IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 9, Issue 6, June 2020

Video Analysis in Social Media with Web Based Mobile Grid Computing

Mr.Aditya Gaikwad[1], Prof.Rokade M.[2]

PG Student, SP'COE, DUMBARWADI (OTUR), SPPU, Maharashtra, India [1]

Associate Professor, SP'COE, DUMBARWADI (OTUR), SPPU, Maharashtra, India [2]

Abstract: In this project, we first survey the current situation of video processing on the edge for multimedia Internet-of-Things (M-IoT) systems in three typical scenarios, i.e., smart cities, satellite networks, and Internet-of-Vehicles. By summarizing a general model of the edge video processing, the importance of developing an edge computing platform is highlighted. Then, we give a method of implementing cooperative video processing on an edge computing platform based on light-weighted virtualization technologies. Performance evaluation is conducted and some insightful observations can be obtained. Moreover, we summarize challenges and opportunities of realizing effective edge video processing for M-IoT systems.

Keywords: M-IoT, SaW, MaaIP, SSCS.

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 deep automatic tagging enables better matching of user interests with the content database and reveals underlying connection 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 analyzed. 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 analyses all the content. 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 client devices, mostly mobiles that usually have underexploited resources while accessing the service. SaW proposes a Mobile as Infrastructure Provider (MaaIP) model, going beyond the Infrastructure as a Service (IaaS) model, and Creating a system related to Mobile Grid Computing 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 backgrounds Web browser applications must balance the mechanism to leverage all available computing resources while provide the best possible user experience. 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 analyzed. 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 analyses all the content. Grid and Cloud technologies provide High Performance Computing systems that aim to satisfy these requirements. However, as pointed in, other under-explored alternatives could enhance the trade-o_ 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. 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 client devices, mostly mobiles that usually have underexploited resources while accessing the service. SaW proposes a Mobile as an Infrastructure Provider (MaaIP) model, going beyond the Infrastructure as a Service (IaaS) model, and creating a system related to Mobile Grid Computing concept with the available CPU and

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 9, Issue 6, June 2020

GPU resources of the different client devices to complement a virtualized cloud server, which provides the social media service. Inspired by the Mobile Grid Computing and the Mobile Cloud Computing (MCC) 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, 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.

II. SYSTEM OVERVIEW

The deployed Saw solution works over client-server architecture. It improves the architecture presented on 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. Algorithms 1 and 2 provide an example of SaW with the client device benchmarking process and the SSCS workflow respectively. SSCS executes two concurrent tasks in Algorithm 2. First, by Enroll Thread, SSCS continuously performs a recruitment loop which orders the new devices connected to the social media service to self-assess their performance scores. Thus, SSCS enrolls the devices in the appropriate queue based on the reported type following Algorithm 1, which is executed in client devices remotely and provides, as an outcome, a normalized device type i according to the classification Second, by Task Distribution, SSCS matches the queued image processing tasks to suitable devices in terms of workload and elasticity factors. To this end, the image size and algorithm complexity are considered.

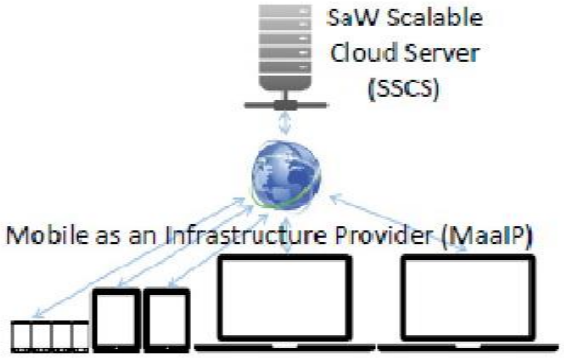

Fig -1: System Design

A) Algorithms

Algorithm 1 Device benchmark example

procedure BENCHMARK(d_{id}) ▷ assessed at each device
Input: d_{id} ▷ device ID from social media session
$\hat{b_d}$ ▷ estimated bandwidth for device
$\hat{F_{cd}}$ ▷ estimated CPU processing capability
$\hat{F_{gd}}$ ▷ estimated GPU processing capability
$i \leftarrow$ getDeviceType($\hat{b_d}$, $\hat{F_{cd}}$, $\hat{F_{gd}}$)
report i ▷ send normalised device type to the SSCS following the classification in Table II

