HPC Cluster Management with Open Source Software

Fernando G. Tinetti[1], **Leopoldo J. Rios**[2]
[1]Fac. de Informática, UNLP, Comisión de Inv. Científicas Prov. Bs. As., La Plata, Argentina
[2]Fac. Ciencias Exactas, UNNE, IMIT-Conicet-UNNE, Corrientes, Argentina

Abstract - *Scientific institutions and laboratories have incorporated and maintain a significant amount of computing resources over the years. Research centers may have unused hardware facilities, with significant processing throughput that could be made available to other institutions. In this paper, we propose a software infrastructure built on existing and new open source software for sharing its computing infrastructure, in a programmed, monitored, and secure way. The Role Based Access Control (RBAC) model is considered a mature and flexible technology, and is a very popular paradigm today. It offers a more secure alternative to the all-or-nothing superuser model. With RBAC, it is possible to apply a specific security policy, and it means that a user has the necessary amount of privileges to perform a task. Taking advantage of the known features of RBAC, it is proposed to add specific HPC monitoring functionality. The HPC monitoring facilities are intended to aid users of scientific programs in their optimization and parallelization tasks of processing intensive jobs.*

Keywords: High Performance Computing, RBAC, Open Source Software, RRDtool, Ganglia.

1 Introduction

In general purpose operating systems such as Linux, it is possible to deploy RBAC technologies to program superuser capabilities in *profiles* or *rights profiles*. These rights profiles are assigned to special user accounts called roles. Then, a user can assume a role to perform a job that requires some of the superuser capabilities. It is possible to configure different predefined rights profiles for different accounts. Rights profiles can provide flexible capabilities for task execution. When creating user roles, it is possible to assign them to some rather wide capabilities range (like that of a superuser), or more restricted capabilities. Specific user permissions are represented as the union of permissions grants that are assigned to user roles [1].

RBAC technology allows to address the possibility of sharing part of the computing infrastructure of an institution. We propose to define a percentage of the equipment to be used by external users, and managed by the software model we present. This software will allow the online monitoring of the occupation of the hardware resources, users, and software in execution. Fig. 1 schematically shows the idea of sharing an HPC cluster with external users, through the implementation of RBAC policies. The possible configurations are multiple and are adapted to each organization.

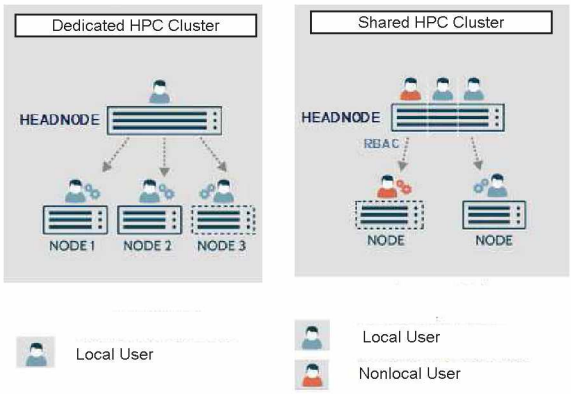

Figure 1: Dedicated and Shared Clusters.

We use a set of definitions/terms in our proposal, which we include here in order to avoid confusions in the rest of the paper:

- HPC cluster: hardware and software infrastructure that allows the concurrent and distributed execution of multiple user program/s in a controlled and monitored environment. A HPC cluster is focused on improving an application performance over and above that provided by a single computer [2].
- Local institution: research/laboratory center having a HPC cluster, which allows nonlocal users to use a fraction of the cluster facilities.
- Local users: local institution researchers and fellows who are granted access to the HPC cluster (computing, storage, etc.) resources.
- Nonlocal users: Researchers and fellows from outside the local institution, who are granted remote controlled access to the HPC cluster (computing, storage, etc.) resources.

We have identified, used, and combined a set of open source software packages [3] with different specific functionalities:

- Operating system (Linux) along with its standard profiling tools (gprof, perf, etc.).

- Batch/job queues processing.
- Job and resources usage monitoring, along with statistics reports.
- Web programming interfaces/web access.
- Database managers.

The first main activity is to program a web interface/access to interact with the nonlocal users of the cluster. Those users should be allowed to edit scripts and execute them, to compile their user programs, and to observe the performance that they obtain. A database will persist every user configuration, as well as the RBAC logistics with user data, access permissions, access times, and permission agreed with each one. The authorities of the local institution will have online information on the actual use of the infrastructure they are sharing, and control of the system for new defining usage policies.

