

UNIVERSIDADE DE SANTIAGO DE
COMPOSTELA



ESCOLA TÉCNICA SUPERIOR DE ENXEÑARÍA

FlinkBWA

**Uso de tecnologías *Big Data* para el
alineamiento de secuencias genéticas**

Autora:

Silvia Rodríguez Alcaraz

Tutor:

Juan Carlos Pichel Campos

Cotutor:

José Manuel Abuín Mosquera

Grado en Ingeniería Informática

Junio 2019

Trabajo de Fin de Grado presentado en la Escuela Técnica Superior de
Ingeniería de la Universidad de Santiago de Compostela para la obtención del
Grado en Ingeniería Informática



D. Juan Carlos Pichel Campos, Profesor del Departamento de Electrónica y Computación de la Universidad de Santiago de Compostela, y **D. José Manuel Abuín Mosquera**, Profesor del Departamento de Electrónica y Computación de la Universidad de Santiago de Compostela,

INFORMAN:

Que la presente memoria, titulada *FlinkBWA: uso de tecnologías Big Data para el alineamiento de secuencias genéticas*, presentada por **D.^a Silvia Rodríguez Alcaraz** para superar los créditos correspondientes al Trabajo de Fin de Grado de la titulación del Grado en Ingeniería Informática, se realizó bajo nuestra dirección en el Departamento de Electrónica y Computación de la Universidad de Santiago de Compostela.

Y para que así conste a los efectos oportunos, expiden el presente informe en Santiago de Compostela, a

El tutor,

El cotutor,

La alumna,

Juan C. Pichel Campos José M. Abuín Mosquera Silvia Rodríguez Alcaraz

Agradecimientos

Resumen

Dada la gran cantidad de datos genómicos generados actualmente por las tecnologías de secuenciación de próxima generación (NGS) la tarea de alineación de secuencias de ADN es especialmente costosa. La paralelización del código de estos alineadores permite mejorar el tiempo de una ejecución secuencial al poder realizar varias tareas a la vez. Además, también permite abarcar problemas cada vez mayores, aprovechando los recursos de entornos con varios nodos de trabajo, como los *clusters*.

El presente proyecto tiene por objetivo paralelizar el *Burrows-Wheeler Aligner* (BWA), uno de los alineadores de secuencias genéticas más ampliamente utilizado, con *Apache Flink*. El fin último es el de mejorar el tiempo de cómputo del programa aprovechando los distintos nodos de los que dispone el *cluster Big Data 1* del *CiTIUS*.

Índice general

1. Introducción	1
1.1. Contextualización	1
1.2. Motivación	1
1.3. Objetivos	2
1.4. Organización de la memoria	2
2. Gestión del proyecto	5
2.1. Gestión del Alcance	5
2.1.1. Descripción del alcance del proyecto	5
2.1.2. Criterios de aceptación	5
2.1.3. Entregables del proyecto	6
2.1.4. Restricciones del proyecto	6
2.1.5. Metodología de desarrollo	6
2.2. Gestión de la configuración	9
2.2.1. Gestión del código	10
2.2.2. Gestión de la documentación	10
2.3. Gestión del tiempo	11
2.3.1. Estructura de Descomposición del Trabajo (EDT)	11
2.3.2. Planificación final: <i>sprints</i>	14
2.4. Gestión de las comunicaciones	15
2.4.1. Identificación de interesados	15
2.4.2. Plan de gestión de los interesados	18
2.5. Gestión de riesgos	19
2.5.1. Identificación de riesgos	19
2.5.2. Análisis de riesgos	20
2.5.3. Planificación de respuesta a los riesgos	23
2.5.4. Materialización de riesgos	26
2.6. Gestión de costes	27
2.6.1. Consideraciones previas	27
2.6.2. Costos directos	27
2.6.3. Costos indirectos	29
2.6.4. Financiamiento	29
2.6.5. Coste total	30

3. Especificación de requisitos	31
3.1. Definición del sistema	31
3.2. Casos de uso	32
3.2.1. Identificación de casos de uso	32
3.2.2. Especificación de casos de uso	33
3.3. Catálogo de requisitos	34
3.3.1. Requisitos funcionales	35
3.3.2. Requisitos no funcionales	37
4. Análisis	41
4.1. Aplicaciones de alineamiento genético	41
4.1.1. <i>Burrows-Wheeler Aligner</i>	42
4.2. Estado del arte: tecnologías <i>Big Data</i>	44
4.2.1. <i>Apache Flink</i>	46
5. Diseño	49
5.1. Arquitectura	49
5.2. Diagrama de clases	50
5.2.1. Explicación del diseño y patrones utilizados	51
5.2.2. Descripción de las clases	52
5.3. Diagramas de secuencia	53
6. Implementación	57
6.1. Requisitos de <i>Apache Flink</i>	57
6.2. Medios utilizados	57
6.2.1. Herramientas de desarrollo	57
6.2.2. Entorno de desarrollo y pruebas	58
6.3. Funcionamiento del programa	58
6.4. CLI FlinkBWA	60
7. Pruebas	63
7.1. Verificación	63
7.1.1. Pruebas de requisitos funcionales	63
7.1.2. Pruebas de requisitos no funcionales	65
7.2. Validación	66
7.3. Pruebas de rendimiento	69
7.3.1. Especificación de las pruebas	69
7.3.2. Resultados	71
7.3.3. Análisis de los resultados	72
8. Conclusiones	75
9. Trabajo futuro	77

A. Manual técnico	79
A.1. Entorno	79
A.2. Requisitos <i>software</i>	80
A.3. Datos de entrada	80
B. Manuales de usuario	81
B.1. Instalación	81
B.2. Compilación	81
B.3. Ejecución	81
Bibliografía	85

Índice de figuras

2.1. Ciclo de vida de un <i>sprint</i>	9
2.2. Estructura de Descomposición del Trabajo (EDT)	13
2.3. <i>Sprint</i> 1	14
2.4. <i>Sprint</i> 2	14
2.5. <i>Sprint</i> 3	15
2.6. <i>Sprint</i> 4	15
3.1. Modelo funcional de la aplicación	32
3.2. Diagrama de casos de uso	33
4.1. Ejemplo del formato FASTQ	43
4.2. Cronograma sobre la evolución de <i>Big Data</i>	44
4.3. Ejemplo del modelo <i>Map Reduce</i>	45
5.1. Arquitectura de <i>Flink</i>	50
5.2. Diagrama de clases	51
5.3. Diagrama de secuencia: <i>Single Reads</i>	54
5.4. Diagrama de secuencia: <i>Paired Reads</i>	55
6.1. Operación <i>join</i> de <i>Flink</i> en un entorno HDFS	60
7.1. Eficiencia temporal <i>FlinkBWA</i>	73
7.2. <i>Speedup</i>	74

Índice de cuadros

2.1. Datos del interesado Juan C. Pichel Campos	17
2.2. Datos del interesado José M. Abuín Mosquera	17
2.3. Datos de la interesada Silvia Rodríguez Alcaraz	18
2.4. Matriz de comunicación: Desarrolladora → Tutor y/o cotutor . .	18
2.5. Matriz de comunicación: Tutor y/o cotutor → Desarrolladora . .	19
2.6. Plantilla para análisis de riesgos	20
2.7. Análisis del RSK-001	20
2.8. Análisis del RSK-002	21
2.9. Análisis del RSK-003	21
2.10. Análisis del RSK-004	21
2.11. Análisis del RSK-005	22
2.12. Análisis del RSK-006	22
2.13. Análisis del RSK-007	22
2.14. Análisis del RSK-008	23
2.15. Análisis del RSK-009	23
2.16. Análisis del RSK-010	23
2.17. Matriz de exposición ante riesgos	24
2.18. Plantilla para control de riesgos	24
2.19. Plan de control para RSK-003	25
2.20. Plan de control para RSK-004	25
2.21. Plan de control para RSK-006	25
2.22. Plan de control para RSK-007	26
2.23. Plan de control para RSK-008	26
2.24. Plan de control para RSK-010	26
2.25. Costos de los Recursos Humanos	29
2.26. Costes del proyecto	30
3.1. Especificación CU-001	33
3.2. Especificación CU-002	33
3.3. Especificación CU-003	34
3.4. Especificación CU-004	34
3.5. Niveles de importancia	34
3.6. Plantilla de especificación de requisitos	35
3.7. Especificación RF-001	35

3.8. Especificación RF-002	36
3.9. Especificación RF-003	36
3.10. Especificación RF-004	36
3.11. Especificación RF-005	37
3.12. Especificación RNF-001	38
3.13. Especificación RNF-002	38
3.14. Especificación RNF-003	38
3.15. Especificación RNF-004	39
3.16. Especificación RNF-005	39
6.1. Opciones de <i>FlinkBWA</i>	61
7.1. Plantilla para Pruebas Unitarias	63
7.2. PU-001: verificación RF-002	64
7.3. PU-002: verificación RF-003	64
7.4. PU-003: verificación RF-004	64
7.5. PU-004: verificación RF-005	65
7.6. PU-005: verificación RNF-001	65
7.7. PU-006: verificación RNF-003	66
7.8. PU-007: verificación RNF-004	66
7.9. Plantilla PV	67
7.10. Especificación PV-001	68
7.11. Especificación PV-002	68
7.12. Especificación PV-003	68
7.13. Especificación PV-004	68
7.14. Especificación PV-005	68
7.15. Especificación PV-006	69
7.16. Ficheros utilizados en las pruebas de rendimiento	70
7.17. Resultados de las pruebas de rendimiento	72

Capítulo 1

Introducción

1.1. Contextualización

Los continuos avances dentro del sector tecnológico han permitido que, en la actualidad, cualquier persona con acceso a los medios informáticos adecuados sea capaz de generar una importante cantidad de datos diariamente. Concretamente, el estudio anual del IDC (*International Data Corporation*) del año 2012 [1] revelaba que en 2020 cada persona generaría al año aproximadamente más de 5200 *gigabytes* de información única.

Consecuencia de este notorio progreso, hoy por hoy, los medios digitales son prioritarios a la hora de salvaguardar información, pero este hecho trae consigo un reto importante: ¿cómo gestionar, almacenar y analizar de manera eficiente tal cantidad de información? En este punto es donde entran en juego las herramientas *Big Data* que, simplificando, se encargan de facilitar el tratamiento de este tipo de datos.

Por su parte, las tecnologías de secuenciación de próxima generación (NGS) también han dado lugar a una gran cantidad de datos genómicos que precisan ser analizados e interpretados. Una de las fases más costosas del análisis es la alineación de secuencias de ADN, que supone el mapeo de miles de millones de pequeñas secuencias sobre un genoma de referencia. Es por esto que los alineadores de vanguardia emplean estrategias de paralelización con el fin de disipar este problema, pero las soluciones suelen mostrar una implementación compleja además de una baja escalabilidad.

1.2. Motivación

El *Burrows-Wheeler Aligner* (BWA) [2] es un *software* empleado para mapear secuencias de baja divergencia contra un genoma de referencia grande, como el

humano. Se trata de uno de los alineadores de secuencias genéticas más ampliamente utilizados, pero sufre el mismo problema que el resto de alineadores al trabajar con grandes volúmenes de datos.

El CiTiUS (Centro Singular de Investigación en Tecnoloxías da Información) cuenta con un proyecto que aplica la plataforma *Big Data Apache Spark* para mejorar el rendimiento del BWA, *SparkBWA* [4]. *Apache Flink* es un proyecto de la *Apache Software Foundation* que aparece posteriormente a *Spark* y que, lógicamente, todavía no cuenta ni con su madurez ni con su comunidad, pero sí muestra grandes posibilidades. Mientras que *Spark* sólo permite el procesado de datos por lotes, *Flink* soporta el procesado por lotes y el procesado de flujos de datos, realmente útil en un contexto de trabajo en tiempo real o *streaming*.

El presente trabajo consiste en usar la tecnología *Apache Flink* [3] para la paralelización del BWA, con el fin de observar su efecto sobre el rendimiento del alineador en términos de eficiencia.

1.3. Objetivos

El objetivo principal de dicho trabajo es usar la plataforma *Big Data Apache Flink* para paralelizar el alineador *Burrows-Wheeler Aligner* (BWA). Concretamente, el trabajo englobaría el siguiente conjunto de objetivos:

- Estudio previo del estado del arte que justifique el uso de tecnologías *Big Data* para paralelizar el BWA y tratar de mejorar sus resultados temporales.
- Formación en tecnologías *Big Data* y en aplicaciones de alineamiento genético.
- Diseño modular del nuevo alineador BWA paralelo que permita la escalabilidad de la aplicación.
- Implementación de la aplicación *FlinkBWA*.
- Análisis del rendimiento de la aplicación en un *cluster*.

1.4. Organización de la memoria

La estructura que sigue la memoria es la siguiente:

- **Capítulo 1. Introducción:** presente capítulo donde se contextualiza el proyecto, se presenta la motivación del mismo y se muestra la estructura de la memoria.

- **Capítulo 2. Gestión del proyecto:** capítulo dedicado a la gestión del proyecto. Incluye la gestión de: alcance, configuración, tiempo, riesgos, costes, comunicaciones y justifica la metodología de desarrollo escogida.
- **Capítulo 3. Especificación de requisitos:** especifica los requisitos que debe cumplir el proyecto.
- **Capítulo 4. Análisis:** incluye una introducción a las aplicaciones de alineamiento genético, una descripción del BWA, un breve estudio del estado del arte de las tecnologías *Big Data* y una justificación de *Flink* como plataforma escogida para el desarrollo.
- **Capítulo 5. Diseño:** explicación detallada del diseño realizado para la aplicación.
- **Capítulo 6. Implementación:** capítulo dedicado a aspectos vinculados a la implementación de la aplicación como el flujo de trabajo de la misma, las herramientas empleadas o los requisitos identificados para su codificación.
- **Capítulo 7. Pruebas:** comprende la descripción de las pruebas realizadas sobre la aplicación y los resultados de las mismas.
- **Capítulo 8. Conclusiones:** menciona y desarrolla las conclusiones obtenidas tras la realización del proyecto.
- **Capítulo 9. Trabajo futuro:** reflexión sobre posibles ampliaciones que podrían llevarse a cabo en el proyecto.
- **Apéndice A. Manual Técnico:** guía que indica todos los requisitos de entorno y *software* necesarios para poder trabajar con *FlinkBWA*.
- **Apéndice B. Manual de Usuario:** guía que indica al usuario cómo obtener el *software*, compilarlo y ejecutarlo.
- **Bibliografía**

Capítulo 2

Gestión del proyecto

Este capítulo incluye todos los contenidos vinculados a la gestión del proyecto. Gestionar un proyecto implica estudiar y elaborar una planificación, organización y control de los recursos disponibles para definir y alcanzar una serie de objetivos concretos. Tomando como base los principios de la 5ª ed. del PMBOK [5] se abordan: la gestión del alcance, la gestión de la configuración, la gestión del tiempo, la gestión de las comunicaciones, la gestión de riesgos y la gestión de costes del proyecto.

2.1. Gestión del Alcance

La Gestión del Alcance comprende el conjunto de procesos necesarios para asegurar que el proyecto incluye todos los contenidos necesarios para su finalización exitosa. Básicamente, el objetivo principal es especificar qué debe incluir el proyecto y qué no de forma detallada.

2.1.1. Descripción del alcance del proyecto

Este proyecto pretende paralelizar el alineador de secuencias genéticas BWA mediante la incorporación de la plataforma *Big Data Apache Flink*.

Para lograr el fin descrito anteriormente es necesario: comprender el funcionamiento del BWA y de *Flink*, además de sus requerimientos; diseñar la nueva aplicación paralela; implementar la aplicación planteada; y, finalmente, estudiar la eficiencia temporal de la misma en un *cluster*.

2.1.2. Criterios de aceptación

Los criterios de aceptación de este proyecto serían la obtención de: una aplicación paralela del BWA que haga uso de *Apache Flink*, un estudio de la eficiencia

temporal de la misma para evaluar el impacto de *Flink* sobre la aplicación original y una memoria que detalle todo el desarrollo del proyecto de manera adecuada.

2.1.3. Entregables del proyecto

Una vez finalizado el proyecto, los contenidos a entregar son los siguientes:

- **Soporte digital con el *software FlinkBWA*:** soporte digital que contenga la aplicación desarrollada junto con la documentación adicional necesaria y la memoria.
- **Memoria del proyecto:** se trata del presente documento, donde se describe detalladamente el proceso llevado a cabo para la finalización exitosa del proyecto y toda la información necesaria sobre el mismo.

2.1.4. Restricciones del proyecto

La única restricción a tener en cuenta para este proyecto es temporal e implica lograr la finalización del mismo en aproximadamente 402 horas de trabajo autónomo. Además, todos los contenidos indicados en 2.1.3 deben ser entregados en la Administración de la ETSE antes de las 11:00h del 1 de julio de 2019, que es el límite oficial establecido por la escuela [6] para realizar la lectura del trabajo en la convocatoria de julio de 2019, concretamente entre los días 18 y 19 de julio.

2.1.5. Metodología de desarrollo

En el marco de un proyecto de *software* es fundamental decidir la metodología que se va a seguir, ya que determina cómo estructurar, planificar y controlar el proceso de desarrollo. La elección de una u otra metodología varía según el caso y depende de las características del proyecto que se vaya a realizar.

Concretamente, este proyecto presenta tres factores importantes a la hora de escoger la metodología a seguir para su desarrollo:

- La aplicación a desarrollar presenta un carácter innovador, ya que la primera versión de *Apache Flink* data de 2015, hace tan sólo 4 años. Por este motivo, la plataforma todavía no tiene una gran comunidad de usuarios ni se encuentra en su mayor nivel de madurez.
- La aplicación está orientada a un ámbito muy específico: el alineamiento de secuencias genéticas.
- El desconocimiento de la tecnología a utilizar y la falta de formación sobre aplicaciones de alineamiento de secuencias genéticas de la desarrolladora.

Consecuencia de los puntos descritos anteriormente, es probable que la aplicación experimente cambios bastante continuos durante el desarrollo. Por este motivo, para el presente trabajo, se ha decidido seguir una metodología ágil. Resumiendo los 12 principios del *Manifiesto Ágil* [7], este tipo de metodologías se centran en:

- Frecuente realización de entregas útiles.
- Continua comunicación y colaboración entre responsables y desarrolladores.
- Auto-gestión del equipo de trabajo, reflexionando a intervalos regulares sobre su propia efectividad y perfeccionando su comportamiento como consecuencia.
- Atención continua a la excelencia técnica y al buen diseño.
- Capacidad para mantener un ritmo constante de forma indefinida.

La metodología ágil escogida en este caso es *Scrum*, ya que presenta un enfoque iterativo e incremental de desarrollo. De esta forma, aporta control y flexibilidad para realizar cambios en situaciones de incertidumbre o falta de conocimiento así como para predecir y/o evitar posibles riesgos.

Scrum

A continuación, se describirán los aspectos más relevantes de la metodología escogida, haciendo uso de los conocimientos expuestos en la Guía de *Scrum* [8].

Para comenzar, *Scrum* se basa en el empirismo y, por tanto, sostiene que las decisiones han de basarse en el conocimiento que proviene de la experiencia previa. Además, como ya se mencionó, aplica un enfoque iterativo e incremental, con el que se pretende optimizar la predictibilidad, así como el control de los riesgos. Podría decirse que sus tres pilares fundamentales son: transparencia, inspección y adaptación. Resumiendo: debe haber una comunicación fluida dentro del equipo de trabajo, revisión continua de los avances y aplicación de cambios en caso de encontrar un aspecto que se desvíe de los límites aceptables.

