New Approaches to Optimize High Performance Scientific Applications

Daniel Orozco

University of Delaware
http://www.udel.edu

Computer Architecture and
Parallel Systems Laboratory
http://www.capsl.udel.edu

The Wedding Cake Problem

A Baker is given the task to prepare a wedding cake as soon as possible.

The question is, how soon can the baker do it?

A Recipe for a Wedding Cake

Get Ingredients



Make several small cakes


Make the cream


Prepare the interlayer

Put cream on it

Assemble

Decorate

How should you write the “cake” program?

A possible cake program:

Call Get_ingredients();

Call Bake_one_layer();

Call Wait_for_others();

Call Make some Cream();

Call Put some cream();

If (I am processor 1)

  Assemble

  Decorate

Many problems remain.

Where is the data?

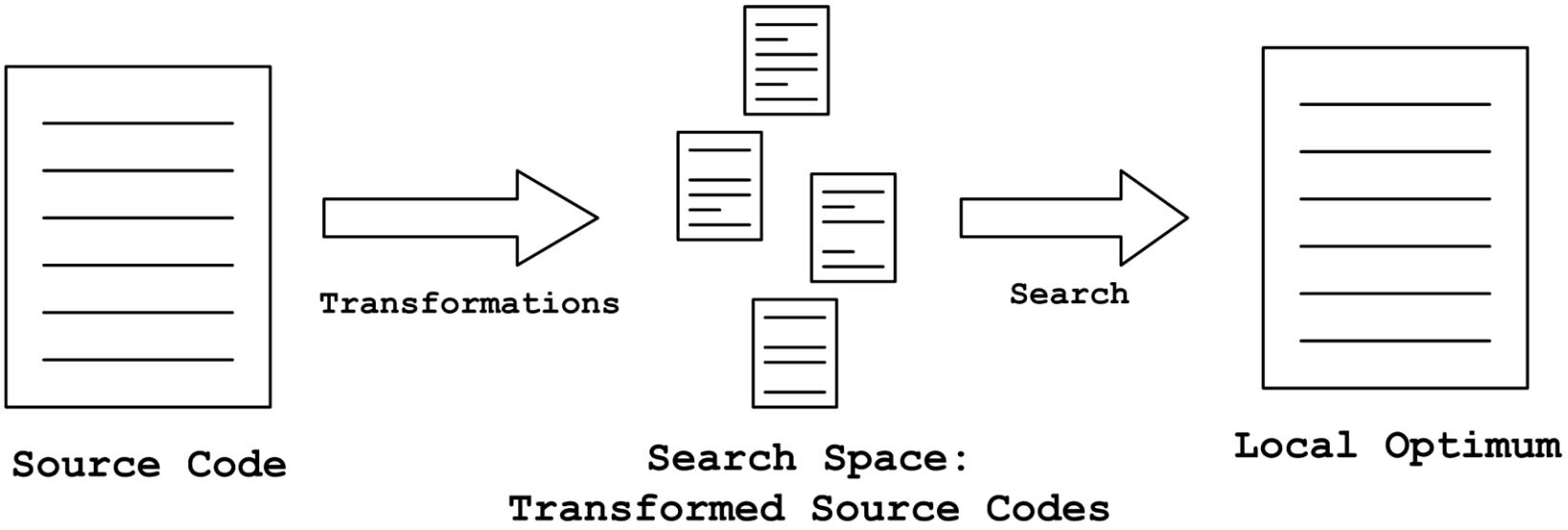
How many processes?

Is my strategy adequate?

Is my program correct?

Will the hardware run fast?

A Popular Approach:
Try several versions of your code.



This approach, although conceptually simple, is fundamentally flawed.

The versions generated depend on the skill of the programmer.

What is the problem?

Traditional programming models try very hard to hide the truth: **Parallel programming is not serial programming.**

Current models are popular because they “feel” serial.

- MPI
- OpenMP
- C, Fortran

Optimization becomes very hard.

Things you need to know to optimize a parallel program (today).

