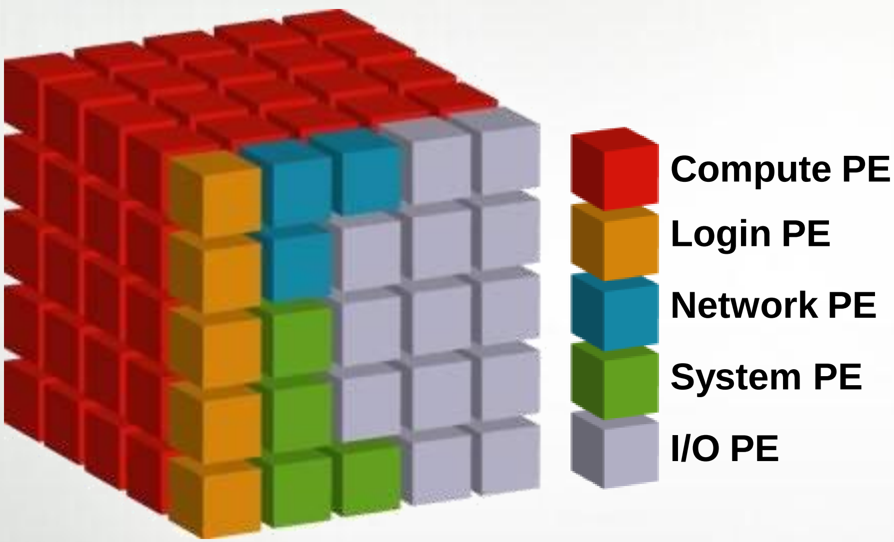
Cray XT/XE Architecture

Cray XT5

Scalable Software Architecture: Cray Linux Environment (CLE)

“Primum non nocere”

