Cray Debugging Support Tools or How to Debug Peta-scale Applications

Cray XE6 Performance Workshop
PDC/KTH
12 – 14 September

Jason Beech-Brandt, Harvey Richardson, Stephen Sachs
Cray

Cray Debugging Support Tools

- **STAT (Stack Trace Analysis Tool)**
- **ATP (Abnormal Termination Processing)**
- **MRNet (Multicast/Reduction Network)**
- **FTD (Fast Track Debugging)**
  - Supported in lgdb and DDT
- ***Plus*: ccdb (Cray Comparative Debugger)**
  - Actually separate from CDST

Stack Trace Analysis Tool (STAT)

-- My application hangs! --

What is STAT?

- **Stack trace sampling and analysis for large scale applications from Lawrence Livermore Labs and the University of Wisconsin**
  - Creates a merged stack trace tree
  - Groups ranks with common behaviors
  - Fast: Collects traces for 100s of 1000s of cores in under a second
  - Compact: Stack trace tree only a few mega bytes
- **Extreme scale**

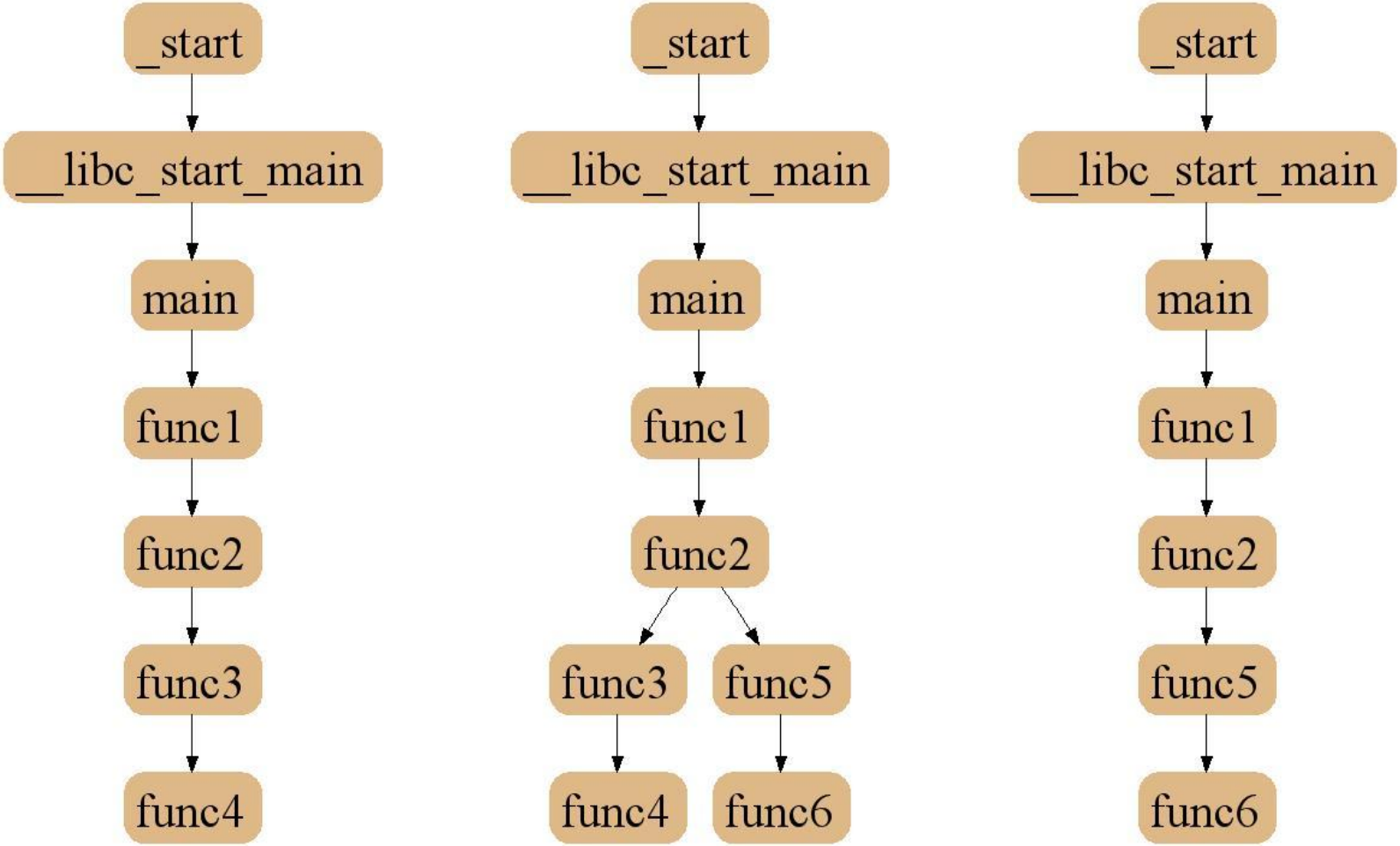
  Jaguar: 200K cores

  Hopper: 125K cores

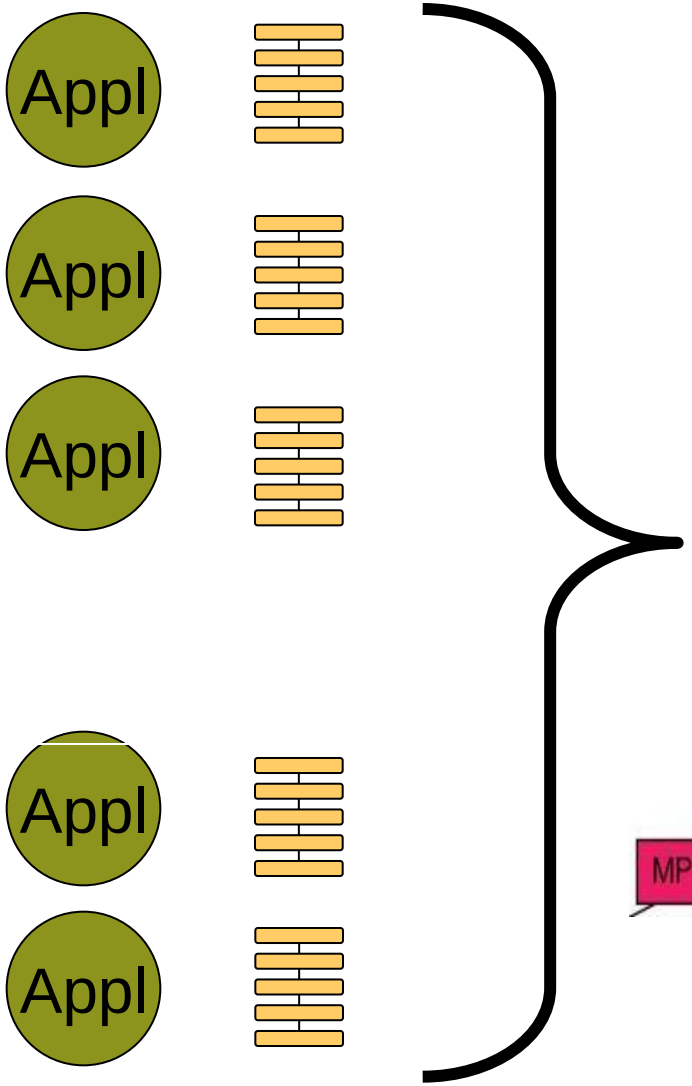
Merged stack trace trees

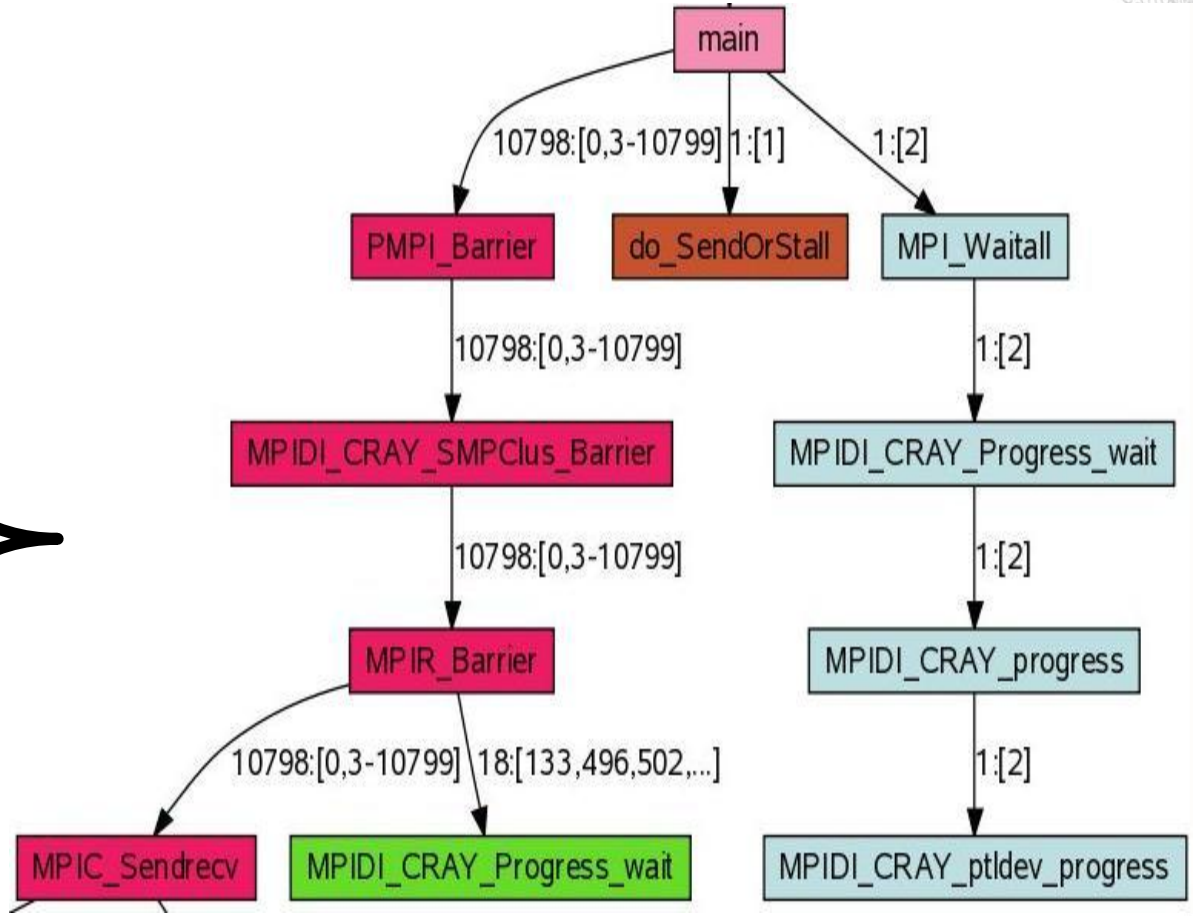
- **Sampling across ranks**
- **Sampling across time**
- **Scalable visualization**
  - Shows the big picture
  - Pin points subset for heavy weight debuggers

