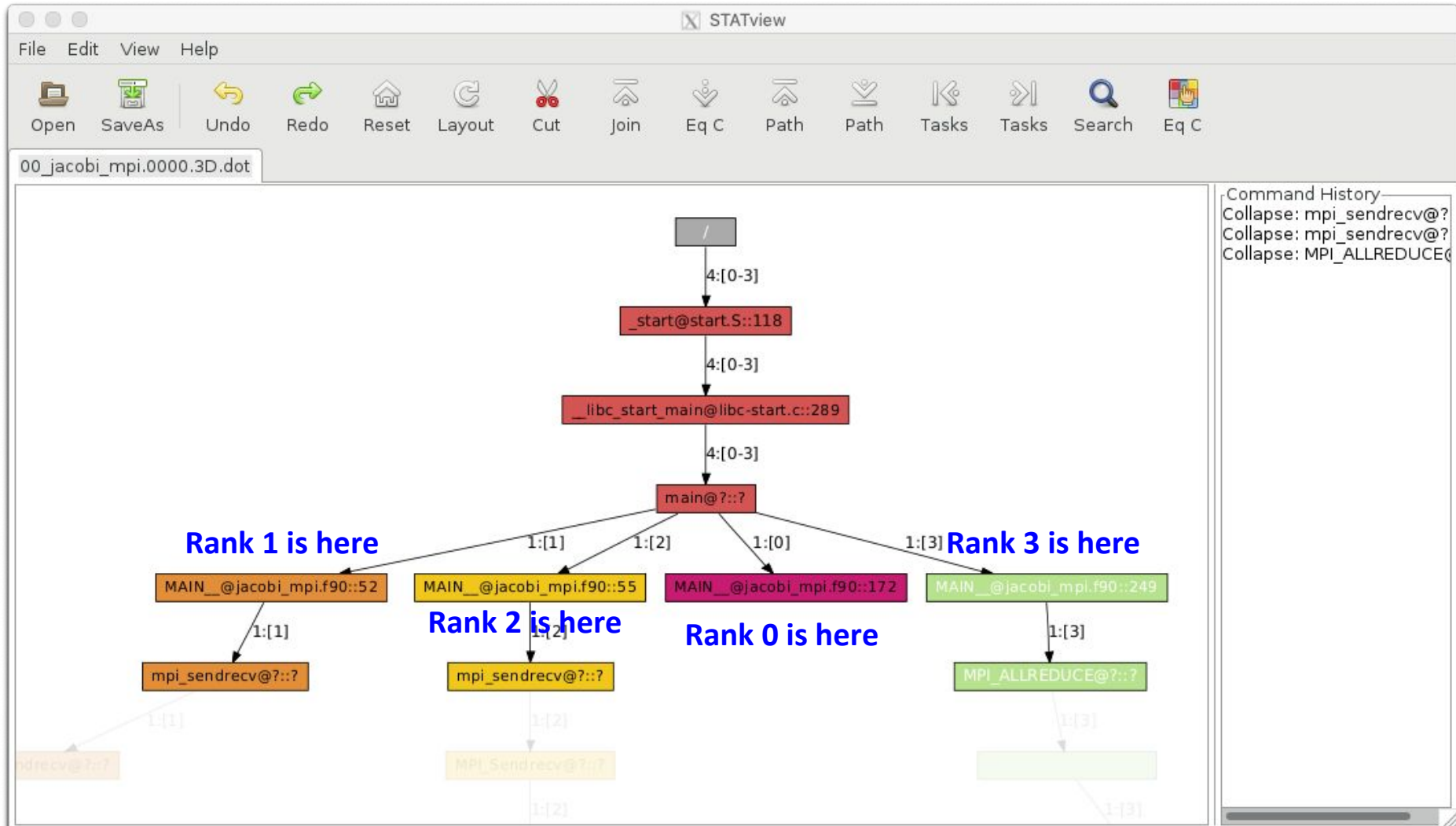
- If your code hangs in a consistent manner, you can use STAT to see whether some MPI ranks got stuck.

```
$ ftn -g -o jacobi_mpi jacobi_mpi.f90          with usual optimization flags, if any
$ salloc -N 1 -t 30:00 -q debug -C knl
...
$ srun -n 4 ... ./jacobi_mpi &
[1] 53634
$ module load stat
$ stat-cl -i 53634          -i to get source line numbers
                             STAT samples stack backtraces a few 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.0000
$ ls -l stat_results/jacobi_mpi.0000/*.dot
-rw-r--r-- 1 wyang wyang 9028 Jun 20 10:42 stat_results/jacobi_mpi.0000/00_jacobi_mpi.0000.3D.dot

$ STATview stat_results/jacobi_mpi.0000/00_jacobi_mpi.0000.3D.dot
```

Debug a hanging application with STAT (Cont'd)



Cray ATP (Abnormal Termination Processing)



- **ATP gathers stack backtraces from all processes *when the 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 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