Research Review 2017

Automated Code Generation for High-Performance, Future-Compatible Graph Libraries

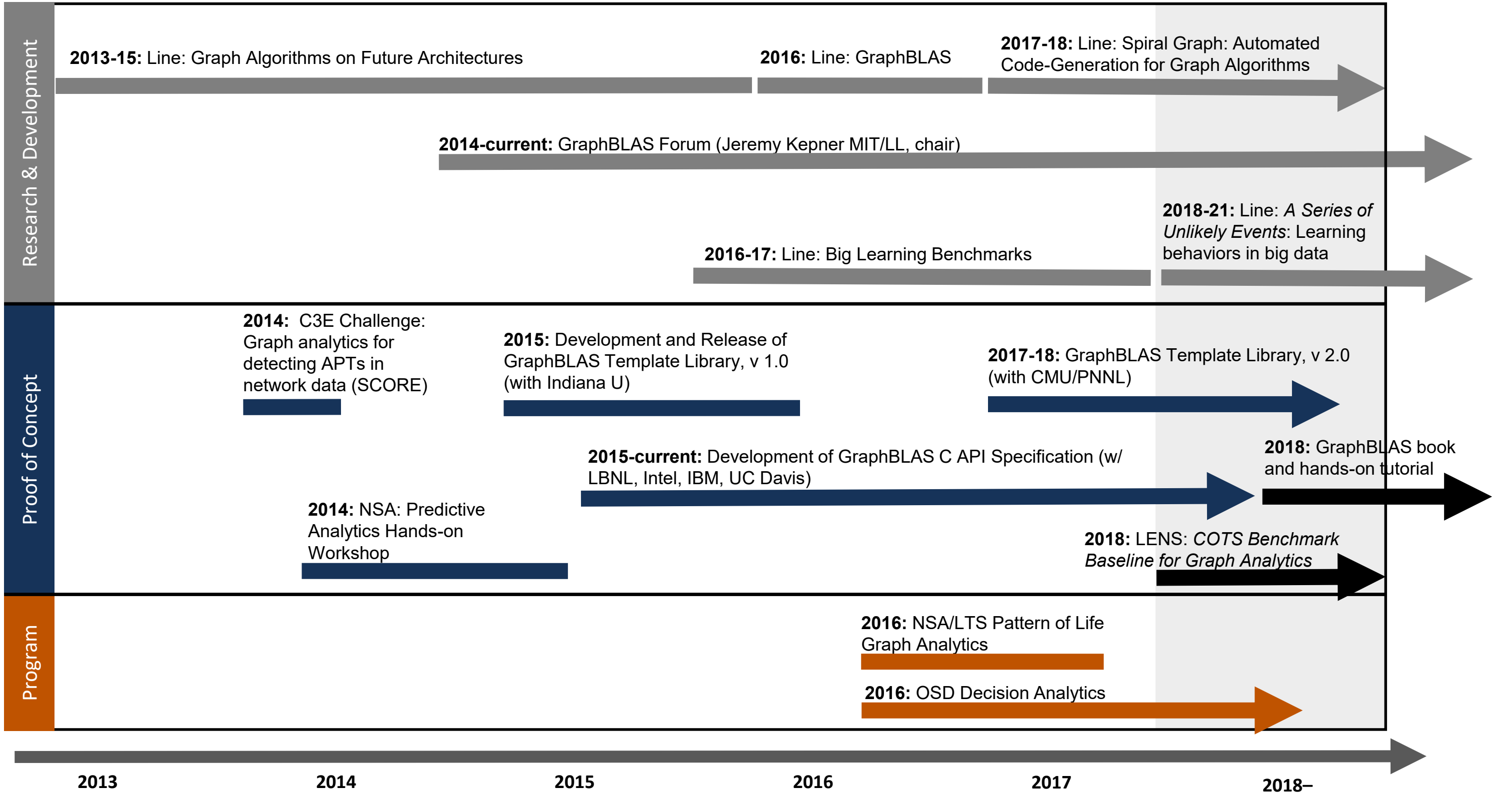
Dr. Scott McMillan, Senior Research Scientist

CMU PI: Prof. Franz Franchetti, ECE

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM17-0774

Data-Intensive Computing Efforts at the SEI



SpiralGraph: Automated Code Generation for *Future-Compatible*, High-Performance Graph Libraries

Problem:

- Heterogeneous high-performance computing (HHPC) architectures are becoming more complex (the NSCI push to exascale).
- Graph algorithms are difficult to program efficiently even on today's hardware architectures.
- Exascale trend: Programming these systems will be much more difficult.[1]

Solution:

- Create an automated code generation tool that produces high-performance graph algorithm implementations for specified hardware.
- **Make graph algorithms performance-portable and future-compatible.**

Approach:

- Create formal abstractions of graph algorithms and primitives (build on GraphBLAS).
- Extend formal abstractions of chosen hardware architectures (build on Spiral and DARPA HACMS, DESA, PERFECT, BRASS).
- Create tool for mapping graph algorithms to hardware architectures for efficient code generation of data-intensive applications.

