Parallel Definition of Tear Film Maps on Distributed-Memory Clusters for the Support of Dry Eye Diagnosis

Jorge González-Domínguez[a,*], Beatriz Remeseiro[b], María J. Martín[a]

[a] *Grupo de Arquitectura de Computadores, Universidade da Coruña*
Campus de Elviña s/n, 15071 A Coruña, Spain
[b] *INESC TEC - INESC Technology and Science*
Campus da FEUP, Rua Dr. Roberto Frias, 4200–465 Porto, Portugal

Abstract

Background and Objectives: The analysis of the interference patterns on the tear film lipid layer is a useful clinical test to diagnose dry eye syndrome. This task can be automated with a high degree of accuracy by means of the use of tear film maps. However, the time required by the existing applications to generate them prevents a wider acceptance of this method by medical experts. Multithreading has been previously successfully employed by the authors to accelerate the tear film map definition on multicore single-node machines. In this work we propose a hybrid message-passing and multithreading parallel approach that further accelerates the generation of tear film maps by exploiting the computational capabilities of distributed-memory systems such as multicore clusters and supercomputers.

Methods: The algorithm for drawing tear film maps is parallelized using Message Passing Interface (MPI) for inter-node communications and the multithreading support available in the C++11 standard for intra-node parallelization. The original algorithm is modified to reduce the communications and increase the scalability.

Results: The hybrid method has been tested on 32 nodes of an Intel cluster (with two 12-core Haswell 2680v3 processors per node) using 50 representative images. Results show that maximum runtime is reduced from almost two minutes using the previous only-multithreaded approach to less than ten seconds using the hybrid method.

Conclusions: The hybrid MPI/multithreaded implementation can be used by medical experts to obtain tear film maps in only a few seconds, which will significantly accelerate and facilitate the diagnosis of the dry eye syndrome.

Keywords: Dry Eye Syndrome, Tear Film Map, Parallel Programming, High Performance Computing, Message Passing

1. Introduction

Dry eye syndrome is a chronic, multifactorial disease of the tears and the ocular surface [1], with an increasing prevalence in the last few years, reaching from 10 to 35% of the general population [2]. It has a negative impact on several common tasks of daily living, such as driving or working with computers. For this reason, it is recognized as a growing public health problem which deserves increased attention and resources [3].

The severity of dry eye is correlated to lipid layer thickness [4], and one of the most common diagnostic tests consists in analyzing the interference patterns observed in the tear film lipid layer of the eye. Guillon designed an instrument for rapid assessment of the tear film thickness

*Principal corresponding author: Jorge González-Domínguez
Email addresses: jgonzalezd@udc.es (Jorge González-Domínguez), bremeseiro@fe.up.pt (Beatriz Remeseiro), mariam@udc.es (María J. Martín)

known as Tearscope Plus [5]. In order to facilitate the use of this instrument, he also defined a grading scale composed of five interference patterns, which in increasing order of thickness are: open meshwork, closed meshwork, wave, amorphous, and color fringe.

This method offers a valuable technique to evaluate the quality and structure of the tear film in a non-invasive way, but it is affected by the subjective interpretation of the observer and by an adequate training [6]. These facts support the use of a systematic, objective computerized system for analysis and classification of interference patterns. For this reason, an automatic version of this clinical test was presented in [7], which includes the characterization of the interference patterns by means of color and texture properties, and a optimization step to extract them in real-time.

Due to the heterogeneity of the tear film lipid layer, in which multiple patterns may be observed, the classification of a Tearscope image into one single category is not always possible. Therefore, the definition of tear film maps was proposed in [8] by means of a weighted voting system based on distances and probabilities. An adapted version of the classic seeded region algorithm was subsequently presented in [9] to improve tear film definition, not only regarding its accuracy (from 80% to 90%), but also in terms of processing time (from more than 60 minutes to less than 10).

However, the time needed to define tear film maps may prevent their clinical use. Parallel computing can be used in order to reduce this runtime and increase their acceptance among medical experts. A multithreaded implementation that provides the same accuracy and further reduces the runtime to around 2 minutes was presented in [10]. This work is a step forward to further accelerate the tear film map definition using Message Passing Interface (MPI) [11], reducing runtime to only a few seconds.

This paper is organized as follows: Section 2 explains the background necessary to understand the rest of the manuscript, Section 3 presents the parallel approach proposed in this research, Section 4 shows the experimental results and discussion, Section 5 includes the related work and, finally, Section 6 includes the conclusions.

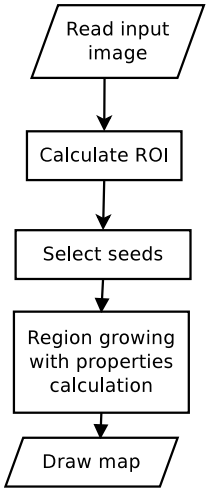

Figure 1: Workflow of the sequential algorithm.

2. Background: tear film mapping

The definition of tear film maps allows to detect multiple patterns per patient, and provides a complete and useful information to support dry eye diagnosis. The algorithm for tear film mapping used in this research was proposed in [9], and is summarized in Figure 1.

Tearscope images include irrelevant parts of the eye such as the pupil or the sclera. Thus, the first step consists in locating the Region of Interest (ROI), in which the tear film map is defined. It is a ring-shaped area automatically located at the most illuminated part of the iris from which the eyelashes, or shadows cast by them, have been removed (see Figure 2).

Next, the ROI is analyzed at a local level using image patches, i.e. small groups of nearby pixels. For each image patch, a feature descriptor and its corresponding probability vector are calculated. The features correspond to color and texture properties, while the probabilities represent the level of membership to each of the five categories proposed by Guillon [5] and mentioned in Section 1.

