Carnegie Mellon University
Software Engineering Institute

## Research Review 2021

# Spiral AI/ML: Co-optimization for High-Performance, Data-Intensive Computing in Resource-Constrained Environments

**November 2021**

Scott McMillan

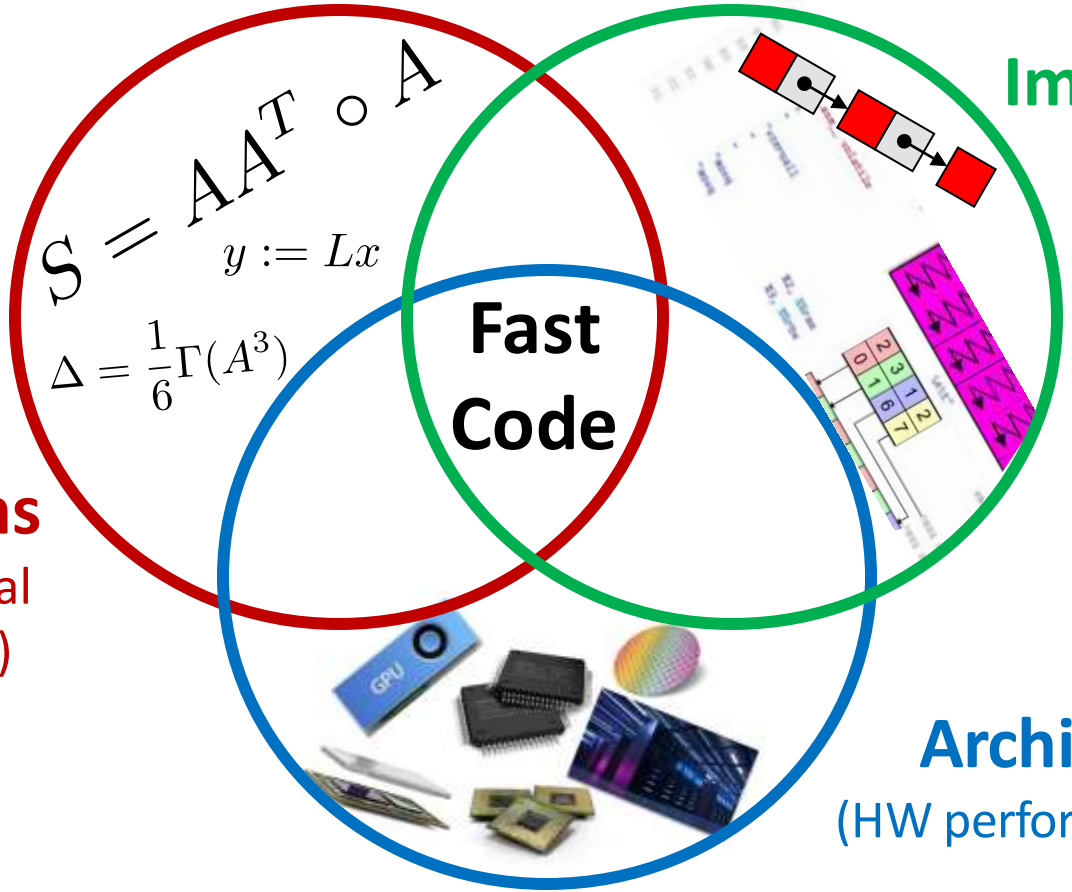
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

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.



DM21-0887

# Fast code depends on three interdependent ingredients




**Algorithms**
(mathematical descriptions)

**Implementations**
(primitive ops and data structures)

**Architectures**
(HW performance models)

# Spiral/AIML: Co-Optimization for High-Performance, Data-Intensive Computing in Resource-Constrained Environments

*"Rapidly delivering artificial intelligence to a combat zone won't be easy." Col. Drew Cukor, USMC.*

**Problem(s)**

- Increasing complexity in computing architectures.
- Mission cost, size, weight, and power (CSWAP) constraints drive increasing use of FPGAs and ASICs (more complexity).
- Achieving performance from these platforms is *hard*.
- Achieving performance from **data-intensive applications** (graphs, ML, AI) is *hard*.

**Solution**

- Automatic code generation for data-intensive computations.
- Simultaneous, automatic co-optimization of hardware within CSWAP constraints.

**Approach**

- Identify common AI/ML/Graph computational primitives and their ***mathematical descriptions***.
- Develop ***hardware performance models*** allowing Spiral to choose between components satisfying CSWAP requirements.
- Encode knowledge about ***efficient implementations*** of graph, ML, and AI computational primitives into **Spiral code-generation technology**.

# What is Spiral?

## Traditionally

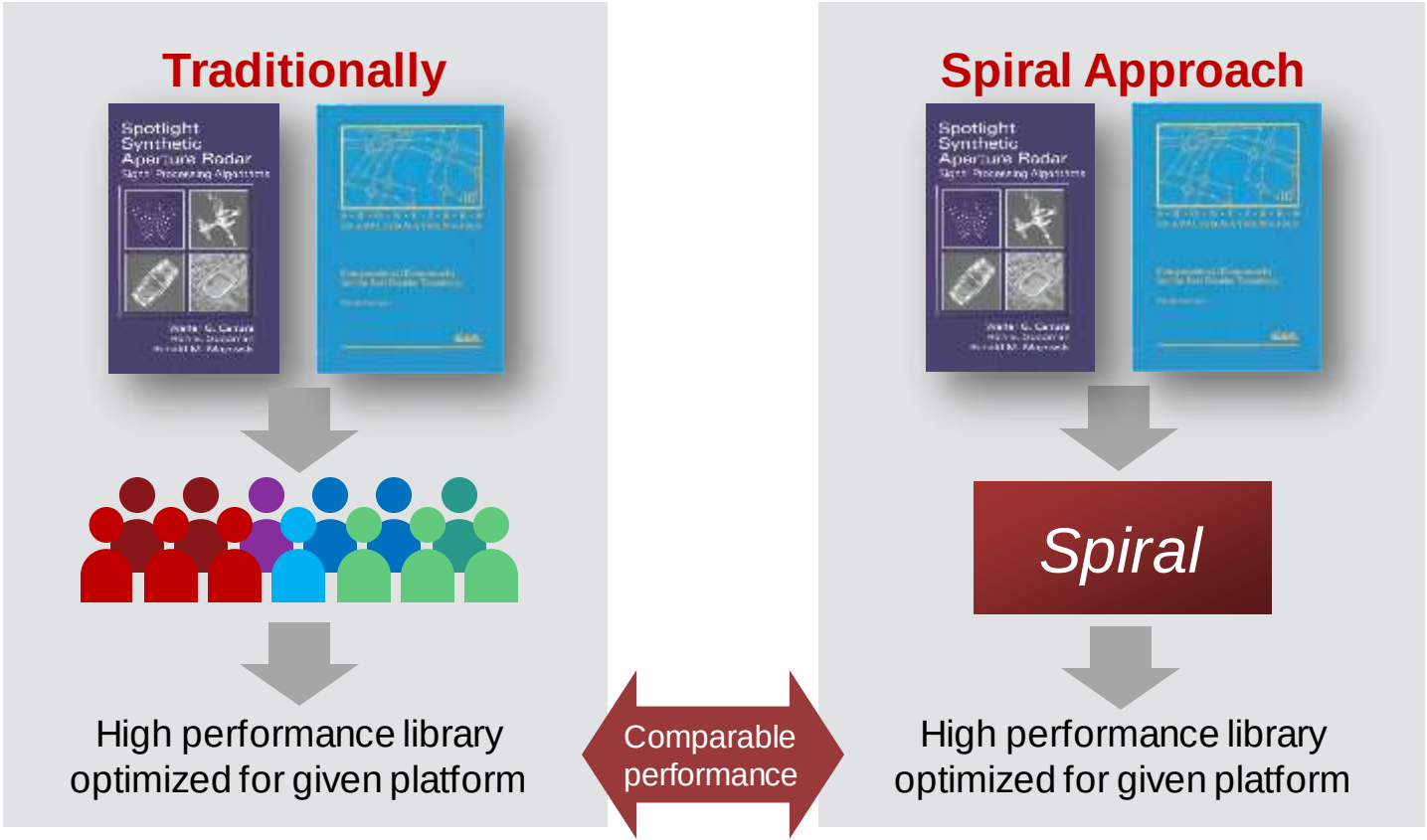

High performance library optimized for given platform

## Spiral Approach


Spiral

High performance library optimized for given platform

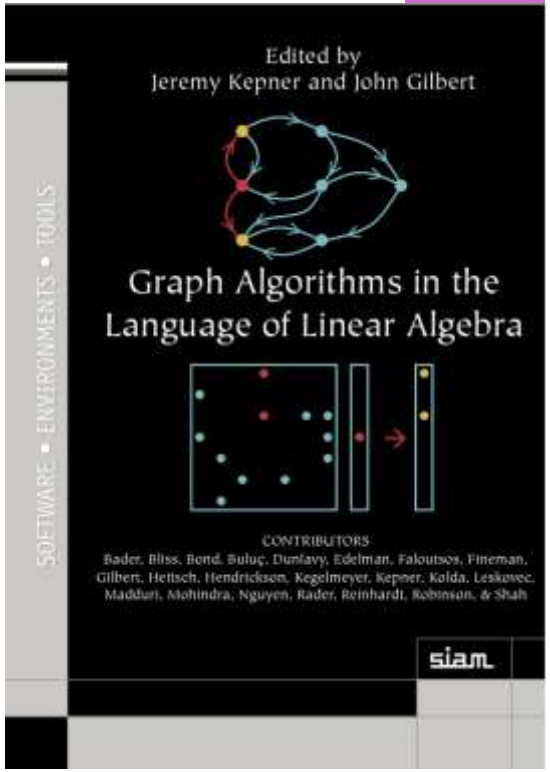
# What is Spiral?



