BioCloud: A Resource Provisioning Framework for Bioinformatics Applications in Multi-Cloud Environments

Izzet F. Senturk[a], Bala Krishnan[b], Anas Abu-Doleh[a,*], Kamer Kaya[a,c], Qutaibah Malluhi[b], Ümit V. Çatalyürek[a]

[a] *Dept. Biomedical Informatics, The Ohio State University, Columbus, OH, 43210*
[b] *KINDI Center for Computing Research, Qatar University, Doha, Qatar*
[c] *Faculty of Engineering and Natural Sciences, Sabancı University, Istanbul, Turkey*

Abstract

The significant advancement in Next Generation Sequencing (NGS) have enabled the generation of several gigabytes of raw data in a single sequencing run. This amount of raw data introduces new scalability challenges in processing, storing and analyzing it, which cannot be solved using a single workstation, the only resource available for the majority of biological scientists, in a reasonable amount of time. These scalability challenges can be complemented by provisioning computational and storage resources using Cloud Computing in a cost-effective manner. There are multiple cloud providers offering cloud resources as a utility within various business models, service levels and functionalities. However, the lack of standards in cloud computing leads to interoperability issues among the providers rendering the selected one unalterable. Furthermore, even a single provider offers multiple configurations to choose from. Therefore, it is essential to develop a decision making system that facilitates the selection of the suitable cloud provider and configuration together with the capability to switch among multiple providers in an efficient and transparent manner. In this paper, we propose BioCloud as a single point of entry to a multi-cloud environment for non-computer savvy bio-researchers. We discuss the architecture and components of BioCloud and present the scheduling algorithm employed in BioCloud. Experiments with different use-cases and scenarios reveal that BioCloud can decrease the workflow execution time for a given budget while encapsulating the complexity of resource management in multiple cloud providers.

Keywords: Cloud computing, cloud broker, interoperability, multi-cloud, bioinformatics

1. Introduction

The last decade have witnessed rapid advances in the field of genomics thanks to the evolution of the genome sequencing technologies which lead to accelerated generation of digital biological information in unprecedented amounts. The emergence of high-throughput NGS has revolutionized genomics research by providing an astounding cost reduction making the whole genome sequencing possible for as low as $1,000 [1] and hence making the technology pervasive. The availability of NGS on a wider scale with its decreased cost and high-throughput have paved the way for more complex NGS data at a rate outpacing the advances in computation and storage capacities [2].

Minimizing the impact of the increased data complexity requires scalable solutions for storing and analyzing massive NGS data. The scalability issues of NGS drives the efforts to cloud computing, which is converging as a frontier to address this class of problems by enabling large scale computing resources on demand, tailored to specific requirements in a pay-per-use manner [2]. Cloud computing renders maintaining large clusters unnecessary while handling peak-time loads and addressing issues such as availability, load balancing and fault tolerance.

Cloud providers tend to offer resources through their custom APIs which restrict the development of the tools with respect to the vendor specific API. In the long term, customers are restricted to the vendor and cannot migrate from one cloud provider to another seamlessly. The proposed BioCloud[1] employs a Multi-Cloud [6] model and acts as a *broker* across the resources of multiple cloud providers. Considering the vast number of hardware profiles available for selection in cloud providers, complexity of determining the hardware profiles to be used and their quantity can be overwhelming not to mention the complexity of resource provisioning and configuration. Note that there are 38 current generation "instance types" in EC2 [7] and 19 "flavors" in RackSpace [8] available to choose. Some of these hardware profiles are optimized for memory, cpu, storage, etc. BioCloud analyzes workflow

*Corresponding author
Email addresses: ifs5@cornell.edu (Izzet F. Senturk), abudoleh.1@osu.edu (Anas Abu-Doleh), kaya@sabanciuniv.edu (Kamer Kaya), qmalluhi@qu.edu.qa (Qutaibah Malluhi), umit@bmi.osu.edu (Ümit V. Çatalyürek)

[1] The term BioCloud is also used by the Beijing Institute of Genomics [3] to denote their bioinformatics cloud system. Recently, the term BioCloud has been used as a general term to indicate the cloud-based bioinformatics applications [4]. In this paper, the term BioCloud denotes our multi-cloud bioinformatics framework [5].

steps and evaluates hardware profiles in available resources while considering user requirements such as deadline, budget, etc. in order to determine the type and number of resources to be used for each of the workflow steps individually. BioCloud ensures availability of resources for workflow execution by provisioning resources in such a way that the resources are neither wasted nor additional delay occurred due to the waiting time for resource initialization (i.e., booting the resource, dynamic cluster configuration on the cloud, etc.). This requires a scheduling algorithm which considers several resource options (hardware profiles in cloud providers, possibility of configuring a cluster in the cloud and determining the number of compute nodes to be used) and the possibility of exploiting parallelism which is the main focus of this paper. Scheduling may not yield the best solution unless an accurate estimation for the running times cannot be attained. BioCloud exploits its profiler to keep the execution times of the tools on the given resources considering the size of the input and output files. This enables a means to estimate the execution time of the workflow steps. Scheduler employs resource manager to ensure availability of the resources before the workflow steps are dispatched for execution. Scheduler also evaluates the workflow steps for parallelism and modifies workflows to enable parallelism if possible. Scheduler cooperates with the workflow manager and takes care of the required manipulations on the data and tool settings.

BioCloud encapsulates all the complexity of resource management and provides a single entry point to create custom workflows and run them in a simple and efficient manner through its user-friendly web-interface. The public virtual machine (VM) image we provide [5] can be employed to start a BioCloud instance. We assume BioCloud users have an existing account in at least one of the cloud providers. In order to start using BioCloud, users create a BioCloud account through the web-interface, and complete their profiles by providing the available resources to be used. The resources can be cloud account(s), local clusters, servers, and datasets. Then a BioCloud instance is started on the cloud using the provided cloud credentials. Once the instance is initialized, leaving workflow manager interface (Galaxy [9]) is presented to the user which runs on one of the resources provided by the user. The workflows created by the user are executed over the computational resources defined earlier. If multiple computational resources are available, jobs in a multi-step workflow can be run on different resources based on the scheduling algorithm and the user requirements. BioCloud strives to exploit parallelism to reduce the overall workflow execution time by running parallel steps using different computing resources or dividing a single step into multiple parallel steps by partitioning the input data and computation, whenever possible.

BioCloud offers a loosely coupled architecture through its service oriented architecture. BioCloud Portal web-service is employed to expose some of the functionalities of the system so that some of the workflow decisions (i.e., when to dispatch a workflow step and where to run this step) are delegated to the BioCloud Portal web-service. This enables modularity where scheduling logic is separated from the core workflow system. This provides the flexibility of updating the scheduling algorithm and other features of the system (i.e., improving abstract workflows submitted by the user and presenting the new workflow for execution) without requiring a software update on the user side.

The rest of the paper is organized as follows. Section 2 compares the features of the proposed work with notable studies from the literature. Section 3 details the proposed BioCloud architecture. The proposed scheduling algorithm is discussed in Section 4. Section 5 demonstrates the features of the proposed system by evaluating BioCloud using two real-life use-cases. Finally, the concluding remarks are given in Section 6.

2. Related Work

The vast amount of data generated by NGS platforms poses a challenge to store, access and manipulate data in an efficient manner within a reasonable amount of time. A single workstation is often not sufficient to complete the analysis in a reasonable amount of time and organizations need to own and maintain specific type of hardware to handle operations in such scale. To remedy the problem, some efforts have focused on parallelizing existing tools using various distributed memory parallelism schemes, such as MPI (Message Passing Interface) [10, 11] or MapReduce [12, 13]. However, both approaches require dealing with complex software frameworks and hence require experienced developers for efficient parallelization, and also experienced users to use developed applications.

In the rest of this section, we discuss various frameworks commonly used in bioinformatics based on the underlying infrastructure they support.

2.1. Web-based Frameworks

Many frameworks, such as Grendel [14], provides a web service based architecture to access high performance computing (HPC) resources. Web services can be invoked remotely so that the functionality of the deployed tools can be exposed on the network without interoperability concerns. However, computational resources are limited with the maintained HPC resources. MG-RAST [15] is an open source platform specialized in metagenome analysis. Users can analyze their data through the offered analysis pipeline. The jobs and the data made public by the user are stored in the system indefinitely which makes MG-RAST a repository for metagenomic data.

