2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing

HPC Applications deployment on distributed heterogeneous computing platforms via OMF, OML and P2PDC

Didier El Baz*†, The Tung Nguyen*†, Guillaume Jourjon‡, and Thierry Rakotoarivelo‡
*CNRS ; LAAS; 7 avenue du colonel Roche, Toulouse, France. Email: {ttnguyen,elbaz}@laas.fr
†Université de Toulouse; Toulouse France.
‡ NICTA, Australian Technology Park, Eveleigh, NSW Australia. Email: firstname.lastname@nicta.com.au

***Abstract*—A new tool and web portal are presented for deployment of High Performance Computing applications on distributed heterogeneous computing platforms. This tool relies on the decentralized environment P2PDC and the OMF and OML multithreaded control, instrumentation and measurement libraries. Deployment on PlanetLab of a numerical simulation application is studied. A first series of computational results is displayed and analyzed.**

***Index Terms*—component; HPC; heterogeneous computing; peer-to-peer computing; application deployment; PlanetLab; numerical simulation; asynchronous iterations**

I. Introduction

The domains of parallel and distributed computing are currently undergoing profound mutations. New concepts like peer-to-peer, global computing and cloud computing have recently emerged (see [1]). More recently, heterogeneous technologies have started to be used with success for High Performance Computing (HPC) applications as well (see [2] and [3]). In particular, the combination of distributed and heterogeneous technologies seems to be very promising for HPC applications. These technologies have led to architectures with hundred thousands or millions of cores where application deployment, heterogeneity, power consumption and fault-tolerance are key issues.

In this paper, we present the principle of an original solution related to a web portal for peer-to-peer HPC application deployment. The Portal is the combination of the decentralized environment for high performance Peer-to-Peer Distributed Computing, P2PDC, (see [4] and [5]) with tools like OML, OMF and OMF Portal that facilitate the deployment, management of P2PDC applications as well as the retrieval and analysis of results [6]–[9]. We introduce also in this paper a new measurement channel for P2PDC on OML.

Section II deals with related work and background material. The motivations of the study are presented in Section III. We detail measurement channel for P2PDC and task deployment in Section IV. Experiments carried out on PlanetLab are presented in Section V. Finally, conclusions and future work are presented in Section VI.

II. Background and Related Work

In this section, we briefly present the three component on which this current work is built, namely the decentralized environment P2PDC, the OMF framework and the OML library.

A. P2PDC

The P2PDC framework [4], [5] is a decentralised environment for high performance peer-to-peer distributed computing. This framework is specialised in task parallel applications and in particular P2PDC is intended for the resolution of numerical simulation and optimization problems via distributed iterative methods that lead to direct and frequent data exchanges between machines [4], [10]–[12]. The P2PDC communication process relies on the use of the Peer-to-Peer Self Adaptive communication Protocol, (P2PSAP) [13], [14], in which a reduced set of communication calls (*e.g.*, P2Psend, P2Preceive and P2Pwait) are used in order to facilitate programming. Thus, the experimenter can concentrate on the choice of distributed iterative computational schemes (*e.g.*, synchronous or asynchronous) he wants to be implemented and does not need to focus on the communication mode between any two peers. In addition to the two generic class of algorithms (*e.g.*, synchronous or asynchronous), P2PDC also offers a hybrid iterative computational scheme, whereby computations are locally synchronous in a subset of peers organised in a cluster and asynchronous between clusters. This hybrid mode is made possible as P2PDC decentralized environment is based on a hybrid topology manager and a hierarchical task allocation mechanism. Further details are available in [10] and [15].

As described previously, the P2PDC framework uses of the P2PSAP communication protocol [13], [14]. In addition to the basic API calls we have explained above, this protocol also dynamically chooses the most appropriate communication mode between any two peers in the system in accordance with both application requirements (*i.e.*, the computational scheme class) and underlying measurements of the network. Thus, in the hybrid case, a synchronous communication mode is implemented between machines in a group/cluster that are relatively close and presenting similar characteristics, while an asynchronous communication mode is implemented between machines in different groups/cluster (*i.e.*, no guarantee of reliability).

The P2PSAP communication protocol was originally designed as an extension of the CTP transport protocol [16] implemented in the CACTUS framework which makes use of micro-protocols [17]. The CTP protocol includes a wide range of micro-protocols including a small set of basic micro-protocols that are needed in every configuration and a set of micro-protocols implementing various transport properties. Recently, the P2PSAP communication protocol takes into account Ethernet and Infiniband clusters [14].

