# Boosting Productivy through Efficient Resource Management

Sergio Iserte
Universistat Jaume I
Castelló de la Plana, Spain
siserte@uji.es

## ABSTRACT

This PhD dissertation has addressed, from two different approaches, the improvement of data centers productivity through an efficient resource management.

Firstly, we study the combination of GPU remote virtualization technologies with workload managers in HPC clusters. On the same basis, cloud computing environments also deal with GPUs, since virtual machines can be equipped with these devices.

Secondly, we design, implement and analyze a malleability solution capable of resizing jobs on-the-fly. To ease the adoption of malleability in scientific applications, we provide two strategies, from an OpenMP-like programming model approach and from an MPI-friendly syntax.

## KEYWORDS

Resource management, Remote GPU Virtualization, GPU cloudification, Process malleability, Job reconfiguration, Cluster productivity

**ACM Reference Format:**
Sergio Iserte. 2018. Boosting Productivy through Efficient Resource Management. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

Nowadays, the vast majority of applications are developed parallel, ready to run on multiple computational units. To leverage parallelism further, massively parallel architectures and different distributed programming models and are utilized.

On the one hand, Graphic Processor Units (GPUs) present several handicaps, such as its high acquisition cost, but also its high power consumption even in an idle state. GPU virtualization techniques allow sharing devices among applications and remote access. For instance, a non-accelerated application may allocate all the cores of a node blocking the use of the GPU from other job. Applying those approaches may incur in a positive impact on the GPU utilization, what is immediately translated into a higher throughput and a lower energy consumption.

On the other hand, distributed applications usually are implemented using the MPI programming paradigm. This methodology enables "malleability" in order to change on-the-fly the number of processes of a given job allowing it to continue its execution with a new process layout. This reconfiguration also assumes data-redistribution among processes, what involves a higher effort from users willing to leverage malleability. For this reason, resource management tasks have become crucial when refereeing to resource utilization and global throughput.


*Conference'17, July 2017, Washington, DC, USA*

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 2 BACKGROUND

In the Sergio Iserte's master thesis [7] we presented an extension to Slurm [8] that support remote GPU virtualization. With this extension Slurm becomes aware of the fact that the assignment would no longer be constrained by the GPU kernels having to be executed in the same node where the invoking application is mapped to. The goal was thus to create a GPU virtualization-aware job scheduler which in turn allowed applications to leverage all the cluster GPUs, independently of their location. However, the master's thesis lacks of a thorough analysis of the Slurm's extension in day-to-day HPC scenarios.

Several solutions have been developed to be specifically used within VMs, such as, for example, gVirtuS [4], vCUDA [24], GViM [6], Shadowfax [19], gCloud [1], Vgris [22]. Although some of these projects are showing a high rate of matureness, they have neglected their integration in real cloud platforms, and their management of the GPU resources.

Regarding processes malleability, in the literature we can find, among others, the following solutions: In [2], malleability is addressed with checkpoint/restart mechanisms. Storing and loading files, however, poses a non-negligible overhead versus runtime data redistribution.

In [18], User Level Failure Migration (ULFM) MPI library is leveraged to cause abortions in the processes to use the shrink-recovery mechanism implemented in the library, and then, resume the execution in a new number of processes.

A resizing mechanism based on Charm++ is presented in [5]. In it, the the benefits of resizing a job in terms of both performance and throughput are demonstrated. Nevertheless, this framework apart from being based on a new programming model, they do not address the data redistribution problem during the resizing.

Notwithstanding the variety of existent methods to adopt malleability in parallel scientific applications, none of them combines the features that we consider crucial in order to gain popularity among developers: i) automatic support for data transfers in job reconfigurations; and ii) a friendly syntax imported from parallel programming models such as OpenMP or MPI. Furthermore, for MPI-like syntax, the solutions must be based on the MPI standard without a dependency to any particular MPI library.

