Understanding Vertical Scalability of I/O Virtualization for MapReduce Workloads: Challenges and Opportunities

Bogdan Nicolae

IBM Research, Dublin, Ireland
bogdan.nicolae@ie.ibm.com

Abstract. As the explosion of data sizes continues to push the limits of our abilities to efficiently store and process big data, next generation big data systems face multiple challenges. One such important challenge relates to the limited scalability of I/O, a determining factor in the overall performance of big data applications. Although paradigms like MapReduce have long been used to take advantage of local disks and avoid data movements over the network as much as possible, with increasing core count per node, local storage comes under increasing I/O pressure itself and prompts the need to equip nodes with multiple disks. However, given the rising need to virtualize large datacenters in order to provide a more flexible allocation and consolidation of physical resources (transforming them into public or private/hybrid clouds), the following questions arise: is it possible to take advantage of multiple local disks at virtual machine (VM) level in order to speed up big data analytics? If so, what are the best practices to achieve a high virtualized aggregated I/O throughput? This paper aims to answer these questions in the context of I/O intensive MapReduce workloads: it analyzes and characterizes their behavior under different virtualization scenarios in order to propose best practices for current approaches and speculate on future areas of improvement.

1 Introduction

Big Data analytics has enabled unprecedented insight into scientific, social and business challenges. Major advances in almost all fields (i.e. meteorology, genomics, complex physics simulations, environmental research, social networking and dynamics, financial forecasting, etc.) were possible thanks to increasing volume and diversity of data gathered and archived form a variety of sources: sensors, experimental data, mobile devices, etc.

Not surprisingly, the rapid rate at which data sizes are growing has prompted the need for bigger and faster systems / techniques capable to perform big data analytics efficiently at an increasingly larger scale. Today, clusters of tens of thousands of nodes are a common occurrence. However, advances that make such systems possible are not homogeneous: while adding more computational power was demonstrated feasible both in terms cost and scalability, the I/O abilities in terms of networking and storage are lagging behind. Given the data-intensive nature of big data workloads, I/O performance is a determining factor in the overall performance of the applications, thus becoming a critical focus area.

An important technique to limit the impact of I/O bottlenecks is to avoid data movements as much as possible, which conserves network bandwidth and thus helps achieve horizontal scalability. Several big data paradigms were developed around this concept, with MapReduce [1] and its open-source implementation Hadoop [2] being widely adopted in both academia and industry. Two key design principles enable MapReduce to avoid data movements. First, it forces the users to think their application in an embarrassingly parallel fashion that transforms the input as much as possible into a digested form (map phase) over which an aggregation is performed (reduce phase). Thus, during the map phase no extra network traffic is generated due to synchronization. Second, it departs from the traditional model of decoupling storage from computation, taking advantage of local storage to schedule the computation close to the data if possible, which again avoids network traffic.

Although avoiding data movements is a powerful concept that helps conserve network bandwidth, at the same time it shifts the burden of I/O on the local storage. With horizontal scalability increasingly difficult to achieve and attention turning to vertical approaches (i.e. more cores per node), the I/O pressure grows large enough to introduce the need for multiple local disks. Thanks to its embarrassingly parallel design and a streaming I/O model that favors adding new data over modifying old data, MapReduce can easily take advantage of multiple disks to achieve a high aggregated I/O throughput, which is a feature already implemented in Hadoop.

As nodes become increasingly complex and expensive to build and maintain in large numbers, big data systems become prohibitively expensive for most users. In this context, IaaS cloud computing emerged as a key technology to enable users to rent computational resources on-demand, paying only for what they have used. Thanks to virtualization, any user can easily create a large virtual big data cluster with the click of a button. However, how to efficiently map virtual resources to physical resources is a difficult challenge, especially when considering the increasing size and complexity of the nodes. In particular, the problem of how to virtualize multiple local disks efficiently to achieve a high aggregated I/O throughput is not well understood yet it is a crucial step in enabling efficient big data analytics on IaaS clouds.

