

Extending Science Gateway Frameworks to Support Big Data Applications in the Cloud

Shashank Gugnani · Carlos Blanco · Tamas Kiss · Gabor Terstyanszky

Received: 26 January 2016 / Accepted: 31 May 2016 / Published online: 13 June 2016
© The Author(s) 2016. This article is published with open access at Springerlink.com

Abstract Cloud computing offers massive scalability and elasticity required by many scientific and commercial applications. Combining the computational and data handling capabilities of clouds with parallel processing also has the potential to tackle Big Data problems efficiently. Science gateway frameworks and workflow systems enable application developers to implement complex applications and make these available for end-users via simple graphical user interfaces. The integration of such frameworks with Big Data processing tools on the cloud opens new opportunities for application developers. This paper investigates how workflow systems and science gateways can be extended with Big Data processing capabilities. A generic approach based on infrastructure aware workflows is suggested and a proof of concept is implemented based on the WS-PGRADE/gUSE science gateway framework and its integration with the Hadoop parallel data processing solution based on the MapReduce paradigm in the cloud. The provided analysis demonstrates that the methods described to

integrate Big Data processing with workflows and science gateways work well in different cloud infrastructures and application scenarios, and can be used to create massively parallel applications for scientific analysis of Big Data.

Keywords Big data · Hadoop · MapReduce · Science gateway · WS-PGRADE · Workflow

1 Introduction

Cloud Computing is a new and emerging computing paradigm that has the potential to completely change the way how commercial and scientific applications are deployed, hosted and executed. By using virtualization technology, Cloud Computing offers scalable, reliable and flexible resources and services.

Although computational power is increasing with time, the requirement to process the ever increasing amount of data is more and more challenging. Traditional sequential data processing algorithms are not good enough to analyze this large volume of data. Hence, new parallel approaches and algorithms are constantly being proposed. One of such examples is Apache Hadoop [1], an open-source implementation of the MapReduce framework [2] introduced by Google in 2004. It allows users to store and process large amounts of data using a distributed cluster environment. In the last few years, Hadoop in

S. Gugnani · C. Blanco · T. Kiss (✉) · G. Terstyanszky
Center for Parallel Computing, University of Westminster,
London, UK
e-mail: T.Kiss@westminster.ac.uk

C. Blanco
University of Cantabria, Cantabria, Spain

the cloud has become an extremely popular system for analyzing Big Data. Many scientific applications (such as weather forecasting [3], DNA sequencing [4], molecular dynamics [5], etc.) have now been parallelized using Hadoop and the MapReduce framework. However, the installation and configuration of a Hadoop cluster are well beyond the capabilities of domain scientists, raising significant barriers for the wider take-up of such technologies in scientific research.

Scientific workflow systems such as Taverna [18], Triana [19] or Kepler [6], provide a convenient way to represent and develop complex applications composed of multiple steps and executables. These workflows can be used for large scale experimentation and can be run on distributed computing infrastructures to provide fast and efficient program execution. Workflow systems are also frequently combined with science gateway frameworks [28], such as WS-PGRADE/gUSE [7], to provide a user friendly execution environment for domain scientist end-users. The integration of Big Data processing frameworks and solutions, such as MapReduce and Hadoop, to science gateways and workflows could take the burden of setting-up and managing these computation environments from domain scientists and facilitate the wider take-up of such technologies.

This paper describes a generic approach based on infrastructure aware workflows [22] when integrating workflow-based science gateway frameworks with Big Data processing. It also evaluates a concrete implementation of this generic approach when integrating Hadoop with the WS-PGRADE/gUSE gateway and workflow solution. The paper significantly extends earlier work of the authors [26] by providing a far more generic and flexible solution. While [26] was tightly coupled with a particular implementation of OpenStack, the current solution successfully demonstrates its applicability on various open source and commercial clouds.

2 Related Work

Scientific workflows are an efficient way of representing applications. Applications developed with Hadoop generally involve chaining MapReduce jobs and complex relationships between jobs which can be easily

represented with workflows. There have been several attempts to integrate Hadoop with scientific workflows to provide an easy base to execute Hadoop jobs in applications.

One such attempt is demonstrated by Wang J. et al. [8]. They present an approach to integrate Hadoop with the Kepler workflow system. They do so by using sub-workflows to define Mapper and Reducer functions in the MapReduce framework to define a Hadoop job. This allows users to create Hadoop jobs with ease via a GUI.

Another example is by Fei X. et al. [9] where they propose a new framework with XML based scientific workflow language called WSL. They create constructs to incorporate Map and Reduce functions in the workflow specification language. Their workflow composition framework is unique in that workflows are the only operands for composition.

A paper by Phuong Nguyen and Milton Halem [10] presents a MapReduce based workflow system for executing data intensive applications. The system consists of a C++ API for workflow design, a job scheduler and a runtime support for Hadoop. They show performance improvements by executing a climate satellite data analysis application using the new workflow system.

Oozie [11] is a workflow scheduler for Hadoop developed by Yahoo that has recently come to light. It uses directed acyclic graphs to define workflows. It is integrated with the Hadoop stack supporting several types of Hadoop jobs out of the box as well as system specific jobs (such as Java programs and shell scripts).

Chen Q. et al. [12] present a MapReduce enabled workflow system for Geographical Systems called MRGIS. The system consists of a design interface, a job scheduler and a runtime support system. The scheduler analyzes data dependencies and accordingly dispatches MapReduce jobs to Hadoop clusters. The user has the option of designing workflows with either a GUI-based designer or a python API.

As the above summary shows, there have been several efforts previously to execute Hadoop/MapReduce jobs in the cloud and to combine these with computational workflows. However, all these approaches were either suggesting new workflow systems or were tightly coupled to an existing workflow solution. The integration approach suggested in this paper allows existing and well established workflow and science

gateway frameworks to be extended with Big Data processing capabilities in general.