Finally, reference are also made to [1] and [18] for more details on peer-to-peer computing.

B. OMF

In order to evaluate new networking technologies, researchers have developed and deployed large facilities like PlanetLab complementarily to preliminary simulated results. These platforms aim at providing real conditions for testing research works while proposing repeatability in a semi-closed environment. Nevertheless, offering and performing repeatability requires the development of management frameworks. During the last decade, the cOntrol and Management Framework (OMF) [7] has been developed to tackle this difficult challenge. This framework offers a suite of management, control and measurement services for networking platforms.

Researchers have built large experimental facilities (aka testbeds) such as Planetlab to evaluate new networking technologies at scale. Such an evaluation with real conditions complements any preliminary simulated results, and similarly needs to be repeatable within the semi-closed environment provided by the testbed. Providing such systematic repeatability is challenging when the evaluation involves testbeds with large number of heterogeneous resources. The cOntrol and Management Framework (OMF) [7] offers a software suite to control and manage resources in networking testbeds, and to define and orchestrate experiments using these resources. OMF provides testbed operators with several services to provision and configure various resource types (e.g. virtual machines, wireless devices, etc...). From a researcher's point of view, it provides a high level domain-specific language (OEDL) to describe an experiment and a set of tools to automatically deploy and orchestrate it on a given testbed.

An OMF experiment starts with the definition of an Experiment Description using OEDL. It describes the resources to use, the measurements to collect, and the tasks to perform during the experiment run. The researcher passes this description to the OMF system which performs all the required operations to deploy, configure and execute the different elements and steps of the experiment. During the experiment run, if the researcher requested some measurements from any OML-instrumented resources [6], then the corresponding measurement streams will be automatically created and collected.

This framework is currently used on many testbeds worldwide, and has been integrated with other research and educational tools. For example, it is used as the experiment orchestration system for the IREEL e-learning platform [9], which allows students to perform classroom lab experiments on networking testbeds. OMF has also been integrated to the LabWiki web-based portal [8], which aims at providing researchers with a software replacement for laboratory notebooks. Labwiki provides unified tools to define research objectives, plan corresponding experiments, deploy and execute them (using OMF), analyze the collected data (using an R interpreter[1]), and present and share the results with peer researchers.

C. OML

OML [6] is a multithreaded instrumentation and data collection library. It is a stand-alone open source software, which allows the collection of any type of measurements from any type of distributed applications, and their storage in a unified format (e.g. SQL). OML data reporting can operate alongside an application's original reporting mechanism or as its replacement. Using OML allows researcher to easily collect and correlate data from different distributed sources to evaluate prototypes, test research hypotheses, or investigate anomalies.

The typical OML data path starts with the definition of a Measurement Point (MP) within the instrumented application. This MP is an abstraction for a tuple of related metrics, which will be reported as a measurement sample to the OML library by the application. During run-time, the OML library turns the received samples into a Measurement Stream (MS). This MS is then directed to an optional Processing Points (PP), which implement processing functions that consume the incoming MS and produce new resulting MSs (e.g. min/max over a window, weighted average of two metrics). The Resulting MSs are then directed to local or remote Termination Points (TP), which store the measurements in various format (e.g. SQL).

An evaluation of OML has been performed [6] to assess its impact on the the instrumented application's performance and on the data reporting channel when such channel is being shared with the experiment's own traffic. This evaluation showed that OML has a statistically non-significant impact on the instrumented application, and that its impact of the data channel can be minimal when it is used with adequate PPs.

III. Motivation

While peer-to-peer networks have demonstrated many advantages as compared to a centralised architecture, including their openness as well as their robustness at scale, they still present numerous challenges in the particular case of HPC. Among these challenges we are particularly interested in the application/task deployment and fault-tolerant capabilities.

In our architecture, using P2PDC, we have described in [4] how the computational scheme is initiated. In particular, in our framework a self-elected peer has to initiate the computation. This initialisation can be seen as a series of successive task, first this peer decomposes the data set, then it sends theses subsets as application code to workers, these workers later receive computed results either directly from other peers or from coordinator peers. In this particular scenario, as only one peer is transmitting the data to all the other peers, it constitutes

[1]http://www.r-project.org/

a bottleneck in the system especially if this peer does not have access to a fast network.

A second generally accepted problem related to HPC over a P2P network concerns the duration of the computation in P2PDC. Indeed, although subtasks are distributed in order to be computed among several peers, this duration may be long. This results in a mandatory commitment from the submitter to always stay connected until the completion of the computation. Hence, if the submitter disconnects, the computation terminates immediately.

