Design and Evaluation of Elastic Media Resource Allocation Algorithms using CloudSim Extensions

Rafael Xavier, Hendrik Moens, Bruno Volckaert and Filip De Turck
Ghent University – iMinds, Department of Information Technology
Gaston Crommenlaan 8/201, B-9050 Gent, Belgium
e-mail: rafael.xavier@intec.ugent.be

***Abstract*—With the maturity of Cloud computing comes research into converting a range of traditionally best effort programs into cloud-enabled services. One such service currently under investigation in the Elastic Media Distribution (EMD) project, is how to enable qualitative, reliable and scalable real-time media collaboration services using proven Cloud technology. While existing best-effort solutions provide plenty of features, they do not provide the quality guarantees and reliability required for critical services in globally distributed corporations. On the other hand, some pricey dedicated solutions do offer these low-delay, reliable cooperation services, but without the benefits that clouds can bring in terms of scalability. In this paper we describe results attained in the EMD project on novel resource provisioning algorithms for a mixture of end-to-end Audio/Video streams with file-based transfers, allowing for configurable trade-offs between service response time and cost. We extended the CloudSim simulator with models allowing us to simulate collaborative interactive sessions (more specifically educational real-time collaboration), and evaluated the performance of our proposed provisioning heuristics. The results show that the proposed dynamic algorithm allows for automated cost-performance trade-off by reducing average total Virtual Machine (VM) cost by a maximum of 58% compared to more naive approaches, while keeping average time for clients to join a meeting in line.**

***Keywords*—Cloud, Media, Collaboration, Elasticity**

I. Introduction

The proliferation and industrial uptake of Cloud technology has given rise to a multitude of novel services and service models (SaaS, PaaS, IaaS, etc.). One such service, which is the core of the research done in the EMD project [1] is professional online Audio/Video (A/V) collaboration. Many non-professional solutions provide such collaboration services over the Internet, but they lack guarantees with regard to quality, latency, and availability (e.g., Skype, Google Hangouts). State-of-the-art solutions for professional low-delay applications (e.g. Cisco Unified Communications [2], Microsoft Skype for Business [3], IBM Sametime [4]) can guarantee a stable, high-quality service, but to realize this they rely on dedicated, static networks and specialized costly equipment. In addition, they:

- impose considerable configuration efforts (and thus cost) to connect media sources over public/private networks (e.g. in remote operating or control rooms),
- lack the necessary security protection to enable remote A/V access to critical data and collaboration over public/private networks,
- fail to guarantee Quality of Experience (QoE) for professional end users at home, in the office, or on the move, and
- cannot exploit the scalability potential offered by public/private cloud technology when local media processing resources become scarce.

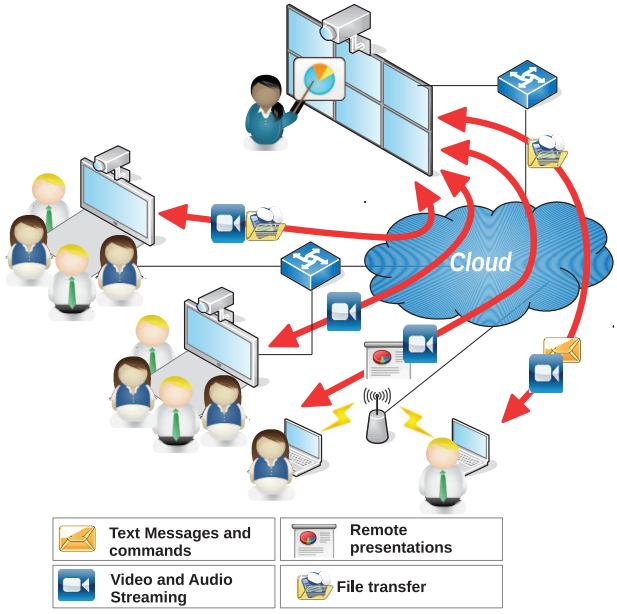

Fig. 1. An educational collaboration scenario: teachers and students engage in interactive and dynamic sessions consisting of multiple A/V and file streams. The connection between various actors is managed and facilitated by cloud-hosted resources responsible for e.g. encoding, decoding, and transcoding.

In order to close this gap, researchers in the EMD project are investigating how Cloud technology can be employed to solve these drawbacks. A collaboration scenario which is of particular importance is that of educational A/V cooperation. In this scenario, illustrated in Figure 1, a tutor is engaging in an interactive session with his or her students, some of whom are physically present while others may be joining remotely. The tutor's video streams (composed of e.g. course material and a camera-captured physical tutor) is sent to all students, and the tutor in turn can decide on interactively switching to specific students' streams (composed of e.g. course material, exercise solutions, and questions they wish to share along with a webcam-captured video stream). Different composition, transcoding, encoding and decoding services, tied together in

978-3-901882-77-7

service workflows need to be deployed and intelligently and automatically managed to guarantee that such a form of real-time collaboration can take place without hick-ups.

One key challenge is how to properly deal with scalability and elasticity: clients will wish to join and leave a collaborative meeting[1] at any time, and meetings can be very dynamic in terms of which content (live / archived video streams, files, annotations, etc.) is streamed to what devices. As the aim is to provide a professional commercial service, guarantees need to be provided in terms of availability and quality of the collaboration experience. To that end, adaptive real-time Cloud resource provisioning algorithms are needed that balance the resource usage (and subsequently price) required for offering such a service with the actual collaboration needs. In order to effectively evaluate the incepted resource provisioning algorithms for a wide range of collaboration scenarios, the CloudSim simulator [5] was extended with bandwidth usage schemes, data transfer models and job workflows.

