Colorado State University

SAIMI: Separating the Algorithm from the Implementation Details

Michelle Mills Strout

with Christopher Krieger, Andrew Stone, Christopher Wilcox, Amanreet Bajwa, and Samantha Wood

DOE ASCAC Meeting -- August 24, 2011

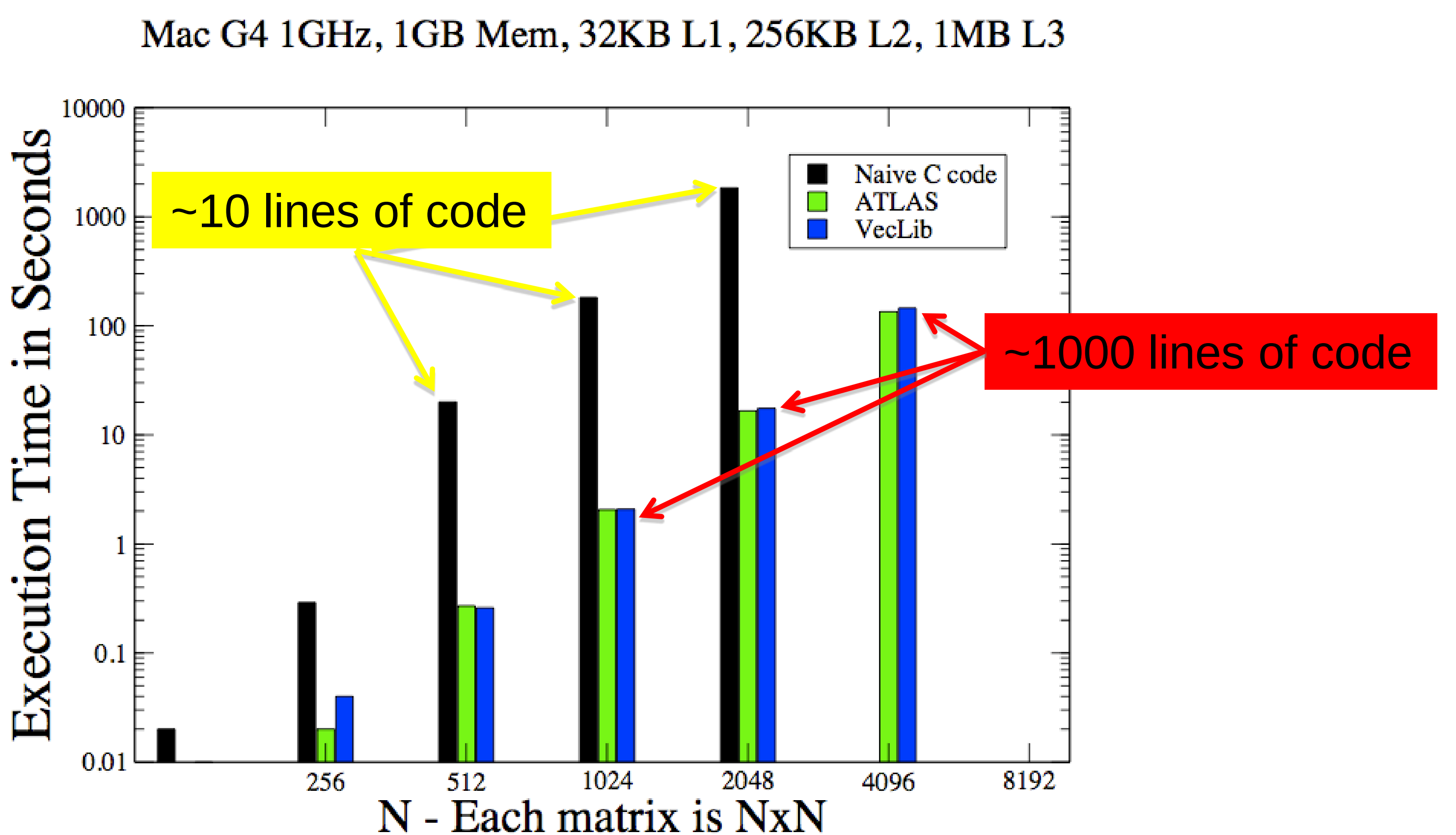
U.S. DEPARTMENT OF ENERGY

SAIMI is funded as a DOE Early Career Award #DE-SC3956

The Problem
(from the perspective of a compiler person)

- Scientific simulations need to run faster!
  - while being easier to write, evolve, and maintain
  - while using less energy
  - while staying portable
- Developers make them run faster by doing performance tuning by hand
  - Compilers great at low-level optimizations
  - General purpose compilers struggle with automating higher-level optimizations
- After hand-tuning, the algorithm and implementation details are tangled!

