Exploring Online Nuclear Segmentation on Large Fluorescence Brain Tumor Images using CometCloud

Xin Qi[1,2], Daihou Wang[2,3], Javier Diaz-Montes[3], Ivan Rodero[3], Tony Pan[5], Abulimit Aji[5], Lee Cooper[5], Fuyong Xing[6], Manish Parashar[3], David J. Foran[1,2,4], Lin Yang[6]

[1]Department of Pathology and Laboratory Medicine.
[2]Center for Biomedical Imaging & Informatics, The Cancer Institute of New Jersey,
[3]NSF Cloud and Autonomic Computing Center & Rutgers Discovery Informatics Institute,
Dept. of Electrical and Computer Engineering
[4]Department of Radiology
Rutgers, The State University of New Jersey
[5]Center for Comprehensive Informatics
Emory University
[6]Division of Biomedical Informatics, Department of Biostatistics
Department of Computer Science
University of Kentucky.

Abstract. A bottleneck of histopathology image segmentation is its execution speed and memory capacity for large images containing hundreds and thousands of objects, such as cells. In this paper we propose an approach to perform online nuclear segmentation on large fluorescence brain tumor images using CometCloud, an autonomic Cloud Engine. Based on our previously developed cellular segmentation algorithm, the seed detection and contour generation were parallelized on CometCloud, using the HPC infrastructure available at Rutgers University. The method was tested on some fluorescence brain tumor images (4096x4096x3), containing thousands of nuclei within each image. We have achieved more than 100 times speed up compared to the original sequential implementation.

1. Introduction

Automatic image segmentation of cell nuclei is a fundamental operation in computational pathology image analysis, and is the first step in quantitative analysis of nuclear morphometry. Variations in tissue processing and staining make it difficult to achieve robust segmentation performance over large image datasets. Additional computational complexity is necessary to address variations and produce quality segmentation results, especially in neoplastic tissues where nuclei are densely packed. This complexity adds significant computational costs that must be addressed through software optimization, including the usage of parallel computational architectures [1]. Tissue segmentation using color or multispectral images by combining spatial clustering with multiphase vector level set active contours was proposed for segmenting histological prostate cancer tissues in [2]. There are many approaches utilizing color and texture to separate epithelial nuclei, stroma, and background regions in breast microarray [3] and hematoxylin and eosin stained prostate cancer [4]. Many researchers proposed efficient methods to tackle touching cells within the image, such as a rule-based approach [5] for merging over-segmented regions, and Voronoi diagram [6] to correct the overlapped regions. However, it is difficult to derive a generalized rule to merge the over-segmented regions in various image types. An ellipse-fitting technique was applied to segment based on these concavities among nuclei within overlap regions [7]. However, the ellipse may not be suitable to model certain shapes of the cells, especially for cancerous cells. Wen et al. [8] reported their study on decomposing clumps of nuclei using high-level geometric constraints derived from maximum curvatures. This approach is quite effective in separating touching nuclei. However, for some touching objects, the common connecting region may not have local maximal curvatures. Kothari et al. [7] proposed a semi-automatic method for touching cell segmentation, in which they applied concavity detection at the edge of a cluster to find the points of overlap between two nuclei. Concave vertex graph [9] and another graph based method [10] were proposed to address touching cell segmentation. Chang et al. [11] proposed an approach to segment touching objects through iterative voting, level set is then followed to get final segmentation of each object. This approach fails when two objects overlaid with large areas. We have developed an cellular segmentation algorithm to handle this issue for overlapping cells in histopathology tissue microarray [12].

