Accelerated protein structure comparison using TM-score-GPU

Ling-Hong Hung1.∗, and Ram Samudrala1
1Department of Microbiology, University of Washington, Seattle WA.
Associate Editor: Prof. Anna Tramontano

ABSTRACT
Motivation:
Accurate comparisons of different protein structures play important roles in structural biology, structure prediction and functional annotation. The RMSD after optimal superposition is the predominant measure of similarity due to the ease and speed of computation. However, global RMSD is dependent on the length of the protein and can be dominated by divergent loops which can obscure local regions of similarity. A more sophisticated measure of structure similarity, TM-score, avoids these problems, and is one of the measures used by the community wide experiments of Critical Assessment of protein Structure Prediction (CASP) to compare predicted models with experimental structures. TM-score calculations are, however, much slower than RMSD calculations. We have therefore implemented a very fast version of TM-score for Graphical Processing Units (TM-score-GPU), using a new and novel hybrid Kabsch/quaternion method for calculating the optimal superposition and RMSD that is designed for parallel applications. This acceleration in speed allows TM-score to be used efficiently in computationally intensive applications such as for clustering of protein models and genome-wide comparisons of structure.

Results:
TM-score-GPU was applied to 6 sets of models from Nutritious Rice for the World for a total of 3 million comparisons. TM-score-GPU is 68 times faster on an ATI 5870 GPU, on average, than the original CPU single threaded implementation on an AMD Phenom II 810 quad-core processor.

Implementation:
The implementation is in C++/OpenCL

Availability:
The complete source, including the GPU code and the hybrid RMSD subroutine can be downloaded and used without restriction at http://software.compbio.washington.edu/misc/downloads/tmscore/.

1 INTRODUCTION

For protein structure comparisons, the simplest method is to calculate a transformation that superimposes corresponding atoms from one structure onto a second structure and minimizes the root-mean-square-deviation (RMSD) between the coordinates of the superimposed structures (equation 1). This can be obtained from the single value decomposition (SVD) of the covariance matrix (Kabsch) (Kabsch, 1976) or from the solution of the eigenvalue equation of a quaternion derived matrix (Liu, et al., 2010) (Hung, et al., 2011). While RMSD is a fast and easily calculated metric of structural similarity, a globally optimal transformation that minimizes the distances between all superimposed atom pairs can be dominated by a small set of divergent atoms in loop regions. Furthermore, RMSD is not only dependent on the overall goodness of fit but is also dependent on the length of the proteins. TM-score (Zhang and Skolnick, 2004) uses a variant of the Levitt-Gerstein (LG) metric (Gerstein and Levitt, 1998) which provides a length independent measure and limits the impact of divergent pairs of atoms in superimposed structures (equation 1).

\[
\text{RMSD} = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} d_i \quad \text{LG} = \frac{1}{L} \sum_{i=1}^{L} \frac{d_i}{d_i^*} \quad (1)
\]

In the above formula \(L\) is the length of the protein, \(d_i\) is the distance between the \(i^{th}\) matched Cα atom and \(d_i^*\) a scaling factor to normalize the matches. The LG based metric gives a value between 0 and 1 where 1 is an exact match. The maximum value of LG that can be obtained by superposition is the TM-score. Unlike RMSD, there is no simple relationship between the covariance matrix and the optimal transformation that maximizes LG. Instead, different subsets of atoms are superimposed using the Kabsch algorithm and the LG score evaluated over the entire protein. By sampling a large number of subsets, an approximately optimal superposition can be obtained. Because of the numerous local superpositions that must be sampled, the TM-score algorithm is much slower than the calculation of global RMSD. We present TM-score-GPU which is a fast Graphical Processing Unit (GPU) implementation of TM-score using a new hybrid RMSD algorithm that is suitable for parallel single instruction multiple data (SIMD) applications.