Matrix-Matrix Multiply Example:
Writing fast code is hard!



U.S. DEPARTMENT OF ENERGY

Colorado State University

Why writing fast code is hard

- In most prevalent programming models …
  - Schedules for computations specified with loops
  - Storage allocation specified with array declarations and accesses
- Separation of the when and where the from algorithm is an important idiom that needs library and compiler support.

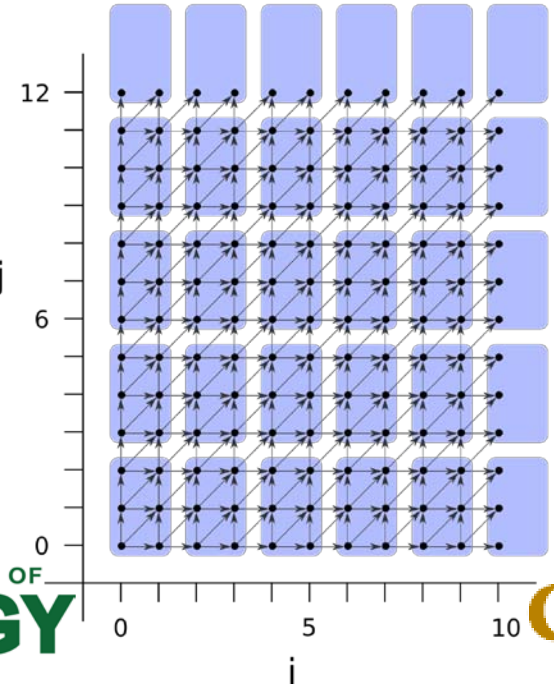
Hand-tuning Example (Tiling)

Original Loop

```
for (i=0; i<11; i++) {
  for (j=0; j<13; j++) {
    A[i,j] = 1/2 * (A[i,j-1] + A[i-1,j-1] + A[i-1,j]);
  }
}
```

Tiled Loop

```
TiLB = -1; TiLB = ((int)ceild(tiLB,2) * 2);
for (Ti = TiLB; Ti <= 10; Ti += 2) {
 TjLB = -2; TjLB = LB_SHIFT(TjLB,3);
 for (Tj = TjLB; Tj <= 12; Tj += 3) {
   for (i= max(Ti,0);i<=min(Ti+2-1,10);i++) {
     for (j= max(Tj,0);j<=min(Tj+3-1,12);j++) {
       A[i,j] = 1/2 * (A[i,j-1] + A[i-1,j-1] + A[i-1,j]);
     }
   }
 }
}
```



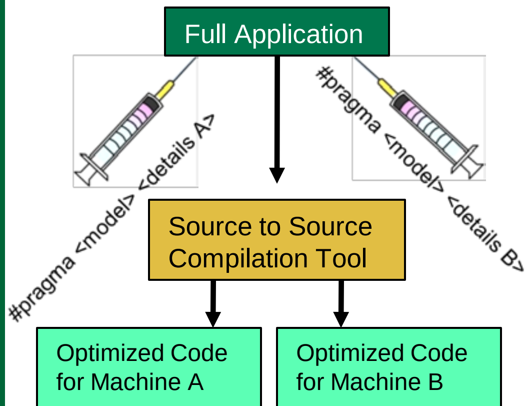
Orthogonal Loop Scheduling is Possible: Chapel Example

Tiled Loop

```
D = [0..10 , 0..12];
for (i,j) in tile(D,(2,3)) {
  A[i,j] = 1/3∗(A[i,j−1]+A[i−1,j−1]+A[i−1,j]);
}
```

- tile is an iterator construct.
- Iterators in Chapel enable the orthogonal specification of the schedule to use when visiting points in a domain.
- The Domain construct in Chapel has some limitations.