3 Proposed Generic Approach

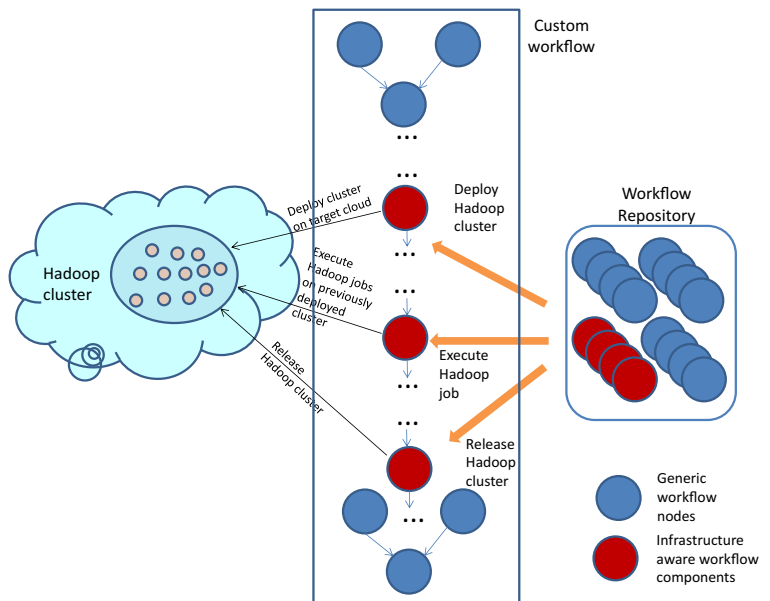
In order to extend a workflow system with Big Data processing capabilities in the cloud, three major tasks need to be performed. First, the specific processing environment, for example a Hadoop cluster, needs to be set up and configured in the cloud. This set-up needs to be part of the workflow and needs to be automatic and easily parameterized. Second, the data processing tasks, in our case the Hadoop jobs, need to be run in workflow nodes utilizing the virtual infrastructure set up in the previous step. In other words, the input and output files of data processing jobs have to be mapped into the inputs and outputs of a workflow node. Finally, the resources of the Big Data processing infrastructure, in this case the Hadoop cluster, need to be released in the cloud. This step again should be automated and part of the workflow.

In general, we can call the above concept as an infrastructure aware workflow, as introduced in [22]. Infrastructure aware workflows consist of the above described three steps (deploy, execute, release) and can be implemented as one or as multiple workflow nodes, as it will be demonstrated later in this paper.

When compared to the work in [22], this paper does not utilize the infrastructure aware workflow concept to address workflow interoperability, but it applies the same concept to integrate applications requiring specific external execution environments to workflow systems. The method is generic in the sense that once components of the infrastructure aware workflow are implemented as applications, these components can be easily embedded and called from any workflow system.

Utilisation of the infrastructure aware workflow concept, combined with a workflow repository, for the integration of big data processing capabilities to workflow systems is illustrated in Fig. 1. In order to utilize the infrastructure aware workflow, the three basic operations, deploy, execute and release, need to be implemented as workflows in the targeted workflow system. Once these workflows are available, they can be published in a generic workflow repository and made available for workflow developers. When building a custom workflow, these infrastructure aware workflows can be retrieved from the repository, parameterized, and embedded into the custom workflow.

Fig. 1 Generic Integration Concept based on Infrastructure Aware Workflows



batch job), making the approach easily extendable to various workflow systems.

4 Background

The investigation and results presented in this paper are focused around the extension of science gateway frameworks and grid/cloud workflow systems with Big Data handling and MapReduce based parallelism. As implementation environment, the WS-PGRADE/gUSE science gateway framework and some of its related and extending technologies have been applied. This section briefly describes these base-line technologies. This set of technologies enables the execution and sharing of scientific workflows in a cloud computing environment providing the basis for the cloud-based Big Data integration.

4.1 WS-PGRADE/gUSE

gUSE (Grid and Cloud User Support Environment) [7] is an open source scientific gateway framework providing users with easy access to cloud and grid infrastructures. gUSE provides with WS-PGRADE, a Liferay based portal to create and execute scientific workflows in various Distributed Computing Infrastructures (DCIs) including clusters, grids and clouds.

WS-PGRADE offers a user friendly interface for easy creation and execution of workflows. It has a graph editor which can be used to build workflows and specify job configurations. Application developers can create complex applications by using this workflow editor and upload these applications to internal and external repositories. For domain scientists, WS-PGRADE gives full access to the parameterization and execution of applications downloaded from repositories. Thus, end-users can import applications available in these repositories, configure them with their own input files/parameters and run them in the infrastructure of their choice.

WS-PGRADE supports parameter sweep applications [23]. Parameter sweep applications are scientific simulations where the same simulation needs to be executed for multiple input data sets. This feature enables the same workflow to be submitted with multiple input data sets simultaneously.

gUSE utilizes the DCI-Bridge for submitting jobs to different DCIs using the OGSA Basic Execution

Service 1.0 (BES) interface [24]. The DCI-Bridge enables nodes of a complex workflow to be executed in different DCIs providing access to a large variety of resources. In addition, two sets of APIs are provided for remote access [25]. The Application Specific Module (ASM) API allows developers to create a portal interface for a specific scientific application, while the Remote API enables developers to integrate gUSE functionality into existing graphical user interfaces.

4.2 CloudBroker Platform

The CloudBroker Platform [13] is an easy-to-use platform and application store primarily for high performance computing in the cloud, developed and provided by CloudBroker GmbH. The user can simply use a web browser or one of the available programming interfaces to select from different scientific and technical applications and immediately execute them on one of the available cloud infrastructures, using various payment options. In the platform marketplace, the user can also offer his/her own software application in the cloud, as well as access to institutional private cloud compute and storage infrastructure resources. The CloudBroker Platform can interface with various cloud middleware and infrastructure including both commercial (e.g. Amazon or CloudSigma) and open source (e.g. OpenStack or OpenNebula) cloud solutions.

The CloudBroker Platform has been integrated with gUSE/WS-PGRADE framework via an API provided by CloudBroker [27]. This allows a WS-PGRADE workflow node to execute a job through the CloudBroker Platform using applications already deployed in the platform. A generic wrapper is also available facilitating the execution of any job in the cloud without requiring pre-deployment on the CloudBroker Platform.