## 3 REMOTE GPUS MANAGEMENT

GSaaS [16] (GPU Scheduling as a Service) is a service to cloudify and schedule the access to physical GPUs from VMs, aimed to public cloud infrastructures (see Figure 1). The proposed service relies

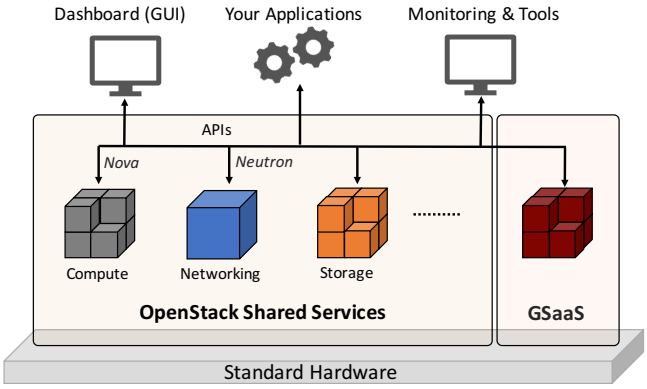


**Figure 1: GSaaS: A service to cloudify and schedule GPU access in OpenStack.**

on a set of 3 distributed components, integrated between the cloud infrastructure and the physical GPUs (see Figure 2). In this sense, rCUDA [20][21] enables remote access to the GPUs, which are previously cloudified and scheduled by RODVR and GPGPUMS [9], respectively. The main benefits achieved by GSaaS are: flexible

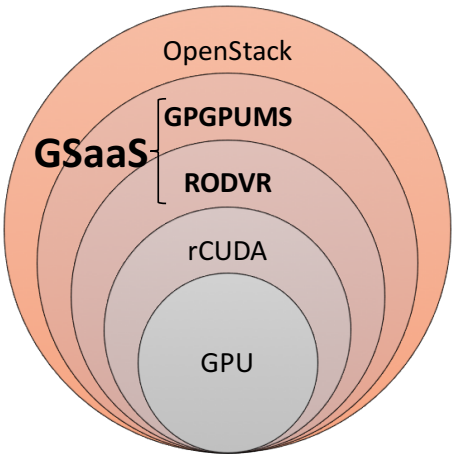


**Figure 2: Technologies integration scheme.**

scheduling of GPU resources, decoupling of the interface between the client and the rCUDA server, masking the location of the GPUs, preventing GPU unauthorized accesses and detaching VM traffic from GPU traffic by using a dedicated network. Besides, the proposed solution automates the configuration of its distributed components.

## 4 DYNAMIC JOB RECONFIGURATION

DMR (Dynamic Management of Resources) [12] is a complete solution for malleability which provides the necessary mechanisms to implement malleable applications and process adaptive workloads.

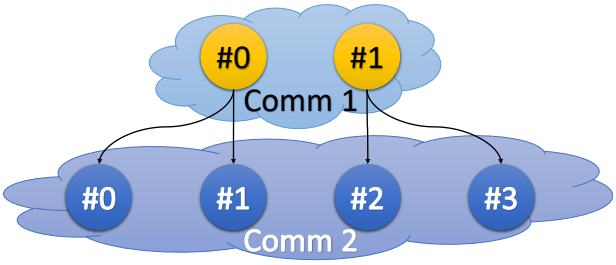


**Figure 3: Job expansion.**

DMR simplifies the development of malleable applications by providing 2 frameworks with syntaxes based on OpenMP and MPI, in order to reach a wider audience. Furthermore, DMR is responsible for spawning the new processes, redistribute the data among processes and resuming the execution where the it was left for reconfiguring. Figure 3 shows a job expansion example where a 2-process job is expanded to 4 processes.

In order to dynamically adapt a workload, we need two main tools: (1) a resource manager system (RMS) capable of reallocating the resources assigned to a job; and (2) a parallel runtime to rescale an application. DMR connects these components by developing a communication layer between the RMS and the runtime (see Figure 4). Specifically, DMR enables the communication with Nanos++ (the OmpSs runtime) and the RMS Slurm. The RMS is aware of

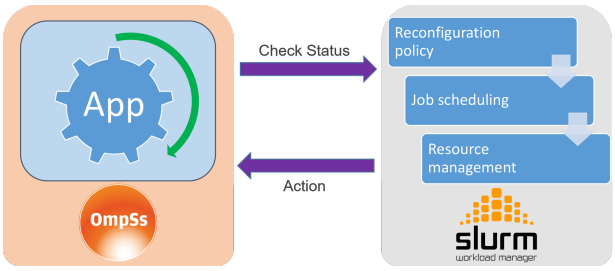


**Figure 4: Scheme of the interaction between the RMS and the runtime.**

resource utilization and the queue of pending jobs. When an application is in execution, it periodically contacts the RMS, through the runtime, communicating its rescaling willingness (to expand or shrink the current number of allocated nodes). The RMS inspects the global status of the system to decide whether to initiate any rescaling action, and communicates this decision to the runtime. If the framework determines that a reconfiguration action is beneficial and possible, the RMS, runtime and application will collaborate to continue the execution of the application scaled to a different number of processes.