Stack Trace Merge Example



2D-Trace/Space Analysis

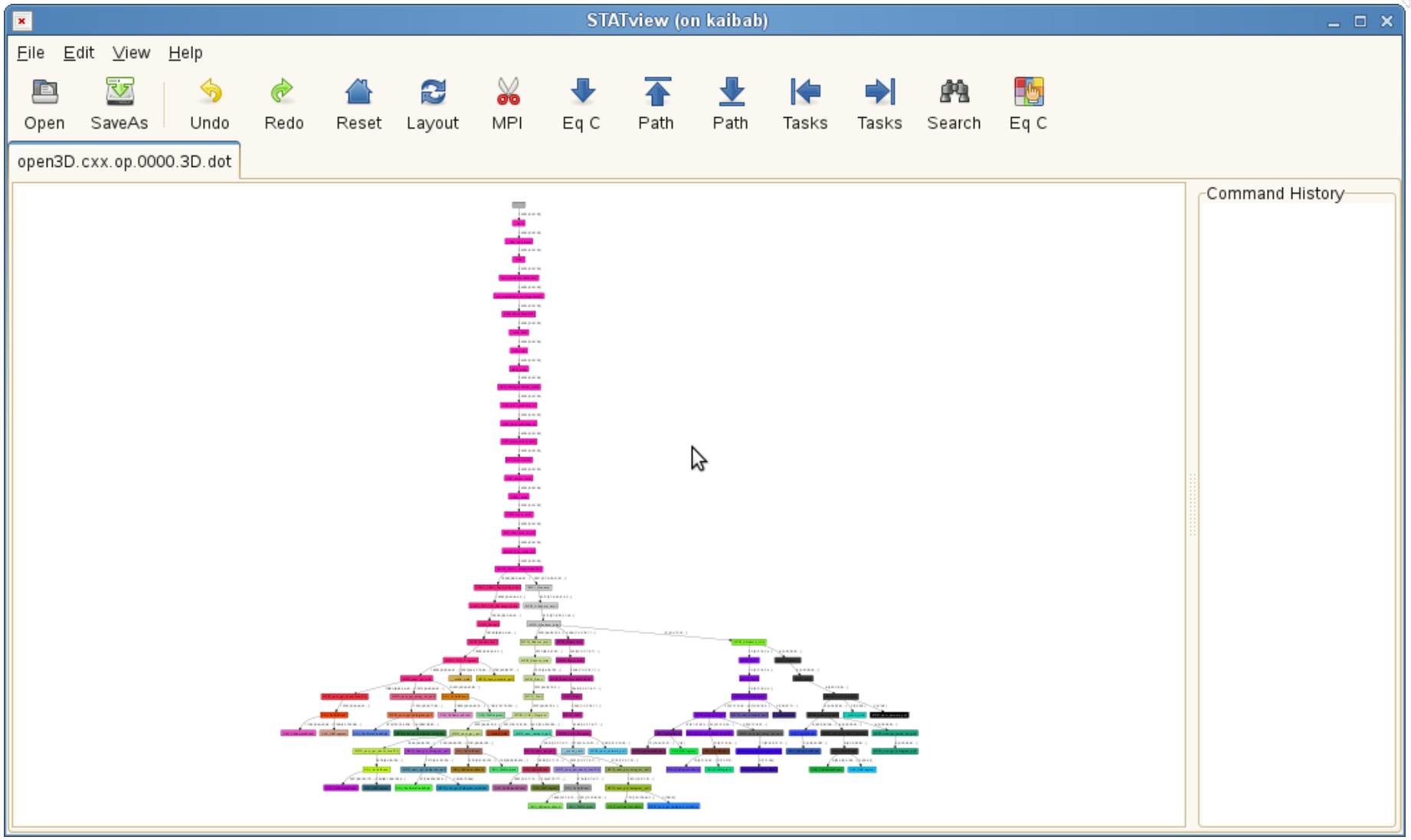




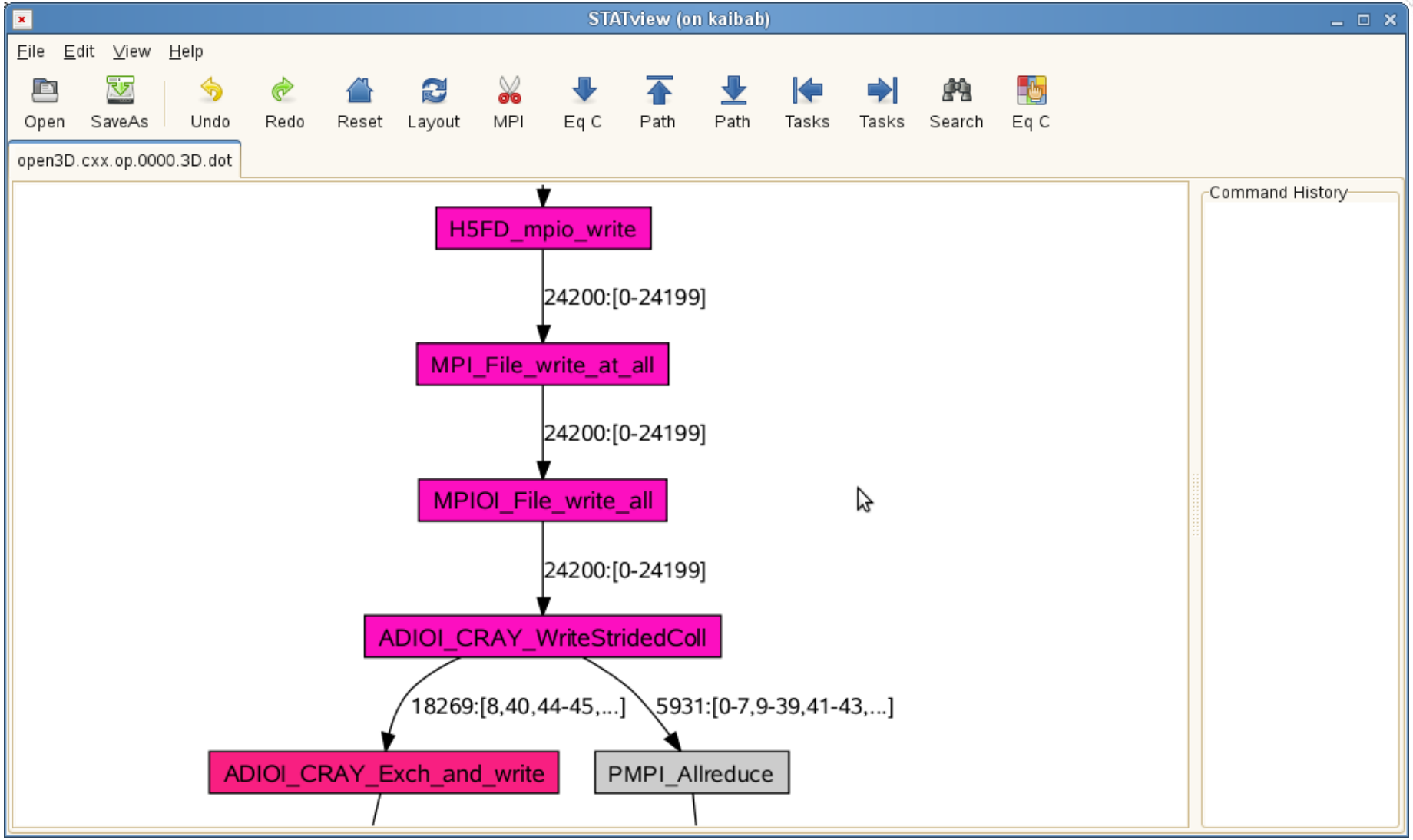
NERSC Plasma Physics Application

- Production, plasma physics PIC (Particle in Cell) code, run with 120K cores on hopper, and using HDF5 for parallel I/O
- Mixed MPI/OpenMP
- STAT helped them to see the big picture, as well as eliminate code possibilities since they were not in the tree

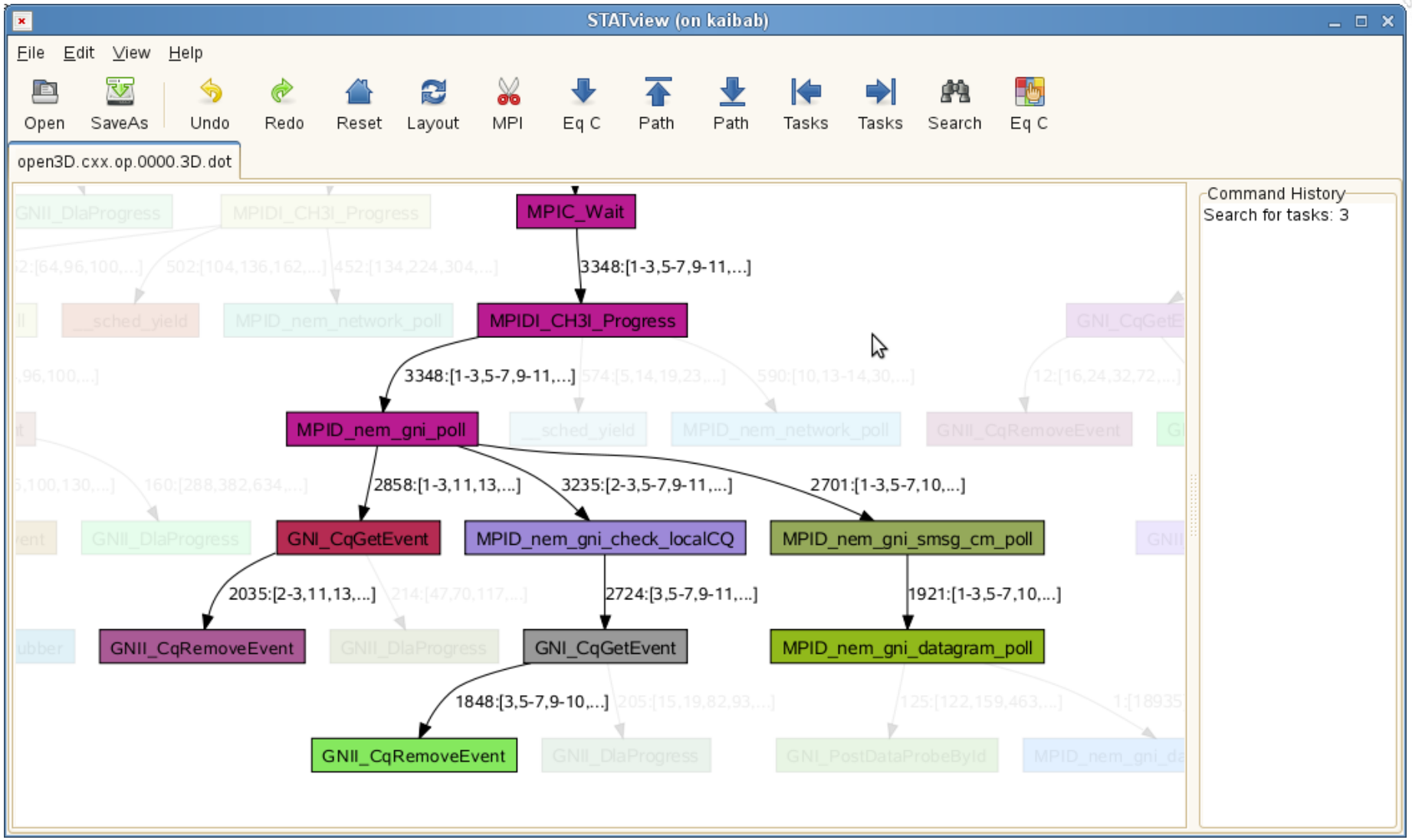
statview


STATview (on kaibab)
File Edit View Help
Open SaveAs Undo Redo Reset Layout MPI Eq C Path Path Tasks Tasks Search Eq C
open3D.cxx.op.0000.3D.dot
Command History

statview


STATview (on kaibab)
File Edit View Help
Open SaveAs Undo Redo Reset Layout MPI Eq C Path Path Tasks Tasks Search Eq C
open3D.cxx.op.0000.3D.dot
Command History
H5FD_mpio_write
24200:[0-24199]
MPI_File_write_at_all
24200:[0-24199]
MPIOI_File_write_all
24200:[0-24199]
ADIOI_CRAY_WriteStridedColl
18269:[8,40,44-45,...]
5931:[0-7,9-39,41-43,...]
ADIOI_CRAY_Exch_and_write
PMPI_Allreduce