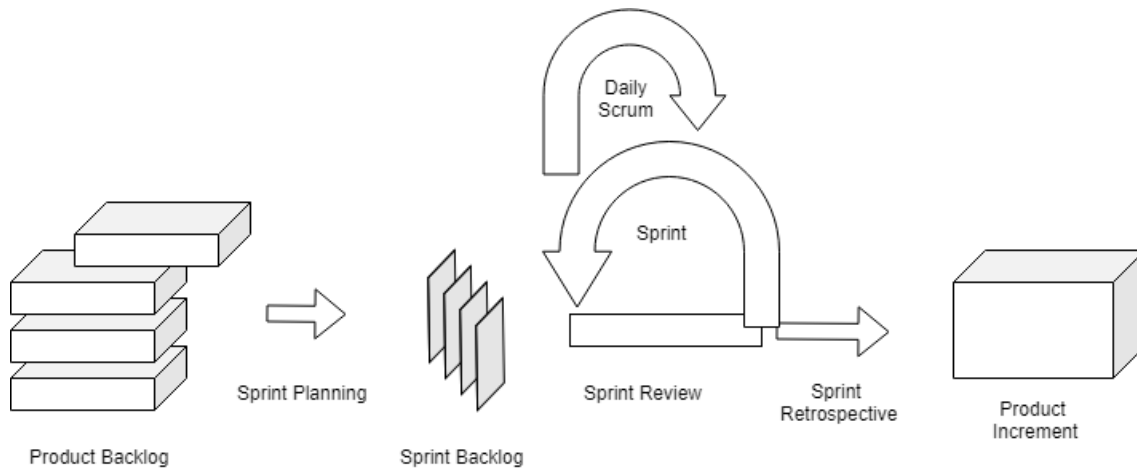
Los equipos de trabajo que presenta esta metodología son auto-organizados y multifunción. La entrega de productos es iterativa e incremental, aumentando las oportunidades de recibir retroalimentación. Los principales roles del *Scrum team* son:

- **Dueño del producto (*Product Owner*):** responsable de maximizar el valor del producto y del trabajo del Equipo de Desarrollo.

- **Equipo de desarrollo (*Development Team*):** se encarga de realizar el trabajo correspondiente a cada incremento.
- ***Scrum Master*:** líder del proyecto y encargado de asegurar que se está siguiendo correctamente la metodología.

En cuanto al ciclo de trabajo propuesto por *Scrum*, existen una serie de eventos predefinidos que buscan crear cierta regularidad y evitar reuniones no previstas. El principal evento se denomina *sprint* y, básicamente, actúa como contenedor del resto de eventos. Los *sprints* suelen tener una duración de entre 2-4 semanas, su ciclo de vida puede observarse en la Figura 2.1. Dentro de cada *sprint* se identifican los siguientes eventos:

- **Reunión de planificación del *sprint*:** se planifica el trabajo a realizar durante el *sprint* concretando: qué se va a entregar y cómo se va a conseguir. Con esta información y la del *product backlog* (lista completa de características que debe tener el producto a desarrollar) se genera el *sprint backlog* que, básicamente, es la lista de objetivos que se deben cumplir al finalizar el *sprint*.
- ***Scrum* diario:** breve reunión entre los miembros del equipo de desarrollo para determinar un plan para las siguientes 24 horas y sincronizar sus tareas. Para ello, se debe inspeccionar el trabajo realizado desde el último *Scrum* diario y considerando qué trabajo podría avanzarse hasta el siguiente. Se trata de una reunión clave de inspección y adaptación.
- **Trabajo de desarrollo:** es la parte del *sprint* dedicada a trabajar para avanzar en los objetivos propuestos.
- **Revisión del *sprint*:** revisión donde se inspecciona el incremento realizado y se adapta el *Product Backlog* en caso de ser necesario. Es una reunión de carácter informal entre *stakeholders* y Equipo *Scrum* para fomentar la colaboración y comunicación. En conjunto, se trata de determinar qué es lo siguiente que se debe hacer.
- **Retrospectiva del *sprint*:** el Equipo *Scrum* reflexiona sobre su propia actividad y busca mejoras que incorporar en el siguiente *sprint*.

Figura 2.1: Ciclo de vida de un *sprint*

Aplicación de *Scrum*

Dado que el marco de trabajo de este proyecto no es exactamente el idóneo para el que fue ideado *Scrum*, ya que se centra en optimizar la productividad y comunicación de un equipo de trabajo, se han realizado una serie de adaptaciones para adecuar la metodología al mismo.

En primer lugar, se entenderá por reuniones diarias el tiempo empleado por la desarrolladora para organizar su actividad del día en función de los avances realizados previamente, dado que en este caso el equipo de desarrollo se reduce a una única persona. Por este motivo, la alumna también tomará el rol de *Scrum Master*, gestionando que la metodología se cumpla apropiadamente. El tutor y cotutor, por su parte, serán *Product Owners*, ya que se encargarán de revisar el trabajo realizado por la alumna intentando que el valor del producto final sea el máximo posible.

En cuanto a los *sprints*, se fijó una duración de 3-4 semanas para los mismos, revisando al final de este tiempo: los avances realizados y sus posibles mejoras, el trabajo por realizar y qué hacer en el siguiente *sprint*.

2.2. Gestión de la configuración

El proceso de Gestión de la Configuración permite tener control de los elementos claves del proyecto en todo momento, los cuales experimentan gran cantidad de cambios durante su desarrollo. Estos cambios son a veces imprevistos, y pueden llegar a generar un impacto potencialmente negativo sobre el proyecto. Por esta

razón, es crucial impedir que se produzcan pérdidas de información que puedan afectar al desarrollo normal del proyecto.

Se denominan elementos de configuración todos aquellos expuestos a cambios durante el desarrollo y que, por tanto, es necesario controlarlos. En el caso de este proyecto, dichos elementos son:

- Código fuente de la aplicación *FlinkBWA*
- Conjunto de información sobre el proyecto expuesta en la presente memoria.
- Ficheros varios en relación con el proyecto (actas de reunión, gráficos, diseños, etc.).

Teniendo en cuenta los elementos de configuración identificados, a continuación se plantea una gestión de la configuración para el código fuente y otra para la documentación que, englobaría el contenido de la memoria y los ficheros auxiliares al desarrollo del proyecto que pueden encontrarse volcados en la memoria o no (p.e.: las imágenes correspondientes al diseño de la aplicación se incluirán en la memoria pero no documentos como actas de reunión o similares).

2.2.1. Gestión del código

Para una gestión eficiente del código fuente de la aplicación se debe garantizar que la desarrolladora pueda acceder al código actualizado en cualquier momento y, en caso de requerirse, ser capaz de regresar a versiones anteriores del mismo.

Teniendo en cuenta los requisitos mencionados, se decidió alojar el código en un repositorio de la plataforma *Github*, empleando *git* como *software* de control de versiones. Además, se empleará *Google Drive* como *backup* adicional del código, puesto que también permite mantener un historial de versiones anteriores.

2.2.2. Gestión de la documentación

La memoria del proyecto será realizada en *LaTeX* sobre la plataforma *online Overleaf*, ya que permite la edición en línea de este tipo de documentos. Por tanto, además de servir como editor, la plataforma también proporciona un alojamiento seguro para el documento. De todas formas, al igual que con el código, cada vez que se finalice un incremento de contenido significativo, se realizará un *backup* en *Google Drive*.

2.3. Gestión del tiempo

La Gestión del Tiempo incluye procesos enfocados a administrar el proyecto para lograr su finalización a tiempo.

En esta sección se describirá la planificación inicial del proyecto mediante un EDT y la planificación final del mismo, mediante la descripción de los diferentes *sprints* que se realizaron para completar el proyecto.

2.3.1. Estructura de Descomposición del Trabajo (EDT)

El PMBOK describe a la Estructura de Descomposición del Trabajo (EDT) como una descomposición jerárquica orientada al trabajo que es ejecutado por el equipo del proyecto para lograr los objetivos del mismo y crear los entregables requeridos. En otras palabras, una EDT proporciona una representación sencilla del trabajo que debe ser realizado para completar el proyecto. Realizar una EDT antes de iniciar la ejecución del proyecto proporciona grandes ventajas como: mayor determinación del alcance, mayor control sobre los objetivos a cumplir, facilidad para medir el desempeño, etc.

Las unidades de trabajo que identifica la EDT de este proyecto (Figura 2.2) son:

- **Gestión del proyecto:** es la fase del proyecto dedicada a la gestión del mismo: determinar su alcance, planificar tareas, identificar y controlar riesgos, determinar el presupuesto, etc. El mayor esfuerzo en cuanto a planificación es al inicio del proyecto, pero este proceso deberá continuar durante todo el desarrollo, completándolo con detalles que sólo pueden saberse una vez finalice (p.e.: información sobre qué riesgos llegaron a materializarse).
- **Análisis y formación:** durante esta etapa la desarrolladora recopilará información acerca del contexto que rodea a la aplicación a realizar y sobre las tecnologías a usar para su desarrollo. También se formará tanto en aplicaciones de alineamiento de secuencias genéticas como en tecnologías *Big Data*.
- **Diseño:** es la etapa dedicada a estudiar y realizar el mejor diseño posible para la aplicación. Se tratará de buscar un diseño modular que permita separar el código nativo del *BWA* del código correspondiente a *Flink*.
- **Implementación:** la fase de implementación engloba tanto la configuración del entorno de trabajo como la codificación de la propia plataforma *FlinkBWA*.

- **Pruebas de rendimiento:** en esta etapa se diseñarán y realizarán una serie de pruebas de rendimiento para observar los resultados relativos a eficiencia temporal de la aplicación.
- **Documentación:** fase en la que se completarán los contenidos de la memoria del proyecto y se realizará la presentación a utilizar durante la defensa del trabajo.

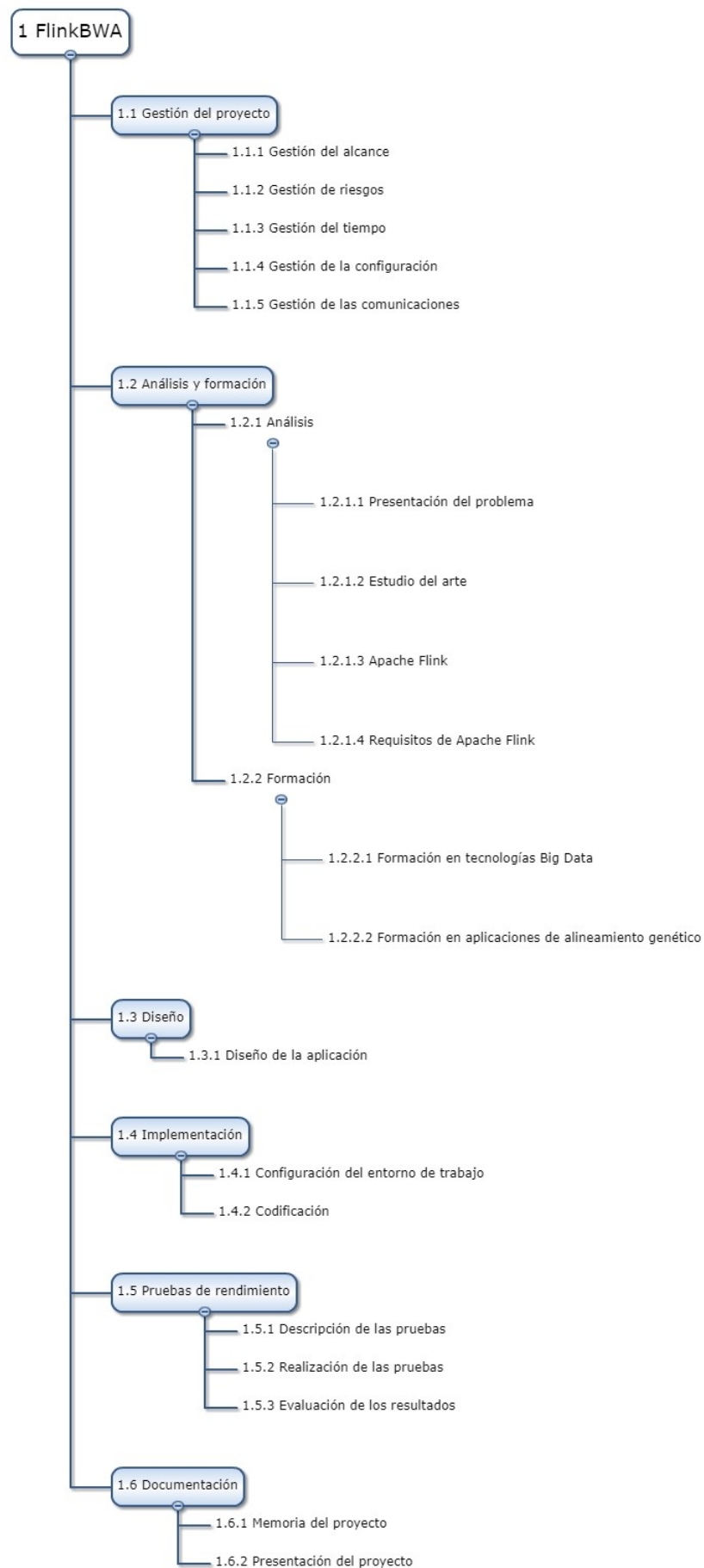


Figura 2.2: Estructura de Descomposición del Trabajo (EDT)

2.3.2. Planificación final: *sprints*

Para describir la planificación final del proyecto, dada la metodología escogida, se ha optado por utilizar gráficos *burn-down* junto con una descripción de las tareas llevadas a cabo en cada *sprint*. Los gráficos *burn-down* permiten mostrar la cantidad de trabajo realizada a lo largo del *sprint* en comparación con la cantidad de trabajo ideal que se debería haber realizado. Se decidió que el proyecto total serían 400 unidades de trabajo (similar al número de horas de trabajo personal que deben realizarse), correspondiendo 100 unidades a cada *sprint* de, en principio, 3 semanas de duración. Por tanto, en el eje X de dichos gráficos se encuentran el número de días del *sprint* y en el eje Y la cantidad de trabajo a realizar.

- ***Sprint 1***: durante el *sprint* inicial se realizó toda la planificación del proyecto junto con la fase de análisis: presentación del problema, introducción y justificación de la tecnología a emplear: *Apache Flink*. Dado que fue una fase de preparación del proyecto y documentación, el trabajo realizado diariamente se ajustó bastante a la planificación, como puede verse en la Figura 2.3.
- ***Sprint 2***: el segundo *sprint* incluyó la realización del diseño de la aplicación, la fase de formación y la configuración del equipo personal para comenzar la implementación de la aplicación. Igual que en el caso del *sprint 1* y como refleja la Figura 2.4, el ritmo de trabajo se ajustó a la planificación y no hubo ningún retardo.

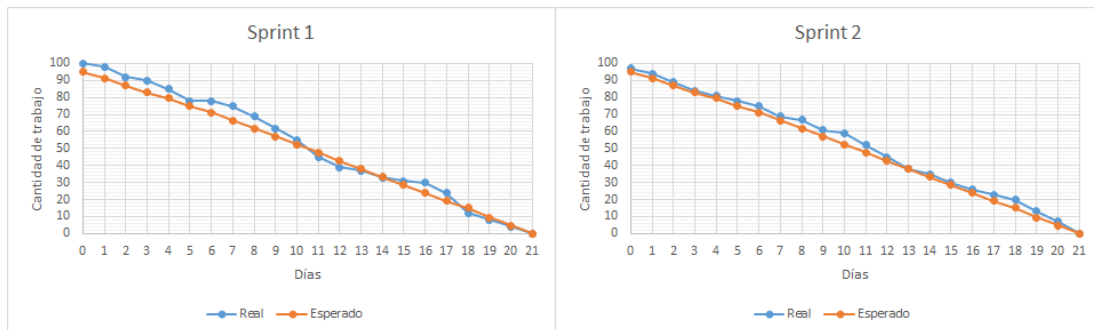
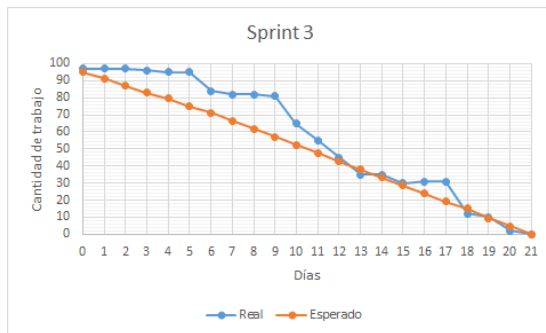
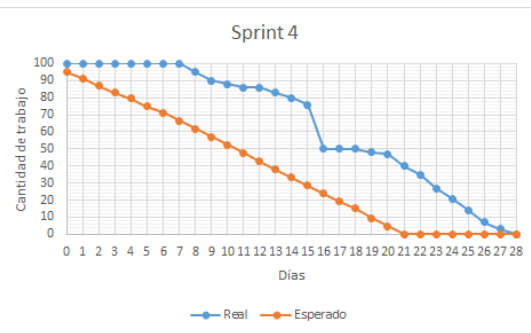


Figura 2.3: *Sprint 1*

Figura 2.4: *Sprint 2*

- ***Sprint 3***: el tercer *sprint* se dedicó íntegramente a la implementación de la aplicación. Como puede verse en la Figura 2.5, los primeros 5 días muestran un ritmo de trabajo lento debido a la falta de experiencia con proyectos *Maven* y con la plataforma *Apache Flink*. Progresivamente, la adaptación a las nuevas tecnologías fue superándose hasta lograr alcanzar la cantidad de trabajo estipulada en el tiempo marcado.

- **Sprint 4:** el último *sprint* del proyecto incluyó la depuración del código, ejecutándolo directamente en el *cluster*, y la realización de pruebas sobre el mismo. Este proceso sufrió prácticamente 1 semana de retardo dado que hubo problemas con la conexión VPN necesaria para acceder al *cluster*. Solventado este problema, se comenzó la depuración del código en este entorno, identificando múltiples errores que necesitaron ser corregidos. Una vez corregidos y siendo el código funcional, se comenzaron las pruebas de rendimiento (aproximadamente en la semana 16). En este punto, se necesitó algo de tiempo para determinar los recursos necesarios para ejecutar el programa con los distintos niveles de paralelismo. Finalmente y como muestra la Figura 2.6, fue necesario ampliar la duración de este *sprint* a 4 semanas en lugar de 3.

Figura 2.5: *Sprint 3*Figura 2.6: *Sprint 4*

2.4. Gestión de las comunicaciones

Gestionar apropiadamente la comunicación dentro de un proyecto es fundamental para tratar de lograr un entendimiento común entre todos los interesados o *stakeholders* del mismo, conectando sus niveles de experiencia, perspectivas e intereses. Esta actividad incluye todos los procesos de vinculados a la correcta generación, recopilación, distribución, almacenamiento, recuperación y disposición final de la información del proyecto.

2.4.1. Identificación de interesados

Un interesado o *stakeholder* es toda aquella persona u organización involucrada de manera activa con el proyecto y, por tanto, sus intereses pueden verse afectados -positiva o negativamente- por el desarrollo o por la finalización del mismo. Es crucial identificar los *stakeholders* en la fase inicial del proyecto, dado que en función de su importancia, sus demandas pueden tener una gran influencia sobre el proyecto.

Dado que este proyecto se corresponde con un Trabajo de Fin de Grado, se asumen como principales interesados al tutor, cotutor y a la desarrolladora, sin tener en cuenta ningún interesado adicional en el futuro. Los siguientes cuadros que se encuentran en esta subsección contienen la información completa de todos los interesados del proyecto. Concretamente, los datos que se mencionan son los siguientes:

- **Nombre:** incluye el nombre completo del interesado.
- **Empresa o grupo:** señala la empresa o grupo al que pertenece el interesado. En este caso, el grupo siempre será la Universidad de Santiago de Compostela.
- **Localización:** lugar en el que se encuentra habitualmente el interesado.
- **Rol en el proyecto:** señala el papel que tiene el interesado dentro del proyecto.
- **Información de contacto:** medio por el cual se puede contactar con el interesado. En este caso, se empleará siempre el correo electrónico como contacto.
- **Requerimientos primordiales:** principales responsabilidades del interesado en relación con el proyecto.
- **Expectativas principales:** conjunto de resultados que el interesado espera del proyecto una vez concluya.
- **Influencia potencial:** impacto que genera el interesado en el proyecto.
- **Fase de mayor interés:** determina en qué fase o fases el interesado cobra mayor importancia.
- **Interno/Externo:** indica si el interesado pertenece o no al grupo que llevará a cabo el proyecto. En este caso, todos los interesados son internos.
- **Apoyo/Neutral/Opositor:** posición del interesado en relación con el proyecto. Si el interesado sirve de apoyo, significa que ayuda al desarrollo del proyecto; un interesado neutral muestra una posición de indiferencia en cuanto a que el proyecto se complete o no; finalmente, un interesado opositor no desea que el proyecto finalice correctamente.

