Ladder: *Enabling Efficient Low-Precision Deep Learning Computing through Hardware-aware Tensor Transformation*

Lei Wang [†◊], Lingxiao Ma [◊], Shijie Cao [◊], Quanlu Zhang [◊], Jilong Xue [◊], Yining Shi [‡◊]
Ningxin Zheng [◊], Ziming Miao [◊], Fan Yang [◊], Ting Cao [◊], Yuqing Yang [◊], Mao Yang [◊]

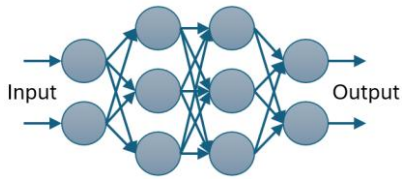
† *University of Chinese Academy of Sciences*

‡ *Peking University*

◊ *Microsoft Research*

https://github.com/microsoft/BitBLAS

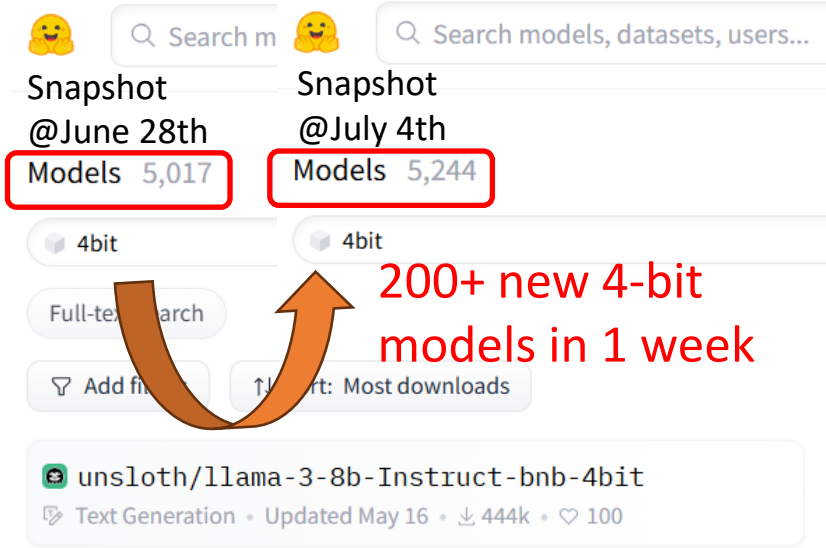
Microsoft



Stable Diffusion

LLAMA-65B
LLAMA-2-70B
LLAMA-3-400B

FP32 → FP16 | FP8 | MXFP | INT4 | FP4 | INT1 | .....



200+ new 4-bit models in 1 week

Larger Scale, Fewer Bits

1000X Speedup in 10 years
From FP32@M60 to **FP4**@B200

NVIDIA GPU

FP32: 9 TFlops

Maxwell
FP32 FMA

Pascal
+FP16 HFMA2
+INT8 DP4A

Volta
+FP16 HMMA

Ampere
+TF32 HMMA
+BF16 HMMA
+INT8/4/2 IMMA
+INT1 BMMA

Hopper
+FP8 QMMA

FP32: 80 TFlops
FP4: 9000 TFlops

Blackwell
+FP6 MMA
+FP4 MMA

2014 2016 2017 2020 2022 2024

Supporting Low-Precision DNN Computing is Challenging

- Lower-bit numeric precision
- Group-wise precision scaling
  - E.g., MXFP, group quantization
- Mixed-precision operations
  - E.g., FP16 X INT4/NF4, INT8 X INT1

Still in fast evolving ……

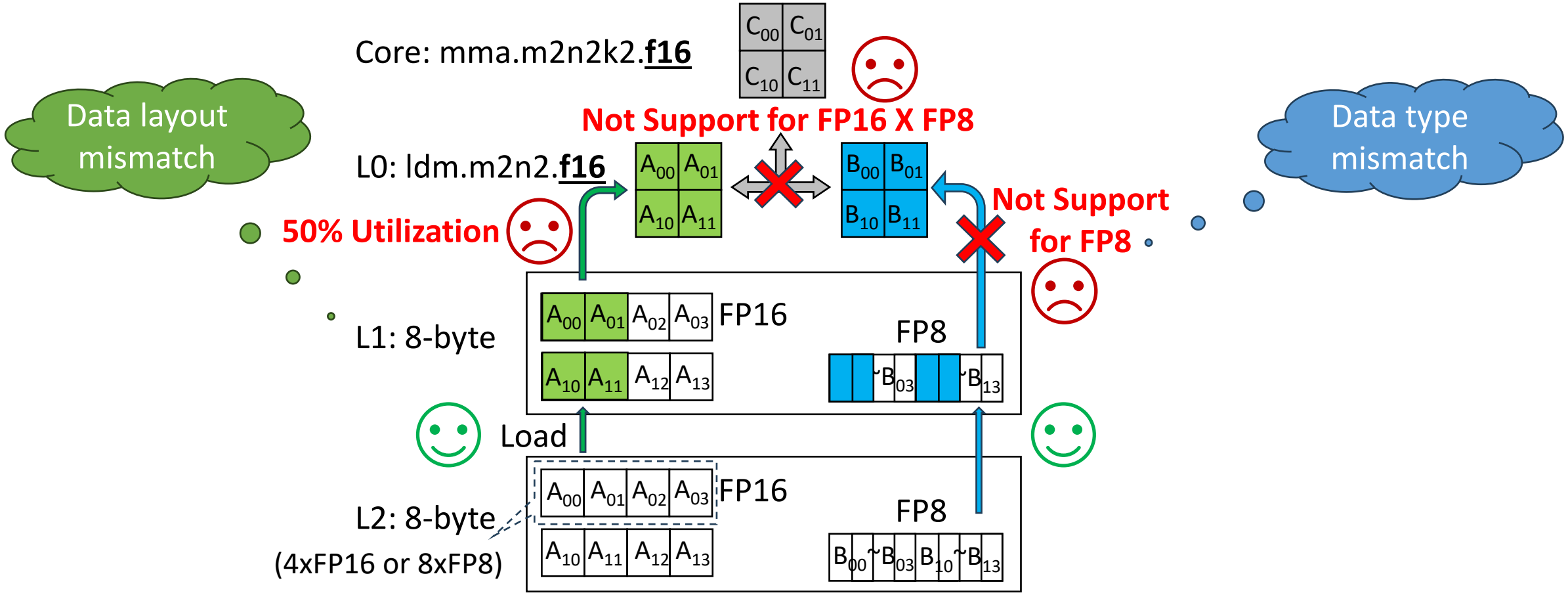
Insufficient precision supports
- Should use proxy types: e.g., FP16 for FP4

