26th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing

Heterogeneous Computing and Multi-Clustering Support via Peer-To-Peer HPC

Bilal FAKIH
LAAS-CNRS, Université de Toulouse, CNRS
Email: bfakih@laas.fr

Didier EL BAZ
LAAS-CNRS, Université de Toulouse, CNRS
Email: elbaz@laas.fr

***Abstract*—This paper aims at presenting Peer-To-Peer HPC a decentralized environment that facilitates the use of heterogeneous multi-cluster platform for loosely synchronous applications. The goal is to exploit all the computing resources (all the available cores of computing nodes) as well as all networks, e.g., Ethernet, Infiniband and Myrinet. Peer-To-Peer HPC functionality relies on a reconfigurable multi network protocol RMNP for controlling multiple network adapters and on OpenMP for the exploitation of all the available cores of computing nodes. We report on efficiency obtained with Grid5000 testbed by combining synchronous and asynchronous iterative schemes of computation with Peer-To-Peer HPC. The experimental results show that our environment scales well.**

***Keywords*-computing environment, multi-cluster computing, multi-threading, peer-to-peer computing, distributed computing, High Performance Computing, loosely synchronous applications.**

I. Introduction

In this paper, we are mainly interested in the solution on large scale distributed computing systems of High Performance Computing (HPC) applications that belong to the class of loosely synchronous applications and which exhibit iterative compute communication stages [1]. We concentrate on numerical simulation problem that are solved via parallel synchronous or asynchronous iterative algorithms. We propose the decentralized environment Peer-To-Peer HPC designed to provide an efficient, scalable and portable support for high performance computing applications in a multi-cluster, multi-core context. Peer-To-Peer HPC facilitates the use of large scale distributed systems and the work of programmers. In particular, it uses a limited number of communication operations.

A first environment, called P2PDC, was proposed in 2008 (see [6], [7]). This environment presented several limitations like the use of one type of network, i.e., Ethernet and the use of only one CPU core per computing node. As an attempt to overcome P2PDC limitations and in order to reduce the solution time to solve HPC applications, the development of Peer-To-Peer HPC is presented in this paper to take into account two goals. The first objective is to use simultaneously several networks like Ethernet, Infiniband and Myrinet. This feature is particularly important since we consider loosely synchronous applications that present frequent data exchanges between computing nodes. Hence we privilegiate to use several high speed networks simultaneously, i.e., Infiniband and Myrinet in the same application session. Note that the reconfigurable multi network protocol RMNP supports data exchanges via multi-network configuration. The second objective is to use all the computing resources of modern muli-core CPUs, i.e., all CPU cores. The data exchange between cores inside a computing node is made via OpenMP [2].

In the sequel, we study the combination of Peer-To-Peer HPC and distributed synchronous or asynchronous iterative schemes of computation for the obstacle problem. Our computational experiments are carried out on the Grid5000 platform [3]. They show that Peer-To-Peer HPC scales well and that the combination of Peer-To-Peer HPC and asynchronous iterative schemes of computation is efficient.

The remainder of the paper is organized as follows : related work is presented in Section II. Section III presents the RMNP protocol; it depicts the context and contribution of our work and the different issues to be addressed in supporting communication in a multi-cluster, multi-core context. Section IV presents the architecture of Peer-To-Peer HPC and task allocation. Computational results for large scale numerical simulation problems using Peer-To-Peer HPC are displayed and analyzed in Section V. Finally, Section VI concludes this paper and briefly presents future work.

II. Related Work

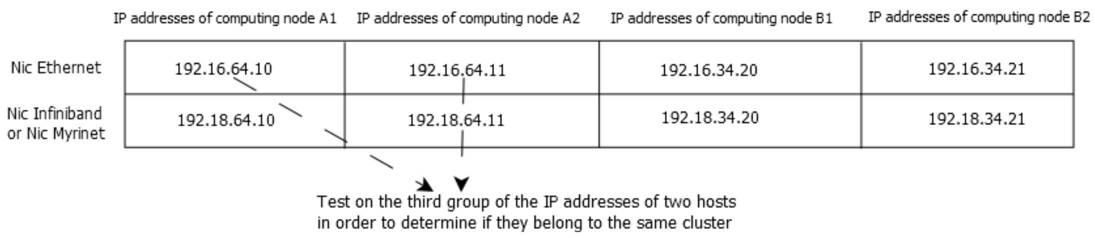
This section presents a brief survey of works that are related to the use of large scale distributed computing systems for several HPC applications.

