Render Farm for Highly Realistic Images in a Beowulf Cluster using Distributed Programming Techniques

Enrique Lee Huamaní[1], Patricia Condori[2], Brian Meneses-Claudio[3], Avid Roman-Gonzalez[4]
Image Processing Research Laboratory (INTI-Lab)
Universidad de Ciencias y Humanidades
Lima, Perú

***Abstract*—Now-a-days, photorealistic images are demanded for the realization of scientific models, so we use rendering tools that convert three-dimensional models into highly realistic images. The problem of generating photorealistic images occurs when the three-dimensional model becomes larger and more complex, so the time to generate an image is much greater due to the limitations of hardware resources, about this problem is implemented the render farm, which consists in a set of computers interconnected by a high-speed network that provides a strip of the global image distributed in each participating computers with the intention of reducing the processing time of highly complex computational images. The research was implemented in a high-performance Beowulf group of the Universidad de Ciencias y Humanidades using a total of 18 computers. To demonstrate the efficiency of a rendering farm implementation, scalability tests were performed using a 360° equirrectangular model with a total of 67 million pixels, the work is carried out to achieve highly complex renderings in less time to benefit the direction of the research.**

Keywords—Distributed programming; computational parallelism; Beowulf cluster; high-efficiency computing; render farm

I. Introduction

In the last decades, a great demand began to grow in the generation of high-realism images where three-dimensional models require greater hardware resources to solve renderings of high complexity due to this need the rendering farms are implemented [1] which are groups of computers that they are interconnected with each other through a network that distributes the work of the images to each participating computers to obtain results of photorealistic images in less time.

The problematic of this investigation is the rendering of high realism images that due to the complexity of the project, the rendering processes can take hours or days to return a result of a complex image. Because of this problem, the research direction of the Universidad de Ciencias y Humanidades puts at your disposal the use of the high performance Beowulf cluster implemented by [2], which is located in the embedded systems laboratory. This work uses this architecture for the integration of a rendering farm using the Python programming language that will determine the amount of participating computers to distribute each strip of the image in order to achieve complex renderings in less time.

There are research related to the benefit of using rendering farms in a Beowulf cluster, as in the case of [3] that renders complex images using a total of six computers where it isn't necessary to buy specialized equipment at high costs, other research that try to emphasize the reduction of costs that it has [4] that compare the performance of their existing systems that of the high-ending machines used in the modern animation industry, likewise in this paper, it uses the existing hardware resources. There are different ways to get rendering farms, as is the research of [5] that uses a queue management software to get distributed renderings, instead this project uses distributed programming techniques using Python programming language of which the reader can customize the size of the image by each participating computer. There is also research related to cloud computing, a clear example is [6], which performs a hybrid rendering farm that can be adapted to the server cloud to increase computational capacity, therefore the use of a rendering farm in a Beowulf cluster will achieve greater efficiency in generating complex images and lower costs.

II. Methodology

For the realization of the rendering farms the Beowulf cluster architecture of the embedded systems laboratory of the Universidad de Ciencias y Humanidades, it was used with 17 slave computers and a master computer, all of them with the same hardware characteristics and under the Linux platform with Ubuntu distribution. The main elements that the machines must have are Blender[1], SSH open, ImageMagick and the Python programming language that will divide the original image into small strips of images that will be distributed to each of the participating computers to perform the rendering. Fig. 1 shows the overall architecture of the project.

A. *Beowulf Cluster Physical Architecture*

Beowulf's high-performance clusters are a group or conglomerates of computers interconnected through a high-speed network that are typically used to apply computational parallelism techniques in order to obtain results in a shorter time [7]. This architecture works with low-cost computers that try to give the similarity of having a supercomputer that is possible thanks to the manipulation of the central processing unit (CPU) therefore, this architecture has a total of 204 cores

[1] Blender Foundation, "blender.org - Home of the Blender project - Free and Open 3D Creation Software" 2019. [Online]. Available: https://www.blender.org/

to perform rendering, Fig. 2 shows the high performance cluster type Beowulf performing a rendering of a prototype of a three-dimensional image.

Each computers used in the Beowulf cluster architecture has the same hardware characteristics, as shown in Table I.

