Future Generation Computer Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A balanced scheduler with data reuse and replication for scientific workflows in cloud computing systems

Israel Casas[a,c,*], Javid Taheri[e], Rajiv Ranjan[b,c], Lizhe Wang[d], Albert Y. Zomaya[a]

[a] School of Information Technologies, The University of Sydney, Sydney, Australia
[b] School of Computing Science, Newcastle University, Newcastle-Upon-Tyne, United Kingdom
[c] Data61, CSIRO, Australia
[d] School of Computer Science, China University of Geosciences, Wuhan 430074, China
[e] Department of Mathematics and Computer Science, Karlstad University, Karlstad, Sweden

HIGHLIGHTS

- Experimentation stage includes computer and data intensive workflows.
- Incrementing number of resources does not guarantee execution time reduction.
- System utilization significantly drops as number of virtual machines increases.
- Optimal number of virtual machines depends on workflow characteristics.

ARTICLE INFO

Article history:
Received 13 April 2015
Received in revised form
13 October 2015
Accepted 5 December 2015
Available online xxxx

Keywords:
Cloud computing
Scientific workflow
Scheduling
Virtual machine
Data-intensive computing
Big data

ABSTRACT

Cloud computing provides substantial opportunities to researchers who demand pay-as-you-go computing systems. Although cloud provider (e.g., Amazon Web Services) and application provider (e.g., biologists, physicists, and online gaming companies) both have specific performance requirements (e.g. application response time), it is the cloud scheduler's responsibility to map the application to underlying cloud resources. This article presents a Balanced and file Reuse–Replication Scheduling (BaRRS) algorithm for cloud computing environments to optimally schedule scientific application workflows. BaRRS splits scientific workflows into multiple sub-workflows to balance system utilization via parallelization. It also exploits data reuse and replication techniques to optimize the amount of data that needs to be transferred among tasks at run-time. BaRRS analyzes the key application features (e.g., task execution times, dependency patterns and file sizes) of scientific workflows for adapting existing data reuse and replication techniques to cloud systems. Further, BaRRS performs a trade-off analysis to select the optimal solution based on two optimization constraints: execution time and monetary cost of running scientific workflows. BaRRS is compared with a state-of-the-art scheduling approach; experiments prove its superior performance. Experiments include four well known scientific workflows with different dependency patterns and data file sizes. Results were promising and also highlighted most critical factors affecting execution of scientific applications on clouds.

1. Introduction

Cloud computing has great impact on Information Technology (IT) solutions for both scientific and business applications [1,2]. Cloud computing environment has important features that are important for both science and business applications/purposes. Clouds also offer solutions to computationally intensive applications similar to HPC (High Performance Computing) environments such as supercomputing centers [3]. From the business perspective, clouds offer flexible platforms to both cloud providers and application owners. Cloud computing offers a unique computing ecosystem where providers and application owners can establish elastic relationship driven by application performance requirements (e.g. availability, execution time, monetary budget, etc.) and

* Corresponding author at: School of Information Technologies, The University of Sydney, Sydney, Australia.
E-mail addresses: icas8033@uni.sydney.edu.au, israel_fime@hotmail.com (I. Casas).

http://dx.doi.org/10.1016/j.future.2015.12.005
0167-739X/© 2015 Elsevier B.V. All rights reserved.

characteristics (e.g. input data size, number of end-users connecting to that application, output data size, etc.) [4,5].

Current computing applications demand research on cloud environments for their efficient use of resources [6]. Cloud providers are concerned and need to maintain their services in a relatively unreliable environment, while producing profitable revenues. Cloud application owners, on the other hand, want services/resources that meet strict performance requirements. Both require access to a cloud scheduler framework and algorithm that can automatically map applications to cloud resources while ensuring application-level performance and provider-level resource utilization goals [7,8].

Due to their data intensive nature, modern scientific applications can benefit if cloud schedulers include data reuse and replication techniques in executing their workflows [9–11]. Data reuse avoids file transfers by allocating tasks to same computing units, e.g., Virtual Machines (VMs) in context of cloud computing systems such as Amazon EC2 and Microsoft Azure. For tasks requiring the same set of files, the data replication technique duplicates the files across the VMs hosting those tasks. Both techniques can influence (enhance or degrade) optimization objectives. It is up to the scheduler to decide when to apply one or both of these techniques. The scheduler also needs to decide the number of resources (e.g., VMs) it needs to provision for an application workflow. All aforementioned considerations depend on the following two main factors: the type of application and the monetary cost model of the cloud provider (e.g., Amazon EC2 pricing[1]).

This article presents a scheme for efficient scheduling of scientific workflows in cloud environments. Main contributions of our scheduling scheme (BaRRS) include: concurrently optimizing two important, yet conflicting, objectives in cloud environments, i.e., execution runtime and monetary cost, (2) exploring the trade-off between the aforementioned two objectives through scheduling sample configurations, and (3) computing the exact number of required resources (VMs). To achieve these objectives, BaRRS combines three scheduling mechanisms to manage a workflow plan according to its task dependencies, file sizes, task execution times, and network bandwidth, as well as the underlying VMs' characteristics (e.g., Amazon EC2 instances[2]).

The article is organized as follows: Section 2 discusses the related work; Section 3 presents our system model; Section 4 presents the problem statement; Section 5 details the BaRRS; Section 6 explains experiments. Section 7 discusses the results, followed by Section 8 that concludes this work.

2. Related work

Most of the existing approaches in the context of scheduling scientific workflows consider fixed or static pool of VM resources. Recently, approaches have emerged that consider more realistic scenarios in which resources can be scaled up or down, while optimizing the monetary cost of leasing the cloud resources. To cover related literature, this section is divided into three groups. The first group discusses schedule strategies that consider only a fixed number of VM resources, while the second group surveys scientific workflow scheduling approaches which can dynamically estimate the required number of VM resources. Last group of related work considers grid and heterogeneous computing systems.

2.1. Scheduling approaches for fixed number of resources

Kloh et al. [12] proposes a scheduler for cloud environments based on multiple objectives: runtime, cost and/or reliability.

This approach is evaluated by incorporating the objectives into well-known scheduling mechanisms including dynamic scheduling [13], bi-criteria scheduling [14], cost-based scheduling [15], multiple QoS constrained schedule strategy [16], fault tolerance policies [17], and scheduling decisions based on service level agreements [18]. This algorithm outperformed the Join the Shortest Queue (JSQ) scheduling algorithm [19]. This study uses a number of resources at a higher abstraction level regardless of the incurring costs. Because it only considers a fixed number of resources for each service class (basic, master and premium), it provides no guarantee on optimum system utilization. Users also have no information about how to increase efficiency within a budget when selecting a service class.

Achar et al. [20] proposes a scheduling approach based on the Virtual Machine Tree (VMT) data structure. Scheduling decisions assign priority to tasks and VMs based on the task size in MIPS (Million Instructions per Second). Results indicate VMT outperforms the FCFS (First Come First Serve). Although their experiments used different number of VMs, they did not dynamically scale up or down to maintain/satisfy utilization constraints; this may lead to system under-utilization or over-utilization. In another approach, Fan Zhang et al. [21] describe a bi-objective scheduling approach to execute scientific workflows based on the Ordinal Optimization (OO) method. The authors' focus was to decrease the scheduling time overhead through pruning the solution space. As the modified OO did not consider scaling up/down the number of VMs as well as transfer of data files to save on network communication, it is not suitable for realistic cloud scenario considered by our work. In another approach, Deng et al. [22] proposed a linear programming approach to schedule resources to reduce extra cost, power consumption, and response time. They implemented a dynamic scheduling model that adapts to different uncertainties (e.g. overutilization) at runtime. Nevertheless, they neither recommend a precise policy to select the number of required VMs nor mention the involved monetary costs.

