The Performance Evaluation Research Center (PERC)

Philip J. Mucci
pjmucci@lbl.gov

SC2003
Phoenix, AZ
November 20th

http://perc.nersc.gov

4 Universities: Jack Dongarra (UTK)

4 DOE Laboratories: David Bailey (LBNL)

Participating Institutions:

Argonne Natl. Lab.

Lawrence Berkeley Natl. Lab.

Lawrence Livermore Natl. Lab.

Oak Ridge Natl. Lab.

Univ. of California, San Diego

Univ. of Illinois

Univ. of Maryland

Univ. of Tennessee, Knoxville

Many supplemental contributors

PERC Overview

Mission:

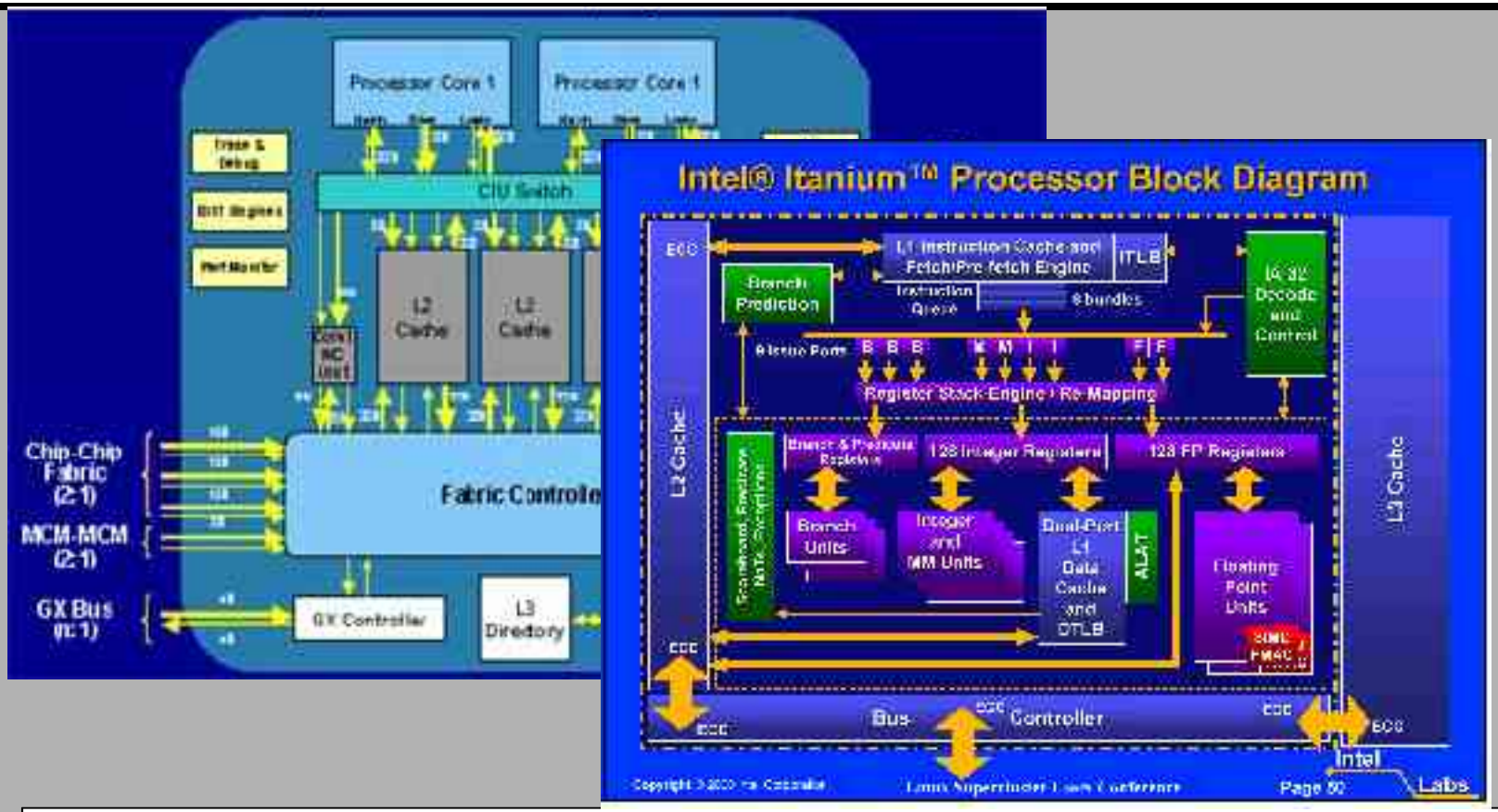
Develop a science of performance.

Engineer tools for performance analysis and optimization.

Focus:

Large, grand-challenge calculations, especially large-scale scientific codes used in SciDAC projects.

Increased System Complexity drives need for Understanding



PERC provides the

tools, models, benchmarks, and insights

Needed to understand and optimize

new computer systems and applications

Specific Objectives

Understand key factors in scientific codes that affect performance.

Understand key factors in computer systems that affect performance.

Develop models that accurately predict performance of codes on systems.

Develop an enabling infrastructure of tools for performance monitoring, modeling and optimization.

Validate these ideas and infrastructure via close collaboration with DOE Office of Science and others.

Transfer the technology to end users.

Four Primary Goals

Performance tools.

Performance modeling.

Performance optimization.

Collaboration.

Four Primary Goals

Performance tools.

Performance modeling.

Performance optimization.

Collaboration.

REAL codes.

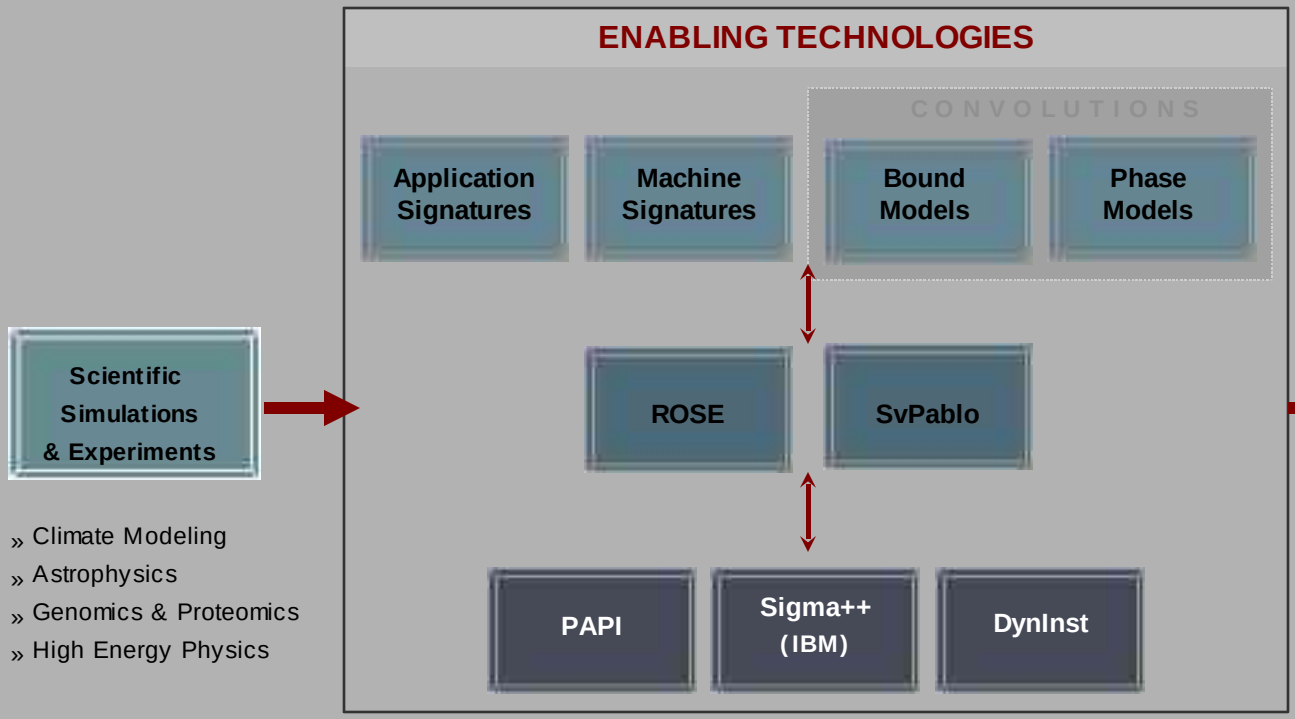
ACTUAL application teams.

DELIVERED results.

Oh My!

PERC Organization

Developing a *science* for understanding performance of scientific applications on high-end computer systems, and *engineering* strategies for improving performance on these systems.



DOE LABS

ANL LBNL
LLNL ORNL

UNIVERSITIES

UCSD UI-UC
UTenn UMD

GOALS

Optimize and Simplify:

- Profiling of real applications
- Measurement of machine capabilities
  (emphasis on memory hierarchy)
- Performance prediction
- Performance monitoring
- Informed tuning

Understand the key factors in applications that affect performance.
Understand the key factors in computer systems that affect performance.
Develop models that accurately predict performance of applications on systems.
Develop an enabling infrastructure of tools for performance monitoring, modeling and optimization.
Validate these ideas and infrastructure via close collaboration with DOE SC and other application owners.
Transfer the technology to end-users.

Performance Tools

- Measurement tools
  - PAPI (Tennessee)
    - http://icl.cs.utk.edu/projects/papi
  - SvPablo (Illinois)
    - http://www-pablo.cs.uiuc.edu/Software/SvPablo
  - Dyninst (Maryland)
    - http://www.dyninst.org
  - ROSE (LLNL)
- Tuning tools
  - Active Harmony (Maryland)
  - Performance Assertions (LLNL)
  - dsd Regularity Measurement (LLNL)
- And tools from other collaborators
  - TAU (U. Oregon)
  - PerfVar (GERRA)

