moTuner: A Compiler-based Auto-tuning Approach for Mixed-precision Operators

Zewei Mo, Zejia Lin, Xianwei Zhang, Yutong Lu

mozw5@mail2.sysu.edu.cn

Computing Frontiers 2022

May 17th-19th, 2022, Torino, Italy

中山大學 SUN YAT-SEN UNIVERSITY

ARCSYSU Refines Computing SYStem Uses

西北工業大學 NORTHWESTERN POLYTECHNICAL UNIVERSITY

Agenda

- Background
  - Mixed-precision
  - Operator
  - Compilation
- Motivation
- Design
  - Overview
  - Data Dependency Analysis
  - Setting Tuning
- Evaluation
- Summary

Mixed-precision

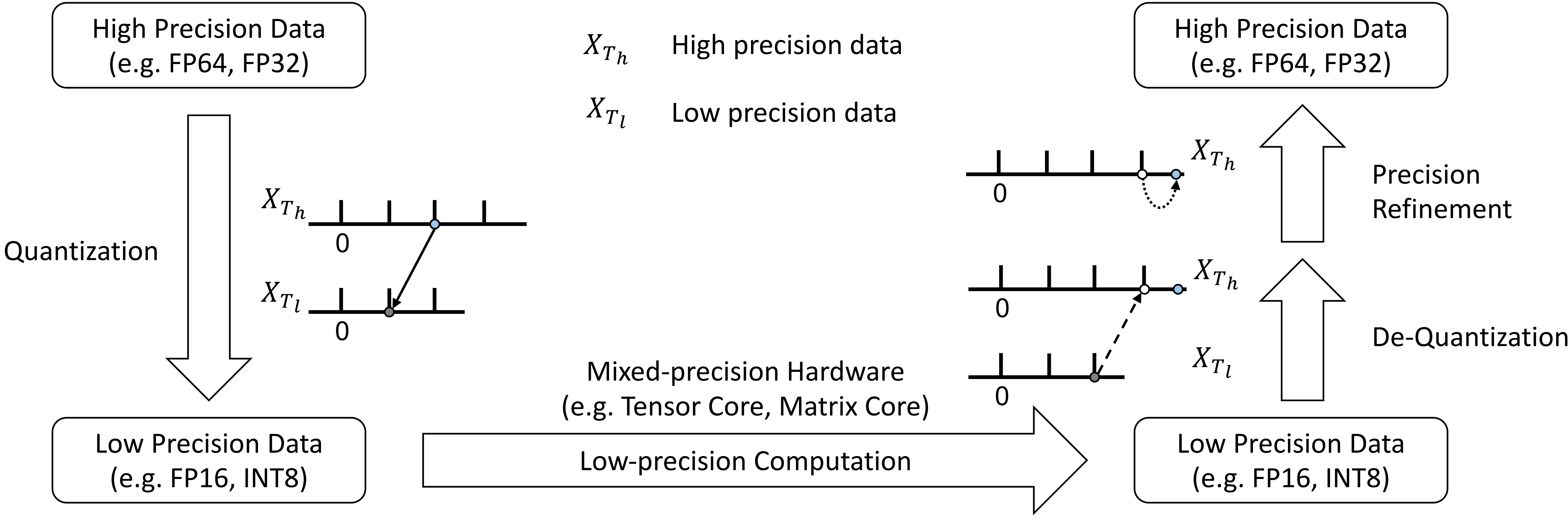
- A computation with multiple precisions
  - Different precision of input and output in a computation

- Different precision presentations
  - Various sizes and unit precisions
  - Specific computation hardware
  - Software support

Precision Presentation	Bit Number	Minimum Value	Maximum Value	Unit Precision
INT8	8	0	1.27×10^{2}	1.0×10^{0}
BF16	16	1.2×10^{-38}	3.4×10^{38}	3.9×10^{-3}
FP16	16	6.1×10^{-5}	6.6×10^{4}	4.9×10^{-3}
FP32	32	1.2×10^{-38}	3.4×10^{38}	6.0×10^{-8}
FP64	64	2.2×10^{-308}	1.8×10^{308}	1.1×10^{-16}

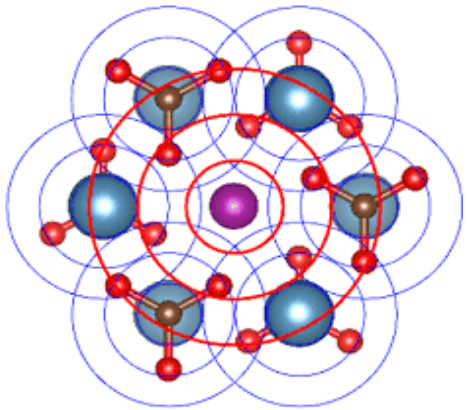
Mixed-precision (cont.)

- Various error in different domain-specific applications



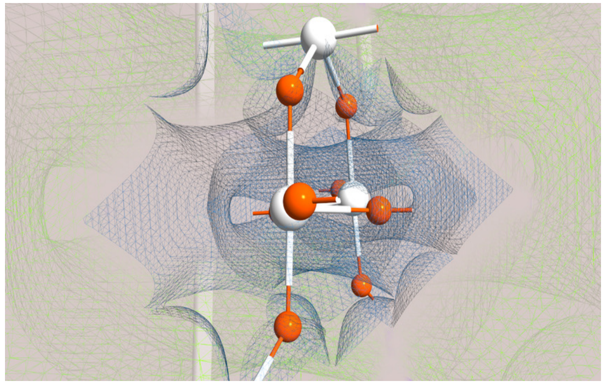
Mixed-precision (cont.)

- A lot of applications take advantages of mixed-precision



Ab initio Molecular Dynamics

Extending the limit of molecular dynamics with ab initio accuracy to 10 billion atoms



Density Functional Theory

Dynamic Precision for Electron Repulsion Integral Evaluation on Graphical Processing Units (GPUs)

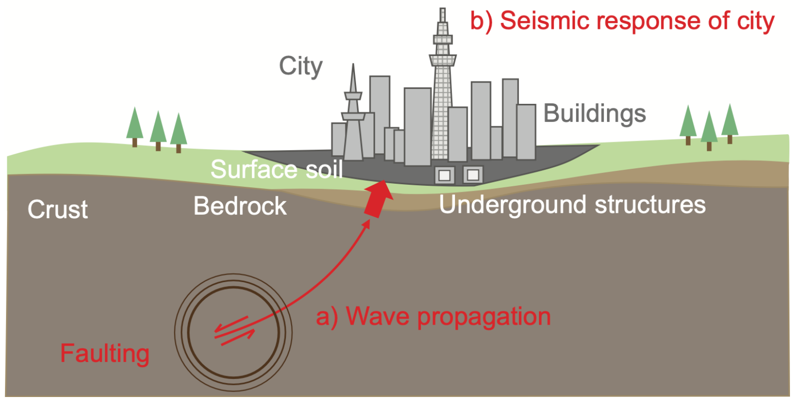
Lower Triangular

$$\begin{bmatrix} A00 & A01 & A02 \\ A10 & A11 & A12 \\ A20 & A21 & A22 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ L10 & 1 & 0 \\ L20 & L21 & 1 \end{bmatrix} \begin{bmatrix} U00 & U01 & U02 \\ 0 & U11 & U12 \\ 0 & 0 & U22 \end{bmatrix}$$

Upper Triangular

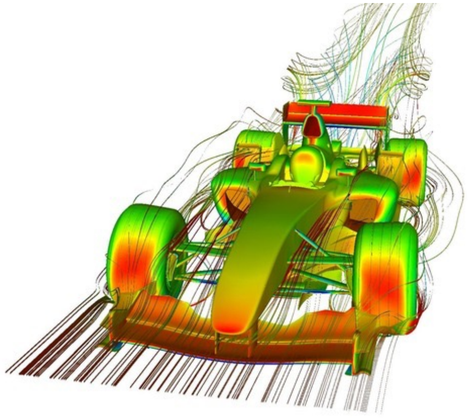
HPL-AI

Harnessing GPU Tensor Cores for Fast FP16 Arithmetic to Speed up Mixed-Precision Iterative Refinement Solvers