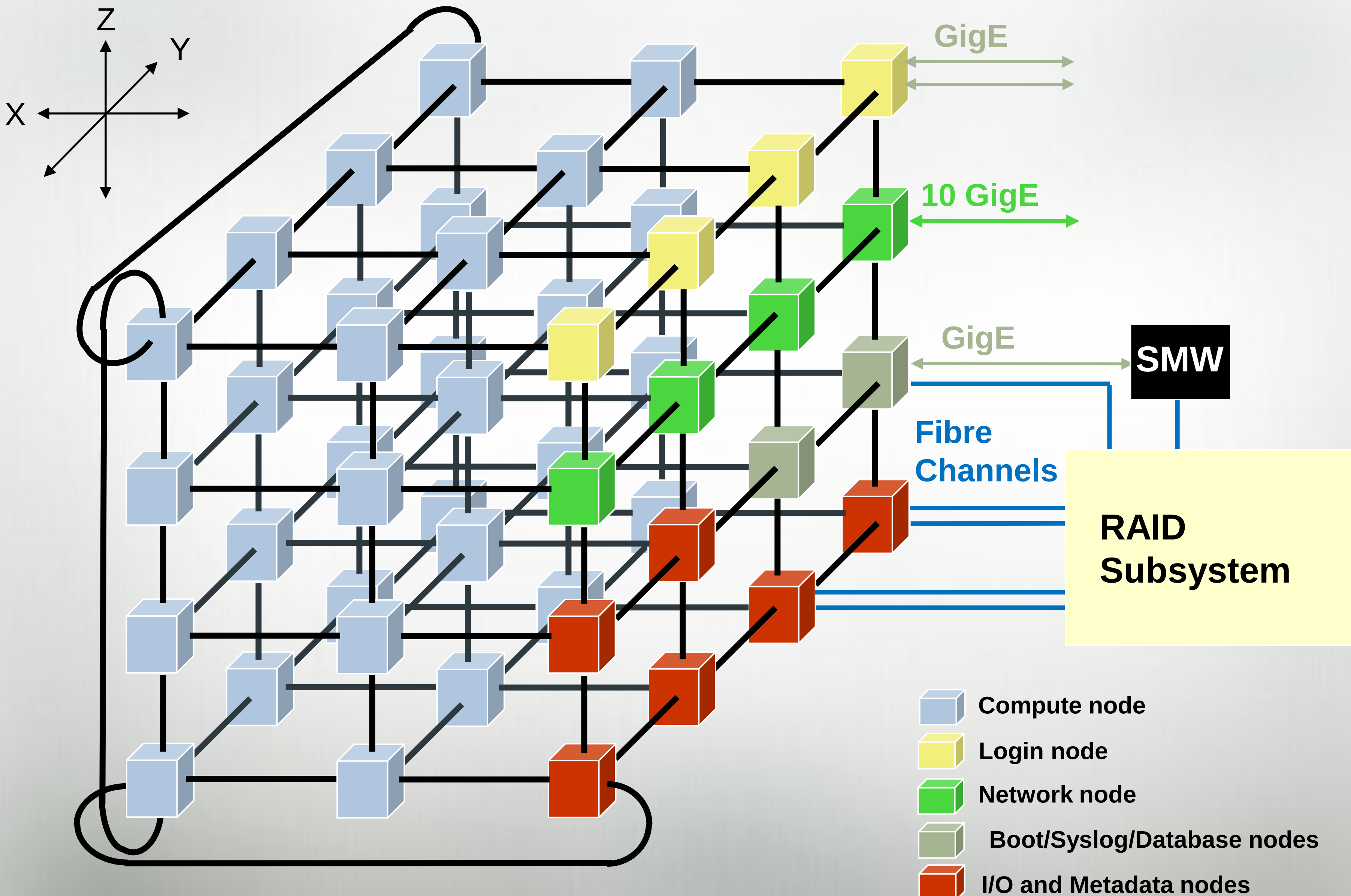


Service Partition

Specialized Linux nodes

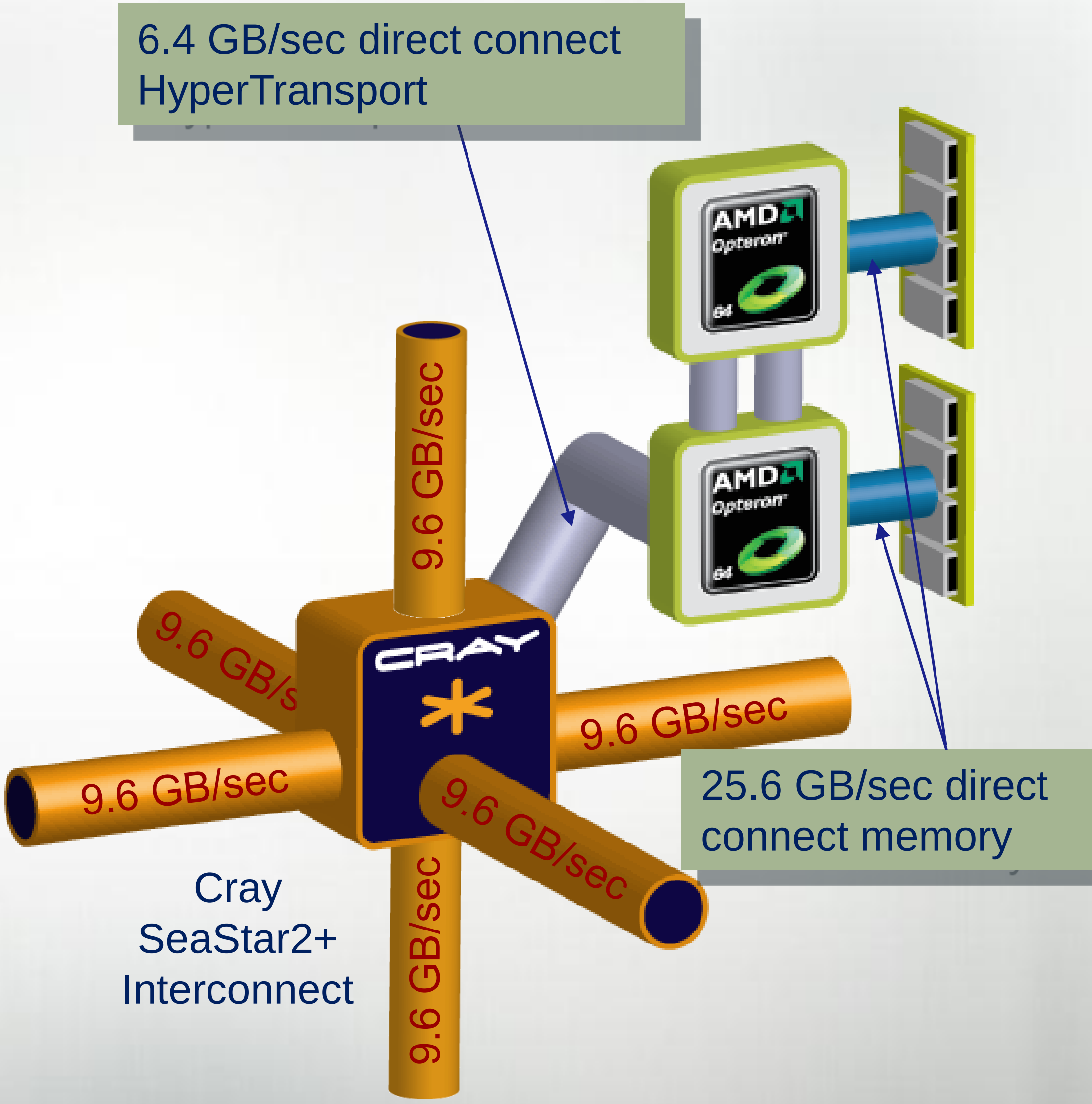
- Microkernel on Compute PEs, full featured Linux on Service PEs.
- Service PEs specialize by function
- Software Architecture eliminates OS “Jitter”
- Software Architecture enables reproducible run times
- Large machines boot in under 30 minutes, including filesystem

XT System Configuration Example



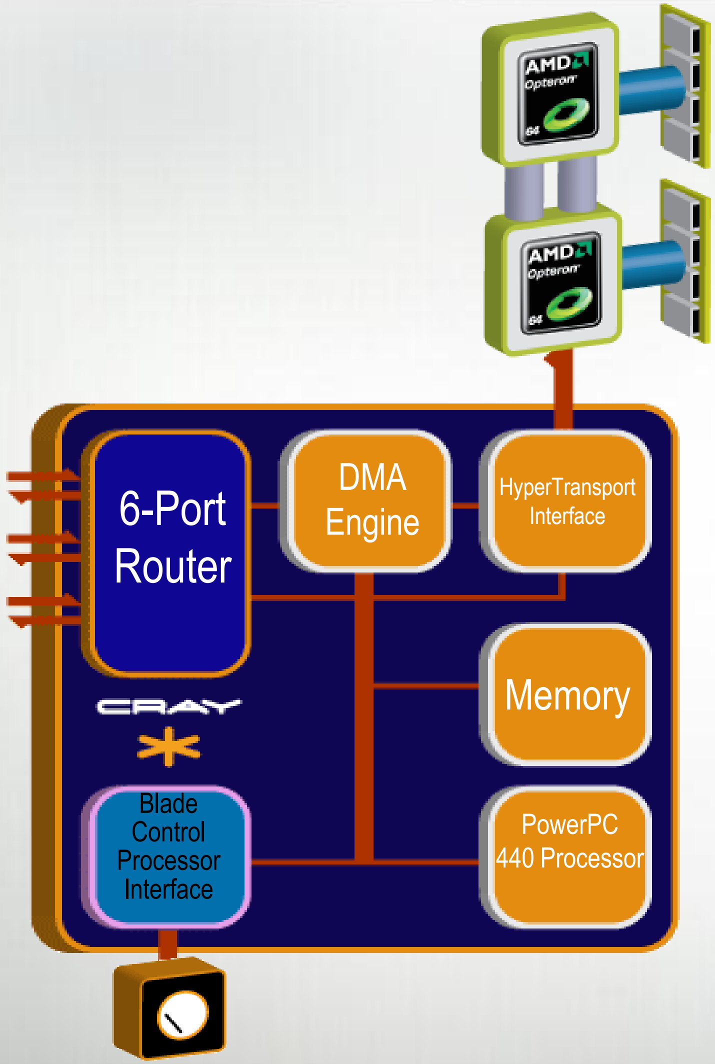
Cray XT5 Node

Characteristics	
Number of Cores	12
Peak Performance Istanbul (2.6)	124 Gflops/sec
Memory Size	16 GB per node
Memory Bandwidth	25.6 GB/sec



