

This is a postprint version of the following published document:

Armando Barron-Lugo, J., Gonzalez-Compean, J. L., Carretero, J., Lopez-Arevalo, I. & Montella, R. (2021). A novel transversal processing model to build environmental big data services in the cloud. *Environmental Modelling & Software*, 144, 105173.

DOI: [10.1016/j.envsoft.2021.105173](https://doi.org/10.1016/j.envsoft.2021.105173)

© 2021 Elsevier Ltd. All rights reserved.



This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License](https://creativecommons.org/licenses/by-nc-nd/4.0/).

A novel transversal processing model to build environmental big data services in the cloud.

J. Armando Barron-Lugo^a, Jose L. Gonzalez-Compean^{a,*}, Jesus Carretero^b, Ivan Lopez-Arevalo^a and Raffaele Montella^c

^aCINVESTAV - Tamaulipas, Mexico

^bUC3M, Spain

^cUniversity of Naples Parthenope, Italy

ARTICLE INFO

Keywords:

Big data
Cloud computing
Environmental data
Climate data
Machine learning
Data analytic

ABSTRACT

This paper presents a novel transversal, agnostic-infrastructure, and generic processing model to build environmental big data services in the cloud. Transversality is used for building processing structures (PS) by reusing/coupling multiple existent software for processing environmental monitoring, climate, and earth observation data, even in execution time, with datasets available in cloud-based repositories. Infrastructure-agnosticism is used for deploying/executing PSs on/in edge, fog, and/or cloud. Genericity is used to embed analytic, merging information, machine learning, and statistic micro-services into PSs for automatically and transparently converting PSs into big data services to support decision-making procedures. A prototype was developed for conducting case studies based on the data climate classification, earth observation products, and making predictions of air data pollution by merging different monitoring climate data sources. The experimental evaluation revealed the efficacy and flexibility of this model to create complex environmental big data services.

1. Introduction

Visualization and analysis services have become key for processing and managing large volumes of environmental data, which have been keystone for scientific community to conduct complex scientific studies such as climate [43], environment [19] [51] and earth observation [41] [30].

Online repositories following FAIR principles (Findable, Accessible, Interoperable, and Reusable) [49] [59] 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 [48] [60]. These repositories not only represent a source of solutions for organizations to produce useful information for decision makers [54], but also an opportunity area to create environmental big data services based on software processing structures (PS). These structures are built by grouping and reusing available software and datasets, pipelines, patterns, service mesh, and workflows are examples of PSs available for organizations to build big data services. A PS is modelled as the interconnection of a set of processing stages (a stage executes a given software) represented by a *Directed Acyclic Graph (DAG)* where the nodes represent *stages* and edges represent the interconnection of a stage with any of other stage, data source/sink, or other PSs. These PSs process and/or manage environmental monitoring data from repositories [27]. For instance, in a traditional big data scenario, a PS commonly consider the


following stages: i) *acquisition of environmental monitoring data* from a source (e.g., either a dataset published in an online repository [27], data monitoring repositories[45], [46], or a downloaded/produced dataset); ii) *data preparation* for converting extracted data into a given format (e.g., GeoJSON, JSON or queries for a database); iii) *pre-processing* for removing potential erroneous data, and/or enriching the data by calculating/fulfilling missing data; iv) *processing or analysis* to convert data into useful information by using software for classification, grouping, prediction of values, merging data/information to name a few; v) *visualization* that prepare information for being consumed by decision makers.

Today, multiple PSs are already available for scientific community to process and analyze climate [20], pollution [7] [31], earth observation data [41] [32], processing contents [17] [33] or predicting meteorological changes [34] [18]. These PSs produce useful information [10] [60] that results critical [20] for decision-making procedures (e.g., to prevent disasters [36]).

Nevertheless, building environmental big data services based on existent software pieces and/or PSs is not trivial because of the following restrictions:

1. The programming languages used by the available frameworks for creating PSs is not necessarily the same, which is a problem when the environmental processing software has been written by using a different programming languages.
2. PSs and big data frameworks could require to be executed on a given platform and because of this, IT staff commonly ends up installing third party software and performing troubleshooting processes to overcome errors due to installation, configuration, data unavail-

*Corresponding author

 juan.barron@cinvestav.mx (J.A. Barron-Lugo);
jose Luis.gonzalez@cinvestav.mx (J.L. Gonzalez-Compean)
ORCID(s): 0000-0002-9619-8116 (J.A. Barron-Lugo);
0000-0002-2160-4407 (J.L. Gonzalez-Compean); 0000-0002-1413-4793 (J. Carretero); 0000-0002-7464-8438 (I. Lopez-Arevalo); 0000-0002-4767-2045 (R. Montella)

- ability, and path mistakes [3]. Moreover, these frameworks commonly impose the installation of third party solution for the management of computational resources.
3. Reusing solutions is not an immediate option for end-users to build a new service. This means, in most cases, a new solution must be built when creating a PS and is not feasible for end-users to reuse existent and already tested PSs.
 4. Analytic, statistic, and machine learning are commonly added to the solutions as a third party software [20], which could produce issues of programming languages, portability, and interoperability could arise when considering these frameworks in a solution.
 5. The access to the data at any stage: in traditional big data solutions, the PSs are created as black boxes, where a solution retrieves data from a *source* (e.g., any of folder, cloud location, or data lake), executes a set of applications at different stages for converting incoming data into useful information that is stored in a *sink* (e.g., cloud storage location). The end-user thus only gets access to the final results placed at the sink, but not necessarily to the results created/produced by intermediate stages, which could be useful either as input data for other solutions or being consumed by other end-users or applications.

In this context, it makes sense that the scientific community can reuse, either portions or whole, successfully installed and configured PSs [35] [53] to create comprehensive environmental big data services, which will result in saving time and human/infrastructure resources. It also makes sense that these PSs can be agnostic from programming languages, infrastructure, and platforms for enabling end-users to deploy/execute them on/in different infrastructures such as edge (e.g., personal computers), fog (e.g., servers of a company/organization) and/or cloud (e.g., virtual containers and machines provided by public providers) [40] without needing to download data and software or making installing configurations, which reduces the need for IT staff of organizations to perform troubleshooting processes.

However, reusing already installed and configured PSs, coupling it to existent PSs [12], sharing data produced in a PS with another one [56], publishing and consuming information at any stage of an existing PS during the data life-cycle [55] are challenging tasks, which have been only partially addressed by solutions in the state-of-the-art, such as pipelines [23], workflows [29] [4] [5] or services [9] [2]. Currently, for our best knowledge, there are no feasible and immediate options provided by different traditional frameworks for organizations to create environmental big data including the aforementioned features without making hard-working adaptations.

This paper presents a novel transversal, agnostic-infrastructure, and generic processing model to build environmental big data services in the cloud.

Transversality property is used for building PS by reusing/-coupling multiple existent software for processing environmental monitoring, climate, and earth observation data, even

in execution time, with datasets available in cloud-based repositories. The transversal model adds the *transversality property* to PSs by creating and using a set of *coupling* and *extracting transversal points*.

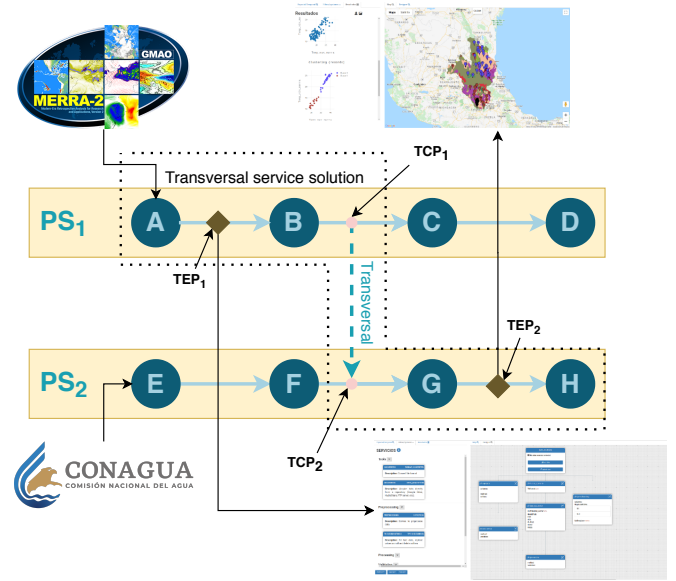


Figure 1: Transversal property for converting two existent PSs into a new transversal service.

Figure 1 shows an example of a transversal service built in the cloud by reusing two existent PSs (one for processing re-analysis MERRA-2 products and the other for processing monitoring data from ground stations by CONAGUA¹ placed at Mexico). As it can be seen, the *Transversal Coupling Points* (TCPs) interconnect two stages, one from PS₁ and the other one from PS₂ (transversal union between PS₁ and PS₂ in Figure 1 through stages B and G). This virtual coupling is performed, even in execution time, to create a new independent *Transversal Service* (TS). The *Transversal Extraction Points* (TEPs) enable end-users or applications to consume and process data (see TEP₁ and TEP₂ in Figure 1). The extraction services enable either end-users or other services to retrieve, deliver, and/or visualize data/results produced by a given stage of an existing PS (See TEP₁ between A and B of PS₁, and TEP₂ between G and H of PS₂ in Figure 1).

Genericity, in the context of this paper, refers to the property of analyzing and producing information produced by any environmental software without restrictions of language programming, infrastructure, platform, and software. This property is used to embed analytic, merging information, machine learning, and statistic into generic micro-services coupled to PSs for automatically and transparently converting them into big data services to support decision-making procedures. The embedded micro-services take advantage of TEPs, providing services such as data fusion to produce useful information to support decision-making procedures. Figure 1 shows how two PSs are processing two different cli-

¹<https://smn.conagua.gob.mx/tools/GUI/EMAS.php>

mate data sources, and how embedded micro-services consume data from TEPs for producing/visualizing information. The first embedded micro-service, connected to the TEP₁ executes a classification tool to merge metrics from two data sources, and then to group climate metrics by statistic similitude values. The second embedded micro-service, connected to TEP₂ executes a visualization tool (a Geoportal in this case) for extracting results produced by stage G (classification of metrics) and to show them in a map by using GIS embedded service.

Infrastructure-agnosticism is used for deploying/executing PSs on/in edge, fog, and/or cloud without end-users requiring to perform installation, configuration steps, which reduces the need for performing troubleshooting procedures. This model also adds agnosticism from infrastructure to the TS (any of PS, systems of PSs or big data services) by encapsulating environmental software into virtual containers and managing them by using micro-service architecture.

The transversal, generic and agnostic properties, added to existent PSs, enable end-users to create complex big data services by reusing multiple existing software created with different frameworks and deployed on different IT infrastructures. The building of transversal environmental big data services thus can be performed in three simple steps: i) Builds a PS by choosing existing software and datasets[49]. ii) Creates a new independent service by defining TCPs reusing already configured and tested PSs (any of online available, downloaded in a given infrastructure or created by this model). iii) Creates a big data service by using TEPs and the embedded micro-services that produce useful information.

when performing these three steps, a new transversal service is created for end-users and / or decision makers to consume the information produced by this solution through independent transversal processes service (TPS). Applying the three steps to the creation of environmental services also enable decoupling the environmental software, from the PSs as well as from the analytic and machine learning tools. In short, the TPSs converts a traditional set of software pieces into an environmental big data service.

A prototype was developed and implemented based on this model to create environmental big data services, which were created based on existing environmental software for conducting three case studies: the first one for processing of satellite imagery to build *Earth Observation Products* (EOPs) and yielding environmental indexes. The second one for processing monitoring and re-analysis climate data. The resultant data were processed by clustering algorithms to create groups of ground stations by using statistical similitude values. The third one is a data fusion service for producing pollution predictions using machine learning tools (included in embedded micro-services) by using multiple data sources, such as monitoring and re-analysis climate (MERRA [27] REDMET [46]), and monitoring air pollution (RAMA [45]) sources. The experimental evaluation, based on the three case studies, revealed the efficacy and flexibility of this model to create complex environmental big data services for processing heterogeneous data (such as re-analysis and moni-

toring climate and environment data). It also revealed the feasibility of this model to perform data/information fusion to produce useful results, which can be consumed by decision makers through online services (TPS).