2 Open Source Tools Environment

Our proposal has been implemented using several specific well known software packages, most of them already used (independently of each other) in many HPC clusters:

- 64 bits Linux operating system.
- Torque PBS [4].
- Ganglia [5] combined with RRDtool [6].
- LAMP (MySQL + PHP + Apache) [7].
- CakePHP Framework [8].
- Perf [9].

The Linux distribution is at the user's choice, and corresponds to conform to free distribution and open source standards. For this work, openSUSE has been chosen [10], both for the *headnode* (the cluster front-end node, as shown in Fig. 1), and for the computing nodes. However, it is possible to achieve the same result using any other distributions such as Debian, Ubuntu, or RedHat.

Torque PBS (supported on OpenPBS) is a computing resource manager software that provides control over batch jobs and distributed computing nodes. Torque is considered as open source software using the OpenPBS license, it has Fault Tolerance, Scalability, Scheduling, and Usability features. The essential function of this product is to organize the execution of calculation programs in work allowing a large number of users to run their jobs. An alternative to Torque PBS would be PBS Pro [11], which is a similar open source tool. The basic idea is that jobs run in a cluster - not to in specific nodes - and the queue manager assigns the work to node/s with sufficient resources (mostly RAM and CPU/s). Most queue managers (including OpenPBS) handle parallel and distributed computing programs, where distributed processes communicate through message passing mechanisms [12] [13]. Queue managers, such as OpenPBS naturally handle a cluster with many users, each one with specific job requirements such as RAM amount, sequential processing, shared memory parallel processing (e.g. with OpenMP threads), and shared memory parallel processing (e.g. with Message Passing Interface -MPI- processes). A configured queuing system ensures all users the execution of their programs, without any taking indefinite use of the nodes, and avoiding runtime resource contention.

Ganglia is a software tool for management of and access statistics of computational resources usage in HPC environments. It is deployed as a web solution, and allows to produce graphs generated on demand, about the actual (cluster wide) use of memory, CPU, and network of an HPC cluster. The proposal that we present includes Ganglia for the general, cluster wide system statistics. However, additional code is integrated to register statistics of use of elements not contemplated in Ganglia, such as the software used, and the distribution of work per user. Ganglia takes advantage of RRDtool for the management of time series data. We have already made some tuning of RRDtool for specific (and different from its default) statistical data sampling [14], later handled and shown in Ganglia web graphics. Cluster wide resource usage statistics/graphics is intended mainly for a role of local institution Director, but it is also possible to grant access to roles such as Researcher and Scholar to observe statistics of their own work.

The RBAC infrastructure programming uses a combination of MySQL [15] (a relational database manager or RDBMS), and CakePHP, a PHP framework widely used in the programming and computing environment. Users have access to the cluster via a web portal where they are handled according to the RBAC policies. Fig. 2 schematically shows the implementation of our RBAC strategy:

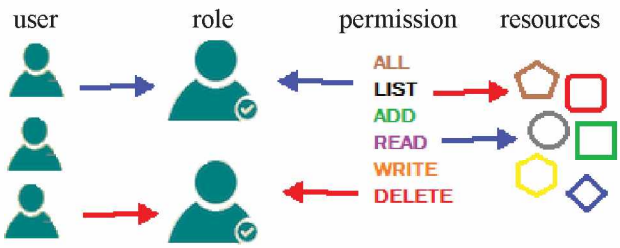

Figure 2: RBAC Strategy.

Authorizations, privileges, and commands with certain security attributes are assigned to profile roles. Roles are, in turn, assigned to users for access control. Our proposal does not make references to linux user accounts, instead, we handle our own (RBAC/web access) user accounts, created in the RBAC database. By default, when a new user is created, this user has no assigned roles, which means that the users does not have access to perform any task behind the RBAC.

Our proposal also includes tools to generate profiling statistics, which are related to but different from Ganglia resource usage reports. Profiling data aids in optimization

and parallelization, because it provides data for performance bottlenecks identification, for example. It is possible to produce summary statistics of hardware performance counter, thus aiding in performance evaluation and optimization [16]. Performance analysis based on profiling involves three steps:

- Instrumentation or modification of the program to generate performance data.
- Measurement/sampling of interesting aspects during execution that generate performance data.
- Analysis of runtime collected performance data.