Authors in [23] produced an algorithm to schedule multiple workflows into cloud systems. The main characteristic of this scheduler is that it gives priority to schedule tasks with high computing demands. In another approach for cloud systems, authors in [24] presented a scheduling heuristic based on the Particle Swarm Optimization (PSO) to map workflows' tasks into resources optimizing computation and communication cost. The approach presented in [25] tackles the workflow scheduling with makespan and energy consumption as its objectives. This hybrid solution is built from a multi-objective genetic algorithm and an energy-conscious scheduling algorithm.

2.2. Scheduling approaches with resource estimation policies

Moschakis et al. [26] presented a base case study for the execution of applications deployed on the Amazon Elastic Compute Cloud (EC2). They evaluated two gang scheduling strategies on the EC2: AFCFS (Adaptive First Come First Serve) and LJFS (Largest Job First Serve). This approach however does not include deep application analysis. It schedules jobs as they arrive regardless of their tasks' dependencies, and thus seems inappropriate for scientific workflows.

Authors of [27] designed a scheduling scheme that calculates the number of required VMs to execute an application. This distribution approach follows a microeconomic scheme where users' intention is to decrease application execution time, while maintaining a budget. The cloud providers' intention is to maximize revenue. The goal, in a microeconomics context, is to reach an equilibrium point. In this model, the equilibrium point is reached when adding any more VM leads to more cost for the

[1] http://aws.amazon.com/ec2/pricing/.
[2] http://aws.amazon.com/ec2/instance-types/.

cloud provider, while removing any already scheduled VM leads to longer execution time within a user's budget. Because this study do not consider dynamic scheduling configuration where number of VMs can be changed dynamically, it cannot be used to analyze different scheduling configurations.

Oliveira et al. developed an adaptive scheduling approach for parallel scientific workflows in cloud environments [28]. Their solution includes estimating the number of VMs for allocating to workflow tasks. Their heuristic considered following variables: execution time, monetary cost, and reliability. This enables users to include constraints on execution time and monetary cost. Results showed that their heuristic outperforms native MapReduce [29] implementation. This approach does not provide a complete scheduling plan prior execution; instead, it schedules and executes a group of tasks in an iterative manner. As a consequence, the scheduler does not analyze the entire workflow at once.

2.3. Grid and heterogeneous systems

Important research has been done on scheduling of workflows in grids. As an example in [30] authors considered the problem to execute workflows into grid systems minimizing execution cost while meeting a time deadline. In their solutions they consider different types of tasks with different dependencies patterns; based on their characteristics the scheduler allocates tasks to resources fulfilling a deadline with the lowest possible monetary cost. Even though cloud systems involve a higher number of variables compared to grids incrementing the problem difficulty, furthermore the scheduling of jobs is an NP-complete problem in a general form [31] demanding a higher analysis of these systems.

In [32] they produce a scheduling algorithm for DAGs for heterogeneous systems. Their solution considers application structure as well as computation and communication cost from resources. It first assigns ranks to all tasks and finally map them to resources following an earliest finishes time methodology. In another approach for heterogeneous systems, in [33] authors developed an efficient and simple scheduler for workflows. Their solution finds the earliest finishing time for every task in a given workflow. This method first organizes tasks into a pre-scheduling based on a critical path. Then every task is assigned to the resource that contributes with the earliest finishing time.

After analyzing all the mentioned techniques, we have noticed that most of existing scheduling approaches produce a scheduling plan based on a static number of resources/VMs. A few allowed users to execute applications using different pool sizes (mainly based on users' budgets). Further most of these schedulers target simple applications with no dependencies among tasks; they lack the potential to be applicable for dynamic scheduling of scientific workflows on clouds.

To overcome the limitations of existing approaches we designed and implemented BaRRS. In summary, BaRRS (1) implements scheduling polices to allocate tasks with dependencies in scientific workflows, (2) provides users with multiple scheduling plans—with different execution times and monetary cost, and (3) implements a policy to scale up/down the number of VMs according to users' demands at run-time.

3. Framework

3.1. The cloud environment

The cloud model consists of a set of VMs with g different types (Table 1). The characteristics of each VM's type are network bandwidth, number of cores, memory and disk size. This model adopts hours as the minimum accountable time unit to hire a VM. The VM hourly cost is given by $\langle c_1, \ldots, c_g \rangle$.

Table 1
Parameter definitions.

Workflows	
$W = [t_1, \ldots, t_n]$	Set of tasks for the workflow W
in_i^{size}, $\hat{t}_i^{transfer}$	Data input size and transfer time for t_i
$\hat{t}_i^{exe}$	Task execution time
$t_i^{parents}$	Set of parents for t_i
Environment and VM configuration	
$\mathbf{VM} = [vm_1, \ldots, vm_v]$	Set of VMs
vm_j^{cores}	Total number of virtual cores in vm_j
vm_j^{cost}	Monetary cost for vm_j per quantum of time
vm_j^{mem}	VM's main memory size
vm_j^{disk}	VM's hard disk size
vm_j^{bw}	VM's network bandwidth
Scheduler	
vm_j^{queue}	Scheduling queue assign to vm_j
$\hat{vm}_j^{time}$	Estimated time to execute vm_j^{queue} on vm_j
Optimization technique	
max^{time}, min^{time}	Maximum and minimum runtime
max^{cost}, min^{cost}	Maximum and minimum monetary cost
w_1	Execution time optimization weight
w_2	Monetary cost optimization weight

3.2. Workflows

The workflow model is based on [34]. Each workflow W contains a fixed number of tasks. Each task has a set of input file(s) with size in_i^{size}. The parent set for a task t_i is given by $t_i^{parents}$. In order to execute each task, a central manager needs to transfer its respective input file set to the respective VM resource. The total estimated time to execute the ith task is denoted as:

$$\hat{t}_i^{total} = \hat{t}_i^{transfer} + \hat{t}_i^{exe}.$$

Fig. 1 presents four scientific workflow examples extracted from [35]; each with a specific dependency pattern between tasks. A workflow level is a group of tasks with a single parent group. The parallelism, P, of a level is the number of tasks building the given level; for instance, the Montage workflow in Fig. 1(a) has nine levels, where the second level has the maximum parallelism, $max(P)$, with six tasks.

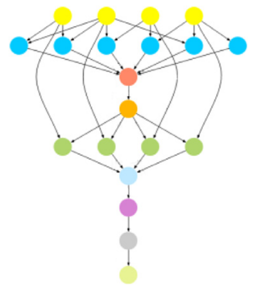
This model considers the following assumptions: (1) the system executes W at a time and contemplates computing and data intensive workflows, (2) each VM works under a particular bandwidth vm_j^{bw} that is assumed to be fixed during the execution of W, and (3) resources are requested from a cloud provider prior to execution and released after the execution of each task in the workflow. Additionally, it is assumed that users will provide the estimated execution time $\hat{t}_i^{exe}$ for all tasks in W. The file transfer time is given by the total size of file(s) divided by the minimum bandwidth value between VMs, expressed as:

$$\hat{t}_j^{transfer} = \frac{in_i^{size}}{min\left(vm_p^{bw}, vm_i^{bw}\right)}$$

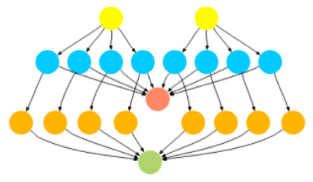
where vm_p^{bw} and vm_i^{bw} refer to the VMs executing t_i and its parent t_i^{parent} respectively.

4. Problem statement

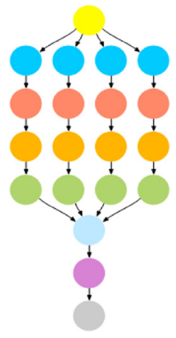
The scientific workflow scheduler formulates the scheduling problems as a weighted optimization problem of two objectives (monetary cost and runtime). It assigns tasks to VMs to minimize execution time and monetary cost based on user requirements. To formulate this problem, the considered tasks are distributed



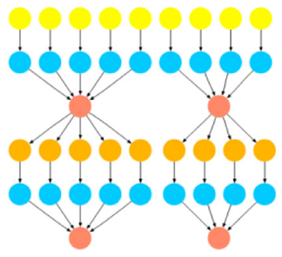
(a) Montage workflow.



