SAMARA UNIVERSITY

Experience in organizing flexible access to remote computing resources from the JupyterLab environment using the technologies of the Everest and Templet projects

Опыт организации гибкого доступа к удаленным вычислительным ресурсам из среды JupyterLab с использованием технологий проектов Everest и Templet

Sergei Vostokin, Samara University
Stefan Popov, Samara University
Oleg Sukhoroslov, IITP RAS

9th International Conference "Distributed Computing and Grid Technologies in Science and Education" (GRID'2021)

5-9 July 2021, 2021

New tools for automating workflows in the fields of data science, scientific computing, and machine learning are under active development. One of the significant advances is

the interactive web-based development environment JupyterLab.

JupyterLab allows you to quickly create **a convenient multi-window web interface for a distributed application** that runs from a browser and does not require local installation.

Problem: A scientific computing applications require not only a rich user interface, but flexible access to a wide range of computing resources.

The standard solution – deploying JupyterLab where the computation is done – **doesn't work in cases**:
– there is no technical feasibility of such deployment;
– a distributed application needs to work with several resources at the same time.

We present a solution that covers these use cases, an alternative to commercial cloud solutions such as Google Colab, Yandex DataSphere, JetBrains Datalore, which are also based on Project Jupyter (https://jupyter.org/).

1. We use simple and affordable, but resource-limited options for deploying JupyterLab:
- public cloud deployment based on MyBynder.org;
- virtual machine deployment in a private cloud powered by The Littlest JupyterHub.

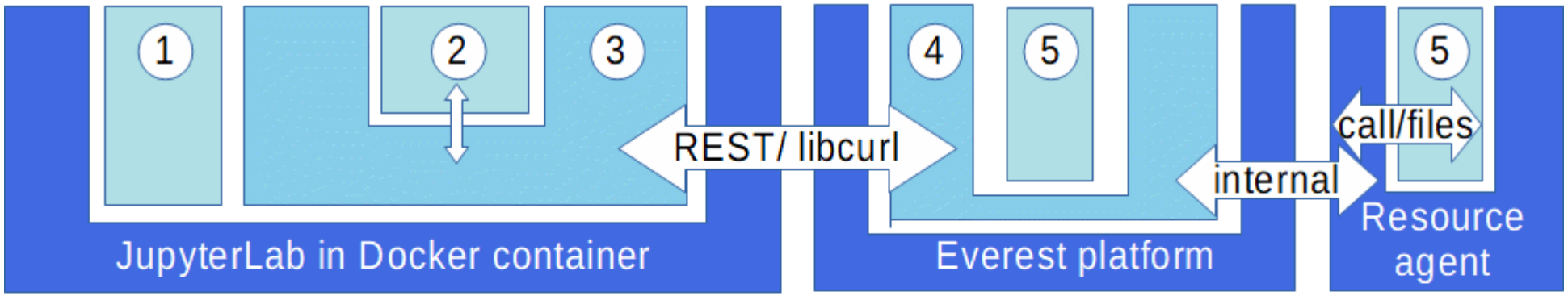
JupyterLab server
– implements the interface (in the form of a Jupyter notebook) and
– starts the orchestrator.

2. We implement an orchestrator for ordering tasks in many-task applications using the Templet SDK (developed by Samara National Research University).

