Distributed System Load Visualization*

Martín L. Larrea
Sergio R. Martig
Silvia M. Castro
Departamento de Ciencias e Ingeniería de la Computación
VyGLab
Laboratorio de Investigación en Visualización y Computación Gráfica
Universidad Nacional del Sur
Avenida Alem 1253
Argentina, CP 8000, Bahía Blanca, Buenos Aires
{mll, srm, smc}@cs.uns.edu.ar

Abstract

In this article we show how the interface design for the visualization of distributed system load can benefit from the combination of concepts and techniques from Information Visualization and Human Computer Interaction (HCI). The use of visual representations and interactions to accelerate the insight into complex data is what distinguishes visual analytic software form other types of analytical tools. Visual representations translate data into a visible form that highlights important data features, including commonalities and anomalies. These visual representations helps the users to quickly perceive salient aspects of their data. Every distributed systems administrator must handle a high volume of information and the exploration and analysis of this data is becoming increasingly difficult. We propose an entirely novel approach to visualize the parameters involved in the system load for a distributed system to obtain an effective visualization tool in order to reduce the user cognitive workload and help the user to make the right decisions in a productive way.

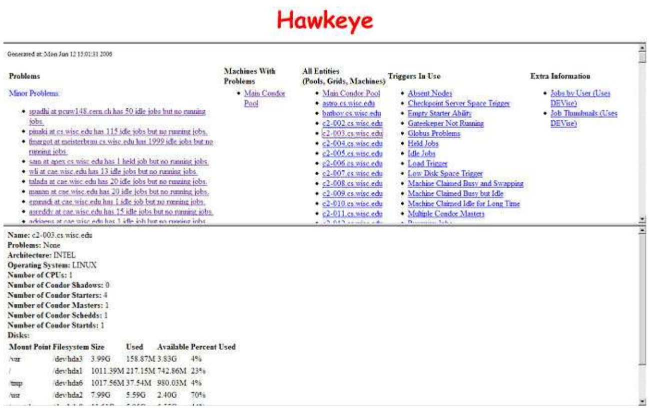
Keywords: Distributed Systems, Human-Computer Interaction, Information Visualization, Focus + Context Visualization

1 INTRODUCTION

A process is an operating system abstraction representing an instance of a running computer program. Process migration refers to the process transference between two machines during its execution that enables dynamic load distribution, fault resilience, eased system administration, and data access locality. Despite these goals and ongoing research efforts, migration has not achieved widespread use. With the increasing deployment of distributed system in general, and distributed operating systems in particular, process migration is again receiving more attention in both research and product development. Distributed system typically operates under continuously changing conditions and load balancing is critically important for efficient utilization of their resources since it maximizes their performance, and minimizes the process response time. The average process response time is usually considered the most important value for measuring the actual performance of a multitasking system. Due to the shift in the high performance facilities from supercomputers to workstations network and

*This work was partially supported by the Universidad Nacional del Sur.

the ever increasing role of the World Wide Web, distributed system and process migration will play a more importan role and will be eventually widely adopted. In many systems, the state of each node and process is distributed among a number of tables in the system, making it hard to extract the information about its behavior and load. This forces the user to explore large volumes of information using unnatural representations which may lead to misinterpretations and wrong decisions. Several systems exists today that can collect all this information for the user, but the final representation is made in a spreadsheet style which may force the user to increase his/her mental workload in order to understand the information. The Condor Project [12] is a project that supports High Throughput Computing on large collections of distributive computing resources. Its monitoring tool is Hawkeye which collect information from all computing resources and presents it to the user in a spreadsheet style as we can see in figure 1. All this information can be rendered in entirely novel way, e.g. by graphical visualization; this is the main topic of this work. The problem of exploring large datasets has been studied