Inefficiency of low-precision computing
- Avg. utilization < 60% in FP16/INT8 MatMul

……

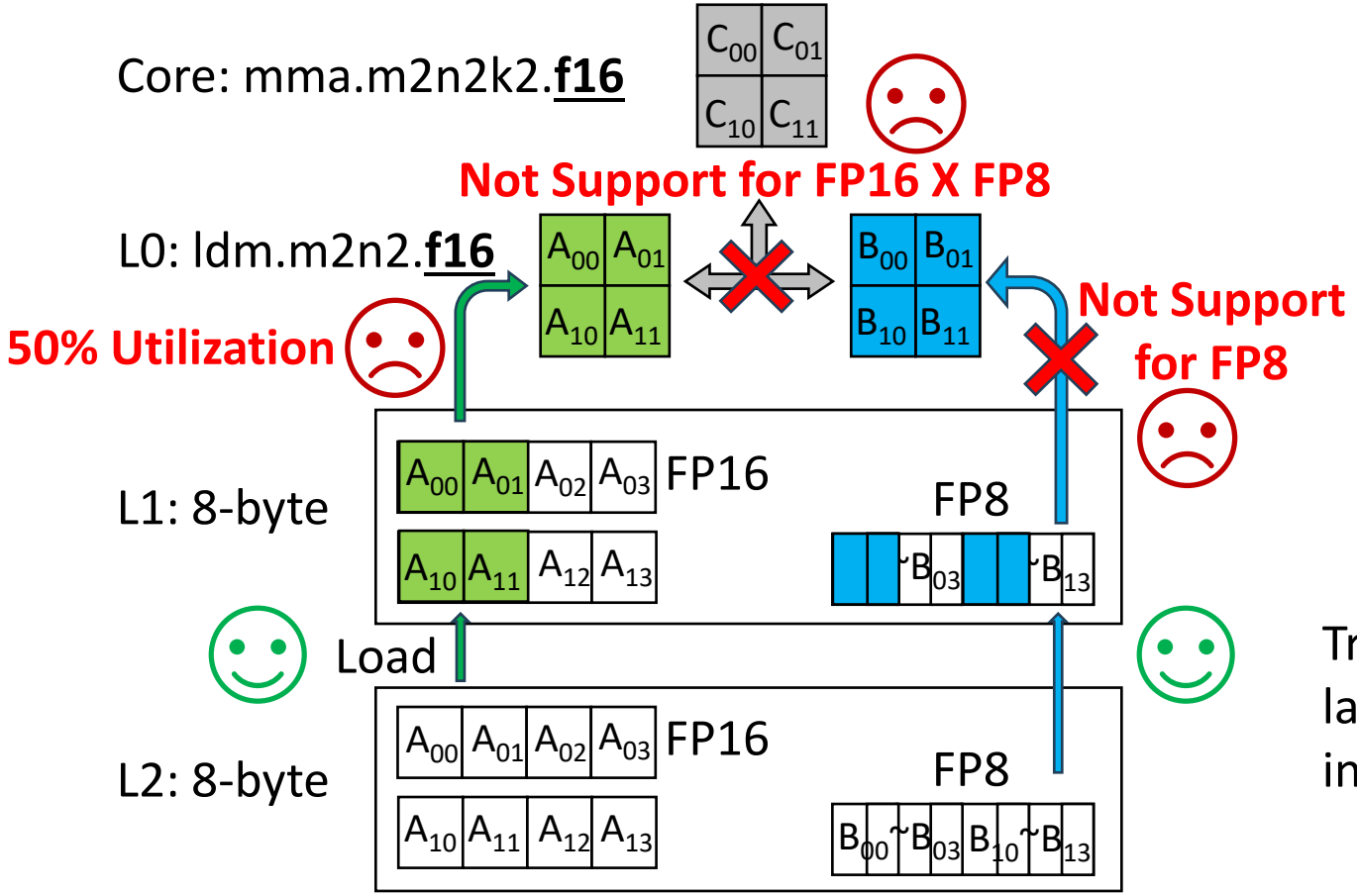
Example of Inefficiency and Insufficiency

