Isidoro Gitler
Jaime Klapp (Eds.)

Communications in Computer and Information Science 595

High Performance Computer Applications

6th International Conference, ISUM 2015
Mexico City, Mexico, March 9–13, 2015
Revised Selected Papers

Springer

Dynamic Communication-Aware Scheduling with Uncertainty of Workflow Applications in Clouds

Vanessa Miranda[1], Andrei Tchernykh[1(✉)], and Dzmitry Kliazovich[2]

[1] CICESE Research Center, Ensenada, B.C., Mexico
vanessa.vsphinx@gmail.com, chernykh@cicese.mx
[2] University of Luxembourg, Luxembourg, Luxembourg
Dzmitry.Kliazovich@uni.lu

Abstract. Cloud computing has emerged as a new approach to bring computing as a service, in both academia and industry. One of the challenging issues is scientific workflow execution, where the job scheduling problem becomes more complex, especially when communication processes are taken into account. To provide good performance, many algorithms have been designed for distributed environments. However, these algorithms are not adapted to the uncertain and dynamic nature of cloud computing. In this paper, we present a general view on scheduling problems in cloud computing with communication, and compare existed solutions based on three models of cloud applications named CU-DAG, EB-DAG and CA-DAG. We formulate the problem and review several workflow scheduling algorithms. We discuss the main difficulties of using existed application models in the domain of computations on clouds. Finally, we show that our CA-DAG approach, based on separate vertices for computing and communications, and introducing communication awareness, allows us to mitigate uncertainty in a more efficient way.

Keywords: Cloud computing · Scheduling · Workflow · Communication awareness · Uncertainty · DAG

1 Introduction

Nowadays, there is a need for new technologies that allow rapid access to services, applications and information. The high cost of these technologies has hindered progress in science and business over the years. Cloud computing is a new approach that brings users closer to computer capabilities without licensing software, investing in hardware, repairing and updating issues.

Shane Robinson [1] defined cloud computing as "Everything as a Service", a new approach to deliver services in a pay per use basis. Computer Sciences Corporation [2] shows that 3645 users reported that improved data center efficiency and lower operational costs are the main reasons for the adoption of cloud solutions.

On the other hand, cloud computing still has many issues that need to be addressed including efficient service provisioning, virtual machine migration, energy efficiency,

I. Gitler and J. Klapp (Eds.): ISUM 2015, CCIS 595, pp. 169–187, 2016.
DOI: 10.1007/978-3-319-32243-8_12

server consolidation, traffic management, data security, software frameworks, and many other [3, 25, 26].

According to the National Institute of Standards and Technology (NIST) cloud computing is defined as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [4]. In general, cloud computing provides the hardware and platform level resources as services on an on-demand basis.

A key dimension of scheduling policies concerns with how to map a set of tasks to a set of resources. Typically, there are two ways: static scheduling and dynamic scheduling. With the static approach, we know the detailed information about job characteristics and network topology characteristics in advance making it possible to achieve a near optimal schedule even for complex workflows. The static approach produces schedules only when all tasks are ready. Unfortunately, the availability and performance of cloud resources is difficult to predict. These resources are not dedicated to one particular user, and, besides, there is no knowledge of network's topology.

Though service-oriented architecture assumes everything as a service, most popular offers of cloud providers are Software (SaaS), Platform (PaaS), and Infrastructure (IaaS) as a service.

In SaaS, there are extensive and complex workflows which cover interaction between clients (users), service providers and data bases. For example, to access to a bank account it is necessary to check the user identity, access database, verify balance, decide if it is sufficient, and, finally, return the result to the user.

With the adoption of the Cloud computing and development of corresponding technologies, scientists and engineers can build more and more complex applications and workflows to manage and process large data sets, and execute scientific experiments on distributed resources, etc. [5]. Pandey et al. [6] defines a workflow as a process consisting of a series of steps that simplifies the complexity of execution and management of applications. Workflow implies complex communications between tasks. However, communication-aware scheduling problems that require availability of communication resources are rarely addressed. The communication properties are captured weakly by current application models and scheduling solutions. Unfortunately, it may result in inefficient utilization of cloud infrastructure and communication media.

Cloud applications and services can be represented by workflows modeled by DAGs [23, 24, 27, 29]. The vertices of a DAG represent the amount of computing that has to be processed for successful execution of the application, while the edges define precedence constraints. Such a workflow model works well for HPC applications, but fails in clouds where communication processes often become a bottleneck.

There are three main approaches to adapt the standard DAG model to the increased complexity: communication-unaware (CU-DAG), edges-based (EB-DAG), and CA-DAG introduced in [11].

In EB-DAG, vertices represent both computing and communication requirements of a job. In CU-DAG, edges are associated with the communications performed by the tasks. However, both models appeared to have shortcomings. The first model fails to make

distinction between the computing and communication tasks of a job preventing their efficient scheduling on processors and communication network. The latter approach does not allow a single communication process to precede two computing tasks, as a single edge cannot lead to two different vertices in a DAG. In CA-DAG, two types of vertices are considered: one for computing and one for communications. Details are discussed in Sect. 5.

1.1 Uncertainty

Current approaches for cloud scheduling are not very efficient. Most of them focus on deterministic environments and assume having complete knowledge about user tasks and the system. However, in real clouds, there is a considerable level of uncertainty, being the main problem of cloud computing bringing additional challenges to end-users, resource providers, and brokering [9].

Providers might not know the quantity of data that can be managed, or the amount of computation required for a group of tasks. For example, every time a user inquires about the status of a bank account, the reply time may differ. Furthermore, the use of virtualization techniques almost completely prevents obtaining exact knowledge about the system. Parameters like an effective processor speed, number of available processors, and actual bandwidth constantly change over time. Therefore, providers are always searching how to improve the management of resources to ensure Quality of Service (QoS). Uncertainty may be presented in different components of the computational and communication process.