SAIMI - Separating Algorithm and Implementation via programming Model Injection



- Keep code in existing general purpose programming languages
- Annotate sub computations with pragmas to inject implementation details
- Focus on three "injectable" programming models: expressions, sparse polyhedral model, task graphs due to sparse tiling
- Show approach can be used on DOE applications (e.g., CGPOP miniapp)

U.S. DEPARTMENT OF ENERGY

Colorado State University

Specifying Implementation Details Orthogonally

Source-to-Source compilation tool	Algorithm Specification	Implementation Details
OpenMP	for loops (some restrictions)	static or dynamic, block or not, private and shared, …
CUDA	for loop (some restrictions)	unroll, vectorize, data movement, …
ORIO, POET, …	for loop (some restrictions)	unroll, tile, various loop transformations, …

SAIMI project focus

Mesa	expressions	lookup table optimization
IEGen	for loops with indirect accesses	inspector/executor strategies specified using Sparse Polyhedral Framework (SPF)

Key SAIMI Components

- Mesa transformation tool
  - Algorithm: expressions
  - Implementation details: look-up table optimization
- Sparse Polyhedral Framework (SPF)
  - Algorithm: loops with indirect array accesses
  - Implementation details: inspector/executor strategies
- Evaluation within the context of applications relevant to DOE

Mesa: Lookup Tables for Expressions as an Injectable Programming Model

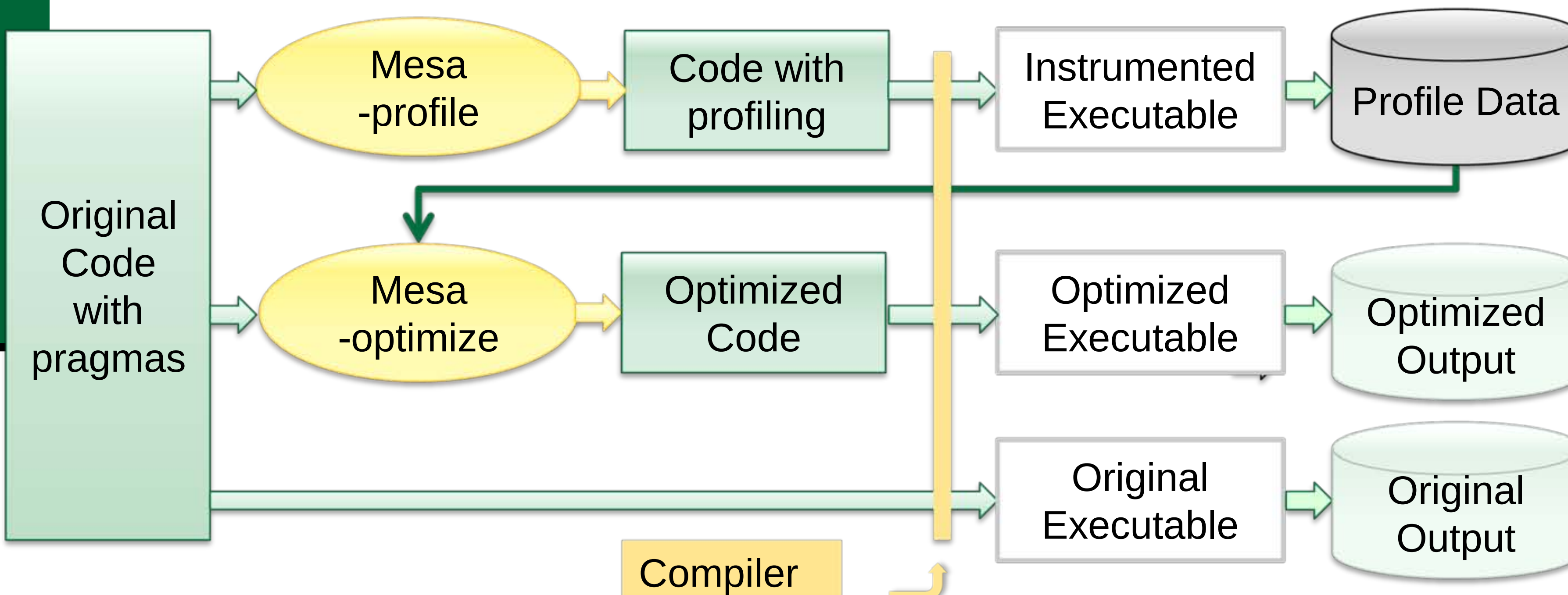
```
// Iterate steps (outer loop)
for (step = 0; step < 1000; ++step) {
  // Iterate atoms (middle loop)
  for (atom1 = 0; atom1 < vecAtoms.size(); ++atom1) {
    // Iterate atoms (inner loop)
    for (atom2 = atom1; atom2 < vecAtoms.size(); ++atom2) {

      // Compute distance between atoms
      float fDistance = distance(atom1, atom2);
      // Compute scattering angle
      float fTheta = m_fStep * (float)(step + 1);
      // Combine parameters to scatter
      float rTheta = fDistance * fTheta;

      // Optimize subexpression shown below
      #pragma LUTOPTIMIZE
      fIntermediate = sinf(FOURPI * rTheta) / (FOURPI * rTheta);
    }
  }
}
```

U.S. DEPARTMENT OF ENERGY

Colorado State University

Approach used in Mesa

- Automate the tedious and error prone elements of look-up table optimization via the Mesa tool (based on ROSE)
- Help programmers to improve performance (5x to 7x on 3 real applications) with clear knowledge of the effect on accuracy.



