Scientific Discovery through Advanced Computation

Performance Engineering Research Institute (PERI)

Bob Lucas (USC/ISI)
David Bailey (LBNL)

Outline

Organization of PERI

Performance modeling and prediction

Automated performance tuning

Application engagement

SciDAC

Scientific Discovery through Advanced Computation

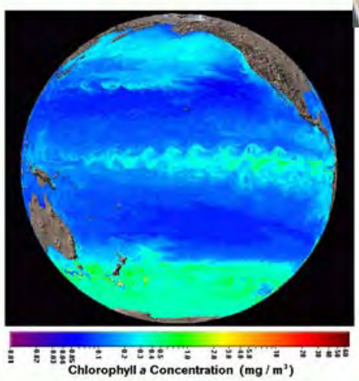
DOE Office of Science's path to petascale computational science

Maximizing performance is getting more difficult:

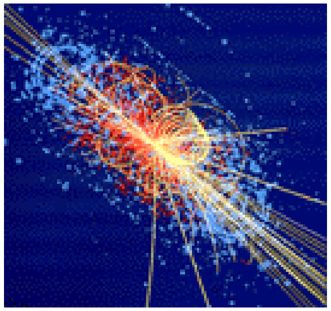
- Systems are more complicated
  - O(100K) multi-core CPUs
  - SIMD extensions
- Codes are more complicated
  - Multi-disciplinary
  - Multi-scale

IBM BlueGene at LLNL

Cray Xt3 at ORNL



POP model of El Nino



BeamBeam3D accelerator modeling

SciDAC-1 PERC

Performance Evaluation Research Center (PERC)

Initial goal was to develop performance related tools

- **Benchmarks**
- **Analysis**
- **Modeling**
- **Optimization**

Second phase refocused on SciDAC applications incl.

- **Community Climate System Model**
- **Plasma Microturbulence Project**
- **Omega3P accelerator model**

Some Lessons Learned

Performance portability is critical:

- Codes outlive machines.
- Scientists can’t publish that they migrated code.

Computational scientists are not interested in tools:

- They want experts to work with them.
- Such experts are not scalable.

Performance Engineering Research Institute

Performance modeling of applications:

- How fast do we expect to go?

Automatic tuning:

- Long term research goal.
- Remove burden from scientific programmers.

Application engagement:

- Near-term impact on SciDAC applications.

The PERI Team

Argonne National Laboratory	Lawrence Berkeley National Laboratory	Lawrence Livermore National Laboratory	North Carolina State University	Oak Ridge National Laboratory	Portland State University	Rice University
Paul Hovland Boyana Norris	David Bailey Katherine Yelick	Bronis de Supinski Daniel Quinlan	G. Mahinthakumar	Philip Roth Jeffrey Vetter Patrick Worley (PI)	Karen Karavanic	John Mellor-Crummey

University of California–San Diego	University of Maryland	University of North Carolina	University of Oregon	University of Southern California	University of Tennessee	University of Utah
Allan Snavely	Jeffrey Hollingsworth	Rob Fowler	Allen Malony Sameer Shende	Jacqueline Chame Robert Lucas (PI)	Jack Dongarra Shirley Moore	Mary Hall

PERI Organization

- Distributed leadership
  - Overall: Bob Lucas and David Bailey
  - Modeling: Allan Snavely
  - Autotuning: first Kathy Yelick, now Mary Hall
  - Application engagement: Pat Worley
    - Tiger teams: Bronis de Supinski
- Coordination mechanisms
  - Two all-hands meetings every year.
  - Phone calls approximately every two weeks
  - Opportunistic meetings
    - Monday mornings at SC

Funding and Teamwork

- **All PERI principal investigators have other sources of funding for performance-related research.**
  - **E.g., CScADS also supports HPCToolkit**
- **Thus, this research is highly leveraged from other funding sources, including non-DOE sources such as DOD and NSF.**
- **PERI, like other large SciDAC centers and institutes, consists of 10 (soon to be 11) independent awards from DOE.**
- **To date, the separate institutions have gone out of their way to work together with colleagues at other PERI institutions, thus facilitating a remarkable level of teamwork.**
- **PERI also has a large number of connections with other SciDAC centers and institutes.**
- **Nevertheless, our resources are limited, and focusing these resources on a few key application projects is a continuing challenge.**
  - **Focused on SciDAC applications, not CS or Math centers and institutes**

Performance Modeling

Modeling is critical for automation of tuning

- **Need to know where to focus effort**

  Where are the bottlenecks?
- **Need to know when we're done**

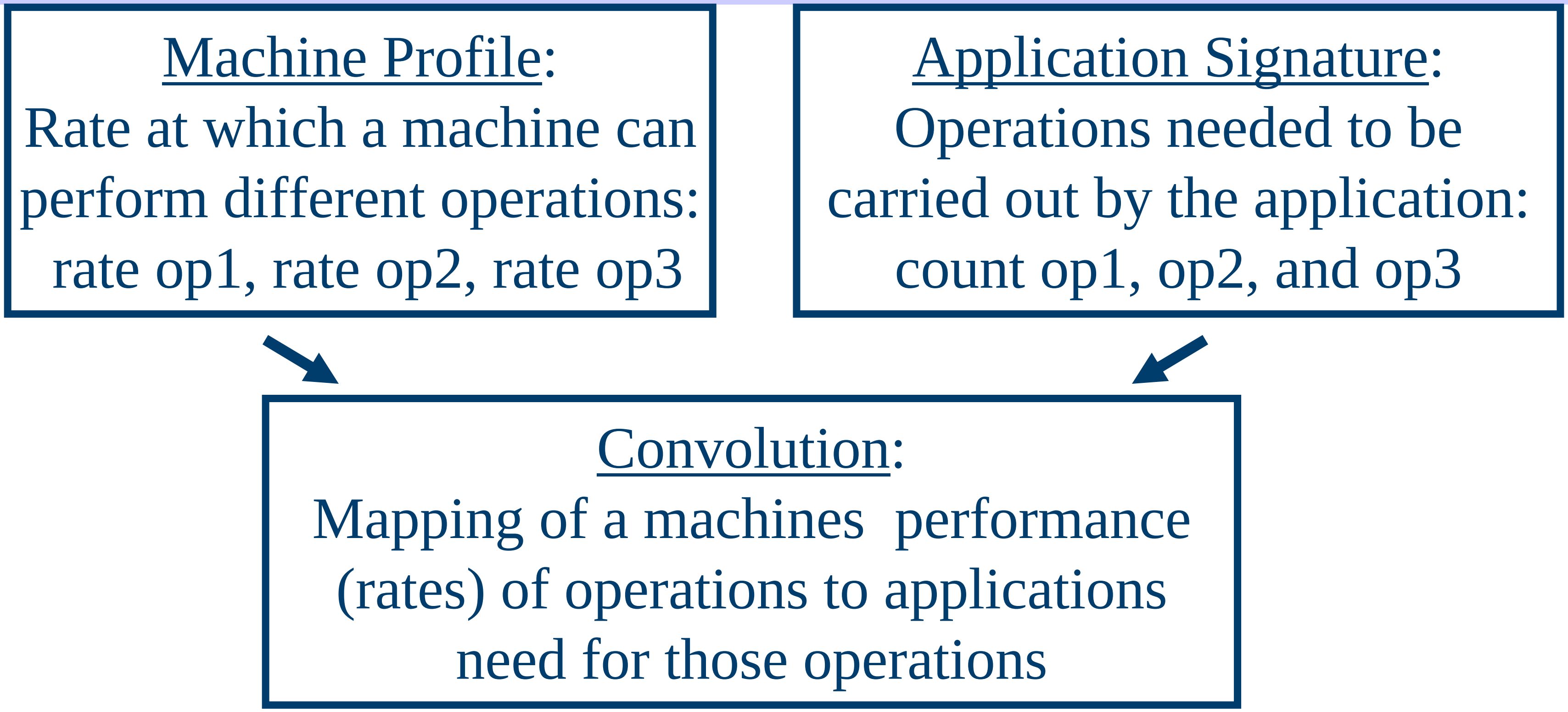
  How fast can we hope to go?

Obvious improvements:

- **Greater accuracy**
- **Reduced cost**

Modeling efforts contribute to procurements and other activities beyond PERI automatic tuning.

Convolution Model



<u>Machine Profile</u>:
Rate at which a machine can perform different operations: rate op1, rate op2, rate op3

<u>Application Signature</u>:
Operations needed to be carried out by the application: count op1, op2, and op3

<u>Convolution</u>:
Mapping of a machines performance (rates) of operations to applications need for those operations

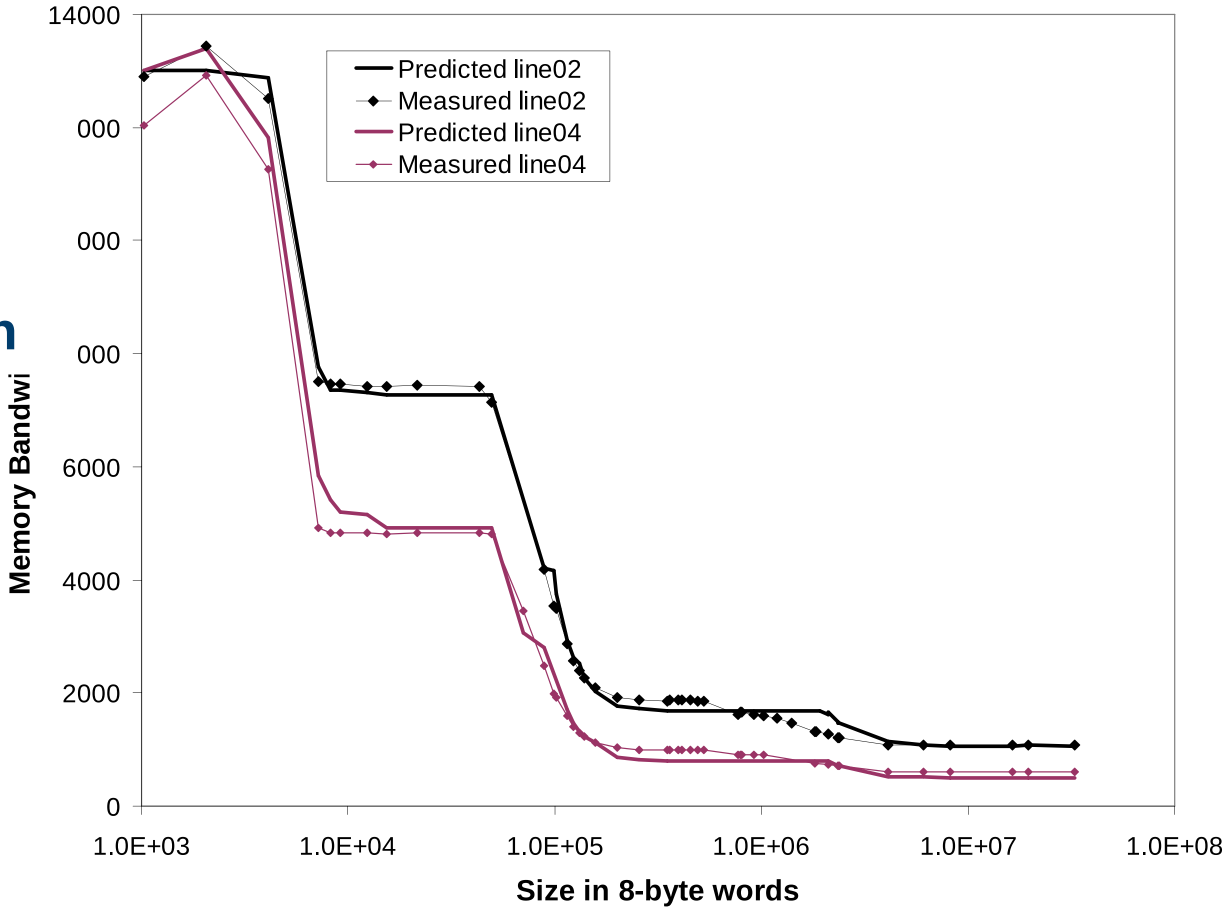
$$\text{Execution time} = \frac{\text{operation1}}{\text{rate op1}} \qquad \frac{\text{operation2}}{\text{rate op2}} \qquad \frac{\text{operation3}}{\text{rate op3}}$$

where   operator could be + or MAX depending on operation overlap

Machine Model

MultiMAPS
Memory Centric

Measured bandwidth
& Predicted bandwidth



Application Model

Consider a sparse Matrix-vector multiply ala NPB CG

```
for (p = 0, j = 0; j < n; ++ j)
  for (i = ja(j); i < ja(j + 1); ++ i)
    y(j) += A(p++) * x(ia(i));
```

