User's manual of the resources administrator in SALOME2

Bernard Sécher (CEA)

I. Introduction

Within the framework of SALOME, a user must have the possibility of launching a computer code (via a service of a SALOME component) on a machine which answers a certain number of criteria will have chosen itself. These criteria are mainly the type of the program: sequential or parallel, the access type to the machine (batch or interactive), the operating system, the memory size, the CPU clock, the number of processors, …

From a machines catalogue contained in a "xml" file, the resource administrator will have to determine the machines guarantors with the user criteria, to eliminate among those, those which are out, and finally to select the machine on which the manager of Containers will have to launch the computer code (determination of the less charged machine for example).

The user must also be able to impose the machine on which he wants to work (massively parallel machine for example), and possibly to impose the container to profit from the co-localization of two components.

In addition, the user must have access, via the IAPP with information related to the resource adminitrator: machines list defined in the catalogue of machines with their characteristics, machines lists in service among those. A user of the administrator type must be able to add or remove a machine in the catalogue of machines via the IAPP. One can also envisage two catalogues of machines, a general one defined in the files of configuration of SALOME, second specific to each user. The catalogue of machines will be then the whole of these two catalogues or only the catalogue user.

II. The catalogue of the machines

The resource administrator is based on a catalogue of the machines contained in the file « $KERNEL_ROOT_DIR/share/salome/resources/CatalogResources.xml ». To conclude its services, it must have access to a certain number of characteristics for each machine of the catalogue. All these parameters are defined in a static way in the "xml" file.

Here an example of "xml" file fulfilling the role of catalogue of the machines :

```
<!DOCTYPE ResourcesCatalog>
<resources>
<machine hostname="is111790" protocol="rsh" mode="inter" OS="LINUX"
CPUFreqMHz="2992" memInMB="1024" nbOfNodes="1" nbOfProcPerNode="1"
preReqFilePath="/home/secher/SALOME2_INSTALL/KERNEL_INSTALL/salome.sh" >
<modules modulePath="/home/secher/SALOME2_INSTALL/KERNEL_INSTALL"
moduleName="KERNEL" /> <modules
modulePath="/export/home/LGLS/Products/MED_2.2.2" moduleName="MED" /> <modules
modulePath="/export/home/LGLS/Products/GEOM_2.2.2" moduleName="GEOM" />
<modules modulePath="/export/home/LGLS/Products/SMESH_2.2.2"
moduleName="SMESH" /> <modules
modulePath="/export/home/LGLS/Products/SUPERV_2.2.2" moduleName="SUPERV" />
<modules modulePath="/export/home/LGLS/Products/VISU_2.2.2" moduleName="VISU"
/> <modules modulePath="/home/secher/SALOME2_INSTALL/MYCOMPO_INSTALL"
moduleName="MYCOMPO" /> </machine>
</resources>
```

This catalogue of the machines contains only one machine whose name is "is111790". This machine is a mono-processor workstation. The access protocol is "rsh" and the access mode is of interactive type. Its operating system is "linux". It has a frequency of clock of 3GHz and a read-write memory of 1Go. The catalogue also gives information on the installation of SALOME on this machine. The workspace file of pre-necessary is "/home/secher/SALOME2_INSTALL/KERNEL_INSTALL/salome.sh" and the directories of installation of the principal modules of salome are specified. The directory of installation of the specialized component "MYCOMPO" is also specified: "/home/secher/SALOME2_INSTALL/MYCOMPO_INSTALL".

