NVIDIA

TENSOR CORE ACCELERATED ITERATIVE REFINEMENT SOLVERS AND ITS IMPACT ON SCIENTIFIC COMPUTING

Azzam Haidar | Sr. Engineer – CUDA Math Libraries

Harun Bayraktar | Sr. Manager – CUDA Math Libraries

Sparse Days, CERFACS Toulouse virtual, Nov 23, 2020

NVIDIA A100 VS NVIDIA TESLA V100

Feature Highlights for Math Libraries

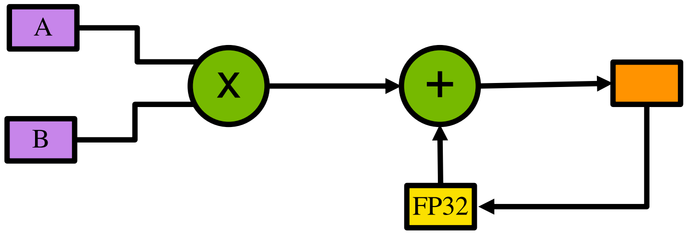
Data Center GPU Name	NVIDIA Tesla V100
GPU Codename	GV100
GPU Architecture	NVIDIA Volta
SMs	80
GPU Boost Clock	1530 MHz
Peak FP16 Tensor Core TFLOPS[1]	125
Peak Bfloat16 Tensor Core TFLOPS[1]	NA
Peak TF32 Tensor TFLOPS[1]	NA
Peak FP64 Tensor TFLOPS[1]	NA
Peak INT8 Tensor TOPS[1]	NA
Peak FP16 TFLOPS[1]	31.4
Peak Bfloat16 TFLOPS[1]	NA
Peak FP32 TFLOPS[1]	15.7
Peak FP64 TFLOPS[1]	7.8
Peak INT32 TOPS[1]	15.7
Memory Interface	4096-bit HBM2
Memory Size	32 GB / 16 GB
Memory Data Rate	877.5 MHz DDR
Memory Bandwidth	900 GB/sec
L2 Cache Size	6144 KB
Shared Memory Size / SM	Configurable up to 96 KB

1. Peak rates are based on GPU Boost Clock

NVIDIA A100 VS NVIDIA TESLA V100

Feature Highlights for Math Libraries

FP16/FP32 input	no precision loss product	FP32 accumulation	Output FP16, FP32



Data Center GPU Name	NVIDIA Tesla V100
GPU Codename	GV100
GPU Architecture	NVIDIA Volta
SMs	80
GPU Boost Clock	1530 MHz
Peak FP16 Tensor Core TFLOPS[1]	125
Peak Bfloat16 Tensor Core TFLOPS[1]	NA
Peak TF32 Tensor TFLOPS[1]	NA
Peak FP64 Tensor TFLOPS[1]	NA
Peak INT8 Tensor TOPS[1]	NA
Peak FP16 TFLOPS[1]	31.4
Peak Bfloat16 TFLOPS[1]	NA
Peak FP32 TFLOPS[1]	15.7
Peak FP64 TFLOPS[1]	7.8
Peak INT32 TOPS[1]	15.7
Memory Interface	4096-bit HBM2
Memory Size	32 GB / 16 GB
Memory Data Rate	877.5 MHz DDR
Memory Bandwidth	900 GB/sec
L2 Cache Size	6144 KB
Shared Memory Size / SM	Configurable up to 96 KB

1.Peak rates are based on GPU Boost Clock

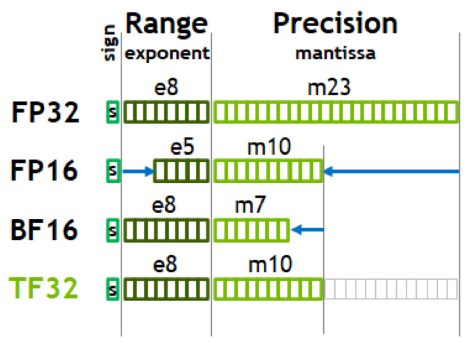
NVIDIA A100 VS NVIDIA TESLA V100

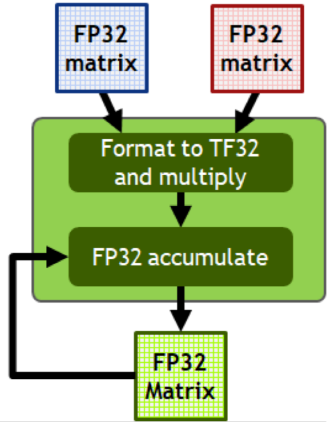
Feature Highlights for Math Libraries

Tensor Cores for:

- Mixed-precision Tensor Float 32 (TF32) and Bfloat16
- Full double precision (FP64) DMMA

Increased memory bandwidth 1.6 TB/s





Data Center GPU Name	NVIDIA Tesla V100	NVIDIA A100
GPU Codename	GV100	GA100
GPU Architecture	NVIDIA Volta	NVIDIA Ampere
SMs	80	108
GPU Boost Clock	1530 MHz	1410 MHz
Peak FP16 Tensor Core TFLOPS[1]	125	312
Peak Bfloat16 Tensor Core TFLOPS[1]	NA	312
Peak TF32 Tensor TFLOPS[1]	NA	156
Peak FP64 Tensor TFLOPS[1]	**NA**	**19.5**
Peak INT8 Tensor TOPS[1]	NA	624
Peak FP16 TFLOPS[1]	31.4	78
Peak Bfloat16 TFLOPS[1]	NA	39
Peak FP32 TFLOPS[1]	15.7	19.5
Peak FP64 TFLOPS[1]	7.8	9.7
Peak INT32 TOPS[1]	15.7	19.5
Memory Interface	4096-bit HBM2	5120-bit HBM2
Memory Size	32 GB / 16 GB	40 GB
Memory Data Rate	877.5 MHz DDR	1215 MHz DDR
Memory Bandwidth	900 GB/sec	1.6 TB/sec
L2 Cache Size	6144 KB	40960 KB
Shared Memory Size / SM	Configurable up to 96 KB	Configurable up to 164 KB

