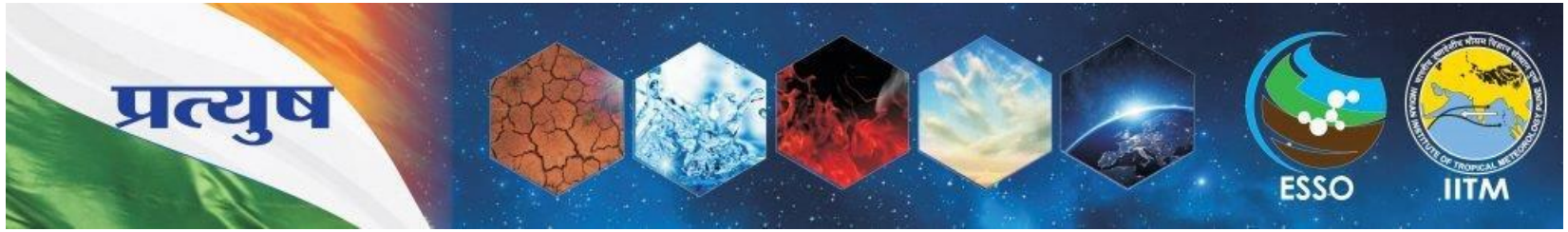
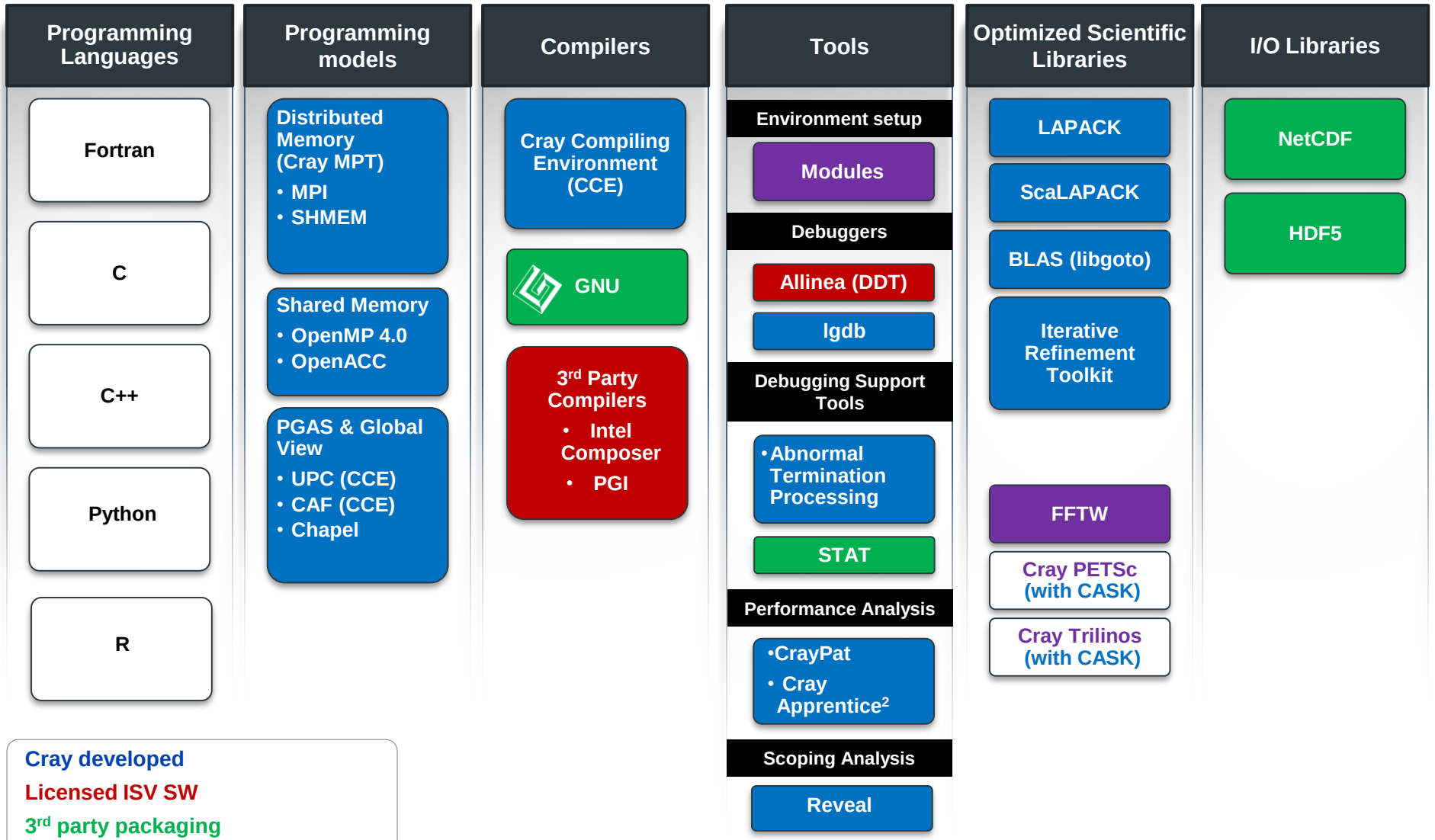
Cray Software



