Ultramicroscopy 115 (2012) 109–114

Contents lists available at SciVerse ScienceDirect

Ultramicroscopy

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

Hybrid computing: CPU+GPU co-processing and its application to tomographic reconstruction

J.I. Agulleiro [a,*], F. Vázquez [a,*], E.M. Garzón [a], J.J. Fernández [b,**]

[a] Supercomputing and Algorithms Group, Associated Unit CSIC-UAL, University of Almería, 04120 Almería, Spain
[b] National Centre for Biotechnology, National Research Council (CNB-CSIC), Campus UAM, C/Darwin 3, Cantoblanco, 28049 Madrid, Spain

ARTICLE INFO

Article history:
Received 13 December 2011
Received in revised form
7 February 2012
Accepted 11 February 2012
Available online 18 February 2012

Keywords:
CPU
GPU
Hybrid computing
CPU–GPU co-processing
High performance computing
Electron tomography
Tomographic reconstruction

ABSTRACT

Modern computers are equipped with powerful computing engines like multicore processors and GPUs. The 3DEM community has rapidly adapted to this scenario and many software packages now make use of high performance computing techniques to exploit these devices. However, the implementations thus far are purely focused on either GPUs or CPUs. This work presents a hybrid approach that collaboratively combines the GPUs and CPUs available in a computer and applies it to the problem of tomographic reconstruction. Proper orchestration of workload in such a heterogeneous system is an issue. Here we use an on-demand strategy whereby the computing devices request a new piece of work to do when idle. Our hybrid approach thus takes advantage of the whole computing power available in modern computers and further reduces the processing time. This CPU+GPU co-processing can be readily extended to other image processing tasks in 3DEM.

1. Introduction

High performance computing (HPC) plays a major role in three-dimensional electron microscopy (3DEM) as it allows successfully dealing with the increasing complexity of algorithms and the amount of data needed to push the resolution limits [1]. Different types of HPC platforms have historically been used in 3DEM [1]. In the last few years, graphics processing units (GPUs) have shaken up the HPC field because of their tremendously powerful computing power at an incomparable performance-to-cost ratio [2]. The 3DEM community has rapidly adopted them and many methods and codes have been ported to take advantage of these devices [e.g. 3–5]. In addition, the advent of multicore processors is providing standard computers with awesome computing power, easily exploitable by means of classical HPC strategies [6], which is approaching GPU performance [7]. Some works in 3DEM have already pointed out the importance of these platforms [8,9].

The problem of 3D reconstruction in electron tomography (ET) has long been studied from the HPC perspective with great interest. The main reason is the fact that iterative reconstruction methods (e.g. SIRT [10]) provide better solutions under noisy and low contrast conditions [11] but at the expense of significant processing time, which has prevented their extensive use. Traditionally, there have been HPC proposals for tomographic reconstruction for supercomputers [12], computational grids [13–15] or clusters [16–19]. The trend has turned toward GPUs, and a number of proposals and improvements have lately been presented in the field [3,20–24], including the use of a multi-GPU approach [25,26]. Recently, our group has shown that current multicore computers may exhibit a performance similar to, or even better than, GPU solutions when code optimization, vector processing and multithreading are exploited [9,27].

This work presents a hybrid computing approach that collaboratively combines the CPUs and GPUs available in a computer, and we have applied it to tomographic reconstruction. This strategy is motivated by the performance of our multicore approach [9,27], which showed that a multicore computer and a GPU might be considered as peers. Combining these two computing engines could thus harness and make the most of all computing power latent in standard desktop computers or servers.

2. Hybrid CPU+GPU approach in tomographic reconstruction

CPU cores and GPUs present significant differences in their internal architecture, exploit different levels of parallelism and work at a different pace [6]. The architecture of a system

* Co-first authors with equal contribution.
** Corresponding author. Tel.: +34 91 585 4619; fax: +34 91 585 4506.
E-mail address: JJ.Fernandez@csic.es (J.J. Fernández).

