Escuela Politécnica Superior de Linares

Grado en Ingeniería Telemática

**UNIVERSIDAD DE JAÉN**
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

# INFRAESTRUCTURA CLOUD PARA SU USO EN BIOINFORMÁTICA

**Alumno:** **María Teresa Martos Velasco**

**Tutor:** José Enrique Muñoz Expósito
**Depto.:** Ingeniería de Telecomunicación

**Octubre, 2016**

**UNIVERSIDAD DE JAÉN**

***ESCUELA POLITÉCNICA SUPERIOR DE LINARES***

Trabajo Fin de Grado

Curso 2016 – 2017

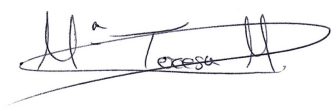
# Infraestructura Cloud para su uso en bioinformática

**Alumno:** Martos Velasco, María Teresa

**Tutor:** Muñoz Expósito, José Enrique

**Departamento:** Ingeniería de Telecomunicación

**Firma de la autora** **Firma del tutor**



# ÍNDICE GENERAL

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# ABREVIATURAS

| | |
|---|---|
| **A** | Adenina |
| **ADN** | Ácido Desoxirribonucleico |
| **ARN** | Ácido Ribonucleico |
| **AWS** | Amazon Web Services |
| **BLAST** | Basic Local Alignment Search Tool |
| **C** | Citosina |
| **CERN** | Organización Europea para la Investigación Nuclear |
| **CRUD** | Create, Receive, Update, Delete |
| **DAO** | Data Access Object |
| **DDL** | Lenguaje de Definición de Datos |
| **DI** | Inyección de Dependencias |
| **DRMAA** | Distributed Resource Management Application API |
| **DRMS** | Sistema Gestor de Recursos Distribuidos |
| **DSA** | Distributed Systems Architecture |
| **DTO** | Data Transport Object |
| **EC2** | Elastic Compute Cloud |
| **GUI** | Interfaz Gráfica de Usuario |
| **IDE** | Integrated Development Environment |
| **IaaS** | Infrastructure as a Service |
| **KVM** | Kernel-based Virtual Machine |
| **LAN** | Red de área local |
| **LSF** | Load Sharing Facility |
| **MAN** | Red de área Metropolitana |
| **MVC** | Modelo-Vista-Controlador |
| **NCBI** | National Center for Biotechnology Information |
| **NFS** | Network File System |
| **OGS/GE** | Open Grid Scheduler/Grid Engine |
| **RHEL** | Red Hat Linux Enterprise |
| **RR** | Round-Robin |
| **SELinux** | Security-Enhanced Linux |
| **SGBD** | Sistema Gestor de Bases de Datos |
| **SGE** | Sun Grid Engine |
| **SISSL** | Sun Industry Standards Source License |
| **SL** | Scientific Linux |
| **SO** | Sistema Operativo |

| | |
|---|---|
| **SSH** | Secure SHell |
| **STS** | Spring Tool Suite |
| **T** | Timina |
| **TFG** | Trabajo de Fin de Grado |
| **TCP** | Transmission Control Protocol |
| **U** | Uracilo |
| **UDP** | User Datagram Protocol |
| **VDC** | Virtual Data Center |
| **VNC** | Virtual Network Computing |
| **WAN** | Red de área extensa |

# 1 RESUMEN

## 1.1 Resumen

Este trabajo de fin de grado (TFG) tiene como objetivo la creación de una infraestructura cloud para la ejecución de dos aplicaciones bioinformáticas: *Basic Local Alignment Search Tool* (BLAST) y Trinity RNA-Seq, ambas de gran utilidad en el ámbito científico.

La creación de dicha infraestructura se llevará a cabo empleando la plataforma IaaS OpenNebula y estará compuesta por una colección de máquinas virtuales en las que se instalará el sistema gestor Open Grid Scheduler/Grid Engine (OGS/GE) con el fin de crear un sistema grid, de manera que dichas máquinas virtuales ejecuten en paralelo y de forma distribuida las tareas asociadas al programa de secuenciación Trinity.

Además, se diseñará e implementará una aplicación web para, a través de la cual, interactuar con las aplicaciones bioinformáticas anteriormente mencionadas.

Finalmente, se realizará una recopilación de resultados relativos a tiempo de ejecución y exactitud de las soluciones obtenidas por las aplicaciones utilizadas.

## 1.2 Abstract

The objective of this Capstone Project is the design and implementation of a cloud-based infrastructure to support the conduction of experiments through two of the most widely used applications in the field of Bioinformatics: *Basic Local Alignment Search Tool* (BLAST), and Trinity RNA-Seq.

The development of said infrastructure will be based on OpenNebula. The resulting IaaS will be composed of a series of virtual machines in the form of a grid system that will allow distributed, parallel execution of tasks associated to the Trinity sequencing program. Each virtual machine will have an Open Grid Scheduler / Grid Engine (OGS/GE) installed to support the establishment of the grid.

Furthermore, a web application will be built to ease the interaction with the aforementioned applications, through which any user will be able to conduct different bioinformatics experiments.

Lastly, a series of results regarding execution time and accuracy of the obtained solutions through the use of the applications will be gathered and analysed.

# 2 INTRODUCCIÓN

En este capítulo se presentan el contexto en el que se encuadra el presente trabajo y las circunstancias que han motivado su desarrollo. Además, se describe la estructura del presente documento, detallando el contenido de cada uno de los capítulos que lo componen.

## 2.1 Justificación y contexto del TFG

El vertiginoso ritmo con el que han evolucionado la ciencia y la tecnología ha motivado el planteamiento de diversas cuestiones sobre el origen de la vida, así como el comienzo de la búsqueda de la cura o el tratamiento de enfermedades de gran impacto social como Alzheimer, cáncer, Parkinson y SIDA. Para dar respuesta a estas preguntas, en los últimos años se han desarrollado numerosas aplicaciones bioinformáticas con las que poder llevar a cabo todos los análisis y estudios pertinentes.

En la actualidad existen algunas plataformas que permiten realizar online diversos análisis bioinformáticos como Galaxy [1] o el Navegador Genómico de la Universidad California - Santa Cruz [2]. No obstante, debido a la falta de privacidad asociada al uso de estas plataformas, al estar el servidor compartido por todos los usuarios, en ocasiones suele ser conveniente utilizar herramientas que se ejecuten de forma local en un ordenador personal.

Para la realización de este TFG, se han seleccionado dos de estas herramientas, Trinity y BLAST, encargadas de la secuenciación y alineamiento de muestras biológicas (ADN, ARN o proteínas), respectivamente.

Se define alineamiento al proceso mediante el cual una secuencia problema es comparada con un base de datos de secuencias conocidas realizándose una búsqueda de patrones que se repitan en ambas secuencias. Por su parte, la secuenciación es el proceso mediante el cual se reconstruye la secuencia de una molécula de ARN mediante el re-ensamblado de un conjunto de segmentos de partida.

Por otro lado, con la finalidad de mejorar el rendimiento de la primera, se ha desplegado una infraestructura cloud para la ejecución distribuida de las distintas tareas asociadas a la misma.

Finalmente, el procedimiento habitual de manejo de estas aplicaciones conlleva a menudo el empleo de una línea de comandos, lo cual puede llegar a ser intimidante para algunos usuarios. Para atajar esta situación se ha desarrollado una aplicación web, bautizada como **BioCloud**, de manera que sea posible una interacción cómoda y sencilla

con estas herramientas bioinformáticas. Por este motivo, uno de los objetivos que se ha perseguido en todo momento es que la interfaz gráfica diseñada sea lo más amigable e intuitiva posible.

## 2.2 Estructura del documento

Este documento consta de una serie de capítulos en los que se detallan el proceso de diseño e implementación de la infraestructura cloud. En concreto, la distribución del contenido es la que se detalla a continuación:

- **Capítulo 1. Resumen.** Se relacionarán de manera muy sintética los objetivos de este trabajo.
- **Capítulo 2. Introducción.** En este capítulo se establece el contexto en el que se encuadra la necesidad que se desea satisfacer y las soluciones adoptadas para solventar el problema de partida.
- **Capítulo 3. Objetivos.** Se detallarán los fines que se desean conseguir con este TFG.
- **Capítulo 4. Materiales y métodos.** Recoge toda la información relativa al proceso de diseño del trabajo: las tecnologías y técnicas utilizadas, modelos de arquitectura adoptados y fragmentos de código relevantes acerca de la implementación.
- **Capítulo 5. Resultados y Discusión.** Se especificarán los distintos resultados obtenidos tras la finalización de este trabajo, así como tiempos de ejecución y precisión de las soluciones obtenidas con las aplicaciones bioinformáticas utilizadas.
- **Capítulo 6. Conclusiones.** En este capítulo se exponen las conclusiones extraídas a lo largo del desarrollo de este trabajo. También se describen los conocimientos y destrezas adquiridas durante el proceso.
- **Capítulo 7. Líneas futuras.** Se mencionarán algunas líneas de trabajo futuras.
- **Capítulo 8. Referencias.** Reseñas a los libros, documentos online, revistas y sitios web que se han consultado durante la realización del TFG.
- **Capítulo 9. Anexos.** Se han incorporado una serie de anexos a modo de referencia donde se explicarán algunos conceptos básicos de Biología y se realizará una introducción sobre el funcionamiento de cada una de las herramientas bioinformáticas y los programas empleados para implementar la Infraestructura Cloud. Junto a ellos, se incluirán, además, un manual de instalación y otro de uso de la aplicación web implementada.

# 3 OBJETIVOS

Como ya se ha comentado anteriormente, el objetivo principal del presente TFG es el despliegue de una infraestructura cloud para la ejecución de aplicaciones bioinformáticas. Para cubrir dicho objetivo general es necesario satisfacer una serie de objetivos subordinados que se relacionan a continuación:

- **Familiarización con el Sistema Operativo Scientific Linux**

En este trabajo tanto las máquinas virtuales como los equipos físicos se basan en este Sistema Operativo (SO), por lo que es necesario conocer algunos comandos Linux básicos de manera que sea posible llevar a cabo las configuraciones pertinentes e instalar los paquetes requeridos.

- **Estudio y familiarización con las herramientas bioinformáticas Trinity y BLAST**

Será necesario conocer en profundidad las bases del funcionamiento de las herramientas BLAST y Trinity, y los procedimientos de instalación y configuración para que puedan ser integradas y manejadas adecuadamente desde la aplicación web.

- **Estudio y familiarización con la tecnología grid y el gestor distribuido OGS/GE**

Para poder conseguir la ejecución distribuida de Trinity, es necesario en primer lugar conocer los fundamentos teóricos de los sistemas grid, para seguidamente instalar y configurar adecuadamente el gestor OGS/GE.

- **Estudio y familiarización con la virtualización de SO y la plataforma OpenNebula**

La infraestructura grid estará conformada por una serie de máquinas virtuales que serán las encargadas de realizar las tareas de secuenciación. Debido a ello, es conveniente adquirir nociones sobre virtualización e instalación y configuración de la plataforma OpenNebula.

- **Diseño e implementación de una aplicación web para la interacción con herramientas bioinformáticas**

Esta aplicación será la encargada de recoger todos los datos proporcionados por el usuario a través de la Interfaz Gráfica de Usuario (GUI) y procesarlos para la ejecución de tareas de forma local (en el caso de BLAST) o bien, el envío de las mismas al sistema gestor para su ejecución distribuida (en el caso de Trinity). Adicionalmente, en todo momento se llevará un registro de las tareas ejecutadas y del estado de las mismas.

# 4 MATERIALES Y MÉTODOS

En este capítulo se relacionan las distintas tecnologías utilizadas durante el desarrollo del presente TFG y las razones que han propiciado su elección. A continuación, se describe la infraestructura desplegada, la estructura de la base de datos, y la distribución del código fuente en clases y paquetes. Finalmente, se detallan una serie de pruebas realizadas a la infraestructura desplegada.

## 4.1 Tecnologías usadas

A continuación, se relacionan las principales tecnologías y aplicaciones utilizadas durante la implementación de la infraestructura: patrones de arquitectura, lenguajes de programación, frameworks, entorno de desarrollo...

### 4.1.1 *Arquitectura del sistema*

La aplicación se ha diseñado siguiendo el patrón de arquitectura Modelo – Vista – Controlador (MVC). Este patrón fue introducido por primera vez en 1979 para Smalltalk y puede ser empleado en múltiples frameworks. Su principal virtud es la separación de la lógica de negocio de la interfaz de usuario, lo que facilita un desarrollo independiente de ambas partes y proporciona una gran flexibilidad al desarrollador.

### 4.1.2 *Framework Spring*

Existen numerosos frameworks para el lenguaje Java. En la implementación y desarrollo de esta aplicación web se ha utilizado Spring [3], un completo stack tecnológico que proporciona numerosas características tales como Inyección de Dependencias (DI), abstracción de acceso a datos, manejo transaccional y más. Este framework se utiliza para el desarrollo de aplicaciones empresariales, es de código abierto y sigue el patrón de diseño MVC.

### 4.1.3 *Lenguaje de programación*

Para la implementación de la aplicación web se ha escogido el lenguaje de programación Java, un lenguaje compilado de relativa sencillez que permite abordar problemas complejos facilitando la generación de código bien ordenado, estructurado y mantenible. Así mismo, es un lenguaje de programación multihilo, aspecto clave para el correcto funcionamiento de la aplicación web implementada.

Fue un proyecto iniciado por James Gosling, en aquel momento empleado de la extinta Sun Microsystems, allá por el año 1995 e inicialmente conocido como Oak. En los últimos años se ha extendido enormemente debido a su potencial y robustez. En 2009 el lenguaje pasó a ser propiedad de la corporación Oracle.

### 4.1.4 *Lenguajes de marcas*

Las diferentes pantallas de la aplicación web se han creado mediante páginas HMTL y hojas de estilo CSS. Dado que en los últimos tiempos se está apostando por el diseño web minimalista, en este trabajo se ha seguido esta tendencia, empleando en el mismo, tonos neutros como el blanco y el negro.

Para dar estilo a cada una de las vistas se ha empleado el framework Bootstrap [4], en su versión 3.3.7. Creado originalmente por Twitter, este framework con base HTML permite crear interfaz web mediante el uso de CSS y JavaScript, presentando como principal característica el que la interfaz se adapte al tamaño de la pantalla de cada dispositivo en que se visualice, es decir, permite un "R*esponsive Design*" o diseño adaptativo.

Bootstrap proporciona varios elementos con estilos predefinidos y listos para utilizar: botones, formularios, menús desplegables… lo que permite un diseño rápido, fácil e intuitivo de interfaz de usuario, aunque es extensible, pudiendo ser adaptado a las necesidades de cada momento.

### 4.1.5 *Sistema Gestor de Base de datos (SGBD)*

Con el fin de garantizar que la aplicación sea lo más ligera posible y minimizar los costes de la misma, se ha decidido escoger un SGBD que permita un uso libre, tanto comercial como privado. De entre todas las opciones disponibles se ha elegido el SGBD SQLite, en su versión 3, ya que apenas requiere configuración y ofrece una gran velocidad

de acceso a datos. Además, Scientific Linux incluye de serie los paquetes de SQLite preinstalados, por lo que no se requiere ninguna acción adicional.

### 4.1.6 *Servidor de aplicaciones*

El funcionamiento de la aplicación web, requiere única y exclusivamente un contenedor de Servlets. En el mercado existen diferentes alternativas de contenedores, siendo la más conocida y ampliamente utilizada, Apache Tomcat. En este TFG, la versión que se ha utilizado es la 7.0, la cual incluye las especificaciones de *Servlet* 3.0 y *JavaServer Pages* 2.2 [5].

### 4.1.7 *Entorno de Desarrollo*

Para la implementación de la aplicación web, se ha utilizado Eclipse [6], un entorno de desarrollo integrado (IDE) de código abierto. Desarrollado originariamente por IBM, actualmente se encuentra bajo la tutela de una organización sin ánimo de lucro llamada *Eclipse Fundation* que promueve el software libre. Eclipse fue distribuido originalmente bajo la *Common Public License*, pero después fue re-licenciado bajo la *Eclipse Public License*.

Existen diferentes ediciones de la plataforma, que permiten programar en Java y C/C++ entre otros lenguajes. De entre todas ellas, se ha empleado *Eclipse IDE for Java EE Developers*, versión Neon, en la que se ha instalado el plugin *Spring Tool Suite (STS) for Eclipse 3.8.1.RELEASE.*

### 4.1.8 *Scientific Linux*

Todas las máquinas virtuales, los equipos anfitriones de las mismas y el equipo servidor se basan en el SO Scientific Linux (SL) [7]. SL es un sistema recompilado basado en Red Hat Enterprise Linux (RHEL), co-desarrollado por el Laboratorio Nacional Fermilab y la Organización Europea para la Investigación Nuclear (CERN). Esta distribución cuenta con un reducido número de paquetes pre-instalados, lo que la hace una distribución muy ligera y rápida. La arquitectura empleada es de 64 bits, ya que así lo exige la herramienta bioinformática Trinity.

### 4.1.9 *Herramientas Bioinformáticas*

En este TFG se han utilizado dos herramientas de gran uso en el ámbito de la bioinformática: BLAST y Trinity. Para evitar que la extensión de este capítulo sea desmedida, se han incluido dos anexos, apartados 9.5 y 9.6, respectivamente, en los que se habla con mayor detalle de cada uno de ellos. Se remite pues al lector a consultar dichos anexos en caso de que desee profundizar en algún aspecto sobre los mismos.

### 4.1.10 *Open Grid Scheduler/Grid Engine*

El Sistema Gestor de Recursos Distribuidos (DRMS) empleado para gestionar los distintos recursos computacionales que componen la infraestructura grid es OGS/GE. Se invita al lector a consultar el apartado 9.7 en el caso de querer conocer más sobre este sistema gestor.

### 4.1.11 *API DRMAA v1.0*

La *Distributed Resource Management Application* API (DRMAA) [8], es una especificación de alto nivel desarrollada por el *Open Grid Forum* para la asignación y control de trabajos en un DRMS, como por ejemplo un cluster o una infraestructura de computación grid. Esta API proporciona la funcionalidad necesaria para controlar y monitorizar los trabajos asignados a los distintos recursos computacionales que componen el sistema distribuido.

Gestiona la ejecución, asignación y planificación de tareas mediante la definición de plantillas, en las que se deben especificar los parámetros a usar durante la ejecución de estas tareas, así como las variables de entorno, en el caso de que las hubiera.

Los desarrolladores tienen a su disposición un conjunto de librerías escritas en diferentes lenguajes para poder consumir este API. En el contexto de este TFG se ha utilizado la librería de conexión para Java.

DRMAA es proporcionada por el propio programa OGS/GE y se puede encontrar en el subdirectorio ge2011.11/lib.

### 4.1.12 *OpenNebula*

Para las tareas de virtualización se ha utilizado la plataforma OpenNebula, de la cual se habla en detalle en el apartado 9.8.

### 4.1.13 *NFS*

OGS/GE y OpenNebula presentan un mecanismo de funcionamiento basado en el uso del protocolo Network File System (NFS), un protocolo de nivel de aplicación, según el modelo OSI. Fue desarrollado por Sun Microsystems en 1984 y permite que un nodo servidor pueda compartir ficheros y directorios con otros nodos, de tal modo que éstos puedan acceder a estos archivos remotos.

El sistema NFS suele estar conformado por dos partes principales:

- **Servidor:** Alberga los directorios que se van a compartir.
- **Cliente/s:** Van a acceder a los directorios remotos compartidos.

## 4.2 Análisis y Diseño

En los siguientes sub-apartados se realiza un análisis de la aplicación web diseñada. Para ello, primeramente se identifican los actores que de algún modo pueden interaccionar con la aplicación web, se definen los requisitos que ésta debe cumplir, especialmente los relacionados con el comportamiento del sistema y los distintos escenarios de uso del mismo. Seguidamente, se describe la solución implementada en términos de código fuente y distribución en paquetes, estructura de la base de datos y esquema de la infraestructura desplegada. Finalmente, se reseña la metodología empleada para realizar un banco de pruebas sobre esta infraestructura.

### 4.2.1 *Identificación de usuarios*

Se pueden identificar los siguientes usuarios del sistema:

- Usuario final, para el que va destinada la aplicación web.
- El administrador del sistema, cuya función es llevar a cabo tareas de mantenimiento y actualización del sistema.

### 4.2.2 *Definición de requisitos*

En esta sección se especifican los requisitos que debe satisfacer la aplicación implementada, tanto respecto a los servicios proporcionados por el sistema (Requisitos funcionales), como a las propiedades emergentes del mismo (Requisitos no funcionales).

#### 4.2.2.1 *Requisitos funcionales*

- **RF1**. Posibilidad de realizar un análisis utilizando la herramienta BLAST.
- **RF2**. Posibilidad de llevar a cabo un análisis empleando el programa Trinity.
- **RF3**. Posibilidad de ejecutar de manera distribuida y en paralelo las tareas asociadas al programa Trinity.
- **RF4.** Aplicación web para la interacción con las herramientas bioinformáticas anteriores.
- **RF5**. No se requerirá autenticación ni registro de los usuarios ya que toda la información sobre los parámetros o el programa a utilizar son especificados en cada análisis.

#### 4.2.2.2 *Requisitos no funcionales*

- **RNF1**. Siempre que sea posible, se deberá hacer uso de software libre para minimizar los costes asociados al TFG.
- **RNF2**. La interfaz gráfica deberá ser lo más amigable posible.
- **RNF3**. La aplicación deberá desarrollarse siguiendo las normas y estándares establecidos con el fin de facilitar su comprensión a futuros desarrolladores.

### 4.2.3 *Casos de uso*

Los distintos casos de uso identificados para la aplicación web BioCloud son los que se relacionan a continuación y se muestran en el diagrama de contexto de la Figura 4.1:

- **Usar BLAST -** Realizar un análisis para el alineamiento de una muestra problema utilizando la herramienta BLAST.
- **Crear una base de datos de BLAST -** En un análisis con BLAST se efectúa una búsqueda contra una base de datos local. Esta base de datos deberá ser registrada en el sistema para que pueda ser utilizada en un futuro.
- **Usar Trinity -** Llevar a cabo un análisis empleando la herramienta bioinformática Trinity. Con dicha herramienta se realizará la secuenciación *de novo* de muestras de ARN extraídas y purificadas.
- **Visualizar tareas -** Consultar las tareas ejecutadas, el estado de las mismas y descargar los informes de salida generados al finalizar un análisis.

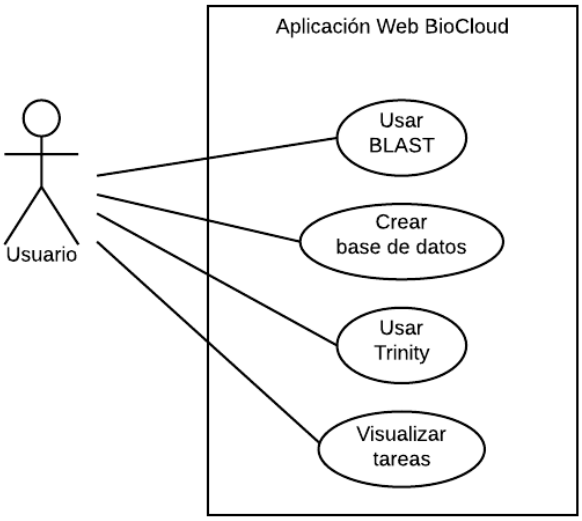



*Figura 4.1. Diagrama de contexto de los casos de uso*

#### *4.2.3.1 Usar BLAST*

**Actor principal: Usuario**

**Precondiciones: Ninguna**

**Flujo Básico:**

1. El usuario accede a la aplicación a través de la interfaz web.
2. El usuario introduce los parámetros del análisis a realizar y adjunta la secuencia problema.
3. El sistema construye el comando a ejecutar e inicia el análisis. Se añade una nueva tarea al listado.
4. La tarea se marca como completada y se activa un enlace para la descarga del informe de salida del análisis.
5. Se devuelve la vista de partida.

**Flujo Alternativo 1:**

1. El usuario accede a la aplicación a través de la interfaz web.
2. El usuario introduce los parámetros del análisis a realizar y adjunta la muestra problema.
3. El usuario ha introducido algún valor no válido en el formulario.
    a. El sistema devuelve nuevamente la vista mostrando los mensajes de error correspondientes.

**Flujo Alternativo 2:**

1. El usuario accede a la aplicación a través de la interfaz web.

2. El usuario introduce los parámetros del análisis a realizar y adjunta la muestra problema.
3. Comienza la ejecución de la tarea. El algún momento la ejecución falla.
    a. La tarea aparecerá como pendiente durante 5 horas. Pasado este tiempo, será marcada como errónea.
4. El sistema devuelve la vista de partida.

#### *4.2.3.2 Crear base de datos BLAST*

**Actor principal: Usuario**

**Precondiciones: Ninguna**

**Flujo Básico:**

1. El usuario accede a la aplicación a través de la interfaz web.
2. El usuario especifica el tipo de base de datos y adjunta el fichero a partir del cual se generará la base de datos en futuros análisis.
3. El sistema registra la base de datos en el sistema.
4. Se devuelve la vista de partida.

**Flujo Alternativo 1:**

1. El usuario accede a la aplicación a través de la interfaz web.
2. El usuario especifica el tipo de base de datos y adjunta el fichero a partir del cual se generará la base de datos en futuros análisis.
3. El usuario ha introducido algún valor no válido en el formulario.
    a. El sistema devuelve nuevamente la vista mostrando los mensajes de error correspondientes.

**Flujo Alternativo 2:**

1. El usuario accede a la aplicación a través de la interfaz web.
2. El usuario especifica el tipo de base de datos y adjunta el fichero a partir del cual se generará la base de datos en futuros análisis.
3. Comienza la ejecución de la tarea. El algún momento la ejecución falla.
    a. La base de datos no se mostrará entre las alternativas disponibles para ser usada en futuros análisis.
4. El sistema devuelve la vista de partida.

**Flujo Alternativo 3:**

1. El usuario accede a la aplicación a través de la interfaz web.
2. El usuario especifica el tipo de base de datos y adjunta el fichero a partir del cual se generará la base de datos en futuros análisis.

3. La base de datos ya ha sido dada de alta previamente en el sistema, es decir, el resumen SHA-1 de la base de datos a registrar ya existe en el sistema.
    a. La base de datos no será registrada nuevamente.
4. El sistema devuelve la vista de partida.

#### *4.2.3.3 Usar Trinity*

**Actor principal: Usuario**

**Precondiciones: Ninguna**

**Flujo Básico:**

1. El usuario accede a la aplicación a través de la interfaz web.
2. El usuario introduce los parámetros del análisis a realizar y adjunta las muestras problema.
3. El sistema envía la tarea a la cola de tareas para que el planificador la asigne al primer recurso computacional disponible.
4. La tarea se marca como completada y se activa un enlace para la descarga del informe resultante del análisis.
5. Se devuelve la vista de partida.

**Flujo Alternativo 1:**

1. El usuario accede a la aplicación a través de la interfaz web.
2. El usuario introduce los parámetros del análisis a realizar y adjunta la muestra problema.
3. El usuario ha introducido algún valor no válido en el formulario.
    a. El sistema devuelve nuevamente la vista mostrando los mensajes de error correspondientes.

**Flujo Alternativo 2:**

1. El usuario accede a la aplicación a través de la interfaz web.
2. El usuario introduce los parámetros del análisis a realizar y adjunta la muestra problema.
3. Comienza la ejecución de la tarea. El algún momento dicha ejecución falla.
    a. La tarea aparecerá como pendiente durante 5 horas. Pasado este tiempo, será marcada como errónea.
4. El sistema devuelve la vista de partida.

#### *4.2.3.4 Visualizar listado de tareas*

**Actor principal: Usuario**

**Precondiciones: Ninguna**

**Flujo Básico:**

1. El usuario accede a la aplicación a través de la interfaz web.
2. El usuario consulta de tareas del sistema.
3. El sistema devuelve una colección de tareas en forma de tabla.

**Flujo Alternativo:**

1. El usuario accede a la aplicación a través de la interfaz web.
2. El usuario consulta las tareas pendientes o en ejecución.
3. No existe ninguna tarea pendiente o en curso en el sistema.
    a. La tabla mostrada estará vacía.

### 4.2.4 *Diseño*

La aplicación web se ha diseñado siguiendo el paradigma cliente – servidor. El servidor ofrecerá una Interfaz Web para acceso mediante navegador y se divide en dos partes fundamentales:

- **Front-end o parte cliente**: Hace referencia a dicha interfaz web y los distintos componentes gráficos. Es en esta capa donde se hace uso de las páginas HTML, hojas de estilo CSS y funciones JavaScript.
- **Back-end o parte servidor**: Hace referencia a todos los servicios sobre los que se apoya el front-end. Es la parte encargada de que todo funcione correctamente y en ella se utiliza el lenguaje de programación Java y el SGBD SQLite.

En base a los casos de uso identificados en el apartado anterior, se han diseñado 4 vistas. Para cada una de las ellas, se mostrará a continuación un wireframe o diagrama de contenido:

- Formulario para el programa BLAST.
- Formulario para la herramienta Trinity.
- Formulario para registrar en el sistema una base de datos de BLAST.
- Pantalla para mostrar el listado de tareas.

Tal y como recoge la Figura 4.2, la pantalla de inicio está compuesta por una serie de imágenes que actúan a modo de enlace hacia páginas de uso frecuente en el ámbito científico.

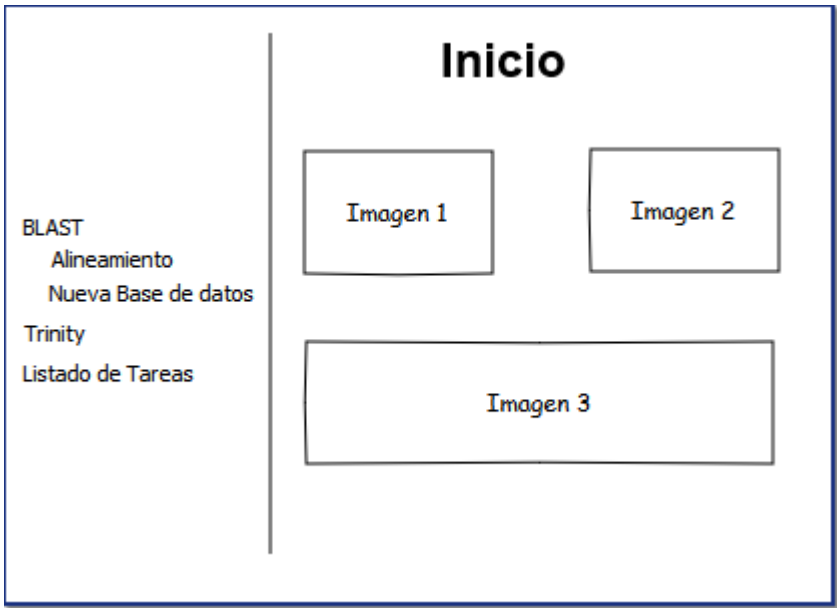



*Figura 4.2. Wireframe para la pantalla de inicio*

La pantalla para registrar en el sistema una nueva base de datos de BLAST (Figura 4.3), consiste en un formulario en el que se puede adjuntar el fichero a partir del cual se generará la base de datos y especificar el tipo de la misma, es decir, si es de proteínas o ácidos nucleicos. Debajo de cada campo se ha reservado un espacio destinado a mensajes de validación, para informar al usuario sobre valores de entrada erróneos.

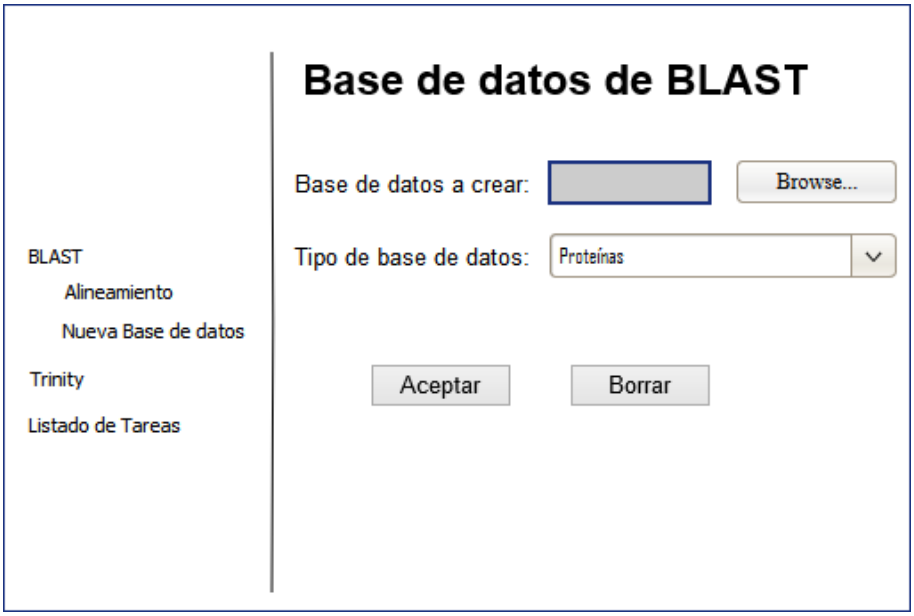



*Figura 4.3. Wireframe para la pantalla de registro de nuevas bases de datos*

La pantalla para realizar un análisis con el programa BLAST (Figura 4.4), consta de un formulario con una serie de campos a través de los cuales se pueden especificar los valores de los parámetros utilizados en cada análisis con dicha herramienta y adjuntar el fichero que contiene la secuencia problema. Nuevamente, debajo de cada campo se ha reservado un espacio destinado a mensajes de error.

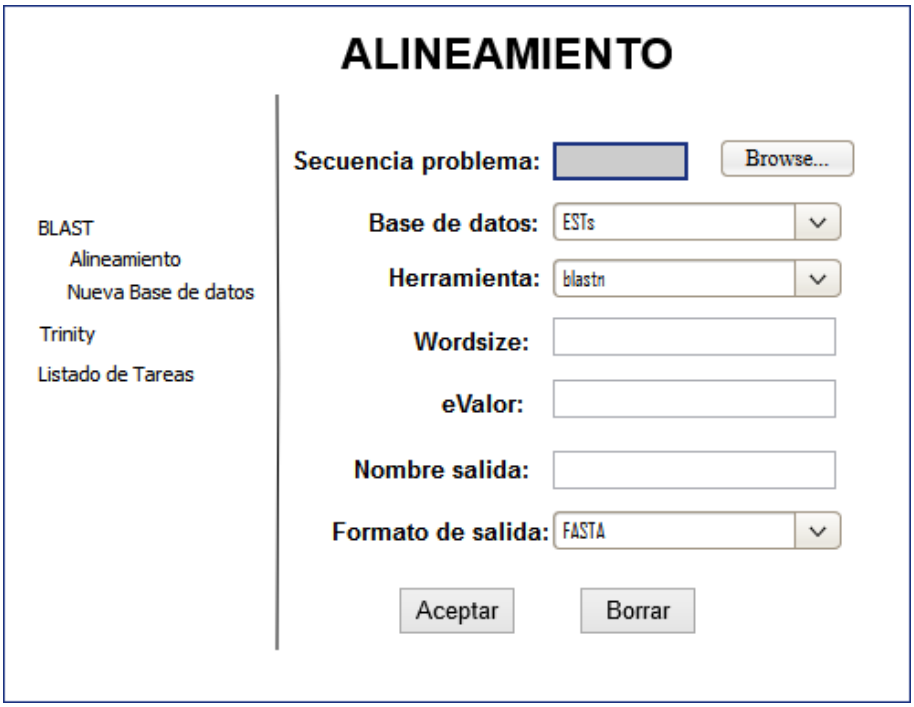



*Figura 4.4. Wireframe para la pantalla de formulario del programa BLAST*

Por otro lado, la pantalla para llevar a cabo un análisis con el programa Trinity (Figura 4.5), contiene un formulario con una serie de campos y zonas reservadas para mensajes de error.

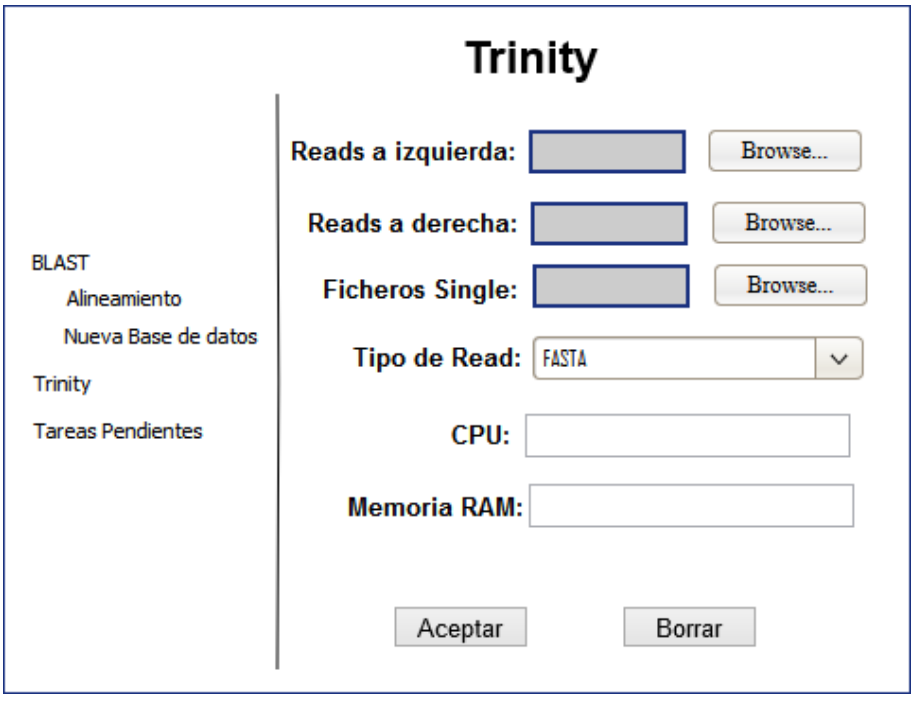



*Figura 4.5. Wireframe para la pantalla de formulario del programa Trinity*

La vista de listado de tareas (Figura 4.6) consta de una tabla que recogerá todas las tareas que ejecutadas, especificándose si ésta está pendiente, ha finalizado o ha ocurrido algún error durante su ejecución. Una vez que la tarea ha terminado, y si todo ha ido correctamente, se activará un URL a través del cual se puede descargar el informe de

salida correspondiente. Si no existiese ninguna tarea pendiente o en curso, la tabla se mostrará vacía.


*Figura 4.6. Wireframe para la pantalla de tareas pendientes*

### 4.2.5 *Modelo de clases de diseño*

La aplicación web se compone de una serie de clases Java y ha sido diseñada de tal modo que se maximice la flexibilidad del sistema y se facilite la reutilización y el reemplazo de componentes. Además, en todo momento se ha pretendido que la cantidad de código contenida en cada una de las clases sea la menor posible y que cada una de ellas posea una clara función definida: almacenar ficheros, borrar ficheros, construir comandos, registrar tareas en el sistema, declaración de constantes…

Para obtener más información sobre cada clase así como los métodos definidos en cada una de ellas, se remite al lector a la documentación Javadoc del proyecto. Ésta se puede encontrar en el interior de una carpeta llamada javadoc que está incluida en el fichero zip anexado a esta memoria y que contiene el código fuente de la aplicación web.

La estructura del proyecto es la que se muestra en la Figura 4.7. En los siguientes sub-apartados se hablará de los distintos paquetes y clases que la componen.

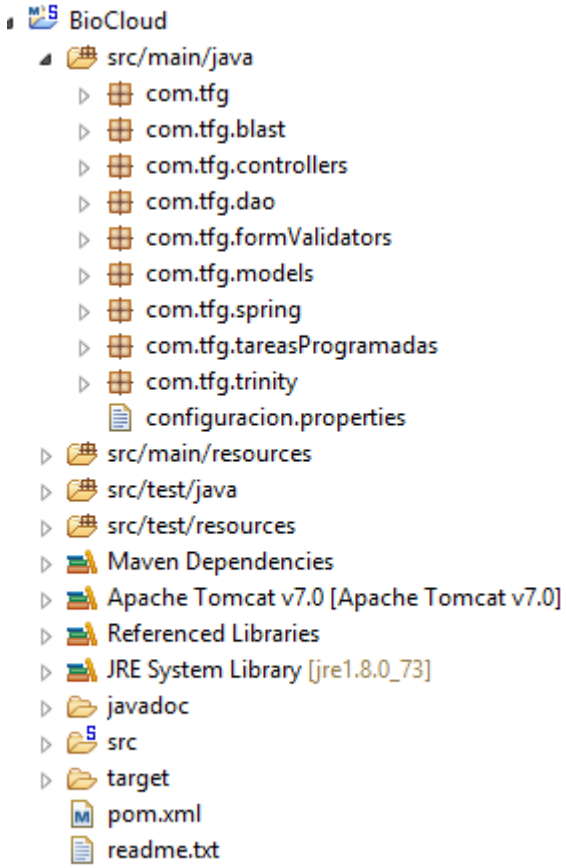



*Figura 4.7. Estructura de la Aplicación Web*

#### *4.2.5.1 Paquetes del sistema*

Para implementar la aplicación web se han creado 9 paquetes. A continuación, se enumera cada uno de estos paquetes, se indica las clases que contiene y la función de cada una de ellas.

##### 4.2.5.1.1 Paquete com.tfg

Consta de una serie de clases con diversa funcionalidad:

- **AlmacenadorFicheros:** Almacena en disco los ficheros subidos al servidor a través de la GUI.
- **Command:** Interfaz para ejecutar una operación. Implementa un tipo de patrón de diseño de software. Para conocer más acerca del mismo, consultar el apartado 9.12.1.

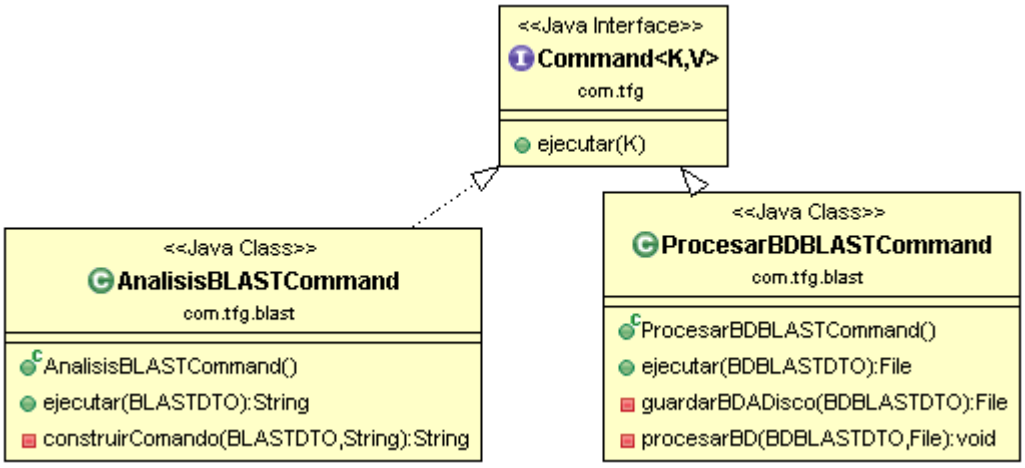



*Figura 4.8. Patrón de diseño Command*

- **Configuracion:** Permite hacer uso de un archivo llamado *configuration.properties*. En el interior del mismo se han declarado una serie de variables de configuración de la aplicación web.
- **Constants:** Define una serie de constantes utilizadas por la aplicación.
- **CopiarFicheros:** Se encarga de copiar el informe de salida del programa Trinity al directorio de salida de la aplicación web.
- **DigestUtils:** Genera un resumen del contenido de los ficheros a partir de los cuales se crean bases de datos de BLAST. Para generar dicho resumen se hace uso del algoritmo SHA-1.
- **FormateadorFechas:** Codifica las fechas en un formato acorde al uso que se les vaya a dar.
- **IPAddressUtils:** Obtiene la dirección IP del equipo servidor.
- **LanzarComandoJava:** Permite ejecutar un comando de sistema desde Java.
- **MultipartFileHolder:** Contenedor de objetos MultipartFile. Permite acceder a cada fichero adjuntado través de la vista Trinity y realizarle a cada uno de ellos de manera individual las validaciones oportunas.
- **UUIDUtils:** Genera un UUID aleatorio para cada ejecución.

#### 4.2.5.1.2 Paquete com.tfg.blast

Este paquete contiene todas las clases relacionadas con el programa bioinformático BLAST. Consta de las siguientes clases:

- **AnálisisBLASTCommand**: Implementa la interfaz Command y construye un comando para realizar un análisis utilizando la herramienta BLAST.
- **BLASTConstants**: Contiene constantes utilizadas por el programa BLAST.

- **BLASTJob**: Se encarga de todos los procesos necesarios para realizar un análisis con BLAST: guardar en disco el fichero con la secuencia problema, generar la base de datos local, ejecutar el análisis propiamente dicho y registrar la tarea en el sistema.
- **ProcesarBDBLASTCommand**: Obtiene del sistema la información para crear una base de datos de BLAST, la guarda en disco, construye el comando para generarla y lo ejecuta.
- **RegistrarBDBLASTJob**: Registra una nueva base de datos de BLAST en el sistema.

#### 4.2.5.1.3 Paquete com.tfg.controllers

Las clases incluidas en este paquete son controladores que cargan el modelo de datos e invocan a las vistas correspondientes. En este trabajo se ha creado un controlador independiente para cada vista:

- **BDBLASTController:** Permite al usuario registrar en el sistema una nueva base de datos de BLAST.
- **BLASTController:** Se encarga de ofrecer la funcionalidad de realizar un análisis empleando el programa BLAST.
- **DocumentosController:** Permite que los usuarios puedan descargarse los informes de salida generados en cada análisis.
- **HomeController:** Se encarga de mostrar al usuario la página de inicio del sistema.
- **JobsController:** Ofrece un listado de las tareas ejecutadas y el estado de las mismas.
- **TrinityController:** Realiza la misma funcionalidad que la clase *BLASTController* salvo que en este caso el programa a utilizar es Trinity.

#### 4.2.5.1.4 Paquete com.tfg.dao

Este paquete contiene las implementaciones de los *Data Access Objects* (DAO) que interactuarán con la base de datos BITACORA (Figura 4.9). Estas implementaciones se corresponden con dos tipos de patrones de diseño: Adapter y DAO (Para obtener más información sobre los mismos puede consultar los apartados 9.12.2 y 9.12.3 respectivamente).

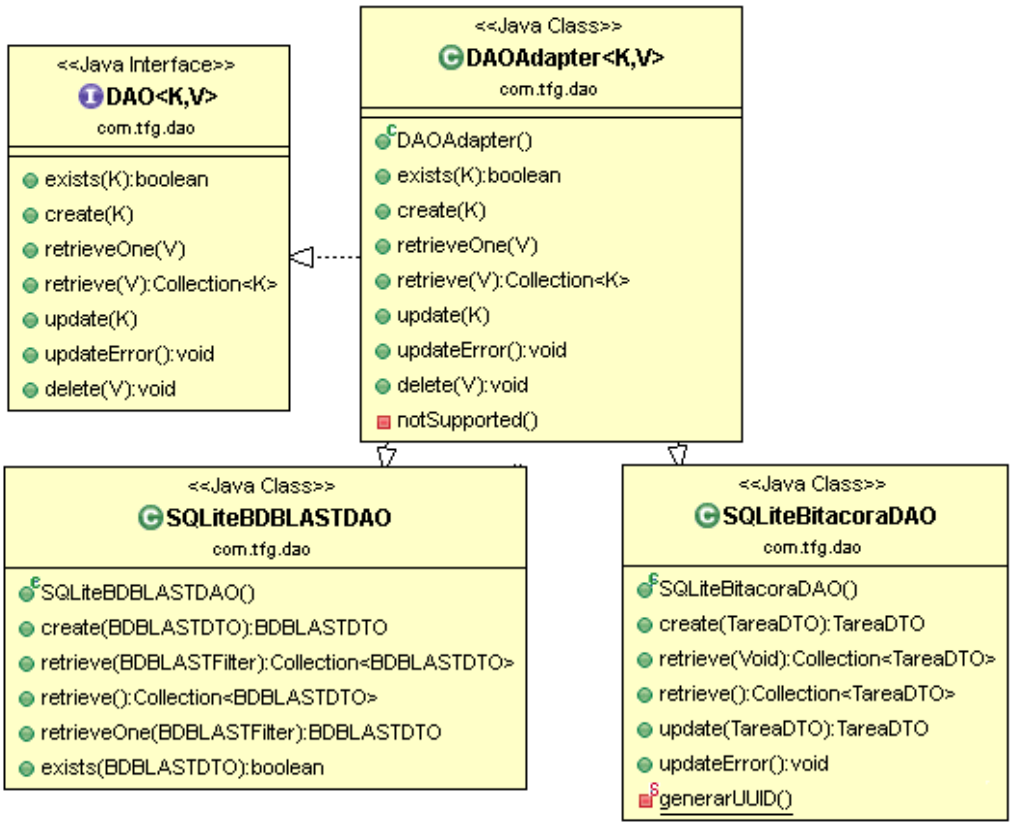



*Figura 4.9. Patrones de diseño DAO y Adapter*

- **BDBLASTFilter**: Clase cuyas variables de instancia se utilizarán como filtro en alguna consulta a la entidad BDBLAST.
- **DAO:** Esta interfaz define una implementación DAO genérica para las operaciones Create, Retrieve, Update y Detele (CRUD).
- **DAOAdapter:** Adapta la anterior interfaz para que pueda ser utilizada por las clases *SQLiteBDBLASTDAO* y *SQLiteTareaDAO*.
- **DocumentsFilter**: Clase cuya variable de instancia se utilizará como filtro en alguna consulta a la entidad Documentos.
- **SQLiteBDBLASTDAO:** Hereda de la clase *DAOAdapter* y se ocupa de recopilar y gestionar información sobre las bases de datos de BLAST.
- **SQLiteDocumentosDAO:** Hereda de la clase *DAOAdapter* y se encarga de reunir y administrar la información sobre los informes de salida generados tras la finalización de cada análisis.
- **SQLiteTareaDAO:** Hereda de la clase *DAOAdapter* y su función es la de coleccionar y gestionar información sobre las tareas que se han iniciado hasta el momento y el estado de las mismas.

#### 4.2.5.1.5 Paquete com.tfg.formValidators

Este paquete contiene las clases encargadas de validar, en el lado del servidor, los datos introducidos a través de la GUI. Para ello, se han creado las clases que se detallan a continuación:

- **BDBLASTFormValidator:** Implementa la interfaz Validator y su función es comprobar los datos introducidos en el formulario de la vista "*Nueva Base de datos*". Se han configurado una serie de reglas de validación para esta clase:
  - **Fichero:** Se requiere que se adjunte algún archivo. Además, se comprobará que el nombre de este fichero sea cualquier cadena, sin distinguir mayúsculas o minúsculas, que se componga como mínimo de un carácter alfanumérico (letras y números), guión medio, guión bajo o arroba, y opcionalmente esté seguido de una extensión compuesta del carácter punto y al menos una letra.
  - **tipoBd:** Se requiere que este parámetro sea no nulo y no vacío.
- **BLASTFormValidator:** Implementa la interfaz Validator y verificará los datos introducidos en el formulario de la vista "*Alineamiento*". De nuevo, se han configurado una serie de reglas de validación para esta clase:
  - **Fichero:** Se requiere que se adjunte un archivo. Su nombre deberá ser cualquier cadena, sin distinguir mayúsculas o minúsculas, que se componga como mínimo de un carácter alfanumérico (letras y números), guión medio, guión bajo o arroba, y opcionalmente esté seguido de una extensión compuesta del carácter punto y al menos una letra.
  - **Word size:** Se precisa que este parámetro sea un entero, no nulo y mayor que 4. No puede estar vacío.
  - **eValor:** Se requiere que este parámetro sea un número real, no nulo y no vacío.
  - **Nombre del informe de salida:** Se necesita que este parámetro sea no nulo esté formado por cualquier cadena, sin distinguir mayúsculas o minúsculas, que se componga como mínimo de un carácter alfanumérico (letras y números), guión medio, guión bajo o arroba, y opcionalmente esté seguido de una extensión compuesta del carácter punto y al menos una letra. Su extensión no puede exceder los 25 caracteres.

- **TrinityFormValidator:** Implementa la interfaz Validator y se encarga de comprobar los datos introducidos en el formulario de la vista "*Trinity*". Se han configurado una serie de reglas de validación para esta clase:
    - **Ficheros:** Se requiere que se adjunte como mínimo un archivo. Además, se comprobará que el nombre de estos ficheros sea cualquier cadena, sin distinguir mayúsculas o minúsculas, que se componga como mínimo de un carácter alfanumérico (letras y números), guión medio, guión bajo o arroba, y opcionalmente esté seguido de una extensión compuesta del carácter punto y al menos una letra.
        - **Left y Right reads:** Si se adjunta un archivo en alguno de estos dos campos, obligatoriamente se hacer lo mismo en el otro. En caso contrario se mostrará un mensaje de error.
        - **Single reads:** Sólo se requiere adjuntar un archivo en este campo. Si los dos campos anteriores tienen un fichero adjunto, este campo puede estar vacío.
    - **CPU:** Se precisa que este parámetro sea un entero igual o mayor a 2. No puede estar vacío.
    - **maxMemory:** Se requiere que este parámetro sea un entero igual o mayor a 1. No puede estar vacío.
    - **seqType:** Se necesita que este parámetro sea no nulo y no vacío.