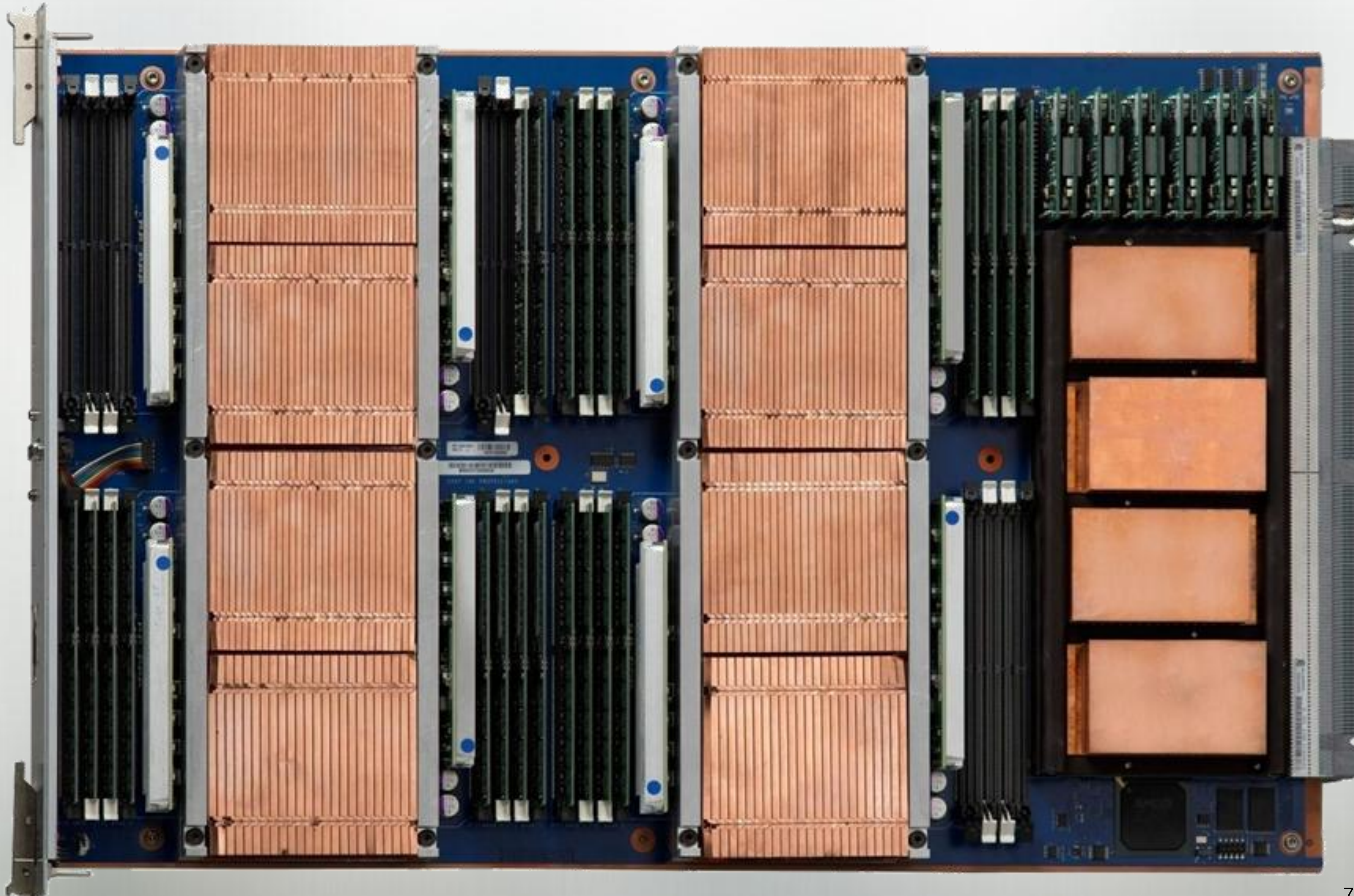
Cray SeaStar2+ Interconnect

Now Scaled to 225,000 cores

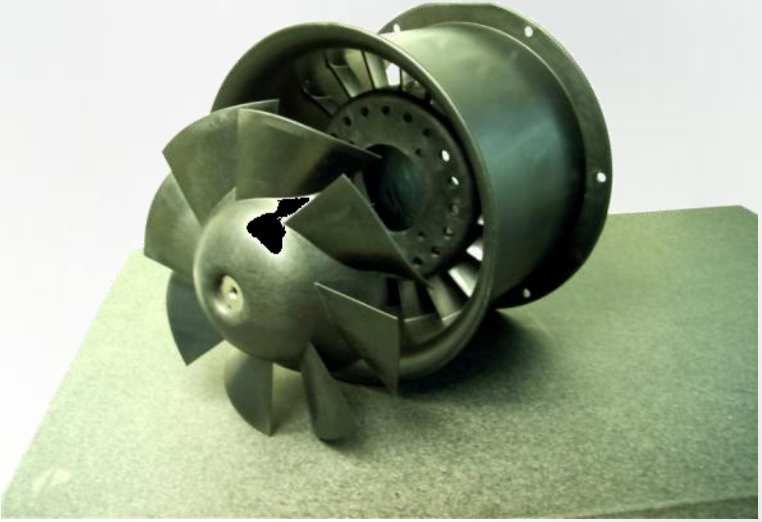


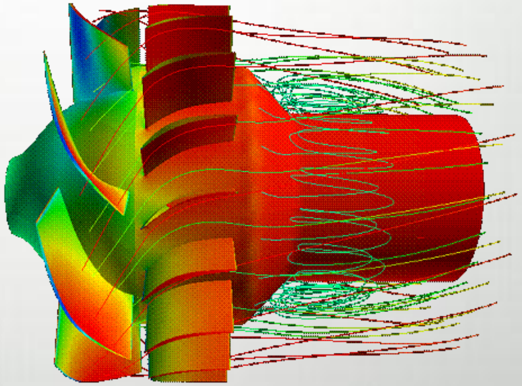
- Cray XT5 systems ship with the SeaStar2+ interconnect
- Custom ASIC
- Integrated NIC / Router
- MPI offload engine
- Connectionless Protocol
- Link Level Reliability
- Proven scalability to 225,000 cores

