CS486C – Senior Capstone Design in Computer Science

Project Description

Project Title: A Graphical Framework for Distributed Software Deployment	
Sponsor Information:	Hélène Coullon, Assistant professor STACK team IMT Atlantique, Inria, France helene.coullon@inria.fr Frédéric Loulergue School of Informatics Computing and Cyber Systems Northern Arizona University frederic.loulergue@nau.edu

Project Overview:

The complexity of computing, storage and network infrastructures continuously increases. Developers, infrastructure administrators, scientists, or any user have to juggle between smart objects, small devices, personal computers, clusters, private, public or hybrid Clouds as well as future generations of infrastructures such as Fog and Edge computing. Similarly, the complexity of distributed software continues to increase. Most distributed software is modular or component-based, with possibly complex communications, interactions and synchronizations between each component.

Distributed applications and systems need to be installed on distributed infrastructures, and their life cycle need to be handled through time (updates, faults etc.). Such installation and management processes are complex administration tasks composed of many steps and well-defined procedures. This is called the ‘deployment’. Because of this complexity, the deployment of distributed software onto distributed infrastructures is very difficult and must be automated. This is for example what is tackled by the well-known frameworks Kubernetes or OpenStack, that are specific to a given type of virtualization, or by generic low-level administration tools such as Puppet, Chef and Ansible.

However, among the huge amount of deployment frameworks and tools available on the market, none of them takes interest in a critical point: the deployment time, its efficiency and scalability. Without such properties though, future very large-scale infrastructures (e.g. Fog and Edge) and very large-scale applications (e.g. smart-everything apps) cannot be handled.

In the STACK research team at Inria, France, Helene Coullon works on Madeus a new deployment model and its implementation MAD (Madeus Application Deployer) in Python. At SICCS, Frederic Loulergue is interested in the formal analysis of component-based software, and in particular of Madeus assemblies.

The aim of Madeus is to offer efficient deployments by exposing more parallelism in the deployment process. By exposing more parallelism in the deployment process though, Madeus also introduces more complexity for the end-user who wants to deploy an application or a system. Actually, the user has to think about the dependencies between the different tasks of its deployment process and as to code it in MAD. This is necessarily more difficult than writing sequential set of actions. Therefore, a Graphical User Interface (GUI) would be more convenient and would ease the use of MAD, thus increasing its usability and end-user community.

The framework developed in this project will be composed of at least 3 parts:

1. A GUI to allow the graphical design of software deployment according to the Madeus model; the design of the tool should be based on the Model-View-Controller (MVC) architectural pattern,

2. A tree-like data structure to represent Madeus assemblies, and a *documented* API to manipulate it (Visitor pattern),
3. The framework should be extendable by plugins, and the provided plugins should be able to:
   - Read/Write Madeus assemblies from/to files,
   - Simulate the execution of Madeus assemblies,
   - Launch MAD deployments and follow in real time their execution
4. If time permits, other plugins could be provided including:
   - A plugin to generate images of Maedus assemblies (in pdf or/and png formats),
   - A plugin to generate LaTeX (using the tikz packages) representation of Maedus assemblies,
   - More advanced verification plugins such as interactions with the Coq proof assistant.

The GUI code will partly use the MAD implementation. MAD has been implemented in *Python*, under a *GPL* license. Yet, the GUI could be implemented in a different language if preferred or if better libraries can be found in different languages. For example, *JavaScript* could be an interesting language proposing many different graph drawing libraries. Besides architectural decisions, the main difficulty of this project is to choose a well-adapted set of libraries to be able to inherit from already implemented graph layout algorithms (which could be very complex), while being able to override or add MAD specific graphical elements. Moreover, it will probably be necessary to handle threading/multi-process visualization as MAD execute parallel actions. This could also be already handled by existing libraries. You can find some *useful links* at the end of this project description.

MAD is available for free on an Inria GitLab repository. It is entirely documented under ReadTheDocs. Tutorials and examples could also be found in the repository. It is asked to the student team to use those same examples in the repository and documentation of the MAD-GUI.

Knowledge, skills, and expertise required for this project:

The team will need:

- knowledge in graphical user interfaces,
- basics in Python,
- basics on distributed software and Cloud Computing,
- motivation.

Equipment Requirements:

The requirements are:

- development workstations, ideally under a Linux distribution or under Mac OS,
- account on GitHub or GitLab,
- account on documentation platforms if needed during the project (ReadTheDocs for instance).

Software and other Deliverables:

Expected deliverables are:

- The complete professionally-documented codebase, delivered as a repository in GitHub or Gitlab.
- A html developer documentation detailing the design and implementation of the product in a complete, clear and professional manner.
- A html end-user documentation containing some tutorials and examples.

Useful Links:

Madeus formal model:

https://hal.inria.fr/hal-01858150/document

MAD documentation:

https://mad.readthedocs.io/en/v0.2/

MAD repository:

https://gitlab.inria.fr/Madeus/mad/tree/mad_new_implementation

To learn more about graph drawing and layout algorithms:
https://en.wikipedia.org/wiki/Graph_drawing#Layout_methods
Some graph drawing libraries:
https://graph-tool.skewed.de/
http://sigmajs.org/
https://github.com/anvaka/graph-drawing-libraries