CARTS: Enabling Event-Driven Task and Data Block Compilation for Distributed HPC

Rafael Andres Herrera Guaitero

PhD Candidate
University of Delaware

Saturday, March 1

Contributors

- **Joseph B. Manzano Franco**
  Pacific Northwest National Laboratory
- **Joshua D. Suetterlein**
  Pacific Northwest National Laboratory
- **Xiaoming Li**
  University of Delaware
- **Andres Marquez**
  Pacific Northwest National Laboratory

Outline

MOTIVATION & CONTEXT

CARTS: COMPILER FOR ARTS

FUTURE WORK

Motivation & Context - I

- **Evolving Architectures:** Modern HPC and AI/ML workloads demand architectures that efficiently leverage memory systems.
- **Hardware Heterogeneity:** The rise of multi-core CPUs, GPUs, and specialized accelerators creates complexity in managing resources.
- **Performance Pressure:** Increased demand for efficient concurrency, synchronization, and communication in large-scale systems.

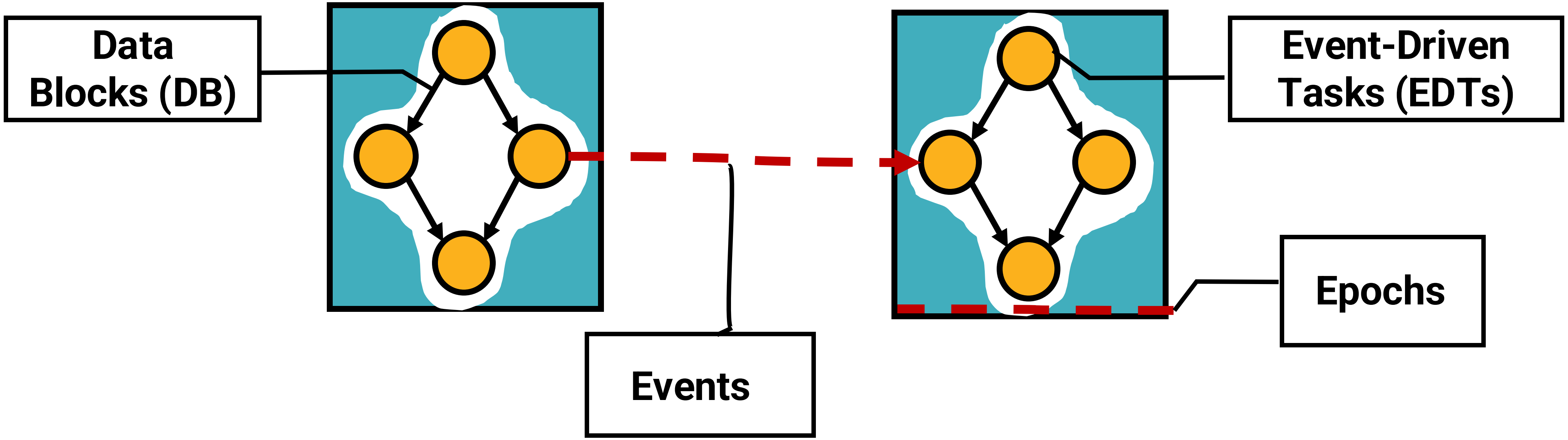
Motivation & Context - II

- **OpenMP:** A widely adopted standard in C/C++ for specifying parallel regions, tasks, and dependencies through intuitive pragmas.
- **MLIR:** A flexible intermediate representation framework that supports multi-level optimizations and custom dialects, bridging high-level abstractions with low-level execution details.
- **ARTS (Abstract RunTime System)** is a runtime infrastructure engineered for fine-grained concurrency and efficient task scheduling in distributed systems

Motivation & Context - III

Asynchronous RunTime System (ARTS)

It provides users with a distributed global address space, a distributed memory model, and synchronization constructs to write efficient applications on a massively parallel system.