statview…


STATview (on kaibab)
File Edit View Help
Open SaveAs Undo Redo Reset Layout MPI Eq C Path Path Tasks Tasks Search Eq C
open3D.cxx.op.0000.3D.dot
Command History
Search for tasks: 3
MPIC_Wait
3348:[1-3,5-7,9-11,...]
MPIDI_CH3I_Progress
3348:[1-3,5-7,9-11,...]
MPID_nem_gni_poll
2858:[1-3,11,13,...]
3235:[2-3,5-7,9-11,...]
2701:[1-3,5-7,10,...]
GNI_CqGetEvent
MPID_nem_gni_check_localCQ
MPID_nem_gni_smsg_cm_poll
2035:[2-3,11,13,...]
2724:[3,5-7,9-11,...]
1921:[1-3,5-7,10,...]
GNII_CqRemoveEvent
GNI_CqGetEvent
MPID_nem_gni_datagram_poll
1848:[3,5-7,9-10,...]
GNII_CqRemoveEvent

STAT: Since We Were Here Last Year

- **A new addition: STATGUI**
  - Work bench for repeated requests
    - Change granularity
    - Change sampling
    - Continue then resample
  - Launches or attaches

STAT 1.2.1.1

- module load stat
- man STAT
- STAT <pid_of_aprun>
  - Creates STAT_results/<app_name>/<merged_st_file>
- statview <merged_st_file>
- STATGUI
- Scaling no longer limited by number file descriptors

My questions for you

- **How many of you are using STAT?**
  - Those of you who aren't, why not?
- **What would you like to see from STAT?**

Abnormal Termination Processing (ATP)

-- My application crashes! --

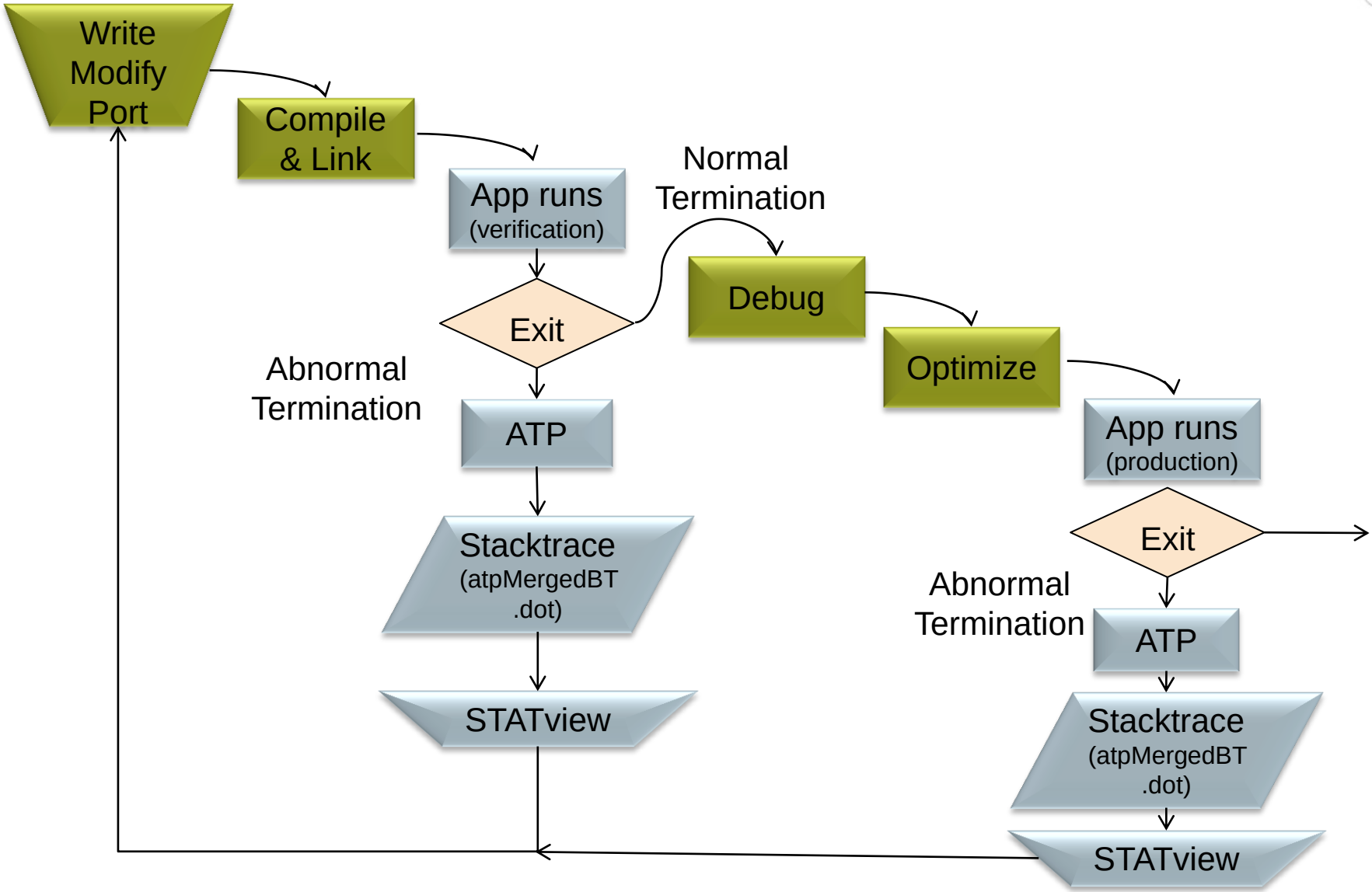
The Problem Being Solved

- Applications on Cray systems use hundreds of thousands of processes
- On a crash one, many, or all of them might trap
- No one wants that many core files
- No one wants that many stack backtraces
- They are too slow and too big.
- They are too much to comprehend

