# ARQUITECTURA y TECNOLOGIaS DEL SOFTWARE

Pràctica 1 – Big Data

**Enginyeria Informàtica**

**Universitat Autònoma de Barcelona (UAB)**

Cristian Vega Sánchez - 1426805

Ricard Bernal Petit - 1395031

Tabla de contenidos

[Analisis 3](#_Toc3839764)

[Propuesta 4](#_Toc3839765)

[Disseño 5](#_Toc3839766)

[Disseño UML 5](#_Toc3839767)

[Trabajo realizado 6](#_Toc3839768)

[Conclusiones 6](#_Toc3839769)

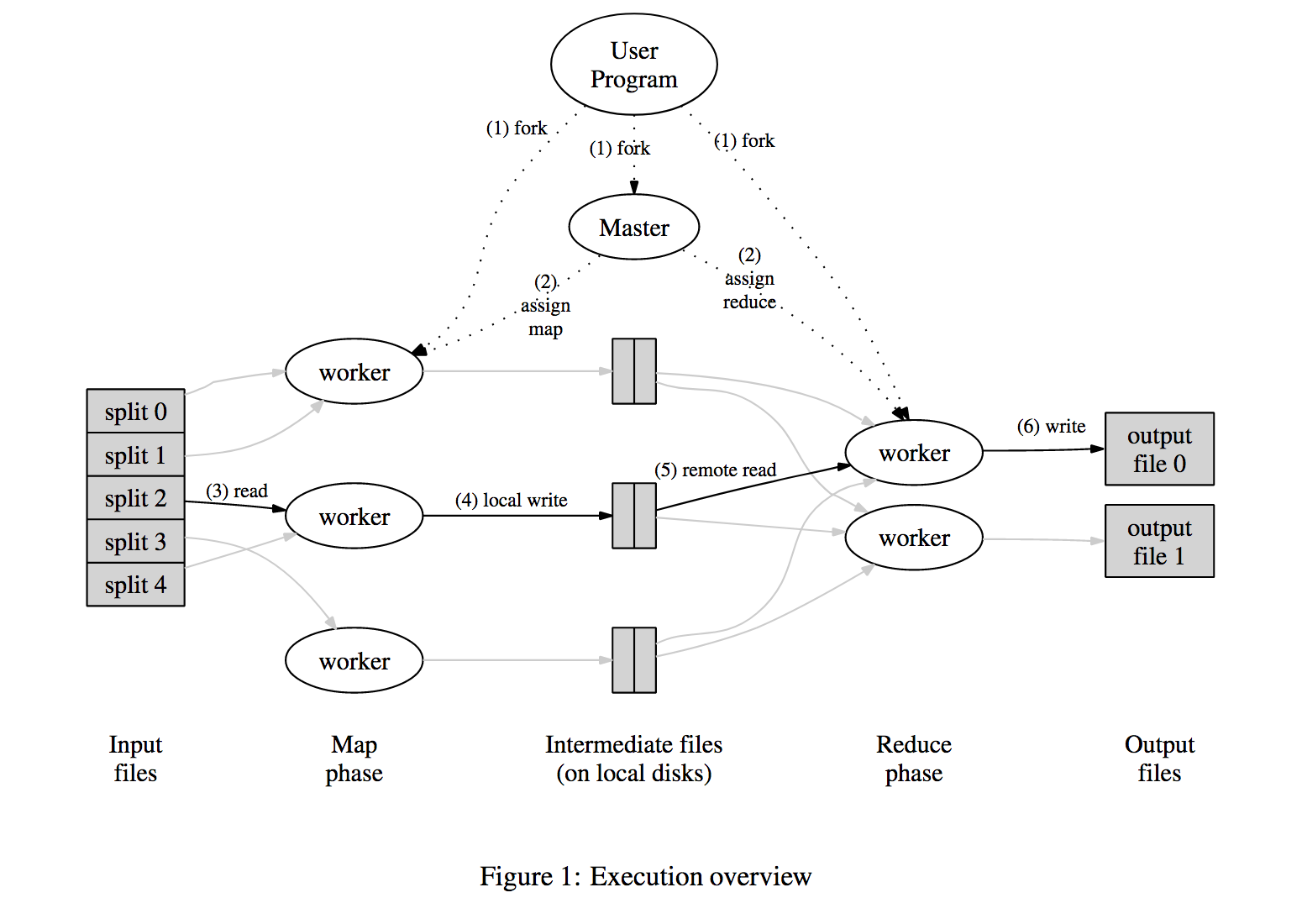
[Compilacion - Ejecucion 7](#_Toc3839770)

