# UNACLOUD MSA: PLATAFORMA BASADA EN UNACLOUD PARA LA GENERACIÓN Y ANÁLISIS DE ALINEAMIENTOS MÚLTIPLES DE SECUENCIAS

**ARTHUR ALEJANDRO OVIEDO URAZMETOV**

**UNIVERSIDAD DE LOS ANDES**
**FACULTAD DE INGENIERÍA**
**INGENIERÍA DE SISTEMAS Y COMPUTACIÓN**
**BOGOTÁ**
**2011**

# UNACLOUD MSA: PLATAFORMA BASADA EN UNACLOUD PARA LA GENERACIÓN Y ANÁLISIS DE ALINEAMIENTOS MÚLTIPLES DE SECUENCIAS

**ARTHUR ALEJANDRO OVIEDO URAZMETOV**

**Tesis de Grado presentada como requisito para optar al título de**
**Magíster en Ingeniería de Sistemas y Computación**

**Director:**
**Ph. D. Harold Enrique Castro Barrera**
**Profesor Asociado**

**UNIVERSIDAD DE LOS ANDES**
**FACULTAD DE INGENIERÍA**
**INGENIERÍA DE SISTEMAS Y COMPUTACIÓN**
**BOGOTÁ**
**2011**

# Tabla de Contenido

# Índice de Figuras

# Índice de Tablas

# 1 RESUMEN

La bioinformática consiste en la aplicación de técnicas informáticas para el apoyo de diferentes proyectos biológicos. Estos proyectos incluyen desde grandes despliegues de infraestructura para compartir información, hasta modelamientos matemáticos y computacionales de procesos biológicos. Uno de los problemas con el que se enfrentan los investigadores en biología es el de Alineamiento Múltiple de Secuencia (MSA por sus siglas en inglés), en el cual se parte de conjunto de secuencias de ADN o de proteínas, y se busca alinearlas de tal manera que se logre identificar información pertinente como patrones comunes en las secuencias. La información obtenida en el alineamiento múltiple, sirve de entrada a diferentes tipos de tareas como inferencias filogenéticas y análisis de metagenómica.

Existen diferentes herramientas informáticas que permiten encontrar aproximaciones heurísticas, sin embargo, encontrar el mejor alineamiento es un problema computacionalmente intratable. En la práctica se utilizan diferentes programas computacionales que permiten encontrar alineamientos múltiples, pero los resultados presentados por cada programa no necesariamente son completamente confiables. Adicionalmente, si bien se cuenta con un número muy grande de herramientas disponibles en internet para hacer análisis más profundos sobre los resultados encontrados, en la práctica no existen maneras muy amigables para integrarlas, por lo que en muchos casos, los investigadores biológicos no realizan un estudio detallado de su alineamiento múltiple con lo que se pierde información importante.

En este trabajo se presenta UnaCloud MSA, una plataforma computacional para la generación y análisis de alineamiento múltiples de secuencias que le permiten al investigador en biología tener una mayor información sobre sus datos. Adicionalmente UnaCloud MSA integra de manera transparente una infraestructura dedicada que soporta su funcionalidad, con una infraestructura oportunista basada en UnaCloud con lo que se aprovecha el poder computacional ocioso de los laboratorios de computación. Esta integración se hace a través de un esquema de despliegue dinámico que permite que la infraestructura oportunista se utilice cuando el nivel de carga supere cierto umbral. Se presenta el análisis, diseño, implementación, y evaluación de la solución, se presentan las conclusiones del trabajo y se presentan futuras líneas posibles de trabajo.

En este trabajo se utiliza la terminología técnica en inglés. Las palabras y siglas grid, middleware, cluster, gap, workflow por nombrar las más frecuentes, son dejadas en ese idioma de forma intencional.

# 2 MOTIVACIÓN

## 2.1 Bioinformática

La bioinformática consiste en el empleo de herramientas computacionales para generar, analizar, almacenar y distribuir información biológica. Debido a diferentes desarrollos tecnológicos como técnicas más eficientes y menos costosas de secuenciación de ADN se ha presentado una explosión en la cantidad de información biológica disponible. Debido a esta gran cantidad de información, surge la necesidad de integrar herramientas computacionales para analizar, almacenar, compartir y presentar toda esta información. [Bio01]

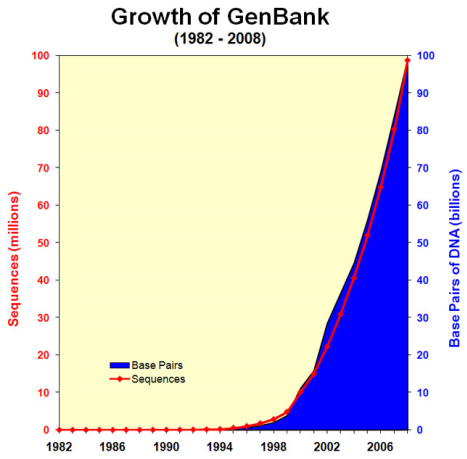


**Ilustración 1: Crecimiento de Genbank en los últimos años [Gen01]**

El nacimiento de la bioinformática ha traído nuevos campos de acción a diferentes ramas de la informática: El estudio de diferentes algoritmos para analizar la información contenida en la secuencias de ADN, técnicas en manejo de datos distribuidos para compartir información entre diferentes laboratorios de investigación, infraestructuras distribuidas como grid para compartir recursos computacionales y poder completar trabajos en tiempos razonables, entre muchas otras.

Todos estos nuevos retos que trae consigo el estudio biológico desde un punto de vista computacional, abren un nuevo espectro de posibilidades para la aplicación de la informática en nuevos campos. Debido a este amplio panorama y a los nuevos retos que se introducen, surge la motivación de desarrollar un trabajo en bioinformática.

## 2.2 Campus Grid[Cam01]

Campus-Grid Uniandes consiste en un esfuerzo conjunto entre diferentes grupos en la Universidad de los Andes para brindar una infraestructura grid para el desarrollo de diferentes proyectos. Se busca con Campus Grid el poder incrementar la capacidad computacional a nivel de universidad, para atraer a nuevos usuarios de diferentes áreas de investigación que requieran de un alto poder computacional para llevar a cabo sus investigaciones. En el marco de campus grid se han desarrollado diferentes trabajos de investigación en áreas de bioinformática con el proyecto Modelamiento de *Bacillus thuringiensis* [Bac01], en el campo de ingeniería industrial y optimización se desarrollaron los proyectos JG$^2$A[Jga01] y pALS[Pal01].

El gran atractivo que ofrece Campus Grid Uniandes es el hecho de integrar tanto una infraestructura de clústers dedicados con una infraestructura grid oportunista. El modelo de computación oportunista se ofrece como una

alternativa muy viable en términos económicos puesto que permite agregar a la capacidad de cómputo global, los recursos ofrecidos por computadores de trabajo personal dispersos a lo largo de toda la universidad.

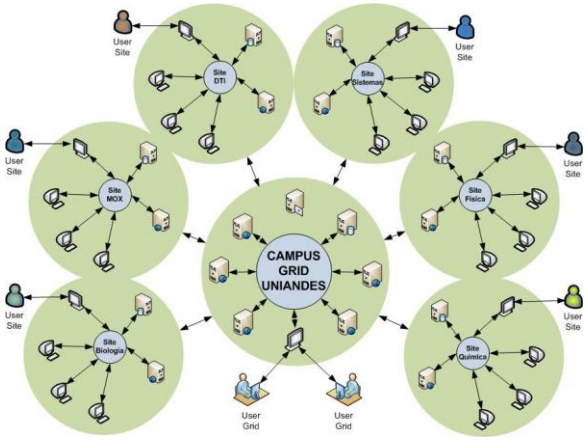


**Ilustración 2: Arquitectura de CampusGrid, adaptado de [Cam01]**

Campus Grid se ofrece como una arquitectura muy interesante para el desarrollo de proyectos puesto brinda la posibilidad de disponer de una gran cantidad de poder computacional. Por esta razón, surge el interés por realizar un proyecto que utilice esta infraestructura con el ánimo de apoyar la investigación biológica.

## 2.3 UnaCloud

UnaCloud[Una01], el cual es uno de los proyectos de mayor importancia dentro de nuestro grupo de investigación, ofrece una implementación del modelo de Infraestructura como Servicio (IaaS por sus siglas en inglés) basado en una infraestructura oportunista. Esta implementación ofrece diferentes recursos computacionales como lo son procesamiento, almacenamiento, memoria y red para la ejecución de diferentes tipos de aplicaciones. UnaCloud está basado en dos grandes pilares: Una estrategia oportunista mediante la cual se aprovechan recursos computacionales ociosos de una manera no intrusiva y una estrategia de virtualización que permite la ejecución de ambientes personalizados. UnaCloud se constituye como la evolución de otro proyecto, UnaGrid[Una02], complementado el portafolio de servicios que ofrecía como lo son: Usabilidad a través de una interface web fácil de utilizar y auto-servicio al permitir que los usuarios configuren y desplieguen sus propios clusters, entre otros.

UnaCloud se presenta como una herramienta sobre la cual se pueden desarrollar aplicaciones y servicios basados en infraestructuras oportunistas de manera sencilla. Debido a esto, se propone investigar la integración de este trabajo con la infraestructura que ofrece UnaCloud con el ánimo de utilizar los servicios ya desarrollados para el aprovechamiento de los recursos computacionales disponibles.

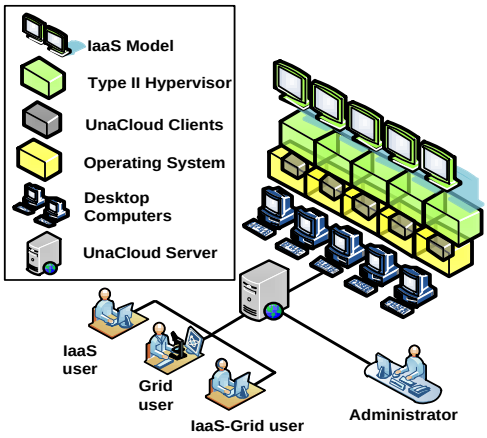


**Ilustración 3: Arquitectura de UnaCloud, adaptado de [Una01]**

## 2.4 Alineamiento Múltiple de Secuencias

El ánimo para investigar el problema del alineamiento múltiple de secuencias surge de un proceso de diálogo con grupos de investigadores biológicos. En estas reuniones se identifica que uno de los procesos claves al momento de realizar un trabajo de investigación biológica consiste en una primera recolección de información de secuencias ya sea mediante procesos experimentales o a través de herramientas bioinformáticas como bases de datos online. Una vez identificadas las secuencias de ADN o proteínas que se vas a trabajar, el paso siguiente consiste en el de su alineamiento múltiple. En este paso, a través de un programa de computador, se reorganizan las secuencias de tal manera que se identifiquen estructuras comunes y se puedan hacer predicciones sobre su funcionalidad y sobre su historia evolutiva. Este proceso tiene suma importancia ya que brinda una primera visión de la información que se contiene. Dependiendo de la información que se consigue en este proceso y dependiendo de la pregunta biológica que se esté investigando, el workflow de trabajo biológico toma rumbos diferentes. A pesar de que muchas investigaciones bioinformáticas parten de un alineamiento múltiple, éste es un problema complejo computacionalmente. La importancia a nivel biológico y su complejidad en términos matemáticos y algorítmicos generan el interés por su estudio e investigación.
La creación de un alineamiento múltiple constituye el primer paso de diferentes flujos de trabajo en biología. Algunos de los diferentes usos que tienen los alineamientos múltiples son:

- Extrapolación – Culpabilidad por Asociación: Si un gen o proteína cuya función es desconocida, se alinea contra un conjunto de otras secuencias y se identifican fragmento conservados comunes, la información que se tiene sobre las otras secuencias se puede transferir a la secuencia desconocida. Esto ha permitido la anotación de muchísimas secuencias de ADN sin llegar a realizar procedimientos experimentales utilizando solo la información que proveen los alineamientos múltiples. A través de este proceso se determinan dominios proteicos y motivos.

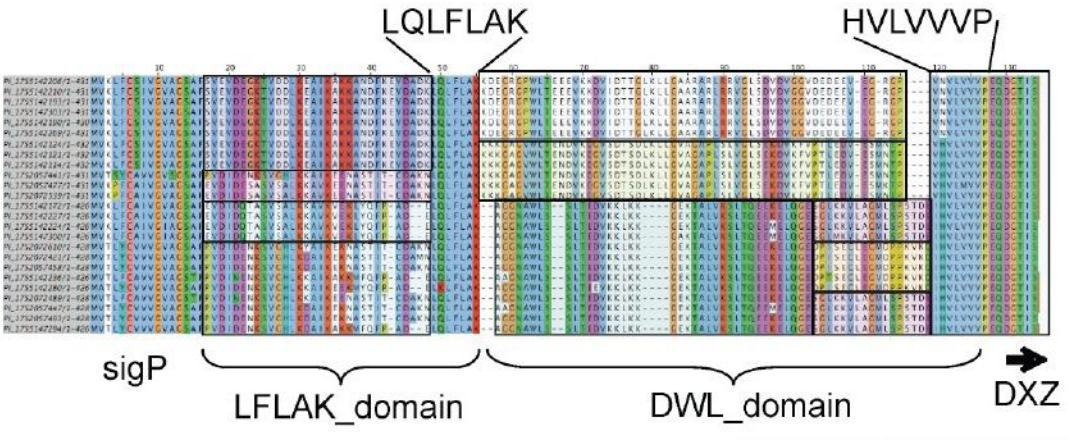


**Ilustración 4: Identificación de dominios en un alineamiento múltiple**

- Análisis de filogenias: En muchos casos es de interés el conocer el proceso evolutivo que han seguido diferentes especies de organismos. Generalmente esta información se presenta como un árbol filogenético donde se evidencia que los organismos más cercanos en el árbol están más cerca a nivel evolutivo. Para la generación de estos árboles se utilizan alineamientos múltiples sobre diferentes genes.
- Identificación de regiones de poca información: Al realizar un alineamiento múltiple, las regiones que no pertenecen a dominios claros, tienden a causar problemas al momento se ser utilizados en otros programas. Identificar que regiones no son muy confiables, le permite al biólogo decidir si utiliza las secuencias en su totalidad o si prefiere enmascarar estas regiones. Esta decisión depende del análisis que se esté realizando.

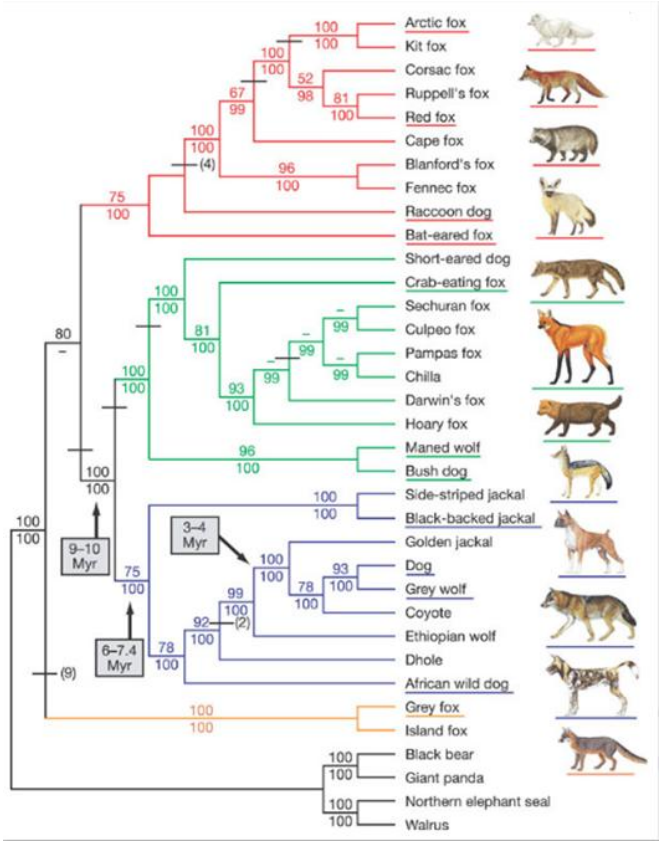


**Ilustración 5: Representación de un árbol filogenético entre las diferentes especies de caninos**

# 3 MARCO CONCEPTUAL

## 3.1 Similitud de Secuencias

Gran parte del análisis genético se basa en el concepto de similitud de secuencias. Conociendo qué tan similares o diferentes son dos secuencias se pueden lograr un gran número de inferencias. Estudiar la similitud de secuencias requiere incorporar conceptos biológicos puesto que las secuencias estudiadas provienen de organismos, ya sean de ADN o de proteínas y conceptos matemáticos puesto que las matemáticas permiten definir objetivamente ciertos conceptos.

### 3.1.1 Mutaciones

Los cambios más comunes que puede sufrir una secuencia son de sustitución (Cuando un residuo es intercambiado por otro), de inserción (Cuando un residuo externo entra a la secuencia) o de eliminación (Cuando un residuo de la secuencia es eliminado). [Fun01]

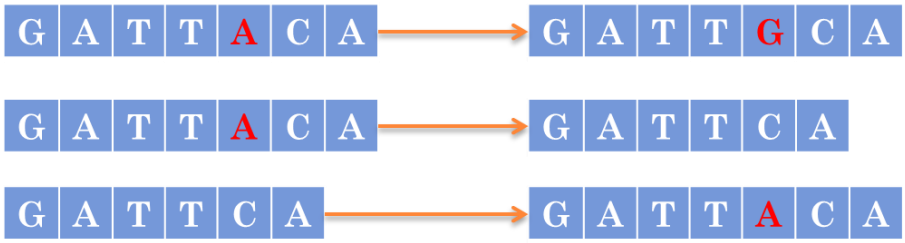


**Ilustración 6: Tipos de mutaciones**

### 3.1.2 Distancia de Edición

Para definir qué tan similares son dos secuencias se pueden utilizar diferentes aproximaciones. Una de las métricas más utilizada sobre secuencias de texto es la distancia de Levenshtein[Lev01], que cuenta el mínimo número de cambios necesarios para transformar una secuencia en otra. Los cambios admitidos para esta métrica son precisamente los que se toman en cuenta desde el punto biológico: Substitución, inserción y eliminación.

| b | i | o | l | o | g | i | a | |
|---|---|---|---|---|---|---|---|---|
| _ | i | o | l | o | g | i | a | Eliminación |
| _ | i | _ | l | o | g | i | a | Eliminación |
| _ | i | _ | l | o | g | i | c | a | Inserción |
| _ | i | _ | l | o | g | i | c | o | Substitución |

**Ilustración 7: Distancia de edición de la cadena "biologia" a "ilógico". La distancia entre las cadenas es 4, porque se necesitan 4 operaciones para tranformar una secuencia en la otra.**

## 3.2 Alineamientos Pareados [Alg01, Gri01]

### 3.2.1 Definición

Necesitamos definir formalmente el concepto de alineamiento de dos secuencias de caracteres. Esta definición se aplica de igual manera a secuencias de ADN como a secuencias de proteínas.
Para poder definir un alineamiento pareado entre dos cadenas necesitamos los siguientes conceptos:
Un alfabeto $\Sigma$: Si estamos hablando de un alineamiento de cadenas de ADN, el alfabeto estaría compuesto por (A,C,G,T). Si por otra parte, hablamos de un alineamiento de proteínas, el alfabeto que se emplea es el de los 20 aminoácidos (A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y)
Un símbolo de (-) tal que $-\notin\Sigma$. Este símbolo se denomina gap, y en un alineamiento representa una inserción o deleción en alguna de las cadenas, es decir que un caracter en una de las dos cadena ha sido eliminado.
Un alineamiento de dos secuencias S,T es una pareja de nuevas secuencias (S', T') sobre el alfabeto $\Sigma$ U {-} en las que se satisfacen las siguientes condiciones:

- La longitud de S' y T' son iguales: |S'|=|T'|, a esta longitud la llamamos *l*.
- Al eliminar todos los símbolos de gap en S', nos da como resultado S, análogamente con T' y T.
- Si en la posición i de S' está el caracter de gap, en la posición i de T' no está el caracter gap.
- Simétricamente, si en la posición i de T' hay un caracter de gap, en S' no hay un caracter de gap en la posición i.

Si tomamos las cadenas S= ATTCGGCAT y T=TTGCAGCTCT
Dos posibles alineamientos de las cadenas serían:

```
S'=    A T T - C A G C G - T
T'=    - T T G C A G C T C T

S'=    A T - T C A G C G - T -
T'=    - T T G C - A G C T C T
```

### 3.2.2 Función de Score

Intuitivamente, de los dos alineamientos propuestos anteriormente vemos que el primero es mejor en la medida que nos permite identificar más características comunes entre las dos cadenas. Para formalizar el concepto de que tan bueno es un alineamiento, se introduce el concepto de función de Score.
Para definir la función de score de un alineamiento, primero se introduce una función f de match/missmatch que recibe dos elementos del alfabeto $\Sigma$ U {-} y retorna un número real.
Para el caso de secuencias de ADN, se puede utilizar la función definida:

$$f(a,b) = \begin{cases} +2\ si\ a = b \\ -1\ si\ a \neq b \\ -2\ si\ a = - \\ -2\ si\ b = - \end{cases}$$

La función que se define, por lo general es simétrica y en general f(a,-) = f(-,b) = g y se conoce como penalización de gap.
La función de score de un alineamiento es entonces la suma de la función de match evaluada en cada una de las *l* columnas del alineamiento.

$$Score(S', T') = \sum_{i=1}^{l} f(s'_i, t'_i)$$

Utilizando los valores presentados para la función f, el cálculo de la función de score del primer alineamiento nos da:
Score (ATT-CAGCG-T, -TTGCAGCTCT) = -2+2+2-2+2+2+2+2-1-2+2 = 7
Score (AT-TCAGCG-T-, -TTGC-AGCTCT) = -2+2-2-1+2-2-1-1-1-2-1-2 = -11
En términos de la función de score ahora, vemos que el primer alineamiento nos da un mayor valor, por lo que es un alineamiento mejor que el segundo.

Habiendo definido el score de un alineamiento, podemos definir claramente el problema de encontrar el mejor alineamiento como un problema de optimización donde se desea maximizar la función objetivo de score.

### 3.2.3 Matrices de Substitución

Al definir la función de match/missmatch en cadenas de ADN, se utilizó un criterio donde cada substitución de un carácter por otro diferente tiene igual costo. Para cadenas de proteínas esta suposición no se puede tomar ya que los diferentes aminoácidos no tienen las mismas propiedades moleculares, y en algunos casos cuando ambos aminoácidos tienen características similares, su alineamiento puede aportar a la función de score. También aminoácidos muy diferentes entre ellos, penalizan de manera considerable su alineamiento. Para tomar en cuenta este aspecto, se crearon las matrices de substitución.

Existen dos grandes grupos de matrices que son los más utilizados en la actualidad: Las Matrices PAM y las Matrices BLOSUM.

