CloudMeasure: A Platform for Performance Analysis of Cloud Computing Systems

Luis Eduardo Bautista Villalpando
Department of Electronic Systems
Autonomous University of Aguascalientes
Aguascalientes, Mexico
E-mail: lebautis@correo.uaa.mx

Alain April
Department of Software Engineering and Information Technology
ETS – University of Quebec
Montreal, Canada
E-mail: alain.april@etsmtl.ca

Alain Abran
Department of Software Engineering and Information Technology
ETS – University of Quebec
Montreal, Canada
E-mail: alain.abran@etsmtl.ca

***Abstract*— Concepts such as price, performance, time to completion (availability), probability of failure and non-repudiation are key to being able to produce a comparison service, in order to establish Service Level Agreements (SLA) or design better mechanisms to improve the performance in Could Computing Systems (CCS). This work presents the design of a platform for performance analysis, which provides infrastructure, a framework for performance measurement and tools to facilitate the design, validation, and comparison of performance models and algorithms for CCS. The purpose of this platform is to help to establish attribute–performance relationships relating to specific applications with relatively well-known demands on systems to be able to determine how comparison services may be formulated. The design of the CloudMeasure platform is based on a framework for implementing big data science in organizations (DIPAR) and the three-dimensional performance measurement model for CCS which defines the basis for the analysis of Cloud Computing concepts that are directly related to performance and have been identified from international standards such as ISO 25010.**

Keywords— cloud computing; performance; analysis; model; platform; framework; ISO 25010 quality model, maintenance

I. INTRODUCTION

Cloud Computing (CC) is defined by ISO and IEC as the paradigm for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable cloud resources accessed through services which can be rapidly provisioned and released with minimal management effort or service provider interaction [1].

Cloud services are categorized into three service models: 1) Infrastructure as a Service (IaaS), 2) Platform as a Service (PaaS), and 3) Software as a Service (SaaS) [2]. These three service models include all the technical resources that clouds need in order to process information, such as: software, hardware, and network elements. For example, the service model that relates the most to the software engineering community is the SaaS model while the IaaS model is most related to hardware architectures and virtualization. Software engineers focus on software components, and customers use IT provider's applications running on a cloud infrastructure to process information according to their processing and storage requirements. One of the main characteristics of IaaS model is that customers do not manage or control the underlying Cloud infrastructure (including network, servers, operating systems, and storage), except for limited user-specific application configuration settings.

One of the most important challenges in delivering Cloud Services is to ensure that they are fault tolerant and minimize anomalies which can degrade its services or impact their quality, and even their availability. According to Coulouris [3], a failure occurs in a distributed system (DS), like a CC system (CCS), when a process or a communication channel departs from what is considered to be its normal or desired behavior. An anomaly is different, in that it slows down a part of a CCS without making it fail completely, impacting the performance of tasks within nodes, and, consequently, of the system itself.

Performance analysis models (PAM) for CCS must propose a means to identify and quantify "normal cluster behavior," which can serve as a baseline for detecting possible failures and anomalies in the computers (i.e. nodes in a cluster) that may impact the overall cloud performance. To achieve this goal, measurement methods are needed to collect the necessary base measures specific to CCS performance, and analysis models must be designed to determine the relationships that exist among these measures.

The ISO International Vocabulary of Metrology (VIM) [4] defines a measurement method as a generic description of a logical organization of operations used in measurement, and an analysis model as an algorithm or calculation combining one or more measures obtained from a measurement method to produce evaluations or estimates relevant to the information needed for decision making.

An important aspect in the creation of the above models is data requirements. Data is necessary to carry on experiments, simulating different scenarios, in order to select the best models that fit our requirement. Therefore, it is necessary to have access to performance data repositories, to help figuring out performance models that subsequently can be implemented in live CCS. In addition, it is necessary

measurement frameworks which help during the design process of PAM for CCS. Unfortunately, currently there are no CC performance platforms that help in the design and evaluation of such models.

This paper presents the CloudMeasure project, which aims to develop and make available a public platform for the performance analysis of CCS. The purpose of this research project is to provide a framework, data, and infrastructure to facilitate the design, validation, and comparison of performance analysis models and algorithms for CCS. One of the most important aspects of this project is the integration of the above components that constitute the CloudMeasure platform. This integration is very important because this will determine the efficiency and reliability of the results obtained from the PAM. In addition, such integration will help to define the group of CCS elements and the attributes that determine the performance of CCS. For instance, one group of elements could be data related to applications running on the CCS such as the *Job history*, and some of their attributes could be the *number of success tasks executed* or the *time taken to process them*.

This paper is structured as follows. Section 2 presents related work on platforms for data analysis in software engineering and Cloud Computing. Section 3 presents the proposed CloudMeasure platform architecture, which describes the elements in which is based on and its operation. Section 4 presents the CloudMeasure DataFlow (CMDF), which defines the steps to follow for designing, developing and validating performance analysis models for CCS. Finally, section 5 summarizes the contributions of this work and suggests future work.

