SCALABILITY EVALUATION OF MATLAB ROUTINES FOR PARALLEL IMAGE PROCESSING ENVIRONMENT

JAMIL ABDULHAMID MOHAMMED SAIF[1]
AND PIOTR SUMIONKA[2]

[1]*Hodeidah University*
Hodeidah, Yemen
Currently: University of Bisha
Bisha, KSA

[2]*Gdansk University of Technology, Academic Computer Center CI TASK*
Narutowicza 11/12, 80-233 Gdansk, Poland

(received: 10 August 2017; revised: 11 September 2017;
accepted: 19 September 2017; published online: 6 October 2017)

Abstract: Image edge detection plays a crucial role in image analysis and computer vision, it is defined as the process of finding the boundaries between objects within the considered image. The recognized edges may further be used in object recognition or image matching. In this paper a Canny image edge detector is used which gives acceptable results that can be utilized in many disciplines, but this technique is time-consuming especially when a big collection of images is analyzed. For that reason, to enhance the performance of the algorithms, a parallel platform allowing speeding up the computation is used. The scalability of a multicore supercomputer node, which is exploited to run the same routines for a collection of color images (from 2100 to 42 000 images) is investigated.

Keywords: scalability, parallel image processing, MATLAB

DOI: https://doi.org/10.17466/tq2017/21.4/i

1. Introduction

Image processing and multimedia processing have become very popular scientific and technical disciplines [1, 2]. From the research point of view many old methods are improved, and many others are proposed. From the practical point of view, many platforms and libraries are offered for direct use. In the paper we limit our consideration to the MATLAB (MATrix LABoratory) computing environment. Presently it has over 2 million users from the engineering, science and economics areas, it offers many functions and subroutines (MATLAB libraries)

written in many programming languages (C, Java, Perl, Python), which can be called MATLAB. It can be connected to Maple or Mathematica. We can use MATLAB as a programming language and create our own programs based on many MATLAB library functions. Therefore, extra libraries called toolboxes are used which are oriented on solutions of domain-oriented problems. One of them is the Image Processing Toolbox, which contains software on image processing. In the paper we concentrate on this toolbox. To illustrate our considerations we consider one of the functions of image edge detection. Image edge detection plays a crucial role in image analysis and object recognition, it is defined as a process of finding the boundaries between the objects within the considered image, such techniques are named boundary based methods [3, 4]. In such methods abrupt changes are searched to produce a binary image which represents the edges and the background of a considered image. Many useful features can be extracted from edges, thus these features are exploited by upper-level computer vision algorithms [5–7]. Edge detection is regarded as a crucial role in several applications such as object detection, recognition for medical diagnosis, and many others. Image edge detection is an open and promising field of research, many researches deal with single based derivative edge detectors [3, 2, 8]. With the necessity of a huge amount of multimedia information, an approach is required to cope with it. A natural solution is to parallelize the computation aiming at enhancing the performance [9–11]. The purpose of our work is to evaluate the scalability of MATLAB functions for image edge detection using a parallel platform. The remaining content of the paper is organized as follows: in Section 2 the problem methodology is described in details, then the experiments and results are presented in Section 3, finally, conclusions and future works are proposed and recommended.

2. Scalability Problems

The main goal of our work is to investigate the scalability of a parallel platform for the MATLAB image processing toolbox. A specific routine was selected for our research work (*e.g.* Canny image edge detection) that is described in detail, to evaluate the performance and scalability of programs. Scalability is one of the most important features of computing systems. The idea of our work is considered for several points:

- The set of system resources, where adding new resources (system nodes), we obtain performance improvement;
- The size of input data, if the size increases the system is acceptably efficient;
- The number of users, if the number of users increases the system is also acceptably efficient;
- Low time – a complexity of algorithms take place, if the number of algorithm operations is increasing according to n, the amount of required resources goes less than n. In other words, the number of the required cores/processors is not increasing too radically.