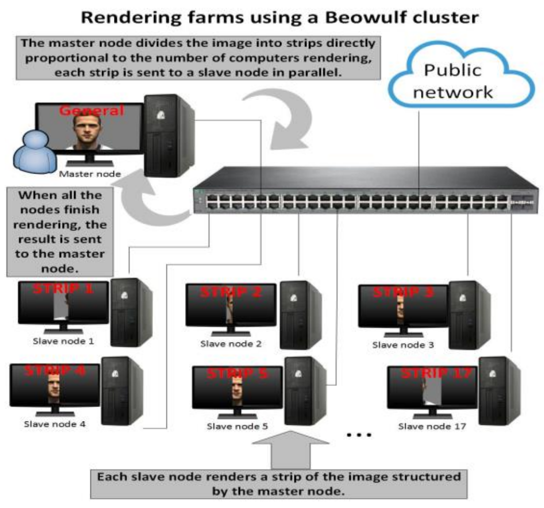

Rendering farms using a Beowulf cluster
The master node divides the image into strips directly proportional to the number of computers rendering, each strip is sent to a slave node in parallel.
Public network
General
Master node
When all the nodes finish rendering, the result is sent to the master node.
STRIP 1
Slave node 1
STRIP 2
Slave node 2
STRIP 3
Slave node 3
STRIP 4
Slave node 4
STRIP 5
Slave node 5
STRIP 17
Slave node 17
Each slave node renders a strip of the image structured by the master node.

Fig. 1. Design of Rendering Farms with the Beowulf Cluster.

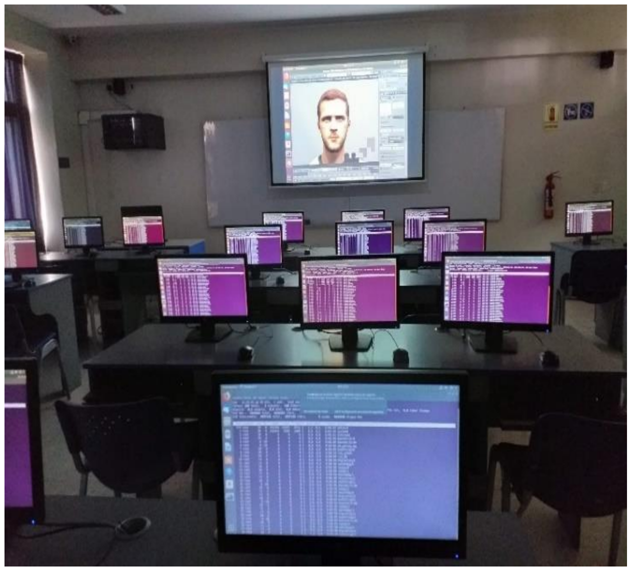


Fig. 2. Render Farm in a Beowulf Cluster.

TABLE. I. HARDWARE CHARACTERISTICS OF THE BEOWULF CLUSTER COMPUTERS

	Description
Modell	HP EliteDesk 800 G1 SFF
HDD	1 TB
RAM	8 GB
Processor	Intel(R) Core (TM) i7-8700
Total Cores	12
Type of Operating System	64-bit
Operative System	Ubuntu 18.04

B. *Master Node*

The computer in charge of distributing the strips of images to the participating computers is called the master node because it assigns the number of participating computers and the level of complexity to be solved.

C. *Slave Node*

The computer in charge of obtaining the strip of the image, rendering and returning it to its origin, it is called slave node which complies with the direct orders assigned to it from the master node, the slave and master nodes are interconnected through a high-speed network.

D. *Dependencies*

The following tools must be taken into account in each of the high-performance cluster machines.

1) Blender: As indicated by [8] blender is an open source 3D modeling and animation software, it is currently used by 2 million visual effects artists, animators and a growing number of astrophysicists. The advantage of using this software is that it is multiplatform and easy to install, it also includes methods for rendering complex surfaces and volumes, image composition, stereoscopic support for tracing graphics and the ability to export models and lighting in a variety of formats.

2) Python: It is an interpreted programming language where its syntax is legible code, this language is multiparadigmic [9] and easy to use, therefore is ideal for distributing the strips of images in each of the slave nodes, thanks to its simplicity the reader can configure and add new features to the algorithm to use it in different ways.

3) ImageMagick: It is a program built in open source C programming language, this software is used to create, edit, compose or convert images into bitmaps [10]. With respect to the project, the program executes the creation of the image in each one of the slave nodes to finally return them to the master node.

4) SSH protocol: Secure Shell (SSH) is a remote administration protocol that allows the reader to initiate commands and copy files from master node to slave nodes [11]. The SSH makes communication between computers secure due to a key generated from the master computer and copied to each of the slave nodes, this allows external devices can not access the Beowulf cluster because they do not have password identification.