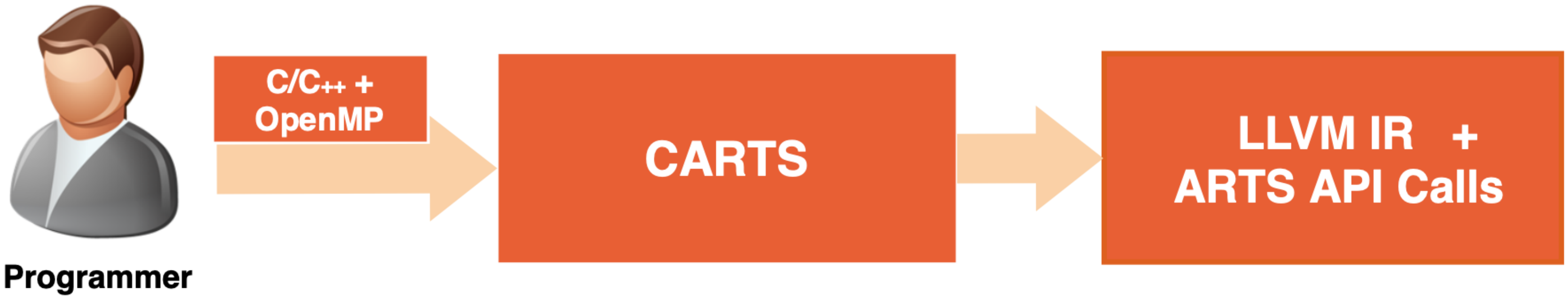


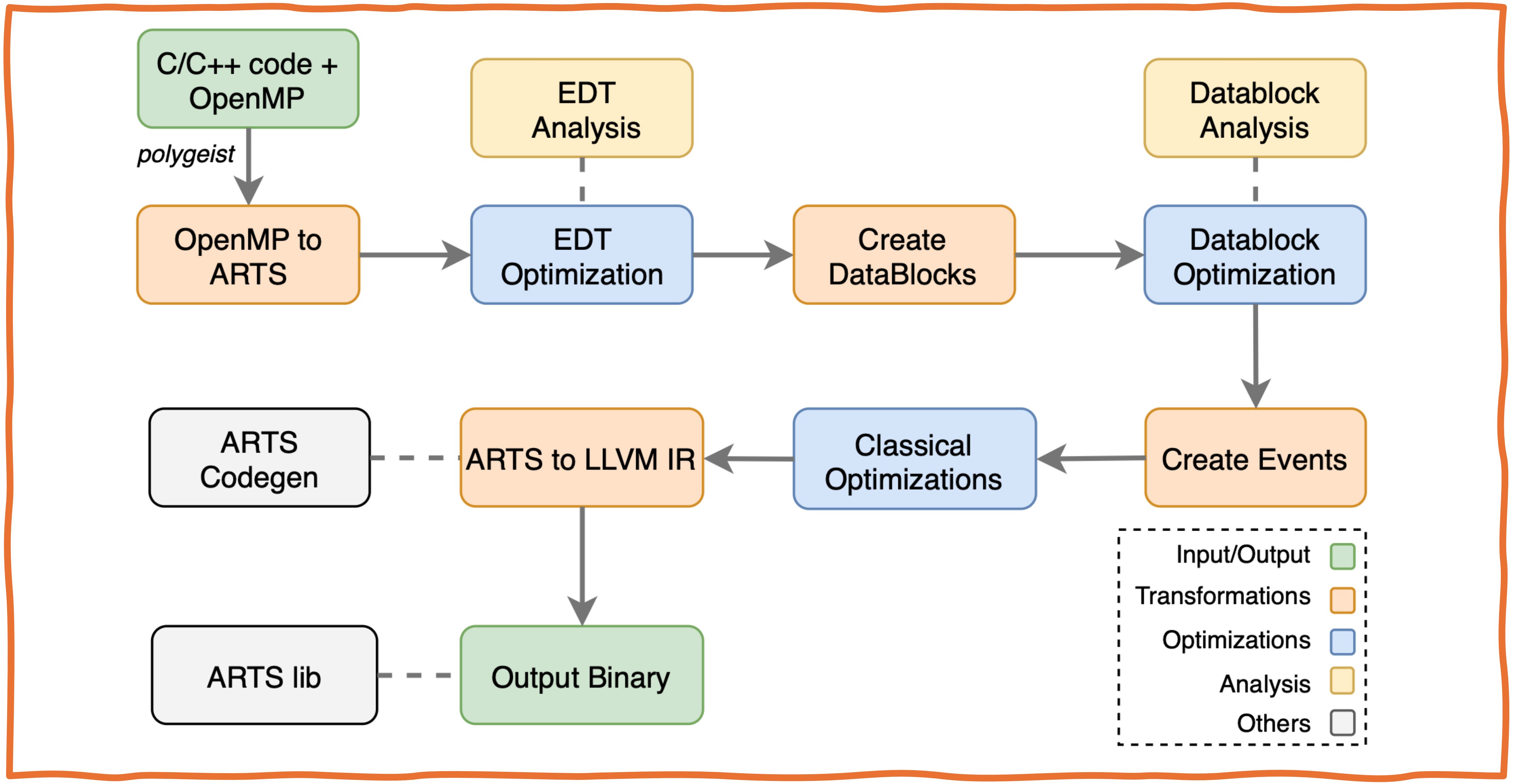
What is the need?

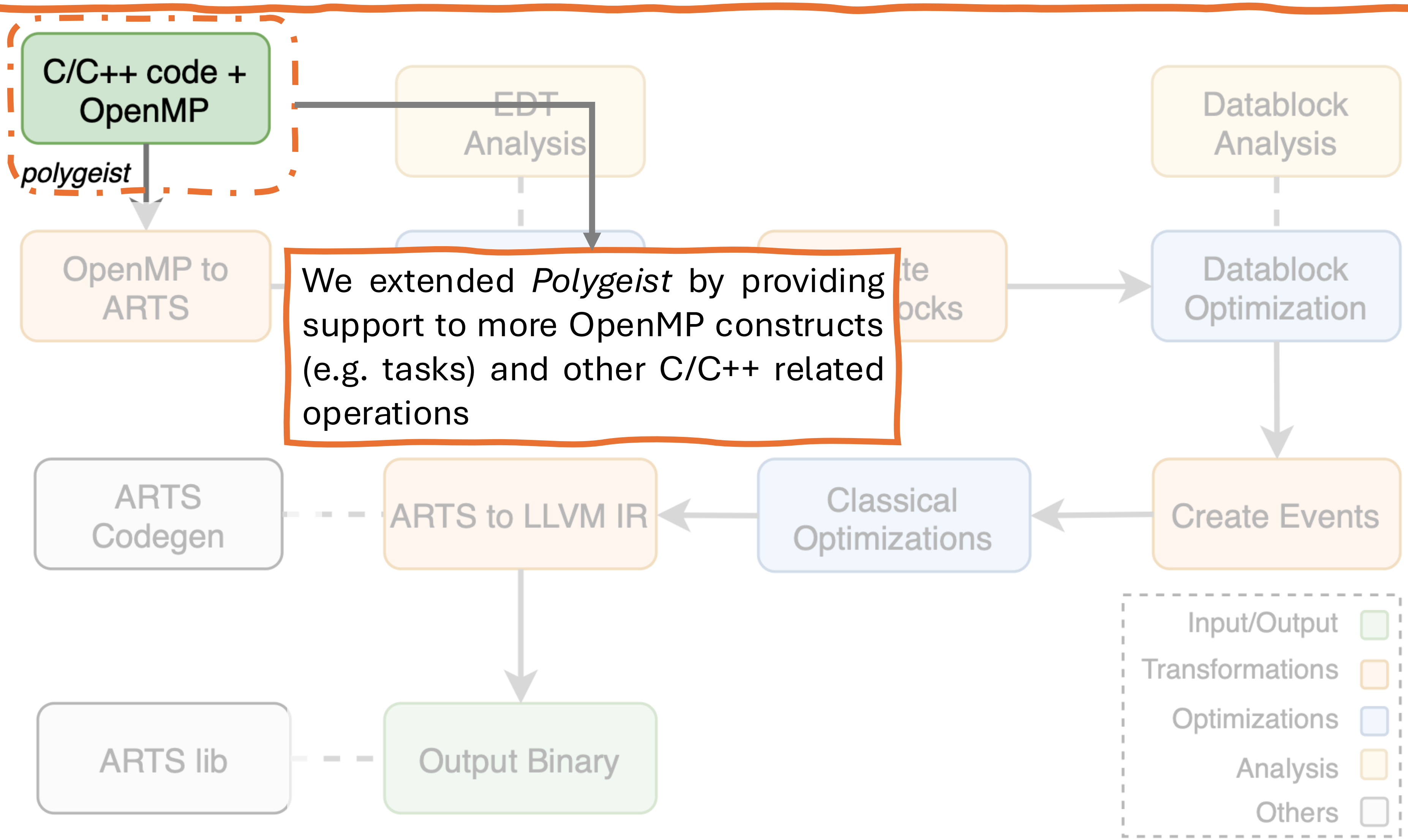
Compiler Infrastructure for ARTS

Compiler for ARTS

A task-centric compiler pipeline that transforms OpenMP-annotated C/C++ code into an ARTS-friendly format using MLIR.