A() and ia() are stride-one

x() stride is pseudo-random

Need to automatically model applications

- There aren’t enough specialists to do it by hand

PERI performance modeling is memory centric

- To first order, nothing else matters

Trace instrumented applications and record addresses

- Use statistical methods to keep this reasonable

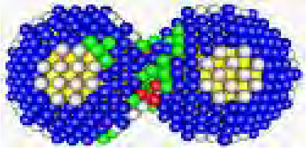
History of Success

Examples of Applications:

AVUS (CFD)

S3D (Combustion)

OVERFLOW (CFD)



LAMMPS (Materials)

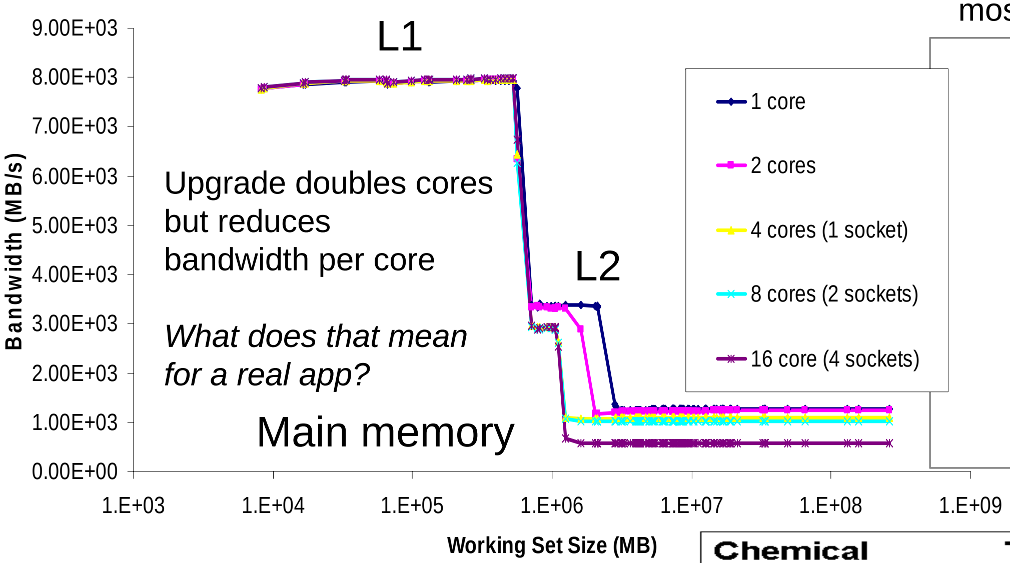
~90% accuracy exhibited on many architectures including:

Opteron, Xeon, Itanium, MIPS, and IBM Power

Recent Modeling Result

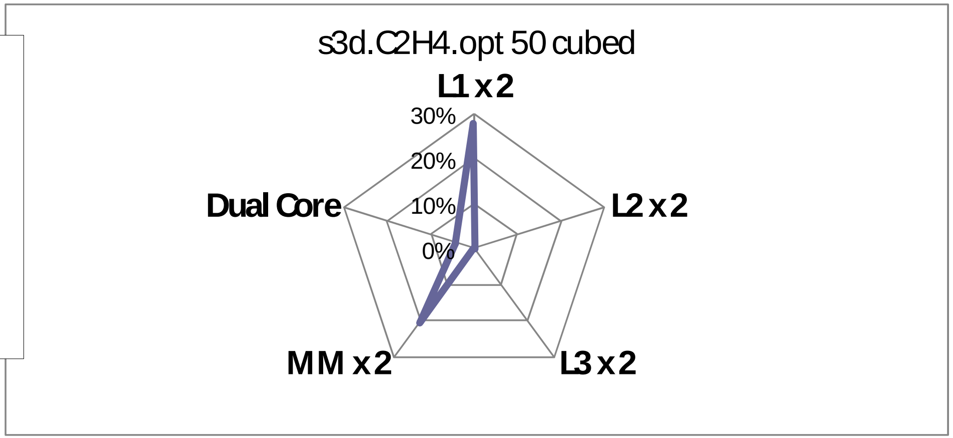
Forecast performance impact of the Jaguar quadcore upgrade on S3D. Will system run as expected? Will code scale?

Abstract performance model: predicted sensitivity of S3D to memory hierarchy bandwidth doubling reveals most sensitive to L1 and main memory bandwidth



Upgrade doubles cores but reduces bandwidth per core

What does that mean for a real app?



Concrete model: plug in anticipated quadcore bandwidths above to yield performance predictions

Full system runs (weak scaling)

Chemical grid and opt	Time (µs)
H_2 50³ orig	51
C_2H_4 50³ opt	132
C_2H_4 35³ opt	133
C_2H_4 18³ opt	172

microseconds per grid point per core

Predictions within 5% of observed post upgrade.

Performance Tuning

Humans have been doing this for 50 years

Compilers have been doing it statically for 40 years

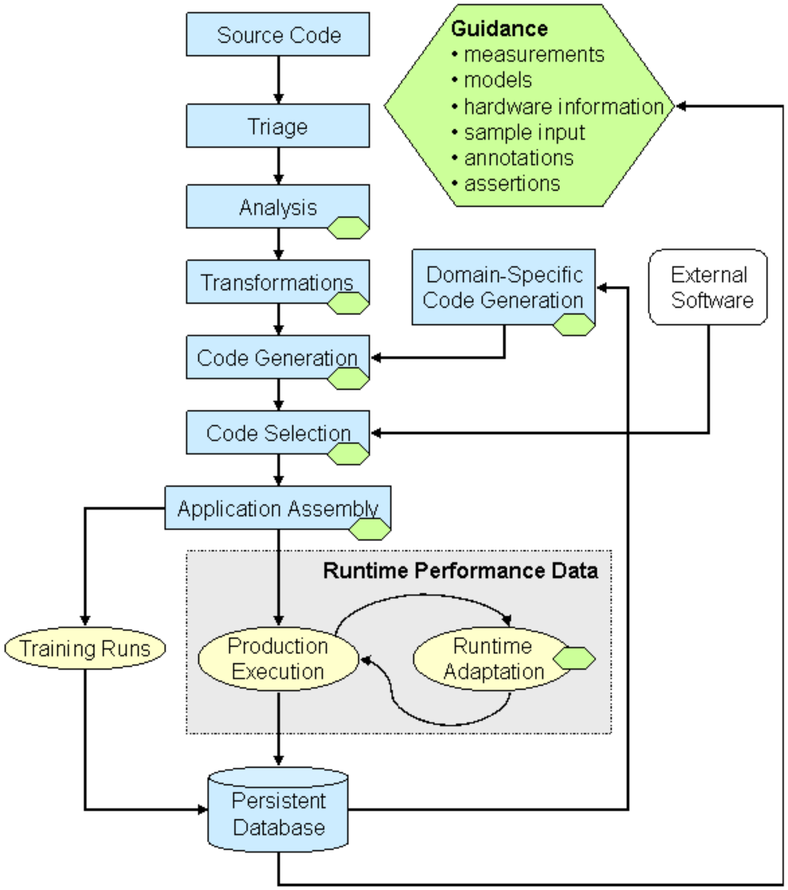
Recent self-tuning libraries:
PHIPAC, ATLAS, FFTW, SPIRAL, SPOOLES

Next logical step: automatic performance tuning of applications

Automatic Tuning

Long-term goals for PERI:

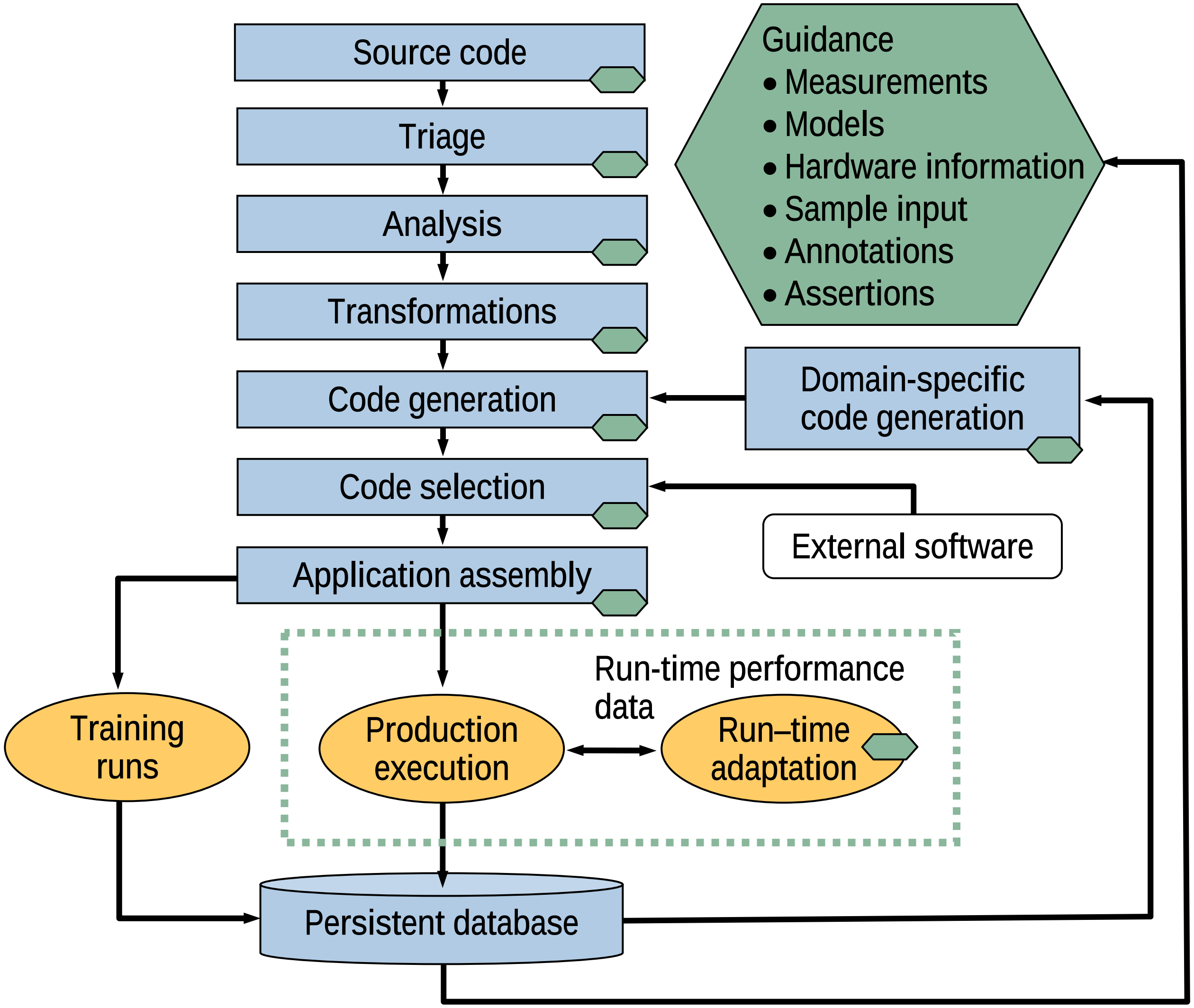
- **Automate the process of tuning software to maximize its performance**
- **Reduce the performance portability challenge facing computational scientists.**
- **Address the problem that performance experts are in short supply**
- **Build upon forty years of human experience and recent success with linear algebra libraries**