This paper aims to understand the problem mentioned above. What makes it particularly challenging is the multitude of factors that play a role in the I/O virtualization overhead (i.e. how many VMs per node, how many virtual disks per VM, virtual disk placement, etc.) that need to be analyzed. This is further augmented by missing functionality in state-of-art cloud middleware to enable users to express placement constraints for virtual disks. We summarize our contributions as follows:

- We introduce an experimental framework that emulates a cloud middleware and enables fine-grain control over the hypervisor in order to easily express mapping constrains for virtual disks. Using this approach, experimental setups can be easily defined and the experimental conditions can be tightly monitored and controlled.
- We experiment with I/O intensive MapReduce workloads in several virtualization setups. In particular, we analyze how well the striping mechanism implemented in Hadoop scales when using a variable number of VMs per node, virtual disks per VM and different virtual disk placement strategies.

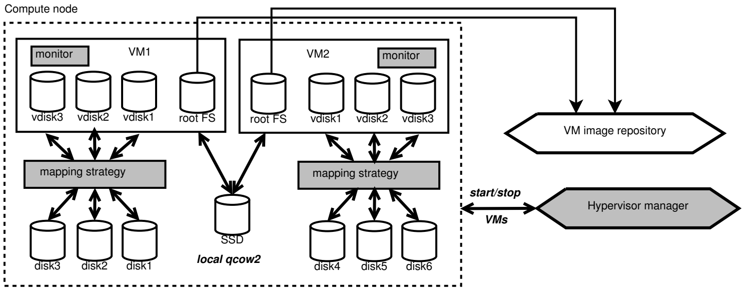


Fig. 1. Architecture of the experimental framework

- Based on the results, we identify several potential areas of improvement and comment on the associated research opportunities.

2 Architecture

To facilitate fine-grain control over the hypervisor and the mapping between virtual and physical resources, we constructed an experimental framework that emulates a typical cloud middleware yet is highly configurable. The simplified architecture of this framework is depicted in Figure 1. For better clarity, the building blocks that are of special interest are emphasized with a darker background.

The *VM image repository* is the storage service responsible to hold the disk image templates for the root file systems of the VM instances. For the purpose of this work, we build our own custom template (based on Debian Sid) with a pre-installed Hadoop environment. All templates are read-only and serve as a base image for locally derived *qcow2* [3] images, where the VM instances are allowed to write into their root file-system. For better performance, the locally derived images are stored on a SSD. Note that Hadoop does not write to the root file-system directly, but uses a separate set of dedicated virtual disks attached to the VM instance.

The *hypervisor manager* is responsible to control all compute nodes and prepare the hypervisors to launch the VM instances in the desired configuration. It can be configured to use a variable number of nodes with a variable number of VMs per node, as well as attach a variable number of virtual disks per VM, according to a *mapping strategy*. For the purpose of this work, we implemented two mapping strategies: (1) *round robin*, which spreads the virtual disks to as many physical disks as possible in order to avoid I/O contention; and (2) *consolidated*, which places as many virtual disks as possible on the same physical disk, thus minimizing the number of required physical disks. Once a configuration was established, the hypervisor manager calculates the number of cores and amount of RAM per VM instance, creates the virtual disks according to the strategy and spawns the VM instances and attaches the virtual disks to them. In a final step, it generates the necessary configuration files for the Hadoop deployment (in particular it, enables striping on the attached virtual disks) and then deploys the Hadoop cluster. Striping is enabled both for HDFS [4], the default storage layer of Hadoop, as well as for the intermediate data that is generated by the mappers and that is used by the reducers as input.

To enable detailed analysis of the results, a *monitor* is deployed on each VM instance to gather performance information at fine granularity (5 seconds). This information includes CPU, memory, networking and virtual disk utilization and is kept both in raw form and in an aggregated fashion that is representative of the whole cluster utilization.

3 Experimental analysis

Using the experimental framework described in the previous section, we study in this section the behavior of I/O intensive MapReduce workloads under different virtualization scenarios, in order to understand what aspects play an important role with respect to performance and scalability.