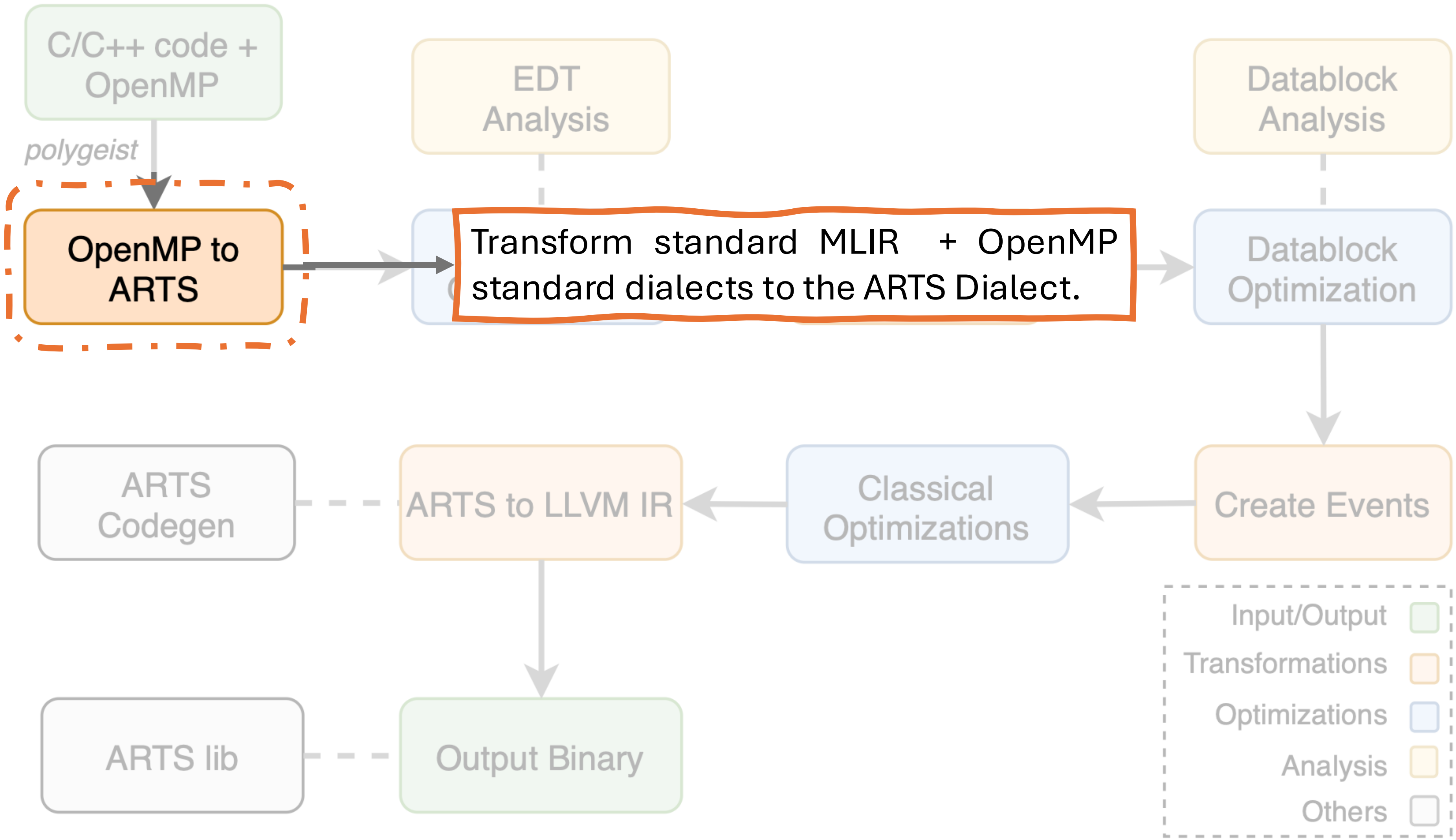



C/C++ code + OpenMP
polygeist
OpenMP to ARTS
EDT Analysis
EDT Optimization
Create DataBlocks
Datablock Analysis
Datablock Optimization
Create Events
Classical Optimizations
ARTS to LLVM IR
ARTS Codegen
Output Binary
ARTS lib
Input/Output 🟩
Transformations 🟧
Optimizations 🟦
Analysis 🟨
Others ⬜



