COSMOS

Conceptual architecture for Data Science applied to Big Data

Whitepaper

Version 1.1, 27-MAR-2019

Carlos Godinho (cma.godinho@gmail.com)

Table of Contents

Introduction

COSMOS is a conceptual architecture created for solving complex problems with large data volumes and using data science with advanced computing techniques.

The concept of COSMOS tackles two issues:

- Increase in data collection and the need to process this data;
- Capability to use distributed computing power to process in an acceptable timeframe.

Although initially created to address the TELCO domain, COSMOS architecture is by default generic and may be used in other businesses. COSMOS principles are governed by a set of well-defined rules which work as requirements. Developers of COSMOS artefacts are guided by those rules.

Conceptual Architecture vs Framework Myth

Opposite to a framework, a conceptual architecture defines in detail how to build and use software artefacts, using frameworks already available. So, COSMOS does not position itself as a framework, it uses general propose frameworks available in the market.

Nowadays, SW market offers frameworks with high quality both in functionality and performance. Creating additional frameworks is a major effort with high demand of resources and large costs. Therefore, the concept of COSMOS is to continuously evaluate and choose the best frameworks. Evaluation criteria is based on stability, performance and ease of use.

By using best of class frameworks, COSMOS developers focus on costumer use cases, providing a light and flexible architecture with fast development cycles.

Environment

COSMOS is based on the architecture framework presented in Figure 1.

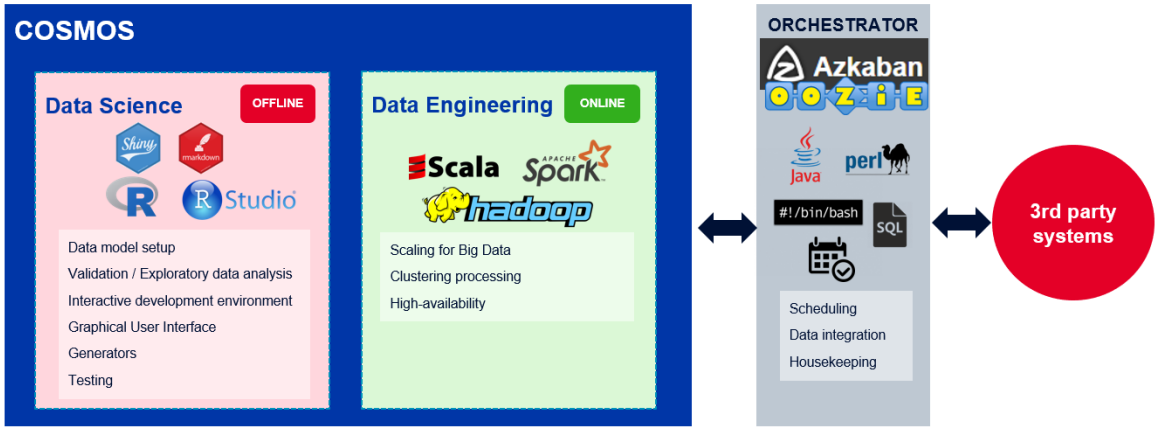

Figure 1 - COSMOS environment

At COSMOS core, two blocks are identified:

- **Data Science** – Used by data scientists, this block offers the necessary tools for performing exploratory data analysis, data visualization and model creation. It is based on R programming language and its stack of software. It is marked as **OFFLINE** in the sense that is not offered to customers directly;

- **Data Engineering** – Used by data engineers, this block offers the necessary tools for large data processing with cluster computing. It uses the Spark framework for data processing, Hadoop file system for distributed storage and Scala as programming language. It is marked **ONLINE** in the sense that the result of its usage are artefacts delivered to customers.

A third block, outside COSMOS, is also depicted. It is called the **Orchestrator**. Although outside COSMOS scope, the Orchestrator are responsible for all actions related to workflow management, schedule, housekeeping and data integration. Two possible alternatives for orchestrators are Azkaban and Oozie. Integration activities may be executed over different technologies as represented in the picture.