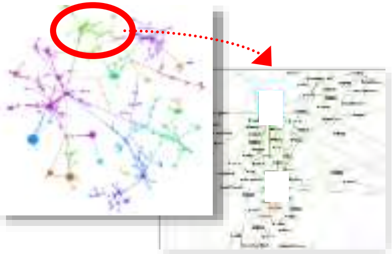
[1]*FACT SHEET: National Strategic Computing Initiative, 29 July 2015.*

Graph Analysis is *Important* and *Pervasive*

ISR

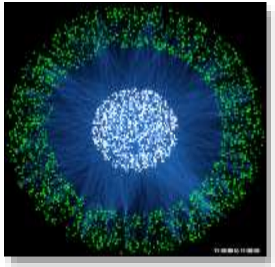
- Graphs represent entities and relationships detected through multi-INT sources
- 1,000s – 1,000,000s tracks and locations
- GOAL: Identify anomalous patterns of life

Social



- Graphs represent relationships between individuals or documents
- 10,000s – 10,000,000s individual and interactions
- GOAL: Identify hidden social networks

Cyber

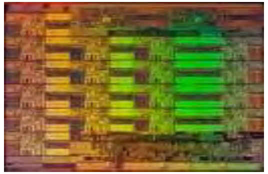


- Graphs represent communication patterns of computers on a network
- 1,000,000s – 1,000,000,000s network events
- GOAL: Identify cyber attacks or malicious software

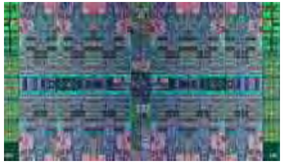
Common Goal: Detection of subtle patterns in massive graphs

Slide credit: Jeremy Kepner, et al. “Mathematical Foundations of the GraphBLAS”, IEEE HPEC, Sept. 2016.

Today's Computing Landscape



Intel Xeon E5-2699v3
662 Gflop/s, 145 W
18 cores, 2.3 GHz
4-way/8-way AVX2



IBM POWER8
384 Gflop/s, 200 W
12 cores, 4 GHz
2-way/4-way VMX/VSX

NVIDIA Tesla P100
10.6 Tflop/s, 250 W
3584 cores, 1.48 GHz
64-way SIMT

Intel Xeon Phi
1.2 Tflop/s, 300 W
61 cores, 1.24 GHz
8-way/16-way LRBni

Qualcomm Snapdragon 810
10 Gflop/s, 2 W
4 cores, 2.5 GHz
A330 GPU, V50 DSP, NEON

Intel Atom C2750
29 Gflop/s, 20 W
8 cores, 2.4 GHz
2-way/4-way SSSE3

Dell PowerEdge R920
1.34 Tflop/s, 850 W
4x 15 cores, 2.8 GHz
4-way/8-way AVX

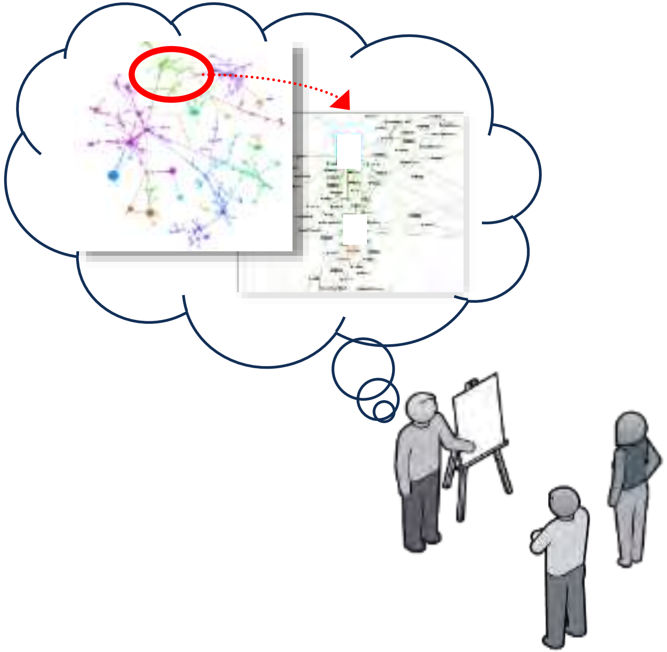
IBM BlueGene/Q
10 Pflop/s, 8 MW
48k x 16 cores, 1.6 GHz
4-way QPX

