CA-DAG: Communication-Aware Directed Acyclic Graphs for Modeling Cloud Computing Applications

Dzmitry Kliazovich
University of Luxembourg
6 rue Coudenhove Kalergi, Luxembourg
dzmitry.kliazovich@uni.lu

Johnatan E. Pecero
University of Luxembourg
6 rue Coudenhove Kalergi, Luxembourg
johnatan.pecero@uni.lu

Andrei Tchernykh
CICESE Research Center
Ensenada, B.C. Mexico
chernykh@cicese.mx

Pascal Bouvry
University of Luxembourg
6 rue Coudenhove Kalergi, Luxembourg
pascal.bouvry@uni.lu

Samee U. Khan
North Dakota State University
Fargo, ND 58108-6050, USA
samee.khan@ndsu.edu

Albert Y. Zomaya
University of Sydney
Sydney, NSW 2006, Australia
a.zomaya@usyd.edu.au

***Abstract*—The review of the requirements of different cloud applications identified the need to consider communication processes explicitly and equally to the computing tasks. Following this observation, we propose a new communication-aware model for cloud computing applications, called CA-DAG. This model is based on Directed Acyclic Graphs (DAGs) 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, such as application database requests.**

Keywords—Cloud computing, communication awareness, resource allocation, scheduling

I. INTRODUCTION

Cloud computing is an emerging paradigm for providing services and solving large-scale problems in science, engineering, and commerce. 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. As a consequence, 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.

The scheduling of jobs on multiprocessors is generally well understood and has been studied for decades. Many research results exist [2], [6]. Some of them provide theoretical insights while others give hints for practical implementation. However, the communication-aware scheduling problems that require an availability of communication resources are rarely 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 the classical scheduling, the communication system modeled either as homogeneous completely connected network [1], which assumes constant communication delays, or heterogeneous, where delays inside a cluster of processors are smaller than that between clusters. Moreover, communication delays can be completely neglected, for example, when a predecessor task and a successor task are executed on the same processor [2]. This is known as the locality assumption. The essential property of such models is that task duplication can avoid communication delays. In fact, there are only few scheduling solutions available that take into account large communication delays [3], [4]. The most widely used approaches to balance communication delays and processing times are task clustering, using critical path analysis or decomposition of the precedence task graph [4].

Cloud applications and services can be represented with workflows, defined as a composition of tasks with precedence constraints, and modeled by DAGs. The vertices of a DAG represent the amount of computing job that has to be processed for successful execution of a task, while the edges define precedence constraints. Such a workflow model works well for HPC applications [5], but fails in the cloud where communication processes often become a bottleneck. Several researchers proposed to adapt the standard DAG model by either allowing vertices to represent both computing and communication requirements of a task [6] or by associating edges with the communications performed by the tasks [7]. However, both models appeared to have shortcomings analyzed in this paper. The first model fails to make distinction between the computing and communication jobs of a task preventing their proper scheduling on fundamentally different resources: processors and communication network. The latter approach representing communication work with edges does not allow a single communication process precede two computing tasks as a single edge cannot lead to two different vertices in a DAG.

In this paper, we define the communication-aware model of cloud applications, called CA-DAG. 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. 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.

The contribution synopsis of the paper is as follows.

978-0-7695-5028-2/13 $26.00 © 2013 IEEE
DOI 10.1109/CLOUD.2013.40

- Analysis of communication requirements of different cloud applications and motivation of need for communication awareness in resource allocation.
- Definition of new communication-aware model for cloud applications.
- Definition of properties of communication tasks.
- Analysis of the impact of the proposed communication-aware model onto existing resource allocation solutions
- Numerical comparison and validation of the proposed model.

The rest of the paper is structured as follows: Section II gives motivation for communication awareness in task scheduling reviewing different cloud applications and analyzing existing models, Section III introduce the concept of communication awareness and the CA-DAG model, Section IV discusses the properties of communication vertices, Section V presents performance comparison, while Section VI concludes the paper with a summary and an outlook of future work on the topic.

