Randomization Algorithm to Compute Low-Rank Approximation

Student: Ru HAN (The Chinese University of Hong Kong)
Mentors: Ed D'Azevedo (ORNL), Ichitaro Yamazaki(UTK)

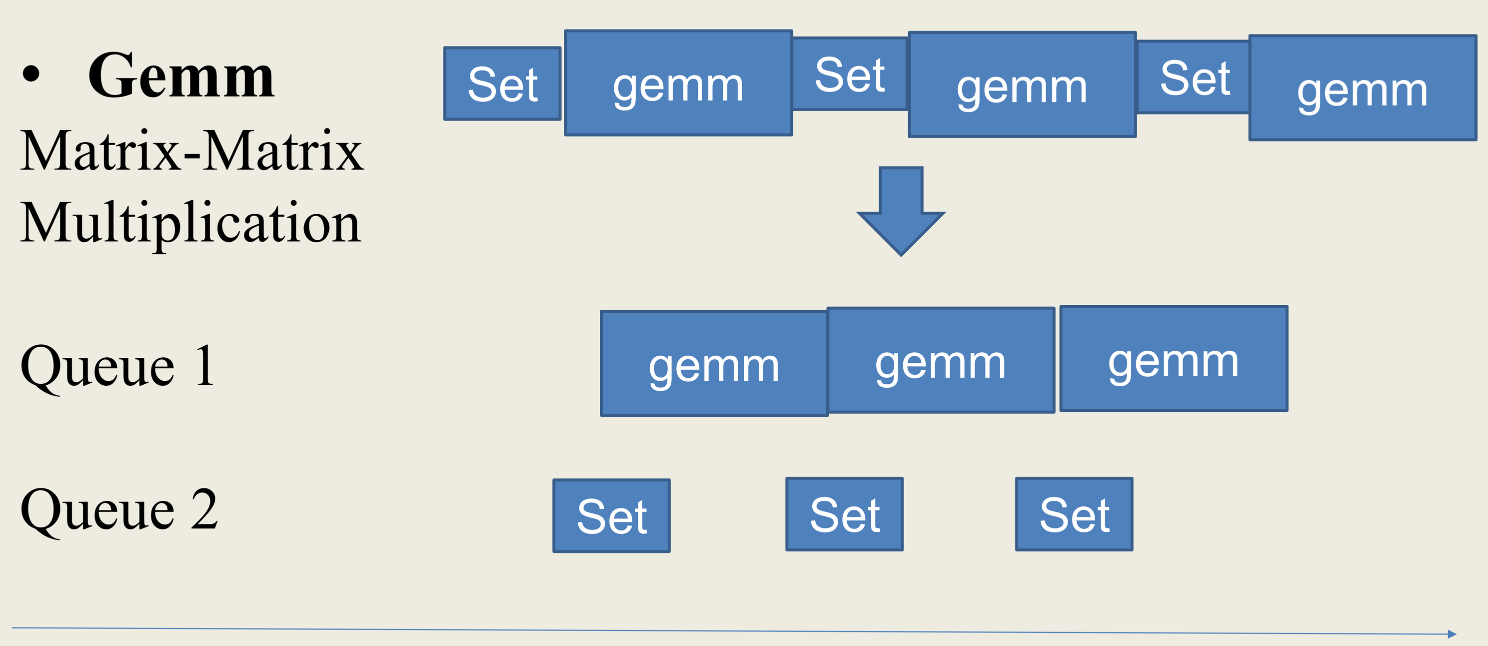
Abstract

A low-rank representation of a matrix provides a powerful tool for analyzing the data represented by the matrix.
In this project, we implement "randomized" algorithm to compute the low-rank representation in the LAPCK/MAGMA/cuBLAS-XT software framework.

