Future Generation Computer Systems 37 (2014) 297–308

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Towards understanding the runtime configuration management of do-it-yourself content delivery network applications over public clouds

Zheng Li [a], Karan Mitra [c], Miranda Zhang [b], Rajiv Ranjan [b,*], Dimitrios Georgakopoulos [b], Albert Y. Zomaya [d], Liam O'Brien [e], Shengtao Sun [f]

[a] School of Computer Science, Australian National University and NICTA, Australia
[b] CSIRO Computational Informatics, Canberra, Australia
[c] Luleå University of Technology, Sweden
[d] University of Sydney, Sydney, Australia
[e] Geoscience Australia, Canberra, Australia
[f] Yanshan University, Qinhuangdao, China

HIGHLIGHTS

- We present the concepts and factors related to DIY content management applications.
- We show a prototype of the Cloud-hosted DIY content management applications.
- A typical methodology is demonstrated for evaluating Cloud-based application systems.
- Extensive experiments were conducted for understanding the runtime configurations.

ARTICLE INFO

Article history:
Received 29 August 2013
Received in revised form
12 October 2013
Accepted 3 December 2013
Available online 17 January 2014

Keywords:
Application runtime configuration
Cloud services evaluation
Content delivery network
Experimental design and analysis
Evaluation methodology
Mediawise cloud content orchestrator
Public clouds

ABSTRACT

Cloud computing is a new paradigm shift which enables applications and related content (audio, video, text, images, etc.) to be provisioned in an on-demand manner and being accessible to anyone anywhere in the world without the need for owning expensive computing and storage infrastructures. Interactive multimedia content-driven applications in the domains of healthcare, aged-care, and education have emerged as one of the new classes of big data applications. This new generation of applications need to support complex content operations including production, deployment, consumption, personalization, and distribution. However, to efficiently provision these applications on the Cloud data centres, there is a need to understand their runtime resource configurations. For example: (i) where to store and distribute the content to and from driven by end-user Service Level Agreements (SLAs)? (ii) How many content distribution servers to provision? And (iii) what Cloud VM configuration (number of instances, types, speed, etc.) to provision? In this paper, we present concepts and factors related to engineering such content-driven applications over public Clouds. Based on these concepts and factors, we propose a performance evaluation methodology for quantifying and understanding the runtime configuration of these classes of applications. Finally, we conduct several benchmark driven experiments for validating the feasibility of the proposed methodology.

1. Introduction

CISCO (a network technology giant) predicts that by 2016, 90% of internet traffic will be multimedia content (3D images, high resolution video, and audio). In addition to entertainment and advertising applications, the new multimedia content-driven applications in the domain of healthcare, aged-care and education will contribute significantly to this traffic. The new applications' contribution to the traffic will be due to their unprecedented processing (storage, distribution, and indexing) requirements for hundreds of petabytes of content. In the *healthcare domain*, live as well as archived videos will be used as a medium to educate patients about the aftercare treatment (follow-ups), once the patient is at home. This will include video instructions about how to change the dressing on a healing wound or how to brush their teeth after having braces installed. Other scenario from the healthcare domain

* Corresponding author.
E-mail address: rranjans@gmail.com (R. Ranjan).

0167-739X/$ – see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.future.2013.12.019

will arise from the problem of managing petabytes of multimedia content produced by advanced medical imaging devices. In conjunction with traditional X-rays, medical imaging can now delve deeper into the human body, discovering and analysing smaller and smaller details. In the *aged-care domain*, health professionals will rely on real-time or recorded video feeds from patient's home to monitor clinical signs and indicators such as skin colour, moods and activities to determine whether a patient is utilizing devices and medications appropriately. Finally, in the *education domain*, students will need to have the opportunity to access teachers from home; especially, the students in rural or remote areas need an opportunity to be able to receive interactive lessons or instructions via live video streaming from specialized teachers or trainers who are not available locally.

In the aforementioned application scenarios, hundreds of petabytes of multimedia content will be generated which will be required to be processed (stored, distributed, and indexed with a schema and semantics) efficiently in a way that does not compromise end-users' Quality of Service (QoS) in terms of content availability, content search delay, content distribution delay, etc. Many of the existing ICT systems (such as Content Delivery Networks (CDNs) provided by Akamai, Amazon CloudFront, Limelight Networks, etc.) that store, distribute, and index hundreds of petabytes of multimedia content either fall short of this challenge or do not exist. Hence, there is a need for powerful and sophisticated software tools and technologies that can support scalable storage and fast content distribution, while ensuring QoS on a case-by-case basis.

In our previous work [1,2], we presented MediaWise Cloud Content Orchestrator (MCCO)—a novel system that facilitates Do-It-Yourself (DIY) CDN application orchestration for simplifying the management of multimedia content (e.g., audio and video) using public Cloud services. Unlike existing commercial CDN providers such as Limelight Networks and Akamai, MCCO eliminates the need to own and manage expensive infrastructure while facilitating content owner requirements pertaining to price, privacy and QoS. It offers enhanced flexibility and elasticity as it supports the pay-as-you-go model. MCCO content orchestration operations include: (i) production: create and edit; (ii) storage: uploading and scaling of storage space; (iii) indexing: keyword-based content tagging and searching; and (iv) distribution: streaming and downloading. Similar to any other Cloud application, MCCO suffers from performance unpredictability due to many unknown factors. The availability, load, and throughput of Cloud services, for example, CPU, storage, network and software appliances can vary in unpredictable ways. Thus, the assurance of QoS targets for the DIY CDN applications can be challenging.

Alhamazani et al. [3] discussed that QoS uncertainty is the chief technical obstacle to successful adoption of Cloud computing. The recent very high-profile crash [4] of Amazon EC2 cloud, which took down the enterprise applications of many SMEs, is a salient example of unpredictability in Cloud environments. Some applications were down for hours, others for days. Theoretically, the elasticity provided by Cloud computing can accommodate even unexpected changes in capacity, adding Cloud services when needed, and reducing them during the periods of low demand. However, the decisions to adjust capacity must be made frequently, automatically and accurately to be cost-effective. Hence, understanding the relationships between application workload, Cloud service configuration, and QoS delivered to end-users is mandatory. To understand such relationships, there is a need to conduct several experimental evaluation studies in a systematic way where a systematic approach may involve the following steps: (i) requirement recognition; (ii) service feature identification; (iii) QoS metrics and benchmarks listing; (iv) metrics and benchmarks selection; (v) experimental factors listing; (vi) experimental factors selection; (vii) experimental design; (viii) experimental implementation; (ix) experimental analysis; and (x) conclusion and reporting. However, these steps are specific to the type of applications (DIY CDN, web applications, and the like) being evaluated as well as the target computing infrastructure (web services, Clouds, grids, and the like).

In this paper, we: (i) present concepts and factors related to hosting DIY content management applications over public Clouds; (ii) present extensive performance evaluation techniques for quantifying and understanding the runtime configuration management of DIY content management applications; and (iii) conduct several benchmark driven experiments for understanding the runtime configuration of DIY content management applications and service configuration (virtual machines, BLOB storage, Cloud location, pricing, etc.) offered by public Clouds.

The remainder of this paper is organized as follows. Section 2 summarizes the existing work related to CDNs and Cloud service evaluation methodology. Section 3 briefly introduces the MediaWise Cloud. The comprehensive methodology of evaluating MediaWise Cloud is specified step by step in Section 4, with reporting the early-stage experimental results. Conclusions and some future work are discussed in Section 5.

2. Related work

Due to recent emergence of the Content Distribution Networks (CDN), increasing attention has been paid to their performance characteristics [5]. For example, Johnson et al. [6] conducted one of the earliest CDN performance evaluations to compare different commercial products. More specifically, performance evaluation has been employed to study different CDN architectures [7]; and some other studies used to study performance evaluation to identify the best algorithm that met hierarchical streaming requirements [8]. Given the emerging computing paradigm—Cloud computing, researchers and practitioners have started using network virtualization and Cloud techniques to satisfy CDN requirements. In particular, suitable Cloud resources are used to help provide viable and cost-effective solutions for realizing CDN services [9]. Since Cloud usage requires deep understanding of how the relevant Cloud services may (or may not) match particular demands, Cloud services evaluation would act as a crucial part of CDN performance evaluation.

Compared to the traditional CDN systems, the performance evaluation of Cloud-based CDN systems would be more challenging. There are two main reasons for this. First, the back-ends (e.g., configurations of physical infrastructure) of Cloud services are normally uncontrollable (often invisible) from the perspective of consumers. Unlike consumer-owned computing systems, Cloud users have little knowledge and control over the precise nature of Cloud services even in a "locked down" environment [10]. Evaluations in the context of Cloud computing are then inevitably more challenging than that for systems where the customer is in direct control of all aspects [11]. Second, the open indicators (e.g., compute units, memory size, service price, etc.) often lack the provision of comprehensive information about a service regarding specific tasks [12]. As a result, there could be uncertainty in the runtime of Cloud services due to the various Quality of Service (QoS) consumption requirements of different applications [13]. Consequently, service evaluation would be one of the prerequisites of employing Cloud-based systems.