Tchernykh et al. [9] discuss various types and sources of uncertainty: variety of data types and their values, dynamic elasticity, dynamic performance changing, virtualization with loose coupling of applications to the infrastructure, resource provisioning time variation, migration, consolidation, inaccuracy of application runtime estimation, variation of processing times and data transmission, workload uncertainty, cost (dynamic pricing), processing time constraints, effective bandwidth variation, resource availability, and other phenomena. The authors describe their impact on service provisioning, effective performance, effective bandwidth, processing time, available memory, number of available processors, available storage, data transfer time, resource capacity, network capacity, etc.

Uncertainty can be viewed as a lack of precise knowledge for future needs and parameters, or the lack of complete vision for the possible outcomes [10]. It can be classified in several different ways according to their nature as: (1) The long-term uncertainty is due to the object being poorly understood and inadvertent factors that can influence its behavior; (2) Retrospective uncertainty is due to the lack of information about the behavior of the object in the past; (3) Technical uncertainty is a consequence of the impossibility of predicting the exact results of decisions; (4) Stochastic uncertainty is a result of the probabilistic (stochastic) nature of the studied processes and phenomena, where the following cases can be distinguished: there is reliable statistical information; the situation is known to be stochastic, but the necessary statistical information to assess its probability characteristics is not available; a hypothesis on the stochastic nature requires verification; (5) Constraint uncertainty is due to partial or

complete ignorance of the conditions under which the solutions have to be taken; (6) Participant uncertainty occurs in a situation of conflict of main stakeholders: cloud providers, users and administrators, where each side has own preferences, incomplete, inaccurate information about the motives and behavior of opposing sides; (7) Goal uncertainty is associated with conflicts and inability to select one goal in the decision or building multi objective optimization model. It addresses the problem of competing interests and multi-criteria choice of optimal decisions under uncertainty; (8) Condition uncertainty occurs when a failure or a complete lack of information about the conditions under which decisions are made; (9) Objective uncertainty occurs when there is no ambiguity when choosing solutions, there is more than one objective function to be optimized simultaneously, and there exists a possibly infinite number of Pareto optimal solutions.

As already discussed, cloud scheduling algorithms are generally split into an allocation part and a local execution part. At the first part, a suitable machine for each job is allocated using a given selection criterion.

To mitigate uncertainty of such a scheme, prediction of job execution time and queue waiting times is important to increase resource allocation efficiency. However, accurate job runtime prediction is a challenging problem. It is difficult to improve prediction by historical data, prediction correction, prediction fallback, machine learning techniques including linear, quadratic and nonparametric regression, decision trees, support vector machine and k-nearest neighborhood, statistical prediction, self-similarity and heavy-tails characteristics, etc. [13, 27, 30–32].

1.2 Quality of Service

Low levels of QoS can be caused by poor scheduling decisions, leading to unacceptably long task execution times and low throughput. Accordingly, the scheduling algorithms for cloud computing must be able to adapt to dynamic changes of the system that are not adequately provided by traditional approaches to resource optimization. Moreover, most of these algorithms do not take into account an important aspect of variation in communication delays.

The need for data transfer among tasks, which is often heavy, can slow down task execution significantly. Therefore, accounting for these communications is essential to attain efficient hardware and software utilization. Given its importance, several heuristic methods have been developed for considering communications in the scheduling problems. The most widely used are list scheduling, task clustering and genetic algorithms.

Most cloud computing applications require availability of communication resources for their operations. All of the surveyed cloud applications impose communication requirements in terms of the network bandwidth, delay or both. Applications, such as video streaming, cloud storage, and cloud backup require high bandwidth to transfer large amounts of data to or from the end users, while performing almost no computations.

The availability of the communication resources becomes crucial and determines how cloud applications interact with the end users. Indeed, most of the cloud applications

process requests from and deliver results to many parts of the Internet. In addition to these external communications, cloud applications interact among themselves produce internal datacenter traffic, which may account for as much as 75 % of the total traffic [7].

Current models of cloud applications rely mostly on the HPC concepts [8, 27, 28]. These models are based on DAGs. Such models perfectly fit to the computationally intensive HPC applications, but fail for most part of cloud applications, where communications must be taken into account as well.

2 Communication Awareness

In this section, we discuss general properties of the communication-aware model of cloud applications. As we already mentioned most scheduling algorithms employ an idealized model where the problem is represented as a DAG.

Most of these algorithms are based on a very simple model, which does not accurately reflect real parallel systems. They imply several simplified assumptions: inter-processor communications are supported by dedicated sub-systems, communications are completely concurrent, and communication networks are fully connected. Last two assumptions avoid the consideration of contention for communication resources in task scheduling.

In order to solve these drawbacks, several models have been proposed to adapt the standard DAG model by either allowing vertices to represent both computing and communication requirements of a task (Communication-unaware models, see Sect. 5) and associating edges with the communications performed tasks (Edges-based models, see Sect. 5).

Nevertheless, these approaches are not quite adequate either. In the first model, if a resource is occupied by one communication, any other communication requiring the same resource has to wait until it becomes available. In turn, the task depending on the delayed communication is also forced to wait. Thus, conflicts among communications result in significantly higher overall execution time. The second model, called edge scheduling, achieved contention awareness by scheduling the edges of the DAG onto the links of the topology graph, though, this approach has a drawback. With the purpose of mapping edges onto the links, the topology graph is assumed to contain all the necessary information about nodes, edges and their connectivity information, at any time.

The above mentioned models provide a dedicated bandwidth along a predefined network path for the whole duration of the communication. On the other hand, most of current communication networks are packet-switched and packet routing decisions are taken at every hop, independently. Moreover, most of the data-center network topologies introduce multipath connections as a mean to provide resilience and load balancing. All these create excellent possibility to parallelize data center communications allowing us to divide a single communication task into n different independent communication tasks.