Workflow & Deliveries

Besides internal documentation and processes, COSMOS team defines a complete development cycle in collaboration with customers to deliver:

- **Requirements documentation** - Shared document with a full description of functional and non-functional requirements;

- **Design documentation** – Technical description of created artefacts, allowed configurations and execution details;

- **Code artefacts** – In the form of Scala jar delivers, ready to be integrated in a SPARK cluster (other technologies may be supported if needed).

- **Test cases** – Complete description of tests cases, including performance tests.

NOTE: Artefacts orchestration and data integration from/to 3rd party systems are out of scope of COSMOS team. Such activities may be performed by integration teams at customer request.

Example: Detecting anomalies, outage and business impact in a telco mobile network

Introduction

Mobile telco networks are built with thousands of network elements providing a large set of KPIs. Monitoring KPIs in a network and understand its variations is a real big data problem, requiring specific techniques from data science and data engineering. COSMOS is being used in this scope to provide the following major requirements:

- Calculate neighbour telco sites based on basic information about sites (latitude and longitude);
- Learn and evaluate network cell KPIs from its multiple instances and flag possible anomalies;
- Aggregate detected anomalies for cell to site and neighbour sites, calculating outage and business impact.

To solve such requirements, COSMOS team has created four distinct artefacts, as presented in Figure 2.

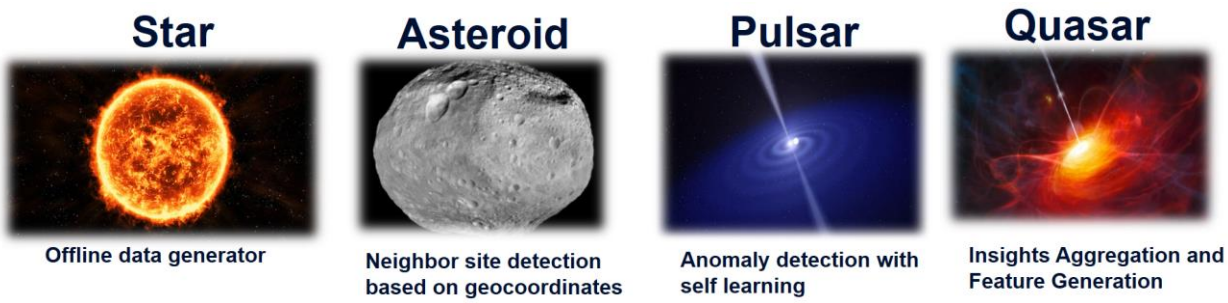

Figure 2 - COSMOS artefacts for anomaly, outage and business impact

Star – Data generator

Star is a data generator able to simulate a complete set of data necessary for COSMOS anomaly, outage and business requirement implementation.

It is not easy to get real data from customers. Data is subject to legal issues, quality validations and technical limitations in transfer and storage. With Star the necessary data may be generated offline, allowing the creation of multiple scenarios. Like this, data creation for tests and demos is easily generated.

Star is an example of COSMOS flexibility and creative capacity of its architecture.

Asteroid – Finding neighbour sites

Outage and business impact applied at cell location is quite limited. At cell failure execution, a subscriber may be served by another cell at the same site or at a neighbour site. So, analysis must be extended at site / neighbour site.

Getting information about cell location and cell-site relationship is trivial. However, finding the neighbour relationship between sites is more complicated. To solve this issue, COSMOS Asteroid application is available. By using basic site metadata (latitude and longitude), Asteroid calculates the neighbour sites.

Asteroid uses advanced calculations based on spherical trigonometry and min/max distance configurations to calculate the neighbours of each site.

A representation of Asteroid execution is available in Figure 3.