When it comes to performance evaluation of Cloud-based CDN systems, it has been recognized that Cloud services evaluation [14–16] belongs to the field of experimental computer science [11] which requires a suitable evaluation methodology as a strategic role in directing experimental studies [17]. An evaluation methodology instructs a complete evaluation implementation that may cover various aspects, for instance, workload selection, experimental design, and result analysis [17]. Therefore, a concrete methodology adopted in Cloud service evaluation should distinguish

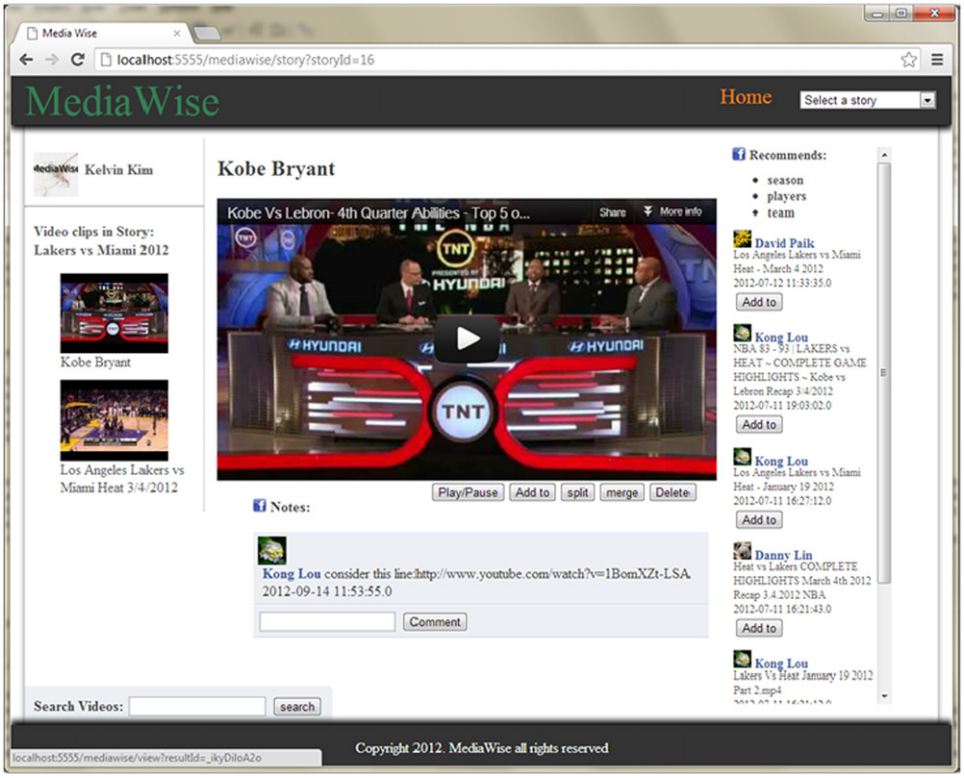

Fig. 1. Social Content Authoring and Sharing Platform's (SCASP) web interface. This web interface can be used for commenting and annotating then media content, content recommendation, consumption and authoring.

between the aforementioned detailed steps [18–21]. Stantchev [11] extended the ASTAR method [22] and specifically suggested a five-step methodology (Identify benchmark, Identify configuration, Run tests, Analyse, and Recommend) for evaluating Cloud services. A more detailed evaluation methodology was specified in [12], which used the business process modelling notation to describe the general steps of developing, executing, and evaluating a Cloud benchmark suite.

However, according to our systematic literature review [23], most evaluators did not strictly define or specify their evaluation steps, not to mention using a sound methodology to guide Cloud service evaluation. Although the existing evaluation implementations must have followed particular approaches, different approaches described in different evaluation reports vary, and even with flawed considerations. For example, the evaluation methodology has been treated as experimental setup and/or preparation of experimental environment [24] where some authors only focused on metrics [25], while others only highlighted benchmarks [26] when specifying their evaluation approach. In these studies, an inappropriate concern was to separate evaluation metrics and experimental implementation from the corresponding methodology [27]. Furthermore, even in the studies with concrete Cloud service evaluation methodologies [12,11], some important steps like the selection of metrics and experimental factors were missed out.

Therefore, in this paper, we decided to employ CEEM [28] to investigate the runtime performance of MediaWise Cloud. CEEM is a practical evidence-based methodology for Cloud service evaluation. On one hand, CEEM summarizes key evaluation activities into ten generic steps. On the other hand, it provides a pre-experimental knowledge base and corresponding suggestions, which makes this methodology more practical in the Cloud computing domain. Overall, by offering systematic guidelines together with evaluation experiences, CEEM may help this study reduce human bias and facilitate implementations of Cloud service evaluation.

3. MediaWise Cloud

The MediaWise Cloud [1,2] is the project initiated by Commonwealth Scientific and Industrial Research Organisation (CSIRO), Australia, to develop a Cloud-based architecture for efficient content production, indexing, consumption and distribution. The MediaWise Cloud aims to circumvent the limitations of traditional CDNs such as lack of dynamic content production, indexing, personalization, recommendation and flexible pricing. It supports DIY CDN orchestration operations for simplifying media production, management and distribution over public and private Clouds such as Amazon EC2 [29] and CSIRO Cloud [30]. Compared to the traditional CDNs such as, Akamai [31], limelight [32], Ooyala [33] and MetaCDN [34], the MediaWise Cloud eliminates the need to own and manage expensive infrastructure while facilitating content owner requirements (e.g., News and media companies), SLA, security and privacy and QoS. As it is based on the public Cloud infrastructure, it supports pay-as-you-go models, scalability and elasticity making it suitable for small, medium and large enterprises.

To facilitate efficient media content production, management and distribution, the MediaWise Cloud supports a number of orchestration operations. These include:

1. *Content production and deployment:* is performed by the content owners. Content can be produced dynamically by a number of users using the Social Content Authoring and Sharing Platform (SCASP) [35]. The SCASP is a web application as shown in Fig. 1. It supports content creation in the form of storytelling and supports collaborative multimedia content creation and authoring by supporting operations such as video/audio split and merge. Once the media is created, it can be uploaded to the Cloud for distribution.
2. *Content indexing:* is performed by the content owners and the audience/users by attaching the appropriate metadata in the

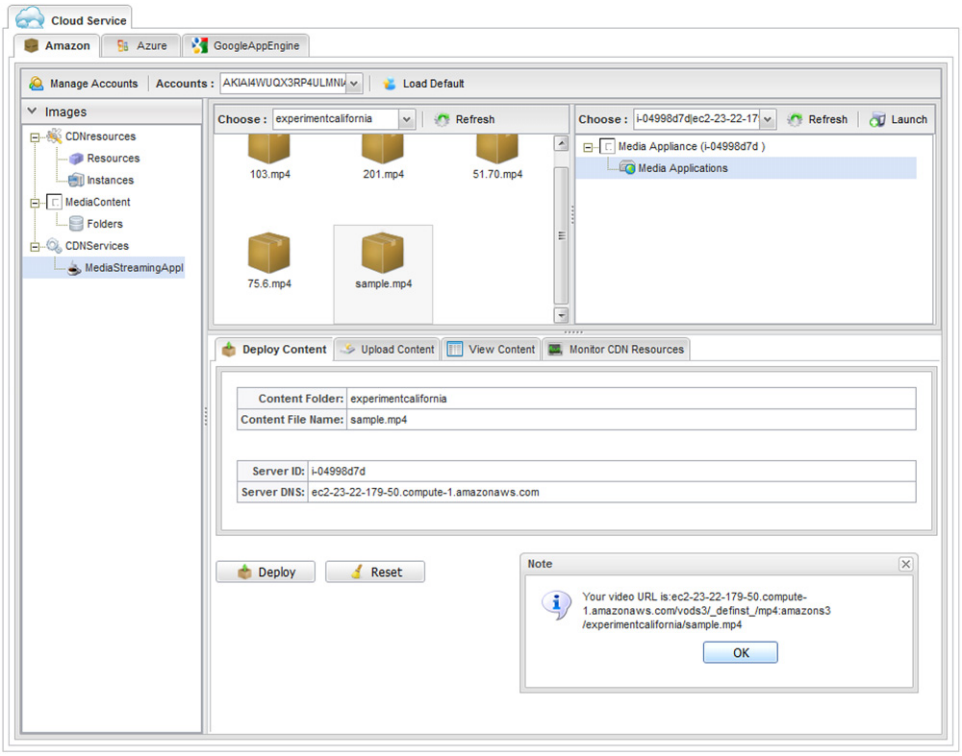

Fig. 2. Screenshot of the MediaWise Cloud Content Orchestrator (MCCO).

form of keywords, textual/visual annotations and comments. The content indexing enables fast content search and recommendation.

3. *Content consumption:* by users is done via the web-based interface as part of the SCASP. Using the web-based interface, the users can search and view the content (see Fig. 1).
4. *Content distribution:* is done efficiently by placing the relevant content and different geographical locations around the globe. The request for the content is then mapped dynamically to the location that is closest to the user, thereby supporting users' QoS requirements.

These operations are facilitated using the MediaWise Cloud Content Orchestrator (MCCO) [1,2] as shown in Fig. 2. MCCO was developed using Java programming language and is built using several publically available and CSIRO's in-house APIs. It incorporates an easy-to-use widgets based interface that hides the underlying complexity to support aforementioned orchestration life-cycle operations. The MCCO operations span across entire Cloud layers namely, Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). For instance, the MCCO application itself sits on the SaaS layer where it manages media production, management and distribution applications or appliances such as indexing server, content management server and the content streaming server at the PaaS layer. At the IaaS layer, it manages operations such as starting and stopping virtual machines (VMs) and content storage.