#### 4.2.5.1.6 Paquete com.tfg.models

Este paquete contiene clases que transportan información sobre las herramientas bioinformáticas empleadas por la aplicación:

- **BDBLASTDTO:** Los objetos de esta clase portan información sobre bases de datos de BLAST. Define una serie de variables de instancia:
    - **fichero –** Fichero que será usado para generar la base de datos.
    - **nombreBD –** Nombre de la base de datos.
    - **tipoBD –** Tipo de la base de datos: proteínas o nucleótidos.
    - **contenido –** Contenido del fichero.
    - **resumen –** Resumen SHA-1 del contenido.
- **BLASTDTO:** Los objetos de esta clase transportan información sobre distintos parámetros utilizados en los análisis con BLAST. Define una serie de variables de instancia:
    - **fichero –** Fichero que contiene la secuencia problema.

    - **nombreDirectorioTemporal –** Nombre del directorio temporal en el que se almacenará el fichero mientras dure el análisis.
    - **nombreBd –** Nombre de la base de datos de BLAST contra la que se efectuará la búsqueda.
    - **bd –** Fichero a partir del cual se generará la base de datos de BLAST.
    - **herramienta –** Herramienta del conjunto de programas de BLAST que se va a usar en el análisis.
    - **wordSize –** Longitud de la palabra utilizada en el alineamiento.
    - **evalue –** Umbral que determina el grado de significación de los alineamientos producidos.
    - **nombreInformeSalida –** Nombre que tendrá el informe de salida.
    - **formatoSalida –** Extensión del informe de salida.
- **DocumentosDTO:** Los objetos de esta clase transportan información sobre los informes de salida generados al finalizar un análisis. Define una serie de variables de instancia:
    - **UUID –** Identificador único para cada tarea que se inicia.
    - **nombreInforme –** Nombre del informe de salida.
    - **extensionInforme –** Formato que tendrá el informe de salida generado.
- **TareaDTO:** Los objetos de esta clase transportan información sobre tareas finalizadas, pendientes o que se están ejecutando. Define una serie de variables de instancia:
    - **programa –** Nombre del programa utilizado para realizar el análisis: Trinity o BLAST.
    - **fechaInicio –** Fecha y hora a la que se inició el análisis.
    - **estado –** Estado de la tarea: (1) En ejecución. (2) Tarea finalizada correctamente. (3) Tarea con errores.
    - **uuid –** Identificador único universal asignado a cada tarea. Es generado aleatoriamente.
    - **comandoLanzado –** Comando que se va a ejecutar.
- **TrinityDTO:** Los objetos de esta clase poseen información sobre parámetros utilizados en los análisis con Trinity. Define una serie de variables de instancia:
    - **ficheros –** Listado de los ficheros que contienen las secuencias problema.
    - **nombreDirectorioTemporal –** Nombre del directorio temporal en el que se almacenarán estos ficheros mientras dure el análisis.

- **seqType -** Tipo de secuencia de las muestras problema: FASTA o FASTQ.
- **cpu –** El número de CPU a utilizar en el análisis, es decir, el número de hebras que se ejecutarán en paralelo.
- **maxMemory –** Máxima cantidad de memoria reservada para Trinity.

#### 4.2.5.1.7 Paquete com.tfg.spring

Este paquete consta una única clase llamada **BioCloudConfig** que contiene un bean de tipo *JdbcTemplate* que será posteriormente inyectado en las clases DAO. La clase *JdbcTemplate* encapsula todos los detalles de conexión a la base de datos y gestión de recursos, siendo el propio framework Spring el encargado de establecer y finalizar las conexiones con la misma.

#### 4.2.5.1.8 Paquete com.tfg.tareasProgramadas

Este paquete contiene una serie de clases que se encargan de realizar tareas de forma periódica y programada:

- **ActualizarTareasErroneas:** Marca como una tarea como errónea si ésta tienen una antigüedad superior a 5 horas y no han finalizado.
- **BorradorBDBLAST:** Elimina los ficheros asociados a las bases de datos de BLAST utilizadas durante un análisis. Éstos serán borrados si no han sido modificados en las últimas 3 horas.
- **BorradorDirectorios:** Elimina los directorios temporales de trabajo cuya última modificación haya tenido lugar hace más de 3 horas.

#### 4.2.5.1.9 Paquete com.tfg.trinity

- **SubmitTareaTrinityCommand:** Se encarga de establecer los parámetros de configuración de las tareas que se mandan al gestor de colas OGS/GE y del envío propiamente dicho de las mismas. Es una versión modificada de una serie de clases ejemplo proporcionadas por la comunidad de desarrolladores de DRMAA. La versión original puede ser descargada desde el siguiente enlace: http://gridscheduler.sourceforge.net/howto/drmaa_java.html
- **TrinityJob**: Se ocupa de todas las tareas necesarias para realizar un análisis utilizando la herramienta Trinity: almacenar en disco los ficheros

subidos a través del servidor, comprobar si ha finalizado la tarea, marcarla como finalizada...

### 4.2.6 *Estructura de la base de datos*

En las siguientes secciones, se describe la estructura de la base de datos BITACORA en términos de las entidades que la componen así como sus atributos y tipo de datos.

#### *4.2.6.1 Diagrama Entidad – Relación*

La lógica aplicada a la gestión de la información generada durante el funcionamiento de la aplicación web se resuelve en un esquema relacional y es implementada sobre una base de datos de SQLite.

En la Figura 4.10 se muestra el diagrama entidad-relación de la base de datos diseñada. Dado que las necesidades de la aplicación en relación a la cantidad de información a gestionar no son muy grandes, se ha visto que es suficiente emplear tres entidades independientes:

- **BDBLAST**: Es la entidad donde se representa la información relativa a las bases de datos de BLAST: tipo de base de datos (nucleótidos o proteínas), fecha de alta en el sistema, contenido de los ficheros subidos a través de la Interfaz Gráfica de Usuario (GUI) y resumen SHA-1 del mismo.
- **DOCUMENTOS**: Esta entidad almacena información referente a los informes de salida generados por ambas herramientas informáticas una vez que un análisis ha concluido: identificador universal asignado al análisis, nombre y extensión del informe de salida y contenido de dicho informe de salida.
- **TAREASLANZADAS**: Representa la información relacionada con las tareas que registradas en el sistema: Nombre del programa utilizado, fecha y hora de inicio y finalización de la tarea, comando ejecutado, identificador universal asignado a cada una de ellas y estado de la misma [(1) Pendiente, (2) Finalizada, (3) Error].

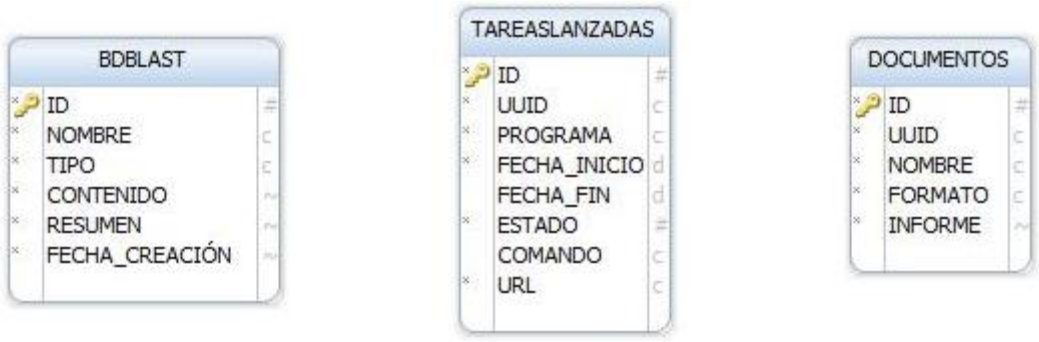



*Figura 4.10. Diagrama Entidad-Relación*

### *4.2.6.2 Modelo Relacional*

En las tablas Tabla 4.1, Tabla 4.2 y Tabla 4.3 se describen cada una de las entidades anteriores: sus atributos, tipo de datos y descripción de los mismos.

**BDBLAST**

| Atributo | Tipo | Longitud | Obligatorio | Descripción |
|---|---|---|---|---|
| **ID** | Entero | - | Sí | Clave primaria de la entidad. Autonumérico |
| **NOMBRE** | Cadena de Caracteres | 25 | Sí | Nombre de la base de datos de BLAST |
| **TIPO** | Cadena de Caracteres | 5 | Sí | Tipo de base de datos |
| **CONTENIDO** | BLOB | - | Sí | Contenido del fichero proporcionado por el usuario |
| **RESUMEN** | BLOB | - | Sí | Resumen SHA-1 del contenido anterior |
| **FECHA_CREACION** | Datetime | - | Sí | Fecha y hora de registro en el sistema |

*Tabla 4.1. Entidad BDBLAST*

**DOCUMENTOS**

| Atributo | Tipo | Longitud | Obligatorio | Descripción |
|---|---|---|---|---|
| **ID** | Entero | - | Sí | Clave primaria de la entidad. Autonumérico |
| **UUID** | Cadena de Caracteres | 30 | Sí | Identificador único para cada documento |
| **NOMBRE** | Cadena de Caracteres | 30 | Sí | Nombre del informe de salida |
| **FORMATO** | Cadena de Caracteres | 30 | Sí | Extensión del informe de salida |
| **INFORME** | Blob | - | Sí | Contenido del informe de salida |

*Tabla 4.2. Entidad DOCUMENTOS*

**TAREASLANZADAS**

| Atributo | Tipo | Longitud | Obligatorio | Descripción |
|---|---|---|---|---|
| **ID** | Entero | - | Sí | Clave primaria de la entidad. Autonumérico |
| **UUID** | Cadena de Caracteres | 30 | Sí | Identificador único para cada tarea |
| **PROGRAMA** | Cadena de Caracteres | 10 | Sí | Programa usado |
| **FECHA_INICIO** | Datetime | - | Sí | Fecha y hora de comienzo de la tarea |
| **FECHA_FIN** | Datetime | - | No | Fecha y hora de finalización de la tarea |
| **ESTADO** | Entero | - | Sí | Estado de ejecución de la tarea |
| **COMANDO** | Cadena de Caracteres | 100 | No | Comando que se va a ejecutar |
| **URL** | Cadena de Caracteres | 100 | Sí | URL de descarga del informe de salida |

*Tabla 4.3. Entidad TAREASLANZADAS*

### 4.2.7 *Planificación de tareas*

El correcto funcionamiento de la aplicación web requiere una serie de tareas que se ejecutarán de forma planificada cada 30 minutos. Para conseguirlo se ha utilizado la anotación **@Scheduled** de Spring [9]. A continuación, se detalla en qué consiste cada una de estas tareas.

#### *4.2.7.1 Control de trabajos erróneos*

El mecanismo utilizado por la aplicación web para comprobar si una tarea ha finalizado o no es la existencia del correspondiente informe de salida. Si una tarea falla durante su ejecución no se generaría dicho informe, quedando ésta como pendiente de forma indefinida. Por esta razón, se ha planificado una tarea que las marque como erróneas si han pasado más de 5 horas desde su inicio, no han sido declaradas ya como fallidas y no han finalizado, es decir, su estado es igual a 1 (Figura 4.11).

```
@Scheduled(cron = "* 0,30 * * * ?")
public void actualizaTareaErronea(){
    dao.updateError();
}
```

*Figura 4.11. Tarea de actualización de estados*

La sentencia SQL utilizada para llevar a cabo dicha actualización es la que se muestra a continuación:

*UPDATE TAREASLANZADAS*
*SET ESTADO=3, FECHA_FIN=datetime('now','localtime')*
*WHERE FECHA_INICIO <= datetime('now', 'localtime','-5 hour') AND ESTADO=1;*

#### *4.2.7.2 Borrado de ficheros obsoletos*

Para evitar sobrecargar el servidor con ficheros utilizados en ejecuciones anteriores, se ha implementado una tarea que borrará aquellos directorios cuya última fecha de modificación sea superior a 3 horas, es decir, 10.800.000 milisegundos (Figura 4.12).

```
@Scheduled(cron = "* 0,30 * * * ?")
public void borraDirectorio() {

    File dir = new File(DIRECTORIO_ENTRADA_ANALISIS);

    for (String nombreFichero : dir.list()) {

        File fichero = new File(DIRECTORIO_ENTRADA_ANALISIS + nombreFichero);

        if (System.currentTimeMillis() - fichero.lastModified() >= 10800000) {

            fichero.delete();

            try {
                FileUtils.deleteDirectory(fichero);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

*Figura 4.12. Tarea de borrado de directorios obsoletos*

Por otro lado, la aplicación web se ha diseñado de tal manera que cuando un usuario desee registrar una nueva base de datos, el contenido del fichero que éste proporciona a través de la GUI sea almacenado en el sistema. Así, cada vez que un usuario desee realizar un análisis con BLAST, obtendrá del sistema la información empleada para generar la base de datos solicitada, ésta se almacenará en disco y se creará a continuación la base de datos local. Todos estos archivos dejarán de ser útiles una vez que el análisis haya concluido, por ello se ha programado una tarea adicional que los elimine siempre y cuando no se hayan modificado desde hace más de 3 horas (Figura 4.13).

```
@Scheduled(cron = "* 0,30 * * * ?")
public void borraBDBLAST() {

    File dir = new File(DIRECTORIO_BBDD_BLAST);

    for (String nombreFichero : dir.list()) {

        File fichero = new File(DIRECTORIO_BBDD_BLAST + separator + nombreFichero);

        if (System.currentTimeMillis() - fichero.lastModified() >= 10800000) {

            fichero.delete();

        }
    }
    }
}
```

*Figura 4.13. Tarea de borrado de bases de datos de BLAST*

### 4.2.8 *Esquema de la Infraestructura Cloud*

Finalmente, se describe el esquema general de la infraestructura desplegada (Figura 4.14). Esta infraestructura está conformada por un equipo central llamado Servidor y en él se instalarán los programas y se ubicarán todos los archivos necesarios para el correcto funcionamiento de la infraestructura cloud. Todos los ficheros se situarán en un directorio compartido vía NFS por el Servidor, siendo así accesibles por los recursos computacionales, es decir, las máquinas virtuales (Grid01, Grid02,…, GridN). Estos recursos computacionales serán los encargados de ejecutar de manera distribuida las tareas asociadas con el programa Trinity. Junto a ellos, se encuentran los host anfitriones, llamados Nodo01, Nodo02 y Nodo03, que proporcionarán a las máquinas virtuales los recursos computacionales que éstas necesitan.

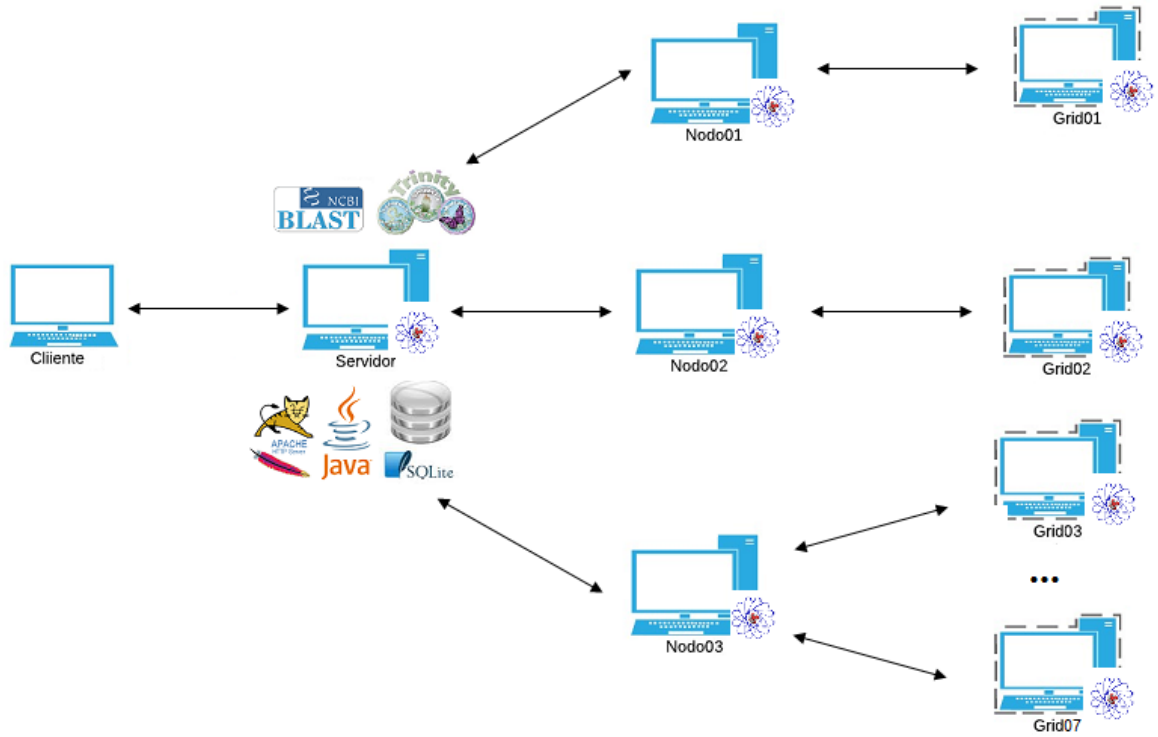



*Figura 4.14. Esquema General de la Infraestrucutra Cloud*

La infraestructura desplegada funcionará en su conjunto de la siguiente manera: Los clientes accederán a la aplicación web y realizarán una determinada tarea: iniciar un análisis con BLAST o Trinity, registrar una nueva base de datos de BLAST o consultar las tareas pendientes. En el caso de BLAST, el programa se ejecutará de forma local en el propio servidor, mientras que, en el caso de Trinity, al requerir tareas de mayor complejidad se utilizará un DRMS para que éstas sean ejecutadas de forma distribuida por el conjunto de máquinas virtuales que componen la infraestructura grid. Finalizadas todas estas tareas, los resultados serán devueltos al servidor para su reorganización y generación del informe de salida correspondiente.

Además, cada 30 minutos se llevarán a cabo una serie de tareas de mantenimiento para el borrado de archivos y directorios obsoletos y actualización del estado de las tareas erróneas.

### 4.2.9 *Medidas de tiempo de ejecución y de precisión de resultados*

A continuación, se detalla la metodología que se ha seguido para realizar una serie de estudios y medidas de tiempos de ejecución de la infraestructura cloud. En primer lugar, se describe el entorno en el que se han realizado todos estos estudios. A continuación, se reseña el procedimiento seguido para estudiar cómo los valores asignados a dos parámetros diferentes afectan a la exactitud y precisión de los resultados obtenidos con el programa BLASTN. Finalmente, se describen los escenarios en los que se han realizado una serie de medidas de tiempos de ejecución con el programa Trinity y a través de las cuales se ha comprobado cómo distintas configuraciones afectan a estos tiempos de ejecución.

#### *4.2.9.1 Descripción del entorno*

Las especificaciones técnicas de cada uno de los nodos que componen la infraestructura desplegada son las que se indican a continuación:

- **Servidor:**
  - Hardware: Core2 Duo CPU 1.60GHz, 4 GB de memoria RAM
  - Núcleos: 2
  - Sistema Operativo anfitrión: Scientific Linux 7.2
- **Nodo01:**
  - Hardware: Intel(R) Core(TM) i5 CPU 760 @ 2.80GHz, 8GB de memoria RAM
  - Núcleos: 4
  - Sistema Operativo anfitrión: Scientific Linux 7.2
- **Nodo02:**
  - Hardware: Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz, 8GB de memoria
  - Núcleos: 4
  - Sistema Operativo anfitrión: Scientific Linux 7.2

- **Nodo03:**
  - Hardware: Intel(R) Xeon(R) CPU X5450 @ 3.00GHz, 32 GB de memoria RAM.
  - Núcleos: 8
  - Sistema Operativo anfitrión: Scientific Linux 7.2
- **Entorno de ejecución homogéneo:**
  - Sistema Operativo: Scientific Linux 7.2
  - Número de máquinas virtuales: 4
    - CPU: 1
    - vCPU: 4
    - RAM: 4GB
  - OpenNebula: 4.12
- **Entorno de ejecución heterogéneo:**
  - Sistema Operativo: Scientific Linux 7.2
  - Número de máquinas virtuales: 3
    - CPU: 1
    - vCPU: 4
    - RAM: 4GB
  - Número de máquinas virtuales: 1
    - CPU: 2
    - vCPU: 8
    - RAM: 8GB
  - OpenNebula: 4.12

#### *4.2.9.2 Exactitud de soluciones con BLAST*

Para demostrar la influencia de los valores asignados a los distintos parámetros de BLAST sobre la validez de los resultados y la velocidad con la que éstos se obtienen, se ha llevado a cabo un estudio en el que se ha analizado el número de alineamientos producidos y el tiempo de ejecución requerido en función de tres parámetros: e-Valor, tamaño de palabra (Word size) y Threshold. En caso de que desee conocer el significado de estos parámetros, se remite al lector al apartado 9.5.7.

En los análisis se ha utilizado una base de datos llamada ESTs, conformada por un conjunto de 2000 secuencias de nucleótidos pertenecientes al nematodo *Caenorhabditis elegans* [10]. La secuencia problema, EST, es una de las secuencias contenidas en dicha base de datos. Ambos ficheros se pueden descargar desde la URL: http://examples.oreilly.com/9780596002992/

Para representar gráficamente algunos de los alineamientos obtenidos en los distintos análisis se ha empleado una herramienta llama Mview [11]. En el apartado 9.10 se explica con detalle el procedimiento de uso de dicho programa y el significado de cada símbolo y cada color empleado.

#### *4.2.9.3 Tiempos de ejecución con Trinity*

En el caso del programa Trinity, se ha realizado un estudio comparativo sobre los tiempos de ejecución necesarios tanto si se emplea un DRMS como si no se hace uso del mismo.

Para el primer caso, se han implementado dos escenarios diferentes:

- Escenario homogéneo en el que todas las máquinas virtuales están dotadas de 4 vCPU y 4GB de memoria RAM.
- Escenario heterogéneo en el que se desplegarán 4 máquinas equipadas con 4 vCPU y 4GB de memoria RAM y 3 dotadas con 8 vCPU y 6GB de memoria RAM.

En el otro caso, las pruebas se han realizado utilizando una máquina virtual con 4vCPU y 4GB de memoria RAM.

En ambos casos, se han realizado 10 ejecuciones hallándose a continuación, la media aritmética de los valores obtenidos en cada caso.

La cantidad de tareas asignadas a cada recurso computacional se especifica a través de un fichero de configuración propio de Trinity, llamado *BroadInst_SGE.test.conf,* y más concretamente, por medio del parámetro llamado *cmds_per_node*.

Para llevar a cabo este estudio, se han utilizado dos lotes de muestras de distinto tamaño. El primero de ellos, está compuesto por muestras de pequeño tamaño. Éstas son proporcionadas por el propio programa Trinity y se pueden encontrar en el directorio *trinityrnaseq-2.2.0/sample_data/test_Trinity_Assembly*. Además, empleando las muestras de pequeño tamaño, se llevará a cabo otro análisis comparativo del efecto que el modo de asignación de tareas tiene sobre el tiempo de ejecución. Así, se realizarán 10 análisis asignando las muestras de forma individual, es decir, tarea a tarea, y otros 10, asignándolas en lotes de 10. Los valores asignados cada parámetro en estos análisis son los que se muestran en la Tabla 4.4.

| Parámetro | Valor |
|---|---|
| **Reads a izquierda** | reads.left.fq |
| **Reads a derecha** | reads.right.fq |
| **Tipo de secuencia** | fq |
| **CPU** | 2 |
| **Máxima cantidad de memoria** | 1 |
| **Cantidad de comandos asignados a cada recurso** | 1 ó 10 |

*Tabla 4.4. Parámetros usados con el programa Trinity para muestras pequeñas*

El segundo lote contiene muestras de mayor tamaño. Estas muestras han sido descargadas desde el sitio web: http://evomics.org/learning/genomics/trinity/. Los valores utilizados en este caso se muestran en la Tabla 4.5.

| Parámetro | Valor |
|---|---|
| **Reads a izquierda** | mouse_left.fasta |
| **Reads a derecha** | mouse_right.fasta |
| **Tipo de secuencia** | fa |
| **CPU** | 2 |
| **Máxima cantidad de memoria** | 1 |
| **Cantidad de comandos asignados a cada recurso** | 50 |

*Tabla 4.5. Parámetros usados con el programa Trinity para muestras mayores*

Finalmente, se determinará el tiempo necesario para secuenciar las muestras de mayor tamaño en relación al número de máquinas virtuales desplegadas.

Entre cada ejecución y para evitar que el SO de las máquinas virtuales se sature al quedar sin memoria RAM disponible se vea obligado a hacer uso de la memoria Swap, se limpiarán ambas memorias ejecutando el siguiente comando.

```
$ sudo sync && sudo sysctl -w vm.drop_caches=3 && sudo swapoff -a ; sudo swapon -a
```

### 4.2.10 *Prueba de carga*

Una prueba de carga se define como el proceso que se le impone a un sistema basado en una cantidad predefinida de peticiones o procedimientos con la finalidad de determinar su comportamiento en esta situación [12].

Cuando se está desarrollando una aplicación web es recomendable realizar sobre ella diferentes pruebas de estrés y de carga y concurrencia para comprobar cómo se comporta dicha aplicación y detectar la existencia de cuellos de botella y por lo tanto, si es

necesario implementar estrategias de balanceo de carga. En el mercado existen numerosas herramientas que permiten realizar este tipo de pruebas. De ente todas ellas, en este TFG se ha utilizado la herramienta ***Apache Bench*** [13].

Durante la prueba de carga se harán 10.000 peticiones, realizándose 100 peticiones de manera concurrente. Para ello, se ejecutará el siguiente comando:

```
$ ab -n 10000 -c 100 -g grafica.tsv http://localhost:8080/biocloud/
```

Para crear una gráfica a partir los resultados de la prueba y representarlos de una manera más visual se utilizará la herramienta Gnuplot. Esta herramienta utiliza plantillas para generar gráficas. En este caso, la plantilla utilizada es la siguiente:

```
# La imagen resultante estará en formato PGN
set terminal png size 600

# La imagen se guardará con el nombre resultados.png
set output "resultados.png"

# Título del gráfico
set title "1000 peticiones, 100 peticiones concurrentes"

# Relación entre la altura y la anchura de la gráfica
set size ratio 0.6

# Añade una cuadrícula al eje Y
set grid y

# Etiqueta eje X
set xlabel "Peticiones"

# Etiqueta eje Y
set ylabel "Tiempo de respuesta (ms)"

plot " grafica.tsv " using 9 smooth sbezier with lines title "BioCloud"
```

Finalmente, para crear el gráfico correspondiente se ejecuta el siguiente comando, donde plot.p es el nombre de la plantilla usada:

```
$ gnuplot plot.p
```

## 4.3 Conclusiones

En este capítulo se ha realizado un análisis las diferentes tecnologías usadas durante el desarrollo de este trabajo en cuanto a modelo de arquitectura del sistema, lenguajes de programación y de marcas, frameworks, SGBD, entorno de desarrollo, herramientas bioinformáticas, plataforma de virtualización…

Por otro lado, se ha llevado a cabo la descripción de la infraestructura cloud desplegada en términos de estructura de la base de datos y de la infraestructura propiamente dicha, identificación de actores y casos de uso de la aplicación web BioCloud, relación de paquetes y clases que la componen y procedimiento para la ejecución de un banco de pruebas de evaluación del sistema.

# 5 RESULTADOS Y DISCUSIÓN

En las siguientes páginas se detallan los resultados obtenidos en los diferentes test y análisis realizados a la infraestructura desplegada respecto a tiempos de ejecución con Trinity, precisión de las soluciones obtenidas con el programa BLAST y prueba de carga del servicio web.

## 5.1 Resultado de este TFG

El resultado de este TFG ha sido el diseño e implementación de una infraestructura cloud para su uso con las herramientas bioinformáticas BLAST y Trinity. Para ello, primeramente, ha sido necesaria la adquisición de una serie de conocimientos y destrezas para llevar a buen término la ejecución de todos los objetivos planteados.

Una vez finalizada la fase de implementación de dicha infraestructura, se ha conseguido llevar a cabo con éxito una serie de análisis haciendo uso de ambas herramientas, pudiéndose asignar diferentes valores a los distintos parámetros de cada herramienta a través de la GUI diseñada para ello.

OpenNebula y OGS/GE poseen una fuerte dependencia del protocolo NFS. Así, es necesario que los directorios compartidos utilizando dicho protocolo estén disponibles para los distintos recursos computacionales que componen la infraestructura cloud.

## 5.2 Medidas de tiempo de ejecución y de precisión de resultados

En los siguientes sub-apartados se recogen los distintos resultados obtenidos en lo que respecta a precisión de resultados con el programa BLASTN y a los tiempos de ejecución que posee la herramienta Trinity utilizando un DRMS y sin hacer uso del mismo.

### 5.2.1 *Precisión de resultados con BLAST*

En las siguientes páginas, se muestran los resultados obtenidos en términos de tiempo de ejecución y de alineamientos producidos respecto al valor de diferentes parámetros tales como el tamaño de palabra o el parámetro e-Valor. Para ello, se ha hecho uso de las herramientas BLASTN y BLASTP.

### 5.2.1.1 BLASTN

En este caso, se ha determinado la influencia que los valores asignados a los parámetros Word size y E-Valor tienen sobre el número de alineamientos producidos y el tiempo necesario para realizar cada análisis.

A continuación, y para evitar que extender demasiado la longitud de este sub-apartado, se mostrará la representación gráfica de los 60 primeros símbolos de algunos de los alineamientos obtenidos. Concretamente, se expondrán los alineamientos obtenidos para un tamaño de palabra de 4, 11 y 30 y un e-Valor igual a 10 (Figura 5.1, Figura 5.2 y Figura 5.3).

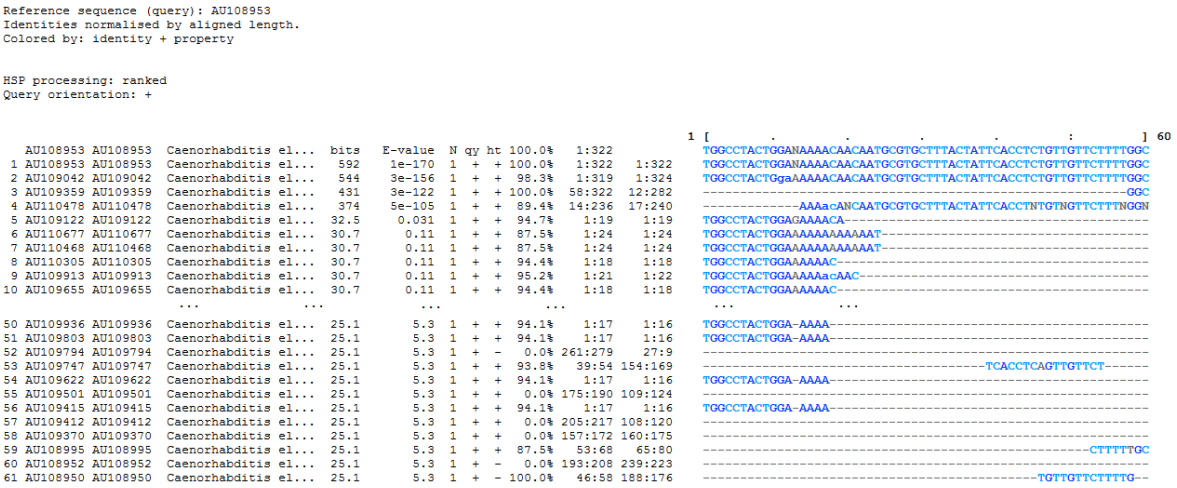



*Figura 5.1. Alineamientos producidos con Word size=4 y e-Valor=10*

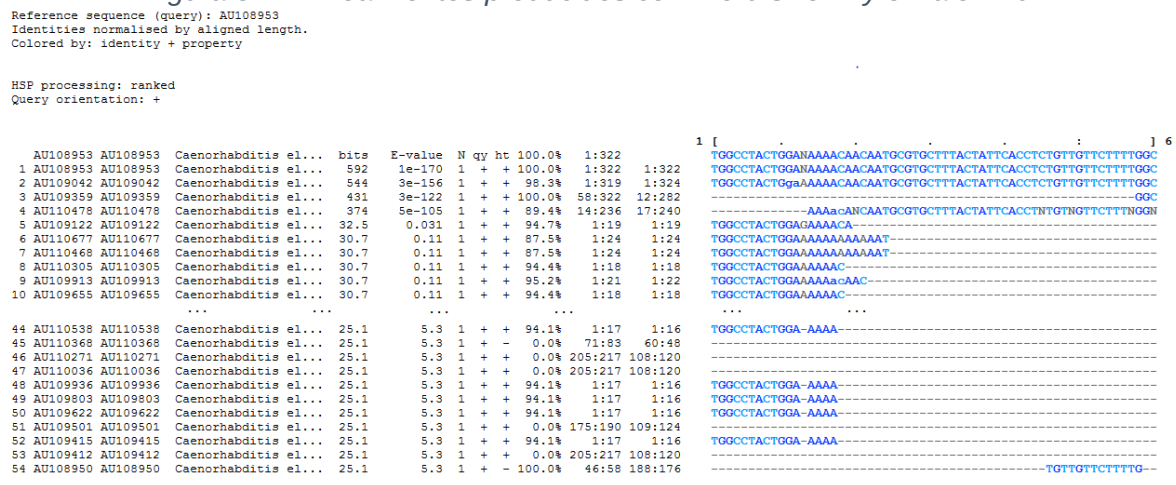



*Figura 5.2. Alineamientos producidos con Word size=11 y e-Valor=10*

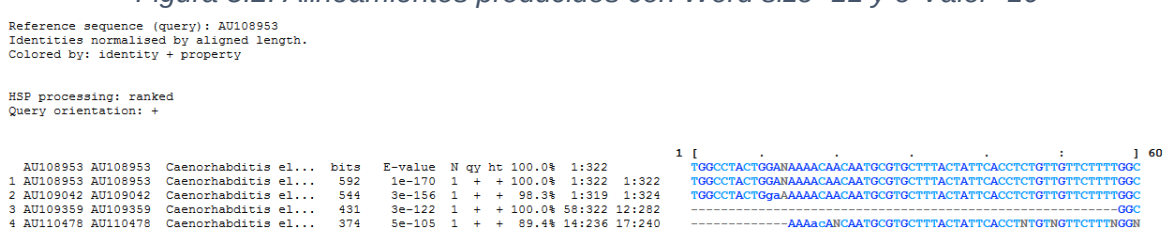



*Figura 5.3. Alineamientos producidos con Word size=30 y e-Valor=10*

Los valores obtenidos en los experimentos con BLASTN son los que se muestran en la Tabla 5.1. Tal y como reflejan la Figura 5.4 y la Figura 5.5, a medida que aumenta el tamaño de palabra, para un valor constante del parámetro e-Valor, disminuye el tiempo de ejecución, pero también, el número de alineamientos conseguidos al realizarse una búsqueda con menor precisión, es decir, la búsqueda es menos sensible pero más rápida.

| Tamaño de palabra (Word size) | E-Valor | Alineamientos producidos | Tiempo de ejecución (segundos) |
|---|---|---|---|
| 4 | 10 | 61 | 0,6 |
| 11 | 10 | 54 | 0,0475 |
| 30 | 10 | 4 | 0,024 |
| 4 | 1 | 12 | 0,517 |
| 11 | 1 | 12 | 0,028 |
| 30 | 1 | 4 | 0,024 |
| 4 | 0,01 | 4 | 0,513 |
| 11 | 0,01 | 4 | 0,026 |
| 30 | 0,01 | 4 | 0,0245 |

*Tabla 5.1. Influyencia de Word size y e-Valor en los análisis con BLASTN*

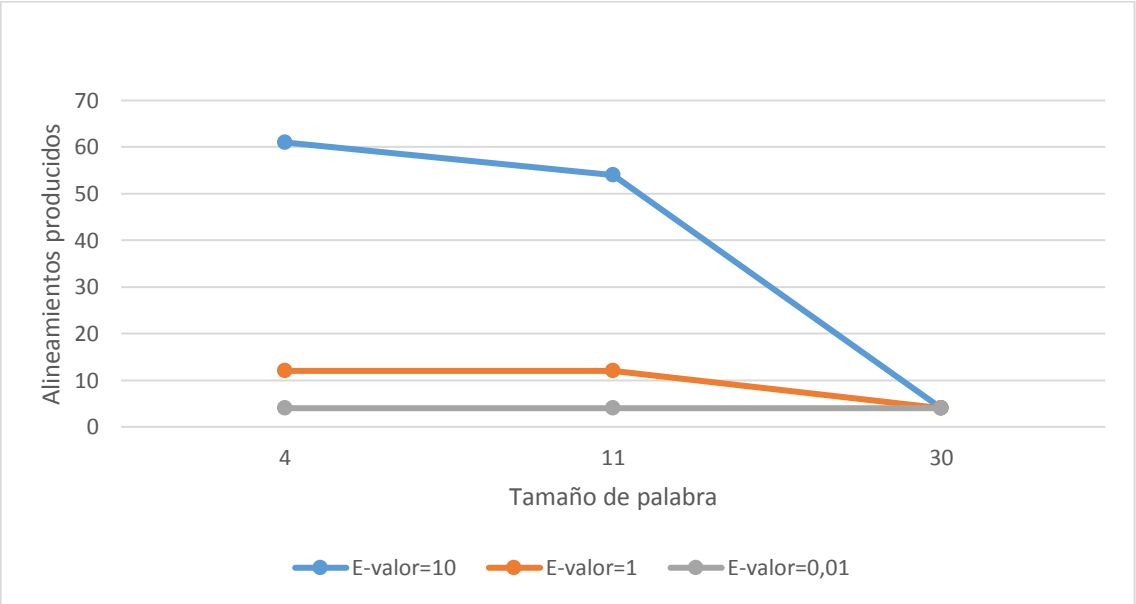



*Figura 5.4. Alineamientos producidos en función del tamaño de palabra*

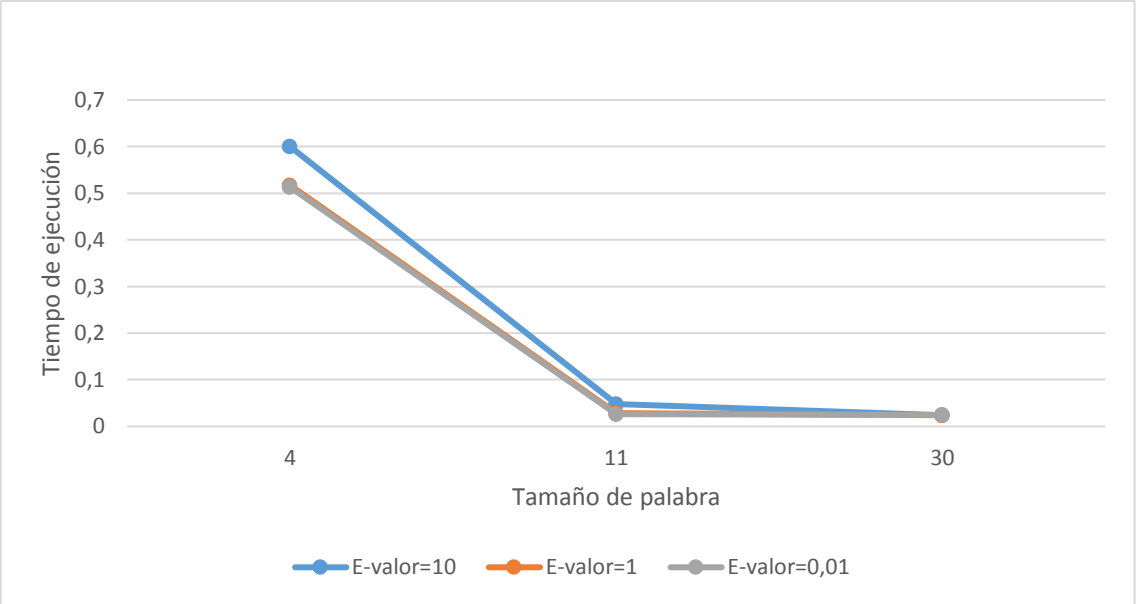



*Figura 5.5. Tiempo de ejecución en función del tamaño de palabra*

Por otro lado, si lo que permanece constante es el tamaño de palabra, conforme disminuye el parámetro E-valor, se reducen considerablemente tanto los alineamientos producidos como el tiempo de ejecución, reportándose sólo aquellos con mayor grado de significación estadística (Figura 5.6 y Figura 5.7).

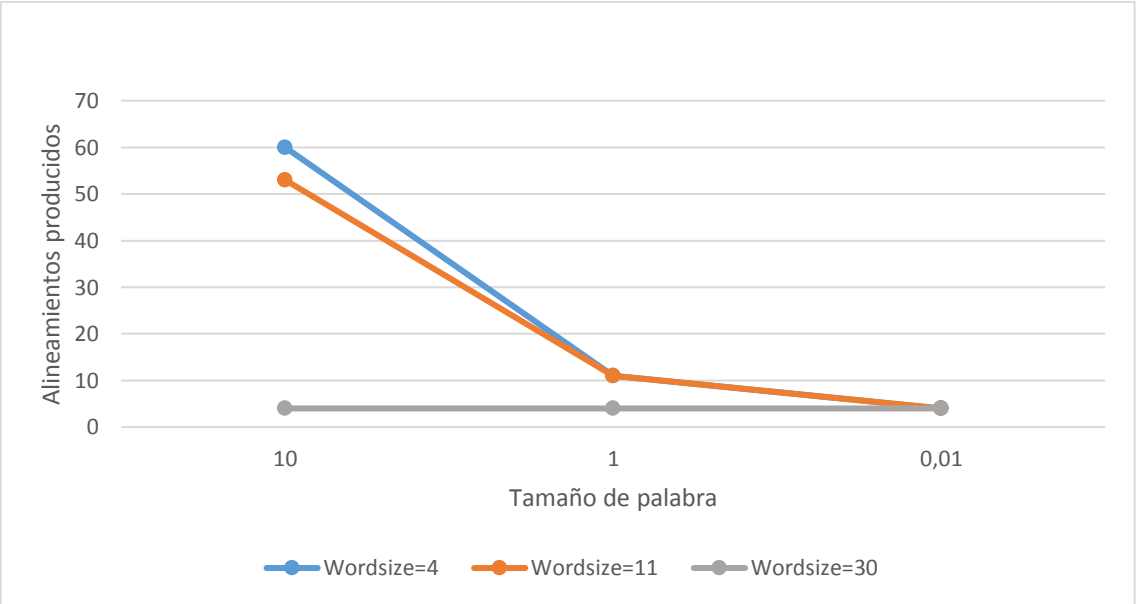



*Figura 5.6. Número de alineamientos producidos en función del parámetro E-valor*

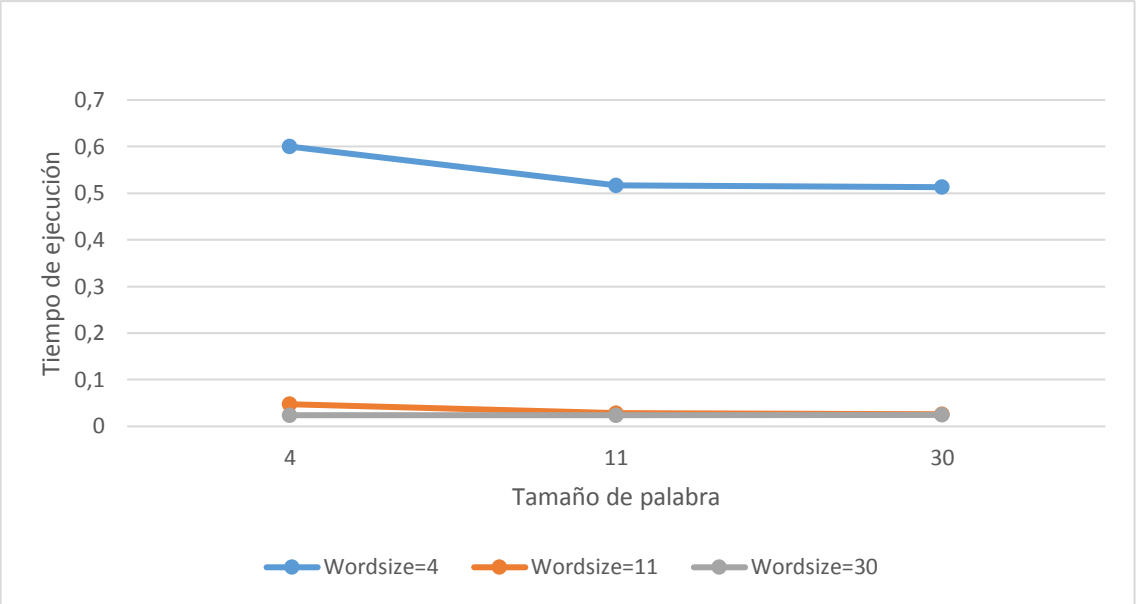



*Figura 5.7. Tiempo de ejecución respecto al parámetro E-valor*

Por tanto, es necesario prestar especial atención a los valores asignados a ambos parámetros, ya que un tamaño de palabra muy grande o un e-Valor excesivamente pequeño pueden reducir el número de alineamientos producidos, o incluso, que éstos no se produzcan.

#### *5.2.1.2 BLASTP*

El segundo programa empleado, BLASTP, efectúa una búsqueda de una secuencia problema de proteínas en una base de datos también de proteínas. En este caso, se ha estudiado el valor del parámetro Threshold y el tamaño de palabra. Los resultados obtenidos se recogen en la Tabla 5.2.

El parámetro Threshold determina el tamaño de la vecindad, de tal modo que cuanto mayor es su valor, la vecindad es más pequeña y por lo tanto, menor es el tiempo de ejecución (Figura 5.8).

Por otro lado, se observa que ninguno de los dos parámetros ha tenido un efecto sobre el número de alineamientos producidos.

| Tamaño de palabra (Word size) | Threshold | Alineamientos | Tiempo de ejecución (Segundos) |
|---|---|---|---|
| **3** | 11 | 250 | 1,313 |
| **3** | 200 | 250 | 0,995 |
| **3** | 500 | 250 | 0,99 |
| **3** | 999 | 250 | 1,017 |
| **5** | 11 | 250 | 3,379 |
| **5** | 200 | 250 | 1,28 |
| **5** | 500 | 250 | 1,211 |
| **5** | 999 | 250 | 1,209 |
| **7** | 11 | 250 | 1,955 |
| **7** | 200 | 250 | 0,726 |
| **7** | 500 | 250 | 0,729 |
| **7** | 999 | 250 | 0,721 |

*Tabla 5.2. Efecto de los parámetros Word size y Threshold con BLASTP*

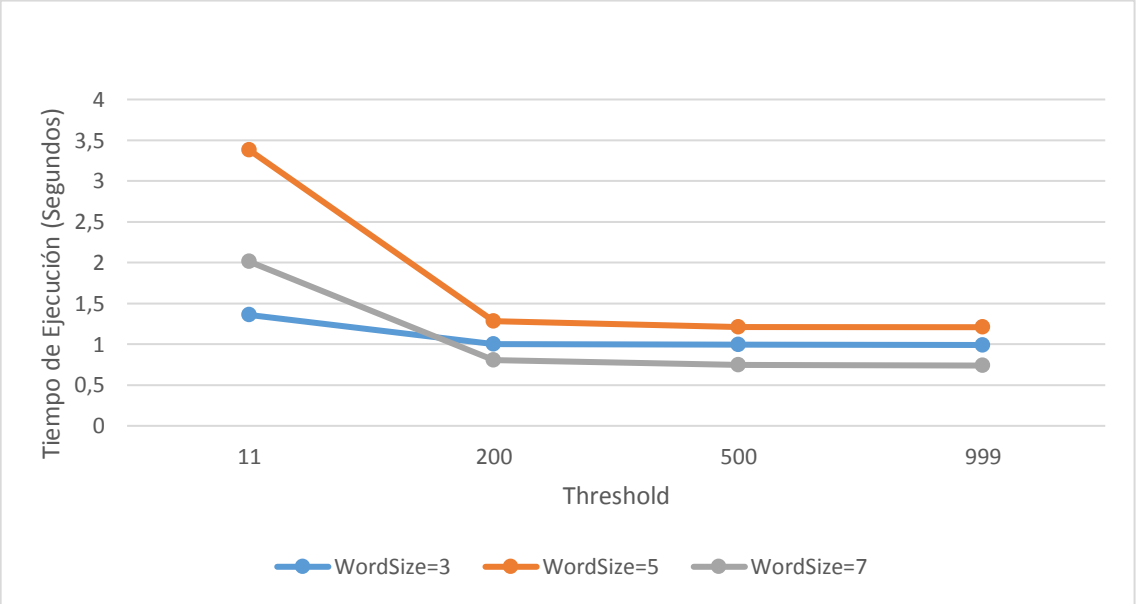



*Figura 5.8. Tiempos de Ejecución respecto al valor del parámetro Threshold*

En este caso, y dado que en todos los análisis llevados a cabo se han producido 250 análisis, variando únicamente el tiempo requerido para conseguirlos, sólo se mostrará una representación gráfica de estos alineamientos (Figura 5.9).

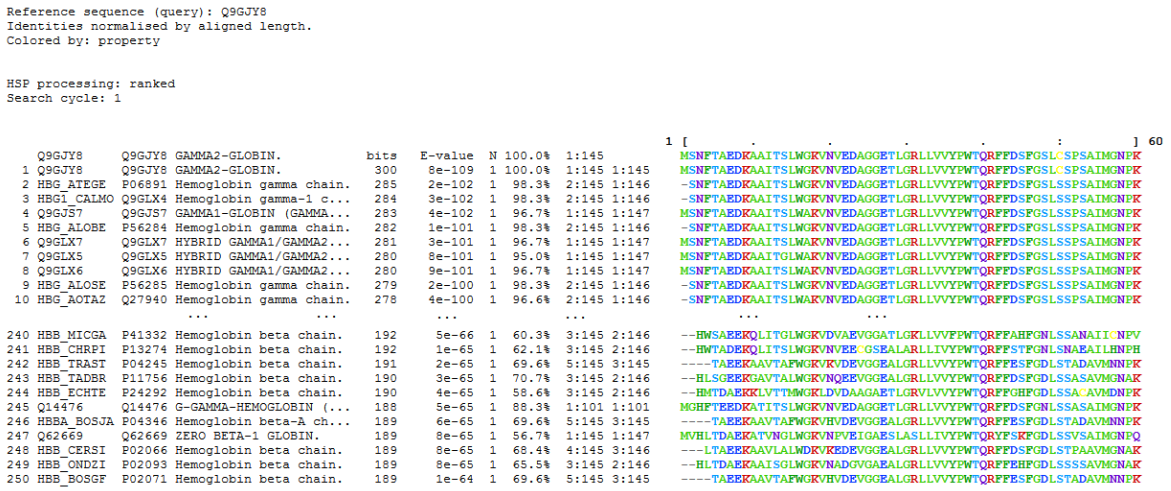



*Figura 5.9. Representación gráfica de secuencias de aminoácidos*

### 5.2.2 *Tiempos de ejecución de Trinity*

Se ha realizado un estudio sobre los tiempos de ejecución en un nodo necesarios cuando se realiza un análisis con el programa bioinformático Trinity utilizando un DRMS y sin emplearlo.

En el primer caso, es decir, haciendo uso de un sistema gestor, la infraestructura cloud gestionada por el planificador grid estaba conformada por 4 máquinas virtuales con 4 vCPU y 4 GB de memoria RAM cada una y las tareas han sido asignadas de forma individual o en lotes de 10 si las muestras son pequeñas o, en lotes de 50, si éstas son de mayor tamaño. En el otro caso, las ejecuciones se han llevado a cabo en una de estas máquinas virtuales. En ambos, se han realizado 10 repeticiones, siendo el resultado final la media aritmética de los valores obtenidos.

Además, se ha medido el tiempo de ejecución necesario para analizar las muestras de mayor tamaño cuando la infraestructura grid estaba conformada por 3 máquinas virtuales de 4 vCPU y 4 GB de memoria RAM y una con 8 vCPU y 8 GB de memoria RAM.

#### 5.2.2.1 *Muestras pequeñas*

Los tiempos de ejecución requeridos, tanto si se usa un DRMS como si no se hace uso del mismo, se recogen en la Tabla 5.3. Se comprueba que si se hace uso de una infraestructura cloud, el tiempo de ejecución necesario es menor que si no se utiliza (Figura 5.10).

| Ejecución | Con DRMS y tareas en lotes de 10 (mm:ss) | Sin DRMS (mm:ss) |
|---|---|---|
| **1** | 02:58 | 03:59 |
| **2** | 03:34 | 04:05 |
| **3** | 02:57 | 04:02 |
| **4** | 02:51 | 03:55 |
| **5** | 02:27 | 03:59 |
| **6** | 02:47 | 04:25 |
| **7** | 02:48 | 04:14 |
| **8** | 02:46 | 04:05 |
| **9** | 03:46 | 04:18 |
| **10** | 03:28 | 03:55 |
| | **02:54** | **04:03** |

*Tabla 5.3. Tiempos de ejecución con muestras de pequeño tamaño*

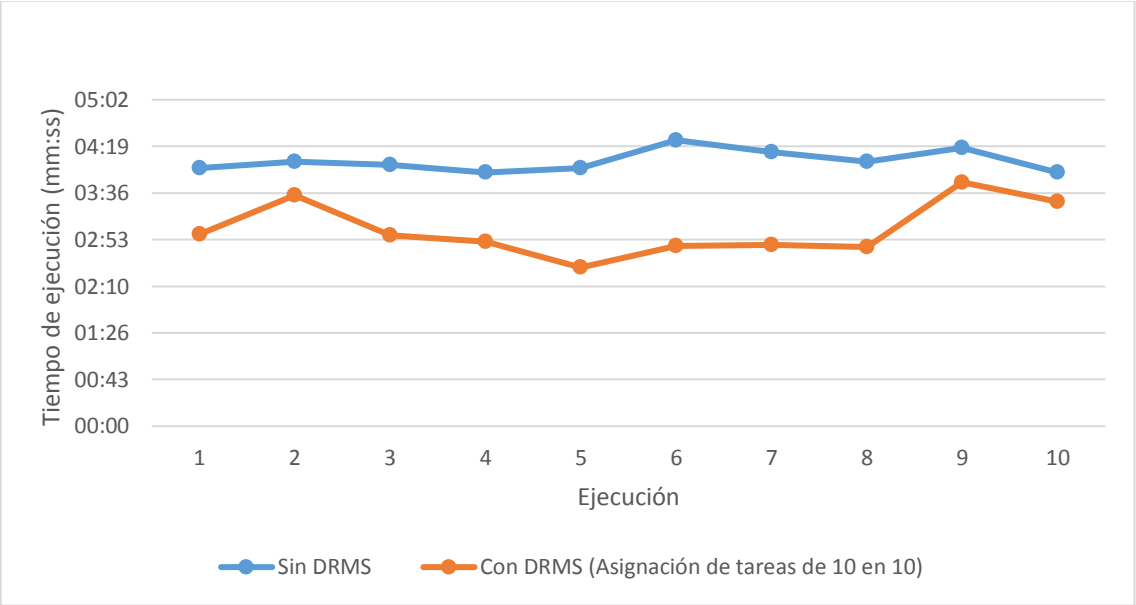



*Figura 5.10. Tiempos de ejecución con muestras de pequeño tamaño*

Otro factor que también debe tenerse en cuenta, es la manera en la que son asignadas las tareas a los recursos computacionales es decir, si se hace por lotes o de una en una. En la Tabla 5.4 y la Figura 5.11 se puede observar que, para muestras de pequeño tamaño, si las tareas son asignadas en lotes de 10 se consigue un ahorro de 1 minuto y 19 segundos, lo que supone un 31,22% del tiempo total requerido si las tareas se asignasen individualmente. Este tiempo ahorrado se corresponde con el *overhead* incurrido durante el encolado de las tareas y la asignación de las mismas a los recursos computacionales.

| Ejecución | Tareas asignadas en lotes de 10 (mm:ss) | Tareas asignadas individualmente (mm:ss) | Diferencia (mm:ss) |
|---|---|---|---|
| **1** | 02:58 | 04:13 | 01:15 |
| **2** | 03:34 | 04:01 | 00:27 |
| **3** | 02:57 | 03:48 | 00:51 |
| **4** | 02:51 | 04:22 | 01:31 |
| **5** | 02:27 | 04:38 | 02:11 |
| **6** | 02:47 | 03:55 | 01:08 |
| **7** | 02:48 | 04:45 | 01:57 |
| **8** | 02:46 | 04:14 | 01:28 |
| **9** | 03:46 | 04:45 | 00:59 |
| **10** | 03:28 | 03:44 | 00:16 |
| | **02:54** | **04:13** | **01:19** |

*Tabla 5.4. Tiempos de ejecución para tareas asignadas individualmente y por lotes usando 4 máquinas virtuales iguales*

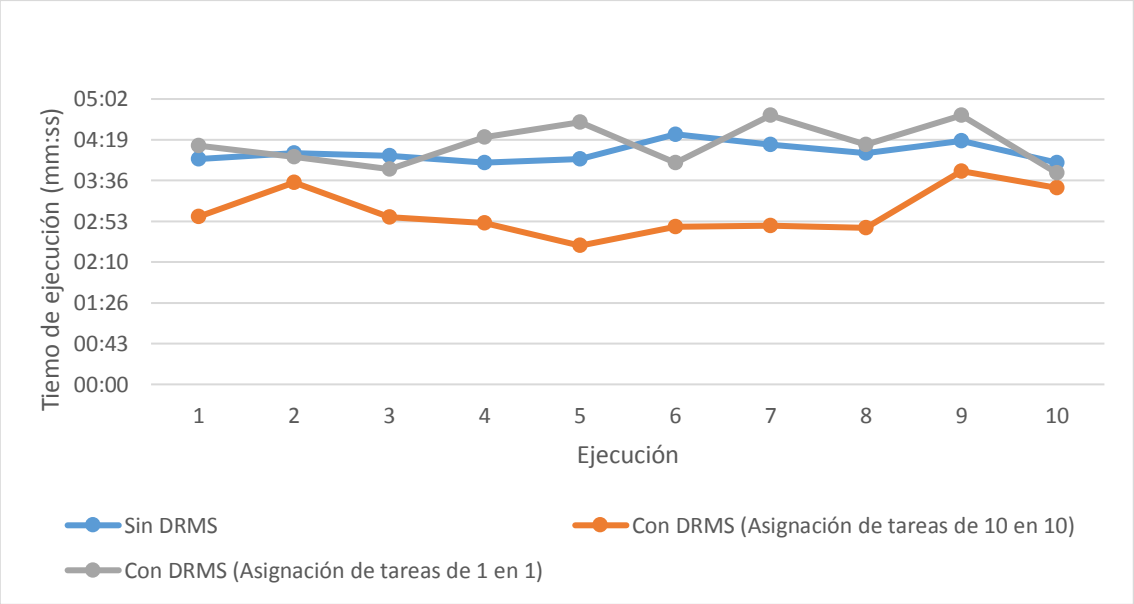



*Figura 5.11. Tiempos de ejecución para tareas asignadas individualmente y por lotes*

Finalmente, se determinó el tiempo que ejecución requerido para analizar las muestras pequeñas en una infraestructura compuesta por 3 máquinas con 4vCPU y 4 GB de memoria RAM y una cuarta dotada con 8vCPU y 8GB de RAM. Tal y como se puede observar en la Tabla 5.5 y la Figura 5.15, no existen diferencias significativas entre los tiempos obtenidos en ambos entornos.

| Ejecución | Entorno Homogéneo | Entorno Heterogéneo |
|---|---|---|
| **1** | 02:58 | 03:33 |
| **2** | 03:34 | 02:36 |
| **3** | 02:57 | 02:39 |
| **4** | 02:51 | 03:20 |
| **5** | 02:27 | 03:06 |
| **6** | 02:47 | 02:34 |
| **7** | 02:48 | 02:59 |
| **8** | 02:46 | 02:28 |
| **9** | 03:46 | 02:44 |
| **10** | 03:28 | 02:47 |
| | **03:02** | **02:53** |

*Tabla 5.5. Tiempos de ejecución en un entorno homogéneo y uno heterogéneo*

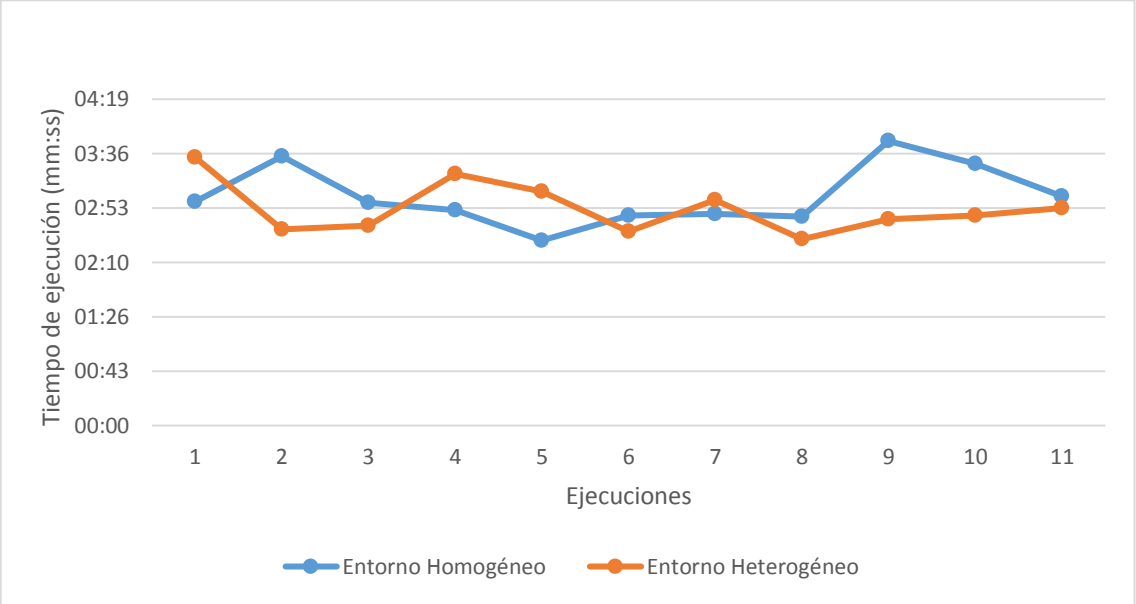



*Figura 5.12. Tiempos de ejecución en un entorno homogéneo y uno heterogéneo*

#### *5.2.2.2 Muestras grandes*

En primer lugar, se ha analizado la relación entre el tiempo de ejecución empleado para realizar un análisis de las muestras de mayor tamaño respecto al número de recursos computaciones activos. Tal y como puede observarse en la Tabla 5.6 y la Figura 5.13, a medida que aumenta el número de máquinas, el tiempo decrece de forma exponencial. No obstante, si el número de máquinas virtuales es superior a 4 se observa que el tiempo de ejecución vuelve a crecer. Ello es debido conforme aumenta el número de recursos

computacionales, se produce un incremento del tiempo de transferencia de datos a los distintos recursos computacionales, pudiendo incluso superar este tiempo al tiempo de ejecución de la tarea. Por esta razón, es muy importante adaptar la infraestructura grid a las tareas que vayan a ejecutar, para minimizar lo máximo posible los tiempos de ejecución.

| Número de máquinas virtuales activas | Tiempo de ejecución (hh:mm:ss) |
|---|---|
| **1** | 01:48:49 |
| **2** | 01:15:32 |
| **3** | 01:04:42 |
| **4** | 00:53:26 |
| **5** | 00:56:02 |
| **6** | 00:57:50 |
| **7** | 00:59:43 |

*Tabla 5.6. Tiempo de ejecución en función del número de recursos computacionales*

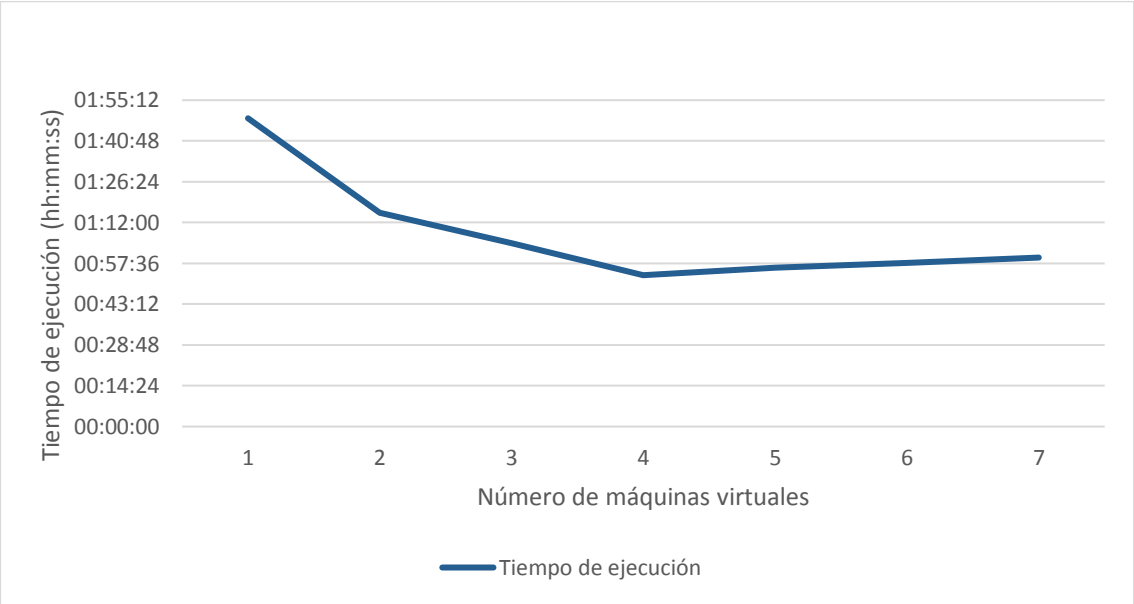



*Figura 5.13. Tiempo de ejecución en función del número de recursos computacionales activos*

Por otro lado, cuando las tareas son de un tamaño más considerable, sí se aprecian diferencias algo más significativas si se hace uso de un sistema gestor o si no se utiliza, tal y como recogen la Tabla 5.7 y la Figura 5.14.

| Ejecución | Con DRMS y tareas en lotes de 50 (mm:ss) | Sin DRMS (hh:mm:ss) | Diferencia (hh:mm:ss) |
|---|---|---|---|
| **1** | 00:54:56 | 02:07:42 | 01:14:16 |
| **2** | 00:53:34 | 02:10:18 | 01:16:44 |
| **3** | 00:52:54 | 02:12:21 | 01:19:27 |
| **4** | 00:50:33 | 02:02:34 | 01:12:01 |
| **5** | 00:52:18 | 02:10:27 | 01:18:09 |
| **6** | 00:53:38 | 02:17:08 | 01:23:30 |
| **7** | 00:54:52 | 02:16:48 | 01:21:56 |
| **8** | 00:52:40 | 02:25:41 | 01:33:01 |
| **9** | 00:51:04 | 02:26:28 | 01:35:24 |
| **10** | 00:52:43 | 02:20:01 | 01:27:18 |
| | **00:52:48** | **02:16:58** | **01:22:11** |

*Tabla 5.7. Tiempos de ejecución con muestras de mayor tamaño usando usando 4 máquinas virtuales iguales*

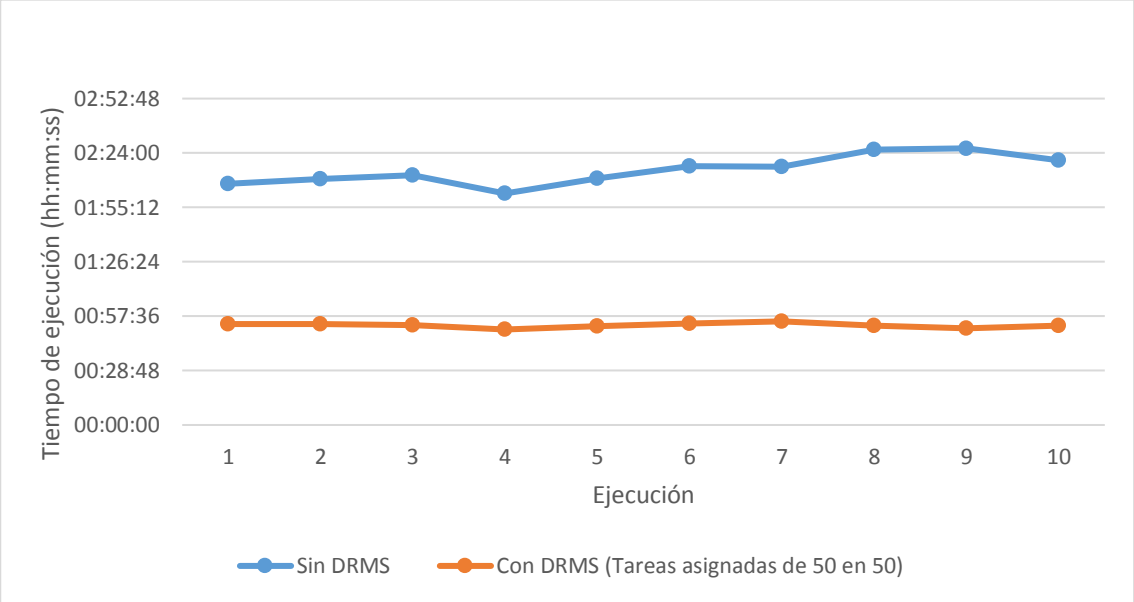



*Figura 5.14. Tiempos de ejecución con muestras de mayor tamaño*

Finalmente, se ha determinado el tiempo de ejecución necesario para analizar las muestras cuando una de las máquinas virtuales era diferencia a las demás. Tal y como recogen en la Tabla 5.8 y la Figura 5.15, los tiempos requeridos no presentan diferencias significativas.

| Ejecución | Entorno Homogéneo (mm:ss) | Entorno Heterogéneo (mm:ss) |
|---|---|---|
| **1** | 53:26 | 52:58 |
| **2** | 53:34 | 52:19 |
| **3** | 52:54 | 55:11 |
| **4** | 50:33 | 52:58 |
| **5** | 52:18 | 50:22 |
| **6** | 53:38 | 54:05 |
| **7** | 54:52 | 52:53 |
| **8** | 52:40 | 55:19 |
| **9** | 51:04 | 47:36 |
| **10** | 52:43 | 49:10 |
| | **52:46** | **52:17** |

*Tabla 5.8. Tiempos de ejecución en un entorno homogéneo frente a uno heterogéneo*

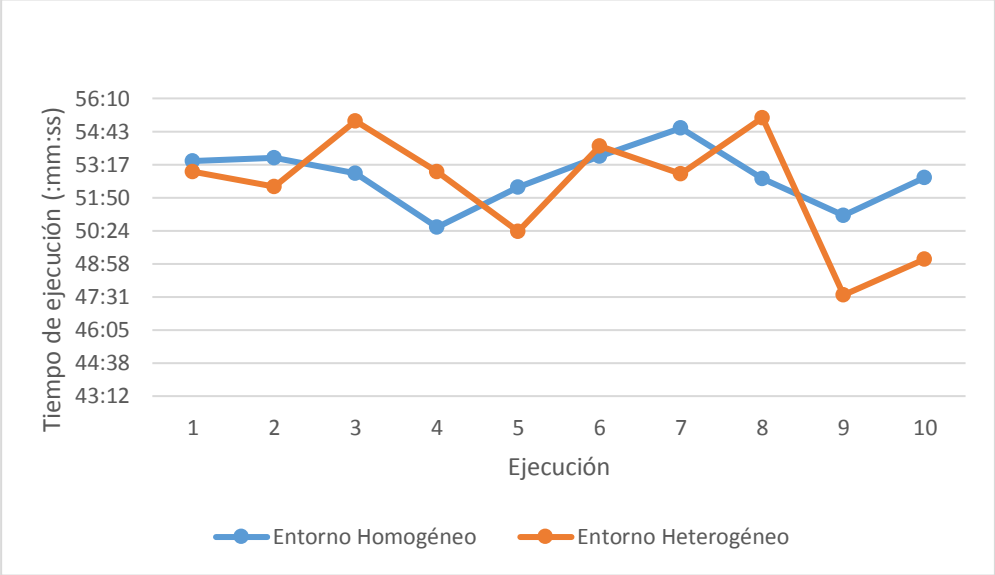



*Figura 5.15. Tiempos de ejecución en un entorno homogéneo y uno heterogéneo*

## 5.3 Prueba de carga

Para llevar a cabo esta prueba de carga se ha utilizado la herramienta Apache Bench. En dicha prueba, se han realizado un total de 10.000 peticiones, efectuándose 100 peticiones de forma recurrente. Como resultado de dicha prueba se ha obtenido el siguiente informe:

```
Server Software:        Apache-Coyote/1.1
Server Hostname:        localhost
Server Port:            8080

Document Path:          /biocloud/
Document Length:        3741 bytes

Concurrency Level:      100
Time taken for tests:   23.074 seconds
Complete requests:      10000
Failed requests:        0
Write errors:           0
Total transferred:      40180000 bytes
HTML transferred:       37410000 bytes
Requests per second:    433.39 [#/sec] (mean)
Time per request:       230.739 [ms] (mean)
Time per request:       2.307 [ms] (mean, across all concurrent requests)
Transfer rate:          1700.55 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   18  16.4      6     113
Processing:    14  211  92.4    209     525
Waiting:        1  201  93.5    198     513
Total:         15  230  90.1    233     542

Percentage of the requests served within a certain time (ms)
  50%  233
  66%  257
  75%  273
  80%  283
  90%  347
  95%  396
  98%  440
  99%  512
 100%  542 (longest request)
```

Lo más destacable de dicho informe es lo siguiente:

- ***Requests per second**:* Por cada segundo se atienden 433.39 peticiones de media.
- ***Time per request (mean)**:* El tiempo medio que el servidor ha tardado en atender cada bloque de 100 peticiones concurrentes es 230,739 ms.
- ***Time per request (mean, across all concurrent requests):*** El tiempo que el servidor ha tardado en atender una petición individual ha sido de 2,307 ms.

La representación gráfica de estos resultados se recoge en la Figura 5.16. En dicha figura se observa cómo a medida que se efectúan las peticiones, el tiempo promedio de respuesta del servidor se va incrementando paulatinamente.

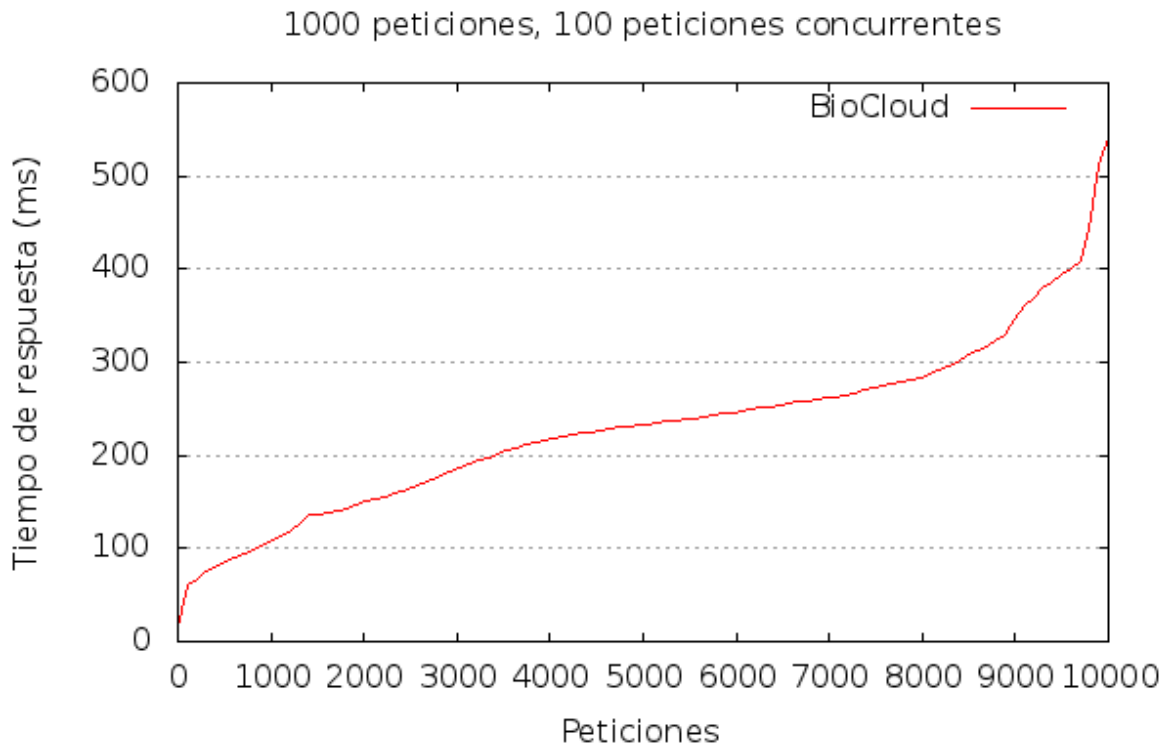



*Figura 5.16. Resultados de la prueba de carga*

## 5.4 Conclusiones

En este capítulo se han reseñado los resultados obtenidos en las diferentes pruebas llevadas a cabo sobre la infraestructura cloud.

Por un lado, se ha demostrado que hay que prestar especial atención a los valores asignados a los diferentes parámetros específicos de BLAST ya que pueden afectar significativamente tanto en el número como en la precisión de los alineamientos producidos.

Por otro lado, se ha comprobado que la ejecución distribuida de tareas complejas reduce considerablemente su tiempo de ejecución. Además, se ha demostrado, que es necesario configurar adecuadamente el sistema gestor para evitar que una mala configuración pueda ocasionar una ralentización del sistema.

Finalmente, se ha analizado el comportamiento del sistema frente a situaciones de gran carga en las que existe número considerable de conexiones simultáneas. Con estas pruebas se pueden detectar cuellos de botella en la aplicación.

# 6 CONCLUSIONES

Uno de los aspectos más llamativos de este trabajo reside en la combinación de disciplinas que abarcan: por un lado, la utilización de conceptos y técnicas tales como la secuenciación y el alineamiento de muestras biológicas (proteínas, nucleótidos, etc), más propios del ámbito de la Biología; por otro lado, el empleo de algoritmos computacionalmente eficientes que trasladan los conceptos teóricos a la práctica, así como el desarrollo de una aplicación web para simplificar la realización de los experimentos asociados. A tal fin, ha sido necesario adquirir multitud de conocimientos de ambos mundos para la consecución de los objetivos planteados.

En este trabajo se han explorado dos herramientas bioinformáticas relativamente novedosas en el ámbito de la Biología Molecular: Trinity, orientada a la secuenciación *de novo* de muestras, y BLAST, cuya finalidad es el alineamiento de secuencias. Si bien la autora cuenta con amplios conocimientos en dicho campo al ser Licenciada en Biología, la asimilación y el aprendizaje de estas herramientas le ha requerido un esfuerzo considerable. Esto es debido principalmente a que sus respectivas implementaciones se sustentan sobre algoritmos y conceptos particularmente complejos. Adicionalmente, tanto Trinity como BLAST ofrecen soporte limitado para sistemas operativos ajenos a Unix / Linux, por lo que ha sido necesario adquirir previamente competencias básicas en la administración y el uso de sistemas basados en Linux, concretamente RHEL, con el fin de que la instalación de estas herramientas, así como la del resto de dependencias requeridas, pudiera llevarse a buen puerto.

Entre los objetivos que se perseguían con la consecución de este proyecto, se identificó el ofrecer una plataforma para simplificar la ejecución de experimentos de secuenciación y alineamiento. Para satisfacer este requisito, se ha diseñado una interfaz de usuario minimalista e intuitiva que logra abstraer a los usuarios finales de los distintos comandos Linux implicados en la experimentación. Las funcionalidades que ofrece la interfaz Web, tal y como se recoge en la definición de casos de uso, contemplan la creación de bases de datos de secuencias propias y la ejecución de experimentos mediante las herramientas BLAST y Trinity, atendiendo a los parámetros introducidos por el usuario en sendos formularios Web.

Otro de los puntos fuertes de este trabajo reside en el diseño y la implementación de una infraestructura Grid virtualizada con la que distribuir adecuadamente las tareas a realizar en el marco de un experimento concreto utilizando la herramienta Trinity, de manera que se puedan reducir al máximo los tiempos de ejecución. Para poder realizar esta infraestructura, la autora ha necesitado reforzar sus conocimientos de Computación

Grid, así como adquirir competencias nuevas en materia de virtualización, planificación y asignación de tareas. Esto se ve reflejado en la inclusión de componentes software como OpenNebula, OGS/GE y la API DRMAA para la ejecución de tareas en sistemas distribuidos. Realmente, este punto y el anterior constituyen los principales valores del proyecto.

Una parte considerable del tiempo de desarrollo se ha destinado a la construcción de la interfaz Web para captación de información y ejecución de experimentos. Dado que se trata de una pieza de software de tamaño y complejidad considerables, se ha intentado preservar los parámetros de mantenibilidad, flexibilidad y modularidad del código en unos márgenes aceptables.

Para lograr lo expuesto en el apartado anterior, el diseño de la aplicación Web se ha basado en el empleo de patrones de diseño, soluciones recurrentes a problemas frecuentes y que han sido probadas en la práctica en numerosas ocasiones. Todos los patrones de diseño empleados se han recopilado en un Catálogo (Apartado 9.12), que actúa a modo de referencia y ayuda a homogeneizar el lenguaje técnico. A pesar de lo anterior, ha sido la decisión de incluir el framework Spring en el diseño lo que en última instancia ha facilitado la aparición de esos patrones en el código de la aplicación.

En cuanto a la complejidad de cada una de las aplicaciones instaladas o tareas de configuración realizadas a lo largo de todo el proceso de implementación, sin duda, lo que más esfuerzo y trabajo ha supuesto a la autora es la instalación y configuración del sistema gestor OGS/GE y su integración con la plataforma OpenNebula. Ello es debido, en el primer caso, a que la documentación disponible en Internet sobre este sistema gestor es bastante escasa y dispersa, y, en el segundo caso, no se ha encontrado ningún trabajo previo en el que ambas aplicaciones funcionen conjuntamente y que pudiera servir como punto de partida para este TFG. Respecto al resto de programas, la dificultad de éstos ha sido moderada.

Por último, y dejando a un lado el aspecto más puramente de implementación, se ha perseguido la elaboración de una Memoria lo más completa y fiel posible en relación a los trabajos realizados. Con el fin de facilitar la comprensión y agilizar la presentación de los conceptos clave, se han proporcionado una serie de diagramas UML y artefactos comunes a todos los proyectos software (definición de casos de uso, colección de requisitos funcionales y no funcionales, etc); esto ha supuesto una complejidad adicional, dado que al inicio de los trabajos la autora no disponía de estos conocimientos.

# 7 LÍNEAS FUTURAS

A pesar de que la infraestructura implementada cumple con los requisitos establecidos al inicio del trabajo, se han detectado varias líneas de trabajo que se podrían desarrollar en un futuro:

- Implementar funcionalidades adicionales a través del uso de nuevas herramientas, permitiendo hacer en un futuro estudios cada vez más completos y más complejos que incluyan análisis estadísticos o representaciones gráficas de los resultados obtenidos. En relación a ello, se añadiría una vista más, a través de la cual, los usuarios podrían solicitar estas nuevas funcionalidades.
- Actualmente, el mecanismo por el que los usuarios pueden comprobar el estado de sus tareas es acceder a la vista "*Listado de Tareas*". Se podría implementar un sistema de notificaciones que generase un aviso tan pronto como haya finalizado alguna de las tareas.
- Programación de la aplicación para entornos móviles como Android, iOS o Windows Phone, para aumentar la versatilidad de la aplicación.
- Implementación del paginado de resultados con SQLite para agilizar la obtención de información desde la base de datos. De este modo, todos los datos se organizarán en páginas.
- Desplegar en un futuro la infraestructura grid de manera virtual, es decir, usando hosting cloud, de tal manera que todas las máquinas virtuales estén en la nube y no sea necesario hacer uso de ningún equipo físico como nodo anfitrión.

# 8 BIBLIOGRAFÍA


**[1].** **THE GALAXY PROJECT.** *The Galaxy Project* [en línea]. Disponible en: https://galaxyproject.org/ [Consulta: 17-08-2016]

**[2].** **UCSC.** *UCSC Genome Browser* [en línea]. Disponible en: https://genome.ucsc.edu/ [Consulta: 17-08-2016]

**[3].** **SPRING.** *Introduction to the Spring Framework* [en línea]. Disponible en: http://docs.spring.io/spring/docs/current/spring-framework-reference/html/overview.html [Consulta: 12-02-2016]

**[4].** **BOOTSTRAP.** *Bootstrap.The world's most popular mobile-first and responsive front-end framework* [en línea]. Disponible en: http://getbootstrap.com/ [Consulta: 12-08-2016]

**[5].** **APACHE TOMCAT.** *Apache Tomcat 7. Documentation Index* [en línea]. Disponible en: https://tomcat.apache.org/tomcat-7.0-doc/index.html [Consulta: 17-08-2016]

**[6].** **ECLIPSE.** *Eclipse Neon* [en línea]. Disponible en: https://eclipse.org/ [Consulta**:** 15-10-2015]

**[7].** **SCIENTIFIC LINUX.** *Scientific Linux* [en línea]. Disponible en: https://www.scientificlinux.org/ [Consulta: 18-08-2016]

**[8].** **DRMAA WORKING GROUP.** *OGF DRMAA Working Group* [en línea]. Disponible en: https://www.drmaa.org/ [Consulta: 17-08-2016]

**[9].** **SPRING.** *Scheduling Tasks* [en línea]. Disponible en: https://spring.io/guides/gs/scheduling-tasks/ [Consulta: 21-08-2016]

**[10].** **WORMBASE.** *Explore Worm Biology* [en línea]. Disponible en: http://www.wormbase.org [Consulta: 18-08-2016]

**[11].** **BROWN, N., LEROY, C. & SANDER, C**. MView: A Web compatible database search or multiple alignment viewer. *Bioinformatics*, 1998, **14**(4): 380-381

**[12].** **COHEDERO**. *Cómo Hacer Pruebas de Carga a Servidores Web* [en línea]. Disponible en: http://codehero.co/como-hacer-pruebas-de-carga-servidores-web/ [Consulta: 12-09-2016]

**[13].** **APACHE**. *ab – Apache HTTP server benchmarking tool* [en línea]. Disponible en: http://httpd.apache.org/docs/2.0/programs/ab.html [Consulta: 12-09-2016].

**[14].** **QEMU**. *QEMU main page* [en línea]. Disponible en: http://wiki.qemu.org/Main_Page [Consulta: 03-04-2016]

**[15].** **TORALDO G**. *Opennebula 3 Cloud Computing*. U.K.: Packt Publishing. 2012. 978-18-495-1746-1

**[16].** **NCBI.** *Standalone BLAST Setup for Unix: NCBI* [en línea], 31-05-2010. Disponible en: http://www.ncbi.nlm.nih.gov/books/NBK52640/ [Consulta: 06-03-2016]

**[17].** **TRINITY.** *Installing Trinity* [en línea], 24-09-2015. Disponible en: https://github.com/trinityrnaseq/trinityrnaseq/wiki/Installing%20Trinity [Consulta: 06-03-2016]

**[18].** **BOWTIE.** *Bowtie. An ultrafast memory-efficient short read aligner* [en línea]. Disponible en: bowtie-bio.sourceforge.net/index.shtml [Consulta: 17-11-2015]

**[19].** **OPENNEBULA.** *Quickstart: OpenNebula on CentOS 7 and KVM* [en línea]. Disponible en: http://docs.opennebula.org/4.12/design_and_installation/quick_starts/qs_centos7_kvm.html#qs-centos7-kvm [Consulta: 22-03-2016]

**[20].** **MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE.** *NFS: Sistema de archivos de red* [en línea], 25-08-2009. Disponible en: http://recursostic.educacion.es/observatorio/web/gl/software/software-general/733-nfs-sistema-de-archivos-de-red [Consulta: 30-06-2016]

**[21].** **SON OF GRID ENGINE SOGE (BIRD/NAF2)**. *Installing Guide* [en línea]. Disponible en: DOI: 73957906-054733-44 [Consulta: 16-03-2016]

**[22].** **DIGITALOCEAN.** *How To Install Apache Tomcat 7 on CentOS 7 via Yum* [en línea], 15-06-2015. Disponible en: https://www.digitalocean.com/community/tutorials/how-to-install-apache-tomcat-7-on-centos-7-via-yum [Consulta: 06-03-2016]

**[23].** **NCBI - PUBMED.** *PubMed* [en línea]. Disponible en: http://www.ncbi.nlm.nih.gov/pubmed [Consulta: 18-08-2016]

**[24].** **STRYER, L., BERG, J. M. & TYMOCZKO, J. L.** *Bioquímica.* Barcelona: REVERTÉ, S.A, 2003. 978-84-291-7584-9

**[25].** **VOET, D., PRATT, C. W. & VOET, J. G.** *Fundamientos de Bioquímica. La vida a nivel molecular.* Buenos Aires: Médica Panamericana, 2007. 978-950-06-2314-8

**[26].** **KORF, I., YANDELL, M. & BEDELL, J.** *BLAST.* O'Reilly, 2003. 978-1-4493-8396-1

**[27].** **QUIMICO.** *Químico: Deriva Genética* [en línea], 12-2009. Disponible en: http://cienciasenbachillerato.blogspot.com.es/2009/12/deriva-genetica.html [Consulta: 15-10-2016]

**[28].** **BLANCO GARCÍA, E.** *Genómica computacional.* Barcelona: Editorial UOC, 2014. 978-84-9029-910-4

**[29].** **NCBI.** *BLAST topics* [en línea]. Disponible en: https://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=BlastHelp [Consulta: 25-09-2016]

**[30].** **NCBI.** *Options for the command-line applications* [en línea]. Disponible en: http://www.ncbi.nlm.nih.gov/books/NBK279675/ [Consulta: 18-08-2016]

**[31].** **RABINOVICH, G. A.** *Inmunopatología Molecular: Nuevas fronteras de la medicina. Un nexo entre la investigación biomédica y la práctica clínica.* Madrid: PANAMERICANA, 2004. 978-84-7903-683-6

**[32].** **GRABHERR, MANFRED G. et al**. Full-length transcriptome assembly from RNA-Seq data without a reference genome*. Nature Biotechnology*, 2011, **7**(29): 644-652

**[33].** **HAAS, B.J., PAPANICOLAOU, A. *et al***. De novo transcript sequence reconstruction from RNA-seq using the Trinity platform for reference generation and analysis. *Nature Protocols*, 2013, **10**(8): 1-21

**[34].** **TRINITY.** *Wiki de Trinity* [en línea]. Disponible en: https://github.com/trinityrnaseq/trinityrnaseq/wiki [Consulta: 13-07-2016]

**[35].** **GENETAQ.** *BIOINFORMATICA PARA NO INICIADOS: CAPITULO I* [en línea]. Disponible en: http://genetaq.com/es/blog/bioinformatica-para-no-iniciados-capitulo-i [Consulta: 30-09-2016]

**[36].** **MAQUEIRA MARÍN, J.M. & BRUQUE CÁMARA, S.** *Las Tecnologías Grid de la Información como Nueva Herramienta Empresarial.* Oviedo: Septem Ediciones, 2012. 978-84-9649-190-8

**[37].** **FOSTER, I., KESSELMAN, C. & TUECKE S.** The Anatomy of the Grid. Enabling Scalable Virtual Organizations*. International Journal of High Performance Computing Applications*, 2001, **3**(15): 200-222

**[38].** **TEXTOS CIENTÍFICOS.** *Arquitectura Grid* [en línea], 23-03-2007. Disponible en: http://www.textoscientificos.com/redes/computacion-grid/arquitectura [Consulta: 28-07-2016]

**[39].** **SÁNCHEZ SANTIAGO, A**. *Planificación local de tareas con sistemas basados en reglas borrosas. Aplicación a la detección de defectos en tableros de fibra*. Directores: Sebastián García Galán y José Enrique Muñoz Expósito. Tesis doctoral, Universidad de Jaén. Departamento de Ingeniería de Telecomunicación, 2012

**[40].** **OPEN GRID SCHEDULER.** *Open Grid Scheduler* [en línea]. Disponible en: http://gridscheduler.sourceforge.net/ [Consulta: 23-04-2016]

**[41].** **SOFTPANOMARA.** *SGE Parallel Environment* [en línea]. Disponible en: http://www.softpanorama.org/HPC/Grid_engine/parallel_environment.shtml [Consulta: 07-10-2016]

**[42].** **DIE.NET.** *ksge_pe(5) - Linux man page* [en línea]. Disponible en: https://linux.die.net/man/5/sge_pe [Consulta: 07-10-2016]

**[43].** **ORACLE.** *Configuring a New Parallel Environment* [en línea]. Disponible en: https://blogs.oracle.com/templedf/entry/configuring_a_new_parallel_environment [Consulta: 05-10-2016]

**[44].** **OPENNEBULA PROJECT.** 2015. *OpenNebula 4.12 Design and Installation Guide [en línea]*. Disponible en: http://docs.opennebula.org/pdf/4.12/opennebula_4.12_design_and_installation_guide.pdf [Consulta: 14-07-2016]

**[45].** **OBSERVATORIO NACIONAL DEL SOFTWARE DE FUENTES ABIERTAS (CENATIC).** *Introducción a la Nube – OpenNebula como caso de éxito* [en línea], 17-09-2013. Disponible en: http://observatorio.cenatic.es/index.php?option=com_content&view=article&id=820:introduccion-a-la-nube-open-nebula-como-caso-de-exito&catid=108:blog-cenatic&Itemid=150 [Consulta: 03-07-2016]

**[46].** **OPENNEBULA.** *Managing Images* [en línea]. Disponible en: http://docs.opennebula.org/4.12/user/virtual_resource_management/img_guide.html [Consulta: 03-07-2016]

**[47].** **MVIEW.** *Introduction* [en línea]. Disponible en: https://desmid.github.io/mview/manual/intro.html [Consulta: 11-10-2016]

**[48].** **MVIEW**. *Manual* [en línea]. Disponible en: https://desmid.github.io/mview/manual/manual.html [Consulta: 11-10-2016]

**[49].** **TICBEAT.** *¿Qué es 'cloud computing'? Definición y concepto para neófitos* [en línea]. Disponible en: http://www.ticbeat.com/cloud/que-es-cloud-computing-definicion-concepto-para-neofitos/ [Consulta: 23-09-2016]

**[50].** **AMAZON**. *Amazon EC2 – Hospedaje de servidores virtuales* [en línea]. Disponible en: https://aws.amazon.com/es/ec2/ [Consulta: 22-09-2016]

**[51].** **AMAZON.** *AMAZON WEB SERVICES SIMPLE MONTHLY CALCULATOR* [en línea]. Disponible en: http://calculator.s3.amazonaws.com/index.html [Consulta: 01-10-2016]

**[52].** **AMAZON.** *Tipos de instancias de Amazon EC2* [en línea]. Disponible en: https://aws.amazon.com/es/ec2/instance-types/ [Consulta: 01-10-2016]

**[53].** **MICROSOFT**. *Microsoft Azure: plataforma y servicios de informática en la nube* [en línea]. Disponible en: https://azure.microsoft.com/es-es/ [Consulta: 01-10-2016]

**[54].** **MICROSOFT**. *Azure Marketplace* [en línea]. Disponible en: https://azure.microsoft.com/es-es/marketplace/ [Consulta: 01-10-2016]

**[55].** **MICROSOFT**. *Calculadora de precios* [en línea]. Disponible en: https://azure.microsoft.com/es-es/pricing/calculator/?service=virtual-machines [Consulta: 01-10-2016]

**[56].** **INTEROUTE**. *Hosting cloud gratis para startups* [en línea]. Disponible en: http://www.interoute.es/producto/hosting-cloud-gratis-para-startups [Consulta: 22-09-2016]

**[57].** **INTEROUTE.** *Calculadora de precios de VDC* [en línea]. Disponible en: http://www.interoute.es/vdc/calculadora-de-precio [Consulta: 22-09-2016]

**[58].** **ACENS.** *Cloud Datacenter* [en línea]. Disponible en: https://www.acens.com/cloud/cloud-datacenter/ [Consulta: 23-09-2016]

**[59].** **ACENS**. *Cloud Datacenter. Precios* [en línea]. Disponible en: https://www.acens.com/cloud/cloud-datacenter/precios/ [Consulta: 22-09-2016]

**[60].** **ACENS**. *Facturación en Cloud Datacenter* [en línea]. Disponible en: http://cloud.acens.com/preguntas-frecuentes-cloud-datacenter/ [Consulta: 22-09-2016]

**[61].** **SARCAR, V.** *Java Design Patterns.* Anaya, 2016. 978-14-8421-801-3

**[62].** **DOFACTORY.** *.NET Design Patterns and Architectural Guidance* [en línea]. Disponible en: http://www.dofactory.com/ [Consulta: 21-08-2016]

**[63].** **ORACLE.** *Core J2EE Patterns - Data Access Object* [en línea]. Disponible en: http://www.oracle.com/technetwork/java/dataaccessobject-138824.html [Consulta: 21-08-2016]

# 9 ANEXOS

Este capítulo está conformado por una serie de anexos en los que se desarrollan distintos aspectos presentados en capítulos anteriores. Así, en primer lugar se ofrece al lector dos manuales: uno de instalación de las distintas aplicaciones que componen la infraestructura cloud y otro de manejo de la misma. Seguidamente, se ha incorporado un anexo por cada una de estas aplicaciones: BLAST, Trinity, OGS/GE y OpenNebula. En ellos se habla con más detalle sobre la utilidad, funcionamiento y manejo de las mismas. Se ha incluido, además, una serie de anexos en los que se introducen algunos conceptos básicos del ámbito de la Biología y se presentan al lector los distintos ficheros de configuración utilizados por la infraestructura cloud. Finalmente, se ha incorporado un catálogo con los patrones de desarrollo empleados en la implementación de la aplicación web BioCloud.

## 9.1 ANEXO I. MANUALES DE INSTALACIÓN

En el presente anexo se detallan los pasos realizados para instalar los diferentes programas y herramientas que componen la infraestructura cloud.

### 9.1.1 *Información técnica*

**Versiones de Software:**

- **Scientific Linux:** 7.2. Arquitectura: x86_64
- **OpenNebula:** 4.12.1
- **BLAST:** 2.4.0
- **Trinity**: 2.2.0
- **Bowtie**: 1.1.2

**Fuentes de descarga:**

- **Scientific Linux:** https://www.scientificlinux.org/downloads/
- **OpenNebula:** http://opennebula.org/software/
- **BLAST:** ftp://ftp.ncbi.nlm.nih.gov/blast/executables/LATEST/
- **Trinity:** https://github.com/trinityrnaseq/trinityrnaseq/wiki
- **Bowtie:** http://bowtie-bio.sourceforge.net/index.shtml

### 9.1.2 *Creación de una máquina virtual*

Las máquinas virtuales se han creado utilizando QEMU [14], una aplicación que permite crear y simular máquinas virtuales dentro de un SO. Dado que SL no incluye de serie los paquetes de esta herramienta, es necesario instalarla previamente a través del comando:

```
$ sudo yum install qemu
```

QEMU funciona mediante el uso de imágenes, es decir, se debe crear una imagen en la que posteriormente se instalará el SO que se desea emular. Para ello utilizar el siguiente comando. En él se ha especificado que el tipo de la imagen a crear es qcow2 y la capacidad de la misma es de 15 GB:

```
$ qemu-img create -f qcow2 Scientific_Linux.img 15G
```

Para poder realizar la instalación del SO sobre la imagen, es necesario cambiar los permisos del archivo creado y otorgarle permisos de escritura:

```
$ chmod 666 Scientific_Linux.img
```

A continuación, se inicia la instalación del SO usando el siguiente comando. En él se indica que el SO a instalar es de 64 bits, se usará un 1GB de memoria RAM y se habilitará la aceleración con *Kernel-based Virtual Machine* (KVM) para conseguir un mejor rendimiento:

```
$ qemu-system-x86_64 -m 1024 -enable-kvm -hda Scientific_Linux.img –cdrom SL-7-DVD-x86_64.iso -boot d
```

Finalizada la instalación (este proceso se detalla en el apartado siguiente), y apagado el instalador, se puede iniciar la máquina virtual con el siguiente comando:

```
$ qemu-system-x86_64 -m 1024 -enable-kvm Scientific_Linux.img
```

### 9.1.3 *Instalación del SO Scientific Linux*

En este sub-apartado se explica el procedimiento para instalar este SO en cada uno de los nodos que componen la infraestructura cloud.

Cuando se inicia el instalador, aparece la pantalla que se muestra en la Figura 9.1. En dicha ventana, se selecciona la opción “Install Scientific Linux 7.2” y a continuación, establecer el español como idioma a usar durante el proceso.

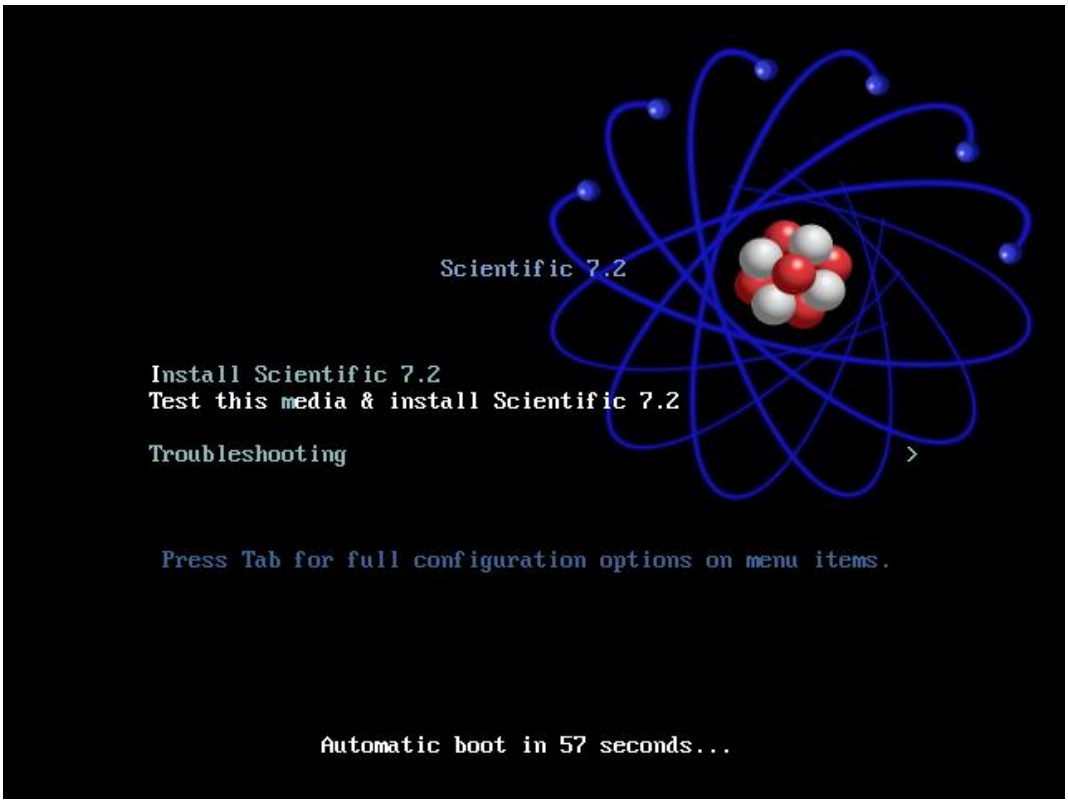



*Figura 9.1. Menú de instalación*

Seguidamente, indicar el destino de la instalación, seleccionando que la configuración del particionado se haga automáticamente (Figura 9.2).


*Figura 9.2. Configuración del destino de la instalación*

A continuación, configurar la contraseña para el usuario root y crear un nuevo usuario (Figura 9.3).

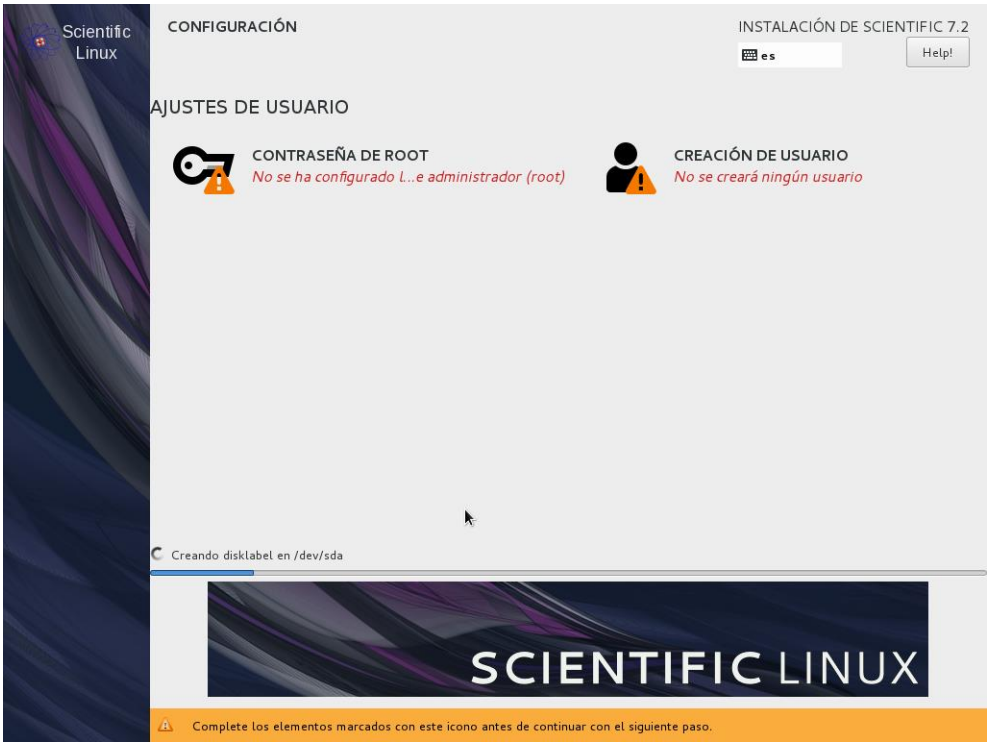



*Figura 9.3. Configuración de usuarios*

Finalmente, tras reiniciar el equipo y volver a iniciar sesión, indicar el idioma y la distribución del teclado a utilizar, quedando así configurado por completo el equipo.

### 9.1.4 *Consideraciones previas*

En las siguientes secciones, se relaciona una serie de configuraciones iniciales que se han llevado a cabo en los distintos nodos antes de comenzar con la implementación de la infraestructura cloud propiamente dicha: cambio de nombres de equipos, creación de usuarios, creación y actualización de variables de entorno, habilitado de los servicios SSH y NFS, generación de pares de claves RSA, configuración de firewall, etc.

#### *9.1.4.1 Cambio del nombre de los equipos*

Es aconsejable cambiar el nombre a todos los equipos para que sea más fácil identificarlos. Para ello, hay que modificar el fichero /etc/hostname de cada uno de los equipos.

En el contexto de este TFG al equipo maestro se le ha asignado el nombre servidor, a los host anfitriones: nodo01, nodo02 y nodo03, y a los recursos computacionales, es decir, a las máquinas virtuales: grid01, grid02...

Para verificar que efectivamente se han aplicado los cambios ejecutar el siguiente comando:

```
$ hostname
```

#### 9.1.4.2 Creación de un nuevo usuario

El gestor distribuido OGS/GE precisa que tanto en el planificador como en todos los recursos computacionales exista un usuario con idénticos nombre y uid. El usuario que se ha creado en el contexto de este TFG se llama sgeadmin.