ATP Description

- System of light weight back-end monitor processes on compute nodes
- Coupled together as a tree with MRNet
- Automatically launched by aprun
- Leap into action on any application process trapping
- Stderr backtrace of first process to trap
- STAT like analysis provides merged stack backtrace tree
- Leaf nodes of tree define a modest set of processes to core dump
- Or, a set of processes to attach to with a debugger
- To use….
  - Set “ATP_ENABLED=1” in your job script prior to launch

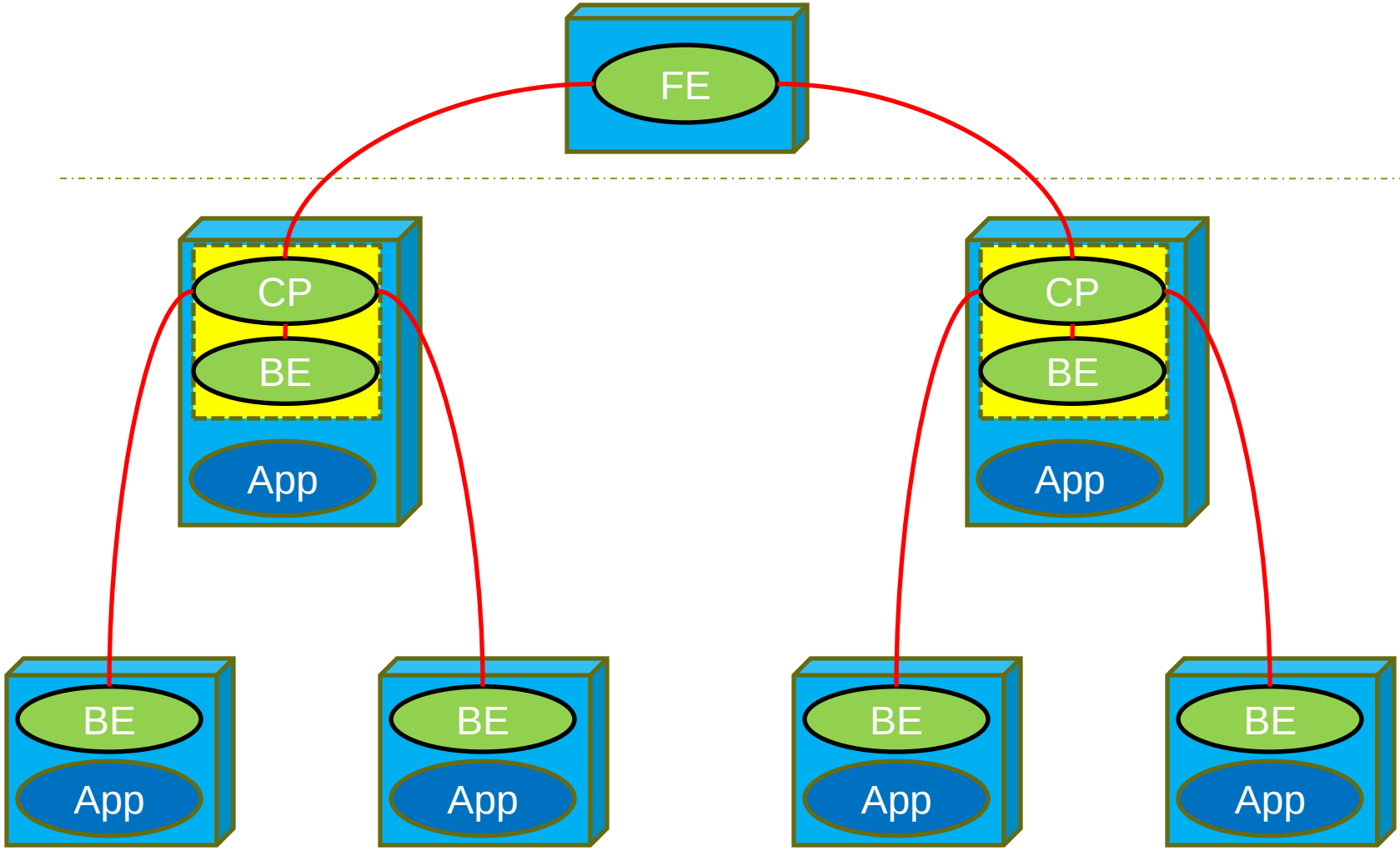
ATP – Abnormal Termination Processing



ATP Components

- **Application process signal handler (atpAppSigHandler)**
  - triggers analysis
- **Back-end monitor (atpBackend)**
  - collects backtraces via StackwalkerAPI
  - forces core dumps as directed using core_pattern
- **Front-end controller (atpFrontend)**
  - coordinates analysis via MRNet
  - selects process set that is to dump core
- **Once initial set up complete, all components comatose**

ATP Communications Tree



ATP Since We Were Here Last Year

- **Added support for:**
  - Dynamic Applications
  - Threaded Applications
  - Medium memory model compiles
  - Analysis on queuing system wall clock time out
- **Eliminated use of LD_LIBRARY_PATH**
- **Numerous bug fixes.**

Current Release on Lindgren: ATP 1.4.4

- **Automatic**
  - ATP module loaded by default
    - Signal handler added to application and registered
  - Aprun launches ATP in parallel with application launch
  - Run time enabled/disabled via ATP_ENABLED environment variable (can be set by site)
- **Provides:**
  - backtrace of first crash to stderr
  - merged backtrace trees
  - dumps core file set (if limit/ulimit allows)

What's Next for ATP?

- Support for Checkpoint/Restart
- Support higher scale
- Improved output file naming system
- E-mail on crash, if user requesting HOLD

My questions for you

- **Who is from a site that does not run ATP as the default?**
  - Why don't you?
- **What problems are you seeing with ATP?**
- **Would a comprehensive list of signal per rank be useful?**
- **What would you like to see from ATP?**

The Cray Comparative Debugger

-- My application gives the wrong answer! --

What is ‘ccdb’?

