Supercomputing Power to the People

Arun Chauhan
Indiana University

Collaborators: At IU

Randall Bramley
Dennis Gannon
Joshua Hursey
Andrew Lumsdaine
Pooja Malpani
Daniel McFarlin
Beth Plale
Craig Shue

Collaborators: Outside IU

Rice University (TeleGen)

Bradley Broom
Jason Eckherdt
Rob Fowler
Ken Kennedy
Cheryl McCosh
John Mellor-Crummey

Ohio-State University (ParaM)

Muthu Baskaran
Aakash Dalwani
John Eaton
David Hudak
Ashok Krishnamurthy
Rajkiran Panuganti
P. Sadayappan

Programming Languages: A Buddhist View

Programming Languages: A Buddhist View

Programming Languages = ***Math***

Programming Languages: A Buddhist View

Programming Languages = ***Math***

Programming Languages = ***Evil***

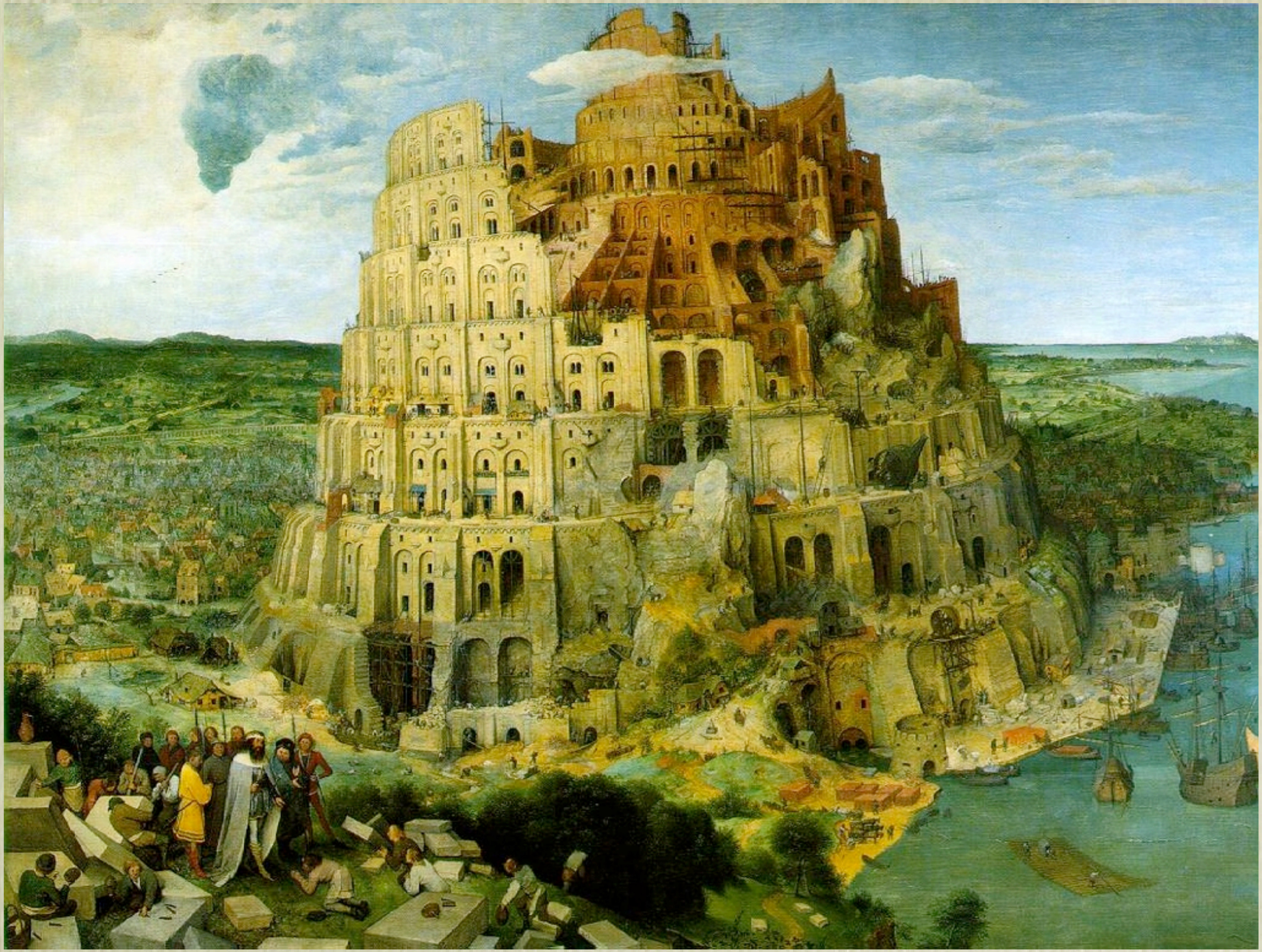
Programming Languages: A Buddhist View

Programming Languages = ***Math***

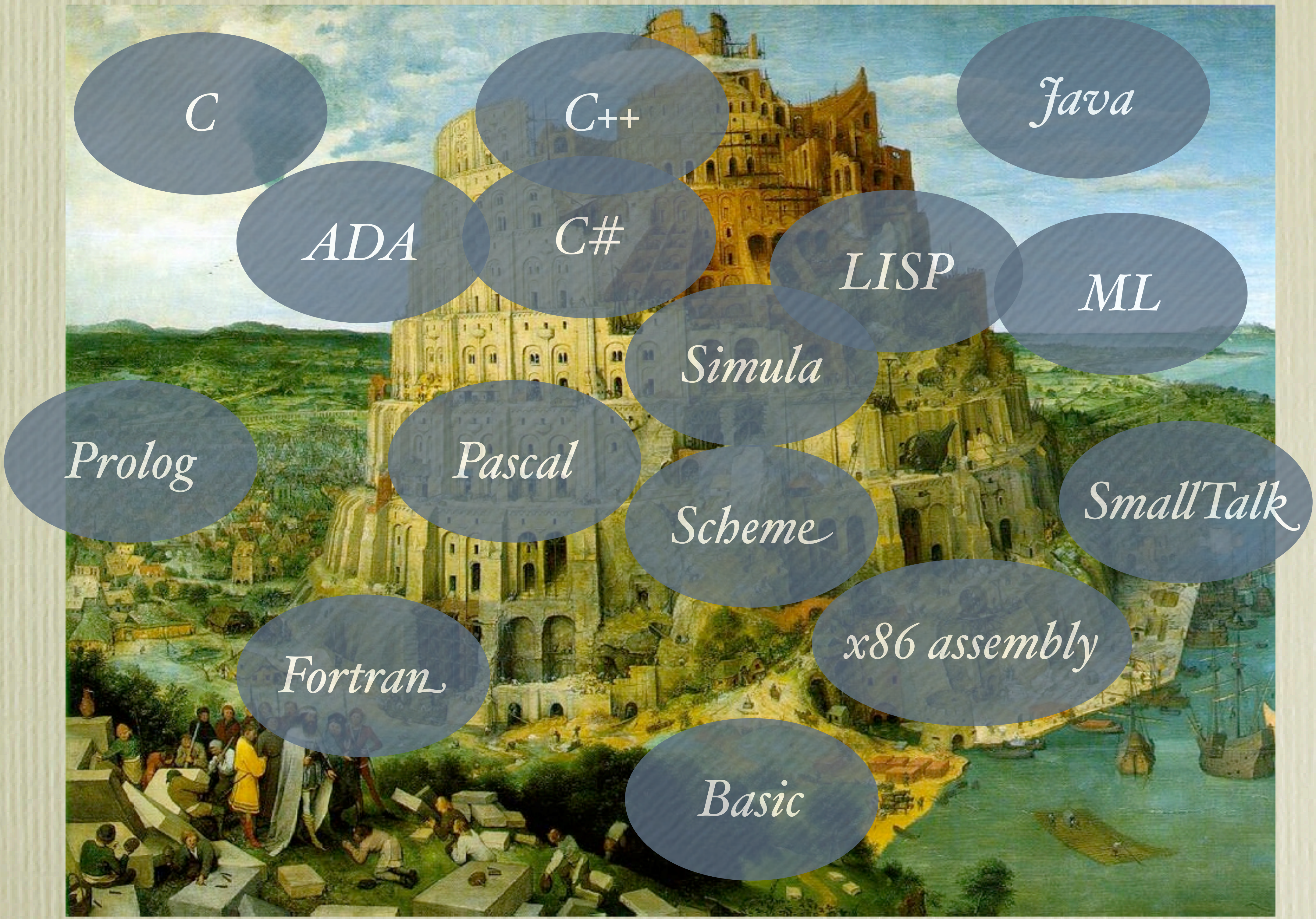
Programming Languages = ***Evil***

- Interface between humans and computers
- Enablers of complex communication with computers