In the paper the above aspects of scalability are taken into account. Let denote that $T(App,k)$ is the application (App) execution time, when it runs on k-cores. Then, the speed-up $S(App,k)$ is calculated in the following way:

$$S(App,k) = \frac{T(App,1)}{T(App,k)} \tag{1}$$

However, the application execution time depends also on the input data size, where d represents the number of images being under processing, then the formula should be changed as follows:

$$S(App,k,d) = \frac{T(App,1,d)}{T(App,k,d)} \tag{2}$$

To find the function $T(App,i,d)$, we should choose App and make some experiments on I cores, the schema of which is presented in Figure 1. In other words we should prepare a parallel version of the application, and establish a suitable set of data (pictures) consisting of d images. Then, we should compile the prepared program and run it on a different number of nodes $i \in \{1,...,k\}$ measuring the execution time. After this, we should change the size of the data and repeat calculations for a new stream of data.

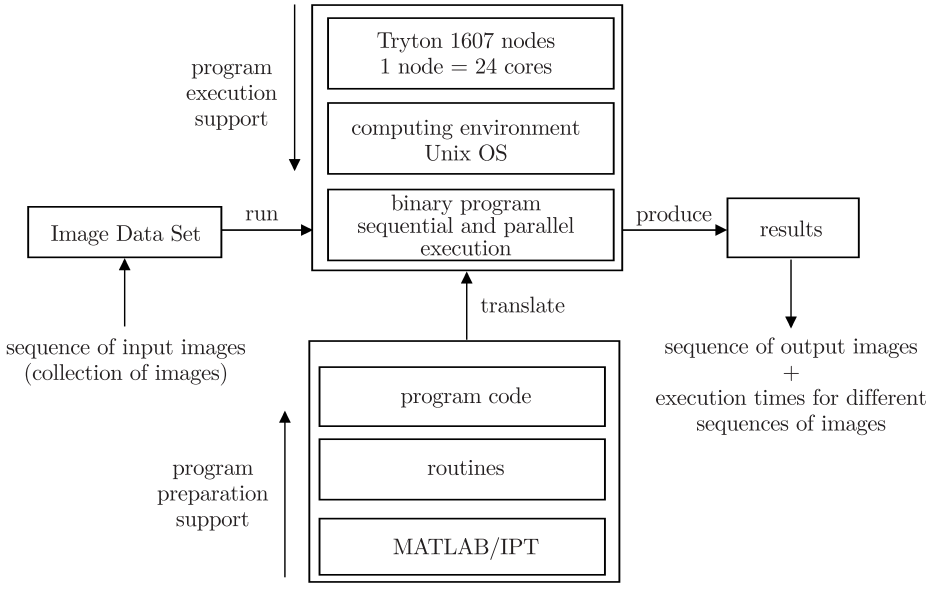


Figure 1. The testing Environment for MALAB/Image Processing Toolbox (IPT)

3. The representative applications

To choose the representative applications we should recognize the MATLAB Image Processing Toolbox (MIPT). It is an amazing toolbox that allows us to use a variety of image processing, image analysis, image segmentation, image registration and many other algorithms which are already implemented. The image

processing toolbox implements a huge amount of image processing algorithms that may consume too much time to be implemented especially when we have to run algorithms for a data set of images which can reach several hundred thousands of images varying in their size and resolution. The toolbox can deal with whole images as well as regions within the images that are more effective, particularly when we are interested in a specific area in the image (Region of Interest RIO). A Canny image edge detector routine [12] that is provided by the MIPT is used in our research for a set of images that give an edge output image as shown in Figure 2. A Canny edge detector is one of several of image edge detectors that have been used [8, 7, 13], but the Canny image edge detector is considered to be more efficient, it finds edges with more accuracy thanks to minimization of the signal to noise ratio. The Canny edge detector is characterized by the following features:

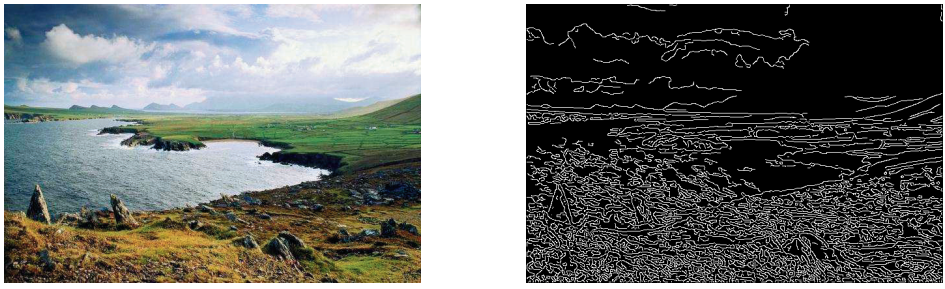
I. The signal to noise ratio of the gradient is maximized, leading to high quality of edge detection;
II. Good localization, localized edges should be with maximal accuracy to real ones;
III. Minimal responses, the detector should give a single edge and eliminate the false edge caused by noises.

The steps of the Canny algorithm are given as follows:

1. Image smoothing by convolution with a Gaussian filter, noises are removed as a result;
2. The magnitude and angle of gradient calculation for each pixel of the smoothed image in the horizontal and vertical directions;
3. Thinning the image by application of a non-maximum suppression of the image pixels, that suppress all pixels except those with local maxims, thin edges are preserved as a result;
4. Hysteresis, two dynamic upper and lower thresholds represented by th and tl, respectively are used, these are local thresholds which are dependable on the local content within the image. Pixels with gradient values greater than th regarded as the edge, and those that are smaller than tl are neglected, those between both thresholds are considered as edges, if they are connected to pixels with values higher than th, otherwise they are rejected.

In Figure 2 (b) the results of the Canny detector are presented for the given image in Figure 2 (a). It recognizes plenty of land/water frontiers. The experiments were made on the supercomputer Tryton in the TASK (Tri-City Academic Computer Network) Computing Centre at the Gdansk University of Technology, Gdansk, Poland, and are considered to be a real solution.

Moreover, the MATLAB toolbox is provided with a parallel computation Toolbox (MPCT), enabling it to be adaptable with tremendous data and heavy computation using a multicore supercomputer and a cluster network or other parallel platforms. Two approaches of parallel computation are available that allow



(a) (b)

Figure 2. Representation of: (a) image, (b) its corresponding edge image produced by the Canny detector

us to perform parallel image edge detection using the functions (`parfor` or `spmd`) which are provided by that toolbox. The extra software allows distribution of the workload among the workers automatically. The multicore nodes, or many nodes in the cluster, can be used to distribute tasks among them. The implemented codes for both approaches are show in Figures 3–4.

```
App(parfor)

% Number of workers
workers = 24

% Creating a special job on a pool of workers
parpool('local', workers)

% Read test data from directory
Files = dir('/users/kdm/jamils/MATLAB/img_42000/*.jpg');

I = cell(length(Files));     % Matrix for input data
m = cell(length(Files));     % Matrix for grey scale images
b = cell(length(Files));     % Matrix for edges (output data)

tic% Start of time measurement
% Start of parallel region.....................................
parfor i = 1 : length(Files)
  I{i} = imread(fullfile(Files(i).folder, Files(i).name));
  m{i} = rgb2gray(I{i});
  b{i} = edge(m{i}, 'Canny');
end
% End of parallel region.....................................
toc % end of time measurement

delete(gcp)% Delete space of parallel job
```

