

Dynamic Load Balancing in Image Retargeting using Pipeline Architecture

Ganesh V. Patil¹, Santosh L. Deshpande²

¹ VTU Belgaum, India, pganeshv@gmail.com¹

² VTU Belgaum, India, sld@vtu.ac.in

Article Info

Article history:

Received Jun 12th, 201x

Revised Aug 20th, 201x

Accepted Aug 26th, 201x

Keyword:

Image Retargeting,
Image Resizing,
Image Quantization,
Image Compression,
Pipeline Architecture.

ABSTRACT

In today's smart world demand of efficient multimedia based communication has increased at a rapid rate. Diversity on display sizes of gadgets used for multimedia communication confines the quality of images. Image retargeting is used as the focal solution to this problem which results in images with appropriate sizes. Enormously mounting demand of image retargeting expedites the rate of increment in computational load. This research paper expatiates and experiments a dynamic load balancing based three phase image retargeting methodology using pipeline architecture. In the first phase of image retargeting resize operation is performed on input image which results in multiple sized image copies of the same image. In the second phase resized images undergo quantization operation. In the final phase lossless compression is performed to have an expedient image. In the proposed exhibit think, we have done statistical analysis of results obtained, to confirm an impartial dynamic load balancing with a better degree of underlying resource utilization. We extend the approach to achieve significant storage optimization using three phase image retargeting.

Corresponding Author:

Santosh L. Deshpande,
VTU Belgaum, India, sld@vtu.ac.in

1. Introduction

In today's smart generation rate of communication using internet based astute handheld gadgets increasing explosively. Survey of year 2017 puts a light on a fact that total 1.2 Trillion pictures are caught using the smart cell phones [30]. As time goes, it turns into a typical prerequisite by every client to have an advanced cell with fundamental computational capabilities. Presently unstable growth in graphical correspondence, requests an ever increasing number of computational assets with more web transmission capability. The worldwide trend of manufacturing astute mobile devices results in each device displays with varying resolutions. This fact creates challenge to a web administrator, to retain the same quality on different sized display gadgets. This powerfully changing user interest overburdens the endeavors of web executive. It makes a thoughtful undertaking for a web administrator to give a proper estimated picture of an individual device with great picture determination. Process of operational sequence which results a suitable image for a respective display sized device by preprocessing is called image Retargeting. A lot of processing power as well as battery backup is utilized to scale a small image into large one as per the display requirement.

To have a suitable optimized image for a respective device we have formulated a three phase Image Retargeting mechanism. Three phases are as follows:

1. Image Resizing
2. Image Quantization
3. Image Compression.

Distributed system gives the coordinated multimodal operations to give better efficiency and throughput at large scale. Its potential of Distributed systems to offer a platform for sharing and aggregation of computational utilities [23]. Distributed system consists of a large number of worker nodes connected to the interconnection network to have a collaborative execution environment which seems to be a single system. A distributed computing system is considered as an aggregated system which gives the collective effect of underlying computing and communication resources. A major challenge in a distributed system is to achieve a good extent of resource utilization. Resource utilization is a combined effect of resource scheduling and load balancing. In load balancing good response time is achieved by suitable provisioning of computational resources. Load balancing is classified in two classes i.e. 1. Static Load Balancing 2. Dynamic Load Balancing. Static load balancing requires the prior information of underlying resources and based on that resource scheduling is carried out. In contrast to this Dynamic Load Balancing is based on the dynamic and instantaneous i.e. Run time information about resources. Dynamic load balancing is used to have a good response time in highly computationally intensive setups which does run time task scheduling.

In this paper, we are focusing on dynamic load balancing based distributed computing environment for image retargeting using pipeline architecture. Dynamic load balancing is achieved using two Image retargeting pipelines. As Resizing, Quantization and compression operation is to be applied sequentially.

In proposed experimental study, we have used pipeline architecture with dynamic load balancing mechanism. The basic objectives of our experimental work are as follows,

1. To achieve Dynamic load balancing in Image Retargeting using Pipelined architecture.
2. To achieve a better degree of resource utilization.
3. To reduce the communication time in comparison with computational time.
4. To achieve storage optimization along with image retargeting.

The paper is organized in six sections. In section I, Introduction of problem is given and it contains clear objectives of proposed work. In section II, literature review of Dynamic load balancing and existing Image retargeting techniques is given. In section III, methodology and architecture of proposed research work is discussed. Section IV, contains results obtained in 3 phase image retargeting process and its statistical inference is discussed and attainment of stated objectives is verified. In section V, we conclude the work with remarkable achievements.

2. Related work

Enormous work is done by the researchers in field of dynamic load balancing in different areas of computation. Load balancing is becoming unavoidable part of modern computing environments. It is required to have a system with better resource utilization and better throughput. Static load balancing is based on the average behavior of the system whereas dynamic load balancing considers runtime state information of the system [14]. Due to overwhelming demand of multimedia communication, modern era of image processing includes heavy computational loads. In proposed work we have used adaptive resource based dynamic load balancing. The load in image retargeting changes rapidly with respect to time. To inculcate computational efficiency in task of image retargeting and to achieve impartial resource utilization, dynamic load balancing is

used. Mainly load balancing mechanism concentrates on task to resource mapping and success of this system depends on proper scheduling mechanism. The main purpose of load balancing is to make suitable and efficient provisioning of available resources in the system to achieve a good response time [30]. According to Grosu and Chronopoulos [21] load balancing strategies can be classified into three approaches: 1. Global approach 2. Cooperative approach 3. Non-cooperative approach. In global approach, global decision maker is responsible for taking all the possible decisions related to resource allocation and job mapping over a distributed set of resources. In cooperative approach, multiple decision makers results in load balancing with the help of message passing based communication mechanism. In non-cooperative approach, each node acts as an independent decision maker and works to minimize its own response time.

In dynamic load balancing [1,4,5,9,11,12,24,26,27,28,29], run time decision making is done which results in better resource provisioning whereas in static compile time decision making is done.

According to Watts and Taylor [26] dynamic load balancing can be solved practically using load evaluation, profitability determination, work transfer vector calculation, task selection and task migration. In general, a dynamic load-balancing policy consists of three components namely, information rule, transfer rule and location rule [28]. Dynamic state information is collected using information rule, decision making of load transfer is done using transfer rule and location rule gives best possible location for execution of job.

Classification of dynamic load balancing strategies was given by Willebeek-LeMair and Reeves [27] as,

1. Sender initiated diffusion (SID)
2. Receiver initiated diffusion (RID)
3. Hierarchical balancing method (HBM)
4. Gradient model (GM)
5. Dimension exchange method (DEM).

SID approach finds a lightly loaded near neighbor and transfers the excessive load to them. In RID approach, lightly loaded processors demands and pulls load from heavily loaded processors. HBM uses progressive decision based load balancing applied to a subset of nodes hierarchy. In GM [28] global decision making is done. Global decision making process takes aggregated local information as an input for decision making algorithm. DEM is used as dimension wise synchronous and iterative approach of load balancing.

Chow and Kohler [29] have proposed a queuing model based dynamic load balancing in simple heterogeneous multiple processor systems. Further the dynamic load balancing has been classified in deterministic and nondeterministic approach. Non-deterministic approach uses state independent branching probabilities whereas deterministic approach uses criteria functions to enhance the performance of computing.

In centralized GA-based mechanism Zomaya and Teh [24] have proposed an approach of dynamic load balancing using genetic algorithm. Threshold based sliding window technique is used for generating a job schedule. The research work [28] gives a framework of adaptive dynamic load balancing strategy for 3D rendering task using Blender software based on dynamic CPU and RAM utilization.

A new dynamic task scheduling algorithm [14] proposed for heterogeneous systems i.e. Clustering Based HEFT with Duplication (CBHD). The CBHD algorithm combines features of both Heterogeneous Earliest Finish Time (HEFT) and the Triplet Clustering algorithms. HEFT algorithm does ranking of nodes based on the computational capabilities and Triplet clustering algorithm is used for clustering of nodes according to their computational configurations.

Experimental investigation [34] gives experimental test bed on unique load adjusting methodology (DLBS) calculation, utilized for hypercube organization in multiprocessor framework [21]. Weighted Round

Robin Load Balancing [3] mechanism works well for cloud systems. Distributed architecture based SALB, a dynamic and adaptive load balancing algorithm [26] is a threshold based load balancing approach which tries to minimize the load caused by message exchanges and presents a online load prediction model. Dynamic load balancing in grid system [23] classifies the existing load balancing algorithms based on task migration requirements and grid resource topology used. This work considers a grid resource topology as flat and hierarchical.

An adaptive dynamic load balancing model [12] is a agent based distributed simulations which gives a distributed approximate optimized scheduling algorithm with partial information (DAOSAPI). This algorithm is an integration of the distributed mode, approximate optimization and agent set scheduling approach.

We are applying a concept of dynamic load balancing to a Image Retargeting task so it is obligatory task to review existing image retargeting algorithms and their features. Research domain of image retargeting expands its scope as user requirements from different domains exceeds in a wide range. Today in every sector multimedia based formal and informal way of communication is increasing explosively. J. Kim et. al. in [17] proposed a image and video retargeting based on adaptive scaling function. In this work first importance map is constructed using gradient, saliency and motion difference. In next step adaptive scaling function is calculated which gives a scaling factor of each column in source image. Desired image is then constructed using those scaling factors. The same algorithm is used for video sequence. Bin Zhou et. Al. in [27] proposed seam carving based Image Retargeting method. This work evaluates the image compressibility using wall seam model and then assigns a respective number of seams in each direction. This work ensures the content preserving Image Retargeting as compared with existing Image Retargeting methods.