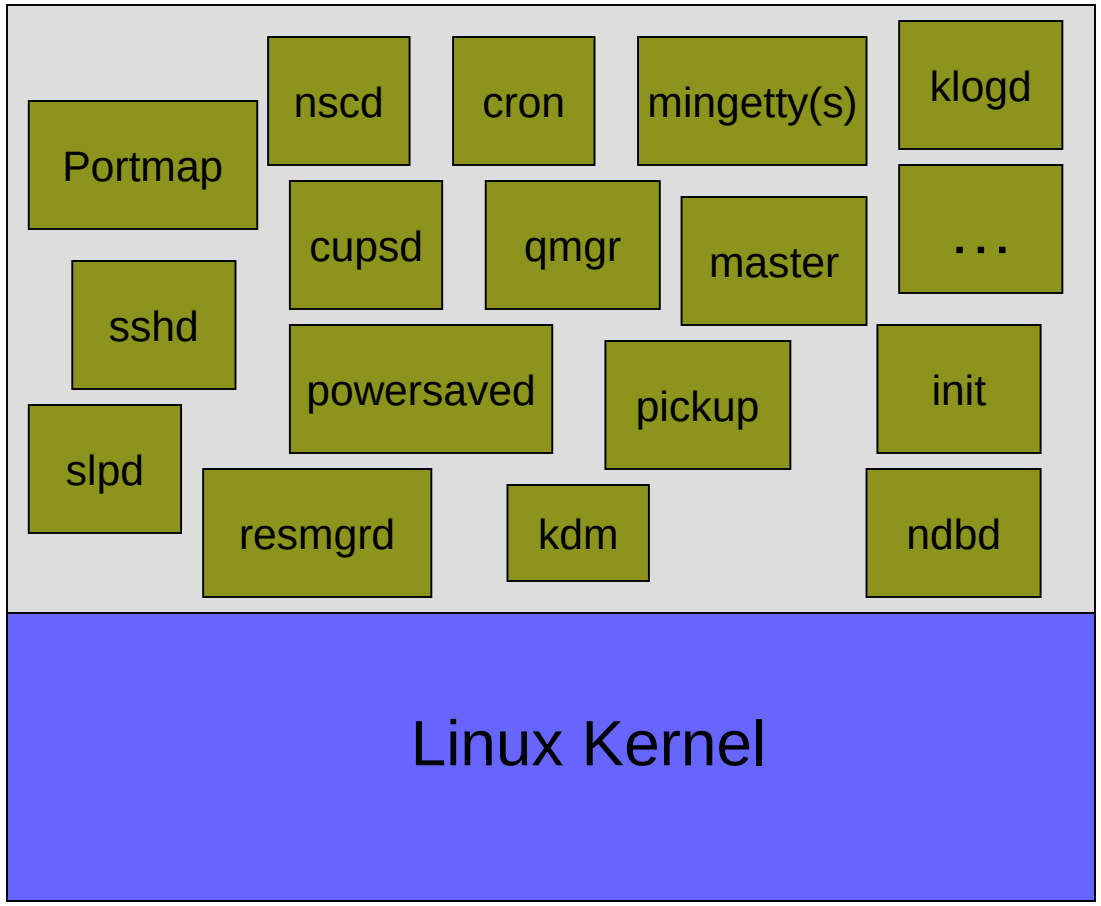
Cray's Supported Programming Environment



