On Exploiting Data Locality for Iterative Mapreduce Applications in Hybrid Clouds

Francisco Clemente-Castello, Bogdan Nicolae, Rafael Mayo, Juan Carlos Fernandez, M. Mustafa Rafique

On Exploiting Data Locality for Iterative MapReduce Applications in Hybrid Clouds

Francisco J. Clemente-Castelló
Universidad Jaume I, Spain
fclement@uji.es

Bogdan Nicolae
Huawei Research, Munich, Germany
bogdan.nicolae@acm.org

Rafael Mayo
Universidad Jaume I, Spain
mayo@uji.es

Juan Carlos Fernández
Universidad Jaume I, Spain
jfernand@uji.es

M. Mustafa Rafique
IBM Research, Ireland
mustafa.rafique@ie.ibm.com

ABSTRACT

Hybrid cloud bursting (i.e., leasing temporary off-premise cloud resources to boost the capacity during peak utilization), has made significant impact especially for big data analytics, where the explosion of data sizes and increasingly complex computations frequently leads to insufficient local data center capacity. Cloud bursting however introduces a major challenge to runtime systems due to the limited throughput and high latency of data transfers between on-premise and off-premise resources (weak link). This issue and how to address it is not well understood. We contribute with a comprehensive study on what challenges arise in this context, what potential strategies can be applied to address them and what best practices can be leveraged in real-life. Specifically, we focus our study on iterative MapReduce applications, which are a class of large-scale data intensive applications particularly popular on hybrid clouds. In this context, we study how data locality can be leveraged over the weak link both from the storage layer perspective (when and how to move it off-premise) and from the scheduling perspective (when to compute off-premise). We conclude with a brief discussion on how to set up an experimental framework suitable to study the effectiveness of our proposal in future work.

Keywords

Hybrid Cloud; Big Data Analytics; Data locality; I/O and Data Management; Scheduling

1. INTRODUCTION

Due to exploding data sizes and the need to combine multiple data sources, traditional on-premise big data analytics becomes insufficient as the capacity of private-owned data centers cannot accommodate the increasing scale and scope of the applications. To address this issue, *cloud bursting* [1] has seen a rapid increase in popularity among big data analytics users. It is a form of *hybrid cloud computing* that enables temporary boosting of on-premise resources with additional off-premise resources from a public cloud provider for the duration of peak usage, with the purpose of overcoming the limitations of their private data centers in a flexible, pay-as-you-go fashion.

Leveraging hybrid clouds for temporary bursting does not come without challenges: sharing, disseminating and analyzing the data sets may result in frequent large-scale data movements between the on-premise and the off-premise sites. This raises an important challenge for cloud bursting, as users typically access public clouds via *high-latency, low-throughput wide-area networks*, making inter-site data transfers up to an order of magnitude slower than intra-site data transfers. For this reason, we refer to the inter-site link for the rest of this paper as the *weak link*. To make the problem even worse, studies show that the inter-datacenter traffic is expected to triple in the following years [2, 3].

In this context, popular large-scale big data analytics frameworks like Hadoop [4] or Spark [5] have not achieved an acceptable level of scalable, cross-datacenter implementation. Specifically, design choices like coupling the analytics runtime with the storage layer to enable scheduling the computation close to the data are fundamental aspects that haven proven effective in single data centers. However, in a cloud bursting scenario, the newly provisioned off-premise resources do not hold any data to begin with. Therefore, data locality cannot be directly taken advantage of: it is necessary to ship the data to the off-premise resources over the weak link before the extra computational capability can be leveraged. This aspect has proven particularly difficult, to the point where many users tend to avoid spreading the same application over both on-premise and off-premise resources, preferring to split their workload in independent, loosely coupled jobs that can be confined to a single site.