# Previous Work Addressed Spiral for Graphs

- Spiral understands the rules of linear algebra.
- Much research in graph algorithms using linear algebra
- However, it involves *sparse* and *irregular* data.
- FY2016-2018: working to "teach" the data structures, operations, and optimizations used in graph algorithms
  - Needed a mathematical specification of common operations
  - Led to involvement in the **GraphBLAS** community

2011



# Foundational GraphBLAS References

## Mathematical Foundations of the GraphBLAS

IEEE HPEC 2016

Jeremy Kepner (MIT Lincoln Laboratory Supercomputing Center), Peter Aaltonen (Indiana University), David Bader (Georgia Institute of Technology), Aydın Buluç (Lawrence Berkeley National Laboratory), Franz Franchetti (Carnegie Mellon University), John Gilbert (University of California, Santa Barbara), Dylan Hutchison (University of Washington), Manoj Kumar (IBM), Andrew Lumsdaine (Indiana University), Henning Meyerhenke (Karlsruhe Institute of Technology), Scott McMillan (CMU Software Engineering Institute), Jose Moreira (IBM), John D. Owens (University of California, Davis), Carl Yang (University of California, Davis), Marcin Zalewski (Indiana University), Timothy Mattson (Intel)

## Design of the GraphBLAS API for C

IEEE HPEC 2017

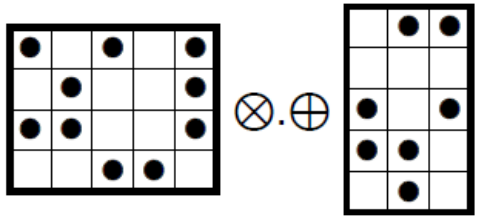
Aydın Buluç[†], Tim Mattson[‡], Scott McMillan[§], José Moreira[¶], Carl Yang[*,†]

[†]*Computational Research Division, Lawrence Berkeley National Laboratory*
[‡]*Intel Corporation* [§]*Software Engineering Institute, Carnegie Mellon University* [¶]*IBM Corporation*
[*]*Electrical and Computer Engineering Department, University of California, Davis, USA*

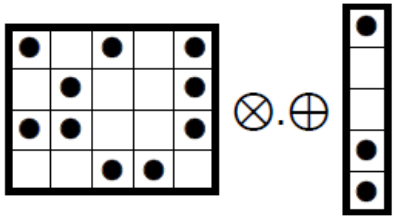
# GraphBLAS Primitives

- Basic objects (opaque types)
  - Matrices (sparse or dense), vectors (sparse or dense), algebraic operators (semirings)
- Fundamental operations over these objects

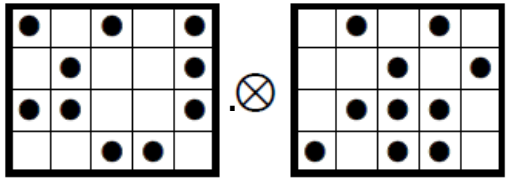
Sparse matrix times sparse matrix



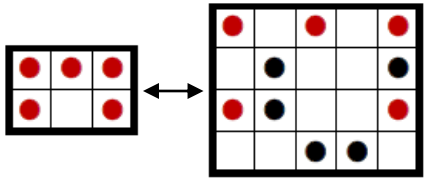
Sparse matrix times sparse vector



Element-wise multiplication (and addition)



Sparse matrix extraction (and assignment)



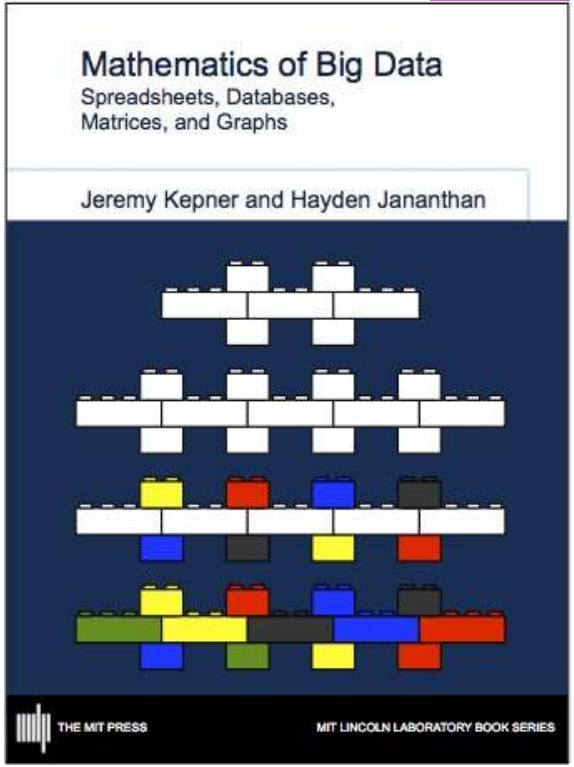
…plus reduction, transpose, Kronecker product, filtering, transform, etc.

Buluc, A.; Mattson, T.; McMillan, S.; Moreira, J.; &Yang, C. "The GraphBLAS C API Specification," v1.0 May 2017, v1.1 May 2018, v1.3 Sep 2019. **v2.0 Nov 2021 (at SC21).** http://graphblas.org

# GraphBLAS Is Not Just for Graphs

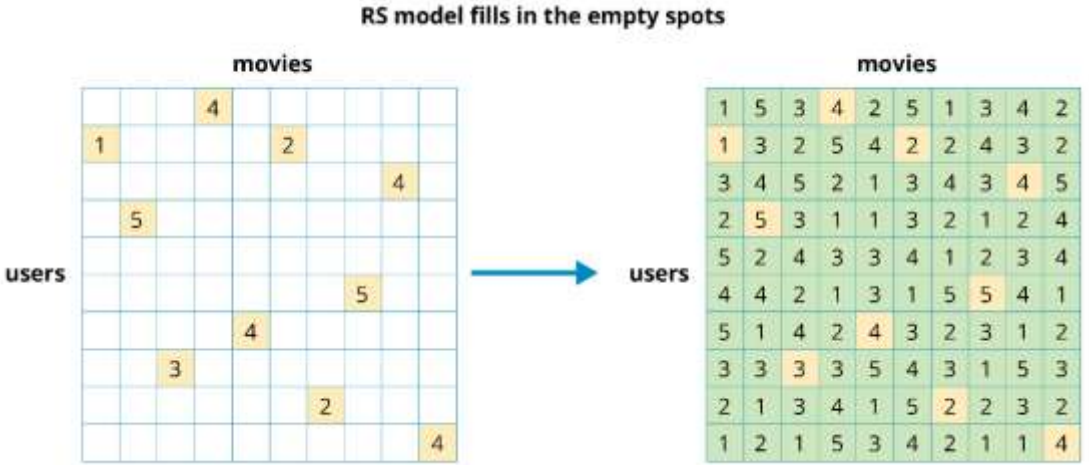
- It is a general linear algebra specification with a rich set of operations on any type of matrix and vector data.
- Recent work has shown applicability of GraphBLAS for operations on data in tabular form.
- **AI, specifically ML, workloads use both sparse and dense data…**