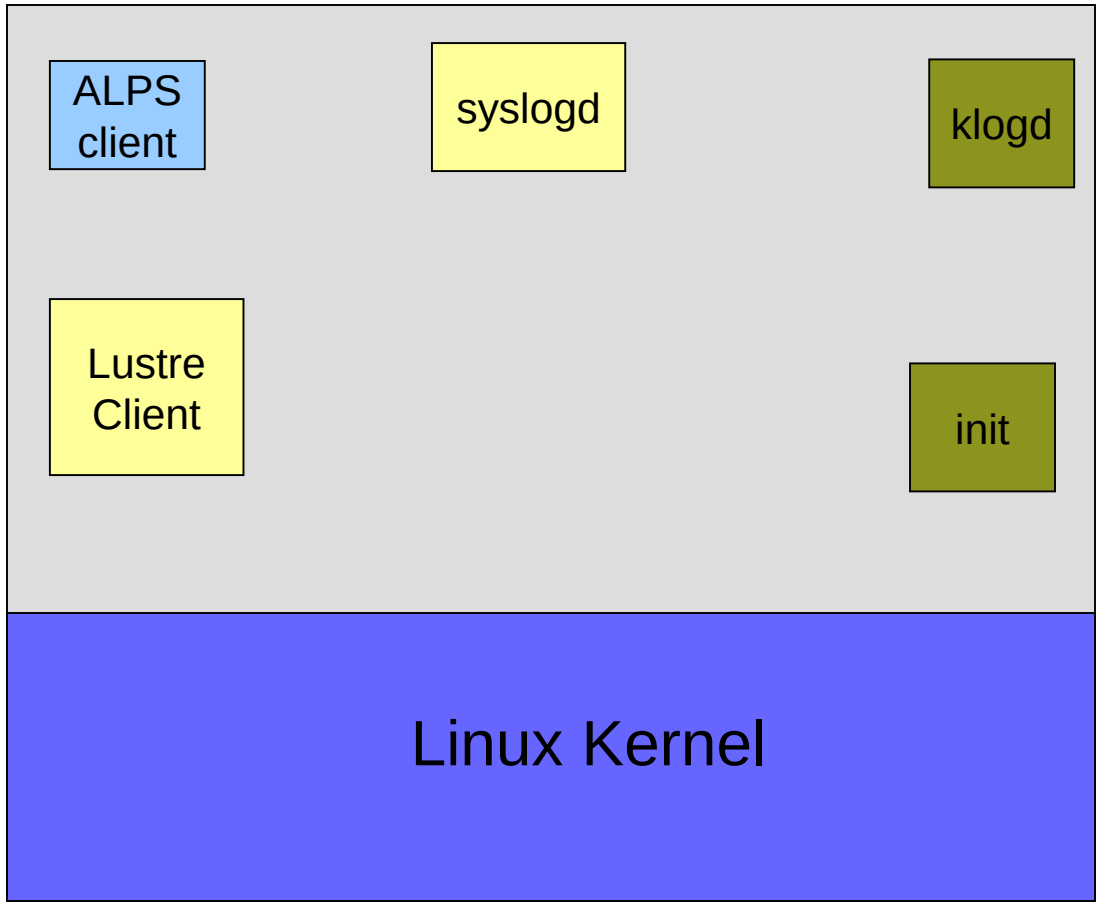
Vision

- **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

Trimming OS – *Standard Linux Server*



Linux on a Diet – *CLE*



CLE Can Adapt to Different Application Requirements

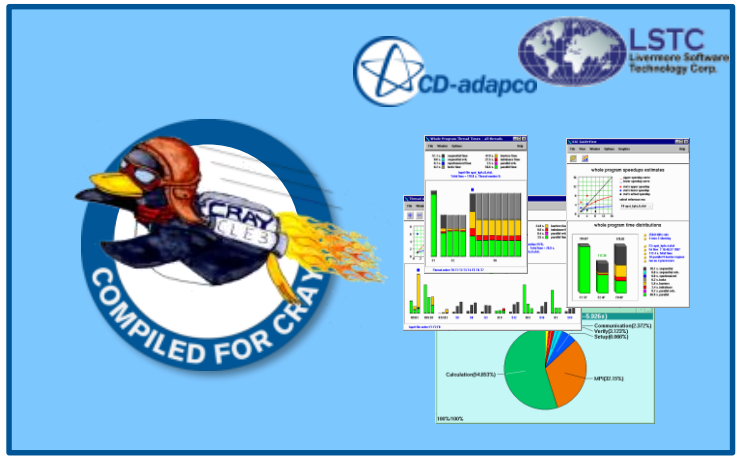
CRAY LINUX ENVIRONMENT

CCM *–Cluster Compatibility Mode*

- No compromise *compatibility*
- Fully standard x86/Linux
- Standardized Communication Layer
- Out-of-the-box ISV Installation
- ISV applications simply install and run

ESM *– Extreme Scalability Mode*

- No compromise *scalability*
- Low-Noise Kernel for scalability
- Native Comm. & Optimized MPI
- Application-specific performance tuning and scaling



CLE run mode is set by the user on a job-by-job basis to provide full flexibility

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 shared memory parallelization

- **Excellent standards support for multiple languages and programming models**
  - Fortran 2008 standards compliant
  - C++11 compliant
  - OpenMP 4.5
  - OpenACC 2.0 compliant

- **Full integrated and optimised support for PGAS languages**
  - UPC 1.2 and Fortran 2008 coarray support
  - No preprocessor involved
  - Full debugger support (With Allinea DDT)