Figure 3. Illustration of `parfor` Parallel Canny routine

```
App(spmd)

% Number of workers
workers = 24

% Creating a special job on a pool of workers
parpool('local', workers)

% Read test data on directory
Files = dir('/users/kdm/jamils/MATLAB/img_42000/*.jpg');

workflow = round(length(Files)/workers);  % Number iterations on worker

I = cell(workers, workflow);  % Matrix for input data
m = cell(workers, workflow);  % Matrix for grey scale images
b = cell(workers, workflow);  % Matrix for edges (output data)

tic% Start of time measurement
% Start of parallel region.....................................
spmd
  maxworkers = numlabs;       % Total number of workers operating
                              % in parallel on current job
  numworker = labindex;       % Index of this worker
  for i = 1 : maxworkers      % Iterations of workers
    if numworker == i         % Select worker number
      for j = 1 : workflow    % Iterations inside worker
        I{i, j} = imread(fullfile(Files(i).folder, Files(i).name));
        m{i, j} = rgb2gray(I{i, j});
        b{i, j} = edge(m{i, j}, 'Canny');
      end
    end
  end
end
% End of parallel region.....................................
toc% end of time measurement

delete(gcp)% Delete space of parallel job
```

Figure 4. Illustration of SPMD Parallel Canny routine

4. Experiments and Results

In consequence we will test two different routines `App(parfor)` and `App(spmd)` and collect results for both of them, using one node of the Tryton Supercomputer. Each node of Tryton has two Intel Xeon E5v3, 2.3 GHz processors each with 12 cores, 128 GB RAM, and an InfiniBand FDR 56 Gb/s network connected in a fat tree structure. For our experiments we utilized and ran the Canny Image detector, which is a part of the MIPT, besides the MPCT is combined using the two available functions for parallelizing the computation. Image collections starting at 2100 images and ending with 42 000 images were entered

as an input, for each set of images a variable number of cores (workers) from one core increasing it by one each time up to the limit core number of 24, Canny algorithms were executed with varying numbers of workers ranging from one to a maximum number of 24. The execution time for each combination of the number of cores and the Image Data size allowing us to calculate the speed up defined as the ratio of execution time using one core divided by the execution time using parallel computation with a specific number of cores which is given by the formula (2).

Table 1 gives the execution times for workers (cores) varying from 1 up to 24 with different image collection numbers starting at 2100 and ending at 42 000 images and using two different parallel functions (`parfor` and `spmd`).

In Figure 5 the execution time of the systems is illustrated with increasing cores from 1 up to 24 for a different data size (the number of images in a collection) which started at 2100 images and ended up with the maximal data size as 42 000 images.

The speed-up for the tested routines as a function of the number of workers for different data sizes $d \in \{2100, 4200, 8400, 16\,800, 21\,000, 42\,000\}$ is presented in Figure 6.

5. Conclusion and future works

The obtained results show that the execution times for the examined two types of applications `App(parfor)`, and `App(spmd)` for a smaller data size (d) and using cores $i > 1$, are relatively comparable, but with increasing d the execution time is improved in favor of the latter application. It is also shown how the number of cores affects the execution time for specific d. However, the scalability of the latter application is more realistic than of the former. Moreover, the node architecture also impacts the scalability. In Figure 6 a general description of the node work is illustrated. After the program compilation, it is located in the RAM memory, where also a set of data is located. Then, a program is executed according to the assumed management strategy. Such strategy describes in what ways the processors can be activated, and in what way three level registers can be used for data processing. To recognize this, we have to have deep knowledge about the compilation procedures and the management strategy. Moreover, we should be able to monitor the behavior of processors, and their cores, nearly at the binary levels.

It is important, from the scalability point of view, that the impact of the processor architecture in comparison to the data set size is not dominant for one node. Scalability considerations for the computing architectures built up of many nodes should be analyzed. Then, architectures play much more significant roles.

In Figure 7. The relationship between the execution time and the data size (number of images) is illustrated using different numbers of cores $i \in \{1, 6, 12, 18, 24\}$.

