

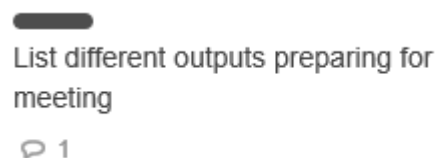
ID	PR - 29.06
Fase	Implementación
Cumplimiento	60%

Objetivos de la semana

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

Tareas realizadas

Listar diferentes salidas en preparación para la reunión



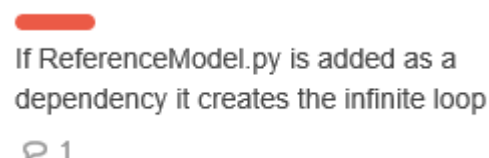
Tras múltiples intentos infructuosos de ejecutar correctamente el programa bajo Spark, se planifica una reunión para el día 19/06. Como preparación para esta reunión, se genera un documento listando múltiples variaciones del algoritmo y la salida que producen.

El documento indicado es *"/Documentación/Reuniones/[M03] - (res) Errores.pdf"*

Commits

- [New test outputs](#)

Añadiendo la dependencia "ReferenceModel" se genera un bucle infinito



Al añadir este archivo como dependencia, los *workers* en Spark son capaces de importarlo como módulo. El bucle infinito es causado porque, al solucionar este error, se llega a un punto posterior de la ejecución donde se lanza una excepción. Esta excepción no llega a ser manejada porque es atrapada por Spark, el método que se estaba ejecutando no finaliza correctamente y, por lo tanto, no llega a devolver la instancia de *PHSparkWorker* actualizada. Esto supone que el RDD se vacía y el programa se queda en bucle solicitando resultados mientras sólo se devuelven listas vacías.

Error creando "minos"

Error executing with spark installation,
minos can't be created

5

Cuando ejecutamos el algoritmo sobre la instalación de spark (en lugar de utilizar *setMaster("local")*), no es posible inicializar "minos" como ejecutable del solver y se muestra un warning pero la ejecución continúa normalmente. Esto sucede porque el entorno de ejecución dentro de los workers es distinto al local, por lo que la ruta donde se encuentra el ejecutable "minos" no es accesible desde Spark.

Para solucionar esto añadimos el ejecutable como una dependencia en Spark con:

```
self._sparkContext.addFile("/home/crist/Downloads/minos/minos")
```

Posteriormente conseguimos la ruta del ejecutable accesible en Spark y la pasamos como keyword al método de inicialización.

```
data = pyutilib.misc.Bunch(**task['data'])
if data.action == "initialize" and data.solver_type == "minos":
    minosPath = SparkFiles.get("minos")
    kwds["solver_path"] = os.path.dirname(os.path.abspath(minosPath))
```

Commits

- [Fixed minos dependency on workers](#)
- [Really fixed minos dependency](#)

AttributeError: 'PHSolverServer' object has no attribute 'solver_map'

AttributeError: '_PHSolverServer'
object has no attribute '_solver_map'

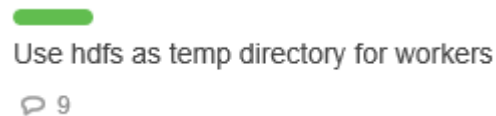
5

Algunas de las llamadas a pyomo se estaban haciendo al módulo instalado en el entorno virtual de python y no al código fuente del proyecto. Estas dos versiones son diferentes y por ello causaban este error.

Solucionado instalando el proyecto actual manualmente desde el código fuente.

```
/home/crist/python-venv/pyomo3/bin/python [setup.py](http://setup.py/) develop
```

Usar hdfs como directorio para los trabajos de Spark



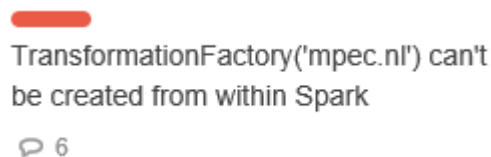
Para calcular la solución a un problema, en cada iteración, el programa lee el árbol de escenarios y lo guarda en un archivo con extensión ".nl". Este archivo funciona como entrada para el solver elegido. Para asegurar el correcto funcionamiento de estas funciones en un entorno distribuido se usará un sistema de archivos distribuido, en este caso, Hadoop.

Para usar Hadoop desde Python se hace uso de la librería [pydoop](#).

Commits

- [Using hdfs to store temp files](#)
- [Fixed hdfs write to file](#)

"TransformationFactory" no se crea correctamente



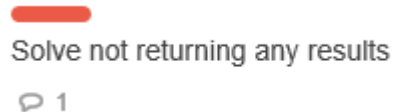
En el *presolve* se solicita una instancia a esta *TransformationFactory* para generar la entrada correcta para el solver. A esta factory se le pasa un string para identificar la instancia que se generará. Ejecutando esto desde Spark no es posible determinar esta instancia y no se genera correctamente. Esto, de nuevo, se debe a un problema de localización de dependencias desde el entorno de Spark.

Como solución, se generan las instancias necesarias antes de paralelizar los objetos en Spark. Estas instancias persisten correctamente tras la serialización y pueden ser usadas normalmente.

Commits

- [Forced instances created by factories to workers](#)

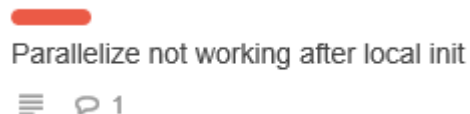
Método "solve" no devuelve resultados



Solve not returning any results
P 1

Error causado por una excepción previa generada por alguna de las pruebas realizadas.

Método "parallelize" no funciona correctamente con los objetos inicializados



Parallelize not working after local init
P 1

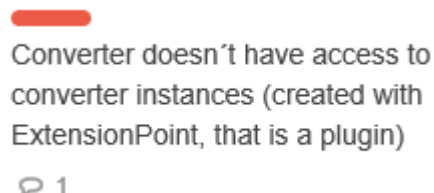
Inicializando los workers localmente para posteriormente generar el RDD con estas instancias ya iniciadas causa el siguiente error:

En estas tres líneas, falla el segundo *assert*:

```
self._rddWorkerList = self._sparkContext.parallelize(self._localWorkerList)
assert len(self._localWorkerList) > 0
assert self._rddWorkerList.count() > 0
```

Esta situación desemboca en el caso del bucle infinito definido anteriormente. Por lo tanto, se concluye que está causado por una excepción no controlada dentro de los *PHSparkWorker*.

Converter no encuentra una instancia correcta



Converter doesn't have access to
converter instances (created with
ExtensionPoint, that is a plugin)
P 1


De nuevo en el *presolve* se necesita convertir el modelo actual a un archivo aceptado por el solver escogido. Para esto dependemos de otro patrón *factory* que no funciona correctamente desde Spark.

Como solución se instancia fuera de Spark y se serializa.

Commits

- [Forced instances created by factories to workers](#)

Error en la ejecución del solver



Solver error executing, written file
might be empty


🗨 4

Error en llamada a *ReaderFactory*

Commits

- [Forced instances created by factories to workers](#)

Comprobar entradas y salidas del solver



Check input and output of the solvers

🗨 1

Usando la opción `--keep-solver-files` podemos reutilizar el archivo `.pyomo.nl` como entrada de minos ejecutado localmente y comprobar la salida con el archivo `.sol` guardado desde Pyomo.

Las salidas coinciden por lo que confirmamos que el solver funciona correctamente aun siendo invocado desde Spark.