#### 3.2.3.1 *Matrices PAM*

Las matrices PAM[Pam01] (Puntos Aceptados de Mutación) fueron propuestas por Dayhoff y se basan en observaciones experimentales de las diferentes tasas de cambio de unos aminoácidos por otros en diferentes procesos biológicos. PAM no es una matriz, sino que constituye una familia de matrices relacionadas entre sí. La matriz PAM1 indica la probabilidad de que una secuencia tenga una mutación puntual por cada 100 aminoácidos. Las siguientes matrices PAM se calculan multiplicando sucesivamente la matriz PAM1. Así, la matriz PAM50 se obtiene multiplicando PAM1 por sí misma 50 veces. A medida que se trabaja con secuencias más distantes, se prefiere la utilización de matrices PAM con valores más grandes.

PAM 250 Matrix

```
#
# This matrix was produced by "pam" Version 1.0.7 [13-Aug-03]
#
# PAM 250 substitution matrix, scale = ln(2)/3 = 0.231049
#
# Expected score = -0.844, Entropy = 0.354 bits
#
# Lowest score = -8, Highest score = 17
#
   A  R  N  D  C  Q  E  G  H  I  L  K  M  F  P  S  T  W  Y  V  B  Z  X  *
A  2 -2  0  0 -2  0  0  1 -1 -1 -2 -1 -1 -3  1  1  1 -6 -3  0  0  0  0 -8
R -2  6  0 -1 -4  1 -1 -3  2 -2 -3  3  0 -4  0  0 -1  2 -4 -2 -1  0 -1 -8
N  0  0  2  2 -4  1  1  0  2 -2 -3  1 -2 -3  0  1  0 -4 -2 -2  2  1  0 -8
D  0 -1  2  4 -5  2  3  1  1 -2 -4  0 -3 -6 -1  0  0 -7 -4 -2  3  3 -1 -8
C -2 -4 -4 -5 12 -5 -5 -3 -3 -2 -6 -5 -5 -4 -3  0 -2 -8  0 -2 -4 -5 -3 -8
Q  0  1  1  2 -5  4  2 -1  3 -2 -2  1 -1 -5  0 -1 -1 -5 -4 -2  1  3 -1 -8
E  0 -1  1  3 -5  2  4  0  1 -2 -3  0 -2 -5 -1  0  0 -7 -4 -2  3  3 -1 -8
G  1 -3  0  1 -3 -1  0  5 -2 -3 -4 -2 -3 -5  0  1  0 -7 -5 -1  0  0 -1 -8
H -1  2  2  1 -3  3  1 -2  6 -2 -2  0 -2 -2  0 -1 -1 -3  0 -2  1  2 -1 -8
I -1 -2 -2 -2 -2 -2 -3 -2  5  2 -2  2  1 -2 -1  0 -5 -1  4 -2 -2 -1 -8
L -2 -3 -3 -4 -6 -2 -3 -4 -2  2  6 -3  4  2 -3 -3 -2 -2 -1  2 -3 -3 -1 -8
K -1  3  1  0 -5  1  0 -2  0 -2 -3  5  0 -5 -1  0  0 -3 -4 -2  1  0 -1 -8
M -1  0 -2 -3 -5 -1 -2 -3 -2  2  4  0  6  0 -2 -2 -1 -4 -2  2 -2 -2 -1 -8
F -3 -4 -3 -6 -4 -5 -5 -5 -2  1  2 -5  0  9 -5 -3 -3  0  7 -1 -4 -5 -2 -8
P  1  0  0 -1 -3  0 -1  0  0 -2 -3 -1 -2 -5  6  1  0 -6 -5 -1 -1  0 -1 -8
S  1  0  1  0  0 -1  0  1 -1 -1 -3  0 -2 -3  1  2  1 -2 -3 -1  0  0  0 -8
T  1 -1  0  0 -2 -1  0  0 -1  0 -2  0 -1 -3  0  1  3 -5 -3  0  0 -1  0 -8
W -6  2 -4 -7 -8 -5 -7 -7 -3 -5 -2 -3 -4  0 -6 -2 -5 17  0 -6 -5 -6 -4 -8
Y -3 -4 -2 -4  0 -4 -4 -5  0 -1 -1 -4 -2  7 -5 -3 -3  0 10 -2 -3 -4 -2 -8
V  0 -2 -2 -2 -2 -2 -2 -1 -2  4  2 -2  2 -1 -1 -1 -1  0 -6 -2  4 -2 -2 -1 -8
B  0 -1  2  3 -4  1  3  0  1 -2 -3  1 -2 -4 -1  0  0 -5 -3 -2  3  2 -1 -8
Z  0  0  1  3 -5  3  3  0  2 -2 -3  0 -2 -5  0  0 -1 -6 -4 -2  2  3 -1 -8
X  0 -1  0 -1 -3 -1 -1 -1 -1 -1 -1 -1 -1 -2 -1  0  0 -4 -2 -1 -1 -1 -1 -8
* -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8 -8  1
```

**Ilustración 8: Matriz de score PAM250**

### 3.2.3.2 *Matrices BLOSUM*

Las matrices BLOSUM[Blo01] (Block Substitution Matrix) corrigen algunos problemas que presentan las matrices PAM para secuencias divergentes. El cálculo de estas matrices tiene en cuenta bloques, o segmentos muy conservados en diferentes secuencias. BLOSUM también constituye una familia de matrices, el número que acompaña a una matriz de este tipo indica el nivel de similaridad entre las secuencias observadas para crear la matriz. Así, BLOSUM62 fue construida utilizando secuencias que eran idénticas hasta un 62%. BLOSUM62 es una de las matrices más utilizadas como parámetro por defecto en diferentes programas como BLAST.

```
Ala  4
Arg -1  5
Asn -2  0  6
Asp -2 -2  1  6
Cys  0 -3 -3 -3  9
Gln -1  1  0  0 -3  5
Glu -1  0  0  2 -4  2  5
Gly  0 -2  0 -1 -3 -2 -2  6
His -2  0  1 -1 -3  0  0 -2  8
Ile -1 -3 -3 -3 -1 -3 -3 -4 -3  4
Leu -1 -2 -3 -4 -1 -2 -3 -4 -3  2  4
Lys -1  2  0 -1 -3  1  1 -2 -1 -3 -2  5
Met -1 -1 -2 -3 -1  0 -2 -3 -2  1  2 -1  5
Phe -2 -3 -3 -3 -2 -3 -3 -3 -1  0  0 -3  0  6
Pro -1 -2 -2 -1 -3 -1 -1 -2 -2 -3 -3 -1 -2 -4  7
Ser  1 -1  1  0 -1  0  0  0 -1 -2 -2  0 -1 -2 -1  4
Thr  0 -1  0 -1 -1 -1 -1 -2 -2 -1 -1 -1 -1 -2 -1  1  5
Trp -3 -3 -4 -4 -2 -2 -3 -2 -2 -3 -2 -3 -1  1 -4 -3 -2 11
Tyr -2 -2 -2 -3 -2 -1 -2 -3  2 -1 -1 -2 -1  3 -3 -2 -2  2  7
Val  0 -3 -3 -3 -1 -2 -2 -3 -3  3  1 -2  1 -1 -2 -2  0 -3 -1  4
    Ala Arg Asn Asp Cys Gln Glu Gly His Ile Leu Lys Met Phe Pro Ser Thr Trp Tyr Val
```

Ilustración 9: Matriz de score BLOSUM62

### 3.2.4 Alineamientos Globales vs Alineamientos Locales

Dependiendo del tipo de análisis a veces se desea encontrar que tan similares son un par de secuencias en su totalidad, o en alguna región específica. Dependiendo del interés biológico, se presentan los alineamientos globales y locales. Un alineamiento global busca alinear ambas secuencias en su totalidad mientras que un alineamiento local busca alinear subcadenas de las secuencias originales que sean más similares de acuerdo a la función de scoring. Un alineamiento local permite, entre otros, detectar dominios entre proteínas: Si bien las

proteínas pueden ser muy diferentes, al compartir una región conservada como un dominio, el alineamiento local puede ayudar a identificarlo.

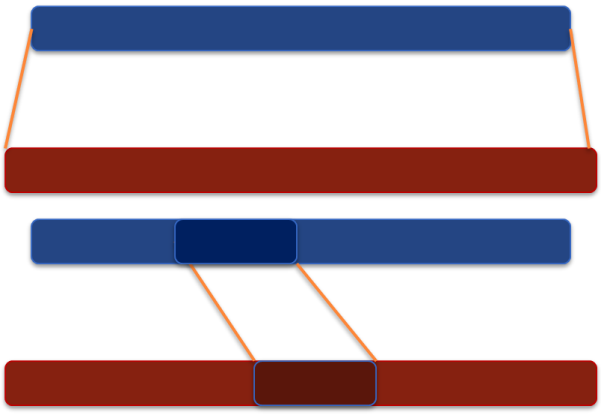

**Ilustración 10: Diferencia entre un alineamiento global (Arriba) y un alineamiento local (Abajo)**

### 3.2.5 Algoritmos

El proceso básico para encontrar el mejor alineamiento consiste en encontrar el máximo puntaje posible entre todos los alineamientos, y luego reconstruirlo en un proceso de *traceback*. El cálculo del mejor puntaje del alineamiento se calcula utilizando programación dinámica:
Conociendo el mejor puntaje logrado con los prefijos de ambas cadenas, se puede ir construyendo la solución en cada paso. La función de score varía dependiendo del tipo de alineamiento que se esté calculando.

#### 3.2.5.1 *Needleman-Wunch para alineamientos globales* [Ned01]

La función de puntuación entre dos cadenas S y T se define en términos recursivos de la siguiente manera:

$$\max -score(i,j) = max \begin{cases} max - score\ (i-1,j) + f(S_i,-) \\ max - score\ (i,j-1) + f(-,T_j) \\ max - score\ (i-1,j-1) + f(S_i,T_j) \end{cases}$$

$$\text{max-score}(0,0) = 0$$

Esta recursión nos indica que conociendo el mejor puntaje de los respectivos prefijos de ambas cadenas, podemos ir calculando incrementalmente este puntaje. Los términos $f(S_i,-)$ y $f(-,T_j)$ indican las penalizaciones por insertar gaps en el alineamiento y el término $f(S_i,T_j)$ indica la función de score definida al momento de evaluar un alineamiento al alinear el carácter i-ésimo de la secuencia S con el carácter j-ésimo de la secuencia T.
El cálculo de todas las entradas de la matriz, tiene una complejidad tanto temporal como espacial de O(m*n) para dos cadenas de longitud *m* y *n* respectivamente. Para simplificar, decimos que el algoritmo toma $O(l^2)$ donde *l* es la longitud promedio de ambas cadenas.

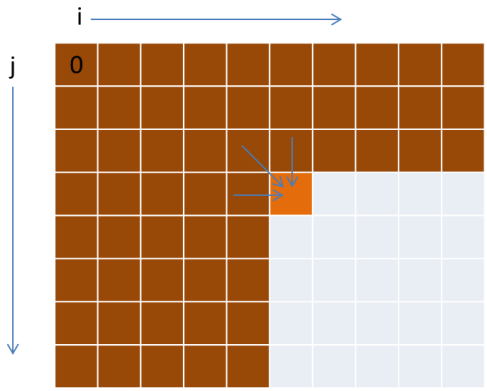


**Ilustración 11: Diagrama de dependencia para el cálculo del score del mejor alineamiento**

### 3.2.5.2 *Smith-Waterman para alineamientos locales* [Smi01]

El cálculo para el mejor alineamiento local se basa en la misma estrategia que en el caso anterior, pero se hace una leve modificación de la función objetivo:

$$\max -score(i,j) = max \begin{cases} max - score\ (i-1,j) + f(S_i,-) \\ max - score\ (i,j-1) + f(-,T_j) \\ max - score\ (i-1,j-1) + f(S_i,T_j) \\ 0 \end{cases}$$

$$\text{max-score}(0,0) = 0$$

Esto implica que si en algún momento el puntaje del alineamiento pasa a ser negativo, se prefiere descartar el fragmento alineado.
El cálculo de esta matriz, al tener que evaluar cada una de las casillas, también implica que la complejidad temporal de este algoritmo es de $O(l^2)$

### 3.2.5.3 *Reconstrucción del alineamiento*

Al momento de calcular el puntaje, en cada paso se escoge la mejor opción entre alinear ambos caracteres, introducir un gap en la primera secuencia o introducir un gap en la segunda secuencia. Para reconstruir el alineamiento, se necesita guardar en cada paso, la decisión tomada y a partir de la última casilla del alineamiento se procede siguiendo el camino indicado por las opciones tomadas.

| | - | A | G | C | C | T | A | C | T |
|---|---|---|---|---|---|---|---|---|---|
| - | 0 | -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 |
| A | -1 | 2 | 1 | 0 | -1 | -2 | -3 | -4 | -5 |
| G | -2 | 1 | 4 | 3 | 2 | 1 | 0 | -1 | -2 |
| A | -3 | 0 | 3 | 2 | 1 | 0 | 3 | 2 | 1 |
| C | -4 | -1 | 2 | 5 | 4 | 3 | 2 | 5 | 4 |
| A | -5 | -2 | 1 | 4 | 3 | 2 | 5 | 4 | 3 |
| C | -6 | -3 | 0 | 3 | 6 | 5 | 4 | 7 | 6 |
| T | -7 | -4 | -1 | 2 | 5 | 8 | 7 | 6 | 9 |

**Ilustración 12: Cálculo de la matriz de puntuación y camino del alineamiento para las cadenas AGCCTACT y AGACACT con los siguientes parámetros: Penalizacion por gap=-1, match=+2, mismatch= -3**

Como se observa en el ejemplo, en algunos casos dos o tres valores para una casilla nos dan iguales, en estos casos, cualquier camino seguido produce un alineamiento con igual puntaje. Al momento de generar el alineamiento, se puede escoger cualquier camino posible.

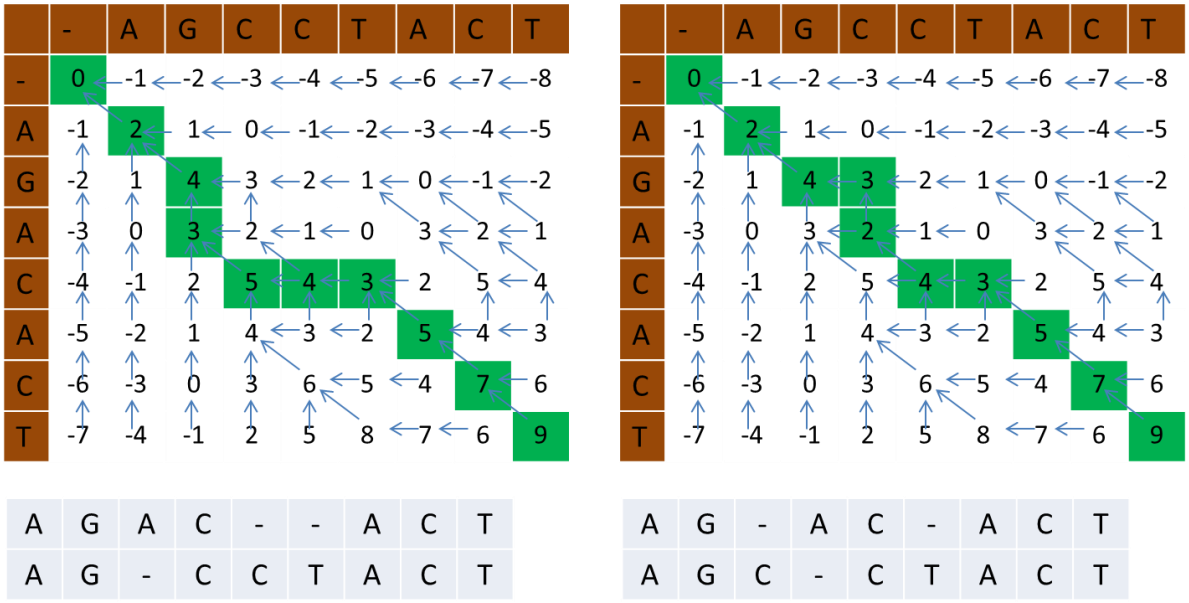


**Ilustración 13: Diferentes caminos producen diferentes alineamientos con el mismo puntaje**

### 3.2.5.4 *Modelo de gaps extendidos y algoritmo de Gotoh* [Got01]

El modelo presentado anteriormente implica que la inserción de un gap en una secuencia es independiente de la posición. Biológicamente este modelo no es muy adecuado, puesto que el trabajo necesario para crear un gap es diferente al trabajo necesario para extenderlo.

Debido a esto, uno de los modelos propuestos más utilizados para tomar en cuenta esta restricción es el de gaps afines (Affine gap penalties).

En este caso, la penalización a la función de score de un gap de longitud $\gamma$ se calcula como:

$$g + e*(\gamma - 1)$$

Donde g es la penalización por la apertura del gap y *e* es el costo de extensión. En este modelo la penalización de un gap crece proporcionalmente a su longitud.

Incorporar esta condición en los algoritmos presentados anteriormente implicaría hacer una revisión de toda una columna o fila al momento de evaluar el valor de una casilla. Esto aumentaría la complejidad del algoritmo de $O(l^2)$ a $O(l^3)$.

Para evitar aumentar la complejidad temporal del algoritmo, Gotoh propuso el cálculo de dos matrices auxiliares V y H que guardan la puntuación del mejor alineamiento que termina en un gap horizontal o vertical (En la secuencia S o en la secuencia T). Si bien se utilizan ahora tres matrices en vez de una, la complejidad espacial no se aumenta ya que sigue siendo $O(l^2)$.

La matriz de score ahora se calcula apoyándose en las otras dos matrices.

$$\max -score(i,j) = max \begin{cases} H(i,j) \\ V(i,j) \\ max - score\ (i-1, j-1) + f(S_i, T_j) \end{cases}$$

$$H(i,j) = max\begin{cases} max - score(i-1,j) + g \\ H(i-1,j) + e \end{cases}$$
$$V(i,j) = max\begin{cases} max - score(i,j-1) + g \\ V(i-1,j) + e \end{cases}$$

### 3.2.5.5 *Miller-Myers*[Mil01]

En muchos casos el factor limitante al momento de realizar un alineamiento entre dos secuencias es el espacio en memoria. Fácilmente una cadena de ADN puede contener miles de residuos y al momento de ejecutar los algoritmos presentados en la sección anterior, pueden fácilmente fallar debido a limitaciones en memoria principal. Debido a esto, se presenta la necesidad de encontrar un algoritmo más eficiente en espacio que las implementaciones anteriores que toman $O(l^2)$.

El algoritmo se basa en la observación de que para calcular solamente el costo del mejor alineamiento, basta un algoritmo lineal en espacio, es decir que toma O(l). Para lograr esto, basta observar que para calcular el valor de una casilla de la matriz, únicamente se necesitan conocer tres valores: Uno que está en la misma columna y dos que están en la columna anterior. Luego para calcular una columna, basta con tener en memoria principal la columna actual que se está calculando y la inmediatamente anterior.

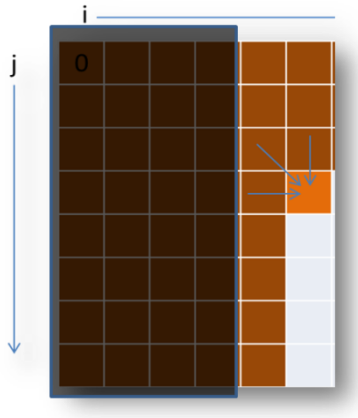


**Ilustración 14: Columnas necesarias para realizar el cálculo del puntaje**

Sin embargo, esta información no es directamente aplicable al problema original, puesto que se necesita almacenar la información para realizar el traceback y solamente con esta información no es suficiente.

Una segunda observación clave es que al momento de almacenar las decisiones tomadas, podemos reconstruir todos los posibles alineamientos, siguiendo todas las posibles rutas desde la casilla final hasta la casilla inicial. Sin embargo, al tener todos estos alineamientos el mismo puntaje, en términos prácticos no es necesario conocerlos todos, sólo con uno bastaría.

El algoritmo lineal en espacio, se basa en un algoritmo propuesto por Hirschberg en 1975 [Hir01] para el problema de encontrar la máxima subsecuencia común más grande entre dos secuencias. El algoritmo propuesto por Myers y Miller, se basa en encontrar recursivamente el punto medio del alineamiento óptimo utilizando el algoritmo lineal en espacio, partiendo las secuencias en 2. Sobre la primera mitad se aplica recursivamente el procedimiento y sobre la segunda mitad se aplica en orden inverso, es decir desde el final hacia el inicio.

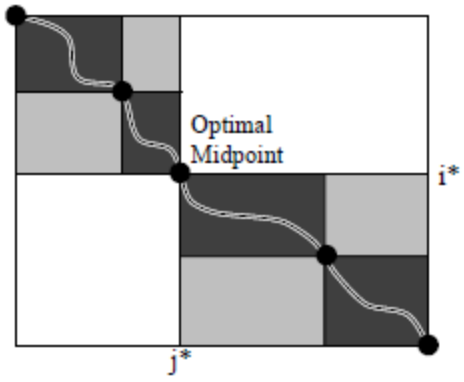


**Ilustración 15: Cálculo recursivo del punto medio en cada alineamiento óptimo**

## 3.3 Alineamientos Múltiples [Fun01, Alg01, Mul01]

### 3.3.1 Definiciones

#### 3.3.1.1 *Alineamiento Múltiple de Secuencias*

La definición empleada para los alineamientos pareados, se puede extender ahora a un alineamiento múltiple de manera sencilla. Un alineamiento sobre las cadenas S1, S2,…, Sn sobre un alfabeto Σ, es otro conjunto de secuencias S1', S2',…, Sn' sobre el alfabeto Σ U {-} donde el carácter {-} no pertenece al alfabeto original. Además todas las secuencias del alineamiento múltiple tienen la misma longitud.

```
>Secuencia1                    >Secuencia1                    >Secuencia1
MSSTSTTIRSHSSSRRGFSAGSARL      -MSSTSTTIRSHSSSRRGFSAGSARL------ MS-STSTTIRSHSSSRRGFSAGSAR------L
>Secuencia2                    >Secuencia2                    >Secuencia2
MASTSTTIGSRSLYGLGG             -MASTSTTI----GSRSLYGLGG--------- MA-STSTTIG----SRSLYGLG---------G
>Secuencia3                    >Secuencia3                    >Secuencia3
MSRQSTVTFHSGSRRGFSTAS          --MSRQSTVTFHSGSRRGFSTAS--------- MS-RQSTVTF-HSGSRRGFSTA---------S
>Secuencia4                    >Secuencia4                    >Secuencia4
KSIRKSQTITHCGFSAGSARLPGAGRS KSIRKSQTI-----THCGFSAGSARLPGAGRS KSIRKSQTI-----THCGFSAGSARLPGAGRS
>Secuencia5                    >Secuencia5                    >Secuencia5
MSRKSQTITHRGFSAGSARLPGA        -MSRKSQTI-----THRGFSAGSARLPGA--- MS-RKSQTI-----THRGFSAGSARLP---GA
>Secuencia6                    >Secuencia6                    >Secuencia6
MASTSTTIRSSRRGFSAGSARLPG       -MASTSTTIR---SSRRGFSAGSARLPG---- MA-STSTTIRS---SRRGFSAGSARLP----G
```

**Ilustración 16: Grupo de Secuencias de Ejemplo y 2 diferentes alineamientos múltiples**

#### 3.3.1.2 *Secuencia Consenso*

Una manera sencilla de resumir un alineamiento múltiple es a través de la secuencia consenso. Esta secuencia consiste en por cada columna del alineamiento, seleccionar el aminoácido que más veces se presenta. Diferentes programas computacionales toman diferentes decisiones respecto a columnas donde no hay una letra que sea mayoría. Algunos utilizan letras en minúscula para representar letras que se presentaron en la columna más que otras letras pero que no alcanzan a ser mayoría. La siguiente figura presenta la secuencia consenso de los dos alineamientos presentados en la figura 16.

```
>Consenso1
MSxrtSTTIxxxxxSRRGFSAGSARlpxxxxs

>Consenso2
xMAStSTTIxxxxxSRRGFSAGSARLpgxxxx
```

**Ilustración 17: Secuencias consenso de los 2 alineamientos**

#### 3.3.1.3 *Perfil*

Si bien un consenso es útil para resumir un alineamiento, uno de sus mayores problemas es la pérdida de información asociada. Una representación de un alineamiento que no pierde tanta información y que sirve también para resumir un alineamiento es un perfil. Para cada columna de un perfil, se indica la frecuencia de cada uno de los aminoácidos que se encontraron en el alineamiento en esa columna. Así por lo tanto, no solamente se tiene la información del aminoácido más ocurrente, sino que todos los aminoácidos en esa columna se tienen en cuenta.