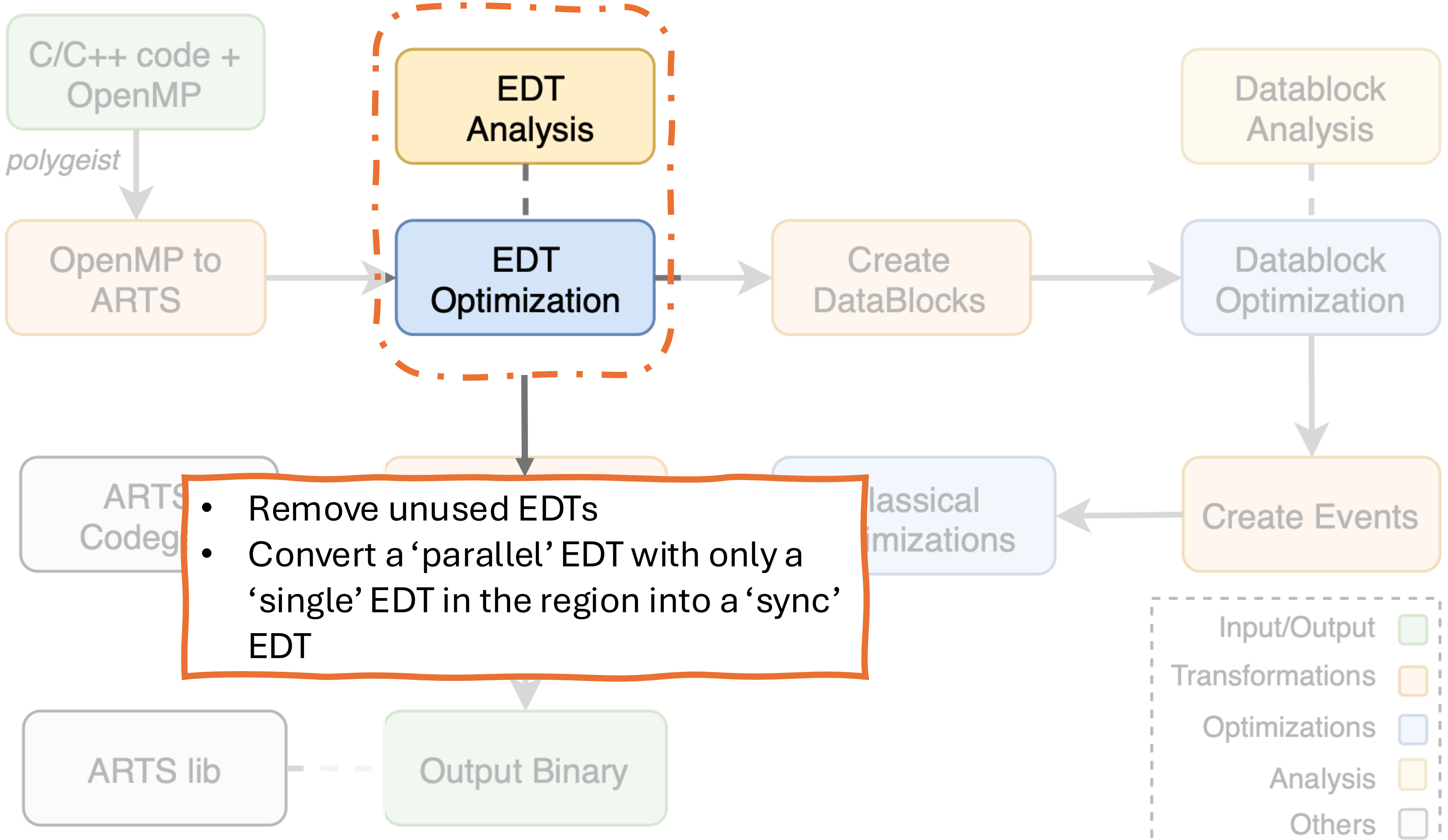
We extended *Polygeist* by providing support to more OpenMP constructs (e.g. tasks) and other C/C++ related operations

1 - C/C++ + OpenMP support



Transform standard MLIR + OpenMP standard dialects to the ARTS Dialect.

2 – Mapping to ARTS


C/C++ code + OpenMP

polygeist

OpenMP to ARTS

EDT Analysis

EDT Optimization

Create DataBlocks

Datablock Analysis

Datablock Optimization

Create Events

lassical imizations

ARTS Codeg

ARTS lib

Output Binary

Input/Output 🟩

Transformations 🟧

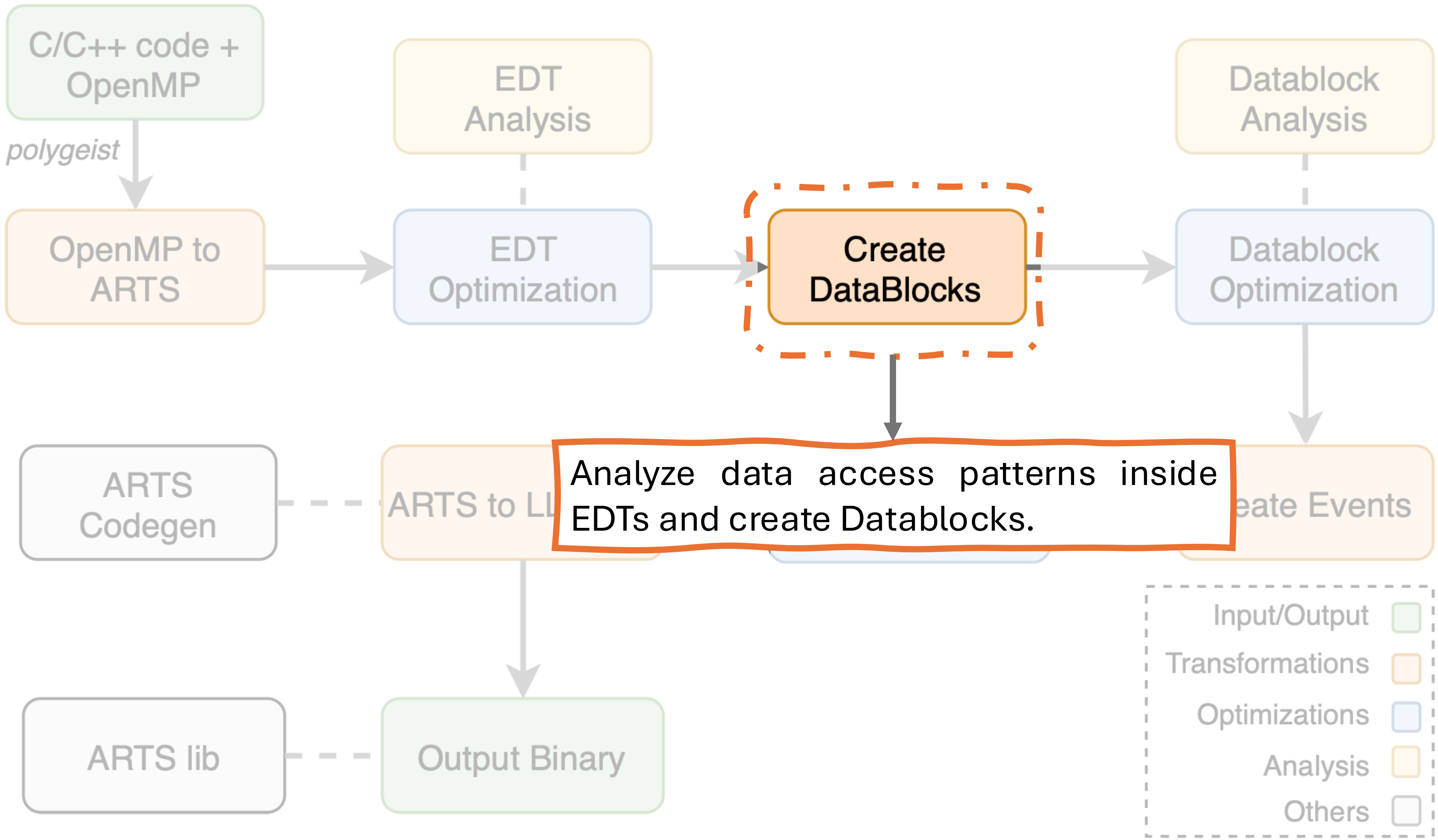
Optimizations 🟦

Analysis 🟨

Others ⬜

- Remove unused EDTs
- Convert a ‘parallel’ EDT with only a ‘single’ EDT in the region into a ‘sync’ EDT

