Initializing a National Grid Infrastructure — Lessons Learned from the Swiss National Grid Association Seed Project

Nabil Abdennadher[a], Peter Engel[b], Derek Feichtinger[c], Dean Flanders[d], Placi Flury[e], Sigve Haug[b], Pascal Jermini[f], Sergio Maffioletti[g], Cesare Pautasso[h], Heinz Stockinger[i], Wibke Sudholt[j], Michela Thiemard[f], Nadya Williams[g,j], Christoph Witzig[e]
[a]Haute École Spécialisée de Suisse Occidentale (HES-SO), [b]University of Bern (UniBE), [c]Paul Scherrer Institute (PSI), [d]Friedrich Miescher Institute (FMI), [e]SWITCH, [f]École Polytechnique Fédérale de Lausanne (EPFL), [g]Swiss National Supercomputing Centre (CSCS), [h]University of Lugano (USI), [i]Swiss Institute of Bioinformatics (SIB), [j]University of Zurich (UZH), Switzerland
seed-wg@swing-grid.ch

Abstract

In addition to multi-national Grid infrastructures, several countries operate their own national Grid infrastructures to support science and industry within national borders. These infrastructures have the benefit of better satisfying the needs of local, regional and national user communities. Although Switzerland has strong research groups in several fields of distributed computing, only recently a national Grid effort was kick-started to integrate a truly heterogeneous set of resource providers, middleware pools, and users. In the following article we discuss our efforts to start Grid activities at a national scale to combine several scientific communities and geographical domains. We make a strong case for the need of standards that have to be built on top of existing software systems in order to provide support for a heterogeneous Grid infrastructure.

1. Introduction

Grid computing projects and infrastructures are typically organized in multi-national ways, building on resources offered by several organizations and countries. For instance, the EGEE [1] and PRAGMA [2] projects with their world-wide infrastructures comprise partners in several continents. International Grid projects are typically characterized by rather homogeneous Grid middleware systems such as gLite, UNICORE, Globus, or ARC, which makes it rather easy to use the provided Grid infrastructures in a homogeneous way.

There also exist many national Grid efforts world-wide, such as the Open Science Grid in the USA, ChinaGrid, Narengi in Japan, the e-Science program in the UK, D-Grid in Germany, Austrian Grid, and others. These projects usually share the objective to provide a Grid infrastructure at a national scale to support a variety of scientific user communities. To fulfill their mandates, most of these projects receive funding from their local governments.

In Europe, the European Grid Initiative (EGI) [3] has the goal to provide a Europe-wide, sustainable Grid infrastructure that goes beyond the scope of traditionally funded Grid projects. This initiative foresees that each member country has only one Grid contact point organized as a National Grid Initiative (NGI), with the mandate to represent the scientific and Grid community of a country. With this, EGI follows the model of the national research networks such as SWITCH, DFN, BELNET, etc., which provide the basic networking technology to academic and governmental institutions on top of which software services should be deployed. Several of the NGIs exist already (e.g., Austrian Grid, D-Grid, Grid-Ireland, Israeli Academic Grid, MD-Grid in Moldavia, Montenegro Grid Initiative, NorGrid in Norway, PL-Grid in Poland, etc.), several others are currently being created. In Switzerland no such organization existed. This led to the recent foundation of the Swiss National Grid Association (SwiNG) [4], to join EGI as an NGI.

National Grid efforts typically support local or regional user communities and provide local service contact points. They often operate in a way similar to international projects by focusing on single middleware systems. This is possible if the national initiatives are well aligned with the goals of multi-national projects. Switzerland has started to create such a national effort rather late. This means that several

research groups and institutions are already part of different Grid projects that use a wide variety of different middleware systems. Due to this heterogeneity it is rather difficult to combine existing systems and knowledge communities in a seamless way.

To address the specific technical and operational issues of heterogeneous Grid middleware systems and applications from different scientific domains such as physics, chemistry, biology, etc., a so-called "Seed Project" was initiated to bootstrap the activities of the SwiNG association. On one hand, the Seed Project kick-started some of the technical activities in the organization. On the other hand, initial knowledge communities and networks have been created to identify middleware and applications that can be deployed on a Grid. Several middleware pools and scientific applications have been identified, established, and enhanced to allow for national-scale Grid activities. Due to this diversity, several obstacles have been faced which provide strong arguments in favor of standardization for Grid middleware and applications.