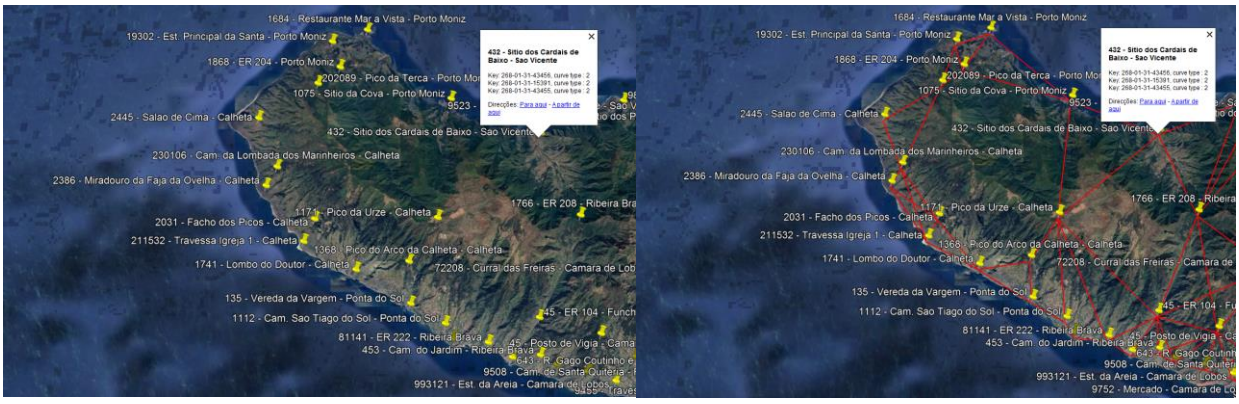

Figure 3 - Asteroid site neighbour before and after calculation at Madeira Island, Portugal (red lines represent neighbourhood)

Pulsar – Anomaly detection

Pulsar is a general anomaly detection application able to address multi-instance of a KPI. Cells in a network generate several KPIs. The complexity, in this case, is not to evaluate a single cell, but to evaluate the complete set of cells instances, which may grow to hundreds of thousands.

To make anomaly detection possible at this level, Pulsar supports:

- Automatic learning - Learns with previous values and uses the values to build memories about past experiences. Anomalies are extrapolated from variations between received and predicted values;
- Zero configuration – No configuration is necessary per cell. With auto learning, Pulsar creates an expected behaviour;
- Advance configuration – Pulsar sensibility is configured to accommodate for different cases;
- Parallel KPI processing – Different KPIs may be monitored in parallel.

Pulsar executes per KPI and per time-period (called a BIN). Recommended BINs start at 5 m. Per each KPI/BIN Pulsar produces a tidy dataframe with an indication per cell if an Anomaly has been identified. Figure 4 presents a graphical representation of the daily variation of a cell KPI. Values above and below the expected deviation, are marked as anomalies.

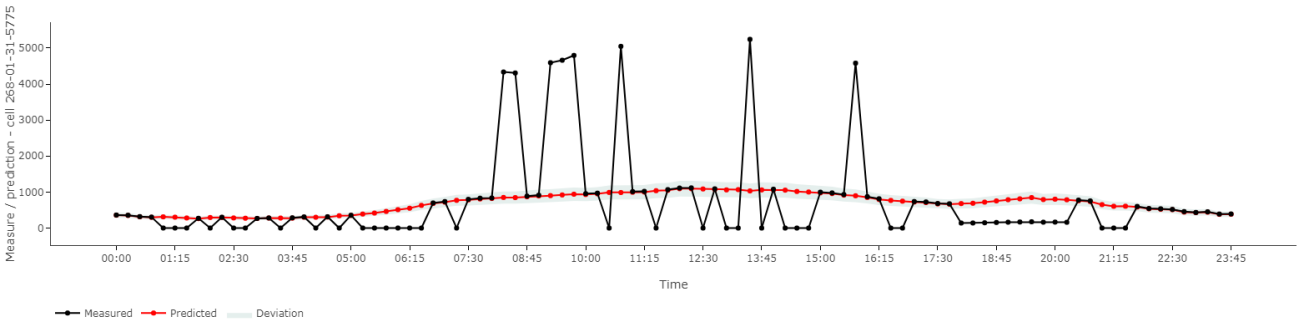

Figure 4 – Daily representation of measured and predicted data and the acceptable deviation for a cell KPI

Quasar – Extending to outage and business impact

Quasar is a post processing engine, able to use data generated by Asteroid and Pulsar to produce a daily anomaly analysis. It defines a set of internal KPIs, evaluating anomaly occurrence, outage and business impact along the day.