- **Cray Comparative Debugger**
  - **ccdb**: A command line, parallel debugger, leveraging **gdb**
    - Typical debugger commands: break, continue, step, where, print, etc.
    - Process sets: restrict focus
  - **ccdb**: A comparative debugger

What is a comparative debugger?

- **Originally from Monash University in Australia**
- **Uses a working application to find bugs in a failing version**
  - Compares data for two simultaneous runs
  - Stops and announces when data fails to compare
- **Data centric – no additional complexity as scale increases**
- **Well, not for the user, but…**
  - Cray and Monash are working together under a grant from the Australian government on scalability research.

CD Scenarios

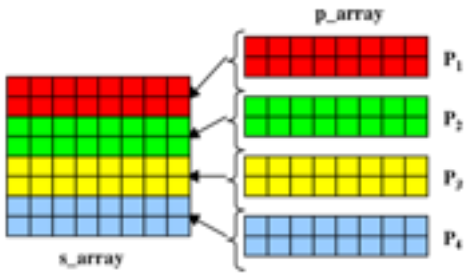
- **What common scenarios provide a working and non-working application?**
  - Yesterday vs. today
  - Varying scale: small parallel vs. larger parallel
  - Varying libraries: new vs. old release
  - Varying optimizations: -O2 vs. –O3, scalar vs. vector
  - Port from serial to parallel
  - Language port: C vs. Fortran
  - Varying architectures: IBM vs. Cray

Assertions

- Data assertions are the heart of comparative debugging.
- Assert that data1 at location1 matches data2 at location2.
- The debugger verifies this assertion as the applications execute.
- When the assertion fails, one now has a specific region of the failing application to inspect.
  - This often means that the user iterates with a refined assertion to further narrow the search area.
- Assertions can be simple (scalar to scalar) or more complex (serial to parallel multidimensional arrays),

Data decomposition

- **Mapping one app's data layout to another's.**
  - In particular, mapping the data associated with the processes.
- **Blockmap uses HPF syntax**
  - Block
  - Cyclic
  - *



ccdb updates

- **Focus has been on the DARPA/Cascade requirements**
  - Added MRNet and response aggregation for scalability
  - Added FORTRAN support
  - Extended Blockmap for more complex decompositions
  - Tuned up *array slicing* for decomposition halo support
- **Numerous bug fixes and usability improvements.**

CCDB 1.0.0.7

- **Prototype currently on our internal systems**
  - `load module ccdb`
  - `Man ccdb`
  - `Scales to over two, 4K application`

My questions for you

- **What scenarios are most important to you?**
- **How important is debugging C++?**
- **Application idioms**
  - How common are non-uniform data decompositions?
  - How common are FORTRAN automatic arrays with runtime computed sizes?
  - Is thread local data commonly used?

