Accelerated protein structure comparison using TM-score-GPU

Ling-Hong Hund† and Ram Samudrala
Department of Microbiology, University of Washington, Seattle, WA 98195-7735, USA

BIOINFORMATICS APPLICATIONS NOTE

Vol. 28 no. 16 2012, pages 2191–2192
doi:10.1093/bioinformatics/bts345

ABSTRACT

Motivation: Accurate comparisons of different protein structures play important roles in structural biology, structure prediction and functional annotation. The root-mean-square-deviation (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 that can obscure local regions of similarity. A more sophisticated measure of structure similarity, Template Modeling (TM)-score, avoids these problems, and it is one of the measures used by the community-wide experiments of critical assessment of protein structure prediction 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 six 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.

Availability and implementation: 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/. The implementation is in C++/OpenCL.

Contact: ram@compbio.washington.edu

Supplementary Information: Supplementary data are available at Bioinformatics online.

Received on April 10, 2012; revised on May 23, 2012; accepted on June 9, 2012

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 of the covariance matrix (Kabsch, 1976) or from the solution of the eigenvalue equation of a quaternion derived matrix (Hung et al., 2011; Liu et al., 2010). Although 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 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) that provides a length independent measure and limits the impact of divergent pairs of atoms in superimposed structures [Equation (1)].

\[
RMSD = \sqrt{\frac{1}{L-1} \sum_{i=1}^{L} \sum_{j=1}^{L} LG} \sum_{i=1}^{L} \sum_{j=1}^{L} 1 + \left(\frac{d_{ij}}{d_0}\right)^2
\]

In the above formula, \(L\) is the length of the protein, \(d_{ij}\) is the distance between the \(i\)th matched Ca atom and \(d_0\) a scaling factor to normalize the matches. For small proteins the optimal value of \(d_0\) is 4.5 Å. 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 square of the covariance matrix \(R\) by analytically solving for the roots of the cubic characteristic polynomial [Equation (4)]. 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) and from Equation (5) 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) as Supplementary material]. This hybrid method avoids the complicated branching code used to calculate...
When comparing single-threaded CPU implementations, our hybrid RMSD algorithm gives rise to a 45% speedup over an implementation using qcprot (Liu et al., 2010) (red) and the CPU and GPU versions of our implementation (green and yellow). TM-score-GPU is on average 68±3 times faster than the original Kabsch-based implementation (black), an implementation of TM-score using qcprot (Kabsch, W. (1976) A solution for the best rotation to relate two sets of vectors. Acta Crystallogr. A, 32, 222–225), the CPU and superposition subroutines, is available for unrestricted use from http://software.compbio.washington.edu/misc/downloads/tmscore/index.html.

Memory IO 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 four floating point values. ATI GPUs are optimized for I/O and memory resources. The implementation also rearranges the coordinates into vectors of four floating point values. ATI GPUs are optimized for I/O and memory resources.

Memory IO 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 four floating point values. ATI GPUs are optimized for I/O and memory resources. The implementation also rearranges the coordinates into vectors of four floating point values. ATI GPUs are optimized for I/O and memory resources.