U.S. DEPARTMENT OF ENERGY

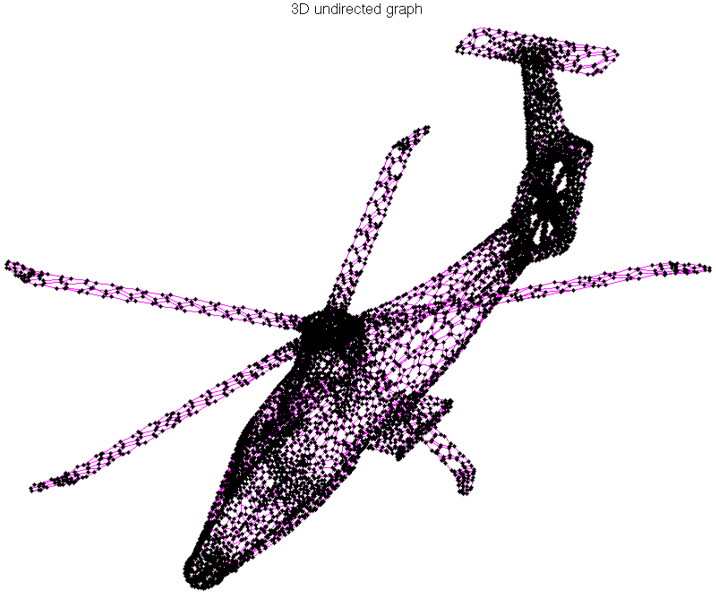
Colorado State University

Key SAIMI Components

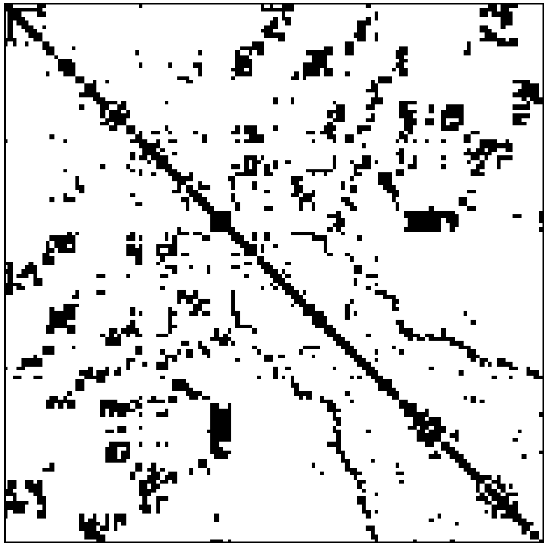
- Mesa transformation tool
  - Algorithm: expressions
  - Implementation details: look-up table optimization
- Sparse Polyhedral Framework (SPF)
  - Algorithm: loops with indirect array accesses
  - Implementation details: inspector/executor strategies
- Evaluation within the context of applications relevant to DOE