The paper is organized as follows: preliminaries and background information are given in Section 2; Section 3 frames the state-of-the-art about the topic; the novel transversal computing model is described in Section 4; the experimental evaluation is explained in Section 5; finally the Section 6 is dedicated to the conclusion remarks and future directions.

2. Preliminaries

2.1. An overview of processing models for Big data

In a traditional big data scenario, the stages are created as blocks executing a given application or tool by using an *Extract-Transform-Load* (ETL) processing model. In this model, a block *extracts* data from a *source* (e.g., any of folder, cloud location, or data lake), executes the applications encapsulated into the stage for *transforming* the incoming data into useful information that is *loaded* in a *sink* (e.g., cloud storage location).

Pipelines and computerized workflows are examples of PSs that allow end-users to process data through multiple blocks (stages such as acquisition, pre-processing, processing, and preservation) that are chained by following a given sequence. The simplest coupling method is a pipeline, where the stages are interconnected sequentially following an ETL, the coupling is performed by using the data sources and sinks of the stages. For instance, in a traditional big data pipeline, an acquisition block (stage) *extracts* data from a web page, creates indexes (*transforms*), and *loads* the indexes in a database, placed commonly in a cloud location, where a pre-processing stage *extracts* the indexes from the database to locate the extracted data for cleaning them (e.g., removing outliers and noise data or calculating missing data by using extrapolation). This pre-processing commonly *transforms* raw data into a cleaned data version, which is also *loaded* as indexes in the database and stored in a cloud location. A processing stage (e.g., analytics or machine learning tools) *extracts* the indexes to get the cleaned data and to transform them into reduced information (e.g., any of categories, classification or ordered groups), which is *loaded* in a database as information. At this point, this information can be consumed by end-users in decision-making processes [52].

From the point of view of the users participating in a decision-making process, all these stages are integrated in a single service, and they only has access to the raw data and/or resultant information.

2.2. An overview of stages coupling method and data management

The ETL model is used for the coupling of stages through data sources and sinks. It is commonly represented as a *DAG* [5] [9], which is used for a wide range of tools [6] and frameworks [29] [2] [47] to create analytic and processing services in either an automatic or semi-automatic manner.

Some frameworks impose to end-users a programming language (e.g., Python [29] [9] [4] or Java [47]) for building PSs. Other frameworks also impose infrastructure, platform requirements [5] [4], or they only enable the users to use solutions in the cloud [8] [1] [6].

In real-world scenarios, this results in a complex PS ecosystem where multiple PSs are built by multiple frameworks, using different programming languages, and deployed on multiple platforms over different infrastructures. The following tasks could be required in such an ecosystem, which are addressed by the model presented in this paper:

1. Reusing partial or whole PSs (any of applications, stages, pipelines, or workflows).
2. Building services based on multiple existing PSs by interconnecting them either in execution or configuration times.
3. Allowing on-the-fly and on-demand changes in existing PS.
4. Requiring that different stages sharing a data source (e.g., data warehouse or data lake) or sinks.
5. Consuming data produced by intermediate stages, not only from the starting data sources and ending sinks.
6. Creating information crossing processes by including multiple already developed applications processing multiple data sources or sinks (including the sources and/or sinks of intermediate stages).

3. Related work

Engines and frameworks have been key for the scientific community to build big data PSs [58]. Processing workflows and pipelines are not a new technology, there are multiple tools available that allow users to design and manage the execution of tasks, either locally or in a distributed environment. Taverna [21], for example, is a framework created for the execution of bioinformatic workflows, offering a variety of processes (services) in a catalog used for the construction of multiple workflows. It also enables designers to invoke external services to incorporate them into their workflows, as well as interfaces for the construction and execution of these processes. In this way, users can build their processing workflows without much experience in computing areas. These solutions can be shared through myExperiment [14] (a repository of workflows) which allows using algorithms for detecting useful fragments or workflows to create tasks (hierarchy of tasks). Similar to this framework, a wide catalog of workflow systems (workflow engines) has been reported in the state-of-the-art including but not reduced to popular examples, such as COMPS [5, 50], Makeflow [2], Pegasus [9], Galaxy [15], and DagOnStar [29]. Although those engines and frameworks have different characteristics, such as the programming model or the tasks' execution method, all of them have solutions based on DAGs. These structures do not consider explicitly the coupling of multiple workflows by building meta-workflows (workflows over workflows). For instance, it is not feasible for these

solutions to reuse data already processed by previously executed PS (or being executed) without changing codifications of applications. Moreover, the installation and configuration of a new PS must be created instead to reuse existing PSs, which only can be modified at configuration time (not in running time).

After studying these frameworks, we observed that the challenge of managing ecosystems of PSs has been focused mainly on two directions: The first one is focused on the construction of solutions based on hierarchical levels; the second one is focused on the composition of new solutions by reusing PS fragments.

The main usage of the solutions based on a hierarchical approach [57] is the discovery of PSs and the composition of different solutions as a single one. It is usually based on two hierarchy levels. The first one (e.g., *Decaf* [11]) for the *in situ* workflow composition, with tasks that exchange messages through memory composed of a set of fields that may or may not be fully utilized by the solutions. The second one is the development of new solutions by using Meta-DAGs (e.g., by using PyCOMPS and *Decaf* [50] [5]) to execute *in situ* workflows as tasks in a distributed environment. In this type of solution, the management of dependencies and the transport of data between tasks (associated to the edges of the DAG) is performed by following the Meta-DAG and encompassing the task execution, task coordination, task parallelization and transport of data.

In turn, the solutions that reuse workflow fragments for the composition of new solutions are mainly focused on algorithms that search for fragments of tasks within workflow repositories, that are suitable to be reused when building new solutions [61] [13]. Some of these works produce the automatic design of solutions by using fragments of workflows, but there are still limitations and areas of opportunity of this type of approaches that the model proposed in this paper supports.

There are three main differences between the former proposals and the transversal model proposed in this paper. The first one are the transversal coupling services, that provides a new scope for the problem of dealing with a PS ecosystem by managing PSs as services. These services can be coupled with other PS (mainly external ones) through a new independent services (transversal processing service). This can be done without affecting/modifying the applications of an existing PS by using some of these stages on different solutions. The second one is the management and analytical processing of data based on extracting/publishing TSs, which are focused on transform data into information as well as enabling crossing information processes. This converts multiple PS into a configurable big data service instead of a single purpose solution as performed in traditional approaches. The last one is that this model allows creating external solutions, which means that this model can be also used by traditional frameworks to create PSs as services by reusing solutions and the data produced by their stages, as shown in the experimental evaluation of this paper.

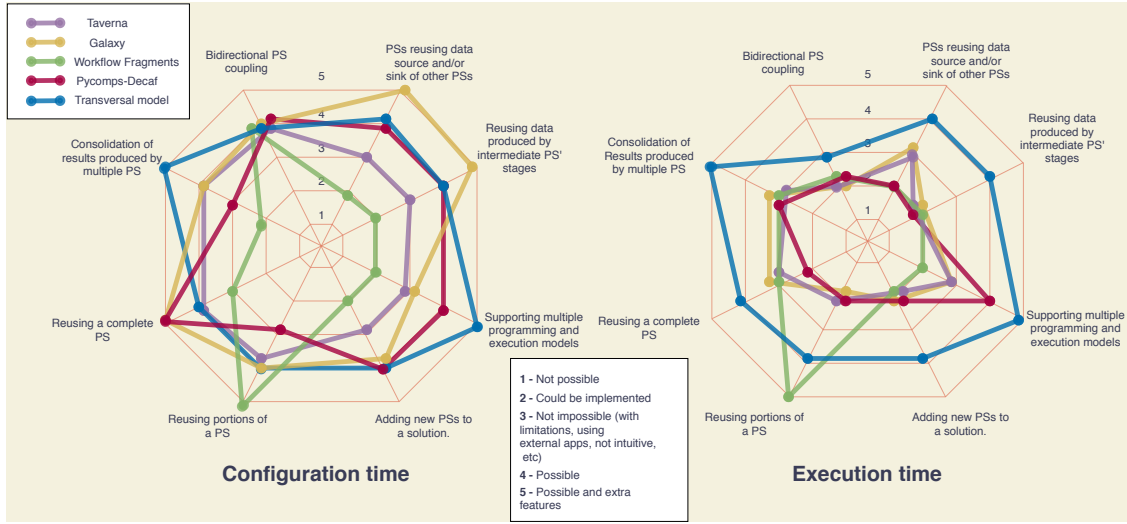


Figure 2: Usability comparison study.

Usability comparison study

In order to show the aforementioned differences, a comparative usability study, from the end-user point of view, was carried out between some frameworks found in the literature that address the problem described in this paper, such as Taverna [21], Galaxy [15], cross-workflow fragments [61], and Pycomps-Decaf [57], and the transversal model proposed in this paper.

Figure 2 shows the differences considering eight different characteristics:

- *Bidirectional PS coupling.* It refers to the situations where a solution requires coupling two different PSs bidirectionally.
- *PSs reusing data source and/or sink of other PSs.* It refers to situations where PSs to reuse results produced by other PSs.
- *Reusing data produced by intermediate PS' stages.* It refers to situations where reusing data produced by intermediate stages of other PS without modifying the code of applications.
- *Supporting multiple programming and execution models for each PS belonging to a solution.*
- *Adding new PSs to a solution.*
- *Reusing portions of a PS.*
- *Reusing a complete PS.*
- *Consolidation of results produced by different PSs.* Referring to the ability to consolidate the results (analyze, index, publish, visualize, etc.) and to convert data into information.

Each characteristic was rated on a scale of 1 to 5:

1. It is not currently supported by the tool/model/schema to perform the activity.

2. It is not currently supported by the tool/model/schema to perform the activity, but it can be implemented according to its design principles.
3. It is currently supported by the tool/model/scheme to carry out the activity, but external tools are required to do it.
4. It is currently supported by the tool/model/schema to perform the activity.
5. It is currently supported by the tool/model/scheme to perform the activity, and additionally, it offers non-functional requirements (scalability, efficiency, security, modularity, reliability, etc.).

As it can be seen, Figure 2 describes, from an end-user point of view, the opportunity areas of more prominent frameworks and engines, and how its integration with a transversal model could fulfill these areas for improving the flexibility and functionality of current frameworks and engines. The proposed transversal model takes into account these opportunity areas, allowing a management of solutions composed by others, using the workflow engines themselves as the task execution tool, unifying the programming model, and maintaining the characteristics of each model of execution. Table 1 shows a summary of the qualitative differences of the works considered in this study.

4. A novel transversal processing model for environmental big data services

In this section, we present a novel *transversal processing* model for building big data services in fog-cloud environments. The model allows the integration of multiple existing/deployed software and/or applications into new solutions exposed as a service.

Transversal processing structures

A *PS* can be represented as a tuple including a set of stages (V) and their corresponding connections (E):

Table 1

Usability comparison between related jobs. *E* represents the action at running time; *C* represents the action at configuration time.

Work	Reuse					Consolidation of results	
	Processing structures		Data			Indexing	Functions as a service
	Complete	Partial	Initials	intermediate	Final		
Taverna [21, 26]	C ✓	C ✓	C ✓		C ✓		C ✓
Galaxy [15]	C ✓	C ✓	C ✓	C ✓	C ✓	E ✓	C ✓
Workflow fragments [61]	C ✓	C ✓					
Pycomps-Decaf [57]	C ✓	C ✓	C ✓		C ✓		
Transversal model	E ✓ C ✓	E ✓ C ✓	E ✓ C ✓	E ✓ C ✓	E ✓ C ✓	E ✓ C ✓	E ✓ C ✓

$$PS = (V, E) \quad (1)$$

where:

- *V*: Represents a set of stages $STAGE_i$ in the form $V = \{STAGE_1, STAGE_2, \dots, STAGE_v\}$.
- *E*: represents a set of ordered pairs including elements of *V*, representing the interconnections between the stages, $E = \{(x, y) \in V \times V | x \neq y\}$.

