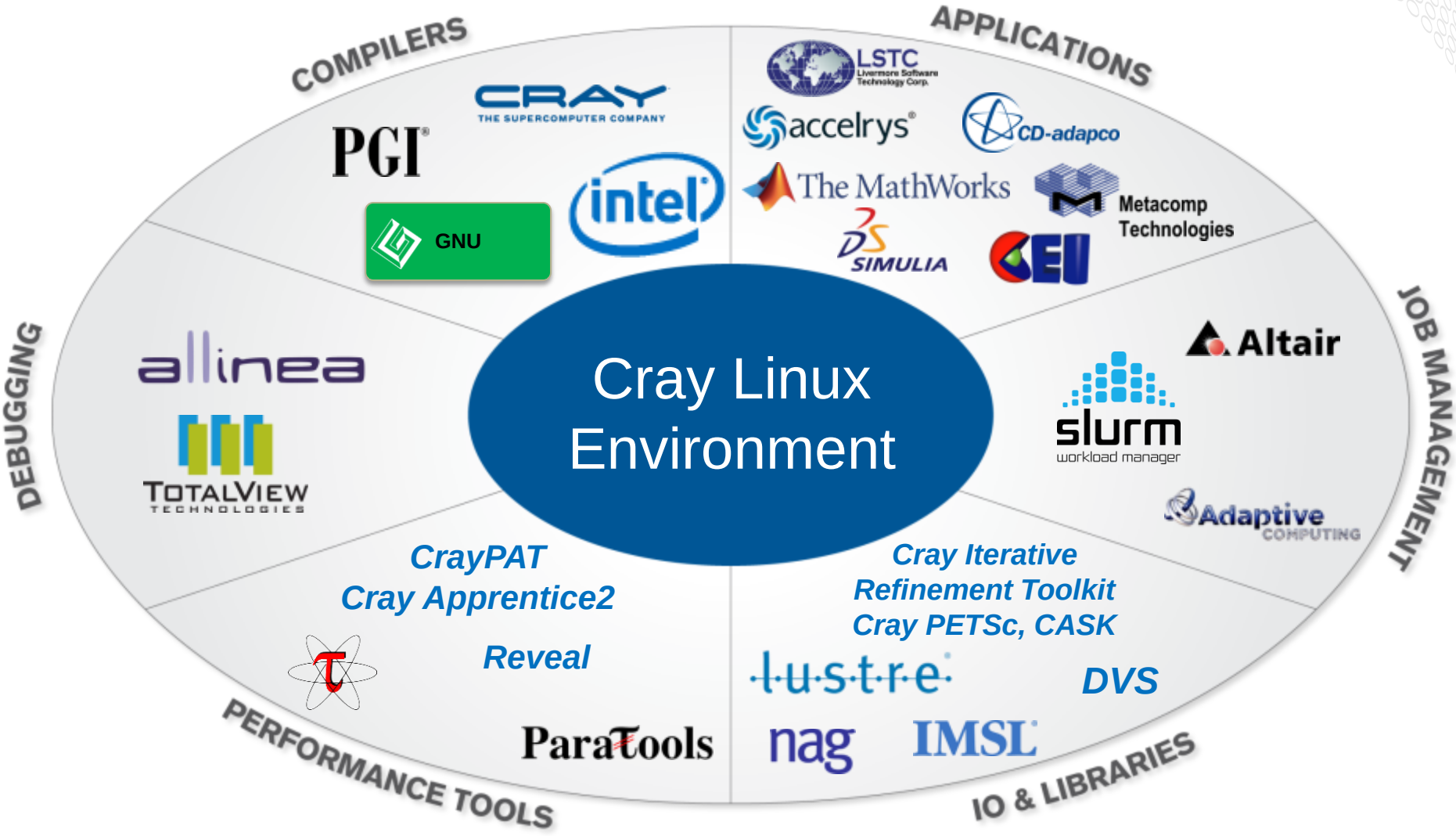
Programming Environment

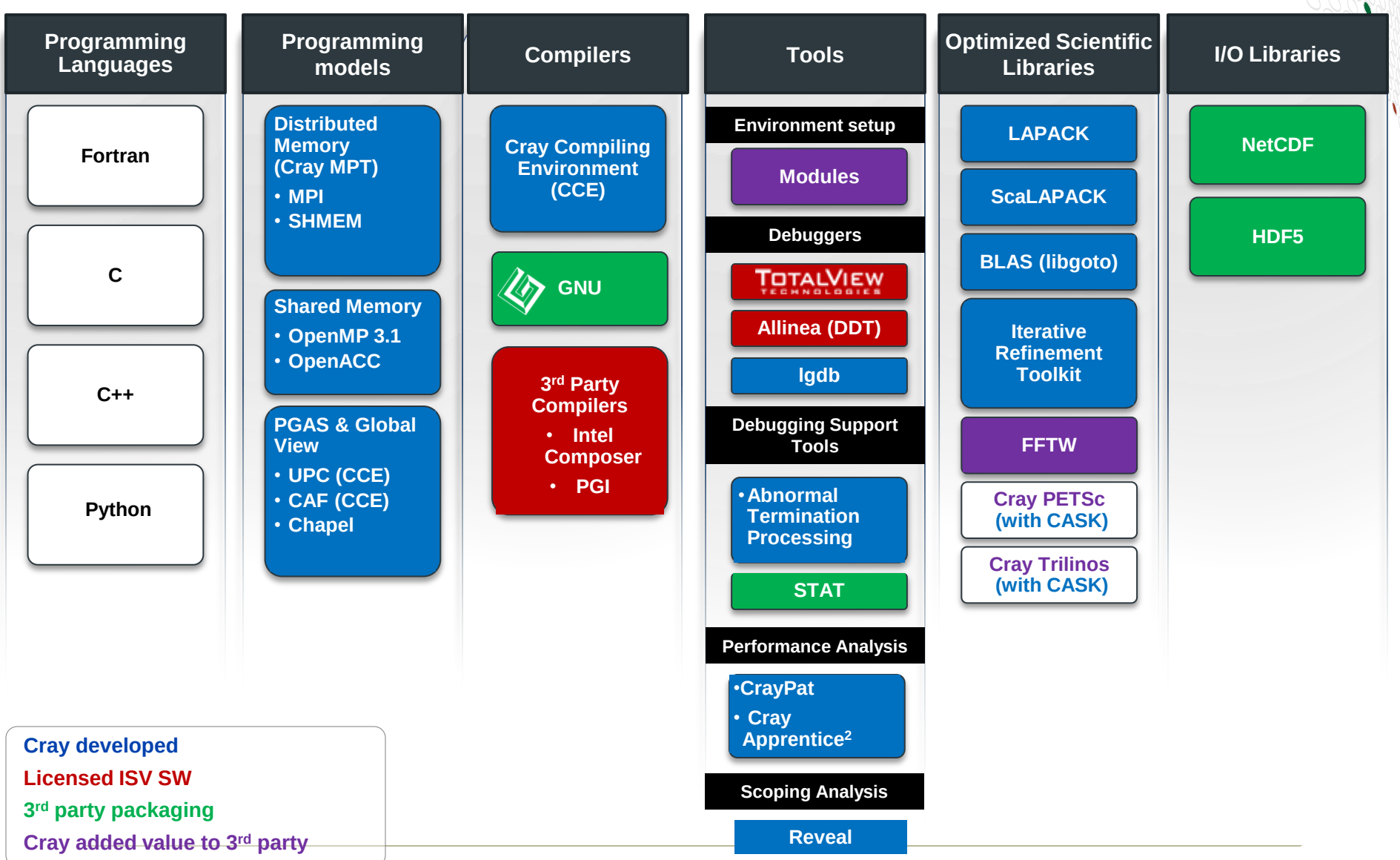
Vision

- **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

Cray Software Ecosystem



Cray's Supported Programming Environment



Cray developed
Licensed ISV SW
3rd party packaging
Cray added value to 3rd party

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 parallelization

- **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 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)

