Journal of Structural Biology 172 (2010) 407–412

Contents lists available at ScienceDirect

Journal of Structural Biology

journal homepage: www.elsevier.com/locate/yjsbi

Technical Note

GPU-enabled FREALIGN: Accelerating single particle 3D reconstruction and refinement in Fourier space on graphics processors

Xueming Li [a], Nikolaus Grigorieff [b], Yifan Cheng [a,*]

[a] The W.M. Keck Advanced Microscopy Laboratory, Department of Biochemistry and Biophysics, University of California San Francisco, 600 16th Street, San Francisco, CA 94158, United States
[b] Howard Hughes Medical Institute, Brandeis University, Waltham, MA 02454, United States

ARTICLE INFO

Article history:
Received 2 April 2010
Received in revised form 2 June 2010
Accepted 8 June 2010
Available online 15 June 2010

Keywords:
Single particle electron microscopy
GPU computing
FREALIGN
Image processing

ABSTRACT

Among all the factors that determine the resolution of a 3D reconstruction by single particle electron cryo-microscopy (cryoEM), the number of particle images used in the dataset plays a major role. More images generally yield better resolution, assuming the imaged protein complex is conformationally and compositionally homogeneous. To facilitate processing of very large datasets, we modified the computer program, FREALIGN, to execute the computationally most intensive procedures on Graphics Processing Units (GPUs). Using the modified program, the execution speed increased between 10 and 240-fold depending on the task performed by FREALIGN. Here we report the steps necessary to parallelize critical FREALIGN subroutines and evaluate its performance on computers with multiple GPUs.

1. Introduction

Single particle electron cryo-microscopy (cryoEM) is a versatile tool for studying the three-dimensional (3D) structures of biological macromolecules at high resolution. Recently, this technique has been used to determine near-atomic resolution structures of icosahedral viruses (Zhang et al., 2008, 2010; Yu et al., 2008; Chen et al., 2009; Wolf et al., 2010) and large asymmetric protein complexes (Cong et al., 2010). In single particle cryoEM, a 3D reconstruction is calculated from a large number of individual images representing different views of the same molecule. The resolution of a 3D reconstruction is improved iteratively by refining the geometric parameters of each particle, including three Euler angles and two in-plane shifts, and microscope parameters, including defocus and magnification. To reach high resolution, especially when a particle exhibits little or no symmetry, a large number of images are needed to improve the signal-to-noise ratio (SNR) of the 3D reconstruction and to provide finer sampling in Fourier space. With the development of advanced electron microscope systems and large format digital image recording devices, it is now feasible to acquire very large single particle datasets. The time required to process these datasets depends almost linearly on the number of particles in the datasets. Therefore, shortening the time needed for data processing becomes one of the key issues in high-resolution single particle cryoEM.

A commonly used method in single particle cryoEM data processing is to split a dataset into many smaller subsets and distribute them over a cluster of many computer nodes with single Central Processing Units (CPUs), or to parallelize the algorithm to take advantage of a multi-CPU architecture (Bilbao-Castro et al., 2009; Fernandez, 2008; Yang et al., 2007). The data processing speed on this kind of multi-CPU architecture depends on the number of CPUs. An alternative approach is to use Graphics Processing Units (GPUs), which, unlike CPUs, have hundreds of vector processing cores, integrated into one unit. With a large number of cores in one processor, a GPU is capable of performing massively parallelized scientific computing by concurrently applying a single instruction to multiple data (the so-called SIMD architecture that is commonly used in vector supercomputing). For cryoEM, the idea of using GPUs for data processing has been extensively evaluated and tested in the past few years (Castano-Diez et al., 2008; Schmeisser et al., 2009). Many recent efforts of using GPUs to accelerate cryoEM data processing have shown some significant speed increases (Castano Diez et al., 2007; Schmeisser et al., 2009; Tan et al., 2009).

FREALIGN is a stand-alone program for very efficient refinement and 3D reconstruction (Grigorieff, 2007). A number of subnanometer resolution (Cheng et al., 2004; Fotin et al., 2006; Rabl et al., 2008) or near-atomic resolution (Chen et al., 2009; Wolf et al., 2010; Zhang et al., 2008, 2010) structures were reconstructed and refined using this program. In FREALIGN, refinement can easily be implemented on a multi-CPU cluster for distributed computing. The time needed to complete a cycle of refinement depends on the

* Corresponding author. Fax: +1 415 514 4145.
E-mail address: ycheng@ucsf.edu (Y. Cheng).

1047-8477/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jsb.2010.06.010

number of particles in the dataset and the number of CPUs used for the refinement. The 3D reconstruction step is often carried out on a single CPU. The latest version of FREALIGN (version 8.08) offers some parallelization of the reconstruction step using OpenMPI to accelerate 3D reconstruction (about 3-fold on an 8 core CPU system).

We have modified FREALIGN to utilize the parallel computing power of GPUs. Using a test dataset of images of 20S proteasome, we have achieved a ~10-fold speed increase in the search and refinement steps and a ~240-fold speed increase in calculating a 3D reconstruction. While the detailed computational procedures are modified to enable GPU enhancement, the algorithms for single particle refinement and 3D reconstruction used by the original FREALIGN code were unaltered. Thus, the results obtained with our GPU-enabled FREALIGN code are essentially the same compared to those obtained from the original code. Here we report the computational procedures of our modification to FREALIGN, hardware configurations and test results.

2. Parallelization of FREALIGN procedures

The general concept and technical details of utilizing the CUDA language to program GPUs for cryoEM image processing have been described in detail, for examples (Castano-Diez et al., 2008; NVIDIA, 2009; Schmeisser et al., 2009). Our current study aims to specifically accelerate FREALIGN on NVIDIA GPUs using CUDA.

The mode of operation in FREALIGN is controlled by an integer called IFLAG. In each mode, the computational procedures can be divided into various subroutines (Grigorieff, 2007). A primary function evaluated multiple times during a search or refinement (IFLAG $\neq 0$) is the weighted correlation coefficient, or its inverse cosine (FREALIGN's phase residual). It is calculated between the Fourier transform (FT) of a two-dimensional (2D) particle image and a central section through the FT of the 3D reference. During an iterative search (IFLAG = 4) and refinement (IFLAG = 1) a Powell optimization function is used for maximizing the correlation coefficient of each individual image by optimizing all geometric and microscopy parameters. In a 3D reconstruction (IFLAG = 0), the FTs of all particle images, weighted by using their phase residuals, are inserted into the 3D FT of the new 3D reconstruction using an interpolation function. Thus, two major procedures in FREALIGN are used to calculate correlation coefficients and to interpolate between grid points.