Generated at: Mon Jun 12 15:01:31 2006

Problems

Minor Problems

- spadhi at pcuw148.cern.ch has 50 idle jobs but no running jobs.
- pinaki at cs.wisc.edu has 115 idle jobs but no running jobs.
- fmargot at meisterbrau.cs.wisc.edu has 1999 idle jobs but no running jobs.
- sam at apex.cs.wisc.edu has 1 held job but no running jobs.
- wli at cae.wisc.edu has 13 idle jobs but no running jobs.
- talnda at cae.wisc.edu has 20 idle jobs but no running jobs.
- manan at cae.wisc.edu has 20 idle jobs but no running jobs.
- epuradi at cae.wisc.edu has 1 idle job but no running jobs.
- asreddy at cae.wisc.edu has 15 idle jobs but no running jobs.
- adrianus at cae.wisc.edu has 1 idle job but no running jobs.

Machines With Problems

- Main Condor Pool

All Entities (Pools, Grids, Machines)

- Main Condor Pool
- astro.cs.wisc.edu
- batboy.cs.wisc.edu
- c2-002.cs.wisc.edu
- c2-003.cs.wisc.edu
- c2-004.cs.wisc.edu
- c2-005.cs.wisc.edu
- c2-006.cs.wisc.edu
- c2-007.cs.wisc.edu
- c2-008.cs.wisc.edu
- c2-009.cs.wisc.edu
- c2-010.cs.wisc.edu
- c2-011.cs.wisc.edu

Triggers In Use

- Absent Nodes
- Checkpoint Server Space Trigger
- Empty Starter Ability
- Gatekeeper Not Running
- Globus Problems
- Held Jobs
- Idle Jobs
- Load Trigger
- Low Disk Space Trigger
- Machine Claimed Busy and Swapping
- Machine Claimed Busy but Idle
- Machine Claimed Idle for Long Time
- Multiple Condor Masters

Extra Information

- Jobs by User (Uses DEVise)
- Job Thumbnails (Uses DEVise)

Name: c2-003.cs.wisc.edu
Problems: None
Architecture: INTEL
Operating System: LINUX
Number of CPUs: 1
Number of Condor Shadows: 0
Number of Condor Starters: 4
Number of Condor Masters: 1
Number of Condor Schedds: 1
Number of Condor Startds: 1
Disks:

Mount Point	Filesystem	Size	Used	Available	Percent Used
/var	/dev/hda3	3.99G	158.87M	3.83G	4%
/	/dev/hda1	1011.39M	217.15M	742.86M	23%
/tmp	/dev/hda6	1017.56M	37.54M	980.03M	4%
/usr	/dev/hda2	7.99G	5.59G	2.40G	70%

Figure 1: A screenshot from Hawkeye monitoring tool.

over the last decade in two major areas: Information Visualization ([3], [8]) and Human Computer Interaction ([2], [11]). Both areas have presented solutions to similar problems in different contexts. In this paper we show how the design of the interface for the visualization of distributed system load can benefit from the combination of Information Visualization and Human Computer Interaction (HCI) concepts and techniques. Visual representations translate data into a visible form that highlights important data features, including commonalities and anomalies. These visual representations make it easy for users to perceive salient aspects of their data very fast. The visualization of the distributed system load will help the user to make decisions and allow him/her a better and quick comparison among the states of the nodes involved in the distributed system. This paper continues the work done by [6] and [4], presenting a new visualization technique and adding the system topology as a new visual parameter. This work was developed at the Dpto. de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur, Bahía Blanca, as a interdisciplinary research between the Laboratorio de Investigación en Visualización y Computación Gráfica (VyGLab) and the Laboratorio de Investigación en Sistemas Distribuidos (LISiDi). This paper is organized as follows: in the next section we present an introduction to the problem of distributed system load balance and the limitations considered for the visualization. Then we review the distributed system parameters that will be considered for the visual representation and afterwards the visual elements are presented and described. We also present an analysis of visualizations techniques and their application in this context. We then apply this result to create an effective visualization tool that reduces the user cognitive workload and help the user decision making process. This last section naturally leads to an evaluation of the visualization in section 4. An overview of the work and some considerations for future work conclude this paper.

