High Performance Computing based on a Smart Grid approach

Alexandru Mihai Vulcan
University Politehnica of Bucharest, Department of Industrial Informatics and Automatic Control
Bucharest, Romania
vulcan.mihai@gmail.com

Maximilian Nicolae
University Politehnica of Bucharest, Department of Industrial Informatics and Automatic Control
Bucharest, Romania
max.nicolae@upb.ro

***Abstract*— High Performance Computing (HPC) facilities serve as infrastructure for solving demanding computational problems met in various domains. HPC involves expensive resources both material and operational. Therefore, leveraging its operation at his peak capacity is very important from economic efficiency perspective. Nowadays, it is very difficult for a regular researcher that needs HPC services to have ease of access to such resources. With the work presented in this paper we propose and evaluate a different approach to HPC service by mapping the problem on the same bases as smart grid philosophy. Mainly, the same as smart grid tries to increase the efficiency of resource utilization together with the concept of co-generation, our proposed model tries to link powerful computational resources own by domestic user into a smart HPC service with for profit objectives.**

Keywords*— *HPC service, GPU virtualization, energy consumption

I. INTRODUCTION

It is well known that HPC facilities use virtualization techniques to better cope with the dynamic demand of utilization. The use of cloud techniques for virtualization increased the efficiency of resource operation, thus, resulting in a lower cost for computation (or storage) services. HPC infrastructures consist of computers that in most cases incorporate last generation graphical processing units (GPUs) especially for their capacity to accelerate computation [1][2].

This approach is known as general-purpose computing on graphics processing units (GPGPU). As mentioned before, one idea for using resource virtualization is increasing the resources' exploiting efficiency. This idea can be extended to computational resources that not necessarily exist in a compact physical area.

In opposition, our proposed architecture will have some drawbacks in performance due to communication issues but it will make use of some "spares" resources that otherwise would be under capacity exploited, and, overall, the result will provide a more efficient solution than exists at this moment.

Basically, we address the domain of distribute heterogeneous computing which is not a new technology. The novelty comes from using two concepts met in distributed computing with the idea of smart grid. We address the large number of personal computers (PCs) that have high performance resources in terms of video cards (GPU). These computers are mainly intended for rendering multimedia content. One person acquires such powerful computer capable of running demanding game engines with entertainment purposes.

Most of the time these resources are used way beneath their capacity. One problem is how someone would be willing to share his resource for others purpose, especially if using it triggers some energy consumption which, in the end, cost money. The idea is to quantify the energy it consumes in opposition to the service it could provide. This way, the person who is willing to share his or her resource would benefit afterword.

By using a smart grid approach, we can provide a service to encourage persons to share their resources with computational purpose. Let us take the following use case scenario. One researcher (or generally, someone), farther denoted by the term client, developed an algorithm for solving a specific problem with high computational demands. Normally, a realistic execution time for his algorithm (further referred as job) will require the use of a HPC infrastructure. With today procedures, obtaining some time on such infrastructure is very complicated and it cannot be achieved easily by any researcher or interested company. Several steps are needed in advance.

Someone could perceive this approach as discouraging. On the other hand, in order for an expansive HPC infrastructure to pay back the investment, the way it is managed is very important. The time scheduler for the jobs loaded on it being carefully monitored, normally having all the time ready jobs prepared in its buffers. Thus, "our client" would have to address a lot of overhead for his job. As opposite, we propose a service that will be easily accessed by clients at the cost of the energy consumed by the resources needed and an interest.

Fig. 1 illustrates how our solution resembles the smart grid architecture. There are clients that both consume and generate "energy". The structure serving the job changes dynamically, being capable of reconfiguration.

Further, the paper discusses some similar approaches that we consider laying in the same problems we must overcome.

2379-0482/17 $31.00 © 2017 IEEE
DOI 10.1109/CSCS.2017.100

In this context, we detail our proposed solution, followed by a feasibility evaluation from the cost perspective, and the proof of concept. In the end, some conclusions and further directions are discussed.