0304-3991/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.ultramic.2012.02.003

comprising multicore processors and GPUs is thus heterogeneous. Then, proper distribution of the workload among the various processing elements is an issue, if maximum exploitation is intended [28,29]. Therefore, there is a need for an efficient mechanism to orchestrate the workload in order to maximize the use of the processing elements (CPU cores and GPUs) while minimizing latencies and waiting times. In the HPC field, load balancing has been a subject of research for long [28]. One dynamic load balancing technique that has turned out to be effective consists in splitting the problem into tasks, which are assigned to the processing elements as they become idle (hence 'dynamic') [28]. In our hybrid CPU+GPU approach, we use this strategy to dynamically distribute the workload among the CPU cores and the GPUs.

The single-tilt axis geometry typically used in ET allows decomposition of the 3D reconstruction problem into a set of independent 2D reconstruction sub-problems corresponding to the slices perpendicular to the tilt axis. This data decomposition has often been used in the field for 3D reconstruction with HPC techniques [e.g. 1,9,12,13,17]. The independent 2D slices of the volume can then be computed in parallel by the different processing elements using any reconstruction method (WBP, SIRT, etc.). To ameliorate latencies and overheads, the slices are usually packaged into slabs [15]. In particular, in our multicore approach, and in this work, the slices are clustered in slabs of four slices [9]. This is because vector instructions (SSE, Streaming SIMD Extensions) in the CPU cores are exploited to reconstruct four slices simultaneously [27].

Our hybrid approach to tomographic reconstruction then works as follows. A pool of slabs to reconstruct (each slab containing four slices) is maintained. A number of threads run concurrently in the system at the CPU level (so-called C-threads). In addition, for each GPU in the system a specific thread (so-called G-threads) is created, which will be in charge of sending out the projection data (sinograms) to the GPU and receiving the reconstructed slices with the CUDA function cudaMemcpy(). As the threads become idle, they are assigned a new slab to reconstruct. When a slab is allotted to a C-thread, the calculations will be performed on one of the CPU cores, and the four slices in the slab will be computed simultaneously thanks to the vector instructions. In the case of the G-threads, the calculations will be carried out on the GPU using fine-grain parallelism at the level of the voxel. When a thread (either C- or G-) finishes its work, it requests another slab to reconstruct. This asynchronous dispatching of workload on-demand is the implementation of the dynamic load balancing technique referred to above. As the GPU normally makes the computations much faster than a single CPU core, a G-thread will be allotted more slabs than a C-thread. Fig. 1 shows an illustrative scheme of our hybrid CPU+GPU approach to tomographic reconstruction.

3. Results

To evaluate our hybrid implementation, we have chosen two different platforms. The first one is an eight-core machine (two quad-core processors Intel Xeon E5640 at 2.66 GHz) equipped with two GPUs Nvidia Tesla C2050, each GPU with 448 cores at 1.15 GHz and 2.6 GB memory. The second platform is a quad-core machine (based on an Intel Xeon W3520 at 2.66 GHz) and a GPU Nvidia GTX285 with 240 cores at 1.4 GHz and 2 GB memory. The two most common ET reconstruction methods, WBP and SIRT