Each stage ($STAGE_i$) performs a transformation task following the ETL processing model, obtaining data from a source (ds_j) and depositing the transformed data in a sink (ds_k), which can be used by either end-users or other stages for getting access to the transformed data. Both the source and the sink belong to a data repository (*PSData*). In this context, a stage can be represented as follows:

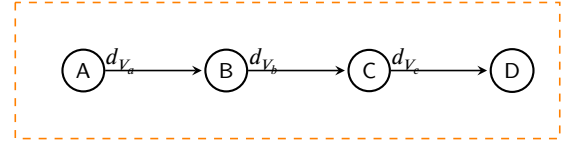
$$STAGE_i = ds_j \rightarrow Task \rightarrow ds_k \mid ds_j, ds_k \in PSData \wedge j \neq k \quad (2)$$

where:

- ds_j : Represents the input data, which is an input for the transformation task.
- ds_k : Represents the output data deposited in a sink (e.g., data warehouse).
- *Task*: Represents a task, an activity, or a data transformation process.
- *PSData*: It is a set of all the data sources used by the stages of a PS.

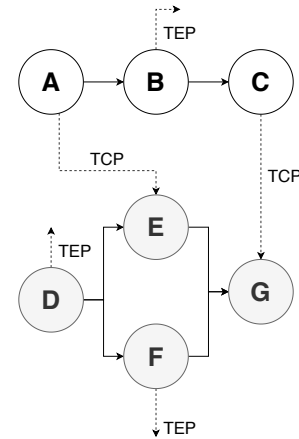
To represent a structure as a DAG, nodes represent each stage ($STAGE_i \in V$) of a PS, while the edges represent the corresponding ordered pair (*E*), which defines the sequence of execution of each stage. Figure 3 shows an example of a DAG defined for a *PS* (a processing pipeline in this example), where the ordered execution of the tasks (A, B, C, D) of each stage produces different versions of data (d_{V_x}) from one stage to another one.

The model mainly considers scenarios where a set of *PS* is used to build a transversal service (*TS*) that is defined by


Figure 3: DAG describing a pipeline.

its DAG representation (DAG_{TP}), which is created by using a set of *Transversal Processing Points* (*TPP*). Two types of *TPP* have been defined in this model (Figure 4):

- *Transversal Coupling Points* (*TCP*). These points create an abstract intersection between tasks that belong to different PSs.
- *Transversal Extraction Points* (*TEP*). These points create input/output interfaces for solutions to extract-/deliver data from/to existing PSs (either at setup or run times). These points also includes a link to access data or to be used as input in a TPS (see Section 4.1) to transform data into information.


Figure 4: Conceptual representation of TPPs: TCP coupling two PSs, and TEP enabled for data extraction.

A DAG_{TP} thus can be represented with the following expression:

$$DAG_{TP} = (SolT, Links) \quad (3)$$

where $SolT$ is the set of PS included in a transversal solution, while $Links$ is a set of TPP that represents the virtual interconnections of the $PS \in SolT$. Each TPP_i can be a TCP or a TEP . This is represented as follows:

$$Links = \{TPP_1, TPP_2, \dots, TPP_m\} \quad (4)$$

$$m \geq 1$$

$$SolT = \{PS_1, PS_2, \dots, PS_n\} \quad (5)$$

$$(n = 1 \wedge TPP_i \text{ is a } TEP) \vee$$

$$(n \geq 2 \wedge TPP_i \text{ is a } (TCP \vee TEP))$$

A TCP represents an interconnection between stages through a data source belonging to the set $PSData_k$ of a PS_k ($ds_j \in PSData_k$) and a data source belonging to the set $PSData_l$ of a PS_l ($ds_h \in PSData_l$). In this sense, at least two PS ($n \geq 2$) must exist to define a TCP . On the other hand, a TEP is connected to a data source ds_j in a PS_k , which is in the $PSData_k$ set. This is represented as:

$$TCP = (x, y) | (x, y) \in PSData_k \times PSData_l \quad (6)$$

$$k \neq l$$

$$TEP = ds_j \in PSData_k \quad (7)$$

4.1. Transversal Processing Services (TPS)

Collaborative work between different PS does not necessarily imply reusing either stages or information generated to be consumed by other PS , but it can also be used by an external process to provide useful information to the end-user. We call this type of process a *Transversal Processing Services* (TPS). A TPS takes advantage of data produced by a PS for performing a transformation process to create useful information. A TPS performs data collection (ds_j) by using $TEPs$ ($TEP_i = ds_j \in PSData_k$). Multiple data sources can be joined together (e.g., to carry out a data fusion) and published for later use by TPS to produce information. This is defined by joining structured data as follows:

$$TEP_i = ds_j = (REG_j, ATT_j, KG_j) \quad (8)$$

where:

- REG_j is a set of records of data collection: $REG_j = \{reg_1, reg_2, \dots, reg_n\}$.
- ATT_j is a set of attributes of data collection: $ATT_j = \{a_1, a_2, \dots, a_m\} |$
 $a_k = (name_k, type_k, role_k)$.
 1. *name*. Name of the attribute.
 2. *type*. Data type (int, double, char, string, etc.).
 3. *role*: Role of the attribute, which can assume the values of “value” or “keygroup”.
- KG_j represents a set of attributes a_k whose role (*role*) assumes the value of “keygroup”: $KG_j = \{a | a_k \in ATT_j, role_k = keygroup\}$.

To match data from two sources, a relation between them must be created. In the case of structured data, the model considers creating joins among multiple records from one table to another based on keys or group attributes (*keygroups*). In this sense, two $TEPs$ are consolidated in a single data source by joining the records from both sources based on the KG_j , as defined below:

$$REG\tau = REG_j \cup REG_k \quad (9)$$

$$\forall a_i \in KG_j \exists a_h \in KG_k \wedge j \neq k$$

$$ATT\tau = ATT_j \cup ATT_k | j \neq k \quad (10)$$

$$KG\tau = \{a | a_i \in ATT\tau, role_i = keygroup\} \quad (11)$$

$$ds\tau = (REG\tau, ATT\tau, KG\tau) \quad (12)$$

$$ds\tau \text{ is a } TEP$$

As it is a recursive process, multiple extracted data sources ($TEPs$) can be joined in pairs at different levels. For example, TEP_1 and TEP_2 can be joined into a single TEP_3 , and later, join TEP_3 with a TEP_4 .

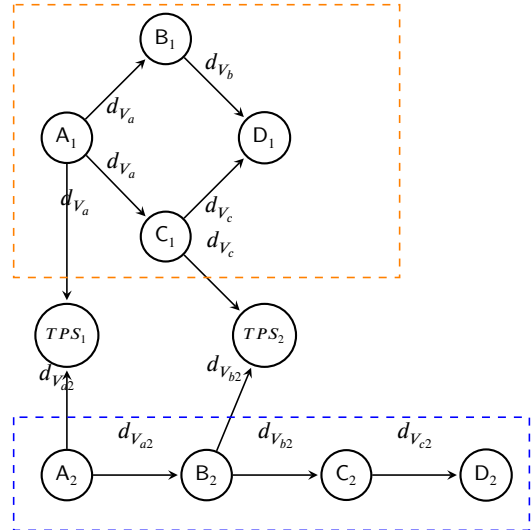


Figure 5: Representation of TPS as a DAG. TPS_1 and TPS_2 consolidate the results produced by two different PS .

A TPS_i (see the example in Figure 5) follows an ETL process with one input (TEP_j), one embedded task (*EmbTask*), and one output (TEP_k). In this sense, the input of a TPS can be any of the union of data sources ($ds\tau$), a data source obtained by a TEP , or the output of another TPS (TEP_j). Thus, the data is extracted from a TEP_j and deposited in TEP_k . However, there are no restrictions as to which TEP to use for the results, so j can be equal to k ($j = k$ or $j \neq k$):

$$TPS_i = TEP_j \rightarrow EmbTask \rightarrow TEP_k \quad (13)$$

In this paper, a *service mesh* is defined as a pool of transversal coupling and extracting/publish services $TCPs$, $TEPs$, as well as PSs and the TS created by the end-user TPS .

Thus, a mesh can be represented by the set $mesh = \{TPS, TCP, PS, TEP\}$ where each component is exposed as a service.

Transversal service (TS) components: design details

The proposed model has been developed by using the following components for managing, in distributed environments, the TSs registered in the service mesh:

- *Coupler*, uses TCP for coupling either tasks that belong to different PS or multiple solutions through inputs and outputs. This entity creates new solutions as a service.
- *Manager*, coordinates the integration of the TCPs into a new service (in either configuration or execution times). Supervises the arisen of cycles in the resulting solution (DAG_{TP}). Publishes the solutions as a service in a service mesh.
- *Extractors*, use TEP for data retrieval from any intersection of tasks (stages) of a solution, executes embedded services (e.g., analytics, machine learning, statistics, or probabilistic methods), and index these data to make them available as a service.

The *transversal processing model* considers the following services of transversal processing:

- *Integrated solutions as a service (TPIS)*. These services represent the union of either tasks that belong to different PS (TSs included) into a new service. These services are created by integrating a set of coupling TCP into a DAG_{TP} .
- *Transversal processing services (TPS)*. This is an ETL-based service for consuming extracted data from a DAG_{TP} and transforming them into information. A TPS thus extracts data from a TEP, transforms data into useful information by using embedded services (e.g., any combination of analytics processing, statistics, machine learning), and delivers/load it to other transversal extracting/publish point.

Figure 6 depicts the interactions of the model components, the dataflows and the materialization of the transversal model when creating big data transversal solutions as a service.

The following methods have been designed to create a TPIS:

1. *Adjacent coupling (AC)*. This method enables developers to create new services based on existing TPS and/or PSs. This method is useful when the data produced by one solution (DAG_{TP_1}) must be used as input data for another solution (DAG_{TP_2}). In this method, DAG_{TP_1} is reused, even in execution time, and no new installation or configuration is required to create a new solution; the execution schedule of DAG_{TP_1} and DAG_{TP_2} is coordinated by the *manager*.

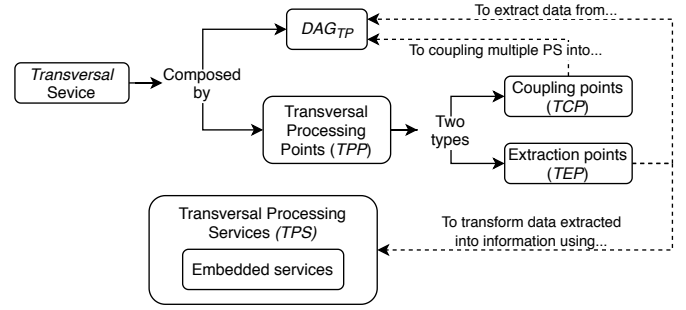


Figure 6: Graphic summary of the components of the model.

2. *Recursive Coupling (RC)*. This method allows to create a new service by using portions of a PS, allowing to divide large solutions into small modules with simpler objectives. In this method, a DAG_{TP} includes a set of solution portions connected through TCP (see the example of solution in Figure 7).
3. *Multiple-Sink Consume scheme (MSC)*. This method creates TEPs to compare results produced by different solutions by using TPSs. *MSC* is useful when the data produced by multiple PS ($PS \in SolT$) and/or TPS can be used together to get information. This offers a way to provide users with *crossing information functions as a service* (CIaaS) (see Figure 8).

A prototype based on the transversal processing model

Figure 9 shows a conceptual representation of the prototype, which depicts all the components of the prototype. It also considers the services developed to materialize the transversal model (Pink), the existent software reused by this model (orange bold lined) and the existent software available but not used (orange).