1.Peak rates are based on GPU Boost Clock

TENSOR CORE ACCELERATED LIBRARIES

Multi-precision numerical methods

Solving linear system of equations *Ax=b*

LU factorization is used to solve a linear system Ax=b

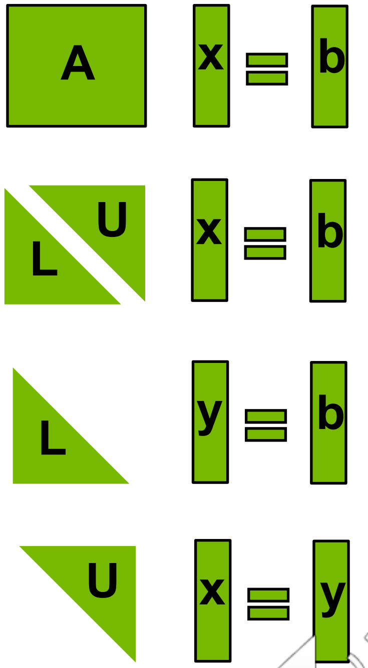
$A\ x = b$

$LUx = b$

$Ly\ = b$

then

$Ux\ = y$

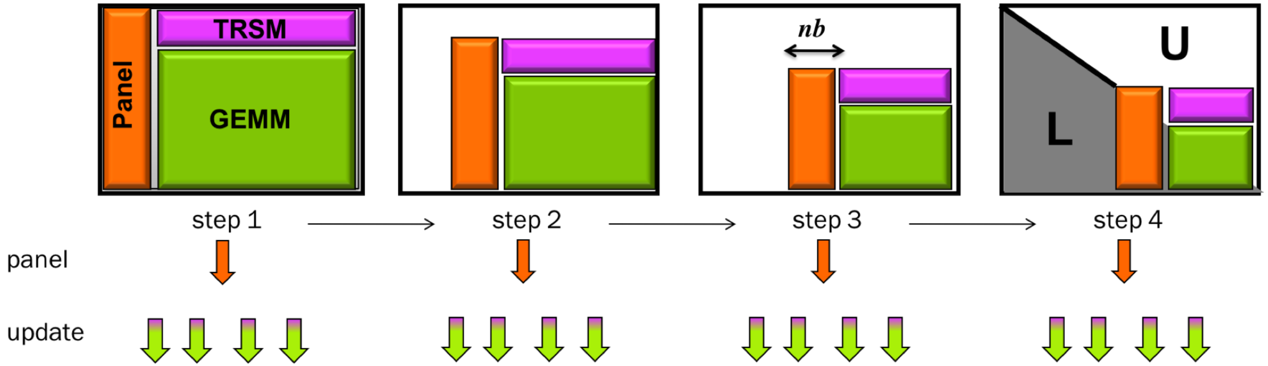


TENSOR CORE ACCELERATED LIBRARIES

Multi-precision numerical methods

Solving linear system of equations *Ax=b*

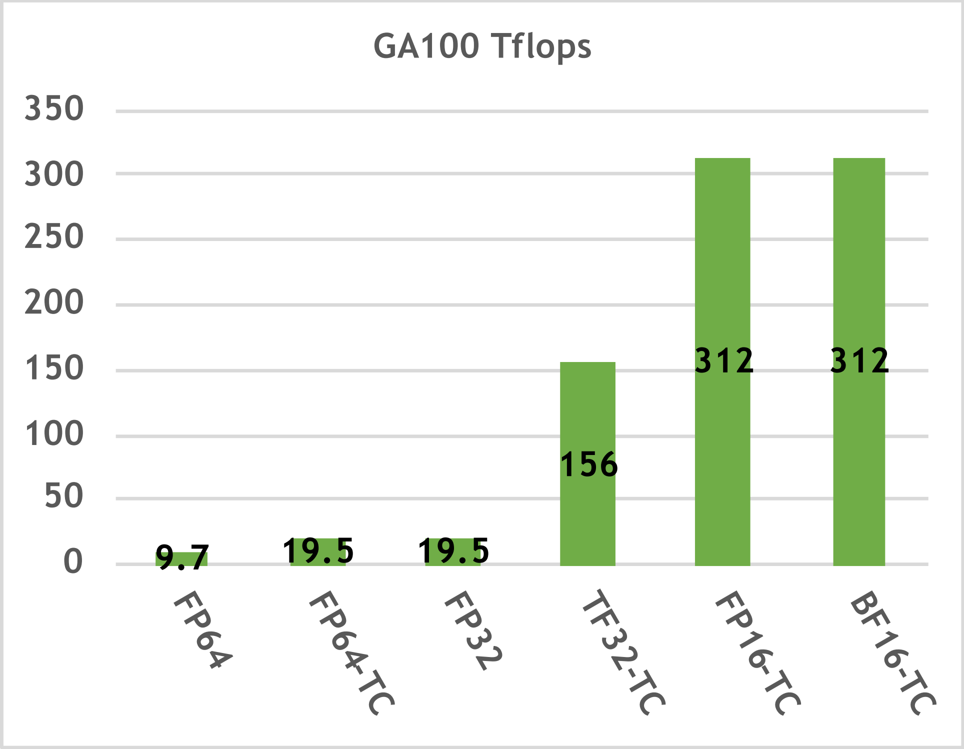
LU factorization used to solve Ax=b is dominated by GEMMs