Considering the lack of standards and complicated reproducibility in NGS experiments, various genomics research frameworks are presented such as GenePattern [16],

Mobyle [17], and Galaxy [9] within the concept of "Reproducible Research System" to support reproducible computational research. These frameworks provide a unified web-interface to access tools, form multi-step analysis pipelines, run the experiments and share analysis with others for reuse. New tools can be added by writing a tool configuration file in Galaxy, a server configuration file in Mobyle, and through the web-interface in GenePattern.

Web service based solutions encapsulate the complexity of the maintained infrastructure and tools and provide interoperability between different platforms. Workflow management systems contribute reproducibility of the experiments. Despite the convenience of the mentioned platforms, users are bounded to the limitations of the provided resources. Our BioCloud framework leverages the commonly used Galaxy framework for creating workflows, but extends that by enabling not only elasticity of the cloud, but also provides a mechanism to use multi-cloud providers together. Furthermore, it can improve submitted abstract workflows by generating a new workflow to exploit parallelism if possible.

2.2. Frameworks with Cloud Computing

Cloud computing is already embraced by many bioinformatics projects [13, 18, 19, 20]. The common approach of these frameworks is applying the MapReduce model to provide parallelism for a particular problem. CloudBurst [18] is a parallel mapping algorithm optimized to map NGS data to reference genomes. It is modeled by using a short-read mapping program to report alignments with mismatches. CloudAligner [19] is an alternative tool for mapping short reads. It also supports pair-end mapping and longer reads. Crossbow [13] focuses on human resequencing and SNP detection and unlike CloudBurst and CloudAligner, it considers billions of reads. Crossbow presents a pipeline of short-read alignment and SNP calling by combining the features of Bowtie and SOAPsnp [21] respectively. Myrna [20] provides a pipeline to calculate differential gene expression of the RNA-seq data by integrating short read alignment, interval calculations, normalization, aggregation and statistical modeling steps. Myrna requires Bowtie, R, and Bioconductor. These frameworks focus on a single problem such as sequence alignment and SNP detection. Our BioCloud framework is not limited to a single problem and can be extended as long as new tools are added to the workflow manager.

2.3. Frameworks as a Service

Bioinformatics frameworks which are presented as a virtual machine also exist [22, 23, 24, 25, 26]. They provide a means of convenience by eliminating the installation and configuration of tools while ensuring configuration consistency which facilitates reproducible research. Also, by eliminating the time required to configure the operating system with the essential tools and data, these frameworks can alleviate the time to start conducting research. We also note that virtual machine images can be run on personal computers with a virtualization software as well as in cloud with an option of forming a cluster.

CloVR [23] is pre-configured with automated sequence analysis pipelines for microbial genomics including whole genome and metagenome sequence analysis. Besides the analysis tools and pipelines, CloVR is bundled with BioLinux, job schedulers, and a workflow management system. CloudBioLinux [24] provides several bioinformatics tools including alignment, clustering, assembly, phylogenetics, etc. Tools are complemented with the provided web-based documentation which explains tool functionalities. Also, scripts are provided to access a repository of reference genomes on an Amazon S3 bucket.

Galaxy CloudMan [22] is built on top of CloudBioLinux and it provides an integrated solution using the Galaxy workflow management system. Thanks to the user-friendly Galaxy interface, it is possible to design custom workflows for various scenarios and exploit elasticity of cloud by modifying resources at run-time. Another platform, which is based on Galaxy, is presented in [26]. By integrating a tool, called Globus Provision, the platform automatically deploys and configures the tools and applications required by Galaxy. Also, Globus Transfer is integrated in order to ensure reliable, high-performance data transfer.

2.4. Federated and Multi-Cloud Brokers

The primary objective of OPTIMIS [27] toolkit is to provide a new cloud ecosystem that provisions the resources and services using multiple coexisting cloud providers in both federated and multi-cloud fashion. However, OPTIMIS requires its agents to be deployed at the cloud provider side. This is not always feasible since the deployment depends on the permission from the cloud provider. In our approach, we use *Deltacloud* [28] adapters to enable seamless communication between different cloud providers and BioCloud so that the requirement of deploying agents on the cloud provider is avoided.

In Contrail [29] project, the applications get the resources from multiple cloud providers using federation or multi-cloud manner. Also, it uses internal adapters to realize a cloud federation with multiple cloud providers running the contrail software and external adapters for multi-cloud environment with the cloud providers which do not have it. In order to support multi-cloud environment, it requires the development of external adapters which is not fully matured.

Aeolus [30] is a cloud management software that provides tools and services for the creation, management and monitoring of instances across multiple clouds. BioCloud leverages *Deltacloud* and TIM components of Aeolus.

2.5. Cloud Workflow Scheduling Algorithms

Though the workflow scheduling has the potential opportunity to utilize cloud computing, very few initiatives are made to integrate cloud environments. For example,

critical path allocation based workflow scheduling is proposed in Rahman et al. [31], Chen et al. [32] describe an QoS constraint ant colony optimization algorithm and Yu et al. explain about budget constraint scheduling on utility Grids using Genetic Algorithms in [33]. However, they do not provide optimal solutions for cloud environments. Mao and Humphrey depict a vibrant method to schedule workflow steps on clouds [34] . The proposed approach tries to minimize the execution cost by considering several VM instance types with different prices. But they do not provide a near-optimal solution. Malawski et al. [35] explicate a deadline and budget constraint scheduling algorithm which tries to maximize the amount of work completed. But, it considers a single type of VM and fails to consider the heterogeneous nature of clouds. Abrishami et al. [36] propose a scheduling solution which tries to optimize tasks in partial critical path by executing them on cloud resources. But they do not provide global optimization by considering complete workflow structure. Particle Swarm Optimization(PSO) based near-optimal scheduling algorithm is proposed by Wu et al. [37]. Though it could handle different VM types, it do not reap the benefit of cloud elasticity since it presumed that it has an initial set of VMs available ahead of time. The scheduling algorithm propose by Byun et al. [38] estimated the minimum number of resources needed to execute the workflow in a cost-effective way. Though the algorithm utilizes the elastic nature of cloud, it fails to consider the heterogeneous nature of cloud by assuming a single VM type. Maria et al. [39] propose a PSO based algorithm to minimize the workflow execution cost satisfying deadline constraints. BioCloud solves the above mentioned issues using the knowledge gained from application profiler. Firstly, it classifies each tool/software used in each workflow step into CPU,memory,storage or I/O intensive. Secondly, it gathers the knowledge about the relationship of execution time and cost with each instance type for that tool/software. During workflow execution, the class of the instance for each workflow step can be decided by the software used in that step. Next, the initial assignment of an apt instance from that class will be decided by analyzing their peak usage from previous runs. The number of counts in each instance type will be decided based on the budget and deadline constraints. The execution time adjustment can be achieved either by assigning the high capacity instance type which in turn reduces the execution time or combining the dependent tasks into groups and execute them in a single instance thereby reducing their data transfer time. The cost adjustment can be done by either upgrade or downgrade the instance type without affecting deadline constraints.

3. BioCloud System Design

BioCloud follows a service oriented architecture and consists of two main components, namely, BioCloud *Portal* and BioCloud *Workflow Manager* (BCWM). BioCloud Portal implements and encapsulates functions to orchestrate workflow execution across disparate platforms. These functions are exposed as a service to enable interoperability and flexibility across platforms. BioCloud Portal also hosts a web application to register and start using the system. BioCloud Portal can be regarded as the single access point of the BioCloud for all users (virtual organizations). BCWM, on the other hand, is the workflow management component of the system. While a unique BioCloud Portal is employed for all virtual organizations (VOs), an exclusive BCWM is created and employed for each VO. Although we provide virtual machines for both components, it is also possible to use a custom workflow manager to consume BioCloud Portal services due to the loosely-coupled architecture.

A VO is regarded as a single entity (organization) with multiple users. Once a VO is registered to BioCloud, admin of the VO can manage its own users. We note that BioCloud does not provide computing resources and it is assumed that VO has its own resources; which can be a cloud provider account, local cluster, or a personal computer. BioCloud uses the VO-provided resources to host the BCWM and run the workflows. A VO may have multiple users, sharing datasets and workflows. The overall BioCloud system architecture for multiple VOs is illustrated in Fig. 1.

It is possible to start using the system through the web application hosted on BioCloud Portal. Upon registration, VO can simply define the resources to be used by BioCloud. These resources will be employed to host BCWM for the corresponding VO and run the workflows. VO can select the resource to host the BCWM. If a cloud resource is selected BioCloud initializes the BCWM instance using the pre-configured BCWM image in the corresponding cloud provider. Once the BCWM is initialized, BioCloud Portal enables the link to access the BCWM so that the VO can visit BCWM in a seamless manner without leaving the web-page. A local cluster or a PC can also be used to host the BCWM if OpenStack is available on the target system.

