Gridification of a Radiotherapy Dose Computation Application with the XtremWeb-CH Environment

Nabil Abdennadher[1], Mohamed Ben Belgacem[1], Raphaël Couturier[2], David Laiymani[2], Sébastien Miquée[2], Marko Niinimaki[1], and Marc Sauget[3]

[1] University of Applied Sciences Western Switzerland, hepia Geneva, Switzerland
nabil.abdennadher@hesge.ch,mohamed.benbelgacem@unige.ch,markopekka.niinimaeki@hesge.ch
[2] Laboratoire d'Informatique de l'université de Franche-Comté
IUT Belfort-Montbéliard, Rue Engel Gros, 90016 Belfort - France
{raphael.couturier,david.laiymani,sebastien.miquee}@univ-fcomte.fr
[3] FEMTO-ST, ENISYS/IRMA, F-25210 Montbéliard , FRANCE
marc.sauget@univ-fcomte.fr

Abstract. This paper presents the design and the evaluation of the gridification of a radiotherapy dose computation application. Due to the inherent characteristics of the application and its execution, we choose the architectural context of volunteer computing. For this, we used the XtremWeb-CH environment. Experiments were conducted on a real volunteer computing testbed and show good speed-ups and very acceptable platform overhead, letting XtremWeb-CH be a good candidate for deploying parallel applications over a volunteer computing environment.

1 Introduction

The use of distributed architectures for solving large scientific problems seems to become mandatory in a lot of cases. For example, in the domain of radiotherapy dose computation the problem is crucial. The main goal of external beam radiotherapy is the treatment of tumors while minimizing exposure to healthy tissue. Dosimetric planning has to be carried out in order to optimize the dose distribution within the patient. Thus, to determine the most accurate dose distribution during treatment planning, a compromise must be found between the precision and the speed of calculation. Current techniques, using analytic methods, models and databases, are rapid but lack precision. Enhanced precision can be achieved by using calculation codes based, for example, on the Monte Carlo methods. The main drawback of these methods is their computation times which can rapidly become huge. In [18] the authors proposed a new approach, called Neurad, using neural networks. This approach is based on the collaboration of computation codes and multi-layer neural networks used as universal approximators. It provides a fast and accurate evaluation of radiation doses in any given environment for given irradiation parameters. As the learning step is often very time consuming, in [5] the authors proposed a parallel algorithm that enables to decompose

the learning domain into subdomains. The decomposition has the advantage of significantly reducing the complexity of the target functions to approximate.

Now, as there exist several classes of distributed/parallel architectures (supercomputers, clusters, global computing...) we have to choose the best suited one for the parallel Neurad application. The volunteer (or global) computing model seems to be an interesting approach. Here, the computing power is obtained by aggregating unused (or volunteer) public resources connected to the Internet. In our case, we can imagine, for example, that a part of the architecture will be composed of some of the different computers of the hospital. This approach presents the advantage of being clearly cheaper than a more dedicated approach like the use of supercomputers or clusters. Furthermore and as we will see in the remainder, the studied parallel algorithm corresponds very well to this computation model.

The aim of this paper is to propose and evaluate a gridification of the Neurad application (more precisely, of the most time consuming part, the learning step) using a volunteer computing approach. For this, we focus on the XtremWeb-CH environment[2]. We chose this environment because it tackles the centralized aspect of other global computing environments such as XtremWeb[11] or Seti[1]. It tends to a peer-to-peer approach by distributing some components of the architecture. For instance, the computing nodes are allowed to directly communicate. Experiments were conducted on a real global computing testbed. The results are very encouraging. They exhibit an interesting speed-up and show that the overhead induced by the use of XtremWeb-CH is very acceptable.

The paper is organized as follows. In Section 2 we present the Neurad application and particularly its most time consuming part, i.e. the learning step. Section 3 details the XtremWeb-CH environment and Section 4 exposes the gridification of the Neurad application. Experimental results are presented in Section 5 and we end in Section 6 by some concluding remarks and perspectives.

