Curator

An Efficient LLM Execution Engine with Optimized Integration of CUDA Libraries

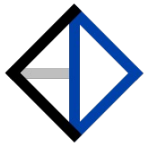
Yoon Noh Lee
Yonsei University
Seoul, Republic of Korea
yoonnohlee@yonsei.ac.kr

Yongseung Yu
Yonsei University
Seoul, Republic of Korea
dydtmd1991@yonsei.ac.kr

Yongjun Park
Yonsei University
Seoul, Republic of Korea
yongjunpark@yonsei.ac.kr

Hwanggeun Yi

2025. 06. 20



CODE Lab
Computing Optimization and
Data-driven Exploration Lab

Contents

I. Backround & Motivation

II. CUrator: An Efficient LLM Execution Engine Using Multiple CUDA Libraries

III. Evaluation

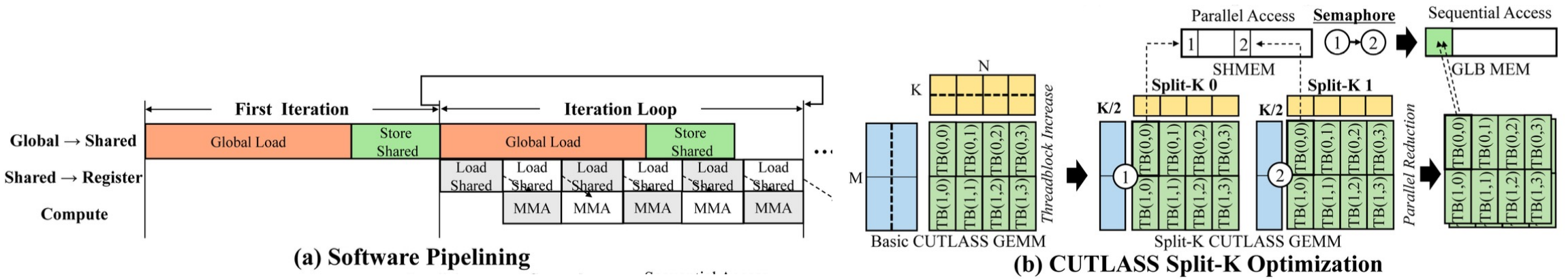
IV. Conclusion

I. Background & Motivation

Background : CUTLASS

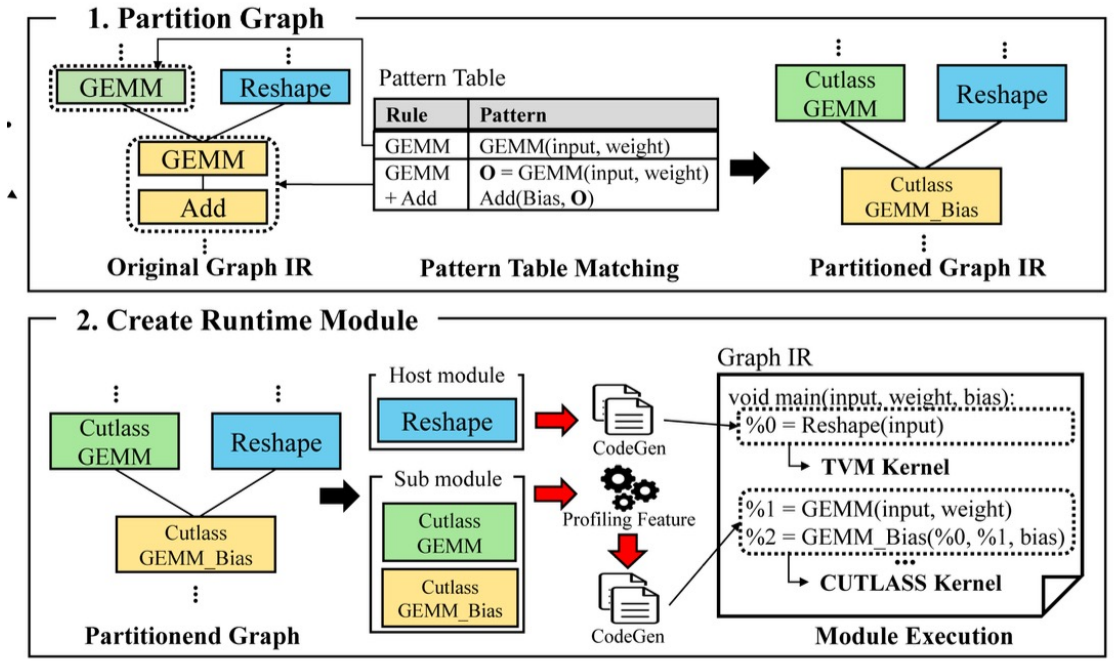
- An open-source template-based library **->** Enable developers to optimize GEMM
- Divide GEMM into **TBTs(Thread Block Tiles),** and each divided by **WTs(Warp Tiles)** <M,N,K>
- **Software Pipelining :** Global -> Shared -> Register -> Compute (Concurrently)
- **Split-K Optimization :** Each Thread Block Tile is divided by K-factor