During a search or refinement, the correlation coefficient is calculated by accumulating the co-variances between two images or their FTs. One image is the particle image shifted to the image origin and multiplied by the CTF. The other is a central section of the 3D FT of the reference volume masked by a cosine-edged mask and multiplied by the square of the CTF. To calculate a central section of the 3D volume the value of each pixel in the resulting 2D image is calculated as the sum of the contributions from the nearest $n \times n \times n$ neighbors (n = 1, 3, or 5). To calculate a 3D reconstruction the contribution of each 2D image pixel is added to its eight nearest neighbors according to the view presented by the particle in the image.

Table 1 lists the major FORTRAN subroutines in FREALIGN that make use of the correlation coefficient or interpolation functions. Supplementary Table 1 lists the execution times in microseconds needed to complete each subroutine for a particular image size, and Supplementary Table 2 lists the percentage of total runtime taken by each subroutine. Subroutines *cc3m*() and *ccp*() are used in search and refinement. Both include three parts: extracting a central section from the 3D FT, applying the cosine mask and CTF to the section, and computing the correlation coefficient. The total time used by these two subroutines in a search or refinement cycle depends on the parameter setting, because they are executed more than once, but together they account for more than 95% of the execution time. The main function of subroutine *pinsert_s*() is the interpolation used in the 3D reconstruction procedure. It accounts for most of the execution time used for 3D reconstruction and is invoked N times for each contributing image, where N is the symmetry of the particle. The other three subroutines are used less often in a search/refinement/reconstruction cycle.

Table 1
List of primary FREALIGN subroutines that can be parallelized.

	Subroutine	Description
Refinement	*cc3m*()	Calculate overall phase residue in Fourier space
	ccp()	Calculate cross-correlation in Fourier space
Reconstruction	*pinsert_s*()	Interpolate a particle image into 3D Fourier transform of the volume being reconstructed
Other	*sigma2*()	Calculate correlation coefficient in real space
	presb()	Calculate amplitude correlation coefficient in Fourier space
	ccoef()	Calculate correlation coefficient in real space

In the original version of FREALIGN, the subroutines listed in Table 1 consist of calculating FTs and loops over each pixel in a 2D image for interpolation, masking and summation. These loops execute the same computation for each pixel and can, therefore, be parallelized following the SIMD architecture. The computations within a loop are first cast into a function to be executed on a GPU, named *kernel* in CUDA (NVIDIA, 2009). Furthermore, a thread array is created according to the dimension of the pixel array of the 2D image. All threads in the array execute the kernel concurrently, one for each pixel in the image. For the computation in real space, the dimension of both thread array and image array is $m \times m$, where m is the dimension of the particle image. In Fourier space, FREALIGN uses reduced image arrays to save memory. Thus, the dimension of both the image and thread arrays is $m \times (m/2 + 1)$. The entire thread array needs to be divided into many blocks in order to run on multiple processors (MPs) of the GPU. In the current version, the dimension of blocks for computing the correlation and interpolation is 32×16 and 16×16 threads, respectively, following recommendations by the manufacturer (NVIDIA, 2009).

Most loops in FREALIGN's procedures can be parallelized following the method mentioned above except those involving summation. In CUDA, the addition operation includes three individual steps: an argument is read from memory, a number is added to it, and the result is written back to the same memory location. If executed by multiple threads concurrently, there are two potential problems that are commonly known as data hazard. First, when an argument is read by one thread for addition, the same argument could be read by another thread for the same operation before it is updated from the result of the first thread. Second, if more than one thread writes their results to the same address at the same time, CUDA only guarantees one of the writes to succeed (NVIDIA, 2009). In both cases, CUDA will return an undefined result. In the parallelized FREALIGN subroutines, a potential data hazard occurs in two places: one is the interpolation for the 3D reconstruction, when the contributions from different image pixels are added simultaneously to the same pixels of the FT of the new reconstruction; the other is the summation of co-variances for calculating the correlation coefficient.

The first situation occurs infrequently, but the memory address accessed by multiple threads is usually unpredictable. We avoid data hazard by setting a barrier to prevent the addition operation to be interrupted. Although this approach effectively converts the addition to the same address into a sequential operation, it does not have a serious impact on the performance due to its rare occurrence. The second situation is more complicated because all

threads calculate additions to the same address. With the barrier directive, the resulting sequential addition would slow execution substantially. To overcome this problem, we adopted a parallel tree-structured summation algorithm as shown in Fig. 1A. The summation between every two pixels is carried out in a synchronized manner to get a new set of data with half the size. This procedure is repeated until the final result is obtained. According to CUDA, the synchronization is only available for the threads in the same blocks and blocks in the same grid. Therefore, the tree-structured summation is at first calculated within blocks. The result from each block is written to global memory. After all blocks have finished, the results are summed up using the same method again until the final result is obtained.

In order to minimize data transfer between host and GPU all data are stored in the global memory of the GPU whenever possible. When a FREALIGN job is launched, either for a search/refinement or a 3D reconstruction, the 3D reference volume and/or the new 3D reconstruction are loaded into the global memory of the graphics card at the beginning. A 2D particle image is loaded when it is used for the first time. In addition, all temporary arrays are allocated in the global memory at the beginning of FREALIGN to avoid the time needed by dynamic memory allocation during the execution.

Because of the relatively high latency of the global memory, we adopted the following strategies to improve execution performance. First, the temporary arrays used in the summation, which will be accessed many times, are loaded from the global memory into the in-chip shared memory of the GPU at the beginning of the kernel launch. Only the final results are written back to the global memory. Second, the pixel arrays for the 2D images and CTF are accessed following a sequential pattern, thus enabling the GPU to coalesce many accesses to a single access. Third, the FT of the 3D reference is bound to the read-only texture memory space. When sectioning the FT of the 3D reference map, only pixels of the 3D FT along a defined orientation are accessed by the threads. The random orientation of all 2D images leads to a random memory access pattern. By using the texture hardware, which is designed for such random access, the high latency of the global memory is avoided. However, the size of this texture memory limits the size of the volume it can handle. Currently, we limit the image size that can be handled by the GPU-enabled FREALIGN to 500 × 500.

