A COMPILE AGNOSTIC AND ARCHITECTURE AWARE PREDICTIVE MODELING FRAMEWORK FOR KERNELS

Problem Statement
- Predictive modeling becomes increasingly difficult as heterogeneous architectures are adopted and many compilers are available
- Application codes need to target multiple architectures, but it is not feasible to explore a complete optimization search space.
- A predictive modeling framework should be able to handle any binary targeting an architecture regardless of compiler.
- Optimizations and transformations applied can take one nested loop and generate thousands of variants, all of which may not be correct.

Instruction Set Feature Extraction

Decouple the compiler-architecture pair problem by:
- Considering the final code emitted by the compiler after all optimizations are applied.
- Handling heterogeneous codes by specializing instruction feature mapping for different Instruction Set Architectures (ISA)

Map each instruction to a common instruction format for learning (ComIL), storing the ISA target as a feature as well as:
- Data: type (integer, floating point), size (in bytes), vector size
- Memory: # bytes read, # bytes stored, push/pop instructions
- Control Flow: call, return, jump, conditional jump
- Computation: add, sub, mul, div, bitwise, flags, special, cond

Experiment Setup
- PolyBench [1] (30) and ARES LoopSuite (4) kernels
- Used RAJA [2] as tuning framework for code transformations – some transformations applied could be unsafe (forced parallelization)
- Compiled each kernel version with GCC 7.1 and Clang 4.0
- Train models with all-but-one PolyBench kernel (cross validation)
- Evaluate with LoopSuite and the remaining PolyBench kernel
- Modeling Objectives: execution time and maximum absolute error

Evaluation
- Generate all models (90) (gcc, clang, gcc+clang)x(PolyBench)
- Process unseen program versions through each model
- Evaluate each model for error (mean relative error) and ordering by rank of the optimization speedup curve (r²)

Conclusion and Future Work
- We show that we are able to decouple the compiler-architecture pair by constructing and evaluating models constructed from PolyBench kernels. Predicted objective outcomes show:
  - Execution Time: 4.7% average error rate and r² of 0.865
  - Maximum Absolute Error: 10% average error rate and r² of 0.624
- Extend work to additional target architectures and compilers
- Evaluate models with proxy applications CoMD [3] and KRIPTE [4]