CATEGORY: **FINANCE - FI01**

POSTER **P5293**

CONTACT NAME
Easwar Subramanian: easwar@atc.tcs.com

GPU TECHNOLOGY CONFERENCE

Accelerating PCA for applications in finance using cuBLAS

Anubhav Jain, Easwar Subramanian and Amit Kalele
Tata Consultancy Services, India
anubhav.jain1@tcs.com, easwar@atc.tcs.com, kalele.amit@tcs.com

ABSTRACT

In this work, we provide a way to accelerate the computation of principal components of large correlation matrices. The input correlation matrix is formed from the historical prices of the financial assets present in a client portfolio. Given a correlation matrix, a partial Eigen spectrum containing the leading Eigen values and Eigen vectors of the correlation matrix are found using the DSYEVR routine of the LAPACK package. We note that in finding the partial Eigen spectrum of a large correlation matrix, the main computational bottleneck is the tri-diagnolization of the input correlation matrix. To this end, we perfrom tri-diagnolization of the correlation matrix completely on the GP-GPU. Our tri-diagnolization algorithm is based on the DSYTRD routine of the LAPACK package and uses several CuBLAS routines. In addition, we propose a multi-stream framework to handle multiple PCA requests concurrently.

INTRODUCTION

Principal component analysis (PCA)

1. A mathematical procedure to extract the important components of data.
2. The important step in a PCA calculation is to compute the partial Eigen spectrum of the input matrix.
3. Wide applications in various fields ranging from computer vision, neuroscience, social science and computational finance.
4. Specifically, in the field of computational finance, PCA has relevance in portfolio replication, statistical arbitrage, yield curve analysis, haircut computation and formulation of trading strategies.
5. In application such as haircut, PCA computations may have to be performed very frequently all through the day and big investment banks have a many client portfolios which mandates simultaneous computation of PCA for all such portfolios.

Need for High performance computing in PCA

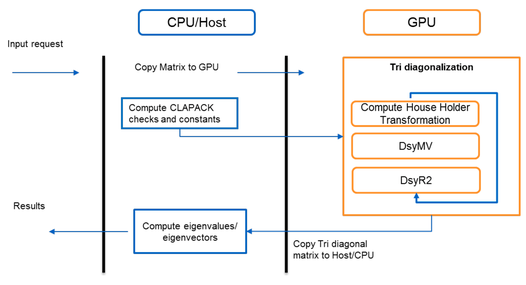
1. For large input matrices (for sizes > 2000)
2. For large number of matrices or PCA requests.

In computing the partial Eigen spectrum of the input matrix, the compute intensive step is mainly the tri-diagnolization of the input matrix. To this end, we propose a solution that involves porting the tri-diagnolization operation to GP-GPU. To handle multiple simultaneous requests we use the multi-stream framework.

PCA ALGORITHM & DESIGN

Main components of Principal Component Analysis

1. Client – Server based system for handling multiple PCA requests
2. Multi Stream – Multi Threaded application for concurrent request processing
3. Fast Eigenvalue and Eigenvector computation module



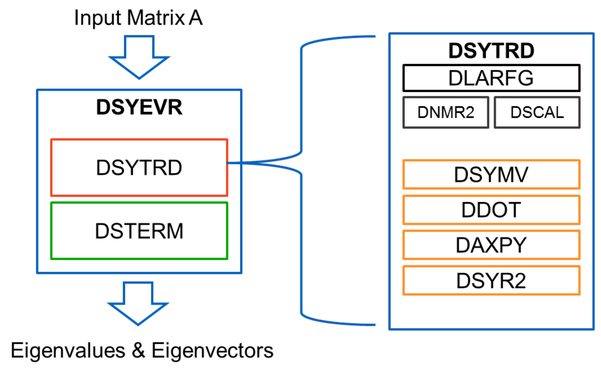
CLAPACK's DSYEVR algorithm was used for computing Eigenvalues and Eigenvectors

- Tri-diagonalization done using Householder transformation

DSYEVR ALGORITHM

```
for j = 0 to n-1

• Compute v^j the Householder Reflectors DLARFG(A, j)
• Compute vector p^j = DsyMV(A, v^j)
• Compute w^j = p^j – (1/2)*(p^jT * v^j) v^j
• Update A = A – v^j w^jT – w^j v^jT = Dsyr2(A, v, w)
• Compute eigenvalues and eigenvectors DSTERM()
```