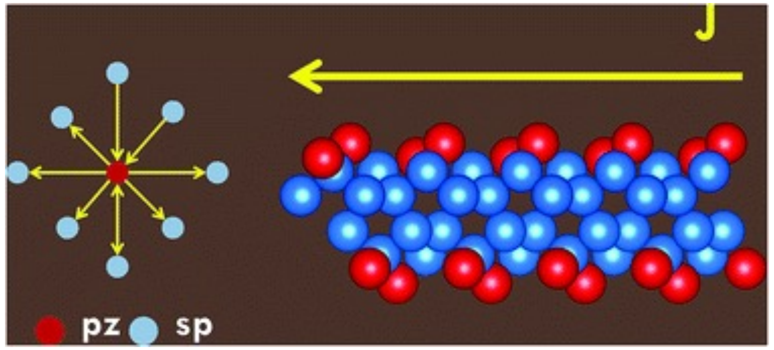
Earthquake Prediction

A Fast Scalable Implicit Solver for Nonlinear Time-Evolution Earthquake City Problem on Low-Ordered Unstructured Finite Elements with Artificial Intelligence and Transprecision Computing



Computational Fluid Dynamic

A Mixed Precision Multicolor Point-Implicit Solver for Unstructured Grids on GPUs

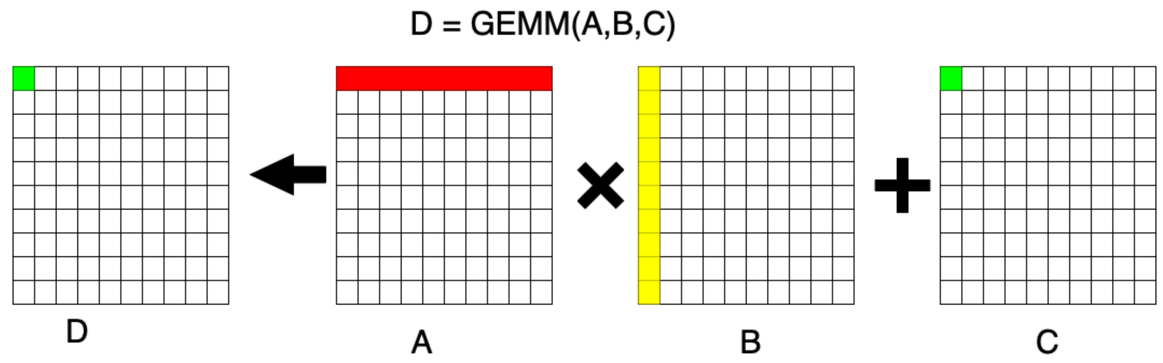


Quantum Transport

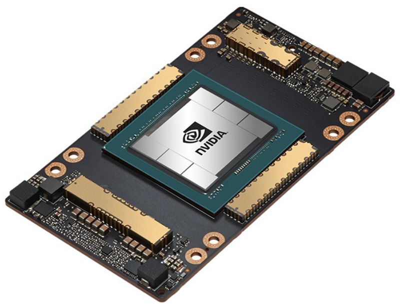
A Data-Centric Approach to Extreme-Scale Ab initio Dissipative Quantum Transport Simulations

Mixed-precision Operator

- Operator: A function accompolishes specific computation
  - General Matrix-Multiply (GEMM)
    - Most widely used in HPC and deep learning applications.



- Mixed-precision support for operators
  - Hardware: GPU, CPU, TPU, NPU, …
  - Software: cuBLAS, rocBLAS, MKL, …

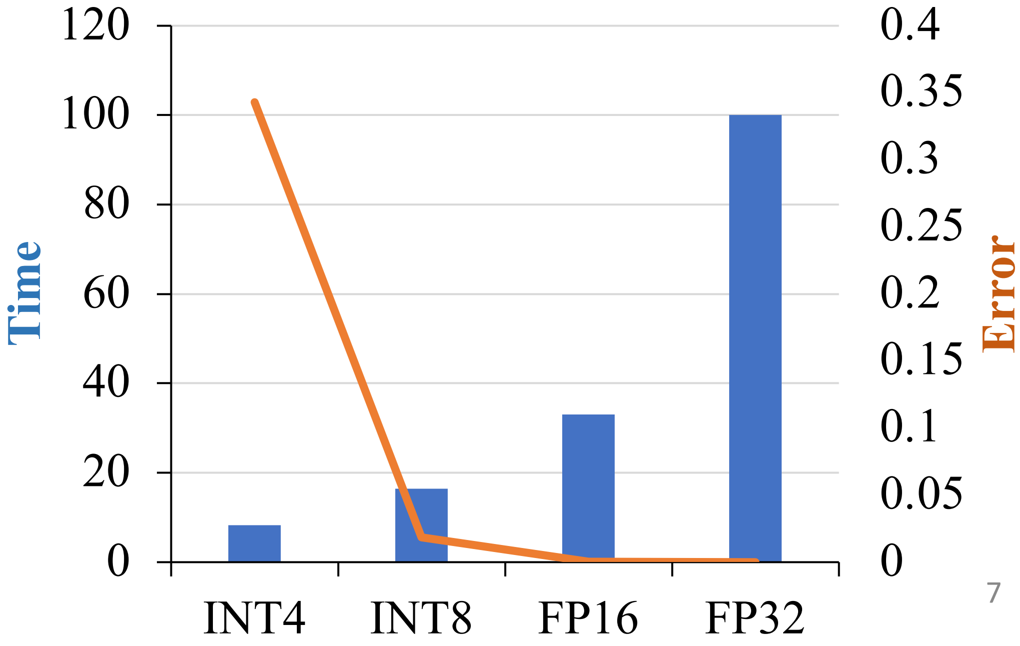


Mixed-precision Operator (cont.)

TC: Tensor Core
MC: Matrix Core

Hardware	FP64	FP32	FP16	BFloat16	INT8
A100	9.7 TFLOPS TC: 19.5 TFLOPS	19.5 TFLOPS TC: 156 TFLOPS	78 TFLOPS TC: 312 TFLOPS	TC: 312 TFLOPS	TC: 624 TFLOPS
MI100	11.5 TFLOPS	23.1 TFLOPS MC: 46.1 TFLOPS	MC: 184.6 TFLOPS	MC: 92.3 TFLOPS	MC: 184.6 TOPS
Intel Xeon Platinum 8180	-	3.57 TFLOPS	-	-	5.18 TOPS
TPU v3	-	-	-	90 TFLOPS	-

- Mixed-precision setting of GEMM operator
  - Different input precision and output precision
  - Tradeoff between performance and accuracy