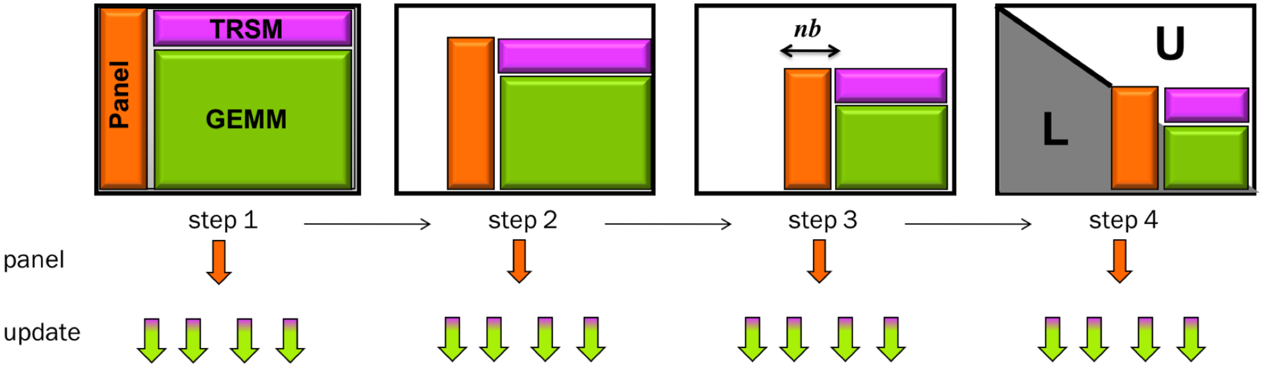
TENSOR CORE ACCELERATED LIBRARIES

Multi-precision numerical methods

Solving linear system of equations *Ax=b*



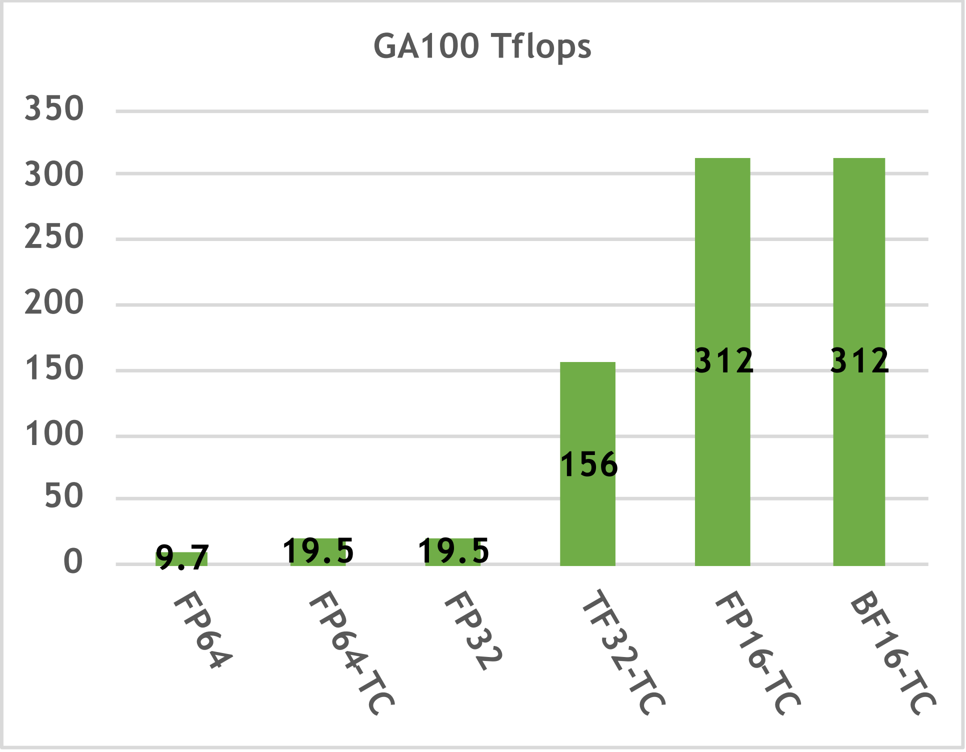
LU factorization used to solve Ax=b is dominated by GEMMs



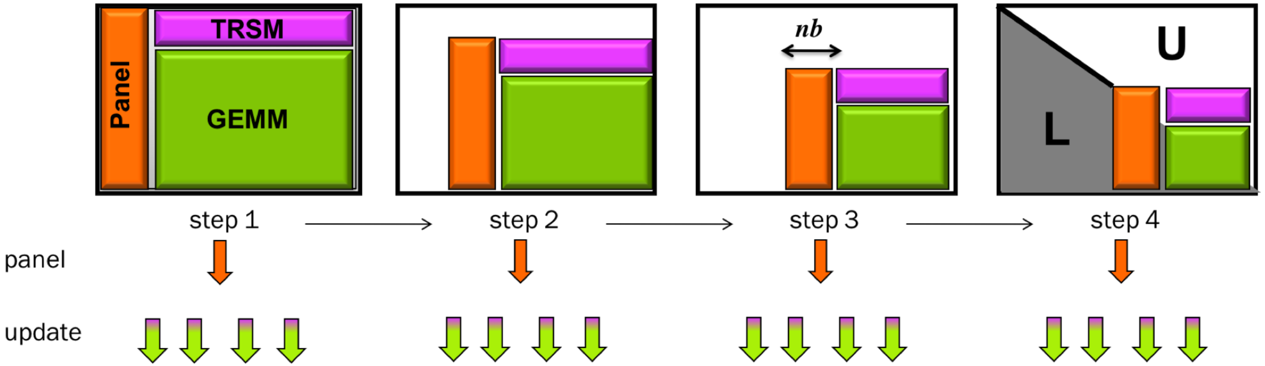
TENSOR CORE ACCELERATED LIBRARIES

Multi-precision numerical methods

Solving linear system of equations *Ax=b*



LU factorization used to solve Ax=b is dominated by GEMMs

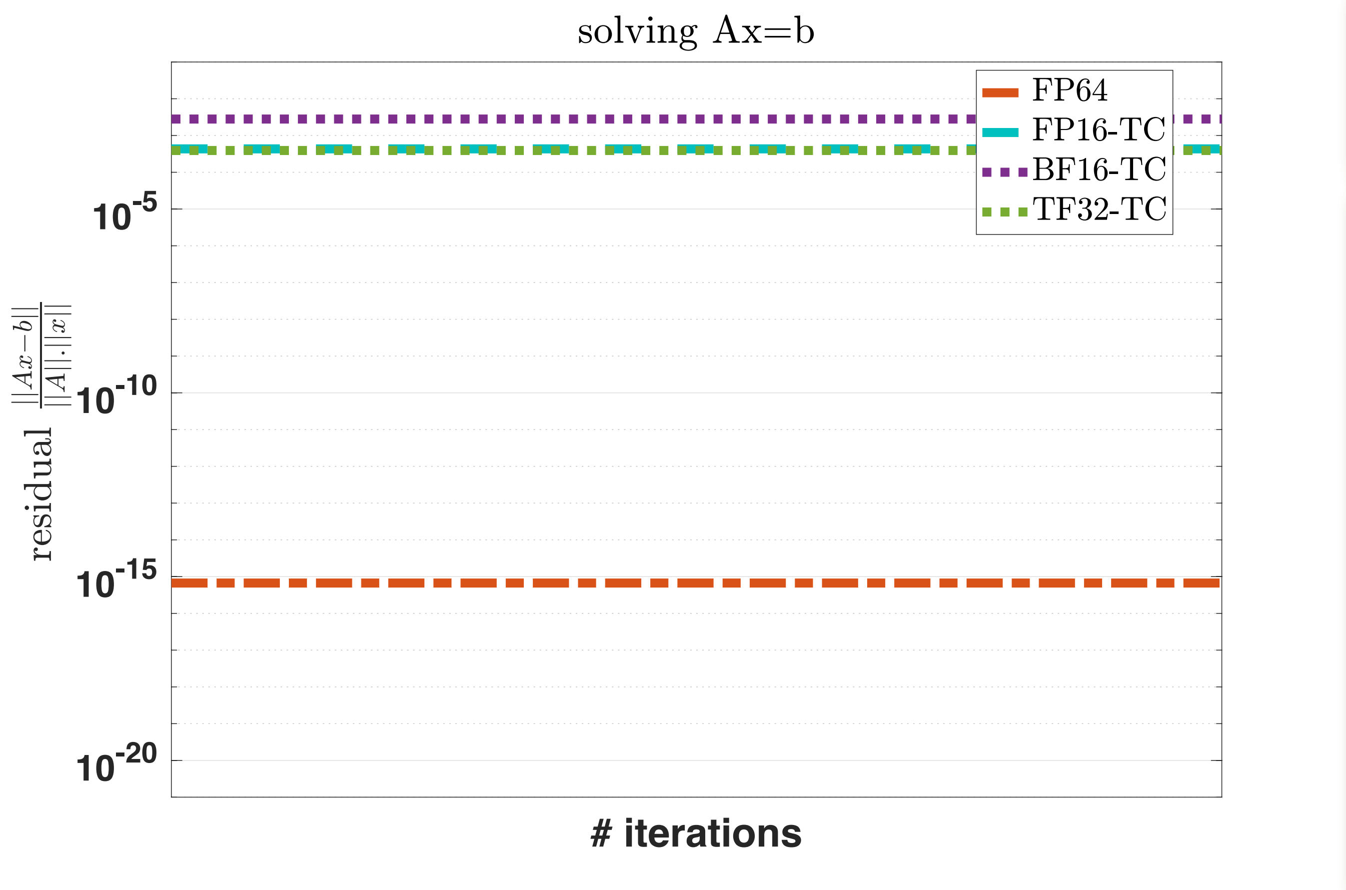


How about a multi-precision LU then ?