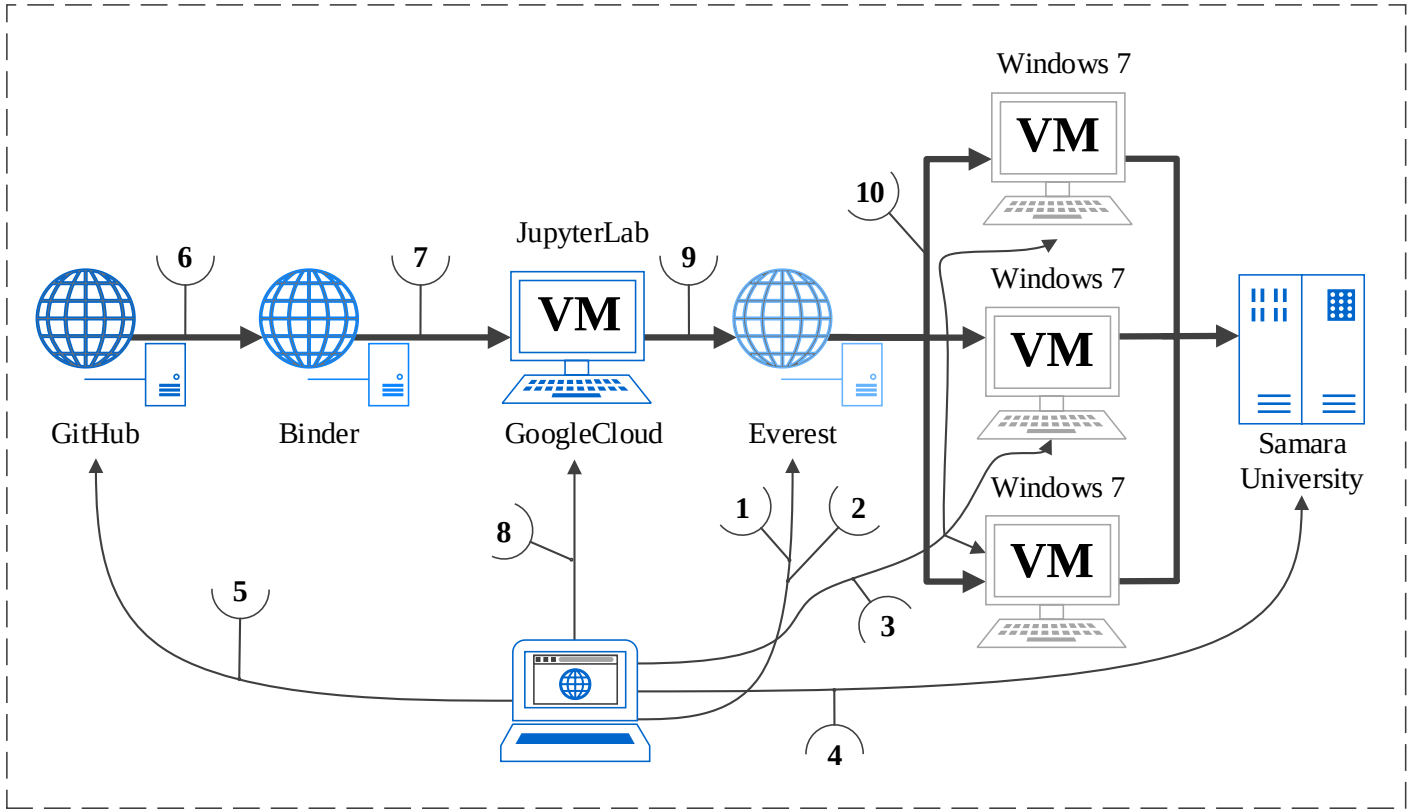
The orchestrator accesses the Everest platform through the REST protocol to perform tasks. It implements a version of actor model.

3. We use the Everest platform (developed by the Institute for Information Transmission Problems of the Russian Academy of Sciences) to control the launch of tasks on computing resources.

Everest server
- defines the policy of access to resources;
- distributes application tasks across resources;
- returns the results of tasks to the orchestrator, which generates the following tasks in accordance with the calculation logic.



1. Jupyter notebook – defines app workflow
2. Orchestrator – dynamically forms DAG of tasks
3. Templet runtime library – uses libcurl to communicate with Everest
4. Everest application(s) – defines format for tasks
5. Code to be run on resource – deployed on resource by Everest