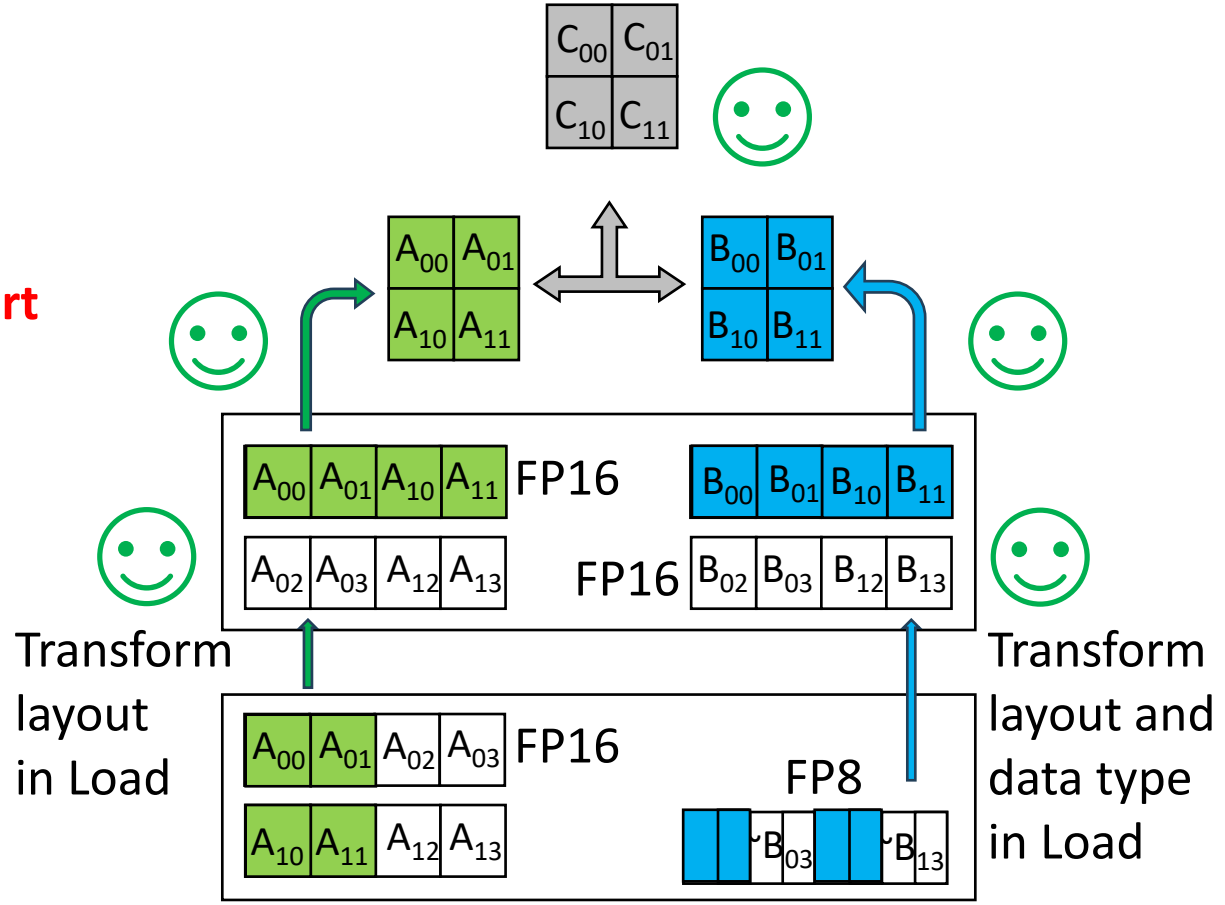
- Matrix multiplication: C[M,N]@FP16=A[M,K]@FP16 X B[N,K]@FP8, M=2, N=2, K=4
- Computation-coupled scheduling results in mismatches in hardware hierarchy



Insights

- De-couple ***computation*** and ***storage*** to enable data flexibilities in each layer
- Low-bit data type can be ***converted*** to a lossless higher-bit data type for processing
- Align with hardware through ***transformations***

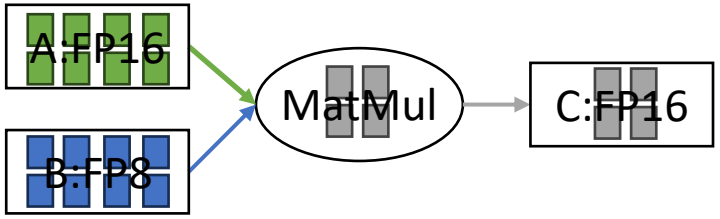




Ladder

FP16: shape=[1], nElemBits=16

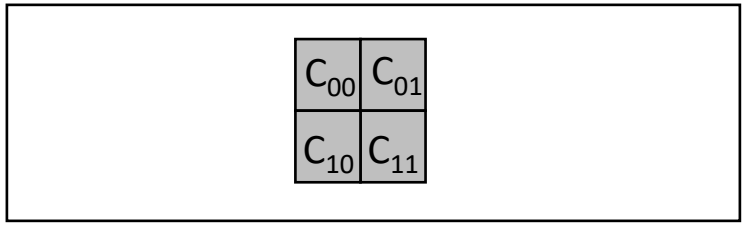
FP8: shape=[1], nElemBits=8, c_tTypes={FP16}



- Challenge 1: expose the representation for evolving data types
- Solution: TypedTile (tTile) abstraction
  - tType: a group of homogeneous n-bit elements
  - Augment tile-based tensor abstraction with tType
  - Abstract hardware as tTile processor to represent hardware capability

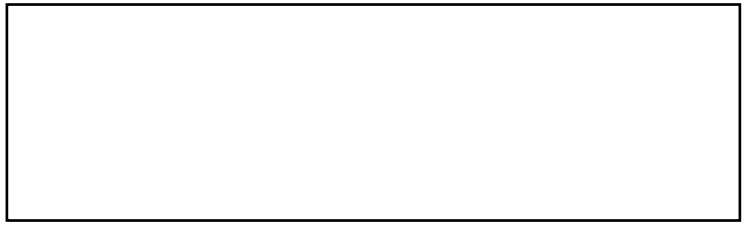
Core: mma.m2n2k2.f16

tTile: shape=[2,2], type=FP16



L0: ldm.m2n2.f16

tTile: shape=[2,2], type=FP16



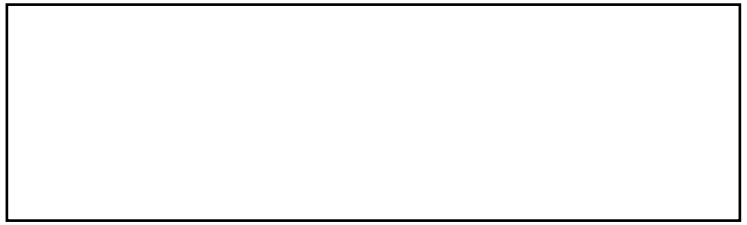
```
class tType {
 TileShape shape;
 size_t nElemBits;
 struct metadata;
 map<TileType, c_func> c_tTypes;
};
```

```
class tTile {
 TileShape shape;
 tType type;

 struct metadata;
};
```

