On Developing and Operating of Data Elasticity Management Process

Tien-Dung Nguyen, Hong-Linh Truong*, Georgiana Copil, Duc-Hung Le, Daniel Moldovan, and Schahram Dustdar

Distributed Systems Group, TU Wien, Austria
`{d.nguyen,truong,e.copil,d.le,d.moldovan,dustdar}@dsg.tuwien.ac.at`

Abstract. The Data-as-a-Service (DaaS) model enables data analytics providers to provision and offer data assets to their consumers. To achieve quality of results for the data assets, we need to enable DaaS elasticity by trading off quality and cost of resource usage. However, most of the current work on DaaS is focused on infrastructure elasticity, such as scaling in/out data nodes and virtual machines based on performance and usage, without considering the data assets' quality of results. In this paper, we introduce an elastic data asset model for provisioning data enriched with quality of results. Based on this model, we present techniques to generate and operate data elasticity management process that is used to monitor, evaluate and enforce expected quality of results. We develop a runtime system to guarantee the quality of resulting data assets provisioned on-demand. We present several experiments to demonstrate the usefulness of our proposed techniques.

1 Introduction

To provide flexible access to vast amounts of data, several types of Data-as-a-Service (DaaS) have emerged to allow users to execute data analytics atop a vast, rich set of data sources, such as Azure's Data Market[1], Infochimps[2], Factual[3], and a number of research systems [1–3]. Many of these systems support the concept of elasticity by scaling data nodes, re-assigning data partitions and reconfiguring data clusters to automatically adapt to dynamic changes in their workload [1–3]. Overall, these systems support the elasticity of data services at the infrastructure level. The question of how to ensure the quality of resulting analytics, covering quality of data, prices, analytics time, and forms of outputs for data analytics, has not been in the focus of existing elastic data systems.

To solve this problem, we examine another aspect of elasticity in data analytics within DaaS by considering how we could provide data assets resulting from data analytics in an elastic manner. That is, data consumers must be able

* Corresponding author
[1] http://datamarket.azure.com
[2] http://www.infochimps.com
[3] http://factual.com

to request, obtain and utilize data assets with different quality and cost based on their requirements. For example, if the quality of data associated with the resulting data assets is high, the cost for the resulting data assets might be increased. To achieve this in an elastic manner, besides dealing with the elasticity of computing resources for data processing, the DaaS provider must have mechanisms to deal with the elasticity of quality of data and cost.

In our approach, a data provider can provision a data asset based on quality of results (QoRs) by utilizing DaaS. Within DaaS there are different data analytics functions, which can be represented as a workflow of data analytics tasks, produces a data asset. A QoR model [4] can be used to specify, e.g., the quality of the data asset, the output form of the data asset, the time to deliver the data asset, and the price of the data asset. To ensure the QoR of the data asset, the DaaS provider needs to deploy, control, monitor not only its underlying computing systems (i.e., virtual machine), but also the data used to offer data assets. We refer to process having the above-mentioned features as data elasticity management process (DEP).

In this paper, we introduce a novel model of elastic data asset to associate the data asset with its QoR. Elastic data asset can have states, w.r.t. quality and cost, that can be changed over time to reflect the QoR in the interest of the data consumer. Based on that, we present techniques to support generating DEP to ensure QoR for data asset and the runtime environment to operate DEP for DaaS. In this paper, we illustrate our approach through real-world applications of data assets provisioning near-real time data.

The rest of this paper is organized as follows: Section 2 presents the motivation for our work. Section 3 introduces the model for elastic data assets. Section 4 presents techniques to generate data elasticity management process and its runtime. Section 5 presents experiments. Section 6 discusses related work. Finally, we conclude the paper and discuss our future work in Section 7.

2 Motivation and Approach

Let us consider a scenario with a GPS DaaS provider and several DaaS consumers, such as public transportation and taxi companies. They want to sell and buy near-real time GPS data of vehicles in the HoChiMinh City. The provider owns this data source and wants to sell potential data assets to DaaS consumers, trading off (i) accuracy of moving-vehicle – abbreviated by $vehicleAcr$ – the percentage of on-street vehicles over the total number of vehicles, (ii) accuracy of speed of vehicles – $speedAcr$ – the percentage of vehicles have the difference of measured speed and estimated speed higher than a threshold, $deliveryTime$ – the minimum time to deliver data asset in seconds, and cost. The DaaS provider wants to sell data assets in a flexible way based on a QoR associated with the data assets. For example, a data asset is sold with a QoR $=\{vehicleAcr \geq 81\%$, $speedAcr \geq 81\%$, $deliveryTime \leq 55(seconds)$, and cost per a window of data asset €0.09$\}$. Furthermore, a customer might accept lowered values of $vehicleAcr$ and $speedAcr$ and a higher $deliveryTime$, if the price is reduced.

Currently, it is challenging to build such a DaaS that can fulfill the above requirements from both the provider and customer perspectives. First, the provider must provision data assets, which have business values, from suitable data analytics processes (e.g., enriching near-real time GPS data with estimated average speeds of vehicles in different areas). Second, the DaaS provider must define metrics w.r.t QoR for the data asset and for each type of data asset, the provider can choose many metrics associated with it. Third, the provider must develop DEP for monitoring and controlling quality of data, performance of DaaS and resource usage to ensure QoR. Finally, the provider must develop a suitable runtime environment for executing the DEP together with data analytics process.