Nevertheless, many real-life big data analytics applications are iterative in nature: they run the same job over and over again while relying on the same input data and intermediate data from the previous iterations. For this class of applications, data locality can be leveraged over and over again once the input data has been replicated off-premise. Thus, the problem of running the same application over a mix of both on-premise and off-premise resources can be considered from a new perspective. However, the current understanding of the challenges in this particular context

BDCAT'16, December 06-09, 2016, Shanghai, China

DOI: http://dx.doi.org/10.1145/3006299.3006329

and how to address them is limited. To this end, our paper contributes with an analysis of such challenges and potential solutions to address them. We focus our study on MapReduce [6] as a reference analytics framework. Specifically, we make the following contributions in this paper:

- We outline the fundamental issues in operating iterative MapReduce over hybrid cloud setups comprising both on- and off-premise VMs. In particular, we highlight the issue of lacking data locality on the off-premise part and associated consequences in terms of I/O interactions with the underlying storage layer, task scheduling and data shuffling (Section 3).
- We propose two complementary strategies to address the identified issues and trade-offs: (1) extend the storage layer to the off-premise part and rebalance the data; (2) locality-enforced scheduling to avoid redundant data transfer over the weak link. Furthermore, we detail a real-life implementation of our proposal through a reference prototype specifically designed for running in a hybrid cloud setup (Section 4).
- We discuss how to create an experimental framework suitable to study the effectiveness of the strategies for real-life iterative MapReduce application (Section 5).

2. RELATED WORK

MapReduce applications have been studied extensively on cloud computing platforms [7, 8]. However, unlike hybrid clouds, where the on-premise resources are limited, the general assumption is that the whole application is running remotely and has access to virtually unlimited computing resources thanks to elasticity.

Several efforts are focused on improving the performance of MapReduce frameworks for hybrid environments. HybridMR [9] proposes a solution to leverage hybrid desktop grids and external voluntary computing nodes. However, the aspect of expanding and shrinking a MapReduce setup dynamically is not considered. HadoopDB [10] proposes a hybrid system combining a Hadoop deployment and a parallel database system to simultaneously leverage the resilience and scalability of MapReduce together with the performance and efficiency of parallel databases. In [11–13] GPUs are used to improve the performance of MapReduce applications in accelerator-enabled clusters. These techniques uses hybrid computing to improve the performance of MapReduce applications, but the hybrid aspect is on the computational side rather than the networking side.

Bicer [14] tackles the challenge of data analysis in a hybrid cloud where data is pre-partitioned and stored across on-premise and off-premise resources. It proposes a user-transparent runtime where a *head* component acts as MapReduce job scheduler between the local and the remote cloud systems. A similar assumption about data being already available remotely off-premise is made in [15], where the focus is on speeding up deadline-constrained MapReduce applications. Unlike our approach, data transfers between on-premise and off-premise nodes are not a central concern.

Authors in [16] propose a model based on workload factoring to enable proactive workload management. It is based on an algorithm specifically designed to detect frequently accessed data items, which is then used as a prediction of when to scale out to public clouds beyond the on-premise resources to achieve load balancing. This approach is used in applications where the processes are loosely coupled (i.e., little need for inter-process communication) and highly sensitive to the number of requests per time unit (e.g. video streaming). By contrast, the application class we are targeting has non-trivial data dependencies.

Scheduling strategies for Hadoop MapReduce applications running on hybrid clouds have been proposed in numerous studies. In [17] the authors focus on the problem of how map and reduce processes are split over hybrid cloud platforms. Heintz et al. [18] shows strategies for scheduling map tasks over distributed cloud platforms in order to minimize the performance degradation that can result from large communication times between the platforms.

Optimizing the cost-performance trade-off through elasticity in the context of clouds is a another prominent directions. User-transparent elasticity for storage space [19] and I/O throughput [20] was shown to lead to a massive reduction in data-related costs, while maintaining similar levels of performance to the case when resources are over-provisioned to accommodate the peak utilization.

Performance studies have frequently pointed out the negative impact of the weak link between the on-premise and off-premise resources of hybrid clouds on the performance and scalability of big data analytics applications. Ohnaga et al. [21] focus on the performance of Hadoop applications in particular. Roman et al. [22] focus on Spark and point out significant overhead in the shuffle phase when the bandwidth between the on-premise and off-premise resources is sufficiently small. This is unlike previous studies that emphasized low end-impact of network communications in Spark [23]. How to leverage performance metrics about both the application and the hybrid cloud platform to predict completion time under variable scale-out scenarios is shown in [24], where the focus is on the map phase of iterative MapReduce applications.

This work focuses on iterative MapReduce applications running on hybrid clouds, where the data is initially on-premise only. Our specific focus is on how to efficiently exploit data locality while migrating data off-premise over the weak link. To the best of our knowledge, we are the first to explore iterative MapReduce applications in this challenging context.