A. *MPICH/Madeleine*

MPICH Madeleine [4] aims at enabling an efficient and exhaustive use of underlying communication software and hardware functionalities for distributed applications. MPICH Madeleine is based on a generic multi-protocol communication library called Madeleine to deal with several networks simultaneously.

B. *Software And Middleware For Peer-To-Peer and Volunteer Computing*

Middleware like BOINC [5] have been developed in order to exploit the CPU cycles of computers connected to the Internet in peer-to-peer applications. Those systems are generally dedicated to peer-to-peer applications where



	IP addresses of computing node A1	IP addresses of computing node A2	IP addresses of computing node B1	IP addresses of computing node B2
Nic Ethernet	192.16.64.10	192.16.64.11	192.16.34.20	192.16.34.21
Nic Infiniband or Nic Myrinet	192.18.64.10	192.18.64.11	192.18.34.20	192.18.34.21

Test on the third group of the IP addresses of two hosts
in order to determine if they belong to the same cluster

Figure 1: Example of content of Htable and test on the location of the hosts thanks to comparison of IP addresses

tasks are independent and direct communication between machines is not needed.

C. Grid Computing

Globus [8] and Legion [9], are open source software libraries for the grid computing community. They support many operational grids and their applications on an international basis. Globus and Legion are representative of large scale meta-computing systems. They address issues such as heterogeneity, programmability, scalability or interoperability and coupling of high performance architectures or networks of computing nodes.

III. Context and Contribution

A. Heterogeneous Multi-Cluster Environment

Multi-cluster systems and grids generally consist of interconnected stand-alone computers that can work cooperatively as a single integrated computing resource. Supporting heterogeneous multi-cluster mainly consists in integrating switching functionality to switch from one network to another, according to the communication needs.

B. Reconfigurable Multi Network Protocol RMNP

The reconfigurable multi network protocol RMNP aims at enabling an efficient use of the complete set of underlying communication softwares and hardwares available in a given multi-network system. It is able to deal with several networks via the management of several networks adapters. The user application can dynamically switch from one network to another, according to the communication needs.

1) Network Selection: The main function consists in managing several networks adapters within the same application session. Several network interface cards (NICs) are added to the interface of the RMNP and information about these NICs are stored in a data structure called Htables (see Fig.1). This data structure permits each computing node to switch between the networks according to the communication needs. The network management procedure has two steps. First step corresponds to the test of the locality between the computing nodes and the second step corresponds to the choice of the appropriate network for data exchange depending on their locality. The locality test is based on comparing the IP addresses of two computing nodes and according to this comparison, we deduce if the computing nodes are in the same cluster or not (see Fig.1). The second step is based on choosing the best interface network (high speed and low latency network) from the Htables according to the result of the locality test. In the Htables, the ip addresses that are given on the first line correspond to Ethernet network and the ip addresses given on the second line if any correspond to fast network like Infiniband or Myrinet. Consequently, if the locality test returns that the computing nodes are in different clusters, then the Ethernet network interface is chosen from the Htables to perform the communication since the Grid5000 platform uses only Ethernet network between clusters in different sites. If the locality test returns that the computing nodes are in the same cluster, then the best network interface in the Htables is selected.

2) Communication operations: The idea is to facilitate programming of large scale Peer-To-Peer applications and hide complexity of communication management as much as possible. RMNP has a reduced set of communication operations, there are only a send, receive and wait operations: P2P_Send, P2P_Receive and P2P_Wait, respectively. Contrarily to MPI communication library where communication mode is fixed by the semantics of communication operations, the communication mode of a given communication operation which is called repetitively depends on the context at application level like distributed iterative schemes of computation, e.g., synchronous or asynchronous iterative schemes and elements of context like topology at network level, i.e., inter or intra cluster communication.

IV. Environment architecture

In this section, we detail components that support specificity of multi-core and heterogeneous-networks configuration.

A. Environment architecture of Peer-To-Peer HPC

The decentralized environment Peer-To-Peer HPC natively supports any combination of networks and multi-core CPUs by using the reconfigurable multi network protocol RMNP and OpenMP. Peer-To-Peer HPC works with tools called helper programs that are responsible for the analysis of the application and building the topology and routing tables. It then spawns the session processes and connects them together. The helper programs are composed of four components : Job Initialization, Job Execution, Topology Initialization and RMNP-OpenMP.