Sparse Matrix Computations

Mesh



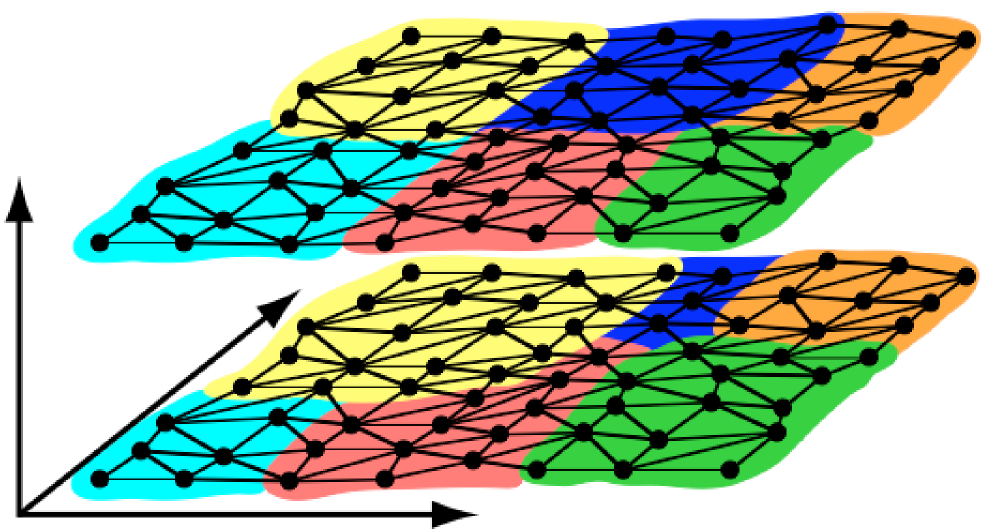
Sparse Matrix



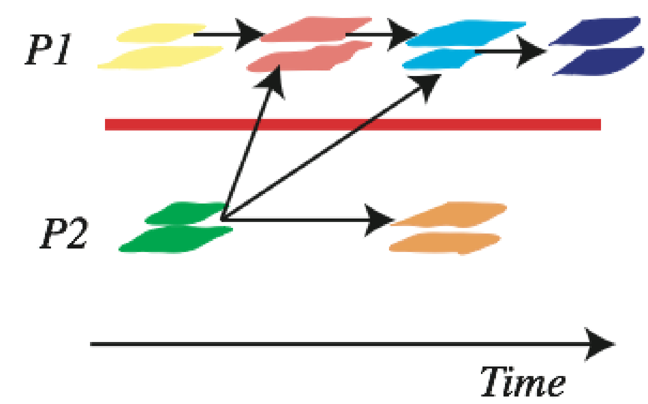
http://www.cise.ufl.edu/research/sparse/matrices/Pothen/commanche_dual.html

Parallelizing Iterative Sparse Matrix Computations

Break computation that sweeps over mesh, or sparse matrix, into chunks/sparse tiles



