J Grid Computing
DOI 10.1007/s10723-015-9337-8

CA-DAG: Modeling Communication-Aware Applications for Scheduling in Cloud Computing

Dzmitry Kliazovich · Johnatan E. Pecero · Andrei Tchernykh · Pascal Bouvry · Samee U. Khan · Albert Y. Zomaya

Received: 6 December 2014 / Accepted: 19 April 2015

Abstract This paper addresses performance issues of resource allocation in cloud computing. We review requirements of different cloud applications and identify the need of considering communication processes explicitly and equally to the computing tasks. Following this observation, we propose a new communication-aware model of cloud computing applications, called CA-DAG. This model is based on Directed Acyclic Graphs that in addition to computing vertices include separate vertices to represent communications. Such a representation allows making separate resource allocation decisions: assigning processors to handle computing jobs, and network resources for information transmissions. The proposed CA-DAG model creates space for optimization of a number of existing solutions to resource allocation and for developing novel scheduling schemes of improved efficiency.

D. Kliazovich (✉) · J. E. Pecero · P. Bouvry
University of Luxembourg, Luxembourg, Luxembourg
e-mail: dzmitry.kliazovich@uni.lu

A. Tchernykh
CICESE Research Center, Ensenada BC, México

S. U. Khan
North Dakota State University, Fargo ND, USA

A. Y. Zomaya
University of Sydney, Darlington, Australia

Keywords Cloud computing · Communication awareness · Resource allocation · Scheduling

1 Introduction

In recent years, parallel computers and clusters have been deployed to support computation-intensive and communication-intensive applications, and have become part of cloud computing. Such clouds are emerging as a new paradigm for providing services and solving large-scale problems in science, engineering, and commerce. Clouds comprise heterogeneous nodes (typically, clusters and parallel supercomputers) with a variety of computational resources. Clouds are becoming almost commonplace, with many projects using them for production services. The initial challenges of Cloud computing – how to provide a service, how to manage multiple virtual machines on different systems – have been resolved to the first degree. Therefore, researchers can now address the issues that will allow more efficient use of the resources. The use of cloud resource management is far from ubiquitous. This is due to the fact that scheduling and mapping decisions have to take into account the myriad standards, procedures, and devices in a highly dynamic environment. Consequently, resource management procedures must be able to adapt to changes in state and data communication requirements to meet their desired QoS constraints as traditional approaches to resource optimization become insufficient.

New job representations are also being developed for modeling problems in e-Commerce and e-Science as workflows. A workflow is the automation of a processes, which involves the orchestration of a set of services, agents, and actors that must be combined together to solve a problem or to define a new service. Workflows can be modeled as DAGs or other precedence constraint structures. Many precedence constraint problems have been studied in classical scheduling theory, and proved to be NP-hard. Approximation algorithms, linear programming, combinatorial, and stochastic solutions have been addressed for problem solution. Solutions for precedence constraint problems from classical scheduling theory are not suitable for cloud problems. This is due to the fact that they do not take into account the: (a) dynamic behavior of the execution context, (b) job mix workloads, or (c) uncertainty of the workflow properties. Of particular interest is to study: how resource unavailability and communication bandwidth dynamics trigger load balancing, and how they impact the workflow allocation plans under deterministic and nondeterministic scheduling paradigms.

The scheduling of jobs on multiprocessors is generally well understood and has been studied for decades. Many research results exist for different variations of the scheduling problem; some of them provide theoretical insights while others give hints for the implementation of real systems. However, the communication-aware scheduling problems that require the availability of communication resources have rarely been addressed. The communication properties are either completely ignored or highly generalized and weakly captured by current task models and scheduling approaches. Unfortunately, it may result in inefficient cloud infrastructure and communication media utilization.

In this paper, we define a model for cloud computing applications taking into account a variety of communication resources of various types used in real systems. This communication-aware model of cloud applications, called CA-DAG, 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.

The contribution synopsis of the paper is as follows.

1. Review of communication requirements of different cloud applications.
2. Motivation of communication awareness in resource allocation.
3. Definition of new communication-aware model of cloud applications.
4. Definition of properties of communication tasks.
5. Comparison of the proposed CA-DAG (communication-aware) model onto existing resource allocation solutions based on CU-DAG (communication-aware) and EB-DAG (Edge based) models.
6. Demonstration that separate resource allocation decisions to handle computing and communication jobs provide scheduling flexibility and, under certain conditions, reduce the requirement for network links and computational resources.

The rest of the paper is structured as follows: We introduce background on task scheduling in Section 2 and discuss main properties and requirements of cloud computing applications in Section 3. In Section 4, we introduce communication-aware DAG model and compare it with other DAG models. In Section 5, we present the properties of communication vertices of our model. Next, we discuss different aspects of communication-aware scheduling in Section 6. In Section 7, we present performance evaluation results to confirm the benefits of the proposed CA-DAG model. Finally, we conclude with a summary and an outlook in Section 8.

2 Background on Task Scheduling

A workflow is a composition of tasks with precedence constraints. Workflows are modeled by a directed acyclic graph $G_j = (V_j, E_j)$, where V_j is the set of tasks, and $E_j = \{(T_u, T_v) \mid T_u, T_v \in V_j,\ u \neq v\}$, with no cycles $T_u \rightsquigarrow T_v \rightsquigarrow T_u$ is the set of arcs between tasks in V_j. Each arc $(T_u, T_v) \in E_j$ represents the precedence constraint between tasks T_u and T_v, such that T_u must be completed prior to the initiation of the execution of T_v.

Arcs can be related to communication requirements of the underlying algorithm. The weights associated with the nodes and edges represent their computation costs and communication costs.

Tasks are released over time according to the precedence constraints. A task can start its execution only after all of the corresponding dependencies have been satisfied. At the release date, a task must be mapped to a resource. However, we do not demand that the specific resource is immediately assigned to a task at its release time. That is to state that the processor allocation of a task can be delayed. Scheduling problems involving precedence constraints are among the most studied problems in the domain.

In the homogeneous scheduling delay model, each arc represents the potential data transfer between tasks [1]. Communication system is considered here to be homogeneous and complete network. The LogP model presents a more detailed characterization of the communication delay. It is assumed that the communication delay consists of three parameters: (a) the latency, which is an upper limit on the time of transferring the message data in the communication medium, (b) the overhead of processing the message at the sender and the receiver communication, and (c) the interval in which the sender cannot send and the receiver cannot accept any new messages [2]. In the hierarchical communication model, the communication delays are not homogeneous. This is due to the fact that the processors are connected to clusters, and the communications inside a same cluster are faster than those between processors belonging to different ones.

In the classical scheduling, communication delays disappear if a predecessor task and a successor task are executed on the same processor [3]. This assumption is known as the locality assumption. The essential property of such models is that the task duplication avoids communication delays.

In the processor network communication model, the structure of the underlying network and the contention in accessing the shared medium are taken into account [4, 5]. However, it is assumed that communication channel is a resource equivalent to a processor.

Different variations of the scheduling problem with communication delays and classification of existing results are discussed in [6]. There are only few results available on scheduling that take into account the presence of large communication delays [7–9, 48, 49]. The most widely used approach is task clustering to balance communication delays and processing times. Mainly two classes of task clustering algorithms are studied based on the (a) critical path analysis [10] and (b) decomposition of the precedence task graph [8, 9, 11].