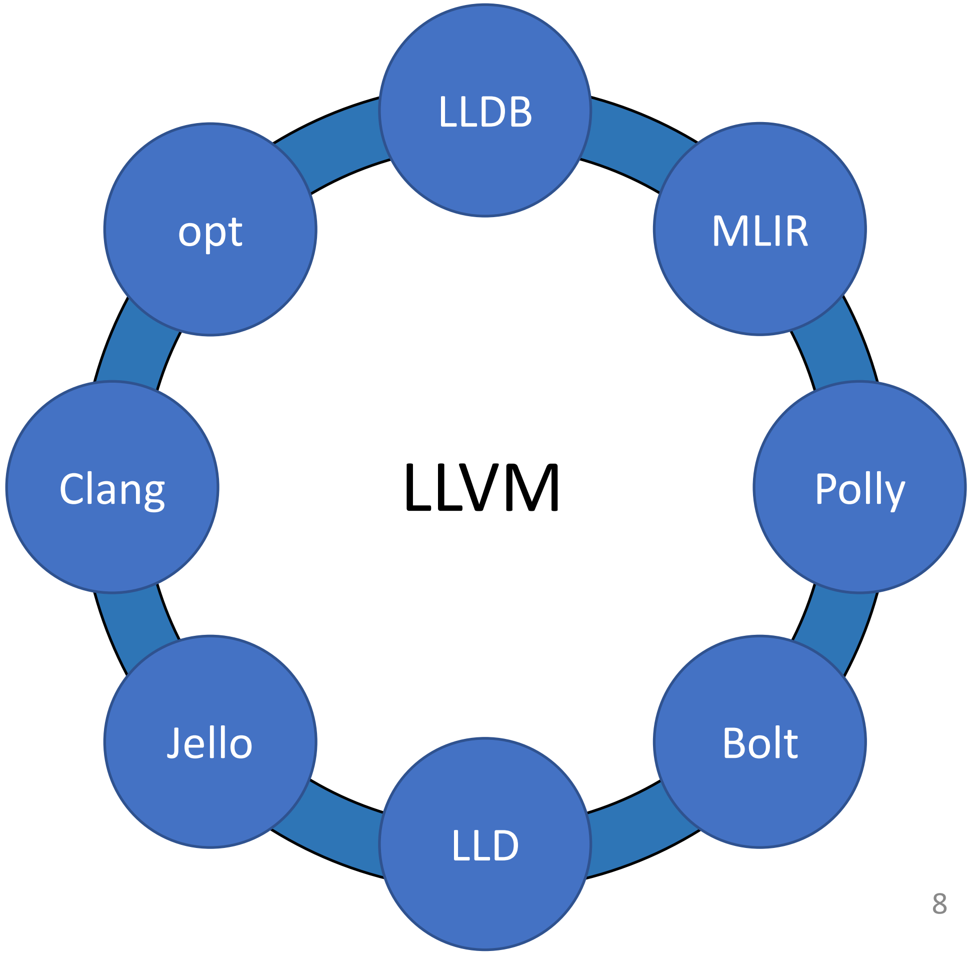


Compilation

- LLVM: A compiler framework consists of multiple tools
  - FrontEnd
  - IR generator and optimizer
  - Binary generator and optimizer
  - Just-in-Time Optimizer
  - …

- Brings operator library into play

- Provides all static information about program



Compilation (cont.)

- IR (Intermidiate Representation): Key of Optimization
  - Not platform-related
  - Function call (e.g. operator) remains in its API form
  - SSA (Static Single Assignment)
    - Name of each assignment is unique

V = 4	SSA Transform $\Rightarrow$	V1 = 4
Z = V + 5		Z1 = V1 + 5
V = 6		V2 = 6
Y= V + 7		Y1= V2 + 7

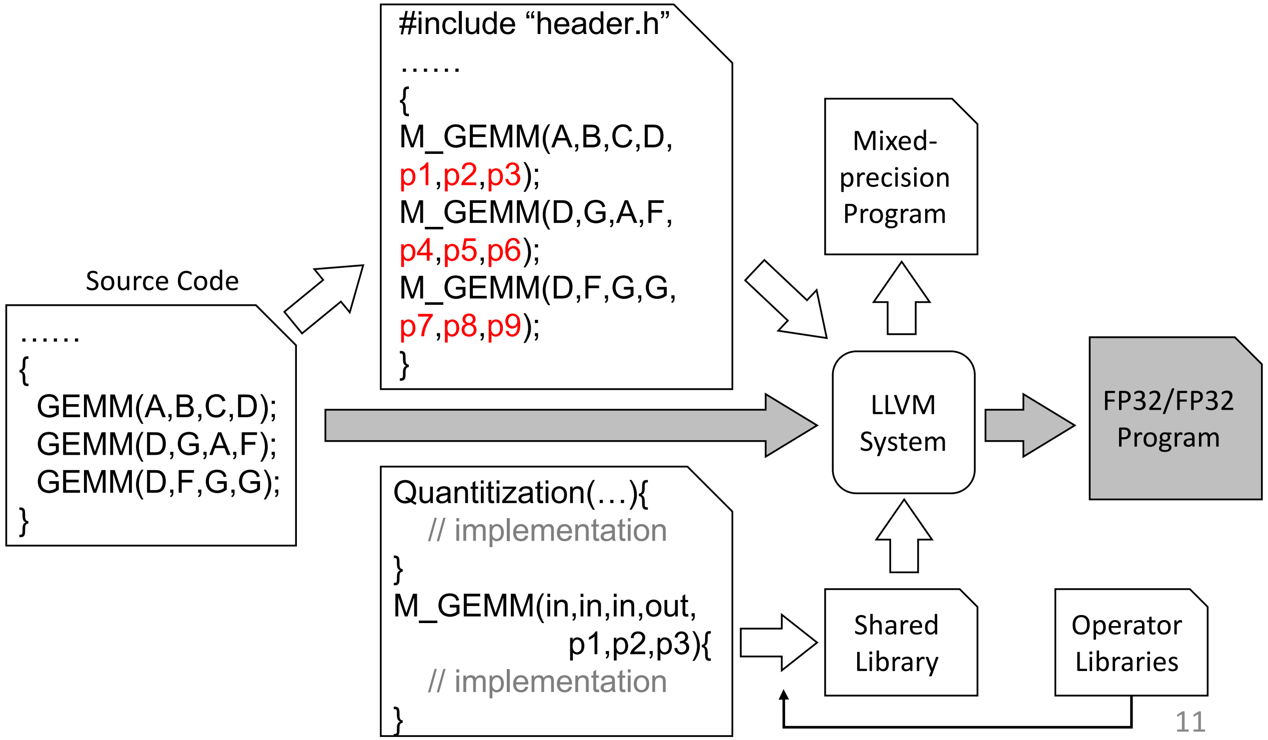
- Optimization Pass
  - Data Dependency Analysis
  - Code Transform
  - …

Agenda

- Background
  - Mixed-precision
  - Operator
  - Compilation
- Motivation
- Design
  - Overview
  - Data Dependency Analysis
  - Setting Tuning
- Evaluation
- Summary

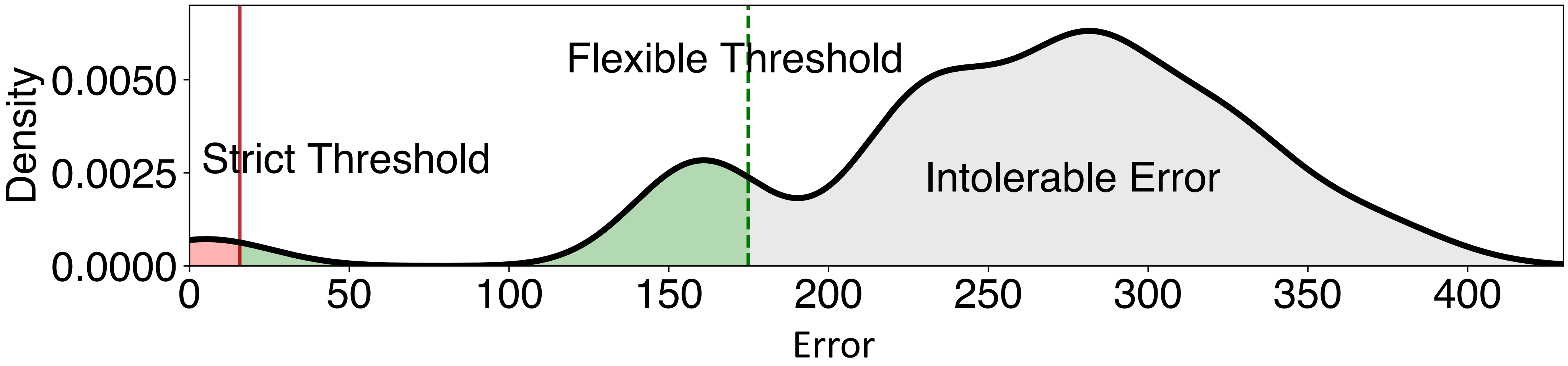
Motivation

- Much burden comes with using mixed-precision operator
  - Customized mixed-precision operator library: five component of mixed-precision computation
  - Modifying source code: replace target operators with mixed-precision ones
  - Setting mixed-precision setting parameter
- Huge setting space for N operators
  - Considering 4 settings for each operators
  - Total 4^N settings



Motivation (cont.)