II. Need for Communication Awareness

Most of the cloud computing applications require the availability of communication resources for their operations. Table I 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 [8]. All of 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 [9]. The cloud applications located in the top half of the Table I, with cloud gaming and video conferencing being the leaders, impose tight constraints on both the network bandwidth and the delay.

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 producing internal to the datacenter traffic, which may account for as much as 75% of the total traffic [10].

Current models of cloud applications rely mostly on the HPC concepts [5]. 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. 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

TABLE I
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

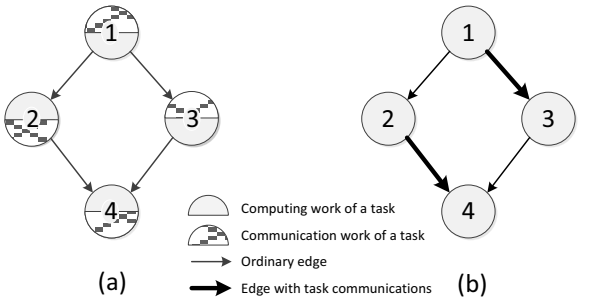

Fig. 1. Modeling communications DAGs: (a) communication-unaware and (b) edges-based model.

and communication requirements of a task (communication-unaware model) or associating edges with the communications performed tasks (edges-based model). Both approaches have significant drawbacks that we detail below.

Communication-unaware model: Joining computing and communication demands of a task together, and representing them as a single vertex [6], as represented in Fig. 1 (a), 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 [11], which is comparable with the time the search engines return results. During this time the computing work, being scheduled for execution, stays on hold waiting for input data. For the DAG example presented in Fig. 1 (a), 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

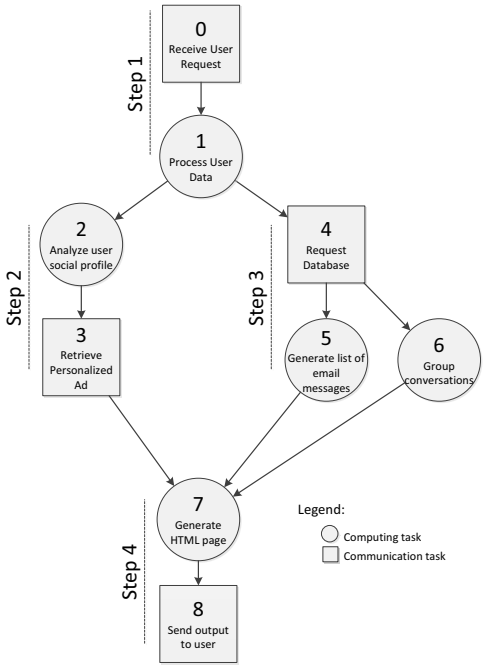

Fig. 2. Communications-aware DAG Model.

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.

Edges-based model: Associating DAG edges with task communications [7] is an attempt to treat communication and computing works 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 the 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. 1 (b). 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 [7]. 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 [12]. 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) [13], 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 [14], which are often generated by the routers in repose to routing failures or congestion problems. 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. Estimating the available bandwidth has been a hot research topic for a number of years with many solutions proposed [12]. 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 [15] and a delay between the moment a probe senses network traffic and the time the measurement becomes available when the probe is returned.