A single unique BCWM instance is employed per VO to avoid data replication across multiple cloud providers and local clusters defined by the VO. This is realized without sacrificing the ability of using resources in multiple cloud providers simultaneously which is the key contribution of the presented system. Based on the workflow and the available resources, BioCloud can determine the computational resources to be used for particular steps.

3.1. Web Interface and User Interaction

Despite the underlying distributed architecture of multiple components (i.e., BioCloud Portal and BCWM), users access BioCloud through a single, unified web interface. This user-friendly web-interface enables users to manage resources (i.e., local clusters and cloud services), design and run workflows, and collect results. Workflow management component of BioCloud is extended from

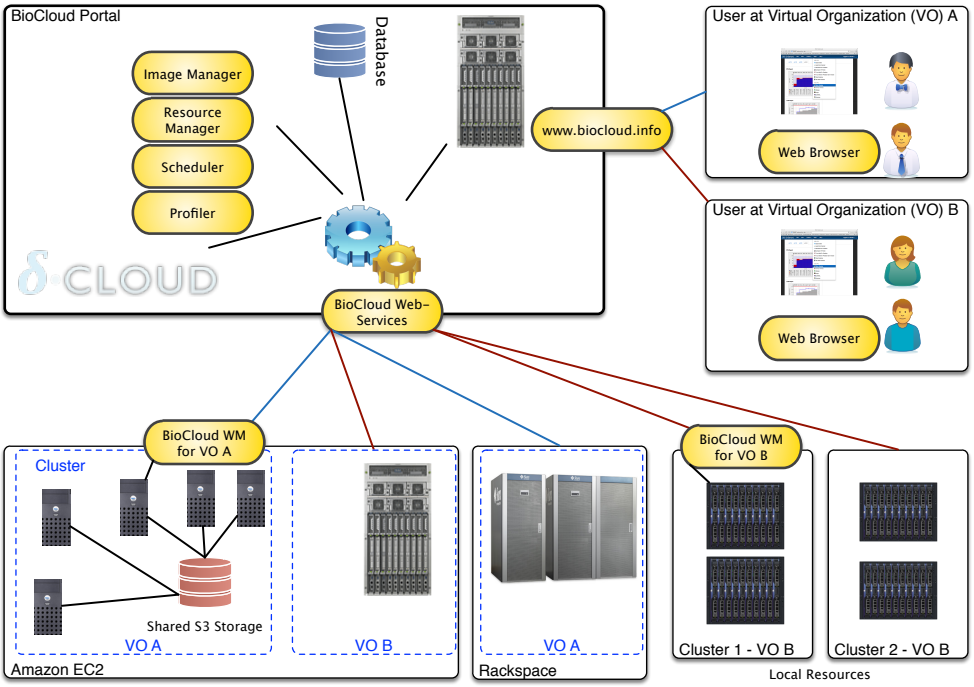

Figure 1: Overall architecture of BioCloud.

Galaxy [9] so that multi-step pipelines can be created in a simplified manner. It is possible to specify the computational resource to be used for a particular workflow step. Unless a particular resource is specified, BioCloud exploits its scheduling algorithm to designate the resources to be used for each step based on the workflow and user requirements such as cost and time. The details are provided in sections 3.2, 3.7.

3.2. Workflow Management and Execution

BioCloud workflow management component (BCWM) enables workflow creation in a drag-and-drop fashion thanks to the underlying Galaxy [9] application. On the design pane, each workflow step is represented by an independent box. Each box is associated with a tool (i.e., application) to be used in the corresponding workflow step. Two boxes can be connected with a directed edge to indicate the data flow and the resulting data dependency between them. Incoming and outgoing edges connected to the box represent input and output data respectively. Data dependency between workflow steps require preceding steps to be completed before initiating following steps. Galaxy assumes execution of the whole workflow on a single resource which also hosts Galaxy and cannot ensure data dependency when multiple resources of various cloud providers and local clusters are to be used. BioCloud eliminates such constraints and enables running different workflow steps on different resources simultaneously or sequentially. BioCloud segregates workflow management and workflow execution through its service oriented architecture. While web-services in BioCloud Portal is responsible from determining the resources to be used for each workflow step and ensuring availability of the resources through resource management and provisioning, BCWM is responsible from dispatching workflow steps for execution on the resources pre-determined by BioCloud Portal and monitoring them.

Initially, BCWM informs BioCloud Portal about the workflow to be run by sending the related information through the web-services. In order to designate the workflow execution schedule and determine the resources to be used at each step, BioCloud scheduler 3.7 is employed by BioCloud Portal. Scheduler receives the submitted workflow as a DAG (Directed Acyclic Graph) along with the associated tool names at each step and input data size. One of the key features of the scheduler is inherent workflow improvement through data partitioning and parallelism. Scheduler automatically manipulates the DAG to enable parallelism. Scheduler employs profiler 3.4 to estimate expected running times of the tools, the amount of data to be produced, and the cost of execution to be incurred considering available resources. Based on this information, scheduler identifies the resources to be used at each step considering cost and time requirements. Scheduler employs resource manager 3.5 to ensure availability of the resources before executing a workflow step. When the resources are provisioned BCWM is allowed to dispatch the next available workflow step for execution. This enables dynamic scaling up of the resources right before the execution of a particular workflow step to meet the resource demand of the corresponding VO. Resource management module also tracks the provisioned resources to

scale down based on the supply-demand balance in the next billing cycle of the provisioned resources.

BioCloud not only provides an efficient scheduler to minimize execution cost while meeting cost and time requirements which is the key contribution of this paper, it also offers a user-friendly platform to encapsulate the complexity of identifying resources to be used among several options, using resources simultaneously on multiple cloud providers to execute workflows while handling data partitioning and parallelism, dynamic resource scaling and cluster configuration in the cloud. Hiding such a complexity from the user enables her to focus on the workflow design. The user simply clicks the run button to execute the workflow. Figure 2 depicts the steps involved in execution of a workflow where the user has Amazon EC2 and Rackspace cloud accounts as well as local clusters.

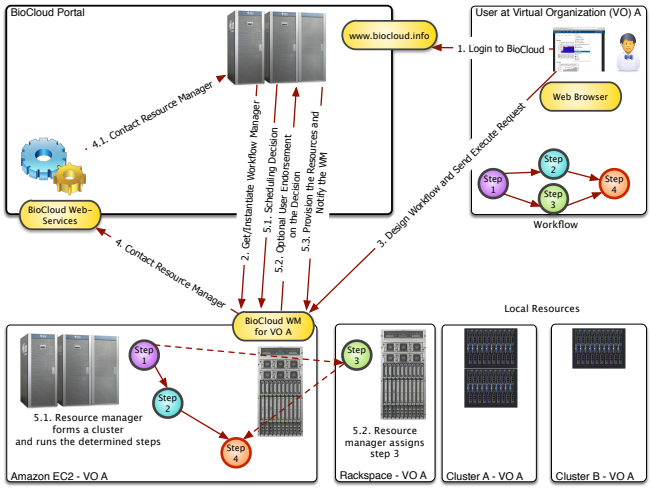

Figure 2: Steps of a workflow execution.

3.3. Authentication

Alberich policy engine [40] is leveraged by BioCloud to authenticate the users based on the defined roles and permissions. BioCloud Portal exploits permissions of the account provided by the user for the initial deployment of the BCWM and to run the workflows. Considering the fact that various steps of the workflow can be executed using different computational resources, the Alberich policy engine authenticates user for the particular resource. The users provide cloud service credentials so that the Alberich policy engine retrieves roles, permissions, and privileges to authenticate and authorize users for the resource pools. Accessing the image details, profiler information, resource information and the allowed actions are determined based on the access rights. The policy engine is extended so that the resources from different cloud resources can be used.

3.4. Profiler

Scheduling distributed applications can be challenging in a multi-cloud environment due to the lack of knowledge about the application characteristics. In order to realize a versatile multi-cloud scheduling algorithm, the knowledge about the application's runtime behavior on various resources is needed. Besides, not all the applications exhibit same kind of resource consumption pattern in all stages. Thus, looking into the resource consumption pattern, extracting the knowledge and classifying the applications can assist the scheduling algorithm for a better decision making in a multi-cloud environment. In BioCloud, we present a profiler component that monitors the resource consumption of applications and stores it in a profile database. BioCloud monitors execution of the workflow steps individually and collects profiling information such as running time and output file size for the tool used in the corresponding workflow step considering the resources exploited such as CPU, memory, number of compute nodes if a cluster is used, input file size, etc.