3 Cloud Applications

To understand how well the available scheduling solutions are applicable in the cloud computing environment, the main properties and requirements of cloud computing applications must be reviewed. Table 1 presents the classification of cloud computing applications according to the key factors determining their performance, namely: (a) computing load, (b) communication bandwidth requirement, (c) tolerance to high communication delays, (d) degree of interactivity, and (e) storage usage. More details on the cloud application requirements can be obtained from [12]. The applications are sorted top-to-bottom according to their average demand for resources.

Cloud gaming is probably the most cloud resource demanding application. Game execution, processing, and rendering are done on the cloud provider servers. The thin client at the user side simply receives and displays multimedia intensive content, which consumes a considerable amount of network bandwidth. The communication delay should be imperceptible to allow the fast reaction to the highly frequent user actions. The only noncritical demand by the cloud gaming resource is the storage that is required for the game play objects.

Videoconferencing and video streaming consume high bandwidth, but they differ in other requirements, such as videoconferencing requires a high computing power for signal processing and multiplexing, and low communication delays for imperceptible interaction. However, all of the above mentioned requirements are eased in video streaming making it similar to cloud storage and cloud backup services. Moreover, video conferencing does not impose storage requirements as it discards real-time video data after displaying. On the contrary, the availability of a very large storage space is a key requirement for video streaming, cloud storage, and cloud backup services.

Online office (e.g., Google Docs, Microsoft Office 365) and Customer Relation Management (CRM) services are mostly utilized by small and medium businesses as solutions. Both services are highly interactive, have moderate bandwidth requirements, moderate storage requirements, and cannot tolerate high communication delays. Collaborative editing is similar to online office applications. However, it imposes a tight communication delay constraints to allow document synchronization between collaborators in real-time.

Remote desktop services consume cloud computing and bandwidth resources moderately, but are highly interactive and require low communication delay to provide close to the typical desktop experience. They require storage resources intensively as all the data are streamed in real-time.

Cloud synchronization service stores data, such as music files, and keeps it synchronized among multiple devices, such as desktop computers, laptops, smart phones, or tablets. Even though, the user activity and variation of the content may be high, the cloud synchronization service operates periodically by aggregating the data modification requests. It operates well in networks with average delays, demands moderate amount of the computing, and requires medium bandwidth. However, it is sensitive to the availability of storage. Other storage-hungry services are video streaming, cloud storage, and cloud backup. The aforementioned services require high throughput, but produce almost no computational requirements and are tolerant to large communication delays. In contrast, voice conferencing, such as Skype, does not require computing, bandwidth resources, and storage. However, any increase in the communication delay degrades the quality of user experience greatly.

Social networking is a Web based service that facilitates people to post their profiles and establish social relations. Typically implemented using HTTP, it utilizes low bandwidth, and requires moderate computing power and storage requirements. It is also tolerant to moderate network delays with almost no requirements for live voice or video interactions.

The High Performance Computing (HPC) paradigm utilizes applications that are mostly of the scientific nature and are composed of highly computational intensive tasks. HPC applications are significantly different from other cloud computing services. Computing is the only resource critical for execution capability, while bandwidth, delay, and storage are insignificant, with the exception of a few data intensive applications, such as climate modeling. To run HPC applications effectively, the computing clouds have to be specifically designed or adapted to HPC as a service model [13, 14], as supercomputers and computer clusters still remain dominant for HPC.

4 Communication-Aware DAG Model

In this section, we motivate and define our CA-DAG model for communication intensive cloud applications and compare it with known CU-DAG (Communication-unaware) and EB-DAG (Edges-based) models.

4.1 Need for Communication-Awareness

Most of the cloud computing applications require the availability of communication resources for their operations. In Table 1, the surveyed cloud applications impose communication requirements in terms of the network bandwidth, delay, or both. The only exception is HPC, which is predominantly dependent on the computing power. 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. Other applications, such as voice conferencing, produce very light traffic load on the network, but require tight delay constraints, as imposed by the audio codec, and limits of human delay perception [15]. The cloud applications located in the top half of the Table 1, with cloud gaming and video conferencing being the leaders, impose tight constraints on both the network bandwidth and the delay.

The availability of communications 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 producing internal to the datacenter traffic which may account for as much as 75 % of the total traffic [16, 17].

Current models of cloud applications rely mostly on the HPC concepts [13, 14]. These models are based on DAGs that are formed of the collection of vertices, each representing a computing task, and directed edges, which show the relations between the tasks.

Table 1 Classification of cloud applications

Cloud application	Resource requirement				
	Computing	Bandwidth	Low communication delay	Degree of interactivity	Storage
Cloud gaming	H	H	H	H	L
Video conferencing	H	H	H	H	L
Online office	H	M	M	H	M
Collaborative editing	M	M	H	H	M
CRM	M	M	M	H	M
Remote desktop	M	M	H	H	L
Cloud Synchronization	M	M	M	M	H
Video streaming	L	H	L	L	H
Cloud storage	L	H	L	L	H
Cloud backup	L	H	L	L	H
Voice conferencing	L	L	H	H	L
Social networking	M	L	M	M	M
HPC	H	L	L	L	L

H: High, M: Medium and L: Low

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. Several researchers have realized this shortcoming and proposed adapting the standard DAG model by either allowing vertices to represent both computing and communication requirements of a task (communication-unaware model - CU-DAG) or associating edges with the communications performed tasks (edges-based model - EB-DAG). Both approaches have significant drawbacks that we detail below.

CU-DAG - Communication-Unaware Model Joining computing and communication demands of a task together, and representing them as a single vertex [18, 19], as represented in Fig. 1a, makes it almost impossible to schedule the task execution properly. 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 significantly beyond several milliseconds [20, 21], which is comparable with the time the search engines return results [22, 23]. During this time the computing work, being scheduled for execution, stays on hold waiting for input data. For the DAG example presented in Fig. 1a, we could ask how many processors or cores should be used to schedule tasks 2 and 3 in parallel? It may be enough to allocate a single core and share it in time, i.e. perform computing for the task 2, while task 3 waits for the input, and process task 3, while task 2 is sending its output. However, to answer this question properly, a precise knowledge of the communication patterns of both tasks should be available. There is another shortcoming of the reviewed model. Suppose task 2 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 the task 4. With such a DAG representation, task 4 will need to wait for the successful completion of the task 2 including database update. On the other hand, the task 4 could be started in parallel to the database update.

Summarizing, having a single vertex for representing both computing and communication of a task makes it difficult to properly schedule them: computing work at the servers and communication work in the network. It would be logical to separate these two fundamentally different activities and schedule them separately for an efficient execution.

EB-DAG - Edges-Based Model Associating DAG edges with task communications [4, 24, 25] is an attempt to treat communication and computing works

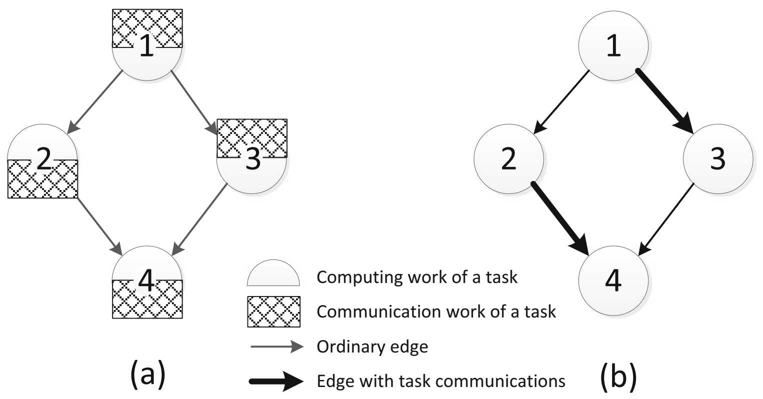

Fig. 1 Modeling communications DAGs: **a** CU-DAG (communication-unaware) and **b** EB-DAG (edges-based)

differently. 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 $c(e_{ij})$ denotes 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.

