A Distributed Approach for Development of Deformable Model-Based Segmentation Methods

Daniel Reska, Cezary Boldak, and Marek Kretowski

Faculty of Computer Science
Bialystok University of Technology, Bialystok, Poland
{d.reska,c.boldak,m.kretowski}@pb.edu.pl

Summary. This paper presents a distributed solution for the development of deformable model-based medical image segmentation methods. The design and implementation stages of the segmentation methods usually require a lot of time and resources, since the variations of the tested algorithms have to be constantly evaluated on many different data sets. To address this problem, we extended our web platform for development of deformable model-based methods with an ability to distribute the computational workload. The solution was implemented on a computing cluster of multi-core nodes with the use of the Java Parallel Processing Framework. The experimental results show significant speedup of the computations, especially in the case of resource-demanding three-dimensional methods.

1 Introduction

Medical imaging methods provide a non-invasive insight into the human body and are an invaluable source of information in clinical and research practice. The constant progress of the imaging techniques calls for the development of more accurate and efficient image analysis algorithms. One of the most important task is segmentation, i.e. partitioning of the image into distinct regions in order to extract, for example, specific tissues, pathological changes or entire organs.

The usefulness of the segmentation methods is determined by their ability to provide fast, accurate and reproducible results, while dealing with the increasing size of the analyzed data. Image sets obtained with magnetic resonance imaging (MRI) or computed tomography (CT) can contain hundreds of hi-resolution images, which poses a performance challenge to the image processing and analysis algorithms. The time and resource-consuming computations are especially evident in the development stage of the segmentation methods, since the designed algorithms have to be fully tested on different

R.S. Choraś (ed.), *Image Processing and Communications Challenges 5,*
Advances in Intelligent Systems and Computing 233,
DOI: 10.1007/978-3-319-01622-1_3,

data sets and parameter values. Such a computational overhead urges for the use of parallelization tools that will utilize the power of the modern computers. The ubiquity of powerful, multi-core desktop machines encourages the usage of commodity hardware for the parallelization and distribution of the computations.

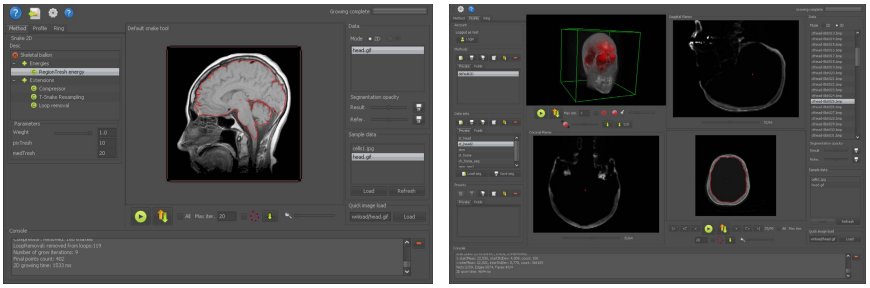
In this paper we present a distributed platform for development of the deformable model-based methods - a group of algorithms suitable for medical image segmentation [1], which use a deforming shape to match and extract an interesting object from the image. We used our extensible Java-based segmentation platform [2], originally build in a client-server architecture, and employed the Java Parallel Processing Framework (JPPF) [3] to distribute of computations over a cluster of commodity machines. The distribution scheme also makes use of parallelization within one process with the help of Java 7 Fork/Join Framework [4]. This solution aims for the simplicity of the computational model, reusability of the existing code and low deployment and maintenance cost. The framework was tested on different data sets and segmentation methods of various complexity and showed a significant performance speedup.

The paper is organized as follows. Section 2 outlines the deformable models background and the work in distributed image analysis. Section 3 presents our web platform for the development of segmentation methods. Section 4 describes the distribution method integrated in the platform. Experimental results are presented in Section 5. Conclusions and plans for future research are sketched in the last section.

2 Background and Related Work

Deformable models are a class of segmentation methods based on a deforming shape that tries to adapt to a specific image region. The shape is usually represented as a 2D contour or a 3D surface and undergoes a deformation process under the influences of external and internal forces. The external forces attract the shape to the segmented image features, while the internal forces control its smoothness and continuity. The classical forms of the deformable models, like the influential 2D "snake" algorithm [5] and a 3D active surface [6], were modified and extended, e.g., with adaptive topology mechanisms [7] or robust expansion forces [8]. The distinctive features of deformable models, like their continuous nature and noise insensitivity, make them well suited for medical image segmentation [1].