Nombre	Juan C. Pichel Campos
Empresa o grupo	Universidad de Santiago de Compostela (USC)
Localización	Despacho 111, CiTIUS, Campus Vida, Santiago de Compostela
Rol en el proyecto	Tutor
Información de contacto	juancarlos.pichel@usc.es
Requerimientos primordiales	Coordinar, servir de apoyo y resolver las dudas de la desarrolladora.
Expectativas principales	Finalización exitosa del proyecto, cumpliendo los requisitos señalados para el mismo
Influencia potencial	Elevada
Fase de mayor interés	Todo el proyecto
Interno/Externo	Interno
Apoyo/Neutral/Opositor	Apoyo

Cuadro 2.1: Datos del interesado Juan C. Pichel Campos

Nombre	José M. Abuín Mosquera
Empresa o grupo	Universidad de Santiago de Compostela (USC)
Localización	Laboratorio P1, CiTIUS, Campus Vida, Santiago de Compostela
Rol en el proyecto	Cotutor
Información de contacto	josemanuel.abuin@usc.es
Requerimientos primordiales	Servir de apoyo a la implementación y resolver las dudas de la desarrolladora.
Expectativas principales	Finalización exitosa del proyecto, cumpliendo los requisitos señalados para el mismo
Influencia potencial	Alta
Fase de mayor interés	Fase de implementación
Interno/Externo	Interno
Apoyo/Neutral/Opositor	Apoyo

Cuadro 2.2: Datos del interesado José M. Abuín Mosquera

Nombre	Silvia Rodríguez Alcaraz
Empresa o grupo	Universidad de Santiago de Compostela (USC)
Localización	Escuela Técnica Superior de Ingeniería (ETSE), Campus Vida, Santiago de Compostela
Rol en el proyecto	Desarrolladora
Información de contacto	silvia.rodriguez.alcaraz@rai.usc.es
Requerimientos primordiales	Diseñar, desarrollar, estudiar el rendimiento de <i>FlinkBWA</i> y realizar la documentación del proyecto.
Expectativas principales	Finalización exitosa del proyecto, cumpliendo los requisitos señalados para el mismo y pudiendo exponerlo en julio de 2019.
Influencia potencial	Alta
Fase de mayor interés	Todo el proyecto
Interno/Externo	Interno
Apoyo/Neutral/Opositor	Apoyo

Cuadro 2.3: Datos de la interesada Silvia Rodríguez Alcaraz

2.4.2. Plan de gestión de los interesados

Es preciso definir un plan de comunicación y seguirlo a lo largo del proyecto para asegurar que se intercambia información de manera apropiada entre todos los interesados. Este plan debe mostrar la información que se intercambia entre los interesados y ciertos detalles vinculados, como el medio que se usa para la comunicación o el idioma. Se han elaborado dos cuadros que pretenden mostrar los dos flujos principales de comunicación que existen y las principales características de las mismas:

Emisora	Silvia Rodríguez Alcaraz
Receptores	Tutor y/o cotutor
Propósito	Exponer los avances del proyecto, preguntar dudas y/o solicitar reuniones extra.
Nivel de detalle	Alto
Importancia	Muy importante
Periodicidad	3-4 semanas
Formato	Alta
Idioma	Castellano o gallego

Cuadro 2.4: Matriz de comunicación: Desarrolladora → Tutor y/o cotutor

Emisor/es	Tutor y/o cotutor
Receptora	Desarrolladora
Propósito	Guiar el desarrollo, resolver dudas, indicar pautas a seguir, matizar requisitos y preguntar sobre el avance del proyecto.
Nivel de detalle	Alto
Importancia	Muy importante
Periodicidad	3-4 semanas
Formato	Alta
Idioma	Castellano o gallego

Cuadro 2.5: Matriz de comunicación: Tutor y/o cotutor → Desarrolladora

2.5. Gestión de riesgos

Gestionar los riesgos de un proyecto es uno de los aspectos más relevantes de la planificación, ya que multitud de proyectos bien planteados en un inicio fracasan por la aparición de determinadas situaciones no contempladas a priori. De hecho, la gestión de riesgos consiste en planificar, organizar, dirigir y controlar los recursos disponibles con el fin de evitar, minimizar o prever posibles riesgos.

2.5.1. Identificación de riesgos

Los principales riesgos identificados en este proyecto son los que se citan a continuación:

- **RSK-001:** dificultades en el proceso de formación.
- **RSK-002:** falta de comprensión del código nativo del BWA.
- **RSK-003:** desajustes en la planificación.
- **RSK-004:** pérdida de información vinculada al proyecto.
- **RSK-005:** errores en la elección de las tecnologías de desarrollo.
- **RSK-006:** insuficiente disponibilidad de la desarrolladora.
- **RSK-007:** insuficiente disponibilidad de los expertos.
- **RSK-008:** avería en el ordenador de desarrollo.
- **RSK-009:** no disponibilidad de las plataformas *online* empleadas.
- **RSK-010:** no disponibilidad del *clúster*.

2.5.2. Análisis de riesgos

A continuación, se analizarán los riesgos identificados previamente tomando como referencia el formato de la Tabla 2.6:

Identificador del riesgo	
Nombre	Nombre del identificador.
Descripción	Pequeña descripción del riesgo.
Probabilidad	Alta, media o baja.
Impacto	Tolerable, serio o catastrófico.
Indicador	Hechos que ponen de manifiesto la materialización de un riesgo.

Cuadro 2.6: Plantilla para análisis de riesgos

Los aspectos más relevantes a considerar en el análisis son la probabilidad, el impacto y la exposición al riesgo:

- **Probabilidad:** determina las posibilidades que tiene dicho riesgo de ocurrir.
- **Impacto:** señala cuán significativo para el desarrollo normal del proyecto es que dicho riesgo se materialice.
- **Exposición:** muestra la frecuencia de aparición que puede tener el riesgo.

RSK-001	
Nombre	Dificultades en el proceso de formación.
Descripción	Debido a la inexperiencia de la desarrolladora con las tecnologías escogidas para la implementación y con la naturaleza de la propia aplicación, es posible que aparezcan dificultades o dudas a nivel conceptual durante la fase de formación.
Probabilidad	Media
Impacto	Tolerable
Indicador	Aparición de dudas de la desarrolladora y dificultad para avanzar en el proyecto.

Cuadro 2.7: Análisis del RSK-001

RSK-002	
Nombre	Falta de comprensión del código nativo BWA.
Descripción	Debido a la inexperiencia de la desarrolladora en cuanto a aplicaciones de alineamiento de secuencias genéticas es posible que se encuentren dificultades de comprensión a la hora de interpretar y trabajar con el código nativo del BWA.
Probabilidad	Media
Impacto	Tolerable
Indicador	Aparición de dudas de la desarrolladora y dificultad para avanzar en el proyecto.

Cuadro 2.8: Análisis del RSK-002

RSK-003	
Nombre	Desajustes en la planificación.
Descripción	A causa de multitud de factores pueden darse retrasos con respecto a la planificación realizada en un primer momento. Es decir, algunas fases pueden atrasarse o, al contrario, pueden finalizarse tareas antes de lo previsto, desajustándose la planificación inicial.
Probabilidad	Alta
Impacto	Serio
Indicador	Finalización de un <i>sprint</i> sin todos los contenidos previstos finalizados. O bien, finalización de todos los contenidos previstos antes del fin del <i>sprint</i> .

Cuadro 2.9: Análisis del RSK-003

RSK-004	
Nombre	Pérdida de información vinculada al proyecto.
Descripción	Puede ser que por problemas con los medios físicos o con las plataformas <i>online</i> empleadas, se pierda información vinculada al proyecto en un momento dado.
Probabilidad	Media
Impacto	Catastrófico
Indicador	Pérdida de información valiosa y crucial para la entrega a tiempo del proyecto.

Cuadro 2.10: Análisis del RSK-004

RSK-005	
Nombre	Errores en la elección de las tecnologías de desarrollo.
Descripción	Puede ser que exista una incompatibilidad no observada a priori entre la plataforma <i>Big Data</i> a utilizar y la aplicación del alineador de secuencias genéticas u con otros aspectos del entorno de desarrollo.
Probabilidad	Baja
Impacto	Catastrófico
Indicador	Imposibilidad de continuar con la implementación.

Cuadro 2.11: Análisis del RSK-005

RSK-006	
Nombre	Insuficiente disponibilidad de la desarrolladora.
Descripción	Por problemas de carácter personal o la aparición de compromisos inesperados podría suceder que la desarrolladora no contase con todo el tiempo previsto inicialmente para dedicar al proyecto.
Probabilidad	Media
Impacto	Serio
Indicador	Ritmo de trabajo más lento por parte de la desarrolladora.

Cuadro 2.12: Análisis del RSK-006

RSK-007	
Nombre	Insuficiente disponibilidad de los expertos.
Descripción	Por demasiada carga a nivel laboral, problemas de carácter personal o la asistencia a congresos en el extranjero puede ser que la disponibilidad de los expertos se reduzca.
Probabilidad	Media
Impacto	Serio
Exposición	Media
Indicador	Problemas a la hora de comunicarse o fijar reuniones con los expertos.

Cuadro 2.13: Análisis del RSK-007

RSK-008	
Nombre	Avería en el ordenador de desarrollo.
Descripción	Cabe la posibilidad de que el ordenador de desarrollo se averíe por un accidente o simplemente por el uso, ya que no es un equipo demasiado nuevo.
Probabilidad	Media
Impacto	Catastrófico
Indicador	Imposibilidad de avanzar con el proyecto hasta encontrar o conseguir un nuevo equipo, además de la posible pérdida de información local que podría suponer.

Cuadro 2.14: Análisis del RSK-008

RSK-009	
Nombre	No disponibilidad de las plataformas <i>online</i> empleadas.
Descripción	Una caída duradera de las plataformas <i>overleaf</i> y/o <i>Github</i> podrían frenar considerablemente el avance normal del proyecto si no se han hecho copias en local.
Probabilidad	Baja
Impacto	Catastrófico
Indicador	Detención del avance del proyecto hasta que dichas plataformas vuelvan a tener disponibilidad.

Cuadro 2.15: Análisis del RSK-009

RSK-010	
Nombre	No disponibilidad del <i>cluster</i> .
Descripción	El <i>cluster</i> empleado tanto para las pruebas como para dar soporte a la plataforma <i>Hadoop</i> utilizada en la implementación puede sufrir una caída o problemas técnicos.
Probabilidad	Media
Impacto	Serio
Indicador	No se puede acceder a los datos genómicos ni al resto de recursos del <i>cluster</i> .

Cuadro 2.16: Análisis del RSK-010

2.5.3. Planificación de respuesta a los riesgos

Tomando como referencia la probabilidad y el impacto de los riesgos analizados anteriormente, se ha generado una matriz de exposición con el fin de facilitar el

control de los mismos. Dicha matriz muestra la probabilidad que existe de que los riesgos se lleguen a materializar y, por tanto, la elección de técnicas de control sobre los mismos.

		Impacto		
		Catastrófico	Serio	Tolerable
Probabilidad	Alta		RSK-003	RSK-001
		RSK-004	RSK-006	RSK-002
	Media	RSK-008	RSK-007	
			RSK-010	
	Baja	RSK-005		
		RSK-009		

Cuadro 2.17: Matriz de exposición ante riesgos

En función de los resultados mostrados en la Tabla 2.17, se ha decidido controlar todos los riesgos con un nivel de exposición medio (área con fondo amarillo en la matriz), ya que no se detectó ningún riesgo con un nivel de exposición alto. Por tanto, se entiende que ante los riesgos con baja exposición (área de fondo verde) la estrategia de control es la aceptación: no se realizará ninguna acción con respecto a dichos riesgos hasta su activación, en caso de darse.

En cuanto a los riesgos que van a ser controlados (RSK-003, RSK-004, RSK-006, RSK-007, RSK-008 Y RSK-010), las estrategias a seguir son las siguientes:

- **Prevención:** conjunto de acciones a realizar antes de que se materialice un riesgo con el fin de que este no llegue a suceder.
- **Contingencia:** conjunto de acciones a realizar una vez se materializa un riesgo con el fin de reducir o minimizar sus efectos.

Identificador del riesgo	
Nombre del riesgo	Nombre completo del riesgo.
Acciones de prevención	Acciones a aplicar antes de que ocurra el riesgo.
Acciones de contingencia	Acciones a aplicar una vez ocurra el riesgo.

Cuadro 2.18: Plantilla para control de riesgos

A continuación, se presenta el plan de control de riesgos propuesto:

RSK-003	
Nombre del riesgo	Desajustes en la planificación.
Acciones de prevención	Para tratar de evitar posibles desajustes se realizará una planificación basada en <i>sprints</i> que cuente con un margen relativo para posibles imprevistos.
Acciones de contingencia	Si el riesgo se materializa, el margen dedicado a imprevistos de los sprints posteriores deberá reducirse con el fin de ganar algo de tiempo. En el peor de los casos, será necesario recortar el alcance o alguno de los requisitos del proyecto.

Cuadro 2.19: Plan de control para RSK-003

RSK-004	
Nombre del riesgo	Pérdida de información vinculada al proyecto.
Acciones de prevención	Se almacenará todo el proyecto tanto en la plataforma <i>Github</i> como en <i>Google Drive</i> , con el fin de tener respaldados de forma <i>online</i> los contenidos del proyecto, además de en un disco duro externo.
Acciones de contingencia	En caso de que el riesgo llegue a materializarse se accederá a las copias <i>online</i> del mismo o a la copia del disco externo. Teniendo en cuenta que son 3 copias (un par en 2 plataformas <i>online</i> diferentes y la restante en un dispositivo físico) es altamente improbable que ninguna de esas copias sea accesible.

Cuadro 2.20: Plan de control para RSK-004

RSK-006	
Nombre del riesgo	Insuficiente disponibilidad de la desarrolladora.
Acciones de prevención	Tratar de avanzar lo máximo posible de forma diaria para que dicho trabajo compense la pérdida de tiempo que puede generar un imprevisto.
Acciones de contingencia	Dependiendo del punto del proyecto en el que se materialice el riesgo y la duración del mismo en el tiempo se podría: bien, tratar de cubrir el tiempo perdido con el ahorrado en tareas que finalizaron antes; bien, replantear el alcance del proyecto con para tratar de finalizarlo exitosamente en el tiempo establecido.

Cuadro 2.21: Plan de control para RSK-006

RSK-007	
Nombre del riesgo	Insuficiente disponibilidad de los expertos.
Acciones de prevención	Fijar reuniones periódicas, determinadas con suficiente antelación. Acompañar esta medida con contacto frecuente vía correo electrónico.
Acciones de contingencia	Solicitar la atención de otro experto en caso de que la disponibilidad sea alarmantemente reducida.

Cuadro 2.22: Plan de control para RSK-007

RSK-008	
Nombre del riesgo	Avería en el ordenador de desarrollo.
Acciones de prevención	Tratar con el mayor cuidado posible el equipo.
Acciones de contingencia	Tratar de repararlo o buscar un equipo con el que finalizar el proyecto.

Cuadro 2.23: Plan de control para RSK-008

RSK-010	
Nombre del riesgo	No disponibilidad del <i>cluster</i> .
Acciones de prevención	Estudiar la posibilidad de trabajar en local sin el <i>cluster</i> .
Acciones de contingencia	Configurar el entorno <i>hadoop</i> en local y guardar los datos genómicos también en local.

Cuadro 2.24: Plan de control para RSK-010

2.5.4. Materialización de riesgos

Durante la realización del proyecto se materializaron los siguientes riesgos:

- **RSK-004. Pérdida de información vinculada al proyecto:** debido a un fallo en un sector del disco del equipo personal cierta información perteneciente al trabajo se perdió. Afortunadamente, las copias en remoto del proyecto estaban actualizadas y se pudo acceder a ellas por lo que sólo supuso una pérdida a nivel temporal.

- **RSK-008. Avería en el ordenador de desarrollo:** vinculado con la materialización del riesgo RSK-004. Un sector del disco del equipo local se dañó, perdiendo información. Como el resto del equipo funcionaba apropiadamente no fue necesario llevar a cabo ninguna acción de contingencia.
- **RSK-010. No disponibilidad del *cluster*:** problemas con el acceso al mismo y su configuración: falta de directorio de trabajo, permisos, etc. Dado que la aplicación precisa un entorno *hadoop* con varios nodos de trabajo para su ejecución, este hecho supuso desarreglos a nivel temporal en la planificación.

2.6. Gestión de costes

La Gestión de Costes incluye el conjunto de actividades que tienen por objetivo estimar, presupuestar y controlar los costos para que el proyecto pueda finalizarse exitosamente sin exceder el presupuesto fijado.

2.6.1. Consideraciones previas

A continuación se aclaran una serie de aspectos sobre la información que figura en esta sección:

- **Unidad de medida:** la unidad de medida empleada para los costes será el euro (€), dado que el proyecto se desarrolla en España, país perteneciente a la Unión Económica y Monetaria europea (UEM).
- **Nivel de exactitud:** esta sección muestra en su mayoría datos estimados, dado que no se posee información exacta del coste de los recursos humanos y materiales. Por tanto, aunque se han tratado de hacer aproximaciones lo más cercanas a la realidad posible, pueden existir ciertas diferencias con el coste total del proyecto, calculado con todos los datos exactos de los recursos utilizados.
- **Nivel de precisión:** las cifras mostradas sólo llegarán a los céntimos de euro para evitar el uso de fracciones, ya que más bien complican los cálculos en lugar de aportar información relevante. En caso de aparecer una cifra de este tipo se realizará un redondeo de la misma.

2.6.2. Costos directos

Los costos directos de un proyecto son todos aquellos relacionados estrechamente con el producto a realizar. Es decir, todos aquellos materiales tangibles o intangibles empleados para la realización del proyecto. En este caso, se desglosará

el cálculo de los costos directos en: costos de los recursos *materiales* y costos de los Recursos Humanos.

Costos en recursos *materiales*

Los recursos materiales empleados en este proyecto son los siguientes:

- **Equipo de desarrollo:** compuesto por un ordenador portátil, cargador del mismo, ratón USB y en ocasiones una pantalla auxiliar, teclado y adaptador HDMI. El valor del equipo portátil se estima en unos 600 que, al incluir todos los accesorios adicionales, serían finalmente unos 700. La vida media del equipo sería aproximadamente de 4 años (48 meses), utilizándolo para el proyecto un total de 4 meses. Por tanto, el coste en el proyecto de este equipo sería de 58.33 €:

$$\frac{700\text{euros}}{48\text{meses}} = \frac{X\text{euros}}{4\text{meses}} \rightarrow X = 58.33 \text{ €}$$

- **Software:** todo el *software* a utilizar durante el proyecto es libre, o bien, se cuenta con licencia de estudiante para utilizarlo y, por tanto, no supondrá un gasto (como es el caso del IDE de desarrollo: *IntelliJ* de *Jetbrains*).
- **Material fungible:** el costo de folios, carpetas, bolígrafos, fotocopias y demás utilizado para el proyecto se estima en aproximadamente 50 € (incluyendo una copia de la memoria y el soporte digital en el que debe entregarse el *software*).

Costos en *Recursos Humanos*

Los costos relativos a los Recursos Humanos son todos aquellos asociados al equipo del proyecto, en este caso: alumna, tutor y cotutor.