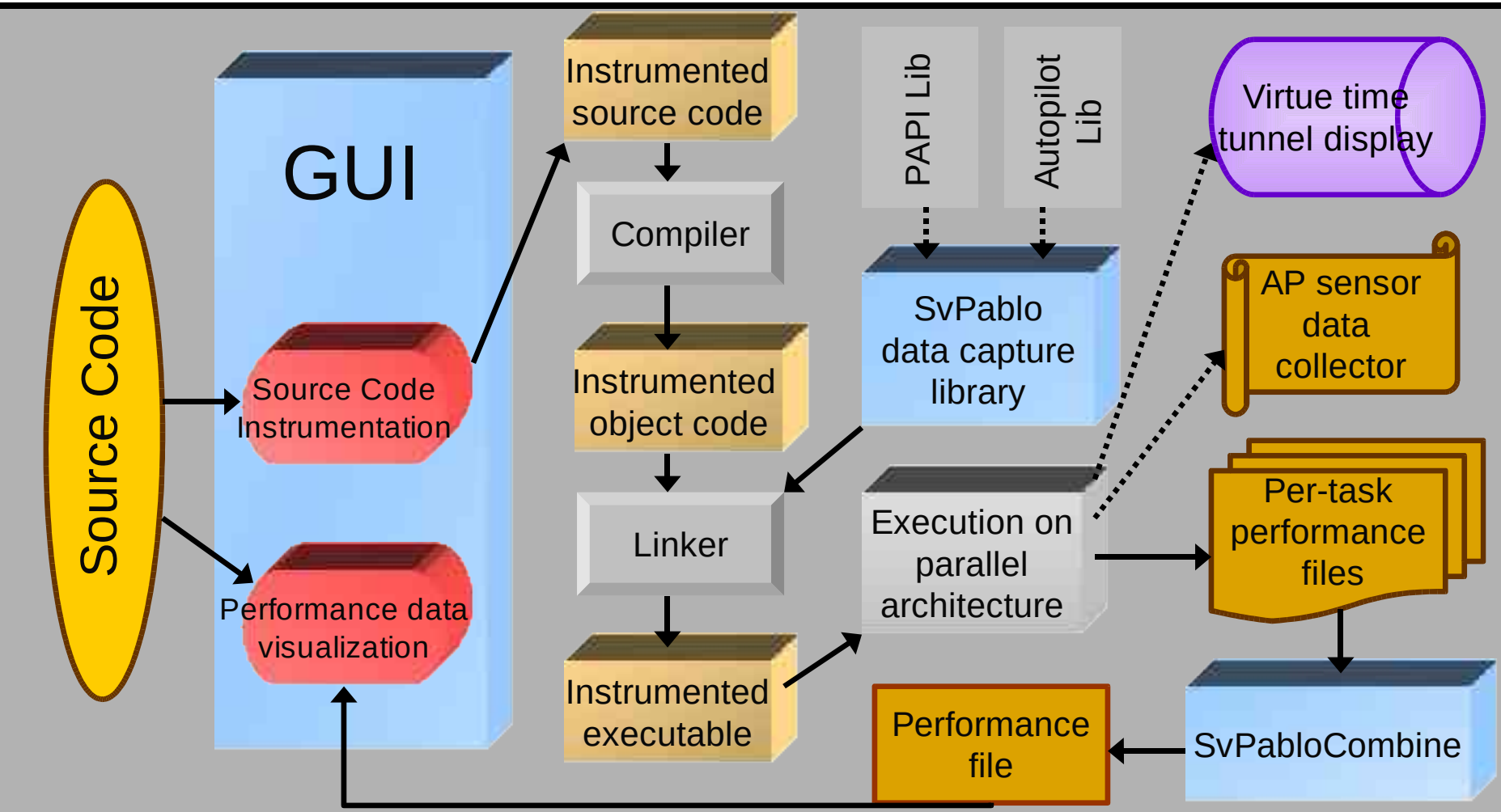
PAPI

- Performance Application Programming Interface
  - Access to hardware performance monitor counters
    - cycle counts, cache misses, floating point operations
    - Statistical profiling, multiplexing, hardware/software info
  - An ad-hoc standard for the industry
  - 2 APIs
    - High-level provides ease of use with presets
    - Low-level provides detailed access and management
- Tool uses
  - Vampir Guide View, SvPablo, HPCView, TAU, EPILOG, PSRun and many others…

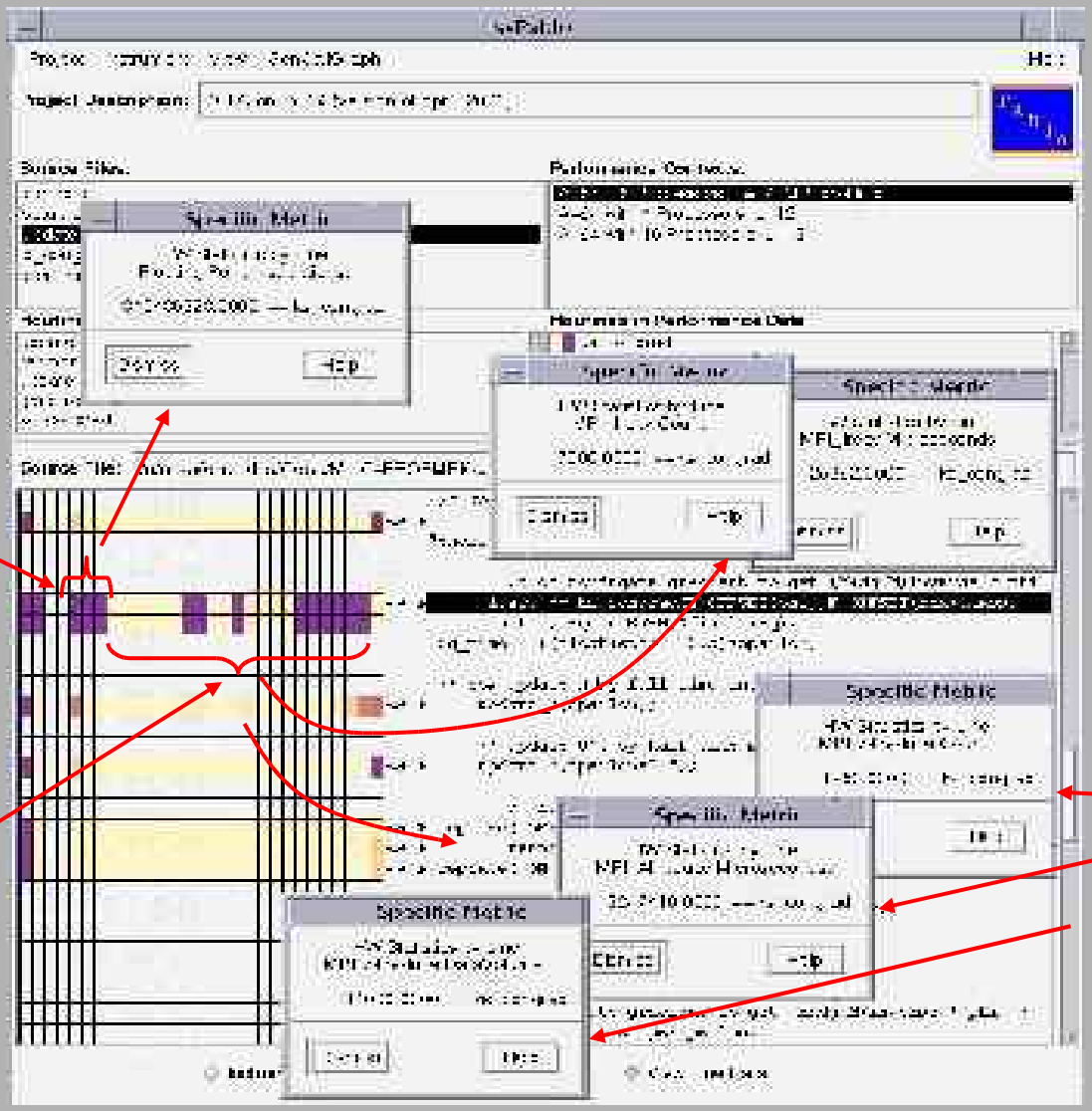
SvPablo Overview

- Graphical performance analysis environment
  - source code instrumentation
  - performance data capture, browsing & analysis
  - F77/F90 and C language support
- Performance data capture features
  - software-based instrumentation (default)
  - hardware performance counter data optional - PAPI
  - statistical summaries for long-running codes (no traces)
  - option for real-time data transmission via Autopilot
- Most HPC Platforms supported!

SvPablo Components



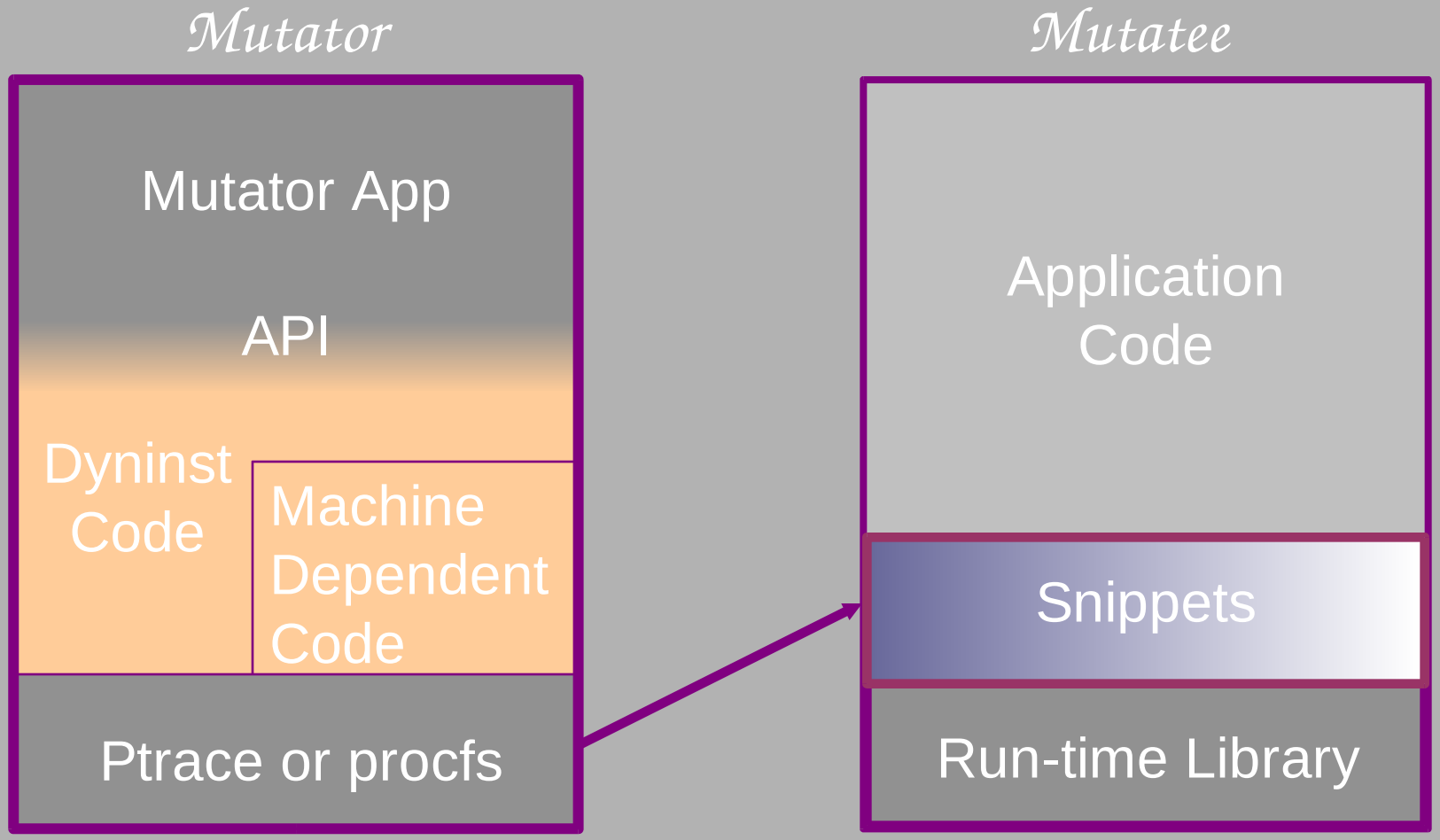
SvPablo Performance Data



dyninstAPI

- API for runtime code patching
  - new code can be added to a program while it executes
  - permits instrumentation and modification of programs
- Provides processor independent abstractions
  - same patching can be applied to multiple systems
- Includes meta-instrumentation
  - tracks overhead on inserted code

Structure of the Dyninst Library



Advantages of Runtime Code Patching

- No forethought needed
  - No user inserted probes
  - No special compiling or linking
  - Start anytime during execution
- Only insert code when needed
  - No wasted checks for “ disabled” code
  - Can add new code during execution