Fast Track Debugging

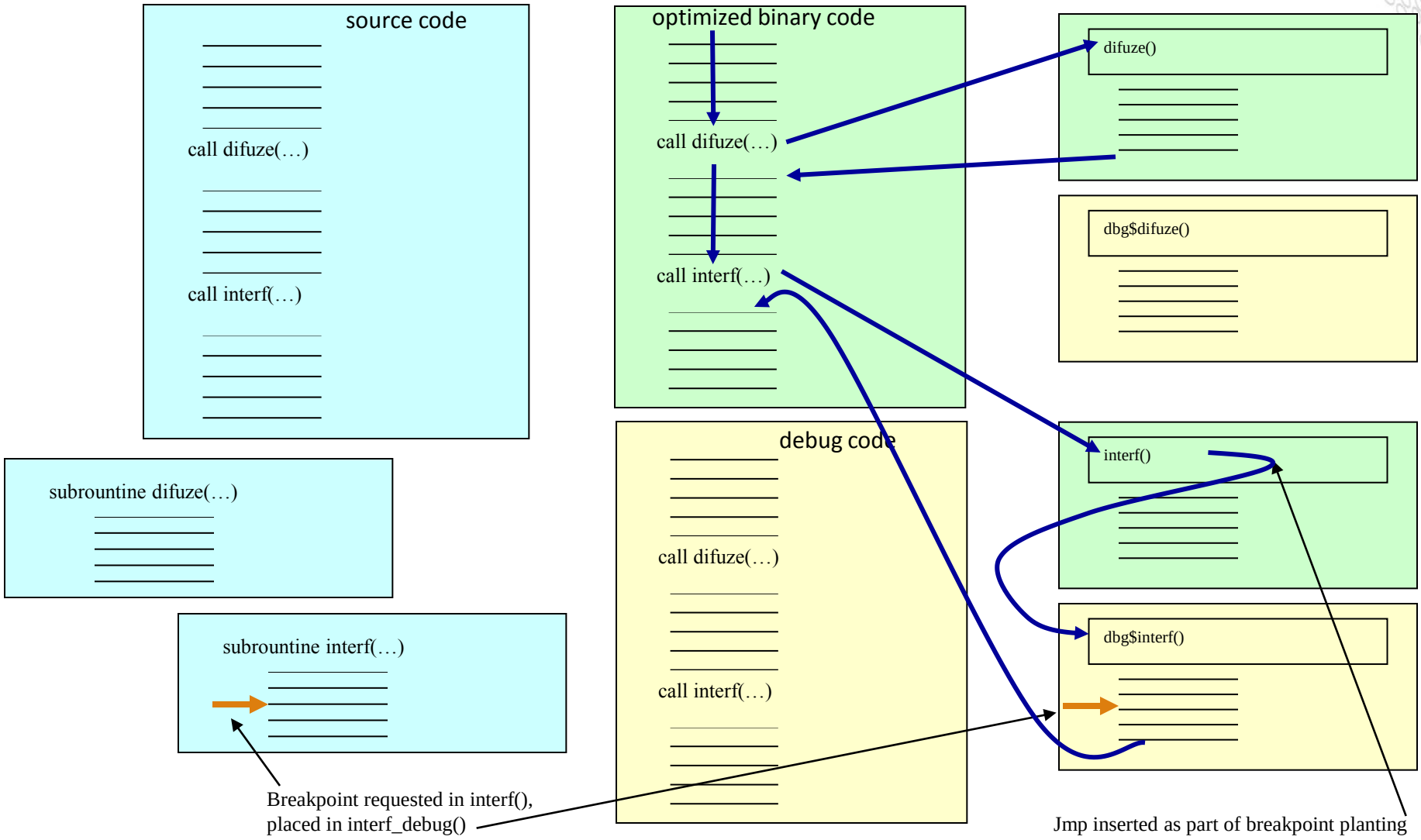
The Problem

- **Debug compiles eliminate optimizations**
  - Today's machines really need optimizations
  - Slows down execution
  - Problem might disappear
- **Fast Track Debugging addresses this problem**

What is "Fast Track Debugging"?

- Compile such that both debug and non-debug (optimized) versions of each routine are created
- Linkage such that optimized versions are used by default
- Debugger overrides default linkage when setting breakpoints and stepping into functions

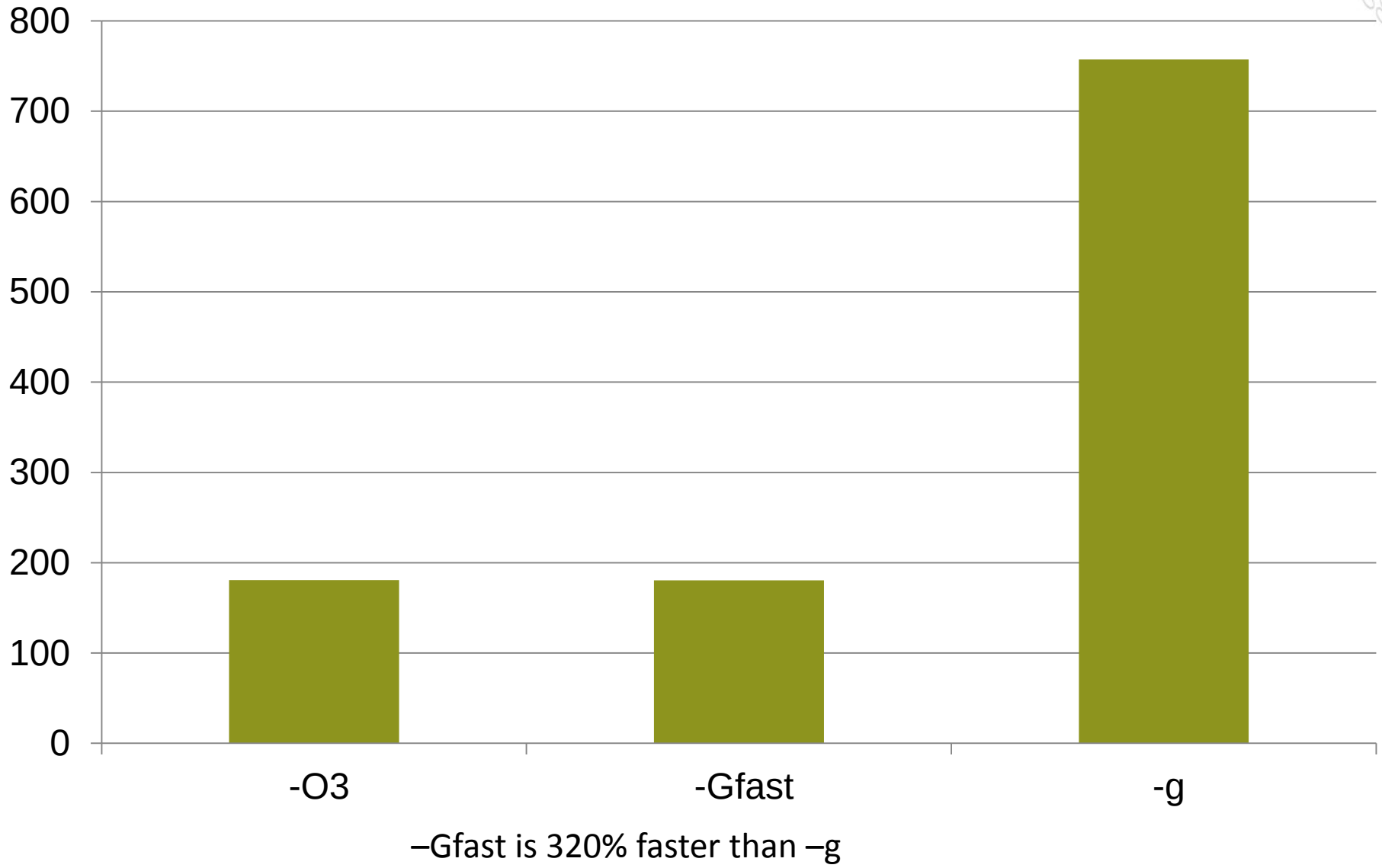
A Closer Look at How FTD Works



Where Things Stand Today

- **Only currently supported in CCE**
  - Compile code with -Gfast
- **Fully supported in Allinea's DDT**

Tera TF Execution Time



–Gfast is 320% faster than –g

Cost

- **Compiles are slower**
- **Executable uses more disk space**
- **Inlining turned off**
  - 1.7% average slow down of all SPEC2007MPI tests
  - Range of slight speedup to 19.5% slow down
- **Uses more memory**
  - 4% larger at start up
  - 0.0001% larger after computation

Questions?

- **What other debugging tools are you using – on Lindgren or other systems**
  - DDT?
  - Totalview?
  - GDB?
  - Others?