PERI automatic tuning framework

Automatic Tuning Flowchart

1: Triage	Where to focus effort
2: Semantic analysis	Traditional compiler analysis
3: Transformation	Code restructuring
4: Code generation	Domain- specific code
5: Code selection	Modeling and empirical search
6: Assembly	Choose the best components
7: Training runs	Performance data for feedback
8: Run–time adaptation	Optimize long-running jobs



Source code

Triage

Analysis

Transformations

Code generation

Code selection

Application assembly

Guidance
- Measurements
- Models
- Hardware information
- Sample input
- Annotations
- Assertions

Domain-specific code generation

External software

Training runs

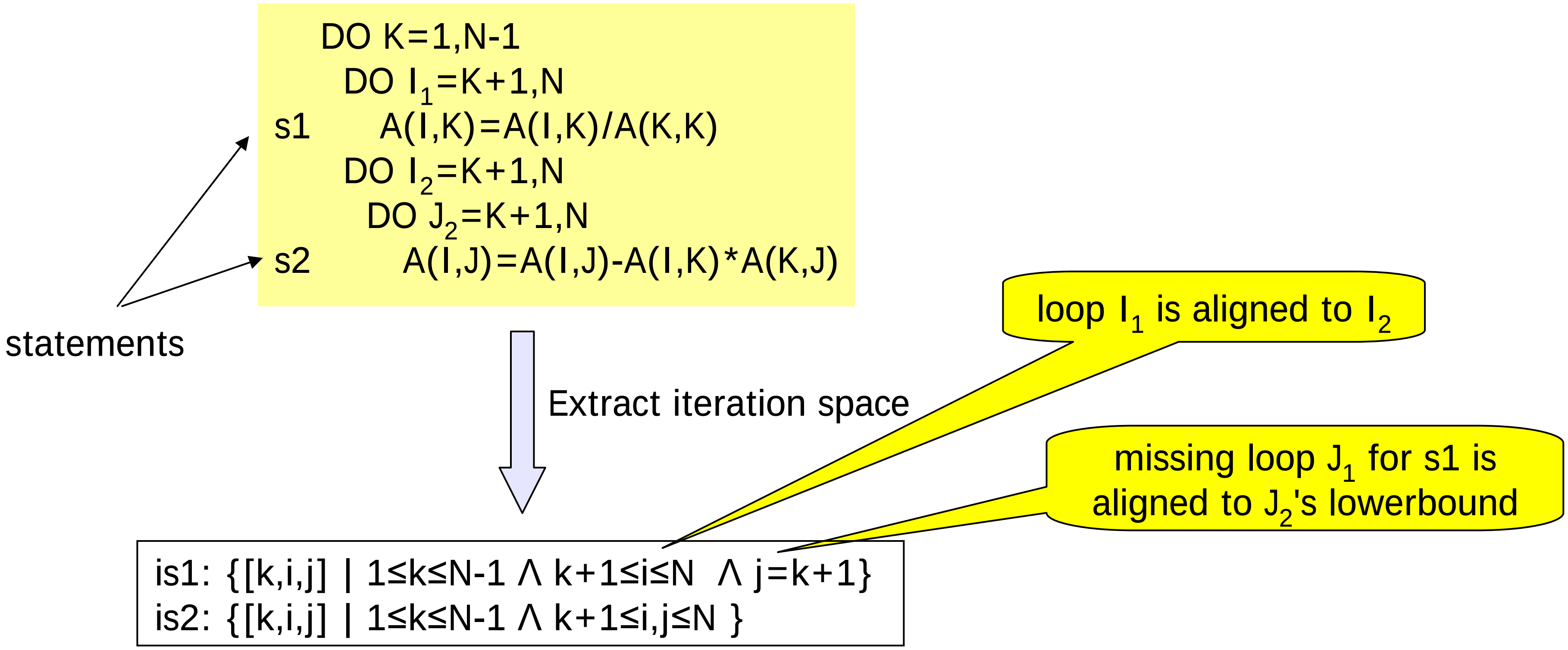
Production execution

Run-time performance data

Run–time adaptation

Persistent database

LU Decomposition Straightforward Code



```
      DO K=1,N-1
        DO $I_1$=K+1,N
s1        A(I,K)=A(I,K)/A(K,K)
        DO $I_2$=K+1,N
          DO $J_2$=K+1,N
s2          A(I,J)=A(I,J)-A(I,K)*A(K,J)
```

statements

Extract iteration space

loop I_1 is aligned to I_2

missing loop J_1 for s1 is aligned to J_2's lowerbound

is1: $\{[k,i,j] \mid 1 \le k \le N-1 \wedge k+1 \le i \le N \wedge j=k+1\}$

is2: $\{[k,i,j] \mid 1 \le k \le N-1 \wedge k+1 \le i,j \le N \}$

- All statements are aligned in a single iteration space
- Alignment is valid if data dependences do not violate original code semantics

From Chun Chen's thesis defense

LU Example: Loop Transformations

Existing iteration space:
is1: {[k,i,j] | 1≤k≤N-1 ∧ k+1≤i≤N ∧ j=k+1}
is2: {[k,i,j] | 1≤k≤N-1 ∧ k+1≤i,j≤N }

Mapping relations:
t1: {[k,i,j]->[0,k,0,i,0,j,0]}
t2: {[k,i,j]->[0,k,0,i,1,j,0]}

constant loops for lexicographical order of different loops at the same loop level

Transformed iteration space:
is1: {[0,k,0,i,0,j,0] | 1≤k≤N-1 ∧ k+1≤i≤N ∧ j=k+1}
is2: {[0,k,0,i,1,j,0] | 1≤k≤N-1 ∧ k+1≤i,j≤N }

Omega code generation

```
DO T2=1,N-1
    DO T4=T2+1,N
        A(T4,T2)=A(T4,T2)/A(T2,T2)
         DO T6=T2+1,N
             A(T4,T6)=A(T4,T6)-A(T4,T2)*A(T2,T6)
```

Transformed Code

```
REAL*8 P1(32,32),P2(32,64),P3(32,32),P4(32,64)
OVER1=0
OVER2=0
DO T2=2,N,64
 IF (66<=T2)
  DO T4=2,T2-32,32
   DO T6=1,T4-1,32
    DO T8=T6,MIN(T4-1,T6+31)
     DO T10=T4,MIN(T2-2,T4+31)
      P1(T8-T6+1,T10-T4+1)=A(T10,T8)
    DO T8=T2,MIN(T2+63,N)
     DO T10=T6,MIN(T6+31,T4-1)
      P2(T10-T6+1,T8-T2+1)=A(T10,T8)
    DO T8=T4,MIN(T2-2,T4+31)
     OVER1=MOD(-1+N,4)
     DO T10=T2,MIN(N-OVER1,T2+60),4
      DO T12=T6,MIN(T6+31,T4-1)
       A(T8,T10)=A(T8,T10)-P1(T12-T6+1,T8-T4+1)*P2(T12-T6+1,T10-T2+1)
       A(T8,T10+1)=A(T8,T10+1)-P1(T12-T6+1,T8-T4+1)*P2(T12-T6+1,T10+1-T2+1)
       A(T8,T10+2)=A(T8,T10+2)-P1(T12-T6+1,T8-T4+1)*P2(T12-T6+1,T10+2-T2+1)
       A(T8,T10+3)=A(T8,T10+3)-P1(T12-T6+1,T8-T4+1)*P2(T12-T6+1,T10+3-T2+1)
     DO T10=MAX(N-OVER1+1,T2),MIN(T2+63,N)
      DO T12=T6,MIN(T4-1,T6+31)
       A(T8,T10)=A(T8,T10)-P1(T12-T6+1,T8-T4+1)*P2(T12-T6+1,T10-T2+1)
   DO T6=T4+1,MIN(T4+31,T2-2)
    DO T8=T2,MIN(N,T2+63)
     DO T10=T4,T6-1
      A(T6,T8)=A(T6,T8)-A(T6,T10)*A(T10,T8)
```

TRSM

data copy

unroll by 4

unroll cleanup

Transformed Code (Cont.)

GEMM

```
IF (66<=T2)
  DO T4=1,T2-33,32
    DO T6=T2-1,N,32
      DO T8=T4,T4+31
        DO T10=T6,MIN(N,T6+31)
          P3(T8-T4+1,T10-T6+1)=A(T10,T8)
      DO T8=T2,MIN(T2+63,N)
        DO T10=T4,T4+31
          P4(T10-T4+1,T8-T2+1)=A(T10,T8)
      DO T8=T6,MIN(T6+31,N)
        OVER2=MOD(-1+N,4)
        DO T10=T2,MIN(N-OVER2,T2+60),4
          DO T12=T4,T4+31
            A(T8,T10)=A(T8,T10)-P3(T12-T4+1,T8-T6+1)*P4(T12-T4+1,T10-T2+1)
            A(T8,T10+1)=A(T8,T10+1)-P3(T12-T4+1,T8-T6+1)*P4(T12-T4+1,T10+1-T2+1)
            A(T8,T10+2)=A(T8,T10+2)-P3(T12-T4+1,T8-T6+1)*P4(T12-T4+1,T10+2-T2+1)
            A(T8,T10+3)=A(T8,T10+3)-P3(T12-T4+1,T8-T6+1)*P4(T12-T4+1,T10+3-T2+1)
        DO T10=MAX(T2,N-OVER2+1),MIN(T2+63,N)
          DO T12=T4,T4+31
            A(T8,T10)=A(T8,T10)-P3(T12-T4+1,T8-T6+1)*P4(T12-T4+1,T10-T2+1)
DO T4=T2-1,MIN(N-1,T2+62)
  DO T8=T4+1,N
    A(T8,T4)=A(T8,T4)/A(T4,T4)
  DO T6=T4+1,MIN(T2+63,N)
    DO T8=T4+1,N
      A(T8,T6)=A(T8,T6)-A(T8,T4)*A(T4,T6)
```