### 3.3.2 Métricas de Evaluación

Dado un alineamiento múltiple, también nos interesa tener un método que nos permita evaluarlos y asignar un puntaje a cada uno. Debido a que un alineamiento múltiple nos ofrece más posibilidades, también tenemos diferentes maneras de evaluarlos.

#### 3.3.2.1 *Sum of Pairs Score*

Esta métrica consiste en una generalización de la función de score utilizada para alineamientos pareados. Dado un alineamiento múltiple S1', S2',…, Sn' se define el SP-Score como:

$$SP - Score(S1', S2', \dots, Sn') = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=1}^{l} f({Si'}_k, {Sj'}_k)$$

Donde *l* es la longitud de todas las cadenas del alineamiento múltiple, *n* es el número de secuencias en el alineamiento y f es la misma función que evalúa el alineamiento de dos caracteres como se definió para un alineamiento pareado. En resumen, esta función consiste en aplicar la función de score a cada par de secuencias y luego sumar todos los resultados.

#### 3.3.2.2 *Distance to Consensus*

En esta métrica se calcula la distancia de cada una de las secuencias del alineamiento a la secuencia consenso. La fórmula para su cálculo sería:

$$Distance\_Consensus\_Score(C, S1', S2', \dots, Sn') = \sum_{i=1}^{n} \sum_{k=1}^{l} f({Si'}_k, C_k)$$

Donde $C_k$ es el aminoácido en la posición k-ésima de la secuencia consenso, *l* es la longitud de las secuencias alineadas, $S'i_k$ es el carácter en la posición k-ésima de la secuencia i del alineamiento.

## 3.4 Dificultad del cálculo de Alineamientos

Al igual que la definición de alineamiento múltiple es una generalización de la definición de alineamiento pareado, el algoritmo para encontrar el mejor alineamiento para maximizar la función de score basado en programación dinámica para algoritmos pareados también se puede generalizar para un número arbitrario de secuencias.

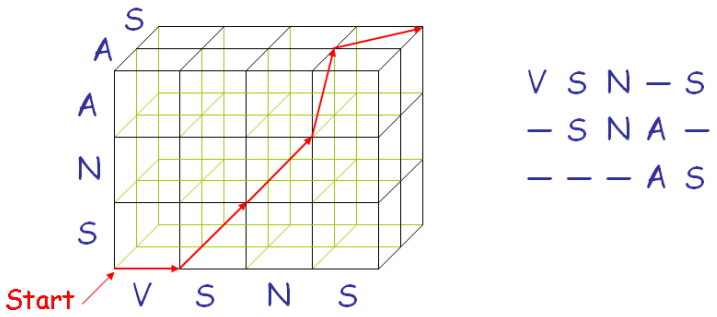


**Ilustración 18: Alineamiento de 3 Secuencias**

Cuando el número de secuencias es 2, en la matriz se deben revisar 3 entradas para decidir el óptimo valor de una celda. Cuando el número de secuencias es 3, se deben revisar 7 secuencias, y en general, para k secuencias, el número a revisar es $2^{n-1}$. El número de entradas en la matriz con *n* secuencias con longitud máxima es de $n^k$ lo que indica que el algoritmo basado en programación dinámica tiene una complejidad temporal de $O(2^{n-1} k^n)$. Como se aprecia el algoritmo tiene un crecimiento exponencial y se vuelve impráctico de utilizar incluso para un número bajo de secuencias.

Además, diferentes trabajos se han enfocado en demostrar que el problema de alineamiento múltiple utilizando diferentes supuestos es un problema intratable. En [Con03] se demostró que el problema es NP-Completo bajo la métrica de secuencia consenso y en [Sum01] se demostró que el problema es NP-Hard cuando se utiliza la métrica de suma de pares.

Debido a la dificultad del problema de alineamiento múltiple, los programas actuales utilizan diferentes estrategias heurísticas con el fin de obtener alineamientos de buena calidad en un tiempo razonable.

# 4 Estado del Arte

Hablar del estado del arte sobre alineamientos múltiples y sus implementaciones sobre ambientes distribuidos nos lleva a hablar de diferentes categorías de trabajos. Por un lado encontramos diferentes programas para generar alineamientos múltiples y encontramos a su vez diferentes implementaciones paralelas de los programas. Adicionalmente encontramos diferentes herramientas que nos permiten analizar y profundizar en el estudio de un alineamiento ya generado. Otro grupo de trabajos, más directamente relacionados con nuestro trabajo consiste en el desarrollo de workflows e integración de diferentes herramientas computacionales para analizar diferente tipo de información bioinformática.

## 4.1 Programas para generar Alineamientos

### 4.1.1 Clustal

Clustal [Clu01] fue uno de los primeros programas que se lanzó para realizar alineamientos múltiples. Surge como la necesidad de alinear un número grande de secuencias, ya que las estrategias de programación dinámica utilizadas para alineamientos pareados son altamente costosas en términos computacionales y no son escalables a más secuencias. La primera versión que se lanzó de este programa fue en 1988 [Clu02] e inmediatamente fue asimilado por la comunidad de biólogos.

ClustalW y Clustal X son dos paquetes que se encuentran actualmente. Clustal W es una versión actualizada de la primera versión que se lanzó, e incorpora un sistema de pesos para cada secuencia (Weights), entre otras mejoras. Clustal X ofrece la misma funcionalidad que Clustal W, pero añade una interfaz gráfica que permite visualizar en colores las diferentes columnas del alineamiento.

Clustal W utiliza un método de alineamiento progresivo y consiste en tres tiene tres grandes etapas:

- Matriz de distancia, alineamientos pareados: En este paso, por cada pareja de secuencias se calcula su distancia utilizando un algoritmo de alineamiento de dos secuencias. Actualmente, se ofrecen dos diferentes maneras para realizar este alineamiento pareados, la versión rápida aproximada, o la versión que utiliza programación dinámica.
- Árbol guía: A partir de la matriz de distancias, se calcula un árbol utilizando ya sea el algoritmo de Neighbor Joining o el de UPGMA (Unweighted Pair Group Method with Arithmetic Mean). Inicialmente se crea un árbol sin raíz y luego se ubica la raíz en el punto intermedio.
- Alineamiento progresivo: En este paso se alinean parejas de grupos. El proceso empieza por las ramas más cercanas y va avanzando hacia la raíz del árbol. Este proceso es el más costoso en términos de computación, y para realizar el alineamiento se utiliza el algoritmo de Myers y Miller).

La versión actual de Clustal W es la versión 2.1 que contiene pequeñas mejoras y arreglo de errores con respecto a la versión 2.0 [Clu03].

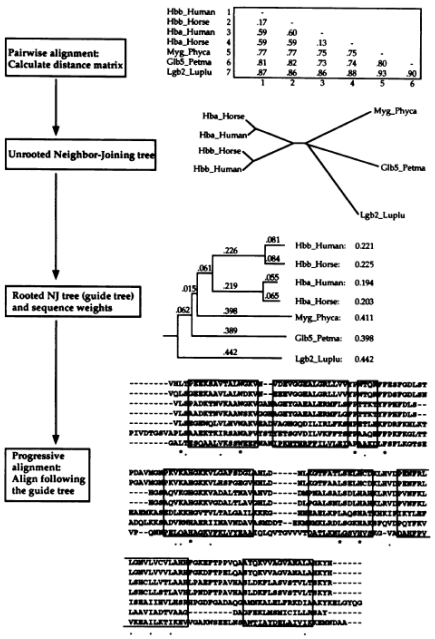


**Ilustración 19: Las tres etapas de ClustalW**

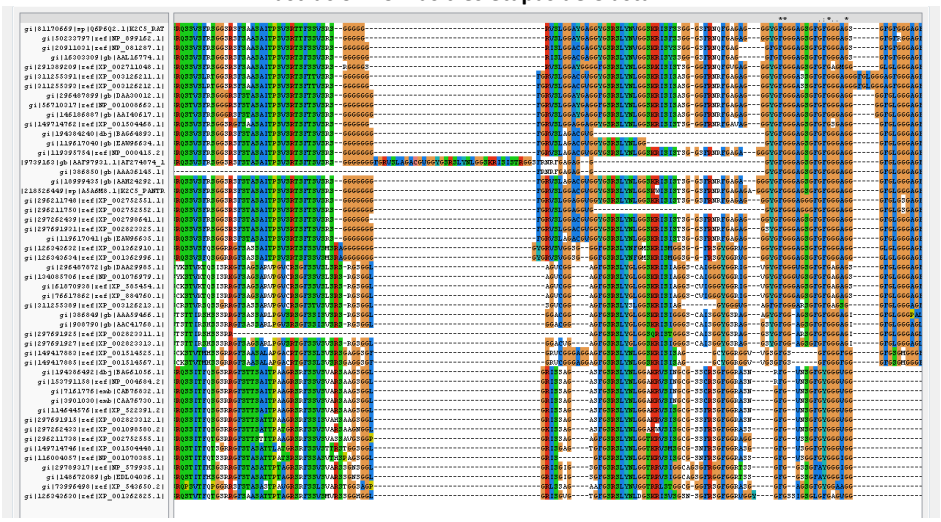

**Ilustración 20: Fragmento de un alineamiento múltiple generado por ClustalW y visualizado en la interfaz gráfica ClustalX**

Debido al rápido crecimiento en la cantidad de secuencias reportadas en las diferentes bases de datos, los biólogos cuentan cada vez con un mayor número de secuencias de diferentes organismos con los cuales realizar sus diferentes experimentos. Debido a la necesidad de incorporar cada vez más un número mayor de secuencias, y debido a las limitaciones técnicas que ofrece la ejecución secuencial de los diferentes programas de alineamiento múltiple, se han desarrollado diferentes versiones paralelas de éstos.

Diferentes versiones del programa Clustal han sido desarrolladas utilizando diferentes estrategias de paralelización.

#### 4.1.1.1 *ClustalW-MPI[Clu04]*

En esta implementación de Clustal, se utilizó la infraestructura MPI (Message Passing Interface) para la paralelización de las diferentes etapas que sigue este programa.
En la etapa de construcción de la matriz de distancias, se crearon grupos de alineamientos pareados de tamaño similar. Cada uno de estos grupos de alineamientos pareados fue asignado a uno de los procesadores con los que se contaba y cada grupo se procesó en paralelo.
Para el siguiente paso que consiste en la creación del árbol guía se utilizó una versión modificada del algoritmo de *neighbor-joining* que emplea $O(N^2)$ en tiempo. En este paso se paralelizó la búsqueda de las secuencias más divergentes.
En la última etapa del programa, en el alineamiento progresivo se mezcló paralelización con alto y bajo nivel de granularidad. En esta etapa se alinean las ramas del árbol con mayor nivel de similitud, y por lo tanto las diferentes parejas de ramas se pueden procesar de manera paralela, este paso constituye la paralelización gruesa. En el proceso de alinear cada una de las ramas, se utiliza una versión del algoritmo de Myers y Miller adaptada a alinear parejas de perfiles, o grupos de secuencias previamente alineadas. Este algoritmo fue implementado en su versión paralela también, lo que constituye la paralelización fina.
En este trabajo, se realizó la prueba utilizando un *cluster* de ocho computadores de doble procesador (Pentium III de 800 Mhz) interconectados con Fast Ethernet.
Los resultados de este trabajo muestran que en el proceso de paralelización de la primera etapa del programa (Creación de la matriz de distancia) se logró un speedup casi lineal (De 15.8x utilizando 16 procesadores) que es lo que se espera debido a que este proceso es completamente independiente entre los diferentes procesadores. La etapa del alineamiento progresivo, presentó un speedup de 4.3x. Se observa además que el mayor tiempo que emplea el algoritmo es en la construcción de la matriz de distancia, por lo que el tiempo total del alineamiento se ve dominado por esta etapa. Como conclusión de este trabajo, se observa que se paralelizó la etapa más costosa en tiempo computacional del programa ClustalW utilizando una infraestructura sencilla de *cluster*.

#### 4.1.1.2 *Clustal sobre GPUs*

En el trabajo de MSA-CUDA [MSA01] se implementó una versión de Clustal W utilizando la tecnología CUDA. Las pruebas fueron realizadas sobre una tarjeta gráfica GeForce GTX 280. En este trabajo se paralelizaron las tres etapas del ClustalW.
Si bien la estrategia de paralelización seguida en este trabajo es similar al de ClustalW-MPI, las restricciones que trae consigo CUDA hacen que la aproximación sea diferente. El cálculo de la matriz de distancia se realiza utilizando el algoritmo de Myers-Miller, sin embargo como este algoritmo es recursivo, se desarrolló una versión iterativa basada en la estructura de datos de pila.
Los siguientes pasos fueron paralelizados siguiendo un esquema similar al de ClustalW-MPI adaptándolo al lenguaje de CUDA. Las pruebas realizadas compararon MSA-CUDA con un *cluster* utilizando ClustalW-MPI que consistía en 16 nodos utilizando procesadores dual-core de 3.0Ghz. Los resultados evidencian que MSA-CUDA logró mejores *speedups* que la versión basada en MPI. Se observa además que ambos programas varían sus mejoras a medida que se cambian tanto las longitudes de las secuencias utilizadas como el número de éstas. MSA-CUDA provee una solución de paralelización al programa Clustal con una relación costo/desempeño mucho mejor que cualquier solución basada en *clusters.*

Un trabajo similar a MSA-CUDA, GPU-ClustalW[Gpu01], atacó el mismo problema utilizando también la infraestructura de GPUs pero en este trabajo solamente se paralelizó la primera etapa del programa, es decir la creación de la matriz de distancias. En este trabajo, sin embargo, se utilizó el algoritmo de Smith-Waterman y éste fue paralelizado tomando en cuenta que cada elemento de la antidiagonal se puede calcular de manera independiente de los otros. Otra diferencia con MSA-CUDA es que en este trabajo, se utilizó el lenguaje de programación de alto nivel para GPUs *GLSL* (OpenGL Shading Language) [Gls01].

#### 4.1.1.3 *Otros trabajos*

Otros trabajos relacionados con la paralelización de ClustalW incluyen implementaciones en diferentes tipos de tecnologías como SGI [Sgi01], y FPGAS[Fpg01] donde se obtienen speedups de 10x y 12x respectivamente.

### 4.1.2 T-COFFEE [Tco01]

T-Coffee (Tree based Consistency Objective Function for alignment Evaluation) es otro paquete de software que realiza alineamientos múltiples. Al igual que ClustalW utiliza un enfoque de alineamiento progresivo, la cual se basa en un árbol filogenético aproximado y se va construyendo la solución incrementalmente siguiendo el orden que indica el árbol. El proceso de alineamiento progresivo es un proceso avaro, por lo que tomar errores generados en etapas iniciales se propagan hasta el final. Para solucionar este problema T-Coffee utiliza información de diferentes fuentes, que pueden ser alineamientos pareados globales y alineamientos locales. Al permitir combinar información sobre diferentes alineamientos, T-Coffee provee un nivel de flexibilidad que no brindan otras herramientas con propósito similar.
Las etapas principales que sigue T-Coffee para realizar el alineamiento múltiple son:

- Creación de la Biblioteca Primaria: En este paso se realiza un alineamiento pareado entre las secuencias. Como se mencionó, la creación de la bilbioteca es muy flexible y se pueden utilizar diferentes algoritmos para este paso. En la ejecución por defecto del programa se realiza un alineamiento pareado global utilizando el programa ClustalW y se utilizan los 10 mejores resultados de alineamientos locales generados por el programa Lalign del paquete de software de FASTA. La información que se almacena corresponde a una lista de residuos alineados (El residuo x de la secuencia A, se alinea con el residuo y de la secuencia B)
- Cálculo de pesos de la Biblioteca Primaria: Una vez calculada la Biblioteca, se procede a asignar un peso a cada pareja de residuos. Este proceso es importante porque no todos los alineamientos pareados tienen la misma calidad. Para este paso, se le asigna a cada pareja de residuos un peso equivalente al porcentaje de identidad del alineamiento local o global del que proviene. Este proceso permite combinar la información de los alineamientos locales y globales, pues si se encuentra en la biblioteca que dos entradas alinean los mismos residuos, estas entradas se fusionan en una sola donde el peso es igual a la suma de los pesos individuales.
- Extensión de la Biblioteca: Este proceso se realiza para generar mayor información a la biblioteca. En esta etapa, se evalúan los residuos alineados y se verifica si existen relaciones de transitividad. Si el residuo x de la cadena A está alineado con el residuo y de la cadena B, y si adicionalmente encontramos que tanto el residuo x de A y el residuo y de B están alineados con el residuo z de C con costos w1 y w2 respectivamente, esta información nos dice que los residuos x,  y están alineados también a través de z. Al peso del alineamiento del residuo x con el residuo y se añade min{w1,w2}. Este proceso es uno de los más costosos en procesamiento, pues se revisan triplas de residuos alineados y toma $O(N^3L^2)$ en tiempo donde *N* es el número de secuencias y *L* es la longitud promedio de las secuencias.
- Alineamiento progresivo: Este proceso es similar al utilizado en otros programas de alineamiento progresivo como ClustalW. La mayor diferencia con estos métodos es que T-Coffe emplea la biblioteca que va indicando que residuos se alinean con cuales tomando inicialmente los de mayor peso, en vez de utilizar el árbol filogenético basado en la matriz de distancia que utilizan los otros programas.

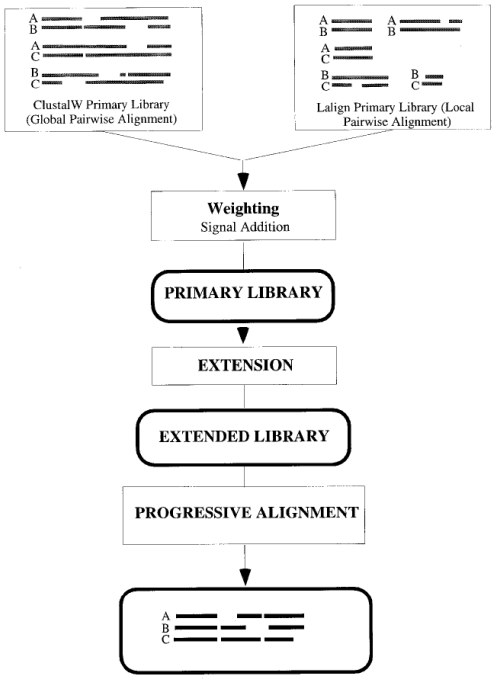


**Ilustración 21: Etapas de T-Coffee**

El análisis de complejidad en tiempo del programa muestra que T-COFFEE toma $O(N^2L^2) + O(N^3L)+O(N^3)+O(NL^2)$, donde *N* es el número se secuencias y *L* es la longitud promedio de las secuencias. El término $O(N^2L^2)$ proviene del cálculo de todos los alineamientos pareados. El término $O(N^3L)$ proviene de la extensión de la biblioteca. El sumando $O(N^3)$ se genera al momento de realizar el árbol utilizando Neighbor-Joining. Por último, el término $O(NL^2)$ proviene del alineamiento múltiple progresivo. Empíricamente se observó que las etapas de creación de la biblioteca primaria y el del alineamiento progresivo son mucho mayores que el tiempo empleado en la extensión de la biblioteca, es decir que los factores dominantes del costo computacional del algoritmo son los de $O(N^2L^2) + O(NL^2)$.

El resultado que genera T-COFFE son generalmente más exactos que los generados por otros programas. Sin embargo, esta ganancia en cuanto a exactitud del alineamiento viene a expensas de un gran tiempo de procesamiento. T-COFFEE en la práctica está limitado un número relativamente pequeño de secuencias (Del orden de 100). Para un número mayor de secuencias, el programa toma un tiempo sumamente grande volviéndolo una opción inviable en estos casos.

Al igual que otros programas de alineamiento múltiple T-Coffee ha sido paralelizado utilizando diferentes aproximaciones.

#### 4.1.2.1 *Parallel-TCOFEE*[Par01]

En este trabajo se presenta una implementación paralela del programa T-Coffee utilizando MPI. Se paralelizó cada una de las etapas del programa incluyendo la opción de 3D-Coffee que utiliza plantillas descargadas de internet para brindar información estructural al alineamiento.

- Generación de la biblioteca: La creación de los alineamientos pareados iniciales se paraleliza de manera similar como se realiza en ClustalW-MPI, es decir se agrupan tareas de alineamiento y a cada procesador se le asigna un conjunto de tareas. Para el proceso de agrupamiento de alineamientos de residuos, el alineamiento del residuo x de la cadena i con el residuo y de la cadena j se asigna al procesador `(i+j) mod p` donde p es el número de procesador disponibles.

- Alineamiento progresivo: Esta etapa constituye la de mayor complejidad para paralelizar. En este trabajo se modeló el orden de los alineamientos que se deben realizar como un grafo acíclico dirigido (DAG) que indica un orden topológico para alinear las cadenas. Las tareas que no son dependientes entre sí son procesadas de manera paralela.

Las pruebas realizadas sobre esta versión se realizaron utilizando diferentes números de procesadores (Variando desde 16 hasta 80) Intel Xeon de 3Ghz. Los resultados presentan una evidente mejora en los tiempos de ejecución del programa llegando a alinear 1048 secuencias de longitud máxima de 523 residuos en un tiempo de 8 horas con 16 nodos y rebajando este tiempo a casi dos horas y media usando 80 nodos.
Otro resultado importante de este trabajo es que el *speedup* en la etapa de generación de la biblioteca es casi lineal, llegando a un máximo de 5 usando los 80 nodos. Sin embargo, la etapa de alineamiento progresivo presenta un *speedup* cercano a 1 que no parece incrementar a medida que se aumentan los nodos. Esto sugiere que el esquema utilizado en esta etapa no fue muy eficiente. Sin embargo, el tiempo total de la ejecución del programa si se disminuyó de manera considerable debido a que la etapa de creación y extensión de la biblioteca domina a la etapa del alineamiento múltiple.
Actualmente Parallel T-Coffee se encuentra en la versión 1.93 publicada en el año 2009. El programa se puede descargar de [Par02]

#### 4.1.2.2 *Cloud-Coffee*[Clo01]

En esta versión paralela de T-Coffee se paralelizan todas las etapas del programa.

