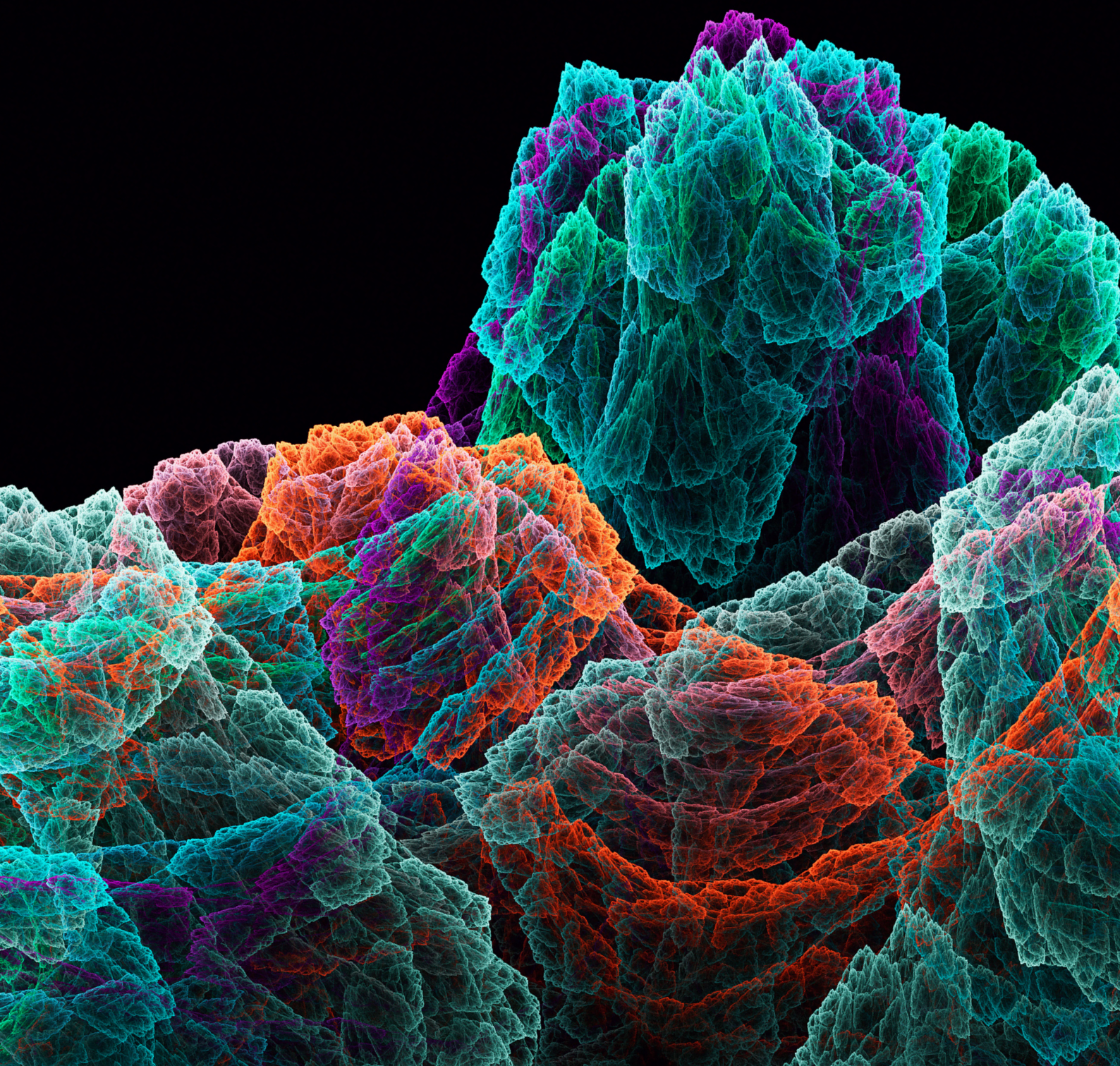


HPE CRAY PROGRAMMING ENVIRONMENT

An essential toolset for organizations developing their own HPC codes



Check if the document is available
in the language of your choice.