3.1 Setup

The platform used to run our experiments is a custom testbed consisting of 6 nodes, each equipped with 32 x86_64 cores with support for virtualization, 96 GB of RAM and several Gigabit Ethernet networking interfaces (one of which is used for the experiments). With respect to local storage, each node is equipped with 12 HDD disks (capacity per disk: 1 TB, measured I/O throughput per disk: 160 MB/s) and 2 SDD disks (capacity per disk: 256GB, measured I/O throughput per disk: 430 MB/s).

The hypervisor running on all compute nodes is QEMU/KVM 1.2.0, while the operating system is a recent Ubuntu distribution. The base image used to deploy the VMs is a recent Debian Sid distribution, on top of which we installed Hadoop 2.0.4. All VM instances share the same base image but write locally into their own derived copy-on-write image (using the *qcow2* format) that is stored on one of the SSDs. All extra virtual disks attached to the VMs that are used by Hadoop in striping mode correspond to *raw* files (256GB) that are stored on the HDDs (according to the mapping strategies presented in Section 2). Each VM formats all its extra virtual disks at boot time using the *ext4* file system. To maximize I/O performance, KVM is configured to run in paravirtualized mode using the *virtio* driver.

3.2 Methodology

We create a series of scenarios that involve a variable number of VMs per node and a variable number of virtual disks per VM using a combination of round robin and consolidated virtual disk mapping strategies.

In all of our experiments, the virtual Hadoop cluster leverages the physical resources of all 6 nodes. More specifically, we reserve 60 GB of RAM and 30 CPU cores for VMs on each node, leaving the rest to deal with jitter and virtualization overhead. These physical resources are leveraged in three configurations: 1 VM / node (using all reserved cores and RAM), 2 VMs / node (each of which is allocated 30 GB of RAM and 15 CPU cores) and 3 VMs / node (each of which is allocated 20 GB of RAM and 10 CPU cores).

Thus, we create a virtual Hadoop cluster of 6, 12 and 18 VMs respectively. For each configuration, we vary the number of virtual disks attached to each VM from 1 up to 8. These disks are mapped to physical HDDs using a per-VM round robin policy, i.e. all VMs on the same node share the same physical disk for their vdisk1, another physical disk for their vdisk2, etc.

The Hadoop deployment itself is performed using YARN. Each VM runs a HDFS *datanode* and a YARN *nodemanager*. As mentioned in Section 2, Hadoop is configured to stripe both its intermediate data and persistent data (i.e. data stored by HDFS). Furthermore, each nodemanager is configured to accept a number of parallel mappers that matches the number of cores allocated to the VM. One of the nodes is chosen as the master and runs the HDFS namespace manager and the YARN resourcemanager, in addition to the datanode and nodemanager.

As a representative workload to perform our study on, we chose the *sort* benchmark, a standard MapReduce workload that is part of the Hadoop distribution. It consists of two phases. In the first phase, a predefined amount of random data is generated using *randomwriter*. This workload writes variable-sized key-value pairs (keys between 10-1000 bytes, values between 0-20000 bytes) directly into HDFS. The mappers do not emit any output and the reduce phase is not used. For the purpose of this work, we configured *randomwriter* to use a total of 180 mappers (i.e. the total number of cores available in the Hadoop cluster), each of which is writing 2GB. After the first phase is complete, all previously generated data is sorted. In this case, the mapper is the predefined IdentityMapper and the reducer is the predefined IdentityReducer, both of which just pass their inputs directly to the output. The sorting itself is achieved thanks to the shuffling that is performed by the MapReduce framework. To parallelize this process as much as possible, we configured *sort* to use a maximum of 180 reducers, which matches the number of mapper slots. Thanks to this minimalist setup in terms of data processing itself, *sort* is heavily data intensive and emphasizes the I/O part, which is the reason why we chose it.