3. DATA LOCALITY CHALLENGES AND TRADE-OFFS

The MapReduce paradigm is specifically designed to facilitate a high degree of data parallelism: massive amounts of data are transformed in an embarrassingly parallel fashion in a *map* phase, after which they are aggregated in a *reduce* phase. Unsurprisingly, this leads to a highly concurrent I/O intensive access pattern that can quickly overwhelm the networking infrastructure with remote data transfers. To address this issue, *data locality* awareness is a key feature of MapReduce: the storage layer is co-located with the runtime on the same nodes and is designed to expose the location of the data blocks, effectively enabling the scheduler to bring the computation close to the data and to avoid a majority of the storage-related network traffic. However, in a cloud bursting scenario, the premises for leveraging data locality are different: the input data is present only on the on-premise VMs initially, so it has to be shipped to the off-

premise VMs before they can contribute to the computation.

Furthermore, despite heavy emphasis on exploiting data locality efficiently, this optimization can be applied for the map phase only. During the reduce phase, which is responsible for the aggregation of the intermediate results, the runtime faces significant overhead related to the synchronization and collection of relevant data pieces (i.e., *data shuffling*). At scale, this involves concurrent remote transfers of significant data amounts between the nodes, which cannot be avoided and leads to stress on the networking infrastructure.

Given that hybrid clouds face the problem of reduced latency and throughput between the on-premise and off-premise VMs due to the weak link, both aspects discussed above lead to important challenges and trade-offs, which we develop next.

3.1 Off-Premise Data Replication

Since the input data is present initially only on the on-premise VMs, any map task that is running off-premise needs to access the on-premise data, which involves a data transfer over the weak link. However, there are multiple alternatives to achieve this.

An obvious choice is to simply leave the input data on-premise and pull it on-demand from the off-premise map tasks. This approach has two advantages: (1) it works out of the box with no modification necessary to the default runtime; (2) it overlaps the I/O with the computation, as map tasks can start off-premise right away without any need to wait for data transfers. On the other hand, there is also a major disadvantage: if the iterative application re-uses the input data blocks, they will be transferred over the weak link multiple times unnecessarily, which leads to performance degradation.

Another choice is to replicate the input data off-premise before running the MapReduce application. Using this approach, the MapReduce runtime can fully exploit locality as if it were running on a single cluster, which also avoids the problem of sending the same input block over the weak link repeatedly. However, creating replicas off-premise is a time-consuming process that adds to the overall completion time. Furthermore, it also leads to an extra off-premise storage space utilization.

To avoid waiting for the replication process to finish, a third option is to use asynchronous replication. Using this approach, the input data is shipped off-premise at the same time as the map tasks are running, under the assumption that a majority map tasks will benefit from the data locality. However, the background data transfers create additional overhead that interferes with the computation and may create enough slowdown to offset the benefit of starting the computation right away.

3.2 Scheduling

By default, Hadoop tries to leverage locality as much as possible: when no slot is available to run a map task on the same node as the input data, Hadoop schedules that map task with preference on the same rack (rack-local) and failing to do so it picks any other free node for execution. This behavior is justified by the relatively small penalty for running a remote map task: the network links often keep up with the disk throughput and there are relatively few other concurrent remote data transfers over the network, since most map tasks can benefit from locality.

If the data is replicated off-premise before running the MapReduce application, then the original scheduling strategy will behave similarly in a hybrid cloud too, since most of the map tasks will benefit from locality and the weak link will not be stressed. However, the situation is more complex in the other two cases. If the data is kept on-premise only and there is a sizable amount of off-premise VMs, then the reverse is true: most of the map tasks (regardless of the iteration) will not benefit from locality and the weak link will be constantly under stress. This leads to severe performance degradation for the off-premise map tasks when the input data is large and creates an imbalance in the execution waves, which in turn has a negative overall impact due to the fact that the reduce phase needs to wait for the map phase to finish.

If the data is replicated off-premise in an asynchronous fashion, then the default scheduling strategy has a higher chance of leveraging locality for off-premise map tasks as some input data blocks may already be present off-premise when the map task is scheduled. However, if that is not the case, then a similar situation arises as with the case when the data is kept on-premise, albeit at a smaller degree. For this reason, it is important to explore alternative scheduling strategies when using asynchronous replication.