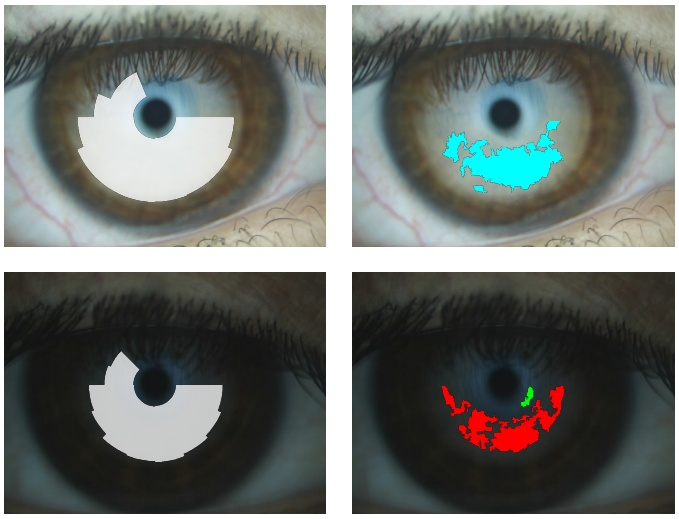

Figure 2: From left to right, ROIs and tear film maps represented over two input images.

Based on this information, tear film maps are created by applying an adapted version of the classic seeded region growing algorithm [12]. First, the seeds, i.e. the initial points of the regions, are automatically selected. The non-overlapped patches inside the ROI are analyzed in the terms previously explained and their maximum probability (p_{max}) is compared with a threshold (α). The seeds are the center of those patches which satisfy $p_{max} > \alpha$, and their corresponding regions are labelled as class i, where $p_{max} = p[i]$.

Then, the region growing is applied by analyzing the neighborhood of each region in such a way that a neighbor is added to a region if it satisfies $\delta < \beta$, where δ is the so-called homogeneity criterion, and β is the growing threshold. Note that this criterion is defined as: $\delta = |p[i] - avg[i]|$, where i is the class of the region, p is the probability vector of the neighbor, and avg is the average probabilities calculated over the region's pixels.

Next, the regions obtained are post-processed in order to eliminate the small areas that may correspond to false positives or noise. Finally, a set of colors is used to represent the five interference patterns and to illustrate the tear film map (see Figure 2).

2.1. Parallel implementation on multicore platforms

To make an efficient use of current multicore processors the sequential applications has to be redesigned so that they are executed using multiple threads. In [10] two multithreaded approaches to accelerate the definition of tear film maps on multicore machines were implemented using the multithreading support available in C++11 standard:

- On-demand approach (Figure 3). All the steps but the region growing (the most computationally expensive) are sequential. The parallelism is included in this step by distributing the initial seeds among the threads and analyzing them in parallel. The distribution of seeds to threads can be performed statically or dynamically.

- Full approach (Figure 4). This proposal includes an additional initial step that processes the whole input image to calculate all the feature vectors and probabilities of each point inside the ROI. Once finished this first step, it behaves like the original sequential version since the probabilities previously computed are simply accessed to evaluate the homogeneity criterion. The multithreaded parallelization is included in the new additional step, as it is very computationally demanding. As the descriptors and probabilities are previously computed, the region growing step is very fast in this implementation.

3. Parallel implementation on multicore clusters

A performance analysis of the sequential algorithm to define tear film maps shows that the bottleneck is gathered in one single step: the seeded region growing. In this section we explain how to parallelize this step on distributed-memory systems such as multicore clusters and supercomputers.

Multicore clusters can be defined as distributed-memory systems that consist of several nodes interconnected through

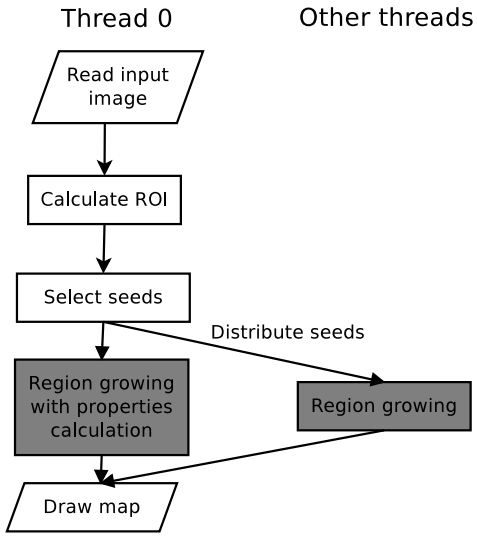

Figure 3: Workflow of the on-demand multithreaded algorithm. In gray the parallel steps.

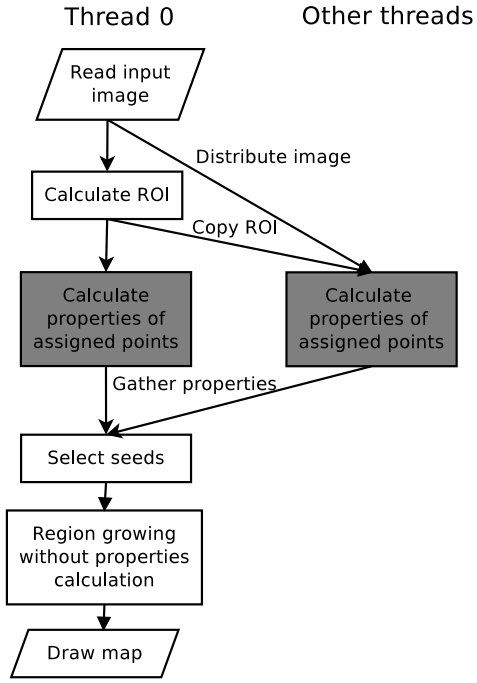

Figure 4: Workflow of the full multithreaded algorithm. In gray the parallel steps.

