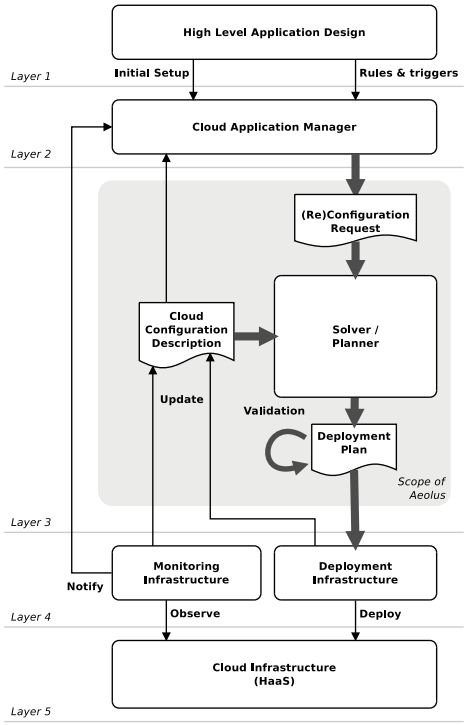
Aeolus: Mastering the Complexity of Cloud Application Deployment

Michel Catan[1], Roberto Di Cosmo[2], Antoine Eiche[1], Tudor A. Lascu[3], Michael Lienhardt[2], Jacopo Mauro[3], Ralf Treinen[2], Stefano Zacchiroli[2], Gianluigi Zavattaro[3], and Jakub Zwolakowski[2]

[1] Mandriva SA
{mcatan,aeiche}@mandriva.com
[2] Univ Paris Diderot, Sorbonne Paris Cité, PPS, UMR 7126, CNRS
roberto@dicosmo.org, michael.lienhardt@inria.fr,
{treinen,zack,zwolakowski}@pps.univ-paris-diderot.fr
[3] Lab. Focus, Department of Computer Science/INRIA, University of Bologna.
{lascu,jmauro,zavattar}@cs.unibo.it

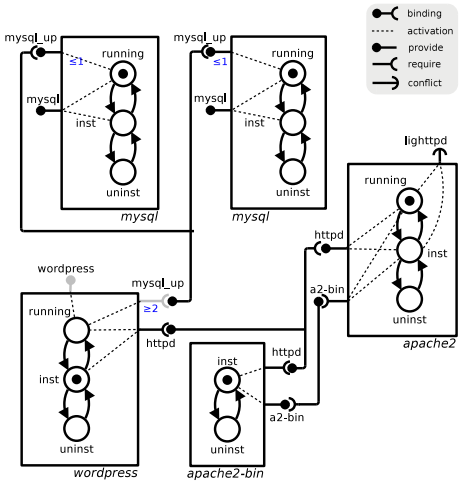
Cloud computing offers the possibility to build sophisticated software systems on virtualized infrastructures at a fraction of the cost necessary just few years ago, but deploying/maintaining/reconfiguring such software systems is a serious challenge. The main objective of the Aeolus project, an initiative funded by ANR (the French "Agence Nationale de la Recherche"), is to tackle the scientific problems that need to be solved in order to ease the problem of efficient and cost-effective deployment and administration of the complex distributed architectures which are at the heart of cloud applications.

The approach taken in Aeolus is to bridge the gap between Infrastructure as a Service and Platform as a Service. In fact, as shown in the picture, applications leveraging the power of the Cloud need to allow efficient deployment and configuration of their components at the level of IaaS and at the level of Services. For this, it is necessary to develop advanced tools that propose a deployment configuration according to the requirements of the user or of a higher level application.



Integrated solutions to this problem needs to deal at the same time with both fine grained software components, like packages to be installed on one single virtual machine, and coarse grained services possibly obtained as composition of distributed and properly connected sub-services. To this aim, in [3] we have proposed the Aeolus component model: a component is a grey-box showing relevant internal states and the actions that

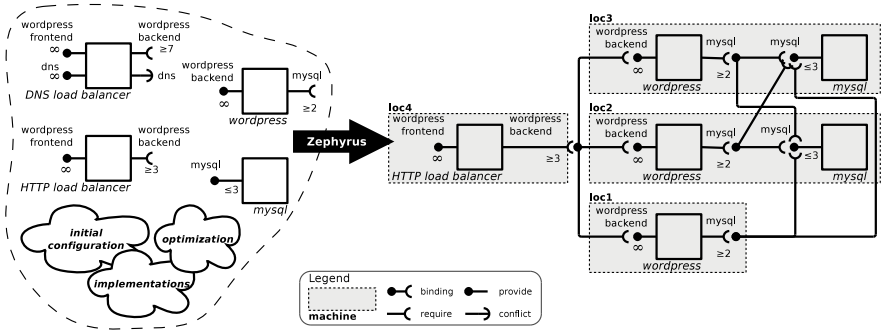
can be acted on the component to change state during deployment and reconfiguration, each state activates provide, require and conflict ports, active require ports must be bound to active provide ports of other components and active conflict ports prohibit the presence of components with specific active ports.