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 9, Issue 6, June 2020

Algorithm 2 SSCS workflow example

```
procedure ENROLLTHREAD( )                          ▷ SSCS recruitment loop
Data: qD                          ▷ N queues by type of available devices
    for each newSession d_id do                    ▷ new connection
        i ← Benchmark(d_id)                        ▷ get type from the device
        queue(qD_i, d_id)                          ▷ queue device based on type

function TRANSFER(Σ_j, Ω_k, d_id, i)               ▷ send task to a device
                                                     of type i
Input: Σ_j                                         ▷ image to be processed
Input: Ω_k                        ▷ programmed image processing algorithm
Input: d_id                         ▷ device ID from social media session
Input: i                                           ▷ normalised device type
Data: qD_i                        ▷ queue with available devices of type i
    deliverTask (Σ_j, Ω_k) to d_id                 ▷ deliver task to resource
    waitTTL(i)                    ▷ wait estimated TTL for device type i
    if error then
        return error_msg                           ▷ error
    if timeout then
        dequeue(qD_i, d_id)                        ▷ remove unresponsive device
        return error_msg                           ▷ timeout
    return ok                                      ▷ ok

function TASKDISTRIBUTION(Ω_k)                     ▷ Tasks dispatching loop
Input: Ω_k                        ▷ programmed image processing algorithm
Data: qΣ                                           ▷ images queue
Data: qD                  ▷ N queues with available devices based on type
    while !empty(qΣ) do                ▷ more images in the queue qΣ
        Σ_j ← dequeue(qΣ)                          ▷ next image to be processed
        z ← getTargetDevice(Σ_j, Ω_k)              ▷ target device type under
                                                     elasticity factors
        for i = N to z do                          ▷ start from more powerful devices
                                                     N, stop at target devices z
            if !empty(qD_i) then
                d_id ← dequeue(qD_i)
                Transfer(Σ_j, Ω_k, d_id, i)        ▷ assign task to resource
                if !ok then
                    queue(qΣ, Σ_j)                 ▷ re-queue image
    return completed                               ▷ completed

procedure MAINLOOP( )                              ▷ SSCS main loop
    EnrollThread()                                 ▷ recruitment loop
    while !empty(qΩ) do            ▷ more algorithms in the queue qΩ
        Ω_k ← dequeue(qΩ)                          ▷ next batch processing
        TaskDistribution(Ω_k)                      ▷ tasks dispatching loop
```

III.RESULTS AND ANALYSIS

A. Data storage

The report server stores all objects, properties and metadata in a SQL Server database. Stored data includes Published information, models of reports and the folder ladder organized by the report server. Internal storage for a single Reporting Services installation or for multiple eport servers that are part of a scale-out deployment is provided by report server.

B. Data processing extensions

Data Processing extensions are used to query a data source and return a flattened row set. Different extensions are used by reporting services to interact with different types of data sources. The extensions included in reporting can be used or your own extensions can be developed. Data processing extensions are provided for data sources such as SQL

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319-5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 9, Issue 6, June 2020

Server, Analysis Services, Oracle, SAP Net Weaver Business Intelligence, Hyperion Ess base, Terradata, ODBC, and OLEDB. ANY ADO.NET data provider can also be used by Reporting Services. Data processing extensions process query requests from the Report Processor component by carrying out the below tasks:

- A connection to a data source should be opened.
- Study a query and return a list of field names.
- Run a query against the data source and return a row set.
- If required, Pass parameters to a query.
- Retrieve data by Iterating through the row set.

Some extensions can also perform the following tasks:

- Analyze a query and return a list of parameter names used in the query.
- A query should be analyzed and the list of fields used for grouping and sorting should be returned.
- A username and password should be provided to connect to data source..
- Iterate through rows and retrieve auxiliary metadata.
- XML rendering extension: Reports are rendered in XML files by the XML rendering extension. . These files can then be stored or read by other programs. An XSLT transformation can also be used to turn the report into another XML schema to be used by another application. The XML generated by the XML rendering extension is UTF-8 encoded.

- Image rendering extension: The Image rendering extension renders reports to bitmaps or metafiles. Reports are rendered by the extension in following formats: GIF, JPEG, BMP, EMF, PNG, TIFF, and WMF. By default, the image is rendered in TIFF format that can be shown with the default image viewer of your operating system .Using the Image rendering extension to render reports ensures that the report looks the same for every client. When a report viewed in HTML by the user, the appearance of the report can differ depending on the version of the user's browser. The extension that renders image also renders the report on the server, so all users see the same image. As the report is rendered on the server, installation of all the fonts used in the server must be done.