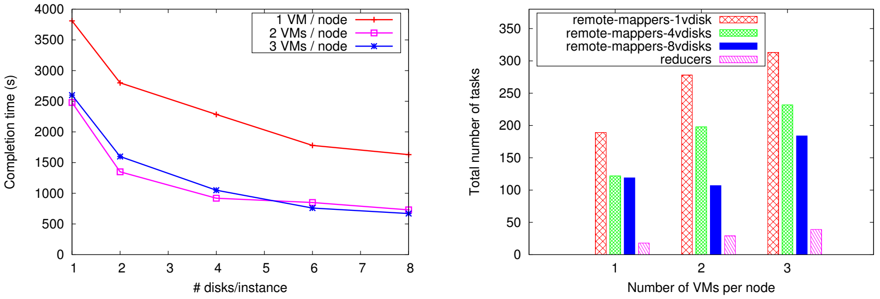
Each experiment consists in fixing the number of VMs per node and the number of virtual disks per VM, then running the *sort* benchmark to completion, while recording cluster-wide monitoring information (using the *monitors* presented in Section 2) and the completion times.

3.3 Results

The completion times for the *sort* benchmark using a variable number of VMs per node and a variable number of disks per VM is depicted in Figure 2(a).

As can be observed, in all three configurations, there is a sharp drop in completion time with an increasing number of virtual disks attached to the VMs. This fact confirms that I/O performance plays a crucial role in the overall application performance: when increasing the number of virtual disks from 1 to 8, a reduction in execution time of up to 70% is observable.

Focusing on the single VM per node scenario, two main factors contribute to the results mentioned above. First, as can be observed in Figure 3(b), an increasing number of virtual disks dramatically lowers the overall I/O pressure in the Hadoop cluster: from an aggregated utilization that tops 100% in the case of 1 virtual disk, a drop to



(a) Completion time

(b) Statistics on remote mappers and reducers

Fig. 2. Performance results for the *sort* benchmark, using a variable number of VMs per node and a variable number of disks per VM

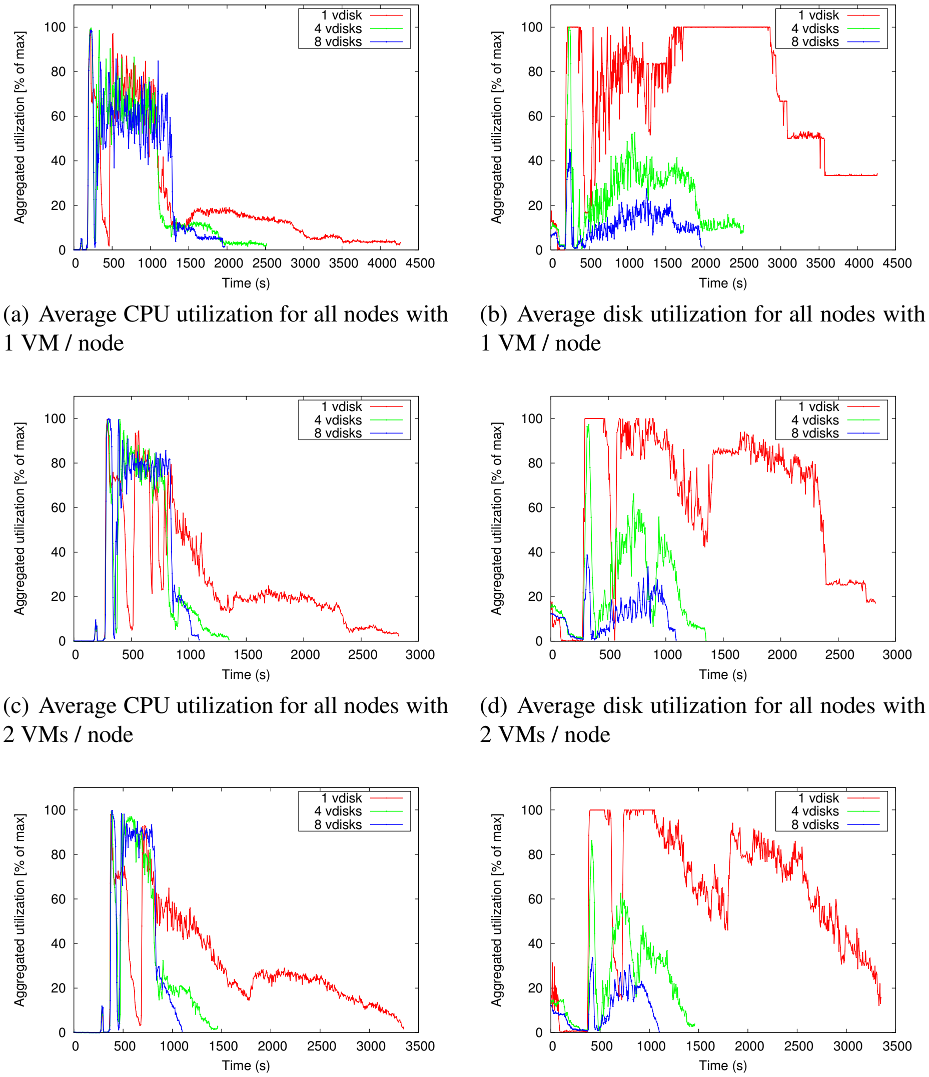
a maximum of 50% and 25% is noticeable in the case of 4 and 8 disks respectively. Figure 3(a) reveals an interesting fact: a lower I/O pressure does not significantly affect the aggregated CPU utilization: all curves follow a similar pattern up to a point when the CPU utilization drops sharply for the rest of the execution: this is the point when the mappers have finished and only reducers are still running. Thus, I/O is the dominating factor during the reduce phase and it oversaturates the disk bandwidth in the case of 1 vdisk, leading to a longer execution time. This is also confirmed by Figure 3(b): using only 1 vdisk results in 100% disk utilization for a significant portion of the reduce phase.

