# The Cray Programming Environment: Current Status and Future Directions


Luiz DeRose

Cray Programming Environments Director

# Legal Disclaimer

*Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.*

*Cray Inc. may make changes to specifications and product descriptions at any time, without notice.*

*All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.*

*Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.*

*Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.*

*Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.*

*The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, and URIKA. The following are trademarks of Cray Inc.: APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, THREADSTORM. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT, and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used in this document are the property of their respective owners.*


Copyright 2016 Cray Inc.

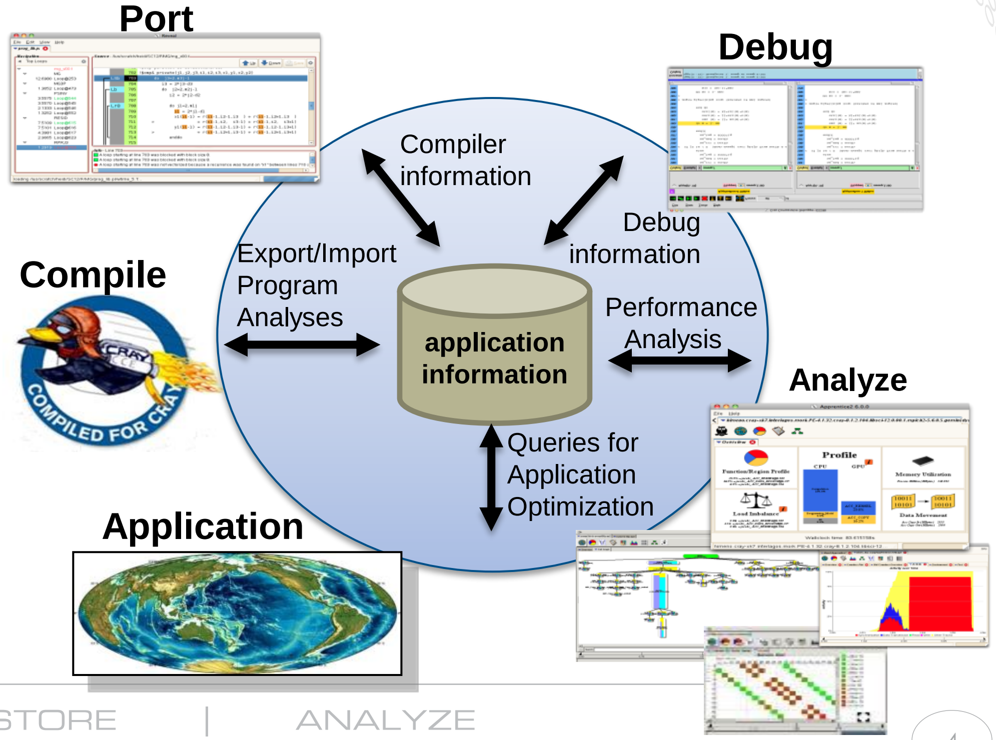
# Agenda

- **Cray PE Overview**
- **Highlights since last CUG and roadmap overview**
  - CCE
  - MPT
  - CPMAT
  - CDST
  - CSML
  - CrayPE & Modules
  - Chapel
- **Cray Directives**
  - Accelerators
  - Memory directives
- **Summary**


Copyright 2016 Cray Inc.

# The Cray Programming Environment Mission

- Focus on **Performance** and **Programmability**
  - It is the role of the Programming Environment to **close the gap** between observed performance and achievable performance
- Support the **application development life cycle** by providing a **tightly coupled** environment with compilers, libraries, and tools that will **hide the complexity** of the system
  - Address issues of scale and complexity of HPC systems
  - Target **ease of use** with extended **functionality** and increased **automation**
  - Close **interaction with users**
    - For feedback targeting functionality enhancements





Copyright 2016 Cray Inc.

# Cray PE for the CS Series

**The Cray Programming Environment Suite for Clusters is a fully integrated programming environment with compilers, tools, and libraries designed to maximize programmer productivity, application scalability, and performance. It consists of:**