You need to know the application,

and the Operating System Behavior,

and the architecture of the machine,

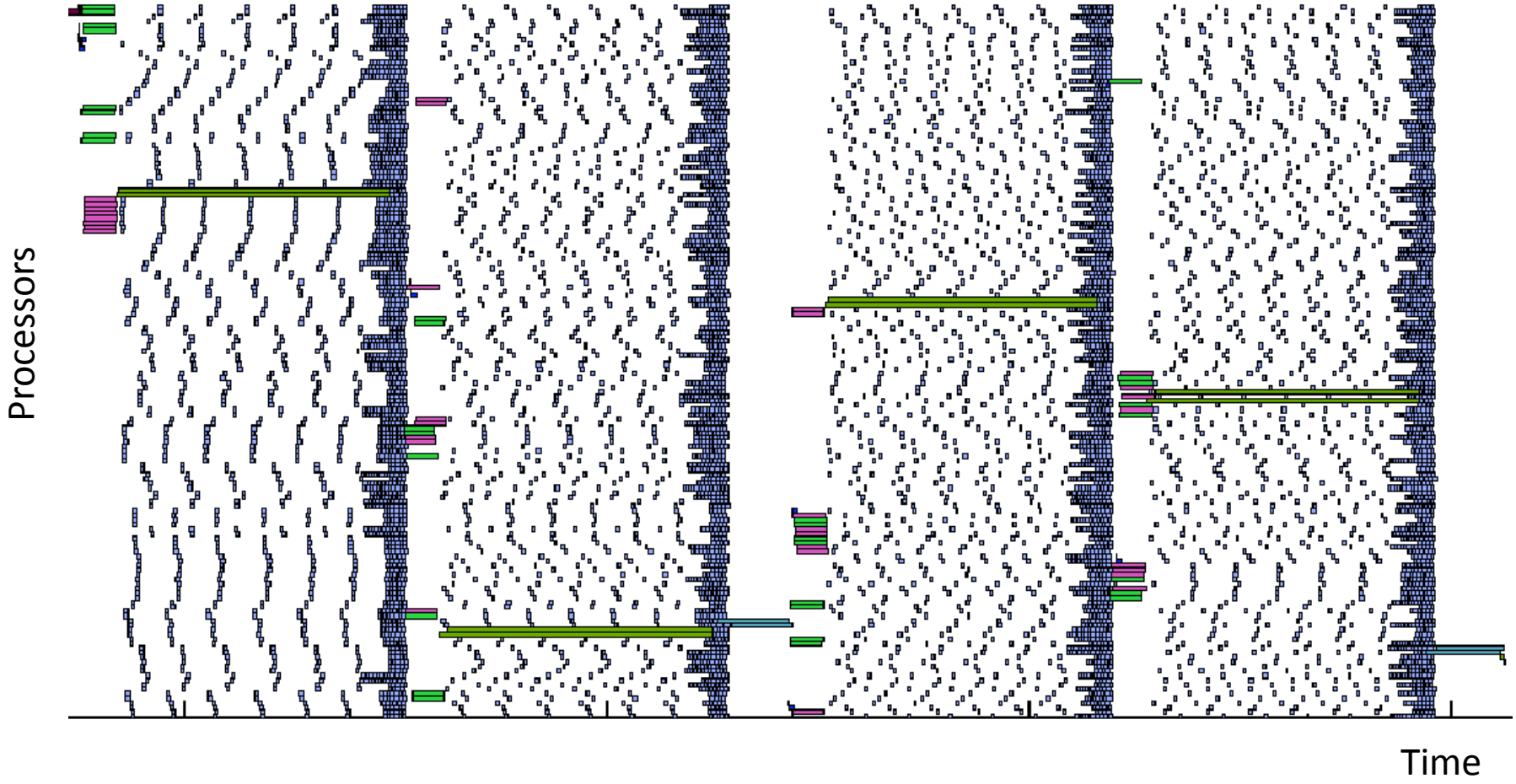
and the type of network,

and about synchronization,

and about scheduling…

It took me 10 years of continuous study!

Or else you get this!



A profile of Matrix Multiply on 160 processors.
Processors are idle on the white regions. Colored regions represent functions called.

Dataflow can change things

The scientist provides a description, rather than a program.

It should be the responsibility of the machine to decide on a good way to execute your code.

This was the original intent of computing, but they have only complicated things!

Who understands what goes one behind Linux anyway?

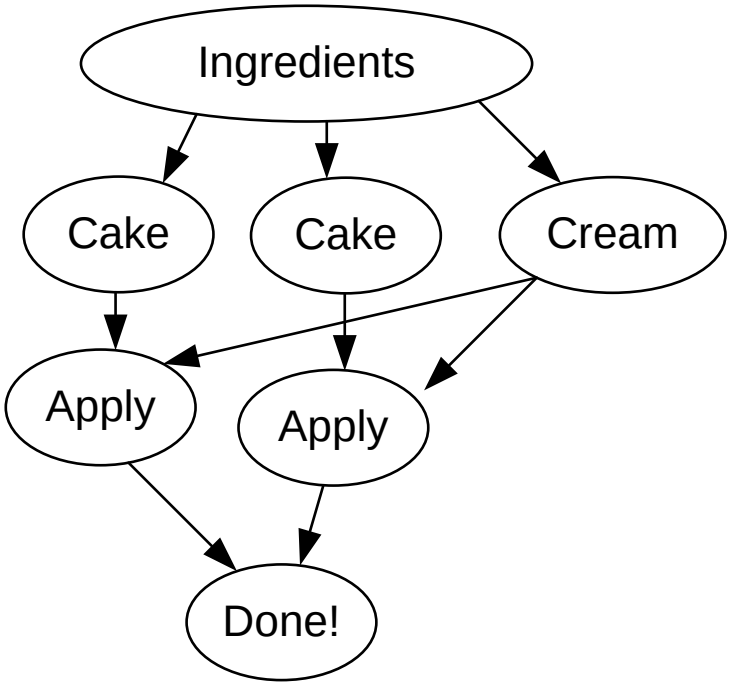
How should you write the “cake” program?

Dataflow!

The programmer **describes** the problem.

The compiler or the operating system take all other decisions:

- Where do I put the memory?
- When do I run each subroutine?
- How many processors can be used?



Matrix Multiply: A study case

Traditional approach: 8 months of work for an optimized version (and 10 years of education).

- Achieved 61 GFLOPS of a maximum of 80 GFLOPS.

Dataflow approach: 3 weeks of work for an optimized version.

- Achieved 53 GFLOPS.
- Description at the high level
- Code at the low level.

Dataflow Matrix Multiply

Closeup of a profile of Matrix Multiply.
Functions are called as they become available.

Discussion (Conclusions?)

Optimization is hard: A large amount of papers are about optimization of scientific programs.

Dataflow can provide a solution, but:

Should we change 60 years of education?

Should we lose billions in investment?

And the most difficult of all: **How do you convince people to stop using something that "works" for them?**