- Búsqueda de Templados: Esta etapa es utilizada cuando se desea hacer un alineamiento estructural usando 3D-Coffee. Esta búsqueda de plantillas es un proceso arduo pero se paraleliza trivialmente sobre cada una de las secuencias que se tienen.
- Cálculo de la biblioteca y extensión: La paralelización de estas etapas se logra agrupando las tareas de alineamiento pareado y de cálculo de la extensión dependiendo del número de procesadores disponibles.
- Alineamiento progresivo: En este paso, los diferentes nodos del árbol que son independientes se alinean utilizando el algoritmo de programación dinámica de NeedleMan y Wunch. El cálculo del árbol guía no se paralelizó debido a que este proceso no constituye un factor importante en el tiempo total del algoritmo.

La implementación de esta versión se realizó sobre la infraestructura de Amazon Elastic Compute Cloud[Ama01].
Se realizó esta prueba utilizando diferentes combinaciones de infraestructuras (Variando desde 1 CPU hasta 8). Se evidencia en los resultados que el programa si aprovecha la infraestructura pues los tiempos de ejecución si disminuyen a medida que se añaden más CPUs. En la última parte de la discusión se propone esta alternativa como un modelo aceptable y competitivo para la instalación y mantenimiento de servidores privados mantenidos localmente por razones de costo.

### 4.1.3 MUSCLE[Mus01]

MUSCLE es otro programa que genera alineamientos múltiples como T-Coffee y Clustal. Sin embargo, a diferencia de otros programas de alineamientos MUSCLE utiliza una estrategia de refinamiento iterativo. En la primera etapa se busca generar un primer alineamiento, que no es de alta calidad, priorizando rapidez. En iteraciones sucesivas el algoritmo va generando alineamientos mejores y si éstos son mejores, va actualizando.
Los principales pasos del programa son:

- **Primer alineamiento:** Utilizando una métrica de k-meros (k-tuplas) se genera una matriz de distancias entre las diferentes secuencias. Se utiliza esta métrica por su velocidad, en vez de utilizar alguna técnica más elaborada como Smith-Waterman. Utilizando UPGMA (Unweighted Pair Group Method with Arithmetic Mean) se genera un primer árbol binario. Con este árbol se genera un alineamiento de perfiles con las hojas que se hay en el árbol, como se realiza en otros métodos.

- **Alineamiento mejorado:** Utilzando el alineamiento múltiple del primer paso, se calcula una segunda matriz de distancias usando la métrica de Kimura [Kim01] , lo que a su vez genera un segundo árbol TREE2 el cual sirve para guiar otro alineamiento múltiple MSA2. Este alineamiento posee mejor precisión que el primero que se generó.
- **Refinamiento:** En este paso se escoge alguno de los vértices del árbol 2 y se elimina. Esto da a lugar a dos subárboles. A cada uno de los subárboles se le calcula su perfil y se realiza un alineamiento de ambos. Si este alineamiento es mejor que MSA2, éste lo reemplaza y se toma como base para las siguientes iteraciones. El proceso sigue hasta que en alguna iteración no se mejore el alineamiento.

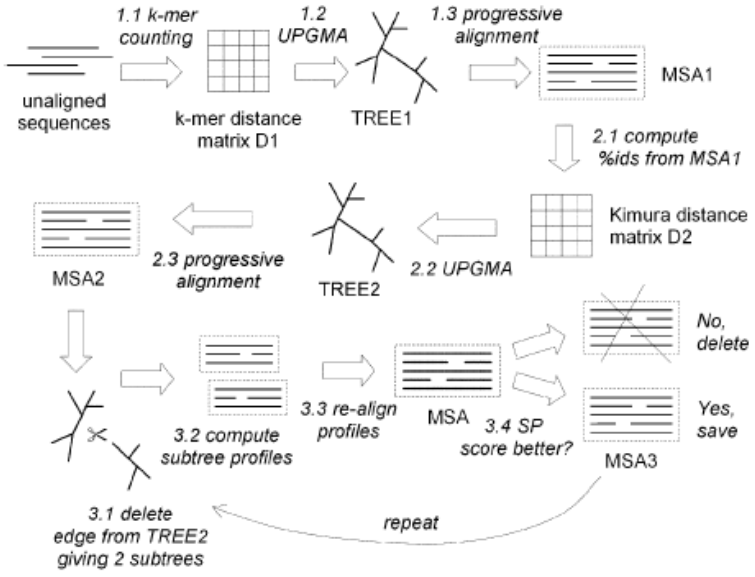


**Ilustración 22: Diagrama de las etapas de MUSCLE**

El análisis de complejidad de MUSCLE nos muestra que el algoritmo toma $O(N^2L) + O(NL^2) + O(N^3L)$. Los dos primeros términos provienen de las dos primeras etapas del programa, mientras que el término $O(N^3L)$ es generado por el alineamiento refinado.

#### 4.1.3.1 *Parallel MUSCLE [Mus02]*

En este trabajo se presenta una versión paralela del programa MUSCLE. Esta implementación utiliza el modelo de paso de parámetros MPI, más específicamente utilizando el compilador Intel 8.0 C++ OpenMP[Ope01]. En este trabajo se paralelizan las etapas de creación del primer y segundo alineamiento.

En el primer alineamiento, se genera un árbol basado en la matriz de distancia utilizando los k-meros. Este árbol sirve como guía para la generación del primer alineamiento. En este paso, se paralelizó el alineamiento de las diferentes hojas siguiendo el modelo de *workqueuing* de Intel [Int01] donde cada tarea de alineamiento de un par de hojas se define como una tarea para el compilador.

La segunda tarea de paralelización se llevó a cabo en el cálculo de las matrices de distancia en el segundo alineamiento. En este caso, se identificó fácilmente que las comparaciones de cada par de secuencias eran independientes. Se consideró la utilización de un `for` paralelo para el ciclo externo, pero esto genera que las tareas no tengan el mismo tamaño pues un segundo ciclo anidado depende el tamaño del primer ciclo. Por esta razón se volvió a utilizar el modelo de *workqueing.*

Las pruebas de este trabajo se realizaron sobre un sistema basado en 16 procesadores Intel-Xeon de 3Ghz. Se evidencia que el *speedup* obtenido en la ejecución de la aplicación es casi lineal llegando hasta un 15x con 16 nodos. Las pruebas se realizaron con conjuntos de datos, mostrando que a mayor número de secuencias, se obtienen mejores *speedups.*

## 4.2 Herramientas para el análisis de alineamientos múltiples

A partir de un alineamiento múltiple diferentes tipos de tareas surgen con el ánimo de profundizar en su estudio. También a partir de las tareas que surgen, se han desarrollado diferentes herramientas computacionales disponibles para los investigadores a través de internet.

### 4.2.1 Representación Gráfica

Una manera novedosa que ha sido muy adoptada en los grupos de investigación biológica para la representación de alineamientos múltiples es el generación del logo de un alineamiento [Web01, Web02].

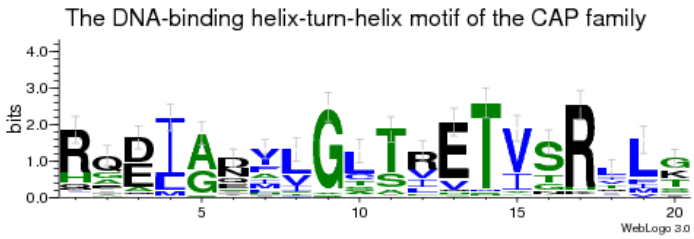


**Ilustración 23: Logo del alineamiento múltiple de varias secuencias de la familia CAP**

Esta representación gráfica de un alineamiento es un método muy útil puesto que permite visualizar que posiciones del alineamiento son más conservadas y poseen mayor información, también permite detectar regiones de baja información donde el programa no pudo alinear satisfactoriamente los residuos. El logo utiliza la información del perfil de un alineamiento, dibujando por cada una de las columnas los aminoácidos con más frecuencia con letras más grandes.

### 4.2.2 Comparación de Alineamientos

Si bien un alineamiento múltiple da gran información sobre las secuencias que se poseen, en ocasiones tener diferentes alineamientos permite tener una visión más global de las secuencias. En particular regiones no conservadas tienden a ser alineadas de maneras diferentes por los diversos algoritmos. En ocasiones es tan alto el nivel de desacuerdo entre diferentes alineamientos múltiples, que se decide descartar estas regiones para futuros análisis porque que no proveen información útil y pueden entorpecer los resultados de diferentes procesos posteriores.

Para comparar los resultados de diferentes alineamiento existen varias herramientas. Uno de éstos programas es AltaVist[Alt01, Alt02] que se accede vía web. Este programa permite identificar que regiones de las secuencias fueron alineadas igual entre dos programas y que regiones fueron ambiguas, es decir que fueron alineadas de manera distinta por los dos programas. Ofrece la funcionalidad de poder introducir un alineamiento múltiple generado con cualquier programa, siempre y cuando se encuentre en formato FASTA.

**Ilustración 24: Fragmento de los resultados generados por Altavist. El color azul indica fragmento alineados de igual manera por los dos programas, las regiones rojas representan las regiones más disímiles**

### 4.2.3 Bases de Datos

Para poder evaluar la calidad de un alineamiento múltiple se podrían utilizar las métricas presentadas o alguna otra. Sin embargo este criterio no sería muy útil puesto que diferentes programas se enfocan en una métrica en particular entonces su comparación no sería objetiva. Adicionalmente, las métricas no necesariamente incorporan toda la información biológica que se desearía. Una alternativa a este problema son las bases de alineamientos. Estas bases se componen de diferentes alineamientos que han sido revisados por expertos biólogos, con lo cual se añade el conocimiento científico a la elaboración de los alineamientos. Cuando se presenta un nuevo algoritmo o programa para alinear, el método de evaluación estándar consiste en comparar los resultados contra los de algún alineamiento de estas bases de datos y medir la proporción de aminoácidos alineados correctamente. Diferentes bases de datos existes, entre ellas encontramos BaliBASE en sus tres versiones [Bal01, Bal02, Bal03], HOMSTRAD [Hom01] que incorpora información sobre la estructura de las secuencias a alinear, PREFAB [Clu01, Pre01] cuyo objetivo es reducir la dependencia de los expertos para la incorporación de alineamientos de referencia, SABMark [Sab01] enfocado a secuencias muy disímiles, entre otros programas. En la práctica, la evaluación de la mayoría de algoritmos se hace utilizando BaliBASE.

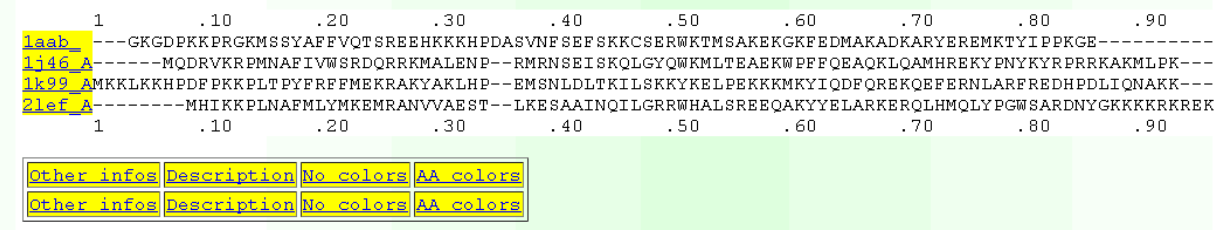


**Ilustración 25: Ejemplo de un alineamiento de referencia de BaliBASE**

### 4.2.4 Visualización de Alineamientos

Si bien un alineamiento se puede presentar como un archivo plano de texto, esta representación por lo general no es útil para entender propiedades del alineamiento ni para identificar patrones, bloques o regiones largas de inserciones. Debido a esto, diferentes herramientas han sido desarrolladas con el fin de poder visualizar alineamientos. Algunas de estas herramientas son ClustalX[Clu01], JalView[Jal01], UGene[Uge01], Geneious[Gen02] entre muchos otros.Estas herramientas pueden venir como parte de una suite de herramientas como UGene o Geneious, o como un applet de java para ser embebido en una aplicación web como JalView. Estos programas de visualización incorporan un esquema de colores para cada aminoácido con lo cual se pueden identificar regiones del alineamiento con ciertas características. También por cada posición del alineamiento, presentan el grado de conservación de la columna, que es muy útil para identificar dominios proteicos entre otros.


**Ilustración 26: Visualización de un alineamiento utilizando JalView**

## 4.3 Integración de Herramientas Bioinformáticas

Otro grupo de trabajos relacionados consiste en el desarrollo de motores de workflows orientados a aplicaciones científicas. Estos motores de workflows ofrecen diferentes características como la posibilidad de incorporar aplicaciones nuevas, generar workflows de manera gráfica (Drag and Drop), posibilidad de ejecutar los workflows sobre infraestructuras grid, posibilidad de descargar workflows y programas desde repositorios centrales entre otros.

### 4.3.1 Taverna

Taverna [Tav01] es un sistema de manejo de workflows, open source, creado dentro del proyecto myGrid. Entre sus principales características están la posibilidad de crear workflows mediante un editor gráfico bastante fácil de utilizar, posibilidad de incorporar nuevas tareas o módulos y de importar tareas ya creadas por otros usuarios desde un repositorio central. Permite ser ejecutado en diferentes servidores como una aplicación web, y permite ser descargado, configurado y desplegado en una infraestructura grid local. Taverna utiliza el plugin JSDL para el envío de jobs usando GridSAM[Gri02] el cual es un ofrece una interface para el envío de jobs a diferentes administradores de recursos como SGE, Condor, PBS, entre otros.

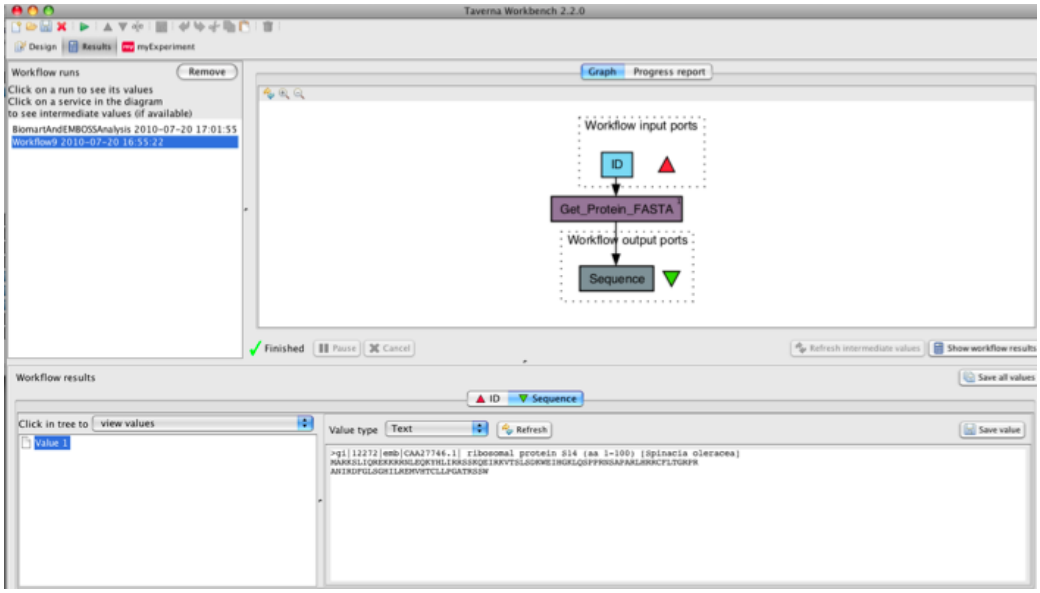


**Ilustración 27: Ejemplo de un workflow sencillo utilizando Taverna**

En WPSA[Wps01] (Workflow for Protein Sequence Analysis) se desarrolla un workflow para el análisis de secuencias de proteínas utilizando el software de manejo de workflows Taverna[Tav01]. Si bien Taverna cuenta con un repertorio grande de workflows biológicos prediseñados y disponibles para su utilización por parte de los usuarios, WPSA se destaca por ser uno de los workflows más completos para el análisis de secuencias de proteínas.

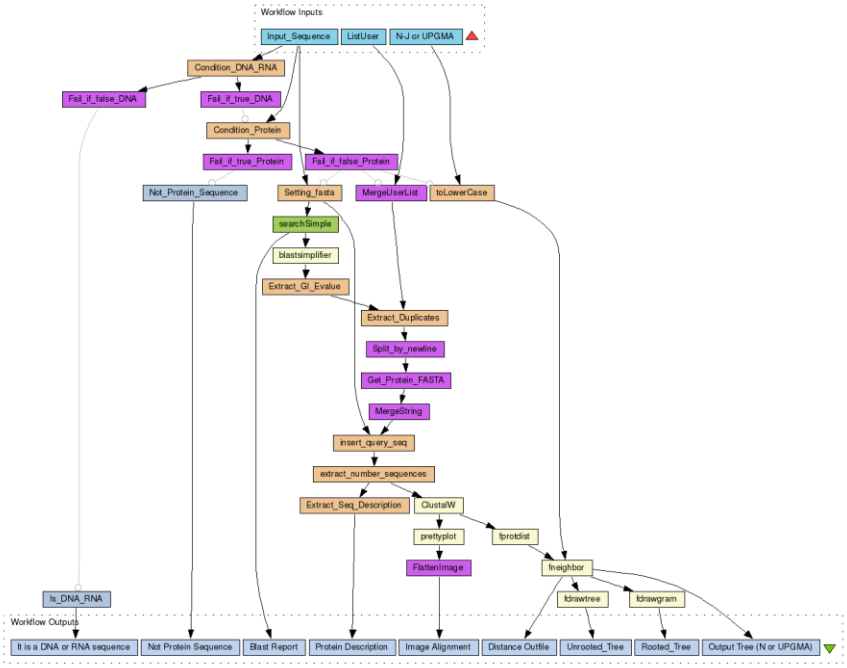


**Ilustración 28: Implementación de WPSA en Taverna**

Las principales tareas que realiza WPSA son la búsqueda de secuencias homólogos utilizando BLAST [Bla01], a partir de este resultado se realiza un proceso de limpieza de datos, posteriormente se realiza un alineamiento múltiple utilizando ClustalW[Clu01] y finalmente se genera un análisis filogenético utilizando PHYLIP[Phy01].

WPSA se puede acceder desde el repositorio de workflows myexperiments en [Wps02]

### 4.3.2 Armadillo

Armadillo[Arm01] es una plataforma para la construcción de workflows bioinformáticos. Cuenta con un número de herramientas preinstaladas y configuradas para la realización de diferentes tipos de tareas: Búsquedas en bases de datos, generación de alineamientos múltiples, generación y visualización de árboles filogenéticos, inferencia de modelos evolutivos entre otros. Para poder ejecutar un workflow, el usuario debe descargar la aplicación y a partir de las herramientas incorporadas, armar su workflow de acuerdo a sus necesidades. Al igual que la mayoría de sistemas de administración de workflows, Armadillo permite el desarrollo de éstos de manera gráfica. Armadillo no tiene posibilidades de integración con una infraestructura grid, por lo que se limita al uso personal en computadores de escritorio.

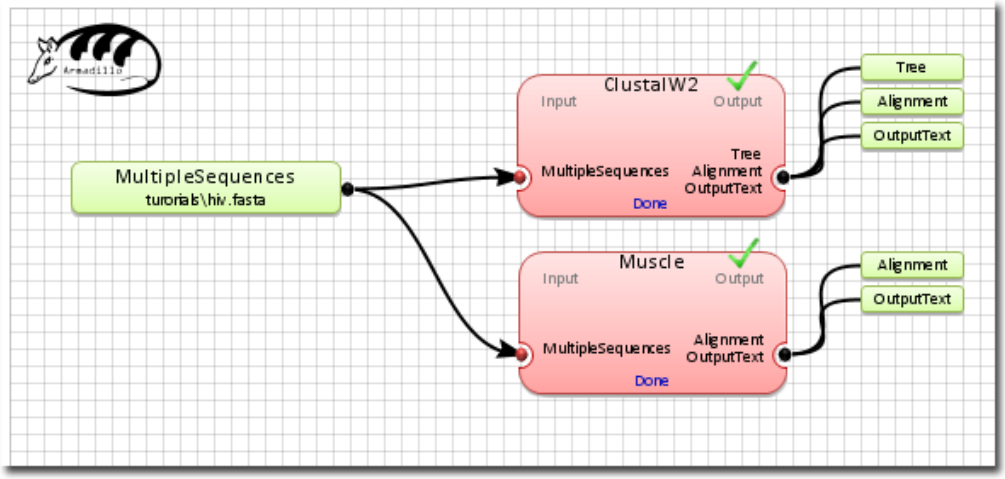


**Ilustración 29: Ejemplo de un workflow de alineamiento múltiple con Armadillo**

### 4.3.3 Galaxy

Galaxy [Gal01] es una plataforma web para el acceso a diferentes recursos biocomputacionales. Galaxy incorpora una gran variedad de diferentes herramientas para realizar análisis sobre información biológica. Adicionalmente permite el desarrollo de workflows y cuenta con un repositorio central donde los usuarios pueden compartir sus diferentes trabajos. Galaxy permite acceder a todos los recursos a través de su interface web donde los usuarios tienen acceso a todas las herramientas. Adicionalmente Galaxy permite ser descargado y desplegado en un ambiente local, y permite ser integrado a través de DRMAA [Drm01] con diferentes plataformas grid como Sun Grid Engine [Sge01], PBS [Pbs01], Plattform LSF[Pla01] y actualmente existen proyectos para integrarlo con otras plataformas.