Besides cells, Quasar uses configuration and Asteroid output to perform a geographical aggregation for its internal KPIs. Each KPI is documented with its formulation, as presented in Figure 5.

$$\sum_{i=1}^{N} if\ [(A_i = A_h)\ \mathrm{V}\ (A_i = A_l)\ ]\ 1\ else\ \emptyset$$

Figure 5 – Quasar KPI for # anomaly count

Data is aggregated per site (as presented in Figure 6) and by neighbour site.

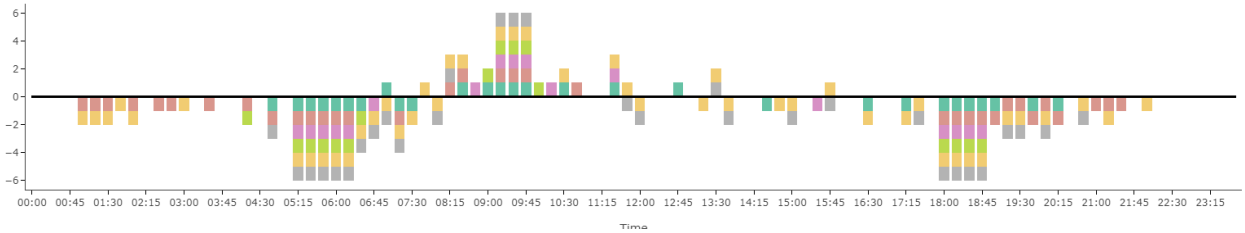

Figure 6 – # of anomalies detected daily. Anomalies above and below expected values are represented.

Outage and anomaly are taken as internal special KPIs and applied both to site and neighbour site level. Variations of outage and anomalies are supported and described formally in documentation. An example of an anomaly formulation is presented in Figure 7, together with a graphical representation.

$$\sum_{i=1}^{N} if\ [\ (\ A_i = A_h)\ \mathrm{V}\ (\ A_i = A_l)\ ]\ Mea_i - \ Prd_i\ else\ \emptyset$$

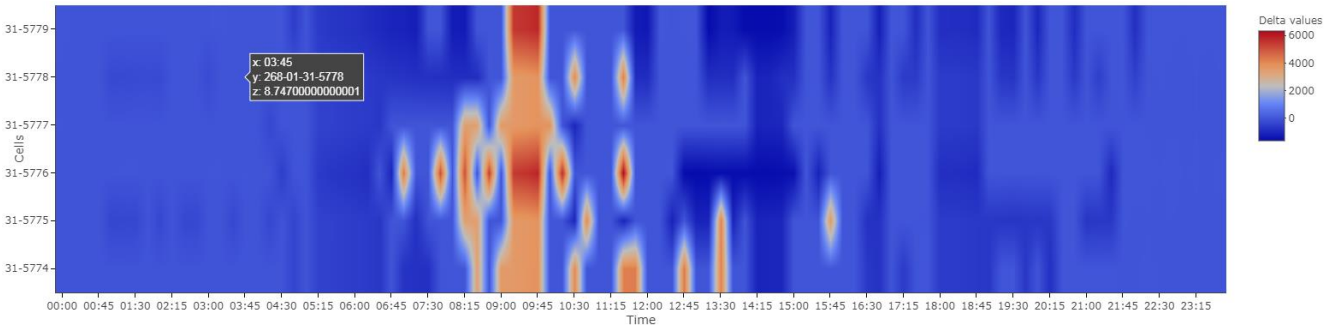

Figure 7 – Site heat map with variations between measured and predicted values. Site cells are depicted.

To allow easy data visualization and reuse, a set of tidy dataframes are generated and extreme cases are sorted and presented with rankings.

Conclusion

COSMOS conceptual architecture provides a full development cycle to tackle complex requirements with the use of data science techniques and scaling up for big data.

The concept is built on top of state of the art technology, able to provide advanced working environments, fast development cycles, high availability, efficient use of HW and performance. Additionally, the concept is agnostic to a specific business domain.

With COSMOS and a close collaboration with customers, it is possible to deliver tailored solutions in small timeframes with quality. Costumers gain access to advance a toolset, allowing the creation of sophisticated scenarios for prediction, trends and alerts.