Our proposal integrates scripts available to users, and allows the compilation of source code with specific parameters. The measurement is triggered with the execution of scripts, and the analysis is aiding via personalized monitoring options.

3 Implementation Details and System Functionality

We use a relational database (using MySQL as RDBMS, as explained in the previous section) for maintaining our RBAC system in order to define and enforce users, roles, permissions, etc. Fig. 3 shows our relational database (i.e. tables, keys, etc.), which is relatively simple and does not imply any performance problem.

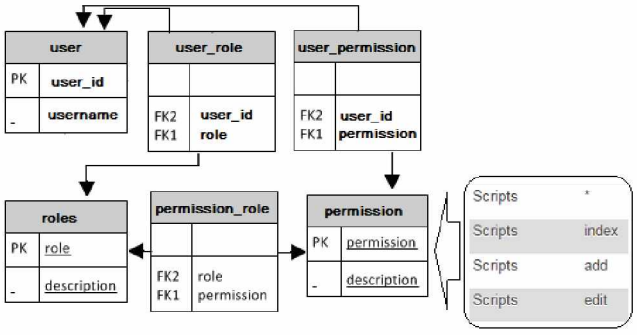

Figure 3: RBAC Database.

Given the relatively small number of users and data involved in permissions and access control (regarding *traditional* databases) the whole system could be implemented on top of files. We have chosen to use MySQL for several reasons:

- Relational databases provide a strong data model/ structure we take advantage of in this specific scenario for RBAC data organization and maintenance.
- Integration with other open source software for web portal implementation/s and web applications deployment.

Every user (disregarding whether a Local or Nonlocal user) interacts with the system through a login process with enabled credentials as shown in Fig. 4. The user 'admin' corresponds to the administrator system/account to manage the entire system, integrates a full menu of options (e.g. for adding users or changing permissions accesses), as shown in Fig. 5.

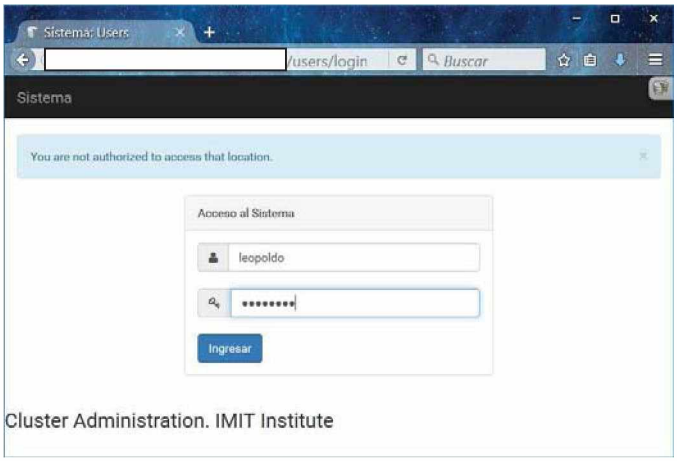

Figure 4: System Login.

Figure 5: System Administrator Menu.

Roles: Role handling/management is an administrator functionality, and it is used to define and associate roles to system users. The roles defined in our proposal current implementation are: "Becario" (Fellow), "Investigador" (Researcher), "Director" (Director) and "Administrador" (Administrator). Each role assigns permissions to jobs that will be dispatched on resources. User accounts are defined by a name, and its assigned roles (a user account may have more than one role). The user can edit his account profile, as usual, but is not allowed to modify roles (other than the Administrator user). It is possible to program that a user with Fellow role does not have access to visualize cluster usage statistics (a Ganglia resource), if a Director role user so requires. The cluster usage statistics are allowed by default to Researcher and Director roles.

The application allows the system Administrator to assign specific permissions to defined roles. For example, the role "Fellow" is assigned "Scripts / *" permission, which implies access to list, add, and edit scripts of the database. Otherwise, if a role is assigned "Scripts / index" permission, users with this role will only have access to list the available

scripts. Fig. 6 shows that when selecting the "Becario" (Fellow) role it is possible to edit its permissions, initially viewing the current ones. Fig. 7 shows the role edition for this example, i.e. the system reads the current permissions stored in the database and lets the administrator to change them. Once the administrator selects "save", the changes are made permanent by storing them in the database.

