Milu, the three middleware user interface

Tyanko Aleksiev[*]
Grid Computing Competence Center,
Organisch-Chemisches Institut,
University of Zurich,
E-mail: tyanko.aleksiev@oci.uzh.ch

Stefano Cozzini
CNR-IOM Democritos, Trieste, Italy,
E-mail: cozzini@democritos.it

Riccardo Murri
Grid Computing Competence Center,
Organisch-Chemisches Institut,
University of Zurich,
E-mail: riccardo.murri@gmail.com

Many Grid infrastructures are nowadays serving the scientific communities. Those infrastructures are routinely accessed through some client software generally called: User Interface (UI). Installing and configuring such Grid clients is a challenging task for non expert scientific users even today. Miramare Interoperable Lite User Interface (MILU), is a software which targets mostly this kind of users by providing them a Plug&Play command line interface which contains multiple clients for accessing the gLite, ARC and GT4 middlewares. The package has been developed and evolved significantly during time. Through this evolution process we have always considered the incoming users' requests as natural requisites for the future development. This strategy provides the evident benefit of having a software which better accomplish its purpose, which, in MILU, is to allow a concurrent usage of different Grid infrastructures and create the foundation for more complex scientific gateways.

This paper provides a complete overview of MILU and reports on its usage experiences gathered in the EU-IndiaGrid and e-NMR communities, and as a production UI in our local Grid environment.

EGI Community Forum 2012 / EMI Second Technical Conference,
26-30 March, 2012
Munich, Germany

[*]Speaker.

1. Introduction

The establishment of different Production Grid Infrastructure (PGI) in the past few years has raised a new problem: how to exploit the aggregate computing power. Since different middleware systems are often not interoperable, users are forced to install and maintain several independent sets of commands in order to access and use those PGIs.

As an example we could consider the fact that in order to access a gLite computational Grid infrastructure, a dedicated host working as UI is needed. The machine on which this service is running should be also properly configured. The natural consequence of these two facts is that some hardware resources and system administration skills are required. And what if a user wants to use the same machine to interoperate with more than one middleware?

Direct access to different middlewares from the user's host, enabling a sort of low level interoperation is one of the possible solutions. The basic idea behind MILU approach is to fill the gap between the final user and the installation and configuration of the most popular middlewares.

MILU was developed within the EU-IndiaGrid projects (see reference [1] for a recent overview of the activity of projects) and is currently maintained on within the SISSA/CNR eforge platform (see http://eforge.escience-lab.org/gf/project/milu/). The package is based on and is an evolution of the ICTP_UI and EGRID_UI tools, that were developed a few years ago within the EGRID project [2] in order to make available a portable gLite UI . From this basic task MILU evolved in order to cope with the request coming from users of the EU-IndiaGrid project to facilitate the access to Indian GRID infrastructure (GARUDA) as well.

GARUDA and gLite users are using different methods to access and use resources belonging to the two different infrastructures. To avoid this burden and to make the usage of both infrastructures as simple as possible the MILU tool, was further developed and enhanced becoming a tool which allows seamless usage of different Grid infrastructures from the same Linux workstation. It is important to note that MILU has developed over the years taking into account the requests and experiences coming from its user communities. For this reason, we have mainly based our software development procedure on the following concepts:

- create an usable software which does not require too much expertise.
- take into account the requirements that came out during various training activities where the software was actively used.
- use the feedback provided by the users who autonomously downloaded and tried the software.

This approach allows us to meet the actual needs of the users and satisfy some real use cases. As a result, the following is a list of requirements that have successfully been turned into features:

- **Easy to use and configure**: MILU is easily configurable either for single user and for multi-user environment. No experience in installing Grid services nor about their configuration is required. Using the installer script, included in the tarball, MILU's installation is completed in less than a minute.

- **All in one**: Installing the three different UI separately on your host can pollute the host environment with tens of environment variables and path components. MILU solves this problem by just adding some lines in your .bashrc file and two configuration files in the user's $HOME. Removing/updating the software is thus simple and easy.

- **Portable**: MILU has been tested on different GNU/Linux distributions. We target three popular distributions (CentOS, Ubuntu and Mandriva) and MILU is fully tested against them. Some additional settings and packages may be necessary in order to guarantee the correct functionality which are duly listed in the installation instructions. The known issues are described in MILU User Guide [3].