The experimental work [16] proposes an adaptive image and video retargeting algorithms based on Fourier analysis. This work utilizes Gradient information of image to divide the image in similar complexity strips. Fourier transformation is used to formulate the distortion caused by image scaling. This work ensures that aggregation of sizes, all strips should equal to size of target output. Lagrangian multiplier technique is used to solve the constrained optimization problem. S. Wang et.al. [18] proposed warping based image retargeting technique. This research work is based on an adaptive image resizing algorithm. The proposed experimental work consists of a series of operations i.e. Bilinear interpolation, line detection, joint-bilateral upsampling. It does computations on the low-resolution layer which claims to be an efficient warping-based retargeting technique. In research work proposed by J. Sun et.al. [15] concentration is given on thumbnail retargeting method. This work considers most prominent issues regarding thumbnail retargeting i.e. thumbnail scales, object completeness and local structure smoothness. To solve these issues Jin Sun et.al. has proposed Scale and Object Aware Retargeting (SOAR) algorithm. The important components of SOAR are a) Saliency map b) Objectness c) Cyclic seam carving algorithm d) Thinplate-spline (TPS) retarget warping algorithm. Retargeting of a pair of stereo images is given in [24] which consider 3D structure of scene. This research work experiments a extended version of the seam carving algorithm which works on pair of images. The prominent feature of this work is this research considers visibility relations between pixels in the image pair.

In [29] improved seam carving based image retargeting is given. Seam carving is a content based image retargeting algorithm which removes the pixels with less energy value. It is very difficult to maintain the accuracy of the energy function. This research work proposes a combination of L-1 norm of gradient with 3D saliency to obtain energy map

In [19] Discrete Cosine Transform is used to evaluate both scaling and shape distortion. The proposed image resizing algorithm avoids loss of semantic information of images and it claims to be a computationally efficient as compared with traditional state of art resizing methods. In research work [26] authors investigated impact of perceptual relevance information on content aware image retargeting. This work integrates fixation density maps and region-of-interest maps into a contemporary image retargeting algorithm. This experiment

puts a light on impact of perceptual relevance information, image content, and retargeting ratio on the overall performance of Image Retargeting process. A saliency detection [10] based adaptive image retargeting is used in regions of interest extraction with image resizing. In [10] proposed saliency detection model crossed the limiting boundary of image retargeting by achieving the results in compressed domain of image retargeting.

Many researchers have experimented image processing using pipeline methodology. In [13] authors have implemented Line Buffer Based Image Reconstruction Pipeline. It's a software based pipeline framework for image processing. Image reconstruction pipeline consists of combination of color filter Array interpolation, noise filtering and several color correction operations. The mobile captured images are given as a raw input to these pipelines and image reconstruction is done to enhance the quality aspects [35] of the image. The LONI Pipeline Processing Environment for NeuroImage processing is proposed in [22]. It consists of pipeline of individual executable programs. In this work NeuroImage processing task is divided into number of subtasks. Each stage in pipeline performs a subtask and intermediate results are traversed between individual modules in pipeline. Functional pipeline used in [25] works for stream of data sets which is given as a input to the functional pipeline and throughput is maximized within the given latency constraints. It considers only two performance parameters of the functional pipeline i.e. throughput and latency.

3. Methodology of the work

An image is a two-dimensional array of individual colored pixels [31]. Each pixel is represented by a RGB vector in different color spaces which ranges from monochrome where color ranges from black to white to full RGB spectrum of colors. The representation format of each image is based on the visual effect requirement by each image. Today as the innovation in smart display technology goes towards the pick point, image resolution and clarity requirements also goes on increasing tremendously. Color complexity and storage requirements of images increasing rapidly as user demand for high resolution images increasing enormously. Here with the proposed solution we are trying to achieve best optimized three phase image retargeting as shown in Fig. 1. Resizing gives better resized image with good aspect ratios as per the user demands. Quantization and compression gives color palletization and lossless compression which reduces the storage requirements of images in acute handheld gadgets.

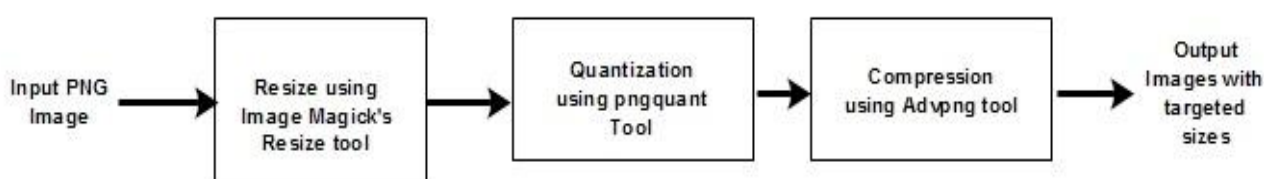


Fig. 1. Three phase image retargeting process

3.2 Image Resizing

Various software image resizing tools are available on web. Some of them are open source and others require licensing. In our experimental work we are using Image Magick's resizing tool [32]. Convert command is used to resize the image in required dimensions. By using this command we can enlarge the image as well as can make it smaller. In our experimentation along with convert command we are using Resize option for resizing. We have fixed the dimensions of target image by using absolute size option in Image Magick.

e.g. `convert -resize 320 X 213 input.png output.png.`

Filtering is the process of determining which pixels make it into the new image and what color they are. Filtering of resized image is required to decide what the new image will look like [32]. In our experimentation we have used default Lanczos filter of Image Magick.

e.g. `convert -resize 320 X 213 -filter lanczos input.png output.png`.

3.3 Image Quantization

In a regular 24-bit color image there are 16.7 million color possibilities for every pixel. But a human eye can distinguish only a small part of it because most of these color possibilities goes unutilized. We can utilize this aspect by grouping similar colors in a given image by creating a color palette, and then using a number to represent a color from this color palette. Such palletization of image can vastly improve the amount of storage required for the image. Lesser the color in the palette, lesser is the storage needed for each pixel. If the palette size is only 256, then only one byte is sufficient to store the entire pixel instead of three bytes [33]. Color palletization is a process of grouping similar colors in same palette. The size of resulting image depends on palette size. If the palette size is small, then the resulting image will also be small so the main aim of color quantization is to find the best color palette with the least differences between the original image and the quantized one. We are achieving color quantization of PNG images by using command-line utility pngquant [36]. Pngquant is a advanced tool which uses modified version of Median Cut quantization algorithm [20]. In modified version of median cut algorithm splitting boxes are selected to minimize variance from their mean value. An improved median-cut algorithm which, improves pre-quantization precision, calculates the cutting position based on variance and searches reversely the color map. It significantly promotes both the speed and quality of the color quantization [20].

3.1 Image Compression

For image compression we are using Advpng [35] software tool which does lossless compression.

Table 1 shows the list of software tools used in pipelined 3 phase image retargeting process.

Table 1 Software tools used in 3phase image retargeting

Sr. No.	Name of Software	Use of Software
1	Prometheus 1.5.2	Used as a State Information collector
2	Grafana 4.2.0 [35]	Works to visualize computational loads on the slave nodes
3	ImageMagick [32]	Used for image resizing
4	Pngquant [36]	Used for quantization of PNG images
5	Advpng 1.2.3 [34]	Used for lossless compression of PNG images.

4. Architecture of work

The architecture of pipeline image retargeting is as shown in Fig. 2. The proposed architecture is push-pull type architecture. In this architecture first information is pulled from slave nodes and based on that work is pushed towards the slave nodes for execution. The basic methodology selected for the execution of experimental work is Master-Slave architecture. The architecture consists of one master node and two slave pipelines. Each pipeline consists of three nodes which are performing the task of Resizing, Quantization and Compression respectively. Each input PNG image undergoes three phase image retargeting operation. The order of image retargeting is fixed. In first phase image undergoes Resize operation, in second phase the output of resize phase is given as a input to Quantize phase and in last phase the output of quantization is given as a input to compression phase.

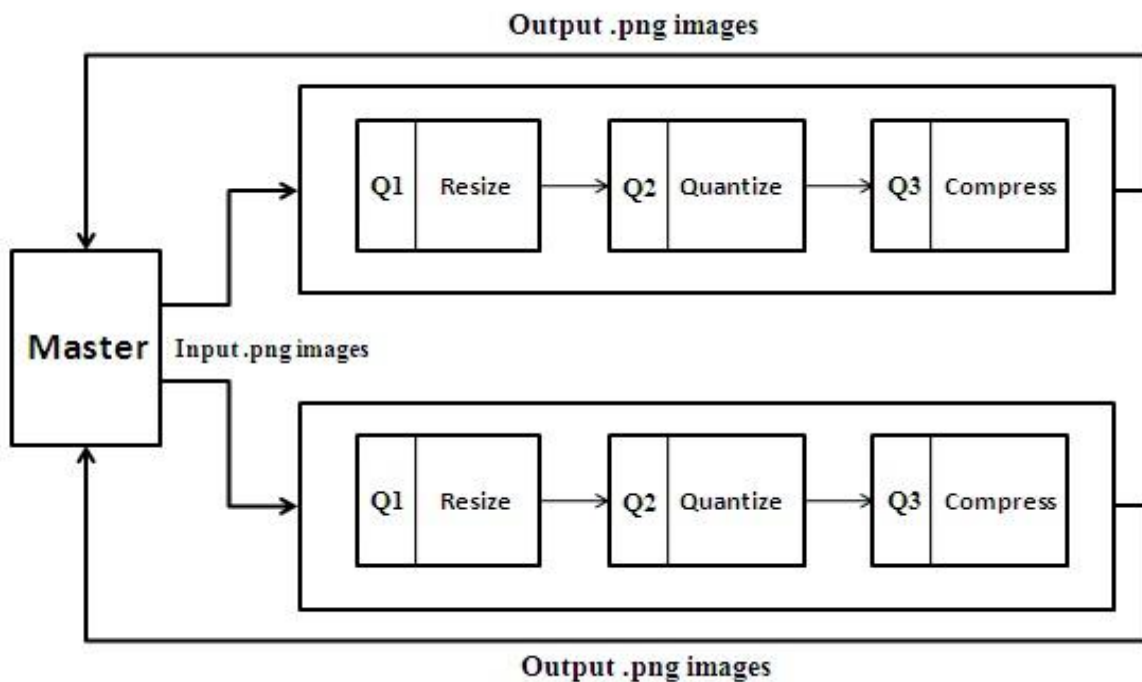


Fig. 2. Architecture of Proposed 3phase Image Retargeting

The compression phase returns final output to the master node. The overall working of image retargeting is carried out using waterfall model of software development paradigm. Q1, Q2 and Q3 represent queues in each node of Pipeline1 and Pipeline2. Before execution each job checks the availability of resource in each node and if node is free then job is considered for execution and if not then job has to wait in a node's waiting queue.