  -> Allows to utilize of the GPU's computing resources



Background : TVM BYOC

- Backend Infrastructure of TVM (TVM IR -> User Defined IR)
- Allows users to transform the TVM graph IR into a **user-desirable graph IR.**
- TVM graph IR can be converted to third-party libraries, such as **cuBLAS and CUTLASS.**

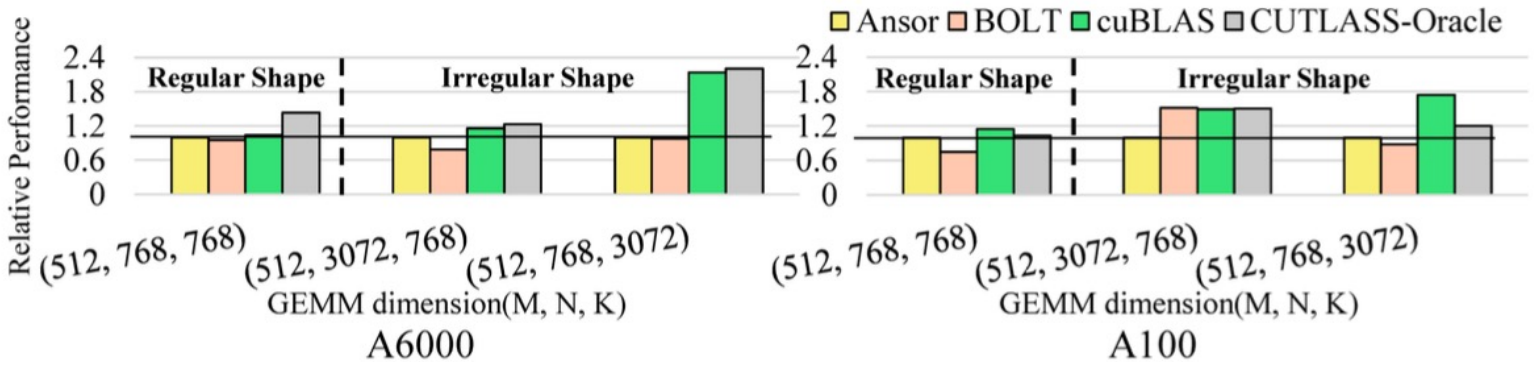


(c) BOLT BYOC

Limitations of Prior Research

- **Ansor**
  - Profiled a subset of the search space and trained the cost model to find the best tiling setting.
  - **Limitation** : Can't consider the hardware resources of the target GPU compared to CUTLASS or cuBLAS
- **BOLT**
  - Framework which utilizes the GEMM and Batch GEMM from the CUTLASS Library.
  - **Limitation** : Has a narrow search space for CUTLASS GEMM tuning.