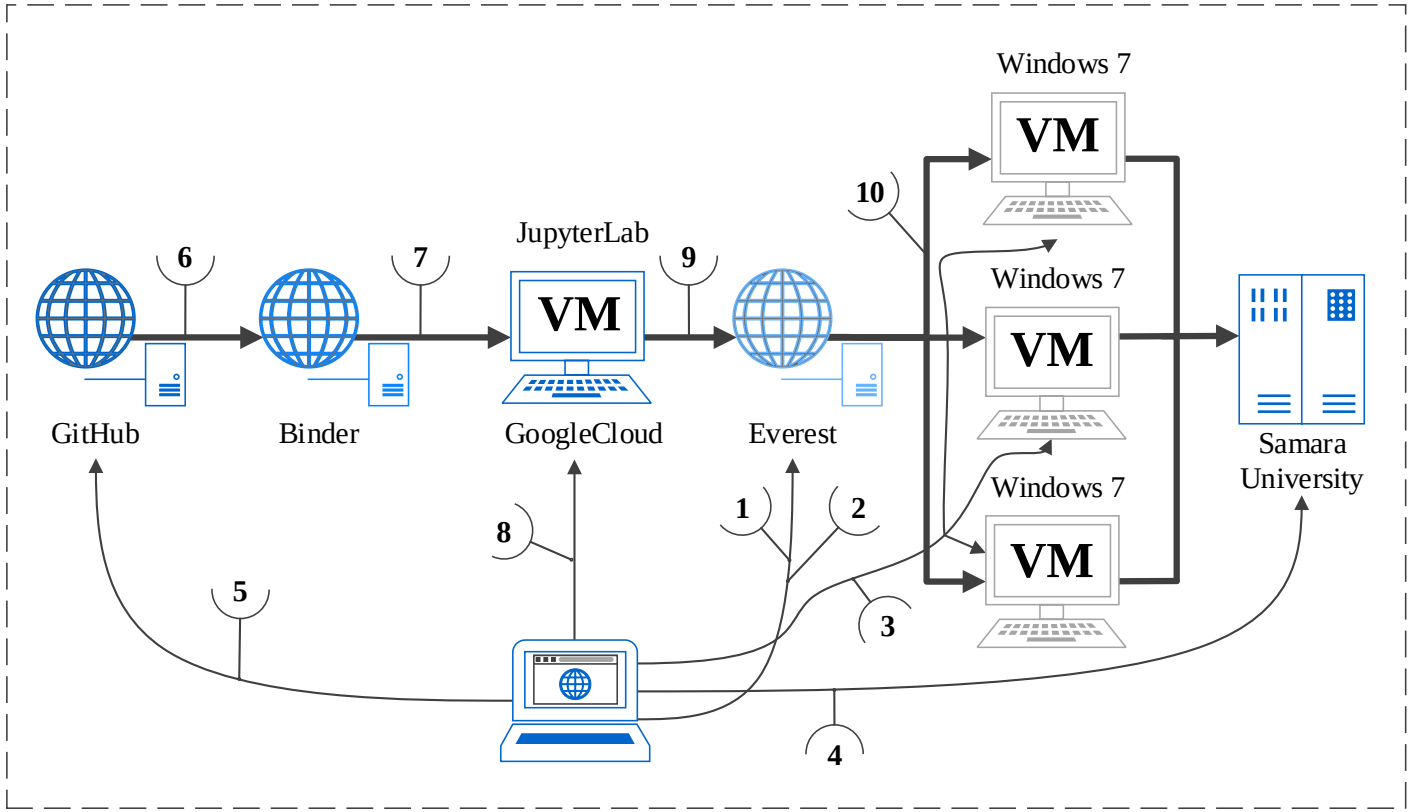


Step 1. Registration of computing resources of the application on the Everest platform. Obtaining access tokens for agent programs through the web interface of the Everest platform.

Step 2. Installing application components. This installation is performed through the web interface of the Everest platform.

Step 3. Running Windows 7 virtual machines in the corporate cloud of Samara University. Installing agent programs on them using the access tokens obtained in Step 1. Verifying the activity of agent programs through the web interface on the Everest platform.

Step 4. Uploading the data to a file server in the corporate cloud of Samara University (if needed). This download can be performed through one of the virtual machines that you started in Step 3.



Step 5. Launching the application orchestrator - from the GitHub code repository via the web interface.

Step 6. Automatically access the Binder service (after completing Step 5) to build a docker container with the application orchestrator running in the JupyterLab environment.