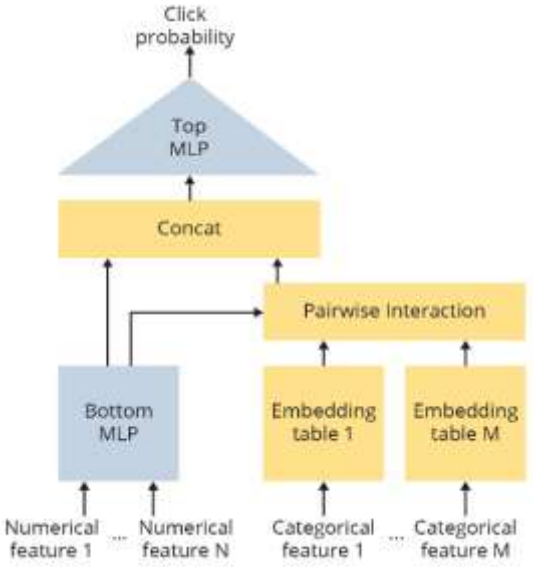
2018



# GraphBLAS Operations Can Also Perform ML

- Recommender Systems predict broad information from sparse data
- Deep Learning Recommendation Models (DLRM) has a mixed sparse-dense workloads

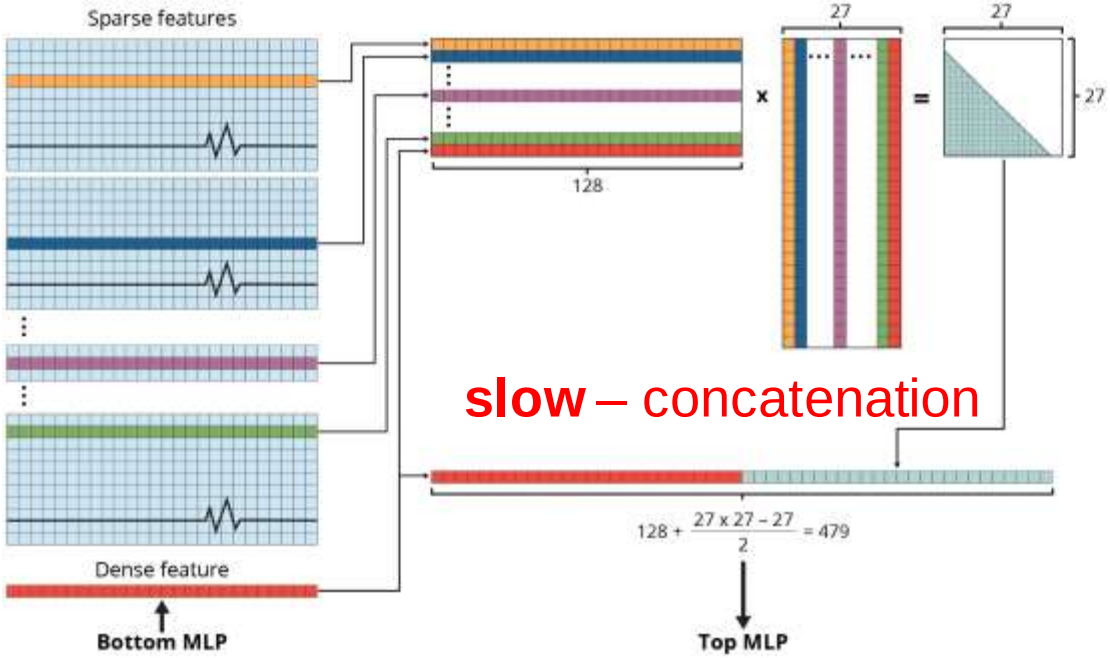


# Deep Learning Recommendation Model (DLRM)

Co-Funded by DARPA SDH

**slow** – DRAM Access

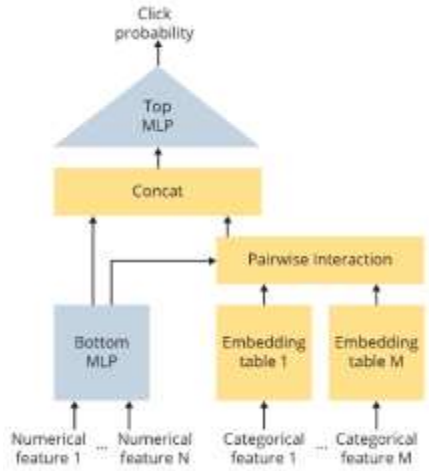



**slow** – concatenation

*fast* – Vector Math

*fast* – Vector Math

*fast* – Vector Math



# FY19-21: Adding More GraphBLAS for ML to Spiral

Three primary lines of R&D

- **Algorithms**
  - Analyzing graph, ML and AI workloads
  - Identifying computational kernels
  - Developing data structures
- **Implementations**
  - Expanding the GraphBLAS API
  - Optimizing GraphBLAS Template Library
  - Developing the Graph Kernel Collection
- **Architectures**
  - Developing performance models for GPU and multi-core CPU


## Spiral Approach


*Spiral*

High performance library optimized for given platform

# SPIRAL is an Advanced Source-to-Source Compiler

**Compiler Analogies**

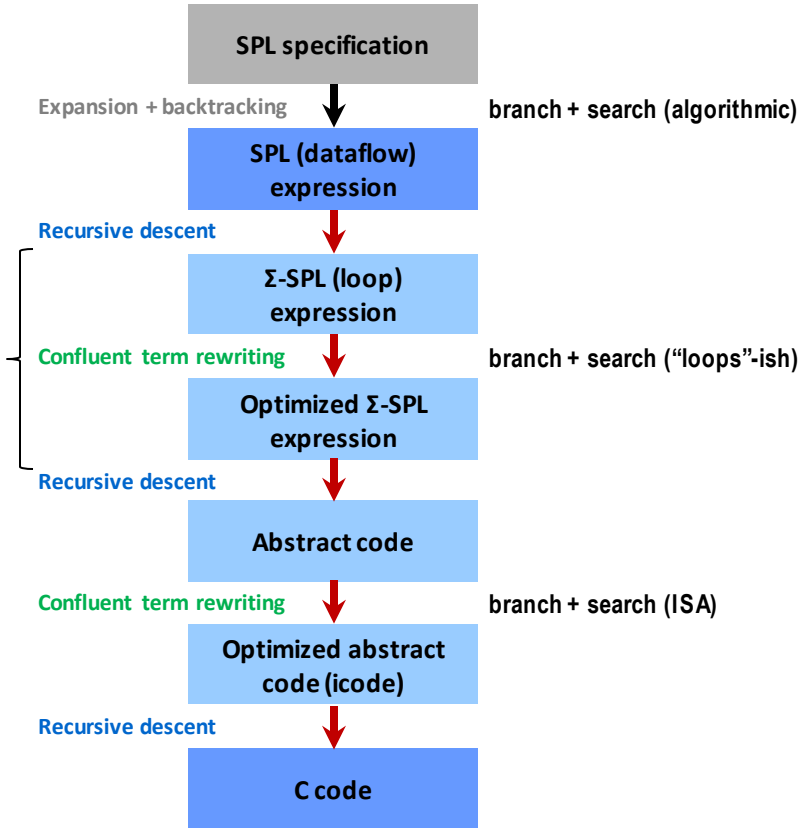
AST

Finest grained AST
algorithm breakdown, and some architecture-specific features; e.g., SIMD vs. warp, number of threads, etc.

Optimization Phase

IR
of a compiler

IR
mapping to HW specific instruction sets



# SPIRAL is an Advanced Source-to-Source Compiler

**Compiler Analogies**

AST

Finest grained AST
algorithm breakdown, and some architecture-specific features; e.g., SIMD vs. warp, number of threads, etx

Optimization Phase

Performance Models

IR
of a compiler

IR
mapping to HW specific instruction sets


Mathematical Descriptions

SPL specification

Expansion + backtracking

branch + search (algorithmic)

SPL (dataflow) expression

Recursive descent

Σ-SPL (loop) expression

Confluent term rewriting

branch + search ("loops"-ish)

Optimized Σ-SPL expression

Recursive descent

Abstract code

Confluent term rewriting

branch + search (ISA)