II. Related Work

A. Software Project Platforms and Repositories

The CloudMeasure project provides a conceptual model for the design of PAM as well as the definition of a standard format for performance attributes of CCS and tools for data analysis. In addition, the project offers a collaborative data loading mechanism that will address different participant's role (for example: performance data providers, PAM contributors and infrastructure users).

There are several data repositories made available publicly to share information related to software projects, web systems or data sets of failures traces. For example, The International Software Benchmarking Standard Group (ISBSG) [5] maintains the largest publicly available repository of software projects. The ISBSG mission is to help to improve software engineering practices and the business management of information technology resources through the provision of project and application data.

Other platforms such as the Web Metrics repository [6], defines a catalogue of web metrics which allows evaluators and stakeholders to have a service and a consultation mechanism to support different phases of the development of the software life cycle in web development. This metrics repository is used to support different quality assurance process such as non-functional requirements definition and specification, metrics understanding and selection, quality testing definition, development or maintenance phase. In addition, the Web Metrics repository provides tools to data collection and cataloguing to analyze the web system and improve its performance.

Other data platforms record failures traces to allow the comparison and cross validation of a fault-tolerant model or algorithms across identical trace data sets. The Failure Trace Archive (FTA) [7], defines a standard format for failure traces and provides a toolbox that facilities comparative trace analysis for its users. A failure trace is a data set collected from different distributed systems, which contains records of their availability represented in time series. The FTA repository presents uniform statistical analysis and failure models for nine diverse distributed systems. According to Kondo, this can be used to assess if a model can be generalized to another context (in terms of reliability or user base, for example) or to determine which trace data set is better suited or applicable for a given algorithm or model. Although the above projects present interesting design of data repositories, none of them cover Cloud Computing technologies.

B. Performance Measurement Framework for Cloud Computing

The Performance Measurement Framework for Cloud Computing (PMFCC) [8] integrates software quality concepts from ISO 25010. This framework defines that CCS performance is determined by two main sub concepts: 1) performance efficiency, and 2) reliability. In addition, it establishes that when a CCS receives a service request, there are three possible outcomes (the service is performed correctly, the service is performed incorrectly, or the service cannot be performed). The outcome will determine the sub concepts that will be applied for performance measurement.

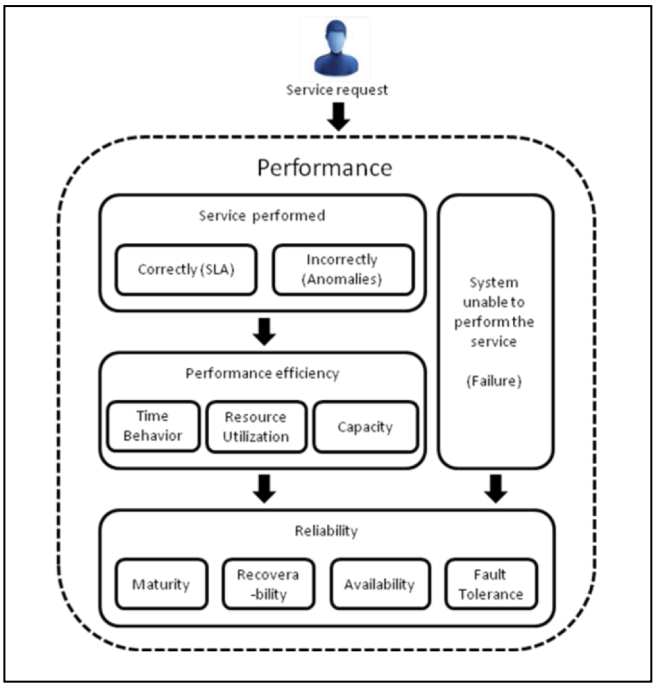

Fig. 1. Quality concepts and sub concepts associated to the performance measurement of CCS.

For example, suppose that the CCS performs a service correctly, but, during its execution, the service failed and was later reinstated. Although the service was ultimately performed successfully, it is clear that the system availability (part of the reliability sub concept) was compromised, and this affected CCS performance. Figure 1 presents the quality concepts and sub concepts associated to the performance measurement of CCS.

The performance measurement framework does not define the type of data format related to the CCS. The framework only defines the concepts that best represent the type of attributes, and which can be measured to assess whether or not the CCS satisfies the stated requirements from a quantitative viewpoint. These types of attributes are grouped into such performance concepts, which are responsible for conducting the measurement process using a combination of base measures through a data collector. They are associated with the corresponding ISO 25010 quality derived measures, as presented in Table I.

