

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

## Problems

- The need exists for increased computational power to process, exploit, and disseminate information for decision makers.
- Massive amounts of information, along with AI/ML algorithms, generate data and computational-intensive applications.
- Implementing these applications efficiently on increasingly complex HW/SW architectures is challenging.
- Too few engineers have the expertise to optimize algorithms for the wide variety of hardware currently available.

## Solution

- Automatic code generation for data-intensive computations
- Simultaneous, automatic co-optimization for targeted hardware

## Approach

- Identify and encode data-intensive compute primitives into CMU's SPIRAL code generation technology.
- Develop and encode hardware performance models into Spiral.
- Use Spiral to co-optimize for a set of target hardware platforms.

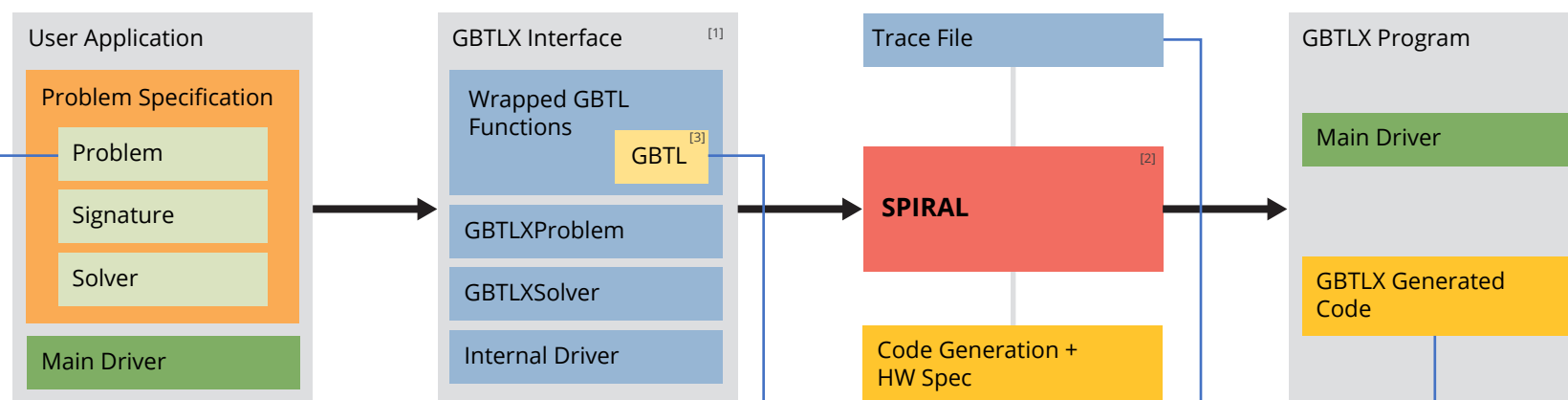
**Graph algorithms in the language of linear algebra** supports a rich notation for specifying graph, ML and AI algorithms. For example, counting triangles in graph  $L$ :

$$\Delta = ||L \cdot X (L + \wedge L)||$$

includes use of semiring algebraic operations and masked matrix multiplies.

[6]

**Hardware-software co-optimization** promises timely, high-performance, and cost-effective implementation and re-implementation of AI/ML workloads on new **DoD** hardware platforms.



**GBTX implements the GraphBLAS specification** that allows simpler implementation of the math in code:

```
uint64_t triangle_count(Matrix<bool> const &L) {
    Matrix<uint64_t> B(L.nrows(), L.ncols());

    // Masked matrix multiply: B = L .* (L +. ^ L)
    mxm(B, L, NoAccum(), PlusAndSemiring<uint64_t>(), L, L);

    //Perform reduction: ||B||
    uint64_t count;
    reduce(count, NoAccum(), PlusMonoid<uint64_t>(), B);
    return count;
}
```

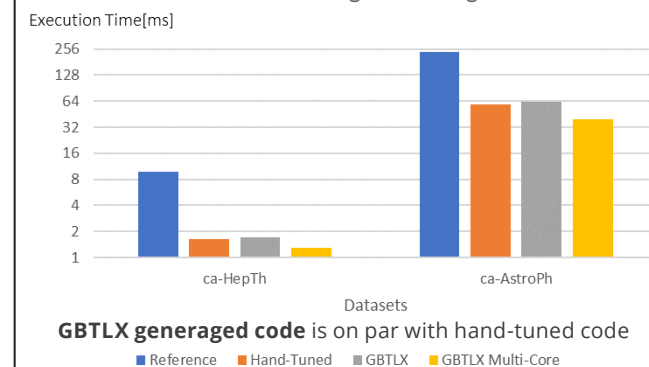
[3,4]

**Spiral wraps GBTX functions** to build a trace file used for analysis during code generation:

```
spiral_session := [
    rec(op := "triangle_count"), //function name
    rec(op := "MatrixCreation", row:= 9877, col:= 9877,
        ptr := 0x7fffff45bb60, mat = 0x7fffff45bb60),
    rec(op := "Matrix Multiplication",
        output = IntHexString("0x7fffff45bb60") ,
        mask = IntHexString("0x7fffff45ba30"),
        inputA = IntHexString("0x7fffff45bb30"),
        inputB = IntHexString("0x7fffff45bb30"),
        semiring = "PlusAnd"),
    rec(op := "reduce(matrix->scalar)",
        /*many more arguments*/),
];
```

[1]

Performance of GBTX on Triangle Counting



**GBTX generated code** is on par with hand-tuned code

■ Reference ■ Hand-Tuned ■ GBTX ■ GBTX Multi-Core

[1,5]

## References

1. S. Rao, A. Kutuluru, S. McMillan, F. Franchetti, "GBTX: A First Look", in 2020 IEEE High Performance Extreme Computing Conference (HPEC), 2020. **Outstanding Student Paper Award.**
2. SPIRAL Project, Version 8.1.2. Available at <https://www.spiral.net>.
3. GraphBLAS Template Library (GBTX), Version 3.0. Available at <https://github.com/cmu-sei/gbtl>, June 2020.
4. A. Buluç, T. Mattson, S. McMillan, J. Moreira, and C. Yang, "Design of the GraphBLAS API for C," in 2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp. 643–652, 2017.
5. T. M. Low, V. N. Rao, M. Lee, D. Popovici, F. Franchetti, and S. McMillan, "First look: Linear algebra-based triangle counting without matrix multiplication," in 2017 IEEE High Performance Extreme Computing Conference (HPEC), pp. 1–6, 2017.
6. J. Kepner, D. Bader, A. Buluç, F. Franchetti, J. Gilbert, A. Lumsdaine, T. Mattson, S. McMillan, et al., "Mathematical Foundations of the GraphBLAS," in 2016 IEEE High Performance Extreme Computing Conference (HPEC), pp. 1–9, 2016.

Copyright 2020 Carnegie Mellon University.

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.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Internal use:\* Permission to reproduce this material and to prepare derivative works from this material for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use:\* This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other external and/or commercial use. Requests for permission should be directed to the Software Engineering Institute at [permission@sei.cmu.edu](mailto:permission@sei.cmu.edu).

\* These restrictions do not apply to U.S. government entities.

DM20-0900