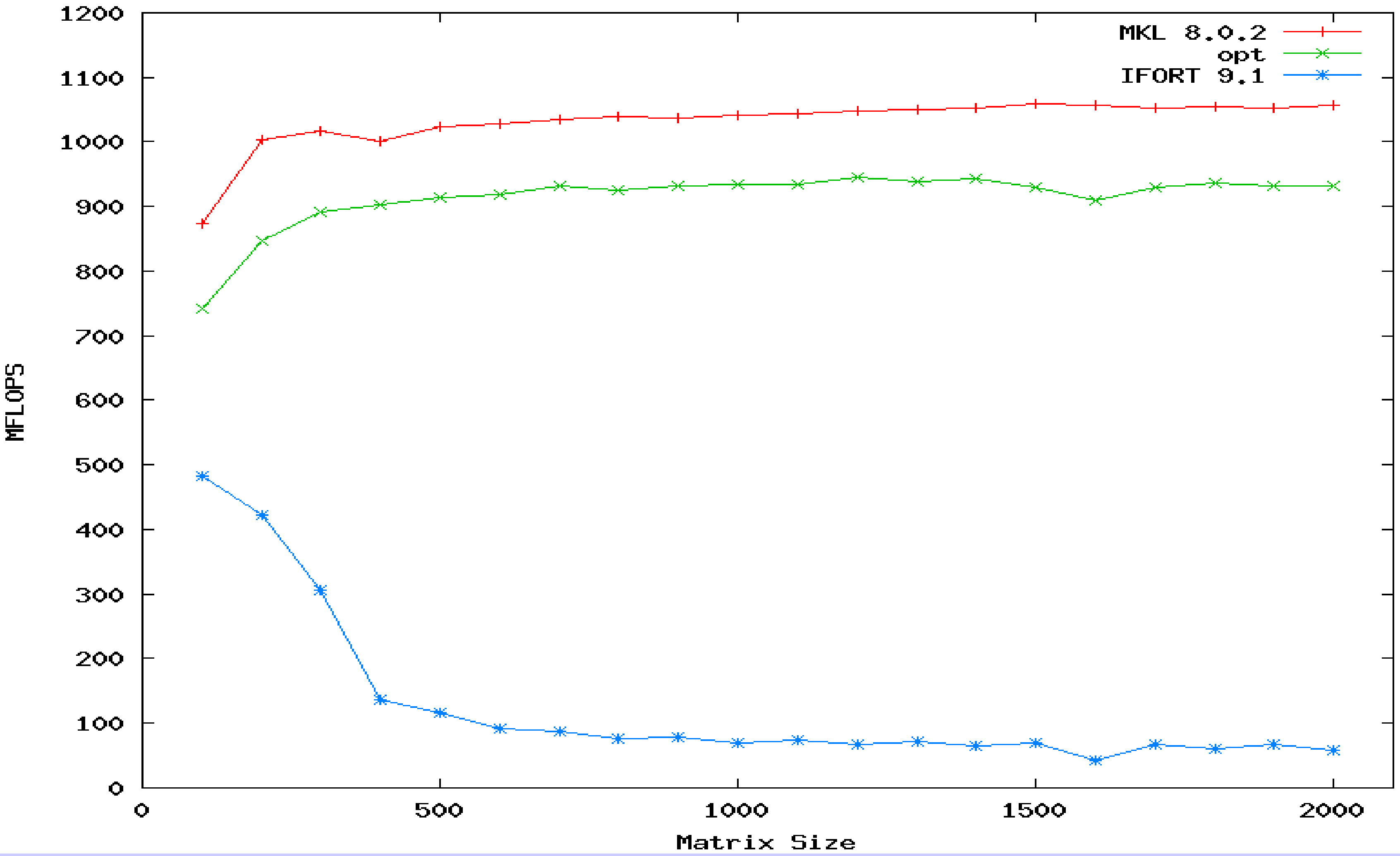
Mini-LU

data copy

unroll by 4

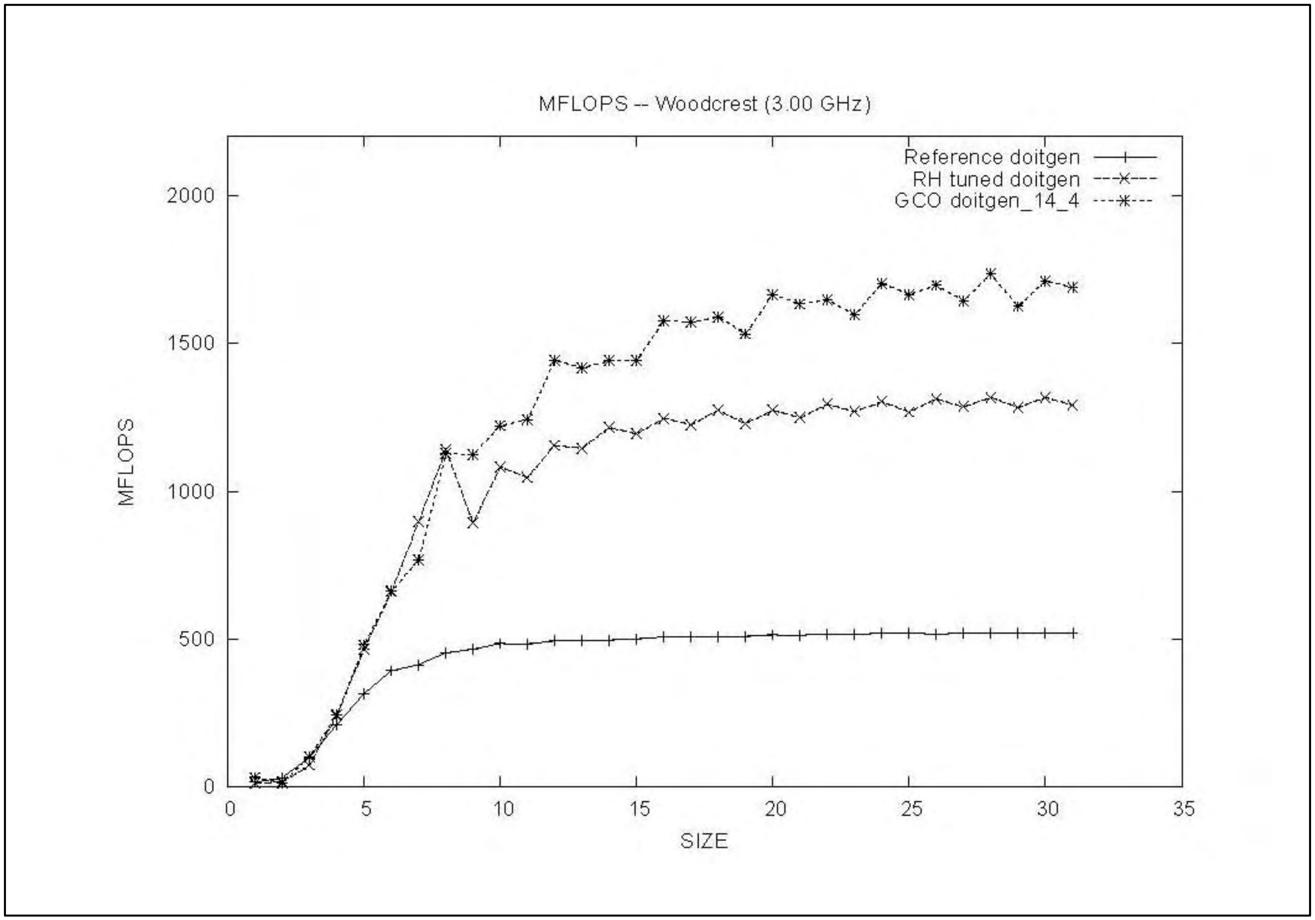
unroll cleanup

Autotuned LU Results

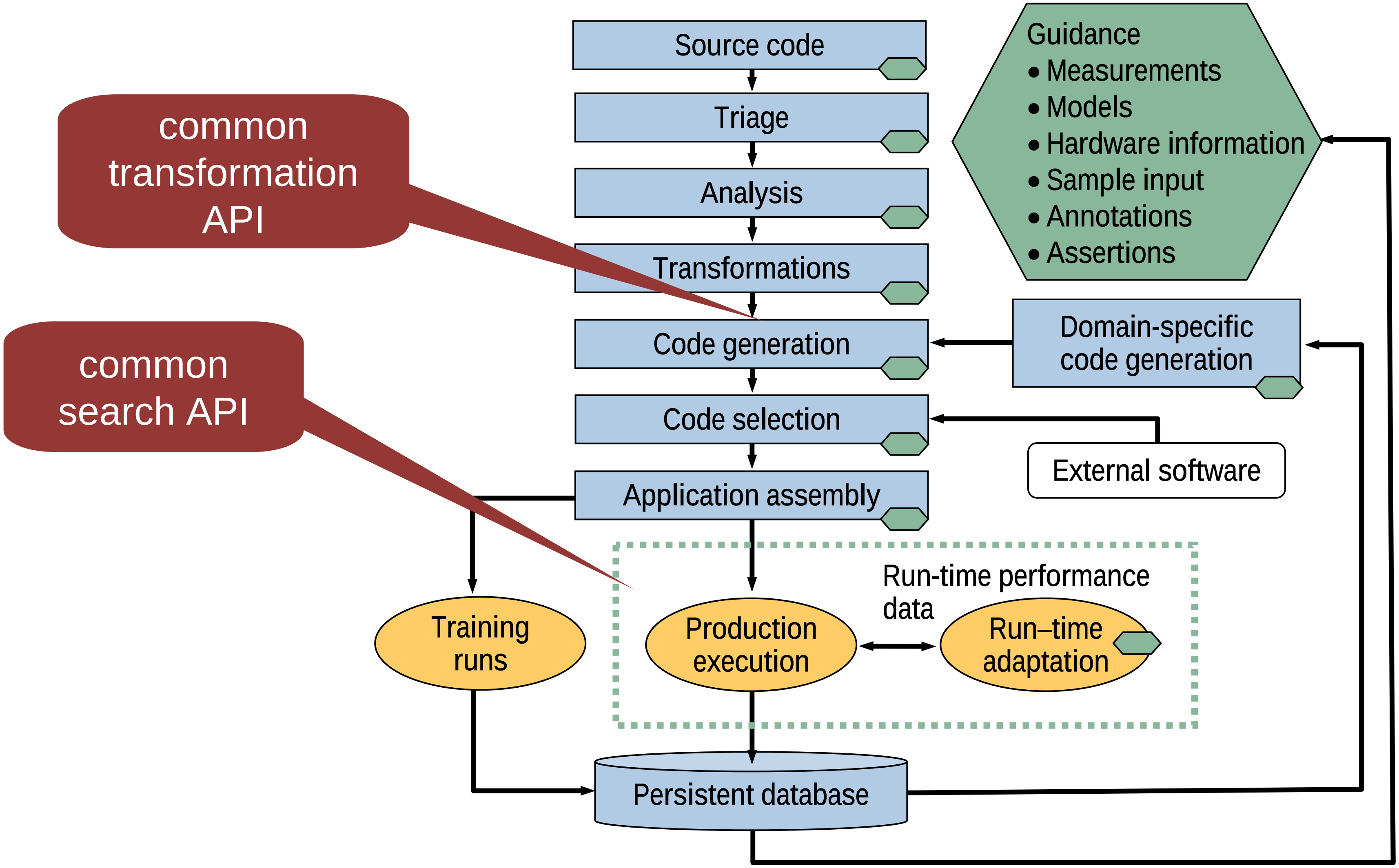


Early Results for MADNESS

Empirical optimization of Madness kernel (Moore,UTK)