3 – EDTs Analysis and Optimization



C/C++ code + OpenMP

polygeist

OpenMP to ARTS

EDT Analysis

EDT Optimization

Create DataBlocks

Datablock Analysis

Datablock Optimization

eate Events

ARTS Codegen

ARTS to LL

ARTS lib

Output Binary

Input/Output 🟩

Transformations 🟧

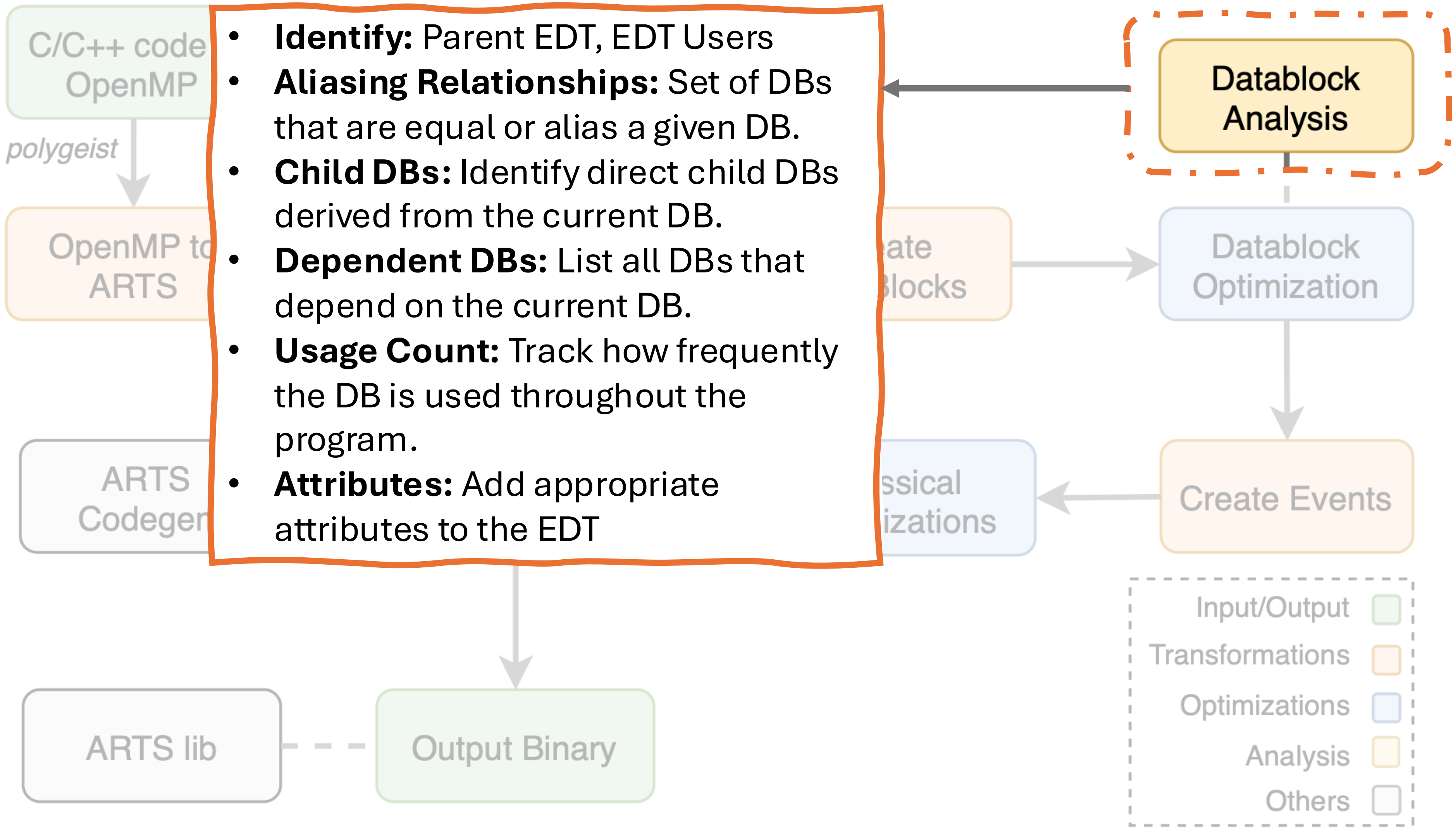
Optimizations 🟦

Analysis 🟨

Others □

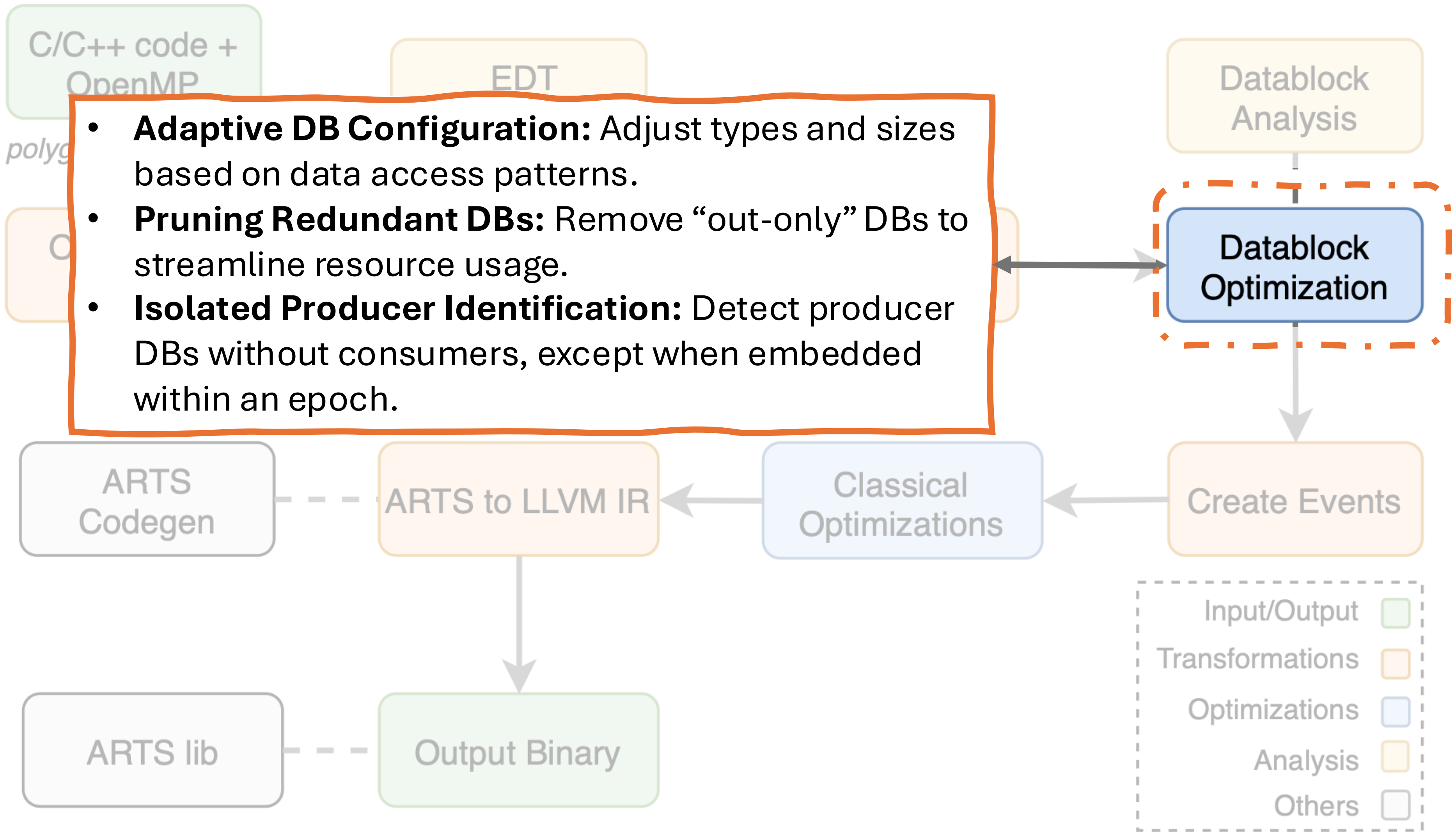
Analyze data access patterns inside EDTs and create Datablocks.