2 The Neurad application

The *Neurad* [4] project presented in this paper takes place in a multi-disciplinary project, involving medical physicists and computer scientists whose goal is to enhance the treatment planning of cancerous tumors by external radiotherapy. In our previous works [14, 16, 18], we have proposed an original approach to solving scientific problems whose accurate modeling and/or analytical description are difficult. That method is based on the collaboration of computational codes and neural networks used as universal interpolator. Thanks to that method, the *Neurad* software provides a fast and accurate evaluation of radiation doses in any given environment (possibly inhomogeneous) for given irradiation parameters. We have shown in a previous work ([5]) the interest of using a distributed algorithm for the neural network learning. We use a classical RPROP [4] algorithm with a HPU [5] topology to do the training of our neural network.

[4] Resilient backpropagation
[5] High order processing units

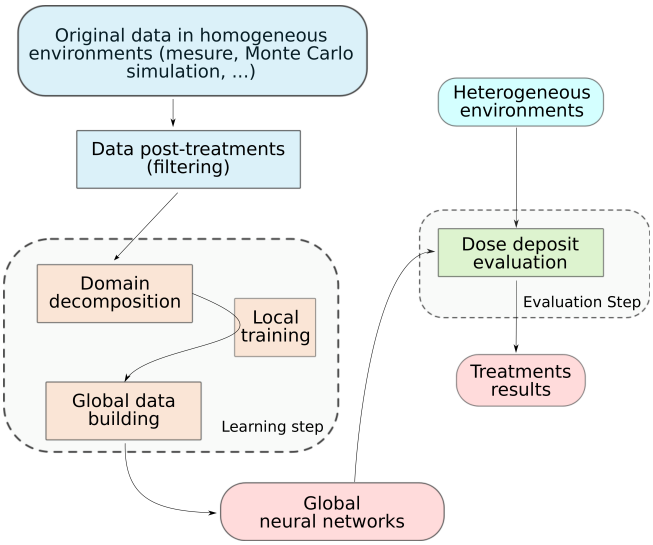

Fig. 1. The Neurad project

Figure 1 presents the *Neurad* scheme. Three parts are clearly independent: the initial data production, the learning process and the dose deposit evaluation. The first step, the data production, is outside of the *Neurad* project. They are many solutions to obtain data about the radiotherapy treatments like the measure or the simulation. The only essential criterion is that the result must be obtained in an homogeneous environment.

The secondary stage of the *Neurad* project is the learning step and this is the most time consuming step. This step is performed offline but it is important to reduce the time used for the learning process to keep a workable tool. Indeed, if the learning time is too huge (for the moment, this time could reach one week for a limited domain), this process should not be launched at any time, but only when a major modification occurs in the environment, like a change of context for instance. However, it is interesting to update the knowledge of the neural network, by using the learning process, when the domain evolves (evolution in material used for the prosthesis or evolution on the beam (size, shape or energy)). The learning time is related to the volume of data which could be very important in a real medical context. Some work has been done to reduce this learning time with the parallelization of the learning process by using a partitioning method of the global dataset. The goal of this method is to train many neural networks on sub-domains of the global dataset. After this training, the use of these neural networks all together allows to obtain a response for the global domain of study.

However, performing the learning on sub-domains constituting a partition of the initial domain may not be satisfying depending on the chosen quality of the results. This comes from the fact that the accuracy of the approximation performed by a neural network is not constant over the learned domain. Thus,