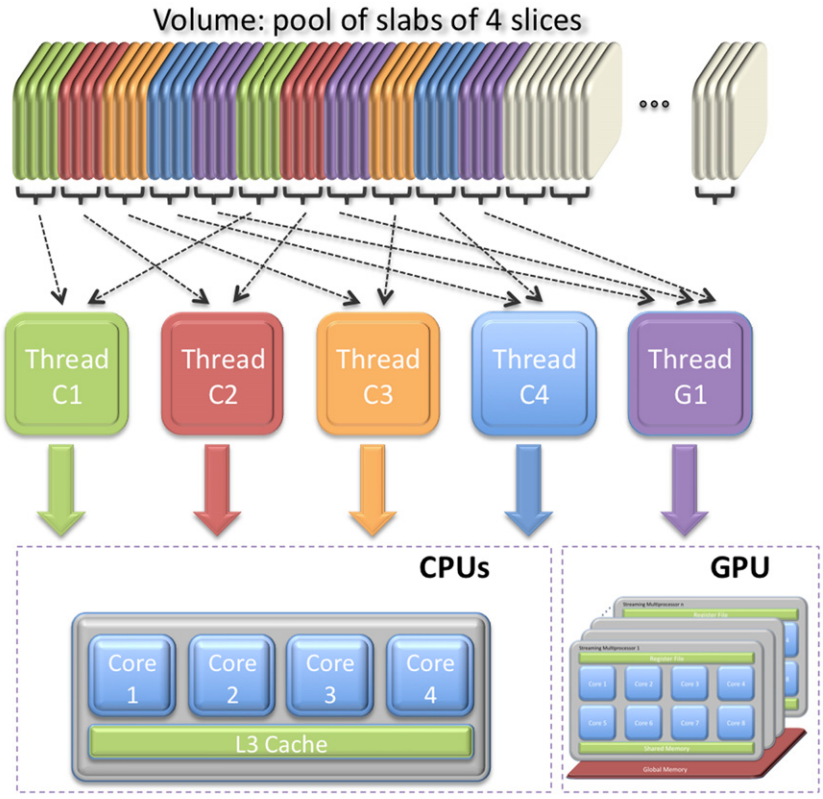

Fig. 1. Hybrid CPU+GPU approach to tomographic reconstruction. The system keeps a pool of slabs of four slices to be reconstructed. A number of threads to be mapped to CPU cores (denoted by C-threads) are running concurrently in the system. Also, specific threads (denoted by G-threads) in charge of the tasks to be computed on the GPUs are also running. The slabs are asynchronously dispatched to the threads on-demand. In the figure, allocation of slabs to threads are color-coded. Note that the G-threads will request tasks more often than the C-threads as GPUs make the calculations faster than a single CPU core. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

(30 iterations), have been tested. For the GPU, we have chosen the matrix approach of these methods as it proves to be one of the fastest implementations so far [22,23]. For the CPU, our fast multithreaded vectorized and optimized implementation has been used [9,27]. We have selected three representative sizes for the datasets: tilt-series of 122 images of 1024×1024, 2048×2048, and 4096×4096 to yield reconstructions of the same size in X and Y, and thickness 256. In the particular case of the quad-core machine, the GPU did not have enough memory to accommodate the matrix requirements for the largest (4096) case, so a volume of $4096 \times 4096 \times 128$ was reconstructed instead. The experiments were executed five times and the average time was selected for analysis.

For the evaluation, several types of experiments were carried out and reported here: (1) the pure multicore approach using all cores available in the platform, (2) the pure multiGPU/GPU strategy exploiting all the GPUs in the platform, (3) the hybrid approach launching threads for all the CPU cores and all GPUs (i.e. eight C-threads and two G-threads in Platform 1 – denoted by 8C/2G – and 4C/1G in Platform 2) and (4) the hybrid approach launching G-threads for all the GPUs but keeping the total sum of threads no higher than the CPU-cores available (i.e. 6C/2G in Platform 1 and 3C/1G in Platform 2). Additional experiments were conducted in this study. For brevity, these results are not included here because with less C-threads than the configurations tested here, the performance was poorer. In the case of Platform 1, we also tested configurations involving only one GPU (i.e. 8C/1G, 7C/1G, 0C/1G), which showed the same behavior as that reported here.

Tables 1 and 2 summarize the results obtained for Platform 1 with WBP and SIRT, respectively, whereas Tables 3 and 4 refer to the second platform. In the tables, the processing time required for the reconstruction as well as the distribution of workload (i.e. how many slices have been reconstructed by the CPUs and by the

Table 1
Processing time and distribution of workload—WBP on Platform 1 (8 CPUs, 2 GPUs).

Configuration	1024		2048		4096	
(C-/G-threads)	T.Rec. (s)	Slices (%) CPU/GPU	T.Rec. (s)	Slices (%) CPU/GPU	T.Rec. (s)	Slices (%) CPU/GPU
Pure CPU (8C)	2.80	100.0/- - - -	11.67	100.0/- - - -	51.84	100.0/- - - -
MultiGPU (2G)	2.35	- - - -/100.0	9.43	- - - -/100.0	41.14	- - - -/100.0
Hybrid (8C/2G)	1.93	58.40/41.60	7.17	54.00/46.00	30.30	52.10/47.90
Hybrid (6C/2G)	1.73	44.63/55.37	6.43	41.85/58.15	28.27	41.50/58.50

Table 2
Processing time and distribution of workload—SIRT on Platform 1 (8 CPUs, 2 GPUs).