List of machines characteristics:

- hostname it is the key which determines the machine in a single way (example « nickel.ccc.cea.fr »).
- OS operating system name : « LINUX », « OSF », …
- memInMB central memory size in Mb
- CPUFreqMHz frequency of clock of the processor in MHz
- nbOfNodes number of nodes (higher than 1 in the case of a cluster)
- nbOfProcPerNode number of processors per node
- protocol access to the machine « rsh », « ssh », …

- mode « interactif » or « batch »
- userName name of the user with whom one will connect oneself on the machine (example: "durand"). This parameter is optional.
- preReqFilePath absolute name of the workspace file for the pre-necessary Salome. Example : « /usr/local/products/salomev2.0.sh ». This file **must be** a «sh» file.
- moduleName and modulePath directory of installation of various modules SALOME of version identical to the local version: essential to define the environment at the time of the connection on the machine. That perhaps a single path if all the modules are installed at the same place (example : « /usr/local/SalomeV2.0.0 »).

It appears important to us to consider that a user who launched SALOME with a given version on his machine locally, will have to use the same version on the distant machine for the whole of the modules which it will need for his distant component. The list of the path of installation will have thus to be included in the catalogue of the machines to define the environment of the Container launched on the distant machine. The coherence of the numbers of version of various modules SALOME included in the catalogue of the machines is responsibility for the administrator who builds this "xml" file. The definition of the environment of the container (definition of variables PATH, LD_LIBRARY_PATH, PYTHONPATH) is made in an automatic way by the resources manager in a file Shell script starting from the list of path contained in the catalogue of the machines. This file is sent on the target machine via good protocol (rcp, scp). Then, this file is loaded on the target machine, before the launching of the container itself.

III. CORBA life cycle

The purpose of the SALOME CORBA life cycle is to instance and to destroy the instances of components via containers. The management of these containers is delegated to the containers server. The FindElseLoadComponent() function of the life cycle must have access to the reference of the object "ContainersManager". For that Salome life cycle requires of Naming Service reference CORBA of this object. It can then ask him to launch a container where necessary. For that the Salome life cycle transmits to the containers server the MachineParameters structure filled by the user.

However the search for a component inside a list of Containers (Find() function) and the loading of a component in a given Container (Load() function) remain with the task of the CORBA life cycle.

There can be several instances of the LifeCycleCORBA object. Thus, in python, with each "import salome", an instance of the life cycle is created. It is necessary to be able to manage the fact that several requests for instantiation of the same component can be made simultaneously. The result must impose the launching of only one instance of this component unless the user explicitly asks for the loading of several instances different from the same component. This mechanism can be installed while requiring of an object of the type singleton (for example ContainersManager) to position a bolt.

IV. Use cases

1. c++ examples:

This first example makes it possible to the user to load his component "MYCOMPO" on a machine of the Linux type with a processor whose frequency of clock is higher than 2GHz and with a read-write memory than 500 Mo higher. The component is sequential. The resource administrator will undertake to find the optimal machine corresponding to the criteria requested.

```
SALOME_NamingService *NS=new SALOME_NamingService(orb);
SALOME_LifeCycleCORBA LCC(NS);
Engines::MachineParameters params;

Params.hostname = "";
Params.container_name = "";
Params.OS = "LINUX";
Params.mem_mb = 500;
Params.cpu_clock = 2000;
Params.nb_proc_per_node = 1;
Params.nb_node = 1;
Params.isMPI = false;
Engines::Component_var myCompo = LCC.FindElseLoad_Component( params,
"MYCOMPO" );
```

This second example shows how the user can force the use of a particular machine to launch his component.

```
SALOME_NamingService *NS=new SALOME_NamingService(orb);
SALOME_LifeCycleCORBA LCC(NS);
Engines::MachineParameters params;

Params.hostname = "mymachine";
Params.container_name = "";
Params.OS = "LINUX";
Params.mem_mb = 0;
Params.cpu_clock = 0;
Params.nb_proc_per_node = 1;
Params.nb_node = 1;
Params.isMPI = false;
Engines::Component_var myCompo = LCC.FindElseLoad_Component( params,
"MYCOMPO" );
```

