BIGhybrid: A Simulator for MapReduce Applications in Hybrid Distributed Infrastructures Validated with the Grid5000 Experimental Platform

Julio C. S. Anjos, Gilles Fedak, Claudio Geyer

CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2015; **00**:1–26
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe

BIGhybrid: A Simulator for MapReduce Applications in Hybrid Distributed Infrastructures Validated with the Grid5000 Experimental Platform

Julio C. S. Anjos[1*] Gilles Fedak[2] and Claudio F. R. Geyer[3]

[1]*Federal University of Rio Grande do Sul, Brazil Institute of Informatics - PPGC - CNRS-INRIA/LIP - UCBL ENS Lyon, France, Email: jcsanjos@inf.ufrgs.br*
[2]*INRIA/LIP, Univ. Lyon, France, Email: Gilles.Fedak@inria.fr*
[3]*Federal University of Rio Grande do Sul, Brazil Institute of Informatics - PPGC, Email: geyer@inf.ufrgs.br*

SUMMARY

Cloud computing has increasingly been used as a platform for running large business and data processing applications. Conversely, Desktop Grids have been successfully employed in a wide range of projects, because they are able to take advantage of a large number of resources provided free of charge by volunteers. A *hybrid infrastructure* created from the combination of Cloud and Desktop Grids infrastructures can provide a low-cost and scalable solution for Big Data analysis. Although frameworks like MapReduce have been designed to exploit commodity hardware, their ability to take advantage of a hybrid infrastructure poses significant challenges due to their large resource heterogeneity and high churn rate. In this paper is proposed BIGhybrid, a simulator for two existing classes of MapReduce runtime environments: BitDew-MapReduce designed for Desktop Grids and BlobSeer-Hadoop designed for Cloud computing, where the goal is to carry out accurate simulations of MapReduce executions in a hybrid infrastructure composed of Cloud computing and Desktop Grid resources. This work describes the principles of the simulator and describes the validation of BigHybrid with the Grid5000 experimental platform. Owing to BigHybrid, developers can investigate and evaluate new algorithms to enable MapReduce to be executed in hybrid infrastructures. This includes topics such as resource allocation and data splitting.

Received . . .

KEY WORDS: Hybrid Infrastructure, MapReduce, Distributed Systems, Simulation

1. INTRODUCTION

Mankind is producing an ever increasing amount of data. According to IDC[†], 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] is a programming framework proposed by Google and currently adopted by many large companies, which has been employed as a successful solution of data processing and analysis. Hadoop [2], the most popular open-source implementation of MR, abstracts the management of task parallelism by 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),

*Correspondence to: Federal University of Rio Grande do Sul - Institute of Informatics - PPGC, Postal Code 15064, CEP: 91501-970 Porto Alegre - RS - Brazil Email: jcsanjos@inf.ufrgs.br
[†]IDC's Digital Universe Study, sponsored by EMC, December 2012.

storage, communication, and queue services to customers for which they pay per resource usage. These resources can be used for deploying Hadoop clusters for data processing and analysis.

In addition to Cloud computing, several other types of infrastructure are able to support data-intensive applications. Desktop Grids (DG) [4], for instance, have a large number of users around the world who donate idle computing power to multiple projects. DGs have been applied in several domains such as bio-medicine, weather forecasting, and natural disaster prediction. Merging DG with Cloud Computing (Cloud) into Hybrid Infrastructures could provide a more affordable mean of data processing. Nevertheless, although MR has been designed to exploit the capabilities of commodity hardware, its use in a hybrid infrastructure is a complex task because of the large resource heterogeneity and a high churn rate. This is usual for Desktop Grids but uncommon for Clouds. In addition, Hybrid infrastructures are environments which have geographically distributed resources in heterogeneous platforms such as Cloud, Grids and DG.

The adaptation of an existing MR framework or the development of new software for hybrid infrastructures raises a number of research questions: how to create efficient strategies for data splitting and distribution, how to keep communications between the infrastructures to a minimum, how to deal with failures, sabotage, and data privacy. Moreover, the use of real-world test beds to evaluate MR applications is almost impossible due to the lack of reproducibility in the experimental conditions for DG and the complexity of fine-tuning Cloud software stacks.

BIGhybrid is a toolkit for MR simulation in hybrid environments and was previously introduced in [5], with a focus on Cloud and DG. The simulator itself is based on the SimGrid framework [6]. The main purpose of this study is to demonstrate that the BIGhybrid simulator has features that allow it to carry out accurate simulation and that it is able to simulate the execution behavior of two types of middleware for two distinct infrastructures: BitDew-MR [7, 8] for Desktop Grid Computing and Hadoop-Blobseer [9] for Cloud computing. BIGhybrid has several desirable features: a) it is built on top of SimGrid with two different simulators - *MapReduce over SimGrid* (MRSG), a validated Hadoop simulator [10], and *MapReduce Adapted Algorithms to Heterogeneous Environments* (MRA++), a simulator used for heterogeneous environments [11]; b) it has a trace toolkit that can enable analysis, monitoring and graphically plot the task executions; c) it is a trace-based simulator that is able to process real-world resource availability traces to implement realistic fault-tolerance scenarios. These traces are available in a web site called Failure Trace Archive (FTA), which is a centralized public repository of resource availability traces for various parallel and distributed systems [12]; and d) its modular design allows for further extension.

BIGhybrid can be used for evaluating scheduling strategies for MR applications in hybrid infrastructures. We believe that this kind of tool is of great value to researchers and practitioners who are working on big data applications and scheduling. For validation purposes, the experiments are executed over Grid5000 [13]. Grid5000 is an experimental testbed, supported by INRIA, CNRS, RENATER and several universities in France. This study demonstrates that there is a similarity between the simulations of BIGhybrid and those of the MapReduce real experiments, which can serve to validate the simulator.

The rest of this work is structured as follows. Section 2 examines related work, and provides an overview of the MR framework together with the other systems used. This work analyzes more detailed characteristics of the hybrid MR environment in Section 3; Section 4 introduces the BIGhybrid and there is an examination of new features like a volatile module and communication model in Subsection 4.5, and a more detailed evaluation in Section 5 with new experiments, including a statistical evaluation in Subsection 5.5, to make comparisons with a real-world environment in Grid5000. The conclusion and suggestions for future work are summarized in Section 6.

2. BACKGROUND AND RELATED WORK

This section shows the main concepts about the MapReduce framework and other systems that have been used to compose Big Data ecosystem in hybrid infrastructures. The related work demonstrates

DOI: 10.1002/cpe

the effort of the scientific community to find a solution for data intensive computing in different platforms.

2.1. MapReduce

MR is a programming framework that abstracts the complexity of parallel applications by partitioning and scattering datasets across hundreds or thousands of machines, and by bringing computation and data closer together [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 task 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 and monitors the progress of the task. The machine that is assigned a *Map* task, executes 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 the 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 will compose a *Reduce* task. The group of selected keys forms a partition. Each partition is transferred to a single machine 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.

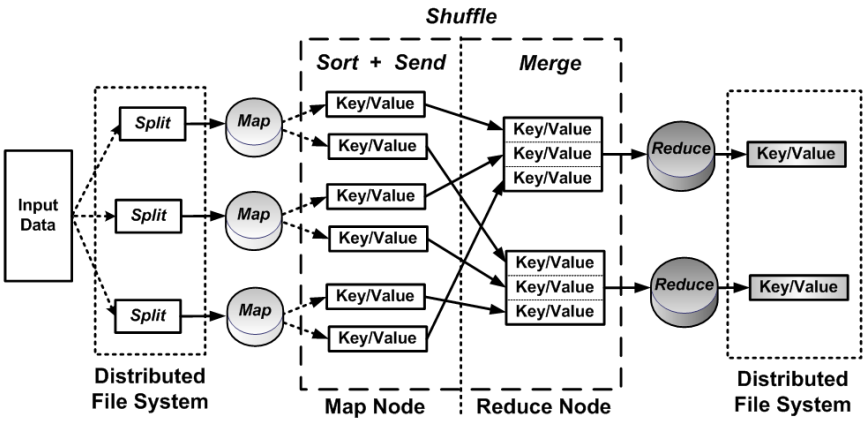


Figure 1. MapReduce data flowchart model

MR uses management systems for data replication and execution control. In addition, it has a management architecture based on the master/worker model, while a slave-to-slave data exchange requires a P2P model [2]. The worker is a node that can run a *Map* or *Reduce* functions in the *MapReduce* environment. A machine is characterized as a *straggler* when their task in progress is above execution average for the cluster. If a machine is characterized as a *straggler* after the first task distribution, it will not be assigned new tasks to their free slots.

2.2. BlobSeer-Hadoop

BlobSeer is a DFS that manages a huge amount of data in a flat sequence of bytes called BLOBs (**B**inary **L**arge **O**bject**s**). The data structure format allows a fine-grained access control. An unbalance workload is checked in the Hadoop file system (HDFS), when it receives new data from the incremental updates [9]. The existing storage file system has limited throughput under heavy access concurrency. HDFS does not support concurrent writes for the same file, and the data cannot be overwritten or appended to. BlobSeer maintains a most recent version of a particular file in a DHT (Distributed Hash Table) to favor efficient concurrent access to metadata, which enables

DOI: 10.1002/cpe

the incremental updating of database files, and a high throughput with concurrent reading, writing and updating from data [14]. This is the main reason for using another file system like BlobSeer.

This data structure is completely transparent for the Hadoop users. The fault-tolerance mechanism is a simple data replication across the machines, and enables the user to specify the replication level needed. The classical execution of MR on Hadoop was not changed and explores data locality similar to HDFS. In view of this, the BlobSeer was the best choice to implement the features of the incremental update quickly, without having to develop a new MapReduce framework for Cloud implementation. The incremental update is necessary for data management in a hybrid infrastructure.

2.3. *BitDew-MapReduce*

BitDew is a middleware that exploits protocols like P2P, http, BitTorrent and ftp. The architecture is decentralized and has independent services. These services control the behavior of the data system, such as replication, fault-tolerance, data placement, incremental update, lifetime, protocols and event-driven programming facilities.

The *Data Catalog* maintains a centralized and updated meta-data list for the whole system. 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 is an API that controls the master and worker daemon programs. This MR API can handle the *Map* and *Reduce* functions through BitDew services.

Result checking is controlled through a majority voting mechanism [8]. In the Hadoop implementation when the network experiences unavailability, a heartbeat mechanism signals to the master that the host is dead. Nevertheless, 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 to mitigate the host churn behavior [16]. The computation of *Reduce* nodes starts as soon as the intermediate results are available.

These properties of BitDew-MapReduce described earlier, such as data placement, incremental update and fault-tolerance mechanism, are important to implement a hybrid infrastructure. In addition, the computing power offered by the DG infrastructure is also of value to provide new infrastructures, starting from the allocation of free resources.

2.4. *Related work*

Big Data applications have several implementations, nevertheless, dispersal data can be found in biological research studies, where the researchers need to investigate different databases, such as, in the protein structure analysis. These applications seek a genetic mapping that require a pre-existing reference genome to be employed for the read alignment of a gene [17]. The data processing is characterized by its ability to compare input data with different databases. This processing consists of several phases of search-merge-reduce, where the data are given an incremental update [18]. Another question to consider is that several biological databases are dispersed across different institutions like Gene Report [19], Ensembl [20] and others. The solutions proposed for the hybrid infrastructure consider this heterogeneous scenario and are based on the scope of the MapReduce ANR project[‡] [14], in the context of biochemical research to produce medicines.

Some researchers [21, 22, 23] have put forward Hadoop implementations based on a geo-distributed dataset in multiple data centers. The authors state that, for instance, it is possible to have multiple execution paths for carrying out a MapReduce job in this scenario, and the performance can carry out a great deal. Nevertheless, a popular MapReduce open source, like Hadoop, does not support this feature naturally, and the major Cloud Service Providers (CSPs) do not usually provide a bandwidth guarantee [24].