We use the communication-aware model of cloud applications, called CA-DAG [11]. It allows making separate resource allocation decisions, assigning processors to handle computing jobs and network resources for information transmissions, such as

application database requests. It is based on DAGs that in addition to computing vertices include separate vertices to represent communications. This communication-aware model creates space for the optimization of many existing solutions to resource allocation, as well as developing completely new scheduling schemes of improved efficiency.

3 Related Work

In this section, we give a brief overview of the distributed computing scheduling techniques that take into account communication costs.

Table 1 shows the summary of algorithm domains. Table 2 presents the main characteristics of described algorithms, and Table 3 summarizes the criteria used to study quality of the algorithms.

Table 1. Environments and models of related work's algorithms

Algorithms		Data centers	Clusters	Grid computing	Distributed computing	Cloud	Three-level (tier)	Two level (tier)
mGACA	Genetic Acyclic Clustering Algorithm		•					
CCTS	Communication Contention in Task Scheduling				•			
PSO	Particle Swarm Optimization-based Heuristic					•		
BAR	Balance-Reduce					•		
SHEFT	Scalable Heterogeneous-Earliest-Finish-Time					•		
DBA	Distributing Bartering Algorithm					•		
DENS	Data center energy-efficient network-aware scheduling	•					•	
NS2	Network simulator 2			•				
NAMC	Network-aware Migration control	•						•
CA-DAG	Communication-Aware Directed Acyclic Graph					•		

mGACA – Genetic Acyclic Clustering Algorithm [19]. It combines task clustering with a meta-heuristic for efficient scheduling of applications with communication costs, especially for large communications. The major feature of the algorithm is that it takes advantage of the effectiveness of task clustering for reducing communication delays combined with the ability of the genetic algorithms for exploring and exploiting information of the search space of the scheduling problem.

CCTS – Communication Contention in Task Scheduling [21]. The authors study the theoretical background of edge scheduling including aspects like heterogeneity, routing and causality. They propose a new system model for task scheduling that

abandons the assumptions of a fully connected network and concurrent communication, capturing both end-points and network contention.

PSO – Particle Swarm Optimization-based Heuristic [6]. This paper presents a particle swarm optimization (PSO) based heuristic to schedule applications to cloud resources that takes into account both computation cost and data transmission cost. The optimization process uses two components: the scheduling heuristic that assign all the "ready" tasks, and the PSO steps for task-resource mapping optimization. The algorithm updates the communication costs, based on average communication time between resources, in every scheduling loop. It also re-computes the task-resource mapping so that it optimizes the cost of computation, based on the current network and resource conditions.

BAR – Balance Reduce [17]. The authors focus on data locality of systems where workflows are split into many small blocks, and all blocks are replicated over several servers. To process workflow efficiently, each job is divided into many tasks, and each task is allocated to a server to deal with a workflow block. They propose heuristic task scheduling algorithm called BAlance-Reduce (BAR), in which, firstly, an initial task allocation is produced, then, the job completion time is reduced by tuning the initial task allocation. They define the cost of a task as the sum of the execution time and the input data transferring time. The authors conclude that it is hard to obtain a near-optimal solution when the remote cost changes frequently. So, in this context, in a poor network environment, BAR tries its best to enhance data locality.

SHEFT - Scalable-Heterogeneous-Earliest-Finish-Time algorithm [16]. The authors propose SHEFT algorithm that used a priority list to address the scheduling problem where the number of resources cannot be automatically determined on demand by the size of a workflow, and the resources assigned to a workflow usually are not released until the workflow completes an execution. They assume that resources within one cluster usually share the same network communication and have the same data transfer rate. The model accommodates both heterogeneous and homogeneous computing environments in terms of computing capability and data communication.

DBA – Distributing Bartering Algorithm [18]. The authors proposed a decentralized affinity-aware migration technique that incorporates heterogeneity and dynamism in network topology and job communication patterns to allocate virtual machines on the available physical resources. Their technique monitors network affinity between pairs of virtual machines and uses a distributed bartering algorithm coupled with migration to dynamically adjust virtual machine placement such that communication overhead is minimized.

DENS – Data center Energy-efficient Network-aware Scheduling [15]. The authors present a scheduling methodology that combines energy efficiency and network awareness. Their approach is to receive and analyze a runtime feedback from the data center switches and links as well as take decisions and actions based on the network feedback. The goal is to achieve the balance between individual job performances, QoS requirements, traffic demands and energy consumed by the data center. They select the computing resource for job execution based on the load level and communication potential of data center components. Their methodology is relevant in data centers running data-intensive jobs which require low computational load, but produce heavy data streams.

NS2 - Network Simulator 2 [22]. The authors use a Grid model as a set of interconnected geographically dispersed Grid sites. At each site, one or more resources are presented. The status and properties of these resources are stored in an information service repository. Based on this information and the network management, they calculate job execution speed and end time.

Table 2. Main characteristics of related work's algorithms

Algorithm	Centralized	Decentralized	Hierarchical	On line	Off line	Clairvoyant	Nonclairvoyant	QoS	Migration cost	Network delay	Static	Dynamic	Communication cost	Migration
mCAGA	•				•				•	•				
CCTS	•				•					•	•	•		
PSO					•	•						•	•	
BAR	•		•		•	•					•		•	
SHEFT			•	•			•			•		•	•	
DBA	•		•	•			•		•	•		•		
DENS	•		•	•			•	•		•		•		
NS2	•				•	•			•	•	•			
NAMC		•	•	•		•				•		•		•
CA-DAG	•			•			•			•		•		

Table 3. Criteria for evaluation of algorithms

Algorithm	Utilization	Performance	Overhead	Response time	Scalability	Throughput	Energy	Makespan
mCAGA			•					•
CCTS								•
PSO	•						•	
BAR								•
SHEFT								•
DBA		•	•					
DENS		•	•					
NS2		•		•		•		
NAMC		•			•	•		
CA-DAG								•

NAMC - Network-Aware Migration Control [22]. The authors introduced a network topology aware scheduling model for VM live migrations. They take network bandwidth requirements of migrations and network topologies into account. They execute migrations over a dedicated bandwidth limited network link, and try to minimize the maximum bandwidth usage at each point in time while holding migrations deadlines. During a migration, the bandwidth usage is increased up to a user defined maximum limit.