In this study, we used two computer systems with four or eight GPU devices (Table 2) for our testing. The parallelization of FREALIGN discussed above was expanded to a multi-GPU architecture by partitioning and distributing the dataset evenly between multiple devices, i.e. each GPU device processes only a few particles of the whole dataset. Fig. 1B and C shows schematic diagrams of parallelization on a multi-GPU setup, in which the POSIX thread library is used to create host threads on each CPU that controls GPU devices. Each CPU thread then launches corresponding GPU kernel/threads to process a subset of the data.

In a multi-CPU/GPU architecture, the entire dataset is split sequentially into a number of subsets for the search/refinement. The number of subsets is equal to the total number of GPU devices available. Each subset is then processed independently on each CPU/GPU device and the output parameter files are merged when completed (Fig. 1B). For calculating a 3D reconstruction, assuming there are N GPU devices (N must be an even number), the dataset is split into N subsets in such a way that the particles assigned to the nth subset ($n = 1, 2, \ldots, N$) have the particle numbers of $mN + n$, where m is an integer variable (Fig. 1C). In FREALIGN, a 3D reconstruction is calculated from a 3D FT of a 3D volume in Fourier space where each sample point is calculated as following (Grigorieff, 2007):

$$R_i = \frac{\sum_j w_j^2 b^2 c_j P_{ij}}{f + \sum_j (w_j b c_j)^2}.$$

Here, R_i represents sample i in the 3D FT of the reconstruction, P_{ij} is a sample from the FT of particle image j (before CTF correction) contributing to sample R_i, c_j is the CTF for image j corresponding to that point in the image, b is the interpolation function (for example, box transformation), and w_j is a weighting factor describing the quality of the image. f is a constant, similar to a Wiener filter constant, that prevents over-amplification of terms when the rest of the denominator is small. Thus, a total of N numerator and denominator arrays are calculated in Fourier space from particles of N subsets by N GPUs concurrently. Next, two numerator and denominator arrays are calculated by merging $N/2$ corresponding arrays from the even and odd numbered GPUs, respectively. Two 3D reconstructions are calculated from these two sets of numerator and denominator arrays, respectively, corresponding to all odd and even numbered particles. These two 3D reconstructions

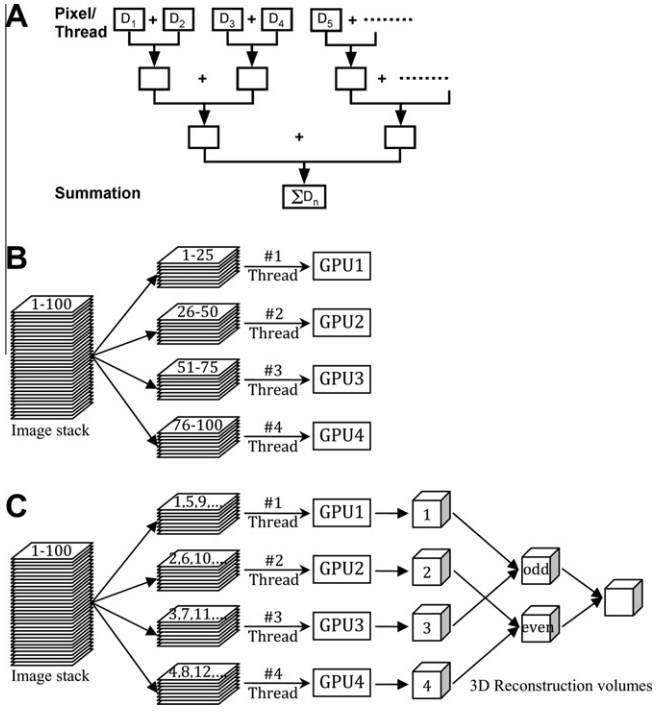

Fig. 1. Implementing parallelization of FREALIGN. (A) Tree-structured parallel summation algorithm. The sums between every two pixels within the same block are computed in synchronization, and repeated until the final result is obtained. (B) Dividing and distributing subsets of the data on multiple GPUs for search and refinement. (C) Dividing and distributing subsets of the data on multiple GPUs for 3D reconstruction.

Table 2
Configurations of the computer systems used for the test.

System	I	II
Operating system	Fedora 11	Centos 5
Motherboard	Asus P6TD Deluxe	SuperServer 7046GT-TRF
CPU	1 × Intel i7 920 quad core, 2.66 GHz	2 × Intel Xeon X5550 quad core, 2.66 GHz
RAM	12 GB DDR3-1333	24 GB DDR3-1333
Graphic Card	2 × NVIDIA GeForce GTX 295	4 × NVIDIA GeForce GTX 295
Total number of GPUs[a]	4	8

[a] Each GTX 295 graphic card was connected through a PCI-E 2.0 16X expansion slot. In both systems, all GPUs were used for computation only (no display). In System I, a third NVIDIA graphic card, GeForce 8400 GS was used to drive a display. In System II, an onboard VGA controller is used for the display.

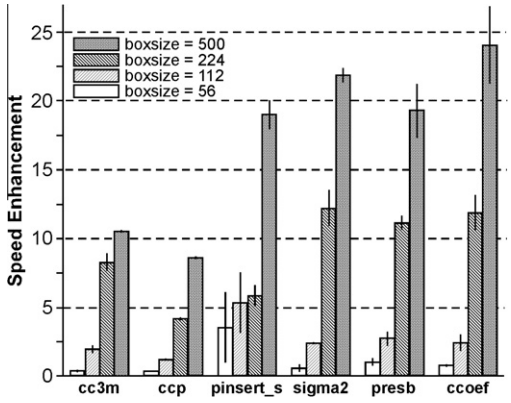

Fig. 2. Speed enhancement factors of individual FREALIGN subroutines. Acceleration of individual subroutines with different image sizes: 56 × 56–500 × 500 pixels. For the measurement of the speed enhancement factors, only one GPU on System I was used.

are used to calculate a Fourier Shell Correlation (FSC) curve. The final 3D reconstruction is then calculated by merging the two 3D reconstructions from odd and even numbered particles. Fig. 1C shows an example of calculating a 3D reconstruction using 4 GPU devices.