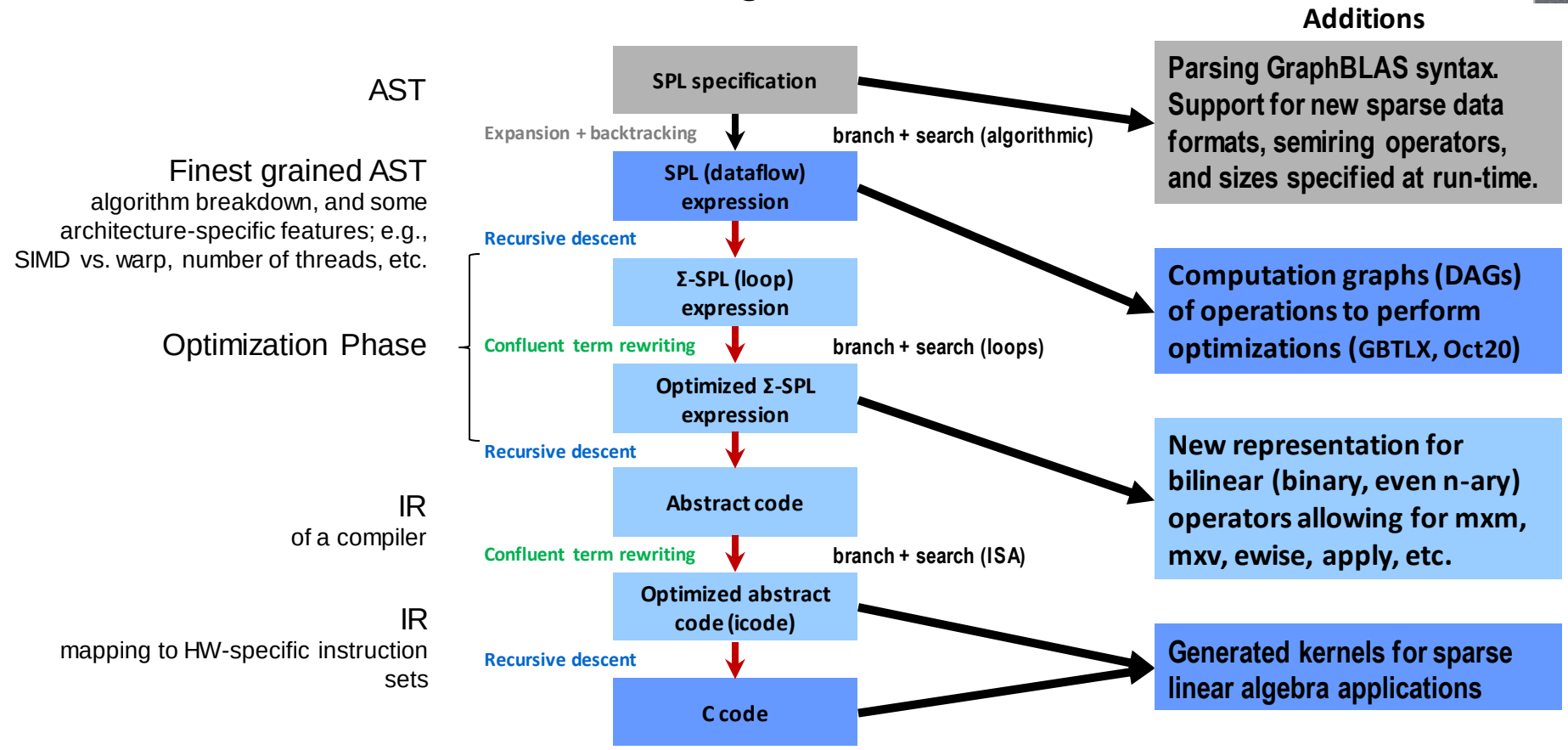
Optimized abstract code (icode)

Recursive descent

C code

Expert System

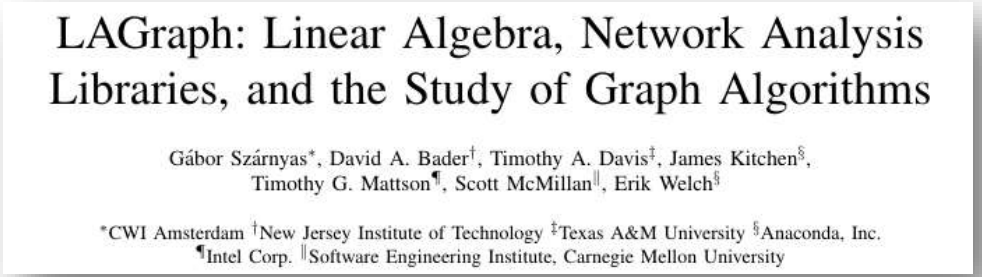
# SPIRAL: Code Generation Stages/Additions



# Artifact Highlights: Algorithms

## Publications and Implementations

1. **LAGraph: a community effort to curate algorithms built on top of GraphBLAS (IPDPSW'19)**
2. Linear algebraic formulation of edge-centric k-truss algorithms with adjacency matrices (HPEC'18)
3. Graph signal processing and deep learning (Signals'20)
4. Delta-stepping SSSP: from vertices and edges to GraphBLAS implementations (IPDPSW'19)
5. Linear algebraic depth-first search (ARRAY'19)
6. Linear algebraic Louvain clustering algorithm (IPDPSW'20)
7. Edge entropy as an indicator of effectiveness of Graph Neural Networks for node classification (SSC'20)
8. Bipartite graph generation using Kronecker product (IPDPSW'20)
9. **The design of LAGraph (SIAM CSE'21)**
10. **LAGraph: Linear Algebra, Network Analysis Libraries, and the Study of Graph Algorithms (IPDPSW'21)**
11. GraphSAINT, graph neural networks
12. Deep Learning Recommender Models
13. GAP Benchmark algorithms
14. Triangle Centrality analysis



LAGraph: Linear Algebra, Network Analysis Libraries, and the Study of Graph Algorithms

Gábor Szárnyas*, David A. Bader†, Timothy A. Davis‡, James Kitchen§, Timothy G. Mattson¶, Scott McMillan‖, Erik Welch§

*CWI Amsterdam †New Jersey Institute of Technology ‡Texas A&M University §Anaconda, Inc. ¶Intel Corp. ‖Software Engineering Institute, Carnegie Mellon University

Download and/or contribute code at:
https://github.com/GraphBLAS/LAGraph

# Artifact Highlights: Implementations

## APIs:

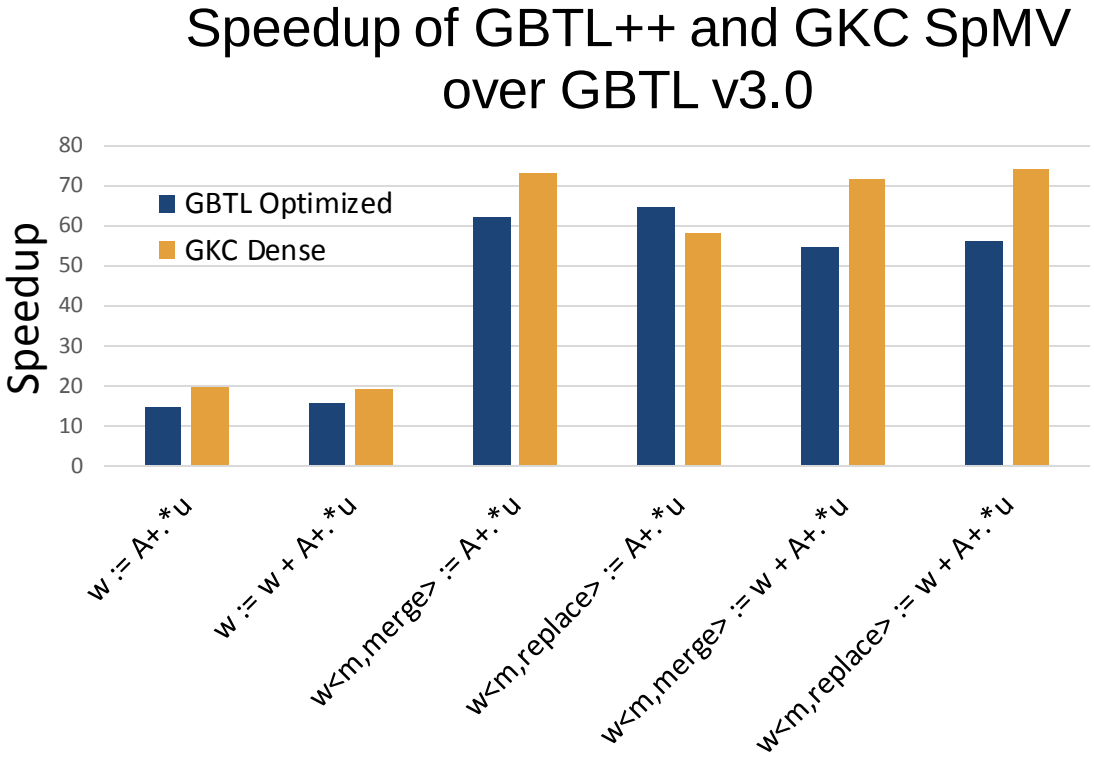
15. GraphBLAS C API Spec v1.3 (9/19)
16. GraphBLAS C API Spec v2.0 for multithreading (IPDPSW'21)
17. GraphBLAS C++ API Spec (draft, IPDPSW'20)
18. Distributed GraphBLAS API (IPDPSW'20)
19. Spiral/GBTLX source-to-source translator, (HPEC'20)