In the following article we will discuss our experience from taking the initial technical steps towards such a national Grid infrastructure. Thereby, Grid computing is not only about tackling technical issues [5], but also organizational ones. These become more important when a Grid needs to be built on top of existing infrastructures and projects.

2. Bootstrapping a Grid at National Scale

In May 2007, SwiNG has been created with the aim to (1) foster Swiss science, education, and industry through resource sharing, (2) provide a sustainable Grid infrastructure, (3) establish an interdisciplinary collaboration platform, and (4) represent the interests of the national Grid community. These goals are well in line with the common ideas of e-science and Grid computing.

The Swiss academic research landscape is organized in two federal technical universities, cantonal (i.e., state) universities, universities of applied sciences, as well as other research organizations that are partly funded by the government. Network connectivity is provided by the Swiss research network SWITCH that connects and supports all academic institutions. Similar to the research network, a Grid infrastructure is foreseen to provide access to all these institutions.

Several of the organizations are already involved in national or international Grid projects such as EGEE, CoreGrid, Diligent, EMBRACE, Chemomentum, PRAGMA, SEPAC, etc., which all have their specific project goals and research agendas. Although these projects support different user communities, there is not yet a common infrastructure nor portable applications that can be run on all of the existing infrastructures. The first example of a Swiss-wide Grid activity is the SwissBioGrid (SBG) [6], which was mainly limited to the life sciences. It provides Grid infrastructures based on the Univa UD Grid MP and the NorduGrid ARC middleware systems for protein-ligand docking and bioinformatics applications. While SBG represents a predecessor of SwiNG, more applications and more institutions need to cooperate in order to build a full-scale national infrastructure. Note that CERN, the European Organization for Nuclear Research, although it is partly located on Swiss territory and provides a Grid infrastructure, is an international institution and therefore not member of SwiNG.

To bootstrap some of these activities, we have created the Seed Project in November 2006, with the aim to find partners with Grid interests, as well as to identify some key applications and middleware systems. The goal was to initialize, test, and demonstrate the SwiNG collaboration by means of a common project. This effort did not have direct funding, and was realized through time and computer resource contributions by the involved partners. This means that we had to look for realizable tasks with an integrative potential. We started by submitting a survey to the Swiss Grid community, asking what resources (people, projects, hardware, middleware, applications, ideas) are available and of strong interest. The result is a unique knowledge base that provides an overview of the hardware, software, and Grid expertise in Switzerland. It turned out that — as expected from the involvement of Swiss partners in many different external Grid projects — middleware and applications in Switzerland are relatively diverse. We identified a few prominent Grid middleware tools and exemplary scientific applications, and used them for creating a cross-product (matrix) infrastructure. In this approach, each application from a selected set of applications is "gridified" or enabled to run using each middleware tool from a selected set of Grid middleware products.

By choosing already known middleware and addressing early adopters on the application side, we avoided the "chicken-and-egg" dilemma in bootstrapping a Grid infrastructure: while the scientists are only attracted by a large Grid infrastructure that is worth the extra effort for porting their applications, the resource providers Grid-enable and scale their resources only if there is an explicit demand. In the next sections, we will discuss the selection of the Seed Project Grid infrastructure, and provide details about the scientific applications.

3. Heterogeneous Infrastructure

The main approach of the Seed Project is to build on existing infrastructures and extend them where necessary, rather than using a single, homogeneous Grid platform. This approach is different from other national Grid efforts

that have started earlier, but provides a typical use case and a strong argument in favor of Grid standardization on the protocol and interface level instead of on the implementation level. We believe that our experience is not unique, but representative of many other nation-wide Grid initiatives.

We selected a few representative Grid middleware tools for the Seed Project. Each tool initially requires the building of its own resource pool. The final idea is of course to allow interoperability among all heterogeneous pools. We agreed on the following selection criteria: each middleware product should already be deployed at as many as possible involved partner sites, has to be accompanied by sufficient expertise and manpower, and must be supported within existing larger Grid efforts. In addition, the selected Grid middleware should not have too complex requirements, must be diverse, and needs to provide a sufficient set of capabilities. The initial focus is on compute-intensive, not on data-intensive applications.