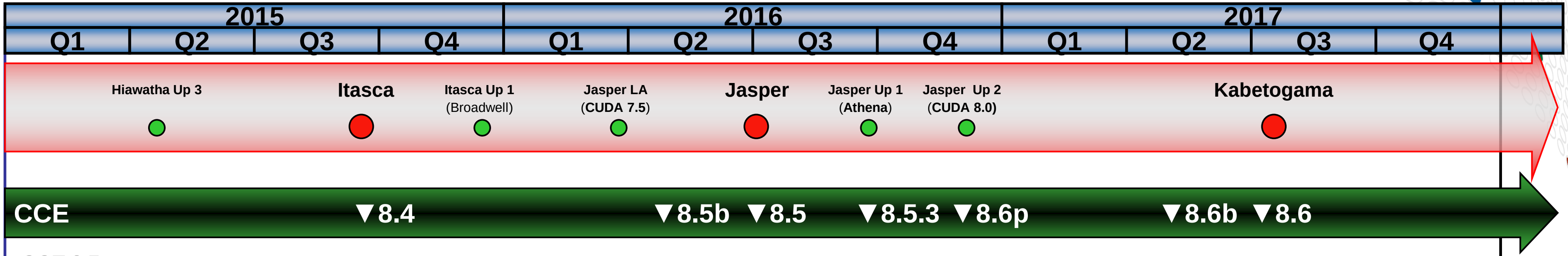
- **Cray Compiling Environment (CCE)**
  - Fortran, C and C++ compilers supporting OpenMP 4.0 (with OpenMP 4.5 Target) and OpenACC 2.0
- **Cray Performance, Measurement, Analysis, and Porting Tools (CPMAT)**
  - PerfTools (CrayPAT, CrayPAT-light, & Cray Apprentice2)
  - Reveal
- **Cray Scientific Libraries (CSML)**
  - Cray LibSci with Autotuned BLAS library, LAPACK, ScaLAPACK, and Iterative Refinement Toolkit (IRT)
  - Cray LibSci_ACC providing accelerated BLAS and LAPACK routines for GPUs
  - Cray optimized FFTW
- **Cray Comparative Debugger (CCDB / lgdb)**
- **Cray Environment Setup and Compiling support – CENV**
- **MPI libraries supported**
  - Intel MPI
  - MVAPICH

# Cray Programming Environment Roadmap



Copyright 2016 Cray Inc.

# CCE 8.5 Highlights and Roadmap




**CCE 8.5**

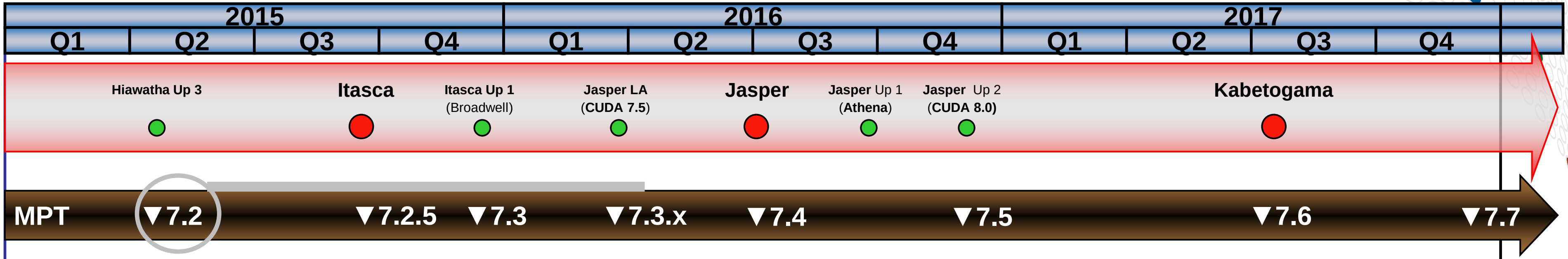
- **Intel KNL support**
- **Memory directives**
- **OpenMP 4.5 "target" (accelerator) support**
- **C11 support**
  - Not default with CCE C
    - *'-hstd=c99'* is the default
    - use *'-hstd=c11'* to enable
- **Compile time improvements (especially for large codes)**
- **Fortran coarray, UPC, and coarray C++ support on CS**
  - Full Fortran 2008 support
- **Athena support (CCE 8.5.3)**

## CCE 8.6 and beyond

- **NVIDIA Pascal GPU with CUDA 8.0 support**
- **C++14 support**
- **OpenMP 4.5 full support**
- **Fortran 2015 support**
  - Fortran coarray teams


Copyright 2016 Cray Inc.