4.3 SHIWA Workflow Repository

The SHIWA Workflow Repository [15] is an online repository that stores and manages workflow applications. Workflow developers can upload their workflows to the repository and domain researchers can browse and search the repository to find and download workflows they want to run. In addition, the repository is integrated with WS-PGRADE/gUSE so that users

can download and execute workflows from the portal with ease.

5 Design and Implementation

5.1 Implementation Architecture

Based on the generic concept described in Section 3, the solution was implemented using the WS-PGRADE/gUSE gateway framework and workflow system integrated with the CloudBroker Platform to submit jobs to academic and commercial cloud resources. As CloudBroker has several cloud adapters, the implemented solution is able to utilize a large variety of clouds (e.g. CloudSigma, OpenStack and OpenNebula clouds are shown on Fig. 2) and not bound to one specific cloud middleware. Figure 2 shows the overall architecture of the implementation and also illustrates how an end-user can interact with WS-PGRADE and CloudBroker to run Hadoop jobs on different types of clouds.

From the developers' point of view, three major tasks need to be performed: CloudBroker deployment, workflow development on WS-PGRADE, and

workflow publication in the SHIWA Workflow Repository. Before actually running an application on the gUSE portal through the CloudBroker Platform, the application must first be deployed on the platform. This includes deploying the required software to a virtual instance and saving it as a snapshot to be used by the application later. Hence, the Hadoop application was first deployed in the platform and Hadoop installed in a virtual instance of choice. Next, the actual infrastructure aware WS-PGRADE workflows, which will be described in full detail in Sections 5.2 and 6, needed to be developed and deployed in the SHIWA Workflow Repository. Once these workflows are available, users can import and incorporate them in their workflow applications.

From the users' point of view the system works as follows. A typical end-user accesses the WS-PGRADE portal from his/her computer and creates a workflow to execute a Hadoop job. The Hadoop related workflow components are already published in and can be directly downloaded from the SHIWA Workflow Repository significantly simplifying and speeding up the workflow development process. The user is also provided with a catalog of all available clouds, regions and flavors to classify nodes easily

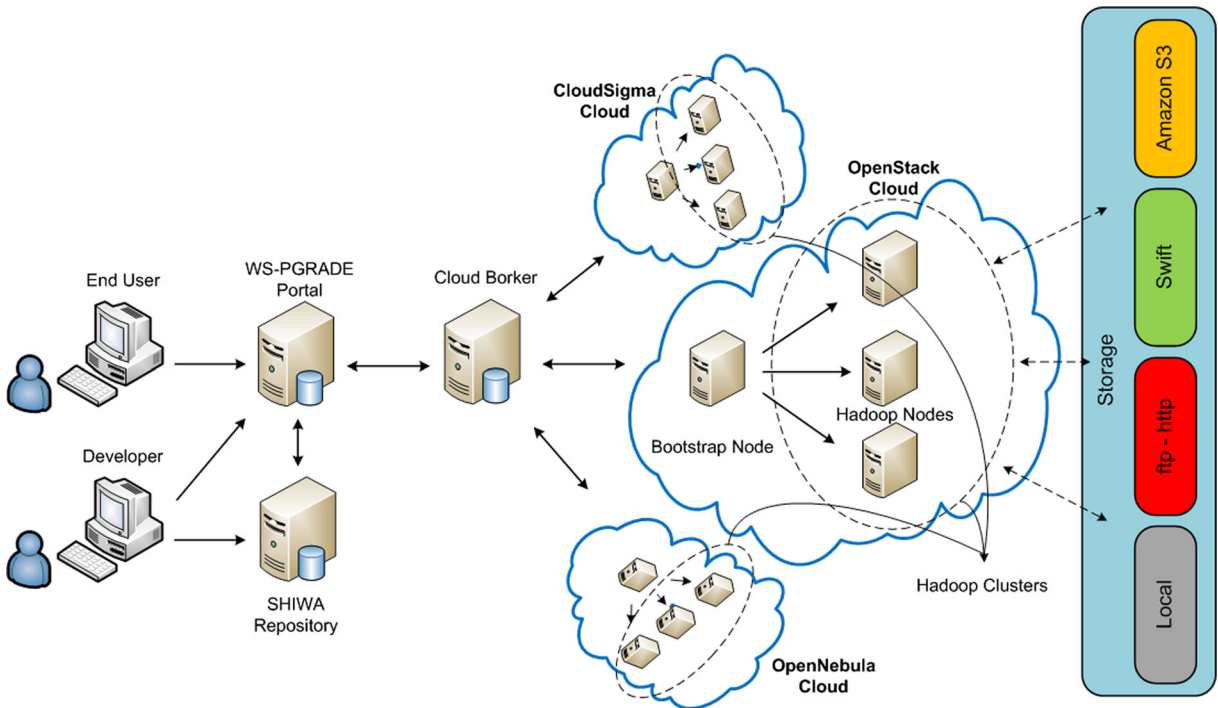


Fig. 2 Implementation Architecture

(e.g. Small, Medium, and Large). Thus, besides the number of nodes of the Hadoop cluster, the user can also select other features such as the number of cores or the RAM memory that the application requires.

Support for four different storage sources was added to allow flexibility in specifying the input and output data and allowing to manage large sets of files. The storage sources supported are local storage (on users' local machine), FTP and SFTP storage, OpenStack swift object storage, and Amazon S3 storage. FTP/SFTP, Swift and S3 were added to allow large input and output files to be transferred with ease. The problem with local storage is that files are transferred multiple times generating a bottleneck in case of large file-sizes. For example, when uploading local input files, the files are first copied from the WSPGRADE portal to the CloudBroker Platform, then to the bootstrap node, and finally to HDFS (Hadoop Distributed File System), resulting in three file transfers. However, with OpenStack swift, FTP/SFTP and Amazon S3 the files are copied directly from the source to HDFS. Additionally, the files are transferred using Hadoop's distributed copy application where a MapReduce program is launched for transferring the files, (one map task is run for each input file) thus making it even faster. The replication factor used is the same as the number of nodes, and it is specified at file creation time. The storage plugins were implemented in a transparently interoperable way, i.e. input files of a workflow could come from any data-source and output files could also go to any destination. The user can specify these input and output data sources in the job configuration file.