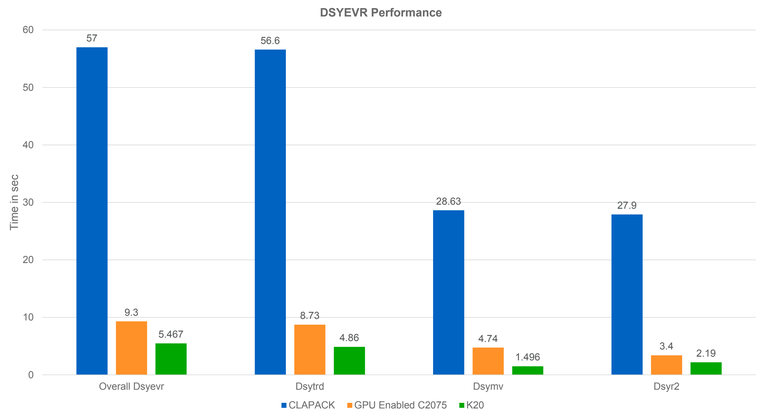


GPU IMPLEMENTATION

- Compute time for the 4500 size matrix was ~60 seconds on CPU
- Over 95% of time was spent in the tri-diagonalization routine DSYTRD
- DSYMV and DSYR2 together contributes 99% of the time for DSYTRD
- Our approach – offload DSYTRD on GPU
- DSYTRD function was ported to GPU using cuBLAS library
- Following cuBLAS functions were used
  - DNRM2, DSCAL, DSYMV, DDOT, DAXPY & DSYR2

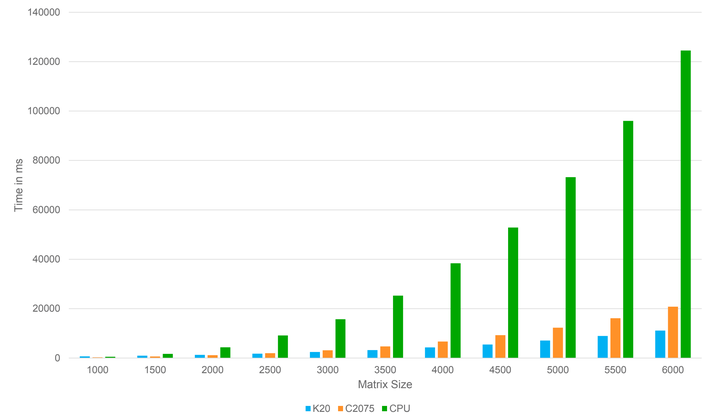
Performance optimization

- cuBLAS: Optimal performance library was used for all BLAS routines
- Minimizing data transfers between host and device: Several other functions were also ported to GPU to avoid frequent data transfers between host and device
- Multi-stream implementation for concurrent processing: Several simultaneous compute requests need to be processed
  - Each request is processed in a separate thread which is assigned to a unique CUDA stream



DSYEVR Performance: CPU Vs C2075 Vs K20

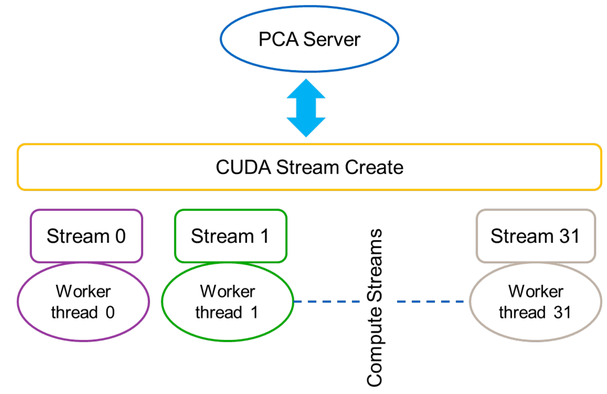
DSYTRD PERFORMANCE



DSYEVR Performance for Varying Matrix sizes

MUTLI REQUEST PCA SERVER WITH HYPERQ

- Multi threaded multi stream Implementation to handle simultaneous compute requests
- A pool of 32 threads and streams were created
- Each request is processed in a new thread from the pool
- Each thread is attached with a unique stream
- Streams perform better with HyperQ on Kepler



Approximately 4000 requests of varying matrix size were processed in 104 seconds on Kepler K20 GPU in 32 streams

CONCLUSIONS

1. **Achieved over 11x speed up for DSYTRD computation for larger matrices**
2. **Achieved 5x speed up for batch jobs ranging from size 3 to 6000**

REFERENCES

1. J. J. Dongarra, D. C. Sorenssen and S. J. Hammarling, Block reduction of matrices to condensed forms for Eigenvalue decompositions, Journal of computational and applied mathematics, 215-227, 1989.
2. I. Yamazaki,T. Dong, R. Solcà, S. Tomov, J. J. Dongarra and T. Schulthess, Tridiagonalization of a dense symmetric matrix on multiple GPUs and its application to symmetric eigenvalue problems, Concurrency and computation : Practice and Experience, 2013.
3. I. Dhillon, A new O(n^2) algorithm for the symmetric tridiagonal eigenvalue/eigenvector problem, Computer Science Division, Ph.D thesis, UC Berkeley, 1997.