CUG 2008 HELSINKI • MAY 5–8, 2008
CROSSING THE BOUNDARIES

Cray Programming Environment Update & Roadmap

Luiz DeRose
Programming Environment Director
Cray Inc.

Cray Programming Environment Focus

- It is the role of the Programming Environment to **close the gap** between observed performance and peak performance
  - Help users achieve ***highest possible performance*** from the hardware

- The Cray Programming Environment addresses issues of scale and complexity of high end HPC systems.
  - The Cray Programming Environment helps users to be more **productive**
  - It is the place at which the **complexity** of a system **is hidden** from the user

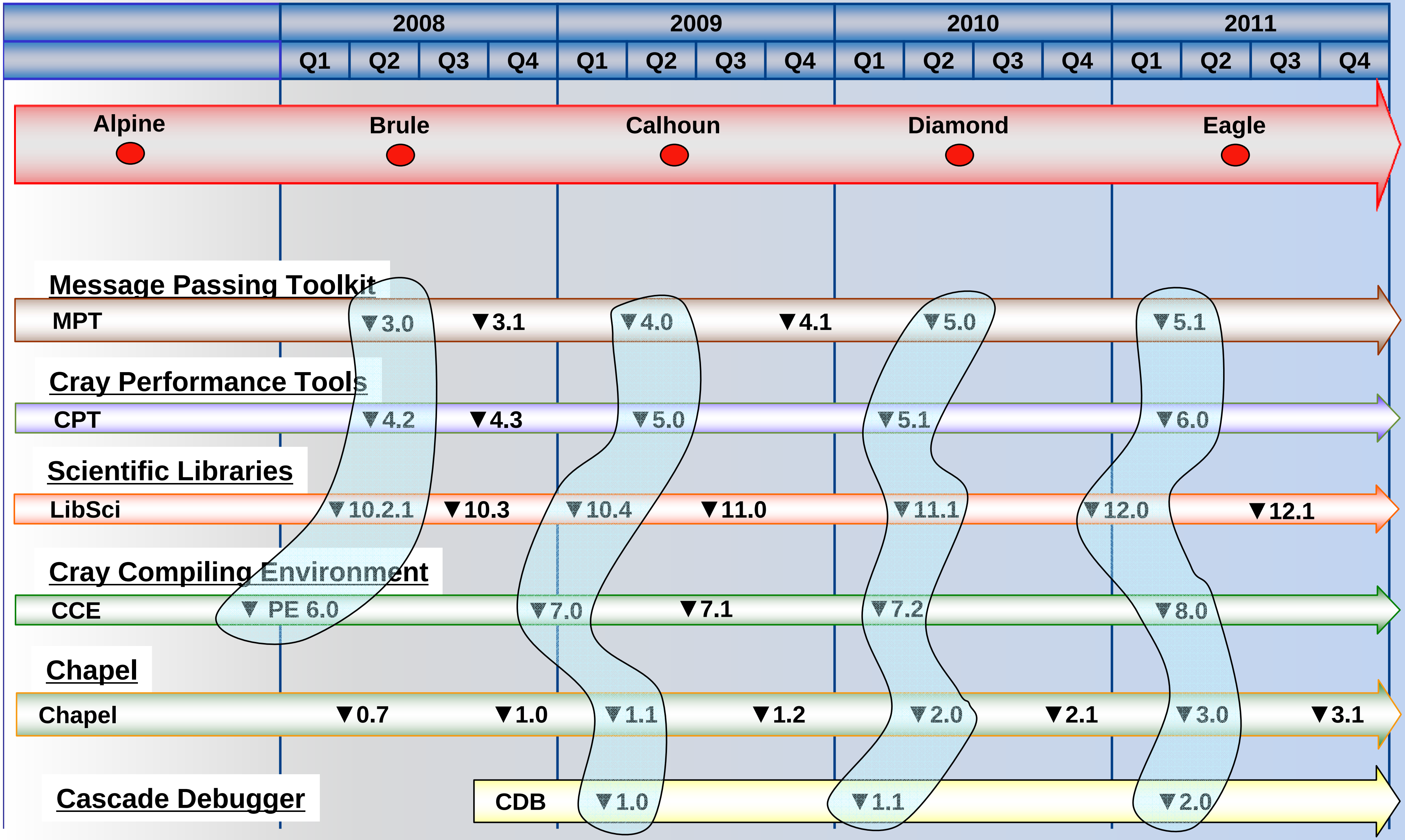
- User **productivity** is **enhanced** with
  - Increase of ***automation***
  - ***Ease of use***
  - Extended ***functionality*** and improved ***reliability***
  - Close ***interaction with users*** for feedback targeting functionality enhancements

Cray Programming Environment

- Programming Languages
  - Fortran
  - C
  - C++
  - Chapel [#]
  - Java (Service nodes)
- Programming models
  - Distributed Memory
    - MPI
    - SHMEM
  - Shared Memory
    - OpenMP
  - PGAS
    - UPC
    - CAF [1]
- Tools
  - Environment setup
    - Modules
  - Debuggers
    - TotalView
    - DDT [2]
    - lgdb [#]
  - Performance analysis
    - CrayPat
    - Cray Apprentice[2]
- Optimized Math Libraries
  - LibSci
    - libgoto [2]
    - Iterative Refinement Toolkit
    - LAPACK
    - ScaLAPCK
    - SuperLU
  - Cray PETSc
    - CASK [2#]
  - CRAFFT [2#]
  - Fast-mv [2#]

