

UFMG



Transfer Learning Across MLIR Dialects

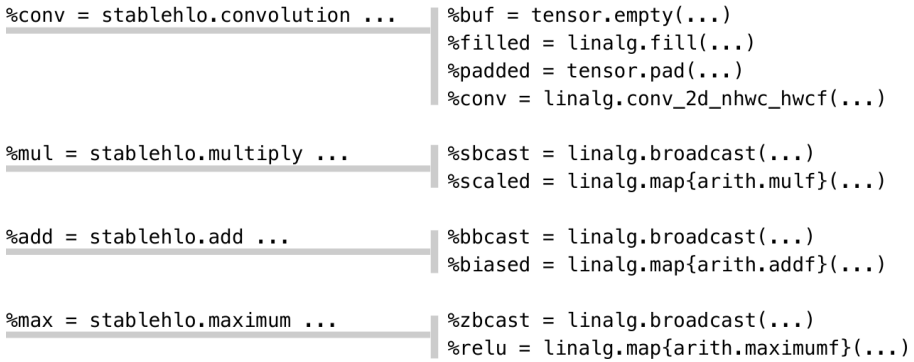
Douglas Viana Coutinho

Orientador: Fernando Magno Quintão Pereira
Coorientador: Vanderson Martins do Rosário

MLIR: uma infraestrutura, muitos dialetos

Multi-Level Intermediate Representation: compiladores são construídos a partir de dialetos, conjuntos modulares de operações e tipos, cada um capturando um domínio ou nível de abstração.

- A mesma computação, várias formas.
- Lowering progressivo.
- Extensível por design.



O mesmo fragmento em dois dialetos: stablehlo (alto nível, tensores inteiros) e linalg (iteração explícita sobre buffers)

O gargalo: dialetos novos não têm dados

- Treinar modelos requer muitos dados rotulados (Wang & O'Boyle 2018).
- Novos dialetos MLIR surgem antes de existirem benchmarks para eles.
- Quando o dialeto evolui, os dados envelhecem: é preciso recoletar e retreinar.

Mesmo GGNN prevendo tempo de treino (τ de Kendall; 1 = ranking perfeito):

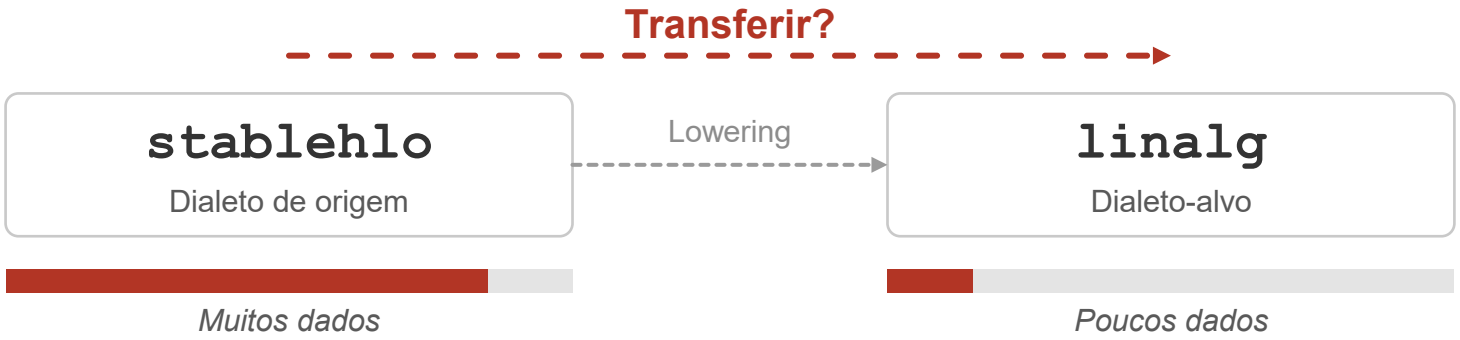
τ ≈ 0,65
com 2,5 mil programas

τ > 0,97
com 228 mil programas

Dialeto novo ou em evolução → Sem benchmarks → Sem modelo confiável

Dá para transferir conhecimento entre dialetos?