The rest of this paper is organized as follows: section 2 will discuss some related works in the area of portable and interoperable UI at command line level. In section 3 we describe in some details the package itself and in section 4 we discuss how MILU can be used as interoperability tool for accessing different PGI based on Advanced Resource Connector (ARC) and GARUDA middleware. Finally, some conclusions and future perspective are illustrated in the final section.

2. Related work

MILU has been developed in order to offer a basic command level access to different PGIs. Our specific aim was to provide a Plug&Play Command-Line Interface (CLI) software for users who routinely use command line as working environment. We do not therefore compare here our work with Portals and/or Gateways (like for instance Pgrade portals [4]) which provide Graphical UI (GUI); despite the easiness of usage, such portals remain always less flexible and scriptable than the CLI approach. Thus, for our users, the command level access remains the preferred method for the interaction with the Linux Clusters and Farms.

So, we mention here three similar projects, which were developed keeping in mind the easiness of installing and managing the package.

Two of them are being developed within the gLite developer community: the INFN UI Plug & Play [5] and the CERN TAR_UI [6], both preconfigured to access different Virtual Organization in the EGI production infrastructure. The third one is the ARC Standalone UI, which is the client implementation for accessing the NorduGrid [7]. All of them are available for download and installation as tar files.

These three solutions are sharing the same idea: to make the gLite and ARC UIs portable on Linux platforms without the burden to reinstall from scratch a new machine. The evident advantages of MILU over those solutions are:

- the flexibility in enabling the access to different PGIs from the same Linux host.

- the usability, by offering an easy to install and configure environment and by trying to unifying the command set of the three middlewares.

We also mention here the GridWay project [8] which is actually a metascheduler which provides interoperability among different Grid middleware. This tool is actually fully compatible with

MILU as discussed later in Section 4.2 ”MILU for Interoperability Usage/Testing.”. This makes MILU a good candidate for the base component in a layered implementation of a cross-grid scientific gateway.

As last remark in this section we would like to clarify how is currently MILU related with The Unified Middleware Distribution (UMD), the integrated set of software components that EMI [17] makes available from technology providers within the EGI Community. Two facts are to be considered as important here:

- The necessity of unifying all the available middleware UIs has been realized to be a valid requisite more than 2 years ago, when the work of gathering the different softwares inside MILU has actually started. Thus, the necessary scientific recognition should be given to MILU as a pioneer in its field.
- Adding the Unicore as a supported middleware in MILU would render our UI and the EMI one at the same operational level.

Our challenge in the future development of MILU, thus, remains the integration with the recently-released EMI *2 Matterhorn* distribution [17] which should bring the MILU tool to a higher level of interoperability and give to the final users all the advantages of using our UI.

3. The MILU software package

MILU package is created by gathering together pieces of software from the three different middlewares (gLite, ARC and GT4) UI implementations and by making them work together as a unique one.

gLite [9] middleware originally developed by the Enabling Grids for E-sciencE (EGEE), an EU project, provides a framework for building Grid applications using the available distributed computing and storage resources across the Internet. At the moment MILU supports only gLite version 3.2, for 64-bit x86 architectures. Previous versions enabled also the gLite 3.1 for 32-bit x86 architectures which is now deprecated because no releases have been done since 2010.

ARC version 1.0.0 [7], mainly known for powering the NorduGrid collaboration infrastructure is also supported. The support was added in collaboration with the Grid Computing Competence Center (GC3) [16]. The ARC UI consists of a small set of well-behaved command-line clients, which are fully-supported in MILU.

Globus Toolkit [11] is also present at version 4.0.8. The relatively old web-service based GT4 clients are actually provided due to the fact this GT version is still in use within GARUDA, the Indian Production Grid Infrastructure.

3.1 Architecture

Figure 1 graphically describes the main idea behind MILU.

The main result obtained by MILU is to make usage of three different middleware clients transparent to the final users: this has been achieved by the usage of a wrapping mechanism which enables an abstract layer between the user and the three different middlewares.

