Harnessing GPU Tensor Cores for Fast FP16
White Paper-WPL063

Introduction:

In order to test the performance of FP16 compared to FP32 and FP64 in the area of solving linear equations, we experimented with different combinations with precision.

The test was performed in NVIDIA RTX 2070 Super GPU which consists of 320 tensor cores. One of our main targets was to get compare the performance accuracy of mixed precision including fp64 and without including fp64. For that we used mixed precision library available in magma where FP-16 TC (Tensor Core Version) was used together with fp64 arithmetic. They used fp16 for solving Ax=b and fp64 for computing the residual in each step. We replaced fp64 with fp32 and made the comparison between both.

Performance Evaluation:

We experimented with different sizes of matrix for different precision format to determine its performance.

1) DP_solve: here, we used FP64 format for all transient values of iterative refinement.
2) SP_solve: here, we used FP32 precision for all transient values of iterative refinement.
3) MP_Solve: here, we used FP32 to solving and FP64 to compute residual.

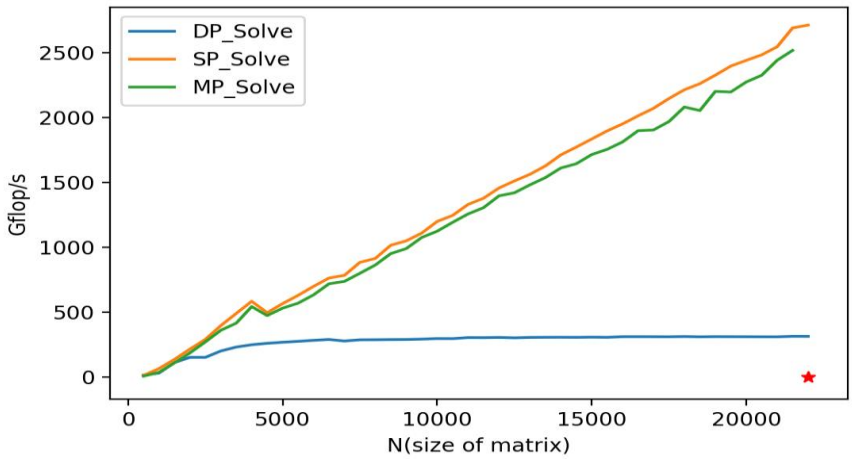

Fig 1. Performance measurement of mixed and single precision

We can see significant improvement in performance for mixed and single precision. We also experimented with half precision. Here we used fp16 + fp32 for mixed precision.

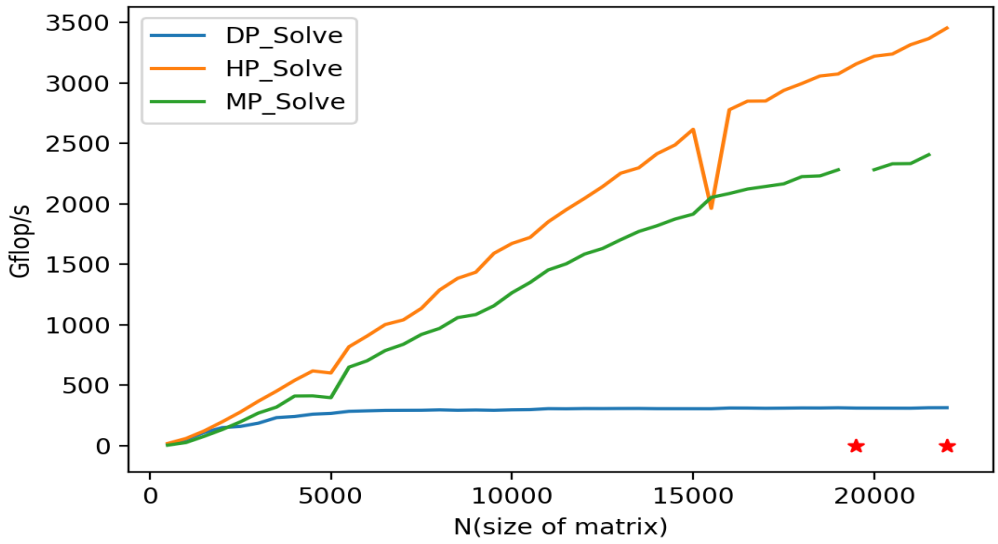

Fig 2. Performance measurement of fp16 and mixed precision

Here, we can observe more notable improvement in performance in fp16 as well as mixed. Also the throughput is faster than the single precision counterpart. But it is also less accurate as we'll see later. We find few cases where the value didn't converge specially with large size of matrix.(denoted by red star).