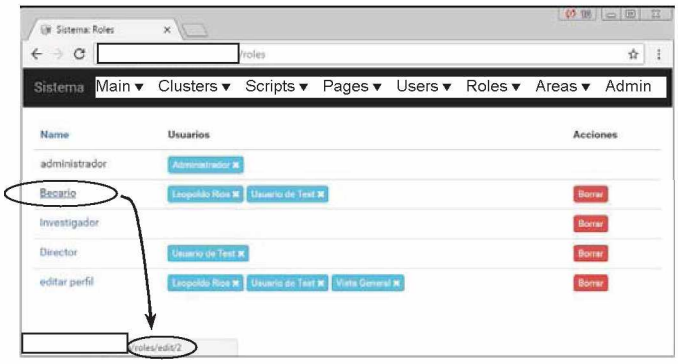

Figure 6: Users and Role Edit.

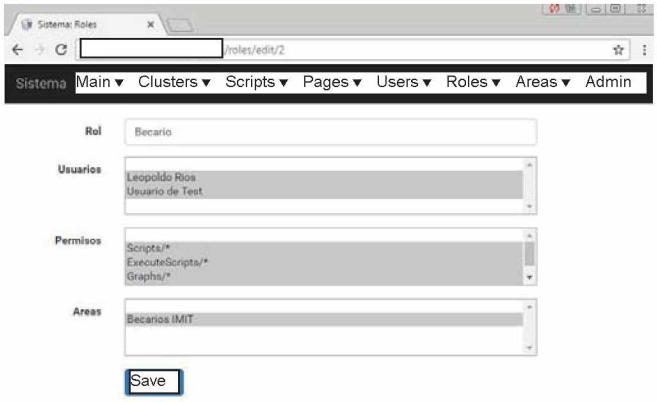

Figure 7: Specific "Becario" Role Edit.

Program execution: the actual usage of the cluster processing facilities is made via the queue manager (Torque PBS, pbs_server application). The user should select some of the available script models and configure the application name, input data, and name of the output file, along with the required CPU and RAM resources. The queue manager, depending on the availability of the resources invoked by the user will determine when to start the execution. During execution, the system reports the status of jobs to users, using monitoring scripts. Once the job is finished, the resulting output files will be stored in temporary folders available to the user. A typical user command line could be:

$ ./user_app input1.dat > stat-out.log

At the end of the execution, the user can visualize their results at the tarminal, or transfer the output file through internet services.

Performance evaluation: Initially, we expect to use the basic profiling tools: gprof, perf, and OProfile [17] [9] [18]. They allow post-mortem analysis. i.e. once the program has ended its runtime it will be possible to analyze several performance indexes (runtime call graph, CPU usage, cache related events, bottlenecks, etc.). The system choses the node on which to run the program along with the specific performance evaluation tool. A user command line, with perf could be:

$ perf stat -e cache-misses ./matmul 1000

which, once completed its runtime in the selected node, will provide the information similar to that shown in Fig. 8. Performance data could be stored in a database, so that the user would be able to analyze not only individual experiments and optimization results but also runtime trends for specific applications and input data sets.

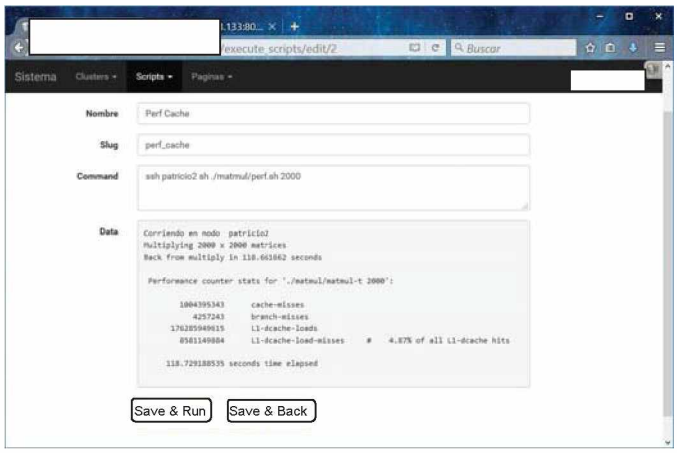

Figure 8: Performance Evaluation Data (perf).