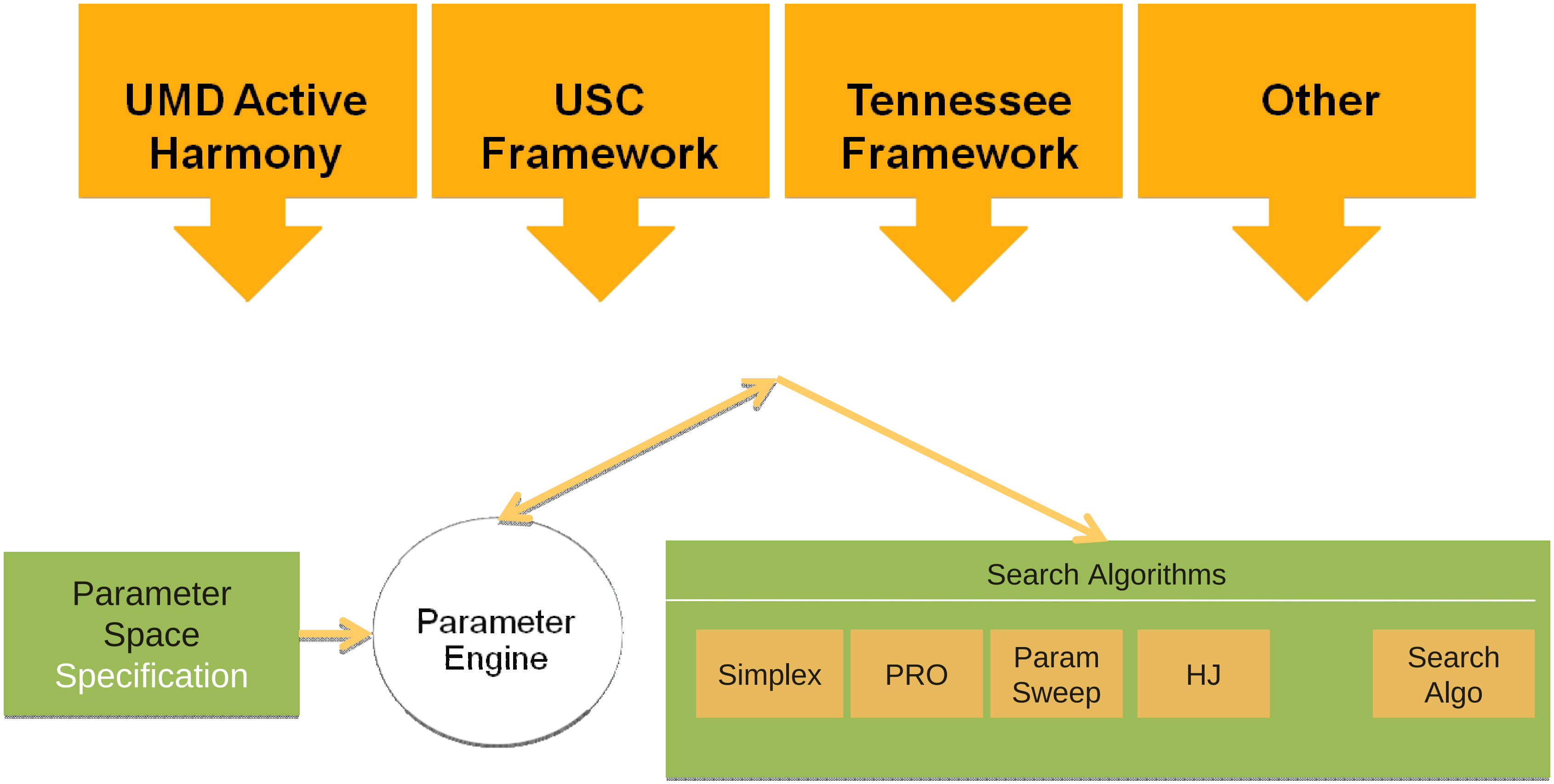


Automatic Tuning Recent Progress



PERI Search API

Search algorithms can be plugged into a generalized search framework



UMD Active Harmony

USC Framework

Tennessee Framework

Other

Parameter Space Specification

Parameter Engine

Search Algorithms

Simplex

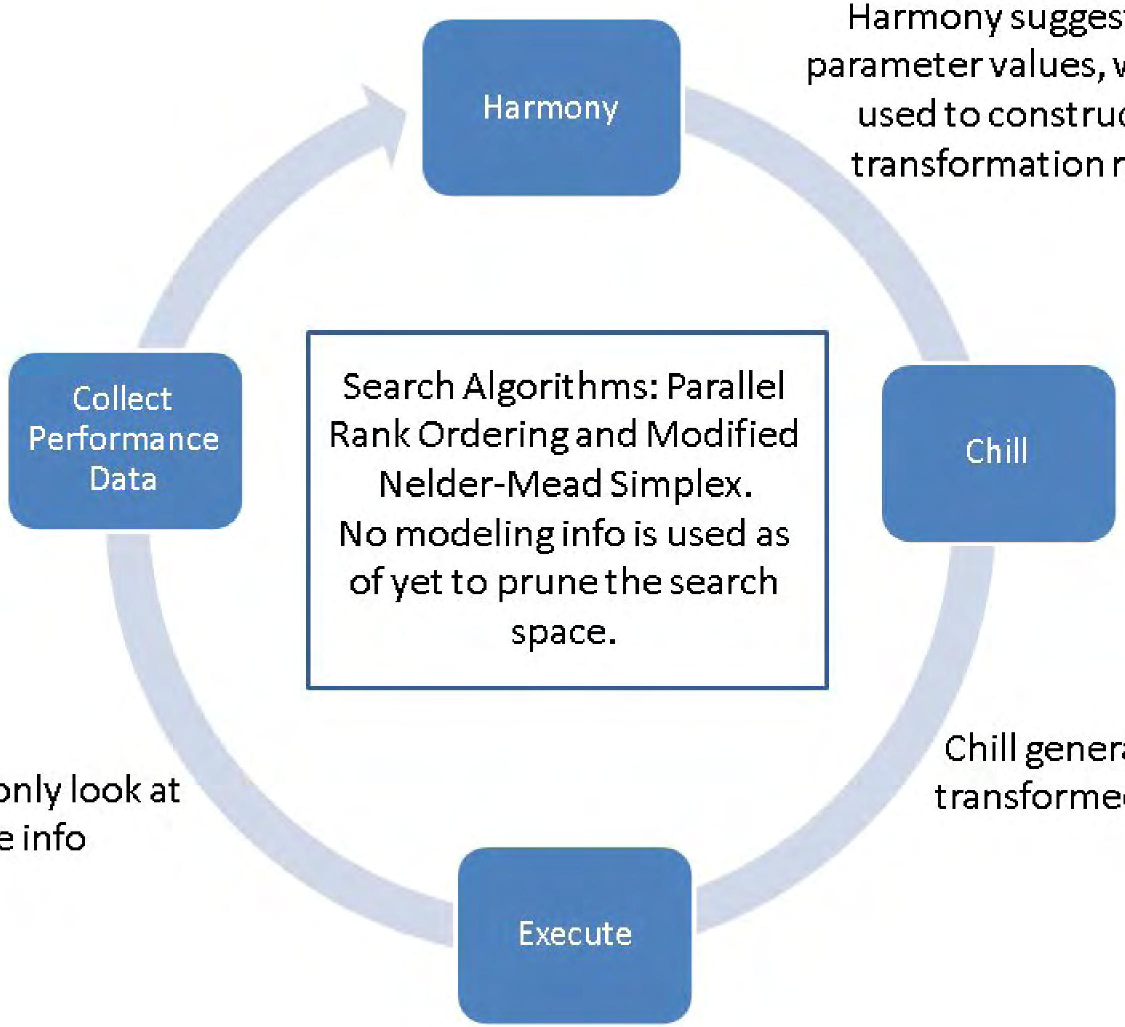
PRO

Param Sweep

HJ

Search Algo

Active Harmony & CHiLL



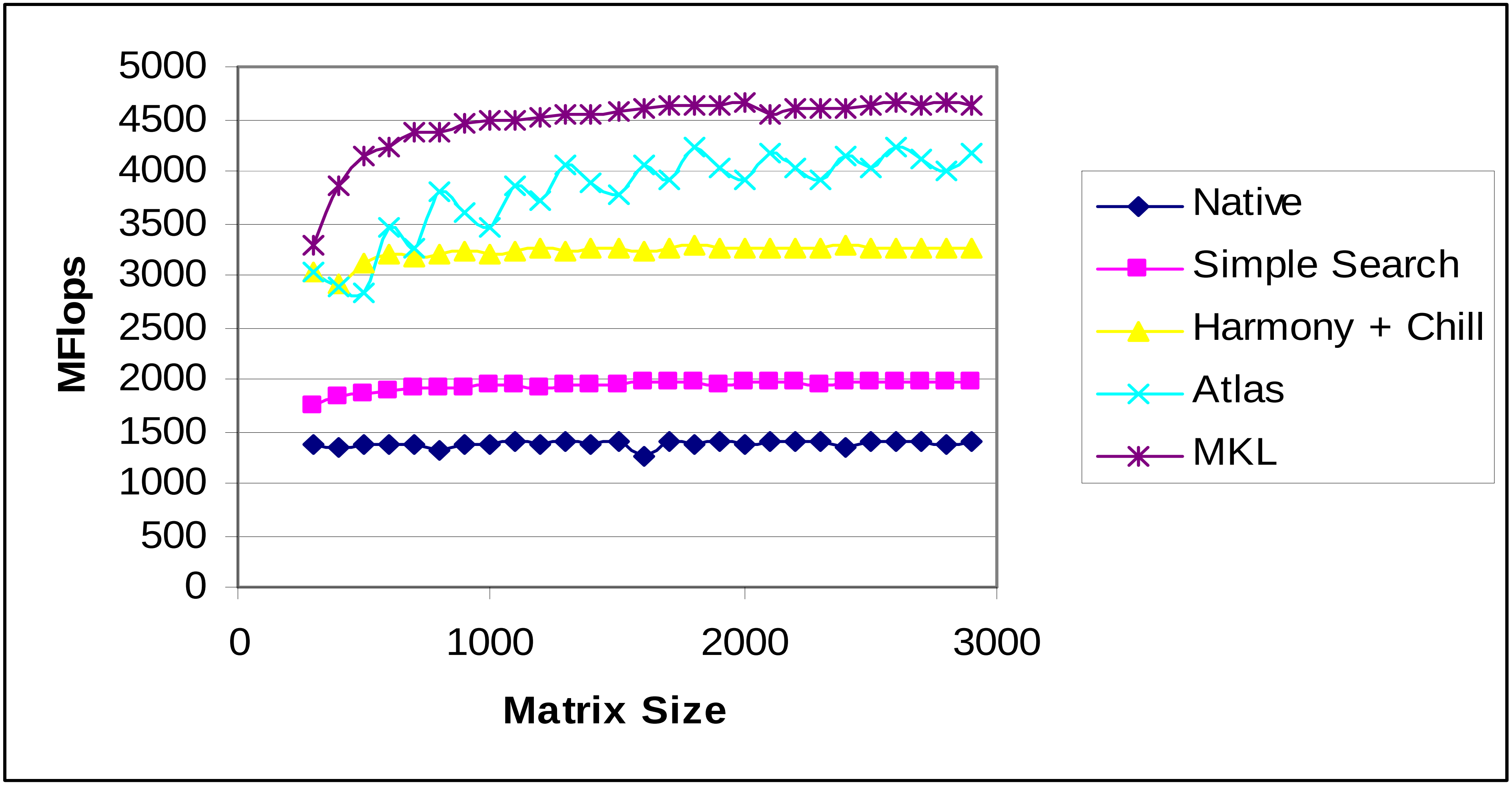
Harmony suggests new parameter values, which are used to construct chill transformation recipe.

Search Algorithms: Parallel Rank Ordering and Modified Nelder-Mead Simplex. No modeling info is used as of yet to prune the search space.