III. Communication-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* $V_c \subseteq V$ *represents computing tasks, and the set* $V_{comm} \subset V$ *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* $\omega(v_i^c)$ *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

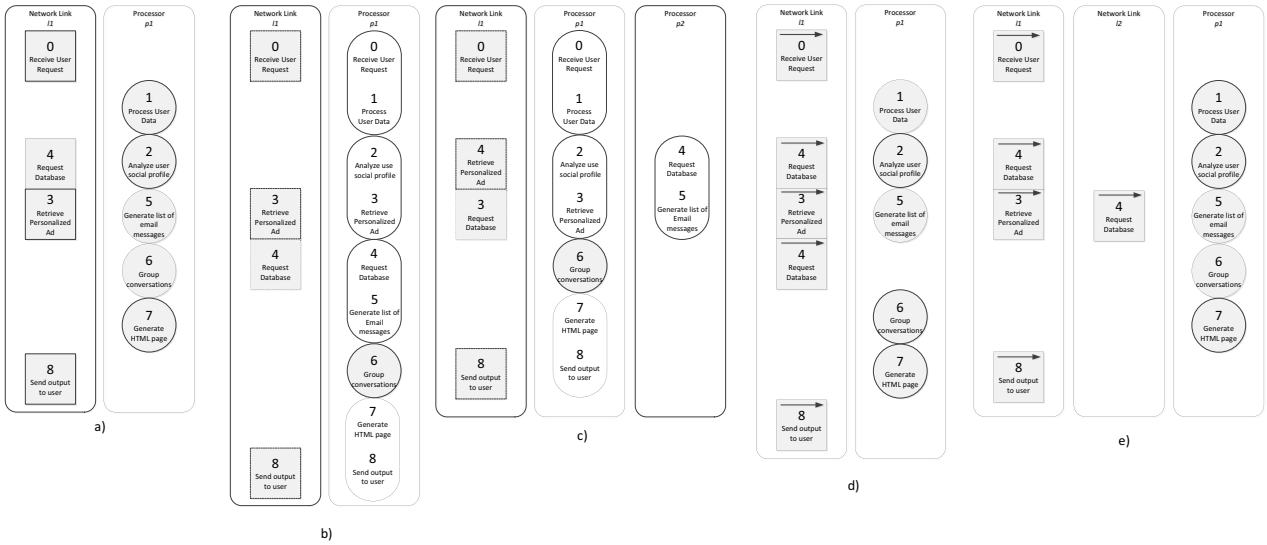

Fig. 3. Schedules for a) communication-aware DAG (CA-DAG) model, b) communication-unaware DAG model and one processor, c) communication-unaware DAG model and two processors, d) edges-based communication model and one network link, and e) edges-based communication model and two network links.

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 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.

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. 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: communication-unaware (Fig. 1 (a)), edges-based communication DAG (Fig. 1 (b)), and communication-aware DAG (Fig. 2).

Fig. 3 shows several possible schedules using these representations for variable number of processors and communication links: **(a)** Communication-Aware DAG (CA-DAG) model, **(b)** communication-unaware DAG model and one processor, **(c)** communication-unaware DAG model and two processor, **(d)** edges-based communication model and one network link, and **(e)** edges-based communication model and one network link.

Fig. 3 (a) shows a possible schedule for the CA-DAG. Computing Tasks 1, 2, 5, 6, and 7 are scheduled 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 unavailable when communication work is seen as a part of a task description.

For the purpose of comparison, Fig. 3 (b) presents a schedule for communication-unaware DAG, depicted in Fig. 1 (a). The inability to control allocation of network resources and 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. 3 (c)).

The DAGs that use edges to model communication processes (Fig. 1 (b)) 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. 3 (d) 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.

IV. Properties of Communication Vertices

In this section, we discuss and explain the properties associated with the communication vertices in more details.

A. Task Parallelization

While it is often assumed that a single vertex v_c represents a piece of computing code that cannot be further parallelized, the communication vertex 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}, $j = 1, \dots, n$, 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. The bits can be transmitted on different paths of the network and reassembled in the original sequence at the destination node. As a result, each communication vertex 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 (include only the sender and the receiver as the common nodes) or partially overlapped. The number of the parallel flows depends on the number of network paths available, the size of data, available effective bandwidth, and an overhead of the protocol used for communication.

B. 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. The above mentioned mandates a dedicated bandwidth along a predefined network path for the whole duration of the communication. However, in real-life 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. Moreover, 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.

