TSQR on Tensor Cores

Hiroyuki Ootomo
Tokyo Institute of Technology
Tokyo, Japan
ootomo.h@rio.gsic.titech.ac.jp

Rio Yokota
Tokyo Institute of Technology,
AIST-Tokyo Tech RWBC-OIL, AIST
Tokyo, Japan
rioyokota@gsic.titech.ac.jp

ABSTRACT

Tall-Skinny QR (TSQR) is an efficient algorithm for calculating the QR decomposition of $m \times n$ matrices where $m \gg n$ [1], which is done by recursively performing QR decomposition on subdivided blocks of the tall and skinny matrix. Such operations are useful for low-rank approximation methods, which are replacing more and more dense linear algebra in both scientific computing and machine learning fields. The present work focuses on the implementation of this important algorithm on Tensor Cores, which are available on the latest NVIDIA GPUs. We evaluate the speed, accuracy, and stability of TSQR on TensorCores.

CCS CONCEPTS

• **Computing methodologies** → **Massively parallel algorithms**.

KEYWORDS

QR decomposition, TSQR, GPU, Tensor Core, half-precision

ACM Reference Format:
Hiroyuki Ootomo and Rio Yokota. 2018. TSQR on Tensor Cores. In *SC '19: The International Conference for High Performance Computing, Networking, Storage, and Analysis, November 17–22, 2019, Denver, CO.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

1.1 TSQR

The Tall-Skinny QR (TSQR) computes the QR decomposition of a tall and skinny matrix by subdividing it into shorter submatrices, performing the QR decomposition on them, then concatenating the R matrices and performing QR decomposition on them again until it ends up with a single R matrix, as shown in Figure 1. The QR decomposition of the submatrices can be performed concurrently, so batched QR algorithms on GPUs can be used to efficiently compute them.

1.2 Tensor Cores

Starting from the Volta architecture, NVIDIA GPUs come with Tensor Cores, which can perform the following operation

$$\mathbf{D} \leftarrow \mathbf{A} \times \mathbf{B} + \mathbf{C}, \tag{1}$$

SC '19, November 17–22, 2019, Denver, CO

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

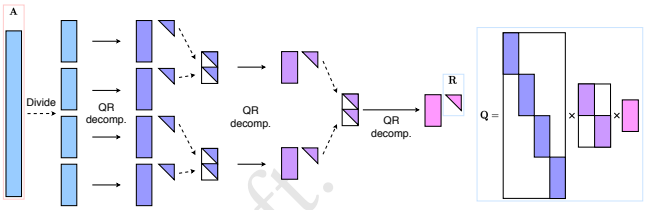

Figure 1: Schematic of the TSQR computation

where $\mathbf{A}, \mathbf{B} \in \mathrm{FP16}^{4\times4}$ are 4×4 matrices with 16 bit (half precision) floating point elements, while $\mathbf{C}, \mathbf{D} \in \mathrm{FP16/FP32}^{4\times4}$ are 4×4 matrices with 16 or 32 bit floating point elements. This operations is performed by 1 Warp (32 Threads). When the accumulation is done in single precision, it can reduce the loss of significance during the accumulation, and thus provide an accuracy that is higher than 16 bit, even though the matrix multiplication is done in 16 bit.

2 IMPLEMENTATION

We use the Householder transformation for each QR decomposition, which requires matrix multiplications as part of the algorithm. We perform these matrix multiplications on the Tensor Core. There are a few different possibilities of where half precision and Tensor Cores can be used and we test for four different combinations, as shown in Table 1. In order to simplify the implementation we limit the number of columns in the TSQR to be less than 16. The number of rows have no such restriction.

3 EXPERIMENTS

3.1 Setup

We generate a $m \times 16$ random matrix $\mathbf{A}$ and compute the TSQR on it. The relative residual norm, orthogonality, and timings are compared with the QR decomposition in NVIDIA's cuSOLVER. Computations are performed on a single NVIDIA V100 GPU.

Table 1: Frequency of Special Characters

Abbreviation	Precision	Use Tensor Core
TSQR-FP32-TC	float	yes
TSQR-FP16-TC	half	yes
TSQR-FP32	float	no
TSQR-FP16	half	no