2 SCOPE OF THIS APPROACH

The goal of this paper is to present the visualization of a distributed system load according to the parameters that will be established in the next section. In what follows we consider a heterogeneous distributed systems with *N* interconnected nodes. Each node has an identification string, a memory size, a CPU Usage value and the size of the currently memory in use. On each node we have *M* processes, $M > 1$, being executed. We assume that the visualization system will not make decisions nor it will propose them; the user will play that role based on the information presented by the visualization. The system will limit itself to offer all the necessary visual aids to help the user.

3 VISUALIZATION PARAMETERS

All visualizations presented in [6] had one thing in common, the use of 2D graphics. In this work our visualization is base on a 3D representation. This extra dimension will give us more space to work with. But gaining more space is not the only advantage of using 3D. Because of general human familiarity with 3D physical world, 3D lends itself to the creation of real world metaphors that should help in perceiving complex data structures.
The system parameters to be visualized can be divided into three groups.

1. Node base parameters
   (a) Memory Size
   (b) CPU Clock Speed
   (c) CPU Usage
   (d) Total Memory in Use
2. Processes base parameters
   (a) Memory size for each process
   (b) Number of files opens
   (c) Number of messages sent
   (d) Number of messages received
   (e) Volume of information sent
   (f) Volume of information received
3. Distributed system topology

In the next section, we develop all the present aspects in the visualizations in detail, analyzing the different parameters and their visual representation. Finally, the visual elements are integrated to allow a view of the load state of the the distributed system.

3.1 Node Base Parameters

3.1.1 Memory Size

The graphical representation of the memory size will be created using a square prism. The height of the prism is a constant value and the width of the base depends on the memory size. In order to easily compare different nodes, all widths are normalized according to the maximum memory size present in the system. Figure 2 shows the visual representation for a node with 256 MB and another one with 128 MB. The user can see that the left prism represents approximately the double of the memory that the one on the right.

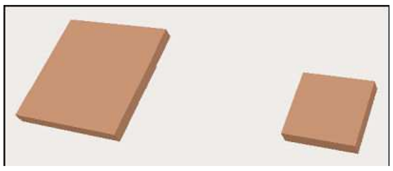


Figure 2: The left prism shows a double memory size that the one on the right.

3.1.2 CPU Clock Speed

The visual representation for the CPU clock speed will be also created by using a square prism. In this case, the height represents the clock speed. As in the memory size case, all prisms are normalized according to the maximum CPU clock speed present in the system. Figure 3 shows the visual representations for two CPUs.

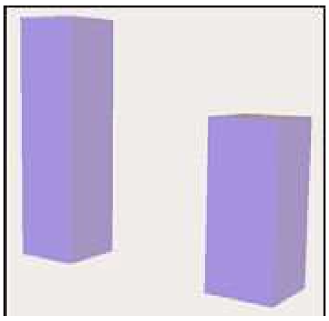


Figure 3: The visual representation on the left shows a CPU clock speed higher than the right one.

3.1.3 CPU Usage

Combining the visual representation of the CPU clock speed with a new opaque square prism, figure 4 we will create a new visual indicator. The CPU clock speed prism will have a high degree of transparency, and the height of the new prism will depend on the CPU percentage of use as showed in figure 4 and figure 5.
When the opaque prism reaches the height of the transparent one, it means that the CPU usage is 100%; when the opaque prism is not visible the CPU usage is around to 0%. Any intermediate height level of the opaque prism will be according to the currently CPU Usage percentage. (Figure 5.)
Figure 6 shows how the combination of all three visual representations will create the basic icon for a node in the distributed system.

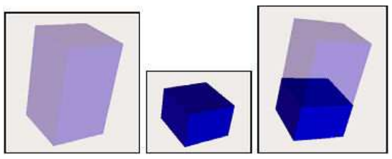


Figure 4: The composition of the two prisms allows us to represent two parameters, the CPU clock speed and CPU usage

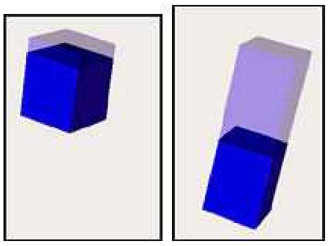


Figure 5: The CPU on the left has a CPU usage close to 100% while the one on the right is close to 50%. However the one on the right double the CPU clock speed of the left one.