The types of attribute presented in Table I are categorized as performance concepts. These concepts were designed to shares intermediate results from common performance measures, reducing the number of operations in the measurement process at the time of calculation.

In addition, the framework determines how to measure a quality characteristic, for example how can we measure the CCS availability characteristic (presented in Table I) using the framework? To start with, three performance concepts are needed: 1) the time concept, 2) the task concept, and 3) the transmission concept. The time concept can use several different measured attributes, such as *CPU utilization* by the user, *job duration*, and *response time*. These measures are obtained using a data collector, and then inputted to the time concept that calculates a derived measure of the time. The combination of results of each concept determines a derived measure of the availability that contributes to CCS performance, as defined in the framework.

TABLE I. FUNCTIONS ASSOCIATED WITH CLOUD COMPUTING PERFORMANCE CONCEPTS

Attribute Type	***Performance Concept***	***ISO 25010 Quality Characteristic***
Failures avoided Failures detected Failures predicted Failures resolved	Failure concept	Maturity Resource utilization Fault tolerance
Breakdowns Faults corrected Faults detected Faults predicted	Fault concept	Maturity Fault tolerance
Tasks entered into recovery Tasks executed Tasks passed Tasks restarted Tasks restored Tasks successfully restored	Task concept	Availability Capacity Maturity Fault tolerance Resource utilization Time behavior
Continuous resource utilization time Down time Maximum response time Observation time Operation time Recovery time Repair time Response time Task time Time I/O devices occupied Transmission response time Turnaround time	Time concept	Availability Capacity Maturity Recoverability Resource utilization Time behavior
Transmission errors Transmission capacity Transmission ratio	Transmission concept	Availability Capacity Maturity Recoverability Resource utilization Time behavior

On important aspect of the PMFCC is that the type of attribute only defines the group in which performance data will be classified. Performance data most of the time comes from a number of sources, such as application logs, database logs, monitoring system tools, etc. This makes very difficult to know what type of data will be ingested and then used in the PAM. One of the main problems that arises following the ingestion of performance data is its cleanliness. This problem calls for the quality of the data to be verified prior to performing the performance analysis. Among the most important data quality issues to consider during data cleaning are corrupted records, inaccurate content, missing values, and formatting inconsistencies, to name a few.

The quality of the performance data can affect the results of the PAM and as consequence the decision-making process in different stages such as definition of organization's requirements, processes evaluation, aspects of interoperability and design of applications only to name a few.

III. THE CLOUD MEASURE PLATFORM

The CloudMeasure project proposes a platform for the design, development and validation of PAM for CCS. The purpose of this project is to provide a PMFCC, performance data sets as well as infrastructure to facilitate the design, validation, and comparison of performance models and algorithms for CCS. One of the main reasons for the creation of the CloudMeasure project is the current lack of information that can help in understanding and defining how to measure availability, reliability and non-repudiation of CCS. Actual measurement of concepts such as price, performance, time to completion (availability), likelihood of completion (probability of failure) and penalty (non-repudiation) are key to being able to compare services or to establish Service Level Agreements (SLA) for CCS.

According to Li [9], commercial CCS currently enable the capture of price–performance information relating to specific applications with relatively well-known demand patterns. This allows the user to gain useful information to compare the service between suppliers. Comparisons can become complex as they can depend on both the performance requirements of the user, the current availability of the system, as well as the price the user can afford. According to Gangadharan [10], the pricing of Cloud Computing services is currently associated with differentiated levels of service based on varying capacity of memory, computing units used, and types of platforms. The

pricing also varies with respect to the choice of operating systems and the geographical location of the user. The criteria for pricing of Cloud Services can also be based on hourly usage, CPU cycle usage, or other usage approach. In addition Gangadharan mentions that pricing of infrastructure Cloud Services depends upon levels of use, layers of service, or a mix of these. Thus, the CloudMeasure project could provide a useful tool for maintainers, users and developers to help to define the performance data which allows to create performance models to gain knowledge that can contributes to understand an SLA and help them to analyze the performance of CCS.

In particular, the CloudMeasure platform contains the following:

- A PMFCC which defines attributes, concepts and a measurement method for the performance analysis of CCS.
- Performance data sets from CCS, differing in scale and granularity, which contributes to create models to analyze concepts as availability, non-repudiation, capacity, etc.
- Infrastructure consisting of a cluster of computers running Hadoop technology, which can be used for developing and testing PAM.
- The CloudMeasure DataFlow (CMDF), which defines the workflow for designing and validating performance analysis models for CCS.