Improve performance of your applications with HPE Cray Programming Environment, a fully integrated software suite with compilers, tools, and libraries designed to increase programmer productivity, application scalability, and performance.

ABOUT HPE CRAY PROGRAMMING ENVIRONMENT

HPE Cray Programming Environment suite offers programmers a comprehensive set of tools for developing, porting, debugging, and tuning of applications.

The programming environment simplifies the transition to new hardware architectures and configurations by automatically applying optimizations on HPC applications that use existing programming models with a simple recompile.

The environment provides:

- Compilers, programming languages, and models
- Scientific, math, and I/O libraries
- Deep learning plug-in
- Debugging tools
- Profiling and performance optimization tools

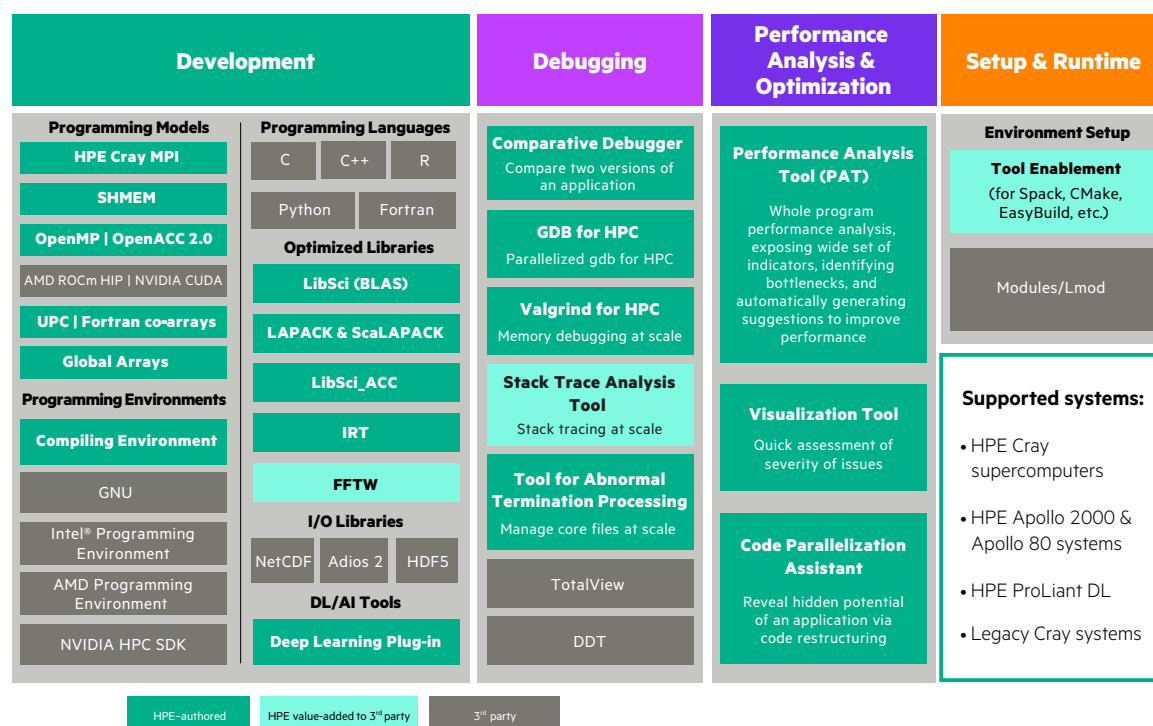
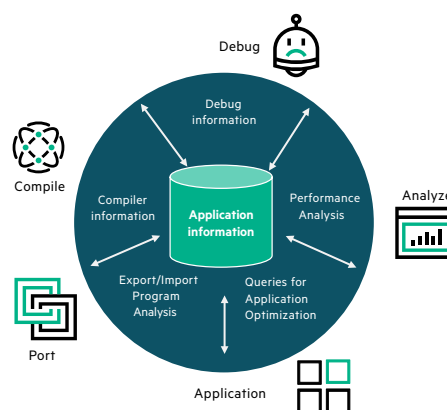


FIGURE 1. Overview of tools included in HPE Cray Programming Environment

WHY HPE CRAY PROGRAMMING ENVIRONMENT?