The aforementioned representation of communication processes with DAG edges has one significant drawback. It prevents two different computing tasks from using the same data transfer to receive an input. Consider tasks 2 and 3, in Fig. 1b. Suppose the tasks require the same data object 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 the processing of edge scheduling. To schedule communications, the DAG edges E are mapped to the network links represented by the topology graph of the network [4]. The topology graph is assumed to contain accurate information on network nodes, connections between them, and data transfer rates of all of the links. Even if the connectivity information may be available for the network, accurate knowledge of the available network capacity remains mainly inaccessible [26]. 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. Part of the network traffic is broadcasted and not accounted for by the edge scheduling. For example, it is common in Address Resolution Protocol (ARP) [27], which is used to find the correspondence between IP and MAC address every time a node communicates with a new destination, or in Internet Control Message Protocol (ICMP) messages [28], which are often generated by the routers in repose to routing failures or congestion problems [29]. As a consequence, knowing capacities of the links helps to estimate the upper bound of the achievable transmission rate, but what remains available to the edge scheduler is commonly referred as available bandwidth [30]. Estimating the available bandwidth has been a hot research topic for a number of years with many solutions proposed [26]. However, it is widely accepted to be difficult or even impossible to accurately estimate it, partially due to the requirement to use active probing of network links [31] and a delay between the moment a probe senses network traffic and the time the measurement becomes available when the probe is returned.

4.2 Communications-aware DAG Model

In this section, we propose new Communication-Aware DAG (CA-DAG) model to overcome limitations of the classical DAG representations, discussed in the previous sections, for cloud computing applications.

Definition of CA-DAG Model The program is represented by a directed acyclic graph $G = (V, E, \omega, \varphi)$. The set of vertices $V = \{V_c, V_{comm}$ is composed of two non-overlapping subsets V_c and V_{comm}. The set $\subseteq V_c V$ represents computing tasks, and the set $\subset V_{comm} V$ represents communication tasks of the program.

A computing task $v_i^c \in V_c$ is described by a pair (I, D_c) with the number of instructions I(amount of work) that has to be executed within a specific deadline D_c. A communication task $v_i^{comm} \in V_{comm}$ is described by parameters (S, D_{comm}), and defined as the amount of information S in bits that has to be successfully transmitted within a predefined deadline D_{comm}. Positive weights $\omega\left(v_i^c\right)$ and $\varphi(v_i^{comm})$ represent the cost of computing at the node $v_i^c \in V_c$, and cost of communication at the node $v_i^{comm} \in V_{comm}$, respectively.

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.

Representative Example Consider a typical cloud computing application of webmail. On a highly abstract level its operation can be represented with the following four steps:

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

Each of the aforementioned steps involves a communication process, and can be represented by the communication-aware DAG.

In Fig. 2, the DAG vertices related to the computing tasks V_c are represented by circles, while the communication related vertices V_{comm} are shown using square shapes. Task 0 is associated with the arrival of user request and its delivery to computing resources over the data center network. Task 1 processes the request, identifies a user, and prepares a database query. Task 2 analyses user profile to determine traits for targeted advertisement. During the execution of Task 3 the requested personalized advertisement is obtained from the database.

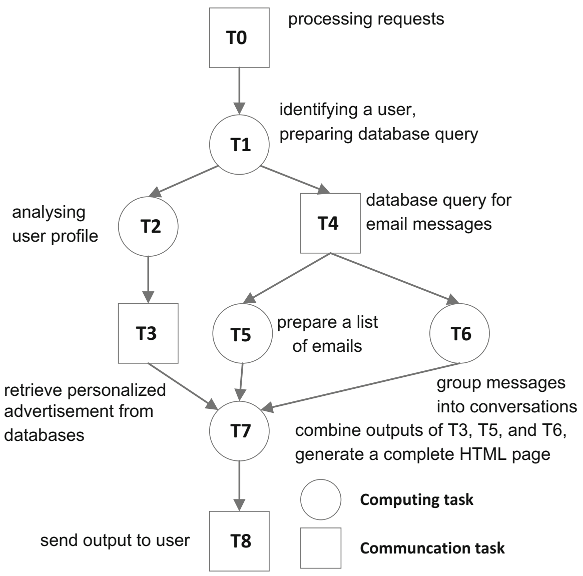
T0
processing requests
T1
identifying a user, preparing database query
T2
analysing user profile
T4
database query for email messages
T3
T5
prepare a list of emails
T6
retrieve personalized advertisement from databases
group messages into conversations
T7
combine outputs of T3, T5, and T6, generate a complete HTML page
T8
send output to user
Computing task
Communcation task

Fig. 2 CA- DAG model of cloud mail appliocation

In Task 4, the database is queried for the list of user email messages. When the reply is received, it is fed into Task 5 and Task 6 running parallel. Task 5 prepares a list of email messages, while Task 6 determines which messages can be grouped into conversations.

Finally, Task 7 combines the outputs of Task 3, Task 5, and Task 6, and generates a complete HTML page, which is sent to the user in Task 8.

Comparison of Models Let us consider a scheduling of tasks with communications on a set of identical computers to optimize the total execution time (makespan). Computing resources are represented by two processors of a data center p1 and p2 (see Fig. 3). The communication resources are represented with network links l1 and l2 interconnecting computing resources and database DB. Now let us see how the described webmail application can be represented by three types of DAGs: CU-DAG (Fig. 1a), EB-DAG (Fig. 1b), and CA- DAG (Fig. 2).

Figure 4a shows a possible schedule for the CA-DAG. Computing Tasks 1, 2, 5, 6, and 7 are scheduled

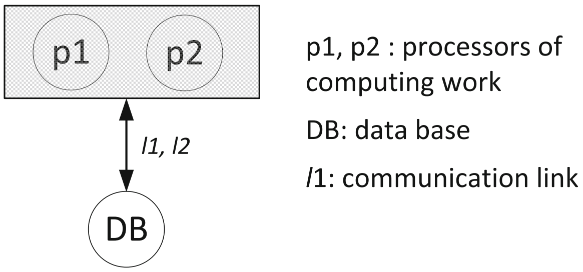

Fig. 3 Example of infrastructure for scheduling

on the processor p1, while communication-related Tasks 0, 3, 4, and 8 are scheduled at the network link l1. Representing communication tasks with their own distinct vertices allows us to control an allocation and execution time at the network resources in addition to the processor unit. The processor time is not wasted by waiting for communications to complete. For example, a data base query (Task 4) is executed simultaneously with the analysis of a user profile (Task 2), while at the next step, the list of email messages (Task 5) can be generated, while database is being queried for a personalized advertisement (Task 3). Such a scheduling flexibility is not available when communication work is seen as a part of a task description.