## 5 EVALUATION

Combining rCUDA and Slurm, we prove that we can boost the throughput of a GPU-enabled cluster by placing the accelerators in any location, depending on the system restrictions or user necessities. Specifically, we have experienced reductions of 48% in

the workload completion time and of 40% in the energy consumption with respect to traditionally CUDA-based configurations. This more efficient management of the GPUs has also been translated into GPU utilization, doubling the baseline provided with native CUDA [26], [27], [25] and [17] .

Similarly, we have experimented with rCUDA in cloud environments, concluding that public cloud solutions such as AWS are not ready for HPC at the moment when the experiments were performed (June 2015) [10] and [9]. However, thanks to our approach, users may reduce their economic budget by customizing the number of GPUs they need for the VMs, thus skipping the restrictions of AWS when it comes to CUDA-enabled VMs. This new approach presents a series of new working modes for CUDA-enabled VMs that may be leveraged to increase the performance of GPU parallel applications, as well as to provide more flexibility regarding VM-GPU assignation [16].

On the other hand, although malleability has already proven its advantages in HPC workloads, in this dissertation we have demonstrated that our solution not only can improve the cluster productivity, but also the coding productivity. Our solution [14] and [15] has demonstrated to reduce the completion time of the jobs in a workload by reducing their waiting time in the queue.

Our evaluation was performed using 129 nodes from the Marenostrum IV supercomputer at Barcelona Supercomputing Center. Each node in this facility is equipped with 2 Intel Xeon Platinum 8160 sockets (24 cores at 2.10 GHz each) for a total of 48 cores with 96 GB of RAM. The nodes are interconnected through a 100 Gbit/s Intel Omni-Path network.

Concretely, our experiments have proven a reduction of 75% in the needed time to complete a workload when combining moldability and malleability [13], [12] and [11].

Dynamically reconfiguring jobs and reallocating resources with DMR has favored an increased utilization of the underlying resources, what has been translated in to an increase of the global throughput and remarkable reduction in the amount of energy needed to process the workload. Our studies have proven that it is not necessary to port all the system applications into malleable; just adapting the right type of applications, the throughput may be dramatically increased and the energy consumption reduced more than 75%.

# 6 CONCLUSIONS

From a high-throughput perspective, this PhD dissertation provides a deep insight into 2 different techniques for productivity in HPC facilities. Those techniques have been applied in various scenarios, demonstrating their usefulness and appropriateness for High-throughput computing (HTC).

## 6.1 Ongoing Work

Thanks to all the work done in this thesis, which has supposed an interesting source of ideas, we have prepared projects for students and established collaborations with other researchers:

- With the Unit of Medicine at Universitat Jaume I (Spain), we are studying how to increase their productivity and reduce their costs by performing part of their genetic alignment (using the malleable version of HPG-aligner, developed in this thesis) in our HPC facilities.
- With the Czestochowa University of Technology (Poland), we are developing the malleable version of MPDATA [23]. MPDATA is the main module of a multiscale fluid model, which supposes an innovative solver in the field of numerical modeling of multiscale atmospheric flows. Furthermore, MPDATA is CUDA-capable and with this application we aim to experiment the effect of a malleable application in a GPGPU virtulized cluster, combining the 2 approaches for productivity studied in this dissertation.

## 6.2 Future Work

In a near future we plan to integrate the remote GPU scheduling in containerized environments as a microservice. This new approach aims to improve the computational elasticity in cloud computing infrastructures.

Furthermore, we understand that the next natural step in process malleability corresponds to the integration of intranode malleability tools, such as the dynamic load balancing (DLB) [3]. This extension may improve the productivity when:

- A user wants to send a second application in the resources previously allocated to another application. Instead of requesting more resources the user may launch the analysis application in the same resources where the simulation was running.
- Slurm detects that a node is being underutilized and decides to reduce the number of resources assigned to the processes running on that node.