Recently, many tools for distributed computing have been used in general and medical image analysis. Apache Hadoop [9], an open-source implementation of MapReduce paradigm, was successfully used for the task of image processing, proving to be particularly effective in analysis of large volumes of data [10]. Notably, a distributed active contour [11] was implemented using the Globus Toolkit [12] - a powerful tool for building grid systems. JPPF [3]



(a) (b)

Fig. 1. Client interface of the segmentation application in 2D contour (a) and 3D surface mode (b)

is an another distributed computing framework, written in the Java language, that exhibited usefulness in medical image analysis [13].

3 Web-Based Segmentation Framework

We have developed a web-based platform [2] for the task of design and evaluation of deformable model-based biomedical image segmentation methods. The system is built in a server-client architecture, offering client applications in a form of Java applets, while the computations and data storage are performed on the server. The clients are provided with a set of tools for development, evaluation and visualization (2D planar reconstruction and 3D volume rendering) of the methods, a remote image data repository (with planned DICOM support) and a system of user profiles. The segmentation framework is based on an extensible template system for constructing two and three-dimensional methods. Each segmentation method can be composed from a set of existing components, like deformation models, internal and external energies, constraints or custom extensions. The users can also create their own components with a build-in scripting language and store them in the profile system. This decomposition scheme enforced the creation of a unified method interface for computation, visualization and comparison of the results. Currently, the platform support the 2D active contour, working on a set of images, and a 3D active surface, working on a volumetric data set (see Fig. 1).

The deformable models computations are performed in a request-response manner. The client request contains a user-defined script with the method definition and its parameters, an address of the segmented data (stored in the repository) and, optionally, a location of a reference segmentation, essential for the evaluation of the results. Furthermore, the client can submit a batch job of many requests with the same image data, but different method

parameters. This case is a typical development scenario for finding the optimal set of parameter values and is the most time and resource consuming. After the computation, a response with the segmentation and evaluation results is returned to the client.

4 Distributed Solution

Our segmentation system was extended with the ability to parallelize the computational workload using a JPPF cluster of desktop machines. The service-oriented architecture of our framework allowed the distribution of the existing computing requests, making this process opaque from the client perspective. To fully utilize the hardware potential, the task are not only divided into many processes and distributed among the cluster worker nodes, but also parallelized on the single node level.

4.1 Task Distribution

The distribution of the requests among different machines was achieved with the JPPF Framework. The framework turned out to be a powerful tool with a very low integration cost.

The system structure was extended by the addition of the JPPF cluster (see Fig. 2). Previously, the system consisted of an Axis2/Java [14] web service server that was performing the computations, a data repository and an LDAP server for user authentication and profile storage. The cluster is composed of a single master server (called the Driver) and a set of worker desktop machines (nodes), contained in a private network. After the integration, the Axis2 server is only a facade for the computational requests, hiding the cluster from the public access. Requests passed to the Driver are distributed and performed independently on the nodes. After the computations, the results are send to the Driver, gathered and returned to the Axis2 server, that transmits them to the client using the original protocol.

Each node is directly connected to the data repository and loads the images required by the executed request. To avoid the unnecessary overhead of reloading of the recurring data, the nodes are equipped with an in-memory cache for the recent images. The cache is implemented with the Google Guava library [15], that provides lightweight and thread-safe caching utilities.

One of the most useful features of JPPF is the dynamical deployment of the services: the framework automatically copies and distributes the client code among the cluster nodes. This process relieves the developer from the maintenance overhead and provides a way for rapid prototyping on the distributed level, which is especially beneficial during the development stage. However, the auto-deployment can open a possibility of abuse, therefore, in our system, the cluster is hidden behind the Axis2 web service facade.

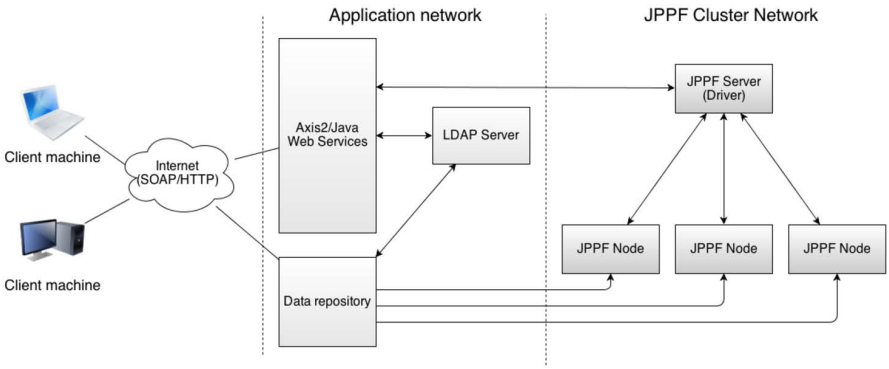


Fig. 2. Diagram of the application network system

The programming model of the framework is also very simple and encourages the reusability of the existing code. There is no need for adaptation of a specific programming model (like MapReduce in Apache Hadoop) and no labor-intensive creation and configuration process (like in Globus Toolkit).

