HAL
archives-ouvertes.fr

Evaluation of Data Locality Strategies for Hybrid Cloud Bursting of Iterative MapReduce

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

▶ **To cite this version:**

Francisco Clemente-Castello, Bogdan Nicolae, M. Mustafa Rafique, Rafael Mayo, Juan Carlos Fernandez. Evaluation of Data Locality Strategies for Hybrid Cloud Bursting of Iterative MapReduce. CCGrid'17: 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, May 2017, Madrid, Spain. <hal-01469991>

HAL Id: hal-01469991
https://hal.inria.fr/hal-01469991

Submitted on 17 Feb 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Evaluation of Data Locality Strategies for Hybrid Cloud Bursting of Iterative MapReduce

Francisco J. Clemente-Castelló*, Bogdan Nicolae†, M. Mustafa Rafique†,
Rafael Mayo*, Juan Carlos Fernández*

*Universidad Jaume I, Spain. Email: {fclement, mayo, jfernand}@uji.es

†IBM Research, Ireland. Email: {bogdan.nicolae, mustafa.rafique}@ie.ibm.com

***Abstract*—Hybrid cloud bursting (i.e., leasing temporary off-premise cloud resources to boost the overall capacity during peak utilization) is a popular and cost-effective way to deal with the increasing complexity of big data analytics. It is particularly promising for iterative MapReduce applications that reuse massive amounts of input data at each iteration, which compensates for the high overhead and cost of concurrent data transfers from the on-premise to the off-premise VMs over a weak inter-site link that is of limited capacity. In this paper we study how to combine various MapReduce data locality techniques designed for hybrid cloud bursting in order to achieve scalability for iterative MapReduce applications in a cost-effective fashion. This is a non-trivial problem due to the complex interaction between the data movements over the weak link and the scheduling of computational tasks that have to adapt to the shifting data distribution. We show that using the right combination of techniques, iterative MapReduce applications can scale well in a hybrid cloud bursting scenario and come even close to the scalability observed in single sites.**

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

I. Introduction

One important class of big data analytics applications is iterative refinement, i.e. running the same job over and over again to improve an intermediate result by reusing the initial data and an intermediate state updated at each successive iteration. However, due to exploding data sizes and the need to combine multiple data sources, traditional iterative refinement using 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 private data centers in a flexible, pay-as-you-go fashion.

However, leveraging hybrid cloud bursting for iterative big data applications is non-trivial: it involves highly concurrent access to massive data sizes, which is bottlenecked by the limited capacity of the network link between the on-premise and off-premise computational resources (often orders of magnitude smaller than single-site links). This *weak link* has important consequences in terms of how core big data design principles can be leveraged, in particular exploiting *data locality awareness* to ship the computation close to the data.

The current understanding of the challenges in this particular context and how to address them is limited. Our previous work [2] introduces an analysis of such challenges and potential solutions to address them. We focused our study on MapReduce [3] as a reference analytics framework and pointed out two major issues: the need to replicate the initial input data off-premise to avoid repeated concurrent data transfers over the weak link at each iteration, while adapting the scheduling of computation tasks such that it can take advantage of this dynamic data locality created by the data transfers. To address these issues, we introduced an asynchronous data migration strategy and a corresponding task scheduling policy.

This work completes our previous work with a study on how to combine the data migration and scheduling strategies for real life iterative MapReduce iterations in the context of hybrid cloud bursting. Our focus is on experimental results that enable insight into the complex interactions between the data transfers and task scheduling, which reveal novel, non-trivial conclusions regarding best practices. To this end, we run scalability experiments with two real-life iterative MapReduce applications at scale and discuss the various trade-offs for different hybrid cloud bursting strategies when compared with a baseline on-premise only scenario.

II. Related Work

Several efforts are focused on improving the performance of MapReduce frameworks for hybrid environments. HybridMR [4] 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 [5] 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 [6], [7] 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 [8] tackles the challenge of data analysis in a hybrid cloud where data is pre-partitioned and stored across on-premise and off-premise resources. A similar assumption about data being already available remotely off-premise is made in [9], where the focus is on speeding up deadline-constrained MapReduce applications. Our work by contrast assumes no data is available off-premise and focuses on the implications of this aspect.

