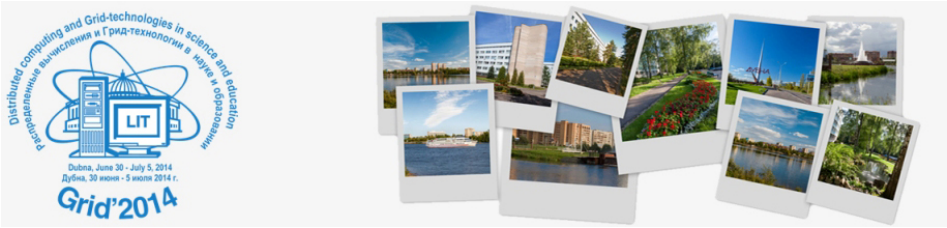

Contribution ID: **45** Type: **sectional reports**

Implementation of computing resource agent for the Everest cloud platform

Monday, June 30, 2014 5:50 PM (20 minutes)

Everest [1] is a cloud platform that supports publication, sharing and reuse of scientific applications as web services. In contrast to traditional service development tools, Everest follows the Platform as a Service (PaaS) cloud delivery model by providing all its functionality via remote interfaces. A single instance of the platform can be accessed by many users in order to create, run and share services with each other. While the platform doesn't provide its own computing infrastructure, it enables users to connect services to external computing resources. Everest can handle the problems of resource allocation, job management, data transfer and so on without the interference of users.

Described in this work is the implementation of integration of computing resources to Everest platform. Such integration is achieved by using a special component, called "Agent", which is an application executed on a resource under a total control of resource's owner, that acts as a mediator between the platform and the resource.

The interaction between agent and the platform is based on WebSocket [2] and HTTP(S) protocols. These protocols allow two-way communication even in the most restricted environments, such as resources behind a firewall. The only requirement for agent's operation is allowed outbound connectivity on a standard HTTP(S) port. On startup agent establishes a connection to the platform on that port, and tries to reestablish it in cases of disconnection. Upon successful connect, agent authenticates itself by sending a secret key issued to resource owner by the platform. After that it starts a waiting loop, which accepts two types of commands: starting a new task and canceling a previously started one. In return agent reports changes in tasks' states and periodically sends information on resource state itself.

Agent is implemented in Python language and uses ws4py library [3] for basic communication with the platform. Agent consists of five components: AgentProtocolClient, Sender, TaskManager, TaskWorker and TaskHandler. AgentProtocolClient is responsible for receiving messages from the platform, Sender handles sending messages back. TaskManager converts messages to actual tasks and starts TaskWorkers which are responsible for task data transfer, execution and monitoring of task state. The latter two operations are performed via TaskHandler which serves as a back-end, which translates generic commands such as "execute task"or "get task state"to resource specific commands. To this moment three types of task handlers have been implemented. LocalTaskHandler executes tasks on a standalone server with Unix-based OS, SLURM- and TORQUE- TaskHandlers execute tasks on clusters with corresponding resource managers.

At this stage the agent is a fully functional component of the Everest platform and has been used in some practical applications. It is still under active development and new features are planned to be implemented in the future, for example support for other types of computing resources.

REFERENCES

1. Sukhoroslov O., Afanasiev A.. Everest: A Cloud Platform for Computational Web Services // 4th International Conference on Cloud Computing and Services Science (CLOSER 2014)

2. Fette, I., Melnikov A., "The WebSocket Protocol", RFC 6455, December 2011.

3. ws4py by Sylvain Hellegouarch, https://github.com/Lawouach/WebSocket-for-Python

Primary author: Mr RUBTSOV, Anton (IITP RAS)

Co-author: Dr SUKHOROSLOV, Oleg (Institute for Information Transmission Problems of the Russian Academy of Sciences)

Presenter: Mr RUBTSOV, Anton (IITP RAS)

Session Classification: Section 1 - Technologies, architectures, models, methods and experiences of building distributed computing systems. Consolidation and integration of distributed resources

Track Classification: Section 1 - Technologies, architectures, models, methods and experiences of building distributed computing systems. Consolidation and integration of distributed resources