A third problem related to HPC over P2P concerns the accurate estimation of the number of available peers at any given time. For example, there might be more free peers during the night than during the day, but users can not always connect and start their application during the night.

Finally, we point out a last drawback of the current version of P2PDC: the received results are in raw format so that users have to make further treatment in order to obtain more sophisticated representations such as graphs or statistical analysis.

In order to overcome these drawbacks and to facilitate the deployment, management of P2PDC applications as well as the retrieval and analysis of results, we propose in this paper a solution based on the combination of P2PDC with tools developed at NICTA (*i.e.*, OML, OMF and OMF Portal).

IV. Task Deployment

In this section, we first introduce a new measurement channel for P2PDC based on the OML library that reduces the volume of collected measurements and which limits the impact of the measurements on the computation. We then present an innovative application of this measurement scheme as task deployment architecture.

A. New measurement Channel

As detailed in Section II-C, the OML framework provides users with filters enabling some preprocessing on a specific measurement stream in a processing point located either at the resource that produces the measurement or on the path of the measurement stream.

We note also that not only unnecessary data are stored in the database while further manipulations need to be made in order to extract necessary information. Hence, we have proposed to provide users with a new type of filter that allows users to perform some preprocessing on several measurement streams from different resources. Such a preprocessing can dramatically reduce the volume of collected measurements and thus limit the impact of the measurements on the computation. In order to minimize the impact on the current architecture, the new type of filter is implemented in an OML proxy-server [19]. The proxy-server can be placed in the same machine as OML Server or in a separate machine. The measurement architecture is displayed in Figure 1.

In the context of P2PDC, we have developed a new type of filters that allows users to perform some preprocessing on several measurement streams from different resources. Such preprocessing can significantly reduce the volume of collected measurements and thus limit the impact of the measurements on the computation and the experimental network. In order to minimize the impact on the current architecture, the new type of filter is implemented in an enhanced OML proxy-server [19] which constitutes an OML Processing Point [6]. The proxy-server can be placed in the same machine as OML Server or in a separate machine. Thus, the measurement architecture is shown in Figure 1.

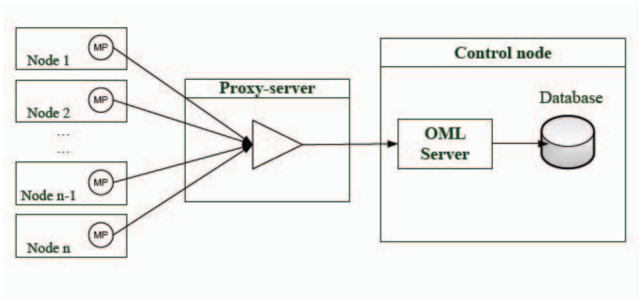

Figure 1: Hierarchical measurement architecture.

In Figure 1, a node does not inject measurement stream directly to OML Server but to processing point (proxy-server). A max(.) filter is implemented on the proxy-server that calculates the maximum computational error from n entering streams (where n is the number of nodes) at each time step and forwards this value to OML Server. Hence, the volume of collected measurements stored in database at OML Server is reduced by a factor n. Moreover, users do not need to make any further manipulation on collected measurements.

In large scale experiments, where the number of nodes involved may be important and spread over a large network, if all measurement streams from all nodes are collected by single OML server (or a proxy-server), this entity may become a bottleneck leading to a loss of efficiency in measurement collection process.

With the presence of filters on proxy-server, we can deploy a hierarchical measurement architecture that not only avoids the bottleneck at OML Server (or proxy-server) but also reduces the volume of measurement data sent over long-distance link. Then, we can put inside a group of nearby nodes a proxy-server implementing a filter that processes measurement streams, in a streaming fashion, injected by peers in this group. Afterward, a top-level proxy-server integrates measurement streams injected by group's proxy-servers and forwards integrated metrics to OML Server.

We note that measurement streams of a group of nearby nodes are pre-processed locally inside this group and only one measurement stream is sent from a given group to top-level proxy-server without a loss of meta-data information thanks to the capabilities of OML [6]. The multi-level hierarchical measurement architecture is displayed in Figure 2.