- **Job Initialization** is responsible for task splitting into sub-tasks, sub-tasks distribution. In particular, the job initialization manages task decomposition and assignment to individual CPU cores. It decomposes the initial task into sub-tasks and sub-sub-tasks so that they are balanced fairly on the CPU cores. Note that this component uses all the CPU cores in a given computing node.
- **Job Execution** executes sub-tasks and takes care of data exchanges, i.e., communication of iterates (updates) of the parallel iterative method. It is responsible for task execution on the different CPU cores and results collection.
- **Topology Initialization** organizes connected computing nodes into clusters and maintains links between clusters. In particular, it relies on a concept based on storing in the Htable informations about the network interface card (NIC) used in the application by each computing node (see Figure 1). This information is needed when making communications with a heterogeneous-network multi-cluster configuration.
- **RMNP-OpenMP** provides support for directed data exchange between computing nodes on several high speed networks like Infiniband and Myrinet using the Reconfigurable Multi Network Protocol RMNP and between the cores in a computing node via OpenMP.

B. *Task allocation in Peer-To-Peer HPC*

Task allocation in Peer-To-Peer HPC is based on the hierarchical Master-Worker paradigm. The Hierarchical Master-Worker paradigm consists of three entities: a master, several sub-masters and several workers. The master or submitter is the unique entry point, it gets the entire problem as a single task, i.e., the root task. The root task decomposes the problem into sub-tasks and distributes these sub-tasks amongst a farm of workers. The root task is responsible for gathering the scattered results in order to produce the final result of the computation. The sub-masters are intermediary entities that enhance scalability. They forward sub-tasks from the master to workers and return results to the master. The workers run in a very simple way: they receive a message from the sub-master that contains their assigned sub-tasks, distribute the sub-tasks to the different cores, perform computations, exchange data with neighboring computing nodes and in the end of the application, when the iterative scheme has converged, they regroup the results from all the cores and send them back to the sub-master. Note that the number of workers in a group cannot exceed 32 in order to ensure efficient management of a sub-master.

V. Evaluation and Computing Results

This section presents an evaluation of the overall efficiency and scalability of Peer-To-Peer HPC in a multi-core and multi-network context for the obstacle problem.
The obstacle problem occurs in many domains like mechanics and finance and can be formulated as follows:

$$\begin{cases} Find\ u^*\ such\ that \\ A.u^* - f \geq 0, u^* \geq \varnothing\ everywhere\ in\ \Omega, \\ (B.u^* - f)(\varnothing - u^*) = 0\ everywhere\ in\ \Omega, \\ B.C., \end{cases} \tag{1}$$

where the domain $\Omega \in \mathbb{R}^2(or\ \mathbb{R}^3)$ is an open set, A is an elliptic operator, $\varnothing$ a given function and B.C. denotes the boundary conditions on $\sigma\Omega$.

We consider the discretization of the obstacle problem. The distributed solution of the associated fixed point problem via the projected Richardson method combined with several iterative schemes of computation is considered; reference is made to [10] for the mathematical formulation of synchronous and asynchronous projected Richardson methods. The interest of asynchronous iterations for various problems including boundary value problems and optimization has been shown in [11].
Several experiments are carried out via Peer-To-Peer HPC to solve the 3D obstacle problem with different schemes of computation, i.e. synchronous and asynchronous schemes of computation. We consider cubic domains with n = 256 points where n denotes the number of points on each edge of the cube. In the distributed context, i.e., for several machines, we have considered the case where machines either belong to a single cluster or to several clusters connected via Internet.

A. *Experiments*

We display on Figure 2 and 3 the computing time and computing gain of the parallel synchronous and asynchronous iterative algorithms. The computing gain is given as follows:

$$Computing\ gain\ C_g = t_1/ts \tag{2}$$

where t_1 is the parallel time on one multi-core computing node and ts is the parallel time on several computing nodes.

The different iterative methods are denoted by : Syn-ETH-IB-MYRI and Asyn-ETH-IB-MYRI in Figure 2 and Syn-ETH and Asyn-ETH in Figure 3 where Syn and Asyn denote synchronous and asynchronous iterative schemes, respectively; ETH, IB and MYRI denotes the type of network that is used in the test, i.e., Ethernet, Infiniband and Myrinet, respectively. Table I displays the characteristics of the machines used in the computational experiments.