To achieve this, the communication network is presented by a new topology graph for the representation static and dynamic networks of arbitrary, possibly heterogeneous structure.

4 Problem Definition

4.1 Job

We consider n independent jobs $J_1, J_2, \ldots, J_n$ that must be scheduled on federation of clouds. The job J_j is described by a tuple $\{r_j, G_j\}$ that consists of job release time $r_j \geq 0$, and the program represented by a directed acyclic graph $G = (V, E, \omega, \varphi)$. We use CA-DAG model of applications, where the set of vertices $V = \{V_c, V_{comm}\}$ is composed of two non-overlapping subsets V_c and V_{comm}. The set $V_c \subseteq V$ represents computing tasks, and the set $V_{comm} \subset V$ represents communication tasks of the program.

The set of edges E consists of directed edges e_{ij} representing dependence between node $v_i \in V$, and node $v_j \in V$, meaning that a task v_j relies on the input from the task v_i, and v_j cannot be started until this input is received. A particular case is when the size of this input is zero. It helps to define the execution order of tasks, which exchange no data.

The main difference between communication vertices V_{comm} and edges E is that V_{comm} represents communication tasks occurred in the network, making them a subject to communication contention, significant delay, and link errors. Edges E represent the results of exchange between tasks considered to be executed on the same physical server. Such communications often involve processor caches. They are fast and the associated delay is multiple orders of magnitude lower than the delay in a network and can be neglected. Consequently, the edge set E corresponds to the dependences between computing and communication tasks defining the order of their execution.

The release time of a job is not available before the job is submitted, and its processing time is unknown until the job has completed its execution.

4.2 Cloud Infrastructure

We address scheduling problem in the hierarchical federated cloud environment, with k independent clouds $C_1, C_2, \ldots, C_k$ and local provider cloud C.

Cloud C consists of m nodes (data centers) $D_1, D_2, \ldots, D_m$. Each node D_i consists of b_i servers (blades, boards) and p_i processors for all $i = 1..m$. We assume that

processors are identical and have the same number of cores. Let m_i be the number of identical cores of one processor in D_i.

We denote the total number of cores belonging to the data center D_i by $\bar{m}_i = b_i \cdot p_i \cdot m_i$, and belonging to all data centers in the cloud C by $\bar{m} = \sum_{i=1}^{m} \bar{m}_i$.

The processor of data center D_i is described by a tuple $\{m_i, s_i, mem_i, band_i, eff_i\}$, where s_i is a measure of instruction execution speed (MIPS), mem_i is the amount of memory (MB), $band_i$ is the available bandwidth (Mbps), and eff_i is energy efficiency (MIPS per watt). We assume that data center processors have enough resources to execute any job, but these resources are not infinite. Due to virtualization and resource sharing the amount of available resources are constantly changes, which causes uncertainty in job assignment.

A job can be allocated to one cloud only, while replication of jobs is not considered. Jobs submitted to one cloud can be migrated to another one. The admissible set of data centers for a job J_j is defined as a set of indexes $\left\{a_1^j, \ldots, a_l^j\right\}$ of data centers that can be used for migration of the job J_i. A data center contains a set of routers and switches that transport traffic between computing servers and to the outside world. They are characterized by the amount of traffic flowing through them (Mbps). A switch connects to other switches or computational nodes. The interconnection of processors is static, but their utilization changes. An overload can occur due to a large amount of I/O being pushed through them. The interconnection network architecture is three-tier fat tree, the most common nowadays. It consists of the access, aggregation, and core layers. The interconnection between clouds is done through public Internet.

5 Communication Models

We distinguish three models based on the standard DAG model: communication-unaware (CU-DAG), edges-based (EB-DAG), and communication-aware model (CA-DAG). As mentioned before, the vertices of the EB-DAG represent both computing and communication requirements of a task. In the CU-DAG, edges are associated with the communications performed by tasks. In CA-DAG, two types of vertices are considered: one for computing and one for communications. Edges define dependences between tasks and order of execution.

Let us briefly discuss their main properties. For more details, see [11].

Communication-Unaware Model (CU-DAG). The representation of computing and communication demands of a task as a single vertex (see Fig. 1) makes it almost impossible to produce an efficient schedule. Let us consider a computing task that requires information from a database as an input. The delay of sending and handling a database query as well as receiving a reply can be significant. During this time the computing work, being scheduled for execution, stays on hold waiting for input data, even if a part of it does not depend on waited data. For the example presented in Fig. 1, we are able to schedule T2 and T3 in parallel or share a single core in time, i.e., perform computing for the T2, while T3 waits for the input, and process T3, while T2 is sending its output.

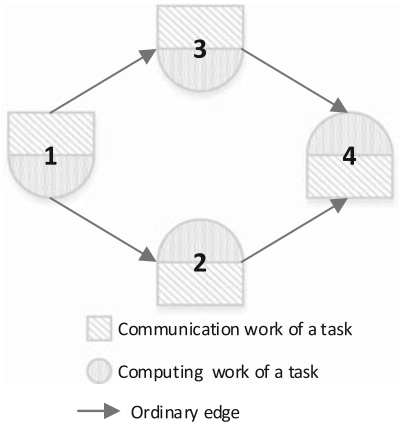

Fig. 1. CU-DAG model

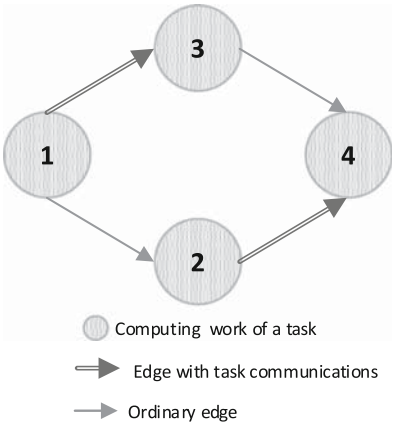

Fig. 2. EB-DAG model

However, a precise knowledge of the communication patterns of both tasks should be available. There is another shortcoming of the reviewed model. Suppose T2 computes data, and (a) sends them to the network for the database update (represented by a grey segment of the vertex), and (b) feeds them as an input to T4. With CU-DAG, T4 will wait for the successful completion of T2 including database update. On the other hand, T4 could be started in parallel to the database update. It would be logical to separate these two fundamentally different activities and schedule them separately for an efficient execution.