3.1.4 Total Memory in Use

Finally, we show how to visualize the total memory in use. In order to do this, we will use the base size of the two prisms that conforms the CPU Clock Speed and the CPU usage. The base of this two elements will increase or decrease according to the memory in use. The memory size indicator represents the total real memory that a process can use.
However, when the total memory in use exceeds the real memory size and the virtual memory is used, there will be an occlusion on the memory size indicator; this can be observed in figure 8.
To avoid this situation the icon that represents the node will flip horizontally every time the total memory in use became larger than the real memory size.

3.2 Processes Base Parameters

All the visual elements described so far belong to node base parameters. However, there exist other parameters that the user may wish to visualize at some point. These are the process base parameters. Process base parameters include:

a. Memory size for each process

b. Number of files opens

c. Number of messages sent

d. Number of messages received

e. Volume of information sent

f. Volume of information received

To show all the information relative to the nodes and to each process will create a high volume of information that it will be impossible for the user not only to handle but also to understand. To solve

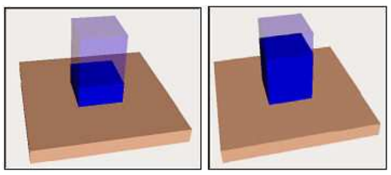


Figure 6: The visual representation of a node.

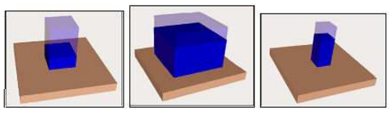


Figure 7: Different states of the memory in use indicator.

this problem, a technique used by both Information Visualization and HCI is semantic zoom. By means of the semantic zoom, the scale representation of an object is not just a graphical zoom but also a semantic one. We define three levels of semantic zoom, each one presenting more information than the previous one to the user. On the first level, each visual representation of a node shows the node identification. When the user positions his mouse over the node, the second level of semantic zoom is active; in this level the node identification is followed by a set of relevant data, previously defined by the user. Finally, the third level of semantic zoom is reached when the user clicks on a node. This level shows a list of all running processes on the node and the processes parameters values previously defined. This information is located on the faces of the memory size prism. To allow this, the height of the prism will grow according to the number of processes. These three levels are showed in figure 10.

3.3 Distributed System Topology

A system parameter that neither belongs to the node nor to the process is the system topology. Our focus is on visualizing not only the data associated with the nodes but also the structure of the network itself, that is, the system topology. In this case, the distributed system topology can be easily viewed as an undirected graph in a three dimensional space, figure 11, in a way that mimics the real network topology. Hence the navigation and exploration of the distributed system can be thought as navigation and exploration of a graph. With the increased use of large distributed system the navigation and exploration of these structures present a very important challenge. This challenge has been studied over the last decade within two major areas such as Information Visualization and Human Computer Interaction. Both areas have presented solutions to similar problems in different contexts. In the next subsections we outline an introduction to visualizations techniques and later we detail the technique used for the visualization of a distributed system load.

3.3.1 Visualization Techniques

One of the most important challenges in a visualization system is to present as much important information as possible in a given finite display area. When the structure of interest is too big to be viewed in detail all at once, the most straightforward solution is to allow the user to pan and zoom the visible area. The disadvantage of simply providing these interactions is that users often lose track of its current position with respect to the global structure. However, the addition of a smaller sec-

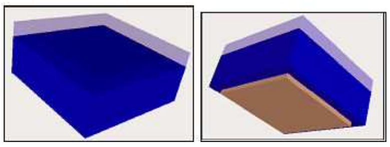


Figure 8: When the memory in use became larger that the real memory size, the two prism hide the total memory size indicator.

