LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-TR-226785

Report on New Capabilities for the Purple Development Environment

W. S. Futral, J. C. Gyllenhaal, M. E. Wolfe, C. M. Chambreau

December 13, 2006

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Purple L1 Milestone Review Panel

Report on New Capabilities for the Purple Development Environment

Scott Futral, Chris Chambreau, John Gyllenhaal, Matthew Wolfe
12/12/2006

Deliverable

As part of the deliverables for the Development Environment for Purple, additional capabilities to improve the tools offerings and to address unique Purple system requirements, such as increased processor count, were expected. This report details some of the new capabilities that have been incorporated into the development environment tools for Purple.

Report

The shift on Purple to 64-bit applications (from 32-bit on White) initially broke many debugging and memory tools. Most tools were updated to support 64 bit well before Purple was delivered to LLNL, but the company that provided the popular heavy-weight 32-bit AIX memory tool, ZeroFault, was reluctant to port to 64 bit due to perceived lack of market. LLNL tried offering financial incentives to the ZeroFault developers, which were turned down, but eventually they did give vague promises to try to port to AIX 64-bit mode when they got time. The ZeroFault developers have been making intermittent and very slow progress over the last two plus years, but despite getting close, have not released a version of ZeroFault that yet meets our needs for 64-bit applications. However, given the critical need for memory tools and the uncertainty of ZeroFault development, other memory tool options were actively pursued and delivered.

Both of the new memory tools for Purple (TotalView memory tools and Valgrind Memcheck) came out of a tri-lab effort starting in July 2001 to produce a common set of memory tools for all ASC platforms. Using DOE path forward funding, the tri-labs funded and directed the development of TotalView's memory tools from Etnus starting in 2003. TotalView's memory tools are a light-weight memory tool set that is essentially a high-quality and more powerful debug malloc, due to close integration with the TotalView debugger. For example, the TotalView debugger will tell you if pointers are currently valid, or pointing to freed memory, and will stop the program if you free a pointer twice. TotalView's memory tools also include a very useful functionality for tracking down memory growth issues (a common problem when scaling codes), queries for determining what portions of your code are consuming what memory (a very common user request, again when scaling up codes), and tools for finding leaked memory. TotalView's tools are "light-weight" since they minimally impact the run-time of the application. They are typically 5-25% slower than normal execution, with 2X slower being the worst case observed. TotalView's memory tools were developed and available for use on Purple (and other ASC platforms) in September 2005. LLNL fine-tuned

TotalView's memory toolset user-interface well into 2006 and continues to push the vendor's development efforts to make these tools useful with very large scale runs.

The second memory tool technology developed, Valgrind Memcheck, is also a product of the same Tri-lab effort to produce common memory tools across ASC platforms. The Valgrind Memcheck tool is an open-source tool that was made portable (was x86 linux only) and ported across platforms. Tri-lab PSE funding beginning in July 2004 will continue until at least Sept 2007. Valgrind Memcheck complements the functionality of the TotalView memcheck tools. Valgrind Memcheck is a heavy-weight memory tool that can tell you exactly where write or read to out of bounds memory occurs, reports instances of reading un-initialized data, etc. as well as finding memory leaks. Valgrind Memcheck is called "heavy-weight" since applications typically run 5X (best case) to 100X (worst case) slower (25X-40X typical) under the tool. Valgrind Memcheck can find a similar class of errors to Purify, and can find very hard-to-track-down memory errors that TotalView memory tools cannot find. The initial port to Purple for Valgrind was completed in September 2006 and uncovered several operating system(OS) and compiler issues. The OS and compiler issues have been fixed or worked around, and Valgrind will soon be deployed for general use on Purple.

The TotalView debugger was improved and new capabilities added for the benefit of the ASC Purple Development Environment. In addition to the memory debugging discussed, TotalView now detects when it is being run in AIX large pages, and warns the user. Several AIX and POE problems have been reported to, and fixed by, IBM. TotalView now uses SLURM to more efficiently launch its debugger servers on AIX. Fast conditional breakpoints and data watch points are now operational on Purple-class systems as a result of work contracted with IBM and Etnus as part of the Purple acquisition.

The mpiP lightweight MPI profiling tool (http://mpip.sourceforge.net/) has been enhanced in two important ways to address the scalability of the tool and to simplify evaluating MPI performance data at scale.

The first enhancement modifies the manner in which mpiP generated output data. Originally, mpiP collected all task data at a single collector task and then generated a report from this data. With this approach, the amount of memory used to hold this data depends on the number of tasks, the number of MPI call sites, and the number of stack frames unwound at each MPI call (an mpiP runtime setting). As the size of the stack trace increases, each unique stack trace is considered a new call site, which can cause the number of unique call sites quickly to grow very large. Since the use of mpiP with large stack traces and large task counts could lead to swapping on Purple and failure on other systems such as BG/L, the mpiP reporting process was modified such that instead of accumulating all profile data at a single task, call site report data is produced on a call site by call site basis using MPI collectives after the application calls MPI_Finalize. While this does add some overhead from the MPI collectives, the memory use is a small fixed amount, greatly reducing the risk of swapping or memory exhaustion. In practice, the increase in report generation time is on the order of seconds.

To interpret mpiP output data more easily, a new mpiP summary output format has been implemented. Instead of wading through long sections of per-task information, the summary output format provides concise summary statistics regarding application and MPI time as well as the top 20 call sites for MPI time and total MPI message volume. A run-time flag allows users to generate a concise or verbose report or both. The memory reduction functionality and report formats were verified by running an MPI benchmark with mpiP over 64K tasks and with Qbox over 16K tasks on BG/L as well as runs of ALE3D on Purple at 8192 tasks.

Another enhancement to the performance tools offered for Purple was the development of the Open Trace Format (OTF) specification and implementation for Purple. After it became clear that the scalable functionality of the Structured Trace Format (STF) developed by Pallas/Intel would not be available to us indefinitely, we pursued the development of the Open Trace Format (OTF) to provide a scalable trace format and reader/writer library that would be open source and would allow us to make use of the trace format on our various platforms of interest. Our partnership with ParaTools and TU-Dresden resulted in a functional, scalable, open source trace format and reader/writer trace library. The OTF reader/writer library is now supported by several tools such as TAU, KOJAK, and Vampirtrace for generating OTF trace files and has been tested on all LLNL production platforms, including ASC Purple.

Currently, the primary means of analyzing OTF trace data is the VampirNextGeneration (VNG) analysis tool developed by TU-Dresden. We have existing license arrangements with TU-Dresden and an understanding for future support. OTF traces can also be converted to other formats for use with other trace or profile tools.

OTF verification on LLNL systems with applications of interest to LLNL was specified as an item in the second OTF contract. This included trace generation with various applications and ASC Purple benchmarks. These various tests were recreated and viewed by the LLNL technical contact on LLNL systems. A 32 GB trace file was generated by ParaTools from the Miranda application and viewed with VNG by the LLNL technical contact

The combination of IBM enhancements along with the tools detailed in this report, insure that the ASC Purple user development environment is in the same leadership class as the performance of the machine.