Optimization

- **Gemm**

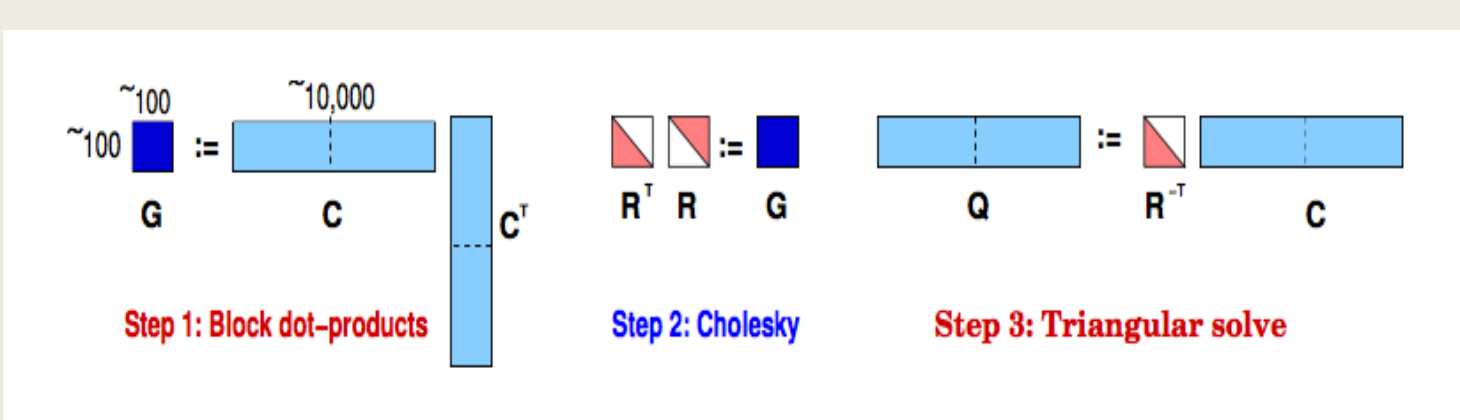
Matrix-Matrix Multiplication



- **QR**

Use CholQR instead of ordinary QR to compute the QR factorization of a matrix B in the following three steps :

- Form a Gram matrix G; i.e., $G = BB^T$.
- Compute the Cholesky-factor R of the Gram matrix G; i.e., $R^TR = G$, where R is upper-triangular with non-negative diagonals.
- Compute the orthogonal matrix Q by the backward-substitutions; i.e., $Q = R^{-T} B$.



Future Work

- Sampling and updating the out-of-core Matrix.
- Applications of including Latent Semantic Indexing (LSI), genetic clustering, subspace tracking, and image processing.

Randomized SVD Algorithm

```
q = randn(n,k+l);
[q,r] = qr(q,0);
   for iter=1:(max_iters-1)
      p = A*q;
      q = A'*p;
      [q,r] = qr(q,0);
   end
p = A*q;
   [p,b] = qr(p,0);
end
[x,s,y] = svd(b);
u_k = p*x(:,1:k);
s = s(1:k,1:k);
v_k= q*y(:,1:k);
```

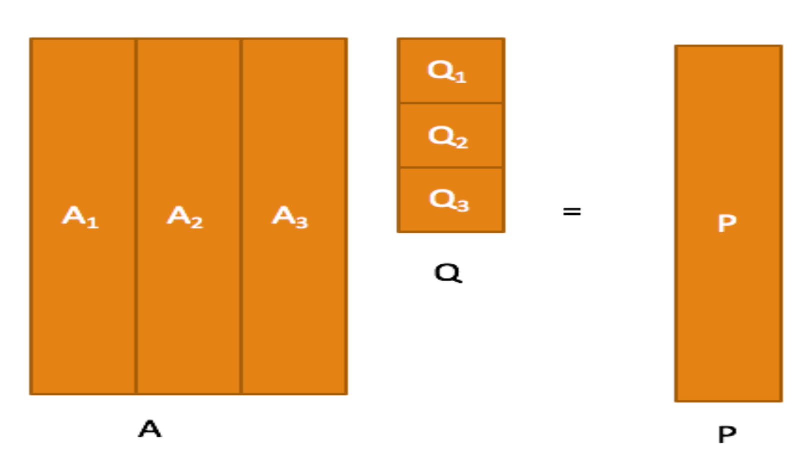
Error $= \|A-U_K S_K V_K^T\|_2$
$= (k+1)_{th}$ largest singular value of A

Out-of –core Randomized SVD

Method 1. Manual pipeling

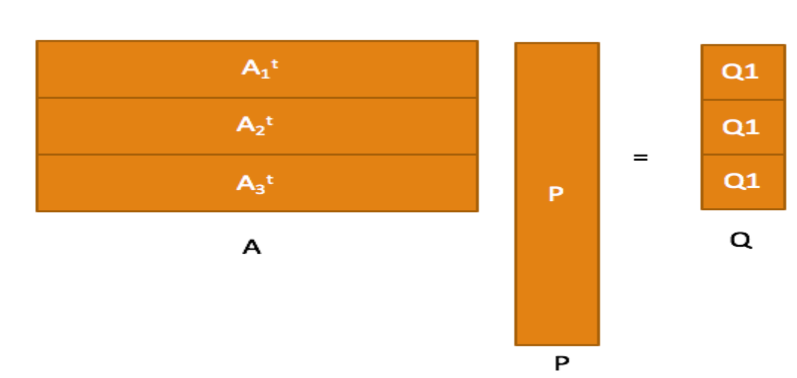
- **P=A*Q**

```
P=0;
for  k=1,2,3…..
        set (A_k to dA);
         P=P+A_kQ_k;
end
```



