

A Project Report On

Connected Components Labelling

For the course

Distributed Computing Systems

Submitted by

Divija Nagaraju (14IT112)

Mukta Kulkarni (14IT220)

Pooja Soundalgekar (14IT230)

Yogitha A. N. (14IT251)

V Sem B.Tech (IT)

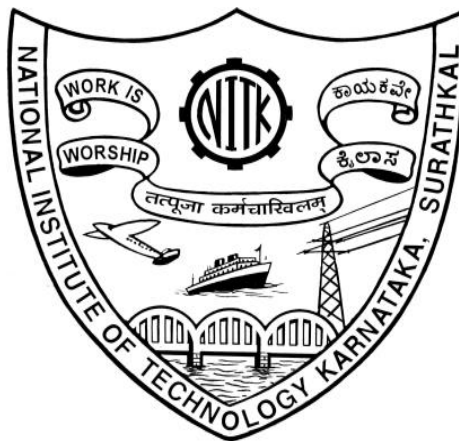
Under The Guidance of

Ms. Sangeetha S. H.

Asst. Lecturer

Information Technology

At



Department of Information Technology

National Institute of Technology Karnataka, Surathkal.

November 2016

CERTIFICATE

This is to certify that the project entitled “Database System for Ecommerce Website” is a bonafide work carried out as part of the course Database Management Systems (IT301), under my guidance by

1. Divija Nagaraju – 14IT112
2. Mukta Kulkarni - 14IT220
3. Pooja Soundalgekar - 14IT230
4. Yogitha A. N. – 14IT251

students of V Sem B.Tech (IT) at the Department of Information Technology, National Institute of Technology Karnataka, Surathkal, during the academic semester Jul - Dec 2016 in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Information Technology, at NITK Surathkal.

Place:

Date:

Signature of the Instructor

ABSTRACT

Connected-component labelling (alternatively called region extraction) is an algorithmic application of graph theory, where subsets of connected components are uniquely labeled based on a given heuristic. It is used in computer vision to detect connected regions in binary digital images, although colour images and data with higher dimensionality can also be processed. This project is an implementation of Two pass algorithm i.e. the concepts mentioned in the second category. The algorithms are implemented in C/C++ using MPI. MPI stands for Message Passing Interface and is a technique majorly used to implement distributed algorithms. An important characteristic of the MPI implementation is that the processors within the system do not have access to the shared memory. All the processors perform computation by message passing, thus successfully computing in a distributed environment.

TABLE OF CONTENTS

| | |
|--|-------|
| ABSTRACT | i |
| TABLE OF CONTENTS | ii |
| ACKNOWLEDGEMENT | iii |
| LIST OF FIGURES | iv |
| LIST OF TABLES | v |
| 1. INTRODUCTION | 1 |
| 1.1 Report Organisation..... | 2 |
| 2. LITERATURE SURVEY | 3-4 |
| 2.1 Previous Studies..... | 3-4 |
| 2.2 Outcome of Literature Review..... | 4 |
| 3. PROBLEM DEFINITION | 5 |
| 4. METHODOLOGY | 6-9 |
| 5. REQUIREMENTS AND DESIGN | 10 |
| 6. IMPLEMENTATION | 11-13 |
| 7. CONCLUSION AND FUTURE WORK | 14 |
| REFERENCES | |

ACKNOWLEDGEMENT

We would like to take this opportunity to thank and express our gratitude to all those without whom the completion of this project would not have been possible. This project has helped expand our horizons and improved our analytical skills. We would like to extend our thanks to our project advisor and project co-ordinator, Ms. Sangeetha S. H., Asst. Lecturer , Department of Information Technology for her constant help and support. We express my heartfelt gratitude to our HOD, Dr. Ram Mohan Reddy Department of Information Technology who has extended support, guidance and assistance for the successful completion of the project. Finally, we would like to thank all our classmates, friends and seniors for their suggestions and support.

Divija Nagaraju

Mukta Kulkarni

Pooja Soundalgekar

Yogitha A.N.