[‡]National Research Agency (ANR), ARPEGE 2010 call. Project number: ANR-10-SEGI-001

DOI: 10.1002/cpe

These problems can be overcome by means of a hybrid infrastructure, if there is a file system that supports the incremental updates and highly concurrent data sharing, such as BlobSeer. The solution involves integrating BlobSeer into a distributed file system on Hadoop by making use of Cloud environment. Otherwise, desktop grids are a large-scale infrastructure with specific characteristics in terms of volatility, reliability, connectivity, security and storage space. Both architectures are suitable for large-scale parallel processing. Finally, more complex combinations can be envisaged of platforms resulting from the use of multiple Clouds through an extension to a DG [14].

The BOINC [25], XtremWeb [26] and BitDew [7] systems are successful implementations of DG environments. Nevertheless, BOINC and XtremWeb have a centralized infrastructure for scheduling and management; in contrast, BitDew is an evolution of a distributed infrastructure designed for data management that supports well incremented updates and fault tolerance mechanisms. MapReduce-BitDew [16] is a MapReduce implementation adapted to a volatile environment, that has already been combined with Cloud like a hybrid infrastructure [27] to improve performance and reduce costs through the bag-of-tasks application.

GroudSim, a Grid and Cloud simulation toolkit for scientific applications, was introduced by [28]. This simulator is based on a scalable simulation-independent discrete-event. This simulator, which is used for scientific applications, was an attempt to simulate two complex environments, like Grid and Cloud. *GroudSim* provides support for traces used for capturing both hosts and event traces.

Stochastic distributions make it possible to run deterministic and non-deterministic simulations. A failure rate model follows a stochastic distribution of failure properties like the size of the failure, the duration of the failure and the *Mean Time To Failure* (MTTF) for jobs and file transfers. Nevertheless, the simulation architecture is composed of a single thread. The infrastructures are only very simple synthetic entities and, for this reason, it is difficult to capture discrete executions. Unlike GroudSim, Bighybrid makes possible complex simulations including volatile environments.

CloudSim [29] is an extension of GridSim [30] for Cloud simulation. The simulator supports modeling of large-scale Cloud computing environments, including data centers, on a single physical computing node. This means that Clouds, service brokers, provisioning, and allocation policies can be modeled. The main features enable the creation and management of multiple, independent, and virtualized services in a data center.

The simulation is based on Java and has dedicated management interface for VMs, memory, storage and bandwidth. A host can support multiple sets of VMs to simulate applications based on Software-as-a-Service providers. The authors assume that provisioned virtual machines are predictable and stable in their performance. There is an I/O contention that has been verified in read/write storage devices and has an impact on the performance [31]. Although CloudSim can simulate Federated Cloud with a *Cloud Coordinator*, the simulator is not compatible with data-intensive applications [29] like in the model of the MapReduce framework.

CSPs with different geographical locations over the Internet have to coordinate their load distribution across data centers. The study of [32] introduces the InterCloud simulator as a possible architecture that extends CloudSim to the Cloud Federation infrastructures. The main problem is that the service providers expect the users to choose the service that is nearest to their physical location. Otherwise, the clients have difficulty in determining the best location for hosting their services in advance, since a CSP may not know the origin of the consumers of their services. As a result, the CSP may not be able to provide the quality of service contracted in the local if the customers originate from multiple geographical locations.

InterCloud software architecture has a coordinator and brokers to locate resources for clients. The functions of the coordinator are scheduling, resource allocation, dynamic monitoring and application composition. Nevertheless, this architecture does not take account of security mechanisms or a minimal SLA [33]. In addition, the broker is not prepared for data-intensive management.

Kohne introduces a simulation of Cloud Federation [33] to reduce the complexity of the experiments called FederatedCloudSim. CSPs can use resources of other CSPs with the aim of improving resource optimization while respecting SLAs. The migration services must be executed automatically and a Service Level Agreement - SLA - has to be negotiated in advance. The purpose

DOI: 10.1002/cpe

of this is to study standard interfaces to exchange services and establish an orchestration framework that creates and monitors distributed services based on SLAs. FederatedCloudSim is implemented with the presented CloudSim.

The scheduling process has several levels and invokes the brokers. It may be dedicated or employ a pass-through model. In the dedicated model, the tasks are executed by a broker locally, and in the pass-through model the tasks are passed to a remote CSP member of the Cloud Federation. A special case is a virtual CSP that is outside the federation and can accept jobs from customers. Otherwise, the services will be "best-effort"and are described as Service Level Objectives - (SLO) in the SLA. Again, this implementation is not designed for data-intensive management.

AweSim simulator is defined in [34] and based on a network simulation framework that involves a fine-grained simulation for workflow computation and data movement across multiple Clouds. The proposal attempts to overcome the problems of provisioning and allocating resources for multiple Cloud scientific workflows that require task placement and data movement between distributed multi-domain computing sites. The AweSim is a client/server architecture. The implementation uses workload traces from a production data analysis service and is thus similar to the BIGhybrid simulator that adopts its behavior from traces in a real-world volatile environment.

The data-intensive approach avoids unnecessary data movement in the Workflow simulator. A ratio is calculated for the most expensive computational task (E_c), as $E_c = T_{run}/S_{in}$, where T_{run} is the runtime and S_{in} is the input data size. A historical job determines the average E_c that defines the most data-intensive task. The scheduling considers the distance between the server and computing resources. The CSP may have a different data size to adjust its distribution and explore the design of large-scale storage, network architecture and distributed data. The authors assume the computing resources are homogeneous except for the network bandwidths for the data server. Otherwise, the environment is different from the hybrid infrastructures where the workloads and resources are heterogeneous.

DynamicCloudSim is an extension of the popular simulation CloudSim toolkit that is used in the study of [31]. The goal is to model the instability inherent in computational Clouds and similar distributed infrastructures. This instability is demonstrated in the study of [35], where considerable performance variations were found, that fell into two bands, depending on the selected processor type. The simulator allocates resources to the VMs in terms of compute units, similar to Amazon EC2. Furthermore, in contrast with CloudSim, DynamicCloudSim does not assign new VMs to the host with the most available resources, but to a random machine within the data center. The heterogeneity is simulated through this random choice and represents permanent variance in the performance of VMs caused by differences in hardware. The stragglers (nodes with poor performance) are simulated through coefficient parameters of performance.

The related work, summarized in Table I, shows that there is a long way to go to find solutions for Cloud and multiple Cloud environments. This illustrates the need to investigate new environments such as hybrid infrastructures. BIGhybrid simulator makes it possible to study complex environments like hybrid environments and capture more fine grained runtime. The hardware infrastructure is modeled like real-world machines and the node behavior can be determined from real-world traces. These traces originate from resource availability in volatile environments and are obtained from the FTA website. BIGhybrid approach allows us to analyze generic data-intensive applications with MapReduce through traces of real executions (as in [36] and as demonstrated in Section 5.6).

3. HYBRID INFRASTRUCTURE

The hybrid infrastructure uses an Orchestrator to manage the results and data input for users. This must be decentralized to improve data distribution in the network. In the special case of Cloud and DG, fault tolerance mechanisms adopt different policies to detect faults. A more specialized system is applied to DG due to its node volatility. It is important to define the main features of hybrid infrastructures.

DOI: 10.1002/cpe

Table I. BIGhybrid vs Related Work Simulators

Simulators	Simulated Features							
	Grid	Cloud	Federated Cloud	Hybrid	Big Data Support	Failure Support	Trace Support	SLA Support
GroudSim	Yes	Yes	No	No	No	Yes	Yes	No
CloudSim	No	Yes	Yes	No	No	No	No	No
InterCloud	No	Yes	Yes	No	No	No	No	No
FederatedCloudSim	No	Yes	Yes	No	No	No	No	Yes
AweSim	No	Yes	Yes	No	Yes	No	No	No
DynamicCloudSim	No	Yes	Yes	No	No	Yes	No	No
BIGhybrid	No	Yes	No	Yes	Yes	Yes	Yes	No

Table II summarizes the main architectural features of BlobSeer-Hadoop, BitDew-MapReduce and the Hybrid MR environment. The hybrid infrastructure enables the use of highly heterogeneous machines, with 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. The application profile is optimized for all file sizes in hybrid infrastructures, as the systems are independent and thus the different data size can be handled at the same time. The bandwidth and computational capacity of machines influence the initial assumptions for defining a straggler machine and because of this, each system must be treated in a different way.

Figure 2 illustrates the solution proposed to model a hybrid system and introduces *Global Dispatcher* and *Global Aggregator* to be used with the BIGhybrid simulator. The *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.

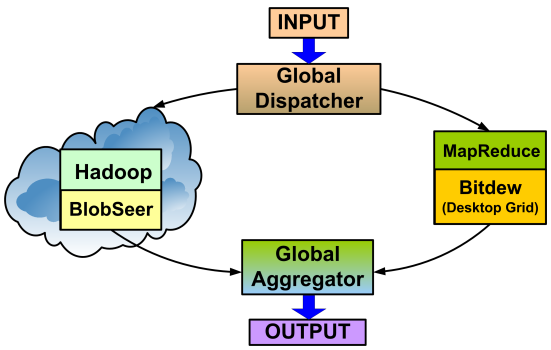


Figure 2. Hybrid infrastructure

Global Aggregator receives all the *key/values* of *Reduce*, and keys with the same index in each system are joined to the last *Reduce* function to obtain a consistent result. Nevertheless, the iterative MR computations, that are consumed by the *Global Aggregator*, are not supported by an original MR model. It is not an easy task to combine all *Reduces* from heterogeneous platforms, although it is possible to carry out a new stage for MR [37]. One possible approach is to use the *MapIterativeReduce* [38] which creates an *Aggregator* to collect all the outputs of the *Reduce* tasks and combines them into a single result. At the end of each iteration, the reducer checks to find out whether or not it is the last. Otherwise, according to [39], this schema might be ineffective for large

DOI: 10.1002/cpe

Table II. Comparison between MR systems

Characteristics	BlobSeer-Hadoop with Hadoop	BitDew-MapReduce	Hybrid-MapReduce
Heterogeneity	Moderate	High	High
Network	High Bandwidth	Low Bandwidth with distributed cache	Hybrid Bandwidth
Architecture	Decentralized	Decentralized	Decentralized
Storage	Distributed	Remote (Cloud Storage) + local	Distributed, Remote Cloud Storage and local
Management	Master/Slave	Master/Slave	Hierarchical Orchestrator
Metadata	Distributed by DHT	Centralized on Data Catalog	Distributed DHT/Data Catalog
App profile	Any	Low Communication in *Shuffle* phase	Optimized for all file sizes
File system API	Posix	Tuple Space model	Hybrid (Posix + Tuple Sace)
Data locality	Yes (One Rack only)	Yes (Affinity by node)	Implemented according to each platform
Chunk size	Fixed - 64MB	Fixed - 32MB	According each platform
Host model	Stable	Stable and Volatile	Stable and Volatile
FT mechanism	Data and Task Replication	Data Replication and transient failure support	Data and Task Replication, and transient failure support
Load balance	Strong Dynamic	Soft	Soft
Computation	Hadoop Compatilble	Barrier-free	Hybrid
Semantic of data concurrency	Multiple write, version update	Single write	Single write
Straggler management	Average execution task	Machine computational capacity	Hybrid
Storage Elasticity	High	High	High
MapReduce Semantic	More Compliance (Limited)	Restrict	Restrict

workloads. BIGhybrid enables the study of variations and patterns for the implementation of the aggregation module.

4. BIGHYBRID SIMULATOR

4.1. Introduction

The idea behind the BIGhybrid simulator is to optimize hybrid infrastructure environments such as Cloud services with the available resources of a DG system. BIGhybrid is modular and built on top of Simgrid framework [6]. Simgrid is a simulation-based framework for evaluating clusters, Clouds, grid and P2P (peer-to-peer) algorithms and heuristics. SimGrid is responsible for the simulation of all the network communication and task processing in our implementation. 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, as well as combining two different platforms, and making use of parallel simulation. In view of this, it is possible to setup several types of network and architecture platforms with simple modifications in the Bighybrid simulator, which can lead to a more generic hybrid infrastructure.