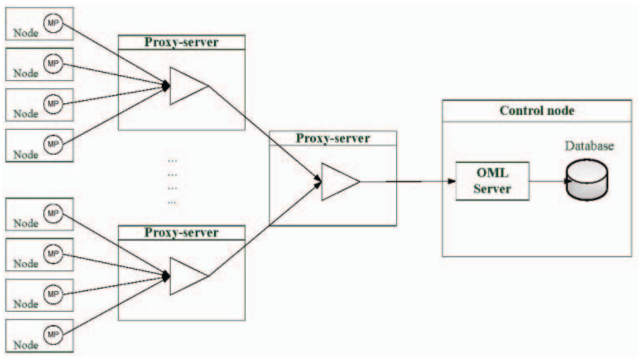

Figure 2: Multi-level hierarchical measurement architecture.

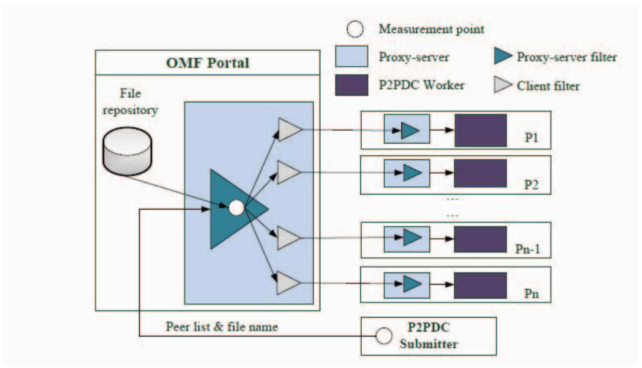

Figure 3: Task deployment via OML

B. Reverse Measurement Framework for HPC Task Deployment and P2PDC Web Portal

In this section, we detail a method that makes use of the OML measurement library in order to deploy tasks on peers. As previously explained, the OML measurement library allows to define measurement points inside programs and create measurement streams to either a local or a remote server.

In our case, we use this library in a reverse manner, whereby data to distribute are injected to several clients instead of the generic case where several clients inject measurements that would be collected by a server. Figure 3 displays the architecture of the system for task deployment.

At the beginning of the solution of a given HPC application (*e.g.*, a numerical simulation problem), the initial dataset is often decomposed into n parts, each part being assigned to a given machine or peer.

In P2PDC architecture, when programmers define a task, they usually need to read the dataset from a binary file, decompose it into subsets and integrate data subsets to subtasks as parameters; then, data subsets are sent along with subtasks to peers. This process is usually quite repetitive and error prone. With the integration of P2PDC and OMF/OML, the task submission and deployment are done through a centralised web server or portal. On this portal, the data file of a task is uploaded to a file Repository and needs to be distributed to peers when the computation begins.

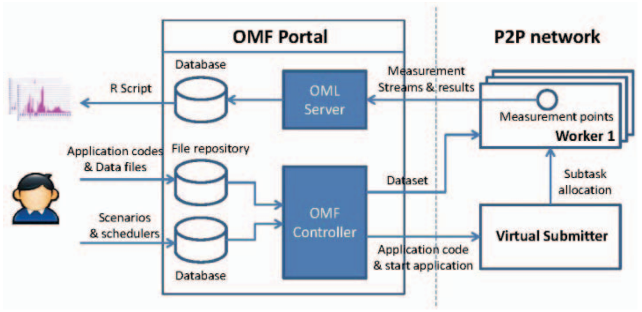

Figure 4: Web portal for HPC applications.

For clarification, let's take the example of the resolution of a discretized numerical simulation problem via a distributed or parallel iterative algorithm. In this class of problems, the dataset is often in the form of a matrix with dimension $d = 1, 2, 3, \ldots$. A solution over four peers can give rise to the following pillar decomposition of the dataset for a matrix with size $128 \times 128 \times 128$: $[0 \ldots 63][0 \ldots 63][0 \ldots 127]$, $[0 \ldots 63][64 \ldots 127][0 \ldots 127]$, $[64 \ldots 127][0 \ldots 63][0 \ldots 127]$ and $[64 \ldots 127][64 \ldots 127][0 \ldots 127]$. During the deployment, each sub-matrix needs to be sent to a peer.

In our method, the portal generates an XML file that defines the deployment/filtering process. This XML file is later stored in the Portal File Repository alongside with data files. Dataset are then distributed automatically to peers via a proxy-server using mechanisms adapted from OML.

Indeed, in order to deploy the dataset to the participating peer, the P2PDC Submitter [4] injects the dataset in an OML Measurement Point (MP). This MP creates the measurement stream in direction to the local proxy-server and therefore free the submitter from the deployment procedure burden.