The results displayed in Figure 2 are obtained with a multi-cluster configuration located in Lille, i.e., Chinqchint

Site	Cluster	Processors Type	Cores	Interconnection Networks	Speed Ghz	RAM GB	Time sec	problem size
Lille	Chinqchint	Intel Xeon E5440 QC	8	Ethernet and Myrinet	2.83	8	3298	256^3
Nancy	Graphene	Intel Xeon X3440	4	Ethernet and Infiniband	2.53	16	3119	256^3
Rennes	Paravance	Intel Xeon E5-2630v3	32	Ethernet	2.4	128	1131	256^3

Table I: Characteristics of machines and parallel time on one multi-core computing node on each site

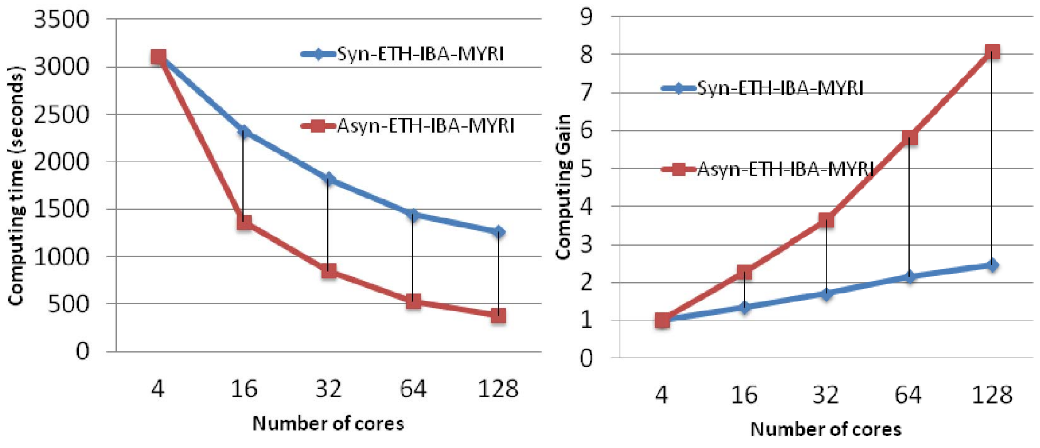


Figure 2: Computation results over Ethernet + Infiniband + Myrinet on Chinqchint cluster in Lille (eight cores per computing node) and Graphene cluster in Nancy (four cores per computing node) in the case of the obstacle problem with size 256^3

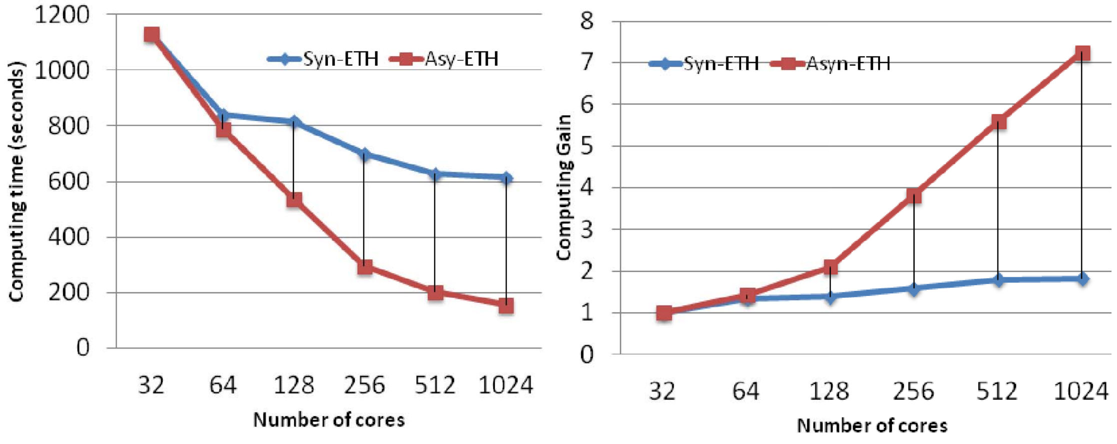


Figure 3: Computation results over Ethernet on Paravance cluster in Rennes (32 cores per computing node) in the case of the obstacle problem with size 256^3

cluster and Nancy, i.e., Graphene cluster of the Grid5000 testbed. Lille and Nancy are two French towns three hundred kilometers apart. The experiments are carried out with up to 128 cores. We note that there is the same number of cores in the different clusters, i.e., 64 cores in Graphene cluster and 64 cores in Chinqchint cluster. Data exchange is made via Infiniband network in Graphene cluster and via Myrinet network in Chinqchint cluster and the communications between clusters are done via 10 Gb/s Ethernet network.