For mixed precision using fp32 and fp64, we calculated the number of iterations required to meet the tolerance (30 x epsilon) for different size of matrix. And we get the following result.

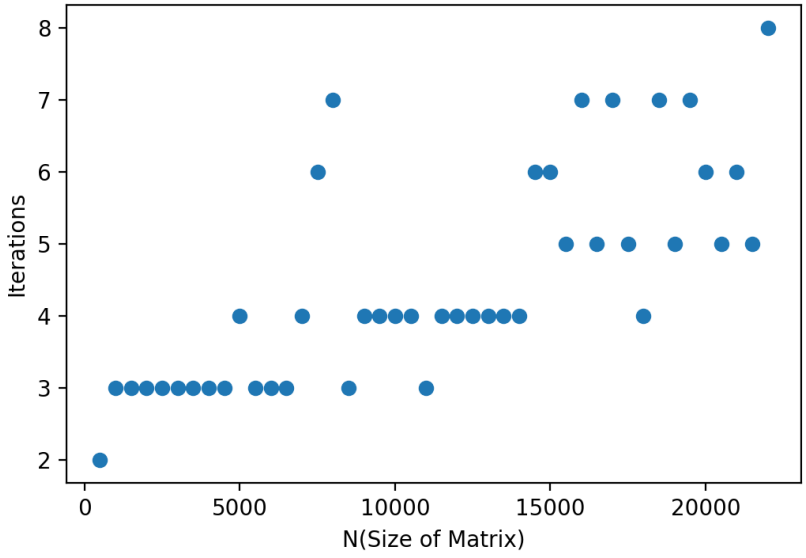

Fig 3. Iterations vs size of matrix representation for fp32 and fp64 mixed precision

We found that as the matrix size increases, the number of iterations required is generally increased but not proportionally. Also the number of iterations required is limited to 8 for n=20000.

Similarly, just using fp16 for factorization and fp32 for residue evaluation, we found the following.

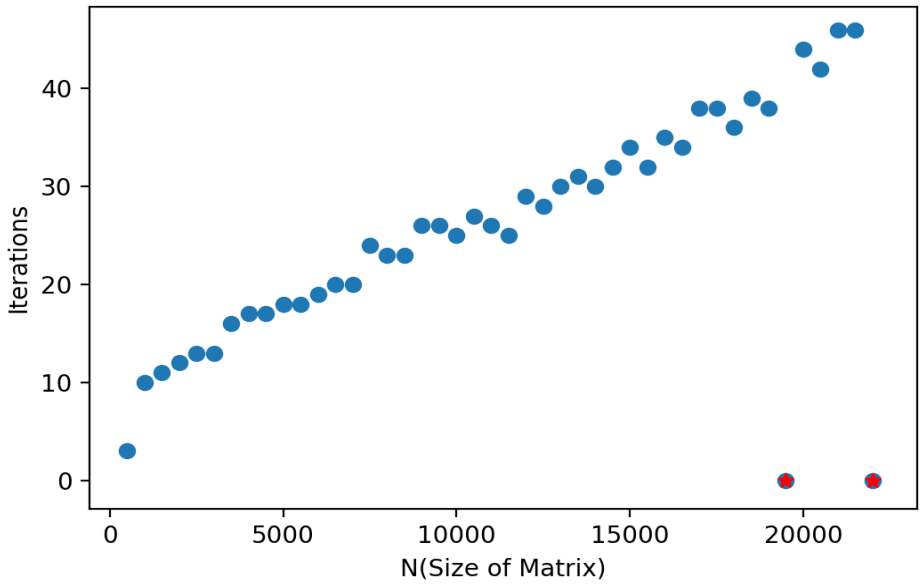

Fig 4. Iterations vs size of matrix of fp16 and fp32

Iterations required is much higher than the previous one and is also susceptible to error as we increase the matrix size (denoted by red star). Here error means to fail to coverage to value below tolerance limit even with very high iterations less than 500.

Accuracy Evaluation:

We also compared the residue by using FP16+FP32 and using mixed (fp16 + fp64).

