# Integrating Image Segmentation Algorithm with MIDAS

***Abstract*-We perform image segmentation using the Watershed algorithm, and then propose an implementation to run several image segmentation tasks on parallel using RADICAL-Pilot's API. Apparently, we conducted some experiments to characterize the performance of our application, and found the results to be very encouraging.**

## 1 Introduction

Image segmentation [1],[2] is the process of partioning a digital image into multiple segments (subsets of pixels). Its goal is to cluster pixels into salient image regions, changing the representation of an image into something that is easier to analyze. Image segmentation could be used for object recognition, image processing, image compression, or image database look-up.

Since there are many possible partitions to the domain of an image into subsets, is there a way to pick one? The answer to the previous question is that there may not be a single correct partition of an image. The partition of an image could vary depending on the nature of the application that needs the image to be segmented.

Our goal is to perform image segmentation- we do this by using the Watershed algorithm [3],[4],[5] - on a large volume of images in parallel. Therefore, we develop a program using RADICAL-Pilot's API [6]. Furthermore, we conduct some experiments so as to characterize the behavior and performance of our application. We do not intend to approach the matter of image segmentation from the point of view of a computer vision scientist, yet we would like to implement a segmentation algorithm that could process a lot of images in parallel, using our lab's middleware (RADICAL-Pilot) and characterize its behavior and scalability.

This document is stuctured as follows: In Section 2 we mention a few words about the algorithm that was chosen to perform image segmentation. We continue in Section 3, describing our implementation. In Section 4, we mention the experiments that were carried out and provide the corresponding results. We conclude with a discussion on the results, as well as relevant future work in Section 5.

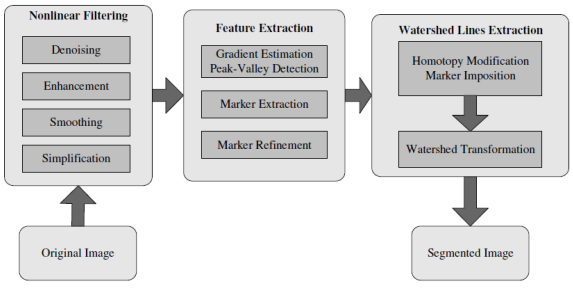
## 2 The Watershed Algorithm

The watershed alogorithm is an example of mathematical morphology [7], a nonlinear image analysis methodology that is based on set and lattice theoretic approaches whose goals are to quantify the geometrical structure of images.

The idea of the watershed segmentation algorithm can be summarized in three stages [8]. At a first stage the image is simplified and processed in such a way that the presence of noise is reduced and redundant infor-

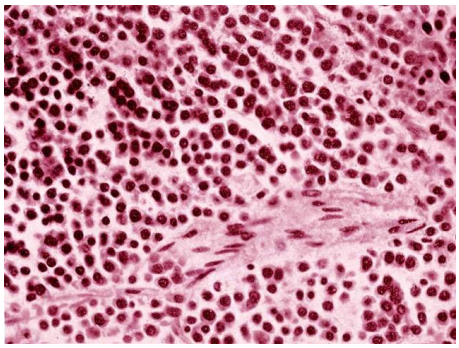
mation is removed, thus producing an image that consists mostly of flat and large regions. The second stage involves the region-feature extraction, where the goal is to extract some special features such as small homogeneous regions, called markers, which will be used as the starting points for the flooding process. The selection of the markers is probably the most difficult task and the strategies for finding them are diverse and problem dependent. The third stage is the application of the watershed algorithm, where a gradient image [9] is constructed and its topographic surface is flooded by sources placed at the position of the markers. The watershed construction grows the markers until the exact contours of the objects are found. Catchment basins without sources are flooded by already flooded neighboring catchment basins. In order to avoid the merging of lakes produced by different sources, dams are erected to keep them separated. The set of dams constitutes the boundaries of the resulting segmentation. We would like to point out that the preprocessing tuning which takes place in the first stage of the algorithm is dependant on the nature of the image to be segmented.

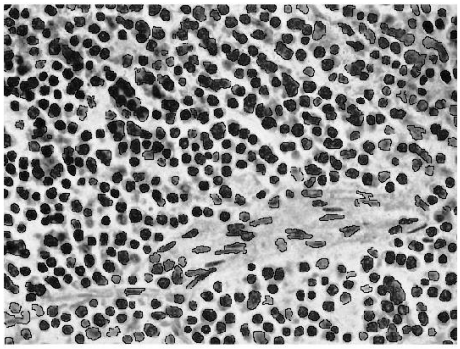
We present a typical pipeline of the algorithm described.



Basic stages of watershed segmentation

Following, there is an image before and after the application of the segmentation process.





## 3 Watershed Parallel Iplementation using RADICAL-Pilot

For the implementation in which we can process a large number of images in parallel, we use RADICAL-Pilot [10], an implementation of the Pilot-Abstraction [11]. RADICAL-Pilot consists of a client module with the Pilot-Manager and Unit-Manager and a set of RADICAL-Pilot-Agents running on the resource. The Pilot-Manager is responsible for managing the lifecycle of a set of Pilots. The role of Unit-Manager and RADICAL-Pilot-Agents is to manage the application's workload. More details can be found in [12].

