Abstract—Cloud computing has increasingly been used as a platform for running large business and data processing applications. Although clouds have become extremely popular, when it comes to data processing, their use incurs high costs. Conversely, Desktop Grids, have been used in a wide range of projects, and are able to take advantage of the large number of resources provided by volunteers, free of charge. Merging cloud computing and desktop grids into a hybrid infrastructure can provide a feasible low-cost solution for big data analysis. Although frameworks like MapReduce and desktop grids into a hybrid infrastructure raise some challenges due to their large resource heterogeneity and high churn rate. This study introduces BIGhybrid, a toolkit that is used to simulate MapReduce in hybrid environments. Its main goal is to provide a framework to address the issues of Hybrid MapReduce. In this paper, we describe the framework which simulates the assembly of two existing middlewares: BitDew-MapReduce for Desktop Grids and Hadoop-BlobSeer for Cloud Computing. The experimental results that are included in this work demonstrate the feasibility of our approach.

I. INTRODUCTION

Mankind is producing an ever increasing amount of data. According to IDC1, by 2020 there will be around 40 Zettabytes (40,000,000 Petabytes) of data that will require processing of some sort. This data volume requires processing capabilities beyond those that current IT infrastructure can provide.

MapReduce (MR) [1], a programming framework proposed by Google and currently adopted by many large companies, has been employed as a successful means of data processing and analysis for some time. Hadoop, the most popular open-source implementation of MR [2], abstracts task parallelism from programmers who only need to implement applications such as Map and Reduce functions. Cloud computing has increasingly been used as a platform for business applications and data processing [3]. Cloud providers offer Virtual Machines (VMs), storage, communication, and queue services to customers for which they pay an hourly fee. These resources can be used for deploying Hadoop clusters for data processing and analysis.

In the case of data processing, the computing power offered by other types of infrastructure is also of interest. Desktop Grids (DG) [4], for instance, have a large number of users around the world who donate idle computing power to multiple projects. DG have been applied in several domains such as biomedicine, weather forecasting, and natural disaster prediction. Merging DG and Cloud Computing (Cloud) into Hybrid Infrastructures could provide a more affordable mean of data processing. However, although MR has been designed to exploit commodity hardware, its use in a hybrid infrastructure is a complex task owing to large resource heterogeneity and a high churn rate that is normal for these environments. Hybrid infrastructures are environments with large-scale distributed resources, that are spread geographically in heterogeneous platforms such as Cloud, grids and DG. These resources are available for users in the form of storage and computational capacity, e.g. social networking, Internet service providers and content delivery networks (CDNs) in CDN analytic applications [5].

The adaptation of an existing MR framework or the development of new software raises research issues related to a priori data splitting and distribution, strategies to avoid communication between the infrastructures, tasks and data co-scheduling, fault and sabotage tolerance, and data privacy, among several other issues. Moreover, using real testbeds to evaluate the use of MR in a hybrid infrastructure is quite difficult due to the lack of reproducibility in the experimental conditions for DG and the complexity of fine-tuning Cloud software stacks.

This study introduces BIGhybrid, a toolkit for simulating MR in hybrid environments, with a focus on Cloud and DG. We analyze the characteristics of the hybrid MR runtime environment and design the simulator accordingly. The simulator is based on SimGrid [6] and leverage solutions proposed within the scope of the MapReduce ARN project [7].

It is able to simulate two middleware for two distinct infrastructures: BitDew-MR [8], [9] for Desktop Grid Computing and Hadoop-Blobseer [10] for Cloud computing. BIGhybrid has several desirable features: it is built on top of MRSG, a validated Hadoop simulator [11], and MRA++, a simulator for heterogeneous environments [12]; it has a trace toolkit that can enable analyze, monitor and graphically plot the task executions; it is a trace-based simulator that is able to process real available traces in the infrastructure [13]; and its modular design allows for further extension. BIGhybrid can be used for evaluating scheduling strategies for MR applications in hybrid infrastructures. We believe that such a tool is of great value to researchers and practitioners working on big data applications and scheduling.