This proxy-server will in turn takes charge of the dataset transfer to the peers based on a filter describing the decomposition process. This filtering process is called the Init_Portal_Proxy filter in Figure 3. This specific filter creates n OML Measurement Streams, where n denotes the number of peers, in direction to remote proxy-servers on peer-side where another filter, called type Init_Portal_Client, filters out the dataset and presents it to the P2PDC worker in an adequate format. Like the Init_Portal_Proxy filter, Init_Peer filter does not write any data to output but sends data to P2PDC worker. The communication between Init_Peer filter and P2PDC Worker is made via local socket.

We conclude this section by presenting the principle of an HPC web portal for application deployment and measurement as shown in Figure 4.

In this foreseen HPC web portal, users would upload their application codes as well as datasets in file format to the Portal via a web interface. Thanks to this web interface, they can easily customize their application according to different scenarios(*e.g.*, change the dataset, the number of workers or topology description). Moreover, users can schedule to start the application at a given time or when experiment requirements are met.

Based on scheduling information, the OMF Experiment Controller on the Portal selects a given machine on the network to start the application; this machine is called a Virtual Submitter. Virtual Submitters can be dedicated machines managed by Portal administrators or peers with attractive characteristics in the network. Once the application is launched on a given Virtual Submitter, dataset is sent directly from the Portal to workers as described above using the dedicated OML filters.

Once the resolution is complete, results can be collected through an OML measurement stream to the OML Server either from Virtual Submitter or directly from workers. The former case, where results are usually sent from workers to the Virtual Submitter, constitutes the legacy results collection in P2PDC environment [4]. In this case, the Virtual Submitter makes results aggregation and sends the final measurement to the OML server. Finally, the OML Server on the Portal stores results measurement streams into database and offers them to the user once the experiment is finished.

Furthermore, users can write R scripts in order to be able to create graphs or tables representation of the results on the portal. We acknowledge that there may be several Portals on the network as any organization or even any individual user can install its own Portal. Furthermore, with the presence of the Portal, users do not need to stay connected when the computation is running and they can reconnect later on and retrieve the result from the Portal.

Finally we consider that the introduction of a Web Portal facilitates application deployment and measurements. Moreover, knowledge of the state of the network and machines is made possible through a web service.

We also note that the creation of events in order to dynamically tune the solution process is now possible thanks to the OML new channel and the use of appropriate filters. This leads to the possibility to implement easily self-adaptive distributed methods or multi-methods. Thus gains in efficiency become possible.

V. Experiments on PlanetLab

HPC applications have been deployed on PlanetLab thanks to P2PDC and the OMF, OML libraries. We present now a first series of experiments obtained with PlanetLab. We note that the new measurement channel for task deployment has been used in order to carry out these experiments.

In order to test our implementation of OML integration inside P2PDC, we have conducted a measurement campaign over PlanetLab. We present hereafter a first series of experiments obtained with PlanetLab.

PlanetLab is a global research network that supports the development of new network services. Since the beginning of 2003, more than 1,000 researchers at top academic institutions and industrial research labs have used PlanetLab in order to develop new technologies for communication protocols, distributed storage, network mapping, peer-to-peer systems, distributed hash tables, and query processing. PlanetLab consists of more than 1100 nodes at 512 sites.

The HPC application that we have considered concerns the resolution of the obstacle problem [4]. Obstacle problems occur in many scientific domains like mechanics; they occur also as sub-problems of various problems in finance, e.g., Black-Scholes problem for options pricing.

In the stationary case, the obstacle problem can be formulated as follows.

$$\left\{\begin{array}{l} Find\ u^*\ such\ as \\ \Lambda.u^* - f \geq 0,\ u^* \geq \phi\ everywhere\ in\ \Omega \\ (\Lambda.u^* - f)(\phi - u^*) = 0, everywhere\ in\ \Omega \\ B.C. \end{array}\right. \tag{1}$$

where $\Omega \subset \mathbb{R}^2$ (or $\mathbb{R}^3$) is an open set, Λ is an elliptic operator, ϕ a given function and $B.C.$ denotes the boundary conditions on $\partial\Omega$.