For the purpose of comparison, Fig. 4b presents a schedule for CU-DAG, depicted in Fig. 1a. The inability to control allocation of network resources and

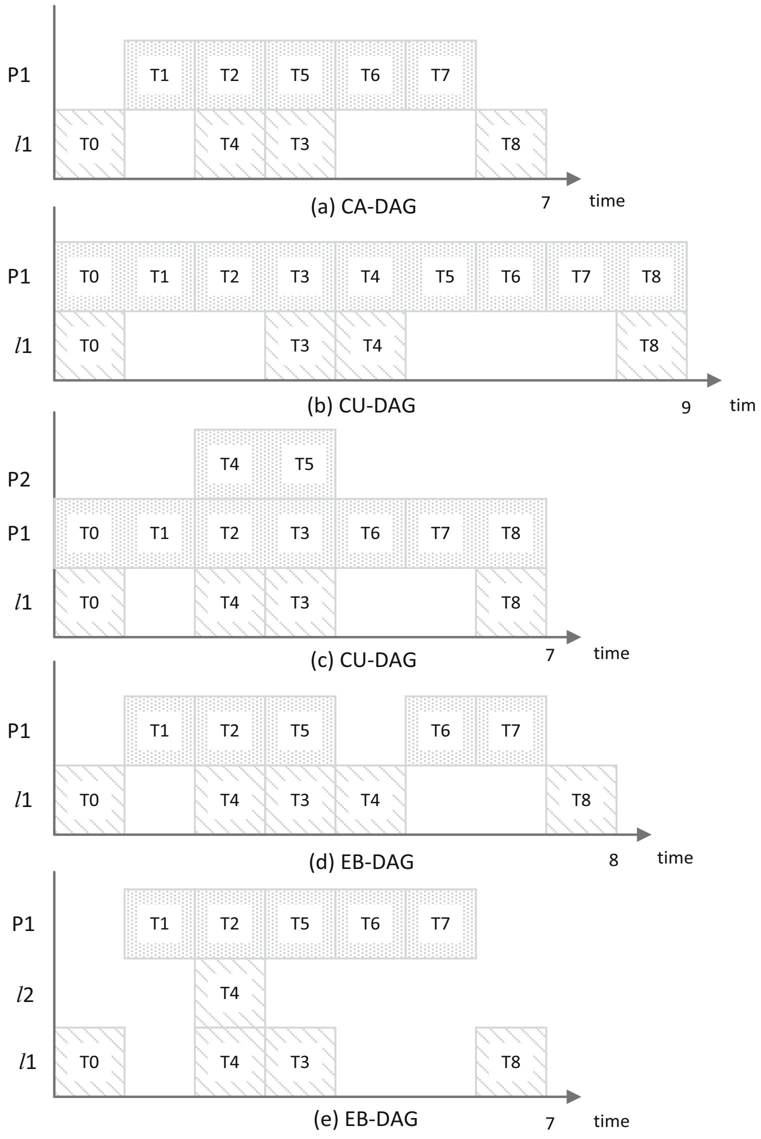

Fig. 4 Schedules for: **a** CA-DAG model, **b** CU-DAG model and one processor, **c** CU- DAG model and two processor, **d** EB-DAG model and one network link, and **e** EB-DAG model and one network link

distinguish the size of task communications, results in a larger makespan. The processor is often forced 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. 4c).

The DAGs that use edges to model communication processes (Fig. 1b) cannot model certain required communication types. In our example, for instance, consider using an edge for the representation of the database request (Task 4). It will make it not possible to make a single edge lead to two different computing tasks, Task 5 and Task 6, while having an additional edge will unnecessarily duplicate the communication effort. Fig. 4d shows an example of edges-based communication scheduling. It requires scheduling Task 4 for two edges leading from Task 1 to Tasks 5 and 6. Matching the schedule of the CA-DAG model becomes possible only when additional network link is available, such that both edges can be scheduled in parallel.

5 Properties of Communication Vertices

In this section, we discuss and explain the aforementioned properties in more details.

5.1 Task Parallelization

While it is often assumed that a single vertice v_c represents a piece of computing code, which cannot be further parallelized, the communication vertice v_{comm} does not imply such an assumption.

Communication-related tasks significantly differ from the computing tasks. Their most distinct property is the task parallelization: each communication task $v_i^{comm} \in V_{comm}$ can be divided into n different independent communication tasks v_{ij}^{comm}, $j = 1, \ldots, n$, with a size of communication task in bits equals to $\varphi(v_i^{comm})/n$.

All of the bits that are to be transferred are independent. They can be transmitted on different paths of the network and reassembled in the original sequence at the destination node. As a result, each communication vertice v_i^{comm} can be split into a number of data flows scheduled in parallel or sequentially. Network paths used for their transmission can be either completely different, i.e. include only the sender and the receiver as common nodes, or partially overlapped. The number of parallel flows depends on the number of network paths available, the size of data, available effective bandwidth, an overhead of the protocol used for communication, etc.

5.2 Multipath Routing

In a fully deterministic system, a schedule can be computed by finding an association between the DAG representing cloud applications, and topology graph representing a data center network, which includes network nodes, switches, and communication links with their transmission rates. This approach has a number of limitations. It assumes circuit switching and static routing. This guarantees a dedication of certain bandwidth resources along a predefined network path for the whole duration of communication. However, in real systems, these assumptions do not hold. Nowadays, most of the communication networks are packet-switched, and packet routing decisions are taken at every hop independently. Furthermore, most of the data center network topologies, including the most commonly used fat tree topology, introduce multipath connections as a mean to provide resilience and load balancing. The availability of multiple paths is essential to benefit from the parallelization of communication tasks discussed earlier in this section.

5.3 5.3 Task Completion Time

In computing, the task completion time corresponds to the time a processing resource is released from executing computing instructions, after which a computing result is available. In packet-switched networks, multiple links are involved into communication task execution. They operate at different data rates, and process a packet transmission sequentially.

Figure 5 attributes communication delays with different network components. The information processing and packetization delay d_{proc} as well as queuing delay d_{queue} are occurred at the network node. The transmission delay d_{tx} defines a time interval of network link occupancy. For a data segment of the length S and link data rate r, the transmission delay is defined as a ratio S/r. The propagation delay d_{prop} corresponds to the time the signal travels from a sender to receiver. It is defined as a ratio between the link length l_{link} and propagation speed of the link medium c. Combining

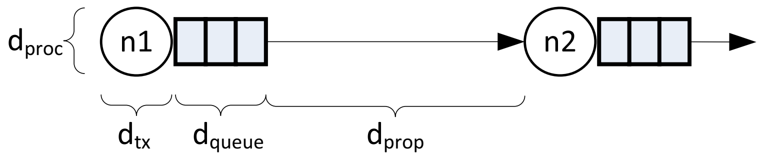

Fig. 5 Communication delays

the aforementioned delays together, we can compute task completion delay for the network path of N hops as follows:

$$d_{comm} = \sum_{i=1}^{N} \left(d^i_{proc} + d^i_{queue} + d^i_{tx} + d^i_{prop} \right). \qquad (1)$$

The minimum communication delay will correspond to the system with very fast processing ($d_{proc} \to 0$) and empty buffers ($d_{queue} = 0$), and will be expressed by $\sum (d_{tx} + d_{prop})$.

5.4 Available Bandwidth

Typically, communication resources are associated with the residual capacity – the amount of the path capacity left unoccupied by the traffic flowing along the path. However, in practice, residual capacity reflects only the minimum amount of bandwidth that a newly introduced flow can obtain. The communication flows sharing the same path or a segment of a path in the network compete for bandwidth resources. As a result, a newly introduced data flow along with relying on the residual capacity may grab a share of the bandwidth currently used by the other flows.

The communication protocol and its performance are two of the most important factors. Overwhelming majority of data transmissions is performed using Transmission Control Protocol (TCP). It is the only protocol in the standard TCP/IP protocol stack able to guarantee both reliability and flow control. It uses a positive feedback loop with the receiver. Based on the feedback information, TCP triggers retransmissions for the packets, which are lost due to congestion or link errors, and adjusts sending rate. The sending rate is additively increased for every feedback message received unless a packet loss is detected. In the latter case, the TCP reduces its sending rate multiplicatively, typically by a factor of 2.