A. The CloudMeasure Conceptual Mode

The CloudMeasure platform is based on a conceptual model, which defines and describes the performance data involved in the performance analysis of CCS. This allows to document each measure providing, as a result, a comprehensive catalogue of attributes than can be used by different stakeholders. The conceptual model is organized hierarchically (see Figure 2) and includes:

- **Performance data design**: Defines the type of data related to performance, which is baseline of performance platform. This stage ensures the data quality which is the basis for the creation of datasets and analysis models.
- **Performance data collection**: Describes the source and procedure for the data collection process. This stage identifies the type of CC architecture, platform, and application development frameworks and so on.
- **Performance data analysis**: Among the most important data quality issues to consider during the data analysis stage is data cleaning which refers to corrupted records, inaccurate content, missing values, and formatting inconsistencies, to name a few. An important issue in data integration is formatting inconsistencies; caused by the very different forms that data can take as result of data collection from different architectures, platforms and applications, and their combination in order to figure out relationships.
- **Development of performance analysis models**: Once the performance data have been integrated and analyzed, it is necessary to develop models, which can be used during the decision-making process by stakeholders. Such models will help to understand the behavior and performance of CCS.

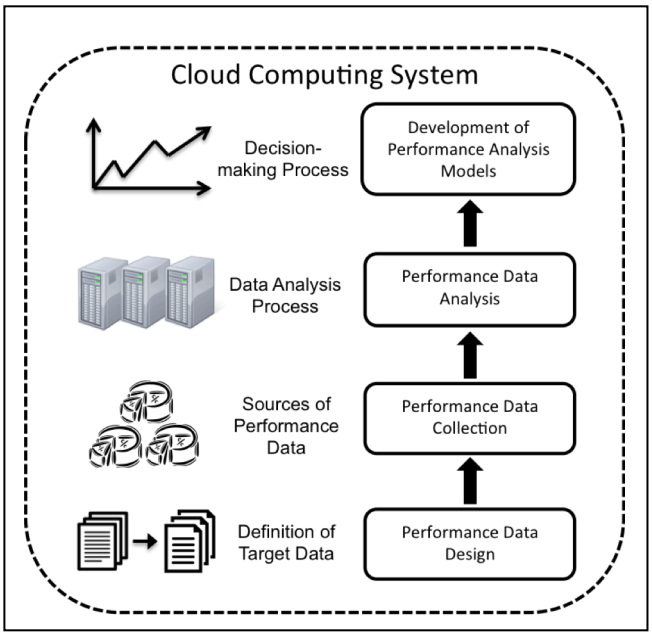

Fig. 2. Cloud Measure conceptual model.

One important aspects of the CloudMeasure platform is that users can query specific performance data to create models or simply analyze content into the data analysis infrastructure. To facilitate the design of the platform, a template was defined to build a catalogue of performance data that allows populating and updating a data performance repository.

B. The CloudMeasure Template Data Content

Performance data quality is a key component of the CloudMeasure platform. Enhancing the quality of performance data is important for a number of reasons, including:

- The existence of defective data will produce erroneous results in PAM, contributing to an unsatisfactory decision-making process.
- The dispersion of performance data among different sources of CCS, such as hypervisors, individual virtual machine (VM), VM scheduling information across multiple hardware cores, etc., does not provides a coherent and integrated vision for the analysis of the performance of CCS
- The co-existence of legacy architectures, cloud platforms and frameworks for development of applications are normally gathered under different standards.

One of the goals of the CloudMeasure platform is to improve the quality of data performance regardless of cloud architecture, platform or application development framework used in CCS. This improvement is possible by identifying the data quality requirements related to performance, which help ensuring the quality of the information used in the design and validation of PAM for CCS.

The CloudMeasure data platform is based on ISO 25012: Software Engineering: Software Product Quality Requirements and Evaluation (SQuaRE) – Data Quality Model [11]. This standard defines a data quality model, which focuses on the quality of the data as part of a computer system and defines quality characteristics for target data used users and operators of the system. According to ISO 25012, target data are those that the organization decides to be analyzed and validated through the model. In this case target data is related to CCS performance. Moreover, ISO 25012 describes characteristics that may be used as a term of reference to define data quality requirements and data quality measures to establish criteria for the assessment and evaluation of the quality of the data managed by a computer system according to organization's objectives

One of the main aspects considered for the design of a standard template is to categorize each attribute to be measured into the different ISO quality concepts described in the PMFCC (see Figure 1). For this, we are working on the design of a *Three-Dimensional Performance Measurement Model for Cloud Computing* (P2M2C-3D) which defines data types and a measurement method that makes it possible to measure the CC concepts that are directly related to performance from different perspectives.

C. A three-dimensional Performance Measurement Model for Cloud Computing

In the Three-Dimensional Performance Measurement Model for Cloud Computing (P2M2C-3D), performance concepts have been reviewed and updated according to different type of perspectives as shown in Table II where the different sub concept measurements are defined according to each perspective.

For example, the measurement of the sub concept of time behavior from perspective number one ($TBMP_1$) corresponds to the provider, while the same type of measurement from perspective number two ($TBMP_2$) corresponds to the costumer perspective, and so on.