1 Gflop/s = one billion floating-point operations (additions or multiplications) per second

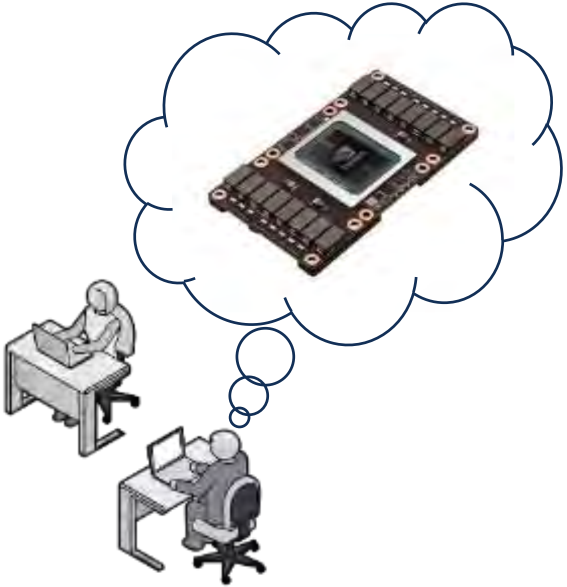
Slide credit: Franz Franchetti, "SPIRAL: Automated Code Generation of Performance Libraries," 2016.

Separation of Concerns

Separate the complexity of graph analysis from the complexity of hardware systems:



Separation of Concerns



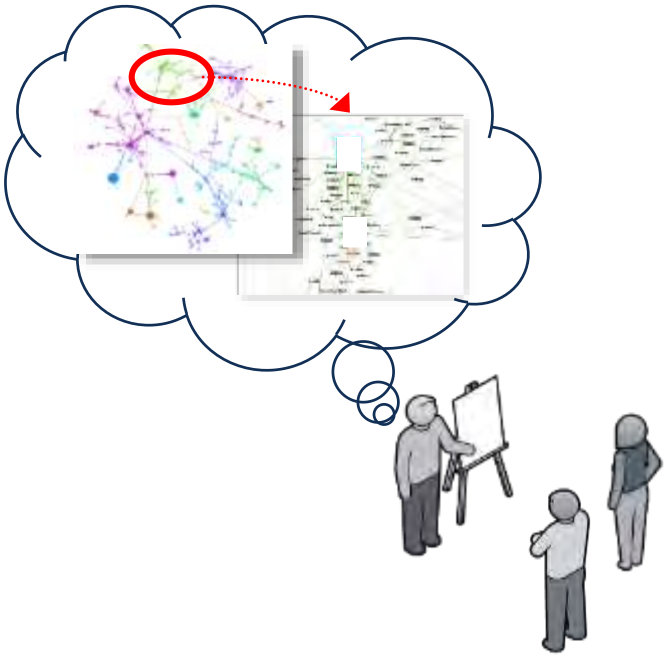
GraphBLAS Primitives

Operation	Description
mxm, mxv, vxm	Perform matrix multiplication (e.g., breadth-first traversal)
eWiseAdd, eWiseMult	Element-wise addition and multiplication of matrices (e.g., graph union, intersection)
extract	Extract a sub-matrix from a larger matrix (e.g., sub-graph selection)
assign	Assign to a sub-matrix of a larger matrix (e.g., sub-graph assignment)
apply	Apply unary function to each element of matrix (e.g., edge weight modification)
reduce	Reduce along columns or rows of matrices (vertex degree)
transpose	Swaps the rows and columns of a sparse matrix (e.g., reverse directed edges)
build	Build a matrix representation from row, column, value tuples
extractTuples	Extract the row, column, value tuples from a matrix representation

http://graphblas.org "A. Buluc, T. Mattson, S. McMillan, J. Moreira, C. Yang, "The GraphBLAS C API Specification, v 1.0.2," updated August 2017.