A. ***Complexity***

Complexity Measures

In this section, we briefly explain the basic ideas behind Kolmogorov complexity and sparse signal representation. The Kolmogorov Complexity The complexity of image or video is somewhat related to its randomness. Let take a string1 complex as compared to another string2 because the string2 contains some regular occurring word whereas in string1 there is no randomness. Kolmogorov Complexity use this type of features. Given a finite object X, its Kolmogorov complexity K(X) is defined as the length of the shortest program that can effectively reconstruct X on an universal computer, such as a Turing machine. K(X) is the ultimate lower bound among all measures of information content of X. The conditional Kolmogorov complexity K(X,Y) of X relative to Yis defined as the length of the shortest program that can generate X when Y defined to be the length of the shorte concatenated string XY. The normalized distance is given by formula NID = Normalized Information Distance X - String1 Y-String2

One of the issues with Kolmogorov complexity is that the briefest depiction of a string can run gradually. Truth be told, it should frequently do as such, for fear that the arrangement of arbitrary strings have an unending calculably enumerable subset. One exceptionally helpful variation of established many-sided quality is the time-limited variant.

B. Sparse Representation

The main idea of sparse representation is to represent data with linear combination of very tiny number of basis function. Sparsely is not so easy to achieve. Wavelets and sinusoidal succeeded in scarifying some of the data from raw dataset, but in practice, it is not like that. In reality the dataset are mixed structures that is hard to be captured by wavelets and sinusoidal alone.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 9, Issue 6, June 2020

C. ***Graphical representation***

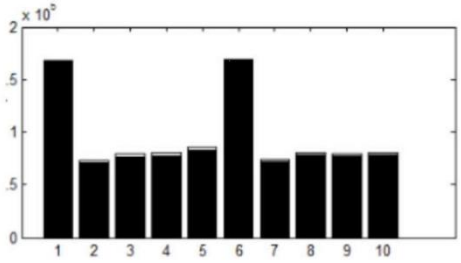

Fig2:- Count of total bit per frame

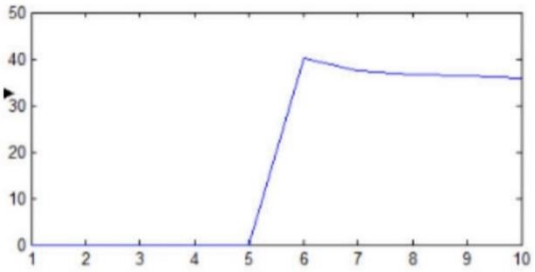

Fig3:- Count of PSNR value per frame

IV. CONCLUSION

In this report 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 virtualized 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 resource Related to the spontaneous presence of users. 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 according to their capabilities, and saving cloud costs to service providers.

ACKNOWLEDGMENT

I want to thanks to my guide **Prof.Rokade** for her esteemed guidance and encouragement, especially through difficult times. His suggestions broaden my vision and guided me to succeed in this work. I am also very grateful for his guidance and comments while designing part of my research paper and learnt many things under his leadership.

REFERENCES

[1]. Mikel Zorrilla, Juli´an Fl´orez, Alberto Lafuente, Angel Martin, Jon Montalban, Igor G. Olaizola, Member, IEEE, and Inigo Tamayo SaW: Video Analysis in Social Media with Web-based Mobile Grid Computing DOI 10.1109/TMC.2017.2766623, IEEE Transactions on Mobile Computing
[2]. Botan I et al. (2010) SECRET: a model for analysis of the execution semantics of stream processing systems. Proc VLDB Endow 3(1–2):232–243
[3]. Salathé M et al. (2012) Digital epidemiology. PLoSComputBiol 8(7):1–5
[4]. Bollen J, Mao H, Zeng X (2011) Twitter mood predicts the stock market. J ComputSci 2(3):1–8
[5]. Chandramouli B et al (2010) Data stream management systems for computational finance. IEEE Comput 43(12):45–52
[6].Chandrasekar C, Kowsalya N (2011) Implementation of MapReduce Algorithm and Nutch Distributed File System in Nutch. Int J ComputAppl 1:6–11
[7].Cioffi-Revilla C (2010) Computational social science. Wiley Interdiscip Rev Comput Statistics 2(3):259–271
[8].Galas M, Brown D, Treleaven P (2012) A computational social science environment for financial/economic experiments. In: Proceedings of the Computational Social Science Society of the Americas, vol 1, pp 1–13
[9].Hebrail G (2008) Data stream management and mining. In: Fogelman-Soulié F, Perrotta D, Piskorski J, Steinberger R (eds) Mining Massive Data Sets for Security. IOS Press, pp 89–102
[10]Mr.Aditya Gaikwad, Prof.Rokade (2020) A Survey on Video Analysis in Social Media with Web Based Mobile Grid Computing, Vol-6 Issue-3 2020 IJARIIE-ISSN(O)-2395-4396