1: X2 Only 2: XT Only #: Under development

Programming Environment Releases



Compilers for the XT Systems

- PGI
  - Provide C, C++, F77, F90, & 95
  - PGI 7.1.6 released in March 2008
- PathScale
  - Provide C, C++, F77, F90, & 95
  - PathScale 3.1 released in January 2008
- GNU
  - XT gcc 4.2.3 released in February 2008
  - XT gcc 4.2.0 (Quad core only) released in March 2008
  - XT 4.3 planned for May 2008
- UPC
  - XT UPC 1.0.2 Released in September 2007
    - BUPC
    - GCCUPC

Chapel

- Chapel Version 0.7 Released in March 08
  - Limited availability
  - Revised chapters of language specification
    - Parallelism and locality
  - Initial support for task parallelism on multiple locales
  - Support for execution on the Cray XT

- First public release of Chapel targeted to 4Q08

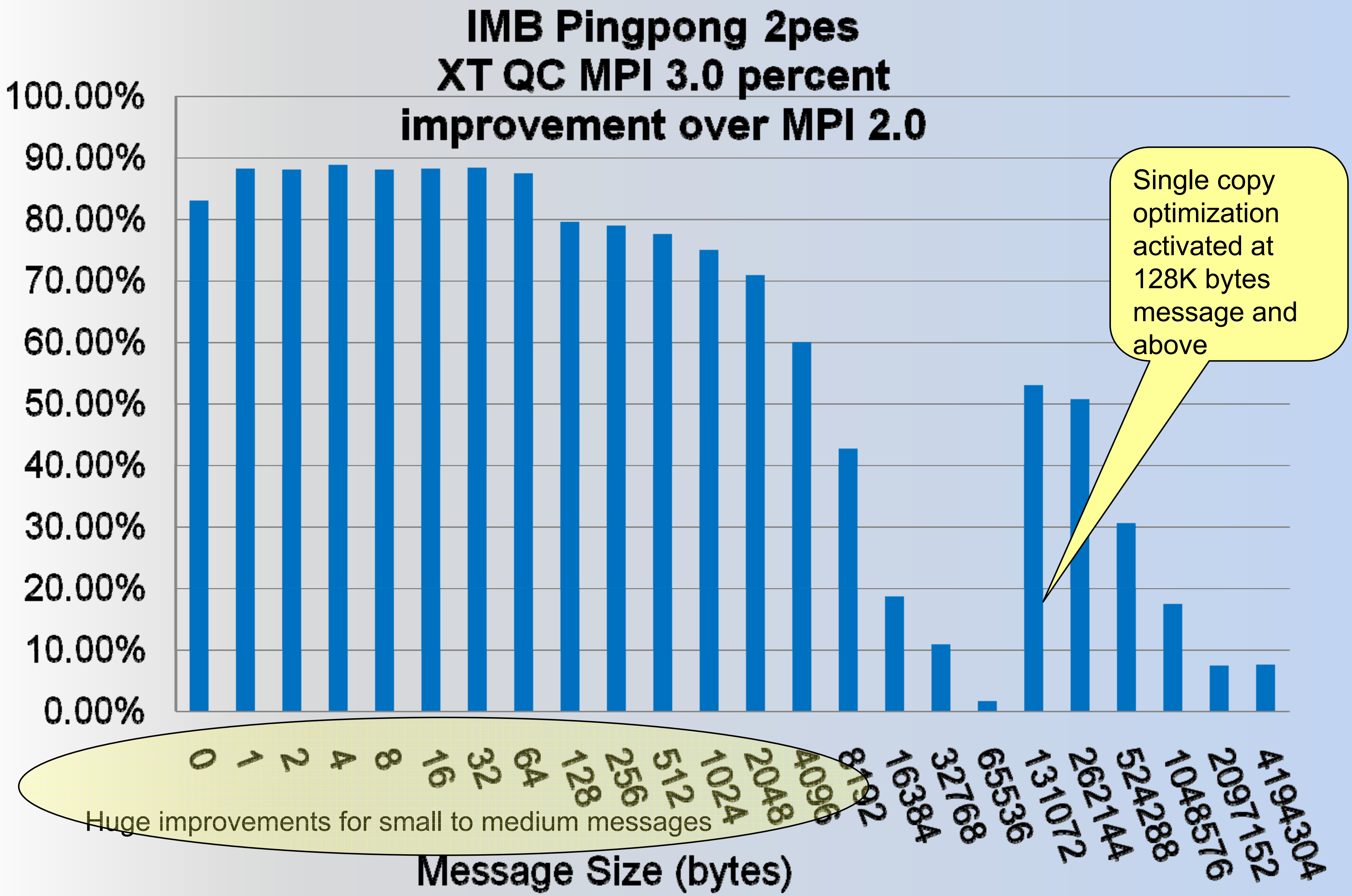
MPI & Cray SHMEM

- MPI
  - Implementation based on MPICH2 from ANL
  - 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)

- Cray SHMEM
  - Fully optimized Cray SHMEM library supported
    - XT4 implementation close to the T3E model
      - Cray SHMEM is layered directly on top of Portals