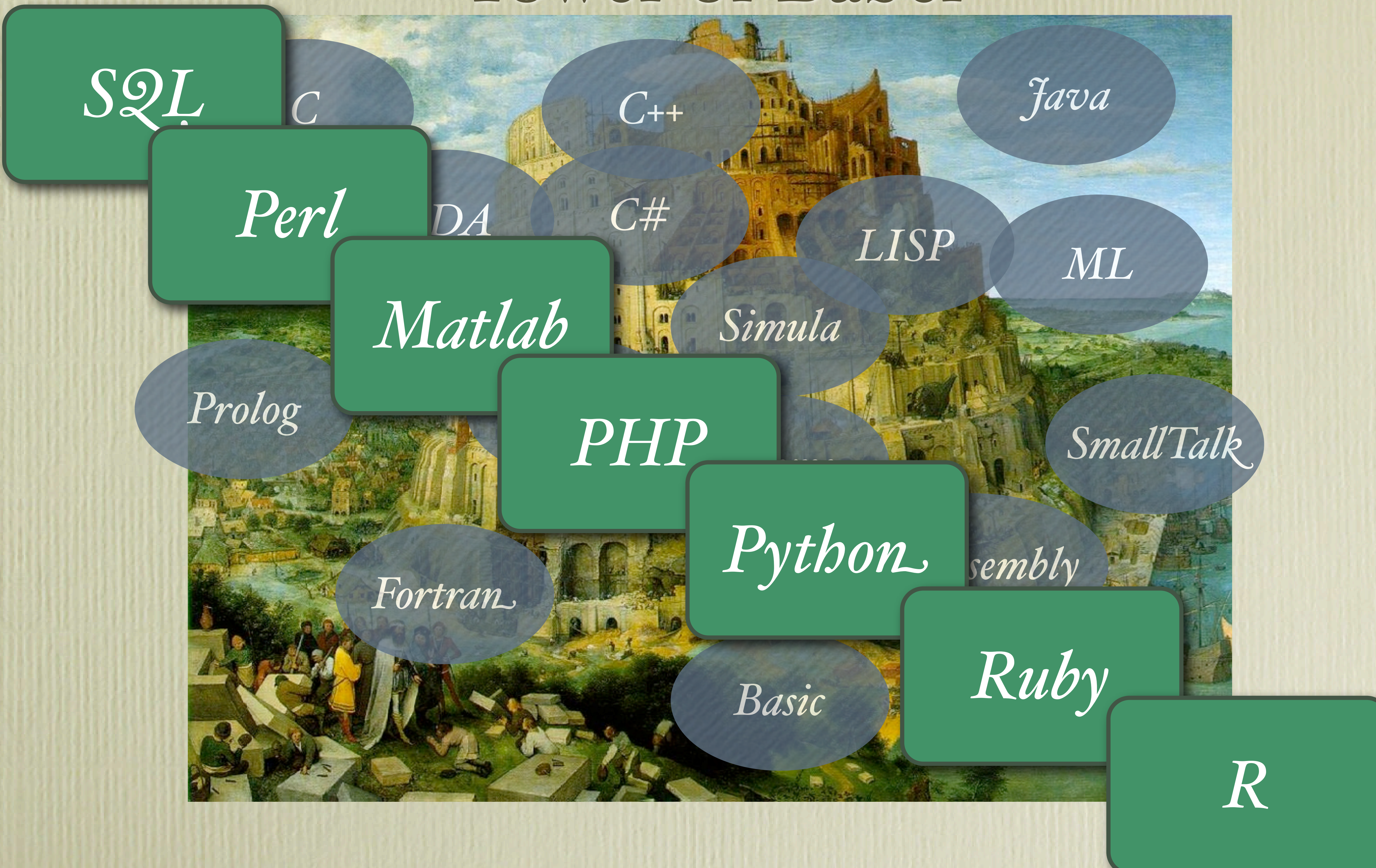
Tower of Babel



Tower of Babel

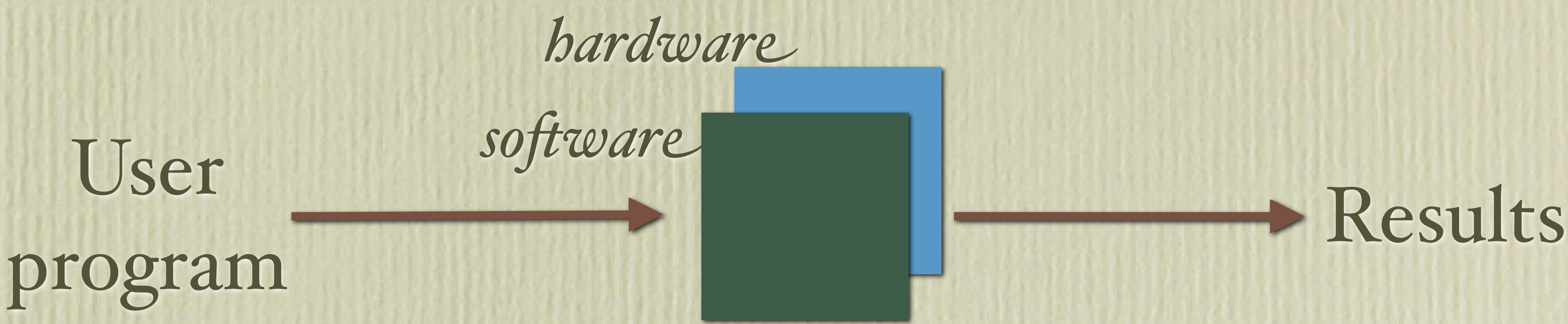


Tower of Babel

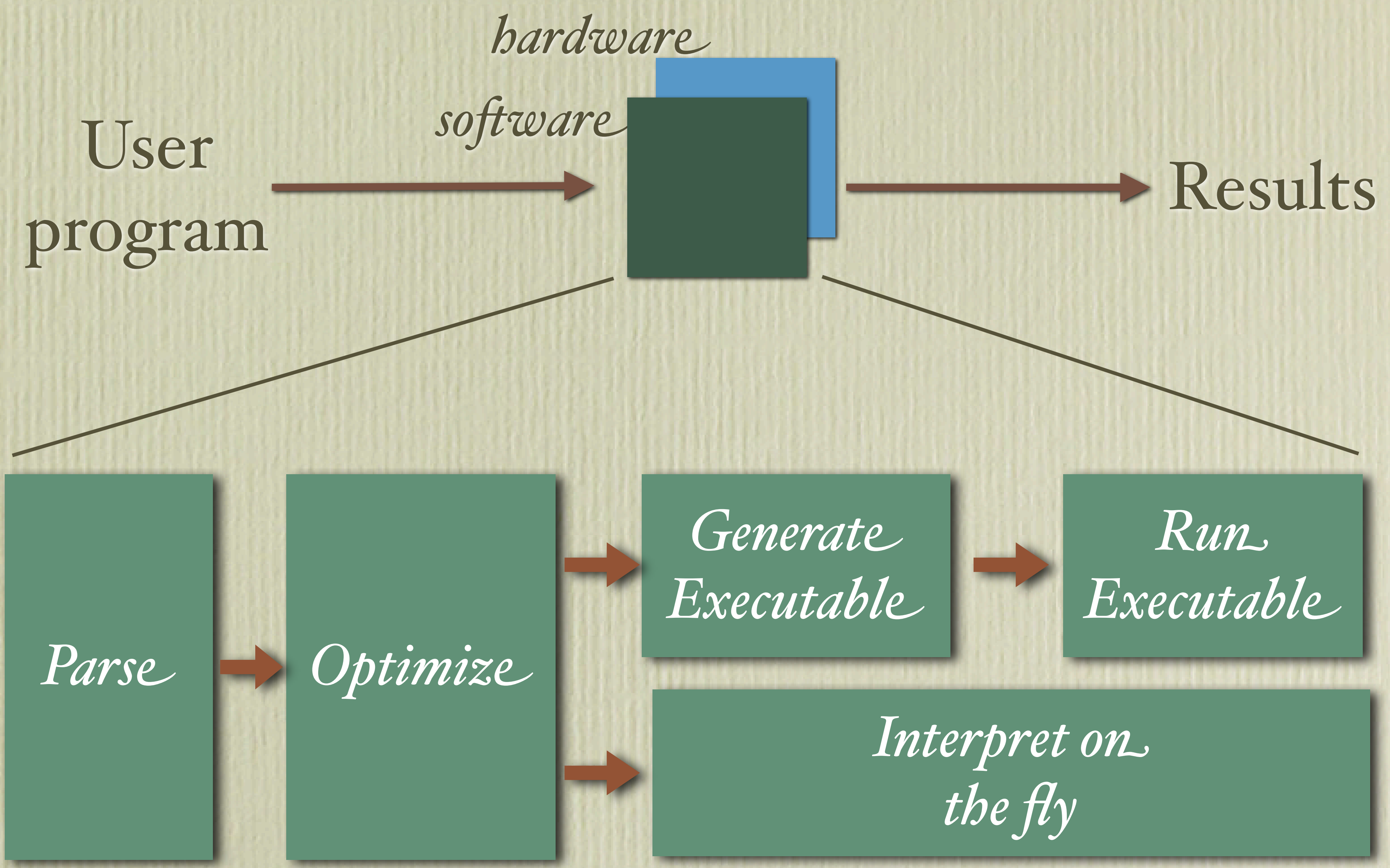


Who Speaks Computerese?

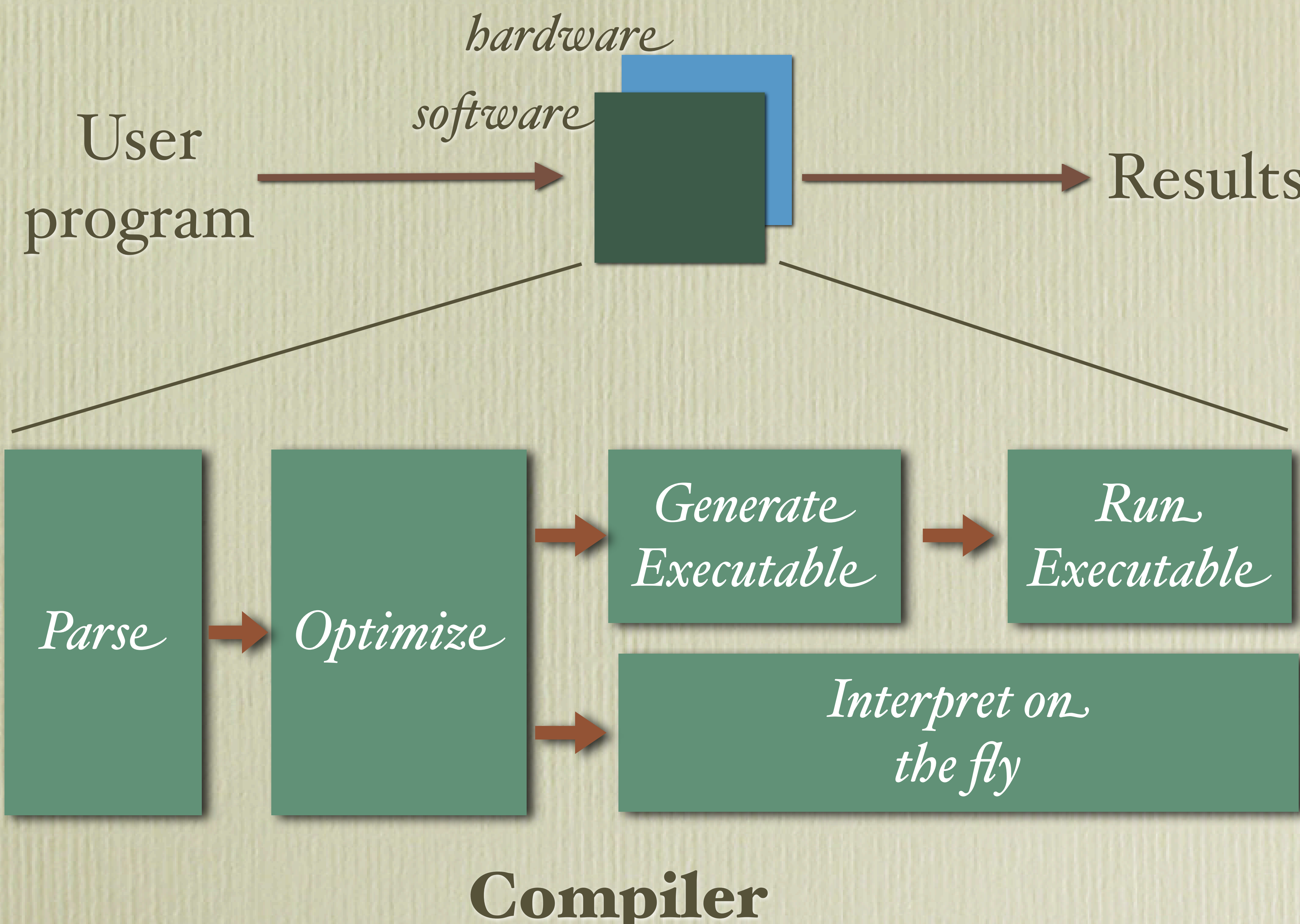
Who Speaks Computerese?



Who Speaks Computerese?



Who Speaks Computerese?



Compilers have been Around

"[T]he next revolution in programming will take place only when *both* of the following requirements have been met: (a) a new kind of programming language, far more powerful than those of today, has been developed and (b) a technique has been found for executing its programs at not much greater cost than that of today's programs."