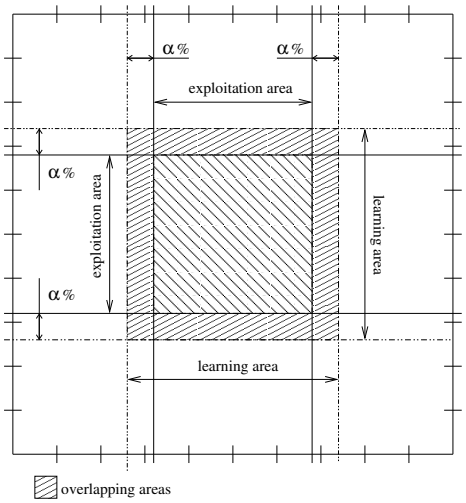

Fig. 2. Overlapping for a sub-network in a two-dimensional domain with ratio α

it is necessary to use an overlapping of the sub-domains. The overall principle is depicted in Figure 2. In this way, each sub-network has an exploitation domain smaller than its training domain and the differences observed at the borders are no longer relevant. Nonetheless, in order to preserve the performance of the parallel algorithm, it is important to carefully set the overlapping ratio α. It must both be large enough to avoid the border's errors, and as small as possible to limit the size increase of the data subsets [5].

3 The XtremWeb-CH environment

High performance computing environments like MPI (Message Passing Interface) [12] are widely used and have proved their efficiency. This class of systems are very tightly coupled and powerful but not very error tolerant. Cluster computing environments like Condor [17] and volunteer computing systems like BOINC [3] are loosely coupled and have a scheduler that distributes tasks to computing nodes. Cluster computing environments assume the fact that nodes are in general directly accessible, one to another, but this does not apply to volunteer computing systems.

XtremWeb-CH (XWCH) is a volunteer computing inspired large-scale computing platform for distributed applications. In fact, it tends to be a good compromise between cluster computing and volunteer computing. It is originaly based on another platform called XtremWeb [11]. It is easy to install, maintain, and it is supported by a Grid middleware (ARC [10]) and a workflow engine (JOpera [9]). It consists in three components: one coordinator, a set of workers, and at least one warehouse. Client programs use these components.

The coordinator is the main component of the XWCH platform. It controls user access and schedules jobs to workers. It provides a web interface for managing jobs and users, and a set of web services. These are user services and worker/warehouse services implemented using WSDL (Web Service Description Language) [8], that simplifies client development for languages that support it (and most popular programming languages do).

A worker is a Java daemon that runs on the user machine. Assumed to be volatile, the workers periodically report themselves to the coordinator, accept jobs, retrieve input, compute jobs, and store the results of the computation on warehouses. If the coordinator does not receive a signal from a worker, it will simply remove it from the scheduling list, and if a job had been assigned to that worker, it will be re-assigned to another one. A schema of the architecture is shown in Figure 3.

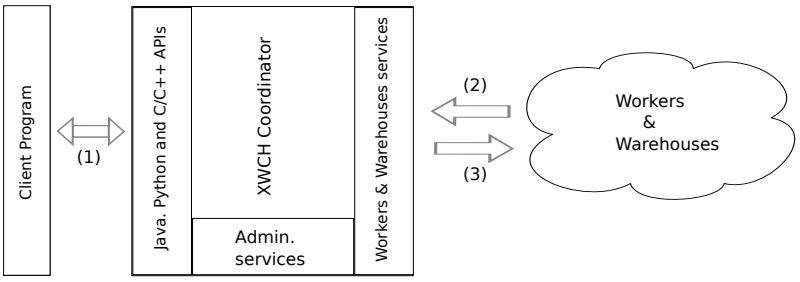

Fig. 3. The XtremWeb-CH architecture

A warehouse is a file server that acts as a data storage system for workers and client programs. Workers may not necessarily be able to communicate directly with each others, due to firewalls and NAT sub-networks. For these reasons, warehouses are used as intermediaries to exchange, store and retrieve data.

Job submission is done by a client program which is written using a flexible API, available for Java and C/C++ programs. The client program runs on a "client node" and calls the user services to submit jobs (Figure 3, (1)). The main flexibility provided by the use of this architecture is to control and dynamically generate jobs especially when their number cannot be known in advance. Communications between the coordinator and the workers are always initiated by the workers following a pull model (Figure 3, (2)):

- Workers receive jobs (Figure 3, (3)) only if they send a "work request" signal;
- When a worker finishes its job, it stores its output file on a warehouse and sends a "work result" signal to the coordinator;
- During its execution, a worker (respectively warehouse) periodically sends "work alive" to the worker service (respectively warehouse service) to report itself to the coordinator.

