A Novel Computational Framework for Reactive Flow and Multiphysics Simulations

James C. Sutherland
and
Tony Saad

AICHE Annual Meeting
October 16 - 21, 2011
Minneapolis, MN, USA
Exascale

One Billion, Billon Operations per Second
That’s 1,000,000,000,000,000,000
“It says it’s sick of doing things like inventories and payrolls, and it wants to make some breakthroughs in astrophysics”
Hardware

2 Giga-Watts of power!!!

Power is a dominant constraint.
#include <stdio.h>
int main(void)
{
    int count;
    for (count = 1; count <= 500; count++)
        printf("I will not throw paper airplanes in class.");
    return 0;
}
- Hardware
- Software
- Money
Software
formulate problem

difference equations

algorithm

IMPLEMENT

WE'RE GOING TO TRY SOMETHING CALLED AGILE PROGRAMMING.

THAT MEANS NO MORE PLANNING AND NO MORE DOCUMENTATION. JUST START WRITING CODE AND COMPLAINING.

I'M GLAD IT HAS A NAME.

THAT WAS YOUR TRAINING.
What If?

other discretization

other models

parallel framework
Software Complexity

courtesy of Philip J. Smith,
Institute for Clean & Secure Energy
Imagine...

- self assembly
- mesh independent
- cluster, thread, GPU
Data Dependencies!
\[ J_h = -\lambda \nabla T + \sum_{i=1}^{n_s} h_i J_i \]

**MODEL A**

\[ \lambda = \lambda_0 = \text{const} \]

\[ J_i = -\sum_{j=1}^{n_s} D\nabla Y_j \]

\[ h_i = h_i(T) \]

**MODEL B**

\[ \lambda = \lambda(T, p, Y_i) \]

\[ J_i = -\sum_{j=1}^{n_s} D_{ij}(T, p, Y_k)\nabla Y_j \]

\[ -D_i^T(T, p, Y_k)\nabla T \]

\[ h_i = h_i(T) \]

Expression Concepts
\[
\frac{\partial \phi}{\partial t} + \nabla \cdot u\phi = \nabla \cdot \Gamma \nabla \phi + S_\phi
\]

- An Expression is a software representation of a mathematical expression
- An Expression computes fields it represents
- Each Expression indicates which expressions it depends on
In Practice...
\[
\frac{\partial \phi}{\partial t} + \nabla \cdot \mathbf{u}\phi = \nabla \cdot \Gamma \nabla \phi + S\phi
\]

\[\Gamma \equiv \Gamma(T, p, y_i)\]

- Register all expressions
- Determine root expression
- Construct tree
- Deduce storage requirements, and other metrics from graph
- Execute graph in reverse order: That's the algorithm!
\[ \frac{\partial m_k}{\partial t} + m_{k+1} = 0; \quad k = 0, 1, \ldots, 2n - 1 \]

\[ m_{n+1} = \sum_{i=1}^{n} w_i r_i^i \]
Parallelism
- Domain Decomposition
- Algorithm Decomposition
- Fine Grained Parallelism

\[ \nabla^2 \phi + S \]

One expression
(calculated on a patch/workset)
Priority Queue Threading

Allows “backfilling” based on graph

Each expression receives signals from its dependents when they complete execution. When all are done, the expression enters the priority queue.
Overhead?

\[
\frac{\partial T}{\partial t} = -\frac{1}{\rho c_p} \nabla \cdot (-\lambda \nabla T)
\]

- Staggered, structured FV mesh
- Gradient, interpolant & divergence operators.

The overhead of the expression graph approach does not contribute in any meaningful way to the execution time.
• Independent of parallel framework
Acknowledgments

- **James C. Sutherland (PI)**
  Assistant Prof. Chemical Engineering

- **Matthew Might**
  Assistant Prof., School of Computing

- **Devin Robinson**
  M. S. Student

- **Christopher Earl**
  Ph.D. Candidate

This material is based upon work supported by the National Science Foundation under Grant No. PetaApps-0904631.

This research was sponsored by the National Nuclear Security Administration under the Accelerating Development of Retrofittable CO2 Capture Technologies through Predictivity program through DOE Cooperative Agreement DE-NA0000740.
Questions?

THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:

"MY CODE'S COMPILING."

HEY! GET BACK TO WORK!

COMPILING!

OH. CARRY ON.