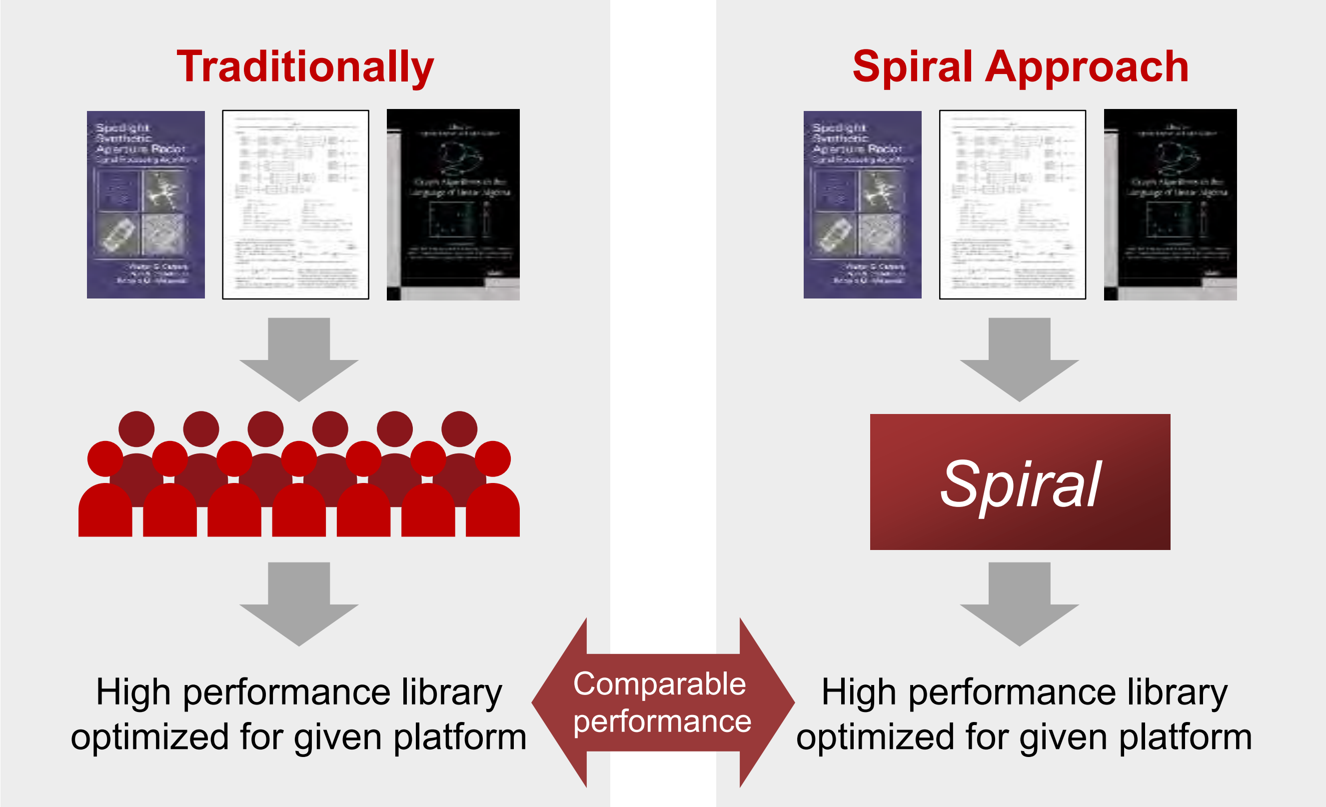
Separation of Concerns

GOAL: write once, run everywhere…fast (**with** help from hardware experts).



GraphBLAS Application Programming Interface (API)

What is Spiral?



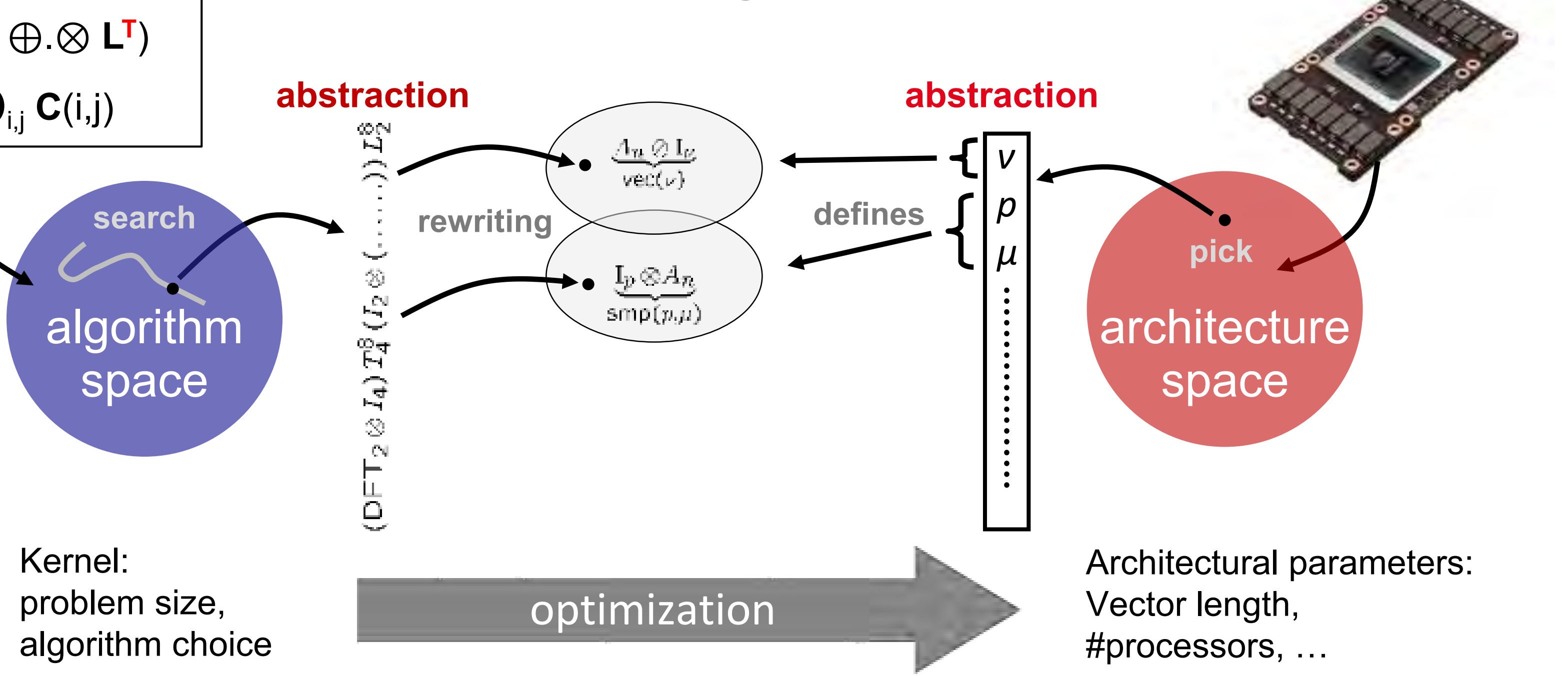
Spiral: Platform-Aware Formal Program Synthesis