LIST OF FIGURES

| | |
|---|----|
| Figure 2.1 Decision tree for labelling a pixel..... | 3 |
| Figure 2.2 Image Labelling Mask..... | 3 |
| Figure 6.1 Screen shot for readings on processors of one system..... | 11 |
| Figure 6.2 Screen shot for readings on processors of two systems..... | 11 |

LIST OF TABLES

| | |
|--|----|
| Table 1: The execution time given n distributed processors on same core. | 10 |
| Table 2: The execution time given n distributed processors on different cores..... | 10 |

CHAPTER 1

INTRODUCTION

One of the most fundamental operations in pattern recognition is the labelling of connected components in a binary image. Connected component labelling (CCL) is a procedure for assigning a unique label to each object (or a connected component) in an image. Because these labels are key for other analytical procedures, connected component labelling is an indispensable part of most applications in pattern recognition and computer vision, such as finger print identification, character recognition, automated inspection, target recognition, face identification, medical image analysis, and computer-aided diagnosis. In many cases, it is also one of the most time-consuming tasks among other pattern-recognition algorithms. Therefore, connected component labelling continues to be an active area of research.

There exist many algorithms for computing connected components in a given image. These algorithms are categorized into mainly four groups : 1) repeated pass algorithms 2) two-pass algorithms 3) Algorithms with hierarchical tree equivalent representations of the data parallel algorithms. This project analyses and implements the approach of two pass computation and implements in a distributed computing system.

1.1 Report Organisation

Chapter 2 contains the Literature Survey describing various algorithms used for labelling connected components in binary images in the past. Chapter 3 defines the problem statement. Chapter 4 defines the methodology and the algorithm implemented. Chapter 5 describes the requirements and design of the distributed system. Chapter 6 consists of screenshots of the implemented algorithm. Finally, Chapter 7 includes conclusion and future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Previous Studies

The algorithms [1-3] repeat passes through an image in forward and backward directions alternately to propagate the label equivalences until no labels change. Suzuki et al. [1] classified connected algorithm which optimizes the algorithms in this category. Suzuki's algorithm promises a linear time complexity. Suzuki et al. show that maximum of four scans is sufficient for the images with complex geometrical shapes. Wu et al. [2] provide an optimization on Suzuki algorithm that reduces the number of neighbouring pixels needed to be examined during the scans. He et al. [3], present a similar algorithm which scans the whole image once and then resolves the label equivalences using recorded run data.

In any two-pass algorithm[4], there are two steps in scanning step: 1) examining neighbours of current pixel which already assigned labels to determine label for the current pixel, 2) storing label equivalence information to speed up the algorithm. Thus the two pass algorithm scans and labels two pixels at a time and computationally speeds up the labelling process. There are two strategies to implement the two pass algorithm. First strategy employs a decision tree, which reduces the average number of neighbours accessed by a factor of two. Second strategy replaces the conventional pointer based union-find algorithm, which is used for storing label equivalence, by adopting array based union-find algorithm that uses less memory. The union-find algorithm is implemented using Link by Rank and Path Compression technique.

The first part of the two pass strategy employs a scanning technique, which processes image two lines at a time and process two image pixels at a union-find implementation thus allows us to process the pixels at a time. Another strategy uses decision tree and for the second part we have union-find approach instead of Link by Rank and Path Compression technique.

The algorithm for two pass strategy as described above for connected components labelling described in [4] is as follows in Algorithm 1:

Algorithm 1 Pseudo-code for CCLREMSP

Input: 2D array *image* containing the pixel values

Output: 2D array *label* containing the final labels

```

1: function CCLREMSP(image)
2:   Scan_CCLRemSP(image) ▷ Scan Phase of CCLREMSP
3:   ▷ count is the max label assigned during Scan Phase
4:   flatten(p, count)    ▷ Analysis Phase of CCLREMSP
5:   for row in image do    ▷ Labeling Phase of CCLREMSP
6:     for col in row do ▷ e is the current pixel to be labeled
7:       label(e) ← p[label(e)]
8:   end function

```

The connectivity mask and decision tree is as described in Figure 2.1 and 2.2 respectively