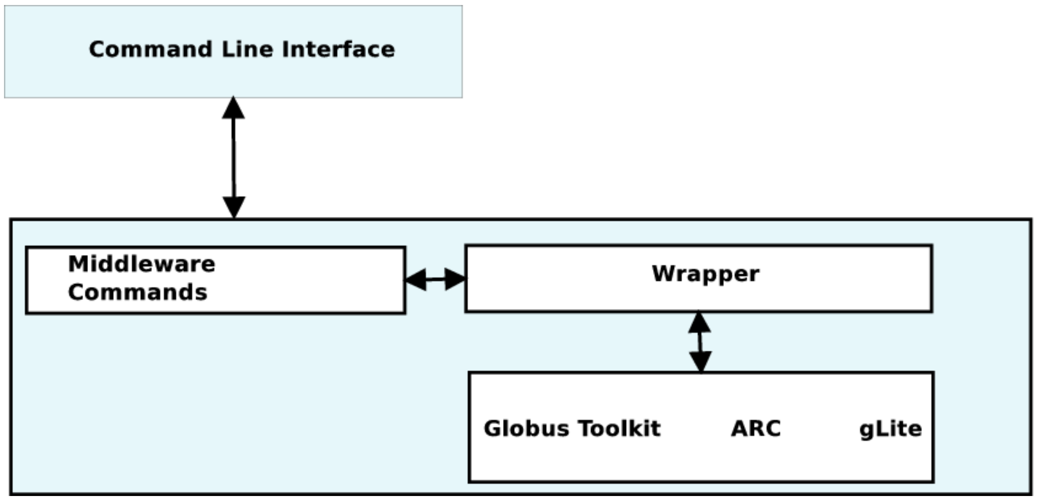

Figure 1: MILU architecture: see text for discussion.

MILU has been created by taking the three different middleware UIs and by making their components work together as a unique one. This process is almost fully automated by a bash script. We provide here some technical details of how exactly this task is accomplished.

Basically, all the stand-alone UIs are initially downloaded and placed in a specific location so that the generating script can recognize them. The execution of the script will have the following effects:

- The distribution directory layout is first created.
- All the configuration files are then copied to the specific locations that have been created in the first step.
- A recursive inspection is performed of the different middleware directories looking for executable binaries or bash script commands.
- Two symbolic link are then created: one linking the real executable and another one linking the wrapper script. We explain this part with more details in the next paragraph.
- The script is then performing some basic checks whether all the required libraries are correctly linked by the binaries and prints a brief report.
- At this point testing procedures can be initialized on the new distribution.

Once the testing procedures are finished on the different Linux distributions a new MILU release can be done. MILU is being distributed as a single "tar" file; all its contents are unpacked in a single directory (named `MILU-`*Version*), on the user's computer. As briefly mentioned before in the heart of MILU functionality is the wrapping of the real executable binaries. In order to better understand this functionality we post some lines of the script bellow:

```
...
source $HOME/.VO.sh
case $PROG in
        ...
        lcg-*)
        source $MILU_INSTALL_DIR/sw/etc/profile.d/common.sh
        export LD_LIBRARY_PATH=$MILU_INSTALL_DIR/sw/external/usr/lib64:...
        ...
        exec "$MILU_INSTALL_DIR/bin/.lnk/$PROG" "$@"
        ;;

        ;;
        ...
...
```

Although we referenced the `lcg-*` commands in the code above it can be considered as a valid example of the MILU's core functionality which performs the following tasks:

- Sources the Run Time Environment (RTE) of the currently supported VO.
- Determines the name of the command being called, which is generally unique across the different middlewares.[1]
- Given the name, the MILU generic wrapper creates an appropriate execution environment by loading environment variables from a middleware configuration file and, if needed, set-up some specific middleware environment variables.
- The wrapper script can now execute the real command with the arguments passed by the user on the command line.

Although conceptually simple, this approach has proved to be very flexible and extensible. Some features are worthy of note:

- Some of the difficulties in making three different middlewares working together are related to library versioning and incompatibilities; these can be overcome by setting the environment variables `LD_PRELOAD` and `LD_LIBRARY_PATH` so that the middleware-specific version of a compatible library is found first by the Linux runtime linker. For example: all the `glite-wms*` commands request the libraries distributed with gLite, thus, the `LD_LIBRARY_PATH` is supposed to be set like:
  `LD_LIBRARY_PATH=$MILU_INSTALL_DIR/sw/glite/lib64...`
  so that the linker will first check this directory when a `glite-wms-*` command has been invoked.

[1] In the rare cases when it is not, the commands have the same functionality and purpose so we can just use the more advanced/compatible version.