GraphBLAS Math:

C⟨L, z⟩ = (**L** ⊕.⊗ **L**^{T})

count = $\bigoplus_{i,j}$ **C**(i,j)

Model: common abstraction
= spaces of matching formulas



GraphBLAS Primitives: The Math

Operation	Mathematical Description	Output	Inputs
mxm	$\mathbf{C}\langle\neg\mathbf{M}, z\rangle = \mathbf{C} \odot (\mathbf{A}^{T} \oplus.\otimes \mathbf{B}^{T})$	$\mathbf{C}$	$\neg$, $\mathbf{M}$, z, $\odot$, $\mathbf{A}$, $\mathbf{T}$, $\oplus.\otimes$, $\mathbf{B}$, $\mathbf{T}$
mxv, (vxm)	$\mathbf{c}\langle\neg\mathbf{m}, z\rangle = \mathbf{c} \odot (\mathbf{A}^{T} \oplus.\otimes \mathbf{b})$	$\mathbf{c}$	$\neg$, $\mathbf{m}$, z, $\odot$, $\mathbf{A}$, $\mathbf{T}$, $\oplus.\otimes$, $\mathbf{b}$
eWiseMult	$\mathbf{C}\langle\neg\mathbf{M}, z\rangle = \mathbf{C} \odot (\mathbf{A}^{T} \otimes \mathbf{B}^{T})$	$\mathbf{C}$	$\neg$, $\mathbf{M}$, z, $\odot$, $\mathbf{A}$, $\mathbf{T}$, $\otimes$, $\mathbf{B}$, $\mathbf{T}$
eWiseAdd	$\mathbf{C}\langle\neg\mathbf{M}, z\rangle = \mathbf{C} \odot (\mathbf{A}^{T} \oplus \mathbf{B}^{T})$	$\mathbf{C}$	$\neg$, $\mathbf{M}$, z, $\odot$, $\mathbf{A}$, $\mathbf{T}$, $\oplus$, $\mathbf{B}$, $\mathbf{T}$
reduce (row)	$\mathbf{c}\langle\neg\mathbf{m}, z\rangle = \mathbf{c} \odot [\oplus_{j} \mathbf{A}^{T}(:,j)]$	$\mathbf{c}$	$\neg$, $\mathbf{m}$, z, $\odot$, $\mathbf{A}$, $\mathbf{T}$, $\oplus$
apply	$\mathbf{C}\langle\neg\mathbf{M}, z\rangle = \mathbf{C} \odot f(\mathbf{A}^{T})$	$\mathbf{C}$	$\neg$, $\mathbf{M}$, z, $\odot$, $\mathbf{A}$, $\mathbf{T}$, f
transpose	$\mathbf{C}\langle\neg\mathbf{M}, z\rangle = \mathbf{C} \odot \mathbf{A}^{T}$	$\mathbf{C}$	$\neg$, $\mathbf{M}$, z, $\odot$, $\mathbf{A}$ ($\mathbf{T}$)
extract	$\mathbf{C}\langle\neg\mathbf{M}, z\rangle = \mathbf{C} \odot \mathbf{A}^{T}(\mathbf{i},\mathbf{j})$	$\mathbf{C}$	$\neg$, $\mathbf{M}$, z, $\odot$, $\mathbf{A}$, $\mathbf{T}$, $\mathbf{i}$, $\mathbf{j}$
assign	$\mathbf{C}\langle\neg\mathbf{M}, z\rangle(\mathbf{i},\mathbf{j}) = \mathbf{C}(\mathbf{i},\mathbf{j}) \odot \mathbf{A}^{T}$	$\mathbf{C}$	$\neg$, $\mathbf{M}$, z, $\odot$, $\mathbf{A}$, $\mathbf{T}$, $\mathbf{i}$, $\mathbf{j}$
build (meth.)	$\mathbf{C} = \mathbb{S}^{mxn}(\mathbf{i},\mathbf{j},\mathbf{v},\odot)$	$\mathbf{C}$	$\odot$, m, n, $\mathbf{i}$, $\mathbf{j}$, $\mathbf{v}$
extractTuples (meth.)	$(\mathbf{i},\mathbf{j},\mathbf{v}) = \mathbf{A}$	$\mathbf{i},\mathbf{j},\mathbf{v}$	$\mathbf{A}$

S. McMillan, et al., "Design and Implementation of the Graph Template Library (GBTL)," SIAM Annual Meeting (AN16), July 2016.
Updated IEEE HPEC Conference, Sep 2017.