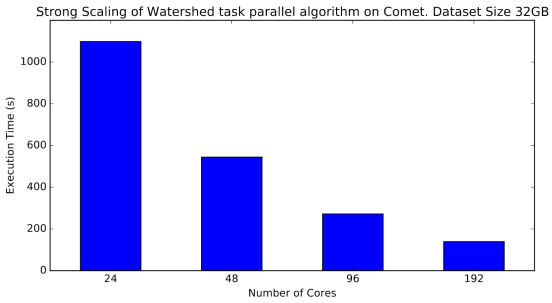
We do not parallelize the algoritm in the sense that every image is still being processed

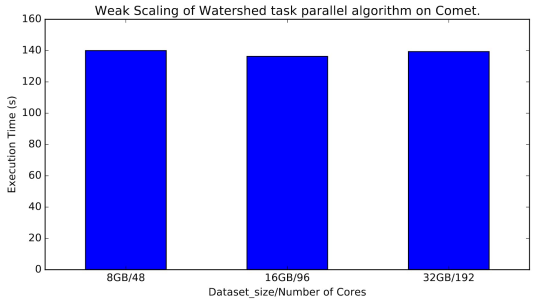
by a single Compute-Unit(Thread). Our goal is to use a number of Compute-Units, each of which processes a subset of the total images that have to be segmented. For performance reasons, we need to have load-balance and hence we must distribute our data effectively. This can be done by assigning to each Compute-Unit (almost) the same size of input data.

## 4 Experiments and Evaluation

To evaluate the performance of our application we conducted both strong and weak scaling experiments. All experiments are performed on Comet, an XSEDE allocated machine. On Comet every node has 24 cores and 128GB of memory.

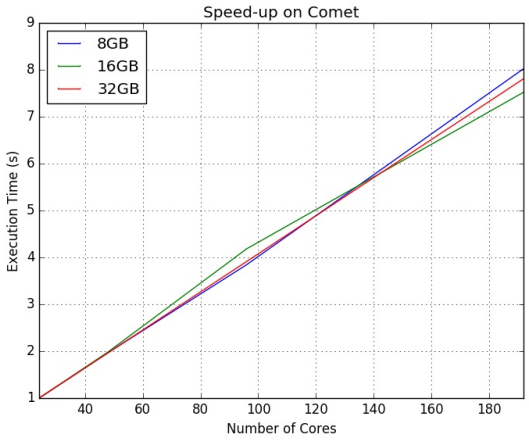
We experimented with different configurations, regarding the dataset size (we used datasets of size 8GB, 16GB and 32GB) and the number of cores used for both weak and strong scaling.Some indicative results are being shown in the following graphs.




Both experiments give us encouraging results. The scalability of our application is satisfactory in both cases, and therefore we cannot classify the application as cpu or memory bound.

There is also a speed-up graph that shows us this behaviour.




In the graph above, we observe linear speed-up for every configuration. To compute speed-up we used the formula $\frac{T_n}{T_p}$, where $T_n$ corresponds to the execution time using 1 node (24 cores in Comet), and $T_p$ corresponds to the execution time using p$\in\{2,4,8\}$ nodes, so using c$\in\{48,96,192\}$ cores. The full set of profiling data and plots is available at [13].

## 5 Conclusion and Future Work

The Watershed algorithm seems to be a good choice for the purpose of image segmentation. Furthermore, we have shown that it can be executed efficiently in parallel using RADICAL-Pilot.

*Future Work:* This work provides a starting point for research on a variery of directions. One could use other image segmentation algorithms, parallelize them as we did for the wa-

tershed algorithm, and finally compare their performance both in the means of execution-time/scalability and precision in segmentation. Another research path would be to implement the algorithm using different HPDC architectures [14] and ecosystems such us Apache Hadoop [15] or Apache Spark [16]. Their performances could then be analyzed and compared to each other.

## References


[1] Image segmentation — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Image_segmentation, 2017.

[2] A.D. Jepson and D.J. Fleet. Image segmentation. http://www.cs.toronto.edu/ jepson/csc2503/segmentation.pdf, 2007.

[3] Watershed (image processing) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Watershed_(image_processing), 2017.

[4] Serge Beucher. The watershed transformation. http://cmm.ensmp.fr/ beucher/wtshed.html, 2010.

[5] Lamia Jaafar Belaid and Walid Mourou. Image segmentation: a watershed transformation algorithm. *Image Analysis & Stereology*, 28(2):93–102, 2011.

[6] Pilot API. http://radicalpilot.readthedocs.org, 2010.

[7] Mathematical morphology — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Mathematical_morphology, 2017.

[8] Petros Maragos. Segmentation, chapter 18 of textbook Image Analysis and Computer Vision, 2005.

[9] Image gradient — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Image_gradient, 2017.

[10] RADICAL-Pilot. http://radicalpilot.readthedocs.org, 2010.

[11] Andre Luckow, Mark Santcroos, Andre Merzky, Ole Weidner, Pradeep Mantha, and Shantenu Jha. P: A model of pilot-abstractions. In *E-science (e-science), 2012 IEEE 8th International Conference on*, pages 1–10. IEEE, 2012.

[12] Andre Merzky, Mark Santcroos, Matteo Turilli, and Shantenu Jha. Executing dynamic and heterogeneous workloads on super computers. *arXiv preprint arXiv:1512.08194*, 2015.

[13] Experiments repository. https://github.com/radical-experiments/midas_exps/tree/master/watershed.

[14] Best Papers of HPDC. http://www.hpdc.org/best.php.

[15] Apache Hadoop. http://hadoop.apache.org, 2017.

[16] Apache Spark. http://spark.apache.org, 2017.