According to these criteria, we decided to build middleware pools based on gLite, the middleware of the EGEE project, ARC, the middleware previously used in the SBG project, XtremWeb-CH, a desktop Grid platform developed at HES-SO, and Condor, the middleware used for the EPFL Campus Grid. In this way we support some of the most popular Grid flavors, given that gLite and ARC build on the Globus Toolkit. The details of each middleware infrastructure, and Grid security and interoperability issues spanning across all pools will be discussed in the following sections.

3.1. Grid Security and Certificates

Grid security and certificate-based authentication are among the most commonly agreed standards in the Grid community and are supported by many middleware implementations. This has also been proven true in our experience. Therefore, a virtual organization has been created that allows for X.509-based user certificates from several certification authorities, including a Swiss national one operated by the research network.

In addition to standard X.509 user certificates, we support ad-hoc generated short lived certificates from a *Short Lived Credential Service (SLCS)* [7]. A user can obtain these certificates by password authentication to the *Authentication and Authorization Infrastructure (AAI)* [8] system. The AAI service delegates the authentication process back to the home institution of a user. This considerably lowers the entry barrier for new users since experience shows that obtaining and managing X.509 certificates is cumbersome for the average user and often a source of problems. Additionally, the identity management becomes simpler since the central CA is not required to keep a separate master user database. Apart from the need for certificate renewals for long-running jobs, SLCS thus provides an attractive service for a nation-wide Grid infrastructure. gLite, ARC, and Condor support and have been successfully tested with short lived X.509 certificates. XtremWeb-CH relies on a central user database instead of certificates, requiring an interface solution.

3.2. gLite Pool

Overview: gLite [9] is the middleware developed under the umbrella of the EGEE project [1]. It has a large user base and is deployed on many resources since it is the middleware of the LHC Computing Grid (LCG) project [10], the world's largest Grid infrastructure providing the computational power for analyzing data from CERN's LHC (Large Hadron Collider) experiments.

gLite is based on a number of components from Globus [11], among them the GSI. gLite adds support for virtual organizations (VOs) and extends data management functionalities. Among the principal components of a gLite site are the computing element (CE) and the storage element (SE). The CE is a gateway between the external Grid and the site's compute nodes (termed *worker nodes*) by interfacing to a local batch scheduling system. The SE offers a number of data transfer-oriented services with standard interfaces and gives access to the site's disk and tape storage. All resources announce their availability and capabilities through an information system which is used by the Grid resource management to render decisions on where to place jobs.

A major challenge in CERN's high energy physics projects is dealing with petabytes of distributed data. gLite offers many Data Grid components such as file catalogs and Grid transfer protocol-enabled storage management systems. gLite also offers a fairly advanced authentication and authorization model based on VOs, where VO users can be further organized into hierarchical subgroups. Since this has to be supported by all resources, a large part of the development effort in gLite is spent on getting the resource services into compliance. This rich functionality and the involvement of many vendors result in complex package interdependencies and a substantial configuration effort needed for setting up gLite resources. This is also the reason for the middleware being officially supported only on the Scientific Linux platform. Intensive usage within LCG coupled with large scale data challenges has led to the abandonment and rewriting of several components. Functionality late in coming often had to be implemented by the LCG groups, thereby adding to the large number of gLite components, and resulting in multiple competing implementations for some packages. However, as the start of the LHC experiment is drawing nearer, an increased amount of consolidation is observed. In summary, successfully running gLite services requires substantial manpower, and is quite complex and intrusive for a computing center.

Deployment: In Switzerland, gLite and its predecessors have been in production use since 2002 by a collaboration of the Swiss high energy physics groups and the national supercomputing center CSCS. In addition, the middleware was used by the DILIGENT project. Within the SwiNG Seed Project, a small dedicated test environment was set up among the resource providers that already ran gLite pools. SWITCH provided a new CA for issuing credentials to users and integrated it with their SLCS. This allowed tests to be done without impacting the existing production infrastructure.

Experience: Installing a gLite user interface (UI) needed for submitting jobs is relatively straightforward, but a large effort is required from the resource providers for installing and running the gLite service components. Even though gLite is the heaviest in terms of required manpower, the fact that there are already substantial resources running it and its data management capabilities, make it a candidate for a national Grid infrastructure.