Complete toolchain	Cross platform	Programmability	Scalability	Holistic support	From HPC experts for HPC experts
Comprehensive set of tools for the whole application development process offering full system view.	Providing programing environment consistency across HPE and HPE Cray HPC platforms for easy porting and optimized performance from all types of HPE HPC architectures.	Offering users intuitive behavior, automation of tasks and best performance for their applications with the least amount of effort.	Improving performance of applications on systems of any size—up to Exascale deployments.	HPE Pointnext Services supports the whole suite, not just the tools we developed for investment protection.	Developed for over 30 years in close interaction and contributions from our users.

COMPILING ENVIRONMENT, PROGRAMMING LANGUAGES, AND MODELS

Compiling environment

Our **Fortran, C, and C++ compilers** are designed to help **extract maximum performance from the systems** regardless of the underlying architecture—supporting x86-64 processors (both Intel and AMD), Arm®-based HPC processors as well as accelerators.

The compilers identify regions of computation that are either sequential scalar or vector parallel, and automatically exploit these capabilities for the targeted system.

The compiler gives programmers optimization feedback with an annotated listing of source code for easier application tuning. The compiler also **integrates with the performance optimization tools** in the suite to help users get a better understanding of what changes will produce the ideal performant code.

The suite also includes **integration with GNU, Intel, AMD, Arm Alinea, and NVIDIA® programming environments** so developers can choose between multiple compilers and still use the libraries, debuggers, and performance analysis tools included in our suite to help optimize application performance.

We focus on standards compliance for **code safety, application portability, and investment protection**—our compilers support standard programming languages (Coarray Fortran, C/C++, and UPC) and standard programming models such as OpenMP and OpenACC.

Scalable communication libraries (MPI and SHMEM)

The suite offers full support for porting and developing distributed memory applications using HPE Cray MPI or performance-optimized SHMEM.

- **HPE Cray MPI** is an MPICH ABI compatible library that is tuned for Intel, AMD, and Arm CPUs and variety of GPUs. It is customized for low latency and high bandwidth, both on-node and off-node, for point-to-point and collective communications. Strategic optimizations for MPI I/O, MPI_THREAD_MULTIPLE, and remote memory access (RMA), and integration with the performance analysis tools in the suite contribute to deliver ideal application performance for today's HPC codes.



SCIENTIFIC, MATH, AND I/O LIBRARIES

The suite offers comprehensive collection of **highly tuned linear algebra subroutines** designed to help extract maximum performance from the system with the least amount of effort.

Customized **LibSci** (including BLAS, LAPACK, and ScaLAPACK), our iterative refinement toolkit, and **LibSci_ACC** (accelerated BLAS, and LAPACK) are designed to take full advantage of the underlying hardware, optimizing for both intra-node and inter-node performance on all HPE and HPE Cray HPC systems.

Using auto-tuning and adaptation, the libraries choose the known best algorithms at runtime. They also feature simplified interfaces into complex software (no source code changes required to access optimized algorithms) and integrate with the compiling environment for better productivity.

NetCDF and HDF5 **I/O libraries** are built with the supported compiling environments and included in the suite for convenience.

DEEP LEARNING PLUG-IN

With the plug-in, data scientists can easily perform deep learning training on HPE and HPE Cray systems—utilizing either CPUs (Intel or AMD) or GPUs. The tool helps them **achieve optimized scaling and performance for multiple machine learning frameworks**, such as TensorFlow™ and PyTorch.

The plug-in automatically determines which nodes to use, removing the burden of figuring out how many to use and where to put them. It is easy to launch—the users do not need to specify hosts and port numbers for workers. Adding parallelism is simple—data scientists can start from a serial training script and include the plug-in for parallelism.

DEBUGGING TOOLS

The HPE Cray Programming Environment offers traditional debuggers combined with new innovative techniques. Together, these technologies allow users to address debugging problems at a **broader range and scale than conventional techniques**. This means that programmers can spend less time debugging and more time creating.