Can it be accelerated using Tensor Cores and still get fp64 accuracy?

TENSOR CORE ACCELERATED LIBRARIES

Accuracy just after the reduced precision LU factorization



Accuracy of the obtained solution

- FP64-TC provide a solution down to the FP64 accuracy
- TF32 and FP16 provide a solution to around 1E-05 accuracy
- **Obtained solution has 11 digits loss compared to the FP64 one,**
- **can we do better and achieve the FP64 accuracy?**

Results obtained using CUDA 11.0 and A100 GPU

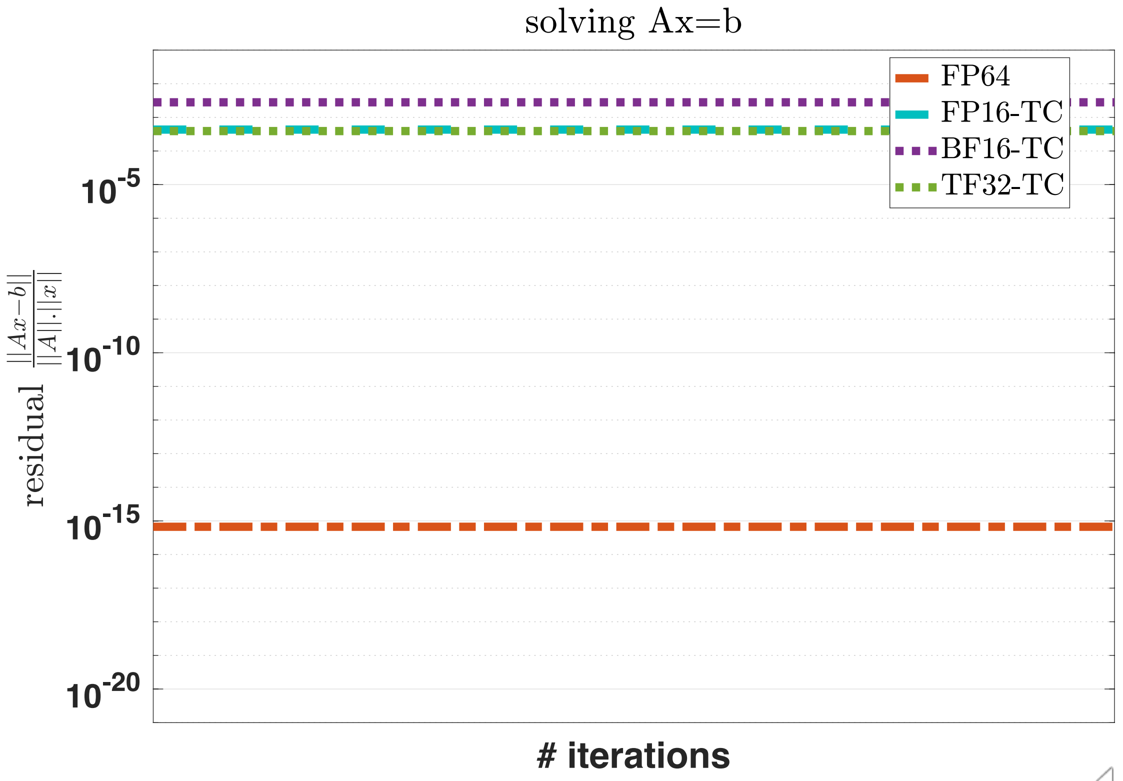
TENSOR CORE ACCELERATED LIBRARIES

How can we get to FP64 accuracy?

Idea: use reduced precision to compute the expensive flops (LU $O(n^3)$) and then iteratively refine the solution ($O(n^2)$) in order to achieve the FP64 level of accuracy

Iterative refinement for solving *Ax* = *b*:

Perform a factorization in reduced precision *A* = *LU*



E. Carson, N. J. Higham, **Accelerating the Solution of Linear Systems by Iterative Refinement in Three Precisions,** SIAM J. Sci. Comput., 40(2), A817-A847.
A. Haidar, S. Tomov, J. Dongarra, and N. J. Higham, **Harnessing GPU Tensor Cores for Fast FP16 Arithmetic to Speed up Mixed-Precision Iterative Refinement Solvers**, SC-18 Dallas, 2018
A.Haidar, H. Bayraktar, S. Tomov, J. Dongarra, N. J. Higham **Mixed-Precision Iterative Refinement using Tensor Cores on GPUs to Accelerate Solution of Linear Systems,** submitted Royal Society Journal UK 2020.