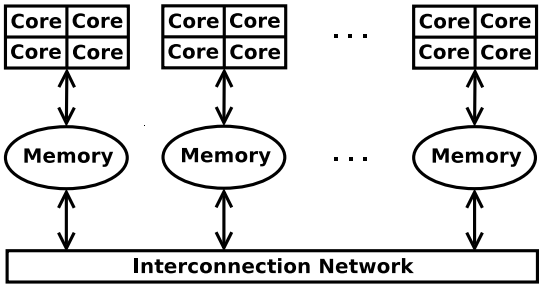

Figure 5: Abstraction of a distributed-memory system with several cores and one memory module per node.

a network, each of them with a memory module and several CPU cores (see Figure 5). Parallel computing on this kind of systems usually follows the Single Program Multiple Data style, i.e., it splits the workload into different tasks that are executed on multiple CPUs so that all nodes and cores collaborate to accelerate computation. The computational capability of the cluster depends on factors such as the number of nodes, the number of cores per node, the network characteristics, the memory bandwidth, etc.

The most common programming model for high-performance clusters systems is message-passing. MPI [11] is established as a *de-facto* standard for message-passing as it is based on the consensus of more than 40 participating organizations, including vendors, researchers, software library developers, and users. MPI provides a portable, efficient, and flexible standard for message-passing. Note that it is only a definition of an interface, that has been implemented by several developers for different architectures. Nowadays there exists a plethora of implementations whose routines or functions can be directly called from C, C++ and Fortran code.

A parallel MPI program consists of several processes with associated local memory. In a pure MPI program each process is linked to one core. In hybrid MPI and multithreaded programs each process is usually mapped to one node and it has several associated threads (often the same number of threads as cores within the node). If the tasks of different processes are completely indepen-

dent they do not need to exchange information. Otherwise, data communication must be performed. The traditional MPI communication style is two-sided, i.e. the source and destination processes must be synchronized through send and receive routines. MPI also provides collective routines for communication patterns that involve a group of processes. These collectives are usually very efficient as there exist optimized versions for specific architectures [13, 14].

3.1. MPI-based tear film mapping

Our experimental evaluation in [10] showed that the on-demand approach with dynamic distribution obtains the best performance on multicore platforms, but also that its scalability is limited (it never scales for more than 32 threads). Two reasons cause this limitation. On the one hand, the parallelism consists in different threads simultaneously performing the region growing of different seeds, but the time of the region growing depends on the region size, and might be significantly variable. Consequently, cores working over larger regions would work longer while other have already finished, and thus the parallel capability of the system is not fully exploited. On the other hand, all threads need to share a variable that identifies the next region to analyze. Accesses to this variable are synchronized using locks to avoid race conditions, which leads to additional overhead. As the impact of these two drawbacks increases with the number of total cores, we discarded the on-demand version for a MPI extension.

In the full approach the scalability is higher as the computation on each core is completely independent and the workload is well balanced (the calculation cost for each point is the same and we distribute the same number of points to each core). The main drawback is that it analyzes all points inside the ROI, and some of them are not necessary for the region definition, while the on-demand approach only computes the feature descriptors and probability vectors associated to those pixels which are in the neighborhood of the regions. It means that the sequential

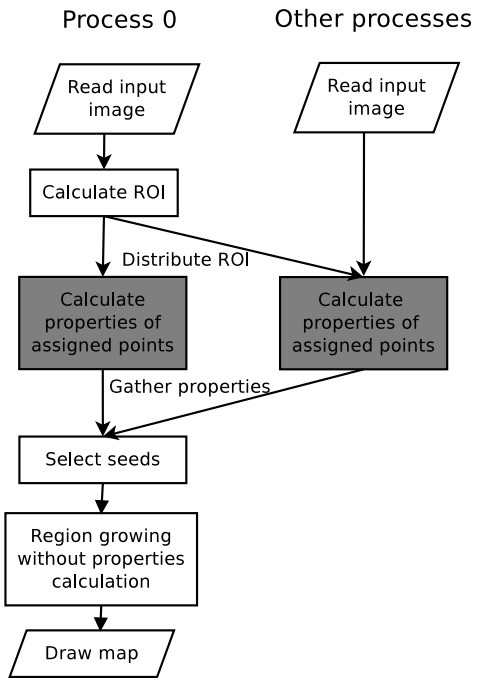

Figure 6: Workflow of the pure MPI algorithm. In gray the parallel steps.

time is higher compared to the on-demand version. Nevertheless, the good scalability and the use of additional parallel resources should overcome the overhead caused by calculating the properties of unnecessary points. Thus, we use this full approach as base for the MPI implementation.

Figure 6 illustrates the adaptation of the full approach to be executed using several MPI processes. All the processes simultaneously read the input image using the efficient parallel MPI I/O library. Thus, the input image is replicated on the different local memories and directly accessible to all processes. The second step consists in the calculation of the ROI. As this step is extremely fast, it is executed in sequential. Once the ROI is calculated, its points are distributed among processes with the corresponding collective routine (*MPI_Scatter*) and each process calculates in parallel the properties of its assigned points. As the cost of the properties calculation is the same for every point, they are distributed in a round-robin way that ensures the same number of points per process.