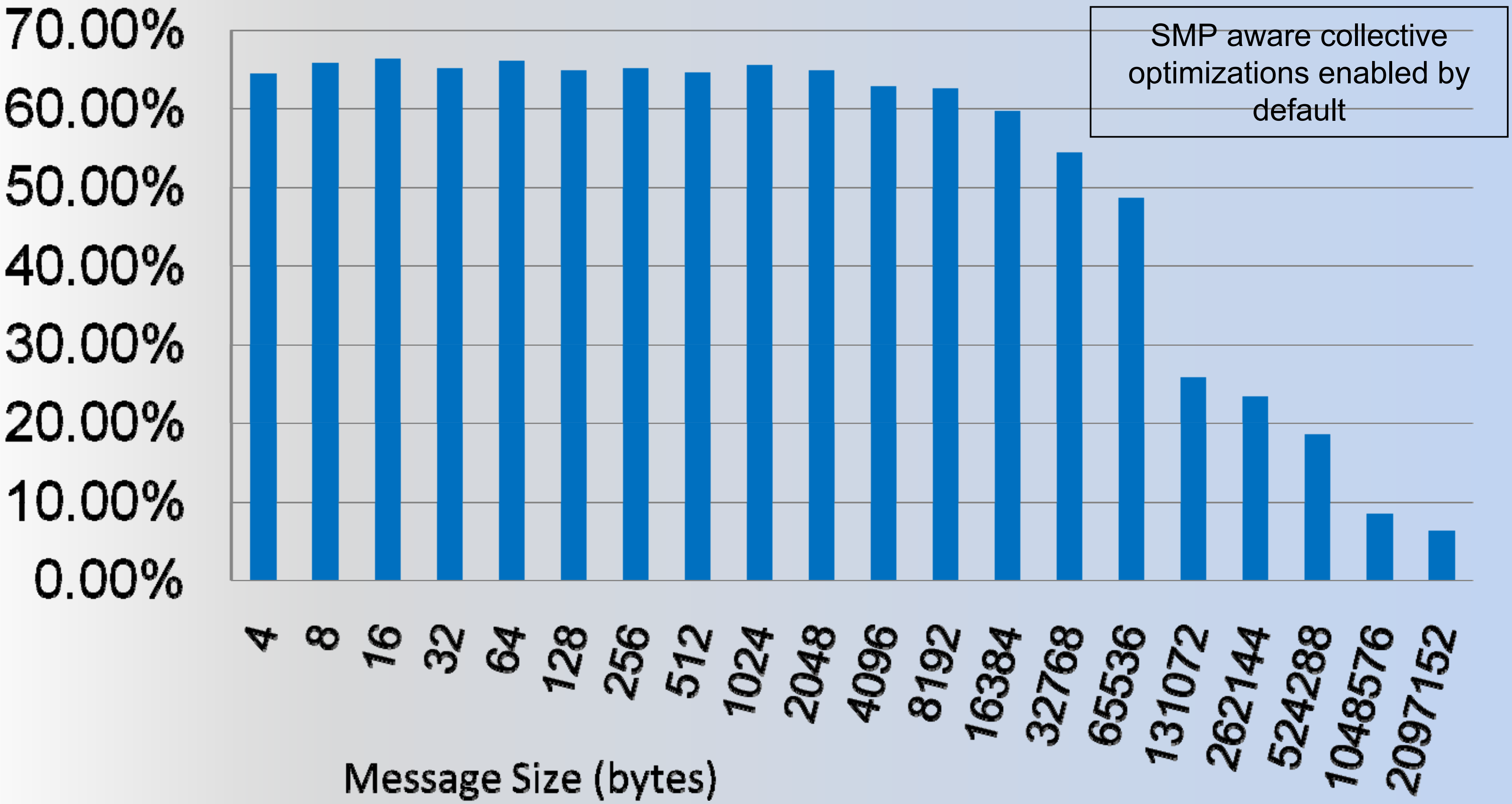
New XT MPI implementation (Cray MPI 3.0)

- Cray XT MPI 3.0 uses Cray X2 MPI as base and merge of MPICH 1.0.5
- Cray MPI 3.0 (Released in April 08)
  - On-node 0 byte latency **less than .4 usecs**
  - Off-node 0 byte latency less than 6 usecs
  - Supports the following MPI ADI devices
    - **Portals device**
      - Used between nodes on XT (completely rewritten from MPI 2.0)
    - **Shared memory device**
      - Used for X2 and XT MPI 3.0 and future Cray platforms
      - Used for on-node messaging
    - **Distributed Memory device**
      - Scalable device used between nodes on the X2
  - Supports multiple ADI devices running concurrently
    - **Fastest path automatically chosen**
  - More environment variables set by default (example MPI_COLL_OPT_ON)
    - SMP aware optimized collectives now default