Edge-Based Model (EB-DAG). In this model, the DAG is defined as a directed acyclic graph $G = (V, E, w, c)$, where vertices V represent computing tasks, and a set of edges E describes communications between tasks. $w(n)$ is a computation cost of a node $n \in V$, and $c(e_{ij})$ is the communication cost of the link $e_{ij} \in E$. Task scheduling implies mapping tasks V on a set of processors specifying starting time, and duration for each task.

One significant drawback is that it prevents two different computing tasks from using the same data transfer to receive an input. Let us consider T2 and T3, Fig. 2. Suppose that the tasks require the same data from the database to start their execution. In practice, it can be done with a single database query, which implies a single edge of the graph. However, a single edge cannot lead to two different vertices. As a result, either two different edges trigger two different queries, or an empty vertex needs to be added as a mean to branch a DAG edge.

Another shortcoming of this model is in edge scheduling. To schedule communications, the DAG edges are mapped to the network links represented by the topology graph of the network. The topology graph is assumed to contain accurate information on network nodes, connections between them, and data transfer rates of all of the links. Accurate knowledge of the actual bandwidth is mainly inaccessible. This is due to the diverse nature of the network traffic that is produced at different layers of the protocol stack and mixed in the communication links and network routers.

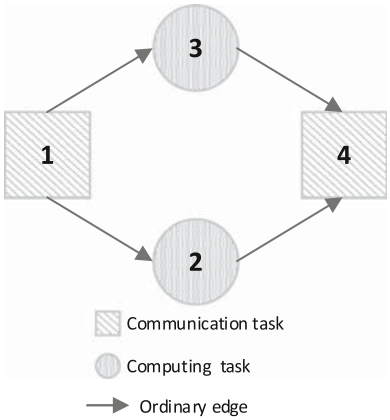

Fig. 3. CA-DAG communication model

Communication-Aware Model (CA-DAG). CA-DAG could solve some shortcomings of previous models, see Fig. 3. The main advantage of this model is that it allows separate resource allocation decisions, assigning processors to handle computing jobs and network resources for information transmissions.

5.1 Scheduling Example

Let us consider a typical cloud webmail application. On a highly abstract level its operation can be represented with the following four steps [11]:

```
Step 1: Receive user request and process it.
Step 2: Generate personalized advertisement.
Step 3: Request list of email messages from database.
Step 4: Generate HTML page and send it to the user.
```

In Fig. 4, the DAG vertices related to the computing tasks are represented by circles, while the communication related vertices are shown using square shapes.

Let us consider scheduling of nine tasks T0–T8 on identical computers to optimize total execution time:

```
T0: processes the arrival of user request
T1: identifies a user, and prepares a database query
T2: analyses user profile to determine targeted advertisement
T3: retrieves personalized advertisement from the database
T4: queries database for the list of user email messages
T5: prepares a list of email messages
T6: groups messages into conversations
T7: combines outputs of T3, T5, T6, and generates HTML page
T8: sends output to user
```

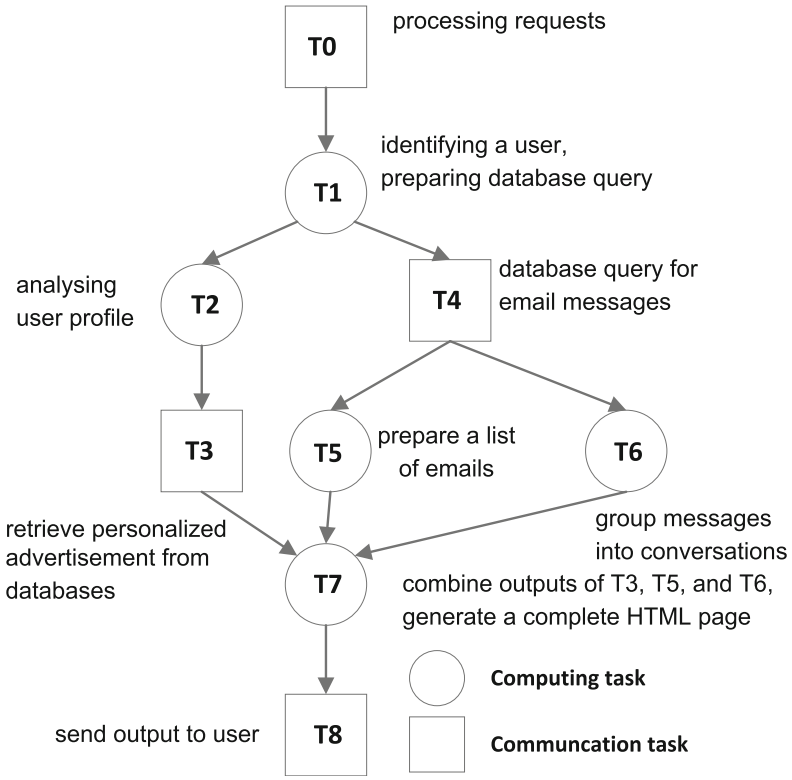

Fig. 4. Example of CA-DAG [11]

Computing resources are represented by two processors of a data center p_1 and p_2. The communication resources are represented with network links l_1 and l_2 interconnecting computing resources and database DB.