The BIGhydrid simulator generates traces from each system to allow an individual or collective analysis to be conducted within the same time frame. The simulator enable several strategies to be investigated to determine the best data distribution and resource allocation of MR applications in

DOI: 10.1002/cpe

hybrid infrastructures, to address the bottleneck issues. The BIGhybrid simulator will help to choose the best strategies to achieve this goal.

BIGhybrid is built on two components described in previous work: MRSG that simulates BlobSeer-Hadoop with Hadoop; and MRA++ that simulates BitDew-MapReduce. Figure 3 illustrates the architecture of BIGhybrid, which comprises four main components: input data management (Global Dispather), the BlobSeer-Hadoop module, the BitDew-MapReduce module and an integration module for results (Global Aggregator). SimGrid simulate the platform, network and CPU computation on nodes. The communication between BIGhybrid and SimGrid is achieved through the use of MSG, one of the many application programming interfaces provided by SimGrid.

MapReduce has three main phases: 1) The *Map phase* reads the data from the distributed file system and calls the user map function to emit (key, value) pairs as intermediate results. 2) In the *Shuffle phase*, the map nodes sort their output keys in partitions, that are then pulled by the *Reduce* nodes. Therefore, each reduce task will process the keys that belong to a specific partition. When all data transfers are done, the *Reduce* nodes execute a sort to merge the data pulled from the map nodes. 3) Finally, the *Reduce phase* calls the user's reduce function and writes the output back into the DFS. These phases are simulated in the BlobSeer-Hadoop and BitDew-MapReduce simulations. More specific details about the MRSG simulator and MRA++ can be found in [10] and [11].

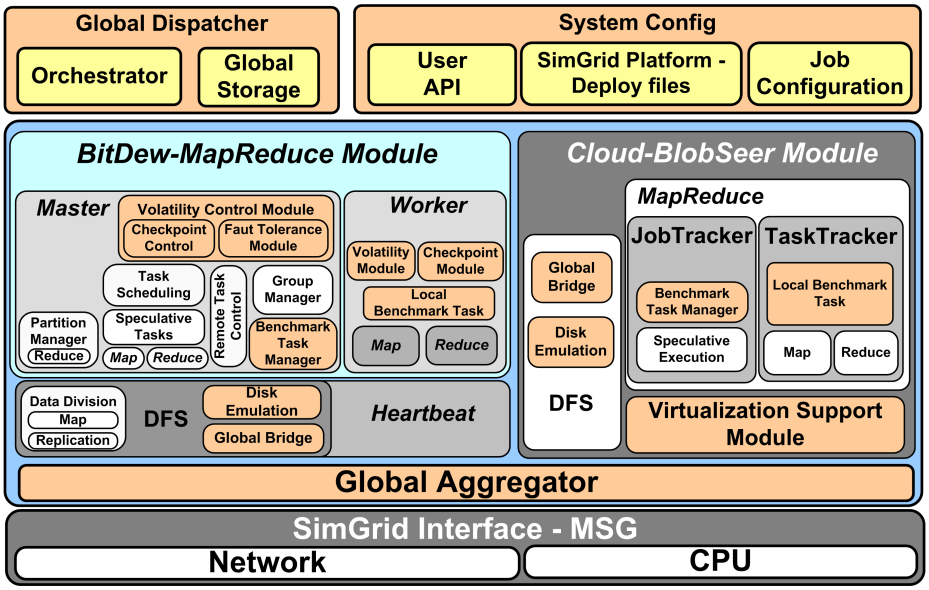


Figure 3. Architecture of the BIGhybrid simulator

A user can specify an input function for each system, as well as for individual *Map* and *Reduce* functions. With BIGhybrid it is possible to build platforms for real infrastructures through the platform description of configurations and real environments using the FTA. This means that the BIGhybrid simulator can provide up to 256 settings of configurations in the same simulator, *i.e.*, 2^n setups with 8 different modules. In addition, it is possible to make adjustments to several kinds of strategies and configurations in both BlobSeer-Hadoop with Hadoop and BitDew-MapReduce, to find the best load balance without data loss and with suitable strategies to achieve an efficient data partition between the two environments.

4.2. BlobSeer-Hadoop Simulation Module

The BlobSeer-Hadoop module reproduces the behavior of the MR framework, and invokes SimGrid operations whenever a network transfer or processing task must be performed. This simulation follows the Hadoop implementation, with a heartbeat mechanism to control the task execution.

DOI: 10.1002/cpe

The architecture of BlobSeer-Hadoop comprises the following modules: 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", as it occurs in the real implementation. Moreover, each chunk can be linked to more than one node, which allows chunk replica simulation. The BlobSeer-Hadoop simulation implements the node distribution in a single rack. The next version of BIGhybrid will use the API storage 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 simulated as an additional task cost and implements disk contention. Disk contention represents an additional computational cost where an user is sharing the same hardware resource over another virtual machine. The virtualization support module will later be integrated with the aid of SimGrid virtualization support, as described in [40].

4.3. BitDew-MapReduce Simulation Module

The implementation of MapReduce over BitDew is mainly targeted at desktop grid systems [16], and employs mechanisms to alleviate the impact of host churn and the problem of unavailability of direct communication between the hosts and a lack of host trust. The implementation relies on master and worker daemon programs. A MR API on top of BitDew handles 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. With the Hadoop implementation, when the network experiences unavailability, a heartbeat interval signals to the master that the host is dead. Nevertheless, in BitDew the network can be temporarily offline without undergoing any failure. The FT needs a synchronization schema, 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 5).

BIGhybrid implements speculative tasks to create compatibility with the implementation of the MapReduce framework. The speculative task is launched in the execution end for both *Map* and *Reduce* phases to accelerate the executions of stragglers. The task scheduling is implemented trough *task scheduling* module in each simulator, which follows the locality principle described. Hence, when the master node receives a heartbeat from a worker, and has checked the available slots for map processing, it will try to schedule a task in accordance with the following criteria:

1. An unassigned task that processes a chunk stored locally in the worker;
2. An unassigned task that is stored in another worker;
3. A speculative task that processes a local chunk;
4. A speculative task with non-local input.

A *Reduce* task does not have a locality and its input is spread among the workers that processed the *Map* tasks. For this reason, when assigning *Reduce* tasks, the scheduler distinguishes between unassigned and speculative tasks. In both the *Map* and *Reduce* phases, a speculative task is scheduled when all the regular tasks have already been processed or assigned to other workers.

4.4. BIGhybrid Integration 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, like the number of *Map* and *Reduce* tasks, for both BlobSeer-Hadoop and BitDew-MapReduce systems, 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 system execution analysis was implemented to assist in creating both homogeneous and heterogeneous platforms, and make execution traces based on visualization

DOI: 10.1002/cpe

traces supported by SimGrid. This toolkit enables users to analyze the whole execution system and change the strategies when needed. The traces can be individual, as well as for all the simulations in the system.

4.5. *Details of Volatility Module and Communication Process*

The volatility control module implements the FT mechanism, which is an environment to recover data and tasks of nodes that have a failure. Figure 4, adapted from [15], shows the BitDew-MapReduce synchronization schema that was implemented in the BIGhybrid simulator for failure detection. The node updates an *alivetime* variable at each *synchronization interval*. The *synchronization interval* is a period when the node synchronizes its state with the master. If the node goes offline, the status of the *alivetime* variable will be changed to unconnected and a failure will be detected. A period of failure time-out is the time that the master waits to change the node status from online to offline.

This is necessary because in the DG environment, several hosts are behind firewalls and slow links that can cause a lack of momentaneous connection without the node having a shutdown. This is defined by the user in the configuration file of the simulator to indicate a waiting time that begins with the last valid heartbeat. As each node is able to begin its own processing in an undefined time, each node has its own *period of failure time-out*. The master waits for a *transient period* during which it does not take any recovery action and does not send any new data or tasks to this node. When the *transient period* achieves a period of failure time-out, the *alivetime* status becomes offline. At this moment, the system emits a backup task from the replicas in the FT mechanism and removes the node from the database to avoid management overhead.

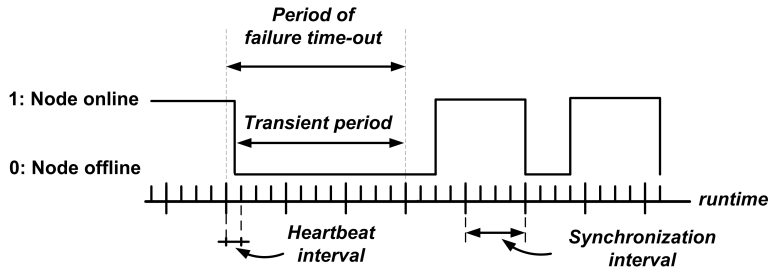


Figure 4. BitDew-MapReduce synchronization schema

The communication process occurs through a message exchange mechanism of SIMGRID Mailbox API. Short messages (SMS) are sent by the BIGhybrid simulator to task control and data transfers in a full-duplex communication channel. Each node initializes the worker, heartbeat and mailbox processes.

Figure 5 shows the main communication schema between the master and workers. When a worker starts, it sends a message to become registered on master and receives an identification called “wid”. This SMS registration give information about the processor type/characteristics and local disk size of the worker. After this, the master determines the computational capacity of the nodes and how much data will be sent by the Data Transfer service, after the data has been distributed to each node. The main communication mechanism is the heartbeat, which has a global time defined by the number of nodes.

After the node has received both the data and tasks, it begins the processing and the heartbeat goes sleep until the next synchronization interval. When the heartbeat is turned on, the node sends a SMS_STATUS to the master giving information about the progress of the task or conclusion of the execution. If the master does not receive a heartbeat, it calculates the period value of failure time-out and wait, and during this period it does not sends nothing to worker. If the worker sends a heartbeat before completing the waiting time, the period value of failure time-out is reinitialized. On the other hand, if the master detects a failure, it triggers a fault tolerance mechanism (FTM).

DOI: 10.1002/cpe

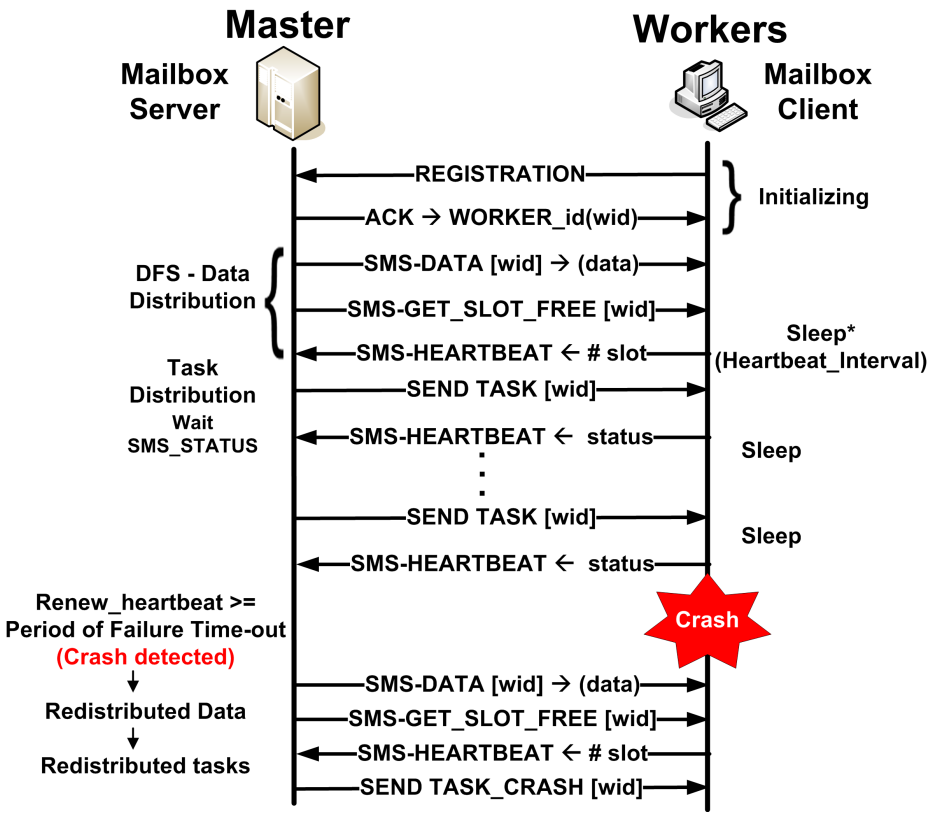


Figure 5. Communication schema

The FTM discovers which tasks have been finished by the worker shutdown, and finds a data replication to relaunch the task for another free node. The replication mechanism updates the data replication factor. The status of the worker is upgraded in the database and its "wid" is removed from the database. FTM is an important mechanism that is implemented in BIGhybrid and simulates real-world desktop grid environments. This behavior is demonstrated in Section 5.