4 – Data Blocks Identification



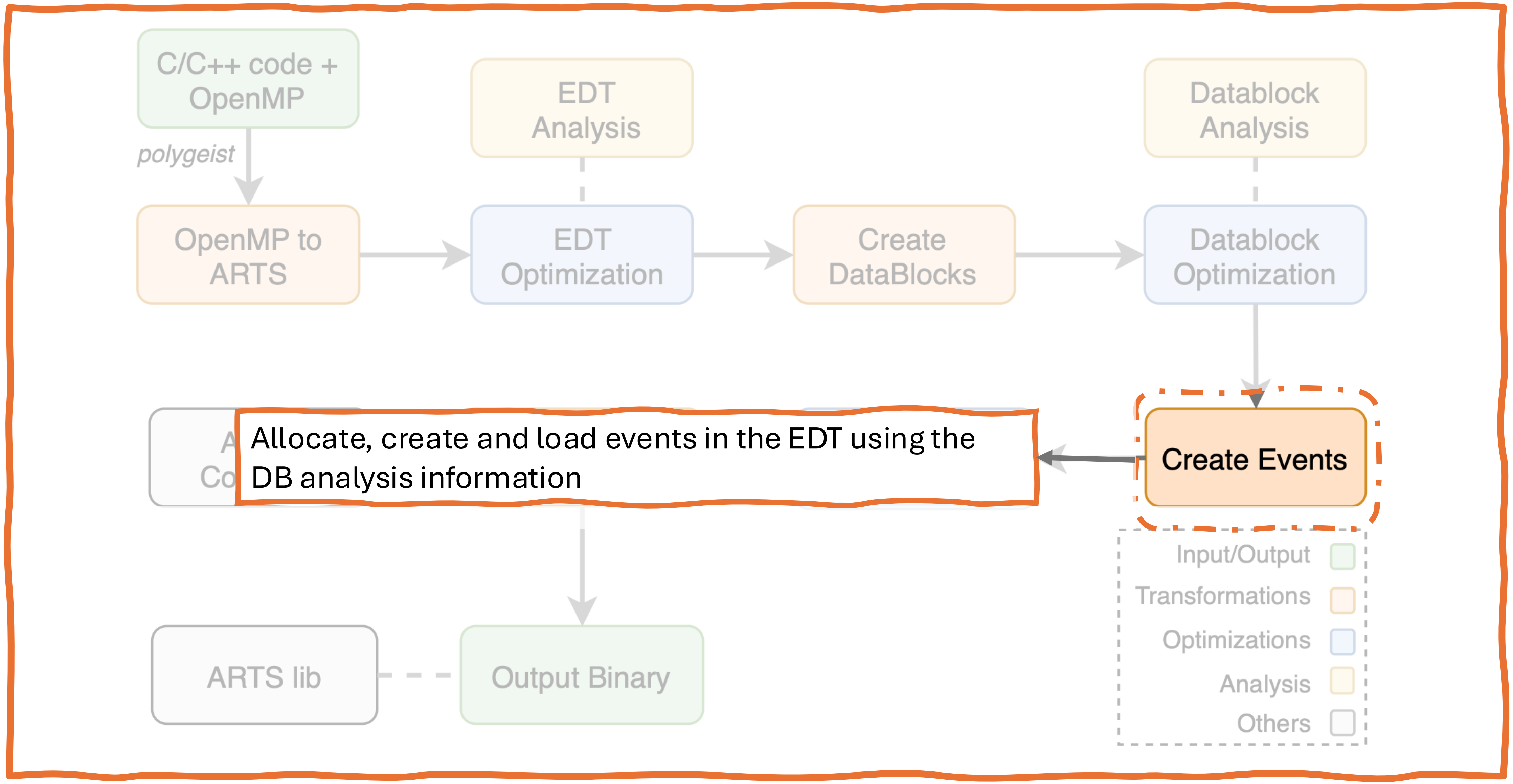
- **Identify:** Parent EDT, EDT Users
- **Aliasing Relationships:** Set of DBs that are equal or alias a given DB.
- **Child DBs:** Identify direct child DBs derived from the current DB.
- **Dependent DBs:** List all DBs that depend on the current DB.
- **Usage Count:** Track how frequently the DB is used throughout the program.
- **Attributes:** Add appropriate attributes to the EDT