Figure 5 shows several possible schedules using these representations for processors and communication links: (a) CA-DAG, p_1, l_1; (b) CU-DAG, p_1, l_1; (c) CU-DAG, p_1, p_2, l_1; (d) EB-DAG, p_1, p_2, l_1; (e) EB-DAG, p_1, l_1, l_2.

Figure 5(a) shows a Gantt chart schedule for the CA-DAG. Computing tasks T1, T2, T5, T6, and T7 are scheduled on the processor p_1, while communication-related tasks T0, T3, T4, and T8 are scheduled at the network link l_1.

Representing communication tasks T0, T3, T4, T8 by vertices allows us to allocate them to the network resources independently from the processor unit. The processor time is not wasted by waiting for communications to be completed.

T4 is executed simultaneously with the analysis of a user profile T2, the list of email messages T5 can be generated, while database is being queried for a personalized advertisement T3. Such a scheduling flexibility is unavailable when communication work is seen as a part of a task description.

Figure 5(b) presents a schedule for CU-DAG. The inability to control allocation of network resources and distinguish the size of task communications, results in a larger makespan. The processor has to wait for finishing communications before it can start the computational portion of the task. To match the makespan of the CA-DAG, an additional processing unit would be required (see Fig. 5(c)).

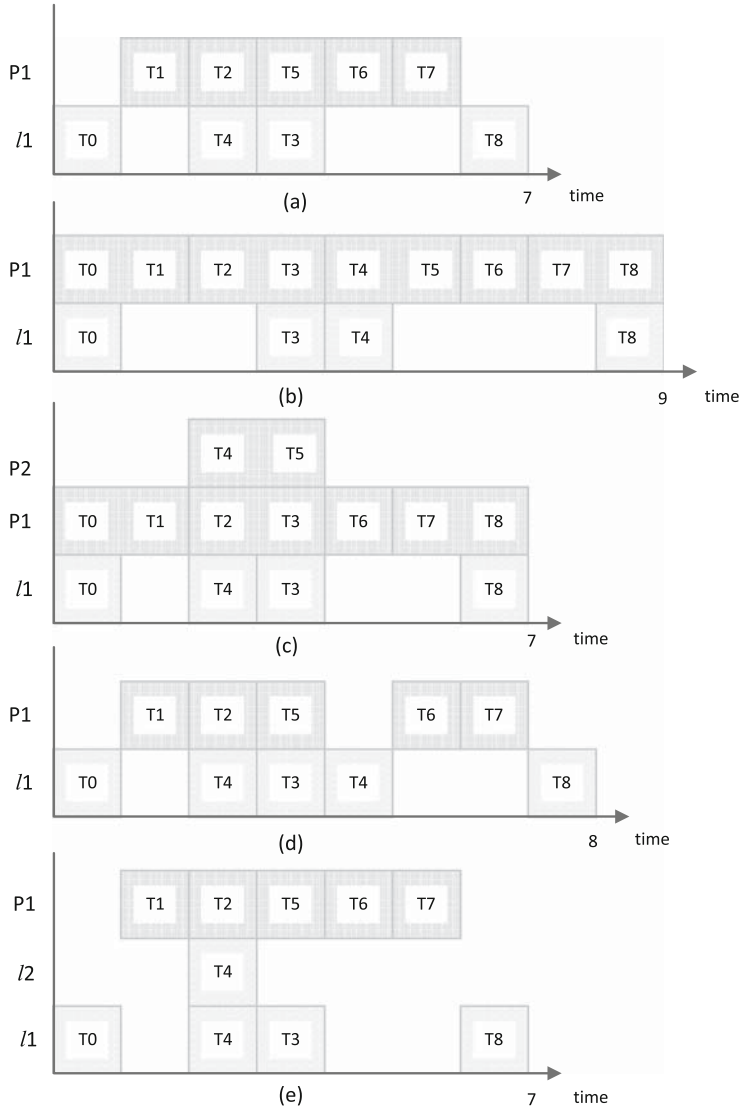

Fig. 5. Gant charts: (a) communication-aware CA-DAG model, (b) communication-unaware CU-DAG model and one processor, (c) communication-unaware CU-DAG model and two processors, (d) edges-based communication EB-DAG model and network link, (e) edges-based communication EB-DAG model and two network links

The EB-DAG with edges to model communication processes cannot model certain required communication types. In our example, for instance, T4 is the edge that represents the database request. It is impossible that this single edge leads to two different computing tasks T5 and T6. An additional edge has to duplicate the communication effort.

Figure 5(d) shows other EB-DAG scheduling. It requires scheduling two copies of the edge T4 leading from T1 to T5 and T6. To produce a schedule with makespan equals to the CA-DAG schedule, an additional network link is needed, so that both copies of the edge T4 can be scheduled in parallel (Fig. 5(e)).

5.2 Scheduling Approach

We use a two-level scheduling approach as shown in Fig. 6.

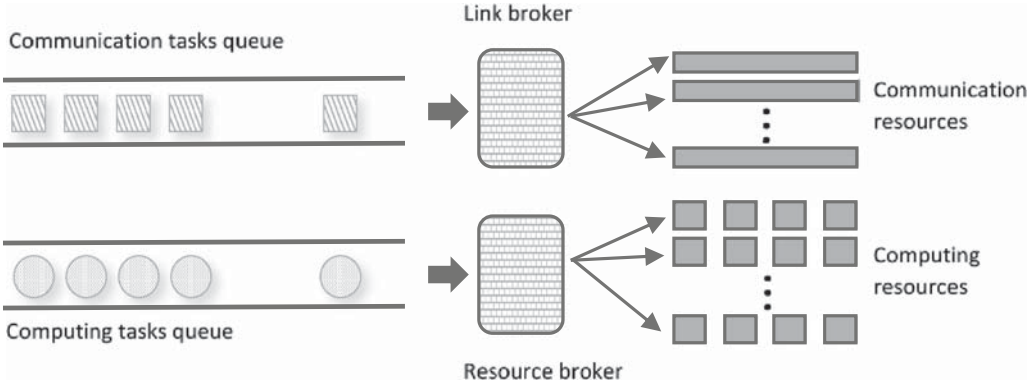

Fig. 6. General diagram of the scheduling approach

At the upper level, two lists of tasks that are ready to be started are maintained. Independent computing tasks with no predecessors, with predecessors that completed their execution, and all available data are entered into the computing tasks queue. Upon completion of the computing task its immediate successors may become available and enter into the computing list, and communication tasks enter into communication task queue. Communication tasks can also start their execution only after all their dependencies have been satisfied.