The BIGhybrid simulator also implements a "slow beginner". This means that if a node begins late, any task failure and data from the node failure, will be launched for this node to be executed. This behavior is checked in BitDew-MapReduce if the failure occurs when the data is still begin distributed. Otherwise, if the job executions have already begun, the node does not receive any data or task.

4.6. BIGhybrid Project Characteristics

The BIGhybrid simulator was developed to achieve an accurate simulation and maintain features compatible with BlobSeer-Hadoop and MapReduce-BitDew for a real hybrid environment. In this way, the main properties of the hybrid environment are simulated. The simulator main features are summarized in Table III.

5. EVALUATION

This section describes the environment setup and results of the evaluation so that it can demonstrate the features and scalability of the simulator. The experiments are separated into the execution time profile and behavior execution. The execution time profile demonstrates how similar the simulation execution is to the real execution times and the behavior execution demonstrates which features are

DOI: 10.1002/cpe

Table III. BIGhybrid Simulator Features

Characteristics	BlobSeer-Hadoop	BitDew-MapReduce
Behavior	Simulated or found by trace files	Simulated or found by trace files
Build hardware platforms	Without limits	Without limits
Chunk-size	Defined by user	Defined by user
Computational Semantic	Hadoop-MapReduce Compatilble	Barrier-free computation
Data distribution	Data locality	Data locality according to the computational capacity of the machines
Environment	Homogeneous or Heterogeneous	Homogeneous, Heterogeneous and Volatile
DFS	Simulated by Matrix	Simulated by Matrix
Fault Tolerance Mechanism	Data and Task Replication	Data, Task Replication and Host Failure Recovery
Generation Traces and Logs	Yes	Yes
Storage	Simulated by cost or disk emulation	Simulated by cost or disk emulation
Straggler management	Average execution task	Machine computational capacity
Synchronization schema	Heartbeat by time stamp	Heartbeat by time stamp and Failure of Time-out Period
Virtualization Support	Disk contention	No
Network	Flat-Three, Ethernet, Token-Ring, P2P, Hierarchical and Non-Hierarchical	Flat-Three, Ethernet, Token-Ring, P2P, Hierarchical and Non-Hierarchical

reproduced in the simulator.The executions are repeated 30 times for each experiment in real-world experiments and a calculation is made of the means, standard deviation and coefficient of variation. Since the BIGhybrid is a deterministic simulator, its measures are compared through a statistical evaluation from the results of real-world experiments.

5.1. The Environment Setup

Four environments have been considered. The first experiment has three different clusters in a simulated environment. One cluster simulates homogeneous environment and has a 5-node cluster of 1 CPU with 2 cores each, 5.54 GFlops of processing capacity and 1 Gbps network. Two other clusters simulate the heterogeneous environment where one contains 5 heterogeneous machines with 1 CPU of 2 cores each and a network of 10 Mbps, and another contains 15 heterogeneous machines with 1 CPU of 2 cores each and a network of 10 Mbps, with 20% of volatile nodes. The machines in a heterogeneous environment have a processing capacity ranging from 4.76 GFlops to 6.89 GFlops, where this processing capacity is determined by a log-normal distribution according to [41]. The second experiment is formed of clusters from the Grid5000 environment. This grid is a experimental testbed, carried out under the INRIA ALADDIN development plan with support from CNRS, RENATER and several universities in France. The experiments were divided into two environments: First, there is a homogeneous environment formed of clusters (as described in Table IV) with a 1 Gbps network. Second, there is a heterogeneous environment formed of machines (as described in Table V.

Table IV. Grid5000 environment for homogeneous experiments

Site	# Host	Properties (# Processor,# Cores, RAM, HDD)	Performance (GFlops)
Sophia	16	2 x Intel Xeon E5520 @ 2.27 GHz, 4, 32 GB, 557 GB	55.46
Reims	32	2 x AMD Opteron 6164 HE @ 1.7 GHz, 12, 47 GB, 232 GB	121.30
Grenoble	64	2 x CPUs Intel Xeon E5520 @ 2.27 GHz, 4, 23 GB, 119 GB	55.45
Nancy	128	1 x Intel Xeon X3440 @ 2.53 GHz, 4, 16 GB, 298 GB	31.01

DOI: 10.1002/cpe

Table V. Grid5000 environment for heterogeneous experiments

Site	# Host	Properties (# Processor,# Cores, RAM, HDD)	Performance (GFlops)
Sophia	30	2 x AMD Opteron 2218 @ 2.6 GHz, 2, 4 GB , 232 GB	16.80
Sophia	22	2 x Intel Xeon E5520 @ 2.27 GHz,4, 32 GB, 557 GB	55.46
Total	52	50 workers and 2 BitDew servers	

The third experiment considers a cluster of 2,000 nodes and each node has 1 processor Intel Xeon X3440 @ 2.53 GHz, 4, 16 GB and a network of 1 Gbps. This configuration represents a characterization of MR applications devised by Chen [36] which was drawn on to define the large-scale setup. Chen examined the MR traces of two production environments from Yahoo and Facebook. The Yahoo traces were obtained from a 2,000 node cluster and contained 30,000 jobs spanning a period of over 3 weeks. The cluster was used to run applications that require batch, interactive and semi-streaming computations. For the purposes of this work, only "*aggregate, fast job*" applications characterized by Chen are considered. Table VI shows the details of these applications, including the number of *Jobs*, input average data size for each *Job*, *Map* time and *Reduce* time. This *Job* has 568 GB of input and 9,088 tasks with an execution time of 322.64 seconds from *Map* and 703.32 seconds from *Reduce*.

Table VI. Yahoo traces (2,000 machines cluster)

# Jobs	Input	*Shuffle*	Output	*Map* Time	*Reduce* Time	Label
21,981	174 MB	73 MB	6 MB	412	740	**Small jobs**
838	568 GB	76 GB	3.9 GB	270,376	589,385	**Aggregate, fast job**
91	206 GB	1.5 TB	133 MB	983,998	1,425,941	**Expand and aggregate jobs**
1,330	36 GB	15 GB	4 GB	15,021	13,614	**Data transformation**

The operational system is Debian Wheezy-x64 with Hadoop 1.2 for homogeneous environments and Debian Wheezy-x64 with BitDew 0.2.2 and Java SUN 1.6 for heterogeneous environments. The simulator is BIGhybrid version 1.0 - Build 3.11 with SimGrid 3.11.1 and it is available in https://github.com/Julio-Anjos/Bighybrid.

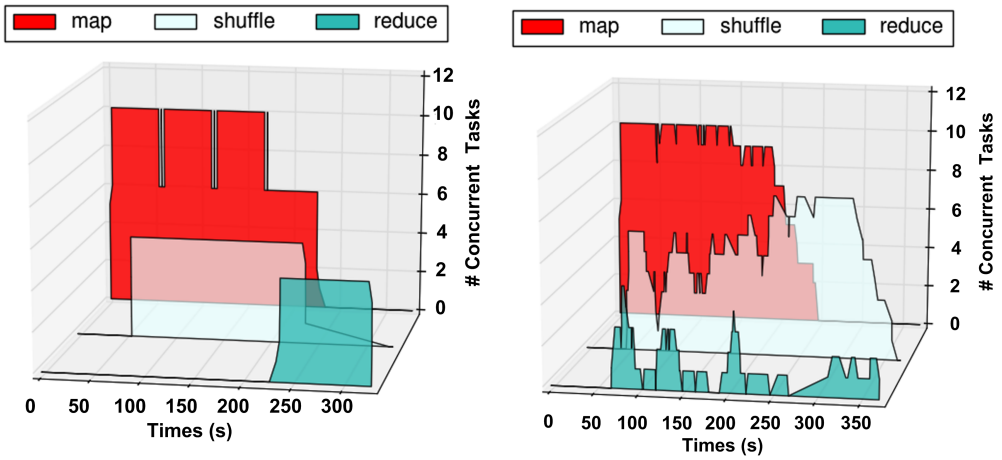
5.2. *Study of a Simulated MapReduce Execution*

In this experiment, an attempt is made to evaluate if the BIGhybrid simulator is able to simulate the main features of two existing MapReduce runtime execution environments, namely, BlobSeer-Hadoop and BitDew-MapReduce. The experiment consists of the simulation of MR execution using BlobSeer-Hadoop and BitDew-MapReduce in two different infrastructures, in homogeneous and heterogeneous cluster respectively. In this experiment, we seek to obtain the execution profile of the MR execution, *i.e.* the number of concurrent task executions during the *Map* and *Reduce* phases, and the number of data transfers during the *Shuffle* phase. The homogeneous environment was used to process 2GB of data, 5 mappers, 5 reducers and chunk size of 64 MB, while the heterogeneous environment was used to process 1.1 GB of data, 5 mappers, 5 reducers and chunk size of 16 MB.

Figure 6 and Figure 7 show the MR execution profile simulated by the BIGhybrid simulator. In Figure 6, the colors red, white and blue represent the *Map*, *Shuffle* and *Reduce* phases respectively. The execution time in the x-axis is measured in seconds, and the number of concurrent tasks for *Map*, *Reduce* and *Shuffle* in the y-axis are measured in units. Figure 6.a shows an execution of BlobSeer-Hadoop @ homogeneous environment. The *Map* tasks produce intermediary keys that are sent to reducers during the *Shuffle* phase, and the *Reduce* tasks begin once the *Map* tasks have been completed. The *Map* tasks are restricted to 10 concurrent tasks (two tasks per node). The

DOI: 10.1002/cpe

Shuffle begins after 5% of the *Map* tasks have been completed. The number of *Reduce* tasks is restricted to concurrent *Reduce* tasks, in this case 5 tasks. The *Reduce* phase begins after all the *Map* tasks have been completed. This shows the correct execution of the *barrier* implementation. This is a synchronization *barrier* between the *Map* and *Reduce* phase, in exactly the same way as was described in the execution of MR framework in Section 2. The BitDew-MapReduce execution @ heterogeneous environment is shown in Figure 6.b. The *Reduce* phase shows that the tasks start as soon as the machines have some data to process. This shows that the *barrier-free* behavior that was implemented in the BitDew-MapReduce can be reproduced. It should be noted that, as the link is 10 Mbps, the data transfers take longer to complete during the *Shuffle* phase.



(a) BlobSeer-Hadoop @ homogeneous environment (b) BitDew-MapReduce @ heterogeneous environment

Figure 6. MapReduce execution profile simulated by BIGhybrid simulator

Figure 7 shows the *MapReduce* execution time in a hybrid environment simulated by a BIGhybrid simulator. The colors red, green and blue are represented for *Map*, *Shuffle* and *Reduce* phases respectively. The execution time in the x-axis is measured in seconds, and the number of concurrent tasks for *Map*, *Reduce* and *Shuffle* in y-axis is measured in units. This shows the MR execution in a Hybrid environment from the previous experiment in a parallel execution. *Map* tasks have restricted number of amount concurrent tasks from each system. The chart demonstrates that *Reduce* tasks begin the data prefetching earlier in the hybrid infrastructure.