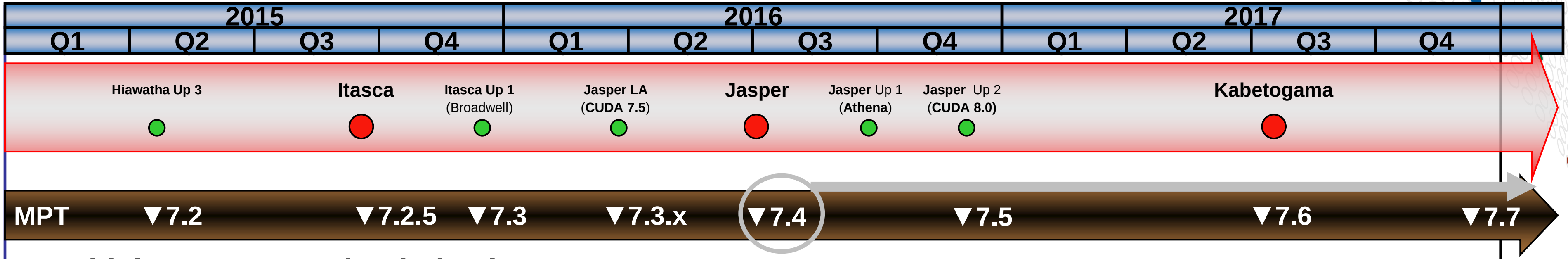
# MPT - Highlights Since Last CUG (MPT 7.2)




- Cray MPI now supports the MPI 3.1 standard (based on MPICH 3.2rc1)
- Added display of high water mark of the memory used by MPI
- GPU-to-GPU support for MPI-3 RMA
- Support to the OpenSHMEM Specification Version 1.3
- Lots of optimizations on MPI and Cray SHMEM
- Reduced MPI memory footprint optimizations


Copyright 2016 Cray Inc.

# MPT 7.4 Highlights

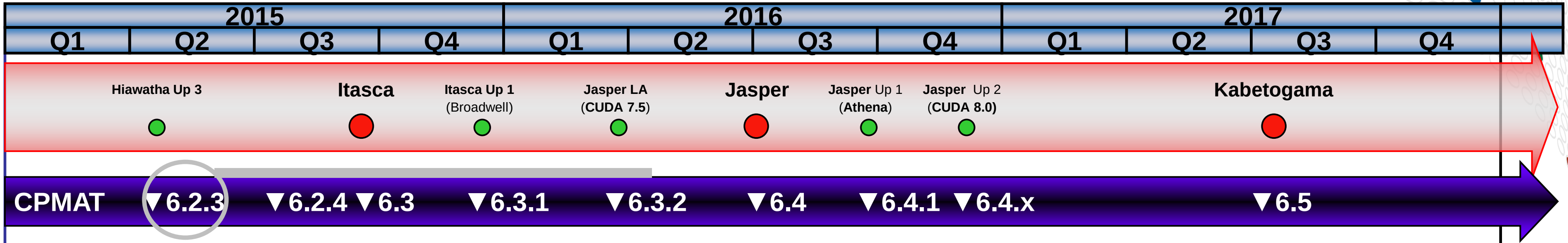



- **Initial KNL support/optimizations**
- **Cray MPI hugepage support for MCDRAM on KNL**
  - (MPI_Alloc_mem and MPI_Win_Allocate)
- **Cray SHMEM support for MCDRAM on KNL**
- **Additional MPI_THREAD_MULTIPLE optimizations (“Thread Hot”)**
- **Initial MPI-IO optimizations for Cray DataWarp**
- **MPI-IO Collective Buffering mode allowing multiple aggregators per OST**
  - Requires Lustre 2.7 client and server (Lock Ahead feature)

***Krishna Kandalla’s MPT talk at 10:30 on Thursday will cover some of these items in-depth**


Copyright 2016 Cray Inc.

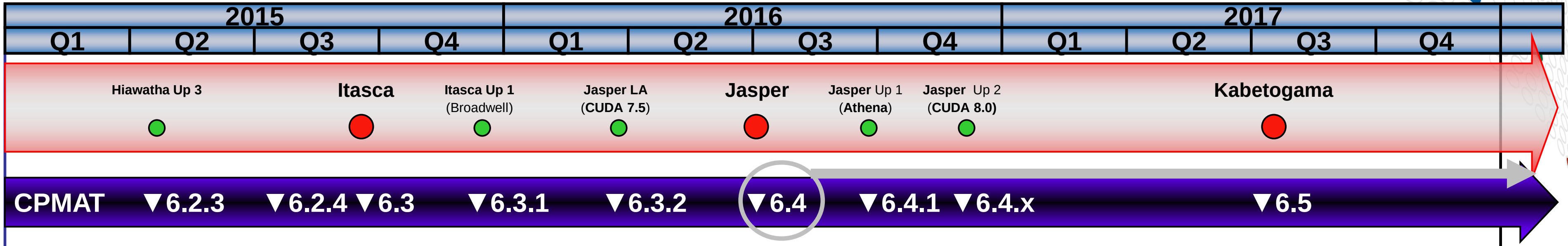
# CPMAT - Highlights Since Last CUG (CPMAT 6.2.3)




- Sampling Over Time (6.2.3)
- Power over time display (6.2.3)
- Support to multiple GPUs per node (Cray CS-Storm) (6.2.4)
- Support for Intel MPI on Cray CS Systems (6.3.0)
- New perftools modules (6.3.0)
- SNB / IVB / HSW uncore counter support with privileged access (6.3.0)
- Apprentice2 new sample data over time plots (6.3.0)
- Apprentice2 mosaic in runtime summarization mode (6.3.0)
- Observation for helper threads in reports (6.3.0)
- Apprentice2 compare (6.3.1)


Copyright 2016 Cray Inc.