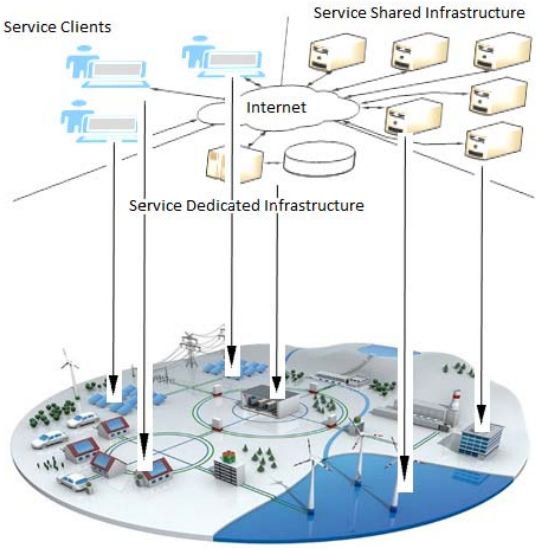

Fig. 1 Our proposed service mapped over smart

II. SIMILAR APPROACHES

From a broader view, our solution is related to distributed computing. Going a bit deeper in the architecture of each constitutive node, we get in the heterogeneous computing domain, as we make use of both CPU and GPU. Distributed and heterogeneous computing are domains that benefited for a lot of attention in the last period. Moreover, parallel and distributed computing are currently undergoing profound mutations [3].

As mentioned in the introduction, they are not new approaches, instead, they rich some level of maturity. These technologies have led to architectures with hundred thousand or millions of cores where application deployment, heterogeneity, power consumption and fault-tolerance are key issues [3]. The idea of using home PC for distributed computing is not new also.

One of the first model is the SETI@home project [4], managed by a group of researchers at the Space Sciences Laboratory of the University of California, Berkeley. It used large-scale distributed computing to perform a sensitive search for radio signals from extraterrestrial civilizations. Further, the same group started the project Berkley Open Infrastructure for Network Computing (BOINC). But the principle of this type of resource searing is the Volunteer Computing that includes volunteer hosts [5]. We know from other domains of social life that volunteer doesn't necessarily equals large quantity or/and high performance or quality. On the other hand, these volunteer approaches can gather consensus when involves non-for profit goals or subjects with large interest. Resuming to the use case scenario described in the introduction, there are little chances for our researcher to access a volunteer network for his own benefit. One issue will be how much he is willing to pay, and for this it is important to evaluate how much will it cost. However, the researches addressing volunteer computing systems concerning availability and performance under unreliable Internet connected host are of great use in our approach [6], [7], [8].

In parallel computing on distributed system, the communication cost is one of the most important drawbacks. Geographical spares computers are not suitable for many parallel algorithms. But considering heterogeneous computing (especially the use of high performance GPUs) the load of communication involved in parallel processing will decrease. Working remotely with GPU through virtualization was also addressed in the past [9]. Framework such as CUDA [10] and OpenCL [11] offer the possibility to use the GPU for general-purpose computing. Solution like DS-CUDA [12][13], rCUDA [14] offers remote GPU virtualization. Using GPU from different vendors seems not to be an issue [15].

When it comes of integrating distributed heterogeneous computing with volunteer network but on for-profit bases, we found only one similar solution in the literature – Crowdware [16]. That work deals more with the auction-based price model and use the energy consumption from GPU specification. Afterwards there is not a continuation. We want to address the feasibility of the concept and copy the smart grid model.

III. OUR SOLUTION

The objective is to bypass the need for a datacenter, cluster or cloud service and not to be region based. Our model is based on a client – server architecture which uses TCP/IP (Transmission Control Protocol/ Internet Protocol) infrastructure. Our model is composed of nodes, service server and the communication infrastructure.

The nodes are composed of client machines and server machines. A node has three states: down, available and busy. At any given time, a client machine can become a server machine and vice versa. The service server handles task and resource scheduler and availability; the server is the connection between the client service and shared service.

