# The Cray Programming Environment

**An Introduction** 

© Cray Inc 2013

### Vision

- Cray systems are designed to be High Productivity as well as High Performance Computers
- The Cray Programming Environment (PE) provides a simple consistent interface to users and developers.
  - Focus on improving scalability and reducing complexity

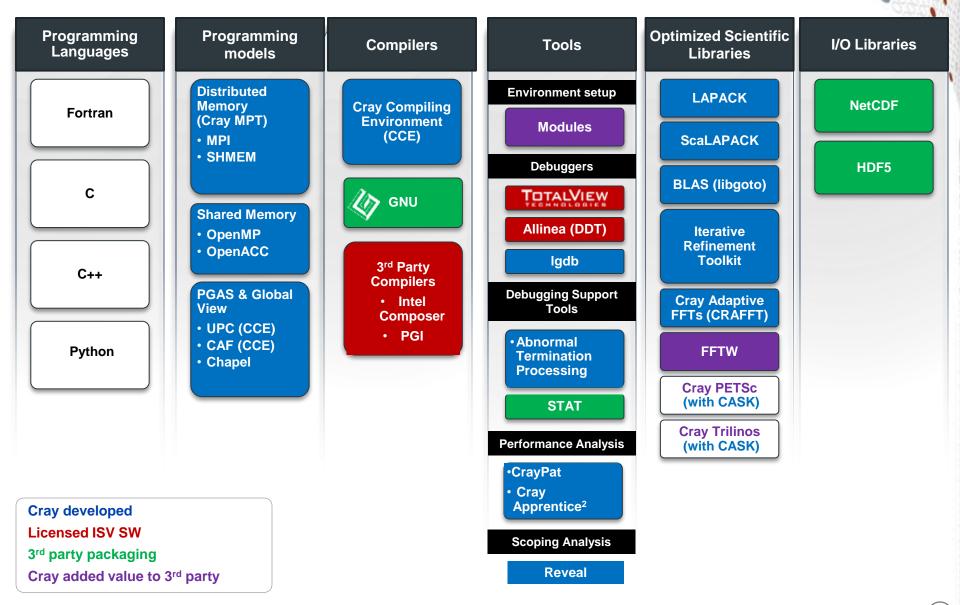
### • The default Programming Environment provides:

- the highest levels of application performance
- a rich variety of commonly used tools and libraries
- a consistent interface to multiple compilers and libraries
- an increased automation of routine tasks

### • Cray continues to develop and refine the PE

- Frequent communication and feedback to/from users
- Strong collaborations with third-party developers

# **Cray's Supported Programming Environment**



# The Cray Compilation Environment (CCE)

### • The default compiler on XE and XC systems

- Specifically designed for HPC applications
- Takes advantage of Cray's experience with automatic vectorization and and shared memory paralleization
- Excellent standards support for multiple languages and programming models
  - Fortran 2008 standards compliant
  - C++98/2003 compliant (working on C++11)
  - OpenMP 3.1 compliant, working on OpenMP 4.0
  - OpenACC 1.0 compliant (working on OpenACC 2.0)



### • Full integrated and optimised support for PGAS languages

- UPC 1.3 and Fortran 2008 coarray support
- No preprocessor involved
- Full debugger support (With Allinea DDT)

### OpenMP and automatic multithreading fully integrated

- Share the same runtime and resource pool
- Aggressive loop restructuring and scalar optimization done in the presence of OpenMP
- Consistent interface for managing OpenMP and automatic multithreading