## Open-source code:

**20. GraphBLAS Template Library (GBTL) v3.0**
**21. Graph Kernel Collection**
22. LAGraph Algorithms Repository, v1.0: a community effort to curate algorithms built on top of GraphBLAS
23. NWGraph: A Library of Generic Graph Algorithms and Data Structure in C++20 (with PNNL/UW)

## Tutorials:

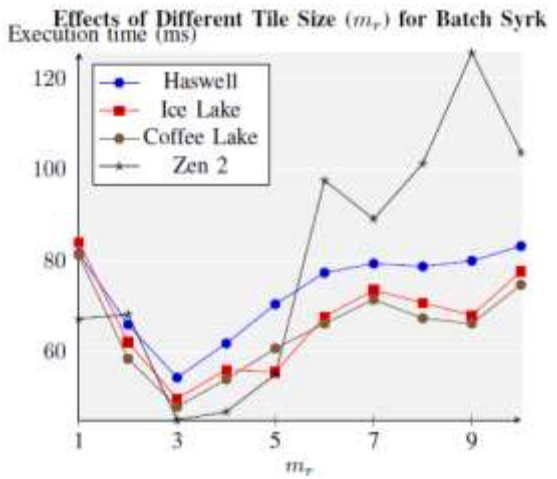
- "A Hands-on Introduction to GraphBLAS"
  24. Tutorial in C (HPEC'18, '19)
  25. Tutorial in python (SIAM CSE'21, ICS'21, HPEC'21)

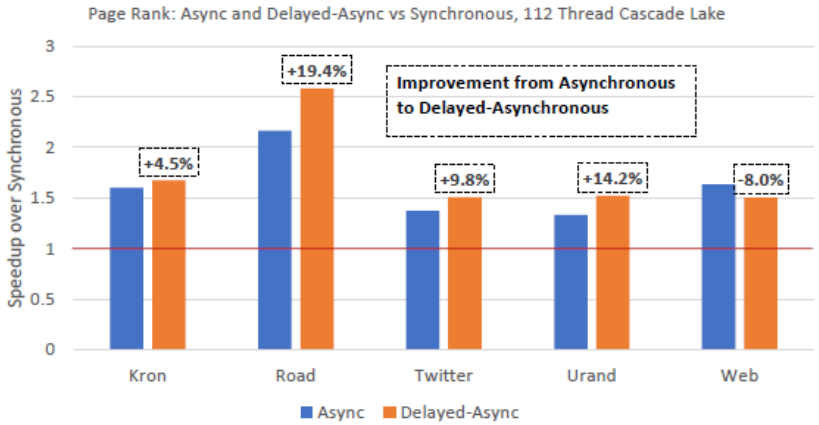


# Artifact Highlights: Architectures

**Analyses and Publications:**

26. PageRank acceleration for large graphs with scalable hardware (FPGA) and 2-step SpMV (HPEC'18)
27. Preliminary exploration of large-scale triangle counting on shared-memory, multicore systems (HPEC'18)
28. Exploration of fine-grained parallelism for load balancing eager k-truss on GPU and CPU (HPEC'19)
29. Efficient SpMV operation for large and highly sparse matrices using scalable multi-way merge parallelization (Micro'19)
30. FESIA: Fast, efficient set intersection approach on modern CPUs (ICDE'20)
31. Toward an objective metric for exact triangle count (HPEC'20)
32. Triangle counting with cyclic distribution (HPEC'20)
33. Evaluation of graph analytic frameworks using the GAP benchmark suite (IISWC'20)
34. **Analytical models for portable Deep Learning Recommender Model (DLRM) implementations (manuscript)**
35. A method for fusing non-element-wise layers in DNNs (HPEC'21)
36. Modeling matrix engines for portability and performance (manuscript)
37. **Delayed asynchronous iterative graph algorithms (HPEC'21)**



# Team Members

## SEI

**Scott McMillan**
Principal Engineer
The AI Division
Softw are Engineering Institute

**John Wohlbier**
Senior Research Scientist
The AI Division
Softw are Engineering Institute

**Oren Wright**
Senior Researcher
The AI Division
Softw are Engineering Institute

**Jason Larkin**
Senior Researcher
The AI Division
Softw are Engineering Institute

## CMU ECE

**Franz Franchetti**
Kavčić-Moura Professor, ECE

Associate Dean for Research
College of Engineering

**Jose Moura**
Associate Department Head
for Research and Strategy

**Tze Meng Low**
Assistant Research Professor

**James C. Hoe**
Professor of Electrical and
Computer Engineering

# Team Members

## CMU Staff

Dr. Het Mankad

Dr. Daniele Spampinato
(now at Hauwei)

## SEI Interns

Courtney Rankin

Sandra Sanjeev
(now at Microsoft)

## CMU Students

Elliott Binder

Mark Blanco

Peter Oostema

Sanil Rao

Upasana Sridhar

Mark Cheung

Guanglin Xu

George Ralph

Arvind Srinivasan

John Shi

Dr. Fazle Sadi
(Intel)

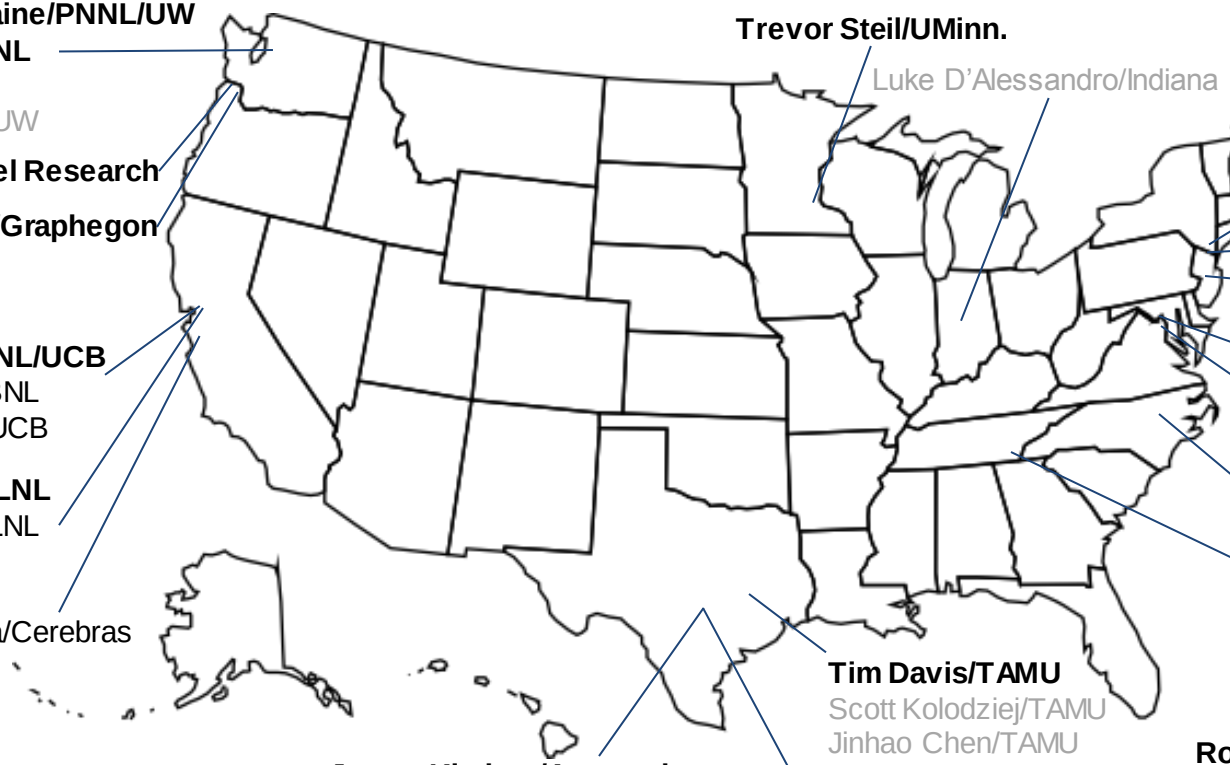
Dr. Anuva Kulkarni
(Motional)

Dr. Jiyuan Zhang
(Facebook)

Paul Brouwer

Dr. Vadim Zaliva
(Cambridge Univ. Digamma)