C. Task Completion Time

In computing, the task completion time corresponds to the time a processing resource is released. In packet-switched networks, multiple links are involved in the execution of a communication task. They operate at various data rates and sequentially process a packet transmission.

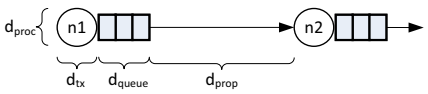

Fig. 4. Communication delays.

Fig. 4 illustrates the communication delays with various network components. The information processing and packetization delay d_{proc} as well as the queuing delay d_{queue} 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 a receiver. It is defined as a ratio between the link length l_{link} and propagation speed of the link medium c. Combining the aforementioned delays together, we can compute task the completion delay for the network path of N hops as follows:

$$d_{comm} = \sum_{i=1}^{N} (d_{proc}^{i} + d_{queue}^{i} + d_{tx}^{i} + d_{prop}^{i}). \quad (1)$$

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 $\sum(d_{tx} + d_{prop})$.

D. Available bandwidth

Typically, communication resources are associated with the residual capacity, which is the amount of the path capacity left unoccupied by the traffic flowing along the path. However, in practice, the residual capacity corresponds to the minimum amount of bandwidth that a newly introduced flow can utilize. The communication flows sharing the same path or a segment of a path in the network compete for bandwidth resources. Consequently, a newly introduced data flow, besides relying on the residual capacity, may also 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 pertaining to the system performance. An 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 [12]. 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 [16]:

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

where MSS is a maximum segment size typically selected to fit maximum packet size, which will not trigger the 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 Eq. (2), for the RTT of 200 ms, which is common in Internet, typical for Ethernet MSS of 1,500 bytes, and typical for wired links error rates in the order of 10^7, the maximum achievable TCP sending rate is less than 200 Mbps. To estimate the protocol-related overhead, we may consider an upper bound of the link capacity and use a more precise model from [17]. For a typical per-server available bandwidth of 300 Mbps 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%.

E. 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. These communications 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.

To cope with the uncertainty in data size of communications, adaptive scheduling approaches must 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 the statistical data mining approaches. Each node can include a software module, based on the precedent experience to 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.

V. Performance Comparison

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 communication-unaware and edge-based models reviewed in Section II. For the purpose of comparisons, 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.

A. 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 a 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.

B. 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:

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

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 (CA-DAG), communication-unaware and edge-based models, denoted by "CA-DAG", "Comm-unaware DAG", and "Edge-based 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.

C. 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

$$C^*_{max} \geq \tilde{C}^*_{max} = max\left\{\max(blevel(t_i)), \frac{\sum_{i=1..n}(p_i)}{m}\right\}, \quad (3)$$

where $\max(blevel(t_i))$ 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.

D. 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: Fig. 5 and Fig. 6 show the obtained results for the approximation factor for DAGs with occasional and frequent 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

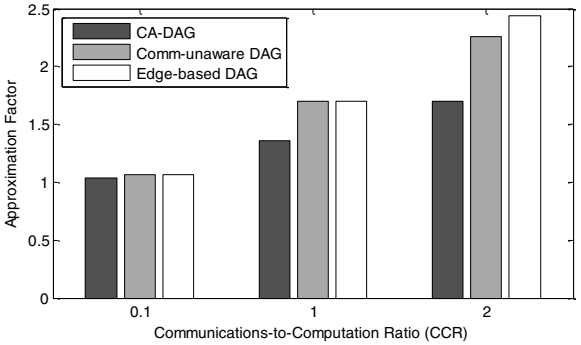

Fig. 5. Approximation factor for DAGs with occasional communications.