4.1 Master

1. Master acts as central decision maker. The base of decision making is the dynamic load information collected from all slave nodes of architecture. Master-slave communication is carried out using HTTP protocol. Master is started first.
2. We are using single binary file for both master and worker. Role of binary is decided by the run time parameters passed to the program.
3. Master first starts HTTP server. The registration module in master node starts registration of all slave nodes. This process collects the basic configuration of each slave node i.e. Maximum CPU and RAM available at each node.
4. Periodically master collects the run time state information from all slave nodes. Scheduler utilizes this dynamic state information for dynamic load balancing.

4.2 Scheduling Algorithm

Table 2 exhibit detail scheduling algorithm used for three phase image retargeting process.

Table 2 Scheduling Algorithm used for 3 phase Image Retargeting

Algorithm 1 Scheduling algorithm

Input: Dynamic state information of Pipeline 1 and 2.

Output: Dynamically balanced load between Pipeline 1 and 2.

Algorithm

- 1: Start Master
 - 2: Start State Info Collector for periodic collection of state information from all slave nodes.
 - 3: For all $i=1$ to N
 - 4: Register all nodes with Maximum CPU and RAM available at that node.
 - 5: ConFig. Pipeline1 and pipeline 2 by assigning operational role to each slave node.
 - 6: After every time period t , For all $i=1$ to N
 - 7: Receive(CPU and RAM utilization)
 - 8: Calculate the average CPU and RAM utilization for each Pipeline P1 and P2.
 - 9: Assign the incoming load to the pipeline which has Minimum CPU and RAM utilization.
 - 10: Receive the results from selected pipeline.
 - 11: END
-

System used for three phase image retargeting process is a homogeneous system. Table 3 shows the configuration details of experimental set up used for experimentation of three phase image retargeting. In experimental exhibition we have used one master node and six slave nodes. Each pipeline consists of three slave nodes.

Table 3 Configuration details of set up used for experimentation

Sr. No.	Node Name	Configuration Details
1	Master	Dell Optiplex 3050, CPU-Intel Core (TM) i3-7100, 3.90 GHz, RAM-4GB, OS-Windows 7 Professional (64 Bit)
2	Slave	Dell Optiplex 3050, CPU-Intel Core (TM) i3-7100, 3.90 GHz, RAM-4GB, OS-Windows 7 Professional (64 Bit)
3	Network	Giga Bit Network

The assumptions made by our system are as follows:

1. Dynamic load balancing used in our system is non-preemptive.
2. The key parameters used for load balancing are each node's run time values of CPU and RAM.

3. Computational weightage given to both CPU and RAM is same.
4. Experimental set up contains one master and six slave nodes.

We have implemented the dynamic load balancing using Parallel and distributed pipelined approach.

5. Results and discussion

In this section we present results with statistical analysis to verify the attainment of objectives of proposed work. We have carried out the dynamic load balancing task for 59 different sample images with variable dimensions. The Table 4 shows time required for image retargeting including resizing time, quantization time, compression time, Queuing time and communicational time. Equation 1 represents Total time T as a function of resizing time Rt , quantization time Qt , compression time Ct , Queuing time Wqt .

$$T = f(Rt, Qt, Ct, Wqt) \quad (1)$$

As shown in Fig. 2 every slave node in a pipeline architecture has a queue associated with it. If the node is busy in doing existing computational work, newly arrived job has to wait in a queue until the node becomes free. Proposed image retargeting process contains an ordered sequence of Resizing, Quantization and Compression operation. We have selected pipeline architecture for experimentation because of ordered execution of operations. Each intermediate node has to wait for a result of previous node. This fact becomes a bottlenecking problem in pipeline image retargeting. The total actual computational time required by any node in three phased image retargeting is much less than the waiting time in a queue. If we compare the actual computational time required with waiting time in a queue, queuing time exceeds over the actual computational time. This factual analysis reflects in a Table 3 and Fig. 3. Fig. 3 represents the graphical analysis of Computational time required versus queuing time.

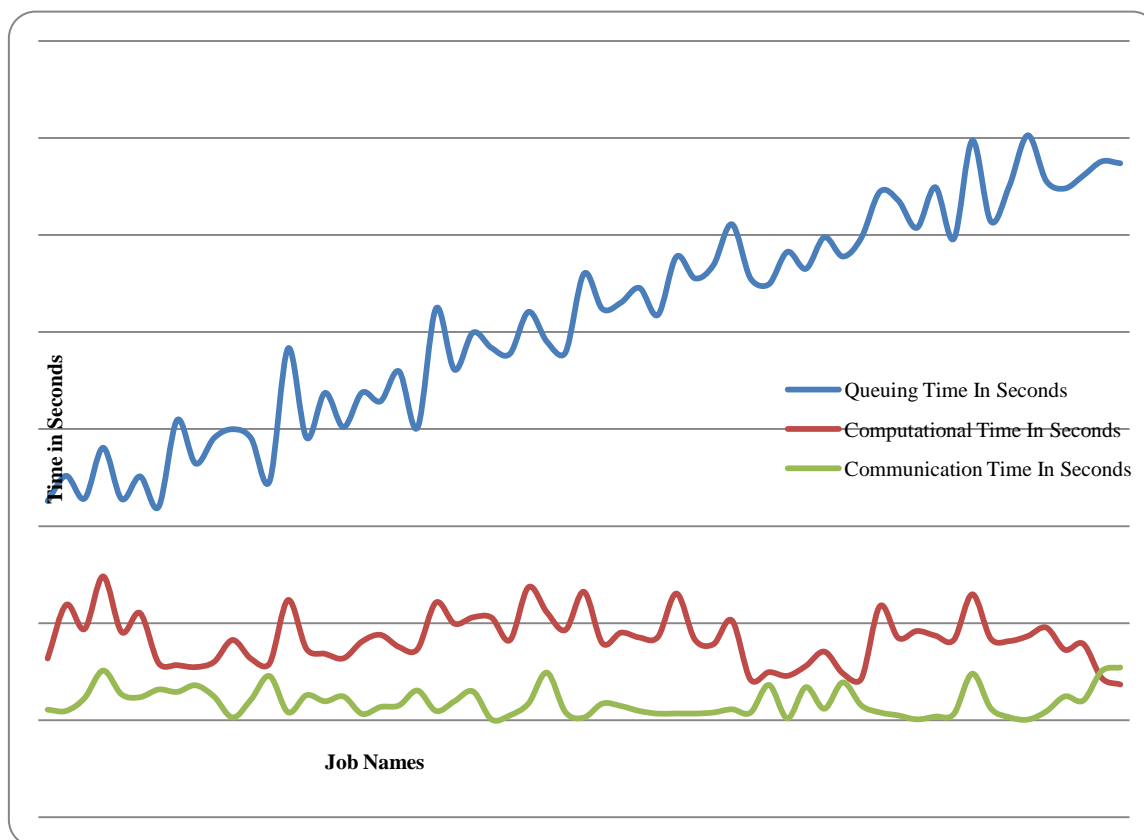


Fig. 3 Comparison of Queuing time, Computational time and Communication time

We assume that total time required for image retargeting is a linear function of Resize time, Quantize time, Compress time and Queuing time. With this assumption we have done analysis by using multivariate correlation and regression technique. This analysis technique determines the interrelation between principal attributes under consideration. We have done analysis of Pipeline 1 and Pipeline 2 separately, and try to differentiate the processing effects in two different pipelines. In every analysis height and width of incoming image remains as common attribute. We are analyzing the effect of dimension changes on each phase of image retargeting.

5.1 Analysis of Resize Time

Resize is the primary operation which is applied to the incoming PNG images. In our three phase image retargeting, resizing of incoming images is done by the head node of every pipeline as shown in Fig. 2.

Here in this subpart we have analyzed the effect of following factors on total Resize time of Pipeline i.e. $T(\text{Resize})$:

1. Height of Image (H)
2. Width of Image (W)
3. Time of communication from Master node to Q1($T(M, Q1)$)
4. Waiting time of job in Q1($W(Q1)$)
5. Time required for job to travel from Node1 to Queue2 ($T(N1, Q2)$)
6. Waiting time of job in Q2($W(Q2)$)
7. Time required for job to travel from Node2 to Queue3 ($T(N2, Q3)$)
8. Waiting time of job in Q3($W(Q3)$)
9. Time required for job to travel from Node3 to Master ($T(N3, M)$)

We apply the regression analysis to the results in Table 3, we get equation 2 and equation 3. These equations i.e. 2 and 3 gives weighted relation of image retargeting factors to calculate Resize time in Pipeline1 and Pipeline2. Here $T(\text{Resize}_{p1})$ is the resize time of Pipeline1 and $T(\text{Resize}_{p2})$ is the resize time of pipeline2.

$$T(\text{Resize}_{p1}) = -30377.3 - 0.614(H) + 0.641(W) + 4.096(T(M, Q1)) + 0.925(T(N1, Q2)) - 0.028(W(Q1)) \quad (2)$$

$$T(\text{Resize}_{p2}) = 274039.4 - 0.017(H) + 0.53(W) - 20.89(T(M, Q1)) + 1.04(T(N1, Q2)) - 0.007(W(Q1)) \quad (3)$$