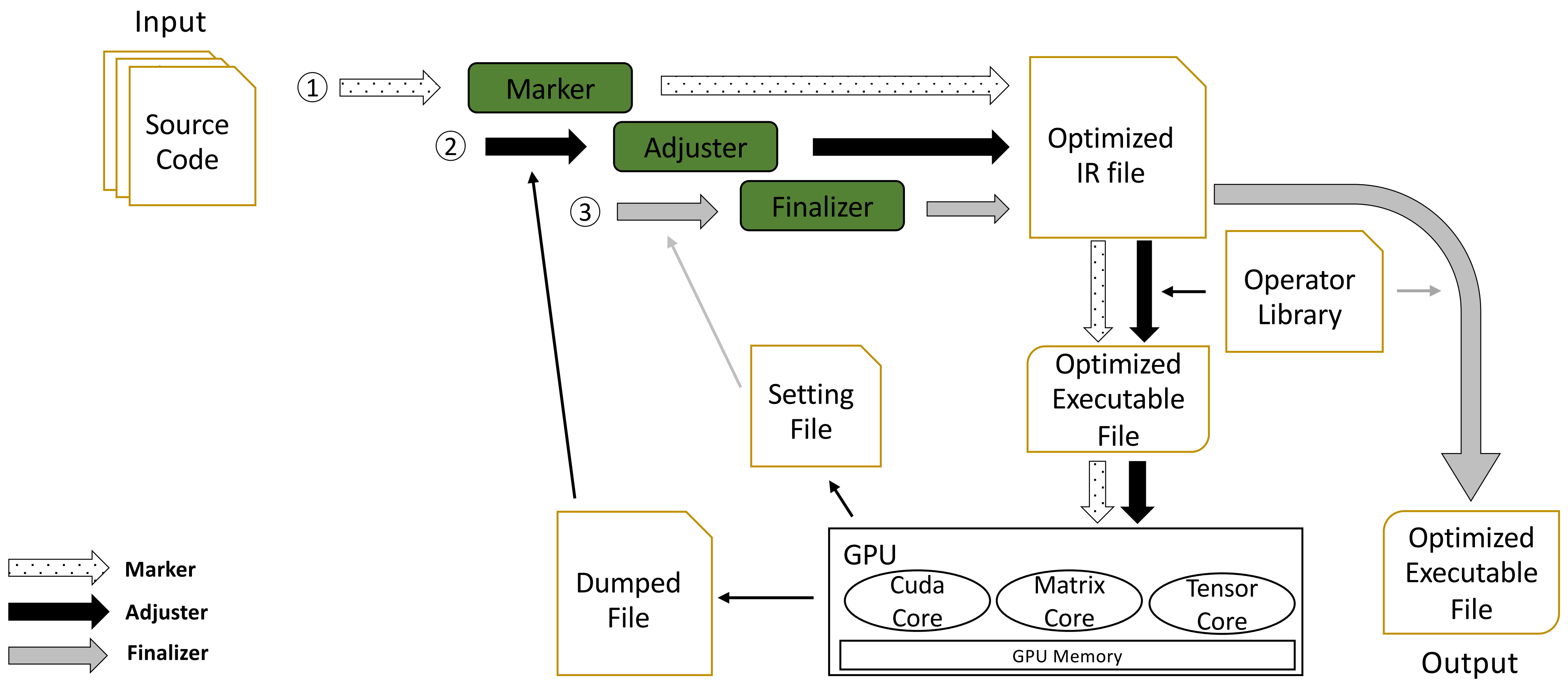
- Different applications demand different accuracy
  - An efficient tuning tool is required for different scenarios
  - Considering about the following scenario:
    - A output of GEMM is the output of one application
  - Density means the occurrence frequency of different error value



Agenda

- Background
  - Mixed-precision
  - Operator
  - Compilation
- Motivation
- Design
  - Overview
  - Data Dependency Analysis
  - Setting Tuning
- Evaluation
- Summary

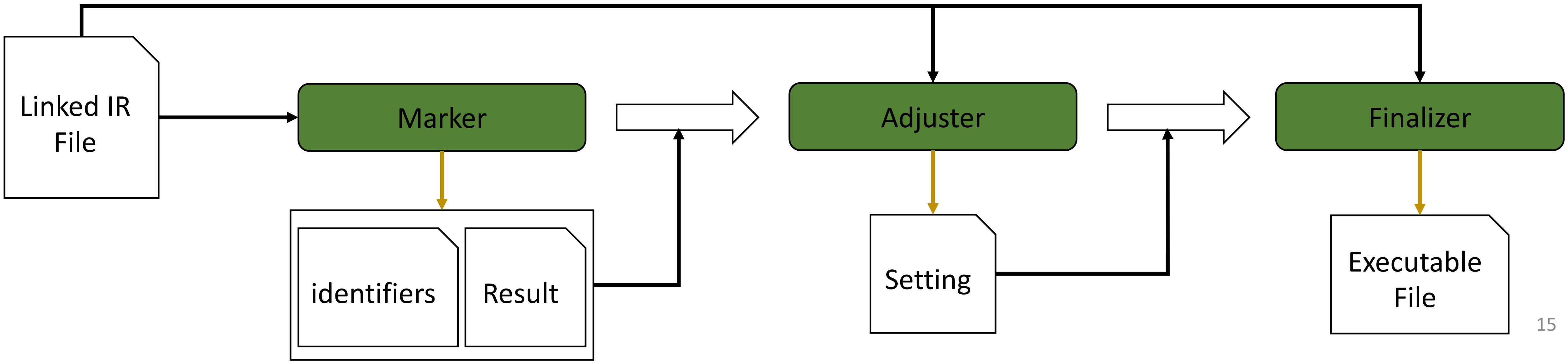
Overview



Input
Source Code
①
Marker
②
Adjuster
③
Finalizer
Optimized IR file
Operator Library
Optimized Executable File
Setting File
Dumped File
GPU
Cuda Core
Matrix Core
Tensor Core
GPU Memory
Optimized Executable File
Output
Marker
Adjuster
Finalizer

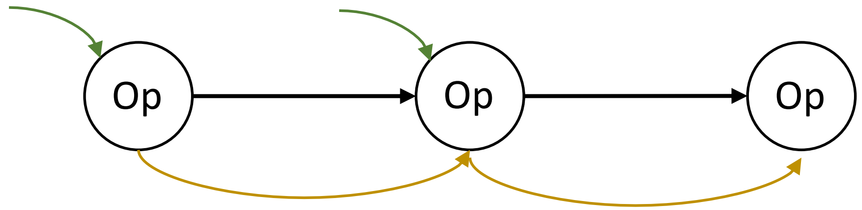
Overview (cont.)

- Marker
  - Unique ID for each operator: order number of first execution and executed count
  - Original result and performance of operators
- Adjuster
  - Analyzes related operators
  - Optimized efficient search strategy
  - Qualified settings without performance downgrade

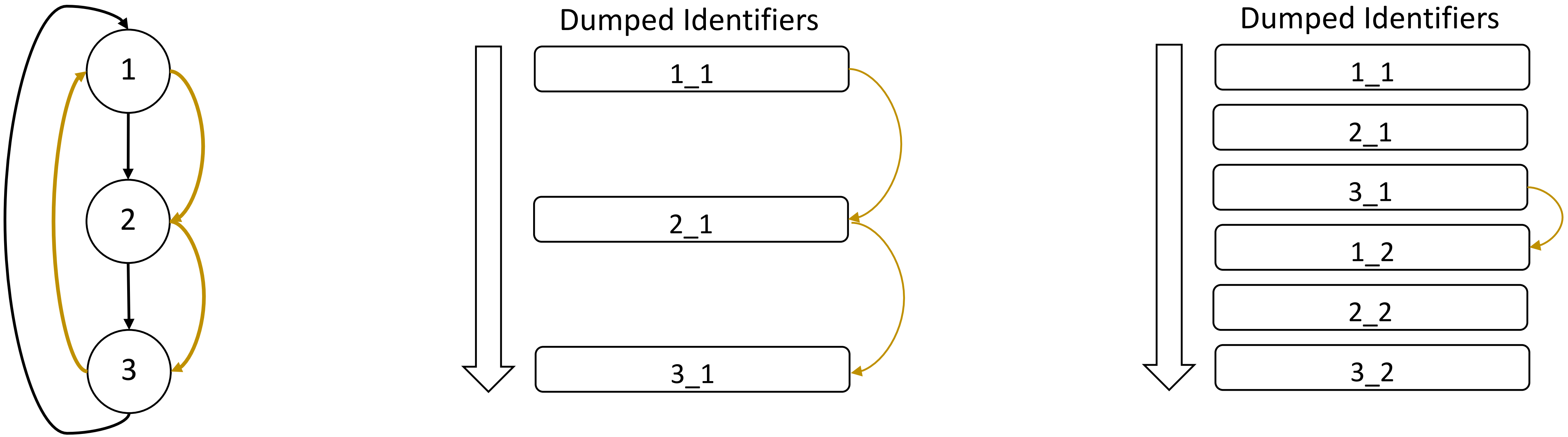


Data Dependency Analysis

- Data dependency between operators
  - One input variable of an operator is the output of another operator