[Bibliografia 8](#_Toc3839771)

## Analisis

En esta practica se nos plantea el problema del tratamiento de grandes cantidades de datos, con la finalidad de obtener información útil, para llevar a cabo esta operación de tratamiento se nos pedía implementar el algoritmo *Map-Reduce*, extensamente utilizado debido a la implementación OpenSource llamada Hadoop. Por lo tanto se ha analizado dicho algoritmo y se han obtenido una serie de conclusiones que se mencionan a continuación.

El algoritmo nos permite trabajar con *datasets* muy grandes debido a su versatilidad, la cual refleja la aplicación del principio “divide y vencerás”, mediante los dos métodos de programación Map y Reduce, por lo tanto los problemas en lo que este algoritmo pueda aportar una solucions son aquellos que de puedan dividir mediante las dos operaciones mencionadas, es decir que se puedan agrupar por una característica común en los datos. El algoritmo *Map-Reduce* tiene un flujo de ejecución como el que se muestra a continuación, el cual se ha obtenido de una fuente de internet (ver bibliografía):



Como podemos observar en la figura 1, es un flujo de trabajo en paralelo que nos permite utilizar la máxima capacidad de computo para abordar ciertos problemas en los que implica el tratamiento de grandes cantidades de datos.

## Propuesta

Una vez analizado el algoritmo y visto los posibles problemas que nos puede plantear la implementación del mismo, hemos llegado a la conclusión que nuestra propuesta debe incluir los siguientes puntos:

* Entrada de datos (*dataset*) mediante fichero con extensión txt
* *Dataset* dividido en bloques, para poder ejecutarlo en paralelo
* Mapeado de pareja clave-valor mediante una lista de listas
* Ejecución del proceso en dos escenarios, uno con dos *threads* y el otro con cuatro

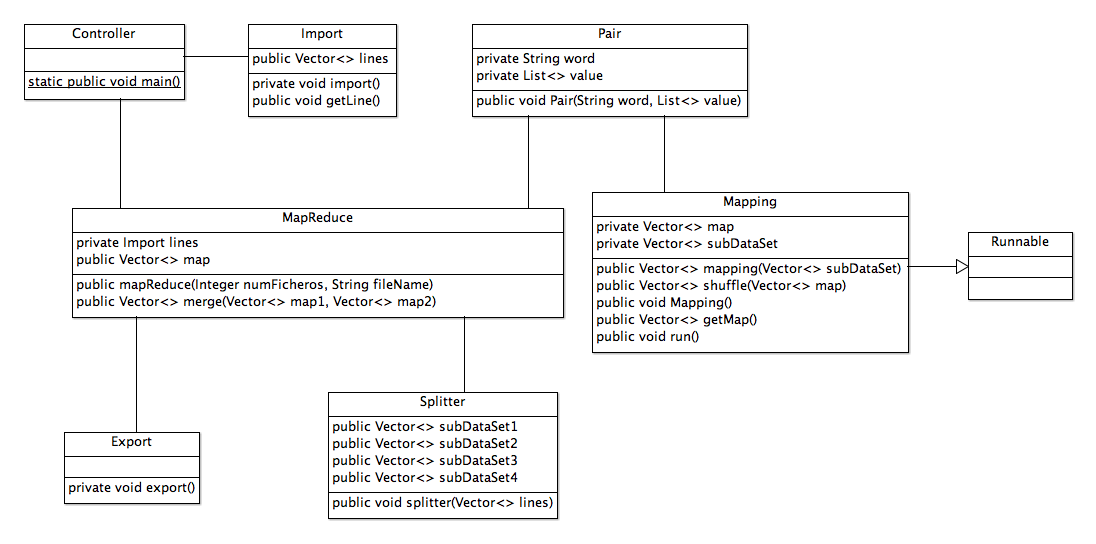
## Disseño

Para realizar el diseño se han analizado varias herramientas que nos permiten trabajar con modelado de datos UML. Dentro de todas las herramientas encontradas se han tenido en cuenta las siguientes: ArgoUML, BoUML y yEd.