Table 1. Execution times for tested routines

number of workers	data size (number of images)											
	2100		4200		8400		16 800		21 000		42 000	
	parfor	Spmd	parfor	spmd	parfor	spmd	parfor	spmd	parfor	spmd	parfor	spmd
1	85.607	40.521	132.121	78.006	259.208	152.662	622.832	300.662	624.311	377.188	1404.079	749.282
3	24.157	23.034	46.153	41.641	89.801	53.475	174.809	134.199	270.879	200.449	433.904	379.564
6	12.700	14.515	24.473	22.159	47.375	50.235	91.076	97.017	111.586	122.251	219.443	192.613
9	9.168	10.242	17.223	16.331	34.674	35.353	62.616	67.000	79.832	85.412	154.174	132.277
12	7.374	8.173	13.348	13.250	27.640	27.064	53.464	52.047	63.083	66.214	119.993	101.435
15	6.552	6.846	11.496	11.015	22.253	22.066	49.351	43.702	57.006	54.477	105.486	82.523
18	6.207	5.985	10.638	9.554	19.973	18.754	42.850	37.977	52.308	48.204	99.004	71.274
21	5.877	5.491	10.168	8.457	19.412	16.427	40.462	33.322	49.765	42.000	95.972	63.516
24	5.828	5.049	10.153	9.264	19.303	16.276	40.641	32.799	50.090	38.231	94.326	57.884