The idea behind this model is to "co-generate" services in the network. Nowadays the number of powerful gamming computers has increased and is still growing; the owners of such computers usually uses them for amusement, learning, web navigation and other small office activities. Thus, from the investment point a view, these computers are not proper managed. Generally, a gamming computer will run for a small amount of time during the day and, afterwards, in the remaining time this expensive resource is unused.

We asked ourselves three questions: how efficient will be the operation of this type of service, how can this resource be exploited at its capacity and is this type of resource useful in other purposes? The introduction of this papers offers positive answer for the last question. In this section, we tried to answer the second question while in the next section we talk about energy consumption.

We lay down the following principles based on the proposed model:

• Every server node requires the server engine application (SEA) to be installed. This way, the server node will be enrolled in the service server registers and its state updated as needed.

• Client nodes can use the service only if they run the client engine application (CEA).

• The CEA analyzes the client program or application, connects to the service server which, in turn, will provide information about available resources and execution profiles available in the network. The client will also receive some recommendation profiles based on his program (manual selection Fig. 2). The recommendations profiles will be based on characteristics like the shortest execution time or cheapest profile. Basically, a profile will be defined based on a priory information and estimation, and it will consist of a list of resources involved and their relative time utilization. The system is meant to learn in time so profile recommendations will become more advanced over a period of time.

•The client also has the option of using an automatic profile selection (automatic selection Fig 3)

•The server engine has the option to queue additional information on the GPU virtualization service. The service server is able to distribute the processing task on multiple server nodes, forming a dedicated cluster.

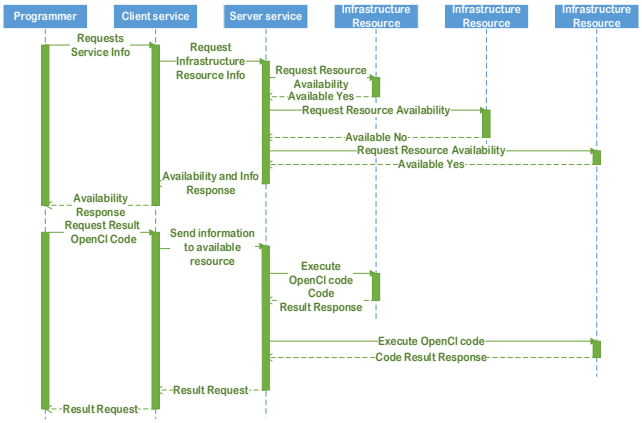

Fig. 2 Manual selection

When a client will choose the automatic option, the service will analyze the complexity of the algorithm, will analyze the size and type of the input data and will determine if it is possible to run the program on the local machine using the CPU or the GPU. If executing the algorithm on the local

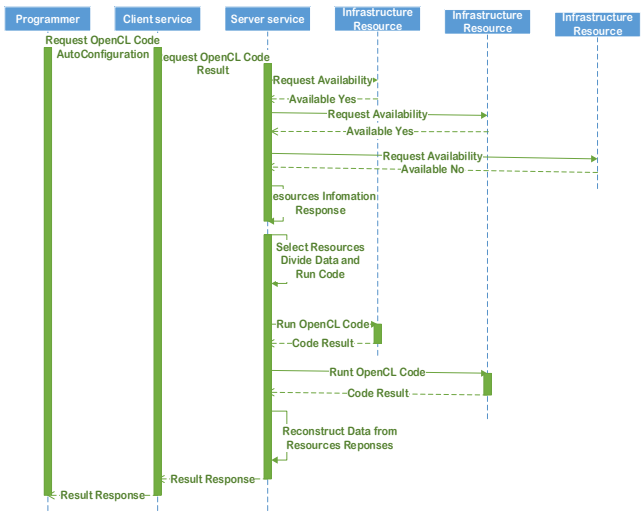

Fig. 3 Automated selection

GPU will take time the service will parallelize the algorithm in the service so it can return the result in more efficient time. The service will handle the load balancing of the algorithm in the cluster, thus, if one of the nodes fails, another one running the same program will take his place.

