PHRONESIS, A DIAGNOSIS AND RECOVERY TOOL FOR SYSTEM ADMINISTRATORS

C. HAEN, V. BARRA, E. BONACCORSI AND N. NEUFELD
christophe.haen@cern.ch

Université Blaise Pascal

LHCb ONLINE

GOALS

Assist the system administrators by providing a solution to:

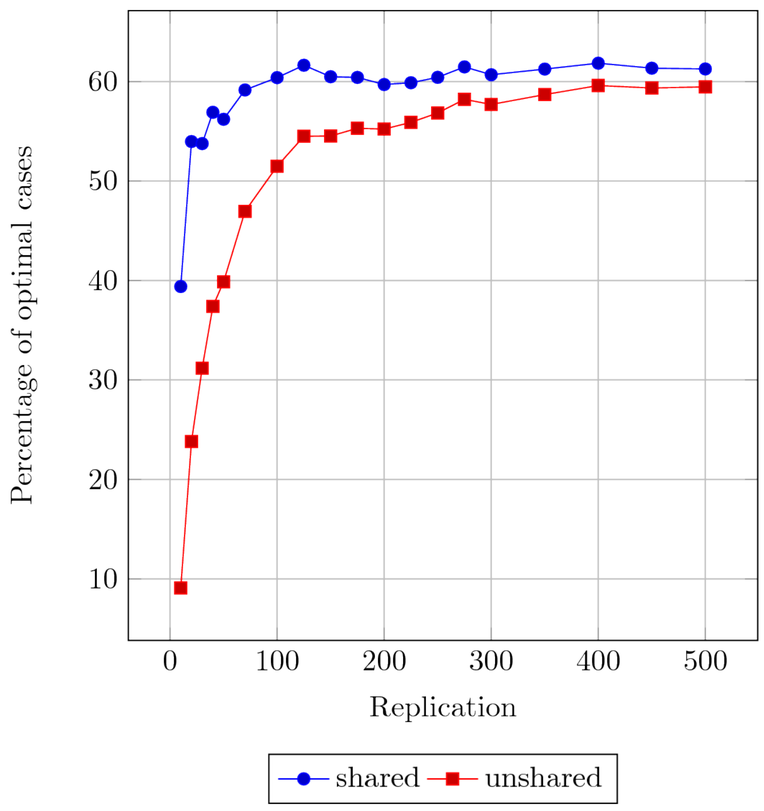
- Reduce their workload
- Propose a diagnostic and a recovery solution
- Improve with experience
- Act as a knowledge base
- Act as a problem history base