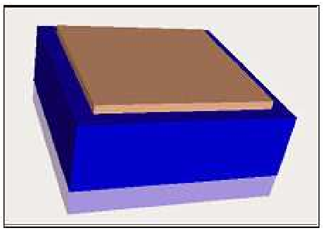


Figure 9: The visual representation for a node using virtual memory.

ondary window that shows a global overview with the current viewport location marked, can provide some guidance but forces the user to continually switch his/her focus of attention from one window to another, leading to disorientation. A large class of visualization techniques has been developed to address this problem by attempting to smoothly integrate detailed views with as much surrounding context as possible, so that users can see all relevant information in a single view ([5], [1], and [10]).

In order to interactively browse and manipulate a complex graph, an efficient visual interface is required. Such an interface combines a simple visual representation of it with a set of easy to use operations. However, as the amount of information and the complexity of a given graph increases, a traditional visual interface loses its appeal. Advanced visualization interfaces attempt to overcome the size limitation problems associated with conventional interfaces by exploiting new visualization techniques. A full review of a large group of techniques can be found in [5]. An effective visualization must allow the user to know intuitively what sector of the distributed system he/she is looking at. Because of this, it is import to maintain the context of the user location at any moment. All undistorted techniques present a lack of context which make them a poor selection for our objective. Among distorted techniques, those ones with a non continuous magnification fail to provide a smooth transition between the focus areas and the context areas and force the user to mentally create this transition, increasing the cognitive overhead.
Techniques with continuous magnification provide the best framework, see figure 12. In this paper we have chosen to use a Fisheye View applied to a graph for the distributed system model.

3.3.2 Fisheye View of a Graph

A Fisheye View of a graph [7] shows things near the center of view in high magnification and detail; at the same time, it shows the whole structure with decreasing magnification and less detail as we get further away from the center of the view. Thus, a Fisheye View seems to have all the advantages of the other approaches without suffering from any of their drawbacks. This concept is formalized in [9]. In figure 14(a) we can see a normal view of the distributed system and in figure 14(b) we show how the user make focus on the USS-LAB6 node. As we can see, the distributed system is affected

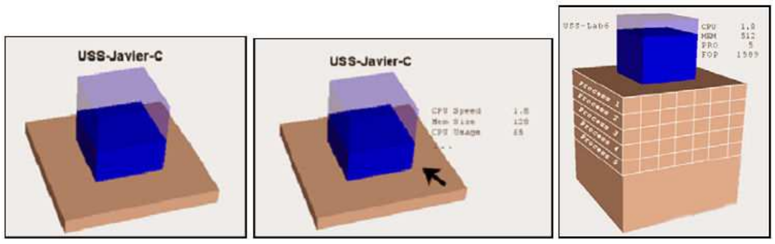


Figure 10: The three level of the semantic zoom.

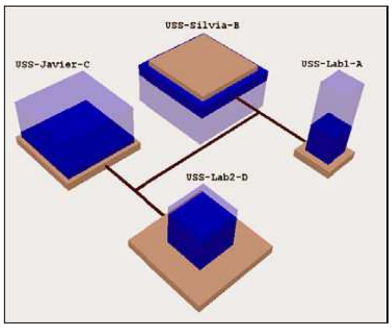


Figure 11: Distribute system composes of four nodes.

by the Fisheye distortion and the USS-Lab6 node shows all processes and their associate information.

4 USER EXPERIMENTS WITH VISUALIZATION

We now describe a comparative experiment between the visualization described so far and a spreadsheet type visualization. Subjects performed tasks relating to the state of a distributed system. Task completion times, correctness and user satisfaction were measured. Comparisons between information visualization systems can provide valuable information about the effectiveness and the ease of use of a system.