3.3 Reduce-like Aggregation

Once a map task finishes, its result is written in temporary file on the local disk of the node where it was executed. This intermediate data is then collected from all nodes during the reduce phase by each reducer. At scale, this leads to an all-to-all mapper-reducer communication pattern that stresses the weak link. This overhead cannot be avoided, because intermediate data is present both on-premise and off-premise. However, the scheduling of the mappers affects the distribution of the intermediate data, which in turn affects the way in which the weak link is stressed.

The computational aspect of the reduce phase (sorting and applying the user-defined aggregation) is not affected by weak link, because it is performed locally where the intermediate data from all map tasks are collected. However, once the output of the reduce phase was successfully generated, it is typically written to the storage layer. If the storage layer is deployed on-premise only, then the weak link may be stressed again when each reducer needs to write its output. This aspect is application-specific and depends on the proportionality of input to output size ratio (some MapReduce applications have trivial output such as a counter).

4. DATA LOCALITY STRATEGIES

In this section, we propose two complementary strategies to address the data locality challenges discussed in Section 3.

4.1 HDFS Replica Rebalancing

We propose a solution that addresses the problem of off-premise data replication based on *rack awareness*, which was discussed in Section 3.2. Specifically, rack-awareness is also a feature of *HDFS* [25], the default storage layer of Hadoop.

Originally, rack awareness was designed to enhance replication-based fault tolerance: each node that joins the HDFS group can be designated to be part of a sub-group (rack). Then, whenever a data block is written to HDFS, it is replicated (typically three times) both inside the rack where the write

originated from, as well as at least on one node outside that rack. Using this approach, HDFS can survive catastrophic failures where a whole rack would fail simultaneously.

When a new node joins HDFS in a rack, it is initially empty. This leads to an imbalanced data distribution where older nodes tend to be heavier loaded than more recently added nodes, which leads to competition of the heavier loaded nodes and thus negatively affects the ability to leverage data locality efficiently. To this end, HDFS employs a rebalancing mechanism that tries to reshuffle the data block replicas according to the rack awareness constraint.

It is this feature that we leverage in the context of hybrid clouds. We define two racks for the on-premise and off-premise VMs that host HDFS. Then, when the off-premise VMs are added to the Hadoop deployment, we extend the HDFS deployment to the off-premise VMs and start the rebalancing process. This effectively leads to the migration of one replica from each block stored on-premise to the off-premise side. The rebalancing can be either blocking (wait for it to finish before running the MapReduce application) or asynchronous (start it together with the MapReduce application), each corresponding to the scenarios described in Section 3.1.

The main advantage of this approach is that it minimizes the amount of data transferred off-premise (a single replica) to achieve full potential of exploiting the data locality, while maintaining the resilience constraints. Moreover, it is a non-invasive solution that works out-of-the box without deviating from the standard HDFS, which is a major concern for many users.

4.2 Scheduling Based on Enforced Rack-Locality

The default Hadoop scheduler uses data locality only as a preferential matching mechanism between map tasks and free slots. However, if asynchronous HDFS rebalancing is employed, it will suffer from the issues mentioned in Section 3.2: a map task may be scheduled off-premise before a replica of its corresponding data block was migrated, which triggers a pull and leads to a double data transfer of the same block. In this case, it may be beneficial to delay the scheduling of such off-premise map tasks, under the assumption that avoiding stress on the weak link leads to a a smaller overall overhead. To this end, we propose an enforced rack-locality scheduling policy: a map task will never be scheduled to a rack where there is no replica of its input data. This effectively leads to the desired behavior in our case: a map task will never be scheduled off-premise if a replica was not already migrated there. We implemented this policy in Hadoop by modifying the Resource Manager to make use of the *relaxLocality* flag. Thus, unlike the HDFS replica rebalancing, this is an intrusive modification to Hadoop that requires the user to deploy a custom version.

5. EXPERIMENTAL FRAMEWORK

Understanding the impact of the different strategies and how every map or reduce task has been scheduled is non-trivial and requires additional instrumentation and tools to analyze the logs and results. To this end, we have developed a complete solution to aggregate important metrics for every executed job: map, shuffle, sort and reduce times; HDFS data distribution; task distribution between on-premise and off-premise; system-level statistics such as CPU, I/O network, I/O disk and memory utilization.