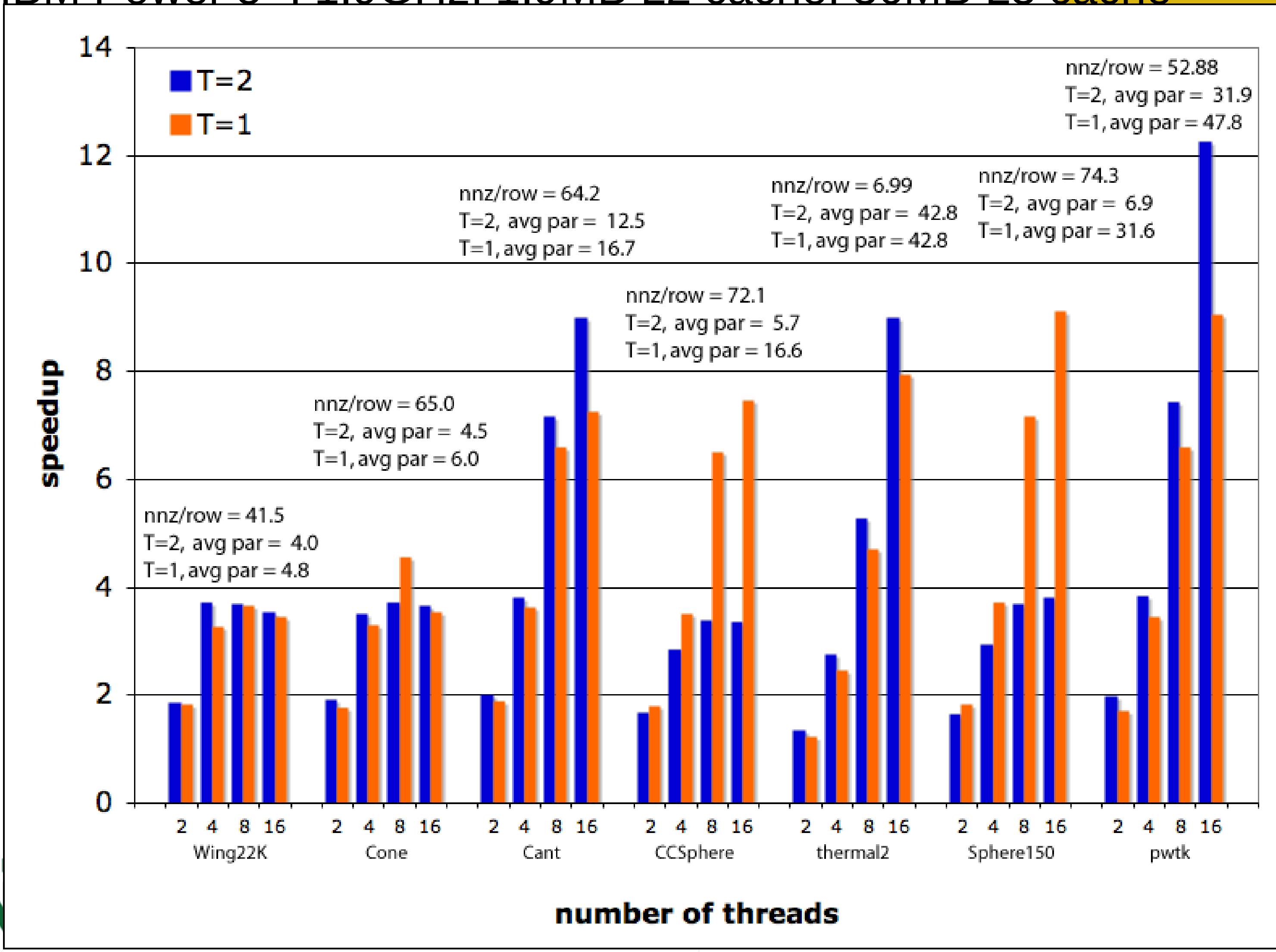
Full Sparse Tiled Iteration Space



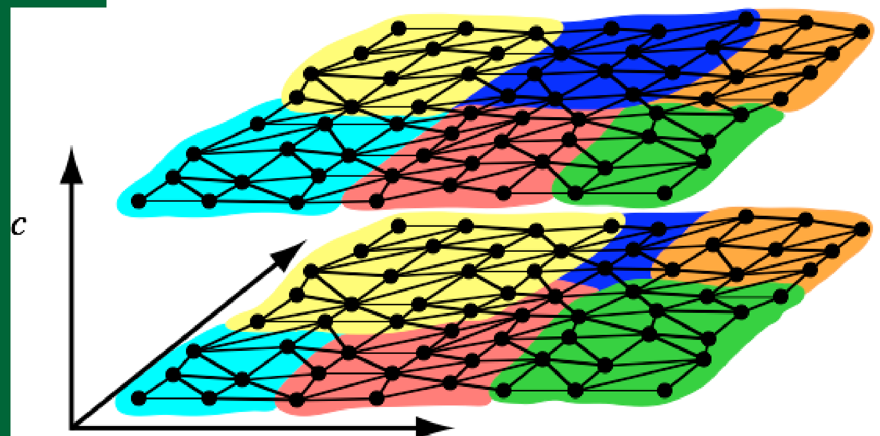
Task Graph

Experimental Results for Blue Ice

IBM Power 5+, 1.9GHz, 1.9MB L2 cache, 36MB L3 cache



Sparse Polyhedral Framework: An Injectable Programming Model



- Specifying sparse computations

```
for (c=1; c<=2; c++) {
  for (i=0; i<N; i++) {
    Z[i] += f(i's neighbors);
  }
}
```

- Specifying transformations like full sparse tiling

$$\{[c,i] \rightarrow [t,c,i] \mid t = \Theta(c,i)\}$$

Key SAIMI Components

- Mesa transformation tool
  - Algorithm: expressions
  - Implementation details: look-up table optimization
- Sparse Polyhedral Framework (SPF)
  - Algorithm: loops with indirect array accesses
  - Implementation details: inspector/executor strategies
- Evaluation within the context of applications relevant to DOE