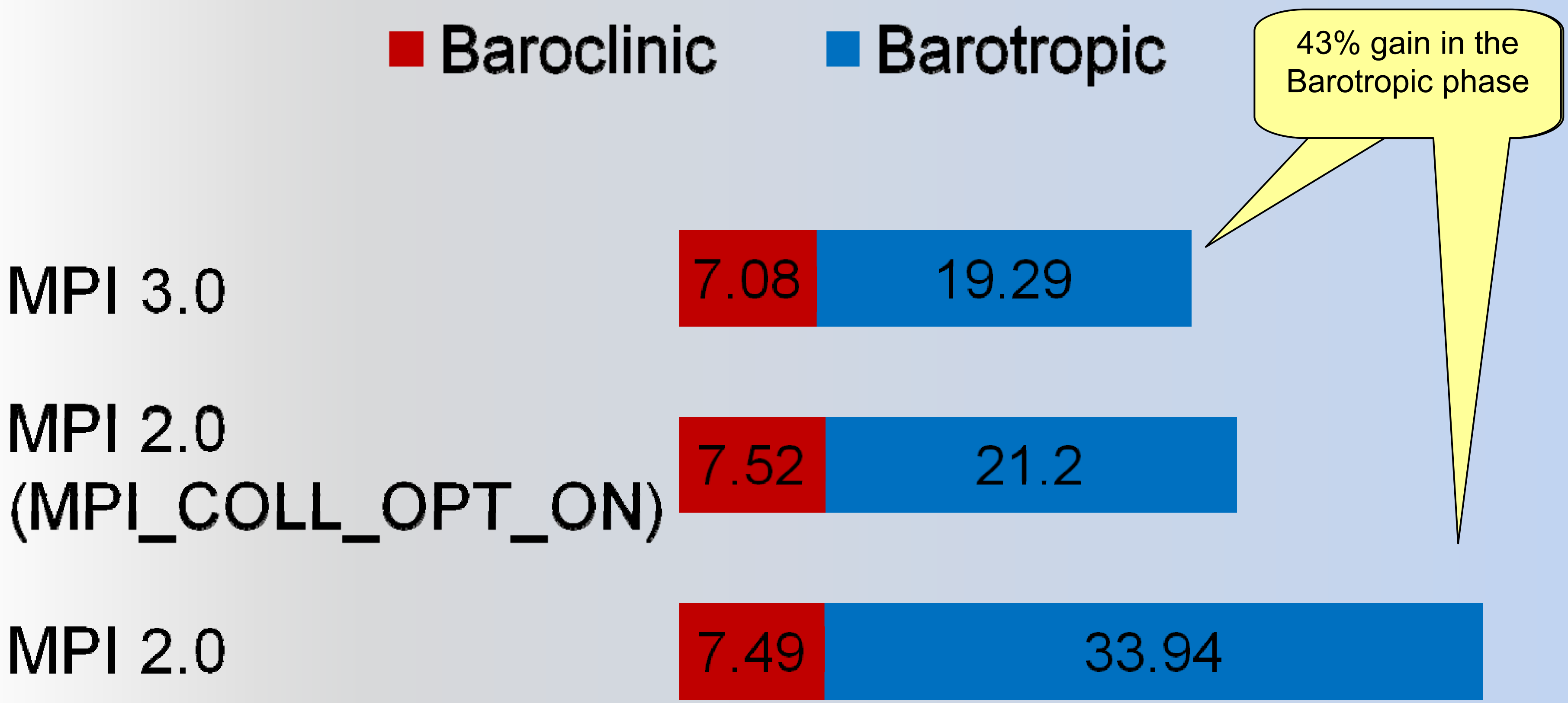


IMB Pingpong 2pes
XT QC MPI 3.0 percent
improvement over MPI 2.0
100.00%
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
0
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
131072
262144
524288
1048576
2097152
4194304
Message Size (bytes)
Single copy optimization activated at 128K bytes message and above
Huge improvements for small to medium messages

IMB Allreduce 256pes
XT QC MPI 3.0
percent improvement over MPI 2.0



POP 512pes XT QC (in seconds)
10 nday steps with 1 degree
(small case near end of scaling)



The Cray Performance Tools Strategy

- Must be easy and flexible to use
  - Automatic program instrumentation
    - No source code or makefile modification needed
  - Automatic Profiling Analysis (APA)
  - Profile Guided Rank Placement Suggestions

- Integrated performance tools solution
  - Multiple platforms
  - Multiple functionality
    - Measurements of user functions, MPI, I/O, memory, & math SW
    - HW Counters support

Cray Performance Tools Recent Work

- Focus on reliability, scalability, and automation
- Focus on new systems support (X2, QC, CLE)
- Expand types of performance statistics available
  - Load balance metrics
  - OpenMP support available with Cray Tools 4.2
    - Sampling
    - Support of OpenMP trace points within Cray compiler (X2 only)
    - New user API for OpenMP tracing (for ISV compilers)
      - Support of OpenMP trace points within PGI 7.2
    - Support for OpenMP runtime library calls (all compilers)
    - OpenMP runtime library calls grouped separately from OpenMP API calls

Cray Performance Tools Directions

- Automatic performance analysis
  - Use of performance models to automatically identify and expose performance anomalies
    - Load imbalance
    - Communication / synchronization / I/O problems
    - Environment variables
    - etc
- Recent work towards automatic performance analysis
  - Determined pattern representation
    - Will expand on existing infrastructure
  - Built basic recommendation infrastructure in CrayPat
    - Support MPI rank placement suggestions
  - Increasing level of data collection/analysis automation
    - Automatic Profiling Analysis
- Scalable visualizer

Automatic Profiling Analysis

- Example of our approach to analyze the performance data and direct the user to meaningful information
- Simplifies the procedure to instrument and collect performance data for novice users
- Based on a two phase mechanism
  1. Automatically detects the most time consuming functions in the application and feeds this information back to the tool for further (and focused) data collection
  2. Provides performance information on the most significant parts of the application

