Distributed toolkit for Virtual Resources Allocation

- In recent years the evolution of software architectures led to the rising prominence of the Service Oriented Architecture (SOA) concept.
- The services can be deployed in distributed environments, executed on different hardware and software platforms, reused and composed into complex services.
- Efficient use of available computing resources is currently one of the major challenges facing designers of SOA systems.
- On the other hand the Service Oriented Architecture and virtualization come closer to each other, then the need to combine them in an efficient way becomes one of the key challenges for designers of systems based on SOA paradigm.
- The Computational Resources Allocator allows to apply to the service of various hardware resources, dynamically matched to satisfy the requirements of the service.

Features of Computational Resources Allocator

Computational Resources Allocator is a set of tools that enable the design and construction of software platforms, providing flexible execution of the services, in particular:
- effective execution of the service by selecting an service instance at the chosen location
- optimal use of available computational and communication resources
- creating request service instance in the selected computer center and control of available resources resources
- managing software platform using a graphical user interface and admin
- execution of complex services with the use of transformation rules defined SOAP protocol messages

Functionalities of Computational Resources Allocator

- Designation and an indication of available computing resources, to ensure effective realization of the service and its execution.
- Indication the computational resources the knowledge about the current load of computational resources and the historical data collected during the services are performed is used - the data are stored in an internal database.
- Creation and use of available services irrespective of the hardware architecture and ensuring the efficient use of hardware resources.
Characteristics of Computational Resources Allocator

- Multiple runtime environments located in different data centers,
- Execution environments offer virtual spaces, allowing services to run,
- Services included in the virtual capsules - images of the operating system with service.

Architecture of Computational Resources Allocator

- Broker - handles user requests and distributes them to proper instances of virtualized execution resources. Decision making is based on specified criteria of the request distribution: in turn, based on non-functional requirements. It also performs internal requests to coordinate the operation of the system components as well as obtain some necessary information.
- Facade - separates the rest of components, supports communication with them, and collects necessary information for the Broker. It also provides special services for the Broker to test the current state of a processing environment (e.g. characteristics of communication links).
- Controller - manages all components behind the Facade. It is responsible for control processing according to the capabilities of the environment and its current state. It can also route the requests to the services (capsules) independently, taking into account computational resource utilization and performing decisions to start/stop another instance of service.
**Architecture of Computational Resources Allocator**

- Virtualizer - offers access to hypervisor commands. Uses `libvirt` to execute commands that give the project independence from a particular hypervisor.
- Monitoring - collects information about particular physical servers as well as running virtual instances.
- Matchmaker - module responsible for properly matching the requirements of the request with capabilities of the environment and current state of it.

**Atomic & complex services**

- The complex service is put together with basic atomic services, i.e. the ones that cannot be partitioned afterward. The atomic services can be localized in different execution systems and executed by service instances running in these systems.
- The set of execution systems, atomic services, complex services, and service instances are constant for some time, but generally over longer periods, can change.
- The Broker maintains a repository of all known services and components that support them, i.e.: the set of atomic services (AS) available (and seen) for clients, the set of service instances localized at given execution systems, and the set of execution systems (ES). It also maintains information about the available composite services.

**System instantiation**

1. Registration of execution system
2. Registration of atomic service
3. Registration of atomic service instance
4. Getting service information
5. Registration of composite service
Parameters value estimation

To support distribution using dynamic parameters, its current values must be used. For the best effort, a distribution strategy following estimations of values of current parameters can be used:

- best last – the best (minimum) last monitored value of estimated parameter for all instances of service;
- best mean - the best (minimum) average value in k-window for all instances of service;
- best max - the best (minimum) maximum value of in estimated parameter for all instances of service;
- best prediction – the forecasted value derived using any forecasting method, e.g. artificial intelligence approach.

Service Composition

To compose a complex service means to find a set of atomic services and bind them together so that they, as a new service, fulfilling all user functional and non-functional requirements.

Typically automated composition process requires a semantic query (description of a composite service required, often this description is referred to as Service Level Agreement - SLA).

In our case it consists of two parts: service composition and service mapping.

The first is responsible for functional composition of the composite service, while the second adds necessary communication services which connect the functionalities while preserving the QoS defined in the SLA.
When to Use Computational Resources Allocator

- Apply to manage virtual machines in distributed environments to ensure the effective use of available computational resources and communication
- Apply in tasks requiring replication of services to meet non-functional requirements such as time of service
- Apply in the systems that require automatic realization of services compose of atomic services with defined transformation rules
Execution Anomaly Detection

The aim of the anomaly detection is discovering of all abnormal states of the system in relation to the network traffic, users activity and system configuration that may indicate violation of security policy.

Instead Conclusions

- Mechanism for the creation and use of services - using the SOA paradigm and virtualization.
- This makes services independent from the available hardware architecture, and ensures the efficient use of hardware resources,
- Method of scheduling delivery of services in a virtual machine environment - are taken into account the performance parameters of the virtual machine, service and equipment on which a virtual capsule is installed,
- Tool architecture and its constituent modules is open, communication takes place via defined interfaces using XML-RPC for internal communication and the SOAP protocol for external communication.
- All these features taken together offer flexible and extensible environment for communication and composition of the Web services.