- Although most commands only rely on the correct setting of environment variables, some require additional special treatment (e.g., placing a configuration file in a fixed location): these can be handled on a case-by-case basis by having the wrapper script run specific preparation/cleanup actions.
- Environment variable values from the Grid setup do not pollute the normal user environment, as they are only loaded when needed and act just in the scope of the Grid command being executed. We give three examples where this is advantageous. *(1)* Loading the RTE of the currently supported VO only during command execution prevents of having persistent VO set-up and offers thus the major flexibility of managing different VOs from the same host. *(2)* A typical gLite installation for the Large Hadron Collider (LHC) VOs requires setting tens of environment variables, which makes the output of the `env` command unnecessarily verbose; the confinement technique used by MILU allows users to keep all the Grid settings hidden from the user. *(3)* Older version of the Globus Toolkit and gLite used a custom version of the OpenSSL libraries: running system commands with those, or —conversely— running Grid commands with the system-provided OpenSSL libraries would lead to program crashes. The confinement technique used in MILU allowed to use the modified OpenSSL libraries with Grid commands, and keep the system libraries for the rest.
- Finally, note that the technique is easily extensible: the MILU could, e.g., be used to install different version of a middleware and use them concurrently for testing and comparison purpose.

The rest of the software included in the MILU package consists of the real software components (e.g., binary executable files, libraries) included in the different middlewares. An accurate selection has been done in order to reduce the size of the final package. At the end they are all gathered, tested and reorganized inside the MILU directory tree.

3.2 Milu tools

The package contains also some other tools to help user in installing, testing and managing it. The top-level MILU directory contains the installation script, which offers different options and execution modes: for example, single- or multi-user installation can be chosen. From the previous description of MILU's architecture, it is clear that all configuration is already distributed with MILU and stored in MILU's directory tree. So, basically, the only work that has to be performed by the installation script is to replace paths in configuration files to reflect the installation location. This helps in keeping the installation quick and simple, and focus on providing user-level functionality. The installation script also checks if all the necessary software is present on the target host. A `-dry-run` variant of this option is also provided. The integrated help menu facilitates the usage an gives an overview of all the available options.

In order to check the correct installation and operation MILU comes with a self-test script. It runs test cases for all the main gLite and ARC commands, and checks for common signs of misconfiguration (e.g., missing reverse DNS entries, which can confuse GridFTP clients) and potential problems (e.g., clock skew with respect to well-known NTP servers). An effort has been made to test the commands in dependency order and to make user-friendly reports that point out what exact error has been detected and steps to take to correct it. A large number of test cases are additionally available and discussed in the MILU User Guide [3].

Finally, MILU allows switching from one Virtual Organization to another. This is performed with a single command, called `milu_change_VO` which replaces all the Virtual Organization (VO)-dependent configuration files with a version appropriate for the new selected VO. This command is also script-based, and like the previously seen installation script, offers variety of options to the final user: listing all the supported VOs, checking the currently used one, etc.

Switching between different VOs is made possible because of the wrapper-based architecture of MILU: since all configuration is loaded at command execution time, all the work that the VO-switch command has to do, is to change a link to the set of configuration files and environment variables to be loaded by the wrapper script. Each user can independently select a VO of her choice, among the ones supported by MILU, which, at the time of writing are: `euindia, gilda, enmr.eu, gridseed, elab, compchem, gridats`. A step by step tutorial is available in the MILU User Guide [3] for writing configurations for VOs not currently supported in MILU.
Finally it is worthy to underline that when a switch to a VO has been accomplished a new VOMS proxy should be created in order to permit future interaction, but this falls outside the scope of what MILU can do.

4. MILU at work

In this section we describe some of the use cases where MILU has been successfully deployed and used. They represent the most typical usage of the software giving thus a global overview of its capabilities. These significant results have been achieved mostly because of the active collaboration with the end users and the constant feedback they give to the development team. The added value of such kind of development is in giving the final users exactly what they want, meeting their real needs and making the software more usable.

4.1 MILU as portable UI configured for specific user communities and as production UI

MILU, in its previous versions was made available and presented in several events and we received requests to officially include and distribute several Virtual Organizations (compchem, eela, enmr.eu etc.) within the EGI communities.

MILU has been then actively used by EU-IndiaGrid and e-NMR VOs during various training events or just as an alternative way for users to access their Grid infrastructures.