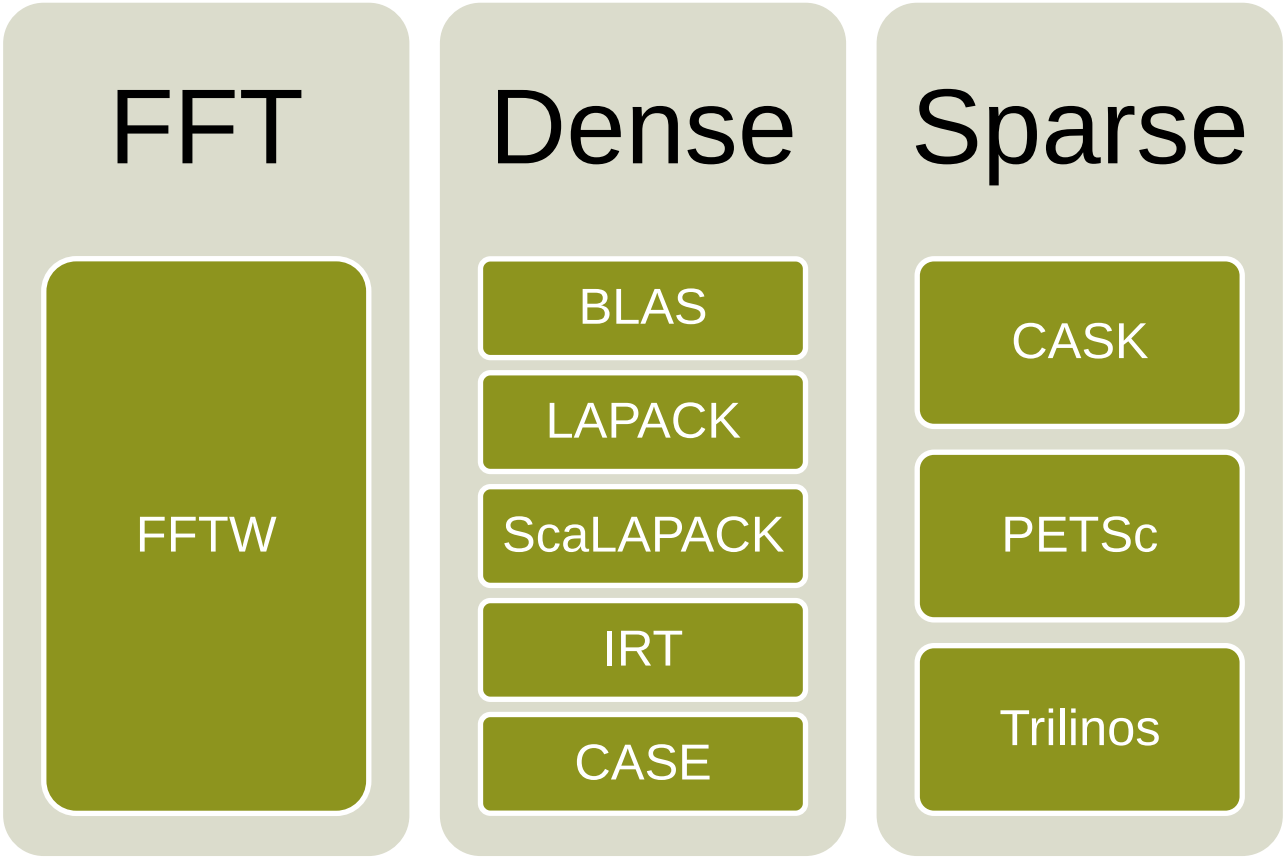
- **OpenMP and automatic multithreading fully integrated**
  - Share the same runtime and resource pool
  - Aggressive loop restructuring and scalar optimization done in the presence of OpenMP
  - Consistent interface for managing OpenMP and automatic multithreading

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 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

Controlling the environment with modules

Modules

- The Cray Programming Environment uses the GNU “modules” framework 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 installed and added to the Programming Environment as a new version, while earlier versions are retained to support legacy applications
- System administrators will set the default version of an application, or you can choose another version by using modules system commands
- Users can create their own modules, or administrators can install site specific modules available to many users

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
```

Default modules example

```
ccb-login2:crayhr$ module list
Currently Loaded Modulefiles:
  1) modules/3.2.6.7
  2) eswrap/1.1.0-1.020200.1130.0
  3) switch/1.0-1.0502.50885.3.4.ari
  4) craype-network-aries
  5) craype/2.1.1
  6) cce/8.2.7
  7) cray-libsci/12.2.0
  8) udreg/2.3.2-1.0502.8413.2.9.ari
  9) ugni/5.0-1.0502.8670.4.22.ari
 10) pmi/5.0.3-1.0000.9981.128.2.ari
 11) dmapp/7.0.1-1.0502.8638.9.93.ari
 12) gni-headers/3.0-1.0502.8554.6.6.ari
 13) xpmem/0.1-2.0502.50559.4.2.ari
 14) job/1.5.5-0.1_2.0502.49000.2.39.ari
 15) csa/3.0.0-1_2.0502.49605.4.45.ari
 16) dvs/2.4_0.9.0-1.0502.1696.2.39.ari
 17) alps/5.2.0-2.0502.8594.12.4.ari
 18) rca/1.0.0-2.0502.49765.5.41.ari
 19) atp/1.7.2
 20) PrgEnv-cray/5.2.14
 21) pbs/12.2.401.141761
 22) craype-ivybridge
 23) cray-mpich/6.3.1
```

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, e.g.**

```
ccb-login2:crayhr$ module avail gcc

------------------------------ /opt/modulefiles ---------------------------
gcc/4.8.0          gcc/4.8.1          gcc/4.8.2(default) gcc/4.9.0
```

Further refining available modules

- `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

module commands and standard output

- Be aware that module commands output to standard error
- This makes it tricky to search the (voluminous) module avail output

- csh/tcsh

```
module avail >& mavail.txt ; grep netcdf mavail.txt
( module avail >/dev/null )  |& grep netcdf
```

- ksh

