SaW: Video Analysis in Social Media with Web-based Mobile Grid Computing

Mikel Zorrilla, Julián Flórez, Alberto Lafuente, Angel Martin, Igor G. Olaizola, *Member, IEEE*, and Iñigo Tamayo

***Abstract*—The burgeoning capabilities of Web browsers to exploit full-featured devices can turn the huge pool of social connected users into a powerful network of processing assets. HTML5 and JavaScript stacks support the deployment of social client-side processing infrastructure, while WebGL and WebCL fill the gap to gain full GPU and multi-CPU performance. Mobile Grid and Mobile Cloud Computing solutions leverage smart devices to relieve the processing tasks to be performed by the service infrastructure. Motivated to gain cost-efficiency, a social network service provider can outsource the video analysis to elements of a mobile grid as an infrastructure to complement an elastic cloud service. As long as users access to videos, batch image analysis tasks are dispatched from the server, executed in the background of the client-side hardware, and finally, results are consolidated by the server. This paper proposes SaW (Social at Work) to provide a pure Web-based solution as a mobile grid to complement a cloud media service for image analysis on videos.**

***Index Terms*—Distributed computing, image analysis, multimedia databases, multimedia systems, social media, web-based architecture**

I. Introduction

THE social media paradigm has led to a significant rise in the volume of user generated content managed by social networks with millions of users accessing services, each of them often using multiple devices at the same time. Service providers aim to engage audience, eager for contents, by boosting the media relevance. To this end, a deeper automatic tagging enables better matching of user interests with the content database and reveals underlying connections between items, such as applying face detection mechanisms or content-based indexing to find related videos. Image analysis algorithms empower automatic retrieval of salience features but they also involve computing-intensive functions. Therefore, the processing requirements grow substantially when all the media items comprising the social network database are analysed. Here, on the one hand big data challenges arise when social services have continuously increasing databases, while on the other hand more and more processing resources are required to analyse all the content.

Grid and Cloud technologies provide High Performance Computing systems that aim to satisfy these requirements. However, as pointed in [1], other under-explored alternatives could enhance the trade-off between infrastructure cost, elapsed time and energy saving. It would depend on the number of available processing nodes, the inherent characteristics of the tasks to be performed in parallel and the data volume.

M. Zorrilla is with the Department of Digital Media and Broadcasting technologies, Vicomtech-IK4, San Sebastian, 20009, Spain. e-mail: (mzorrilla@vicomtech.org).

Manuscript received July 15, 2015; revised January 29, 2016.

To deal with the aforementioned context, this paper introduces a new concept of Social at Work: SaW. It aims to complement a Web-based social media service with all the idle devices, mostly mobiles, that usually have underexploited resources while accessing the service. SaW goes beyond the Infrastructure as a Service (IaaS) model, creating a system related to Mobile Grid Computing [2] concept with the available CPU and GPU resources of the different client devices to complement a virtualised cloud server, which provides the social media service.

Inspired by the Mobile Grid Computing and the Mobile Cloud Computing (MCC) [3] research fields over a social network mainly based on video content, SaW aims to bring together the huge pool of users permanently connected to media services in social networks and the ever increasing processing capabilities of most of their devices. As a consequence, service providers will embrace the community assets building a device centric grid to improve the social service by means of media analysis. Thus, Social at Work (SaW) concept enables service provider to recruit spare CPU/GPU cycles of client devices into an active gear of the social platform, saving cloud resources to the server when the connected clients can perform those tasks.

To achieve a SaW system, some remaining issues must be faced such as turning a Web client into a runtime application framework, shrinking Web engines performance gap between native applications and Web-apps, deploying communication layer to distribute background analysis and tracking processing request volume dealing with uncertainty of resources availability and elasticity.

First, the current device ecosystem is highly heterogeneous, with different operating systems and programming languages, resulting in complex software cross-platform development. Here, SaW proposes a pure Web-based approach since Web technologies overcome the interoperability barriers. HTML5 turns the Web into a real application platform middleware accessing hardware resources of the appliances through JavaScript [4].

Second, in order to exploit native GPU and multi-CPU potential of a device, WebGL and WebCL bindings to OpenGL and OpenCL run hardware-accelerated, parallel and cross-platform programs. So, they endow Web applications with parallel computing capabilities, accelerating Web applications for intensive image processing [5].

Then, Ajax (Asynchronous JavaScript and XML) [6] and

Websockets [7] are employed as a vehicle for establishing and maintaining mainstream communication between server and clients, transforming the classical synchronous request-response model into a full bidirectional one. This feature enables the server to send asynchronously updates to the client-side browser and to deliver background data.

Finally, the possibility to perform image processing tasks in parallel, such as feature extraction, segmentation, clustering and classification, eases to leap scalability. Due to the video stream nature, composed by individual frames, they can be easily split into independent tasks ready to be distributed. Beyond, the intrinsic presence of key frames in video coding, makes easier navigation and selection of representative images. Servers can dispatch the tasks to users' devices where they are run in the background. These background Web browser applications must balance the mechanism to leverage all available computing resources while provide the best possible user experience.

Thus, the validation of the approach can be explained in terms of the net benefit obtained in the server by delegating part of the tasks. The equation includes two relevant keys, which are based on certain parameters that are dependent on the technological state-of-the-art and the users social media consumption habits: (1) the amount of work that can be distributed to the client devices, which depends on the number of available clients, their capabilities, and the fraction of resources they can dedicate to background tasks without disturbing the user experience, and (2) the extra work created in the server to manage the task scheduling, which should be residual in comparison with the saved work or at least not exceed it.

A. Contributions

This paper proposes a SaW system for media analysis of the content collection in a social service. Drawing inspiration from volunteer computing initiatives for big data, this paper presents a pure Web-based distributed solution. SaW is deployed on top of the user appliances of a social media community, including the hardware-accelerated features for suitable devices. Thereby, the service provider gains the immense processing ability of its big social community to perform independent background hardware-accelerated image processing tasks. To achieve it, these queued tasks are embedded to the different social media services accessed by the users.

More specifically, the main contribution can be translated to two different topics. On the one hand, we propose a pure Web-based architectural design of the SaW concept enabling an interoperable solution. On the other hand, the article provides a proof-of-concept implementation of SaW using WebGL and WebCL technologies, in order to evaluate the SaW approach supported by experimental results and an analysis of the performance based on a previous model extending it by terms of GPU usage.

B. Paper structure

This paper starts with the related work in Section II, exploring the different Internet-based computing models and analysing their existing mechanisms for the interoperability, task distribution, support for parallel processing and different data structures. Section III presents the main contribution of the paper with the definition of the SaW concept. It depicts the contributions of SaW to the aforementioned related work, presents a suitable scenario for SaW on a social media service, defines the design objectives of the SaW architecture, presents a pure Web-based architectural design, and analyses the considerations of the architecture regarding the defined design objectives.

It follows with the evaluation of the SaW approach in Section IV. Subsection IV-A describes the experimental results in terms of scalability over a proof-of-concept implementation of SaW using WebGL and WebCL, subsection IV-B presents a performance analysis, extending the already published model in [8] in terms of GPU, and in subsection IV-C some remarks regarding the validation of the SaW hypothesis are presented. Finally, Section V presents the conclusions.

II. Related work

This section presents the related work, providing a definition of the Internet-based computing models and focusing on the different topics addressed by distributed computing: the interoperability, the task distribution managing, the parallel processing capabilities and the different data structures.

A. Computing Models

This section describes the main involved concepts in terms of Internet-based computing, where shared resources, data and information are provided to computers to reach a common goal.

1) Grid Computing: Grid Computing [9] has been an important paradigm in distributed systems for the last two decades. Basically, a grid is a network system where computing tasks are distributed to use non-dedicated computing resources, which may include servers or client computers. The high potential of the nowadays abundant and frequently idle client hardware boosts the opportunistic and delay-tolerant [10] use of client resources in the grid. In this volunteer computing SETI@home is the most popular example. SETI@home [11] approach has been the pioneer of big data grid infrastructures taking benefit of Internet-connected computers of volunteers. SETI@home has spread the collaborative network model to other unselfish research in areas such as astronomy, climate, astrophysics, mathematics, genetics, molecular biology and cryptography where volunteers and donors share the computing time from personal devices.

2) Mobile Grid Computing: Grid Computing is characterised by the heterogeneity of the resources in both amount and nature, by the sporadic availability, churn and unreliability of the devices, and by their anonymity and lack of trust. These issues are more relevant in Mobile Grid Computing (MGC) [2], where computing resources include mobile devices with wireless communications, and therefore prone to disconnections and other eventualities.

3) Cloud Computing: More recently, Cloud Computing [12], a new paradigm of distributed computing where virtualised computing resources are provided on-demand, has experienced a dramatic growth. Nowadays the cloud is a cost-saving opportunity for many enterprises [13] and many cloud vendors [14]. Amazon is a popular cloud service provider with solutions like Amazon Simple Storage Service S3 and the Elastic Cloud Computing EC2 as an interface to them. Eucalyptus [15] is an open source cloud implementation on top of Amazon EC2.

Being not tied to a specific hardware model, Cloud Computing enables an improved time-to-market for services achieving: a reduced infrastructure deployment time thanks to an increased service availability and reliability; rapid creation of additional service instances; and cloud interoperability, which lets professionals deploy a service on multiple clouds. Thus, cloud computing provides theoretically unlimited scalability and optimised service performance.

Since the costs of cloud solutions are a key factor, new models are required to fit better with specific applications, infrastructure environments and business contexts. These new models are classified in three, according to the different virtualisation layers: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). An example of how client devices can be integrated into cloud services is STACEE [1], which proposes a peer-to-peer (P2P) cloud storage where different devices can contribute with storage to the cloud.

4) Mobile Cloud Computing: In Mobile Cloud Computing (MCC) [3] computing resources include mobile devices as clients of the virtualised services, usually following the classical client-server asymmetric model, which involves a one way communication direction produced by requests from mobile clients to cloud services. Nevertheless symmetric MCC models have been also proposed [1], where mobile devices populate a cloud offloading the tasks to be performed by the service infrastructure. This pushes a user-centric strategy to MCC solution shifting to new models: Mobile as a Service Consumer (MaaSC), Mobile as a Service Provider (MaaSP), and Mobile as a Service Broker (MaaSB).

MaaSC model is inherited from the traditional client-server design where mobile devices are mere service consumers. Here, mobile devices outsource their computation and storage functions onto the cloud. MaaSP switches the role of the device from a service consumer to a service provider. Last but not least, MaaSB can be considered as an extension of MaaSP, where the device gateways other handhelds or sensing nodes. Moreover, the proxy mobile device can also provide security and privacy protections to the data.

MaaSC is the most common MCC service model. Most of the MaaSC solutions, such as CloneCloud, MAUI, ThinkAir, Dropbox, GoogleDrive provide computation task offloading service for mobile devices, keeping the mobile device thin. However, with the recent advances, the features of handheld device are getting closer to regular laptops catalysing new opportunities for MaaSP deployments, as it is the case of STACEE.

B. Interoperability

The interoperability in heterogeneous networks implies two abstraction levels. First, the deployment of solutions over specific architectures and operating systems. Then, the interfaces for remote operation and orchestration over a distributed system.

Despite the wide support of SETI@home, HTCondor or Eucaliptus to different architectures and operating systems, including GNU/Linux, Windows and some of the Mac OS platforms, the main drawbacks of these solutions lay on the heterogeneous computing on a variety of modern CPUs, GPUs, DSPs, and other microprocessor designs. The trend towards heterogeneous computing and highly parallel architectures has created a strong need for software development infrastructure in the form of parallel programming languages and subroutine libraries supporting heterogeneous computing on hardware platforms produced by multiple vendors [16]. In response to this completely new landscape, OpenCL [17] is a new industry standard adopted by Intel, AMD, Nvidia, Altera, Samsung, Qualcomm and ARM holdings.