The EU-IndiaGrid [1] project was actively promoting the MILU development: as we already discussed EU-IndiaGrid's purpose is to allow interoperability between the gLite-based European infrastructure, and the Indian "GARUDA" infrastructure based on GT4. Inside this community the package is permanently used not only for training and "Grid schools", but also for production day-to-day use; we are getting some very useful feedback and requests for some future improvements. The continuous collaboration with the EU-IndiaGrid community during the various training activities and schools organized locally permits us to gain even more useful experience doing live interaction and occasional “user testing” - an essential step for improving and making the MILU software more usable for the specific needs of this community. An important recent example of this positive interaction is the integration of the metascheduler as part of the MILU distribution as we discuss in next subsection.

Another major user community is the e-NMR [15] VO. They were interested in using MILU and make it available to the user community as an alternative, preconfigured Plug&Play tool for accessing the e-NMR Grid Infrastructures. They provide us constant feedback about the usage and the experience gained with the UI. MILU was also being actively used during their training activities around the world. We mention some of them below:

- the HADDOCK tutorial where MILU has been used during two different workshops: the EMBO global exchange lecture course in Beijing (April 27 to May 6, 2011) and the one in Taiwan (May 2, 2011)
- the HADDOCK WeNMR workshops in Istanbul and Frankfurt.

MILU is also integrated in the GRIDSEED [10] Virtual Environment. The main idea of GRIDSEED is to provide a simple tool to setup a portable and interoperable Grid infrastructure based on Virtual machines. Within GRIDSEED, MILU is actually serving as a standard User Interface. Two different Virtual Machines (VMs) (a 32-bit and a 64-bit one) are available with MILU already preinstalled and configured. Depending on the hardware and the number of people using the infrastructure, more MILU VM User Interfaces (UIs) can be added, up to a maximum of 10. Such UIs have a set of predefined accounts with a script to create Grid certificates on the fly. This solution is being widely used during several Grid training activities around the world.

Finally MILU is also deployed as standard UI on the SISSA/eLab Portal UI (`portal.grid.sissa.it`) at disposal of our local users. A system-wide installation has been done but users can switch between different Virtual Organization and have access to more than one Grid infrastructures from the same host. From the systems administrator’s point of view this solution allows to significantly reduce the maintenance and upgrade time. From the point of view of the users this provide an unique point of submission for both local resources and GRID ones: the portal machine does allow infact to submit on our local High Performance Computing resources.

4.2 MILU for Interoperability Usage/Testing

We report in this subsection where MILU has been the crucial ingredient in solving a Grid interoperability problem. It is worth to note that this real-world case studies drove the recent evolution of the package and motivate further its development.

MILU was firstly used as a single interface to both gLite and ARC within the GRIDSEED training environment [12]. The purpose of the exercise was to provide a training facility where users could learn how two similar Grid middleware can interact and have a first knowledge on interoperability issues. The UI from both middleware have been unified in the MILU UI where users are enabled to use same credential as well as same VOMS proxy to interact with both CE front-ends. Both LCG_CE and ARC_CE were configured to access the same computing resources. Both gLite and ARC are using the same LFC file catalog as well as the same storage element.

The above services are complemented by a fake Certification Authority provided by GRIDSEED. That work represented a first attempt to setup a training facility that would provide to end users the possibility of facing Grid interoperability features and issues when using quite widely deployed European Grid middleware solutions. While using MILU and GRIDSEED users are able to experience different levels of interoperability: middleware level, scientific gateways level, application access level.

Very recently MILU, jointly with GridWay metascheduler was a key software package to enable interoperation between the GARUDA and EGI Grid infrastructures. Starting with a basic interoperability study exploiting the virtual environment of GRIDSEED involving an LCG-CE and a Globus toolkit 4 web service flavour computing element [13], and on the results of the above example, the project interoperability team of the project lined-out a complete interoperability scenario based on production resources available from both Grid infrastructures as CREAM-CE EGI resources and GT4-WS GARUDA resources exploited simultaneously from a MILU installation completed with Gridway scheduler. In this regard, the GridWay meta-scheduler has been also chosen to facilitate the end-users in spawning their jobs transparently on the resources available. Figure 2 depicts the basic idea for this interoperability issue.