The first implementation of the transversal model was performed over the DagOnStar engine by using the dependency management schema (known as *DagOnStar workflow://schema* [42]). This schema manages the data used by tasks of the stages running in a PS, either locally or on a remote node. TCPs based on this schema were defined to manage the dependencies of multiple PSs by using reserved tag $\langle T \rangle$: and then to apply the transversal model to create services. This label represents the virtual path in which the results produced by a task can be found by a TPPs. In this way, a user can establish the data path for processing it using the following syntax: $\langle T \rangle : \langle PS \rangle / \langle task \rangle /$. In this notation, $\langle PS \rangle$ is an identifier of the PS and $\langle task \rangle$ is the identifier of the task which produce the data (e.g., $\langle T \rangle : PS1 / TaskA /$). This analysis of the dependency management in an engine gave us the insights to identify the place where the TPCs could be invoked by an engine. With this experience, the transversal model was adjusted not only for the creation of transversal solutions, TCP, TEP, and embedded services, but also for the creation of rules that the workflow engines could understand, in order to create PS by using the programming

Transversal processing model

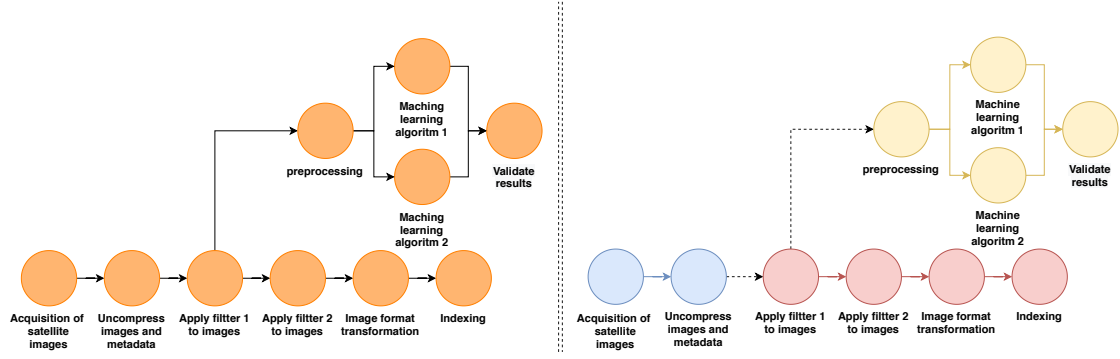


Figure 7: Example of three module PS: Acquisition (blue), Images transformation (red) and machine learning techniques (yellow).

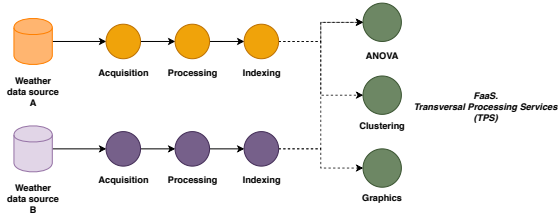


Figure 8: MSC scenario example: leveraging results through ClaaS. Results are used by a clustering service, an ANOVA (analysis of variance) service, and a graphic service

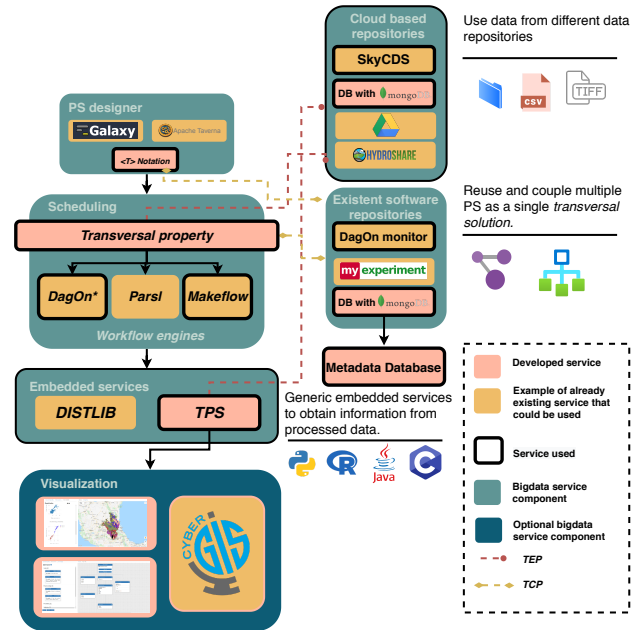


Figure 9: Conceptual representation of the prototype.

models used by these engines. In this context, these rules were defined for DagOnStar [29] and Makeflow[2]. These rules showed that this model is quite extensible to other engines and PS frameworks by creating rules based on the input/output results management of each framework. Figure 9 shows a landscape of the multiple services that can integrate a transversal service.

Implementation details of transversal processing prototype components

The TEPs, TPS, the coupler, extractors, and manager were developed in Python. The TPS APIs included REST I/O messages exchange and FTP for data exchange². Moreover, the end-user can consume data from online repositories: Hydroshare [49], Google Drive, OneDrive, FTP Servers, and SkyCDS [16]. For this, it is only necessary to provide a URL that points directly to the data (the filename and catalog in the case of SkyCDS), in this way, an acquisition service is in charge of obtaining it and applying the user-defined processing (PS or TPS).

The components of the TPSs were encapsulated into virtual containers (Docker³ was used for this prototype). The manager organized the virtual containers in the form of micro-services including a REST-based I/O management and a request dispatcher. Virtual an underlying container management (Docker compose⁴ for this prototype) is used for the deployment of *TSs*.

Table 2 shows the infrastructure used for deploying the prototype on container-based cloud where the experiments considered in the case studies were conducted in.

5. Experimental evaluation and results

For testing the functionality of the prototype implementing the transversal processing model, three case studies were considered: a) processing Landsat8 imagery to produce Earth Observation Products, b) processing climate data from the MERRA-2 project for building an online climate map, and c) a multiple-sink crossing information service based on machine learning for the classification of air pollution values.

Case study I: Processing Landsat8 imagery for producing EOPs by using a recursive coupling of PSs

The first case study is based on the building and operation of a solution for processing satellite images captured by

²A content delivery network [16] is currently on development for data management in the transversal model and a parallelism management schema [37] as well

³<https://docs.docker.com>

⁴<https://docs.docker.com/compose>

Machine	Sockets	Cores per socket	Threads per socket	Memory
Compute6	1	6	2	64GB
Compute8	2	8	1	64GB
Compute11	1	12	1	64GB

Table 2

Computing resource table.

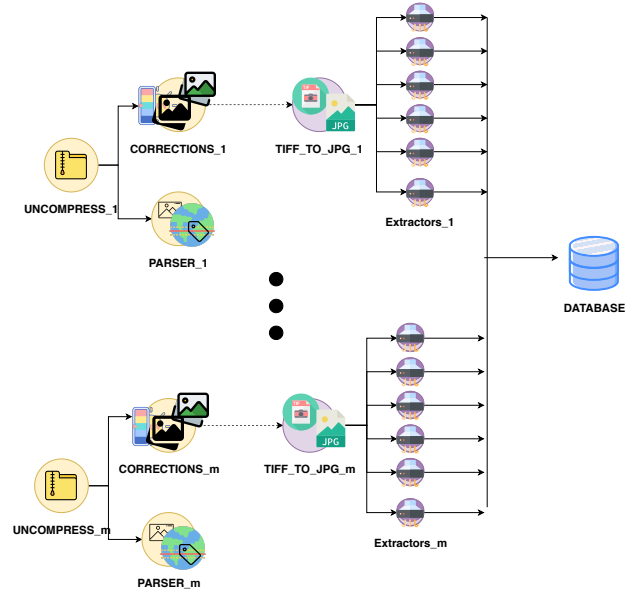
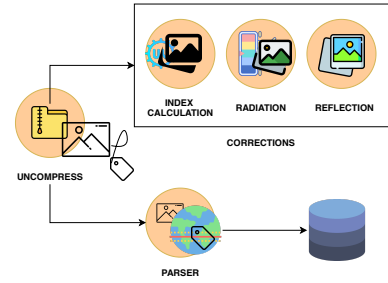
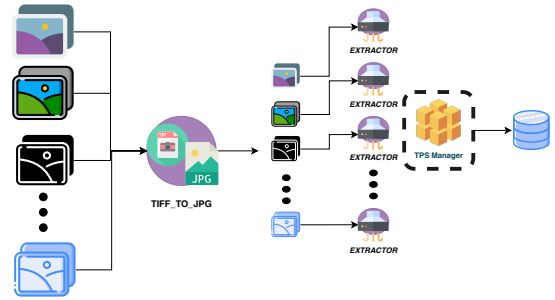
Landsat8⁵ [39] satellite (see Figure 10). This solution considers PSs for image decompression, the correction of bands, the application of filters to obtain and produce a set of Earth Observation Products (EOPs), and its publication in a geoportal. These PSs consider the following four tasks:

- *Uncompress*: It decompresses TAR files that contain all the bands of Landsat8 images and corresponding metadata.
- *Corrections*: It performs radiometric and atmospheric corrections to each image in the Landsat8 imagery and produces corrected images.
- *Indexes and EOPs*: It receives corrected images and creates Surface Reflectance-Derived Spectral Indices such as: Normalized Difference Vegetation Index (NDVI), Enhanced Vegetation Index (EVI), Soil Adjusted Vegetation Index (SAVI), Modified Soil Adjusted Vegetation Index (MSAVI), Normalized Difference Moisture Index (NDMI), Normalized Burn Ratio (NBR), and Normalized Burn Ratio 2 (NBR2).
- *Parser*: It obtains information from the metadata to determine spatio-temporal parameters of each EOP created by the PS based on geographical location of each EOP.
- *TIFF2JPG*: It converts the EOP's TIFF image format into a JPG for reducing the resolution and file size for efficiently visualizing by a geoportal.

For this case study, the PSs were organized in the form of two PS by using the *Recursive Coupling (RC)* method (Method 2) to create a big data transversal service (*BigData_{TP}*). The first PS, called *Corrections Landsat8 (CL8)*, includes the *Uncompress* and *Corrections* tasks to transform the bands of the images in the Landsat8 repository into new corrected products (see Figure 11). The second one, called *Processing Landsat8 (PL8)*, includes the *Indexes and EOPs*, and *TIFF2JPG* tasks for producing and indexing new EOPs and lowering the resolution of these EOPs (see Figure 12).

The *BigData_{TP}* service included extractors (TEP) for indexing each of the products (original images, corrected products and index derivative EOPs) by using the *Multiple-Sink Consume (MSC)* method (Method 3). This means that the end-users not only can get access to the raw data and derivative EOPs, but also to the corrected images (radiometric and atmospheric corrections) and the indexes.

⁵<https://glvis.usgs.gov/app>.

**Figure 10:** Landsat8 images processing workflow (CL8&PL8).**Figure 11:** CL8 processing structure.**Figure 12:** PL8 processing structure.

The processing and data management of the *BigData_{TP}* service is expensive in terms of time execution and memory consumption. For example, for each Landsat8 image, with an average size of 250 MB, are produced a large set of products to be indexed, managed, and preserved when the PS finishes its executions. In average, 10 GB of data are produced per processed image.

The execution of experiments was carried out based on the following characteristics:

- **Dataset**: it includes 23 satellite images (m) Landsat8

of 250 MB of average size, with 16/32 bands (each band is an image), which were processed one-by-one by the *BigData_{TP}* service.

- Memory and storage: the processing of each image produced, an average of, 10 GB of data (corrected original images and indexed images), which required 230 GB of storage for processing the whole dataset. It is important to note that this is only a fraction of imagery to conduct EOP production for spatio-temporal studies; as a result, it is expected that this capacity considerably increases in production, resulting in a big data issue.
- A TEP is created for each index. The TEPs run in parallel.
- The experiments were performed 31 times and the median *response time* metric was evaluated. The response time metric represents the sum of the times produced by each stage of each PS considered by the big data service (*BigData_{TP}*) evaluated in this case study.

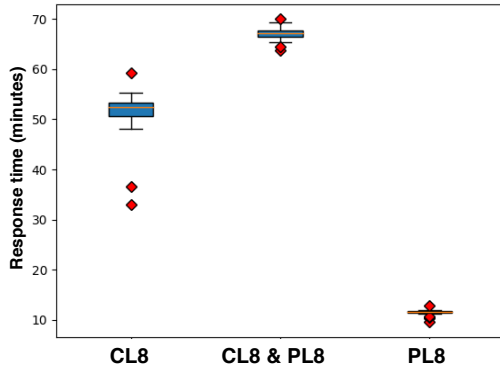


Figure 13: Response times of the CL8, PL8, and CL8&PL8 solutions.

A comparison of the response times was performed by each request attended by *BigData_{TP}* service. Figure 13 shows the response times in minutes (vertical axis) for each component of the *BigData_{TP}*. In this case, CL8 produced significantly longer response time than PL8, meaning this PS creates a bottleneck, and it would be re-structured to solve this issue.