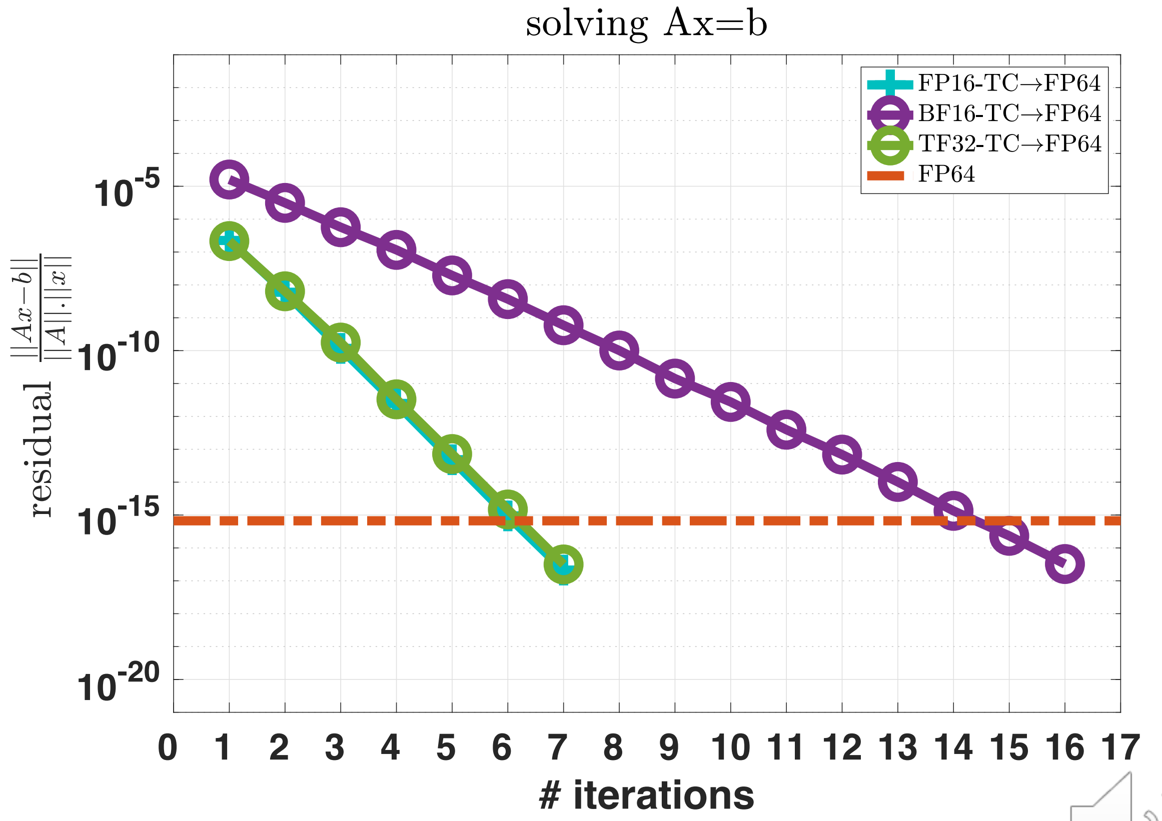
TENSOR CORE ACCELERATED LIBRARIES

How can we get to FP64 accuracy?

Idea: use reduced precision to compute the expensive flops (LU $O(n^3)$) and then iteratively refine the solution ($O(n^2)$) in order to achieve the FP64 level of accuracy

Iterative refinement for solving *Ax* = *b*:

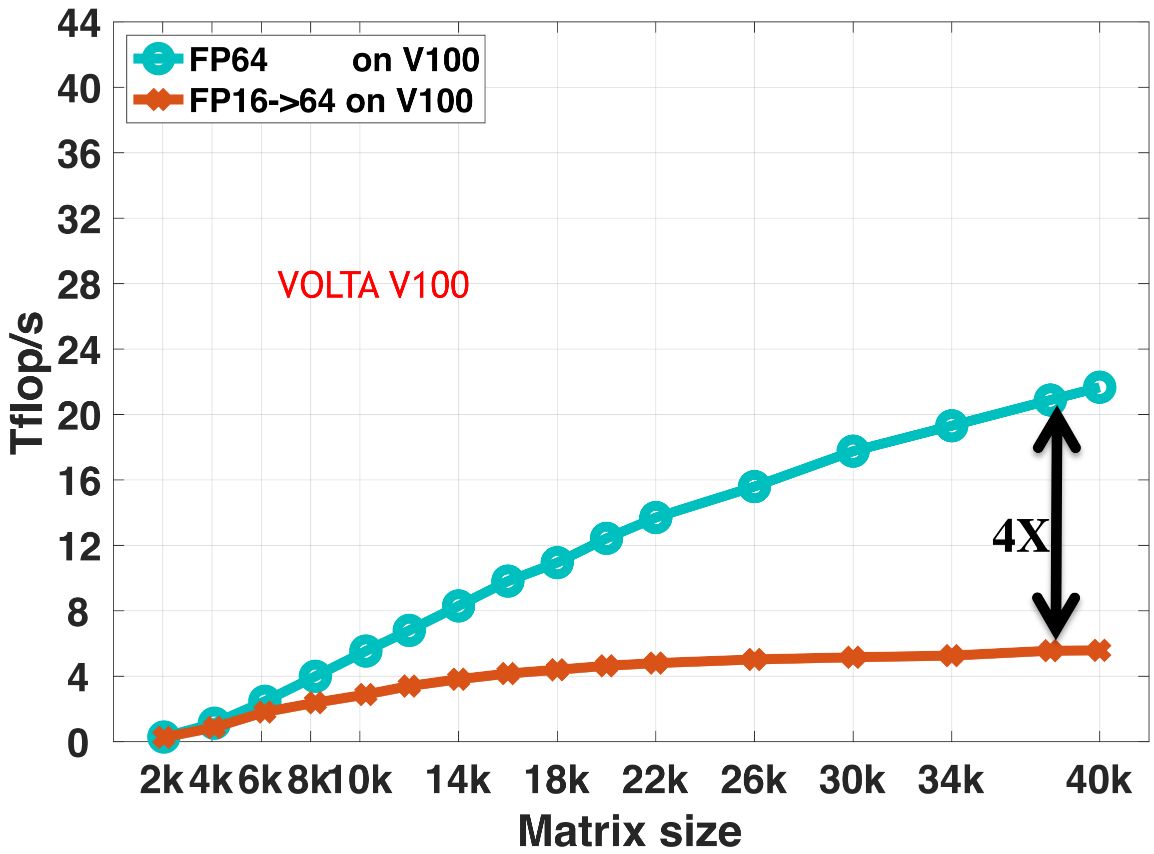
Perform a factorization in reduced precision *A* = *LU*
refine
WHILE || r || > eps_FP64
1. Find correction c such that *Ac* = *r*, *c* = U\(L\r)
2. *x* = *x* + *c*
3. *r* = *b* - *Ax* (with original A).
END



E. Carson, N. J. Higham, **Accelerating the Solution of Linear Systems by Iterative Refinement in Three Precisions,** SIAM J. Sci. Comput., 40(2), A817–A847.
A. Haidar, S. Tomov, J. Dongarra, and N. J. Higham, **Harnessing GPU Tensor Cores for Fast FP16 Arithmetic to Speed up Mixed-Precision Iterative Refinement Solvers**, SC-18 Dallas, 2018
A.Haidar, H. Bayraktar, S. Tomov, J. Dongarra, N. J. Higham **Mixed-Precision Iterative Refinement using Tensor Cores on GPUs to Accelerate Solution of Linear Systems,** submitted Royal Society Journal UK 2020.