To orchestrate the operations using MCCO (see Fig. 3), first the content providers create the content and host them on the Cloud, for example on Amazon S3. Second, using the content indexer appliance(s) at the PaaS layer, the content is annotated and the metadata is created and linked to the content. Once the content is indexed, it can be consumed by the end users using the content appliance using which they can view the content, comment on the content as well as manipulate the content. To support the view operation, the content is streamed using the streaming appliance. Based on the number of requests and the load on appliances, the resources at the IaaS layer can be scaled up or down, for example, by starting new virtual machines (VMs) or stopping the underutilized VMs. In the following sections, we will evaluate MediaWise Cloud using the CEEM methodology [28] in the light of aforementioned orchestration operations.

4. Cloud Evaluation Experiment Methodology (CEEM)

We followed the ten-step Cloud Evaluation Experiment Methodology (CEEM) [28] to perform an initial performance evaluation of the related Cloud services, as illustrated in Table 1. For the conciseness and convenience of writing, here we briefly introduce our evaluation activities using six parts instead of elaborating the individual steps one by one. Section 4.1 clarifies the evaluation requirement and the relevant service features. Section 4.2 mainly identifies suitable metrics and benchmarks for this study. Section 4.3 lists and selects useful experimental factors. Section 4.4 particularly specifies the experimental design. Section 4.5 reports the experimental results together with analyses. Conclusions of this study are combined with Section 5.

4.1. Requirement recognition and service feature identification

The recognition of an evaluation requirement is not only to understand a problem related to Cloud service evaluation, but also to achieve a clear statement of the evaluation purpose, which is an obvious while a nontrivial task [36]. To help recognize a requirement, CEEM suggests preparing a set of specific and easily answerable questions, so that evaluators can conveniently define clear evaluation objectives, and then employ the strategy of sequential experiments to satisfy the overall evaluation requirement. As for the service feature identification, CEEM suggests exploring and locating the relevant features in a pre-established service feature list [23].

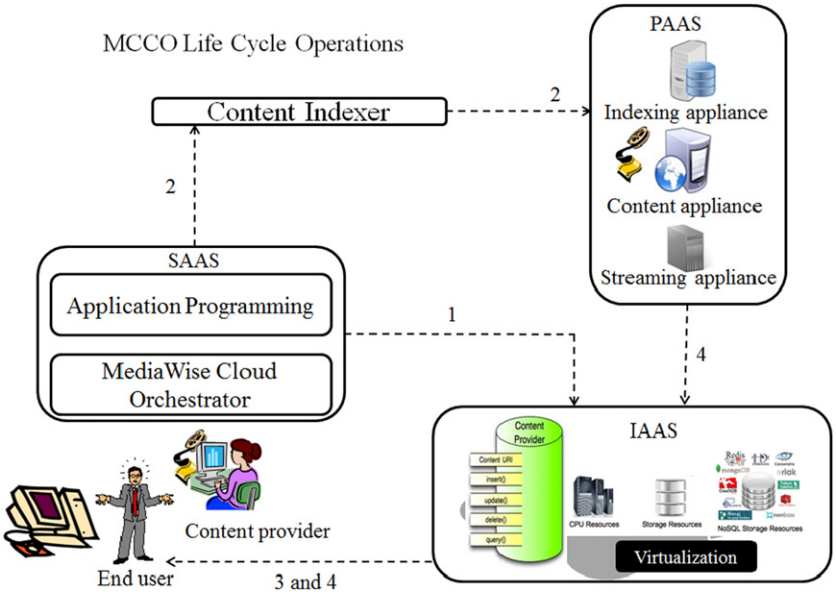

Fig. 3. Orchestration operations supported by the MediaWise Cloud Content Orchestrator (MCCO).

Table 1
Cloud evaluation experiment methodology [28].

ID	Evaluation activity	Section
1 2	*Requirement recognition:* Recognize the problem, and state the purpose of a proposed evaluation. *Service feature identification:* Identify Cloud services and their features to be evaluated.	4.1
3 4	*Metrics and benchmarks listing:* List all the metrics and benchmarks that may be used for the proposed evaluation. *Metrics and benchmarks selection:* Select suitable metrics and benchmarks for the proposed evaluation.	4.2
5 6	*Experimental factors listing:* List all the factors that may be involved in the evaluation experiments. *Experimental factors selection:* Select limited factors to study, and also choose levels/ranges of these factors.	4.3
7	*Experimental design:* Design experiments based on the above work. Pilot experiments may also be done in advance to facilitate the experimental design.	4.4
8 9	*Experimental implementation:* Prepare experimental environment and perform the designed experiments. *Experimental analysis:* Statistically analyse and interpret the experimental results.	4.5
10	*Conclusion and reporting:* Draw conclusions and report the overall evaluation procedure and results.	5

In this case, to help adjust the configuration of MediaWise Cloud, the main objective of this initial evaluation is to understand the variability and scalability of the related Cloud services. In general, variability describes the state of spread of a set of data. In this paper, we use variability to indicate the extent of fluctuation in values of an individual service performance index. Considering Cloud service variability could be related to time and location [37], we define two requirement questions around variability:

- How variable are the related Cloud services when running MediaWise Cloud for a period of time?
- How variable are the related Cloud services when running MediaWise Cloud with respect to different locations?

As for the scalability, we have identified two perspectives for the definitions. First, from the perspective of changing resource (with a certain amount of workload), scalability refers to the ability of a system to enlarge itself to accommodate increased workload. Moreover, two different directions can be further distinguished: one is *horizontal scalability*, and the other is *vertical scalability*. *Horizontal scalability* means the ability of employing more resources, while *vertical scalability* stands for the ability of increasing the power of resources [38]. Second, from the perspective of changing workload (with unchanged resource), scalability refers to the ability of a system to deal with the gradually increasing amount of work in a graceful manner. Imagine that the naming convention of *horizontal* and *vertical scalability* is in coordinates (see Fig. 4), we name the scalability from the second perspective of changing workload as *original scalability*.

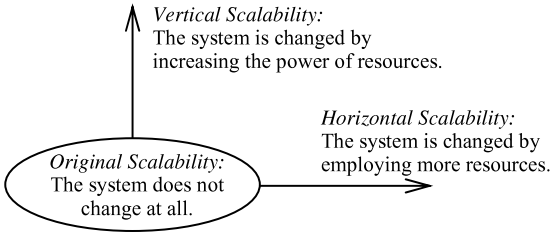

Fig. 4. Three different types of scalability.

Given the three specific scalability definitions, we can define three requirement questions, as listed below.

- How originally scalable are the related Cloud services when varying MediaWise Cloud workload?
- How horizontally scalable are the related Cloud services when varying the amount of Cloud resources?
- How vertically scalable are the related Cloud services when varying the power of Cloud resources?

Moreover, it has been identified that Cloud service variability and scalability have to be reflected by the change of value of other performance features [37]. Corresponding to the aforementioned MediaWise architecture, we may evaluate the variability and scalability of Cloud services through observing the performance change around three physical properties, namely Communication (e.g., Ethernet), Computation (e.g., VM Instance), and Storage

Table 2
Metrics and benchmarks for the IaaS evaluation.

Service feature		Metric	Benchmark
Physical property	Capability property		
Communication	Data throughput	Upload/Download Byte rate	Self-Monitor
Communication	Scalability	Saturation point	Self-Monitor
Computation	Load	CPU utilization	Self-Monitor
Computation	Scalability	Saturation point	Self-Monitor
Storage	Data throughput	Disk I/O Bit rate	Unknown
Storage	Scalability	Saturation point	Unknown

(e.g., BLOB). Thus, we can further define a set of evaluation requirement questions for observing those physical properties, as shown below.

- How MediaWise Cloud does behave in upload and download of media files with respect to particular Cloud service configurations (communication and/or storage)?
- How MediaWise Cloud deals with streaming of media files, with respect to particular Cloud service configurations (computation)?
- How MediaWise Cloud deals with metadata of media files, with respect to particular Cloud service configurations (computation and/or storage)?

4.2. Metrics and benchmarks listing and selection

According to the rich research in the evaluation of traditional computer systems, the selection of metrics plays an essential role in evaluation implementations [39]. Although traditional evaluation lessons treat metrics selection as one of the prerequisites of benchmark selection [40], we found that there were always trade-offs between metrics and benchmarks selection when evaluating Cloud services. For example, only two metrics (Benchmark Runtime and Benchmark FLOP Rate) are available to respectively measure computation latency and transaction speed if adopting NAS Parallel Benchmarks to evaluate Cloud services [41]. Therefore, we believe that metrics and benchmarks could be determined together within one step.

As suggested in [28], available metrics and benchmarks for Cloud service evaluation can be conveniently explored in the existing metric catalogue [42]. Since the choice of appropriate metrics depends on the features to be evaluated, evaluators are suggested in using particular Cloud service features as the retrieval key to quickly locate candidate evaluation metrics in the catalogue. In particular, an evaluated service feature is usually represented by a combination of a physical service property and its capability, for example, Communication Latency, or Storage Reliability. Therefore, we further split the Service Feature dimension into two parts: Physical Property and Capability Property.