In the Aeolus component model, as depicted, one can express also numerical constraints indicating the maximal number of require ports that can be connected to a provide port, as well as a minimal number of provide ports that need to be connected to a require port. In the example, for instance, the `wordpress` service require two instances of the `mysql` database (for example, to have both a primary and a backup instance).

Based on the Aeolus formalization of software components, we have studied the *reconfigurability* problem: given an initial configuration, an universe of available components, and a target component, verify the existence of a sequence of low-level actions that bring the initial configuration to a new one in which the target component is correctly deployed. This study (see [3, 2] for the details) allowed us to precisely characterize the theoretical limits of the reconfigurability problem: it is undecidable in the general case, EXP-Space hard if no numeric constraints are considered, and Polytime if also conflicts are not taken into account.

The current research in Aeolus is devoted to the identification of efficient solutions. We plan to gain effectiveness by identifying interesting sub-cases in which feasible solutions are possible. The Zephyrus tool [1] is a first achievement along this direction.



Zephyrus operates on a simplified cloud configuration model, abstracting over the dynamic aspect of Aeolus, and focusing on the problem of finding a target final configuration. When searching for such a configuration that satisfies a user reconfiguration request, Zephyrus takes into account multiple factors: the current cloud deployment status, the *software universes* (repository of available components on the different ma-

chines) and desired optimization criteria (e.g. minimize the amount of deployed machines and, hence, the total cost of operation). Zephyrus is guaranteed to find an optimal solution if one exists, and to do so relies on an external constraint solver. Description of the software components that Zephyrus can grasp includes information about their dependencies, as well as resource consumption information (e.g. memory, bandwidth, disk space, etc.) and the distribution packages which they require to work properly. Thanks to this last piece of information, Zephyrus can assign components to virtual machines guaranteeing that they are *actually installable* there. We are currently extending Zephyrus with simple internal states like *installed*, *running*, and *stopped*. We have already implemented a prototype that computes a sequence of state transitions under the assumption that the initial configuration is empty and that the numerical constraint and the conflicts are considered only in the final configuration.

In order to practically experiment and trial our tools in an industrial environment, we are currently developing a n-tiers application deployment engine. This engine applies, on an IaaS, a final configuration proposed by the Zephyrus tool. First, this tool provisions the required virtual machines thanks to the cloud operating system Openstack. When the virtual machines are running, we go to the second step which is *installation and configuration*. In this step, the engine will connect to each virtual machine involved in the n-tiers application to install required packages and configure services. This is done using the deployment tool MSS (Mandriva Server Setup). Finally, the third step consists of launching each services in order to have an application running. These steps are performed, without any human interaction, by combining the MSS configuration informations and the Zephyrus solution. Currently, we have already deployed a varnish load balancer with several instances of Wordpress. We are now integrating more examples with complex databases configuration containing master and slave requirements.

As future work, we plan to extend the Aeolus model to deal also with different administrative domains to support a more realistic representation of complex and possibly multi-cloud applications. In fact, different administrative domains could impose to contemporaneously deal with different deployment and reconfiguration policies. At the moment, the unique form of heterogeneity that we are able to deal with in the Zephyrus model is at the level of virtual machines: those could provide different resource levels as well as different universes of basic packages depending on the installed operating system.

References

1. Roberto Di Cosmo, Michael Lienhardt, Ralf Treinen, Stefano Zacchiroli, and Jakub Zwolakowski. Optimal provisioning in the cloud. Technical report, Aeolus project, Juin 2013. `http://hal.archives-ouvertes.fr/hal-00831455`.
2. Roberto Di Cosmo, Jacopo Mauro, Stefano Zacchiroli, and Gianluigi Zavattaro. Component reconfiguration in the presence of conflicts. In *ICALP 2013: 40th International Colloquium on Automata, Languages and Programming*, volume 7966 of *LNCS*, pages 187–198. Springer-Verlag, 2013.
3. Roberto Di Cosmo, Stefano Zacchiroli, and Gianluigi Zavattaro. Towards a formal component model for the cloud. In *SEFM 2012: 10th International Conference on Software Engineering and Formal Methods*, volume 7504 of *LNCS*, pages 156–171. Springer-Verlag, 2012.