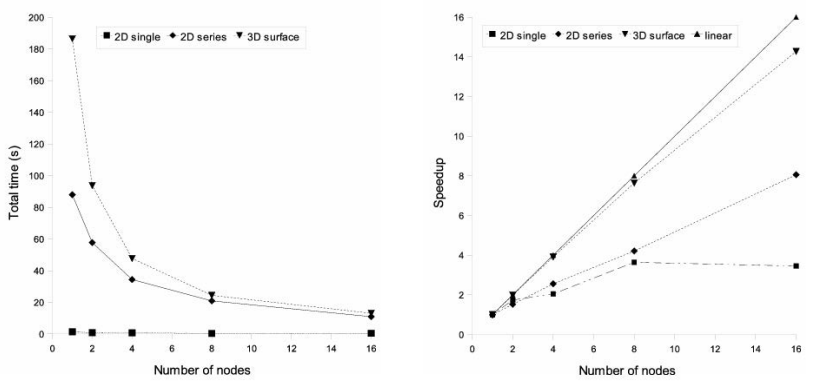
4.2 Single Task Parallelization

Apart from the distribution of the workload among different machines, much of the computations could be also parallelized within the single node process. This hybrid approach was used to fully utilize the power of multi-core worker machines and reduce the communication overhead.

The implemented algorithms make an extensive use of Java Fork/Join Framework [4], introduced in Java SE 7. The framework helps to recursively divide the work among many threads and uses a work-stealing algorithm to uniformly distribute the computations. This approach was used in two cases: to parallelize the execution of multi-image 2D request and to speed up the 3D active surface. In the first case, multiple active contours are uniformly split over the cores of a single node and executed in parallel. The active surface evolution time benefited from the parallelization of the polygon mesh voxelization algorithm [16]. This method is used to find the internal region of the surface, necessary for the calculation of its statistics [17] and during the generation of the segmentation results.

5 Experimental Evaluation

The distributed system was evaluated on a cluster of 16 worker machines. The experiments were performed with deformable model methods of various computational complexity and data demand.



(a) (b)

Fig. 3. Results of the experiments: total processing time (a) and speedup (b)

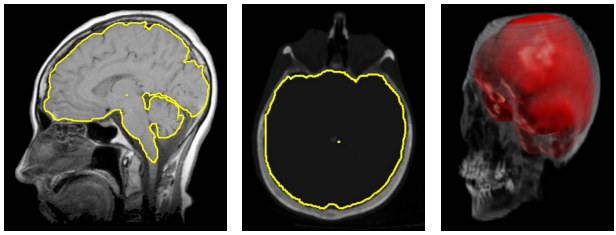
The cluster consisted of 16 desktop computers with Intel Xeon CPU E31270 4 core processors and 8 GB of RAM, each running a single instance of JPPF node on Linux 2.6 and connected by Gigabit Ethernet network. The JPPF Driver was deployed on a rack server with 6 core Intel Xeon E5645 CPU with 16 GB RAM. The Axis2 server configuration was similar to the Driver. The cluster was built using the 3.2 version of JPPF.

The system performance was tested on a 3 real-life segmentations tasks. Each task used a single deformable method, but was divided into a number of request with different sets of method parameters. The experiments consisted of:

- **2D single**: a brain segmentation on a single 512x512 MRI image with the adaptive 2D active contour [18] (20 sets of parameters);
- **2D series**: a segmentation of human cranium on a set of 90 256x256 CT images with the mentioned 2D active contour (16 sets of parameters);
- **3D surface**: a segmentation of human cranium with 3D deformable surface [17] (100 256x256 CT images, 16 sets of parameters).

Figure 3a presents the total calculation time of the tasks and the speedup is shown in Fig. 3b. Sample results of the segmentations are presented in Fig. 4.

The performance gain is clearly visible for the last two cases (2D series and 3D surface), while the speedup benefit of the first case is less impressive. The 2D single contour was not computationally intensive enough to overcome the framework overhead (i.e. network communication, data retrieval) and even demonstrated a slight decrease of the speedup between 8 and 16 cluster nodes. The two other, more demanding tasks showed a significantly better scalability, where the 3D surface speedup was the closest to the ideal linear reference.