John Backus
"The History of Fortran I, II, and III"
ACM SIGPLAN Notices, Vol. 13, No. 8, August 1978

The Productivity Problem

INFORMATION PROCESSING TECHNOLOGY OFFICE

PROGRAMS

Search for

Go

Personnel
Programs
Solicitations
Briefings
News/Information
DARPA Offices
Contact Us
IPTO Home

IPTO Home Page >> Programs >> High Productivity Computing Systems (HPCS)

High Productivity Computing Systems (HPCS)

HPCS

Mr. Robert Graybill

Overview
Mission
Vision
Background

Supplemental
Meetings

Technical Program
Objectives
Challenges
Program Plan
Assessment
Goals

Mission:

- Provide a focused research and development program, creating new generations of high end programming environments, software tools, architectures, and hardware components in order to realize a new vision of high end computing, high productivity computing systems (HPCS). Address the issues of low efficiency, scalability, software tools and environments, and growing physical constraints.
- Fill the high end computing gap between today's late 80's based technology High Performance Computing (HPCs) and the promise of quantum computing.
- Provide economically viable high productivity computing systems for the national security and industrial user communities with the following design attributes in the latter part of this decade:
  - **Performance:** Improve the computational efficiency and performance of critical national security applications.
  - **Programmability:** Reduce cost and time of developing HPCS application solutions.
  - **Portability:** Insulate research and operational HPCS application software from system specifics.
  - **Robustness:** Deliver improved reliability to

The Problem

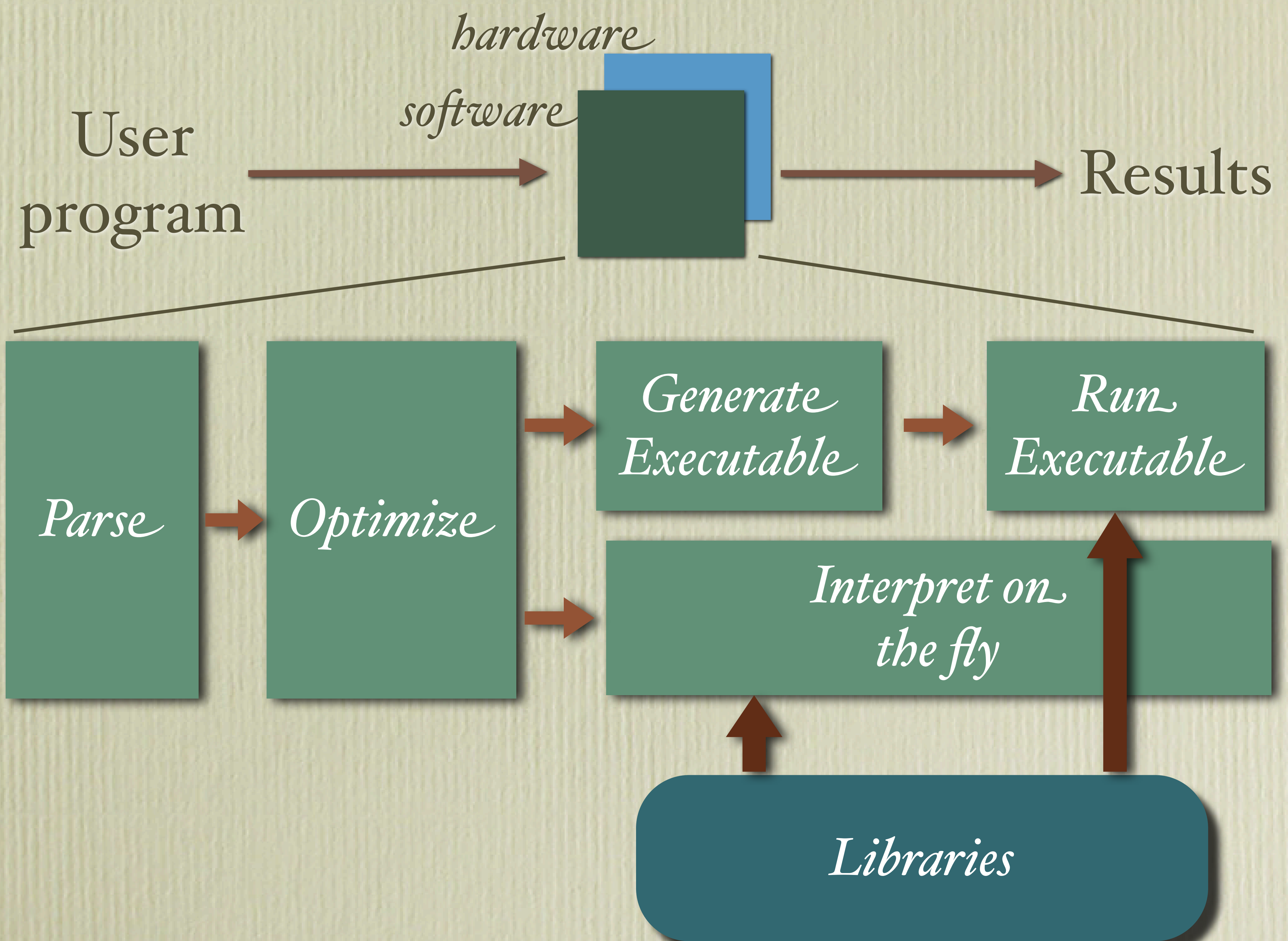
```
function z = my_add (x, y)
t = 2*x;
z = t + y;
```

The Problem

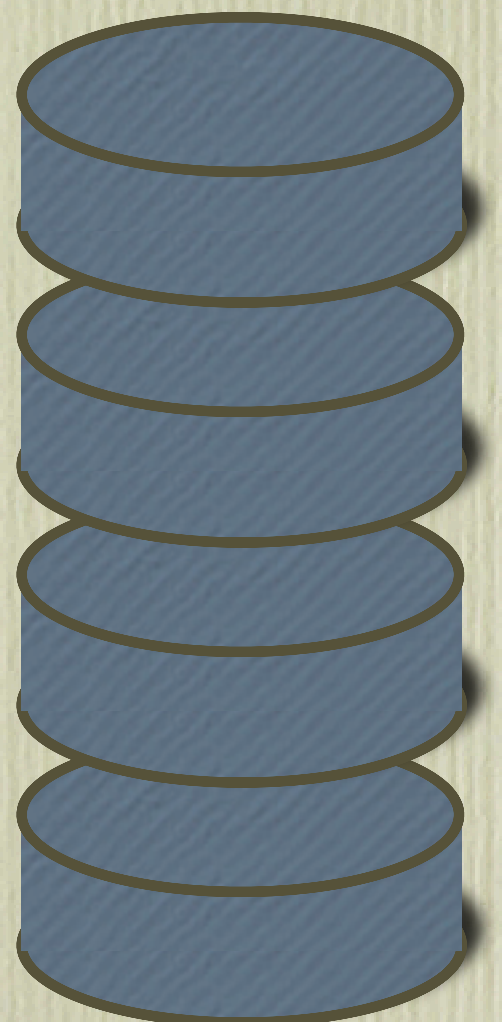
```
function z = my_add (x, y)
t = 2*x;
z = t + y;
```

```
mxArray* Mmcc_my_add (mxArray *x, mxArray *y)
{
  ...
  mxArray *t = NULL;
  mxArray *z = NULL;
  mlfAssign (&t, mclMtimes(_mxarray1_, mclVv(x, "x"));
  mlfAssign (&z, mclPlus(mclVv(t, "t"), mclVv(y, "y"));
  mxDestroyArray(t);
  ...
  return z;
}
```