In this paper we present EMD resource provisioning heuristics that reserve capacity for elastic media services, and present the various extensions which have been developed for CloudSim in order to allow simulation of real-time A/V collaboration services. The remainder of this paper is structured as follows. First, related work is discussed in Section II. Elastic media resource provisioning algorithms are presented in Section III. In Section IV, a detailed view of the proposed CloudSim extensions needed to model the projected scenario is presented. This is followed by an explanation on how the scenarios are generated in Section V, and an evaluation of results in Section VI. Finally, in Section VII we present our conclusions.

II. Related Work

The algorithms and CloudSim extensions presented in this paper are used to solve and evaluate the solution to an elastic media delivery problem. As this problem entails the allocation of Virtual Machines (VMs) within cloud environments, this is a cloud resource provisioning problem. Much work has been done in literature related to cloud resource [6], most of it focusing on allocation of resources within a single data center. In this paper, we however focus on how services are deployed within networks which potentially include multiple small cloud nodes, making it important to take the network into account. Some cloud resource management approaches do focus on multi-cloud deployments [7], [8], [9], [10]. These approaches however do not focus on how VMs hosted in these multiple clouds can be connected to end users, which is critical for the elastic media delivery problem.

More network-focused resource allocation approaches have also been developed over the years. Virtual Network Embedding (VNE) [11] is a concept where virtual networks are mapped onto a physical substrate. This approach focuses on the allocation of VMs to nodes, and the assignment of the interconnections between VMs to network links. While VNE approaches allocate VMs and network resources, and could be simulated and evaluated using the proposed CloudSim extensions, they generally focus on the interconnection of VMs, and not on how clients are connected to these interconnected VMs, which is the focus of this paper.

A similar problem is the Virtual Network Function (VNF) placement problem [12]. VNF placement allocates network functions on cloud nodes within a network, taking a network-centric approach connecting source and sink nodes with a given set of network functions in between. Elastic media delivery is a specialized version of this problem, allocating VMs responsible for A/V processing and requiring very fast reactions. These factors make it important to pre-allocate capacity, ensuring capacity is always present before media streams start, which is not a prerequisite of the general VNF placement problem.

To support the EMD scenario, a simulator supporting both networks and clouds is needed. DaSSF [13] is a highly scalable simulator, which can be used to efficiently simulate network environments. For the purposes of this paper, the simulator would however need to be extended to support Cloud and SDN concepts. NSGrid [14] can be used to simulate Grid computing environments, and uses ns-2 [15] to simulate the underlying network. The underlying packet-based simulation could however lead to scalability issues, especially when simulating large media transfers. In addition, the network simulator would have to be extended to demonstrate all of the required SDN functionality. GreenCloud [16] is a cloud simulator, but is aimed specifically at energy consumption. DCSim [17] can be used to simulate cloud data centers, but does not model the underlying network. CloudSim [5] is commonly used to simulate cloud scenarios, and focuses on the evaluation of resource provisioning systems. As CloudSim isn't packet-based, this enables quicker large-scale evaluations, and allows the user to model the higher-level behaviour of the underlying SDN network. Due to these considerations, we chose to extend CloudSim when evaluating elastic media delivery systems.

This work relates to our previous work on network scheduling in media production networks [18], where we studied network capacity allocation when the network requests are known beforehand, making a distinction between video transfers and file transfers. In this paper we do not assume that the requests are known beforehand. As meetings can be very dynamic, we also consider the dynamic allocation of VMs, and present CloudSim extensions that can be used to evaluate these dynamic scenarios.

III. Elastic Media Resource Provisioning Algorithms

Aiming to provide reliable collaboration experiences by the use of elastic mechanisms, the resources available should be allocated taking into account QoE[2] and cost-efficiency[3]. In this section, we will first describe how the scenarios are modelled, after which we will present a set of heuristic algorithms exhibiting different quality trade-offs.

[1] A meeting, also called session in this paper, is a certain period of time where the participants electronically collaborate and share media like live camera-generated video, audio, files, commands and other interactions

[2] The EMD project uses the time to respond to a request and start a user session as a user experience metric.

[3] Costs must be minimized while keeping in line with QoE requirements.

A. Data Transfer Model

To represent the data transfers illustrated in Figure 1, we use a data transfer model utilizing two types of streams:

- **Fetch** streams, which model a real time transmission between two points, with a constant bitrate and a constant amount of used bandwidth[4].
- **Pre-fetch** streams, which simulate file downloads. While a fetch stream is used to model A/V streams, pre-fetch transfers are used for all other data transfers: inter-client messages, files, presentations, etc.

Each stream is composed of components that are executed in VMs. For fetch streams, **encoders** receive raw data, process it and bring it into the system in a digital form. **Decoders** deliver the content to end-user applications, and **transcoders** transform the data being transmitted to fit characteristics required by the different client devices. For pre-fetch streams, we also employ **encoder** and **decoder** components, which are used to send and receive the transmitted files, but without transcoding.

The duration of fetch streams is determined by when users start and stop the transfer, meaning their duration is always fixed based on user actions. The duration of pre-fetch streams is however variable as it is impacted by the bandwidth available during the transfer. This implies that, whenever a new stream is allocated or deallocated, or whenever topology changes occur[5], the bandwidth allocated to pre-fetch streams may vary, impacting the time when the stream will finish.

