On-the-Fly Data Layout Conversion for GEMM on AI Accelerators

Frank (Fang) Gao^{1}, Xingyu Yue^{1}, Chenchen $Tang^{1}$, Hongyi $Chen^{1}$, Amy $Wang^{1}$, Tarek S. $Abdelrahman^{2}$

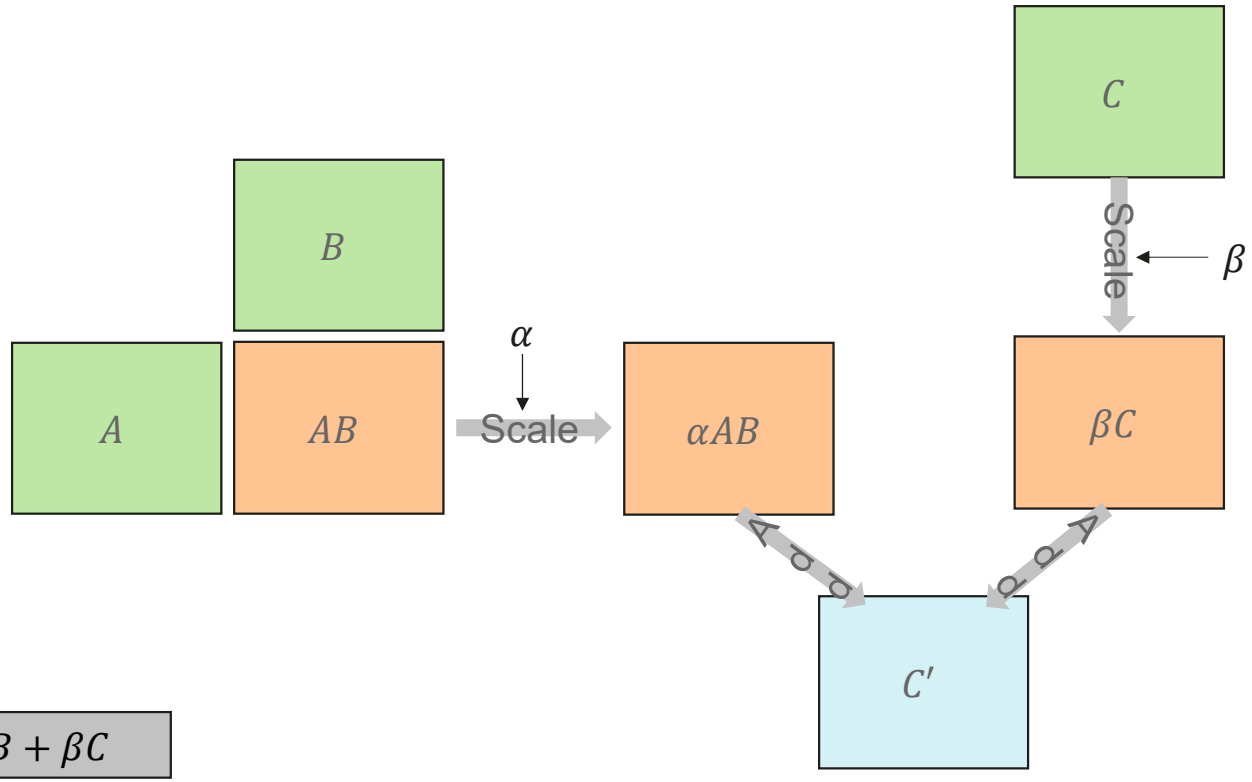
[1] Huawei Toronto Research Centre
[2] University of Toronto

