Amer Diwan

Jeremy Siek

Bor-Yuh Evan Chang

Sriram Sankaranarayanan

Programming Languages Research at the University of Colorado, Boulder

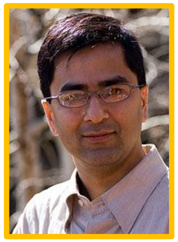
PL research at CU has ***breadth***!

How do we effectively express computation?

language design, type systems, logic

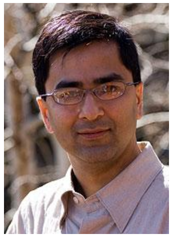
How do we make programs run efficiently?

performance analysis, compilation



How do we assist reasoning about programs?

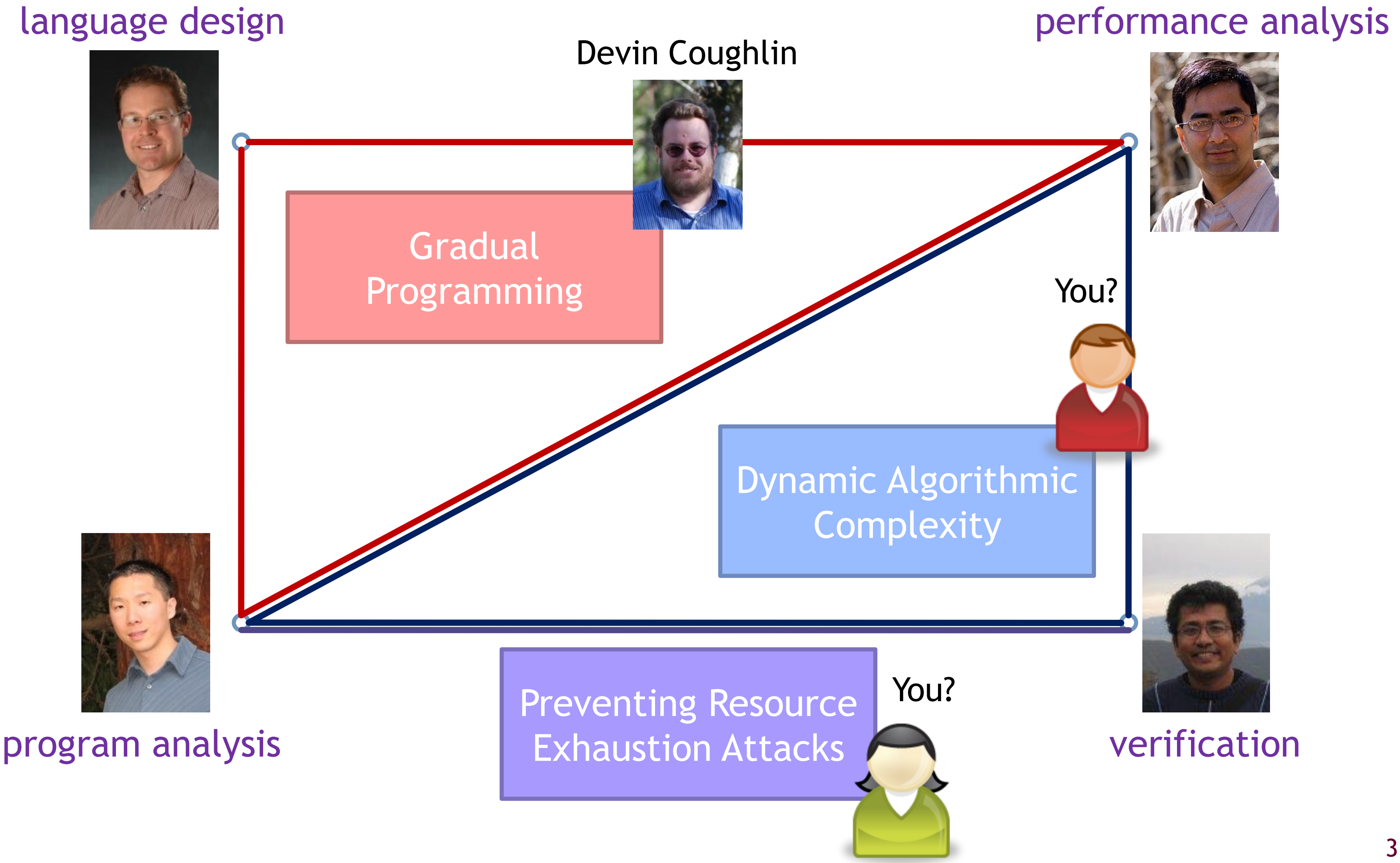
program analysis, development tools



How do we get reliable, secure software?

verification, model checking

PL researchers at CU *collaborate*!



language design

Devin Coughlin

performance analysis

Gradual Programming

You?