Role of Libraries



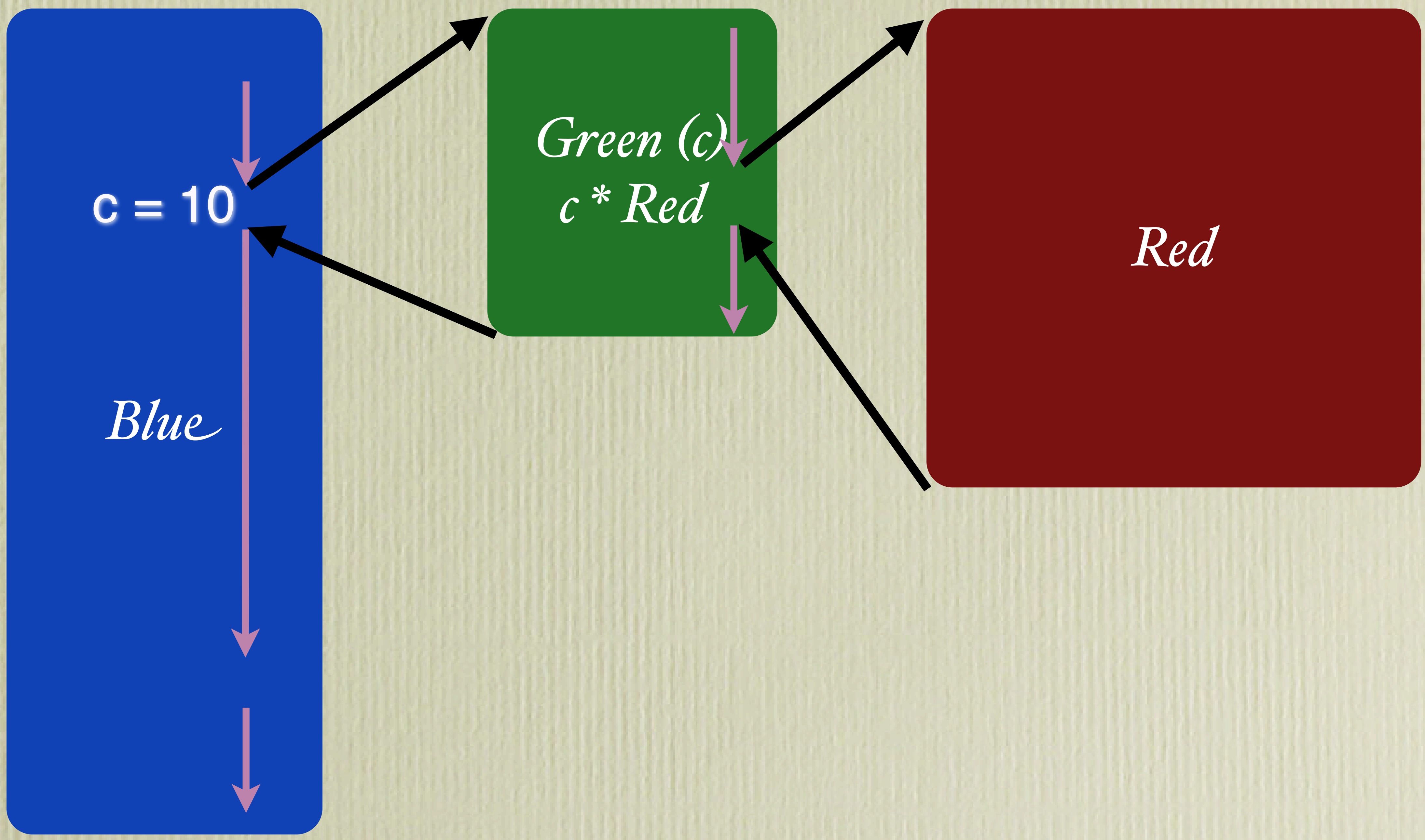
Hierarchy of Libraries



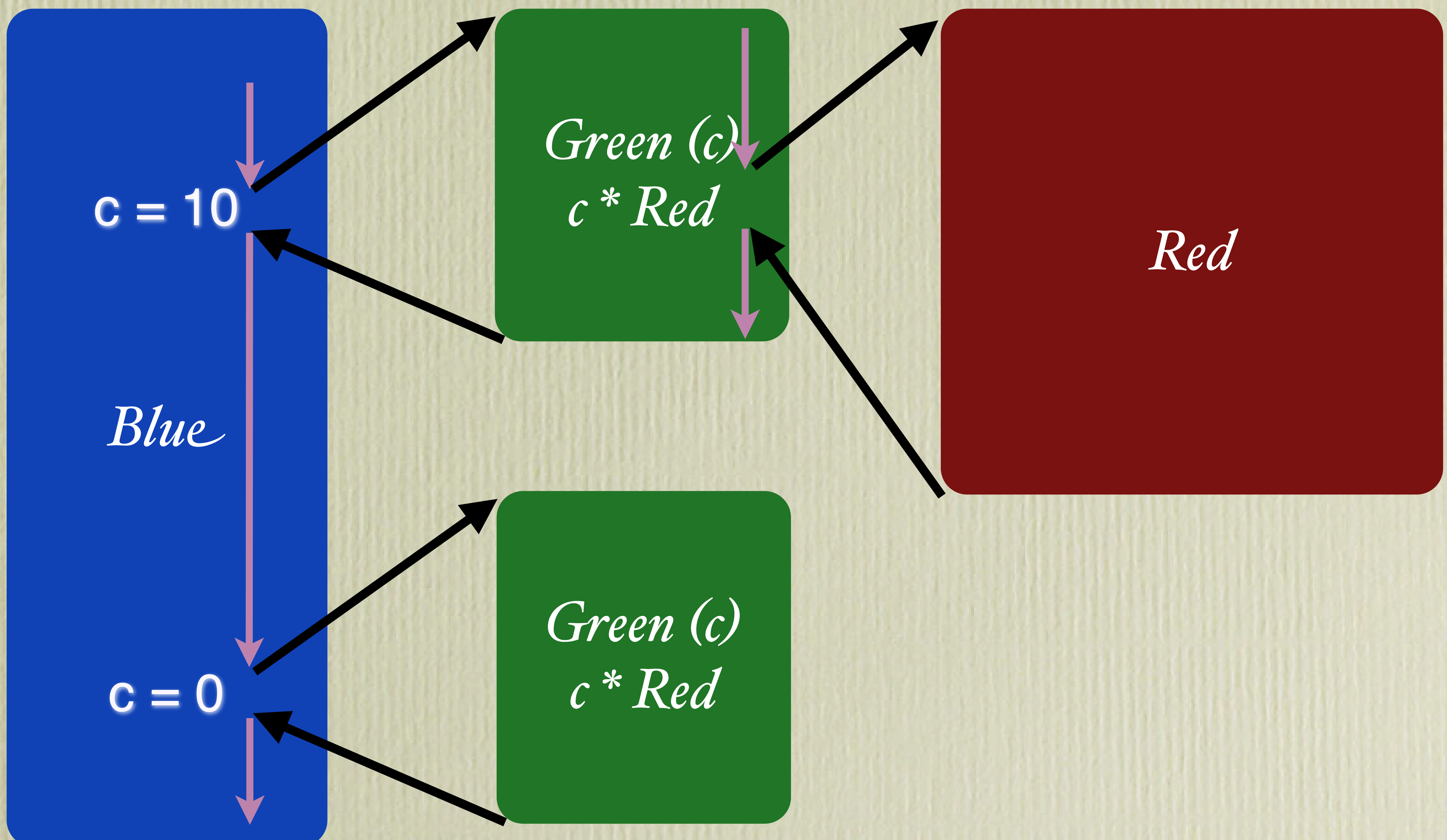
Two Ways to Handle Libraries

- Black Boxes
  - Separately compiled
  - Almost no inter-procedural optimizations
- Whole Program Compilation
  - High level of inter-procedural optimizations
  - Inefficient: Octave library has 300,00 lines