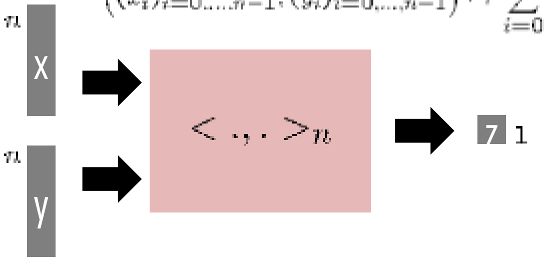
Notation: **i,j** – index arrays, **v** – scalar array, **m** – 1D mask, **other bold-lower** – vector (column), **M** – 2D mask, **other bold-caps** – matrix, **T** – transpose, $\neg$ - structural complement, z – clear output, $\oplus$ monoid/binary function, $\oplus.\otimes$ semiring, **blue** – optional parameters, **red** – optional modifiers

SPIRAL's Math Framework

High Level Operators

$$\langle .,. \rangle_n : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$$

$$\left((x_i)_{i=0,\ldots,n-1}, (y_i)_{i=0,\ldots,n-1}\right) \mapsto \sum_{i=0}^{n-1} x_i y_i$$



Basic Operators

$$\text{Pointwise}_{n,f_i} : \mathbb{R}^n \to \mathbb{R}^n$$
$$(x_i)_i \mapsto f_0(x_0) \oplus \cdots \oplus f_{n-1}(x_{n-1})$$

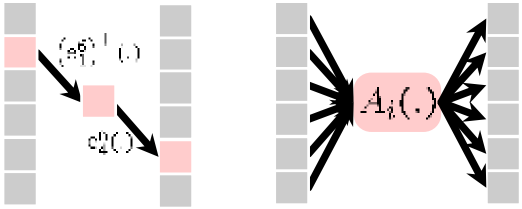
$$\text{Atomic}_{f(.,.)} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$$
$$(x, y) \mapsto f(x, y)$$

$$\text{Pointwise}_{n\times n,f_i} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$$
$$\left((x_i)_i, (y_i)_i\right) \mapsto f_0(x_0, y_0) \oplus \cdots \oplus f_{n-1}(x_{n-1}, y_{n-1})$$

$$\text{Reduction}_{n,f_i} : \mathbb{R}^n \to \mathbb{R}$$
$$(x_i)_i \mapsto f_{n-1}(x_{n-1}, f_{n-2}(x_{n-2}, f_{n-3}(\ldots f_0(x_0, \text{id}()) \ldots)))$$

Loop Abstraction

$$\coprod_{i=0}^{n-1} : (D \to R)^n \to (D \to R)$$
$$A_i \mapsto (x \mapsto A_0(x) \sqcup \cdots \sqcup A_{n-1}(x))$$



Rule Based Compiler

$$\text{Code}\left(y = (A \circ B)(x)\right) \to \left\{\texttt{decl}(t), \text{Code}\left(t = B(x)\right), \text{Code}\left(y = A(t)\right)\right\}$$

$$\text{Code}\left(y = \left(\sum_{i=0}^{n-1} A_i\right)(x)\right) \to \left\{y := \vec{0}, \texttt{for}(i = 0..n-1)\ \text{Code}\left(y += A_i(x)\right)\right\}$$