In order to optimize Hadoop cluster node deployment, if a submitted job finds available running instances then these instances will be used rather than creating new ones. If there are no instances available or the number of them is not enough according to job configuration, the system will launch as many instances as needed. When the job finishes, all running instances can continue running or can be destroyed by the job. If they are not destroyed then these instances can be recycled by future jobs reducing significantly the start-up time. Moreover, as the number of nodes increases, the time needed to set up and destroy these nodes also increases linearly. To address this issue, the process of starting and destroying instances is concurrently managed by a pool of python threads which interact directly with the CloudBroker Platform. Therefore, the job execution time is independent from the Hadoop cluster size. It only depends on the availability of instances in each cloud.

5.2 Methods for implementing the architecture aware workflows

Two methods have been implemented to realize the infrastructure aware workflows: Single Node Method and Three Node Method.

The Single Node Method uses a single node workflow to deploy the Hadoop environment, execute the application and destroy the cluster. Figure 3 shows the working of this method. A python script was written which launches a bootstrap node so that it will create all cluster nodes by using the CloudBroker API. If there are running instances available it only gets their IP addresses to connect to them. Then, the bootstrap

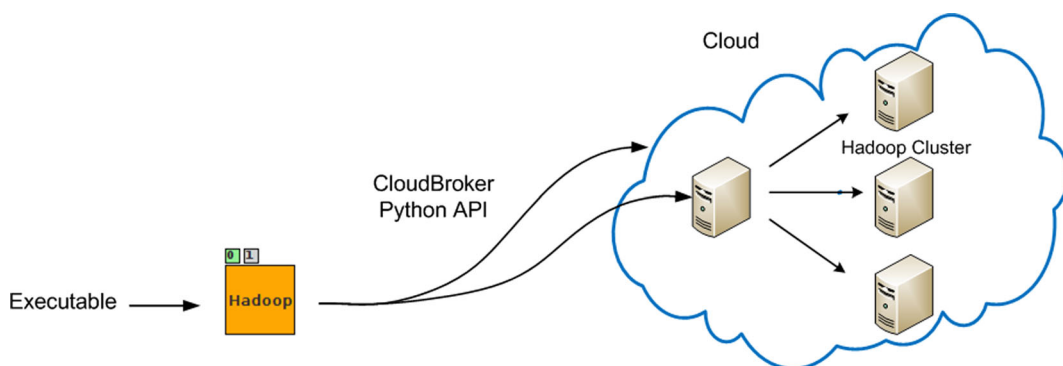


Fig. 3 Single Node Method

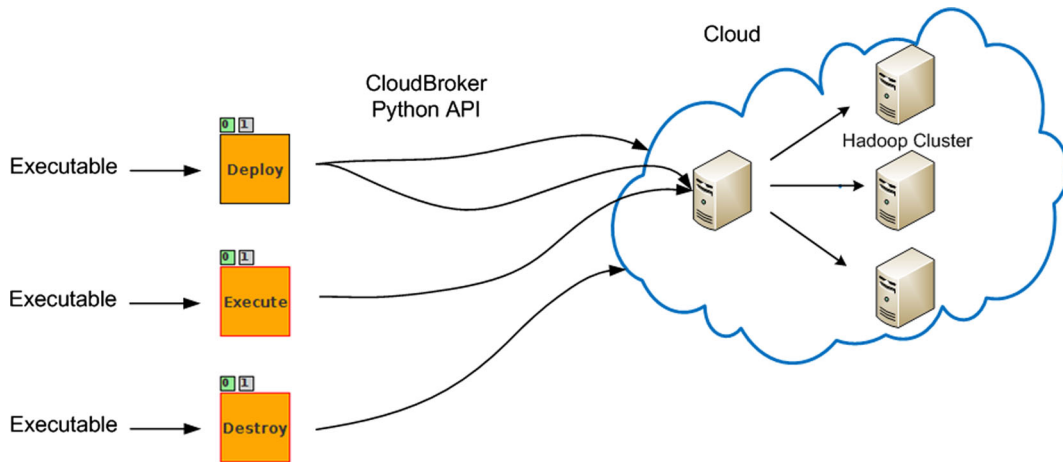


Fig. 4 Three Node Method

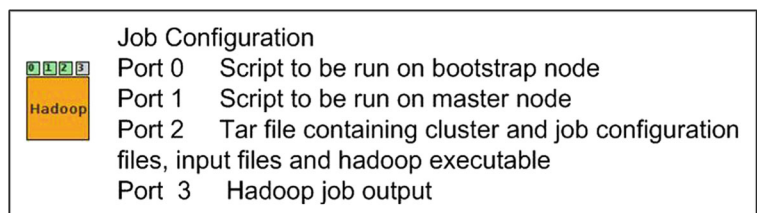
node sets up the cluster, executes the job, gets the result back and destroys all cluster nodes. If it is indicated in the cluster configuration file, instances will also be destroyed.

The Three Node Method works similarly to the Single Node Method but divides the task into three stages. The first stage creates the Hadoop cluster, the second executes the Hadoop job and the third destroys the cluster. Each stage can be considered as a workflow node executing a particular task. Figure 4 shows the working of this method. For the Three Node Method, the python script is divided into three parts to support the three node types. For each node of the workflow submitting jobs to the CloudBroker Platform, only one instance will be launched in the cloud that will be reused by each job. This instance is considered as a bootstrap instance from which the cluster nodes are created, set and destroyed. The deploy node, as it is illustrated on Fig. 6, receives an input tar file that contains cluster configuration data (port 1), and also the script that needs to be executed on the bootstrap node (port 0). The deploy node launches virtual machines, sets up the Hadoop cluster and creates two output files, a python pickle file which contains all information

about the cluster (port 3), and a cluster node IP list file (port 2). The execute node uses the cluster node IP list file to get the IP address of the master node, and transfers the input data (if the input data source is local), the job executable and other required files to the master node. It then launches a script on the master node which transfers input files to HDFS, starts the Hadoop job and then copies the result folder from HDFS to the appropriate output source (as specified by the user). The destroy node uses the python pickle file and destroys all instances.