Dynamic Algorithmic Complexity

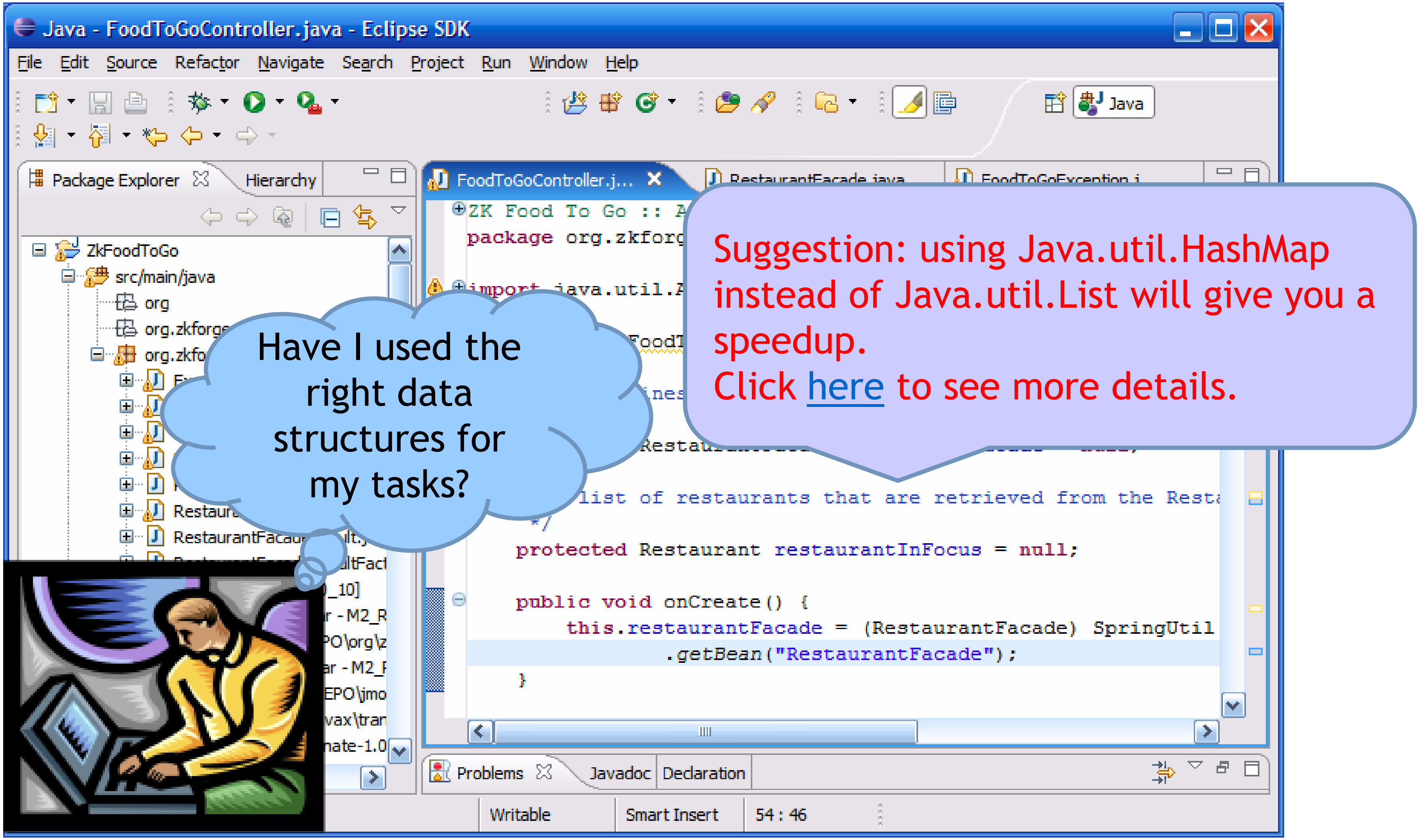
program analysis

Preventing Resource Exhaustion Attacks

You?

verification

Application: Auto Code Improvement



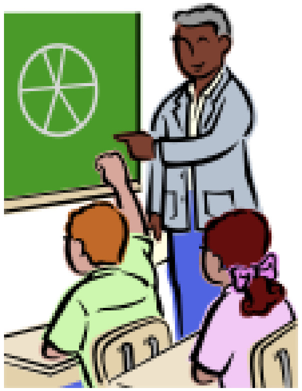
```
ZK Food To Go :: A
package org.zkforg

import java.util.A

            FoodT

            ines

            Restau                            null;

        list of restaurants that are retrieved from the Resta
     */
    protected Restaurant restaurantInFocus = null;

    public void onCreate() {
        this.restaurantFacade = (RestaurantFacade) SpringUtil
                .getBean("RestaurantFacade");
    }
```

Have I used the right data structures for my tasks?

Suggestion: using Java.util.HashMap instead of Java.util.List will give you a speedup.
Click here to see more details.