Configuration	1024		2048		4096	
(C-/G-threads)	T.Rec. (s)	Slices (%) CPU/GPU	T.Rec. (s)	Slices (%) CPU/GPU	T.Rec. (s)	Slices (%) CPU/GPU
Pure CPU (8C)	147.14	100.0/- - - -	617.84	100.0/- - - -	2662.53	100.0/- - - -
MultiGPU (2G)	153.92	- - - -/100.0	625.64	- - - -/100.0	3009.91	- - - -/100.0
Hybrid (8C/2G)	95.77	52.25/47.75	390.11	52.30/47.70	1748.03	54.93/45.07
Hybrid (6C/2G)	91.45	46.88/53.12	372.54	46.88/53.12	1602.54	47.19/52.81

Table 3
Processing time and distribution of workload—WBP on Platform 2 (4 CPUs, 1 GPU).

Configuration	1024		2048		4096	
(C-/G-threads)	T.Rec. (s)	Slices (%) CPU/GPU	T.Rec. (s)	Slices (%) CPU/GPU	T.Rec. (s)	Slices (%) CPU/GPU
Pure CPU (4C)	5.06	100.0/- - - -	21.74	100.0/- - - -	48.14	100.0/- - - -
Pure GPU (1G)	7.77	- - - -/100.0	29.50	- - - -/100.0	73.97	- - - -/100.0
Hybrid (4C/1G)	3.77	60.55/39.45	15.32	60.35/39.65	35.80	63.77/36.23
Hybrid (3C/1G)	3.75	56.64/43.36	14.84	53.13/46.87	33.38	56.25/43.75

Table 4
Processing time and distribution of workload—SIRT on Platform 2 (4 CPUs, 1 GPU).

Configuration	1024		2048		4096	
(C-/G-threads)	T.Rec. (s)	Slices (%) CPU/GPU	T.Rec. (s)	Slices (%) CPU/GPU	T.Rec. (s)	Slices (%) CPU/GPU
Pure CPU (4C)	283.85	100.0/- - - -	1187.35	100.0/- - - -	2499.48	100.0/- - - -
Pure GPU (1G)	416.41	- - - -/100.0	1621.24	- - - -/100.0	3195.90	- - - -/100.0
Hybrid (4C/1G)	213.27	60.94/39.06	845.07	59.18/40.82	1715.80	58.40/41.60
Hybrid (3C/1G)	205.40	53.12/46.88	792.88	51.56/48.44	1617.22	51.56/48.44

GPUs) are presented. The latter is expressed in percentage to allow easy comparison between the different dataset sizes and configurations.

The hybrid approach (any of the two configurations) clearly outperforms the pure CPU or pure GPUs strategies in terms of processing time, regardless of the platform and reconstruction method, as shown by Tables 1–4. Some processing times are especially remarkable: a SIRT reconstruction of a 2048 dataset, which is currently a standard case in the ET field, only requires 6 and 13 min in Platforms 1 and 2, respectively. Interestingly, all those results point out that the second configuration of the hybrid approach is better than the first one. This means that, for maximum performance, it is important to keep the total number of threads running in the system no higher than the number of CPU-cores in the platform. In order to analyze the net gain of the hybrid approach with regard to the standard pure CPU or GPU strategies, we computed the average speed-up factor taking into account all the dataset sizes and reconstruction methods. Fig. 2(left) shows that, compared to the pure multicore implementation, the acceleration factor exhibited by the hybrid approach is around 1.5. When compared to the multiGPU (Platform 1) or GPU (Platform 2) strategy, the speed-up factor ranges from 1.5 to 2.0 (Fig. 2(right)). Finally, restricting the total number of threads to the number of CPU-cores involves a further improvement in the speed-up factor that amounts to 5–8% as shown in Fig. 2.