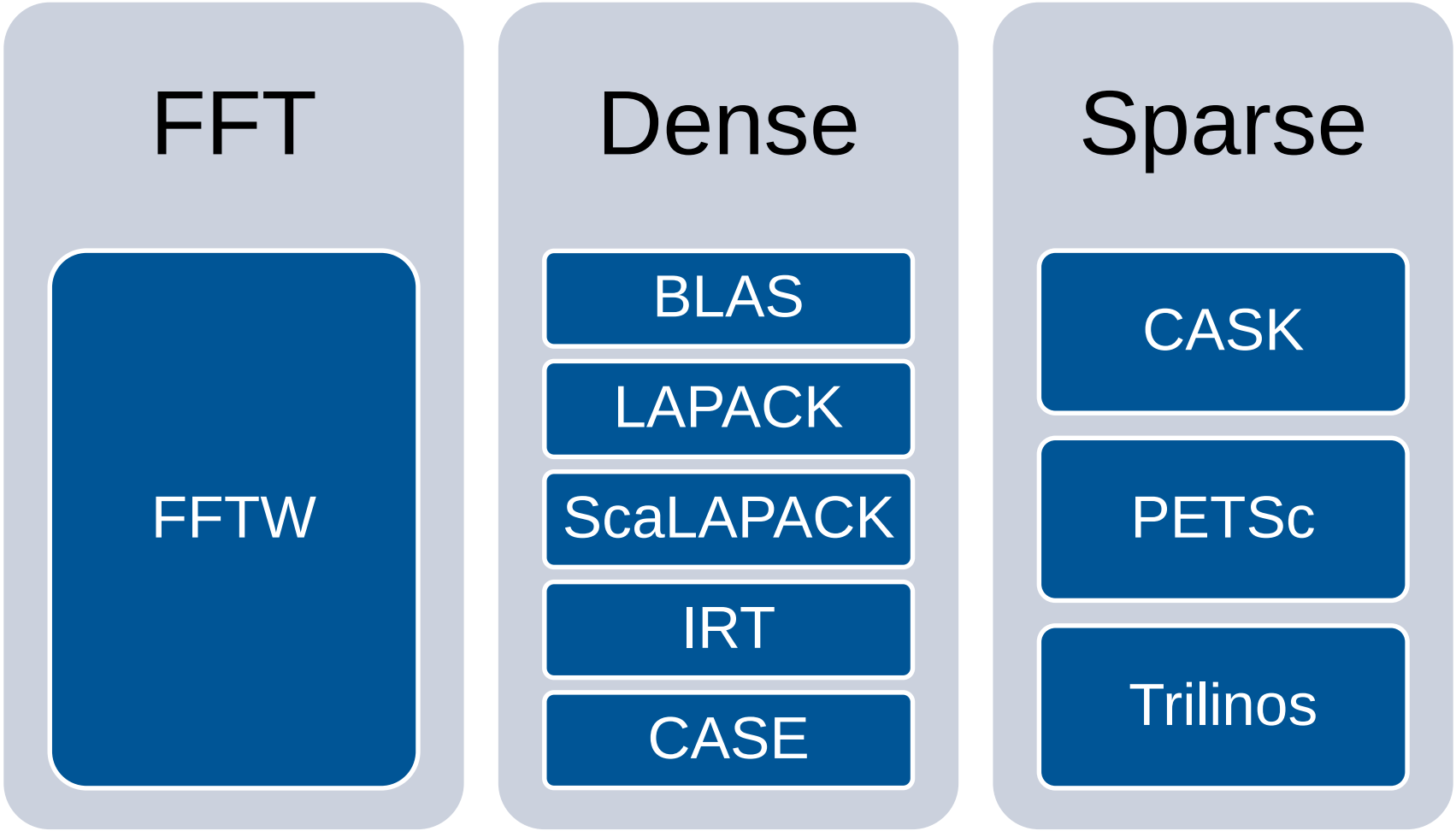
- **OpenMP and automatic multithreading fully integrated**
  - Share the same runtime and resource pool
  - Aggressive loop restructuring and scalar optimization done in the presence of OpenMP
  - Consistent interface for managing OpenMP and automatic multithreading

Cray MPI & SHMEM

- **Cray MPI**
  - Implementation based on MPICH3 source 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-3 support with the exception of
    - Dynamic process management (eg. MPI_Comm_spawn)
    - MPI_LONG_DOUBLE and MPI_C_LONG_DOUBLE_COMPLEX for CCE
  - Includes support for Fortran 2008 bindings (from CCE 8.3.3)

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

Cray Scientific Libraries



IRT – Iterative Refinement Toolkit
CASK – Cray Adaptive Sparse Kernels
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
      - **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
    - DDT, gdb and TotalView

An introduction to modules

What are Environment Modules?

- Provides for the dynamic modification of a user's environment via modulefiles
- Each modulefile contains the information needed to configure the shell for an application
  - Typically alter or set shell environment variables such as PATH, MANPATH, etc.
- Modules can be **loaded** and **unloaded** dynamically and atomically, in an clean fashion
- All popular shells are supported
  - including *bash*,*ksh*, *zsh*, *sh*, *csh*, *tcsh*, as well as some scripting languages such as *perl* and *python*
- Useful in managing different applications and versions of applications
- Can be bundled into **metamodules**
  - load an entire suite of different applications
- Check **http://modules.sourceforge.net/**

Environment Setup

- The Cray XC system uses 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
- Users can create their own modules or admins can install sitespecific modules available to many users
- Modules are very flexible and powerful and allow the user to dynamically manage their programming environment