- **$Q=A^T*P$**

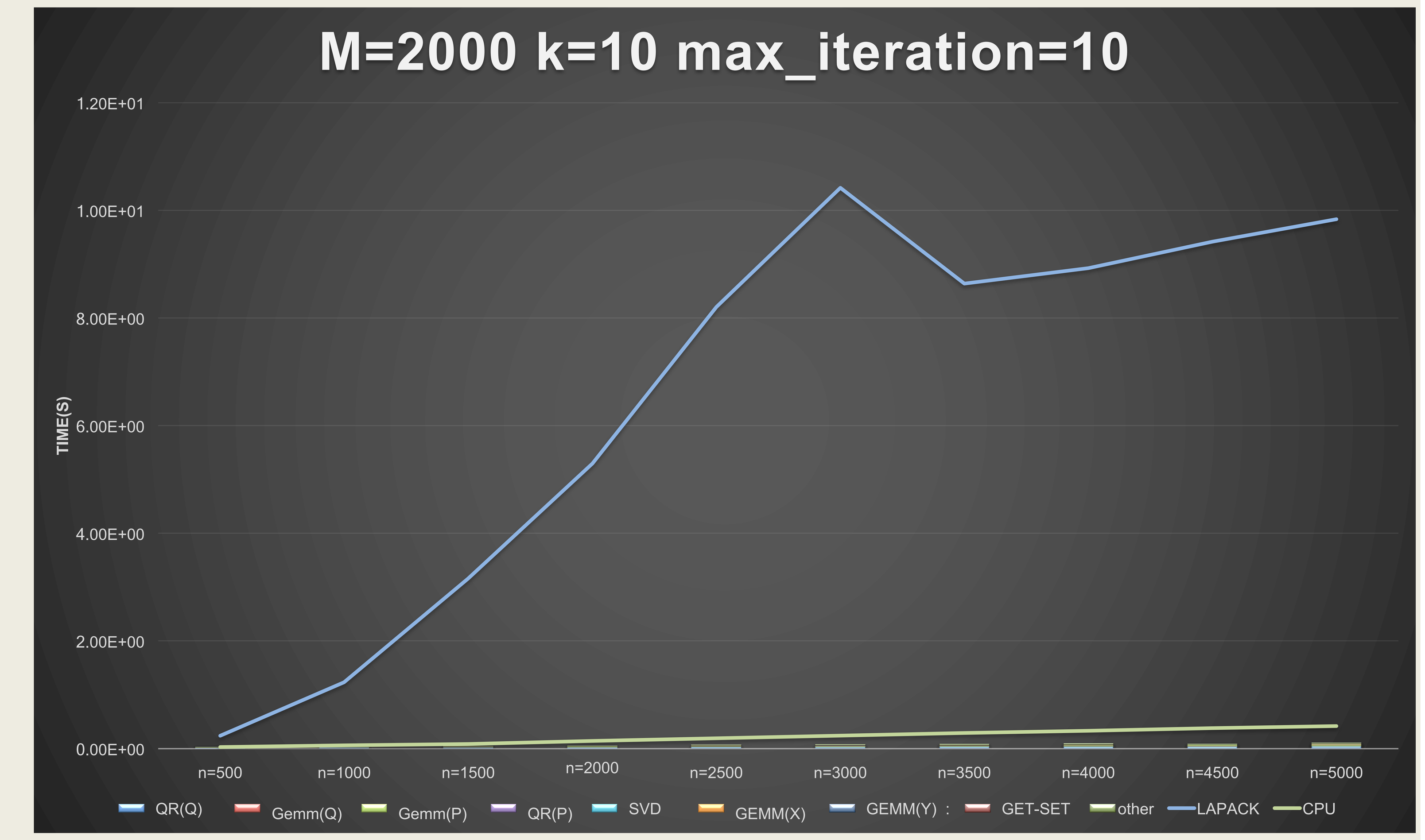
```
for  k=1,2,3…..
        set (A_k to dA);
         Q_k=A_k^T P;
end
```

