Future Generation Computer Systems 55 (2016) 444–459

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Resource management for bursty streams on multi-tenancy cloud environments☆

Rafael Tolosana-Calasanz [a,*], José Ángel Bañares [a], Congduc Pham [b], Omer F. Rana [c]

[a] Computer Science and Systems Engineering Department, Aragón Institute of Engineering Research (I3A), University of Zaragoza, Spain

[b] LIUPPA Laboratory, University of Pau, France

[c] School of Computer Science & Informatics, Cardiff University, United Kingdom

HIGHLIGHTS

- We provide a system for simultaneous bursty data streams on shared Clouds.
- We enforce QoS based on a profit-based resource management model.
- We provide real experiments within an OpenNebula based data centre.

ARTICLE INFO

Article history:
Received 5 May 2014
Received in revised form
9 March 2015
Accepted 11 March 2015
Available online 19 March 2015

Keywords:
Data stream processing
Cloud computing
Profit-based resource management
SLA management
Admission control
QoS provisioning

ABSTRACT

The number of applications that need to process data continuously over long periods of time has increased significantly over recent years. The emerging Internet of Things and Smart Cities scenarios also confirm the requirement for real time, large scale data processing. When data from multiple sources are processed over a shared distributed computing infrastructure, it is necessary to provide some Quality of Service (QoS) guarantees for each data stream, specified in a Service Level Agreement (SLA). SLAs identify the price that a user must pay to achieve the required QoS, and the penalty that the provider will pay the user in case of QoS violation. Assuming maximization of revenue as a Cloud provider's objective, then it must decide which streams to accept for storage and analysis; and how many resources to allocate for each stream. When the real-time requirements demand a rapid reaction, dynamic resource provisioning policies and mechanisms may not be useful, since the delays and overheads incurred might be too high. Alternatively, idle resources that were initially allocated for other streams could be re-allocated, avoiding subsequent penalties. In this paper, we propose a system architecture for supporting QoS for concurrent data streams to be composed of self-regulating nodes. Each node features an envelope process for regulating and controlling data access and a resource manager to enable resource allocation, and selective SLA violations, while maximizing revenue. Our resource manager, based on a shared token bucket, enables: (i) the re-distribution of unused resources amongst data streams; and (ii) a dynamic re-allocation of resources to streams likely to generate greater profit for the provider. We extend previous work by providing a Petri-net based model of system components, and we evaluate our approach on an OpenNebula-based Cloud infrastructure.

1. Introduction

The number of applications that need to process data continuously over long periods of time has increased significantly over recent years. Often the raw data captured from the source is converted into complex events—which are subsequently further analysed. Such applications include weather forecasting and ocean observation [1], text analysis (especially with the growing requirement to analyse social media data, for instance), "Urgent Computing" [2], and more recently data analysis from electricity meters to

☆ This work was partially supported by the OMNIDATA project funded by the Aquitaine–Aragon collaboration research program between Aquitaine region (France) and Aragon region (Spain); and by the Spanish Ministry of Economy under the program "Programa de I+D+i Estatal de Investigación, Desarrollo e innovación Orientada a los Retos de la Sociedad", project identifier TIN2013-40809-R.

* Corresponding author.
E-mail address: rafaelt@unizar.es (R. Tolosana-Calasanz).

http://dx.doi.org/10.1016/j.future.2015.03.012
0167-739X/

support "Smart (Power) Grids" [3]. The emerging Internet of Things and Smart Cities scenarios also strongly confirm that increasing deployment of sensor network infrastructures generate large volumes of data that are often required to be processed in real-time. Data streams in such applications can be large-scale, distributed, and generated continuously at a rate that cannot be estimated in advance. Scalability remains a major requirement for such applications, to handle variable event loads efficiently [4].

Multi-tenancy Cloud environments enable such concurrent data streams (with data becoming available at unpredictable times) to be processed using a shared, distributed computing infrastructure. When multiple applications are executed over the same shared elastic infrastructure, each stream must be isolated from the other in order to either: (i) run all instances without violating their particular Quality of Service (QoS) constraints; or (ii) indicate that, given current resources, a particular stream instance cannot be accepted for execution. The QoS demand of each stream is captured in a Service Level Agreement (SLA)—which must be pre-agreed between the stream owner/ generator and the service provider (hosting the analysis capability) a priori. Such an SLA identifies the cost that a user must pay to achieve the required QoS and a penalty that must be paid to the user if the QoS cannot be met [5].