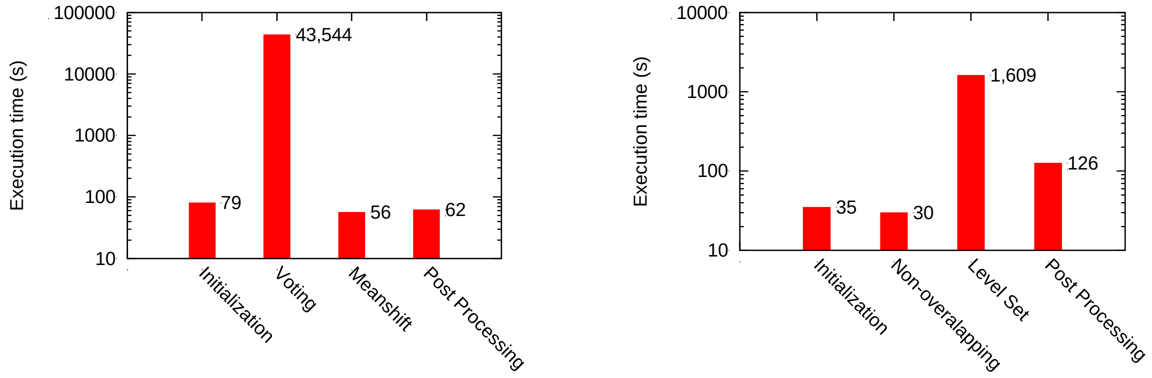
Due to the computational power required to process the cellular segmentation, it is necessary to explore innovative solutions, involving the use of various distributed computational resources, to provide answers within a reasonable time limit. In this paper, we present the use of CometCloud to process the cellular segmentation algorithm in parallel on multiple high performance computing (HPC) resources with the goal of reducing the overall computational time. CometCloud is an autonomic Cloud framework that enables dynamic and on-demand federation of distributed infrastructures such as HPC, Grid, and Cloud. It also provides a flexible programming platform that supports several models (MapReduce, Workflow, Master-Worker/BOT) to easily develop applications that can run across the federated resources. Therefore, our proposed solution exploits the potential large parallelism of the problem by making effective use of distributed HPC resources at Rutgers University.

2. Parallelizing the Nuclear Segmentation Algorithm

The bottleneck of our cellular segmentation is its execution speed and memory load for large images containing hundreds and thousands of cells. Mostly it is hard to load and process a large image in a single system, and it is a very slow process to get contours of all the cells (nuclei). Using our developed cellular segmentation algorithm, for example a 4096x4096 microscopic image taken under 20x objective containing 5,817 nuclei, which will take several hours to sequentially process the whole image by nuclei detection and contour generation. The aim of this paper is to study efficient methods to execute nuclear segmentation on large microscope images and explore the feasibility of running such an algorithm online by parallelizing our previous methods [12] using CometCloud. In this paper, we focus on the federation of HPC resources but the same framework and implementation can be used in other environments such as Clouds (e.g., Amazon Elastic Compute Cloud) as well.

The data used for this study were 20X objective magnification scans of glioblastoma brain tumor tissues stained with DAPI. Slides were digitized with an Olympus VS120 microscope, compressed with JPEG at a quality factor of 80. Images were tiled into 4096x4096 pixel tiles, and example tiles with moderate to high nuclear density were selected from each whole-slide image in order to test segmentation performance.

The whole cellular segmentation contains two sequential steps, seed detection and contour generation. Figure 1 shows the execution time for each step of seed detection and contour generation using an image (4096x4096) with 4,397 nuclei. From calculation profiles, voting and level set evolution are the bottlenecks for seed detection and contour generation, respectively.



(A) Seed Detection (B) Contour Generation

Fig 1. Execution time (seconds) for each step of seed detection and contour generation using an image (4096x4096) with 4,397 nuclei.

Seed detection is a process to find the center of nuclei. Based on our algorithm [12], a gradient image $\mathrm{F}(x,y)$, including magnitude $|\nabla I\ x,y\ |$ and phase $\theta(x,y)$, were first calculated from the whole image $\mathrm{I}(x,y)$. The voting area $A(x,y;r_{min},r_{max},\Delta)$ of each pixel with high gradient magnitude is defined by a cone-shape whose vertex is the pixel (x,y), and dependent on the radial and angular range defined by $A(x,y;\ r\_min, r\_(max,)\ \Delta) = \{(x + rcos\emptyset, y - rsin\emptyset)\},\ here\ r\_min \le r\ \le r\_(\max\ and)\ \theta(x,y) - \Delta \le\ \emptyset\ \le\ \theta(x,y) + \Delta$. The voting area of each pixel is marked by a 2-D Gaussian kernel $g(x,y)$ whose center is located at the center (centerX, centerY) of the voting area A and oriented in the voting direction $\alpha\ x,y$. Here $g\ x,y;\ \sigma\ = (\frac{1}{2\pi\sigma}e^{-(\ x-centerX\ ^2+\ y-centerY\ ^2)/2\sigma^2}$. For each pixel (x,y), the voting direction is defined as its negative gradient direction to be $F(x,y) = -(\nabla I(x,y))/(||\nabla I(x,y)||) = (\cos(\theta(x,y), -\sin(\theta(x,y))$.