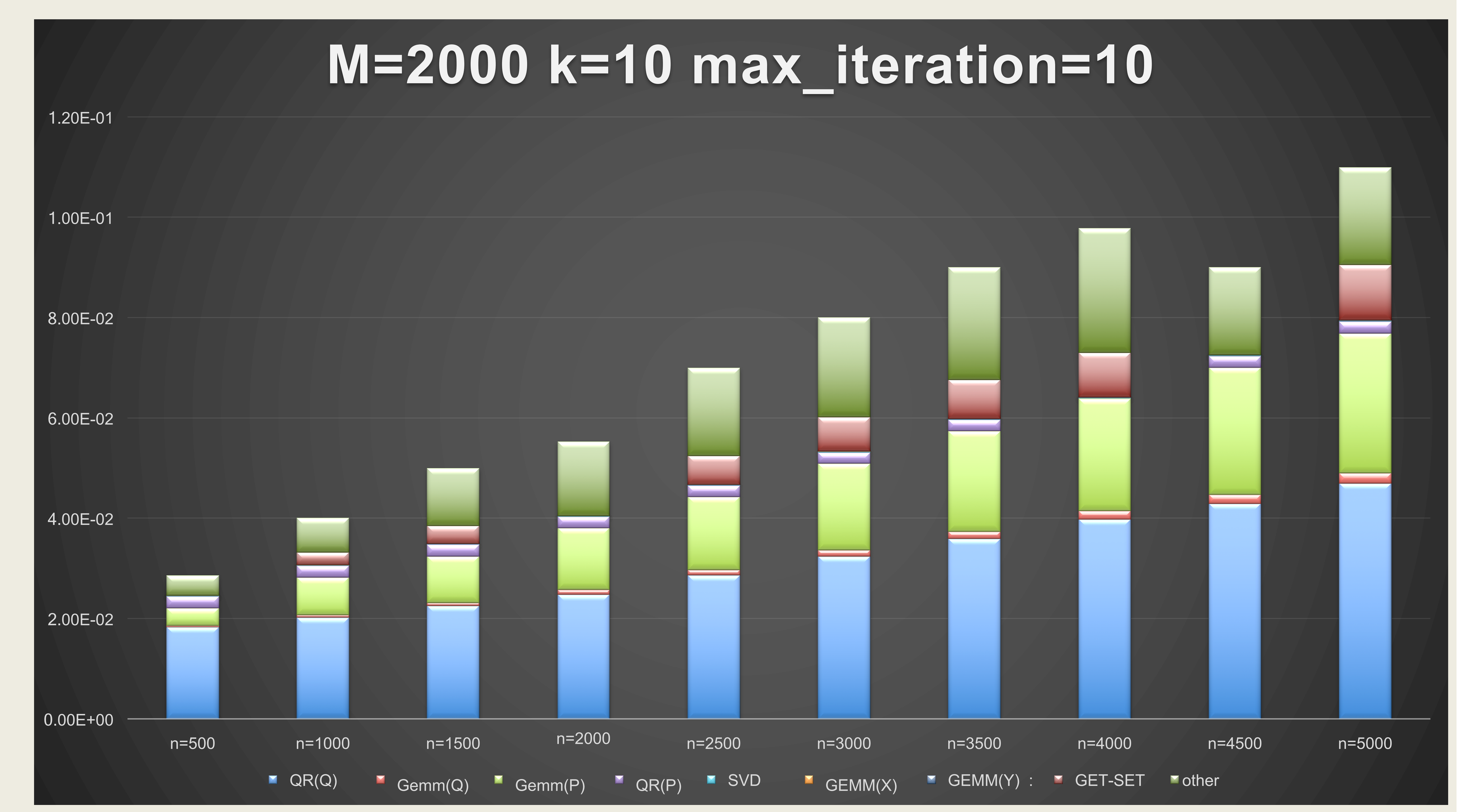


Mehod 2. UMA&CUBLAS-XT

UMA is a programming model, Unified Memory Access. Unified Memory creates a pool of managed memory that is shared between the CPU and GPU.
The NVIDIA cuBLAS library is a fast GPU-accelerated implementation of the standard basic linear algebra subroutines (BLAS).

Experiment Result





Acknowledgements

This project is supported by National Science Foundation. Thanks to University of Tennessee Knoxville, JICS, and Oak Ridge NaHonal Lab and the Chinese University of Hong Kong.

References

[1]Harris, M. and →, V. (2017). *Unified Memory in CUDA 6*. [online] Parallel Forall. Available at: https://devblogs.nvidia.com/parallelforall/unified-memory-in-cuda-6/ [Accessed 21 Jun. 2017].
[2]Mahoney, M. (2011). *Randomized algorithms for matrices and data*. Hanover, Mass.: Now Publishers.
[3]Drinea, E., Drineas, P. and Huggins2, P. (2017). *A Randomized Singular Value Decomposition Algorithm for Image Processing Applications*. [ebook] 1 Computer Science Department, Harvard University Cambridge, MA 02138, USA 2 Computer Science Department, Yale University New Haven, CT 06520, USA. Available at: http://ai2-s2-pdfs.s3.amazonaws.com/e881/439705f383468b276415b9d01d0059c1d3e5.pdf [Accessed 26 Jun. 2017].
[4] En.wikipedia.org. (2017). *Latent semantic analysis*. [online] Available at: https://en.wikipedia.org/wiki/Latent_semantic_analysis [Accessed 29 Jun. 2017].
[5] Cs.virginia.edu. (2017). *Pinned vs. non-pinned memory*. [online] Available at: https://www.cs.virginia.edu/~mwb7w/cuda_support/pinned_tradeoff.html [Accessed 11 Jul. 2017].