Dyninst Memory Instrumentation Features

- Finding memory access instructions
  - loads, stores, prefetches
- Builds on arbitrary instrumentation
- Decoded instruction information
  - type of instruction
  - constants and registers for computing
    - the effective address
    - the number of bytes moved
  - available in the mutator before execution
- Memory access snippets
  - effective address in process space
  - byte count

ROSE is a tool for building optimizing preprocessors

Inputs

Description of abstraction

Uses Optimizing Transformation (and *where* it applies)

Automatically introduce *user-defined* optimizations

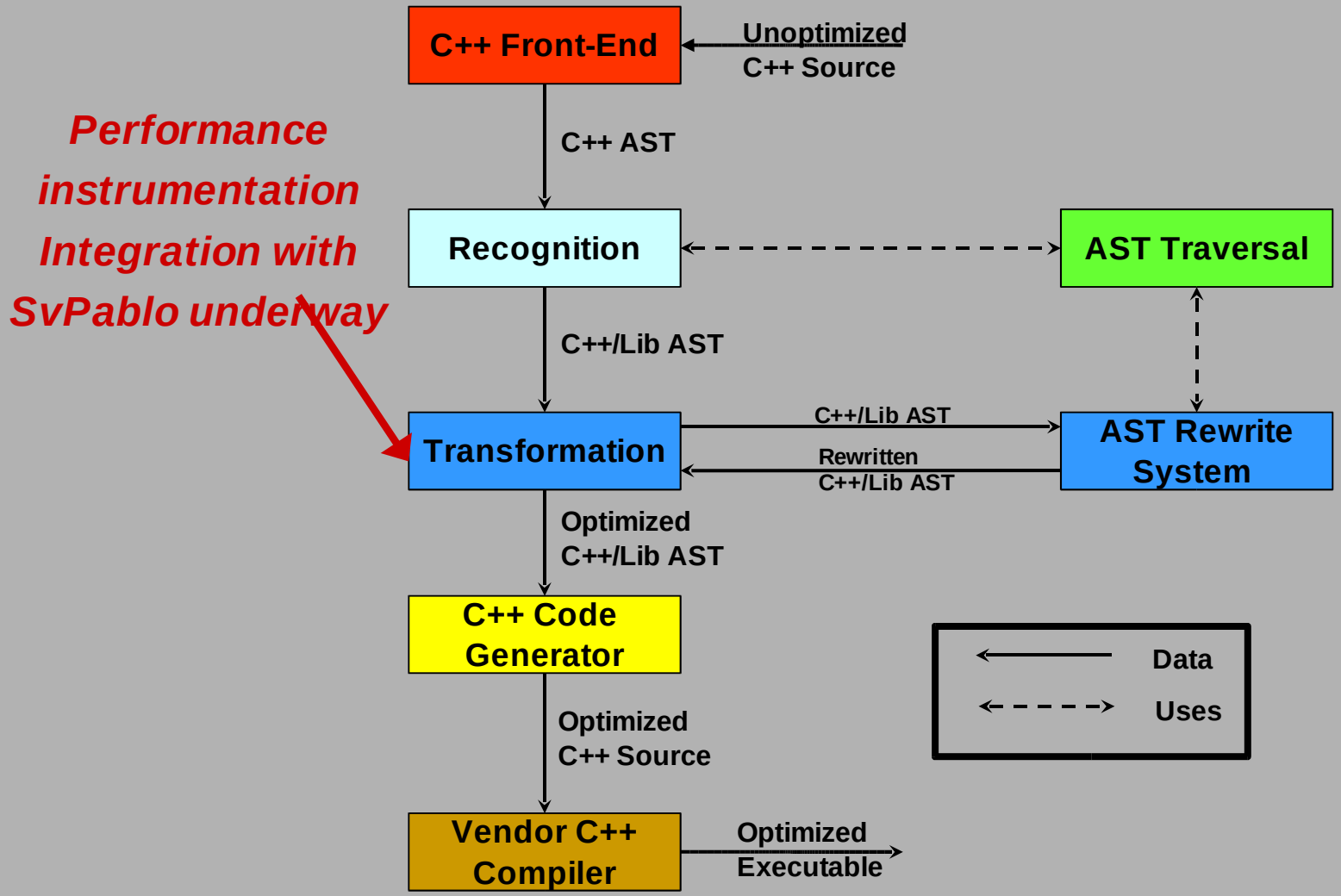
Optimization of parallel libraries within applications (e.g. MPI, OpenMP)

Target

Extract information from source code
C++ Language (including C)

Use is optional to simplify development

Overview of ROSE Approach



Why performance model?

One of two primary original goals of PERC was/remains to advance a science for understanding the performance of computers

It’ s an amusing (sad?) state of affairs when computers have usefully accurate models of climates and stars but not of themselves or other computers!

What is a Performance Model?

A calculable explanation of why a {program, application,input} tuple performs as it does

Should yield a prediction (quantifiable objective)

Performance models embody *understanding* of the factors that affect performance

- Inform the tuning process (of application and machine)
- Guide applications to the best machine
- Enable applications driven architecture design
- Extrapolate to the performance of future systems

Goals of Performance Modeling

Generation of performance models should be automated, or at least as regular and systemized as possible

Performance models *must* be time-tractable

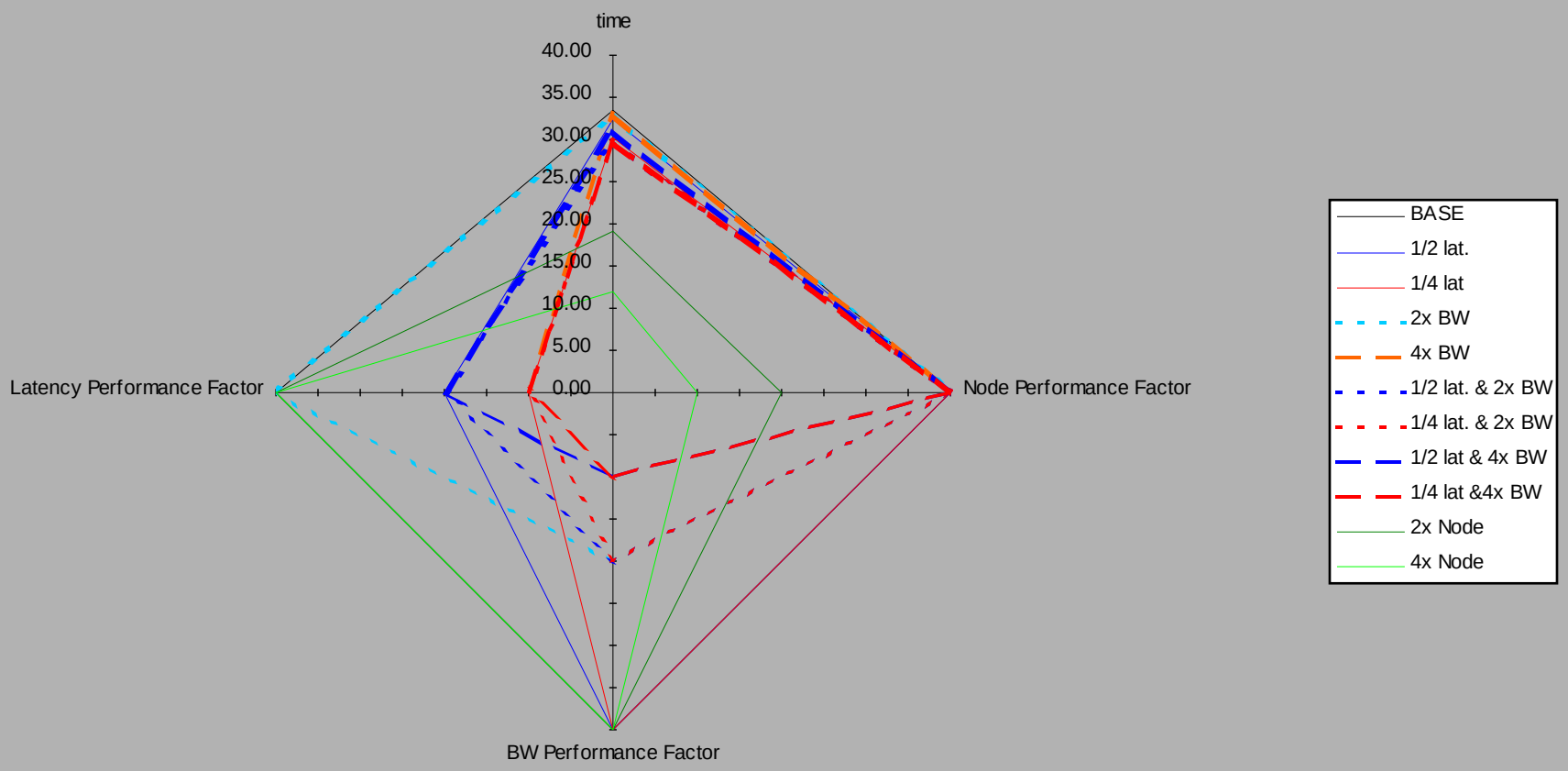
Error is acceptable if it is bounded and allows meeting these objectives

Highlight: Accurate Modeling Methodology

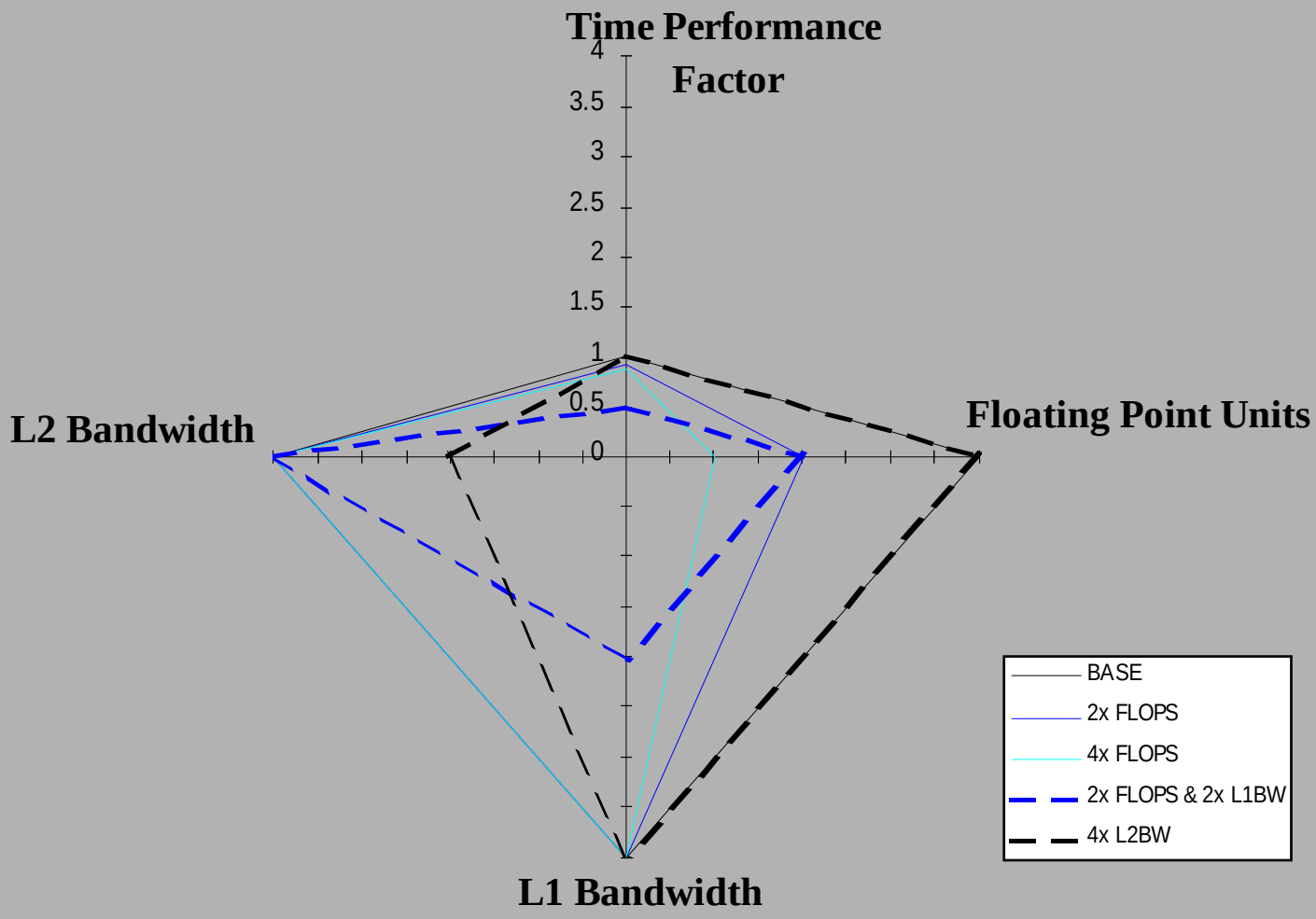
# CPUs	Real Time	Predicted Time	% Error
2	31.78	31.82	0.13
4	29.07	31.27	7.57
8	36.13	33.72	6.67
64	44.91	43.91	2.23
96	48.87	47.15	3.52
128	52.88	52.46	0.79