As far as distribution of workload is concerned, from the data in Tables 1–4 the global averages were computed and charted in Fig. 3. It can be observed that, in Platform 1 and the first hybrid configuration (8C/2G), the CPU-cores are in charge of around 55% of the slices whereas the remaining 45% are for the two GPUs. In Platform 2 using 4C/1G, the CPUs reconstruct around 60% of the slices while the GPU gets 40%. When the total number of threads is restricted (second hybrid configuration), a re-distribution of the workload toward the GPUs is seen (around 45% CPU, 55% GPUs in Platform 1; around 55% CPU, 45% GPU in Platform 2). That is, each GPU gets around 5% more work than with the first configuration. Fig. 4 plots the average workload per thread. It proves that a G-thread processes substantially more slices than a single C-thread. Interestingly, it can be observed that in the second hybrid configuration not only the G-threads get more workload, but the C-threads actually increase their burden as well, though at lower rate (around 1% and 3% in the two platforms, respectively, compared to the 5% of the G-threads).

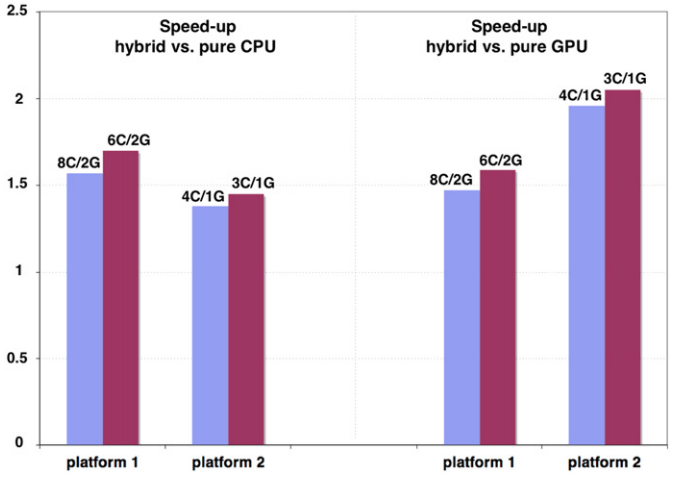

Fig. 2. Speed-up factors of the hybrid approach compared to the pure multicore strategy (left) and pure multiGPU/GPU strategy (right). The speed-up with the multicore scheme is around 1.5, with the multiGPU is approaching 1.5, and with the GPU in Platform 2 increases up to 2.

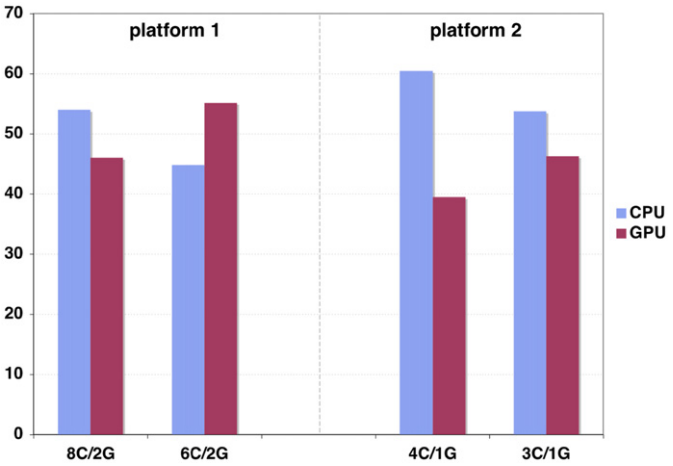

Fig. 3. Distribution of workload in the hybrid approach in the two platforms. The percentage of slices reconstructed by the CPU cores and GPUs are shown under different configurations of C- and G-threads.

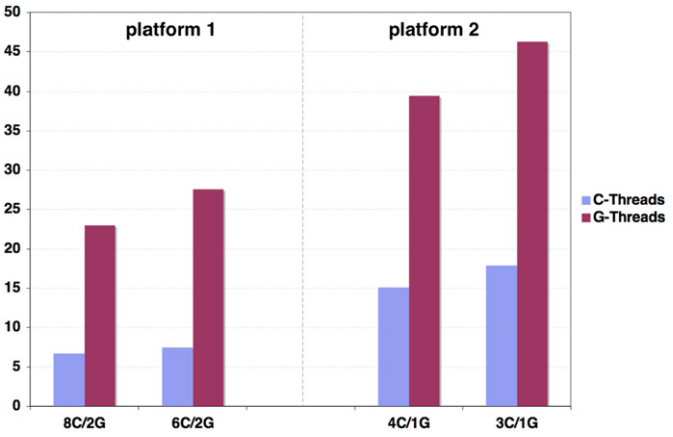

Fig. 4. Distribution of workload per thread (in percentage) in the hybrid approach in the two platforms for different configurations. As seen, the workload allotted to the G-threads are much higher than that to the C-threads since the GPUs are much more powerful than a single CPU core.

