Grid Computing in 3D-EM Image Processing using Xmipp

S.H.W. Scheres[1]*, A.J. Merino[1]*, C.O.S. Sorzano[1,2], J.M. Carazo[1,3]
[1]National Center for Biotechnology, Campus Univ. Autónoma, 28049, Madrid, Spain
[2]Escuela Politécnica Superior, Univ. San Pablo-CEU, 28668, Madrid, Spain
[3]Escuela Politécnica Superior, Univ. Autónoma de Madrid, 28049, Madrid, Spain
E-mail: carazo@cnb.uam.es
** These authors contributed equally to this work*

Abstract

Image processing in three-dimensional electron microscopy (3D-EM) is characterized by large amounts of data, and voluminous computing requirements. Here, we report our first experience with grid computing in this area. We present an interface between grid computing middleware and our image processing package Xmipp. The efficacy of this approach was illustrated with an Xmipp application for estimation of the contrast transfer function. In addition, we report our experience with grid computing in the development of a novel image refinement algorithm based on maximum likelihood principles. Its extensive CPU-requirements might have seriously hampered the algorithm development, if not for the far-reaching resources of grid computing. Our results suggest that electron microscopy image processing may be particularly well suited for grid computing.

1. Introduction

Three-dimensional electron microscopy (3D-EM) may yield valuable information about the 3D structure of large macro-molecular complexes at medium-low resolution. To increase the low signal-to-noise ratios, large numbers of experimental images need to be averaged. Nowadays, typically (ten) thousands of images are recorded, while future studies at even higher resolutions may routinely comprehend hundreds of thousands of images. Since many years, these large quantities of experimental data have tightly linked 3D-EM image processing to high-performance computing. Parallel computing was introduced in the field well over a decade ago [1, 2], and nowadays cluster computing plays an important role in 3D-EM data processing. For electron tomography, which studies larger biological specimens in the sub-cellular range, grid computing has already been reported [3, 4]. But, so far the use of grids in 3D-EM has been limited.

Over the past eight years, our group has been developing a program package for single-particle 3D-EM image processing called Xmipp [5]. Whereas up to now its approach to high-performance computing has been based on MPI (message passing interface) implementations of the most time-consuming algorithms, current efforts have also addressed grid computing. In this paper, we report our initial experiences with grid computing using Xmipp. First, we present our approach to provide the experimentalist with a familiar interface to run standard Xmipp applications on these novel computing resources. Second, we present how grid computing speeded up considerably the development of a novel image processing algorithm and how parallelization using grid computing may greatly improve its practical use.

2. An Xmipp-interface to the grid: CTF-estimation as a test case

The main goal of this part of our work is to implement an interface that allows the experimentalist to run standard Xmipp applications on grid computing resources, with minimum changes in the user-interaction. For this purpose, we created a specific (C^{++}-) class in Xmipp that serves as an interface to the grid. Grid-related data transfer (publication, replication, etc.) as well as job management (submission, status report etc.) are handled using LCG2-middleware tools, which among others are based on Globus GT2 and Condor. The new Xmipp class receives information regarding the input data for the task at hand, the corresponding Xmipp executable, and the resulting output data. With this information, the relevant LCG2-middleware job-description language (JDL) file and three shell scripts that make calls to the LCG2 tools are created. Upon calling the Xmipp program on the user front-end (the local machine), a daemon is launched. This daemon locally executes the first script, which publishes the input data and the (statically compiled, *i.e.* stand-alone) Xmipp executable in the grid. Subsequently, the daemon submits the JDL-file to the grid resource broker machine, also using standard LCG2-middleware tools. Upon arrival at the working node (the remote machine), the second script is executed. This script takes care of retrieving the input data and the Xmipp executable on the remote machine, executing the program and publishing the resulting output data. Meanwhile, at the local machine the daemon checks for the proper execution of the job at regular intervals. If the job is aborted, the daemon re-submits it, until successful completion of the job. Then, the third script is executed on the local machine, retrieving the output data and deleting all temporary files, including those published in the grid. In this way, all the overhead related to job submission to the grid as well as input/output data transport is handled automatically. From the user perspective, the Xmipp application is invoked in the same way as its standard implementation, but for an extra flag indicating that the application should be run on the grid.