```
# useradd sgeadmin --uid 500
```

El directorio raíz de dicho usuario se compartirá vía NFS y será en él donde se ubiquen todos los programas y archivos necesarios para el correcto funcionamiento de la infraestructura cloud. De este modo, serán accesibles por todos los recursos computacionales que la componen.

#### 9.1.4.3 Creación del directorio de trabajo

Para facilitar la localización de todos los ficheros relacionados con la infraestructura, se ha creado una carpeta llamada BioCloud en el directorio raíz del usuario sgeadmin y se les ha otorgado permisos de lectura y escritura a todos los usuarios:

```
$ mkdir /home/sgeadmin/BioCloud/
$ sudo chmod 777 /home/sgeadmin/BioCloud/
```

En el interior de este directorio, se han creado otros cuatro llamados entrada, salida, temp y bbdd. Así, en el primero se almacenarán temporalmente los ficheros mientras se realizan los análisis. En el segundo, se alojarán los informes de salida generados al finalizar cada ejecución. El tercero, albergará de forma temporal los ficheros subidos al servidor mientras se generan los directorios temporales utilizados en un análisis. Finalmente, el cuarto alojará en su interior la base de datos BITACORA.

De modo que una vez que se han instalado tanto Trinity como BLAST y creado la base de datos, el contenido de este directorio debería ser el que se muestra en la Figura 9.4:

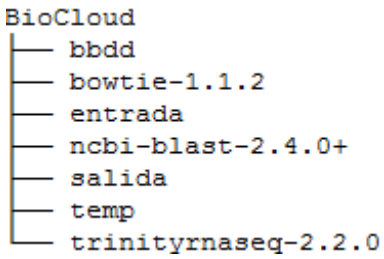



*Figura 9.4. Contenido del directorio de trabajo*

#### 9.1.4.4 Modificación del fichero de inicio .bashrc

Se debe cambiar el fichero /etc/profile para que todas las variaciones realizadas en él sean cargadas automáticamente cuando se inicien los equipos. Para ello, modificar en todos los nodos los archivos /home/sgeadmin/.bashrc y /root/.bashrc. Justo debajo de la línea #bashrc escribir:

```
. /etc/profile
```

Además, en los recursos computacionales se deben incluir, además, las siguientes líneas para evitar que aparezcan errores durante la instalación de OGS/GE:

```
TERM=xterm-256color
SGE_ROOT=/home/sgeadmin/ge2011.11
export TERM SGE_ROOT
```

Tras guardar los cambios y cerrar los ficheros, se recargan para que los cambios realizados surtan efecto:

```
$ source /home/sgeadmin/.bashrc
$ sudo source /root/.bashrc
```

#### 9.1.4.5 Actualización de las variables de entorno

Es necesario añadir ciertas variables de entorno al sistema. Para ello, añadir las siguientes líneas en el fichero Shell /etc/profile del servidor:

```
BLAST=/home/sgeadmin/BioCloud/ncbi-blast-2.4.0+
BLASTDB=$BLAST/db
TRINITY=/home/sgeadmin/BioCloud/trinityrnaseq-2.2.0
BOWTIE=/home/sgeadmin/BioCloud/bowtie-1.1.2

JAVA_HOME=/usr/lib/jvm/jdk
SGE_ROOT=/home/sgeadmin/ge2011.11
```

*PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:$BLAST/bin:$TRINITY:$BOWTIE:$JAVA_HOME/bin:$SGE_ROOT/bin/linux-x64*

*export BLASTDB JAVA_HOME SGE_ROOT PATH*

En el caso de los recursos computacionales, añadir las siguientes líneas:

JAVA_HOME=/usr/lib/jvm/jre-openjdk
SGE_ROOT=/home/sgeadmin/ge2011.11

PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:$JAVA_HOME/bin:*$SGE_ROOT/bin/linux-x64*

export PATH JAVA_HOME SGE_ROOT

#### *9.1.4.6 Configuración del firewall*

Para el correcto funcionamiento de las distintas aplicaciones a instalar, se debe configurar adecuadamente el firewall. Para ello, en primer lugar, pulsar sobre el menú Aplicaciones > Varios > Cortafuego. En la pestaña Servicios, marcar *nfs* (Figura 9.5).

Por otro lado, es necesario abrir una serie de puertos en el firewall de cada uno de los equipos:

- **Servidor**
  - **6444 (TCP/UDP)** → Puerto de escucha del demonio sge_qmaster.
  - **8080 (TCP)** → Puerto de escucha de Tomcat.
  - **9869 (TCP)** → Puerto de escucha de OpenNebula.
- **Nodo anfitrión**
  - **59XX (TCP)** → El puerto 5900 es el puerto base de la herramienta Virtual Network Computing (VNC), en el que se escuchan las conexiones [15]. Para cada máquina virtual, se establece automáticamente como puerto de escucha el puerto 5900 + su ID, es decir, al puerto base se le debe sumar el ID de la máquina virtual. De modo que si la máquina virtual tiene ID=0, su puerto de escucha será el 5900; si su ID es 1, su puerto será el 5901 y así sucesivamente.
- **Máquinas virtuales**
  - **6445 (TCP/UDP)** → Puerto de escucha del demonio sge_execd.

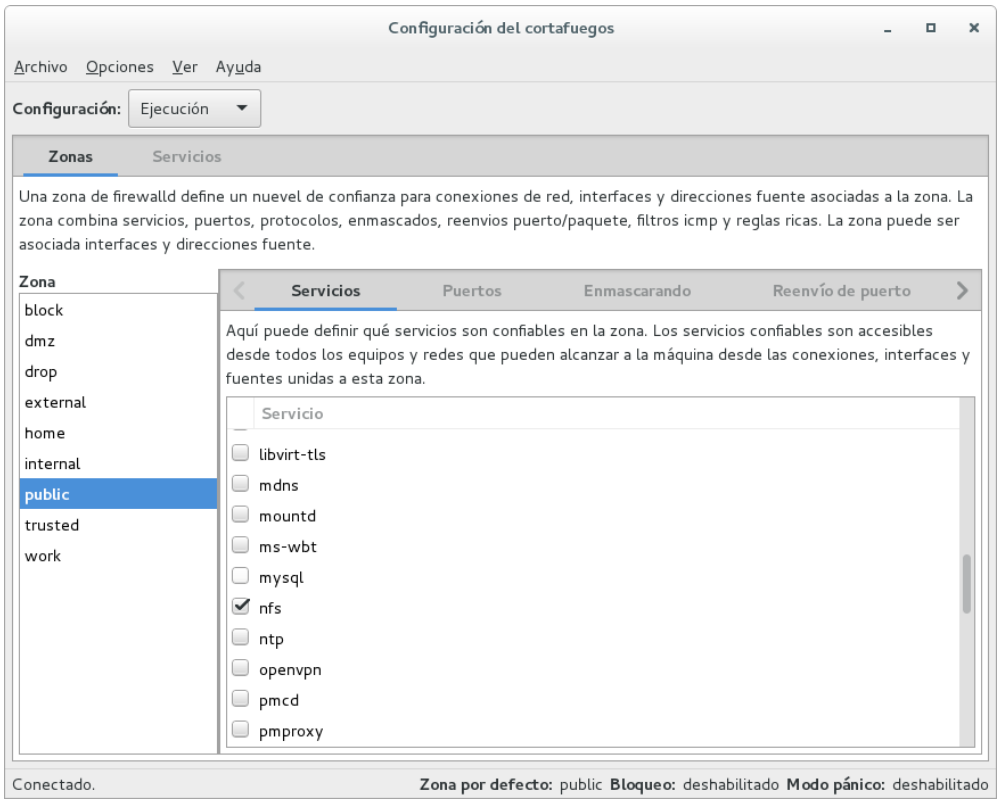



*Figura 9.5. Configuración de confiabilidad del servicio nfs*

Para abrir los puertos necesarios, acceder a la pestaña Puertos y pulsar sobre el botón Añadir. Ingresar los distintos puertos y el protocolo de nivel de transporte empleado y, a continuación, aceptar los cambios (Figura 9.6).

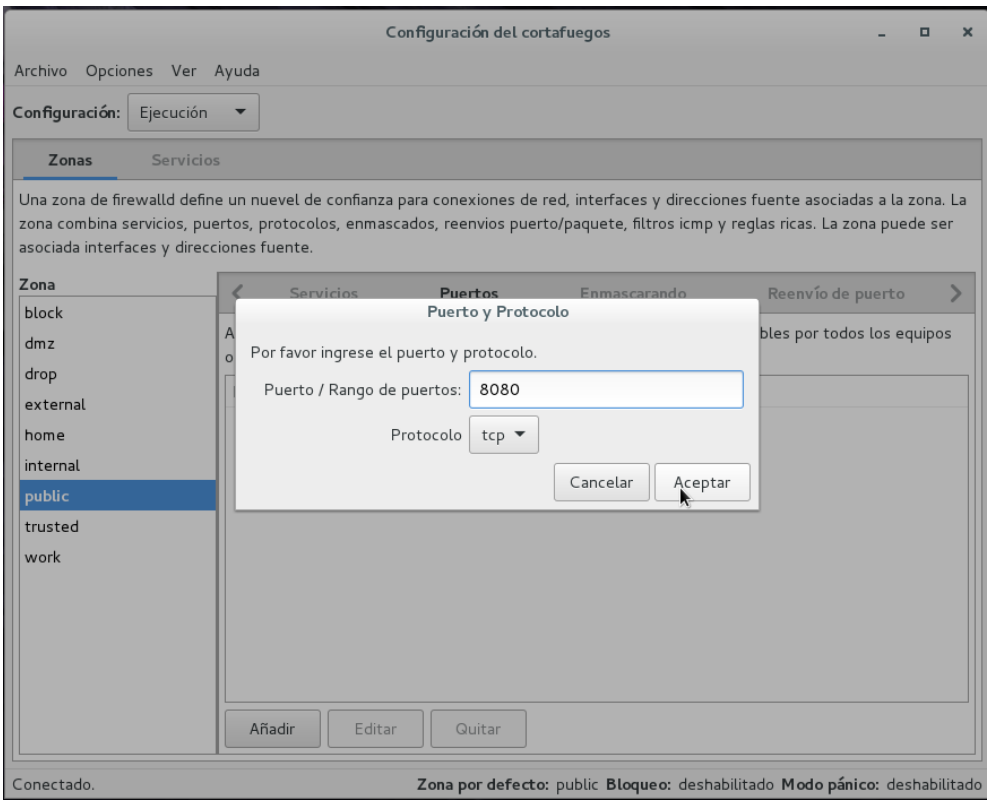



*Figura 9.6. Apertura de puertos en el firewall*

Finalmente, pulsar sobre Opciones > Runtime To permant para guardar la configuración y recargar el firewall para que se apliquen los cambios (Figura 9.7).

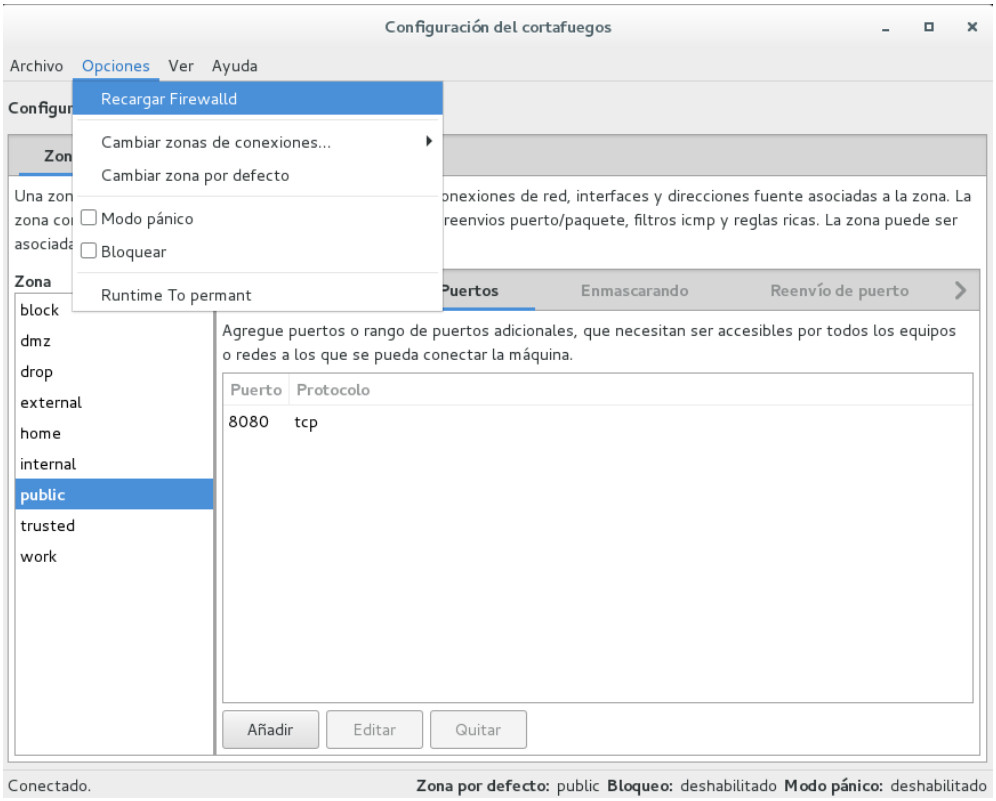



*Figura 9.7. Recarga del firewall*

### 9.1.4.7 Comprobación de la versión de Java

La herramienta Trinity requiere que la versión de Java sea la 1.7 o superior. Si la versión de Java existente en el sistema fuera inferior, éste deberá ser actualizado.

### 9.1.4.8 Configuración del fichero /etc/hosts

Para que la resolución de nombres se lleve a cabo correctamente durante el proceso de instalación del programa OGS/GE, modificar en todos los equipos el fichero /etc/hosts.

Añadir, por cada nodo que compone la infraestructura, una línea con la dirección IP, el nombre de la máquina y un alias para la misma (Figura 9.8).

```
192.168.1.22    servidor.localdomain servidor
192.168.1.23    nodo01.localdomain nodo01
192.168.1.24    nodo02.localdomain nodo02
192.168.1.25    nodo03.localdomain nodo03

192.168.1.200   grid01.localdomain grid01
192.168.1.201   grid02.localdomain grid02
192.168.1.202   grid03.localdomain grid03
192.168.1.203   grid04.localdomain grid04
192.168.1.204   grid05.localdomain grid05
192.168.1.205   grid06.localdomain grid06
192.168.1.206   grid07.localdomain grid07
192.168.1.207   grid08.localdomain grid08
192.168.1.208   grid09.localdomain grid09
192.168.1.209   grid10.localdomain grid10
```

*Figura 9.8. Configuración del fichero /etc/hosts*

#### *9.1.4.9 Desactivado de SELinux*

Es recomendable deshabilitar el módulo de seguridad Security-Enhanced Linux (SELinux) en todos los equipos modificando el archivo /etc/sysconfig/selinux. Para ello, establecer el parámetro SELINUX a disabled, guardar los cambios, salir del archivo y a continuación, ejecutar los siguientes comandos:

```
# setenforce 0
# getenforce
```

Si todo ha ido correctamente, el resultado de este último debería ser Permissive.

#### *9.1.4.10 Configuración de SSH*

Para poder establecer conexiones haciendo uso del protocolo Secure SHell (SSH) es necesario instalar en todos los equipos el paquete OpenSSH y seguidamente, habilitar el servicio:

```
# yum install openssh-server openssh-clients
# chkconfig sshd on
# service sshd start
```

#### *9.1.4.11 Configuración de NFS*

OpenNebula y OGS/GE hacen uso el protocolo NFS. Para instalar el paquete NFS, habilitar e iniciar los servicios se deben ejecutar los siguientes comandos:

```
# yum install nfs-utils
# systemctl enable rpcbind
# systemctl enable nfs-server
# systemctl start rpcbind
# systemctl start nfs-server
```

#### *9.1.4.12 Generación de claves RSA en el servidor*

OGS/GE necesita establecer conexiones SSH con el usuario root de los recursos computacionales sin que sea necesario hacer uso de contraseñas (*Passwordless root SSH*). Para ello, es necesario generar en el servidor un par de claves pública/privada de RSA (dejando en blanco el campo *passphrase*):

```
$ sudo bash
# ssh-keygen -t rsa
```

La clave pública generada se copiará en el apartado de contextualización de las plantillas instanciadas para crear las distintas máquinas virtuales que harán la función de recursos computacionales.

#### *9.1.4.13 Instalación del paquete de contextualización OpenNebula*

Para poder cambiar la configuración de las máquinas virtuales y adaptarlas a la red en la que se encuentran o introducir en ellas parámetros tales como claves públicas de SSH, se requiere instalar en cada máquina virtual el paquete *opennebula-context*. Para ello, ejecutar el comando:

```
$ wget https://github.com/OpenNebula/addon-context-linux/releases/download/v4.10.0/one-context_4.10.0.rpm
```

Tras ubicarse en el directorio de descarga, iniciar la instalación a través el siguiente comando:

```
$ rpm -i one-context-4.10.0.rpm
```

### 9.1.5 *Instalación de la herramienta BLAST*

En esta sección se detallan los pasos a seguir para descargar e instalar la herramienta bioinformática BLAST [16].

#### *9.1.5.1 Requisitos previos*

EL programa BLAST se puede descargar desde el servidor FTP del *National* Center *for Biotechnology Information* (NCBI). Dado que SL no incluye de serie ningún cliente FTP es necesario instalarlo previamente:

```
$ sudo yum install ftp
```

#### *9.1.5.2 Descarga*

Una vez se ha ubicado en el directorio en el que se desea instalar este programa, establecer una conexión FTP con el servidor del NCBI para proceder a su descarga:

```
$ ftp ftp.ncbi.nlm.nih.gov
```

Autenticarse contra el servidor con el usuario anonymous, dejando en blanco el campo de contraseña. Una vez establecida la conexión, situarse en el directorio donde se encuentra alojado el programa:

```
> cd blast/executables/LATEST/
```

Indicar que la transferencia sea de tipo binaria:

```
> bin
```

Obtener la aplicación del servidor:

```
> get ncbi-blast-2.4.0+-x64-linux.tar.gz
```

Por último, finalizar la conexión con el servidor FTP:

```
> bye
```

#### *9.1.5.3 Instalación*

El proceso de instalación es tan sencillo como descomprimir el archivo descargado y crear las variables de entorno correspondientes (Apartado 9.1.4.5):

```
$ tar zxvpf ncbi-blast+2.4.0+-x64-linux.tar.gz
```

En este trabajo, además, se ha creado una nueva carpeta llamada db en el interior del directorio raíz del programa. Dicha carpeta almacenará las bases de datos locales utilizadas en los análisis con esta herramienta.

Finalmente, para comprobar que todo ha ido correctamente, al ejecutar el siguiente comando deberían aparecer todas las opciones y argumentos que se pueden utilizar con esta herramienta:

```
$ blastp -help
```

### 9.1.6 *Instalación de la herramienta Trinity*

En las siguientes páginas se explican los distintos pasos que se han seguido para descargar e instalar la herramienta Trinity [17].

#### *9.1.6.1 Requisitos previos*

Para que la instalación de Trinity finalice correctamente, deben estar instalados en los nodos una serie de paquetes. Esta lista de paquetes es la necesaria para SL 7.2, para otras versiones o distribuciones, el número de paquetes, así como el nombre de los mismos, pueden variar ligeramente:

```
$ sudo yum install -y gcc-c++ ncurses-devel zlib-devel perl-Data-Dumper
```

Por otro lado, y como ya se ha comentado anteriormente, Trinity requiere que la versión de Java instalada en el sistema sea la 1.7 o superior.

Finalmente, Trinity hace uso de una herramienta llamada Bowtie [18], un sistema para alinear cadenas cortas, considerado hasta la fecha uno de los más rápidos del mercado. Una vez que se ha situado en el directorio de instalación, descargar el programa Bowtie ejecutando para ello el siguiente comando y descomprimir el archivo descargado.

```
$ wget https://sourceforge.net/projects/bowtie-bio/files/bowtie/1.1.2/bowtie-1.1.2-linux-x86_64.zip/download
```

#### *9.1.6.2 Descarga e instalación*

Para instalar el programa Trinity, es necesario situarse en el directorio de instalación, descargar el programa y extraer el contenido del fichero.

```
$ wget https://github.com/trinityrnaseq/trinityrnaseq/archive/v2.2.0.tar.gz
```

Tras ubicarse en el directorio raíz del programa, instalarlo ejecutando el siguiente comandos.

```
$ make
```

Si se desean compilar componentes adicionales para dar soporte a algunos análisis tales como la estimación de abundancia usando RSEM, lanzar el siguiente comando:

```
$ make plugins
```

#### 9.1.6.3 Comprobación de la instalación (Opcional)

Se puede verificar que la instalación se ha realizado con éxito ejecutando los siguientes comandos:

```
$ cd /home/sgeadmin/BioCloud/trinityrnaseq-2.4.0/sample_data/test_Trinity_Assembly/
$ ./runMe.sh
```

### 9.1.7 Instalación de OpenNebula

En los siguientes apartados se detallan los pasos a seguir para instalar el programa OpenNebula [19]. Todos los comandos precedidos por # deberán ser ejecutados como root y aquellos precedidos por $, como usuario oneadmin.

#### 9.1.7.1 Instalación en el servidor

En esta sección se detallarán los pasos a seguir para instalar y configurar adecuadamente el programa OpenNebula en el equipo servidor, es decir, en el front-ed.

##### 9.1.7.1.1 Instalación del repositorio de OpenNebula

En primer lugar, instalar el repositorio de EPEL.

```
$ sudo yum install epel-release
```

Si el comando anterior no funciona puede deberse a que los repositorios extra para SL están deshabilitados. Para instalar manualmente dicho repositorio, ejecutar el siguiente comando.

```
$ wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
$ sudo rpm -Uvh epel-release-latest-7*.rpm
```

Por último, añadir el repositorio de OpenNebula.

```
# cat << EOT > /etc/yum.repos.d/opennebula.repo
[opennebula]
name=opennebula
baseurl=http://downloads.opennebula.org/repo/4.12/CentOS/7/x86_64/
enabled=1
gpgcheck=0
EOT
```

#### 9.1.7.1.2 Instalación de OpenNebula

La instalación completa de OpenNebula en el servidor requiere los paquetes *opennnebula-server* y *opennebula-sunstone:*

```
# yum install opennebula-server opennebula-sunstone
```

Finalizada la descarga de los mismos, ejecutar *install_gems* para instalar una serie de gemas de Ruby. En el siguiente menú, seleccionar la segunda opción:

```
# /usr/share/one/install_gems
lsb_release command not found. If you are using a RedHat based
distribution install redhat-lsb
Select your distribution or press enter to continue without
installing dependencies.

0. Ubuntu/Debian
1. CentOS/RedHat/Scientific
```

#### 9.1.7.1.3 Configuración y arranque del servicio

Se deben arrancar dos procesos en el servidor: el demonio principal de OpenNebula (oned) y la interfaz gráfica de usuario: sunstone.

Por motivos de seguridad, sunstone sólo escucha peticiones en la interfaz de Loopback. Para que escuche peticiones en cualquier interfaz es necesario modificar el fichero /etc/one/sunstone-server.conf y cambiar el parámetro :hosts: 127.0.0.1 a :hosts: 0.0.0.0.

Realizado dicho cambio, habilitar e iniciar dichos servicios:

```
# systemctl enable opennebula
# systemctl start opennebula
# systemctl enable opennebula-sunstone
# systemctl start opennebula-sunstone
```

#### 9.1.7.1.4 Compartición del directorio /var/lib/one

OpenNebula requiere que se comparta vía NFS el directorio /var/lib/one. Para ello, modificar el fichero /etc/exports y añadir la siguiente línea. En ella se especifica que se va a compartir el directorio en modo lectura/escritura (rw), que se comunicarán a los usuarios los cambios realizados sobre los archivos cuando realmente se hayan ejecutado (sync), que no se va a comprobar el camino hasta el directorio que se exporta (no_subtree_check) y que los accesos realizados como usuario root desde los clientes serán también como usuario root en el servidor (no_root_sqash) [20].

```
# echo "/var/lib/one/ *(rw,sync,no_subtree_check,no_root_squash)" >> /etc/exports
```

A continuación, se activa la compartición de recursos:

```
# exportfs -ra
```

Finalmente, se reinicia el servicio:

```
# systemctl restart nfs.service
```

#### 9.1.7.1.5 Configuración de SSH

Además, OpenNebula también precisa establecer conexiones SSH sin contraseña con el usuario oneadmin desde cualquier nodo a cualquier otro nodo. Para lograrlo, es necesario previamente conocer la contraseña asociada a dicho usuario ejecutando el siguiente comando:

```
# cat /var/lib/one/.one/one_auth
```

Se devolverá una cadena con la estructura oneadmin:contraseña. Conocida la contraseña, se inicia sesión como dicho usuario usando el siguiente comando:

```
# su - oneadmin
```

Además, se debe configurar el fichero ~/.ssh/config para evitar confirmar que se desea añadir al fichero knwon_hosts la clave pública de otro equipo cada vez que se establece una conexión SSH con un equipo por primera vez.

```
$ cat << EOT > ~/.ssh/config
Host *
StrictHostKeyChecking no
UserKnownHostsFile /dev/null
EOT
$ chmod 600 ~/.ssh/config
```

#### 9.1.7.1.6 Modificación del fichero oned.conf

Para que al subir una imagen al sistema no se obtenga ningún error informando de que no hay espacio libre en el datastore, aunque esto no sea cierto, es necesario deshabilitar la comprobación de la capacidad del datastore; para ello, modificar el fichero /etc/one/oned.conf y establecer el parámetro DATASTORE_CAPACITY_CHECK a “no”. Una vez efectuada esta operación, reiniciar OpenNebula para que los cambios surtan efecto.

### *9.1.7.2 Instalación en los nodos anfitriones*

Seguidamente, se detallará el proceso necesario para configurar e instalar el programa OpenNebula en los nodos anfitriones de las máquinas virtuales.

#### 9.1.7.2.1 Comprobación previa

Primeramente, es necesario comprobar que el equipo tenga habilitadas las extensiones de virtualización. Para ello, usar el comando:

```
# grep -E ‘vmx|svm’ /proc/cpuinfo
```

En caso no obtener nada como respuesta, el procesador no soporta virtualización, con lo cual este equipo no podría ser utilizado como nodo anfitrión.

#### 9.1.7.2.2 Instalación del repositorio de OpenNebula

En primer lugar es necesario añadir el repositorio de OpenNebula. Para ello, introducir las siguientes líneas en una consola de comandos:

```
# cat << EOT > /etc/yum.repos.d/opennebula.repo
[opennebula]
name=opennebula
baseurl=http://downloads.opennebula.org/repo/4.12/CentOS/7/x86_64/
enabled=1
gpgcheck=0
EOT
```

#### 9.1.7.2.3 Instalación del paquete KVM

Dado que se va a virtualizar con kvm, se debe instalar el siguiente paquete en cada uno de los nodos anfitriones:

```
# yum install opennebula-node-kvm
```

Seguidamente, se inician los servicios:

```
# systemctl start messagebus.service
# systemctl enable libvirtd.service
# systemctl start libvirtd.service
```

#### 9.1.7.2.4 Configuración de los dispositivos de red

Es necesario que la interfaz principal esté conectada a una interfaz de red en puente virtual. OpenNebula, además, requiere que el nombre asignado a este puente sea el mismo en todos los nodos anfitriones. Para ello, modificar el fichero /etc/sysconfig/network-scripts/ifcfg-enp0s3 y configurarlo con la siguiente información:

```
DEVICE=enp2s0
NM_CONTROLLED=no
ONBOOT=yes
TYPE=Ethernet
BRIDGE=br0
```

A continuación, crear un nuevo fichero llamado /etc/sysconfig/network-scripts/ifcfg-br0 con el siguiente contenido:

```
DEVICE=br0
TYPE=Bridge
ONBOOT=yes
BOOTPROTO=dhcp
NM_CONTROLLED=no
```

Una vez efectuados todos estos pasos, reiniciar los servicios de red:

```
# systemctl enable network.service
# systemctl restart network.service
```

#### 9.1.7.2.5 Configuración de NFS

Finalmente, es necesario montar en cada nodo anfitrión el directorio compartido por el servidor. Para ello, ejecutar los siguientes comandos:

```
# echo "192.168.1.22:/var/lib/one/ /var/lib/one/ nfs soft,intr,rsize=8192,wsize=8192" >> /etc/fstab
# mount /var/lib/one/
```

Además, se debe permitir al usuario oneadmin gestionar el servicio libvirt como si fuera usuario root. Para ello, comprobar que existe el archivo /etc/libvirt/qemu.conf y que éste contiene la siguiente información:

```
user = "oneadmin"
group = "oneadmin"
dynamic_ownership = 0
```

En caso contrario, crearlo con este contenido y reiniciar el servicio libvirt para que surtan efecto los cambios realizados:

```
# systemctl restart libvirtd.service
```

### 9.1.8 *Instalación de OGS/GE*

En los siguientes apartados, se detallará el proceso de instalación del programa OGS/GE [21]. En este caso, los comandos precedidos por # deberán ser ejecutados como root y aquellos precedidos por $, como usuario normal.

#### *9.1.8.1 Configuración del servidor*

A continuación, se explicarán las distintas operaciones y configuraciones que se deben realizar en el equipo servidor. Éste desempeñará las funciones tanto de gestor como de planificador.

#### 9.1.8.1.1 Instalación del sistema gestor

En primer lugar, descargar el programa en el equipo:

```
$ wget http://dl.dropbox.com/u/47200624/respin/ge2011.11.tar.gz
```

Tras extraer el contenido, moverlo al directorio raíz del usuario sgeadmin y cambiar el propietario del mismo:

```
$ mv ge2011.11 /home/sgeadmin/
$ sudo chown -R sgeadmin:sgeadmin /home/sgeadmin/
```

Finalmente, modificar los permisos de este directorio para que todos los nodos puedan leer y ejecutar del mismo.

```
$ sudo chmod –R 755 /home/sgeadmin/ge2011.11
```

#### 9.1.8.1.2 Compartición del directorio /home/sgeadmin/

En primer lugar, exportar el directorio compartido por el servidor, indicando que se va a compartir el directorio en modo lectura/escritura (rw), que se comunicarán a los usuarios los cambios realizados sobre los archivos cuando realmente se han ejecutado (sync), que no se va a comprobar el camino hasta el directorio que se exporta (no_subtree_check) y que los accesos realizados como usuario root desde los clientes serán también de root en el servidor NFS (no_root_sqash) [20].

```
# echo “/home/sgeadmin/ *(rw,sync,no_subtree_check,no_root_squash)” >> /etc/exports
```

Seguidamente, activar la compartición de recursos.

```
# exportfs -ra
```

Finalmente, reiniciar el servidor NFS.

```
# systemctl restart nfs-server
```

### *9.1.8.2 Configuración de los recursos computacionales*

En cada recurso computacional, es decir, en cada máquina virtual, se debe crear también el usuario sgeadmin para poder instalar en ellos el programa OGS/GE.

```
$ sudo adduser sgeadmin -uid 500
```

#### 9.1.8.2.1 Configuración de NFS

Montar el directorio compartido por el planificador configurándolo para que éste sea montado automáticamente en el inicio.

```
$ sudo mount –t nfs 192.168.1.22:/home/sgeadmin/ /home/sgeadmin/
# echo “ 192.168.1.22:/home/sgeadmin/ /home/sgeadmin/ nfs” >> /etc/fstab
```

### *9.1.8.3 Establecimiento de conexiones SSH*

Modificar en los recursos computacionales el fichero /etc/ssh/sshd_config de tal modo que los siguientes parámetros tengan los valores que se indican:

```
PermitRootLogin yes
RSAAuthentication yes
PubKeyAuthentication yes
```

Desde el planificador establecer una conexión SSH con cada recurso, especificando que no se pregunte si se desea aceptar la clave púbica de la misma.

```
# ssh root@grid01 –o StrictHostKeyChecking=no
```

Finalmente, para que no surjan problemas relacionados con la verificación de claves EDCSA durante la instalación del programa OGS/GE, es necesario ejecutar el siguiente comando en el planificador:

```
# ssh -o StrictHostKeyChecking=no root@servidor.localdomain
# cp ~/.ssh./id_rsa.pub >> /.ssh/authorized_keys
```

### *9.1.8.4 Configuración del sistema Grid*

Tras comprobar que se pueden establecer conexiones SSH sin que se solicite ninguna contraseña, se puede iniciar el instalador GUI en el planificador [21].

```
$ sudo bash
# cd /home/sgeadmin/ge2011.11
# ./start_gui_installer
```

En la pantalla “*Choose component*” (Figura 9.9) seleccionar las siguientes opciones:

- *Qmaster host*
- *Execution host(s)*
- Instalación personalizada *(Custom installation*).

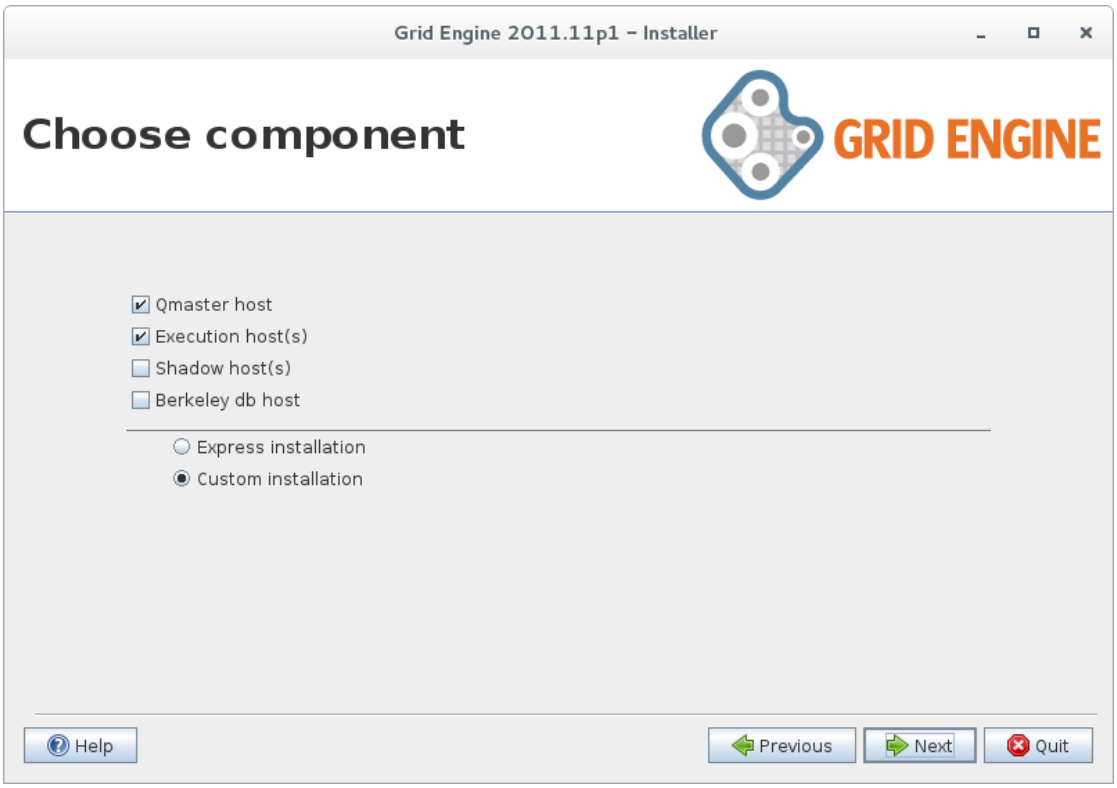



*Figura 9.9. Ventana de elección de componentes*

A continuación, en la pantalla "*Main configuration*" (Figura 9.10), cambiar el nombre del cluster, el puerto a utilizar, etc. (En este trabajo se ha mantenido la configuración por defecto). Desmarcar la opción "*Use JMX"* y pulsar *Next.*

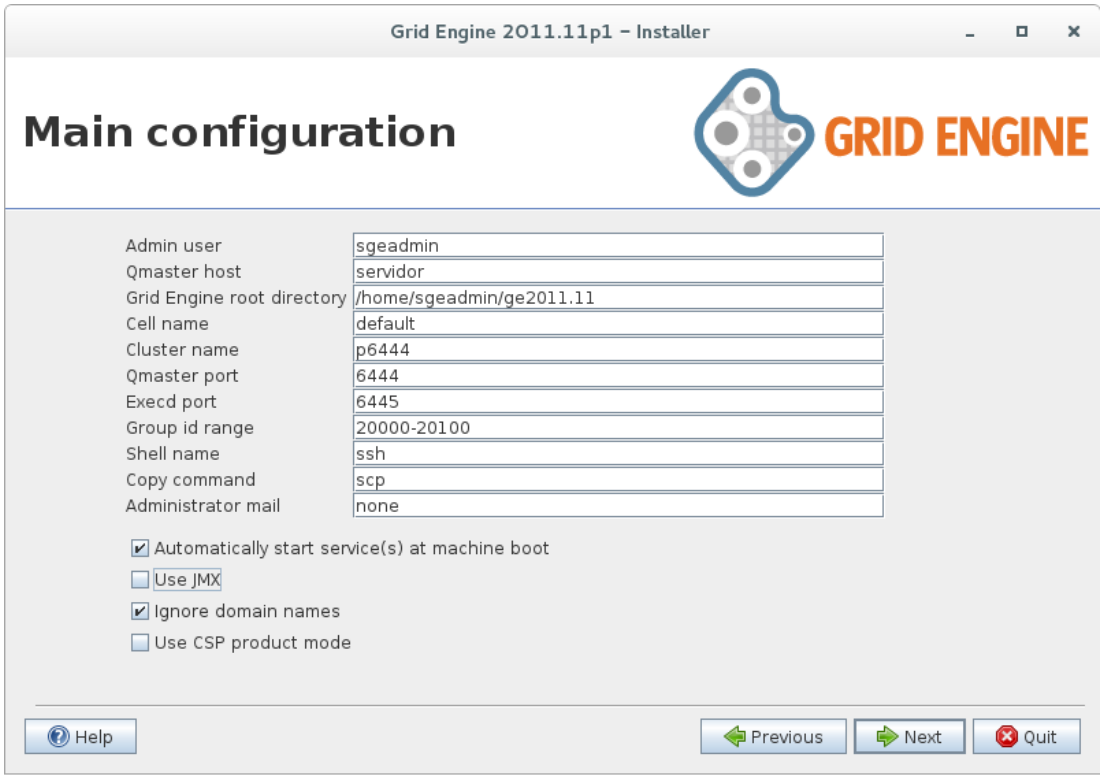



*Figura 9.10. Ventana de configuración principal*

Seguidamente, en la pantalla "*Spooling configuration*" (Figura 9.11) seleccionar Classic como método de encolamiento. De este modo, el encolado de tareas se lleva a cabo empleando un formato legible para el ser humano.

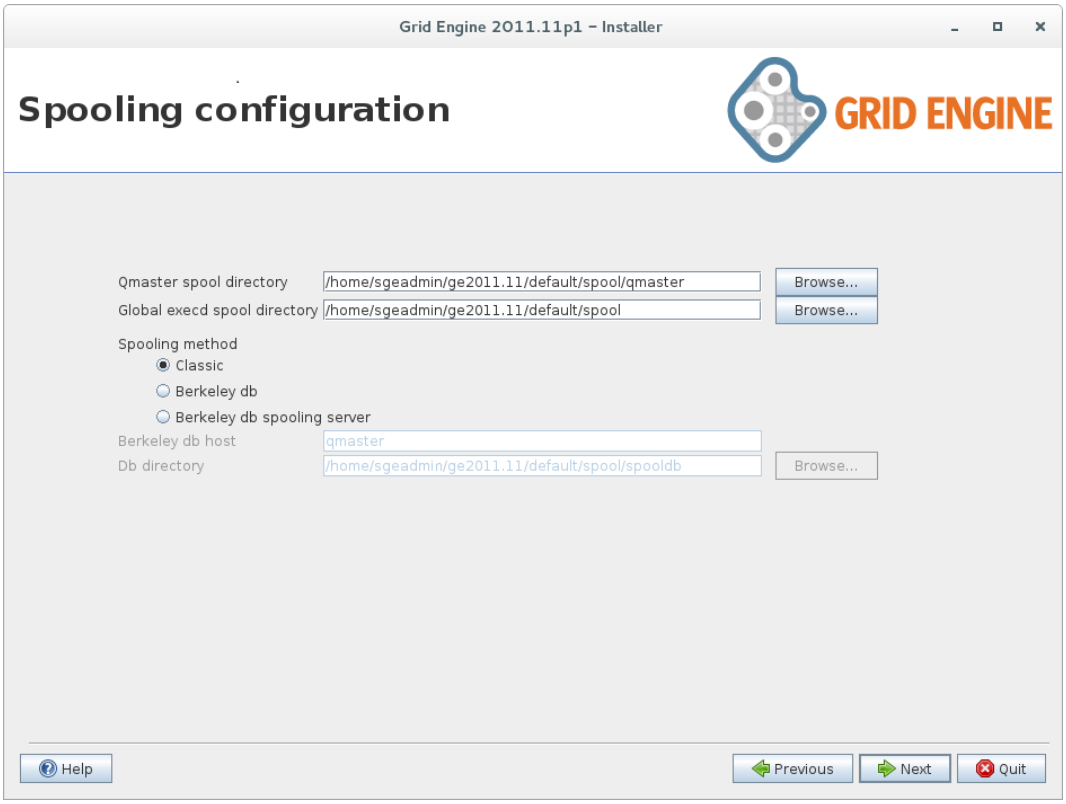



*Figura 9.11. Ventana de configuración de encolamiento*

Finalmente, en la pantalla "*Select Hosts*" (Figura 9.12), añadir los hosts que formarán parte del sistema grid. A cada uno de estos nodos se les debe asignar un rol:

- **Master/Admin** es el planificador.
- **Execution** son los hosts que realizarán tareas.
- **Submit,** nodo que puede añadir trabajos a la cola de tareas.

Sólo si el estado de los mismos es alcanzable (*Reachable*) la instalación llegará a buen término.

Una vez que la instalación ha finalizado, se puede comprobar el estado de los distintos recursos computacionales ejecutando el siguiente comando.

```
$ qstat -f
```

Se ofrecerá un listado de los distintos recursos computacionales tal y como se muestra en la Figura 9.13.

Si tras reiniciar algún recurso no se iniciase automáticamente el servicio sge_execd, ejecutar los siguientes comandos:

```
# /sbin/chkconfig sgeexecd.p6444 on
# /sbin/chkconfig sgeexecd.p6444 start
```

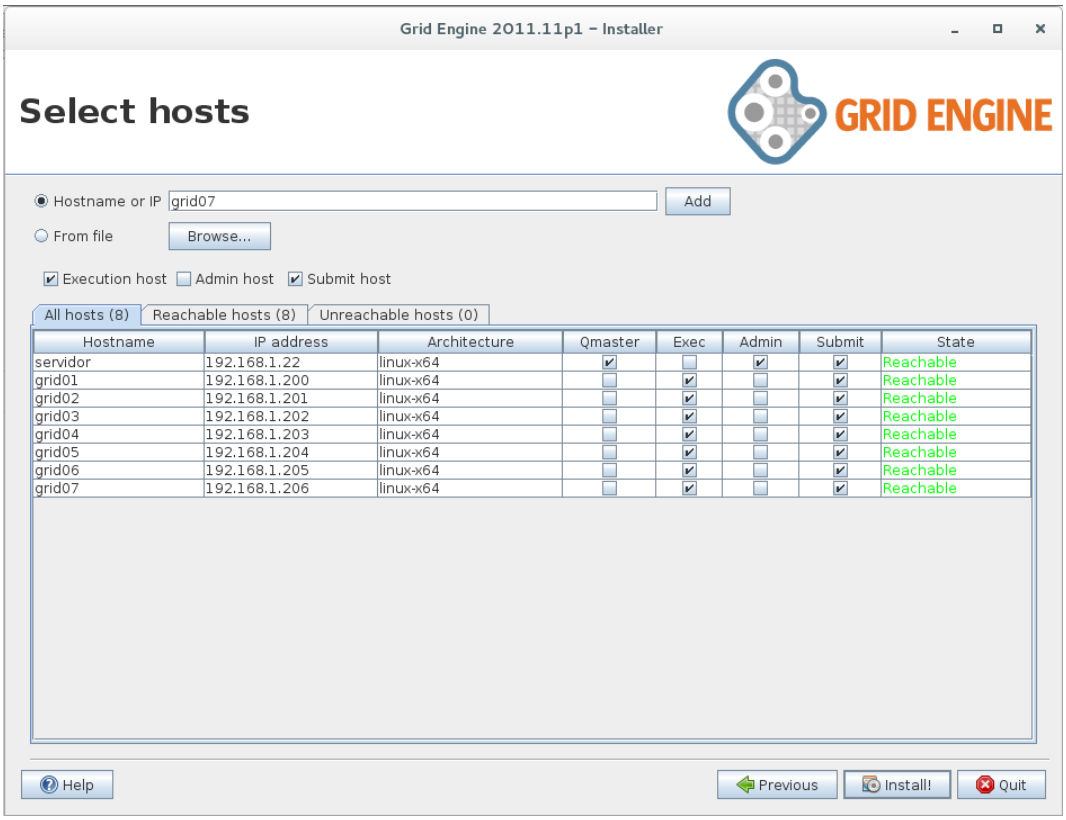



*Figura 9.12. Ventana de selección de hosts*

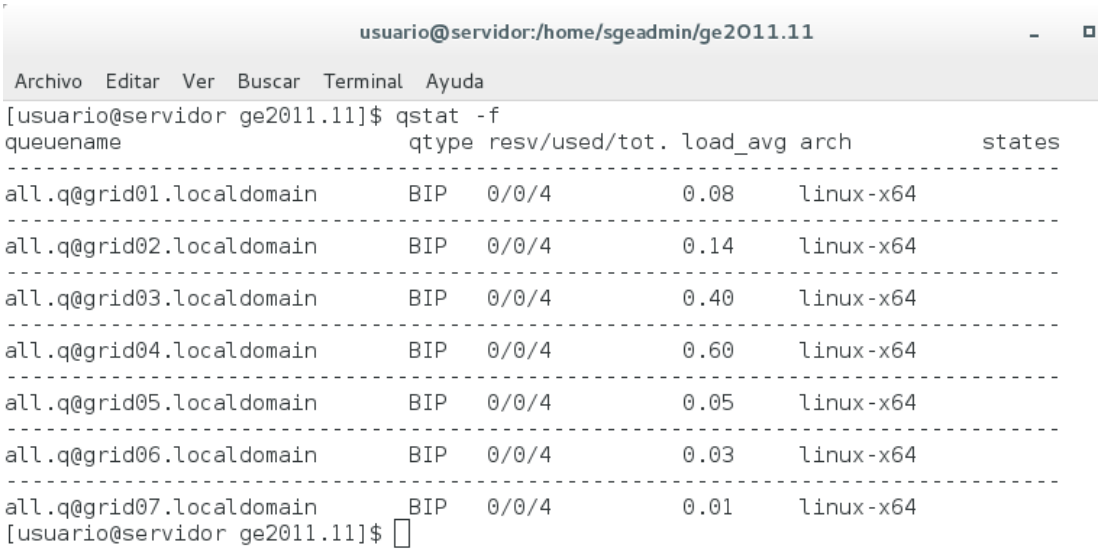



*Figura 9.13.Conulta del estado de los recursos computacionales*

### 9.1.9 *Creación de la base de datos*

Como se ha comentado anteriormente, en el contexto de este trabajo se ha utilizado SQLite como SGBD. La base de datos empleada, llamada BITACORA, consta de tres entidades: TAREASLANZADAS, BDBLAST y DOCUMENTOS.

Para crear la base de datos, ejecutar el siguiente comando:

```
$ SQLite3 BITACORA.db
```

El Lenguaje de Definición de Datos (DDL) que define la estructura de esta base de datos es el siguiente:

```
CREATE TABLE TAREASLANZADAS(
ID INTEGER PRIMARY KEY,
UUID CHAR(30) NOT NULL,
PROGRAMA CHAR(10) NOT NULL,
FECHA_INICIO DATETIME NOT NULL,
FECHA_FIN DATETIME,
ESTADO INT NOT NULL,
COMANDO CHAR(100),
URL CHAR(100) NOT NULL
);

CREATE TABLE BDBLAST(
ID INTEGER PRIMARY KEY,
NOMBRE CHAR(25) NOT NULL,
TIPO CHAR(5) NOT NULL,
CONTENIDO BLOB NOT NULL,
RESUMEN BLOB NOT NULL,
FECHA_CREACION DATETIME NOT NULL
);

CREATE TABLE DOCUMENTOS(
ID INTEGER PRIMARY KEY,
UUID CHAR(30) NOT NULL,
NOMBRE CHAR(30) NOT NULL,
FORMATO CHAR(30) NOT NULL,
INFORME BLOB NOT NULL,
);
```

### 9.1.10 *Instalación de Apache Tomcat 7*

Como servidor web se ha empleado Apache Tomcat. Para su instalación y configuración en SL se deben seguir los pasos que se indican a continuación [22].

#### *9.1.10.1 Instalación*

La instalación de Tomcat se lleva a cabo ejecutando el siguiente comando:

```
$ sudo yum install tomcat
```

Si se desea instalar la página raíz por defecto de Tomcat (tomcat-webapps), así como la aplicación de administración web y el gestor de host virtuales (tomcat-admin-webapps), se emplea el siguiente comando:

```
$ sudo yum install tomcat-webapps tomcat-admin-webapps
```

Finalmente, configurar su inicio automático para que en caso de reinicio del equipo, el servidor web vuelva a funcionar:

```
$ sudo systemctl enable tomcat.service
```

#### *9.1.10.2 Configuración de Tomcat Web Management Interfaz*

Es necesario añadir un login al servidor para permitir el acceso a las secciones de administración. Editar para ello el fichero tomcat-users.xml y añadir entre las pestañas <tomcat-users></tomcat-users> la siguiente información:

```
<role rolename="manager-gui"/>
<role rolename="admin-gui"/>
<user username="admin" password="admin" roles="manager-gui,admin-gui"/>
```

#### *9.1.10.3 Gestión del servidor*

Para iniciar, parar o reiniciar el servidor Tomcat, se utilizan los siguientes comandos, respectivamente:

