

BOF Report: Techniques and Strategies for Extreme Scale Debugging

This BOF was an opportunity for users to share experiences debugging very high scale applications on systems capable of expressing those high levels of parallelism. We started with 5 talks summarizing and commenting on the techniques and strategies that 5 different leading sites have used.

Ray Loy gave an *“ALCF Scalable Debugging Update”*. He reviewed the tools that they use, including DDT and TotalView, but also bgq_stack, coreprocessor, and STAT. He talked about using DDT at 16k cores. He also highlighted that the I/O nodes in the BG architecture represent a bottleneck that can keep tools from working. Tools need to work within that limited resource.

Tobias Hilbrich presented a talk about *“Scalable Runtime Error Detection for MPI with MUST”*. Tobias and his Coauthor Martin Schulz talk about how extreme scale makes it more likely that even extremely unlikely timing problems will manifest. They review how their MUST scalable runtime correctness tool helps analyze these problems by recording the precise interleaving that the MPI implementation itself selects and then uses this information to help with root cause analysis for simulations that give incorrect numerical results or crash.

Ignacio Laguna spoke about *“Probabilistic Debugging at Large Scale with AutomaDeD”*. Ignacio and his co-authors Dong Ahn, Bronis de Supinski, Todd Gamblin, Martin Schulz, Saurabh Bagchi presented a case study about a hang in the molecular dynamics code ddcMD that only happened intermittently with 8192 or more cores on BG/L. They developed a technique based on MPI wrappers to analyze the progress of the code, developing a progress dependency graph and identifying the least progressed task.

Dong Ahn reviewed *“Cases for Extreme-Scale Errors: Today’s Practice and Next Challenges”*. Dong shared some of his insights in debugging at extreme scale. He talked about a strategy that focuses on extremely scalable and lightweight limited function debugging tools like STAT and then if and when needed transitions to using a more full featured debugger like TotalView on a subset of the full set of tasks. STAT helps identify the subset and avoid the cost of running the full featured debugger across. He reviewed a set of scenarios that included a bug at more than 1M processes and another across the full 96 BG/Q racks of Sequoia. He concluded by emphasizing that an important remaining challenge is in the area of non-deterministic errors that only manifest at scale.

Bernd Mohr spoke about *“Blue Gene/Q Debugging at Julich Supercomputing Centre”*. He wrapped up the discussion with observations about the experiences they have had there in Julich with tools such as TV, STAT, MUST and Scalasca. He highlighted some challenges for TV (they have had trouble with MPMD programs) and stat (one user figured out that they would need 100000x4000 pixel screen to show the graph) and their successes (he showed Scalasca trace data set from a 1,000,000 MPI task job).

We had a lively Q&A and discussion session following the talks.

We had approximately 30 people in total at the BOF. We distributed a 7 question survey and got 6 responses. I've tried to highlight interesting responses here and the full data can be obtained using the link below.

What do you consider extreme scale?

“Provides great performance but difficult to take advantage of it”, “>100k concurrency”, “Hard to write code, IDEs or debuggers are not very useful for such problems”, “anywhere from 10k cores to >100k cores”, “I am a student so for me extreme scale is my 10 node cluster”

What is the largest scale at which you have encountered a bug? Where you able to debug it? If so what techniques did you use? If not, why not?

“128 nodes. I could not debug due to the complexity of the target and ignorance about debugging tools.”, “32k. I used core files and code inspection.”, “4096 MPI ranks, I used bisection and the experience was painful”, “768k cores on MIRA. The crash-type debugging using corefiles. Not able to debug hung-type applications because DDT crashes at large scale”, “overflow of buffer allocation for MPI communication.”

What is it about extreme scale bugs or the debugging experience at extreme scale that is different?

“Huge parallelism makes debugging more difficult, especially for non-deterministic bugs”, “Hard to access back end nodes. Queue wait times are sometimes 1-2 days long”, “It is difficult because we can't tests at scale frequently. It may be months before another similar scale job is executed”, “Large amounts of debug data and the impossibility of interactive debugging”

What flaws or complaints do you have about current tools or techniques that you or your colleagues use then debugging at extreme scale?

“Simple coredump and printf is not a good way to detect non-deterministic bugs. Tools that can easily capture such bugs are needed.”, “TotalView is too expensive for lots of licenses. STAT is difficult to install.”, “DDT crashes at large scale.”

What new techniques did you learn about here?

MUST, AutomaDeD, DDT offline mode, Program slicing and looking at least/most progressed tasks.

What tools and techniques do you expect to use when debugging on exascale systems?

TotalView and TAU, core files, Unit tests, logging on failing cores, ddt/totalview, printout, MUST, STAT, correctness checks.

What is the most important thing that needs to be developed to fill the gap between current capabilities and what will be needed for exascale?

“Guidelines and best practices to achieve performance”, “It must be lightweight. It would be much more helpful if static code analysis warns me because we cannot try huge scale jobs frequently”, “1. Debugger scalability, 2. Unified way of running debuggers, 3. Hardware debugger/helper capabilities, 4. Ability to suggest likely code region that is the cause of a bug/problem”

The slides presented to accompany the 5 presentations are available at

<https://www.dropbox.com/sh/ouco7iv9fbcj2ky/5ob9rIhAFd>

Transcriptions of all the survey responses are also available.

<https://www.dropbox.com/s/ka1jyx9xybsan3/transcribed%20survey%20responses-2013-12-30.xlsx>