(b) CyberShake.



(c) Epigenomics.



(d) LIGO.

Fig. 1. Scientific workflows.

among the VM's queues, $\left[vm_j^{queue}, \ldots vm_v^{queue}\right]$. A queue is defined as a decomposition of a set into disjointed subsets whose union is the original set. Based on this model, the scheduling problem is defined as finding the total number of queues and its corresponding elements to maximize the following augmented objective function (F):

$$F = w_1 \frac{(max^{time} - Runtime)}{(max^{time} - min^{time})} + w_2 \frac{(max^{cost} - Cost)}{(max^{cost} - min^{cost})}.$$

Variables max^{time}, min^{time}, max^{cost} *and* min^{cost} are continuously updated during scheduling process reflecting the best and worst VM configurations. At the same time the goal of function F is to highlight the best scheduling configuration among all the available ones based on the selected optimization weights.

Runtime is defined as the maximum time taken by the slowest or least powerful VM to execute the current queues of jobs i, expressed as:

$$Runtime = Max_{j=1}^{|VM|}[vm_j^{time}]$$

where,

$$vm_j^{time} = \sum_{i=1}^{|vm_j^{queue}|} t_i.$$

Cost or *Monetary* is defined as the sum of runtimes of each VM multiplied by its respective cost, expressed as:

$$Cost = \sum_{j=1}^{|VM|} \lceil Runtime \rceil vm_j^{cost}.$$

Since application owner do not have tools accurately estimate total execution time or monetary cost for their workflows, our approach offers an abstract and flexible way to choose a particular scheduling configuration driven by the percentage scale of the optimization constraints, where:

$$w_1 + w_2 = 1.$$

In this way, workflow owner gives a percentage weight to constraint based on his needs.

5. BaRRS approach

The purpose of BaRRS is to produce an estimation table for large workflows. This table is presented to application owners for comparing monetary costs and execution time tradeoffs for executing their large workflow considering heterogeneous VM configurations (e.g., CPU Type, CPU Speed, cores, memory, renting cost, etc.). Firstly, BaRRS estimates results for subset of VMs denoted as **subVM** where ***subVM*** $\subset$ ***VM*** and |***subVM***| $= \alpha$ |***VM***| size ($0 < \alpha \leq 1$). Next, BaRRS employs a trade-off analysis technique [36–38] for building the complete set of solutions considering exhaustive set of **VM configurations**. In this work, the trade-off is modeled by the exponential shape graph, $f_e(x) = A_e \times \exp(xk_e)$, that maps the number of VMs to the execution time and its respective monetary cost. As in [36], we found out exponential function has a lower mean square error (MSE) in comparison to the linear regression and other distributions. Hence we selected exponential graph to model the scheduling estimations trade-offs. Given that $subVM_i^{time}$ is the execution time of ith configuration in ***subVM*** then $A_e = \max\left(subVM_i^{time}\right)$ where $i = 1, 2, \ldots, v$. From $f_e(x)$, each VM configuration $x_i = subVM_i$ produces a different k_i:

$$k_i = \frac{\ln\left(\frac{f_e(x)}{A_e}\right)}{x_i}.$$

Then,

$$k_e = \frac{\sum_{i=1}^{|subVM|} k_i}{|subVM|}.$$

Finally,

$$f_c(x) = \sum_{j=1}^{|VM|} \lceil f_e(x) \rceil vm_j^{cost}.$$

An example of $f_e(x)$ and $f_c(x)$ is shown in Fig. 2(a) and (b). Fig. 2(c) presents an example of a complete trade-off graph with all possible combinations of VM configurations with their respective runtime and monetary cost.

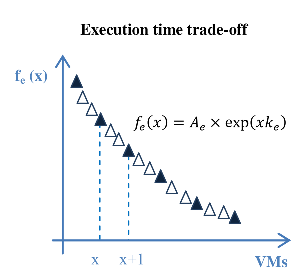
This algorithm includes the parameter $0 < \alpha \leq 1$ to control the number of estimations in order to decrease the scheduling overhead time. For $\alpha = 1$, BaRRS behaves as a brute force algorithm because it produces all possible configurations with their respective runtime and monetary cost to execute a workflow. For $\alpha < 1$, the number of configurations is uniformly distributed among all possible configurations. As an example, for $|VM| = 10$ and $\alpha = 0.5$, BaRRS will produce five $\alpha\,(|VM|)$ estimations point; they will be for $x = \{2, 4, 6, 8, 10\}$.

5.1. Balanced with file Reuse and Replication techniques Scheduling algorithm (BaRRS)

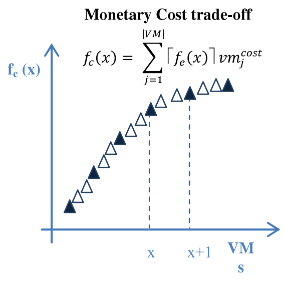
Algorithm 1 presents the BaRRS heuristic. It first computes runtime and monetary cost for the selected number of estimations (line 1). Line 2 builds the complete set of solutions for the different possible number of VM resource configurations. Finally, line 3 presents the trade-off solutions to a user.

Algorithm 1 The BaRRS Approach

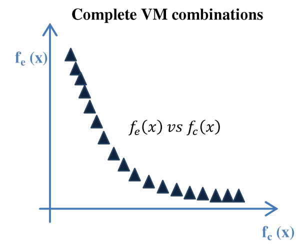
```
Input: Workflow W,
Output : Trade-off scheduling plans

1:  Estimate Runtime and Monetary Cost based on Algorithm 2
2:  Build Trade-off graph
3:  Present solutions
```



(a) Execution time.



(b) Cost.



(c) Trade-off.

Fig. 2. Trade-off frontier.

Table 2
Trade-off MSE values example.

Parameter	Value	MSE
k_1	−0.2672	0.00043
k_2	−0.3683	0.00018
k_3	−0.3539	0.00014
k_4	−0.3068	0.00014
k_5	−0.2582	0.00056
$\bar{k}$	−0.3109	0.00011

$\bar{k} = \frac{1}{n}\sum_{i=1}^{n} k_i$.

In this context, the trade-off graph is defined as a set of solutions where each solution has a different number of VMs with its respective execution time and monetary cost. The set of trade-off values connected together are called the trade-off frontier. This trade-off follows a similar shape as Pareto frontier [36,37], the main difference is that trade-off offers flexibility to analyze complete spectrum of number of VMs, which is a key feature from this study.

The exponential function $f_e(x)$ is then used to obtain runtime where x is the number of VMs. Selection of k_e and A_e values demands important attention since they drive the final trade-off shape [36]. In this work, the variable A_e corresponds to the maximum execution time for each approach. From the previous estimated solutions, k_e is obtained as

$$\log\left(\frac{f_e(VM)}{A_e}\right) = VMk_e.$$

A different k is produced for each combination of VMs with a different MSE. Based on our experimental practice, we found mean value of k_e offers minimum MSE. Fig. 3 and Table 2 present an example of this concept. From an original graph $g(VM)$, five estimated k values produce different MSE, Table 2. Then $f(VM)$ reconstruct $g(VM)$ using $\bar{k}$ as shown in Fig. 3. Since left tail on $f(VM)$ graph does not match $g(VM)$ this approach does $f(1) = A_e$ to overcome this issue.

5.2. Runtime and monetary cost estimation

Algorithm 2 computes runtime and monetary cost. It first applies workflow contraction (line 1). It then creates a VM pool. The size of the VM pool ($|VM|$) and the maximum estimated number are set in lines 2 and 3. The cycle from lines 4 to 7 obtains the best scheduling plan for the selected maximum number of estimations.