PETSc kernel, run on IBM SP at SDSC.

Performance Sensitivity Profiles



Break-out on node

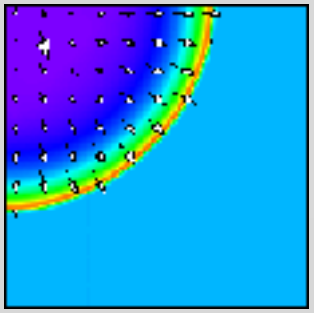


Role of Collaboration within PERC

Collaboration with the science projects is an integral part of PERC,

- providing motivation and feedback for PERC research activities,
- assuring relevance of research to the goal of improving the performance of SciDAC application codes on HPC systems, and
- enabling PERC to contribute directly to the computational science projects, accelerating achievement of science goals.

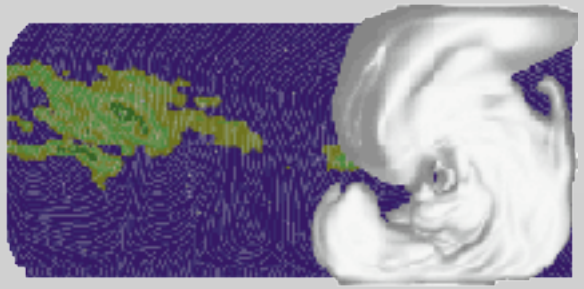
Interactions with Application Groups



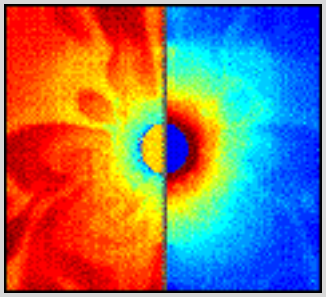
PERC has forged a set of collaborations with an emphasis on application groups in those areas that were stressed in the SciDAC call for proposals.

Collaboration with six SciDAC application groups and two SciDAC ISICs has led to application analysis and benchmarking work in five key areas:

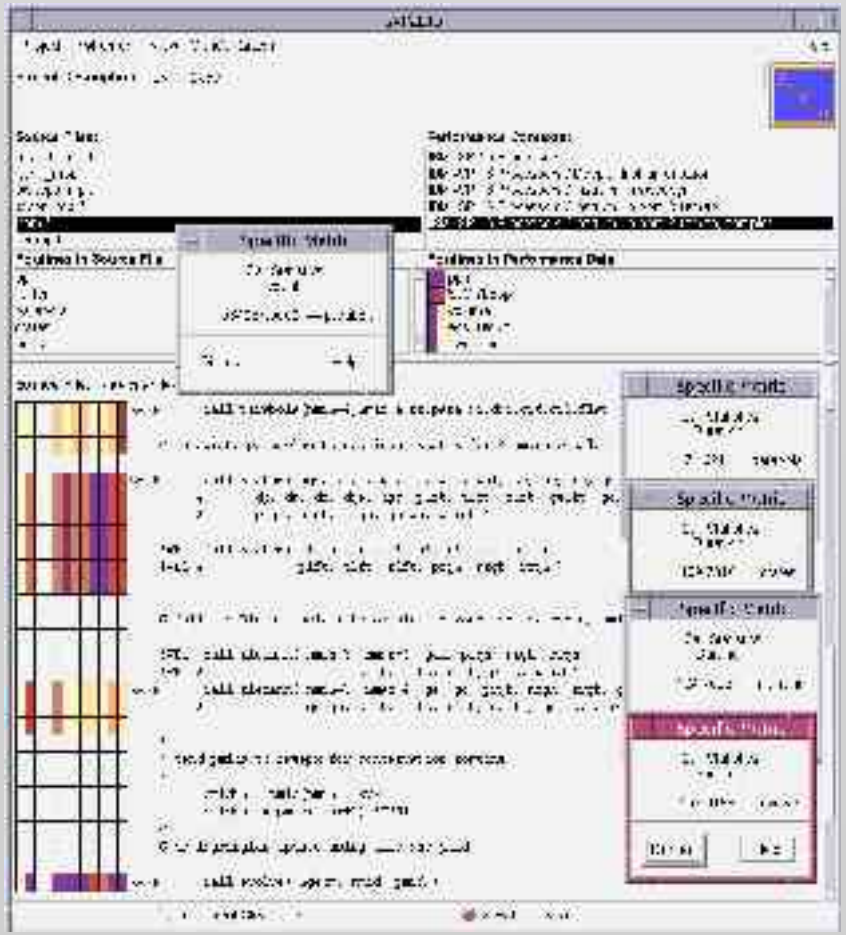
- High Energy and Nuclear Physics – EVH1, Agile-Boltztran, MILC
- Biology and Environmental Research – PCTM, CAM, CCM3
- Fusion Energy Sciences – AORSA3D
- Chemical Sciences
- Advanced Scientific Computing - pVarden



High Energy and Nuclear Physics
EVH1



Enhanced Virginia Hydrodynamics #1 represents an important kernel in the "TeraScale Simulations of Neutrino-Driven Supernovae and Their Nucleosynthesis" SciDAC project. The program is currently configured to run a simulation of the Sedov-Taylor blast wave solution in 2D spherical geometry.

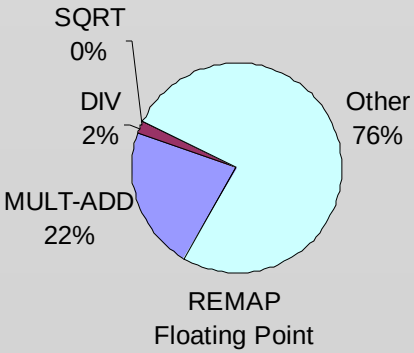


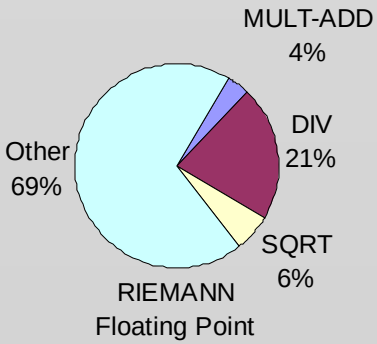
Extensive performance studies have been done to collect performance measures on dominant routines, and to study the parallel speedup performance and the communication efficiency of the full code.

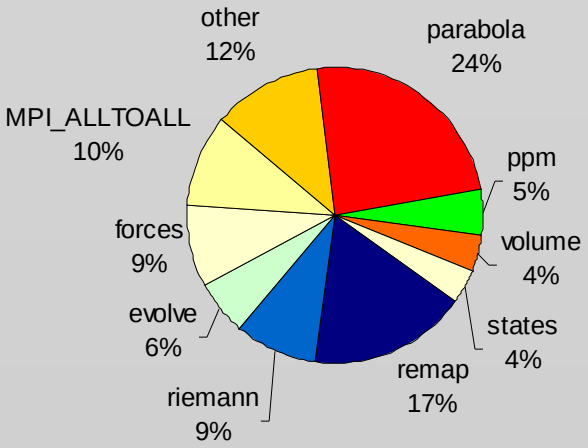
Using PAPI and SvPablo, we identified four dominant routines and two candidates for fine-tuning.

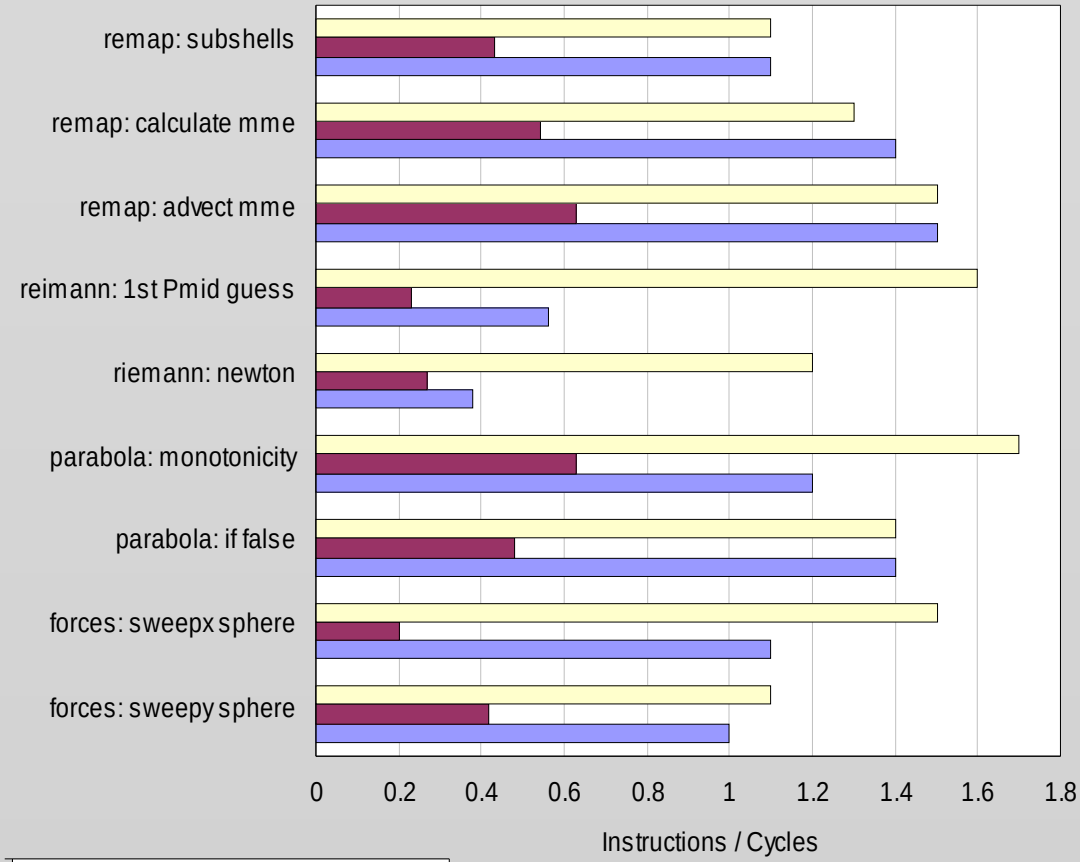
High Energy and Nuclear Physics
EVH1

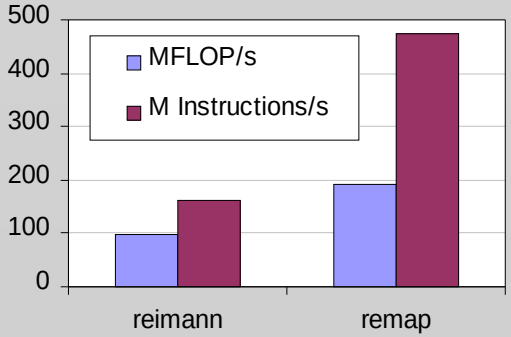
Aggregate performance measures over all tasks for a .1 simulation-second run. Collected with PAPI on an IBM SP (Nighthawk II / 375MHz).