B. Algorithm Description

The presented elastic media resource provisioning algorithm consists of two separate steps: first we determine where various components are allocated in the network using a component allocation algorithm. Afterwards, we provision VMs on the available nodes in the network to accommodate the demand, potentially overprovisioning to ensure capacity is present to handle future collaboration requests efficiently.

1) Component Allocation Algorithm: The decision of where to allocate and run components is made by the Algorithm 1. Encoders and decoders are allocated close to the source and the destinations of the stream respectively. However, the decision where to run a transcoder has to be made taking the bandwidth in account, in order to optimize bandwidth usage. If the transcoder output bandwidth is higher than its input (e.g. when the stream is upscaled or transformed in a less bandwidth-efficient format), it makes sense to allocate the transcoder closer to the destination as a way to reduce bandwidth utilization. The same is valid when downscaling: allocation of the transcoder closer to the encoder will lower overall bandwidth usage.

When the decision in what resource pool each component will run has been made, the VM allocation algorithm is invoked to decide how these components will be allocated to VM instances.

Result: According to the bandwidth usage, the resource pool where the component must run or null to not allocate
Data: $src\_pool \leftarrow$ source resource pool identifier
Data: $src\_bw \leftarrow$ bandwidth used before transcoding
Data: $dst\_pool \leftarrow$ destination resource pool identifier
Data: $dst\_bw \leftarrow$ bandwidth used after transcoding
Data: $type(c)$: the type of a component c as encoder, decoder or transcoder
Data: c : the component to be placed
if $type(c) == encoder$ **then**
  return src_pool;
else if $type(c) == decoder$ **then**
  return dst_pool;
else
  if $(dst\_bw/src\_bw) > 1$ **then**
    return dst_pool (upscaling transcoder)
  else if $(dst\_bw/src\_bw) < 1$ **then**
    return src_pool (downscaling transcoder)
  else
    return $null$

Algorithm 1: Algorithm to allocate components taking into account bandwidth usage.

2) VM Allocation Algorithm: The goal of this algorithm is to optimize the allocation of stream components to VMs with the intent to reduce total resource usage cost. A parameterizable collection of VM images with varying CPU, memory, storage and bandwidth capacities should be provided as input. Taking this list of images into account, and the image usage history from previous allocations, Algorithm 2 decides in which VM a specific component will be deployed. VMs can be created both before requests arrive and on-demand. This will have an impact on the cost of providing the service (as more VMs may be created than the current need requires) and may impact the QoE provided to clients (as starting to deploy a VM when a request arrives will result in considerable delays). To address this, four different algorithm behaviours were implemented:

- **ALL:** based on historical component requirements, all VMs needed to properly run the collaboration service are instantiated in the beginning of the session. A reduction in Average joining Time is expected, but also an increment in Average VM Cost, since the VMs will run longer during the session.
- **NONE:** contrasting with ALL, NONE means that all VMs will be created on-demand, raising the time to deploy a stream but with expected reductions in Average VM Cost.
- **STATIC:** aiming for a better cost-benefit balance, a configurable percentage of the historically requested number of VMs is deployed when starting the session, reserving some VMs at the start and deploying the rest on-demand, based on user requests.
- **DYNAMIC:** to improve results obtained with the STATIC behaviour, the amount of VMs is monitored in order to dynamically keep a configurable percentage of the VM usage history instantly available when requested.

[4]Simulating variable bitrate streams is desirable for future work, but is out of scope for this paper.

[5]Topology changes are out of the scope of this paper, but are important to handle network and component failures in future work.

Result: A managed pool of VMs and the choice of which VM to run each component
Data: $templates \leftarrow$ list of VM Image types available for instantiation
Data: $type$: algorithm type
Data: $history(i)$: the average amount of VMs used in previous sessions for a image i
Data: $vm\_pool(i)$: list of instantiated VMs for a image i
Data: $new(i, c)$: create c VMs for image i
Data: $size(l)$: size of a list l
Data: $next(i)$: the next available VM for a image i
Data: P_s : percentage of the VM Image usage history to create when using STATIC
Data: P_d : percentage of the VM Image usage history to keep available when using DYNAMIC
Procedure `init()`
  for *each image* $i \in templates$ **do**
    if $type == ALL$ **then**
      $new(i, history(i))$;
    else if $type == STATIC$ **then**
      $new(i, history(i) \times P_s)$;
Procedure `vmRequest()`
  if $type == DYNAMIC$ **then**
    for *each image* $i \in templates$ **do**
      $diff \leftarrow (history(i) \times P_d) - size(vm\_pool(i))$;
      if $diff \geq 1$ **then**
        $new(i, diff)$;
  if $size(vm\_pool(i)) > 0$ **then**
    return $next(i)$;
  else
    return $new(i, 1)$;

Algorithm 2: Algorithm to populate a VM pool and to select or, if unavailable, deploy a VM to run each collaboration processing component.

IV. CloudSim Extensions

CloudSim is an event-based simulator that can be used to simulate large-scale cloud scenarios and evaluate resource allocation algorithms. Figure 2 presents the architecture of the original simulator [5]. Highlighted components were added or modified as part of the extensions proposed in this paper.

CloudSim is divided in two subsystems: user-defined code, to parametrize and instantiate the cloud and customized provisioners, and core CloudSim code, which defines basic constructs and runs the simulation. In CloudSim, the simulated cloud environment is modelled as a chain of **datacenters**, **hosts** and **VMs**. A VM runs **cloudlets**, which represent tasks according to allocation decisions made by resource provisioners and schedulers. The user-defined **broker** requests resources from the cloud and the **network topology** manages the network and the delay between all nodes. CloudSim allows the definition of complex scenarios, but lacks some functionalities provided by the items represented in Figure 2 and detailed next:

- **Data Streams:** representing an Audio/Video stream, including parameters like source, destination, stream type, duration in seconds, priority, amount of data to transfer, etc..