Each group of the sub concept measurements is combined according to the same perspective number in order to obtain its respective concept indicator (see Figure 3). For example, the measurement of the sub concepts, MMP_1, RMP_1, AMP_1 and $FTMP_1$ are combined to obtain the indicator for the reliability concept (RI_1). Similarly, the measurement of the sub concepts $TBMP_1$, $RUMP_1$ and CMP_1 are combined to obtain the indicator of the performance efficiency concept (PEI_1). Finally, the indicators related to reliability and performance efficiency concepts, are integrated to obtain a Key Performance Indicator (KPI_1) from the perspective number one or also called provider perspective.

TABLE II. PERSPECTIVES FOR PERFORMANCE MEASUREMENT CONCEPTS IN CLOUD COMPUTING

Performance Measurement Concept	*Sub concept measurement*	*Description*
Performance efficiency		
	$TBMP_1$	Time behavior measurement - provider perspective
	$TBMP_2$	Time behavior measurement - costumer perspective
	$TBMP_3$	Time behavior measurement - user perspective
	$RUMP_1$	Resource utilization measurement - provider perspective
	$RUMP_2$	Resource utilization measurement - costumer perspective
	$RUMP_3$	Resource utilization measurement - user perspective
	CMP_1	Capacity measurement - provider perspective
	CMP_2	Capacity measurement - costumer perspective
	CMP_3	Capacity measurement - user perspective
Reliability		
	MMP_1	Maturity measurement - provider perspective
	MMP_2	Maturity measurement – costumer perspective
	MMP_3	Maturity measurement - user perspective
	AMP_1	Availability measurement - provider perspective
	AMP_2	Availability measurement - costumer perspective
	AMP_3	Availability measurement - user perspective
	$FTMP_1$	Fault tolerance measurement - provider perspective
	$FTMP_2$	Fault tolerance measurement - costumer perspective
	$FTMP_3$	Fault tolerance measurement - user perspective
	RMP_1	Recoverability measurement from provider perspective
	RMP_2	Recoverability measurement from costumer perspective
	RMP_3	Recoverability measurement from user perspective

The different key performance indicators values (kpi_1, kpi_2 and kpi_3) shown in Figure 4, represent the values of the three dimensions, each placed on its tetrahedron side, describing a sloped plane section (or hyper plane) in the space returning a quantitative assessment. The key indicators KPI_1, KPI_2 and KPI_3, represent the different dimensions of the regular tetrahedron according to the perspectives of provider, customer and user.

IV. THE CLOUDMEASURE DATAFLOW

The CloudMeasure DataFlow (CMDF) is the result of designing a platform for performance data collection, data analysis, analysis models processing and delivering of results, designed to organize the diverse and complex dataflow generated by different measures which contributes to the performance of CCS. The CMDF is based on the

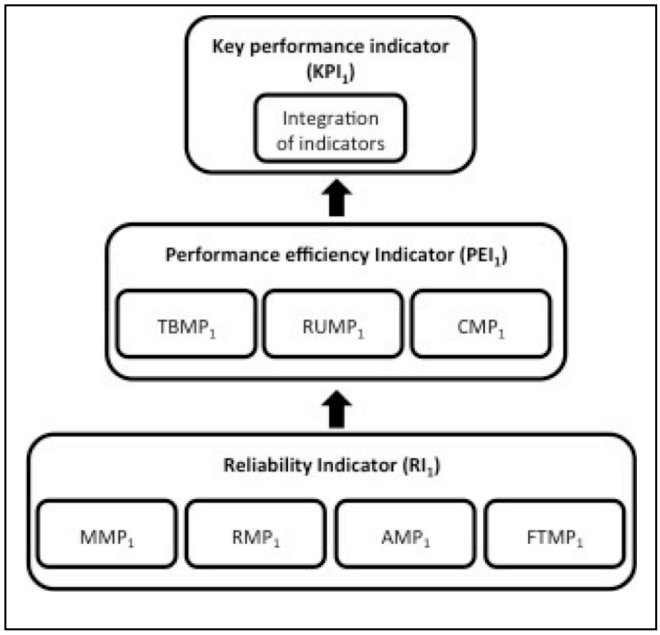

Fig. 3. Key Performance Indicator from the provider perspective.

Framework for Implementing Big Data Science in Organizations (DIPAR) [12].

A. *DIPAR: A Framework for Implementing Big Data Science in Organizations*

The DIPAR framework proposes a means to implement Big Data Science (BDS) in organizations, and defines its requirements and elements. The framework consists of five stages: Define, Ingest, Preprocess, Analyze, and Report, and is based on the ISO 15939 Systems and software engineering – Measurement process standard [13], the purpose of which is to collect, analyze, and report data relating to products to be developed.