The service will accept as input information: the algorithm written in any programming language, input data and the type of the expected result. After introducing the input information, the service will analyze the algorithm, will parallelize it and distribute it over the nodes.

IV. PROOF OF CONCEPT

Our POC (proof of concept) is composed of three applications: client service, node service and OpenCL service. The client and node services are written in C#, because it offers the possibility of creating low-level, high-performance asynchronous client server software systems. It also offers the possibility of build software solution for Windows, Linux, OSX, and embedded systems like FPGAs.

The OpenCL service is written in C++ and it is called using several parameters: input type and input data, output type, kernel file and the desired global size and local size of the algorithm. The service analyzes if the system on which is running is capable of allocating memory for the input data, if the GPU supports the chosen level of parallelism and if the GPU is capable of running the kernel. At the end of the execution, the service will provide the output and the execution time of the algorithm.

The algorithm chosen for this POC was the multiplication of two matrices; two inputs matrices A, B and C matrix output Fig. 4. The access pattern for matrix A, above, is strided; for matrix B it is linear. Thus, a tile of matrix A is loaded into shared memory, which support broadcasting of values. A work-group loads a tile of matrix A into shared memory. The

outer loop runs for the number of work-groups that can fit in matrix A's width Fig. 4. The inner loop consists of loading for int4 values from matrix A's tile, and the corresponding values from matrix B's global memory, to compute a partial sum. Each work-item computes four int4s values, and writes to matrix C's global memory. The number of global work-items is width divided by for and height divided by for of matrix C.

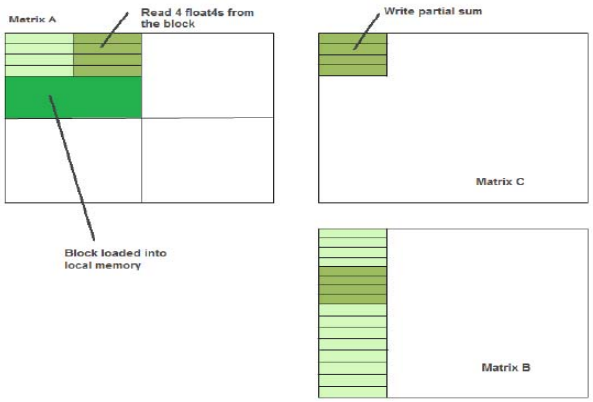

Fig. 4 Matrix multiplication

We test to see how much time does it take to calculate the result of matrix multiplication with different dimension; compering the execution time on the local machine and the time needed to transfer the matrices to the node and receiving the result (Table 1). In this case, it is quickest to send information across the internet to a much powerful node and wait for the result radar then executing on the local machine.

Table 1. Matrix multiplication result CPU vs GPU

Matrix Dimension	CPU time millisecond	GPU Node service time millisecond
1024x1024	14.906	466
2048x2048	304.423	2.292
4096x4096	2.982.733	3.586

V. FEASIBILITY EVALUATION

The most important factor that separates the volunteer approach from our approach is the possibility to make the sharing of resource attractive to the owner of resources. In the smart grid, we have the co-generation model. Here we only have the energy consumption model that generate cost. Further we should evaluate the energy consumption to understand the cost involved in exploiting the resource. If, by chances, the execution time for a specific algorithm decreases considerable, then the potential beneficiary of the faster result would be interested to pay the costs plus interest. It is clear that by parallelizing an algorithm on N computers we will not gain an N acceleration factor. Depending on the code structure, there are well known prediction models like Amdahl's law or Gustafson's law.