There are many equivalent formulations of the obstacle problem in the literature like complementary problem, variational inequality and constrained optimization problem; the reader if referred to [20], [21] and [22] for more details. We concentrate here on the following variational inequality formulation.

$$\left\{\begin{array}{l} Find\ u^* \in K\ such\ as \\ \forall v \in K, \langle \Lambda.u^*, v - u^* \rangle \geq \langle f, v - u^* \rangle \end{array}\right. \tag{2}$$

where K is a closed convex set defined by

$$K = \{v | v \geq \phi\ everywhere\ in\ \Omega\}, \tag{3}$$

and $\langle .,. \rangle$ denotes the dot product $\langle u, v \rangle = \int_\Omega uvdx$

The discretization of the above problem leads to the following large scale fixed point problem whose solution via parallel algorithms present many interest [20], [22].

$$\left\{\begin{array}{l} Find\ u^* \in V\ such\ as \\ u^* = F(u^*), \end{array}\right. \tag{4}$$

where V is an Hilbert space and the mapping $F : v \mapsto F(v)$ is a fixed point mapping from V into V.

We consider the distributed solution of the above fixed point problem via the projected Richardson method combined with asynchronous iterative schemes of computation. Reference is made to [4] for the mathematical formulation of asynchronous projected Richardson methods.

We recall that asynchronous iterative algorithms are successive approximation methods. They correspond to a general model of distributed or parallel computations whereby each processor goes at its own pace and updates without order, nor synchronization, the components of the iterate vector that have been assigned to it. The pace depends on processor characteristics and computational load. The reader is also referred to [23] for more details on asynchronous iterations. The convergence of asynchronous projected Richardson method has been established in [20]. Furthermore, advantages of asynchronous iterative schemes of computation for various problems including boundary value and optimization problems have been shown in particular in [22], [24]–[31].

We note that asynchronous iterations are well suited to

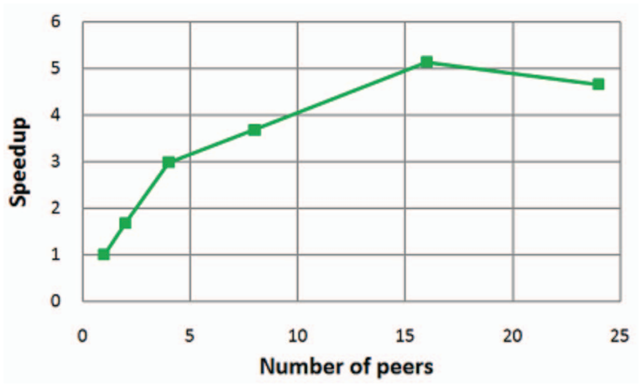

Figure 5: Computational results with PlanetLab.

HPC applications carried out on a network like PlanetLab where latency can be important. Asynchronous algorithms present also some interesting fault-tolerance properties since they permit one to cope with message loss that can occur in a peer-to-peer network. In the asynchronous context, message loss is not critic: it does not lead to system deadlock and the information contained in a missing message can be easily replaced via the one contained in a new message.

We have considered a 3D obstacle problem with size: $192 \times 192 \times 192$. We have used 24 machines from 12 sites (2 machines on each site): 4 sites in the US and 8 sites in Europe. Latency between machines at a same site was about 0.1 ms while latency between machines of different sites varies from 30 ms to 330ms. Machines are heterogeneous; processor's frequency varies from 2.4 to 3.0 GHz.

Experiments have been carried out on 1, 2, 4, 8, 16 and 24 machines. OMF and OML were used to facilitate task deployment as well as measurements like error and solution time.

Computational time in the sequential case (*i.e.*, with one machine), varies from 3158 s to 6555 s according to the configuration of the machine. Synchronous iterative schemes of computation are not suited to this type of networks, since latency is much greater than the duration of a single relaxation in that case. Hence, we have considered only asynchronous iterative schemes of computations in our experiments. Moreover, PlanetLab limits the bandwidth used in 24 hours; thus we have reduced update's frequency in order to respect PlanetLab user's charter: a node sends updates to its neighbors every 10 relaxations. Through experiments, we found that the reduction of update's frequency increases computational time from 5% to 10%.

Computational results are displayed in Figure 5 where the sequential computational time of the fastest machine was used in order to calculate speedup.

We note that the combination of asynchronous iterative schemes of computation with the decentralized environment for high performance peer-to-peer distributed computing P2PDC and OMF, OML libraries is attractive. As a matter of fact despite heterogeneity and sometimes big latencies due to machines located on two continents, non-negligible speedup is nevertheless obtained.

VI. Conclusion

The integration of the OML measurement tool within the P2PDC decentralized environment for HPC on heterogeneous computing platforms has been considered in this paper. This integration presents the following three benefits.