5.3 Lower Level Allocation Strategies

On the second level, allocation policies of the brokers are responsible for selecting suitable computing servers, also called machines in the rest of the paper, and communication link (path) for a task allocation. These computing- and communication-aware resource allocation policies can be characterized by the type and amount of information used for allocation decision, and optimization criteria.

Schedulers can target several different objectives, including response time, Quality of Service (QoS), energy and cooling efficiency, and a combination of them. To this end, schedulers should use multi-criteria decision support. Especially, it is important in scenarios that contain aspects which are multi-objective by nature, for instance, system performance related issues and user QoS demands.

One set of criteria may correspond to the optimization of the load of computing servers. Another one could be oriented on data traffic and/or network optimization. They are interdependent and may have conflicting goals due to computing resource allocation, that has a big impact on data traffic, and data traffic on the computing resource optimization.

Important criteria are related with energy, heat, and cooling optimization. Cloud resource providers and users also have conflicting performance goals: from minimizing response time to optimizing user and provider cost optimization.

Workflow scheduling has diversified into many research directions: analysis of workflow structure properties in order to identify tasks clusters; minimization of critical path execution time; selection of admissible resources; allocation of suitable resources for data intensive workflows; scheduling subject to QoS constraints, fine tuning workflow execution and performance analysis, etc.

The typical examples of workflow scheduling strategies used in many performance evaluation studies are: HEFT (Heterogeneous Earliest Finishing Time First), and CPOP (Critical Path on Processor). However, DAG scheduling algorithms designed for static DAGs and static machine settings with full information are not suitable for cloud scheduling contexts [9, 27]. The information about local schedules traditionally used for workflow allocation, task runtime, length of the critical path, amount of transmission data, machine speed, communication bandwidth, etc. is not accurate and does not help to improve the outcome of the workflow scheduling strategies.

Other typical examples of computing task allocation strategies include, just to name a few: MLp allocates a task to the machine with least load per core, MLB allocates a task to the machine with least computational work per core, MST allocates a task to the machine with earliest start time, MCT allocates task to the machine with earliest task completion time, etc. [13, 29].

MTT allocates communication task to the link with earliest transmission time, MaxB allocates communication task to the link with highest bandwidth, LL allocates communication task to the link to balance transmission load, etc.

They can be categorized as: knowledge-free, with no information about tasks and resources; energy-aware, with power consumption information; speed-aware with speed of machines information; bandwidth-aware, with actual bandwidth information, etc.

CA-DAG is based on separate vertices for computing and communications. Such a dual computing-communication task representation allows making separate resource allocation decisions to handle computing jobs and network resources for information transmissions. It mitigates uncertainty in more efficient way, making decisions based on the current or predicted information, and dynamically adapt them to cope with different objective preferences, workloads, and cloud properties.

The proposed communication-aware model creates space for optimization of many existing solutions to resource allocation as well as developing completely new scheduling schemes of improved efficiency.

6 Conclusions

In this paper, we discuss a problem of scheduling workflow applications in clouds under uncertainty. We present an overview of the job scheduling under a dynamic context of cloud computing using three models of cloud applications with communication CU-DAG, EB-DAG and CA-DAG. We show that communication-aware model CA-DAG has advantages under certain conditions.

However, further study is required to assess its actual efficiency and effectiveness. This will be subject of future work requiring a better understanding of workflow scheduling on real data with presence of uncertainty. We design and implement data intensive workflow scheduling strategies considering a cloud model. We study CA-DAG

and scheduling strategies addressing optimization of response time and energy consumption. We will conduct a comprehensive performance evaluation study using real logs to demonstrate that CA-DAG performs well with respect to several metrics that reflect both user- and system-centric goals.

Acknowledgment. This work is partially supported by CONACYT (Consejo Nacional de Ciencia y Tecnología, México), grant no. 178415. The work of D. Dzmitry Kliazovich is partly funded by National Research Fund, Luxembourg in the framework of ECO-CLOUD (C12/IS/3977641) project.

References

1. Robison, S.: HP Shane Robison Executive Viewpoint: The Next Wave: Everything as a Service. http://www.hp.com/hpinfo/execteam/articles/robison. Accessed 30 January 2014
2. CSC: CSC cloud usage index latest report, Computer Sciences Corporation. http://www.csc.com/au/ds/39454/75790-csc_cloud_usage_index_latest_report. Accessed 20 January 2014
3. Zhang, Q., Cheng, L., Boutaba, R.: Cloud computing: state-of-the-art and research challenges. J. Internet Serv. Appl. **1**(1), 7–18 (2010)
4. N. US Department of Commerce, Final Version of NIST Cloud Computing Definition Published. http://www.nist.gov/itl/csd/cloud-102511.cfm. Accessed 20 January 2014
5. Hollinsworth, D.: The workflow reference model. In: Workflow Management Coalition, vol. TC00–1003 (1995)
6. Pandey, S., Wu, L., Guru, S.M., Buyya, R.: 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), pp. 400–407 (2010)
7. Kandula, S., Sengupta, S., Greenberg, A., Patel, P., Chaiken, R.: The nature of data center traffic: measurements & analysis. In: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference, New York, NY, USA, pp. 202–208 (2009)
8. AbdelBaky, M., Parashar, M., Kim, H., Jordan, K.E., Sachdeva, V., Sexton, J., Jamjoom, H., Shae, Z.Y., Pencheva, G., Tavakoli, T., Wheeler, M.F.: Enabling high-performance computing as a service. Computer **45**(10), 72–80 (2012)
9. Tchernykh, A., Schwiegelsohn, U., Alexandrov, V., Talbi, E.: Towards understanding uncertainty in cloud computing resource provisioning. SPU 2015 - solving problems with uncertainties (3rd Workshop). In: Conjunction with the 15th International Conference on Computational Science (ICCS 2015), Reykjavík, Iceland, 1–3 June 2015. Procedia Computer Science, Elsevier, vol. 51, pp. 1772–1781 (2015)
10. Tychinsky A.: Innovation Management of Companies: Modern Approaches, Algorithms, Experience. Taganrog Institute of Technology, Taganrog (2006). http://www.aup.ru/books/m87/
11. Kliazovich, D., Pecero, J., Tchernykh, A., Bouvry, P., Khan, S., Zomaya, A.: CA-DAG: modeling communication-aware applications for scheduling in cloud computing. J. Grid Comput., 1–17 (2015). Springer, Netherlands
12. Tsafrir, D., Etsion, Y., Feitelson, D.G.: Backfilling using system-generated predictions rather than user runtime estimates. IEEE Trans. Parallel Distrib. Syst. **18**(6), 789–803 (2007)