Como no se conocen con exactitud los sueldos de los componentes del equipo, por una parte, para determinar los sueldos del tutor y cotutor se consultaron las Tablas de Retribuciones del Personal Docente e Investigador (PDI) de la USC [9]. Así, suponiendo para el tutor el sueldo mensual de un profesor contratado doctor (2506,81 €) y para el cotutor el de profesor ayudante doctor (1957,56 €), solo faltaría calcular el costo por horas:

$$\text{Tutor} \rightarrow \frac{2506,81 \text{ euros}}{22 \text{ días/mes} * 8 \text{ h}} = 14,24 \text{ €/h} \quad \text{Cotutor} \rightarrow \frac{1957,56 \text{ euros}}{22 \text{ días/mes} * 8 \text{ h}} = 11,12 \text{ €/h}$$

Por otra parte, para determinar el sueldo de la alumna se consultaron las Bases y tipos de cotización de la Seguridad Social [10], observando que el sueldo mínimo para ayudantes no titulados es de 1050€/mes. Calculando el costo por horas para este caso:

$$\text{Alumna} \rightarrow \frac{1050 \text{ euros}}{22 \text{ días/mes} * 8 \text{ h}} = 5,97 \text{ €/h}$$

A continuación, se presenta la Tabla 2.25 que contiene el cálculo total de los costos que suponen los Recursos Humanos del proyecto:

Costos Recursos Humanos					
Recurso	Rol	Sueldo (€ / mes)	Costo (€/hora)	Horas	Costo total (€)
Tutor	Profesor contratado doctor	2506,81	14,24	11,25	160,2
Cotutor	Profesor ayudante doctor	1957,56	11,12	11,25	126
Alumna	Desarrolladora analista	1050	5,97	412,99	2465,55
Costo Total RRHH (€)					2751,75

Cuadro 2.25: Costos de los Recursos Humanos

2.6.3. Costos indirectos

Los costos indirectos son aquellos que no tienen un impacto directo sobre el proyecto, pero que son indispensables para la realización del mismo. Por tanto, se considerarían costos indirectos el gasto en luz, labores administrativos, etc.

Para determinar los costos indirectos de este proyecto se ha tomado como referencia el valor que establece de forma oficial la USC para los proyectos TIC [11], que es un 21 % adicional sobre el valor de los costos directos.

Teniendo en cuenta que el valor de los costos directos es de 2751,75€, los costos indirectos de este proyecto supondrían un total de 617,42€.

2.6.4. Financiamiento

Dada la naturaleza del presente proyecto, un Trabajo de Fin de Grado (TFG), no habrá ningún tipo de pago por parte de ningún cliente.

Por tanto, en este contexto concreto, se ha omitido la realización de cálculos en cuanto al desembolso inicial, la realización de gráficos de flujo de caja (gráficos que muestran cobros y pagos en un determinado período) y demás técnicas empleadas para controlar la financiación de un proyecto.

2.6.5. Coste total

Los costes totales del proyecto serían el resultado de la suma de los costes directos y los indirectos que, como ya se ha mencionado, suponen el 21 % de los mismos. La Tabla 2.26, que se presenta a continuación, incluye el recuento de todos los costes del proyecto:

Costes del proyecto		
Nombre	Tipo de coste	Coste (€)
Equipo de desarrollo	Directo	58,33
Material fungible	Directo	50
RRHH	Directo	2751,75
Total costes directos (€)		2860,08
Costes indirectos (€)		617,42
Coste Total (€)		3477,50

Cuadro 2.26: Costes del proyecto

Capítulo 3

Especificación de requisitos

Según el PMBOK, se conoce como requisitos al conjunto de condiciones o capacidades que debe tener un sistema, producto, servicio o componente para satisfacer un contrato, estándar, especificación, u otros documentos formalmente establecidos. Es, por tanto, fundamental y necesario cumplir con los requisitos del proyecto para alcanzar el éxito del mismo.

Este capítulo está dedicado a la identificación y descripción de los requisitos que debe cumplir el proyecto para considerarse finalizado con éxito.

3.1. Definición del sistema

Esta memoria se corresponde con un proyecto que no es exactamente un desarrollo de *software* enfocado a un usuario final, sino que trata de incluir una tecnología concreta en una aplicación ya existente y funcional con el fin de mejorar su eficiencia temporal.

La Figura 3.1 pretende mostrar, con gran nivel de abstracción, el funcionamiento de la aplicación para tratar de simplificar la identificación de requisitos. Dicha aplicación precisa una serie de parámetros y datos de entrada, que determinan el comportamiento de la misma. Su núcleo se basaría en el funcionamiento del BWA nativo, paralelizado por medio de la utilización de la tecnología *Big Data Apache Flink*. Finalmente, el *output* del programa sería un fichero en formato SAM.

Como puede observarse, es una aplicación que carece de GUI (*Graphic User Interface*), por lo que su ejecución se realiza íntegramente mediante la consola de comandos, empleando conexión SSH para acceder al *cluster* donde está alojada.

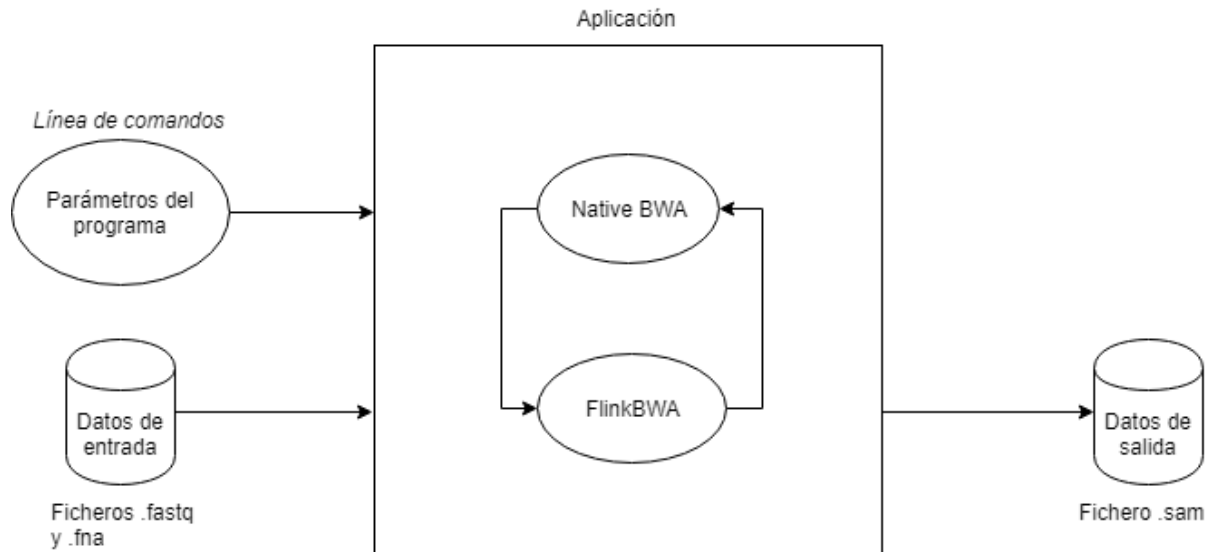


Figura 3.1: Modelo funcional de la aplicación

3.2. Casos de uso

Los casos de uso describen acciones o actividades. Un diagrama de casos de uso representa un sistema o subsistema como un conjunto de interacciones entre casos de uso y entre dichos casos y sus pertinentes actores en respuesta a un evento iniciado por un actor principal.

Por su parte, en el Lenguaje Unificado de Modelado (UML) [12], un actor se emplea para modelar un rol jugado por un usuario, entidad, *hardware* u otro que interactúa con el sistema. En este caso concreto, en principio, sólo la desarrolladora y los expertos interactuarán con el sistema, pues lo que se busca dentro de este proyecto es estudiar su rendimiento.

3.2.1. Identificación de casos de uso

Aclarados los conceptos previos, se ha concluido que en la presenta aplicación sólo tiene sentido hablar de un actor: el usuario que manipula la aplicación vía línea de comandos. A continuación, la Figura 3.2 muestra el diagrama con los casos de uso identificados:

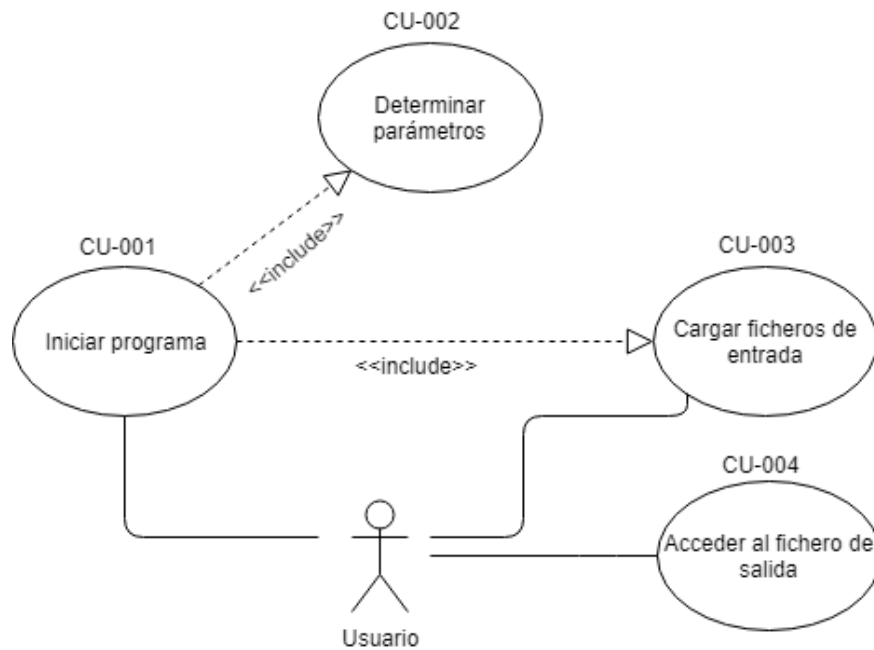


Figura 3.2: Diagrama de casos de uso

3.2.2. Especificación de casos de uso

CU-001	Iniciar programa
Descripción	El usuario de la aplicación debe ser capaz de ejecutar el programa en cualquier momento que lo desee siempre y cuando el entorno de ejecución (el <i>cluster</i> correspondiente) esté disponible.
Importancia	Vital.

Cuadro 3.1: Especificación CU-001

CU-002	Determinar parámetros
Descripción	El usuario debe poder modificar los parámetros de entrada que acepta el programa (algoritmo a emplear, n° de ficheros de entrada, etc) con el fin de modificar el comportamiento del mismo.
Importancia	Vital.

Cuadro 3.2: Especificación CU-002

CU-003	Cargar ficheros de entrada
Descripción	El usuario debe poder especificar la ruta donde se encuentran los ficheros de entrada necesarios para la ejecución de la aplicación.
Importancia	Vital.

Cuadro 3.3: Especificación CU-003

CU-004	Acceder al fichero de salida
Descripción	El usuario debe ser capaz de acceder al fichero de salida que genera el programa con el fin de ver su contenido.
Importancia	Vital.

Cuadro 3.4: Especificación CU-004

3.3. Catálogo de requisitos

En esta sección se identificarán los principales requisitos funcionales y no funcionales detectados en las reuniones iniciales del proyecto y durante la fase de análisis. Además, se describirán dichos requisitos en base a su importancia y a sus criterios de aceptación. Cabe destacar que se distinguen los siguientes tres niveles de importancia:

Nivel de importancia	Descripción
Vital	Su cumplimiento es fundamental para la finalización exitosa del proyecto.
Importante	Su cumplimiento afecta únicamente de cara a incrementar el valor del proyecto, pero su no cumplimiento no supondría una pérdida sustancial.
Deseable	Su cumplimiento aumentaría la calidad del proyecto, pero no es obligatorio cumplirlo.

Cuadro 3.5: Niveles de importancia

La plantilla que se seguirá para la especificación de los requisitos es la que se muestra en la Tabla 3.6:

ID	Nombre
Descripción	Breve explicación sobre qué implica y en qué consiste el requisito.
Importancia	Determina la importancia del requisito.
Criterios de aceptación	Menciona cuáles serán los criterios que validan el cumplimiento del requisito.

Cuadro 3.6: Plantilla de especificación de requisitos

3.3.1. Requisitos funcionales

Los requisitos funcionales son aquellos que identifican funcionalidades que definen al *software* a realizar. Cabe decir que este caso concreto es ligeramente diferente al de un desarrollo partiendo de cero, dado que el punto de partida es una implementación ya funcional del BWA.

Identificación de requisitos funcionales

- **RF-001:** paralelización del BWA con *Apache Flink*.
- **RF-002:** acceso a los ficheros de entrada.
- **RF-003:** modificación de parámetros.
- **RF-004:** generación de un fichero *.sam* como *output*.
- **RF-005:** interfaz de usuario basada en la línea de comandos.

Especificación de requisitos funcionales

RF-001	Paralelización del BWA con Apache Flink
Descripción	Implementación del BWA con una tecnología <i>Big Data</i> , en este caso <i>Apache Flink</i> , que permita paralelizar el trabajo entre varios nodos de computación, aprovechando la infraestructura para tratar de mejorar su eficiencia temporal.
Importancia	Vital.
Criterios de aceptación	Se considera validado cuando se tenga una versión final y funcional de FlinkBWA.

Cuadro 3.7: Especificación RF-001

RF-002	Acceso a los ficheros de entrada
Descripción	El programa debe poder acceder a los ficheros .fastq, con las secuencias de ADN, y al fichero .fna que contiene la referencia del genoma humano.
Importancia	Vital.
Criterios de aceptación	Se considera validado en el momento que la aplicación es capaz de aceptar los ficheros de entrada y procesar su información sin mostrar problemas.

Cuadro 3.8: Especificación RF-002

RF-003	Modificación de parámetros
Descripción	El programa debe permitir al usuario modificar una serie de parámetros que determinan su comportamiento.
Importancia	Vital.
Criterios de aceptación	Se considera validado en el momento en que las opciones del programa funcionan correctamente.

Cuadro 3.9: Especificación RF-003

RF-004	Generación de un fichero .sam como <i>output</i>
Descripción	La ejecución del programa debe dar como resultado un fichero .sam con el resultado de todas las operaciones realizadas sobre las secuencias de ADN.
Importancia	Vital.
Criterios de aceptación	Se considera validado en el momento en que se comprueba que la salida del programa es correcta y posee el formato adecuado.

Cuadro 3.10: Especificación RF-004

RF-005	Interfaz de usuario basada en la línea de comandos
Descripción	El programa debe proporcionar mensajes que interactúen directamente con el usuario por línea de comandos y una opción de ayuda que facilite su uso.
Importancia	Vital.
Criterios de aceptación	Se considera validado en el momento en que el usuario sea capaz de acceder a la ayuda del programa y pueda interactuar correctamente con el programa por línea de comandos.

Cuadro 3.11: Especificación RF-005

3.3.2. Requisitos no funcionales

Los requisitos no funcionales son aquellos que recogen características generales y restricciones que afectan a la aplicación que se está desarrollando. Cuestiones vinculadas a la seguridad o el rendimiento del programa serían un buen ejemplo de requisitos no funcionales. A continuación, se identifican y detallan los requisitos no funcionales de este proyecto.

Identificación de requisitos no funcionales

- **RNF-001:** extensibilidad.
- **RNF-002:** escalabilidad.
- **RNF-003:** mejora de la eficiencia temporal del BWA.
- **RNF-004:** utilización de software libre o gratuito con fines educativos.
- **RNF-005:** utilización de Java como lenguaje de desarrollo.

Especificación de requisitos no funcionales

RNF-001	Extensibilidad
Descripción	El diseño de la aplicación debe ser modular, de forma que desacople la implementación del BWA nativo del código introducido para lograr su paralelización mediante <i>Flink</i> .
Importancia	Importante.
Criterios de aceptación	Se considera validado en el momento que se consiga una implementación funcional con un diseño modular que desacople el BWA nativo del código añadido.

Cuadro 3.12: Especificación RNF-001

RNF-002	Escalabilidad
Descripción	La paralelización del programa debe permitir que conforme se incrementa el número de particiones del fichero o ficheros de entrada entre los distintos nodos de trabajo el tiempo de cómputo debe reducirse progresivamente.
Importancia	Importante.
Criterios de aceptación	Se considera validado en el momento en que se realicen pruebas de rendimiento y se verifique que el tiempo disminuye conforme se aumenta el número de particiones.

Cuadro 3.13: Especificación RNF-002

RNF-003	Mejora de la eficiencia temporal del BWA
Descripción	Tratar de mejorar la eficiencia temporal del alineador gracias a su paralelización mediante <i>Flink</i> .
Importancia	Deseable.
Criterios de aceptación	Se considera validado en el momento en que se compruebe que su tiempo de ejecución se ha mejorado gracias a la paralelización del trabajo.

Cuadro 3.14: Especificación RNF-003

RNF-004	Utilización de software libre o gratuito con fines educativos
Descripción	Se deberá emplear durante todo el proyecto software considerado open-source o libre para fines estudiantiles.
Importancia	Deseable.
Criterios de aceptación	Se considera validado hasta que surja la necesidad de pagar una cantidad determinada por el uso de un servicio o producto necesario para el desarrollo del proyecto.

Cuadro 3.15: Especificación RNF-004

RNF-005	Utilización de Java como lenguaje de desarrollo
Descripción	<i>Apache Flink</i> acepta los lenguajes <i>Scala</i> , <i>Java</i> y <i>Python</i> (este último todavía en fase <i>beta</i>). Puesto que la desarrolladora tiene experiencia desarrollando en Java, su utilización podría simplificar la fase de implementación.
Importancia	Deseable.
Criterios de aceptación	Se considera validado en el momento que se finalice el desarrollo de FlinkBWA empleando este lenguaje.

Cuadro 3.16: Especificación RNF-005

Capítulo 4

Análisis

Este capítulo está dedicado a introducir al lector al contexto que engloba el presente trabajo, que sería: por una parte, las aplicaciones de alineamiento genético; por otra, el mundo de las tecnologías *Big Data*. Se hará especial hincapié en presentar tanto el alineador de secuencias genéticas como la plataforma *Big Data* que se emplean en este proyecto: el BWA y *Apache Flink*.

4.1. Aplicaciones de alineamiento genético

La biotecnología es un área multidisciplinar en auge con múltiples aplicaciones en la actualidad. Tomando como referencia el Artículo 2 de la “Convención sobre la diversidad biológica” de las Naciones Unidas de 1992 [13], se puede definir la biotecnología como toda aquella aplicación tecnológica que utilice sistemas biológicos y organismos vivos o sus derivados para la creación o modificación de productos o procesos para usos específicos.

Una sus múltiples aplicaciones es el alineamiento de secuencias genéticas. En los ámbitos de la bioquímica, genética y biología molecular, se conoce como homología [14] la similitud entre dos o más secuencias de proteínas o ácidos nucleicos por mostrar un mismo origen evolutivo. Determinar estas homologías proporciona información de gran utilidad: una similitud puede señalar relaciones funcionales o evolutivas entre los genes o proteínas consultados; además, si tienen un ancestro común, las no coincidencias pueden interpretarse como mutaciones puntuales. Cabe suponer la notoria cantidad de información que reside en dichas secuencias, por lo que, para encontrar rápidamente discrepancias o similitudes entre ellas se han llevado a cabo aplicaciones bioinformáticas que facilitan potencialmente esta comparación.

Los alineadores de secuencias genéticas se encargan de mapear los datos almacenados contra una secuencia de referencia conocida y previamente indexada,

con el fin de acelerar las búsquedas de los algoritmos de alineamiento. Existen distintos tipos de alineadores según el algoritmo que emplean internamente, el uso de memoria que requieren, su velocidad, etc. Según el algoritmo que utilizan, se distinguen dos grupos [15]: los basados en *hashing* y los basados en la transformación *Burrows-Wheeler* [16]. Teniendo en cuenta que en este proyecto se trabaja con un alineador basado en este último algoritmo, la explicación se centrará en este segundo grupo.

Los alineadores *Burrows-Wheeler* se basan en este algoritmo, el cual consiste en la transformación de una cadena de caracteres en otra mucho más sencilla de comprimir. Dicho reordenamiento permuta los caracteres, haciendo que la cadena resultante agrupe los caracteres similares en la cadena original, siendo más sencillo comprimirla. Por tanto, se puede decir que es un algoritmo de compresión de datos sin pérdida, ya que los caracteres no cambian su valor tras la transformación. Otro punto a favor de la transformación es que su resultado es reversible, por lo que a partir del resultado final es posible volver a la cadena inicial realizando una serie de operaciones. Recientemente, se han desarrollado varios alineadores que emplean este algoritmo para tratar de reducir el uso de memoria que se emplea en los alineamientos, con el que se trabaja en el proyecto actual es el *Burrows-Wheeler Aligner* (BWA).