Due to the uncertainty on the end-to-end network path, and operational TCP dynamics, accurate calculation of a node sending rate becomes unrealistic making it difficult to predict completion time of communication tasks [26, 47]. However, for the purpose of scheduling, it is important to estimate the boundaries for this value.

A good estimate of the steady-state TCP performance can be obtained as the following [32]:

$$B(p) = \frac{MSS}{RTT \cdot \sqrt{p}}, \qquad (2)$$

where MSS is a maximum segment size typically selected to fit maximum packet size, which will not trigger fragmentation at the network interface card, RTT is the round-trip time between the sender and the receiver, and p is an error probability, which includes both congestion- and link-related packet losses. According to (2), for the RTT of 200 ms, which is common in Internet, typical for Ethernet MSS of 1500 bytes, and typical for wired links error rates in the order of 10^{-7}, the maximum achievable TCP sending rate is less than 200 Mb/s. To estimate the protocol-related overhead, we may consider an upper bound of the link capacity and use a more precise model from [33]. For a typical per-server available bandwidth of 300 Mb/s and round-trip delay of 100 ms, the transmission of 500 MB data fragment using TCP protocol will take almost 15 seconds versus theoretical 13 seconds in case of raw data transmission with no protocol used. In this example, the overhead of TCP protocol is round 13 %.

5.5 Uncertainty in Data Size

Communication actions performed by a task executed in data center can be classified into unidirectional and bidirectional. Unidirectional communications are typically related to the task outputs to the user or another service in data center. They have a well-defined size of the information that needs to be transferred. Bidirectional communications are related to the request-response actions performed by the task, such as database queries. In this regard, while the size of outgoing request is well-known, the amount of information that will be received back is often unknown. For example, in Fig. 2, in Task 4 the list of email messages is received from the database. The list can be completely empty or has a large size depending on the number of emails stored in the user mailbox.

In order to cope with the uncertainty in data size of communications adaptive scheduling approaches should be used. However, to make resource allocation efficient, it is important to estimate task completion delays and usage of network resources for such bidirectional communications. One of the most promising approaches is to use statistical data mining approaches. Each node can include a software module which, based on the precedent experience, will estimate a query processing delay, the round-trip time to a database server, as well as the size of the data output reducing uncertainly and assisting resource allocation.

Scheduling with uncertainly have been also extensively studed in grid networks with the relation to application demands and resource availability [44], and uncertainty of communication demands [45, 46].

6 Communication-aware Scheduling

In this section, we discuss the impact of the introduction of the communication-aware DAG model on existing scheduling solutions. The problem of DAG scheduling is known to be NP-complete even for DAGs with no communication-related vertices or edges [34]. Therefore, classical exact and enumerative methods are only useful to solve the reduced size problems. Approximation algorithms, linear programming, combinatorial, and stochastic solutions have to be adopted for the problem solution. Communication is not only a problem on the algorithmic level, but also for the scheduling model as well. Most of the scheduling algorithms employ a strongly idealized model of the targeted system. It is assumed that all of the processors are fully connected. The information governing scheduling decisions is assumed to be known in advance. However, it is not always the case when the communication-aware model is considered. The most widely used heuristics are usually classified into three categories: duplication-based scheduling, cluster-based scheduling, and priority-based scheduling.

Task duplication is a special scheduling method that replicates selected tasks to reduce inter-processor communications [35, 36]. One of the main problems in obtaining high performance of data intensive applications is the inevitable communication overhead when tasks executed on different processors exchange data. Duplication-based scheduling can reduce this overhead by allocating the tasks redundantly on more than one processor. Task duplication can decrease in the cost of communications at the expense of bringing additional computational load into computing system. Nevertheless, task replication seems to be an impractical option in the context of cloud computing since it tends to redundantly use and waste resources for replicas.

Task clustering exploits the idea of grouping the heavily communicating tasks and executing them on the same computing resource [37, 38]. A problem arises when the number of clusters is larger than the number of the available processors [9]. This would require scheduling of several clusters onto the same processor inevitably increasing the overall length of the schedule. The mapping or post-clustering phases can refine the obtained clusters and acquire the final task-to-resource map. The problem becomes more difficult under the condition of resources with different and varying capabilities. Known static load balancing mechanisms become impractical. Even effectively modified, clustering algorithms cannot be directly applied under these conditions.

In **priority-based scheduling**, tasks are scheduled onto resources according to their priorities [5]. The heuristics use different strategies to decide on the task priorities and allocated resources for each task. This type of scheduling techniques is usually relatively easy to implement, but, typically they do not consider inter-processor communication delay when assigning scheduling priorities.

Uncertainties inside applications (amount of computing and communication works) and of the execution environment (number of available machines, their location, their capabilities, the network topology, effective communication bandwidth, etc.) add new dimension to the scheduling problem.

Solutions for precedence-constrained problems from classical theory are not suitable for the cloud scenarios. Scheduling in cloud computing environments poses challenges not found in other distributed service and computing environments mainly due to the dynamicity of computing resources and communications, mix in job workloads, and different properties of the workflows. Traditional scheduling algorithms for high-performance computing are often afforded many assumptions that do not hold in cloud computing, such as relatively negligible communication costs, precise knowledge of the application structure, and the communication patterns. Most of the existing scheduling algorithms employ a strongly idealized model of the target system making difficult to apply them in the cloud.

Another important feature of existing scheduling strategies is a consideration of a single objective criterion. Most of the scheduling strategies for DAGs adopt the best-effort approach, which minimizes only performance objective. However, scheduling in the cloud is inherently multi-objective. For instance, system performance related objectives, cost based, energy consumption based, and QoS based objectives must be considered.

In cloud computing environments, resource scheduling should be able to meet QoS requirements for individual DAG instances at the same time maximizing performance for multiple DAGs running concurrently. Workflow or DAG scheduling has diversified into many research directions: minimization of critical path execution time, selection of admissible resources, allocation of suitable resources for data intensive workflows, QoS constraint scheduling, as well as fine-tuning of the workflow execution and performance analysis. Most of them have considered single DAG scheduling problems. As already mentioned cloud applications are typically small, but arrive to the data center in millions. Therefore, the scheduler should be designed to take into account not only dependencies within a single job, but also multiple DAGs.

Several research works address the scientific workflow-scheduling problem on clouds. The workflows are represented by DAGs. Unlike tightly coupled applications in which tasks communicate in the network directly, workflow tasks typically communicate using file system. Most the scheduling approaches are based on list scheduling, clustering, and meta-heuristic search [39–41].

The communication-aware scheduling problem considered in this work is hybridization between network scheduling and classical machine scheduling. The scheduling algorithm should be aware of the computing and communication tasks to be scheduled. We reduce an online DAG scheduling problem with communications to the problem without communications since communications are included into the new DAG model as tasks to be performed. Computing tasks have to be mapped to computing resources and communication tasks to the communicating media resources.

To treat uncertainly and dynamism of the problem, stochastic models, non-clairvoyant and knowledge-free scheduling strategies can be applied. The latter corresponds to the strategies where scheduling decisions are free from information of resources and characteristic of running applications. Online list scheduling algorithms are an example of non-clairvoyant knowledge-free scheduling strategies since list scheduling requires no knowledge of unscheduled tasks as well as of all tasks currently being processed. It is very powerful in dynamic environments and especially in online non-clairvoyant scheduling [42]. One standard alternative is to consider randomized algorithms that make random choices as they construct a schedule. More sophisticated greedy strategies that can be adopted with some knowledge of the scheduled application are Prioritizing Round Robin, First In First Out, Earliest Deadline First, Least Laxity First, Multilevel Feedback methods. To deal with multipath routing, the list scheduling algorithm can be coupled with a routing algorithm to select the links involved into data forwarding.