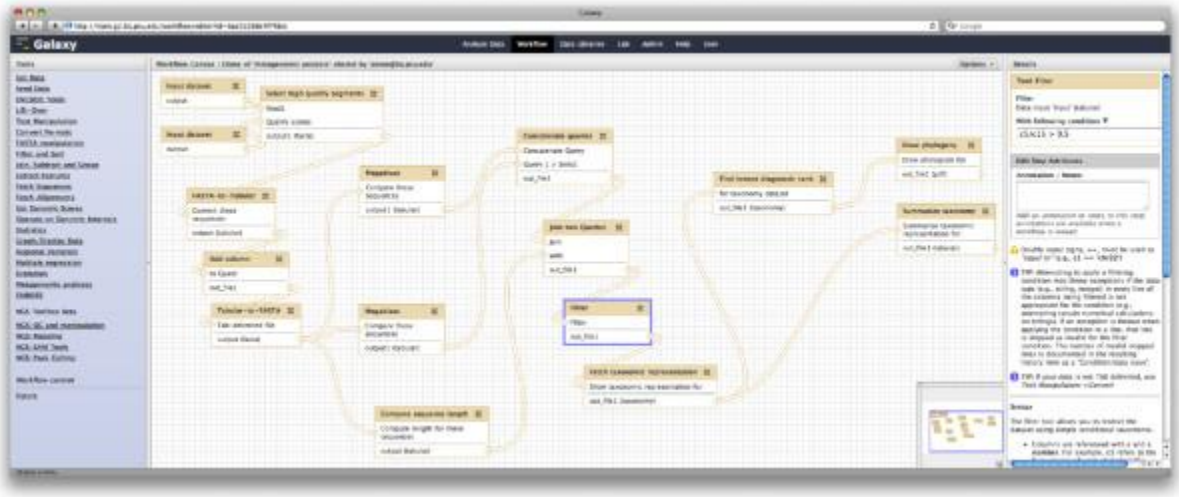


**Ilustración 30: Ejemplo de definición de un workflow desde la interfaz web de Galaxy**

### 4.3.4 Kepler

Kepler [Kep01] es un motor de workflows basado en el modelo de Ptolemy II[Pto01]. A diferencia de otros motores de workflow, Kepler no está orientado a un área de conocimiento específica. Kepler, trae incorporados diferentes módulos o actores que realizan diferentes tipos de tareas. Entre los actores principales se encuentran módulos de integración con R[R01] y con MatLab[Mat01], módulos para el acceso a archivos de Excel, y actores para la invocación remota de WebServices basados en WSDL. Kepler permite la integración con infraestructuras grid mediante el proyecto Serpens [Ser01]. Serpens provee una serie de actores y workflows dedicados que proveen soporte a diferentes arquitecturas grid como gLite[Gli01] y UNICORE[UNI01].

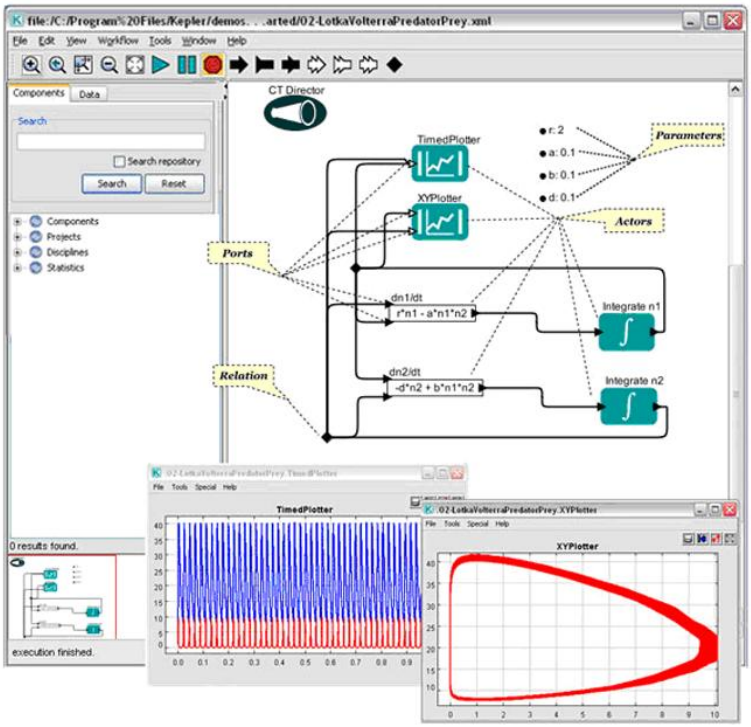


**Ilustración 31: Ejemplo de un workflow científico en Kepler**

### 4.3.5 Loni Pipeline.

Loni Pipeline [Lon01] es un motor de workflows creado en UCLA enfocado al procesamiento y visualización de imágenes médicas.Trabaja bajo el esquema de Cliente/Servidor, donde el servidor de Loni se instala en una máquina dedicada y desde un computador de escritorio se accede utilizando la aplicación cliente. Si bien Loni Pipeline fue desarrollado con un campo en particular, su modelo es tan genérico que cualquier tipo de workflow se puede incorporar. Permite definir módulos a partir de ejecutables ya instalados en el servidor lo que lo vuelve muy flexible. Permite la integración con diferentes motores de grid como SGE, Condor a través del plugin DRMAA y JGDI [Jgd01].

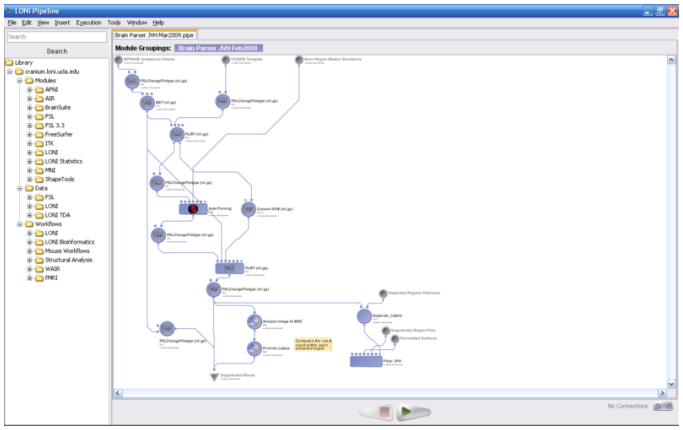

**Ilustración 32: Ejemplo de un workflow en Loni Pipeline**

La tabla 1 presenta el resumen de las características más relevantes que ofrecen los sistemas de manejo de workflows científicos.

| Motor de Workflow | Área de Conocimiento | Diseño de Workflows | Posibilidad de incorporar nuevos módulos | Integración con Grid | Recompilación de las Herramientas |
|---|---|---|---|---|---|
| **Taverna** | Bioinformática | GUI: Drag & Drop | Sí | Sí: GridSAM | Sí |
| **Armadillo** | Bioinformática | GUI: Drag & Drop | No | No | N/A |
| **Galaxy** | Bioinformática | GUI: Drag & Drop | No | Sí: DRMAA | N/A |
| **Kepler** | Ciencia en general | GUI: Drag & Drop | Sí | Sí: Con gLite y UNICORE | Sí |
| **Loni PipeLine** | Visualización médica | GUI: Drag & Drop | Sí | Si: DRMAA y JGDI | No |

**Tabla 1: Principales Características de Sistemas de Manejo de Workflows**

A partir de esta exploración se puede concluir que existen muchas herramientas para lograr generar y analizar alineamientos múltiples de una manera más completa a la que se hace en la práctica. Adicionalmente se observa que diversos motores de workflow que permiten la integración de herramientas computacionales y que permiten ser integrados con infraestructuras grid para lograr un alto nivel de desempeño. Con este panorama se identifica la posibilidad de generar un workflow dedicado a la generación y análisis de alineamientos múltiples basados en la infraestructura oportunista UnaCloud.

## 4.4 Infraestructuras Elásticas

Otro de los temas a considerar en el estado del arte es el de infraestructuras elásticas, es decir infraestructuras cuyos recursos asignados crecen cuando el nivel de carga, definido de cierta manera, se ve incrementado por encima de algunos criterios. Una de las principales ventajas de este tipo de las infraestructuras elásticas es el de permitir un mejor aprovechamiento de los recursos, ya que solamente se asignan cuando hay un alto nivel de uso. UnaCloud cuenta con la problemática de tener poca cantidad de recursos disponibles, ya que actualmente solo se encuentra desplegado en 3 salas de computadores, por lo que en ocasiones se presentan conflictos entre los diferentes investigadores por el acceso a los recursos.

Existen diferentes implementaciones de infraestructuras elásticas, y principalmente son ofrecidas en el esquema de cloud computing. A pesar de que diferentes proveedores ofrecen un modelo de infraestructura elástico, la mayoría se basan en el esquema de Amazon Elastic Cloud.

### 4.4.1 Amazon Elastic Compute Cloud

Amazon Elastic Compute (EC2) Cloud consiste en la implementación del modelo IAAS desarrollada por Amazon. Como su nombre lo indica, EC2 es un cloud elástico, lo que permite a la infraestructura crecer de manera dinámica bajo demanda. Con el fin de proveer esta funcionalidad, EC2 cuenta con los siguientes componentes:

#### 4.4.1.1 *Amazon CloudWatch*

Este primer componente, Amazon CloudWatch [Clo02], se encarga del monitoreo de monitorear los recursos de cada usuario tomar diferentes tipos de acciones cuando se presenten ciertos eventos. El monitoreo de los recursos se puede hacer por vía manual, es decir, desde la interface web, el usuario puede observar en tiempo real cada una de las métricas disponibles, o a través de un API que permite programar el monitoreo. Las métricas monitoreadas en CloudWatch son métricas genéricas como es el uso de CPU, cantidad d información leída y escrita en el disco duro, nivel de uso de red entre otros, por lo cual son independientes del tipo de aplicación que se esté corriendo en la infraestructura.

#### 4.4.1.2 *AutoScaling*

Este segundo componente es el encargado del crecimiento de la infraestructura asignada a un usuario. A partir de la información registrada por CloudWatch y a partir de ciertas reglas definidas, AutoScaling se encarga de aumentar o disminuir el número de instancias asignadas.

#### 4.4.1.3 *Elastic Load Balancing*

Este último componente se encara de asignar trabajo de manera efectiva a las instancias asignadas al usuario. Si el componente de AutoScaling incrementa el número de instancias en cierto momento, el componente de balanceo de carga debe reconocer este incremento y asociar la carga de tráfico a estas nuevas instancias. Así mismo si CloudWatch detecta que existen nodos no saludables, es decir con muy alto nivel de carga, el componente de balanceo de carga se encarga de redirigir solicitudes hacia otros nodos.

# 5 Objetivos

## 5.1 Motivación

Los principales motivadores para el desarrollo de este trabajo se pueden resumir así:

- La dificultad que tiene la generación de un buen alineamiento múltiple debido a sus características algorítmicas, que lo convierten en un problema intratable.
- El gran número de diferentes heurísticas que existen para generarlos y el gran número de herramientas diversas que existen para analizarlos, aunque le dan una gran variedad de posibilidades a los investigadores, no permiten tener una visión consolidada del problema.
- Los diferentes motores de workflow permiten la integración de diferentes herramientas, y ofrecen una excelente opción para el desarrollo de un workflow orientado a la generación y análisis de alineamientos múltiples de secuencias.
- La posibilidad de integrar los motores de workflow con infraestructuras grid, abren la posibilidad de utilizar nuestra infraestructura oportunista UnaCloud para brindar una solución que ofrezca un alto poder computacional a un muy bajo costo.
- La facilidad de uso de una aplicación para una persona por fuera del campo de la informática es un factor importante en su aceptación o no. Debido a esto, se decide que se debe brindar una interface gráfica amigable para que el usuario no tenga que interactuar con el motor de workflow directamente.
- A diferencia de otros proyectos desarrollados sobre la infraestructura UnaCloud donde el tamaño de la tarea a realizar se conocía inicialmente, la herramienta a desarrollar ofrecerá sus servicios como una aplicación web por lo que su carga será variable. Debido a esto se debe proponer e implementar un mecanismo que permita utilizar la infraestructura oportunista de una manera que vaya de acuerdo al nivel variable de carga.

## 5.2 Objetivo General

El objetivo principal de este trabajo consiste en ofrecer una solución para el alineamiento múltiple de secuencias basada en la integración de recursos dedicados y oportunistas con el fin de abordar el problema desde diferentes puntos de vista y generar una visión unificada de toda la información.

## 5.3 Objetivos específicos

Como objetivos específicos de este trabajo se tienen los siguientes:

- Definir qué grupo de tareas son importantes para el análisis de alineamientos múltiples de secuencias.
- Diseñar e implementar un workflow que realice las tareas de análisis y comparación de alineamientos múltiples que se definieron.
- Integrar el workflow sobre una infraestructura de grid oportunista con el fin de brindar un alto nivel de desempeño y aprovechar las capacidades computacionales subutilizadas de los diferentes laboratorios de computadores.
- Diseñar y ofrecer una interface web amigable para que la comunidad bioinformática pueda hacer uso del workflow diseñado sin requerir el aprendizaje de nuevas herramientas computacionales.
- Integrar la aplicación web y la ejecución del workflow, con el proyecto UnaCloud de manera que se aprovechen servicios ya desarrollados.

# 6 Propuesta de Solución: UnaCloud MSA

## 6.1 Análisis de Disponibilidad de la Infraestructura

El primer paso tomado consistió en un estudio sobre la infraestructura física con la que se contaba. Este estudio se hace con el ánimo de determinar si la infraestructura realmente cuenta con un nivel de uso lo suficientemente bajo para justificar el modelo de despliegue oportunista. Con este fin, se realizó un estudio del nivel de disponibilidad de la infraestructura física. Se tomó medida del nivel de uso de CPU libre de las máquinas físicas a través de un agente que utilizando el api de SIGAR[Sig01] que se comunicaba con una base de datos central SQLServer 2008[Sql01]. El objetivo del estudio era identificar por un lado, el nivel de CPU disponible en cada máquina y por otro lado identificar los momentos de no disponibilidad, es decir, reinicios y apagones de las máquinas. Se buscaba determinar el comportamiento promedio de las máquinas e identificar cuales tienen mayor probabilidad de reinicio durante un día de estudio común. Se tomó la información durante un mes en las salas del departamento (Waira 1, Waira 2 y Alan Turing). Cada minuto, cada una de las máquinas reportaba a una base de datos central, su nivel de CPU ocupado. Los momentos de no disponibilidad se identificaron cuando se presentaban dos lecturas de la misma máquina y del mismo día, con una diferencia superior a 3 minutos. Este valor de 3 minutos se escogió empíricamente luego de observar que en ocasiones había cierto nivel de retraso en la comunicación, y aunque el lapso de tiempo era de 2 o 3 minutos, la máquina física siempre estuvo disponible.

Luego de analizar la información obtenida, se evidencia que el nivel de uso promedio de las salas es inferior a un 10%. La figura 33 muestra la relación entre el nivel de uso promedio de CPU y la hora del día.

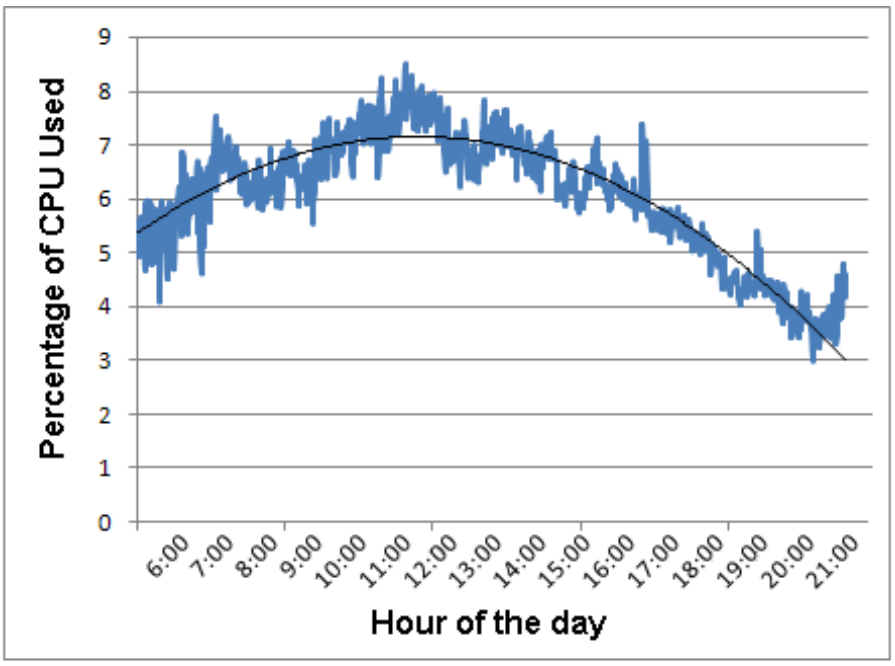


**Ilustración 33: Fluctuación de uso de CPU promedio a lo largo del día**

Este análisis nos permite concluir que la infraestructura física de las salas de laboratorios posee un bajo nivel de uso por parte de los estudiantes, por lo que se convierte en un candidato excelente para el despliegue de plataformas oportunistas. Por otro lado, los resultados del nivel de uso de CPU indicaron que no hay una

diferencia significativa en los usos de CPU entre las máquinas, lo que implica que éste no es un criterio válido para la selección de las máquinas oportunistas a desplegar.

El segundo objetivo del estudio era identificar las máquinas que más momentos de no disponibilidad presentaban, luego de procesar la información se obtuvo un mapa por cada una de las salas indicando el número de reinicios esperados por cada máquina en un día. Se añadió una escala de colores a cada máquina donde los colores más cercanos a verde representan los equipos con menos número de fallas esperadas al día. Las máquinas en rojo representan las máquinas son mayor número de fallas esperado y las máquinas en negro son máquinas que no se pudieron monitorear por lo que se les asignó un valor igual al promedio del resto de las máquinas de la sala.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ISC201 0,19 | ISC202 0,39 | ISC203 0,25 | ISC204 0,25 | ISC205 0,21 | ISC206 0,5 | ISC301 0,16 | ISC302 0,137 | ISC303 0,172 | ISC304 0,266 | ISC305 0,31 | ISC306 0,16 |
| ISC207 0,59 | ISC208 0,23 | ISC209 0,21 | ISC210 0,16 | ISC211 0,18 | ISC212 0,18 | ISC307 0,2 | ISC308 0,23 | ISC309 0,3 | ISC310 0,218 | ISC311 0,193 | ISC312 0,125 |
| ISC213 0,21 | ISC214 0,16 | ISC215 0,13 | ISC216 0,19 | ISC217 0,21 | ISC218 0,18 | ISC313 0,161 | ISC314 0,25 | ISC315 0,25 | ISC316 0,2 | ISC317 0,23 | ISC318 0,218 |
| ISC219 0,24 | ISC220 0,103 | ISC221 0,129 | ISC222 0,062 | ISC223 0,129 | ISC224 0,125 | ISC319 0,2 | ISC320 0,23 | ISC321 0,2 | ISC322 0,258 | ISC323 0,21 | ISC324 0,28 |
| ISC225 0,096 | ISC226 0,096 | ISC227 0,096 | ISC228 0,096 | ISC229 0,36 | ISC230 0,1 | ISC325 0,23 | ISC326 0,2 | ISC327 0,25 | ISC328 0,24 | ISC329 0,24 | ISC330 0,24 |
| ISC231 0,137 | ISC232 0,133 | ISC233 0,161 | ISC234 0,2 | ISC235 0,16 | | ISC331 0,43 | ISC332 0,2 | ISC333 0,28 | ISC334 0,17 | ISC335 0,22 | |

| | | | | | |
|---|---|---|---|---|---|
| ISC401 0,21 | ISC402 0,16 | ISC403 0,125 | ISC404 0,368 | ISC405 0,22 | ISC406 0,176 |
| ISC407 0,105 | ISC408 0,1 | ISC409 0,052 | ISC410 0,2 | ISC411 0,232 | ISC412 0,13 |
| ISC413 0,27 | ISC414 0,2 | ISC415 0,22 | ISC416 0,25 | ISC417 0,2 | ISC418 0,33 |
| ISC419 0,16 | ISC420 0,35 | ISC421 0,232 | ISC422 0,27 | ISC423 0,25 | ISC424 0,232 |
| ISC425 0,16 | ISC426 0,11 | ISC427 0,31 | ISC428 0,28 | ISC429 0,16 | ISC430 0,23 |
| ISC431 0,58 | ISC432 0,5 | ISC433 0,26 | ISC434 0,2 | ISC435 0,232 | |

**Ilustración 34: Mapa de riesgo de fallas de los laboratorios Waira1, Waira2 y Alan Turing**

Al ser el comportamiento de una máquina independiente del comportamiento de las otras máquinas, el número de fallas esperadas por cada sala se puede calcular como la suma de las fallas de cada uno de los computadores. Realizando este cálculo se obtienen los siguientes resultados:

| Sala | Número de Reinicios Esperados |
|---|---|
| **Waira 1** | 6.935 |
| **Waira 2** | 7.927 |
| **Alan Turing** | 8.064 |

**Tabla 2: Número de fallas diarias en promedio por laboratorio**

La información sobre el número de reinicios por cada una de las máquinas fue ingresada en la base de datos de UnaCloud, en la tabla de la infraestructura física. Esta información es utilizada por UnaCloud para seleccionar las mejores máquinas a desplegar tratando de minimizar la probabilidad de que una máquina desplegada sea apagada o reiniciada. Este esquema mejora la base de mejor esfuerzo con la que cuenta UnaCloud puesto que se escogen las máquinas que a través de la historia han presentado un mejor comportamiento. Si bien este modelamiento de la infraestructura nos permitió conocer ciertos patrones de uso, el comportamiento de los estudiantes con respecto a las máquinas varía dependiendo a muchos factores como: La carga académica y el horario de las clases, las fechas de entregas de trabajos en las diferentes materias, la ubicación de cada máquina entre otros. Debido a todos estos factores, el modelo que se hizo puede funcionar bien en el corto plazo, pero fácilmente puede quedar desactualizado, por lo que un modelo de retroalimentación continuo donde la información de disponibilidad se actualice automáticamente a partir de las mediciones hechas periódicamente, permitiría incorporar información más actual para poder predecir mejor el comportamiento de las máquinas.

## 6.2 Workflow MSA:

Esta primera capa de UnaCloud MSA consiste en el desarrollo e integración de las diferentes herramientas computacionales que realizan los respectivos análisis del archivo de secuencias que ingresa el investigador en biología.

El workflow se implementó utilizando el motor de workflows Loni Pipeline. Esta selección se tomó debido a que cumplía los requerimientos de integración con infraestructuras grid, facilidad de incorporación de nuevos módulos desarrollados, facilidad en el desarrollo de workflows mediante una interface gráfica. Si bien Loni Pipeline no fue desarrollado para la implementación de workflows bioinformáticos, este aspecto no fue relevante ya que debido a su flexibilidad, cualquier tipo de aplicación ejecutada por consola se puede integrar. Adicionalmente, dentro del grupo de investigación, se han desarrollado varios proyectos con esta herramienta, esta experiencia fue importante también para la selección ya que se conocen muchos detalles técnicos de la instalación lo que facilita su implementación. Otro aspecto por el cual se escogió Loni Pipeline es por su facilidad para ser integrado con una infraestructura grid. Si bien Taverna, el principal competidor de Loni, también ofrece esta posibilidad de integración a través de GridSAM, en la práctica se ha demostrado que a mayor número de capas intermedias, aumentan también los posibles errores que pueden surgir. Taverna invoca un WebService expuesto por GridSAM que a su vez se comunica con el manejador de recursos de Globus GRAM, y éste finalmente se comunica con el sistema de scheduling. En Loni Pipeline esta integración es más directa, ya que el servidor de Loni, se comunica directamente con SGE o el sistema de scheduling correspondiente para hacer la invocación a través del plugin de JGDI o DRMAA. El último aspecto que se consideró para esta selección fue la necesidad de recompilar las herramientas que se iban a incorporar. Este proceso es demasiado dispendioso y con algunas herramientas imposible ya que no se tiene su código fuente. Todos estos factores definieron la seleeción sde Loni Pipeline como motor de procesos a utilizar.