5 – Data Blocks Analysis



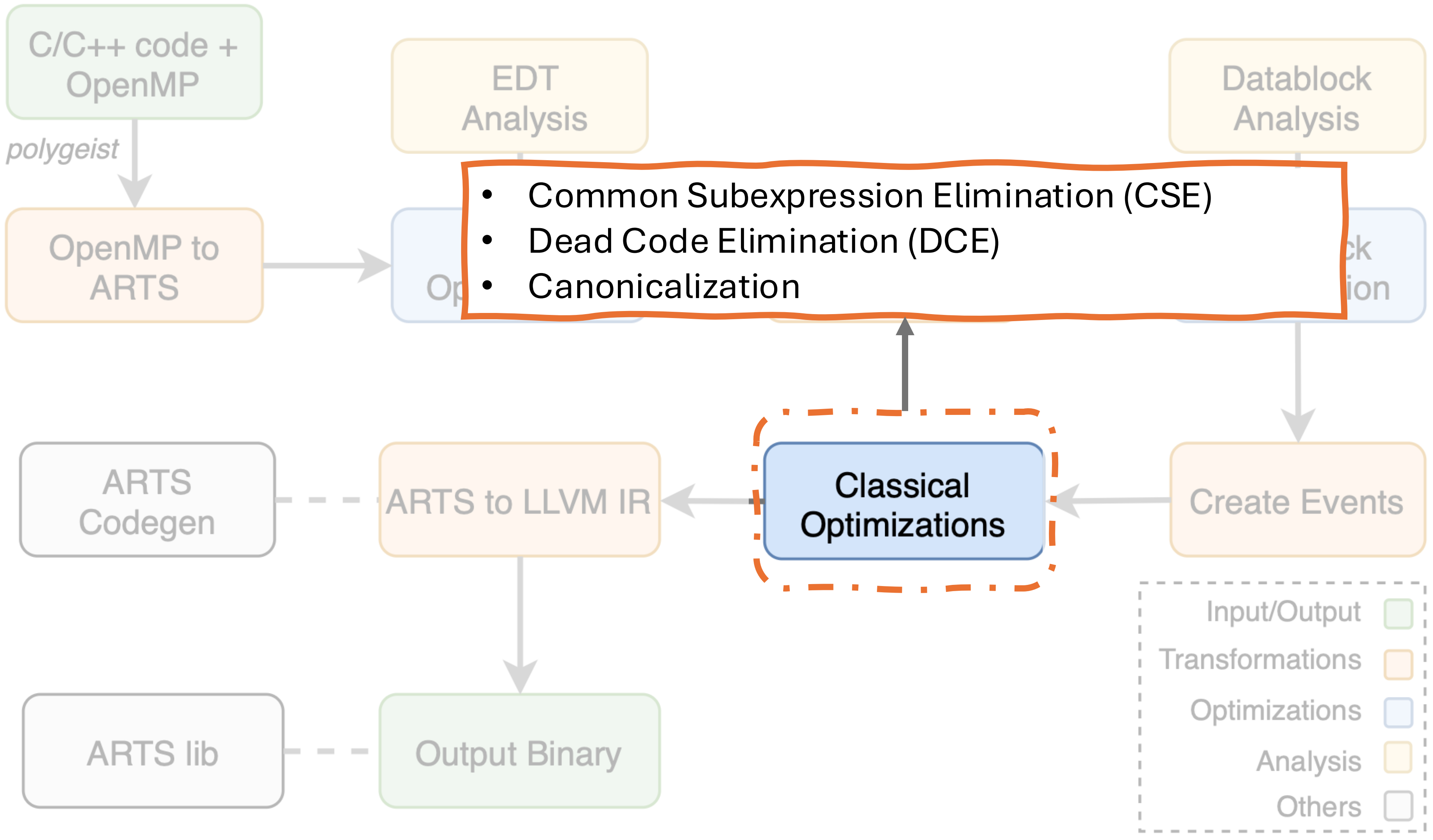
- **Adaptive DB Configuration:** Adjust types and sizes based on data access patterns.
- **Pruning Redundant DBs:** Remove “out-only” DBs to streamline resource usage.
- **Isolated Producer Identification:** Detect producer DBs without consumers, except when embedded within an epoch.

6 – Data Blocks Optimization



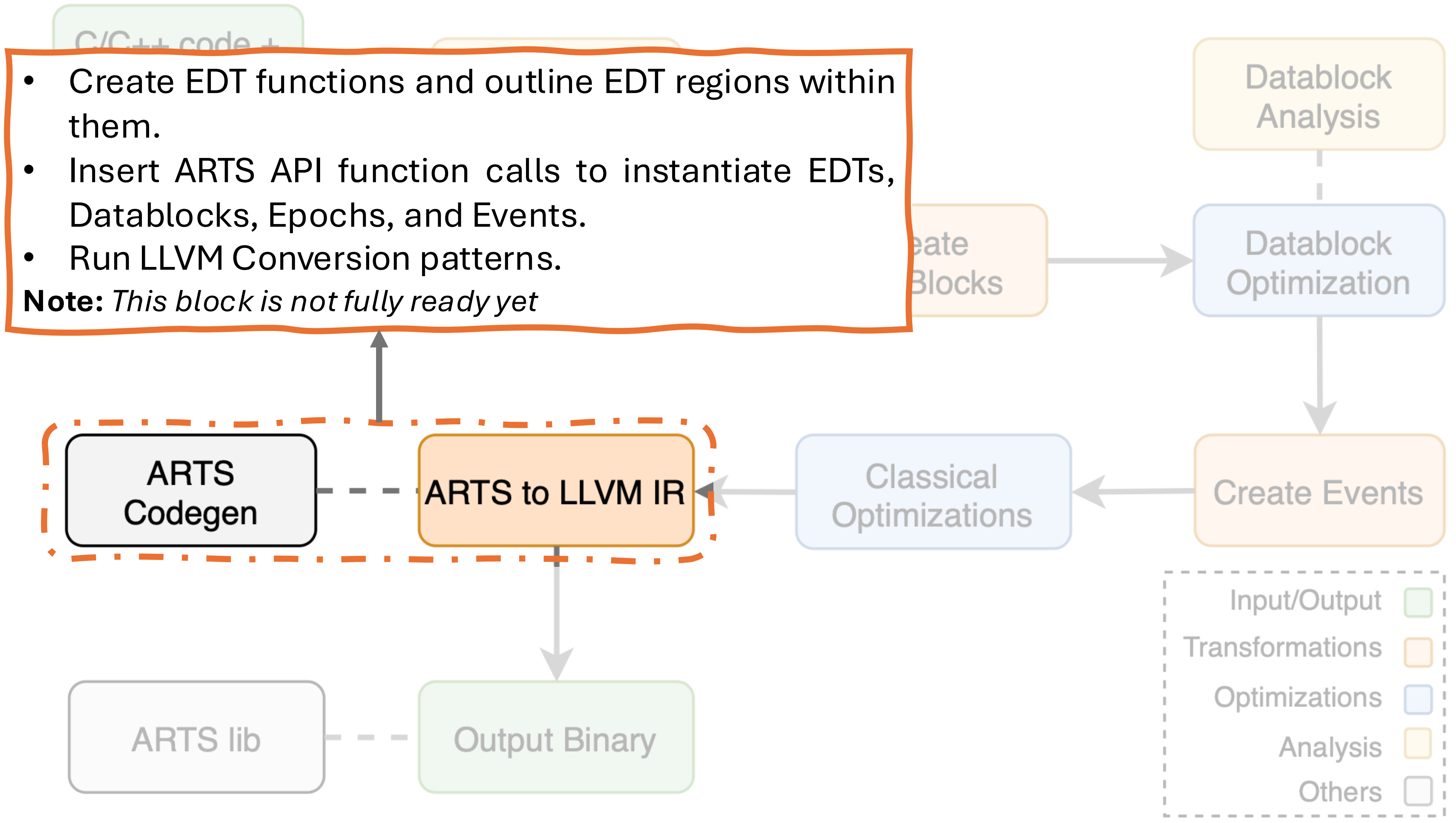
Allocate, create and load events in the EDT using the DB analysis information