3.5. Resource Manager

Resource manager is a component to collect the resource information about various resources hosted in multiple cloud environments periodically. This information includes but not limited to hardware, OS image, network, secondary storage and memory of all the instances present in multiple clouds. For advanced metrics, the instances are enabled with cloud monitoring tools such as cloud watch and then these metrics are up-streamed to the broker via *Deltacloud* APIs. This resource information together with image information gives the unified view about the multi-cloud environment that will be used by the scheduler.

3.6. Image Manager

Image manager is a component to collect the available VM image information from multiple cloud providers. This information includes but not limited to Operating System, metadata about the software installed and the description of all the images present in multiple clouds. It can be collected from the images deployed in various cloud providers using *Deltacloud* API via controller. BCWM uses image manager to lunch new instances.

3.7. Scheduler

Dynamic nature of the multi-cloud environment and availability of wide variety of resources with diverse characteristics and capabilities demands provisioning appropriate set of resources in a dynamic manner in order to satisfy the requirements of an application. Some of the recent work focuses on managing applications modeled as bag of tasks. For example, the scheduling algorithm in [41] uses a linear programming model to calculate the optimal deployment configuration. The scheduler adds and eliminates instances based on the incoming requests. The work of [42] focuses on the bag of distributed tasks and introduces a heuristic algorithm that takes into account the location of the running tasks and their data sources. In contrast to these studies, the resource provisioning scheduler in BioCloud manages the workflow as a directed acyclic graph (DAG).

Efficient cloud computing requires solving multi-objective combinatorial problems such as partitioning and scheduling. Our goal is providing a scheduler which considers cost and time to complete tasks so that the resources are allocated in a such way that the execution time is reduced for the given budget while the throughput and resource utilization is improved. Therefore, scheduler should be aware of the cost model, resource availability and favorable submission time of all the cloud providers to estimate the cost involved in resource schedule. The scheduler should have the capability to estimate the completion time of an application using profiling information of the tools based on the earlier executions. One can model the execution time and utilize that for deciding the optimum number of resources under different scheduling scenarios [43].

BioCloud scheduler regards the submitted workflows as DAG and aims to maximize parallelization. For steps that can be executed in a data-parallel manner, BioCloud partitions the data and hence the associated computation as much as possible to decrease the overall running time. Once the user submits the workflow for execution, BioCloud partitions the data to match with the available resources. The details of the scheduling algorithm is provided in the next section.

Galaxy [9] also provides a partitioning capability for the tools. However, it cannot fully optimize the execution of workflows where two (or more) consecutive steps can be run using the partitioned data. In such cases, Galaxy would redundantly merge and partition the data in between the consecutive steps. As illustrated in Fig. 3, the partitioning scheme we employed in our scheduler postpones the merging step until it is required.

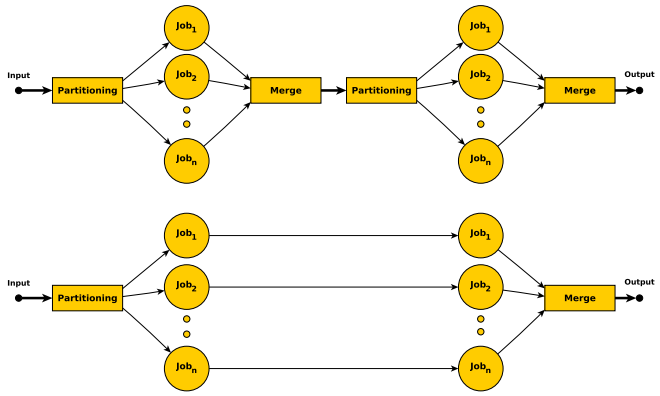

Figure 3: Partitioning scheme in BioCloud's scheduler avoids redundant merging and partitioning.

4. Scheduling Algorithm

BioCloud scheduler aims to designate a schedule for the given abstract workflow so that the execution of the improved workflow can be completed within the given deadline using the supplied budget. Scheduler also determines the resources to be used in each workflow step and cooperates with the Resource Manager to ensure provisioning of these resources. A formal definition of the scheduling algorithm is given in Alg. 1 and Alg. 2. Notations used in the algorithm are summarized in Table 1.

The scheduling algorithm initially evaluates whether the given workflow can be improved or not through data partitioning and restructures the provided abstract workflow considering execution time and cost of the workflow when different partition counts are used (Alg. 2). During this phase of the algorithm, $g \in G$, the subset of the tasks that can be executed in a data-parallel manner, is identified (Alg. 2, line 1). Having been subject to DAG, g may contain one or more tasks. For example, in the ExomeSeq workflow, presented in Section 5.1.2, sampe and alignment steps form g due to data dependency between them and thus the same partition count will be used for both steps. On the other hand, no data dependency exists between the functional annotation steps in the Transcriptome Assembly workflow, presented in Section 5.1.3, and therefore different partition counts may be used. Remark that the partition count denotes the number of compute nodes to be used.

As mentioned earlier, BioCloud exploits its profiler to estimate the execution time of the tasks based on the input file size and the resource profile to be used while considering earlier executions. Similar to our earlier work [44], we use kNN [45] as the model learning algorithm. The k nearest executions in our profile are retrieved based on a distance metric on the input parameters and their execution times are averaged and used to estimate the execution time. Cost can be calculated by considering the estimated execution time and the hourly cost of the resource profile to be used (Alg. 2, line 2).

The abstract workflow is restructured based on the desired partition count in accordance with the number of available resources that can be used in parallel and improved workflow is obtained (Alg. 2, line 4). Enabling parallelism may decrease the overall execution time of g up to a certain point and increasing the partition count beyond that may not yield the best solution due to the increased cost of resources and the inherent limits of parallelization which is related to the dynamics of the application. Considering this, we define an improvement function, I, and two thresholds, T and T', to determine the partition count while considering changes in execution time and cost. I is defined based on the relative execution time reduction and the cost increase (Alg. 2, line 6). If the relative time/cost improvement and the absolute time reduction are above the thresholds T and T' respectively, (Alg. 2, line 7), application of data partitioning to the abstract workflow will be pursued with the increased partition count.

BioCloud scheduler pursues a heuristic where the allotments of the workflow steps are initialized with the cheapest resource profile (Alg. 1, line 1) and gradually ameliorated while the workflow execution time is above the deadline and the cost is less than the available bud-

Table 1: Notation

$G = (\mathcal{V}, \mathcal{E})$	DAG of tasks $\mathcal{V}$, task dependency edges $\mathcal{E}$
$R = \mathcal{Z} \cup \mathcal{U}$	Resources (Private and Public cloud)
D	Deadline
B	Budget
L	Set of tasks in critical path
$W(L)$	Overall runtime of DAG
$C(G)$	Overall cost of DAG
g	Subset of G
a_{mc}	Resource profile ($\in R$) incurring the cheapest cost
I	Relative runtime improvement based on cost
A_j	List of profiles ($\in R$) ordered by cost for v_j
I_j	List of improvements upon using profiles in A_j for v_j
T	Relative runtime/cost improvement threshold
T'	Absolute runtime reduction threshold

Algorithm 1 Schedule(G, R, D, B)

1: setResourceProfile(G, a_{mc})
2: improveWF(G)
3: **for** $j = \{1, 2, \ldots, |\mathcal{V}|\}$ **do**
4: A_j, I_j = getValidAllotments(v_j)
5: **end for**
6: groupTasksOnCP(L, D)
7: $W(L), C(G)$ = getRunningTimeAndCost(G)
8: **while** $W(L) > D$ and $C(G) < B$ **do**
9: $maxI = maxIStep = -1$
10: **for** $\forall v_j \in L$ **do**
11: I = getNextAllocationImpr(v_j, I_j)
12: **if** $I > max$ **then**
13: $maxI = I$
14: $maxIStep = v_j$
15: **end if**
16: **end for**
17: **if** $maxI == -1$ **then**
18: break ▷ deadline/budget cannot be met
19: **end if**
20: updateGroupAllocation(G,getGroup($maxIStep$))
21: groupTasksOnCP(L, D)
22: $W(L), C(G)$ = getRunningTimeAndCost(G)
23: **end while**

get. This procedure introduces two challenges that need to be addressed. The first one is identifying the workflow step(s) for allotment improvement and the second one is determining the new allotment(s) for the selected workflow step(s). To address these issues, a list of allotments, A_j, and the resulting improvement, I_j, are assigned to each individual workflow step (Alg. 1, lines 3-5). A_j is not only sorted in the ascending order of cost but also it ensures a certain improvement, I, upon using the next allotment in the list. We use the same improvement function, I, as defined earlier to assess the resource profiles for the corresponding workflow step. The list of "valid" allotments, A_j, may contain a subset of available resource profiles and may vary for different workflow steps.