The main idea behind dividing the complete process into three stages was to decouple the set-up and release of the Hadoop cluster from executing the Hadoop jobs. With this decoupled structure, users can keep and reuse deployed Hadoop clusters for multiple Hadoop jobs. As an additional advantage users can place these three nodes anywhere in the workflow, as long as the Deploy Hadoop Cluster is placed before any Execute Job and the Destroy Hadoop Cluster is placed after all Execute Hadoop Jobs. This allows the user complete freedom and control of the Hadoop cluster so that it can be used according to his/her application and workflow.

Fig. 5 Single Node Method - Node Configuration



Please note that in both methods only a single bootstrap node in one of the target clouds is enough to create several Hadoop clusters in any of the connected clouds. The user has to configure the number of clusters and their desired destination. This flexibility also permits deploying a Hadoop cluster composed of nodes from several clouds supposing that each cluster node has a public IP address. Therefore, the user can simply indicate the number of nodes per cloud and the bootstrap node will make arrangements for assembling and integrating all nodes into one cluster.

6 Results and Evaluation




The experimental testbed consisted of the CloudSME production WS-PGRADE/gUSE gateway (set up for the CloudSME, Cloud-based Simulation platform for Manufacturing and Engineering, European project) [14] configured to submit jobs via the CloudBroker Platform. An application was deployed in the CloudBroker Platform which installed Hadoop 2.7.1 on an Ubuntu 14.04 trusty server and saved the instance as a snapshot to be used for submitting jobs and for launching nodes in the Hadoop cluster. For testing and benchmarking purposes, jobs were submitted to two target clouds: the University of Zaragoza BIFI OpenStack [20] cloud, and the commercial CloudSigma [21] cloud. The goal of the following evaluation is to test and compare the performance of the two designed

methods and various storage solutions on different cloud infrastructures.

For testing both Single Node and Three Node methods, numerous experiments were conducted to compare the methods as well as to demonstrate their ability to handle large scale and diverse applications. Figures 5 and 6 show the node workflow configurations of both methods. For each experiment, the base case used was the same, and different parameters were varied to see the effect they had on the execution time of the workflow. The base case was executing the Hadoop Wordcount example with negligible input data size on a 2 node Hadoop cluster using local input data source and running instances through the CloudBroker Platform. The parameters that were varied were: number of nodes in Hadoop cluster, Hadoop application executed, number of sequential Hadoop jobs to be run, input data source, type of the instance and whether a running instance is used or not. Launching instances in both clouds and transferring data files are tasks where the execution time may vary at different times. To ensure that this did not affect our observations, we ran each test three times and took the average time of the three runs as the final execution time of that test.

Figure 7a shows the variation of the workflow execution time in relation to the number of nodes (task-trackers) in the cluster for both methods and clouds. It can be observed that the execution time varies linearly with the number of nodes in case of both methods and clouds, and the results are almost always better

Fig. 6 Three Node Method - Node Configuration

	Job Configuration		
	Port 0	Script to be run on bootstrap node	
	Port 1	Tar file containing cluster configuration file	
	Port 2	Cluster node IP list file	
		Port 3	Pickle configuration file
<hr/>			
	Job Configuration		
	Port 0	Script to be run on bootstrap node	
	Port 1	Tar file containing job configuration file, input files and hadoop executable	
	Port 2	Cluster node IP list file (channel)	
	Port 3	Hadoop job output	
		Port 4	Job status
<hr/>			
	Job Configuration		
	Port 0	Script to be run on bootstrap node	
	Port 1	Tar file containing cluster configuration file	
	Port 2	Pickle configuration file (channel)	
		Port 3	Job status (channel)

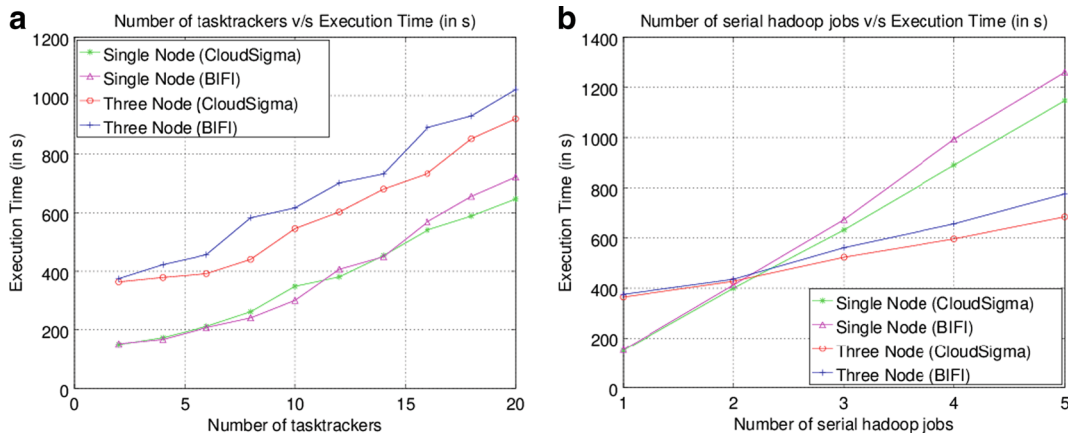


Fig. 7 a. Number of task-trackers v/s Execution Time, b. Number of serial Hadoop jobs v/s Execution time

on CloudSigma. In addition, the Single Node Method works better than the Three Node Method for executing a Hadoop job irrespective of the cluster size and the cloud type. The difference between these methods was expected as running the Three Node Method involves executing two more workflow nodes than in case of the Single Node Method, resulting in more execution overhead.