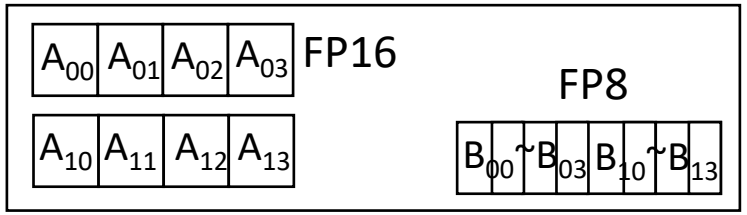
L1: 8-byte

tTile: shape=[8], type=1B



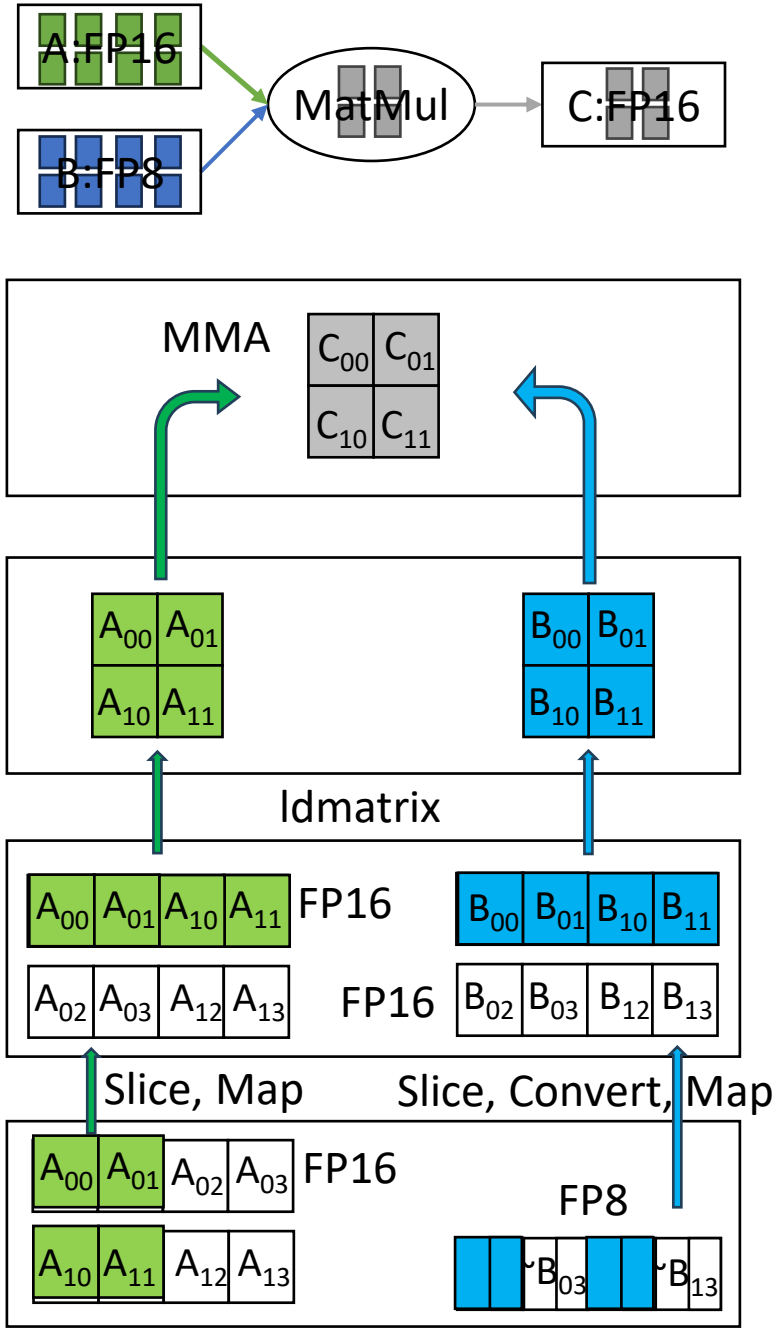
L2: 8-byte

tTile: shape=[8], type=1B

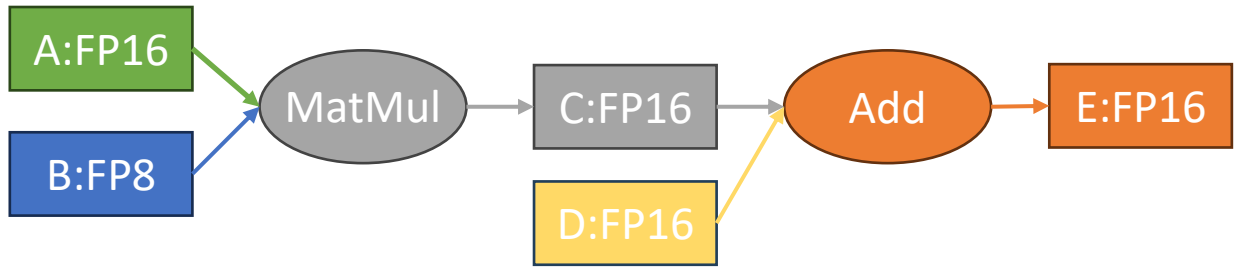


Ladder

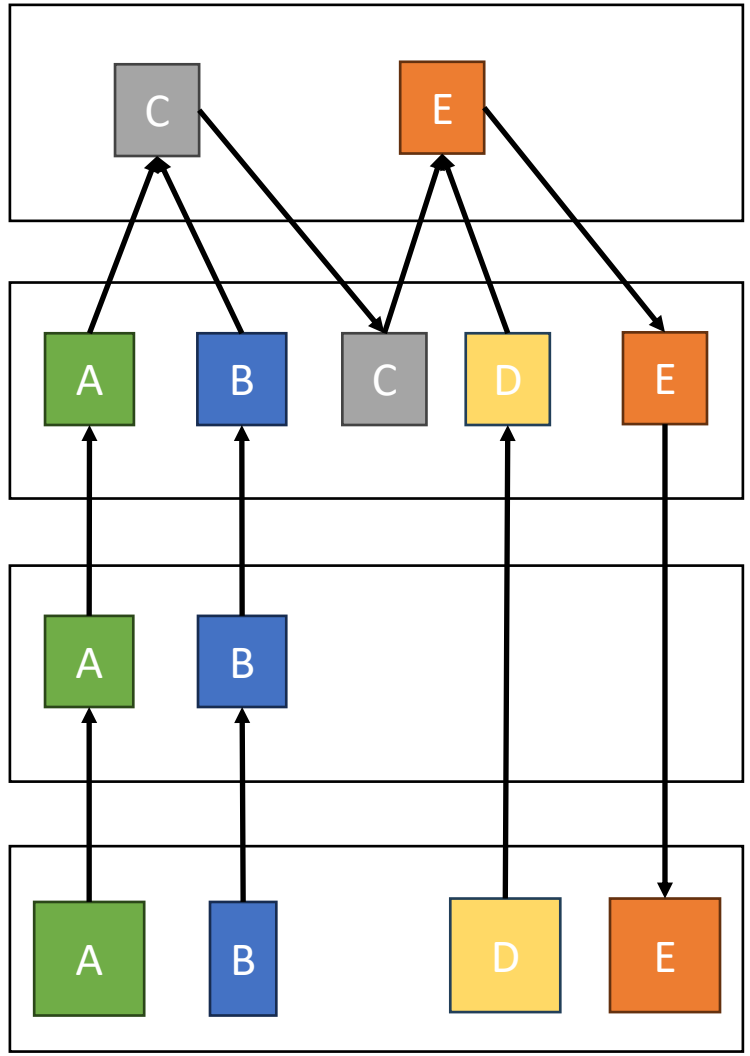
- Challenge 2: tTiles from model mismatch hardware support
- Solution: de-couple computation as pipeline of tTile transformations
- Pipeline:
  - Transform-Load: load and transform a tTile
  - Compute: process a tTile on the core
  - Transform-Store: transform and store a tTile
- tTile transformation:
  - Slice: slice a tTile from the input tTile
  - Map: modify tTile element layout
  - Pad: pad tTile shape with given value
  - Convert: convert tType to another tType



Ladder



- Challenge 3: tTile transformations introduce much larger space for tile-based model scheduling
  - New latency-memory trade-off: may trade-off memory consumption to reduce transformation cost
- Our practice: consider tiling and transformation separately
  - Hardware hints: preferred data access granularity
  - Tiling schedule: calculate traffic and decide tiling
    - Can leverage existing methods (e.g., TVM, Welder)
  - Transformation schedule: add transformations
  - Effective but potentially sub-optimal