HUAWEI

AI Accelerators

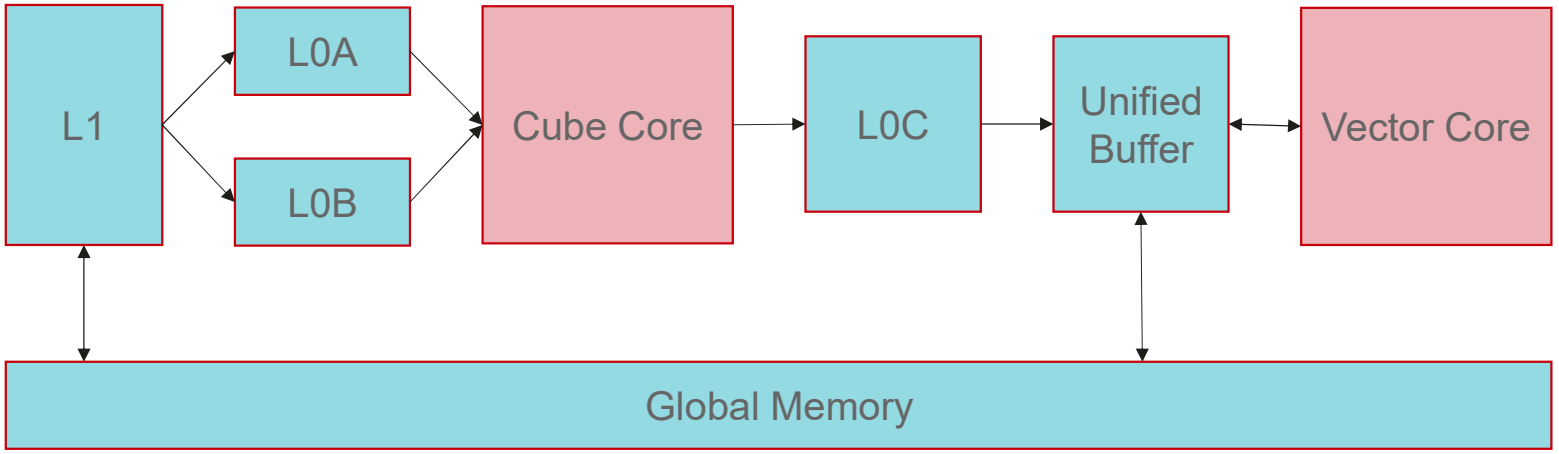
HUAWEI	Google	intel	NVIDIA
Ascend	TPU	Gaudi, AMX	A100, H100, …

GEMM in a Nutshell



$C := \alpha AB + \beta C$

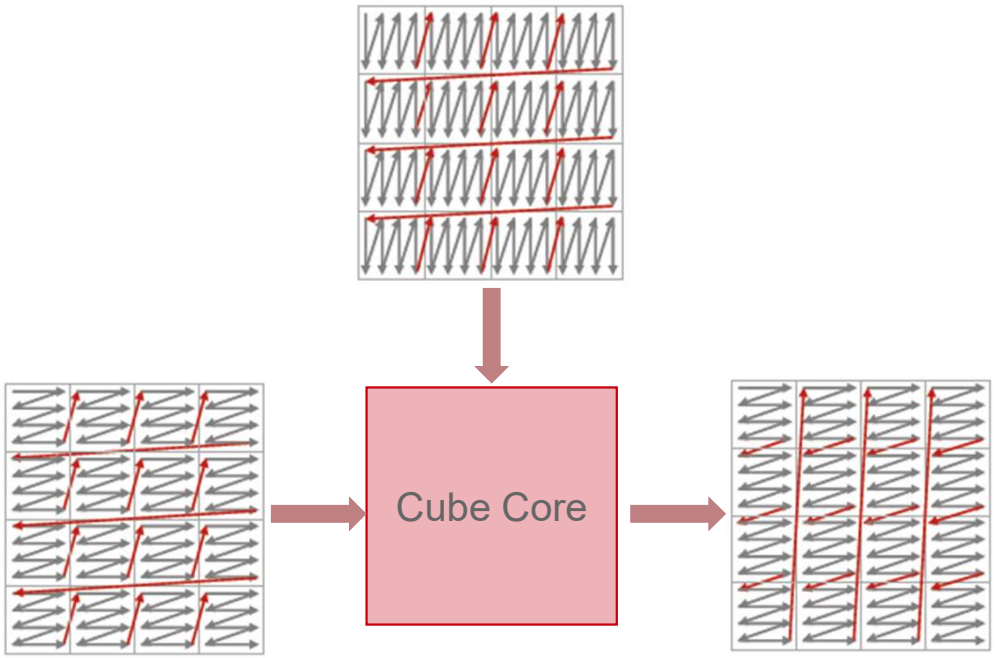
Huawei Ascend 910 Architecture (Last Gen)



- Vector Core performs scaling and addition
- Cube Core accelerates Matrix Multiply
  - Requires special data layout
- DMA/Cube/Vector cores are pipelined, can execute in parallel

The Cube Core

- Systolic Array
- Accepts A in zZ format, B in nZ format
- Outputs C in zN format