Service interoperability between different cloud providers requires standard interfaces and formats for managing virtual appliances. Nowadays, due to the lack of standard way for cloud managing, each provider publishes its own APIs. In order to establish a universal connection, some proposals have been released [18]:

- OCCI [19] defines a protocol and API specification for remotely managing of cloud computing infrastructures,
- CIMI [20] targets to set an interface and a logical model for managing resources within a cloud, and
- CDMI [21] establishes an interface for manipulating data elements from the cloud.

OpenNebula [22] and Eucalyptus have made important contributions in the deployment of interoperable cloud platforms. OpenNebula implements the OCCI and CDMI specifications to enable interoperability among heterogeneous cloud platforms, whereas Eucalyptus incorporates different well-known interfaces using Amazon Web Services (AWS) [23] as a de facto standard.

The rapidly increasing use of the Web as a software platform [4] with truly interactive applications is boosted by emerging standards such as HTML5 and WebGL that are removing limitations, and transforming the Web into a real application platform middleware to address the interoperability problem. HTML5 applications can be packed for the different execution environments providing interoperability with minor changes through independent OSs. That is why HTML5 is being strongly promoted by the standardisation bodies and a sector of the market to achieve a HTML5 marketplace instead of the available proprietary ones, such as Android Market, iOS App Store, etc. All the previously described technologies put aside new breakthroughs that turn the Web into a real interoperable application framework over the heterogeneous mobile platforms.

In this line, the ComputePool component of the Nebula cloud provides computation resources through a set of volunteer compute nodes [24]. Compute nodes within a

ComputePool are scheduled by a ComputePool master that coordinates their execution. The task is executed on a compute node inside a Google Chrome Web browser-based native client sandbox. Thus it provides a secure way to access local user device computational resources inheriting Web security policies to avoid compromising users' local data.

C. Task Distribution

In our application area, social media analysis, the batch processing to be executed can be easily split into independent tasks ready to be distributed. This way, servers can dispatch the work to different processing nodes.

Focusing on generic purpose massively collaborative computation with Web technologies, MapReduce [25] has a noticeable position. It has been employed by Google to generate its search engine's index of the World Wide Web. In [26] another solution is proposed to overcome server-side task dispatching over a set of nodes, based on open source Apache Hadoop [27] frameworks. The work proposed in [28] highlights the ubiquitous nature of the image matching problems analysing some image processing algorithms specifically implemented for MapReduce technology. Another image processing projects hold by this technology are: HIPI [29] [30] that provides an API for performing image processing tasks in a distributed computing environment; and many more [31] [32] [33]. Current research goes further aggregating client-side nodes to work together with the server ones. In this direction, JSMapReduce [34] [35] is an implementation of MapReduce which exploits the computing power available in the computers of the users of a Web platform by giving tasks to the JavaScript engines of any Web browser. JSMapReduce provides simple and unique frontend for Web developers that only have to focus in JavaScript code.

MapReduce defines a programming model for processing large data sets with a parallel, distributed algorithm on a cluster. An alternative to transform the Web browser into a distributed computer middleware [36] can be also created on top of Node.js [37]. It provides more freedom to meet new requirements, to keep full code control and to ease third parties integration.

Nebula [24], which uses volunteer edge resources for both computation and data storage, assigns tasks based on application-specific computation requirements and data location. Nebula also implements numerous services and optimisations to address these challenges, including location-aware data and computation placement, replication, and recovery. Nebula considers network bandwidth along with resources computation capabilities in the volunteer platform. Consequently, resource management decisions optimise computation time as well as data movement costs. In particular, computational resources can be selected based on their locality and proximity to the input data, whereas data might be staged closer to efficient computational resources. In addition, Nebula implements replication and task re-execution to provide fault tolerance.

Finally, concerning who launches the requests for task distributions, two approaches are possible: a push model where the service delegates a set of tasks over a set of available resources, and a pull model where idle computing nodes request for new jobs to be performed.

D. Parallel Processing

The definition of smaller tasks could bring finer granularity easing efficient processing strategies based on parallel execution in some scenarios. Hence, independent job scheduling can produce significantly better performance. Here, the social media nature brings some computational benefits. First, media can be easily decomposed in independent frames or clips. Second, the possibility to perform tasks in parallel of the multimedia processing algorithms such as segmentation, clustering and classification eases to leap the scalability dimension. Third, the multimedia processing work fits with continuous advances on parallel processing of multimedia data over GPU architectures.

With the emerging hardware acceleration technologies to exploit GPU and multi-core architectures, the parallel programming languages and the hardware computing platforms are getting closer. The most representative languages that aim to enable dramatic increases in computing performance by harnessing the power of the GPU are [38]: CUDA [39] for NVIDIA devices provides a general purpose scalable parallel programming model for writing highly parallel algorithms; OpenMP [40] has established a method and language extension for programming shared-memory parallel computers. OpenMP, combined with MPI [41] specification for message passing operations, is currently the de-facto standard for developing high-performance computing applications on distributed memory architecture. The underlying mechanism consists of partitioning loop iterations according to the performance weighting of multi-core nodes in a cluster. Another mainstream options are pthreads, Cilk, Ct/RapidMind/ArBB, TBB and Boost threads [42]. These solutions remove barriers by providing abstraction layers for thread block managing, shared memory handling and synchronisation scaling.

MaaSP solutions must meet heterogeneity of browser ecosystem (Chrome, Firefox, Opera, Safari, Edge, IE). The SaW system targets all of them being able to exploit underlying hardware. The cross-entry point is bridged by WebGL and WebCL. In essence, WebGL allows communication between JavaScript applications and the OpenGL software libraries, which access the host's graphics processor. Thereby, it enables use of the hardware's full capabilities not only to perform advanced 3D objects and effects rendering but also for general purpose algorithms, such as image processing. WebCL is designed to enable Web applications with high performance and general purpose parallel processing on multi-core/many-core platforms with heterogeneous processing elements. It provides ease of development, application portability, platform independence, and efficient access through a standards-compliant solution [5]. Thus, WebGL excels in graphics applications while WebCL fares better when more flexibility is required in execution platform selection, load balancing, data formats, control flow, or memory access patterns [43].

An implementation example is CrowdCL [44]. It presents an open source framework for volunteer computing with OpenCL applications on the Web.

E. Data Structures

The scale and diversity of big data problems has inspired many innovations in recent years. Different alternatives to Relational Database Management Systems (RDBMS) have emerged to fit different big data applications.

Not only Structured Query Language (NoSQL) systems, are rapidly gaining popularity and market traction overcoming limitations of relational databases [45]. The NoSQL databases were designed to offer high performance, in terms of speed and size, with a trade-off of full ACID (Atomic, Consistent, Isolated, Durable) features [46]. These storing systems include commercial solutions such as Amazon DynamoDB, Google BigTable, and Yahoo PNUTS, as well as open source ones such as: Cassandra, used by Twitter, Facebook and some other corporations; HBase, as part of the Hadoop project; and MongoDB. All of them focus on scalability and elasticity on commodity hardware. Such platforms are particularly attractive for applications that perform relatively simple operations (create, read, update, and delete). They combine low-latency features with scaling capabilities to large sizes querying engine schedules and optimizing its execution.

NoSQL data stores offer various forms of data structures such as document, graph, row-column, and key-value pair enabling programmers to model the data closer to the format as used in their application.

III. SaW: Social at Work

Influenced by the underlying concepts and technologies, SaW system deploys an opportunistic and delay-tolerant distributed computing platform queuing media analysis tasks over a set of trusted devices. As said before, cloud services imply a cost-saving opportunity to service providers, but depending the requirements of the service, it could still be highly demanding. This kind of services usually have a huge pool of users permanently connected to it. Moreover, second screen and multi-device media experiences are becoming very popular [47] [48] [49]. In the social media scenarios considered in this paper, users access them usually from mobile devices, which have increasing processing capabilities that are mostly idle. This pushes service providers to go deeper in the cost-saving opportunity using the mobiles as an infrastructure, replacing partially cloud resources. Regarding the computing model, SaW extends the cloud computing Infrastructure as a Service (IaaS) concept to the MCC paradigm, coining a new term of Mobile as an Infrastructure Provider (MaaIP) working together with a cloud service, in a kind of MaaSP. In SaW requests match a two-way communication pattern, since servers request clients to hire resources from mobile devices. However, the different scenarios to be performed on top of the SaW system do not require a symmetric model.

To address the heterogeneity of infrastructure, and sharing Nebula design, SaW system does not require to download or install any software in the client-side thanks to a fully Web-browser based execution stack. SaW goes beyond Nebula's ComputeTool performance by emphasising the Web stack that foster hardware-accelerated parallel programming for GPU and multi-CPU over the Web browser.

Regarding task distribution, in order to address the design objectives of elasticity, performance and security, SaW server-side schedules the queued tasks meeting computation requirements and processing availability of the client devices. This asynchronous execution model ensures control to avoid duplicities for a same task, but it does not provide support for intertask communication. Moreover, to exploit parallel processing capabilities in client devices, SaW brings hardware-accelerated performance through the JavaScript engine, WebGL and WebCL, leveraging full GPU and multi-CPU potential.

Finally, related to the data structures, SaW takes advantage of the document-oriented NoSQL technologies for media repositories fitting into one-to-many relationships of a social service.

A. SaW Use Case

Service providers aim to engage audience, eager for contents, by boosting the media relevance. Therefore, it is necessary to improve the matching of user interests with the huge content database, and reveal hidden connections between items through a deeper tagging. In other words, the service is enhanced by improving the media content indexing.

The target scenario of SaW is a Web-based social media content service, such as YouTube[1] or Vimeo[2]. This target scenario brings beneficial features to the SaW system. First, this scenario provides a continuous communication channel, since users are typically consuming video content for some minutes without interruption, with an active application that provides the content through an adequate bandwidth. Users are aware of the required bandwidth to watch media content so they will try to select a high speed network or an appropriate coverage of 3G/4G mobile network. On second place, the SaW approach introduces a residual bandwidth overhead comparing with the video itself on this scenario. SaW will add an extra frame with a processing code to the connection, but it will be residual comparing to the data volume of a progressive download or streaming of a video. On third place, users do not usually perform any other task on the device while consuming video content. This ends to an underusing device with idle resources.

Finally, the use of an additional screen (e.g., a smartphone) accessing related content while consuming the mainstream video on a first screen (e.g., a TV set) [50] boosts a very favourable scenario for the SaW approach, since a single user provides multiple idle devices at the same time connected to a single service.

B. SaW Design Objectives

SaW targets a MaaIP scenario where a mobile device provides a computing component within a cloud resource system. This concept holds a key driving force moving the provisioning of processing core assets to harvesting huge amounts of available devices. Nevertheless, MaaIP, as an extension of

[1]YouTube Web site: https://www.youtube.com
[2]Vimeo Web site: https://vimeo.com

MaaSP and Mobile Grid Computing, opens some challenges such as elasticity, performance, security and privacy, that are design objectives for the SaW architecture proposed in this article.

1) Elasticity: In terms of service elasticity, it is mandatory to gain cloud ability to automatically scale services and infrastructures for cost reduction when infrastructure and platform sizes are adapted to service demands. This needs of rapid and dynamic provisioning mechanisms to provide efficient service virtualisation. This factor is even more critical in SaW, when mobiles devices come into action as an available infrastructure which is unstable, with a discontinuous connectivity and with specific features such as limited battery autonomy. This means dealing with uncertainty of the availability of the resources managed by a notification mechanism providing presence awareness and performance information. This aspect turns task independence into a major condition.