Figure 14 also shows a performance comparison between *BigData_{TP}* service and the traditional pipeline using in both cases two parallel threads per task. A reduction in response time of approximately three minutes was obtained in the case of CL8 and PL8 joined by the transversal model in comparison with a solution built traditionally. In terms of performance, it was observed that the solutions built by using the transversal model not only enable the reusing of the parts of a PS independently and extracting/indexing data from these parts, but also it is performed without evident overhead. Even a reduction in time was observed in the experiments by this

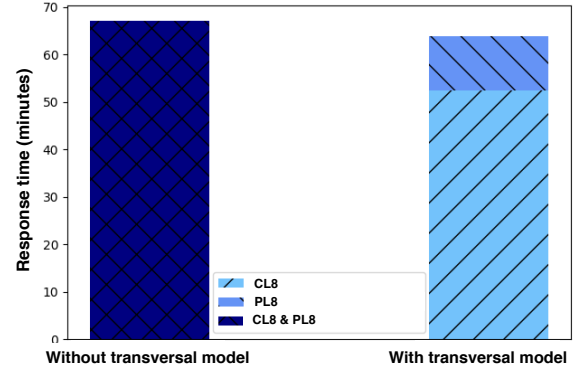


Figure 14: Response time comparison between *BigData_{TP}* service and the traditional pipeline.

model because CL8 and PL8 are executed independently, and some bottlenecks were reduced. In this context, the usage of implicit parallelism management inside a solution could even be studied to improve the performance of the resulting services, which is evaluated in case study II.

Figure 14 shows that with the traditional method, it is not feasible to identify the stages that represents bottlenecks in of PSs (where one stage begins and the other ends). Moreover, it is important to note that the solutions created with the transversal processing model are reusable and can be coupled with other existing solutions.

In addition, when using the transversal processing model, it is possible not only to modularize the PSs to identify performance differences, but also to create clones of the slow stages to improve response times. Although this could be implemented in a simple manner according to the solution design principles, this is not the main scope of the model presented in this paper.

Case study II: a big data service for processing climate data extracted from the MERRA-2 project

To conduct this case study, a solution was created to acquire, interpolate, and index climate data produced by the NASA project called MERRA-2 [27] (see Figure 15) by using the following processing applications:

1. *Acquisition (M – Acq)*. The climate data and products that the MERRA-2 project makes available through various URLs, were acquired by a crawler by using spatio-temporal parameters. The pattern in the web page is defined for the crawler to download data, which is available in the NC binary file format.
2. *Interpolation (M – Int)*. In this stage were carried out the extraction and interpolation of data contained in sets of NC files. This stage receives two elements as input: the path of the folder containing the downloaded NC files, and the path of a file that contains the geographical points (latitude, longitude) that will be used to find values in MERRA products. These points are used in the interpolation process.

3. *Formatting and indexing (M-F&I)*: In this stage, the metrics of temperature recovered by $M - Int$ from MERRA products are standardized from Fahrenheit scale to Celsius scale. The indexing process stores the all the metrics from both sources in a database service.

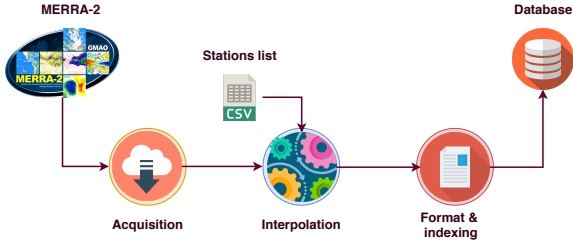


Figure 15: MERRA pipeline.

The above applications were connected as a processing pipeline, then were coupled to the following embedded services (TPS):

1. *Clustering embedded service*: This service executes a clustering algorithm (k-means) for grouping the results obtained by the PS for temperature parameters. The number of groups (value of k) in k-means is defined in the TPS by either the end-user or automatically calculated by a quality clustering validation index (silhouette index [38]).
2. *Visualization*: This service receives the temperature groups produced by the clustering service and creates a temperature map by grouping geographical points with similar values, which are placed in a map, calculated on-the-fly and on demand, depending on the spatio-temporal parameters delivered to the $BigData_{TP}$ service by the end-user.
3. *Acquisition and control of data*: This service supervises the continuous data delivery for each component in the solution. Moreover, this service controls the data distribution in scenarios where the cloning of a PS be required to solve bottleneck issues as those presented in previous case study.

The solution performs the following sequence of processes: i) retrieves data from MERRA-2 project by using $M - Adq$ task; ii) executes $M - Int$ for the interpolation process of a list of coordinates (latitude, longitude), which are obtained from either a geoportal (in production) or from a configuration file (for laboratory experiments); iii) indexes the results by using $M - F&I$ tasks, and produces data in JSON format; iv) executes the clustering of each geographical point included in the spatio-temporal parameters received in the $BigData_{TP}$ service by using the values obtained from MERRA-2, and uses these metrics for creating groups (e.g., maximum, medium temperatures or water precipitation, etc.) with statistical similarity; and v) creates the climate map for visualizing the results in a web geoportal.

In this case study, the $BigData_{TP}$ service was used on-the-fly to retrieve climate data from the MERRA-2 project

corresponding to a series of geographical points of a shape that cover the entire Mexican territory, and to create a map of climate groups depending on the metric chosen as input parameter in the $BigData_{TP}$ service.

This service deployed 32 PSs clones, this is because Mexican territory has 32 administrative regions (states, one PS per each state); as a result, the dataset was processed in parallel by the 32 PSs (see this $BigData_{TP}$ service depicted in Figure 16).

The case study considered the acquisition of meteorological data corresponding to seven days, subsequently, the interpolation process ($M - Int$) was run by the 32 TPSs to process a total of 302,099 geographical points, corresponding to all the locations of Mexico. The list of geographical points for the Mexican territory was generated by an additional task called $M-Split$, which takes as input statistical data from INEGI⁶ (National Institute of Statistics and Geography) for each location in Mexico, an identifier and geographic coordinates were extracted, generating the list of locations. This task, additionally, creates sub-lists of the geographical points per each state, which are delivered to the corresponding PS clone for processing the data found in MERRA-2 products per each geographical point.

The programming model was used to create scripts for DagOnStar and Makeflow engines, which created the original PS (a single PS) and deployed it on a containerized cloud infrastructure. The PSs created by each engine were used in the transversal prototype to create the above described $BigData_{TP}$ service (see Figure 16).

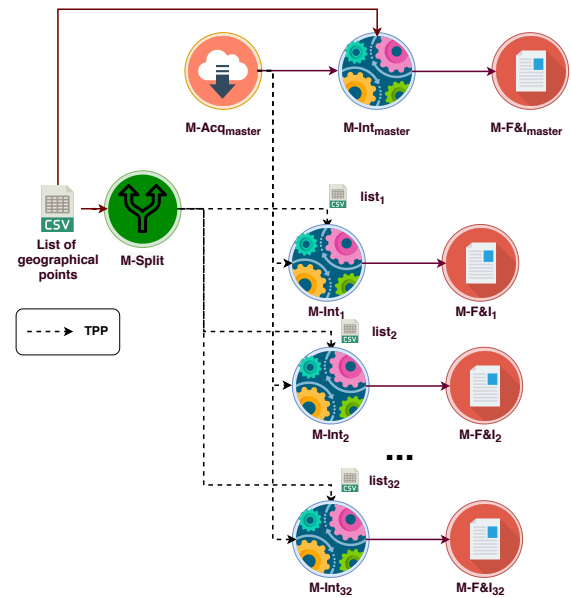


Figure 16: MERRA PSs for processing climate data from the 32 states of Mexico.

Figure 17 shows the results of the execution times produced by $BigData_{TP}$ service as a single PS created by using DagOnStar and Makeflow. As it can be seen, there is a difference in the response times (vertical axis) between DagOn-

⁶<https://www.inegi.org.mx/app/ageeml/>

Star and Makeflow (horizontal axis), which is caused by the different processing schemes used by each engine to create a PS. The experiments were executed on the same infrastructure and running the same tasks.

The times shown in Figures 18 and 19 correspond to the response times for each PS (vertical axis) obtained by Makeflow and DagOnStar respectively by following the transversal processing model when processing the data from each of 32 states of Mexico (horizontal axis). There are subtle differences in the processing times, but the performance behavior is similar for both engines, which is proportional to the number of locations that each state has and must be processed.

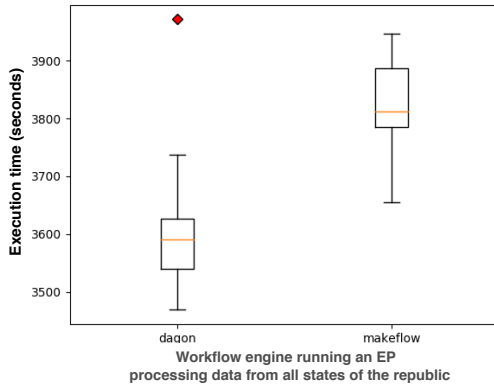


Figure 17: Master instance processing 302,099 geographic points.

When comparing the results obtained by the traditional PS with the *BigData_{TP}* service, a significant reduction in the execution times was observed for each engine. The PS created by Makeflow, processed all the geographical points in 3930 seconds (65.5 minutes), whereas the PS created by Makeflow with *BigData_{TP}* service processed the dataset in 888 seconds (14.8 minutes), which gives us a 76% improvement difference in execution times in comparison with using the original PS.

In the case of the PSs created by DagOnStar, similar results were observed. The execution time produced by a single PS was 3588 seconds (59.8 minutes), while the PSs produced by this engine deployed by using *BigData_{TP}* service was 828 seconds (13.8 minutes), which also produced an improvement difference in execution times of up to 76%.

As it can be seen, it was possible to process the same number of geographic points in a shorter time when using the *BigData_{TP}* service, in comparison with the times obtained with a single original PS. Although the purpose of the transversal model is not to improve the PS performance, it is possible for engines to generate solutions that take advantage of previously processed data in order to improve the performance of a solution by cloning PSs and execute them in overlapped manner, which also can be combined with the parallelism model of each engine to improve the performance of big data services. These experiments basically showed that the model can be used by available engines to couple mul-

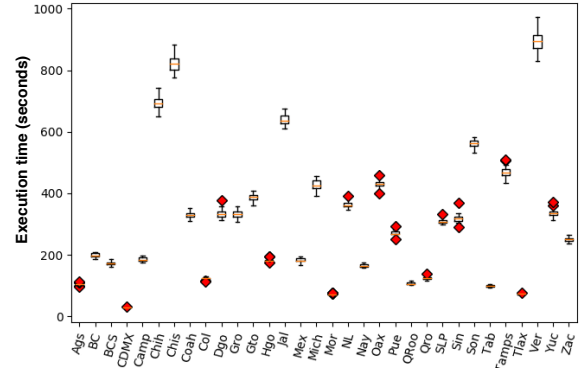


Figure 18: Thirty-two MERRA pipeline instances processing locations of each state, using Makeflow.

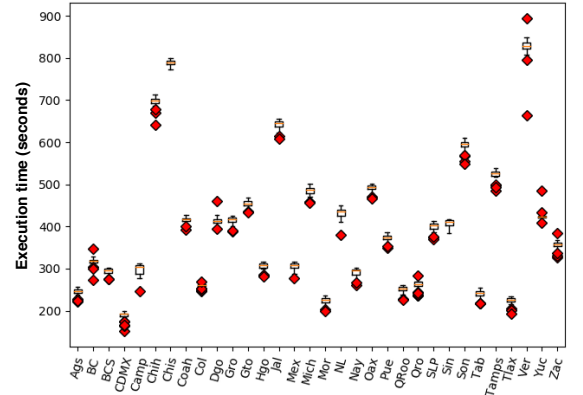


Figure 19: Thirty-two MERRA pipeline instances processing locations of each state, using DagOnStar.

tiples PSs and to extract information from different points in the final service.