Due to independency of each pixel (x,y), the whole image voting process can be parallelized with independent tasks (i.e., bag-of-task paradigm). After the completion of the independent tasks, the results of voting area from each pixel(x,y) are consolidated to create a whole voting image $V\ x,y;r_{min},r_{max},\Delta$, which is dependent on the radial and angular ranges and has the same dimensions as the original image $\mathrm{I}\ x,y$.

To calculate the voting image, the voting image $V\ x,y;r_{min},r_{max},\Delta\ = 0$ is reset for all pixels x,y as initialization. For each pixel (x,y), the voting image is updated as $V\ x,y;r_{min},r_{max},\Delta\ = V\ x,y;r_{min},r_{max},\Delta\ + F\ x+u,y+v\ \times g(x,y;\ \sigma,A)$, where $(u,v) \in A(x,y;\ r_{min},r_{max},\Delta)$. The final voting image is the sum of all voting images at each pixel. The centers of mass are determined by mean shift on the possible centers of each cell after thresholding the voting image at various ratios ($\mathrm{R} \in [0.1 : 0.1 : 0.9]$) of maximum value of final voting image.

After identifying the center of each nucleus, contour generation was performed using our proposed level set based interaction model. For a microscopic image containing thousands of nuclei, some nuclei are isolated from each other; some nuclei are touching boundaries or overlapping with each other. We first divide the whole image into two images. One contains nuclei isolated from each other, and the other contains nuclei overlapping with one another and some large size nuclei. Using morphology processing, a mask containing isolated nuclei and its contour were created. For the image containing large nuclei and overlapping nuclei, given the accurately detected seeds, the touching cell segmentation was performed using our level set based interaction model with a repulsion term to prevent the contours of adjacent cells from overlapping and separate the touching cell boundaries [12]. Previously due to hundreds and thousands of nuclei within an overlapping image, it would take multiple hours to finish the second-order level set evolution within the whole image. To make a reduction in memory use and boost the performance of level set evolution, the level set is restricted by evolution to a smaller neighborhood around the seeds. Meanwhile each seed with its neighbor was assigned as one parallel task to achieve parallel calculation of level set evolutions on CometCloud.

2.1 Image Segmentation across Federated Resources using CometCloud

The main challenges of the nuclear segmentation algorithm execution are the high computational power required to process the images, which translates into long execution time, and the large amount of RAM memory required during the algorithm execution. In order to overcome these challenges, we split the input images into multiple chunks. These chunks are processed as independent parallel tasks as we described above. Nevertheless, this comes at the cost of increasing the complexity of the workflow execution. In particular, it is needed to orchestrate both the execution of the tasks and the federation of the appropriated distributed resources to effectively compute those tasks. Thus, we decided to develop a framework based on CometCloud that can efficiently deal with the whole execution process. CometCloud creates a cloud abstraction on top of the resources that allow applications to make use of those resources on demand. Next, we highlight the main CometCloud features that are essential to overcome the challenges of our application workflow.

1) **Usability.** CometCloud creates a unified view of resources hiding their actual location and architecture. Users are able to interact with the federated resources using a programming abstraction. This abstraction eases the development of applications by decoupling the application from the particularities of the infrastructure. It supports several common distributed programming paradigms, including the master/worker, workflow and MapReduce.
2) **Scalability**. It allows scientific applications to scale across institutional and geographic boundaries. This is essential because oftentimes a single resource is not sufficient to execute a given scientific workload (e.g. because resource is of limited scale, or it mismatches application requirements). Of course, this must be achieved without hindering the usability.
3) **Autonomic management**. The manager enables the autonomic management and multi-objective optimization (including performance, energy, cost, and reliability criteria) of application execution through cross-layer application/infrastructure adaptations. This component offers QoS by adapting the provisioned resources to the application's behavior as well as system configuration, which can change at run time, using the notion of elasticity at the application level.
4) **Fault tolerance**. CometCloud has built in fault tolerance mechanisms to deal with unpredicted failures at the application and infrastructure levels. These mechanisms are

essential to guarantee the execution of the application without the intervention of the user. Thus, in case of failed tasks the master recognizes the error and either directly resubmits task (hardware error), or regenerates it by, for example increasing the minimal hardware requirements (application error).

