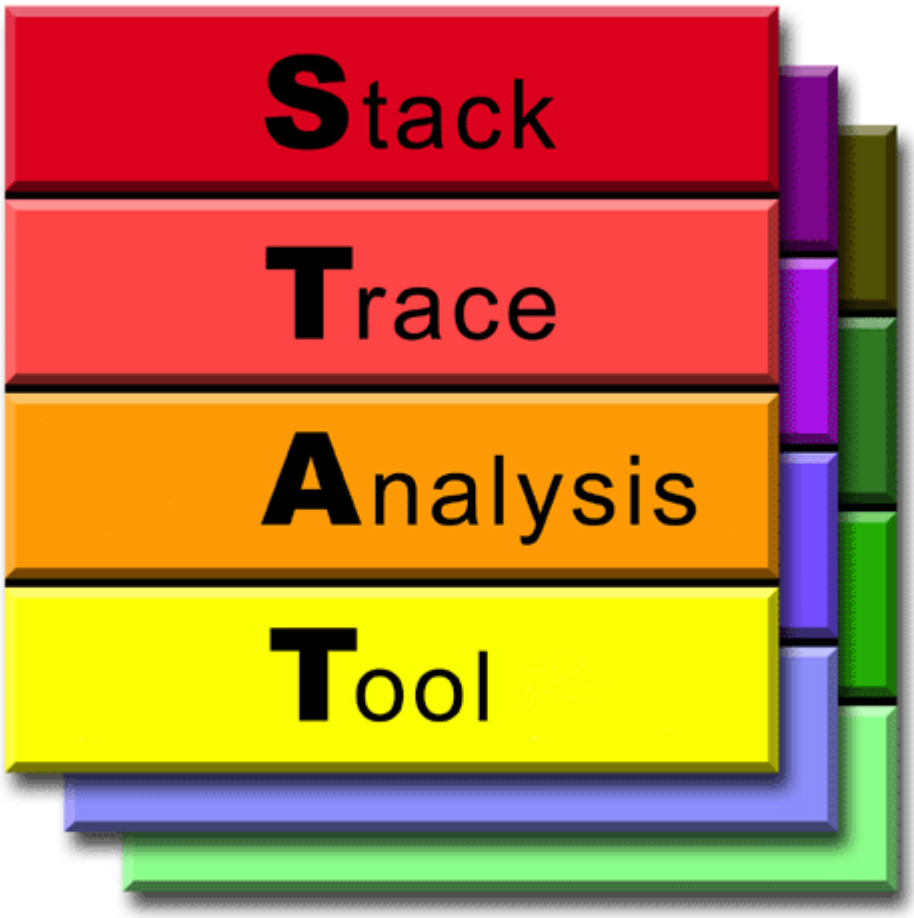
STAT: the Stack Trace Analysis Tool Quick Start Guide



Gregory L. Lee

Dorian C. Arnold

Dong H. Ahn

Bronis R. de Supinski

Barton P. Miller

Martin Schulz

STAT: the Stack Trace Analysis Tool Quick Start Guide

by Gregory L. Lee

by Dorian C. Arnold

by Dong H. Ahn

by Bronis R. de Supinski

by Barton P. Miller

by Martin Schulz

Table of Contents

Disclaimer

Auspice

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

License

of the United States Government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

Chapter 1. When To Use

STAT is most effective for diagnosing parallel applications that are hung (i.e., deadlock or livelock), however, its analysis can also be useful for debugging other errors. The outputted, merged stack traces indicate where in the code all of the processes are at a given point in time, giving the user insight into where the bug may be. The merged traces also form process equivalence classes, which can be used to select a subset of tasks to feed into a full-featured parallel debugger such as DDT or TotalView for root cause analysis.

Chapter 2. Using the STAT GUI

After launching the GUI via the **STATGUI** command you will first need to attach to the application via the **Attach** button. This will bring up the attach dialog (Figure 2-1). You will need to select the job launcher (i.e., **mpirun**, **srun**, or equivalent process). STAT will list processes owned by you on the localhost. If your job launcher process is on a remote host, you will need to enter that hostname in the remote host text entry box. Contact your local system administrator if you are not sure where to find your job launcher process. Once the appropriate process has been selected, click on the **Attach** in the lower right hand corner. STAT will then launch its daemons and gather an initial stack trace.