Our profiling tool extracts information from a combination of Hadoop counters, Hadoop logs, the Rumen tool [26] and Systat [27] to generate two kind of files: a profiling statistic file and a trace file. Our tool is able to generate that trace file that can be processed using the visualization tool Paraver [28] to visually represent how the tasks have been distributed among the available slots of the on-premise and off-premise VMs.

In terms of experimental setup, it is important to use an environment as close as possible to a real hybrid cloud setup. However, at the same time, it is important for the study to be able to control the environment. Specifically, running unrelated VMs together with the VMs of the applications to be studied introduces noise and interference in the system that makes it hard to interpret the results. Furthermore, it is important to be able to control the capacity of the weak link in order to study how the behavior of the application changes when the weak link capacity changes.

To enable such a controllable environment, we build two separate IaaS clouds (*on-premise* and *off-premise*). We used OpenStack Icehouse as the reference middleware for both IaaS clouds. To create a realistic networking model, we rely on OpenStack's *Neutron*: all communication outside of the same cloud is passing through a dedicated network node that acts as a proxy. Thus, in a hybrid OpenStack setup the weak link corresponds to the link capacity between the two proxies. Note that the VM instances of the same cloud are not subject to this limitation: they communicate with each other via the links of their compute node hosts.

6. CONCLUSIONS

Hybrid cloud bursting is opening new opportunities to leverage big data analytics in a cost-effective fashion despite overflowing the capacity of the computational resources available on-premise. Such an approach is however highly challenging due to the limited ability to directly leverage data locality and the limited network link capacity between the on-premise and the off-premise resources.

This paper has contributed with an analysis of the challenges that arise when running iterative MapReduce applications in hybrid cloud bursting. To deal with these challenges, it proposed a series of complementary strategies that are either directly leveraged or implemented in Hadoop. Finally, it proposed an experimental framework to study in-depth the behavior of iterative MapReduce applications in practice using a controlled environment.

As a next step we plan to run extensive experiments to evaluate our proposal in practice. Furthemore, we also envision a refinement of our proposal in two directions: (1) an improved data migration scheme that prioritizes the chunks to be moved off-premise based on scheduling decisions; (2) an improved reduce algorithm specifically designed to minimize the stress on the weak link. To this end, we envison a subsequent extended version of the paper.

Acknowledgments

This work was supported by the MINECO/CICYT projects TIN2011-23283, TIN2014-53495-R and FEDER.

7. REFERENCES

[1] T. Guo, U. Sharma, T. Wood, S. Sahu, and P. Shenoy, "Seagull: Intelligent cloud bursting for enterprise

applications," in *Proceedings of the 2012 USENIX Conference on Annual Technical Conference*, ser. USENIX ATC'12, Berkeley, CA, USA, 2012, pp. 33–33.

[2] T. Kosar, E. Arslan, B. Ross, and B. Zhang, "Storkcloud: Data transfer scheduling and optimization as a service," in *ScienceCloud'13: Proceedings of the 4th ACM Workshop on Scientific Cloud Computing*, New York, New York, USA, 2013, pp. 29–36.

[3] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez, "Inter-datacenter bulk transfers with netstitcher," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 74–85, Aug. 2011.

[4] T. White, *Hadoop: The Definitive Guide*. USA: Yahoo! Press, 2010.

[5] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *HotCloud'10: Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, Boston, MA, 2010, pp. 10–10.

[6] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[7] T. Gunarathne, T.-L. Wu, J. Qiu, and G. Fox, "Mapreduce in the clouds for science," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*. IEEE, 2010, pp. 565–572.

[8] X. Zhang, L. T. Yang, C. Liu, and J. Chen, "A scalable two-phase top-down specialization approach for data anonymization using mapreduce on cloud," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 25, no. 2, pp. 363–373, 2014.

[9] B. Sharma, T. Wood, and C. R. Das, "Hybridmr: A hierarchical mapreduce scheduler for hybrid data centers," in *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on*. IEEE, 2013, pp. 102–111.

[10] A. Abouzeid, K. Bajda-Pawlikowski, D. Abadi, A. Silberschatz, and A. Rasin, "Hadoopdb: an architectural hybrid of mapreduce and dbms technologies for analytical workloads," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 922–933, 2009.