Algorithms based on dynamic priority scheduling can also be adapted to the dynamic context. The basic idea is to continue focus on policies that assign priorities based on temporal parameters and maximize of resource utilization. In such algorithms, the priorities of the ready tasks are calculated during the execution of the system. The aim is to adapt to dynamically changing progress and form an optimal configuration in self-sustained manner. Some dynamic priority scheduling algorithms are Earliest deadline first scheduling and Least slack time scheduling.

In general, selecting appropriate heuristic requires understanding the application, the system and the objectives. However, general frameworks as list scheduling can be adapted to the communication-aware scheduling model.

7 Experimental Results

This section presents performance evaluation results that confirm the benefits of the proposed CA-DAG model for scheduling cloud computing applications. The CA-DAG model is compared against CU-DAG and EB-DAG models reviewed in Section 2. We first generated the application workloads according to the CA-DAG model. Thereafter, the obtained workloads were converted according to the communication-unaware and edge-based models and scheduled with the list scheduling algorithm.

7.1 System Architecture and Workload Generation

The target system architecture is composed of a set of identical computing resources. The communication resources are represented with a shared network link (bus network), interconnecting computing resources, and a database. The network topology allows only one node to communicate at a time, while other nodes must detain their transmissions until the link becomes free.

The Winkler graph generator [18] was used to produce the workloads. The generator is based on random orders methods making the generated graphs to be representative of multidimensional orders. The two-dimensional orders graphs were generated. To achieve the aforementioned, n points were selected randomly in the [0;1] × [0;1] square. Each point becomes a node and there is an edge between two points a and b. If b is greater than a, then in both directions. To generate large sets of graphs, the following two parameters were varied: the number of nodes n and the number of communications. The graphs had a size of 20, 30, 40, and 50 nodes. Moreover, the obtained DAGs fell into two categories according to their communication intensity: DAGs with occasional communications and DAGs with frequent communications. To model these categories, two probabilities representing the amount of communication were used: 0.3 to represent occasional communications and 0.7 for frequent communications. Moreover, to rank communications the Communication-to-Computation Ratio (CCR) was used. The CCR measure indicates whether a DAG is communication intensive, computation intensive or balanced. For a given DAG, the CCR ratio is computed by the average communication cost divided by the average computation cost on a target system. A high value of CCR indicates that the DAG is communication intensive. We used the following three values of CCR: 0.1 for computationally intensive DAGs, where communication is of low significance compared to the cost of computations, 1 for the balanced DAGs, and 2 for communication intensive DAGs where the significance of communication processes is high. There are 30 graphs generated for each combination input parameters, while the total number of generated DAGs for each application model is 720.

7.2 Scheduling Algorithm

All of the evaluated DAG models were compared using an offline (deterministic) scheduling algorithm with an assumption of zero release times of DAGs and clairvoyant execution and communication time. Many offline scheduling algorithms exhibit good performance also in the online scenario. From theory, it is known that the performance bounds of the offline scheduling strategies can be approximated for the online case [19]. As the aim of this section is to compare the application models and not the scheduling algorithms, the same heuristic is employed for each of the described model. For the different models, list scheduling is employed. A list scheduling algorithm is a two-phases scheduling algorithm that maintains a list of all of the ready tasks of a given graph. A task is considered ready to be scheduled when all of its predecessors have been already scheduled. In the common variant of list scheduling, the nodes are ordered according to a priority in the first part of the algorithm. The task with the highest priority is selected. Thereafter, in the second phase, a suitable processor that minimizes a predefined cost function (in this case the processor that allows the earliest finish time of a task) is selected. A common priority is the task's bottom level (blevel), which is the length of the longest path leaving the task. The blevel of a task is bounded from above by the length of a critical path. The blevel of a current task is computed by adding the computation cost along the longest path of the task from the exit task (a task without successors) in the task graph including the computation cost of the current task and excluding the communication costs.

The *blevel* of any task t_i is recursively calculated as follows:

$$blevel\,(t_i) = p_i + max_{t_j \in succ(t_i)} \left\{ blevel\left(t_j\right) \right\}, \quad (3)$$

where p_i denotes the execution time of task t_i and succ (t_i) is the set of immediate successors of task t_i.

We adapted the list of scheduling algorithm to consider three different communication models. The algorithm schedules computational tasks in computing resources and communications in the network link. The list scheduling is applied under the communication-aware, communication-unaware and edge-based models, denoted by "CA-DAG", "CU-DAG", and "EB-DAG" respectively. Using the same algorithm for each of the model allows analyzing the impact of the model on the quality of the produced schedules under no influence of different scheduling techniques.

7.3 Scheduling Criteria

The following criteria are used to evaluate the schedule produced by the algorithm: approximation factor and schedule efficiency. Let C_{max} be the maximum completion time or makespan of the schedule produced by the scheduling algorithm under a given DAG application model. The approximation factor [19] is defined as $\rho = C_{max}/C^*_{max}$, where C^*_{max} is the optimal makespan. As it is generally not possible to determine the optimal makespan experimentally, we use the lower bound $\tilde{C}^*_{max}$ of the optimal makespan C^*_{max} instead

$$\begin{aligned} C^*_{max} &\geq \tilde{C}^*_{max} \\ &= max\left\{\max\left(blevel\left(t_i\right)\right)\ ,\ \frac{\sum_{i=1..,n}\left(p_i\right)}{m}\right\}, \end{aligned} \tag{4}$$

where $\max(\text{blevel}(t_i))$ represents the critical path of the DAG without considering communication costs and m denotes the number of computing resources. The efficiency of the schedule S defined as $\text{eff}(S) = \frac{\sum_{i=1..,n}(p_i)}{C_{max} \times m}$ is the ratio of the sequential execution time of the graph to the makespan of the schedule by the number of computing resources. It measures how well-utilized the computing resources are in scheduling of a given application, compared to how much effort is wasted during communication.

7.4 Results

To summarize, a large set of randomly generated DAGs is scheduled by a list scheduling algorithm under the CA-DAG, communication-unaware, and edge-based models onto two configurations of a target system with four and eight computing nodes arranged into bus network topology.

Approximation Factor Figures 6 and 7 show the obtained results for the approximation factor for DAGs with occasional and frequent communications

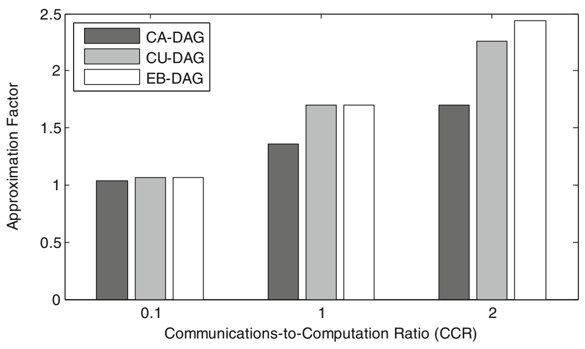

Fig. 6 Approximation factor for DAGs with occasional communications

respectively. The benefits of the CA-DAG model can be observed in both figures. For computation intensive DAGs (CCR=0.1) the values of approximation factor are close for all the models. With small CCR values the communication is less important than computation, and the communication awareness of CA-DAG does not lead to significant benefits over Comm-unaware DAG and Edge-based DAG models. A small approximation factor indicates that the results of a schedule for a given communication DAG model are close to the lower bound. It can be observed that the approximation factor degrades when the amount of communications increases. However, for the CA-DAG model the degradation of the approximation factor is smaller than in related models. The improvement of CA-DAG model becomes significant for balanced (CCR=1) and communication intensive (CCR=2) DAGs where communication awareness can benefit from the increase amount of transmissions.