Second, according to Figure 2(b), a different distribution of map and reduce tasks is observable: increasing the number of virtual disks results in improved locality (less remote mappers, which means more data-local mappers) and thus better performance due to less data movement. However, considering the total number of mappers is around 2800, even for 1 vdisk there are less than 7% of remote mappers. Thus, we suspect the impact of improved locality on the overall application performance is small compared to the impact of lower I/O pressure due to more virtual disks.

What scalability is concerned, there is a noticeable drop in the benefits of adding more virtual disks (i.e. 27% reduction from one to two virtual disks compared to 9% reduction from 6 to 8 virtual disks). This is understandable considering the lower overall I/O utilization and it leads to an important observation: while there is considerable performance improvement due to lower I/O pressure, Hadoop striping does not fully leverage the aggregated I/O bandwidth offered by multiple local virtual disks.

Counter-intuitively, adding more VMs per node also benefits overall performance, despite more virtualization overhead and more data movements due to VMs on the same node being isolated from each other. As can be observed in Figure 2(a), the completion times for 2 and 3 VMs per node follow the same shape as the curve corresponding to 1 VM per node, however they are smaller by a significant near-constant factor.

To explain this effect, notice the better overall CPU utilization in Figure 3(c) and Figure 3(e): from an average of 65% in the case of 1 VM per node, it has risen to 80% and 90% for 2 and 3 VMs respectively, leading to a shorter map phase. Thus, we



(a) Average CPU utilization for all nodes with 1 VM / node

(b) Average disk utilization for all nodes with 1 VM / node

(c) Average CPU utilization for all nodes with 2 VMs / node

(d) Average disk utilization for all nodes with 2 VMs / node

(e) Average CPU utilization for all nodes with 3 VMs / node

(f) Average disk utilization for all nodes with 3 VMs / node

Fig. 3. Aggregated statistics for overall CPU and disk utilization with a variable number of VMs per node and virtual disks per VM

conclude that the load balancing implemented in Hadoop is currently tuned towards horizontal scalability rather than vertical scalability. This tendency is also confirmed by a shorter reduce phase, for which the explanation is found in Figure 2(b): as the size of the Hadoop cluster grows, more reducers are used, despite an overall constant number of reducer slots being available in all configurations.

Nevertheless, when the number of VMs that share the same node increases, so does the virtualization overhead, limiting the potential to exploit horizontal scalability simply by adding more VMs per node. This trade-off can be observed from the completion times of 2 and 3 VMs per node, which are very close to each other (Figure 2(a)). Higher I/O pressure slightly tips the balance in favor of 2 VMs per node for few virtual disks per instance, while the opposite holds for higher number of virtual disks per instance.