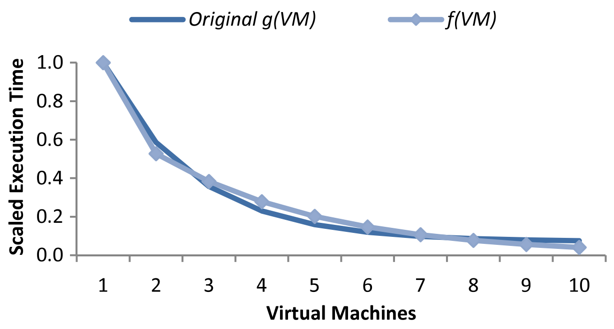


Fig. 3. Trade-off values example.

Algorithm 2 Runtime and Monetary Cost
Input: Workflow W, VM set
Output : Scheduling plan

```
1:  Workflow contraction
2:  $|VM| = max(P)$; $Solutions = \emptyset$
3:  $MaxEstimations = \alpha|VM|$
4:  For i = 1 to $MaxEstimations$
5:      Do Scheduling Algorithm
6:      $Solutions = Solutions + \{CurrentSchedule\}$
7:  End
8:  $BestSolution = Max_{i=1}^{total} F_i$
```

5.3. The scheduling algorithm

Algorithm 3 presents the Scheduling Algorithm. Its objective is to produce VMs queues. This algorithm analyzes the total workflow task levels ($|L|$). It first enumerates the total number of descended tasks from the actual level (line 2). Then all the tasks are placed as in group A (line 3). Lines 4–11 add each task to the available scheduling queues based on file reutilization (line 6) and replication (line 7) techniques. Based on these techniques, the selected tasks are added to the scheduling queue that contributes with the greatest Objective Function value.

5.4. File reutilization and replication

The file reutilization mechanism reduces the number of file transfers during workflow execution. This technique identifies parent and descended tasks and allocates them into the same VM. File replication objective is to transfer a parent task's file replica to VMs where its descended tasks will be deployed. In this approach, the policy to apply one rule or the other is based on the file transfer time saved against the task execution.

Algorithm 3 Scheduling
Input: Workflow W, VM set
Output : Scheduling plan for the given VM set

```
1:   For l=1 to |L|
2:   s = total number of descend from actual level
3:       A ← {tasks in l}
4:       while parents ≠ ø
5:         for j=1 to s //for all descended
6:             file reuse and replication as per Algorithm 4
7:             queue balancing as per Algorithm 5
8:             add task(s) to the q with the highest function value
9:         End
10:      End
11:  End
```

Algorithm 4 presents the file reutilization and replication mechanism. The complete algorithm analyzes all tasks in the workflow (lines 1–8). The bandwidth value is set to the minimum value among the machine holding the input files and a selected target VM (line 2). Line 3 presents the fundamental part of this algorithm. If the total time to transfer the input files exceeds execution time, then the task is added to the same task parent queue, e.g., reutilization. Otherwise the task is added to another VM, causing a new transfer (replication).

Algorithm 4 File reuse and replication

```
Input: Workflow $W$, $VM$ set
Output : Reorganize $vm_i^{queue} \in \{VM\}$ with file reutilization

1:  For all $t_i \in W$
2:    $bandwidth = \min(vm_i^{bw} | t_i^{parents} \in vm_i^{queue}, vm_{selected}^{bw})$
3:        If $\frac{in_i^{size}}{bandwidth} \geq \hat{t}_i^{total}$ and $t_i^{parents} = [\emptyset]$ then
4:          $vm_i^{queue} \leftarrow t_i | t_i^{parents}$
5:        Else
6:          $vm_i^{queue} \leftarrow t_i | t_i^{parents}$
7:        End
8:  End
```

5.5. Queue balance

The objective of this technique is to balance all VM queues. Scheduler rules can overload a particular queue leading to unbalanced load across VMs. To balance scheduling queues, this procedure interchange tasks between all queues in order to lower the difference between their loads without worsening any optimization goal. Algorithm 5 presents the balance technique. It first computes the average number of tasks between VMs (line 1). Line 2 empty the *bag* set. Later, lines 3–9 analyze all vm_i^{queue} for all VMs. Tasks are moved from overloaded queues, those with higher queue length than the average, to this *bag*. If a queue's size is lower than the average, tasks from *bag* are transmitted to it, considering its incurred data transfer time.

Algorithm 5 Queue balance

```
Input: Workflow $W$, $VM$ set
Output : Balanced $vm_i^{queue} \in VM$

1:  $average\_queue = |W|/|VM|$
2:  $bag = [\emptyset]$
3:  For all $vm_i^{queue} \in VM$
4:      $queue\_size = |vm_i^{queue}|$
5:      If $vm_i^{queue} > average\_queue$ then
6:          $bag \leftarrow (vm_{average_{queue}}^{queue}, vm_{queue\_size}^{queue}]$
7:      else if $vm_i^{queue} < average\_queue$ and $bag \neq [\emptyset]$
8:          $vm_i^{queue} \leftarrow [bag_1, bag_{average\_queue - queue\_size}]$
9:      End
10: end
```

5.6. Workflow contraction

BaRRS tries to group tasks of a workflow for faster distribution among VMs; it starts by grouping serial tasks. Fig. 4 exemplifies this procedure; the serially connected tasks inside dotted semicircle in Fig. 4(a) are grouped together to produce the contracted workflow in Fig. 4(b).

6. Experimental setup

To evaluate the performance of BaRRS, four scientific workflows are tested against our specialized scheduling approach. This DAGs workflow structure is generated based on [39,40] to be executed on high performance computing system. We used our VMware-ESXi-based (version 5.5) private cloud to validate our solutions. Our cloud consists of three Krypton Quattro R6010 with 4-way AMD Opteron™ 6300 series (64-Cores each), ESXi for our experiments. 12 VMs were prepared to perform as workers. Pegasus-WMS (4.2) on Ubuntu 14.04 was used as the workflow management system, where BaRRS was implemented.

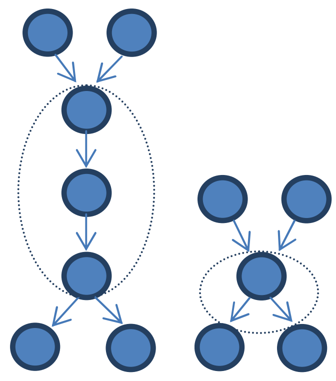

(a) Original. (b) Contracted.

Fig. 4. Workflow contraction example.

6.1. Scientific workflows

All four workflows contain 100 tasks each. Dependency details are found in [39] including their source code.

Epigenomics

This workflow has its tasks grouped across eight task levels. The *MaxLevel* (level with maximum number of tasks) contains 24 tasks. The average data size and makespan per task are 749 MB and 2346 s, respectively.

Montage

The Montage workflow's tasks are distributed across nine levels and 62 tasks build its *MaxLevel*. The average data size and makespan per task are 20.6 MB and 11.34 s, respectively.

CyberShake

The CyberShake's tasks are grouped across five levels and it has two *MaxLevel* with 48 tasks each. The average data size and makespan per task are 1156.1 MB and 51.70 s, respectively.

LIGO

The total tasks from this workflow are convened across eight levels with 24 tasks in its *MaxLevel*. The average data size and makespan per task are 55.6 MB and 222.0 s, respectively.

6.2. Comparing algorithms

As described in Section 2, most algorithms intended for cloud environment use a fixed number of VMs in their scheduling procedures. To the best of our knowledge, Provenance Adaptive Scheduling Heuristic [28] is among the state-of-the-art approaches that is also able to produce scheduling plans with different number and combinations of VMs based on execution runtime, monetary cost and reliability requirements. For this reason, the Provenance Scheduling approach is selected for comparison against to BaRRS.

