|  |  |
| --- | --- |
| Universidad Nacional del Centro  de la Provincia de Buenos Aires  Facultad de Ciencias Exactas |  |

**Introducción a Grid Computing**

Integrantes:

Pérez Fuentes, Joaquín Alejandro [joaquinpf@gmail.com](mailto:rodrigocavo@gmail.com)

Steimbach, Marcos Maximiliano msteimba@alumnos.exa.unicen.edu.ar

## Introduccion

Se plantea como objetivo la inclusion de la plataforma Condor en el proyecto JGRIM.

La meta consiste en realizar una solucion acorde al diseño actual de JGRIM, proveyendo una manera simple y univoca de invocar ejecutables en un cluster Condor, obtener su respuesta y brindar mecanismos de configuracion al usuario.

## Desarrollo de la solucion

### Diseño de la solución

Partiendo de la premisa de obtener una solucion simple y que respete los estandares de diseño actuales de JGRIM, se comenzo haciendo un analisis de la plataforma para identificar partes reutilizables del diseño del modulo *parallelization*.

Esto resulto en que el esquema actual (propuesto por IbisServer, IbisClient, IbisMethods, IbisExecutionRequest, IbisResult e IbisInterceptor) podia adaptarse sin mayor problemas a Condor.

En este esquema, el usuario debe generar primero la instanciacion particular de IbisMethods (tanto interface como implementacion), junto con su correspondiente XML para la ejecucion de Spring. El usuario entonces corre uno de los metodos definidos en IbisMethods. Estos son capturados por el interceptor, que genera el cliente pertinente y envia la solicitud de ejecucion al servidor, para luego aguardar por la respuesta de este y devolversela al usuario.

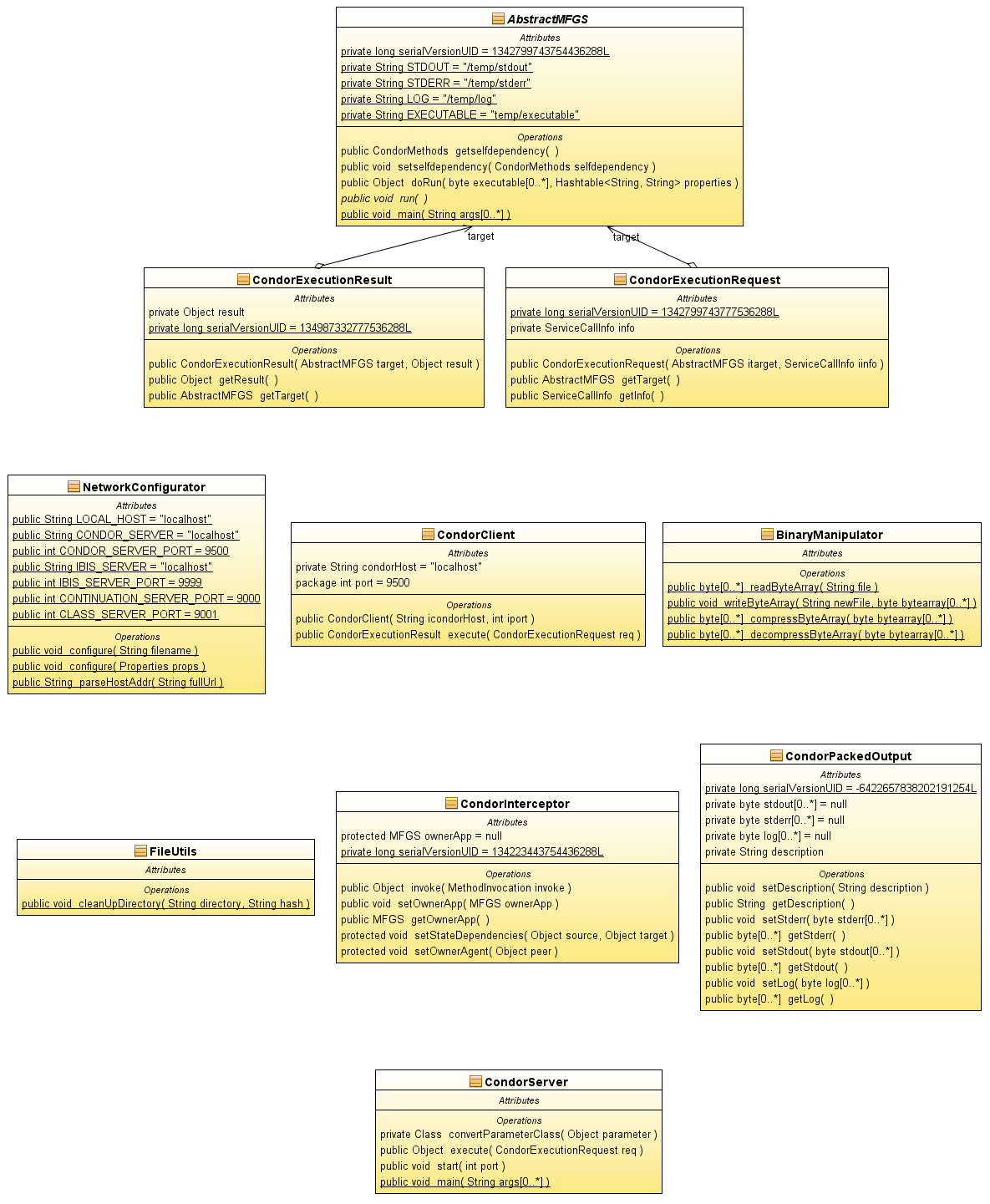
A simple vista las diferencias con el esquema actual son:

* Necesidad de Condor de serializar y enviar el ejecutable en el pedido de invocacion.
* Menor cantidad de puntos de configuracion dado que Condor no esta atado al lenguaje y simplemente corre “ejecutables”en un universo dado. No existe necesidad de generar la interface “IbisMethods”para cada aplicacion en particular ya que siempre se terminara corriendo el mismo metodo. De la misma manera, ya no es necesario generar un XML particular para Spring, este sera provisto por el framework. De este modo se le deja al usuario unicamente la carga del ejecutable y sus opciones de invocacion.

Una linea de ejecución de la aplicación deberia ser:

*El usuario carga el ejecutable y opciones de ejecucion de condor en memoria e invoca el metodo de ejecución de Condor. Este ultimo sera capturado por Spring y enviado como solicitud al servidor, que realizara la ejecución real segun la configuración de Condor que posea y las opciones enviadas por el usuario. Luego de ejecutarse, se deben obtener los archivos resultantes, serializarlos y empaquetarlos, para entonces devolver normalmente la respuesta al usuario.*

El diseño que soporta todo esto puede verse reflejado en el siguiente diagrama de clases.



El metodo “run()”de la clase AbstractMFGS es unico hook que provee la implementacion de Condor sobre JGRIM. En este metodo se deberan cargar el ejecutable (utilizando la clase BinaryManipulator) y las opciones de ejecucion de Condor, definidas como pares “clave-valor”validos para los “submit description” de Condor.

Valores importantes a tener en cuenta son:

* **universe-standard/vanilla/java**: Define el universo de ejecucion de Condor. Si se quiere correr un binario Java (class) se deberia definir “java”como el universo. De manera similar, si se desea correr un ejecutable standard de consola de Windows/Dos se puede utilizar el universo “vanilla”.

Para mantener la generacion de trabajos sencilla, no se tendran en cuenta los valores (seran utilizados valores por defecto):

* **output**: Define la ubicacion del archivo que contendra la salida estandar de la ejecución de Condor.
* **error**: Define la ubicacion del archivo que contendra la salida estandar de errores de la ejecución de Condor.
* **log**: Define la ubicacion del archivo de log de la ejecución de Condor.
* **log\_xml**: Define si se utilizan logs con formato XML.