Recall that the MediaWise Cloud architecture covers different service layers; therefore, we decided to measure different performance indexes at different service layers, respectively. The selected metrics and benchmarks for IaaS evaluation are listed in Table 2.

Here we particularly clarify the capability property: Data throughput. Traditionally, throughput is the term used mainly for measuring data communication speed. Here we consider throughput as a general concept that describes an amount of data processed in a particular period of time (from input to output) by any physical property of Cloud services. Moreover, we also replaced another widely-used term Bandwidth with throughput, though they are subtly different concepts. In fact, the ideal bandwidth of a physical property can be viewed as one of the inherent characteristics of a Cloud service [43,44], which has to be reflected though evaluating the actual throughput. In other words, the actual throughput is equal to the effective bandwidth of a particular physical property in real evaluation experiments. In particular, as described in [42],

Table 3
Metrics and benchmarks for the PaaS evaluation.

Service feature		Metric	Benchmark
Platform property	Capability property		
Indexing service	Latency	Response time	JMeter
Indexing service	Scalability	Saturation point	JMeter
Streaming service	Latency	Response time	SSPT tool
Streaming service	Data throughput	Bit rate	SSPT tool
Streaming service	Scalability	Speedup	SSPT tool

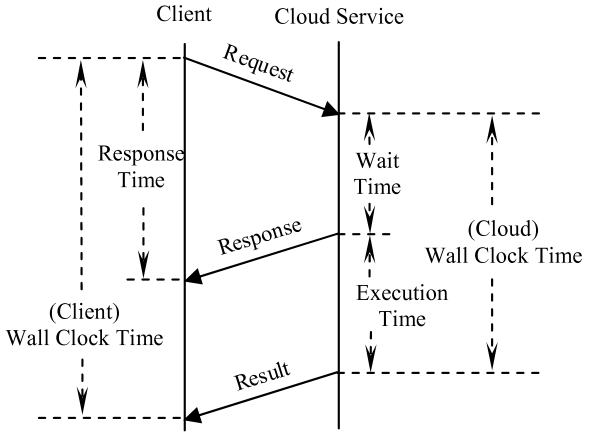

Fig. 5. Different time windows for defining latency.

CPU load is usually used together with other performance evaluation metrics to identify saturation points. Saturation point is a point at which some capacity is at its fullest limit. Here we use saturation point as a metric to reflect the original scalability of different service capacities. In the PaaS level, we mainly focus on the Database service and the streaming service in the aforementioned architecture. The selected metrics and benchmarks for PaaS evaluation are then listed in Table 3.

Here we particularly explain the capability property latency. Latency is mainly related to the measure of time delay for a particular job. In the literature, we found various definitions of latency depending on different contexts or perspectives. Here we give latency the broadest meaning to describe all the time-related capacities of a commercial Cloud service. As for the detailed contexts and perspectives, we can make further analysis by distinguishing different time windows, as demonstrated in Fig. 5. On the one hand, from the perspective of client, latency can refer to the *response time* (the elapsed time before the first reacting to the request) from a Cloud service while the service may be still working, and it can also refer to the total time taken for achieving the result of a job (i.e. the *(client) wall clock time*). Note that the response time will be equal to the *(client) wall clock time* if the response from service is given only after the job is finished. In such a case, we suggest using only the *(client) wall clock time* to avoid the possible confusion. On the other hand, from the perspective of Cloud service, latency may refer to the delayed time before really executing a job (e.g. the average *wait time* in queue [38]), and it may also refer to the required time of completing the job (e.g. the *execution time* for Computation [45] or the transfer time for Communication [46]). Meanwhile, the sum of *wait time* and *execution time* is then the *(Cloud) wall clock time* from the Cloud service's perspective [47].

As for benchmarks, JMeter can be used to simulate different numbers of concurrent user requests to read from the indexing service. By testing the *response time* under various request loads, we can identify the request *saturation point* to reflect the original scalability of the service.

JMeter provides yet another throughput definition. Throughput is calculated as requests/unit of time [48]. The time is calculated from the start of the first sample to the end of the last

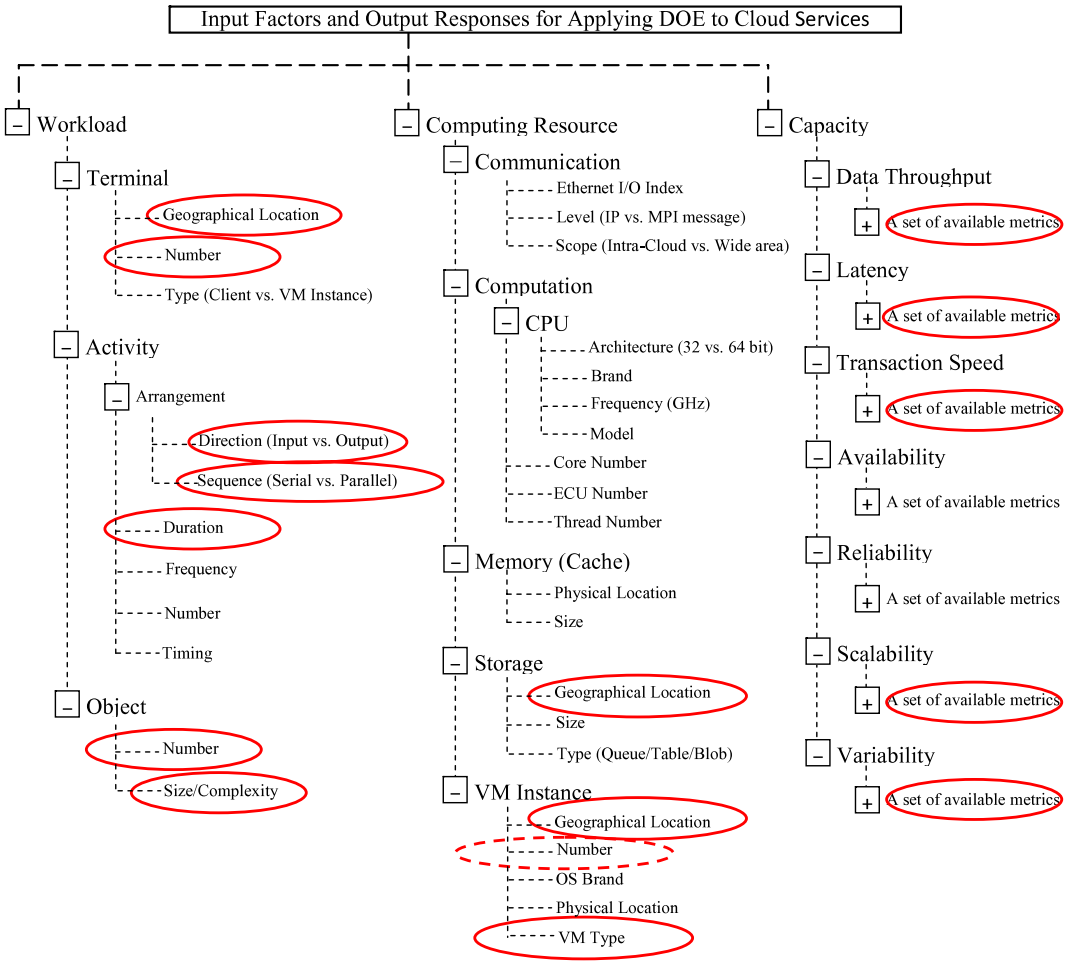

Fig. 6. Experimental factor listing and selection.

sample. This includes any intervals between samples, as it is supposed to represent the load on the server. The formula is: throughput = (number of requests)/(total time). The throughput is calculated from the point of view of the sampler target (e.g. the remote server in the case of HTTP samples). JMeter takes into account the total time over which the requests have been generated. That means if other samplers and timers are in the same thread, these will increase the total time, and therefore reduce the throughput value. So we avoided the situation where two identical samplers with different names which will have half the throughput of two samplers with the same name. As for latency, JMeter's measurement is well aligned with our definition. JMeter measures the latency from just before sending the request to just after the first response has been received. Thus, the time includes all the processing needed to assemble the request as well as assembling the first part of the response, which in general will be longer than one byte. Protocol analysers (such as Wireshark) measure the time when bytes are actually sent/received over the interface. The JMeter time should be closer to that which is experienced by a browser or other application client. As for the streaming service, we employ the Smooth Streaming Performance Testing (SSPT) tool. First, we may pre-configure and vary the VM instance types and the streaming workloads. Then, by measuring different streaming *response time* and *bit rate* with different configurations, we can calculate *speedup* to reflect streaming service's different scalabilities.

4.3. Experimental factor listing and selection

Before evaluating a Cloud service feature, knowing all factors (also called parameters or variables) that affect the service feature is a tedious but necessary task [49]. Although listing a complete scope of experimental factors may not be easily achieved, at all times evaluators should keep the factor list as comprehensive as possible, for further analysis and decision making about the factor selection and data collection [40]. When it comes to a particular evaluation experiment, however, it is better to start with limited design factors distinguished from nuisance ones and those that are not of interest. In other words, the factors that are expected to have high impact should be preferably selected [40]. As mentioned in [28], we may refer to the existing evaluation experiences to quickly lookup and identify design factors, as shown in Fig. 6.

For variability evaluation.