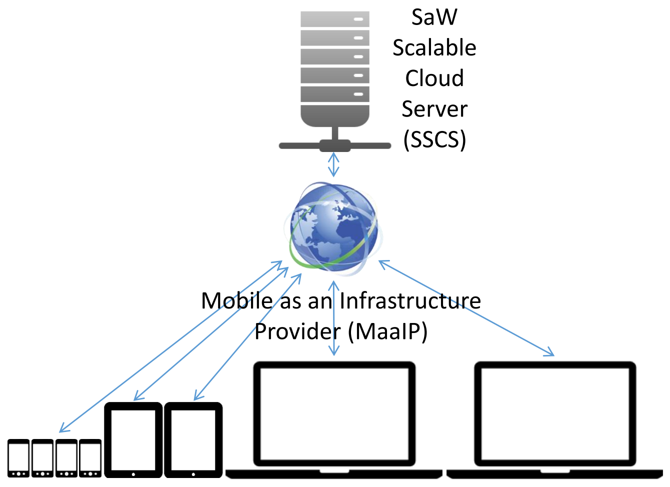
2) Performance: The objective is not only to analyse all the dataset but to perform it efficiently and in a cost-saving way. In big data a residual inefficiency is multiplied by the dataset dimension with severe impact on the global system. This means scheduling and dispatching mechanisms must be implemented to orchestrate all the elements keeping efficiency for data transmission overhead related to work delivery. Furthermore, it is important to match processing needs with device capabilities and minimise re-execution of uncompleted tasks. Thus, the Web client must perform a benchmarking test in order to assess the processing capabilities and the hardware assets disposal of the user's device

3) Security and Privacy keys: Security and privacy are major concerns for cloud infrastructures even when data is hosted on a corporative data center. However, it turns into a severe issue once the data leaves the corporative firewall. The media managed in social networks consists of images, audio and video elements shared with friends. Such information is highly privacy-sensitive, and malicious attackers may access a target user's obtaining private information. Additional issues comes when dealing with security and privacy of the node provider, the owner of the device. However, to meet both dimensions, content an device owner, a combination of policies should be applied over the data transmission and storage.

C. SaW Architecture

The deployed SaW solution works over a client-server architecture (see Figure 1). It improves the architecture presented on [8] towards a hardware accelerated approach, considering all the new aspects introduced by usage of GPU resources within SaW concept. On the server-side there is a SaW Scalable Cloud Server (SSCS) which manages server resources in order to provide a consistent, scalable and a single service front-end to the clients. It deals with balancing the load through the different available servers. The SaW client-side is completely Web browser oriented. Hence, emerging technologies such as HTML5, JavaScript, WebGL or WebCL play a crucial role by providing interoperability to cope with hardware and software heterogeneity.

All the computing and data transmission overhead in the client-side cannot affect the experience of the consumed content. Hence, on a first step, the SaW system has to create a device capabilities profile. To this end, the server adds in the first response to the client a benchmarking test in order to assess the processing capabilities and the hardware assets disposal of the user device. The score is sent to the task distribution manager of the SSCS, which decides the complexity of the background image analysis tasks that fit into that client following the global task distribution strategies.


SaW Scalable Cloud Server (SSCS)
Mobile as an Infrastructure Provider (MaaIP)
Fig. 1. General SaW system architecture diagram

On a second step, once the SSCS has set specific tasks to be run on a suitable client device, a data transfer is initiated from the server with the image frame and the image processing JavaScript script or scripts. These are classified by complexity and invoke different technologies such as WebGL or WebCL to exploit the GPU and/or multi-core assets of the device. The client applies the scripts over the images as a background process, avoiding any user experience damage. The computed results are sent back to the SSCS and it harvests, formats and stores all the incoming image computing outcome to be mined later by the service provider. While a user is enjoying a social service similar to YouTube or Vimeo, SaW allows the service to deliver independent image analysis tasks queued to the different clients until each session finishes. In case the server does not receive a result in an elapsed maximum time from a specific client, it considers that device is not available anymore and queues it to another one. Figure 2 depicts a more detailed client-server SaW architecture and the communication between them.

1) Client Web Browser SaW Architecture: The SaW approach is designed to run the client-side application over a standard Web browser composed by the following modules (see Figure 2):

Main Service: This is the main application of the service provider and gates what the user wants to consume. Note that a client using the Main Service has committed to join SaW by allowing service providers to gain idle resources in the users device to add background activities while preserving a good Quality of Experience (QoE).

Communication Layer: This module enables the communication between the client and the server with widely sup-

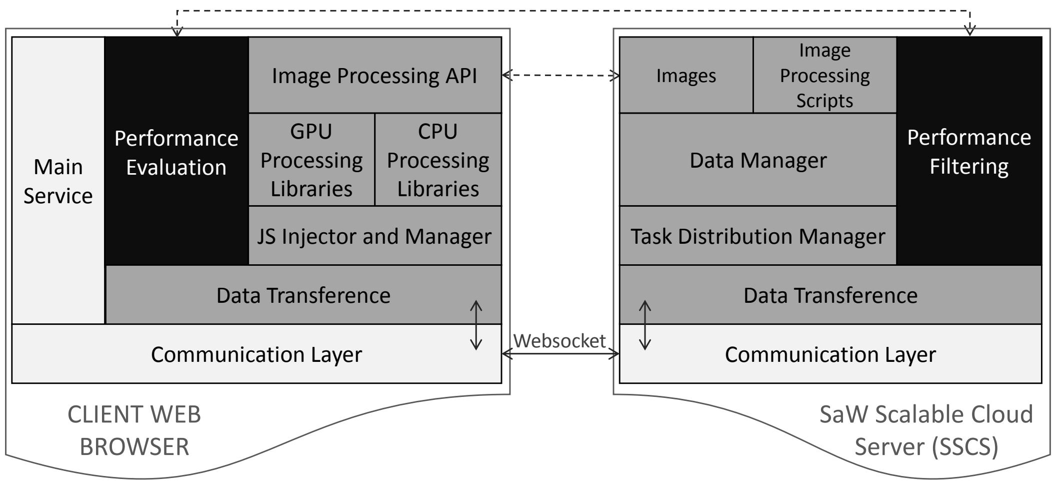


Fig. 2. SaW Client-server block diagram and its communication

ported Web communication protocols: Websocket and AJAX. The WebSocket Protocol enables two-way communication between a client and the server. Here the security model used is origin-based that is widely used by Web browsers. The protocol consists of an opening handshake followed by basic message framing, layered over TCP. The goal of this technology is to provide a mechanism for browser-based services that need two-way communication with servers that does not rely on opening multiple HTTP connections [7]. Even if the implementation of the WebSocket protocol is widely implemented and on the roadmap of all the Web browsers, nowadays there are some restrictions to use Websockets over some devices, such as mobiles. Anyway, a less efficient polling approach with Ajax is a feasible alternative to WebSocket. Ajax is a group of interrelated Web development techniques used on the client-side to create asynchronous Web applications. With Ajax, Web applications can send data to, and retrieve data from, a server asynchronously keeping visual fluidity and behavior of the foreground Web application [6].

Performance Evaluation: This module launches a performance test at the beginning of the application runtime in order to profile the capabilities of the device in terms of CPU, GPU and other features such as the autonomy of the battery or the bandwidth. Then, the benchmark results are sent to the SSCS and it settles the complexity threshold for image processing that this device can deal with considering all the discovered aspects. This evaluation test is not repeated during an active session but it could be performed again to tune the background tasks to a new context, specially if the main service is very changeable from a processing requirement perspective.

Data Transference: This module works hand in hand with the Communication Layer, dealing with the data transference between the client and the SSCS. HTML5 Web Storage facilities are used to create and maintain the incoming data. On the one hand, the client receives a new image for each image processing task and one or various scripts to be applied for that image. These are stored locally and once the processing is over, this module fits the format of the results to transfer them to the SSCS. It is important to highlight that SaW runs entirely in the memory of a Web browser.

JS Injector and Manager: It takes charge of handling the scripts received from the server. This module injects and deletes the scripts on runtime without interfering on the user experience and prepares them to execute the background tasks in an optimal way. It is very close to the following two modules enabling the scripts to take advantage of the GPU and CPU resources of an appliance.

CPU Processing libraries: Depending the performance ranking, the server decides the scripts delivered to the client oriented to exploit CPU resources. This module manages the CPU processing JavaScript libraries enabling multi-core tasks through WebCL.

GPU Processing libraries: This module wraps the JavaScript libraries ready to foster hardware acceleration by the GPU of the client device using WebGL and WebCL technologies. The availability of these assets obeys the performance assessment that are essential due to the high efficiency of the GPU oriented computing.

Image Processing API: This core layer provides the Web application an API to perform the image processing scripts on the client side. It runs image analysis tasks in the background on top of the JS Injector and Manager and using the CPU and GPU processing libraries.

2) SaW Scalable Cloud Server Architecture: The SSCS has different modules to manage all the SaW service infrastructure on the server side. These elements are presented on Figure 2 and briefly explained below:

Communication Layer: It manages the communication between the SSCS and the client. With the same functionality mentioned in the client side, this module is deployed on top of WebSocket and AJAX protocols.

Data Transference: It is supported by the Communication Layer and it is the responsible for exchanging the data with the client (e.g. the images and the scripts for each processing task).

Task Distribution Manager: This block has the global view of the SSCS to categorise and dispatch all the image analysis tasks that the service provider wants to perform through the clien device community. It splits and queues the processing jobs by connecting an image with some specific scripts and estimating the complexity of each computing work. It collaborates with the Performance Filtering module and requests and exchanges data with the Data Manager module.

Performance Filtering: This element receives the performance assessment from the clients and analyses the capabilities of the devices to inform the Task Distribution Manager module, who assigns a specific task to that device gaining specific hardware (GPU or CPU) acceleration according to its assets disposal.

Data Manager: This block manages all the data involved in the SSCS interrelated from different data-bases containing the images to be processed, the image processing scripts (some of them to be run over CPU architectures and others over GPU ones) and the results obtained by the clients. At the end, it links an outcome to data (one image) and logic (one script).

D. Considerations regarding the Design Objectives

1) Elasticity: Saw assures elasticity through Performance Evaluation and Performance Filtering modules. They profile the capabilities of the client devices in terms of CPU and GPU, but also regarding other features such as the level of battery, which is one of the most critical parameters on mobile devices. For this reason, SaW will not only consider the processing capabilities of the client devices, but also other features in order to provide flexibility to the service and let the service provider to parametrise the QoE by terms of elasticity. The elasticity parameter is considered in Section IV-B.

2) Performance: Due to the relevance of the performance in the SaW system, Section IV-B presents a performance modeling with a validation that demonstrates the feasibility of the SaW approach, and the cost-saving of complementing a cloud service with a mobile grid as an infrastructure.

3) Security and Privacy: SaW takes into consideration the security aspects regarding confidentiality and integrity. Those are assured by the well-known standard mechanisms of authentication, authorisation, encryption and auditory. As mentioned above, in SaW a client has to commit the hiring of its device resources in order to access the social media service. Thus, reciprocity conditions concerning privacy and security should be observed by the registered client and the server.

It is mandatory to verify social identity of the computation node to check its rights and permissions. The use of a centralised mechanism eases handling frequent user access privilege updates (such as invitation or revocation of access rights) in large dynamic systems like social networks.

Once the trustworthy handshake has been done, the data must be encrypted to prevent man-in-the-middle attacks. SaW deploys a temporal token based solution to limit access permissions and encrypts the data flows, with TLS protocols, for the Web communication layer.