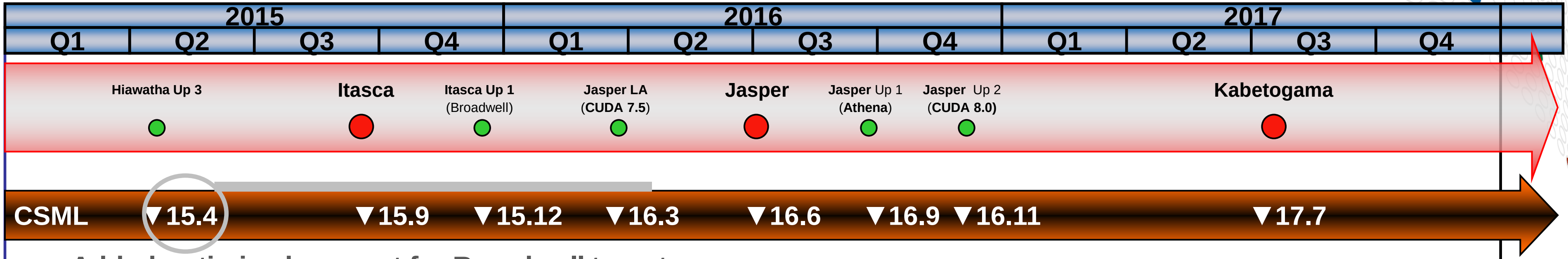
# Roadmap Highlights - CPMAT 6.4 and Beyond




- **Reveal**
  - Include time for loops sorted by compiler messages (e.g., find most time consuming loops that didn't vectorize)
  - auto-parallelization
  - Client for OS X
  - MCDRAM data allocation assistance
- **New trace groups for OpenCL, Lustre API, MemKind, Parallel NetCDF**
- **MPI insight (communication 'strategy' thresholds and advice on environment variables)**
- **Memory usage information per NUMA domain**
- **Support for CHARM++**


Copyright 2016 Cray Inc.