- Related operators: have data dependency and execute in given input
  - Only setting of related operators should be fixed



Levelized Setting

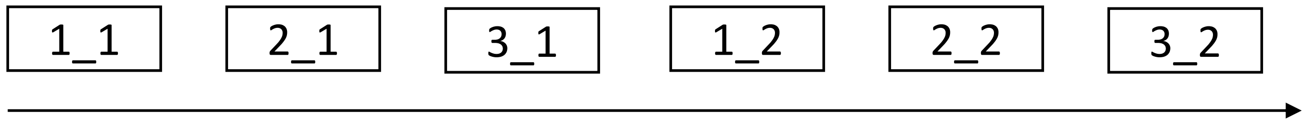
- Settings with different levels
  - High performance comes with low accuracy
  - Reduces search space

#	Input Precision	Output Precision	Performance Rank	Accuracy Rank	Level
1	FP32	FP32	3rd	1st	3
2	FP16	FP32	2nd	2nd	2
3	INT8	FP32	1st	3rd	1

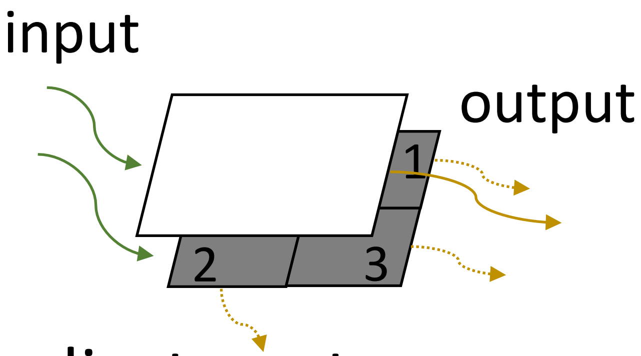
- Fix of setting
  - Detects error of tuned operators with different settings
  - Fixes settings of related operators when an operator introduces intolerable error
  - Each time of fix needs a level up of settings

Tuning Process

- An ordered executed operator list



- Output and performance of operators from “shadow execution"



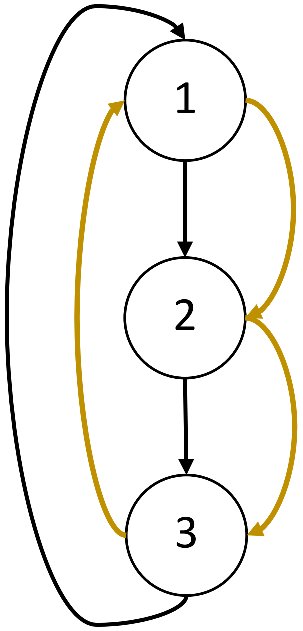
- One operator tuned in each adjustment

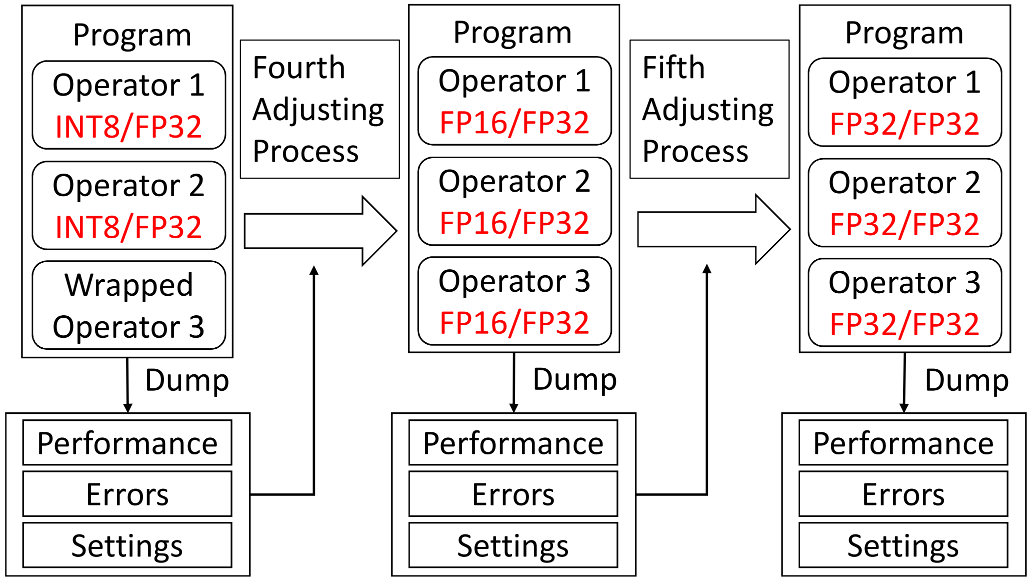
- Extra tuning process to guarantee accuracy

Tuning Process (cont.)

- For each operator, tries all settings and choose the best one
- Checks error and performance of tuned operators in each adjusting process
- Updates settings of related operator when one introduces intolerable error

Input Precision	Output Precision	Level
FP32	FP32	3
FP16	FP16	2
INT8	FP32	1





Agenda

- Background
  - Mixed-precision
  - Operator
  - Compilation
- Motivation
- Design
  - Data Dependency Analysis
  - Setting Tuning
- Evaluation
- Summary

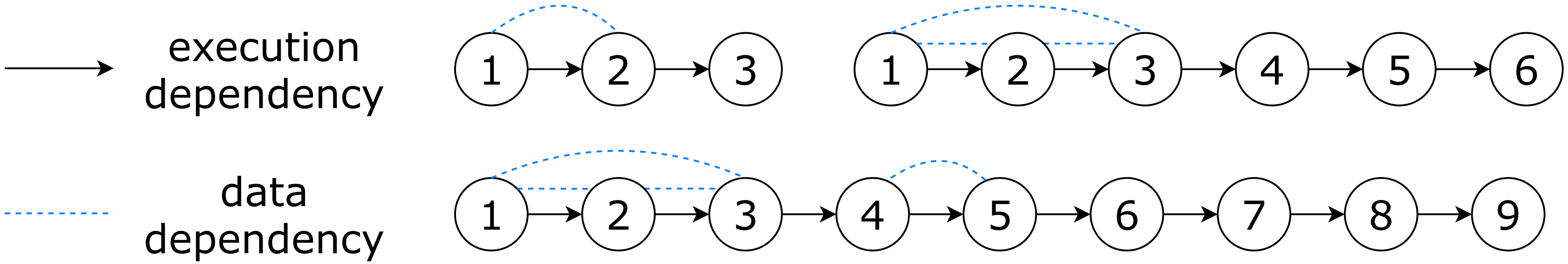
Environment

- Programs are written in HIP
- Programs are compiled with –O3

Hardware		Software	
CPU	EPYC 7302 Freq.: 3.0-3.3 GHz	Operating System	CentOS 7.9
CPU Memory	256 GB	Operator Library	hipBLAS@4.3.1 rocSolver@4.3.1
GPU	MI100 FP32 Perf.: 46.1 TFLOPS FP16 Perf.: 184.6 TFLOPS	Compiler	HIPCC@4.3.1
GPU Memory	32 GB	Backend	HIP@4.3.1