# **Cray MPI & SHMEM**

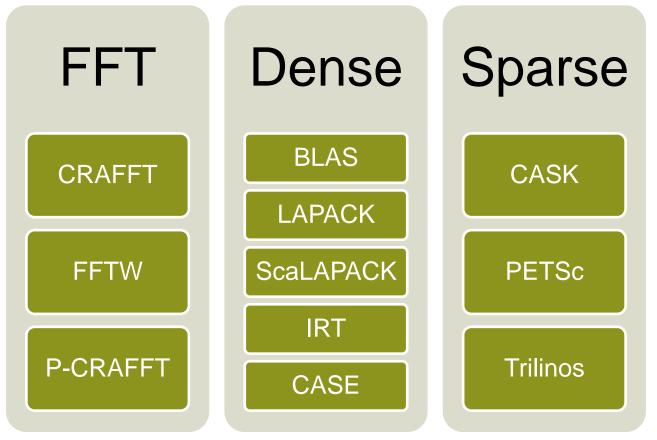
# Cray MPI

- Implementation based on MPICH2 from ANL
- Includes many improved algorithms and tweaks for Cray hardware
  - Improved algorithms for many collectives
  - Asynchronous progress engine allows overlap of computation and comms
  - Customizable collective buffering when using MPI-IO
  - Optimized Remote Memory Access (one-sided) fully supported including passive RMA
- Full MPI-2 support with the exception of
  - Dynamic process management (MPI\_Comm\_spawn)
- Much of MPI-3 supported

# • Cray SHMEM

- Fully optimized Cray SHMEM library supported
  - Fully compliant with OpenSHMEM v1.0
  - Cray XE and XC implementation close to the T3E model

### **Cray Scientific Libraries**



IRT – Iterative Refinement Toolkit CASK – Cray Adaptive Sparse Kernels CRAFFT – Cray Adaptive FFT CASE – Cray Adaptive Simplified Eigensolver

# **Cray Performance Analysis Tools (PAT)**

- From performance measurement to performance analysis
- Assist the user with application performance analysis and optimization
  - Help user identify important and meaningful information from potentially massive data sets
  - Help user identify problem areas instead of just reporting data
  - Bring optimization knowledge to a wider set of users

### • Focus on ease of use and intuitive user interfaces

- Automatic program instrumentation
- Automatic analysis

### Target scalability issues in all areas of tool development

## **Debuggers on Cray Systems**

- Systems with hundreds of thousands of threads of execution need a new debugging paradigm
  - Innovative techniques for productivity and scalability
    - Scalable Solutions based on MRNet from University of Wisconsin
    - STAT Stack Trace Analysis Tool
      - Scalable generation of a single, merged, stack backtrace tree
      - running at 216K back-end processes
      - ATP Abnormal Termination Processing
        - Scalable analysis of a sick application, delivering a STAT tree and a minimal, comprehensive, core file set.
    - Fast Track Debugging
      - Debugging optimized applications
      - Added to Allinea's DDT 2.6 (June 2010)
    - Comparative debugging
      - A data-centric paradigm instead of the traditional control-centric paradigm
      - Collaboration with Monash University and University of Wisconsin for scalability
  - Support for traditional debugging mechanism
    - TotalView, DDT, and gdb

# An introduction to modules

### **Environment Setup**

- Cray systems use modules in the user environment to support multiple software versions and to create integrated software packages
  - As new versions of the supported software and associated man pages become available, they are added automatically to the Programming Environment as a new version, while earlier versions are retained to support legacy applications
  - You can use the default version of an application, or you can choose another version by using Modules system commands

# The module tool on the Cray XE

• How can we get appropriate Compiler, Tools, and Libraries?

- The modules tool is used to handle different versions of packages
  - e.g.: module load compiler\_v1
  - e.g.: module swap compiler\_v1 compiler\_v2
  - e.g.: module load perftools

# • Taking care of changing of PATH, MANPATH, LM\_LICENSE\_FILE,.... environment

- Modules also provide a simple mechanism for updating certain environment variables, such as PATH, MANPATH, and LD\_LIBRARY\_PATH
- In general, you should make use of the modules system rather than embedding specific directory paths into your startup files, makefiles, and scripts.

#### It is also easy to setup your own modules for your own software

# **Useful module commands**

### • Which modules are available?

module avail, module avail cce

### • Which modules are loaded?

module list

## Load software

module load perftools

### Change programming environment

module swap PrgEnv-cray PrgEnv-gnu

### Change software version

- module swap cce/8.0.2 cce/7.4.4
- Display module help/release notes
  - module help cce

### Show module environment variables

module show cce

# Which SW Products and Versions Are Available

#### • avail [avail-options] [path...]

• List all available modulefiles in the current MODULEPATH

### Useful options for filtering

- -U, --usermodules
  - List all modulefiles of interest to a typical user
- -D, --defaultversions
  - List only default versions of modulefiles with multiple available versions
- -P, --prgenvmodules
  List all PrgEnv modulefiles
- -T, --toolmodules
  - List all tool modulefiles
- -L, --librarymodules
  - List all library modulefiles
- % module avail <product>
  - List all <product> versions available