4.1.1. *Burrows-Wheeler Aligner*

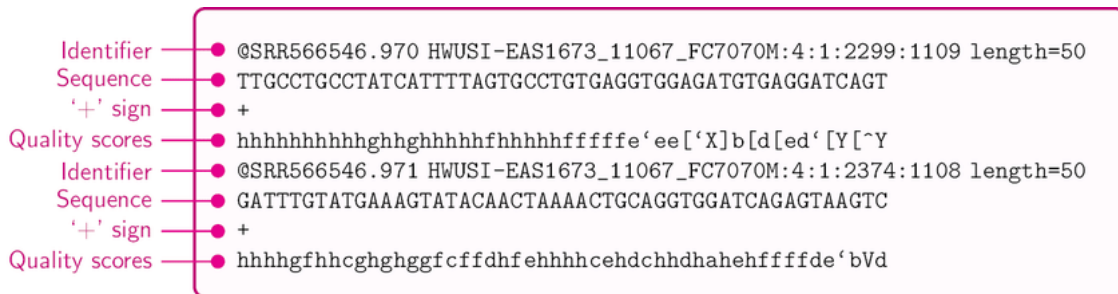
El *Burrows-Wheeler Aligner* (BWA) [17] es un paquete de *software* libre que usa la transformación *Burrows-Wheeler* (BWT, *Burrows-Wheeler Transformation*) para indexar un genoma de referencia grande, como el humano, y mapear secuencias de baja divergencia (es decir, muy similares entre sí) contra dicho genoma.

Este alineador utiliza tres algoritmos para conseguir optimizar el alineamiento y la rapidez del mismo: BWA-*backtrack* [18], BWA-SW [19] y BWA-MEM [20]. El primero, está diseñado para leer secuencias de un tamaño pequeño (hasta 100 pares de bases), mientras que los otros dos se centran en secuencias más largas. BWA-MEM es el más reciente y ha mostrado mejor rendimiento en comparación con otros alineadores de lectura de última generación a la hora de mapear lecturas de 100 pares de bases o más.

Dado que la alineación de secuencias es un proceso costoso a nivel temporal, como ya se ha mencionado, BWA cuenta con una implementación paralela que permite reducir este tiempo. El problema es que dicha implementación sólo es compatible con máquinas de memoria compartida, haciendo que la escalabilidad

siempre esté limitada por el número de núcleos y la memoria disponible en el nodo de computación.

El BWA puede aceptar como entrada tanto archivos BAM [21] no alineados como archivos en formato FASTQ. El formato BAM es una versión comprimida del formato SAM, que permite realizar un indexado para tener acceso directo a las posiciones genómicas. Básicamente, un fichero SAM (*Sequence Alignment/Map format*) se compone de texto tabulado cuyo fin es la representación de alineamientos de secuencias contra un genoma o secuencia de referencia, compuesto por una sección de cabecera (opcional) y una sección de alineamiento. Por su parte, el formato FASTQ es su entrada habitual y puede considerarse el *input* estándar de las herramientas bioinformáticas encargadas del alineamiento. A continuación, se incluye un ejemplo que muestra la estructura de este tipo de ficheros:



```

Identifier —● @SRR566546.970 HWUSI-EAS1673_11067_FC7070M:4:1:2299:1109 length=50
Sequence —● TTGCCTGCCTATCATTTAGTGCCTGTGAGGTGGAGATGTGAGGATCAGT
'+' sign —● +
Quality scores —● hhhhhhhhhghghghhhhhfhhhhhhfffffe'ee['X]b[d[ed'[Y[~Y
Identifier —● @SRR566546.971 HWUSI-EAS1673_11067_FC7070M:4:1:2374:1108 length=50
Sequence —● GATTTGTATGAAAGTATACAACATAAACTGCAGGTGGATCAGAGTAAGTC
'+' sign —● +
Quality scores —● hhhhgfhhcghghggfcffdhfehhhhcehdchhdhahehffffde'bVd
  
```

Figura 4.1: Ejemplo del formato FASTQ

La Figura 4.1 contiene dos lecturas, dado que en cuatro líneas se describe una secuencia o lectura, incluyendo como información: un identificador único, la secuencia e información de calidad de la lectura, separada del resto de datos por el símbolo “+”.

Cabe destacar que este alineador es capaz de aceptar como entrada un único fichero FASTQ (lectura de un sólo extremo) o dos archivos FASTQ (lecturas de extremo emparejado). En el segundo caso, están disponibles dos secuencias que se corresponden a los dos extremos de un mismo fragmento de ADN. Dichas lecturas se corresponden con distintos ficheros de entrada, como ya se mencionó, pero comparten identificador.

El formato SAM mencionado anteriormente es, de hecho, el formato que emplea el BWA como *output*, ya que es más legible que el BAM de cara a ser interpretado por un humano.

4.2. Estado del arte: tecnologías *Big Data*

Pese a que hoy por hoy el término *Big Data* tiene un uso extendido y sus tecnologías están en pleno auge, no existe un consenso claro a la hora de dar una definición exacta. Muchos profesionales determinan que el *Big Data* recoge todos los procesos de extracción, transformación y carga (operaciones ETL: *extraction, transform and load*) que se realizan sobre grandes volúmenes de datos. Otra descripción comúnmente utilizada se basa en tres atributos de los datos, conocidos como “3Vs”: volumen, velocidad y variedad. Pero ninguna de estas definiciones engloba realmente todos los aspectos que implica el término *Big Data*. Por lo pronto, en lo que respecta a este trabajo, basta con entender el término *Big Data* como el conjunto de tecnologías específicas que facilitan la tarea de almacenamiento, procesamiento y análisis de datos cuyo volumen y/o complejidad supera las capacidades de cómputo de una máquina convencional.



Figura 4.2: Cronograma sobre la evolución de *Big Data*

En la Figura 4.2 extraída del libro “*Big Data: Principles and Paradigms*” [22] se muestra un cronograma que recoge los hitos más importantes de la historia del

Big Data hasta el año 2015.

Se podría concluir que estas tecnologías surgen a partir de la necesidad de las principales empresas tecnológicas de finales de los 90 y principios del 2000 de explotar su banco de datos para potenciar su negocio (lo que hoy se conoce como *Data Mining*). Un ejemplo sería el caso de *Google*, quien desarrolló su propia solución para automatizar eficientemente la distribución de grandes volúmenes de datos en un conjunto de máquinas. Dicho *software* fue bautizado como *MapReduce* y permitió reducir las tareas del programador a definir los detalles de dos funciones: “map” y “reduce”, las cuales aluden a las dos etapas principales del procesamiento de datos. La primera se encargaría de transformar un conjunto de datos inicial en otro conjunto intermedio estructurado de la forma “clave-valor” y, la segunda, procesa los valores de los datos intermedios correspondientes a una misma clave proporcionando un resultado final. En la Figura 4.3 [23] se muestra un ejemplo de este modelo de programación para contar el número de apariciones de ciertas palabras en un documento:

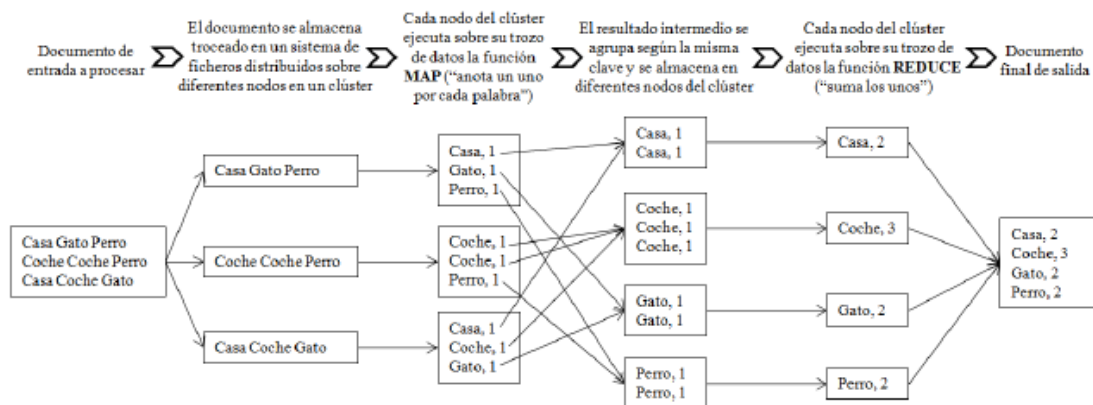


Figura 4.3: Ejemplo del modelo *Map Reduce*

La publicación de *MapReduce* por parte de *Google* inspiró a otros proyectos similares. Es el caso de *Apache Hadoop*¹, un sistema que implementa *MapReduce*, siendo capaz de procesar de forma distribuida enormes volúmenes de datos. Esta plataforma de código abierto facilitó la creación de nuevas tecnologías que aprovechaban su funcionalidad, de hecho, hoy en día existe un gran ecosistema de herramientas que lo utilizan como base. Algunas de estas herramientas serían: *Apache Pig*, *Apache Hive*, *Apache Storm*, etc.

Con el tiempo, el modelo *MapReduce* empezó a presentar limitaciones en cuanto a su eficiencia en determinadas situaciones. Consecuencia de este hecho, empiezan

¹<https://hadoop.apache.org/>

a abrirse nuevas vías de desarrollo. Una, orientada a mejorar dicho modelo; y otras, orientadas a crear modelos alternativos de procesamiento que mantengan los principios de *MapReduce* pero mejoren sus capacidades. Como ejemplo de la primera vía se podría destacar la evolución de *Hadoop*, quien mejora su estructura y gestión interna gracias al nuevo gestor de recursos denominado YARN². En la segunda vía, se destacaría la aparición de *Apache Spark*³ en 2010, basado en un uso prioritario de la memoria principal frente al almacenamiento en disco y la utilización de la abstracción de datos *Resilient Distributed Datasets* (RDD⁴) para ofrecer mejores resultados que *Hadoop* ante procesos iterativos o casos de consultas extremos.

4.2.1. *Apache Flink*

Como se mencionaba anteriormente, la aparición de tecnologías de almacenamiento de grandes volúmenes de datos, como *Hadoop* cambió totalmente la dinámica de dichas empresas que, en lugar de descartarlos, pudieron empezar a trabajar con ellos. Posteriormente, aparecieron herramientas cuyo objetivo era el procesamiento de estos datos, como: *Pig*, *Hive* y *Spark*.

Es en este punto donde las empresas descubrieron que la gran cantidad de datos con los que contaban se debía a que los recibían de manera continua en forma de flujos, por lo que el procesamiento por lotes no bastaba para extraer el máximo potencial a dichos datos. Por tanto, aparecen las soluciones de *streaming*, destacando *Apache Spark Streaming*, *Apache Storm* y, por supuesto, *Apache Flink*.

Apache Flink es un *framework* de código abierto desarrollado por la *Apache Software Foundation*, cuya primera versión oficial data del año 2015. *Flink* permite el procesamiento de datos tanto en flujos como en lotes. Cabe destacar que las soluciones para el procesamiento de flujos de datos deben enfrentarse a una serie de problemas que no afectaban a las herramientas de procesamiento por lotes [24]:

- Las transformaciones (p.e.: troceado de palabras) y el procesamiento de los datos deben tener en cuenta que estos llegarán de manera indefinida, ya no se puede esperar a leer todas las líneas para empezar a tratar la información.
- En caso de indisponibilidad del servicio, se deben disponer de mecanismos para no perder ningún dato del flujo. Es necesario separar la etapa de adquisición y la de procesamiento.
- Existen requisitos de tiempo de procesamiento.

²<https://yarnpkg.com/lang/en/>

³<https://spark.apache.org/>

⁴<https://spark.apache.org/docs/1.6.2/api/java/org/apache/spark/rdd/RDD.html>

- Resistencia frente a fallos que asegure que cada evento sólo se procesa una vez (mecanismos de sincronización y control de errores).

La principal diferencia entre *Flink* y tecnologías similares es que su implementación es la que, según varios estudios, mejor resuelve dichos problemas sin perder capacidad y velocidad de procesamiento. *Flink* ofrece un procesamiento *streaming* nativo, mientras que *Spark*, quien fue ideado en un principio sólo para el procesamiento de datos por lotes, se ha adaptado al procesamiento en *streaming* utilizando *micro-batching*.

En este punto, cabe recordar que la mejor tecnología siempre será la que mejor se adapte a las condiciones del problema concreto que se esté resolviendo [25] y puede ser una tecnología u otra según el caso. De todas formas, lo cierto es que *Flink* presenta grandes ventajas en cuanto al procesamiento de flujos de datos, pero todavía no cuenta con la gran comunidad de usuarios que posee, por ejemplo, *Apache Spark*. Por este motivo, este trabajo pretende ser un estudio más que determine las posibilidades de esta plataforma *Big Data*.

Capítulo 5

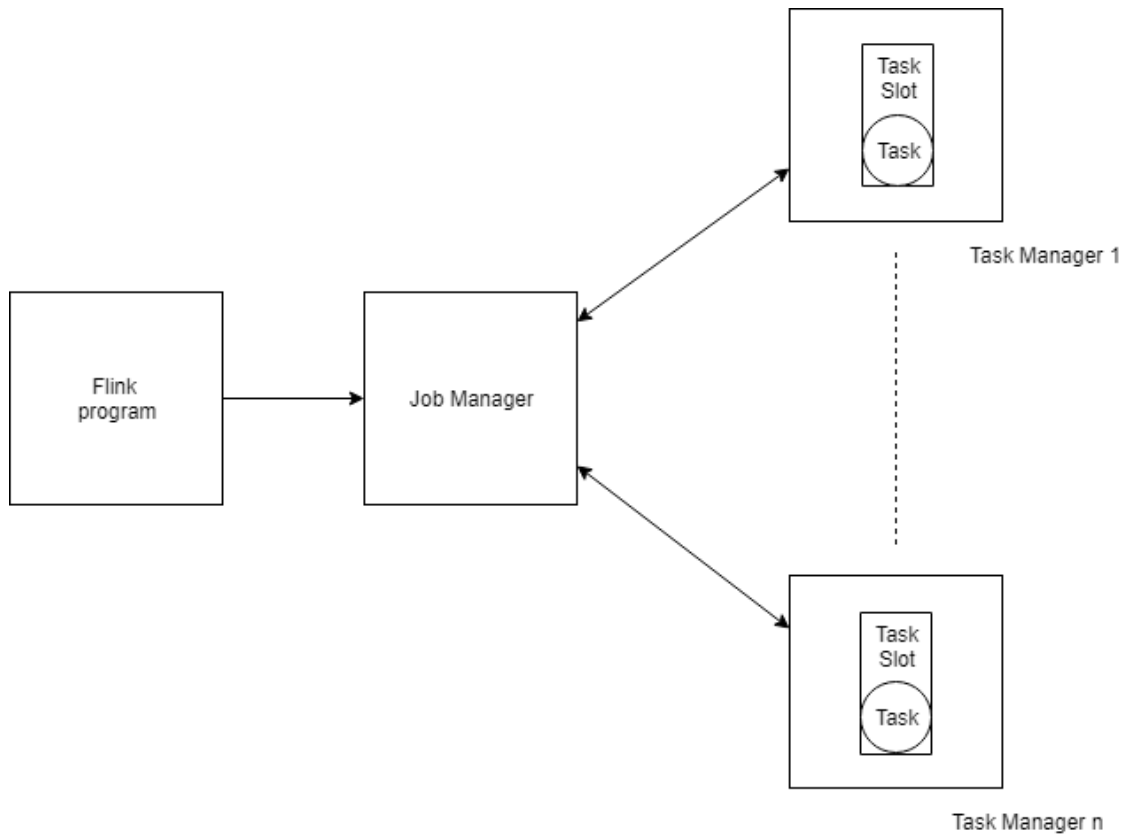
Diseño

Este capítulo está dedicado a cuestiones relacionadas con el diseño de la aplicación, describiendo su estructura interna mediante una serie de diagramas y figuras que pretenden facilitar su comprensión.

5.1. Arquitectura

Para comprender en mayor medida la paralelización del alineamiento de datos llevada a cabo, es necesario explicar cuál es la arquitectura de trabajo interna de *Flink*. Sus elementos principales aparecen representados en la Figura 5.1 y son los siguientes:

- ***Job Managers*:** son los encargados de gestionar la ejecución distribuida programando tareas, coordinando puntos de control, etc. Siempre hay por lo menos un *Job Manager* por ejecución, aunque en entornos de alta disponibilidad suele haber más.
- ***Task Managers*:** también denominados “workers” son los encargados de ejecutar las tareas o, más bien, subtareas en caso de haber más de uno, como suele ser el caso.
- ***Task Slots*:** dado que cada *Task Manager* es un proceso JVM, puede ejecutar una o más subtareas utilizando distintos hilos. Para determinar el número de tareas que puede realizar un *Task Manager* posee los denominados *Task Slots* o ranuras de trabajo que, por defecto, es sólo una. Cada una de estas ranuras tendría una cantidad de recursos finita del *Task Manager*, es decir, los recursos totales se repartirán equitativamente entre las distintas ranuras. Cabe mencionar que las tareas ejecutadas en un mismo *Task Manager* o proceso JVM comparten conexiones TCP y pueden incluso compartir *datasets* y estructuras de datos, reduciendo la sobrecarga por tarea.

Figura 5.1: Arquitectura de *Flink*

En el caso concreto de *FlinkBWA*, sólo se emplea un *Job Manager*, puesto que no se precisa recrear un entorno de alta disponibilidad. Además, el número de *Task Managers* a utilizar coincidirá con el número de particiones indicado para el conjunto de datos de entrada, contando con sólo una ranura de datos por *Task Manager*. Esto es, cada *Task Manager* realizará una parte del proceso de alineamiento, paralelizándolo, y dando lugar a un conjunto de resultados parciales que podrán unificarse en un único fichero según las opciones del programa introducidas por el usuario.