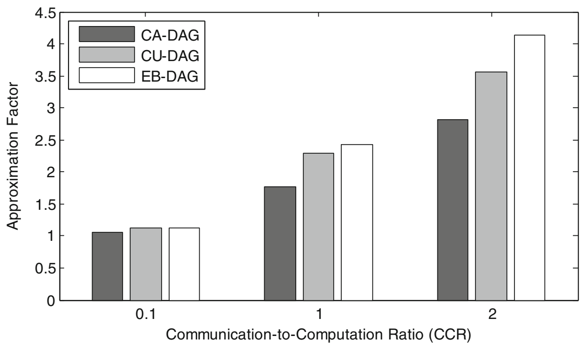

Fig. 7 Approximation factor for DAGs with frequent communications

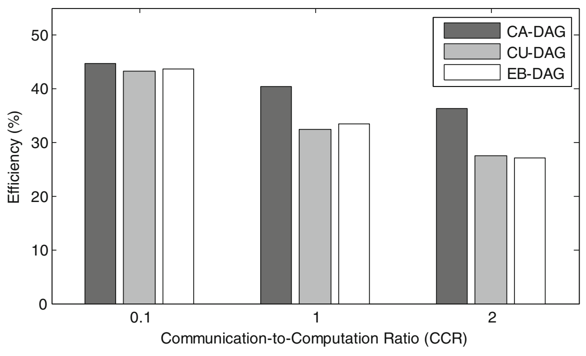

Fig. 8 Schedule efficiency for DAGs with occasional communications

Efficiency Figures 8 and 9 show the efficiency of the schedule produced by different communication models. They confirm the results obtained with approximation factor. Indeed, the communication-aware schedulers under the CA-DAG model achieve better efficiency for all CCRs. This is especially evident for balanced and communication intensive DAGs. Figure 9 confirms that not only the cost of the communications is important, but also their amount.

In summary, CA-DAG significantly improves the approximation factor and efficiency of the produced schedules. However, we have only conducted experiments considering a single shared network link. Therefore, the communications are serialized. It would be interesting to consider more than one link to parallelize communications. We have also considered only one scheduling algorithm. The high importance of communication under the CA-DAG model seems to demand the development of more sophisticated algorithms in order to exploit full potential of this new model.

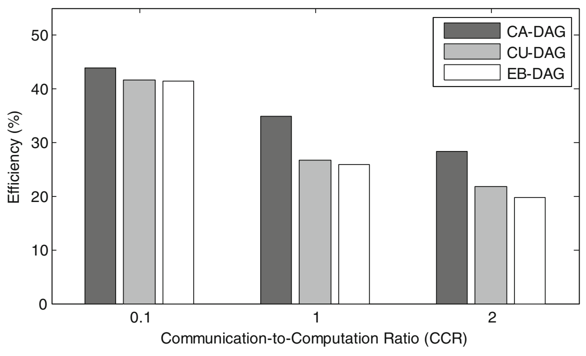

Fig. 9 Schedule efficiency for DAGs with frequent communications

8 Conclusions

The main reason that traditional cluster and grid resource allocation approaches fail to provide efficient performance in clouds is that most of cloud applications require availability of communication resources for information exchange between tasks, with databases or the end users. Moreover, execution environment of cloud applications is not known at development time—the number of available machines, their location, their capabilities, the network topology, and effective communication bandwidth cannot be predicted ahead. In general, it will be different for each program/service invocation. To deal with the dynamics of the execution environment, either the programmer must explicitly write adaptive programs or cloud software environment, such as a runtime scheduling system, must deal with the dynamics.

For an effective utilization of the Cloud, the programs must be decoupled from the execution environment. Programs should be developed for a uniform and predictable virtual services, thus, simplifying their development; the runtime system should deal with the dynamics. Cloud application model has to allow high level representation of computing and communication based on the nature of the problem, and independent of the executing environment. Mapping computation on machines, balancing the loads among different machines, removing unavailable machines from a computation, mapping communication tasks and balancing the communication loads among different links have transparently be provided by the runtime system.

In this paper, we discuss new CA-DAG model for cloud computing applications, which overcomes shortcomings of existing approaches using communication awareness. It is based on a DAG, which along with computing vertices has separate vertices to represent communications. Such a representation allows making separate resource allocation decisions, assigning processors to handle computing jobs and network resources for information transmissions. The proposed communication-aware model creates space for optimization of many existing solutions to resource allocation and, together with performance and energy efficiency metrics of communication systems [50], will become an essential tool in the design of completely new scheduling schemes of improved efficiency.

In the future work, we will focus on developing novel communication-aware resource allocation solutions based on the proposed model. We will generalize CA-DAG model to capture dynamics of cloud environment. One of the important issues is the comprehensive simulations using GreenCloud simulator [43], and practical validation of proposed solutions.

Acknowledgment The authors would like to acknowledge the funding from National Research Fund, Luxembourg in the framework of ECO-CLOUD (C12/IS/3977641) project.