Viewing the current module state

- Each login session has its own module state which can be modified by loading, swapping or unloading the available *modules*
- This state affects the functioning of the compiler wrappers and in some cases runtime of applications
- A standard, default set of modules is always loaded at login for all users
- Current state can be viewed by running:

```
$> module list
```

Viewing available modules

- **There may be many hundreds of possible modules available to users**
  - Beyond the pre-loaded defaults there are many additional packages provided by Cray
  - Sites may choose to install their own versions

- **Users can see all the modules that can be loaded using the command:**

```
$> module avail
```

- **Searches can be narrowed by passing the first few characters of the desired module**

Modifying the default environment

Loading, swapping or unloading modules:

- The default version of any individual module can be loaded by name

  e.g.: `module load cce`

- A specific version can be specified after the forward slash

  e.g.: `module load cce/8.6.3`

- Modules can be swapped out in place

  e.g.: `module swap cce cce/8.6.0`

- Or removed entirely

  e.g.: `module unload perftools`

Modifying the default environment

- **Modules will automatically change values of variables like `PATH`, `MANPATH`, `LM_LICENSE_FILE`... Etc**
  - 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

More module commands

`$> module list`	• Prints actual loaded modules
`$> module avail [-S str]`	• Prints all module available containing the specified **str**ing
`$> module (un)load [mod_name/version]`	• Adds or remove a module to the actual loaded list • If no version specified, loading the default version
`$> module switch [mod1] [mod2]`	• Unload mod1 and load mod2 • e.g. to change versions of loaded modules
`$> module whatis/help [mod]`	• Prints the module (short) description
`$> module show [mod]`	• Prints the environmental modification
`$> module load user_own_modules`	• add $HOME/privatemodules to the list of directories that the module command will search for modules

What module does ?

```
crayadm@elogin04:~> module show cce
-------------------------------------------------------------------
/opt/cray/pe/modulefiles/cce/8.6.3:

conflict        cce
setenv          GCC_X86_64 /opt/gcc/6.1.0/snos
setenv          CRAY_BINUTILS_ROOT_X86_64 /opt/cray/pe/cce/8.6.3/binutils/x86_64/x86_64-pc-linux-gnu/../
setenv          CRAY_BINUTILS_BIN_X86_64 /opt/cray/pe/cce/8.6.3/binutils/x86_64/x86_64-pc-linux-gnu/bin
setenv          LINKER_X86_64 /opt/cray/pe/cce/8.6.3/binutils/x86_64/x86_64-pc-linux-gnu/bin/ld
setenv          ASSEMBLER_X86_64 /opt/cray/pe/cce/8.6.3/binutils/x86_64/x86_64-pc-linux-gnu/bin/as
setenv          FTN_X86_64 /opt/cray/pe/cce/8.6.3/cce/x86_64
setenv          CC_X86_64 /opt/cray/pe/cce/8.6.3/cce/x86_64
setenv          CRAY_CXX_IPA_LIBS_X86_64 /opt/cray/pe/cce/8.6.3/cce/x86_64/lib/libcray-c++-rts.a
setenv          CRAYLIBS_X86_64 /opt/cray/pe/cce/8.6.3/cce/x86_64/lib
prepend-path    INCLUDE_PATH_X86_64 /opt/cray/pe/cce/8.6.3/cce/x86_64/include/craylibs
setenv          GCC_AARCH64 /opt/gcc-cross-aarch64/6.1.0/aarch64
setenv          CRAY_BINUTILS_ROOT_AARCH64 /opt/cray/pe/cce/8.6.3/binutils/cross/x86_64-aarch64/aarch64-unknowun-linux-gnu/../
setenv          CRAY_BINUTILS_BIN_AARCH64 /opt/cray/pe/cce/8.6.3/binutils/cross/x86_64-aarch64/aarch64-unknowun-linux-gnu/bin
setenv          LINKER_AARCH64 /opt/cray/pe/cce/8.6.3/binutils/cross/x86_64-aarch64/aarch64-unknowun-linux-gnu/bin/ld
setenv          ASSEMBLER_AARCH64 /opt/cray/pe/cce/8.6.3/binutils/cross/x86_64-aarch64/aarch64-unknowun-linux-gnu/bin/as
setenv          CRAY_CXX_IPA_LIBS_AARCH64 /opt/cray/pe/cce/8.6.3/cce/aarch64/lib/libcray-c++-rts.a
setenv          CRAYLIBS_AARCH64 /opt/cray/pe/cce/8.6.3/cce/aarch64/lib
prepend-path    INCLUDE_PATH_AARCH64 /opt/cray/pe/cce/8.6.3/cce/aarch64/include/craylibs
setenv          CRAYLMD_LICENSE_FILE /opt/cray/pe/cce/cce.lic
setenv          CRAY_BINUTILS_ROOT /opt/cray/pe/cce/8.6.3/binutils/x86_64/x86_64-pc-linux-gnu/../
setenv          CRAY_BINUTILS_VERSION /opt/cray/pe/cce/8.6.3
setenv          CRAY_BINUTILS_BIN /opt/cray/pe/cce/8.6.3/binutils/x86_64/x86_64-pc-linux-gnu/bin
setenv          CRAY_CCE_SHARE /opt/cray/pe/cce/8.6.3/cce/x86_64/share
setenv          CRAY_CXX_IPA_LIBS /opt/cray/pe/cce/8.6.3/cce/x86_64/lib/libcray-c++-rts.a
setenv          CRAY_FTN_VERSION 8.6.3
setenv          CRAY_CC_VERSION 8.6.3
```