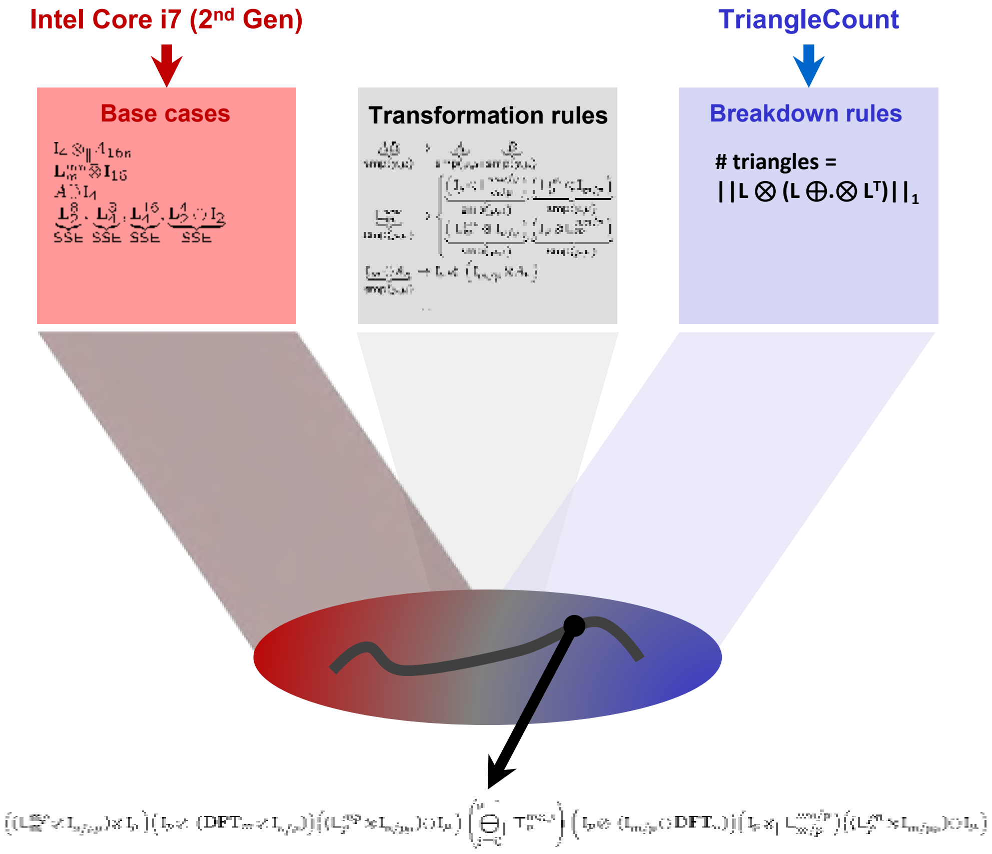
$$\text{Code}\left(y = (e_i^n)^\top(x)\right) \to y[0] := x[i]$$

$$\text{Code}\left(y = e_i^n(x)\right) \to \left\{y = \vec{0}, y[i] := x[0]\right\}$$