3. Test and results

We have parallelized the code of FREALIGN (version 8.06) as described above and tested its performance with two single particle cryoEM datasets. One test dataset contained a total of 44,794 images of archaeal 20S proteasome with 224 × 224 pixels size. Previously, we published a 3D reconstruction calculated from this dataset at a resolution of ~6.8 Å using the unmodified version of FREALIGN. Acquisition and processing of this dataset is described in Rabl et al. (2008). We also generated a series of new test datasets of yeast 20S proteasome with image sizes between 100 and 500. We tested the GPU-enabled FREALIGN on two different multi-GPU systems (System I and II in Table 2). To compare the perfor-

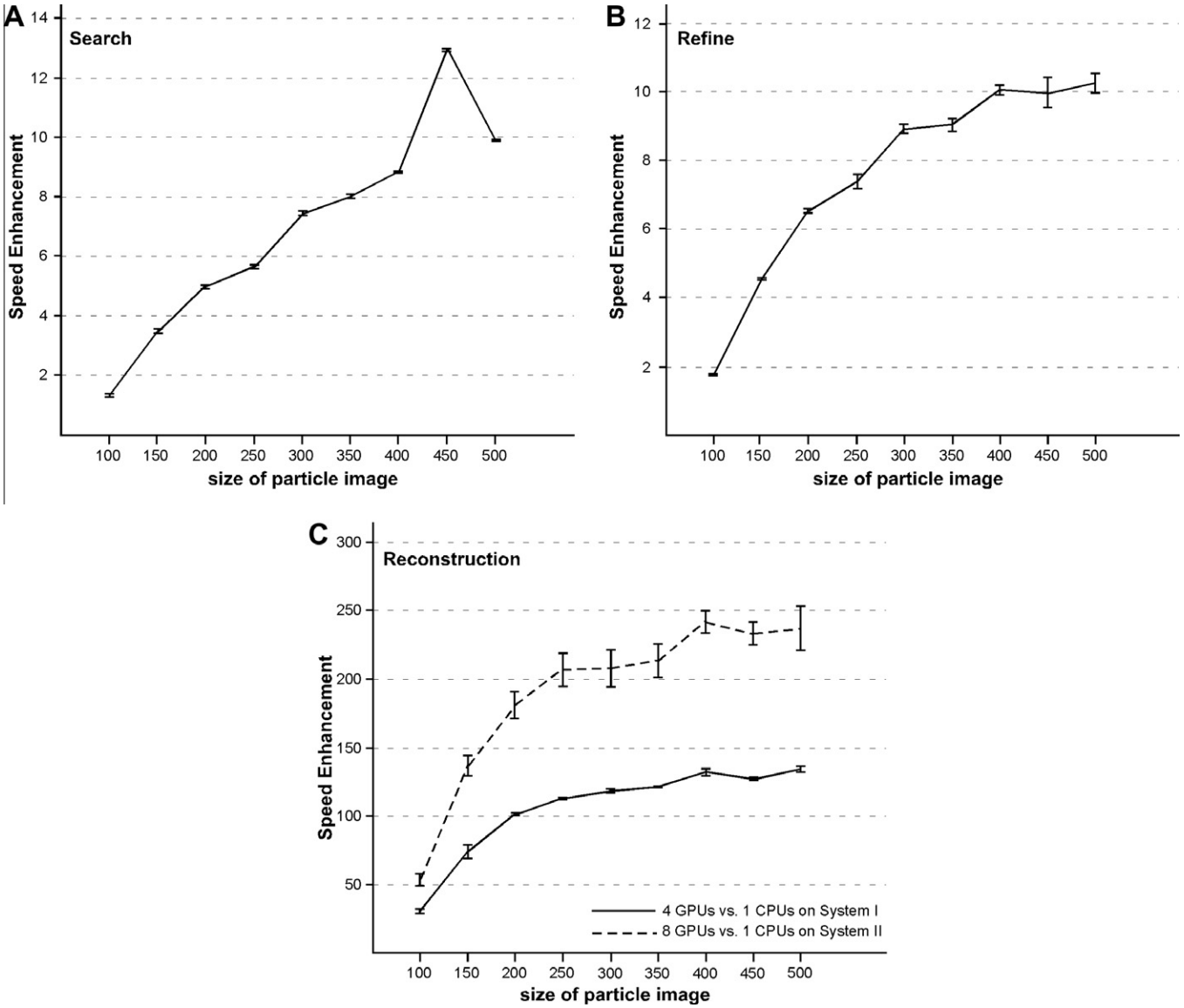

Fig. 3. Acceleration of search and refinement by GPU-enabled FREALIGN with various image sizes. (A) Search and (B) refinement. The accelerations factors were measured on a single GPU on System I vs. a single i7 CPU core on the same system. The corresponding execution times are listed in Supplementary Tables 3 and 4. (C) Reconstruction. The acceleration factors were measured on 4 and 8 GPUs on System I and II, respectively, vs. a single CPU core on the same system.

mance of the unmodified (non-GPU) version of FREALIGN running on a single CPU core with the performance of the GPU-enabled code running on a single GPU, we used System I with only one GPU enabled. This ensured that the hardware configuration remained the same in these tests. The performance of the GPU-enabled FREALIGN was characterized by measuring the speed enhancement factors with respect to the single core CPU reference (Figs. 2 and 3, Supplementary Tables 3 and 4). Furthermore, we determined the additional acceleration gained by using multiple GPUs (four on System I and eight on System II).

The GPU-enabled version of FREALIGN is a mixture of CUDA, C/C++ and FORTRAN77 code. The parts that control data flow and computations were coded in C/C++ and CUDA as described above. All code was compiled and linked using G77, GCC 3.4 and CUDA 2.3. The CPU version of FREALIGN version 8.06 was compiled using default flag of the original package, and the same flags were also used to compile the GPU-enabled version.

We first evaluated the speed increase of six major FREALIGN subroutines listed in Table 1 when processed on a single GPU on System I. For the tests we used a number of different image sizes of the archaeal 20S proteasome dataset, 56×56, 112×112, 224×224 and 500×500 pixels (Fig. 2 and Supplementary Table 1). The speed enhancement factor increases steeply with the increase of the image size. A ~25-fold speed increase was achieved in subroutine "*ccoef*()", and a ~20-fold speed increase was observed for subroutine *pinsert_s*(). The highest speed gain is obtained for 500×500 pixel images, the largest image size we have tested. However, the acceleration factor is less than 1 for images of 56×56 pixels and, therefore, the GPU-enabled FREALIGN is slower than the original version for such small images. The overall dependency of the GPU-enabled FREALIGN on image size is consistent with other reports on similar algorithms (Castano-Diez et al., 2008; Tan et al., 2009).

