ITERATIVE SINGULAR VALUE DECOMPOSITION OF LARGE SPARSE MATRICES USING MEMSTAC™

MADUSHANKA SOYSA
JON COKER
SYSTEM ARCHITECTURE
OMNITIER, INC
ROCHESTER, MN
## TABLE OF CONTENTS

Executive summary ........................................................................................................................................... 2
Introduction ...................................................................................................................................................... 2
Results ............................................................................................................................................................... 3
- Matrix size versus decomposition time ........................................................................................................... 3
- Singular value decomposition of a reduced rank matrix ................................................................................... 4
  - Number of singular values vs Relative difference in the norm ................................................................. 4
  - Decomposition time vs Number of singular values ..................................................................................... 5
- Hardware configuration ................................................................................................................................... 6
- Matrix storage format ...................................................................................................................................... 6
Conclusions ....................................................................................................................................................... 6
References ......................................................................................................................................................... 7
EXECUTIVE SUMMARY

MemStac™ is a high-performance replacement for MemCached [1] systems, using tiered memory solutions in which the majority of the cache information resides in low-cost, NAND Flash SSDs. MemStac™ systems exhibit a best-in-class tiered memory solution for high-performance key-value caches and temporary stores, achieving network limited performance for the large sparse matrix singular value decomposition problem. MemStac™ users will see substantial cost savings using drop-in replacements to standard MemCached systems with little or no change in system performance.

In this work, we employ a MemStac™ server as a temporary key-value store in tandem with a specially-optimized algorithm running on a compute server, in order to compute the thin singular value decomposition (SVD) of very large, sparse matrices. We demonstrate that matrices up to 10TB in size can easily be decomposed; far larger than is possible with DRAM-based servers. Our implementation is a prototype of future OmniTier CompStor™ compute/store clusters, which will significantly extend these results.

INTRODUCTION

The singular value decomposition (SVD) of a matrix $A$ of size $m \times n$ is

$$A = U\Sigma V^*,$$

where $U$ and $V$ are unitary matrices of size $m \times m$ and $n \times n$, respectively, and $\Sigma$ is a diagonal matrix of size $m \times n$. $V^*$ is the conjugate transpose of $V$. The diagonal values of $\Sigma$ are called the singular values of $A$. Columns of $U$ are the left-singular vectors and the columns of $V$ are the right-singular vectors [2].

SVD has a wide range of applications, including the calculation of pseudoinverse of a matrix, least square minimization, homogeneous linear equation solution, and low-rank matrix approximation [2]. For a low-rank matrix approximation, a subset of singular values and associated singular vectors are calculated. This is widely applied in the areas of signal processing [3] and information retrieval [4].

Lanczos bidiagonalization enables computation of a partial singular value decomposition of a large sparse matrix [5, 6]. This is an efficient way of extracting a subset of singular values and the corresponding singular vectors.

Let $A (m \times n)$ be a sparse matrix. The approach involves the factorization

$$A = PBQ^*,$$

where $B (m \times n)$ is a bi-diagonal matrix and $P (m \times m)$ and $Q (n \times n)$ are unitary matrices. Then, $B$ has the same singular values as $A$. To perform a partial singular value decomposition, we compute $P_k(m \times k)$, $B_k(k \times k)$, and $Q_k(k \times n)$, such that

$$AQ_k = P_kB_k.$$

The decomposition algorithm, called Golub-Kahan-Lanczos (GKL) bidiagonalization, is presented in [6].
We consider a case where matrix $A$ is too large to store in DRAM. We use MemStac™ key-value store with Memcached protocol to store and access the sparse matrix $A$. We require approximately $16(m + n + k) + 1,152,000 N_t$ bytes of DRAM on the compute server during the iterative process, where $N_t$ is the number of parallel computation threads. For the SVD of a 10 TB matrix, we require only 256 MB DRAM in the compute server for intermediate calculations. The DRAM requirements in this SVD implementation are typically a negligible fraction of the matrix size.