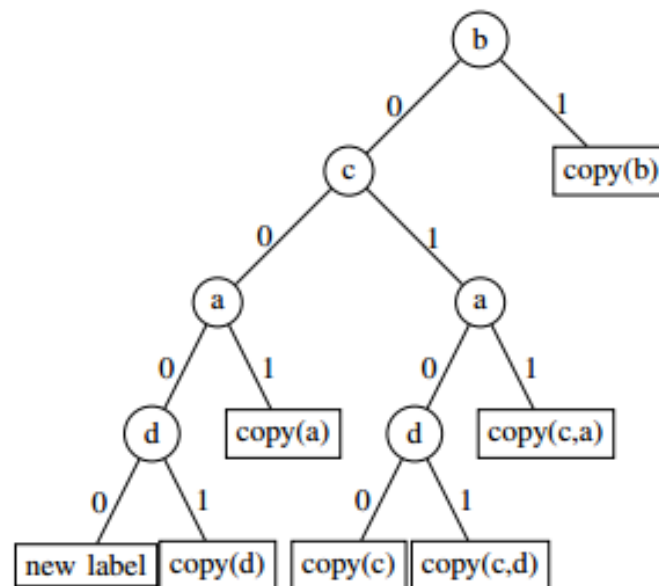


Figure 2.1 Decision tree for labelling a pixel

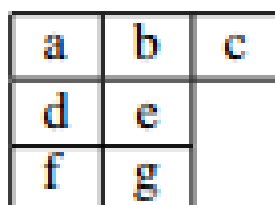


Figure 2.2 Image Labelling Mask

2.2 Outcome of the Literature Review

The literature survey in the above section reveals that the two pass algorithm is computationally better compared to the naive and suzuki algorithm. However as the image to be labelled increases in size, the two pass algorithm too takes considerable computation time. Hence the algorithm can be optimised using the concepts of distributed computing. The pixels that do not have any immediate dependency in the decision tree can be distributed over the systems and then joined together to obtain the result by distributed union-find flatten concept.

CHAPTER 3

PROBLEM STATEMENT

The project aims to compute the connected components in a binary image and label them. The algorithm used to do so is the two pass algorithm of category B. The implementation of the algorithm is in the distributed computing cluster system with no shared memory or common global clock. The algorithm is being coded in C language. The distributed nodes are made to interact with each other using Message Passing Interface (MPI). The computational speed for labelling the binary image is compared to the single system computational speed. Based on the result conclusive remarks are analysed and future work is being proposed.

CHAPTER 4

METHODOLOGY

Use of the decision tree suggested in the literature review for scanning and REM's union-find algorithm for storing label equivalence is made. In the first scan step, we process an image two lines at a time and two pixels at a time using the mask shown in Figure 2b. A label is assigned to both *e* and *g* simultaneously. If both *e* and *g* are background pixels, then nothing needs to be done. If *e* is a foreground pixel and there is no foreground pixel in the mask, we assign a new provisional label to *e* and if *g* is a foreground pixel, we will assign the label of *e* to *g*. If there are foreground pixels in the mask, then we assign *e* any label assigned to foreground pixels. In this case, if there is only one connected component in the mask then there is no need for label equivalence. Otherwise, if there are more than one connected components in the mask and as they are connected to *e*, all the labels of the connected components are equivalent labels and hence need to be merged.

We used a method similar to pipeline in order to exploit distribution. The following is a detail description of the proposed method to distribute the algorithm over multiple nodes in the cluster. At first the main function scans through the image two rows at a time. The image is divided in chunks of rows and distributed among the target distributed nodes. Each node computes a decision tree for a pixel within the allotted chunk. Since there is a dependency between a pixel and its previous pixel, all nodes cannot begin computation at once.

Assuming there are 2 nodes (N1 and N2) and the image has 128 rows and 128 columns. The first 64 pixels in the first row will be labelled by N1 and the next 64 pixels by N2. However, N2 cannot start labelling in the same row as N1, until N1 is done with labelling its portion due to the sequential nature of the Two Pass algorithm. The N2 can start labelling when N1 is done, also N1 can start labelling the first 64 pixels of the next row at the same time. After N1 is done with labelling the first portion of the second row, N2 can label its portion of the next row, and so forth. With this method, we maintain the order of execution of the algorithm as well as exploiting distribution of load over the distributed systems.

The conceptual algorithm for the distribution is given as follows in Algorithm 1:

Input is a matrix of 0 and 1 size $m \times n$

Output is a matrix of labelled components size $m \times n$

1. chunk= row/number of nodes
2. distribute total rows among available nodes
3. calculate start index k of image pixel for each node
4. while condition for node lock is not 1 make node wait
5. for k in row to chunk
 label the pixels
6. unset lock of consecutive node by setting it to 1

CHAPTER 5

DESIGN AND IMPLEMENTATION

5.1 Design

The Algorithm has been implemented in C using MPI (Message Passing Interface). Thus every system requires MPICH2-1.4 version.