One major issue that arises from the above scenario is that the DEP are tightly coupled to the metrics in specified and expected QoR, which makes the DEP very difficult to be extended with other QoR. Moreover, information of data analytics, which is used to provision data asset, must be considered when creating DEP. It is also difficult for the DaaS provider to modify the DEP, after the DaaS has already been deployed. We need methods to create DEP that can be reusable and extensible w.r.t. changes of the QoR metrics associated with data asset. However, current models of data services [1] [2] [5] [3] are not yet associated with QoR that can support the above-mentioned requirements. These challenges motivate us to develop a technique to support generating DEP. The goal is to support the DaaS provider easily provisioning data asset from data analytics and QoR. Based on these inputs, we can support to generate DEP to achieve the elasticity of the provisioned data assets.

3 Elasticity model for data assets

3.1 Data assets and their quality of results

A provider might utilize different data sources to offer data assets. We assume that a data item can be a record in a relational database, a document in document-based database or a key-value pair in key-valued database. An analytics of a set of data items will produce a data asset. In our work, we support to provision data assets from batch and near real time data analytics; a near real time data asset is delivered to customers with a predefined time window. A window of data asset also includes a set of data items. Let DAF be a set of data analytics functions defined based on a data model for data sources. Such function can be simple, e.g., including several data queries, or complex, e.g., including different data analysis algorithms. For provisioning data assets from the data sources, each data asset can be produced by a data analytics function (DAF) $dataAsset = \{daf, daf \in DAF\}$. The DaaS provider can provision data assets they want to sell by defining and executing the DAF. For example, a DAF may include activities for (i) reading streaming GPS data, (ii) clustering vehicle location, (iii) estimating the average speed of vehicles in each cluster, and (iv) outputting data in the form of comma-separated values (csv).

Data assets offered to data consumers can be characterized by many metrics. These metrics are rich and dependent on data sources and DAF, such as data accuracy, data completeness and data consistency [6]. Moreover, the monetary value of the data assets may depend on the values of these quality metrics. To support the DaaS provider to present the expected values of these metrics and cost, we reuse the QoR model in [4, 7]. The QoR model includes a set of metrics, a set of QoR elements (qElement) and a form of data asset.

- A set of metrics measures quality of data assets. Each metric can be accessed/adjusted by primitive actions detailed in the next subsection.
- A qElement is defined as a set of conditions established based on these metrics in specific ranges and a specific cost for data assets.
- The form of data assets indicates the format of the output asset (e.g., comma-separated values or a bar chart) resulted from DAF.

We choose this QoR model to associate QoR metrics [4] to data asset because this model is easy for DaaS provider to present the expectation of quality and cost of data asset together with DAF.

3.2 Data elasticity management process

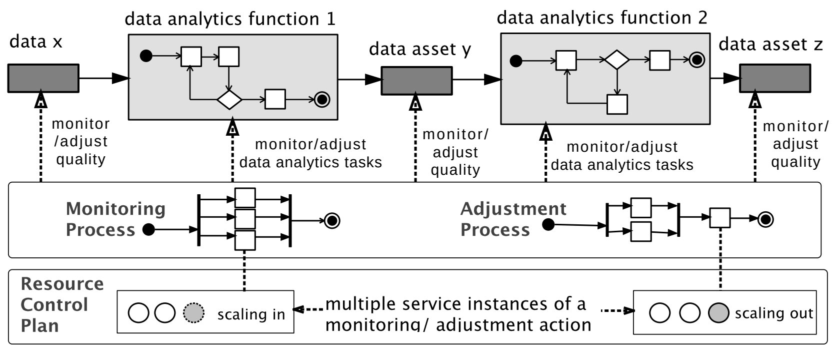

data analytics function 1
data x
data asset y
data analytics function 2
data asset z
monitor /adjust quality
monitor/adjust data analytics tasks
monitor/ adjust quality
monitor/adjust data analytics tasks
monitor/ adjust quality
Monitoring Process
Adjustment Process
Resource Control Plan
scaling in
multiple service instances of a monitoring/ adjustment action
scaling out

Fig. 1: Ensuring QoR for data assets by using data elasticity management process

Fig. 1 presents the relationship between data assets, DAFs and DEP. By executing the `data analytics function 1`, we have *data asset y*. The provider would like to sell *data asset y* to their customers. To ensure the QoR of this data asset, a data elasticity management process is invoked. A data elasticity management process includes sub-processes to monitor/control quality of data and performance (e.g., $deliveryTime$), and a resource control plan to manage resource usage at infrastructure level when executing these sub-processes.