At this point, we recall that the final stage of the *BigData_{TP}* service applies a clustering algorithm to the MERRA temperature values processed by the 32 PSs and produces, on demand and on-the-fly, climate maps depending on spatio-temporal parameters. Figure 20 shows the results of a distribution of geographical points throughout the entire Mexican territory that were grouped by the clustering embedded service by using a $k = 12$ parameter. This number of groups (k) was chosen according to the number of topoforms⁷ defined by INEGI for Mexico. This study enabled us to compare results produced by the *BigData_{TP}* service with topoforms identified for Mexico. Differences were observed when making this comparison for multiples geographical points, which however depend on the temporal patterns. As a result, it is required to carry out an in-depth study on the contrast between the MERRA-2 data and data obtained by ground stations, as well as a possible relation-

⁷A set of landforms associated according to some structural and/or degradative pattern or patterns into which the country has been divided, according to its geology and topography.

ship with the topoforms defined for Mexico with extensive temporal parameters, as well as reduced spatial parameters to quantify and explain these differences. This is quite feasible for end-users to perform by invoking the above evaluated *BigData_{TP}* service as many times as spatio-temporal parameters used in each execution.

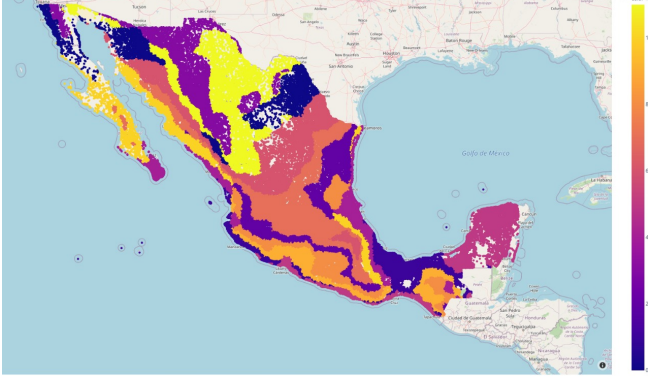


Figure 20: Grouping of locations in Mexico (12 groups) by maximum, minimum, and average temperature provided by MERRA.

Case study III: a multiple-sink crossing information service based on machine learning for the classification of air pollution values

Three datasets were processed in this case study:

- RAMA [45] (Spanish acronym for Automatic Atmospheric Monitoring Network): It contains annual databases with information about the concentrations of pollutants, recorded every hour since 1986.
- REDMET [46] (Spanish acronym for Meteorological Network and Solar Radiation): It contains information on meteorological parameters, recorded every hour since 1986.
- MERRA [27] (Modern-Era Retrospective analysis for Research and Applications): It contains databases with meteorological data generated from reanalysis processes.

Three PSs were created to process the datasets for Mexico City. These PSs were assembled recursively to create a *BigData_{TP}* service for crossing information by using the MSC method (Method 3).

This service consolidates the above described data sources into a single data source by using a TEP. This consolidation of the data sources is feasible since the sources have common fields (spatial and temporal parameters). In this way, it is possible to unify the results using the latitude and longitude data as key groups (see Figure 21).

In this case study, a list with geographic coordinates belonging to spatial records in the RAMA and REDMET stations⁸ were used as input of the crossing information *BigData_{TP}* service.

⁸<http://www.aire.cdmx.gob.mx/default.php?opc=%27ZaBhnmI=%27&dc=ZA>

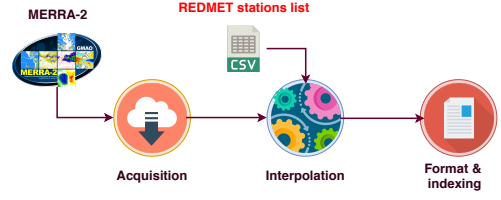


Figure 21: MERRA PS. Data interpolation using the REDMET stations.

The PSs designed for processing and monitoring data repositories (RAMA and REDMET) consider the acquisition of data from the primary source (see Figure 22). These PSs also consider a set of pre-processing services embedded in TPS, which were used for preparing, in automatic and transparent manners, the data in RAMA and REDMET monitoring datasets to adapt these data to data analysis techniques used in processing stage. These embedded services, depicted in Figure 23, include the following data cleaning and preparation services:

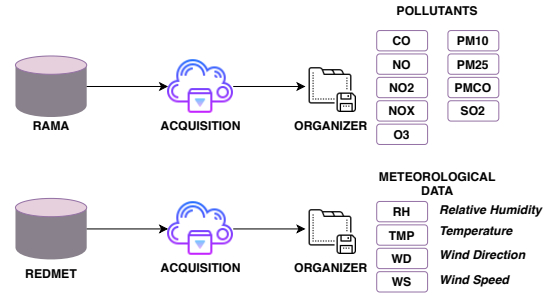


Figure 22: PS processing RAMA to obtain pollution data, and REDMET to collect meteorological data.

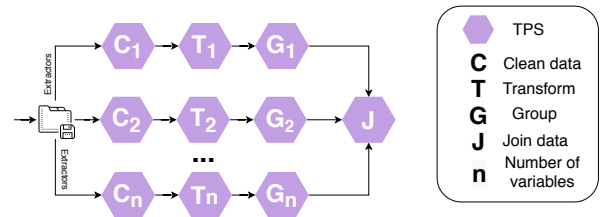


Figure 23: Post-Processing TPSs for RAMA and REDMET.

- *Cleaning data (C in Figure 23)*: This service eliminates missing or atypical data. All non-numerical data are discretized; missing values are normalized by the mode/median of the census station according to the datatype.
- *Transforming data into records (T in Figure 23)*: All data in files (each variable of data) are tables where columns represent the ground stations (RAMA and REDMET stations) deployed on Mexico City, while rows are the measurements made every hour by the stations. This task restructures this table to one with

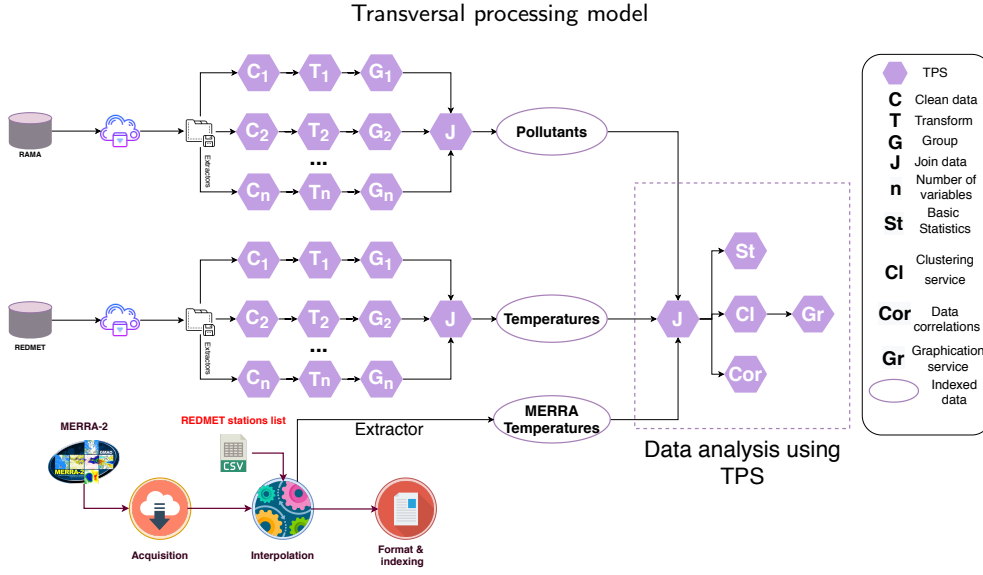


Figure 24: *BigData_{TP}* service combining the three PS built for the case study.

the columns of data, ground station, and value of the variable.

- *Grouping records (G in Figure 23)*: This service groups ground stations by calculating descriptive statistics (such as average, median, mode, standard deviation, etc.). For this case, the records associated to spatial parameters (i.e., describing the location of ground stations) and temporal (the same day but different sensing time) were described by using a descriptive statistic (median used in the experiments). Nevertheless, the statistic metrics (mode, mean, etc.) can also be selected as input parameter, and the service will use this parameter to perform this clustering process.

The PS processing the MERRA-2 repository was described in study case II, and it was reused in this *BigData_{TP}* service. This pre-processing procedure was applied to each of the variables produced by both REDMET and RAMA, to finally unify and index them into a single table (see *pollutants* for of RAMA and *temperatures* for REDMET and MERRA-2 in Figure 23).

The results produced by the solution for processing RAMA and REDMET were used as input parameters by TPSs (see data analysis in Figure 24) such as clustering, descriptive statistics and visualization/gratification, which were reused from the *BigData_{TP}* service previously described in case study II. The overall structure of the solution is shown in Figure 24.

In terms of performance, the execution time produced by processing MERRA data is three times in comparison with RAMA and REDMET, including its corresponding post and pre-processing embedded services (Figure 25). To solve this bottleneck, the *BigData_{TP}* service executes the embedded services in the TPSs for RAMA and REDMET in the *BigData_{TP}* service, which executes all these tasks in parallel.

The next step for the *manager* is to use TPS to conduct

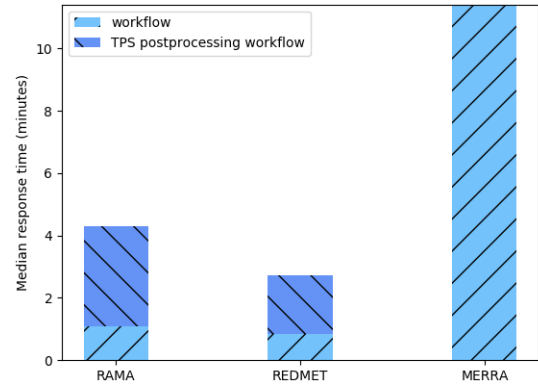


Figure 25: Execution time of the studied services.

the crossing information task, which includes embedded services for analysis and machine learning. An exploratory analysis of all the variables was performed in the data analysis, considering a date range of 2016-2020. In this analysis, the temperatures obtained by the REDMET sensors were compared with those obtained by the MERRA reanalysis, for each station and each day of sensing in the date range. An embedded service calculated a new variable called DIFF that reflects the MERRA (T2MMEAN) and REDMET (TMP) temperature difference for each day and each station. Figure 26 shows a comparative histogram between the temperatures of the REDMET and MERRA sources (DIFF is not represented in this graph).

The *BigData_{TP}* service allowed us to consume the data produced from any PS. This means the data representing the MERRA-REDMET temperature difference is used by an embedded service that executes two clustering algorithms (k-means and the hierarchical clustering algorithm), which were validated by using other embedded service executing

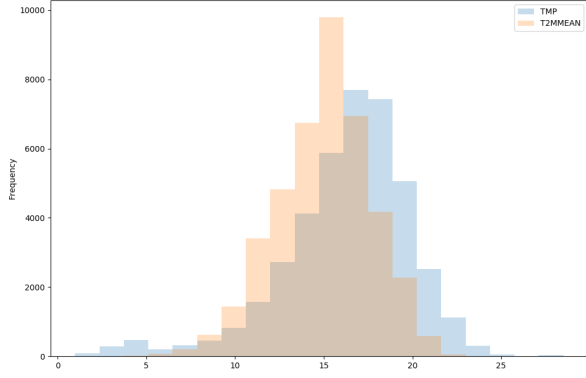


Figure 26: Histogram of T2MMEAN and TMP variables from MERRA and REDMET respectively. The values that do not fit between the histograms do not represent the real value of DIFF, but provide an overview of the different behavior between both variables.

the silhouette clustering index [38]. This validation was performed to observe how the different data sensing stations are grouped based on the value of variable in each record indexed by the TEPs (e.g., max temperature). This clustering service was used in two scenarios:

- Determining groups based on MERRA temperature, REDMET temperature, and DIFF (MERRA temperature — REDMET temperature).
- Determining groups based on all pollutants and air particles, associated to spatio-temporal parameters.

In the first scenario, $k = 2$ for both clustering algorithms yielded the highest score in the silhouette index, which means that the best number for grouping records of the studied datasets, according to the temperature, is 2. As can be seen in Figure 27, the performance of the two algorithms is measured through different k values, and the k-means algorithm yielded the best performance for $k = 2$. The resulting groups are showed in Figure 28, where a clear difference between both groups is evident. Cluster 0 is the group where the value of the difference between temperatures (DIFF) is less than or equal to zero, whereas in Cluster 1 only the INN and AJU stations produced positive DIFF values. As it can be seen, a comparative study could be performed in this *BigData_{TP}* service by using services embedded in its TPSs.