The nodes must be connected within the same cluster. In the implementation of the project, the systems are connected within the same lan network. There is one master node and multiple client nodes. Every node has an entry of every other node in the cluster along with it's IP address.

5.2 Implementation

The merger of the union-find through decision tree computed is shown in Algorithm 3

Input: 1D array p and two nodes x and y
Output: The root of united tree

```

1: function MERGE( $p, x, y$ )
2:    $root_x \leftarrow x, root_y \leftarrow y$ 
3:   while  $p[root_x] \neq p[root_y]$  do
4:     if  $p[root_x] > p[root_y]$  then
5:       if  $root_x = p[root_x]$  then
6:          $p[root_x] \leftarrow p[root_y]$ 
7:         return  $p[root_x]$ 
8:        $z \leftarrow p[root_x], p[root_x] \leftarrow p[root_y], root_x \leftarrow z$ 
9:     else
10:      if  $root_y = p[root_y]$  then
11:         $p[root_y] \leftarrow p[root_x]$ 
12:        return  $p[root_x]$ 
13:       $z \leftarrow p[root_y], p[root_y] \leftarrow p[root_x], root_y \leftarrow z$ 
14:   return  $p[root_x]$ 
15: end function

```

The function that scans the image chunk at each node is given in the next Algorithm 4

Input: 2D array *image* containing the pixel values
InOut: 2D array *label* containing the provisional labels and 1D array *p* containing the equivalence info
Output: maximum value of provisional label in *count*

```

1: function SCAN_CCLREMSP(image)
2:   for row in image do
3:     for col in row do
4:       if image(e) = 1 then
5:         if image(b) = 1 then
6:           copy(b)
7:         else
8:           if image(c) = 1 then
9:             if image(a) = 1 then
10:              copy(c, a)
11:            else
12:              if image(d) = 1 then
13:                copy(c, d)
14:              else
15:                copy(c)
16:            else
17:              if image(a) = 1 then
18:                copy(a)
19:              else
20:                if image(d) = 1 then
21:                  copy(d)
22:                else
23:                  new label
24:   return count
25: end function

```

Thus the complete characterisation of the algorithm is given by integrating the algorithms given in Algorithm 2, 3 and 4 described earlier. Algorithm 2 is the base algorithm that calls functions scan and merge which are described above.

CHAPTER 6

RESULTS

The above algorithm is executed on 2 systems with given cores. The time taken for the execution is shown in table 1 wherein all processors are on the same core.

| No of Processors | Time For Execution (s) |
|------------------|------------------------|
| 1 | 0.020072 |
| 2 | 0.000082 |
| 4 | 0.000194 |

Table 1: The execution time given n distributed processors on same core.

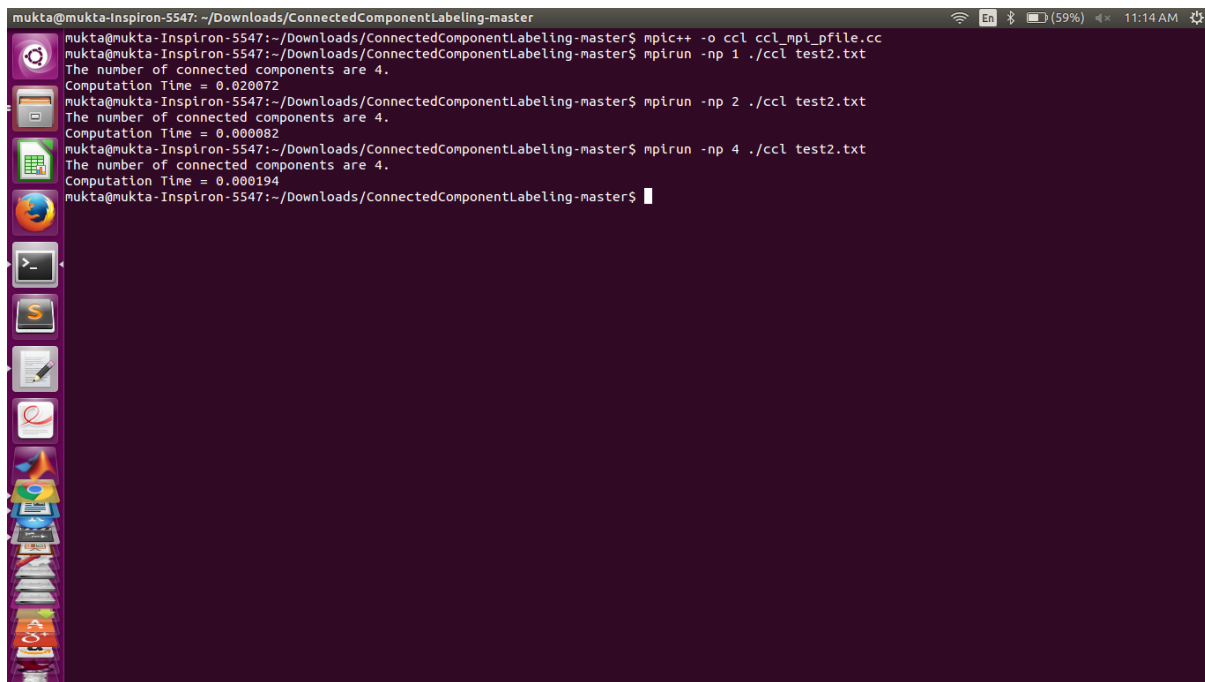
The time taken for the execution is shown in table 2 wherein processors are on the different cores or different systems.