5.2 Analysis of Quantize Time

Quantize is the second phase of image retargeting. Our pipeline implementation follows an ordered execution so performance of every phase depends on the performance of previous phase. We apply the regression analysis to the results in Table 3, we get equation 4 and equation 5 for determining Quantize time. Here $T(\text{Quantize}_{p1})$ is the Quantize time of Pipeline1 and $T(\text{Quantize}_{p2})$ is the Quantize time of pipeline2.

$$T(\text{Quantize}_{p1}) = -264847 - 2.47(H) + 4.40(W) + 29.025(T(M, Q1)) + 0.19(T(N1, Q2)) - 0.031(W(Q1)) - 0.399(W(Q2)) \quad (4)$$

$$T(\text{Quantize}_{p2}) = 160.78 - 0.022(H) - 0.079(W) + 0.185(T(M, Q1)) + 0.002(T(N1, Q2)) + 1.04(T(N2, Q3) - 0.004(W(Q1)) - 0.023(W(Q2)) \quad (5)$$

5.3 Analysis of Compress Time

Here $T(\text{Compress}_{p1})$ is the Compress time of Pipeline1 and $T(\text{Compress}_{p2})$ is the Compress time of pipeline 2. Equation 6 and 7 gives the multivariate regression using the results of Table 3 to find the interdependency of attributes while compression in Pipeline1 and Pipeline2.

$$T(\text{Compress}_{p1}) = -1182387 - 0.0827(H) + 0.066(W) - 3.673(T(M, Q1)) - 0.022(T(N1, Q2)) - 0.20(T(N1, Q2)) - 0.002(W(Q1)) + 0.005(W(Q2)) - 3.673(T(N3M)) \tag{6}$$

$$T(\text{Compress}_{p2}) = 139124 + 0.524(H) + 0.218(W) - 44.54(T(M, Q1)) - 0.1(T(N1, Q2)) - 0.125(T(N1, Q2)) - 0.05(W(Q1)) - 0.124(W(Q2)) + 60.113(T(N3, M)) \tag{7}$$

The accuracy of multivariate analysis is measured in terms of R^2 value which is approximately equals to 1. Expect for Compression in pipeline 2 which is 0.799 in all cases we have achieved maximum accuracy. Table 5 represents the R^2 values for above multivariate analysis.

Table 5. R^2 values of different multivariate analysis

Operation Name	R^2 value for Pipeline1 Analysis	R^2 value for Pipeline2 Analysis
Resize	0.967	0.985
Quantize	0.883	0.997
Compress	0.972	0.799

Actual computational time is very small as compared to queuing time in different queues. Every stage in pipeline computation carries a queue and before actual processing each job has to wait in a queue. At every stage of computation when the computational node becomes free job waiting in a queue will be taken for computation. Results in Table 3 elaborates job properties i.e. its height and width and also by which pipeline it is executed i.e. either Pipeline1 or Pipeline2. Using the results in Table 3 we have plotted the graph of pipeline utilization. Fig. 4 and Fig. 5 shows graphical representation of percent the utilization of Pipeline 1 and Pipeline 2.

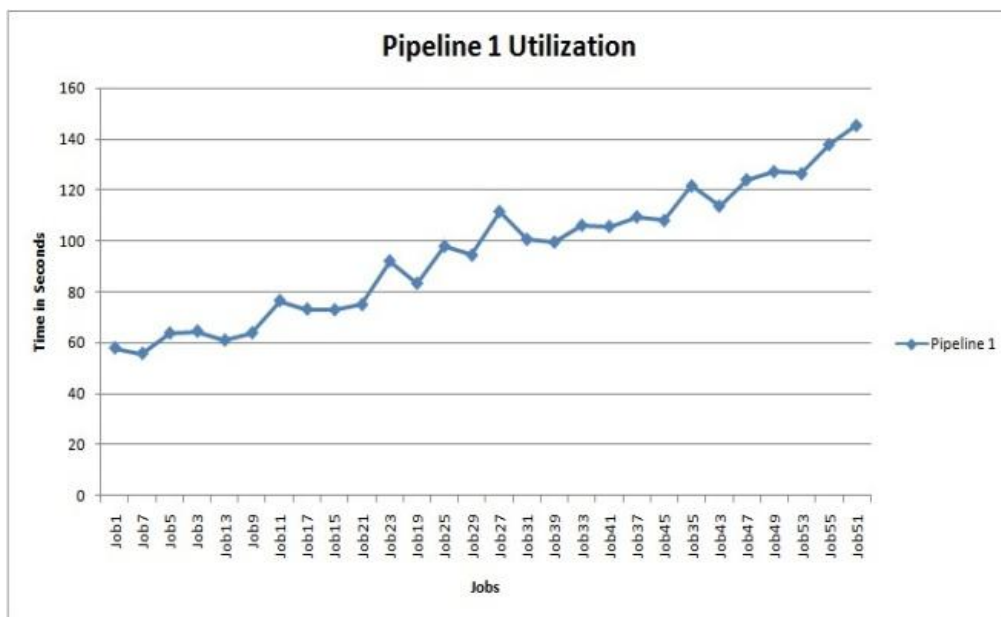


Fig. 4. Utilization of pipeline 1

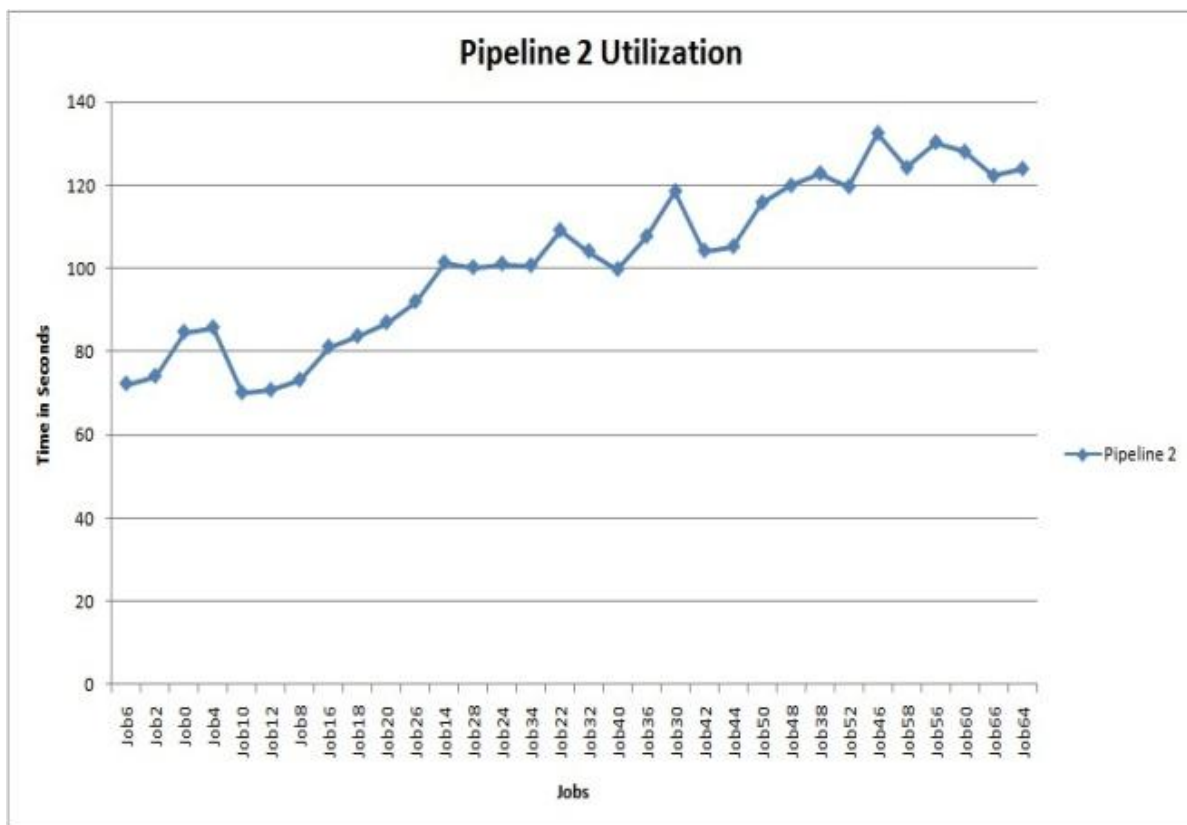


Fig. 5. Utilization of pipeline 2

Basic objective of Dynamic load balancing in our experimentation is to achieve maximum load balancing within two pipelines. We are claiming maximum load balancing but it should be proven by results obtained while experimentation. Using experimental results we can prove the fact by testing hypothesis. There are only two possibilities i.e. task will be assigned to either Pipeline 1 or 2. Equation 8 gives probabilistic relation of Pipeline 1 and pipeline 2 as,

$$P(1) = P(P1) + P(P2) \tag{8}$$

Where, P(1) = Total Probability=1,

P(P1) = Probability of job executed by Pipeline 1 and

P(P2) = Probability of job executed by Pipeline 2.

Requirement of accurate load balancing P(P1)= P(P2)=0.5. As we are testing the hypothesis for proportions we will use Hypothesis testing of Proportions using t-Test: Two-Sample Assuming Unequal Variances.

$$H_0 : \mu(P1) = \mu(P2)$$

$$H_a : \mu(P1) \neq \mu(P2)$$

$$\alpha = 0.05$$

Table 6. t-Test: Two-Sample Assuming Unequal Variances

	Pipeline 1	Pipeline 2
Mean	95.29532	102.0272
Variance	670.1098	372.3669
Observations	28	32
Hypothesized Mean Difference	0	
df	49	
t Stat	-1.12875	
P(T<=t) one-tail	0.13225	
t Critical one-tail	1.676551	
P(T<=t) two-tail	0.264499	
t Critical two-tail	2.009575	

Table 6 shows that the value of t Stat is larger than t Critical two-tail and also P two tail is larger than α i.e. 0.05 we accept H_0 and reject H_a . In this hypothesis, we have tested the load balancing factor in three stage pipeline architecture. This hypothesis testing concludes that we achieved impartial load balancing in the image retargeting process.