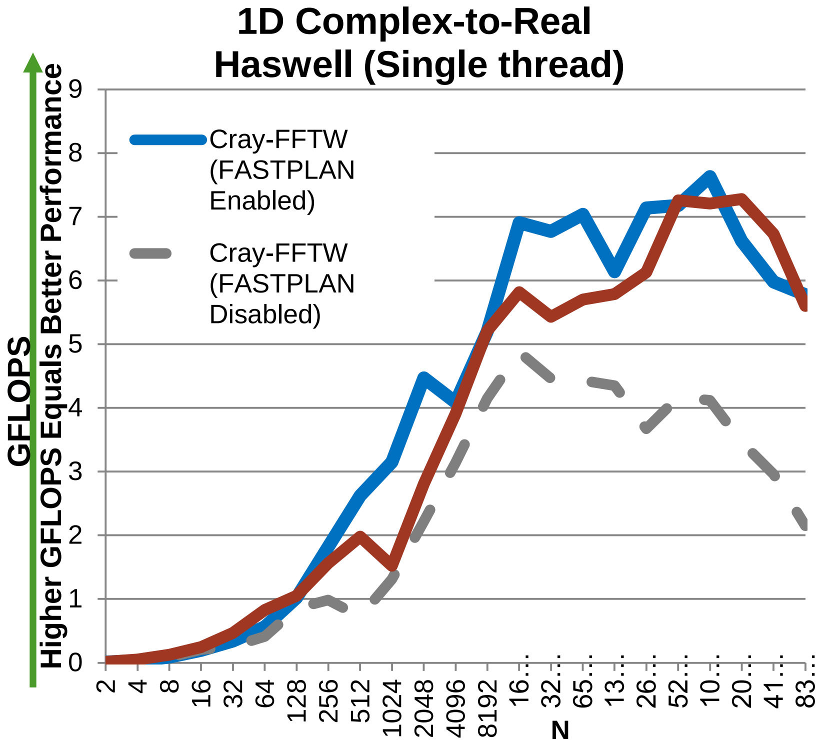
# CSML Highlights Since Last CUG

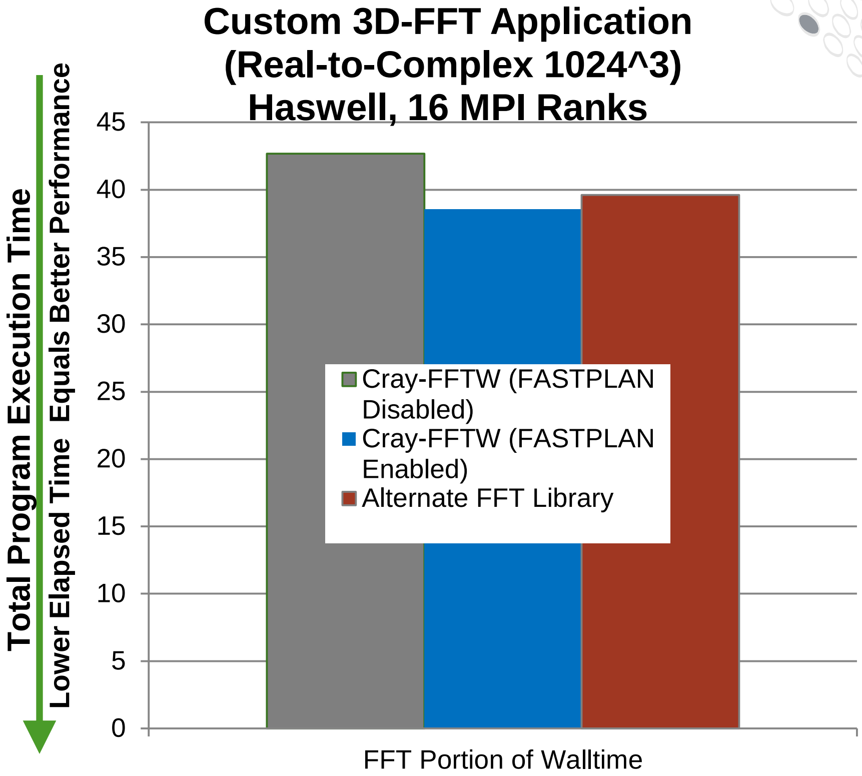



- Added optimized support for Broadwell target
- Improved BLAS GEMM performance for Haswell and Broadwell CPU targets on XC and CS platforms
- Expanded set of optimizations for ScaLAPACK eigensolver routines
- Improved OpenMP threading support for BLAS and LAPACK routines
- Added FASTPLAN optimizations for Haswell and Broadwell targets
- Added FFT support of arbitrary dimension and size for both real and complex data types
- Added 64-bit integer support for Cray PETSc and Cray TPSL
- Added support for HDF5 in Cray PETSc
- Support for latest Kokkos multithreading libraries


Copyright 2016 Cray Inc.

# Cray FFTW FASTPLAN Performance

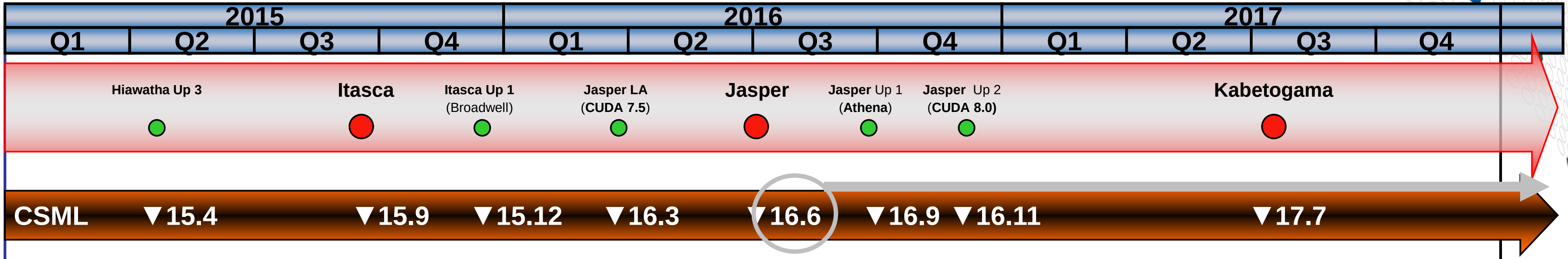



**FASTPLAN enable with environment variable FFTW_CRAY_FASTPLAN=1**


Copyright 2016 Cray Inc.

