Saleve: toolkit for developing parallel Grid applications

PÉTER DÓBÉ, RICHÁRD KÁPOLNAI, IMRE SZEBERÉNYI

Budapest University of Technology and Economics, Centre of Information Technology
{dobe,kapolnai,szebi}@iit.bme.hu

Reviewed

Keywords: Saleve, Grid application development, EGEE, parameter study

We present the Saleve tool, which helps the migration of existing parameter study applications into grid environment. Programs linked against the Saleve library can be integrated into grids using different middleware systems, so the application developer need not deal with the technical details of the middleware.

1. Introduction

In our days we have numerous resources available for running scientific computations, yet in many cases their usage is not encouraged by an adequate support. Although a large amount of development has been carried out, a researcher has many challenges to face in order to benefit from a distributed, parallel computational system.

Our intention is to ease these difficulties by presenting the *Saleve framework* which provides a virtual layer over the underlying infrastructures for the developer of a parallel application. Saleve focuses on implementing a specific type of parallel algorithms called *parameter study* programs. The parallel applications developed using Saleve can be executed on several different distributed infrastructures without any modification or recompilation.

In the next section we give a quick overview of the parameter study problems and of the EGEE which is the most important Grid project in the EU. Then we introduce the motivation and objective of Saleve, and we continue by outlining some details of the operation of the Saleve system. Finally we give a summary and present some future plans.

2. Utilizing the Grid for parameter study tasks

2.1. Parameter study tasks

In practice, there is frequently a need for an algorithm to be run with hundreds or even thousands of different input parameter values: such tasks are called parameter studies or parameter scans. In certain cases the requested result is the set of outputs obtained using each parameter, but the end result is often acquired via a final aggregating step. For instance, in an exhaustive optimization this last step is to seek one specific parameter value. Another simple example is the numeric integration of a non-analytic function over a given domain. We can split the domain into non-overlapping subdomains which will be used as the input parameter for the integrating routine, and as the final step we add up all the integral parts.

The problem of PS emerges in several experimental sciences, especially in physical simulations but also in other areas such as high energy physics, astrophysics, genetics, biomedical research and seismology. Like parameter studies, it is easy to split into subtasks all the engineering problems that can be described with ordinary differential equations. Such problems include statics tasks, like the research project at BME that involves the planning of reinforced concrete bridge beams using a parallelizable algorithm similar to PS [1].

The large number of executions, each with different parameters, would however take very long time when done sequentially. On the other hand, we should note that there is no causal relation among the runnings, so the chronological order of these is arbitrary, and they can even be done parallel according to the Single Program, Multiple Data (SPMD) model [2]. Therefore the presence of multiple CPUs can be taken advantage of, either in the form of a multiprocessing system or a cluster of several nodes – and in the best case, we can even utilize a grid infrastructure for this purpose.

2.2. The current state of Grids

In order to satisfy the rapidly increasing demand for computing and data storage, the plan for a geographically distributed network of resources called the grid [3] emerged more than a decade ago. Since then, several initiatives all over the world have been launched to implement it.

Among these, *Enabling Grids for E-sciencE* (EGEE) [4] is the largest in Europe. It was initially built to process data from the sensors of the Large Hadron Collider (LHC) at CERN, Switzerland, but now it has a wide range of scientific applications, for example in astrophysics, bioinformatics and geophysics. The project brings together more than 240 institutions from 45 countries including Hungary. Currently the grid consists of approximately 41,000 CPUs, can store up to 5 petabytes of data and can process 100,000 concurrent jobs.

BME participates in EGEE, too, by network activities and also by providing resources. In our own grid site called BMEGrid, there are currently 8 quad-core server machines which execute the jobs submitted into the grid. Connected to the site, we have a high capacity, efficient, parallel accessible storage system, the Scalable File Share (SFS), which is capable of storing nearly 3 terabytes of data. Our resources are mostly used by members of the Atlas HEP project and biomedical researchers.