2 METHODS

GPUs rely on the same instructions being executed simultaneously on different data (SIMD) to accelerate the calculations. For each group of data (wavefront), all branches of conditional code are executed which makes complicated branching of code slow for GPUs. Iteration can also be expensive for SIMD applications as all threads wait for the longest iteration to finish. Our hybrid implementation first calculates the eigenvalues of the covariance matrix \(S\) by analytically solving for the roots of the cubic characteristic polynomial (equation 3). This is the first part of the Kabsch algorithm. The quaternion algorithm obtains the optimal superposition and RMSD by solving for the eigenvalues and eigenvectors of matrix \(S\) in equation 4. \(d\) from equation 3 is also an eigenvalue of matrix \(S\). The fast analytical method from qcprot (Liu, et al., 2010) is then used to solve for the eigenvectors of matrix \(S\) and construct the rotation matrix (equations 2-4 and equations 1-8 in supplemental materials). This hybrid method avoids the complicated branching code used to calculate rotations in the Kabsch method and avoids the iterative methods used to solve for the eigenvalues of the \(S\) matrix used in quaternion methods. The resulting algorithm has a non-divergent code path, has no iteration and is significantly faster. The single-threaded TM-score implementation using the hybrid method is 45% faster than the original Kabsch based implementation and 58% faster than qcprot (see figure 1).

\[(S \cdot d) = 0 \quad (2)\]
where \( \mathbf{v} \) are the eigenvectors of \( \mathbf{S} \), \( \mathbf{R} \) is the covariance matrix, \( \mathbf{d} \) is derived from the eigenvalues of \( \mathbf{R} \):

\[
\mathbf{R} = \mathbf{V} \mathbf{d} \mathbf{V}^T \quad \mathbf{d} = \mathbf{e} + \mathbf{c} + (\text{sign}(|\text{det}(\mathbf{R})|) \cdot \text{min}(d_1, c_2, c_3))
\]

(3)

\[
\begin{align*}
R_{11} + R_{22} + R_{33} & \quad R_{23} - R_{12} & \quad R_{31} - R_{13} & \quad R_{12} - R_{21} \\
R_{23} - R_{12} & \quad R_{11} - R_{22} + R_{33} & \quad R_{21} - R_{12} & \quad R_{32} - R_{23} \\
R_{31} - R_{13} & \quad R_{12} - R_{21} & \quad R_{11} + R_{22} - R_{33} & \quad R_{23} + R_{32} \\
R_{12} - R_{21} & \quad R_{31} + R_{13} & \quad R_{21} + R_{32} & \quad R_{11} - R_{22} + R_{33}
\end{align*}
\]

(4)

Memory I/O is the other major bottleneck for GPU implementation. It is advantageous to buffer coordinates into fast local and register memory. The number of superpositions increases roughly by \( N \log N \) where \( N \) is the number of residues whereas the cost to buffer the coordinates grows linearly. Therefore, the caching of coordinates is especially beneficial for larger proteins. A greater number of active threads can also be beneficial by allowing the scheduler to switch between threads when one is stalled during a memory wait state. The application therefore calculates the optimal number of SIMD threads that can be launched without exhausting the scarce fast memory resources. The implementation also rearranges the coordinates into vectors of 4 floating point values. ATI GPUs are optimized for I/O and compute operations on 4-vectors. The implementation is in C++/OpenCL using AMDAPP2.5 for Linux and is optimized for ATI cards. However the OpenCL code can be compiled and optimized for other GPUs and CPUs that support double precision arithmetic.

3 RESULTS

TM-score-GPU was applied to 6 sets of 1000 de novo protein models from Nutritious Rice for the World. The TM-score between each pair of structures within an ensemble was then calculated. TM-score-GPU is 68 ± 3 times faster than an ATi 5870 GPU, on average, than the original CPU single threaded implementation on an AMD Phenom II 810 quad-core processor (see figure 1).

We have implemented a very fast version of the structural comparison TM-score algorithm using a novel hybrid Kabsch/quaternion method for superposition that is suitable for SIMD applications. We anticipate that TM-score-GPU will be useful in applications such as clustering and in methods that compare structures with different sequences such as TM-align (Zhang and Skolnick, 2005). Our particular application is for large community grid projects where few volunteers have access to CPU computational clusters but where GPUs are commonplace and provide most of the compute cycles. In addition, the new SIMD friendly superposition routine will be useful in software where fast parallel superpositions are required. Finally, our implementation is in OpenCL, which is a very new language for GPUs. We anticipate that our implementation be of interest to others who wish to port code from languages that have been deprecated (Brook+) or may soon be deprecated (CUDA). The complete source code, including the GPU and superposition subroutines, is available for unrestricted use from http://software.compbio.washington.edu/misc/downloads/tmscore/.

ACKNOWLEDGEMENTS

We would like to thank Yang Zhang for making the TM-score code freely available; Gaurav Chopra, Jeremy Horst and Ambrish Roy for their helpful comments on the manuscript, and Thomas Wood for testing the source distribution.

Funding: This work was supported by the NIH Director’s Pioneer Award 1DP1OD006779-01.

REFERENCES


