Cray Scientific Libraries

Luiz DeRose
Programming Environments Director
Cray Inc.
ldr@cray.com

CSC, Finland

September 21-24, 2009

Scientific Libraries

- Traditional model
  - Tuned general purpose codes
    - Only good for dense
    - Not problem sensitive
    - Not architecture sensitive

Cray Scientific Libraries Today

- 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

Standardization

- Three separate classes of standardization, each with a corresponding definition of productivity
  1. Standard interfaces (e.g., dense linear algebra)
     - Bend over backwards to keep everything the same despite increases in machine complexity. Innovate 'behind-the-scenes'
     - Productivity -> innovation to keep things simple
  2. Adoption of near-standard interfaces (e.g., sparse kernels)
     - Assume near-standards and promote those. Out-mode alternatives. Innovate 'behind-the-scenes'
     - Productivity -> innovation in the simplest areas
       - (requires the same innovation as #1 also)
  3. Simplification of non-standard interfaces (e.g., FFT)
     - Productivity -> innovation to make things simpler than they are

Hand Tuning

- Algorithmic tuning
  - Increased performance by exploiting algorithmic improvements
    - Sub-blocking, new algorithms
  - **LAPACK, ScaLAPACK**
- Kernel tuning
  - Improve the numerical kernel performance in assembly language
  - **BLAS, FFT**
- Parallel tuning
  - Exploit Cray's custom network interfaces and MPT
  - **ScaLAPACK, P-CRAFFT**

Cray Auto-Tuning Framework CrayATF

- Automation of code optimization
  - Includes automation of the following 'components'
    - Code generation
    - Compilation
    - Parameter Search
    - Batch submission
    - Result Analysis
- Allows many more optimizations to be studied
- 'Search' component means allows massive optimization space to be studied in realistic time
- Currently employed in two projects at Cray
  - Cray Adaptive Sparse Kernels (**CASK**)
  - Cray Adaptive FFT (**CRAFFT**)
- **Cray ATF is the world's first industrial Autotuner**

Adaptive Libraries

- Cray Adaptive model
  - Runtime analysis allows best library/kernel to be used dynamically
  - Extensive offline testing allows library to make decisions or remove the need for those decisions
  - Decision depends on the system, on previous performance info, obtained previously, and characteristics of calling problem
- CASK: Cray Adaptive Sparse Kernels
  - Optimize PETSc and Trilinos on Cray without the user even knowing
  - Produces thousands of tuned variants of major sparse kernels
  - At runtime, analyze matrix, select best kernel via performance model
- CRAFFT: Cray Adaptive FFT
  - Provides one very simple interface into all existing FFT libraries
  - Uses previous performance information to decide where to go
  - Allows ‘advanced’ performance with the simplest interface
  - Sits above third party FFTs and CrayFFT

September 21-24, 2009 Luiz DeRose (ldr@cray.com) © Cray Inc. 7

Cray Scientific/Math Libraries

LibSci	CASK	CRAFFT
BLAS	CASK	CRAFFT
LAPACK	PETSc	FFTW
ScaLAPACK	Trilinos	P-CRAFFT
IRT		

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

September 21-24, 2009 Luiz DeRose (ldr@cray.com) © Cray Inc. 8

PETSc (Portable, Extensible Toolkit for Scientific Computation)

- Serial and Parallel versions of sparse iterative linear solvers
  - Suites of iterative solvers
    - CG, GMRES, BiCG, QMR, etc.
  - Suites of preconditioning methods
    - IC, ILU, diagonal block (ILU/IC), Additive Schwartz, Jacobi, SOR
  - Support block sparse matrix data format for better performance
  - Interface to external packages (ScaLAPACK, SuperLU_DIST)
  - Fortran and C support
  - Newton-type nonlinear solvers
- Large user community
  - DoE Labs, PSC, CSCS, CSC, ERDC, AWE and more.
- http://www-unix.mcs.anl.gov/petsc/petsc-as