Process malleability may be also applied in resilience scenarios where administrative restrictions, such as the maintenance of the infrastructure or power capping capabilities and unexpected node failures could cause the abruptly termination of many jobs losing all their unsaved progress. For this purpose, the communication with the RMS is crucial since it is aware of the node status. For instance, a node status transition to *drain* may imply the migration of jobs from that node to others where the execution could be resumed.

# ACKNOWLEDGEMENTS

The author would like to thank his PhD advisors Rafael Mayo and Antonio J. Peña for their valuable and insightful guidance. This work is supported by the projects TIN2014-53495-R and TIN2015-65316-P from MINECO and FEDER.

# REFERENCES

[1] K. M. Diab, M. M. Rafique, and M. Hefeeda. 2013. Dynamic Sharing of GPUs in Cloud Systems. In *IEEE Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW)*. 947–954.

[2] K. El Maghraoui, Travis J Desell, Boleslaw K Szymanski, and Carlos A Varela. 2009. Malleable Iterative MPI Applications. *Concurrency and Computation: Practice and Experience* 21, 3 (March 2009), 393–413.

[3] Marta Garcia-Gasulla. 2017. *Dynamic Load Balancing for Hybrid Applications*. Ph.D. Dissertation. Universitat Politècnica de Catalunya.

[4] Giulio Giunta, Raffaele Montella, Giuseppe Agrillo, and Giuseppe Coviello. 2010. A GPGPU Transparent Virtualization Component for High Performance Computing Clouds. Springer, Berlin, Heidelberg, 379–391.

[5] Abhishek Gupta, Bilge Acun, Osman Sarood, and Laxmikant V Kalé. 2014. Towards Realizing the Potential of Malleable Jobs. In *21st International Conference on High Performance Computing (HiPC)*.

[6] Vishakha Gupta, Ada Gavrilovska, Karsten Schwan, Harshvardhan Kharche, Niraj Tolia, Vanish Talwar, and Parthasarathy Ranganathan. 2009. GViM: GPU-accelerated virtual machines. In *Proceedings of the 3rd ACM Workshop on System-level Virtualization for High Performance Computing - HPCVirt '09*. ACM Press, New York, New York, USA, 17–24.

[7] Sergio Iserte. 2014. *Un Gestor de GPUs Remotas para Clusters HPC*. Master's thesis. Universitat Jaume I, Castelló de la Plana, Spain.

[8] Sergio Iserte, Adrian Castello, Rafael Mayo, Enrique Quintana-Orti, Federico S. Silla, Jose Duato, Carlos Reano, and Javier Prades. 2014. SLURM Support for Remote GPU Virtualization: Implementation and Performance Study. In *IEEE 26th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. IEEE, 318–325. https://doi.org/10.1109/SBAC-PAD.2014.49

[9] Sergio Iserte, Francisco J. Clemente-Castelló, Adrián Castelló, Rafael Mayo, and Enrique S. Quintana-Ortí. 2016. Enabling GPU Virtualization in Cloud Environments. In *6th International Conference on Cloud Computing and Services Science (CLOSER)*. SCITEPRESS - Science and and Technology Publications, 249–256.

[10] Sergio Iserte, Francisco J. Clemente-Castelló, Rafael Mayo, and Enrique S. Quintana-Ortí. 2015. GPU Virtualization in the Cloud. In *Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems (ACACES)*. Fiuggi (Italy).

[11] Sergio Iserte, Héctor Martínez, Sergio Barrachina, Castillo Maribel, Rafael Mayo, and Antonio J. Peña. 2018. Dynamic Reconfiguration of Non-iterative Scientific Applications: A Case Study with HPG-aligner. *International Journal of High Computing Performance Application (IJHPCA)* (jul 2018).

[12] Sergio Iserte, Rafael Mayo, Enrique S. Quintana-Ortí, Vicenç Beltran, and Antonio J. Peña. 2017. Efficient Scalable Computing through Flexible Applications and Adaptive Workloads. In *46th International Conference on Parallel Processing Workshops (ICPPW)*. IEEE, 180–189.