```
module avail 2> mavail.txt ; grep netcdf mavail.txt
module avail 2>&1 | grep netcdf
```

Modifying the default environment

- **Loading, swapping or unloading modules:**
  - The default version of any inidividual module can be loaded by name
    - e.g.: `module load perftools`
  - A specific version can be specified after the forward slash
    - e.g.: `module load perftools/6.1.0`
  - Modules can be swapped out in place
    - e.g.: `module swap intel intel/13.1.1.163`
  - Or removed entirely
    - e.g.: `module unload perftools`
- **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

Tips for modules

- **Put module list in job scripts**
  - This gives you a record of job context weeks or years later
- **If you want to test for the programming environment**
  - Test the PE_ENV environment variable (but there are no guarantees this won‘t change on eday)
- **Use the module information to find documentation**
  - % module load intel
  - % module show intel (look at output for interesting envars)
  - % ls $INTEL_PATH

```
bin/        Documentation/    ipp/  mpirt/    tbb/
uninstall.sh*
compiler/  eclipse_support/  man/  pkg_bin@  uninstall/
debugger/  foldermap.sc.xml  mkl/  Samples/
uninstall_GUI.sh*
```

Summary of useful module commands

- **Which modules are available?**
  - `module avail, module avail cce`
- **Which modules are currently loaded?**
  - `module list`
- **Load software**
  - `module load perftools`
- **Change programming environment**
  - `module swap PrgEnv-cray PrgEnv-gnu`
- **Change software version**
  - `module swap cce/8.3.4  cce/8.3.7`
- **Unload module**
  - `module unload cce`
- **Display module release notes**
  - `module help cce`
- **Show summary of module environment changes**
  - `module show cce`

Compiling Applications for Cray XC

Compiler driver wrappers

- All applications that will run in parallel on the Cray XC should be compiled with the standard language wrappers
- The compiler drivers for each language are:
  - `cc` – wrapper around the C compiler
  - `CC` – wrapper around the C++ compiler
  - `ftn` – wrapper around the Fortran compiler
- These scripts will choose the required compiler version, target architecture options, scientific libraries and their include files automatically from the module environment.
- Use them exactly like you would the original compiler, e.g. To compile `prog1.f90` run

```
ftn -c prog1.f90
```

Compiler driver wrappers

- **The scripts choose which compiler to use from the PrgEnv module loaded**

PrgEnv	Description	Real Compilers
PrgEnv-cray	Cray Compilation Environment	crayftn, craycc, crayCC
PrgEnv-intel	Intel Composer Suite	ifort, icc, icpc
PrgEnv-gnu	GNU Compiler Collection	gfortran, gcc, g++
PrgEnv-pgi	Portland Group Compilers	pgf90, pgcc, pgCC

- **Use module swap to change PrgEnv, e.g.**
  - `module swap PrgEnv-cray PrgEnv-intel`
- **PrgEnv-cray is loaded by default at login**
  - This may differ on other Cray systems
  - use `module list` to check what is currently loaded
- **The Cray MPI module is loaded by default (cray-mpich)**
- **To support SHMEM load the cray-shmem module**
  - To compile a pure SHMEM code, unload the cray-mpich module

Compiler versions

- **There are usually multiple versions of each compiler available to users.**
  - The most recent version is usually the default and will be loaded when swapping `PrgEnvs`.
  - To change the version of the compiler in use, swap the Compiler Module. e.g. `module swap cce/8.3.4 cce/8.3.7`

PrgEnv	Compiler Module
PrgEnv-cray	cce
PrgEnv-intel	intel
PrgEnv-gnu	gcc

About the –I, –L and –l flags

- **For libraries and include files covered by module files, you should not add anything to your Makefile**
  - No additional MPI flags are needed (included by wrappers)
  - You do not need to add any `-I`, `-l` or `–L` flags for the Cray provided libraries
- **If your Makefile needs an input for `–L` to work correctly, try using '.'**
- **If you really need a specific path, try checking 'module show X' for some environment variables**

OpenMP

- **OpenMP is supported by all of the PrgEnvs**
  - CCE (PrgEnv-cray) recognizes and interprets OpenMP directives by default. If you have OpenMP directives in your application but do not wish to use them, disable OpenMP recognition with –hnoomp.

PrgEnv	Enable OpenMP	Disable OpenMP
PrgEnv-cray	(-homp)	-hnoomp
PrgEnv-intel	-openmp	
PrgEnv-gnu	-fopenmp	

Compiler man pages

- **For more information on individual compilers**

PrgEnv	C	C++	Fortran
PrgEnv-cray	man craycc	man crayCC	man crayftn
PrgEnv-intel	man icc	man icpc	man ifort
PrgEnv-gnu	man gcc	man g++	man gfortran
Wrappers	man cc	man CC	man ftn

- **To verify that you are using the correct version of a compiler, use:**
  - -V option on a cc, CC, or ftn command with PGI, Intel and Cray
  - -dumpversion option on a cc, CC, or ftn command with GNU