The region growing step is not adequate for parallel

computation using message-passing. We could distribute the seeds among the processes, but the work of different processes would not be independent. During the growing procedure of each seed the tear film map changes, and these advances have influence on the growing of other seeds. It means that the state of the tear film map should be shared by all processes, which is extremely expensive in MPI as it requires many communications and synchronizations. Consequently, the region growing procedure is performed by only one process. Nevertheless, the cost of this step is significantly lower than the previous calculation of properties so its MPI parallelization is not critical for performance. In order to perform the region growing the responsible process requires the properties of all the ROI points in its local memory. They are collected with another MPI collective (*MPI_Gather*). The convenience of using collectives instead of point-to-point messages for communications that involve several MPI processes was addressed at the beginning of this section. Finally, the same process also draws the output tear film map. This is a sequential step because its computational cost is almost negligible.

Remark that, besides adapting this approach to the MPI features (replication of the input image, distribution of the ROI among processes, gather of all properties before the region growing, etc.), we have included an additional optimization that is not present in the algorithm of the previous full multithreaded implementation (Figure 4). In the MPI approach only the points inside the ROI are distributed among processes. In the previous multithreaded approach all points of the input image were taken into account for the data distribution and every thread had to check whether the points belong to the ROI.

3.2. Hybrid MPI/multithreaded parallelization

There is a current trend to develop hybrid MPI and multithreaded programs where each process is associated to a group of cores (usually all the cores within one node of the cluster) and it launches several threads to map its tasks among the cores of the group. These hybrid applications usually outperform only-MPI codes on multicore clusters [15, 16]. We use the C++11 support for multithreading to optimize our MPI implementation by launching several threads on two steps of the algorithm:

- In a multicore cluster with N nodes and C cores per node a pure MPI implementation would use $N \cdot C$ processes for the parallel calculation of the properties of the pixels inside the ROI. A typical configuration for our hybrid approach execution creates only N MPI processes (one per node), each one with C threads that collaborate to calculate the properties of the points assigned to their parent process. The benefit of the hybrid implementation is that the number of processes is reduced but all the cores of the system are exploited. Consequently, less processes are involved in the collective routines (*MPI_Scatter* and *MPI_Gather*) which improves communication performance. Furthermore, the memory overhead due to replicating the input image is also reduced, as the threads can work over the same image and thus only one copy per node and MPI process is needed.

- Although the region growing step is much faster than in the original sequential algorithm (thanks to the precalculation of the feature descriptors and probability vectors), its computational cost is not completely negligible. MPI parallelization of the region growing is not advisable as it would require many messages to share the current state of the tear film map (see Section 3.1). However, in our hybrid implementation the MPI process responsible of this step launches several threads and each one performs the growing of different seeds in parallel. Therefore, all cores of one node collaborate in this step even when using only one MPI process. As threads share the

same memory space, modifications in the map performed by one thread are directly accessible to the others without any communication overhead. Remark that the full implementation in [10] does not include multithreaded parallelization of the region growing, which is another novel optimization of this work.

Algorithm 1 shows the hybrid algorithm that generates the matrix of regions R that will be drawn. The MPI collectives indicated in the previous subsection are present in lines 1 and 5. The most computationally expensive part (i.e. the calculation of the properties for the whole ROI) is performed in parallel in lines 3 and 4. Then, the rest of computation is completed only by process 0, but several threads collaborate for the region growing step. Different threads work over different points of the SSL list as explained in Section 2 (lines from 10 to 28).

Figure 7 visually summarizes the parallel behaviour of the hybrid approach. Our implementation is flexible enough to work with different numbers of processes and threads. The configuration is indicated through command line. Anyway, according to the experimental results (see Section 4), the use of one process per node is always the best configuration especially thanks to the reduction of the communication overhead and the increase of the parallelism in the region growing.

4. Results

In order to evaluate the performance of our hybrid parallel tear film map implementation runtime tests were performed on 32 nodes of the Finis Terrae 2 supercomputer [17] at the Galicia Supercomputing Center (CESGA). Each node contains two 12-core Haswell 2680v3 processors (24 cores per node) and 128 GB of memory. They are connected through an InfiniBand FDR network with 56 Gbps of theoretical effective bandwidth. All the experiments were built with the Intel MPI and ICC compilers version 16.0.3 and the -O3 optimizations.

Regarding the images, the experimentation was performed with the VOPTICAL_R dataset [18] which contains 50 images of the preocular tear film with a resolution of 1024×768 pixels. All the images were manually annotated by experts who delimited those regions associated with the five interference patterns. Note that the time needed to generate tear film maps is variable since the region-growing runtime depends on the region sizes. Furthermore, the runtime also depends on the ROI size, as all pixels outside the ROI are not processed.

The experimental evaluation started by comparing the output images provided by each implementation. We conclude that there is no significant difference in the accuracy of the different versions, in comparison with the experts' annotations. Therefore, from now on we will only focus on performance in terms of runtime.

Table 1 shows the runtime of our hybrid parallel implementation for varying number of nodes. One process and 24 threads per node are used in all cases in order to completely exploit the computational capability of each node. Intermediate configurations (more processes per node and less threads per process) are discarded because preliminary evaluations proved that they are less efficient as more processes are involved in the reduction of the feature descriptors and probability vectors of the whole ROI. Three values are shown for each scenario: the average, maximum and minimum execution time of the 50 images of the dataset. The results for the best only multithreaded approach presented in [10] (on-demand version) using the 24 cores of one node are also included for comparison purposes.