The Provenance scheduler analyzes group of tasks ready for execution, it groups them in queues with sizes depending on their historical execution time profile. This approach is able to increment the number of VMs as long as the monetary cost of an application does not exceed its upper limit (monetary constraint).

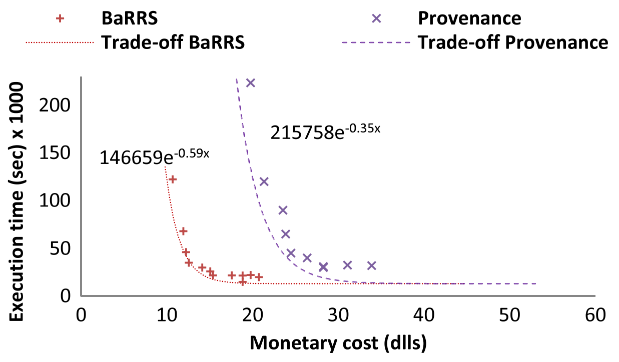


Fig. 5. Epigenomics trade off.

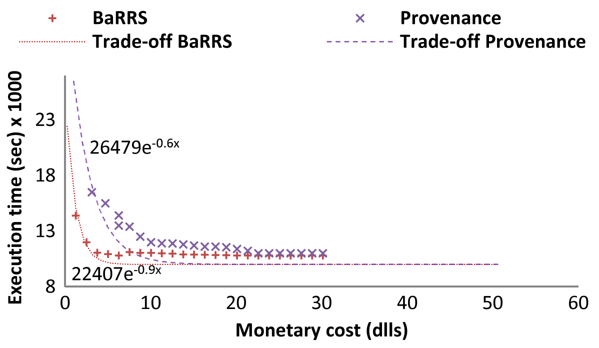


Fig. 7. CyberShake trade-off frontier.

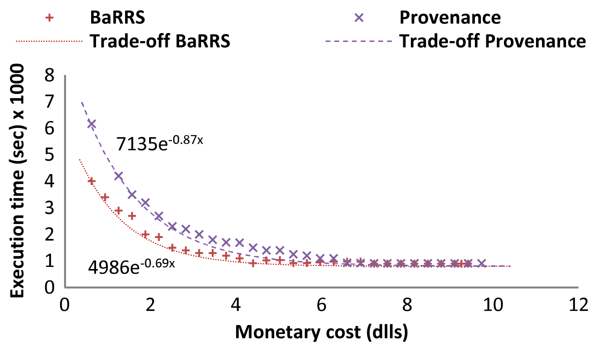


Fig. 6. Montage trade-off frontier.

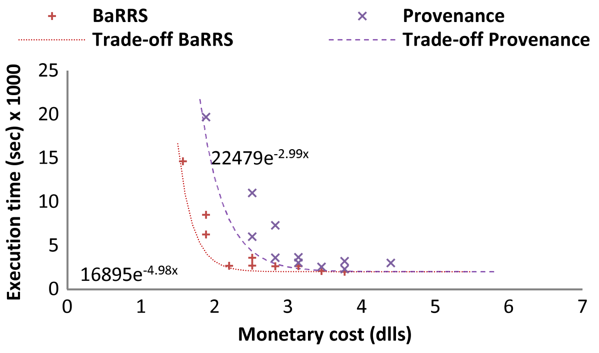


Fig. 8. LIGO trade-off frontier.

6.3. Experimentation setup

The parameter values w_1, w_2 and α are set to 0.50. On the trade-off graphs, each point corresponds to a unique number of VMs expressed as "{ }". For example {3, 5, 7} represent trade-off points for VMs three, five and seven. We based the VMs characteristics or configurations based on the General Purpose Systems model in [41] including: vm_j^{cores}, vm_j^{cost}, vm_j^{mem}, vm_j^{disk}.

7. Results

7.1. Epigenomics workflow

The Epigenomics trade-off graphs in Fig. 5 show how BaRRS outperformed the Provenance. Provenance's low performance is related to the way it schedules tasks of a workflow: one task level at the time. This causes VMs to remain idle until an entire level finishes execution. Only then, VMs can continue to execute the next level task set. Furthermore, VMs do not save files if not used by the next executing task. As a result, files are sent to a central disk and thus adding unnecessary file transfers to increase both execution time and monetary cost.

7.2. Montage workflow

The main characteristic of the Montage trade-off graphs (Fig. 6) is that most of the solutions are executed within one hour. This situation is caused by a low computing demand from Montage workflows. Tasks neither need high computing power nor large scale file transfer. It also presents a particular dependency pattern: each task on the second level depends on two tasks from the first level. BaRRS analyzes both levels and explores data reuse by identifying the pair of parent tasks and their descendants. Following that, it groups and deploys them to the same VM. For this reason, BaRRS's lead to lower execution and cost values in comparison to Provenance approach.

7.3. CyberShake workflow

Fig. 7 presents the trade-off graph for both BaRRS and Provenance approaches. An important characteristic of this workflow is that most of its solutions are executed at the average rate of 11,000 s, while not less than 10,000. The reason for such execution time is the need to transfer 80 GB input files. The network transfer time of these files contributes to about 65% of total execution time or running time.

7.4. LIGO workflow

Fig. 8 presents the trade-off graph for BaRRS and Provenance. Solutions for both approaches tend to execute at an average execution time of 3000 s at the cost of $3.00. The reason for this behavior is the uniformity of the dependency patterns. The first two task levels contract (Workflow Contraction) in a single one, same as levels four and five, converting the problem to a simple map of parallel tasks where four VMs are the correct number to achieve optimization constraints. Incrementing the number of VMs offers no execution time improvement, while increasing the monetary cost.

8. Discussion

This section presents execution time, monetary cost and system utilization results for the complete range of |VM| configurations. This measurement is to analyze the total time VMs are active during execution, which is defined as:

$$Utilization = \frac{\sum_{j=1}^{|VM|} [vm_j^{time}]}{(Runtime)\ |VM|}$$

where,

$$vm_j^{time} = \sum_{i=1}^{|vm_j^{queue}|} \hat{t}_i^{exe}.$$

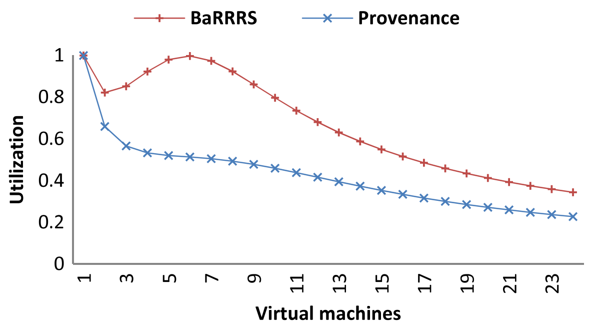


Fig. 9. Epigenomics system utilization.

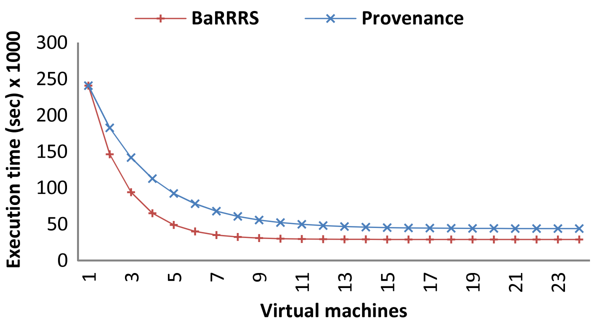


Fig. 10. Epigenomics execution time.