5) **Elasticity and on-demand access**. An important factor that enables the previous features is the ability of CometCloud to seamlessly scale up/down or out resources as needed. What is important, the resulting elasticity makes the infrastructure resilient and hence improves its ability to sustain computational throughput and support fault tolerance mechanisms.

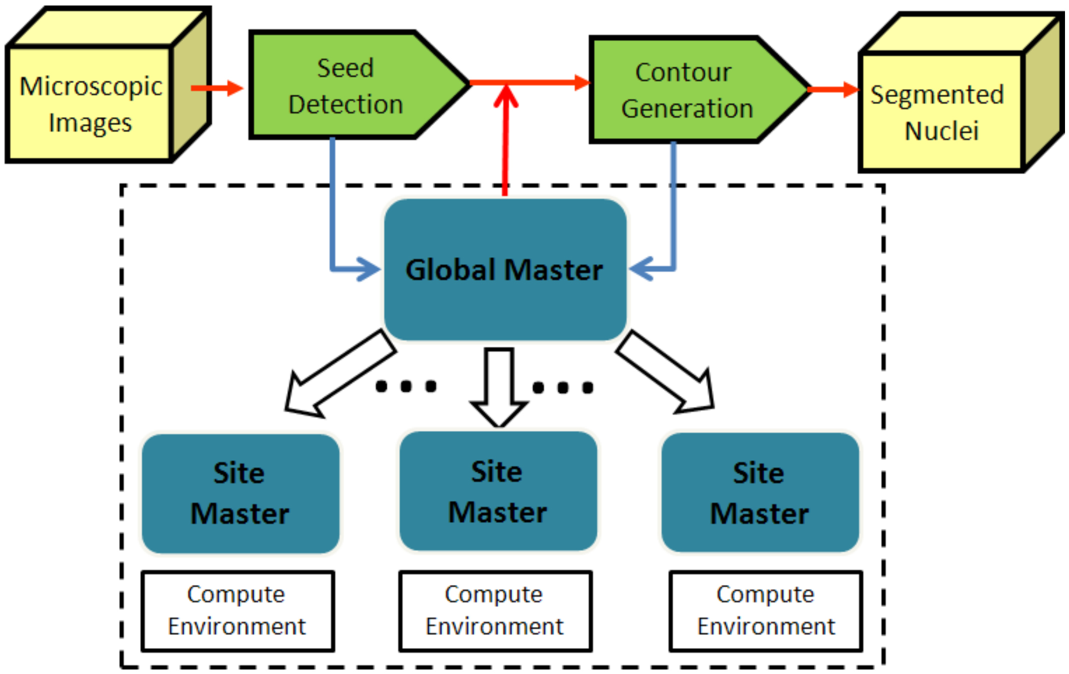


Fig 2. CometCloud workflow architecture for the image processing pipeline.

In order to run our computational problem on a federation of resources, we combined the MATLAB-based implementation of the nuclear segmentation algorithm with CometCloud infrastructure using the master/worker paradigm. In this scenario, the segmentation algorithm serves as a computational engine, while CometCloud is responsible for orchestrating the entire execution. The master/worker model is among several directly supported by CometCloud, and it perfectly matches problems with large pool of independent tasks. The master component takes care of generating tasks, collecting results, verifying that all tasks executed properly, and keeping log of the execution. All tasks are automatically placed in the CometCloud-managed task space for execution. In the proposed approach, the workers' sole responsibility is to execute tasks pulled from the task space. To achieve this, each worker creates a subscription that indicates the type of tasks that is interested on and its computational characteristics. As soon as the tasks are placed into the task space, the autonomic manager notifies each worker when there are tasks matching its subscription. This publish/subscribe model enables an efficient use of the resources by reducing the overhead created when workers blindly query the task space looking for tasks to compute.

Specifically, the workflow we have implemented has two main phases: 1) Seed detection, and 2) Contour generation (see Figure 2). Each of these phases has two stages where the output of one stage is the input of the next one. Note that our framework is prepared to process multiple images simultaneously, and therefore the execution of the workflow of an image is independent from the others. Next, we will describe the details about how the workflow works.