Cluster Sharing and Resource Limits: given that we expect to be able to share a fraction of the HPC cluster resources, we should provide access to nonlocal users. Thus, we should be able to handle security and resource usage details, so that:

- Execution control should be provided to the administrator user/s, for defining application runtime limits, along with computational resources distributed with fairness in amount and usage time.
- External access to the cluster must be assured only to the equipment configured for this purpose. The resto of the hardware should be out of reach of nonlocal users.
- Controlled software installation should be handled by the administrator user/s. The download and compilation of software in the cluster installations must be done or assisted by the cluster administrator, to ensure that their parameterization and configuration does not alter the cluster management and operation.
- User security and control provided basically via the RBAC system, so that each user is isolated and protected, e.g. operations and results are not disclosed and beyond the reach of other users. This requires a detailed programming in the

access controls to files and folders, aspect that is considered in the deployment of RBAC.

Specific handling must be implemented for sharing the cluster with nonlocal users. Software techniques such as RBAC for access control and management of external users have been implemented. The control of jobs in the hardware defined as shared is possible either with the job/queue manager already in production (as shown in Fig. 9) or with a new and isolated queue manager (as shown in Fig. 10).

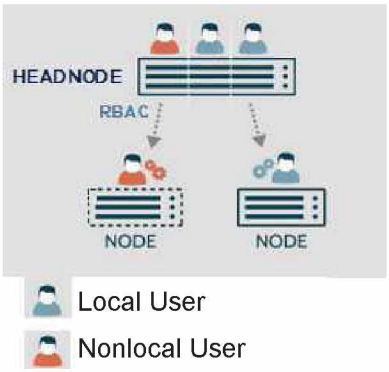

Figure 9: Unified Job Control, Applying RBAC.

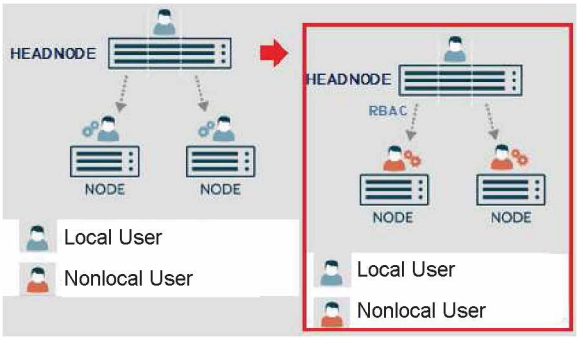

Figure 10: Isolated Job Control, Applying RBAC.

If jobs are handled by a unique job manager, as in Fig. 9, new configurations of users and groups should be defined. Otherwise, if local users' jobs and nonlocal users' jobs are handled with different job managers, then a new standalone manager will be installed in another computer and configure it as the "headnode" of the shared hardware, as shown in Fig. 10.

Security details: The RBAC implementation strategy is supported combining MySQL, Apache, PHP, and CakePHP. Those software technologies along with specific programming determine the correct operation of the proposed system, all of them open source.

The access to resources via web is protected by rules defined for the Apache server. Restrictions and permissions are determined in the ".htaccess" properties for each folder containing resources.

Access to the computer application is controlled by the CakePHP framework. It is not possible to connect to any web portal resource without first doing the login procedure. The model-view-controller chosen for this proposal ensures specific access to predefined views. It can be said that, if something is not programmed in the controller model, it will not be possible to be displayed and used.

In its current version, our proposal does not enable ssh access to users, the users interact with the web system. However, we can change users access in terms of the services allowed from the host "headnode". We can add ssh user acces/ssh user sessions if required. User programs/scripts are stored in the RBAC database, and when the user logs in, they are read and organized in temporary folders for execution.

Runtime constraints: for users in general, and for the "Director" role user/s in particular, it is important to define computational resources usage rules. Runtime limits are useful for preventing single users/user jobs to get resources for an indefinite period of time. Runtime limits are easily defined in job queues of the queue manager tool (Torque PBS in our case). Job queues allow to establish parameters that ensure the correct resource handling in terms of, for example, maximum execution runtime of a job, maximum number of executions per user, users allowed to submit jobs, and maximum number of jobs queued in a cluster, among others. Fig. 11 shows the example configuration of a queue called "cuda".

```
# Create and define queue cuda
#
create queue cuda
set queue cuda queue_type = Execution
set queue cuda max_queuable = 10
set queue cuda max_user_queuable = 10
set queue cuda max_running = 4
set queue cuda resources_default.walltime = 504:00:00
set queue cuda keep_completed = 60
set queue cuda enabled = True
set queue cuda started = True
```

Figure 11: Job Manager Queue Definition: Limits.