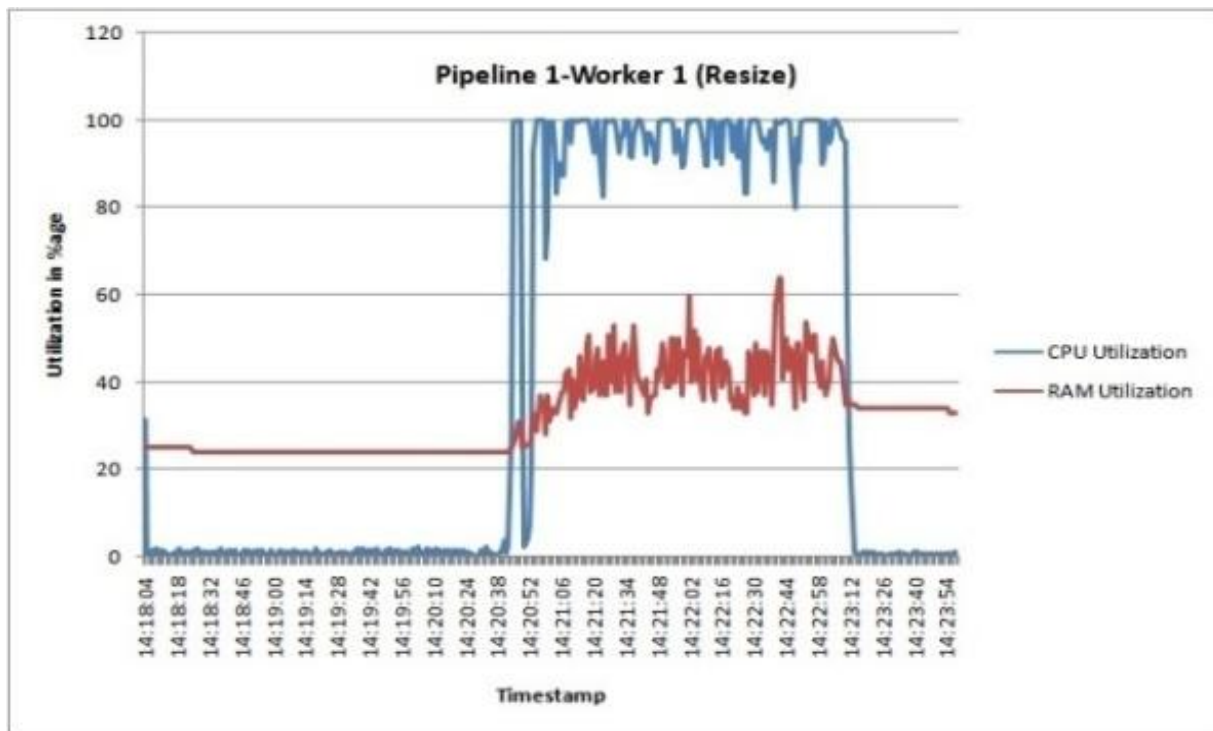


Fig. 6. Utilization of Pipeline 1-Worker 1 (Resize)

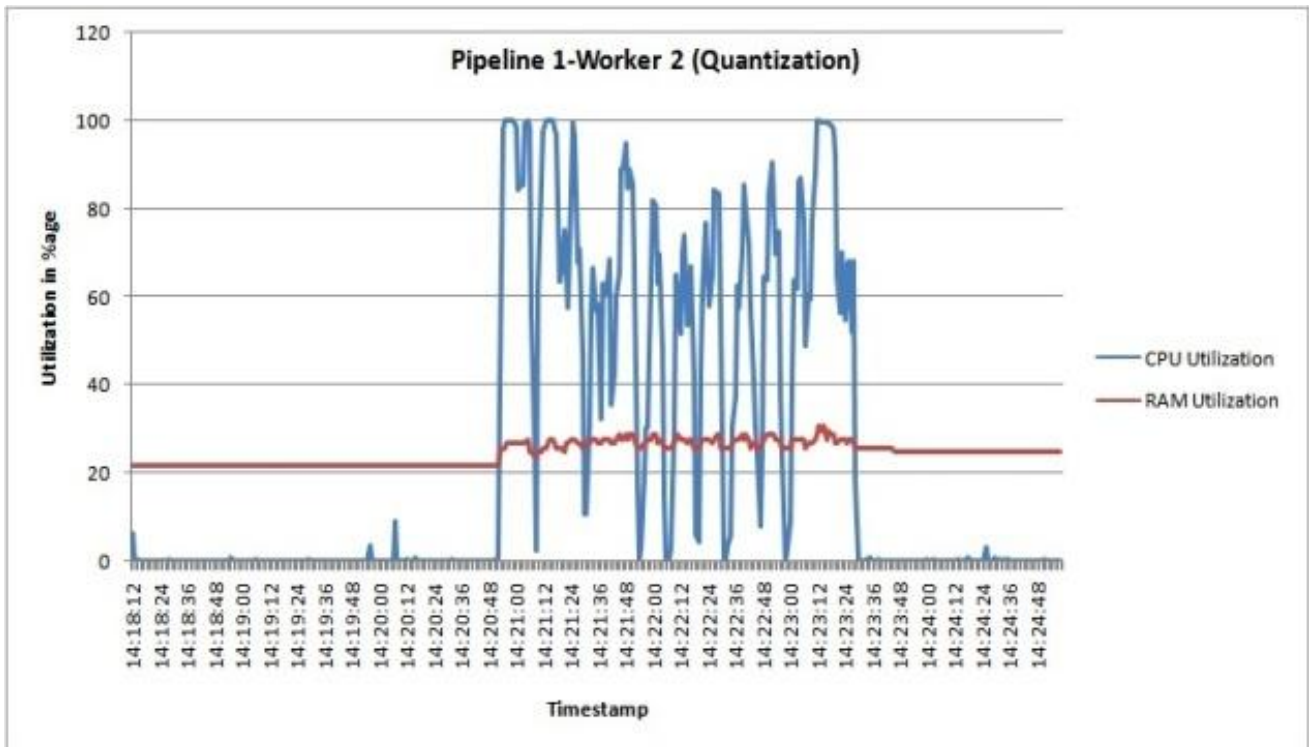


Fig. 7. Utilization of Pipeline 1-Worker 2 (Quantize)

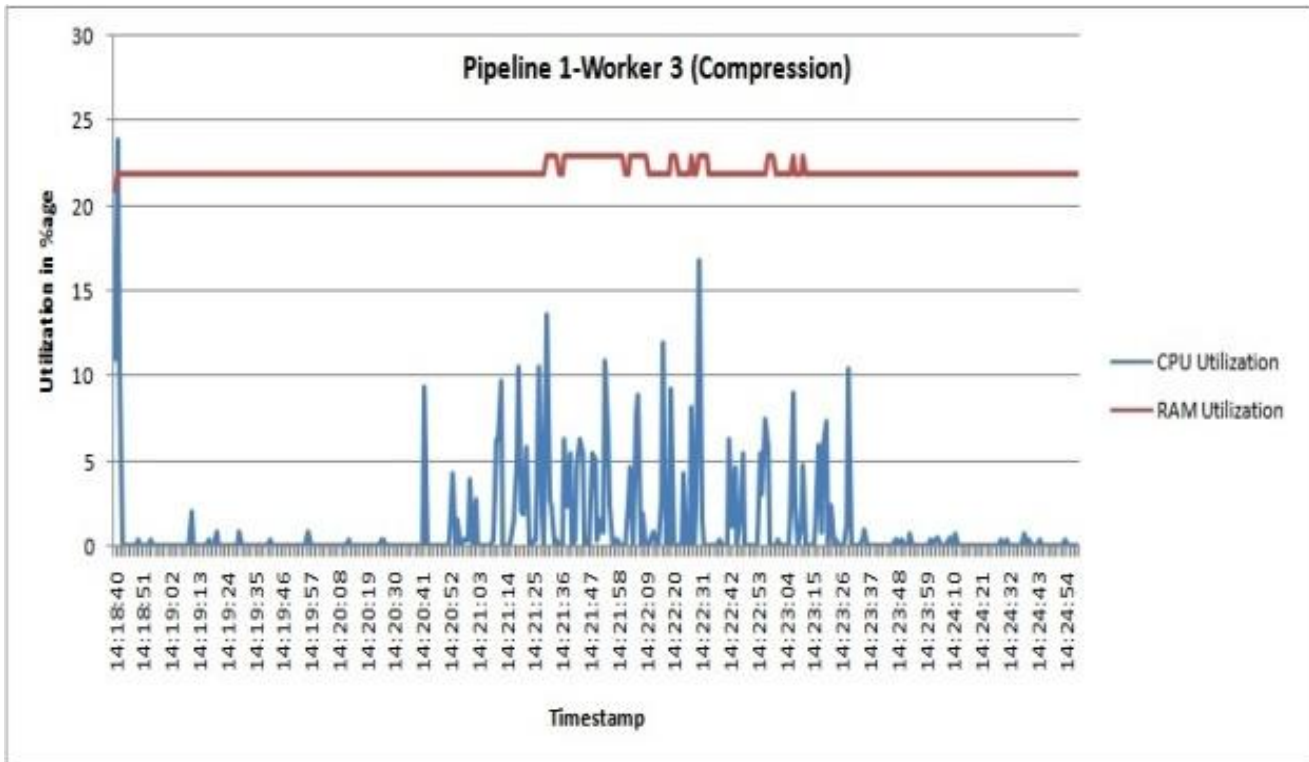


Fig. 8. Utilization of Pipeline 1-Worker 3 (Compress)