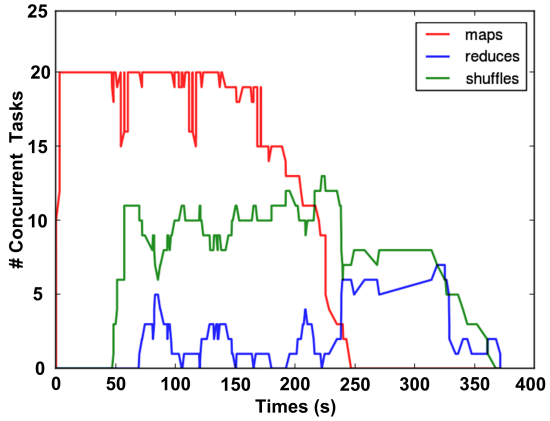


Figure 7. MapReduce execution profile in a hybrid environment simulated by a BIGhybrid simulator

DOI: 10.1002/cpe

5.3. Volatile Behavior and Failure Tolerance Mechanism

In this experiment, an attempt is made to evaluate if the BIGhybrid simulator is able to simulate accurately the Failure Tolerance Mechanism described in the BitDew-MapReduce environment, in Section 4.5. The experiment consists of the simulation of MR execution in a BitDew-MapReduce @ heterogeneous and volatile environment. Fifteen machines were used to process 8GB of data, 120 maps, 30 reduces, with a chunk size of 64 MB, and 20% of these machines are volatile nodes. Figure 8 shows the time period where the node is on-line or off-line. This is a real volatile behavior of Boinc traces, with volatile behavior of three hosts (A, B and C), represented in the y-axis. The x-axis shows the time period in seconds when a host is on-line, in the gray box, and when it is off-line without a box.

The experiment consists of Host A and B (MRA_Host 1 and MRA_Host 2 respectively). Host A and B begin the execution and then stop during a time period, and Host C (MRA_Host 3) begins a late execution. BIGhybrid obtains these traces from a trace file and reproduces the volatile behavior, by comparing the average execution time with the trace profile. The users can define three variables related to a volatile environment in the config files. One is *mra_dfs_replication* which defines the replication factor for the data; the other is *perc_num_volatile_node* which is related to the number of volatile nodes (as a percentage) and determines how many nodes that have a volatile behavior will be read from the traces file; and finally there is *failure_timeout* which determines the *period of failure time-out* for the fault tolerance mechanism as shown in Section 4.5. This time-out is defined in terms of n heartbeat periods.

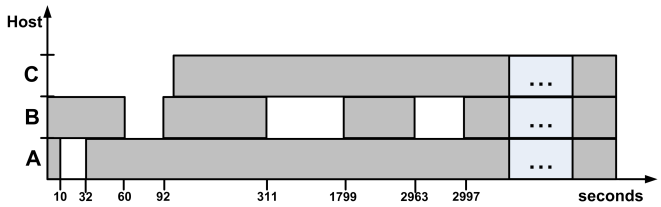


Figure 8. Time periods where a node is on-line in a volatile environment

Figure 9 shows the logs of the BIGhybrid simulator where this behavior is reproduced in the execution time. MRA_Host 1 stops during a synchronization interval and the last valid heartbeat is considered to be the beginning of the period of failure time-out, which in this case is 15 seconds and then the FTM system begins the failure recovery process. MRA_Host 2 stops before the synchronization interval when the next heartbeat detects the failure and, after the *period of failure time-out*, the system begins the FTM recovery for *Map* and *Reduce* tasks. The BIGhybrid simulator also detects late nodes (those that begin the execution time after the other nodes), for instance, in MRA_Host 3 this host is available for execution. After that, it is necessary to move the data from replica as *map_task* 107 and 115 or to do a prefetching as in the case of *reduce_task* 0.

On the basis of these experiments, we concluded that the BIGhybrid simulator is able to simulate a MR execution accurately; in particular, it was confirmed that two distinct *barrier* and *barrier-free* features are correctly executed. As well as this, there was a failure recovery of the fault tolerance mechanism in volatile environments.

5.4. Study of the Behavior Profile in the Grid5000 Environment

In this experiment, our aim is to evaluate if the BIGhybrid simulator is able to simulate the *MapReduce* execution from BlobSeer-Hadoop accurately in real environments with different workloads. The experiment consists of the execution in a homogeneous environment on Grid5000. In the case of BlobSeer-Hadoop execution @ homogeneous environments, the experiment has 16, 32, 64 and 128 nodes, and the configuration of clusters is described in Table IV. In conducting this experiment, we seek to obtain the execution profile of the MR execution that is related to different workloads during the *Job*, *i.e.* the amount of time required for the execution of both the *Map* and *Reduce* phases.

DOI: 10.1002/cpe

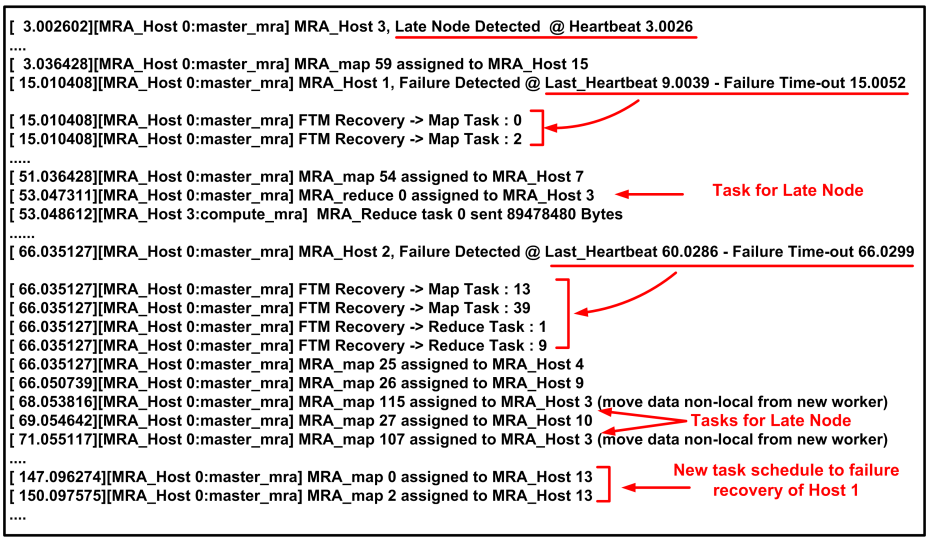


Figure 9. Logs of BIGhybrid simulator in execution time

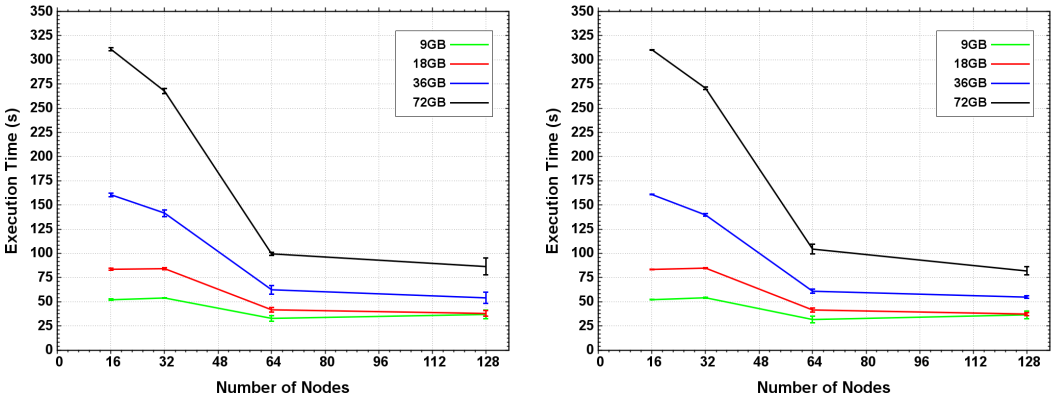
The workload includes 9 GB - 141 chunks, 18 GB - 302 chunks, 36 GB - 571 chunks and 72 GB - 1143 chunks. Each Map processes 64 MB of data and produces an output of 42.85% of input data. The number of mappers and reducers is equal to the number of cluster nodes. The application is a function that calculates the average time when the nodes are free to execute a task. These times are available in a log file. The *Map* function reads the *id*, from each line of the input data. This *id* represents a node identification, where is associated an execution time. If the time is longer than 300 seconds, an intermediate key will be emitted with *id/time*. In the *Reduce* function, the average for each *id* is calculated and a new key *id/average* is emitted. Each experiment was executed 30 times and the result was an average time. In the case of BIGhybrid simulator, it is necessary to carry out a calibration procedure. This calibration is necessary to determine the simulation parameters, such as, task costs and network configuration, to create a platform with the same machine configuration as Grid5000, to define the job configuration and so on. After this calibration the number of machines needed to run the experiments is changed.

Figure 10 shows the BlobSeer-Hadoop execution time in accordance with the number of nodes in the homogeneous environment. Figure 10.a shows the execution in Grid5000 and Figure 10.b shows the execution in a BIGhybrid simulator. The colors green, red, blue and black represents the workloads of 9GB, 18GB, 36GB and 72GB respectively. The execution time in the y-axis is measured in seconds, and the number of nodes in the x-axis is measured in units.

The BIGhybrid simulation has a similar execution time profile and a tendency to display the same behavior in the inclination curve for the *Job*. The execution on Grid5000, in Figure 10.a, has a standard deviation of 8.8% with a workload of 72 GB (more details are given in the statistical evaluation section - Section 5.5 -) that produces the highest data dispersion. The BIGhybrid simulation, in Figure 10.b, shows a simulation variation of $\approxeq 4\%$ for 72 and $\approxeq 5\%$ for 64 nodes, when compared with the Grid5000 execution, as a result of this data dispersion. This variation is a measure of absolute error in percentage (MAPE) and it is calculated as MAPE $= \frac{1}{n}\sum|\frac{\overline{\mu}-X}{\overline{\mu}}| * 100$, where X is the measure of simulation, $\overline{\mu}$ is average of the measures and n is the sample size. This variation occurs because the simulation is an approximation of the real-world execution. This is acceptable when this is compared with the execution time in the Grid5000 environment (Figure 10.a). A statistical analysis is conducted in the next section based these experiments.