```
$ sudo systemctl start tomcat
$ sudo systemctl stop tomcat
$ sudo systemctl restart tomcat
```

#### *9.1.10.4 Acceso a la Interfaz Web*

Una vez que se ha puesto en marcha el servidor, acceder a la página de administración del mismo a través de la siguiente URL: http://localhost:8080. Aparecerá la página de inicio de Tomcat (Figura 9.14).

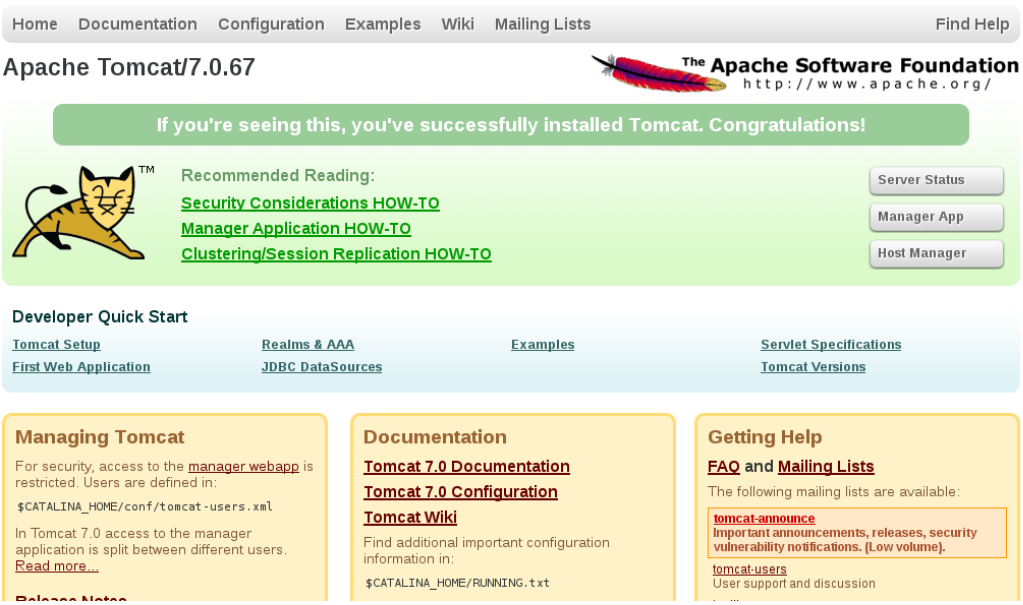



*Figura 9.14. Página de inicio de Apache Tomcat*

Para administrar las aplicaciones desplegadas en el servidor se puede utilizar el ***Web Application Manager*.** Para ello, pulsar sobre ***Manager App***. Aparecerá una nueva página en la que se muestran las aplicaciones desplegadas y desde la cual se podrán parar, recargar o replegar las mismas (Figura 9.15).

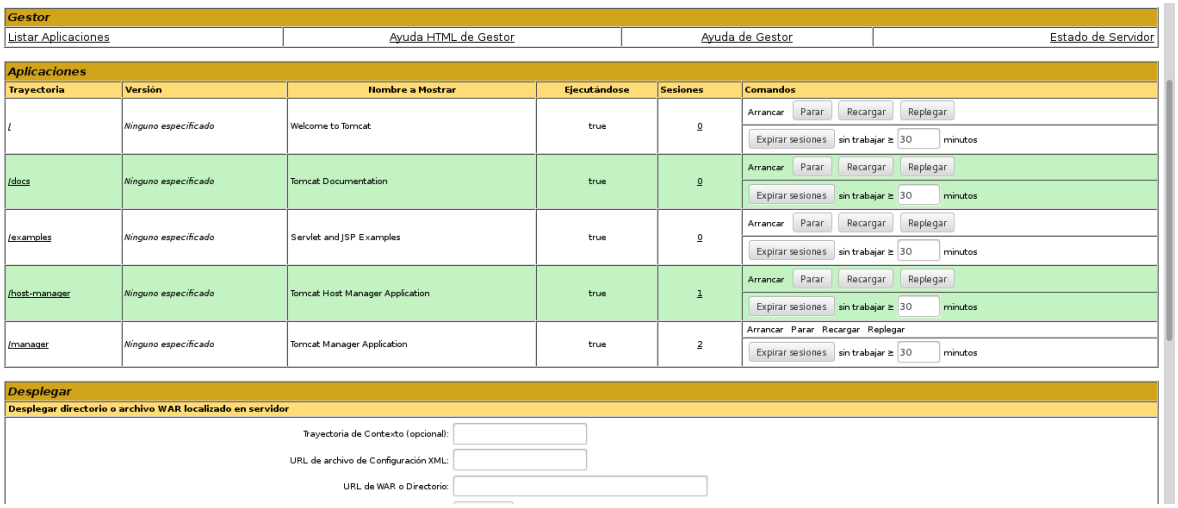



*Figura 9.15. Manager App*

Si se desea desplegar una aplicación, subir el archivo *.WAR generado con Eclipse o con cualquier otro IDE a través del formulario de la sección “Archivo WAR a desplegar” (Figura 9.16).

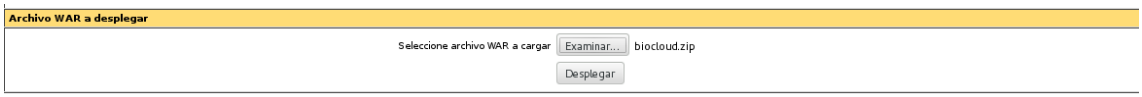
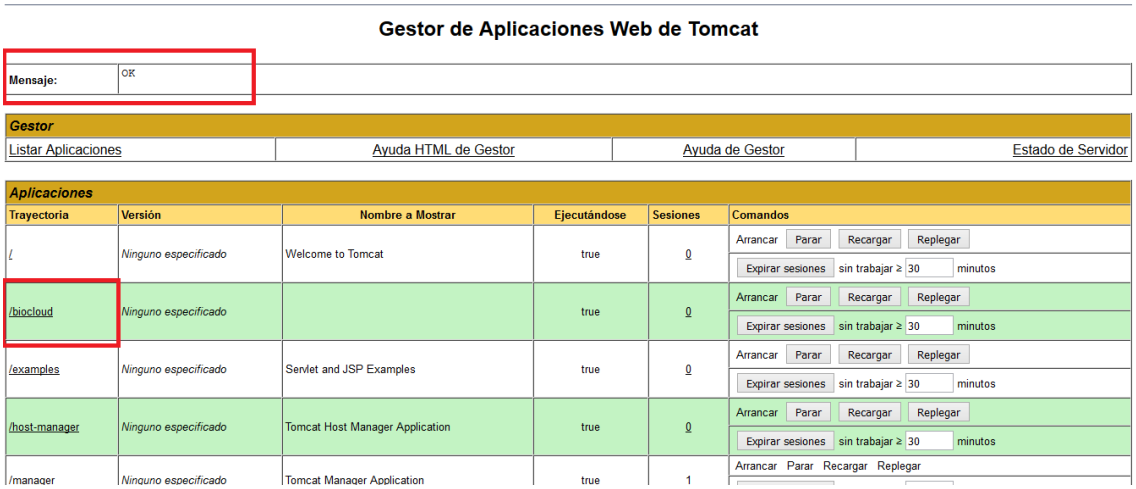



*Figura 9.16. Formulario de subida de archivos WAR*

Si todo ha ido correctamente, debería aparecer un mensaje de confirmación en la parte superior de la pantalla, y el nuevo proyecto desplegado, en la lista de aplicaciones (Figura 9.17).



**Gestor de Aplicaciones Web de Tomcat**

| Mensaje: | OK |
|---|---|

| Gestor | | | |
|---|---|---|---|
| Listar Aplicaciones | Ayuda HTML de Gestor | Ayuda de Gestor | Estado de Servidor |

**Aplicaciones**

| Trayectoria | Versión | Nombre a Mostrar | Ejecutándose | Sesiones | Comandos |
|---|---|---|---|---|---|
| / | Ninguno especificado | Welcome to Tomcat | true | 0 | Arrancar Parar Recargar Replegar<br>Expirar sesiones sin trabajar ≥ 30 minutos |
| /biocloud | Ninguno especificado | | true | 0 | Arrancar Parar Recargar Replegar<br>Expirar sesiones sin trabajar ≥ 30 minutos |
| /examples | Ninguno especificado | Servlet and JSP Examples | true | 0 | Arrancar Parar Recargar Replegar<br>Expirar sesiones sin trabajar ≥ 30 minutos |
| /host-manager | Ninguno especificado | Tomcat Host Manager Application | true | 0 | Arrancar Parar Recargar Replegar<br>Expirar sesiones sin trabajar ≥ 30 minutos |
| /manager | Ninguno especificado | Tomcat Manager Application | true | 1 | Arrancar Parar Recargar Replegar |

*Figura 9.17. Página Gestor de Aplicaciones Web*

## 9.2 ANEXO II. MANUAL DE USUARIO DE BIOCLOUD

Este anexo contiene un manual de uso de la aplicación web BioCloud. En él se explica detalladamente las funciones ofrecidas por dicha aplicación y su procedimiento de uso.

### 9.2.1 *Página de inicio*

Para acceder al servicio Web, ingresar la URL: http://localhost:8080/biocloud/, aparecerá la página de inicio mostrada en la Figura 9.18. Esta página de inicio está formada por varias imágenes que enlazan al sitio web de BLAST, al de Trinity y a un buscador de información online de gran utilidad en el ámbito científico llamado PubMed [23].


*Figura 9.18. Página de inicio del servicio Web*

En el lateral izquierdo de dicha pantalla, se encuentra el menú principal que presenta las distintas opciones de la aplicación (Figura 9.19). Para facilitar su uso, éstas se han dividido en 3 grupos:

- **BLAST**
  - **Alineamiento**: Permite realizar un análisis utilizando la herramienta BLAST.
  - **Nueva Base de datos**: Permite registrar en el sistema una nueva base de datos de BLAST.
- **Trinity:** Permite llevar a cabo un análisis utilizando la herramienta Trinity.
- **Listado de tareas**: Permite consultar las tareas que se han iniciado hasta el momento y el estado de las mismas.

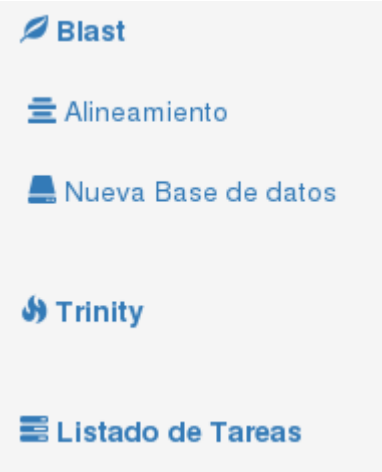



*Figura 9.19. Detalle del menú principal*

### 9.2.2 *Registrar una nueva base de datos*

Para poder llevar a cabo un análisis utilizando el programa BLAST, es necesario registrar previamente en el sistema las bases de datos contra las que se van a efectuar las búsquedas. Para ello, pulsar en la opción *Nueva Base de datos*. Aparecerá la pantalla mostrada en la Figura 9.20.

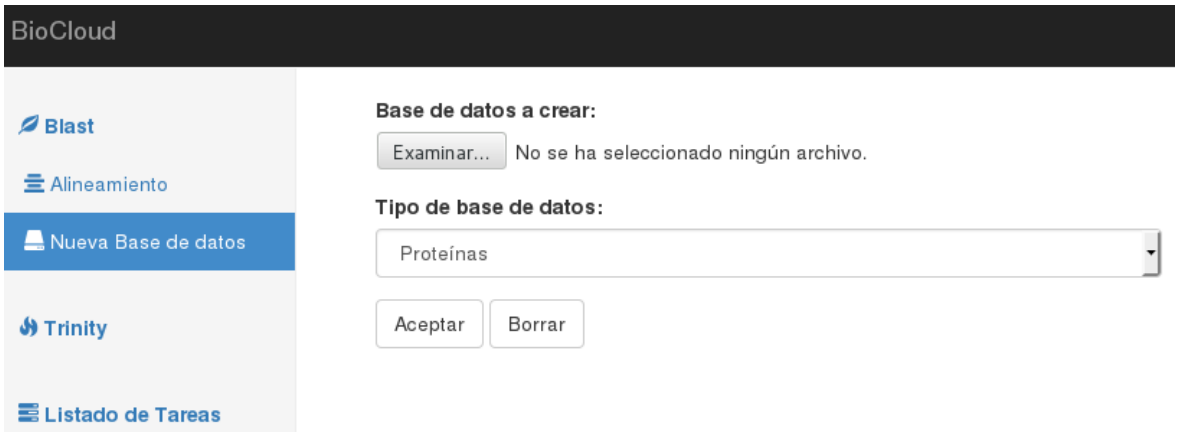



*Figura 9.20. Pantalla Nueva base de datos*

Como se puede observar en dicha figura, esta pantalla está formada por un formulario en el que se debe indicar la siguiente información:

- **Base de datos a crear:** Fichero a partir del cual se generará la base de datos.
- **Tipo de base de datos:** Si la base de datos es de nucleótidos o de proteínas (Figura 9.21).

Al pulsar sobre el botón Aceptar y si todo ha ido correctamente, se registrará la base de datos correspondiente en el sistema, estando disponible en futuros análisis.

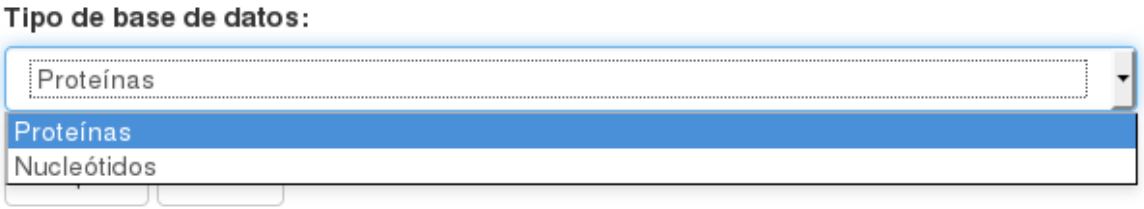



*Figura 9.21. Selección del tipo de base de datos*

En caso de que no se adjuntase ningún fichero, se mostraría el mensaje de error de la Figura 9.22.

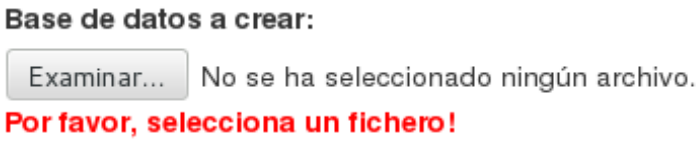



*Figura 9.22. Error por falta de fichero adjunto*

### 9.2.3 *Realizar un análisis con BLAST*

Para realizar un análisis utilizando la herramienta BLAST, pulsar sobre la opción *Alineamiento,* se mostrará la pantalla de la Figura 9.23.

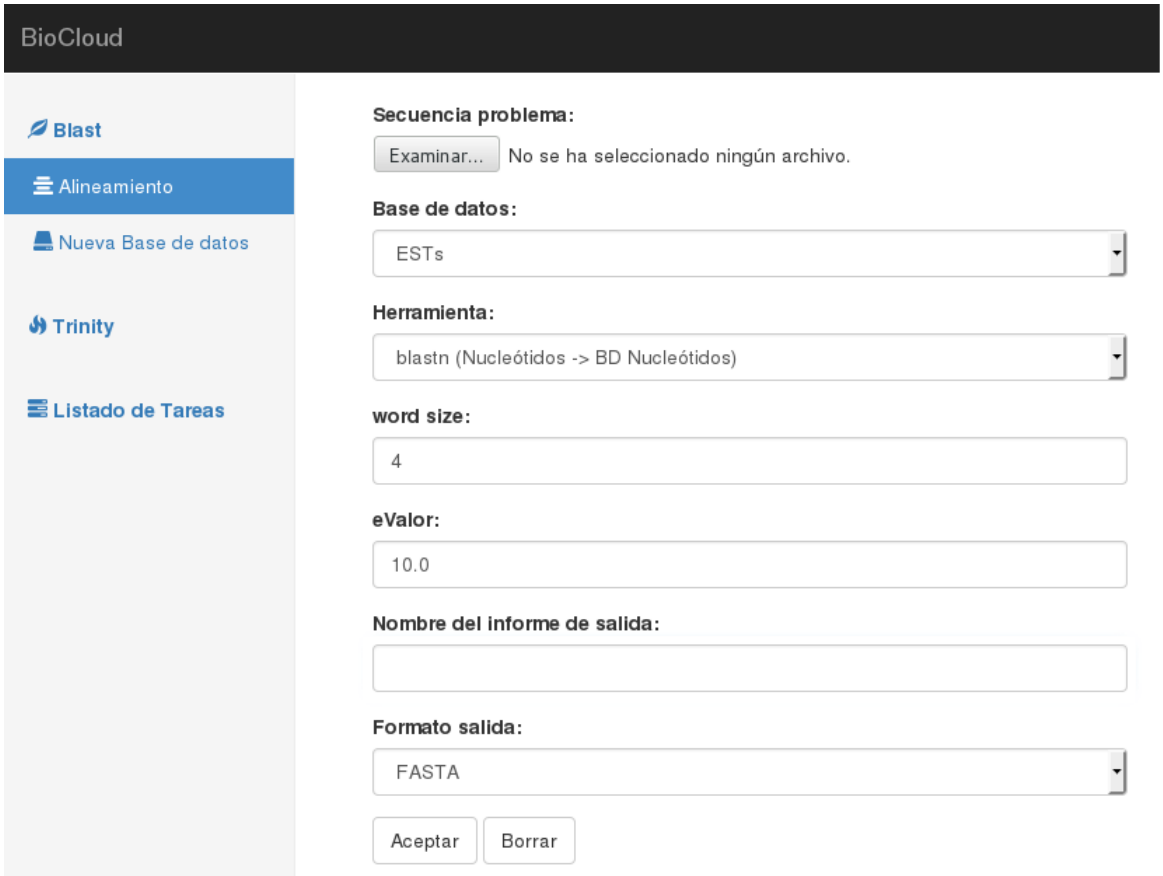



*Figura 9.23. Pantalla Alineamiento*

La información que hay que indicar en cada uno de los campos del formulario que compone dicha pantalla es la siguiente:

- **Secuencia problema:** Archivo que contiene la secuencia problema a analizar. El nombre de este fichero ser cualquier cadena, sin distinguir mayúsculas o minúsculas, que se componga como mínimo de un carácter alfanumérico (letras y números), guión medio, guión bajo o arroba, y opcionalmente esté seguido de una extensión compuesta del carácter punto y al menos una letra. Un ejemplo de nombre válido sería “nombre_fichero” y uno no válido, “nombre\\fichero” o “nombre1.nombre2.fichero.”
- **Base de datos:** Base de datos contra la que se desea realizar la búsqueda (Sólo se mostrarán aquellas que hayan sido registradas previamente en el sistema).
- **Herramienta:** Herramienta del conjunto de programas que componen BLAST: BLASTP, BLASTN… (Figura 9.24).
- **Word size:** Tamaño de la palabra que se buscará en la base de datos.
- **eValor:** Umbral que determina el grado de significación de los alineamientos producidos. Sólo se reportarán aquellos alineamientos que tengan un e-Valor inferior al especificado en este campo.
- **Nombre del informe de salida:** Nombre del informe de salida generado tras el análisis.
- **Formato de salida:** Formato de dicho informe de salida (FASTA, HTML o XML).

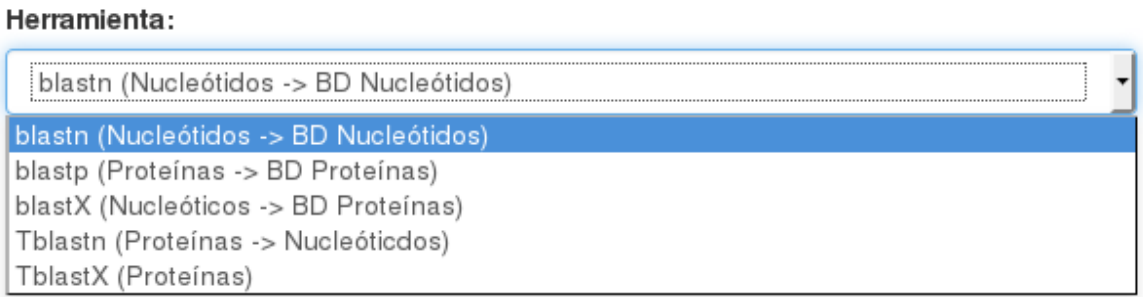



*Figura 9.24. Selección de la herramienta a utilizar*

Si alguno de los datos introducidos en el formulario no fuese válido, se mostrará un mensaje de error informando de ello bajo el campo correspondiente.

Estos mensajes pueden aparecer en alguna de las siguientes situaciones:

- Si no se ha adjuntado ningún fichero (Figura 9.25).

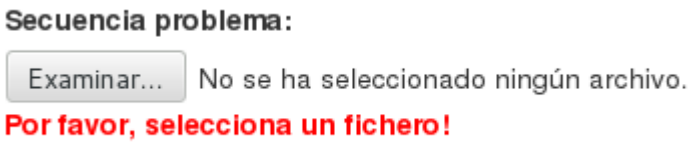



*Figura 9.25. Error por ausencia de fichero adjunto*

- Si valor de Word size es menor a 4 (Figura 9.26).

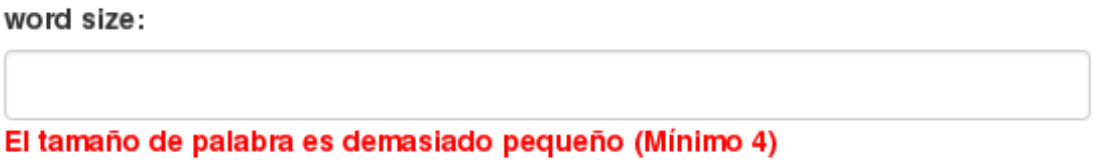



*Figura 9.26. Error por un valor de Word size excesivamente pequeño*

- Si el nombre del fichero de entrada o el del informe de salida contiene caracteres no permitidos como "..", "//" etc.


*Figura 9.27. Nombre de informe de salida erróneo*

- Si se inserta una cadena de caracteres en los campos Word size o e-Valor.

Una vez finalizado dicho análisis, el informe de salida resultante podrá descargarse a través del URL correspondiente disponible en la vista "*Listado de Tareas*".

### 9.2.4 *Realizar un análisis con Trinity*

Si se desea utilizar la herramienta Trinity para realizar un análisis, pulsar sobre la opción *Trinity*. Aparecerá la pantalla de la Figura 9.28.

En este caso, la información a especificar es la siguiente:

- **Left reads:** Lecturas a izquierdas (Sentido 3' → 5'). Si se adjunta algún archivo en este campo, se deberá incluir otro en el campo *Right reads*, ya que de lo contrario aparecerá un mensaje de error.
- **Right reads:** Lecturas a derechas (Sentido 5' → 3'). Al igual que se ha indicado anteriormente, si se adjunta algún archivo en este campo, se deberá incluir otro en el campo *Left reads*, en caso de no hacerlo se mostrará un mensaje de error.
- **Single reads:** Lecturas mono-hebra. En este caso es suficiente adjuntar un fichero únicamente en este campo.
- **Tipo de read:** El tipo de lectura que se va a analizar: FASTA o FASTQ.

- **Número de CPU:** Número de CPU a utilizar por Trinity, es decir, número de hebras que se ejecutarán en paralelo.
- **Máxima cantidad de memoria RAM reservada para Trinity**

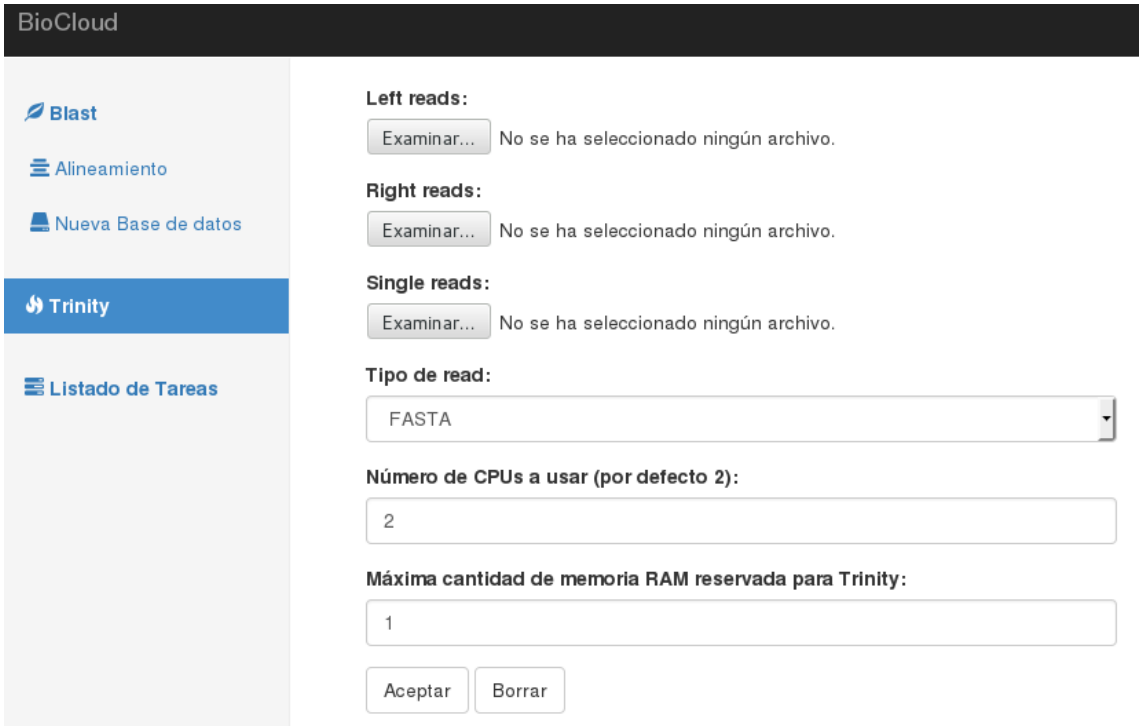



*Figura 9.28. Pantalla Trinity*

Si se introducen valores no válidos en el formulario, aparecerán una serie de mensajes de error:

- Si no se adjunta ningún fichero o si el nombre de los mismos no es válido (Figura 9.29 y Figura 9.30).

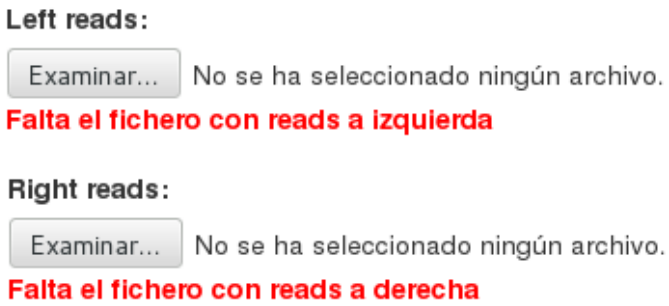



*Figura 9.29. Errores por falta de ficheros adjuntos para secuencias pareadas*

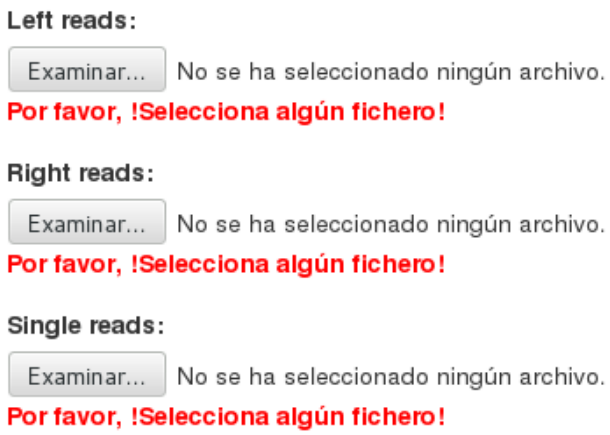



*Figura 9.30. Error por falta de ficheros adjuntos*

- Si se especifica un número de CPU inferior a 2 (Figura 9.31).


*Figura 9.31. Error por valor de CPU demasiado pequeño*

- Si la cantidad de memoria RAM reservada para Trinity es menor que 1.

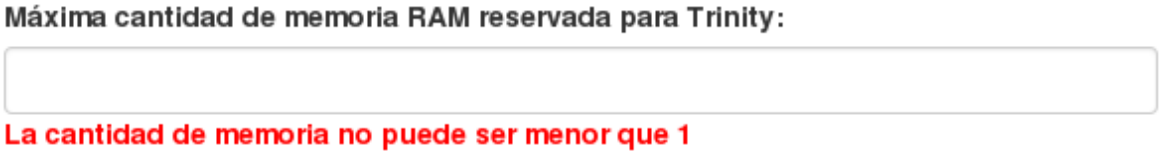



*Figura 9.32. Error por valor de memoria RAM incorrecto*

- Si se introduce una cadena de texto en los campos *Número de CPUs y Máxima cantidad de memoria RAM* (Figura 9.33).


*Figura 9.33. Error por inserción de un tipo de datos incorrectos*

Finalizado un análisis, el informe de salida generado puede descargarse desde la vista "*Listado de Tareas*", a través de su correspondiente URL. El nombre de este fichero tendrá el formato Trinity_out_#.Trinity.FASTA, donde # representa la fecha y la hora a la que se inició el análisis en formato dd_MM_yy-hh_mm_ss.

### 9.2.5 *Consultar listado de tareas*

Para comprobar el estado de una tarea pulsar sobre la opción *"Listado de Tareas",* aparecerá la pantalla de la Figura 9.34.

Cuando se inicia un análisis, éste se mostrará junto con su fecha de inicio y su estado será *Pendiente.* Si dicha ejecución ha finalizado correctamente, aparecerá además la fecha de finalización del mismo, se habilitará un enlace a través del cual se puede descargar el informe de salida generado y se actualizará su estado a *Finalizada*. Si han pasado 5 horas desde que se inició el análisis y éste no ha finalizado, se considerará que ha fallado y por tanto, se actualizará su estado a *Error* (Figura 9.35).


*Figura 9.34. Tareas iniciadas*

| Estado | URL descarga |
|---|---|
| Error | |
| Finalizada | Descargar Informe |
| Finalizada | Descargar Informe |
| Finalizada | Descargar Informe |
| Finalizada | Descargar Informe |
| Finalizada | Descargar Informe |
| Finalizada | Descargar Informe |
| Pendiente | |

*Figura 9.35. Detalle de los estados de las tareas*

## 9.3 ANEXO III. FICHEROS DE CONFIGURACION

En este anexo se incluyen los diferentes ficheros de configuración que se han utilizado en la implementación de la infraestructura cloud desplegada.

### 9.3.1 *Fichero de descripción de trabajo de Trinity*

El programa OGS/GE hace uso de ficheros de descripción de trabajos. El fichero de descripción utilizado en el contexto de este TFG es el que se muestra a continuación y cuya estructura se explica detalladamente en el apartado 9.7.2.3.

```
#!/bin/sh
#

################### PLANTILLA TRINITY ##########################
#####################################################################
#
#$ -N trinity.sh
#$ -S /bin/sh
#$ -V
#
#Se obtienen los parámetros
temp=$1
tipo_secuencia=$2
max_memory=$3
fichero_single=$4
fichero_izquierda=$5
fichero_derecha=$6
cpu=$7
#
directorio_trabajo="/home/sgeadmin/BioCloud/entrada/$temp"
#
# Se lanza la tarea
if [ -z "$fichero_single" ];
then
#
```

```
        /home/sgeadmin/BioCloud/trinityrnaseq-2.2.0/Trinity—seqType $tipo_secuencia—max_memory $max_memory—left $directorio_trabajo/$fichero_izquierda—right $directorio_trabajo/$fichero_derecha—CPU $cpu—output $directorio_trabajo/trinity_out_$temp—full_cleanup—grid_conf /home/sgeadmin/BioCloud/trinityrnaseq-2.2.0/hpc_conf/BroadInst_SGE.test.conf
        else
        /home/sgeadmin/BioCloud/trinityrnaseq-2.2.0/Trinity—seqType $tipo_secuencia—max_memory $max_memory—single $directorio_trabajo/$fichero_single—CPU $cpu—output $directorio_trabajo/trinity_out_$temp—full_cleanup—grid_conf /home/sgeadmin/BioCloud/trinityrnaseq-2.2.0/hpc_conf/BroadInst_SGE.test.conf
        fi
```

### 9.3.2 *Fichero de configuración de SGE*

En el caso del programa Trinity, para indicar que se desea ejecutar tareas de manera distribuida en un sistema grid, se utiliza el parámetro –grid_conf seguido del nombre del fichero de configuración a utilizar. En este TFG, el fichero de configuración empleado es el que se muestra más abajo. En él debe especificarse el número máximo de recursos computacionales que pueden realizar tareas (500), así como el número de tareas que pueden ser asignadas por lotes, es decir, en modo batch a un recurso computacional individual (100).

```
        # grid type:
        grid=SGE
        # template for a grid submission
        cmd=qsub -V -cwd

        # note -e error.file -o out.file are set internally, so dont set them in the above cmd.

        # settings below configure the Trinity job submission system, not tied to the grid itself.

        # number of grid submissions to be maintained at steady state by the Trinity submission system
        max_nodes=500
        # number of commands that are batched into a single grid submission job.
        cmds_per_node=100
```

### 9.3.3 *Plantilla de configuración de las máquinas virtuales*

Para crear las máquinas virtuales que han actuado como recursos computacionales en este TFG, se han empleado las siguientes plantillas de configuración.

#### *9.3.3.1 Escenario Homogéneo*

La plantilla empleada para instanciar las 4 máquinas virtuales que empleados como recursos computacionales en un entorno homogéneo es la siguiente:

```
CONTEXT=[NETWORK=”YES”,SSH_PUBLIC_KEY=”ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDAg5dpC4/jkqhC0sRFRbXZ4W2+5ROfw+F
LgMJKFwNa3DCwnDtCSkiC/uRpHrAI+KwOBZzJMdUd5i/1TgKCjUe74p+MmM5PwxQ6I
SSP4tpXuDc8K3F6KE5UQBOCpDn/wLOYeA1YHgZfy3i6ppjgHakLcf1F70xwHOUevw/a
h9x6a9z2llBynkBK+/02Kqiefu+pBchauPDWxaogZJc2dFqkg9r5k0OGQb86ViGWZ6dVF
9Q/Ss/yd2KgkwIJZ8J9mzXS5Y0pfRG0hG2hWB7JG1/RWF7k2UWOy1nZkvA4ADMktbr
NCWqXiYCiT36OHdQkg4JDO17+PAWLyIGCQAyRVgdF root@servidor”]
CPU=”1”
DISK=[IMAGE=”Scientific Linux”,IMAGE_UNAME=”oneadmin”]
GRAPHICS=[KEYMAP=”es”,LISTEN=”0.0.0.0”,TYPE=”VNC”]
HYPERVISOR=”kvm”
MEMORY=”4096”
NIC=[NETWORK=”Laboratorio”,NETWORK_UNAME=”oneadmin”]
OS=[ARCH=”x86_64”]
SUNSTONE_CAPACITY_SELECT=”YES”
SUNSTONE_NETWORK_SELECT=”YES”
VCPU=”4”
```

#### *9.3.3.2 Escenario Heterogéneo*

En este caso se han empleado dos plantillas diferentes: la que se ha indicado en el apartado anterior y la que se muestra a continuación:

```
CONTEXT=[NETWORK=”YES”,SSH_PUBLIC_KEY=”ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDAg5dpC4/jkqhC0sRFRbXZ4W2+5ROfw+F
LgMJKFwNa3DCwnDtCSkiC/uRpHrAI+KwOBZzJMdUd5i/1TgKCjUe74p+MmM5PwxQ6I
SSP4tpXuDc8K3F6KE5UQBOCpDn/wLOYeA1YHgZfy3i6ppjgHakLcf1F70xwHOUevw/a
h9x6a9z2IlBynkBK+/02Kqiefu+pBchauPDWxaogZJc2dFqkg9r5k0OGQb86ViGWZ6dVF
9Q/Ss/yd2KgkwIJZ8J9mzXS5Y0pfRG0hG2hWB7JG1/RWF7k2UWOy1nZkvA4ADMktbr
NCWqXiYCiT36OHdQkg4JDO17+PAWLyIGCQAyRVgdF root@servidor”]
CPU=”2”
DISK=[IMAGE=”Scientific Linux”,IMAGE_UNAME=”oneadmin”]
GRAPHICS=[KEYMAP=”es”,LISTEN=”0.0.0.0”,TYPE=”VNC”]
HYPERVISOR=”kvm”
MEMORY=”8192”
NIC=[NETWORK=”Laboratorio”,NETWORK_UNAME=”oneadmin”]
OS=[ARCH=”x86_64”]
SUNSTONE_CAPACITY_SELECT=”YES”
SUNSTONE_NETWORK_SELECT=”YES”
VCPU=”8”
```

## 9.4 ANEXO IV. CONCEPTOS BÁSICOS DE BIOLOGÍA

En las siguientes páginas se explican algunos conceptos básicos relacionados con Biología. Aunque si bien es cierto que su lectura no es esencial para la comprensión del funcionamiento de la infraestructura desplegada, se ha creído oportuno redactar este anexo para aquellos lectores que deseen adentrarse en el mundo de la Biología.

Antes de comenzar con la explicación, se plantea al lector la siguiente cuestión: Si nos paramos a pensar en la enorme cantidad de especies que habitan la Tierra, podemos notar que en nada se parecen un elefante a una serpiente o un ratón al ser humano. Sin embargo, existen bastantes similitudes en determinadas regiones del material genético de unos y otros. ¿A qué se debe esta semejanza? Para poder responder a esta pregunta es necesario introducir algunos conceptos básicos sobre Biología Molecular y Biología Evolutiva.

Planteada esta cuestión, se anima al lector a disfrutar de estas páginas tal y como su autora ha hecho durante su redacción.

### 9.4.1 *Secuencias biológicas*

La estructura y función de las diferencias secuencias biológicas que existen: ADN, ARN y proteínas serán presentados en las siguientes páginas.

#### *9.4.1.1 ADN*

El ácido desoxirribonucleico (ADN) es un polímero lineal constituido por dos largas cadenas de nucleótidos, llamadas hebras, unidas ente sí formando una doble hélice. Existen 4 nucleótidos: Adenina (A), Guanina (G), Citosina (C) y Timina (T), cada uno compuesto por una base nitrogenada distinta, a la que deben su nombre. Estas bases nitrogenadas se unen entre sí mediante enlaces de hidrógeno de tal manera que la A sólo se puede emparejar con la T, y la G con la C.

La molécula de ADN posee una dirección determinada por los enlaces establecidos entre los azúcares y los grupos fosfato, confiriendo un sentido a la lectura de la información. Un extremo de la cadena lleva un grupo fosfato libre unido al carbono 5’ (Extremo 5’). El otro extremo, tiene un grupo hidroxilo libre (-OH) en el carbono 3’ (Extremo 3’). Así, cuando dos cadenas de ADN se unen mediante puentes de hidrógeno, ambas están enfrentadas, pero en direcciones opuestas, es decir, son cadenas antiparalelas (5’ a 3’ y 3’ a 5’).

Esta estructura de doble hélice proporciona mayor estabilidad al ADN y permite la corrección de errores en el caso de que alguna base sea dañada accidentalmente por ejemplo por exposición a luz ultravioleta.

#### *9.4.1.2 ARN*

El ácido ribonucleico (ARN) se obtiene a partir del ADN en un proceso denominado **transcripción**. Se trata de un ácido nucleico cuya estructura química es bastante parecida a la del ADN, puesto que es una copia complementaria del mismo, salvo que, en lugar de usar T, emplea Uracilo (U), siendo además de cadena sencilla en lugar de cadena doble. También, al igual que sucede con el ADN, la molécula de ARN tiene una dirección, en sentido 5' → 3'.

Existen diferentes tipos de ARN:

- **ARN mensajero (ARNm)**: Contiene las instrucciones para la síntesis de proteínas.
- **ARN ribosómico (ARNr)**: Constituye el ribosoma, el lugar de la célula donde se realiza la síntesis de proteínas.
- **ARN de transferencia (ARNt):** Actúa como portador de aminoácidos al ribosoma.

#### *9.4.1.3 Proteínas*

Las proteínas son macromoléculas compuestas de aminoácidos. Puede tener una o varias cadenas presentando una determinada estructura, íntimamente relacionada con su función biológica. De tal modo que la alteración de esta estructura supondría la pérdida de su función.

La secuencia de nucleótidos en el ARN puede ser vista como un alfabeto molecular de cuatro señales, los cuatro nucleótidos distintos que tiene esta molécula (U, A, G, C). Sin embargo, el alfabeto molecular de las proteínas contiene 20 señales distintas, es decir, hay 20 aminoácidos diferentes que constituyen las proteínas. Así, cada grupo de tres nucleótidos, lo que se conoce como **codón**, representa un aminoácido.

De los 64 codones posibles ($\mathbf{4^3}$), 61 representan aminoácidos, aunque sólo hay 20 aminoácidos diferentes, los otros tres son señales de terminación (codones de parada o *stop*), de modo que un mismo aminoácido puede estar representado por diferentes codones. Por esta razón, se dice que el material genético está *degenerado*.

Aunque no hay señal de inicio propiamente dicha, la mayoría de las proteínas comienzan por el aminoácido metionina, por lo que su codón AUG, viene a ser una señal de iniciación.

La relación entre codones y aminoácidos, más conocido como **código genético,** se muestra en la Tabla 9.1.

| Primera Posición | Segunda Posición | | | | Tercera Posición |
|---|---|---|---|---|---|
| | U | C | A | G | |
| U | fenilalanina | serina | tirosina | cisteína | U |
| | fenilalanina | serina | tirosina | cisteína | C |
| | leucina | serina | STOP | STOP | A |
| | leucina | serina | STOP | triptófano | G |
| C | leucina | prolina | histidina | arginina | U |
| | leucina | prolina | histidina | arginina | C |
| | leucina | prolina | glutamina | arginina | A |
| | leucina | prolina | glutamina | arginina | G |
| A | isoleucina | treonina | asparagina | serina | U |
| | isoleucina | treonina | asparagina | serina | C |
| | isoleucina | treonina | lisina | arginina | A |
| | metionina | treonina | lisina | arginina | G |
| G | valina | alanina | aspártico | glicina | U |
| | valina | alanina | aspártico | glicina | C |
| | valina | alanina | glutámico | glicina | A |
| | valina | alanina | glutámico | glicina | G |

*Tabla 9.1. El código genético [FUENTE: [24]]*

#### *9.4.1.4 El flujo de la información genética*

La información contenida en el ADN determina los tipos de proteínas que se sintetizan en las células. En el caso de los organismos eucariotas, de los cuales se hablará más adelante, esta información se encuentra siempre en el núcleo de la célula; sin embargo, la síntesis de proteínas tiene lugar fuera del núcleo, en los ribosomas ubicados en el citoplasma de la célula, por tanto, debe existir una molécula intermediaria que actúe de <<mensajero>>, llevando las instrucciones necesarias para que se pueda realizar esta síntesis de proteínas. Esta molécula es el ARN.

Así, la expresión de la información genética consta de varias etapas: De ADN a ARN y de ARN a proteínas.

La primera etapa, el paso de ADN a ARN se denomina **transcripción**. Mientras que la segunda, el paso de ARN a Proteínas, **traducción**.

Otro proceso de vital importancia es el conocido como **replicación**, en el cual se genera una copia idéntica de la molécula de ADN durante la división celular. En este proceso, las dos hebras del ADN se separan de modo que cada una pueda dirigir la síntesis de una hebra hija complementaria, lo que da como resultado dos dobles hélices completas permitiéndose así el paso de información genética entre generaciones [25].

### 9.4.2 *Evolución*

Se pueden producir cambios en el ADN como consecuencia de tres fuerzas diferentes: mutación, selección natural y deriva genética [26].

Aunque en principio se pueda creer que estos cambios pueden desencadenar algún estado patológico, no siempre es así; siendo en ocasiones responsables de un proceso evolutivo al generarse productos más efectivos que suponen una importante ventaja para el organismo que los lleva, e incluso, originar nuevas especies.

#### *9.4.2.1 Mutaciones*

Las mutaciones son cambios en el material genético. Muchos compuestos químicos a los que se encuentran expuestos los seres vivos y determinadas circunstancias como errores durante la replicación del ADN en el proceso de división celular, pueden producir cambios en el mismo. Estas mutaciones pueden ser de varios tipos [24]:

- Sustitución de un nucleótido por otro.
- Pérdida o delección de uno o varios nucleótidos.
- Inserción de uno o más nucleótidos.

Por un lado, una sustitución dará como resultado en la mayoría de los casos un producto génico alterado, es decir, una proteína alterada. Sin embargo, dado que hay varios codones que representan a un mismo aminoácido, la secuencia de aminoácidos podría no cambiar. A estas mutaciones se les llaman *silenciosas*. Por el contrario, aquellas mutaciones que cambian un aminoácido por otro se denominan *de sentido erróneo* y finalmente, las que generan un codón de parada se conocen como mutaciones *sin sentido*.

Por otro lado, las inserciones y delecciones pueden tener efectos más destructivos ya que la pérdida o inserción de un único nucleótido hace que cambie la pauta de lectura y se modifiquen todos los codones sucesivos, produciéndose como consecuencia de ello una proteína completamente distinta a la original y generalmente disfuncional.

#### 9.4.2.2 Selección natural

Esta teoría fue formulada por el naturalista Charles Darwin para explicar la evolución de las especies. Los individuos de una especie se reproducirán en unas condiciones ambientales, que obstaculizarán o beneficiarán este proceso reproductivo según las capacidades de los mismos.

Tal y como se ha comentado anteriormente, los cambios en el ADN no tienen por qué ser perjudiciales, si no que pueden dar lugar a una variabilidad genética que dé lugar a individuos mejor adaptados, es decir, con una ventaja evolutiva, teniendo, por tanto, más probabilidades para sobrevivir y de perpetuar la especie.

Esta idea se ilustra con el siguiente ejemplo: ¿Por qué las jirafas tienen los cuellos tan largos? Porque aquellas jirafas cuyo cuello era más largo eran capaces de llegar a las hojas más altas y por lo tanto estaban mejor adaptadas al entorno viéndose incrementadas sus probabilidades de supervivencia.

#### 9.4.2.3 Deriva genética

Antes de comenzar a hablar de la deriva genética es necesario definir el concepto de alelo. Un alelo es cada una de las formas alternativas que presenta un gen. Los organismos diploides, como los mamíferos, tendrán dos alelos de cada gen, uno procedente del padre y otro, de la madre.

La deriva genética o génica es una fuerza evolutiva que actúa junto con la selección natural cambiando las características de las especies en el tiempo. Es un cambio aleatorio en la frecuencia de alelos de una generación a otra, tendiéndose a la pérdida de los menos frecuentes y la fijación de los más abundantes, causando una disminución en la variabilidad genética de la población [27].

Los efectos de la deriva se acentúan en poblaciones de tamaño pequeño y resultan en cambios que no son necesariamente adaptativos.

#### 9.4.2.4 Teoría Neutral de Evolución

Las mutaciones, de las que ya se ha hablado anteriormente, son completamente aleatorias y pueden producirse en cualquier parte del material genético. Se ha observado que, aunque a lo largo del tiempo pueden acumularse mutaciones en determinadas regiones del ADN, hay otras que prácticamente no han sufrido alteración alguna, lo cual indica que éstas se encuentran bajo presión selectiva. Es decir, algunas de estas regiones pueden ser vitales para la supervivencia del organismo, de tal manera que cualquier

variación en las mismas causaría la muerte del organismo. Se dice que estas regiones están sometidas a una intensa presión selectiva.

#### *9.4.2.5 Homología, Paralogía y Ortología*

Todos los seres vivos tienen un ancestro o antepasado común, no obstante, hay especies que se parecen más entre sí que otras. El estudio de las relaciones evolutivas entre organismos se denomina Filogenia.

Por definición, dos secuencias o genes son *homólogos* si comparten un ancestro común. Este concepto puede afinarse un poco más introduciendo dos nuevos términos: *ortología* y *paralogía*.

Dos genes o secuencias homólogos separados por un proceso de especiación, es decir, por la aparición de nuevas especies, se denominan *ortólogos*. De modo que cada copia del mismo gen o de la misma secuencia que existe en cada especie se dice que son ortólogas. El grado de homología que presentan estas copias será mayor si las especies se han separado recientemente.

En cambio, copias generadas por duplicación se conocen como *parálogas*. O lo que es lo mismo, si un gen de un organismo se duplica ocupando posiciones distintas en un mismo genoma, cada una de estas copias son parálogas.

#### *9.4.2.6 El Árbol de la Vida*

Desde el punto de vista clásico, existen cinco reinos taxonómicos: Animales, Plantas, Hongos, Moneras y Protistas. Sin embargo, la biología Molecular establece una nueva clasificación de los organismos basándose en el contenido genético y no en características externas.

En la Figura 9.36 se muestra un árbol filogenético basado en la secuencia de ARN ribosómico. Existen tres dominios claramente diferenciados que Carl Woese denominó **Bacterias** (*Bacteria*), **Arqueas** (*Archaea*) y **Eucariotas** (*Eucarya*). En base al genoma y a la estructura celular, existen dos divisiones principales: los **procariotas** (Bacterias y Arqueas) y los **eucariotas**.

Los procariotas carecen de núcleo celular por lo que todos los procesos relacionados con el ADN (replicación, transcripción y traducción) tienen lugar en el único compartimento que existe en su interior. Uno de los procariotas más famoso y ampliamente utilizado en el ámbito de la microbiología es *Escherichia coli*, una bacteria que coloniza el tracto gastrointestinal.

Por su parte, los eucariotas pueden tener múltiples formas y tamaños, existiendo desde organismos compuestos por una única célula como es el caso de *Saccharomyces cereviseae*, la levadura empleada en la fabricación de cerveza, hasta aquellos organismos multi-celulares como los mamíferos. Las células eucariotas poseen un núcleo en el cual se encuentra ubicado el ADN, y unos orgánulos, como las mitocondrias, encargadas de suministrar la mayor parte de la energía necesaria para la actividad celular, o los cloroplastos, responsables de la fotosíntesis en las plantas. Estos orgánulos tienen su propio genoma y su origen se debe al establecimiento de relaciones de simbiosis entre bacterias y eucariotas en épocas pasadas.

En esta clasificación no se ha incluido a los virus, puesto que éstos no se consideran organismos vivos ya que no respiran, no se mueven y no crecen, características que se asocian con los seres vivos.

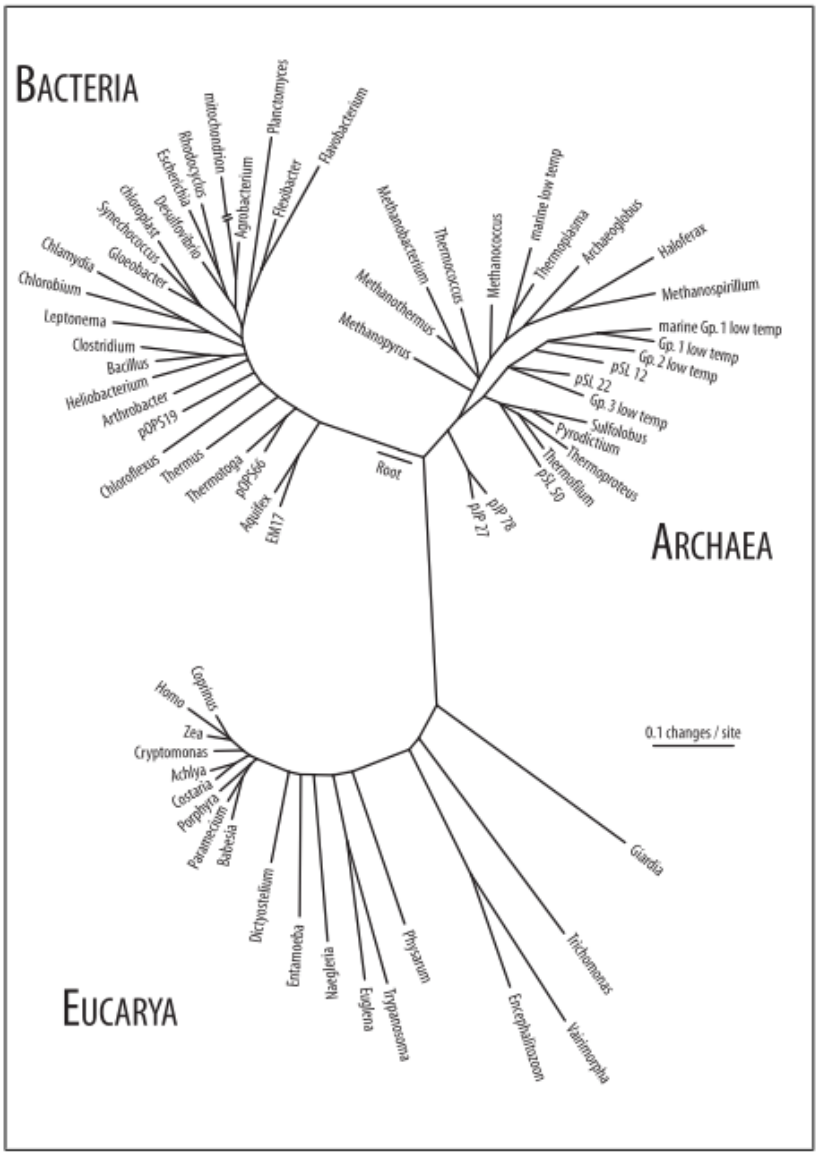



*Figura 9.36. Árbol filogenético basado en el ARN ribosómico [FUENTE: [26]]*

### 9.4.3 *Conclusión*

Retomando la incógnita que se planteó al inicio del capítulo de cómo a pesar de la gran diversidad de especies que habitan en la Tierra, existen algunas similitudes en la secuencia de determinadas regiones del material genético, y después de todo lo anteriormente explicado, se está en condiciones de resolver dicha cuestión y es que estas similitudes se deben a que estas regiones tuvieron un origen común y en algún momento de la historia, siguieron caminos diferentes.

## 9.5 ANEXO V. BLAST

En este anexo se realiza una reseña de la herramienta bioinformática BLAST: fundamento teórico, algoritmo de funcionamiento, programas que lo componen, informe de salida, etc.

### 9.5.1 *¿Qué es BLAST?*

*Basic Local Alignment Search Tool* (BLAST) [26], es uno de los programas más usados para la realización de búsquedas de secuencias biológicas. Desarrollado por el *National Center for Biotechnology Information* (NCBI), está compuesto por una familia de herramientas informáticas de alineamiento de secuencias que realizan búsquedas en bases de datos para encontrar similitudes estadísticamente significativas con una secuencia problema objeto de estudio.

### 9.5.2 *Programas que componen BLAST*

BLAST está formado por un conjunto de programas, cada uno de ellos destinado al análisis de un tipo de muestra, empleando para ello una base de datos distinta (Tabla 9.2).

| Programa | Base de Datos (Target) | Secuencia Problema (Query) | Uso |
|---|---|---|---|
| **BLASTN** | Nucleótidos | Nucleótidos | Comparar una secuencia de nucleótidos contra una base de datos de la misma naturaleza |
| **BLASTP** | Proteínas | Proteínas | Comparar una secuencia de proteínas contra una base de datos del mismo tipo |
| **BLASTX** | Proteínas | Nucleótidos traducidos a proteínas | Comparar una secuencia de nucleótidos traducidos a proteínas contra una base de datos de proteínas |
| **TBLASTN** | Nucleótidos traducidos a proteínas | Proteínas | Comparar una secuencia proteica con una base de datos de nucleótidos traducidos a proteínas |
| **TBLASTX** | Nucleótidos traducidos a proteínas | Nucleótidos traducidos a proteínas | Compara una secuencia de nucleótidos contra una base de datos de nucleótidos, ambos traducidos a proteínas |

*Tabla 9.2. Programas de BLAST [FUENTE: [26]]*

### 9.5.3 *Algoritmo de BLAST*

BLAST hace uso del algoritmo **Smith-Waterman** para realizar sus alineamientos. Se trata de un algoritmo heurístico por lo que la solución encontrada no será la más correcta; sin embargo, BLAST calcula la significación estadística de sus resultados permitiendo al analista juzgar la calidad de los mismos.

El algoritmo de BLAST consta de tres etapas diferentes: ensemillado, extensión y evaluación. A continuación, se describen brevemente cada una de ellas.

#### *9.5.3.1 Primera Etapa: Ensemillado*

En esta primera etapa, BLAST intenta localizar alineamientos potenciales (**ocurrencias**) entre la secuencia problema y la base de datos. Sólo aquellas regiones con ocurrencias serán usadas como semillas en la segunda etapa.

Un concepto relevante en la terminología de los programas BLAST es el de **vecindario** [28]. El vecindario de una palabra, es decir, de una secuencia de nucleótidos o aminoácidos, está formado por la propia palabra y todas las palabras cuya puntuación sea, al menos, tan grande como un umbral predefinido llamado **Threshold**, T. Ajustando el valor de dicho parámetro se puede controlar el tamaño del vecindario, así como el número de ocurrencias en el espacio de búsqueda, de modo que a mayor valor de T menor será el espacio de búsqueda.

Otra variable que también controla el número de ocurrencias es el tamaño de palabra, **Word size**, W. Cuanto más pequeño sea el valor de W mayor será la sensibilidad de la búsqueda; por el contrario, tamaños de palabras mayores harán que la búsqueda sea más rápida pero menos precisa.

#### *9.5.3.2 Segunda Etapa: Extensión*

Concluida la etapa anterior, se generan los alineamientos a partir de cada semilla individual. En la Figura 9.37, estos alineamientos pueden ser vistos como flechas que se extienden hacia ambos lados de las semillas, es decir, de las palabras. Mientras que en el algoritmo de Smith-Waterman los extremos finales de cada alineamiento se determinan una vez que ha sido evaluado todo el espacio de búsqueda, BLAST sólo realiza la búsqueda en una determinada región del espacio, por lo que deben existir mecanismos que permitan saber cuándo debe detenerse este proceso de extensión.

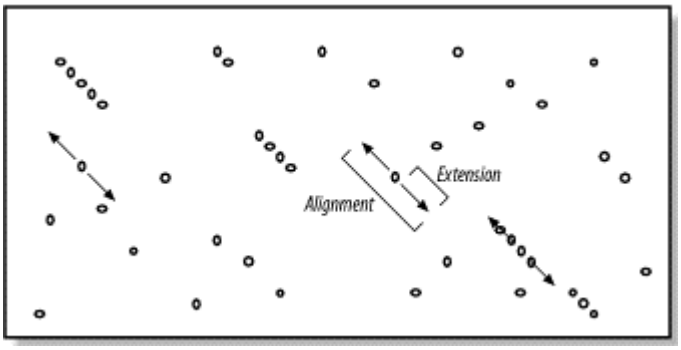



*Figura 9.37. Generación de alineamientos mediante la extensión de semillas [FUENTE: [26]]*

Para explicar este criterio de parada se mostrará un ejemplo en el que las ocurrencias se puntuarán con +1 y las diferencias, con -1 (en este caso no se tendrán en cuenta ni los espacios ni los huecos en las palabras) [26]. Así, suponiendo que se dispone de los siguientes enunciados en los que la semilla es la letra J y la extensión sólo ha tenido lugar hacia la derecha:

Juan Adell se fue a nadar al mar

Juan Adión no oyó a Borja en casa

Si el proceso de extensión finalizara cuando se encontrase la primera diferencia, se perderían algunos alineamientos que podría tener lugar más adelante; sin embargo, dado que los finales de estos enunciados soy muy dispares, no tendría sentido alguno extender hasta el final de los mismos. Para controlar esta extensión se utiliza una variable llamada X que almacena un histórico de los alineamientos más recientes producidos. Más concretamente, indica las diferencias permitidas desde el último máximo alcanzado. Por ejemplo, si X fuese igual a 5:

```
J u a n   A d e l l   s e   f u e   a   n a d a r   a l   m a r
J u a n   A d i ó n   n o   o y ó   a   B o r j a   e n   c a s a
1 2 3 4   5 6 5 4 3   2 1 ← Puntuaciones
0 0 0 0   0 0 1 2 3   4 5 ← Penalizaciones
```

La máxima puntuación es igual a 6 y dado que X=5, la extensión finaliza cuando esta puntuación disminuya hasta 1.

Uno de los problemas que presenta esta estrategia es que el algoritmo puede quedar atrapado en un óptimo local. Debido a ello, la definición de esta variable X es determinante en el resultado del análisis.

#### 9.5.3.3 *Tercera Etapa: Evaluación*

Una vez que las semillas se han extendido en ambas direcciones, los alineamientos producidos son evaluados para determinar si son estadísticamente significativos. Aquellos que lo son, denominados pares de alta puntuación (*High Score Pairs* o HSPs), se reportan en el informe de salida, mientras que los considerados como inconsistentes, es decir, aquellos en los que una misma parte de la secuencia problema se ha alineado con diferentes regiones de una misma secuencia de la base de datos son eliminados por el programa.

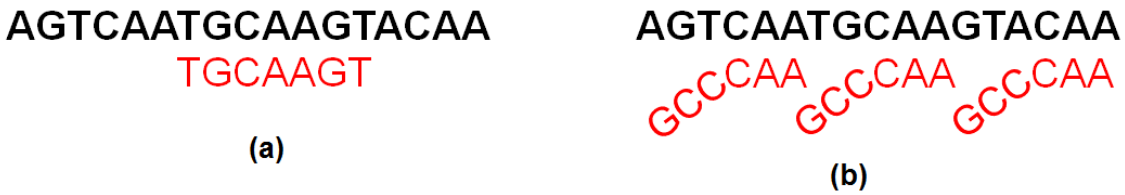



*Figura 9.38. (a) Par de alta puntuación (b) Alineamiento inconsistente*

La puntuación conseguida por cada alineamiento es almacenada en una variable llamada **Score**, S, mientras que su significación estadística se determina a partir de la probabilidad de haber sido obtenido por azar. Sólo se reportarán los alineamientos que hayan obtenido una probabilidad inferior a un parámetro llamado **e-Valor**, E. Este parámetro permite definir los alineamientos que se desean obtener en base a su significación estadística. De manera que cuanto menor sea el valor de E, más significativo será un alineamiento y su puntuación.

### 9.5.4 *Ficheros de entrada*

BLAST trabaja con ficheros FASTA tanto para la entrada como para la salida. Este formato es el estándar utilizado para representar secuencias tanto de ácidos nucleicos como de proteínas y consta de dos partes: una línea de definición y una o más líneas de secuencia [26].

La línea de definición es una única línea que comienza obligatoriamente por el símbolo '>' seguido de un identificador y una descripción de la secuencia. No debe existir ningún espacio entre el símbolo '>' y el identificador ni dentro del propio identificador, puesto que el espacio es utilizado como delimitador. La descripción es un texto de formato libre que puede contener cualquier carácter excepto el de fin de línea. En la Figura 9.39 se ofrece un ejemplo de línea de definición.

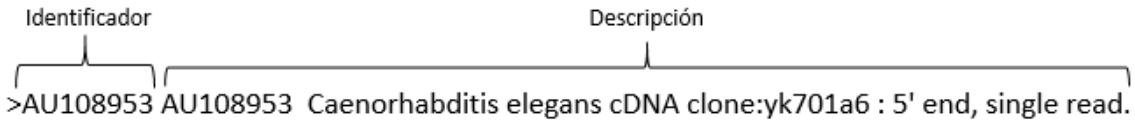



*Figura 9.39. Estructura de la línea de definición*

Por su parte, las líneas de secuencia presentan un formato muy simple: pueden tener cualquier longitud y puede existir cualquier número de líneas. Algunos programas requieren que las líneas no sean excesivamente largas, por lo que se suele adoptar como convención que cada línea contenga entre 50 y 80 caracteres [26]. Un ejemplo estas líneas de secuencia se puede observar la Figura 9.40.

```
tggcctactgganaaaacaacaatgcgtgctttactattcacctctgttgttcttttggctttggctttgttgaggcaaagaagcag
actatcactgtcaagggtacaactatttgtaataagaagagaattcaggcggaggttacccttgggagaaagatactctcgaccc
cgatgacaagctcgcctcaatgcaatcgaacaaagaaggagagttctcacttaccggatccgacgacgagatcacctcaatctct
ccatacctcataatcacccacaacagcaacgtgaagaaggccggatgaagccgtgtttcagag
```

*Figura 9.40. Líneas de secuencia*

Estas secuencias se representan empleando unos códigos estándar llamados IUB e IUPAP, para aminoácidos y ácidos nucleicos respectivamente. Los códigos de ácidos nucleicos soportados son los que se relacionan en la Tabla 9.3.

| Código | Significado |
|---|---|
| **A** | Adenosina |
| **T** | Timina |
| **K** | G/T (Cetona) |
| **M** | A/C (Grupo Amino) |
| **B** | G/T/C |
| **V** | G/C/A |
| **C** | Citosina |
| **N** | A/G/C/T (Cualquiera) |
| **S** | G/C (Interacción Fuerte) |
| **W** | A/T (Interacción Débil) |
| **D** | G/A/T |
| **-** | Hueco de longitud indeterminada |
| **G** | Guanina |
| **U** | Uracilo |
| **Y** | T/C (Pirimidina) |
| **R** | G/A (Purina) |
| **H** | A/C/T |

*Tabla 9.3. Códigos de ácidos nucleicos soportados [FUENTE: [29]]*

En cambio, para aquellos programas que trabajen con secuencias de aminoácidos, los códigos admitidos son los que se muestran en la Tabla 9.4.

| Código | Significado |
|---|---|
| **A** | Alanina |
| **B** | Aspartato/Asparagina |
| **C** | Cisteína |
| **D** | Aspartato |
| **E** | Glutamato |
| **F** | Fenilalanina |
| **G** | Glicina |
| **H** | Histidina |
| **I** | Isoleucina |
| **K** | Lisina |
| **L** | Leucina |
| **M** | Metionina |
| **N** | Asparagina |
| **P** | Prolina |
| **Q** | Glutamina |
| **R** | Arginina |
| **S** | Serina |
| **T** | Treonina |
| **U** | Selenocisteína |
| **V** | Valina |
| **W** | Triptófano |
| **Y** | Tirosina |
| **Z** | Glutamato/Glutamina |
| **X** | Cualquiera |
| ***** | Parada de traducción |
| **-** | Hueco de longitud indeterminada |

*Tabla 9.4. Códigos de aminoácidos soportados [FUENTE: [29]]*

### 9.5.5 *Informe de salida de BLAST*

La estructura básica de un informe de salida de BLAST consta de cuatro partes: cabecera, resumen, alineamientos producidos y pie de informe [26].

#### *9.5.5.1 Cabecera*

Tal y como se observa en la Figura 9.41, la primera línea contiene el nombre del programa y la versión del mismo. A continuación, se cita un artículo científico que debería referenciado en cualquier trabajo en el que se haga uso de NCBI-BLAST. Finalmente, se muestra una descripción de la base de datos contra la que se ha efectuado la búsqueda, se indica el tamaño de la misma y se realiza una definición de la query, es decir, de la secuencia problema.

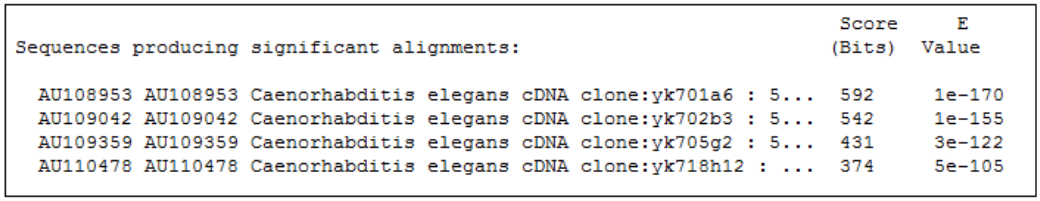
```
BLASTN 2.4.0+


Reference: Zheng Zhang, Scott Schwartz, Lukas Wagner, and Webb
Miller (2000), "A greedy algorithm for aligning DNA sequences", J
Comput Biol 2000; 7(1-2):203-14.



Database: /home/usuario/BioCloud/ncbi-blast-2.4.0+/db/ESTs
           2,000 sequences; 673,456 total letters



Query= AU108953 AU108953  Caenorhabditis elegans cDNA clone:yk701a6 : 5'
end, single read.

Length=322
```

**Cabecera**

*Figura 9.41. Cabecera del informe de salida*

#### *9.5.5.2 Resumen*

Cada línea del resumen se corresponde con un alineamiento producido. Como se muestra en la Figura 9.42, para cada alineamiento se indica el nombre de la secuencia con la que se ha alineado la query, la puntuación conseguida y su E-valor. Este listado está ordenado de mejor a peor resultado empezando por aquellas que tienen una mayor puntuación y menor e-Valor.



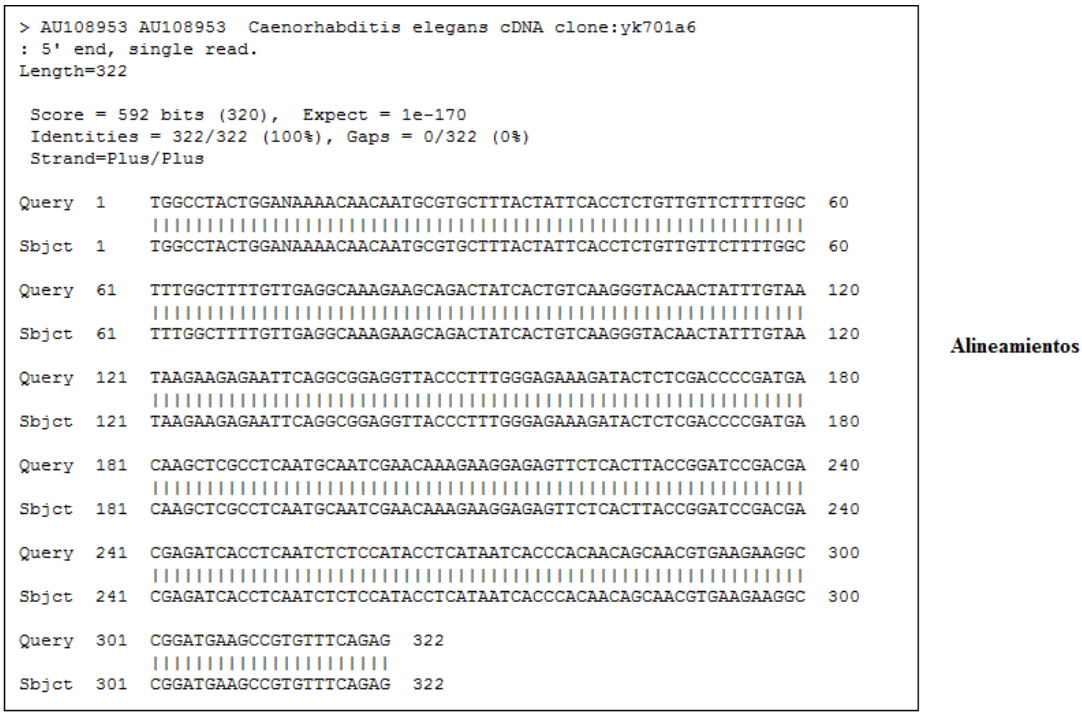
```
                                                                    Score     E
Sequences producing significant alignments:                        (Bits)  Value

  AU108953 AU108953 Caenorhabditis elegans cDNA clone:yk701a6 : 5...  592     1e-170
  AU109042 AU109042 Caenorhabditis elegans cDNA clone:yk702b3 : 5...  542     1e-155
  AU109359 AU109359 Caenorhabditis elegans cDNA clone:yk705g2 : 5...  431     3e-122
  AU110478 AU110478 Caenorhabditis elegans cDNA clone:yk718h12 : ...  374     5e-105
```

**Resumen**

*Figura 9.42. Resumen de los alineamientos producidos*

### 9.5.5.3 Alineamientos

Representa la parte principal del informe de salida y contiene información sobre cada uno de los alineamientos producidos, mostrando algunos parámetros estadísticos sobre los mismos: puntuación, e-Valor esperado y huecos, entre otros (Figura 9.43).



```
> AU108953 AU108953  Caenorhabditis elegans cDNA clone:yk701a6
: 5' end, single read.
Length=322

 Score = 592 bits (320),  Expect = 1e-170
 Identities = 322/322 (100%), Gaps = 0/322 (0%)
 Strand=Plus/Plus

Query  1    TGGCCTACTGGANAAAACAACAATGCGTGCTTTACTATTCACCTCTGTTGTTCTTTTGGC  60
            ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  1    TGGCCTACTGGANAAAACAACAATGCGTGCTTTACTATTCACCTCTGTTGTTCTTTTGGC  60

Query  61   TTTGGCTTTTGTTGAGGCAAAGAAGCAGACTATCACTGTCAAGGGTACAACTATTTGTAA  120
            ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  61   TTTGGCTTTTGTTGAGGCAAAGAAGCAGACTATCACTGTCAAGGGTACAACTATTTGTAA  120

Query  121  TAAGAAGAGAATTCAGGCGGAGGTTACCCTTTGGGAGAAAGATACTCTCGACCCCGATGA  180
            ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  121  TAAGAAGAGAATTCAGGCGGAGGTTACCCTTTGGGAGAAAGATACTCTCGACCCCGATGA  180

Query  181  CAAGCTCGCCTCAATGCAATCGAACAAAGAAGGAGAGTTCTCACTTACCGGATCCGACGA  240
            ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  181  CAAGCTCGCCTCAATGCAATCGAACAAAGAAGGAGAGTTCTCACTTACCGGATCCGACGA  240

Query  241  CGAGATCACCTCAATCTCTCCATACCTCATAATCACCCACAACAGCAACGTGAAGAAGGC  300
            ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  241  CGAGATCACCTCAATCTCTCCATACCTCATAATCACCCACAACAGCAACGTGAAGAAGGC  300

Query  301  CGGATGAAGCCGTGTTTCAGAG  322
            |||||||||||||||||||||
Sbjct  301  CGGATGAAGCCGTGTTTCAGAG  322
```

**Alineamientos**

*Figura 9.43. Alineamientos ocurridos*

### 9.5.5.4 Pie

El pie del informe (Figura 9.44) contiene una descripción detallada de los parámetros de búsqueda e información sobre la base de datos utilizada, incluyendo una breve descripción de la misma, su fecha de publicación y su tamaño.

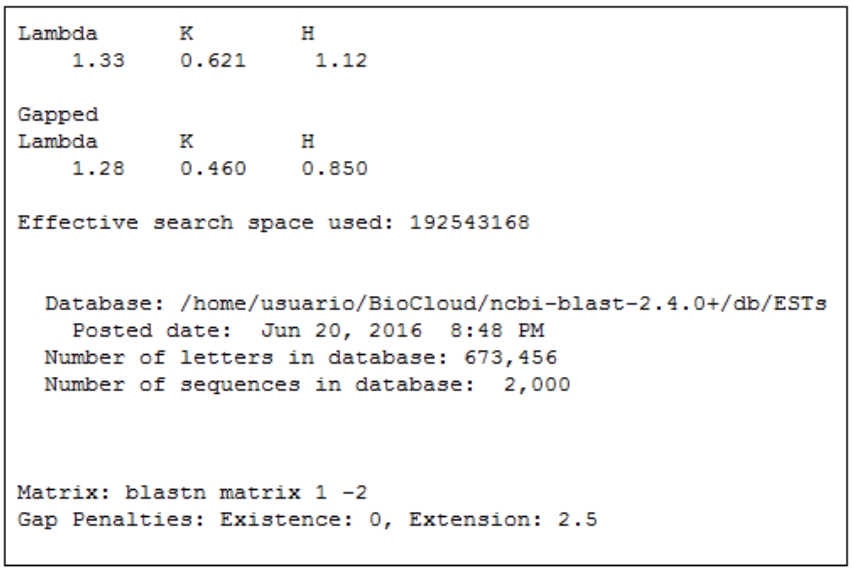



*Figura 9.44. Pie del informe de salida*

### 9.5.6 *Creación de una base de datos*

En cualquier análisis con BLAST, se realiza una búsqueda contra una base de datos local. Existen numerosas bases de datos de muy diversa índole. Muchas de estas bases de datos pueden ser descargadas desde el propio servidor FTP del NCBI a través de la URL: ftp://ftp.ncbi.nlm.nih.gov/blast/db/

Entre las herramientas que componen el programa BLAST existe una llamada **makeblastdb** que permite la creación de bases de datos a partir de ficheros FASTA. Esta herramienta funciona como un parseador o analizador sintáctico formateando los ficheros de entrada para que puedan ser usados como bases de datos locales.

El fichero usado para ilustrar el siguiente ejemplo se puede descargar de la página web de la editorial O'Reilly: http://examples.oreilly.com/9780596002992/

Así pues, un comando para crear una base de datos sería el siguiente:

```
$ makeblastdb -in /ruta/globins –dbtype prot
```

Con dicho comando se ha creado una base de datos sobre la familia globina, indicándose que el tipo de base de datos local es de proteínas. Si la base de datos fuera de nucleótidos, en dbtype se debería especificar el valor *nucl*.

Al ejecutar el anterior comando, y si todo ha ido correctamente, se crearán varios ficheros:

- **globins.pin** → Contiene información sobre el formato de la base de datos.
- **globins.psq** → Contiene las secuencias contra las que se realizarán las búsquedas.
- **globins.phr** → Contiene una cabecera para cada secuencia.

### 9.5.7 *Uso básico de BLAST*

BLAST es ejecutado a través de línea de comandos. Un comando típico para realizar una búsqueda contra la base de datos creada anteriormente es el siguiente:

```
$ blastp –query ruta/fichero/globins –db ruta/a/bd/globins –out nombre_salida.FASTA
```

El significado de cada uno de los parámetros del comando anterior se explica en la Tabla 9.5.

| Parámetro | Valor por defecto | Significado |
|---|---|---|
| **query** | - | Fichero que contiene la secuencia problema |
| **db** | - | Base de datos contra la que se va a efectuar la consulta |
| **out** | - | Nombre del fichero que va a contener el informe de salida. El formato de este fichero de salida puede ser FASTA, HTML o XML |
| **word_size** | **BLASTN** → 11<br>**BLASTP** → 3 | Tamaño de la palabra |
| **evalue** | 10 | Grado de significación de los alineamientos |
| **threshold** | **BLASTP** → 11<br>**BLASTX** → 12<br>**TBLASTN** → 13 | Determina si una palabra será incluida o no en el alineamiento |

*Tabla 9.5. Parámetros de BLAST [FUENTE: [30]]*

El valor que se asigne a cada uno de ellos puede influir significativamente en los resultados obtenidos, por lo que es necesario prestarles especial atención.

## 9.6 ANEXO VI. TRINITY

En este anexo, en primer lugar, se introduce de manera muy breve el concepto de secuenciación. A continuación, se habla de la herramienta Trinity: módulos que la constituyen, parámetros y opciones que se pueden utilizar con este programa, etc.

### 9.6.1 *Secuenciación de ARN*

La determinación de la secuencia de nucleótidos de una molécula de ARN, proceso conocido como lectura o secuenciación de ARN (RNA-seq), es fundamental en la investigación actual en Biología y Medicina. Las técnicas de análisis del ARN han evolucionado notablemente en los últimos años, existiendo hoy día en el mercado numerosas aplicaciones que permiten la secuenciación masiva de ARN, lo que ha hecho posible la obtención patrones de expresión génica y profundizar así en el conocimiento de los mecanismos que tienen lugar en determinados procesos metabólicos o situaciones patológicas.

El conjunto de genes expresados en una célula a partir del ADN genómico en un momento concreto y bajo una determinada condición fisiológica, recibe el nombre de transcriptoma [31]. En todos los organismos, el contenido genético de todas sus células es idéntico; sin embargo, no todos los genes se expresan en todas ellas ni lo hacen al mismo tiempo. El estudio y análisis del transcriptoma es vital para el entendimiento de la función de los genes, de ahí la importancia de estas técnicas de secuenciación.

### 9.6.2 *¿Qué es Trinity?*

Trinity [32], desarrollado por el Instituto Broad del MIT y Harvard y la Universidad Hebrea de Jerusalén, es un programa de secuenciación de ARN empleada en aquellos casos en los que no se dispone de un genoma conocido, es decir, realiza la reconstrucción de la secuencia de un ARN transcrito sin necesitar ninguna secuencia de referencia. Esto ha permitido el estudio de 'organismos no modelo' de gran importancia ecológica y evolutiva.

Está diseñado para ser usado en sistemas operativos UNIX (principalmente Linux) y proporciona una interfaz por línea de comandos, presentando su mejor funcionamiento en un ordenador multi-Core y de memoria elevada o en entornos de computación de alto rendimiento. Se aconseja tener aproximadamente 1GB de memoria RAM por cada millón de lecturas (***Reads***) [33].

### 9.6.3 *Módulos de Trinity*

Trinity combina tres módulos software independientes: Inchworm, Chrysalis y Butterfly [32]. En las siguientes secciones se habla brevemente de cada uno de estos módulos.

#### *9.6.3.1 Inchworm*

Inchworm, reconstruye **cóntigos** lineales, es decir, secuencias superpuestas de ARN, a partir de las lecturas de partida. Su mecanismo de funcionamiento consta de 6 etapas que se recogen en la Figura 9.46(a) [32]:

I. Construye un diccionario de k-meros, es decir, subsecuencias de longitud k. En la práctica, k suele ser igual a 25.
II. Elimina los k-meros que probablemente contengan errores de secuenciación.
III. Selecciona el k-mero más frecuente en el diccionario, utilizándolo como semilla para el ensamblado de un cóntigo.
IV. Se extiende con los k-meros más frecuentes que se solapen en k-1 posiciones en ambas direcciones del cóntigo. Cada vez que un k-mero es utilizado para extender un cóntigo se elimina del diccionario.
V. La extensión se realiza hasta que no se pueda continuar. Reportándose entonces el cóntigo lineal obtenido.
VI. Este proceso de ensamblado se repite con el siguiente k-mero en frecuencia hasta que no quede ningún k-mero en el diccionario.

#### *9.6.3.2 Chrysalis*

Chrysalis utiliza los cóntigos generados en la etapa anterior y presenta un funcionamiento en 3 fases (Figura 9.46(b)) [32]:

I. Agrupa de forma recursiva a aquellos cóntigos que posean un solapamiento en ambos extremos de al menos (k-1)-meros entre ambos.
II. Para cada agrupamiento de cóntigos definidos, Chrysalis construye grafos de Bruijn individuales (**Componentes**).
III. Asigna cada lectura al componente con el que comparte el mayor número de k-meros.

Un grafo de Bruijn es un diagrama empleado para representar solapamientos entre secuencias de símbolos. Suponiendo las lecturas ATCG, CGTA y CCAG, y, un valor de k igual a 3, se obtendría el grafo de Bruijn mostrado en la Figura 9.45:

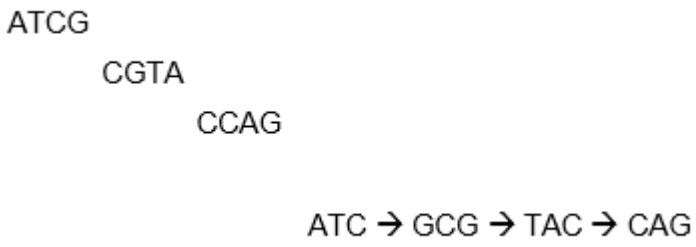



*Figura 9.45. Ejemplo de grafo de Bruijn*

#### *9.6.3.3 Butterfly*

Finalmente, Butterfly construye transcritos lineales uniendo los grafos de Bruijn generados por Chrysalis y su funcionamiento consta de dos etapas (Figura 9.46(c)): Durante la primera parte, Butterfly itera entre (1) unir nodos consecutivos generando caminos lineales en el grafo de Bruijn y (2) eliminar secuencias falsas. En la segunda parte se identifican aquellas ramas que son compatibles con alguna lectura, empleando para ello un procedimiento de programación dinámica [32].

### 9.6.4 *Utilizando Trinity con OGS/GE*

Trinity permite que los procesos de Butterfly puedan trabajar en paralelo en un único equipo multi-Core, indicando el número de procesos que pueden correr en paralelo a través del parámetro –CPU. Sin embargo, las últimas fases de Trinity también pueden ser ejecutadas de manera paralelizada en un sistema distribuido, tal y como se muestra en la Figura 9.47. De modo que Trinity es capaz de trabajar con gestores de recursos distribuidos tales como *Load Sharing Facility* (LSF) o *Sun Grid Engine* (SGE). El comportamiento del sistema distribuido es configurado a través ficheros como el que se presenta en el apartado 9.3.2.

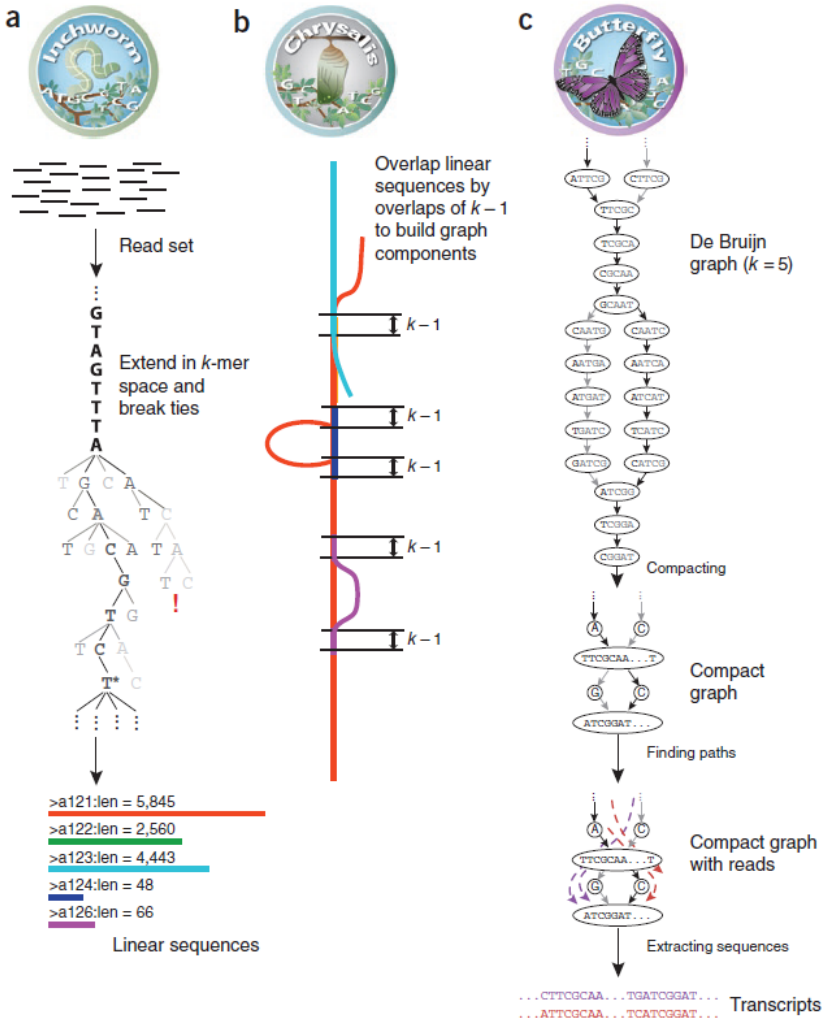



*Figura 9.46. Funcionamiento conjunto de Trinity. (a) Módulo Inchworm. (b) Módulo Chrysalis. (c) Módulo Butterfly [FUENTE: [32]]*

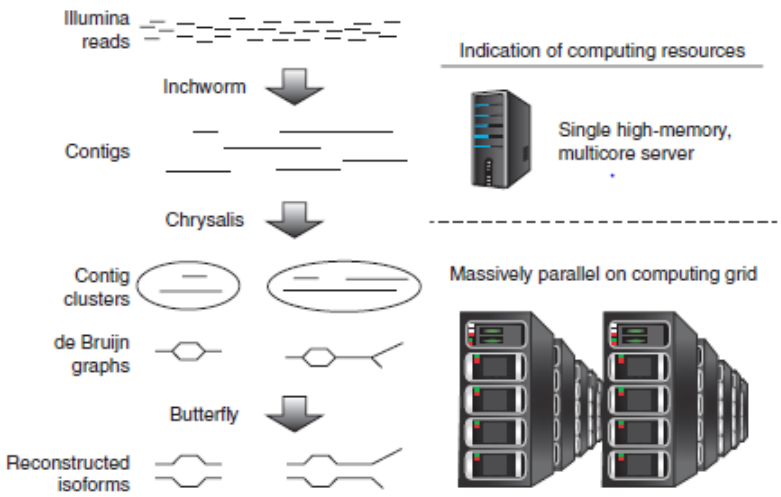



*Figura 9.47. Recursos computacionales requeridos por Trinity [FUENTE: [33]]*

### 9.6.5 *Librerías usadas por Trinity*

A la hora de realizar un análisis con Trinity, es posible que las muestras problema sean hebra-específica (*Strand-specific*), es decir, aquellas en las que es posible distinguir los transcritos sentido/antisentido (sentido 5' → 3' / sentido 3' → 5'). Para ello, Trinity puede utilizar 4 tipos de librerías [34]:

- **Lecturas pareadas** (Formadas por dos hebras complementarias)
  - **RF**: La primera lectura (/1) es secuenciada en sentido inverso (R*everse* (R)), y la segunda, en sentido directo (F*orward* (F)).
  - **FR**: Se realiza el procedimiento inverso al de la librería anterior.
- **Lecturas individuales** (Formadas por una sola hebra)
  - **F**: La lectura de la muestra es realizada en sentido directo (*Forward*).
  - **R**: La lectura de la muestra es realizada en sentido inverso (*Reverse*).

Para determinar el tipo de librería a utilizar se emplea el parámetro –SS_lib_type seguido del nombre de la librería. Por defecto, todas las muestras serán tratadas como no hebra-específicas.

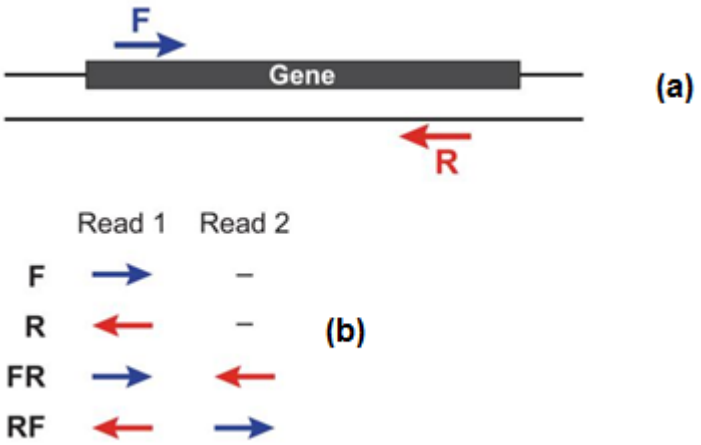



*Figura 9.48. (a) Sentido de lectura de la transcripción. (b)Tipos de librerías a utilizar por Trinity [FUENTE: [34]]*

### 9.6.1 *Ficheros de entrada*

A continuación, se describe brevemente la estructura de los dos formatos que suelen utilizarse con más frecuencia para representar muestras de secuencias biológicas y que pueden ser empleados como formatos de entrada con el programa Trinity: FASTA Y FASTQ.

#### 9.6.1.1 FASTA

Este formato es el estándar utilizado para representar secuencias biológicas (ácidos nucleicos y proteínas). Consta de dos partes: una línea de definición y una o más líneas de secuencia (Figura 9.49):

1. La línea de definición comienza con el carácter '>'. A continuación, aparece el identificador de la secuencia y una descripción de la misma. Es sumamente importante que no exista ningún espacio entre el carácter '>' y el primer carácter del identificador ya que el espacio se emplea como delimitador.
2. Una o más líneas de secuencia.

```
>61DFRAAXX100204:1:79:11167:21441/1
CATTCCCAGCCTCATCCCAAGTCCCAGGCCTACACATAGGGCGTGTGGANNNNNNNNNNNNNNNNNNNNNNNNNNN
```

*Figura 9.49. Ejemplo de formato FASTA*

#### 9.6.1.2 FASTQ

FASTQ es un formato basado en texto plano empleado para almacenar secuencias biológicas acompañándoolas de una puntuación sobre la calidad de las mismas. Desarrollado inicialmente por el Instituto Sanger, en los últimos años se ha convertido en el estándar de entrada de herramientas de secuenciación, junto con el formato FASTA.

Para representar cada secuencia, este formato normalmente emplea 4 líneas, tal y como se muestra en la Figura 9.50 [35]:

1. La primera línea comienza con el carácter '@', seguido de un identificador de la secuencia y opcionalmente, de una descripción de la secuencia.
2. La secuencia biológica, en la cual no se permite el uso de espacios en blanco o de tabulaciones. Esta secuencia es cualquier combinación de los 4 nucleótidos: A, G, C y T.
3. Indicador de fin de secuencia e inicio de calidades asociadas: el carácter '+'. Esta línea puede constar únicamente de dicho carácter.
4. Calidades asociadas a la secuencia. Estas calidades están codificadas por un conjunto de caracteres ASCII.

```
@61DFRAAXX100204:1:100:10497:13422/1
GTAATTTCCGTACCTGCCACAGTGTGGGCTCACCCTGCTTAGAGGACAGGGAAGGACCCTAAAGGTAGGCTGATGC
+
CCCCCCCCCCCCCCCCCCCCCCDCDCCCCCCCCCCCCCCCCCCDDCCDDCDCBDCCDDDDBADDADDB@DBBBA@
```

*Figura 9.50. Ejemplo de formato FASTQ*

### 9.6.2 *Informe de salida de Trinity*

Cuando Trinity finaliza, se genera un informe de extensión FASTA. La estructura de este informe es la típica del formato FASTA, tal y como puede observarse en la Figura 9.51.

```
>TRINITY_DN16_c0_g1_i1 len=485 path=[956:0-52 957:53-65 958:66-484] [-1, 956, 957, 958, -2]
GAGTTCCAGGACAGCCTGGACTATACAGAGAAACCCTGTCTCGAAAAAAAAAAACAAAAAA
AGTGTCTATACGACATTCTCTAGCTCAGTATAACTTAGATCTCAGACATCACAGGCACAG
CACAAATGTTTCATGTTAGTGAGTCAGGTTGGTAGTCAGCCTAAGGACAAGGGCAATAGC
ATTTCCTGTAGGCTGATCACTGGAACAAAAAATAAATAAACGAACGGGTGTTGTGACTTC
CTTCTGAGCCAAGTACTAAAGTTTAGAACTGACTGACCTATTTAAACAAACCCCATCCGC
CATCTTTAGATAATGATGACAGAGAAGAAAAACCAGACTTCCTACTTTGACACGTATGTA
TGAACTGGGAAGGAACTTGGTGGTAAAGCACATACTGAGCATGTGCAAGGCACCAGAGTC
TGTCTTCAGCAACATTTAAAACAAACAAACAAACAAACAACAAAACCCAAGCCTGCTAAT
CTCAA
```

*Figura 9.51. Ejemplo de informe de salda de Trinity*

### 9.6.3 *Uso básico de Trinity*

Trinity posee una interfaz por línea de comandos. Un comando típico para el análisis de muestras no hebra-específicas sería el que se muestra a continuación:

*$ Trinity --seqType fa --max_memory 3G --left lecturas_izquierda.fa --right lecturas_derecha.fa --CPU 2 –output /ruta/de/salida*

Se puede analizar varias muestras problema en una misma ejecución, separando cada una de las muestras por una coma:

*$ Trinity --seqType fa --max_memory 3G --left lectura_izquierda1.fa, lectura_izquierda2.fa, lectura_izquierda3.fa –right lecturas_derecha1.fa, lecturas_derecha2.fa, lecturas_derecha3.fa --CPU 2 --output /ruta/de/salida*

El significado de estos parámetros y de algunos otros se recogen en la Tabla 9.6:

| Parámetro | Significado |
|---|---|
| **seqType <string>** | Tipo de secuencia (Fa o Fq) |
| **max_memory <string>** | Máxima cantidad de memoria RAM a usar por Trinity |
| **left <string>** | Archivo con lecturas a izquierdas |
| **rigth <string>** | Archivo con lecturas a derechas |
| **single <string>** | Lecturas individuales, es decir, de una sola cadena |
| **CPU <int>** | Número de procesadores a usar por Trinity, es decir, el número de hebras (Por defecto su valor es 2) |
| **output <string>** | Directorio donde se creará los ficheros de salida |
| **SS_lib_type <string>** | Orientación de las muestras hebra-específicas |
| **grid_conf <string>** | Fichero de configuración a usar por el sistema grid |

*Tabla 9.6. Parámetros de Trinity [FUENTE: [17]]*

## 9.7 ANEXO VII. OPEN GRID SCHEDULER/GRID ENGINE

En este anexo, en primer lugar se establece el contexto en el que surge y el estado del arte de la Tecnología Grid. A continuación, se explican con detalle las bases teóricas sobre las que se fundamenta el sistema gestor OGS/GE.

### 9.7.1 *Introducción*

Primeramente, en las siguientes páginas, se realiza una introducción sobre la Computación Grid y sus orígenes. A continuación, se comentan las distintas capas que componen una arquitectura grid. Finalmente, se introducen una serie de conceptos relacionados con la computación grid.

#### *9.7.1.1 ¿Qué es la Computación Grid?*

Los avances tecnológicos producidos en las últimas décadas han motivado un incremento gradual de las necesidades computacionales de la sociedad, demandando ordenadores cada vez más potentes que satisfagan estas necesidades. Así, surgieron en la década de los 80 los superordenadores, cuyo funcionamiento está basado en el proceso en paralelo de un elevado número de CPU [36]. Sin embargo, en algunos casos, el uso de un único superordenador no era suficiente, por lo que se comenzó a adoptar la estrategia de unir varios de ellos cubriendo así estos nuevos requerimientos computacionales. Es lo que se conoce como Meta-computación. Años más tarde, gracias a la aparición de las redes de área local (LAN), redes de área metropolitana (MAN) y redes de área extensa (WAN), se pudo avanzar aún más y comenzar a compartir y agregar de manera coordinada recursos heterogéneos y geográficamente dispersos, surgiendo de este modo la computación distribuida.

La **Computación Grid** o **Grid Computing** es un nuevo paradigma de computación distribuida, donde se propone utilizar recursos computacionales distribuidos geográficamente para, mediante la agregación y compartición, formar un ordenador virtual con capacidades computacionales superiores a las de los superordenadores existentes [36].

Son varias las disciplinas que hacen uso de esta tecnología: Bioinformática, Meteorología, Medicina y Física, entre otras. Todas ellas, precisan hoy día de nuevas soluciones tecnológicas capaces de satisfacer sus requerimientos computacionales y de gestión de datos. Así, en el caso de la Bioinformática, la secuenciación del Genoma

Humano ha motivado un cambio importante en la naturaleza y las prioridades de la investigación de esta disciplina, dando lugar a la aparición de las tecnologías de genómica, proteómica y transcriptómica que permiten abordar por completo el estudio de los procesos que codifican la vida.

#### *9.7.1.2 Arquitectura Grid*

Al igual que ocurre con el modelo de arquitectura del protocolo TCP/IP, la arquitectura de un sistema grid se suele describir en términos de "capas", cada una de las cuales desempeña una determinada función (Figura 9.52). Las capas superiores son las más próximas al usuario, mientras que las inferiores, las más cercanas a las redes de ordenadores. Cada una de estas capas son las que se citan a continuación [37], [38]:

- **Capa de aplicación (*Applications*)**. En esta capa se encuentran definidos los protocolos que hacen uso de la infraestructura grid, es decir, que permiten el acceso a la misma.
- **Capa de Recursos o Colectiva *(Collective)***. Responsable de proporcionar herramientas que permiten la coordinación de los distintos recursos en un entorno grid.
- **Capa de Recurso (Resource)**. Especifica una serie de protocolos para una negociación segura, iniciación de transacciones, monitorización, control de los trabajos y servicios de auditoría, que permiten obtener la información de un recurso en particular y gestionarlo.
- **Capa de Conectividad (Connectivity)**. Define protocolos estándar de seguridad y comunicación, para proveer de este modo comunicaciones seguras. Los protocolos de comunicación permiten el intercambio de datos entre la capa inferior y los recursos, mientras que los protocolos de seguridad proporcionan mecanismos de criptografía para identificar usuarios y recursos. Entre los protocolos de comunicación cabrían citar: Internet (IP), Transporte (TCP, UDP) o de Aplicación (DNS), mientras que en el caso de protocolos de seguridad, SSL y certificados X.509.
- **Capa de Infraestructura (*Fabric*)**. En esta capa se ubican los recursos computacionales que serán compartidos por las distintas organizaciones virtuales, tales como un cluster o un simple ordenador personal, sistemas de almacenamiento, bases de datos, repositorios de código, junto con la infraestructura de la red y sus mecanismos de gestión y control. Según la complejidad de esta capa, se permitirán operaciones de compartición más o menos sofisticadas.

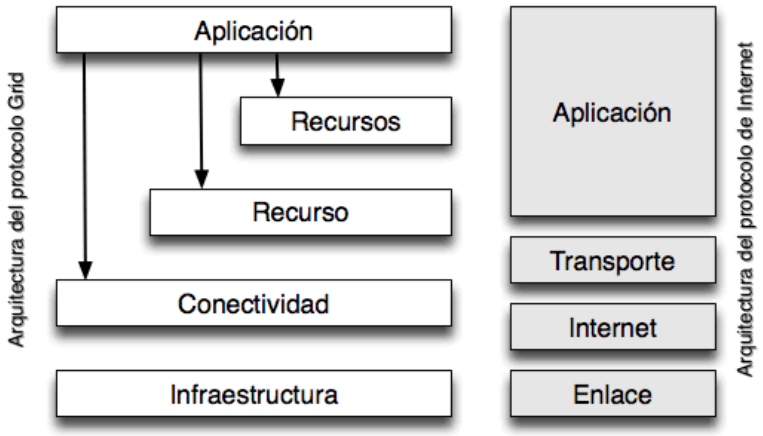



*Figura 9.52. Arquitectura de capas de un sistema grid, en relación con la arquitectura de protocolos de Internet [FUENTE: [38]]*

### *9.7.1.3 Conceptos básicos y terminología*

Antes de comenzar a hablar sobre el sistema gestor empleado en el contexto de este TFG, es decir, OGS/GE, se ha considerado necesario introducir al lector algunos conceptos que aparecen repetidamente en las siguientes páginas [39]:

- **Tarea:** Unidad computacional que se ejecuta en un nodo grid.
- **Trabajo (Jobs):** Conjunto de tareas que pueden precisar de una serie de recursos: CPU, número de nodos, datos, etc., necesarios para la realización de estos trabajos. En el caso más simple, un trabajo estará formado por una sola tarea; en el caso de trabajos más complejos, pueden existir relaciones entre tareas, de tal modo que éstas deberán ser ejecutadas siguiendo unas restricciones respecto a los datos requeridos o generados.
- **Recurso:** Entidad básica de cómputo donde los usuarios pueden ejecutar sus aplicaciones o acceder a archivos independientemente de su localización geográfica. Es en estos recursos donde las tareas, trabajos y aplicaciones son planificadas, alojadas y procesadas.
- **Planificador:** Componente software encargado de alojar tareas, trabajos o aplicaciones en recursos del grid bajo múltiples criterios y configuraciones. Existen diferentes niveles de planificadores: *Metaplanificador*, que selecciona recursos y los conecta en una red virtual y *Planificador local*, que actúa en un ámbito más reducido.
- ***Job Submission*:** Es la acción de delegar en el sistema grid la elección del mejor recurso computacional para ejecutar un trabajo o una tarea y su envío para que éstos sean ejecutados.

- **Organización Virtual:** Conjunto de usuarios y recursos con objetivos de investigación e intereses comunes.

### 9.7.2 *Open Grid Scheduler/Grid Engine*

En los siguientes sub-apartados se explica el fundamento teórico y el procedimiento de trabajo del programa OGS/GE.

#### *9.7.2.1 Introducción*

Para implementar la infraestructura grid se ha hecho uso del sistema gestor Open Grid Scheduler/Grid Engine (OGS/GE) [40]. Basado en el gestor de colas de Oracle, Sun Grid Engine, y publicado bajo licencia *Sun Industry Standards Source License* (SISSL), es un sistema cuya función es la de gestionar de recursos computacionales o procesos distribuidos en ambientes heterogéneos, con el objeto de que dichos recursos sean utilizados de forma eficiente.

#### *9.7.2.2 Arquitectura del gestor de colas OGS/GE*

La arquitectura de este gestor de colas consta de un nodo central, *Master host* o planificador, encargado de recoger los trabajos lanzados por los usuarios del sistema y de procesar las tareas administrativas. Junto a éste, se encuentran los recursos computacionales que serán los encargados de ejecutar los trabajos. Esta arquitectura define una serie de procesos demonios responsables de realizar las funciones designadas a cada tipo de máquina (Figura 9.53):

- ***sge_qmaster***. Controla todos los componentes del sistema grid: colas, trabajos, recursos computacionales y carga del sistema, entre otros. Así mismo, mantiene una serie de tablas con el estado de cada uno de estos componentes. Además, realiza las tareas de planificación propiamente dichas: Decidir a qué cola se envía cada trabajo, cómo reordenar los trabajos en base a su prioridad, etc.
- ***sge_execd***. Presente en aquellos nodos que tienen permisos para realizar trabajos. Se encarga de ejecutar los trabajos y enviar al sge_qmaster información sobre el estado de los mismos cada cierto tiempo.

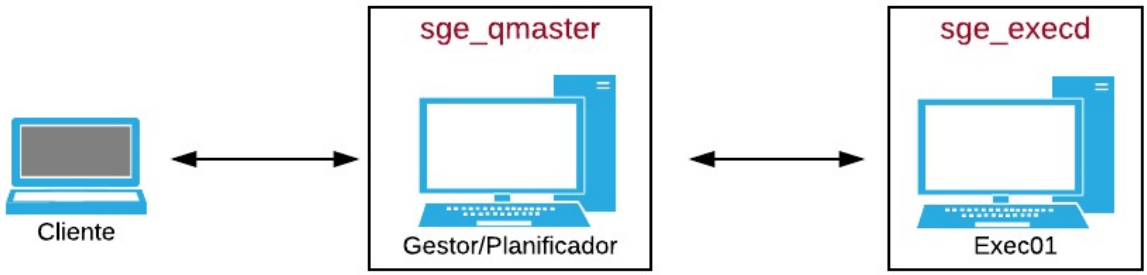



*Figura 9.53. Arquitectura del gestor de colas OGS/GE*

### 9.7.2.3 Ficheros de descripción de trabajos de OGS/GE

OGS/GE trabaja con scripts Shell que definen las características y requisitos de los trabajos de los usuarios, aunque también puede trabajar con binarios ejecutables.

Los ficheros de descripción de trabajos constituyen el mecanismo mediante el cual los usuarios (o aplicaciones) pueden detallar el trabajo que van a ejecutar los recursos computacionales gestionados. De este modo se puede informar al gestor de colas del entorno de ejecución bajo en cual se ha de llevar a cabo el trabajo o la tarea. Este entorno de ejecución suele constar de lo siguiente:

- Opciones que ofrecen información potencialmente útil para la ejecución del trabajo a realizar. Tres opciones que merece la pena mencionar son -V*, - cwd y -j*. Con la primera opción se indica que el trabajo debería tener las mismas variables de entorno que las indicadas junto con el comando qsub. Por su parte, *-cwd* determina que se debe ejecutar el trabajo en el mismo directorio desde el que fue enviado a la cola de tareas. Finalmente, *-j* establece que tanto la salida estándar como el error estándar se guardarán juntos en un mismo fichero. Además, existen otras opciones que permiten asignar un nombre al trabajo, enviar un correo electrónico a una dirección indicada bajo determinadas circunstancias, etc.
- Rutas y nombres de los diferentes programas que ejecutarán los procesos que constituyen el trabajo.
- Argumentos de estos programas.

Este fichero de descripción de trabajo presenta la estructura que se explica a continuación. Para ello, se utilizará un fichero ejemplo proporcionado por OGS/GE llamado simple.sh. Este fichero puede encontrarse en el directorio ge2011.11/examples/jobs. Tal y como puede observarse en la Figura 9.54, la estructura de este archivo es bastante sencilla:

- La línea primera es la línea de inicio típica de un fichero Shell e indica que éste va a ser ejecutado por /bin/sh.
- Las líneas que comienzan por # son comentarios que serán ignorados por el Shell.
- Por su parte, las líneas que comienzan por #$ se corresponden con opciones de OGS/GE y no afectarán en absoluto a la actividad de intérpretes Shell externos a este programa.
- El resto de líneas del fichero de descripción contienen las tareas a realizar. En este caso se mostrará la fecha y hora del momento en el que comienza el trabajo, volviéndose a mostrar 20 segundos después.