APA File Example

```
#  You can edit this file, if desired, and use it
#  to reinstrument the program for tracing like this:
#
#          pat_build -O ft.ind.B.2+pat+5257-770sdt.apa
#
#  These suggested trace options are based on data from:
#
#      /work/users/luizd/COE_Workshop/run/ft.ind.B.2+pat+5257-
      770sdt.xf

# ----------------------------------------------------------------------

#      HWPC group to collect by default.

  -Drtenv=PAT_RT_HWPC=0  # Summary with instructions metrics.

# ----------------------------------------------------------------------

#      Libraries to trace.

  -g mpi

# ----------------------------------------------------------------------

#      User-defined functions to trace, sorted by % of samples.
#      Limited to top 200. A function is commented out if it has < 1%
#      of samples, or if a cumulative threshold of 90% has been
      reached.

  -w  # Enable tracing of user-defined functions.
      # Note: -u should NOT be specified as an additional option.
```

```
# 37.70%
        -T fftz2_

# 26.23%
        -T cffts2_

#  9.37%
        -T transpose2_local_

#  8.96%
        -T cffts1_

#  7.82%
        -T evolve_

#  Functions below this point account for less than 10% of samples.


#  6.43%
#        -T transpose2_finish_

#  2.72%
#        -T cfftz_

#  0.48%
#        -T vranlc_

#  0.28%
#        -T compute_indexmap_

# ----------------------------------------------------------------------

  -o ft.ind.B.2+apa                    # New instrumented program.

  /work/users/luizd/COE_Workshop/bin/ft.ind.B.2  # Original
      program.
```

Math Software Stack + upcoming features

LibSci
- ScaLAPACK
- BLAS (libGoto)
- LAPACK
- SuperLU_dist
- IRT
- CRAFFT
- CASK
- Fast MV

PETSc
- PETSc
- HYPRE
- MUMPS
- SuperLU
- ParMETIS

FFT
- FFTW

ACML
- FFT
- RNG

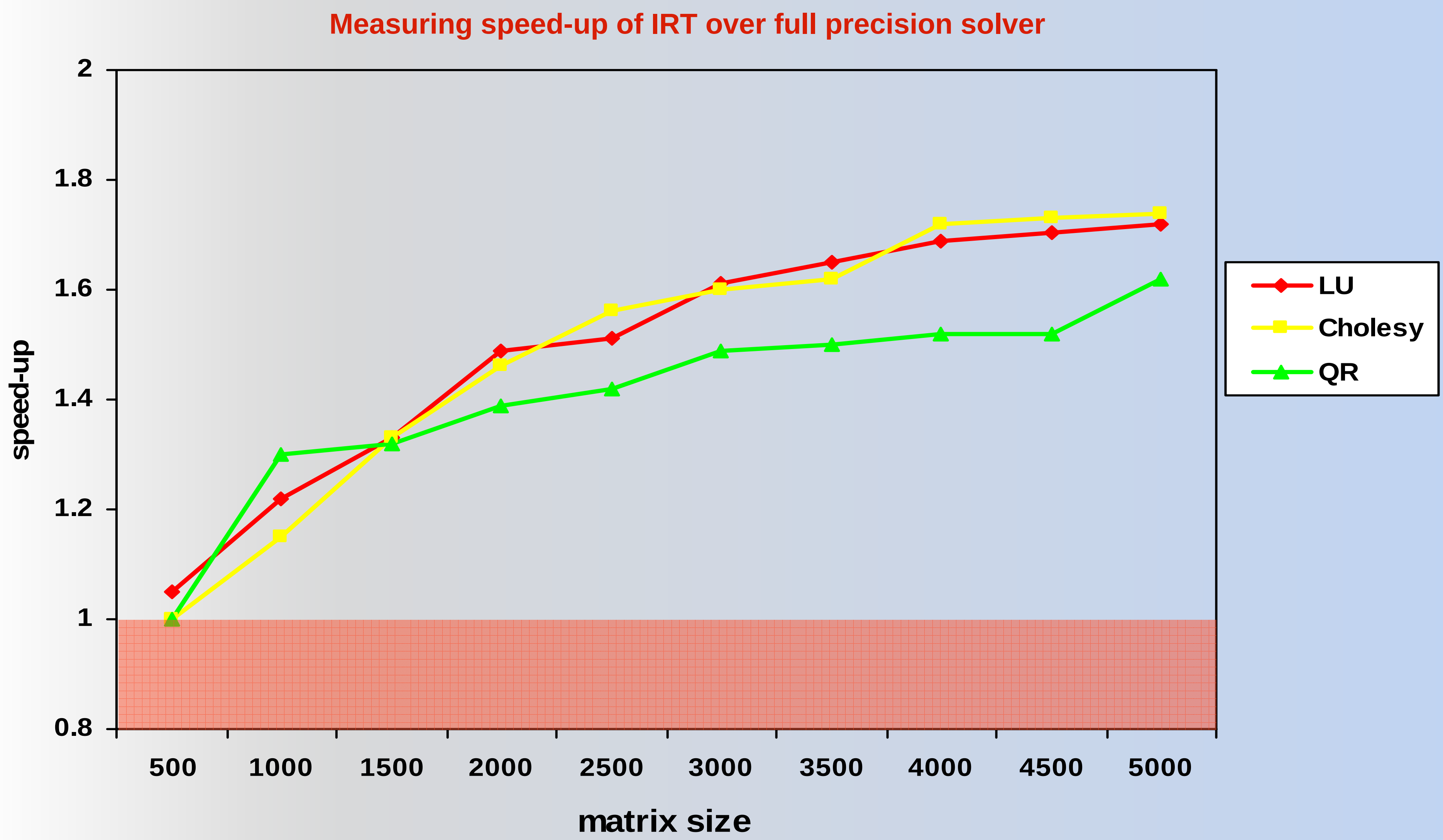
Recent Work

- Released LibSci 10.2.0 (and 10.2.1)
  - Added Goto + custom BLAS / LAPACK
    - Provided **significant performance improvements** over ACML.
  - LAPACK
  - Mixed mode ScaLAPACK support
    - MPI across sockets (1 BLACS process per socket)
    - Threaded BLAS within sockets
- Released PETSc 2.3.3
  - PETSc + HYPRE, SuperLU, MUMPS, ParMETIS
- Released IRT2.0 automatic interfaces
- libsci-10.3.0 will contain considerable performance improvements
  - CASK will improve iterative solver performance by 5-25% (problem dependent)
  - Cray Adaptive FFT