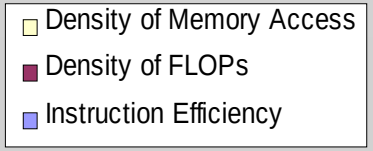




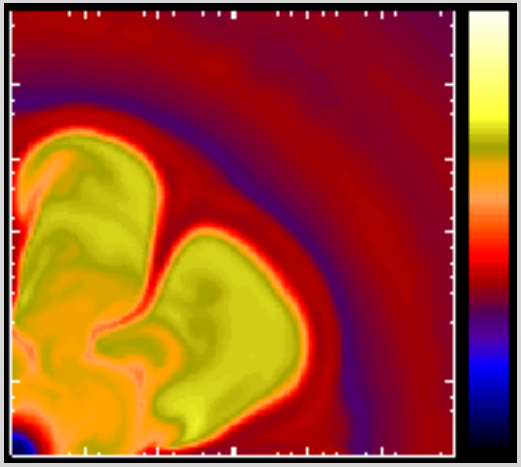








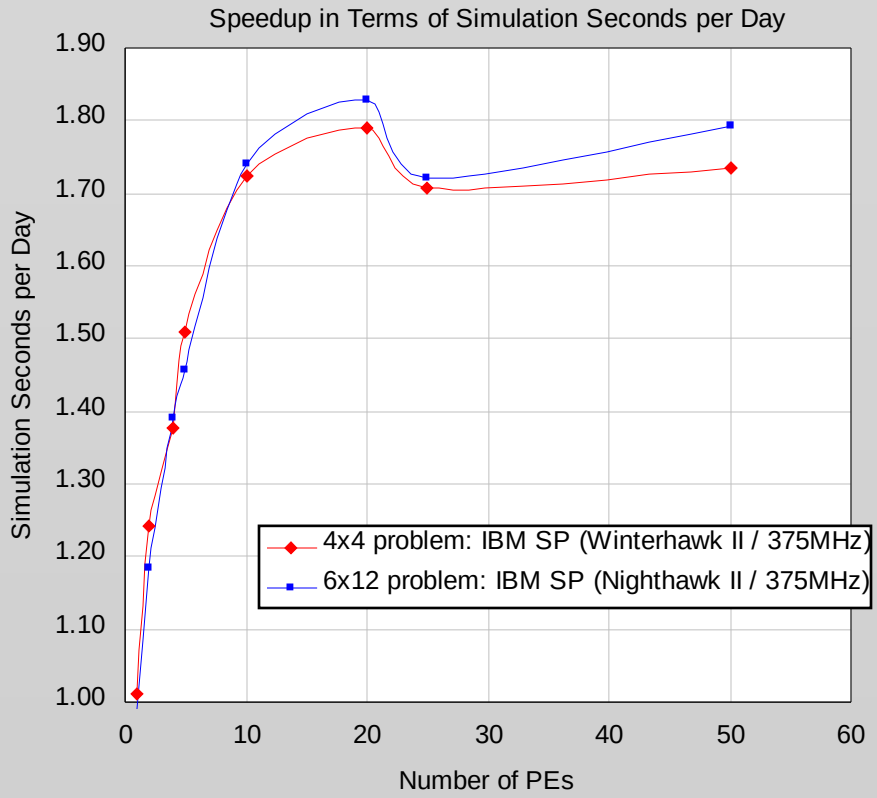
High Energy and Nuclear Physics
AGILE-BOLTZTRAN



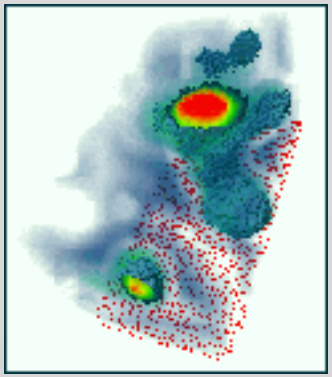
AGILE-BOLTZTRAN is a neutrino radiation hydrodynamics code from the " Terascale Simulations of Neutrino-Driven SuperNovae and Their NucleoSynthesis" SciDAC project. It is used for self-consistent simulations of core-collapse supernovae.

The code incorporates adaptive mesh hydrodynamics and discrete ordinates transport methods in spherical symmetry. Domain decomposition in radius is used, and hydrodynamics is performed redundantly on all nodes. A linear system solve is done via custom ADI preconditioner and various Krylo subspace methods (GMRES, BiCGstab, fixed point iteration).

The results of a preliminary speedup study are shown to the right.



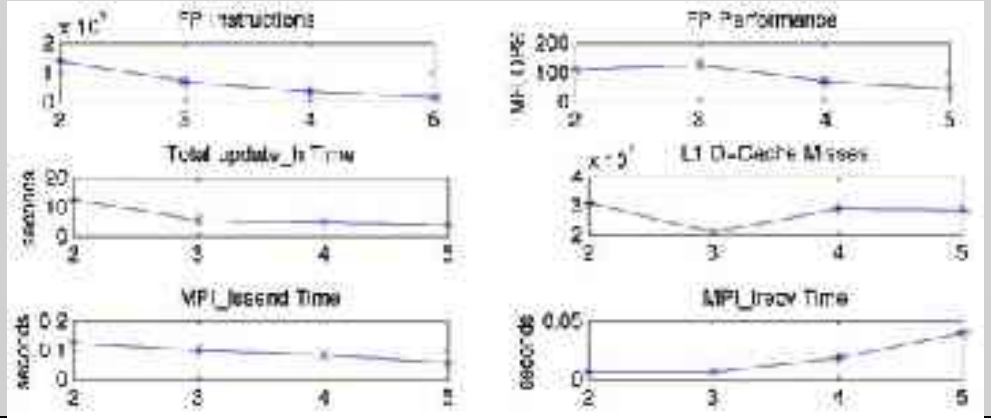
High Energy and Nuclear Physics
MILC

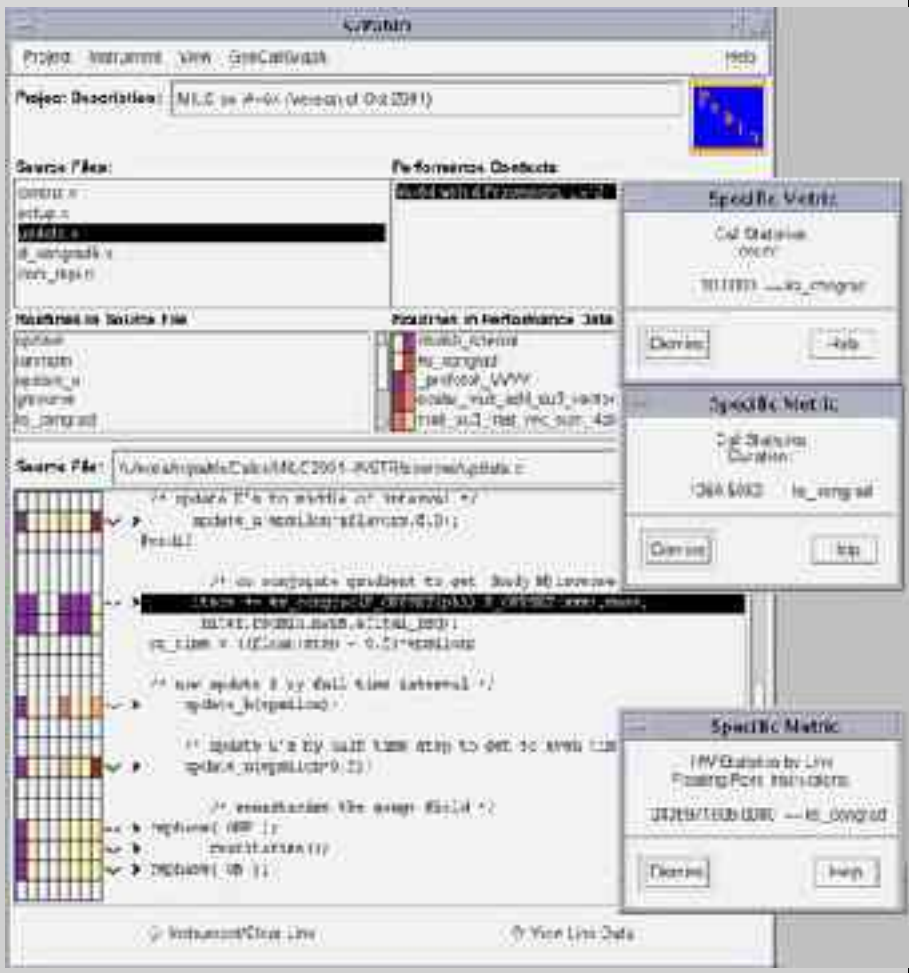


The MILC code, from the National Infrastructure for Lattice Gauge Computing SciDAC project, is a set of codes for doing simulations of four dimensional SU(3) lattice gauge theory on MIMD parallel machines.

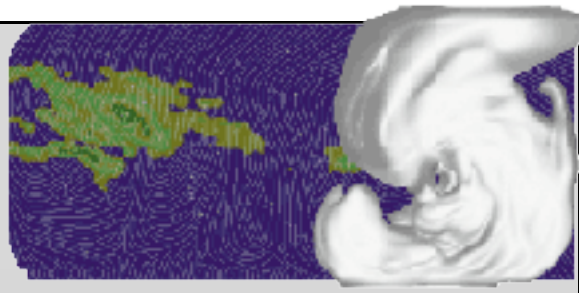
The code's computation and communication were studied using SvPablo on executions having between 4 and 32 processors (two 800MHz processors per node) of NCSA's Itanium Linux cluster.