Cray XT5 Compute Blade

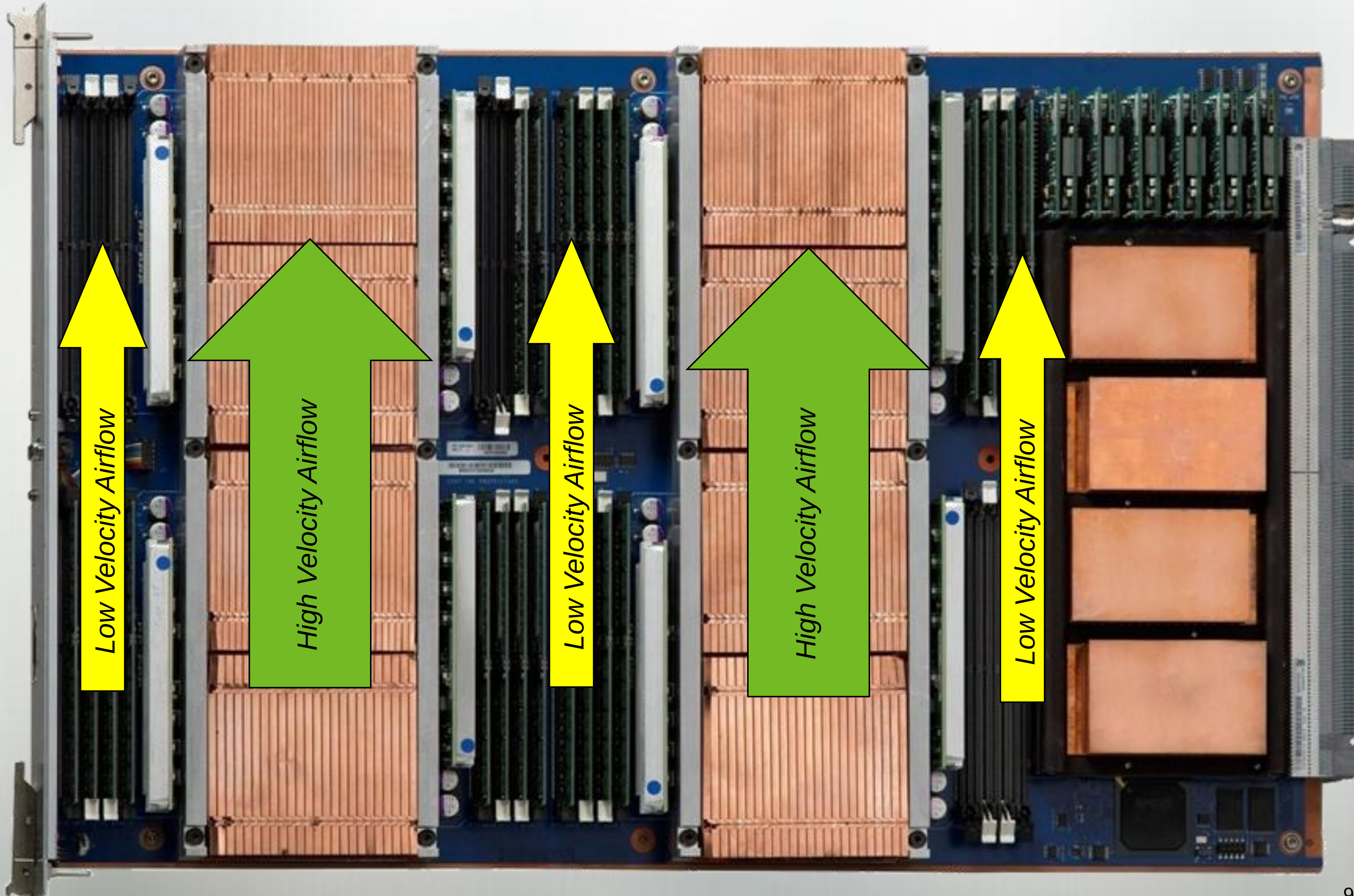


XT5 Axial Turbofan – 78% Efficient

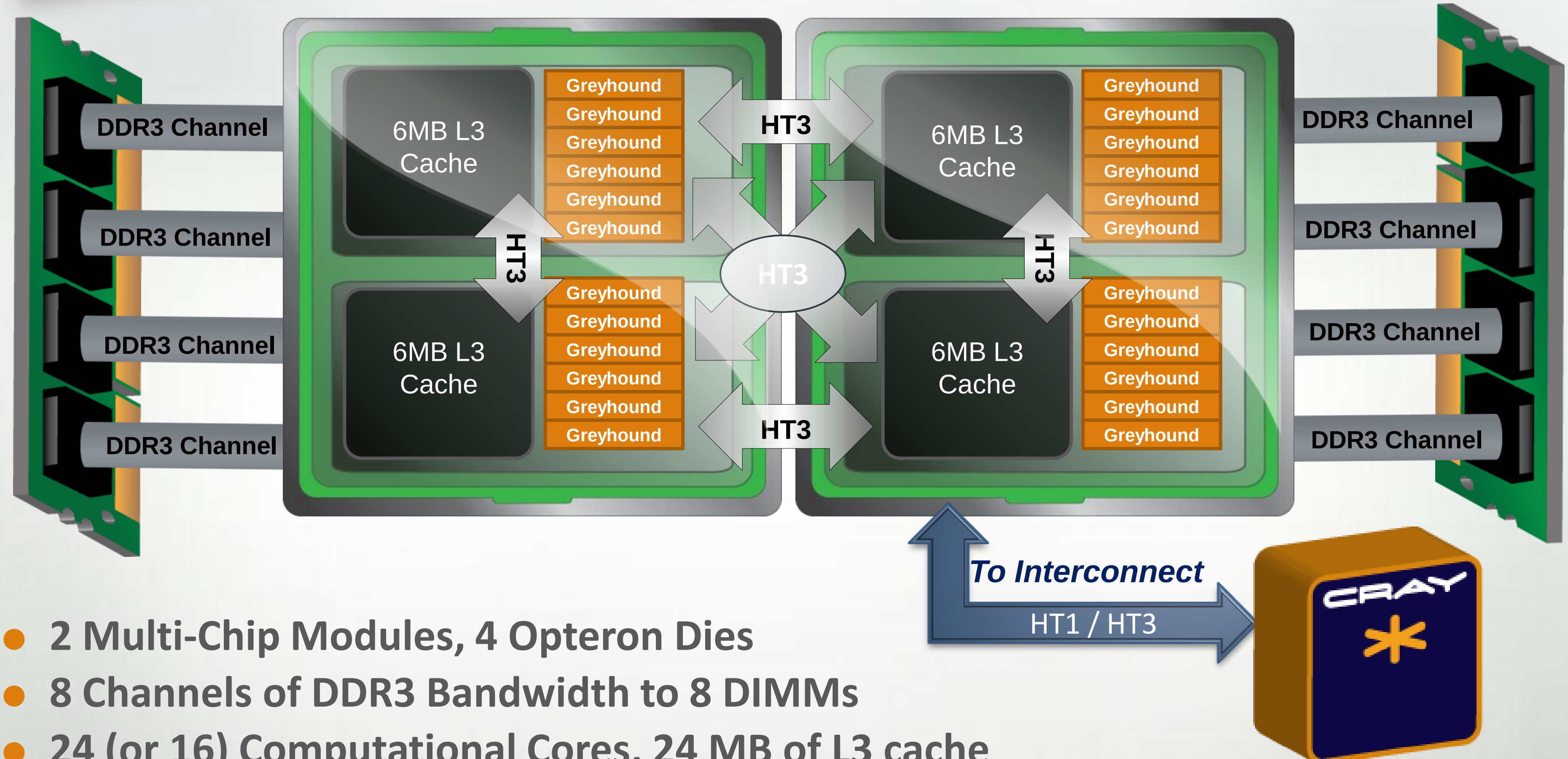




Cray XT5 Compute Blade



XT6 Node Details: 24-core Magny Cours



- 2 Multi-Chip Modules, 4 Opteron Dies
- 8 Channels of DDR3 Bandwidth to 8 DIMMs
- 24 (or 16) Computational Cores, 24 MB of L3 cache
- Dies are fully connected with HT3
- Snoop Filter Feature Allows 4 Die SMP to scale well

Two Magny Cours XE6 Nodes
(Gemini Interconnect)

10 12X Gemini Channels

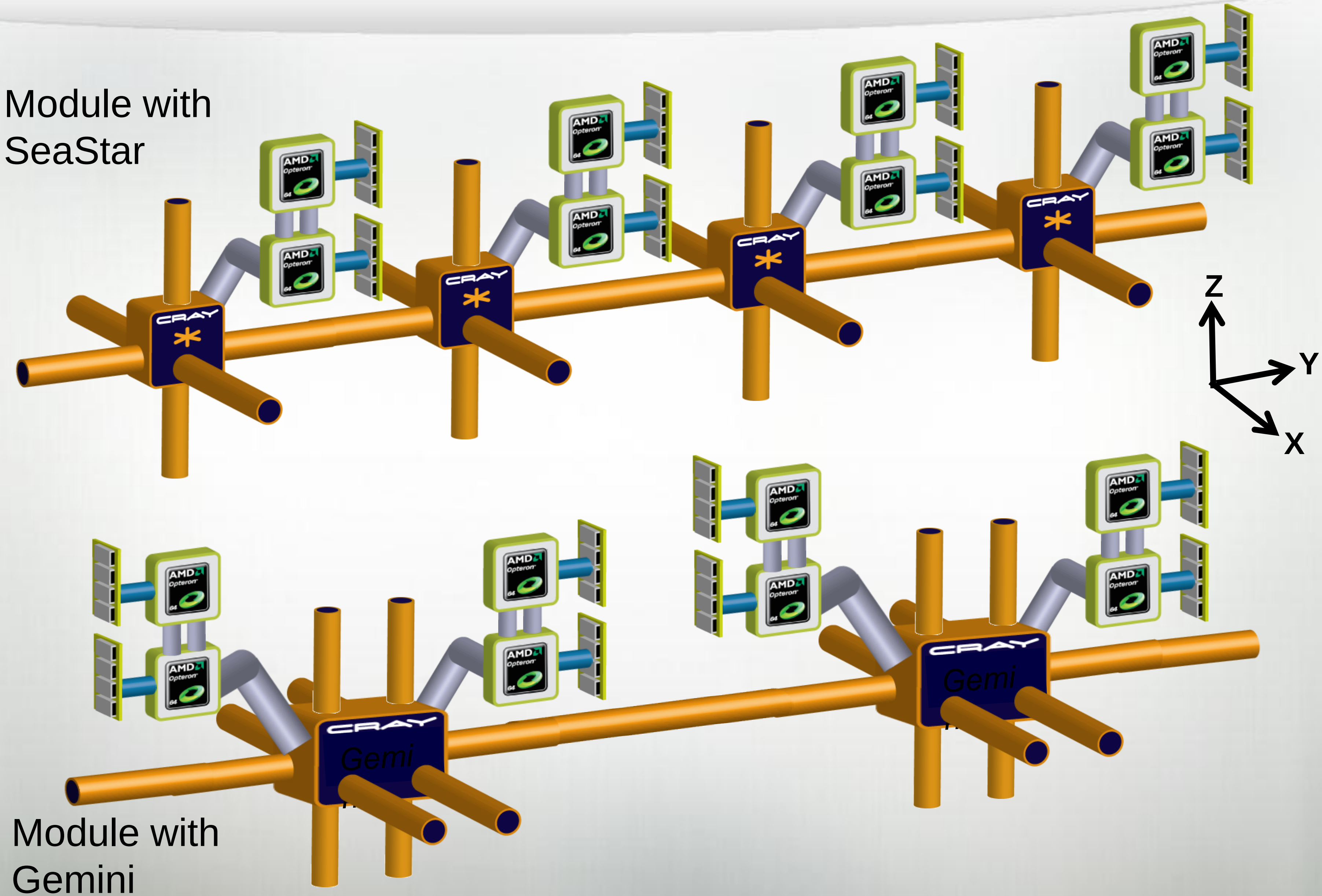
(Each Gemini acts like two nodes on the 3D Torus)