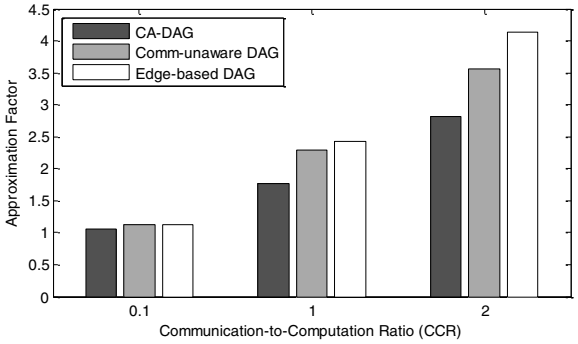

Fig. 6. Approximation factor for DAGs with frequent communications.

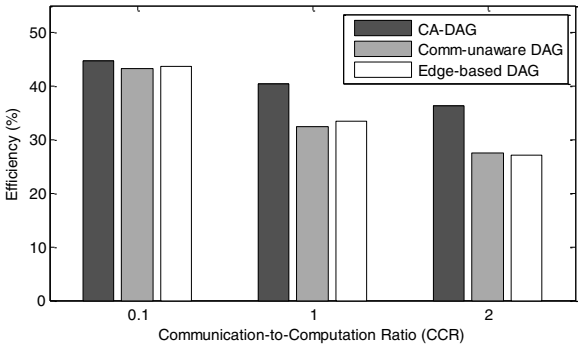

Fig. 7. Schedule efficiency for DAGs with occasional communications.

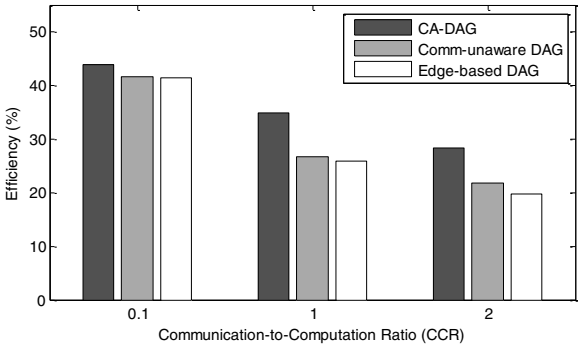

Fig. 8. Schedule efficiency for DAGs with frequent communications.

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.

Efficiency: Fig. 7 and Fig. 8 analyze the efficiency of the schedule produced by different communication models. The obtained efficiency confirms the results obtained with approximation factor. Indeed, the communication-aware schedules under the CA-DAG model achieve better efficiency for all CCRs. This is especially evident for balanced and communication intensive DAGs. Fig. 8 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.

VI. CONCLUSIONS

Cloud computing is an emerging paradigm where traditional resource allocation approaches, inherited from cluster computing and grid computing systems, fail to provide efficient performance. The main reason is that most of cloud applications require availability of communication resources for information exchange between tasks, with databases, or end users. Only few approaches take communication requirements into consideration and often in a highly abstract manner. Moreover, the execution environment of cloud applications is not known at moment of their development — the number of available machines, their location, their capabilities, the network topology, and effective communication bandwidth cannot be predicted in advance. The execution environment will also differ for every next execution of a program or a service. To deal with the dynamics, either software developers must create adaptive programs explicitly or cloud software environment, such as a runtime scheduling system, must adapt.

In this paper, we propose new model for cloud computing applications, which overcomes shortcomings of existing approaches using communication awareness. It is based on a Directed Acyclic Graph (DAG), which along with computing vertices has separate vertices to represent communications. Such 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 as well as developing completely new scheduling schemes of improved efficiency.

The future work will be focused on developing novel communication-aware resource allocation solutions based on the proposed model, generalizing the proposed model to capture dynamics of virtual machines, simulations using GreenCloud simulator [20], and practical implementations.

ACKNOWLEDGMENTS

The authors would like to acknowledge the funding from National Research Fund, Luxembourg in the framework of ECO-CLOUD (C12/IS/3977641) and Green@Cloud (INTER/CNRS/11/03) projects as well as Marie Curie Actions of the European Commission (FP7-COFUND). Samee U. Khan's work was partly supported by the Young International Scientist Fellowship of the Chinese Academy of Sciences, (Grant No. 2011Y2GA01).