5.2. Diagrama de clases

En la Figura 5.2 se pueden observar las principales clases que conforman la aplicación y cómo se relacionan entre sí:

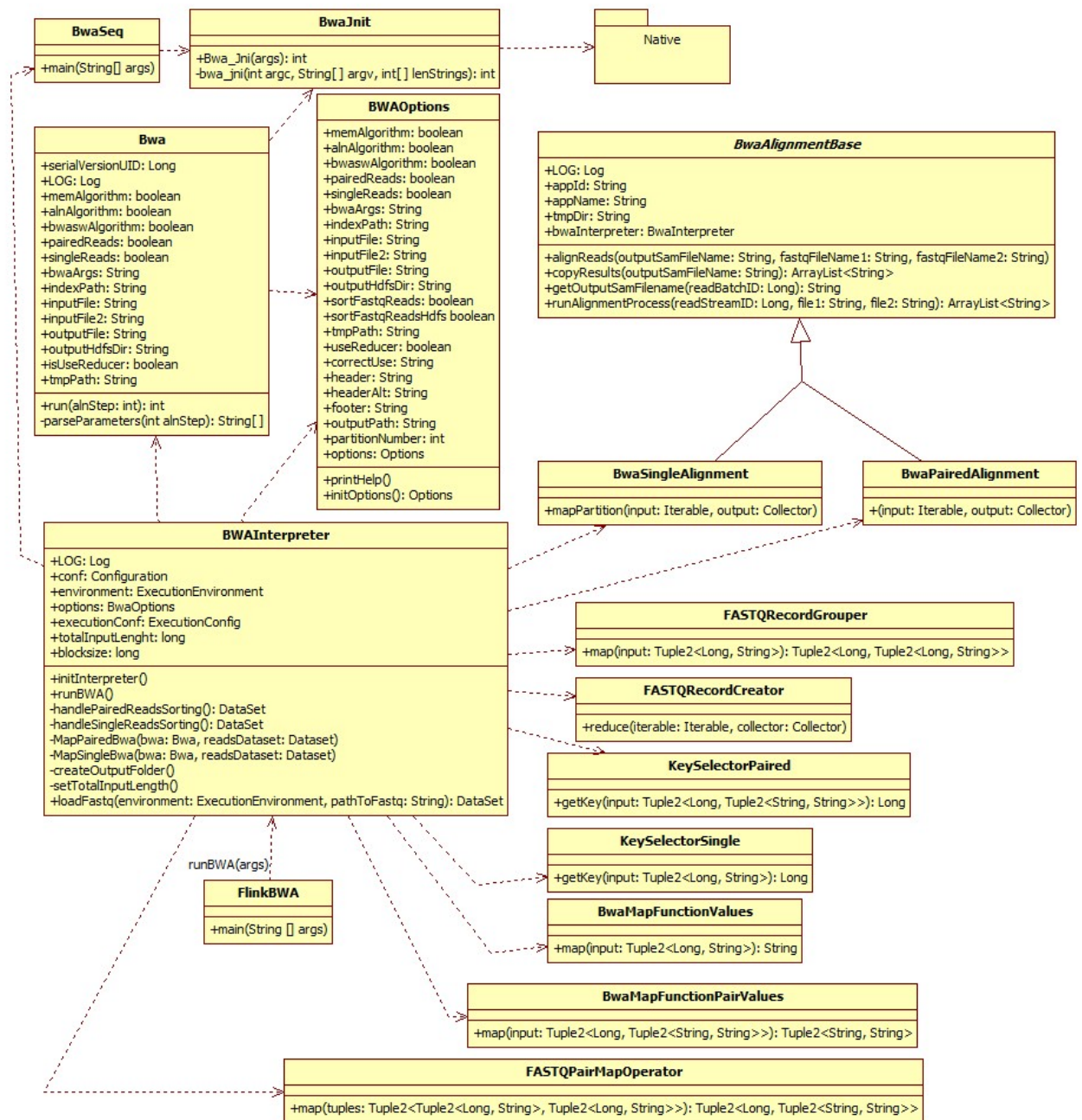


Figura 5.2: Diagrama de clases

5.2.1. Explicación del diseño y patrones utilizados

El paquete “native” que aparece en la Figura 5.2 hace referencia al código de la aplicación original del BWA que, como ya se mencionó anteriormente, no ha sido alterada. Este paquete aparece en el diagrama para indicar que la clase

BwaJnit es la encargada de comunicarse con la aplicación nativa, haciendo uso de la *Java Native Interface* (JNI). Podría decirse que esta interfaz implementa un patrón *Facade* o Fachada, pues aísla al resto de clases de la interacción con el código nativo. Así pues, se consigue separar ambas implementaciones, permitiendo combinar la funcionalidad de ambas pese a que el BWA esté codificado en C/C++ y el código correspondiente a *FlinkBWA* en *Java* (cabe mencionar que, además, *Flink* sólo interpreta los lenguajes Java, Scala y Python, aunque no existe aun ninguna versión estable que soporte este último). Además, este diseño modular también permite sustituir fácilmente la versión utilizada del BWA, por ejemplo, para actualizarla por una más reciente en el futuro (siempre y cuando sus argumentos de entrada no se vean alterados).

Por otra parte, los mecanismos llevados a cabo para el alineamiento de los datos de entrada implementan un patrón *Template Method* o Método Plantilla conformado por las clases: *BwaAlignmentBase*, *BwaSingleAlignment* y *BwaPairedAlignment*. Este patrón permite que las subclases modifiquen parcialmente el comportamiento, evitando la duplicidad de código. La clase abstracta *BwaAlignmentBase* serviría como clase “plantilla”, que define el esqueleto del algoritmo, permitiendo redefinir ciertos pasos a las clases concretas. En este caso concreto, la clase *BwaPairedAlignment* define el manejo de los datos de entrada antes del alineamiento en caso de provenir de dos ficheros y *BwaSingleAlignment* lo define en caso de provenir de un único fichero de entrada. Esta distinción es necesaria ya que las clases tratan un tipo diferente de *Datasets*, por lo que los datos necesitan transformaciones diferentes.

Finalmente, *BWAInterpreter* funciona como un “controlador”, puesto que es la encargada de invocar a las correspondientes funciones según los datos introducidos y las opciones señaladas. Básicamente es la clase que accede a las distintas funcionalidades que implementan el resto de clases, como ya se puede interpretar a partir de las dependencias que muestra el diagrama.

5.2.2. Descripción de las clases

- ***FlinkBWA***: clase principal de *FlinkBWA*. Se encarga de inicializar la ejecución del programa.
- ***BwaInterpreter***: funciona a modo de controlador, definiendo el flujo normal del programa y accediendo a las distintas funcionalidades implementadas por el resto de clases.
- ***Bwa***: clase encargada de comunicarse con el BWA original.
- ***BwaSeq***: clase principal utilizada para ejecutar el programa en caso de querer utilizar el BWA de forma secuencial desde Java.

- ***BwaInit***: se encarga de llamar a las distintas funciones del BWA nativo por medio de la interfaz JNI.
- ***BwaOptions***: clase que analiza los argumentos de entrada introducidos por el usuario y determina las opciones del programa.
- ***BwaAlignmentBase***: clase abstracta que representa el proceso de alineamiento de datos.
- ***BwaSingleAlignment***: clase encargada de iniciar el alineamiento de un conjunto de datos de entrada que provienen de un único fichero de entrada.
- ***BwaPairedAlignment***: clase encargada de iniciar el alineamiento de un conjunto de datos de entrada que provienen de dos ficheros de entrada.
- ***FASTQRecordGrouper***: clase que implementa la funcionalidad de indexar las líneas de entrada FASTQ en grupos de 4 (como ya se comentó en la sección 4, cada lectura viene dada por 4 líneas de datos).
- ***FASTQRecordCreator***: clase que implementa la funcionalidad de agrupar FASTQ sin doble indexado.
- ***FASTQPairMapOperator***: clase utilizada para combinar en un único *dataset* dos *datasets* con la misma clave.
- ***KeySelectorSingle***: clase utilizada para obtener la clave de un *dataset* que contiene datos extraídos de un único archivo FASTQ.
- ***KeySelectorPaired***: clase utilizada para obtener la clave de un *dataset* que contiene datos extraídos de dos archivos FASTQ.
- ***BwaMapFunctionValues***: clase que devuelve el campo de una tupla correspondiente a los datos de un *dataset* que contiene la información proveniente de un único fichero FASTQ.
- ***BwaMapFunctionPairValues***: clase que devuelve el campo de una tupla correspondiente a los datos de un *dataset* que contiene la información proveniente de dos ficheros FASTQ.

5.3. Diagramas de secuencia

Los diagramas de secuencia son utilizados en UML para describir la interacción entre los distintos objetos de un sistema a través del tiempo. En este caso, se proponen dos diagramas de secuencia para mostrar cómo sería la secuencia de acción de los componentes del programa en caso de una ejecución con un sólo fichero de entrada (*Single Reads*) y con dos (*Paired Reads*).

La Figura 5.3 muestra la ejecución en modo *Single Reads* del programa. En primer lugar, el usuario iniciaría el programa por medio del método *main()* de la clase principal, *FlinkBWA*, la cual a su vez ejecutaría el método *runBWA()* de la clase *BwaInterpreter*. Es aquí donde empieza la verdadera ejecución, con la inicialización del intérprete. En este punto se cargarían las opciones del programa y se iniciaría el manejo de los datos con la función *handleSingleReadsSorting()*. Es en esta función cuando se llama a la clase *BwaSingleAlignment* para mapear los datos de entrada, dándoles el formato adecuado para poder ser alineados por el método *runAlignmentProcess()* de la clase *BwaAlignmentBase*.

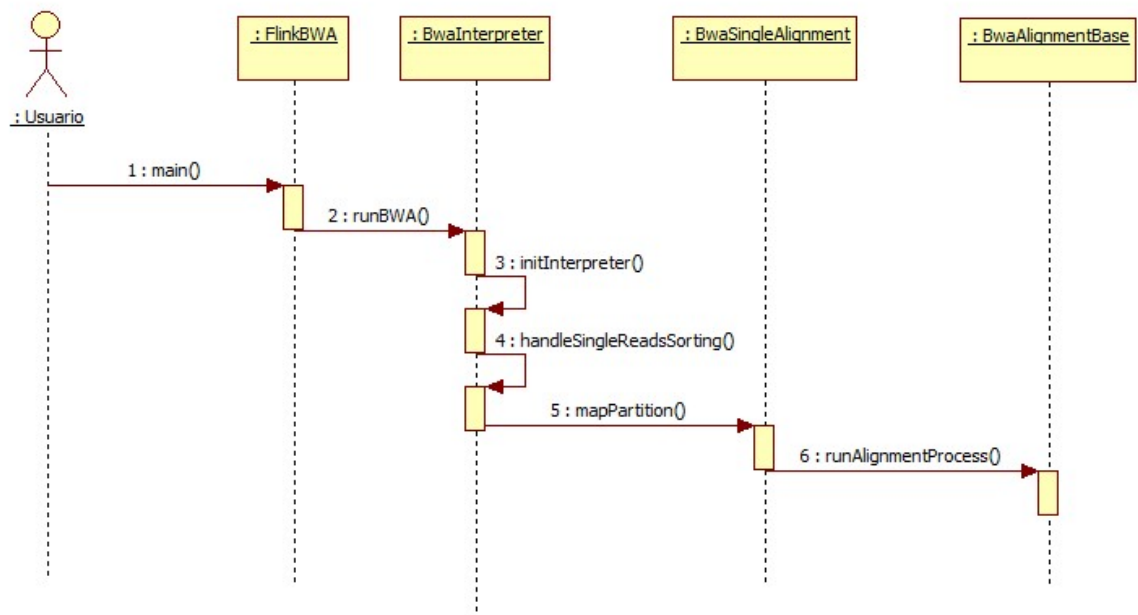
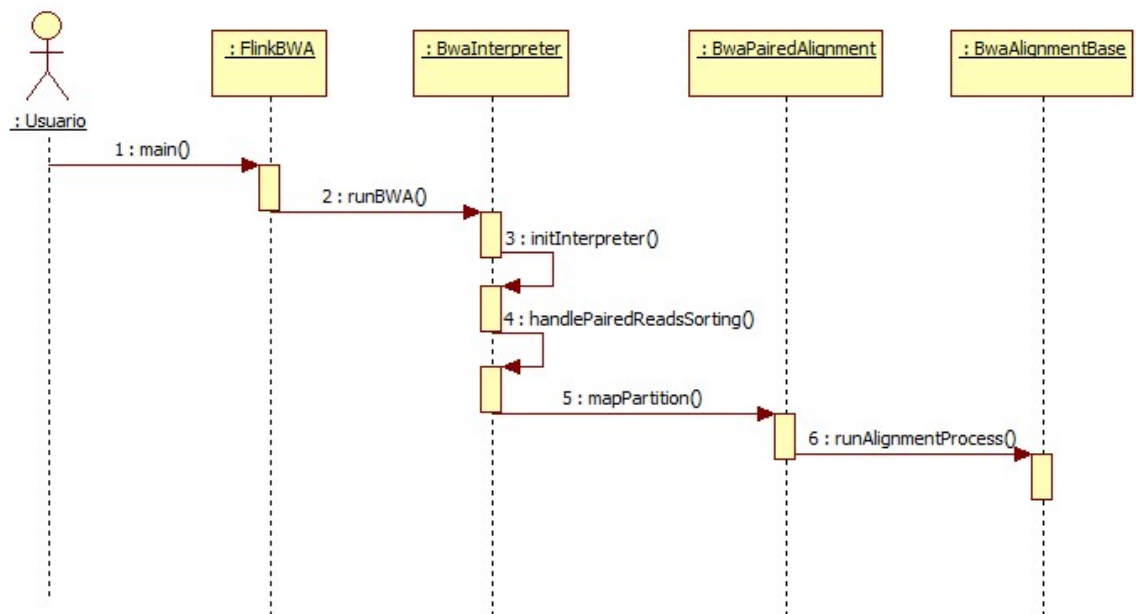


Figura 5.3: Diagrama de secuencia: *Single Reads*

La Figura 5.4 ilustra el mismo proceso pero utilizando dos ficheros como entrada. El procedimiento es el mismo sólo que empleando el método de mapeo proporcionado por la clase *BwaPairedAlignment* antes del alineamiento, que es el apropiado en caso de utilizar dos ficheros FASTQ.

Figura 5.4: Diagrama de secuencia: *Paired Reads*

Capítulo 6

Implementación

Este capítulo tiene por objetivo explicar aspectos vinculados a la implementación de la aplicación, tales como: los requisitos de entorno y *software* identificados para implementar *FlinkBWA*, el funcionamiento del programa o las opciones disponibles en su CLI (*Command Line Interface*).

6.1. Requisitos de *Apache Flink*

- Entorno Unix (Linux, Mac OS X o Cygwin).
- Maven (como mínimo versión 3.1.1, pero recomendable la 3.2.5).
- Java 8.x

6.2. Medios utilizados

6.2.1. Herramientas de desarrollo

- **Ubuntu 16.04 LTS y Windows 10:** en un inicio la implementación de la aplicación se comenzó en un entorno Linux pero, debido a problemas con la conexión por VPN con el CiTIUS, posteriormente el entorno de desarrollo se trasladó a Windows 10. En este último caso fue necesario activar las recientes opciones de desarrollo incorporadas por *Microsoft* para instalar la terminal propia de Ubuntu, facilitando así las operaciones que requerían del uso de una terminal.
- **IntelliJ IDEA Ultimate:** se ha escogido este IDE para el desarrollo debido a que, además de estar recomendado expresamente por el equipo de *Apache Flink*, incorpora soporte para Maven y proporciona licencias gratuitas para fines educativos.

- **Plugin Git para IntelliJ:** *plugin* que da soporte al servicio Git, empleado en este proyecto para llevar a cabo el control de versiones del código fuente.
- **Java JDK 8**

6.2.2. Entorno de desarrollo y pruebas

- **Clúster *Big Data*:** la presente aplicación necesita ser ejecutada en un *cluster* para poder paralelizar los trabajos correspondientes al alineamiento de secuencias entre los diferentes nodos disponibles. Por tanto, se utilizó el clúster *Big Data 1* del CiTIUS para tal fin, el cual cuenta con 16 Servidores Dell EMC PowerEdge R730 con la siguiente configuración:
 - 2 x Intel Xeon E5-2630 v4 (2,2Ghz 10c)
 - 384 GB de RAM: 12 x 32GB RDIMM 2400MT/s
 - 32 TB HDD: 8 x 4TB 7.2k SATA 6Gbps en JBOD
 - 2 x 10Gb BaseT y 2 x 1Gb BaseT
- **Entorno *Hadoop*:** la aplicación asume que es ejecutada en un entorno *Hadoop*, donde los ficheros de entrada se encuentran en el sistema de ficheros distribuido de *Hadoop*: HDFS. Como *Hadoop* ya estaba instalado y configurado en el *cluster* no fue necesaria ninguna acción adicional, concretamente la versión utilizada fue la 2.7.3.
- **Framework Apache Flink:** se trata de la plataforma *Big Data* seleccionada para paralelizar el BWA. La versión instalada en el *cluster* fue la 1.7.2.

6.3. Funcionamiento del programa

El flujo del programa *FlinkBWA* es el siguiente:

1. **Acceso a los datos:** en primer lugar, el programa debe acceder al fichero o ficheros .fastq de entrada. Se asume que dichos ficheros se encuentran en un sistema de ficheros distribuido HDFS, que permita a la aplicación repartir los datos entre los distintos nodos de computación para poder procesarlos en paralelo durante la fase de mapeo.
2. **Carga de datos:** la información contenida en los ficheros .fastq debe ser introducida en una estructura *Dataset* para poder ser manipulados. Los identificadores de los ficheros FASTQ (mencionados en la fase de análisis, capítulo 4, concretamente en la Figura 4.1) son utilizados como clave de los Dataset.

3. **Transformación:** durante esta fase se realizarían distintas operaciones sobre los datos: mapeo, reducción, y operaciones de unión en caso de utilizar dos ficheros de entrada.
4. **Alineamiento:** internamente el programa se encarga de proporcionar al BWA nativo los argumentos correspondientes y los datos de entrada necesarios para realizar el alineamiento.
5. **Escritura de datos:** los distintos datos obtenidos son escritos en uno o varios ficheros .sam de salida en el entorno HDFS.

Como se acaba de mencionar, en caso de tomar como entrada dos archivos FASTQ, sería necesario generar un *Dataset* por fichero, los cuales serían distribuidos a través de los nodos de trabajo. En este caso, no existirían garantías de que ambas estructuras de datos fuesen procesadas por un mismo mapeador. Por tanto, para solventar este problema se presentan dos soluciones:

- **Utilización de las operaciones Join y *sortByKey*:** es la solución implementada por *FlinkBWA*. Consiste en utilizar la operación *join* de Flink para combinar el contenido de los dos ficheros de entrada en un único *Dataset*. En primer lugar se genera un *Dataset* por fichero y, posteriormente, se realiza la operación de unión siempre y cuando los *Dataset* compartan clave. El código utilizado sería el siguiente:

```
// Flink join operation to combine the datasets
DataSet<Tuple2<Long, Tuple2<String, String
    ↪ >>> pairedReadsDataSet = datasetTmp1.
    ↪ join(datasetTmp2).
        where(new KeySelectorSingle()).
            ↪ equalTo(new KeySelectorSingle
            ↪ ()).map(new
            ↪ FASTQPairMapOperator());
```

Listing 6.1: Código correspondiente a una operación de unión con *Flink*

El problema de esta solución es que no se conserva el orden inicial de los ficheros, lo cual se solventa aplicando *a posteriori* una operación *sortByKey*, volviendo a ordenar el *Dataset* según esta clave. La desventaja es que esta última operación es muy costosa en cuanto a consumo de memoria. A continuación, la Figura 6.1 incluye un esquema que ilustraría la operación de unión descrita trabajando en un entorno *Hadoop*, como en este caso:

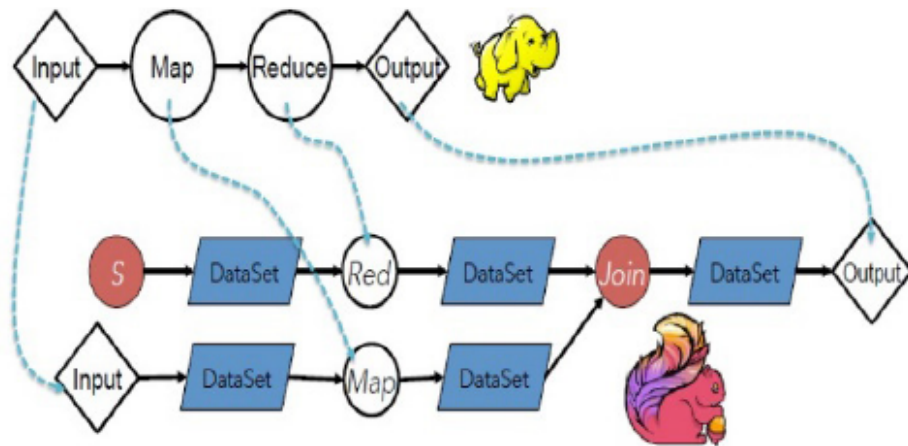


Figura 6.1: Operación *join* de *Flink* en un entorno HDFS

- **SortHDFS:** la alternativa a la solución anterior sería reordenar los archivos FASTQ en un mismo fichero HDFS. Esta solución puede considerarse como una operación de preprocesado ya que se prepararían los ficheros de entrada antes de la ejecución del programa. Esta solución es costosa en cuanto a tiempo, pero ahorra gran cantidad de memoria en comparación con la primera.

6.4. CLI FlinkBWA

Como ya se ha mencionado, la interacción entre el usuario y la aplicación se realiza por medio de la línea de comandos (*Command Line Interface*). Para facilitar su interacción existe una opción de ayuda a la que el usuario puede acudir, obteniendo todas las indicaciones disponibles para interactuar con el programa. A continuación, en la Tabla 6.1 se recogen todas las opciones de *FlinkBWA*:

Opción	Descripción
-h, -help	Muestra la ayuda del programa.
-p, -paired	Opción en caso de querer usar dos ficheros de entrada.
-s, -single	Opción en caso de querer usar un fichero de entrada.
-a, -aln	Utilización del algoritmo ALN.
-b, -bwasw	Utilización del algoritmo BWASW.
-m, -mem	Utilización del algoritmo MEM.
-f, -hdfs	Ordenamiento de los ficheros de entrada usando HDFS.
-k, -flink	Ordenamiento de los ficheros de entrada usando operaciones proporcionadas por <i>Flink</i> .
-t, -tmp	Opción para seleccionar un directorio para el almacenamiento de datos temporales.
-i, -index	Opción para seleccionar el índice creado previamente con el BWA.
-w, -bwa	Opción para pasar argumentos directamente al BWA.
-r, -reducer	Opción para unificar todos los resultados parciales en un único fichero de salida .sam.