As a whole, XWCH is easy to install, maintain and use. Its components are programmed mainly using Java, and their process memory sizes in a typical 32-bit GNU/Linux computer are:

- Coordinator 190 MB including the Glassfish Java container;
- Worker 40 MB;
- Warehouse 80 MB.

Experiments presented in [15] show that the performance of XWCH is comparable with Condor [13], another non-intrusive computing system that has similar functionalities but is somewhat more difficult to install.

The main characteristics of the new version of XWCH, compared to previous ones, are: dynamic job generation, flexible data sharing (data replication) and persistent jobs. These features are presented in [7] and will not be detailed in this paper.

4 The Neurad gridification

As previously exposed, the Neurad application can be divided into three steps. The goal of the first step is to decompose the data representing the dose distribution on an area. This area contains various parameters, like the nature of the medium and its density. This part is out of the scope of this paper.

The second step of the application, and the most time consuming, is the learning in itself. This is the one which has been parallelized, using the XWCH environment. As exposed in section 2, the parallelization relies on a partitioning of the global dataset. Following this partitioning all learning tasks are independently executed in parallel with their own local data part, with no communication, following the fork/join model. Clearly, this computation fits well with the model of the chosen middleware.

The execution scheme is then the following (see Figure 4):

1. We first send the learning application and its data to the middleware. In a first time, we send the application to data warehouses (DW), and the create an "application module" on the coordinator (Coord.) including references retrieved from the previous sending operation. In a second time, we apply the same process to application data.
2. When a worker (W) is ready to compute, it requests a task to execute to the coordinator (Coord.);
3. The coordinator assigns the worker a task. This last one retrieves the application and its assigned data, by requesting them to DW with references sent by the coordinator, and so can start the computation;
4. At the end of the learning process, the worker sends the result to a warehouse.

The last step of the application is to retrieve these results (some weighted neural networks) and exploit them through a dose distribution process. This last step is out of the scope of this paper.

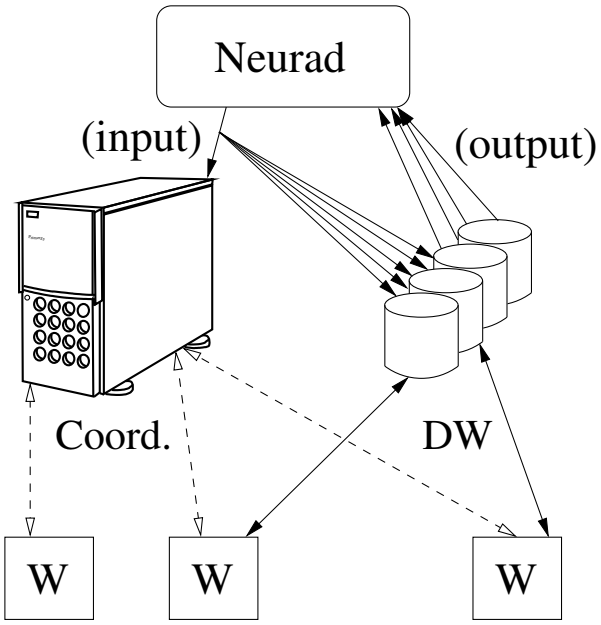

Fig. 4. The proposed Neurad gridification

5 Experimental results

The aim of this section is to describe and analyze the experimental results we have obtained with the parallel Neurad version previously described. Our goal was to carry out this application with real input data and on a real volunteer computing testbed.