We tested this approach with a routine Xmipp application that requires relatively large amounts of computing time: estimation of the contrast transfer function (CTF) from digitized electron micrographs. The electron microscope introduces aberrations in the imaging process that can be described by this CTF. If high-quality reconstructions are to be obtained, these aberrations need to be estimated (and subsequently corrected) for all micrographs. Since the estimation is independent for each of the typically tens to hundreds of recorded micrographs, this task is particularly well-suited for parallelization using grid computing. Each estimation for a test set comprising 100 micrographs required approximately 8 hours of CPU on a single (Intel 2.6 GHz) processor. Consequently, for this data set the total computation time on one machine would amount to more than a month. Automatically launching all processes to the EGEE-grid, the entire calculation could be performed overnight.

3. Grid computing at the development and application stages of a novel algorithm for image refinement.

Our ongoing developments of novel algorithms require ever-increasing computing resources. A recent example is maximum-likelihood (ML) image refinement [6]. This novel algorithm for alignment of structurally heterogeneous data sets requires extensive amounts of CPU time, which might have seriously limited its development, if not for the availability of grid computing resources. The tests that were used to develop protocols and to optimize parameter values for this algorithm comprised different types of data and were repeated multiple times in order to obtain statistically relevant results. Whereas these tests would require one month on a single CPU, they could be performed overnight using EGEE-grid computing resources. For the entire development of the ML algorithm, we estimate to have submitted more than 1,000 jobs to the EGEE-grid during a period of less than two months, using in total almost two years of CPU-time! Clearly, calculations of this size would have seriously limited the speed of our investigations, if not for the resources provided by grid computing.

For normally sized experimental data sets, the extensive amounts of CPU time required may limit the practical use of ML image refinement. Therefore, we explored the possibility of (a coarse-grain) parallelization by separately processing multiple subsets of the experimental images on different nodes in the EGEE-grid. Although within a single iteration image subsets can be processed independently, the ML algorithm requires that the results of all subsets should be combined to provide the starting point for the next iteration. We developed shell scripts to automatically submit calculations for all image subsets and to retrieve and combine the results after each iteration. In this way, using 12 processors of our local Alpha-cluster we were capable of speeding up the calculations almost 12-fold, *i.e.* we obtained a parallelization efficiency of almost 100%. The same protocol was then tested on the EGEE-grid, using 25 CPUs. In this case, parallelization efficiencies of up to 60% were obtained. This decrease in parallelization efficiency is explained by the fact that every iteration required (relatively slow) submission and data retrieval for 25 separate jobs. Still, the refinement of an experimental data set requiring almost two months on a single CPU could be performed in approximately 4 days. These efficiencies may be further improved using a (finer-grain) MPI-implementation of the algorithm, which could be run on multiple processors of a single site in the grid, thus eliminating the need of submitting and retrieving the results of so many separate jobs in the grid.

4. Discussion

Our results illustrate that 3D-EM image processing may greatly benefit from the resources offered by grid computing. This holds both for submission of routine calculations by experimentalists, as for the development of new algorithms by software developers. Therefore, grid computing may become an important technological step forward in the history of high-performance computing by the 3D-EM community.

5. Acknowledgements

We acknowledge the EGEE/NA4 biomedical applications working group for providing grid-computing resources. Funding came from the Spanish Comisión Interministerial de Ciencia y Tecnología (BFU2004-00217/BMC), Comunidad Autónoma de Madrid (GR/SAL/0342/2004), National Institute of Health (HL70472), European Union (IST-2003-508833; FP6-502828; FP6-512092), and Fundación BBVA (UCAM2004030013).

6. References

1. **Zapata, E.L., et al.,** ***Filtered back-projection on shared-memory multiprocessors.*** **Ultramicroscopy, 1990. 34(4): p. 271-282.**
2. **Zapata, E.L., et al.,** ***Image template matching on hypercube SIMD computers.*** **Signal Processing, 1990. 21(1): p. 49-60.**
3. **Peltier, S.T., et al.,** ***The Telescience Portal for advanced tomography applications.*** **J Parallel Distr Com, 2003. 53: p. 539-550.**
4. **Fernandez, J.J., et al.,** ***High-performance electron tomography of complex biological specimens.*** **J Struct Biol, 2002. 138(1-2): p. 6-20.**
5. **Sorzano, C.O., et al.,** ***XMIPP: a new generation of an open-source image processing package for electron microscopy.*** **J Struct Biol, 2004. 148(2): p. 194-204.**
6. **Scheres, S.H., et al.,** ***Maximum-likelihood multi-reference refinement for electron microscopy images.*** **J Mol Biol, 2005. 348(1): p. 139-149.**