A Big data Processing service mesh model for the on-the-fly creation of analysis and deep learning solutions in the cloud

undefined[a]

[a]CINVESTAV, Tamaulipas, The Netherlands

ARTICLE INFO

Keywords:
Service mesh
Big data
Data analysis

ABSTRACT

The term 'As a Service' has taken on great relevance over the years in different computing areas, and as is to be expected, data analysis as a service is no exception. The creation of solutions based on applications available as a service has become a keystone for the realization of big data analysis, allowing users to process large volumes of data using multiple computing resources in a distributed manner. Currently, the creation of solutions, applications, or services in infrastructures such as the cloud presents a challenge, not only due to problems that arise in a distributed environment but also due to the current orientation towards the infrastructure of current works. In this work, we present a model based on a service mesh for the creation of solutions for big data analysis on demand and agnostic to the infrastructure.

1. Introduction

The term 'As a Service' has acquired great relevance over the years in different areas of computing, an example of this can be data analytics as a service. Currently, the presence of online tools for data processing, transformation and visualization is increasingly common, in such a way that the end user does not have to worry about details such as infrastructure, computing resources, installation of dependencies, among other tasks related to preparing the computing environment for data processing.

In this context, end users only need to worry about how to perform data processing using available tools. However, this suggests an almost total delegation of control to a cloud service provider, that is, control of the data and how it is transported and stored in the hands of a third party.

On the other hand, the use of only the tools offered by the cloud service provider can be limiting when processing data. Although there may be a great variety of services that allow the transformation of multiple data sources (structured or unstructured), it is possible that users may need more specific applications, either those that are available to the community or well developed by themselves.

Online repositories following FAIR principles (Findable, Accessible, Interoperable, and Reusable) [**?**] [**?**] have been created for collecting and making available visualization and analysis software as well as datasets and information for scientific community to establish collaborative work by sharing these tools with end-users [**?**] [**?**]. These repositories not only represent a source of solutions for organizations to produce useful information for decision makers [**?**], but also an opportunity area to create big data services.

Based on the foregoing, it makes sense that users may require more specific applications, or those that are in public repositories to perform data processing, and that can also be used as a service in a distributed computing infrastructure such as the cloud. .

**Corresponding author
ORCID(s):

This paper presents a model for building deep learning and analytics solutions in the cloud on the fly. The model makes use of building blocks for the encapsulation of applications and their deployment as a service. Each of the building blocks are part of a mesh, allowing interconnections between the multiple services in order to build customized solutions. The service mesh allows the inclusion of new services, allowing users to create their own from existing applications, whether they are desktop applications or web services.

2. Related work

- Dislib [2]. Is a distributed machine learning library wrote in python on the top of PyCOMPS, a based programming model and run-time for scientific applications. Dislib can be suited in several infrastructures such as clusters or the cloud. PyCOMPS offers to dislib a transparent layer to run in a distributed environment agnostic to the infrastructure. Dislib offers machine learning algorithms as well as many other tools to process structures data in a distributed environment, being able to process data in parallel based in a divide-and-conquer approach. This approach lets dislib deal with large datasets, dividing it in subsets and processing in parallel.

- Apache NiFI [1]. Is an APACHE distribution focused on build and automate data flow between applications and systems. APACHE NiFI allows designing using a GUI a workflow describing the dataflow using existing applications and/or systems. In this sense, this tool performs the data transportation and execution of these apps automatically. NiFI exposes a set of applications to reach this goal, being able to access multiple data repositories, resources in the cloud, apps for data transformations, etc. Nevertheless, does not focus on the offer the experience as a service, this means that NIFI does not worry about the applications used, this

why do not offer any other application to process data, as well as the apps used on the dataflow can not be accessed by other solutions created by APACHE NiFi, since every dataflow created is a black box.

- Weka
- Tensorflow
- Amazon web services

3. Design

This section describes the dynamic coupling model of services based on building blocks. This model considers the following components for creating and managing processing structures:

- **Building blocks**. A building block is a standard building unit, giving applications features for coupling in conjunction with other building blocks. Likewise, encapsulation provides the applications of different features:
  - It gives applications a self-contained runtime environment.
  - It allows application execution to be agnostic to the infrastructure.
  - It provides the ability to deploy applications as a service.
  - Building blocks are replicable units.
- **Black boxes**. It is the abstraction of an application, service or transformation task to an ETL processing model.
- **API Gateway (AG)**. It is a coordinating entity in charge of managing the multiple services within the mesh. The AG provides a consistent abstraction for end-user communication with existing services on the mesh.