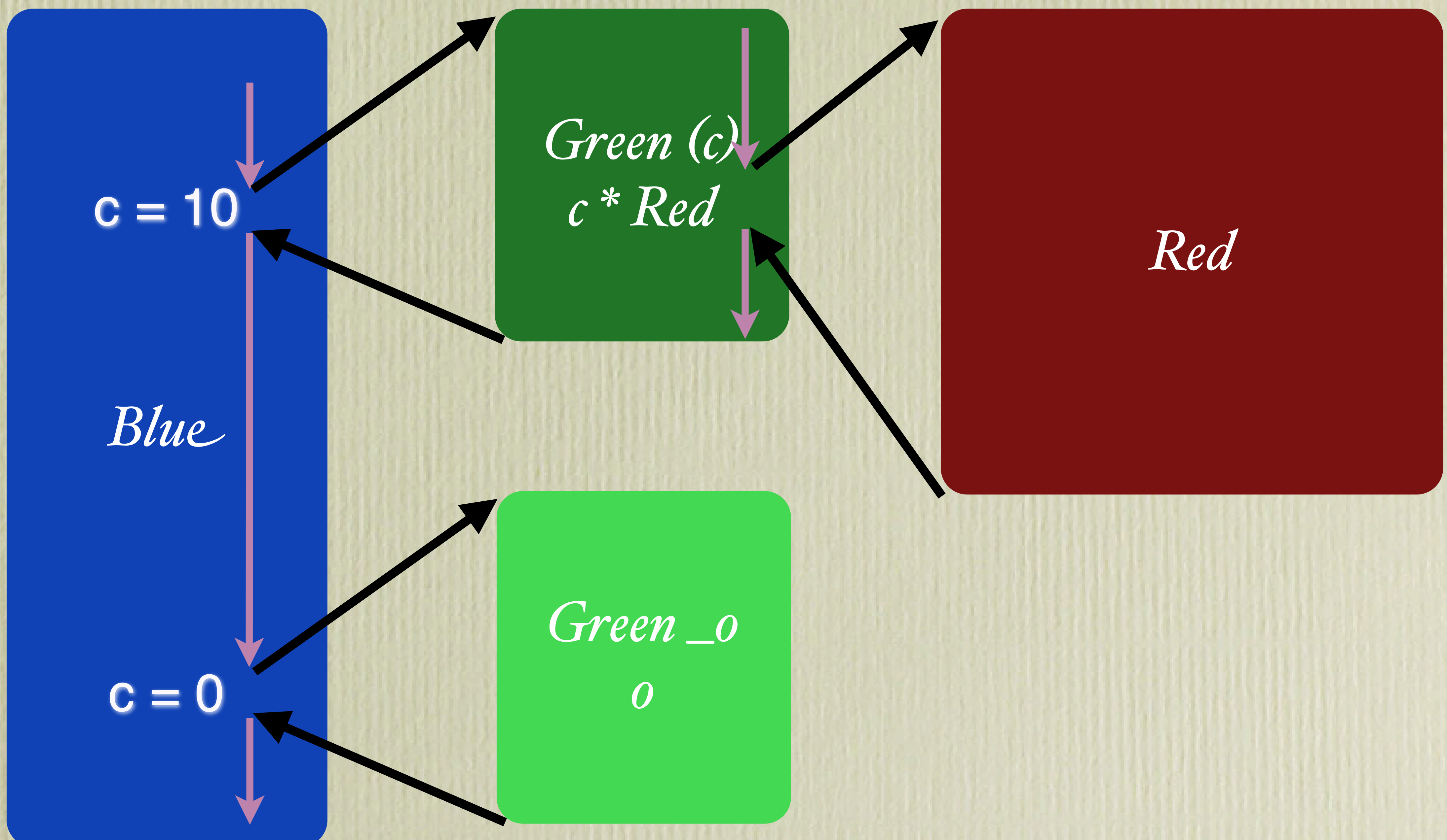
Optimization Example



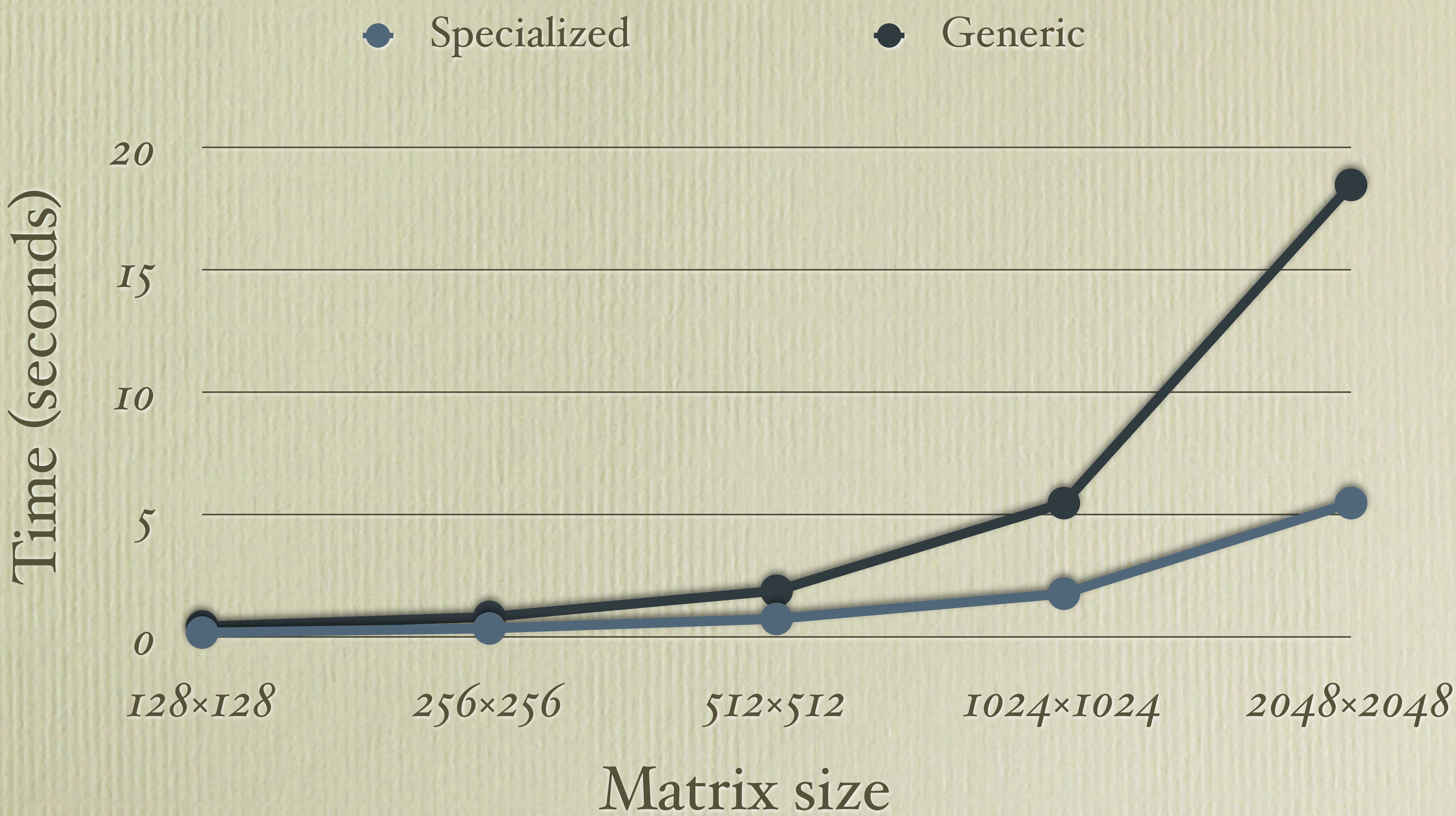
Optimization Example



Specialized Green

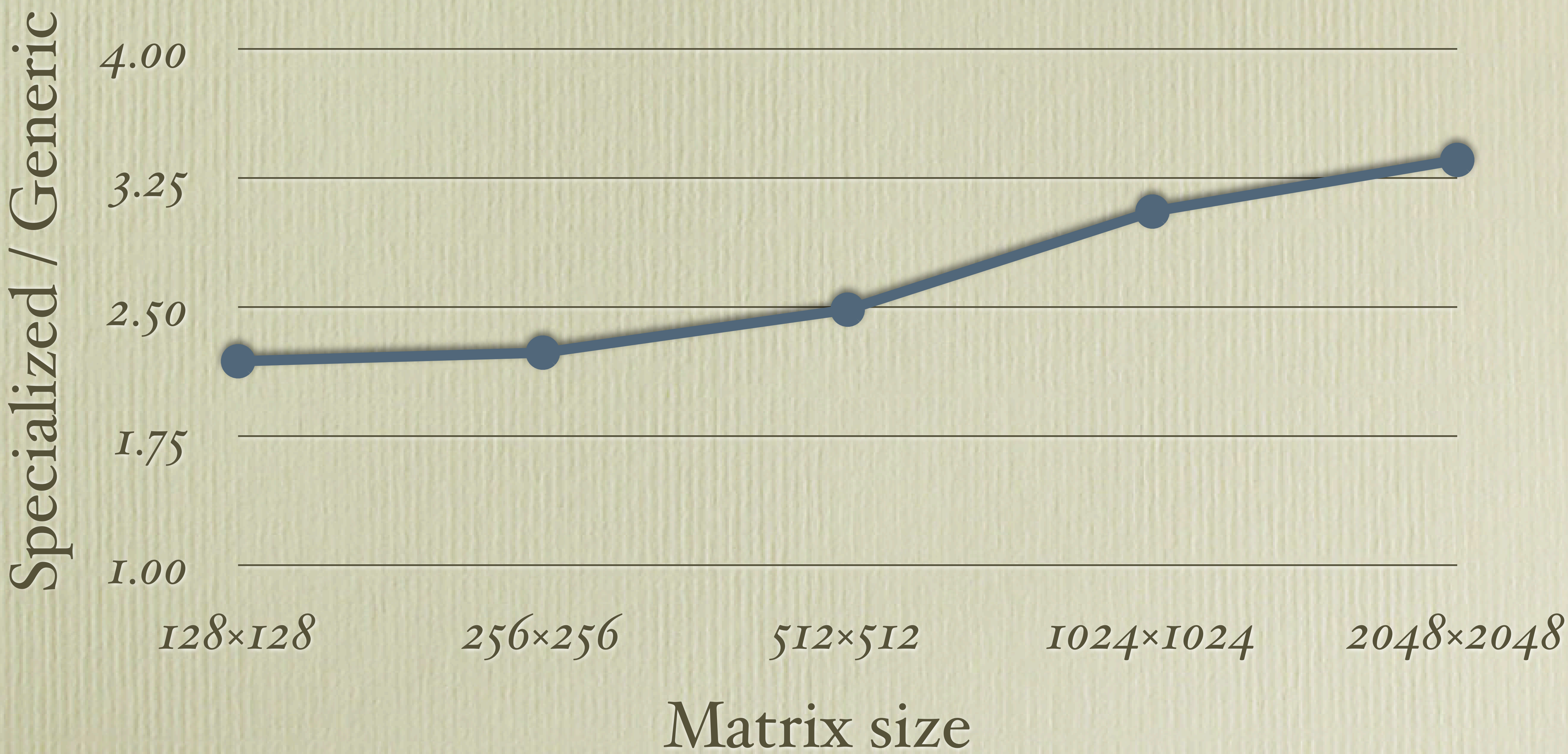


Specialization Pays: Arnoldi



Study by Daniel McFarlin

Specialization Pays: Arnoldi



Strategy

- Pre-compile libraries, but specialize for expected contexts speculatively
  - Eliminate the need for inter-procedural optimization
  - Retain the benefit of separate compilation

Experiments

"It is a capital mistake to theorize before one has data. Insensibly, one begins to twist facts to suit theories instead of theories to suit facts."

Sir Arthur Conon Doyle
The Scandal in Bohemia

Findings

- Identified high pay-off optimizations
  - common sub-expression elimination
  - constant propagation
  - loop vectorization, etc.
- Two new inter-procedural optimizations
  - procedure strength reduction
  - procedure vectorization
- Patterns of library usage

ICS 2001, IJPPP 2002

Patterns of Usage

```
function [s, r, j hist] = min sr1 (xt, h, m, alpha)
    ...
    while ~ok
        ...
        invsr = change_form_inv (sr0, h, m, low rp);
        big f = change_form (xt-invsr, h, m);
        ...
        while iter s < 3*m
            ...
            invdr0 = change_form_inv (sr0, h, m, low rp);
            sssdr = change_form (invdr0, h, m);
            ...
        end
        ...
        invsr = change_form_inv (sr0, h, m, low rp);
        big f = change_form (xt-invsr, h, m);
        ...
        while iter r < n1*n2
            ...
            invdr0 = change_form_inv (sr0, h, m, low rp);
            sssdr = change_form (invdr0, h, m);
            ...
        end
        ...
    end
```

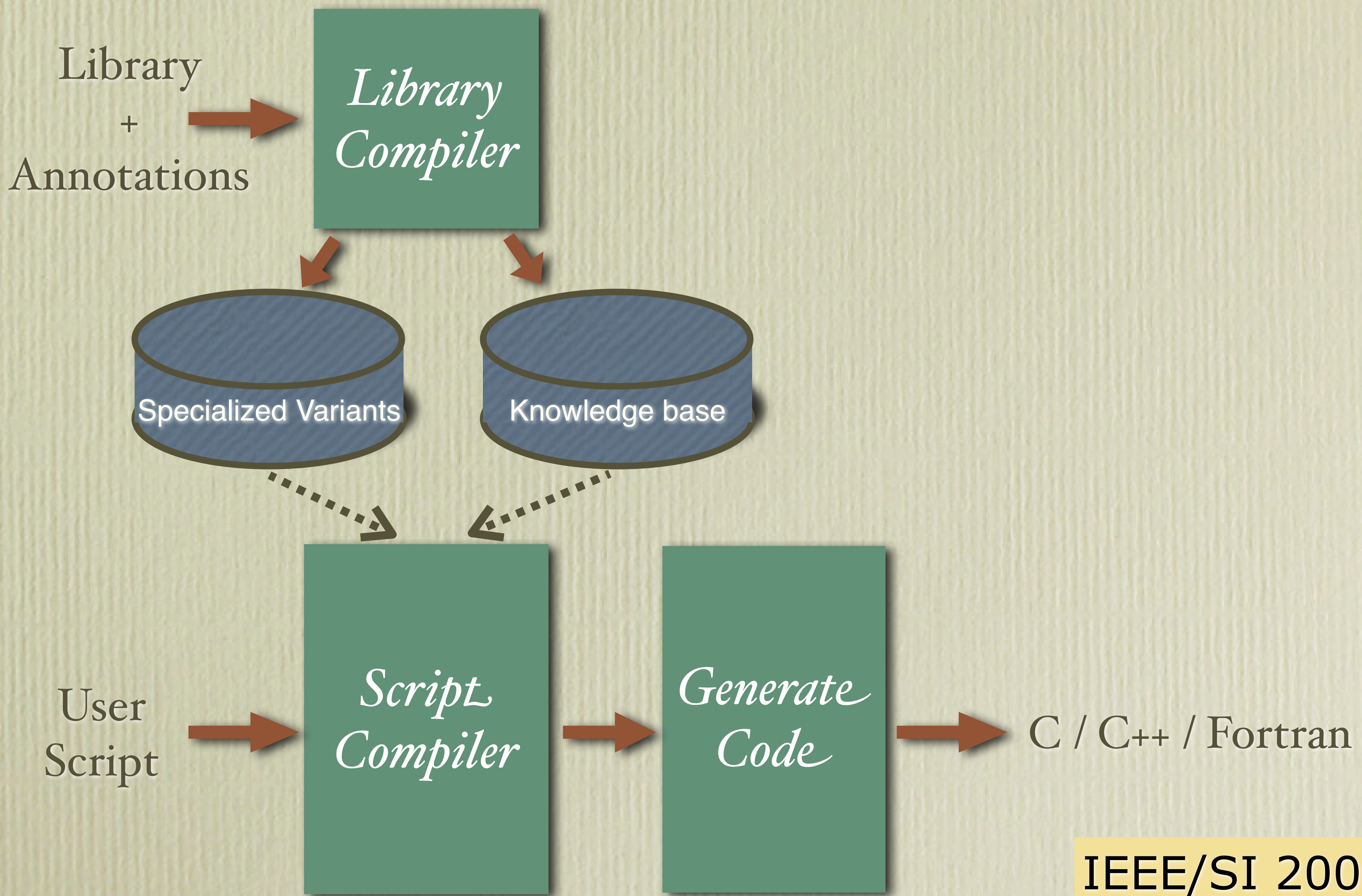
Beyond Patterns

- Library identities
  - Compilers already use algebraic identities
    - $x + 0 = x$
    - $x * 0 = 0$
    - $2*x = x + x$, etc.
- Constraints on arguments
  - E.g., co-varying argument types or values
- Relationship between inputs and outputs and side-effects

Strategy

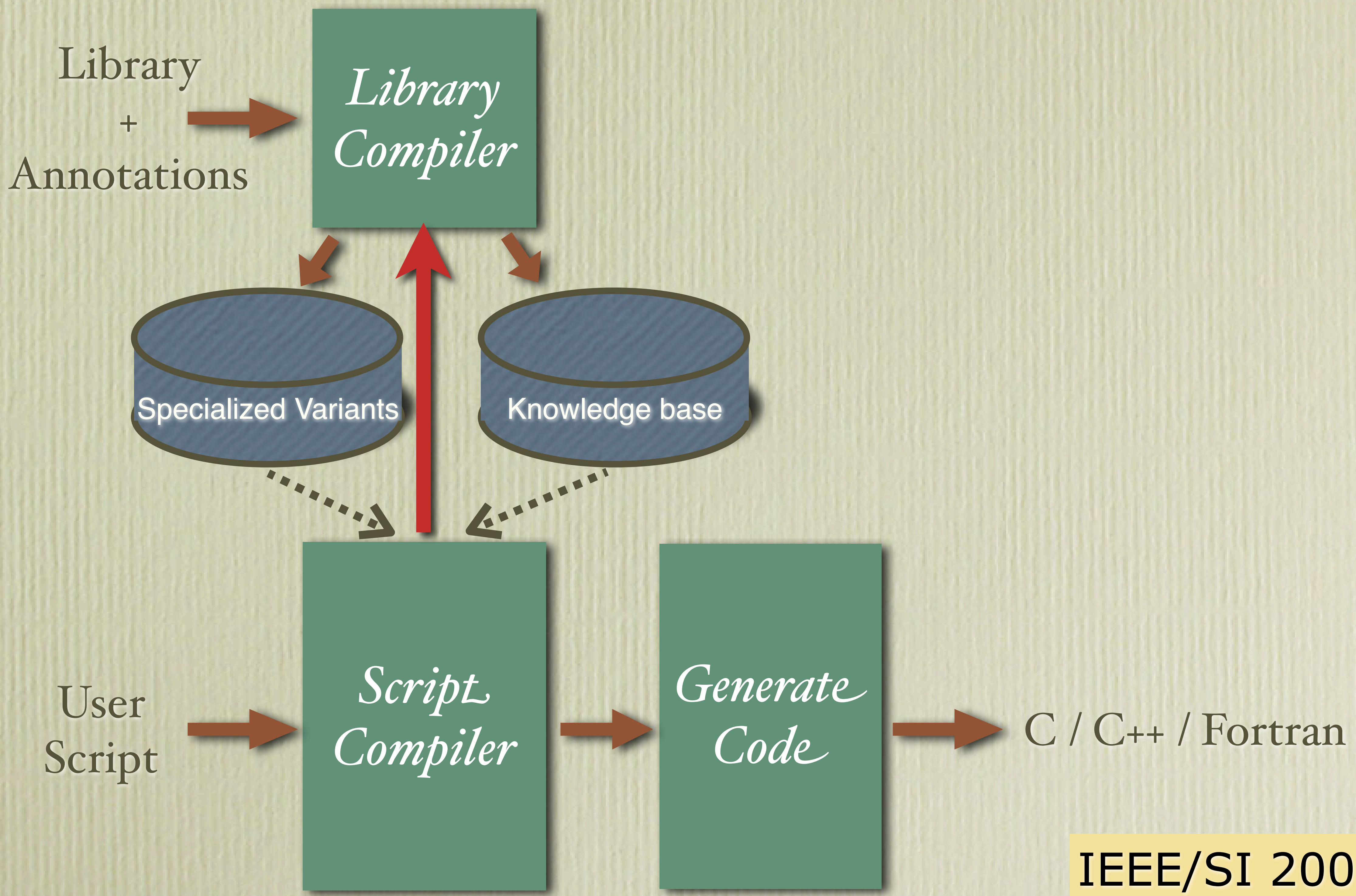
- Pre-compile libraries, but optimize for expected contexts
  - Eliminate the need for inter-procedural optimization
  - Retain the benefit of separate compilation
- Utilize library properties beyond those captured in the code
  - Make use of domain experts' knowledge
  - Utilize semantic knowledge