2. python examples:

```
LCC = LifeCycleCORBA(orb);
MachineParameters params;

params.hostname = “”;
params.container_name = “”;
params.OS = «LINUX»;
params.mem_mb = 500;
params.cpu_clock = 2000;
params.nb_proc_per_node = 1;
params.nb_node = 1;
params.isMPI = false;

myCompo = LCC.FindElseLoadComponent( params, “MYCOMPO” );
```

The user can also directly specify the name of a machine:

```
LCC = LifeCycleCORBA(orb);
MachineParameters params;

params.hostname = “mymachine”;
params.container_name = “”;
params.OS = «LINUX»;
params.mem_mb = 500;
params.cpu_clock = 2000;
params.nb_proc_per_node = 1;
params.nb_node = 1;
params.isMPI = false;

myCompo = LCC.FindElseLoadComponent( params, “MYCOMPO” );
```

and even the name of the container:

```
LCC = LifeCycleCORBA(orb);
MachineParameters params;

params.hostname = “mymachine”;
params.container_name = “myContainer”;
params.OS = «LINUX»;
params.mem_mb = 500;
params.cpu_clock = 2000;
params.nb_proc_per_node = 1;
params.nb_node = 1;
params.isMPI = false;

myCompo = LCC.FindElseLoadComponent( params, “MYCOMPO” );
```

V. Containers manager

Containers manager is a CORBA server which is launched to the starting of Salome. It instances an object singleton "ContainersManager" whose role is of locally launching the containers or remote, to preserve the list of references CORBA of the launched containers, and to destroy them. The user can recover his reference CORBA starting from Naming Service.

```
obj = NS->Resolve("/ContainerManager");
Engines::ContainerManager_var cm =
Engines::ContainerManager::_narrow(obj);
```

From a structure of characteristics machine given by the user and of the name of a component, it is possible to recover the list of the machines containing an installation of this component and satisfying the criteria requested.

```
Engines::MachineList_var listOfMachines = cm->GetFittingResources(
params, componentName );
```

Then, it is possible to recover the best machine among the list (in a transparent way, via the resource administrator):

```
CORBA::String_var bestMachine = cm->FindBest(listOfMachines);
```

Or to find a container already existing on one of the machines of the list, or if not, to directly launch a container on the best machine among the list (always via the resource adminitrator):

```
Engines::Container_var cont = cm->FindOrStartContainer( params,
machineList );
```

This last function returns reference CORBA of the launched container. The ContainersManager object adds this reference to its internal list (if it were already not there).

With the end of a Salome session, it is possible to destroy all the launched containers, via the command:

```
cm->ShutdownContainers();
```

This last command is very practical to destroy containers at the end of the Salome session. That avoids leaving a whole heap of containers scattered on the machines via the network.

There is only one instance of the Containers server (singleton). To be able to ensure that several simultaneous requests for launching of a container on the same machine, lead to the launching of only one container, one positions a system of bolt. The first arrived request lance well the Container, the following ones do nothing but return the Corba reference of the Container launched beforehand.

The standard outputs of the containers are logged in file in local /tmp directory. The name of this file is “containerName_machineName.log”.

VI. Resources manager

According to the information contained in a MachineParameters structure, given by the user, the resource administrator seeks the list of the machines corresponding to the selected criteria. Actually, it starts by eliminating from the list of all the machines contained in the catalogue, the machines incompatible with the user needs. Then it reorders the list of the remaining machines according to their more or less great adequacy with the user needs. The selected machine is then a compromise between the order previously carried out and the load balancing of the various machines.

```
ResManager = new SALOME_ResourcesManager(orb);
vector<string> vec=ResManager->GetFittingResources(params,componentName);
```

To select the good machine starting from this list, it uses an algorithm of load balancing.

```
ResManager=new SALOME_ResourcesManager(orb);
vector<string> vec=ResManager->GetFittingResources(params,componentName);
string theMachine=ResManager->FindBest(possibleComputers);
```

The services of this resource administrator are used by the Containers server in order to select the machine on which it will launch a container.