Data dependency between workflow steps may pose a major burden on meeting the deadline when consecutive workflow steps run on different resource types or cloud providers. This is referred to as inter-cluster data transfer and the cost may not be negligible especially in terms of time considering the large amount of data to be transferred. To ensure feasability of inter-cluster data transfers for the given deadline, it may be imperative to group tasks on the critical path so that the tasks in the same group will be executed on the same resource. Initially, we assume all the tasks to be run on a different resource and thus we regard each task as a separate group. If the inter-cluster data transfers are feasible for the given deadline, grouping of the tasks is completed. Otherwise, two groups with a data dependency that incurs the highest data transfer time are merged as a single group until meeting the deadline requirement. This process is referred to as grouping tasks on the critical path (Alg. 1, line 6).

Algorithm 2 improveWF(G)

1: **while** g = getUnvisitedPartitionableSubset(G) **do**
2: $W(L), C(g)$ = getRunningTimeAndCost(g)
3: $n = 2$ ▷ # nodes that will be used
4: g' = restructureSubset(g, n)
5: $W(L'), C(g')$ = getRunningTimeAndCost(g')
6: $I = \frac{(W(L)-W(L'))/W(L)}{(C(g')-C(g))/C(g)}$
7: **while** $((I > T)$ && $(W(L) - W(L') > T')) \parallel (C(g') - C(g) < 0))$ **do**
8: g' = restructureSubset(g, $++n$)
9: $W(L) = W(L'), C(g) = C(g')$
10: $W(L'), C(g')$ = getRunningTimeAndCost(g')
11: $I = \frac{(W(L)-W(L'))/W(L)}{(C(g')-C(g))/C(g)}$
12: **end while**
13: **end while**

In the rest of the algorithm, we ameliorate the allotments in an iterative manner to meet the deadline as long as sufficient budget exists to cover the costs to be incurred

(Alg. 1, line 8). Recall that we already considered available allotments for each individual step and "valid" allotments along with the resulting improvements are available. Thus, we can identify the workflow step with the highest improvement (Alg. 1, lines 9-16). Note that, the requirements may not always be satisfied due to the lack of sufficient budget or the tight deadline (Alg. 1, lines 17-19). Since we enforce the workflow steps in the same group to be run on the same resource, the new allotment is applied for the steps in the identified group (Alg. 1, line 20). Critical path is updated upon the change of the resource allocation and the execution time and the cost are calculated again. The algorithm proceeds unless the execution time drops below the deadline or the cost exceeds the budget.

5. Experimental Evaluation

5.1. Evaluating Flexibility and Interoperability

Here, we present two different scenarios to demonstrate the smooth transition from a single workstation, the only resource available for the majority of biological scientists, to a multi-cloud environment. In the first scenario, only a single workstation is assumed to be available to the user. On the other hand, besides the workstation we assume availability of multiple cloud resources in the second scenario. Two different cloud vendors are selected to demonstrate the flexibility and interoperability of BioCloud. These scenarios are tested with two different use cases (bioinformatics workflows) as explained below. We first specify the system configurations used.

5.1.1. Environment

Our testbed consist of a local workstation, and multiple instances from two cloud vendors; Amazon and Rackspace. The workstation is equipped with two Intel Xeon E5520 CPU clocked at 2.27Ghz and 48GB memory. Each CPU is a quad-core, with HyperThreading enabled and all cores share a 8MB L3 cache. The "instance type" BioCloud uses on Amazon EC2, is M3 general purpose double extra large (m3.2xlarge). This configuration has 8 cores with a memory of 30GB. The "flavor" selected on Rackspace is Performance 2 with a memory of 30GB and 8 cores. More information regarding the configuration of these particular instance types can be found in the respective cloud vendors' websites. BioCloud forms a dynamic cluster, on the fly, upon needed in the corresponding cloud service using instances of these types.

5.1.2. Use Case I: ExomeSeq *Workflow*

Our first use case is an exome sequence analysis pipeline obtained from our collaborators [46], which we will call as ExomeSeq. Test and control data with paired-end reads are used as input. In the first two steps, sequencing reads are aligned to human reference genome using BWA [47] alignment and sampe steps. The third step sorts the alignments by leftmost coordinates. Duplicates are marked in the following two steps using two different tools. In step six, local realignments around indels are performed and the last step detects indels. The abstract ExomeSeq workflow is depicted in Fig. 4. The readers can refer to [46] for more information regarding the workflow steps.

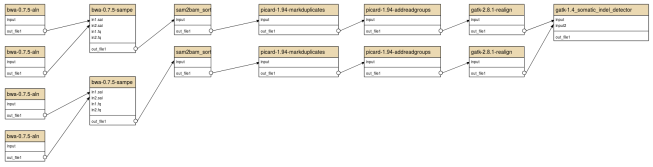

Figure 4: Abstract ExomeSeq workflow for BioCloud.

As mentioned earlier, one of the key features of BioCloud is its inherent workflow improvement facility through data partitioning and parallelism. BioCloud scheduler evaluates the submitted abstract workflow and generates an optimized workflow that would utilize available resources and hence enable parallelism. For example, for the abstract ExomeSeq workflow designed by the user (Fig. 4), BioCloud designates an improved version of this workflow as given in Fig. 5. Here, BioCloud scheduler checks whether data partitioning can be enabled for workflow steps and considering available resources and profiling data of earlier executions BioCloud scheduler determines to dispatch the test data and the control data to separate cloud resources for execution. In our scenario, the workflow steps for test data are run in Amazon EC2 and the steps for the control data are run in Rackspace. The output data of step six are transferred back to workstation and the last step is executed locally where the final result will also be stored.

The execution times of the ExomeSeq workflow using a single workstation and multiple cloud resources are provided in Fig. 6. It can be observed that the total execution time is reduced by half thanks to data partitioning and parallelism in the first two steps. While the jobs are run sequentially on the single workstation, a higher level of scalability is attained by partitioning data, creating a separate job for each data part and dispatching jobs to different compute nodes in the cluster formed in the respective cloud resources. It should be noted that data transfer time is also considered for the cloud environments. Depending on the sequence of workflow steps, data flow and the characteristics of the tools used in the workflow steps, total execution time can be reduced even further through simultaneous execution of the steps on different computing resources.

5.1.3. Use Case II: Transcriptome Assembly and Functional Annotation

De novo transcriptome assembly and functional annotation is considered as an essential but computationally intensive step in bioinformatics. The objective of the assembly and the annotation workflow is to assemble a big

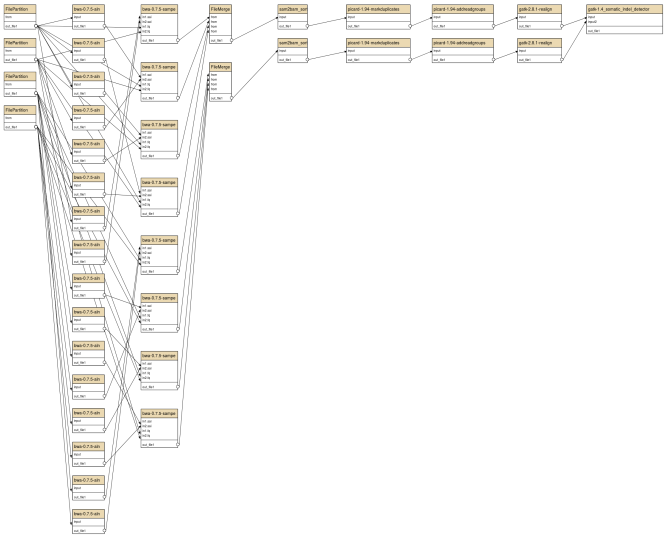

Figure 5: Generated workflow

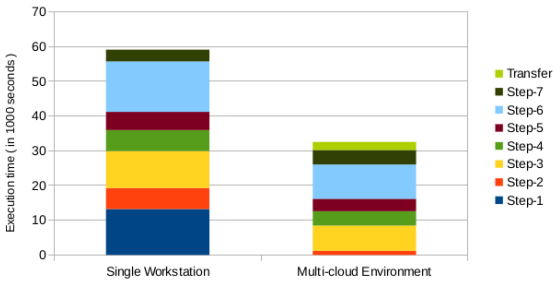

Figure 6: Execution times of the ExomeSeq workflow on a single workstation and multi-cloud environments

dataset of short reads and extract the functional annotation for the assembled contigs. As can be seen in Fig. 7, the workflow consists of four stages. The first stage is data cleaning in which a Trimmomatic [48] tool is applied on the paired-end reads dataset. After that, the output is converted to fasta format. In stage two, the assembly, the clean dataset is used as input to five different denovo transcriptome assemblers: Tran-Abyss [49], SOAPdenovo-Trans [50], IDBA-tran [51], Trinity [52] and Velvet-Oases [53]. The assembled contigs from each assembler are merged and used as input to stage three which includes clustering and removing redundant contigs as well as applying reassembly for the unique contigs. In stage three, we used TGICL tool [54] which utilizes MEGABLAST [55] for contigs clustering and CAP3 [56] for assembly. Functional annotation is done in the last stage and it is the most computational part. The blast comparison and functional annotation used in this workflow follows the pipeline detailed in [57]. Three major sequences databases: NCBI Non-redundant protein sequences (nr), Uniprot/Swiss-Prot and Uniprot/TrEMBL, are used in the sequences comparison steps. The Blastx results are parsed in the last step and their associated GO categories are generated.