4.1 The Experiment

This is the first usability test of a series that will take place along the development of this tool. The aim of the experiment was to determine whether solving tasks in each visualization differs with respect to the task completion times, the accuracy and the user satisfaction. The null hypothesis was that our visualization improves completion time, accuracy and user satisfaction over the spreadsheet visualization. We decided to conduct this test under laboratory conditions using a revision base evaluation. The test consisted in the visualization of a distributed system composed by six computers with different characteristics. Three scenarios were created; on each of these a variation of some of the parameters was introduced. Users had to answer 4 questions in each of the three scenarios, for a total of 12 questions about the visualization of a distributed system. Questions and scenarios were generated and selected by the experimenters in an iterative brainstorming process based on whether or not they were interesting and would naturally occur in the analysis of the respective distributed system by a system administrator.
The following 4 questions were eventually selected:

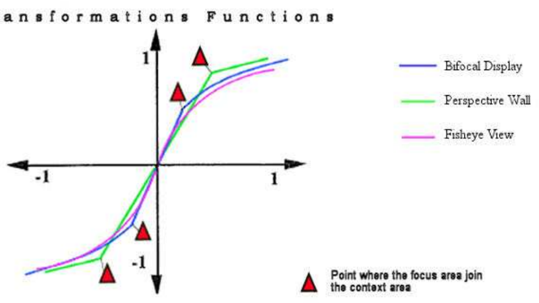
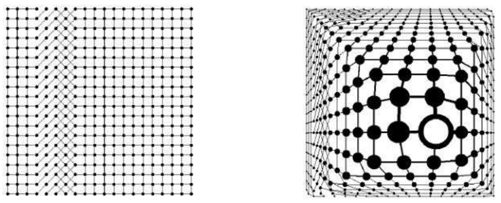


Figure 12: A comparison of three transformation functions and the transition between the context and focus area.



(a) Before the fish-eye view. Image courtesy of [9]

(b) After the fish-eye view. Image courtesy of [9]

Figure 13: The Fisheye view

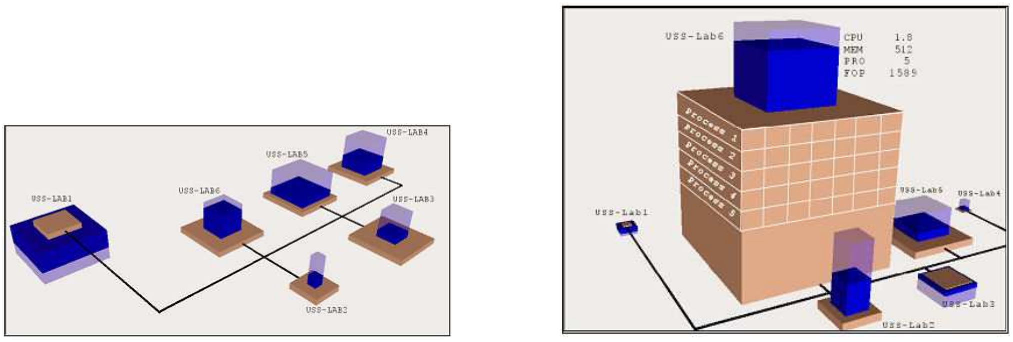
a. Which node is using virtual memory?

b. Which node has the most CPU Usage?

c. Which node has the most free memory?

d. Which node has the least CPU Usage?

14 subjects participated in the experiments. They were students with greater or smaller experiences in Information and Computer Sciences or Engineering who had at least five years of experience working with computers. Two teams were made, team A and team B. Team A evaluated our visualization while team B worked with the spreadsheet visualization. Both teams answered the same

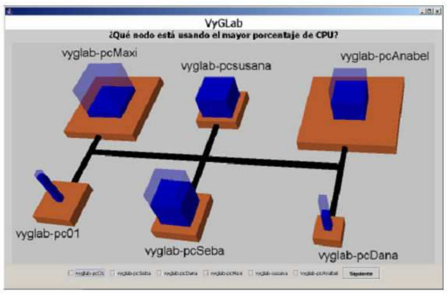


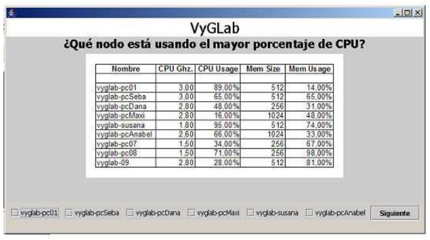
(a) Before the fisheye view. A normal view of the distributed system.