Definitions in Fig. 11 are set in the "headnode" of the cluster, and jobs in this queue ("cuda") will be controlled in order to maintain the specific constraints.

4 Conclusions and Further Work

We have been able to combine different open source projects/libraries in order to implement a number of functionalities for HPC cluster management. Our proposal manages local as nonlocal users, so that the HPC cluster/s can be shared among different institutions easily, without the need of complex grid or hybrid clouds software. We are

planning to deploy our development as open source too, once we carefully test our current implementation.

Scalability will be always an issue to analyze, in terms of number of users, fraction of local/nonlocal users, hardware and software resources, etc. Furthermore, we have to carefully experiment on the number of HPC clusters and different sharing options. In the end, we should be able to determine scenarios in which sharing and controlled access will provide specific gains in processing user/scientific jobs.

5 References

[1] Oracle, Oracle Solaris Administration: Security Services, 2012. https://docs.oracle.com/cd/E23824_01/pdf/821-1456.pdf

[2] A. Grama, A. Gupta, G. Karypis, V. Kumar, Introduction to Parallel Computing, 2nd Ed., ISBN: 0-201-64865-2, Addison Wesley, 2003.

[3] Understanding Open Source and Free Software Licensing, Andrew M. St. Laurent, O'Reilly Media, Print: 2004, Ebook: 2008, Ebook ISBN:978-0-596-15308-3.

[4] Adaptive Computing Enterprises, Inc., Torque Resource Manager, Administrator Guide 6.1.0, 2016.

[5] M. Massie, B. Li, B. Nicholes, V. Vuksan, R. Alexander, J. Buchbinder, F. Costa, A. Dean, D. Josephsen, P. Phaal, D. Pocock, Monitoring with Ganglia, O'Reilly Media, SBN: 978-1-4493-2970-9, 2012.

[6] T. Oetiker, About RRDtool, http://oss.oetiker.ch/rrdtool/

[7] J. Lee, B. Ware, Open Source Development with LAMP: Using Linux, Apache, MySQL, Perl, and PHP, Addison-Wesley Professional, ISBN-10: 020177061X, 2002.

[8] R. Dāsa, Learn CakePHP: With Unit Testing, 2nd Ed., Apress, ISBN-10: 1484212134, 2016. https://cakephp.org/

[9] perf: Linux profiling with performance counters, https://perf.wiki.kernel.org.

[10] SUSE LLC, openSUSE - Linux OS. The makers' choice for sysadmins, developers and desktop users, https://www.opensuse.org/

[11] Altair Engineering, Inc., PBS Professional Open Source Project, http://www.pbspro.org/

[12] A. S. Tanenbaum, M. van Steen, Distributed Systems, 3rd Ed., 2017, https://www.distributed-systems.net/index.php/books/distributed-systems-3rd-edition-2017/

[13] Message Passing Interface Forum, MPI: A Message-Passing Interface Standard, Version 3.1, 2015

[14] L. J. Rios, RRDtool data management in HPC Environments (in Spanish), Trabajo de Especialización en Ing. de Software, March 2016, http://sedici.unlp.edu.ar/handle/10915/53669

[15] P. DuBois, MySQL Cookbook: Solutions for Database Developers and Administrators, 3rd Ed., O'Reilly Media, ISBN-10: 1449374026, 2014.

[16] D. J. Lilja, Measuring Computer Performance: A Practitioner's Guide, David J. Lilja, Cambridge University Press, 2000, ISBN 0-521-64670-7.

[17] S. L. Graham, P. B. Kessler, M. K. McKusick, gprof: a call graph execution profiler, ACM SIGPLAN Notices - 20 Years of the ACM SIGPLAN Conference on Programming Language Design and Implementation 1979-1999: A Selection, pp. 49-57, ISSN: 0362-1340, Volume 39 Issue 4, April 2004.

[18] W. E. Cohen, "Tuning Programs with OProfile", Wide Open Magazine, 2004, pages 53-62, https://people.redhat.com/wcohen/Oprofile.pdf