Included are:

- **Comparative debugger:** This market-unique tool helps programmers uncover issue by running two applications side by side. If the values of the selected data structures diverge, the user is notified that an error may exist. This capability is useful **for locating errors that are introduced when applications are modified** through code, compiler, or library changes, and for application porting between architectures (for example, between CPUs and GPUs) or programming models.
- **GDB for HPC** is based on the popular GDB command-line debugger used to debug applications compiled with Fortran, C, and C++ compilers with enhancements to provide a GDB debugging experience for applications that run at scale across many nodes. The tool enables users to run a **traditional scalable debugging session**—either by launching an application or by attaching it to an already-running application. A GDB for HPC debug session retrieves debug information from thousands of processes and presents merged backtraces and data, removing vast amounts of duplicate information.
- **Valgrind for HPC: Parallel memory analysis tool** based on Valgrind debugger used for applications compiled with Fortran, C, and C++ compilers—it aggregates common errors into a single output record for easier analysis of potential memory problems within applications that run at scale.
- **Stack Trace Analysis Tool (STAT):** Helps developers identify if an application is hung or still making progress when running. Generates a merged backtrace for applications so users can get a better insight into application behavior at a function level.
- **Tool for abnormal termination processing (ATP):** When an application crashes, the tool detects a signal and generates a merged backtrace resulting in a minimal core file set so programmers do not have to plough through an enormous number of core files when debugging the application.

We also offer **support for traditional debugging mechanisms**, via integration with TotalView by Perforce and Arm Forge.



PERFORMANCE ANALYSIS AND OPTIMIZATION TOOLS

Comprehensive collection of tools designed to reduce the time and effort associated with porting and tuning of applications on HPE and HPE Cray systems.

We offer different tools and experiments to fit different developer needs and choice of interfaces for ease of use.

- **Performance Analysis Tool (PAT)** brings valuable insight when analyzing bottlenecks to improve performance of applications that run across the whole system. The tool exposes wide set of indicators, such as computation, communication, I/O, and memory statistics and displays a program's top time-consumers and bottlenecks (via unique and critical load balance analysis) for jobs at scale. It then **automatically generates observations and suggestions to improve code performance**.

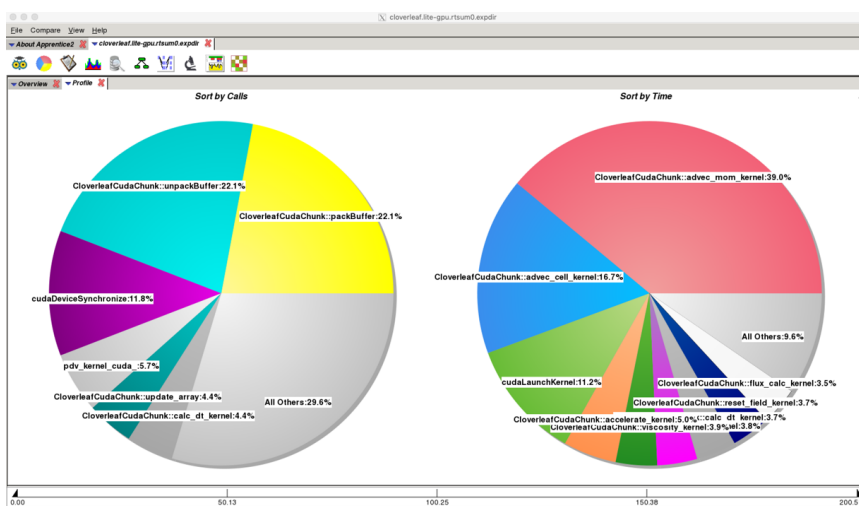


FIGURE 2. System statistics/performance indicators generated by PAT

- As ease of use is an important feature of the tool suite, both simple and advanced interfaces are available, offering both a simple path to get started and a wealth of capability for analyzing the most complex codes.
- Programmers can quickly assess the type and severity of the performance issues by using our **visualization tool** which complements text reports and summarizes programs' performance data in graphs and charts, allowing users to easily drill down to get to the bottom of issues.