High Radix YARC Router with adaptive Routing

168 GB/sec capacity

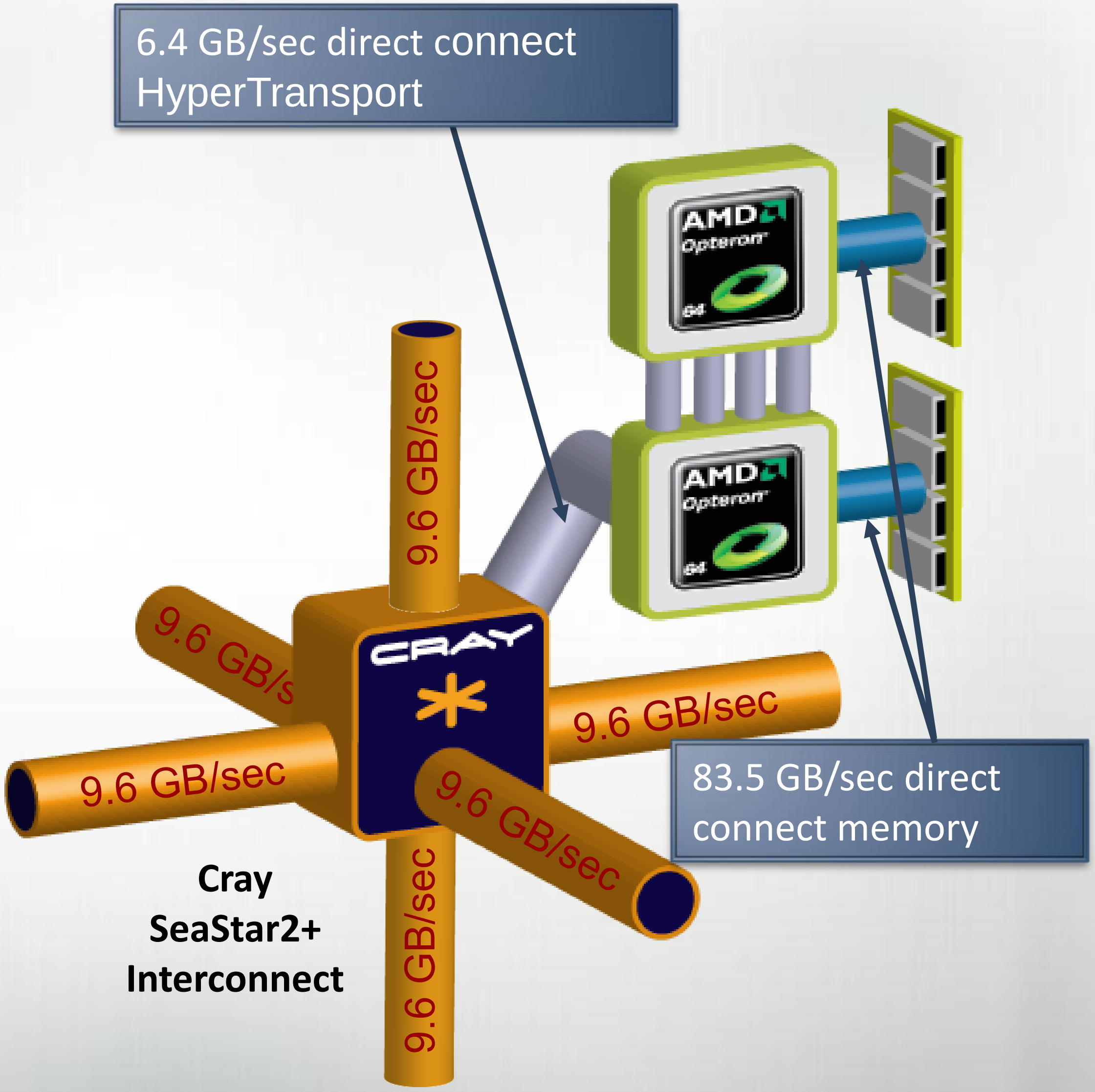
Cray Baker Node Characteristics	
Number of Cores	16 or 24
Peak Performance	140 or 210 Gflops/s
Memory Size	32 or 64 GB per node
Memory Bandwidth	85 GB/sec