Concerning the security threats, a push design lets SaW to prevent search over the database and DoS attacks. Note that in a pull model, an attacker, a malicious computation node, could get a promiscuous mode by notifying a permanent idle status to retrieve a set of processing tasks and associated data (crawling for later search) or to capture all the queued tasks (turning the uniformly distributed dispatching management into a burst one for a DoS attack). To prevent this behavior, in the push model of SaW, the cloud broker employs the queue of batch jobs to delegate them. This way, it is difficult for the nodes to search a specific data or claim a particular task. Moreover, server authentication could be addressed using a Synchronised Token Pattern (CSRF Token) that prevents against Cross-Site Request Forgery (CSFR) attacks [51].

Concerning possible privacy policies, privacy levels and accuracy can be defined differently within a social setting. This avoids the Task Distribution Manager to send private media content to a non-allowed end-user, even to process it in the background. For this purpose, SaW considers three types of media scopes with different set up implications: public, widening the media analysis to any available device; shared with friends, limiting the trusted area to the devices inside the social acquaintance circle; private sharing, constrained to a specific list of computing nodes from the cloud to manipulate data.

IV. Evaluation

In this Section a proof-of-concept implementation of SaW architecture is described, providing experimental evaluation results and an analysis of the performance based on a previous model. The evaluation is focused in two different aspects:

- The scalability of the SaW approach, with a specific comparison between the involved Web technologies: WebGL and WebCL.
- The performance behaviour of the system when considering different types of client devices, according to the target SaW scenarios. This includes a model of the computational cost that considers CPU, GPU and communication resources, as well as some performance figures obtained for different scenarios with realistic combination of device types.

A. WebGL and WebCL scalability comparison

This subsection presents the experimental results of using WebGL and WebCL technologies in order to explore the scalability of current Web browsers to exploit GPU resources. A proof-of-concept implementation of the SaW testbed has been developed to distribute a queue of 100 tasks over a different number of clients with identical capabilities. Using homogeneous devices enables to measure the scalability without loss of generality. The heterogeneity of the devices is addressed in next subsection.

For that purpose, a SSCS implementation has been built with a combination of Node.js and MongoDB to obtain a low latency server and to be able to deal with high concurrency requests. Both technologies provide event-driven systems that enables a non-blocking I/O model that makes it lightweight and efficient in high concurrency environments with a NoSQL data structure. Three different instances of the server have been deployed in order to avoid bottlenecks and provide sufficient resources for the different clients.

In the client side, according to the proposed architectural design, our implementation works over Web standards to cover a wide set of devices and follows a modular design. These modules enable a real-time communication with the server and are able to inject JavaScript libraries in runtime for the background tasks.

As clients, we used a different number of identical PCs with the following capabilities: Windows 8.1 Intel(R) Core(TM) i5-3330 CPU @ 3.00GHz with Intel(R) HD Graphics 2500. To test WebGL, Firefox 42.0 Web browser has been used, that enables WebGL by default [52]. Currently, there is no native support for WebCL in Web browsers. Thus, an experimental extension [53] has been used on top of a portable Firefox 22.0 Web browser.

The server creates a queue of 100 tasks with an image and an associated algorithm for each image. The sever dispatches the tasks to the available clients. Since all the clients have the same capabilities, the tasks are homogeneously distributed through all of them.

The performed algorithm computes the DITEC method (Trace transform based method for color image domain identification) [54] by means of algebraic operations such as matrix dot products that can be highly parallelised at different states (per frame, per angle during the Radon Transform operation, etc.).

We have evaluated two different SaW implementations with the aforementioned workload, the first one using WebGL and the second one using WebCL. In the performed experiments the workload has been distributed among a number of workers going from 1 to 20. The same queue of tasks has been also performed on a single PC with the same capabilities using OpenGL and OpenCL instead of doing it from the Web browser. The results obtained are shown in Table I.

Comparing the values described in Table I, obtained over the same PCs, of using a local server with OpenGL and OpenCL, with a single worker in the distributed approach with WebGL and WebCL respectively, it can be said that the local approach obtains better results. The reasons are mainly two: (1) the latency introduced by the delivery time of the image, the script and the results between the SSCS and the client in the distributed approach, and (2) the performance gap of the bindings of WebGL and WebCL to exploit the hardware resources in comparison with OpenGL and OpenCL.

From the obtained values in Table I regarding the distributed approach with different number of clients, it can be inferred that (1) the speedup is very high for both implementations, which denotes that the parallelisable fraction of the workload is very big, as expected for the described SaW use case; (2) WebCL implementation performs better than WebGL, and (3) the speedup obtained for the WebCL version is not as high as the one obtained for WebGL.

Using Amdahls Law [55] we have calculated the parallelisable fractions of the workload for both WebGL and WebCL versions, which have resulted 98.87% and 94.36% respectively. To obtain these values, we have excluded the measures obtained with one single worker, since it does not reflect the task scheduling effects of managing different workers and consolidating the results. As shown in Figure 3, Amdahls Law interpolates real measures with reasonable fidelity for the range [2, 20] of workers. Accordingly, we have used this approximations to predict results for 50 and 100 workers (see Figure 3). As appreciated in the Figure, WebCL would outperform WebGL until near 100 workers.

TABLE I
COMPUTATIONAL COST IN TERMS OF TIME FOR THE SAME WORKLOAD FOR A LOCAL SEVER WITH OPENGL AND OPENCL, AND FOR A NUMBER OF WORKERS FROM 1 TO 20 IN THE DISTRIBUTED APPROACH WITH WEBGL AND WEBCL

LOCAL SERVER

Number of workers	Computational time in ms with OpenGL	Computational time in ms with OpenCL
1	132,570	61,780

DISTRIBUTED APPROACH

Number of workers	Computational time in ms with WebGL	Computational time in ms with WebCL
1	275,295	111,300
2	161,820	73,765
5	63,260	34,290
10	34,476	19,740
20	17,825	11,120

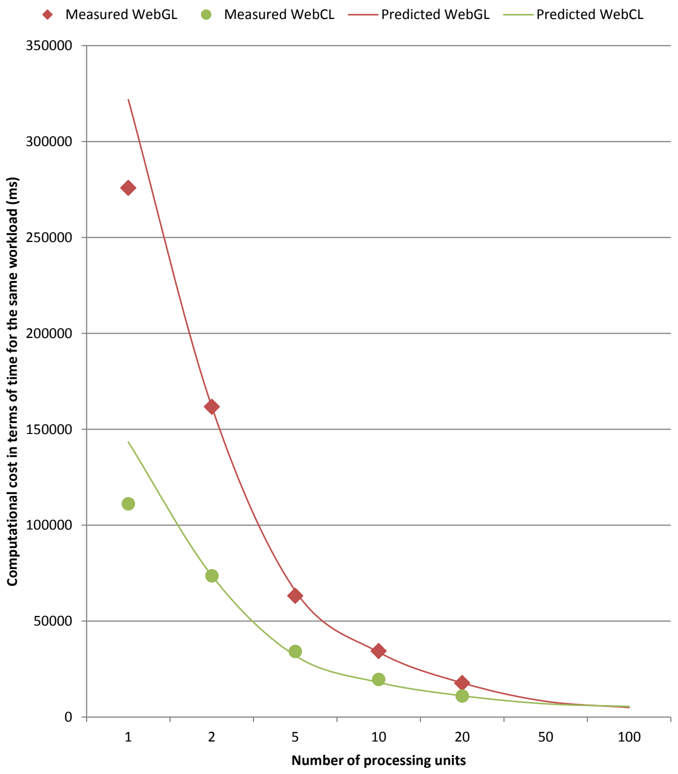


Fig. 3. Computational cost in terms of time for a different number of workers in terms of processing units in the distributed approach for WebGL and WebCL. The real measured values, presented in table I, are shown for a range of workers from 1 to 20, while predicted values, following Amdahl's Law, are shown for a range of workers from 1 to 100.

B. Performance Modeling with heterogeneous devices

In this subsection we present a performance evaluation model of the SaW approach based on the model published in [8] taking into account different type of devices. More specifically, here we take into consideration both CPU and GPU

TABLE II
ESTIMATED PROCESSING AND COMMUNICATION PROPERTIES FOR DIFFERENT TYPE OF DEVICES

ID	**Device**	**Connect.**	**Bandwidth** $\hat{b}_i$	**CPU GFlops** $\hat{F}_{ci}$	**GPU GFlops** $\hat{F}_{gi}$
(m)	Mobile phone	UMTS	3Mbit/s	0.05	0.2
(t)	Tablet	Wifi	8Mbit/s	0.08	0.32
(p)	PC	DSL	20Mbit/s	2.5	10
(s)	Server	SATA	6Gbit/s	p_Sx82.8	–

resources, while in [8] only CPU resources were contemplated.

The performance of SaW can be analysed by following the Bulk Synchronous Parallel (BSP) model [56]. The BSP model is a generalisation of the classical PRAM model [57] for shared memory.

As presented in [8], we will consider 3 different kind of devices as processing units for the distributed approach: smartphones, tablets and PCs. Table II shows different connectivity and processing power values for each one of the device types based on market surveys [58] [59]. The table includes information about a server in order to give a comparative estimation of the computational cost. According to market surveys [60], 50% of the PCs and 30% of tablets have a GPU available, while it decreases until 10% in the case of the smartphones.

Equation 1 extends the Equation 6 of [8], which considers the total computational cost (C_T) as the time to perform a workload unit distributing it across all the different type of devices in parallel.

$$C_T = \left(\sum_{i=1}^{n} \frac{1}{C_{ci}} + \sum_{i=1}^{n} \frac{1}{C_{gi}} \right)^{-1} \tag{1}$$

The equation divides the partial computation time cost for each type of device to perform their part of the workload (C_i) in two:

- C_{ci}: the partial computation time cost for each type of device that only have CPU processing capabilities to perform their part of workload.
- C_{gi}: the partial computation time cost for each type of device that have GPU and CPU processing capabilities to perform their part of the workload.
- n: the number of different type of devices. Note that according to the information of Table II, n will be 3 since the table defines three type of devices for the distributed approach: smartphones, tablets and PCs.

Equations 2 and 3 represent C_{ci} and C_{gi} respectively, extending the equation 7 of [8] and assuming sufficient resources in the server side,

$$C_{ci} = \frac{W_i}{f_{pci} \cdot \hat{F}_{ci} \cdot p_{ci}} + \frac{g_i}{f_{bi} \cdot \hat{b}_i \cdot p_{ti}} + \hat{m} \cdot p_{ci} \tag{2}$$

$$C_{gi} = \frac{W_i}{f_{pgi} \cdot \hat{F}_{gi} \cdot p_{gi} + f_{pci} \cdot \hat{F}_{ci} \cdot p_{ci}} + \frac{g_i}{f_{bi} \cdot \hat{b}_i \cdot p_{ti}} + \hat{m} \cdot p_{gi} \tag{3}$$

where:

- W_i is the computational workload in terms of the computation time assigned for device type i, to be distributed among all the different available processing units of device type i.
- g_i is the communication workload in terms of number of bytes of information to be transmitted from the server to the devices of type i.
- $\hat{m}$ is the estimated cost in terms of computation time to establish a new task to a processing unit and its management.
- $\hat{b}_i$ is the estimated bandwidth for device type i (see Table II).
- $\hat{F}_{ci}$ is the estimated CPU processing capability in terms of flops for device type i (see Table II).
- $\hat{F}_{gi}$ is the estimated GPU processing capability in terms of flops for device type i (see Table II).
- p_{ci} is the number of different processing units of device type i with only CPU capability.
- p_{gi} is the number of different processing units of device type i with both GPU and CPU capabilities. Note that following the assumption that all the devices with GPU capability will also have CPU capabilities, in Equation 3 p_{ci} and p_{gi} will be the same number of processing units.
- p_{ti} is the number of messages exchanged between the processing units of type i and the server.
- f_{pci} is an elasticity factor introduced to determine the percentage of the CPU to be used from the available CPU resources of device type i.
- f_{pgi} is an elasticity factor introduced to determine the percentage of the GPU to be used from the available GPU resources of device type i.
- f_{bi} is an elasticity factor introduced to determine the percentage of the bandwidth to be used from the available bandwidth for device type i.

