Mobyle Network Help

Mobyle 0.96

Contents

1 Introduction

The Mobyle server can run either as a standalone server (this is the default), or as a part of a "Mobyle Network". This feature authorizes the administrator of a server to:

- **import** programs from other servers, allowing these programs to be displayed as available (though external) programs. A user can submit jobs to execute this program, and the jobs will be seamlessly forwarded to the remote server which hosts the program.
- **export** programs. This allows administrators from other Mobyle servers to import these programs into their portal, hence forwarding the corresponding jobs to the execution server.

2 Motivation

Administrators of systems such as Mobyle portals are often asked to integrate new programs very quickly, to answer the need of a user. The installation and maintenance of such programs can be problematic (is this program compatible with my architecture? how does it work? will it be often accessed by my users?). The proposed solution is to provide Mobyle server administrators the possibility to share resources, exporting and importing programs between servers: while users still access the same portal, the jobs corresponding to some programs can be remotely executed and stored.

3 Architecture

An administrator owning a Mobyle server can decide to export a given program to potential external servers by setting the configuration file accordingly (see 4.2). In this context, a Mobyle server accessed by a user, the *portal server*, can import programs from a set of *remote servers* and send

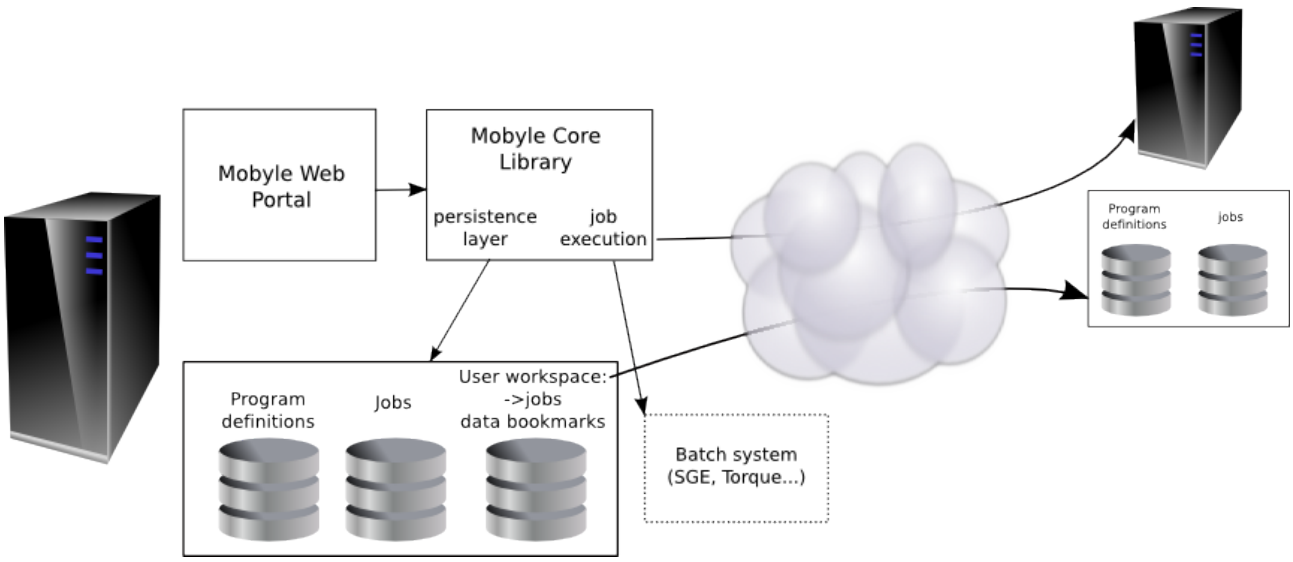

Figure 1: Mobyle components overview.

them job submission requests. In this perspective, Mobyle servers can be seen as a set of Web service providers/consumers, where no central registry is provided (such as BioMOBY or UDDI). Each Mobyle server acts as a node and imports and exports of services should be an agreed-upon process between the two parties (provider and consumer).

3.1 User workspaces and activation

When using the Mobyle system, users are, by default, assigned a user workspace (or sessions), which stores the list of jobs and data bookmarks owned by the user. To avoid potential abuses or malicious usage of the system, this one can be configured to require an activation step before to allow the user to submit a job. This activation can require:

- providing a valid email address and/or answering a CAPTCHA problem for "anonymous" sessions.
- providing a valid email which is confirmed via a confirmation key sent by email for "authenticated" sessions.

3.2 Mobyle Network jobs

Mobyle Network job submissions are routed to an alternative endpoint, which does not require such an activation process in order to execute a job. This alternative endpoint, instead of checking for the existence of an activated session, only verifies that the invoked program is part of the list of exported services. Consequently, as opposed to the classical submission endpoint, all the submitted data values have to be provided, since no session is used to store data on the *remote server*.

3.3 Program imports, job submission, status poll and results display

When a remote program is imported, its definition is downloaded from the *remote server* to the *portal server* to avoid network access overheads, and this definition is indexed during the same deployment process.

Once this task has been performed by the administrator, the *remote server* will not be queried untill a user submits a job. When this happens, the user request is first processed by the *portal server*, which performs a initial set of controls and may replace user workspace parameter references by their values, since the user workspace cannot be directly accessed by the *remote server*. The request for job submission is then forwarded to this last server.

Similarly to job submission, job status polls and job results display are forwarded by the portal to the *remote server*, and the *portal server* only handles the display of the data. The data corresponding to the remote job are not stored on the *portal server*.

4 How do I configure this functionnality into my Mobyle Server?

As of Mobyle 0.96, two variables of the Local/Config.py file are used to configure Mobyle "Network" functionnalities.

4.1 Importing services

The PORTALS variable is a dictionnary which lists all the different servers you *import programs from*. The **key** is the name of the server, as it will be presented in the User Interface. The **url** value represents the path to the CGI scripts of the given server, some of which are used to query the remote server. The **help** value represents the email address to which help requests are sent on the remote server. The **repository** value represents the directory in which the remote XML program descriptions are published on the remote server. The **jobsBase** represents the directory in which the jobs are stored on the remote server.

```
#PORTALS={'portal1': {
#             'url': 'http://otherdomain.fr:port/cgi-bin/MobylePortal',
#             'help' : 'help@otherdomain.fr',
#             'repository': 'http://otherdomain.fr:port/MobyleData/programs',
#             'programs': ['clustalw-multialign'],
#             'jobsBase': 'http://otherdomain.fr:port/MobyleData/jobs'
#             }
#        }
PORTALS={}
```

4.2 Exporting services

The **EXPORTED_SERVICES** variable is a list of all the programs of your server you wish to potentially export to other servers. Adding a program to this list allows its invocation from any remote server.

```
#EXPORTED_SERVICES = [ 'golden', 'abiview' ]
EXPORTED_SERVICES = []
```

4.3 Remarks

- Any change to the list of imported programs needs to be followed by a re-deployment of the impacted programs, using the `mobdeploy` tool.
- Importing a program from a remote server is your responsibility: please ask for the permission of the server administrator. Please do not abuse either any remote resource, and try to comply with the terms of services that server administrators may specify.

5 Current limitations - Future work

It is not currently possible to restrict a program which is exported to a given server or list or servers, although we plan to include this feature in a next release, hopefully very soon.

Additionally, the current configuration process requires to provide much information manually: future releases should improve this step as well.