13. Ramírez-Alcaraz, J.M., Tchernykh, A., Yahyapour, R., Schwiegelshohn, U., Quezada-Pina, A., González-García, J.L., Hirales-Carbajal, A.: Job allocation strategies with user run time estimates for online scheduling in hierarchical Grids. J. Grid Comput. **9**(1), 95–116 (2011)
14. Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications, vol. 63. Shaker, Ithaca (1999)
15. Kliazovich, D., Bouvry, P., Khan, S.U.: DENS: data center energy-efficient network-aware scheduling. Cluster Comput. **16**(1), 65–75 (2013)
16. Liu, K., Jin, H., Chen, J., Liu, X., Yuan, D., Yang, Y.: A compromised-time-cost scheduling algorithm in swindew-c for instance-intensive cost-constrained workflows on a cloud computing platform. Int. J. High Perform. Comput. Appl. **24**(4), 445–456 (2010)
17. Jin, J., Luo, J., Song, A., Dong, F., Xiong, R.: BAR: an efficient data locality driven task scheduling algorithm for cloud computing. In: 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2011), pp. 295–304 (2011)
18. Sonnek, J., Greensky, J., Reutiman, R., Chandra, A.: Starling: minimizing communication overhead in virtualized computing platforms using decentralized affinity-aware migration. In: 39th International Conference on Parallel Processing (ICPP 2010), pp. 228–237 (2010)
19. Pecero, J.E., Trystram, D., Zomaya, A.Y.: A new genetic algorithm for scheduling for large communication delays. In: Sips, H., Epema, D., Lin, H.-X. (eds.) Euro-Par 2009. LNCS, vol. 5704, pp. 241–252. Springer, Heidelberg (2009)
20. Stage, A., Setzer, T.: Network-aware migration control and scheduling of differentiated virtual machine workloads. In: Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, pp. 9–14. IEEE Computer Society (2009)
21. Sinnen, O., Sousa, L.A.: Communication contention in task scheduling. IEEE Trans. Parallel Distrib. Syst. **16**(6), 503–515 (2005)
22. Volckaert, B., Thysebaert, P., De Leenheer, M., De Turck, F., Dhoedt, B., Demeester, P.: Network aware scheduling in grids. In: Proceedings of the 9th European Conference on Networks and Optical Communifications, p. 9 (2004)
23. Malawski, M., Juve, G., Deelman, E., Nabrzyski, J.: 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, p. 22 (2012)
24. Yu, J., Buyya, R.: A taxonomy of workflow management systems for grid computing. J. Grid Comput. **3**(3–4), 171–200 (2005)
25. Tchernykh, A., Pecero, J., Barrondo, A., Schaeffer, E.: Adaptive energy efficient scheduling in peer-to-peer desktop grids. Future Gener. Comput. Systems **36**, 209–220 (2014)
26. Tchernykh, A., Lozano, L., Schwiegelshohn, U., Bouvry, P., Pecero, J.E., Nesmachnow, S., Drozdov, A.Y.: Online bi-objective scheduling for IaaS clouds with ensuring quality of service. J. Grid Comput., 1–18 (2015). Springer
27. Carbajal, A.H., Tchernykh, A., Yahyapour, R., Röblitz, T., Ramírez-Alcaraz, J.M., González-García, J.L.: Multiple workflow scheduling strategies with user run time estimates on a grid. J. Grid Comput. **10**(2), 325–346 (2012). Springer-Verlag, New York, USA
28. Quezada, A., Tchernykh, A., González, J., Hirales, A., Ramírez, J.-M., Schwiegelshohn, U., Yahyapour, R., Miranda, V.: Adaptive parallel job scheduling with resource admissible allocation on two level hierarchical grids. Future Gener. Comput. Syst. **28**(7), 965–976 (2012)
29. Rodriguez, A., Tchernykh, A., Ecker, K.: Algorithms for dynamic scheduling of unit execution time tasks. Eur. J. Oper. Res. **146**(2), 403–416 (2003). Elsevier Science, North-Holland

30. Kianpisheh, S., Jalili, S., Charkari, N.M.: Predicting job wait time in grid environment by applying machine learning methods on historical information. Int. J. Grid Distrib. Comput. 5(3) (2012)
31. Iverson, M.A., Ozguner, F.; Follen, G.J.: Run-time statistical estimation of task execution times for heterogeneous distributed computing. In: Proceedings of 5th IEEE International Symposium on High Performance Distributed Computing, 1996, pp. 263–270 (1996)
32. Ramirez-Velarde, R.V., Rodriguez-Dagnino, R.M.: From commodity computers to high-performance environments: scalability analysis using self-similarity, large deviations and heavy-tails. Concurrency Comput. Pract. Exp. **22**, 1494–1515 (2010)