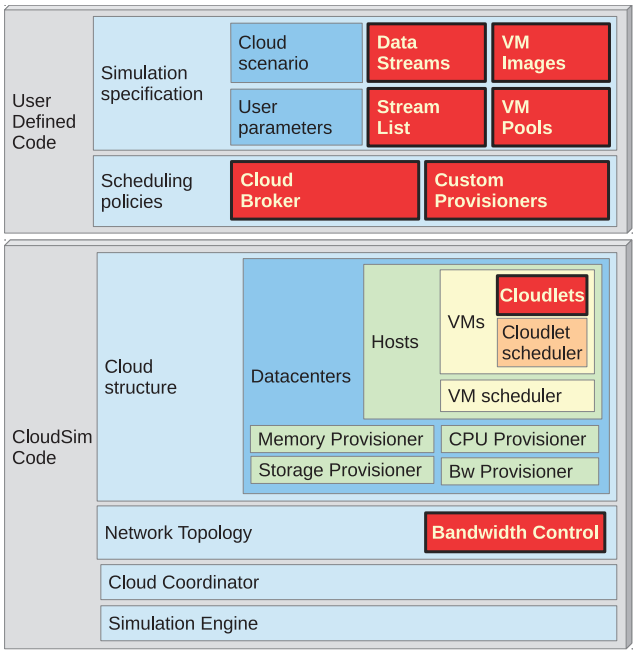

Fig. 2. CloudSim core architecture and highlighted extensions.

- **Stream List:** a list of Data Stream instances are used to model a collaborative meeting, also called **session**.
- **VM Image Templates:** a set of VM templates that can be deployed, each with a customizable amount of resources like memory, CPU, storage and reserved network bandwidth.
- **VM Pools:** each datacenter has a pool of VMs, and each pool is populated with VMs deployed based on available VM Images templates. Algorithm 2 defines which VM image template must be deployed at what time or selects an already available VM.
- **Custom Provisioners:** implementations of the custom resource allocation algorithms described in section III.
- **Cloudlets:** tasks that the simulator deploys and executes on the VMs. Since streams are real-time activities, this class was extended to support Cloudlets processing real-time streams until receiving a client event to stop.
- **Bandwidth Control:** CloudSim does not, by default, support bandwidth capacity limitations in the underlying network. The bandwidth control extension makes it possible to keep track of available end-to-end bandwidth and to make bandwidth reservations.

The components highlighted in Figure 2 were added or modified to implement three extensions to CloudSim: A/V stream modelling (already detailed in subsection III-A), models for bandwidth availability and the message exchange mechanism allowing for synchronization of components responsible for streaming. The next two subsections will explain how to distribute the available network bandwidth to fetch and

pre-fetch streams but also the component synchronisation mechanic required to allow for streaming.

A. Data Transfer Workflow

The stream components are represented in the simulator as Cloudlets, CloudSim entities that represent tasks to be processed on VMs. For each component of the stream one cloudlet will be instantiated and submitted. As the VMs may be located in different data centers, depending on the source and destination of the stream, the interaction between the broker, the data centers and the network control layer must be synchronized, as shown in Figure 3.

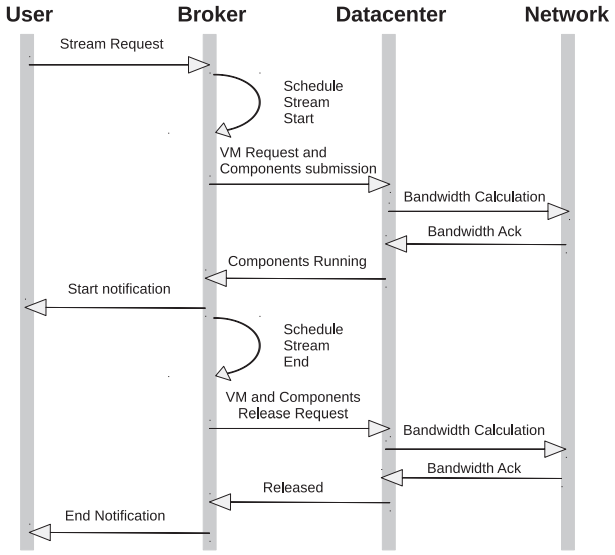

Fig. 3. Sequence diagram showing initialization / synchronization of a stream.

After being requested to start a stream, the broker requests the stream components for the data centers selected to host each component. With all components in place, the broker puts the stream in the running state and schedules a new event to deallocate the components after the desired stream duration. When the stream finishes, the broker releases the allocated resources and changes the stream state to finished. Figure 4 illustrates the behaviour of each component being instantiated in parallel, showing that the stream can only start when all components are in place. Encoding, decoding and transcoding components are requested in parallel. With these three deployed and running, the stream is started for the user. When the user closes the collaborative session, the components are deallocated.

B. Bandwidth Usage Simulation

One of the messages present in the sequence diagram described in Figure 3 is *bandwidth calculation*. This step is responsible for calculating available and reserved network bandwidth capacity after each event.