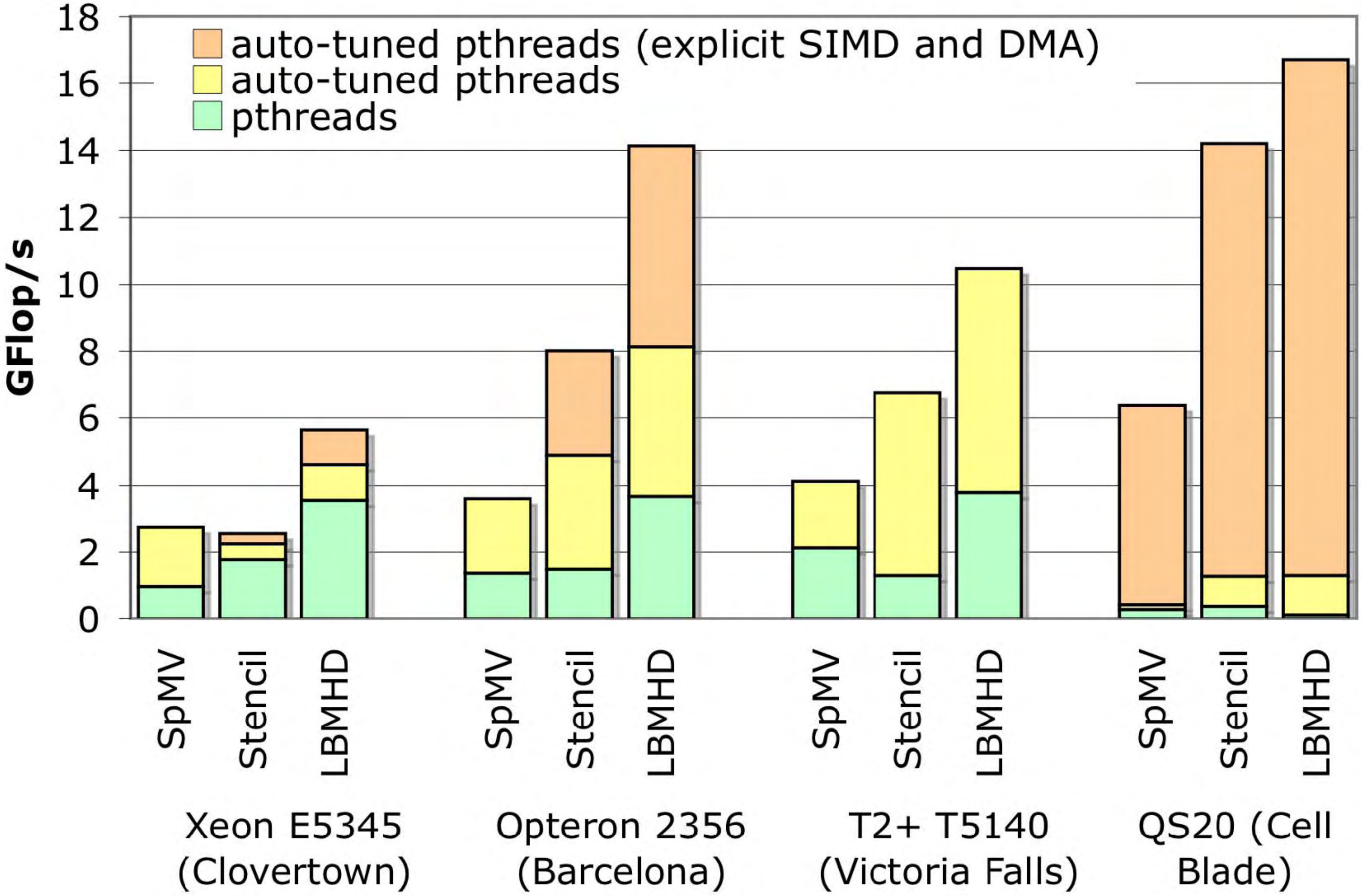
Chill generates the transformed source

For now, we only look at runtime info

Matrix Multiply Performance ActiveHarmony + CHiLL



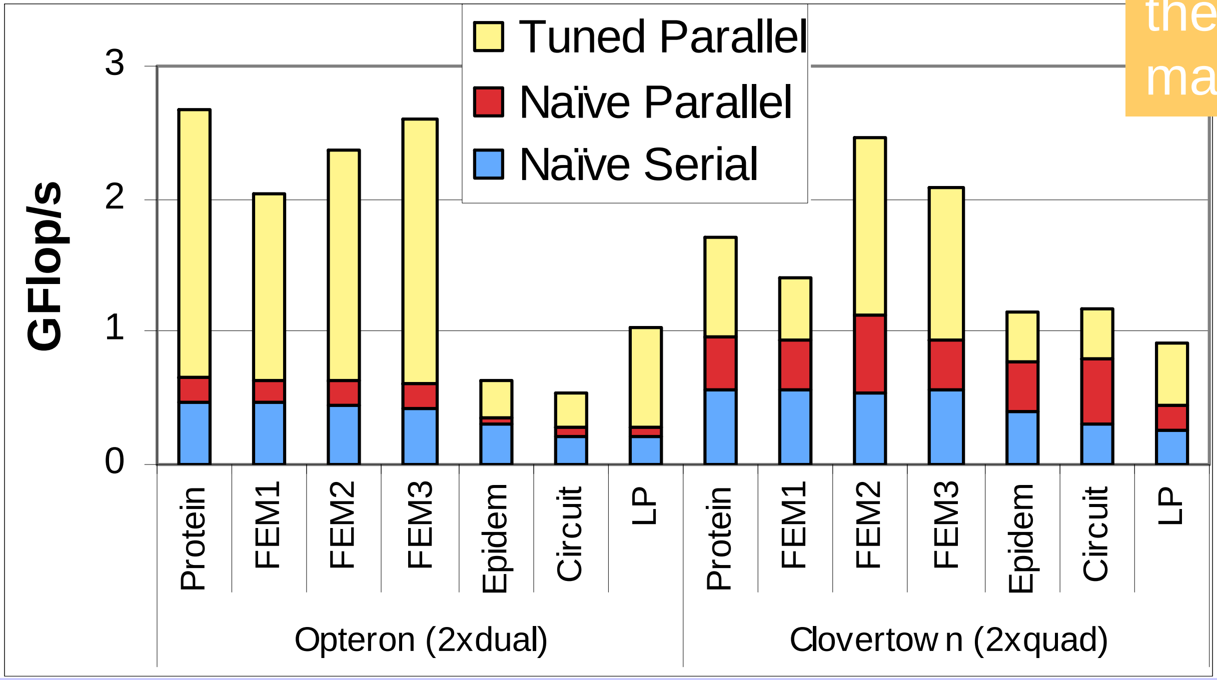
Autotuning Results
3 Apps & 4 Systems



Autotuning SpMV in Applications

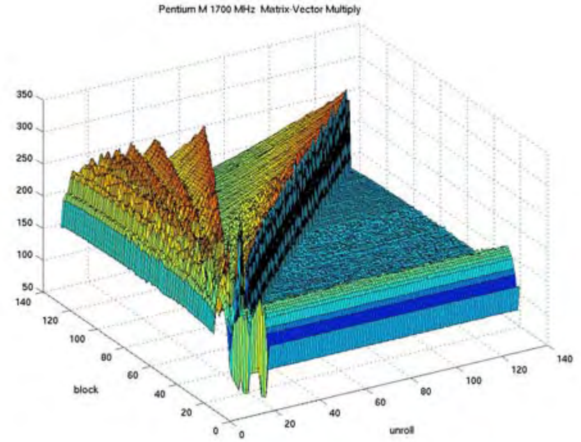
- Sparse Matrix-Vector Multiply (SpMV) tuning steps:
  - Register Block to compress matrix data structure (choose *r1xr2*)
  - Cache Block so corresponding vectors fit in local memory (*c1xc2*)
  - Parallelize by dividing matrix evenly (*p1xp2*)
  - Prefetch for some distance *d*
  - Machine-specific code for SSE, etc.

Autotuning for memory is more important than parallelism on these (admittedly small #core) machines!

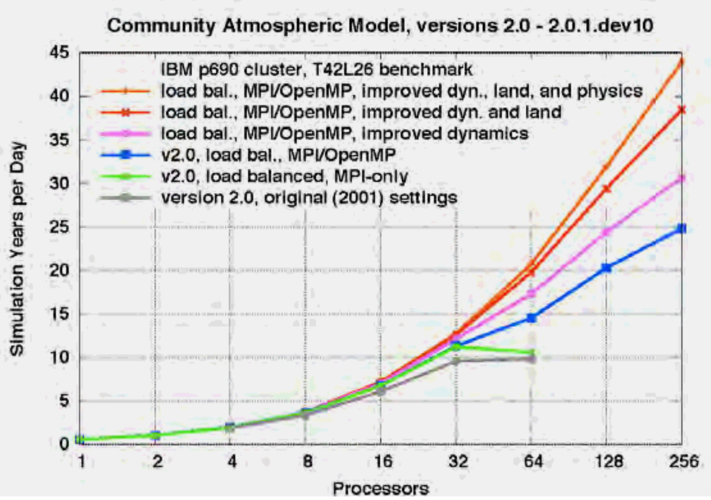


Application Engagement

- **Application Engagement**
  - Work directly with DOE computational scientists
  - Ensure successful performance porting of scientific software
  - Focus PERI research on real problems
- **Application Liaisons**
  - Build long-term personal relationships with PERI researchers and scientific code teams
- **Tiger Teams**
  - Focus on DOE's highest priorities
    - SciDAC-2
    - INCITE
    - Joule metric



Optimizing arithmetic kernels



Maximizing scientific throughput

Currently Active Application Liaisons

Advanced Methods for Electronic Structure Application

Center for Plasma Edge Simulation

Simulations of Turbulent Flows with Strong Shocks and Density Variations

Modeling Multiscale-Multiphase-Multicomponent Subsurface Reactive Flows using Advanced Computing

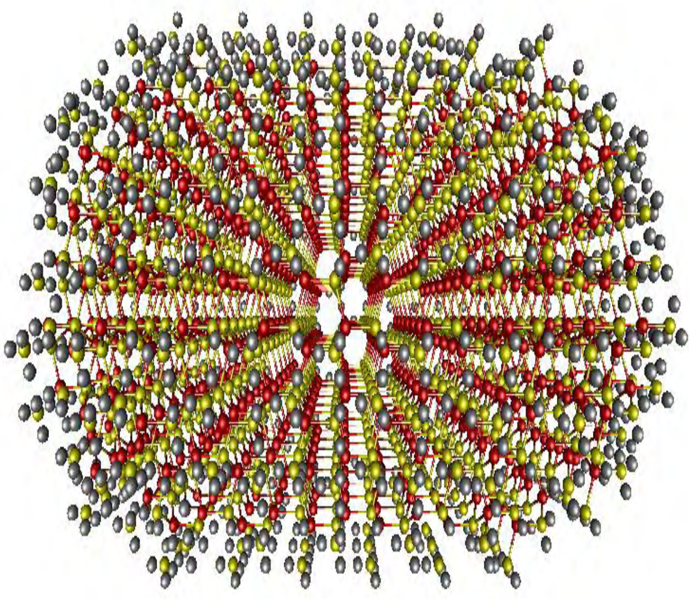
Linear Scale Electronic Structure Calculations for Nanostructures

Hierarchical Petascale Simulation Framework for Stress Corrosion Cracking

Community Petascale Project for Accelerator Science and Simulation

LDS3DF Liaison

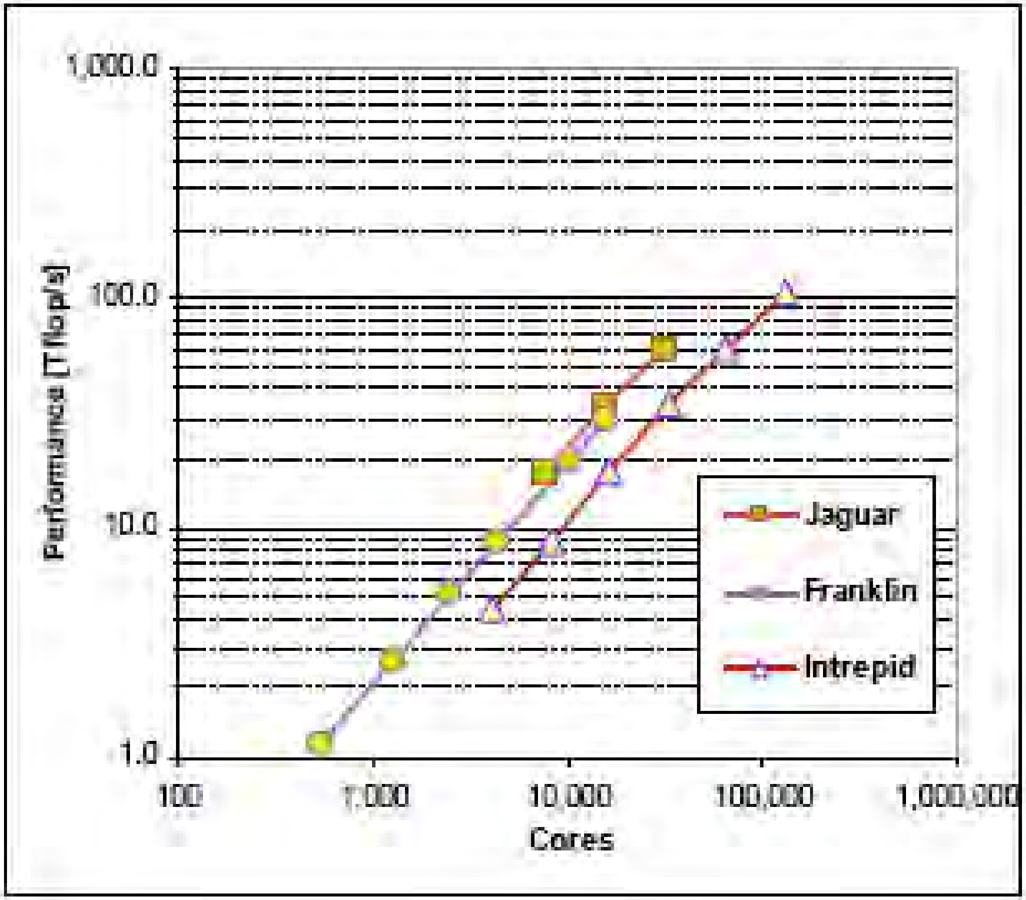
- LS3DF: a novel divide and conquer approach for electronic structure calculations.
- Cost scales as O(n) in number of atoms, rather than $O(n^3)$ as with conventional density functional theory (DFT) approaches.
- Developed by Lin-Wang Wang at LBNL.
- PERI liaison: Bailey, Gunter, Shan.
- Scaling limited to 2048 cores, 3 Tflop/s.
- Performance profiling showed some load imbalance, plus large amount of time in I/O.
- PERI personnel assisted tuning by replacing I/O with MPI communication.
- Other improvements made by Wang and his team.