# External Collaborators



**Andrew Lumsdaine/PNNL/UW**
**Jesun Firoz/PNNL**
**Tony Liu/UW**
Kevin Deweese/UW

**Tim Mattson/Intel Research**

**Michel Pelletier/Graphegon**

**Ben Brock/UCB**
**Aydin Buluc/LBNL/UCB**
Oguz Selvitopi/LBNL
James Demmel/UCB

**Roger Pearce/LLNL**
Geoff Sanders/LLNL
Ben Priest/LLNL

Natalia Vassilieva/Cerebras

**Trevor Steil/UMinn.**

Luke D'Alessandro/Indiana

Jeremy Kepner/MIT/LL

**Jose Moreira/IBM**
Manoj Kumar/IBM

Eric Hein/EMU/Lucata

**David Bader/NJIT**

**Paul Burkhardt/DOD**
**Kaushik Velusamy/UMBC**

DARPA: HIVE, SDH, DSSOC, PAPPA, DPRIVE

Phil Ratzloff/SAS
Matt Galati/SAS

Jeffrey Vetter/ORNL

Albert-Jan Yzelman
Danielle Spampinato
Now at Huawei, Zurich

**Tim Davis/TAMU**
Scott Kolodziej/TAMU
Jinhao Chen/TAMU
Et al.

**James Kitchen/Anaconda**
**Eric Welch/Anaconda**

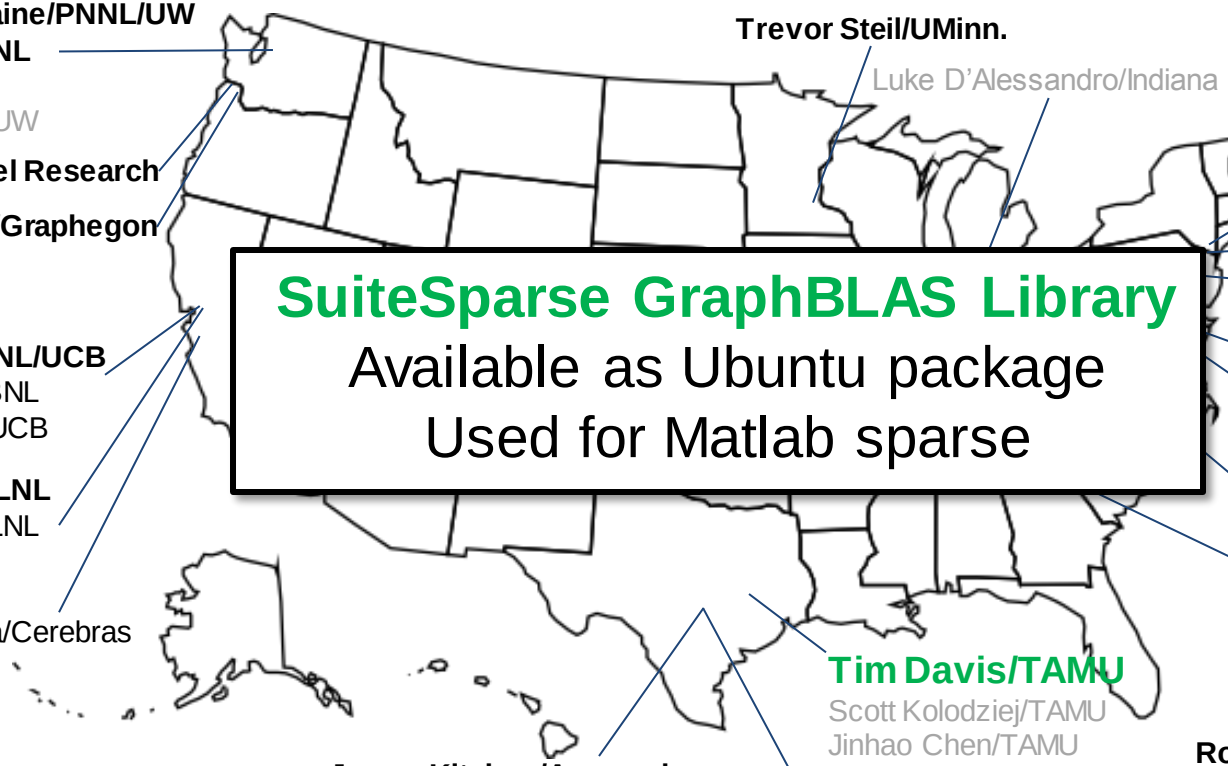
**Roi Lipman**
**RedisLabs, Israel**

Intel MKL/Graph Working Group (Russia)
Andrey Nikolaev, Alexey Kukanov, Kirill Voronin

**Intel MKL/Graph Working Group**
**Henry Gabb, Ramesh Peri, et al.**

**Gabor Szarnyas/**
**Hungarian Academy of Sciences**

# External Collaborators



**Andrew Lumsdaine/PNNL/UW**
**Jesun Firoz/PNNL**
**Tony Liu/UW**
Kevin Deweese/UW

**Tim Mattson/Intel Research**

**Michel Pelletier/Graphegon**

**Ben Brock/UCB**
**Aydin Buluc/LBNL/UCB**
Oguz Selvitopi/LBNL
James Demmel/UCB

**Roger Pearce/LLNL**
Geoff Sanders/LLNL
Ben Priest/LLNL

Natalia Vassilieva/Cerebras

**Trevor Steil/UMinn.**

Luke D'Alessandro/Indiana

Jeremy Kepner/MIT/LL

**Jose Moreira/IBM**
Manoj Kumar/IBM

Eric Hein/EMU/Lucata

**David Bader/NJIT**

**Paul Burkhardt/DOD**
**Kaushik Velusamy/UMBC**

DARPA: HIVE, SDH, DSSOC, PAPPA, DPRIVE

Phil Ratzloff/SAS
Matt Galati/SAS

Jeffrey Vetter/ORNL

Albert-Jan Yzelman
Danielle Spampinato
Now at Huawei, Zurich

**Roi Lipman**
**RedisLabs, Israel**

**Gabor Szarnyas/**
**Hungarian Academy of Sciences**

**SuiteSparse GraphBLAS Library**
Available as Ubuntu package
Used for Matlab sparse

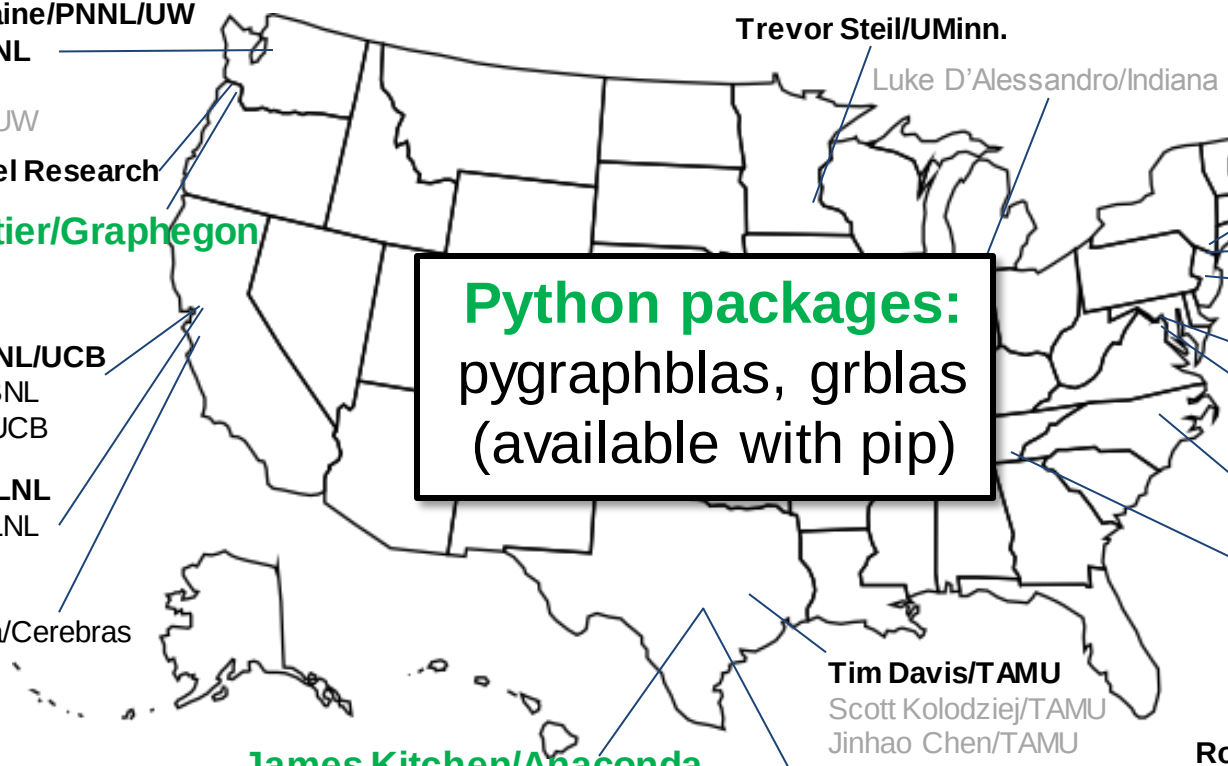
**Tim Davis/TAMU**
Scott Kolodziej/TAMU
Jinhao Chen/TAMU
Et al.