CloudSim originally only uses the network topology to determine network delays. This delay is added to the simulation time to represent the time needed to transfer data when

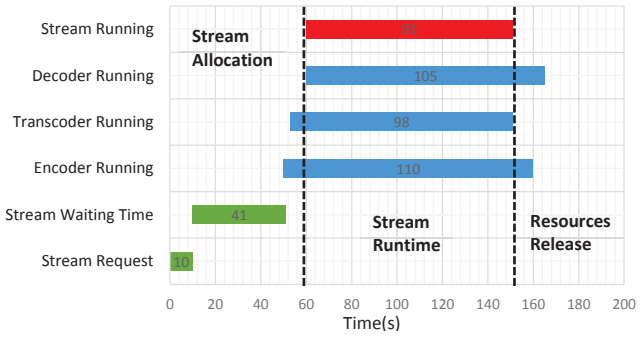

Fig. 4. Example of the time relation between a stream and its respective components.

a cloudlet is submitted to run in a remote data center. To calculate the delay, a Floyd-Warshall algorithm is used, resulting in a delay matrix and a next-hop matrix. The next-hop matrix provides the shortest path available for each source-destination pair of topology nodes and the delay matrix information to calculate its delay.

The proposed approach to calculate and control the bandwidth usage is similar: using the already available Floyd-Warshall matrix, for each pair of network nodes a list of topological links is created, setting up a 3D matrix. Algorithm 3 controls the bandwidth allocation to simulate the behaviour of the scenario network. The algorithm takes as input a list of **links** and **streams**, and iteratively distributes the available bandwidth between all the streams, until either the stream demand is achieved or until the network link capacity limits are reached. When all streams have obtained their required bandwidth or fail when there is insufficient bandwidth capacity, the algorithm terminates. As pre-fetch streams make use of variable bandwidth capacity, they contend for capacity and share all available bandwidth equally amongst all active pre-fetch streams.

With the proposed extensions in place, multiple scenario representations must be generated and simulated to evaluate the proposed definitions and implementation. The next section will describe the Scenario Generator, a tool developed to automate educational collaboration scenario generation, parametrized by the simulation end-user.

V. Scenario Generation

The collaboration scenarios to be simulated using the extended CloudSim simulator can be very heterogeneous and dynamic: they contain multiple clients connected through public/private LAN/WAN networks with a variety of cloud resources (provided by different vendors with differing resource provisions and pricing) and per-client stream/file fetching needs. In order to evaluate resource provisioning algorithms in a variety of realistic scenarios, a scenario generator was developed. The generator is part of a projected simulation chain composed of five steps, as shown in Figure 5. The user interfaces are drawn in light grey and consist of simulation parametrization and reporting.

The simulator, consisting of CloudSim along with the extensions proposed in section IV, receives the scenario to

Result: calculates bandwidth distribution for all streams
Data: $status(s)$: status as $pending$, $failed$ or $success$
Data: $needed(s)$: bandwidth needed to run a stream s
Data: $alloc(x)$: bandwidth currently allocated for a stream/link x
Data: $type(s)$: type of a stream s as $fetch$ or $prefetch$
Data: $capacity(l)$: bandwidth capacity of a link l
Data: $src(s)$: source of a stream s
Data: $dst(s)$: sink of a stream s
Data: $L_{s,d} \leftarrow$ topology link list that composes the path between s and d
Data: $S \leftarrow$ active streams list
Data: $step \leftarrow$ value to increment on each allocation cycle
Data: $continue \leftarrow true$
while $continue == true$ **do**
  $continue \leftarrow false$;
  for *each stream* $s \in S$ **do**
    $allocated \leftarrow false$;
    for *each link* $l \in L_{src(s),dst(s)}$ **do**
      if $status(s) == pending$ **then**
        $alloc(l) \leftarrow alloc(l) + step$;
        $allocated \leftarrow true$;
        if $capacity(l) \leq alloc(l) + step$ **then**
          if $type(s) == fetch$ **then**
            if $alloc(s) < needed(s)$ **then**
              $status(s) \leftarrow failure$;
          else
            $status(s) \leftarrow sucess$;
        if $type(s) == fetch$ **then**
          if $alloc(s) \geq needed(s)$ **then**
            $status(s) \leftarrow success$;
    if $status(s) == pending$ **then**
      $continue \leftarrow true$;
      if $allocated == true$ **then**
        $alloc(s) \leftarrow alloc(s) + step$;

Algorithm 3: Algorithm calculating bandwidth in use by each stream to simulate network contention.

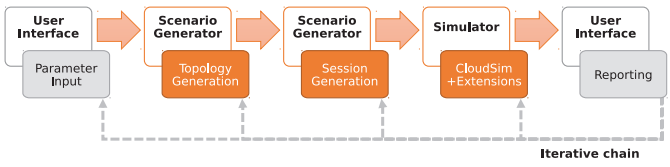

Fig. 5. Simulation Chain

be simulated, simulates it, and sends the results to the user interface. The scenario definition is procured by the Scenario Generator, based on parameters tuned in the user interface. The Scenario Generator consists of two components: a Topology Generator and Session Generator.

A. Topology Generator

CloudSim loads the underlying network topology from a text file, which is the output format of the widely used BRITE Internet topology generator [19]. As we want to generate pools of resources connected to nodes of the topology, resource information is generated and attached to randomly selected nodes from the generated topology.

The proposed topology generator follows four steps. (1) First, it receives parameters from the user, such as resource pool definitions and the number of nodes of the desired topology. (2) Next, it executes the BRITE generator to obtain a base topology. (3) Afterwards it generates the resource pools and attaches them to the base topology. (4) Finally, the topology generator saves all this info as input for the session generator, described next.