The main conclusion that can be obtained is that the hybrid parallel implementation significantly accelerates the generation of tear film maps compared to the previous multithreaded approaches. For instance, the average runtime is reduced from 50 seconds using the best multithreaded version to only 5 seconds with the hybrid implementation on the 32 nodes (8.42 times faster). This effect is even

Algorithm 1: Pseudo-code of the hybrid parallel algorithm.

Data: matrix of points in the region of interest ROI, growing threshold β, MPI rank $myRank$, thread id $myId$

Result: matrix of regions R

```
1  MPI_Scatter(ROI) to all processes
2  initialize matrix of probabilities p := 0
3  for each point t ∈ ROI in parallel all processes and threads do
4  |  p[t] := calculateProperties(t)
   end
5  MPI_Gather(p) to process 0
6  if myRank == 0 then
7  |  if myId == 0 then
8  |  |  initialize list of seeds L := selectSeeds(ROI)
   |  end
9  |  initialize matrix of regions R := 0 and sequentially sorted list SSL := φ
10 |  for each seed s ∈ L in parallel all threads of process 0 do
11 |  |  i := getLabel(s)
12 |  |  R[s] := i
13 |  |  for each neighbor n ∈ getNeighbors(s) do
14 |  |  |  δ = |p[i] − avg[i]|
15 |  |  |  add(SSL, n, i, δ))
   |  |  end
   |  end
16 |  while notEmpty(SSL) in parallel all threads of process 0 do
17 |  |  y := pushFirst(SSL)
18 |  |  N = getLabeledNeighbors(y) and removeBoundaryNeighbors(N)
19 |  |  if sameLabel(N) then
20 |  |  |  i := getLabel(N)
21 |  |  |  δ := getDelta(y)
22 |  |  |  if δ < β then
23 |  |  |  |  R[y] := i
24 |  |  |  |  update(avg[i])
25 |  |  |  |  for each neighbor n ∈ getNoLabeledNeighbors(y) do
26 |  |  |  |  |  δ = |p[n] − avg[i]|
27 |  |  |  |  |  add(SSL, n, i, δ))
   |  |  |  |  end
   |  |  |  end
   |  |  else
28 |  |  |  R[y] := −1
   |  |  end
   |  end
   end
```

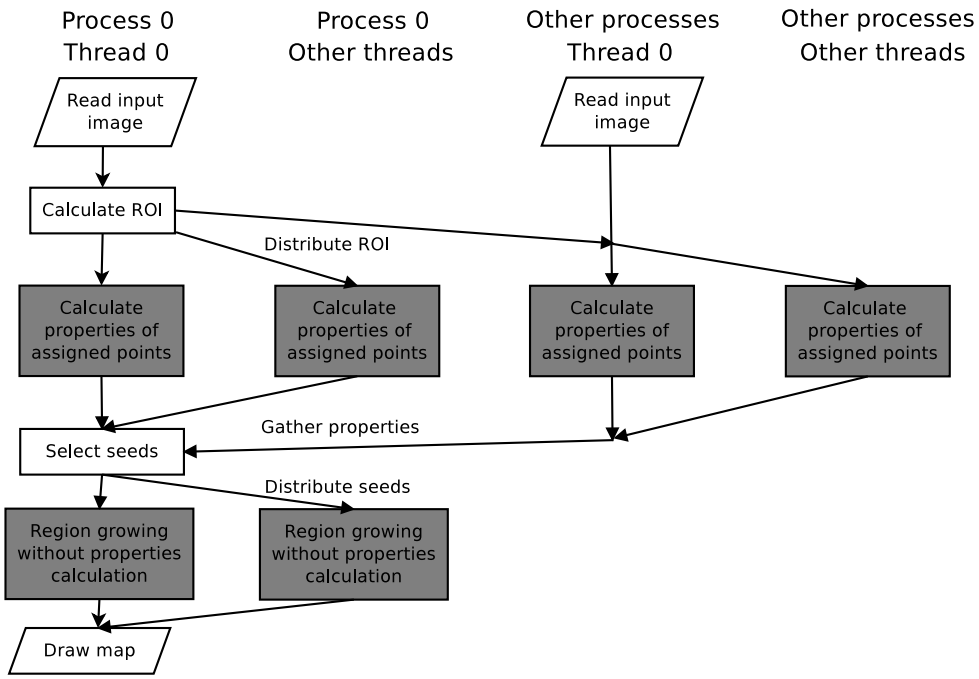

Figure 7: Workflow of the hybrid MPI/multithreaded algorithm. In gray the parallel steps.

Table 1: Runtime (in seconds) of the hybrid parallel implementation. The results are the average, maximum and minimum runtime obtained from the analysis of 50 images. We present results for different number of nodes using one process and 24 threads per node, and compared to the best multithreaded version (on-demand) with 24 threads.

Version	Nodes	Avg	Max	Min
Hybrid full	1	151.21	252.43	73.06
	2	74.74	128.39	37.05
	4	42.39	68.25	21.90
	8	19.94	33.59	9.21
	16	11.23	18.13	5.79
	32	5.96	9.43	3.32
On-demand	1	50.20	117.93	12.59

more significant for the most computationally demanding image, where the acceleration is up to 12.50 (time reduction from almost two minutes to less than 10 seconds).