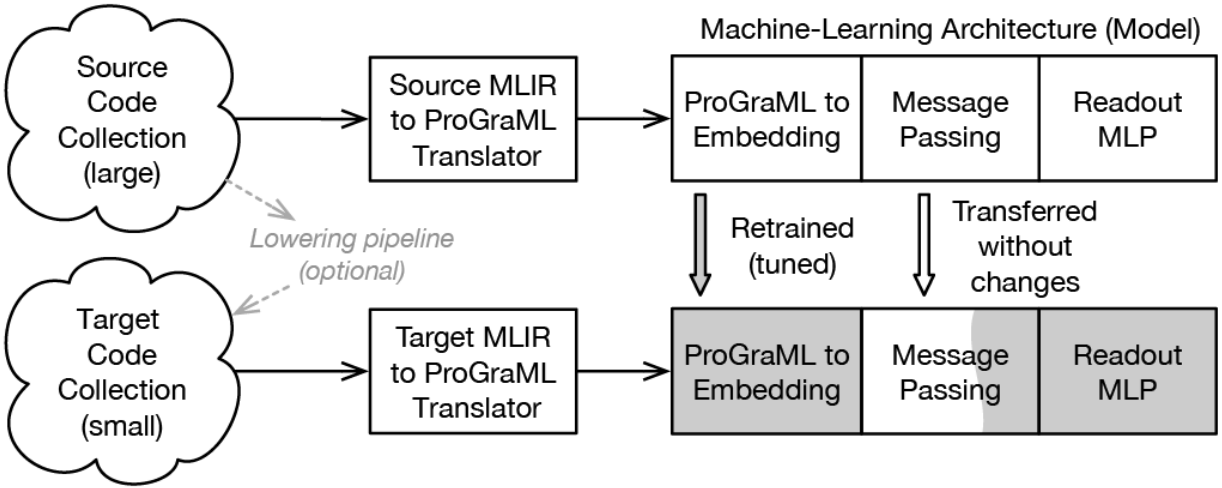
O lowering conecta os dialetos MLIR: o mesmo programa, e sua estrutura, aparece em todos os níveis.



Pré-treinar onde há muitos dados → Adaptar onde há poucos

Pipeline de transferência entre dialetos

Treinar o modelo no dialeto de origem e adaptá-lo ao dialeto-alvo, retreinando só as partes que precisam mudar.



ProGraML no nível do MLIR

DCC

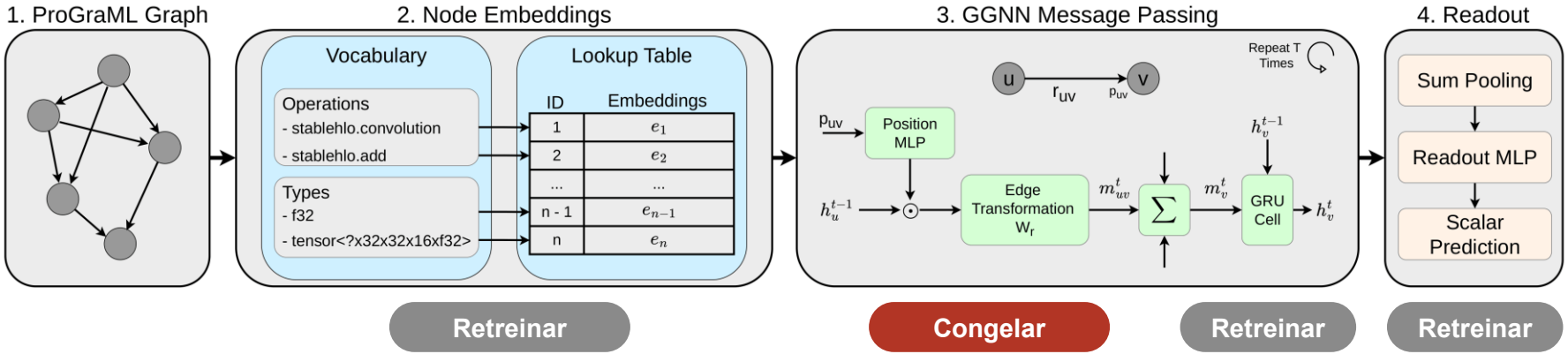
Extraímos o grafo no MLIR, não na LLVM IR, preservando laços e tensores.

Conceito MLIR	Tipo de nó ProGraML
Operação `arith.addf`	Nó de operação
Valor SSA `%res : f32`	Nó de valor
Atributo `stride, padding`	Nó de constante

- Uma travessia recursiva trata as regiões aninhadas do MLIR, então funciona com qualquer dialeto.
- Código aberto: **`github.com/lac-dcc/sphinx`**

GGNN: o que congelar, o que retreinar

A GGNN aprende sobre o grafo do programa: gera embeddings dos nós, propaga informação pelas arestas e produz uma previsão.