Optimizing the cost-performance trade-off through elasticity in the context of clouds is a another prominent directions. User-transparent elasticity for storage space [10] and I/O throughput [11] 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. Such techniques are highly useful as a complement to improve the elasticity aspect of hybrid cloud bursting.

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. [12] focus on the performance of Hadoop applications in particular. Roman et al. [13] 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. These studies confirm our assumptions about the weak link.

Scheduling strategies for Hadoop MapReduce applications running on hybrid clouds have been proposed in numerous studies. In [14] the authors focus on the problem of how map and reduce processes are split over hybrid cloud platforms. Heintz et al. [15] 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. Our own previous work introduces a proposal for data migration (asynchronous rebalancing of the MapReduce storage layer) and scheduling strategies (enforced locality to avoid data pulls over the weak link) that are specifically designed to address the weak link bottleneck in hybrid cloud bursting [2], [16]. This work focuses on the evaluation of such techniques and the resulting trade-offs that can be used to establish best practices.

III. Cloud Bursting Data Locality Challenges and Strategies

This section briefly introduces the challenges of running iterative MapReduce applications in hybrid cloud bursting scenarios and revisits two complementary strategies discussed in greater detail in our previous work [2].

The MapReduce model is designed to facilitate a high degree of data parallelism. In a hybrid cloud bursting scenario, this translates to a highly concurrent I/O access pattern that can quickly overwhelm the weak link with remote data transfers. Although *data locality awareness* is a core MapReduce design principle, in a hybrid cloud bursting scenario it cannot be leveraged directly, because the input data is present only on the on-premise VMs initially. Thus, there is a need transfer the input data off-premise, otherwise map tasks will be scheduled off-premise without local access to input data and need to pull it over the weak link. This is especially important for iterative MapReduce applications that reuse input data for each iteration, which amplifies the overhead of pulling data over the weak link. However, waiting for the data to arrive off-premise also introduces a delay in the computation. Thus, there is a need to address this trade-off both in terms of how to perform the data transfers off-premise without delaying the computation, as well as how to adapt the computation (schedule the MapReduce tasks) such that it can take advantage of this dynamic data locality created by the data transfers. Specifically, we propose two complementary strategies:

***HDFS Replica Rebalancing**:* we extend the storage layer (HDFS deployment) to the off-premise VMs (by defining a separate rack) and start a rack-aware rebalancing process that will migrate one replica for each data chunk off-premise. This generates the maximum data locality (all data available both on-premise and off-premise eventually) with minimum additional cost (only one copy off-premise). The rebalancing can be either blocking or asynchronous.

***Enforced locality**:* we extend the Hadoop scheduling policy with enforced rack-locality where off-premise map tasks are not assigned off-premise unless a data chunk was already migrated there by the HDFS rebalancing process. This scheduling policy can be turned on or off as needed.

IV. Evaluation

A. Experimental Setup

The experiments for this work were performed on the *Kinton* testbed of the HPC&A group based at Universidad Jaume I (4 Xeon E5-2630v3 nodes with 64 GB RAM, 1 TB local storage, 1 Gbps interconnect). We deploy up to 4 VMs on each node (4 virtual cores, 16 GB RAM, 1 Gbps virtual interconnect).

We use two representative real-life iterative MapReduce applications from the Mahout 0.10 distribution: *K-Means* (map-intensive application) and *PageRank* (both map-intensive and reduce-intensive).

We evaluate these applications in two configurations. First, we use a baseline configuration that consists of a single on-premise OpenStack cloud. Then, we configure a hybrid cloud bursting scenario running as two separate on-premise/off-premise OpenStack clouds. All communication outside of the same cloud is passing through a dedicated network node (Neutron) that acts as a proxy and is part of the default OpenStack distribution. Thus, the weak link between two OpenStack clouds is the capacity between the Neutron proxies. For our experimental purposes we use two settings: 100 Mbps (soft limit) and 1 Gbps (physical maximum).