Computational Complexity

AVL Trees, Fibonacci heaps, O(n^2), P=NP?



Memory hierarchy: Caches, page faults, register allocation,..

Have I used the right data structures?

Have I used the right data structures?

```
Class MyContainer
    void addElement( Element x);
    Element chooseElt (...);
    Element findMatch (...);
    Element findMaximum (...);
    void printSorted (...);
```

Function	Hashtable	Balanced Tree
addElement	O(1)	O(log N)
chooseElt	O(1)	O(1)
findMatch	O(N)	O(log N)
findMaximum	O(N)	O(log N)
printSorted	O(N log N)	O(N)

How is the library being used?

Usage Profile for MyContainer

Function	Usage Fraction
addElement	70%
chooseElt	12%
findMatch	5%
findMax	10%
prettyPrint	3%

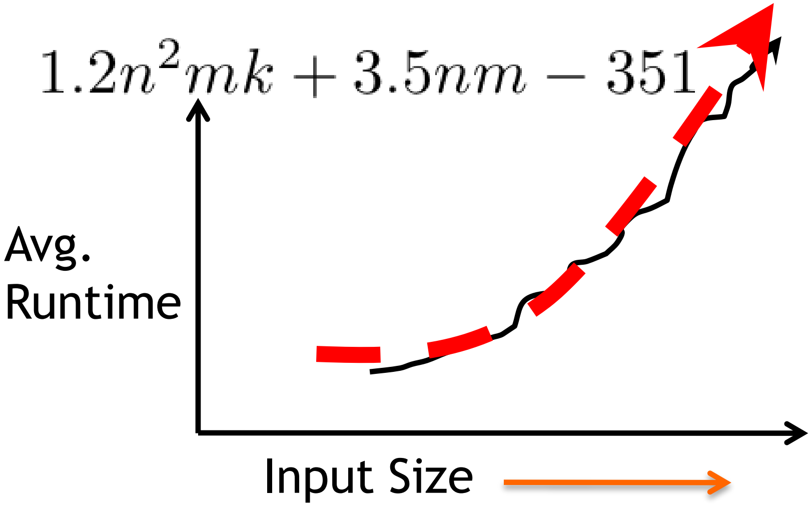
But wait, what would your systems professor say?