3.3. ARC Pool

Overview: The Advanced Resource Connector (ARC) [12] is the middleware developed by the NorduGrid project of the nordic countries in Europe. Similar to gLite, it is based on Globus [11]. ARC provides fundamental Grid services, such as information services, resource discovery, monitoring and management, as well as job submission, management and brokering. Authentication and authorization are based on GSI and are enriched with the support for VOMS certificates. In order to better cope with the ARC architecture, the GridFTP-based file transfer service, and access to the computing resources have been rewritten.

The aggregation model of the computing resources is the cluster, so the ARC resource pool is seen as a collection of federated clusters, each with its own local computing element (CE). ARC has the notion of a storage element (SE) for a Grid-wide persistent data repository. Currently, it is compliant with the SRM interface, but it only provides GridFTP-based file transfer. Unlike gLite, it does not have the notion of a worker node with additional installation requirements. ARC supports up to 22 different Unix distributions for client, SEs and CEs, as well as backends for several Local Resource Management Systems (LRMS). Applications can each be set up in a dedicated runtime environment (RE). This results in a non-intrusive solution for an easy integration of existing computer resources.

However, with the GridFTP SE interface ARC provides only a limited support for complex data management. The file catalog service is based on Globus RLS, and at the moment there is no notion of data proximity to instruct the resource broker to direct requests towards data. If not configured properly, ARC may lead to an inefficient use of the underlying computing resources, as one of the system requirements is that all nodes share a common filesystem. This has been observed already in several projects to restrict performance and stability. The information service is limited and does not scale on large distributions.

Deployment: The SBG project preceding SwiNG used ARC to aggregate resources from several sites in Switzerland. CERN's LHC project provides support for the ARC middleware mainly for Tier 2 and 3 sites. ARC has been used in the Seed Project as several partner sites (CSCS, SIB, UZH) already had it enabled on their resources. Using the SLCS certificates and leveraging the SBG infrastructure provided a simple ARC pool build: the centralized information service used for the SBG project has been extended to accept registration from seed sites, and the required applications have been enabled by creating additional REs.

Experience: ARC can be seen as the most successful middleware pool within the Seed Project. Nevertheless, running, controlling, and managing an ARC pool requires a coordinated effort at all involved partners sites, as stability is still an issue. Even if the services provided by ARC are limited and rather simple to configure, all sites need to agree on a common configuration schema to allow resource sharing. For the scientific applications, each site must cope with the RE specifications, as this cannot be made automatic. Users need to be aware of the RE specifications for creating their job descriptions. Finally, errors coming from both users and resources have to be tracked cooperatively, as the accounting and logging services are still limited.

3.4. XtremWeb-CH Pool

Overview: XtremWeb-CH (XWCH) [13] is a volunteer computing middleware developed at HES-SO that makes it easy to deploy and execute parallel and distributed applications on a public-resource computing infrastructure. XWCH can support diverse HPC applications, including those with large storage or communication requirements. Universities, research centers, and private companies can create their own XWCH platforms while anonymous PC owners can participate in these platforms. They can specify how and when their resources could be used.

The objective of XWCH is to develop a real high performance P2P platform with a distributed scheduling and communication system. The main idea is to build a completely symmetric model where nodes can be providers and consumers at the same time. XWCH can execute distributed and parallel applications where tasks can/should communicate. Communications between compute nodes (*workers*) can take place directly or via warehouses if direct communication is not possible. Workers are not dedicated to computing. They are also often behind network firewalls that do not allow incoming network connections. This requires the

use of a pull model in which workers periodically request work from a central coordinator, rather than the push model used by certain Grid software.

Deployment: Testing within the Seed Project has been carried out on a set of computers belonging to two SwiNG members (two sites at HES-SO, one at UZH). The PHYLIP application has been previously ported to XWCH, and the porting of GAMESS has been started as part of the project.

Experience: XWCH workers are quite easy to install and run on given compute nodes. However, there are still a few issues that need improvement: security is relatively limited, and user management based on a central user database, not on GSI and VOs, as in most other Grid middleware considered here. The porting of applications needs some effort. Additionally, no special data management features are provided up to now.