RESULTS

The singular value decomposition performance of our prototype CompStor™ system is given below. The system consists of a single MemStac™ server and a single compute server, connected by a 10GbE network link.

First, we show the performance of SVD using a full-rank, sparse matrix $A$ over a wide range of matrix sizes. Second, we consider the compression application of SVD using a small-rank $A$.

MATRIX SIZE VERSUS DECOMPOSITION TIME

In this test, we generate and store a sparse random matrix with density 0.001 in a MemStac™ server, and then calculate 20 singular values of the matrix and their associated left and right singular vectors. This test is repeated for matrices with different sizes, while maintaining the ratio between number of rows to number of columns at 100. The decomposition time against the matrix size is plotted in Figure 1. The network bound, i.e. the time taken to transfer the data from MemStac™ server to the client machine via a 10 GbE link, is shown in the dashed line.

![Figure 1: SVD time vs Matrix size](image)
SINGULAR VALUE DECOMPOSITION OF A REDUCED RANK MATRIX

In this test, we generate the sparse matrix \( A = WX \), where \( W (m \times k) \) and \( X (k \times n) \) are sparse matrices with density 0.1. The minimum number of non-zero elements in each row of \( W \) and in each column of \( X \) is ensured to be 1. Note that the rank of \( A \) is bounded by \( k \).

In this section, we show that the SVD can be used to recover the rank \( k \) and a compact representation of the rows of \( A \) in terms of a set of \( k \) basis vectors. The SVD operation viewed in this light is usually called principal component analysis. Alternatively, we may take the \( k \) basis vectors (the columns of \( V \)) as a dictionary, and each of the rows of \( A \) can be represented by only \( k \) scalars which are the coordinates of the row in the dictionary vector space. The SVD operation in this perspective is a data compression technique.

We run the test for two matrices of size 0.92 TB \((m = 4,300,000, n = 43,000, \text{ and } k = 20)\) and 9.2 TB \((m = 13,600,000, n = 136,000, \text{ and } k = 20)\).

NUMBER OF SINGULAR VALUES VS RELATIVE DIFFERENCE IN THE NORM

In this section, we develop a termination method which guarantees that all significant singular values have been identified. This method is superior to simply stopping when singular values reach some absolute scale, because that method is sensitive to the absolute scale of the matrix itself, which may vary widely.

Using the result \( \|A\|_F = \sqrt{\sum_{i=1}^{N_\sigma} \sigma_i^2} \), where \( \|A\|_F \) is the Frobenius norm of matrix \( A \), \( \sigma_i \) are its singular values, and \( N_\sigma \) is the number of singular values, we can estimate a bound on the magnitude of the singular values remaining to be calculated at each iteration of the GKL algorithm.

![Graph showing relative difference between the norms of A and its approximation with K-singular values](image)

**Figure 2:** Relative difference between the norms of \( A \) and its approximation with \( K \)-singular values.
In Figure 2, we show the relative difference of the norm vs the number of singular values calculated. The relative difference of the norm is defined as 

\[ 1 - \frac{\|A_k\|_F}{\|A\|_F}, \]

where \( A_k \) is the approximation for \( A \) based on the partial singular value decomposition of \( k \) singular values. We note that 22 iterations of the GKL algorithm is sufficient to extract all the non-zero singular values.

From [7], we have

\[ \min_{Z: \text{rank}(Z) = k} \|A - Z\|_F = \|A - A_k\|_F = \sigma_{k+1}, \]

where \( A_k \) is the matrix built from the \( k \) non-zero singular values and corresponding singular vectors such that \( A_k = U_{m \times k} \Sigma_{k \times k} V_{k \times n}^*, \) calculated via the GKL bidiagonalization algorithm. Since \( A \) has rank \( k \), we know \( \sigma_{k+1} \) is zero. Therefore, \( A = A_k \). This enables us to compress the original \( m \times n \) matrix into a matrix of size \( m \times k \) and a dictionary matrix of size \( k \times n \), without losing any information.