There is **potential for improving GEMM performance** through tuned **CUTLASS** GEMM kernel or cuBLAS



Limitations of Prior Research

- Lack of Focus on **End-to-End LLM** Execution
- Unnecessary **Runtime Memory Allocation** Overhead
  - Split-K Optimization occurs memory for reduction key during LLM execution
- Insufficient Optimization for Various Operations and Hardware
  - Need to implement efficient GEMM computation across the various GEMM and Hardware

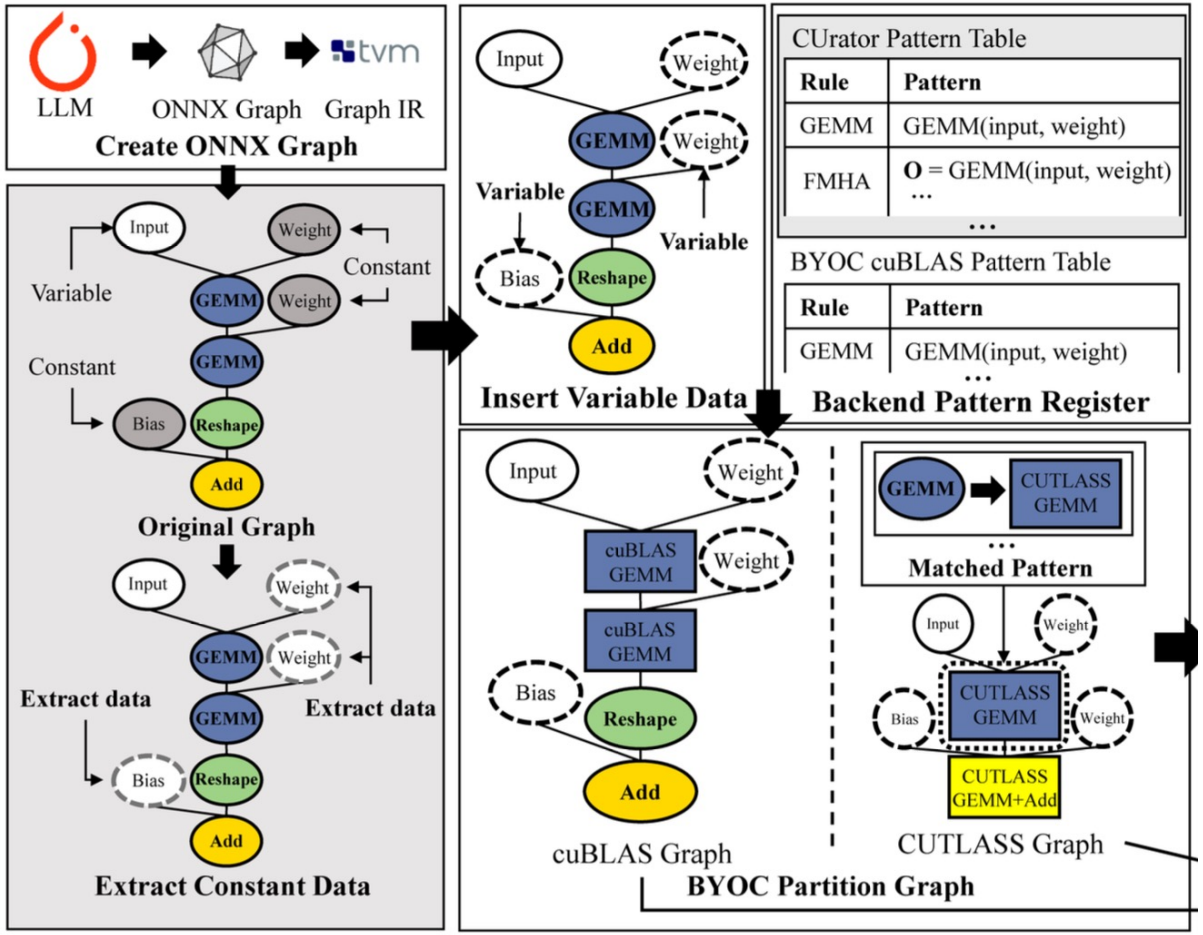
II. CUrator

Overview

- **Graph Rewriter**
  - Transforms ONNX graph into CUTLASS- and cuBLAS-enabled TVM graphs by detecting graph node patterns