(b) After the fisheye view. Focus on USS-LAB6

Figure 14: Fisheye view in the visualization of distributed system load.

questions in the same scenarios, being the visualization the only difference. Each question was presented to the subjects on a computer screen; there was an image associated with the scenario and a timer to help the user self time. We developed a windows application that shows an image next to a question and the user must select the answers from a multiple choice. This application also keeps track of user response time. To collect the data the application creates a log file. Figure 15 shows screenshots of our application and the spreadsheet visualization. The experiment took place in one of our laboratories. Team A required 20 minutes of instruction on the visualization and Team B only 5 minutes. Subjects began the experiments in which they had to answer each one of the 12 questions. Each question was presented to the subject on a computer display next to an image and a checklist to select the possible answers. For Team A the image was a picture of our visualization system, for Team B it was a data table with the same information presented to Team A, but in a spreadsheet style.





VyGLab

¿Qué nodo está usando el mayor porcentaje de CPU?

Nombre	CPU Ghz.	CPU Usage	Mem Size	Mem Usage
vyglab-pc01	3,00	89,00%	512	14,00%
vyglab-pcSeba	3,00	65,00%	512	65,00%
vyglab-pcDana	2,80	48,00%	256	31,00%
vyglab-pcMaxi	2,80	16,00%	1024	48,00%
vyglab-susana	1,80	95,00%	512	74,00%
vyglab-pcAnabel	2,60	66,00%	1024	33,00%
vyglab-pc07	1,50	34,00%	256	67,00%
vyglab-pc08	1,50	71,00%	256	98,00%
vyglab-09	2,80	28,00%	512	81,00%

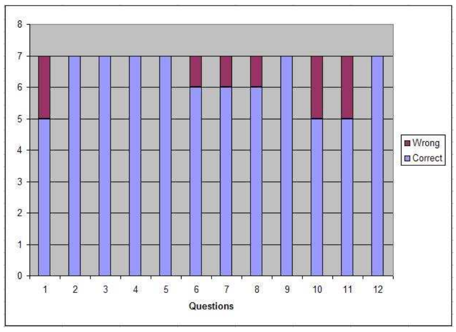
Figure 15: Two screenshots from our application.

In the quantitative analysis, the correctness of the user's task performance and their task completion time were measured based on the answers collected by the application. The user satisfaction data were taken from the final questionnaire.

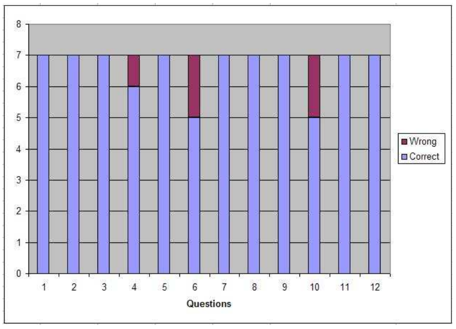
4.2 Quantitative Results

4.2.1 Correctness of Answers

Figure 16(a) and figure 16(b) shows the number of correct and incorrect answers for each visualization. The spreadsheet visualization yielded the highest number of correct answers, 94%, against our visualization, 89%. We consider this a relative small difference, especially since this is our first user evaluation on the visualization system.

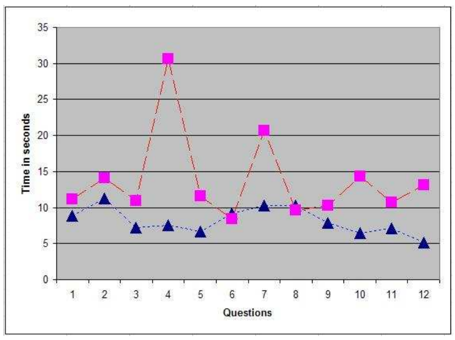


(a) Correctness of answers for our visualization system.