| No of Processors | Time For Execution (s) |
|------------------|------------------------|
| 2 | 0.049428 |
| 4 | 0.000270 |

Table 2: The execution time given n distributed processors on different cores.

The results clearly indicate that the computation speed is faster if the algorithm is run on a distributed system. Another important result is that the computation is faster if all the processors are on the same core. This can be inferred from the fact that the communication cost between processors on the same cores is less than the communication cost between processors on different cores.

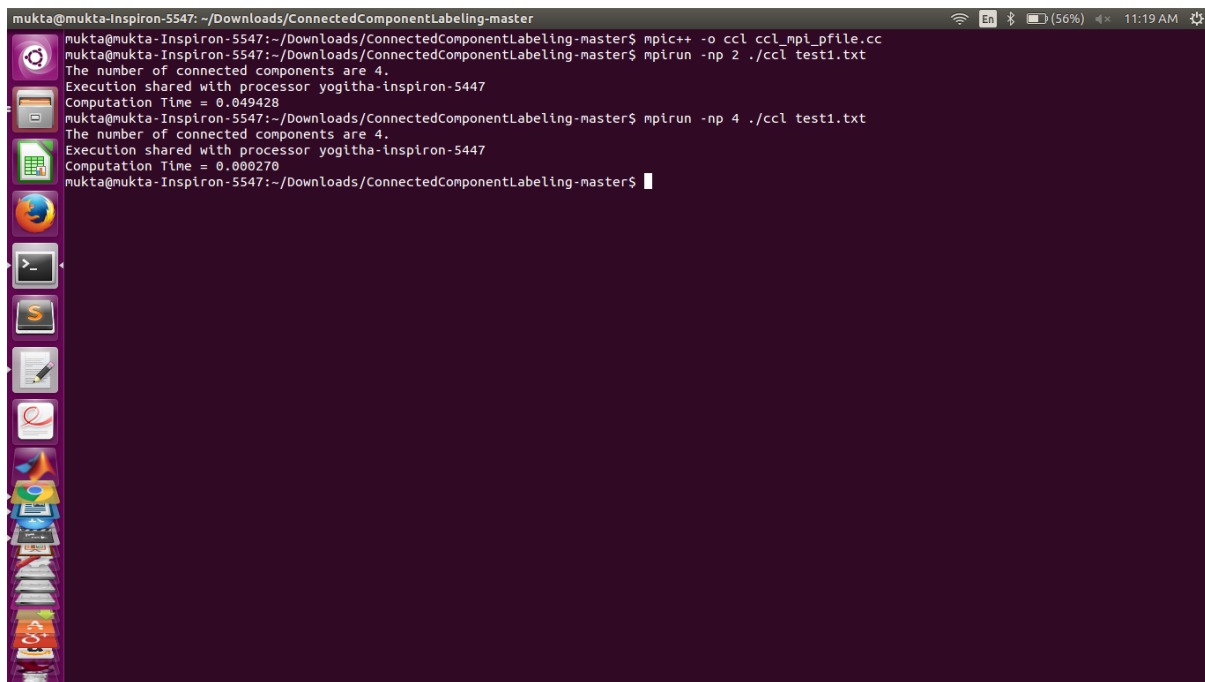
The screenshots of the program are shown in figure 6.1 and figure 6.2