The overall performances of three different FREALIGN modes, i.e. search (IFLAG = 4), refinement (IFLAG = 1) and reconstruction (IFLAG = 0) were evaluated for various image sizes of yeast 20S proteasome dataset (Fig. 3, Supplementary Tables 3 and 4). The acceleration factors for search and refinement were evaluated as the ratios of execution times of the original and CPU-enabled version of FREALIGN to complete processing of the same number of particles (Fig. 3A and B, Supplementary Table 3). Because the GPU-enabled FREALIGN requires an even number of GPUs for 3D

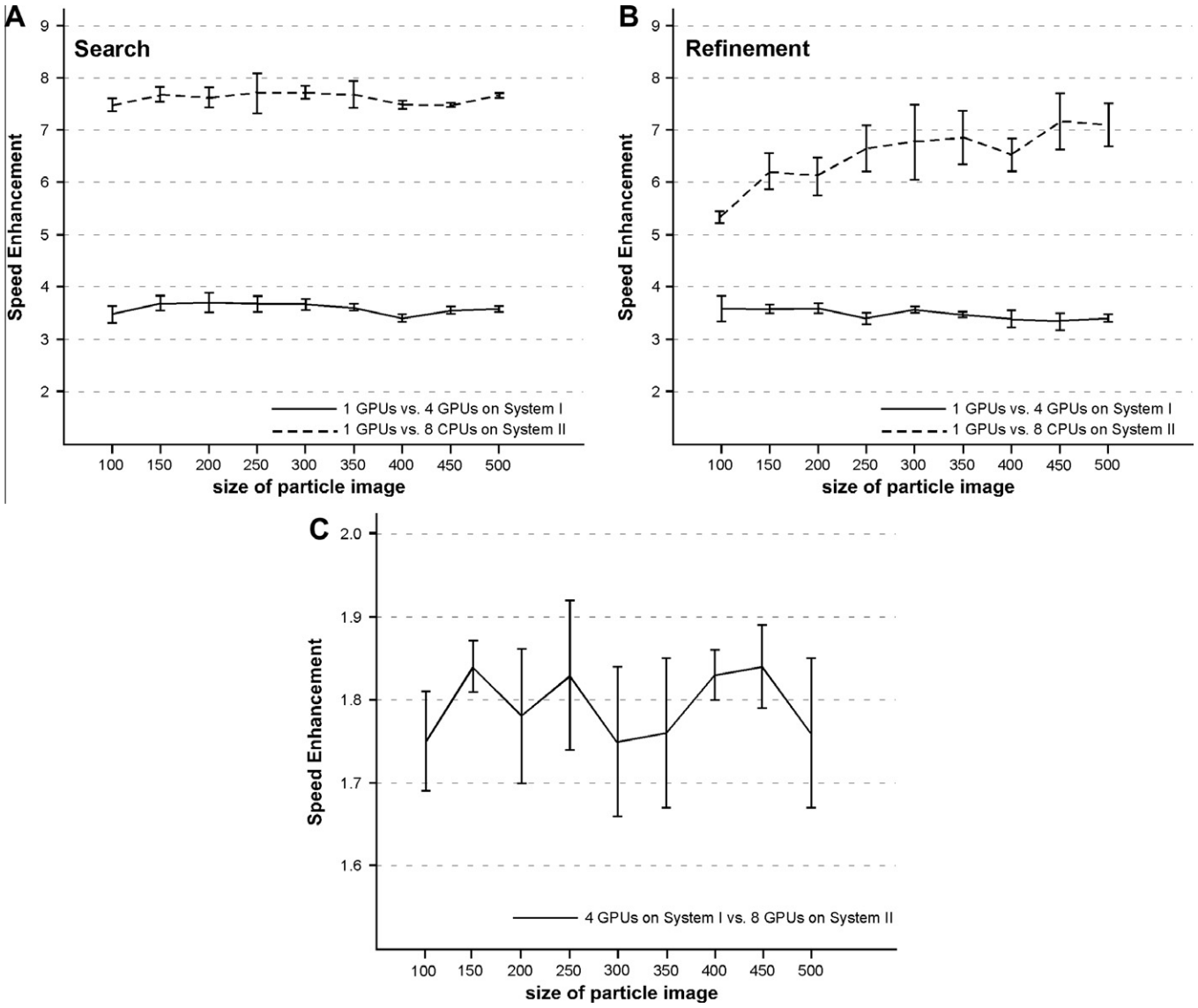

Fig. 4. Acceleration of GPU-enabled FREALIGN on multiple GPUs. (A) Search, (B) refinement and (C) 3D reconstruction. Solid lines are the speed enhancement factors obtained for System I while dashed lines apply to System II. All corresponding data are listed in Supplementary Table 5.

reconstruction (see above), the speed enhancement factors of 3D reconstruction were evaluated by comparing execution times on a single CPU core on System I and System II with execution times using 4 and 8 GPUs on System I or II, respectively. This resulted in speed enhancements of ~130 or ~240-fold for the 4 and 8 GPU systems, respectively (Fig. 3C, Supplementary Table 4).

We also evaluated the acceleration of search and refinement when using multiple GPUs vs. a single GPU (Fig. 4, Supplementary Table 5), and 3D reconstruction when using 8 GPUs vs. 4 GPUs. For all tasks, the acceleration with multiple GPUs is approximately proportional to the total number of GPUs used. The linear behavior parallels the acceleration observed when distributing the parameter search and refinement for particles in a stack as smaller stacks across multiple CPUs in a cluster (see above). One minor difference is that the GPU-enabled FREALIGN handles parallelization over multiple GPUs automatically while job distribution on a cluster is usually handled by a shell script running multiple instances of FREALIGN.

Except for parallelization, we did not alter the original algorithms of FREALIGN. Thus, the results of the GPU-enabled FREALIGN and the original CPU version of FREALIGN are essentially the same (Supplementary Fig. 1).