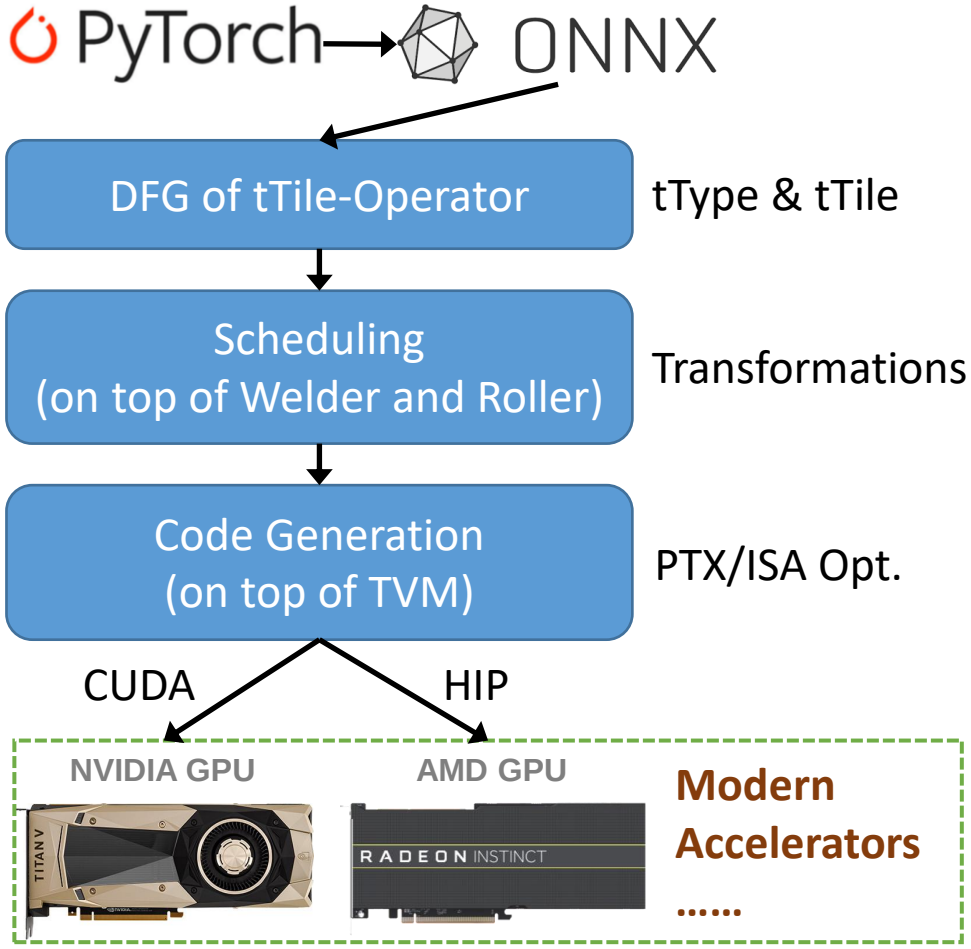


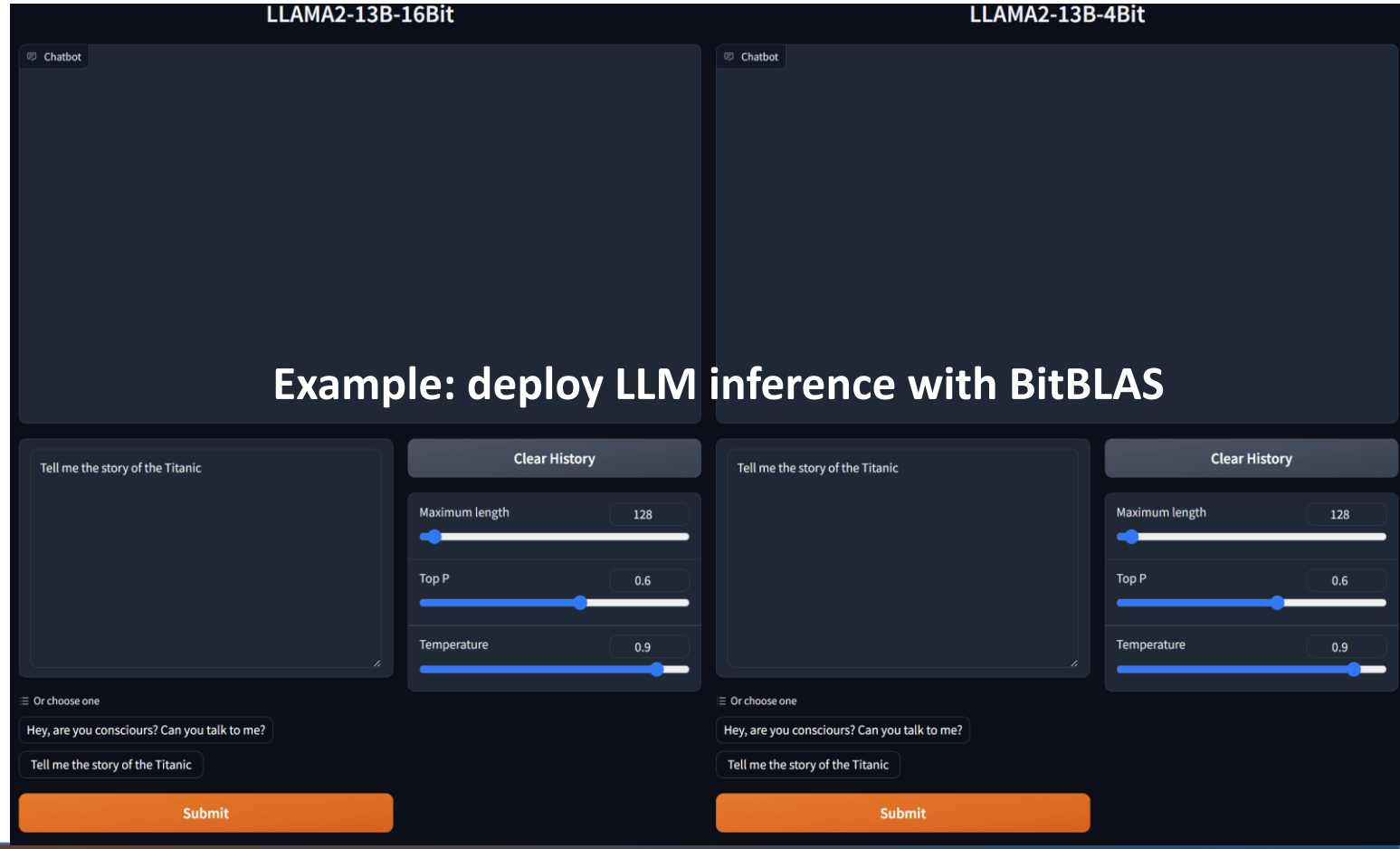
Please find our paper for detailed policy

Ladder Open-Source Implementation

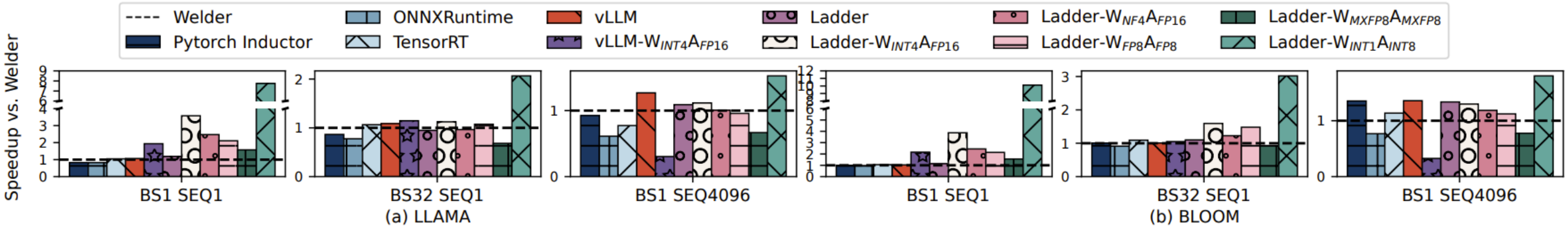
- Artifact: https://github.com/microsoft/BitBLAS/tree/osdi24_ladder_artifact

- BitBLAS: https://github.com/microsoft/BitBLAS
  - PyTorch Library for low-precision DNN operations on GPUs
  - Integration: PyTorch, vLLM (in PR), AutoGPTQ, BitNet-b1.58, BitDistiller