Finally, we wanted to compare our hybrid approach with a recent powerful distributed multiGPU scheme [25]. This is another hybrid strategy working at a higher level, which exploits several GPUs distributed across different computers. Thus, we performed a set of experiments consisting of 10 iterations of SIRT with the three datasets, and measuring the reconstruction time and the wall-time (i.e. the total time including I/O and all other system overhead). We restricted ourselves to our most powerful machine, Platform 1, and the best configuration 6C/2G. Table 5 shows the results we obtained. The times reported by [25] with 10 GPUs distributed across three computers with datasets with the same size are also reproduced. In comparison to that work, our approach turns out to be very well positioned. In particular, the performance is exceptionally good for large sizes. There the time for reconstruction (T.Rec.) is lower, but the point that really makes difference is the system overhead. For a reconstruction of $4096 \times 4096 \times 256$ from 122 images, the scheme by [25] spends 10 min in system overhead, whereas our approach reduces this time to less than 2 min. Obviously, the transmission of data and reconstructed slabs through the network dramatically penalizes the distributed GPU scheme. In our hybrid approach, however, all I/O operations remain at a local level.

Table 5
Comparison of our hybrid approach with that by [25].

	1024		2048		4096	
	Zheng et al. [25]	Our approach	Zheng et al. [25]	Our approach	Zheng et al. [25]	Our approach
Total time (s)	38.9	50.43	194.5	164.54	1422.2	682.70
T.Rec. (s)	22.5	38.96	129.1	130.50	821.2	568.15
Sys. overhead	16.4	11.47	65.4	34.04	601.0	114.55

4. Discussion and conclusion

This work presents a hybrid computing approach, combining GPUs and multicore processors, to fully take advantage of the computing power latent in modern computers. It also presents its application to the problem of tomographic reconstruction. One inherent characteristic of these modern platforms is their heterogeneity, which raises the issue of distribution of the workload among the different processing elements. Adaptive load balancing techniques are thus necessary to properly adjust the amount of work to be done by each computing element. Here we have chosen the 'on-demand' strategy, a well-known technique in the HPC field, by which the different elements asynchronously request a piece of work to do when they become idle, thereby keeping the system fairly well balanced.

Our hybrid approach improves the performance by a factor in the range $1.5\times$ to $2\times$ with regard to the standard strategies based on pure CPU, GPU or multiGPUs. Although these figures are relative to the systems and implementations that were tested here, they are useful to illustrate the potential of the technique. In platforms with a higher number of GPUs, more powerful GPUs, or with better GPU implementations, our hybrid approach will be capable of adapting to the system by assigning more work to the GPUs. On the contrary, if modest GPUs or GPU implementations are used, it will allot more work to the multiple CPU cores in the system. Furthermore, this dynamic load balancing technique may readily accommodate to a variety of heterogeneous platforms, for instance when several GPUs with different computing capabilities are connected to a multicore computer.

The results we have obtained clearly point out that, for the sake of performance, it is important to limit the total number of threads (the sum of C-threads and G-threads) running in the system to the number of CPU-cores, regardless of the number of GPUs available. This limitation involves a further reduction of the processing time. This point is caused by the fact that, even though the work allotted to the G-threads is to be done on the GPUs, there is still some work to do by the CPU, e.g. the transfer of data and the results between CPU and GPU. For that reason, G-threads actually compete with the C-threads. Limitation of the total number of threads to the CPU cores available minimizes such competition. The use of more sophisticated strategies to reduce communication latencies [26] would also help.