The rest of this work is structured as follows. Section II provides an overview on the MR framework and other systems used. In Section III, we describe the main characteristics of the hybrid infrastructures. Section IV introduces BIGhybrid and Section V describes the evaluation methodology. The conclusion and suggestions for future work are summarized in Section VI.

II. BACKGROUND

This section describes MR and introduces the implementations of both Hadoop-BlobSeer and BitDew-MapReduce, that will be used with the BIGhybrid simulator.

A. MapReduce

MR is a programming framework that abstracts the complexity of parallel applications by partitioning and scattering data sets across hundreds or thousands of machines, and by bringing computation and data closer [2]. Figure 1, adapted from [2], shows the MR data flow. The Map and Reduce phases are handled by the programmer, whereas the Shuffle phase is created while the job is being carried out. The input data is split into smaller pieces called chunks, that normally have a size of 64 MB. The data is serialized and distributed across machines that compose the Distributed File System (DFS).

When running an application, the master assigns tasks to workers that then run each processing stage. The machine that receives a Map task, handles a Map function and emits key/value pairs as intermediate results that are temporarily stored in the workers’ disks. The execution model creates a computational barrier, which allows tasks to be synchronized between the producers and consumers. A Reduce task does not start its processing until all the Map tasks have been completed. A hash function is applied with the intermediate data to determine which key partitions will compose a Reduce task. Each key partition is transferred to a single machine in prefetching mode, during the Shuffle phase, to execute the next phase. After a Reduce function has been applied to the data, a new resulting key/value pair is issued. Following this, the results are stored in the distributed file system and made available to the users.

If a machine is characterized as a straggler after the first task distribution, it will not be assigned new tasks to the free slots.

B. Hadoop-BlobSeer

BlobSeer is a DFS that manages a huge amount of data in a flat sequence of bytes called BLOBs (Binary Large Objects). The data structure format allows a fine-grained access control. BlobSeer emits files from version control which enables the incremental updating of database files, and a high throughput with concurrent reading, writing and updating from data. The data chunks have a fixed size of 64 MB so that they can maintain compatibility with the Hadoop Distributed File System (HDFS), that has been replaced by the BlobSeer on Hadoop.

An API implements the calls of Hadoop for the BlobSeer File System (BSFS). The namespace manager is centralized, and it keeps the namespace of BSFS when mapping files for BLOBs. However, this data structure is completely transparent for the Hadoop system and its MR users. The classic execution of MR on Hadoop was not changed and explores data locality similar to HDFS. The BSFS provides a data replication by means of a flat structure. The fault-tolerance mechanism is a simple data replication across the machines. However, it does not explore replication across racks as in HDFS.

C. BitDew-MapReduce

BitDew [8] is a middleware that exploits protocols like P2P, http, BitTorrent and ftp, and selects the best protocol in accordance with the sizes of the data types. BitDew architecture is decentralized and has independent services for Data Scheduler, Data Catalog, Data Repository and Data Transfer. These services are accessed via three API: Active Data to control the behavior of the data system, such as replication, fault-tolerance, data placement, lifetime, protocols and event-driven programming facilities; Transfer Manager to manage the concurrent file transfer completions and the concurrency level of transfers; BitDew to provide functions that create slots and carry out data management.

The Data Catalog maintains a centralized and updated meta-data list for the whole system in a Distributed Hash Table. The model includes both stable and volatile storage. Stable storage is provided by stable machines or Cloud Storage like Dropbox and Google Drive, and volatile storage consists of local disks of volatile nodes. The MR implementation [14] is an API that controls the master and worker daemon programs. This MR API can handle the Map and Reduce functions through BitDew services. The data locality in Hadoop MR was implemented like a data attribute to support the separation of the input data distribution from the execution process.

Result checking is controlled through a majority voting mechanism [9]. In the Hadoop implementation when the network experiences unavailability, a heartbeat mechanism signals to the master that the host is dead. However, in BitDew the network can be temporarily offline without experiencing any failure. The fault tolerance system needs a synchronization schema, as pointed out by [15] where transient and permanent failures can be handled. A barrier-free computation is implemented in BitDew to mitigate the host churn behavior [14].
In BitDew (unlike the case of the classical MR implementation), the computation of \textit{Reduce} nodes starts as soon as the intermediate results are available.