B. Session Generator

A **session** is a set of streams and data transfers composed to model the data flows between users during collaborative meetings. When simulating a classroom with students and a teacher, a session may be composed of a list of video streams between the teacher and the students, the course presentation being sent by the teacher as a pre-fetch stream to each student and sporadic messages between the teacher and students to model questions, answers and file-based data exchanges.

Based on session templates defined by the simulation user and a chosen network topology, the session generator provides a list of streams and data transfers to model the session behaviour and compiles all information required to run the simulation. A small sample of data stream properties is shown in Listing V-B, illustrating one fetch and one pre-fetch stream. The shown values are generated based on distributions provided by the simulator user.

```
<stream from="Teacher_1"
    to="Local_Student_1"
    type="prefetch"
    receiver_mips="26"
    sender_mips="26"
    size_kbytes="572050"
    startdelay="23549216" (...)/>

<stream from="Local_Student_22"
    to="Teacher_1"
    bw_kbits="3000"
    duration="6893509"
    type="fetch"
    receiver_mips="64"
    sender_mips="77"
    startdelay="1522535" (...)/>
```

Listing 1. Sample of generated streams modelling collaborative user sessions.

VI. Evaluation Setup and Results

To show that the proposed CloudSim extensions work and aid in evaluating the performance of resource provisioning algorithms for the scenarios described in Section I, educational A/V collaboration scenarios were generated and evaluated. The results will be presented and discussed in the following subsections.

A. Simulated Scenario

As detailed in Figure 6, the evaluation scenario is composed of a virtual class-room S, containing students, and another room T, where the teacher is located and ready to present

the lessons. This distributed configuration represents situations including remote teaching, or situations where (some) students attend the class remotely.

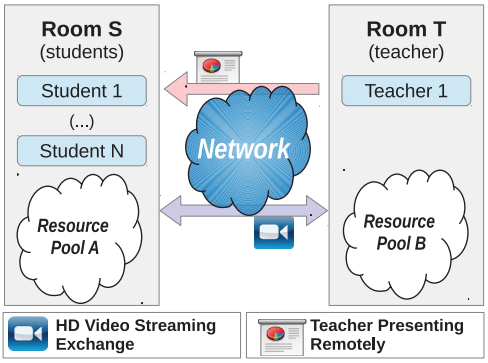

Fig. 6. The remote teaching scenario used to evaluate the CloudSim extensions and resource allocation algorithms.

Following the proposed simulation chain, the generated scenario is submitted to the extended CloudSim simulator. To evaluate the results, some metrics were defined.

B. Simulation Metrics

When allocating resources in the presented scenario, both the deployment cost and QoE must be taken into account. Because of this, we consider the following metrics:

- **Average Join Time:** This metric comprises the time between the request of a stream (made by the user) and the moment when the broker has all components running and can start the stream. Considering N the count of streams that constitute a session, T_{req_n} the time spent to request components for a stream n, S_{vm_n} the set $(T_{vm_1} \dots T_{vm_n})$ of time values to create or select VMs to run the components of a stream n, S_{conf_n} the set $(T_{conf_1} \dots T_{conf_n})$ of time values spent to configure each component to run in the VMs selected for a stream n and $Max(S)$ the major value of a set S, Equation 1 defines the Average Join Time:

$$\frac{1}{N}\sum_{n=1}^{N}(T_{req_n} + Max(S_{conf_n}) + Max(S_{vm_n})) \quad (1)$$

- **Average VM Cost:** The average VM cost represents the average of all costs related to resource usage when running all stream components of the current session. For a session with x streams, each one with Y_x stream components, M_k, S_k, C_k and B_k, respectively represent, memory, storage, CPU and bandwidth costs for a stream component k. Adding α, β, γ and δ as adjustment factors to be calibrated, the total cost is given by Equation 2:

$$\frac{1}{X}\sum_{x=1}^{X}(\frac{1}{Y_x}\sum_{y=1}^{Y_x}(\alpha M_y + \beta S_y + \gamma C_y + \delta B_y)) \quad (2)$$

TABLE I. PARAMETRIZED VM IMAGES

Image	Storage	CPU	Memory	Monthly Cost
Template_01	4 GB	40 MIPS	128 GB	$3.94
Template_02	8 GB	80 MIPS	256 GB	$7.88
Template_03	16 GB	160 MIPS	512 GB	$15.76
Template_04	32 GB	320 MIPS	1024 GB	$31.52
Template_05	64 GB	640 MIPS	2048 GB	$63.05
Template_06	128 GB	1280 MIPS	4196 GB	$127.03

With the simulation scenario defined and metrics in place, we evaluate the resource allocation heuristics introduced in Section III. The obtained evaluation results are detailed next.

C. Simulation Results

The used network comprises 100 nodes with transmission delay and available bandwidth uniformly distributed by the Topology Generator between $[20, 200]ms$ and $[0.1, 1]Gbit/s$, respectively. Attached to this network, two resource pools were defined: a *Teacher* pool, with 1 user, and a *Students* pool, with user count parameterised between 10 to 100. To deploy VMs in the resource pools, six types of VM images were defined, as detailed in Table I. As mentioned in Section III, the duration needed for deploying new VMs differs significantly from the duration needed to start executing a component on an existing VM instance. To this end, we define two variables, T_{deploy} and T_{config}, that determine the duration for instantiating new VMs and the duration for configuring existing VMs respectively. For the simulations, the values of T_{deploy} and T_{config} were defined as uniform distributions between $[45s, 75s]$ and $[0.5s, 2s]$ respectively.