Gemini vs SeaStar – Topology



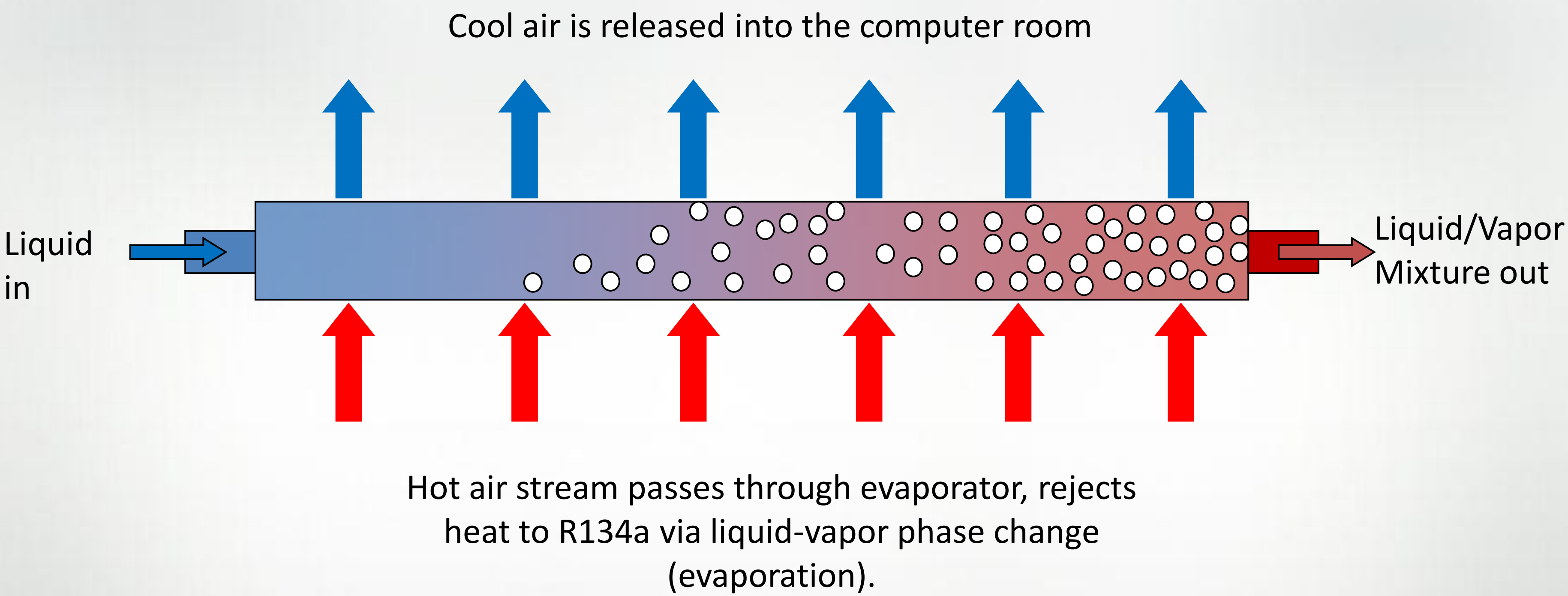
Cray XT6 (Or XT6m) Node

Characteristics	
Number of Cores	16 or 24 (MC) 32 (IL)
Peak Performance MC-8 (2.4)	153 Gflops/sec
Peak Performance MC-12 (2.2)	211 Gflops/sec
Memory Size	32 or 64 GB per node
Memory Bandwidth	83.5 GB/sec



Cray ECOphlex Liquid Cooling

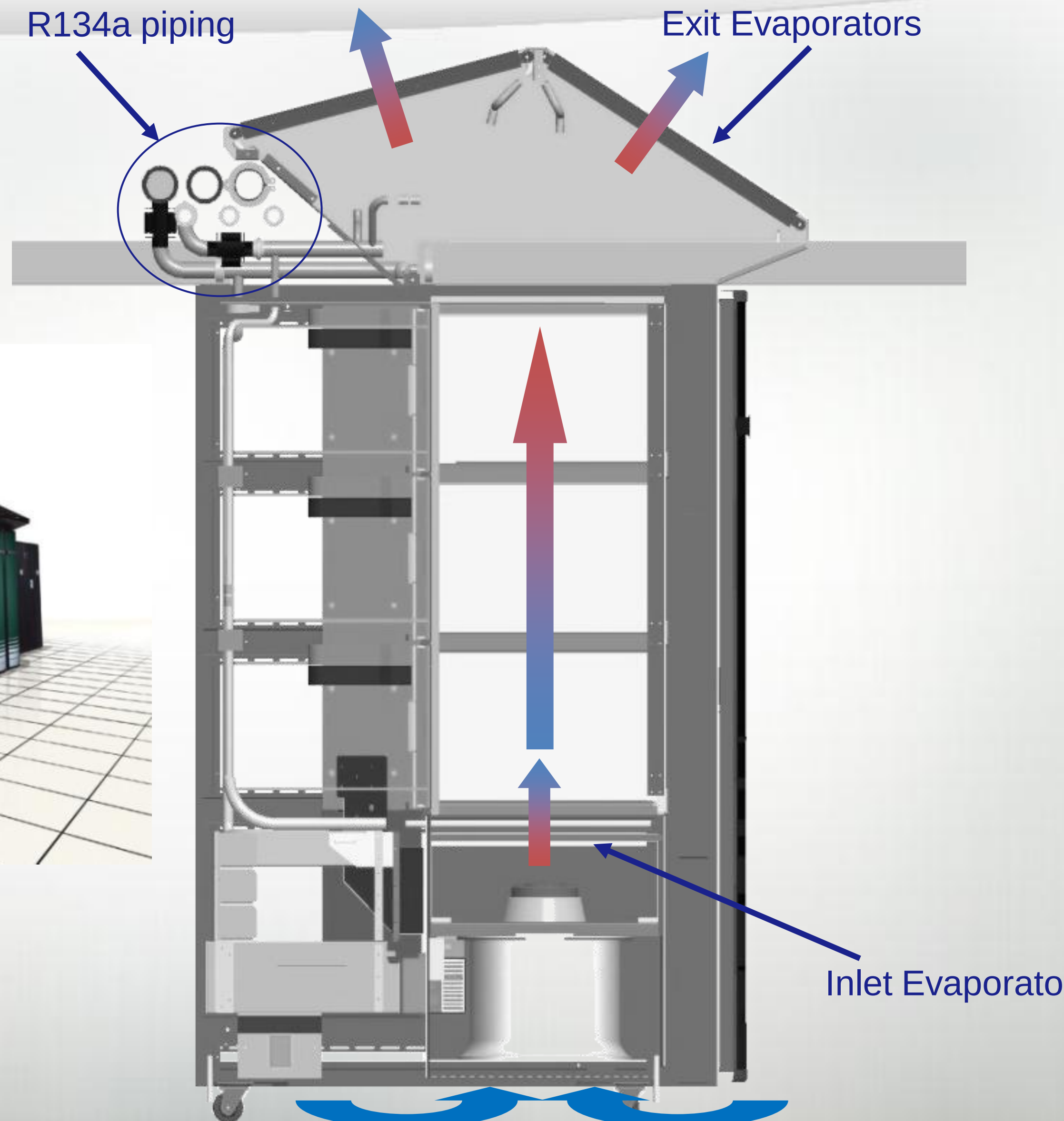
ECOphlex Cooling



R134a absorbs energy only in the presence of heated air.

Phase change is 10x more efficient than pure water cooling.