TENSOR CORE ACCELERATED LIBRARIES

Performance Behavior, Hilbert matrices, V100



Flops = $2n^3/(3 \text{ time})$
meaning twice higher is twice faster

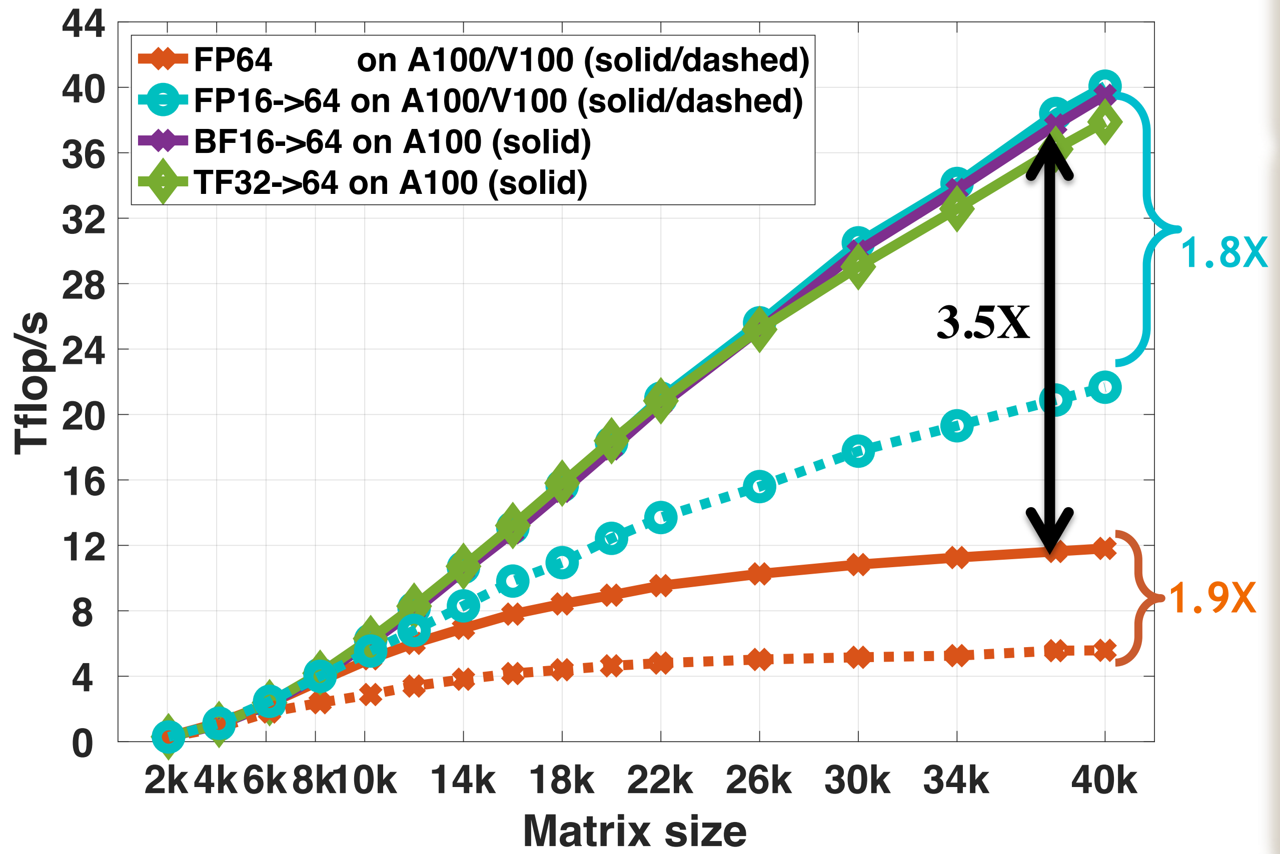
- solving Ax = b using FP64 LU
- solving Ax = b using FP16 Tensor Cores LU and iterative refinement to achieve FP64 accuracy
- FP16 is about 4X faster within a solution to the FP64 accuracy.

Results obtained using CUDA 11.0 and V100 GPU

Problem generated with Hilbert matrices.

TENSOR CORE ACCELERATED ITERATIVE REFINEMENT SOLVER

Performance Behavior, Hilbert matrices, V100 v.s. A100



Flops = $2n^3/(3\ time)$
meaning twice higher is twice faster

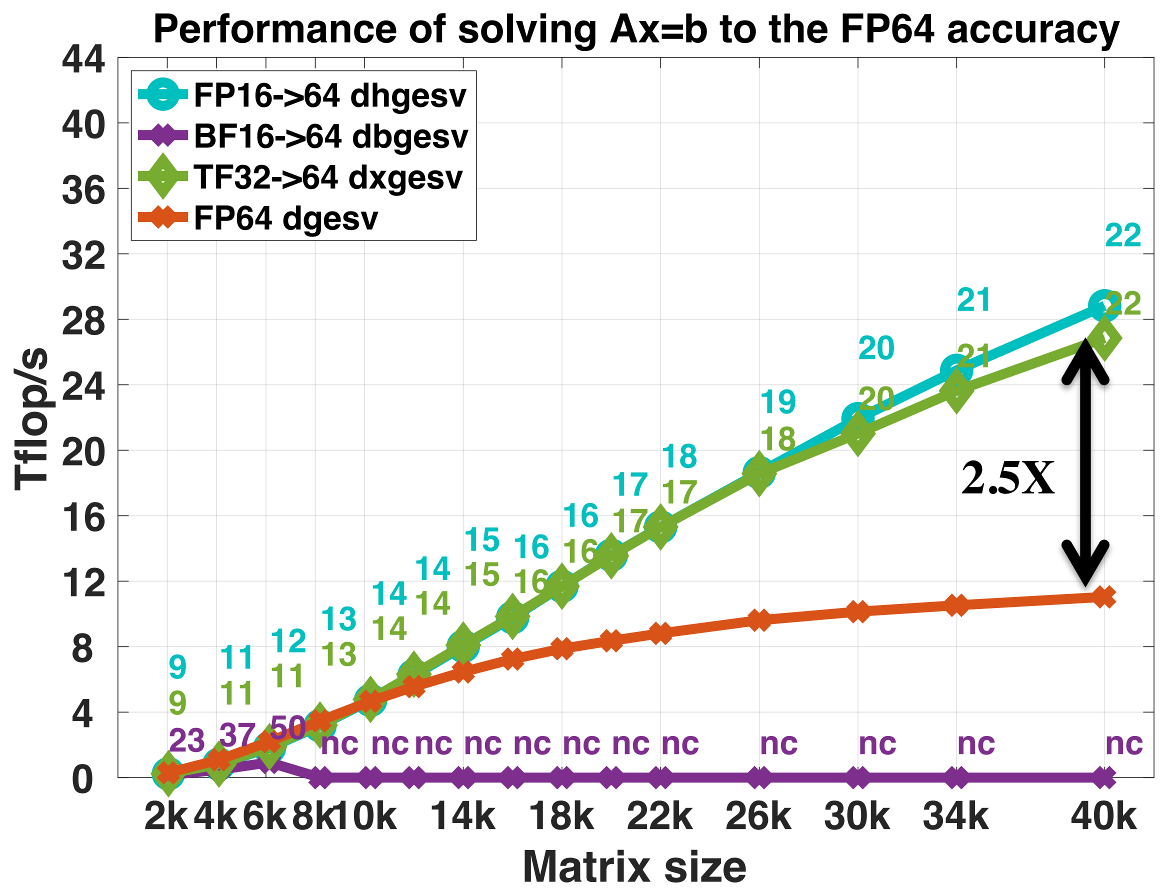
- Speedup compared to FP64 has same trend on both hardware.
- TF32 is 3.3X faster within a solution to the FP64 accuracy.
- FP16 is 3.5X faster within a solution to the FP64 accuracy.
- A100 provides about 1.8X speedup over V100 for both FP16 and FP64 variants

Results obtained using CUDA 11.0 and V100, A100 GPU.

Problem generated with Hilbert matrices.