Assuming the maximization of profit as the main Cloud provider's objective, then it must be decided which streams to accept for storage and analysis; and how many resources to allocate to each stream in order to improve its overall profit. This task is highly challenging with aggregated, unpredictable and bursty data flows that usually make both predictive and simple reactive approaches unsuitable. Even dynamic provisioning of resources may not be useful to provide a profit to the Cloud provider since the delay incurred might be too high—it may take several seconds to add new resources (e.g. instantiate new Virtual Machines (VMs)), and a scaling-up action might generate substantial penalties and overheads.

Our main contributions consist of data admission and control policies to regulate data access and manage the impact of data bursts, and a policy for resource redistribution that tries to minimize the cost of QoS penalty violation, maximizing the overall profit. The rationale behind this latter policy is that current mechanisms for scaling resources in Cloud infrastructures have severe associated delays which may provoke large financial penalties. Overall, our main contributions can be summarized as follows: (i) an improved profit model that takes into account both profit and penalties, (ii) a set of dynamic control actions to manage resources with maximization of a provider's profit, (iii) a unified token-based resource management model for realizing the profit-oriented actions. This model aims at optimizing the utilization of unused resources, enabling dynamic and consistent re-allocation of resources, (iv) the specification of all the control logic in terms of a Reference-net model, (v) extensive simulations of various scenarios demonstrating the effectiveness of our proposed profit-oriented control mechanism, and (vi) an OpenNebula-based deployment showing how the Reference-net model can be turned into an executable model in a straightforward manner.

Our previous contributions in enforcing QoS on shared Cloud infrastructures were described in [6–8]. In [9,10], we proposed a profit-based resource management model for streaming applications over shared Clouds. In [11] we extend this with an improved revenue generation model and identify specific actions to support resource management. In particular, with (i) the re-distribution of unused resources amongst data streams; and (ii) a dynamic re-allocation of resources to streams likely to generate greater revenue for the Cloud provider. This paper extends [10,11], by combining our previous profit-based resource management model with an OpenNebula-based Cloud deployment. We provide a model in terms of Reference nets—particular type of Petri nets. One of the characteristics of Reference nets is that they can also be interpreted and support Java actions in their transitions, so that the models proposed here become executable directly.

The remaining part of this paper is structured as follows. Section 2 presents the revenue-based model for in-transit analysis and the profit-oriented actions to manage resources with maximization of provider's profit. Section 3 describes our system architecture based on the token bucket model, the rule-based SLA management of QoS and the unified token-based resource management model for realizing the profit-oriented actions by optimizing the utilization of unused resources and allowing dynamic and consistent re-allocation of resources. Section 4 describes the Reference net model of the control logic used. Section 5 shows our evaluation scenarios and simulation results. Section 6 presents our deployment and experiments on an OpenNebula-based Cloud infrastructure. In Section 7, most closely related work is discussed. Finally, the conclusions and future work are given in Section 8.

2. Profit-based resource management

2.1. Profit-based model

We consider a provider centric view of costs incurred to provide data stream processing services over a number of available computational resources. If we assume the objective of the provider is to maximize revenue, then it must decide: (i) which user streams to accept for storage and analysis; (ii) how many resources (including storage space and computational capacity) to allocate to each stream in order to improve overall profit revenue (generally over a time horizon); and (iii) what actions could be performed to dynamically modify and adjust the usage of resources. The first two considerations can generally be based on the SLA that a user and a provider have agreed to while the last point could be considered internal to the provider as a way to optimize resource utilization.

A provider may use a (pre-agreed and reserved) posted price, a spot price (to gain profit from currently unused capacity), or an on-demand use (the most costly for the user) of resources, on a per-unit-time basis—as currently undertaken by `Amazon.com` in their EC2 and S3 services. In the case of data stream processing services, this cost may also be negotiated between the user and the provider using QoS criteria. How such a price is set is not the focus of this work, our primary interest is in identifying what are the performance objectives that can be established in an SLA, and what actions the provider can perform to guarantee the agreed QoS and maximize the profit. A key distinction between batch-based execution on a Cloud infrastructure is that the query/computation and data are generally available before the execution commences. In a streamed application, a query is often executed continuously on dynamically available data. An SLA is therefore essential to identify what a user must pay the provider, often based on a previous estimation of resources required/used. Conversely, the provider can also utilize previously similar stream processing capability to identify resources required and any penalties paid in the past (for service degradation that violated the SLA). Due to the greater potential variation likely to be seen in stream processing applications, an SLA therefore protects both the user and the provider.

Defining QoS properties in an SLA is very application dependent. In applications such as commercial Web hosting, QoS levels specify parameters such as request rate, for example expressed as served URLs per period or the number of concurrent users served; and data bandwidth, that specifies the aggregate bandwidth in bytes per second to be allocated in the contract [12]. In other applications such as video-on-demand, QoS levels may represent frame rates and average frame sizes. In the context of a data stream, the