The DIPAR framework integrates the four activities described in ISO 15939, and its main objective is to design Big Data products that have a high impact on organizational performance. Figure 5 depicts the five stages to be executed during the implementation of the DIPAR framework, as well as the order in which should be executed.

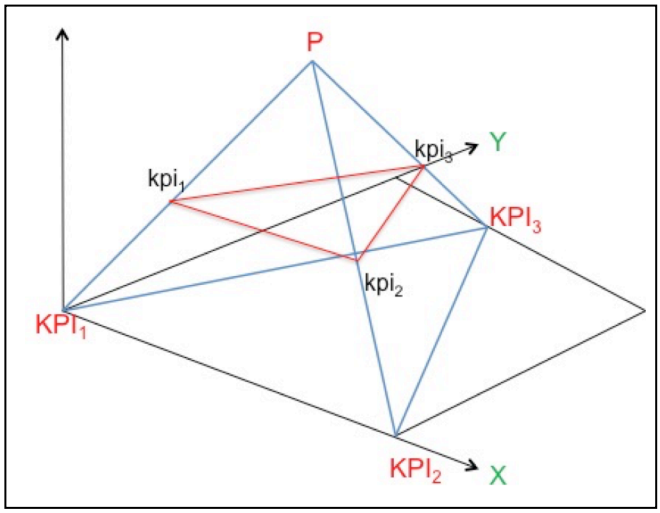

Fig. 4. Regular tetrahedron with KPI1, KPI2 and KPI3 dimensions which represent the perspectives of provider, costumer and user.

Each stage of the DIPAR framework and the elements they involve are:

- **Define:** The first step in the DIPAR framework is to define whether or not a new product is necessary. If it is not, all the analytical work developed to create the product will be a waste of time and resources. For this, it is necessary to define the data needed to develop the product such as a new PAM for our case study.
- **Ingest:** One of the main challenges of ingesting the system is to define the ingestion sources, because most of the time data come from a number of sources, such as Web logs, databases, different types of applications, etc. This makes very difficult to know what type of data will be ingested by the system. One solution to this problem is to use Big Data (BD) software that is designed specifically to collect and aggregate data from different sources. Projects like Flume [14] and Scribe [15] allow large amounts of log data to be collected, aggregated, and moved from many different sources to a centralized data store.
- **Processing:** One of the main problems that arises following the ingestion of a system is the cleanliness of data. This problem calls for the quality of the data to be verified prior to performing data analysis. Consequently, one of the main challenges at the preprocessing stage is how to structure data in standard formats so that they can be analyzed more efficiently. This is often easier said than done: during the process of structuring and merging data into common formats, there is a risk of losing valuable information.
- **Analysis:** Once the data have been preprocessed, they are analyzed to obtain relevant results. For this, it is necessary to develop models that can be used in the creation of new products. One of the main problems arising during the design of such models is to recognize which of the available data are the most relevant to an analysis task. Once it becomes feasible to develop complex models and algorithms for data analysis, it is possible to create products with added value for the organization.
- **Report:** Once data are ingested, processed, and analyzed, users need to be able to access and evaluate the results, which must be presented in such a way that they are readily understood. Here, analysis model results are evaluated as part of decision-making process in order to arrive to relevant conclusions or design new analysis models.

B. *CloudMeasure DataFlow*

CloudMeasure DataFlow (CMDF) was designed to define the workflow to be performed during the process of

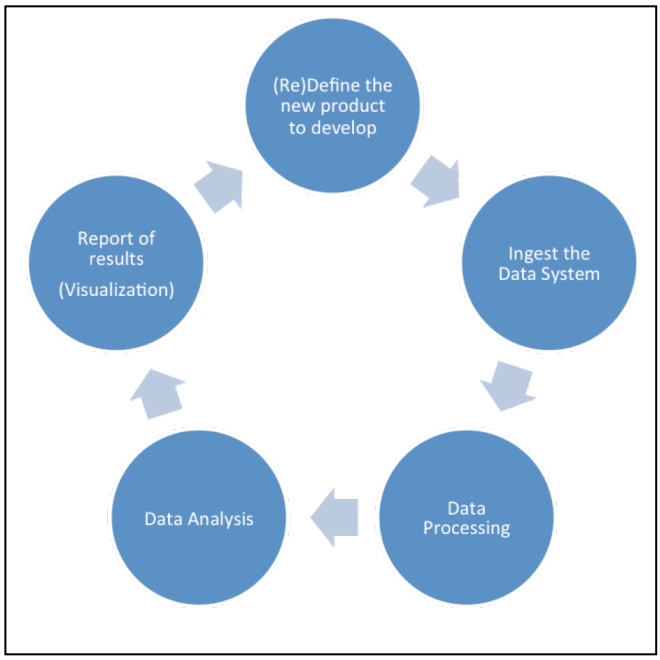

Fig. 5 Stages to develop during implementation of the DIPAR framework.

the design, development and validation of PAM for CCS. One of the objectives of the CMDF is to accelerate time of definition of performance data, data collection, processing, analysis and report of results, from different data sources over a geographically disperse network. Figure 6 shows the stages and elements that constitute the CMDF process.

