

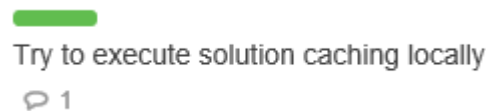
ID	PR - 13.06
Fase	Implementación
Cumplimiento	60%

Objetivos de la semana

- Solucionar los problemas de persistencia de los datos en Spark.

Tareas realizadas

Ejecutar el guardado de soluciones localmente



Tras converger en la iteración 33, el programa falla al imprimir la solución. Una de las últimas operaciones en los *workers* es guardar las soluciones en caché. Esto se hace utilizando el objeto *_PHBase*. Para asegurar que esta caché está disponible para el hilo principal, esta última operación se realiza cargando los workers en local desde spark.


```
if execute_locally:
    localWorkerList = self._rddWorkerList.collect()
    updatedWorkers = []
    for worker in localWorkerList:
        updatedWorkers.append(_do_parallel_work(worker, task, queue_name))
    self._rddWorkerList.unpersist()
    self._rddWorkerList = self._sparkContext.parallelize(updatedWorkers)
```

Con esto no conseguimos solucionar el problema. Ahora que los workers se ejecutan en local podemos ver que los valores que se guardan en caché están vacíos. Esto puede ser porque Spark no los ha guardado en las iteraciones anteriores o se han perdido a la hora de ejecutar el *collect*.

Commits

- [Local init of workers and export of ReferenceModel to spark](#)

Faltan llamadas a PHSpark



[interscenario.py] couple of checks for
phpyro didn't include phspark


1

Algunas de las comprobaciones de tipo del solver manager no estaban actualizadas para incluir PHSpark. Sin embargo, estas funciones no son llamadas en el ejemplo actual y no influyen en el resultado.

Commits

- [Local init of workers and export of ReferenceModel to spark](#)

Probar ejecución local




Tested local execution of the spark
workers

≡

Con el objetivo de asegurar que el problema reside en la implementación de Spark y no en la implementación de los workers, utilizamos la lista de workers que previamente formaba el RDD como una lista en Python. Ejecutando todas las mismas operaciones ahora sobre la lista de Python en lugar de sobre el RDD, la salida es correcta y exactamente igual a la ejecución con Pyro.

De esta forma confirmamos que nuestra implementación y uso de los RDD es incorrecta.

Error No module found ReferenceModel



No module found ReferenceModel

≡ 1

Este error ocurre ocasionalmente al serializar o deserializar los workers, dependiendo del estado en que estos se encuentren. Este ReferenceModel es el archivo del modelo que corresponda al ejemplo que se esté ejecutando y se carga desde el archivo cuando crea el árbol de escenarios.

Forzamos su carga en Spark como dependencia con:

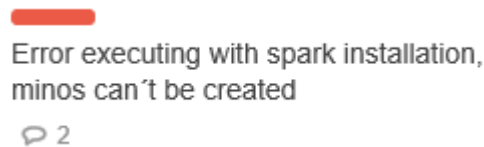
```
self._sparkContext.addPyFile(os.path.join(os.getcwd(), 'models', 'ReferenceModel.py'))
```

Esto parece que causa un bucle infinito porque el RDD devuelve una lista de resultados vacía cuando se solicita.

Commits

- [Local init of workers and export of ReferenceModel to spark](#)

Error creando el solver cuando se ejecuta en la instalación de Spark



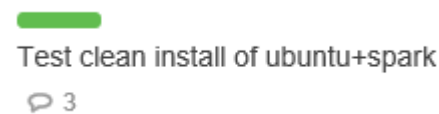
Probando la ejecución del programa sobre la instalación de Spark (en lugar de con `setMaster('local')`) encontramos un error en el que los workers no son capaces de instanciar el solver *minos*.

Ejecutamos la inicialización de los workers localmente para posteriormente paralelizar esta lista de workers inicializados en Spark. De esta forma el RDD es creado directamente con los objetos generados y sus dependencias resueltas.

Commits

- [Local init of workers and export of ReferenceModel to spark](#)

Test con una instalación limpia



Sobre una instalación limpia de ubuntu y spark volvemos a probar la ejecución. Todo funciona de la misma forma, por lo que podemos descartar el entorno como un problema.