As presented in [8], the same model can be used for a multi-core server approach in order to give a comparative estimation with the distributed approach. Equation 4 shows the cost of a multi-core server in terms of time (C_s) with p_s processing units sharing a single memory:

$$C_s = \frac{W}{f_{ps} \cdot \hat{F}_s \cdot p_s} + \frac{g}{f_{bs} \cdot \frac{\hat{b}_s}{p_s}} + \hat{m} \cdot p_s \tag{4}$$

In order to compare a distributed computing approach with a dedicated local server, different C_T and C_s have been calculated applying the aforementioned model. In order to avoid any annoyance in the user experience, the elasticity factor has been set to 0.15 both for bandwidth (f_b) and for processing power (f_p) for the background activities of the distributed approach. However, we considered that the local server is exclusively dedicated to these tasks (f_{ps}=f_{bs}=1).

Figure 4 shows four different cases of performance behavior for several values of model parameters. A lineal increment of processing units is compared for different values of W, g and $\hat{m}$.

As it can be observed in figure 4a, while the total distributed cost decreases for more devices, the local server starts to

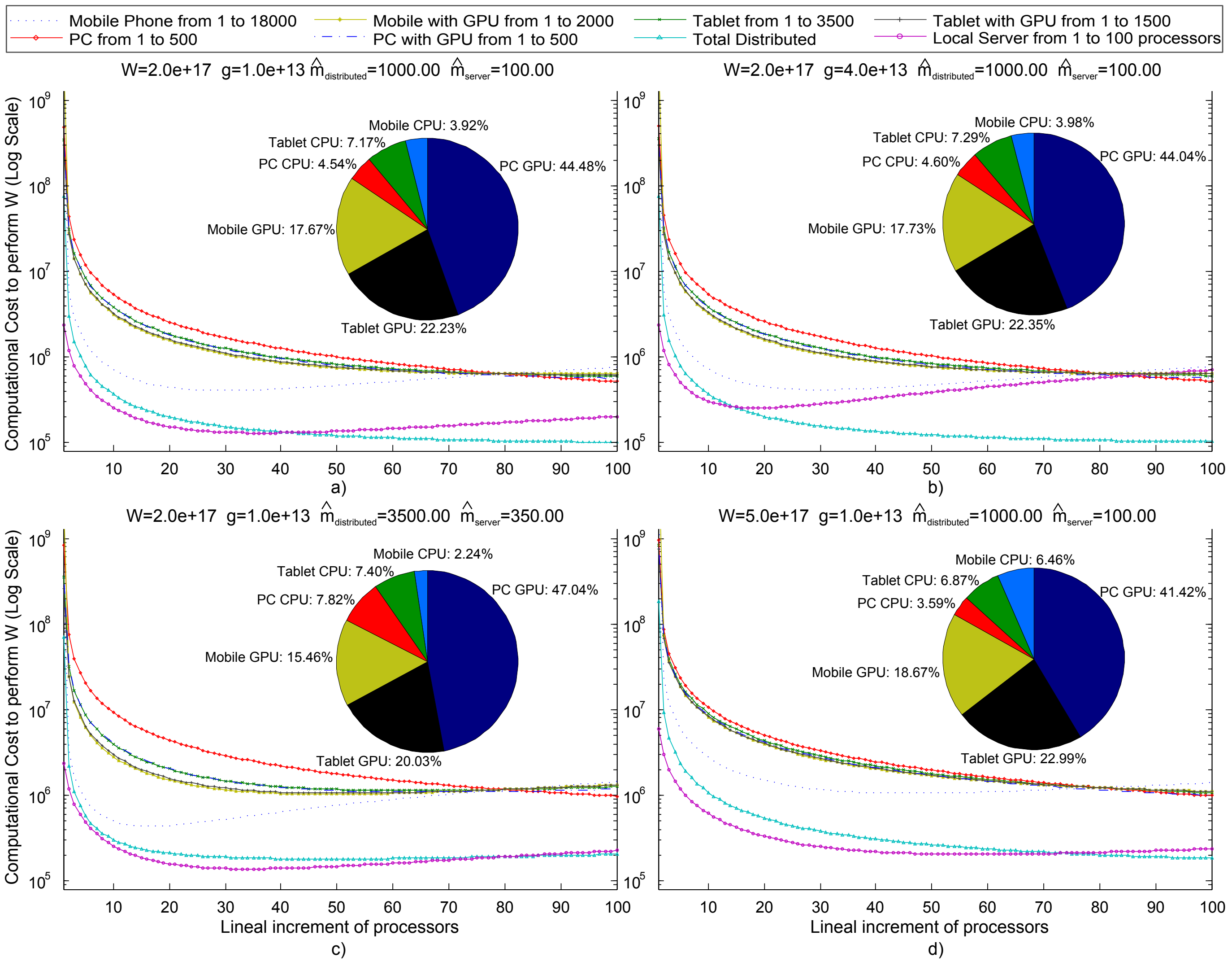


Fig. 4. Computational cost estimation under different sizes of work load (W) communication cost (g) and task management cost ($\hat{m}$: $\hat{m}_{distributed}$ for the client devices and $\hat{m}_{server}$ for the server). The pie chart presents the load balance between the different devices in the distributed approach to have the same computational cost at 80% of the X axis.

increase after reaching a minimum with 30 workers. This reflects that the data communication bus on the server is a bottleneck while, in the distributed solution, the servers should have enough bandwidth to provide the required bandwidth for each device. This trend is more clear in figure 4b, with an increased communication cost (g).

The efficient management of all the created tasks becomes a critical factor as well. This is reflected in Figure 4c, where $\hat{m}$ has been increased comparing to Figure 4a. The distributed approach has to manage thousands of devices while the local server goes from 1 to 100 workers so the management cost is more damaging to the SaW-based distributed solution.

If we increase the global workload (W), maintaining the same communication cost, a bigger gap between the workers in the local server and the devices in the distributed approach is needed to push the distributed solution performance ahead. This is reflected in Figure 4d comparing to Figure 4a. To sum up, with a enough size of user's devices partially dedicated to social service improvement it is possible for a SaW system to lead traditional server based performance. Moreover, the elasticity factor that has been set in a conservative way can be increased considerably in many scenarios without damaging the user experience.

C. Remarks

In this section some remarks are reflected regarding the validation of the SaW hypothesis that has been introduced in Section I. Recall first that the SaW approach is oriented to complement a cloud server, and not intended to beat it in computational performance. In this regard, the delay-tolerant nature of the tasks to be distributed to the clients, such as in a video tagging scenario, plays in favour of the SaW approach.

As an example, consider a task that will require a time t to be executed in the server, and, upon the results obtained in Section IV-A, assume for the workload a parallelisable fraction of about 99%. This means that the server has to spend about 1% of t for the distribution overhead, represented for the third term in Equations 2 and 3. In other words, the server would be able to manage the distribution of about 100 of these tasks in the computational time of t.

Continuing with the same example, assume now that the server is dedicated exclusively to task distribution, and that the SaW ecosystem is composed only of smartphones with about 1/50 of the processing power of the server according to Table II. Note also that to preserve QoE, the smartphones will work with an elasticity factor, say it 0.15 to be conservative, which leads to a processing capability for each one of the smartphones of 0.003 of the server power. For a set of 100 smartphones, the server will take a processing time of t to distribute the queue of 100 tasks, and will obtain all the results back in a time lapse of hundreds of t from the smartphones. This concludes that the server consumes only resources for time t, equivalent to perform a single task, and will asynchronously obtain the results for 100 tasks instead. However, the time period will be of hundreds of t.

Nevertheless, note that the example above reflects a worst-case scenario, according to the current use cases and user habits already mentioned. In a more realistic scenario, like the ones presented in the former subsection, the SaW approach would elastically distribute the workload among the different devices according their features (such as smartphones, tablets and PCs with CPU and/or GPU processing capabilities).

To summarise, the exchange of time-for-resources or, from the service provider perspective, time-for-money explained through the above example is in the core of the SaW approach, since delay-tolerance and elasticity provides sufficient freedom degrees in usual scenarios. Finally, the technological evolution also plays in favour of the approach. The performance gap between the different type of devices has been continuously reduced in the past and still continues. Besides, improvements in the bindings of Web browsers to support WebGL and WebCL can also be expected, which will result in a more efficient use of the hardware resources.

V. Conclusions

In this paper we have introduced the concept of Social at Work, SaW, which aims to complement a Web-based social media service with all the idle devices, mostly mobiles, that usually have underexploited resources while accessing the service. SaW proposes a Mobile as an Infrastructure Provider (MaaIP) model, creating a system related to Mobile Grid Computing concept with the available CPU and GPU resources of the different client devices, to complement a virtualised cloud server providing the social media service.

Aimed to achieve enhanced and automatic media tagging over social media datasets, SaW fosters background dispatching of media analysis over connected clients, providing a high elasticity and dealing with the availability of the resources related to the spontaneous presence of users. Then, SaW copes with hardware-accelerated image processing tasks execution in background, according to the capabilities of each device. The computing tasks are embedded in the foreground social content without draining the users bandwidth or affecting to the perceived Quality of Experience. In harmony with the presented scenario, delay-tolerant background tasks enable the SaW approach to exchange time-for-resources or time-for-money. This means that mobile devices, instead of being as resource intensive as servers, can dedicate the sufficient time to perform the task, preserving the QoE according to their capabilities, and saving cloud costs to service providers.

SaW deploys a powerful pure Web platform for video analysis by means of exploiting high user availability density, and the explained capability to run scripts in background threads of Web browsers. Therefore, the SaW concept targets a device community as a processing grid removing the need for install client applications, adding a delivering computing layer to the stack of the HTML5-based main service instead.

In order to evaluate the approach, we have developed a proof-of-concept implementation of SaW, including versions for existing WebGL and WebCL technologies. Results of the experiments show the high speedup obtained by parallelisation, which confirm the scalability of the approach exploiting GPU resources from Web browsers with both WebGL and WebCL technologies. The scheduling elasticity in the server side has been designed to take advantage from the delay-tolerant target scenarios, with a heterogeneous community of client devices characterised by the assorted availability of resources.

This paper has extended a previous performance model that was focused only in CPU resources, to consider both CPU and GPU capabilities. The model allows to predict the performance of a distributed system including diverse client devices, which have been illustrated through a set of example configurations, in comparison with a local server solution. The maximum benefit is obtained for higher delay-tolerant computational load, with independent tasks able to be distributed to idle devices, being able to compensate the task scheduling management and consolidation overload of the server. The technological evolution, with a clear trend to reduce the performance gap between laptops and mobile devices, as well as to improve the efficient exploitation of hardware resources from a Web browser, favours the SaW approach.

As a summary, SaW deploys a social distributed computing infrastructure on top of pure Web-based technologies, building a grid of resources to perform background media analysis tasks leveraging hardware-acceleration for a social media service.