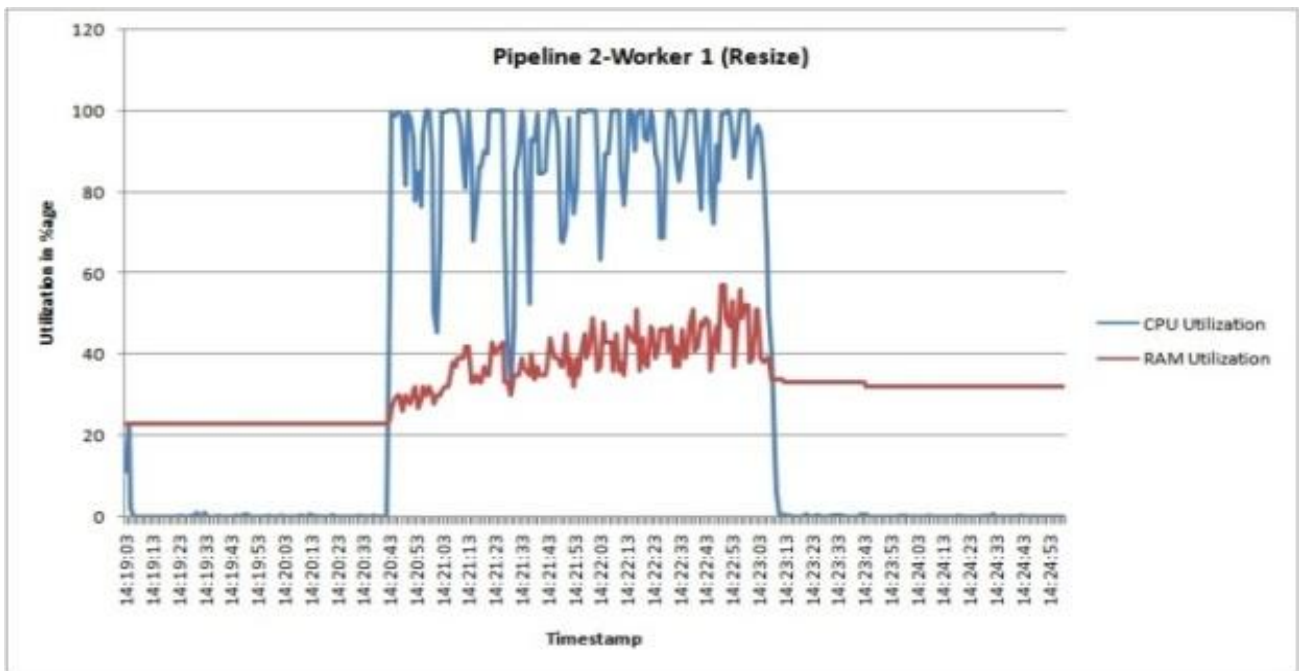


Fig. 9 Utilization of Pipeline 2-Worker 1 (Resize)

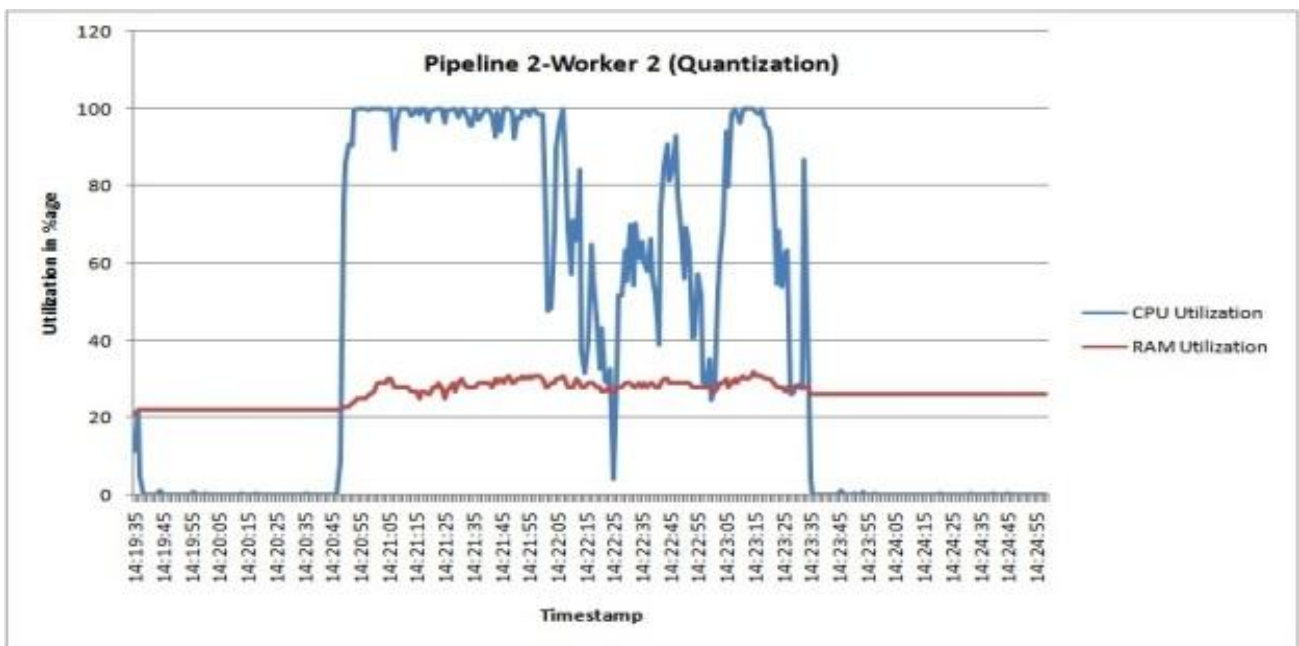


Fig. 10 Utilization of Pipeline 2-Worker 2 (Quantize)

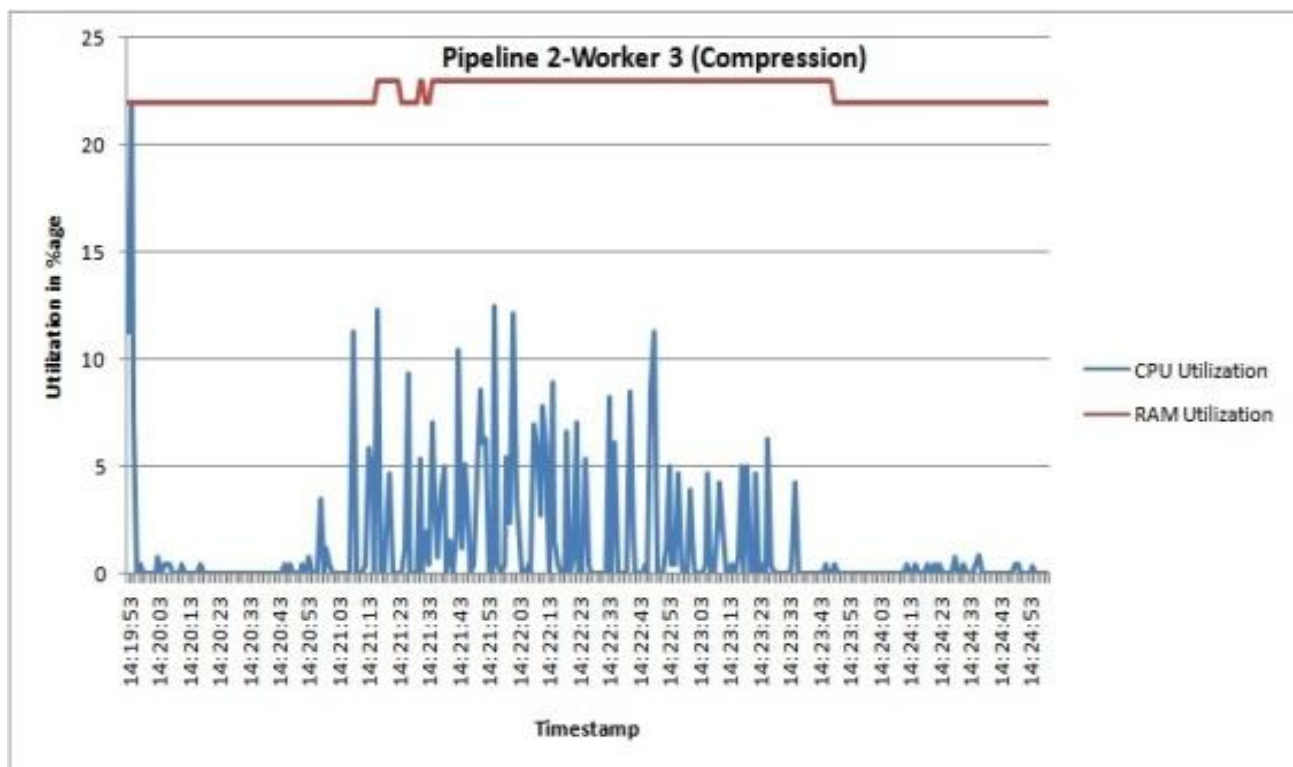


Fig. 11. Utilization of Pipeline 1-Worker 3 (Compress)

Fig. 6 to Fig. 11 shows the utilization graph of CPU and RAM of each node. As our task consists of sequence of Resize, Quantize and Compress operation the utilization of each node change with respect to time. We are achieving maximum utilization i.e. in some time slots 100% utilization of CPU but RAM utilization is quite less as compared to CPU utilization. The CPU utilization ranges in minimum 20% to maximum 100%. The RAM utilization ranges in minimum 23% to maximum 66%. In our experimental studies image retargeting task is experienced as more CPU intensive task than RAM. The Fig. 6 to Fig. 11 shows the achievement of resource utilization objectives which are mentioned in the introduction section.

When we apply regression analysis on results in Table 3, we get a overall regression Equation 11 as,

$$\text{Total Time}(T) = 7.43 + 0.75(Rt) + 0.96(Qt) + 0.068(Ct) + 0.98(WQt)(9)$$

Where, Rt=Resize time, Qt=Quantize time, Ct=Communication Time and WQt=Total Waiting time in Queue.