3.5. Condor Pool

Overview: Condor [14] is a Grid middleware that is typically deployed within the boundaries of a single institution, even though it can be used across multiple institutions or sites. Condor is a high throughput computing environment which uses idle cycles of the existing computing resources for computational tasks. By deploying Condor on desktop machines, there is usually no guarantee that a given resource will always be available. This makes a Condor pool a very volatile environment, where computational resources appear and disappear at random. The typical usage of such a Condor pool includes running single-threaded applications in parameter scan mode.

Condor supports several authentication and authorization mechanisms such as GSI and Kerberos. Currently, it does not support VO-based certificates. Condor does not have a built-in data management system, and relies on a shared file system (NFS or AFS), or on its built-in file transfer mechanism to send input/output data to a compute node. Condor is also able to communicate with Globus sites, via Condor-G, which is the job management part of Condor. On the other hand, from the view of Globus and related Grid middleware tools such as ARC, Condor can be seen as just another LRMS, and corresponding interfaces are available.

Deployment: Testing for the Seed Project has been done on a subset of computing elements of an existing Condor pool. At the same time this pool is used as an institutional production Grid. A Condor pool has been deployed at EPFL, by utilizing non-dedicated computer resources: most of the compute nodes are standard workstations, used by students or staff during the day, and used by Condor during nights and weekends, when the workstations are idle. In the EPFL configuration all job submissions occur from a single submit server in order to centralize accounting data and to provide a uniform environment to the users. This also simplifies the debugging of problematic job submissions by having all user data directly on one server.

Experience: An issue with the Condor configuration at EPFL is that the selected scientific applications were not available on the compute nodes. It was not possible to install the applications on the pool nodes due to the local resource management policy that disallowed external software installation. This means that the application user has to rely on Condor's built-in transfer protocol to transport the application binaries along with any input data to the compute node.

3.6. Interoperability

Due to the fact that most middleware systems implement their own mechanisms for resource and data management, information representation, or job submission, they can not interoperate with each other, despite existing OGF standards. To overcome these interoperability issues, there are two general solutions: (1) A meta-middleware could be developed and installed on all middleware pools. Users would submit their jobs to the meta-middleware system instead of submitting to individual pools. This approach represents quite a complex endeavor, in particular since not all discussed middleware tools expose the same interfaces or adhere to the same standards. This solution is out of scope for the short-term Seed Project. (2) One could enable interoperability by one-to-one wrappers. For example, some middleware systems can operate as bridges, by transforming job submission, data transfer, and other information from one format to another, by interfacing with LRMSs, and by providing information about the resource status in a unified way. Users can then use a single entry point for job submission, without having to learn multiple job description languages.

Following the second approach, the Condor pool described previously has been integrated into the Seed Project ARC pool. The ARC middleware interfaces well with Condor, providing a conversion of the submit files from ARC format to Condor format and reporting to the user the current state of the submitted jobs. There is no need to provide accounts on the local submission servers for users coming from outside of the institution since this aspect is managed by ARC and SLCS certificates. In order to separate local jobs (from EPFL users) and external jobs (from the Seed Project) in the scheduling queues, a gateway machine has been installed. It hosts the ARC middleware in charge of converting ARC jobs into Condor jobs, and converting information back into ARC format when the job is done. A set of Cones application binaries (see below) has been prepared for all platforms controlled by Condor. The ARC backend for Condor has been modified in order to "append" the required binaries to the Condor job on the gateway once the

destination queue has been selected. This step is completely transparent for the user, making the Condor pool behave as if applications are installed on the compute elements.

Note that our work on interoperability is just a basic start. Grid standards such as the Job Submission Description Language (JSDL) developed by OGF have to be supported by all middleware products in order to avoid meta-middleware systems or middleware bridges. An even more important aspect is the need for agreed job submission protocols to interact with middleware systems and computing and storage resources. Therefore, our current middleware pools can be considered as "islands", but we expect to see improvements as the middleware progresses and new standards are adopted.

4. Scientific Applications

One of the main objectives of the SwiNG Seed Project was to initiate collaboration among as wide of a community as possible. Thus, we went beyond previous approaches in Switzerland such as SBG [6], and selected several scientific applications with diverse characteristics and from different domains. This was also important to provide a better test base for the chosen heterogeneous infrastructure.