References

[1] D. Neumann, C. Bodenstein, O. F. Rana, and R. Krishnaswamy, “Stacee: enhancing storage clouds using edge devices,” in *Proceedings of the 1st ACM/IEEE workshop on Autonomic computing in economics*. ACM, 2011, pp. 19–26.

[2] S. P. Ahuja and J. R. Myers, “A survey on wireless grid computing,” *The Journal of Supercomputing*, vol. 37, no. 1, pp. 3–21, 2006.

[3] D. Huang, T. Xing, and H. Wu, “Mobile cloud computing service models: a user-centric approach,” *Network, IEEE*, vol. 27, no. 5, pp. 6–11, 2013.

[4] M. Anttonen, A. Salminen, T. Mikkonen, and A. Taivalsaari, “Transforming the web into a real application platform: new technologies, emerging trends and missing pieces,” in *Proceedings of the 2011 ACM Symposium on Applied Computing*. ACM, 2011, pp. 800–807.

[5] W. Jeon, T. Brutch, and S. Gibbs, “Webcl for hardware-accelerated web applications,” in *TIZEN Developer Conference May*, 2012, pp. 7–9.

[6] J. J. Garrett *et al.*, “Ajax: A new approach to web applications,” 2005.

[7] I. Fette and A. Melnikov, “The websocket protocol,” 2011.

[8] M. Zorrilla, A. Martin, I. Tamayo, N. Aginako, and I. G. Olaizola, "Web browser-based social distributed computing platform applied to image analysis," in *Cloud and Green Computing (CGC), 2013 Third International Conference on*. IEEE, 2013, pp. 389–396.

[9] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *International journal of high performance computing applications*, vol. 15, no. 3, pp. 200–222, 2001.

[10] M. Conti and M. Kumar, "Opportunities in opportunistic computing," *Computer*, vol. 43, no. 1, pp. 42–50, 2010.

[11] "SETI@home Project," http://setiathome.berkeley.edu/, May 1999, [Online; accessed 26-January-2016].

[12] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[13] H. Liu, "Big data drives cloud adoption in enterprise," *IEEE internet computing*, no. 4, pp. 68–71, 2013.

[14] D. Williams, H. Jamjoom, and H. Weatherspoon, "Plug into the supercloud," *Internet Computing, IEEE*, vol. 17, no. 2, pp. 28–34, 2013.

[15] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*. IEEE, 2009, pp. 124–131.

[16] J. E. Stone, D. Gohara, and G. Shi, "Opencl: A parallel programming standard for heterogeneous computing systems," *Computing in science & engineering*, vol. 12, no. 1-3, pp. 66–73, 2010.

[17] "OpenNM Specification," http://www.khronos.org/registry/cl/, Nov 2012, [Online; accessed 26-January-2016].

[18] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Key challenges in cloud computing: Enabling the future internet of services," *Internet Computing, IEEE*, vol. 17, no. 4, pp. 18–25, 2013.

[19] "Open Cloud Computing Interface (OCCI)," http://occi-wg.org, [Online; accessed 26-January-2016].

[20] "Cloud Infrastructure Management Interface (CIMI)," http://dmtf.org/standards/cloud, [Online; accessed 26-January-2016].

[21] "Cloud Data Management Interface (CDMI)," http://www.snia.org/cdmi, [Online; accessed 26-January-2016].

[22] "OpenNebula Project," http://opennebula.org/, [Online; accessed 26-January-2016].

[23] "Amazon Web Services (AWS)," http://aws.amazon.com, [Online; accessed 26-January-2016].

[24] A. Chandra, J. Weissman, and B. Heintz, "Decentralized edge clouds," *Internet Computing, IEEE*, vol. 17, no. 5, pp. 70–73, 2013.

[25] F. O. Catak and M. E. Balaban, "Cloudsvm: training an svm classifier in cloud computing systems," in *Pervasive Computing and the Networked World*. Springer, 2013, pp. 57–68.

[26] H. Lin, X. Ma, J. Archuleta, W.-c. Feng, M. Gardner, and Z. Zhang, "Moon: Mapreduce on opportunistic environments," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*. ACM, 2010, pp. 95–106.

[27] "Apache Hadoop framework," http://hadoop.apache.org/, 2005, [Online; accessed 26-January-2016].

[28] G. De Francisci Morales, A. Gionis, and M. Sozio, "Social content matching in mapreduce," *Proceedings of the VLDB Endowment*, vol. 4, no. 7, pp. 460–469, 2011.

[29] "Hadoop Image Processing Interface," http://hipi.cs.virginia.edu/, 2012, [Online; accessed 26-January-2016].

[30] C. Sweeney, L. Liu, S. Arietta, and J. Lawrence, "Hipi: a hadoop image processing interface for image-based mapreduce tasks," *Chris. University of Virginia*, 2011.

[31] "EyeALike similarity across large datasets," http://www.eyealike.com/, 2012, [Online; accessed 26-January-2016].

[32] "Gum Gum in-image advertising platform," http://gumgum.com/, 2012, [Online; accessed 26-January-2016].

[33] "Kalooga discovery service for image galleries," http://www.kalooga.com/, 2012, [Online; accessed 26-January-2016].

[34] "JSMapReduce JS library," http://jsmapreduce.com/, 2013, [Online; accessed 26-January-2016].

[35] P. Langhans, C. Wieser, and F. Bry, "Crowdsourcing mapreduce: Jsmapreduce," in *Proceedings of the 22nd international conference on World Wide Web companion*. International World Wide Web Conferences Steering Committee, 2013, pp. 253–256.

[36] R. Cushing, G. H. H. Putra, S. Koulouzis, A. Belloum, M. Bubak, and C. De Laat, "Distributed computing on an ensemble of browsers," *Internet Computing, IEEE*, vol. 17, no. 5, pp. 54–61, 2013.

[37] S. Tilkov and S. Vinoski, "Node. js: Using javascript to build high-performance network programs," *IEEE Internet Computing*, no. 6, pp. 80–83, 2010.

[38] C.-T. Yang, C.-L. Huang, and C.-F. Lin, "Hybrid cuda, openmp, and mpi parallel programming on multicore gpu clusters," *Computer Physics Communications*, vol. 182, no. 1, pp. 266–269, 2011.

[39] "CUDA computing platform and programming model," http://www.nvidia.com/object/cuda_home_new.html, Feb 2007, [Online; accessed 26-January-2016].

[40] "OpenNM Specification," http://openmp.org/wp/openmp-specifications/, Mar 2013, [Online; accessed 26-January-2016].

[41] "MPI: Message Passing Interface," http://www.mcs.anl.gov/research/projects/mpi/, 1996, [Online; accessed 26-January-2016].

[42] S. Jarp, A. Lazzaro, and A. Nowak, "The future of commodity computing and many-core versus the interests of hep software," in *Journal of Physics: Conference Series*, vol. 396, no. 5. IOP Publishing, 2012, p. 052058.

[43] E. Aho, K. Kuusilinna, T. Aarnio, J. Pietiainen, and J. Nikara, "Towards real-time applications in mobile web browsers," in *Embedded Systems for Real-time Multimedia (ESTIMedia), 2012 IEEE 10th Symposium on*. IEEE, 2012, pp. 57–66.

[44] T. MacWilliam and C. Cecka, "Crowdcl: Web-based volunteer computing with webcl," in *High Performance Extreme Computing Conference (HPEC), 2013 IEEE*. IEEE, 2013, pp. 1–6.

[45] W. Tan, M. B. Blake, I. Saleh, and S. Dustdar, "Social-network-sourced big data analytics," *IEEE Internet Computing*, no. 5, pp. 62–69, 2013.

[46] J. Han, E. Haihong, G. Le, and J. Du, "Survey on nosql database," in *Pervasive computing and applications (ICPCA), 2011 6th international conference on*. IEEE, 2011, pp. 363–366.

[47] NAPTE, "New NAPTE and CEA research finds show producers and creators see second screen becoming permanent part of viewing experience," https://www.natpe.com/press/release/130, Q1 2014, [Online; accessed 26-January-2016].

[48] NIELSENSOCIAL, "Who's tweeting about tv?" http://www.nielsensocial.com/whos-tweeting-about-tv/, Q1 2014, [Online; accessed 26-January-2016].

[49] NIELSEN, "Sports Fans Amplify The Action Across Screens," http://www.nielsensocial.com/sports-fans-amplify-the-action-across-screens/, Q1 2014, [Online; accessed 26-January-2016].

[50] M. Zorrilla, N. Borch, F. Daoust, A. Erk, J. Flórez, and A. Lafuente, "A web-based distributed architecture for multi-device adaptation in media applications," *Personal and Ubiquitous Computing*, vol. 19, no. 5-6, pp. 803–820, 2015.

[51] A. Barth, C. Jackson, and J. C. Mitchell, "Robust defenses for cross-site request forgery," in *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 2008, pp. 75–88.

[52] "CanIUse.com Support of WebGL in different Web Browsers," http://caniuse.com/, [Online; accessed 26-January-2016].

[53] NokiaResearch, "Nokia WebCL Extension for Firefox," https://github.com/toaarnio/webcl-firefox, [Online; accessed 26-January-2016].

[54] I. G. Olaizola, M. Quartulli, J. Florez, and B. Sierra, "Trace transform based method for color image domain identification," *Multimedia, IEEE Transactions on*, vol. 16, no. 3, pp. 679–685, 2014.

[55] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, ser. AFIPS '67 (Spring). New York, NY, USA: ACM, 1967, pp. 483–485. [Online]. Available: http://doi.acm.org/10.1145/1465482.1465560

[56] L. G. Valiant, "A bridging model for parallel computation," *Communications of the ACM*, vol. 33, no. 8, pp. 103–111, 1990.

[57] S. Fortune and J. Wyllie, "Parallelism in random access machines," in *Proceedings of the tenth annual ACM symposium on Theory of computing*. ACM, 1978, pp. 114–118.

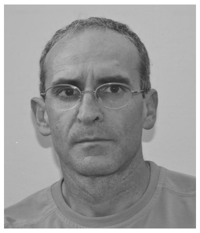
[58] TomsHardware, "AMD Bulldozer Review: FX-8150 Gets Tested," http://www.tomshardware.com/reviews/fx-8150-zambezi-bulldozer-990fx,3043-14.html, Oct 2011, [Online; accessed 26-January-2016].

[59] LegitReviews, "Google Nexus 7 Tablet Review," http://www.legitreviews.com/article/1988/2/, Jul 2012, [Online; accessed 26-January-2016].

[60] "Strategy Analytics," https://www.strategyanalytics.com/, [Online; accessed 26-January-2016].

Mikel Zorrilla is with the Department of Digital Media & Broadcasting Technologies, Vicomtech-IK4 GraphicsMedia.net. He received his Telecommunication Engineering degree in 2007 from Mondragon Unibertsitatea (Spain) and an advanced degree in Computer Science in 2012 from UPV/EHU University of Basque Country (Spain). He is a senior researcher and project manager at Vicomtech-IK4. He leads, designs and develops projects within the Interactive Media and Broadcasting Technologies field, being the technical and scientific manager of MediaScape European project (www.mediascapeproject.eu). Previously he has held positions at Ikerlan S. Coop. as an assistant researcher (2002-2006) in the field of Transport of Multimedia Traffic.

Julián Flórez studied Industrial Engineering at the University of Navarra (1980) and obtained his PhD. degree at the University of Manchester Institute of Science and Technology UMIST (1985). From 1985 to 1997, he worked as Researcher in the Centre of Study and Technical Research of Gipuzkoa (CEIT). From 1985 to 1994, he was Associate Professor in the School of Industrial Engineering of the University of Navarra. From 1997 to 2001, he worked as Director of Corporate Development of Avanzit-SGT in the fields of Communications and Broadcasting. He has a strong background in Digital Television infrastructures being involved in the deployment of one of the biggest Digital TV organizations in Europe: Quiero TV. Since 2001, he is Principal Researcher and General Director in Vicomtech-IK4 with more than 40 journal papers, 100 conference contributions and 10 patents in Industrial and Electrical Engineering.