Considering the energy consumption in case of executing the job on the client computer over Tc hour, we can then estimate the cost of energy consumption of executing the job on N computers for Tr hours, where Tr hours is greater than Tc/N hours. The difference between energy consumed in these two situations should be covered by the benefits of receiving the result much faster. Farther, we wanted to evaluate this principle by measuring the energy consumption of one possible shared resource. For this, we developed a test bench by using a gamming computer and an external board for measuring the energy consumption. The sensor board contains of a non-invasive current transformer (CT). The signal from CT together with the signal proportional to the supply voltage (by mean of a voltage divider) were acquired with a performant mixed signal oscilloscope (Fig. 5 and Fig. 6). The samples were then processed in Matlab to evaluate the energy consumption for each operating state.

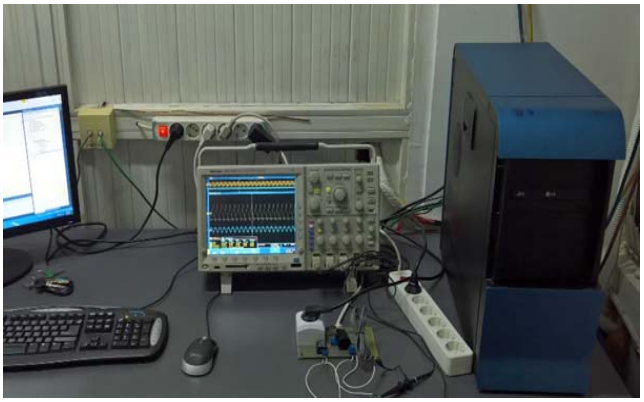

Fig. 5 Test bench for energy consumption evaluation

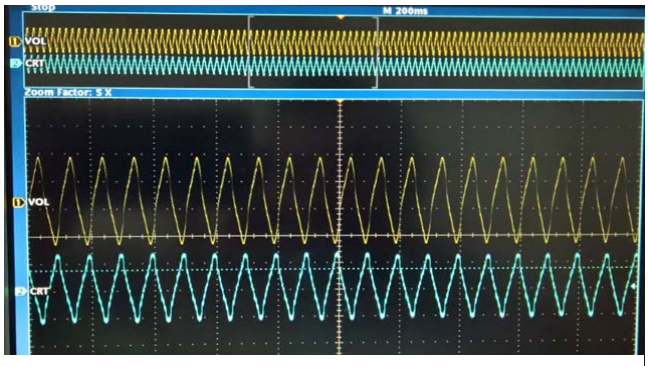

Fig. 6 Voltage (up) and current (down) signals measured when GPUs are used at their peak

The computer used was an i7-960 CPU operating at 3.20GHz, 8GB of RAM DDR3 at 800 MHz and two NVidia GTX 480 in SLI (Scalable Link Interface). We neglected the

energy consumption during sleep mode with the Power on LAN mechanism activated (order of mA). The average energy consumption when the GPUs operate at their peak is about 0,8KWh. Returning to our use case scenario, we can imagine that the required job would take 24 hours on regular laptop (50Wh), consuming a total of 1,2KWh. Assuming that the job could be parallelized on 2 GPUs within 1h running time (240X acceleration), thus consuming 1,6KWh. Then, the 23 hours will value the 0,5KWh cost (the client laptop will run for 1h) plus the interest of the shared resources' owners. We did not address billing considerations like in [16] as the smart grid could provide a mature model also to these aspects of the service.

VI. CONCLUSIONS

Summarizing the work presented in this paper, there are two main ideas closely related. One is the opportunity of developing a HPC infrastructure that resembles volunteer computing, but on a smart grid bases. The second is the possibility to overcome the volunteer approach by evaluating the cost supported by the owners of the shared resources in order for them to be motivated in sharing. Again, also here the smart grid model provides a good example with persons that are interested to invest in alternative energy sources not only to reduce the daily cost but also to cogenerate energy in the grid. The same way, persons that owns powerful computers required for games can share them with an interest.

On the other end, there will be the clients who can rent these expensive resources at low price. Because all the existing solution from volunteer computing that we discussed are dedicated and proprietary, we had to develop our own framework from scratches with the proof of concept as goal.