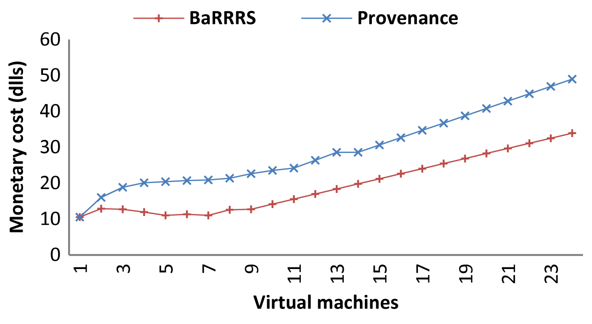


Fig. 11. Epigenomics monetary cost.

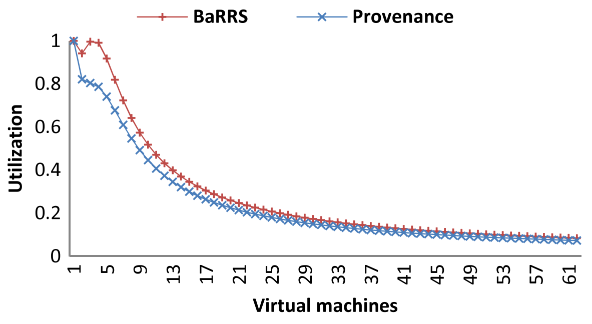


Fig. 12. Montage system utilization.

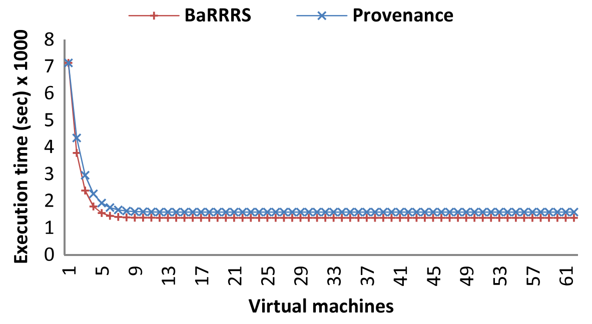


Fig. 13. Montage execution time.

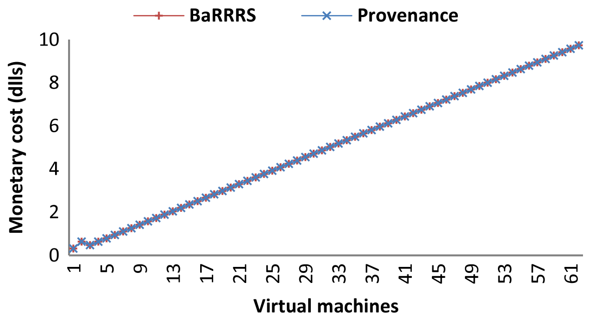


Fig. 14. Montage monetary cost.

8.1. *Epigenomics*

The six-machine configuration presents the highest utilization value for BaRRS as shown in Fig. 9. This condition is due to the following reasons. First, the Workflow's *MaxLevel* consumes 98.5% of the total execution time. Second, the size of files and makespan values are similar for all tasks. Third, the twenty four tasks in this level can distribute uniformly on six VMs. For the same reason, this configuration presents a low execution time and monetary cost as shown in Figs. 10 and 11.

The execution times gradually decrease as the number of VMs increases (Fig. 10). The reason for this behavior is the uniform distribution of tasks in the workflow graph. Furthermore, each task only depends on single parent, allowing a very uniform task deployment across the VMs.

8.2. *Montage*

An important characteristic of the montage workflow is that the maximum number of tasks across the level is 62. For simplicity we refer to this level as *MaxLevel*. Even though this level groups the majority of the tasks, it only contributes to about 57% of the total execution time. This factor causes solutions with higher utilization values to demand only a small number of VMs. Furthermore, as the number of VMs increases, their utilization significantly drops (Fig. 12) (see Figs. 13 and 14).

8.3. *CyberShake*

CyberShake solutions have considerable low system utilization as shown in Fig. 15. The main reason for this performance is the size of input files. The average transfer input time is about 92.4 s, while task average execution time is about 51.7 s. This data intensive workflow is suitable to execute on a small number of VMs to obtain the higher utilization values. Moreover, users are able to evaluate whether they execute their workflow on the cloud or on their own resources based on this analysis. This highlights the importance of our study to guide users (see Figs. 16 and 17).

8.4. *LIGO*

LIGO has two *MaxLevels* with 24 tasks. All tasks in these two groups demand different file sizes with different execution times. This causes lower system utilization values for Provenance experiments (Fig. 18). Both BaRRS and Provenance approach lead to similar results for execution time and monetary cost as presented in Figs. 19, 20; mainly because of fairly equal file transfer and execution times for tasks. Nevertheless, BaRRS still outperforms,

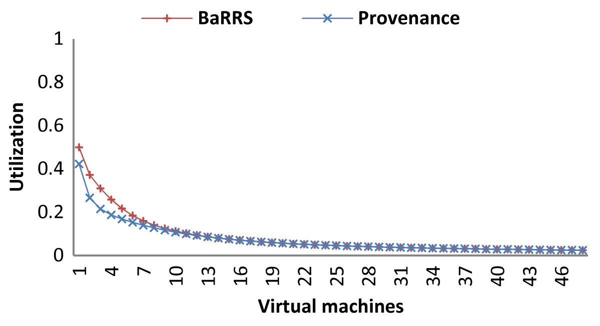


Fig. 15. CyberShake system utilization.

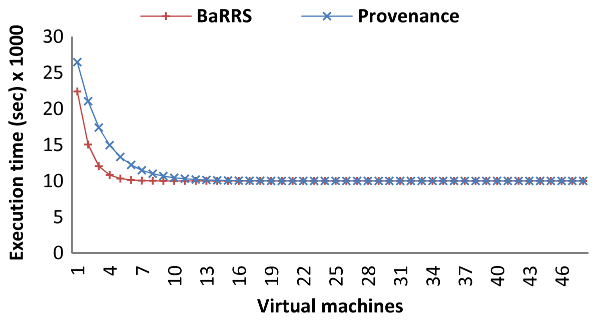


Fig. 16. CyberShake execution time.

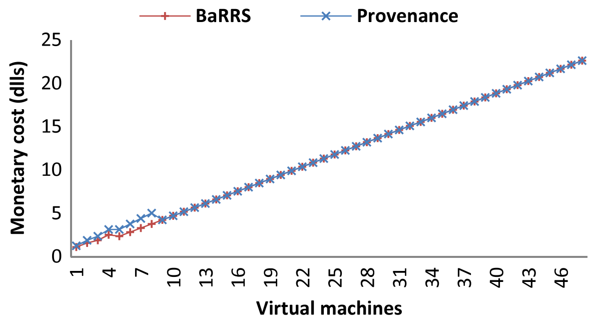


Fig. 17. CyberShake monetary cost.

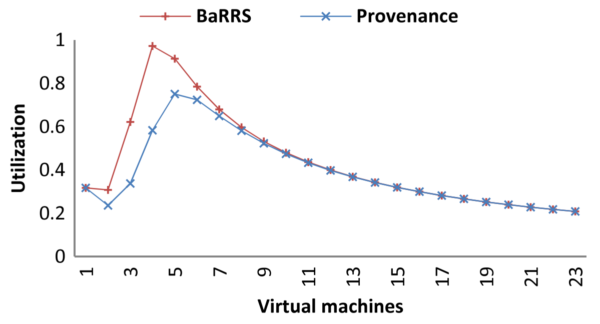


Fig. 18. LIGO system utilization.

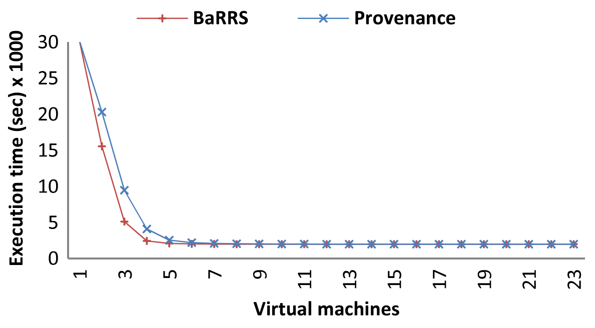


Fig. 19. LIGO execution time.