(a) (b) (c)

Fig. 4. Sample results from the experiments: brain segmentation with a 2D active contour (a), segmentation of the cranium with a 2D snake (b) and with a 3D active surface (c)

The 2D series case achieved a steady performance gain, but the overall speedup was less impressive. All the requests of the 3D surface exhibited a roughly constant time, while the execution time of the 2D active contour was more dependent on the value of the parameters. This time variation leads to the decrease of the scalability, because the optimal distribution of the tasks would require an advance-knowledge of the tasks execution time, or an employment of a more advanced, work-stealing load balancing algorithm.

It should be noted that the single node performance is dependent on the retrieval of the image data from the repository. Fortunately, the usage of the cache mechanism makes the data loading overhead present only during the execution of the first request from the group.

6 Conclusion

In this paper a distributed framework for deformable model-based medical image segmentation is presented. The solution makes use of a workload distribution on a JPPF cluster and shared-memory parallelization with the Java Fork/Join framework. This approach was seamlessly integrated in the existing, centralized client-server architecture of our segmentation framework and demonstrated a significant performance improvement during the experimental validation.

The presented solution is still in its development stage. Before the eventual integration with the production version of the segmentation platform, we will investigate the usage of custom load-balancing algorithms, incorporation of heterogeneous cluster hardware and utilization of general-purpose computing on graphics processing units.

Acknowledgement. This work was supported with the European funds in the frame of the project "Information Platform TEWI" (Innovative Economy Programme). The experiments were run in a laboratory and on equipment co-financed

with the European funds in the frame of the project "Centre for Modern Education of the Bialystok University of Technology" (Operational Programme Development of Eastern Poland).

References

1. Tsechpenakis, G.: Deformable model-based medical image segmentation. In: Multi Modality State-of-the-Art Medical Image Segmentation and Registration Methodologies, vol. 1. Springer Publishing (2011)
2. MESA - a web platform for deformable model-based segmentation method development, http://mesa.wi.pb.edu.pl
3. JPPF – an open-source, Java-based, framework for parallel computing, http://www.jppf.org
4. Lea, D.: A Java fork/join framework. In: Proceedings of the ACM 2000 Conference on Java Grande JAVA 2000, pp. 36–43 (2000)
5. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active contour models. International Journal of Computer Vision 1(4), 321–331 (1988)
6. Miller, J., Breen, D., Lorensen, W., O'Bara, R., Wozny, M.: Geometrically deformed models: a method for extracting closed geometric models form volume data. SIGGRAPH Comput. Graph. 25(4), 217–226 (1991)
7. McInerney, T., Terzopoulos, D.: Topology adaptive deformable surfaces for medical image volume segmentation. IEEE Transactions on Medical Imaging 18(10), 840–850 (1999)
8. Xu, C., Prince, J.: Snakes, shapes, and gradient vector flow. IEEE Transactions on Image Processing 7(3), 359–369 (1998)
9. Apache Hadoop, http://hadoop.apache.org
10. Huang, F., Zhao, H., et al.: Watermarking Massive Remote Sensor Images in Parallel. In: International Conference on Computational Intelligence and Software Engineering, CiSE 2010, pp. 26–29 (2010)
11. Cannataro, M., Guzzi, P.H., Lobosco, M., Dos Santos, R.W.: GridSnake: A Grid-based implementation of the Snake segmentation algorithm. In: 2009 22nd IEEE International Symposium on Computer Based Medical Systems, pp. 2–7 (2009)
12. Globus Toolkit, http://www.globus.org/toolkit
13. Zerbe, N., Hufnagl, P., Schlüns, K.: Distributed computing in image analysis using open source frameworks and application to image sharpness assessment of histological whole slide images. Diagnostic Pathology 6(Suppl. 1), S16 (2011)
14. Apache Axis2 - Apache Axis2/Java - Next Generation Web Services, http://axis.apache.org/axis2/java/core
15. Guava: Google Core Libraries for Java 1.6+, http://code.google.com/p/guava-libraries
16. Thon, S., Gesquière, G., Raffin, R.: A Low Cost Antialiased Space Filled Voxelization of Polygonal Objects. In: GraphiCon 2004, pp. 71–78 (2004)
17. Reska, D., Boldak, C., Kretowski, M.: Fast 3D segmentation of hepatic images combining region and boundary criteria. Image Processing & Communications 17(4), 31–38 (2012)
18. Reska, D., Kretowski, M.: HIST - an application for segmentation of hepatic images. Zeszyty Naukowe Politechniki Bialostockiej. Informatyka 17(8), 71–93 (2011)