4 Related work

Several state-of-art cloud middleware [5] (such as OpenStack [6]) offer dedicated storage solutions that aggregate block storage available on compute nodes to form distributed repositories. However, there is no specific feature or API to control how virtual disks are mapped to physical disks. Thus, we felt the need to contribute with our own experimental framework.

Extensive related work has been undertaken in the area of MapReduce workload characterization. Some works report low resource utilization [7] and suggest potential energy savings by consolidating workloads to fewer nodes. With respect to I/O, Ren et al. [8] conclude that improving data locality has little potential to improve I/O performance, which is also confirmed by our findings. They suggest in-memory storage, potentially in form of a DSM (distributed shared memory) as an alternative to disk storage. Other studies focus particularly on HDFS [9, 10]. Unlike our approach, the focus is on HDFS utilization (i.e. metadata, file access patterns create, read, write, delete, etc.) and does not involve intermediate data. Furthermore, instead of mixed I/O from multiple workloads, we analyze single workloads in isolation, in order to understand potential correlations.

Our own previous work [11] explores how to replace HDFS with a new storage layer based on BlobSeer [12], a versioning-based distributed storage system specifically designed for high throughput under concurrency. This previous work focuses on horizontal scalability and does not involve virtualization issues.

Several efforts have acknowledged the need to optimize MapReduce I/O at node level. Themis [13] implements the MapReduce paradigm using different design decisions than Hadoop. In particular, it introduces a centralized per-node disk scheduler that batches together records produced by different mappers in order to minimize the number of I/O operations. Ibrahim et al. [14] focus on improving I/O virtualization by means of smart coupling of the disk schedulers used at host and guest level that adapts to the workload. Unlike our case, the focus in these efforts is on how to optimize I/O for single disks rather than how to efficiently aggregate the bandwidth of multiple disks.

To our best knowledge, we are the fist to explore the problem of efficient virtualization of multiple local disks for data-intensive MapReduce workloads.

5 Conclusions

With increasing data sizes, big data analytics becomes increasingly challenging. In a quest to keep up with scalability, paradigms such as MapReduce were specifically designed to decouple tasks and improve horizontal scalability of big data systems. However, with horizontal scalability increasingly difficult to achieve, vertical scalability has recently gained increasing attention. Although adding more cores per node is a common occurrence, adding more disks per node to improve local I/O capabilities is not. Since big data applications are I/O intensive, doing so is highly desirable in order to remain scalable. Furthermore, given the tendency to virtualize datacenters in order to improve utilization and/or sell cloud computing services, the problem of how to efficiently virtualize multiple local disks for big data analytics is becoming crucial.

In this work we addressed the above problem. Given that this direction is still emerging, current cloud computing middleware is lacking features to guarantee efficient placement of virtual disks. Thus, our first contribution was to build an experimental framework that is able provide control over virtual disk placement, either spreading them over multiple physical disks in order to improve aggregated I/O or consolidating them on few physical disks.

Based on this experimental framework, we analyzed a data-intensive Hadoop workload in various virtualization settings. First of all, we found that Hadoop workloads can significantly benefit from striping to multiple virtual disks, with reductions in overall completion time of up to 70% when aggregating the I/O of 8 disks compared to a single disk.

However, our findings also show that Hadoop is better designed for horizontal rather than vertical scalability: its striping ability makes increasingly less use of the overall aggregated I/O bandwidth with increasing number of virtual disks. Furthermore, its load balancing ability increases with increasing number of VMs, despite sharing the same physical resources. This presents an interesting trade-off: on one side, increasing the number of VMs per node and/or the number of virtual disks per VM increases the virtualization overhead, but on the other hand it enables Hadoop to leverage the infrastructure better.