4. Conclusion

In our current study, the time-critical subroutines of the single particle reconstruction and refinement program FREALIGN were parallelized using CUDA. The goal of this work is to accelerate the iterative refinement-reconstruction cycles performed by FREALIGN using GPUs without changing the accuracy of the result. We focused on parallelizing subroutines that involve mostly computations in Fourier space, and kept the data structure and all the data processing algorithms unchanged. We achieved between 10 and 25-fold speed enhancements in individual procedures, and a ~10-fold acceleration in the overall performance of the search and refinement operations. Similar to the original (non-GPU) version of FREALIGN, the search and refinement tasks can be distributed over multiple GPUs. We also parallelized the 3D reconstruction on multiple GPUs and achieved an acceleration of ~240-fold on an 8-GPU system compared with calculations on a single CPU core. This factor is likely to increase further with more GPUs. Speed enhancement will be smaller when compared to multithreaded 3D reconstruction on a multi-core CPU system using the latest version of FREALIGN (version 8.08), which is OpenMPI-enabled. As expected, comparison of the final reconstructions from all tests show that the results from the GPU-enabled FREALIGN are essentially the same as those obtained from the original FREALIGN running on CPUs. Our current work represents an important step towards a high-performance data processing system for large single particle cryoEM dataset using FREALIGN.

Acknowledgments

We thank David Agard and Shawn Zheng for critical discussions. This study has been supported in part by grants from NIH (R01GM082893, 1S10RR026814-01 and P50GM082250 (to A. Frankel)) and grants from UCSF Program for Breakthrough Biomedical Research (Opportunity Award in Basic Science and New Technology Award) to Y.C., and grants P01 GM62580 to N.G. N.G. is an Investigator in the Howard Hughes Medical Institute. The GPU-enabled FREALIGN can be downloaded from the websites of laboratories of Grigorieff and Cheng.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.jsb.2010.06.010.

References

Bilbao-Castro, J.R., Marabini, R., Sorzano, C.O., Garcia, I., Carazo, J.M., Fernandez, J.J., 2009. Exploiting desktop supercomputing for three-dimensional electron microscopy reconstructions using ART with blobs. J. Struct. Biol. 165, 19–26.

Castano Diez, D., Mueller, H., Frangakis, A.S., 2007. Implementation and performance evaluation of reconstruction algorithms on graphics processors. J. Struct. Biol. 157, 288–295.

Castano-Diez, D., Moser, D., Schoenegger, A., Pruggnaller, S., Frangakis, A.S., 2008. Performance evaluation of image processing algorithms on the GPU. J. Struct. Biol. 164, 153–160.

Chen, J.Z., Settembre, E.C., Aoki, S.T., Zhang, X., Bellamy, A.R., Dormitzer, P.R., Harrison, S.C., Grigorieff, N., 2009. Molecular interactions in rotavirus assembly and uncoating seen by high-resolution cryo-EM. Proc. Natl. Acad. Sci. USA 106, 10644–10648.

Cheng, Y., Zak, O., Aisen, P., Harrison, S.C., Walz, T., 2004. Structure of the human transferrin receptor–transferrin complex. Cell 116, 565–576.

Cong, Y., Baker, M.L., Jakana, J., Woolford, D., Miller, E.J., Reissmann, S., Kumar, R.N., Redding-Johanson, A.M., Batth, T.S., Mukhopadhyay, A., Ludtke, S.J., Frydman, J., Chiu, W., 2010. 4.0-A resolution cryo-EM structure of the mammalian chaperonin TRiC/CCT reveals its unique subunit arrangement. Proc. Natl. Acad. Sci. USA 107, 4967–4972.

Fernandez, J.J., 2008. High performance computing in structural determination by electron cryomicroscopy. J. Struct. Biol. 164, 1–6.

Fotin, A., Kirchhausen, T., Grigorieff, N., Harrison, S.C., Walz, T., Cheng, Y., 2006. Structure determination of clathrin coats to subnanometer resolution by single particle cryo-electron microscopy. J. Struct. Biol. 156, 453–460.

Grigorieff, N., 2007. FREALIGN: high-resolution refinement of single particle structures. J. Struct. Biol. 157, 117–125.

NVIDIA, 2009. NVIDIA CUDA Programming Guide v2.3. http://www.nvidia.com/cuda.

Rabl, J., Smith, D.M., Yu, Y., Chang, S.C., Goldberg, A.L., Cheng, Y., 2008. Mechanism of gate opening in the 20S proteasome by the proteasomal ATPases. Mol. Cell 30, 360–368.

Schmeisser, M., Heisen, B.C., Luettich, M., Busche, B., Hauer, F., Koske, T., Knauber, K.H., Stark, H., 2009. Parallel, distributed and GPU computing technologies in single-particle electron microscopy. Acta Crystallogr. D: Biol. Crystallogr. 65, 659–671.

Tan, G., Guo, Z., Chen, M., Meng, D., 2009. Single-particle 3D reconstruction from cryo-electron microscopy images on GPU. In: International Conference on Supercomputing, Yorktown Heights, New York, USA. pp. 380–389.

Wolf, M., Garcea, R.L., Grigorieff, N., Harrison, S.C., 2010. Subunit interactions in bovine papillomavirus. Proc. Natl. Acad. Sci. USA 107, 6298–6303.

Yang, C., Penczek, P.A., Leith, A., Asturias, F.J., Ng, E.G., Glaeser, R.M., Frank, J., 2007. The parallelization of SPIDER on distributed-memory computers using MPI. J. Struct. Biol. 157, 240–249.

Yu, X., Jin, L., Zhou, Z.H., 2008. 3.88 Å structure of cytoplasmic polyhedrosis virus by cryo-electron microscopy. Nature 453, 415–419.

Zhang, X., Jin, L., Fang, Q., Hui, W.H., Zhou, Z.H., 2010. 3.3 A cryo-EM structure of a nonenveloped virus reveals a priming mechanism for cell entry. Cell 141, 472–482.

Zhang, X., Settembre, E., Xu, C., Dormitzer, P.R., Bellamy, R., Harrison, S.C., Grigorieff, N., 2008. Near-atomic resolution using electron cryomicroscopy and single-particle reconstruction. Proc. Natl. Acad. Sci. USA 105, 1867–1872.

Supplementary Figure 1. Comparison of the results obtained from GPU-enabled FREALIGN and original FREALIGN.