References

1. Papadimitriou, C.H., Yannakakis, M.: Towards an architecture-independent analysis of parallel algorithms. SIAM J. Comput. **19**(2), 322–328 (1990)
2. Culler, D.E., Karp, R.M., Patterson, D.A., Sahay, A., Santos, E.E., Schauser, K.E., Subramonian, R., von Eicken, T.: LogP: a practical model of parallel computation. Commun. ACM **39**(11), 78–85 (1996)
3. El-Rewini, H., Lewis, T.G.: Scheduling parallel program tasks onto arbitrary target machines. J. Parallel Distrib. Comput. **9**(2), 138–153 (1990)
4. Sinnen, O., Sousa, L.A.: Communication contention in task scheduling. IEEE Trans. Parallel Distrib. Syst. **16**(6), 503–515 (2005)
5. Macey, B.S., Zomaya, A.Y.: A comparison of list scheduling heuristics for communication intensive task graphs. Cybern. Syst. **28**(7), 535–546 (1997)
6. Drozdowski, M.: Scheduling with communication delays. In: Scheduling for Parallel Processing, ser. Computer Communications and Networks, pp. 209–299. Springer, London (2009)
7. Kliazovich, D., Bouvry, P., Khan, S.U.: DENS: Data Center Energy-Efficient Network-Aware Scheduling. Clust. Comput. **16**(1), 65–75 (2013)
8. Pecero, J.E., Trystram, D., Zomaya, A.Y.: A new genetic algorithm for scheduling for large communication delays, pp. 241–252. Euro-Par (2009)
9. Lepère, R., Trystram, D.: A new clustering algorithm for large communication delays. IPDPS (2002)
10. Kwok, Y.-K., Ahmad, I.: Dynamic critical-path scheduling: an effective technique for allocating task graphs to multiprocessors. IEEE Trans. Parallel Distrib. Syst. **7**(5), 506–521 (1996)
11. Ahmad, I., Kwok, Y.-K., Wu, M.-Y.: Analysis, evaluation, and comparison of algorithms for scheduling task graphs on parallel processors. In: Second International Symposium on Parallel Architectures, Algorithms, and Networks, pp. 207–213 (1996)
12. Schatz, R., Varela, M., Timmerer, C.: Challenges of QoE management for cloud applications. IEEE Commun. Mag. **50**(4), 28–36 (2012)
13. Dongarra, J.: Trends in high performance computing: a historical overview and examination of future developments. IEEE Circuits Devices Mag. **22**(1), 22–27 (2006)
14. AbdelBaky, M., Parashar, M., Kim, H., Jordan, K.E., Sachdeva, V., Sexton, J., Jamjoom, H., Shae, Z.-Y., Pencheva, G., Tavakoli, R., Wheeler, M.F.: Enabling High-Performance Computing as a Service. Computer **45**(10), 72–80 (2012)
15. White paper: The impact of latency on application performance. Nokia Siemens Networks (2009)
16. Benson, T., Akella, A., Maltz, D.A.: Network traffic characteristics of data centers in the wild. In: The 10th Annual Conference on Internet Measurement (IMC), ACM, New York, NY, USA, pp. 267–280 (2010)
17. Kandula, S., Sengupta, S., Greenberg, A., Patel, P., Chaiken, R.: The nature of data center traffic: measurements & analysis. In: The 9th ACM SIGCOMM conference on Internet measurement conference (IMC), ACM, New York, NY, USA, pp. 202–208 (2009)
18. Srikanth, G.U., Shanthi, A.P., Maheswari, V.U., Siromoney, A.: A Survey on Real Time Task Scheduling. Eur. J. Sci. Res. **69**(1), 33–41 (2012)
19. Thulasiraman, K., Swamy, M.N.S.: 5.7 Acyclic Directed Graphs. Graphs: Theory and Algorithms, John Wiley and Son, p. 118. ISBN 978-0-471-51356-8
20. Hac, A., Sheng, C.: User Mobility Management in the PCS Network Through the Placement of Hierarchical Databases. Int. J Wireless Inf. Networks **5**(3) (1998)
21. Browne, S.: Communication and synchronization issues in distributed multimedia database systems. Adv. Database Syst. **759**, 381–396 (1993)
22. Macdonald, C., Ounis, I., Tonellotto, N.: Learning to predict response times for online query scheduling. In: 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, Portland OR, USA (2012)
23. Brutlag, J.D., Hutchinson, H., Stone, M.: User Preference and Search Engine Latency. In: JSM Proceedings, Qualtiy and Productivity Research Section, Alexandria, VA (2008)
24. Choudhury, P., Chakrabarti, P.P., Kumar, R.: Online Scheduling of Dynamic Task Graphs with Communication and Contention for Multiprocessors. IEEE Trans. Parallel Distrib. Syst. **23**(1), 126–133 (2012)
25. PengCheng, M., Nezan, J.-F., Raulet, M., Cousin, J.-G.: Advanced list scheduling heuristic for task scheduling with communication contention for parallel embedded systems. Sci. China Inf. Sci. **53**(11), 2272–2286 (2010)
26. Prasad, R., Dovrolis, C., Murray, M., Claffy, K.: Bandwidth estimation: metrics, measurement techniques, and tools. IEEE Network **17**(6), 27–35 (2003)
27. Plummer, D.C.: An Ethernet Address Resolution Protocol - or - Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. RFC 826, Internet Engineering Task Force (1982)
28. Postel, J.: Internet Control Message Protocol. Internet Engineering Task Force, RFC 792 (1981)

29. Spring, N., Wetherall, D., Ely, D.: Robust Explicit Congestion Notification (ECN). RFC 3540, Internet Engineering Task Force (2003)
30. Jain, M., Dovrolis, C.: End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput. IEEE/ACM Trans. Networking **111**(14), 537–549 (2003)
31. Kapoor, R., Chen, L.-J., Sanadidi, M.Y., Gerla, M.: Accuracy of link capacity estimates using passive and active approaches with CapProbe. In: Ninth International Symposium on Computers and Communications, pp. 1085–1090 (2004)
32. Mathis, M., Semke, J., Mahdavi, J., Ott, T.: The macroscopic behavior of the TCP congestion avoidance algorithm. SIGCOMM Comput. Commun. Rev. 27 **3**, 67–82 (1997)
33. Padhye, J., Firoiu, V., Towsley, D., Krusoe, J.: Modeling TCP throughput: A simple model and its empirical validation," In: ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, pp. 303–314 (1998)
34. Ullman, J.D.: NP-Complete scheduling problems. J. Comput. Syst. Sci. **10**, 384–393 (1975)
35. Bozdag, D., Ozguner, F., Catalyurek, U.V.: Compaction of schedules and a two stage approach for duplication-based DAG scheduling. IEEE Trans. Parallel Distrib. Syst. **20**(6), 857–871 (2009)
36. Kruatrachue, B., Lewis, T.G.: Grain size determination for parallel processing. IEEE Software **5**(1), 23–32 (1988)
37. Sarkar, V.: Partitioning and scheduling parallel programs for execution on multiprocessors. MIT Press, MA, USA (1989)
38. Gerasoulis, A., Yang, T.: On the granularity and clustering of directed acyclic task graphs. IEEE Trans. Parallel Distrib. Syst. **4**(6), 686–701 (1993)
39. Juve, G., Deelman, E., Berriman, G.B., Berman, B.P., Maechling, P.: An Evaluation of the Cost and Performance of Scientific Workflows on Amazon EC2. J. Grid Comput. **10**(1), 5–21 (2012)
40. Zhang, F., Cao, J., Li, K., Khan, S.U., Hwang, K.: Multi-objective scheduling of many tasks in cloud platforms. Future Generation Computer Systems, Available online 18. ISSN 0167-739X (2013)
41. Pandey, S., Wu, L., Guru, S.M., Buyya, R.: A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. AINA, 400–407 (2010)
42. Carbajal, A.H., Tchernykh, A., Yahyapour, R., Röblitz, T., Ramírez-Alcaraz, J., González-García, J.-L.: Multiple workflow scheduling strategies with user run time estimates on a grid. J. Grid Comput. **10**(2), 325–346 (2012). doi:10.1007/s10723-012-9215-6. Springer-Verlag New York, Inc. Secaucus, NJ, USA,
43. Kliazovich, D., Bouvry, P., Khan, S.U.: GreenCloud: A packet-level simulator of energy-aware cloud computing data centers. J. Supercomput. **62**(3), 1263–1283 (2012)
44. Batista, D.M., da Fonseca, N.L.S.: Robust scheduler for grid networks under uncertainties of both application demands and resource availability. Comput. Netw. **55**, 3–19 (2011)
45. Batista, D.M., da Fonseca, N.L.S.: Scheduling Grid Tasks in Face of Uncertain Communication Demands. IEEE Trans. Netw. Serv. Manag. **8**, 93–102 (2011)
46. Batista, D.M., da Fonseca, N.L.S., Miyazawa, F.K., Granelli, F.: Self-Adjustment of Resource Allocation for Grid Applications. Comput. Netw. **52**, 1762–1781 (2008)
47. Batista, D.M., Chaves, L.J., da Fonseca, N.L.S., Ziviane, A.: Performance analysis of available bandwidth estimation tools for grid networks. J. Supercomput. **53**, 103–121 (2010)
48. Kliazovich, D., Arzo, S.T., Granelli, F., Bouvry, P., Khan, S.U.: e-STAB: Energy-Efficient Scheduling for Cloud Computing Applications with Traffic Load Balancing. In: IEEE International Conference on Green Computing and Communications (GreenCom), Beijing, China, pp. 7–13 (2013)
49. Guzek, M., Kliazovich, D., Bouvry, P.: A Holistic Model for Resource Representation in Virtualized Cloud Computing Data Centers. In: IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Bristol, UK (2013)
50. Fiandrino, C., Kliazovich, D., Bouvry, P., Zomaya, A.Y.: Performance and energy efficiency metrics for communication systems of cloud computing data centers. IEEE Transactions on Cloud Computing (2015)