METHOD SUMMARY

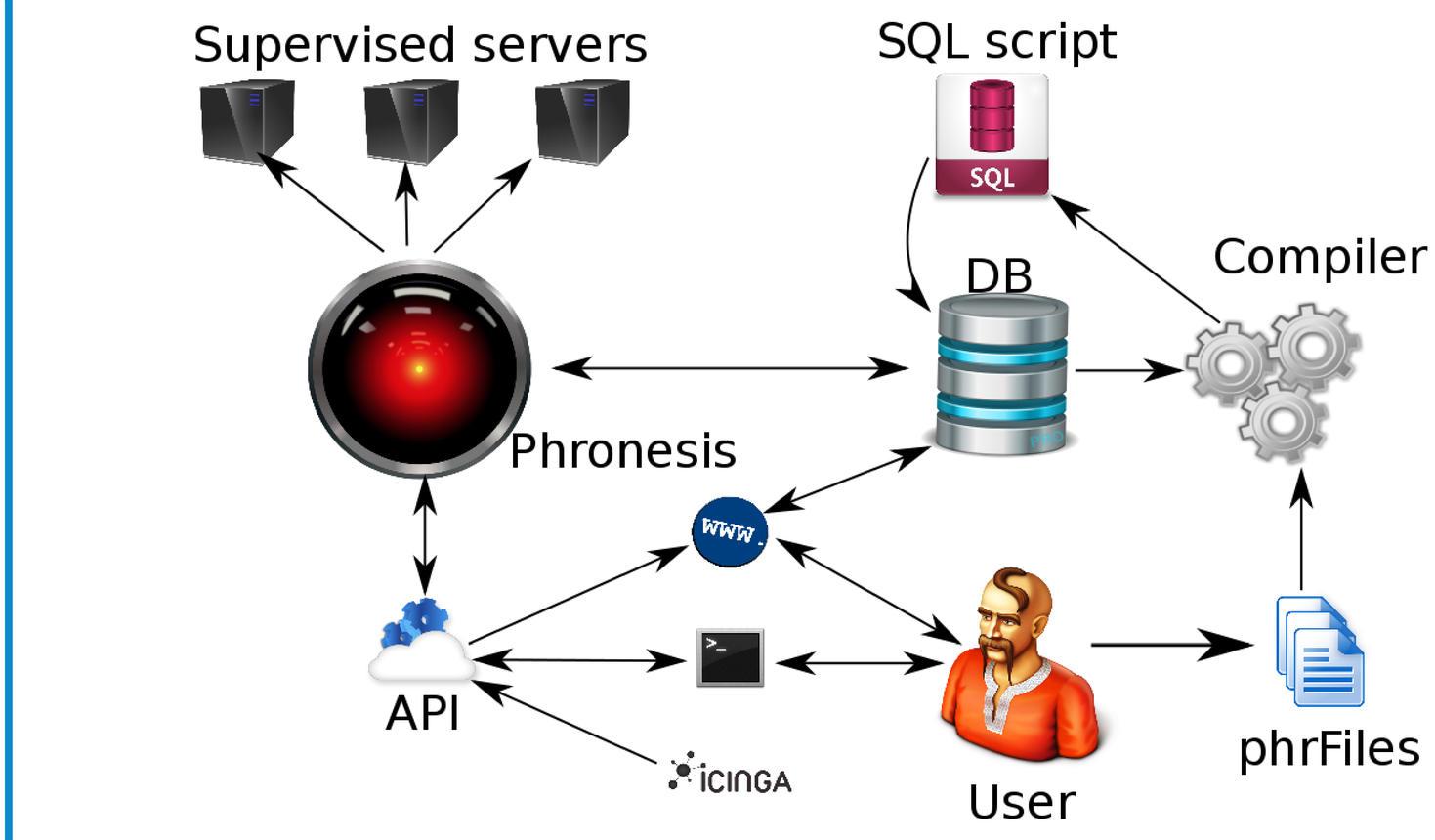
- Linux system diagnoses only
- Generic and non intrusive
- No active monitoring
- Single MAPE-K loop for all systems
- Reinforcement learning algorithm
- Shared Experience principle
- Convention Over Configuration

SHARED EXPERIENCE

This innovative concept consists in grouping similar systems — two web sites for example. It reduces the configuration workload and improves the learning speed of the Reinforcement learning algorithms by sharing the knowledge. The configuration grammar is inspired from the object model whose inheritance concept is similar.