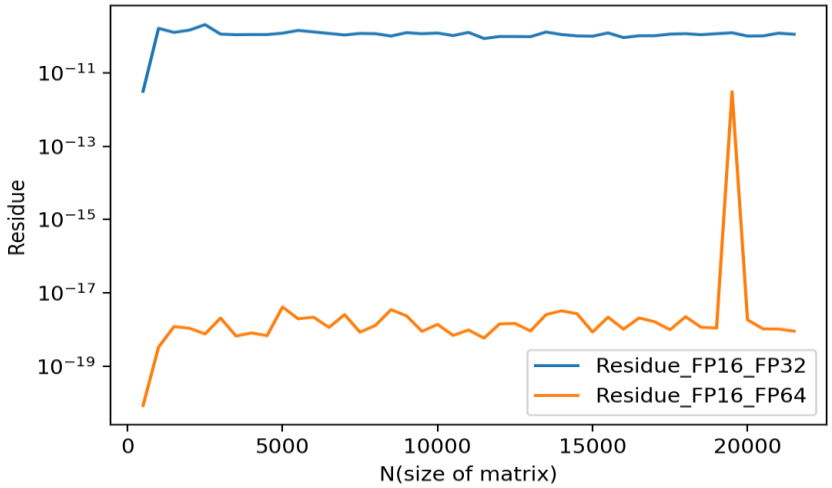

Fig 5. Residue vs size of matrix of FP16+FP32 and FP16+FP64

As we can see using FP16 with FP32 had huge residue leftover. But if we add fp64 precision for residual calculation part, we get much less of it remaining. We can also note that the residue is not dependent upon the size of matrix. Rather it is dependent upon condition number as we'll see.

Finally, We investigated the effect of condition number of a matrix . Condition number basically determines how sensitive the solution of matrix is with small variation in input. During this test we selected matrix with condition number ranging from 100 to 10^13. We find that with increase in condition number , fp16 with fp32 suffers the accuracy drop significantly compared to fp64.

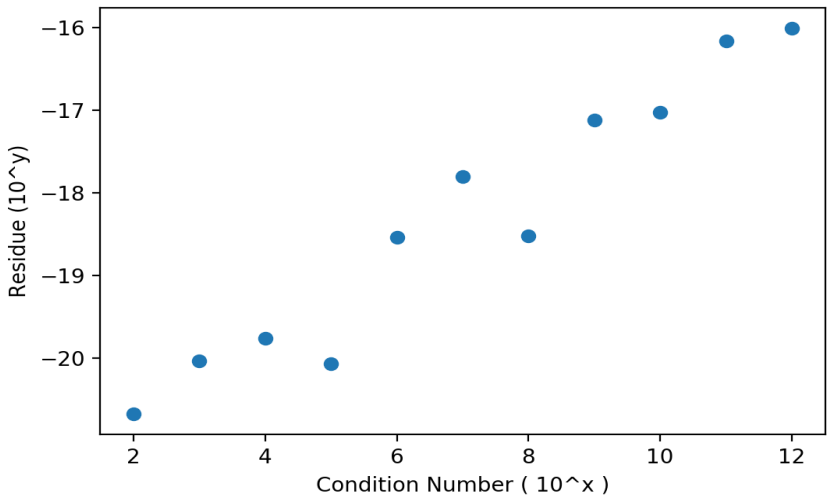

Fig 6. FP64 Condition number vs residue

Here, we can see even at extremely high condition number >10^8, the residue is extremely small which is tolerable in most applications.

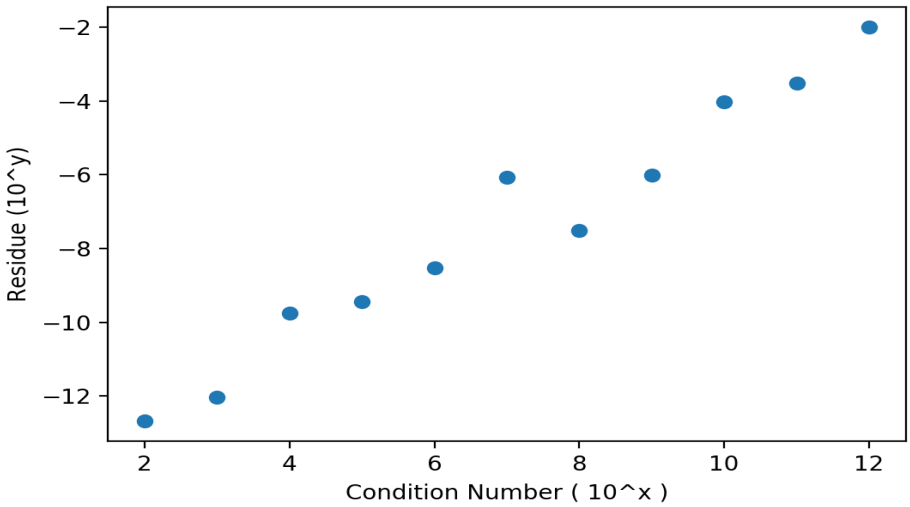

Fig 7. FP16 + FP32 condition number vs Residue

Here, we can see for large condition number, residue is very high which may not be tolerable in some application.
Here is comparison between the two in same figure. Note that each axis is in logarithmic scale.