To select suitable applications, we applied a number of criteria: there should be a need for each application from the Swiss scientific user community, and its computational demand should warrant a Grid execution. Sufficient expertise has to be present, and the applications should not require too complex requirements and should be able to work with the limited resources we had available for the Seed Project. The gridification of the applications would be done in the most simple way, and if possible without changes in the source code. The selected applications should be diverse in as many aspects as possible. Finally, we might also reuse already existing Grid-enabled applications.

Based on these criteria, we selected the following initial applications: Cones, GAMESS, and PHYLIP, which will be discussed below. All these three applications are CPU intensive and therefore ideal candidates for running on a Grid. We also investigated the commercial Huygens [15] software for remote deconvolution of microscopy images in the beginning, but then decided that its Grid-enabling would be too much of an endeavor for the Seed Project.

4.1. Cones

Background: Cones [16] is a mathematical crystallography program to investigate quadratic forms. It was developed by Peter Engel at UniBE. In dimension d, the cone of positive-definite quadratic forms has dimension $d(d+1)/2$. The subdivision of the cone gives an enumeration of the combinatorial types of primitive parallelohedra. For a given quadratic form, the Cones algorithm determines the subcone of equivalent combinatorial types of parallelohedra, and for each wall of the subcone, it determines the adjacent subcone in order to find new types. In the 5-dimensional case, only 222 combinatorial types of primitive parallelohedra exist. However, in the 6-dimensional case a combinatorial explosion happens. With help of the Cones algorithm, 161,299,100 combinatorial types of primitive parallelohedra were found in dimension 6 up to now. It is expected that there are many more, with the total number estimated to be greater than 200,000,000.

From a software engineering point of view, Cones is a relatively simple single-threaded application written in C, with ASCII input and output files. The individual calculations easily run in parallel independently of each other. Therefore, Cones was selected as the first scientific application to port to the Seed Project Grid infrastructure.

Utilization: To make the application deployment easy and uniform, we refactored the source code, cleaned the hard-coded dependencies, and created configure and make files in compliance with the GNU configure and build system. Then Cones has been deployed and tested on the Seed Project gLite, ARC, and Condor pools. Within the ARC testbed, the integration of Cones has been straightforward. The main limiting factor is the destination cluster support rather than the middleware. So Cones now runs in preproduction mode on the ARC infrastructure.

With the Cones application, we were thus able to bring the first scientific user to SwiNG in the Seed Project phase. Peter Engel can now independently execute Cones calculations, controlled from the ARC user interface installed at UZH. Up to now, about 50,000 new combinatorial types of primitive parallelohedra were identified using SwiNG testbed resources. However, there is still a lot of room for improvement: missing in the described setup are a better distribution of Cones jobs in the ARC pool, a more stable environment for the job runs, a more user-friendly interface, the transfer to SLCS certificates, and the installation of additional resources to give more computing power to the application. While some of these issues are bound to the ARC middleware, some could be addressed in SwiNG working groups following the initial Seed Project.

4.2. GAMESS

Background: GAMESS (General Atomic and Molecular Electronic Structure System) [17] is an *ab initio* molecular quantum chemistry program package. It allows computing important features of molecular systems and reactions in gas phase and solution, such as electronic properties, energies, structures, and spectra. For this, a wide range of quantum chemical methods is provided, which are all based on approximations of the Schrödinger equation from quan-

tum mechanics. In its application domain, GAMESS represents a standard, free open source code, which is developed and used by many computational chemists around the world, and maintained at Iowa State University. The group of Kim Baldridge at UZH is one of the core developers and users.

GAMESS is a scientific application code with a long history, written mainly in Fortran 77 and C. It is available for many different hardware architectures and operating systems, most of them Unix based. The program predominantly consists of one main executable with shell script wrappers and helper files. Large parts of the package are well parallelized internally, using its own Distributed Data Interface (DDI) implementation. Individual GAMESS runs typically read a keyword-driven input file, and produce several, usually text-based output files. Another GAMESS use case scenario is parameter scans, where the scientist changes any one or a combination of the input keyword settings, as exemplified in [18].

Utilization: For the Seed Project, we used several test input files and scientific examples. The GAMESS package contains a set of more than 40 test cases, which cover all major parts of the code. In addition, Laura Zoppi (UZH) provided a small parameter scan example, based on density-functional theory (DFT) calculations of the corannulene molecule, and Kim Baldridge (UZH) gave 223 input files with Møller-Plesset (MP2) calculations of various organic molecules.