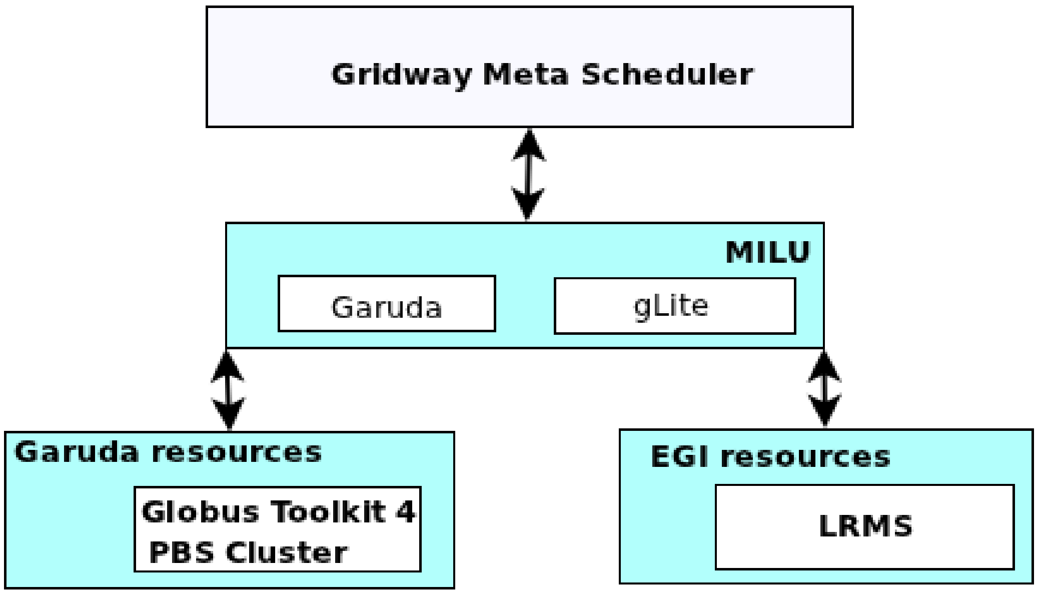

Figure 2: Interoperability between Globus Toolkit 4 and gLite

In the scenario described in the picture the main role is played by the GridWay metascheduler which uses MILU as underlying software to connect with different Grid infrastructures. The key point here is that MILU provides on the same host all the low level software which is needed by GridWay. The Gridway package is therefore compiled and linked against the libraries provided by MILU and this implementation allows cross-submission from the same host toward both Production Grid Infrastructures in a transparent way for the final users. Gridway metascheduler was then appropriately configured in all the components for gathering information about the Grid resources, to accomplish data transfer across EU-IndiaGrid resources in EGI Grid and GARUDA

Grid and finally to execute jobs simultaneously on EU-IndiaGrid VO resources in the EGI Grid and the GARUDA Grid. Submission of MPI jobs was also enabled for both Grid infrastructures as reported in [14]. The GridWay scheduler, properly configured to access both GARUDA and gLite infrastructure, is now available for the MILU 1.2. It is distributed as a separate plug-in on the site hosting the MILU project. User communities have therefore at disposal a simple and efficient interoperability tool to use both infrastructures at the same time. We stress here the fact that this approach can now easily implemented on any Linux host at disposal to the users by simply downloading the software and installing it by themselves.

5. Conclusions

In this paper, the MILU software project, has been presented addressing its two main important features:

- it provides a portable Plug&Play Grid environment for Linux boxes,
- it enables the usage of different middlewares all at the same time.

Details about implementation and some practical use cases have been presented as valid demonstration proofs. The package, together with detailed instructions about its installation and support is freely available and at the time of this writing, we registered more than 2600 downloads of the MILU software.

The positive feedback received by several users and user communities makes us confident that MILU software facilitates the access to the different Grid infrastructures and hides the high complexity of installing and configuring Grid UI. This allows MILU users to be largely shielded from the intricate details of Grid middleware installation and configuration and therefore to dedicate to learn how to use the Grid and not how to install the middleware.

We also believe that MILU can have an impact for the users and developers belonging to emerging communities, as a lower-level tool upon which more sophisticated Grid access mechanisms can be built. There is a clear need for a tool like MILU:

- To provide a smooth transition between different middleware systems, e.g., when the old software still needs to be around as the new one is too immature to completely replace it, or when two infrastructures with different access methods have to be bridged.
- To provide a preconfigured environment that can satisfy the needs of the average scientific user, used to play with command line interface and who does not care about the technical details and only needs a tool that "just works".