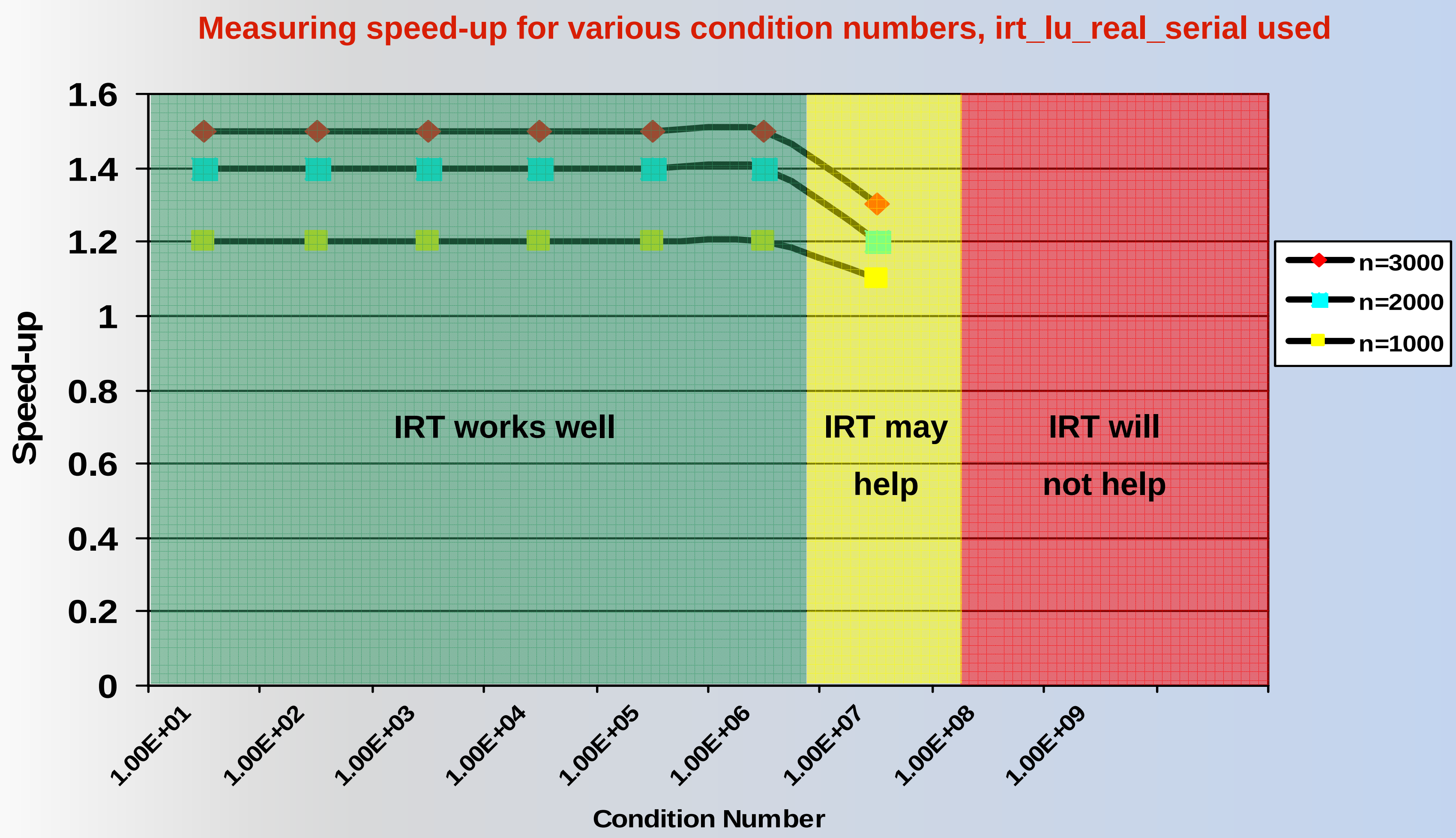
Iterative Refinement Toolkit

- Solves linear systems in single precision whilst obtaining solutions accurate to double precision
  - For well conditioned problems
- Serial and Parallel versions of LU, Cholesky, and QR
- With LibSci-10.2.0, there are now 2 ways to use the library
  1. **IRT Benchmark routines**
     - Uses IRT 'under-the-covers' without changing your code
       - Simply set an environment variable
       - Useful when you just want a quick-and-dirty factor/solve
  2. **Advanced IRT API** (from libsci-10.1.0)
     - If greater control of the iterative refinement process is required
       - Allows
         - condition number estimation
         - error bounds return
         - minimization of either forward or backward error
         - 'fall back' to full precision if the condition number is too high or IRT fails
         - max number of iterations can be altered by users

IRT2.0 performance (serial)