Regarding the status of GAMESS deployment on the SwiNG testbed infrastructure, the program is still in the test process on the ARC resources at UZH and CSCS, and on the ARC-enabled Condor pool at EPFL. For the latter, each job requires sending the executable with the input file, as described above. A port of GAMESS to XWCH is being developed by Afef Fkiri in collaboration with Nabil Abdennadher at HES-SO, and is close to completion. This will allow to perform selected parameter scans within the input files. Unfortunately, in the available time we were not able to have scientific users run GAMESS on the SwiNG testbed infrastructure. This could be one of the tasks for the working groups following the initial Seed Project.

4.3. PHYLIP

Background: PHYLIP (the PHYLogeny Inference Package) [19] consists of programs for inferring phylogenies (evolutionary trees). Developed during the 1980s, PHYLIP is one of the most widely-distributed phylogeny packages that has over 15,000 registered users. The package is free software, and has been ported to work on many different kinds of computer systems. Binaries and source code (in C) are available for Windows, Mac OS X, and Linux.

An evolutionary tree is composed of several branches. Each branch is composed of sub-branches and/or leaf nodes (sequences). To construct the tree, the application defines a "distance" among all pairs of biological sequences. The evolutionary tree is then gradually built by sticking to the same branch the pairs of sequences having the smallest distance between them. Even if the concept is simple, the algorithm is CPU intensive. This complexity is due to two factors: (1) The methods used to group sequences into branches are complex. As an example, the Fitch program, one of the most used methods, takes two hours to execute on a Pentium 4 (3 GHz) computer with 120 sequences. (2) The application constructs not only one tree from the original data set, but a set of trees generated from a large number of bootstrapped data sets (somewhere between 100 and 1000 is usually adequate). These data are randomly generated from the original data. The final (or consensus) tree is obtained by retaining groups that occur as often as possible.

All PHYLIP programs have the same structure: they read their input data from input files, process data, and write results to output files. Users execute PHYLIP programs by following a given chronology (workflow) which depends on their needs. This structure makes it easy to gridify PHYLIP. In this context, a gridification consists of: automatically executing a given workflow "constructed" by the end-user according to his/her needs, and distributing data among the nodes of the Grid platform following a Single Program Multiple Data (SPMD) model.

Utilization: Five modules were ported to the volunteer computing platform XWCH: Seqboot, Dnadist, Fitch-Margoliash, Neighbor-Joining, and Consensus [20, 21]. In order to apply the Fitch module to a large number of sequences, a parallel version of this package was designed and ported to XWCH. The gridified version of PHYLIP was used to generate an evolutionary tree related to HIV sequences and was executed on more than 200 nodes (Windows, Linux, and SunOS), located in two geographically distributed sites in Switzerland. A specific web service was developed in order to allow a dynamic configuration of the application regarding the current state of the platform (number and performance of the nodes) and the parameters of the application (number of sequences and number of bootstrapped data sets).

5. Conclusion

We have demonstrated the initialization of a national Grid collaboration in Switzerland within the limited resources of the SwiNG Seed Project: we successfully tested a new security tool that makes it easier for Swiss users to access the Grid (SLCS). We set up testbed resources for different kinds of middleware (gLite, ARC, XWCH, Condor), and an initial step towards Grid interoperability has been taken (Condor within the ARC pool). We

also explored the porting of different applications (Cones, GAMESS, PHYLIP) onto the corresponding infrastructure, bringing one of them (Cones) into a state where it can be used by a scientist.

This has only been possible since we took an approach based on a heterogeneous set of Grid middleware tools and scientific applications. Given that Grid computing comes with the notion of resource and *knowledge* sharing, we profited from the considerable interest, expertise, and resources in Grid computing that previously existed in Switzerland, but had been directed mostly towards projects with external partners. For initializing a technical collaboration such an approach turned out to be useful.

However, there also exist limitations of such an approach: based to a large extend on the investment of spare time of the participants, one can only reach a test-level setup. Even this level though required a considerable amount of work, constant communication, and active project management. To go beyond this stage towards a production setup, funding is definitively required. Since we were not restricted by the available computer resources, but by people's time, the investment needs to go mainly into having dedicated people.