Figure 7b shows the variation of the workflow execution time with the number of sequential Hadoop jobs run for both methods and clouds. For the Single Node Method, the same node was placed in a sequence one after the other. For the Three Node Method, the Deploy Hadoop Cluster was placed before all Execute Hadoop Jobs, executed one after the other, and then the Destroy Hadoop Cluster was placed after the last Execute Hadoop Job. For both methods the same job

with the same input files was executed multiple times. It can be seen that the Single Node Method works better for executing a single Hadoop job, whereas the Three Node Method works better for executing three or more Hadoop jobs in sequence in both clouds. Interestingly, when executing exactly two Hadoop jobs then all methods and clouds performed almost identically. For the Single Node Method, for each Hadoop job, the method launches a new Hadoop cluster and then destroys the cluster, but not the running instances which are recycled by all jobs. On the other hand, the Three Node Method just creates the Hadoop cluster once and then executes all Hadoop jobs in sequence on the same cluster. Therefore, the Three Node Method works better for executing larger number of Hadoop jobs in sequence. This pattern is repeated in both CloudSigma and BIFI clouds.

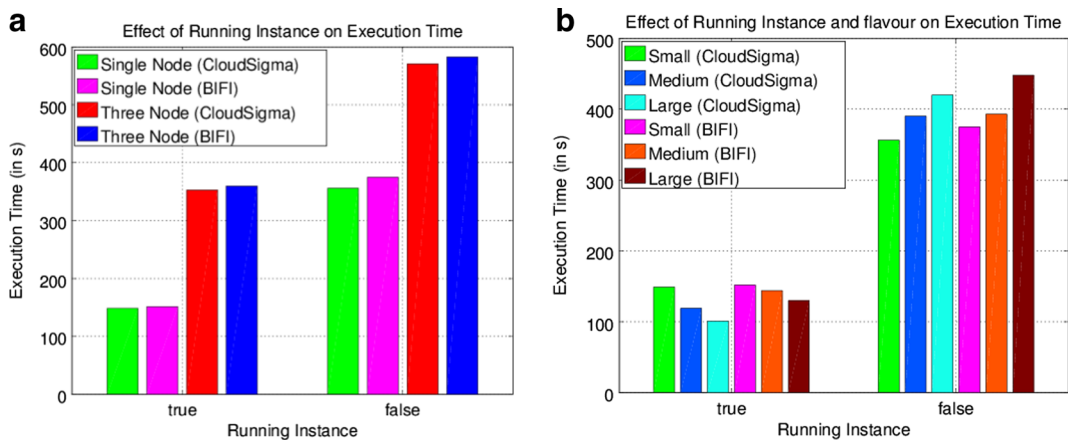


Fig. 8 a. Effect of Running Instance on Execution Time, b. Effect of Running Instance and flavor on Execution

Figure 8a shows the variation of the workflow execution time depending on whether a running instance is used or not, for both methods and clouds. Using a running instance greatly reduces the execution time, especially on the Single Node Method, which reduction is more than double the time. This reduction in time is as expected because of running instance use. There is no need to launch a new instance when a job is submitted. Additionally, this is independent from the type of cloud the tests were run on.

Figure 8b shows the combined effect of whether a running instance is used or not and also the flavor of the instance in both clouds. For this test only the Single Node Method was run. As it is shown in Fig. 8b, if running instances are used then the more powerful the instance gets, the shorter the execution time is, as expected. However, if running instances are not used then execution time actually increases with introducing more powerful instances. This can be explained as although the instance is becoming more powerful and it takes less time to execute jobs, it also takes more time to create a larger instance in the targeted clouds. Please note that instance set-up times and patterns are hugely cloud dependent and results can vary in case of different cloud providers. Therefore, in case of not using running instances, there should be careful consideration of instance types and job execution times before selecting the right flavor.

Figure 9a shows the workflow execution time for three different Hadoop applications using the Single Node Method. The applications used were: WordCount - the standard example that comes with Hadoop, Rule Based Classification [17]- a classification

algorithm adapted for MapReduce, and Prefix Span [16] - MapReduce version of the popular sequential pattern mining algorithm. This experiment was executed to show that the method developed is generic and could be used for a wide range of Hadoop applications.

Figure 9b illustrates the variation of the workflow execution time depending on the input source used. This is an important feature because Hadoop was designed to manage Big Data. For this experiment a dataset composed of 40 files of 1GB each was used. No Hadoop job was run in these experiments. The execution time is the sum of the time for setting up the Hadoop cluster, transferring files to HDFS and destroying the cluster. Because of limitations on local data size exposed by the CloudSME WS-PGRADE portal (max 512 MB local input data is allowed), only FTP, swift and S3 were selected to test. These remote storages were the same for the two clouds. When comparing the results, it can be seen that the performance of FTP storage is the poorest. This can be explained with the legacy nature of this storage protocol that typically demonstrates lower performance. Using swift object storage is much better than using FTP storage, and using S3 storage is the fastest. Therefore, it is recommended to use the S3 or swift storage over FTP for large datasets. Additionally, there is a noticeable difference between the two clouds with CloudSigma significantly outperforming the BIFI OpenStack cloud.

