2017 International Nuclear Atlantic Conference - INAC 2017
Belo Horizonte, MG, Brazil, October 22-27, 2017
ASSOCIAÇÃO BRASILEIRA DE ENERGIA NUCLEAR - ABEN

Agência Brasileira do ISBN
ISBN 978-85-99141-07-6
9 788599 141076

CLUSTER PROCESSING BUSINESS LEVEL MONITOR

Francisco J. Muniz

Centro de Desenvolvimento da Tecnologia Nuclear - CDTN,
31270-901 - Belo Horizonte - MG, Brazil,
muniz@cdtn.br

ABSTRACT

This article describes a Cluster Processing Monitor. Several applications with this functionality can be freely found doing a search in the Google machine. However, those applications may offer more features that are needed on the Processing Monitor being proposed. Therefore, making the monitor output evaluation difficult to be understood by the user, at-a-glance. In addition, such monitors may add unnecessary processing cost to the Cluster. For these reasons, a completely new Cluster Processing Monitor module was designed and implemented. In the CDTN, Clusters are broadly used, mainly, in deterministic methods (CFD) and non-deterministic methods (Monte Carlo).

keywords: Dynamic Load-balancing, Processing Monitor, Cluster.

1. INTRODUCTION

The main core of a dynamic load-balancing system has been subject of investigation for quite a long time [1]. A graphic description of a dynamic load-balancing approach (the EG mechanism) can be found [2]. Two other modules are needed to compose a dynamic load-balancing system: (1) An Administration Module (to be developed); (2) And a Cluster Processing Level Monitor Module, which was designed, implemented and is described in this article - Cluster means a set of PCs (IBM like computers) linked up by a communication device. The issue being investigated here, the Cluster Processing Monitor, is needed to satisfy the users on what concerns their expectations about the Cluster processing occupation level, so they can make better decisions. Considering the friendly appearance that the processing monitor module should have, since it has to be easily observed by the users, it was decided to represent the Cluster Processing Monitor graphically, using, basically, two sets of bars: (1) one stack of horizontal bars, with one bar for each processor to be monitored, showing the processing activity of the respective processor; and (2) a vertical scroll bar, that allows the user to select the set of processors to be monitored, i.e., allows to select (and identify) a window of processors, for whom the processor's activities should be displayed on the stack of horizontal bars. The identification of the horizontal bars were done using the corresponding numbers of the processor on the Cluster, so that, the user could know which processors would be monitored.

2. THE IMPLEMENTATION

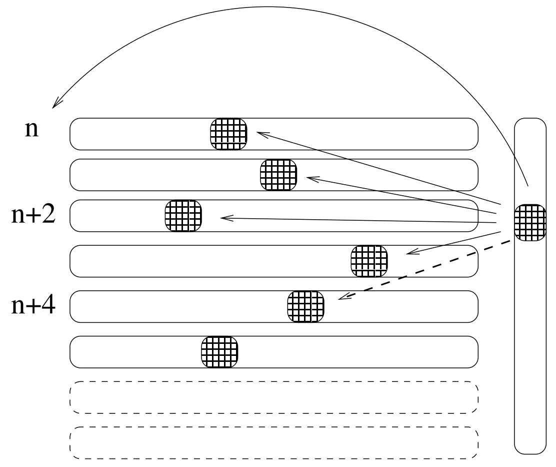
The Qt package [3] was chosen because of its graphical facilities. The Qt Software includes object oriented programming language facilities (Qt uses standard C++), in addition to a graphic library enough to attend the requirements of the application being described. The Qt software, also, provides several tools, among them a graphic editor (the "Qt Designer") that was used to instantiate, graphically, the basics objects needed in the implementation, i.e., objects from the QProgressBar class to represent the stack of horizontal bars, a object from the QScrollBar class to represent the vertical scroll bar and objects from the QLCDNumber class to indicate the processors that were being monitored. The Qt Designer tool allows to save the core code of the objects instantiated, on a text way. Then, later on, the code can be re-edited and patched (expanded) with new features.

Two other methods had to be added to the class QProgressBar. One that is executed during the objects initialization to create a table of commands - a command by line. Such command line is shown below:

```
ssh <node_name> /usr/bin/top -d 0.2 -n 2 -b | grep Cpu | \
    tail -n 1 | awk -F: '{print $2}' | awk -F, '{print $1}'.
```

A file with the processors names, to be monitored, have to be provided as input to the method that is being described. And a second method, that when called will execute selectively a command line of the referred table, which aims to update the occupation level of the processor on the horizontal progress bar, respectively. Variables, such as the maximum number of processors that should be shown on the screen, are established at compilation time. The Cluster Processing Monitor module was running on the Linux Ubuntu (14.4). The trust mechanism needs to be, previously, set on the machine.

In the Qt Software, the communication among objects are done using the connect object, i.e., a mechanism that uses signals and slots. Once established a link between objects (which is done through of the object connect), every time that an event (signal) happens in one object, a method (slot) is triggered to be executed in another object, such as: QObject::connect(object1, signal1, object2, slot1). This mechanism is, also, represented in Figure 1.



Everything is commanded by the slip button of the vertical scroll bar, which allows to select the set of processors to be monitored: the numbers on the left are updated whenever the slip button moves. The stack of horizontal bar represent, respectively, the level of processing business of the each one of the selected processors: the update mechanism of the horizontal bars happens when the vertical scroll bar slip button is released.

Figure 1: Schematic representation of the Cluster Processing Monitor.

3. CONCLUSION

The implementation is working quite well. It has shown adequate for the proposed monitoring module, since it is easy to use and it makes the results easily visualized.

ACKNOWLEDGEMENTS

Financial support for the paper presentation were provided by both: the FAPEMIG ('Fundação de Amparo à Pesquisa do Estado de Minas Gerais') and the CNPq ('Conselho Nacional de Desenvolvimento Científico e Tecnológico') Foundations. I also would like to thank the colleague Vagner Oliveira for his suggestions and comments during the development of this article.

REFERENCES

1. Muniz, F.J., de Azevedo, C.V.G., Dalle, H.M.: Dynamic load-balancing implementation on MIMD machine (and results). *Int. Nuclear Atlantic Conf. - INAC.* Recife, PE, Brazil, Nov. 24-29 (2013), ISBN: 978-85-99141-05-2
2. Muniz, F.J.: Dynamic Load-balancing - extended gradient mechanism - graphic representation. *Int. Nuclear Atlantic Conf. - INAC.* Belo Horizonte, MG, Brazil, Oct. 22-27 (2017), ISBN: 978-85-99141-07-6 (submitted).
3. http://qt-project.org (Wed Aug 13 13:59:10 BRT 2014)