1) A new distributed measurement channel based on the OML measurement library has been introduced. This channel reduces the volume of collected measurements and thus limits the impact of the measurements on the computation.
2) A modular system for the injection of computational data into a network has been proposed. This system is used in particular for efficient distribution of data on the overlay network and deployment of HPC applications on distributed platforms such as PlanetLab.
3) Thanks to the OML new channel and the use of appropriate filters, the creation of events in order to dynamically tune the solution process is made possible. This leads to self-adaptive distributed methods or multi-methods that are more efficient.

A Web Portal for HPC applications deployment and measurement has also been proposed in this paper. This Portal results from the combination of P2PDC with tools developed at NICTA, i.e. OML, OMF and OMF Portal. In particular, we have given the principle of the Portal architecture and explained how this Portal can facilitate the deployment, the management of P2PDC applications as well as the retrieval and analysis of computational results.

Finally, the deployment of an HPC application on PlanetLab has been considered. The numerical solution of an obstacle problem has been carried out on a network with 24 machines in twelve sites and two continents. A first series of computational results has been displayed and analyzed.

In the future, we plan to extend the P2PSAP communication protocol in order to take into account MX/Myrinet networks as well as multi-networks platforms that combine Infiniband, MX/Myrinet and Ethernet networks. We plan also to concentrate on heterogeneity. In particular, we shall consider heterogeneous platforms combining CPUs and GPUs.

Acknowledgement

Part of this study has been made possible via ANR contract ANR-07-CIS7-011. Experiments presented in this paper were carried out using the PlanetLab experimental testbed being developed under GENI with support from several Universities as well as funding bodies.

References

[1] K. Hwang, G. Fox, and J. Dongarra, *Distributed and Cloud Computing, from Parallel Processing to the Internet of Things*. Morgan Kaufmann, 2012.

[2] J. Dongarra, "Emerging heterogeneous technologies for high performance computing," in *Proc.* 27^{th} *IEEE International Parallel & Distributed Processing Symposium and Workshops (IPDPSW)*, Boston, May. 2013.

[3] V. Boyer and D. El Baz, "Recent advances on GPU computing in Operations Research," in *Proc.* 27th *IEEE International Parallel & Distributed Processing Symposium and Workshops (IPDPSW)*, Boston, May. 2013, pp. 1778–1787.

[4] T. T. Nguyen, D. El Baz, P. Spiteri, G. Jourjon, and M. Chau, "High performance peer-to-peer distributed computing with application to obstacle problem," in *Proc.* 24th *IEEE International Parallel & Distributed Processing Symposium and Workshops (IPDPSW)*, Atlanta, May. 2010, pp. 1453–1461.

[5] B. Cornea, J. Bourgeois, T. T. Nguyen, and D. El Baz, "Performance prediction in a decentralized environment for peer to peer computing," in *Proc.* 25th *IEEE International Parallel & Distributed Processing Symposium and Workshops (IPDPSW)*, Anchorage, May. 2011, pp. 1613–1621.

[6] O. Mehani, G. Jourjon, T. Rakotoarivelo, and M. Ott, "An instrumentation framework for the critical task of measurement collection in the future Internet," NICTA, Tech. Rep. 6065, 2012.

[7] T. Rakotoarivelo, M. Ott, I. Seskar, and G. Jourjon, "OMF: a control and management framework for networking testbeds," in *SOSP Workshop on Real Overlays and Distributed Systems (ROADS '09)*, Big Sky, USA, Oct. 2009, p. 6.

[8] G. Jourjon, T. Rakotoarivelo, and M. Ott, "A portal to support rigorous experimental methodology in network research," in *Proc. of TridentCom*, Apr. 2011.

[9] ——, "From Learning to Researching, ease the shift through testbeds," in *Proc. of TridentCom*, ser. LNICST, vol. 46. Berlin Heidelberg: Springer-Verlag, 2010, pp. 496–505.

[10] T. Garcia, M. Chau, T. T. Nguyen, D. El Baz, and P. Spiteri, "Asynchronous peer-to-peer distributed computing for financial applications," in *Proc.* 25th *IEEE International Parallel & Distributed Processing Symposium and Workshops (IPDPSW)*, Anchorage, May. 2011, pp. 1453–1461.

[11] M. Hifi, T. Saadi, and N. Haddadou, "High performance peer-to-peer distributed computing with application to constrained two-dimensional guillotine cutting problem," in *Proc. 19th Conference on Parallel, Distributed and networked-based Processing*, Cyprus, Feb. 2011, pp. 552–559.