- *Duration*: for each evaluation experiment, we decided to take a ten-minute observation after warming up. In other words, we assign the value of *Duration* as ten minutes.
- *(Service) Geographical Location*: to cover the typical geographical locations for servers, we choose VM instances located in US, European and Asian data centres.
- *(Client) Geographical Location*: local machine vs. Amazon EC2 instance.

For scalability evaluation.

- Request Frequency.
- *VM instance type*: to investigate the runtime performance of MediaWise Cloud from a perspective of baseline, we select the Micro, Medium, and Large instance types, as listed in Table 4.
- VM Instance Number.
- *File size*: we choose file size ranging from 1 to 200 MB approximately.

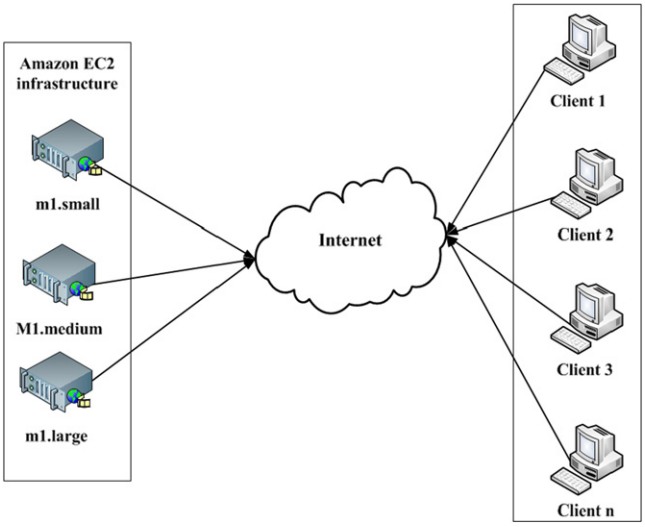

Fig. 7. Test bed to study content distribution capabilities of the MediaWise system.

4.4. Experimental design using conceptual model-based blueprint

Based on our taxonomy, we have built a conceptual model that further rationalizes and emphasizes the detailed relationships among those evaluation elements and classifiers. In essence, the elements/classifiers and their relationships can abstractly define and characterize the actual evaluation work, and therefore enable relatively fair and rational comparison between different performance evaluations according to their abstract characteristics. In practice, the taxonomy-based descriptions are used to mainly help clarify the evaluation requirements, while the conceptual model-based blueprint is used to represent the experimental design clearly and simply. In this case, we can draw the conceptual model-based blueprint for evaluating MediaWise Cloud, as shown in Fig. 8.

Table 4
EC2 VM instance types involved in this evaluation.

VM type	ECU	Memory (GB)	Storage (GB)	Network	Price (windows)
m1.small	1	1.7	1 × 160	Low	$0.091/h
m1.medium	2	3.75	1 × 410	Moderate	$0.182/h
m1.large	4	7.5	2 × 420	Moderate	$0.364/h

The experimental blueprint is composed of four parts, namely Experimental Operation, Workload, Cloud Resource, and Capability. In the Experimental Operation part, Repeat and Set Geographical Location naturally belong to the time- and location-variation evaluation scenarios, and they are therefore variability-related operations; Set Amount, Set Size, and Set Type exist in the workload- and resource-variation evaluation scenarios, and thus they are scalability-related operations, while as explained in the previous subsection, Set Direction and Set Sequence are operations that simulate potentially different customer-request scenarios. In the Workload part, we may use one of three different entities (Client/User, Request, and Media File) to describe performance evaluation workload. Note that, the Request here is client's activity, which is different from the aforementioned experimental operations. In the Cloud Resource part, we are concerned with three types of resources, namely Ethernet, VM instance, and Blob Storage. We consider Ethernet as a special Cloud resource, because Cloud services are employed inevitably through Internet/Ethernet. In fact, the Ethernet I/O Index is usually pre-supplied as a service-level agreement (SLA) by service providers. As for the Capability

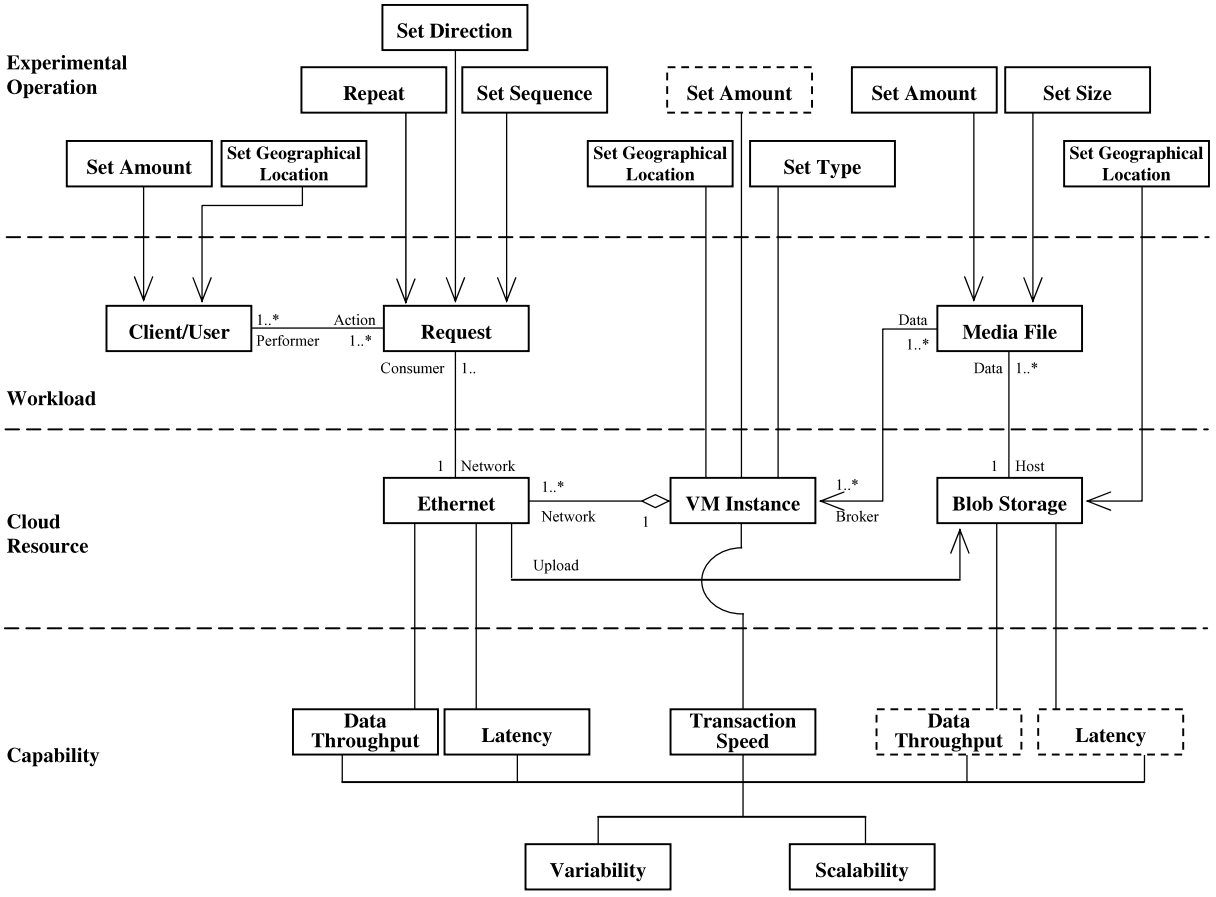

Fig. 8. The conceptual model-based blueprint for evaluating MediaWise.

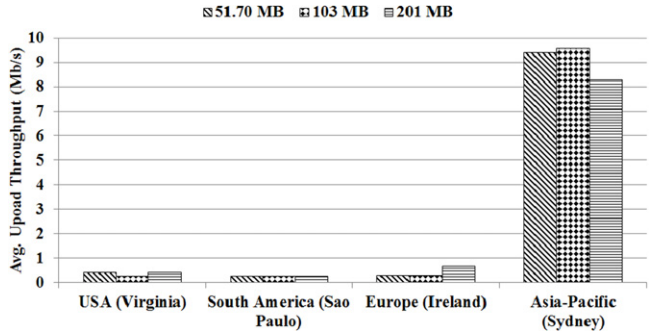

Fig. 9. Content upload throughput computed by uploading files from Canberra, Australia, to multiple geographical locations around the world.

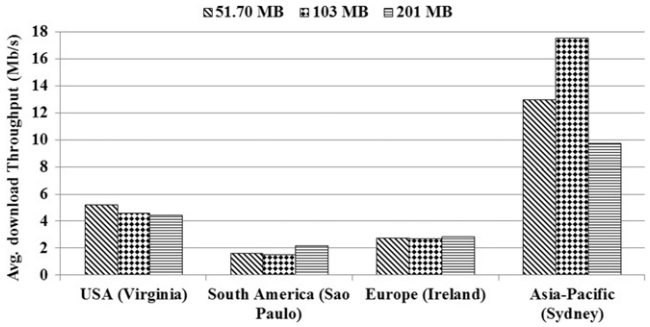

Fig. 10. Content download throughput computed by downloading files from Canberra, Australia, to multiple geographical locations around the world.

part, it is clear that the capabilities of a Cloud computing resource are intangible until they are measured. Meanwhile, the measurement has to be realized by using measurable and quantitative metrics. Therefore, we can treat the values of relevant metrics as tangible representations of the evaluated capacities.