Grid projects focus on facilitating the development of new applications, thus recruiting more grid users. For this purpose, portals [1,10], sometimes extended by development and workflow management tools [11], or other environments with complex functionalities [12] can provide a solution. The Saleve concept differs from these: using it, parallel applications can be created that are capable of transferring themselves to the runtime environment without separate tools, and besides staying lightweight, and can be run on a personal computer as well. The system, just like the aforementioned environments, is not specific to the application area.

3. Overview of Saleve

3.1. Motivation

Most of the researchers and engineers have programming skills, especially in C and Fortran languages, therefore creating sequential programs for performing scientific calculations means no obstacle to them. However, the development of distributed programs running in parallel requires more advanced knowledge and experience in programming.

The situation is made even more difficult by the large number of diverse, rapidly evolving technologies in use. These include different batch systems like PBS, LSF and Condor [5]. Although the final goal of grid development is a worldwide service that is accessible in a standardized way, its implementation is expected to delay several years yet. Currently each grid system has its own middleware system that is incompatible with the others. Getting acquainted with all these technologies and learning to use them in order to solve a general problem, for example PS, would take away effort from the real task of research. In addition, it is quite common that one would like to use an existing program without radical redesign in case the underlying computing infrastructure changes. Such change is for example switching from a single computer to a local cluster, or switching from the local cluster to the execution on a grid system.

3.2. A Proposed Solution

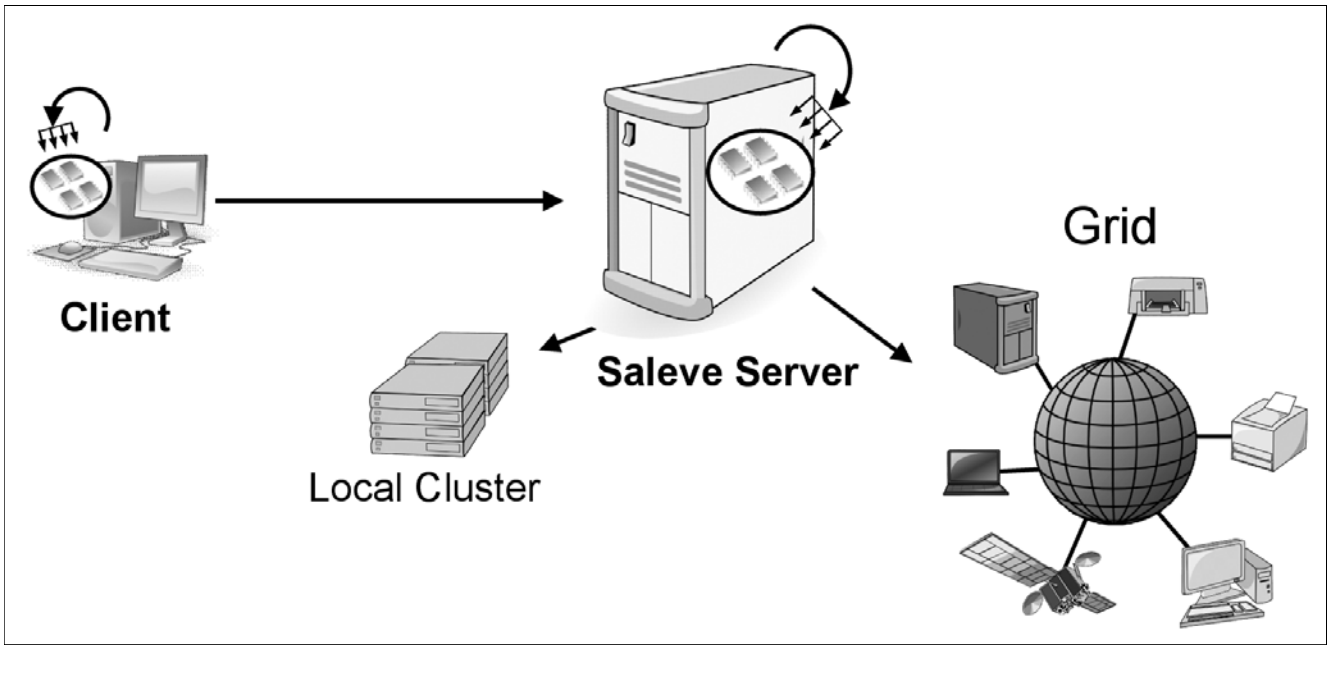
Handling these difficulties is the aim of the Saleve system, an open source tool to aid the development of C programs that are capable of running in parallel. Saleve can be used either to make new programs or to upgrade existing sequentially running ones. The main goal is to hide the details of the distinct computing technologies, and to provide an invariant, easy-to-learn methodology to create PS applications that, in addition to the simple sequential execution, are also able to take advantage of parallel computing systems.