Experimental conditions The size of the input data is about 2.4Gb. In order to avoid noises to appear and disturb the learning process, these data can be divided into, at most, 25 parts. This generates input data parts of about 15Mb (in a compressed format). The output data, which are retrieved after the process, are about 30Kb for each part. We used two distinct deployments of XWCH:

1. In the first one, called "distributed XWCH", the XWCH coordinator and the warehouses were located in Geneva, Switzerland while the workers were running in the same local cluster in Belfort, France.
2. The second deployment, called "local XWCH" is a local deployment where coordinator, warehouses and workers were, in the same local cluster, at the same time.

For both deployments, the local cluster is a campus cluster and during the day these machines were used by students of the Computer Science Department of

the IUT of Belfort. Unfortunately, the data decomposition limitation does not allow us to use more than 25 computers (XWCH workers).

In order to evaluate the overhead induced by the use of the platform we have furthermore compared the execution of the Neurad application with and without the XWCH platform. For the latter case, we want to insist on the fact that the testbed consists only in workers deployed with their respective data by the use of shell scripts. No specific middleware was used and the workers were in the same local cluster.

Finally, five computation precisions were used: $1e^{-1}$, $0.75e^{-1}$, $0.50e^{-1}$, $0.25e^{-1}$, and $1e^{-2}$.

Results Table 1 presents the execution times of the Neurad application on 25 machines with XWCH (local and distributed deployment) and without XWCH. These results correspond to the measures of the same steps for both kinds of execution, i.e. the sending of local data and the executable, the learning process, and retrieving the results. Results represent the average time of 5 executions.

Precision	1 machine	Without XWCH	With XWCH	With local XWCH
$1e^{-1}$	5190	558	759	629
$0.75e^{-1}$	6307	792	1298	801
$0.50e^{-1}$	7487	792	1010	844
$0.25e^{-1}$	7787	791	1000	852
$1e^{-2}$	11030	1035	1447	1108

Table 1. Execution time in seconds of the Neurad application, with and without using the XWCH platform

As we can see, in the case of a local deployment the overhead induced by the use of the XWCH platform is about 7%. It is clearly a low overhead. Now, for the distributed deployment, the overhead is about 34%. Regarding the benefits of the platform, it is a very acceptable overhead which can be explained by the following points.

First, we point out that the conditions of executions are not really identical between, with and without, XWCH contexts. For this last one, though the same steps were achieved out, all transfer processes are inside a local cluster with a high bandwidth and a low latency. Whereas when using XWCH, all transfer processes (between datawarehouses, workers, and the coordinator) used a wide network area with a smaller bandwidth. In addition, in the executions without XWCH, all the machines started immediately the computation, whereas when

using the XWCH platform, a latency is introduced by the fact that a computation starts on a machine, only when this one requests a task.

This underlines that, unsurprisingly, deploying a local coordinator and one or more warehouses near a cluster of workers can enhance computations and platform performances.

6 Conclusion and future works

In this paper, we have presented a gridification of a real medical application, the Neurad application. This radiotherapy application tries to optimize the irradiated dose distribution within a patient. Based on a multi-layer neural network, this application presents a very time consuming step, i.e. the learning step. Due to the computing characteristics of this step, we have chosen to parallelize it using the XtremWeb-CH volunteer computing environment. Obtained experimental results show good speed-ups and underline that overheads induced by XWCH are very acceptable, letting it be a good candidate for deploying parallel applications over a volunteer computing environment.

Our future works include the testing of the application on a larger scale testbed. This implies, the choice of a data input set allowing a finer decomposition. Unfortunately, this choice of input data is not trivial and relies on a large number of parameters.

We are also planning to test XWCH with parallel applications where communication between workers occurs during the execution. In this way, the use of the asynchronous iteration model [6] may be an interesting perspective.