Cuadro 6.1: Opciones de *FlinkBWA*

Capítulo 7

Pruebas

Este capítulo está dedicado a la realización de pruebas de *software* llevadas a cabo para comprobar que la aplicación cumple con todos los requisitos determinados para la misma y para garantizar su calidad.

7.1. Verificación

El proceso de verificación se basa en comprobar que el *software* cumple la especificación propuesta para el mismo. Es decir, se comprueba que el sistema cumple tanto los requisitos funcionales como no funcionales identificados para el mismo. Con este fin, se realizaron una serie de pruebas unitarias sobre el *software* que siguen el siguiente formato:

PU-x	Nombre de la prueba
Descripción	Breve descripción de la prueba realizada.
Resultado esperado	Descripción del resultado que debería ser obtenido.
Estado	Cumplido — Pendiente

Cuadro 7.1: Plantilla para Pruebas Unitarias

7.1.1. Pruebas de requisitos funcionales

En primer lugar, cabe señalar que el **RF-001: Paralelización del BWA con Apache Flink** hace alusión a una funcionalidad del comportamiento genérico del sistema. Por tanto, este requisito se dará por cumplido si la aplicación supera la fase de validación.

PU-001	Verificación RF-002: acceso a los ficheros de entrada
Descripción	Dados un conjunto de ficheros de entrada (un fichero .fna y uno o dos ficheros .fastq) se comprueba que el programa es capaz de acceder a su ubicación en el sistema de ficheros HDFS y procesar su contenido.
Resultado esperado	El acceso a los ficheros es correcto y su contenido procesado por el programa.
Estado	Cumplido

Cuadro 7.2: PU-001: verificación RF-002

PU-002	Verificación RF-003: modificación de parámetros
Descripción	Se debe comprobar que el usuario puede cambiar los distintos parámetros que ofrece el programa y éste los reconoce sin problemas, adaptando su comportamiento según el caso.
Resultado esperado	Los parámetros son correctamente interpretados por el programa.
Estado	Cumplido

Cuadro 7.3: PU-002: verificación RF-003

PU-003	Verificación RF-004: generación de un fichero .sam como output
Descripción	Se debe comprobar que el programa es capaz de proporcionar uno o varios ficheros .sam como salida.
Resultado esperado	Obtención de uno o varios ficheros .sam tras la ejecución del programa.
Estado	Cumplido

Cuadro 7.4: PU-003: verificación RF-004

PU-004	Verificación RF-005: interfaz de usuario basada en la línea de comandos
Descripción	Se debe comprobar que el usuario puede interactuar con el programa mediante una CLI relativamente intuitiva y con la ayuda apropiada en caso de que el usuario necesite consultarla.
Resultado esperado	Visualización de una CLI intuitiva con un comando de ayuda disponible para indicar al usuario la forma de interactuar con el <i>software</i> .
Estado	Cumplido

Cuadro 7.5: PU-004: verificación RF-005

7.1.2. Pruebas de requisitos no funcionales

Los requisitos no funcionales **RNF-002: escalabilidad** y **RNF-003: mejora de la eficiencia temporal del BWA** son condiciones vinculadas al rendimiento de la aplicación y, por tanto, su verificación será estudiada en el apartado de pruebas de rendimiento.

PU-005	Verificación RNF-001: extensibilidad
Descripción	Comprobación de que la aplicación efectivamente es escalable gracias al diseño modular empleado.
Resultado esperado	El código nativo del BWA (siempre y cuando no modifique sus archivos ni parámetros de entrada) es fácilmente intercambiable por nuevas versiones del mismo sin afectar al código <i>Flink</i> gracias al diseño modular utilizado.
Estado	Cumplido

Cuadro 7.6: PU-005: verificación RNF-001

PU-006	Verificación RNF-004: utilización de <i>software</i> libre o gratuito con fines educativos
Descripción	Comprobación de que la aplicación se ha realizado en su totalidad sin tener que utilizar <i>software</i> de pago.
Resultado esperado	En los costos del proyecto no figura ningún tipo de costo adicional provocado por la necesidad de pagar alguna plataforma de desarrollo, diseño o <i>cluster</i> de pago.
Estado	Cumplido

Cuadro 7.7: PU-006: verificación RNF-003

PU-007	Verificación RNF-005: utilización de Java como lenguaje de desarrollo
Descripción	Comprobación de que la aplicación se ha codificado en Java.
Resultado esperado	El código correspondiente a FlinkBWA (sin incluir el código nativo) está escrito en Java.
Estado	Cumplido

Cuadro 7.8: PU-007: verificación RNF-004

7.2. Validación

El proceso de validación del *software* tiene como fin asegurar que la aplicación funciona apropiadamente y realiza de manera correcta las tareas para las que fue implementada.

En el caso de *FlinkBWA*, su funcionamiento quedaría validado en el momento en que se compruebe que sus resultados son los mismos que los del alineador BWA inicial. Por tanto, las pruebas realizadas para validar el *software* se han basado en la comparación de los archivos .sam generados a partir de los mismos ficheros de entrada tanto por *FlinkBWA* como por el BWA original.

Los ficheros de entrada utilizados (archivos .fastq y .fna) pertenecen al *1000 Genomes Project*¹. Concretamente, se han realizado las pruebas con los ficheros ERR000589_1 y ERR000589_2, que pueden ser obtenidos de la siguiente forma:

¹<http://www.1000genomes.org/>

```

$ wget ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase3/
  ↳ data/NA12750/sequence_read/ERR000589_1.filt.fastq.
  ↳ gz
$ wget ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase3/
  ↳ data/NA12750/sequence_read/ERR000589_2.filt.fastq.
  ↳ gz

```

Listing 7.1: Obtención de ficheros de entrada de prueba

Una vez ejecutadas ambas aplicaciones con las mismas opciones y con los mismos ficheros de entrada, la comprobación simplemente se basó en la ejecución del comando *diff* para observar las diferencias entre los ficheros de salida (obviando posibles diferencias de formato sin relevancia sobre el contenido). En base a dicha especificación, se empleo la siguiente plantilla para recoger los resultados de las diferentes pruebas realizadas:

PV-x	Plantilla para Pruebas de Validación
Opciones del programa	<ul style="list-style-type: none"> ▪ Número de ficheros de entrada <ul style="list-style-type: none"> • Single reads (-s): un fichero .fastq • Paired reads (-p): dos ficheros .fastq ▪ Algoritmo a utilizar: ALN (-a) — BWASW (-b) — MEM (-m)
Resultados	<ul style="list-style-type: none"> ▪ Esperado: no hay diferencia de contenidos entre los resultados de las aplicaciones. ▪ Incorrecto: existen diferencias sustanciales de contenido entre los resultados de las aplicaciones.
Estado	Cumplido — Pendiente

Cuadro 7.9: Plantilla PV

A continuación, se especifican las diferentes opciones de la aplicación utilizadas y los resultados de las pruebas:

PV-001	Prueba de Validación 001
Opciones del programa	Single Reads (-s) y algoritmo MEM (-m).
Resultados	Esperado.
Estado	Cumplido.

Cuadro 7.10: Especificación PV-001

PV-002	Prueba de Validación 002
Opciones del programa	Single Reads (-s) y algoritmo ALN (-a).
Resultados	Esperado.
Estado	Cumplido.

Cuadro 7.11: Especificación PV-002

PV-003	Prueba de Validación 003
Opciones del programa	Single Reads (-s) y algoritmo BWASW (-b).
Resultados	Esperado.
Estado	Cumplido.

Cuadro 7.12: Especificación PV-003

PV-004	Prueba de Validación 004
Opciones del programa	Paired Reads (-p) y algoritmo MEM (-m).
Resultados	Esperado.
Estado	Cumplido.

Cuadro 7.13: Especificación PV-004

PV-005	Prueba de Validación 005
Opciones del programa	Paired Reads (-p) y algoritmo ALN (-a).
Resultados	Esperado.
Estado	Cumplido.

Cuadro 7.14: Especificación PV-005

PV-006	Prueba de Validación 006
Opciones del programa	Paired Reads (-p) y algoritmo BWASW (-b).
Resultados	Esperado.
Estado	Cumplido.

Cuadro 7.15: Especificación PV-006

Cabe destacar que estas pruebas podrían haber sido mucho más exhaustivas, repitiéndolas sucesivas veces variando los ficheros de entrada seleccionados y comprobando como, de igual manera, no existen diferencias en cuanto al contenido de los resultados. Debido a la limitación temporal de este trabajo sólo se han realizado pruebas con los dos ficheros mencionados, pero su realización enriquecería y haría más sólida la validación del funcionamiento de la aplicación.

7.3. Pruebas de rendimiento

Las pruebas de rendimiento cobran gran relevancia en el presente proyecto, ya que su objetivo principal podría resumirse en la paralelización del BWA con el fin de mejorar su eficiencia temporal.

Por tanto, las pruebas de rendimiento realizadas sobre *FlinkBWA* pretenden mostrar si se han logrado:

- Reducir el tiempo de cómputo conforme se aumenta el nivel de paralelismo (**RNF-002. Escalabilidad**).
- Mejorar el tiempo de cómputo de una ejecución secuencial del BWA original. (**RNF-003. Mejora de la eficiencia temporal del BWA**).

7.3.1. Especificación de las pruebas

Antes de mostrar los resultados de las pruebas realizadas, es necesario concretar qué opciones de *Flink*, parámetros del programa y ficheros de entrada han sido utilizados en las ejecuciones, justificando su elección.

Parámetros de *Flink*

- **Número de *Task Managers***: el número de *Task Managers* utilizados en cada ejecución coincidirá con el nivel de paralelismo indicado para la ejecución (opción -n del programa). Además, cada *Task Manager* contará con un único *Task Slot*.

- **Memoria por cada *Task Manager*:** tras realizar distintas pruebas, se comprobó que la memoria mínima necesaria por *Task Manager* con los ficheros de entrada empleados es de **30 GB**.

Parámetros del programa

- **Algoritmo:** en las últimas reuniones se determinó que el algoritmo **MEM** era el más relevante y, por tanto, se decidió realizar las pruebas de rendimiento con el mismo.
- **Número de ficheros de entrada:** teniendo en cuenta que la memoria necesaria por *Task Manager* ascendía a 50 GB con la opción *Paired Reads*, la limitación de 3,74 TB de memoria del *cluster* y el hecho de que sean recursos compartidos, se decidió utilizar la opción ***Single Reads*** para las pruebas (es decir, un sólo fichero como entrada).
- **Particiones:** se realizarán pruebas con 4, 8, 16, 32 y 64 particiones del fichero de entrada.
- **Reducción:** la opción “reducer” no se utilizará en las pruebas de rendimiento, ya que unificar todos los ficheros de salida en uno sólo no aporta ningún beneficio al estudio y, además, dichas operaciones a mayores aumentarían el tiempo de cómputo del programa.
- **Ordenamiento:** la opción de ordenamiento tampoco se utilizará por el mismo motivo que la opción anterior.

Ficheros de entrada

Como ya se mencionó, se utilizará la opción *Single Reads* en las pruebas, por lo que sólo será necesario un fichero de entrada y la referencia del Genoma Humano. Concretamente, los ficheros empleados son:

Nombre del fichero	Tamaño (GB)
ERR000589_1.filt.fastq	1,68
GCA_000001405.28_GRCh38.p13_genomic.fna	3,08

Cuadro 7.16: Ficheros utilizados en las pruebas de rendimiento

Cabe recordar que para que el programa pueda interpretar la referencia del Genoma Humano es necesario indexar dicho fichero antes de ejecutar el programa. En el manual de usuario (apéndice B) se explica cómo realizar esta operación en detalle.

7.3.2. Resultados

A continuación, se detallarán los resultados obtenidos tanto con los distintos niveles de paralelismo utilizados en el caso de *FlinkBWA* como el tiempo de una ejecución secuencial del BWA.

BWA secuencial

Para corroborar que el programa mejora el tiempo de una ejecución secuencial del BWA original, fue necesario comprobar cuál era dicho tiempo de cómputo.

Para ello, se ejecutó el BWA tomando el mismo fichero de entrada, el mismo índice del Genoma Humano y utilizando el algoritmo MEM. Finalmente, la ejecución tardó en completarse un total de: **66.43 minutos**.

FlinkBWA

Como ya se ha comentado en la especificación, se han probado 5 niveles de paralelismo diferentes. El criterio de selección ha sido tomar un mínimo de 4 particiones, haciendo que el siguiente nivel fuese el doble que su antecesor hasta llegar a un máximo de 64.

Además, se realizaron un total de 3 ejecuciones por caso para tratar de establecer una cierta fiabilidad en los datos tomados. La media de los mismos fue utilizada para generar un gráfico que mostrase de forma más visual las diferencias en los tiempos según el nivel de paralelismo. Por su parte, la desviación típica calculada permitió conocer si dichos datos divergían mucho entre sí. Teniendo en cuenta que los valores obtenidos se encuentran entre 0,036 y 0,555 se determinó que los datos eran suficientemente válidos.

En la Tabla 7.17 pueden verse recogidos los datos tomados tras las distintas ejecuciones del programa, variando el nivel de paralelismo:

Pruebas			
Nivel paralelismo	Tiempo (minutos)	Media (minutos)	Desviación Típica
4	16,54	17,2133	0,55530
	17,2		
	17,9		
8	10,31	10,2700	0,08641
	10,35		
	10,15		
16	7,23	7,2500	0,03559
	7,22		
	7,3		
32	5,47	5,4333	0,04497
	5,37		
	5,46		
64	5,06	4,7233	0,23977
	4,52		
	4,59		

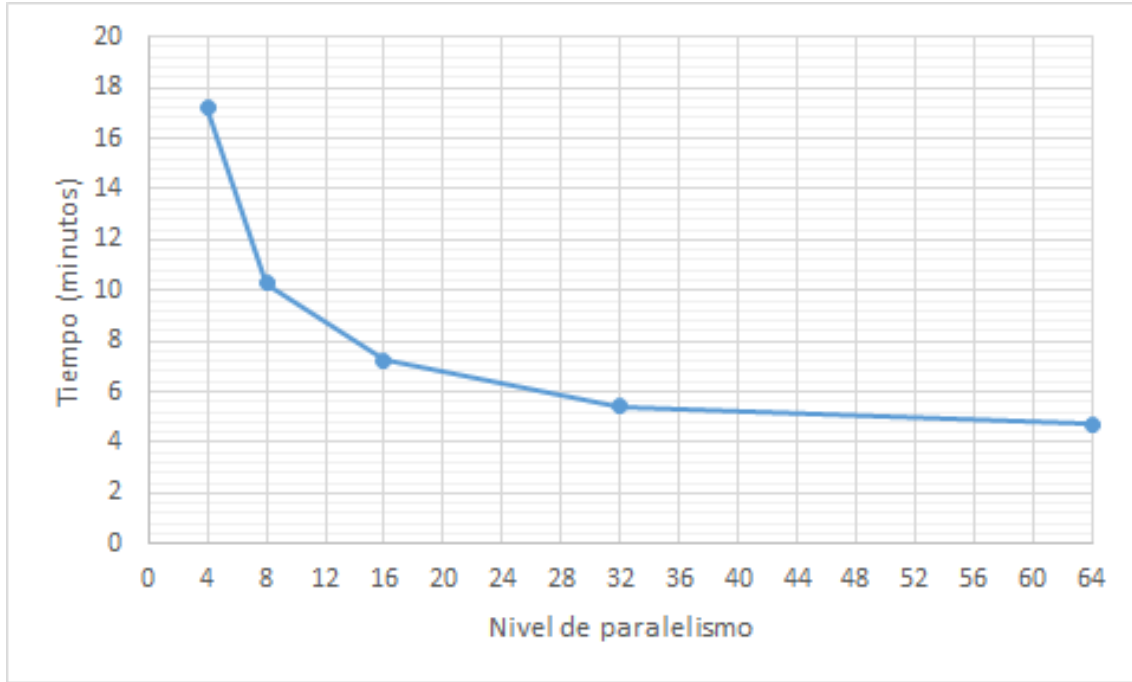
Cuadro 7.17: Resultados de las pruebas de rendimiento

7.3.3. Análisis de los resultados

A partir de los tiempos medios obtenidos anteriormente, se ha generado el gráfico de la Figura 7.1, donde puede observarse como conforme aumenta el nivel de paralelismo, el tiempo de cómputo disminuye.

Observando los resultados, la mejora más notoria se produce al pasar de 4 a 8 particiones, donde el tiempo se reduce un 40,34 %. En el resto de casos, también se produce una mejora conforme aumenta el paralelismo: de 8 a 16, el tiempo se reduce casi un 30 %; de 16 a 32, un 25 %; y de 32 a 64 un 13 %.

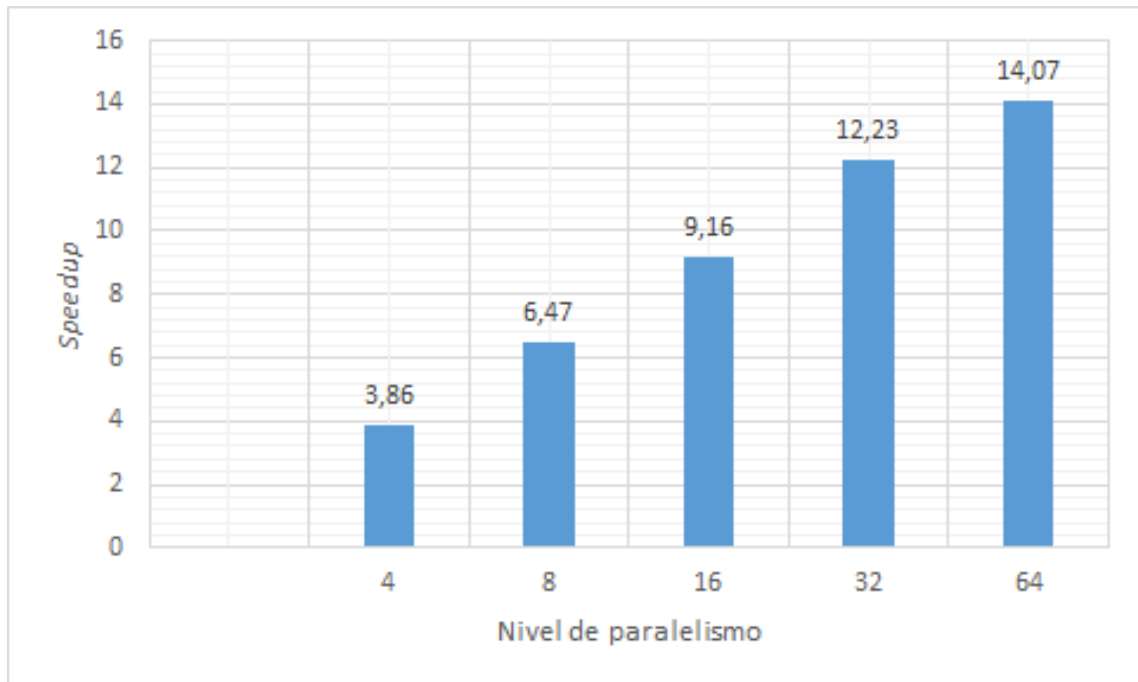
En vista de estos resultados, podrían darse por cumplidos los dos objetivos que se comentaban al principio de la sección y, por tanto, el **RNF-002** y el **RNF-003**. En primer lugar, se da por hecho que la paralelización de la aplicación ha logrado mejorar los 66,43 minutos obtenidos por la ejecución secuencial del BWA. Por otra parte, se ha comprobado como, aumentando el nivel de paralelismo, el tiempo de cómputo del programa se reduce. Ciertamente es que utilizando 32 y 64 particiones la diferencia ya no es muy relevante, lo que probablemente se deba al retardo causado por otras operaciones no paralelizadas del programa cuyo tiempo siempre es el mismo (aceptación del trabajo en el *cluster*, lectura de ficheros del sistema HDFS, etc.).