ECOphlex Technology
in the Cray High Efficiency Cabinet



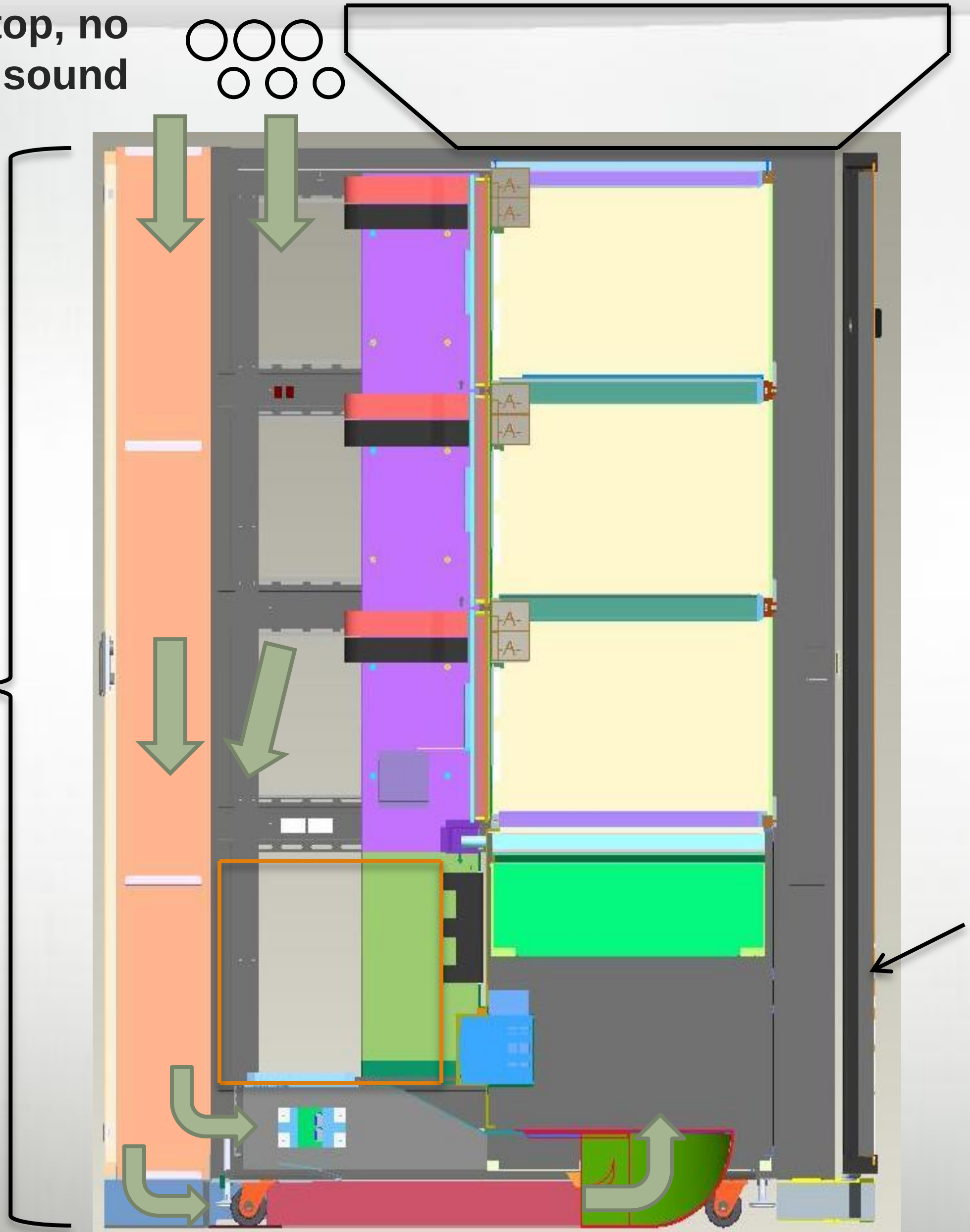
Newer "Flat Top" ECOphlex Design

Other Changes

- New enhanced blower to handle the 130 Watt Magny-Cours Processor
- Enhanced sound kit to reduce noise
- More efficient design
- New VFD (Variable Frequency Diode) for blower
- An upgrade kit product code will be available for existing XT5 customers which will contain the required components

Enhanced Series 6 ECOphlex Cabinet

Air taken from top, no line of sight for sound

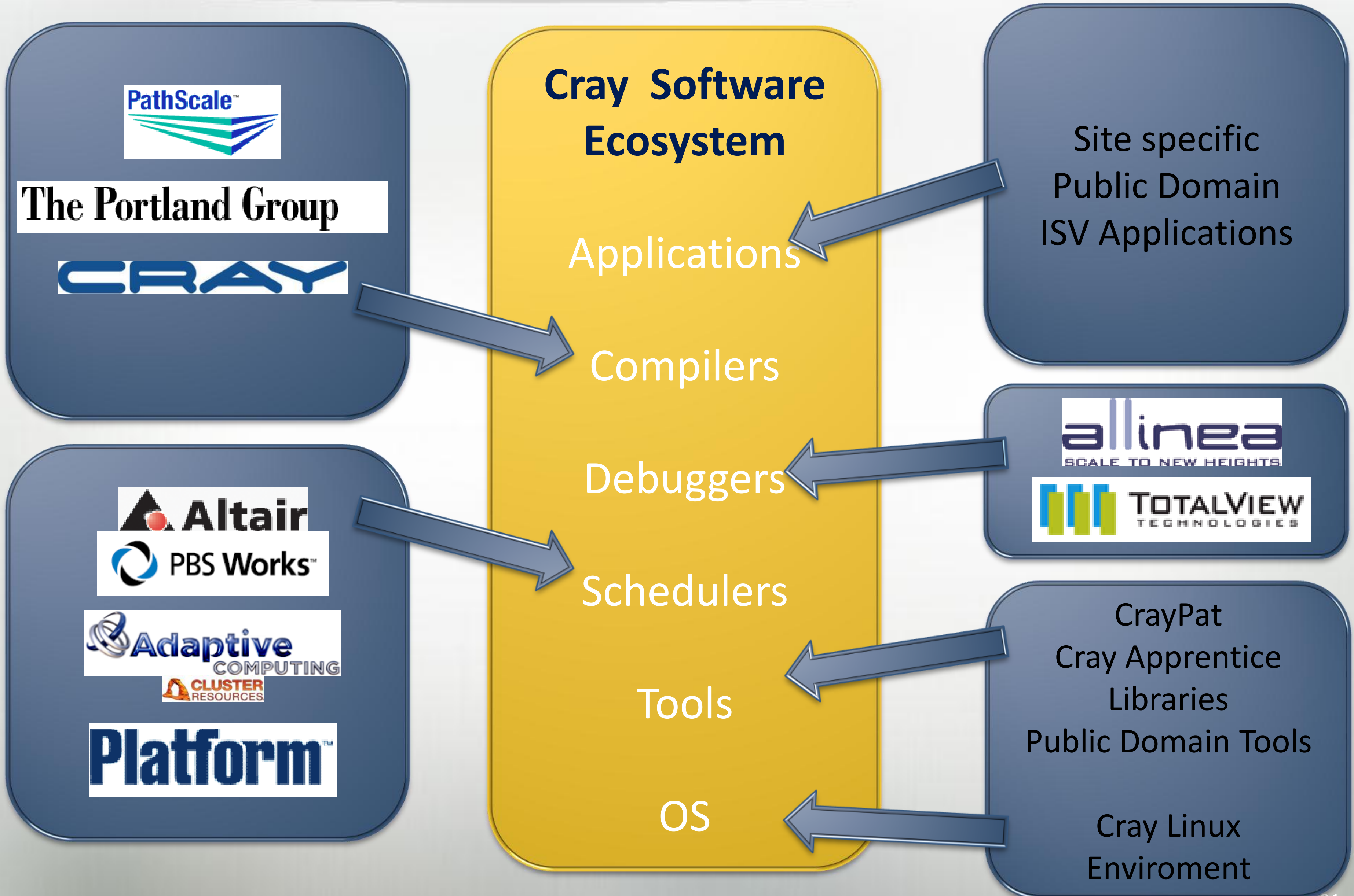


Foam lined duct for sound absorption

Extra foam added to front. Door now seals to front IO extension

Software

Cray Software Ecosystem

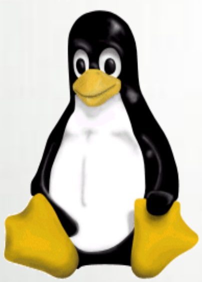


Cray Linux Environment (CLE)

- **Service nodes run a full-featured SLES10 Linux installation**
  - We add our tools, libraries, and services
- **Compute nodes run a slim-line Linux kernel with only necessary services**
  - Only run what's needed so the application can rule the roost
- **Libraries**
  - MPT – Message Passing Toolkit
  - LibSci – Cray Scientific Libraries (BLAS, LAPACK, SCALAPACK, FFTW, etc)
  - I/O Libraries – HDF5 & NetCDF
- **Tools**
  - Compilers – PGI, Cray, GNU, Pathscale, Intel
  - CrayPAT – Performance Analysis Tools
- **ALPS**
  - Application placement, job launching, application clean-up
  - Users interface with ALPS primarily via aprun
- **PBS/TORQUE & MOAB**
  - All jobs on the local XTs are batch jobs
  - MOAB is an advanced job scheduler that is used on Jaguar and Kraken