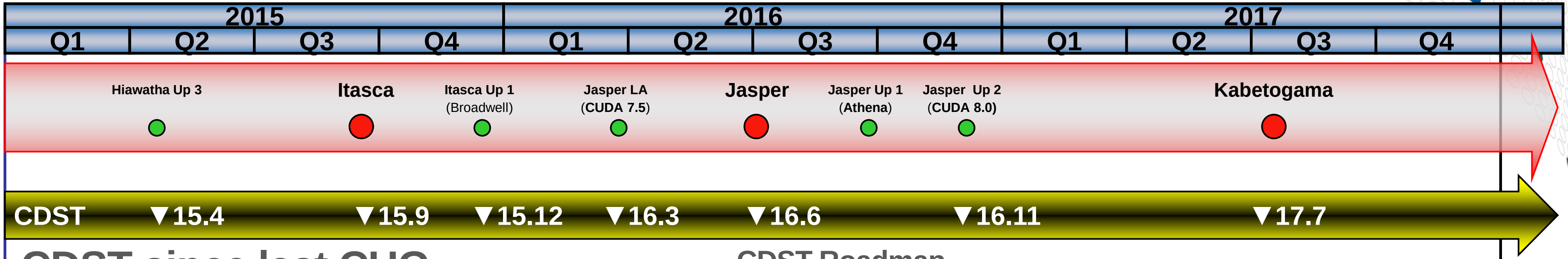
# CSML Roadmap




- Support KNL targets on XC and CS systems
- Small GEMM optimizations for BLAS
- Athena support
- Support for Nvidia Pascal GPU targets on XC and CS systems running CUDA 8.0


Copyright 2016 Cray Inc.

# CDST Highlights and Roadmap




## CDST since last CUG

- STAT 2.1 with ALPS DSL tool helper
- ATP, STAT & LGDB SLURM 15 support
- CCDB and LGDB CUDA 7.5 support
- CCDB and LGDB on Cray CS Systems

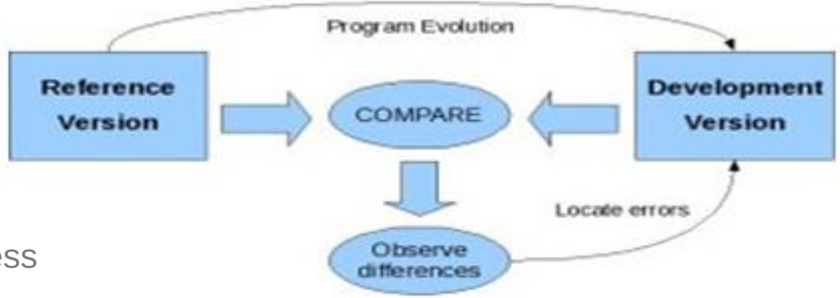
## CDST Roadmap

- KNL support
- ATP, STAT & LGDB single release to support multiple SLURM versions
- LGDB Pascal GPU and CUDA 8.0 support
- LGDB additional improvements for C++
- LGDB better UPC support
- CCDB better comparisons for large data structures


Copyright 2016 Cray Inc.

# CCDB Overview

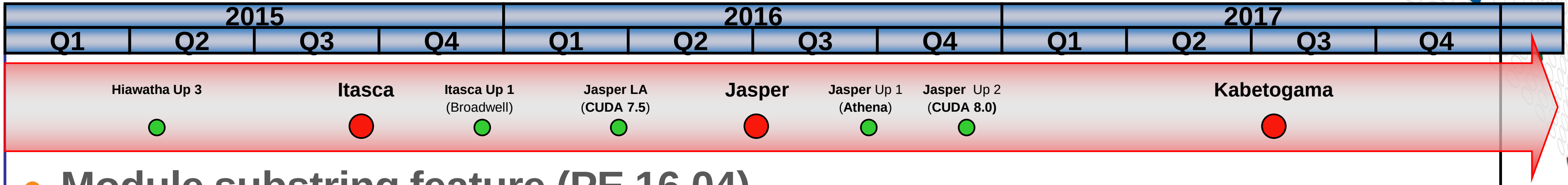
- **What is comparative debugging?**
  - Data centric approach instead of the traditional control-centric paradigm
  - Two applications, same data
  - Key idea: The data should match
  - Quickly isolate deviating variables
- **Comparative debugging tool**
  - NOT a traditional debugger!
  - Assists with comparative debugging
  - CCDB GUI hides the complexity and helps automate process
    - Creates automatic comparisons
    - Based on symbol name and type
    - Allows user to create own comparisons
    - Error and warning epsilon tolerance
    - Scalable
- **How does this help me?**
  - Algorithm re-writes
  - Language ports
  - Different libraries/compilers
  - New architectures
- **Collaboration with University of Queensland**





The University Of Queensland
AUSTRALIA

# Craype/modules Highlights




- **Module substring feature (PE 16.04)**
  - Returns results if the argument is a part of any module name, rather than just modules that start with substring argument
    - Example: "**module –S avail hdf5**" would find all modules with "hdf5" anywhere in the name
- **New CDT module for each XC PE release (PE 16.04)**
  - Switches currently loaded modules to the version associated with a specific CDT release
  - Subsequently loaded modules also load the version associated with CDT release
  - Started with the April 2016 PE release (cdt.16.04)

*Note unloading the cdt module is not sufficient to restore your loaded modules to the system defaults. A script, restore_system_defaults.sh, can be sourced to restore your currently loaded modules to the system defaults


Copyright 2016 Cray Inc.