Figura 7.1: Eficiencia temporal *FlinkBWA*

Los resultados obtenidos tienen bastante coherencia si se tiene en cuenta que, lo que hace la aplicación, es dividir entre los distintos nodos de trabajo seleccionados el procesamiento del conjunto de datos de entrada. De esta forma, aprovechando la infraestructura utilizada y la paralelización del trabajo, se consigue una reducción significativa en el tiempo de cómputo del alineador conforme aumenta el paralelismo.

Para mostrar de forma más representativa la mejora en cuanto a tiempo de cómputo obtenida al paralelizar el código secuencial se ha realizado un gráfico con el *speedup*, el cual puede verse reflejado en la Figura 7.2. Se denomina *speedup* o aceleración a la cantidad de mejora obtenida en el rendimiento de un sistema después de realizar una serie de modificaciones sobre el mismo. Es decir, indica cuántas veces más rápido es el sistema tras las mejoras incluidas. Su cálculo se realiza de la siguiente manera:

$$Speedup = \frac{T_{secuencial}}{T_{paralelo}}$$

Figura 7.2: *Speedup*

En definitiva, el gráfico muestra lo ya mencionado, la mejora aumenta conforme lo hace el nivel de paralelismo. Además, refleja más nítidamente el hecho de que al pasar de 32 *cores* a 64 la mejoría es mucho menor que de 4 a 8. Este fenómeno ya fue comentado anteriormente, aunque no se profundizó en su explicación. En el contexto de un caso ideal, utilizando 4 núcleos el *speedup* debería de ser 4, utilizando 8 sería 8 y así sucesivamente. Este hecho nunca ocurre debido a la ley de *Amdahl* [26]. Lo que sostiene esta ley es que, independientemente del número de *cores* utilizados, el *speedup* tiene un límite superior definido por la parte secuencial del programa, es decir, la parte que no puede paralelizarse. Por este motivo, llega un punto en el que por mucho que se aumente el nivel de paralelismo, éste ya no puede incorporar mayor mejoría sobre el tiempo de cómputo.

Capítulo 8

Conclusiones

Debido a la creciente cantidad de datos que manejan hoy en día prácticamente todas las aplicaciones y programas informáticos, las herramientas y técnicas *Big Data* cobran una especial relevancia a la hora de intentar optimizar su funcionamiento. Las aplicaciones biotecnológicas, como los alineadores de secuencias genéticas, no son una excepción. Este tipo de aplicaciones tienen como reto el alineamiento de una cantidad considerable de datos, proceso especialmente costoso en cuanto a tiempo. El presente trabajo tuvo como objetivo la paralelización del alineador de secuencias genéticas BWA (*Burrows-Wheeler Aligner*), uno de los más ampliamente utilizados, empleando la plataforma *Big Data Apache Flink*.

En cuanto a la metodología de trabajo empleada, teniendo en cuenta el poco conocimiento inicial en aplicaciones de alineamiento genético y el desconocimiento de la herramienta *Big Data* a utilizar, se decidió seguir una metodología basada en *Scrum*. Esta metodología presenta flexibilidad para gestionar cambios frecuentes y situaciones con gran incertidumbre, por lo que fue la idónea para abarcar dicho proyecto dadas las circunstancias.

Se realizó un diseño modular que permitiese separar la implementación del BWA original, codificada prácticamente en su totalidad en C, de la nueva implementación con *Flink* y *Java*. De esta forma, siempre y cuando los argumentos de entrada del programa no se vean afectados, es sencillo modificar la versión nativa del BWA utilizada por el programa para, por ejemplo, actualizarla a una versión más reciente. Además, este diseño también proporciona mayor sencillez a la implementación, facilitando el trabajo en caso de querer añadir o modificar funcionalidades.

Una vez implementado el *software* y tras las pertinentes fases de verificación y validación, se pudo comprobar que el programa cumplía con todas las especificaciones determinadas para el mismo al inicio del proyecto. Además, los resultados

que proporcionaba tras el alineamiento de los datos de entrada eran idénticos a los generados por el BWA original, validando el correcto funcionamiento del *software*.

Finalmente, tras la realización de las pruebas de rendimiento, se observó como el tiempo de cómputo del programa se reducía conforme aumentaba el nivel de paralelismo utilizado. Por tanto, cuanto más se particionaban los datos de entrada, dividiendo su alineamiento entre los distintos nodos de trabajo del entorno de ejecución, mayor era la ganancia temporal obtenida.

En vista de los resultados obtenidos, se muestran una vez más las ventajas que puede proporcionar el uso de herramientas *Big Data* a la hora de tratar grandes volúmenes de datos. En este caso, *FlinkBWA* también sirve como aporte al estudio de las posibilidades que puede ofrecer *Apache Flink*, herramienta que, por su todavía reciente aparición en el mercado, no cuenta con la misma madurez ni con la misma comunidad de usuarios que otras herramientas similares.

Como conclusión, el proyecto se da por finalizado cumpliendo todos los requisitos fijados en un principio y obteniendo unos resultados satisfactorios dentro del margen de tiempo establecido para la realización del trabajo.

Capítulo 9

Trabajo futuro

Pese a que el proyecto ha conseguido cumplir todos los requisitos identificados en un principio y completar los objetivos propuestos, existen una serie de mejoras que, de existir más tiempo, sería interesante explorar.

En primer lugar, durante las pruebas de rendimiento se determinó que el programa usa una **cantidad muy elevada de memoria RAM**. En principio, es normal que una aplicación de estas características consuma gran cantidad de memoria, tanto por las operaciones que realiza como por el número de datos que maneja. Además, el programa está codificado en Java, lenguaje que trata la memoria de forma dinámica sin dar posibilidad al desarrollador de liberarla ni, por tanto, de minimizar su uso. Una de las posibles mejoras, por tanto, sería revisar más en detalle las opciones de *Flink* para estudiar si podría reducirse el consumo de memoria de alguna forma.

La segunda mejora propuesta tiene que ver con el **orden del fichero de salida** en caso de usar la opción “Reducer” y particiones. Pese a que en un inicio al leer los ficheros de entrada y cargarlos en un *Dataset* los datos son ordenados de forma ascendente en función de una clave, en caso de unificar las distintas salidas de los nodos en un único fichero, éste nunca es ordenado de nuevo. La razón es que si el alineamiento se realizase sobre varios *Dataset* en lugar de ficheros .sam parciales, el proceso sería mucho más lento, aunque posteriormente fuese más fácil volver a ordenarlos por clave y unirlos. Además, existe una herramienta denominada SAMtools¹ que permite modificar ficheros .sam, alterando el orden de sus líneas, entre otras acciones. Para hacer *FlinkBWA* una herramienta más completa en sí misma, se propone añadir una opción que realice el ordenamiento sin tener que emplear aplicaciones externas.

¹<http://samtools.sourceforge.net/>

Para concluir, mencionar que en este trabajo sólo se explotan las capacidades de *Flink* en relación con el procesado de datos por lotes, pero, como ya se ha mencionado, *Flink* también permite el procesado de flujos de datos. Actualmente, el BWA sólo admite ficheros de datos estáticos como entrada pero, si en un futuro apareciese la necesidad de procesar datos en tiempo real, una nueva **versión *streaming*** de este *software* que trabajase con datos de tipo *DataStream* en lugar de *DataSet* podría ser una solución interesante.

Apéndice A

Manual técnico

Este manual sirve de guía para todo usuario que desee trabajar con la aplicación *FlinkBWA* descrita en este documento, especificando el entorno y la configuración necesaria para tal fin.

A.1. Entorno

FlinkBWA es una aplicación *Big Data* ideada para ser ejecutada en un entorno con varios nodos de trabajo, como un *cluster*, donde el programa pueda aprovechar dichos nodos para mejorar su rendimiento. Por tanto, el entorno de trabajo debe cumplir las siguientes especificaciones:

- **Cluster Hadoop:** como ya se ha mencionado, el programa debe ejecutarse en un *cluster*. Además, la aplicación sobreentiende que el sistema de ficheros es HDFS, por lo que debe tener *hadoop* correctamente configurado.
- **Espacio en el directorio temporal:** *FlinkBWA* utiliza o bien el espacio local del directorio `/tmp`, o bien el espacio de los directorios temporales que tengan configurados *Hadoop* o *Flink*.
- **Al menos 10GB de memoria por contenedor YARN:** este espacio es necesario debido al tamaño del fichero de datos del Genoma Humano. En caso de sólo querer hacer pruebas con versiones reducidas del mismo no sería necesario tanto espacio.
- **Tener instalado *Flink*:** concretamente, la versión instalada en el *cluster* utilizado para ejecutar esta aplicación fue la 1.7.2, la cual puede obtenerse mediante el siguiente *link*: <https://flink.apache.org/downloads.html#apache-flink-172>

A.2. Requisitos *software*

Los requisitos *software* necesarios son los siguientes:

- **Java JDK 8:** la aplicación no deja de ser un programa Java y, por tanto, es necesaria su máquina virtual.
- **Maven:** necesario en caso de querer compilar el programa, ya que se trata de un proyecto Maven.
- **JAVA_HOME:** es necesario tener correctamente definida la variable de entorno JAVA_HOME para que funcione el programa. También puede indicarse manualmente en el fichero “/src/main/native/Makefile.common”.

A.3. Datos de entrada

Los datos de entrada que acepta esta aplicación son ficheros .fastq y .fna, formatos utilizados en la mayoría de aplicaciones para el alineamiento de secuencias genéticas. Concretamente, en este desarrollo se han utilizado los pertenecientes al proyecto 1000 Genomes Project (www.1000genomes.org).

Por supuesto, estos datos deben encontrarse en el entorno HDFS del *cluster* para que la aplicación puede trabajar con ellos. Excepto el fichero correspondiente al Genoma Humano de referencia (.fna) que es leído desde el directorio de trabajo local del usuario dentro del *cluster*.

Apéndice B

Manuales de usuario

Este manual informa al usuario sobre el procedimiento que debe llevar a cabo para ejecutar e interactuar de manera apropiada con la aplicación.

B.1. Instalación

En principio, el *software* estará subido a uno de los repositorios del CiTIUS con acceso público, por lo que si se desea descargar la aplicación simplemente será necesario ejecutar:

```
$ git clone https://github.com/citiususc/FlinkBWA.git
```

B.2. Compilación

Una vez se tenga una copia del programa disponible, la forma de compilarlo es la misma que en todos los proyectos Maven:

```
$ cd FlinkBWA # Se accede a la carpeta del proyecto
$ mvn clean package # Y se compila
```

B.3. Ejecución

Si la compilación se ha realizado de manera exitosa, en la carpeta *target* debería estar el ejecutable .JAR necesario para lanzar la aplicación: **FlinkBWA-1.0-SNAPSHOT.jar**.

En este punto, deberían obtenerse un par de ficheros de datos de entrada para lanzar la aplicación:

```

$ wget ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase3/
  ↳ data/NA12750/sequence_read/ERR000589_1.filt.fastq.
  ↳ gz
$ wget ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase3/
  ↳ data/NA12750/sequence_read/ERR000589_2.filt.fastq.
  ↳ gz

# Posteriormente se deben descomprimir:
$ gzip -d ERR000589_1.filt.fastq.gz
$ gzip -d ERR000589_2.filt.fastq.gz

# E incorporar al sistema de ficheros HDFS:
$ hdfs dfs -copyFromLocal ERR000589_1.filt.fastq
  ↳ ERR000589_1.filt.fastq
$ hdfs dfs -copyFromLocal ERR000589_2.filt.fastq
  ↳ ERR000589_2.filt.fastq

```

Listing B.1: Preparación de los ficheros de entrada .fastq

La obtención y preparación del fichero de referencia del Genoma Humano es algo más costosa. En primer lugar debe descargarse el fichero con los datos, descomprimirse y, posteriormente, debe ser indexado utilizando el BWA nativo (puede utilizarse el que ya incorpora FlinkBWA o bien obtener la aplicación de *Github*). Los pasos serían los siguientes:

```

# En primer lugar, se obtiene el archivo de datos:
$ wget ftp.ncbi.nlm.nih.gov/genomes/all/GCA
  ↳ /000/001/405/GCA_000001405.28_GRCh38.p13/
  ↳ GCA_000001405.28_GRCh38.p13_genomic.fna.gz

# Se descomprime:
$ gzip GCA_000001405.28_GRCh38.p13_genomic.fna.gz

# Se obtiene el programa BWA nativo
$ git clone https://github.com/lh3/bwa.git
$ cd bwa; make

# Y se indexa el archivo descomprimido
$ ./bwa index GCA_000001405.28_GRCh38.p13_genomic.fna

```

Listing B.2: Preparación de los datos del Genoma Humano de referencia

En el directorio donde se encontraba el fichero `.fna` con el Genoma Humano deberían aparecer ahora también los siguientes ficheros, los cuales necesitará la aplicación durante su ejecución:

- `GCA_000001405.28_GRCh38.p13_genomic.fna.amb`
- `GCA_000001405.28_GRCh38.p13_genomic.fna.ann`
- `GCA_000001405.28_GRCh38.p13_genomic.fna.bwt`
- `GCA_000001405.28_GRCh38.p13_genomic.fna.pac`
- `GCA_000001405.28_GRCh38.p13_genomic.fna.sa`

En este punto, ya se podría acceder al directorio del *cluster* donde está instalado *Flink* y ejecutar la aplicación con las opciones y parámetros que se deseen. A continuación, se muestran algunos ejemplos de cómo lanzar la aplicación:

```
# Ejemplo 1: lanzar la ayuda del programa.
$ ./bin/flink run -c com.github.flinkbwa.FlinkBWA -m
  ↪ yarn-cluster FlinkBWA-1.0-SNAPSHOT.jar -h

# Ejemplo 2: lanzar el programa con nivel de
  ↪ paralelismo 4, 4 task manager, 4GB de memoria para
  ↪ el task manager, un fichero de entrada y el
  ↪ algoritmo MEM.
./bin/flink run -c com.github.flinkbwa.FlinkBWA -m yarn
  ↪ -cluster -yn 4 -ytm 4096m FlinkBWA-1.0-SNAPSHOT.
  ↪ jar -m -s -n 4 --index /Data/GenomaHumano/
  ↪ GCA_000001405.28_GRCh38.p13_genomic.fna /user/
  ↪ silvia.rodriguez.alcaraz/ERR000589_1_short.filt.
  ↪ fastq Output_ERR000589
```

Listing B.3: Ejemplos de ejecución

Téngase en cuenta que:

- `/Data/GenomaHumano/`: sería el directorio donde se encuentran los ficheros de referencia del genoma humano que se han preparado anteriormente.
- `/user/silvia.rodriguez.alcaraz/`: es el directorio HDFS donde están los `.fastq`.

Finalmente, en el caso del ejemplo 2, dentro del directorio `Output_ERR000589` en el directorio HDFS del usuario se encontrarían el conjunto de ficheros `.sam` resultantes, es decir, la salida del programa.

Bibliografía

- [1] Gantz, J. and Reinsel, D. (2012). Executive Summary: A Universe of Opportunities and Challenges. Artículo de EMC (<https://www.emc.com/leadership/digital-universe/2012iview/executive-summary-a-universe-of.htm>). Consultado el 1 de febrero del 2019.
- [2] Documentación sobre el *Burrows-Wheeler Aligner* (BWA). Página de BWA (<http://bio-bwa.sourceforge.net/>). Consultada el 20 de febrero del 2019.
- [3] Documentación completa sobre *Apache Flink*. Página oficial de *Apache Flink* (<https://flink.apache.org/>). Consultada el 29 de enero del 2019.
- [4] José M. Abuín, Juan C. Pichel, Tomás F. Pena and Jorge Amigo. “SparkBWA: Speeding Up the Alignment of High-Throughput DNA Sequencing Data”. PLoS ONE 11(5), pp. 1-21, 2016.
- [5] Project Management Institute, Guía de los Fundamentos para la Dirección de Proyectos (Guía del PMBOK), 5ª edición.
- [6] Calendario de lectura de TFG 2019-2020. (http://www.usc.es/etse/files/u1/3523_001.pdf). Consultada el 8 de junio del 2019.
- [7] Principios del Manifiesto Ágil (<https://agilemanifesto.org/iso/es/principles.html>). Consultada el 22 de febrero del 2019.
- [8] Guía de *Scrum*. *Scrum Guide* (<https://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-es.pdf>). Consultada el 22 de febrero del 2019.
- [9] Táboas de retribucións de Personal Docente e Investigador. USC. (<http://www.usc.es/gl/servizos/sxp/taboas/tabPDI19.html>). Consultada el 27 de febrero de 2019.
- [10] Bases y tipos de cotización 2019. Seguridad Social. (<http://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537>). Consultada el 27 de febrero de 2019.

- [11] Justificación de Costes Indirectos. USC. (http://www.usc.es/export9/sites/webinstitucional/es/congresos/xiiiencontroredeugi/descargas/COSTES_INDIRECTOS_AEI_Carmen_Penas_Justificacion_Costes_Indirectos.pdf). Consultada el 26 de febrero de 2019.
- [12] Sitio oficial de UML. (<http://www.uml.org/>). Consultada el 27 de febrero de 2019.
- [13] Convention on Biological Diversity. United Nations (1992). (<https://www.cbd.int/doc/legal/cbd-en.pdf>). Consultada el 9 de marzo de 2019.
- [14] Apuntes de Bioquímica I. Universidad Complutense de Madrid (UCM). <http://webs.ucm.es/info/biomol2/bioquimicaI/WTa/Homologia.html>. Consultada el 9 de marzo de 2019.
- [15] Biotecnología para no iniciados, Capítulo III. Genetaq, Centro de Genética Molecular (2015). <http://genetaq.com/es/blog/bioinformatica-para-no-iniciados-capitulo-iii>. Consultada el 9 de marzo de 2019.
- [16] M. Burrows, D.J. Wheeler (May 10, 1994). “A Block-sorting Lossless Data Compression Algorithm”. (<https://www.hpl.hp.com/techreports/Compaq-DEC/SRC-RR-124.pdf>). Consultada el 10 de marzo de 2019.
- [17] Li H. and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler Transform. *Bioinformatics*, 25:1754-60. [PMID: 19451168].
- [18] Li H, Durbin R. Fast and Accurate Short Read Alignment with Burrows-Wheeler Transform. *Bioinformatics*. 2009; 25(14):1754–1760. doi: 10.1093/bioinformatics/btp324 PMID: 19451168.
- [19] Li H, Durbin R. Fast and Accurate Long-Read Alignment with Burrows-Wheeler Transform. *Bioinformatics*. 2010; 26(5):589–595. doi: 10.1093/bioinformatics/btp698 PMID: 20080505.
- [20] Li H (2013). Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv:1303.3997.
- [21] The SAM/BAM Format Specification Working Group (2019). “Sequence Alignment/Map Format Specification”. (<http://samtools.github.io/hts-specs/SAMv1.pdf>). Consultada el 10 de marzo de 2019.
- [22] Rajkumar Buyya, Rodrigo N. Calheiros y Amir Vahid Dastjerdi. *Big Data: Principles and Paradigms*, 3ª edición, Morgan Kaufmann (Elsevier), 2016.

- [23] Niño, M., Illarramendi, A.. (2015). (*UNDERSTANDING BIG DATA: ANTECEDENTS, ORIGIN AND LATER DEVELOPMENT*). *DYNA New Technologies*, 2(1). [8 p.]. DOI: <http://dx.doi.org/10.6036/NT7835>.
- [24] José Carlos Baquero, Pablo González. *El futuro de las tecnologías de Streaming: Apache Flink*. <https://openexpoeurope.com/es/tecnologias-streaming-apache-flink/>. Consultada el 12 de marzo de 2019.
- [25] Jeyhun Karimov, Tilmann Rab, Asterios Katsifodimos, Roman Samarev, Henri Heiskanen, Volker Markll. “*Benchmarking Distributed Stream Processing Engines*”. (<https://arxiv.org/ftp/arxiv/papers/1802/1802.08496.pdf>). Consultada el 14 de marzo de 2019.
- [26] Gene Amdahl, “*Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities*”, *AFIPS Conference Proceedings*, (30), pp. 483-485, 1967.