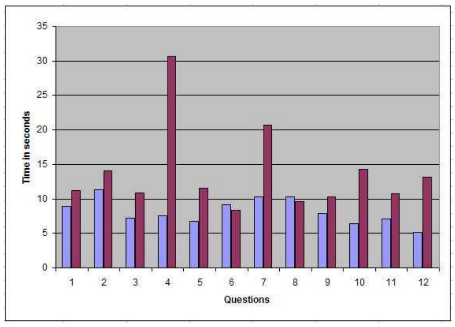


(b) Correctness of answers for the spreadsheet visualization.

Figure 16: Correctness of answers



(a) Average response times. The triangles represent our visualization and the squares the spreadsheet visualization.



(b) Average response times gaps. The light color represents our visualization and the dark one the spreadsheet

Figure 17: Response times between our visualization and the spreadsheet.

4.2.2 *Speed of Task Performance*

Figure 17(a) shows the average performance time for the 12 questions. Except for questions 6 and 8, our visualization system had the shortest task performance times with a finally average time of 7.75 seconds against the 11.4 seconds of the spreadsheet. Questions 6 and 8 are the only ones where the spreadsheet got better results; as figure 17(b) shows, the gaps between the spreadsheet and our visualization is relative small.

5 CONCLUSIONS AND FUTURE WORK

It is possible to transfer the results from Information Visualization and HCI on large data set exploration to the visualization of distributed system load. By doing this we applied techniques specially designed which have proved to be successful for these purposes and we obtained an intuitive view of the system load that will assist the user. We have developed some novel graphical icons for displaying distributed system load data together with $focus + context$ techniques that can help extract meaningfull insights from the data currently available. Our usability test showed a significant improve over the speed of task performances against the spreadsheet visualization. The test result will be fed back to the visualization to improve it and more test will be conducted, included field evaluations. We are extending our current prototype to include all processes information and all possible interactions with the visualization. New usability evaluations will be developed to evaluate all new features. We are now working on techniques that permit the graphs to continue to reveal relationships in the context of much more data and also on an extension of the view to make the visualization a way to control the system. In this context an important future goal is to allow more than one focus element at a time. We are also analyzing how the user will interact with the visualization to migrate a process and finally we are also evaluating the inclusion of SMTP, Symmetric Multiprocessing.

REFERENCES

[1] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman, editors. *Readings in information visualization: using vision to think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[2] Alan Dix, Janet Finlay, Gregory Abowd, and Russell Beale. *Human-computer interaction*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.

[3] Daniel A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8, 2002.

[4] Martig S. Castro S. Echaiz J. Larrea, M. A proposal from the point of view of information visualization and computer human interaction for the visualization of distributed system load. *Journal of Computer Science and Technology*, 5(4), 2005.

[5] Y. K. Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Trans. Comput.-Hum. Interact.*, 1(2):126–160, 1994.

[6] Trutner G. Vitturini M. Alvez C. Di Luca S. Castro S. Echaiz J. Martig, S. and J. Ardenghi. Visualización del balance de carga en un sistema distribuido. *Jornadas Chilenas de Computacin*, 2001.

[7] E. Noik. Layout-independent fisheye views of nested graphs. *Proceedings IEEE/CS Symposium on Visual Languages*, pages 336–341.

[8] Donna L. Gresh Robert Kosara, Helwig Hauser. An interaction view on information visualization. *State-of-the-Art Proceedings of EUROGRAPHICS 2003 (EG 2003)*, pages 123–137, 2003.

[9] Manojit Sarkar and Marc H. Brown. Graphical fisheye views of graphs. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 83–91, New York, NY, USA, 1992. ACM Press.

[10] Manojit Sarkar and Marc H. Brown. Graphical fisheye views. *Commun. ACM*, 37(12):73–83, 1994.

[11] Ben Shneiderman and Catherine Plaisant. *Designing the User Interface: Strategies for Effective Human-Computer Interaction (4th Edition)*. Pearson Addison Wesley, 2004.

[12] Douglas Thain, Todd Tannenbaum, and Miron Livny. Distributed computing in practice: the condor experience. *Concurrency - Practice and Experience*, 17(2-4):323–356, 2005.