Benchmarks

- Micro-benchmark (Micro)



- Cholesky Factorization (CF)
  - Tiled version
  - Input setting denoted by (N, t)

- HPL-AI
  - Tiled version
  - Input setting denoted by (N, t)
  - Validates whether the scaled residual of result matrix is smaller than 16

Schemes

- Baseline: Runs the program in FP32/FP32 precision
- Exhaust: Exhaustively finds the qualified one with highest performance
- PriorK: Is aware of error and performance of each setting combination
  - Micro: randomly selects the fastest 1% qualified settings
  - CF and HPL-AI: randomly selects fastest 50% qualified settings
- moTuner: Uses moTuner to get the optimized executable file

Metrics

- Performance
  - Average of five execution time of whole program (GEMM part for HPL-AI)
- Accuracy
  - Mean related error (MRE): E_γ

$$E_\gamma(X, X') = \| X_{flatten} - X'_{flatten} \|_\infty$$

  - Maximum absolute error (MAE): E_δ

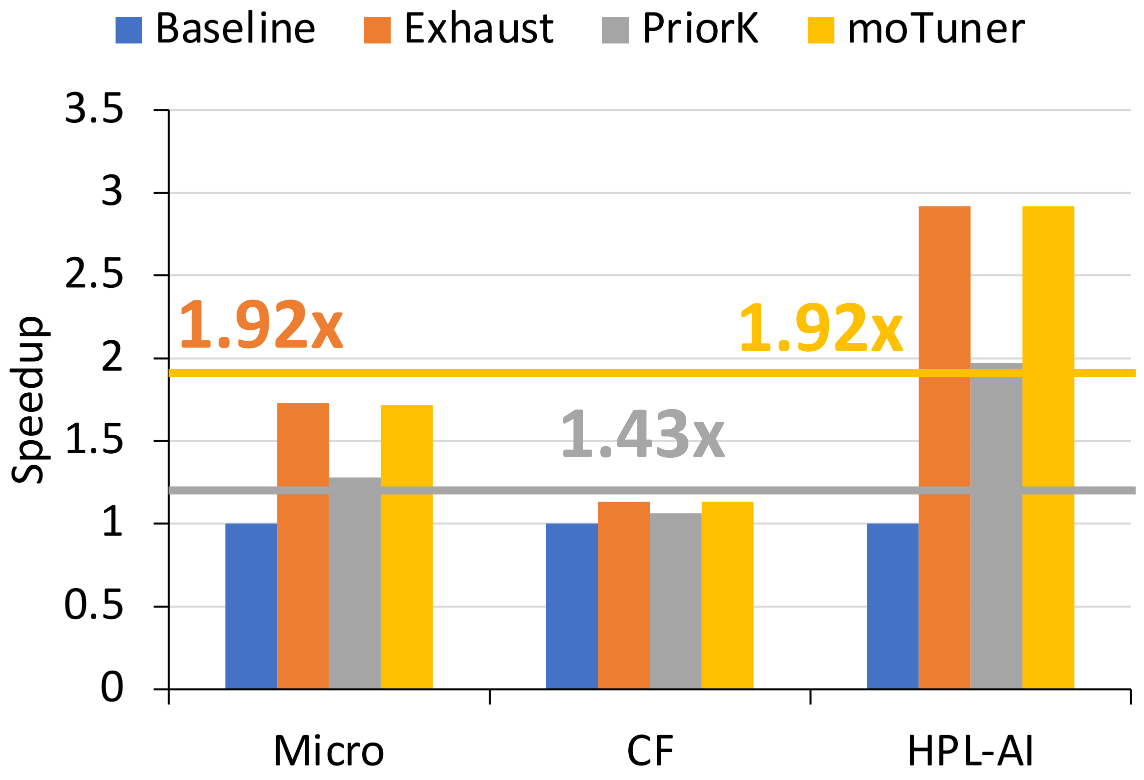
$$E_\delta(X, X') = \|X - X'\|_F / \|X'\|_F$$

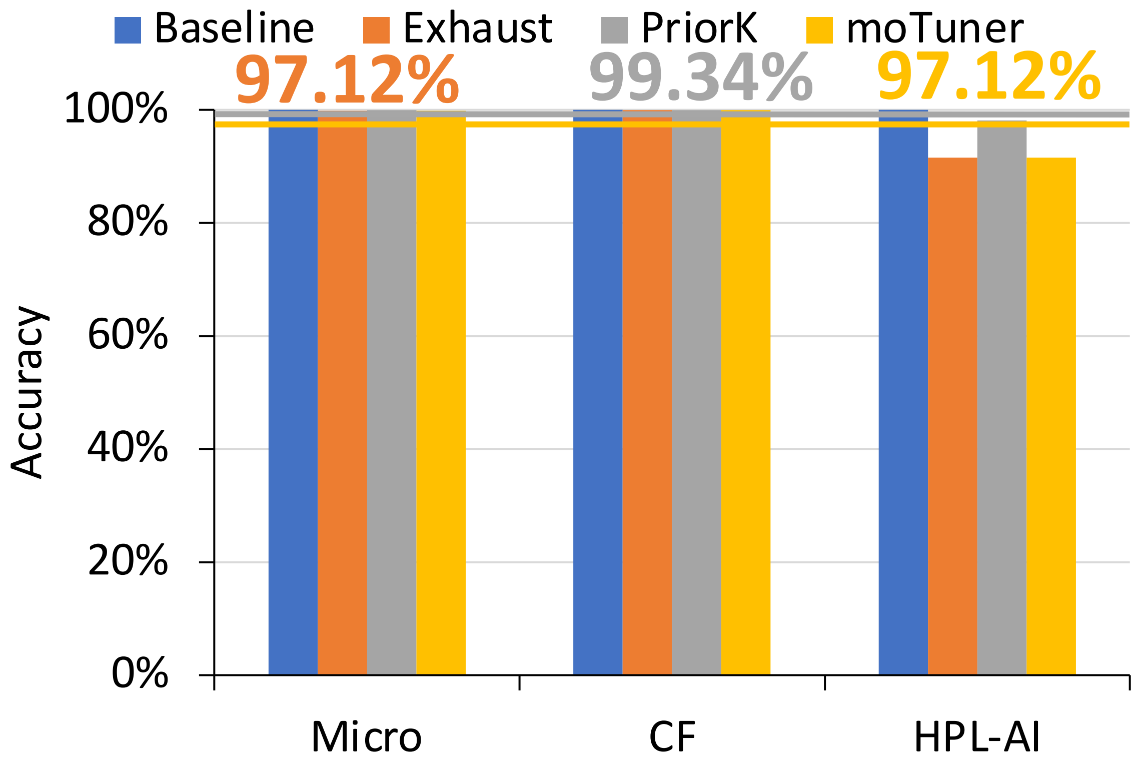
Error Kind	Value	Error Threshold Category
E_γ	0.05	E1
E_γ	0.005	E2
E_γ	0.0005	E3
E_γ	0.00005	E4
E_δ	100	E5
E_δ	10	E6
E_δ	1	E7
E_δ	0.1	E8

- Automation Efficiency
  - Execution count is an objective metric to provide insight of tuning effort