There are different approaches to apply DEP: (i) improve the quality of *data x* which is used for determining *data asset y* from `data analytics function 1`, (ii) adjust *data analytics function 1* by tuning parameters, plugging sub-processes or replacing process fragments to improve the quality of *data asset y*

[4] https://github.com/tuwiendsg/EPICS/blob/master/depic/examples/qor/qor.yml

better, and/or (iii) improve the *data asset y* to produce a better data asset by separate processes. In this paper, we support the third method, through which we can ensure QoR of data asset produced by DAF. The general principle is that we store result data assets from DAF into a data buffer, then we use the sub-processes to ensure quality of data before delivering the data assets to customer. The sub-processes include many actions organized on workflows. We refer to these actions as primitive actions. A primitive action can be used to perform data assessment and quality adjustment, for example, applying regression algorithms [6] to smooth the values of vehicle speeds to increase data accuracy. QoR of data assets also depends on underlying computing infrastructure resources, e.g., delivery time and throughput of data asset. To adjust the values of these metrics, we can carry out primitive actions such as dynamic provisioning infrastructure resources to distribute workloads for many processing services [1]. For example, scaling out computational resources for data cleansing services can help to increase the throughput.

3.3 Managing Primitive Actions

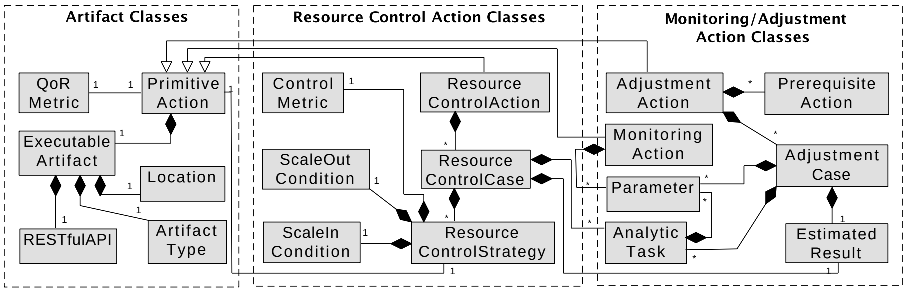


Fig. 2: Model of primitive action metadata

Fig. 2 shows the model for capturing primitive action metadata[5]. Primitive actions to monitor quality of data have to return the values of quality of data assets and can be executed in parallel, while primitive actions to adjust the quality of data do not return values and have to be executed in a particular order to avoid data corruption by overwriting and achieve semantic of adjustment processes. Therefore, a primitive action can be an adjustment action (i.e., an action to adjust the quality of data asset), a monitoring action (i.e., an action to assess quality of data asset) or a resource control action (i.e., an action to scale in/out monitoring/adjustment services at runtime). At runtime, a primitive action is executed through an invocation of an adjustment/monitoring service. A primitive action call will invoke its corresponding service. To deal with different situations of quality of a data asset, an adjustment action can have multiple

[5] An example of primitive action metadata can be found here https://github.com/tuwiendsg/EPICS/blob/master/depic/examples/pam/pam.yml

adjustment cases. Each adjustment case is specified by the parameters of its adjustment action. To determine the right adjustment case, we use information about expected QoR in PAM and analytic tasks in DAF. For example, an adjustment action, which removes missing-value records in a data asset to improve data completeness, has an estimated result 100%. For another example, an analytic task in DAF employs k-means with the stop condition after 5 loops that may lead to a bad data asset because the convergence does not approach global optimum [8]. Previous studies proposed methods to determine metadata for adjustment cases, for example, selecting input parameters for decision trees [9], or selecting input parameters for clustering data [8]. These studies can help to indicate right parameters from estimated QoR metrics and analytics tasks for adjustment cases. We assume that data about primitive action metadata results from these studies, which are out of the scope of our study. A prerequisite action is used to decide which adjustment actions are executed before another one. A resource control strategy has conditions for metrics at system level (e.g., cpuUsage) to scale in/out monitoring/adjustment services.

3.4 Elastic data asset

Considering a data asset (da) resulted from a DAF, we use a monitoring process to determine if the *da* is delivered with the expected qElement or not. If not, we apply an adjustment process to this *da* to create another *da* which will meet the expected qElement. To model the changes of QoR of the da by predetermined monitoring and adjustment processes, we need to model elasticity states (i.e., states w.r.t qElement within the QoR) associated with the *da* and DEP.

Eq. 1 defines an elastic state, which is a binary vector of conditions associated with each metric being monitored. For metric i, let em_i be the current evaluated metric value, while c_{j_i} gives condition j_i for metric i. This eState is evaluated to true if the conjunction over the $eState[j_i]$, considering the current metric values em_i, is evaluated to true. The set of all eStates associated with a data asset, ES_{da}, is given by the eStates obtained from the combination of all conditions available for all metrics specified in the QoR, as defined in Eq. 2.

$$\begin{aligned} eState_{da}[j_1 \dots j_n] = \{[c_{j_1}(em_1), \dots, c_{j_n}(em_n)]| \\ em_i \in EM_{qor}, c_{j_i} \in CD_{qor}\} \end{aligned} \tag{1}$$

$$\begin{aligned} ES_{da} = \{eState_{da}[j_1 \dots j_n]| \\ \forall [j_1 \dots j_n] \in combinations(j_x), \\ j_x \in NumberOfConditions(Metric_x, qor)\} \end{aligned} \tag{2}$$

The *eda*, defined in Eq. 3, is composed of *da*, which is obtained by applying a daf, the ES_{da} defined in Eq. 2, the set of data elasticity management processes DEP, and the initial and final eStates, $eState_{in}$ and $eState_{fi}$. The eStates are managed by a set of data elasticity management processes DEP.