4.5. Experimental results and analysis

Based on the aforementioned experimental design, we performed a number of experiments to validate the MediaWise Cloud. In particular, we performed analysis of content indexing, content consumption and content distribution capability of the MediaWise Cloud, as specified in the following subsections respectively.

4.5.1. MediaWise content production and deployment

In this section, we analyse MediaWise Cloud's deployment performance. We expect a large number of users (consumers) using the MediaWise Cloud to deploy and host their content from around the globe. The hosted content can be accessed by the intended consumers anywhere, anytime at anyplace. As the content producers and the content consumers can be located anywhere in the world, we studied the amount of time and speed associated with content hosting on the Cloud. In particular, upload throughput was computed by uploading the files of various sizes (50.70, 103 and 201 MB) to various geographically distributed Amazon data centres. These include: USA (Virginia), South America (Sao Paulo), Europe (Ireland) and Asia-Pacific (Sydney). We also analysed the amount of time taken by the content consumers to download the files from different data centres and computed the average download throughput. For these experiments, we assumed that the content consumers and the content producers are located at Canberra, Australia.

A client device (in Canberra) running Windows Vista on a laptop, and connected to IEEE 802.11n WLAN downloaded and uploaded three files of sizes 50.70, 103 and 201 MB sequentially, from Amazon's USA, South America, Europe and Asia-Pacific data centres. Fig. 9 shows the average upload throughput in Mb/s for four aforementioned data centres. From our results, we concluded that the distance between client location and data centres can have a significant impact on the average upload throughput. For example, the average upload throughput for uploading files to Sydney data centre was approximately 9 Mb/s whereas for USA, South America and Europe, it was 0.3553 Mb/s, 0.2587 Mb/s, 0.4127 Mb/s, respectively. These results strongly suggest that content producers should deploy their files to the closest data centre. In this case, it was Sydney.

Similarly, to study the average download throughput, the same client downloaded the files of sizes 50.70, 103 and 201 MB from the aforementioned data centres. Fig. 10 shows the results pertaining to average download throughput. As can be noticed, the average download throughput is higher if the files are downloaded from Sydney data centre (closest to the user) followed by USA, Europe and South America. From these results we again conclude that content should be consumed from the data centre closest to the user as the distance between the user and the data centre plays a vital part in hampering the download throughput.

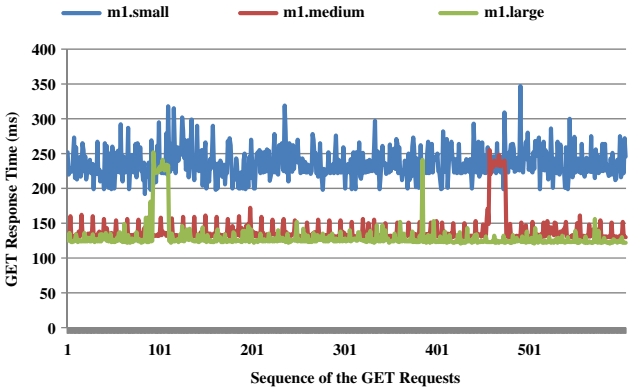

Fig. 11. The sequential GET response time during 10 min.

4.5.2. MediaWise indexing service evaluation

Since GET is one of the most widely used operations (considering web services) before consuming the media content through the MediaWise Cloud, we particularly evaluated the indexing service in regard to the GET performance using JMeter. Before the evaluation, we pre-stored 1000 records in the MySql database hosted in the selected types of VM instances respectively for the MediaWise Indexing Service. By issuing GETs once per second, we performed 10-min sequential requests from each of the VM instances. Several typical indices of the experimental result are shown in Table 5, which can be used to initially answer the evaluation questions about time-related variability and vertical scalability of indexing service. Following the suggestions in Step 9 (see Table 1) of the evaluation methodology, we further visualized the experimental results to better answer the questions and also facilitate experimental analysis, as shown in Fig. 11. Using experimental studies we found that the GET response time from m1.small instance is roughly between 200 and 300 ms, while the GET response time from m1.medium and m1.large is relatively stable and mostly below 150 ms.

To be more specific, we used Boxplot (cf. Fig. 12) to scale different quartiles of the GET response time. Note that the crosses in Fig. 11 indicate the outlier observations falling out of the 1.5 inter-quartile range (IQR). It is clearer that the indexing service using m1.small instance performs much worse than using the m1.medium and m1.large instances, with regard to not only the GET response time but also the variability of the response time. However, by using the metric *Performance/Price Ratio* (cf. Table 5), we can find that the indexing service supported by m1.small could be significantly more cost-effective than that by m1.medium and

Table 5
Response time of GET from the MediaWise Indexing Service.

VM type	Average (ms)	Minimum (ms)	Maximum (ms)	Standard deviation (ms)	Performance/price (ms/$)
m1.small	235.3	192	347	21.6	2585.7
m1.medium	137.8	129	255	18.6	757.1
m1.large	129.6	121	252	18.7	356

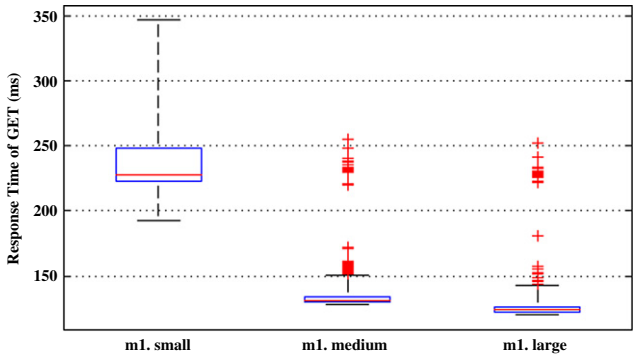

Fig. 12. The sequential GET response time during 10 min shown in boxplot.

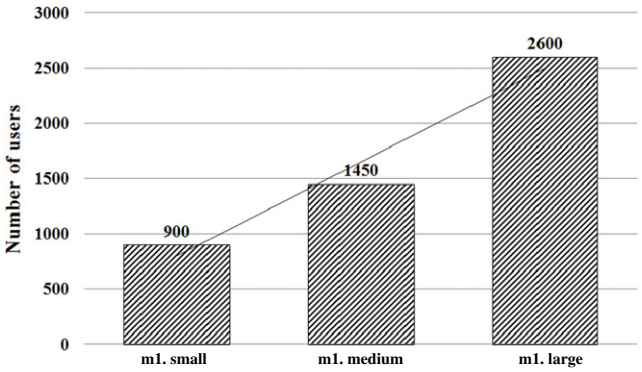

Fig. 13. Total number of users supported by Wowza server on different Amazon EC2 instances.

m1.large. Therefore, we decided to use the m1.small type to conduct horizontal scalability evaluation in the future, for the purpose of balance between potential performance and cost of the indexing service.

4.5.3. MediaWise streaming service evaluation

In Section 4.5.1, we analysed the performance related to content deployment and concluded that the content should place to and consumed from the data centre closest to the user. In this subsection, we study the performance of MediaWise Cloud in regard to content distribution. Content distribution involves distributing the deployed content (by content producers) to content consumers or users. In regard to content consumption tests, we tested our hypothesis that through vertical scalability, more users will be supported by the MediaWise system. We assumed that these users will request high definition (HD) videos from the MediaWise Cloud. In order to test this hypothesis, we deployed our MediaWise Cloud comprising of three streaming appliances running on three different Amazon EC2 instances with different capabilities, as mentioned in Table 4.

The MediaWise streaming appliances comprise of Wowza Media Servers [50] which were performance-tuned based on the specifications provided by Wowza documentation. The experimental test bed is shown in Fig. 7. The Amazon EC2 instances were running in the Virginia (USA) data centre and a large number of clients were situated in Canberra, Australia. We used the Wowza Load Test tool to simulate a large number of clients. Fig. 13 shows the Wowza server saturation points of MediaWise system. This figure shows that a small Amazon EC2 instance (m1.small) supports 900 clients whereas the m1.medium type instance supported 1450 users and the m1.large instance supported 2600 users, respectively. It is clear from Fig. 13 that the aforementioned hypothesis is verified as true. Note that the MediaWise Cloud starts refusing users' requests above those saturation points. One of the major factors influencing the saturation point was CPU utilization. Fig. 14 shows the rate of change of CPU utilization which is directly affected by the number of clients trying to connect the Wowza streaming server. It can be seen from this figure that CPU utilization increases rapidly when more clients connect to the Wowza streaming server. For small instance (m1.small) the CPU utilization reached 100% at 900 users. For medium instance (m1.medium) the CPU utilization reached 100% at 1450 users and for the large instance (m1.large) CPU utilization reached 100% with 2600 users.