At the beginning of the flow, they are found different CCS (CCS_A, CCS_B and CCS_C) from which are extracted performance measures that are the baseline for the design and development of PAM. The extraction of these measures is performed by means of different Application Program Interfaces (API) that allow the extraction of data to the CloudMeasure platform. It is important to mention that such API's are developed for third-party entities such as providers of CCS.

The first stage in the CMDF is the *performance data definition*. In this stage, performance data measures extracted from the different CCS providers are organized according to the different performance concepts and sub concepts defined in the P2M2C-3D (see Table II). For example, the performance measure *CPU utilization* is categorized as a measure of the *Resource Utilization* sub concept from one of the different perspectives. The categorization of all extracted performance measures is necessary in order to design, development and validate different PAM.

Once that performance data is defined, the next step is the storage of this, which is performed in the *data collection* stage. Here, data is stored in some type of distributed storage as the Hadoop Distributed File System (HDFS) [16] or some type of NoSQL data base such as Hbase [17]. This type of storage is very important in order to reduce costs of data processing.

After data collection is performed, the next stage is *data processing*, which consists in verifying the cleanliness of data; this means that quality of the data needs to be verified prior to performing data analysis. As it was mentioned, one of the main challenges at this stage is how to structure data in standard formats so that they can be analyzed more efficiently.

Once the quality data is verified, the next step is the design, development and validation of PAM - *data analysis stage*. For this, statistical methods and machine learning techniques are used in order to analyze the performance of CCS. This is done using data processing tools which are available in the CloudMeasure platform such as MapReduce [18], Hive [19] or Mahout [20].

Finally, the last stage is the *report of the results* obtained from the previously designed PAM. These results will be useful during creating SLA's, for the design and development of new analysis models or improving the performance of CCS. It is important to mention that the CMDF is an iterative process which is oriented to improve the quality of data and results. That is, each stage depends on the previous and as consequence, sometimes it will be necessary to go back to some specific stage in order to ensure the quality of the data input and output.

V. Conclusions and Future Work

The CloudMeasure platform is a research project developed to contribute to the design, development and validation of PAM for CCS. This project proposes an approach to support the analysis of the performance characteristics of CCS during the development, maintenance and operational stages.

Using the information contained in this platform will allow end users to better understand and compare the relationship between performance attributes of different CCS. Moreover, this platform will help during the design, validation and comparison of PAM (and algorithms), which can be used in the design of SLA's.

One of the main issues in formulating SLA is how to capture price–performance information relating to specific applications with relatively well-known demands on systems to be able to determine how such a comparison service may be formulated. In this way, the CloudMeasure platform would be an efficient tool that helps in the creation of models to determine the price-performance comparisons.

Further work is needed to review and increase performance data sets in order to validate the CMDF process within the CloudMeasure platform. In addition, it is necessary to develop and test new tools for creating performance analysis models, which contribute to analyze the performance of CCS that could contribute to validating proposed methodologies.