“Meta”-Module PrgEnv-X

- **PrgEnv-X is a “meta”-module**
  - loading several modules,
    - including the compiler,
    - the corresponding mathematical libs,
    - MPI,
    - system environment needed for the compiler wrappers

```
crayadm@elogin04:~> module show PrgEnv-cray
-------------------------------------------------------------
/opt/cray/pe/modulefiles/PrgEnv-cray/6.0.4:

conflict        PrgEnv
conflict        PrgEnv-x1
conflict        PrgEnv-x2
conflict        PrgEnv-gnu
conflict        PrgEnv-intel
conflict        PrgEnv-pgi
conflict        PrgEnv-pathscale
conflict        PrgEnv-cray
setenv          PE_ENV CRAY
prepend-path    PE_PRODUCT_LIST CRAY
setenv          cce_already_loaded 1
module          load cce/8.6.3
setenv          craype_already_loaded 1
module          swap craype/2.5.13
module          swap cray-mpich cray-mpich/7.6.3
module          load cray-libsci
module          load udreg
module          load ugni
```

Targeting different node types

- **Compiling for the CPU nodes**
  - module load craype-broadwell
    (enables the Broadwell specific instructions. Default is x86_64)
- **Compiling for KNL nodes**
  - module load craype-mic-knl
- **Compiling for the GPU nodes**
  - module load craype-accel-nvidia35
  - “module display craype-accel-nvidia35” tells you that this module also loads cudatoolkit and cray-libsci-acc

Summary

- **Various applications in various versions available**

```
$> module avail          # lists all
$> module avail cce      # cce*
```

- **Dynamic modification of a user's environment**

```
$> module (un)load PRODUCT/MODULE
```

  - E.g. PrgEnv-xxx changes compilers, linked libraries, and environment variables

- **Version management**

```
$> module switch prod_v1 prod_v2
$> module switch PrgEnv-cray PrgEnv-gnu
$> module switch cce  cce/8.6.0
```

- **Metamodules bundles multiple modules**
- **Can create your own (meta)modules**

- **Module tool take care**
  - Environment variables
    - PATH, MANPATH, LD_LIBRARY_PATH, LM_LICENSE_FILE,....
  - Taking care of compiler and linker arguments of loaded products
    - Include paths, linker paths, …