TENSOR CORE ACCELERATED ITERATIVE REFINEMENT SOLVER

Performance Behavior, matrices with SVD clustered distribution, A100



Flops = $2n^3/(3\ \text{time})$
meaning twice higher is twice faster

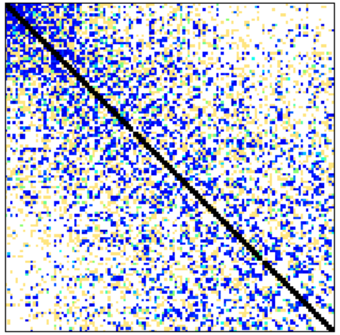
- solving Ax = b using **FP64 LU**
- solving Ax = b using **FP16 Tensor Cores LU** and iterative refinement to achieve FP64 accuracy
- solving Ax = b using **BF16 Tensor Cores LU** and iterative refinement to achieve FP64 accuracy
- solving Ax = b using **TF32 Tensor Cores LU** and iterative refinement to achieve FP64 accuracy

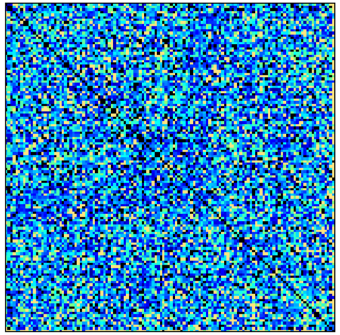
Results obtained using CUDA 11.0 and A100 GPU

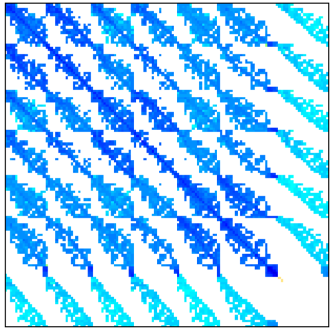
Problem generated with a clustered distribution of the singular values $\sigma = [1, \cdots, 1, \frac{1}{cond}];$

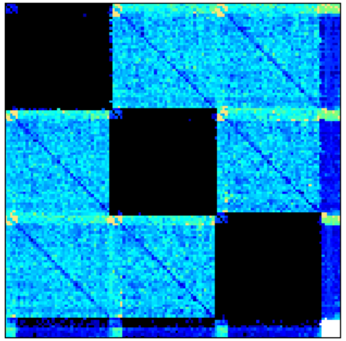
TENSOR CORE ACCELERATED ITERATIVE REFINEMENT SOLVER

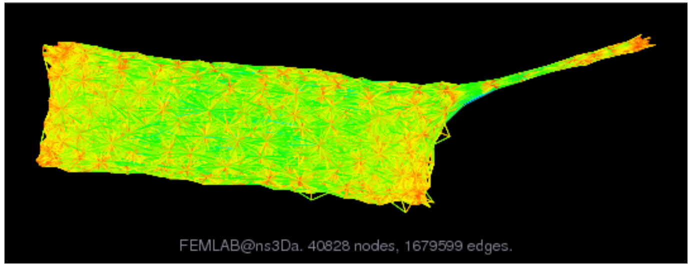
Matrices from SuiteSparse, A100

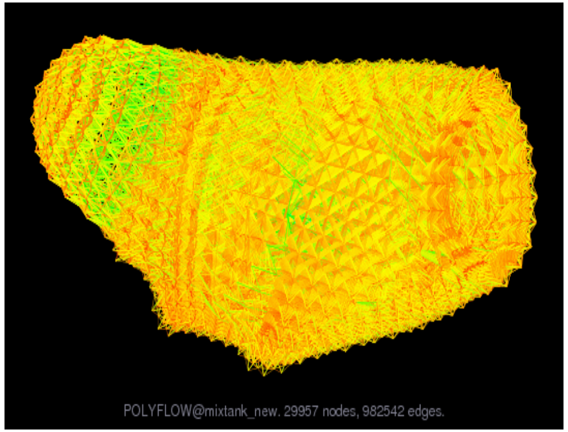


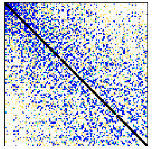
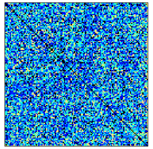






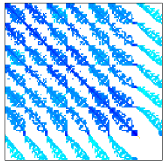
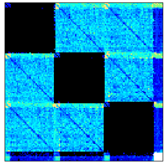