**James Kitchen/Anaconda**
**Eric Welch/Anaconda**

**Intel MKL/Graph Working Group**
**Henry Gabb, Ramesh Peri, et al.**

Intel MKL/Graph Working Group (Russia)
Andrey Nikolaev, Alexey Kukanov, Kirill Voronin

# External Collaborators



**Andrew Lumsdaine/PNNL/UW**
**Jesun Firoz/PNNL**
**Tony Liu/UW**
Kevin Deweese/UW

**Tim Mattson/Intel Research**

**Michel Pelletier/Graphegon**

**Ben Brock/UCB**
**Aydin Buluc/LBNL/UCB**
Oguz Selvitopi/LBNL
James Demmel/UCB

**Roger Pearce/LLNL**
Geoff Sanders/LLNL
Ben Priest/LLNL

Natalia Vassilieva/Cerebras

**Trevor Steil/UMinn.**
Luke D'Alessandro/Indiana

Jeremy Kepner/MIT/LL

**Jose Moreira/IBM**
Manoj Kumar/IBM

Eric Hein/EMU/Lucata

**David Bader/NJIT**

**Paul Burkhardt/DOD**
**Kaushik Velusamy/UMBC**

DARPA: HIVE, SDH, DSSOC, PAPPA, DPRIVE

Phil Ratzloff/SAS
Matt Galati/SAS

Jeffrey Vetter/ORNL

Albert-Jan Yzelman
Danielle Spampinato
Now at Huawei, Zurich

**Python packages:**
pygraphblas, grblas
(available with pip)

**Tim Davis/TAMU**
Scott Kolodziej/TAMU
Jinhao Chen/TAMU
Et al.

**James Kitchen/Anaconda**
**Eric Welch/Anaconda**

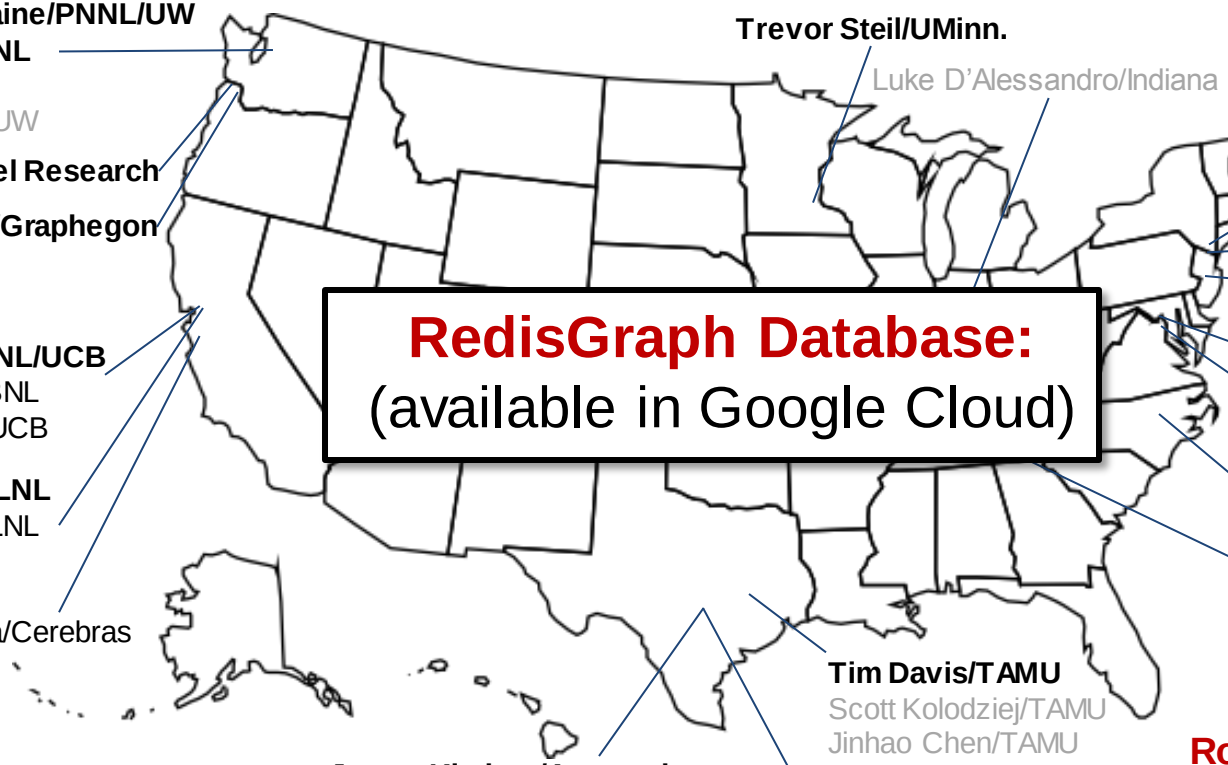
**Roi Lipman**
**RedisLabs, Israel**

**Gabor Szarnyas/**
**Hungarian Academy of Sciences**

Intel MKL/Graph Working Group (Russia)
Andrey Nikolaev, Alexey Kukanov, Kirill Voronin

