Implementation of an email-based alert system for large-scale system resources

Robert Poenaru[1,2]
[1]*Horia Hulubei* National Institute of Nuclear Physics and Engineering, Magurele, Romania
[2]Doctoral School of Physics, University of Bucharest, Romania
robert.poenaru@nipne.ro

***Abstract*—Tackling the current problems of interest for physicists that deal with various topics require lots of computing simulations. Identifying and preventing any unusual behavior within the system resources that execute large-scale calculations is a crucial process when dealing with system administration, since it can improve the run-time performance of the resources themselves and also help the physicists by obtaining the required results faster. In the present work, a simple *pythonic* implementation which 1) monitors a given computing architecture (i.e., its system resources such as CPU and Memory usage), and 2) alerts a custom team of administrators via e-mail in almost real-time when certain thresholds are passed, is presented. Using existing packages written in Python, with the current implementation it is possible to send e-mails to a predefined list of clients containing detailed information about any machine running outside the "normal" parameters.**

***Index Terms*—python, system resources, alerting, email, smtp, monitoring, watchdog**

I. Introduction

The computing resources within a physics department must be up to speed and ready for a continuous run-time of small-, medium-, but also large-scale simulations, in order to assure a consistent and optimal workflow for the research teams that require calculations. Usually, there is a cohesive workflow between the scientists that want to run their simulations and the system administration (sysadmins) team that provides the necessary resources for executing them. The sysadmins must check that the resources which are performing calculations behave well, but they must also take care of the computing equipment that is sitting in *idle*-mode, in case new requests for allocating resources are issued by scientists. The process of resource management is crucial since it involves many factors in deciding which are the most optimal compute nodes that could start a new job, in terms of efficiency and speedup [1], these being deciding factors in the total run-time of the simulations themselves. Proper job allocation, management, and execution will result in minimal impact of resource slowdown, process blocking, or even dead-locks in the executing pipeline, giving the possibility of the researchers to obtain the desired numerical results in as fast as possible. On the other hand, any code optimization [2], [3] on the submitted simulations (done exclusively by the scientific teams) will also take advantage of the allocated resources, since *better code* implies faster execution, lower impact on the memory pool and much lower probability of program interruption. One can conclude that indeed, better resource management (done by the sysadmins) will result in better code execution, helping thus the scientists, but in the same way, any code optimization made by scientists will help the sysadmins, since the degree of failure within the executing simulations stack could be decreased. This reciprocal-mode of improvements (for both *communities*) is sketched in Fig. 1, where the key characteristics of both *communities* (i.e., physicists issuing simulations and sysadmins dealing with computing management) are emphasized. The arrows signify the *reciprocal improvement cycle* between the two.

However, due to the large degree of complexity of the underlying computing infrastructure, issues related to memory bandwidth, network stability, CPU throttling [4], cache availability [5] and so on are highly probable, especially when the machines are running continuously. Frequent updates, unexpected network traffic, errors within the services running in the application layer [6] could also affect the idling nodes. While the former issues will only affect the simulations that are currently being executed, the latter set of issues could produce unexpected delays in the starting process of the job queue [7], which will increase the wait-time of simulation results.

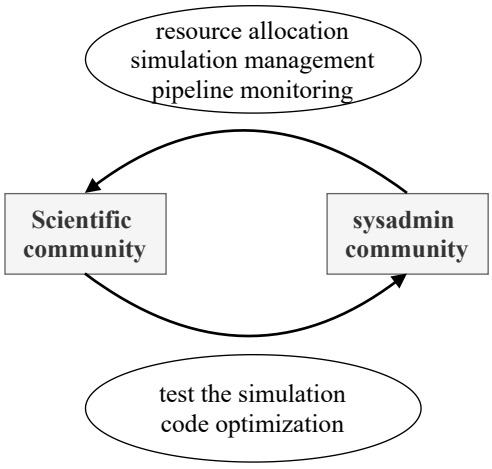

Fig. 1. A basic relationship between the scientific community that issues simulations to be executed on the computing resources, and the system administration team that deals with process allocation, execution and management of resources.

ACKNOWLEDGMENT

This work was possible through the Department of Computational Physics and Information Technology at *Horia Hulubei* National Institute of Physics and Nuclear Engineering.

REFERENCES

[1] Ashkan Paya. Resource management in large-scale systems. *Department Of Computing Science Umea University*, 2015.

[2] PVS Studio. Code optimization, 2017. Last accessed 24 September 2021, https://pvs-studio.com/en/blog/terms/0084/.

[3] Krishna Sai. Code optimization, 2018. Last accessed 24 September 2021, https://www.techtud.com/.

[4] IBM Corporation. Cpu throttling, 2012. Last accessed 24 September 2021, https://www.ibm.com/docs/en.

[5] Caching challenges and strategies. Last accessed 24 September 2021, https://aws.amazon.com/builders-library/.

[6] Pstree - linux man page. Last accessed 24 September 2021, https://linux.die.net/man/1/pstree.

[7] IBM Corporation. Job queues, 2015. Last accessed 24 September 2021, https://www.ibm.com/docs/en.