We compare the baseline configuration with four approaches (representing combinations of the strategies described in Sec-

TABLE I
COMPLETION TIME FOR THE ORIGINAL (3) AND EXTENDED (15) ON-PREMISE ONLY BASELINE SCENARIOS

Application	Completion time: 3 VMs	Completion time: 15 VMs
K-Means	5801.15 s	1516.20 s
PageRank	3150.08 s	777.82 s

tion III) that are running on the hybrid cloud bursting configuration: (1) *No HDFS Off-Premise*: the MapReduce runtime makes use of the off-premise VMs but HDFS is deployed only on-premise; (2) *Blocking Rebalance*: HDFS is deployed on the off-premise VMs, the data blocks are rebalanced and then, after this process finished, the MapReduce application is executed; (3) *Plain Asynchronous Rebalance*: HDFS is deployed on the off-premise VMs and the rebalancing starts in the background at the same time as the application with locality enforced-scheduling turned off; (4) *Local Asynchronous Rebalance*: same as plain asynchronous rebalance except our custom locality-enforced scheduling policy is turned on.

We target strong scalability in our experiments, i.e., we study the performance of the applications (same problem size) for an increasing number of VMs (from 3 VMs up to 15 VMs).

B. Performance Results

To run the two applications described in Section IV-A, we use the input data generator included in Mahout. In case of K-Means, we generate 20 GB of input data and configure the application to run for 30 iterations. In case of PageRank, we generate 2.8 GB of input data and configure it to run for 10 iterations.

First, we run the original on-premise baseline to understand the upper and lower bound of the completion time in a non-hybrid configuration. This experiment indicates if scalability is possible in the first place, and, if so, how much speed-up is possible under ideal conditions (no weak link).The results are summarized in Table I: both applications have excellent scalability potential and a reduction in completion time by 75% is possible in both cases.

Next, we run the hybrid cloud bursting scenario. Tables II and III show a comparison of the four approaches for both 100 Mbps and 1 Gbps weak link. We include a break-down of both: the rebalance time and the application runtime. Note that in the case of blocking rebalance, the completion time is the sum of the application runtime and the duration of the rebalancing. However, in the case of asynchronous rebalance (regardless whether enforced locality is used or not), the completion time is identical with the application runtime. In this case, the rebalance time is interesting to observe as an indication of the duration of interference and competition between the application and the background data migration, which is a cause of degraded performance.

When analyzing the results for K-Means (Table II), it can be observed that local async rebalance is the best strategy for a high inter-cloud network bandwidth (1 Gbps). In this case, the total completion time is very close to the lower bound (15 on-premise VMs). Both blocking and plain async rebalance

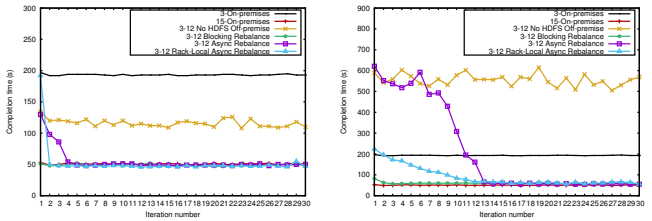
TABLE II
HYBRID CLOUD BURSTING OF K-MEANS (3 VMS ON-PREMISE AND 12 VMS OFF-PREMISE). LOWER IS BETTER.

Approach	Rebalance time (s)	Application runtime (s)	Completion time (s)
1 Gbps			
No HDFS off-premise	N/A	3485.87	3485.87
Blocking Rebalance	216.70	1464.97	1681.67
Plain Async Rebalance	396.90	1647.93	1647.93
Local Async Rebalance	221.1	1571.47	1571.47
100 Mbps			
No HDFS off-premise	N/A	16667.70	16667.70
Blocking Rebalance	2507.50	1803.20	4310.20
Plain Async Rebalance	6160.40	6462.23	6462.23
Local Async Rebalance	2688.6	2713.86	2713.86

TABLE III
HYBRID CLOUD BURSTING OF PAGERANK (3 VMS ON-PREMISE AND 12 VMS OFF-PREMISE). LOWER IS BETTER.