4. Design of Saleve

4.1. Client-Server Architecture

To understand the operation of Saleve we begin with the derivation of the Saleve client. For that purpose, let us consider a traditional, sequential PS application writ-

Figure 1. The features of Saleve



ten in C. From the user's point of view, the only duty is that the original program has to be gently transformed into the Saleve client. First, a more exact structure of the program should be defined i.e. the following modules have to separated: the partition of the parameter space, the calculation over a subdomain and the summary of the subresults. Finally the resulting source code has to be linked against the Saleve programming library.

The client can be launched to compute the subresults over the subdomains and summarize them. By default, the running is similar to the original program: the subresults are computed consecutively, but the client can be configured to launch parallel jobs locally to finish faster in a multi-CPU machine. However, the most important feature of the client is the ability of transmitting itself and the input to a given Saleve server.

After receiving the executable client binary and its input data, the Saleve server forwards a new job for each subdomain to a Grid or to a cluster or simply executes it instead of dispatching. We emphasize that the user is not aware at all which distributed system the jobs were forwarded to, therefore Saleve provides full transparency *(Fig.1)*.

The server monitors the submitted jobs, resubmits them on failure and stores the temporary subresults. Under this phase the client may disconnect from the server and later another instance may resume the session from a different machine.

The server continuously returns the available subresults to connected client. As soon as each subresult has been returned, the client computes the final result from the subresults as the user defined.

Fig. 2. illustrates the course of events during the lifecycle of a task.