The workflow starts with seed detection phase. The first stage of this phase is to compute the voting area from each pixel(x, y). Thus, the master splits each image and generates hundreds of tasks for each image. The number of tasks is given by the size of the image and the number of chunks in which the image is split. These tasks are computed by the workers that return the results to the master upon completion. Once the master has collected all the results of an image, the next stage of the workflow is started. This stage generates a single task to consolidate the results of that image and obtain the center of each nucleus. Next, the contour generation phase is started. In this phase, a single task analyzes the results of the seed detection phase and generates thousands of tasks. Here the number of tasks depends on the number of nuclei in the image.

3. Experimental Evaluation

In this section, we present the results obtained with the proposed parallel nuclear segmentation implementation. Specifically, we first provide results of the segmentation algorithm on some sample imagess. Then we will focus on the execution evaluation of the CometCloud implementation.

Figure 3 (A) and (B) shows segmentation results of two fluorescence stained brain tumor images (4096x4096) at 20x objective. The first and third rows are the non-overlapping nuclei segmentation whose contours are in green with their zoom in sub-images. The second and fourth rows are the overlapping and large nuclei segmentation whose contours are in red with their zoom in sub-images.

The experimental test bed consists of a set of state-of-the-art HPC resources including a 32-core SMP system (Snake) with Intel Xeon X7550 processors and 128GB or RAM memory, and 32 systems with 8 cores (Intel Xeon E56620) and 6GB or RAM memory each (Dell cluster). While Snake supports large memory capacity demand, the cluster requires distributing the tasks among the nodes in such a way that can be executed in parallel and each parallel task has lower memory requirements.

In this paper, we focus on the seed detection algorithm because it has higher potential for parallelization. In the current implementation of the level set component of contour generation (see Figure 1) each detected seed if processed considering the adjacent pixels, which may contain neighbor elements of interest. Thus, instead of parallelizing within images we exploit parallelism at the workload level (i.e., processing whole images in parallel).

The contour generation algorithm execution over the whole data set took more than 3 hours sequentially and 36 minutes using CometCloud. On the other hand, the seed detection algorithm took more than 12 hours sequentially and less than 4 hours using CometCloud. Note that the seed detection algorithm does not scale linearly when smaller chunks of the image are run in parallel, i.e., the execution time for N chunks in parallel is larger than the execution time of processing the whole image sequentially divided by N.