PETSc External Packages

- Cray provides state-of-the art scientific computing packages to strengthen the capability of PETSc
  - Hypre: scalable parallel preconditioners
    - AMG (Very scalable and efficient for specific class of problems)
    - 2 different ILU (General purpose)
    - Sparse Approximate Inverse (General purpose)
  - ParMetis: parallel graph partitioning package
  - MUMPS: parallel multifrontal sparse direct solver
  - SuperLU: sequential version of SuperLU_DIST

Cray Adaptive Sparse Kernel (CASK)

- The CASK Process
  - Analyze matrix at minimal cost
  - Categorize matrix against internal classes
  - Based on offline experience, find best CASK code for particular matrix
  - Previously assign "best" compiler flags to CASK code
  - Assign best CASK kernel and perform Ax
- CASK silently sits beneath PETSc on Cray systems
  - Trilinos support coming soon
- Released with PETSc 3.0 in February 2009
  - Generic and blocked CSR format

Standardization Model

Large-scale application
- Highly portable
- User controlled

All systems

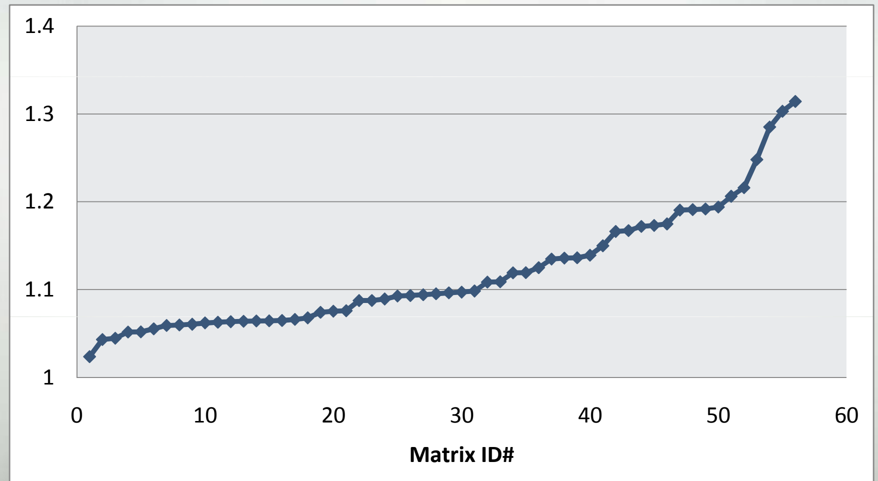
PETSc / Trilinos / Hypre
- Highly portable
- User controlled

Cray only

CASK
- XT4 & XT5 specific / tuned
- Invisible to User

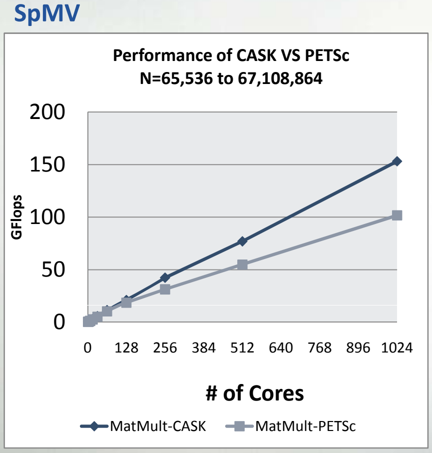
CASK and PETSc: Single Node XT5

Speedup on Parallel SpMV on 8 cores

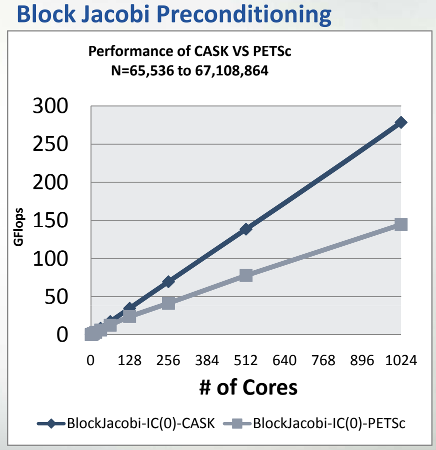


CASK Scalability (XT4)

SpMV



Block Jacobi Preconditioning



Cray Adaptive FFT (CRAFFT)

- In FFTs, the relevant problems are
  - Which library choice to use?
  - How to use complicated interfaces (e.g., FFTW)
- Standard FFT practice
  - Do a plan stage
    - Deduced machine and system information and run micro-kernels
    - Select best FFT strategy
  - Do an execute

Our system knowledge can remove some of this cost!