$Cd_{961}Se_{724}H_{948}$

LS3DF Performance After Tuning

System	Cores	Tflop/s	%peak
Franklin	17,280	32.2	35.8
Jaguar	30,720	60.3	23.4
Intrepid	131,072	107.5	24.2



Gordon Bell Finalist at SC08:

Lin-Wang Wang, Byounghak Lee, Hongshan Shan, Zhengji Zhao, Juan Meza, Erich Strohmaier, David H. Bailey, "Linearly Scaling 3D Fragment Method for Large-Scale Electronic Structure Calculations," SC08, to appear.

2007 Tiger Teams

Joule metric is to double performance or scientific output

2007 Joule codes were:

Chimera	**supernovae**	**Tony Mezzacappa**	**ORNL**
S3D	**combustion**	**Jackie Chan**	**SNL CA**
GTC	**fusion**	**Stephane Ethier**	**PPPL**

PERI focused on S3D and GTC

2007 Tiger Team Participants

GTC Tiger Team:

UTK	Shirley Moore, Lead; Haihang You
LBNL	Hongzhang Shan
Rice	John Mellor-Crummey
Oregon	Kevin Huck

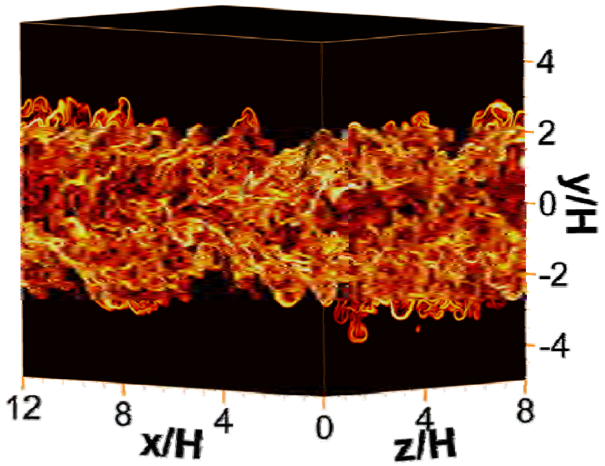
S3D Tiger Team:

LLNL	Bronis de Supinski, Lead
Rice	John Mellor-Crummey
SDSC	Allan Snavely
Oregon	Allen Maloney
ORNL	Pat Worley

Tiger Team Results

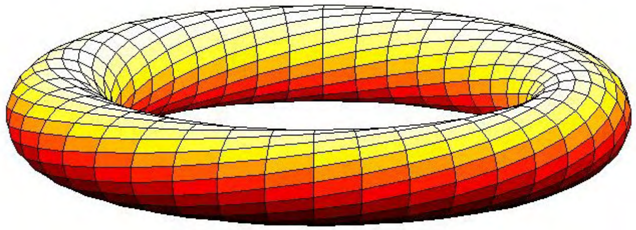
S3D, led by Jacqueline Chen at Sandia:

- **Performance tools found that unrolling by first index yielded a 7.5% overall performance increase.**
- **A change to getrates resulted in 10% overall performance increase on IBM P4.**
- **Several changes to the loop structure resulted in a 7% overall improvement.**



GTC, led by Zhihong Lin at UC Irvine:

- **Overall, performance increased by 13% on Cray XT3/4.**
- **Semi-automatic transformations improved performance by 33% on Itanium2 and 13% on Opteron 275.**
- **Some additional code transformations resulted in 37% increase on Itanium2 nodes.**
- **Changes to chargei improved performance by 10%.**



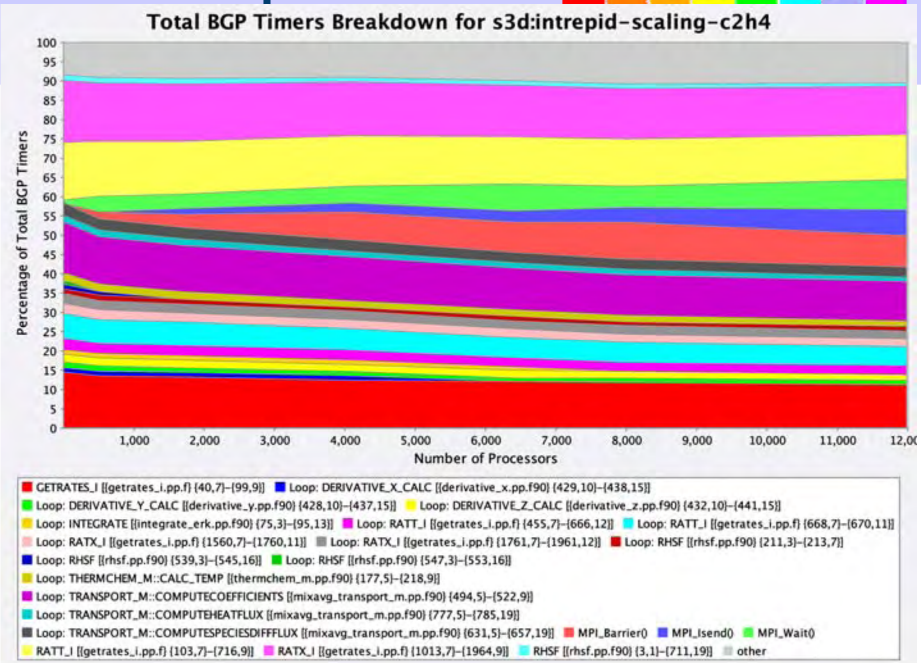
Graphics: Thanks to J. Chen, W. W. Lee and Z. Lin.

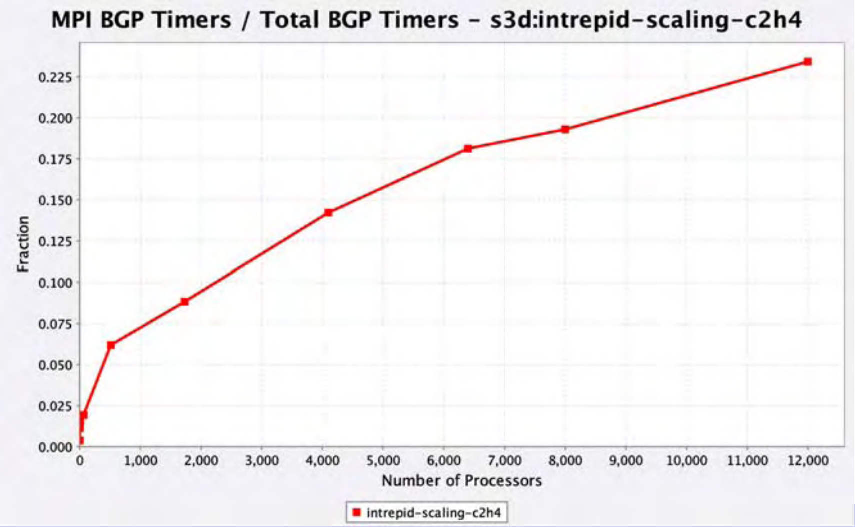
Architecture Tiger Team

- **Recently DOE/SC assigned PERI a new task to study the affinity of applications to architectures, with a focus on Petascale and beyond.**
- **PERI has responded by organizing a new "Architecture Tiger Team" activity:**
  - **Performance analysts to understand current performance.**
  - **Modelers to predict future performance to guide future procurements.**
  - **Initially focused on three carefully chosen Pioneer Applications.**
- **Measuring performance on present-day systems:**
  - **Focus on existing Leadership Facilities (e.g., Jaguar and Intrepid).**
  - **Understand (and improve) baseline performance.**
  - **Ensure highest quality versions used for future projections.**
- **Projecting performance to future systems:**
  - **Must anticipate future architecture trends.**
  - **Extend PERI convolution methods to extrapolations to larger systems.**
  - **Validate through alternative PERI modeling techniques.**

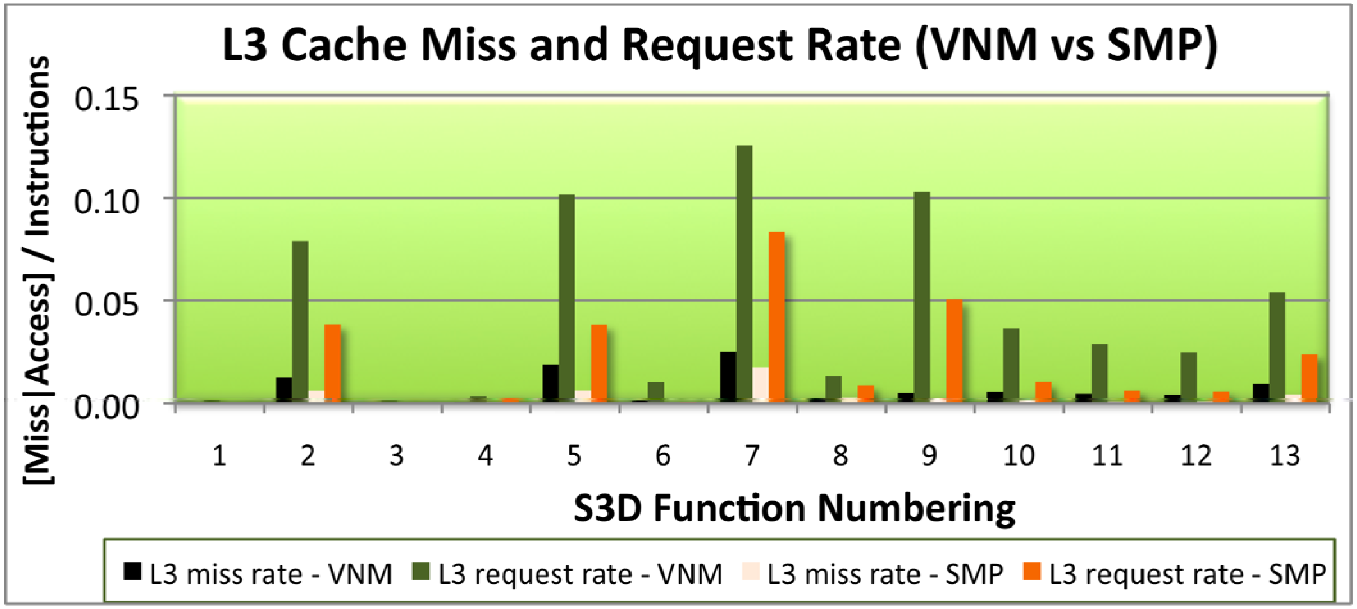
Data Collection
S3D on Intrepid

- Weak scaling experiments performed on up to 12000 cores
  - 30000 core experiment pending
  - All Experiments performed in VN mode
- TAU data collected for time only
  - Instrumentation overhead <5% to ~20%
  - Outer level loops included in instrumentation
  - Lightweight routines excluded
- Computation routines scale well
- Scaling degrades primarily from MPI
  - Load imbalance in MPI_Wait
  - Random node allocation testing will verify MPI topology effect
- Additional results available at http://tau.uoregon.edu/s3d