Conclusion: Use HashTable

Dynamic Complexity Estimation

Parameterized Unit Tests: Design unit test suites to simulate usage pattern and vary input size.

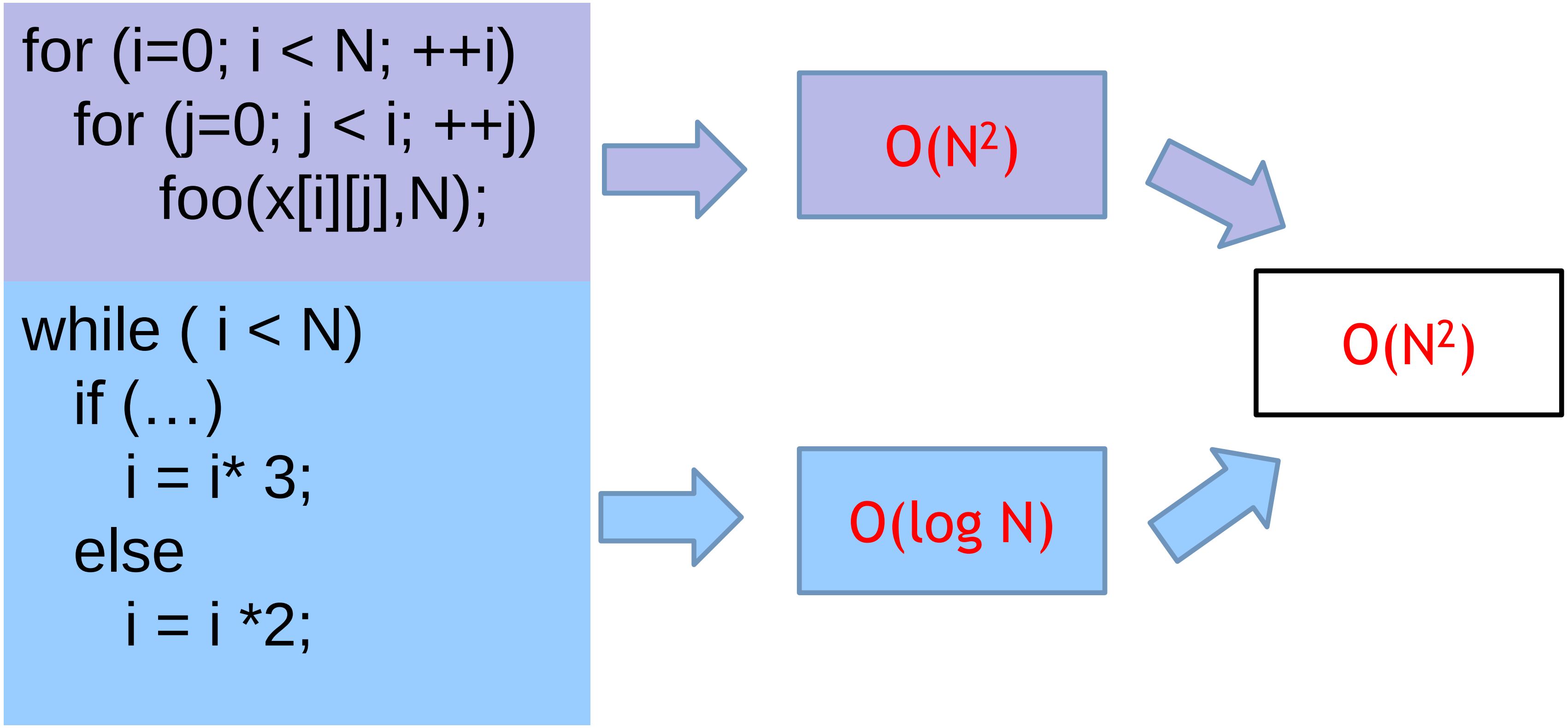
Runtime System with Performance Monitoring



$$1.2n^2mk + 3.5nm - 351$$

What function to fit?