The proposed model is based on a mesh of encapsulated services such as building blocks (bb_i), in this sense, the mesh is composed of a set of building blocks (BB) interconnected (E). This is represented as a complete graph of the form:

$$MESH = (BB, E) \tag{1}$$
$$BB = \{bb_1, bb_2, \dots, bb_n\} \tag{2}$$
$$E = \{e_1, e_2, \dots, e_m\} | m = n(n-1)/2 \tag{3}$$
$$e_i = (x, y) | BB \times BB \tag{4}$$

The tasks are chained through the use of their inputs and outputs, abstracting the process that is carried out as a black box and the abstraction of tasks is done through encapsulation in building blocks. In this sense, in a building block we have a set of black boxes ($BBOX$). For this work, a building block is defined as a tuple:

$$bb_i = (input_i, BBOX_i, output_i) \tag{5}$$

where:

- $BBOX_i$ is the set consisting of m black boxes, represented as $BBOX_i = \{bbox_1, bbox_2, \dots, bbox_m\}$
- $input_i$ resent the input data for the building block.
- $output_i$ represent the results produced by the building block.

The objective of the building block is to self-contain a set of applications and provide them with a generic interface capable of being managed. A building block can contain a set of m black boxes ($BBOX$). The term black box is used to refer to tasks or applications that follow the ETL model. Based on the above, the processing performed by an application becomes in the background, with only the input and output of data being important. Each black box $bbox_j$ is the abstraction of the traditional application which follows the ETL model. In this way we have these stages represented as a data input (E), a transformation process (T) and an output (L). However, the transformation process T is not the final link in this encapsulation, since the number of black boxes can scale horizontally (in $BBOX_i$ set) but also in depth, in this sense, T can be a application (app) or another set of black boxes ($BBOX_k$). This is represented as follows:

$$bbox_j \in BBOX_i \tag{6}$$
$$bbox_j = (E_j, T_j, L_j)| \tag{7}$$
$$T_j = app \vee (BBOX_k | k \neq i) \tag{8}$$

The selection of the black boxes for a set $BBOX_i$ is made based on the relation between the input data or the output data. We defined two kind of relations:

- *Parallel Relation* (PR). The input data of a $bbox_j$ black box and a $bbox_h$ black box are the same, however the outputs are different. For example, assume that $bbox_j$ and $bbox_h$ contain two different classification algorithms. In this context, both $bbox_j$ and $bbox_h$ can receive the same data set as input, however the results produced may not necessarily be the same. Likewise, the data produced by $bbox_j$ cannot be used as input to $bbox_h$ (assuming that the algorithms output is a model and not a data set). PR is represented as follows:

$$PR = (E_j = E_h) \wedge (L_j \neq L_h)| \tag{9}$$
$$(j \neq h) \wedge (bbox_j, bbox_h \in BBOX_i) \tag{10}$$

- *Sequential Relation* (SR). The input data for a black box $bbox_j$ and a black box $bbox_h$ are different, but the output of $bbox_j$ can be the input for $bbox_h$. For example, suppose that $bbox_j$ contains an outlier elimination algorithm and $bbox_h$ contains a classification

algorithm. In this context, the data can go through the $bbox_j$ cleanup process and then go through the $bbox_h$ classification process. In this way it is possible to chain different black boxes creating processing pipes within a building block (bb_i). SR is represented as follows:

$$SR = (L_j = E_h)| \tag{11}$$

$$(j \neq h) \wedge (bbox_j, bbox_h \in BBOX_i) \tag{12}$$

Based on these relationships described, the execution of a solution can vary and become more complex, being able to have processing pipes (SR) executed in parallel (PR) encapsulated in an instance (bb_i), which can be cloned and deployed on various computing resources.

Finally, a solution (SOL) can be built based on all the components. The solution consists of a directed graph formed by a subset of the mesh building blocks set (V). This is represented as follows:

$$SOL = (V, E) \tag{13}$$

$$V \subset BB \tag{14}$$

$$E = \{(x, y)|(x, y) \in V \times V\} \wedge x \neq y \tag{15}$$

The ETL approach is critical to developing a solution. A solution consists of data extraction, a transformation using a black box to finish with the presentation of the resulting transformation. In a recursive way, the components of the black box use the ETL approach, in such a way that for each task there is an ETL process that coordinates the manipulation of the data. The use of the ETL approach allows adding the characteristic of generality since the process of a transformation is detached in these modules (extraction, transformation and loading) that are independent of each other, being possible to make exchanges of transformations without the need to manipulate other modules. On the other hand, modulation allows each transformation to reuse the components that have the same function, therefore,