Figures 8 and 9 provide an insight of the speedups when comparing the hybrid implementation to the two different multithreaded ones. These speedups are calculated as: $speedup = \frac{T_{mult}}{T_{hyb}}$, where T_{mult} and T_{hyb} are the execution times of the multithreaded and hybrid approaches, respectively. Besides the performance gain over the on-demand approach, these figures show that the scalability of the MPI code itself is very high. The average and maximum speedups of the hybrid implementation on 32 nodes over its base version (multithreaded full) are 25.37 and 26.76, respectively, which means parallel efficiency around 80%. Furthermore, the original sequential implementation (one process and one thread) presented in [9] needed up to half hour to complete the tear film map of the most computationally demanding image and now, with the hybrid full version, the map is obtained in only a few seconds. This proves the adequacy of parallel computation for the acceleration of this problem.

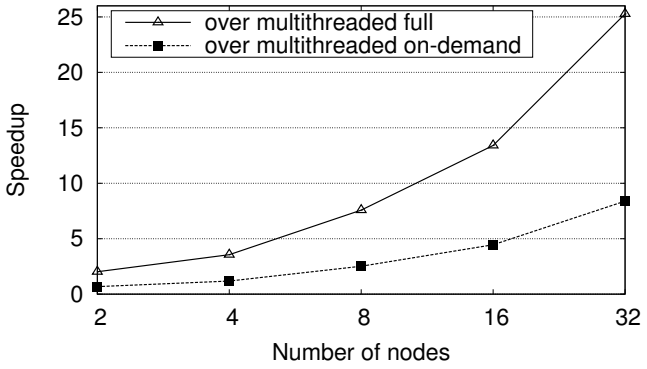

Figure 8: Average speedup of the hybrid parallel implementation using up to 32 nodes (one process and 24 threads per node) over the multithreaded full and on-demand versions with 24 threads.

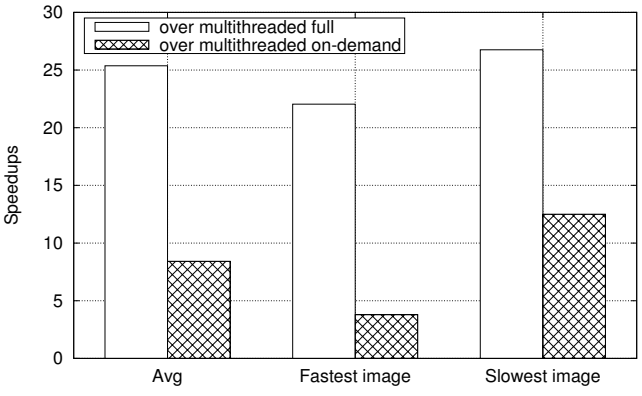

Figure 9: Speedup of the hybrid parallel implementation using 32 nodes (one process and 24 threads per node) over the multithreaded full and on-demand versions with 24 threads. Average speedups, as well as the values for the slowest and fastest images are presented.

On average, the use of four nodes is enough to outperform the best multithreaded version. However, even only two nodes are enough to decrease the runtime of 7 images (14%). This percentage increases to 72% and 100% for four and eight nodes, respectively. The reason why some images benefit more from the hybrid implementation is the variable ROI size. As explained in Section 3.1, the hybrid approach computes all the points of the ROI in a previous step to region growing, while not all of them are later explored. The higher the number of points of the ROI not explored in the region growing phase, the higher the overhead of the previous step and more nodes are necessary to overcome this overhead. This behavior is reflected in Figure 10. It shows that the hybrid implementation spends most of time in the additional step that computes all the points of the ROI, and this step is the main focus of the parallelization. Nevertheless, the region growing is the heaviest step of the multithreaded on-demand approach.

5. Related work

To the best of our knowledge, no other attempts have been made so far, in the literature or commercially, to provide automatic assessments of the tear film lipid layer patterns using Tearscope images. Other clinical instruments have been considered to automatically analyze the interference phenomena, such as the Doane's interferometer which was used to detect tear film breakup regions by means of a texture-based segmentation method [19], and to develop an automated grading system to categorize interferometry images based on color and texture properties [20]. Additionally, other diagnostic tests commonly used have been automated, including the breakup time (BUT) test [21, 22] and the non-invasive breakup time (NIBUT) test [23, 24].

There exist previous parallel algorithms based on region growing that solve biomedical problems. Some examples of multithreaded approaches are [25] and [26] to analyze teeth and abdominal images, respectively. Up to

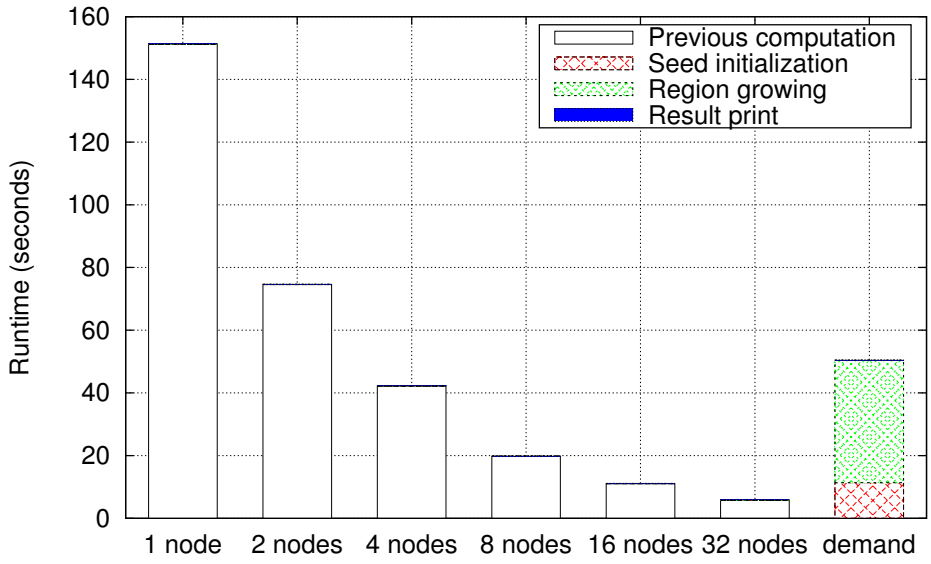

Figure 10: Average performance breakdown of the hybrid parallel implementation using up 32 nodes (one process and 24 threads per node) and the multithreaded on-demand version with 24 threads.