$$eda = (da, ES, eState_{in} \in ES, DEP, eState_{fi} \in ES) \quad (3)$$

The data elasticity management processes (DEP) associated with an eda includes a monitoring process (p_m), adjustment processes (p_c) and resource control plan (p_r). A p_m is used to determine $eState_{in}$ of da, while p_c and p_r are used to do transition of the da from an $eState_{in}$ to an $eState_{fi}$ in ES:

- Monitoring process: let MA be the set of monitoring actions. Each monitoring action is mapped to a specific monitoring service to measure value of a specific QoR metric. Thus, a monitoring process of a da is defined as $p_m = \{ma(parameters)\}, ma \in MA$; $eState = p_m(da)$; and $parameters$ denotes parameters for the monitoring actions.
- Adjustment process: let CA be the set of adjustment actions. Each adjustment action is mapped to a specific adjustment service to adjust quality of a specific QoR metric. An adjustment process can be defined as $p_c = (\{ca(parameters)\})$, where $ca \in CA$
- Resource control plan: let RA be the set of resource control strategy. Each resource control strategy is used to control a metric at infrastructures to ensure the quality of a specific QoR metric. A resource control plan can be defined as $p_r = (\{ra\})$, where $ra \in RA$

4 Generating and operating data elasticity management processes

4.1 Generating data elasticity management processes

Based on the model of eda, our approach to generate DEP for data asset includes 3 steps: (i) generating a monitoring process to understand the quality of data assets, (ii) generating an adjustment process to adjust the quality of data assets, and (iii) generating a resource control plan during the execution of the processes.

Algorithm 1 Algorithm to generate data elasticity management processes

```
1: function GENERATE DATA ELASTICITY MANAGEMENT PROCESSES($qor, daf, pam$)
2:     $listOfMonitoringAction$ =findMonitoringActions($qor.metricList(), pam$)
3:     $monitoringProcess$ =parallelizeMonitoringActions($listOfMonitoringAction$)
4:     $finalEStateSet$ =decompose($qor.qElements(), pam$)
5:     for each $eState$ in $finalEStateSet$ do
6:         $adjustmentCases$ =findAdjustmentCases($eState, pam, daf$)
7:         $adjustmentProcess$ =buildWorkflow($adjustmentCases, pam$)
8:         $resourceControlPlan$=findControlStrategies($eState, pam, daf$)
9:         $elasticProcesses$ =
10: new $ElasticProcesses(eState, monitoringProcess, adjustmentProcess, resourceControlPlan)$
11:     end for
12: end function
```

Algorithm 1 describes the algorithm to generate DEP. The inputs for this algorithm include a QoR, a DAF and the primitive action metadata (PAM).

Based on the list of metrics in the QoR and the monitoring actions associated with these metrics in PAM, the algorithm generates a monitoring process by organizing the monitoring actions in parallel (line 2-3). The final *eState* set of a data asset is determined through decomposing ranges of values of QoR metrics of the *qElements* into conditions of estimated results of corresponding metrics (line 4). For each *eState* in the final *eState* set, the algorithm finds primitive actions based on its corresponding QoR metrics of conditions in the *eState*. If the primitive action is a type of adjustment action, the algorithm finds adjustment cases by matching (i)the conditions in the *eState* with estimated results in PAM and (ii) analytic tasks in the DAF with the ones in PAM (line 6). The adjustment cases are found if both (i) and (ii) are matched. An adjustment process is built from the adjustment action list and prerequisite actions in PAM (line 7). The algorithm creates sub-workflows of the adjustment actions by connecting the prerequisite actions in sequence. Then, these sub-workflows are connected by parallel gateways. If the primitive action is a type of resource control action, the algorithm finds resource control strategies by matching (iii)the conditions in the eState with estimated results in PAM and (iv) analytic tasks in the DAF with the ones in PAM (line 8). The resource control strategies are found and added to the resource control plan if both (iii) and (iv) are matched. The resource control plan is used to create SYBL control strategies[10] for scaling in/out computing resources for data elasticity operations.