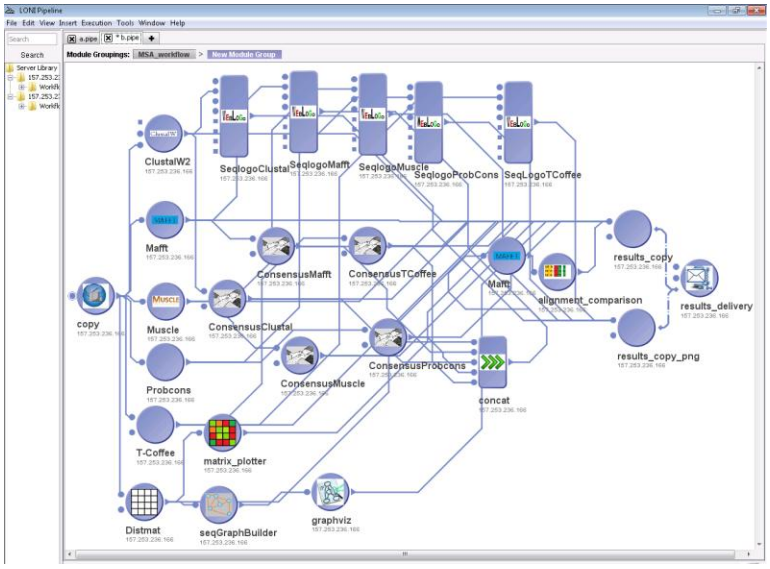


**Ilustración 35: Implementación del Workflow MSA en Loni Pipeline**

Los diferentes módulos o tareas del Workflow MSA se pueden agrupar dependiendo de la funcionalidad o tipo de análisis que realicen.

### 6.2.1 Generación de Alineamientos:

La primera parte del Workflow MSA se encarga de la generación de diferentes alineamientos utilizando distintos programas. En Workflow MSA se incluyeron los programas ClustalW2[Clu01], Muscle[Mus01], Mafft[Maf01, Maf02], ProbCons[Pro01, Pro02] y TCoffee[Tco01] debido a que cada uno emplea diferentes estrategias (Alineamiento Progresivo, Refinamiento Iterativo, Basado en una Transformada Rápida de Fourier, Alineamiento basado en Distribuciones de Probabilidad y Alineamiento Basado en Consistencia respectivamente) garantizando un grado de variedad en los métodos utilizados.

### 6.2.2 Creación de Logos

Una vez generados los diferentes alineamientos, uno de los análisis más interesantes que se pueden hacer sobre éstos es la generación de logos. Para este paso se utilizó el programa seqlogo. Si bien seqlogo admite un gran número de parámetros para presentar los resultados, en el workflow MSA, se configuraron solamente los necesarios como el formato de archivo a generar, el número de caracteres por línea y su longitud y la posibilidad de que los alineamientos se generen con colores. Al igual que con la generación de los alineamientos, al establecer los valores de los parámetros se le resta un poco de flexibilidad al usuario final, pero por otro lado se le simplifica el proceso de análisis que es donde está el trabajo principal.

**Ilustración 36: Ejemplo de un Logo Generado en el Workflow MSA**

### 6.2.3 Creación de Consensos

Con el fin de tener información resumida sobre cada uno de los 5 alineamientos generados, se utilizó el programa *cons* [Con02] de la suite EMBOSS[Emb01]. Para calcular el aminoácido de una columna dada se

calcula el score para la secuencia i en esa columna. Este score se calcula como la suma sobre todos los *j* (*j≠i*) de *score(i,j)*weight(j)* donde *score(i,j)* por defecto se calcula utilizando la matriz de Blosum62 y donde *weight(j)* es el peso de la secuencia j que por defecto es igual para todas las secuencias. Finalmente en la columna se coloca el aminoácido que obtuvo un mayor puntaje en este score. Si el mayor puntaje de algún aminoácido no supera un valor mínimo, en la columna se coloca el carácter x indicando que no hubo un consenso en esa posición del alineamiento.

### 6.2.4 Comparación de Consensos

Al tener 5 alineamientos generados por programas diferentes, surge naturalmente la idea de realizar una comparación para detectar regiones que todos los programas alinean de manera similar y regiones ambiguas que son regiones que programas diferentes alinean de manera diferente. Evidentemente las regiones consistentes entre los diferentes alineamientos son regiones sobre las que se puede tener mayor nivel de confianza y las regiones ambiguas, dependiendo del caso, pueden ser descartadas del alineamiento para mejorar su calidad. Con el fin de realizar esta comparación se partió del consenso debido a la fácil manipulación de ésta al tratarse de sólo una secuencia. Como los consensos no todos necesariamente tienen la misma longitud, se realiza un alineamiento múltiple entre los 5 alineamientos utilizando MAFFT (Inicialmente se había escogido T-COFEE porque en principio genera mejores resultados, pero empíricamente se observó que cuando en este caso, como el número de secuencias a alinear es de solamente 5, no se generan buenos resultados). A partir del alineamiento de los consensos, se desarrolló el módulo alignment_comparison. Éste consiste en una aplicación java que recibe como entrada el alineamiento de los 5 consensos y revisa por cada columna que tan de acuerdo estuvieron los programas. Si 5 ó 4 alineamientos en esa posición tienen la misma letra, se asigna un color verde a ese carácter. Si 3 alineamientos arrojan el mismo carácter, entonces se asigna un color amarillo. Si 2 alineamientos concordaron se asigna un color anaranjado y de lo contrario se asigna un color rojo. Luego se vuelven a pintar los 5 consensos, sin alinear utilizando los colores definidos. Así se puede detectar sobre cada uno de los alineamientos, que regiones son ambiguas y cuales consistentes y sobre cada uno de los alineamientos tomar las acciones que el investigador considere adecuadas.

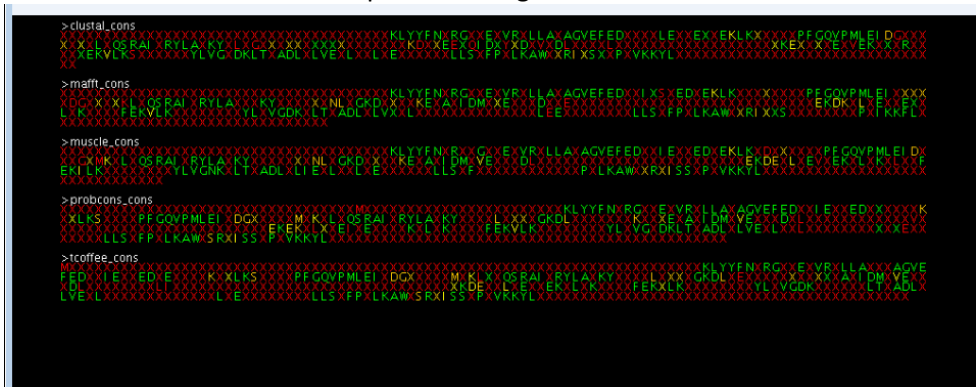


**Ilustración 37: Ejemplo de un archivo de resultado del programa alignment_comparison**

### 6.2.5 Clustering

El objetivo de este grupo consiste en el análisis de las secuencias a partir de su similaridad. Al momento de generar un alineamiento múltiple se debe tener precaución con utilizar grupos de secuencias muy similares entre ellas, pues esto degrada la calidad del alineamiento.

El primer job que se ejecuta es distmat[Dis01] de la suite EMBOSS[Emb01], cuya función consiste en calcular la matriz de distancia de las secuencias. Para cada par de secuencias, se calcula su distancia utilizando el conteo de k-meros. El número de entradas en la matriz $O(n^2)$ donde *n* es el número se secuencias en el archivo de entrada. Como la función de distancia es conmutativa, es decir sim(A,B) = sim(B,A) y como no se tiene en cuenta la comparación de una secuencia con ella misma, el número exacto de entradas en la matriz es n(n-1)/2.

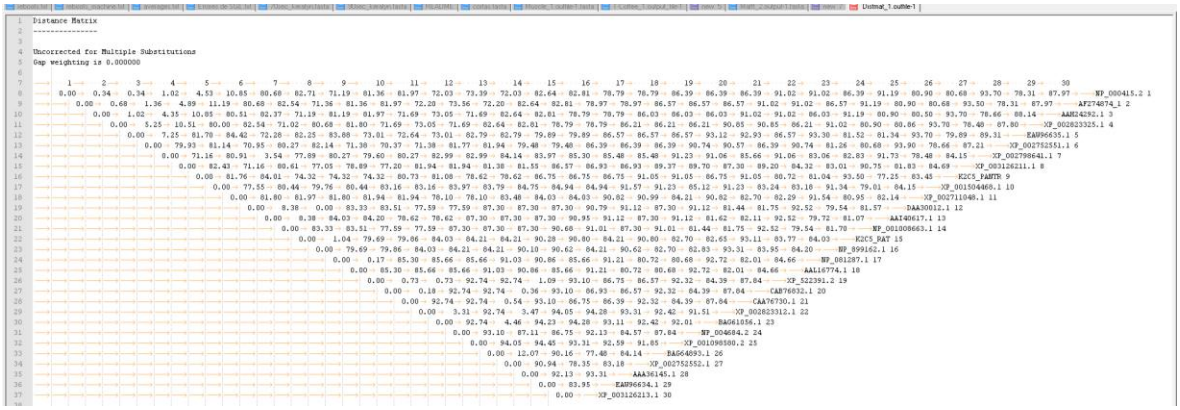


**Ilustración 38: Ejemplo de una matriz de distancias generada con el programa distmat**

Debido a que cuando el número de secuencias aumenta, el número de entradas en la matriz aumenta de manera proporcional al cuadrado de las secuencias, la matriz generada con distmat se vuelve impráctica para su análisis manual. Con un archivo pequeño de 30 secuencias, el número de entradas en la matriz es de 435 y con un archivo un poco mayor de 50 secuencias, la matriz posee 1225 entradas. Debido a esto, se define el job matrix plotter que se encarga de generar una gráfica de la matriz y asociarle una escala de colores a cada valor de la matriz, con lo que se identifican fácilmente los valores pequeños que son los que identifican parejas de secuencias cercanas. A cada posible valor en la matriz de distancia, el cual es un número decimal entre 0 y 100, se le asigna un color basado en la siguiente función:

```
private static int parseColor(float value){
    value=100-value;
    int red,green;
    if(value<=50){
        green=255;
        red= (int)(255.0*value/50.0);
    }
    else{
        red=255;
        green=510-  (int)(255.0*value/50.0);
    }
    return red*256*256+green*256;
}
```

La función parseColor recibe un valor entre 0 y 100 y calcula las componentes rojo y verde. A medida que el número sea más cercano a 0 (Secuencias más similares) la componente roja es mayor y la componente verde menor. Análogamente, a medida que las secuencias sean más diferentes, value es más cercano a 100 y el color asignado tiende a un color más verde. La imagen generada permite ser analizada manualmente de manera más fácil, pues detectar parejas de secuencias cercanas se convierte en detectar las entradas rojas de la matriz.

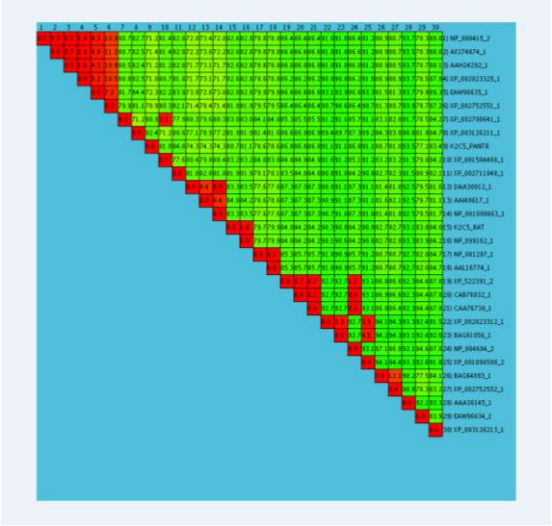

**Ilustración 39: Ejemplo de una matriz generada con el programa matrix_plotter a partir de la matriz generada por distmat**

Aunque detectar las parejas de secuencias similares sea una labor importante, por lo general dentro del grupo de secuencias no sólo se encuentran parejas cercanas, sino que se presentan grupos donde todas las secuencias del grupo son similares entre ellas y diferentes al resto de secuencias. Si bien existen diferentes técnicas avanzadas para realizar clustering de secuencias de ADN y de proteínas, en workflow MSA sólo se tomó en cuenta la matriz de distancia. Para la generación de los clusters de secuencias, se interpretó la matriz de distancias como un grafo con costos y no dirigido donde las secuencias de proteínas son los vértices y la distancia entre cada par de secuencias corresponde al peso del arco entre la pareja. Para generar una imagen indicando los clusters presenten en las secuencias, se utilizó el programa GraphViz[Gra01, Gra02] cuya funcionalidad principal es la creación de representaciones gráficas de arcos. En GraphViz se especifican los vértices y los arcos del grafo, y se puede establecer la longitud de cada arco. Esta última característica fue aprovechada para lograr que en el gráfico, la longitud del arco dibujado fuera proporcional al valor de la distancia entre las dos secuencias. GraphViz ofrece diferentes modos para la generación de los gráficos dependiendo del tipo de grafo que se esté representando. El modo NEATO[Nea01] fue el modo seleccionado por estar dedicado a los grafos no dirigidos. Como el formato de distmat no es compatible con el que utiliza GraphViz, se implementó el job seqGrahBuilder que se encarga de tranformar la matriz de distancias generada por Distmat al formato requerido por Graphviz. En este paso también se hizo un filtro de las distancias mayores a un valor T por dos razones principales: Primero, porque para el análisis de clusters nos interesan los arcos de menor costo y segundo porque al ser el número de arcos muy grande, el programa GraphViz empezaba a fallar por limitaciones de memoria principal.

En la figura 40 se observa un ejemplo de un archivo de resultado de Graphviz generando el grafo de las secuencias. Los grupos de secuencias unidos por arcos (Líneas rojas) son los diferentes clusters que se presentan en las secuencias originales. La distancia entre dos secuencias en la imagen es proporcional a la distancia biológica de la secuencia.

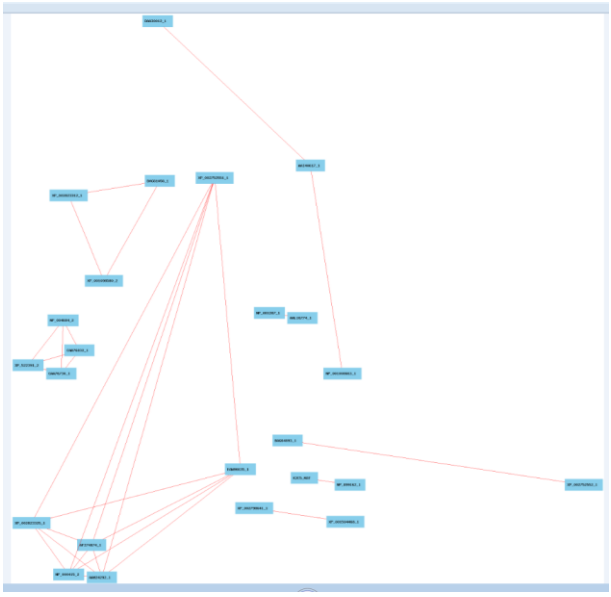

**Ilustración 40: Ejemplo de resultado de la tarea de Clustering**

### 6.2.6 Tareas Auxiliares

Las otras tareas del workflow corresponden a procesos de copiado de archivos que no realizan análisis sobre las secuencias, pero son necesarios para garantizar el correcto flujo de los archivos. El job copy se encarga de copiar el archivo de secuencias desde la carpeta auxiliar donde lo deja la aplicación web a la carpeta de trabajo de LoniPipeline. La tarea de results_copy se encarga de hacer el proceso contrario, es decir copiar los archivos de resultados a la carpeta donde la aplicación web puede encontrarlos para presentárselos al usuario. El job results_copy_png se encarga de hacer lo mismo pero con los archivos de extensión .png de imágenes generados por SeqLogo. ResultsDelivery se encarga de comprimir todos los archivos de resultados en un archivo .zip y de copiar este archivo a la carpeta de resultados de la aplicación web. El archivo comprimido puede ser descargado por el usuario final para su comodidad. El job concat se encarga de recibir los 5 consensos generados a partir de cada uno de los alineamientos y de concatenarlos en un solo archivo que se le pasa como parámetro de entrada al job de Alignment_Comparator.

## 6.3 Interface Web:

Esta capa consiste en la capa de interacción del usuario con la aplicación. El usuario accede vía web a la aplicación UnaCloud-MSA y a través de ésta carga su archivo de proteínas en formato multifasta, realiza la ejecución de la tarea y obtiene sus resultados vía web. La capa web se encarga de solicitar la invocación del workflow al servidor de LoniPipeline. Una vez invocada la ejecución del workflow, la capa web se encarga de esperar a que los resultados se encuentren listos y de presentarlos vía web. También se le ofrece al usuario la posibilidad de descargar los resultados como un archivo comprimido.

El desarrollo de la capa web de UnaCloud MSA se realizó sobre Java JDK 1.6 [Jav01] utilizando el framework de JSF versión 1.2 [Jsf01] y las librerías de Richfaces 3.2 [Ric01]. Como servidor de aplicaciones se utilizó GlassFish 2.1 [Gla01].

La aplicación cuenta con dos diferentes páginas jsp:

La página de emtrada esindex.jsp que es la entrada a la aplicación, en esta página el usuario ingresa su archivo multifasta de secuencias de aminoácidos y selecciona el botón de Submit Job. La página cuenta con un componente de FileUpload que se encarga de la transferencia del archivo desde el computador del usuario de la aplicación al servidor web.

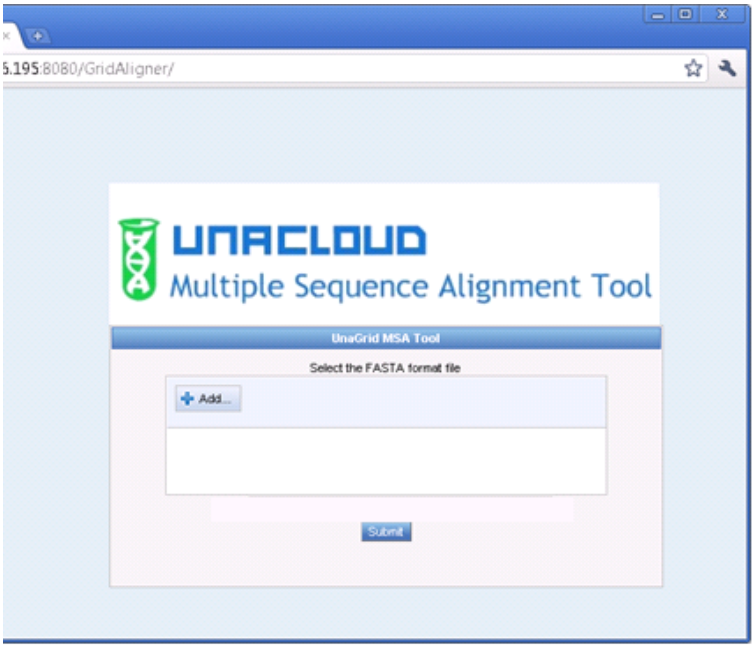


**Ilustración 41: Página principal de la Aplicación Web UnaCloud MSA**

La página de presentación de resultados es results.jsp, al lanzar la ejecución del job, al usuario se le presenta un mensaje indicando que su trabajo está siendo ejecutado y se le indica el tiempo que el workflow ha tomado en su ejecución. Al terminar el workflow, se le presenta un mensaje indicando que sus resultados están listos y se le presenta un menú con un componente de SimpleTogglePanel por cada uno de los diferentes tipos de resultados: Alineamientos Generados y su visualización a través de la herramienta JalView, Logos, Consensos, Comparación de Alineamientos, Matriz de Distancias , Clustering y un panel adicional donde el usuario puede descargas sus resultados a su computador. El componente SimpleTogglePanel, perteneciente al repertorio de componentes gráficos de la librería Richfaces, permite abrir y cerrar el panel oprimiendo en la flecha del encabezado del panel, Así al usuario inicialmente se le presentan todos los paneles colapsados y él decide que elementos le interesan ver.

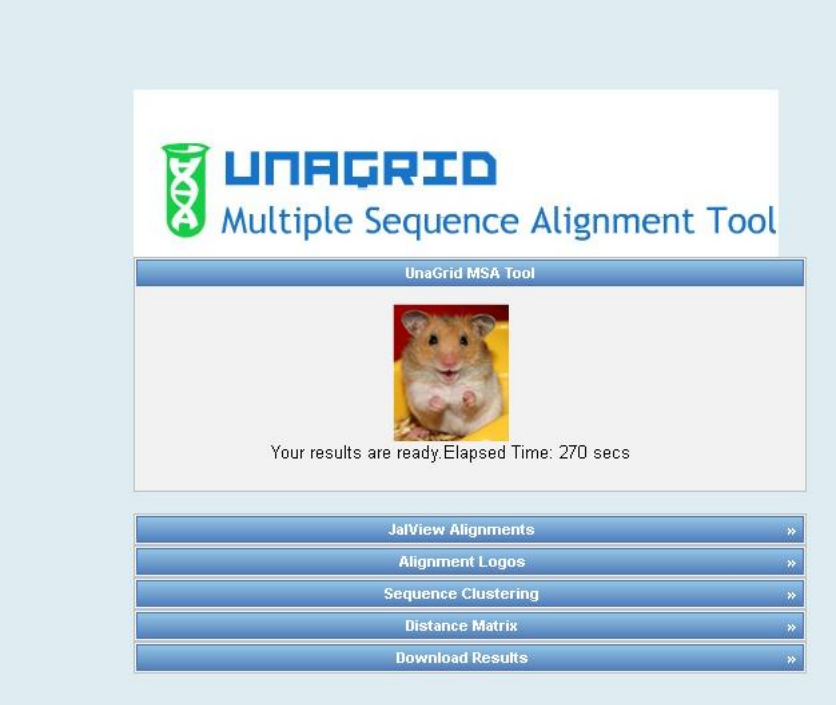


**Ilustración 42: Página de Resultados con los diferentes paneles por cada tipo de resultado**

Para la observación de las diferentes imágenes creadas se utilizó la librería JavaScript MojoZoom[Moj01] que añade la funcionalidad de generar zoom sobre un área de la imagen. Esto permite enfocar en un área en particular de alguna de las imágenes creadas para tener información más precisa.