5. Conclusions and future work

In this paper, we presented an experimental methodology for evaluating the runtime performance management of MediaWise system. We clearly articulated and quantified the QoS parameters related to different components of the MediaWise system. We also identified the important benchmarks which are required for conducting performance evaluation experiments of classes of content-driven applications. The proposed methodology is comprehensive

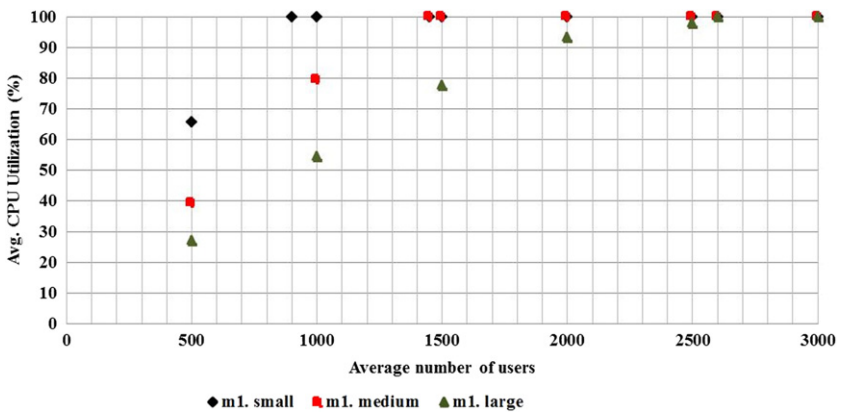

Fig. 14. Rate of change of CPU utilization w.r.t. increase in the number of users.

as it can evaluate the impact of wide-area network as well as Cloud resource configurations on the performance of MediaWise system. Through the detailed experiments in this performance evaluation study, we confirm that geographical locations of CDN servers would have huge impact on the data transfer performance. For example, Australian users would have much better customer experiences if the consumed contents are located in Sydney-region data centre. Furthermore, we found that the tradeoff between performance and cost of Cloud-based CDN systems could be a tricky issue. For example, in this case, although the m1.small type of EC2 instance performs unsurprisingly the worst, it has significant cost advantage for supporting the MediaWise system.

Therefore, in future work, we will unfold the horizontal scalability evaluation to reveal that to what extent the m1.small type could keep its cost advantage for MediaWise system. Moreover, we will apply the proposed methodology to conduct experiments for a large variety of applications that can be built using the MediaWise system. These applications include real-time video surveillance for aged-care monitoring, education, and video analytics. We believe that the methodology can also be applied to other emerging big data applications in the domain of disaster management, high energy physics, genomics, automobile simulations, medical imaging, remote earth sensing, and the like.

References

[1] D. Georgakopoulos, R. Ranjan, K. Mitra, X. Zhou, MediaWise—designing a smart media cloud, in: Proceedings of the International Conference on Advances in Cloud Computing, ACC 2012, Bangalore, India, July 26–28, 2012.

[2] R. Ranjan, K. Mitra, D. Georgakopoulos, MediaWise cloud content orchestrator, J. Internet Serv. Appl. 4 (2) (2013). Springer.

[3] K. Alhamazani, Rajiv Ranjan, F. Rabhi, L. Wang, K. Mitra, Cloud monitoring for optimizing the QoS of hosted applications, in: Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science, IEEE Computer Society, 2012, pp. 765–770.

[4] H. Blodget, Amazon's cloud crash disaster permanently destroyed many customers' data, [Online] http://www.businessinsider.com.au/amazon-lost-data-2011-4 (access date: 12-07-13).

[5] C. Hung, A. Wang, J. Li, K.W. Ross, Measuring and evaluating large-scale CDNs, in: Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement, (IMC 2008), ACM Press, 2008, pp. 15–29.

[6] K.L. Johnson, J.F. Carr, M.S. Day, M.F. Kaashoek, The measured performance of content distribution networks, Comput. Commun. 24 (2) (2000).

[7] G. Fortino, A. Garro, W. Russo, M. Vaccaro, Performance evaluation of content distribution network architectures through agent-based modeling and simulation, in: Proceedings of the 12th Workshop on Objects and Agents, WOA 2011, pp. 158–165.

[8] S. Pallickara, G. Fox, Enabling hierarchical dissemination of streams in content distribution networks, Concurr. Comput.: Pract. Exp. 24 (14) (2012) 1594–1606.

[9] C. Papagianni, A. Leivadeas, S. Papavassiliou, A cloud-oriented content delivery network paradigm: modeling and assessment, IEEE Trans. Depend. Secure Comput. 10 (5) (2013) 287–300.

[10] W. Sobel, S. Subramanyam, A. Sucharitakul, J. Nguyen, H. Wong, A. Klepchukov, S. Patil, A. Fox, D. Patterson, Cloudstone: multiplatform, multi-language benchmark and measurement tools for web 2.0, in: Proceedings of the 1st Workshop on Cloud Computing and its Applications, CCA 2008, October 2008, pp. 1–6.

[11] V. Stantchev, Performance evaluation of cloud computing offerings, in: Proceedings of the 3rd International Conference on Advanced Engineering Computing and Applications in Sciences, (ADVCOMP 2009), IEEE Computer Society, 2009, pp. 187–192.

[12] A. Lenk, M. Menzel, J. Lipsky, S. Tai, P. Offermann, What are you paying for? Performance benchmarking for infrastructure-as-a-service offerings, in: Proceedings of the 4th International Conference on Cloud Computing, (IEEE CLOUD 2011), IEEE Computer Society, 2011, pp. 484–491.

[13] J. Kolodziej, F. Xhafa, Modern approaches to modeling user requirements on resource and task allocation in hierarchical computational grids, Appl. Math. Comput. Sci. 21 (2) (2011) 243–257.

[14] L. Wang, D. Chen, J. Zhao, J. Tao, Resource management of distributed virtual machines, Int. J. Ad Hoc Ubiquitous Comput. 10 (2) (2012) 96–111.

[15] Y. Mhedheb, F. Jrad, J. Tao, J. Zhao, J. Kolodziej, A. Streit, Load and thermal-aware VM scheduling on the cloud, in: J. Kolodziej, et al. (Eds.), Proc. of the 13th ICA3PP 2013, in: LNCS, vol. 8285, 2013, pp. 101–114.

[16] L. Wang, D. Chen, Y. Hu, Y. Ma, J. Wang, Towards enabling cyberinfrastructure as a service in clouds, Comput. Electr. Eng. 39 (1) (2013) 3–14.

[17] S.M. Blackburn, K.S. McKinley, R. Garner, C. Hoffmann, A.M. Khan, R. Bentzur, A. Diwan, D. Feinberg, D. Frampton, S.Z. Guyer, M. Hirzel, A. Hosking, M. Jump, H. Lee, J.E.B. Moss, A. Phansalkar, D. Stefanović, T. VanDrunen, D. von Dincklage, B. Wiedermann, Wake up and smell the coffee: evaluation methodology for the 21st century, Commun. ACM 51 (8) (2008) 83–89.

[18] A. Li, X. Yang, S. Kandula, M. Zhang, CloudCmp: comparing public cloud providers, in: Proceedings of the 10th Annual Conference on Internet Measurement, (IMC 2010), ACM Press, 2010, pp. 1–14.

[19] J. Schad, J. Dittrich, J.-A. Quiané-Ruiz, Runtime measurements in the cloud: observing, analyzing, and reducing variance, VLDB Endow. 3 (1–2) (2010) 460–471.

[20] G. Wang, T.S.E. Ng, The impact of virtualization on network performance of Amazon EC2 data center, in: Proceedings of the 29th Annual IEEE International Conference on Computer Communications, (IEEE INFOCOM 2010), IEEE Computer Society, 2010, pp. 1–9.

[21] J. Kolodziej, S.U. Khan, Multi-level hierarchic genetic-based scheduling of independent jobs in dynamic heterogeneous grid environment, Inform. Sci. 214 (2012) 1–19.

[22] V. Stantchev, C. Schröpfer, Techniques for service level enforcement in web-services based systems, in: Proceedings of the 10th International Conference on Information Integration and Web-Based Applications & Services, (iiWAS 2008), ACM Press, 2008, pp. 7–14.

[23] Z. Li, H. Zhang, L. O'Brien, R. Cai, S. Flint, On evaluating commercial cloud services: a systematic review, J. Syst. Softw. 86 (9) (2013) 2371–2393.

[24] J. Dejun, G. Pierre, C.-H. Chi, EC2 performance analysis for resource provisioning of service-oriented applications, in: Proceedings of the 2009 International Conference on Service-Oriented Computing, (ICSOC/ServiceWave 2009), Springer-Verlag, 2009, pp. 197–207.

[25] K.R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H.J. Wasserman, N.J. Wright, Performance analysis of high performance computing applications on the Amazon web services cloud, in: Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science, (CloudCom 2010), IEEE Computer Society, 2010, pp. 159–168.

[26] M. Alhamad, T. Dillon, C. Wu, E. Chang, Response time for cloud computing providers, in: Proc. 12th Int. Conf. Information Integration and Web-Based Applications & Services, (iiWAS 2010), ACM Press, 2010, pp. 603–606.

[27] D. Kossmann, T. Kraska, S. Loesing, An evaluation of alternative architectures for transaction processing in the cloud, in: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, (SIGMOD 2010), ACM Press, 2010, pp. 579–590.

[28] Z. Li, L. O'Brien, H. Zhang, CEEM: a practical methodology for cloud services evaluation, in: Proceedings of the 3rd IEEE International Workshop on the Future of Software Engineering for/in Cloud (FOSEC 2013) in Conjunction with IEEE 9th World Congress on Services (IEEE SERVICES 2013), IEEE Computer Society, 2013, pp. 44–51.

[29] Amazon EC2, [Online] http://aws.amazon.com/ec2/ (access date: 12-07-13).

[30] CSIRO Cloud, [Online] http://www.csiro.au/en/Outcomes/ICT-and-Services/Data-deluge/cloud-computing.aspx (access date: 12-07-13).