The “Fractal” Data Layout

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

zZ

zN

0	1	4	5	8	9	12	13	2	3	6	7	10	11	14	15

nZ

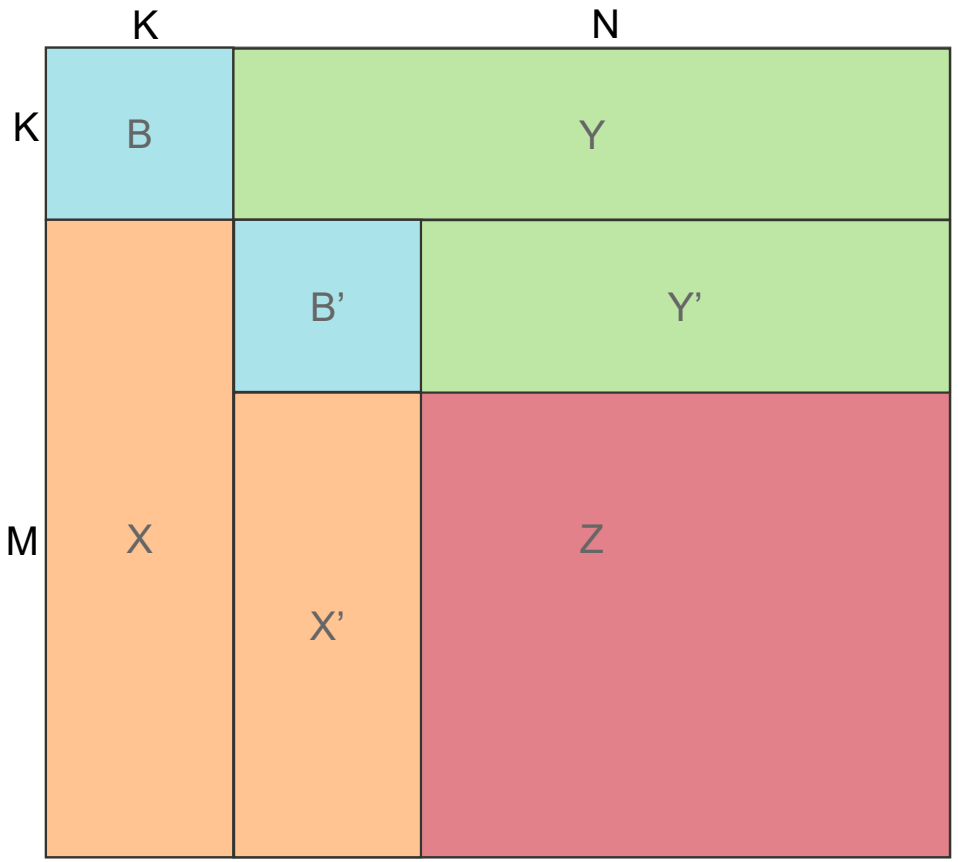
0	4	1	5	2	6	3	7	8	12	9	13	10	14	11	15

- Assuming 2x2 “fractals”
  - Hardware requires 16x16 “fractals”

Motivating Example: Blocked LU Factorization

1. LU Factorization on B
2. TRSM to update X
3. TRSM to update Y
4. **GEMM to update Z**
5. Repeat…

- Traditionally GEMM is >90% of the runtime
- Well established TRSM routines
  - Requires row/column major input
- CPU layout conversion is costly