our knowledge, the only region growing based applications that can be executed on distributed-memory systems are [27] and [28]. The first one accelerates the detection of retinal blood vessels using the parallel library of Matlab, but its scalability is quite limited (only two times faster using 14 processes). The last reference is focused on brain images and needs GPUs installed in the nodes of the cluster to be fully optimized. Therefore, this work is the first one focused on the use of multicore clusters to improve the analysis of the interference patterns on the tear film lipid layer.

6. Conclusions

Dry eye syndrome is a public health problem, and one of the most common conditions seen by eye care specialists. Tear film maps can be employed as part of a clinical routine to diagnose dry eye since they illustrate the spatial distribution of the patterns over the whole tear film and provide useful information to practitioners. Our previous work [10] presented an automatic multithreaded method to draw tear film maps from an image of the tear film lipid layer with high accuracy and runtime of a few minutes on shared-memory systems with 16 and 64 cores. However, medical experts demand applications with *real time* response (a few seconds) in order to provide a faster diagnosis for their patients.

In this work we have presented a hybrid parallel implementation of the tear film map definition in order to fulfill the practitioners runtime requirements without loosing accuracy. The main workload (calculation of the feature descriptors and probability vectors of the ROI points) is distributed among several MPI processes. A second level of parallelization, with several threads per process, allows to reduce the memory overhead and the performance impact of the MPI communication collective routines. Furthermore, although the region growing must be performed by only one MPI process, the use of several threads also accelerates this step.

The experimental evaluation was performed on 32 nodes of an Intel cluster with 24 cores per node. A dataset with 50 tear film images was used in the evaluation. The experimental results using one process per node and 24 threads per process (the best configuration) show that the implementation provides high scalability. The average speedup

on 32 nodes compared to its base only multithreaded version is 25.37, with parallel efficiency around 80%. Compared to the best only multithreaded approach presented in [10], the execution on 32 nodes is on average 8.42 times faster. Remark that in the worst case the hybrid implementation on 32 nodes only takes 9.43 seconds while the best multithreaded approach needs around two minutes (speedup of 12.50).

Finally, a web-based system for dry eye assessment (iDEAS) was presented in [29]. iDEAS is a framework for eye care specialists to collaboratively work using image-based services in a distributed environment, which includes two automatic services: tear film classification and tear film map definition. Using this novel hybrid parallel implementation and a multicore cluster as server of the iDEAS web framework [29], several medical experts can automatically obtain the tear film maps on a few seconds.

Acknowledgements

This research has been partially funded by the Ministry of Economy and Competitiveness of Spain and FEDER funds of the EU (Project TIN2013-42148-P), by the Galician Government and FEDER funds of the EU (reference GRC2013/055), by the ERDF - European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project POCI-01-0145-FEDER-006961, and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia as part of project UID/EEA/50014/2013 and research grant SFRH/BPD/111177/2015.

We gratefully thank CESGA (Galicia Supercomputing Center, Santiago de Compostela, Spain) for providing access to the Finis Terrae II supercomputer. We would also like to thank the Optometry Service from the University of Santiago de Compostela (Spain) and the Center of Physics from the University of Minho (Portugal) for providing us with the annotated dataset.

References

[1] M. A. Lemp, C. Baudouin, J. Baum, M. Dogru, G. N. Foulks, S. Kinoshita, P. Laibson, J. McCulley, J. Murube, S. C. Pfugfelder, M. Rolando, I. Toda, The Definition and Classification of Dry Eye Disease: Report of the Definition and Classification Subcommittee of the International Dry Eye Workshop, The Ocular Surface 5 (2) (2007) 75–92.

[2] J. A. Smith, J. Albeitz, C. Begley, B. Caffery, K. Nichols, D. Schaumberg, O. Schein, The Epidemiology of Dry Eye Disease: Report of the Epidemiology Subcommittee of the International Dry Eye WorkShop (2007), The Ocular Surface 5 (2) (2007) 93–107.

[3] B. Miljanović, R. Dana, D. A. Sullivan, D. A. Schaumberg, Impact of dry eye syndrome on vision-related quality of life, American Journal of Ophthalmology 143 (3) (2007) 409–415.

[4] G. N. Foulks, The correlation between the tear film lipid layer and dry eye disease, Survey of Ophthalmology 52 (4) (2007) 369–374.

[5] J.-P. Guillon, Non-invasive Tearscope Plus routine for contact lens fitting, Contact Lens and Anterior Eye 21 (Suppl 1) (1998) S31–S40.

[6] J. J. Nichols, K. K. Nichols, B. Puent, M. Saracino, G. L. Mitchell, Evaluation of tear film interference patterns and measures of tear break-up time, Optometry & Vision Science 79 (6) (2002) 363–369.

[7] B. Remeseiro, V. Bolón-Canedo, D. Peteiro-Barral, A. Alonso-Betanzos, B. Guijarro-Berdiñas, A. Mosquera, M. G. Penedo, N. Sánchez-Maroño, A Methodology for Improving Tear Film Lipid Layer Classification, IEEE Journal of Biomedical and Health Informatics 18 (4) (2014) 1485–1493.