Telescoping Languages



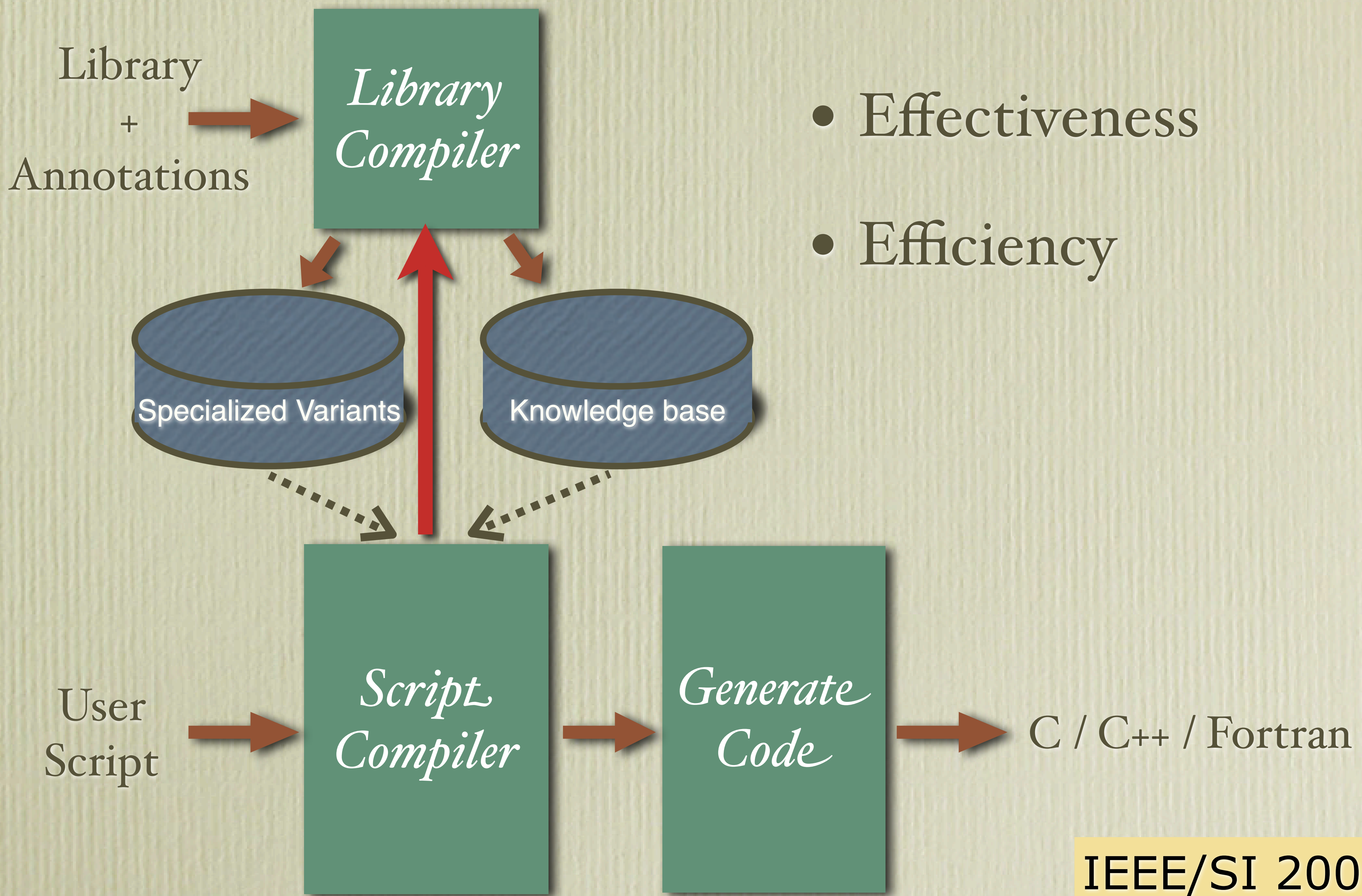
IEEE/SI 2005

Telescoping Languages



IEEE/SI 2005

Telescoping Languages



- Effectiveness
- Efficiency

IEEE/SI 2005

Challenges

- Identifying relevant library properties
  - guided by high-payoff optimizations
- Describing library properties
  - extended type system
  - annotation language
- Building and utilizing the knowledge base
  - cost model
  - “instruction selection”

High-Performance = Parallel

Is Parallel Computing Important?

"It doesn't make good business sense for us to undertake fundamental changes in MATLAB's architecture. There are not enough potential customers with parallel machines."

Cleve Moler
Co-founder of MathWorks, 1995

Is Parallel Computing Important?

Home | Select Country | Contact Us | Store

The MathWorks
Accelerating the pace of engineering and science

Products & Services | Industries | Academia | Support | User Commu

Product Overview

- Description
- Function List
- Demos and Webinars
- Related Products
- System Requirements
- Latest Features

Support & Training

- Technical Support
- Documentation
- Downloads & Trials

Other Resources

- Technical Literature
- User Stories

Tell us about your computing cluster

Distributed Computing Toolbox 2.0.1

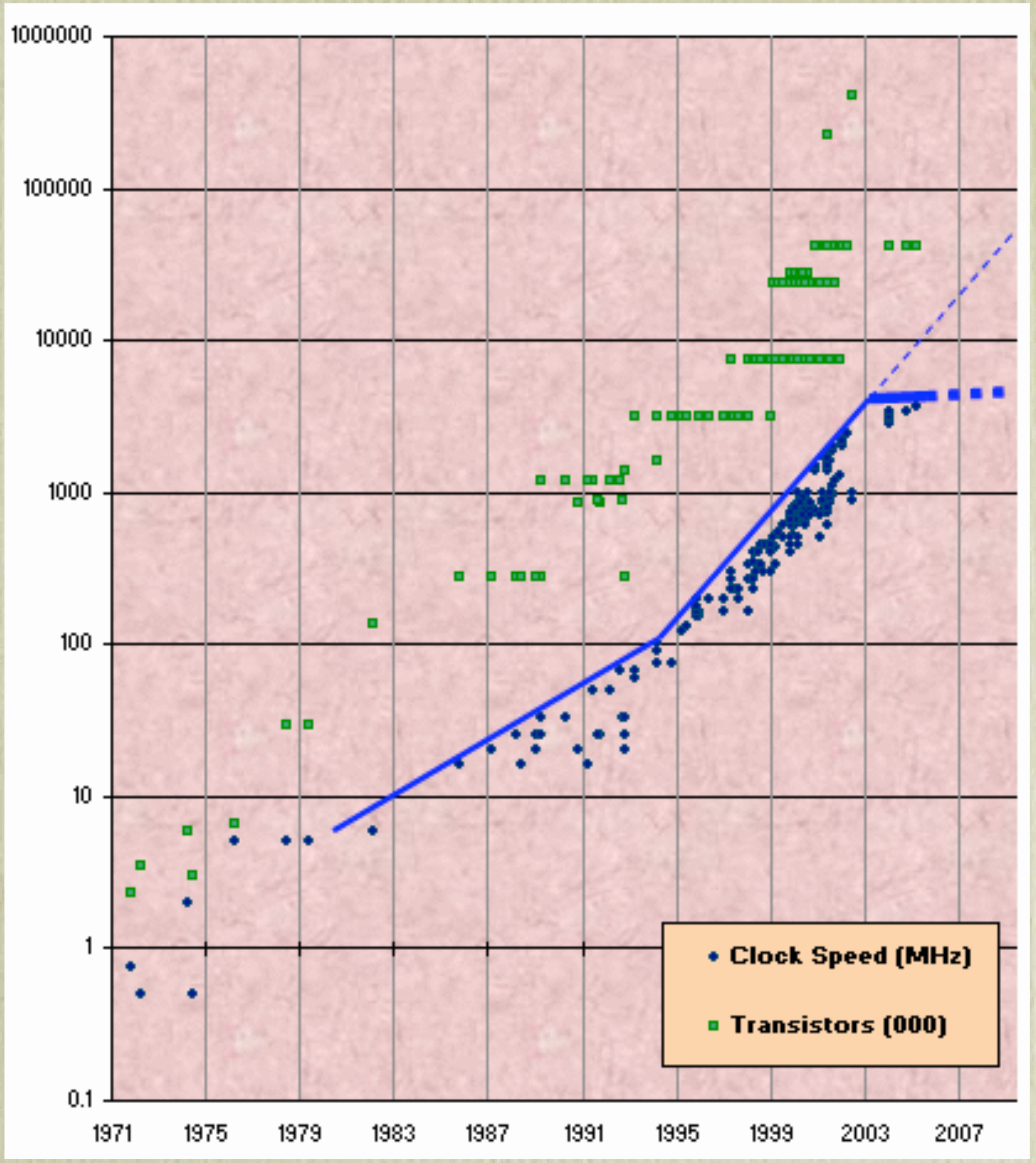
Develop distributed MATLAB applications for execution on a computer cluster

The Distributed Computing Toolbox and the MATLAB Distributed Computing Engine enable you to develop distributed MATLAB applications and execute them in a cluster of computers without leaving your development environment. You can prototype applications in MATLAB and use the Distributed Computing Toolbox functions to define independent or interdependent tasks. Algorithms that require interdependent tasks use the Message Passing Interface (MPI)-based functions provided. The MATLAB Distributed Computing Engine schedules and evaluates tasks on multiple remote MATLAB sessions, reducing execution time compared to running in a single MATLAB session.

- Introduction and Key Features
- Working with the Distributed Computing Toolbox and the MATLAB