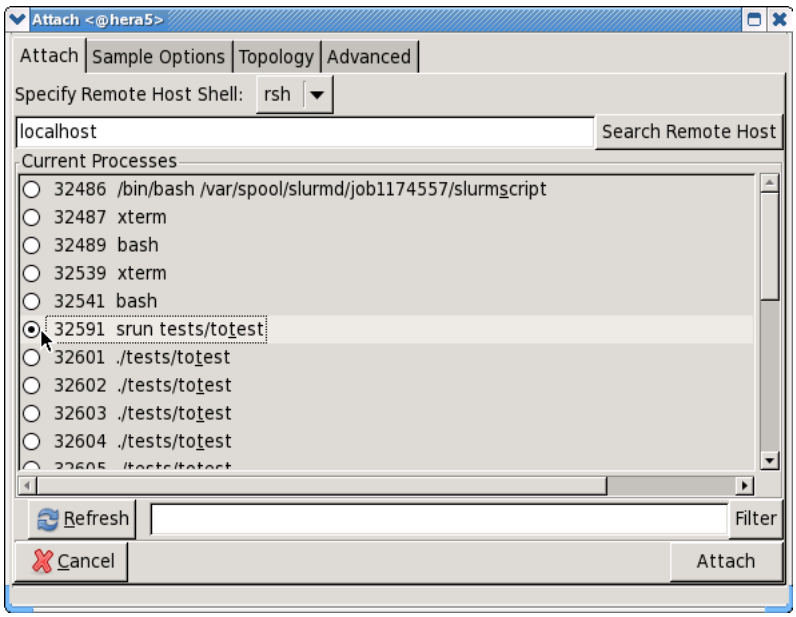


Figure 2-1. Use the attach dialog to select the job launcher process to attach to.

Once STAT has displayed the initial merged stack trace, in the form of a prefix tree, first you may want to look for common buggy patterns. This can be accomplished by using the analysis buttons in the toolbar across the top of the window. This includes operations to look for outliers such as processes with the shortest or longest stack trace or for the stack trace that was exhibited by the least or most processes. Note that these buttons are "traversal" buttons and they all initially operate on the full prefix tree. For example, the first click of the **[Shortest] Path** button will display the shortest path and subsequent clicks will display the next shortest path. Oftentimes bugs in parallel applications are triggered by a single or small subset of outliers in which case the **[Least] Tasks** button can quickly identify the outliers. Another common behavior is for a small subset of processes to be hung and the rest of the processes to be blocked in an MPI barrier or collective. In this case, the hung subset of tasks may have a shorter call path than the tasks in blocking in MPI, since the MPI implementation will usually be several frames deep. In this case, the **[Shortest] Path** can be useful.