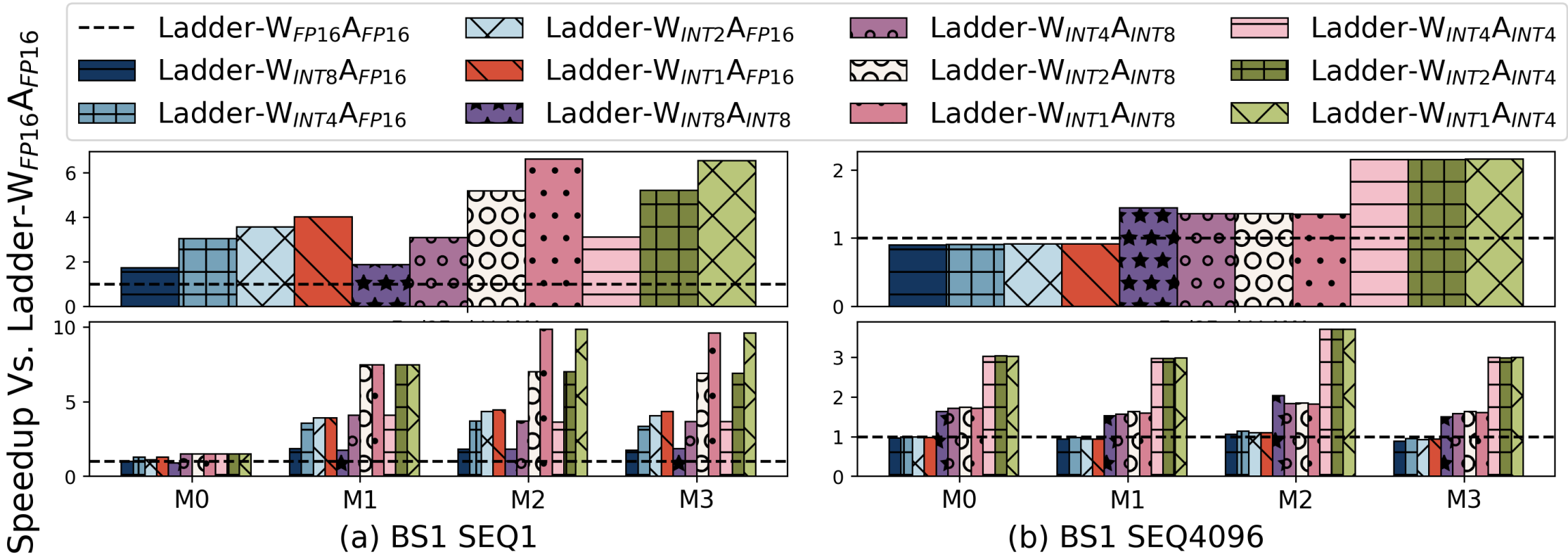
End-to-end LLM Inference on NVIDIA A100 GPU



- $W_{FP16}A_{FP16}$: ~ **1.1x/1.1x** avg. speedup over Welder/TensorRT
- $W_{INT4}A_{FP16}$ (GPTQ) ~ **2.3x** avg. speedup over vLLM-$W_{INT4}A_{FP16}$
- $W_{INT1}A_{INT8}$ (BitNet): up to **8.8x** speedup over Ladder- $W_{FP16}A_{FP16}$ (on BLOOM-176B-BS1SEQ1)

All data are evaluated on a single layer of LLMs (more details and evaluation in paper).

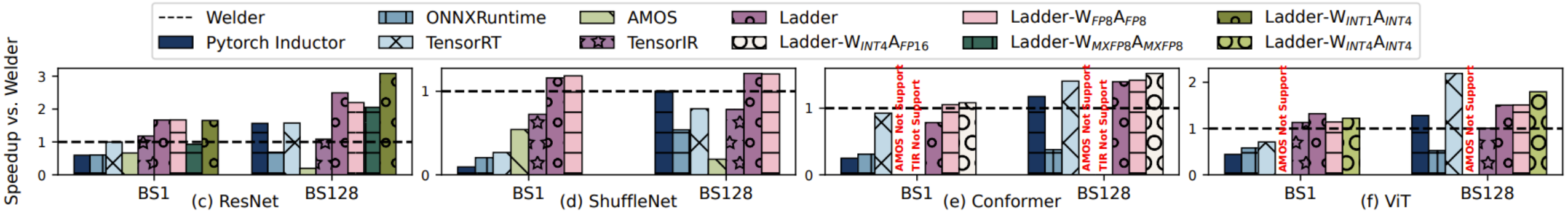
Scaling Bit Width for Low-Precision LLAMA2-70B



- BS1 SEQ1: bounded by memory bw., up to 6.4x speedup (10.1x speedup on kernel)
- BS1 SEQ4096: bounded by tensor core, up to 2.4x speedup (3.7x speedup on kernel)

All data are evaluated on a single layer of LLMs (more details and evaluation in paper).