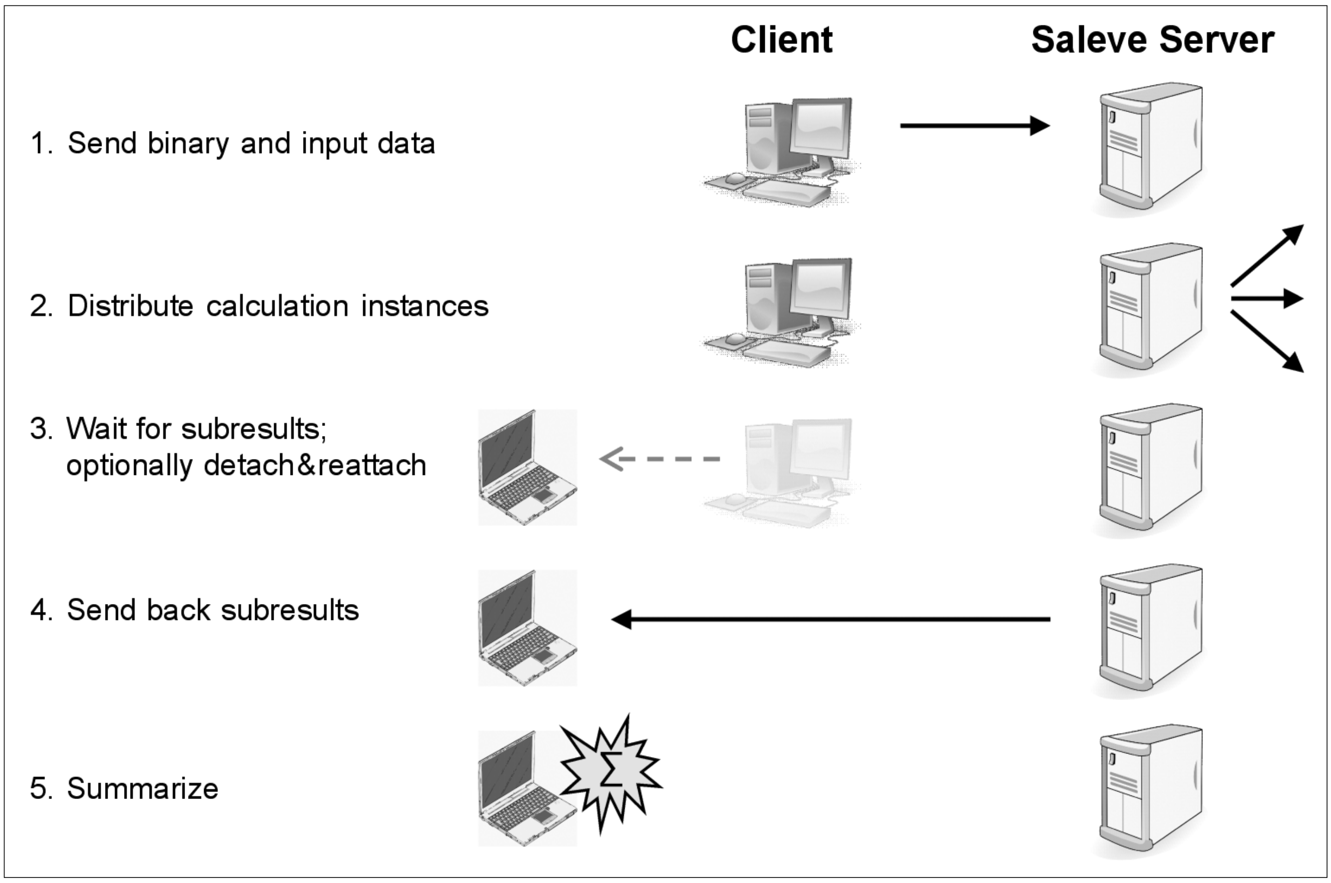
4.2. The Architecture of the Server

To meet our main requirements, the server should support the most popular distributed computing environments and, moreover, it should be easy to extend to operate with a new scheduler or grid middleware. This approach has led to the split-up of the server components into two groups: the components of the first group serve general purposes which are independent of any specific computing environment, the second group contains the components named the *plugins*.

One of the generic components implements the communication interface based on SOAP. The server provides web services towards the clients to upload a task and the input data and to download the subresults. The web services of Saleve are built on the gSoap [6] implementation. The generic group includes more components such as the one responsible for user management or job management which are discussed in detail in [7,8].

The plugins make it possible to the server to cooperate with several different infrastructures, in addition, adapting the server to a new infrastructure would not

Figure 2. Execution of a task in Saleve


Client
Saleve Server
1. Send binary and input data
2. Distribute calculation instances
3. Wait for subresults; optionally detach&reattach
4. Send back subresults
5. Summarize

involve any change in the generic components. For this reason, a dedicated plugin for every distributed system handles the envorinment-specific communication *(Fig. 3)*.

Up to present the following plugins have been made:

– executing jobs parallel on the server host,
– submitting jobs to the Condor scheduler [5] which is widely spread on cluster sites,
– forwarding jobs to the EGEE grid infrastructure through the gLite middleware.

4.3. Interoperation between the EGEE infrastructure and Saleve

Saleve supports submitting tasks to the EGEE grid with the help of the gLite plugin mentioned above. Developing a new Saleve plugin principally requires knowledge of the interface of the corresponding middleware or scheduler, does not presume deep experience in Saleve internals.