In the next experiments, our aim is to evaluate if the BIGhybrid simulator is able to simulate the different phases of *MapReduce* execution accurately from BlobSeer-Hadoop and BitDew-MapReduce in real environments with different workloads. The experiment involves executing two different applications in different infrastructures in homogeneous and heterogeneous environments

DOI: 10.1002/cpe

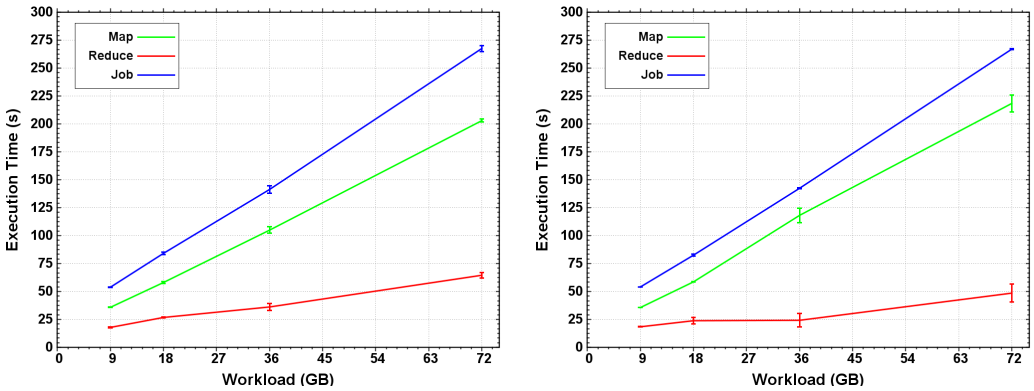


(a) G5000 environment (b) BIGhybrid simulator

Figure 10. BlobSeer-Hadoop execution time according to the number of nodes in the homogeneous environment

in Grid5000. When conducting this experiment, it was necessary to evaluate if the execution time for *Map* and *Reduce* phases, (and the total execution time for the *Job*) requires the same amount of time as the Grid5000 and BIGhybrid simulator and take account of homogeneous and heterogeneous environments. The BlobSeer-Hadoop execution @ homogeneous environments uses 32 nodes and the cluster configuration is described in Table IV. In the BitDew-MapReduce execution @ heterogeneous environments is used 50 nodes for the workers and 2 nodes for the BitDew services; this cluster is described in Table V. The workload is 9 GB - 141 chunks, 18 GB - 302 chunks, 36 GB - 571 chunks and 72 GB - 1143 chunks, and the chunk size is of 64 MB for BlobSeer-Hadoop. In BitDew-MapReduce the workload is 3 GB - 192 chunks, 13 GB - 768 chunks, 26 GB -1536 chunks, and the chunk size is of 16 MB. The application for BlobSeer-Hadoop was described in the previous experiment. The application in BitDew-MapReduce execution is wordcount. Wordcount is a popular micro-benchmark widely used in the community, that is contained in the Hadoop distribution [42]. Each experiment was executed 30 times and the result is an average time.

Figure 11 and Figure 12 show the BlobSeer-Hadoop and BitDew-MapReduce execution times respectively. The green, red and blue colors represent *Map* and *Reduce* phases and *Job* respectively. The execution time in the y-axis is measured in seconds and the workload in the x-axis is measured in gigabytes (GB). Figure 11 shows the average execution time for *Map*, *Reduce* and *Job* executions for BlobSeer-Hadoop @ homogeneous environments.



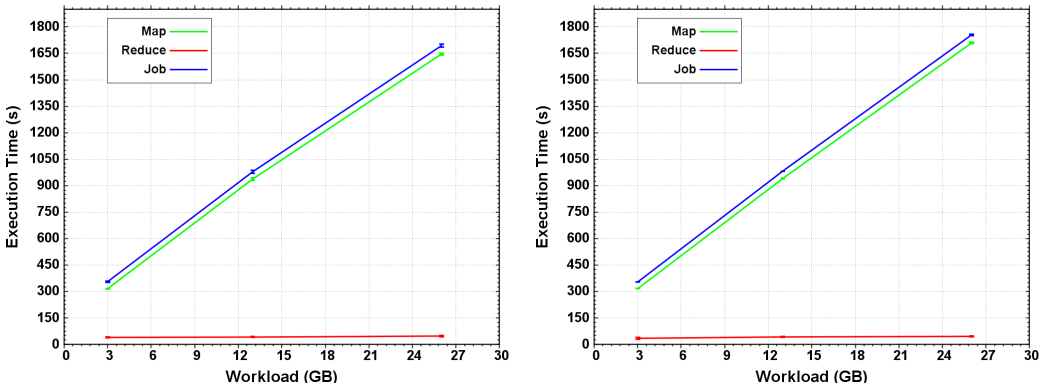
(a) G5000 environments - 32 nodes (b) BIGhybrid simulator - 32 nodes

Figure 11. BlobSeer-Hadoop execution time @ homogeneous environments

DOI: 10.1002/cpe

Figure 11.b shows the simulation carried out by the BlobSeer-Hadoop in BIGhybrid simulator. The behavior simulated is similar, but has a slight distortion in the *Reduce* phase execution that generates a MAPE of 0.58% in the worst case scenario. The distortion has little effect because the *Map* execution time is more significant than the *Reduce* execution time. The *Job* has $\approxeq$ 54 s @ 9 GB and $\approxeq$ 267 @ 72 GB, with a MAPE of 0.44% and 0.24% respectively. The *Map* phase has a greater weight in terms of number of tasks in the *Job* time, than the *Reduce* phase and, as a result of this, the distortion of *Reduce* time execution is minimized for the total time required for *Job*. The execution time has a reasonable approximation, if we consider the *Job* total time.

Figure 12 shows execution time for *Map*, *Reduce* and *Job* executions for BitDew-MapReduce @ heterogeneous environment. Figure 12.a shows the average execution in Grid5000. The Bitdew-MapReduce execution processes the *Combine* function, that is a processing of similar keys about *Map* phase before to send the data to the next phase. It is not possible to determine if a data partition will have more or less key to join in *Combine* function. The simulator have a highest execution time to *Map* function to simulate this additional work, but it is not a approximation easy to define. The execution profile shows that the *Map* spends more execution time on this application and Figure 12.b shows that BitDew-MapReduce simulation follows the same procedure and has a relative precision in the BIGhybrid simulator. *Job* has $\approxeq$ 353 s @ 3 GB and $\approxeq$ 1754 @ 26 GB, with MAPE of 0.58% and 3.59% respectively; more details about *Reduce* are shown in Table IX, in Section 5.5. The *Map* phase has a greater weight in terms of task number and execution time (in the *Job* time), than the *Reduce* phase and because of this the *Job* total time is near to the *Map* time.



(a) G5000 environments - 50 hosts

(b) BIGhybrid Simulator - 50 hosts

Figure 12. BitDew-MapReduce execution time @ heterogeneous environments

As a result of these experiments, we evaluate that the BIGhybrid simulator is able to simulate the different phases of *MapReduce* execution from BlobSeer-Hadoop and BitDew-MapReduce. In Section 5.5, the experiments of Figure 10 and Figure 12, BlobSeer-Hadoop and BitDew-MapReduce executions, respectively, are analyzed from a statistical perspective.

5.5. *Statistical Evaluation*

In this experiment, our aim is to evaluate the accuracy of the BIGhybrid simulator for the BlobSeer-Hadoop execution. Table VII shows the BlobSeer-Hadoop execution time in accordance with the number of nodes in the homogeneous environment (as shown in Figure 10), which is described in the previous section. The measurements in the table are based on the Grid5000 and BIGhybrid experiments. These measurements are grouped in accordance with the node numbers, execution type and the workloads of 9GB, 18GB, 36GB and 72GB. First, in the Grid5000 execution, the average ($\overline{\mu}$) is the mean of the measured results in Grid5000. The standard deviation (σ) of these measures is $\sigma = \sqrt{\frac{\sum(x-\overline{\mu})^2}{n}}$, where x is the measure and n is the sample size. The coefficient of variation (CV) is the ratio of standard deviation to the average (in percentage terms). This value

DOI: 10.1002/cpe

is a standardized measurement for analyzing the average dispersion of all measures together. The value is calculated as $CV = \frac{\sigma}{\overline{\mu}} * 100$. The T_{val} is a confidence interval for the standard deviation measures, in this case equal to 95%. Second, in the BIGhybrid execution, the measure for simulation is printed in DUT (Device under test). The mean absolute percentage error (MAPE) is calculated as MAPE $= \frac{1}{n} \sum |\frac{\overline{\mu} - X}{\overline{\mu}}| * 100$, where X is the measure of simulation and MAPE is an accurate measurement that indicates how near a sample is to the average; if the value is low, it indicates a good degree of accuracy [43].

The BIGhybrid execution measurements (DUT) for the Blobseer-Hadoop simulation shows a MAPE measure of 4.92% in the worst case scenario, which is very acceptable. A slight distortion (with 64 and 128 nodes and 72GB workload) is shown by a higher MAPE variation. One reason for this is network contention that is caused by the network being shared with other users, but this is a common resource share, as in the Internet. Otherwise, the simulator does not capture this variation. Better MAPE results are obtained in BIGhybrid experiments when the real-world experiments have a low standard deviation. This is because when there are only a few machines, there are more tasks to execute locally and the execution has less data movement. The BIGhybrid is a deterministic simulator and simulates data locality feature that is reflected in lower variations for MAPE. Most executions of the BIGhybrid simulator are carried out with in the standard deviation confidence interval of 95%. Hence, the results of the BIGhybrid simulator have a good rate of accuracy for executions of Blobseer-Hadoop in homogeneous environments.

Table VII. Measuremens of the BlobSeer-Hadoop execution in a homogeneous environment

		Grid5000 measures				**BIGhybrid measures**	
# Nodes	**# Input**	$\overline{\mu}$	σ	**CV(%)**	T_{val}	**DUT**	**MAPE(%)**
16	9GB	52.05	0.82557	1.59612	51.66 - 52.44	52.15	0.19
	18GB	83.55	1.09904	1.31543	83.04 - 84.06	83.41	0.16
	36GB	160.55	1.93241	1.20361	159.65 - 161.45	160.77	0.17
	72GB	310.85	1.53125	0.49260	310.13 - 311.57	310.36	0.16
32	9GB	53.9	0.32544	0.61037	53.77 - 54.03	54.14	0.44
	18GB	84.23	0.86389	1.02559	83.90 - 84.56	84.80	0.67
	36GB	141.64	3.42900	2.42085	140.26 - 143.03	142.47	0.59
	72GB	267.67	2.68093	1.00159	266.58 - 268.75	267.01	0.24
64	9GB	32.83	2.98771	9.10130	40.81 - 42.58	31.65	3.59
	18GB	41.69	2.18752	5.24664	40.81 - 42.58	41.51	0.43
	36GB	62.39	4.45377	7.13804	60.70 - 64.09	60.97	2.27
	72GB	99.49	1.52223	1.53000	98.83 - 100.15	104.39	4.92
128	9GB	36.98	4.63255	12.53275	35.22 - 38.74	38.49	4.08
	18GB	37.95	2.87843	7.58405	36.64 - 39.26	37.33	1.63
	36GB	54.02	5.80187	10.74092	51.51 - 56.52	54.78	1.41
	72GB	85.39	8.87972	10.39907	81.55 - 89.23	81.79	4.22

It is necessary to evaluate if the simulations have a similar distribution to that found in a real-world execution. This relation between the Grid5000 and BIGhybrid experiments is analyzed in accordance with the correlation coefficient (Corr$_{x,y}$) [44]. The Corr$_{x,y}$ is calculated by the Pearson Method, as in Equation 1, where "x" is the BIGhybrid measure, "y" is the Grid5000 measure and "n" is the number of measures. The confidence interval (T_{val}) for this analysis is equal to 95%. This coefficient is calculated on the basis of the Equation 1 and in Table VII, and the results are set out in Table VIII. Figure 13 shows the dispersion chart based on the results from Table VIII.

$$r = \frac{n \sum xy - (\sum x \cdot \sum y)}{\sqrt{[n \sum x^2 - (\sum x)^2][n \sum y^2 - (\sum y)^2]}} \quad (1)$$

DOI: 10.1002/cpe

The measures of standard deviation and coefficient of variation, in Table VII, demonstrate that there is a little dispersion around the average. This is proved by the fact that there is a high correlation coefficient between the Grid5000 execution and the BIGhybrid simulation, in Table VIII (near to 99%). The analysis of dispersion (in the diagram) for 16, 32, 64 and 128 nodes, in Figure 13, shows a positive correlation. The red line represents the linear regression which is obtained from the calculation of the correlation. On the basis of these observations, it can be concluded that the BIGhybrid simulations achieve a good approximation to the MapReduce executions in hybrid environments for the Blobseer-Hadoop execution. This means that the BIGhybrid simulation can be used to simulate a hybrid environment and evaluate the likely behavior of the nodes in these environments.