[31] Akamai, [Online] http://www.akamai.com/ (access date: 12-07-13).

[32] Limelight Networks, [Online] http://www.limelight.com/ (access date: 12-07-13).

[33] Ooyala, [Online] http://www.ooyala.com/ (access date: 12-07-13).

[34] MetaCDN, [Online] http://www.metacdn.com/ (access date: 12-07-13).

[35] C. Wang, R. Ranjan, X. Zhou, K. Mitra, S. Saha, M. Meng, D. Georgakopoulos, L. Wang, P. Thew, A cloud-based collaborative video story authoring and sharing platform, CSI J. Comput. 1 (3) (2012) 66–76. Computer Society of India Press.

[36] D.C. Montgomery, Design and Analysis of Experiments, seventh ed., John Wiley & Sons, Inc., Hoboken, NJ, 2009.

[37] Z. Li, L. O'Brien, R. Cai, H. Zhang, Towards a taxonomy of performance evaluation of commercial cloud services, in: Proceedings of the 5th International Conference on Cloud Computing, (IEEE CLOUD 2012), IEEE Computer Society, 2012, pp. 344–351.

[38] N. Yigitbasi, A. Iosup, D. Epema, S. Ostermann, C-Meter: a framework for performance analysis of computing clouds, in: Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, (CCGRID 2009), IEEE Computer Society, 2009, pp. 472–477.

[39] M.S. Obaidat, N.A. Boudriga, Fundamentals of Performance Evaluation of Computer and Telecommjnication Systems, John Wiley & Sons, Inc., Hoboken, New Jersey, 2010.

[40] R.K. Jain, The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling, Wiley Computer Publishing, John Wiley & Sons, Inc., New York, NY, 1991.

[41] S. Akioka, Y. Muraoka, HPC benchmarks on Amazon EC2, in: Proceedings of the 24th International IEEE Conference on Advanced Information Networking and Applications Workshops, (WAINA 2010), IEEE Computer Society, 2010, pp. 1029–1034.

[42] Z. Li, L. O'Brien, H. Zhang, R. Cai, On a catalogue of metrics for evaluating commercial cloud services, in: Proceedings of the 13th ACM/IEEE International Conference on Grid Computing, (Grid 2012), IEEE Computer Society, 2012, pp. 164–173.

[43] L. Wang, C. Fu, Research advances in modern cyberinfrastructure, New Gener. Comput. 28 (2) (2010) 111–112.

[44] L. Wang, G. Laszewski, A. Younge, X. He, M. Kunze, J. Tao, C. Fu, Cloud computing: a perspective study, New Gener. Comput. 28 (2) (2010) 137–146.

[45] P. Bientinesi, R. Iakymchuk, J. Napper, HPC on competitive cloud resources, in: B. Furht, A. Escalante (Eds.), Handbook of Cloud Computing, Springer-Verlag, New York, 2010, pp. 493–516.

[46] Z. Hill, M. Humphrey, A quantitative analysis of high performance computing with Amazon's EC2 infrastructure: the death of the local cluster? in: Proceedings of the 10th IEEE/ACM Interntional Conference on Grid Computing, (Grid 2009), IEEE Computer Society, 2009, pp. 26–33.
[47] J. Li, M. Humphrey, D. Agarwal, K. Jackson, C. van Ingen, Y. Ryu, eScience in the cloud: a MODIS satellite data reprojection and reduction pipeline in the windows azure platform, in: Proceedings of the 23rd IEEE International Symposium on Parallel and Distributed Processing, (IPDPS 2010), IEEE Computer Society, 2010, pp. 1–10.
[48] Jmeter Throughput, [Online] http://jmeter.apache.org/usermanual/component_reference.html#Summary_Report (access date: 27-08-13).
[49] J.-Y. Le Boudec, Performance Evaluation of Computer and Communication Systems, EFPL Press, Lausanne, Switzerland, 2011.
[50] Wowza Media Server, [Online] http://www.wowza.com (access date: 27-08-13).

Zheng Li received his Degree of M.E. by Research from the University of New South Wales (UNSW). He is now a Ph.D. student at the School of Computer Science at the Australian National University (ANU), and a graduate researcher with the Software Systems Research Group (SSRG) at National ICT Australia (NICTA). He is the author of more than 20 journal and conference publications. His research interests include empirical software engineering, software cost/effort estimation, machine learning, Web service composition, and Cloud computing.

Karan Mitra is a Post-Doctoral Research Fellow at Luleå University of Technology, Sweden. He received his Dual-badge Ph.D. from Monash University, Australia, and Luleå University of Technology in 2013. He received his MIT (MT) and a PGradDipDigComm from Monash University in 2006 and 2008, respectively. He received his BIS (Hons.) from Guru Gobind Singh Indraprastha University, Delhi, India in 2004. His research interests include quality of experience modelling and prediction, context-aware computing, cloud computing and mobile and pervasive computing systems. From January 2012 to December 2013 he worked as a researcher at CSIRO, Canberra, Australia. He is a member of the IEEE and ACM.

Miranda Zhang is a Ph.D. student in the Computer System Group at Research School of Computer Science of The Australian National University. She holds a Bachelor of Software Engineering degree from The University of New South Wales, Sydney. She had done a half year internship with CSIRO in 2012 before starting her Ph.D. and currently receiving a Top-up Scholarship from CSIRO. She has published 3 peer-reviewed conference papers and 2 journal articles.

Rajiv Ranjan is a Senior Research Scientist (equivalent to Associate Professor in North American University System) and Julius Fellow in the CSIRO Computational Informatics, Canberra, where he is working on projects related to cloud and big data computing. He has been conducting leading research in the area of cloud and big data computing developing techniques for: (i) Quality of Service-based management and processing of multimedia and big data analytics applications across multiple cloud data centres (e.g., CSIRO Cloud, Amazon and GoGrid), and (ii) automated decision support for migrating applications to data centres. Rajiv has developed a strong publication record which includes 30 journal articles, 30 conference papers, 8 book chapters, and 7 books/edited proceedings. Rajiv has 1960+ lifetime citations and an *h*-index of 21 according to Google Scholar citations. Overall, about 70% of his journal papers and 60% of conference papers are A*/A ranked publication, according to the ERA. Rajiv also has 180+ ISI citations with an *h*-index of 7. His papers have appeared at selective, highly reputed venues including IEEE Transactions on Parallel and Distributed Systems (ERA A*, ISI IF 1.4), Journal of Computer and System Sciences (ERA A*, ISI IF 1.0), World Wide Web Conference (CORE/ERA A*), and IEEE Communications Surveys and Tutorials (#1 Computer Science Journal 2011, ISI IF 6.311).

Dimitrios Georgakopoulos Leads the Information Engineering Program at the CSIRO Computational Informatics. The program has well over a hundred research staff, visitors and Ph.D. students and specializes in the areas that include Service/Cloud Computing, Human Computer Interaction, Machine Learning, and Semantic Data Management. Dimitrios is also an Adjunct Professor at the Australian National University. Before coming to CSIRO in October 2008, Dimitrios held research and management positions in several industrial laboratories in the US. From 2000 to 2008, he was a Senior Scientist with Telcordia, where he helped found and led Telcordia's Research Centers in Austin, Texas, and Poznan, Poland. From 1997 to 2000, Dimitrios was a Technical Manager in the Information Technology organization of Microelectronics and Computer Corporation (MCC), and the Chief Architect of MCC's Collaboration Management Infrastructure (CMI) consortial project. From 1990 to 1997, Dimitrios was a Principal Scientist at GTE (currently Verizon) Laboratories Inc. Dimitrios has received a GTE (Verizon) Excellence Award, two IEEE Computer Society Outstanding Paper Awards, and was nominated for the Computerworld Smithsonian Award in Science. He has published more than one hundred twenty journal and conference papers.

Albert Y. Zomaya is currently the Chair Professor of High Performance Computing and Networking in the School of Information Technologies, The University of Sydney. He is the author/co-author of seven books, more than 400 papers, and the editor of nine books and 11 conference proceedings. He is the Editor in Chief of the IEEE Transactions on Computers and serves as an Associate Editor for 19 leading journals. Professor Zomaya was the recipient of the IEEE TCPP Outstanding Service Award and the IEEE TCSC Medal for Excellence in Scalable Computing, both in 2011.

Liam O'Brien has over 23 years' experience in research and development in software engineering. He is a Software and Applications Architect with Geoscience Australia and was previously Chief Software Architect with CSIRO and a Principal Researcher at NICTA's e-Government Initiative. He is also a Member-at-Large of the Service Science Society Australia which he co-founded in 2010. He has previously worked as a researcher with Lero (Ireland), Carnegie Mellon University's Software Engineering Institute (USA), CSIRO (Australia) and the University of Limerick (Ireland). His main areas of research include enterprise architecture, software architecture, SOA, service science, software reuse, software modernization, and cloud computing. He holds a B.Sc. and Ph.D. from the University of Limerick, Ireland. He is a member of the IEEE and IEEE Computer Society.

Shengtao Sun currently is a lecturer at College of Information Science and Engineering, Yanshan University P. R. China. Dr. Shengtao Sun got Ph.D. from Center of Earth Observation and Digital Earth (CEODE), Chinese Academy of Sciences (CAS) on July 2012. His research interests include expert system, intelligent information retrieval, semantic ontology, Grid/Cloud computing.