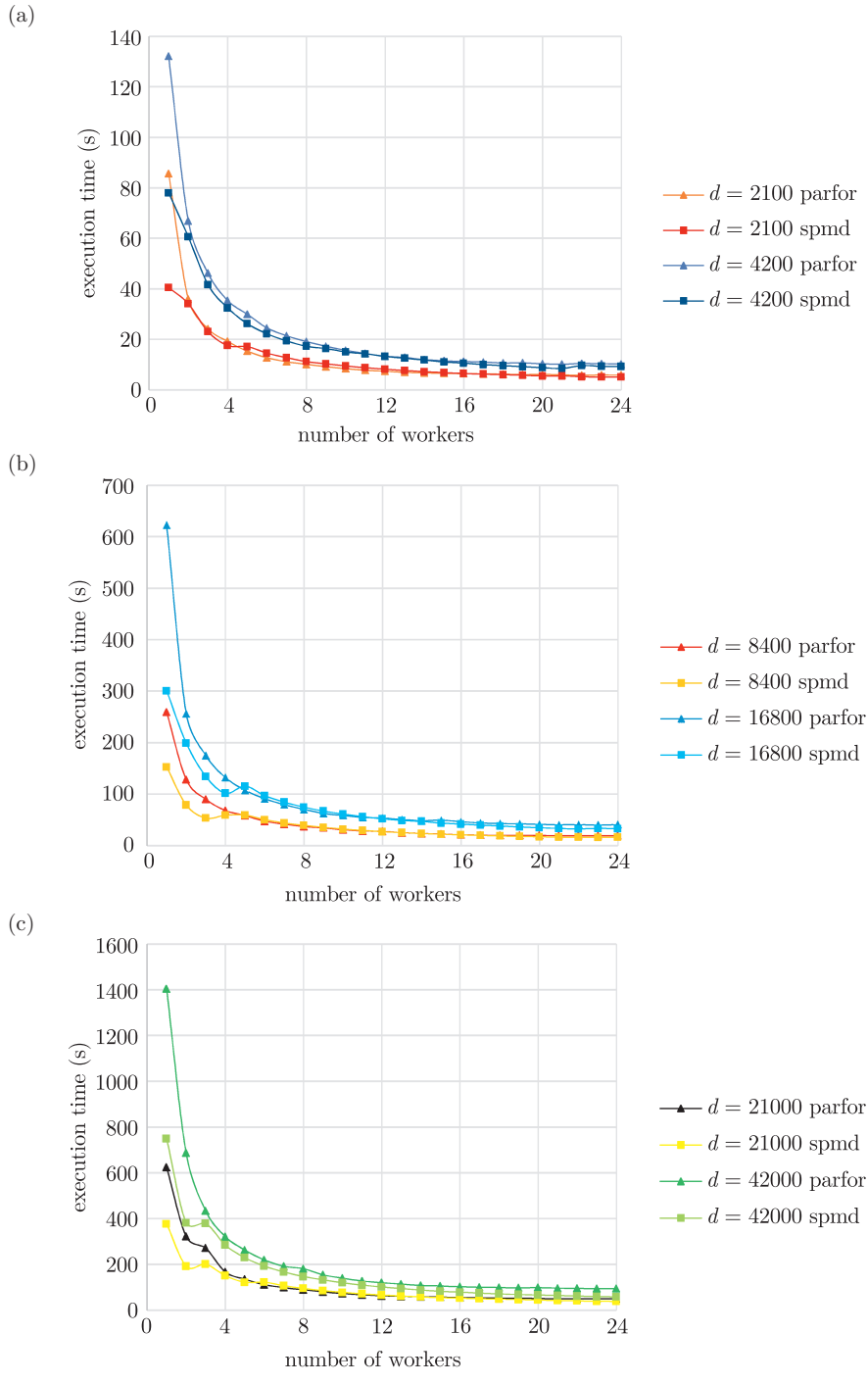


Figure 5. Curves of execution times for: (a) $d = 2100$, $d = 4200$; (b) $d = 8400$, $d = 16\,800$; (c) $d = 21\,000$, $d = 42\,000$

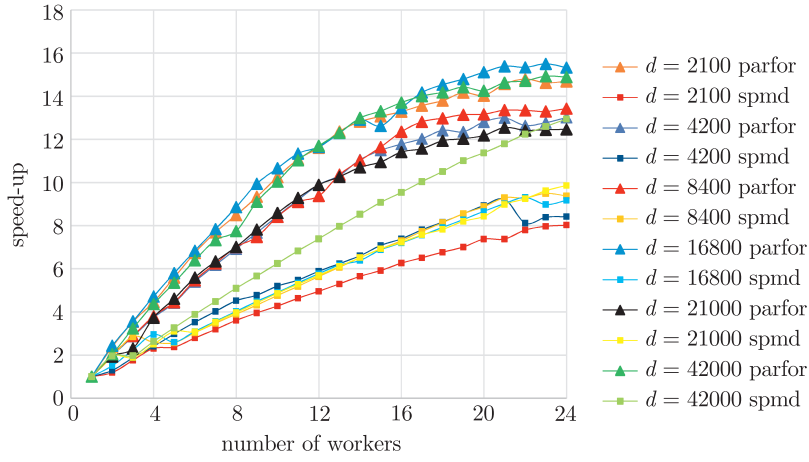


Figure 6. The speed-up for different data sizes $d \in \{2100, 4200, 8400, 16\,800, 21\,000, 42\,000\}$

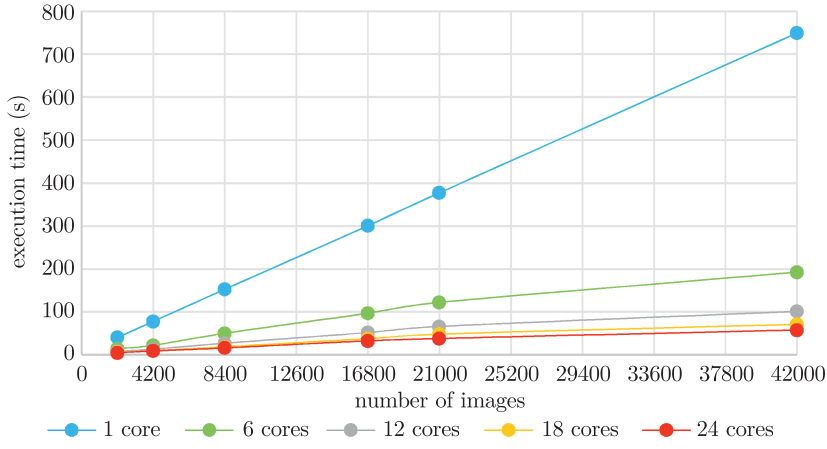


Figure 7. Execution time as a function of data size (d) and cores number (i)