Alberto Lafuente is a computer engineer and received a PhD. degree in Computer Science from the University of the Basque Country (UPV/EHU) in 1989. He is an associate professor of the Computer Architecture and Technology Department at the UPV/EHU, where he is a member of the BAILab research laboratory and a co-founding member of the Distributed Systems Group. His recent research activities include distributed systems and algorithms, dependable computing, pervasive systems, wireless sensor networks, and mobile systems. He has led several research projects in these fields and published more than 70 journal and conference papers.

Angel Martin is with the Department of Digital Media & Broadcasting Technologies, Vicomtech-IK4 GraphicsMedia.net. He received his engineering degree in 2003 from University Carlos III, Spain. He starts his PhD. degree in the Communications and Signal Processing Department of the University Carlos III in the video coding research area in 2003. At the same time he collaborates with Prodys developing a standard MPEG-4 AVC/H.264 codec for DSP. In 2005 he starts to work on Telefonica in projects related to multimodal interactive services for Home Networks. Also in Telefonica I+D, he goes deeper into image processing area in terms of 3D video and multiview coding. From 2008 to 2010 he worked in Innovalia as R&D Project consultant related with smart environments and ubiquitous and pervasive computing. Currently he is on Vicomtech-IK4 managing and developing R&D projects around multimedia content services.

Igor G. Olaizola is the head of Digital Media & Broadcasting Technologies Department in Vicomtech-IK4 GraphicsMedia.net, Spain. He received his PhD. degree from UPV/EHU University of the Basque Country, Spain (2013). He received his Engineering degree from University of Navarra, Spain (2001). He developed his Master thesis at Fraunhofer Institut fr Integrierte Schaltungen (IIS), Erlangen, Germany in 2001. He has participated in many industrial projects related with Digital TV as well as several European research projects in the area of audiovisual content management, being the project coordinator of MediaScape European project (www.mediascapeproject.eu). His current research interests include multimedia content analysis frameworks and techniques to decrease the semantic gap.

Iñigo Tamayo is with the Department of Digital Media & Broadcasting Technologies, Vicomtech-IK4 GraphicsMedia.net. He received his Information Systems Engineering degree in 2007 from Mondragon Unibertsitatea, Spain. He focuses on the research lines related to parallel computation, cloud systems, Web technologies, and interactive media technologies. During his studies he worked in the area of quality management of Fundacion Legarra Etxebeste (2006–2007) developing a quality control framework inside his End of Degree Project. Then, in Geoaim he was involved in the creation of a Geographical information system. Since 2008 he is working at Vicomtech-IK4, where he designs, develops and leads projects of the digital media area.

VI. Comments from the authors

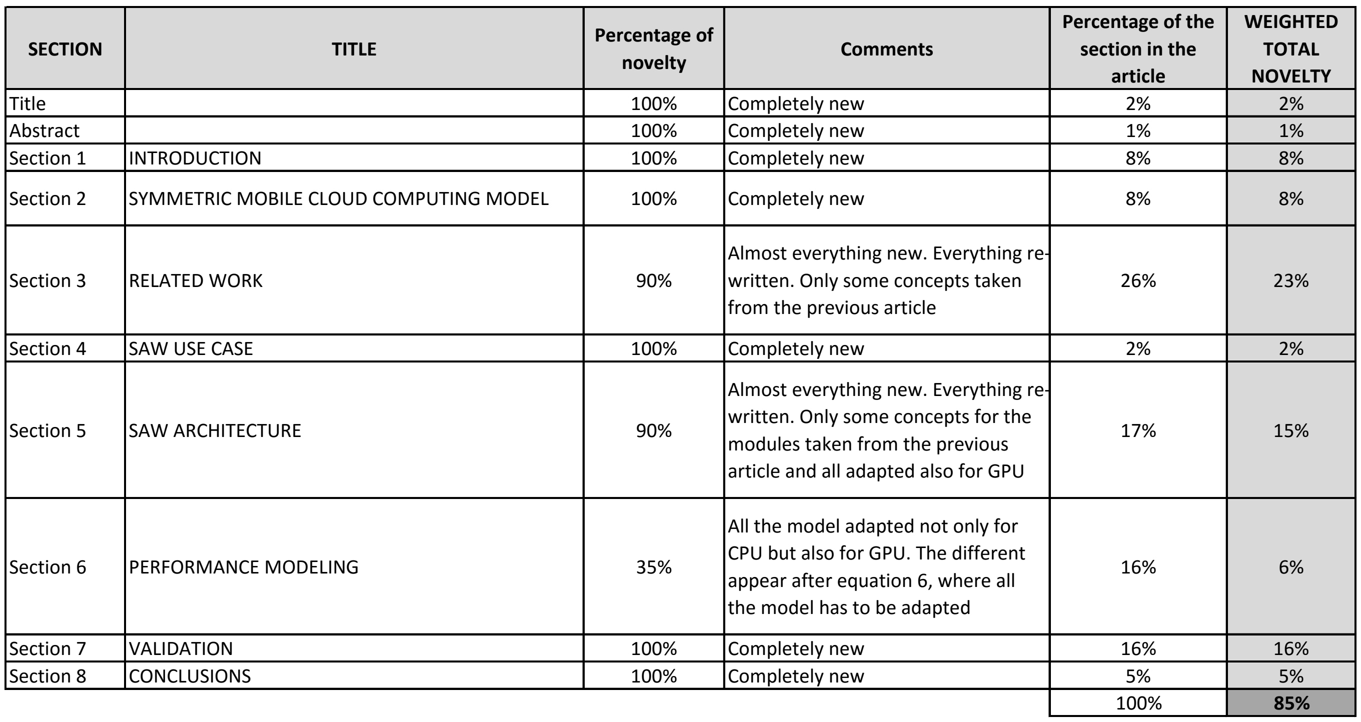
Please, find below the list of the most relevant changes that have been done following the valuable comments from the Associate Editor and the Reviewers. You will also find specific answers to the comments and questions of each one. The comments and considerations received by the authors are highlighted in grey boxes, and following them the answers from the authors can be found.

Summary of changes:

1) Section I: The Introduction has been reformulated in order to clarify the contribution of the article. On the one hand, the hypothesis to be validated has been introduced in the last paragraph of the section, just before subsection I-A. On the other hand, two subsections have been added:
   a) Section I-A: It focuses specifically in the contributions of the paper.
   b) Section I-B: It details the structure of the paper which has been completely re-organised during the revision in order to translate the contributions to the different sections of the article, and facilitate the readability and comprehension.
2) Section II: The related work contains a completely new subsection that has been introduced during the revision (Section II-A) with an overview of the different Internet-based computing models such as Grid Computing, Mobile Grid Computing, Cloud Computing and Mobile Cloud Computing. This will set the basis for the definition of SaW in Section III. On the other hand, the other sections of the related work (from Section II-B to Section II-E) have been synthesised presenting the related work in terms of interoperability, task distribution, parallel processing and data structures.
3) Section III: All the content related to the SaW concept, which is the main contribution of the article, has been completely restructured in this section. On the one hand, it provides a definition of SaW based on the previously presented Computing Models and emphasising the contributions of SaW beyond the related work. After this, the Section is divided in four different subsections:
   a) Section III-A: It reformulates the old section IV of the article with a more specific suitable scenario for the SaW approach considering second screen and multi-device paradigm while consuming media content.
   b) Section III-B: It is a new subsection defining the design objectives of the SaW architectural design.
   c) Section III-C: It contains the architectural design of SaW and has been revised and synthesised. Moreover, the authors have decided to change the name of the SaW server from SaW Scalable Server (S3) in the previous version, to SaW Scalable Cloud Server (SSCS) to avoid confusions with the Amazon S3 service mentioned in the related work.
   d) Section III-D: This new subsection shows the considerations of the aforementioned design objectives for SaW through the proposed architecture.
4) Section IV: This new section called Evaluation, includes a revised and synthesised version of the old Sections VI and VII. The evaluation is focused in three aspects through the subsections:
   a) Section IV-A: This subsection provides experimental results in order to measure the scalability of the SaW approach. Moreover, Figure 3 has been modified to separate real measures (also available in the new Table I) and the predicted results obtained with the Amdahl's Law.
   b) Section IV-B: This section provides a performance modeling addressing device heterogeneity. It has been synthesised a lot to focus only in the novelties of model regarding SaW in comparison with the already presented model in [8]. Due to the introduction of new text in other sections in order to clarify the contributions of the paper, we reduced this section to fit the article to 14 pages. Moreover, one of the concern of the reviewers was the novelty of the research of the article in comparison to the already published article by the same authors in [8]. With the new redaction of this section, it only focuses in the novelties of the model, being the whole section almost completely new content based on an existing model.
   c) Section IV-C: This subsection contains new text with some remarks regarding the validation of the SaW hypothesis that has been introduced in Section I.
5) Section V: The conclusions have been modified in order to adapt them to the new structure of the article and the introduced modifications to clarify the contributions of the paper.
6) The Title and the Abstract: The title of the article have been completely changed, since the part of the performance model has lost relevance in order the paper to gain novelty, and to try to make it easier to the readers to understand the contributions of the paper. On the same way, the abstract has been completely changed accordingly.

A. Associate Editor

Recommendation: Resubmit after Major Revision for Review



SECTION	TITLE	Percentage of novelty	Comments	Percentage of the section in the article	WEIGHTED TOTAL NOVELTY
Title		100%	Completely new	2%	2%
Abstract		100%	Completely new	1%	1%
Section 1	INTRODUCTION	100%	Completely new	8%	8%
Section 2	SYMMETRIC MOBILE CLOUD COMPUTING MODEL	100%	Completely new	8%	8%
Section 3	RELATED WORK	90%	Almost everything new. Everything re-written. Only some concepts taken from the previous article	26%	23%
Section 4	SAW USE CASE	100%	Completely new	2%	2%
Section 5	SAW ARCHITECTURE	90%	Almost everything new. Everything re-written. Only some concepts for the modules taken from the previous article and all adapted also for GPU	17%	15%
Section 6	PERFORMANCE MODELING	35%	All the model adapted not only for CPU but also for GPU. The different appear after equation 6, where all the model has to be adapted	16%	6%
Section 7	VALIDATION	100%	Completely new	16%	16%
Section 8	CONCLUSIONS	100%	Completely new	5%	5%
				100%	**85%**

Fig. 5. Novelty of the previously submitted article requested for major changes in comparison with the already published [8] article

Answer of the authors:

The authors have done major changes during the revision of the article according to all the recommendations pointed in the review. More specifically, the article has been completely restructured and new sections have been added to clarify the contribution of the paper. Thanks to the valuable comments of the reviewers the authors realised that the main contribution was unclear and most of the changes have been focused on that. The previous section provides a complete list of the relevant changes introduced in the article.

Comments to the Author: The authors should clarify the relationship between this submission and a previously published conference paper (reference[15]) by the same authors, and clearly state the new contributions.