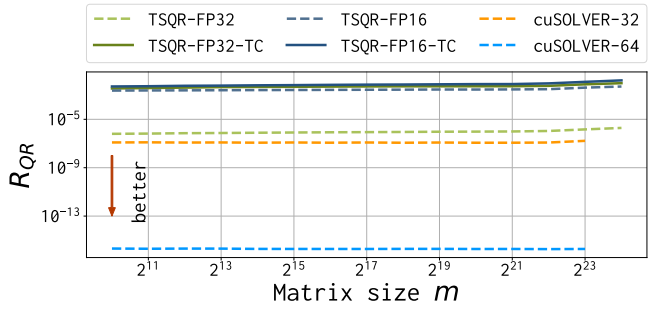

Figure 2: Residual error of the TSQR computation

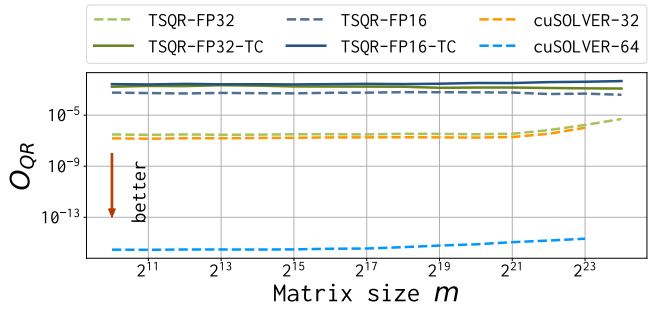

Figure 3: Orthogonality of the TSQR computation

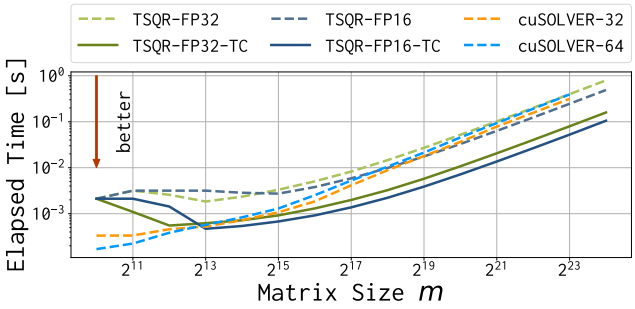

Figure 4: Computation time of TSQR

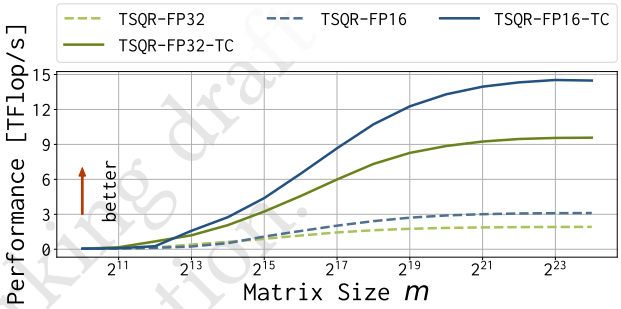

Figure 5: GPU utilization of TSQR

3.2 Accuracy

We evaluate the accuracy of our TSQR on Tensor Core by Calculating the relative residual norm

$$\frac{\|\mathbf{QR} - \mathbf{A}\|_F}{\|\mathbf{A}\|_F} \tag{2}$$

where $\|\cdot\|_F$ is the Frobenius norm. This relative residual norm is plotted against the matrix height m in Figure 2. It can be seen from these results that the accumulation in 32 bit does not necessarily improve the error of the TSQR, which is predominantly cause by the truncation to 16 bit even before the multiplication takes place. Shifting the range of the values through normalization could possibly improve the Tensor Core results, but we did not implement such techniques in the present work. When single precision (FP32) is used for the entire calculation the relative residual norm is around 10^{-6}. However, even when single precision is used for the other parts of the algorithm, the use of Tensor Cores forces the matrix multiplication to happen in half precision (FP16). Therefore, FP32-TC has the same accuracy as the FP16 cases.

3.3 Orthogonality

The orthogonality of the Q matrix is checked by computing

$$\frac{\left\|\mathbf{I} - \mathbf{Q}^T\mathbf{Q}\right\|_F}{\sqrt{16}} \tag{3}$$

The orthogonality is shown for different matrix heights m in Figure 3. The results are similar to the residual, where only FP32 has high accuracy, and any use of FP16 or Tensor Cores decreases the accuracy below 10^{-3}.

3.4 Computation Time

The computation time is plotted against the matrix size in Figure 4. We can see that our baseline implementation TSQR-FP32 is slower than the cuSOLVER QR for small matrices, but is competitive for large matrices. When we use Tensor Cores, we are about an order of magnitude faster than cuSOLVER.

3.5 GPU Utilization

The number of Flop/s is shown for different matrix sizes in Figure 5. From this Figure we can see that the use of half precision and Tensor Cores allows the TSQR to achieve close to 15 TFlop/s when the matrix is tall enough.

4 CONCLUSION

We implemented the TSQR algorithm using Tensor Cores and showed that we are able to outperform cuSOLVER by an order of magnitude by sacrificing accuracy.

ACKNOWLEDGMENTS

This work was partially supported by JST CREST JPMJCR19F5 and JSPS Kakenhi JP18H03248.

REFERENCES

[1] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou. 2008. *Communication-avoiding Parallel and Sequential QR Factorizations.* Technical Report EECS-2008-74. UC Berkeley. http://bebop.cs.berkeley.edu/pubs/mhoemmen2008-tsqr-tech-report.pdf