[12] D. El Baz, M. Hifi, and T. Saadi, "Peer-to-peer solution of 2D-cutting stock problems," in *Proc. 11th Workshop on Graphs and Combinatorial Optimization*, 2012, pp. 116–120.

[13] D. El Baz and T. T. Nguyen, "A self-adaptive communication protocol with application to high performance peer-to-peer distributed computing," in *Proc. of the 18th Conference on Parallel, Distributed and networked-based Processing, PDP 2010*, Pisa, Italy, Feb. 2010, pp. 327–333.

[14] S. Tembo, T. T. Nguyen, and D. El Baz, "Distributed Iterative Solution of Numerical Simulation Problems on Infiniband and Ethernet Clusters via the P2PSAP Self-Adaptive Protocol," in *Proc. of the 21st Euromicro conference on Parallel, Distributed and Network-Base Processing*, Belfast, Feb. 2013, pp. 121–125.

[15] T. T. Nguyen and D. El Baz, "Fault-tolerant implementation of peer-to-peer distributed iterative algorithms," in *Proc. 15th IEEE International Conference on Computational Science and Engineering*, Paphos Cyprus, 2012, pp. 137–145.

[16] G. Wong, M. Hiltunen, and R. Schlichting, "A Configurable and Extensible Transport Protocol," in *Proceedings of IEEE INFOCOM*, 2001, pp. 319–328.

[17] M. A. Hiltunen, "The CACTUS approach to building configurable middleware services," in *Proc. of DSMGC 2000*, 2000.

[18] G. Jourjon and D. El Baz, "Some solutions for Peer to Peer Global Computing," in *Proc. of the 13th Euromicro conference on Parallel, Distributed and Network-Base Processing*, 2005, pp. 49–58.

[19] J. White, G. Jourjon, T. Rakotoarivelo, and M. Ott, "Measurement architectures for network experiments with disconnected mobile nodes," in *TridentCom 2010, 6th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities*, ser. Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, A. Gavras, N. Huu Thanh, and J. Chase, Eds., ICST. Heidelberg, Germany: Springer-Verlag Berlin, May 2010. [Online]. Available: http://www.nicta.com.au/research/research_publications/show?id=3298

[20] P. Spitéri and M. Chau, "Parallel Asynchronous Richardson Method for the Solution of Obstacle Problem," in *Proc. of the 16th Annual International Symposium on High Performance Computing Systems and Applications*, 2002, pp. 133–138.

[21] J.-L. Lions, *Quelques Méthodes de Résolution des Problèmes aux Limites non Linéaires*. Dunod, 1969.

[22] J. C. Miellou, D. El Baz, and P. Spitéri, "A new Class of Asynchronous Iterative Methods with Order Intervals," *Mathematics Of Computation*, vol. 67, no. 221, pp. 237–255, 1998.

[23] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

[24] D. Bertsekas and D. El Baz, "Distributed Asynchronous Relaxation Methods for Convex Network Flow Problems," *SIAM Journal on Control and Optimization*, vol. 25, no. 1, pp. 74–85, 1987.

[25] D. El Baz, "A computational experience with distributed asynchronous iterative methods for convex network flow problems," in *Proc. of the 28th IEEE Conference on Decision and Control*, 1989, pp. 590–591.

[26] D. El Baz, "M-functions and Parallel Asynchronous Algorithms," *SIAM Journal on Numerical Analysis*, vol. 27, no. 1, pp. 136–140, 1990.

[27] D. El Baz, "Asynchronous gradient algorithms for a class of convex separable network flow problems," *Computational Optimization and Applications*, vol. 5, no. 3, pp. 187–205, 1996.

[28] D. El Baz, P. Spiteri, J. C. Miellou, and D. Gazen, "Asynchronous iterative algorithms with flexible communication for nonlinear network flow problems," *Journal of Parallel and Distributed Computing*, vol. 38, pp. 1–15, 1996.

[29] M. Jarraya and D. El Baz, "Implementation of distributed iterative algorithm for optimal control problems on several parallel architectures," *Journal of Systems and Software*, vol. 60, pp. 141–148, 2002.

[30] D. El Baz, A. Frommer, and P. Spiteri, "Asynchronous iterations with flexible communication: contracting operators," *Journal of Computational and Appied Mathematics*, vol. 176, pp. 91–103, 2005.

[31] M. Chau, D. El Baz, R. Guivarch, and P. Spiteri, "MPI implementation of parallel sub-domain methods for linear and nonlinear convection-diffusion problems," *Journal of Parallel and Distributed Computing*, vol. 67, pp. 581–591, 2007.