Pervasive Parallelism

Herb Sutter, *The Free Lunch is Over*, Dr Dobbs Journal, March 2005

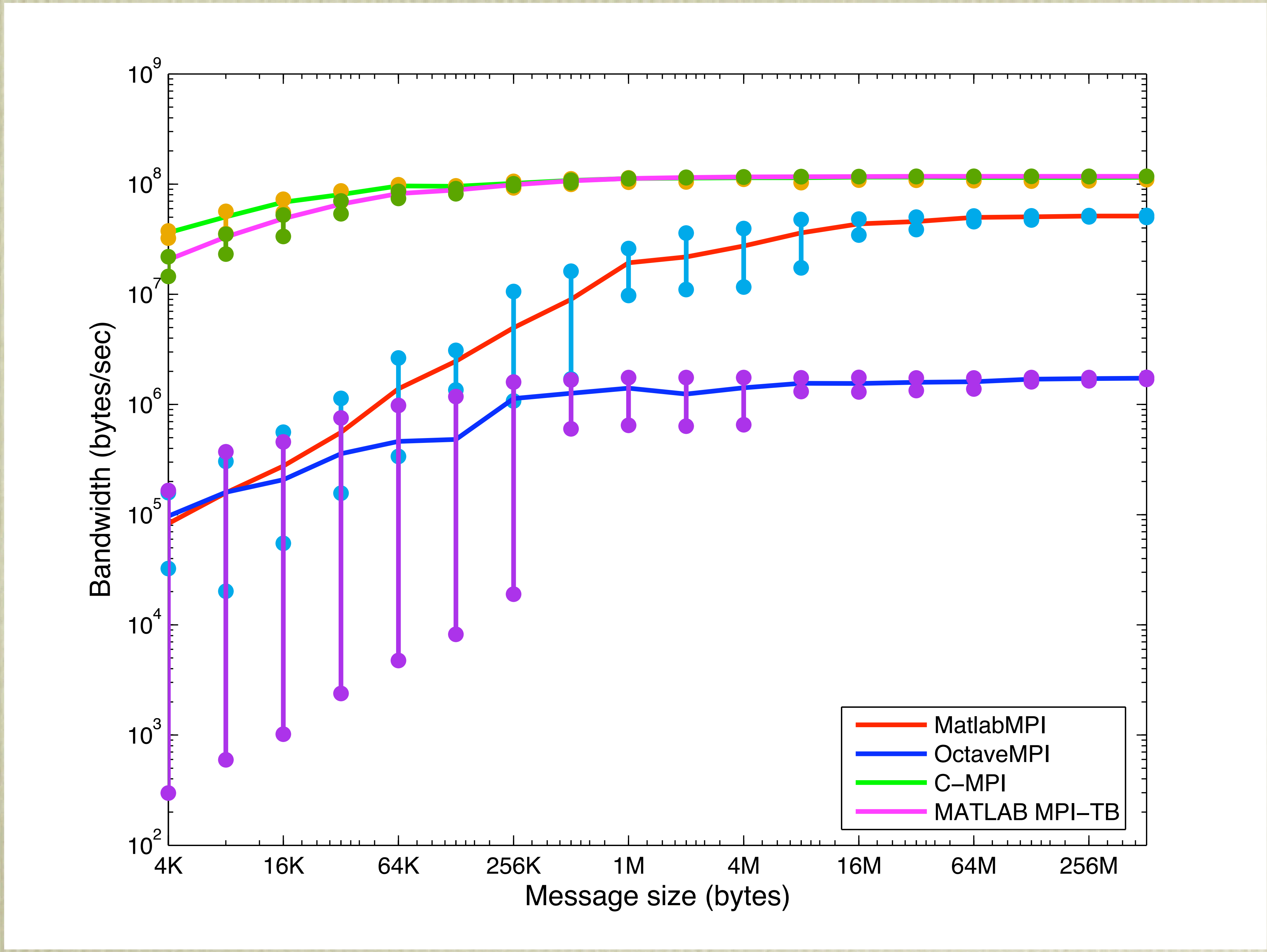


Why Parallelization?

- Improve running time
- Run larger problems
- Compensate for poor uniprocessor performance
- Even MathWorks has come around!

ParaM (*pronounced* per.um) *adj.* Super (in Sanskrit).
Also means **Para**llel **M**atlab

MPI?

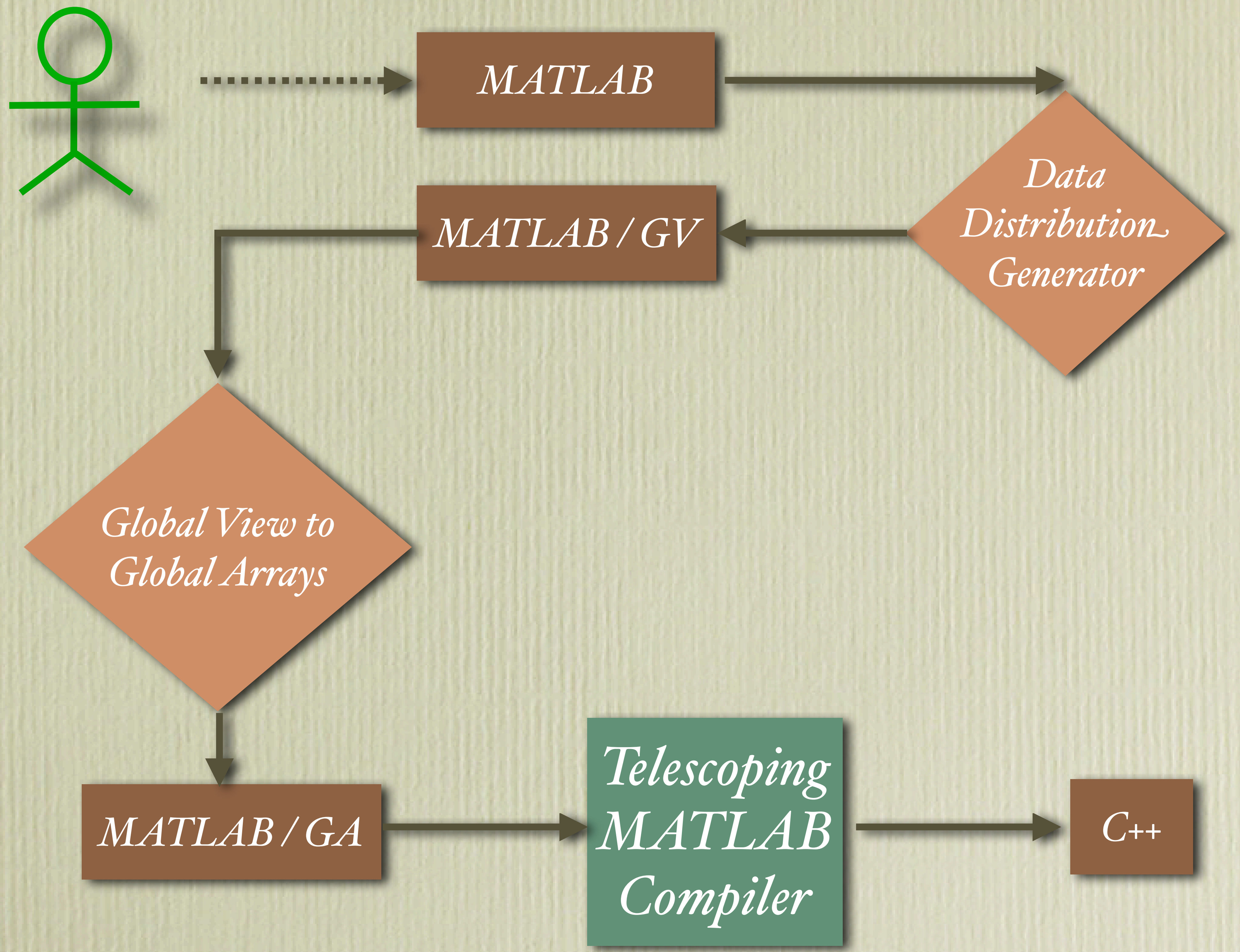


Tech Report IUCS 631, 2006

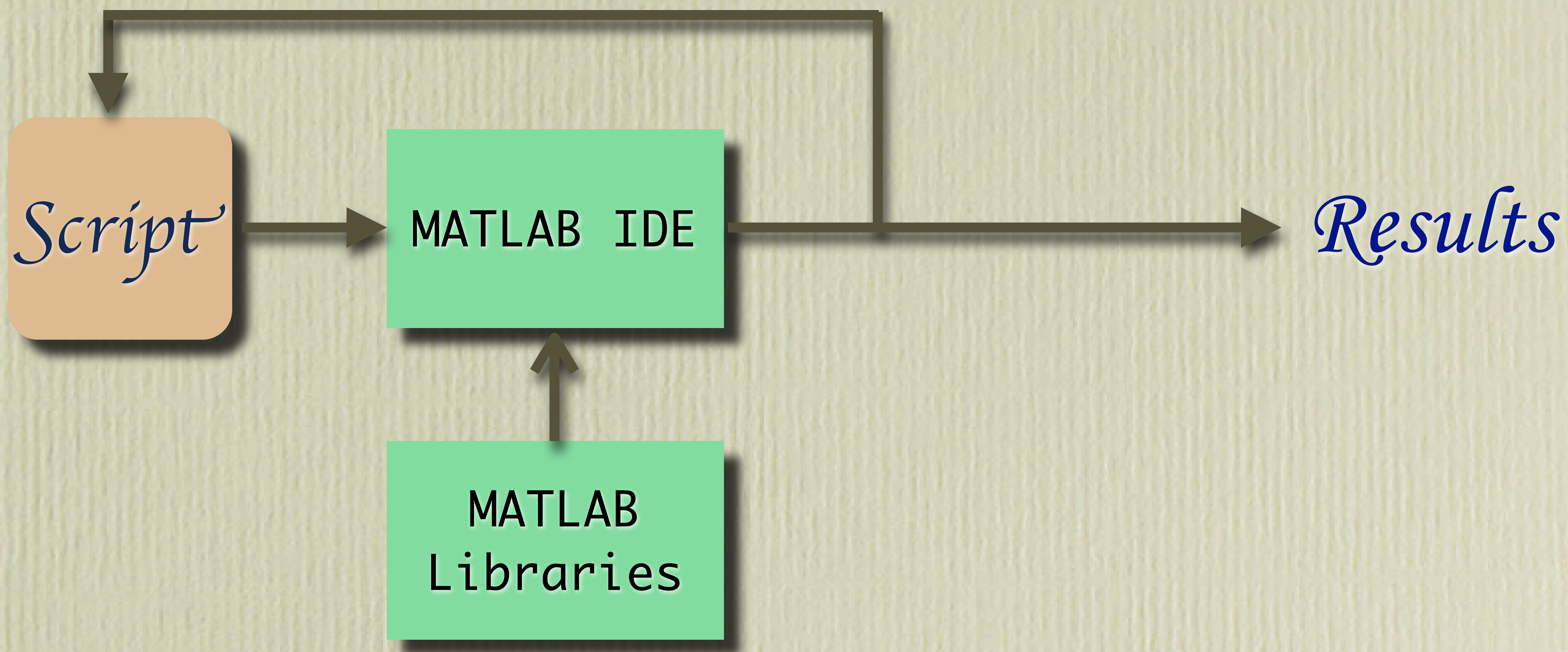
Automatic Parallelization

- Target clusters
- MPI too low-level for MATLAB users
- High-level syntax aids in parallelization
- Simpler semantics eliminate tricky problems (no pointers, no aliasing)