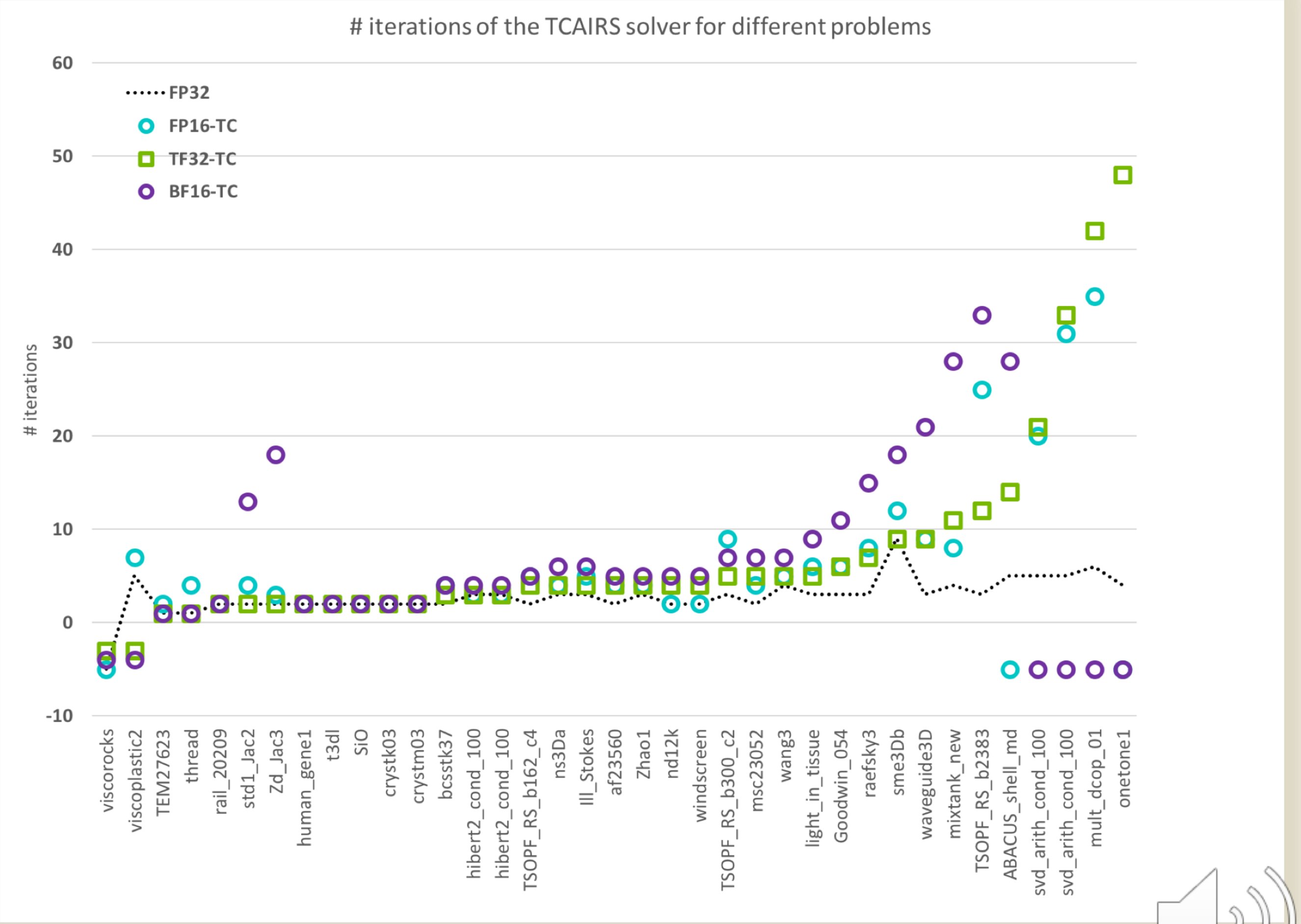
TCAIRS NUMERICAL BEHAVIOR

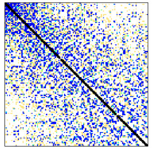
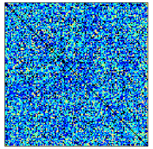
Matrices from SuiteSparse and other problems, A100

- Solving matrices from the SuiteSparse collection corresponding to a wide range of applications in fluid dynamics, structural mechanics, materials science, nuclear energy, oil and gas exploration and others
- TF32 converges faster than both FP16 and BF16 and is able to solve wider range of problems

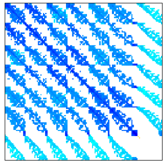
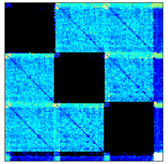
Results obtained using CUDA 11.0 and A100 GPU.



TCAIRS PERFORMANCE BEHAVIOR

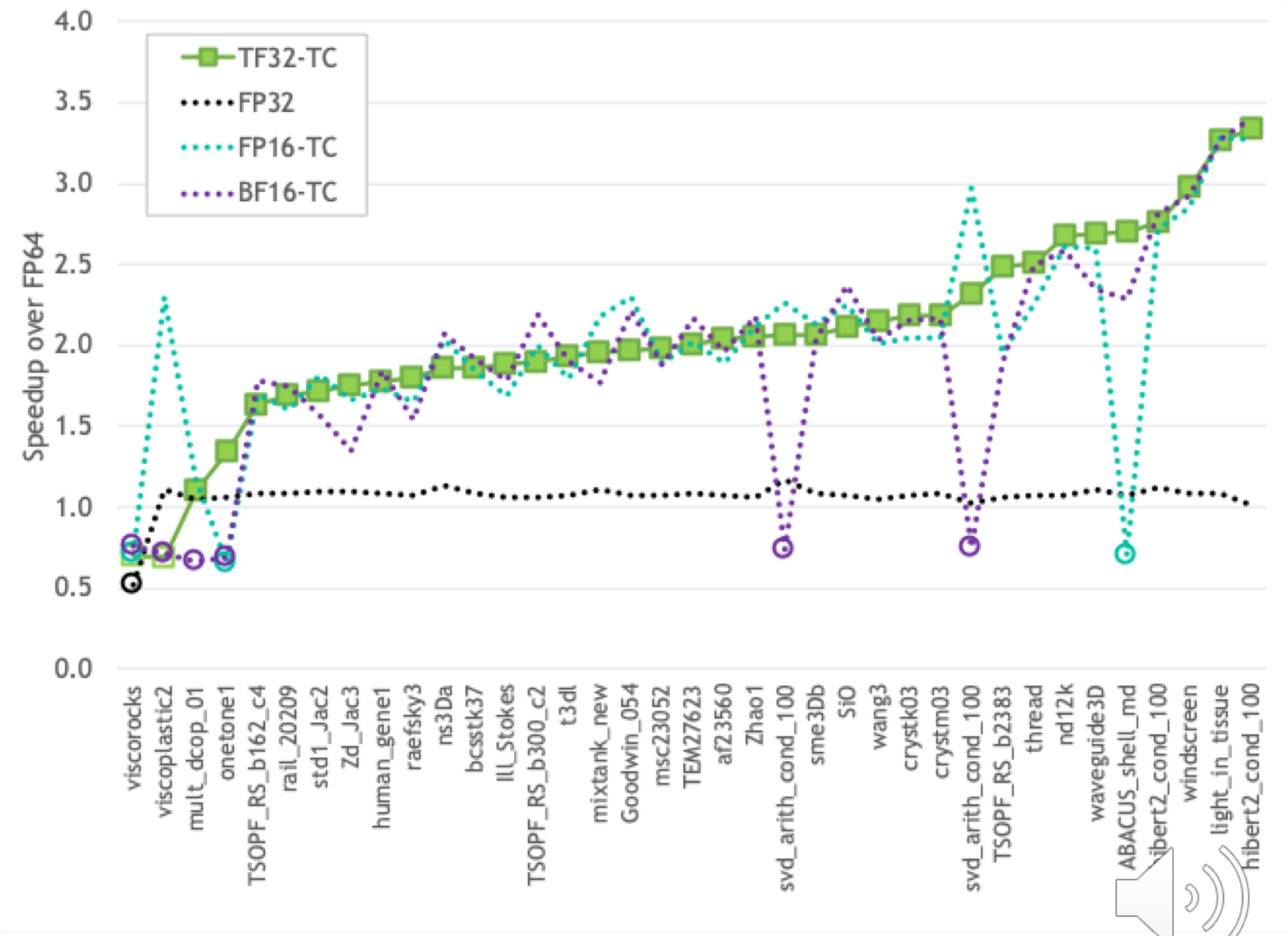
Matrices from SuiteSparse and other problems, A100

	Performance	Fallback cases	Notes
FP32	1x	1	Hard case
TF32	2x	2	Hard case
FP16 scaled	2x	3	Scaling fixes many cases
BF16	2x	6	Loss of precision is an issue for several cases

- TF32 converges faster than both FP16 and BF16 and is able to solve wider range of problems
- In terms of performance TF32 provide time to solution close or better than both BF16 and FP16
- **In summary, TF32 can be considered the most robust and the fastest variant**

Results obtained using CUDA 11.0 and A100 GPU.



TENSOR CORE ACCELERATED ITERATIVE REFINEMENT SOLVER

Tensor Core Accelerated Iterative Refinement Solver (TCAIRS)

January 2019

Magma 2.5.0

TCAIRS-LU real
FP64, FP32, FP16-TC

Nov 2019

cuSOLVER
CUDA 10.2

TCAIRS-LU real
TCAIRS-LU complex
TCAIRS-LU NRHS
FP64, FP32, FP16-TC

May 2020

cuSOLVER
CUDA 11.0

10.2 +
TCAIRS-QR real
TCAIRS-QR complex
TCAIRS-QR NRHS
FP32, FP16-TC
FP64-TC, TF32-TC, BF16-TC
Many other advancements

Mixed Precision Solvers are gaining a lot of attention for their power to provide a solution up to 4X-5X faster and for their energy efficiency.

Thank you very much