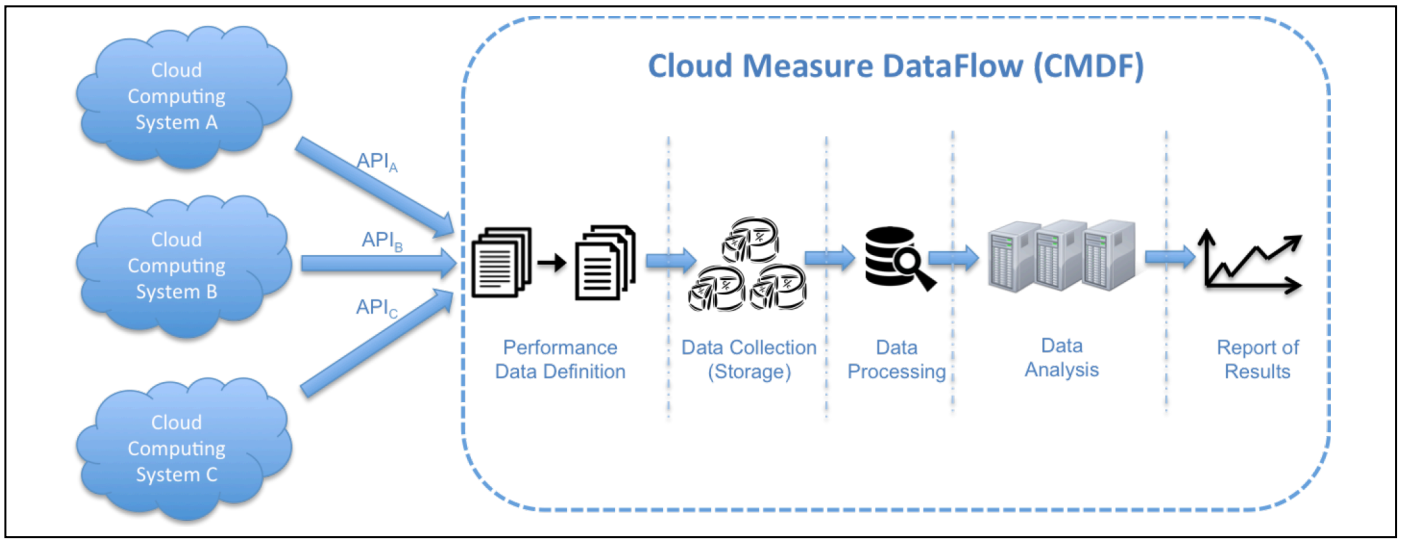

Fig. 6 Stages and elements that constitute the CMDF process.

REFERENCES

[1] ISO/IEC, "ISO/IEC JTC 1 SC38:Cloud Computing Overview and Vocabulary," ed. Geneva, Switzerland: International Organization for Standardization, 2012.

[2] ISO/IEC, "ISO/IEC JTC 1 SC38:Study Group Report on Cloud Computing," ed. Geneva, Switzerland: International Organization for Standardization, 2011.

[3] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, *Distributed Systems Concepts and Design*, 5th ed. Edinburgh: Addison Wesley, 2011.

[4] I. I. G. 99-12, "International Vocabulary of Metrology – Basic and General Concepts and Associated Terms, VIM," International Organization for Standardization ISO/IEC, Geneva, Switzerland2007.

[5] ISBSG, "Title," unpublished|.

[6] L. Olsina, G. Lafuente, and O. Pastor, "Towards a reusable repository for web metrics," *J. Web Eng.*, vol. 1, pp. 61-73, 2002.

[7] D. Kondo, B. Javadi, A. Iosup, and D. Epema, "The Failure Trace Archive: Enabling Comparative Analysis of Failures in Diverse Distributed Systems," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, 2010, pp. 398-407.

[8] L. Bautista, A. Abran, and A. April, "Design of a Performance Measurement Framework for Cloud Computing," *Journal of Software Engineering and Applications,* vol. 5, pp. 69-75, February 2012 2012.

[9] B. Li, L. Gillam, and J. O'Loughlin, "Towards Application-Specific Service Level Agreements: Experiments in Clouds and Grids," in *Cloud Computing: Principles, Systems and Applications, Computer Communications and Networks*. vol. 0, ed London: Springer-Verlag, 2010, pp. 361-372.

[10] G. R. Gangadharan and D. M. Parrilli, "Service Level Agreements in Cloud Computing: Perspectives of Private Consumers and Small-to-Medium Enterprises," in *Cloud Computing for Enterprise Architectures, Computer Communications and Network*. vol. 0, Z. Mahmood and R. Hill, Eds., ed London: Springer-Verlag, 2011, pp. 207-225.

[11] ISO/IEC, "ISO/IEC 25012: Software Engineering: Software Product Quality Requirements and Evaluation (SQuaRE) - Data Quality Model," ed. Geneva, Switzerland: International Organization for Standardization, 2008, p. 18.

[12] L. E. Bautista Villalpando, A. April, and A. April, "DIPAR: A Framework for Implementing Big Data Science in Organizations," in *Continued Rise of the Cloud: Advances and Trends in Cloud Computing*. vol. I, Z. Mahmood, Ed., 1 ed London, England: Springer-Verlag London, 2014, pp. XIX, 410.

[13] ISO/IEC, "ISO/IEC 15939:2007 Systems and software engineering — Measurement process," ed. Geneva, Switzerland: International Organization for Standardization, 2008.

[14] A.F.S. (2012, June 13). *Apache Flume*. Available: http://flume.apache.org/

[15] Facebook. (2012, June 13). *Scribe*. Available: https://github.com/facebook/scribe/wiki

[16] B. Dhruba. (2010). *Hadoop Distributed File System Architecture*. Available: http://hadoop.apache.org/docs/r0.20.2/hdfs_design.html

[17] A.S.F. (2013, June 6th). *Apache HBase, the Hadoop database, a distributed, scalable, big data store.* Available: http://hbase.apache.org/

[18] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM,* vol. 51, pp. 107-113, January 2008 2008.

[19] A. Thusoo, J. Sen Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, *et al.*, "Hive-a petabyte scale data warehouse using Hadoop," in *26th International Conference on Data Engineering*, Long Beach, California, USA, 2010, pp. 996-1005.

[20] A.S.F. (2012). *What is Apache Mahout?* Available: https://cwiki.apache.org/confluence/display/MAHOUT/Overview