For the second scenario, the same clustering algorithms were applied to the pollution data. In order to obtain a result that can be understood graphically and without altering the data to a great extent, the selection of variables was chosen through *Principal Component Analysis* (PCA) [44]. PCA is also provided as an embedded service connected to the clustering embedded service, so it is only necessary to specify the parameters to the service to carry out this task, either by selecting a variance range (that is, selecting a number of components that maintain a specified variance) or by selecting the specified number of components. For this case, three

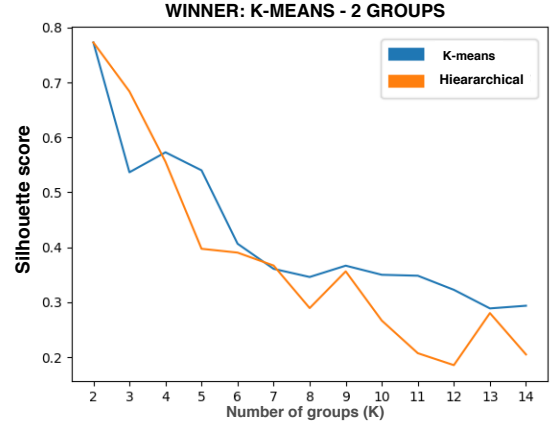


Figure 27: Silhouette scores on temperature data. Blue line corresponds to k-means, orange line corresponds to hierarchical clustering algorithm.

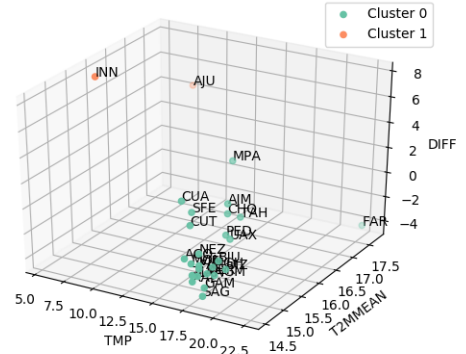


Figure 28: Temperature data clustering results.

main components were chosen among the total of pollution variables, thus maintaining a variance in the data greater than 90%.

Instead of the first scenario, the clustering algorithm with the highest performance in the silhouette index was the hierarchical clustering algorithm with a total of two clusters (Figure 29). Another notable difference is that the stations obtained in the first scenario do not appear in the same group, obtained in the second scenario (Figure 30).

At this point, the data has been grouped in automatic manner executed by the crossing information of the *BigData_{TP}* service by using spatio-temporal parameters. The next step for this service was to execute another embedded service to perform a predictive analysis. This service extracts the meteorological data obtained by the stations RAMA and REDMET as well as the verification by the MERRA data and process them by using a machine learning technique to predict pollution values.

For this study, an embedded service based on multi-layer-perceptron neural network (MLP-NN) was executed in the

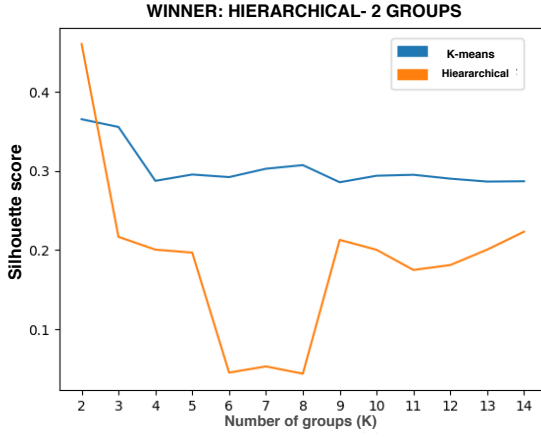


Figure 29: Silhouette scores on pollution data. Blue line corresponds to k-means, orange line corresponds to hierarchical clustering algorithm.

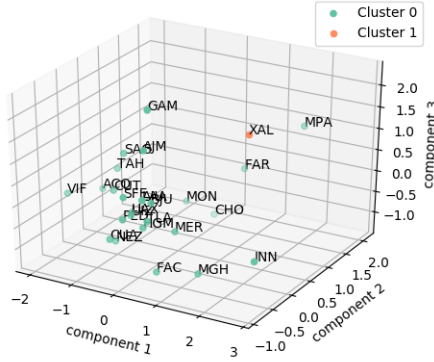


Figure 30: Pollution data clustering results.

BigData_{TP} service for the prediction of pollutants by regression. The service extracts, from the complete datasets, a third part for training the network, taking REDMET and MERRA temperatures, differential, humidity, wind speed and direction, and airborne particles (PM10 and PM25) as input parameters. When the MLP-NN was tested, it was executed by using the remaining dataset, and measuring the quality of the results with the metric R2 (R-squared) [24], a statistical measure to know how close (near to 1.0) the data is to the regression line.

Figure 31 shows the results obtained by predicting CO particles, having an acceptable variance with respect to the real data, represented with an R2 score of 0.92. On the other hand, that performance was not obtained when predicting particles such as NO (Figure 32) and NO2 (Figure 33), obtaining R2 scores in a range of 0.6 and 0.7. Nevertheless, it is possible to improve the performance of the network by configuring the parameters, the number of neurons, increasing the amount of training data or the selection of another

type of neural network in the embedded service of the TPS. These parameters can be passed through the transversal extracting/publish services. In this model, to do this type of prediction, end-users are only required for selecting parameters for the corresponding TPS, which will use it to execute the corresponding algorithm in automatic and transparent manners.

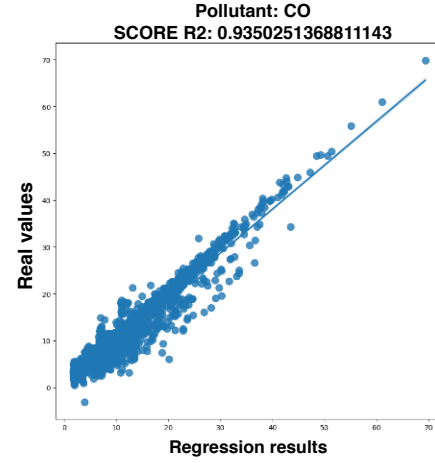


Figure 31: Regression results for CO applying a MLP-NN.

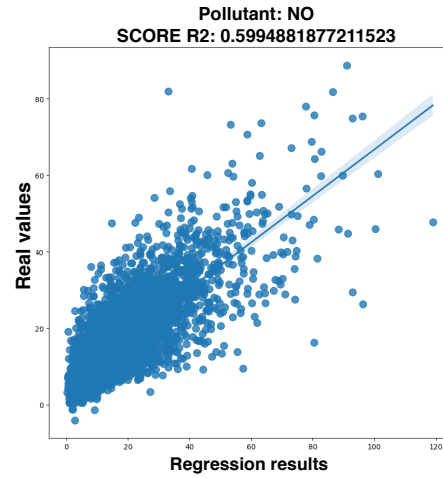


Figure 32: Regression results for NO applying a MLP-NN.

6. Conclusions

This paper presented a novel transversal processing model to build environmental big data services in fog-cloud environments. This model enables the scientific community to reuse applications and to create *BigData_{TP}* services. The transversality property of *BigData_{TP}* services is achieved by creating *Transversal Coupling Points* and *Transversal Extraction Points* through multiple existing solutions. The coupling of transversal services create virtual connections between multiple solutions, even in execution time, to create

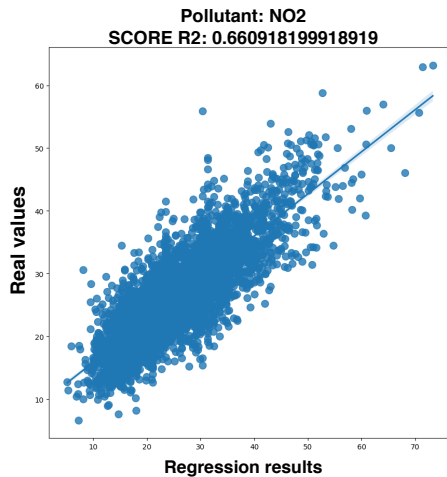


Figure 33: Regression results for NO2 applying a MLP-NN.

a new big data services. The extraction/publish transversal services create pipes for end-users/services to consume data from different points of a solution and processing the extracted data to produce information in crossing information by using analytics and machine learning services embedded in TPS of big data transversal services. In this way, applications for processing environmental products (e.g., corrections, transformations, manufacturing, coding, etc.) and/or for creating useful information by using analytics (e.g., clustering, data cleaning, ANOVA, graphing, etc.) and machine learning (neuronal networks, random forests, etc.), can be encapsulated into TPS as embedded services. The results of these services can be consumed by other applications (web pages, robots, or other services). These services also can be coupled to other PS to create *BigData_{TP}* services by following the transversal model and by using traditional frameworks and engines (rules were created for programming models of DagOnStar and Makeflow engines) for end-users to add transversality property to these engines. The *BigData_{TP}* services also enables end-users to consume both raw, cleaned, and processed data by using extracting/publish services. In this paper was showed that *BigData_{TP}* service can be built for the management of the scientific data life-cycle: from the acquisition of raw data to the preparation to the processing (by using analytics and machine learning tools) to the retrieval/searching of data (raw and prepared data as well as information) until to the crossing information and decision-making process. This processing is performed by reusing existing available applications/services/systems/tools without altering the code of these solutions. The coupling of multiple existing transversal services recursively is also feasible in this model.

A prototype was implemented based on this transversal model to create big data processing services, which were implemented for conducting three case studies based on processing environment, climate, and pollution data as well as the building of earth observation products. Crossing infor-

mation big data services were also developed by using this prototype. The experimental evaluation revealed the efficacy and flexibility of this model to create complex big data processing services by reusing multiple existing applications created with different frameworks and deployed on different IT infrastructures. It also revealed the efficacy of applying extracting/publishing transversal points to create crossing information services and information/content extraction services from different data sources.

The development of a comprehensive workflow technology aware ecosystem, enabling creatives and final users to implement widely interconnected computational pipelines and DAGs devoted to solving ever before handled computational science problems, is our vision. The short term future evolution will be the implementation of continuously running workflow tasks to enable IoT data fed workflows [22] performing on-line data processing in a routinely production-oriented fashion. Peculiar, but strategically relevant real-world applications as crowdsourced bathymetry data processing [25], can leverage on the proposed transversal model producing open data distributed on cloud infrastructures as the instrument as a service (IaaS) model [28].

7. Acknowledgement

This work has been partially supported by the project 41756 “Plataforma tecnológica para la gestión, aseguramiento, intercambio y preservación de grandes volúmenes de datos en salud y construcción de un repositorio nacional de servicios de análisis de datos de salud” by the FORDECYT-PRONACES.

References

- [1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al., 2016. Tensorflow: A system for large-scale machine learning, in: 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16), pp. 265–283.
- [2] Albrecht, M., Donnelly, P., Bui, P., Thain, D., 2012. Makeflow: A portable abstraction for data intensive computing on clusters, clouds, and grids, in: Proceedings of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies, pp. 1–13.
- [3] Attariyan, M., Flinn, J., 2010. Automating configuration troubleshooting with dynamic information flow analysis., in: OSDI, pp. 1–14.
- [4] Babuji, Y.N., Chard, K., Foster, I.T., Katz, D.S., Wilde, M., Woodard, A., Wozniak, J.M., 2018. Parsl: Scalable parallel scripting in python., in: IWSG.
- [5] Badia, R.M., Conejero, J., Diaz, C., Ejarque, J., Lezzi, D., Lordan, F., Ramon-Cortes, C., Sirvent, R., 2015. Comp superscalar, an interoperable programming framework. *SoftwareX* 3, 32–36.
- [6] Brikman, Y., 2019. Terraform: Up & Running: Writing Infrastructure as Code. O'Reilly Media.
- [7] Cabaneros, S.M., Calautit, J.K., Hughes, B.R., 2019. A review of artificial neural network models for ambient air pollution prediction. *Environmental Modelling & Software* 119, 285–304.
- [8] Cloud, A.E.C., 2011. Amazon web services. Retrieved November 9, 2011.
- [9] Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Patil, S., Su, M.H., Vahi, K., Livny, M., 2004. Pegasus: Mapping scientific workflows onto the grid, in: European Across Grids Conference, Springer. pp. 11–20.