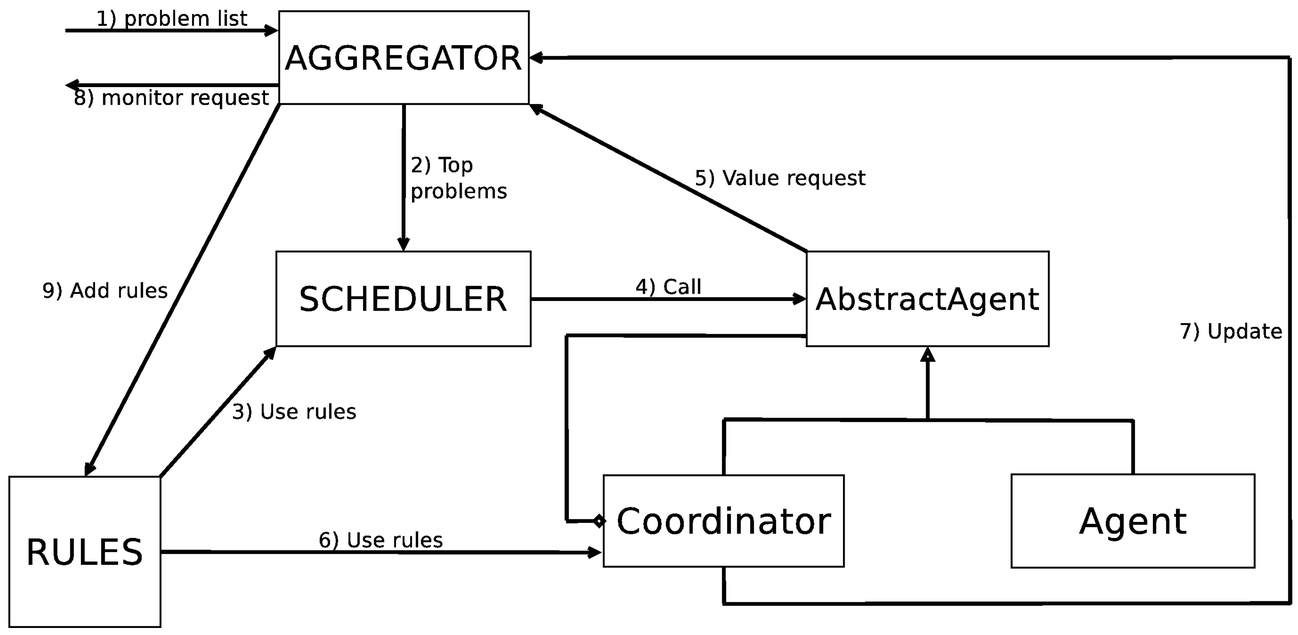


OVERVIEW



The environment is described in configuration files. A compiler parses them and updates the database without loosing the history and the gained experience. Phronesis queries Remote Agents to get information about a particular file, process or the general environment. The interactions with the user go through a common API. A web interface also allows to modify the content of the database.

DIAGNOSIS



The list of problems reported by the user is filtered and sorted based on the dependencies rules between the services. Each service is then explored in turn, following the exploration order established by the Reinforcement Learning algorithms. Once a faulty component is found, a full recovery solution is offered to the user. Based on user's feedback and on the updated problems list, new dependency rules are inferred. The whole process starts again until all the problems are solved or abandoned.

SIMULATIONS

In order to test the algorithms it was important to be able to simulate almost any kind of environment. We developed a complete set of tools to produce Monte-Carlo simulations. They randomly generate problems based on user's input, inject signals to the Core to fake the remote server queries, interact with it to confirm or deny its diagnoses, and produce statistics about Phronesis. Simulations validated the importance of the dependency rules as well as the Shared Experience principle. They also confirmed that the various services' exploration strategies we considered are equivalent in average.

LHCB ONLINE

Phronesis is being deployed on the LHCb Online cluster. Systems under Phronesis' supervision include the Log Aggregation Cluster, the Event Filter Software, the Web Services and the Monitoring Infrastructure. Phronesis' diagnoses and recovery solutions proved to be often correct. Examples are full inodes, bad mount options, corrupted files, no disk space left, processes not running or badly configured. In some cases, Phronesis missed the root cause of the problems. This was either due to a situation not forecast in the design of the code or because of an incomplete configuration.

OUTLOOK

There is still large room for improvements, both technical and functional. Technical improvements concern mainly better performance, higher code quality and more systematic testing. Functional aspects include an extension of the configuration grammar, better handling of clusters with spare parts and dynamic constraints between services. The plan is to add more systems of the LHCb Online environment under the supervision of Phronesis and add coverage for corner cases. We hope to release it as an open source solution that the community would pick up and extend.