The VM costs were based on the Amazon EC2 image c3.8xlarge [20], with a monthly price of $1.680 to provide 32 VCPU, 60 GB of memory and 320 GB. This cost was divided equally between storage, memory and CPU, converted to prices per MB/hour and MI/hour [21] and used to parameterize the simulator. The cost of each VM template, calculated using these parameters, is shown in table I.

With the infrastructure defined, the Session Generator was configured to generate two HD sessions and four file transfers between each Teacher-Student pair. For a HD transmission, stream / collaboration duration was uniformly distributed between $[6800, 7000]$ seconds and parametrized to use CBR 3 Mbit/s bandwidth, with a student able to join in at any point in the 4 hour collaborative session. The file transfers were configured to send data uniformly between $[0.5, 10]$ MB as fast as possible. The simulations were executed on a dedicated VMWare ESXi virtual machine with 8 GB of memory, 8 VCPUs and operational system Linux Ubuntu 14.04 64 bits, and none of the resource provisioning heuristics took longer than 0.5ms to come up with a solution.

Figure 7 shows the evolution of the Average Join Time when the number of users increases. As expected, ALL has the best results and NONE the worst, with STATIC and DYNAMIC in-between (STATIC algorithm parameterised to 50% of historical VM usage instantiated up-front, DYNAMIC parameterised to aim for 10% extra VMs available for immediate use by clients wishing to join the collaboration session). This due to ALL not needing to deploy any VMs on-demand, NONE deploying all VMs on-demand with a consequent time

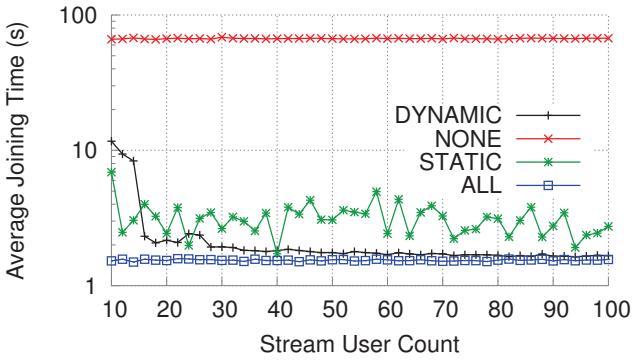

Fig. 7. Average Join Time by number of clients for the simulated scenario.

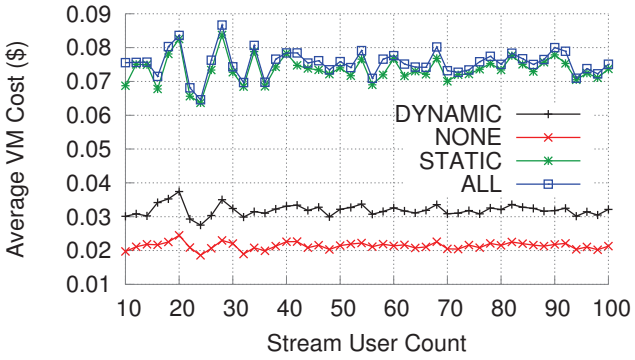

Fig. 8. Average VM Cost by number of clients for the simulated scenario.

penalty while the other algorithms, STATIC and DYNAMIC, try to reduce this penalty by having a percentage-based amount of VMs available for streaming in the VM Pool.

When the Average VM Cost is compared as shown in Figure 8, we observe that NONE has lowest cost and that DYNAMIC is a good alternative to it. DYNAMIC tries to keep an amount of VMs available in the VM Pool to fit the demand by predicting required VMs based on historical usage.

To show the relation between both metrics, we present Figure 9, which shows Average Join Time compared to Average VM Cost (and showing error margins). As expected, NONE and ALL present boundary behaviours, the first with high Average Join Time, and the second raising Average VM Cost, since all VMs are started when the session begins. STATIC presents a good Average Join Time, but with a high cost. However, DYNAMIC is a better cost-benefit choice: the average VM Cost was reduced by 58% while keeping acceptable Average Joining Times.

To end the analysis, it is important to mention that variations observed in the graphs are related to the VM template choices used during the simulation: as templates haves boundaries that define whether they are sufficient to run a cloudlet, or whether the next less resource-constrained VM template must be chosen. As each user has a uniquely generated resource consumption pattern, these VM boundaries may not be sufficient to guarantee that the same VM type will be used for similar streams. This can cause small fluctuations in user demand to result in the need for using a different template,

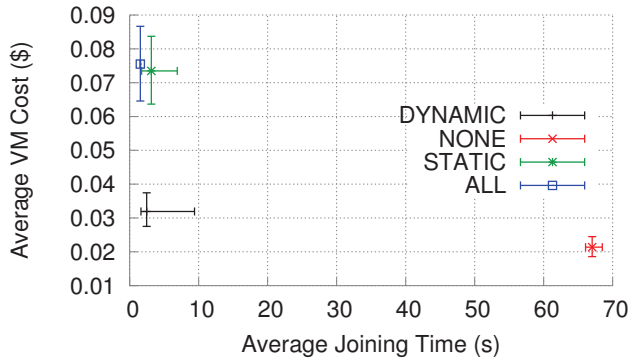

Fig. 9. Average joining time compared to average VM cost.

and therefore lead to an increase in cost. This behaviour can be avoided by offering more fine-grained VM templates.

VII. Conclusions and Future Work

CloudSim is a powerful tool to simulate cloud environments and resource allocation algorithms. When faced with the need to take into account network contention between end-to-end data flows and the behaviour of media stream workflows, multiple extensions to the simulator were proposed and evaluated in this paper.