To perform a new transformation (add a new service to the mesh), just add the Transformation component and reuse the Extract and Load components. In such a way that the following can be defined:

$$\begin{aligned} transformations &= \{T_1, T_2, T_3, T_n\} \\ service_1 &= (E, T_1, L) \\ service_2 &= (E, T_2, L) \\ service_n &= (E, T_n, L) \end{aligned}$$

Where $transformations$ is a set of transformations that make up SMM. Each service can be seen as a tuple where each element is a module of the ETL approach, where for each different service the only thing that changes is the transformation or the task to be performed. On the other hand, the extraction and loading modules are used in the same way in each service, in such a way that there is a generality that allows the modules to be independent and reused as necessary.

The process of a solution can involve a combination of several services, such that for each different combination of services there is a different result. The result of the solution is strictly involved in the flow in which the services are chosen:

$$\begin{aligned} solution_1 &= (service_1, service_3, service_2) \\ solution_2 &= (service_2, service_1, service_3) \\ solution_1 &\neq solution_2 \end{aligned}$$

The use of different services requires that there is communication between them. As defined above, different sets of services make up the building blocks. The flow that defines the combination of services that belong to a single building block is defined by the graph that presents the order of the tasks to be executed shown in the equation 1. However, when you want to combine services that belong to different building blocks, it is necessary to establish communication between them to follow the flow that will define the result. The Client-Server model is implemented in all the building blocks to have the possibility of establishing communication between them, in such a way that if the flow of a solution uses services from different building blocks, the data to be processed can be transferred between the building blocks as well as the desired transformations.

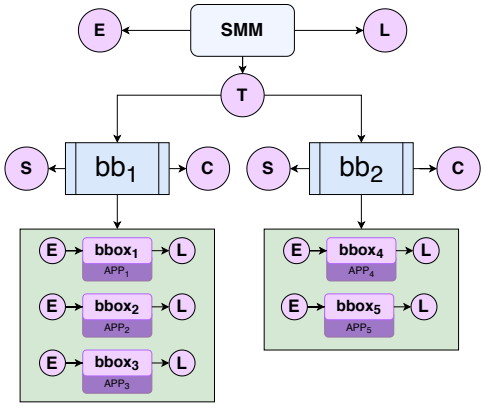

Figure 1: Graphical representation of the proposed model

Prototype implementation

A prototype was developed based on the proposed model. For the encapsulation of applications within building blocks, it was done through the use of virtual containers. Virtual containers allow the creation of a virtual execution environment. This self-contained and lightweight environment contains both the application and all the necessary dependencies for its operation. An access layer (interface) was also developed to manage the input and output of data as well as the execution of the contained applications. This access layer was named middleware. The middleware exposes the building block as a service and runs the applications it manages (in a certain order) based on the instructions that are sent to it. The applications within a building block are managed through a module called black box. The black box works as an access layer, delivering data to the application, executing it when it receives an order, and returning the results to the

middleware, which redirects them to another black box or delivers the results to the SMM. SMM is the coordinator and intermediary layer between building blocks and users. SMM receives a series of instructions as well as a set of input data and performs the coordinated execution of the services. For this work, three building blocks with different characteristics were developed:

- **Deep Learning:** Set of algorithms for supervised classification. Within this building block are three different neural networks: a recurrent neural network (RNN), a convolutional neural network (CNN), and a long short term memory neural network (LSTM). Given the nature of the results generated by neural networks, the applications in this building block cannot be chained.
- **Pre-processing:** Set of applications with techniques for data preprocessing. The following techniques were established:
  - *Imputation*. Aims to fill in the missing data of the dataset.
  - *Outliers*. Try to eliminate the noise that exists in the data based on extreme values.
  - *Normalization*. It aims to transform each one of the variables on a single scale for all. Usually between 0 and 1.
  - *Sampling*. Responsible for obtaining a sample of the set full of data.