4.2 Runtime for Data Elasticity Management Process

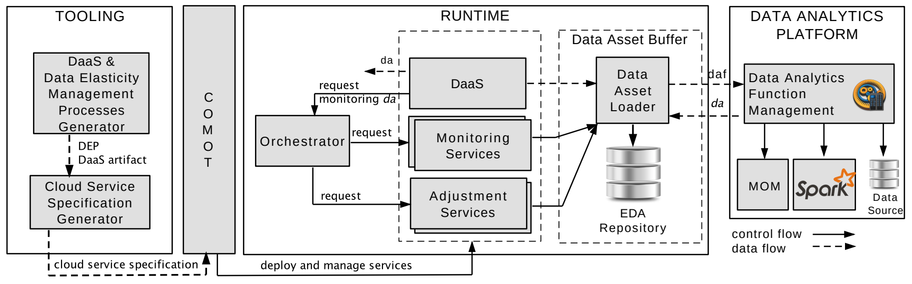


Fig. 3: The architecture of the runtime of data elasticity management processes

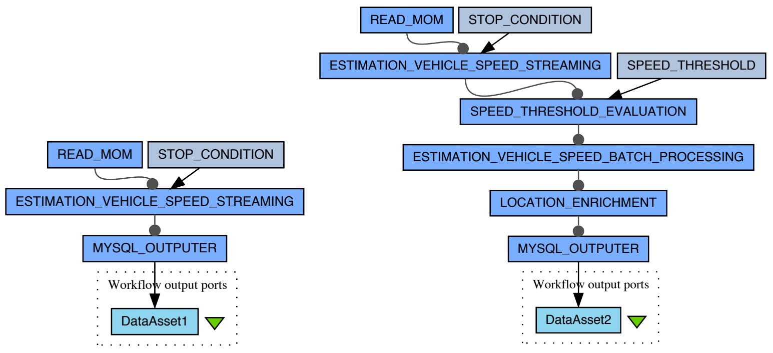
To support the generation and operation of DEP, we have implemented the *Tooling* and the *Runtime*, as shown in Fig. 3. *Tooling* allows the DaaS provider to define DAF and QoR used for generating DEP and to tune parameters' values of monitoring/control actions in DEP. The *Cloud Service Specification Generator* utilizes COMOT services [11] API to determine the non-existing monitoring and adjustment services in the cloud infrastructure and generate a cloud service specification describing the deployment of DEP and other runtime services used for QoR enforcement, such as *Monitoring/Adjustment Services*, *Data Asset Loader* and *EDA Repository*, and resource control plan. In our runtime, the *Data*

Asset Loader is responsible for getting data assets from data analytics functions and storing them into the *EDA Repository* from which data assets will be passed to enrichment actions. The *Orchestrator* executes monitoring process, handling validation and applying appropriate adjustment process. In our prototype[6], we interface to different *Data Analytics Platforms*. These platforms are responsible for processing DAF and returning unqualified data assets.

5 Evaluation

5.1 Experiment settings

We use the scenario described in Section 2 with near-real time GPS data of vehicles in the HoChiMinh City. We obtained this 1.17GB real data from our research collaborators and emulated real-time data sources by sending historical GPS data to scalable message oriented middleware (MOM) located in the same cloud infrastructure with our runtime services. Fig. 4 shows two experimental DAFs used to provision data assets. $daf1$ gets near-real time GPS data from MOM, clustering locations of vehicles based their latitude and longitude - window size of 5000 data items, and estimating vehicle speed in each cluster. $daf2$ is an extension of the $daf1$ that checks if the current estimate speed is over a threshold, historical GPS data will be used to estimate the average vehicle speed, enrich data with address and output data. The DaaS consumers might be interested in using the data asset 1 to detect potential traffic congestion and the $DataAsset2$ to find causes of traffic congestion.



(a) daf1 (b) daf2

Fig. 4: Data Analytics functions for provisioning GPS data

We have two phases in provisioning data assets. First, we deployed DAFs which continuously deliver initial data assets to the Data Asset Buffer; the execution of DAFs represents a streaming data analytics system. Second, when a

[6] Available at: https://github.com/tuwiendsg/EPICS/tree/master/depic

consumer requests a data asset through a DaaS, the corresponding generated DEP will take streaming initial data assets in the runtime *Data Asset Buffer* and then apply monitoring and quality enrichment before returning suitable data assets to the consumer. In the first phase, all customers should share the operation cost, whereas in the second case, each customer pay only the cost due to the delivery of its data assets.

The QoR of the data assets are measured by the following metrics: *speedAcr* – accuracy of speed of vehicles, i.e., the percentage of vehicles have the squared deviations of speed higher than a threshold, *vehicleAcr* – accuracy of moving-vehicle, i.e., the percentage of on-street vehicles over the total number of vehicles, *throughput* – the average number of data assets delivered per second, abbreviated as das/s, and *deliveryTime* – the minimum time (in seconds) to deliver data asset in the second phase.

For primitive action metadata (PAM), we assume that domain experts use the model $primitiveAction = \{qoRMetric, estimatedResult, analyticTask\}$ to fill in right primitive action for specific QoR metric, estimated result and analytic task. Depending on different estimated results and analytic tasks, different primitive actions and their parameters are used.

5.2 Generating data elasticity management processes

We use different QoRs, DAFs and primitive action metadata to evaluate our proposed generation technique, as shown in Table 1. We evaluate the generated DEP using completeness. The completeness has value yes in case the DEP have complete monitoring process, adjustment processes and resource control plans to ensure QoR of the data asset, and value no in case the DEP are unable to ensure QoR of the data asset because at least one of monitoring process, adjustment processes and resource control plans has missing or conflict data.

Table 2 summarizes the results of generating DEP in 5 cases:

– case 1: Because the conditions in the qElement set match the estimated results in the primitive action metadata *pam*1 completely, and *pam*1 has complete information of primitive actions, the generated DEP are complete.
– case 2: We use the *qElement*2, which has a condition of *vehicleAcr* is [91,95]. This condition is a subset of *estimatedResult* [81,100]. Therefore, the estimated result in ranges [81,90] and [96,100] cannot be satisfied and the generated adjustment process is not complete.
– case 3: We use $daf2$, which has stopCondition of K-means is different from the ones in the adjustment cases, leading to the incompleteness of the generated resource control plan.
– case 4: *pam*2 has an adjustment action for *vehicleAcr*, however, the estimated result is missing. So that the adjustment processes are not complete.
– case 5: The resource control plans of *throughput* and *deliveryTime* are conflicted because their resource control strategies control the same monitoring/adjustment services at runtime. Therefore, we have to choose another resource control plan to deal with these metrics manually.