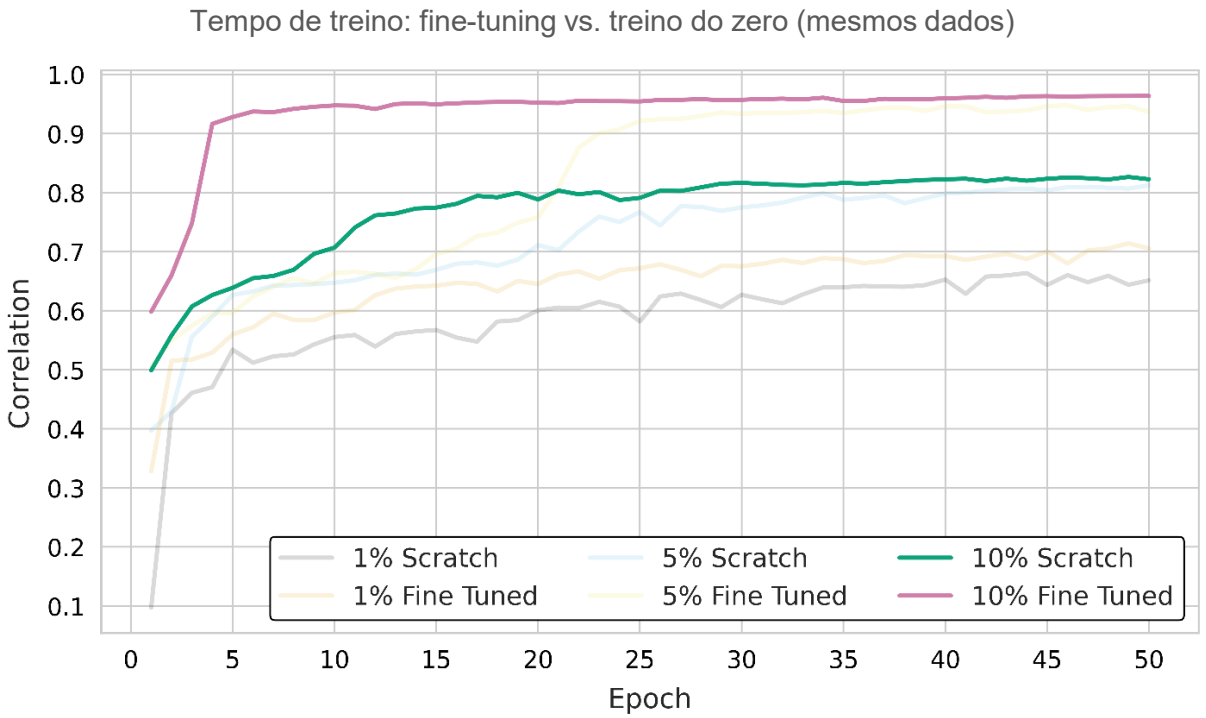


Congelar os MLPs de aresta e posição (padrões relacionais comuns aos dialetos); **retreinar** embeddings, GRU e readout.

Setup experimental

1. Dados: NAS-Bench-101 (423.624 redes) e NAS-Bench-201 (15.625 redes).
2. Tarefas: prever tempo de treino e acurácia; métrica: τ de Kendall.
3. Divisão: 90% no dialeto de origem (stablehlo), 10% no dialeto-alvo (linalg).
4. Modelo: GGNN com ~62 mil parâmetros e 30 iterações de message passing.