S3D Measurements Intrepid vs Jaguar



1	RATT_I
2	RHSF
3	RATX_I
4	COMPUTECOEFFICIENTS
5	COMPUTESPECIESDIFFFLUX
6	MPI_Wait
7	INTEGRATE
8	CALC_TEMP
9	COMPUTEHEATFLUX
10	DERIVATIVE_X_CALC
11	DERIVATIVE_Y_CALC
12	DERIVATIVE_Z_CALC
13	DERIVATIVE_X_COMM

- **Detailed event-based performance measurements: IPC, FLOPS, Control transfer-related measurements; Memory measurements: L1 Data & Instruction, L2, TLB, L3**
- **L3 cache behavior for different core cases: 4 cores (VNM) vs. 1 core per node (SMP)**

Total Runtime Jaguar:	VNM: 813 s	SMP: 613.4 s
Total Runtime Intrepid:	VNM: 3005.74 s	SMP: 3014.55 s

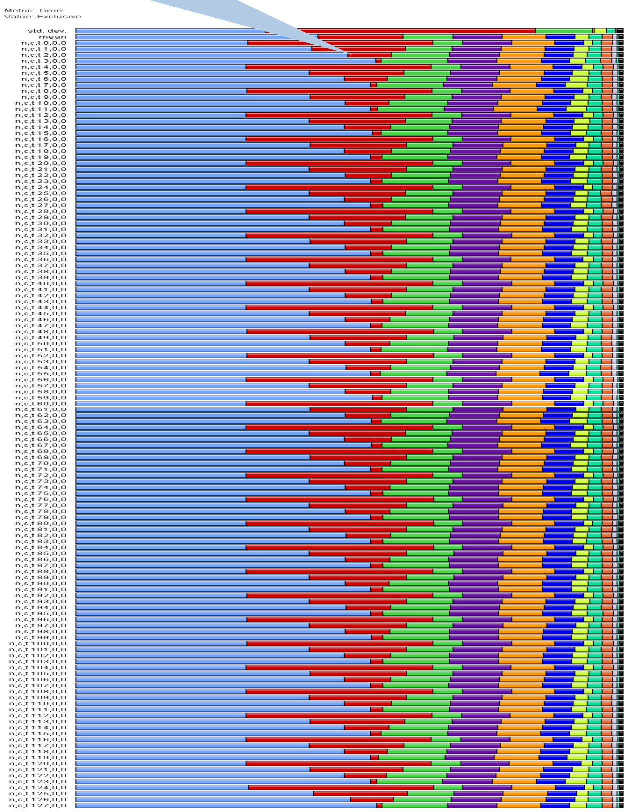
- **L3 serves as victim cache for L2: if data is not in L2, L2 TLB checks L3 (→ L3 request)**
- **Why do L3 requests and misses increase so dramatically in VNM on Jaguar?**

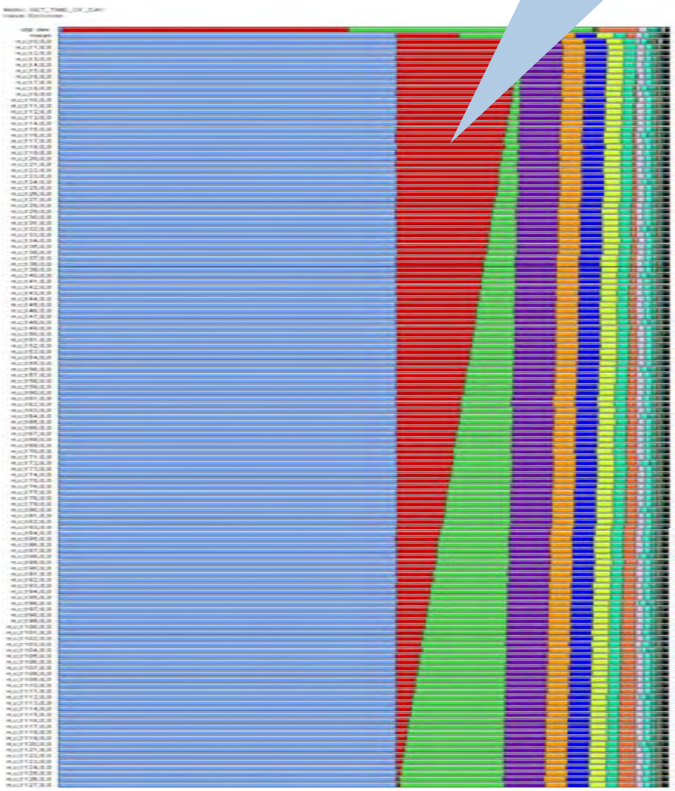
Data Collection
TAU Profiles of GTC

Load imbalance due to incorrect particle initialization

128 process runs on Jaguar

Corrected particle initialization results in less severe load imbalance





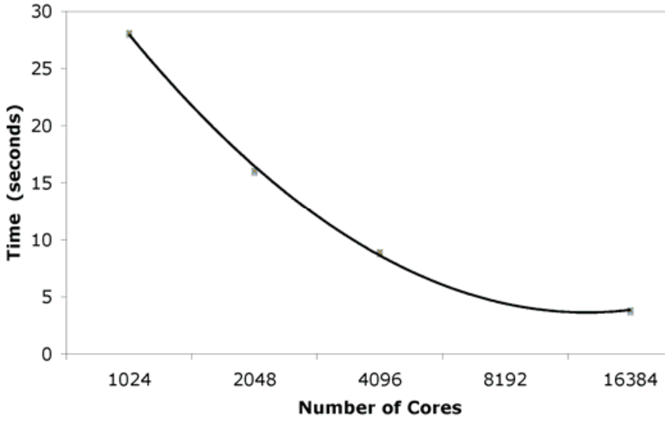
Profiling helps ensure that a valid version is used for modeling.

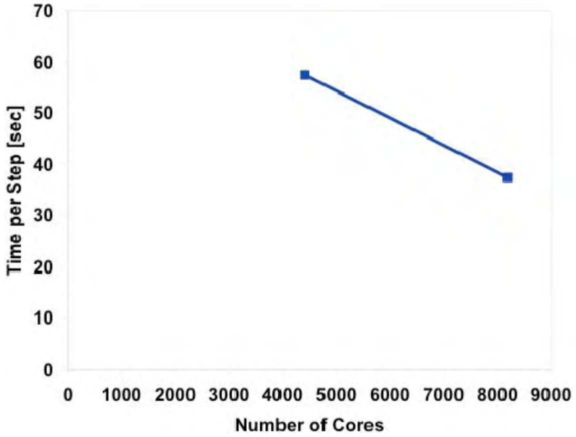
Architecture Tiger Team Early GTC Findings

- Application performance degrades over time as memory locality degrades because of increasing particle disorder.
- Cache utilization is lower than ideal, hurting performance.
  - Vector of structures yields low spatial locality for loops that access only a few fields.
  - Loop nests stream through particles and fail to exploit significant temporal reuse.
- Concerns about scalability with GTC's current domain decomposition of poloidal planes for shaped plasma simulations
  - A new version of GTC with 2-D domain decomposition was not available to PERI for study.
- PERI researchers have observed load imbalances related to particle initialization

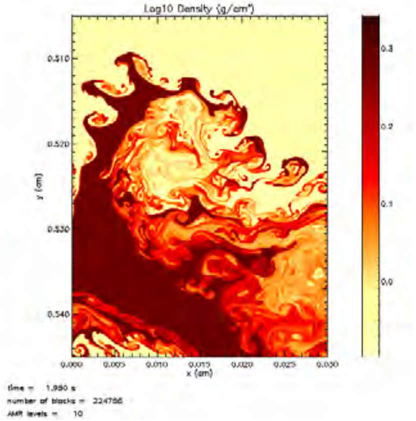
Data Collection FLASH on Intrepid

Strong Scaling

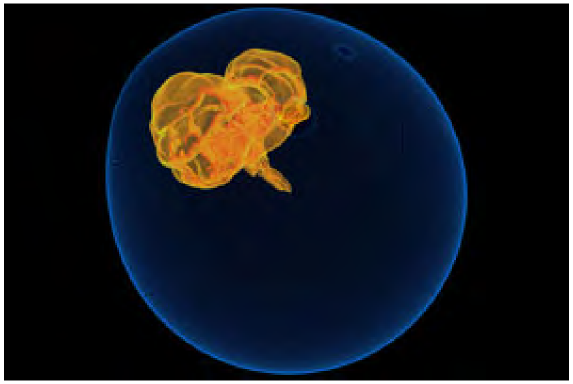




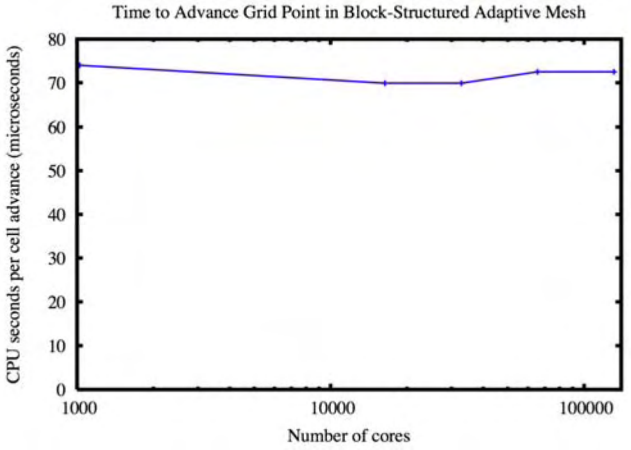
Turbulence-Driven Nuclear Burning

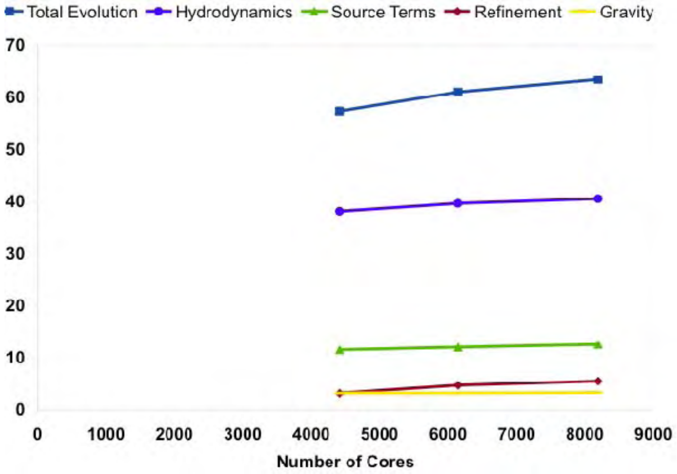


White Dwarf Deflagration



Weak Scaling





Architecture Tiger Team
Looking Forward

- Complete data collection for first three applications
- Report affinity to today’s machines based on measurements
- Project affinity to future machines
  - NDAs are an issue
- Select additional applications, and repeat the process

Summary

- PERI is addressing Petascale performance problems
- Application Engagement
  - Liaisons with SciDAC application teams
  - Tiger Teams
- Modeling
  - Informs tuning efforts
  - Broader impact on system acquisitions
- Automatic tuning
  - Long-term research goal
  - Alleviate recurring burden