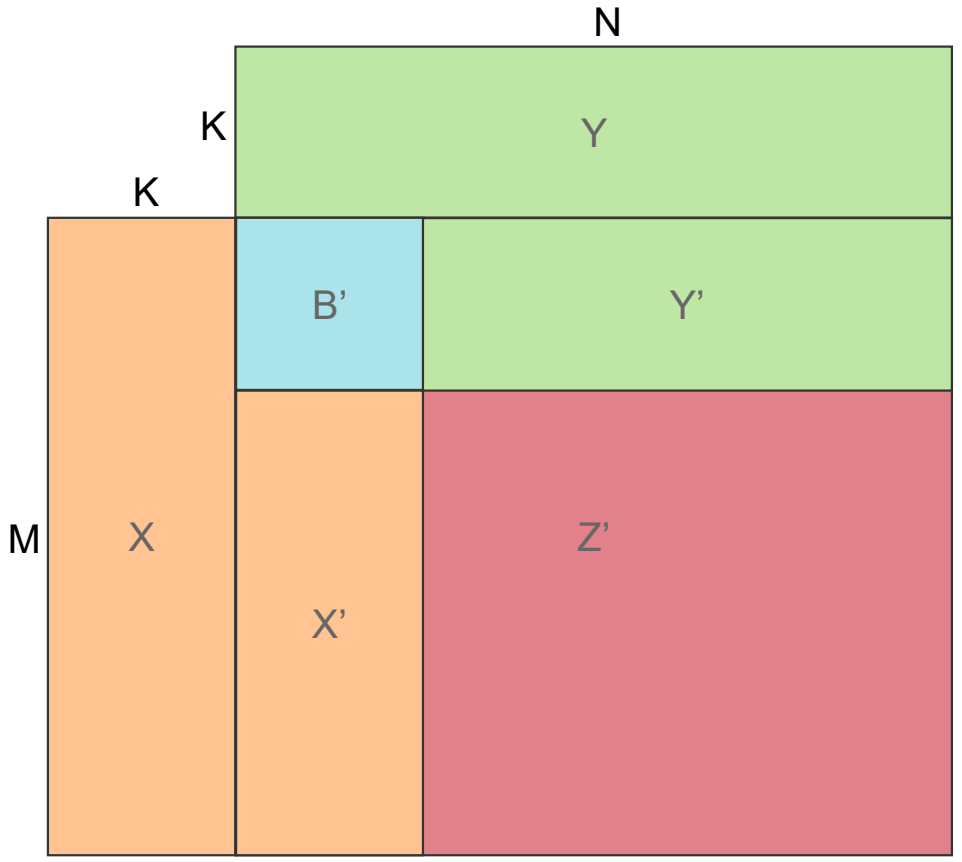


Caveat

- Experiment done on Ascend 910
- Ascend 910B includes hardware features for layout conversion on-the-fly

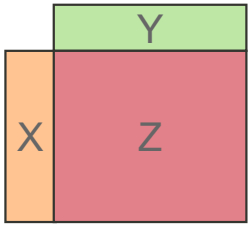
Required Conversions for Blocked LU

- Convert Y on CPU to nZ
- Convert Z on CPU to zN
- Convert X on Device *On-The-Fly*
- Convert output Based on Location



Conversion Using DMA

- DMA Instruction primary parameters:
  - > Length
  - > Repeat
  - > Stride
- DMA instructions can also transpose
  - > Less efficient
  - > Pre-convert Y tile
- Possible Conversions:
  - > zZ ⇔ zN ⇔ Row Major
  - > nZ ⇔ Column Major



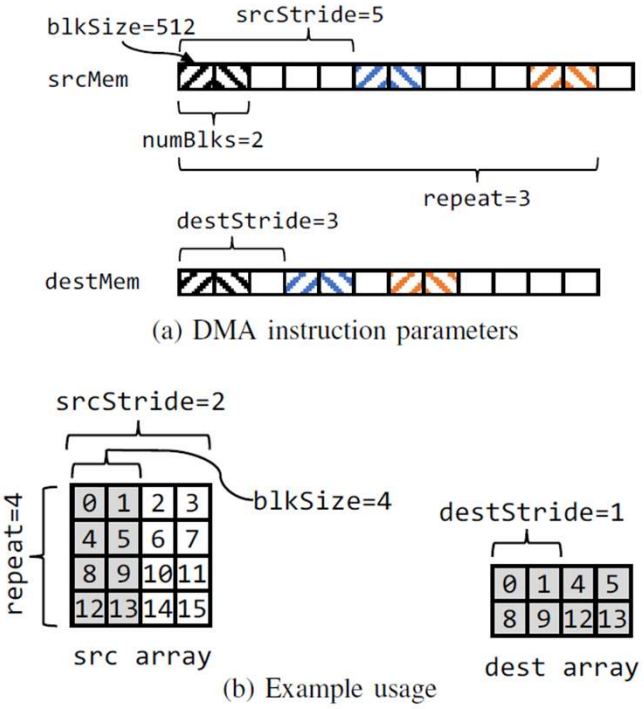
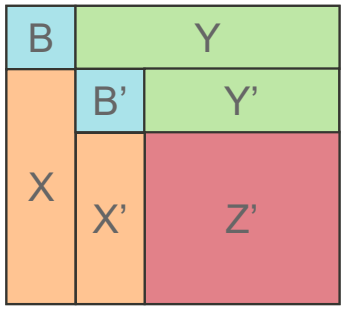