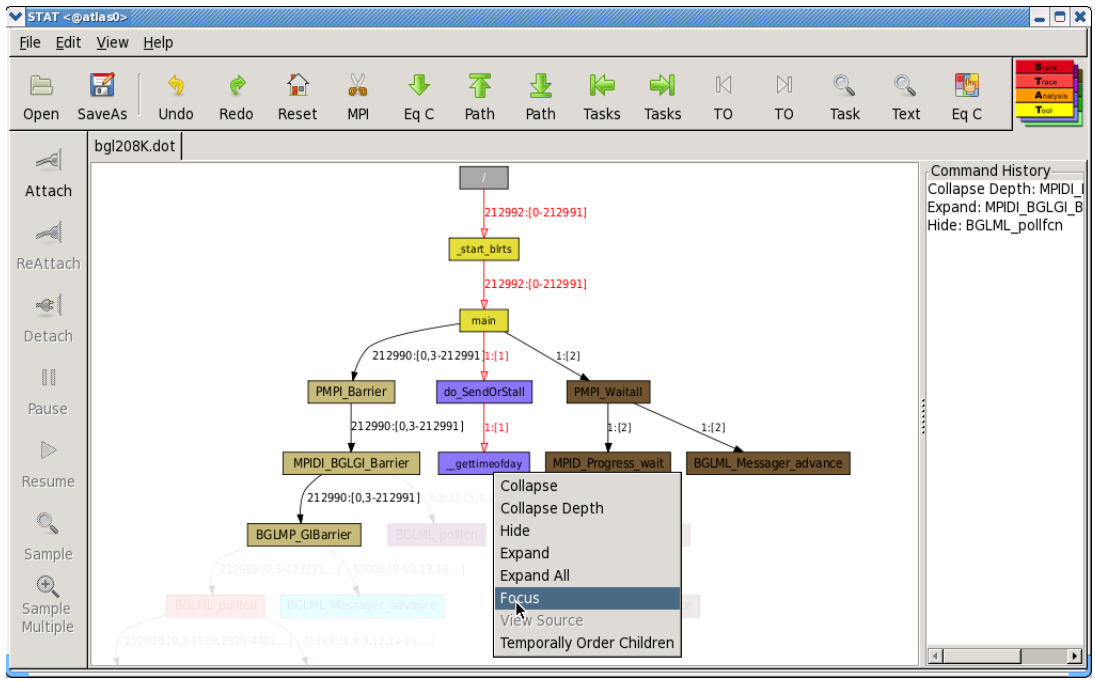


Figure 2-2. A screenshot of the STAT GUI.

Alternatively, you may wish to manually search through the stack traces. There are several buttons to aid in this process too. The **[Traverse] Eq C** will traverse the prefix tree, with each click traversing the down to the next point where there is a branch in equivalence classeses. Another helpful button is the **[Cut] MPI** button, which will cut any frames below an MPI function call, thus allowing you to focus on application code as opposed to the MPI implementation stack frames. There are also buttons to search for specific MPI ranks or for stack frames with specified text. Finally, left or right clicking on a node in the prefix tree gives you the option to expand or collapse the prefix tree. Note also that you can zoom in and out of the prefix tree using the options in the **View** menu or by using the scroll button on your mouse. You can also hold the left mouse button to "grab" the whitespace in the displayed prefix tree and move the focus around.

By default, the initial sample will gather stack traces at the granularity of function names. You can gather an additional sample with more detail, by clicking on the **Sample** button and selecting the **function and line** radio button. Note this typically requires that the code be compiled with the -g flag to get the appropriate debug information. After clicking **OK** a new prefix tree will be generated. By gathering stack traces with line number information, you may now associate stack traces back to the source code using the **View Source** button after clicking on a node in the prefix tree (Figure 2-3).

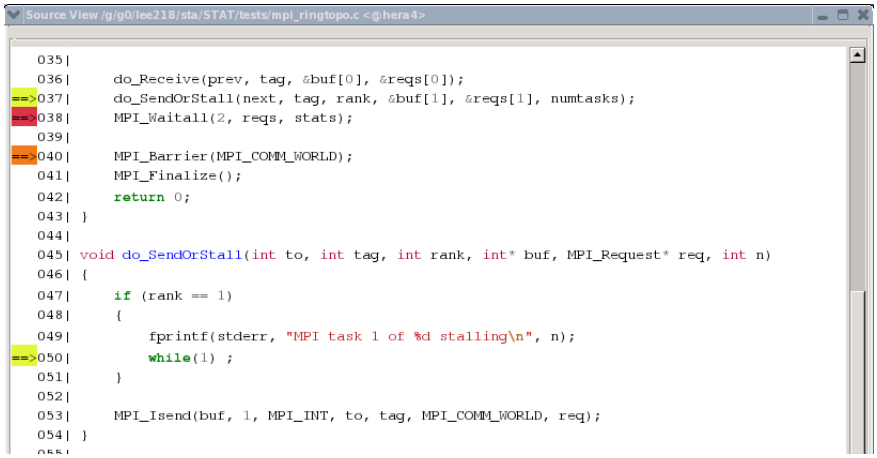


Figure 2-3. A screenshot of the source view window.

STAT was not intended to be a full-featured debugger, so you may ultimately need to employ another debugger such as DDT or TotalView for root cause analysis. STAT includes an interface to launch either of these debuggers (where available) on a subset of the MPI tasks based on the equivalence classes that STAT identifies. This interface can be accessed through the **[Identify] Eq C** button in the upper right hand corner of the window. In order to allow the other debuggers to attach, STAT will first detach itself from the application. Pinpointing a bug may require several iterations of running STAT on the entire application and running a full-featured debugger on a subset. After detaching a full-featured debugger, you can quickly attach to your application again with the **ReAttach** button.