E. *Process Flow of a Rendering Farm*

In the rendering process, distributed programming techniques are applied where a first, a validation is carried out to determine the number of slave nodes, if the condition of having more than one slave computer is met, the distributed rendering is carried out, but if only one computer is assigned, the traditional rendering will be used, in Fig. 3 the detailed rendering steps are explained in the form of a flow diagram.

Because this project is focused on rendering farms, the following distributed programming steps corresponding to the flowchart in Fig. 3 are explained.

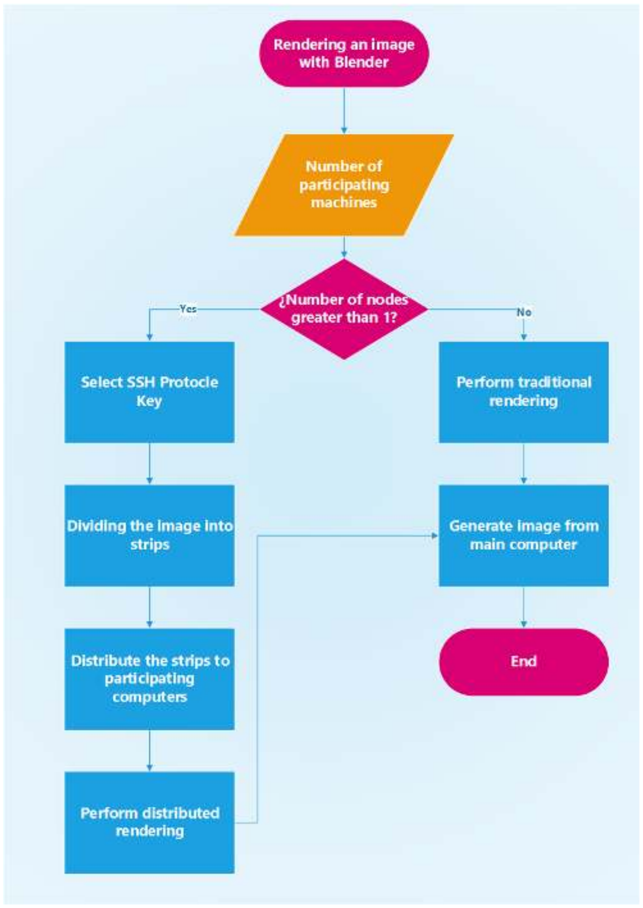

Rendering an image with Blender
Number of participating machines
¿Number of nodes greater than 1?
Yes
No
Select SSH Protocle Key
Perform traditional rendering
Dividing the image into strips
Generate image from main computer
Distribute the strips to participating computers
End
Perform distributed rendering

Fig. 3. Render Flowchart.

1) Selection of slave nodes: Before starting the rendering the reader must indicate the number of slave nodes that will participate. From the Linux terminal, it executes the Python code with the following sentence:

```
blender_network_render.py -m node1 node2 node3 node17
```

With this sentence, it indicates the number of nodes, in this case 17 slave nodes.

2) Selection of SSH protocol unique key: After classifying the participating nodes, they are assigned their identification key, this key allows unrestricted communication so it is assigned from the command line as following sentence:

```
blender_network_render.py -R
```

Which indicates the key that is shared in each of the slave nodes, this is done by security issues.

3) Strips of rendering on slave nodes: The image to be rendered is divided into vertical strips depending on the number of slave nodes assigned. Each strip of the image can take a certain size for rendering, this is useful for slave nodes with different hardware characteristics where the assignment of a strip of image must be inversely proportional to the rendering time of the image. With respect to the machines of the Beowulf cluster, all have the same hardware characteristic therefore the rendering images of high realism are not defined personalized strips for it will be distributed of equitable form as it is shown in the Fig. 4 in which the use of four computers is taken as example.

Depending on the complexity of each strip or on the characteristics in the hardware resources should be allocated an appropriate percentage for better efficiency. It will make the following assumption where 'Computer 0' and 'Computer 2' have fewer cores than 'Computer 1' and 'Computer 3' when the number of cores is not equal between the participating machines should be assigned a custom size of the strip image as shown in Fig. 5.

4) Obtaining the rendered image: Each machine renders a subset of the image as explained in the previous section, when all machines finish part of the render, the program collects all the strips in a final image, it is recommended that all participating machines have the same format with a color depth of 8 to 16 bits with the color specifier RGBA, with this format is possible to assemble images without interruptions, as shown in Fig. 6 to be configured in each slave node.