Fig. 3: The DMA `blockCopy` instruction

Conversion Using Vector Unit

- Three parameters per argument
  - > Block Stride
  - > Repeat
  - > Repeat Stride
- Operates in units of 8 “blocks” per instruction per argument



- Convert to Row Major for B', X', Y'
- Do not convert for Z'

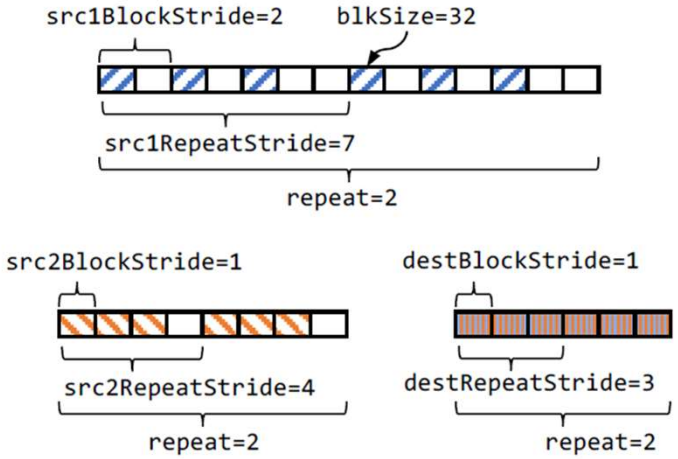
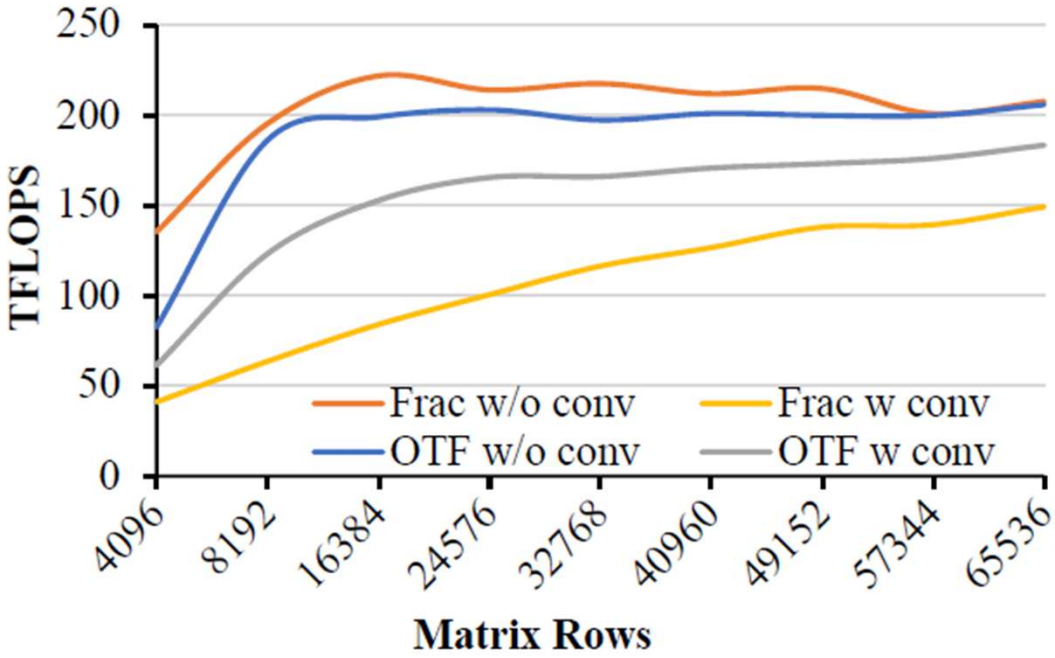