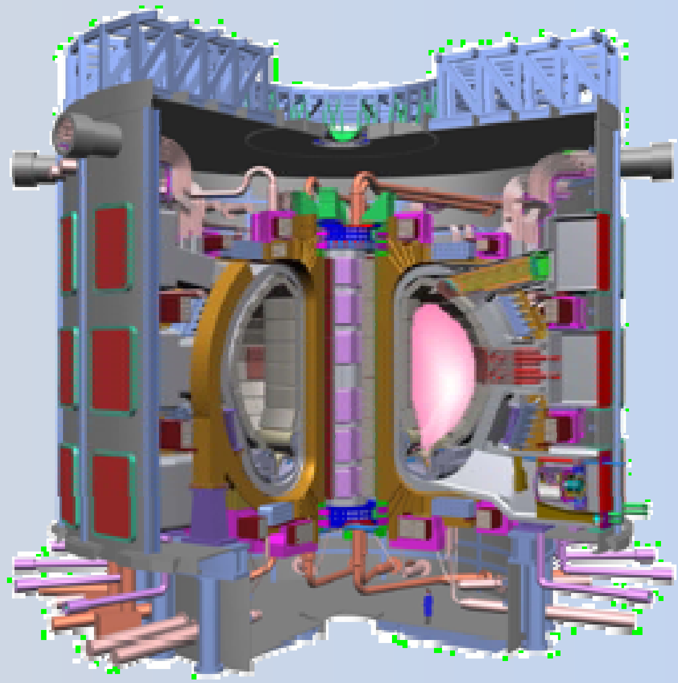


IRT on XT4 (Condition vs. performance)

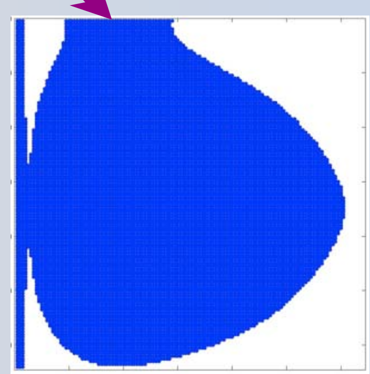


Fusion Energy: AORSA

- rf heating in tokamak
- Maxwell-Bolzmann Eqns
- FFT
- Dense linear system
- Calc Quasi-linear op
- INCITE: “High Power Electromagnetic Wave Heating in the ITER Burning Plasma”