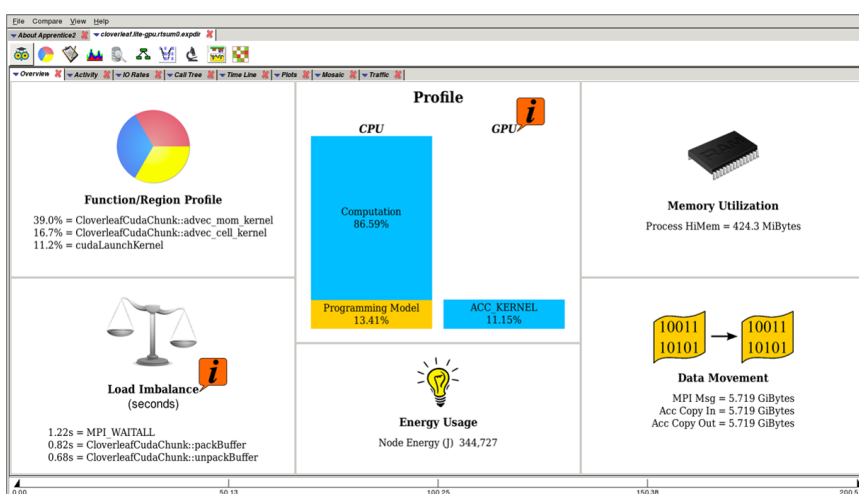


FIGURE 3. Overview of the GPU data in the visualization tool

- **Code parallelization assistant** helps developers **reveal hidden potential of their application via code restructuring**.

The tool extends our existing performance analysis, and visualization technology by combining performance statistics and program source code visualization with our compiling environment optimization feedback. This tool can easily navigate through source code to highlighted dependencies or bottlenecks during the optimization phase of program development or porting.

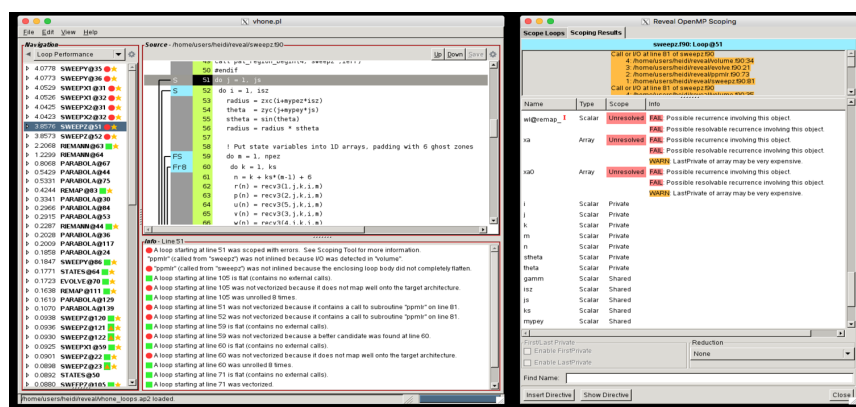


FIGURE 4. Parallelization assistant highlights bottlenecks and dependencies and brings up issues which need user's attention

Using the program library provided by our compiling environment and the performance data collected by our performance, measurement, and analysis tools, the user can navigate through their source code to understand which high-level loops could benefit from OpenMP parallelism. Code parallelization assistant provides dependency and variable scoping information for those loops and assists the user with creating parallel directives.

LEARN MORE AT
HPE Cray Programming Environment Product Page

**Make the right purchase decision.
Contact our presales specialists.**



Chat



Email



Call



Get updates

© Copyright 2020 Hewlett Packard Enterprise Development LP. The information contained herein is subject to change without notice. The only warranties for Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein.

AMD is a trademark of Advanced Micro Devices, Inc. Arm is a registered trademark of ARM Limited. Intel is a trademark of Intel Corporation in the U.S. and other countries. NVIDIA is a trademark and/or registered trademark of NVIDIA Corporation in the U.S. and other countries. All third-party marks are property of their respective owners.

a50002303FNW July 2020