The results in Figure 3 are obtained using a cluster located in Rennes site, i.e., Paravance cluster of the Grid5000 testbed. The experiments are carried out with up to 1024 cores. The interconnection networks is 10 Gb/s Ethernet.

B. Discussion of the experimental results

Figures 2 and 3 show that the computing gain C_g (see equation (2)) of the synchronous iterative schemes increases slowly with the number of cores, the computing gain of the asynchronous iterative schemes increases more rapidly. This is due to the fact that in the case of synchronous iterative schemes of computation fast computing nodes have to wait for slow computing nodes since they are synchronized via message exchange; this leads to idle time due to synchronization. In the case of asynchronous iterative schemes of computation there is no synchronization and communications are covered by computation; which explains the better computing gain. Computing gains in Figure 2 are computed by using computing time with the faster computing node, i.e., Graphene cluster in Nancy site. Experimental results in Figure 3 show that Peer-To-Peer HPC achieves scalability when combined with asynchronous iterations.

VI. Conclusion and Future work

In this paper, we present the Peer-To-Peer HPC decentralized environment for loosely sunchronous applications. In particular, we detail the features induced by multi-core and heterogeneous-networks support.

Peer-To-Peer HPC relies on OpenMP to support the specificity of multi-core computing and on RMNP communication protocol to support data exchanges via multi-network configuration.

We display and analyze computing results on the Grid5000 platform with up to 1024 computing cores for numerical simulation problems, i.e., the obstacle problem. Computing results show that the combination of asynchronous iterative schemes of computation with Peer-To-Peer HPC allows one to solve efficiently large scale numerical simulation problems. Also, the simulation results show that Peer-To-Peer HPC achieves scalability. Future work will focus on problems with large granularity that should exhibit better computing gain. We shall carry out experiments on networks with several thousands computing cores. Distributed application deployment will also be considered. Other types of parallel applications will be studied like planning problems.

Acknowledgment

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by INRIA and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

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] "OpenMP",http://www.openmp.org/wp-content/uploads/openmp-examples-4.5.0.pdf

[3] "Grid5000 platform," http://www.grid5000.fr. [Online]. Available: http://www.grid5000.fr.

[4] O. Aumage, G. Mercier, "MPICH/Madeleine: a True Multi-Protocol MPI for High Performance Networks," 15th International Parallel and Distributed Processing Symposium (IPDPS'01), 2001.

[5] David P. Anderson,"BOINC: A System for Public-Resource Computing and Storage," 5th IEEE/ACM International Workshop on Grid Computing.November 8, 2004, Pittsburgh, USA.

[6] B. Cornea, J. Bourgeois, T. T. Nguyen, and D. El Baz, "Performance prediction in a decentralized environment for peer-to-peer computing," in Proceedings of the 25th IEEE Symposium IPDPSW 2011 / HOTP2P 2011, Anchorage, USA, 2011, pp. 16131621.

[7] D.El Baz, T. T. Nguyen, "A self-adaptive communication protocol with application to high performance peer to peer distributed computing," in Proceedings of the 18th Euromicro conference on Parallel, Distributed and Network-Base Processing, Pisa, Italy, 2010.

[8] I. Foster and C. Kesselman. The Globus project: a status report. Futur Generation Computer System, 40:3548,1999.

[9] A. Grimhaw and W. Wulf. The legion vision of a worldwide virtual computer. Communications of the ACM, 40, Juanary 1997.

[10] 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 Proceedings of the 24th IEEE Symposium IPDPSW 2010 / HOTP2P, Atlanta, USA, 2010.

[11] D. P. Bertsekas and J. N. Tsitsiklis, "Parallel and Distributed Computation: Numerical Methods". Upper Saddle River, NJ, USA: Prentice-Hall, Inc. (republished in 1997 by Athena Scientific), 1989.