- [10] Di Luccio, D., Benassai, G., Budillon, G., Mucerino, L., Montella, R., Pugliese Carratelli, E., 2018. Wave run-up prediction and observation in a micro-tidal beach. *Natural Hazards & Earth System Sciences* 18.
- [11] Dreher, M., Peterka, T., 2017. Decaf: Decoupled dataflows for in situ high-performance workflows doi:10.2172/1372113.
- [12] Gao, F., Yue, P., Zhang, C., Wang, M., 2019. Coupling components and services for integrated environmental modelling. *Environmental modelling & software* 118, 14–22.
- [13] Garijo, D., Corcho, O., Gil, Y., Gutman, B.A., Dinov, I.D., Thompson, P., Toga, A.W., 2014. Fragflow automated fragment detection in scientific workflows, in: 2014 IEEE 10th International Conference on e-Science, IEEE. pp. 281–289.
- [14] Goble, C.A., De Roure, D.C., 2007. myexperiment: social networking for workflow-using e-scientists, in: Proceedings of the 2nd workshop on Workflows in support of large-scale science, pp. 1–2.
- [15] Goecks, J., Nekrutenko, A., Taylor, J., Team, G., et al., 2010. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome biology* 11, R86.
- [16] Gonzalez, J.L., Perez, J.C., Sosa-Sosa, V.J., Sanchez, L.M., Bergua, B., 2015. Skydds: A resilient content delivery service based on diversified cloud storage. *Simulation Modelling Practice and Theory* 54, 64–85.
- [17] Gonzalez-Compean, J., Sosa-Sosa, V.J., Diaz-Perez, A., Carretero, J., Marcellin-Jimenez, R., 2018. Fedids: a federated cloud storage architecture and satellite image delivery service for building dependable geospatial platforms. *International Journal of Digital Earth* 11, 730–751.
- [18] Goodman, S., BenYishay, A., Lv, Z., Runfola, D., 2019. Geoquery: Integrating hpc systems and public web-based geospatial data tools. *Computers & Geosciences* 122, 103–112.
- [19] Hempelmann, N., Ehbrecht, C., Alvarez-Castro, C., Brockmann, P., Falk, W., Hoffmann, J., Kindermann, S., Koziol, B., Nangini, C., Radanovics, S., et al., 2018. Web processing service for climate impact and extreme weather event analyses. flyingpigeon (version 1.0). *Computers & Geosciences* 110, 65–72.
- [20] Hu, F., Yang, C., Schnase, J.L., Duffy, D.Q., Xu, M., Bowen, M.K., Lee, T., Song, W., 2018. Climatespark: An in-memory distributed computing framework for big climate data analytics. *Computers & geosciences* 115, 154–166.
- [21] Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M.R., Li, P., Oinn, T., 2006. Taverna: a tool for building and running workflows of services. *Nucleic acids research* 34, W729–W732.
- [22] Laccetti, G., Montella, R., Palmieri, C., Pelliccia, V., 2013. The high performance internet of things: using gvirtus to share high-end gpus with arm based cluster computing nodes, in: International Conference on Parallel Processing and Applied Mathematics, Springer. pp. 734–744.
- [23] Laster, B., 2018. Jenkins 2: Up and Running: Evolve Your Deployment Pipeline for Next Generation Automation. " O'Reilly Media, Inc."
- [24] Lewis-Beck, M.S., Skalaban, A., 1990. The r-squared: Some straight talk. *Political Analysis* 2, 153–171.
- [25] Marcellino, L., Montella, R., Kosta, S., Galletti, A., Di Luccio, D., Santopietro, V., Ruggieri, M., Lapegna, M., D'Amore, L., Laccetti, G., 2017. Using gpgpu accelerated interpolation algorithms for marine bathymetry processing with on-premises and cloud based computational resources, in: International Conference on Parallel Processing and Applied Mathematics, Springer. pp. 14–24.
- [26] Missier, P., Soiland-Reyes, S., Owen, S., Tan, W., Nenadic, A., Dunlop, I., Williams, A., Oinn, T., Goble, C., 2010. Taverna, reloaded, in: International conference on scientific and statistical database management, Springer. pp. 471–481.
- [27] Modeling, G., Office, A., 2015. Global modeling and assimilation office (gmao). merra-2 statd_2d_slv_nx: 2d, daily, aggregated statistics, single-level, assimilation, single-level diagnostics v5.12.4, greenbelt, md, usa, goddard earth sciences data and information services center (ges disc). URL: 10.5067/9SC1VNTGWV3.
- [28] Montella, R., Agrillo, G., Mastrangelo, D., Menna, M., 2008. A globus toolkit 4 based instrument service for environmental data acquisition and distribution, in: Proceedings of the third international workshop on Use of P2P, grid and agents for the development of content networks, pp. 21–28.
- [29] Montella, R., Di Luccio, D., Kosta, S., 2018. Dagon*: Executing direct acyclic graphs as parallel jobs on anything, in: 2018 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS), pp. 64–73. doi:10.1109/WORKS.2018.00012.
- [30] Montella, R., Di Luccio, D., Marcellino, L., Galletti, A., Kosta, S., Giunta, G., Foster, I., 2019. Workflow-based automatic processing for internet of floating things crowdsourced data. *Future Generation Computer Systems* 94, 103–119.
- [31] Montella, R., Di Luccio, D., Troiano, P., Riccio, A., Brizius, A., Foster, I., 2016. Wacomm: A parallel water quality community model for pollutant transport and dispersion operational predictions, in: 2016 12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), IEEE. pp. 717–724.
- [32] Montella, R., Kosta, S., Foster, I., 2018a. Dynamo: Distributed leisure yacht-carried sensor-network for atmosphere and marine data crowdsourcing applications, in: 2018 IEEE International Conference on Cloud Engineering (IC2E), IEEE. pp. 333–339.
- [33] Montella, R., Ruggieri, M., Kosta, S., 2018b. A fast, secure, reliable, and resilient data transfer framework for pervasive iot applications, in: IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE. pp. 710–715.
- [34] Muller, C., 2020. Historical background of big data in astro and geo context, in: Knowledge Discovery in Big Data from Astronomy and Earth Observation. Elsevier, pp. 21–29.
- [35] Paradis, E., 2020. A review of computer tools for prediction of ecosystems and populations: We need more open-source software. *Environmental Modelling & Software* , 104872.
- [36] Preston, B.L., Yuen, E.J., Westaway, R.M., 2011. Putting vulnerability to climate change on the map: a review of approaches, benefits, and risks. *Sustainability Science* 6, 177–202.
- [37] Reyes-Anastacio, H.G., Gonzalez-Compean, J., Sosa-Sosa, V.J., Carretero, J., Blas, J.F.G., 2020. Kulla, a container-centric construction model for building infrastructure-agnostic distributed and parallel applications. *Journal of Systems and Software* , 110665.
- [38] Rousseeuw, P.J., 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20, 53–65.
- [39] Roy, D.P., Wulder, M.A., Loveland, T.R., Woodcock, C., Allen, R.G., Anderson, M.C., Helder, D., Irons, J.R., Johnson, D.M., Kennedy, R., et al., 2014. Landsat-8: Science and product vision for terrestrial global change research. *Remote sensing of Environment* 145, 154–172.
- [40] Rydning, D.R.J.G.J., 2018. The digitization of the world from edge to core. Framingham: International Data Corporation .
- [41] Sánchez-Gallegos, D.D., Di Luccio, D., Gonzalez-Compean, J., Montella, R., 2019. A microservice-based building block approach for scientific workflow engines: Processing large data volumes with dagonstar, in: 2019 15th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), IEEE. pp. 368–375.
- [42] Sánchez-Gallegos, D.D., Di Luccio, D., Kosta, S., Gonzalez-Compean, J., Montella, R., 2021. An efficient pattern-based approach for workflow supporting large-scale science: The dagonstar experience. *Future Generation Computer Systems* 122, 187–203.
- [43] Schnase, J.L., Duffy, D.Q., Tamkin, G.S., Nadeau, D., Thompson, J.H., Grieg, C.M., McInerney, M.A., Webster, W.P., 2017. Merra analytic services: Meeting the big data challenges of climate science through cloud-enabled climate analytics-as-a-service. *Computers, Environment and Urban Systems* 61, 198–211.
- [44] Schölkopf, B., Smola, A., Müller, K.R., 1997. Kernel principal component analysis, in: International conference on artificial neural networks, Springer. pp. 583–588.
- [45] SEDEMA, 2012a. Red automática de monitoreo atmosférico (rama), accessed: June 2020.

- [46] SEDEMA, 2012b. Red de meteorología y radiación solar (redmet), accessed: June 2020.
- [47] Smart, J.F., 2011. Jenkins: The Definitive Guide: Continuous Integration for the Masses. " O'Reilly Media, Inc."
- [48] Sun, Z., Di, L., Gaigalas, J., 2019. Suis: Simplify the use of geospatial web services in environmental modelling. *Environmental Modelling & Software* 119, 228–241.
- [49] Tarboton, D.G., Idaszak, R., Horsburgh, J.S., Heard, J., Ames, D., Goodall, J.L., Band, L., Merwade, V., Couch, A., Arrigo, J., et al., 2014. Hydroshare: advancing collaboration through hydrologic data and model sharing .
- [50] Tejedor, E., Becerra, Y., Alomar, G., Queral, A., Badia, R.M., Torres, J., Cortes, T., Labarta, J., 2017. Pycompss: Parallel computational workflows in python. *The International Journal of High Performance Computing Applications* 31, 66–82. URL: <https://doi.org/10.1177/1094342015594678>, doi:10.1177/1094342015594678.
- [51] Ujjwal, K., Garg, S., Hilton, J., Aryal, J., 2020. A cloud-based framework for sensitivity analysis of natural hazard models. *Environmental Modelling & Software* 134, 104800.
- [52] Vaghefi, S.A., Abbaspour, N., Kamali, B., Abbaspour, K.C., 2017. A toolkit for climate change analysis and pattern recognition for extreme weather conditions—case study: California-baja california peninsula. *Environmental modelling & software* 96, 181–198.
- [53] Vitolo, C., Elkhatib, Y., Reusser, D., Macleod, C.J., Buytaert, W., 2015. Web technologies for environmental big data. *Environmental Modelling & Software* 63, 185–198.
- [54] Xiang, Z., Demir, I., 2020. Distributed long-term hourly streamflow predictions using deep learning—a case study for state of iowa. *Environmental Modelling & Software* , 104761.
- [55] Xue, Z., Couch, A., Tarboton, D., 2019. Map based discovery of hydrologic data in the hydroshare collaboration environment. *Environmental Modelling & Software* 111, 24–33.
- [56] Yi, H., Idaszak, R., Stealey, M., Calloway, C., Couch, A.L., Tarboton, D.G., 2018. Advancing distributed data management for the hydroshare hydrologic information system. *Environmental Modelling & Software* 102, 233–240.
- [57] Yildiz, O., Ejarque, J., Chan, H., Sankaranarayanan, S., Badia, R.M., Peterka, T., 2019. Heterogeneous hierarchical workflow composition. *Computing in Science & Engineering* 21, 76–86.
- [58] Yue, P., Zhang, M., Tan, Z., 2015. A geoprocessing workflow system for environmental monitoring and integrated modelling. *Environmental Modelling & Software* 69, 128–140.
- [59] Zhang, F., Chen, M., Ames, D.P., Shen, C., Yue, S., Wen, Y., Lü, G., 2019. Design and development of a service-oriented wrapper system for sharing and reusing distributed geoanalysis models on the web. *Environmental Modelling & Software* 111, 498–509.
- [60] Zhang, M., Jiang, L., Yue, P., Gong, J., 2020. Interoperable web sharing of environmental models using ogc web processing service and open modeling interface (openmi). *Environmental Modelling & Software* 133, 104838.
- [61] Zhou, Z., Wen, J., Wang, Y., Xue, X., Hung, P.C., Nguyen, L.D., 2020. Topic-based crossing-workflow fragment discovery. *Future Generation Computer Systems* .