MILU is also an important tool towards interoperability. In this context we successfully exploited interoperation among the Indian production Grid GARUDA and the European gLite-based one by means of GridWay.

Future developments of the package, in addition to the standard update of the middleware supported, will be in the direction of providing on the top of MILU useful tools and software as required by the user community which is steadily increasing.

We finally mention that MILU is an open project made available to the community and we encourage interested people to play with it and contribute to it. We think that MILU could be interesting to community developers, in addition to non-technical users, who have been historically the target audience of MILU. Indeed, MILU can be the ideal tool for quickly enabling Grid client access on a Linux machine, for the purpose of rapid prototyping a new tool, or for deploying test/debug instances of running services.

References

[1] A. Masoni, S. Cozzini, *Applications exploiting e-infrastructures across Europe and India within the EU-IndiaGrid project* invited contribution into the book: ‘"Grid Computing", ISBN 979-953-307-540-1, InTech - Open Access Publisher, to be published.

[2] Ezio Corso, Stefano Cozzini, Angelo Leto, Antonio Messina, Riccardo Murri, Riccardo Di Meo, Alessio Terpin, *The new EGRID infrastructure*, 1st International Workshop on Grid Technology for Financial Modeling and Simulation, Palermo, Italy, February, 2006.

[3] T. Aleksiev *MILU user Guide* available as pdf file at ’http://eforge.escience-lab.org/gf/project/milu/docman/?subdir=2’

[4] Peter Kacsuk, *P-GRADE portal family for grid infrastructures*, Concurrency and Computation: Practice & Experience. Volume 23, Issue 3, pages 235-245, 10 March 2011

[5] The package is available at: https://gilda.ct.infn.it/UIPnP/

[6] The package is available at: https://twiki.cern.ch/twiki/bin/view/LCG/TarUIInstall

[7] http://www.nordugrid.org/arc/

[8] E. Huedo, R. S. Montero and I. M. Llorente. *A modular meta-scheduling architecture for interfacing with pre-WS and WS Grid resource management services*, Future Generation Computing Systems 23 (2): 252-261, 2007

[9] E. Laure and C. Gr and S. Fisher and A. Frohner and P. Kunszt and A. Krenek and O. Mulmo and F. Pacini and F. Prelz and J. White and M. Barroso and P. Buncic and R. Byrom and L. Cornwall and M. Craig and A. Di Meglio and A. Djaoui and F. Giacomini and J. Hahkala and F. Hemmer and S. Hicks and A. Edlund and A. Maraschini and R. Middleton and M. Sgaravatto and M. Steenbakkers and J. Walk and A. Wilson, *Programming the Grid with gLite*, appeared in Computational Methods in Science and Technology, 2006 ,

[10] I. Gregori, S. Cozzini, T. Aleksiev, *GRIDSEED: A Virtual Training Grid Infrastructure*, paper accepted to the CLCAR2011 conference, Colima Mexico September 2011 CLCAR 2011, September 5-9 Colima, Mexico.

[11] I. Foster, *Globus Toolkit Version 4: Software for Service-Oriented Systems*, IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13, 2006.

[12] ”Interoperabilty among gLite and ARC middleware in training grid infrastructures: GRIDSEED testcase“ Demo session during the 5th EGEE User forum at Uppsala (Sweden) April 15 2010. Poster reference: http://gridseed.escience-lab.org/uploads/Main/GridSeed-poster.pdf

[13] B.R. Amarnath, T. Selvi Somasundaram, M. Kashif, A. Masoni and S. Cozzini *EU-IndiaGrid interoperability experiments using GridSeed tool* Multiagent and Grid Systems, An International Journal 6 (2010) 261-269

[14] *Network, Grid infrastructure and Interoperability: state-of-the-art between Europe and India* EUIndiaGrid2 Deliverable D3.1, available at http://www.euindiagrid.eu/index.php/pubblications-doc/doc_download/520-eu-indiagrid2-deliverable-d31

[15] http://www.e-nmr.eu/gLite-users

[16] http://www.gc3.uzh.ch/organization.html

[17] http://www.eu-emi.eu/ and http://www.eu-emi.eu/emi-2-matterhorn