To create a new plugin, one has to implement an interface where the data management is aided by the Saleve development library. The major challenge is dealing with the authentication issues towards the grid and the job management rather than using the Saleve library. Looking after the grid jobs is essential due to the instability of the current infrastructures: some jobs may abort and must be submitted again.

The gLite middleware which is the software engine of the EGEE infrastructure has adopted certificate-based authentication and resource-allocation methods where the permissions of a user are determined by his or her memberships in virtual organisations (VOs). When a user wishes to access to a resource e.g. by submitting a task, a temporary, short range proxy certificate has to be generated using the long range certificate, has to be attached to the submitted task and periodically renewed. This procedure is useful to protect the long range certificate if the proxy certification was compromised.

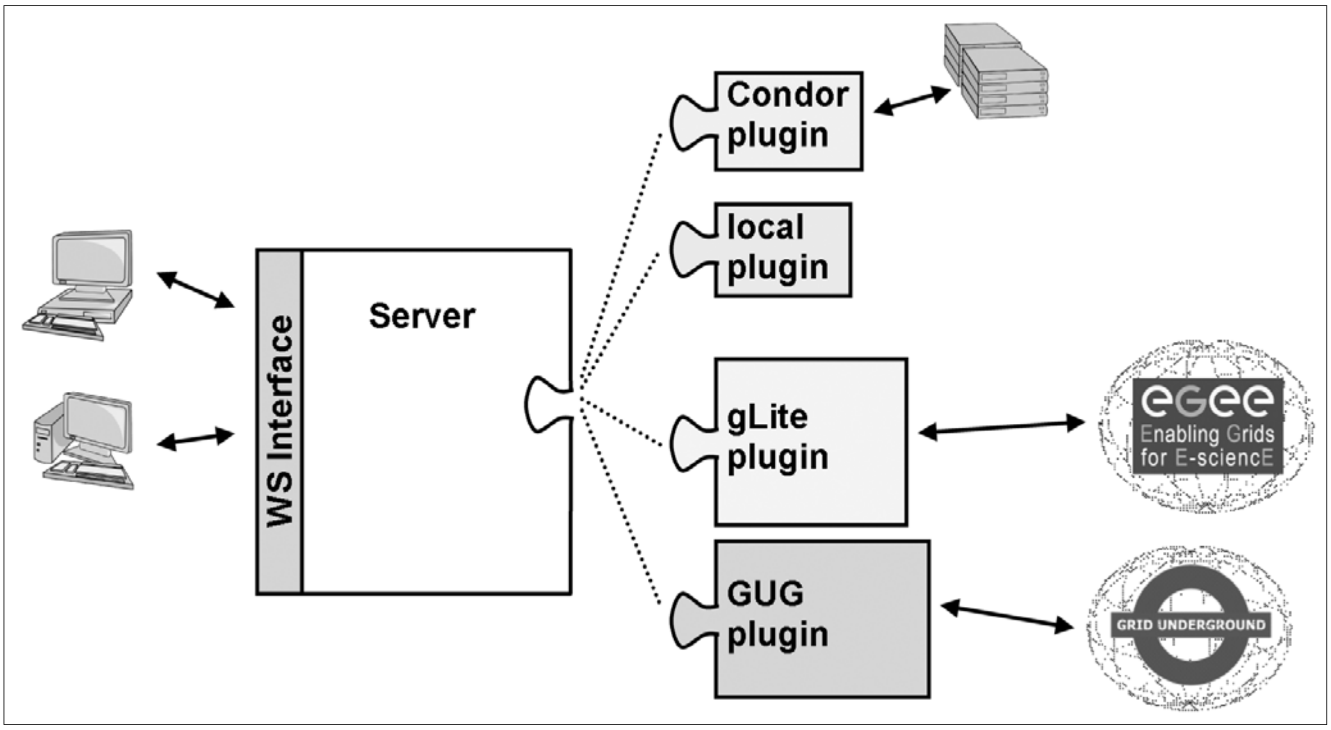
In our current implementation the Saleve server keeps an own certificate for accessing grid resources directly thus the server is a member of some virtual organisation. In this manner the proxy generation and renewal is completely hidden from the user who cannot tell whether the task has been executed in the EGEE grid or in a local cluster.