[13] Sergio Iserte, Rafael Mayo, Enrique S. Quintana-Ortí, Vicenç Beltran, and Antonio J. Peña. 2018. DMR API: Improving the Cluster Productivity by Turning Applications into Malleable. *Parallel Computing (ParCo)* (jul 2018).

[14] Sergio Iserte, Rafael Mayo, Enrique S. Quintana-Ortí, and Antonio J. Peña. 2018. High-throughput computation through MPI Malleability. In *Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems (ACACES)*. Fiuggi (Italy).

[15] Sergio Iserte, Antonio J. Peña, Rafael Mayo, Enrique S. Quintana-Ortí, and Vicenç Beltran. 2016. Dynamic Management of Resource Allocation for OmpSs Jobs. In *1st PhD Symposium on Sustainable Ultrascale Computing Systems (NESUS PhD)*. Timisoara (Romania), 55–58.

[16] Sergio Iserte, Raúl Peña-Ortiz, Juan Gutiérrez-Aguado, Jose M. Claver, and Rafael Mayo. 2018. GSaaS: A Service to Cloudify and Schedule GPUs. *IEEE Access* (jul 2018).

[17] Sergio Iserte, Javier Prades, Carlos Reano, and Federico Silla. 2016. Increasing the Performance of Data Centers by Combining Remote GPU Virtualization with Slurm. In *16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*. IEEE, 98–101.

[18] Pierre Lemarinier, Khalid Hasanov, Srikumar Venugopal, and Kostas Katrinis. 2016. Architecting Malleable MPI Applications for Priority-driven Adaptive Scheduling. In *Proceedings of the 23rd European MPI Users' Group Meeting (EuroMPI)*. 74–81.

[19] Alexander M. Merritt, Alexander M. Merritt, Ada Gavrilovska, Vishakha Gupta, Karsten Schwan, and Abhishek Verma. [n. d.]. Shadowfax: Scaling in Heterogeneous Cluster Systems via GPGPU Assemblies. ([n. d.]).

[20] Antonio J Peña. 2013. *Virtualization of Accelerators in High Performance Clusters*. Ph.D. Dissertation. Universitat Jaume I, Castellon, Spain.

[21] Antonio J Peña, Carlos Reaño, Federico Silla, Rafael Mayo, Enrique S Quintana-Ortí, and José Duato. 2014. A Complete and Efficient CUDA-sharing Solution for HPC Clusters. *Parallel Comput.* 40, 10 (2014), 574–588.

[22] Zhengwei Qi, Jianguo Yao, Chao Zhang, Miao Yu, Zhizhou Yang, and Haibing Guan. 2014. VGRIS: Virtualized GPU Resource Isolation and Scheduling in Cloud Gaming. *ACM Trans. Archit. Code Optim.* 11, 2, Article 17 (July 2014), 25 pages.

[23] Bogdan Rosa, Lukasz Szustak, Andrzej A. Wyszogrodzki, Krzysztof Rojek, Damian K. Wójcik, and Roman Wyrzykowski. 2015. Adaptation of Multidimensional Positive Definite Advection Transport Algorithm to Modern High-Performance Computing Platforms. *International Journal of Modeling and Optimization* 5, 3 (jun 2015), 171–176.

[24] Lin Shi, Hao Chen, Jianhua Sun, and Kenli Li. 2012. vCUDA: GPU-Accelerated High-Performance Computing in Virtual Machines. *IEEE Trans. Comput.* 61, 6 (jun 2012), 804–816.

[25] Federico Silla, Sergio Iserte, Carlos Reaño, and Javier Prades. 2017. On the benefits of the remote GPU virtualization mechanism: The rCUDA case. *Concurrency and Computation: Practice and Experience (CCPE)* (2017), e4072.

[26] Federico Silla, Javier Prades, Sergio Iserte, and Carlos Reano. 2016. Remote GPU Virtualization: Is It Useful?. In *2nd IEEE International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB)*. IEEE, 41–48.

[27] Federico Silla, Carlos Reaño, Javier Prades, and Sergio Iserte. 2016. Benefits of remote GPU virtualization: the rCUDA perspective. In *GPU Technology Conference (GTC)*. Silicon Valley, California (USA).