Table VIII. Grid5000 vs BIGhybrid - Correlation coefficient for Blobseer-Hadoop

# Nodes	$\text{Corr}_{x,y}$	T_{val}
16	0.9999981	0.9999064 - 1.0000000
32	0.9979155	0.9000929 - 0.9999586
64	0.9989227	0.9471165 - 0.9999786
128	0.9983309	0.9192125 - 0.9999669

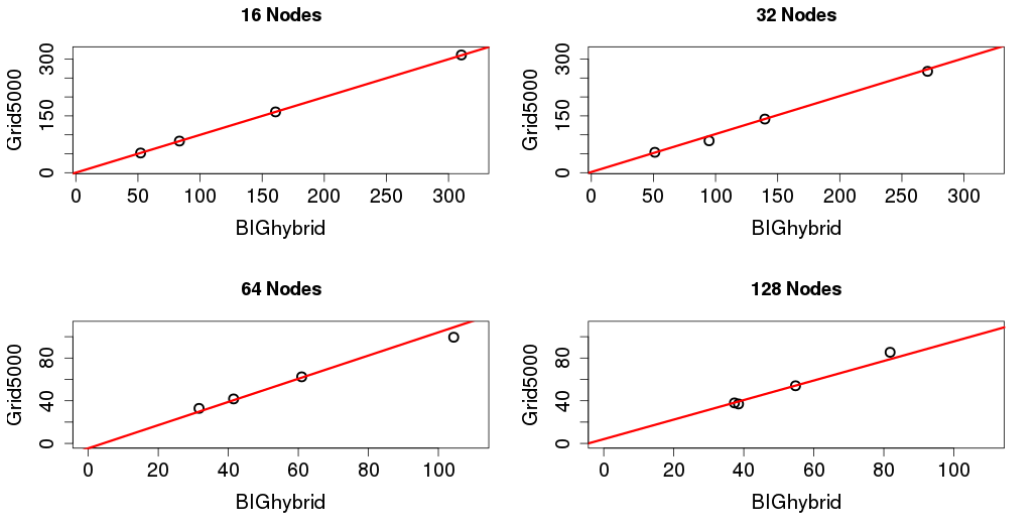


Figure 13. Dispersion chart - Grid5000 vs. BIGhybrid - to Blobseer-Hadoop

In the next analysis, our aim is to evaluate the accuracy of the BIGhybrid simulator for the Bitdew-MapReduce execution. The experiment was described in the previous section. The data in Figure12 for the BitDew-MapReduce @ heterogeneous environment is set out in Table IX. The Table is formed with the aid of Grid5000 and BIGhybrid measures. These measures are grouped in accordance with their "function type" and workload input. The mean ($\overline{\mu}$) is the average of measured results in the Grid5000 execution. The standard deviation (σ), coefficient of variation (CV), confidence interval T_{val} and the mean are evaluated in accordance with the *Map, Reduce* and *Job* executions. The BIGhybrid execution is the DUT measure and for each measure is linked to a mean absolute percentage error (MAPE), that is calculated as described earlier.

The execution in a real-world environment has a high dispersion rate for the *Reduce* function. It is related to the size of data transfers from intermediate data. This size depends on the generated partitions in the *Map* phase that can change as a result of the input data received. The bandwidth fluctuation is another factor to consider, because it reflects on time data transfers. Nevertheless,

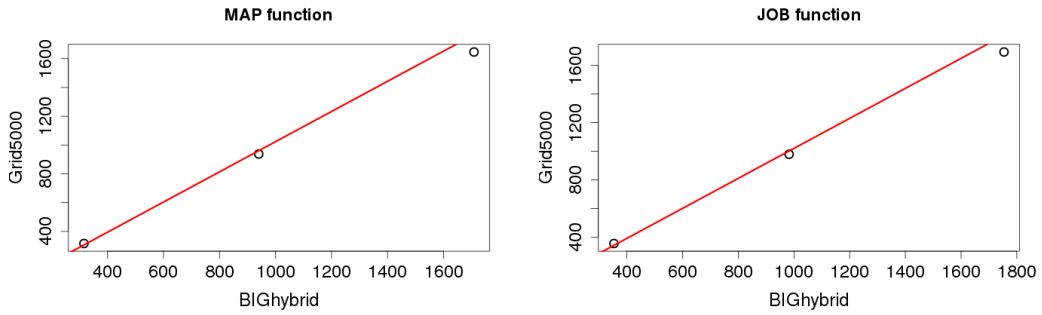
DOI: 10.1002/cpe

Table IX. Measures of BitDew-MapReduce execution a the heterogeneous environment

		Grid5000 measures				BIGhybrid measures	
# Input	**Function type**	$\overline{\mu}$	σ	**CV(%)**	T_{val}	**DUT**	**MAPE(%)**
3GB	MAP	316.12	1.59708	0.50521	314.98 - 317.26	316.00	0.04
	REDUCE	39.48	3.02096	7.65188	37.32 - 41.64	37.52	4.96
	JOB	355.60	3.42085	0.96199	353.15 - 358.05	353.52	0.58
13GB	MAP	937.65	7.77135	0.82881	932.09 - 943.21	940.23	0.28
	REDUCE	41.39	2.80335	6.77301	39.38 - 43.39	42.30	2.20
	JOB	979.04	7.82037	0.79878	973.45 - 984.63	982.53	0.05
26GB	MAP	1,646.06	5.19790	0.31578	1,642.34 - 1,649.78	1,708.80	3.81
	REDUCE	47.35	4.80769	10.15352	43.91 - 50.79	45.44	4.03
	JOB	1,693.41	7.60722	0.44922	1,687.97 - 1,698.85	1,754.24	3.59

the execution time takes up more that 80% of the *Map* function, and as a result, the dispersion effect is minimized. The DUT values within the standard deviation confidence interval show that the BIGhybrid simulation has a good degree of accuracy. The MAPE measure of 4.96% in the worst case scenario is very acceptable too, and the measures for *Map* have a maximum of 3.81%.

The correlation coefficient between the Grid5000 and BIGhybrid executions for BitDew-MapReduce @ heterogeneous environment is 99.9% for the *Job*. The dispersion chart in Figure 14 shows that the dispersion has a positive correlation. The red line represents the linear regression which is obtained from the calculation of the correlation. The chart demonstrates that the values are near this line which suggests there is a good approximation to the simulation.



(a) Map Function - 50 hosts

(b) Job Function - 50 hosts

Figure 14. Dispersion chart - Grid5000 vs. BIGhybrid to BitDew-MapReduce

The statistical evaluation for BlobSeer-Hadoop and BitDew-MapReduce simulations provides a simulation with a relative degree of accuracy. The mean absolute percentage error ($\approxeq$ 5% in the worst case scenario for heterogeneous and homogeneous environments) shows that the simulator can be an efficient evaluative instrument for hybrid infrastructures. The high correlation coefficient between the Grid5000 execution and the BIGhybrid simulation (around 99%) indicates that the real-world behavior is reproduced by the BIGhybrid simulator. It can thus be concluded that the simulator has a reproducible capability and is able to achieve a relative degree of accuracy in real-world experiments.

5.6. A Study of the Reproducibility of Real Experiments

In this experiment, our aim is to evaluate if the BIGhybrid simulator is able to reproduce the results obtained from the synthetic applications from real-world experiments. The experiment consists of

DOI: 10.1002/cpe

simulating BlobSeer-Hadoop from data collected from homogeneous executions carried out in a Yahoo cluster. In conducting this experiment, it was necessary to obtain the execution time for the *Map* and *Reduce* phases and to compare this with the times from Table VI that consider a homogeneous environment in Cloud applications. With regard to the BlobSeer-Hadoop execution @ homogeneous environments, 2000 nodes were used (as described in Section 5.1). The *aggregated fast job* applications characterized by Chen were taken into account. The workload has 568 GB of input and 9,088 tasks, and each *Job* has an execution time of 322.64 seconds from *Map* and 703.32 seconds from *Reduce*. The number of mappers is 2,000 and of the reducers is 1,000.

Figure 15 shows the Bighydrid simulation of the MapReduce execution from 2000 hosts @ Yahoo traces. The red, blue and green colors represent *Map*, *Reduce* and *Shuffle* phases respectively. In the y-axis, the number of concurrent tasks is measured in units and the time of the x-axis is measured in seconds. The number of *Map* and *Reduce* tasks is restricted to two task per node, *i.e.*, 4,000 tasks for the *Map* phase and 2,000 tasks for the *Reduce* phase. The execution time is 305.13 s for the *Map* phase and 673.32 s for the *Reduce* phase, and the simulation error is $\approxeq 5\%$ with regard to one *Job* in Table VI. This error is a calculation of ERROR$= |\frac{\vartheta - X}{\vartheta}| * 100$, where ϑ is the execution time of *aggregated fast job* execution and X is the measured time of simulator.

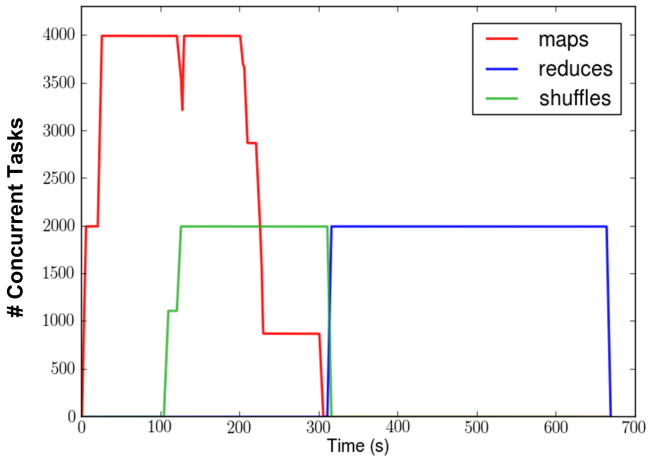


Figure 15. Bighydrid simulation of MapReduce execution from 2000 hosts @ Yahoo traces

As a result of this experiment, we concluded that the BIGhybrid simulator is able to reproduce experiments from synthetic applications of real-world experiments. This shows the scalability of the simulator, its reproducible capabilities and its suitability for investigative research of new strategies based on a hybrid infrastructure.

6. 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 infrastructure might be a feasible low-cost alternative to simply using Cloud environments in function of the free-cost of DG resources available.

In this study, the characteristics of a hybrid infrastructures were introduced and the feasibility of integrating Cloud and DG was demonstrated, by carrying out simulations to define the best strategies for the implementation. The experiments evaluated three different studies. In the study of the simulated *MapReduce* execution, the experiments showed that the BIGhybrid simulator is able to simulate a MR execution accurately. It was found that two distinct *barrier* and *barrier-free* features were correctly executed. As well as this, was shown the implementation of the failure recovery behavior of the fault tolerance mechanism in volatile environments. In the study of the behavior profile in the Grid5000 environment, it was shown that the BIGhybrid simulator is able

DOI: 10.1002/cpe

to simulate the different phases of *MapReduce* execution from BlobSeer-Hadoop and BitDew-MapReduce when this is compared with real environments and different workloads. In the study of the reproducibility of the real-world experiments the BIGhybrid simulator is able to reproduce experiments involving synthetic applications from real-world experiments. This demonstrates the scalability of the simulator, its reproducible capability and its suitability for investigative research into new strategies to evaluate the implementation of a hybrid infrastructure with *MapReduce* applications.

The experiments and statistical evaluation proved that the simulator has a reproducible capability based on real-world experiments and provides evidence that the validation goals have been achieved. This means that, the BIGhybrid simulator makes it possible to evaluate MapReduce strategies that involve the adoption of hybrid infrastructures. This suggests that it is possible to overcome problems through the adoption of determined strategies with a relative degree of accuracy.

It is necessary to conduct further experiments to define what strategies are required for the adoption of hybrid infrastructures with the results obtained from real-world systems. The next stage includes improving disk simulation and also implementing of machine migration support in the BIGhybrid simulator.