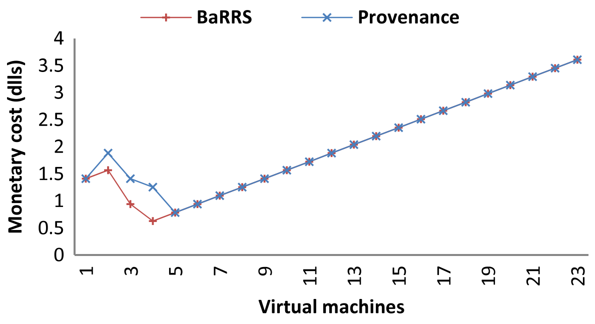


Fig. 20. LIGO monetary cost.

though marginally, Provenance because it considers all tasks during scheduling.

8.5. Discussion summary

Our results show that system utilization is proportional to task parallelism. As parallelism increases the file reuse and/or file replication techniques can influence majority of tasks. Both algorithms (BaRRS and Provenance) are able to produce reasonable solutions, however we observe that only BaRRS is capable of selecting the optimal solutions according to users' optimization requirements. Though Provenance gives users the freedom to select monetary constraints; it leads to degradation of system utilization. BaRRS, on the other hand, offers the option to free monetary limit by modifying w_1 and w_2 values causing the scheduler to still find a solution without monetary limits that does not degrade system utilization. It is important to highlight the system utilization is largely dependent on the nature of the application workflow.

9. Conclusions and future work

In this paper we presented the BaRRS scheduling approach for deploying scientific workflows on cloud-based VM resources. BaRRS is based on three techniques and a deep workflow analysis. It produces a scheduling configuration that gives an application owner the flexibility to choose different combination of VMs based on execution time and monetary cost tradeoff. These techniques include queue balancing, file reuse, and file reutilization. BaRRs approach takes a special consideration of the workflow feature analysis such as file sizes, task parallelism and task interdependencies. Four scientific workflows are selected as the application benchmark to test BaRRS performance. BaRRS was compared against the state-of-the-art Provenance scheduling approach, experiments proved BaRRS's superior performance in meeting conflicting requirements.

Experiments analyzed here reveal two important threads for future investigation. First, as the workflow dependency patterns have an important influence on selection of scheduling polices, enhancing our BaRRS scheduler to inherit diverse set of dependency patterns is important. Our first future direction is to address this issue and provide better solutions. Secondly, scheduling overhead time can impact feasibility of scheduling strategies. For that reason, future investigation is required to filter (detect and dismiss) low quality solutions before the scheduler analyzes it; this should significantly reduce the scheduling overhead time.

Acknowledgments

The authors would like to acknowledge the support of the Commonwealth Scientific and Industrial Research Organisation (CSIRO), Consejo Nacional de Ciencia y Tecnología (Conacyt), and the Australian Research Council for supporting this work.

References

[1] F. Etro, The economic impact of cloud computing on business creation, employment and output in Europe, Rev. Bus. Econ. 54 (2009) 179–208.
[2] L.I. Millett, S.H. Fuller, The Future of Computing Performance:: Game Over or Next Level?, National Academies Press, 2011.
[3] High Performance Computing—HPC Cloud Computing. Available:.
[4] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, A view of cloud computing, Commun. ACM 53 (2010) 50–58.
[5] P. Mell, T. Grance, The NIST definition of cloud computing, 2011.
[6] Q. Zhang, L. Cheng, R. Boutaba, Cloud computing: state-of-the-art and research challenges, J. Internet Serv. Appl. 1 (2010) 7–18.
[7] J.O. Gutierrez-Garcia, K.M. Sim, A family of heuristics for agent-based Cloud bag-of-tasks scheduling, in: 2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC, 2011, pp. 416–423.
[8] J. Taheri, A.Y. Zomaya, S.U. Khan, Genetic algorithm in finding Pareto frontier of optimizing data transfer versus job execution in grids, Concurr. Comput.: Pract. Exper. (2012).
[9] Ata Turk, Kerim Yasin Oktay, Cevdet Aykanat, Query-log aware replicated declustering, IEEE Trans. Parallel Distrib. Syst. 24 (5) (2013) 987–995.
[10] Sharrukh Zaman, Daniel Grosu, A distributed algorithm for the replica placement problem, IEEE Trans. Parallel Distrib. Syst. 22 (9) (2011) 1455–1468.
[11] Bill Allcock, et al., Secure, efficient data transport and replica management for high-performance data-intensive computing, in: 2001. MSS'01. Eighteenth IEEE Symposium on Mass Storage Systems and Technologies, IEEE, 2001.
[12] H. Kloh, B. Schulze, R. Pinto, A. Mury, A bi-criteria scheduling process with CoS support on grids and clouds, Concurr. Comput.: Pract. Exper. 24 (13) (2012) 1443–1460.
[13] L.A.V.C. Meyer, Strategies for dynamic workflow scheduling on grids (Ph.D. Thesis), COPPE, UFRJ, 2007 (in Portuguese).
[14] M. Wieczorek, S. Podlipnig, et al., Bi-criteria scheduling of scientific workflows for the grid, in: CCGRID'08: Pro-ceedings of the 2008 Eighth IEEE International Symposium on Cluster Computing and the Grid, IEEE Computer Society, Washington, DC, USA, ISBN: 978-0-7695-3156-4, 2008, pp. 9–16.
[15] J. Yu, R. Buyya, et al., Cost-based scheduling of scientific workflow application on utility grids, in: E-SCIENCE'05: Proceedings of the First International Conference on e-Science and Grid Computing, IEEE Computer Society, Washington, DC, USA, ISBN: 0-7695-2448-6, 2005, pp. 140–147.
[16] M. Xu, L. Cui, et al. A multiple QoS constrained scheduling strategy of multiple workflows for cloud computing, in: International Symposium on Parallel and Distributed Processing with Applications, Vol. 0, 2009, pp. 629–634.
[17] L. Ramakrishnan, D.A. Reed, Performability modeling for scheduling and fault tolerance strategies for scientific work-flows, in: HPDC'08: Proceedings of the 17th International Symposium on High Performance Distributed Computing, ACM, New York, NY, USA, ISBN: 978-1-59593-997-5, 2008, pp. 23–34.
[18] M. Bandini, A.R. Mury, et al. A Grid-QoS decision support system using service level agreements, in: Congressoda Sociedade Brasileira, de Computacao de 2009.
[19] H.-C. Lin, C.S. Raghavendra, An approximate analysis of the join the shortest queue (JSQ) policy, IEEE Trans. Parallel Distrib. Syst. 7 (3) (1996) 301–307.
[20] R. Achar, P. Thilagam, D. Shwetha, H. Pooja, Optimal scheduling of computational task in cloud using Virtual machine tree, in: 2012 Third International Conference on Paper Presented at the Emerging Applications of Information Technology, EAIT, 2012.
[21] F. Zhang, J. Cao, K. Li, S.U. Khan, K. Hwang, Multi-objective scheduling of many tasks in cloud platforms, Future Gener. Comput. Syst. 37 (2014) 309–320.
[22] L. Deng, Q. Yu, J. Peng, Adaptive scheduling strategies for cloud-based resource infrastructures, Secur. Commun. Netw. 5 (10) (2012) 1102–1111.
[23] M. Xu, L. Cui, H. Wang, Y. Bi, A multiple QoS constrained scheduling strategy of multiple workflows for cloud computing, in: 2009 IEEE International Symposium on Parallel and Distributed Processing with Applications, 2009, pp. 629–634.
[24] S. Pandey, L. Wu, S.M. Guru, R. Buyya, A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments, in: 2010 24th IEEE International Conference on Advanced Information Networking and Applications, AINA, 2010, pp. 400–407.
[25] M. Mezmaz, N. Melab, Y. Kessaci, Y.C. Lee, E.-G. Talbi, A.Y. Zomaya, et al., A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems, J. Parallel Distrib. Comput. 71 (2011) 1497–1508.
[26] I.A. Moschakis, H.D. Karatza, Evaluation of gang scheduling performance and cost in a cloud computing system, J. Supercomput. 59 (2) (2012) 975–992.
[27] K. Tsakalozos, H. Kllapi, E. Sitaridi, M. Roussopoulos, D. Paparas, A. Delis, Flexible use of cloud resources through profit maximization and price discrimination, in: 2011 IEEE 27th International Conference on Paper Presented at the Data Engineering, ICDE, 2011.
[28] D. de Oliveira, K.A. Ocaña, F. Baião, M. Mattoso, A provenance-based adaptive scheduling heuristic for parallel scientific workflows in clouds, J. Grid Comput. 10 (3) (2012) 521–552.
[29] J. Dean, S. Ghemawat, MapReduce: a flexible data processing tool, Commun. ACM 53 (1) (2010) 72–77.
[30] J. Yu, R. Buyya, C.K. Tham, Cost-based scheduling of scientific workflow applications on utility grids, in: 2005. First International Conference on e-Science and Grid Computing, 2005, pp. 8–147.
[31] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, 1979, Freeman, San Francisco, LA, 1979.
[32] R. Sakellariou, H. Zhao, A hybrid heuristic for DAG scheduling on heterogeneous systems, in: Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International, 2004, p. 111.
[33] R. Sakellariou, H. Zhao, E. Tsiakkouri, M.D. Dikaiakos, Scheduling workflows with budget constraints, in: Integrated Research in GRID Computing, Springer, 2007, pp. 189–202.
[34] M.R. Hoseinyfarahabady, H.R. Samani, L.M. Leslie, Y.C. Lee, A.Y. Zomaya, Handling uncertainty: Pareto-efficient BoT scheduling on hybrid clouds, in: 2013 42nd International Conference on Paper Presented at the Parallel Processing, ICPP, 2013.
[35] Shishir Bharathi, et al., Characterization of scientific workflows, in: Workflows in Support of Large-Scale Science, WORKS 2008. Third Workshop on, IEEE, 2008.
[36] J. Taheri, A.Y. Zomaya, S.U. Khan, Genetic algorithm in finding pareto frontier of optimizing data transfer versus job execution in grids, Concurr. Comput.: Pract. Exper. (2012).
[37] Javid Taheri, et al., Pareto frontier for job execution and data transfer time in hybrid clouds, Future Gener. Comput. Syst. 37 (2014) 321–334.
[38] Javid Taheri, et al., Hopfield neural network for simultaneous job scheduling and data replication in grids, Future Gener. Comput. Syst. 29 (8) (2013) 1885–1900.
[39] Pegasus: Workflow Management System [Online]. Available: http://pegasus.isi.edu/.
[40] HTCondor: High Throughput Computing [Online]. Available: http://research.cs.wisc.edu/htcondor/.
[41] AWS | Amazon Elastic Compute Cloud (EC2) [Online]. Available: http://aws.amazon.com/ec2/.