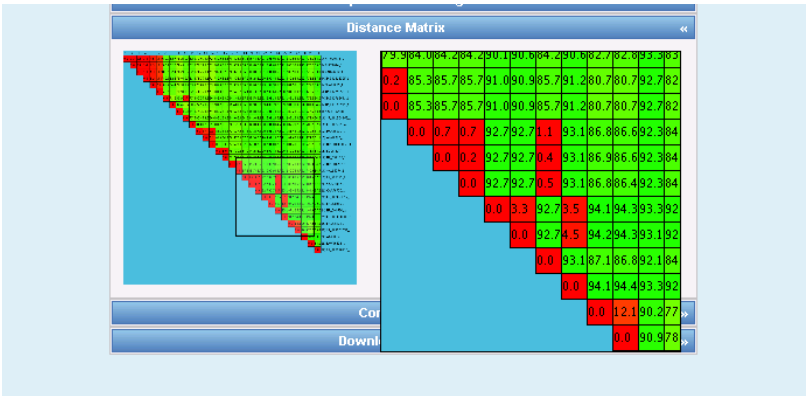


**Ilustración 43: Ejemplo de uso de zoom para enfocar valores bajos en la matriz de distancias**

## 6.4 Mecanismo de despliegue dinámico de máquinas virtuales: OpportunisticDeployer

UnaCloud fue pensado y ha sido utilizado para el despliegue de cluster virtuales que ejecutan tareas de larga duración. Ejemplos de estos trabajos son [Bac01], [Bio02] en los cuales desde el inicio de la ejecución del cluster se tenía conocimiento del número de jobs que se iban a ejecutar. UnaCloud MSA por su parte ofrece su servicio de ejecución de workflow como un servicio a través de una interface web, accesible a cualquier usuario desde cualquier parte del mundo. Las aplicaciones web, al poseer un comportamiento diferente al de un grupo de tareas que se ejecutan en lote, requieren también de un modelo diferente de despliegue. Debido a esto el modelo de despliegue estático que se tenía con UnaCloud se vuelve insuficiente ya que no permite el despliegue de máquinas virtuales de manera dinámica y sería labor del responsable del cluster monitorear y

levantar máquinas cuando se requieran. Para remediar esta situación, en UnaCloud MSA se propone un modelo híbrido, donde hay una infraestructura dedicada que soporta la funcionalidad de la aplicación, y cuando esta infraestructura deja de ser suficiente debido a un incremento en el nivel de carga, se acude a un despliegue oportunista de máquinas virtuales en los laboratorios de computación. Este modelo también toma en cuenta el hecho de que la infraestructura oportunista es compartida entre los diferentes grupos de investigación y que no se deben asignar todos los recursos computacionales a un cluster virtual si no se van a utilizar. El mecanismo de despliegue dinámico, denominado OpportunisticDeployer debido a que utiliza de manera oportunista el poder de cómputo ocioso en las salas de computadores, está desarrollado como una aplicación java que se corre en el computador maestro del cluster. OpportunisticDeployer se constituye en un primer piloto por incorporar un modelo elástico a UnaCloud. Cuenta con los siguientes elementos:

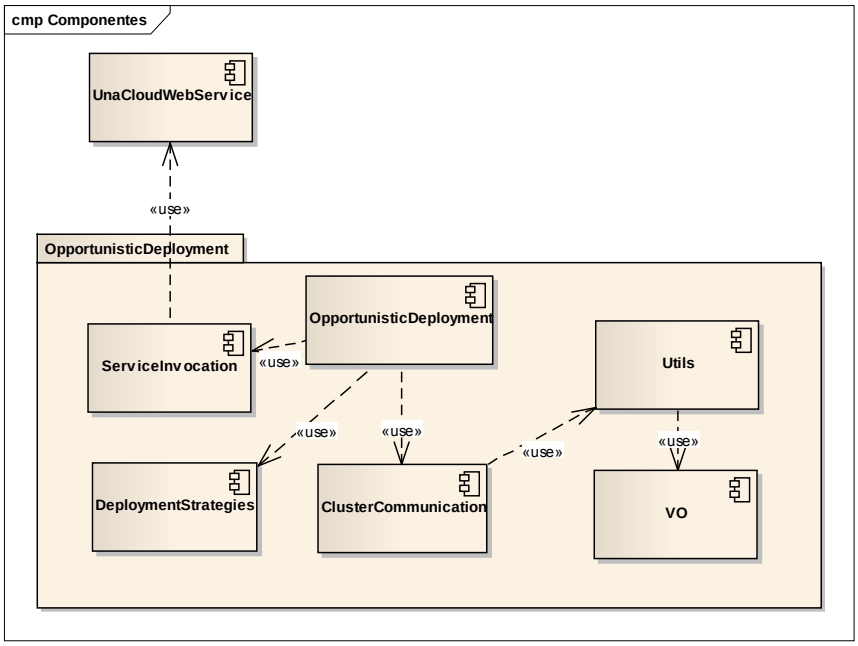


**Ilustración 44: Diagrama de Componentes de OpportunisticDeployer**

### 6.4.1 Monitoreo del Cluster:

El monitoreo del estado de la infraestructura se convierte en la primera tarea a realizar en un modelo de despliegue dinámico. El componente de ClusterCommunication se encarga del monitoreo constante del cluster, revisando tanto la cola de jobs como de las máquinas que se tienen disponibles en el cluster. Se definió la interface IClusterMonitoring que define la información mínima que se debe conocer del cluster para determinar su nivel de carga. Provee los métodos getHosts que retorna los hosts del cluster que se encuentran prendidos y los disponibles para prender en caso de ser necesario. También provee el método getJobs que retorna la lista de tareas que se encuentran en la cola de trabajos. Como se implementó el Workflow MSA sobre Loni Pipeline y éste corre sobre SGE, se realizó la implementación de esta interface llamada SGE_Monitor que se encarga de proveer esta información sobre un cluster basado en SGE. Esta implementación se basa en los comandos qhost y qstat que provee SGE y que permiten conocer la información de los hosts y de la cola de trabajos respectivamente. SGE_Monitor se encarga de ejecutar el comando a nivel de sistema operativo, de recibir y procesar la respuesta y armar los respectivos VOs. La separación entre la interface IClusterMonitoring y su implementación, permite la fácil incorporación de nuevas implementaciones para diferentes sistemas de scheduling como Condor[Con01], PBS [Pbs01] entre otros.

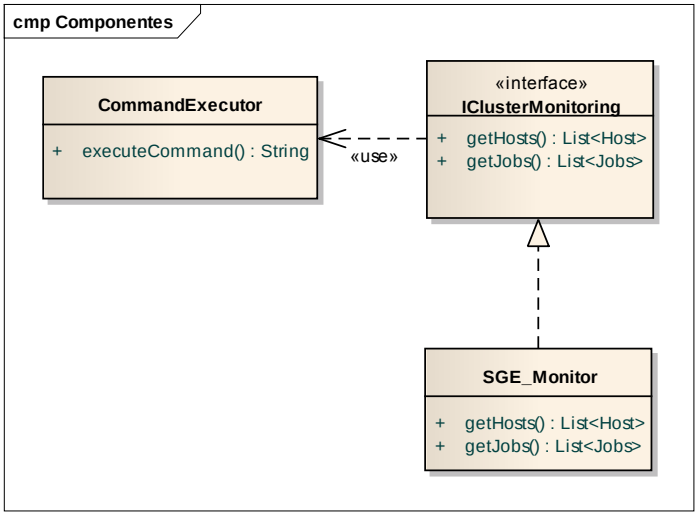


**Ilustración 45: Componente de Monitoreo de Cluster**

### 6.4.2 Estrategia de Despliegue:

Este componente de OpportunisticDeployer se encarga de utilizar la información que provee IClusterMonitoring para determinar si el cluster se encuentra con un alto nivel de carga y de calcular el número de máquinas que serán levantadas. En este componente se definió la interface IDeploymentStrategy cuyo único método recibe los VOs de los jobs y de los hosts y retorna el número de máquinas que deberían ser levantadas de acuerdo a su estrategia. Se implementaron 3 estrategias diferentes:

- AverageLoad: Esta primera estrategia de despliegue revisa el nivel de carga de cada una de las máquinas prendidas, y calcula el mínimo número de máquinas necesarias para garantizar que la carga promedio sea menor a un valor T. Es decir, se busca el valor mínimo *M* para el cual se satisfaga la expresión

$$\frac{\sum_{i \in TurnedOnMachines} Load_i}{N + M} \leq T$$

  Donde $Load_i$ es la carga de las máquinas que actualmente se encuentran prendidas (Máquinas dedicadas y de las salas de computadores), *N* es el número de las máquinas prendidas, T es la carga promedio límite y *M* es el número de máquinas que se van a levantar. Esta estrategia probó no ser tan buena puesto que el nivel de carga de las máquinas no es reportado muy precisamente. Además, esta estrategia no toma en cuenta la cola de tareas actual, por lo cual podría pasar que el despliegue de nuevas máquinas no mejore el rendimiento. Para determinar el comportamiento de esta estrategia al ir variando el parámetro T, se hicieron pruebas con valores de 0.4, 0.5 y 0.75.
- JobsCount: Esta estrategia se basa en el hecho de que todos los jobs que se encuentran en la cola de trabajos son independientes entre ellos, por lo tanto simplemente define que el número de máquinas a levantar es el número de jobs en la cola.
- WeightedJobCount: Esta estrategia extiende la de JobsCount. No todos los jobs del workflow duran lo mismo ni imponen un nivel de carga del procesador de manera igual. Debido a esto, esta estrategia se basa en que cada uno de los diferentes jobs tenga asociado un peso. Adicionalmente se tiene en cuenta que experimentalmente se ha determinado que el tiempo desde que se da la orden para que una máquina virtual sea encendida hasta que está lista y disponible en el cluster es de unos 90 segundos. Esto implica que aunque se tengan muchos trabajos en cola, si éstos duran poco, es mejor no levantar máquinas virtuales pues podría pasar que para el tiempo en que ésta esté disponible, ya las tareas hayan terminado. Así que el número de máquinas a levantar es igual a la suma de los pesos de

cada tarea. Para poder usar esta estrategia, el desarrollador debe determinar adecuadamente estos pesos y establecerlos en un archivo de configuración.

$$M = \sum_{j \in ListJobs} Weight(Job_j)$$

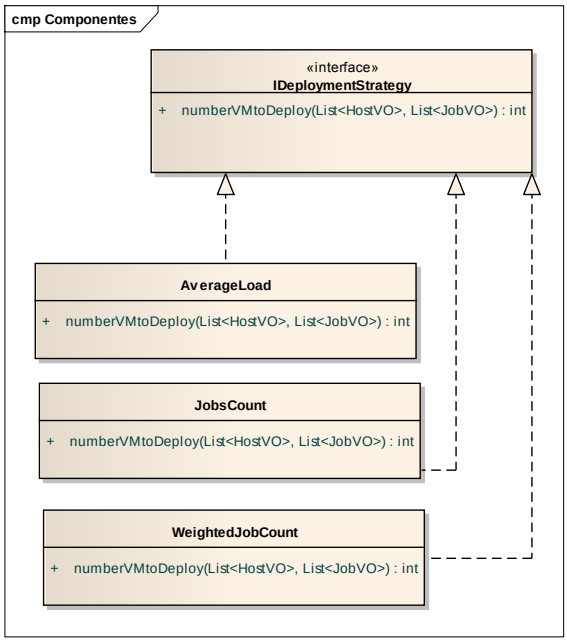


**Ilustración 46: Componente de Estrategia de Despliegue**

### 6.4.3 Invocación y WebService:

Este componente tiene como responsabilidad realizar la invocación del WebService que ofrece el servidor de UnaCloud. El software de UnaCloud se extendió con la funcionalidad de levantar máquinas virtuales mediante un servicio Web. Desde la aplicación de OpportunisticDeployer se importó el WSDL que definía al WS.
Los parámetros que recibe el WebService son: La ruta de la máquina virtual a desplegar, el número de máquinas, el número de horas durante las cuales se van a desplegar las máquinas, el número de cores y la memoria RAM (En megabytes) asignados a cada máquina virtual.

```
 */
public class ServiceInvocator {

    public static void invoke(int numberMachines, int executionHours, int cores, int ram){
        try { // Call Web Service Operation
        OnDemandDeploymentWebServiceService service =
                new OnDemandDeploymentWebServiceService();
        OnDemandDeploymentWebService port = service.getOnDemandDeploymentWebServicePort();
        String route = PropertiesReader.getInstance().getValue("virtual_machine_path");
        String userName   = PropertiesReader.getInstance().getValue("userName");
        String passwd     = PropertiesReader.getInstance().getValue("password");
        port.deployMachine(route, numberMachines, userName,
                passwd, executionHours, cores, ram);
    } catch (Exception ex) {
        ex.printStackTrace();
    }
    }
```

**Ilustración 47: Invocación del WebService desde OpportunisticDeployer**

El WS deployMachine en UnaCloud realiza los siguientes pasos:

- Realiza una consulta SQL para determinar de todas las máquinas, cuáles están disponibles para ser desplegadas. Esta consulta retorna las máquinas virtuales donde la máquina física esté prendida, que tenga el agente funcionando correctamente y que no esté ejecutando actualmente alguna otra máquina virtual.
- Con la lista de máquinas disponibles, las ordena de manera ascendente de acuerdo al número de fallas esperadas (Apagones y reinicios).
- Si el número de máquinas requeridas *n* es menor al número de máquinas disponibles, levanta las mejores *n* máquinas. De lo contrario, levanta todas las máquinas disponibles.

```
public OnDemandDeploymentBean() {
}

public void deployMachine(String route, int numberMachines, String userName, String passwd,int executionHours, int cores, int ram) {
    System.out.println("WS");
    List vms =  persistence.executeNativeSQLList(Queries.getVirtualMachine(route,userName),VirtualMachine.class);
    Collections.sort(vms,new Comparator<VirtualMachine>(){
        public int compare(VirtualMachine o1, VirtualMachine o2) {
            PhysicalMachine p1 = o1.getPhysicalMachine();
            PhysicalMachine p2 = o2.getPhysicalMachine();
            int a = Double.compare(p1.getExpectedFailures(),p2.getExpectedFailures());
            return a==0?Double.compare(p2.getAverageAvailability(),p1.getAverageAvailability()):a;
        }
    });
    ArrayList<VirtualMachine> vmPrender=new ArrayList<VirtualMachine>(numberMachines);
    for(int e=0,i=Math.min(vms.size(),numberMachines);e<i;e++)
        vmPrender.add((VirtualMachine)vms.get(e));
    System.out.println("Levantar WS: "+vmPrender.size());
    for(VirtualMachine v:vmPrender)
        System.out.println(v.getVirtualMachineName()+" "+v.getPhysicalMachine().getPhysicalMachineName());
    System.out.println(executionHours+" "+userName);
    vmServices.turnOnVirtualMachineCloud(vmPrender, executionHours, cores, ram, userName);
}
```

**Ilustración 48: WebService deployMachine en UnaCloud**

## 6.5 Implementación

Para la implementación de UnaCloud MSA se creó un creó un cluster virtual personalizado (CVC). Se utilizó el sistema de administración de trabajos Sun Grid Engine (SGE) [Sge01]. Adicionalmente se integró con Loni Pipeline a través del plugin JGDI [Jgd01] lo que permite que la ejecución de cada uno de los módulos de un

Workflow se ejecuten sean convertidos en jobs para el scheduler SGE. Con esto se le delega la tarea de asignar los jobs en la cola a las diferentes máquinas virtuales al planificador. Se decidió que el servidor tanto web, como el de Loni Pipeline y el de SGE estuvieran en la misma máquina para evitar problemas de conectividad, adicionalmente en la práctica se observa que los tres servicios no son muy demandantes en recursos por lo que la máquina soporta con buen desempeño los tres servicios.

Para el servidor se decidió utilizar una máquina virtual en el servidor VMWare ESX [Vmw02] del grupo Comit debido a su alta disponibilidad. Adicionalmente se decidió el despliegue de 5 máquinas virtuales esclavas del cluster SGE en el mismo servidor ESX, para que hubiera una infraestructura de alta disponibilidad dándole soporte a UnaCloud MSA. En todas las máquinas virtuales se instaló Debian 5 [Deb01] debido a su fácil integración con el software de SGE y de LoniPipeline. Para compartir archivos entre todas las máquinas, tanto servidor como esclavas, se utilizó un servidor de almacenamiento NAS [Nas01]. El CVC se despliega en los laboratorios de computadores, utilizando el servicio web ofrecido por UnaCloud.

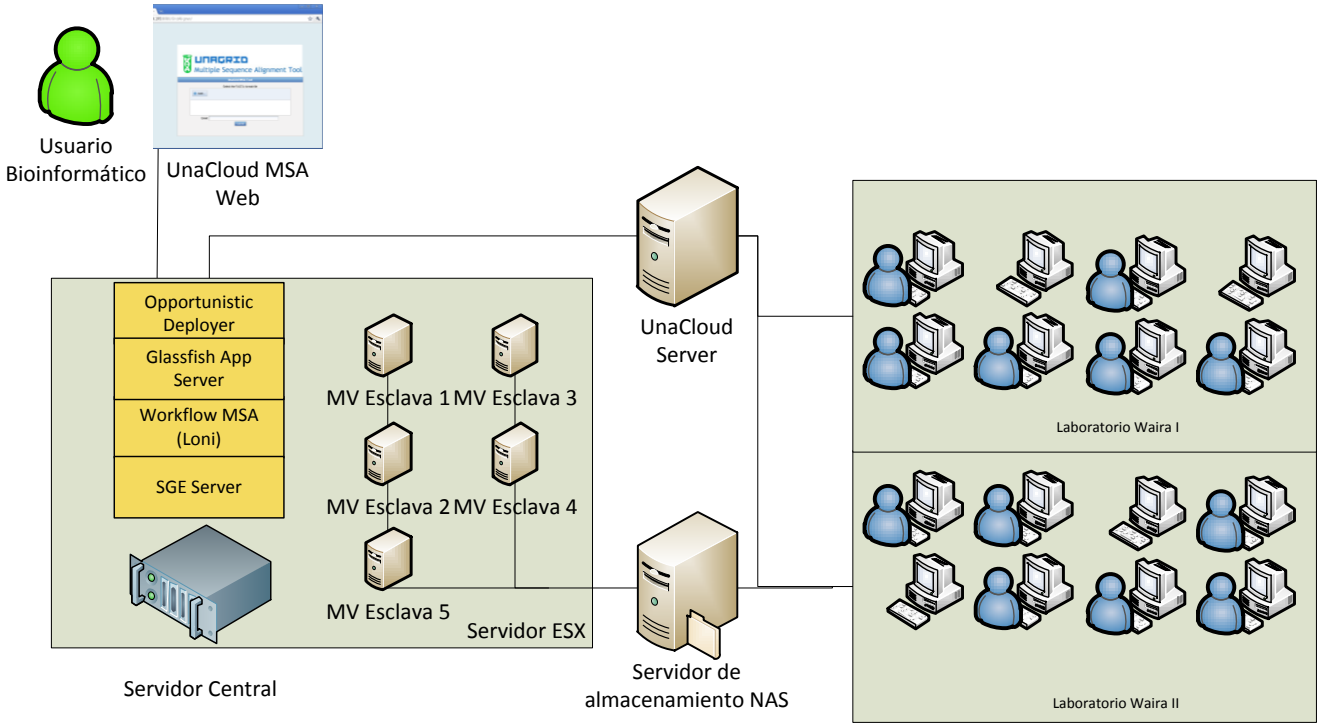


**Ilustración 49: Arquitectura de UnaCloud MSA**

# 7 Evaluación

## 7.1 Validación de Resultados

UnaCloud MSA se puede entender como un sistema de apoyo a la toma de decisiones, donde con la información extra obtenida a partir de la herramienta, el investigador biológico toma una decisión (Que alineamiento utilizar, descartar secuencias porque hay un grupo muy grande de secuencias muy cercanas, eliminar una región ambigua). En esta validación, presentamos una manera de interpretar los resultados de los componentes desarrollados.

### 7.1.1 Alignment Comparator

El componente Alignment Comparator identifica regiones en los cuales los 5 alineamientos concuerdan y regiones en los cuales los 5 alineamientos tuvieron problemas para alinear las secuencias. Esta identificación permite encontrar regiones bastante conservadas en un alineamiento múltiple.

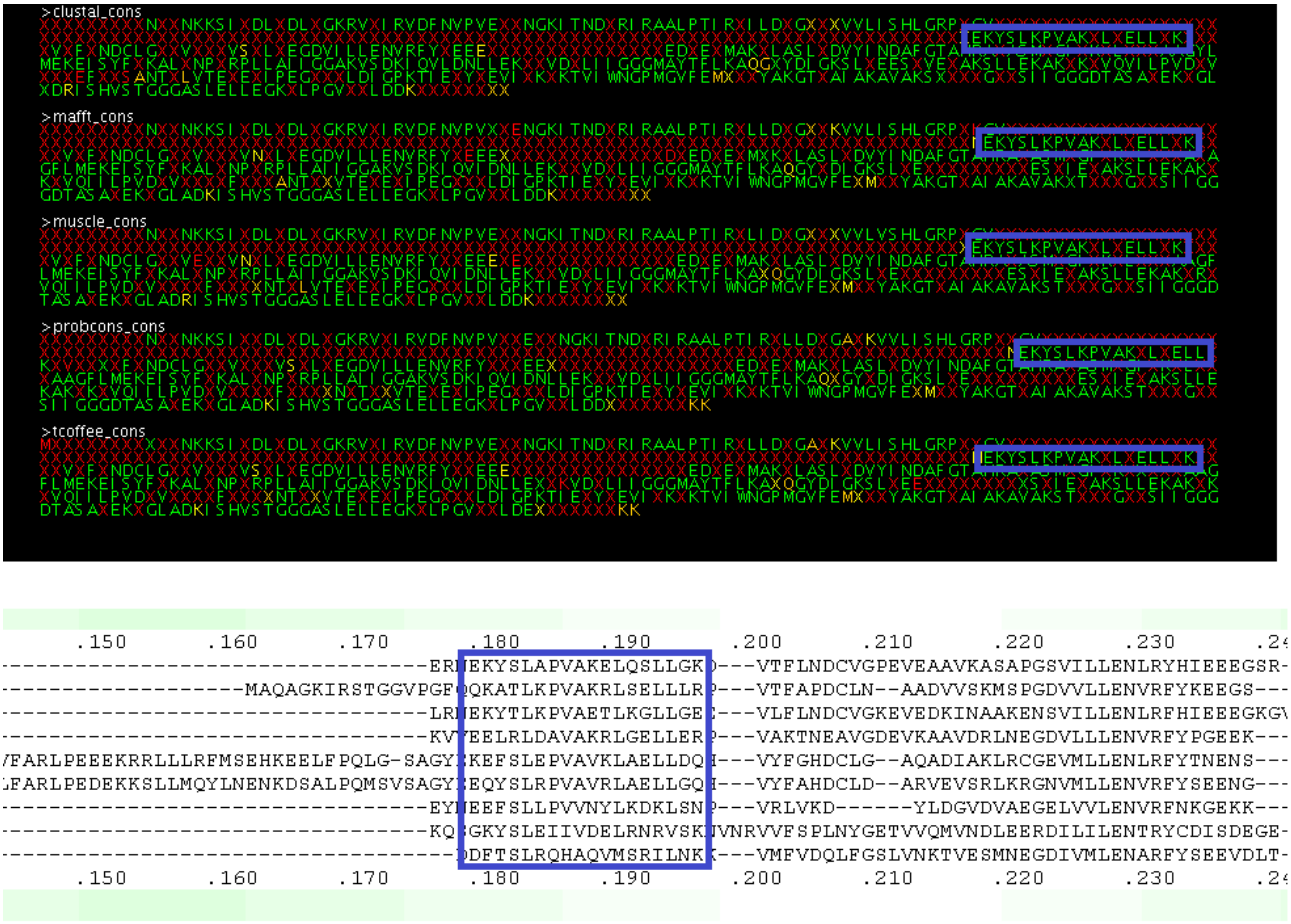


**Ilustración 50: Identificación de fragmentos conservados usando el componente Alignment Comparator del alineamiento de referencia BB50004_1qpg_ref5 de Balibase**

### 7.1.2 Matrix Plotter y GraphViz

Estos dos componentes permiten la detección de secuencias muy cercanas y que pueden llegar a ser problemáticas al momento de generar el alineamiento. Para validar que se encontraran clusters y que se identificaran secuencias similares, se partió de la secuencia keratin, type II cytoskeletal 5 [Homo sapiens] y se buscaron secuencias similares usando BLAST del NCBI. Las secuencias más relevantes de la búsqueda correspondieron a las secuencias con identificación AAF97931.1, AAH24292.1, XP_002823325.1, EAW96635.1 y A5A6M8.1 con scores de 1175, 1172, 1171, 1162 y 870 respectivamente. Adicionalmente se seleccionaron otras secuencias con valores de score menores.