[11] K. Shirahata, H. Sato, and S. Matsuoka, "Hybrid map task scheduling for gpu-based heterogeneous clusters," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*. IEEE, 2010, pp. 733–740.

[12] M. M. Rafique, A. R. Butt, and D. S. Nikolopoulos, "A capabilities-aware framework for using computational accelerators in data-intensive computing," *J. Parallel Distrib. Comput.*, vol. 71, no. 2, pp. 185–197, 2011.

[13] M. M. Rafique, B. Rose, A. R. Butt, and D. S. Nikolopoulos, "Cellmr: A framework for supporting mapreduce on asymmetric cell-based clusters," in *Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*. IEEE, 2009, pp. 1–12.

[14] T. Bicer, D. Chiu, and G. Agrawal, "A framework for data-intensive computing with cloud bursting," in *CLUSTER '11: The 2011 IEEE International Conference on Cluster Computing*, 2011, pp. 169–177.

[15] M. Mattess, R. N. Calheiros, and R. Buyya, "Scaling mapreduce applications across hybrid clouds to meet soft deadlines," in *AINA'13: IEEE 27th International Conference on Advanced Information Networking and Applications*, Barcelona, Spain, 2013, pp. 629–636.

[16] H. Zhang, G. Jiang, K. Yoshihira, and H. Chen, "Proactive workload management in hybrid cloud computing," *Network and Service Management, IEEE Transactions on*, vol. 11, no. 1, pp. 90–100, 2014.

[17] M. Cardosa, C. Wang, A. Nangia, A. Chandra, and J. Weissman, "Exploring mapreduce efficiency with highly-distributed data," in *Proceedings of the Second International Workshop on MapReduce and Its Applications*, ser. MapReduce '11. New York, NY, USA: ACM, 2011, pp. 27–34.

[18] B. Heintz, A. Chandra, and J. Weissman, *Cross-Phase Optimization in MapReduce*. New York, NY: Springer New York, 2014, pp. 277–302.

[19] B. Nicolae, P. Riteau, and K. Keahey, "Bursting the Cloud Data Bubble: Towards Transparent Storage Elasticity in IaaS Clouds," in *IPDPS '14: Proc. 28th IEEE International Parallel and Distributed Processing Symposium*, Phoenix, USA, 2014, pp. 135–144.

[20] ——, "Transparent Throughput Elasticity for IaaS Cloud Storage Using Guest-Side Block-Level Caching," in *UCC'14: 7th IEEE/ACM International Conference on Utility and Cloud Computing*, London, UK, 2014.

[21] H. Ohnaga, K. Aida, and O. Abdul-Rahman, "Performance of hadoop application on hybrid cloud," in *2015 International Conference on Cloud Computing Research and Innovation (ICCCRI)*, Oct 2015, pp. 130–138.

[22] R.-I. Roman, B. Nicolae, A. Costan, and G. Antoniu, "Understanding Spark Performance in Hybrid and Multi-Site Clouds," in *BDAC'15: 6th International Workshop on Big Data Analytics*, Austin, United States, 2015.

[23] K. Ousterhout, R. Rasti, S. Ratnasamy, S. Shenker, and B.-G. Chun, "Making sense of performance in data analytics frameworks," in *NSDI'15: The 12th USENIX Conference on Networked Systems Design and Implementation*, Oakland, USA, 2015, pp. 293–307.

[24] F. J. Clemente-Castelló, B. Nicolae, K. Katrinis, M. M. Rafique, R. Mayo, J. C. Fernández, and D. Loreti, "Enabling big data analytics in the hybrid cloud using iterative mapreduce," in *UCC'15: 8th International Conference on Utility and Cloud Computing*, 2015, pp. 290–299.

[25] K. Shvachko, H. Huang, S. Radia, and R. Chansler, "The Hadoop distributed file system," in *26th IEEE (MSST2010) Symposium on Massive Storage Systems and Technologies*, May 2010.

[26] Apache. (26-06-2016) Apache rumen.

[27] S. Godard, "Sysstat: System performance tools for the linux os, 2004."

[28] V. Pillet, J. Labarta, T. Cortes, and S. Girona, "Paraver: A tool to visualize and analyze parallel code," in *Proceedings of WoTUG-18: Transputer and occam Developments*, vol. 44. mar, 1995, pp. 17–31.