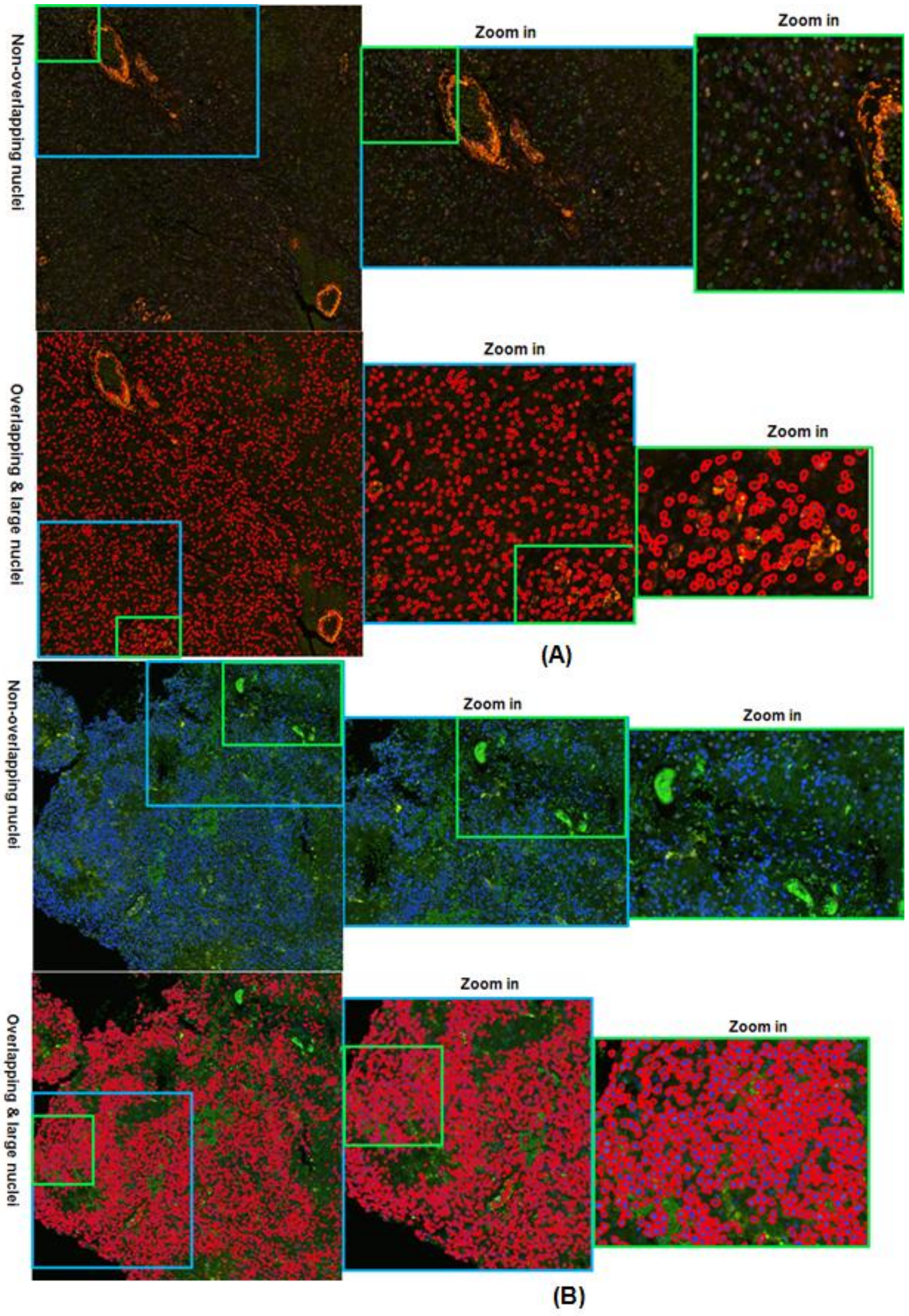


Fig 3. (A) and (B): Segmentation results of two fluorescence stained brain tumor images (4096x4096) at 20x objective.

Figure 4 shows the execution time of the tasks that process the chunks of the images in parallel. Note that tasks in Snake take longer than Dell cluster, because Snake is more subscribed than the Dell nodes, and the Snake cores are slower than the Dell nodes' ones. The average task execution time is 793s in Snake and 563s in Dell. The standard deviation is 172s (21.75%) and 106s (18.80%) in Snake and Dell, respectively. This illustrates that the algorithm is clearly sensible to the content of each image chunk.

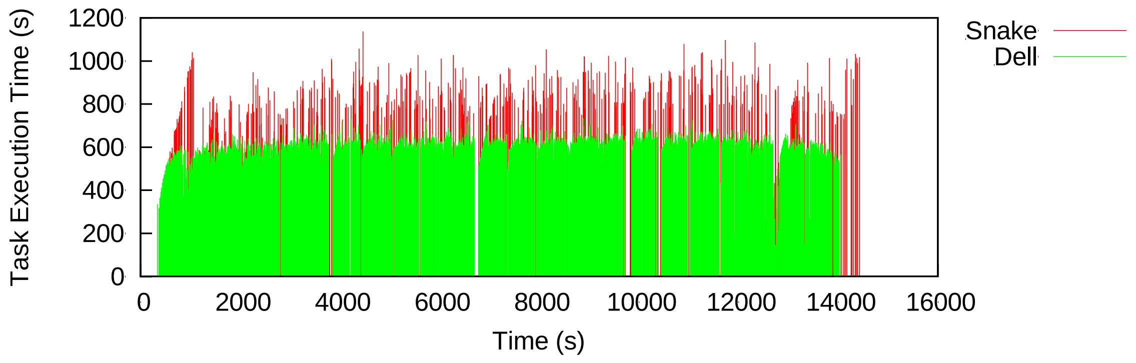
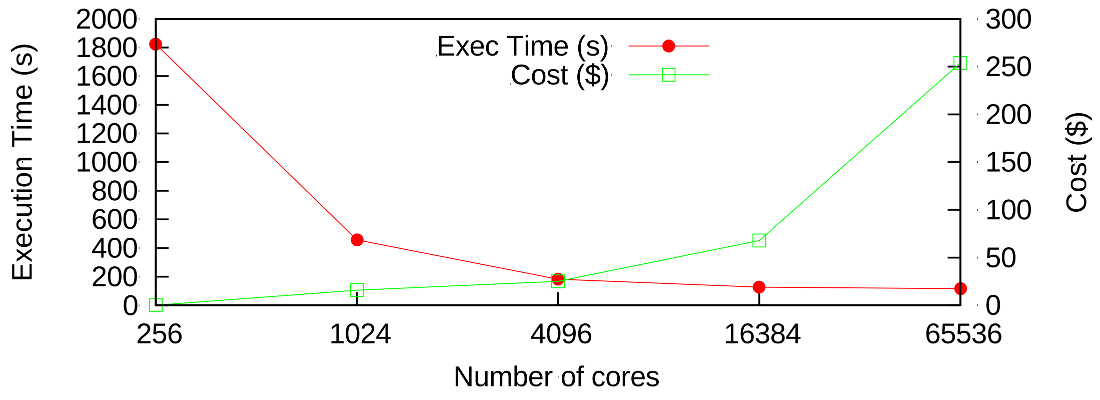