End-to-end DNN Inference on NVIDIA A100 GPU



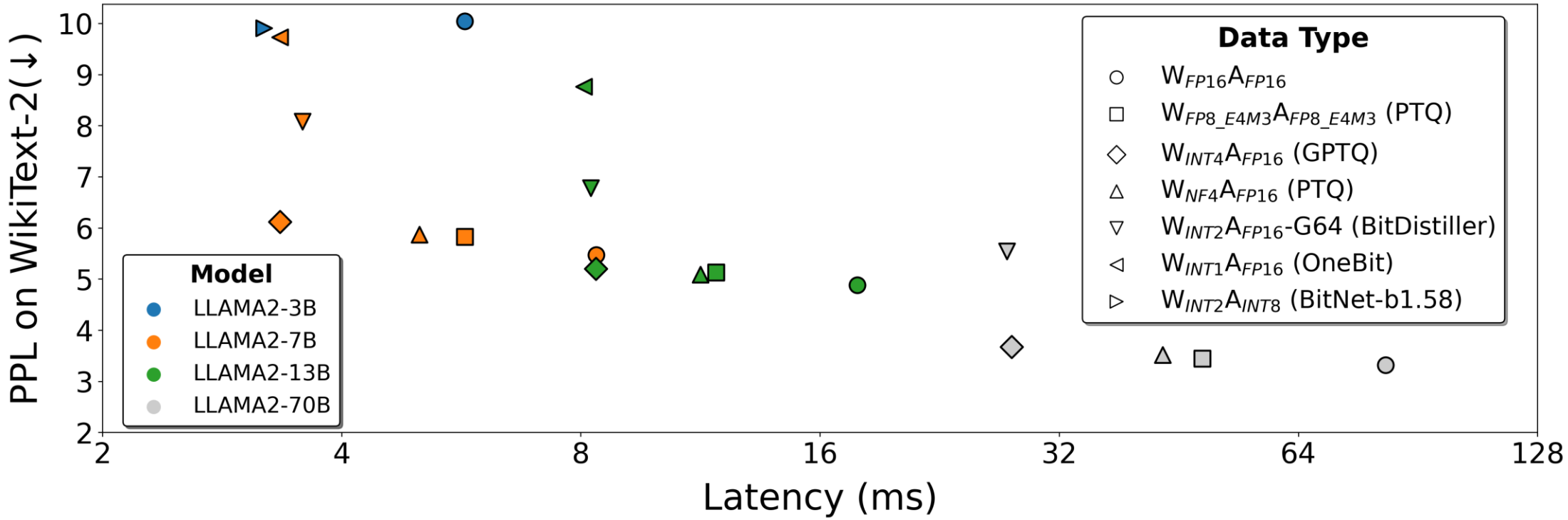
- $W_{FP16}A_{FP16}$: ~ **1.4x/1.7x** avg. speedup over Welder/TensorRT
- Enable low-precision DNN computing with up to **3x** speedup over Ladder- $W_{FP16}A_{FP16}$

Model (BS)	AMOS	TensorIR	Welder	LADDER
ResNet (1)	3852	156	11	31
ResNet (128)	2191	836	18	44
ShuffleNet (1)	3328	128	13	17
ShuffleNet (128)	3121	400	12	29

Compilation time (in minutes) of $W_{FP16}A_{FP16}$ models

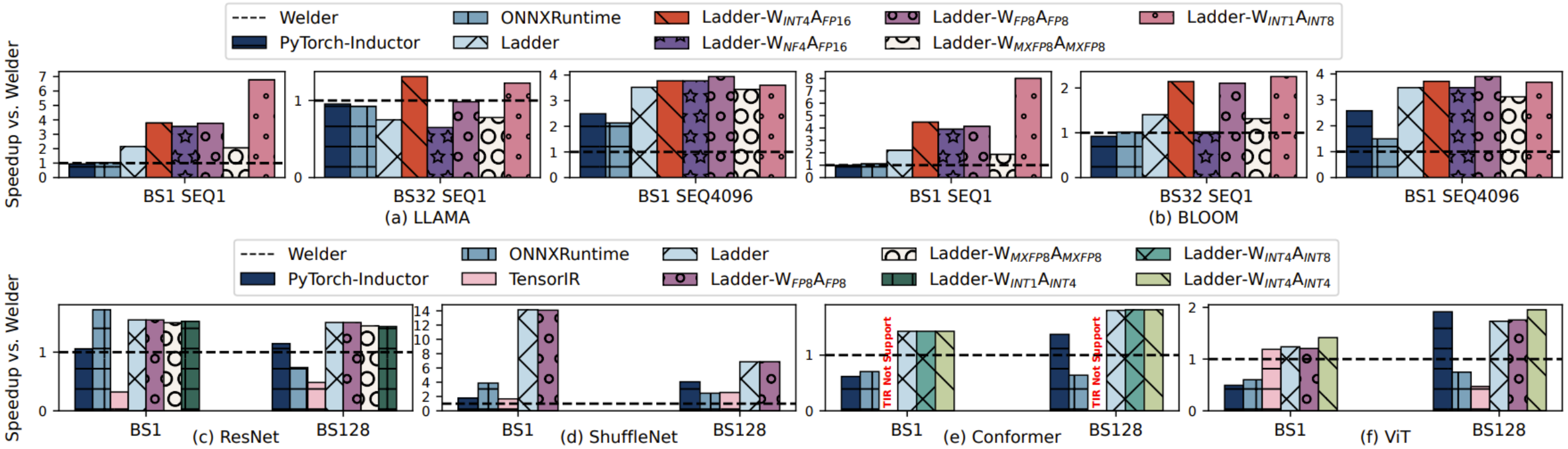
- **< 1h** for compiling an end-to-end model
- **10x** faster than TensorIR
- **100x** faster than AMOS

Efficiency and Accuracy of Low-Precision LLMs



- 4-bit weight quantization (e.g., INT4, NF4) achieves **similar PPL** over $W_{FP16}A_{FP16}$
- Low-precision 7B models achieve **better PPL and efficiency** over 3B on $W_{FP16}A_{FP16}$
- BitNet-b1.58-3B achieves **better PPL with 1.8x speedup** over $W_{FP16}A_{FP16}$ on LLAMA2-3B

End-to-end Inference on AMD MI250 GPU



- $W_{FP16}A_{FP16}$: ~ **3.3x** on average speedup over Welder
- Enable low-precision DNN computing with up to **3.7x** speedup over Ladder- $W_{FP16}A_{FP16}$

Data of LLAMA and BLOOM are evaluated on a single layer (more details and evaluation in paper).

Conclusion

- **Increasing low-precision computing** is observed in deep learning
- Ladder proposes **TypedTile abstraction** and **tensor transformations** that
  - Represent evolving data types
  - Provide a systematic mechanism for low-precision computing
- Hope to empower both model designers and hardware vendors in taking advantage of emerging trends

BitBLAS GitHub Repo

https://github.com/microsoft/BitBLAS