ArgoUML tiene las ventajas de permitir guardar los diagramas en varios formatos uno de los cuales es zip y no solo en un formato especifico como suelen hacer todas estas herramientas además es intuitivo y fácil de usar. BoUML es menos intuitivo pero aparentemente mas completa, a pesar de que para empezar a trabajar hay que realizar una configuración mínima en la cual se pide lenguaje que se usará para programar, ya que luego permite generar el código diseñado. Finalmente yEd que nos permite generar gran cantidad de diagramas, así como tiene tiene una opción de diseño automático que sirve de plantilla para generar los diagramas pero las opciones basicas como añadir atributos o metodos son demasiado rebuscadas. De manera que una vez realizado este analisis se ha escogido ArgoUML por su facilidad de uso y entorno intuitivo.

En este punto solo hacia falta reflejar el analisis anteriormente realizado de manera que el modelado queda de la siguiente forma:

### Disseño UML



## Trabajo realizado

Análisis previo del algoritmo que se nos pedía implementar, en este análisis se ha buscado información tanto en los *slides* proporcionados en el campus como en internet, con la finalidad de comprender la idea base del algoritmo MapReduce, para evaluarlo en perspectiva y no centrarnos solo en como implementarlo para contar palabras, es decir, se a fijado el objetivo de comprender como funciona el algoritmo de forma genérica.

Análisis del algoritmo en si mismo, con el objetivo de comprender las partes que lo forman (Split, map, shuffle, reduce)

Diseño del algoritmo mediante el modelado UML del cual se ha generado el diagrama de clases a seguir para la implementación.

Implementación del algoritmo.

Realización de las pruebas y documentación del proyecto.

## Conclusiones



Figura 2 – Tiempo de ejecución de *MapReduce* según *threads*

Tal y como vemos en la figura 2 para 60k palabras el tiempo de ejecución solo se ve disminuido cuando usamos 2 *threads* de ejecución.