Result & Analysis

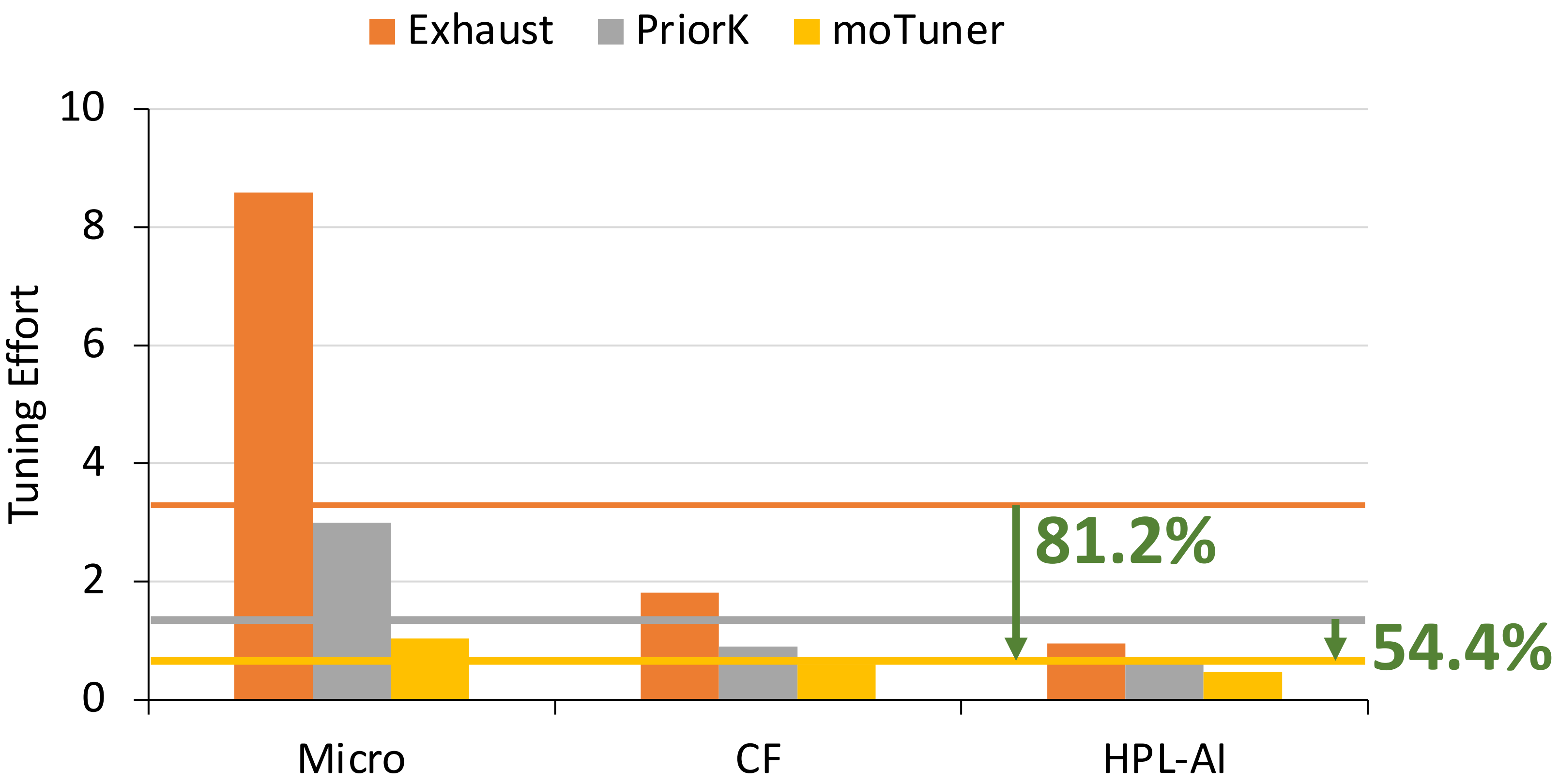
- Performance and Accuracy
  - moTuner gains **1.92x** performance improvement and **97.12%** accuracy in average
  - moTuner gains **32.26%** higher performance than PriorK, only with **2.23%** lower accuracy





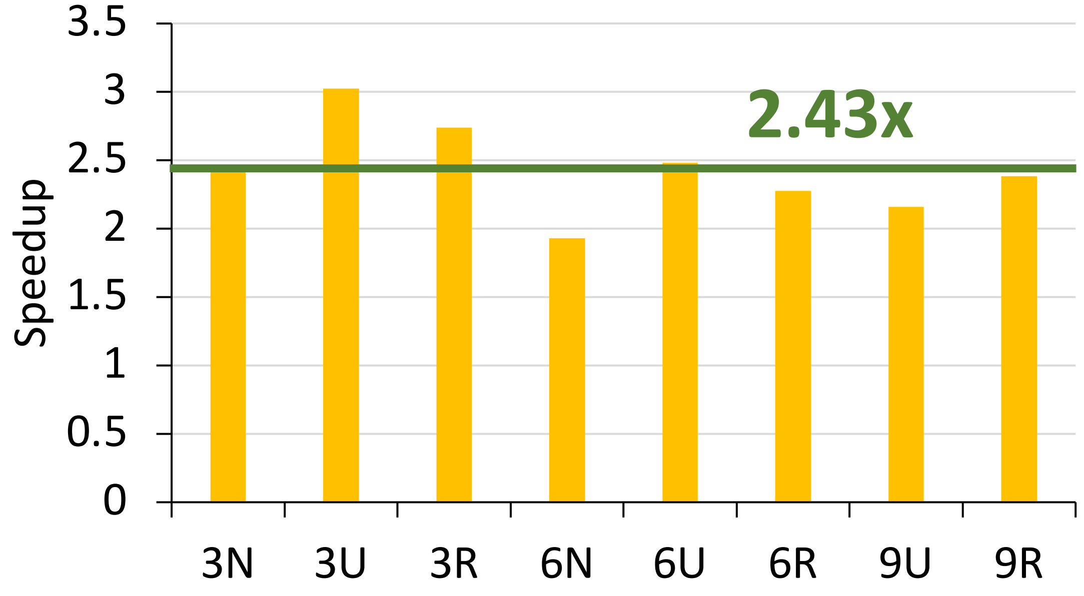
Result & Analysis (cont.)

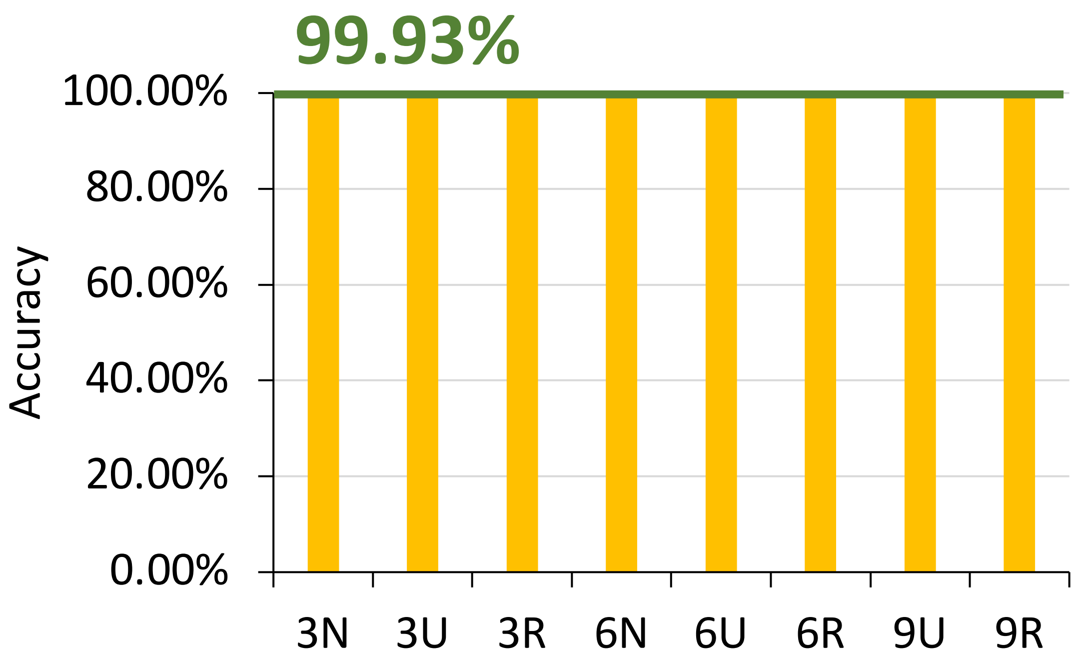
- Automation Efficiency
  - Less is better
  - Average execution count of moTuner is **0.73**
  - moTuner reduces up to **81.2%** tunning effort and **67.8%** in average.



