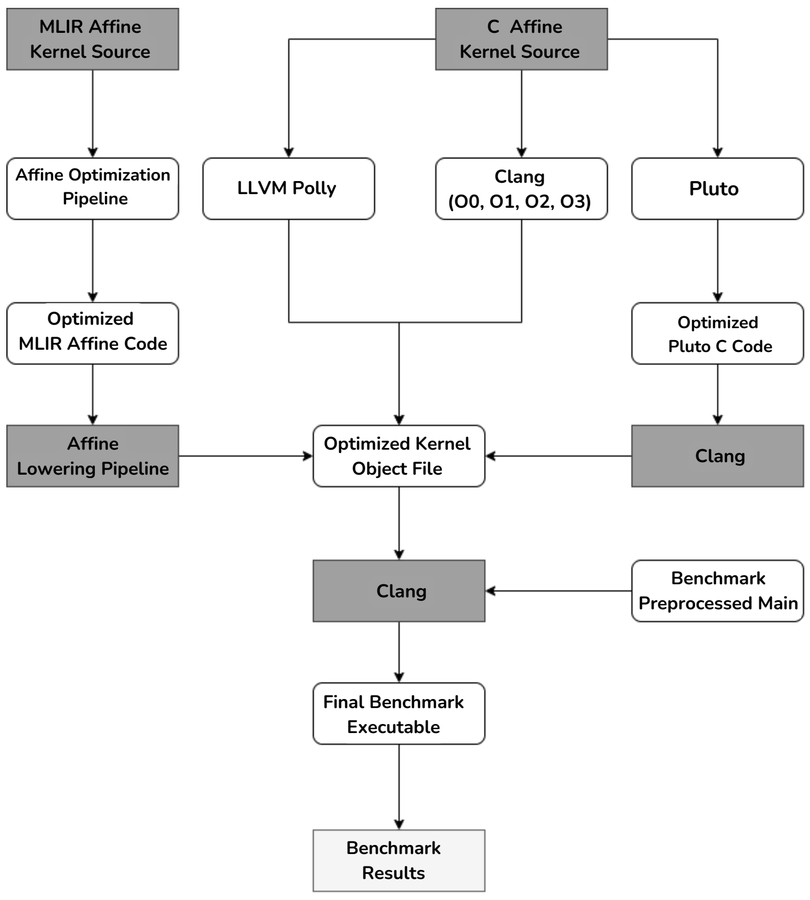
Kiriko: Explorando Compiladores Poliédricos

Um Framework para a Análise Comparativa de Otimizadores Poliédricos

Lucas Victor Costa, Michael Canesche, Fernando Pereira

Kiriko

Kiriko é uma infraestrutura unificada para desenvolver e avaliar otimizações poliédricas sob condições experimentais consistentes. A plataforma integra pipelines automatizados para três otimizadores (**MLIR Affine**, **LLVM Polly** e **Pluto**) e inclui uma implementação completa do PolyBench em MLIR Affine, facilitando experimentos reprodutíveis e análises lado a lado. As contribuições incluem a unificação dos fluxos de teste, a disponibilização do PolyBench-Affine e uma comparação sistemática de desempenho entre Polly, Pluto e MLIR Affine.



Otimizações Poliédricas

Otimizações poliédricas são técnicas que representam loops como objetos geométricos (poliedros) no espaço das iterações. Essa representação permite ao compilador analisar dependências entre iterações e aplicar transformações como **fusion**, **tiling**, **interchange** e **parallelization** de forma automática.

```
void f(int N, float* B, float* A, float*
C) {
    for (int i = 0; i < N; i++) {
        A[i] = B[i] + 1;
    }
    for (int j = 1; j <= N; j++) {
        C[j-1] = A[j-1] * 2;
    }
}
void f(int N, float* B, float* A, float*
C) {
    for (int i = 0; i < N; i++) {
        A[i] = B[i] + 1;
        C[i] = A[i] * 2;
    }
}
```

