

QIMeng-GEMM: 发掘大语言模型自动生成高性能矩阵乘法算子

QIMeng-GEMM: Automatically Generating High-Performance Matrix Multiplication Code by Exploiting Large Language Models

周其睿, 文渊博, 陈睿智, 高科, 熊伟强, 李玲*, 郭崎, 武延军, 陈云霁

Proceedings of the AAAI Conference on Artificial Intelligence. (CCF A)
 2025, 39(21): 22982-22990

主要联系人: 周其睿, zhouqirui22s@ict.ac.cn

研究背景

矩阵乘法在现代科学计算中占据非常重要的地位, 而针对层出不穷的硬件开发高性能的矩阵乘法算子会使代价变得很大。不同硬件厂商会提供官方的高性能计算库, 但由于硬件架构的高速发展, 开发这些库的代价逐渐变得越来越大。而LLMs近期在代码生成领域方面取得的突破性进展给自动生成矩阵乘法代码带来了新的可能。然而LLMs无法生成高性能的矩阵乘法, 原因是现在仍然难以理解复杂的硬件架构, 需要适当的提示词才能使LLMs能够理解硬件特性以在对应平台上生成高性能的矩阵乘法代码。基于此背景和观察, 我们设计了meta-prompt框架以及自动调优算法, 使LLMs在各种硬件架构上生成高性能矩阵乘法代码。

方法: QIMeng-GEMM

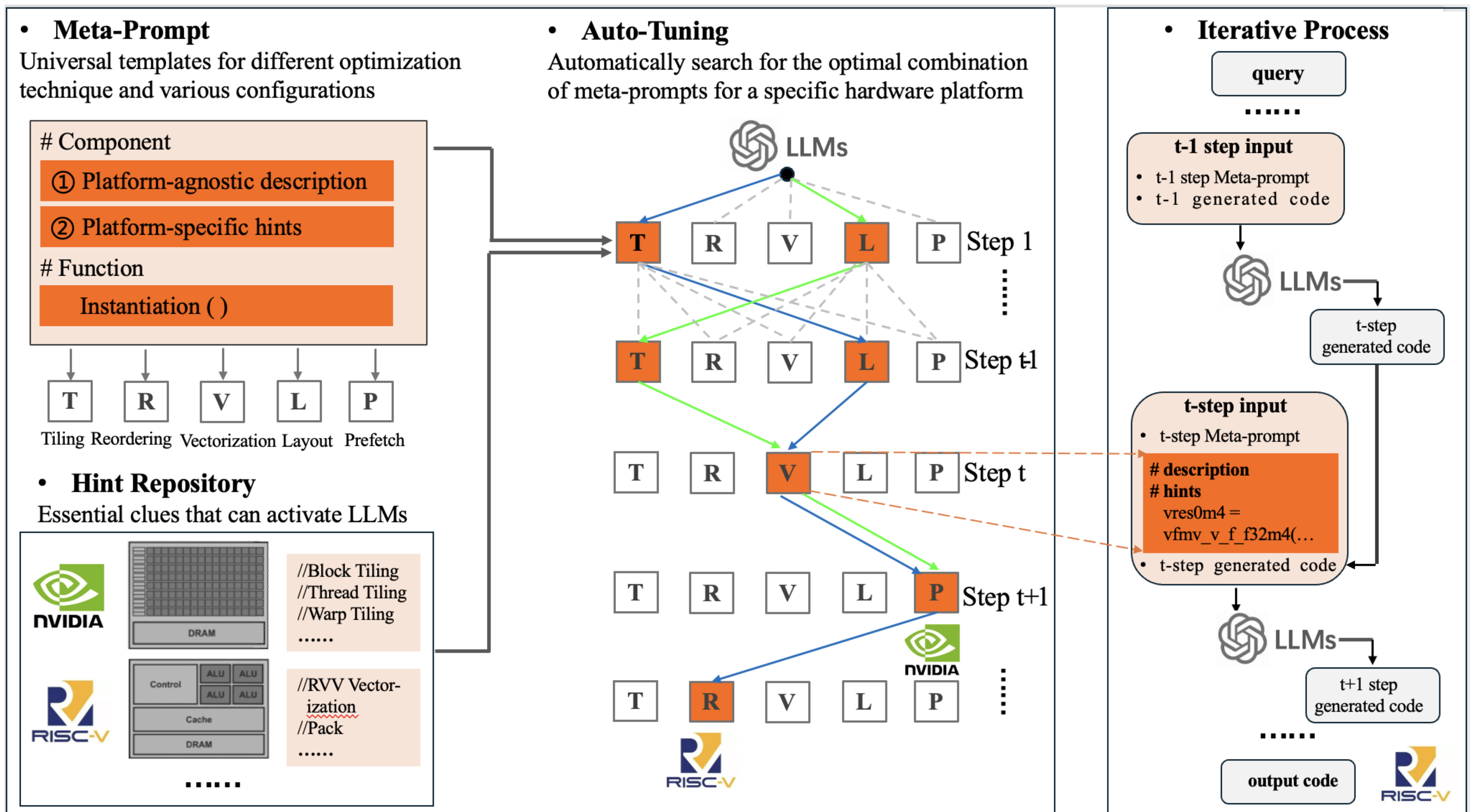


图1 meta-prompt和auto-tuning迭代式代码优化

① Meta-Prompt:

- 硬件无关的优化描述: 利用五种硬件无关的优化方法: 分块、循环重排、向量化、布局优化、流水线, 针对每一个优化预定义好原理和实现的描述;
- 硬件相关的细节提示: 不同硬件设置对应的自然语言描述和核心代码骨架, 来描述特定硬件架构里不同优化的具体实施方法和细节。

② Auto-tuning:

- LLM在meta-prompt基础上搜索不同硬件上的优化原语组合。通过分析当前代码, 从给定的五种优化原语中选取若干个并在特定硬件上实例化, 迭代式的在特定硬件上生成最优的矩阵乘法实现。

实验分析

RISC-V CPU 和NVIDIA GPU两种硬件架构上, 在不同的矩阵维度下测试QIMeng-GEMM的性能。

Hardware	LLMs	256	512	768	1024	1536	2048	4096
C910 (CPU)	GPT-4o (Achiam et al. 2023)	0.27	0.18	0.19	0.14	0.18	0.10	0.09
	+Our QIMeng-GEMM	9.22	9.40	9.93	9.91	9.81	10.08	10.23
		(↑34.15x)	(↑52.22x)	(↑52.26x)	(↑70.79x)	(↑54.50x)	(↑100.80x)	(↑113.67x)
	Claude 3.5 Sonnet (Claude3.5 2024)	2.79	2.62	2.72	2.64	2.75	1.56	0.74
	+Our QIMeng-GEMM	7.42	7.07	6.93	6.83	7.09	6.33	6.35
		(↑2.66x)	(↑2.70x)	(↑2.55x)	(↑2.59x)	(↑2.58x)	(↑4.06x)	(↑8.58x)
	DS-Coder-V2 (Zhu et al. 2024a)	0.76	0.60	0.48	0.61	0.67	0.46	0.10
	+Our QIMeng-GEMM	1.44	1.33	1.30	1.24	1.27	1.17	1.15
		(↑1.89x)	(↑2.22x)	(↑2.71x)	(↑2.03x)	(↑1.90x)	(↑2.54x)	(↑11.50x)
	Codestral-22B (Jiang et al. 2024)	0.16	0.09	0.07	0.05	0.08	0.03	0.02
+Our QIMeng-GEMM	0.86	0.48	0.63	0.28	0.40	0.30	0.17	
	(↑5.38x)	(↑5.33x)	(↑9.00x)	(↑5.60x)	(↑5.00x)	(↑10.00x)	(↑8.50x)	
OpenBLAS 0.3.27 (Zhang et al. 2012)	7.35	5.82	5.49	5.01	4.94	5.11	4.85	
RTX 4070 (GPU)	GPT-4o (Achiam et al. 2023)	1.53	1.68	1.77	1.78	1.65	1.48	1.46
	+QIMeng-GEMM	8.17	10.55	11.47	13.31	14.16	14.28	13.92
		(↑5.34x)	(↑6.28x)	(↑6.48x)	(↑7.48x)	(↑8.58x)	(↑9.65x)	(↑9.53x)
	Claude 3.5 Sonnet (Claude3.5 2024)	1.49	1.59	1.71	1.79	1.61	1.47	1.42
	+Our QIMeng-GEMM	5.13	10.21	10.94	12.10	14.05	14.08	13.11
		(↑3.44x)	(↑6.42x)	(↑6.40x)	(↑6.76x)	(↑8.73x)	(↑9.58x)	(↑9.23x)
	DS-Coder-V2 (Zhu et al. 2024a)	1.42	1.64	1.63	1.71	1.58	1.42	1.38
	+Our QIMeng-GEMM	4.04	8.44	9.14	10.73	12.65	12.48	10.72
		(↑2.85x)	(↑5.15x)	(↑5.61x)	(↑6.27x)	(↑8.01x)	(↑8.79x)	(↑7.77x)
	Codestral-22B (Jiang et al. 2024)	1.35	1.42	1.62	1.68	1.51	1.35	1.27
+Our QIMeng-GEMM	1.55	1.40	1.73	1.82	1.46	1.40	1.30	
	(↑1.15x)	(0.99x)	(1.07x)	(1.08x)	(0.97x)	(1.03x)	(1.02x)	
cuBLAS 12.1 (NVIDIA 2023a)	7.81	9.14	10.79	12.77	12.78	14.25	12.74	

表1 不同硬件架构和矩阵维度下的性能测试

	512	1024	2048
GPT-4o	0.18	0.14	0.10
CoT	0.81	0.63	0.18
	(↑4.50x)	(↑4.50x)	(↑1.80x)
+Our QIMeng-GEMM	9.40	9.91	10.08
	(↑52.22x)	(↑70.79x)	(↑100.80x)
Claude 3.5	2.62	2.64	1.56
+CoT	3.66	2.79	2.47
	(↑1.40x)	(↑1.06x)	(↑1.58x)
+Our QIMeng-GEMM	7.07	6.83	6.33
	(↑2.70x)	(↑2.59x)	(↑4.06x)

表2 QIMeng-GEMM和CoT方法效果对比

	A100 GPU		C910 CPU	
	Time	TFLOPS	Time	GFLOPS
Junior Coder	24 days	11.70	7 days	1.45
Senior Coder	5 days	16.34	3 days	3.08
QIMeng-GEMM	10 mins	15.27	7 mins	9.91

表3 QIMeng-GEMM和人工开发效率对比

1. 在不同的硬件架构和矩阵规模生成高性能矩阵乘法代码
2. QIMeng-GEMM能够远远超过主流LLMs的性能, 最高性能提升**113倍**
3. 相较于CoT方法有较大提升
4. 开发效率相比人工编写最高提升**240倍**