$$\text{Code}\left(y = \text{Atomic}_f(x)\right) \to y[0] := f(x[i])$$

Leverages DARPA HACMS

Autotuning in Constraint Solution Space



OL specification

↓ Expansion + backtracking

OL (dataflow) expression

↓ Recursive descent

Σ-OL (loop) expression

↓ Confluent term rewriting

Optimized Σ-OL expression

↓ Recursive descent

Abstract code

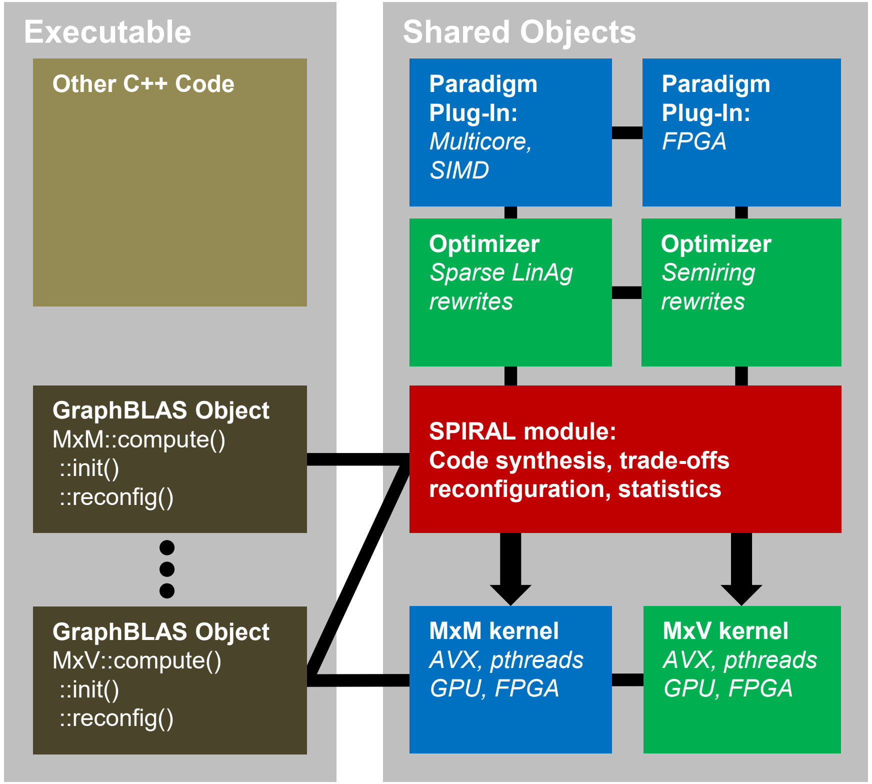
↓ Confluent term rewriting

Optimized abstract code

↓ Recursive descent

C code

Long-Range Goal: SPIRAL as JIT and GraphBLAS Optimizer



Source Code

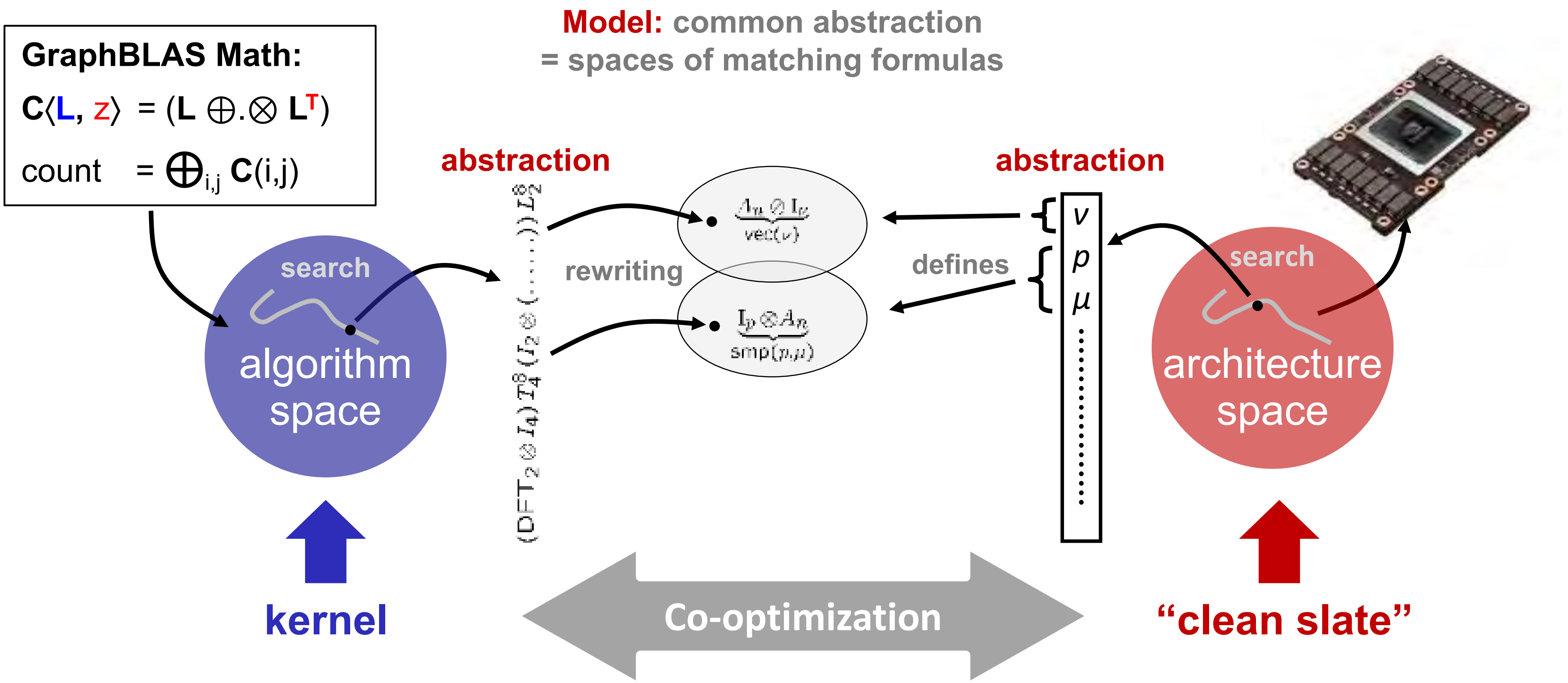
- C++, GraphBLAS calls, other supported libraries
- Code = specification, not program

SPIRAL Module

- Acts as JIT, delayed execution engine, Inspector/executor
- Implements telescoping language ideas
- Rewrites code into better algorithms
- Compiles to range of platforms CPU, GPU, FPGA
- Plug-in mechanism for post deployment reconfiguration and update

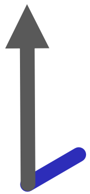
Leverages DARPA BRASS

Formal Approach To Co-Optimization



Leverages DARPA DESA and PERFECT

$$(\hat{\mathcal{A}}, \hat{\mathcal{M}}) = \text{argmin}_{\theta,\xi} \mathrm{C}_{\mathrm{m}}(\mathcal{A}(\theta), \mathcal{M}(\xi))$$



M (»)

> *“What is the right architecture for my application?”*
> *“What architecture features are good for my application?”*

Summary and Future Work

- GraphBLAS C API Specification is complete
  - Mathematical descriptions of primitive operations are complete
  - Algorithm development using the API is in progress (dozens completed)
- Exploration of performant code generation and data structures has begun
- Goals for FY18:
  - Integrate primitives and necessary “knowledge” into Spiral code generation technology
  - Target different hardware platforms
    - Multi-core CPUs
    - Accelerators: FPGAs, Graphics Processing Units (GPUs)
  - Generate code for algorithms in benchmark and Challenge problems
    - Graph 500: breadth-first search, shortest paths
    - DARPA HIVE Graph Challenge: subgraph isomorphism
- Long-Range Goal: Co-synthesis of hardware and software

Contact Information

Presenter / SEI PI

Dr. Scott McMillan

Senior Research Scientist

Email: smcmillan@sei.cmu.edu

Presenter / CMU PI

Prof. Franz Franchetti

ECE Department

Email: franzf@ece.cmu.edu