Answer of the authors:

The authors understand that the Editor is referring to the following published article by the same authors and referenced in the new reviewed paper as [8]: *Zorrilla, M.; Martin, A.; Tamayo, I.; Aginako, N.; Olaizola, I.G., ”Web Browser-Based Social Distributed Computing Platform Applied to Image Analysis,” in Cloud and Green Computing (CGC), 2013 Third International Conference on , vol., no., pp.389-396, Sept. 30 2013-Oct. 2 2013”.*

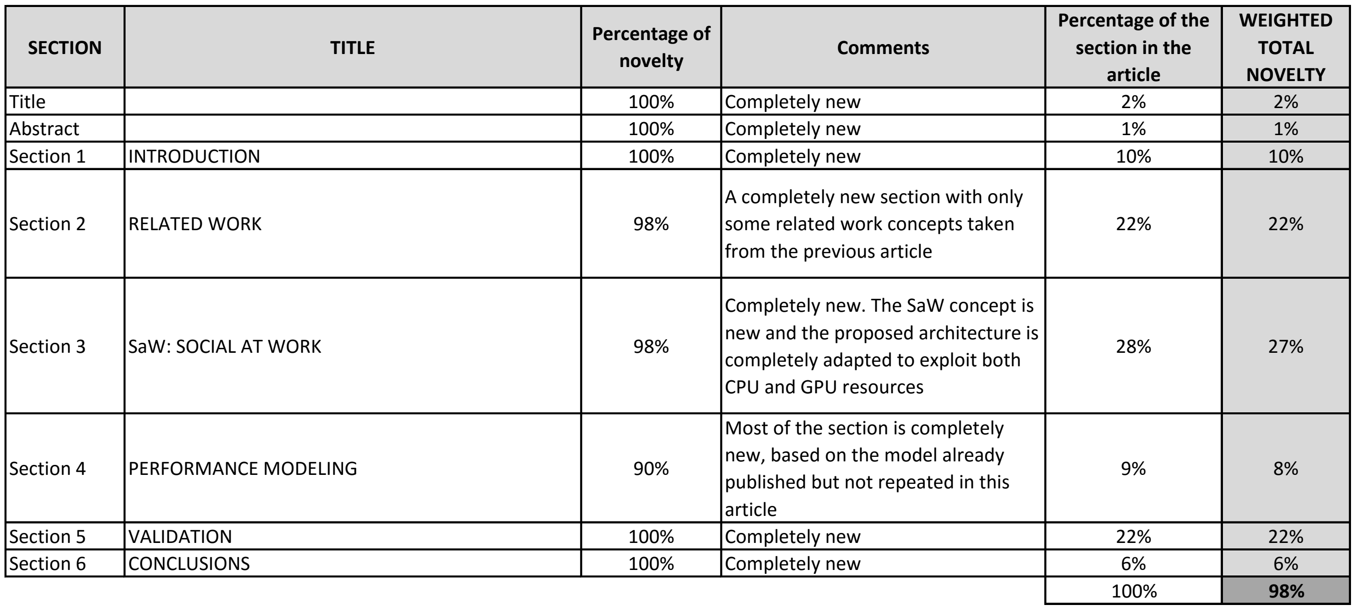
The authors estimate that the 85% of the article that was submitted for the review was new comparing to the aforementioned article. This estimation has been done analysing the percentage of novelty of each one of the sections and then assigning them a weight depending the length of each section in comparison with the whole article. Figure 5 summarises the outcome of the 85% value.

Moreover, after the revision with major changes, the novelty of the article has increased substantially. As it is pointed in change number 4b from the list of ”summary of changes”, the performance modeling section has been synthesised focusing only in the novelties comparing to the model taken as a basis, and adapting it to the needs of SaW. The authors estimate, following the aforementioned methodology, that this article is almost totally new with respect to [8], overtaking the 98% of novelty. Figure 6 summarises the outcome of the 98.2% value.

Please also provide and discuss a more convincing performance metric for the proposed architecture, to address one of the major concerns of the reviewers.

Answer of the authors:

The aforementioned changes address a clarification of the contributions of the paper, including a better explanation of the



SECTION	TITLE	Percentage of novelty	Comments	Percentage of the section in the article	WEIGHTED TOTAL NOVELTY
Title		100%	Completely new	2%	2%
Abstract		100%	Completely new	1%	1%
Section 1	INTRODUCTION	100%	Completely new	10%	10%
Section 2	RELATED WORK	98%	A completely new section with only some related work concepts taken from the previous article	22%	22%
Section 3	SaW: SOCIAL AT WORK	98%	Completely new. The SaW concept is new and the proposed architecture is completely adapted to exploit both CPU and GPU resources	28%	27%
Section 4	PERFORMANCE MODELING	90%	Most of the section is completely new, based on the model already published but not repeated in this article	9%	8%
Section 5	VALIDATION	100%	Completely new	22%	22%
Section 6	CONCLUSIONS	100%	Completely new	6%	6%
				100%	**98%**

Fig. 6. Novelty of the article submitted after major changes in comparison with the already published [8] article

performance metric for the proposed architecture and a deeper discussion about the results obtained in the proof-of-concept implementation for the validation.

Key references that must be included: From reviewers' comments (please check this section): The submitted paper is presented as an original work but it seems to be a refined version of a method already published by the same author in the following conference paper: "Zorrilla, M.; Martin, A.; Tamayo, I.; Aginako, N.; Olaizola, I.G., "Web Browser-Based Social Distributed Computing Platform Applied to Image Analysis," in Cloud and Green Computing (CGC), 2013 Third International Conference on , vol., no., pp.389-396, Sept. 30 2013-Oct. 2 2013"

Answer of the authors:

As explained before, the authors consider that the novelty of the article is clarified following Figures 5 and 6.

B. Reviewer 1

This work aims to develop advanced meta-data creation (content based information retrieval) for video sharing services. Their approaches are based on Social at work (SaW) with WebGL, WebCL.

The submitted paper is presented as an original work but it seems to be a refined version of a method already published by the same author in the following conference paper: "Zorrilla, M.; Martin, A.; Tamayo, I.; Aginako, N.; Olaizola, I.G., "Web Browser-Based Social Distributed Computing Platform Applied to Image Analysis," in Cloud and Green Computing (CGC), 2013 Third International Conference on , vol., no., pp.389-396, Sept. 30 2013-Oct. 2 2013". The novelties proposed in the submitted work are not clear to me. The aim of the work should be better described (Is this a deeper investigation of an already proposed methodology? What are the novelties introduced?).

Answer of the authors:

As already explained to the Editor, the authors estimate that the 85% of the article that was submitted for the review was new comparing to the aforementioned article, following Figure 5. Moreover, after the major changes revision, the novelty of the article has increased substantially. As it is pointed in change number 4b from the list of "summary of changes", the performance modeling section has been synthesised focusing only in the novelties comparing to the model taken as a basis, and adapting it to the needs of SaW. The authors estimate that this article is now almost totally new, overtaking the 98% of novelty in comparison with [8] (Figure 6).

We hope that the new structure of the article, changed to underline the contributions of the paper, helps to understand the aim of the work and the novelties introduced.

Here some additional comments I hope can be helpful to improve the work:

- Page 6, line 8: There is a term ”trusted devices”. But I cannot find any definition what does it mean. How can the system trust a device. What is the authentication/authorizaton system to trust a device?
- The authors said that ”Such information is highly privacy-sensitive, and malicious attackers may access a target user’s obtaining private information”. But in Section V, Subsection C, the authors expalains the security infrastructre of the propsed model. The data is encrypted using SSL/TLS and the model records auditing logs. All of the these modules secures the system not prevents the privacy. Privacy is different from the security. I would suggest to authors that Subsection C should be ”Security”.
- In same subsection the authors said that ”Push” model prevents an attacker or a malicious computation node. But there is still a security leakeage in ”Pull” model. A malicious node in ”Pull” model can be addded to network. What can prevent the system against attackers and malicious nodes.

Answer of the authors:

The reviewer is right. Privacy is different from security and it was not clearly reflected in the previous submission. We changed the title of the subsection as proposed by the reviewer and we tried to reflect both the security and privacy issues. Anyway, this is not a contribution of the paper since the proposed architecture is designed over well-known existing mechanisms for authorisation and authentication as well as for data encryption.

In terms of trusted devides and how to prevent attacks on a pull model, we introduced new text in Section III-D3. Again, it is not a contribution of the paper but using existing mechanisms to overcome security issues in the SaW system.

Overall, this submission needs some improvements in both technical and information delivery aspects.

Answer of the authors:

The authors hope that the list of relevant changes done during the major review of the article help to improve in both technical and information delivery aspects, making clear what the contribution of the paper is.

C. Reviewer 2

This paper introduces an approach to offload the computation from cloud to edge devices, and potentially leverage the GPUs on these devices.

To me the assumption of this paper is not very valid. I agree that we should leverage the devices to do some local computing before they have to use cloud. However, this does not indicate that we should dispatch tasks which are already on cloud to the edge devices, and get the result back to cloud. Doing it most likely will consume more time and energy.

Answer of the authors:

The authors realised that the contribution of the paper was not well reflected in the previous version. The authors have done major changes and completely restructured the article in order to clarify the contributions.

To make clear the validity of the approach, the hypothesis has been formulated at the end of the Introduction (just before Section I-A), and evaluated and validated in Section IV, as well as summarised in the Conclusions. The SaW approach is aimed to obtain a net benefit in the server workload by delegating part of the tasks to the users’ devices without disturbing the user experience. The paper explores the conditions in which the extra work created in the server to manage the task scheduling will not exceed the saved work. We proof that the approach is valid with the current technology and the presented scenario, and will be increasingly convenient in the future, as the performance gap between mobile devices and laptops is being reduced, as well as the implementation of the Web browsers is improving, creating efficient bindings to exploit the hardware resources from the browser.

In the SaW approach the authors are not proposing to do local computing before using the cloud. Instead, the SaW approach is complementary to a cloud service. For the end user perspective, the users will access to a social network media service in the cloud and they won’t perceive any inconvenience in the Quality of the Experience. However, from the service provider point of view, SaW proposes to have a mobile grid as an infrastructure to complement the cloud servers, exploiting their GPU and CPU resources that are usually idle while consuming a social media service. This enables service providers to dispatch delay-tolerant tasks to client devices, saving cloud resources and without interfering the experience of the user thanks to the elasticity factors, to complement their service with video analysis tasks.

The experiments show that, obviously, using more devices will get better parallelism and improve the latency. However:

1) how do you handle unreliable network connection, and power consumption of the devices?
2) what about the per watt computation you get? Is it more economic to offload to devices, or to rent more cloud resources? This evaluation has to be done to show the effectiveness of the proposed approach.

Answer of the authors:

Of course, as the reviewer says and our experiments show, the more devices we have to complement a media service, the better parallelism we obtain. However, this is not the validation we are proposing in the article, but compensating the extra workload added to the server to manage the task scheduling, with the offloaded distributed workload. As our results show, the highly parallelisable nature of the workload allows for high speedups, which could compensate in many scenarios the restrictions of the client devices in terms of performance and QoE.

In the SaW use cases, the client-side background tasks are only performed while consuming the main social media service. Background tasks are delay-tolerant, such as automatically tagging the user generated content to improve the service, but not to supply the main service itself. If a client device is doing a background task and the user closes the Web browser, once the task manager realises that there is a timeout, it will distribute the same task to another device, which is not critical.

One of the limitations of the mobile devices is the battery autonomy. For this reason, our proposal introduces some elasticity factors, both for the usage of the processing capabilities and for the bandwidth. This way, depending the level of battery of the devices and if it is plugged or not to the electrical power, the task manager will assign tasks accordingly. From a service provider perspective, it will be always more economical to offload workload to the devices than renting more cloud resources for the proposed scenarios, when the task scheduling overload can be compensated and the QoE of the users can be preserved thanks to the elasticity factors. Note that the paper is not focused on the energy aspects of consumption and efficiency on the client devices. However, the considered elasticity factors limit the use of client devices in terms of battery autonomy and energy consumption.