Our final experiment was executing multiple Hadoop jobs simultaneously using the *Parameter Sweep* feature of WS-PGRADE. For this experiment,

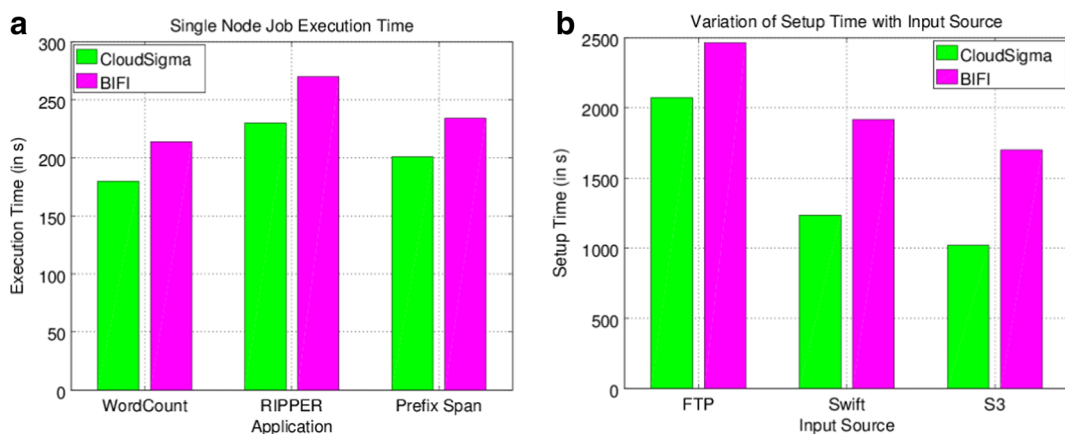


Fig. 9 a. Single Node Job Execution of Different Applications b. Variation of Setup Time Depending on Input Source

Fig. 10 ParameterSweep Job Execution Time and Speed-up



	Cloud Sigma	Bifi
Word Count	3.06	3.07
Rule Based	2.92	3.17
Prefix Span	3.09	3.12

five simultaneous Hadoop jobs were submitted, each with a two node cluster and different input data. The three different applications used for this experiment were the same as used in Fig. 9a. The objective was to verify the convenience and measure the speedup of *Parameter Sweep* jobs where the same application should be executed with different input sets. Figure 10 shows the execution time of the workflows for different Hadoop applications. The execution time is calculated as the time between the submission of the jobs and the time the final job is completed. When comparing the execution time of experiments in Figs. 9a and 10, it can be seen that in case of *Parameter Sweep* Jobs, the execution time is longer, not achieving the ideal speed-up. This is explained by the fact that both cloud infrastructures (OpenStack and CloudSigma) take more time to launch instances when a large number of simultaneous launch requests are submitted. Despite this, the speed up when using the *Parameter Sweep* execution in both clouds is still quite significant, around or exceeding three for all applications.

7 Conclusion and Future Work

This paper presented a generic method based on infrastructure aware workflows, and a concrete implementation of this concept to integrate Big Data processing based on the MapReduce paradigm and Hadoop to scientific workflow systems. A number of experiments were run to test and evaluate the implemented solution on different cloud infrastructures. Although the implementation was based on the WS-PGRADE/gUSE gateway framework and the

CloudBroker Platform, the infrastructure aware workflow concept is generic and can be extended for any gateway and workflow environment and cloud.

By integrating Hadoop with WS-PGRADE, the user can configure one or more workflow nodes to execute Hadoop jobs and can also define the type of Hadoop cluster required based on job requirements. The user can then use this system to create complex applications for large scale scientific simulations and can also utilize the *Parameter Sweep* feature of WS-PGRADE to run Hadoop jobs with multiple input datasets simultaneously.

As demonstrated by the experiments, the solution works for different Hadoop applications. Experiments also demonstrated that the two developed methods work well in different scenarios. The Single Node Method is better for executing a single job whereas the Three Node Method works better for multiple Hadoop jobs. Although there is an overhead when deploying and destroying a Hadoop cluster even when using running instances, this overhead (around seconds) is negligible when compared to hours of CPU time to execute large Hadoop applications.

As future work, further scientific applications will be identified that can utilize the methods developed in this paper, and workflow implementations of these applications will be created. We also plan extending the implementation described in this paper to support more clouds and gateway frameworks. Specifically, this includes extension towards the EGI (European Grid Infrastructure) Federated Cloud that would enable large user communities to utilize the solution. A specific area of further investigation is to be concerned with transferring input and output data to and from the HDFS in the cloud. Although some

measurements regarding the correlation between overall performance and the location of input data have been presented in this paper, in case of exceptionally large datasets it can still cause problems, even when utilizing Hadoop's distributed copy mechanism. Future work needs to investigate how this file transfer scales up and how other options, e.g. streaming data during execution time, could affect performance positively. Finally, further experiments with the Three Node Method are also planned to identify when it is the best to deploy and destroy the cluster and achieve optimal performance.