Scalability plots are over $\log_2$(processors):

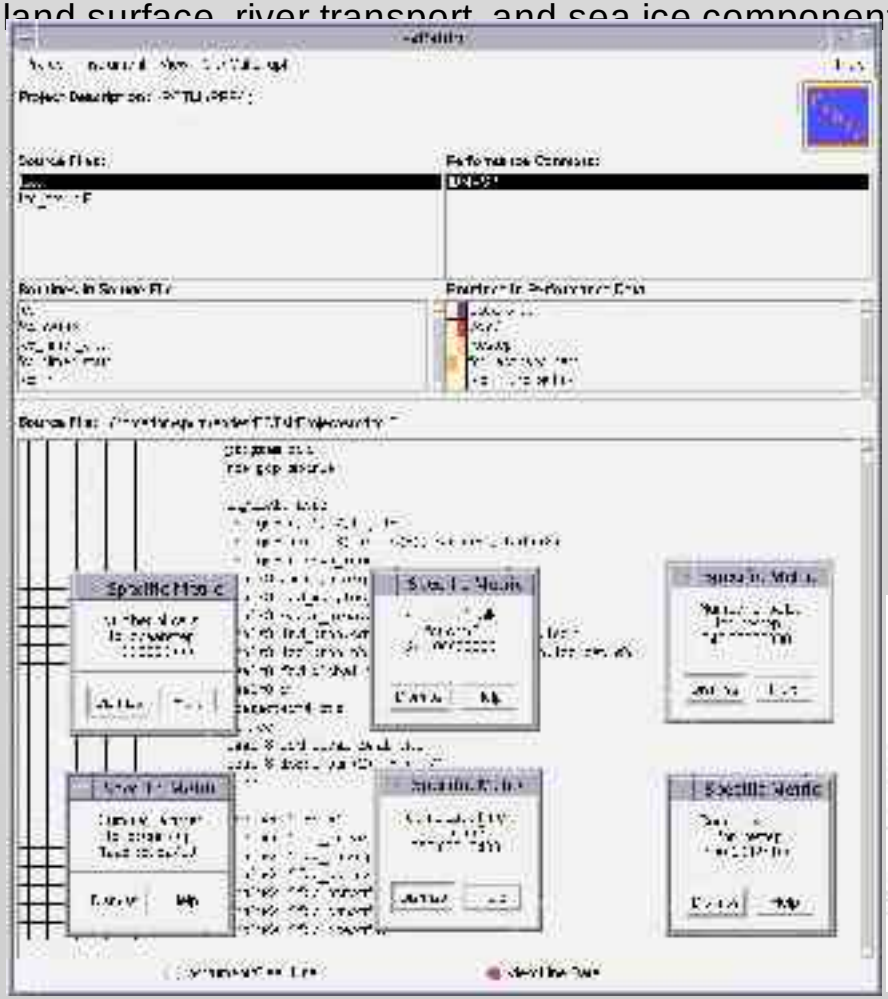




Biology and Environmental Sciences
PCTM

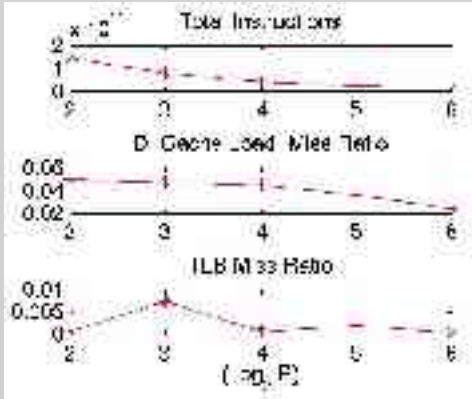


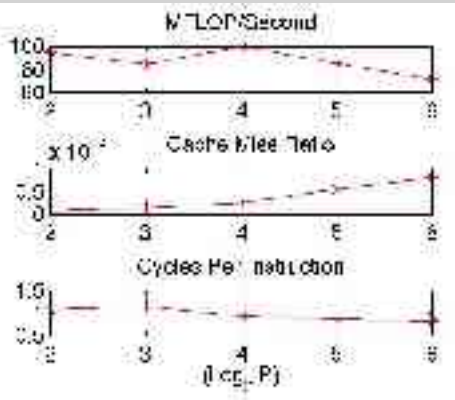
The Parallel Climate Transitional Model (PCTM), from the Climate modeling center, is the next generation of the Parallel Climate Model. It is made up of atmosphere, ocean, land surface, river transport, and sea ice component models,



and a coupler to exchange fluxes between the component models.

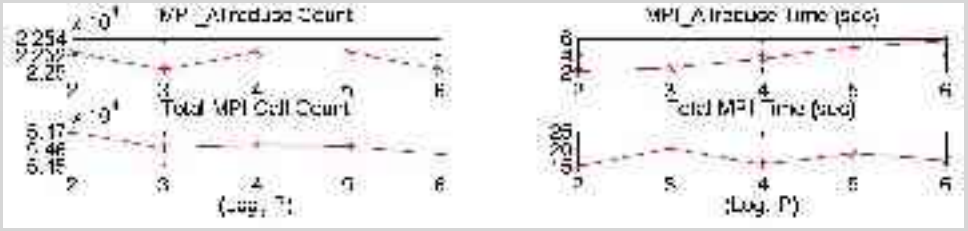
The code has been ported to IBM, Compaq, and Intel platforms. Detailed performance analysis has been done using 4 to 64 processors of an IBM SP using SvPablo.

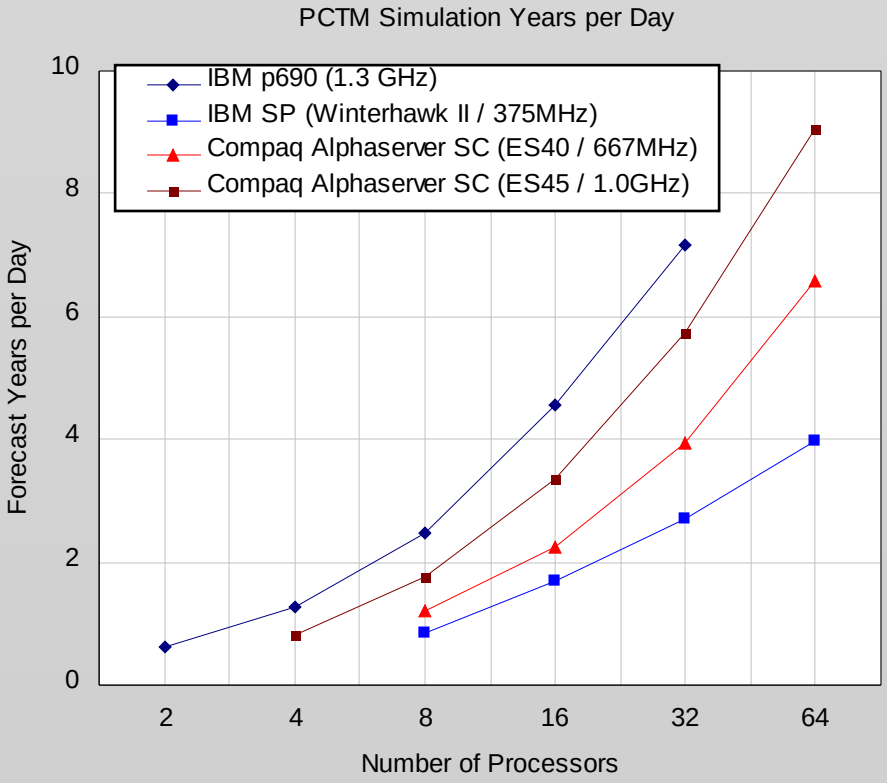


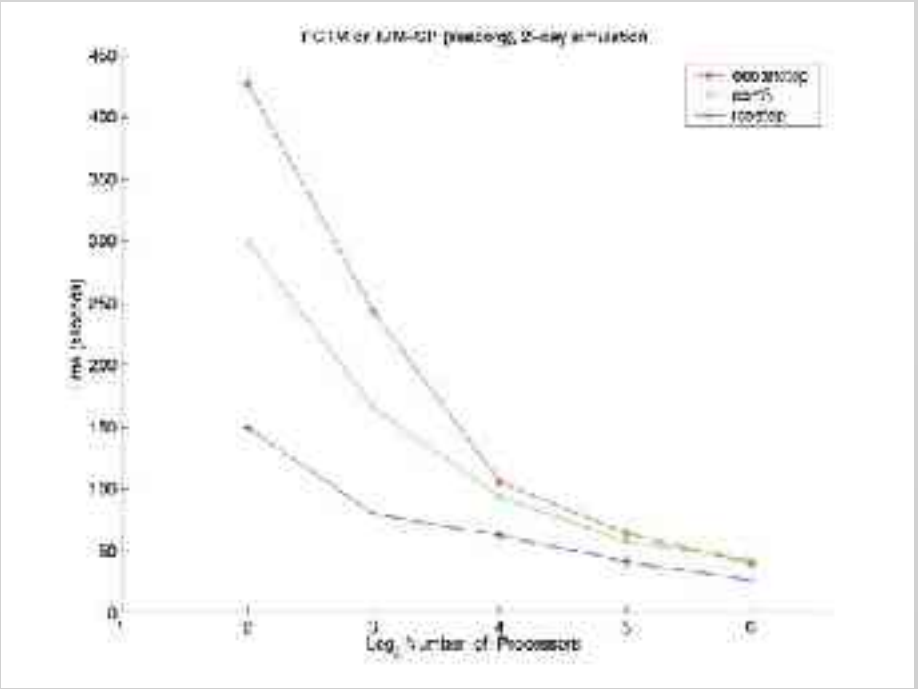


Biology and Environmental Sciences
PCTM

In other studies with PCTM, we examined the impact of different domain decompositions, as well as using fewer MPI processes per SMP node. These studies indicated a strong performance dependence on message-passing performance, where using 64 MPI processes on 32 4-way SMP nodes (leaving 64 processors idle) was 30% faster than when running on 16 SMP nodes.