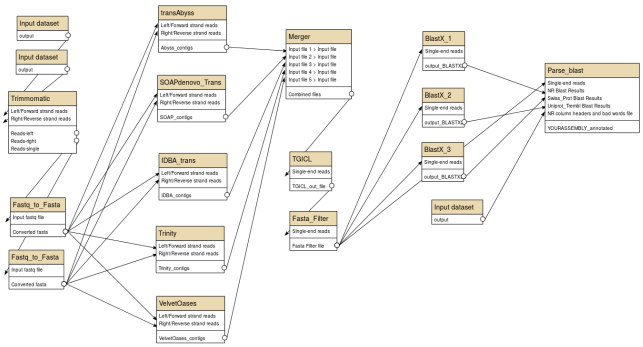

Figure 7: Assembly and annotation workflow

The used dataset is rice transcriptome data from Oryza sativa 9311 (http://www.ncbi.nlm.nih.gov/sra/SRX017631). 9.8M paired-end reads of 75bp length and totaling 1.47 Gbp were generated using Illumina GA platform [58]. The output contigs of TGICL step were filtered by removing contigs of length less than 400 base pairs. For practical issues, the number of sequences of the three proteins databases were reduced to 1% of it's original sequences count and the databases were installed in the single workstation and remote clusters.

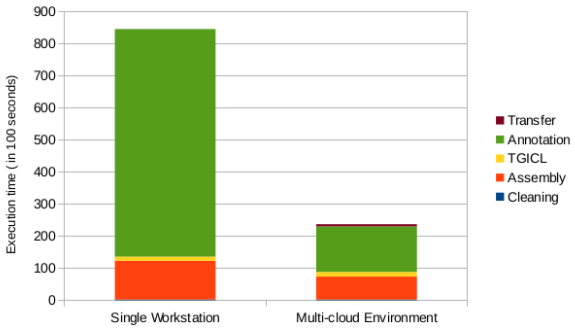

Figure 8: Execution times of the assembly and annotation workflow on a single workstation and multi-cloud environments

Similar to ExomeSeq, transcriptome assembly and annotation use case is tested in two different scenarios. In the first one, we assume that the user has access to a single commodity workstation to run the workflow. The second scenario assumes availability of multiple cloud resources besides the workstation. Fig. 8 illustrates the workflow execution times considering both scenarios. It can be seen that by utilizing the cloud resources BioCloud increases the performance 4 times. It can also be observed that the execution time of the annotation step is reduced the most due to the availability of running this step in parallel after partitioning the input data.

5.2. Evaluating Time and Cost Efficiency

In this subsection, we evaluate BioCloud in terms of cost and execution time efficiency by considering two different user scenarios on ExomeSeq and Transcriptome assembly workflows, in comparison with running each workflow with BioCloud scheduler.

In the first user scenario, we assume a user with limited budget who selects a considerably cheap resource profile in order to minimize the workflow execution cost. In the second scenario, we assume a user with considerable amount of budget so that the user is tempted to use the most powerful configurations, which also has high hourly cost. We provide the results for both ExomeSeq workflow and Transcriptome Assembly workflow. The results for the ExomeSeq workflow are depicted in Figures 9 and 10 while the results for Transcriptome Assembly are shown in Figures 11 and 12 respectively in terms of the resource scaling and the incurred cost. In the figures, we denote the first scenario with "LB" (representing user with *Limited Budget*) and the second scenario with "LT" (representing user with *Limited Time*).

Figures 9 and 11 show the number of allocated nodes while running three scenarios: LB, LT and BioCloud Portal's scheduler (BCP). As expected LT scenario always completes before LB, since LT uses more powerful instances. As seen in the figures 10 and 12, while LB is trying to minimize the cost using cheapest resources, it ends up with a longer workflow execution time and eventually with an even higher cost than the solution offered by BioCloud (BCP). Similarly, while LT is trying to minimize the execution time using expensive configurations, the user only obtains limited benefit in the execution time reduction while incurring much more cost. These results show that while BioCloud provides a flexible and easy to use environment for users to execute their workflows according to their preferences, users could also benefit BioCloud's scheduler to further optimize time vs. cost trade-off.

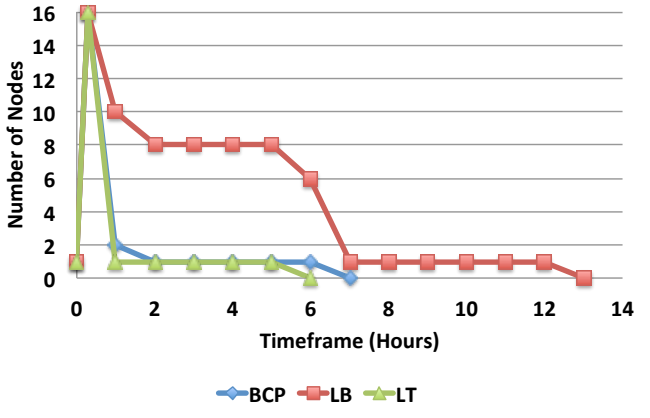

Figure 9: Scaling of the resources during the ExomeSeq workflow execution.

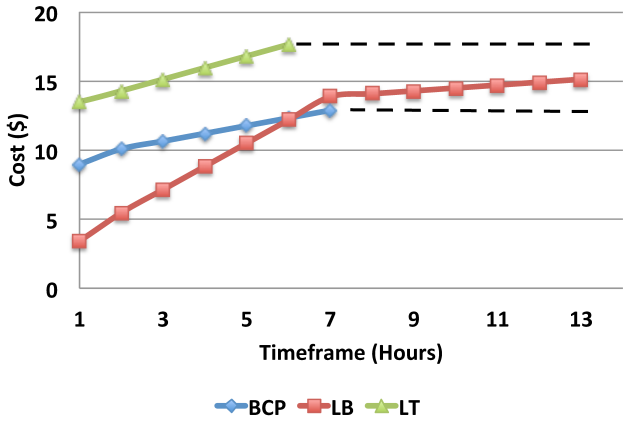

Figure 10: Change in the cost of executing ExomeSeq workflow.

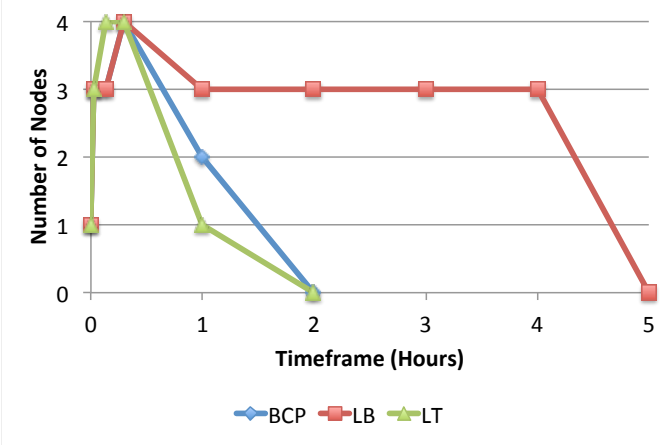

Figure 11: Scaling of the resources during the Transcriptome Assembly workflow execution.