[8] B. Remeseiro, A. Mosquera, M. G. Penedo, C. García-Resúa, Tear film maps based on the lipid interference patterns, in: 6th International Conference on Agents and Artificial Intelligence (ICAART), Vol. 1, Angers, France, 2014, pp. 732–739.

[9] B. Remeseiro, A. Mosquera, M. G. Penedo, CASDES: a Computer-Aided System to Support Dry Eye Diagnosis Based on Tear Film Maps, IEEE Journal of Biomedical and Health Informatics 20 (3) (2016) 936–943.

[10] J. González-Domínguez, B. Remeseiro, M. J. Martín, Acceleration of tear film map definition on multicore systems, in: International Conference on Computational Science (ICCS), Vol. 80, San Diego, USA, 2016, pp. 41–51.

[11] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard Version 3.1, [Online] Available: `http://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf` (2015).

[12] R. Adams, L. Bischof, Seeded Region Growing, IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (6) (1994)

641–647.

[13] A. R. Mamidala, R. Kumar, D. De, D. K. Panda, MPI Collectives on Modern Multicore Clusters: Performance Optimizations and Communication Characteristics, in: 8th International Symposium on Cluster, Cloud and Grid Computing (CCGRID), Lyon, France, 2008, pp. 130–137.

[14] B. Tu, J. Fan, J. Zhan, X. Zhao, Performance Analysis and Optimization of MPI Collective Operations on Multi-Core Clusters, The Journal of Supercomputing 60 (1) (2012) 141–162.

[15] M. J. Chorley, D. W. Walker, Performance Analysis of a Hybrid MPI/OpenMP Application on Multi-Core Clusters, Journal of Computational Science 1 (3) (2010) 168–174.

[16] A. Amer, H. Lu, P. Balaji, S. Matsuoka, Characterizing MPI and Hybrid MPI+Threads Applications at Scale: Case Study with BFS, in: 15th International Symposium on Cluster, Cloud and Grid Computing (CCGRID), Shenzhen , China, 2015, pp. 1075–1083.

[17] Finis Terrae 2 at Galicia Supercomputing Center (CESGA), [Online] Available: `https://www.cesga.es/en/infraestructuras/computacion/FinisTerrae2` (2016).

[18] VOPTICAL_R, VARPA optical dataset acquired and annotated by optometrists from the Optometry Service of the University of Santiago de Compostela (Spain) and the Physics Center of the University of Minho (Portugal), [Online] Available: `http://www.varpa.es/voptical\_r.html` (2015).

[19] D. Wu, K. L. Boyer, J. J. Nichols, P. E. King-Smith, Texture based prelens tear film segmentation in interferometry images, Machine Vision and Applications 21 (3) (2010) 253–259.

[20] B. Remeseiro, K. M. Oliver, A. Tomlinson, E. Martin, N. Barreira, A. Mosquera, Automatic grading system for human tear films, Pattern Analysis and Applications 18 (3) (2015) 677–694.

[21] L. Ramos, N. Barreira, A. Mosquera, M. G. Penedo, E. Yebra-Pimentel, C. García-Resúa, Analysis of parameters for the automatic computation of the tear film break-up time test based on CCLRU standards, Computer Methods and Programs in Biomedicine 113 (3) (2014) 715–724.

[22] H. Pena-Verdeal, C. García-Resúa, L. Ramos, E. Yebra-Pimentel, M. J. Giráldez, Diurnal variations in tear film break-up time determined in healthy subjects by software-assisted interpretation of tear film video recordings, Clinical and Experimental Optometry 99 (2) (2016) 142–148.

[23] W. Lan, L. Lin, X. Yang, M. Yu, Automatic noninvasive tear breakup time (TBUT) and conventional fluorescent TBUT, Optometry & Vision Science 91 (12) (2014) 1412–1418.

[24] A. Carpente, L. Ramos, N. Barreira, M. G. Penedo, H. Pena-Verdeal, M. J. Giraldez, On the Automation of the Tear Film Non-invasive Break-up Test, in: IEEE 27th International Symposium on Computer-Based Medical Systems (CBMS), New York, USA, 2014, pp. 185–188.

[25] H. C. Kang, C. Choi, J. Shin, J. Lee, Y. Shin, Fast and Accurate Semiautomatic Segmentation of Individual Teeth from Dental CT Images, Computational and Mathematical Methods in Medicine 2014 (art. 810796) (2014) 1–12.

[26] S. Saxena, N. Sharma, S. Sharma, An Intelligent System for Segmenting an Abdominal Image in Multi core Architecture, in: 10th International Conference on Emerging Technologies for a Smarter World (CEWIT), New York, USA, 2013, pp. 1–6.

[27] M. A. Palomera-Pérez, M. E. Martínez-Pérez, H. Benítez-Pérez, J. L. Ortega-Arjona, Parallel Multiscale Feature Extraction and Region Growing: Application in Retinal Blood Vessel Detection, IEEE Transactions on Information Technology in Biomedicine 14 (2) (2010) 500–506.

[28] A. M. Westhoff, Hybrid Parallelization of a Seeded Region Growing Segmentation of Brain Images for a GPU Cluster, in: 27th GI/ITG International Conference on Architecture of Computing Systems (ARCS), Luebeck, Germany, 2014, pp. 1–8.

[29] B. Remeseiro, N. Barreira, C. García-Resúa, M. Lira, M. J. Giráldez, E. Yebra-Pimentel, M. G. Penedo, iDEAS: A web-based system for dry eye assessment, Computer Methods and Programs in Biomedicine 130 (2016) 186–197.