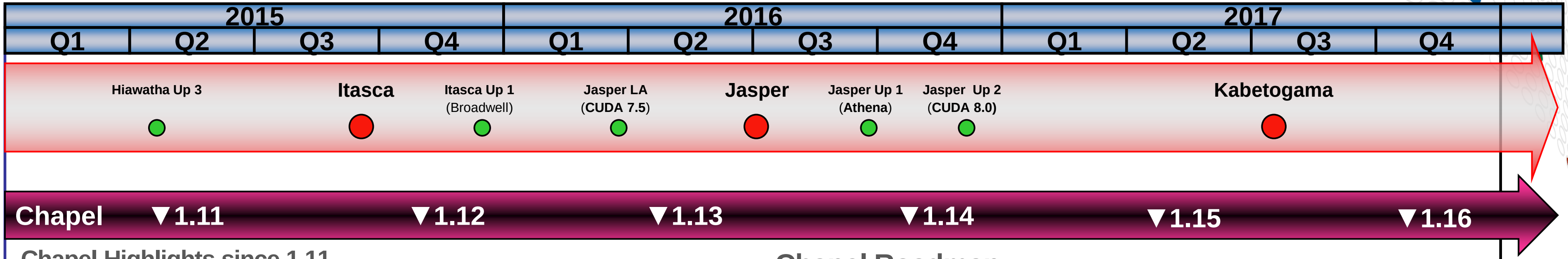
# New cdt module for each PE release on XC

```
ldr@cdt-test:~> module list
Currently Loaded Modulefiles:
  1) modules/3.2.10.3
  2) cce/8.4.3
  3) craype-network-aries
  4) craype/2.5.1
  5) cray-libsci/13.3.0
  6) cray-mpich/7.3.1
…
ldr@cdt-test:~> module avail cce
------------------------------ /opt/modulefiles -------------------------
cce/8.4.3(default)            cce/8.4.5

ldr@cdt-test:~> module avail cray-mpich
---------------------------- /opt/cray/modulefiles -------------------
cray-mpich/7.3.1(default)  cray-mpich-abi/7.3.1(default)
cray-mpich/7.3.3            cray-mpich-abi/7.3.3
```

```
ldr@cdt-test:~> module load cdt/16.04
…
Switching to cce/8.4.5.
Switching to cray-libsci/16.03.1.
Switching to craype/2.5.4.
Switching to modules/3.2.10.4.
Switching to cray-mpich/7.3.3.
…
ldr@cdt-test:~> module avail cce
----------------------------- /opt/modulefiles -------------------------
cce/8.4.3         cce/8.4.5(default)

ldr@cdt-test:~> module avail cray-mpich
---------------------------- /opt/cray/modulefiles ---------------------
cray-mpich/7.3.1             cray-mpich-abi/7.3.1
cray-mpich/7.3.3(default)   cray-mpich-abi/7.3.3(default)
```


Copyright 2016 Cray Inc.

# Chapel Highlights and Roadmap




### Chapel Highlights since 1.11

- Improved string and record implementation
  - dramatically improved string performance
  - greatly reduced memory leaks
  - standard string library routines supported
- New module namespace control features
- Improved compiler stability
- Improved library support
- Parallelize large numeric array initialization
  - improved first touch with NUMA domains
- Performance improvement highlights
  - reduction operations
  - convert RMA to local access as permitted
  - significant performance improvements for 'ugni' comm on Crays
  - bulk communication optimization for improved scaling

### Chapel Roadmap

- Continuously improving performance and scalability
- UTF8 string and library support
- Eliminate memory leaks
- Improved Error handling
- Task Teams
- Improved Object Oriented Story
- Additional general purpose libraries
- Additional platform support
  - KNL, ARM, GPUS, etc.



Copyright 2016 Cray Inc.

