Research Review 2021

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

November 2021

Scott McMillan
Fast code depends on three interdependent ingredients:

- **Algorithms** (mathematical descriptions)
  \[ S = AAT \circ A \]
  \[ y := Lx \]
  \[ \Delta = \frac{1}{6} \Gamma(A^3) \]

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

- **Architectures** (HW performance models)
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.

“Rapidly delivering artificial intelligence to a combat zone won’t be easy.” Col. Drew Cukor, USMC.
What is Spiral?

Traditionally

High performance library optimized for given platform

Spiral Approach

High performance library optimized for given platform
What is Spiral?

Traditionally:
High performance library optimized for given platform

Spiral Approach:
Spiral
High performance library optimized for given platform

Comparable performance
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
Foundational GraphBLAS References

Mathematical Foundations of the GraphBLAS

Jeremy Kepner (MIT Lincoln Laboratory Supercomputing Center), Peter Aaltonen (Indiana University),
David Bader (Georgia Institute of Technology), Aydin 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

Aydin Buluç†, Tim Mattson‡, Scott McMillan§, Jose 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

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

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...
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

Click probability

Top MLP

Concat

Pairwise Interaction

Bottom MLP

Embedding table 1

Embedding table M

Numerical feature 1

Numerical feature N

Categorical feature 1

Categorical feature M
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**

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

IR
of a compiler

IR
mapping to HW specific instruction sets

Mathematical Descriptions

SPL specification

Expansion + backtracking

SPL (dataflow)
expression

branch + search (algorithmic)

Recursive descent

Σ-SPL (loop)
expression

branch + search ("loops"-ish)

Confluent term rewriting

Optimized Σ-SPL
expression

branch + search (ISA)

Recursive descent

Abstract code

Optimized abstract code (icode)

Confluent term rewriting

C code

Recursive descent

Expert System

Performance Models

SPL (dataflow)
expression

branch + search (algorithmic)

Recursive descent

Optimized Σ-SPL
expression

branch + search (ISA)

Confluent term rewriting

Optimized abstract code (icode)

Recursive descent

C code
SPIRAL: Code Generation Stages/Additions

AST

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

Optimization Phase

SPL specification

Expansion + backtracking
branch + search (algorithmic)

Recursion descent

SPL (dataflow) expression

Confluent term rewriting
branch + search (loops)

Recursion descent

Σ-SPL (loop) expression

Confluent term rewriting
branch + search (ISA)

Recursion descent

Optimized Σ-SPL expression

IR

of a compiler

Abstract code

IR

mapping to HW-specific instruction
sets

Optimized abstract code (icode)

C code

Additions

Parsing GraphBLAS syntax.
Support for new sparse data formats, semiring operators, and sizes specified at run-time.

Computed graphs (DAGs) of operations to perform optimizations (GBTLX, Oct20)

New representation for bilinear (binary, even n-ary) operators allowing for mxm, mxv, ewise, apply, etc.

Generated kernels for sparse linear algebra applications
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

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)

---

**Speedup of GBTL++ and GKC SpMV over GBTL v3.0**

![Graph showing speedup comparison between GBTL Optimized and GKC Dense](image)

**Legend:**

- GBTL Optimized
- GKC Dense
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)
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
Intel MKL/Graph Working Group (Russia)
Andrey Nikolaev, Alexey Kukanov, Kirill Voronin
Intel MKL/Graph Working Group
Henry Gabb, Ramesh Peri, et al.
Spiral AI/ML: Co-optimization for High-Performance, Data-Intensive Computing
© 2021 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution.
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

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

SuiteSparse GraphBLAS Library
Available as Ubuntu package
Used for Matlab sparse

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

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

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

Eric Hein/EMU/Lucata
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.

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

Gabor Szarnyas/
Hungarian Academy of Sciences

Roi Lipman
RedisLabs, Israel

SUITEPARSE GRAPHBLAS LIBRARY
AVAILABLE AS UBUNTU PACKAGE
USED FOR MATLAB SPARSE
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

RedisGraph Database:
(available in Google Cloud)

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

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

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

James Kitchen/Anaconda
Eric Welch/Anaconda

Tim Davis/TAMU
Alexey Kukanov, Kirill Voronin

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, D+RIVE

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

RedisGraph Database:
(available in Google Cloud)

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

RedisGraph Database:
(available in Google Cloud)

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

RedisGraph Database:
(available in Google Cloud)
Fast code depends on three interdependent ingredients:

- **Algorithms** (mathematical descriptions)
  
  \[ S = AAT \circ A \]
  
  \[ y := Lx \]
  
  \[ \Delta = \frac{1}{6} \Gamma(A^3) \]

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

- **Architectures** (HW performance models)