Except for parallelization, the original search and refinement algorithms of FREALIGN were not altered. Thus the results of the GPU-enabled FREALIGN and the original CPU version of FREALIGN were essentially the same when comparing the final 3D reconstructions and FSC curves. (**A**) The FSC curves of two 3D reconstructions after one cycle of search/refinement and 3D reconstruction by the original version of FREALIGN (blue) and the GPU-enabled version of FREALIGN (red). The overlap of the FSC curve indicates that the GPU-enabled FREALIGN produced essentially the same 3D reconstruction as the original FREALIGN calculated on a single CPU. The FSC curve between the two 3D reconstructions is shown in green. It remains above 0.93 at a resolution of 6.5Å and remains above 0.8 to the Nyquist limit. The increasing difference at higher resolution of the 3D reconstruction is due to small differences in the alignment parameters, which are caused by rounding errors. The small differences in the alignment parameters do not introduce noticeable differences in the 3D reconstructions in the resolution range exhibiting significant signal (FSC > 0.143). The two 3D reconstructions are identical (FSC = 1.0 up to the Nyquist limit) when calculated from the same alignment parameters. Two 3D reconstructions calculated by the original version (**B**) and by the GPU-enabled version (**C**) of FREALIGN on System I overlap perfectly.

Supplementary Table 1. Execution time needed to complete each FREALIGN subroutine

Execution time in microseconds on a single i7 CPU core of on System I (Table 1):

size	*cc3m*()	*ccp*()	*pinsert_s*()	*sigma2*()	*presb*()	*ccoef*()
56	266±8	7299±52	490±25	378±8	677±192	445±27
112	1,590±41	28,984±95	2,185±123	1,622±111	2,281±73	1578±54
224	19,323±774	140,208±832	9,754±786	38,897±1199	38,262±1,886	37,146±1,450
500	58,428±960	632,200±1621	31,161±1152	18,1172±3255	19,9695±2,6247	190,976±30,632

Execution time in microseconds on a single GTX 295 GPU processor on System I:

size	*cc3m*()	*ccp*()	*pinsert_s*()	*sigma2*()	*presb*()	*ccoef*()
56	637±142	19,283± 278	138±107	588±202	627±12	532±4
112	808±139	23249±262	406±189	660±55	831±181	641±182
224	2,322±279	33,237±314	1,657±338	3,175±434	3,414±322	3,112±450
500	5,527±85	72,553±314	1,639±150	8,280±338	10,344±297	7,931±354

Each subroutine is invoked at least once for each particle. The averages ± standard deviations were calculated from more than ten different particles of the same size.

Supplementary Table 2. Percentage of the total runtime by FREALIGN subroutines

Procedures and/or subroutines in each mode	Percentage of total runtime *italic*: on CPU **bold**: on GPU			invoked subroutines	percentage of total run time *italic*: on CPU **bold**: on GPU		
IFLAG	0	1	4		0	1	4
Read image/FFT *iread*()/*rlft3*()	*2.3%* **12.0%**	*0.1%* **0.7%**	*0.002%* **0.01%**				
ctfapply()	*17.7%* **12.0%**	*9.0%* **5.5%**	*0.15%* **0.07%**				
sigma2()		*2.2%* **1.6%**	*0.04%* **0.02%**				
search			*99.8%* **99.9%**	*psearch*()			*21.3%* **34.0%**
				prefine()			*78.4%* **65.9%**
				presb()			*0.04%* **0.02%**
refinement		*88.7%* **92.2%**		*prefine*()		*84.4%* **88.9%**	
				presb()		*2.3%* **1.7%**	
				ccoef()		*2.0%* **1.6%**	
reconstruction/ *a3d3*()	*80.0%* **76.0%**						

The percentages of total execution taken up by each major FREALIGN procedure/subroutine were estimated from the real execution times of subroutines divided by the total runtime. Here *psearch*() invokes *ccp*(), *prefine*() invokes *cc3m*(), and 3D reconstruction procedure *a3d3*() invokes *pinsert_s*(). The measurements of CPU execution time were performed with a single CPU core on System I, and the GPU execution time measurements were made on System I with GPU-enabled code running on a single GPU device. The percentages were estimated from more than ten FREALIGN executions in each case.

Supplementary Table 3. Total execution time for search and refinement on a single CPU and a single GPU.

Image size	Search (24 particles)			Refinement (100 particles)		
	1 CPU core	1 GPU processor		1 CPU core	1 GPU processor	
	time (s)	time (s)	acceleration	time (s)	time (s)	acceleration
100	209±4.04	165±2.39	1.27±0.01	14.46±0.45	8.40±0.23	1.72±0.01
150	747±21.76	216±2.19	3.45±0.07	54.64±1.41	12.12±0.22	4.51±0.03
200	1247±32.69	251±4.80	4.97±0.04	88.49±3.06	13.56±0.60	6.52±0.07
250	1914±65.26	340±7.27	5.63±0.07	116.95±1.92	15.89±0.74	7.36±0.22
300	2724±76.96	367±6.41	7.42±0.08	153.31±5.26	17.22±0.33	8.90±0.13
350	4101±48.50	512±11.26	8.01±0.08	204.83±3.86	22.70±0.86	9.02±0.17
400	4852±128.41	547±13.40	8.87±0.02	240.36±6.29	23.91±0.92	10.05±0.12
450	9843±178.02	760±11.50	12.95±0.04	296.64±7.60	29.80±2.04	9.95±0.43
500	7921±155.59	801±15.46	9.89±0.00	330.37±3.11	32.32±1.20	10.22±0.28

Search: The total CPU execution times for processing 24 particles were measured on a single i7 CPU core on System I (no GPU-enabled code). The GPU execution times for processing the same 24 particles were measured on a single GTX 295 GPU on System I, with three GPUs disabled during the tests. **Refinement**: The total CPU execution times for processing 100 particles were measured on a single i7 CPU core on System I (no GPU-enabled code). The GPU execution times for processing the same 100 particles were measured on a single GTX 295 GPU on System I, with three GPUs disabled during the tests. The accelerations factors listed in the table correspond to those shown in Figures 3A and B.

Supplementary Table 4. Total execution time for 3D reconstruction on a single CPU and multiple GPUs.