III. HYBRID INFRASTRUCTURE

Table I, summarizes the main architectural features of Cloud-BlobSeer, BitDew-MapReduce and the Hybrid MR environment. The hybrid infrastructure enables the use of highly heterogeneous machines, and stable and volatile storage to avoid data loss. The extent to which a set of data-distribution strategies is applicable to a given scenario depends on how much bandwidth is available. Two independent DFS implementations are required to handle data distribution in two scenarios, namely low-bandwidth and high-bandwidth. A hybrid infrastructure uses an Orchestrator to manage the results and data input from users. This can be decentralized to improve data distribution in the network. In the special case of Cloud and DG, fault tolerance mechanisms have different policies to detect faults. A more specialized system is applied on DG due to its node volatility.

Figure 2 illustrates the solution proposed to model a hybrid system and introduces \textit{Global Dispatcher} and \textit{Global Aggregator} to be used with the BIGhybrid simulator. The \textit{Global Dispatcher} located outside the DG and Cloud has middleware functions for handling task assignments and input data from users. It is a centralized data storage system that manages policies for split data and distribution, in accordance with the needs of each system. The working principle is similar to the publish/subscribe service where the system obtains data and publishes the computing results. This approach is simple, but risks causing a network bottleneck. The BIGhybrid simulator will enable several strategies to be investigated to determine the best data distribution and resource allocation of MR applications in hybrid infrastructures.

\textit{Global Aggregator} receives all the key/values of \textit{Reduce}, and keys with the same index in each system are joined to the last \textit{Reduce} function to obtain a consistent results. The BIGhybrid simulator will help to choose the best strategies to achieve this goal.

![Hybrid Infrastructure](image)

However, the iterative MR computations, that are needed by the \textit{Global Aggregator}, are not supported by an original MR model. It is not an easy task to combine all \textit{Reduces} from heterogeneous platforms, although it is possible to carry out a new stage for MR [16]. One possible approach is to use the \textit{MapIterativeReduce} [17] which creates an Aggregator to collect all the outputs of the \textit{Reduce} tasks and combines them into a single result. At the end of each iteration, the reducer checks to find out whether is the last or not. However, according to [18], this schema might be ineffective for large workloads. BIGhybrid enables the study of variations and patterns for the implementation of the aggregation module.

IV. BIGHYBRID SIMULATOR

The idea behind the BIGhybrid simulator is to optimize MR applications to provide a cloud service with the available resources of a DG system. BIGhybrid is modular and is built on top of Simgrid [6], which is a simulation-based framework for evaluating cluster, clouds, grid and P2P (peer-to-peer) algorithms and heuristics. Unlike other simulators, BIGhybrid has two independent systems. This enables it, to use different configurations for DFS, schedulers, input/output data size, number of workers, homogeneous and heterogeneous environments, combine two different platforms, and make use of parallel simulation. BIGhybrid generates traces from each system to allow an individual or collective analysis to be conducted within the same time frame.

BIGhybrid is built on two components described in previous work: MRSG (MapReduce over SimGrid) that simulates Cloud-BlobSeer with Hadoop; and MRA++ (MapReduce Adapted Algorithms to Heterogeneous Environments) that simulates BitDew-MapReduce. Figure 3 illustrates the architecture of BIGhybrid, which comprises four main components: input data management (Global Dispatcher), Cloud-BlobSeer module, BitDew-MapReduce module and an integration module for results (Global Aggregator). More details about the MRSG simulator and MRA++ can be found in [11] and [19].

![BIGhybrid Simulator Architecture](image)

A user can specify an input function for each system, as well as for individual \textit{Map} and \textit{Reduce} functions. In the next release version, it will be possible to build platforms for real infrastructures using Failure Trace Archive [13]. This means that the BIGhybrid can provide 256 settings of configurations in the same simulator. In addition, it is possible to make adjustments to several kinds of strategies and configurations in both Cloud-BlobSeer with Hadoop and BitDew-MapReduce, to find a load balance without data loss and with suitable strategies to achieve an efficient data partition between the two environments.
A. Cloud-BlobSeer Simulation Module

The Cloud-BlobSeer simulation module reproduces the behavior of the MR platform, and invokes SimGrid operations whenever a network transfer or processing task must be performed, without modifying the SimGrid source code. This simulation follows the Hadoop implementation, with a heartbeat mechanism to control the task execution. This architecture comprises the following: API of input users code, DFS, MapReduce functions, master (Jobtracker) and slaves (Tasktracker).