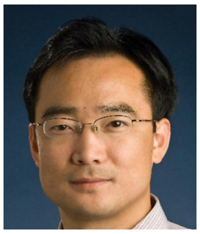
Israel Casas is a researcher on The Information Technology School at The University of Sydney, Australia. He is a member of the Centre for Distributed and High Performance Computing at mentioned School. His main research interests include Cloud Computing, Parallel Computing, Optimization Techniques, Embedded Systems and Microcontrollers. Casas started his research experience at the Electronic and Computer Department at Monterrey Institute of Technology and Higher Education, Mexico. He has also contributed with the University of California (Irvine) for the exploration and evaluation of embedded systems with software focus.

Javid Taheri received his Bachelor and Masters of Electrical Engineering from Sharif University of Technology, Tehran, Iran in 1998 and 2000, respectively. His Master was in the field of Intelligent Control and Robotics. His Ph.D. is in the field of Mobile Computing from the School of Information Technologies in the University of Sydney, Sydney, Australia. He is currently working as a Postdoctoral research fellow at same school. His main areas of research are Optimization Techniques, Artificial Intelligence, Vehicular Ad-hoc Networks, Scheduling, and Parallel Computing.

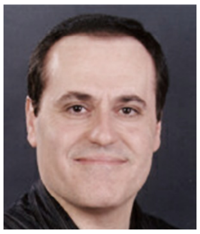
Rajiv Ranjan is a Scientist in the CSIRO ICT Center, Information Engineering Laboratory, Australian National University, Canberra, where he is working on projects related to cloud and service computing. Previously, he was a Senior Research Associate (Lecturer level B) in the School of Computer Science and Engineering, University of New South Wales (UNSW). Dr. Ranjan has a Ph.D. (2009) in Computer Science and Software Engineering from the University of Melbourne. He completed Bachelor of Computer Engineering from North Gujarat University, India, in 2002. Dr. Ranjan is broadly interested in the emerging areas of cloud, grid, and service computing. The main goal of his current research is to advance the fundamental understanding and state of the art of provisioning and delivery of application services in large, heterogeneous, uncertain, and evolving distributed systems.

Dr. Ranjan has more than 50 research publications in journals with high impact factor (according to JCR published by ISI), in proceedings of IEEE's/ACM's premier conferences and in books published by leading. Dr. Ranjan has often been invited to served as Guest Editor for leading distributed systems and software engineering journals including Future Generation Computer Systems (Elsevier

Press), Concurrency and Computation: Practice and Experience (John Wiley & Sons), and Software: Practice and Experience (Wiley InterScience). He was the Program Chair for 2010 and 2011 Australasian Symposium on Parallel and Distributed Computing and 2010 IEEE TCSC Doctoral Symposium. He serves as the editor of IEEE TCSC Newsletter. He has also recently initiated (as chair) IEEE TCSC Technical area on Cloud Computing.



Lizhe Wang is a Professor at Institute of Remote Sensing & Digital Earth, Chinese Academy of Sciences (CAS) and a ChuTian Chair Professor at School of Computer Science, China University of Geosciences (CUG). Prof. Wang received his B.E. & M.E from Tsinghua University and Doctor of Engineering from University of Karlsruhe (Magna Cum Laude), Germany. Prof. Wang is a Fellow of IET, and Fellow of British Computer Society. Dr. Wang serves as Associate Editor of IEEE Transaction on Computers and IEEE Transaction on Cloud Computing. His main research interests include high performance computing, e-Science, and spatial data processing.



Albert Y. Zomaya is the Chair Professor of High Performance Computing & Networking and Australian Research Council Professorial Fellow in the School of Information Technologies, Sydney University. He is also the Director of the Centre for Distributed and High Performance Computing which was established in late 2009. Dr. Zomaya published more than 500 scientific papers and articles and is author, co-author or editor of more than 20 books. He is currently the Editor in Chief of the IEEE Transactions on Computers and Springer's Scalable Computing and serves as an associate editor for 22 leading journals. Dr. Zomaya is the Founding Editor of the Wiley Book Series on Parallel and Distributed Computing.

Dr. Zomaya was the Chair of the IEEE Technical Committee on Parallel Processing (1999–2003) and currently serves on its executive committee. He is the Vice-Chair, IEEE Task Force on Computational Intelligence for Cloud Computing and serves on the advisory board of the IEEE Technical Committee on Scalable Computing and the steering committee of the IEEE Technical Area in Green Computing. Dr. Zomaya has delivered more than 130 keynote addresses, invited seminars, and media briefings and has been actively involved, in a variety of capacities, in the organization of more than 600 conferences.