Conclusão 1: a transferência funciona



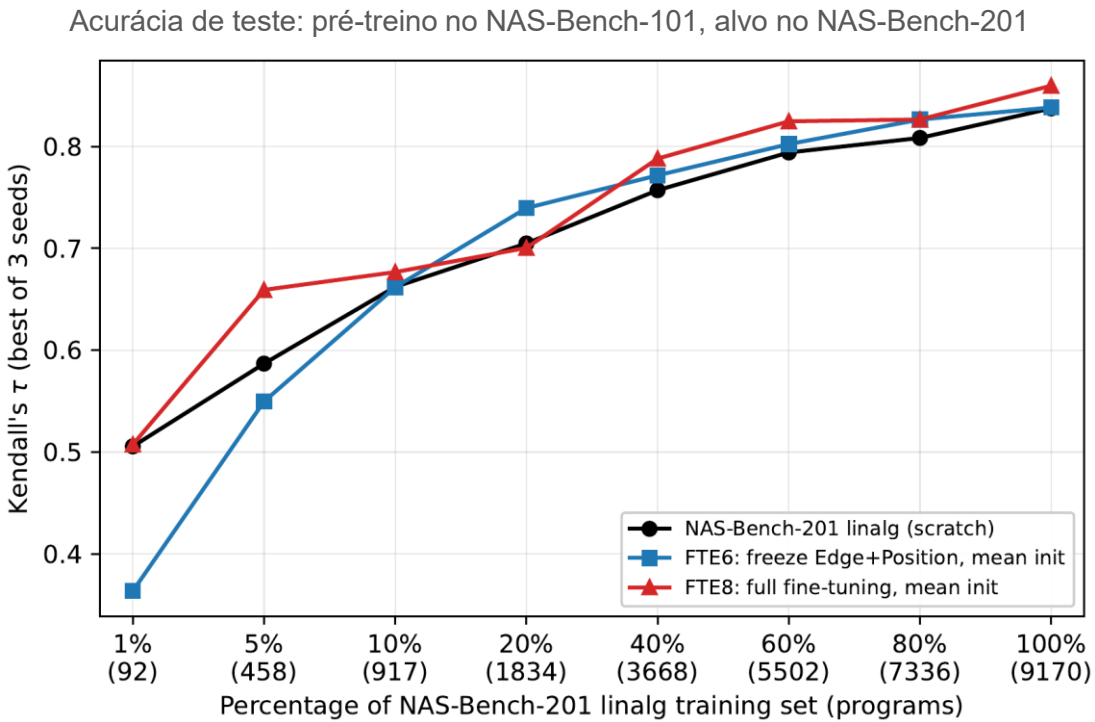
> 50%

menos erro que treinar do zero

48 min

contra mais de 2 h do zero

Conclusão 2: depende da mesma família de aplicações



+0,13

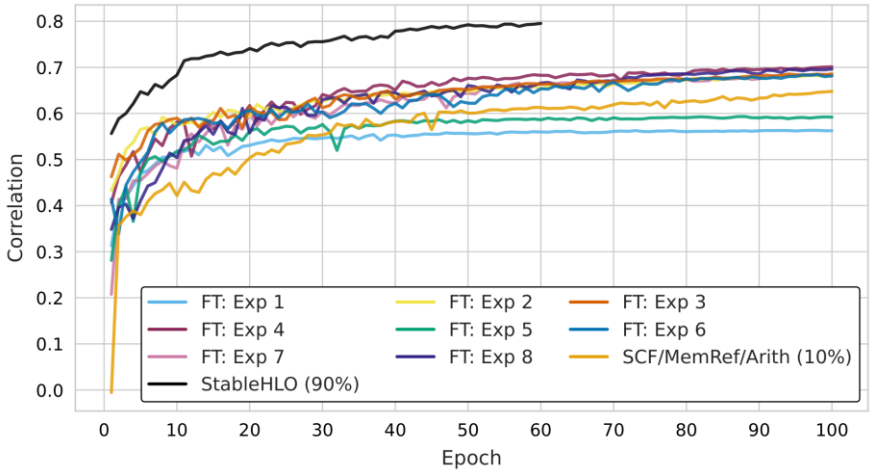
em τ, mesma família de redes

≤ +0,03

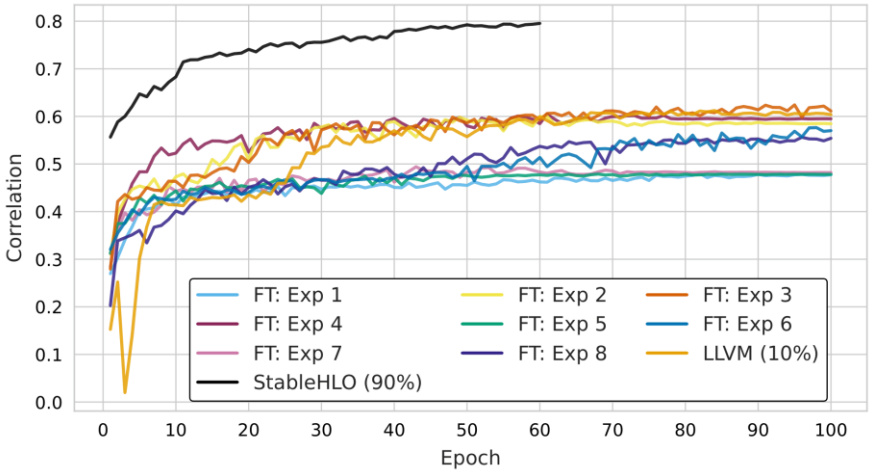
em τ, famílias diferentes

Conclusão 3: depende da distância entre os dialetos

Alvo próximo: scf/memref/arith



Alvo distante: llvm



Fine-tuning ainda ganha do treino do zero

Fine-tuning empata com o treino do zero

Conclusões

DCC

1. A transferência entre dialetos MLIR funciona: supera o treino do zero, com menos dados e menos tempo.
2. Funciona melhor dentro da mesma família de aplicações; ao mudar o universo de programas, o ganho cai.
3. Enfraquece à medida que cresce a distância entre os dialetos de origem e alvo.

Obrigado! · github.com/lac-dcc/sphinx · douglasvc@ufmg.br · cdouglas@cadence.com