Communication time has a very negligible influence on total image retargeting so it has not included in Equation 10. The regression analysis carried out by using equation 11 is obtained at $R^2=0.984$, which shows the accuracy of the relational linear equation for image retargeting operation. The average error calculated with this equation is 0.009834 which is negligible. Table 5 shows difference between expected values of Image retargeting time using Equation 9 and actual experimental image retargeting time.

Table 5. Expected values and Actual values of image retargeting obtained by using equation 9

Job	1	2	3	4	5	6	7	8	9	10
Expected Time	63.0	79.1	69.1	90.3	68.5	77.1	60.7	78.3	69.0	75.5
Actual Time	60	76	69	96	69	77	62	79	71	75
Error	3.08	3.16	0.10	-6.3	-0.5	0.10	-2.3	0.70	-2.0	0.50

Table 6 shows some sample input image and output images obtained by three phase Image retargeting process.

One of the objectives of our image retargeting work is to reduce storage required by the images in any acute multimedia gadget. Limited storage available in handheld communicational devices gives importance to this storage optimization. The vastly expanding era of multimedia communication requires optimized image storage. Table 7 shows the amount of storage required by input image and storage required by output images obtained by three phase image retargeting process.

We assume storage required as a linear function of Height and Width of the image as shown in Equation 10,

$$S = f(H,W) \quad (10)$$

To trace the relation between the storage required i.e S and height and width of image, we perform regression analysis of the values of input image attributes in Table 7. The regression analysis gives us a relational Equation 11.

$$S = -2539.7 + 2.65H + 1.61W \quad (11)$$

Table 9 represents the difference between the values of actual storage obtained by using image retargeting operations and the expected storage values obtained by applying equation 11 to the same image.

Table 9 Actual size and Expected size of output image

Output Image No.	H	W	Actual Storage in KB	Expected storage in KB
1	320	213	40.3	-1348.77
2	320	480	82.2	-918.9
3	320	480	106	-918.9
4	320	480	77.6	-918.9
5	320	480	78.3	-918.9
6	639	426	148	-160.49
7	640	960	330	701.9
8	640	960	383	701.9
9	640	960	267	701.9
10	640	960	280	701.9
11	800	533	234	438.43
12	800	1200	526	1512.3
13	800	1200	579	1512.3
14	800	1200	410	1512.3
15	800	1200	427	1512.3
16	1023	682	379	1269.27
17	1024	1536	881	2646.86
18	1024	1536	912	2646.86
19	1024	1536	682	2646.86
20	1024	1536	691	2646.86
21	1280	853	591	2225.63
22	1280	1920	1370	3943.5
23	1280	1920	1330	3943.5
24	1280	1920	1050	3943.5
25	1280	1920	1040	3943.5

26	1920	1280	1350	4609.1
27	1920	1280	3070	4609.1
28	1920	1280	2810	4609.1
29	1920	1280	2510	4609.1
30	1920	1280	2350	4609.1

Fig. 12 represents a graph of expected storage versus actual storage. In some cases the storage achieved by experimentation is greater than the expected storage, but in most of the cases it is less than expected storage. Hence, overall storage optimization is achieved in our three phase image retargeting process.

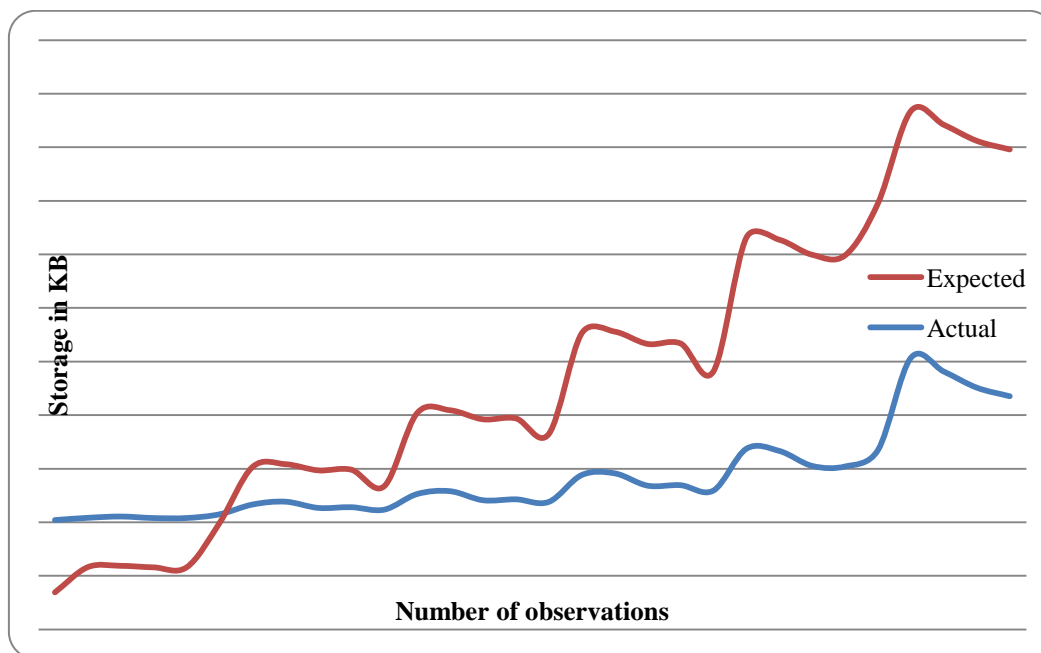


Fig. 12. Expected storage Vs Actual storage

6. Conclusion

In this work, we proposed a dynamic load balancing based three phase image retargeting methodology using pipeline architecture. In the first phase of image retargeting we have resized the input image to attain the aspect ratio of desired display size device. In this phase we have fixed the dimensions of output images. Output of first phase is given as input to quantization phase. In quantization phase, color palletization is achieved to reduce the size of color vector. The output of quantization phase is given as an input to compression phase. Compression phase outputs images by applying lossless compression.

Experimental work of image retargeting is carried out using central scheduling mechanism. The master node performed scheduling of jobs using runtime CPU and RAM load information of slave nodes. We have used two separate pipelines P1 and P2 for dynamic load distribution. Result analysis confirmed the proof of impartial dynamic load distribution among two pipelines using hypothesis testing. As the operations performed in proposed three phase image retargeting process are sequentially ordered, we have used pipeline architecture. But the waiting time in queue is 80% of total time required and only 20% of total processing time is utilized effectively for actual computations. In proposed work we have achieved CPU utilization in range of 20% to 100% and RAM utilization in range of 23% to 66%. The resource utilization of node performing compress operation in each pipeline is less. We have given difference between the storage requirement of input node and output node which confirms the storage optimization. Statistical analysis given in proposed work expounds the regression analysis. The weighted linear relational equations are useful to understand the effect of each operational parameter on overall completion time.

References

1. Ganesh Patil, Santosh Deshpande, "Image optimisation using dynamic load balancing", *International Journal of Knowledge Engineering and Data Mining*, Vol. 5, Nos. 1/2, pp.68-89, 2018.
2. Sumai Khan, Babar Nazir, Iftikhar Ahmed Khan, Shahaboddin Shashirband and Anthony T Chronopoulos, "Load Balancing in Grid Computing: Taxonomy, Trends and Opportunities," *Journal of Network and Computer Applications*, Accepted on 20 February 2017.
3. Shang-Liang Chen, Yun-Yao Chen, Suang-Hong Kuo, "CLB: A novel load balancing architecture and algorithm for cloud services," *Computers and Electrical Engineering*, pp. 1-7, Accepted on 12 January, 2016.
4. Bokhari, M.U., Alam, M., Hasan, F., "Performance Analysis of Dynamic Load Balancing Algorithm for Multiprocessor Interconnection Network," *Perspectives in Science*, pp. 1-5, June 2016.
5. Patil, G. and Deshpande, S.L., "Distributed rendering system for 3D animation with blender," Presented at IEEE Sponsored International Conference on Advances in Electronics, Communication and Computer Technology (ICAECCT 2016), pp.92–98, December 2016.
6. Bin Zhou, Xuanyin Wang, Songxiao Cao, Ke Xiang, Shuo Zhao, "Optimal bi-directional seam carving for compressibility-aware image retargeting," *J. Vis. Commun. Image R.*, September 2016.
7. Yanxiang Chen, Yifei Pan, Minglong Song, Meng Wang, "Improved seam carving combining with 3D saliency for image retargeting," *Neurocomputing* 151, pp. 645–653, October 2014.
8. Tali Dekel (Basha), Yael Moses, Shai Avidan, "Stereo Seam Carving a Geometrically Consistent Approach," *IEEE Transactions on Pattern analysis and machine intelligence*, Vol. 35, No.10, pp. 2513-2525, October 2013.
9. Bin Dong, Xiuqiao Li, Qimeng Wu, Limin Xiao, Li Ruan "A dynamic and adaptive load balancing strategy for parallel file system with large-scale I/O servers," *J. Parallel Distrib. Comput.* 72 (2012) pp. 1254–1268, May 2012.
10. Yuming Fang, Zhenzhong Chen, Weisi Lin, Chia-Wen Lin, "Saliency Detection in the Compressed Domain for Adaptive Image Retargeting," *IEEE Transactions on Image Processing*, Vol. 21, No. 9, pp-3888-3901, September 2012.
11. Doaa M. Abdelkader, Fatma Omara, "Dynamic task scheduling algorithm with load balancing for heterogeneous computing system," *Egyptian Informatics Journal*, pp. 135–145, July 2012.
12. Qingqi Long, Jie Lin, Zhixun Sun, "Agent scheduling model for adaptive dynamic load balancing in agent-based distributed simulations," *Simulation Modelling Practice and Theory* 19, pp. 1021–1034, January 2011.
13. Joni-Matti Maatta, Jarno Vanne, Timo D. Hamalainen, and Jarno Nikkanen "Generic Software Framework for a Line-Buffer-Based Image Processing Pipeline" *IEEE Transactions on Consumer Electronics*, Vol. 57, No. 3, pp. 1442-1449, August 2011.
14. Satish Penmatsa, Anthony T. Chronopoulos "Game-theoretic static load balancing for distributed systems," *J. Parallel Distrib. Comput.* 71, pp. 537–555, 2011, December 2010.
15. Jin Sun, Haibin Ling "Scale and Object Aware Image Retargeting for Thumbnail Browsing," Presented at IEEE International Conference on Computer Vision, pp.1511-1518, November 2011.
16. Jun-Seong Kim, Jin-Hwan Kim, and Chang-Su Kim "Adaptive Image and Video Retargeting Technique Based on Fourier Analysis," Presented at IEEE Conference on Computer Vision and Pattern Recognition,

pp.1730-1737, June 2009.