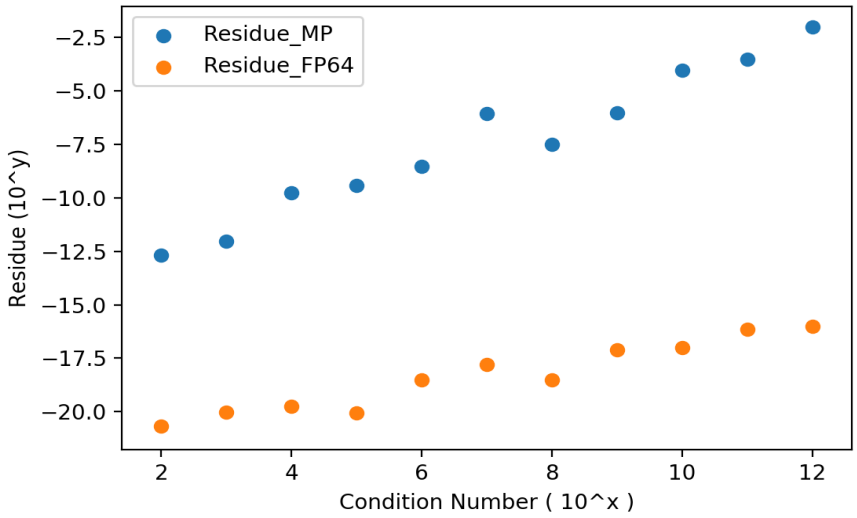

Fig 8. FP64 vs FP16_FP64 residue comparison

Conclusion:

From the experiment, we come to know that the mixed precision, FP16+FP32 performs has notable improvement in performance but significant drop in accuracy even if we increase number of iterations . Morever, FP16+FP32 is very sensitive to condition number than FP64 counterpart. More investigation might be needed to eliminate this problem so that we can totally get rid of heavy fp64 hardware for large range of applications.

References:

[1] . Leveraging the bfloat16 Artificial Intelligence Datatype for Higher-Precision Computations
https://ieeexplore.ieee.org/abstract/document/8877427
[2] . Artificial Neural Network for ordinary differential Equations:
https://arxiv.org/pdf/physics/9705023.pdf
[3] . Artificial Intelligence got its own number system:
https://freenews.live/artificial-intelligence-got-its-own-number-system/
[4] . Bfloat16 number format:
https://en.wikipedia.org/wiki/Bfloat16_floating-point_format
[5] Secret to high Performance on cloud TPUs:
https://cloud.google.com/blog/products/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus
[6] . Universal Approximation Theorem:
https://en.wikipedia.org/wiki/Universal_approximation_theorem
[7]. Neural Network based Approximators for Solving Partial Differential Equations.
https://onlinelibrary.wiley.com/doi/abs/10.1002/cnm.1640100303
[8].Artificial Neural Network for Solving Ordinary and Partial Differential Equations.
https://ieeexplore.ieee.org/abstract/document/712178
[9] . Poisson Equation:
https://en.wikipedia.org/wiki/Poisson%27s_equation
[10] . Diffusion Equation:
https://en.wikipedia.org/wiki/Diffusion_equation
[11] . Lorenz System:
https://en.wikipedia.org/wiki/Lorenz_system
[12] . Solution to first order Volterra Ide:
https://www.hindawi.com/journals/jam/2017/1510267/
[13]. Tensorflow Libraries
https://github.com/tensorflow/tensorflow
[14]. A deep learning Library for Solving Differential Equations
https://arxiv.org/abs/1907.04502, https://github.com/lululxvi/deepxde

Revision History

The following table shows the revision history of this white paper-WPL063.

Date	Version	Details
June 18th, 2021	v1.0	Initial Release

About LogicTronix

LogicTronix provide turnkey Solutions and Intellectual Property (IP) to customers on FPGA Design, Machine Learning Acceleration for various applications including ADAS, Surveillance, Computer Vision, etc.

LogicTronix also offer solutions on "Real Time Traffic Analytics- including ANPR Solution", "Enhancing Financial Trading Algorithms with AI/ML" and "High Frequency Trading (HFT) based Infrastructure".