Result & Analysis (cont.)

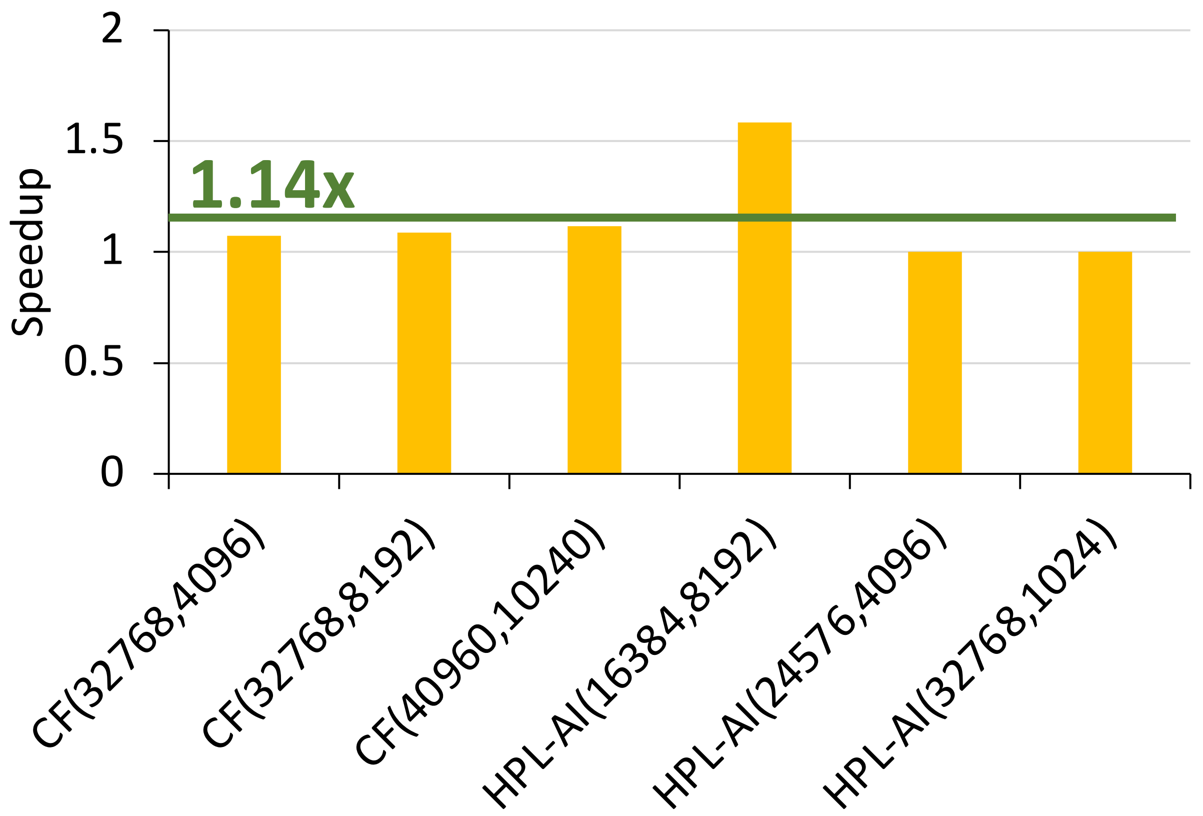
- Micro
  - Input setting is denoted as #GEMM and data distribution
  - N: Normalized (0,0.5), R: Random (-1,1),U: Uniform (-0.5,0.5)
  - moTuner achieves **2.43x** speedup and **99.93%** accuracy in average

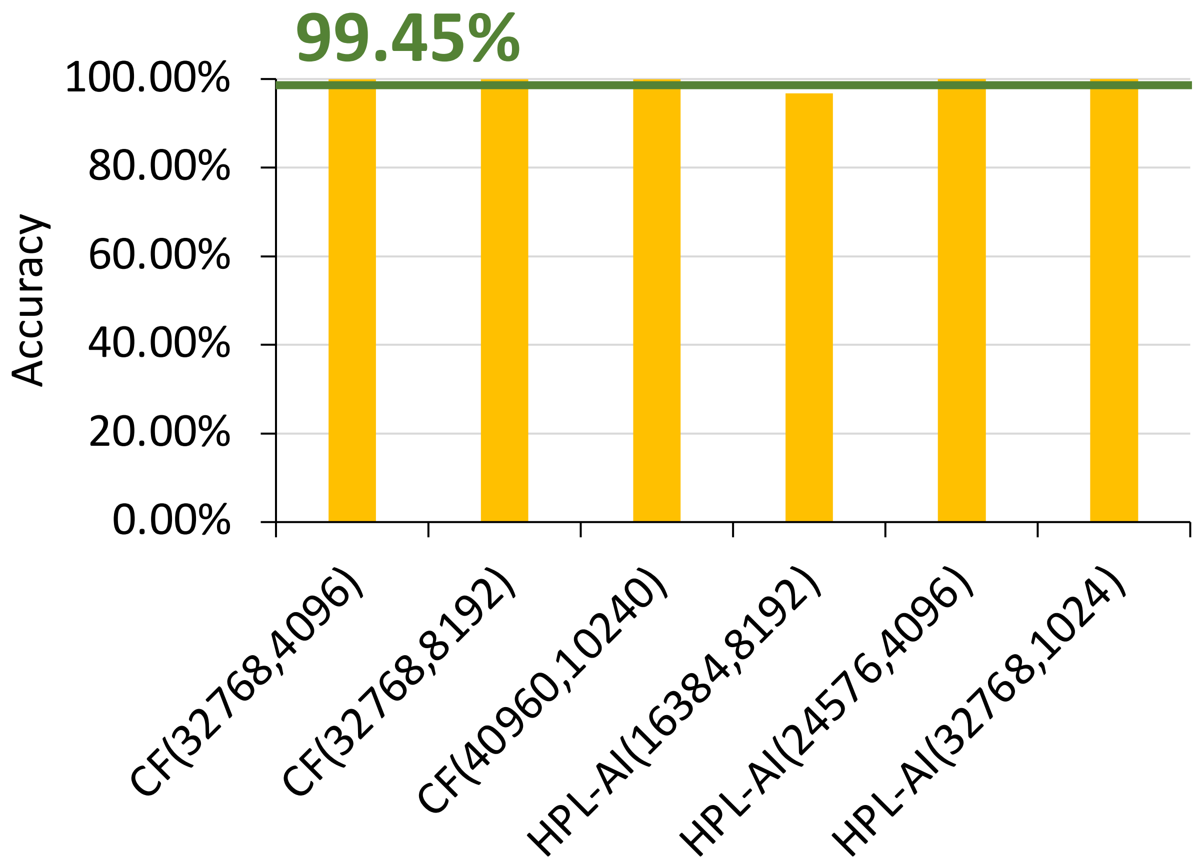




Result & Analysis (cont.)

- CF and HPL-AI
  - moTuner achieves **1.14x** speedup and **99.45%** accuracy in average





Agenda

- Background
  - Mixed-precision
  - Operator
  - Compilation
- Motivation
- Design
  - Data Dependency Analysis
  - Setting Tuning
- Evaluation
- Summary

Summary

- moTuner: An auto-tuning approach aiming at mixed-precision operators

- Basic Design:
  - Finds related operators for one under every given input
  - Upgrades settings of related operators when intolerable error occurs

- Result:
  - Provides **1.92x** speedup and **97.12%** accuracy in average
  - Has great robustness in different scenarios

- Future work:
  - Support more complex operators on various hardware in future

Thank you

moTuner: A Compiler-based Auto-tuning Approach for Mixed-precision Operators

Zewei Mo, Zejia Lin, Xianwei Zhang, Yutong Lu
mozw5@mail2.sysu.edu.cn

SUN YAT-SEN UNIVERSITY

ARCSYSU
Refines Computing SYStem Uses

NORTHWESTERN POLYTECHNICAL UNIVERSITY

moTuner: https://github.com/MoZeWei/moTuner

Backup Slides

Compilation (cont.)

- All executable files are generated through compilation from source code