  The applications in this building block can be chained and interchangeable since the result of each task it's a new version of data.
- **TS:** Application for making requests to a set of external services called TPS. TPS services are a generic set of functions for data indexing, transformation, and analysis. Some of the available TPS are:
  - *Clustering:* Offers unsupervised grouping algorithms.
  - *Describe:* Offers basic analytics services for data sets such as basic statistics (mean, median, mode, quartiles, etc.)
  - *ANOVA:* Correlation, variance and covariance analysis service.
  - *Cleaning Tools:* Data cleaning service, offers tools for deletion of outliers, omission, data replacement (by interpolation, mean, median or mode)
  - *Cluster Validation:* Service for the analysis of clusters from indexes (dunn, silhouette, tau, gamma, Davies Bouldin, etc.)
  - *Transform:* Offers tools for data restructuring and grouping.
  - *Graphics:* Service for the generation of statistical graphics in two and three dimensions (Linear, histogram, points, clusters, etc).
- **GloVE:**
- **Grobid:**

4. Experimental evaluation

For testing the functionality of the prototype implementing the service mesh model, three case studies were considered: a) Processing of climate data from the EMAS source, b) Classification of documents, and c) Clustering of tweets related to Covid-19.

5. Conclusions

References

[1] APACHE, . Nifi. URL: https://nifi.apache.org/.

[2] Cid-Fuentes, J.Á., Solà, S., Álvarez, P., Castro-Ginard, A., Badia, R.M., 2019. dislib: Large scale high performance machine learning in python, in: 2019 15th International Conference on eScience (eScience), IEEE. pp. 96–105.

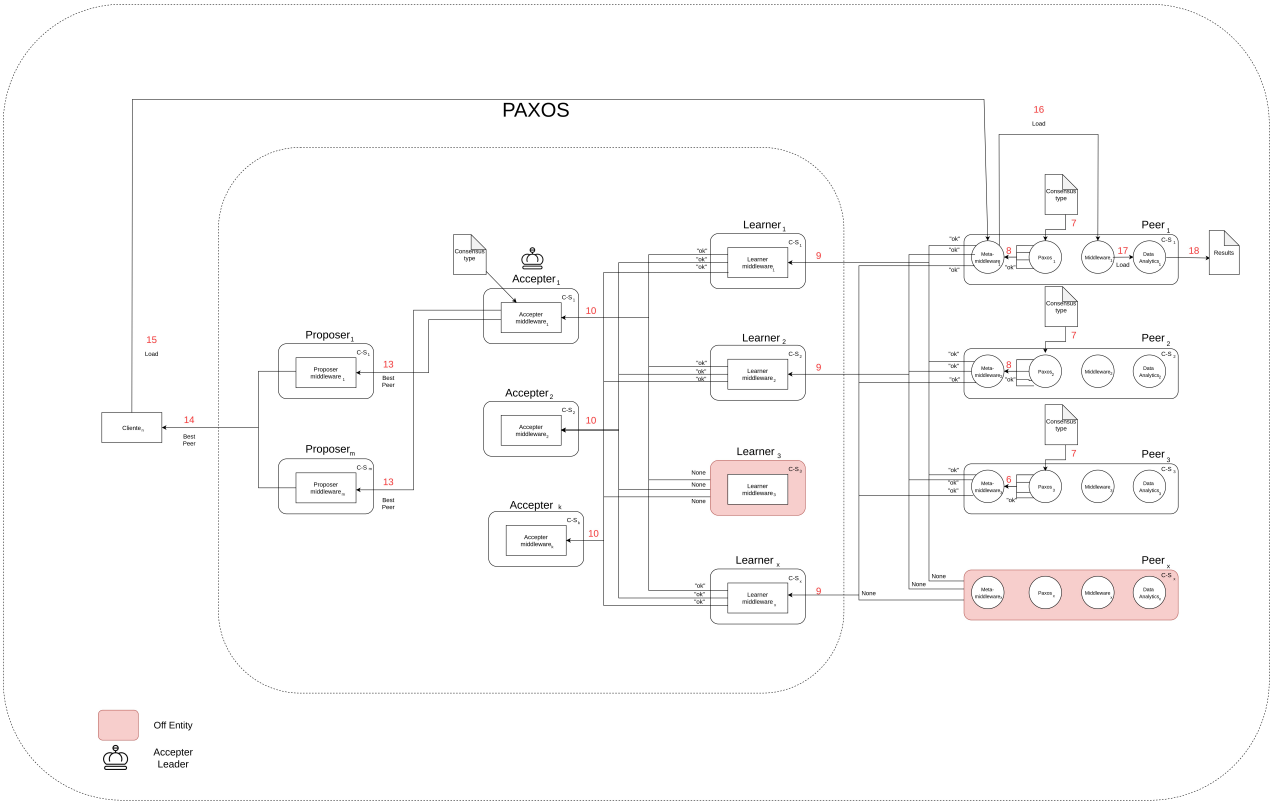

Figure 2: Paxos Flow part 2.