6. Conclusion

In this paper, we present BioCloud, a Cloud Broker system, that can reduce workflow execution time for the given budget thanks to BioCloud scheduler. BioCloud enables extending local resources to a multi-cloud environment to exploit parallelism by simultaneous use of multiple computing resources for the non-computer savvy bio-researchers who lack the sufficient computing resources in order to complete executing their workflows in a reasonable amount of time. BioCloud collects data regarding execution behavior of the tools considering the resources in use so that the profiler can estimate running time and cost for the BioCloud scheduler. This enables determining the most efficient hardware profile to be used for the corresponding workflow steps. BioCloud manages resource provisioning and configuration on disparate cloud resources to assure availability of matching resources to meet the requirements dynamically at the run-time. This requires dynamic cluster configuration and dynamic scaling of the compute nodes in the cluster which would be overwhelming for the users otherwise. Another key feature

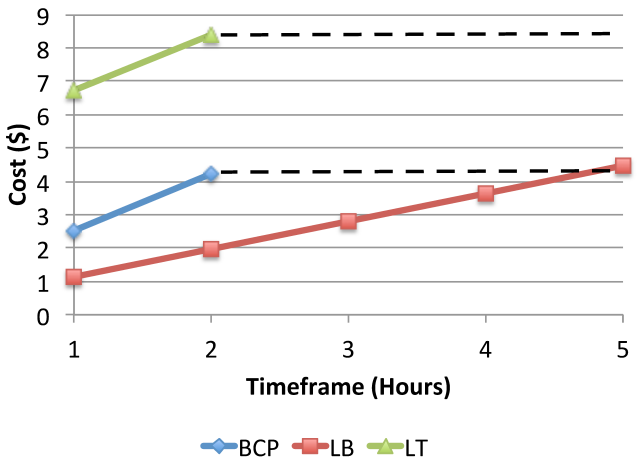

Figure 12: Change in the cost of executing Transcriptome Assembly workflow.

of BioCloud is the inherent workflow improvement facility which enhances the submitted abstract workflows and generates an improved version on-the-fly through data partitioning and parallelism. User-friendly BioCloud platform encapsulates all such complexities and simplifies the migration from single workstation to a multi-cloud environment so that the users can focus on the workflow design.

Acknowledgment

This publication was made possible by NPRP grant #4-1454-1-233 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

[1] M. A. DePristo, The $1,000 genome: The revolution in DNA sequencing and the new era of personalized medicine, The American Journal of Human Genetics 87 (6) (2010) 742 –.

[2] L. Stein, The case for cloud computing in genome informatics, Genome Biology 11 (5) (2010) 207.

[3] BioCloud at the Beijing Institute of Genomics.
URL http://biocloud.big.ac.cn/hindex.jsp

[4] C.-H. Hsu, C.-Y. Lin, M. Ouyang, Y. K. Guo, Biocloud: Cloud computing for biological, genomics, and drug design, BioMed Research International 2013.
URL http://europepmc.org/articles/PMC3808097

[5] BioCloud.
URL http://confluence.qu.edu.qa/display/KINDI/BioCloud

[6] N. Grozev, R. Buyya, Inter-cloud architectures and application brokering: taxonomy and survey, Software: Practice and Experience 44 (3) (2014) 369–390.

[7] Hardware Profiles in EC2.
URL https://aws.amazon.com/ec2/instance-types/

[8] Hardware Profiles in Rackspace.
URL http://www.rackspace.com/cloud/servers

[9] J. Goecks, A. Nekrutenko, J. Taylor, T. G. Team, Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences, Genome Biology 11 (8) (2010) R86+. doi:10.1186/gb-2010-11-8-r86.

[10] A. E. Darling, L. Carey, W. chun Feng, The design, implementation, and evaluation of mpiblast, in: In Proceedings of ClusterWorld 2003, 2003.

[11] K.-B. Li, Clustalw-mpi: Clustalw analysis using distributed and parallel computing, Bioinformatics 19 (12) (2003) 1585–1586. doi:10.1093/bioinformatics/btg192.

[12] A. Matsunaga, M. Tsugawa, J. Fortes, Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications, in: eScience, 2008. eScience '08. IEEE Fourth International Conference on, 2008, pp. 222–229. doi:10.1109/eScience.2008.62.

[13] B. Langmead, M. Schatz, J. Lin, M. Pop, S. Salzberg, Searching for snps with cloud computing, Genome Biol 10 (11) (2009) R134.

[14] A. Hunter, D. Schibeci, H. L. Hiew, M. I. Bellgard, Grendel: A bioinformatics web service-based architecture for accessing HPC resources., in: R. Buyya, P. D. Coddington, P. Montague, R. Safavi-Naini, N. P. Sheppard, A. L. Wendelborn (Eds.), ACSW Frontiers, Vol. 44 of CRPIT, Australian Computer Society, 2005, pp. 29–32.

[15] MG-RAST: Metagenomics Analysis Server.
URL http://blog.metagenomics.anl.gov/

[16] M. Reich, T. Liefeld, J. Gould, J. Lerner, P. Tamayo, J. P. Mesirov, GenePattern 2.0, Nat Genet 38 (5) (2006) 500–501.

[17] B. Nron, H. Mnager, C. Maufrais, N. Joly, J. Maupetit, S. Letort, S. Carrre, P. Tuffry, C. Letondal, Mobyle: a new full web bioinformatics framework., Bioinformatics 25 (22) (2009) 3005–3011.

[18] M. C. Schatz, CloudBurst: highly sensitive read mapping with MapReduce, Bioinformatics 25 (11) (2009) 1363–1369.

[19] T. Nguyen, W. Shi, D. Ruden, CloudAligner: A fast and full-featured MapReduce based tool for sequence mapping, BMC Research Notes 4 (1) (2011) 171+.

[20] B. Langmead, K. Hansen, J. Leek, Cloud-scale RNA-sequencing differential expression analysis with Myrna, Genome Biology 11 (8) (2010) R83+. doi:10.1186/gb-2010-11-8-r83.

[21] R. Li, Y. Li, X. Fang, H. Yang, J. Wang, K. Kristiansen, J. Wang, Snp detection for massively parallel whole-genome resequencing, Genome Research 19 (6) (2009) 1124–1132. doi:10.1101/gr.088013.108.

[22] E. Afgan, D. Baker, N. Coraor, B. Chapman, A. Nekrutenko, J. Taylor, Galaxy cloudman: delivering cloud compute clusters, BMC Bioinformatics 11 (Suppl 12) (2010) S4.

[23] S. Angiuoli, M. Matalka, A. Gussman, K. Galens, M. Vangala, D. Riley, C. Arze, J. White, O. White, W. F. Fricke, CloVR: A virtual machine for automated and portable sequence analysis from the desktop using cloud computing, BMC Bioinformatics 12 (1) (2011) 356+.

[24] K. Krampis, T. Booth, B. Chapman, B. Tiwari, M. Bicak, D. Field, K. E. Nelson, Cloud BioLinux: pre-configured and on-demand bioinformatics computing for the genomics community., BMC bioinformatics 13 (1) (2012) 42+.

[25] G. Minevich, D. S. Park, D. Blankenberg, R. J. Poole, O. Hobert, CloudMap: A cloud-based pipeline for analysis of mutant genome sequences, Genetics 192 (4) (2012) 1249–1269. doi:10.1534/genetics.112.144204.

[26] B. Liu, B. Sotomayor, R. K. Madduri, K. Chard, I. T. Foster, Deploying bioinformatics workflows on clouds with Galaxy and Globus provision., in: SC Companion, IEEE Computer Society, 2012, pp. 1087–1095.

[27] A. J. Ferrer, F. Hernndez, J. Tordsson, E. Elmroth, A. Ali-Eldin, C. Zsigri, R. Sirvent, J. Guitart, R. M. Badia, K. Djemame, W. Ziegler, T. Dimitrakos, S. K. Nair, G. Kousiouris, K. Konstanteli, T. Varvarigou, B. Hudzia, A. Kipp, S. Wesner, M. Corrales, N. Forg, T. Sharif, C. Sheridan, Optimis: A holistic approach to cloud service provisioning, Future Generation Computer Systems 28 (1) (2012) 66 – 77.

[28] Deltacloud.
URL http://deltacloud.apache.org/

[29] E. Carlini, M. Coppola, P. Dazzi, L. Ricci, G. Righetti, Cloud federations in contrail, in: Euro-Par 2011: Parallel Processing

Workshops, Vol. 7155 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 159–168.

[30] Aeolus.
URL www.aeolusproject.org/

[31] M. Rahman, S. Venugopal, R. Buyya, A dynamic critical path algorithm for scheduling scientific workflow applications on global grids, in: Proceedings of the Third IEEE International Conference on e-Science and Grid Computing, E-SCIENCE '07, IEEE Computer Society, Washington, DC, USA, 2007, pp. 35–42. doi:10.1109/E-SCIENCE.2007.3.
URL http://dx.doi.org/10.1109/E-SCIENCE.2007.3

[32] W.-N. Chen, J. Zhang, An ant colony optimization approach to a grid workflow scheduling problem with various qos requirements, Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 39 (1) (2009) 29–43.