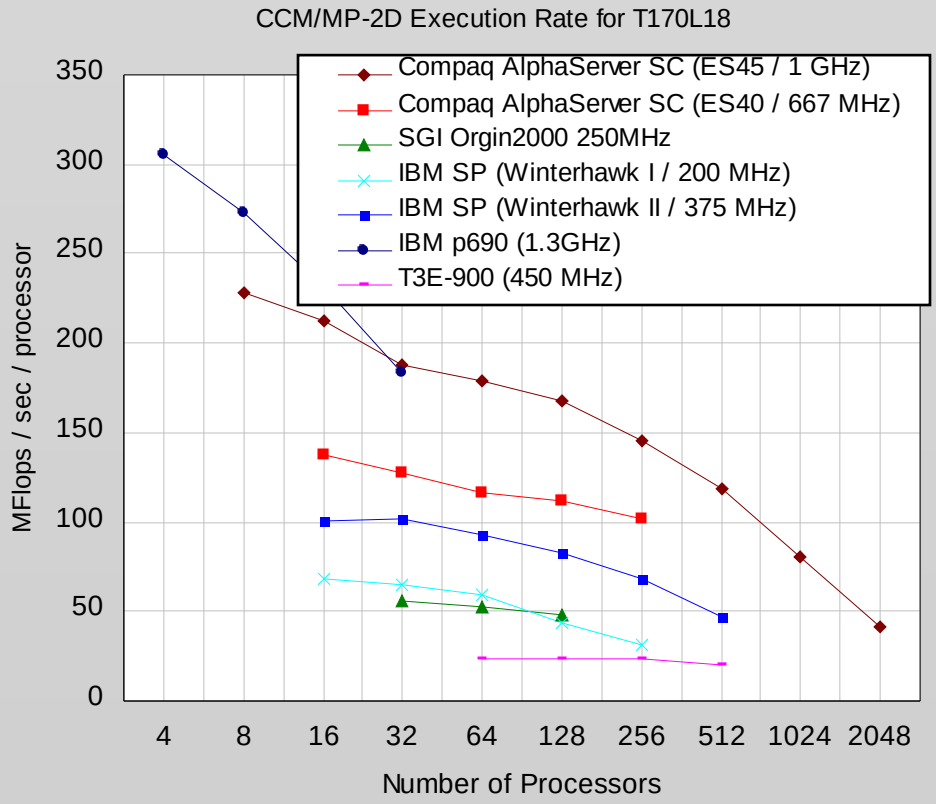


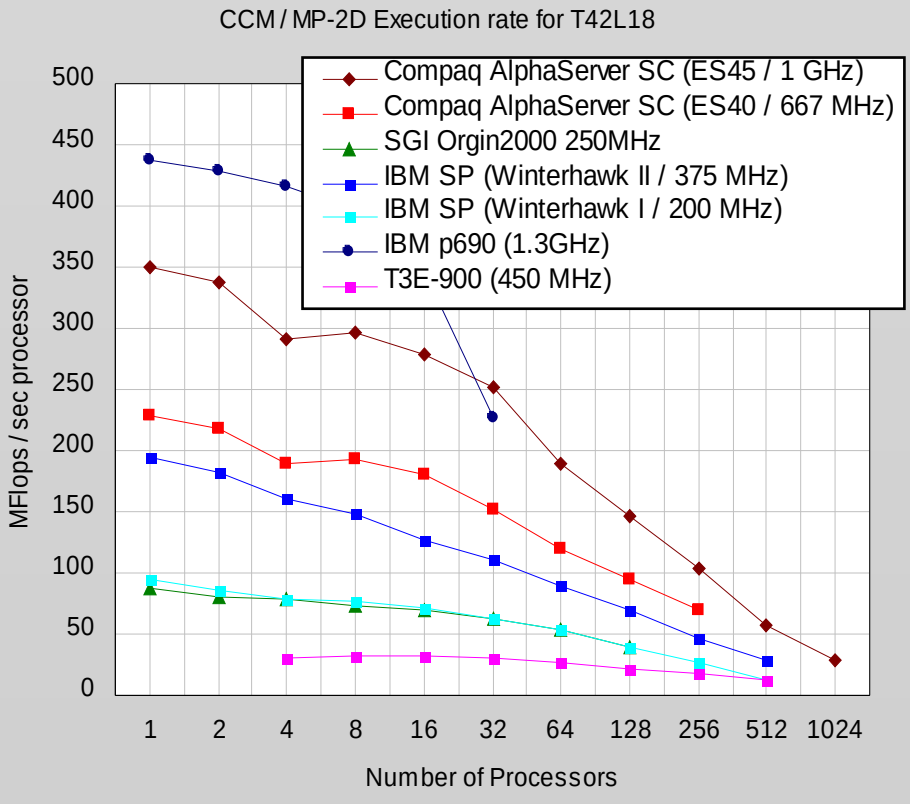




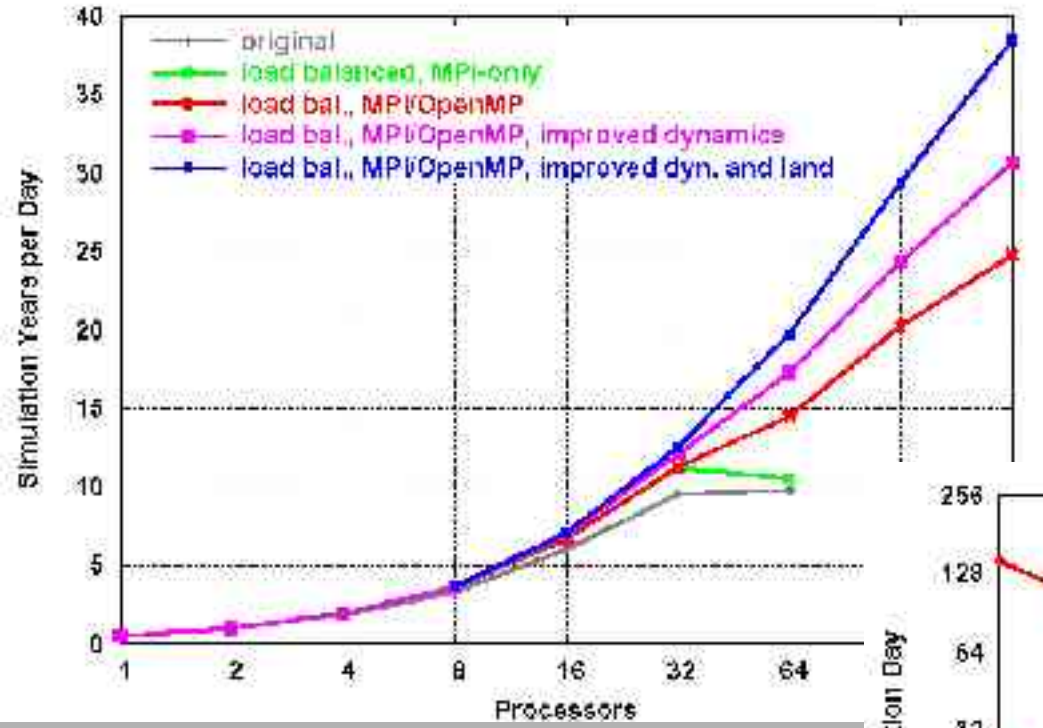
Biology and Environmental Sciences
CCM/MP-2D

CCM/MP-2D is the massively parallel implementation of version 3.6.6 of the Community Climate Model (CCM). It was developed originally to determine how best to parallelize the CCM, and the results from this research are being used in the parallelization of the Community Atmoshperic Model (CAM). CCM/MP-2D is currently used for benchmarking parallel systems. CCM/MP-2D benchmark data for some of our systems is shown below.

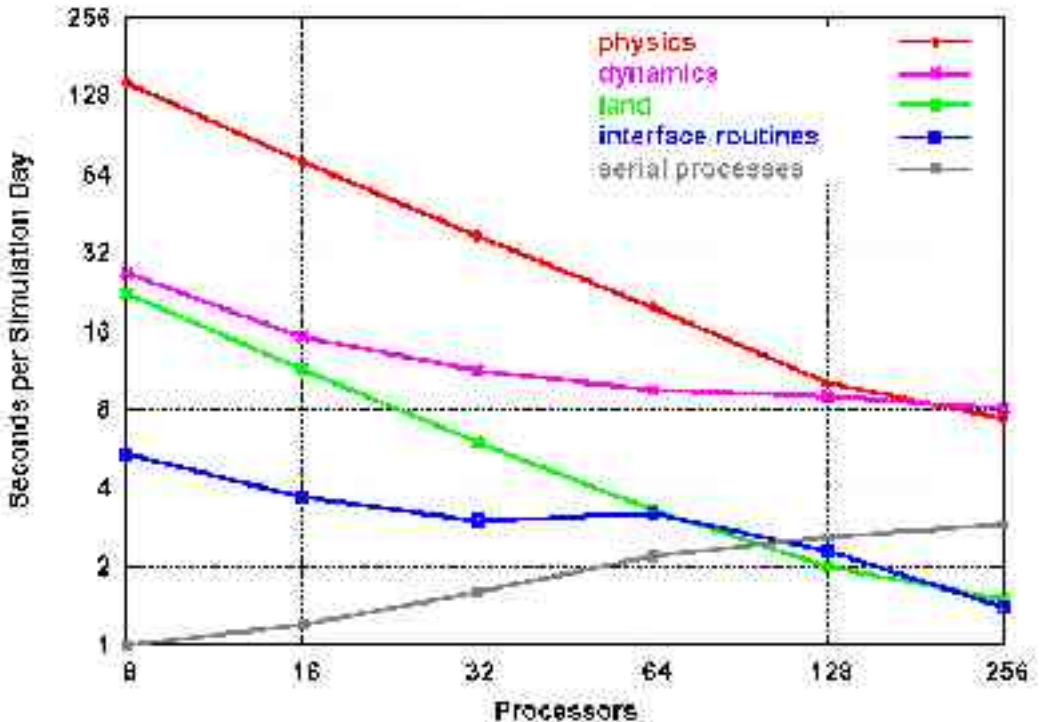




Biology and Environmental Sciences
CAM

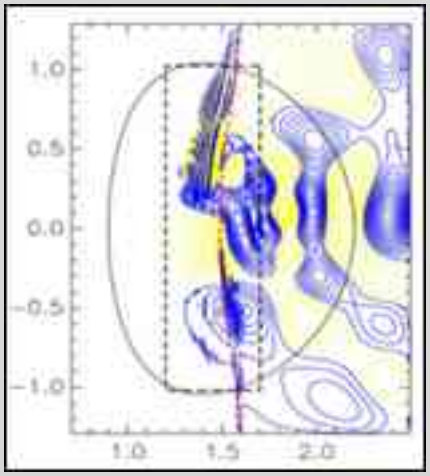


CAM performance measurements on IBM p690 cluster (and other platforms) were used to direct development process. Graph shows performance improvement from performance tuning and recent code modifications.



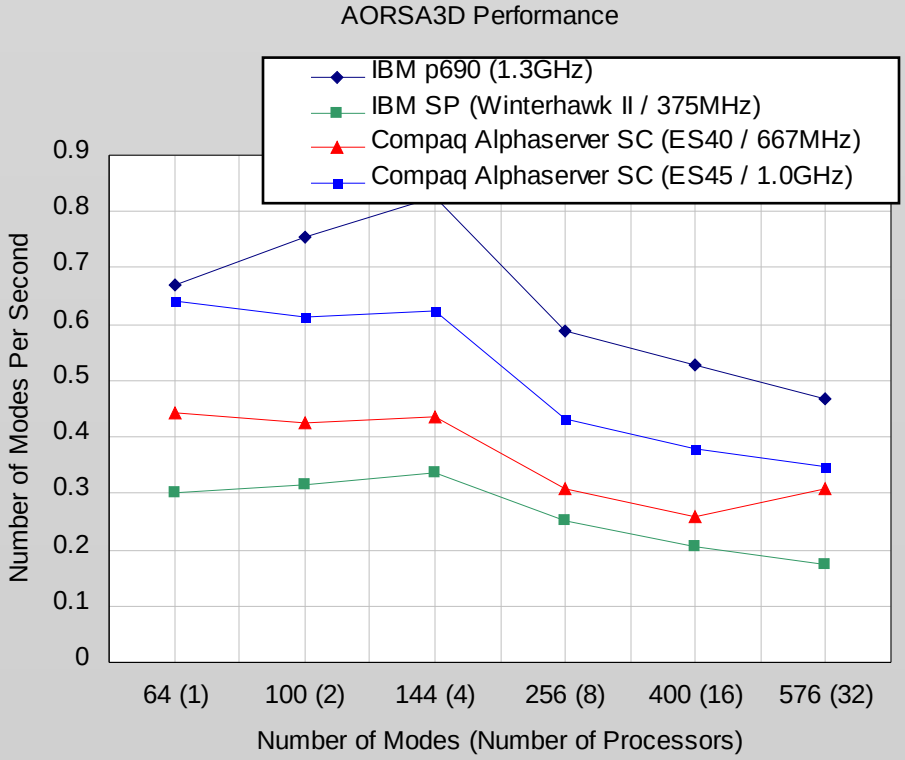
Profile of current version of CAM indicates that improving the serial performance of the physics is the most important optimization for small numbers of processors, and introducing a 2D decomposition of the dynamics (to improve scalability) is the most important optimization for large numbers of processors.