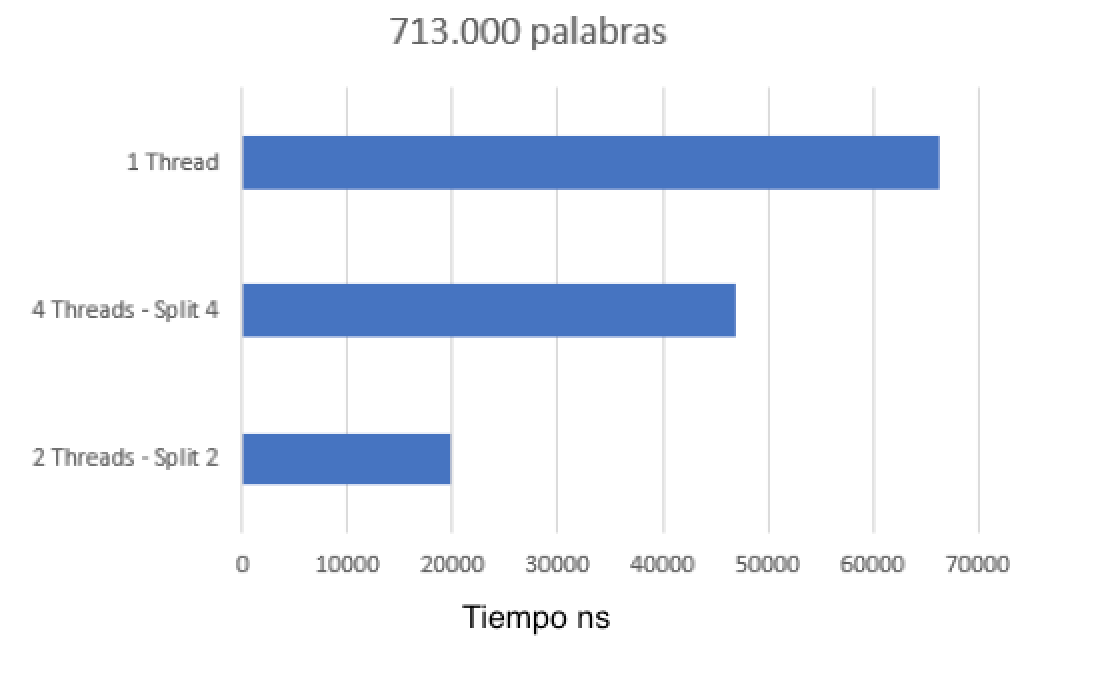


Figura 3 - Tiempo de ejecución de *MapReduce* según *threads*

En este caso ya podemos observar como cuanto mayor es el dataset, mas se reduce el tiempo de ejecución con 2 threads, si extrapolamos los resultados podemos entender que el efecto de los 4 threads se notará cuando el dataset sea mucho mayor, ya que al usar mas threads el numero de shuffles aumenta, en este caso el cuello de botella se encuentra en la función shuffle ya que hasta que no terminan de ejecutar el método shuffle en todos los threads no se ejecuta la función reduce. Otro aspecto importante es el numero de palabras por línea ya que reducimos el dataset en el total de líneas entre el numero de thread, puede darse el caso que una línea contenga mas palabras que otra y esto podría hacer que un thread tardara mas que otro. El algoritmo elegido para realizar el shuffle quizás no es el mas optimo ya que compara el elemento actual con todos los elementos siguientes y así para todos los elementos, esto supone una complejidad de !n-1.

Si comparamos varias ejecuciones, como por ejemplo la figura 2 y la figura 3 podemos observar que cuando el volumen de datos es pequeño, utilizar 4 o mas threads no aumenta la eficiencia. De manera que cuando el volumen de datos.

## Compilacion - Ejecucion

Con tal de facilitar al máximo la compilación y ejecución de nuestro programa hemos creados dos Scripts Shell dentro de la carpeta bin:

* InstallJRE.sh
* TexCounter.sh

Con tal de ejecutar nuestra aplicación MapReduce en Linux necesitamos la máquina virtual de Java: JRE (Java Runtime Environment) la cual se puede instalar ejecutando la siguiente instrucción en nuestro Terminal: (dentro de la carpeta bin)

- **sh InstallJRE.sh**

Este script se encargará de descarga la última versión de JRE.

Una vez instalada la maquina virtual de Java ya podemos ejecutar el Script Shell **TextCounter**, que es el encargado de gestionar los parámetros (ficheros entrada) y la ejecución del archivo **JAR**, mediante el siguiente comando en el terminal:

* sh TextCounter.sh <FileName1>.txt
* sh TextCounter.sh <FileName1>.txt ··n·· <FileNameN>.txt

Ejemplo:

* **sh TextCounter.sh Texto1.txt**
* **sh TextCounter.sh Texto1.txt Texto2.txt**

Este Script Shell se encarga de gestionar el número de parámetros que le pasamos al **TextCounter** y lanza también la ejecución del fichero **JAR** MapReduce.jar:

* java -jar MapReduce.jar File1.txt → (1 Fichero)
* java -jar MapReduce.jar File1.txt File2.txt → (2 Ficheros)

**OutOfMemory (fuera de rango de memoria):**

Si el fichero de entrada es muy grande y Java nos lanza un mensaje de erro de OutOfMemory dentro de los Threads podemos asignar la cantidad de memoria máxima a la hora de ejecutar el programa añadiendo -Xmx(cantidad de memoria en mb):

Ejemplo:

* java -jar -Xmx8024m MapReduce.jar Text1.txt 🡪 Asignamos una memora máxima de

8024mb

## Bibliografia

[1] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. Recuperado de: [http://delivery.acm.org/10.1145/1330000/1327492/p107-dean.pdf](http://delivery.acm.org/10.1145/1330000/1327492/p107-dean.pdf?ip=158.109.94.211&id=1327492&acc=ACTIVE%20SERVICE&key=DD1EC5BCF38B3699%2EDA44A35B463519EE%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1551861839_36283d2fb0e168875662cc5377aa0cd6)

[2] [Rambabu Posa](https://www.journaldev.com/author/rambabu). MapReduce Algorithm Example. Recuperado de: <https://www.journaldev.com/8848/mapreduce-algorithm-example>

[3] Lluís Gesa Bote. Arquitectura i Tecnologies de Software [MO23770]. Recuperado de: <https://e-aules.uab.cat/2018-19/course/view.php?id=17482&section=2>

[4] Steven Krenzel. MapReduce. Recuperado de: <http://stevekrenzel.com/finding-friends-with-mapreduce>