- Static analysis of *complexity trends*.
  - Using invariant + ranking function generation.



```
for (i=0; i < N; ++i)
   for (j=0; j < i; ++j)
        foo(x[i][j],N);
```

```
while ( i < N)
   if (…)
      i = i* 3;
   else
      i = i *2;
```

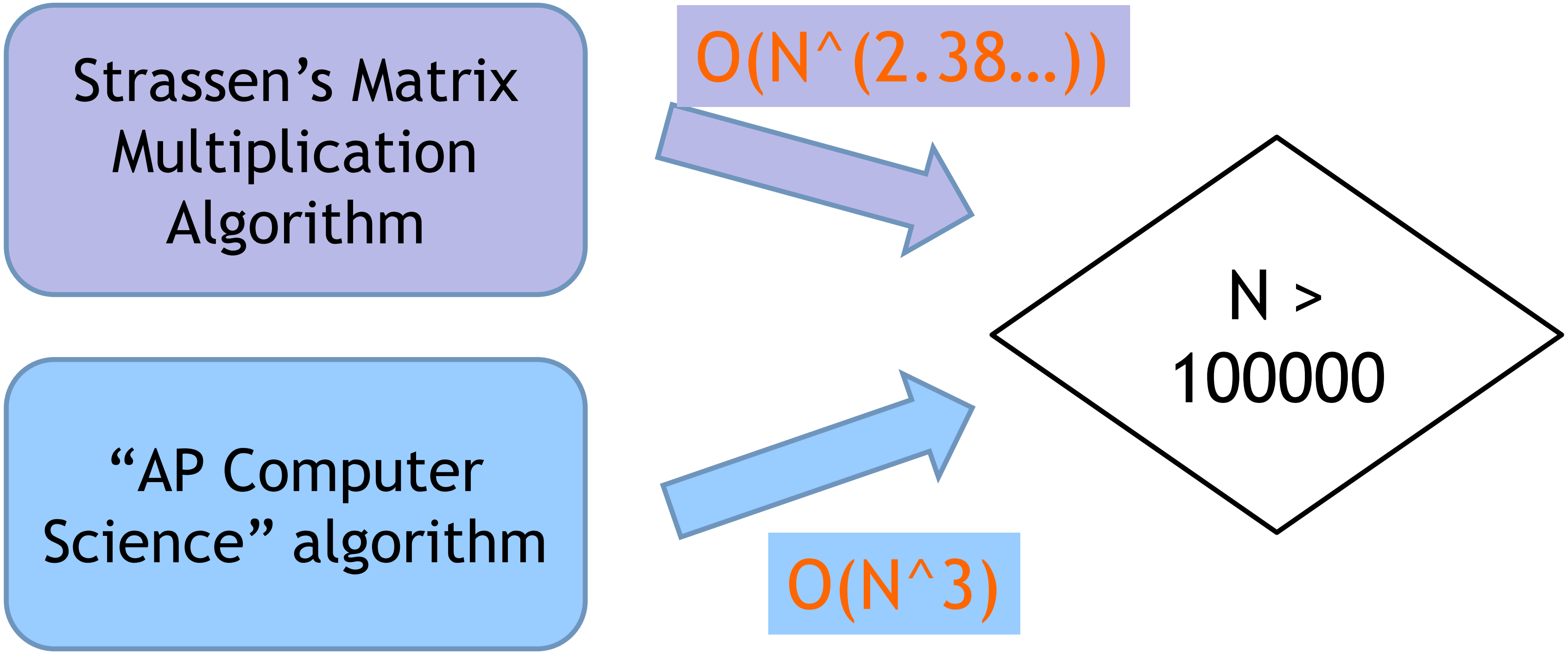
O(N^2)

O(log N)

O(N^2)

Application: Dynamic Algo. Selection

Dynamically select the best algorithm.