- **Extensive search for kernel specific parameters(CUTLASS Tiling)**
  - Lists the available tiling settings and Brute-Force searches for the best tiling parameter
- **Build time reduction key Initialization for CUTLASS Split-k**
- **Split-k support for CUTLASS Batch GEMMs**

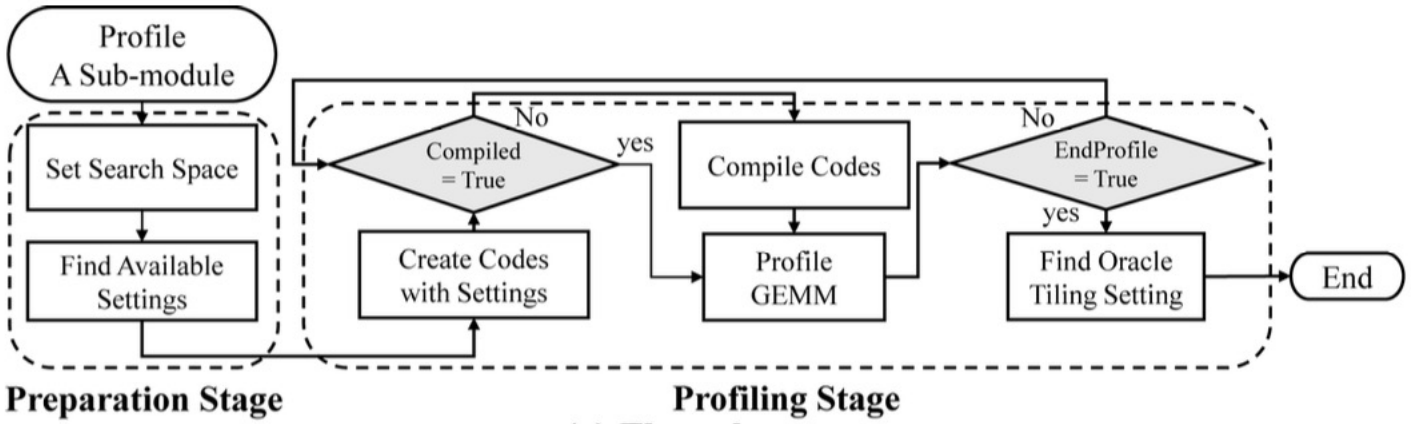
Curator : cuBLAS-Enabled Graph IR

- Extracts all data information if the data format is **constant**
- Constant Data is replaced with the **Variable-Typed Data**
- CUrator generates the modified graph IR according to BYOC table



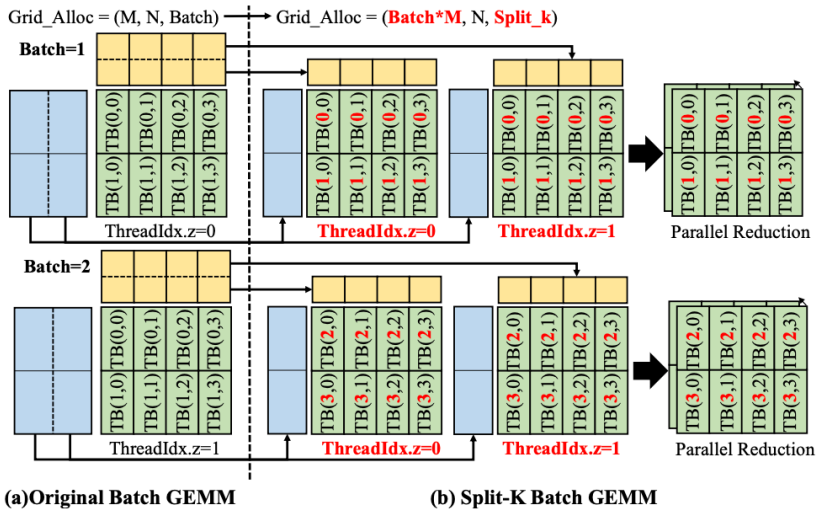
Curator : Best CUTLASS Tiling Setting Exploration Support