Fusion Sciences
AORSA3D



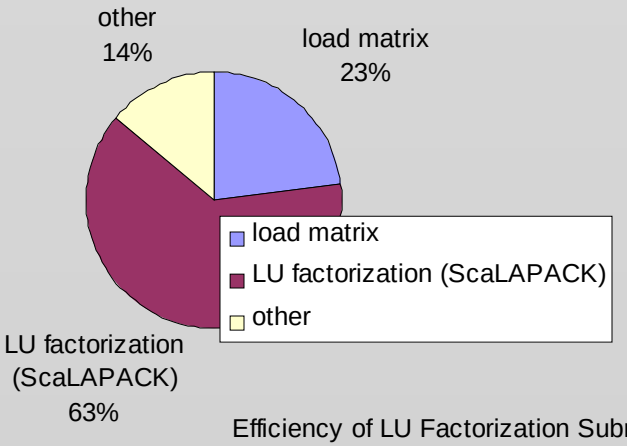
All Orders Spectral Algorithm 3D (AORSA3D) represents an important kernel in the "Numerical Computation of Wave-Plasma Interactions in Multi-dimensional Systems" SciDAC project. The code solves for the wave electric field and heating in a 3-D stellerator plasma heated by radio frequency waves. It is an MPI code that uses SCALAPACK to solve linear systems arising from the spectral discretization.

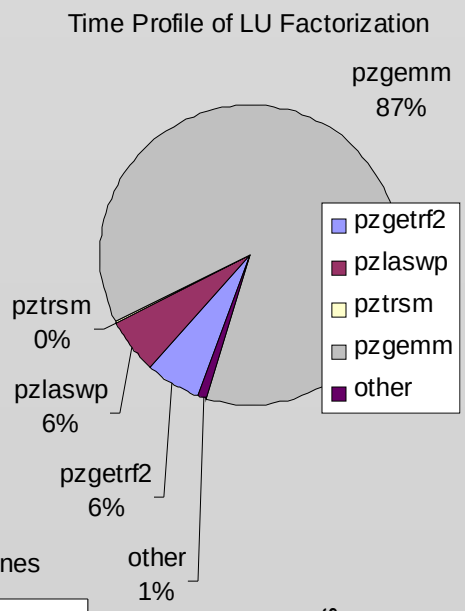
The code employs ScaLAPACK, which is used to solve a set of dense linear equations. The scaling of the code is determined by problem size and the number of rows and columns used in the block cyclic decomposition (used in ScaLAPACK to distribute the matrix across the processors). It is advantageous to pick block sizes that match as closely as possible to cache memory block sizes used in LAPACK.

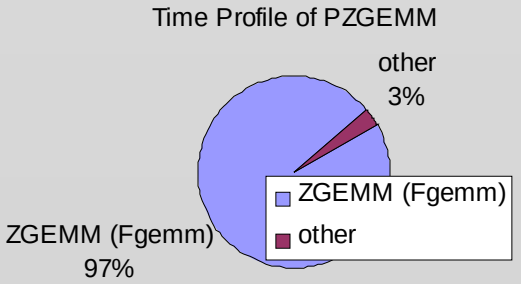


Fusion Sciences
AORSA3D

AORSA3D was ported and benchmarked on IBM and Compaq platforms. A detailed performance analysis has begun using SvPablo and PAPI. The results below are for a 400 Fourier mode run on 16 processors and 1 node of an IBM SP (Nighthawk II / 375MHz).

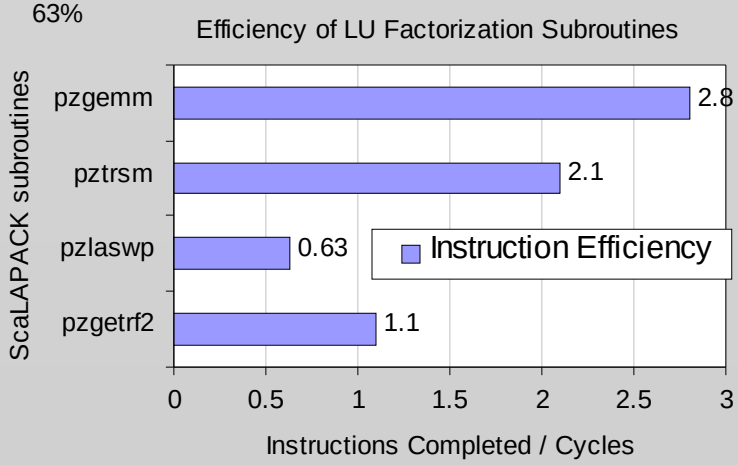


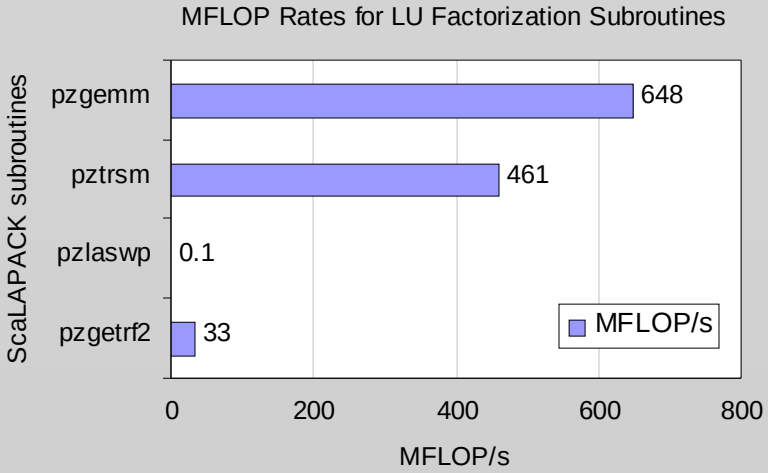




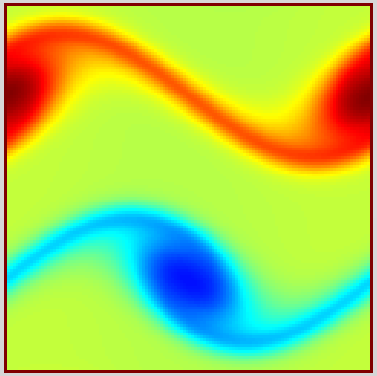
Performance for ZGemm

Denisty of Mem Access	1
Denisty of FLOPs	1.8
MFLOP/s	664
L1 cache hit rate	0.98
L2 cache hit rate	0.96
TLB misses	285034



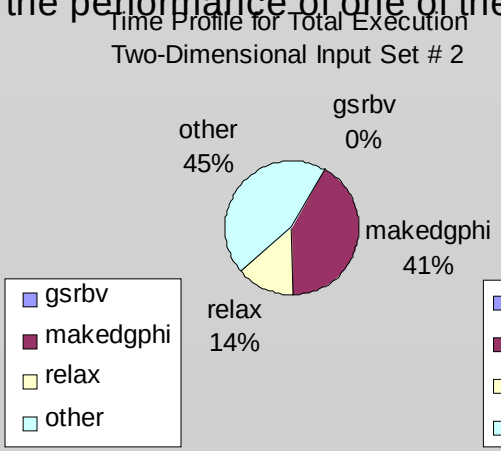


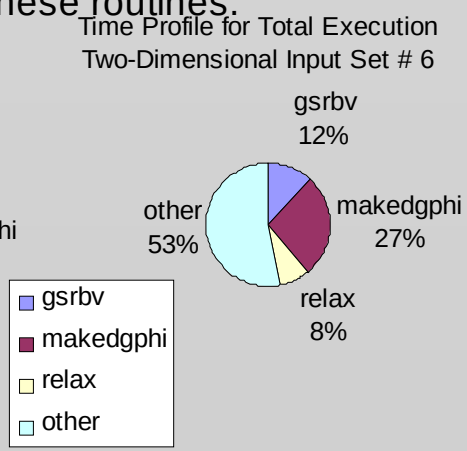
Advanced Scientific Computing
pVarDen

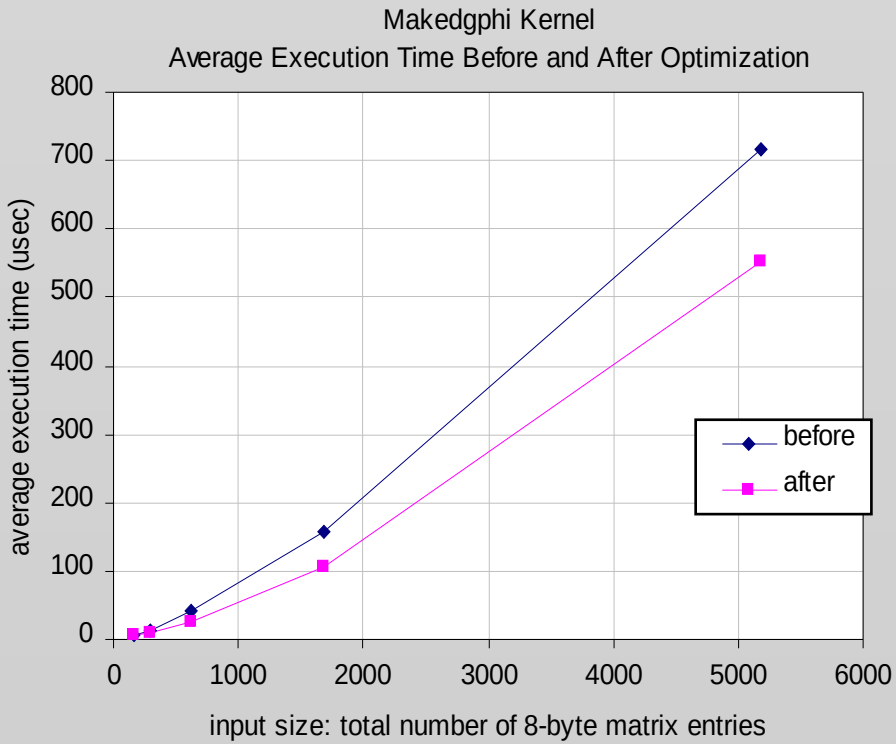


pVarden, or " Parallel VarDen" is an application code from LBNL CCSE for simulating the Variable Density Navier-Stokes equations. It represents an important kernel to the Applied PDEs ISIC. The algorithm used in pVarden is a single grid version of the variable density projection method implemented in CCSE' s IAMR application.

A detailed analysis of pVarden with PAPI has revealed 3-4 dominant routines which vary with the input data set. Some preliminary optimization work has been done to improve the performance of one of these routines.







Collaboration Summary

PERC is highly appreciated by those projects that we have the manpower to work with directly.

PERC has made substantial contributions to the performance improvement of a number of SciDAC application codes.

PERC research is being driven by and validated in SciDAC application codes.

PTOOLS and SC02 tutorials were well-attended and effective.

Demand for PERC expertise is higher than we can satisfy.

PERC Successes

PERC has brought together:

- Most major researchers in performance analysis
- A new set of ideas for measurement and modeling

It has also illuminated performance challenges

- critical SciDAC applications
- parallel architectures and system software

We’ re making great progress!

References

Performance tools, papers, and results can be downloaded from the PERC web site http://perc.nersc.gov.

Tutorials are offered describing both evaluation and optimization methodologies, and how to use the performance tools.

Presentations on PERC research are made at national meetings and at SciDAC computational science project meetings.

References

Performance tools, papers, and results can be downloaded from the PERC web site http://perc.nersc.gov.

Tutorials are offered describing both evaluation and optimization methodologies, and how to use the performance tools.

Presentations on PERC research are made at national meetings and at SciDAC computational science project meetings.

Don’ t hesitate to contact us.

Thank You.