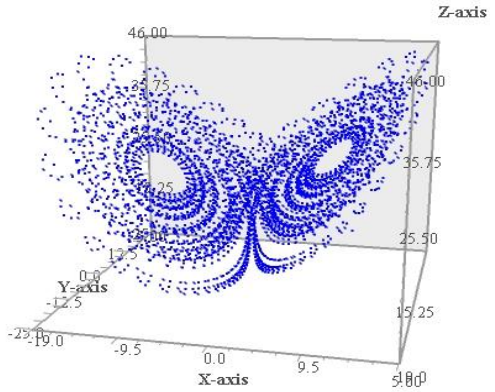
Step 7. Deploy the docker container (from Step 6) in the Google Cloud. Returns the link to the web interface of the orchestrator to the web terminal of the application user.

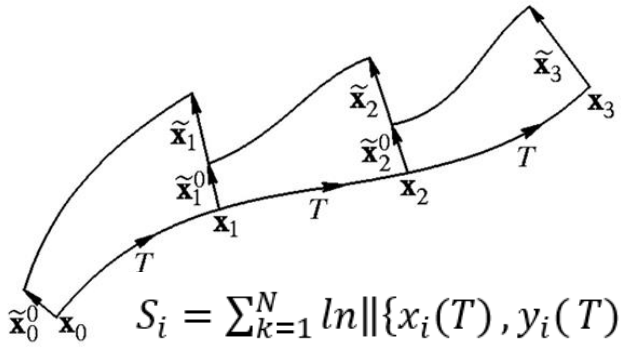
Step 8. Launch the orchestrator by the user via the web interface obtained in Step 7. Start processing.

Step 9. The application orchestrator sends commands to launch next tasks to the Everest platform server and polls the status of previously launched tasks.

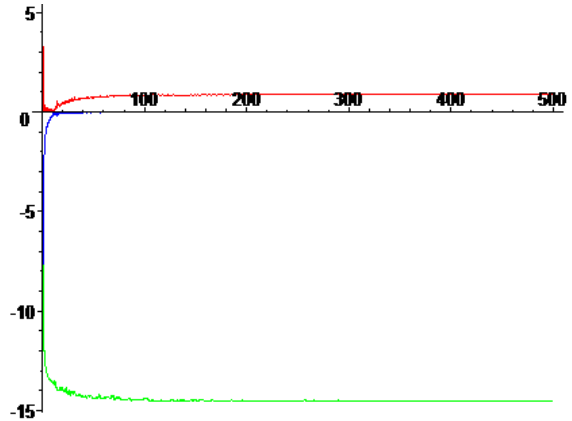
Step 10. The Everest platform server distributes tasks for execution to free virtual machines through resource agent programs (installed in Step 3).

COMPUTE-INTENSIVE EXAMPLE: STUDY OF A DYNAMICAL SYSTEM BASED ON THE CALCULATION OF LYAPUNOV EXPONENTS





$$S_i = \sum_{k=1}^{N} ln\|\{x_i(T), y_i(T), z_i(T)\}\|$$



$$\dot{x} = \sigma(y - x); \dot{y} = rx - y - xz; \dot{z} = -bz + xy$$

$$\begin{cases} \dot{x}_1 = \sigma(y_1 - x_1); \dot{y}_1 = rx_1 - y_1 - x_1 z - xz_1; \dot{z}_1 = -bz_1 + x_1 y + xy_1 \\ \dot{x}_2 = \sigma(y_2 - x_2); \dot{y}_2 = rx_2 - y_2 - x_2 z - xz_2; \dot{z}_2 = -bz_2 + x_2 y + xy_2 \\ \dot{x}_3 = \sigma(y_3 - x_3); \dot{y}_3 = rx_3 - y_3 - x_3 z - xz_3; \dot{z}_3 = -bz_3 + x_3 y + xy_3 \end{cases}.$$

Popov S. N. , Vostokin S.V., Doroshin A.V. *Dynamical systems analysis using many-task interactive cloud computing* // Journal of Physics: Conference Series. — 2020. — Vol. 1694. Issue 1.