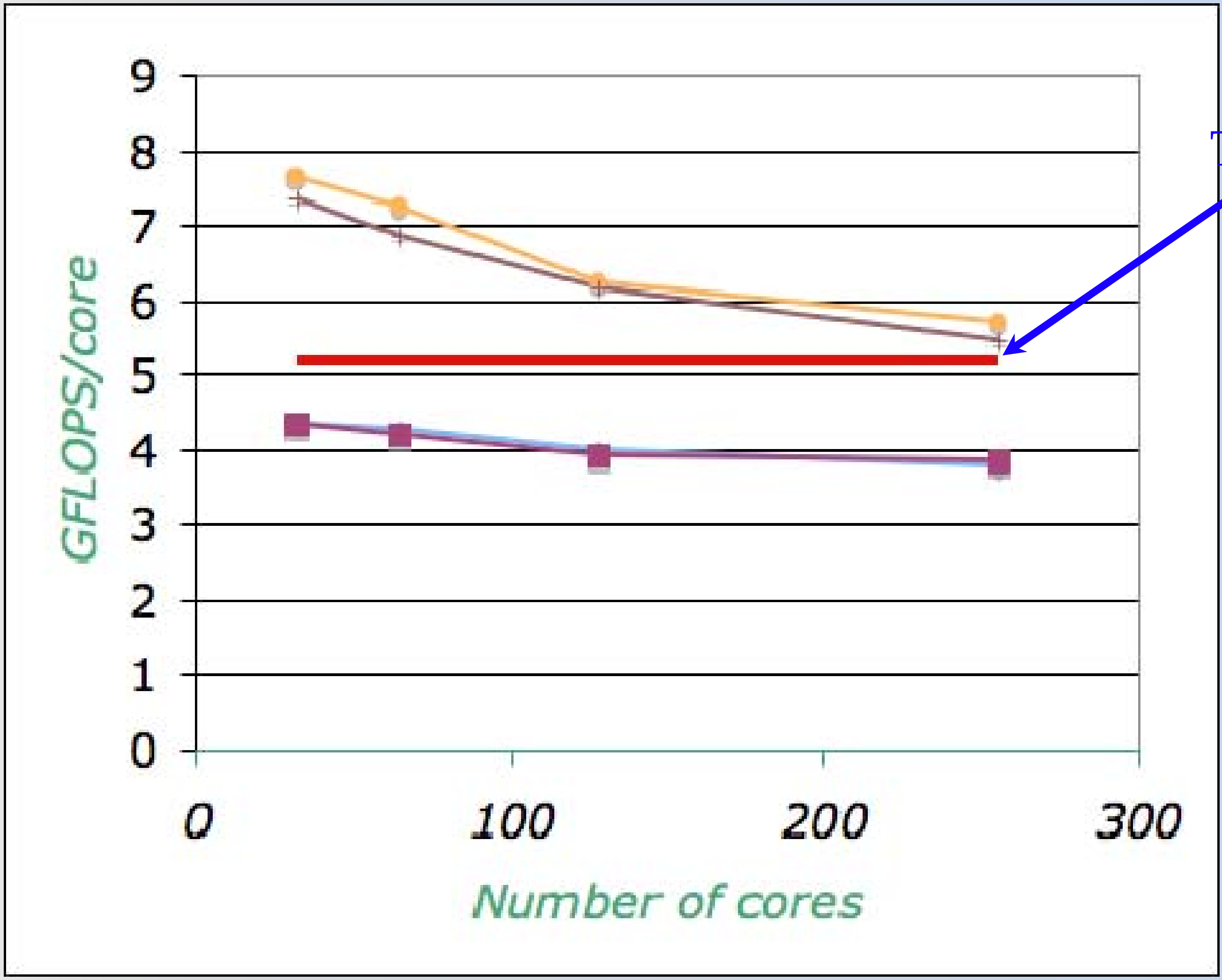
ITER-FEAT



Courtesy
Richard Barrett

Oak Ridge National Laboratory

AORSA solver performance - 128x128 grid



Theoretical Peak

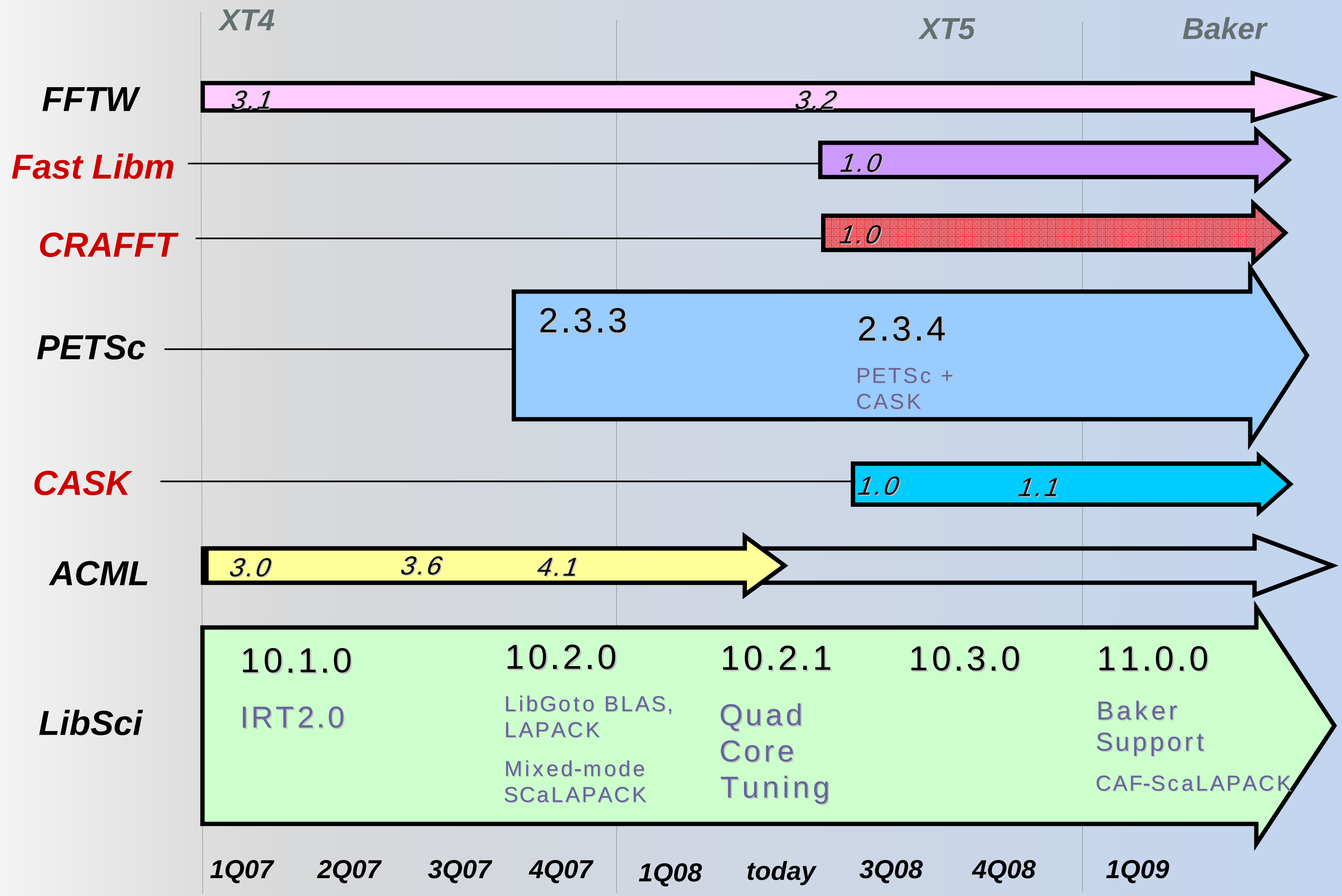
Courtesy
Richard Barrett

Oak Ridge National Laboratory

Math SW Focus in 2008

- **Auto-tuning:** use code generator and automatic tester to develop codes
  - Cray Adaptive Sparse Kernels (CASK)
- **Adaptivity:** make runtime decisions to choose best kernel/library/routine
  - Cray Adaptive FFT (CRAFFT)
  - CASK
- **Performance:**
  - Iterative Solver Performance
  - FFT performance
  - Quad-core optimization
  - Fast libm

Math Software Roadmap



Summary: Cray Programming Environment Strengths

- Leading edge software for HPC
  - State of the art compilers, MPI, math SW, and tools
- Ability to innovate targeting performance improvements
  - Only vendor to have supported PGAS throughout its existence
    - We invented it!
  - More recent advancements in scientific libraries and performance tools than any other vendor
  - Automatic performance analysis
  - Auto-tuned libraries
- Team with extensive HPC experience and advanced knowledge of parallel performance
  - Close user interaction provides essential feedback for features and functionality enhancements, allowing the development of a practical user-driven Programming Environment

Thank You!

Q&A