[33] J. Yu, R. Buyya, A budget constrained scheduling of workflow applications on utility grids using genetic algorithms, in: Workflows in Support of Large-Scale Science, 2006. WORKS '06. Workshop on, 2006, pp. 1–10. doi:10.1109/WORKS.2006.5282330.

[34] M. Mao, M. Humphrey, Auto-scaling to minimize cost and meet application deadlines in cloud workflows, in: High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for, 2011, pp. 1–12.

[35] M. Malawski, G. Juve, E. Deelman, J. Nabrzyski, Cost- and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds, in: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC '12, IEEE Computer Society Press, Los Alamitos, CA, USA, 2012, pp. 22:1–22:11.
URL http://dl.acm.org/citation.cfm?id=2388996.2389026

[36] S. Abrishami, M. Naghibzadeh, D. H. Epema, Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds, Future Gener. Comput. Syst. 29 (1) (2013) 158–169. doi:10.1016/j.future.2012.05.004.
URL http://dx.doi.org/10.1016/j.future.2012.05.004

[37] Z. Wu, Z. Ni, L. Gu, X. Liu, A revised discrete particle swarm optimization for cloud workflow scheduling, in: Computational Intelligence and Security (CIS), 2010 International Conference on, 2010, pp. 184–188.

[38] E.-K. Byun, Y.-S. Kee, J.-S. Kim, S. Maeng, Cost optimized provisioning of elastic resources for application workflows, Future Gener. Comput. Syst. 27 (8) (2011) 1011–1026. doi:10.1016/j.future.2011.05.001.
URL http://dx.doi.org/10.1016/j.future.2011.05.001

[39] M. Rodriguez, R. Buyya, Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds, Cloud Computing, IEEE Transactions on 2 (2) (2014) 222–235.

[40] Alberich.
URL https://github.com/aeolus-incubator/alberich

[41] S. Srirama, A. Ostovar, Optimal resource provisioning for scaling enterprise applications on the cloud, in: Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on, 2014, pp. 262–271. doi:10.1109/CloudCom.2014.24.

[42] L. Thai, B. Varghese, A. Barker, Executing bag of distributed tasks on the cloud: Investigating the trade-offs between performance and cost, in: Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on, 2014, pp. 400–407. doi:10.1109/CloudCom.2014.29.

[43] D. Bozdağ, C. Barbacioru, Ü. Çatalyürek, Parallel short sequence mapping for high throughput genome sequencing, in: Proc. of 23rd Int'l. Parallel and Distributed Processing Symposium, 2009, pp. 1–10.

[44] G. Teodoro, T. D. R. Hartley, Ü. V. Çatalyürek, R. Ferreira, Optimizing dataflow applications on heterogeneous environments, Cluster Computing 15 (2) (2012) 125–144.
URL http://dx.doi.org/10.1007/s10586-010-0151-6

[45] E. Fix, J. Hodges, Discriminatory analysis, nonparametric discrimination, consistency properties, Computer science technical report, School of Aviation Medicine, Randolph Field, Texas (1951).

[46] J. A. Woyach, R. R. Furman, T.-M. Liu, H. G. Ozer, M. Zapatka, A. S. Ruppert, L. Xue, D. H.-H. Li, S. M. Steggerda, M. Versele, S. S. Dave, J. Zhang, A. S. Yilmaz, S. M. Jaglowski, K. A. Blum, A. Lozanski, G. Lozanski, D. F. James, J. C. Barrientos, P. Lichter, S. Stilgenbauer, J. J. Buggy, B. Y. Chang, A. J. Johnson, J. C. Byrd, Resistance mechanisms for the bruton's tyrosine kinase inhibitor ibrutinib, New England Journal of Medicine 370 (24) (2014) 2286–2294, pMID: 24869598. doi:10.1056/NEJMoa1400029.

[47] H. Li, R. Durbin, Fast and accurate short read alignment with burrowswheeler transform, Bioinformatics 25 (14) (2009) 1754–1760. doi:10.1093/bioinformatics/btp324.

[48] A. M. Bolger, M. Lohse, B. Usadel, Trimmomatic: A flexible trimmer for illumina sequence data, Bioinformaticsdoi:10.1093/bioinformatics/btu170.

[49] G. Robertson, J. Schein, R. Chiu, R. Corbett, M. Field, S. D. Jackman, K. Mungall, S. Lee, H. M. Okada, J. Q. Qian, M. Griffith, A. Raymond, N. Thiessen, T. Cezard, Y. S. Butterfield, R. Newsome, S. K. Chan, R. She, R. Varhol, B. Kamoh, A.-L. Prabhu, A. Tam, Y. Zhao, R. A. Moore, M. Hirst, M. A. Marra, S. J. M. Jones, P. A. Hoodless, I. Birol, De-novo assembly and analysis of RNA-seq data, Nature Methods 7 (11) (2010) 912.

[50] Y. Xie, G. Wu, J. Tang, R. Luo, J. Patterson, S. Liu, W. Huang, G. He, S. Gu, S. Li, X. Zhou, T.-W. Lam, Y. Li, X. Xu, G. K.-S. Wong, J. Wang, SOAPdenovo-Trans: De novo transcriptome assembly with short RNA-Seq reads, Bioinformaticsdoi:10.1093/bioinformatics/btu077.

[51] Y. Peng, H. C. M. Leung, S.-M. Yiu, M.-J. Lv, X.-G. Zhu, F. Y. L. Chin, IDBA-tran: a more robust de novo de bruijn graph assembler for transcriptomes with uneven expression levels, Bioinformatics 29 (13) (2013) i326–i334. doi:10.1093/bioinformatics/btt219.

[52] M. G. Grabherr, B. J. Haas, M. Yassour, J. Z. Levin, D. A. Thompson, I. Amit, X. Adiconis, L. Fan, R. Raychowdhury, Q. Zeng, Z. Chen, E. Mauceli, N. Hacohen, A. Gnirke, N. Rhind, F. di Palma, B. W. Birren, C. Nusbaum, K. Lindblad-Toh, N. Friedman, A. Regev, Full-length transcriptome assembly from RNA-Seq data without a reference genome, Nature Biotechnology 29 (7) (2011) 652.

[53] M. H. Schulz, D. R. Zerbino, M. Vingron, E. Birney, Oases: Robust de novo rna-seq assembly across the dynamic range of expression levels, Bioinformaticsdoi:10.1093/bioinformatics/bts094.

[54] G. Pertea, X. Huang, F. Liang, V. Antonescu, R. Sultana, S. Karamycheva, Y. Lee, J. White, F. Cheung, B. Parvizi, J. Tsai, J. Quackenbush, TIGR gene indices clustering tools (tgicl): a software system for fast clustering of large est datasets, Bioinformatics 19 (5) (2003) 651–652. doi:10.1093/bioinformatics/btg034.

[55] L. W. Zheng Zhang, Scott Schwartz, W. Miller, A greedy algorithm for aligning dna sequences, Computational Biology 7 (2000) 203–214. doi:10.1089/10665270050081478.

[56] X. Huang, A. Madan, Cap3: a dna sequence assembly program, Genome Res 9 (1999) 868877. doi:10.1089/10665270050081478.

[57] P. De Wit, M. H. Pespeni, J. T. Ladner, D. J. Barshis, F. Seneca, H. Jaris, N. O. Therkildsen, M. Morikawa, S. R. Palumbi, The simple fool's guide to population genomics via RNA-Seq: an introduction to high-throughput sequencing data analysis, Molecular Ecology Resources 12 (6) (2012) 1058–1067. doi:10.1111/1755-0998.12003.

[58] G. Zhang, G. Guo, X. Hu, Y. Zhang, Q. Li, R. Li, R. Zhuang, Z. Lu, Z. He, X. Fang, L. Chen, W. Tian, Y. Tao, K. Kristiansen, X. Zhang, S. Li, H. Yang, J. Wang, J. Wang, Deep RNA sequencing at single base-pair resolution reveals high complexity of the rice transcriptome, Genome Research 20 (5) (2010) 646–654. doi:10.1101/gr.100677.109.

Supplementary Materials

- Fig. S1 : Full size image of the abstract ExomeSeq workflow Fig. 4.
- Fig. S2 : Full size image of the generated workflow Fig. 5.
- Fig. S3 : Full size image of the assembly and annotation workflow Fig. 7.