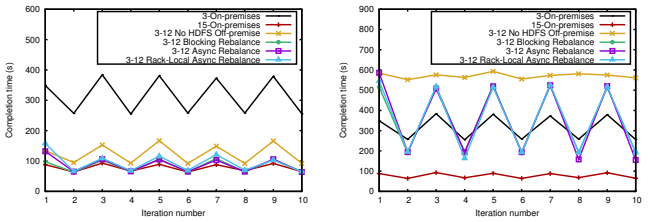
Approach	Rebalance time (s)	Application runtime (s)	Completion time (s)
1 Gbps			
No HDFS off-premise	N/A	1232.77	1232.77
Blocking Rebalance	86.00	838.95	924.90
Plain Async Rebalance	88.00	875.37	875.37
Local Async Rebalance	87.75	944.70	944.70
100 Mbps			
No HDFS off-premise	N/A	5712.00	5712.00
Blocking Rebalance	450.90	3552.60	4003.50
Plain Async Rebalance	527.40	3582.00	3582.00
Local Async Rebalance	569.00	3555.00	3555.00

exhibit a performance degradation of 10%, indicating that interference is relatively small but still significant, which leaves room for improvement when enforcing locality during scheduling. When switching to a modest inter-cloud network bandwidth (100 Mbps), the weak link limitation has a more pronounced impact. Repeated data transfers in the case of no HDFS off-premise lead to a performance level that is worse than the upper bound obtained when running the application on the original on-premise VMs, indicating that this option is not viable. Plain async rebalance experiences a long duration of interference, to the point where the background rebalance overlaps almost fully with the application runtime. In this case, avoiding interference is critical, as demonstrated by the fact that blocking rebalance performs much better. Nevertheless,



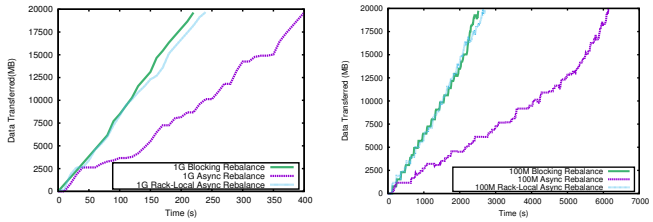
(a) 1 Gbps inter-cloud network link (b) 100 Mbps inter-cloud network link

Fig. 1. K-Means: comparison of per-iteration completion time between the four hybrid cloud approaches and the lower/upper bound baseline. Lower is better.



(a) 1 Gbps inter-cloud network. (b) 100 Mbps inter-cloud network.

Fig. 2. PageRank: comparison of per-iteration completion time between the four hybrid cloud approaches and the lower/upper bound baseline. Lower is better.



(a) 1 Gbps inter-cloud network link. (b) 100 Mbps inter-cloud network link.

Fig. 3. Evolution of K-Means rebalance: Amount of migrated data over time. Higher is better.

the overall winner here is again local async rebalance: thanks to the delayed scheduling of the off-premise map tasks, interference is greatly reduced, which enables it to perform much better than blocking rebalance and to achieve 53% of the higher bound.

For PageRank (Table III), we observe that with a high inter-cloud network bandwidth of 1 Gbps we obtain a total completion time 12% slower than the lower bound. As expected, no HDFS off-premise is the worst performer. However, this time the winner is plain async rebalance, followed by blocking rebalance and finally local async rebalance. This behavior can be explained by the fact that the background rebalance is less stressing on the inter-cloud link (the rebalance time is very close), which leads to a situation where it does not pay off to delay scheduling map tasks off-premise or to wait for the rebalance to finish. When switching to a modest inter-cloud network bandwidth (100 Mbps), the weak link limitation also has a pronounced impact like for K-Means. However, there is a trade-off in this case: plain async rebalance leads to a lower rebalance time but longer application runtime, while local async rebalance optimizes the application runtime but leads to longer rebalance. Thus, overall both approaches are very close. Blocking rebalance is significantly slower.