Acknowledgments This work is partially funded by the CloudSME Cloud-Based Simulation platform for Manufacturing and Engineering Project No. 608886 (FP7-2013-NMP-ICT-FOF). Financial support from Programa de Personal Investigador en Formación Predoctoral from Universidad de Cantabria, co-funded by the regional government of Cantabria, has also been utilized.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Apache Hadoop. <http://hadoop.apache.org/>. [26 November 2015]
2. Dean, J., MapReduce, G.S.: Simplified data processing on large clusters. *Commun. ACM* **51**(1), 107–113 (2008). doi:[10.1145/1327452.1327492](https://doi.org/10.1145/1327452.1327492)
3. Li, L., Ma, Z., Liu, L., Fan, Y.: Hadoop-based ARIMA algorithm and its application in weather forecast. *Int. J. Database Theory Appl.* **6**(5), 119–132 (2013). doi:[10.14257/ijta.2013.6.5.11](https://doi.org/10.14257/ijta.2013.6.5.11)
4. Schatz, M.C.: Cloudburst: highly sensitive read mapping with mapreduce. *Bioinformatics* **25**(11), 1363–1369 (2009). doi:[10.1093/bioinformatics/btp236](https://doi.org/10.1093/bioinformatics/btp236)
5. Jiao, S., He, C., Dou, Y., Tang, H.: Molecular dynamics simulation: Implementation and optimization based on Hadoop. 2012 Eighth International Conference on Natural Computation (ICNC), 2012; 1203–1207. doi:[10.1109/ICNC.2012.6234529](https://doi.org/10.1109/ICNC.2012.6234529)
6. Ludascher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., Tao, J., Zhao, Y.: Scientific workflow management and the Kepler system. *Concurr. Comput. Pract. Exper.* **18**(10), 1039–1065 (2006). doi:[10.1002/cpe.994](https://doi.org/10.1002/cpe.994)
7. Kacsuk, P.: P-GRADE portal family for grid infrastructures. *Concurr. Comput. Pract. Exper.* **23**(3), 235–245 (2011). doi:[10.1002/cpe.1654](https://doi.org/10.1002/cpe.1654)
8. Wang J., Crawl D., Altintas I.: Kepler + Hadoop: A general architecture facilitating data-intensive applications in scientific workflow systems. In: Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science, WORKS '09, pp. 12:1–12:8. ACM, NY, USA (2009). doi:[10.1145/1645164.1645176](https://doi.org/10.1145/1645164.1645176)
9. Fei, X., Lu, S., Lin, C.: Mapreduce-enabled scientific workflow composition framework. *IEEE Int. Conf. Web Services, 2009. ICWS 2009*, 663–670 (2009). doi:[10.1109/ICWS.2009.90](https://doi.org/10.1109/ICWS.2009.90)
10. Nguyen P., Halem M.: A MapReduce Workflow System for Architecting Scientific Data Intensive Applications. In: Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing, SELOUD'11, pp. 577–63. ACM, NY, USA (2011). doi:[10.1145/1985500.1985510](https://doi.org/10.1145/1985500.1985510)
11. Oozie. <http://oozie.apache.org/>. [26 November 2015]
12. Chen, Q., Wang, L., Shang, Z.: MRGIS: A mapreduce-enabled high performance workflow system for GIS. In: Proceedings of the 2008 Fourth IEEE International Conference on eScience, ESCIENCE'08, pp. 646?–651. IEEE Computer Society, DC, USA (2008). doi:[10.1109/eScience.2008.169](https://doi.org/10.1109/eScience.2008.169)
13. Cloudbroker platform. <http://cloudbroker.com/platform/>. [26 November 2015]
14. Taylor, S.J.E., Kiss, T., Terstyanszky, G., Kacsuk, P., Fantini, N.: Cloud computing for simulation in manufacturing and engineering: Introducing the CloudSME simulation platform. In: Proceedings of the 2014 Annual Simulation Symposium, ANSS '14, pp. 12:1–12:8. Society for Computer Simulation International, CA, USA (2014)
15. SHIWA Workflow Repository. <https://shiwa-repo.cpc.wmin.ac.uk/shiwa-repo/>. [26 November 2015]
16. Prefix Span Hadoop. <https://github.com/WCMinor/prefixspanhadoop/>. [26 November 2015]
17. Gugnani, S., Khanolkar, D., Bihany, T., Khadilkar, N.: Rule based classification on a multi node scalable hadoop cluster. In: Fortino, G., Fatta, G.D., Li, W., Ochoa, S., Cuzzocrea, A., Pathan, M. (eds.) *Internet and Distributed Computing Systems*, pp. 174–183. No. 8729 in *Lecture Notes in Computer Science*, Springer International Publishing (2014). doi:[10.1007/978-3-319-11692-1_15](https://doi.org/10.1007/978-3-319-11692-1_15)
18. Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M.R., Wipat, A., et al.: Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics* **20**(17), 3045–3054 (2004). doi:[10.1093/bioinformatics/bth361](https://doi.org/10.1093/bioinformatics/bth361)
19. Churches, D., Gombas, G., Harrison, A., Maassen, J., Robinson, C., Shields, M., Taylor, I., Wang, I.: Programming scientific and distributed workflow with Triana services. *Concurr. Comput. Pract. Exper.* **18**(10), 1021–1037 (2006). doi:[10.1002/cpe.992](https://doi.org/10.1002/cpe.992)
20. Institute for Biocomputation and Physics of Complex Systems(BIFI). <http://bifi.es/>. [26 November 2015]
21. Cloudsigma. <https://www.cloudsigma.com/>. [26 November 2015]

22. Kacsuk, P., Kecskemeti, G., Kertesz, A., Nemeth, Z., Visegradi, A., Gergely, M.: Infrastructure aware scientific workflows and their support by a science gateway. In: 2015 7th International Workshop on Science Gateways (IWSG), pp. 22–27 (2015). doi:[10.1109/IWSG.2015.14](https://doi.org/10.1109/IWSG.2015.14)
23. Kacsuk, P., Karoczkai, K., Hermann, G., Sipos, G., Kovacs, J.: WS-PGRADE: Supporting parameter sweep applications in workflows. In: Third Workshop on Workflows in Support of Large-Scale Science, 2008. WORKS 2008, pp. 10–?10 (2008). doi:[10.1109/WORKS.2008.4723955](https://doi.org/10.1109/WORKS.2008.4723955)
24. Foster, I., Grimshaw, A., Lane, P., Lee, W., Morgan, M., Newhouse, S., Pickles, S., Pulsipher, D., Smith, C., Theimer, M.: Ogsa basic execution service version 1.0 (2007)
25. Balasko, A., Farkas, Z., Kacsuk, P.: Building science gateways by utilizing the generic WS-PGRADE/gUSE workflow system. *Comput. Sci.* **14**(2), 307 (2013). doi:[10.7494/csci.2013.14.2.307](https://doi.org/10.7494/csci.2013.14.2.307)
26. Gugnani, S., Kiss, T.: Extending Scientific Workflow Systems to Support MapReduce Based Applications in the Cloud, 7th International Workshop on Science Gateways, IWSG 2015, 3-5, 2015, Budapest, Hungary, pp. 16–21, doi:[10.1109/IWSG.2015.15](https://doi.org/10.1109/IWSG.2015.15)
27. Farkas, Z., Kacsuk, P., Hajnal, A.: Connecting Workflow-Oriented Science Gateways to Mul-ti-cloud Systems, 7th International Workshop on Science Gateways, IWSG 2015, 3-5, 2015, Budapest, Hungary, pp. 40–46, doi:[10.1109/IWSG.2015.20](https://doi.org/10.1109/IWSG.2015.20)
28. Kacsuk P. (ed.): *Science Gateways for Distributed Computing Infrastructures: Development Framework and Exploitation by Scientific User Communities*, Springer, 2014. pp. 301. (ISBN:978-3-319-11267-1)