In our paper we run the Canny image edge detector in a parallel way using the MATLAB parallel toolbox, as is shown in the figures of the results. Parallelizing the algorithm allowed us to speed up the computation which is evidently considered an exceptionally promising solution, especially when it is required by image analysis for a collection of images. For future work it is recommended to utilize parallelization with a variety of segmentation algorithms or deep learning image retrieval and object recognition which usually deal with Big Data (image data store) as well as to focus on parallelization of the applications for a different number of nodes instead.

References

[1] Acharya T and Ray A K 2005 *Image Processing Principles and Applications*, John Wiley & Sons, Inc.
[2] Saini R, Dutta M and Kumar R 2012 *J. Information and Operations Management*
[3] Madhuri J A 2006 *Digital Image processing*, Prentice Hall

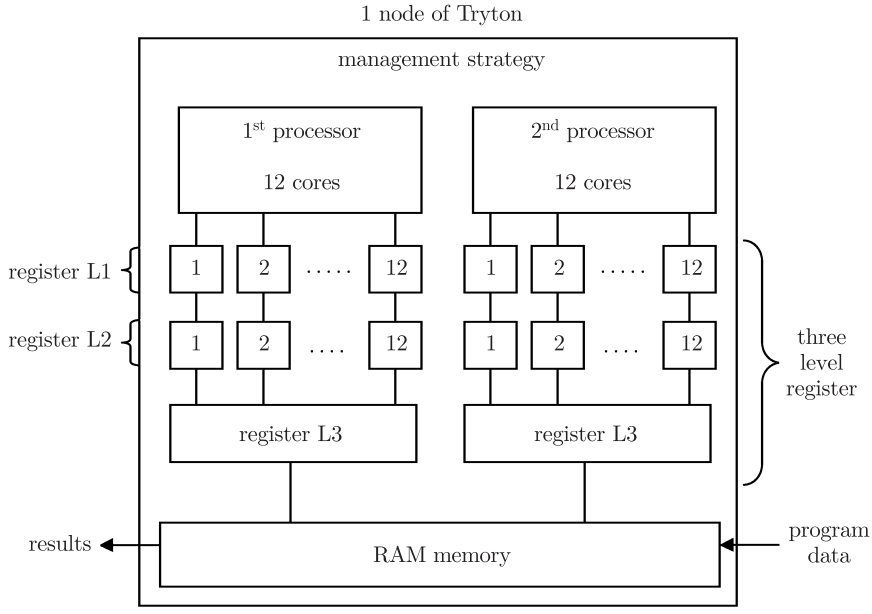


Figure 8. Simplified node architecture of Tryton

[4] Saif J A M, Hammad M H, and Alqubati I A 2015 *ICSST 2015*
[5] Szeliski R 2011 *Computer Vision – Algorithms and Application*, Springer
[6] Thakare P 2011 *IJCSE*
[7] Umbaugh S E 1998 *Computer Vision and Image Processing: A Practical Approach Using CVIP tools*, Prentice Hall
[8] Sonka M, Hlavac V and Boyle R 2008 *Image Processing, Analysis and Machine Vision*, Thomson
[9] Krawczyk H, Proficz J and Ziolkowski T 2012 *Task Quarterly* **16** (1) 145
[10] Krawczyk H and Proficz J 2012 [Online] available at: `https://www.intechopen.com/books/interactive-multimedia/real-time-multimedia-stream-data-processing-in-a-supercomputer-environment`
[11] Proficz J and Krawczyk H *Task Allocation and Scalability Evaluation for Real-Time Multimedia Processing in a Cluster Environment*
[12] [Online] available at: `https://www.mathworks.com/help/images/ref/edge.html`
[13] [Online] available at: `http://suraj.lums.edu.pk/~cs436a02/CannyImplementation.htm`