A terminal window on a Linux system showing the execution of MPI programs. The user is mukta@ mukta-Inspiron-5547. The directory is ~/Downloads/ConnectedComponentLabeling-master. The commands and outputs are as follows:

```
mukta@mukta-Inspiron-5547:~/Downloads/ConnectedComponentLabeling-master$ mpic++ -o ccl ccl_mpi_pfile.cc
mukta@mukta-Inspiron-5547:~/Downloads/ConnectedComponentLabeling-master$ mpirun -np 1 ./ccl test2.txt
The number of connected components are 4.
Computation Time = 0.020072
mukta@mukta-Inspiron-5547:~/Downloads/ConnectedComponentLabeling-master$ mpirun -np 2 ./ccl test2.txt
The number of connected components are 4.
Computation Time = 0.000082
mukta@mukta-Inspiron-5547:~/Downloads/ConnectedComponentLabeling-master$ mpirun -np 4 ./ccl test2.txt
The number of connected components are 4.
Computation Time = 0.000194
mukta@mukta-Inspiron-5547:~/Downloads/ConnectedComponentLabeling-master$
```

Figure 6.1 Screen shot for readings on processors of one system



A terminal window on a Linux system showing the execution of MPI programs. The user is mukta@ mukta-Inspiron-5547. The directory is ~/Downloads/ConnectedComponentLabeling-master. The commands and outputs are as follows:

```
mukta@mukta-Inspiron-5547:~/Downloads/ConnectedComponentLabeling-master$ mpic++ -o ccl ccl_mpi_pfile.cc
mukta@mukta-Inspiron-5547:~/Downloads/ConnectedComponentLabeling-master$ mpirun -np 2 ./ccl test1.txt
The number of connected components are 4.
Execution shared with processor yogitha-Inspiron-5447
Computation Time = 0.049428
mukta@mukta-Inspiron-5547:~/Downloads/ConnectedComponentLabeling-master$ mpirun -np 4 ./ccl test1.txt
The number of connected components are 4.
Execution shared with processor yogitha-Inspiron-5447
Computation Time = 0.000270
mukta@mukta-Inspiron-5547:~/Downloads/ConnectedComponentLabeling-master$
```

Figure 6.2 Screen shot for readings on processors of two systems

CHAPTER 7

CONCLUSION AND FUTURE WORK

The task of connected components labelling is an important step in multiple image processing and computer vision projects. Reducing the speed of computation of this task while maintaining its accuracy is indeed a necessity. The results of the project clearly show that distributed computation of the two pass algorithm for labelling connected components offers better computation speed maintaining accuracy. Thus it can be implemented in practical projects.

Future work in this involves a research of other architectures apart from the pipeline architecture used in this algorithm that may reduce communication costs. Analysis of other methods used for connected components labelling and exploring into possibilities of distributing can also be done.

REFERENCES

- [1]. K. Suzuki, I. Horiba, and N. Sugie, "Linear-time connected-component labelling based on sequential local operations," *Comput. Vis. Image Underst.* 89(1), pp. 1–23, 2003.
- [2]. K. Wu, E. Otoo, and A. Shoshani, Optimizing Connected Component Labeling Algorithms, In *Proceedings of SPIE Medical Imaging Conference*, pp. 1965-1976, Apr. 2005
- [3]. L. He, Y. Chao, K. Suzuki, T. Nakamura, and H. Itoh: A label-equivalence-based one-scan labelling algorithm. *Journal of Information Processing Society of Japan* 50, pp. 1660-1667, 2009.
- [4]. Siddharth Gupta, Diana Palsetia, "A New Parallel Algorithm for 2 pass labelling of connected components"-
ArXiv 20 June 2016
- [5]. Mehdi Niknam and Sergio Camorlinga, "A Parallel Algorithm for Connected Component Labelling of Gray-scale Images on Homogeneous Multicore Architectures ", *High Performance Computing Symposium-2010*