Cray Linux Environment – Adaptive Vision

CRAY THE SUPERCOMPUTER COMPANY



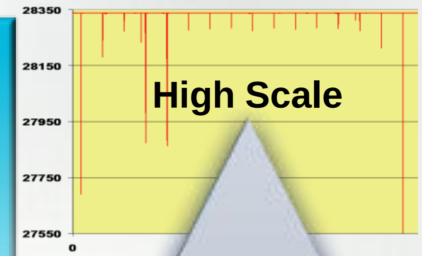
Less Compatibility

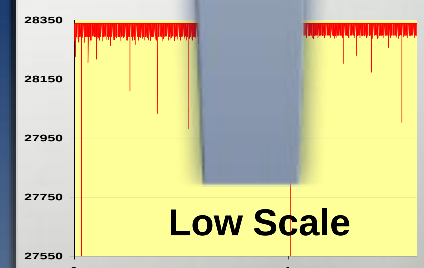
Full Compatibility

Capability (Ultra-light Linux Image)

Capacity/Production
(Mid-weight Linux Image)

Shrink-wrap
3rd Party Application
(Full Linux Image and all services)





A Very Skinny Penguin – Core Specialization

- **Benefit: Eliminate noise with overhead (interrupts, daemon execution) directed to a single core**

- **Rearranges existing work**
  - Without core specialization: overhead affects every core
  - With core specialization: overhead is confined, giving app exclusive access to remaining cores

- **Helps some applications, hurts others**
  - POP 2.0.1 on 8K cores on XT5: 23% improvement
  - Larger jobs should see larger benefit
  - Future nodes with larger core counts will see even more benefit

- ***This feature is adaptable and available on a job-by-job basis***

Programming Environment

Compiler Wrappers

- **Cray XT/XE Supercomputers come with compiler wrappers to simplify building parallel applications (similar the mpicc/mpif90)**
  - Fortran Compiler: ftn
  - C Compiler: cc
  - C++ Compiler: CC
- **Using these wrappers ensures that your code is built for the compute nodes and linked against important libraries**
  - Cray MPT (MPI, Shmem, etc.)
  - Cray LibSci (BLAS, LAPACK, etc.)
  - …
- **Choosing the underlying compiler is via the PrgEnv-* modules, do not call the PGI, Cray, etc. compilers directly.**
- **Always load the appropriate xtpe-<arch> module for your machine**
  - Enables proper compiler target
  - Links optimized math libraries

Compiler Choices – Relative Strengths

…from Cray's Perspective

- **PGI – Very good Fortran and C, pretty good C++**
  - Good vectorization
  - Good functional correctness with optimization enabled
  - Good manual and automatic prefetch capabilities
  - Very interested in the Linux HPC market, although that is not their only focus
  - Excellent working relationship with Cray, good bug responsiveness
- **Pathscale – Good Fortran, C, possibly good C++**
  - Outstanding scalar optimization for loops that do not vectorize
  - Fortran front end uses an older version of the CCE Fortran front end
  - OpenMP uses a non-pthreads approach
  - Scalar benefits will not get as much mileage with longer vectors
- **Intel – Good Fortran, excellent C and C++ (if you ignore vectorization)**
  - Automatic vectorization capabilities are modest, compared to PGI and CCE
  - Use of inline assembly is encouraged
  - Focus is more on best speed for scalar, non-scaling apps
  - Tuned for Intel architectures, but actually works well for some applications on AMD (this is becoming increasingly important)

Compiler Choices – Relative Strengths

…from Cray's Perspective

- **GNU Inproving Fortran, outstanding C and C++ (if you ignore vectorization)**
  - Obviously, the best for gcc compatability
  - Scalar optimizer was recently rewritten and is very good
  - Vectorization capabilities focus mostly on inline assembly, but automatic vectorization is improving
  - Note: may be required to recompile when changing between major version (4.5 -> 4.6, for example)
- **CCE – Outstanding Fortran, very good C, and okay C++**
  - Very good vectorization
  - Very good Fortran language support; only real choice for Coarrays
  - C support is quite good, with UPC support
  - Very good scalar optimization and automatic parallelization
  - Clean implementation of OpenMP 3.0, with tasks
  - Sole delivery focus is on Linux-based Cray hardware systems
  - Best bug turnaround time (if it isn't, let us know!)
  - Cleanest integration with other Cray tools (performance tools, debuggers, upcoming productivity tools)
  - No inline assembly support

Starting Points for Each Compiler

- **PGI**
  - `-fast -Mipa=fast(,safe)`
  - If you can be flexible with precision, also try `-Mfprelaxed`
  - Compiler feedback: `-Minfo=all -Mneginfo`
  - `man pgf90; man pgcc; man pgCC;` or `pgf90 -help`
- **Cray**
  - <none, turned on by default>
  - Compiler feedback: `-rm` (Fortran) `-hlist=m` (C)
  - If you know you don't want OpenMP: `-xomp` or `-Othread0`
  - `man crayftn; man craycc ; man crayCC`
- **Pathscale**
  - `-Ofast` Note: this is a little looser with precision than other compilers
  - Compiler feedback: `-LNO:simd_verbose=ON:vintr_verbose=ON:prefetch_verbose=ON`
  - `man eko` ("Every Known Optimization")
- **GNU**
  - `-O2` / `-O3`
  - Compiler feedback: `-ftree-vectorizer-verbose=1`
  - `man gfortran; man gcc; man g++`
- **Intel**
  - `-fast`
  - Compiler feedback: `-vec-report1`
  - `man ifort; man icc; man iCC`

Cray Scientific Libraries

- Goal of scientific libraries

  Improve Productivity at optimal performance

- Cray use four concentrations to achieve this
  - **Standardization**
    - Use standard or “de facto” standard interfaces whenever available
  - **Hand tuning**
    - Use extensive knowledge of target processor and network to optimize common code patterns
  - **Auto-tuning**
    - Automate code generation and a huge number of empirical performance evaluations to configure software to the target platforms
  - **Adaptive Libraries**
    - Make runtime decisions to choose the best kernel/library/routine

Cray Scientific Libraries

FFT	Dense	Sparse
CRAFFT	BLAS	CASK
FFTW	LAPACK	PETSc
P-CRAFFT	ScaLAPACK	Trilinos
	IRT	
	CASE	

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

LibSci Provides...

- BLAS
- LAPACK
- SCALAPACK
- BLACS
- PBLAS
- ACML
- FFTW 2&3
- PETSC
- TRILINOS
- IRT*
- MUMPS
- ParMetis
- SuperLU
- SuperLU_dist
- Hypre
- Scotch
- Sundials
- CASK*
- CRAFFT*
- CASE*

* Cray-specific

Cray MPT Features

- **Full MPI2 support (except process spawning) based on ANL MPICH2**
  - Cray used the MPICH2 Nemesis layer for Gemini
  - Cray-tuned collectives
  - Cray-tuned ROMIO for MPI-IO

  - Current Release: 5.3.0 (MPICH 1.3.1)
    - Improved MPI_Allreduce and MPI_alltoallv
    - Initial support for checkpoint/restart for MPI or Cray SHMEM on XE systems
    - Improved support for MPI thread safety.
    - `module load xt-mpich2`
- **Tuned SHMEM library**
  - `module load xt-shmem`


JAGUAR
APPLIED
MECHANICS
Railway
Technical
Research
Institute
H L R I S
CRAY XT5m
KRAKEN

Thank You!