Fig 4. Tasks execution time on Snake and Dell

One of the goals of this paper is to study the potential of the proposed implementation and execution framework to run automatic image segmentation in an online manner. This is an important aspect since we may want to determine, for example, how many resources are needed to meet certain deadline requirements. To do this, we have conducted a set of experiments processing with the seed detection algorithm on a sample image using CometCloud and different partition configurations. The experiments were conducted over a representative set of tasks. The partition configurations generate from 256 to 65,536 independent tasks. The obtained results are shown in the summary table of Figure 5. It also shows the estimated cost for running the independent tasks in parallel. We used economic cost considering that local resources (i.e., 256 cores) are available for free, and the rest of the resources are obtained from Amazon EC2 in pay-as-you-go basis. The estimation was done based on the cost of "Medium" Amazon EC2 instances. Figure 5 also shows that overheads are higher for larger number of partitions and the tradeoff between the execution time and cost. The execution results also show that it wouldn't always be the best option by mapping one task per core for high efficiency. For example, with 256 available cores we can use different number of tasks and task mappings:

- 256 tasks: 256 cores and each runs 1 task (3,280s)
- 1024 tasks: 256 cores and each runs 4 tasks (456s*4 = 1,824s < 3,280s)
- 4,096 tasks: 256 cores and each runs 16 tasks (184s*16 = 2,944s < 3,280s)
- 16,384 tasks: 256 cores and each runs 64 tasks (128s*64 = 8,192s > 3,280)
- Etc.

In this example the best configuration is to partition with (32,32) and run the 1,024 tasks on 256 cores. The reason for this behavior can be found in the better utilization of memory with smaller chunks.



# Partitions (x, y)	# Tasks/image	AVG Exec time/task	Estimated cost
(16, 16)	256	3,280 s	0 $ (local)
(32, 32)	1,024	456 s	$15.56
(64, 64)	4,096	184 s	$ 25.1
(128, 128)	16,384	128 s	$ 68
(256, 256)	65,536	117 s	$ 254

Fig 5. The estimated task execution time with different partition configurations. The estimation is done based on the average of a representative set of tasks using a sample image. The cost is based on Amazon EC2 “Medium” instances. Note that the figure considers the best partition configuration for each number of cores.

4. Conclusion

In this paper, we describe a newly developed, online segmentation algorithm which we tested on an ensemble of glioblastoma brain tumor images. Our experimental evaluation stated that the contours of nuclei were delineated effectively using a new seed detection and level set active contour based interaction model. We demonstrated that using our parallelization approaches, the bottleneck of cellular segmentation due to execution speed and memory capacity for large images can be significantly reduced. The approach is automatic and not limited to domain specific prior knowledge. Therefore it can be extended to other imaging modality touching object segmentation applications. At the same time, the CometCloud platform can exploit different types of resources such as public clouds, therefore enabling an online execution of the segmentation algorithm if more resources or budget are available.