CRAFFT library

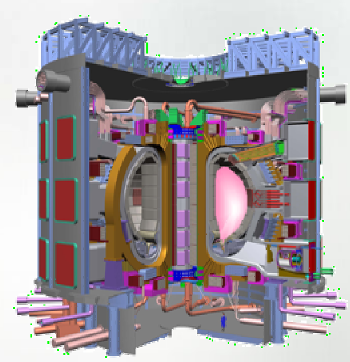
- CRAFFT is designed with simple-to-use interfaces
  - Planning and execution stage can be combined into one function call
  - Underneath the interfaces, CRAFFT calls the appropriate FFT kernel
- CRAFFT provides both offline and online tuning
  - Offline tuning
    - Which FFT kernel to use
    - Pre-computed PLANs for common-sized FFT
      - No expensive plan stages
  - Online tuning is performed as necessary at runtime as well
- At runtime, CRAFFT will **adaptively select the best** FFT kernel to use based on both offline and online testing (e.g. FFTW, Custom FFT)

Iterative Refinement Toolkit

- Solves linear systems in single precision
- Obtaining solutions accurate to double precision
  - For well conditioned problems
- Serial and Parallel versions of LU, Cholesky, and QR
- 2 usage methods
  - **IRT Benchmark routines**
    - Uses IRT 'under-the-covers' without changing your code
      - Simply set an environment variable
      - Useful when you cannot alter source code
  - **Advanced IRT API**
    - 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
        - max number of iterations can be altered by users

Example: AORSA Fusion Energy

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

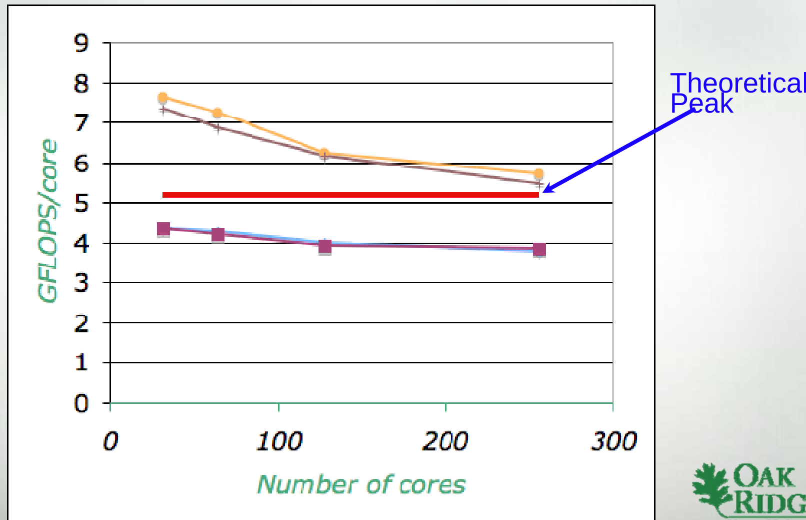


ITER-FEAT

Courtesy
Richard Barrett

Oak Ridge National Laboratory

Example: AORSA Fusion – solver performance on 128x128 grid



OAK RIDGE National Laboratory

September 21-24, 2009 Luiz DeRose (ldr@cray.com) © Cray Inc. 19

CRAY THE SUPERCOMPUTER COMPANY

Cray Scientific Libraries

Questions / Comments
Thank You!

CSC, Finland Luiz DeRose (ldr@cray.com) © Cray Inc. September 21-24, 2009