References

1. Seti@home. `http://setiathome.ssl.berkeley.edu`.
2. N. Abdennadher and R. Boesch. Towards a peer-to-peer platform for high performance computing. In Christophe Cérin and Kuan-Ching Li, editors, *Advances in Grid and Pervasive Computing*, volume 4459 of *Lecture Notes in Computer Science*, pages 412–423. Springer Berlin / Heidelberg, 2007.
3. David P. Anderson and Gilles Fedak. The computational and storage potential of volunteer computing. In *IEEE/ACM International Symposium on Cluster Computing and the Grid*. IEEE Press, May 2006.
4. J. M. Bahi, S. Contassot-Vivier, L. Makovicka, E. Martin, and M. Sauget. Neurad. *Agence pour la Protection des Programmes. No: IDDN.FR.001.130035.000.S.P.2006.000.10000*, 2006.
5. J. M. Bahi, S. Contassot-Vivier, and M. Sauget. An incremental learning algorithm for functional approximation. *Advances in Engineering Software*, 40(8):725–730, 2009. doi:10.1016/j.advengsoft.2008.12.018.
6. J. M. Bahi, R. Couturier, and D. Laiymani. Comparison of conjugate gradient and multisplitting algorithms of nas benchmark with the jace environment. In *IPDPS 2008*. IEEE Computer Society Press, April 2008.
7. M. Ben Belgacem, N. Abdennadher, and M. Niinimaki. Virtual ez grid: A volunteer computing infrastructure for scientific medical applications. In *5th international conference, GPC 2010*. Berlin: Springer, May 10-13 2010.

8. E. Cerami. *Web Services Essentials – Distributed Applications with XML-RPC, SOAP, UDDI and WSDL*. O'Reilly, 2002.
9. Gustavo Alonso Cesare Pautasso. Jopera: a toolkit for efficient visual composition of web services. *International Journal of Electronic Commerce (IJEC)*, 2005.
10. M. Ellert, M. Gronager, A. Konstantinov, B. Konya, J. Lindemann, I. Livenson, J.L. Nielsen, M. Niinimaki, O. Smirnova, and A. Waananen. Advanced resource connector middleware for lightweight computational grids. *Future Generation Computer Systems*, 23(2):219 – 240, 2007.
11. G. Fedak, C. Germain, V. Neri, and F. Cappello. Xtremweb: A generic global computing system. *Cluster Computing and the Grid, IEEE International Symposium on*, 0:582, 2001.
12. W. Gropp, E. Lusk, and A. Skjellum. *Using MPI : portable parallel programming with the message passing interface*. MIT Press, 1994.
13. M. J Litzkow, M. Livny, and M. W. Mutka. Condor-a hunter of idle workstations. In *Condor-a hunter of idle workstations*. IEEE, 1988.
14. L. Makovicka, A. Vasseur, M. Sauget, E. Martin, R. Gschwind, J. Henriet, and M. Salomon. Avenir des nouveaux concepts des calculs dosimétriques basés sur les méthodes de Monte Carlo. *Radioprotection*, 44(1):77–88, jan 2009.
15. M. Niinimaki, M. Ben Belcagem, and N. Abdennadher. Xwch-ccgrid. Technical report, 2011.
16. M. Sauget, R. Laurent, J. Henriet, M. Salomon, R. Gschwind, S. Contassot-Vivier, L. Makovicka, and C. Soussen. Efficient domain decomposition for a neural network learning algorithm, used for the dose evaluation in external radiotherapy. In Konstantinos Diamantaras, Wlodek Duch, and Lazaros S. Iliadis, editors, *Artificial Neural Networks – ICANN 2010*, volume 6352 of *Lecture Notes in Computer Science*, pages 261–266. Springer, 2010.
17. Douglas Thain, Todd Tannenbaum, and Miron Livny. Distributed computing in practice: the condor experience. *Concurrency - Practice and Experience*, 17(2-4):323–356, 2005.
18. A. Vasseur, L. Makovicka, E. Martin, M. Sauget, S. Contassot-Vivier, and J. M. Bahi. Dose calculations using artificial neural networks: a feasibility study for photon beams. *Nucl. Instr. and Meth. in Phys. Res. B*, 266(7):1085–1093, 2008.