7 – Events Creation



- Common Subexpression Elimination (CSE)
- Dead Code Elimination (DCE)
- Canonicalization

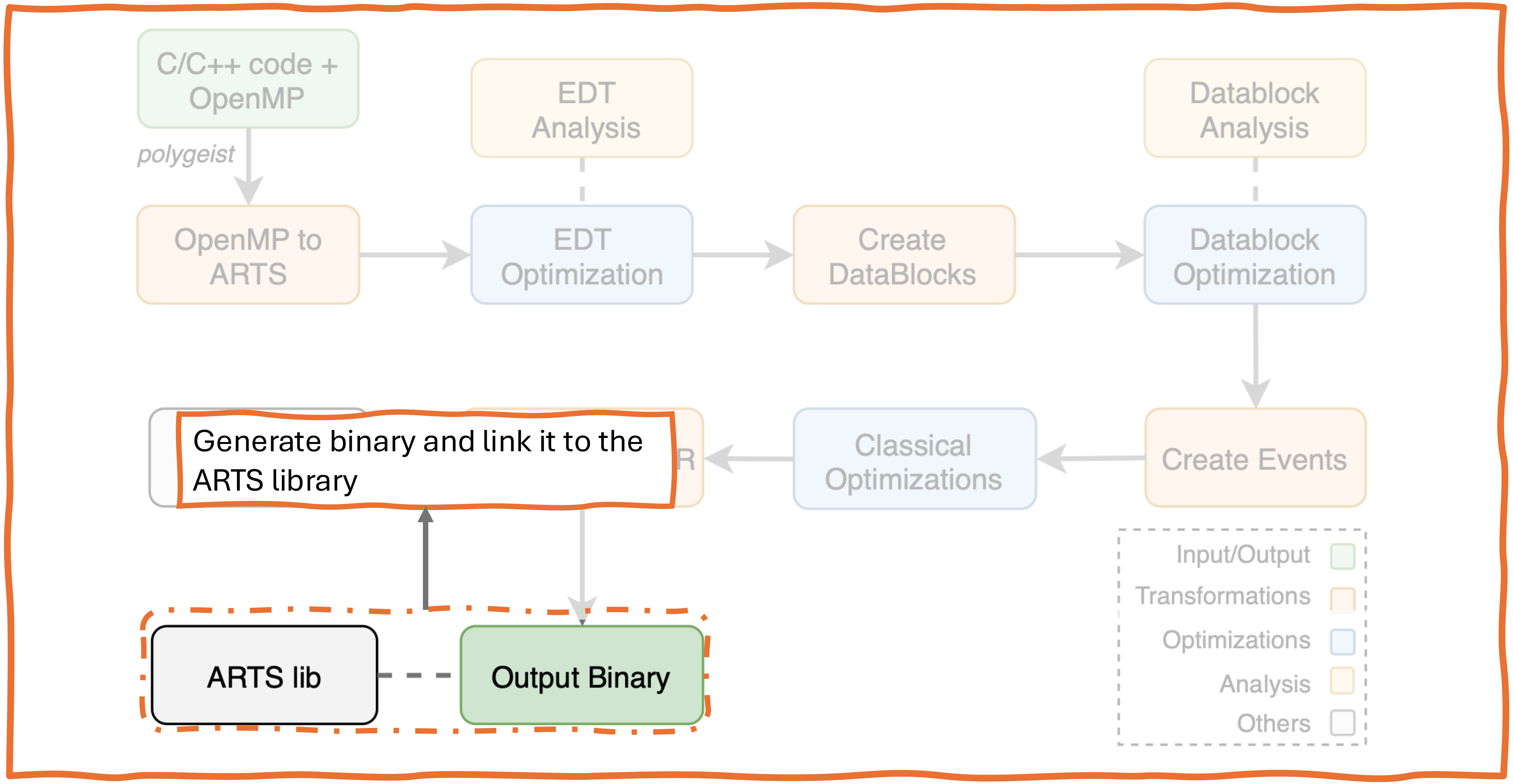
8 – Other Optimizations



- Create EDT functions and outline EDT regions within them.
- Insert ARTS API function calls to instantiate EDTs, Datablocks, Epochs, and Events.
- Run LLVM Conversion patterns.

Note: *This block is not fully ready yet*

9 – Conversion to LLVM IR



Generate binary and link it to the ARTS library

10 - Output

Future Work

- Test the infrastructure with different benchmarks.
- Provide support to more OpenMP Constructs (e.g. for, barriers, locks…)
- Advanced Transformation Passes.
- Memory-Centric Optimizations based on a Memory cost model
- Feedback-Directed Compilation.
- Domain-Specific Extensions

Acknowledgments

- This work is supported by the US DOE Office of Science project "Advanced Memory to Support Artificial Intelligence for Science" at PNNL. PNNL is operated by Battelle Memorial Institute under Contract DEAC06-76RL01830.

- LLVM Foundation for the travel award.

Thank you!
Any questions?

Rafael Andres Herrera Guaitero

rafaelhg@udel.edu

Phd Candidate at University of Delaware