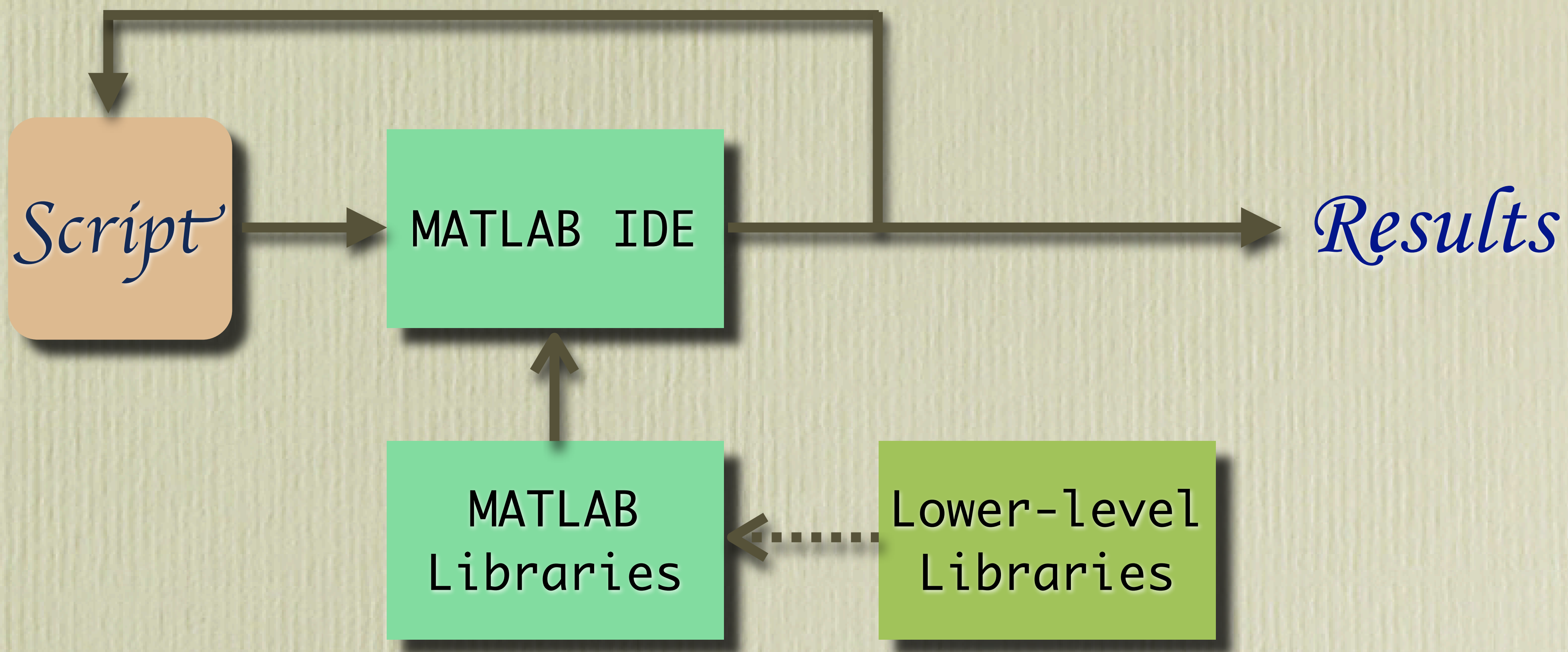
ParaM Architecture



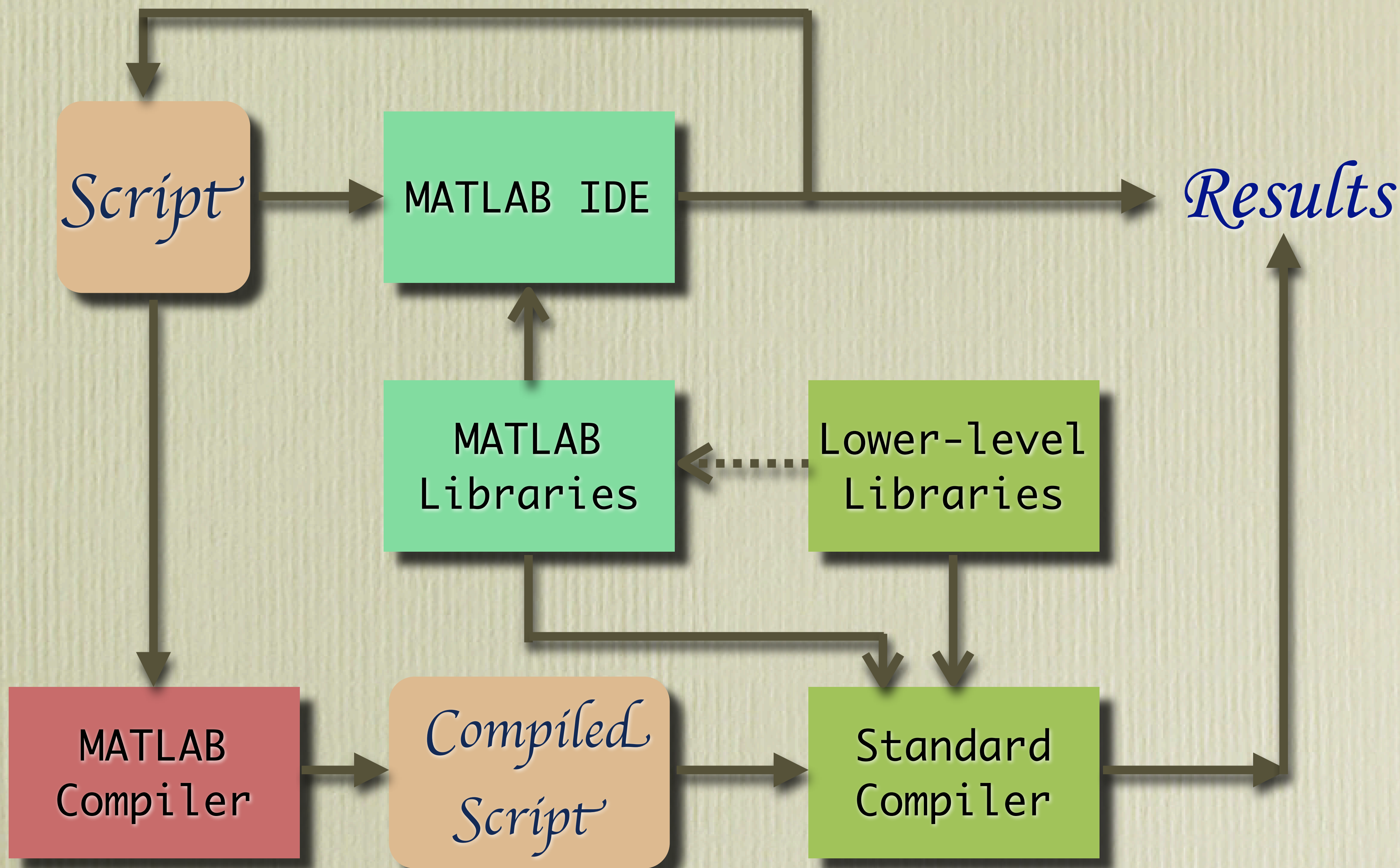
Operating Model



Operating Model



Operating Model



Challenges

- Data distribution
  - distribution types
  - automatic distribution
- Distribution-based parallelization
  - experiences with languages such as HPF
- Global-Array library awareness
  - accounting for performance quirks
- Library selection

Summary

- Domain-Specific Languages a great way to improve productivity
- Need to increase their applicability
  - compiler a critical component
- Need a way to build “aware” compilers
  - utilize domain experts’ knowledge
  - lead to “intelligent” compilers

Collaborations

Grid Computation

- Dynamically evolving runtime
  - can we categorize and speculate scenarios?
- Components-based applications
  - dynamic compilation
  - time-bound compilation
- High-level workflow-based computation
  - high-level semantics provide flexibility

Generic Programming

- Kinds of generalities
  - Data patterns? Algorithms?
- Generic programming ⇒ language constructs?
  - “Concepts” for type-based
- Role of compilers beyond contract enforcement

Other Directions

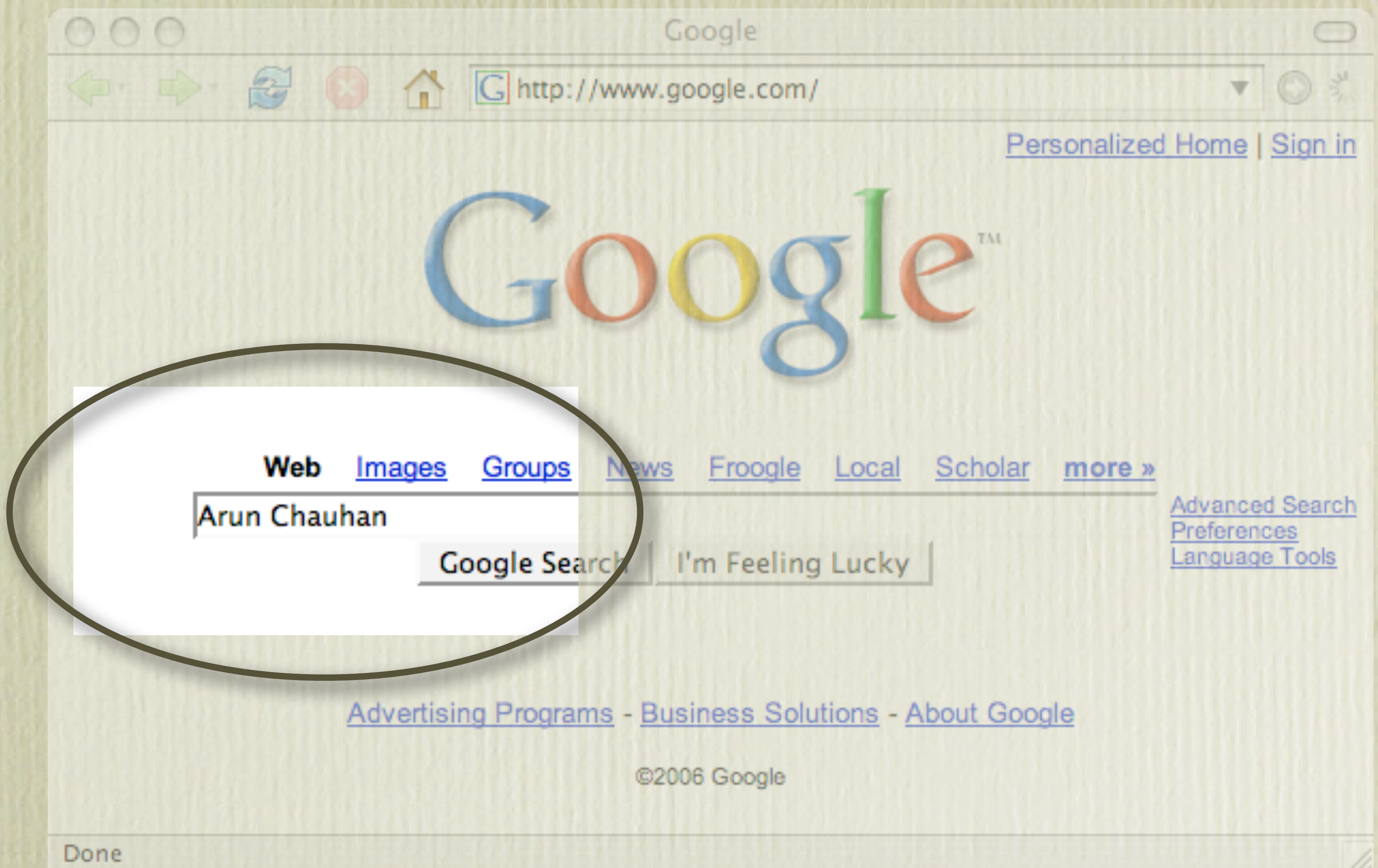
- Adaptive compilation
  - time-bound compilation
  - self-learning
- Security
  - annotations to enforce contracts
  - statistical analysis
- Diversifying the domains
  - VLSI design

Contributions

- Nine published papers over the last few years
- References for today's talk:
  - Ken Kennedy, Bradley Broom, **Arun Chauhan**, Rob Fowler, John Garvin, Chuck Koelbel, Cheryl McCosh, and John Mellor-Crummey. ***Telescoping Languages: A system for Automatic Generation of Domain Languages***. *IEEE Proceedings: Special Issue on Program Generation, Optimization, and Platform Adaptation*, Vol. 93, No. 2, February 2005, pp 387-408.
  - **Arun Chauhan**, Cheryl McCosh, Ken Kennedy, and Richard Hanson. ***Automatic Type-driven Library Generation for Telescoping Languages***. In *Proceedings of the ACM/IEEE SC 2003 Conference on High Performance Networking and Computing (Supercomputing)*, November 2003.
  - **Arun Chauhan** and Ken Kennedy. ***Reducing and Vectorizing Procedures for Telescoping Languages***. *International Journal of Parallel Programming*, Vol. 30, No. 4, August 2002, pp 289-313.
  - Craig Shue, Joshua Hursey, and **Arun Chauhan**, ***MPI Over Scripting Languages: Usability and Performance Tradeoffs***, Technical Report, Department of Computer Science, Indiana University, TR631, 2006.

http://www.cs.indiana.edu/~achauhan/

http://www.cs.indiana.edu/~achauhan/



DSL for Compiler Development

- Stratego (Univ. of Utrecht, the Netherlands)
  - functional interface to tree manipulation
  - C/C++ API
  - constructs to perform dataflow analysis
  - support libraries
- Productivity gains
  - partially built compiler already used at Ohio, UCSD
  - could be transferred back to Rice

Other Activities

- Graduate program
- Co-organizing the Systems Seminar