QoR	#	qElement	[speedAcr(%)];[vehicleAcr(%)];[deliveryTime(s)]; [throughput(das/s)];cost(€)
	qor1	qElement1	[81,100];[81,100];[0,55];[];[0.007]
		qElement2	[81,100];[81,100];[56,∞];[];[0.006]
		qElement3	[61,80];[61,80];[0,55];[];[0.006]
		qElement4	[41,60];[41,60];[0,55];[];[0.005]
		qElement5	[21,40];[21,40];[0,55];[];[0.004]
	qor2	qElement1	[81,100];[91,95];[0,55];[];[0.007]
		qElement2	[81,100];[81,100];[56,∞];[];[0.006]
		qElement3	[61,80];[61,80];[0,55];[];[0.006]
		qElement4	[41,60];[41,60];[0,55];[];[0.005]
		qElement5	[21,40];[21,40];[0,55];[];[0.004]
	qor3	qElement1	[81,100];[81,100];[0,55];[1.05,∞];[0.007]
		qElement2	[81,100];[81,100];[56,∞];[0,1.04];[0.006]
		qElement3	[61,80];[61,80];[0,55];[1.05,∞];[0.006]
		qElement4	[41,60];[41,60];[0,55];[1.05,∞];[0.005]
		qElement5	[21,40];[21,40];[0,55];[1.05,∞];[0.004]
PAM	#	primitive action	[associatedQoRMetrics];[estimatedResult]; [analyticsTask - parameter:value]
	pam1	adjustmentAction1	[speedAcr];[81,100];[kmeans - stopCondition:5]
		adjustmentAction2	[vehicleAcr];[81,100];[]
		resourceControlAction1	[deliveryTime];[0,55];[]
		resourceControlAction2	[throughput];[1.05,∞];[]
	pam2	adjustmentAction1	[speedAcr];[81,100];[kmeans - stopCondition:5]
		adjustmentAction2	[vehicleAcr];[];[]
		resourceControlAction1	[deliveryTime];[0,55];[]
		resourceControlAction2	[throughput];[1.05,∞];[]
DAF	#	DAF task	[analyticsTask - parameter:value]
	daf1	estimation vehicle speed	[kmeans - stopCondition:5]
	daf2	estimation vehicle speed	[kmeans - stopCondition:10]

Table 1: Summary of QoR, primitive action metadata and DAF

	case 1	**case 2**	**case 3**	**case 4**	**case 5**
QoR	qor1	qor2	qor1	qor1	qor3
Primitive Action Metadata	pam1	pam1	pam1	pam2	pam1
Data Analytics Function	daf1	daf1	daf2	daf1	daf1
Completeness of the Processes	Yes	No	No	No	No

Table 2: Completeness of DEP in different cases

We can see that our techniques could generate complete DEP when information of primitive actions are complete. In case of missing information of primitive actions or conflicts in DEP, the DEP can be customized manually. In future, we will develop an algorithm to automatically resolve these conflicts.

5.3 Operating data elasticity management processes

We evaluate the generated DEP in `case 1` at runtime. In the evaluation, we show 3 aspects of elasticity including resource (i.e., virtual machine), quality (i.e., vehicleAcr, speedAcr and deliveryTime) and cost (i.e., processing cost and data asset cost). We use 1 VM (3GB RAM,2 vCPUs, 40GB Disk) for *COMOT* services. We use 1 *m1.small* VM (1GB RAM, 1 vCPU, 40GB) for MOM. We use 1 VM (7GB RAM, 4 vCPUs, 40GB Disk) for *Tooling*, *Orchestrator*, *Data Asset Loader* and *Data Analytics Function Management*. We use 4 VMs for monitoring/adjustment services at the beginning. Each monitoring/adjustment service runs on 1 *m1.small* VM. We test the execution of the generated DEP with 5 concurrent DaaS consumers in case of using consumerRequirement1 $=\{vehicleAcr \geq 81\%$, $speedAcr \geq 81\%$, and $deliveryTime \leq 55(s)\}$ and consumerRequirement2 $=\{vehicleAcr \geq 61\%$, $speedAcr \geq 61\%$, and $deliveryTime \leq 55(s)\}$. To study cost elasticity, we defined the data asset cost as follows:

$$cost_{da} = \sum_{i=1}^{nbMetrics} unitCost(qorMetric_i) * w_i * (\sum_{j=1}^{nbCond_i} (j * qElement_{cond_j})) \quad (4)$$