7. ACKNOWLEDGMENTS

This work was partly supported by CAPES - Foundation for Coordinating the Improvement of Higher Education Personnel - (Process BEX 14966/13-1). This work was supported by the Agence National de la Recherch - French National Research Agency -, under grant ANR-10-SEGI-001. The experiments discussed in this paper were conducted with the aid of the Grid'5000 experimental testbed, under the INRIA ALADDIN development plan with support from CNRS, RENATER and a number of universities (see https://www.grid5000.fr).

8. COPYRIGHT

REFERENCES

1. Dean J, Ghemawat S. MapReduce - A Flexible Data Processing Tool. *Communications of the ACM* 2010; **53**(1):72–77, doi:10.1145/1629175.1629198.
2. White T. *Hadoop - The Definitive Guide*, vol. 1. 3rd edn., OReilly Media, Inc., 2012.
3. Mell P, Grance T. The NIST Definition of Cloud Computing. *Technical Report* Sep 2011. URL http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.
4. Cérin C, Fedak G ((eds.)). *Desktop Grid Computing*. 1st edn., Numerical Analysis and Scientific Computing, CRC Press, 2012.
5. Anjos JCS, Fedak G, Geyer CF. BIGhybrid – A Toolkit for Simulating MapReduce in Hybrid Infrastructures. *Computer Architecture and High Performance Computing Workshop (SBAC-PADW), 2014 International Symposium on*, 2014; 132–137, doi:10.1109/SBAC-PADW.2014.8.
6. Casanova H, Giersch A, Legrand A, Quinson M, Suter F. Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms. *Journal of Parallel and Distributed Computing* Jun 2014; **74**(10):2899–2917.
7. Fedak G, He H, Cappello F. BitDew: a programmable environment for large-scale data management and distribution. *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, SC '08, IEEE Press: Piscataway, NJ, USA, 2008; 45:1–45:12.
8. Moca M, Silaghi G, Fedak G. Distributed Results Checking for MapReduce in Volunteer Computing. *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE Int. Symposium on*, 2011; 1847–1854, doi:10.1109/IPDPS.2011.351.
9. Nicolae B, Moise D, Antoniu G, Bouge L, Dorier M. BlobSeer: Bringing high throughput under heavy concurrency to Hadoop Map-Reduce applications. *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, 2010; 1–11, doi:10.1109/IPDPS.2010.5470433.
10. Kolberg W, Marcos PDB, Anjos JCS, Miyazaki AKS, Geyer CR, Arantes LB. MRSG - A MapReduce simulator over SimGrid. *Parallel Comput.* Apr 2013; **39**(4-5):233–244, doi:10.1016/j.parco.2013.02.001.

11. Anjos JC, Carrera I, Kolberg W, Tibola AL, Arantes LB, Geyer CR. MRA++: Scheduling and Data Placement on MapReduce for Heterogeneous Environments. *Future Generation Computer Systems* jan 2015; **42**(0):22–35, doi:10.1016/j.future.2014.09.001. URL http://www.sciencedirect.com/science/article/pii/S0167739X14001642.
12. Kondo D, Javadi B, Iosup A, Epema D. The Failure Trace Archive: Enabling Comparative Analysis of Failures in Diverse Distributed Systems. *(CCGrid), 10th IEEE/ACM Int. Conference on Cluster, Cloud and Grid Computing*, IEEE Computer Society, 2010; 398–407, doi:10.1109/CCGRID.2010.71.
13. INRIA, CNRS. Grid5000 Mar 2015. URL https://www.grid5000.fr/mediawiki/index.php/Grid5000:Network.
14. Antoniu G, Bigot J, Blanchet C, Bouge L, Briant F, Cappello F, Costan A, Desprez F, Fedak G, Gault S, *et al.*. Scalable Data Management for Map-Reduce-based Data-Intensive Applications: A View for Cloud and Hybrid Infrastructures. *Int. Journal of Cloud Computing* Feb 2013; **2**:150–170.
15. Tang B, Fedak G. Analysis of Data Reliability Tradeoffs in Hybrid Distributed Storage Systems. *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International*, 2012; 1546–1555, doi:10.1109/IPDPSW.2012.195.
16. Lu L, Jin H, Shi X, Fedak G. Assessing MapReduce for Internet Computing: A Comparison of Hadoop and BitDew-MapReduce. *Proceedings of the 2012 ACM/IEEE 13th Int. Conference on Grid Computing*, GRID '12, IEEE Computer Society: Washington, DC, USA, 2012; 76–84, doi:10.1109/Grid.2012.31.
17. Zou Q, Li XB, Jiang WR, Lin ZY, Li GL, Chen K. Survey of MapReduce frame operation in bioinformatics. *Briefings in Bioinformatics* Feb 2013; :2–11doi:10.1093/bib/bbs088. URL http://bib.oxfordjournals.org/content/early/2013/02/07/bib.bbs088.abstract.
18. McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A, Garimella K, Altshuler D, Gabriel S, Daly M, *et al.*. The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Research* 2010; **20**(9):1297–1303, doi:10.1101/gr.107524.110.
19. NCBI. A Base Pathogenic Mutations. *Technical Report* 2014. URL ftp://ftp.ncbi.nlm.nih.gov/snp/organisms/human_9606/gene_report.
20. Kinsella RJ, Kahari A, Haider S, Zamora J, Proctor G, Spudich G, Almeida-King J, Staines D, Derwent P, Kerhornou A, *et al.*. Ensembl BioMarts: a hub for data retrieval across taxonomic space. *Database* Jul 2011; **2011**:1–9.
21. Jayalath C, Stephen J, Eugster P. From the Cloud to the Atmosphere: Running MapReduce across Data Centers. *Computers, IEEE Transactions on* Jan 2014; **63**(1):74–87, doi:10.1109/TC.2013.121.
22. Tudoran R, Costan A, Wang R, Bouge L, Antoniu G. Bridging Data in the Clouds: An Environment-Aware System for Geographically Distributed Data Transfers. *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, 2014; 92–101, doi:10.1109/CCGrid.2014.86.
23. Krish K, Anwar A, Butt AR. HATS: A Heterogeneity-Aware Tiered Storage for Hadoop. *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, 2014; 502–511.
24. Zheng Z, Gui Y, Wu F, Chen G. STAR: Strategy-Proof Double Auctions for Multi-Cloud, Multi-Tenant Bandwidth Reservation. *Computers, IEEE Transactions on* 2014; **PP**(99):1–14, doi:10.1109/TC.2014.2346204.
25. Anderson DP. BOINC: A System for Public-Resource Computing and Storage. *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, GRID '04, IEEE Computer Society: Washington, DC, USA, 2004; 4–10, doi:10.1109/GRID.2004.14. URL http://dx.doi.org/10.1109/GRID.2004.14.
26. Fedak G, Germain C, Neri V, Cappello F. XtremWeb: a generic global computing system. *Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on*, 2001; 582–587.
27. Delamare S, Fedak G, Kondo D, Lodygensky O. SpeQuloS: a QoS service for BoT applications using best effort distributed computing infrastructures. *Proceedings of the 21th international symposium on High-Performance Parallel and Distributed Computing*, HPDC '12, ACM: New York, NY, USA, 2012; 173–186, doi:10.1145/2287076.2287106. URL http://doi.acm.org/10.1145/2287076.2287106.
28. Ostermann S, Plankensteiner K, Prodan R, Fahringer T. GroudSim: An Event-Based Simulation Framework for Computational Grids and Clouds. *Euro-Par 2010 Parallel Processing Workshops*, *Lecture Notes in Computer Science*, vol. 6586, Guarracino M, Vivien F, Träff J, Cannatoro M, Danelutto M, Hast A, Perla F, Knüpfer A, Di Martino B, Alexander M (eds.). Springer Berlin Heidelberg, 2010; 305–313, doi:10.1007/978-3-642-21878-1_38. URL http://dx.doi.org/10.1007/978-3-642-21878-1_38.
29. Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience* 2011; **41**(1):23–50, doi:10.1002/spe.995. URL http://dx.doi.org/10.1002/spe.995.
30. Sulistio A, Cibej U, Venugopal S, Robic B, Buyya R. A toolkit for modelling and simulating data Grids: an extension to GridSim. *Concurrency and Computation: Practice and Experience* 2008; **20**(13):1591–1609.
31. Bux M, Leser U. DynamicCloudSim: Simulating heterogeneity in computational clouds. *Future Generation Computer Systems* 2015; **46**(0):85–99, doi:10.1016/j.future.2014.09.007. URL http://www.sciencedirect.com/science/article/pii/S0167739X14001770.
32. Buyya R, Ranjan R, Calheiros RN. InterCloud: Utility-oriented Federation of Cloud Computing Environments for Scaling of Application Services. *Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing - Volume Part I*, ICA3PP'10, Springer-Verlag: Berlin, Heidelberg, 2010; 13–31, doi:10.1007/978-3-642-13119-6_2. URL http://dx.doi.org/10.1007/978-3-642-13119-6_2.
33. Kohne A, Spohr M, Nagel L, Spinczyk O. FederatedCloudSim: A SLA-aware Federated Cloud Simulation Framework. *Proceedings of the 2Nd International Workshop on CrossCloud Systems*, CCB '14, ACM: New York, NY, USA, 2014; 3:1–3:5, doi:10.1145/2676662.2676674. URL http://doi.acm.org/10.1145/2676662.2676674.
34. Tang W, Jenkins J, Meyer F, Ross R, Kettimuthu R, Winkler L, Yang X, Lehman T, Desai N. Data-Aware Resource Scheduling for Multicloud Workflows: A Fine-Grained Simulation Approach. *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*, 2014; 887–892, doi:10.1109/CloudCom.2014.

19.
35. Schad J, Dittrich J, Quiané-Ruiz JA. Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance. *Proc. VLDB Endow.* Sep 2010; **3**(1-2):460–471.
36. Chen Y, Ganapathi A, Griffith R, Katz R. The Case for Evaluating MapReduce Performance Using Workload Suites. *IEEE 19th Int. Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems*, (MASCOTS), IEEE Computer Society, 2011; 390–399, doi:10.1109/MASCOTS.2011.12.
37. Ekanayake J, Pallickara S, Fox G. MapReduce for Data Intensive Scientific Analyses. *IEEE Fourth Int. Conference on eScience*, eScience '08, 2008; 277 – 284, doi:10.1109/eScience.2008.59.
38. Dobre C, Xhafa F. Parallel Programming Paradigms and Frameworks in Big Data Era. *Int. Journal of Parallel Programming* 2014; **42**(5):710–738, doi:10.1007/s10766-013-0272-7.
39. Tudoran R, Costan A, Antoniu G. MapIterativeReduce: A Framework for Reduction-intensive Data Processing on Azure Clouds. *Proceedings of 3rd Int. Workshop on MapReduce and Its Applications Date*, MapReduce '12, ACM: New York, NY, USA, 2012; 9–16, doi:10.1145/2287016.2287019.
40. Hirofuchi T, Lèbre A. Adding Virtual Machine Abstractions Into SimGrid: A First Step Toward the Simulation of Infrastructure-as-a-Service Concerns. *Cloud and Green Computing (CGC), 2013 Third International Conference on*, 2013; 175–180, doi:10.1109/CGC.2013.33.
41. Javadi B, Kondo D, Vincent JM, Anderson DP. Discovering Statistical Models of Availability in Large Distributed Systems: An Empirical Study of SETI@home. *IEEE Transactions on Parallel and Distributed Systems* 2011; **99**(PrePrints), doi:10.1109/TPDS.2011.50.
42. Huang S, Huang J, Dai J, Xie T, Huang B. The HiBench benchmark suite: Characterization of the MapReduce-based data analysis. *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on*, 2010; 41–51, doi:10.1109/ICDEW.2010.5452747.
43. Makridakis S. Time-Series Analysis and Forecasting: An Update and Evaluation. *International Statistical Review / Revue Internationale de Statistique* 1978; **46**(3):pp. 255–278. URL http://www.jstor.org/stable/1402374.
44. Jain R. *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling*. Wiley professional computing, Wiley, 1991.