These extension integrate methods for calculating used bandwidth between any two nodes of the topology, modelling media-intensive applications as streams, and a collection of component synchronization mechanisms between CloudSim entities. Together with a scenario generator, they allow us to simulate and evaluate media collaboration scenarios, collect results and evaluate the main contribution of this work: to develop acceptable cost-benefit heuristic-based algorithms to allocate cloud resources in support of collaborative real-time media services. The presented heuristics offer varying cost and performance trade-offs, allowing to select the best management algorithm depending on the quality requirements of the session at hand.

With an algorithm runtime smaller than $0.5ms$ for all simulations, the choice for CloudSim and flow-based bandwidth calculation (as opposed to packet-based simulation) is deemed successful for simulating the targetted scenarios, attaining the scalable performance required for future large-scale scenario evaluations. Applying the proposed models and simulation chain to the scenarios described in this paper, we were able to reach good cost-performance trade-offs, reducing average VM cost by 58% while keeping acceptable join times when a user wishes to start or join a session.

The next research steps will be to introduce resource failures and implement job deadlines (whereby Clients are no longer interested in joining a session if the time to set up said session takes too long), and to extend our heuristics to take these aspects into account (focussing on the trade-offs made between cost, performance and reliability).

Acknowledgment

The research described in this paper is partially funded by the iMinds EMD project.

REFERENCES

[1] iMinds, "EMD: Elastic media distribution for online collaboration," 2015. [Online]. Available: http://www.iminds.be/en/projects/2015/03/11/emd

[2] Cisco, "Cisco hosted collaboration solution (HCS)," 2015. [Online]. Available: http://www.cisco.com/web/solutions/hcs/index.html

[3] Microsoft, "Skype for business, formerly lync," 2015. [Online]. Available: http://products.office.com/en-us/skype-for-business/online-meetings

[4] IBM, "IBM sametime," 2015. [Online]. Available: http://www-03.ibm.com/software/products/en/ibmsame

[5] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw. Pract. Exper.*, vol. 41, no. 1, pp. 23–50, Jan. 2011. [Online]. Available: http://dx.doi.org/10.1002/spe.995

[6] B. Jennings and R. Stadler, "Resource Management in Clouds: Survey and Research Challenges," *Journal of Network and Systems Management*, Mar. 2014. [Online]. Available: http://link.springer.com/10.1007/s10922-014-9307-7

[7] G. Koslovski, S. Soudan, P. Goncalves, and P. Vicat-Blanc, "Locating virtual infrastructures: Users and inp perspectives," in *2011 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, May 2011, pp. 153–160.

[8] M. Alicherry and T. Lakshman, "Network aware resource allocation in distributed clouds," in *IEEE INFOCOM*, March 2012, pp. 963–971.

[9] M. Steiner, B. G. Gaglianello, V. Gurbani, V. Hilt, W. Roome, M. Scharf, and T. Voith, "Network-aware service placement in a distributed cloud environment," *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 73–74, Aug. 2012. [Online]. Available: http://doi.acm.org/10.1145/2377677.2377687

[10] Y. Zhu, Y. Liang, Q. Zhang, X. Wang, P. Palacharla, and M. Sekiya, "Reliable resource allocation for optically interconnected distributed clouds," in *2014 IEEE International Conference on Communications (ICC)*, June 2014, pp. 3301–3306.

[11] A. Fischer, J. Botero, M. Till Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 1888–1906, Fourth 2013.

[12] H. Moens and F. De Turck, "VNF-P : A Model for Efficient Placement of Virtualized Network Functions," in *Proceedings of the 10th International Conference on Network and Service Management (CNSM 2014)*, 2014, pp. 418–423.

[13] J. D. Chalfant, "Parallel dassf discrete-event simulation without shared memory," Hanover, NH, USA, Tech. Rep., 1999.

[14] B. Volckaert, P. Thysebaert, F. De Turck, P. Demeester, and B. Dhoedt, "Evaluation of grid scheduling strategies through a network-aware grid simulator," *PDPTA'03: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED PROCESSING TECHNIQUES AND APPLICATIONS, VOLS 1-4*, pp. 31–35, 2003.

[15] T. Issariyakul and E. Hossain, *Introduction to Network Simulator NS2*, 2nd ed. Springer Publishing Company, Incorporated, 2011.

[16] L. Liu, H. Wang, X. Liu, X. Jin, W. B. He, Q. B. Wang, and Y. Chen, "Greencloud: A new architecture for green data center," in *Proceedings of the 6th International Conference Industry Session on Autonomic Computing and Communications Industry Session*, ser. ICAC-INDST '09. New York, NY, USA: ACM, 2009, pp. 29–38. [Online]. Available: http://doi.acm.org/10.1145/1555312.1555319

[17] G. Keller, M. Tighe, H. Lutfiyya, and M. Bauer, "Dcsim: A data centre simulation tool," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, May 2013, pp. 1090–1091.

[18] M. Barshan, H. Moens, J. Famaey, and F. De Turck, "Algorithms for Advance Bandwidth Reservation in Media Production Networks," in *Proceedings of the 14th IFIP/IEEE International Symposium on Integrated Network Management (IM 2015)*. IEEE, may 2015, pp. 183–190.

[19] A. Medina, I. Matta, and J. Byers, "Brite: A flexible generator of internet topologies," Boston, MA, USA, Tech. Rep., 2000.

[20] "Amazon ec2 images," 2015. [Online]. Available: http://aws.amazon.com/pt/ec2/instance-types/

[21] "Benchmarking the new amazon c4 instances," 2015. [Online]. Available: http://www.cmips.net/tag/intel-xeon-e5-2666-v3-2-90ghz/