Strassen’s Matrix Multiplication Algorithm

O(N^(2.38…))

“AP Computer Science” algorithm

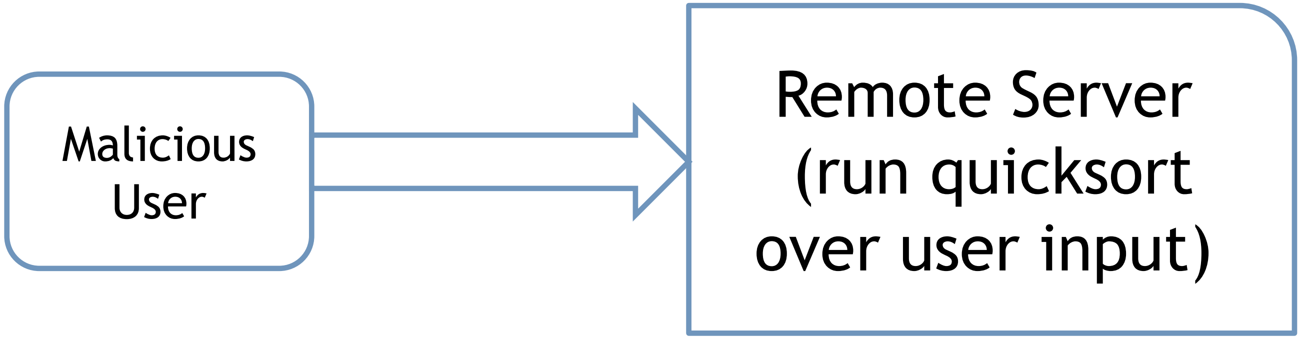
O(N^3)

N > 100000

Application: System Security

- *Denial of Service Attacks* can exploit high complexity worst case.