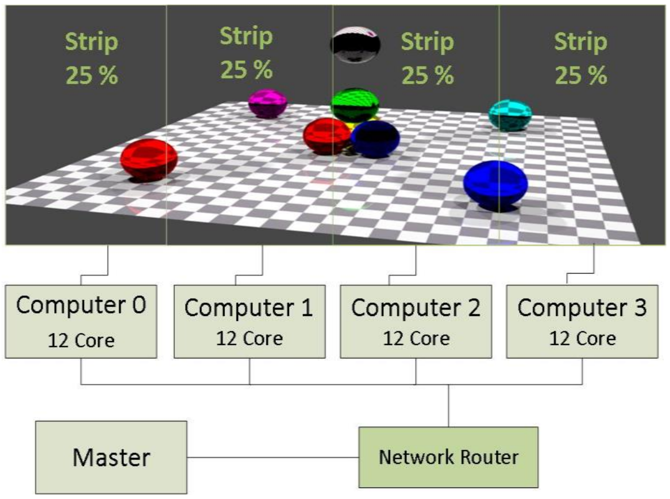


Fig. 4. Equitable Strips of Images.

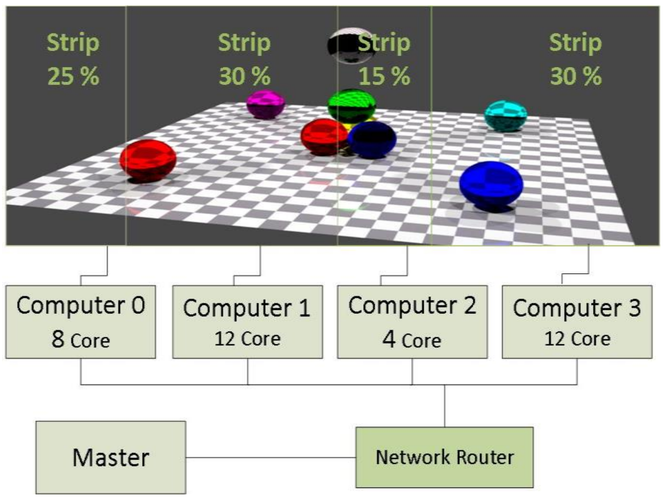
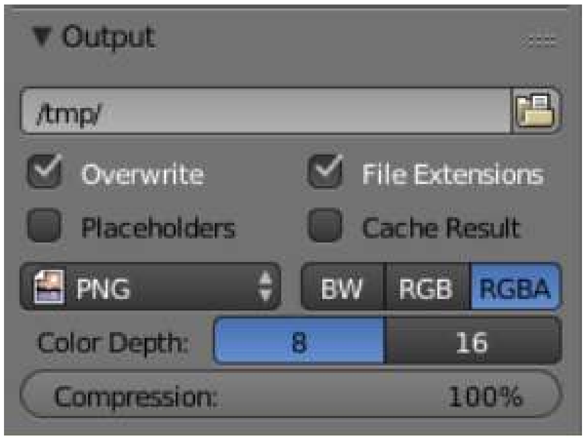


Fig. 5. Strips Images Proportional to the Number of Cores.



▼ Output

/tmp/

✓ Overwrite ✓ File Extensions

Placeholders Cache Result

PNG BW RGB [✓ RGBA]

Color Depth: [✓ 8] 16

Compression: 100%

Fig. 6. Output Image Format.

F. *Prototype of Rendering Tests*

To measure the efficiency of the rendering farms in the Beowulf cluster of the Embedded System laboratory, scalability tests are performed using a large and complex image prototype that is a spatial 3D stereoscopic image with 360° equirectangular projection, which are images used in a dimension of 16384x4096 pixels equivalent to more than 67 million pixels, Fig. 7 shows the prototype designed with Blender software.