Our hybrid approach fully exploits the processing resources available in modern computers at a local level. When compared to other complex hybrid systems, like the recent distributed multi-GPU strategy [25], our approach is well positioned. One of the advantages of the system configured by [25] is the scalability, in the sense that additional nodes with more GPUs are easily added to the system so as to cope with increasing computational demands. Nevertheless, the high network overhead dramatically reduces its potential scalability. On the contrary, working at a local level like in this hybrid approach, the system overhead is remarkably reduced. The two strategies are not incompatible at all. A hierarchical hybrid system based on distributed nodes, which took advantage of the multiple CPU cores and multiple GPUs within the nodes, could maximize the computational resources while ameliorating network overhead.

The application of hybrid computing to jointly exploit CPUs and GPUs in different image processing tasks involved in the 3DEM disciplines is expected to be straightforward as happened with GPU computing thus far. In a nice recent review, the paramount role that hybrid approaches (i.e. integrative use of different methodologies) play in structural biology was highlighted [30]. As far as exploitation of computing technologies is concerned, the future looks hybrid as well.

The program for tomographic reconstruction using CPU+GPU co-processing is freely available through the following web site: http://www.cnb.csic.es/~jjfernandez/tomo3dhybrid. We have also prepared code to help developers implement this hybrid approach or adapt it to their problems, and is available at http://www.cnb.csic.es/~jjfernandez/hybridcomputing.

Acknowledgments

Work partially supported by the Spanish Ministry of Science (TIN2008-01117) and J. Andalucia (P10-TIC-6002, P11-TIC-7176), in part financed by the European Reg. Dev. Fund (ERDF).

References

[1] J.J. Fernandez, High performance computing in structural determination by electron cryomicroscopy, Journal of Structural Biology 164 (2008) 1–6.
[2] D. Kirk, W.M. Hwu, Programming Massively Parallel Processors: A Hands-on Approach, Morgan Kaufmann, 2010.
[3] D. Castano-Diez, D. Moser, A. Schoenegger, S. Pruggnaller, A.S. Frangakis, Performance evaluation of image processing algorithms on the GPU, Journal of Structural Biology 164 (2008) 153–160.
[4] X. Li, N. Grigorieff, Y. Cheng, GPU-enabled FREALIGN: accelerating single particle 3D reconstruction and refinement in Fourier space on graphics processors, Journal of Structural Biology 172 (2010) 407–412.
[5] X. Zhang, X. Zhang, Z.H. Zhou, Low cost, high performance GPU computing solution for atomic resolution cryoEM single-particle reconstruction, Journal of Structural Biology 172 (2010) 400–406.
[6] J.L. Hennessy, D.A. Patterson, Computer Architecture: A Quantitative Approach, fifth ed., Morgan Kaufmann, 2011.
[7] V.W. Lee, C. Kim, J. Chhugani, M. Deisher, D. Kim, A.D. Nguyen, N. Satish, M. Smelyanskiy, S. Chennupaty, P. Hammarlund, R. Singhal, P. Dubey, Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU, SIGARCH Computer Architecture News 38 (3) (2010) 451–460.
[8] J.R. Bilbao-Castro, R. Marabini, C.O.S. Sorzano, I. Garcia, J.M. Carazo, J.J. Fernandez, Exploiting desktop supercomputing for three-dimensional electron microscopy reconstructions using ART with blobs, Journal of Structural Biology 165 (2009) 19–26.
[9] J.I. Agulleiro, J.J. Fernandez, Fast tomographic reconstruction on multicore computers, Bioinformatics 27 (2011) 582–583.
[10] P. Gilbert, Iterative methods for the 3D reconstruction of an object from projections, Journal of Theoretical Biology 76 (1972) 105–117.
[11] J.J. Fernandez, A.F. Lawrence, J. Roca, I. Garcia, M.H. Ellisman, J.M. Carazo, High performance electron tomography of complex biological specimens, Journal of Structural Biology 138 (2002) 6–20.
[12] G.A. Perkins, C.W. Renken, J.Y. Song, T.G. Frey, S.J. Young, S. Lamont, M.E. Martone, S. Lindsey, M.H. Ellisman, Electron tomography of large, multicomponent biological structures, Journal of Structural Biology 120 (1997) 219–227.
[13] S.T. Peltier, A.W. Lin, D. Lee, S. Mock, S. Lamont, T. Molina, M. Wong, L. Dai, M.E. Martone, M.H. Ellisman, The Telescience portal for tomography applications, Journal of Parallel and Distributed Computing 63 (2003) 539–550.
[14] D. Lee, A.W. Lin, T. Hutton, T. Akiyama, S. Shinji, F.P. Lin, S. Peltier, M.H. Ellisman, Global Telescience featuring IPv6 at iGrid2002, Future Generation Computer Systems 19 (2003) 1031–1039.