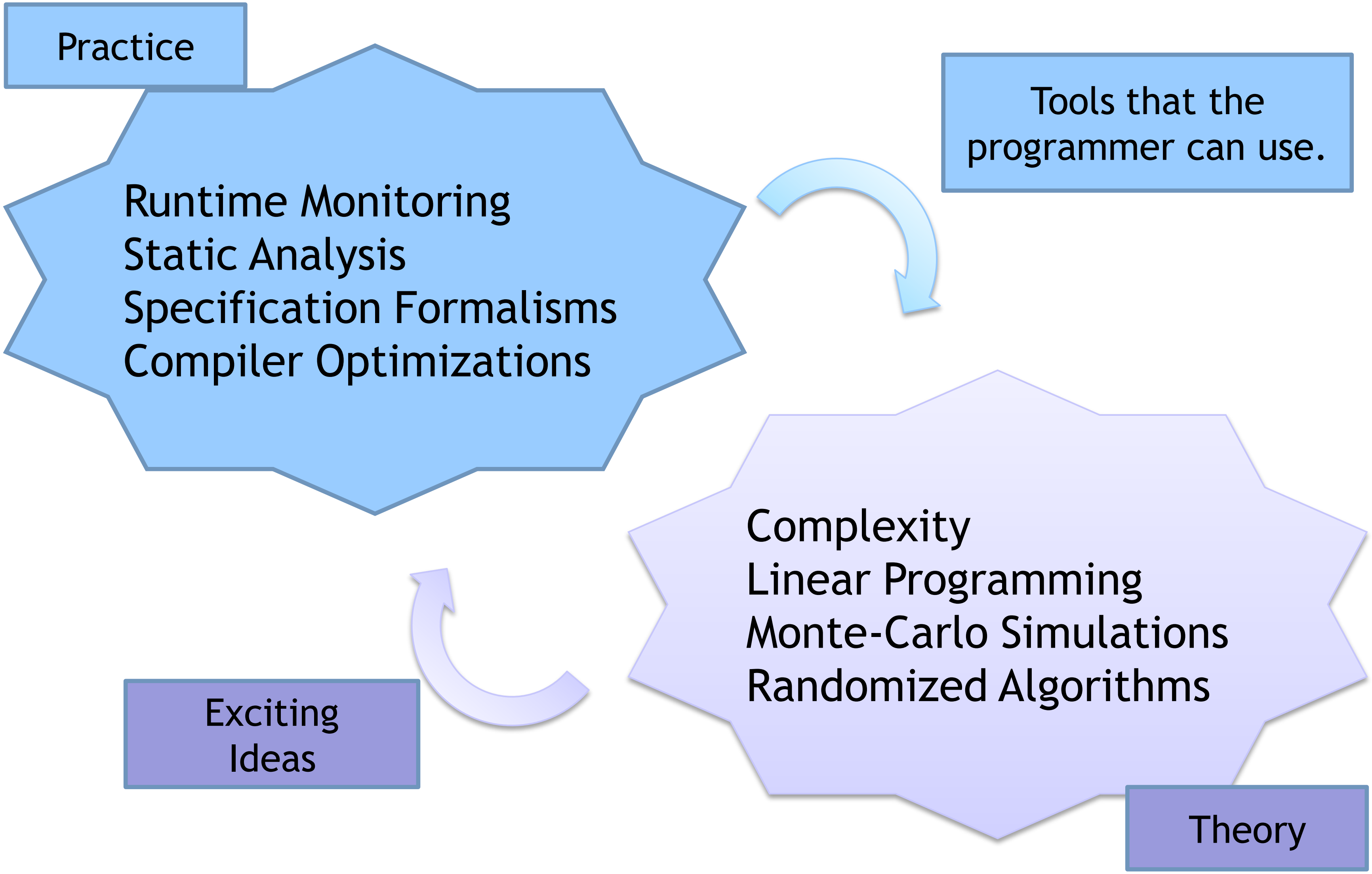
	Worst-Case	Average-Case
Quick Sort	$O(N^2)$	O(N log N)



Malicious User

Remote Server (run quicksort over user input)

Challenges and Opportunities



Practice

Runtime Monitoring
Static Analysis
Specification Formalisms
Compiler Optimizations

Tools that the programmer can use.

Complexity
Linear Programming
Monte-Carlo Simulations
Randomized Algorithms

Exciting Ideas

Theory

PL research at CU is ***successful***!

PLDI 2010 (2) Toronto, Canada 🥇

Mytkowicz, Diwan, Hauswirth, Sweeney. *Evaluating the Accuracy of Java Profilers*.

Khoo, Chang, Foster. *Mixing Type Checking and Symbolic Evaluation*.

POPL 2010 (2) Madrid, Spain 🥇

Harris, Sankaranarayanan, Ivancic, Gupta. *Program Analysis via Satisfiability Modulo Path Programs*.

Siek, Wadler. *Threesomes, With and Without Blame*.

ESOP 2010 Cyprus 🥈

Laviron, Chang, Rival. *Separating Shape Graphs*.

🥇 < 20% acceptance rate

🥈 < 27% acceptance rate

PL research at CU is ***successful***!

ASPLOS 2009 Washington, DC

Mytkowicz, Diwan, Hauswirth, Sweeney. *Producing wrong data without doing anything obviously wrong!*

OOPSLA 2009 (2) Orlando

von Dincklage, Diwan. *Optimizing programs with intended semantics.*

Mytkowicz, Coughlin, Diwan. *Inferred call-path profiling.*

ASE 2009 Auckland, New Zealand

Deshmukh, Emerson, Sankaranarayanan. *Refining the control structure of loops using static analysis.* ACM SIGSOFT Distinguished Paper.

CAV 2009 Grenoble, France

Kanade, Alur, Sankaranarayanan et al. *Generating and analyzing symbolic traces of Simulink/Stateflow models.*

PL research at CU is ***successful***!

