**ACATA DE ENTREGA DE SERVICIO**

**PERIODO 20/02/2020 - 31/03/2020**

|  |  |  |  |
| --- | --- | --- | --- |
| Actividad 1: Liderar técnicamente implementaciones hechas en lenguaje Scala -Spark. | | | |
| Tarea | **Detalle** | **Inicio** | **Final** |
| Refactorización métodos enrich, createScoresModelsInstances,  getScoreAttributes de la clase orchestrator.py motor de score | Pasar método de ejecución de secuencial a paralelo al procesamiento de los objetos de Score, con el objetivo de utilizar adecuadamente el motor de procesamiento en paralelo de Spark. | 23/02/2020  Volumen Data: 1.2 TB  Tiempo Ejecución  (Hrs) 2:07:00 | 01/03/2020  FULL Data: 1.2 TB  Tiempo Ejecución  (Hrs) 1:33:00 |
| Serialización de la clase main.py, utilizando la librería Kryo | Implementación de la librería Kryo, librería recomendada por Spark, para optimizar la serialización de los objetos, mayor compresión, menos latencia en la red, más rápido el procesamiento en paralelo. | 28/02/2020  Volumen Data: 3.2 TB  Tiempo Ejecución  (Hrs) 2:43:00 | 04/03/2020  Volumen Data: 3.2 TB  Tiempo Ejecución  (Hrs)1:36:55 |

|  |  |  |  |
| --- | --- | --- | --- |
| Actividad 2: Tunning a nivel de Spark-Yarn, de los componentes asignados. | | | |
| Tarea | **Detalle** | **Inicio** | **Final** |
| Desarrollo componente PartitionCalculation  Link repositorio: <https://bitbucketglobal.experian.local>  /projects/SCEN/repos/scr\_eng\_app/  browse?at=refs%2Fheads%2FSCOENG-44-despliegue-artifactory | Es un Jar que calcula las particiones o tareas que ejecutara un Job en Spark, según la cantidad de registros de la data y cantidad de core asignados, de acuerdo con el ambiente a ejecutar. El valor calculado de las particiones será registrado en el archivo Partition.properties, luego los Motores leerán del archivo y particionarán adecuadamente la ejecución del Job de Spark. | Pendiente de agendar pruebas en QA |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Actividad 3: Documentar buenas prácticas de codificación aplicadas a los componentes asignados. | | | |
| Buena practica | **Detalle** | **Inicio** | **Final** |
| Uso correcto del nivel del paralelismo  Link Confluence:  <https://confluenceglobal.experian.local>  /confluence/display/CCBR  /Enignes+Partition+Calculation | Evitar en lo posible el procesamiento secuencial de los objetos contenidos en los Jar y/o librerías externas, en el siguiente ejemplo se ajustó una interacción, por la ejecución de un método con una variable global | def createScoresModelsInstances(self):  for scoreInfo in self.scoresBundle:  field  if scoreConfig.get("scoreModule"):  try: # Import module  scoreModule = importlib.import\_module(  scoreConfig.get("scoreModule"))  # Validate score instance field  if scoreConfig.get(  "scoreInstance"):  scoreClass = getattr(scoreModule, scoreConfig.get("scoreInstance")) | self.attributesPathDict.update(  self.getScoreAttributes(  scoreConfig.get  ("scoreAttributes"))) |