[15] J.J. Fernandez, I. Garcia, J.M. Carazo, R. Marabini, Electron tomography of complex biological specimens on the grid, Future Generation Computer Systems 23 (2007) 435–446.
[16] J.J. Fernandez, J.M. Carazo, I. Garcia, Three-dimensional reconstruction of cellular structures by electron microscope tomography and parallel computing, Journal of Parallel and Distributed Computing 64 (2004) 285–300.
[17] S.Q. Zheng, B. Keszthelyi, E. Branlund, J.M. Lyle, M.B. Braunfeld, J.W. Sedat, D.A. Agard, UCSF tomography: an integrated software suite for real-time electron microscopic tomographic data collection, alignment, and reconstruction, Journal of Structural Biology 157 (2007) 138–147.
[18] J.J. Fernandez, D. Gordon, R. Gordon, Efficient parallel implementation of iterative reconstruction algorithms for electron tomography, Journal of Parallel and Distributed Computing 68 (2008) 626–640.
[19] X. Wan, F. Zhang, Z. Liu, Modified simultaneous algebraic reconstruction technique and its parallelization in cryo-electron tomography, in: Proceedings of the 15th International Conference on Parallel and Distributed Systems, IEEE, Washington, DC, USA, 2009, pp. 384–390.
[20] D. Castano-Diez, H. Mueller, A.S. Frangakis, Implementation and performance evaluation of reconstruction algorithms on graphics processors, Journal of Structural Biology 157 (2007) 288–295.
[21] W. Xu, F. Xu, M. Jones, B. Keszthelyi, J. Sedat, D. Agard, K. Mueller, High-performance iterative electron tomography reconstruction with long-object compensation using graphics processing units (GPUs), Journal of Structural Biology 171 (2010) 142–153.
[22] F. Vazquez, E.M. Garzon, J.J. Fernandez, A matrix approach to tomographic reconstruction and its implementation on GPUs, Journal of Structural Biology 170 (2010) 146–151.
[23] F. Vazquez, E.M. Garzon, J.J. Fernandez, Matrix implementation of simultaneous iterative reconstruction technique (SIRT) on GPUs, The Computer Journal 54 (2011) 1861–1868.
[24] W.J. Palenstijn, K.J. Batenburg, J. Sijbers, Performance improvements for iterative electron tomography reconstruction using graphics processing units (GPUs), Journal of Structural Biology 176 (2011) 250–253.
[25] S.Q. Zheng, E. Branlund, B. Kesthelyi, M.B. Braunfeld, Y. Cheng, J.W. Sedat, D.A. Agard, A distributed multi-GPU system for high speed electron microscopic tomographic reconstruction, Ultramicroscopy 111 (2011) 1137–1143.
[26] X. Wan, F. Zhang, Q. Chu, Z. Liu, High-performance blob-based iterative reconstruction of electron tomography on multi-GPUs, Lecture Notes in Computer Science, vol. 6674, , 2011, pp. 61–72.
[27] J.I. Agulleiro, E.M. Garzon, I. Garcia, J.J. Fernandez, Vectorization with SIMD extensions speeds up reconstruction in electron tomography, Journal of Structural Biology 170 (2010) 570–575.
[28] B. Wilkinson, M. Allen, Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers, second ed., Prentice Hall, 2005.
[29] L. Fan, F. Zhang, G. Wang, Z. Liu, An effective scheduling algorithm for linear makespan minimization on unrelated parallel machines, in: Proceedings of the IEEE International Conference on High Performance Computing (HiPC 2009), 2009, pp. 40–49.
[30] A.C. Steven, W. Baumeister, The future is hybrid, Journal of Structural Biology 163 (2008) 186–195.