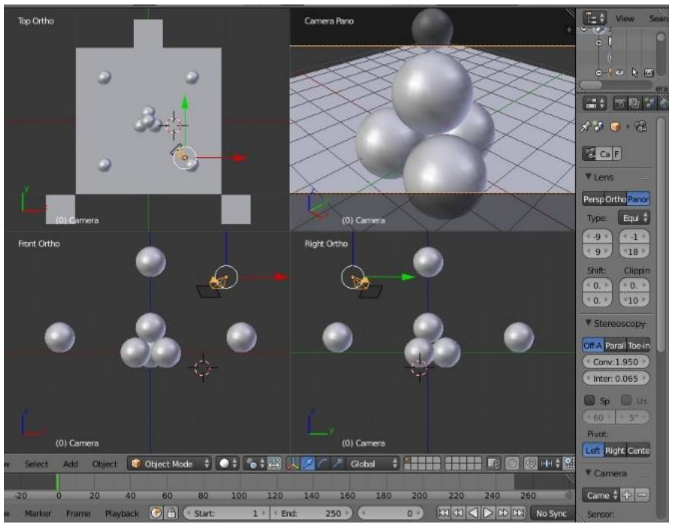


Fig. 7. Prototype of Equirectangular Image of 360°.

III. Result

In this section, the performance tests are carried out using as a project the 360° equirectangular image corresponding to Fig. 6. As shown in Table II, scalability tests are performed where the first column shows the number of participating nodes, the second column shows the time required for rendering and the third column shows the number of cores involved.

Fig. 8 shows graphically the render time with respect to the number of slave nodes involved.

As a result, it provides the image with respect to Fig. 9, which is a realistic 360° image with a dimension of 16384x4096 pixels, which demonstrates the reduction of time for high realism images using distributed programming techniques.

TABLE. II. High Realism Image Rendering Time

Number of Slave Nodes	Time of result	Core
1	1443	12
2	664.8	24
3	673.2	36
4	436.2	48
5	423	60
6	322.2	72
7	321	84
8	257.4	96
9	261	108
10	208.8	120
11	214.8	132
12	190.2	144
13	199.8	156
14	180	168
15	187.2	180
16	144	192
17	153.6	204

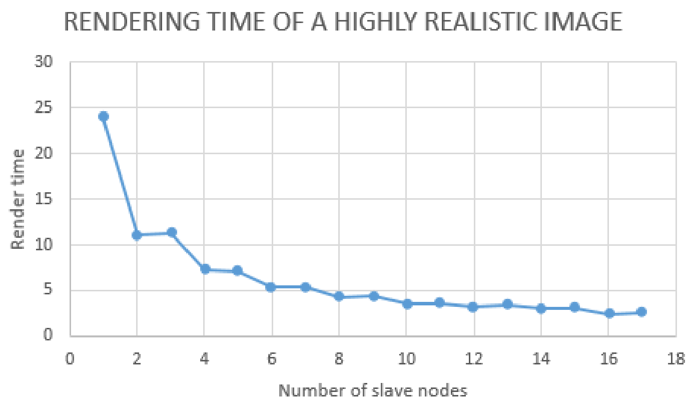


Fig. 8. Rendering Time of a Highly Realistic Image

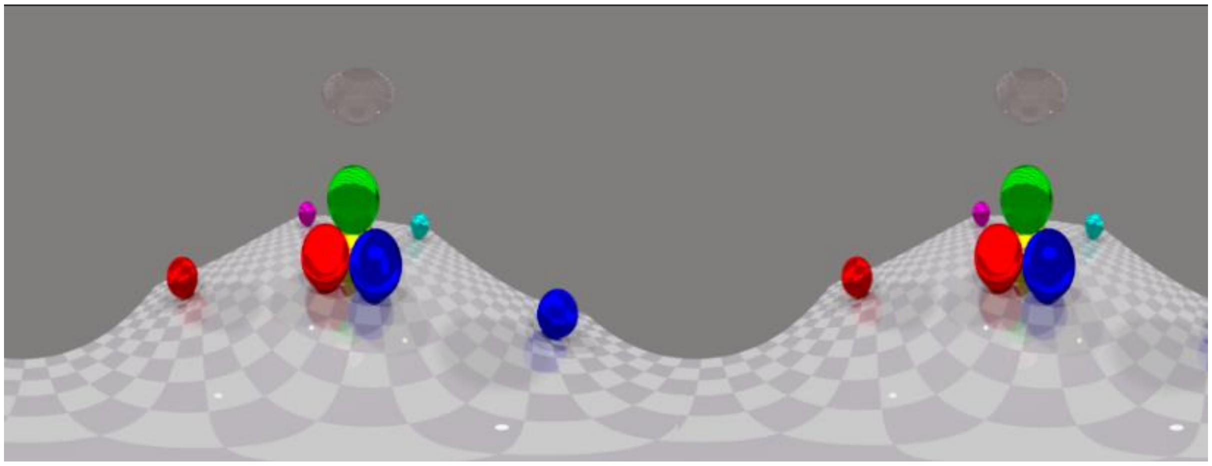


Fig. 9. Rendering Time of a Highly Realistic Image.