Regarding the Grid middleware, our experience is similar to that of previous Grid projects: most of the middleware tools are still fairly demanding to install, maintain, and use, mainly due to their complexity and insufficient documentation. This applies to the system administrators' side, but even more to application developers and users. How application jobs are optimally distributed over a Grid and find the best matching resources is still not fully solved. Interoperability is possible, but tricky to achieve, in particular beyond one-to-one interfacing. There is a strong need for simplification and improvement, and for the agreement on standards, both on a technical level as well as regarding procedures, for example for user setup.

Our work also shows deficiencies on the side of the scientific applications. Application code is very diverse, and to run it on any Grid infrastructure requires direct cooperation among scientific developers and Grid experts. One reason for this is that standards for software writing and packaging are often ignored, which results in poor installation, documentation, and sometimes overall performance. This can be understood since scientists are seldom trained as software developers, and their main interest is the implementation of new scientific methods, not ease of deployment or usage. In order to run applications on the Grid, application developers need to provide well documented and packaged distributions that provide installation, configuration, testing and use procedures.

The Seed Project has officially ended in fall 2007, and the SwiNG association has been formally instantiated. The work described in this article will be continued by new working groups that are currently in the foundation state. It remains to be seen which Grid middleware, scientific applications, and services a future production infrastructure in Switzerland will offer to its users. Thanks to our efforts the capabilities of different middleware tools could be compared and a network of expertise to share this knowledge could be initialized, so that we will be able to provide a solid foundation for establishing a Swiss Grid infrastructure in the near future.

References

[1] Enabling Grids for E-sciencE, http://www.eu-egee.org, Nov. 2007.

[2] Pacific Rim Applications and Grid Middleware Assembly, http://www.pragma-grid.net, Nov. 2007.

[3] EGI, http://www.eu-egi.org, Sep. 2007.

[4] SwiNG, http://www.swing-grid.ch, Jan. 2008.

[5] H. Stockinger, "Defining the Grid: A Snapshot on the Current View", *Journal of Supercomputing*, vol. 42, no. 1, pp. 3–17, 2007.

[6] M. Podvinec, S. Maffioletti, P. Kunszt, K. Arnold, L. Cerutti, B. Nyffeler, R. Schlapbach, C. Türker, H. Stockinger, A. J. Thomas, M. C. Peitsch, and T. Schwede, "The SwissBioGrid Project: Objectives, Preliminary Results and Lessons Learned", *2nd IEEE International Conference on e-Science and Grid Computing (e-Science 2006) — Workshop on Running Production Grids*, Amsterdam, The Netherlands, Dec. 4–6, 2006.

[7] SLCS, http://www.switch.ch/grid/slcs/, Nov. 2007.

[8] AAI, http://www.switch.ch/aai/, Nov. 2007.

[9] gLite, http://www.glite.org, Nov. 2007.

[10] LCG, http://lcg.web.cern.ch/LCG/, Nov. 2007.

[11] Globus, http://www.globus.org, Nov. 2007.

[12] ARC, http://www.nordugrid.org/middleware/, Nov. 2007.

[13] XtremWeb-CH, http://www.xtremwebch.net, Nov. 2007.

[14] Condor, http://www.condorproject.org, Nov. 2007.

[15] Huygens, http://www.svi.nl, Nov. 2007.

[16] P. Engel, "The Combinatorial Types of Primitive Parallelohedra", University of Bern, 2007, unpublished.

[17] GAMESS, http://www.msg.chem.iastate.edu/gamess/, Nov. 2007.

[18] W. Sudholt, K. K. Baldridge, D. Abramson, C. Enticott, S. Garic, C. Kondric, and D. Nguyen, "Application of Grid Computing to Parameter Sweeps and Optimizations in Molecular Modeling", *Future Generation Computer Systems*, vol. 21, pp. 27–35, 2005.

[19] PHYLIP, http://evolution.genetics.washington.edu/phylip.html, Nov. 2007.

[20] N. Abendenadher and R. Boesch, "Porting PHYLIP Phylogenetic Package on the Desktop GRID Platform XtremWeb-CH", *HealthGrid'07*, Geneva, Switzerland, April 2007.

[21] N. Abendenadher and R. Boesch, "Deploying PHYLIP Phylogenetic Package on a Large Scale Distributed System", *CCGrid'07 — BioGrid'07 Workshop*, Rio de Janeiro, Brazil, May 2007.