Fig. 4: Example of the Vector Unit instruction parameters

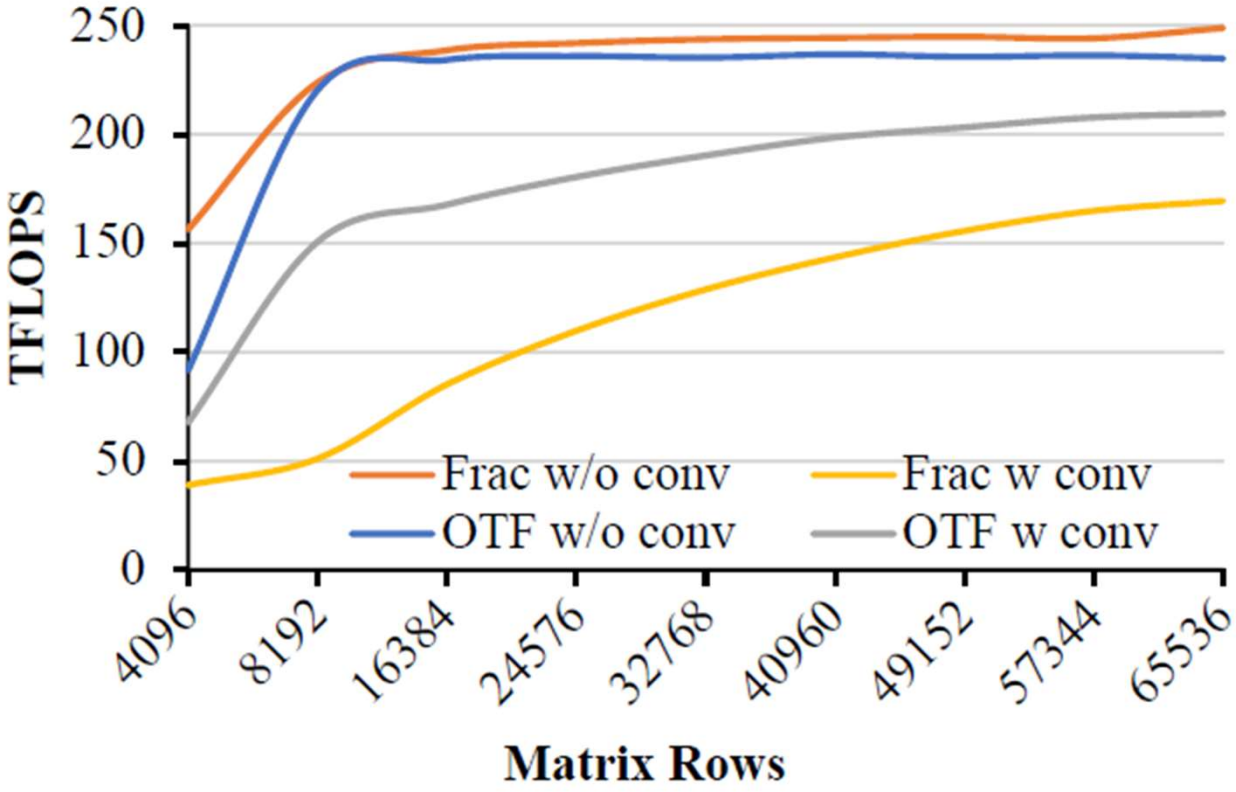
GEMM Performance: M=N, K=512

- On-The-Fly kernel slower by up to 15%
- On-The-Fly conversion better by ~20%

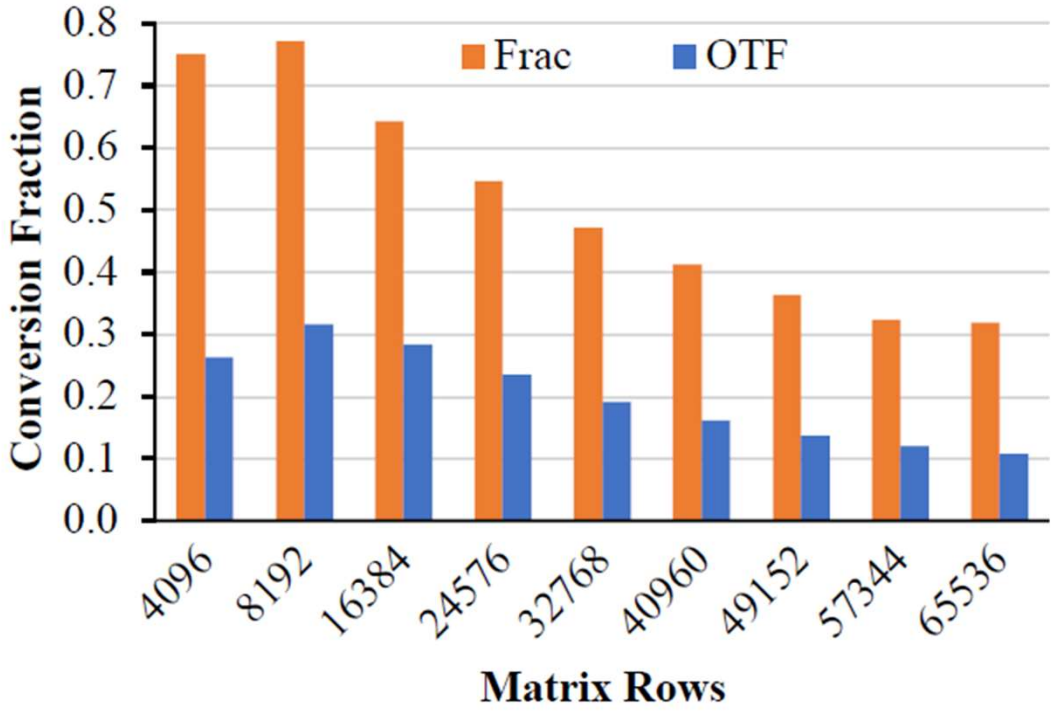
	Frac	OTF
Convert X to Fractal	√	×
Convert Y to Fractal	√	√
Convert Z to Fractal	×	×
Convert B', X', Y' to Row Major	√	×
Convert Z' to Row Major	×	×