Main features of the application:

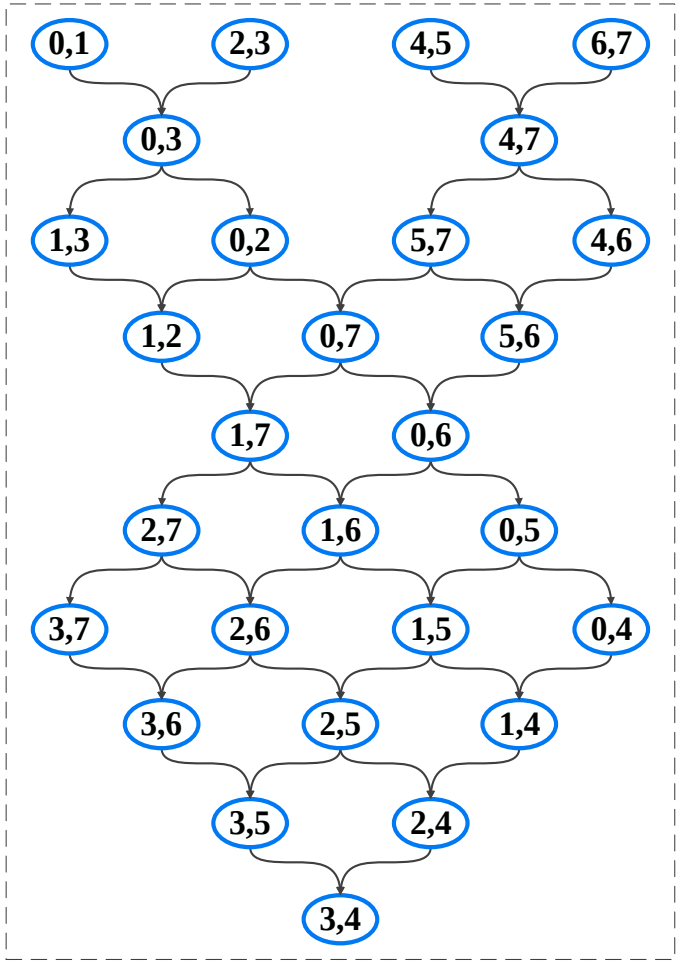
– parallel computing using all licenses of the Maple package of the Samara University;
– sharing virtual machines in corporate cloud and desktop systems;
– distributed bag of tasks with flexible settings.

DATA-INTENSIVE EXAMPLE: BUILDING A DICTIONARY OF WORD FREQUENCY IN TWITTER MICROBLOGS

5.88 GB chunk of data was processed. The size of the input JSON files ranges from 524 MB to 849 MB. The result of processing is an array of 10 text files with a total size of 1.83 MB, 148885 words were found (including word forms and neologisms). The file size after processing is from 158 KB to 223 KB. The type of resulting records in files – <word> <total number of repetitions of a word in 10 files> <CR>. The records were ordered end-to-end across the entire set of 10 files. 3.6x acceleration was obtained on 10 virtual machines.

Vostokin S., Bobyleva I. V. *Implementation of frequency analysis of twitter microblogging in a hybrid cloud based on the Binder, Everest platform and the Samara University virtual desktop service* // CEUR Workshop Proceedings. — 2020. — Vol. 2667. — P. 162-165

Main features of the application:
– DAG of tasks has a complex structure and large dimension;
– tasks interact through the corporate cloud file system.



We have implemented a distributed application design that allows you to:
– work through JupyterLab web interface without local installation;
– deploy JupyterLab separately from computing resources;
– use complex workflow scenarios involving parallel computing on multiple resources.

Current implementations
Everest: http://everest.distcomp.org/
The Templet Project: https://github.com/the-templet-project

Proposal for optimization in future
To minimize the dependency on the JupyterLab deployment method, it is proposed to implement the JupyterLab session as an Everest job.

SAMARA UNIVERSITY

THANK YOU

Sergei Vostokin, Department of Software Systems,
Head of Department (easts@mail.ru)
Stefan Popov, Department of Software Systems,
Postgraduate (stef4n.popov@gmail.com)