17. Jin-Hwan Kim, Jun-Seong Kim, and Chang-Su Kim “Image and Video Retargeting using Adaptive Scaling Function” 17th European Signal Processing Conference (EUSIPCO 2009), Glasgow, Scotland, pp.819-823, August 24-28, 2009.
18. Shu-Fan Wang and Shang-Hong Lai “Fast Structure Preserving Image Retargeting ” Presented at International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1049-1052, April 2009.
19. Kang-Sun Choi, Sung-Jea Ko, “Fast Content-Aware Image Resizing Scheme in the Compressed Domain” IEEE Transactions on Consumer Electronics, Vol. 55, No. 3, August 2009.
20. CHEN Wei-dong, DING Wei “An Improved Median-cut Algorithm of Color Image Quantization” IEEE International Conference on Computer Science and Software Engineering, pp.943-946, December 2008.
21. Grosu, D. and Chronopoulos, A.T., “Algorithmic mechanism design for load balancing in distributed systems,” IEEE Trans. on Systems, Man, and Cybernetics – Part B: Cybernetics, Vol. 34, No. 1, pp.77–84, February 2004.
22. David E. Rex, Jeffrey Q. Ma, and Arthur W. Toga* “The LONI Pipeline Processing Environment” NeuroImage 19, pp.1033-1048, 18 March 2003.
23. Danial Grosue, Anthony T. Chronopoulos, Ming-Ying Leung (2002), “Load balancing in distributed systems: A game theoretic approach”, Presented at IEEE International Parallel and Distributed Processing Symposium (IPDPS 2002), pp.501-510, April 2002.
24. Zomaya, A.Y. and Teh, Y-H., “Observations on using genetic algorithms for dynamic load-balancing,” IEEE Trans. on Parallel and Distributed Systems, , Vol. 12, No. 9, pp.899–911, September 2001.
25. Jaspal Subhlok, Gary Vondran, “Optimal Use of Mixed Task and Data Parallelism for Pipelined Computations ” Journal of Parallel and Distributed Computing Vol. 60, pp.297-319, March 2000.
26. Watts, J. and Taylor, S., “A practical approach to dynamic load balancing,” IEEE Trans. on Parallel and Distributed Systems, March, Vol. 9, No. 3, pp.235–248, March 1998.
27. Willebeek-LeMair, M.H. and Reeves, A.P., “Strategies for dynamic load balancing on highly parallel computer,” IEEE Trans. on Parallel and Distributed Systems, Vol. 4, No. 9, pp.979–993 September 1993.
28. Lin, H-C. and Raghavendra, C.S., “A dynamic load-balancing policy with a central job dispatcher (LBC),” IEEE Trans. on Software Engineering, , Vol. 18, No. 2, pp.148–158, February 1992.
29. Chow, Y-C. and Kohler, W.H. “Models for dynamic load balancing in a heterogeneous multiple processor system,” IEEE Trans. on Computers, Vol. C-28, No. 5, pp.354–361, May 1979.
30. World Wide Survey, <https://mylio.com/true-stories/tech-today/how-many-digital-photos-will-be-taken-2017-repost> accessed on (12th March, 2018)
31. Image Processing available at: <http://interactivepython.org/runestone/static/thinkcspy/MoreAboutIteration/2DimensionalIterationImageProcessing.html> (accessed on 2nd Nov, 2017)
32. Michael Still “The Definitive Guide to ImageMagick” Apress, 2006.
33. Image Quantization available at [https://en.wikipedia.org/wiki/Quantization_\(image_processing\)](https://en.wikipedia.org/wiki/Quantization_(image_processing)) (accessed on 6th Nov, 2017)
34. B. Durakovic, "Design of Experiments Application, Concepts, Examples: State of the Art," Periodicals of

Engineering and Natural Sciences, Vol. 5, No. 3, pp.421–439, December 2017.

35. B. Duraković, H. Bašić, "Continuous Quality Improvement in Textile Processing by Statistical Process Control Tools: A Case Study of Medium-Sized Company", Periodicals of Engineering and Natural Sciences, Vol. 1, No. 1, pp 36–46, December 2013.

36. Advpng available at <http://www.advancemame.it/doc-advpng.html>(accessed on 6th Nov,2017)

37. Grafana available at <https://grafana.com>(accessed on 6th Nov,2017)

38. Pngquant available at <https://pngquant.org>(accessed on 6th Nov,2017)

Table 6. Sample Input Image and Output Images















3072X2014 8	Output Images					
	320X213	639X426	800X533	1023X853	1280X853	1920X1280
						
2048X3072	320 X	639 X 426	800 X 533	1023 X	1280 X 853	1920 X 1280
						

Table 7 Storage required by input and output images

Name of Image	msp_0404	Result 1	Result2	Result 3	Result 4	Result 5	Result 6
Height	3072	320	639	800	1023	1280	1920
Width	2048	213	426	533	682	853	1280
Storage Required	12048 KB	40.3 KB	148 KB	234 KB	379 KB	591 KB	1.35 MB
Name of Image	msp_0507	Result 1	Result2	Result 3	Result 4	Result 5	Result 6
Height	2048	320	640	800	1024	1280	1920
Width	3072	480	960	1200	1536	1920	2880
Storage Required	13072 KB	82.2 KB	330 KB	526 KB	881 KB	1.37 MB	3.07 MB
Name of Image	msp_904	Result 1	Result2	Result 3	Result 4	Result 5	Result 6
Height	2912	320	640	800	1024	1280	1920
Width	4368	480	960	1200	1536	1920	2880
Storage Required	14368 KB	106 KB	383 KB	579 KB	912 KB	1.33 MB	2.81 MB
Name of Image	msp_1306	Result 1	Result2	Result 3	Result 4	Result 5	Result 6
Height	3840	320	640	800	1024	1280	1920
Width	5760	480	960	1200	1536	1920	2880

Storage Required	15760 KB	77.6 KB	267 KB	410 KB	682 KB	1.05 MB	2.51 MB
Name of Image	msp_1306	Result 1	Result2	Result 3	Result 4	Result 5	Result 6
Height	5760	320	640	800	1024	1280	1920
Width	3840	480	960	1200	1536	1920	2880
Storage Required	13840 KB	78.3 KB	280 KB	427 KB	691 KB	1.04 MB	2.35 MB

Table 3. Results obtained in Image Retargeting using dynamic load balancing

Job No	Height	Width	Resize Time	Quantize Time	Compress Time	Pipeline used	Queuing Time	Computational Time	Communication Time	Total Time
	In Pixels	In Pixels	In Sec.	In Sec.	In Sec.		In Sec.	In Sec.	In Sec.	In Sec.
1	252	160	5.585	7.052	6.8	1	45.135	12.705	2.16	60
2	252	160	4.61	19.046	9.7	2	50.341	23.753	1.906	76
3	239	160	6.165	12.573	4.1	1	45.657	18.779	4.564	69
4	238	160	5.176	24.362	8.7	2	56.133	29.625	10.242	96
5	238	160	5.753	12.35	4.4	1	45.55	18.147	5.303	69
6	3072	2048	7.765	14.112	17.2	2	50.216	22.049	4.735	77
7	1023	682	6.113	5.581	5.1	1	43.925	11.745	6.33	62
8	1280	853	5.037	6.211	8.6	2	61.838	11.334	5.828	79
9	1920	1280	5.737	5.109	8.2	1	52.871	10.928	7.201	71
10	320	213	4.014	7.897	8.5	2	58.145	11.996	4.859	75
11	3840	2560	9.33	7.091	8.9	1	59.956	16.51	0.534	77
12	639	426	4.817	7.742	7.2	2	58.102	12.631	4.267	75
13	800	533	5.171	6.432	8.3	1	49.208	11.686	9.106	70
14	3072	2048	6.461	18.222	7.8	2	76.623	24.761	1.616	103
15	1023	682	5.396	9.176	8.4	1	58.186	14.656	5.158	78
16	1280	853	4.529	9.041	8.9	2	67.451	13.659	3.89	85
17	1920	1280	4.931	7.721	8.8	1	60.372	12.74	4.888	78
18	320	213	4.303	11.799	9.5	2	67.511	16.197	1.292	85
19	3840	2560	8.202	9.304	5.8	1	65.716	17.564	2.72	86
20	639	426	4.676	10.303	8.6	2	71.896	15.065	3.039	90
21	800	533	5.317	9.223	7.9	1	60.267	14.619	6.114	81
22	2048	3072	15.814	8.357	8	2	84.863	24.251	1.886	111
23	1024	1536	14.156	5.64	7.7	1	72.23	19.873	3.897	96
24	1280	1920	14.092	7.014	8.1	2	79.876	21.187	5.937	107
25	1920	2880	16.143	4.913	7.6	1	76.724	21.132	0.144	98

Table 8. Input image attributes

Input Image	1	2	3	4	5	6	7	8	9	10
Height	3072	2048	2912	3840	3072	4368	2912	5760	3840	3840
Width	2048	3072	4368	5760	2048	2912	4368	3840	5760	5760

Size in KB	12048	13072	14368	15760	12048	12912	14368	13840	15760	15760
------------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------