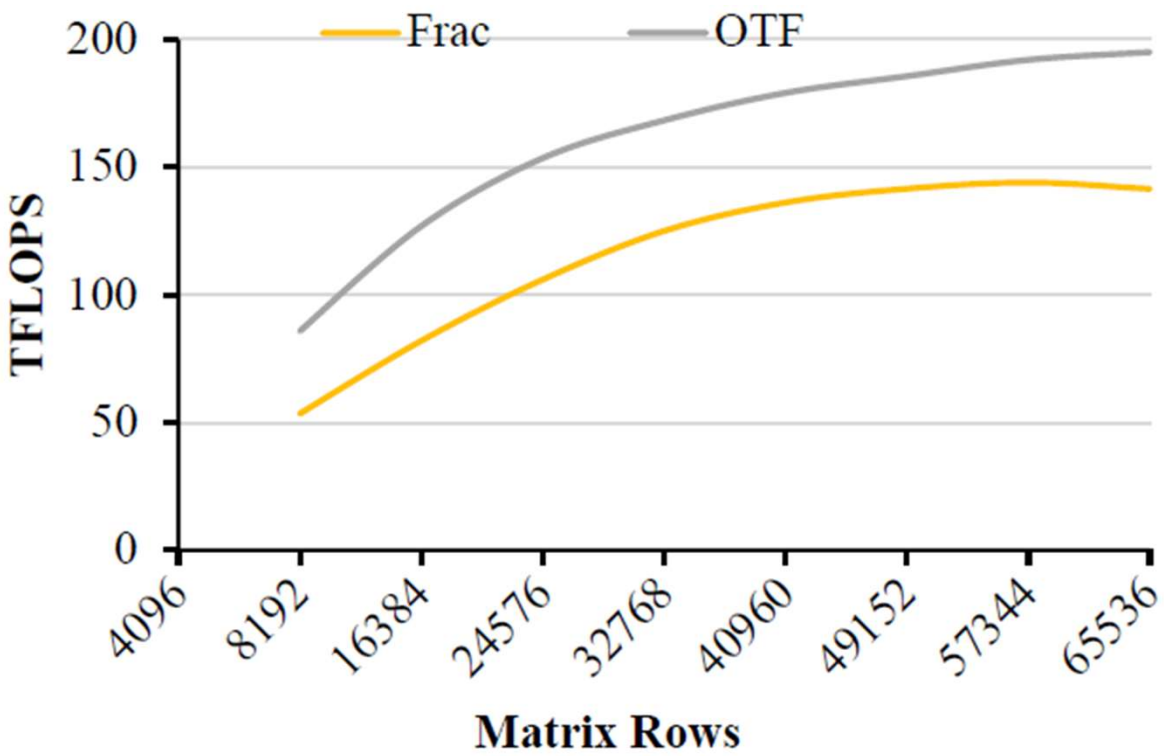
GEMM Performance: M=N, K=1024



Conversion Time as Fraction of Runtime



GEMM Performance Over LU Factorization



Conclusion

- 5% improvement in LU runtime overall
  - > GEMM becomes too fast!
- Method applicable for column major conversions
- Applicable to other applications other than LU
  - > Could have bigger impact!
- Depending on architecture, could also apply to other accelerators

Thank you.

Bring digital to every person, home and organization for a fully connected, intelligent world.