Note that all approaches are unfeasible as a practical solution in this case, as they perform above the upper bound. This can be explained by the fact that PageRank is reduce-intensive, which means the map phase has less overall impact and the weak link is a major bottleneck that cannot be avoided.

C. Zoom on Iteration-Level Granularity

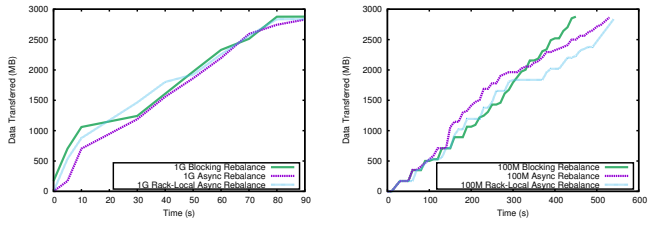
The results presented in Section IV-B are better understood when analyzing the completion time for each iteration, as depicted in Figure 1 and Figure 2. In the case of K-Means, using a 1 Gbps high throughput inter-cloud link leads to a case where no HDFS off-premise is faster than the upper bound, despite constant need to pull data off-premise over the weak link. Since blocking rebalance waits for the off-premise data migration to finish, per-iteration completion time remains flat and is overlapping with the lower bound, which is the expected behavior. However, in the case of plain async rebalance and local async rebalance the interference caused by the background data migration is clearly visible in the beginning, where the iterations take much longer to finish compared with the rest of the runtime. Interesting to note is that local async rebalance generates more interference in the beginning but only for a single iteration, compared with plain async rebalance where the interference lasts until the forth iteration. Ultimately, this leads for a better overall result in the case of local async rebalance.

Moving on to the 100 Mbps case (Figure 1(b)), it is clearly visible that the no HDFS off-premise approach is slower than the upper bound. This behavior is consistent for the whole duration of the runtime. Blocking rebalance is consistently close to the lower bound but has a high initial penalty due to waiting, which enables plain async rebalance and local async rebalance to outperform it. In the asynchronous case, the effect of interference is more pronounced, lasting in both cases up until the 14^{th} iteration (Figure 1(b)). Furthermore, the benefit of enforcing locality in scheduling is also clearly visible, explaining the difference observed in the overall completion time.

Unlike K-Means, in the case of PageRank we observe a distinctive zig-zag pattern, which corresponds to the two types of jobs that alternate during the iterations. This is visible both for the 1 Gbps and 100 Mbps case (Figures 2(a) and 2(b)). In the case of 1 Gbps, the behavior of the lower bound, upper bound, no HDFS off-premise and blocking rebalance is consistent throughout the runtime. Both plain async rebalance and local async rebalance follow blocking rebalance very closely after the first iteration, meaning the difference observed for the hybrid cloud approaches is mainly due to the interference during the first iteration. In the case of 100 Mbps, all approaches exhibit a consistent behavior with little difference between blocking rebalance and the two asynchronous rebalance approaches, even for the first iteration. While there is potential for speed-up for the even iterations when compared with the upper bound, the same is not true for the odd iterations, which eventually leads to an overall completion time that is worse than the upper bound.

D. Zoom on the Asynchronous Rebalance Progress

In this section, we study how the background data migration progresses during the asynchronous rebalance. This complements the per-iteration analysis described in Section IV-C, offering more insight on the slowdown observed during the first iterations.



(a) 1 Gbps inter-cloud network link. (b) 100 Mbps inter-cloud network link.

Fig. 4. Evolution of PageRank rebalance: Amount of migrated data over time. Higher is better.