The DFS is implemented as a matrix that maps chunks to nodes. The master node knows where each chunk is placed, exactly as occurs in the real implementation. Moreover, each chunk can be linked to more than one node, which allows chunk replica simulation. SimGrid is responsible for the simulation of all the network communication and task processing in our implementation, whereas Cloud-BlobSeer simulation only implements the node distribution in one rack. The next version of BIGhybrid will use the storage API simulation of SimGrid, on Disk Emulation Module to simulate the storage behavior. As at the time writing, disk simulation is specified as an I/O cost in the configuration file in the User API. The virtual machine behavior is not simulated in the current version of BIGhybrid, but is abstracted as an additional task cost. The virtualization support module will later be integrated with the aid of SimGrid virtualization support, as described in [20].

B. BitDew-MapReduce Simulation Module

BitDew [8] is a middleware for large scale data management in hybrid distributed computing infrastructures. Its runtime environment supports the use of multiple file transfer protocols: either client/server (http, ftp, scp), P2P (bittorrent), Grid (through the SAGA API) or Cloud (Amazon S3, Dropbox). A set of BitDew services is able to handle a number of high level data management issues: data index and meta-data catalog, fault tolerance, reliable file transfer, data scheduling, replication, data life-cycle, collective communication, data collection management, event-based programing model and more.

The implementation of MapReduce over BitDew is mainly targeted at Desktop Grid systems [14], and employs mechanisms to alleviate host churn, and the problem of unavailability of direct communication between the hosts and lack of host trust. The implementation relies on master and worker daemon programs. This MR API can handle the Map and Reduce functions through BitDew services. The data locality of Hadoop MR was implemented as a data attribute to support the separation of the input data distribution from the execution process.

A function controls the result system checking through the majority voting mechanism, as in Moca [9]. With the Hadoop implementation, when the network experiences unavailability, a heartbeat interval signals to the master that the host is dead. However, in BitDew the network can be temporarily offline without undergoing any failure. The FT needs a synchronization scheme, as pointed out by [15], where transient and permanent failures can be handled. A barrier-free computation is implemented in the BitDew simulation (as can be seen in Section V).

C. Additional BIGhybrid Modules

In BIGhybrid, the Global Dispatcher is either manual or automatic. In the manual version, the user defines a function for data distributions and a job configuration for each system for both Cloud-BlobSeer and BitDew-MapReduce, such as, input data, data size, chunk size and so on. In automatic release, an Orchestrator deals with user queries and distributes tasks to the systems. A Global Storage is used to maintain user-related data, so that the Orchestrator can initialize a new task, if necessary.

The results of the Global Aggregator module are implemented as a single Reduce task after the last current Reduce task has been completed. The processing results are tracked and saved in a file for future analysis. A toolkit for the execution system analysis was implemented to assist in creating both homogeneous and heterogeneous platforms, and make execution traces based on visualization traces supported by SimGrid. This toolkit enables users to analyze the whole execution system and change the strategies as needed. The traces can be individual, as well for all the simulations in system.

V. Evaluation

This section describes the environment setup and results of the evaluation to demonstrate the features and scalability of the simulator.

A. The Environment Setup

Two environments have been considered. The first, in a small-scale server is only a proof of concept to determine the execution features from Table I. The second considers a cluster of 2,000 nodes and evaluates the simulator’s ability to replicate results obtained in previous work. The proof-of-concept first simulated a homogeneous 5-node cluster with