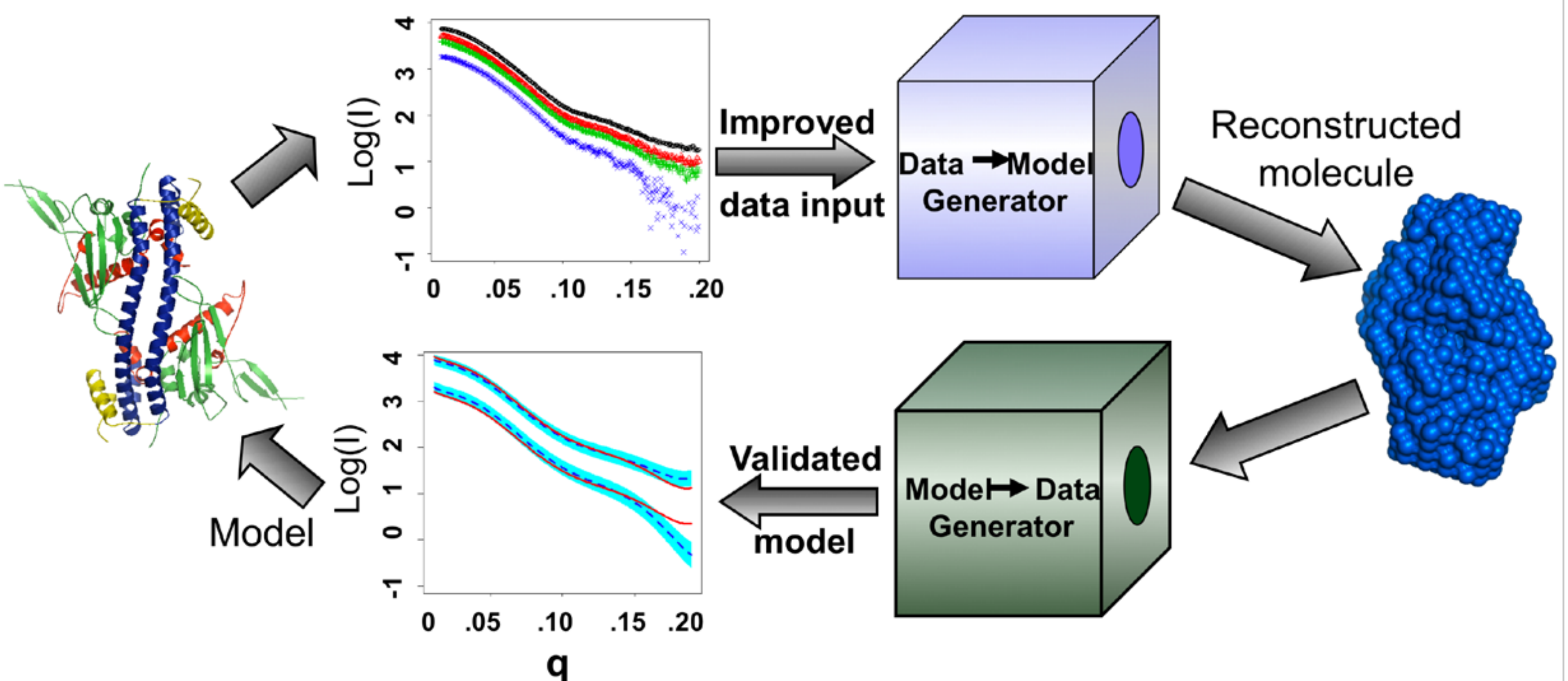
Evaluating the Research

- Apply separate and composed injectable programming models to existing DOE-relevant apps
  - Evaluate programmer control and tangling
  - Evaluate performance
- Current Applications
  - SAXS: Small Angle X-ray Scattering
  - CGPOP: Miniapp for Parallel Ocean Program
  - Matrix Powers Kernel: (CACHE project related)

SAXS Project

http://www.cs.colostate.edu/hpc/SAXS/

SAXS: Small X-ray Scattering



With molecular dynamics and SAXS, biochemists are investigating structure of proteins that interact with DNA.

CGPOP Miniapp Released July 2011

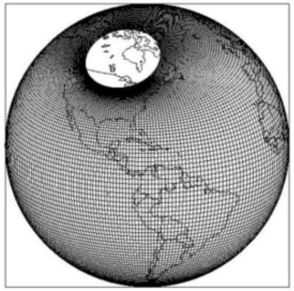
http://www.cs.colostate.edu/hpc/cgpop/

The CGPOP Miniapp

Introduction

The Parallel Ocean Program (POP), developed at Los Alamos National Laboratory, is an important multi-agency code used for global ocean modeling and is a component within the Community Earth System Model (CESM). The motivation for creating a miniapp for the POP developer team is that it will enable them to ensure the performance portability of the most critical portion of the application while also testing new programming models. The CGPOP miniapp is the conjugate gradient solver from LANL POP 2.0, which is the performance bottleneck for the full POP application. The CGPOP miniapp is written in Fortran90 with MPI and is about 3000 source lines of code (SLOC), whereas the POP application is 71,000 SLOC.