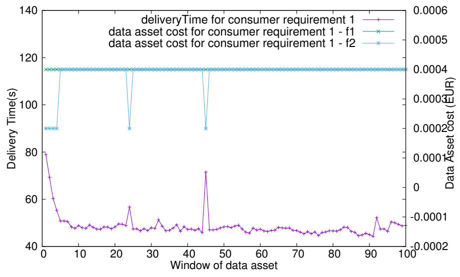
$qElement_{cond_j}$ is a boolean condition associated with $qorMetric_i$, and w_i are weighted factors. For each $qorMetric_i$ in a qElement, only one $qElement_{cond_j}$ has value 1 and the others have value 0. The data asset cost depends on the values of *vehicleAcr* and *speedAcr*, which are divided into 5 ranges to present quality of data from low to high. This cost also depends on the processing time, including *analyticTime* in the first phase and *deliveryTime* in the second phase. From this general function, we have 5 functions for data asset cost assumptions - f1 and f2 ($w_{processingTime}$=1, $w_{vechicleAcr}$=0, $w_{speedAcr}$=0), f3 and f4 ($w_{processingTime}$=0, $w_{vechicleAcr}$=0, $w_{speedAcr}$=1) and f5 ($w_{processingTime}$=0.5, $w_{vechicleAcr}$=0.25, $w_{speedAcr}$=0.25). We use the unit cost for *speedAcr* and *vehicleAcr* as €0.0002, the unit cost for machines in the data analytics phase as 0.104 EUR (the same as the cost of t2.large instance [7]), and the unit cost for machines in the enrichment phase as €0.026 (the same as the cost of t2.small instance).

Fig. 5 shows *deliveryTime* of data asset with different consumer requirements. *deliveryTime* is controlled under the range [0,55] from the 46th and 14th window of data in case of using `consumerRequirement1` and `consumerRequirement2`, respectively. Fig. 6(a) shows the values of *speedAcr* in cases of using and not using the DEP: the values of *speedAcr* are always in ranges [81,100] in case of using the DEP but not in the case of not using the DEP.

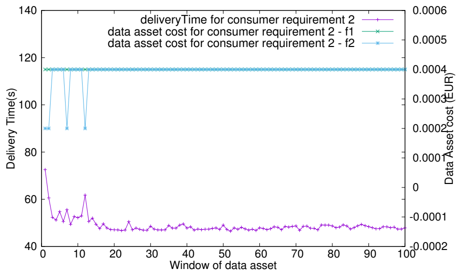
Fig. 5 and Fig. 6(a) show data asset cost defined by f1, f2, f3 and f4. We see the relationships between the values deliveryTime/speedAcr and data asset cost. When the values of QoR do not meet expected values, the data asset cost needs to be lowered. Fig. 6(b) shows the relationship between processing cost and data asset cost from f5 in cases of using `customerRequirement1` and `customerRequirement2`. The processing cost for an data asset is the sum of the quality adjustment cost and data analytic cost. The quality adjustment cost is calculated by multiplication of *deliveryTime*, the number of used VMs and the unit cost of t2.small instance. The total data analytic cost is calculated by

[7] http://aws.amazon.com/ec2/pricing/

the multiplication of analytic time, the number of used VMs and a unit cost of t2.large instance so we assume the data analytic cost per data asset equals to the total data analytic cost divided by the number of consumers. We see that the processing cost at the beginning is high because *deliverytime* is too long. Then, the processing cost increases (i.e., from the 3th window of data asset to the 35th window of data asset) because the adjustment/monitoring services scale out continuously (i.e., more VMs are used). After that, the cost of VMs remains unchanged because *deliveryTime* is stable and there is no need of scaling in/out actions. Fig. 6(b) shows that, with f5 for consumer requirement 2, compared with consumer requirement 1, the processing cost has a higher degree of fluctuations than that of data asset cost, suggesting in certain situations the provider spends more operational costs. We cannot conclude which is the best function for data asset cost, but support the provider to decide an appropriate cost functions.

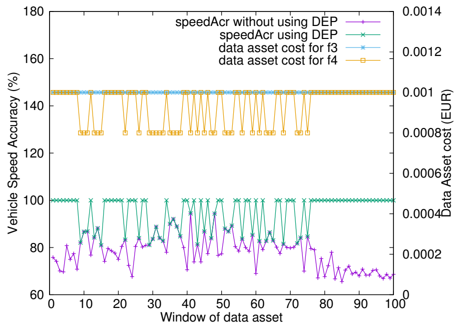


(a) customer requirement 1

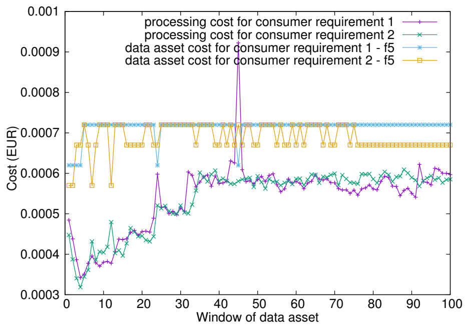


(b) customer requirement 2

Fig. 5: deliveryTime and data asset cost



(a) speedAcr with and without DEP



(b) Processing and data asset cost

Fig. 6: speedAcr, processing cost and data asset cost

6 Related Work

Elasticity Model of Data Provision: Existing studies have been introduced to support certain elasticity models for data provisioning. Examples are ElasTras which supports dynamic partition reassignment [1], functional requirement changes using a database schema evolution rule [2], reconfiguration of data cluster responding to workload changes [5], and changes of mapping of identifiers to storage processes under heavy load [3]. In general, these works support elastic data distribution by dynamically adding/removing data nodes and moving data partitions between data nodes. Our research objective is different in the sense that our data model is introduced to abstract data asset and DEP to ensure QoR.