# CCE Directives Update (OpenMP / OpenACC / Memory)

| CCE 8.4 | CCE 8.5 (June 2016) | CCE 8.6 (tentative) |
|---|---|---|
| OpenMP 4.0 | OpenMP 4.5 (target) | OpenMP 4.5 (complete) |
| OpenACC 2.0 | ~~OpenACC 2.5~~ (no planned support) | |
| | KNL “memory” directive | |

- **Cray will continue to support OpenACC 2.0**
  - **No plans to support OpenACC 2.5**
- **Recommendation: migrate to OpenMP “target”**
  - OpenMP 4.0 provides constructs for base functionality
  - **OpenMP 4.5 provides constructs for competitive performance**
    - Asynchronous data transfers and kernel execution
    - Default scoping behavior of scalars as firstprivate
    - Unstructured data regions, “host data” regions, device pointers
    - Device memory API


Copyright 2016 Cray Inc.

# KNL Memory from CCE Perspective

- **Node has two different types of memory**
  - DDR: high-capacity, low-bandwidth
  - MCDRAM: high-bandwidth, low-capacity
- **Typical memory footprints exceed MCDRAM**
  - Best performance requires MCDRAM
- **Users must allocate specific variables in MCDRAM**
  - Not everything will fit at once
  - Latency-sensitive variables should probably go in DDR

- **CCE solution: provide mechanisms for developers to place specific variables and allocations in MCDRAM**


Copyright 2016 Cray Inc.

# CCE Support for MCDRAM

- **Cray Directive (pragma) to support data allocation in MCDRAM**
  - Provide a directive-only solution
    - Cover more use cases
  - **Support for Fortran, C, and C++**
    - The directive can be used on **both local and global variables**
      - to place the variables in high bandwidth memory
    - The directive **can also be used on a statement**
      - to change any allocation routines on that statement (allocate, malloc, etc.) to use HBM
    - **If Clause** for dynamic control of directive
    - **Fallback Clause** to control behavior if allocation fails
  - Future direction for memory hierarchy control
    - Ideally will become part of a standard, possibly OpenMP


Copyright 2016 Cray Inc.

# CCE Proposed API for KNL HBM

- **Directive (pragma) to control placement for high bandwidth memory**
  - Support for Fortran, C and C++
  - Proposed directive
    - **!dir$ memory(*attributes*) *[list of variables]***
    - **#pragma memory(*attributes*) *[list of variables or allocatable members]***
      - *Attributes* – list of desired memory attributes (bandwidth, capacity, nonvolatile, etc.)
        - Initially “bandwidth” is the only allowed attribute
          - Other attributes may be added in the future

- **Statements**
  - Appears prior to an allocation/deallocation statement
  - Changes explicit allocation routines in the next statement to use HBM
    - **Fortran: allocate**
    - **C/C++: malloc, calloc, realloc, posix_memalign, free**
    - **C++: new, delete, new[], delete[]**
      - Directive on deallocation must match (C/C++ only)


Copyright 2016 Cray Inc.

# CCE Directive for Variable Declarations

```
!dir$ memory(attributes) list-of-vars
#pragma memory(attributes) list-of-vars
```

- **Specified at declaration of variable**
  - For global variables, directive must be visible for every use of global
  - Within type for allocatable members
- **Allowed on:**
  - Local and global variables
  - Scalars, structs and arrays (fixed size and variable length)
  - Fortran allocatables (including members of derived types)
    - Memory allocated will use high bandwidth memory
- **Not allowed on:**
  - Dummy arguments
  - Common blocks or variables within a common block
  - Fortran pointers
  - Variables involved in equivalences
  - Coarray or UPC shared variables


Copyright 2016 Cray Inc.

# Cray Memory Directive – Current Status

- **Initial implementation and basic testing of the Cray memory directive is complete for CCE 8.5**
  - Target June 2016 release
    - Support for Intel's FASTMEM attribute is deferred to a future CCE release
- **Internal users are starting to use the feature and providing feedback**
- **Cray is working with OpenMP to incorporate this feature into the OpenMP 5.0 specification (2017/2018)**
  - Cray presented to the OpenMP accelerator subcommittee in April
  - Intent is to initially include the feature in the annual OpenMP TR by SC'2016


Copyright 2016 Cray Inc.

# Summary

- **Application developers need a programming environment that can address and hide the issues of scale and complexity of supercomputers**

- **Cray's advanced programming environment continue to focus on Performance and Programmability**
  - Cray Compiling Environment (CCE) focused on application performance
    - Fully automatic loop vectorization
    - Directives for accelerators and multiple levels of the memory hierarchy
  - Cray Performance Analysis Tools
    - Focus on automation, scaling, and ease of use
  - Reveal
    - Scoping analysis and parallelization assistant
  - Parallel debugger support
  - Auto-tuned Scientific Libraries support
    - Getting performance from the system … **no assembly required**



Copyright 2016 Cray Inc.

CRAY

COMPUTE | STORE | ANALYZE

# Thank You