Download



- Release 1.0 [.tgz (281 MBs)]
- Tile files [.tgz (1.3 GBs)]

Resources

- Techical report with installation instructions [PDF]
- Doxygen Documentation
- GoogleCode Page

Contributors

- Andrew Stone [webpage]
- John Dennis [webpage]
- Michelle Strout [webpage]

Acknowledgements

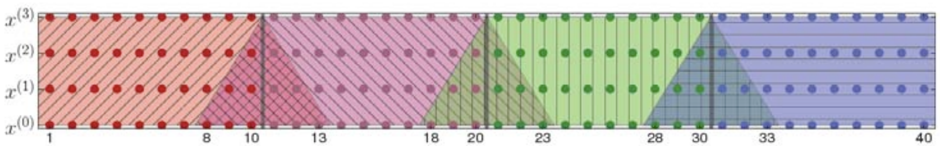
- This work was supported by Department of Energy Early Career Award #DE-SC3956.
- This work was financially supported through National Science Foundation Cooperative Grant NSF01 which funds the National Center for Atmospheric Research (NCAR), and through the grant: #OCI-0749206.

U.S. DEPARTMENT OF ENERGY

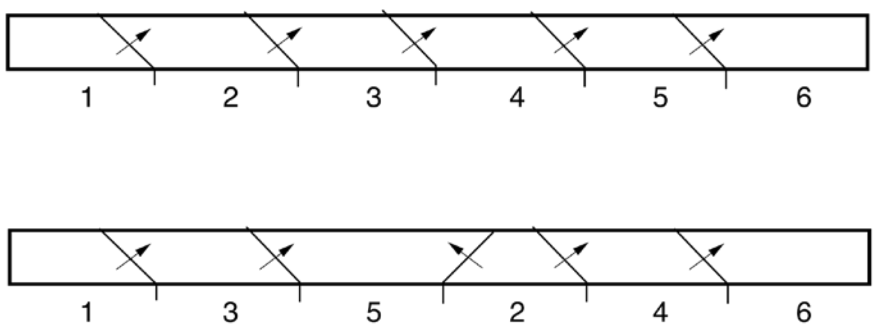
Colorado State University

Variants of Matrix Powers Kernel

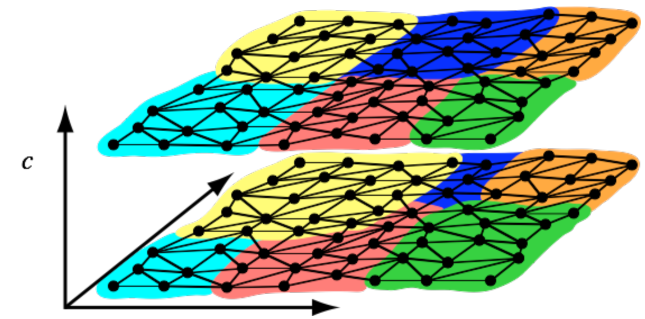
communication avoiding [Mohiyuddin 2009]



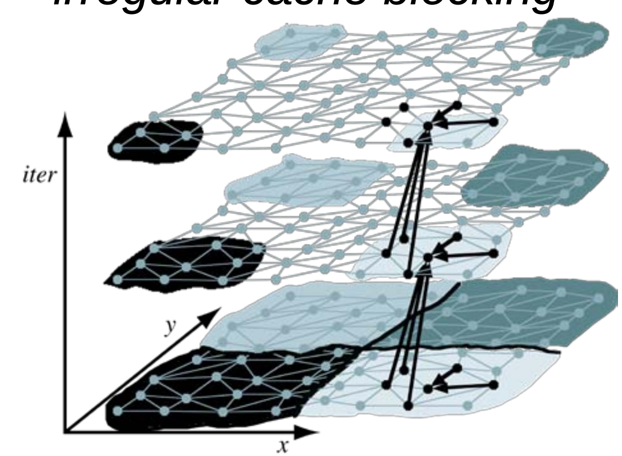
full sparse tiling variants



full sparse tiling



irregular cache blocking



Parameters: sparse tile width and height, graph partitioner, etc.

Conclusions

- Scientific computing needs detangling of simulation codes!
- Complete rewrites are not feasible so gradual approaches need to be developed
- Pragmas already have buy-in and can be used to orthogonally specify implementation details with minimal tangling
- The concept of SAIMI should direct future programming model development

SAIMI Crew

- Chris Krieger – Task graph programming model
- Andy Stone – Orthogonal grid and algorithm specifications for geoscience applications leveraging polyhedral model and SPF
- Chris Wilcox – Mesa and look up table optimizations
- Amanreet Bajwa – Creation of tile dependence graph for moldyn
- Alum: Alan LaMielle – IEGen prototype
- Alum: Jon Roelofs – IEGenCC tool