Exemplo: Loop Fusion! Combina dois loops compatíveis em um único loop

Metodologia

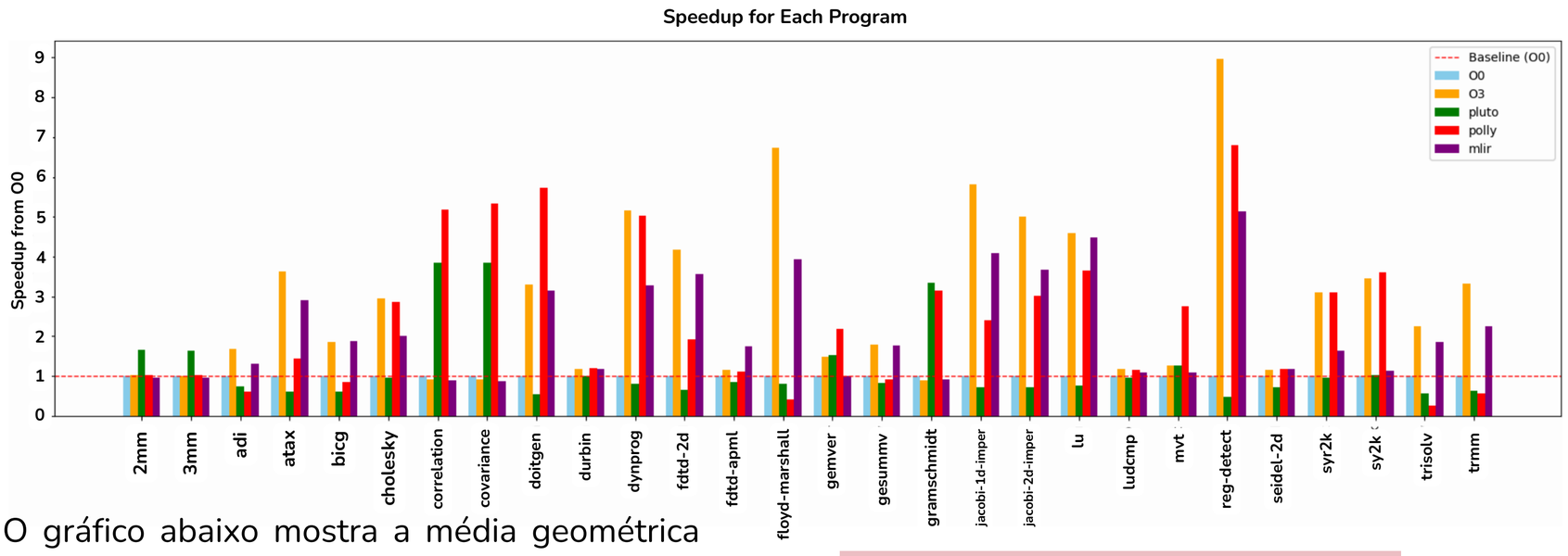
Para construir o Kiriko, foi gerada uma versão completa do PolyBench em MLIR Affine. Os kernels foram extraídos do código C pré-processado, utilizado para evitar problemas com macros e diretivas, e convertidos para MLIR com o Polygeist, com pequenos ajustes manuais. As versões em C e em MLIR partem do mesmo pré-processado, garantindo consistência entre os fluxos de compilação. O Kiriko automatiza a compilação, execução e coleta de métricas, produzindo relatórios detalhados para cada otimizador, incluindo tempos de execução e estatísticas do Linux Perf.

Experimentos

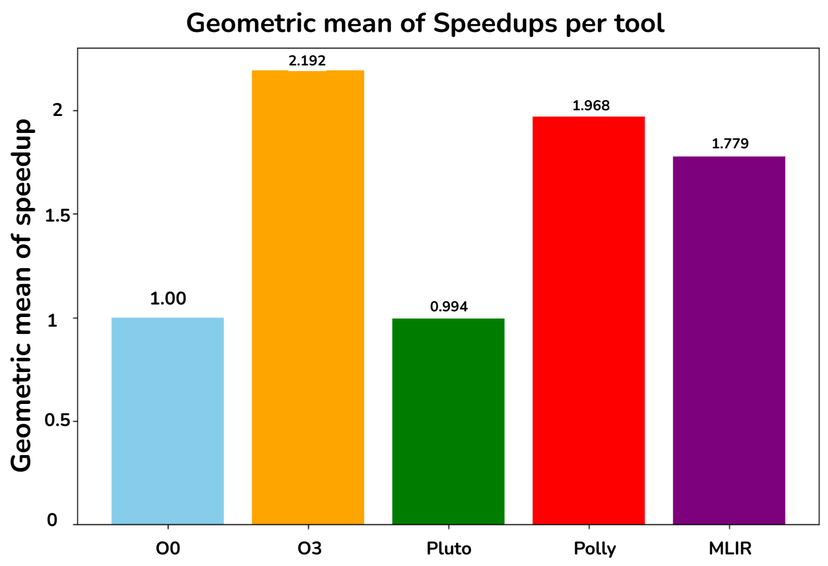
Como prova de conceito, realizamos um experimento inicial avaliando todas as tecnologias de otimização suportadas pelo Kiriko: **Clang** (O0–O3), **LLVM Polly**, **Pluto** e **MLIR Affine**. Cada benchmark foi executado 30 vezes por ferramenta, em uma máquina dual-socket Intel Xeon E5-2680 v2 (20 cores, 40 threads, 32 GB DDR3), usando Ubuntu 20.04. Para cada benchmark, medimos o *tempo médio de execução* e calculamos o speedup relativo ao baseline, permitindo comparar o comportamento das diferentes otimizações e identificar em quais kernels cada ferramenta se destaca ou apresenta limitações.

Resultados

O gráfico mostra o speedup médio de cada ferramenta de otimização em relação ao Clang O0. Cada barra representa o tempo médio de execução de um benchmark (30 rodadas). Valores acima da linha vermelha (>1×) indicam ganho de desempenho.



O gráfico abaixo mostra a média geométrica dos speedups. Clang -O3 apresentou o maior ganho médio (≈2,19×), seguido por Polly (≈1,97×) e MLIR Affine (≈1,78×). Já Pluto mostrou desempenho semelhante ao baseline.



Conclusão

Os resultados mostram que o Kiriko é capaz de integrar e avaliar diferentes tecnologias de otimização de forma sistemática, produzindo métricas consistentes e comparáveis. As análises revelam que Clang O3, Polly e MLIR Affine apresentam ganhos significativos em diversos benchmarks, enquanto Pluto tende a manter desempenho próximo ao baseline. Esses experimentos demonstram o potencial do Kiriko como plataforma unificada para investigação, comparação e desenvolvimento de novas técnicas de otimização.

https://lac-dcc.github.io

DCC DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

UFMG