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Cloud-BlobSeer with Hadoop</th>
<th>BitDew-MapReduce</th>
<th>Hybrid-MapReduce</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heterogeneity</td>
<td>Moderate</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Network</td>
<td>High Bandwidth</td>
<td>Low Bandwidth, distributed cache</td>
<td>Hybrid Bandwidth</td>
</tr>
<tr>
<td>Local Storage</td>
<td>Distributed</td>
<td>Remote (Cloud Storage) + local</td>
<td>Local and distributed + Remote Cloud Storage</td>
</tr>
<tr>
<td>Management</td>
<td>Master/Slave</td>
<td>Master/Slave</td>
<td>Hierarchical Orchestrator</td>
</tr>
<tr>
<td>Application Profile</td>
<td>Any</td>
<td>Low Communication in Shuffle phase</td>
<td>Optimized for all size archives</td>
</tr>
<tr>
<td>File System API</td>
<td>Posix</td>
<td>Tuple Space model</td>
<td>Hybrid (Posix + Tuple Space)</td>
</tr>
<tr>
<td>Hosts</td>
<td>Stable</td>
<td>Stable and Volatile</td>
<td>Stable and Volatile</td>
</tr>
<tr>
<td>FT mechanism</td>
<td>Data and Task Replication</td>
<td>Data Replication and transient failure support</td>
<td>Data and Task Replication, and transient failure support</td>
</tr>
<tr>
<td>Computation</td>
<td>Hadoop Compatible</td>
<td>Barrier-free</td>
<td>Hybrid</td>
</tr>
</tbody>
</table>
2 cores each, (5.54 GFlops of processing capacity and 1 Gbps network). This cluster was used to process 2GB of data, 36 maps and 5 reduces. The second cluster contains 5 heterogeneous machines with 2 CPU cores each, which process capacity is drawn from a log-normal distribution according to [21] (from 4.76 GFlops to 6.89 GFlops, in a network of 10 Mbps). This cluster is used to process 1.1 GB of data, 36 maps and 30 reduces. A characterization of MR applications performed by Chen [22] was drawn on to define the large scale-setup. Chen examined MR traces from two production environments from Yahoo and Facebook.

The traces obtained from Yahoo come from a 2,000 node cluster and contain 30,000 jobs spanning a period of over 3 weeks. The cluster was used to run applications that require bash, interactive and semi-streaming computations. For the purposes of this work, we model the Yahoo cluster and consider the “aggregate, fast job” applications characterized by Chen. Table II shows the details of these applications, including the number of jobs, input data size, job duration, Map time and Reduce time. This job has 568 GB of input and 9,088 tasks with a runtime of 322.64 s from Map, and 703.32 s from Reduce.

B. Results and Analysis

The first experiment is a proof-of-concept introduced by Figures 4, 5 and 6. Figure 4 shows an execution of Cloud-BlobSeer in the homogeneous cluster. The Map tasks produce intermediary keys that are sent during the shuffle phase, and the Reduce tasks are initialized once the Map tasks have been completed.

The results of BitDew-MapReduce execution on a heterogeneous cluster are shown in Figure 5. The Reduce tasks start as soon as the machines have some data to process. It should be noted that, as the link is 10 Mbps, the data transfers take longer to complete during the shuffle phase. The execution time is similar that of the first cluster because the data chunk size of BitDew-MapReduce is 32 MB instead of 64 MB as in Cloud-BlobSeer.

Figure 6 shows the execution of BIGhybrid when the two clusters are taken into account. This chart is especially interesting as it demonstrates the task parallelism for the execution and the beginning of intermediate data transfers.

VI. CONCLUSION

The rapid increase in the amount of data currently being produced will stretch the current infrastructure to its limits. Merging Cloud and DG into a hybrid infrastructures can be a feasible low-cost alternative to simply using Cloud environments.

The BIGhybrid simulator enables the adoption of MR strategies to carried out in hybrid infrastructures. Preliminary experiments have demonstrated that the simulator has a good performance and provide evidence that the initial goals have been achieved. However, it is necessary to conduct further experiments to compare the configuration strategies with the results obtained from real systems. The next steps include
improving volatility, fault tolerance mechanisms and disk simulation. We also intend to implement virtualization support.

ACKNOWLEDGMENTS

This work was partly supported by CAPES Process BEX 14966/13-1. This work was supported by the Agence National de la Recherche under grant ANR-10-SEGI-001. The experiments discussed in this paper were conducted with the aid of the Grid5000 experimental testbed, under the INRIA ALADDIN development plan with support from CNRS, RENATER and a number of universities (see https://www.grid5000.fr).

REFERENCES