```
1  #!/bin/sh
2  #
3  # This is a simple example of a SGE batch script
4  #
5  # request Bourne shell as shell for job
6  #$ -S /bin/sh
7  #
8  # print date and time
9  date
10 # Sleep for 20 seconds
11 sleep 20
12 # print date and time again
13 date
```

*Figura 9.54. Fichero de descripción de trabajo simple.sh [FUENTE: [17]]*

#### *9.7.2.4 Entorno paralelo*

Las características de un entorno paralelo se definen en ficheros de configuración como el que se muestra en la Figura 9.55.

```
pe_name           make
slots             999
user_lists        NONE
xuser_lists       NONE
start_proc_args   NONE
stop_proc_args    NONE
allocation_rule   $round_robin
control_slaves    TRUE
job_is_first_task FALSE
urgency_slots     min
accounting summary TRUE
```

*Figura 9.55. Fichero de configuración de entorno paralelo*

Éste consta de los siguientes parámetros [41], [42]:

- **pe_name –** Nombre con el que el sistema gestor conocerá al entorno paralelo.

- **slots –** Número de procesos paralelos que se pueden ejecutar de manera concurrente bajo el entorno paralelo.
- **user_lists –** Lista de control de acceso de los usuarios que pueden utilizar este entorno distribuido.
- **xuser_lists –** Lista de control de acceso de los usuarios a los que **no** se les permite usar este entorno paralelo.
- **start_proc_args –** Ruta al script de inicio del entorno paralelo, seguido de los argumentos que sean necesarios.
- **stop_proc_args -** Ruta al script de parada del entorno paralelo, junto que con los argumentos pertinentes.
- **allocation_rule –** Estrategia de planificación utilizada por el sistema gestor para la asignación de trabajos a los recursos computacionales.
- **control_slaves –** Establece si la integración del entorno paralelo es "estricta" o "relajada". Es decir, determina si OGS/GE es el creador de las tareas esclavas de una aplicación paralela.
- **job_is_first_task –** Indica si OGS/GE es el creador de tareas esclavas de una aplicación paralelizada.
- **urgency_slots –** Determina cómo afecta la petición de recursos a la prioridad de trabajos paralelos.
- **accounting_summary –** Establece si el registro de contabilidad está disponible para cada tarea esclava iniciada por OGS/GE.

La versión actual del gestor OGS/GE permite utilizar 4 estrategias de planificación distintas [43]:

- Por defecto, el gestor OGS/GE utiliza el algoritmo **Round-Robin (RR)**. RR funciona de la siguiente manera: Si un trabajo precisa 8 slots, se irá reservando un slot en cada recurso empezando por el primer recurso, volviendo al primero en caso de ser necesario.
- Existe otra estrategia llamada, **Fill up**, que consiste en ir reservando todos los slots de cada recurso antes de seguir con el siguiente recurso, es decir, se llena cada recurso antes de continuar con el siguiente, de ahí su nombre.
- Una tercera estrategia, conocida como **pe_slots** reserva todos los slots en un único recurso computacional. De tal modo que a cada recurso sólo se asignarán aquellos trabajos para los que posee el suficiente número de slots que el trabajo precisa.
- Finalmente, se puede especificar numéricamente el número de slots que se pueden reservar en cada host. Si este valor fuese 1, sólo se podría reservar

un 1 slot en cada recurso computacional mientras que el trabajo está siendo asignado.

Para cambiar alguno de los parámetros anteriormente citados se utiliza el siguiente comando:

```
$ qconf -sp <nombreFicheroConfiguracion>
```

#### 9.7.2.5 Manual básico de OGS/GE

Existen dos formas de manejar OGS/GE: a través de interfaz gráfica (QMON) o bien mediante línea de comandos, la utilizada en la realización de este TFG.

Son varios los comandos disponibles para manejar OGS/GE a través de línea de comandos:

- **qalter** → Modificar un trabajo en batch pendiente.
- **qdel** → Borrar un trabajo o una cola de tareas.
- **qhost** → Mostrar el estado de los recursos computacionales, colas y trabajos.
- **qlogin** → Iniciar una sesión interactiva.
- **qrsh** → Mandar trabajos interactivos al sistema.
- **qsub** → Enviar un trabajo a la cola.
- **qstat** → Consultar el estado del trabajo en la cola.

Para ilustrar mejor el uso de estos comandos, a continuación se muestran una serie de ejemplos.

Para enviar una tarea a la cola de tareas, ejecutar el comando:

```
$ qsub simple.sh
```

En este caso, cuando un usuario lanza una nueva tarea, el sistema gestor OGS/GE asigna esta tarea al nodo que menos haya sido usado, es decir, el que haya tenido menor carga de trabajo.

En cambio, para llevar a cabo una ejecución paralela de tareas, se emplea el siguiente comando:

```
$ qsub –pe <nombreEntornoParalelo> <NúmeroProcesadores> <Script.sh>
```

Si lo que se quiere es conocer el estado de las tareas se debe utilizar el comando:

```
$ qstat -f
```

Si por el contrario se desea conocer el estado de una tarea concreta, el comando a emplear es:

```
$ qstat –j <jobid>
```

Cuando un trabajo es enviado a la cola de tareas puede presentar varios estados diferentes: **qw** (El trabajo ha sido asignado a una cola y está a la espera de que haya un recurso computacional disponible para llevarlo a cabo), **t** (El trabajo está siendo transferido al recurso computacional que lo va a realizar) o **r** (El trabajo se está realizando). Si en cualquier momento el trabajo presenta el estado **Eqw**, significa que ha habido un error durante su ejecución porque se desconozca la aplicación a usar, se haya especificado un fichero o directorio que no existe o para el que no se tienen suficientes permisos de acceso, entre otras razones. Para mostrar más información sobre un trabajo fallido se puede utilizar el comando:

```
$ qstat –explain c -f <jobid>
```

Si es posible solucionar el error producido, se puede resetear este estado erróneo con el comando:

```
$ qmod -cj <jobid>
```

En caso contrario, el trabajo deberá ser cancelado usando para ello el comando:

```
$ qdel <jobid>
```

Si se desean borrar todos los trabajos lanzados por un usuario se puede utilizar el comando:

```
$ qdel –u ‘nombreUsuario’
```

También es posible que sea la cola de tareas la que se encuentre en estado de error (aparecerá una letra E junto al nombre de la cola). Para reiniciar una cola con errores se hace uso del siguiente comando:

```
$ qmod -c ‘nombreCola’
```

Si lo que se desea es reiniciar todas las cosas del sistema, lanzar el siguiente comando:

```
$ qmod -c ‘*’
```

Finalmente, si se desea eliminar un recurso computacional del sistema, se debe utilizar el siguiente comando:

```
$ qconf -de 'nombreNodo'
```

Si dicho nodo no puede ser suprimido debido a que existen referencias al mismo en una o varias colas, primeramente hay que borrarlas usando el comando:

```
$ qconf -mhgrp 'nombreCola'
```

Eliminadas todas las referencias, se podrá borrar el nodo sin ningún problema.

## 9.8 ANEXO VIII. OPENNEBULA

En las siguientes páginas se presenta al lector la plataforma de virtualización OpenNebula: sus componentes, ciclos de vida de las imágenes y las máquinas virtuales, opciones y comandos que se pueden usar…

### 9.8.1 *Introducción*

OpenNebula es el estándar de código abierto para la virtualización en centros de datos y la computación en la nube, proporcionando una potente y flexible plataforma para construir soluciones empresariales en la nube y centros de datos virtualizados, distribuidos y heterogéneos [44]. Fue desarrollada por el DSA (*Distributed Systems Architecture)* Research Group, un grupo con sede en la Universidad Complutense de Madrid centrado en la computación distribuida, virtualización y plataformas de Infraestructura como Servicio (IaaS, *Infrastructure as a Service*) [45].

OpenNebula no depende exclusivamente de ningún tipo concreto de hipervisor (gestores de virtualización). Igualmente no tiene requerimientos técnicos específicos, pudiendo adaptarse a infraestructuras ya existentes. Entre los hipervisores soportados dentro de la estructura de OpenNebula cabe destacar soluciones libres como Xen, KVM (Kernel-based Virtual Machine) y QEMU/KVM o soluciones privativas como VMWARE. Igualmente, soporta libvirt como capa de abstracción entre KVM/Xen.

Tal y como se muestra en la Figura 9.56, el sistema OpenNebula consta de una serie de herramientas (interfaz de líneas de comandos, Schedulers, etc.), un Core donde se gestionan las peticiones vía XML-RPC y en el cual reside la gestión de las máquinas virtuales, y los pools de bases de datos SQL. En una capa inferior se encuentran los drivers que hacen de nexo entre la capa hardware y el Core de OpenNebula y en ella se encuentran los Drivers de Almacenamiento (NFS, SSH), los drivers de máquinas virtuales (dependientes de los hipervisores) y los drivers de Monitorización que proporcionan información de las máquinas virtuales [45].

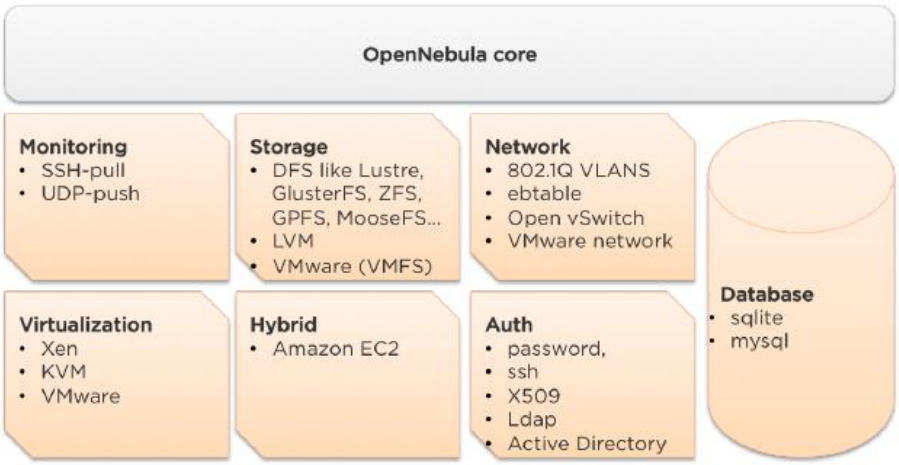



*Figura 9.56. Componentes de OpenNebula [FUENTE: [45]]*

### 9.8.2 *Ciclo de vida de las imágenes*

OpenNebula trabaja con imágenes, que deben ser alojadas previamente en un almacén llamado datastore. Cuando éstas son almacenadas en el datastore, pasan por diferentes estados, mostrando el ciclo de vida mostrado en la Figura 9.57. Así, cuando se añade una nueva imagen, ésta presentará el estado LOCKED hasta que todos los archivos hayan sido copiados al datastore, pasando entonces su estado a READY. Es en este estado cuando puede ser utilizada por cualquier máquina virtual. Si una imagen está siendo usada, mostrará el estado USED.

Cuando se almacena una imagen, ésta puede ser persistente o no persistente. La principal diferencia entre ambas es que, en el caso de ser persistente, la imagen sólo podrá ser usada por una sola máquina virtual, mientras que en el caso de ser no persistente, podrá ser utilizada por varias máquinas virtuales, tal y como se muestra en la región recuadrada en rojo de dicha figura.

Así pues, los diferentes estados en los que se puede encontrar una imagen son los que se recogen en la Tabla 9.7.


*Figura 9.57. Ciclo de vida de una imagen no persistente [FUENTE: [46]]*

| Estado | Significado |
|---|---|
| **LOCKED** | La imagen está siendo copiada o creada en el datastore |
| **READY** | La imagen está lista para ser utilizada |
| **USED** | Una imagen no-persistente está siendo usada por alguna máquina virtual |
| **USED_PERS** | Una imagen persistente es usada por una máquina virtual |
| **DISABLED** | La imagen ha sido deshabilitada por su propietario. No puede ser utilizada por otras máquinas virtuales |
| **ERROR** | La imagen presenta algún tipo de error |
| **DELETE** | La imagen está siendo eliminada del datastore |

*Tabla 9.7. Estados de las imágenes [FUENTE: [46]]*

### 9.8.3 *Ciclo de vida de las máquinas virtuales*

Al igual que ocurre con las imágenes, las máquinas virtuales también poseen un ciclo de vida conformado por una serie de estados. Las operaciones y los comandos que se pueden aplicar sobre una máquina virtual dependerán del estado en el que ésta se

encuentre. No obstante, una máquina virtual puede ser borrada independientemente del estado que presente.

Para evitar que este apartado sea excesivamente denso, se enumerarán sólo algunos de estos estados. Cuando se instancia una plantilla, la máquina virtual presenta el estado PENDING, cambiando a PROLOG cuando comienza a desplegarse en un nodo. La máquina permanecerá en este estado hasta que todos los archivos hayan sido transferidos al mismo, presentando entonces el estado RUNNING. En dicho estado, la máquina puede ser pausada o parada, mostrando el estado SAVE mientras se están guardando los archivos de la máquina virtual. Finalizado este proceso, la máquina presentará el estado SUSPENDED o POWEROFF respectivamente.

Si en algún momento una máquina muestra el estado FAILURE, ésta sólo se podrá borrar, no pudiéndose realizar ninguna otra operación adicional sobre la misma.

### 9.8.4 *Listado de comandos de OpenNebula*

OpenNebula posee una serie de comandos para interaccionar con el sistema. Estos comandos son las que se detallan a continuación:

- **Oneuser** → Referido a los usuarios. Algunas opciones son: "*Create username [Password]*" para crear un nuevo usuario con el nombre y la contraseña especificados; "*Delete range|userid_list*" para eliminar un conjunto de usuarios o un usuario específico del sistema.
- **Onehost** → Referido a los nodos anfitriones. Algunas opciones son: "*Create/Delete*" para añadir o borrar un host; "*List/Show*" para listar los nodos registrados en sistema o mostrar información sobre uno específico; "*Enable/Disable*" para habilitar o deshabilitar un nodo ya creado.
- **Onevm** → Referido a las máquinas virtuales. Algunas opciones son: "*Create/Delete*" para crear o eliminar una máquina virtual en el sistema. "*List/Show*" para listar las todas máquinas existentes o mostrar una sola y finalmente, algunos comandos para manejar las máquinas virtuales como "*Deploy, Shutdown, Migrate, Livemigrate, Hold, Release, Stop, Cancel, Suspend, Resume y Restart*", para desplegar, apagar, migrar, congelar, detener, suspender o reiniciar las máquinas virtuales entre otras tareas.
- **Onevnet** → Referido a las redes virtuales**.** Algunas opciones son: "*Create/Delete*" para crear o borrar una red; "*List/Show*" para mostrar un listado de las redes que se han creado o bien, las características de una red concreta.

## 9.9 ANEXO IX. MANUAL DE USUARIO DE OPENNEBULA

OpenNebula puede ser manejado tanto a través de línea de consola como a través de su propia interfaz gráfica. Las distintas operaciones necesarias para crear y manipular las máquinas virtuales en este TFG se han llevado a cabo a través de Sunstone, la interfaz web de OpenNebula. Para acceder a dicha interfaz ingresar en un navegador la URL: http:/localhost:9869

Aparecerá la siguiente ventana de inicio de sesión en la que se debe introducir como nombre de usuario “oneadmin” y la contraseña asociada al mismo (Figura 9.58). En caso de no disponer de dicha contraseña, se remite al lector a la sección 9.1.7.1.5.


*Figura 9.58. Ventana de Login*

Tras iniciar sesión, aparecerá el panel de control mostrado en la Figura 9.59. Éste ofrece una visión global del número de máquinas virtuales creadas o de hosts que componen la infraestructura, junto al estado de los mismos.

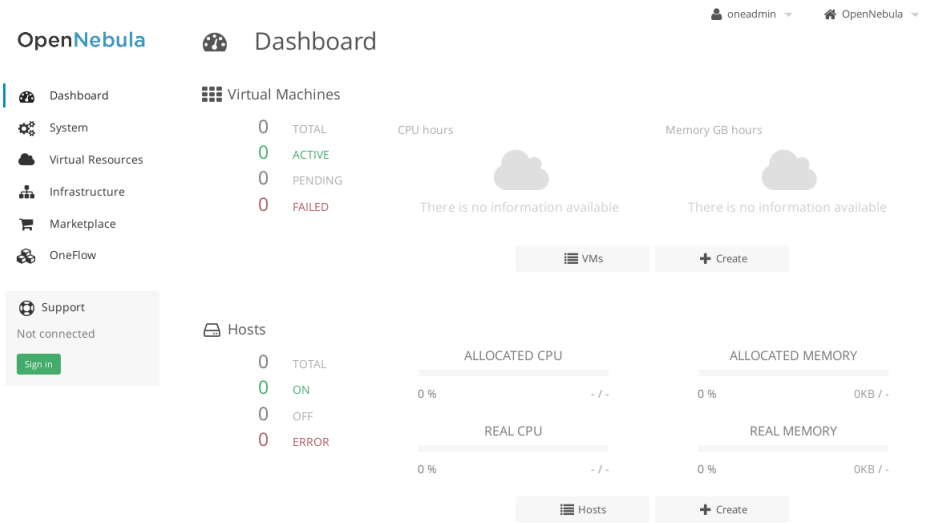



*Figura 9.59. Panel de control*

### 9.9.1 *Cambio de la contraseña del usuario oneadmin*

Para mayor comodidad en el manejo futuro de OpenNebula, es aconsejable cambiar la contraseña del usuario oneadmin. Para ello, pulsar sobre dicho usuario y a continuación, sobre el botón Password (Figura 9.60). Se abrirá un formulario en el que se debe introducir la nueva contraseña. Seguidamente pulsar sobre el botón Change para que se apliquen los cambios (Figura 9.61).

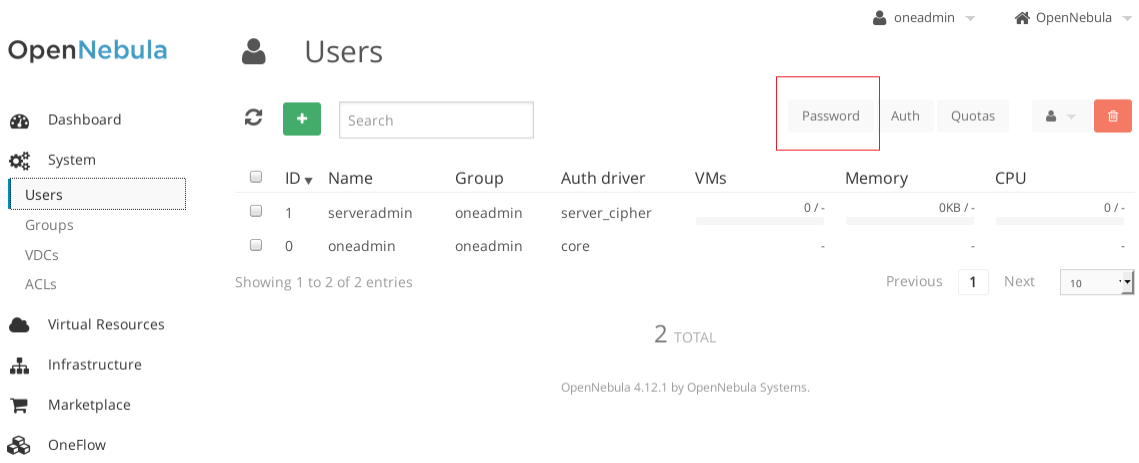



*Figura 9.60. Pantalla de Usuarios*

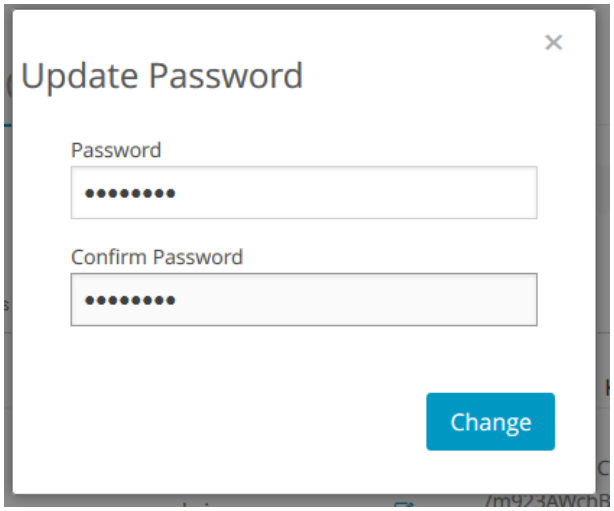



*Figura 9.61. Cambio de la contraseña del usuario oneadmin*

### 9.9.2 *Añadir un host*

Los hosts actuarán como anfitriones y proporcionarán los recursos necesarios para que las máquinas virtuales se ejecuten correctamente. Para ello, seleccionar la opción Infraestructure > Hosts en el menú de la izquierda.

A continuación, pulsar sobre el botón Crear para añadir un host (Figura 9.62). Aparecerá un formulario en el que entre otras cosas se debe especificar el nombre del nodo anfitrión, si éste pertenece a algún clúster y la solución de virtualización utilizada (Figura 9.63).

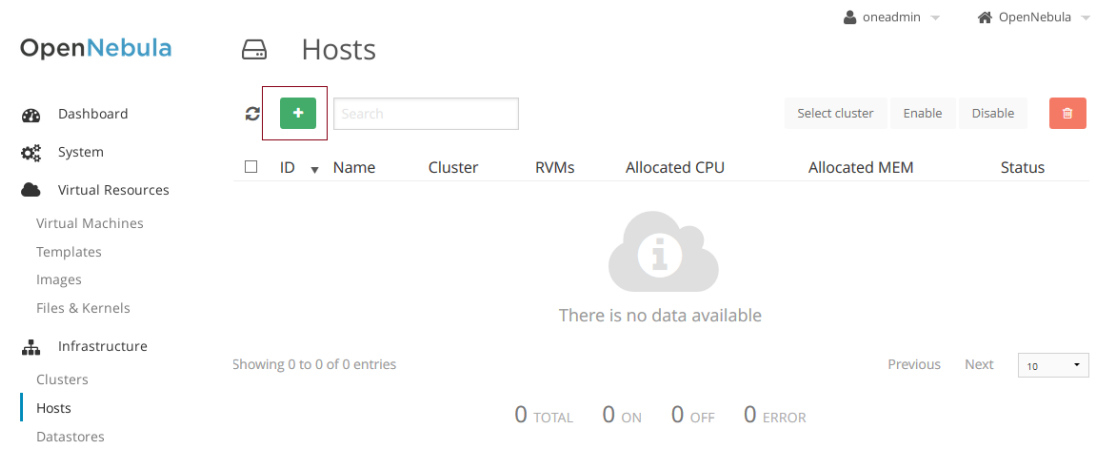



*Figura 9.62. Registro de un nuevo host*

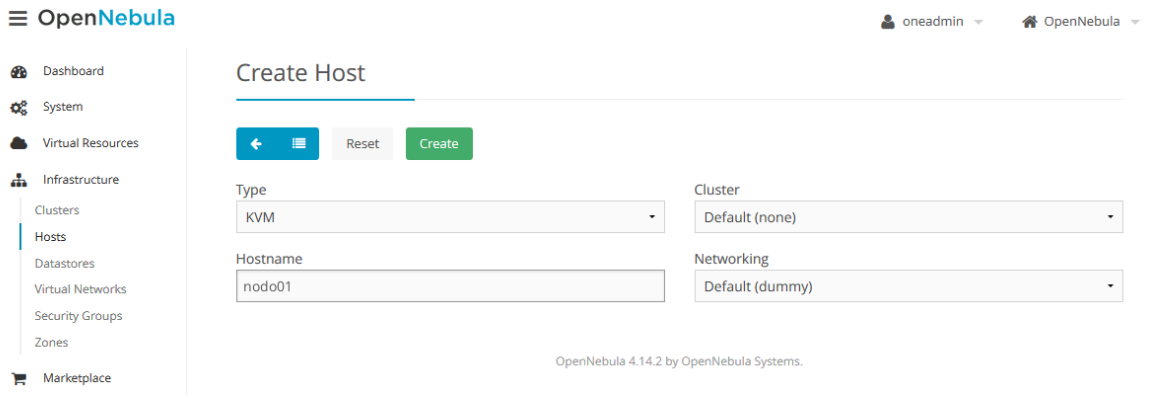



*Figura 9.63. Formulario para añadir un host*

### 9.9.3 *Añadir una imagen*

Como se comentó anteriormente, OpenNebula funciona con imágenes. Para dar de alta una imagen en el sistema pulsar sobre Virtual Resources > Images.

Nuevamente, al pulsar sobre el botón Crear se abrirá un formulario en el que primeramente se debe especificar el nombre de la imagen a crear. En el apartado de localización de imagen, seleccionar “Upload” y especificar la ubicación actual de la imagen (Figura 9.64). Desplegar la pestaña de opciones avanzadas y asignar a los parámetros Driver y Target los valores qcow2 y hda, respectivamente (Figura 9.65).

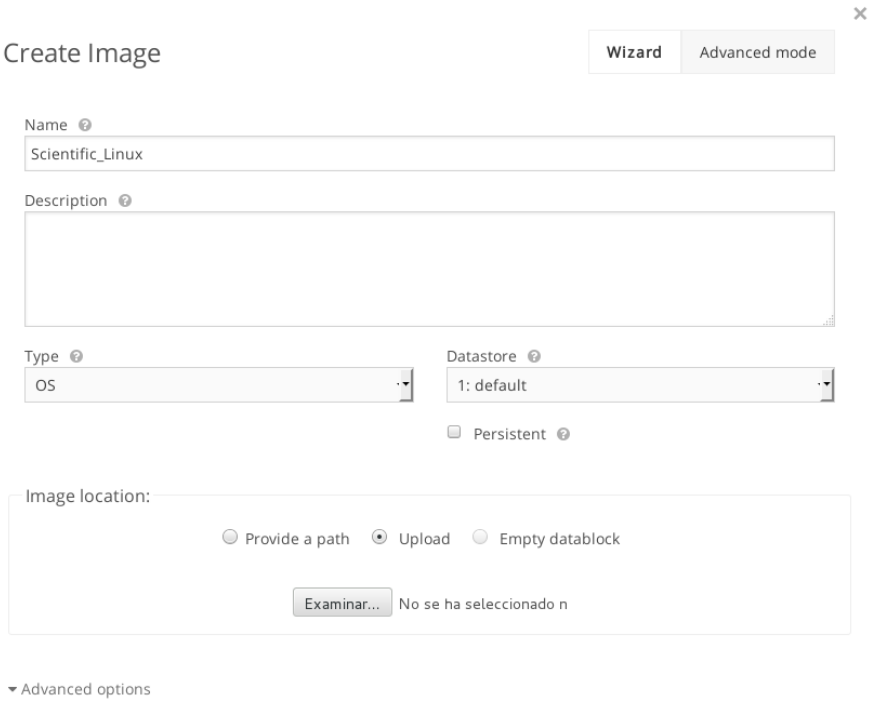



*Figura 9.64. Formulario creación de imagen*

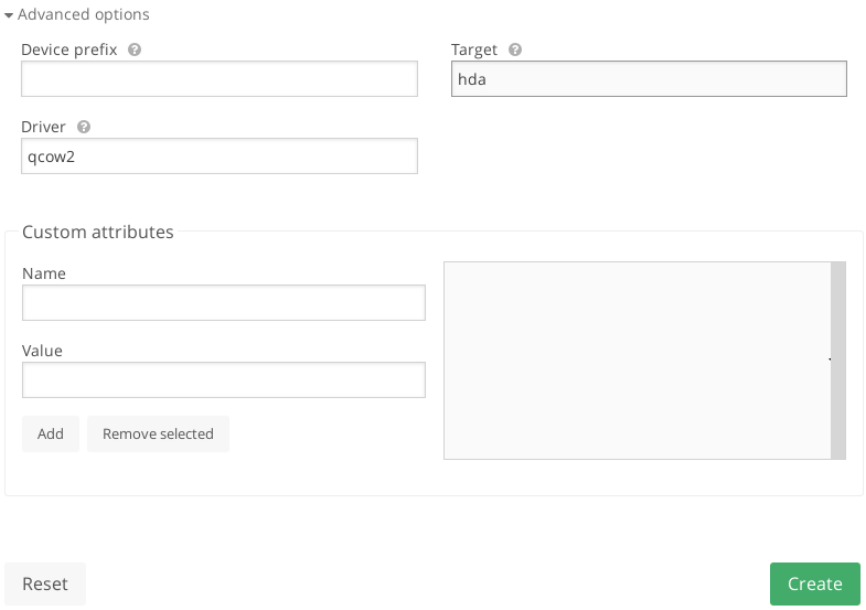



*Figura 9.65. Opciones avanzadas*

### 9.9.4 *Añadir una red virtual*

Una vez que se han añadido los hosts al sistema, el siguiente paso es crear una red virtual. Para ello, seleccionar la opción Infraestructure > Virtual Networks.

De nuevo, pulsar sobre el botón Crear. Se abrirá un formulario en el que se configurarán una serie de parámetros referidos a la red virtual a crear. Así, en la pestaña *General* se debe indicar el nombre de la red virtual (Figura 9.66).

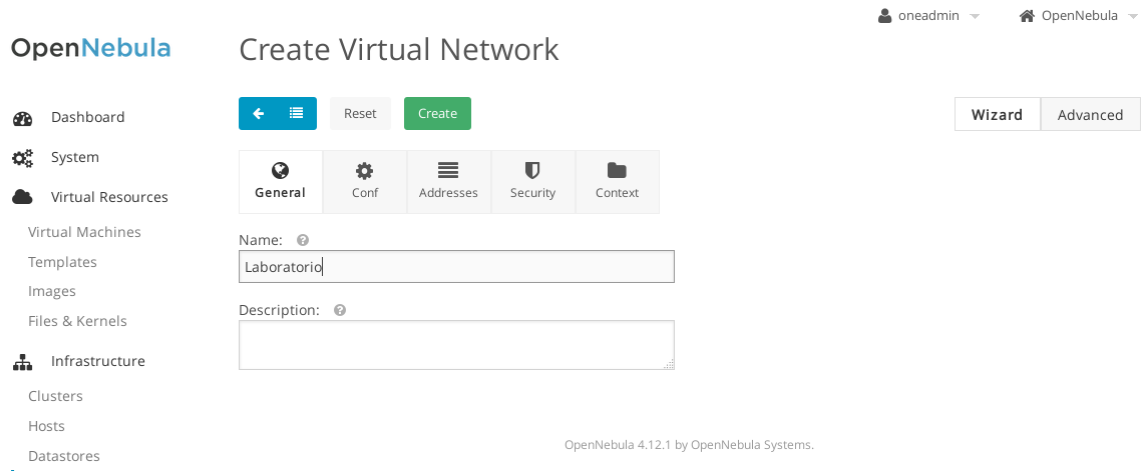



*Figura 9.66. Pestaña General*

En la siguiente pestaña, llamada *Conf*, se determina el nombre de la interfaz de red de puente, tal y como se muestra en la Figura 9.67.

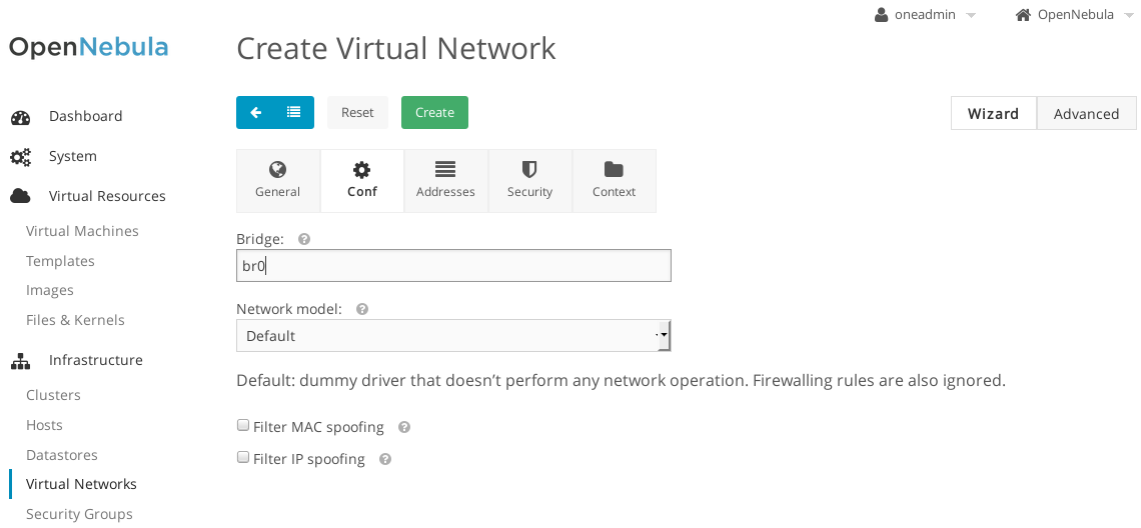



*Figura 9.67. Pestaña Conf*

Seguidamente, en la pestaña *Addresses*, se especifica el rango de direcciones disponible: la dirección IP de inicio y el tamaño del pool de direcciones (Figura 9.68).

Por último, en la pestaña *Context* se indican diversos parámetros de contextualización como la dirección de la red a la que pertenece la máquina virtual, la máscara, puerta de enlace y servidor DNS (Figura 9.69).

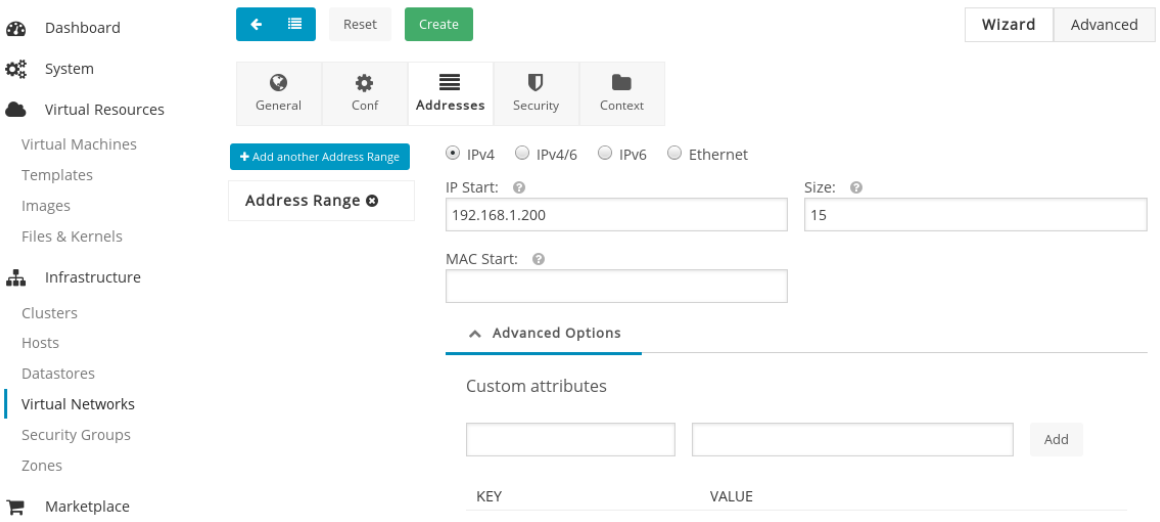



*Figura 9.68. Pestaña Addresses*

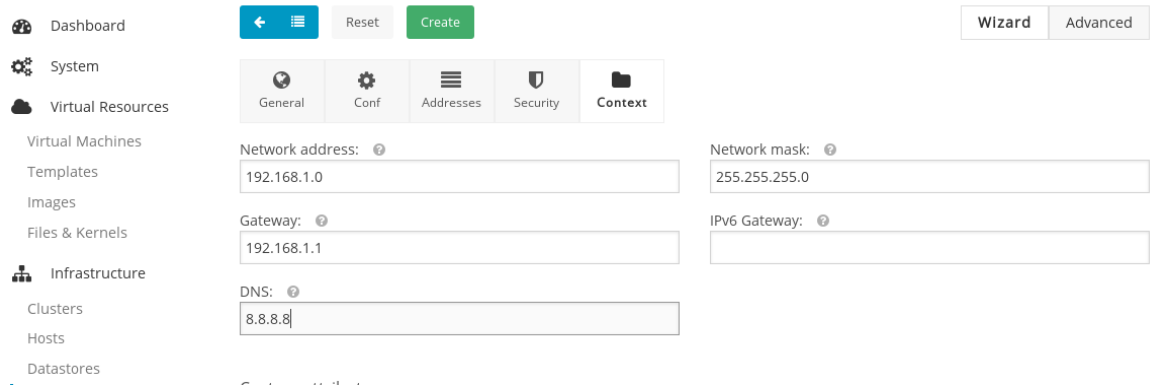



*Figura 9.69. Pestaña Context*

### 9.9.5 *Añadir una plantilla*

En OpenNebula, las máquinas virtuales son definidas utilizando plantillas de configuración. Para crear una plantilla, en primer lugar, pulsar sobre Resources > Templates.

Aparecerá un formulario compuesto por una serie de pestañas. En la pestaña *General* especificar el nombre de la plantilla a crear, la cantidad de memoria RAM, el número de CPU que el host anfitrión reservará para la máquina virtual, así como el número de vCPU de los que va a disponer la máquina (Figura 9.70).

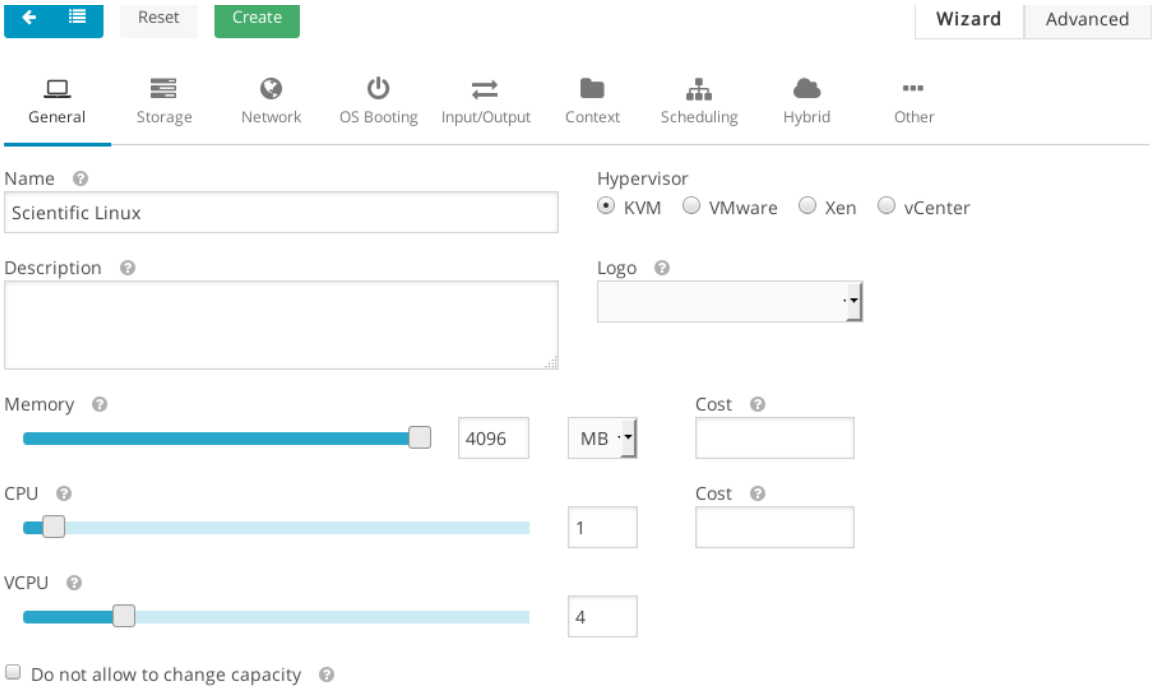



*Figura 9.70. Pestaña General*

En la siguiente pestaña, llamada *Storage*, seleccionar la imagen que utilizará la máquina virtual (Figura 9.71).

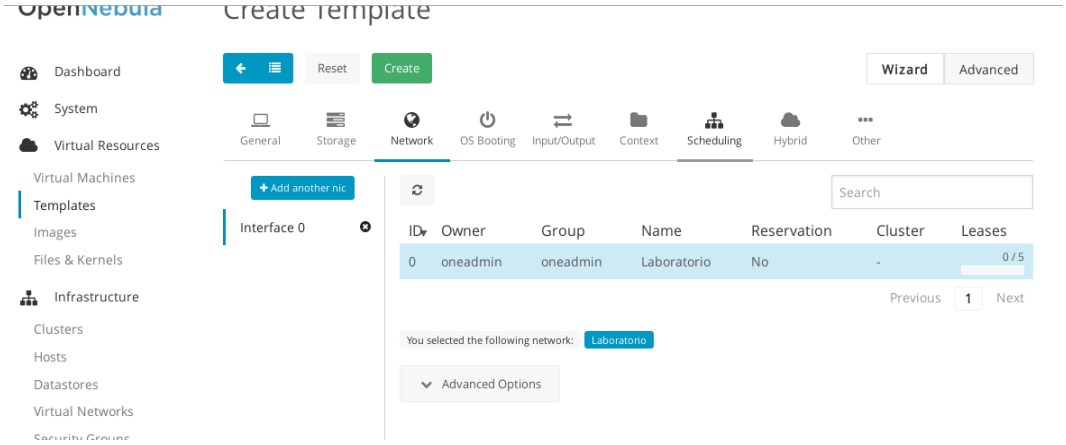



*Figura 9.71. Pestaña Storage*

A continuación, en la pestaña *Network*, escoger la red virtual de trabajo para que la máquina virtual pertenezca a dicha red y sea visible por los demás nodos de la misma (Figura 9.72).

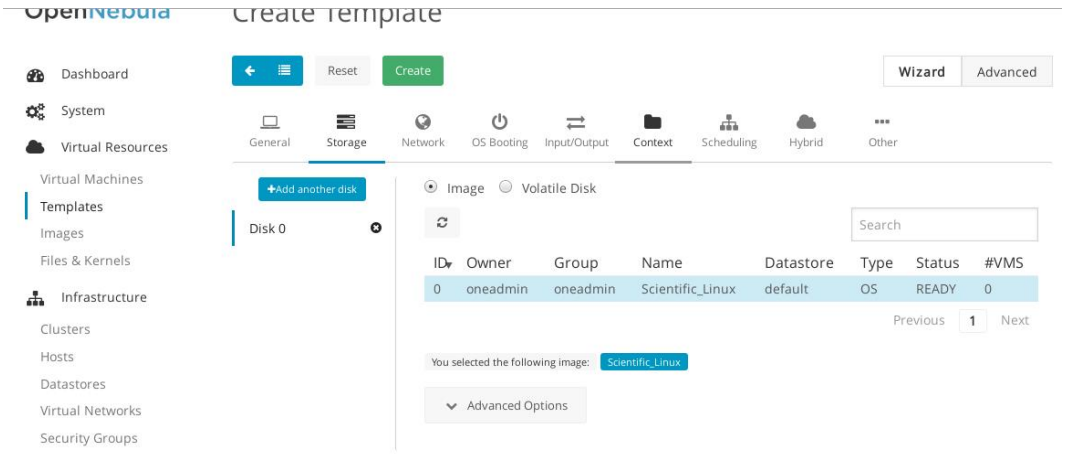



*Figura 9.72. Pestaña Network*

Seguidamente, acceder a la pestaña *OS Booting* e indicar la arquitectura de la red a crear (Figura 9.73).

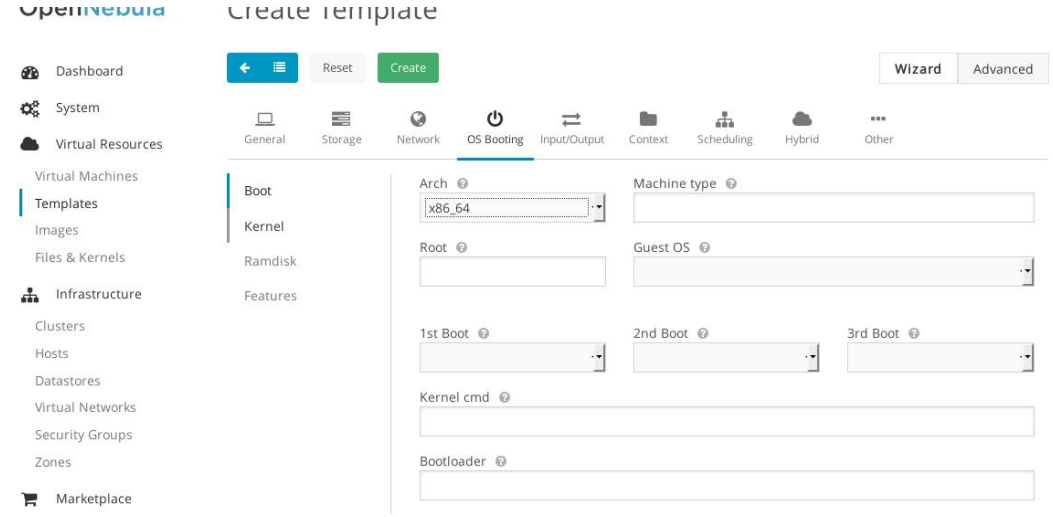



*Figura 9.73. Pestaña OS Booting*

En la pestaña *Input/Output* seleccionar VNC para añadir un servidor VNC a la máquina y poder así conectarse a ésta de forma remota (Figura 9.74).

Posteriormente, acceder a la pestaña *Context* para configurar algunas opciones de Contextualización. Para que la máquina virtual funcione correctamente es necesario seleccionar las opciones "*Add SSH Contextualization*" y "*Add Network Contextualization*". Es en este recuadro de contextualización de SSH, donde se ha copiado la clave pública de RSA del usuario root del servidor para poder conectarse por SSH a las máquinas virtuales sin que se solicite ninguna contraseña y la instalación del programa OGS/GE finalice exitosamente (Figura 9.75).

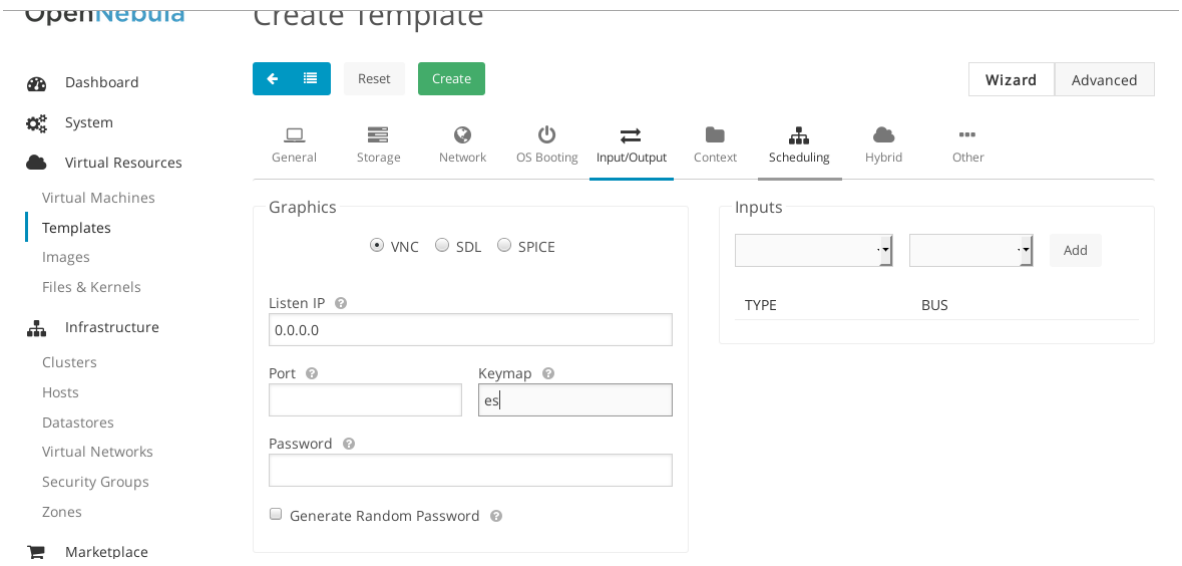



*Figura 9.74. Pestaña Input/Output*

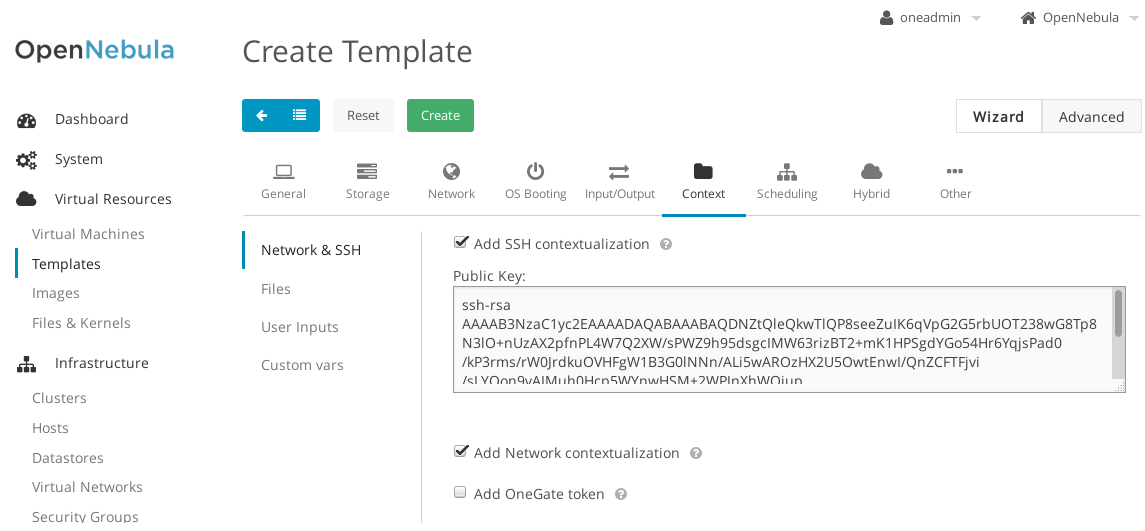



*Figura 9.75. Pestaña Context*

Una vez configuradas todas estas opciones, pulsar sobre Create para crear definitivamente la plantilla.

### 9.9.6 *Crear una máquina virtual*

La última operación a realizar es crear una máquina virtual, es decir, instanciar las plantillas existentes. Para ello, pulsar sobre Resources > Virtual Machines y a continuación, sobre el botón Crear.

Se abrirá un formulario en el que se debe asignar un nombre a la máquina y especificar la plantilla que se va a utilizar (Figura 9.76). Si se necesita instanciar varias veces la misma plantilla escribir el número de máquinas que se desee crear en el campo "*Number of instances*".

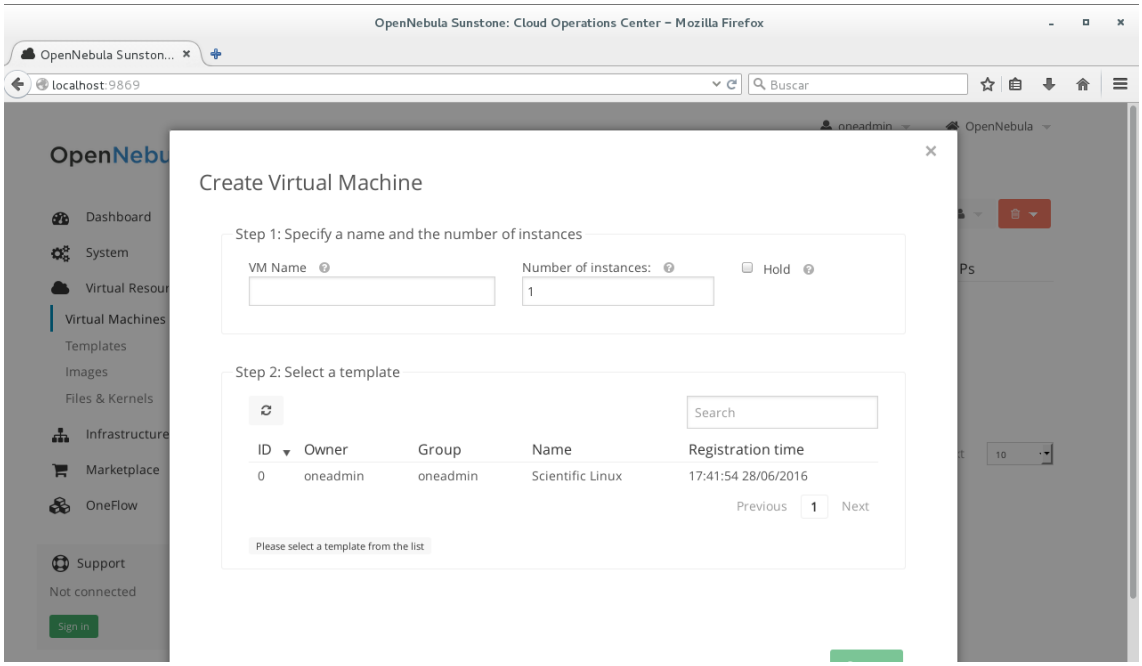



*Figura 9.76. Formulario de creación de una máquina virtual*

Una vez que se ha creado la máquina virtual, ésta permanecerá en estado PENDING (Figura 9.77) hasta que comience a desplegarse, bien de forma manual o bien de manera automática si se detecta un nodo que cumpla los requisitos especificados en la plantilla. Para desplegar manualmente la máquina virtual, pulsar sobre DEPLOY (Figura 9.78) y a continuación, seleccionar el host en el que se desea desplegar la máquina virtual. Mientras se están copiando los archivos en el nodo anfitrión, la máquina virtual mostrará el estado PROLOG (Figura 9.79). Finalizado este proceso, la máquina pasará a mostrar el estado RUNNING.

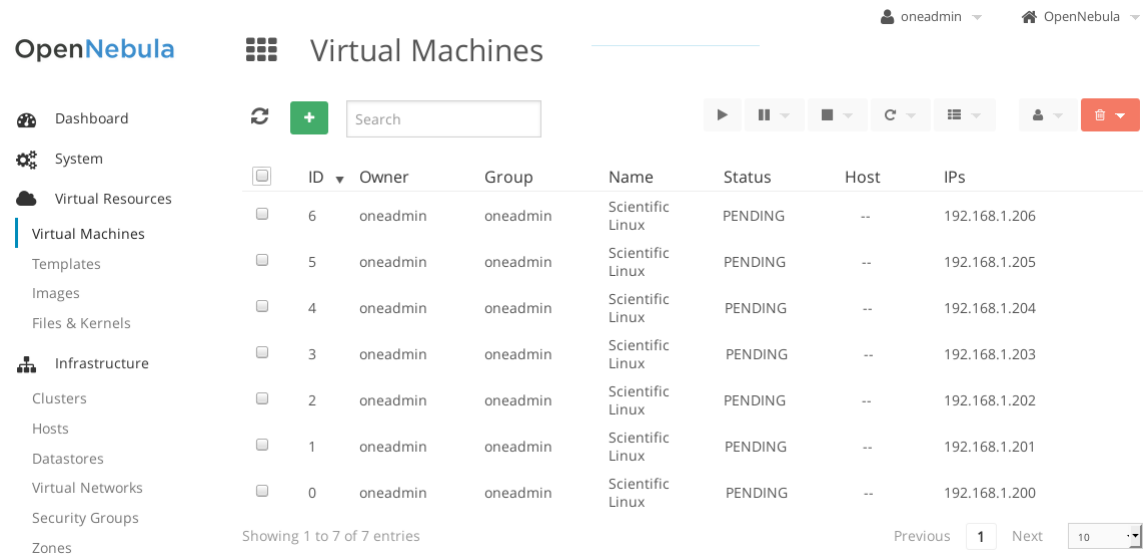



*Figura 9.77. Máquinas pendientes de desplegar*

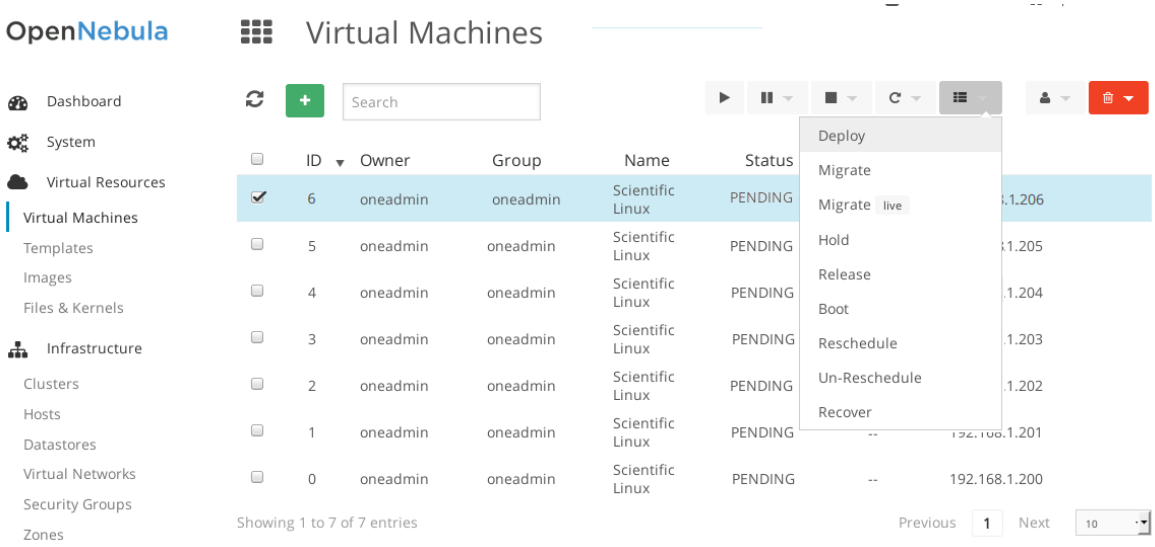



*Figura 9.78. Despliegue de la máquina virtual*

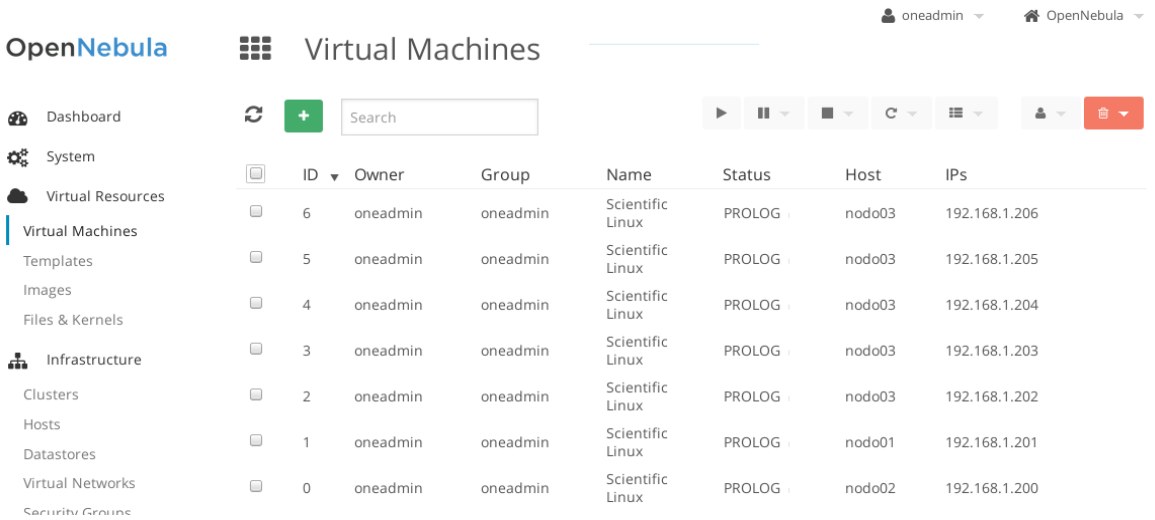



*Figura 9.79. Máquina en proceso de despliegue*

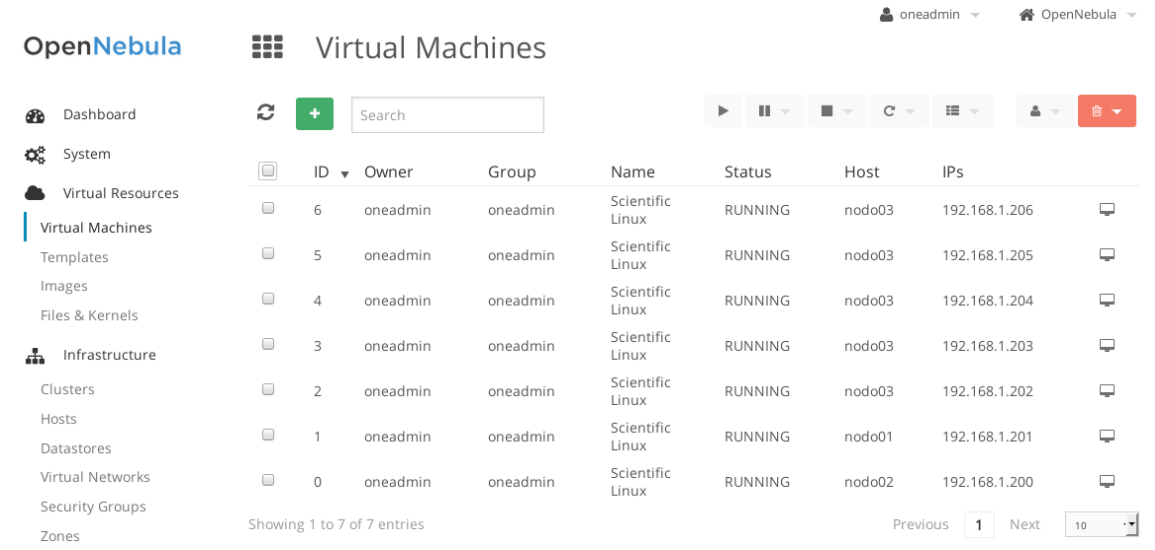



*Figura 9.80. Máquinas en ejecución*

## 9.10 ANEXO X. MVIEW

Los alineamientos producidos en un análisis con BLAST pueden ser representados de una manera visual utilizando diversas herramientas. Una de ellas es Mview [11], una aplicación por consola de comandos que extrae y reformatea el resultado de una búsqueda de secuencia en una base de datos o un alineamiento múltiple de secuencia. También puede ser usado para filtrar, extraer o convertir búsquedas o alineamientos a otros formatos [47].

Los formatos de entrada a este programa son los que se indican a continuación:

- **Búsqueda de secuencias en una base de datos:** BLAST, FASTA suites.
- **Alineamiento múltiple de secuencias:** CLUSTAL, HSSP, MSF, FASTA, PIR, MAF.

Los formatos de salida son:

- HTML, FASTA, CLUSTAL, MSF, PIR, RDB (Separado por tabulaciones).

### 9.10.1 *Instalación*

Para instalar el programa Mview es necesario seguir los siguientes pasos [47]. En primer lugar, una vez que se ha ubicado en el directorio de instalación, descargar el programa ejecutando el siguiente comando:

```
wget http://netix.dl.sourceforge.net/project/bio-mview/bio-mview/mview-1.60.1/mview-1.60.1.tar.bz2
```

A continuación, hay que efectuar una serie de cambios en el fichero mview-1.60.1/bin/mview para que éste funcione adecuadamente. En primer lugar, es necesario establecer una ruta válida para el intérprete de Perl en la máquina local después de #! al comienzo de dicho fichero, por ejemplo.

```
#!/usr/bin/perl
```

A continuación, es necesario modificar la línea use lib '/path/to/mview/lib';, especificando la ruta donde se localizan las librerías que necesita este programa:

```
use lib '/home/sgeadmin/BioCloud/mview-1.60.1/lib';
```

### 9.10.2 *Manual de uso*

En los siguientes apartados, se explicará brevemente el procedimiento de uso y las distintas opciones de esta herramienta con secuencias de nucleótidos y de proteínas [48].

El significado de las distintas opciones utilizadas en el comando anterior, se recogen en la Tabla 9.8:

| Parámetro | Significado | Valor por defecto |
|---|---|---|
| -in | Formato de entrada | - |
| - ruler | Muestra una regla encima de los alineamientos producidos | Off |
| -html | Añade etiquetas de HTML | Off |
| -coloring | Estilo empleado para dar color a los alineamientos | None |
| -bold | Activar el formato de negrita | Off |
| -moltype | Tipo de las secuencias: ADN (dna), ARN (rna), Ácidos nucleicos (na), Aminoácidos (aa) | aa |
| -width | Número de columnas de un bloque de alineamientos | Flat |
| - range | Muestra un subconjunto de los alineamientos producidos | All |

*Tabla 9.8. Opciones del programa MView [Fuente: [48]]*

### *9.10.2.1 Alineamientos de secuencias de nucleótidos*

Para representar visualmente un alineamiento de este tipo, un comando típico sería el siguiente:

```
$ mview -in blast -html head -ruler on -coloring identity -bold -moltype na -width 60 -range 1:60 datos.fasta > datos.html
```

Se obtiene un documento HTML con el que se muestra en la Figura 9.81. Para cada alineamiento producido se indica: puntuación conseguida (bits), el E-valor obtenido (E-value), número de fragmentos asociados (N), orientación de la secuencia problema (qy), orientación de la secuencia local (Hit).

La primera línea se corresponde con la secuencia problema, el resto, son las diferentes secuencias de la base de datos local con las que se ha producido algún alineamiento.

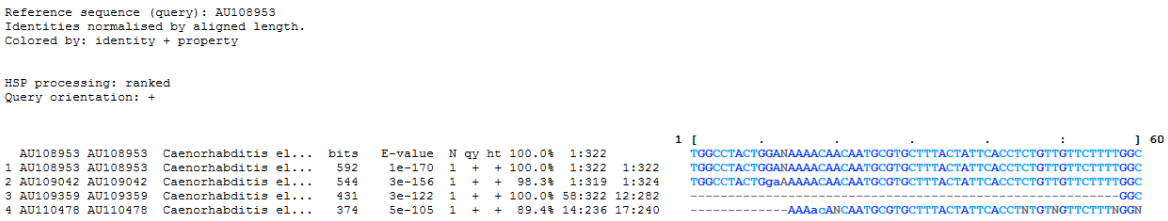



*Figura 9.81. Representación de alineamientos de secuencias de ácidos nucleicos*

La sección de alineamientos se muestra con más detalle en la Figura 9.82.

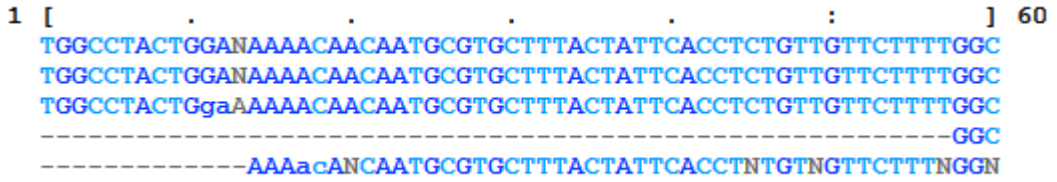



*Figura 9.82. Detalle de los alineamientos*

El esquema de color asigna a las purinas el color azul oscuro, y a las pirimidínicas, el color azul claro. Las letras 'N' de color gris oscuro representan cualquier nucleótido (A, G, C o T). Los demás códigos de colores se recogen en la Figura 9.83.

```
#symbols =>  color              #comment
*          ->  dark-gray        #mismatch
?          ->  light-gray       #unknown
Aa         =>  bright-blue      #purine
Bb         =>  dark-gray        #C or G or T; not A
Cc         =>  dull-blue        #pyrimidine
Dd         =>  dark-gray        #A or G or T; not C
Gg         =>  bright-blue      #purine
Hh         =>  dark-gray        #A or C or T; not G
Kk         =>  dark-gray        #G or T
Mm         =>  dark-gray        #A or C
Nn         =>  dark-gray        #A or C or G or T
Rr         =>  dark-gray        #A or G
Ss         =>  dark-gray        #C or G
Tt         =>  dull-blue        #pyrimidine
Uu         =>  dull-blue        #pyrimidine
Vv         =>  dark-gray        #A or C or G; not T
Ww         =>  dark-gray        #A or T
Xx         ->  dark-gray        #any
Yy         =>  dark-gray        #C or T
```

*Figura 9.83. Código de colores para ADN [FUENTE: [48]]*

Algunas parejas de símbolos están escritas en letras minúsculas, ello significa que la secuencia problema de entrada tenía huecos y éstos se han eliminado para mantener una cadena contigua. De modo que se ha realizado una escisión entre estos símbolos en minúscula (Figura 9.84).

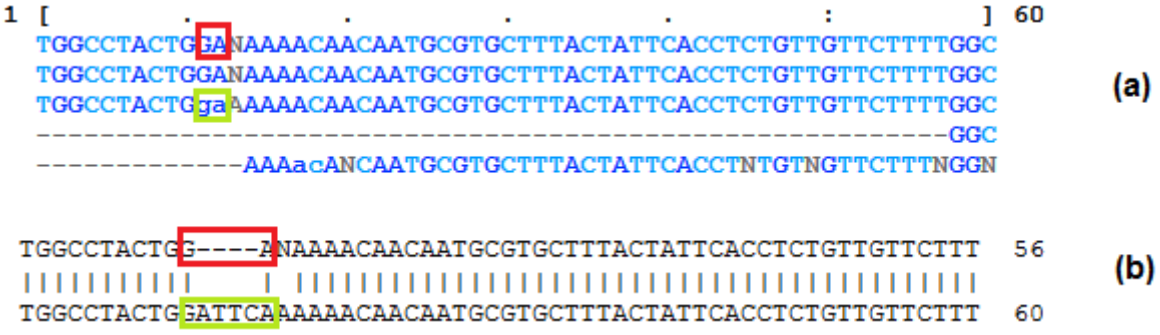



*Figura 9.84. (a) Salida del programa MView (b) Informe de salida de BLASTA*

### 9.10.2.2 Alineamientos de secuencias de proteínas

Si se desea representar un alineamiento de aminoácidos, el comando a emplear es el que se indica a continuación:

*$ mview -in blast -html head -ruler on -coloring any -bold -moltype aa -width 60 -range 1:60 datos.fasta > datos.html*

Nuevamente, se obtiene un documento HTML con el que se puede ver en la Figura 9.85. Para cada alineamiento producido se indica: puntuación conseguida (bits), el E-valor obtenido (E-value) y número de fragmentos asociados (N).

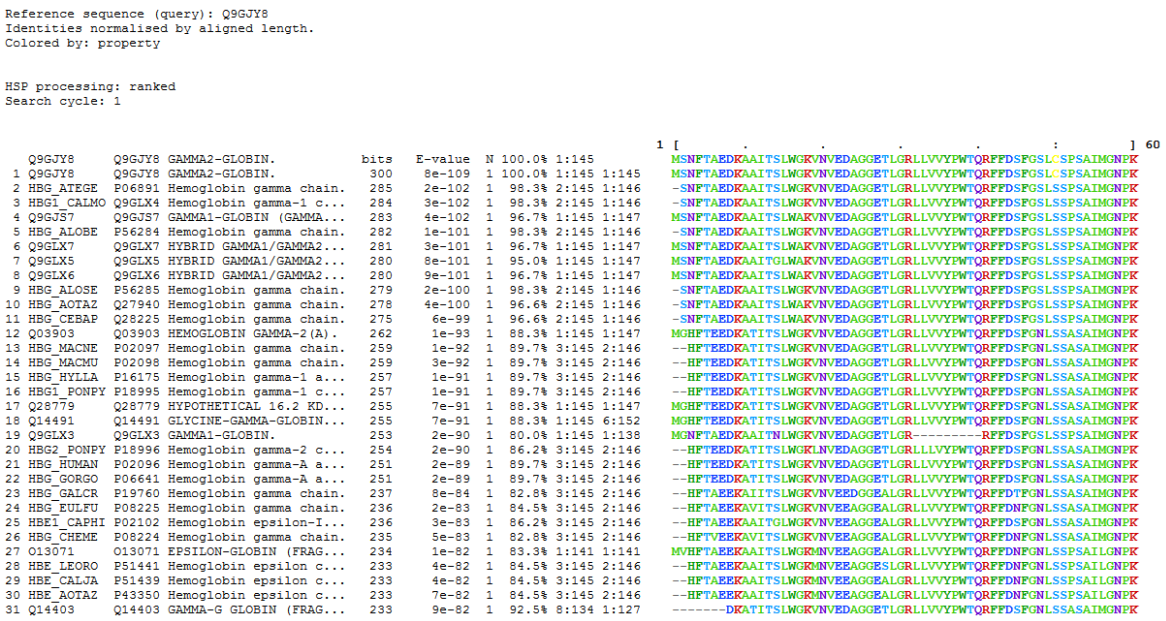


*Figura 9.85. Representación gráfica de alineamientos de proteínas*

Un mayor detalle de los alineamientos se representa en la Figura 9.86. A cada símbolo, cuyo significado se explica en el apartado 9.5.4, se le asigna un color. La paleta de colores utilizada cuando las secuencias son de aminoácidos se recoge en la Figura 9.87.

El formato de color elegido, *any*, asigna a cada símbolo su color correspondiente de acuerdo a la paleta de colores elegida.

Por otro lado, los caracteres '.' y ':' que aparecen en la regla representan agrupaciones de aminoácidos débiles y fuertes, respectivamente.

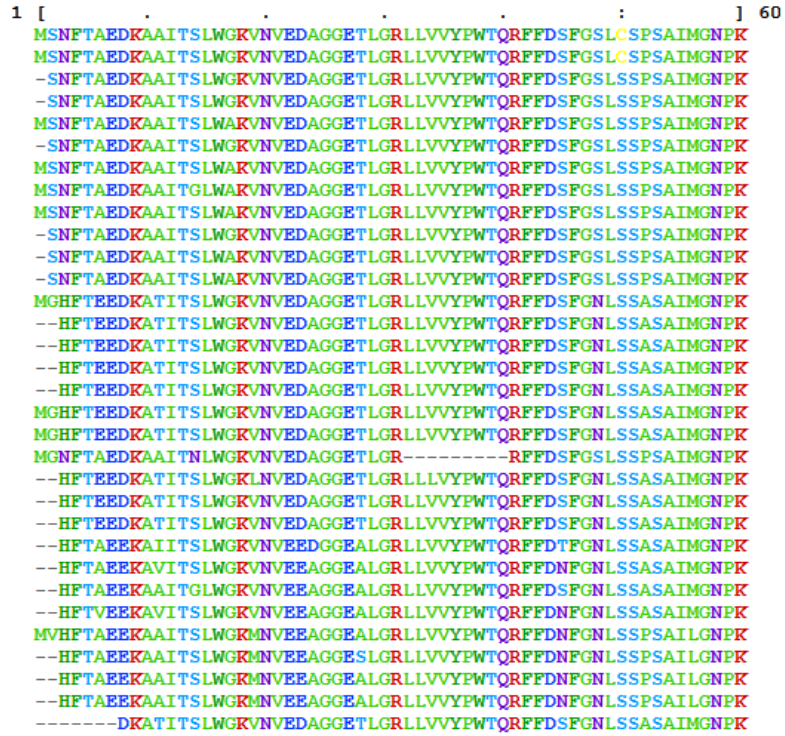



*Figura 9.86. Detalle de los alineamientos*

```
#symbols =>  color              #comment
*         ->  dark-gray         #mismatch
?         ->  light-gray        #unknown
Aa        =>  bright-green      #hydrophobic
Bb        =>  dark-gray         #D or N
Cc        =>  yellow            #cysteine
Dd        =>  bright-blue       #negative charge
Ee        =>  bright-blue       #negative charge
Ff        =>  dark-green        #large hydrophobic
Gg        =>  bright-green      #hydrophobic
Hh        =>  dark-green        #large hydrophobic
Ii        =>  bright-green      #hydrophobic
Kk        =>  bright-red        #positive charge
Ll        =>  bright-green      #hydrophobic
Mm        =>  bright-green      #hydrophobic
Nn        =>  purple            #polar
Pp        =>  bright-green      #hydrophobic
Qq        =>  purple            #polar
Rr        =>  bright-red        #positive charge
Ss        =>  dull-blue         #small alcohol
Tt        =>  dull-blue         #small alcohol
Vv        =>  bright-green      #hydrophobic
Ww        =>  dark-green        #large hydrophobic
Xx        ->  dark-gray         #any
Yy        =>  dark-green        #large hydrophobic
Zz        =>  dark-gray         #E or Q
```

*Figura 9.87. Código de colores para proteínas [FUENTE: [48]]*

## 9.11 ANEXO XI. HOSTING VIRTUAL. ESTUDIO ECONÓMICO

### 9.11.1 *Introducción*

Los avances tecnológicos logrados en los últimos años han propiciado el desarrollo de equipos con mayores prestaciones. Ello ha hecho que los usuarios sean más exigentes y demanden equipos cada vez más potentes. Para cubrir esta demanda, se han ido adoptando distintas estrategias a lo largo del tiempo: Metacomputación, Computación Grid... Sin embargo, en los últimos años se ha dado un paso más y se ha desarrollado un nuevo paradigma, la computación en la nube o *cloud computing*.

El término *cloud computing* puede ser concebido como una concepción tecnológica y un modelo de negocio en el que se prestan servicios de almacenamiento, acceso y uso de recursos informáticos esencialmente radicados en la red [49]**.** De entre todas las soluciones que se ofrecen en la red destacan los servicios de hosting virtual, que permiten a los usuarios almacenar información en servidores en la nube, instalar un amplio abanico de aplicaciones e incluso, desplegar y ejecutar máquinas virtuales.

Hoy día son muchas las empresas que proporcionan servicios de hosting virtual, ofreciendo paquetes y servicios adaptables a las necesidades de cada cliente.

En las siguientes páginas se realizará un análisis de algunas empresas que ofrecen servicios de hosting virtual o *Virtual Data Center (*VDC*)*. Además, llevará a cabo un estudio económico sobre el coste que supondría desplegar una infraestructura compuesta por 7 máquinas virtuales dotadas con 4vCPU y 4 GB de memoria RAM y 150 GB de espacio en disco.

### 9.11.2 *Amazon EC2*

*Amazon Elastic Compute Cloud* (Amazon EC2) es un servicio web que proporciona capacidad de cómputo en la nube de tamaño modificable. Con dicho servicio, el cliente podrá elegir entre varios tipos de instancia (máquina virtual), sistemas operativos y paquetes de software, y seleccionar su propia configuración de memoria, CPU y almacenamiento de la instancia y el tamaño de la partición de arranque óptimo para su sistema operativo y las aplicaciones instaladas en el mismo [50].

Amazon EC2 permite lanzar una instancia de forma gratuita, lo cual puede resultar bastante interesante para probar el servicio antes de contratarlo. La capa gratuita de Amazon Web Services (AWS) incluye 750 horas de instancias t2.micro de Windows y Linux

al mes durante un año. El usuario podrá disfrutar de esta capa gratuita durante 12 meses desde la fecha de inscripción en AWS y recibirá los siguientes servicios de EC2 cada mes:

- 750 horas por mes de uso de instancia t2.micro en Linux, RHEL o SLES.
- 750 horas por mes de uso de instancia t2.micro en Windows.
- 750 horas de *Elastic Load Balancing* más 15 GB de procesamiento de datos.
- 30 GB de Amazon Elastic Block Store en cualquier combinación de almacenamiento de uso general (SSD) o magnético, más 2 millones de E/S (con almacenamiento magnético) y 1 GB de almacenamiento de *snapshots*.
- Se añaden 15 GB de ancho de banda saliente en todos los servicios de AWS.
- 1 GB de transferencia de datos regionales.

Amazon ofrece una calculadora de costo mensual de AWS [51], a través de la cual cada usuario puede estimar el importe de su factura mensual (Figura 9.88).

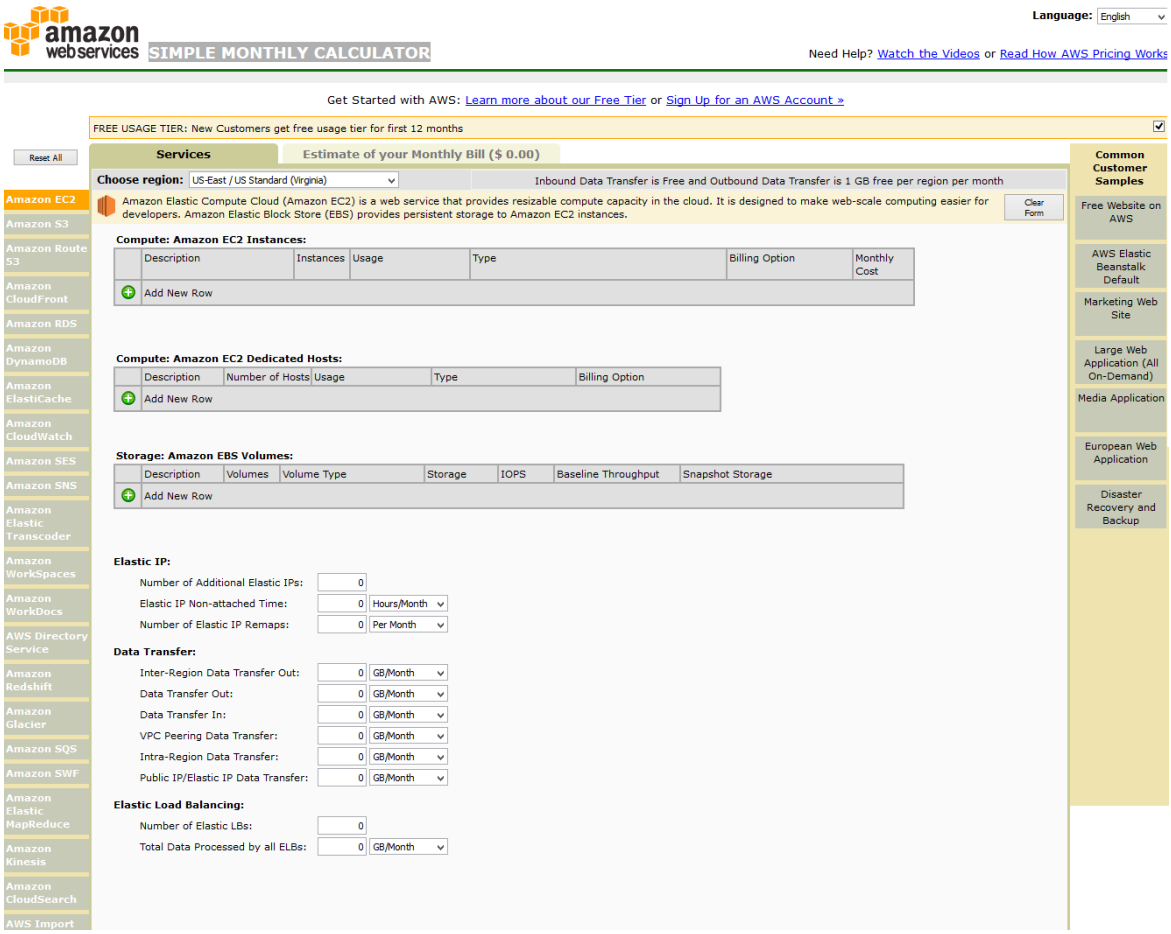



*Figura 9.88. Calculadora de costo de Amazon AWS [FUENTE: [51]]*

Amazon ofrece diferentes tipos de instancias cada una de las cuales están dotadas de distintas combinaciones de capacidad de CPU, memoria, y almacenamiento. Así, el usuario podrá elegir aquella que mejor se adapte a sus necesidades. De entre todos estos

tipos, se ha elegido la instancia m4.xlarge para estimar el costo de los servicios necesitados. Las características de este tipo de instancia se muestran en la Figura 9.89.

Así, si un usuario implementase la infraestructura anteriormente explicada y suponiendo que las instancias están activas durante todo el mes, tendría que pagar mensualmente 1.557,88 dólares, que al cambio son 1.385,76 €/mes, impuestos no incluidos (Figura 9.90).

## M4

Las instancias M4 son la última generación de instancias de uso general. Esta familia proporciona un equilibrio de recursos informáticos, de memoria y red, por lo que constituye una buena opción para muchas aplicaciones.

**Características:**

- Procesadores Intel Xeon® E5-2676 v3 (Haswell) de 2,4 GHz
- Optimizados para EBS de manera predeterminada sin costos adicionales
- Soporte para redes mejoradas
- Equilibrio entre recursos de informática, memoria y red

| Modelo | CPU virtual | Memoria (GiB) | Almacenamiento en SSD (GB) | Ancho de banda de EBS dedicado (Mbps) |
|---|---|---|---|---|
| m4.large | 2 | 8 | Solo EBS | 450 |
| m4.xlarge | 4 | 16 | Solo EBS | 750 |
| m4.2xlarge | 8 | 32 | Solo EBS | 1 000 |
| m4.4xlarge | 16 | 64 | Solo EBS | 2 000 |
| m4.10xlarge | 40 | 160 | Solo EBS | 4 000 |

*Figura 9.89. Características de las instancias M4 [FUENTE: [52]]*

Services | **Estimate of your Monthly Bill ($ 1557.88)**

**Estimate of Your Monthly Bill**

Show First Month's Bill (include all one-time fees, if any)

Below you will see an estimate of your monthly bill. Expand each line item to see cost breakout of each service. To save this bill and input values, click on 'Save and Share' button. To remove the service from the estimate, jump back to the service and clear the specific service's form.

Save and Share

| | | |
|---|---|---|
| Amazon EC2 Service (US-East) | | $ 64.64 |
| Compute: | $ 14.64 | |
| EBS Volumes: | $ 50.00 | |
| EBS IOPS: | $ 0.00 | |
| Amazon EC2 Service (Europe) | | $ 1369.25 |
| Compute: | $ 1352.75 | |
| EBS Volumes: | $ 16.50 | |
| EBS IOPS: | $ 0.00 | |
| AWS Support (Business) | | $ 141.63 |
| Support for all AWS services: | $ 141.63 | |
| Free Tier Discount: | | $ -17.64 |
| Total Monthly Payment: | | $ 1557.88 |

*Figura 9.90. Estimación del coste mensual de EC2*

### 9.11.3 *Microsoft Azure*

Microsoft Azure [53] es una plataforma en la nube ubicada en los centros de datos de la empresa y está conformada por 24 nodos situados en diferentes países repartidos por todo el mundo.

Esta plataforma permite una mayor selección de sistemas operativos, lenguajes de programación, *frameworks* y bases de datos relacionales y no relacionales. Así, las máquinas virtuales desplegadas pueden estar basadas tanto en Windows Server como en Linux. Además, los usuarios disponen de una serie de máquinas virtuales pre-configuradas y listas para ser usadas. Éstas pueden ser descargadas desde el Marketplace de Microsoft Azure [54].

Tal y como se puede observar en la Figura 9.91, el sitio web de Microsoft Azure ofrece una calculadora que permite a cada cliente determinar el importe de los servicios que necesite [55].

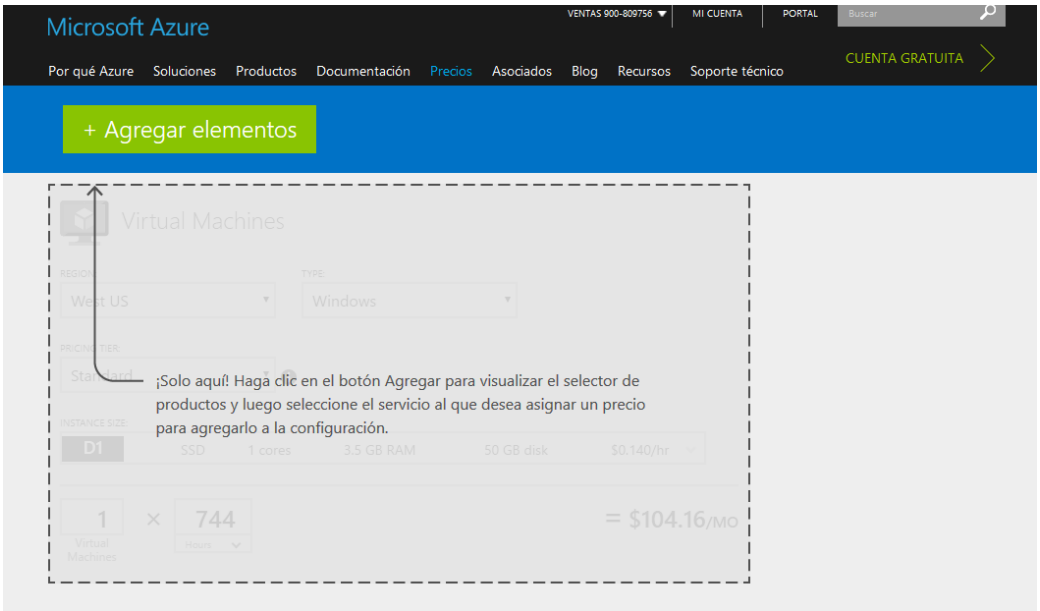



*Figura 9.91. Calculadora de Microsoft Azure [FUENTE: [55]]*

El coste mensual de una infraestructura compuesta por **7 máquinas virtuales dotadas con 4vCPU y 4 GB de memoria RAM** es de 1.054,06 € tal y como se puede observar en la Figura 9.92.

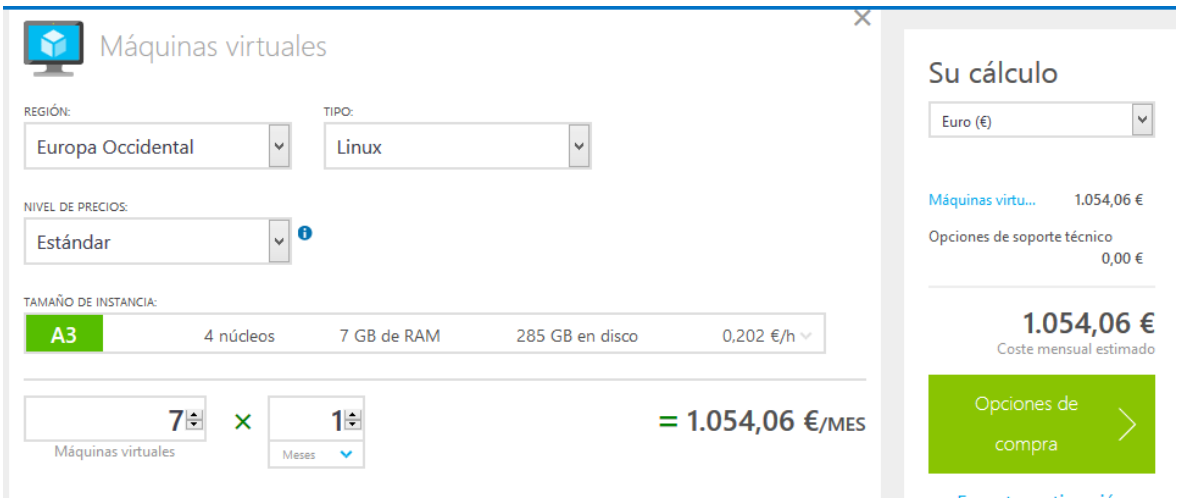



*Figura 9.92. Estimación del coste mensual de Microsoft Azure*

### 9.11.4 *Interoute*

Interoute ofrece servicios de Cloud Computing para *startups*. De entre todos ellos, proporciona un paquete de hosting cloud GRATUITO, llamado ***Interoute JumpStartUp***, concebido para proveer a los desarrolladores y empresas de nueva constitución 1 año de acceso al servicio VDC [56].

Las características de este paquete son las siguientes:

- 2 vCPU.
- 2 GB de RAM.
- 60 GB de espacio de almacenamiento virtual [vHD].
- Acceso a 2 zonas de datos de VDC (París y Berlín).
- Acceso directo por consola.
- Transferencias de datos entrantes y salientes gratuitas entre zonas de VDC.

Así, se pueden crear 1 ó 2 máquinas virtuales de 2vCPU, 2GB de RAM y 60 GB de espacio de almacenamiento o dotadas con 1vCPU, 1 GB de RAM y 30GB, respectivamente.

Esta empresa ofrece una calculadora de precios con la que se puede determinar el precio de los servicios que necesite cada cliente (Figura 9.93):

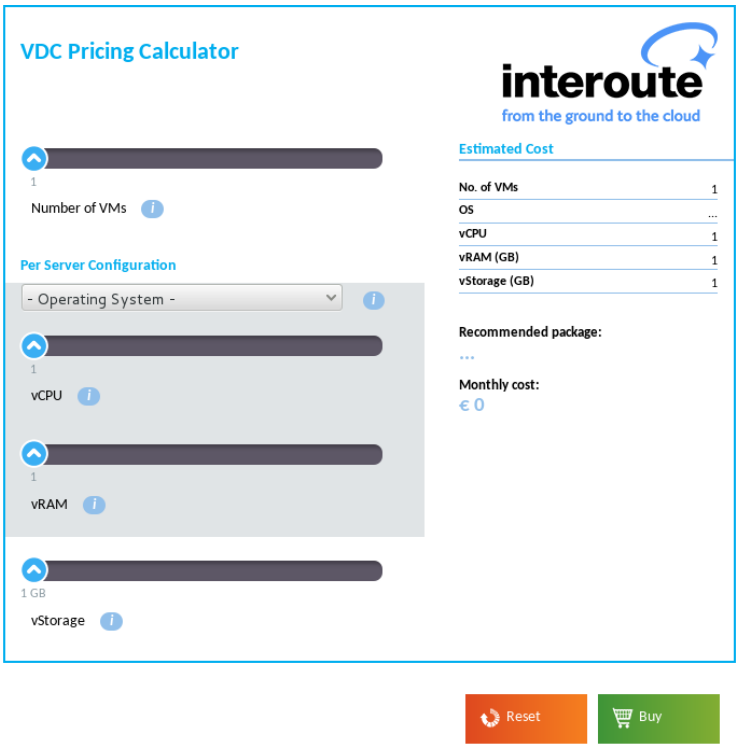



*Figura 9.93. Calculadora de precios de Interoute [FUENTE: [57]]*

Se determinará el coste que tendría una infraestructura cloud compuesta por 7 máquinas virtuales. Cada una de ellas estará dotada de 4vCPU y 4GB de memoria RAM. El importe de este servicio es de 6.173,73 €/mes tal y como muestra la Figura 9.94.

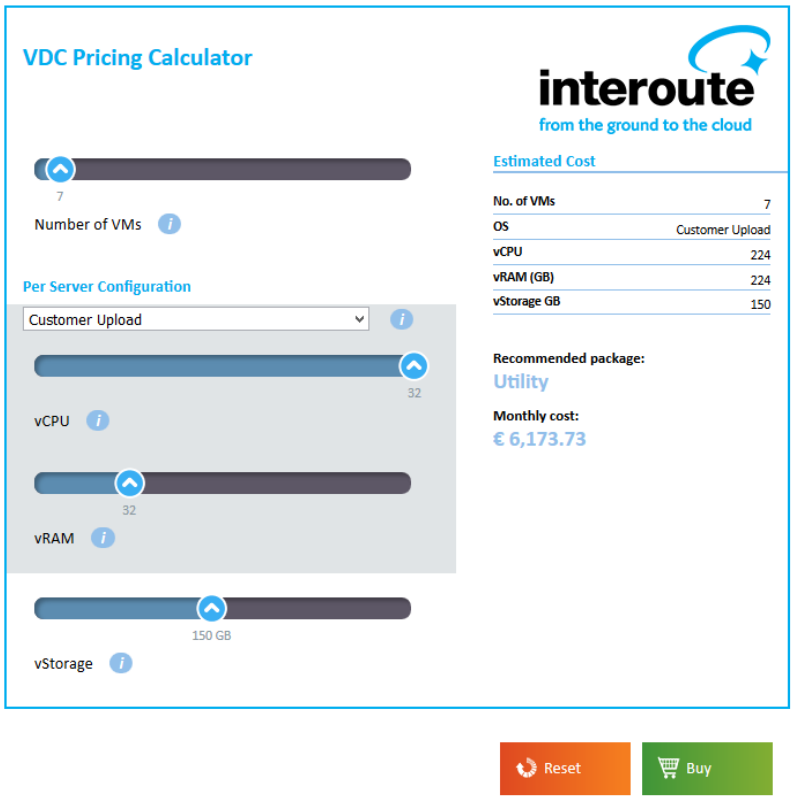



*Figura 9.94. Estimación del coste mensual de Interoute [FUENTE: [57]]*

### 9.11.5 *Acens*

Acens [58] es un proveedor de servicios de hosting. Ofrece a sus usuarios dos modalidades de pago diferentes [59]:

- **Pago por paquetes:** Estos paquetes, llamados ***Cloud Datacenter***, poseen diferentes precios en función del número de máquinas virtuales que se pueden tener encendidas simultáneamente: 2, 5, 10, 25 y 50.
- **Pago por uso:** El usuario pagaría únicamente por recursos consumidos realmente. Permite tener ilimitadas máquinas virtuales, pudiendo estar todas encendidas de manera simultánea.

Para hallar el coste de la infraestructura desplegada, se considerará que el paquete contratado es el de “Pago por uso” (Figura 9.95).

| | PAGO POR USO (€/MES) |
|---|---|
| vRAM (GB) | 10,00 € |
| vCPU de 2Ghz Xeon | 10,00 € |
| Disco | 0,20 €/GB |
| Licencias Windows (por MV) | 14,40 € |
| Red Híbrida | 14,40 € |
| Redes Privadas | 5,00 € |
| VPN IPSec | 5,00 € |
| Balanceo de Carga – grupo de servidores | 7,20 € |
| Direcciones IP | 5,00 € |

*Figura 9.95. Precios paquete "Pago por Uso" [FUENTE: [60]]*

Suponiendo un **servidor virtual basado Linux/UNIX de 32 vCPUs, 32 GB RAM y 150 GB de Disco** que ha estado operativo un mes completo, el precio a pagar por dicho servicio es el que se determina en la Tabla 9.9.

| Recursos | Unidades | €/Hora | Importe Facturable (€) |
|---|---|---|---|
| **vRAM** | 32 | 0,01 | 230,40 |
| **vCPU** | 32 | 0,01 | 230,40 |
| **Disco** | 150 | 0,0003 | 32,40 |
| **TOTAL** | | | **493,30** |

*Tabla 9.9. Estimación del coste mensual de Acens*

### 9.11.6 *Conclusiones*

En la Tabla 9.10 se recapitula el precio de las diferentes soluciones tecnológicas vistas en este anexo. Tal y como puede observarse, la opción más económica sería Acens ya que en este caso se paga única y exclusivamente por las prestaciones de vCPU y memoria RAM que realmente se consumen.

| Solución Tecnológica | Coste mensual (€) |
|---|---|
| **Amazon EC2** | 1.385,76 |
| **Windows Azure** | 1.054,06 |
| **Interoute** | 6.173,73 |
| **Acens** | 493,3 |

*Tabla 9.10. Resumen del coste mensual de las soluciones tecnológicas analizadas*

Por otro lado, y dado que en el contexto de este TFG la infraestructura cloud implementada lleva a cabo las tareas de secuenciación del programa Trinity. Se determinará el coste de cada servicio para un análisis particular suponiendo un tiempo de ejecución de 1 hora. Nuevamente, la infraestructura desplegada está compuesta por 7 máquinas virtuales y cada una de ellas tendrá como mínimo 4vCPU y 4GB de memoria RAM.

El precio a pagar para cada alternativa se recoge en la Tabla 9.11. Dado que la calculadora de Interoute sólo permite estimar el coste mensual y no es posible fraccionar por horas o días, ha resultado imposible determinar su coste por análisis.

| Solución Tecnológica | Coste/análisis (€) |
|---|---|
| **Amazon EC2** | 62,28 |
| **Windows Azure** | 1,42 |
| **Interoute** | - |
| **Acens** | 0,685 |

*Tabla 9.11. Coste de cada solución por análisis*

## 9.12 ANEXO XII. CATÁLOGO DE PATRONES DE DISEÑO

En este anexo se habla brevemente de los distintos patrones de diseño empleados en la implementación de la aplicación web BioCloud. Para cada uno de ellos se indica su objetivo o intención, la estructura que presenta en cuanto a clases que componen en el patrón y los participantes, es decir, las entidades que intervienen en el mismo [61], [62].

### 9.12.1 *Patrón Command*

La finalidad de este patrón es encapsular peticiones en forma de objetos, permitiendo parametrizar clientes con distintas peticiones, encolarlas, llevar un registro de las peticiones realizadas y deshacer el efecto de estas peticiones. La interfaz *Validator* implementada para comprobar los datos introducidos en un formulario web es un caso particular de patrón *Command*.

#### *9.12.1.1 Estructura*

Este patrón posee la estructura que se muestra en la Figura 9.96.

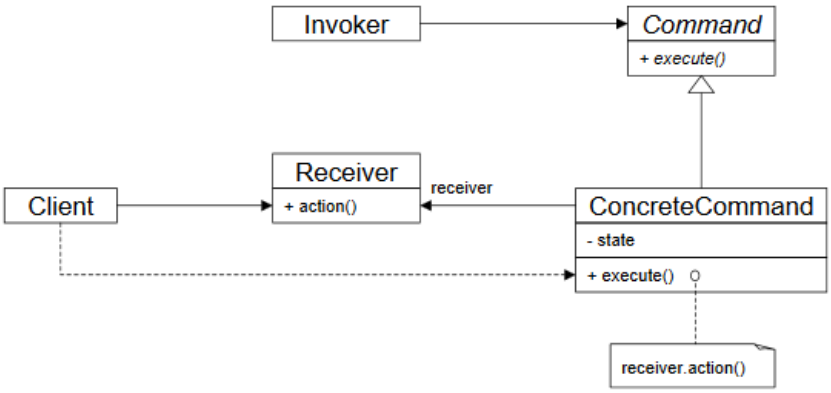



*Figura 9.96. Estructura del patrón Command [FUENTE: [62]]*

#### *9.12.1.2 Participantes*

- **Command**: Declara una interfaz para la ejecución de operaciones.
- **ConcreteCommand**: Implementa la interfaz de ejecución.
- **Client:** Crea un comando concreto.
- **Invoker**: Contiene el comando asociado a la petición.
- **Receiver**: Conoce cómo llevar a cabo operaciones asociadas a una petición.

### 9.12.2 *Patrón Adapter*

Convierte la interfaz de una clase en otra que el cliente espera. Permite que clases con interfaz incompatibles puedan trabajar juntas.

#### *9.12.2.1 Estructura*

Este patrón posee la estructura que se muestra en la Figura 9.97.

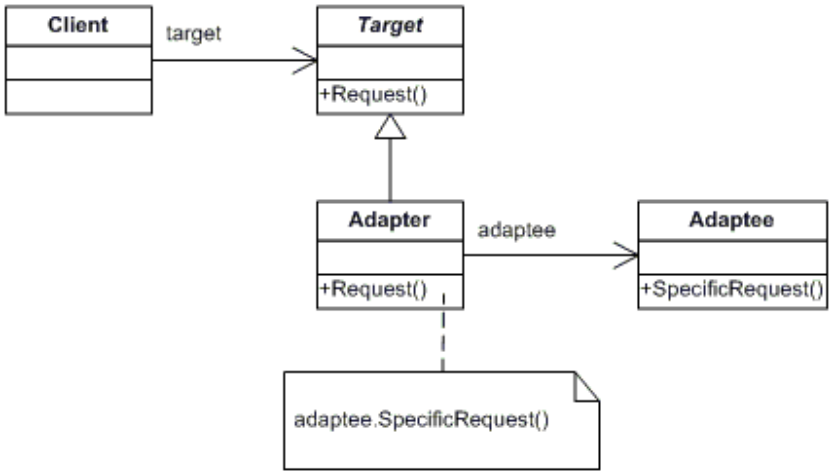



*Figura 9.97. Estructura del patrón Adapter [FUENTE: [62]]]*

#### *9.12.2.2 Participantes*

- **Target:** Define la interfaz específica del dominio que el *Client* utiliza.
- **Adapter:** Adapta la interfaz de *Adaptee* a la definida por el *Target*.
- **Adaptee:** Define una interfaz existente que necesita ser adaptada.
- **Client:** Colabora con los objetos que implementan la interfaz *Target*.

### 9.12.3 *Patrón DAO*

Tiene como objetivo abstraer y parametrizar los accesos a los orígenes de datos, ocultando completamente detalles de la implementación de esta fuente de datos a los objetos cliente. Este patrón forma parte del catálogo *Core J2EE Patterns* [63].

#### 9.12.3.1 Estructura

Posee la estructura que se muestra en la Figura 9.98.

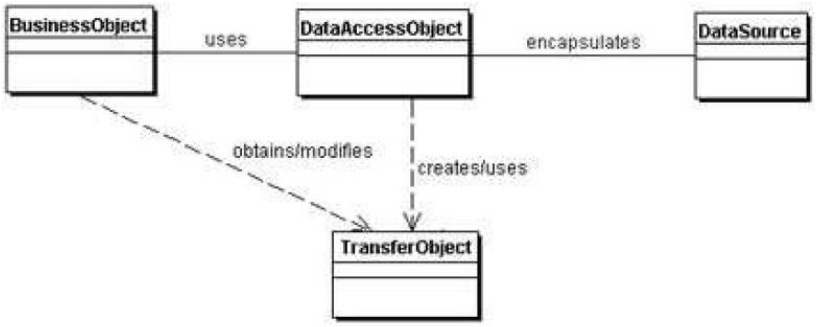



*Figura 9.98. Estructura del patrón DAO [FUENTE: [63]]*

#### 9.12.3.2 Participantes y Responsabilidades

- **BussinesObject:** Representa el cliente que desea acceder al origen de datos para obtener y almacenar información.
- **DataAccessObject:** Abstrae la implementación de los datos permitiendo un acceso transparente al origen de datos. En este objeto se delegan las operaciones de consulta e inserción de datos.
- **DataSource:** Representa la implementación de un origen de datos como puede ser una base de datos relacional (MySQL, SQLite) o un repositorio de documentos XML
- **TransferObject:** Representa el objeto usado como portador de la información.