ESOP 2009 York, UK

Siek, Garcia, Taha. *Exploring the design space of higher-order casts.*

TACAS 2009 York, UK

Kahlon, Sankaranarayanan, Gupta. *Semantic reduction of thread interleavings in concurrent programs.*

CC 2009 York, UK

Knights, Mytkowicz, Sweeney, Mozer, Diwan. *Blind optimization for exploiting architectural features.*

and more …

Papers ⇒ Travel + PhD

PL research at CU has *world-wide collaborations*!

MSR – WA
Google - CA
Apple - CA
IBM – NY
NEC– NJ
ENS - Paris
U. Edinburgh
U. Lugano

Collaborators ⇒ Internships and Jobs

PL students have *interned* at …

Microsoft® Research

Google™

IBM®

NEC

parc®
Palo Alto Research Center

After ***graduation***,
PL students have gone to …

IBM®

Università
della
Svizzera
italiana

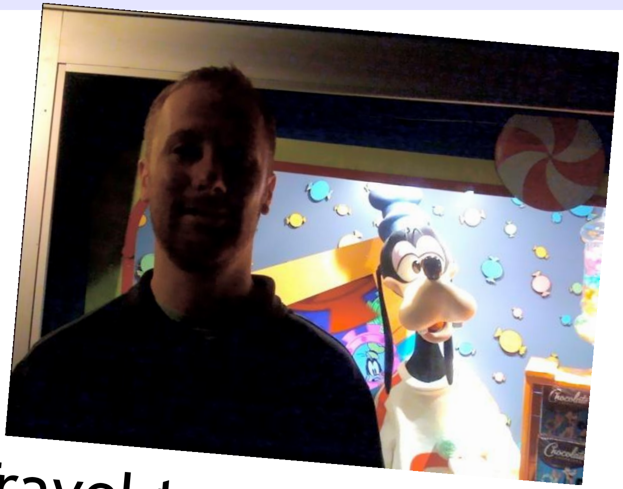
faculty

Google™

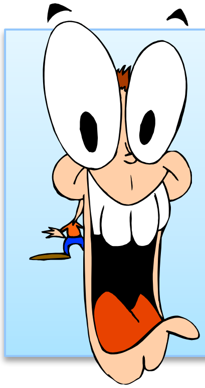
intel

The PL group has *fun* together!

Group meetings at the Boulder Tea House twice a month



Travel to conferences (Todd at OOPSLA'09)

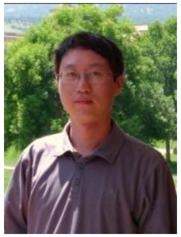


Our mentoring: Guide you to research that excites you!

Our group

PhD

Devin


Weiyu

Robert

Jonathan

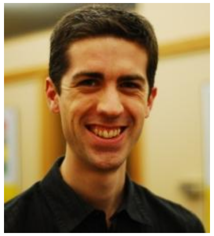
You?

You?

Postdoc

Todd

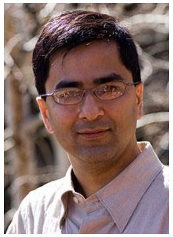
MS


Daniel

BS

James

Faculty


Amer

Jeremy

Evan

Sriram

Some of our other research projects

- Understanding performance
- Program metamorphosis
- Lightweight data collection
- Blind optimization
- Algorithmic optimizations
- Validating architectural simulators
- Using non-linear dynamics to understand computer systems
- Tools for teaching programming languages
- End-user program analysis
- Post-mortem analysis and error reporting
- Security policies for power-grids
- Analysis of web languages
- Modeling and validating building security policies
- Confident program analysis
- Checking low-level code
- Generic programming
- Meta-programming
- Gradual type checking
- High-level optimizations for memory efficiency
- Finding bugs in parallel programs
- Cyber-physical systems verification
- And soon projects created by you!