Image size	Reconstruction (1000 particles)					
	System I			System II		
	1 CPU core	4 GPU processors		1 CPU core	8 GPU processors	
	time (s)	time (s)	acceleration	time (s)	time (s)	acceleration
100	8.97±0.02	0.29±0.01	30.78±1.21	7.61±0.02	0.17±0.01	53.80±3.97
150	30.56±0.02	0.41±0.03	74.10±4.86	26.43±0.02	0.22±0.01	136.15±7.00
200	54.88±1.45	0.54±0.02	101.62±0.91	46.93±0.33	0.30±0.02	181.01±9.66
250	85.43 ±0.15	0.76±0.00	112.77±0.50	73.38±0.11	0.41±0.02	206.58±11.52
300	115.99±0.63	0.98±0.02	118.49±1.33	109.39±7.18	0.56±0.04	207.92±12.83
350	163.03±0.31	1.34±0.01	121.25±0.47	150.44±1.72	0.77±0.04	212.99±12.02
400	207.06±0.41	1.57±0.03	132.15±2.02	191.95±1.25	0.86±0.03	241.56±7.96
450	269.00±0.49	2.12±0.02	126.79±0.88	257.10±18.27	1.16±0.04	232.83±7.96
500	331.15±0.66	2.46±0.04	134.64±1.84	307.31±5.35	1.40±0.10	237.20±15.96

System I: The CPU execution times for calculating 3D reconstructions from a total of 1,000 particles were measured on a single i7 CPU core on System I (no GPU-enabled code). The GPU execution times for processing the same 1,000 particles were measured using 4 GPUs of System I. **System II**: The CPU execution times were measured on a single Xeon X5550 CPU core on the System II (no GPU-enabled code), which is used as the CPU reference for this system. The GPU execution times were measured using 8 GPUs of System II. The accelerations factors of this table correspond to those shown in Figure 3C.

Supplementary Table 5. Acceleration of search, refinement and 3D reconstruction on multiple GPUs.

(A) Search

Search	System I			System II			4GPUs/8GPUs
	1 GPU	4 GPUs		1 GPU	8 GPUs		
Image size	time (s)	time (s)	acceleration	time (s)	time (s)	acceleration	acceleration
100	662±4	190±10	3.48±0.16	682±8	91±3	7.48±0.12	2.08±0.05
150	868±5	236±10	3.68±0.14	938±49	122±4	7.68±0.14	1.93±0.02
200	999±2	270±14	3.70±0.19	1095±14	144±6	7.62±0.20	1.88±0.03
250	1367±2	371±15	3.68±0.15	1495±20	194±12	7.71±0.39	1.91±0.04
300	1473±2	402±13	3.67±0.11	1667±30	216±7	7.71±0.12	1.86±0.01
350	2057±4	570±11	3.61±0.06	2318±44	302 ±16	7.68±0.26	1.89±0.06
400	2180±2	640±14	3.41±0.07	2366±63	316 ±12	7.49±0.08	2.03±0.03
450	3051±4	860±16	3.55±0.06	3346±126	447 ±19	7.48±0.03	1.92±0.05
500	3205±4	894±13	3.59±0.05	3497±142	457 ±21	7.66±0.04	1.96±0.06

(B) Refinement

Refinement	System I			System II			4GPUs/8GPUs
	1 GPU	4 GPUs		1 GPU	8 GPUs		
Image size	time (s)	time (s)	acceleration	time (s)	time (s)	acceleration	acceleration
100	33.6±0.1	9.4±0.7	3.58±0.24	35.5±0.9	6.7±0.3	5.33±0.11	1.41±0.03
150	48.7±0.3	13.6±0.5	3.57±0.09	52.3±0.3	8.4±0.5	6.20±0.35	1.62±0.05
200	54.3±0.3	15.2±0.5	3.57±0.10	59.5±0.3	9.7±0.6	6.11±0.36	1.56±0.05
250	63.6±0.4	18.8±0.8	3.39±0.12	68.4±2.5	10.3±1.1	6.64±0.46	1.82±0.12
300	69.2±0.3	19.4±0.4	3.56±0.06	77.9±1.1	11.5±1.4	6.78±0.73	1.69±0.17
350	90.7±0.4	26.3±0.5	3.45±0.05	100.8±3.3	14.7±1.6	6.86±0.51	1.79±0.16
400	95.8±0.7	28.4±1.7	3.37±0.17	104.8±4.1	16.0±1.4	6.53±0.31	1.77±0.05
450	119.9±0.7	36.0±2.0	3.33±0.17	136.6±1.9	19.1±1.7	7.16±0.55	1.89±0.06
500	130.0±1.0	38.4±1.1	3.39±0.07	140.6±3.4	19.8±1.6	7.09±0.42	1.93±0.11

(C) Reconstruction

Reconstruction	System I 4 GPUs	System II 8 GPUs	4GPUs/8GPUs
Image size	time (s)	time (s)	acceleration
100	0.29±0.01	0.17±0.01	1.75±0.06
150	0.41±0.03	0.22±0.01	1.84±0.03
200	0.54±0.02	0.30±0.02	1.78±0.08
250	0.76±0.00	0.41±0.02	1.83±0.09
300	0.98±0.02	0.56±0.04	1.75±0.09
350	1.34±0.01	0.77±0.04	1.76±0.09
400	1.57±0.03	0.86±0.03	1.83±0.03
450	2.12±0.02	1.16±0.04	1.84±0.05
500	2.46±0.04	1.40±0.10	1.76±0.09

(A) Search: The total GPU execution times in second for completing an orientational search for 96 particles on 1 and 4 GPUs on System I, and on 1 and 8 GPUs on System II. **(B) Refinement**: The GPU execution times for completing parameter refinement for 400 particles on 1 and 4 GPUs on System I, and

on 1 and 8 GPUs on System II. **(C) Reconstruction:** The GPU execution time for 3D reconstruction with 1,000 particles on 4 GPUs (System I) and 8 GPUs (System II). The last column is the ratio between the execution times of 4 and 8 GPUs.

Note: In all tables, the execution times are shown as the average ± standard deviation calculated from multiple FREALIGN executions with the same data. The standard deviation of CPU and GPU execution time, $\sigma_{CPUTime}$ and $\sigma_{GPUTime}$, were calculated from these multiple time measurements. The standard deviation of the speed enhancement, σ_{accel}, is calculated using:

$$d(\frac{x}{y}) = \frac{1}{y}dx - \frac{x}{y^2}dy$$

,

and is

$$\sigma_{accel} = \frac{1}{GPUTime}\sigma_{CPUTime} - \frac{CPUTime}{GPUTime^2}\sigma_{GPUTime}$$