IV. Discussion and Conclusions

The farms of distributed rendering for images of high realism and complexity will be very useful for the research so the need to have more cores is beneficial to improve the rendering time so this project can be improved in increasing the number of cores without the need to use more computers, as it has the case of [12] that uses GPU-based engines which has a greater number of cores in order to reduce time. In the rendering of the prototype, it can have imbalances in a certain number of nodes in Table II, they are appreciated five moments of imbalance in the render these are due to several factors of which it has the issue of communication between nodes, more pixels in a strip of rendering and processes outside the rendering therefore it is desirable that the participating nodes are not doing other work that weaken to have an effective result. With regard to the future work of this research, high-realism animations will be carried out with respect to aerospace engineering prototypes, image processing, Big Data techniques and other investigations related to the existing lines in the direction of the investigation. These works will be carried out on the basis of distributed programming using the blender software as a rendering engine for the unification of rendered images and can be seen as a highly realistic animation in a shorter time. In this paper, it demonstrates that rendering farms under a Beowulf cluster reduces the time for obtaining large and complex images. Therefore, it is a benefit to use them in projects that require a realistic project model proposed for the research direction of the Universidad de Ciencias y Humanidades.

References

[1] M. Z. Patoli, M. Gkion, A. Al-Barakati, W. Zhang, P. Newbury, and M. White, “An open source grid based render farm for Blender 3D,” 2009 IEEE/PES Power Syst. Conf. Expo. PSCE 2009, no. April 2014, 2009.

[2] E. L. Huamaní, P. Condori, and A. Roman-Gonzalez, “Implementation of a Beowulf Cluster and Analysis of its Performance in Applications with Parallel Programming,” Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 8, pp. 522–527, 2019.

[3] N. Seticaya, “Implementasi Rendering Farm dengan Teknologi Cluster Computing Menggunakan Back Burner di Laboratorium Multimedia,” vol. 7, no. 2, pp. 41–56, 2018.

[4] G. V. Patil and S. L. Deshpande, “Distributed rendering system for 3D animations with Blender,” 2016 IEEE Int. Conf. Adv. Electron. Commun. Comput. Technol. ICAECCT 2016, no. March, pp. 91–98, 2017.

[5] A. Sheharyar and O. Bouhali, “A Framework for Creating a Distributed Rendering Environment on the Compute Clusters,” Int. J. Adv. Comput. Sci. Appl., vol. 4, no. 6, pp. 117–123, 2013.

[6] K. Cho et al., “Render Verse: Hybrid Render Farm for Cluster and Cloud Environments,” Proc. - 7th Int. Conf. Control Autom. CA 2014, pp. 6–11, 2014.

[7] L. Chuquiguanca, E. Malla, F. Ajila, and R. Guamán-quinché, “Arquitectura clúster de alto rendimiento utilizando herramientas de software libre,” vol. 2, no. 1, pp. 1–8, 2015.

[8] J. P. Naiman, “AstroBlend: An astrophysical visualization package for Blender,” Astron. Comput., vol. 15, pp. 50–60, 2016.

[9] A. Asadulina, M. Conzelmann, E. A. Williams, A. Panzera, and G. Jékely, “Object-based representation and analysis of light and electron microscopic volume data using Blender,” BMC Bioinformatics, vol. 16, no. 1, pp. 1–9, 2015.

[10] J. Petke, M. Harman, W. B. Langdon, and W. Weimer, “Specialising Software for Different Downstream Applications Using Genetic Improvement and Code Transplantation,” IEEE Trans. Softw. Eng., vol. 44, no. 6, pp. 574–594, 2018.

[11] E. L. Huamaní, P. Condori, and A. Roman-gonzalez, “Virtualizing a Cluster to Optimize the Problems of High Scientific Complexity within an Organization,” vol. 10, no. 6, pp. 618–622, 2019.

[12] A. Martos and B. Ruiz, “Realistic virtual reproductions. Image-based modelling of geometry and appearance,” Proc. Digit. 2013 - Fed. 19th Int’l VSMM, 10th Eurographics GCH, 2nd UNESCO Mem. World Conf. Plus Spec. Sess. fromCAA, Arqueol. 2.0 al., vol. 1, pp. 127–134, 2013.