We didn't use energy consumption values provided by the software applications, because they were not reflecting the energy that the owner of the resource will actually pay for using the resource. Instead, we develop a simple platform but with accurate measurements for evaluating the energy consumption. The results we obtained are very encouraging to continue. The next steps will be to implement the entire framework, evaluate overall performance and, then, develop an application programing interface (API) for easily exploiting the grid.

REFERENCES

[1] Mocanu, Şt., R. Din, D. Saru, C. Popa , "Using Graphics Processing Units for Accelerated Information Retrieval", in Studies in Informatics and Control, vol.23, Issue 3, 2014, pp. 249-256

[2] GPU□ACCELERATED APPLICATIONS, http://images.nvidia.com/content/tesla/pdf/Apps-Catalog-March-2016.pdf

[3] D. E. Baz, T. T. Nguyen, G. Jourjon and T. Rakotoarivelo, "HPC Applications Deployment on Distributed Heterogeneous Computing Platforms via OMF, OML and P2PDC," 2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, Torino, 2014, pp. 617-623.

[4] Korpela, Eric; Dan Werthimer; David Anderson; Jeff Cobb; Matt Lebofsky. "SETI@home — Massively Distributed Computing for SETI". Computing in Science & Engineering. 3: 78–83

[5] S. Peyvandi, R. Ahmad and M. N. Zakaria, "Association among independent and dependent factors of host in volunteer computing," 2014 International Conference on Computer and Information Sciences (ICCOINS), Kuala Lumpur, 2014, pp. 1-6.

[6] B. Javadi, D. Kondo, J. M. Vincent and D. P. Anderson, "Discovering Statistical Models of Availability in Large Distributed Systems: An Empirical Study of SETI@home," in IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 11, pp. 1896-1903, Nov. 2011.

[7] B. Javadi, K. Matawie and D. P. Anderson, "Modeling and analysis of resources availability in volunteer computing systems," 2013 IEEE 32nd International Performance Computing and Communications Conference (IPCCC), San Diego, CA, 2013, pp. 1-9.

[8] S. Peyvandi, R. Ahmad and N. Zakaria, "End hosts clustering based on resources availability in Volunteer Computing," 2013 International Conference on Research and Innovation in Information Systems (ICRIIS), Kuala Lumpur, 2013, pp. 588-593.

[9] Federico Silla, Javier Prades, Sergio Iserte, and Carlos Reano: Remote GPU Virtualization: Is It Useful? 2016 IEEE 2nd International Workshop on High-Performance Interconnection Networks Towards the Exascale and Big-Data Era.

[10] NVIDIA, CUDA Runtime API 7.0, 2015

[11] KHRONOS, OpenCL Runtime API 2.2, 2016

[12] T. Y. Liang and Y. W. Chang, "GridCuda: A Grid-Enabled CUDA Programming Toolkit," 2011 IEEE Workshops of International Conference on Advanced Information Networking and Applications, Biopolis, 2011, pp. 141-146.

[13] M. Oikawa, A. Kawai, K. Nomura, K. Yasuoka, K. Yoshikawa and T. Narumi, "DS-CUDA: A Middleware to Use Many GPUs in the Cloud Environment," 2012 SC Companion: High Performance Computing, Networking Storage and Analysis, Salt Lake City, UT, 2012, pp. 1207-1214.

[14] Antonio J. Peña, Carlos Reaño, Federico Silla, Rafael Mayo, Enrique S. Quintana-Ortí, José Duato, A complete and efficient CUDA-sharing solution for HPC clusters, Parallel Computing, Volume 40, Issue 10, December 2014, Pages 574-588, ISSN 0167-8191

[15] Munteanu, G., Ş. Mocanu, D. Saru, "GPGPU optimized parallel implementation of AES using C++AMP", in Journal of Control Engineering and Applied Informatics, Vol. 17, No. 2, pp. 73-81, 2015

[16] Z. Dong, Y. C. Lee and A. Y. Zomaya, "Crowdware: A Framework for GPU-Based Public-Resource Computing with Energy-Aware Incentive Mechanism," 2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom), Vancouver, BC, 2015, pp. 266-273.