In our first example, the partial SVD allows us to compress the \( 4,300,000 \times 43,000 \) matrix (using 0.92TB) into a \( 4,300,000 \times 20 \) matrix (using 688MB), which can be reconstructed using the dense dictionary matrix of size \( 20 \times 43,000 \). In the second example, we are able to compress a \( 13,600,000 \times 136,000 \) matrix (9.2TB) into a \( 13,600,000 \times 20 \) matrix (2.18GB).

**DECOMPOSITION TIME VS NUMBER OF SINGULAR VALUES**

In Figure 3, we plot the decomposition time against the number of singular values calculated. The corresponding network bound is shown in the dashed curve.

**Figure 3: Decomposition Time vs Number of Singular Values Computed**
HARDWARE CONFIGURATION

The hardware configuration of the servers used in this demonstration are as shown in Table 1.

<table>
<thead>
<tr>
<th>Key-value MemStac™ Server</th>
<th>NEC 2-socket Intel Xeon CPU E5-2699 V4, 2.2Ghz, 20 cores per socket</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client Compute Server</td>
<td>NEC 2-socket Intel Xeon CPU E5-2699 V4, 2.2Ghz, 22 cores per socket</td>
</tr>
<tr>
<td>Operating system</td>
<td>Ubuntu Server 16.04.02 LTS</td>
</tr>
<tr>
<td>Solid State Drives</td>
<td>3.2TB Toshiba PX04PMC320 NVMe SSD (4KiB random reads: 660K IOPS; 4KiB random writes 185K IOPS; sequential read 3100 MiB/s; sequential write 2350 MiB/s). Four SSDs per server.</td>
</tr>
<tr>
<td>NVMe driver</td>
<td>Standard Linux NVMe driver V1.0 for Linux 4.4</td>
</tr>
<tr>
<td>Network Interface Card</td>
<td>10GbE Intel X550T (rev 01)</td>
</tr>
<tr>
<td>NIC driver</td>
<td>ixgbe 4.2.1-k</td>
</tr>
<tr>
<td>10GbE network switch</td>
<td>N/A, direct connected NICs</td>
</tr>
</tbody>
</table>

**TABLE 1: HARDWARE CONFIGURATION**

The key-value server is installed with four instances of the general availability release of MemStac™ software. The client compute server running the SVD algorithm is connected to the MemStac™ server via a 10Gbps Ethernet link.

MATRIX STORAGE FORMAT

The sparse matrices are stored in the key-value server as columns and rows, to facilitate straightforward left and right vector multiplication. The non-zero values of each row (or column) is stored as one or more keys, maintaining the maximum value length of a key at 1MB, consistent with the standard Memcached protocol limitations. The corresponding indexes of the non-zero elements are stored in separate keys.

CONCLUSIONS

Performance of the GKL singular value decomposition as optimized in this work is limited by a network bound. With MemStac™’s use of SSDs, storage can be increased with no noticeable reduction in performance compared to DRAM-only storage servers; i.e. MemStac™ provides optimal storage processing performance for singular value decomposition of large sparse matrices computed over a 10GbE network.

It is to be expected that matrices small enough to be represented locally in the compute server’s DRAM can be decomposed significantly faster than the same size matrix decomposed by the methods developed for this work. In practice these matrices cannot exceed 1TB. We have shown that matrices exceeding 10TB in size can easily be analyzed using a single compute server in very reasonable times using a low-cost, low-power MemStac™ server.
OmniTier’s future CompStor™ products will extend these impressive results significantly by collocating data and compute resources, and by sharing not only storage resources, but also compute resources over a networked CompStor™ cluster. These clusters will allow analysis of even larger matrices at a fraction of the compute times demonstrated here. OmniTier is prepared to integrate this GKL singular value decomposition solution in a user-defined application to demonstrate the performance benefits of our CompStor™ prototype.

REFERENCES