Thanks to these findings, we propose two interesting directions as future work. The first direction deals with how to improve Hadoop itself in order to enable it to leverage multiple virtual disks efficiently, both at the level of intermediate data and persistent data that needs to be saved in HDFS. In this context, the relationship to virtualization would be interesting to explore: would Hadoop benefit from being virtualization-aware? If so, what optimizations would be possible? Furthermore, does this go both ways (in other words, are there any hints it can give to the virtualization layer so that the latter can perform specific optimizations)?

Second, since Hadoop striping does not fully leverage the aggregated local I/O bandwidth to its full potential, another interesting direction to explore is whether presenting a single virtual disk to the VM and doing striping transparently in the background at hypervisor level can make better use of the aggregated bandwidth. In this context, we propose the concept of *bandwidth-elastic virtual disk*: a virtual disk that stripes to more physical disks under high I/O pressure and consolidates to less physical disks when the I/O pressure is lower, thus improving I/O resource utilization and enabling more efficient multi-tenancy and lower operational costs (e.g. saving energy by powering off disks).

References

1. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Communications of the ACM **51**(1) (2008) 107–113
2. White, T.: Hadoop: The Definitive Guide. O'Reilly Media, Inc. (2009)
3. Gagné, M.: Cooking with Linux—still searching for the ultimate Linux distro? Linux J. **2007**(161) (2007) 9
4. Shvachko, K., Huang, H., Radia, S., Chansler, R.: The Hadoop distributed file system. In: MSST '10: The 26th Symposium on Massive Storage Systems and Technologies. (2010)
5. Zhang, Z., Wu, C., Cheung, D.W.: A survey on cloud interoperability: taxonomies, standards, and practice. SIGMETRICS Perform. Eval. Rev. **40**(4) (April 2013) 13–22
6. Baset, S.A.: Open source cloud technologies. In: SoCC '12: Proceedings of the 3rd ACM Symposium on Cloud Computing, New York, NY, USA, ACM (2012) 28:1–28:2
7. Kavulya, S., Tan, J., Gandhi, R., Narasimhan, P.: An analysis of traces from a production mapreduce cluster. In: CCGRID '10: Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, IEEE Computer Society (2010) 94–103
8. Ren, Z., Xu, X., Wan, J., Shi, W., Zhou, M.: Workload characterization on a production hadoop cluster: A case study on taobao. In: IISWC '12: Proceedings of the 2012 IEEE International Symposium on Workload Characterization, San Diego, USA, IEEE Computer Society (2012) 3–13
9. Abad, C.L., Roberts, N., Lu, Y., Campbell, R.H.: A storage-centric analysis of mapreduce workloads: File popularity, temporal locality and arrival patterns. In: IISWC '12 Proceedings of the 2012 IEEE International Symposium on Workload Characterization, San Diego, USA (2012) 100–109
10. Abad, C.L., Luu, H., Roberts, N., Lee, K., Lu, Y., Campbell, R.H.: Metadata traces and workload models for evaluating big storage systems. In: UCC '12: Proceedings of the 5hth International Conference on Utility and Cloud Computing, Chicago, USA, IEEE Computer Society (2012) 125–132
11. Nicolae, B., Moise, D., Antoniu, G., Bougé, L., Dorier, M.: Blobseer: Bringing high throughput under heavy concurrency to hadoop map/reduce applications. In: IPDPS '10: Proc. 24th International Parallel and Distributed Processing Symposium, Atlanta, USA (2010) 1–12
12. Nicolae, B., Antoniu, G., Bougé, L., Moise, D., Carpen-Amarie, A.: Blobseer: Next-generation data management for large scale infrastructures. J. Parallel Distrib. Comput. **71** (2011) 169–184
13. Rasmussen, A., Lam, V.T., Conley, M., Porter, G., Kapoor, R., Vahdat, A.: Themis: an i/o-efficient mapreduce. In: SoCC '12: Proceedings of the Third ACM Symposium on Cloud Computing, San Jose, USA, ACM (2012) 13:1–13:14
14. Ibrahim, S., Jin, H., Lu, L., He, B., Wu, S.: Adaptive disk i/o scheduling for mapreduce in virtualized environment. In: ICPP '11: The 2011 International Conference on Parallel Processing, Taipei, Taiwan (2011) 335–344