The results for K-Means are depicted in Figure 3. As expected, the fastest is blocking rebalance approach, due to lack of interference with the application runtime. Note that local async rebalance is very close to blocking rebalance. This is explained by the fact that local async rebalance uses the weak link almost entirely to progress with the background data migration. A completely different behavior is observable for plain async rebalance. Since a large number of map tasks need to pull input blocks concurrently with the background data migration, the latter is slowed down visibly between 2×–3×. The overall behavior of all the three approaches is very similar both for the 1 Gbps and 100 Mbps scenarios.

In the case of PageRank (Figure 4), the similarity between 1 Gbps and 100 Mbps is also visible, however there is a marked difference compared with K-Means. The plain async rebalance is much closer to the other two approaches, which can be explained by the fact that the overall size of the input data is smaller. This is reflected also in the rebalance time. Another important difference is that the progress of the background data migration is much slower, which is visible from the fact that the slope of the curves is more gentle compared with K-Means.

V. Conclusions

Hybrid cloud bursting opens new opportunities to leverage big data analytics in a cost-effective fashion when the on-premise resources are insufficient. However, such an approach is 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 VMs.

This paper has studied the feasibility of hybrid cloud bursting for iterative MapReduce applications. Results show that obtaining a speed-up is possible for applications that scale on-premise, but specialized data locality strategies are necessary to achieve it. In particular, asynchronous rebalancing consistently produces better results than blocking rebalancing. Enforcing locality in the scheduling policy is particularly effective for the case when the weak link is of limited capacity. Furthermore, using the right combination can even match the performance levels of similar single-site Hadoop deployments.

Encouraged by these promising results, we see several avenues for future work. First, the rebalancing and the scheduling policy is currently decoupled, but it would be interesting to explore how to prioritize the order of data migration based on the scheduling decisions. Second, the reduce tasks are scheduling using the default policy but a more complex policy that minimizes stress on the weak link could lead to significant improvement in the application runtime.

Acknowledgments

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

References

[1] T. Guo, U. Sharma, T. Wood, S. Sahu, and P. Shenoy, "Seagull: Intelligent cloud bursting for enterprise applications," in *USENIX ATC'12: USENIX Annual Technical Conference*, Boston, USA, 2012, pp. 33–33.

[2] F. J. Clemente-Castelló, B. Nicolae, R. Mayo, J. C. Fernández, and M. M. Rafique, "On exploiting data locality for iterative mapreduce applications in hybrid clouds," in *BDCAT'16: 3rd IEEE/ACM International Conference on Big Data Computing, Applications and Technologies*, Shanghai, China, 2016, pp. 118–122.

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

[4] B. Sharma, T. Wood, and C. R. Das, "Hybridmr: A hierarchical mapreduce scheduler for hybrid data centers," in *ICDCS'13: 33rd International Conference on Distributed Computing Systems*, Philadelphia, USA, 2013, pp. 102–111.

[5] 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.

[6] K. Shirahata, H. Sato, and S. Matsuoka, "Hybrid map task scheduling for gpu-based heterogeneous clusters," in *CloudCom'10: 2nd International Conference on Cloud Computing Technology and Science*, 2010, pp. 733–740.

[7] 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.

[8] 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*, Newport Beach, USA, 2011, pp. 169–177.

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

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

[11] ——, "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.

[12] H. Ohnaga, K. Aida, and O. Abdul-Rahman, "Performance of hadoop application on hybrid cloud," in *ICCCRI '15: International Conference on Cloud Computing Research and Innovation*, 2015, pp. 130–138.

[13] 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 (in conjunction with SC15)*, Austin, USA, 2015.

[14] M. Cardosa, C. Wang, A. Nangia, A. Chandra, and J. Weissman, "Exploring mapreduce efficiency with highly-distributed data," in *MapReduce '11: 2nd International Workshop on MapReduce and Its Applications*, San Jose, USA, 2011, pp. 27–34.

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

[16] 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 IEEE/ACM International Conference on Utility and Cloud Computing*, Limassol, Cyprus, 2015, pp. 290–299.