Acknowledgements

This research was funded, in part, by grants from the National Institutes of Health through contract 5R01CA156386-06 and 5R01CA161375-02 from the National Cancer Institute; and contract 5R01LM011119-02 from the National Library of Medicine. Additional support was provided by a gift from the IBM International Foundation. This research is also supported by the National Science Foundation (NSF) Office of Industrial Innovation and Partnerships Industry/University Cooperative Research Center (I/UCRC) Program under award 0758596 and via grants numbers OCI 1310283, OCI 1339036, DMS 1228203 and IIP 0758566. CometCloud was developed as part of the NSF Cloud and Autonomic Computing Center and the Rutgers Discovery Informatics Institute at Rutgers University. The project is also partially supported by the National Center for Research Resources and the National Center for Advancing Translational Sciences, National Institutes of Health, through Grant UL1TR000117 (or TL1 TR000115 or KL2 TR000116). The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH.

References

[1] L. A. D. Cooper, A. B. Carter, A. B. Farris, F. Wang, J. Kong, D. A. Gutman, P. Widener, T. C. Pan, S. R. Cholleti, A. Sharma, T. M. Kurc, D. J. Brat, and J. H. Saltz, "Digital pathology: data-intensive frontier in medical imaging," *Proceedings of IEEE,* vol. 100, pp. 991-1003, 2012.

[2] A. Hafiane, F. Bunyak, and K. Palaniappan, "Evaluation of level set-based histology image segmentation using geometric region criteria," *Proceedings of IEEE International Symposium on Biomedical Imaging,* vol. MP-PA1, pp. 1-4, 2009.

[3] T. Amaral, S. McKenna, K. Robertson, and A. Thompson, "Classification of breast-tissue microarry spots using colour and local invariants," *Proceedings of IEEE International Symposium on Biomedical Imaging,* pp. 999-1002, 2008.

[4] M. Datar, D. Padfield, and H. Cline, "Color and texture based segmentation of molecular pathology images using hsoms," *Proceedings of IEEE International Symposium on Biomedical Imaging,* pp. 292-295, 2008.

[5] C. Wahlby, J. Lindblad, M. Vondrus, E. Bengtsson, and L. Bjorkesten, "Algorithms for cytoplasm segmentation of fluorescence labelled cells," *Anal. Cell. Pathol.,* vol. 24, pp. 101-111, 2002.

[6] X. Zhou, K. Y. Liu, N. Bradley, N. Perrimon, and S. T. Wong, "Towards automated cellular image segmentation for RNAi genome-wide screening," *proc. Med. Image Comput. Comput.-Assisted Intervention,* pp. 885-892, 2005.

[7] S. Kothari, Q. Chaudry, and W. D. Wang, "Automated cell counting and cluster segmentation using convavity detection and ellipse fitting techniques," *Proceedings of IEEE International Symposium on Biomedical Imaging,* vol. TO2-R3.2, pp. 795-798, 2009.

[8] Q. Wen, H. Chang, and B. Parvin, "A delaunary triangulation approach for segmenting clumps on nuclei," *Proceedings of IEEE International Symposium on Biomedical Imaging,* vol. MP-PA3, pp. 9-12, 2009.

[9] L. Yang, O. Tuzel, P. Meer, and D. J. Foran, "Automatic image analysis of histopathology specimens using concave vertex graph," *International Conference on Medical Image Computing and Computer Assisted Intervention,* vol. 5241, pp. 833-841, 2008.

[10] G. M. Faustino, M. Gattass, S. Rehen, and C. J. P. Lucena, "Automatic embryonic stem cells detection and counting method in fluorescence microscopy images," *Proceedings of IEEE International Symposium on Biomedical Imaging,* vol. TO2-R3.3, pp. 799-802, 2009.

[11] H. Chang, Q. Yang, and B. Parvin, "Segmentation of heterogeneous blob objects through voting and level set formulation," *Pattern Recognition Letters,* vol. 28, pp. 1781-1787, 2007.

[12] X. Qi, F. Xing, D. J. Foran, and L. Yang, "Robust segmentation of overlapping cells in histopathology specimens using parallel seed detection and repulsive level set," *IEEE Transactions on Biomedical Engineering,* vol. 59, pp. 754-765, 2012.