- **Merge Several graph IR nodes into a fused GEMM**
  - Operation Funsion of GEMM operation and epilogue functions
  - Each epilogue function is an element-wise, and execution time is smaller than GEMM
- **Find the best tiling setting for the target fused GEMM (Consider only single GEMM)**
  - $Preparation\ Stage$: Verify the available tiling setting by considering detailed checklist
  - $Profiling\ Stage$: The Available parameters are compiled as an object file and record the performance



Curator : Split-K Support on Batch GEMM

- LLM computations results in underutilization of GPU cores due to insufficient GEMM size.
  - For this reason, CUrator exploits a batch GEMM integrated with the split-K mechanism
- Original Batch GEMM use $blockIdx.z$ index as the batch index
  - It makes it difficult to use the $blockIdx.z$ index as the semaphore key
  - By changing the threadblock organization, the $blockIdx.z$ can be used as as the semaphore key



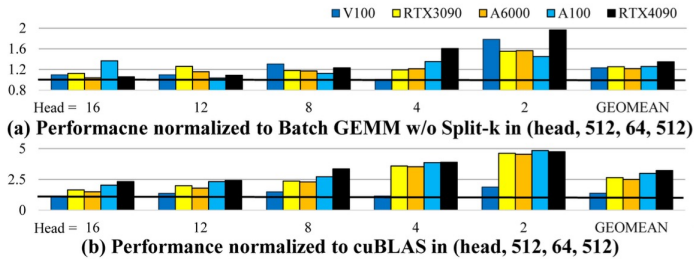
III. Evaluation

Evaluation : Single Precision

- **TensorRT-LLM**
  - CUrator w/FMHA achieves an average speedup of up to 1.18X, 1.40X
  - Slightly lower performance than TensorRT in certain cases because CUrator doesn’t perform target-GPU-specific
- **Ansor**
  - Ansor can perform better than other frameworks when the model size, batch size are small
  - However, cannot outperform when the size of GEMMs increase because Ansor does not understand GPU
- **BOLT**
  - Worse than CUTLASS-Oracle because of narrow search space
  - BOLT’s Graph IR does not consider the performance overhead of reduction key

Evaluation : Techniques of CUrator

- **Split-K Optimization**
  - The inference with the split-K outperforms without the split-K by **1.09×** and **1.08×** for batch sizes of 1 and 4
  - According to the above result, a significant number of GEMMs should be tuned with the split-K optimization
- **CUTLASS-Friendly Graph IR Modification**
  - The CUTLASS-friendly Graph IR outperforms the original Graph IR in all cases
  - Split-K algorithm is often required on modern GPUs consisting of many SMs



- **Split-K Support on Batch GEMMs**
  - As the number of cores in GPUs increases, the Batch GEMM w/ split-K becomes more effective
  - Batch GEMM with split-K is effective when the input size is small, resulting in improved performance over cuBLAS

IV. Conclusion

Conclusion

1. Generates CUTLASS-/cuBLAS-friendly graph IRs on various LLMs on the TVM framework
2. Perform comprehensive search for tuning parameters in CUTLASS library
3. Various Optimization Technique
   - build-time reduction key initialization support for CUTLASS Split-K GEMMs
   - Split-K support for CUTLASS Batch GEMMs
   - Finally CUrator selects mapping path between cuBLAS and CUTLASS
4. 1.50x, 4,99x for representative LLMs on the A100 GPU in single and half precision

CUrator can provide the best direction for next-generation tuning frameworks