**Intel MKL/Graph Working Group**
**Henry Gabb, Ramesh Peri, et al.**

# External Collaborators



**Andrew Lumsdaine/PNNL/UW**
**Jesun Firoz/PNNL**
**Tony Liu/UW**
Kevin Deweese/UW

**Tim Mattson/Intel Research**

**Michel Pelletier/Graphegon**

**Ben Brock/UCB**
**Aydin Buluc/LBNL/UCB**
Oguz Selvitopi/LBNL
James Demmel/UCB

**Roger Pearce/LLNL**
Geoff Sanders/LLNL
Ben Priest/LLNL

Natalia Vassilieva/Cerebras

**Trevor Steil/UMinn.**
Luke D'Alessandro/Indiana

Jeremy Kepner/MIT/LL

**Jose Moreira/IBM**
Manoj Kumar/IBM

Eric Hein/EMU/Lucata

**David Bader/NJIT**

**Paul Burkhardt/DOD**
**Kaushik Velusamy/UMBC**

DARPA: HIVE, SDH, DSSOC, PAPPA, DPRIVE

Phil Ratzloff/SAS
Matt Galati/SAS

Jeffrey Vetter/ORNL

Albert-Jan Yzelman
Danielle Spampinato
Now at Huawei, Zurich

**RedisGraph Database:**
(available in Google Cloud)

**Tim Davis/TAMU**
Scott Kolodziej/TAMU
Jinhao Chen/TAMU
Et al.

**James Kitchen/Anaconda**
**Eric Welch/Anaconda**

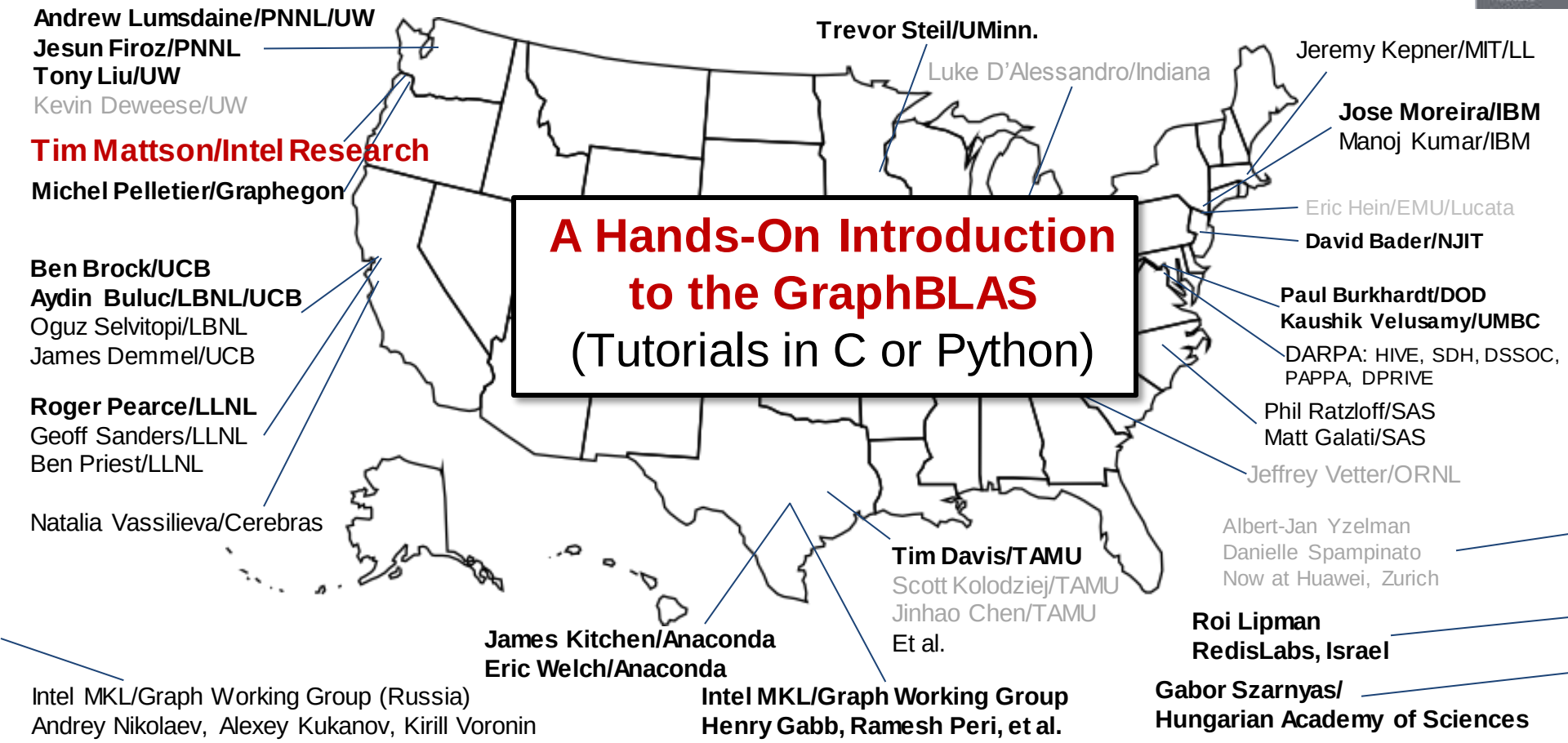
**Roi Lipman**
**RedisLabs, Israel**

**Gabor Szarnyas/**
**Hungarian Academy of Sciences**

Intel MKL/Graph Working Group (Russia)
Andrey Nikolaev, Alexey Kukanov, Kirill Voronin

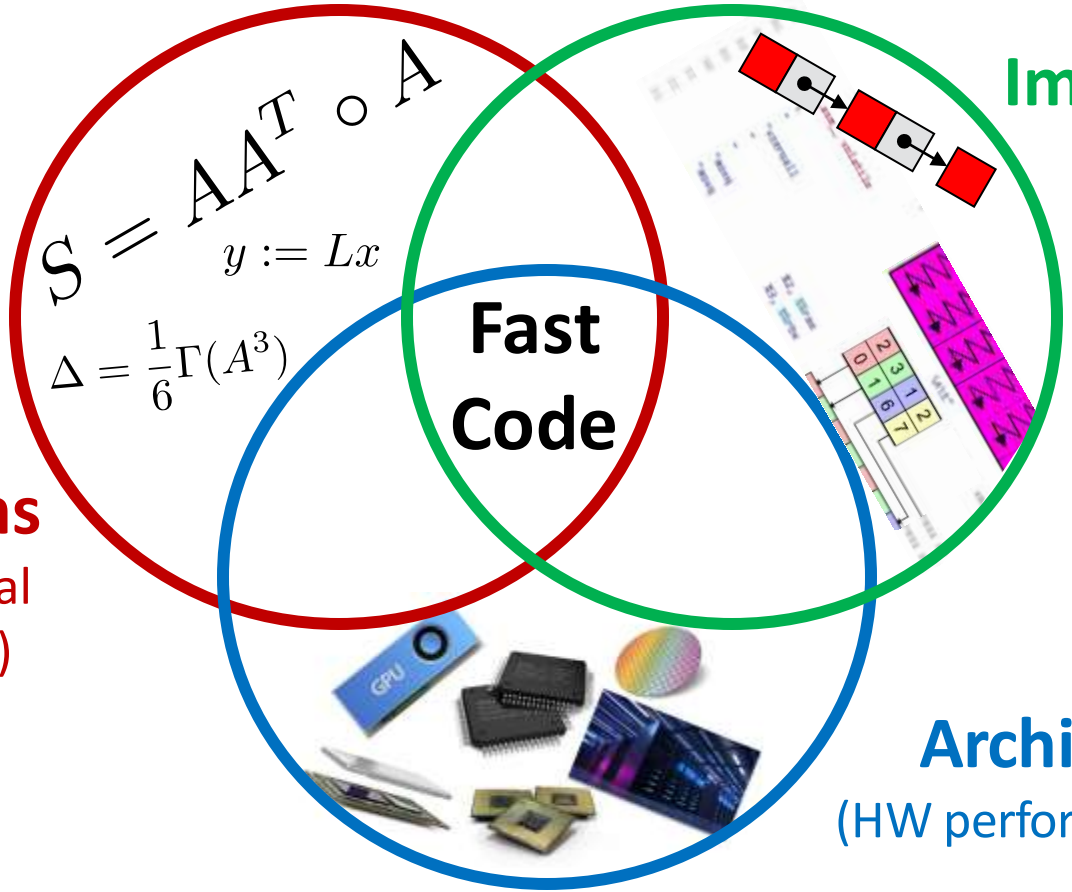
**Intel MKL/Graph Working Group**
**Henry Gabb, Ramesh Peri, et al.**

# External Collaborators




**A Hands-On Introduction to the GraphBLAS**
(Tutorials in C or Python)

# Fast code depends on three interdependent ingredients




**Algorithms**
(mathematical descriptions)

**Implementations**
(primitive ops and data structures)

**Architectures**
(HW performance models)

# Backup slides

# Promotional

The Department of Defense and U.S. intelligence community have an overwhelming data collection capability, and commanders and warfighters in the field rely on the timely processing of data. But they lack processing power, and most AI/ML techniques are difficult to implement efficiently in an ever-changing hardware landscape. What's more, too few engineers have the expertise to optimize this data-intensive software for the latest hardware.

The result? Decision makers can't make the most of the data available to them.

I'm Scott McMillan, principal investigator for Spiral AI/ML, an SEI project to automate code generation for data-intensive computations while simultaneously optimizing for targeted hardware. In partnership with Carnegie Mellon University, we seek to build a hardware-software co-optimization system that provides for high-performance, data-intensive computing across existing and future DoD hardware platforms. By providing a high-level interface based on GraphBLAS that allows application developers to develop for any hardware, the Spiral code generation system promises to speed the development of high-performance AI and ML applications and enable the rapid deployment of those new capabilities on new hardware platforms at levels of performance equaling or exceeding hand-tuned software at in a fraction of the time and cost.

To learn more about GraphBLAS and how Spiral automatically generates high-performance code email my team through info@sei.cmu.edu.

# Spiral/AIML: Co-optimization for High-Performance, Data-Intensive Computing in Resource Constrained Environments

*"Rapidly delivering artificial intelligence to a combat zone won't be easy." Col. Drew Cukor, USMC.*

**Problem(s)**

- Complexity is increasing in computing architectures.
- Mission cost, size, weight, and power (CSWAP) constraints drive increasing use of FPGAs and ASICs (more complexity).
- Achieving performance from these platforms is hard.
- Achieving performance from data-intensive applications (graphs, ML, AI) is hard.

**Solution**

- Automatic code generation for data-intensive computations
- Simultaneous, automatic co-optimization of hardware within CSWAP constraints

**Approach**

- Identify common AI/ML/Graph computational primitives.
- Encode knowledge about efficient implementations of graph, ML, and AI computational primitives into Spiral code-gen technology.
- Develop hardware performance models allowing Spiral to choose between components satisfying CSWAP requirements.

# Spiral/AIML: Co-optimization for High-Performance, Data-Intensive Computing in Resource-Constrained Environments

*"Rapidly delivering artificial intelligence to a combat zone won't be easy." Col. Drew Cukor, USMC.*

**Put more simply…**

- Optimizing software is *hard*.
- Optimizing hardware is *hard*.
- Co-optimization = $hard^2$.
- Automating it is $hard^n$, for some n > 2.

The goal: high-performance analytic platforms *faster* and *cheaper*.