Data elasticity management processes for quality of analytics: Although there exist techniques to enable DEP [7, 12, 13], they do not support runtime QoR-aware delivery and runtime environment for DEP. Our previous Drain supports selecting and configuring process fragments to ensure QoR of data [7]. However, it does not consider factors which impacts on the QoR of data from analytics functions and does not address resource controls at runtime; it also does not generate DEP for DaaS. Hauder [14] presented a framework to generate data processes from an abstract process, but this approach uses only one criterion (valid/invalid) to evaluate generated process and do not consider QoR of data and elasticity at runtime. Liu et al. presented the requirement for data service composition as nested tables [12] and proposed an algorithm for data service composition based on links and weights of links between actions. Wang et al. proposed a model to specify a pre-defined data process and requirements of quality of services as well as cost and presented an improved version of a genetic algorithm to select data-intensive service when generating service composition [13]. This algorithm optimizes the service selection process based quality of services and cost (e.g., data cost, access cost, and transfer cost). Different from existing approaches, we use DAF to provision a data asset and a QoR model to present requirements of this data asset. We develop a technique to generate DEP to monitor, adjust quality of data asset and control resource at runtime. Moreover, previous studies have not investigated elasticity when generating service composition, while we consider elasticity at runtime.

7 Conclusions and Future Work

Data assets produced from data analytics functions should be provided based on different quality of results in an elastic manner. In this paper, we show that, with appropriate knowledge about primitive actions for monitoring, adjustment and resource control, we can support the provider to generate DEP as well as can leverage underlying elastic platforms to manage and operate DEP to provision QoR-aware data assets. We also support the provider to study data asset cost functions based on QoR and cost of resource usage.

We are currently testing several experiments of generated DEP with many other types of data. We are working on optimizing the execution of data analytics functions. Moreover, we will develop a programming framework to support the DaaS provider to easily build services for primitive actions.

Acknowledgment: This work is supported by the European Commission in terms of the CELAR FP7 project (FP7-ICT-2011-8 #317790).

References

1. Das, S., Agrawal, D., El Abbadi, A.: Elastras: An elastic, scalable, and self-managing transactional database for the cloud. ACM Trans. Database Syst. **38** (2013) 5:1–5:45
2. Ishida, Y.: Scalable variability management for enterprise applications with data model driven development. In: International Software Product Line Conference Co-located Workshops. SPLC, New York, NY, USA, ACM (2013) 90–93
3. Unterbrunner, P., Alonso, G., Kossmann, D.: High availability, elasticity, and strong consistency for massively parallel scans over relational data. The VLDB Journal (2013) 1–26
4. Truong, H.L., Dustdar, S.: Principles of software-defined elastic systems for big data analytics. In: International Conference on Cloud Engineering. IC2E, Washington, DC, USA, IEEE Computer Society (2014) 562–567
5. Cruz, F., Maia, F., Matos, M., Oliveira, R., Paulo, J.a., Pereira, J., Vilaça, R.: Met: Workload aware elasticity for nosql. In: European Conference on Computer Systems. EuroSys, New York, NY, USA, ACM (2013) 183–196
6. Batini, C., Cappiello, C., Francalanci, C., Maurino, A.: Methodologies for data quality assessment and improvement. ACM Comput. Surv. **41** (2009) 16:1–16:52
7. Murguzur, A., Schleicher, J., Truong, H.L., Trujillo, S., Dustdar, S.: Drain: An engine for quality-of-result driven process-based data analytics. In Sadiq, S., Soffer, P., Vlzer, H., eds.: Business Process Management. Volume 8659 of Lecture Notes in Computer Science. Springer International Publishing (2014) 349–356
8. Guo, J.J., Luh, P.: Selecting input factors for clusters of gaussian radial basis function networks to improve market clearing price prediction. Power Systems, IEEE Transactions on **18** (2003) 665–672
9. D'heygere, T., Goethals, P.L., Pauw, N.D.: Use of genetic algorithms to select input variables in decision tree models for the prediction of benthic macroinvertebrates. Ecological Modelling **160** (2003) 291 – 300 Modelling the structure of acquatic communities: concepts, methods and problems.
10. Copil, G., Moldovan, D., Truong, H.L., Dustdar, S.: Sybl: an extensible language for controlling elasticity in cloud applications. In: Cluster, Cloud and Grid Computing (CCGrid), 13th IEEE/ACM International Symposium on, IEEE (2013) 112–119
11. Truong, H.L., Dustdar, S., Copil, G., Gambi, A., Hummer, W., Le, D.H., Moldovan, D.: Comot–a platform-as-a-service for elasticity in the cloud. In: IEEE International Workshop on the Future of PaaS, colocated with IC2E, IEEE (2014)
12. Liu, C., Wang, J., Han, Y.: Situation-aware data service composition based on service hyperlinks. In Huang, Z., Liu, C., He, J., Huang, G., eds.: Web Information Systems Engineering WISE 2013 Workshops. Volume 8182 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2014) 153–167
13. Wang, L., Shen, J., Luo, J., Dong, F.: An improved genetic algorithm for cost-effective data-intensive service composition. In: Semantics, Knowledge and Grids (SKG), 2013 Ninth International Conference on. (2013) 105–112
14. Hauder, M., Gil, Y., Liu, Y.: A framework for efficient data analytics through automatic configuration and customization of scientific workflows. In: IEEE 7th International Conference on E-Science, e-Science 2011, Stockholm, Sweden, December 5-8, 2011. (2011) 379–386