5. Summary

The presented Saleve system aids the development of parameter study type parallel applications by forming a transparent abstraction layer above the middleware or batch systems of the different distributed environments. Its main advantage is that a slightly modified version of the legacy application called the Saleve client can be run without change in several types of runtime environment and even on the local machine, so it is easy to develop applications possibly for EGEE, the largest infrastructure of Europe, without knowing the technical details.

Regarding the possible upgrades of the system, we are going to focus on implementing the dynamic loading and unloading of plugins and a better management of jobs submitted into the grid. Our plans also include improving the flexibility of client-server communication with the help of webstreams. For more information on the current status of the Saleve project, please visit the web page [9].

Figure 3. The architecture of Saleve



Acknowledgements

Part of this work was funded by the Péter Pázmány program (RET-06/2005) of the National Office for Research and Technology, and the authors would like to thank EGEE project (EU INFSO-RI-031688) and NKFP MEGA (2_009_04) project.

References

[1] D. Pasztuhov, A. Sipos and I. Szeberényi: Calculating Spatial Deformations of Reinforced Concrete Bars Using Grid Systems, MIPRO 2007 – Hypermedia and Grid Systems, Opatija, Croatia, 2007. pp.189–194.

[2] P. E. Black: Algorithms and Theory of Computation Handbook, CRC Press LLC, U.S. National Institute of Standards and Technology, 1999.

[3] I. Foster and C. Kesselman: The Grid 2: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers Inc., 2003.

[4] EGEE-II Information Sheet, 2007. http://www.eu-egee.org/sheets/uk/egee-ii.pdf

[5] D. Thain, T. Tannenbaum and M. Livny: Distributed Computing in Practice: the Condor Experience, Concurrency and Computation: Practice and Experience, 2005. pp.323–356.

[6] R. A. van Engelen and K. Gallivan: The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks, In the Proc. of the 2nd IEEE Int. Symposium on Cluster Computing and the Grid (CCGrid'02), Berlin, Germany, 2002. pp.128–135.

[7] Zs. Molnár and I. Szeberényi: Saleve: simple web-services based environment for parameter study applications. In Proc. of the 6th IEEE/ACM International Workshop on Grid Computing, 2005. pp.292–295.

[8] P. Dóbé, R. Kápolnai and I. Szeberényi: Simple grid access for parameter study applications. In 6th International Conference on Large-Scale Scientific Computations, Sozopol, 2007. (in press)

[9] Saleve project, http://egee.ik.bme.hu/saleve/

[10] P. Kacsuk, Z. Farkas and G. Hermann: Workflow-level parameter study support for production Grids, Proc. of ICCSA'2007, Kuala Lumpur, Springer LNCS 4707, pp.872–885.

[11] P. Kacsuk, G. Dózsa, J. Kovács, R. Lovas, N. Podhorszki, Z. Balaton and G. Gombás: P-GRADE: A Grid Programming Environment. Journal of Grid Computing, Vol. 1, No. 2, 2004. pp.171–197.

[12] T. Fahringer, A. Jugravu, S. Pllana, R. Prodan, C. Seragiotto Jr. and H.-L. Truong: Askalon: a tool set for cluster and Grid computing, Concurrency and Computation: Practice and Experience, Vol. 17, 2005. pp.143–169.

Author

Richárd Kápolnai received his M.Sc. degree in technical informatics from Budapest University of Technology and Economics (BME), Hungary, in 2006. His M.Sc. thesis is about designing networks for selfish users. Currently he is pursuing a Ph.D. degree at the Centre of Information Technology, BME, in grid systems. He helped develop the Saleve system, and participated in the 2nd phase in the project EGEE. His research interests include supporting grid application development, and mechanism design.