REFERENCES

[1] C.H. Papadimitriou and M. Yannakakis. "Towards an architecture-independent analysis of parallel algorithms," SIAM Journal on Computing, 19(2):322-328, 1990.

[2] H. El-Rewini and T. G. Lewis. "Scheduling parallel program tasks onto arbitrary target machines," Journal of Parallel and Distributed Computing, 9(2):138-153, 1990.

[3] D. Kliazovich, P. Bouvry, and Samee U. Khan, "DENS: Data Center Energy-Efficient Network-Aware Scheduling," Cluster Computing, vol. 16, no. 1, pp. 65 – 75, 2013.

[4] J. E. Pecero, D. Trystram, and A. Y. Zomaya: "A new genetic algorithm for scheduling for large communication delays," Euro-Par 2009.

[5] M. AbdelBaky, M. Parashar, Hyunjoo Kim, K. E. Jordan, V. Sachdeva, J. Sexton, H. Jamjoom, Zon-Yin Shae, G. Pencheva, R. Tavakoli and M. F. Wheeler, "Enabling High-Performance Computing as a Service," Computer, vol.45, no.10, pp.72-80, Oct. 2012.

[6] G. U. Srikanth, A. P. Shanthi, V. U. Maheswari, and A. Siromoney, "A Survey on Real Time Task Scheduling," European Journal of Scientific Research, vol. 69, no. 1, pp. 33-41, 2012.

[7] O. Sinnen and L. A. Sousa, "Communication contention in task scheduling," IEEE Transactions on Parallel and Distributed Systems, vol. 16, no. 6, pp. 503- 515, June 2005.

[8] R. Schatz, M. Varela and C. Timmerer, "Challenges of QoE management for cloud applications," IEEE Communications Magazine, vol. 50, no. 4, pp, 28 - 36, April 2012.

[9] "White paper: The impact of latency on application performance," Nokia Siemens Networks, 2009.

[10] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: measurements & analysis," ACM SIGCOMM IMC, ACM, New York, NY, USA, 202-208, 2009.

[11] S. Browne, "Communication and synchronization issues in distributed multimedia database systems," Advanced Database Systems, vol. 759, pp 381-396, 1993.

[12] R. Prasad, C. Dovrolis, M. Murray and K. Claffy, "Bandwidth estimation: metrics, measurement techniques, and tools," IEEE Network, vol. 17, no. 6, pp. 27- 35, Nov.-Dec. 2003.

[13] D. C. Plummer, "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.

[14] J. Postel, "Internet Control Message Protocol," IETF, RFC 792, 1981.

[15] R. Kapoor, Ling-Jyh Chen, M. Y. Sanadidi, and M. Gerla, "Accuracy of link capacity estimates using passive and active approaches with CapProbe," Ninth International Symposium on Computers and Communications, pp. 1085- 1090, 2004.

[16] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm," SIGCOMM Comput. Commun. Rev. 27, vol. 3, pp. 67-82, 1997.

[17] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe, "Modeling TCP throughput: A simple model and its empirical validation," ACM SIGCOMM, pp. 303-314, 1998.

[18] Winkler, P., "Random orders". Order 1, 317–331 (1985).

[19] A. Hirales Carbajal, A.Tchernykh, R. Yahyapour, T. Röblitz, J. Ramírez-Alcaraz, J.-L González-García. "Multiple Workflow Scheduling Strategies with User Run Time Estimates on a Grid," Journal of Grid Computing, vol. 10, no. 2, pp. 325-346, 2012.

[20] D. Kliazovich, P. Bouvry, and S. U. Khan, "GreenCloud: A Packet-level Simulator of Energy-aware Cloud Computing Data Centers," Journal of Supercomputing, vol. 62, no. 3, pp. 1263-1283, 2012.