En los resultados obtenidos por MatrixPlotter y GraphViz se observa que se identificaron correctamente las secuencias seleccionadas manualmente por ser las más cercanas a la secuencia de interés y que se identificó el cluster que las contenía. Adicionalmente se identificaron otros grupos de secuencias que también eran cercanos entre ellos.

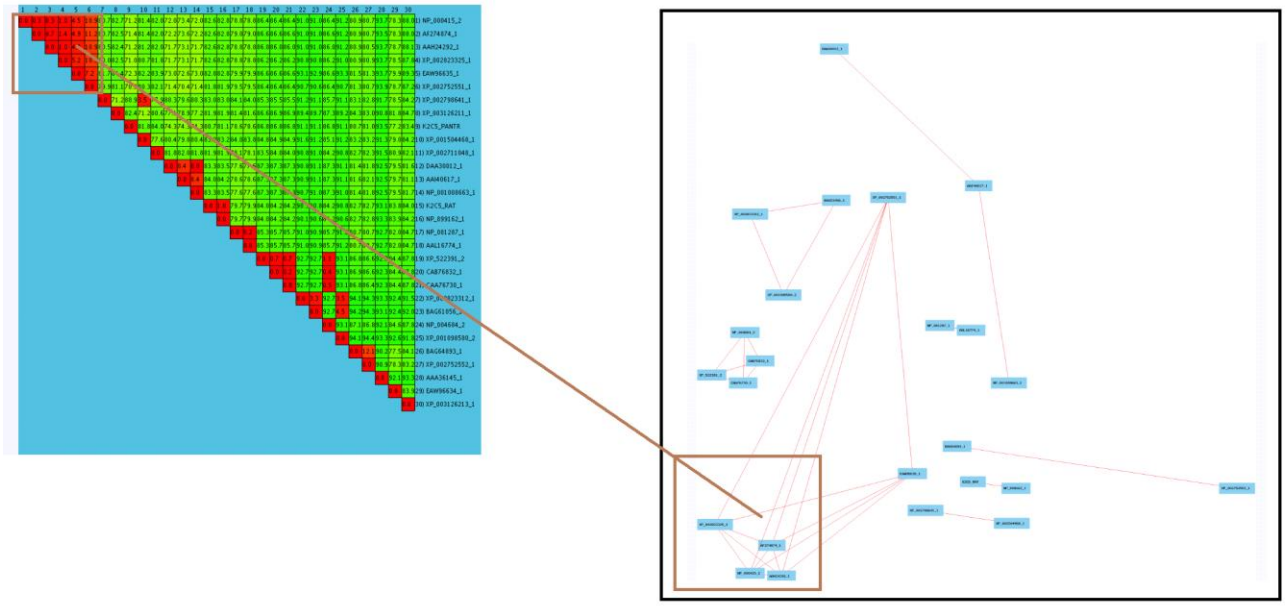

**Ilustración 51: Identificación de secuencias similares y clusters**

El segundo paso a evaluar en la solución propuesta es su interacción con la infraestructura oportunista.
Para esto, se realizar diferentes pruebas en las cuales se corría un número diferente de ejecuciones

## 7.2 Evaluación de desempeño con respecto al número de máquinas

En este escenario se lanzaron 100 ejecuciones simultáneas del workflow MSA con un archivo de 35 secuencias. A lo largo de la prueba se iba cambiando el número de máquinas virtuales y se observó cómo se afectaba el tiempo total para el procesamiento de todas las tareas.
Para la prueba de desempeño, la infraestructura base consistía en 5 máquinas virtuales en el servidor ESX y se iban levantando máquinas virtuales de las salas de cómputo.
Los resultados obtenidos en esta prueba se presentan en la tabla 3 y en la figura 52:

| | Tiempo de Ejecución | | | | | | |
|---|---|---|---|---|---|---|---|
| | 0 pcs | 10 pcs | 20 pcs | 30 pcs | 40 pcs | 50 pcs | 60 pcs |
| Medición 1 | 12424 | 4071 | 2551 | 1995 | 1811 | 1495 | 1311 |
| Medición 2 | 12157 | 4058 | 2570 | 2013 | 1716 | 1485 | 1291 |
| Medición 3 | 12224 | 4023 | 2579 | 2051 | 1840 | 1498 | 1296 |
| Promedio | 12268,33 | 4050,67 | 2566,67 | 2019,67 | 1789,00 | 1492,67 | 1299,33 |
| Desv. Estándar | 138,91 | 24,83 | 14,29 | 28,59 | 64,86 | 6,81 | 10,41 |
| Tiempo Estimado | 10856 | 3925 | 2505 | 1809 | 1416 | 1163 | 987 |
| T. Total / T. Estimado | 1,13 | 1,03 | 1,06 | 1,16 | 1,31 | 1,33 | 1,37 |

**Tabla 3: Tiempos de ejecución al aumentar el número de máquinas virtuales**

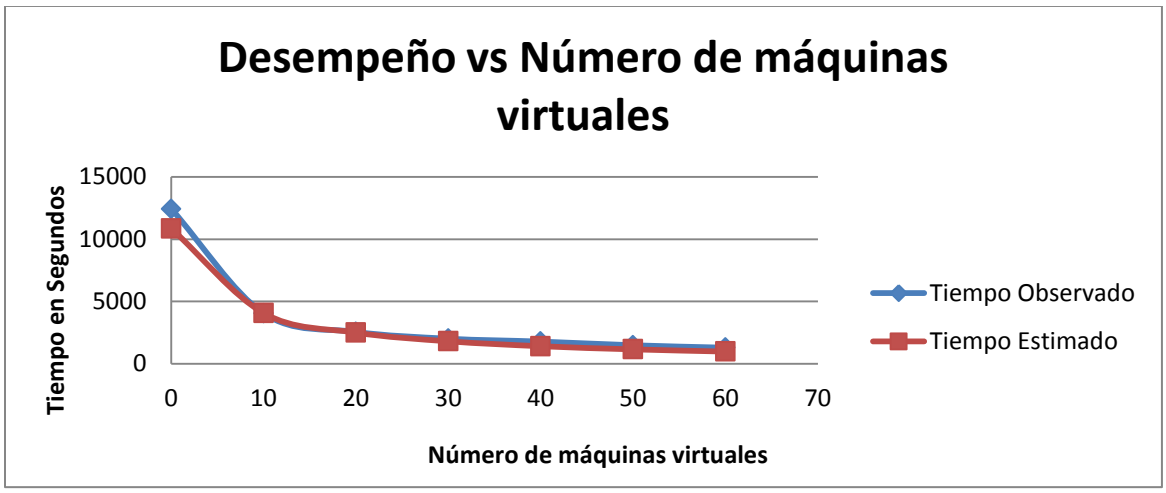


**Ilustración 52: Desempeño Real y Estimado con respecto al número de Máquinas Virtuales**

Los resultados muestran que la infraestructura distribuida de los laboratorios de cómputo permite obtener valores de tiempo muy cercanos a los tiempos estimados si la división de trabajo fuera completamente equitativa y no se incorporaran sobrecargas en el proceso de asignación de trabajos. También se aprecia que a medida que el número de máquinas se incrementa, el desempeño observado se alejada cada vez más del desempeño ideal: Esto se observa en el cociente de T.Total / T. Estimado, el cual vale 1 cuando el tiempo medido es igual al tiempo estimado.

## 7.3 Evaluación de desempeño con respecto a niveles de uso de la máquina física

En esta prueba se buscaba evaluar cómo se afectaba el rendimiento de la ejecución de varios workflows simultáneos bajo diferentes niveles de uso de la infraestructura oportunista. Para esta prueba se lanzaron 20 ejecuciones simultáneas del Workflow MSA con un archivo de 30 secuencias. Se desplegó el cluster virtual con 6 máquinas de la sala Waira2. Se probaron los siguientes niveles de uso: Cuando los computadores no tenían ningún usuario, cuando los computadores tenían cada uno un usuario haciendo tareas típicas de un estudiante (Navegación en internet, visualización de videos por streaming, desarrollo en NetBeans y/o Eclipse, manejo de archivos en Word, Excel, entre otros) y cuando el computador estaba corriendo una tarea intensiva en procesamiento (Se corrió una versión del cálculo de la secuencia de Fibonacci de manera recursiva).

| | Tiempo de Ejecución | | |
|---|---|---|---|
| | Libre | Uso Normal | Tarea Intensiva |
| Medición 1 | 779 | 827 | 1317 |
| Medición 2 | 766 | 801 | 1202 |
| Medición 3 | 783 | 751 | 1242 |
| Promedio | 776,0 | 793,0 | 1253,7 |
| Desviación Estándar | 8,9 | 38,6 | 58,4 |
| CPU promedio utilizado por el usuario | 0% | 4% | 56% |

**Tabla 4: Resultado de pruebas bajo diferentes niveles de uso**

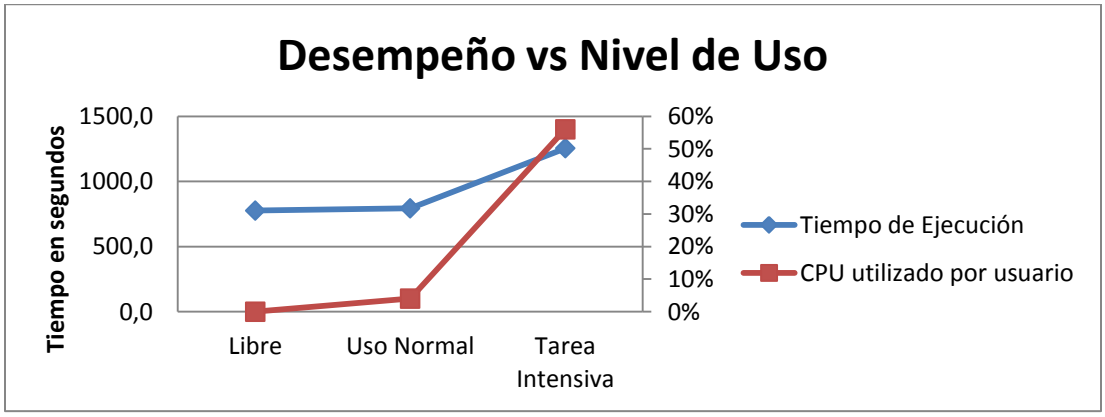


**Tabla 5: Desempeño observado bajo los diferentes niveles de uso**

A partir de los resultados obtenidos se puede concluir el desempeño del cluster virtual se ve directamente relacionado con el nivel de uso de la máquina física. Sin embargo se observa que una sesión normal promedio de un estudiante, no impone un nivel de carga significativo en términos de uso de CPU, y este nivel de uso se ve reflejado en un incremento casi imperceptible del tiempo de ejecución del Workflow MSA. Cuando el nivel de uso de la máquina física se incrementa sustancialmente, por ejemplo a través de una ejecución de un programa intensivo en CPU, el tiempo empleado para completar la tarea si se incrementa de manera apreciable. Afortunadamente, a partir del estudio del comportamiento de la infraestructura oportunista, se observa que la infraestructura por lo general mantiene un nivel de uso no superior al 10% en promedio, lo que garantiza que el cluster virtual va a presentar un buen nivel de desempeño.

## 7.4 Evaluación de desempeño y uso de CPU con respecto a la configuración de las máquinas virtuales

En esta prueba se buscaba evaluar como cambiaba el desempeño en realizar las mismas 100 ejecuciones simultáneas utilizando los 65 pcs (5 dedicados y 60 de las salas de los laboratorios) al ir cambiando el número de cores asignados a las máquinas. Los resultados obtenidos en esta prueba se aprecian en la tabla 6 y en la figura 53.

| | Tiempo De Ejecución | | |
|---|---|---|---|
| | 1 core | 2 cores | 4 cores |
| Medición 1 | 1311 | 748 | 707 |
| Medición 2 | 1291 | 733 | 758 |
| Medición 3 | 1296 | 787 | 731 |
| Promedio | 1299,3 | 756,0 | 732,0 |
| Desv. Estándar | 10,4 | 27,9 | 25,5 |
| Tiempo Estimado | 1299,3 | 649,65 | 324,825 |
| T. Total / T. Estimado | 1,00 | 1,16 | 2,25 |

**Tabla 6: Resultado de prueba variando el número de cores asignados a las MVs.**

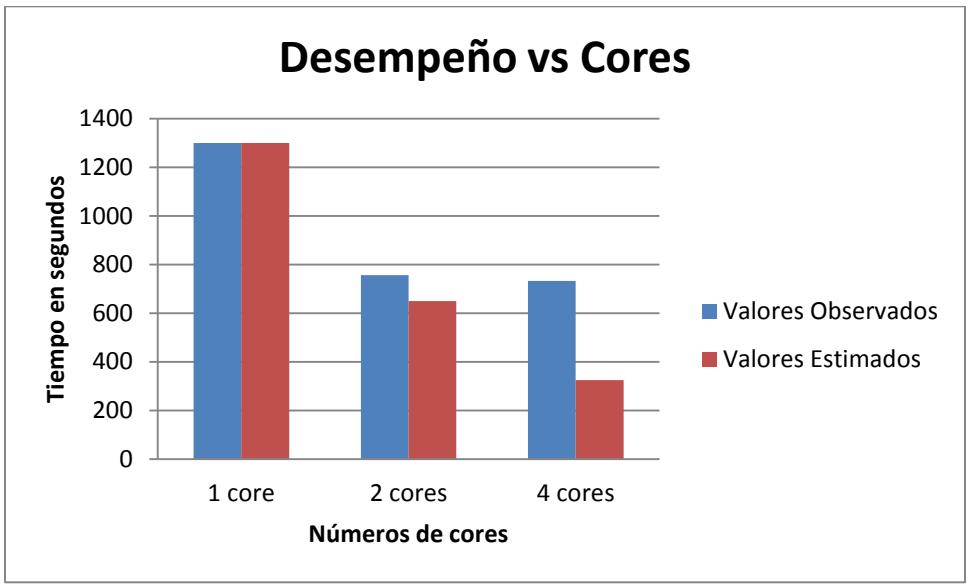


**Ilustración 53: Desempeño Real y Esperado con relación al número de cores**

El tiempo estimado para esta prueba se calcula tomando como base la observación hecha para el escenario de 1 core. Al pasar a 2 cores, se esperaría que el tiempo de ejecución se disminuyera a un 50% y al pasar a 4 cores, se espera que se reduzca a un 25%.

Los resultados muestran que al pasar de 1 a 2 cores, el tiempo de ejecución se redujo a casi la mitad (A un 57%), lo cual está cercano al valor estimado. Al pasar de 2 a 4 cores sin embargo, se obtiene una reducción casi imperceptible en el tiempo de ejecución cuando el valor esperado en la reducción es de un 50%.

La razón principal por la que no se obtiene una disminución significativa en este escenario es que aunque Windows reporta la presencia de 4 threads de ejecución, físicamente sólo cuenta con 2 cores. Para lograr

aprovechar los 4 threads que ofrece el procesador, habría que rediseñar todas las aplicaciones haciendo optimizaciones a bajo nivel. Este tipo de tareas se vuelven imprácticas por la cantidad de programas que se ejecutan en el workflow y porque sería necesario entrar a recompilar incluso la máquina virtual de java.
El porcentaje de uso de CPU registrado por Windows en cada caso fue cercano a 25%,50% y 100% respectivamente. Esta prueba nos permite concluir que una asignación de 2 cores a cada máquina virtual obtiene casi el mismo rendimiento que una asignación de 4 cores y que para Windows, la ejecución de la máquina virtual solo toma un 50% de la CPU.

## 7.5 Evaluación de Desempeño con Respecto al Tamaño del Archivo de Entrada

Es claro que el tamaño de los datos de entrada es un factor importante en el desempeño de cualquier algoritmo o programa. Con esta prueba se buscaba evaluar cómo se afecta el tiempo de ejecución total del workflow MSA e identificar cuáles de las diferentes tareas son las más exigentes en tiempo.
Para la prueba se utilizaron archivos de 10, 30, 50, 70 y 100 secuencias. Con cada archivo se ejecutó el workflow MSA y se midió el tiempo de su ejecución. Cada prueba fue ejecutada 3 veces. En la tabla 7 se presentan los resultados de la prueba.

| | 10 Sec | | 30 Sec | | 50 Sec | | 70 Sec | | 90 Sec | | 100 Sec | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Tarea | Prom. | Desv. | Prom. | Desv. | Prom. | Desv. | Prom. | Desv. | Prom. | Desv. | Prom. | Desv. |
| Copy | 0,3 | 0,6 | 0 | 0,0 | 0 | 0,0 | 0,333 | 0,6 | 0,3 | 0,6 | 0 | 0,0 |
| ClustalW2 | 1,3 | 0,6 | 9,7 | 0,6 | 30,3 | 4,5 | 53,3 | 1,5 | 83,3 | 3,2 | 100,0 | 2,6 |
| MAFFT | 0,3 | 0,6 | 0,3 | 0,6 | 3,0 | 4,4 | 3,7 | 2,1 | 8,7 | 10,7 | 4,3 | 2,1 |
| MUSCLE | 0,7 | 0,6 | 2,3 | 0,6 | 13,0 | 13,0 | 10,0 | 4,4 | 40,3 | 31,8 | 92,7 | 36,3 |
| PROBCONS | 8,0 | 0,0 | 73,7 | 1,5 | 271,7 | 22,5 | 549,3 | 61,0 | 938,3 | 111,1 | 1231,7 | 37,2 |
| T-COFFEE | 10,0 | 1,0 | 99,3 | 1,2 | 403,0 | 3,0 | 1207,0 | 93,8 | 2656,7 | 9,0 | 3976,0 | 185,5 |
| Distmat | 0,3 | 0,6 | 0,0 | 0,0 | 0,7 | 1,2 | 1,7 | 1,5 | 4,7 | 0,6 | 3,7 | 3,2 |
| seqlogoClustal | 3,0 | 1,7 | 2,3 | 0,6 | 8,7 | 6,4 | 4,3 | 0,6 | 12,3 | 10,2 | 18,0 | 20,8 |
| seqlogoMafft | 5,0 | 2,6 | 3,0 | 0,0 | 4,7 | 1,2 | 7,3 | 1,5 | 9,0 | 3,0 | 33,3 | 42,3 |
| seqlogoMuscle | 3,0 | 0,0 | 3,0 | 0,0 | 18,7 | 24,5 | 10,0 | 9,5 | 24,3 | 27,5 | 6,0 | 1,0 |
| seqlogoProbcons | 2,7 | 0,6 | 3,0 | 0,0 | 7,7 | 5,5 | 5,3 | 0,6 | 9,3 | 5,1 | 15,0 | 13,1 |
| seqlogotcoffee | 2,3 | 0,6 | 3,0 | 0,0 | 4,0 | 0,0 | 14,0 | 17,3 | 18,0 | 2,6 | 12,7 | 13,3 |
| matrix_plotter | 4,0 | 5,2 | 1,7 | 0,6 | 9,0 | 11,3 | 4,0 | 1,7 | 49,3 | 17,6 | 7,3 | 4,0 |
| seqgraphbuilder | 0,3 | 0,6 | 0,7 | 0,6 | 0,3 | 0,6 | 1,0 | 1,0 | 7,3 | 6,7 | 9,7 | 16,7 |
| graphviz | 0,7 | 0,6 | 0,7 | 0,6 | 1,0 | 1,0 | 1,0 | 0,0 | 1,0 | 1,0 | 1,0 | 0,0 |
| consClustal | 0,0 | 0,0 | 0,7 | 0,6 | 0,0 | 0,0 | 0,7 | 0,6 | 1,7 | 0,6 | 0,7 | 1,2 |
| consMafft | 0,0 | 0,0 | 1,0 | 1,0 | 0,7 | 1,2 | 1,0 | 1,0 | 0,7 | 0,6 | 0,3 | 0,6 |
| consMuscle | 1,7 | 2,9 | 0,0 | 0,0 | 1,7 | 2,1 | 3,3 | 3,2 | 1,3 | 1,2 | 0,3 | 0,6 |
| consProbCons | 0,3 | 0,6 | 0,0 | 0,0 | 0,0 | 0,0 | 1,3 | 1,5 | 1,3 | 0,6 | 4,0 | 3,6 |
| consTCoffe | 0,0 | 0,0 | 0,3 | 0,6 | 0,7 | 1,2 | 0,7 | 1,2 | 4,0 | 5,3 | 5,0 | 5,3 |
| concat | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,3 | 0,6 | 0,7 | 0,6 |
| t-coffee_1 | 5,0 | 3,5 | 6,3 | 4,9 | 8,0 | 1,0 | 40,3 | 54,3 | 233,7 | 114,9 | 77,7 | 111,1 |
| alignment_comparison | 2,7 | 0,6 | 1,3 | 0,6 | 1,0 | 0,0 | 2,0 | 1,0 | 6,0 | 4,4 | 10,0 | 15,6 |
| results_copy | 0,3 | 0,6 | 0,3 | 0,6 | 0,3 | 0,6 | 1,0 | 0,0 | 0,3 | 0,6 | 1,0 | 1,0 |
| results_copy_png | 0,3 | 0,6 | 0,3 | 0,6 | 0,3 | 0,6 | 1,0 | 0,0 | 0,7 | 0,6 | 1,3 | 1,5 |
| results_delivery | 10,3 | 0,6 | 11,3 | 1,5 | 11,3 | 1,2 | 11,7 | 1,5 | 13,7 | 1,5 | 12,7 | 2,5 |
| Tiempo Total | 62,7 | | 224,3 | | 799,7 | | 1935,3 | | 4126,7 | | 5625,0 | |

**Tabla 7: Resultados de pruebas de desempeño variando el tamaño del archivo de entrada**

La gráfica 54 presenta el porcentaje de tiempo de cada una de las tareas para los archivos de 10 y 100 secuencias. Inicialmente se aprecia que cuando el número de secuencias es poco, el tiempo promedio de la las tareas es balanceado, sin embargo a medida que se incrementan las secuencias a procesar, las tareas de alineamiento con T-Coffee y ProbCons empiezan a tomar mucho más tiempo. Este comportamiento se debe a la complejidad temporal de estos dos algoritmos, los cuales crecen asimptóticamente más rápido que las otras tareas. Esta identificación de tiempos de tareas es un paso fundamental al momento de utilizar la estrategia de despliegue basada en pesos de jobs.

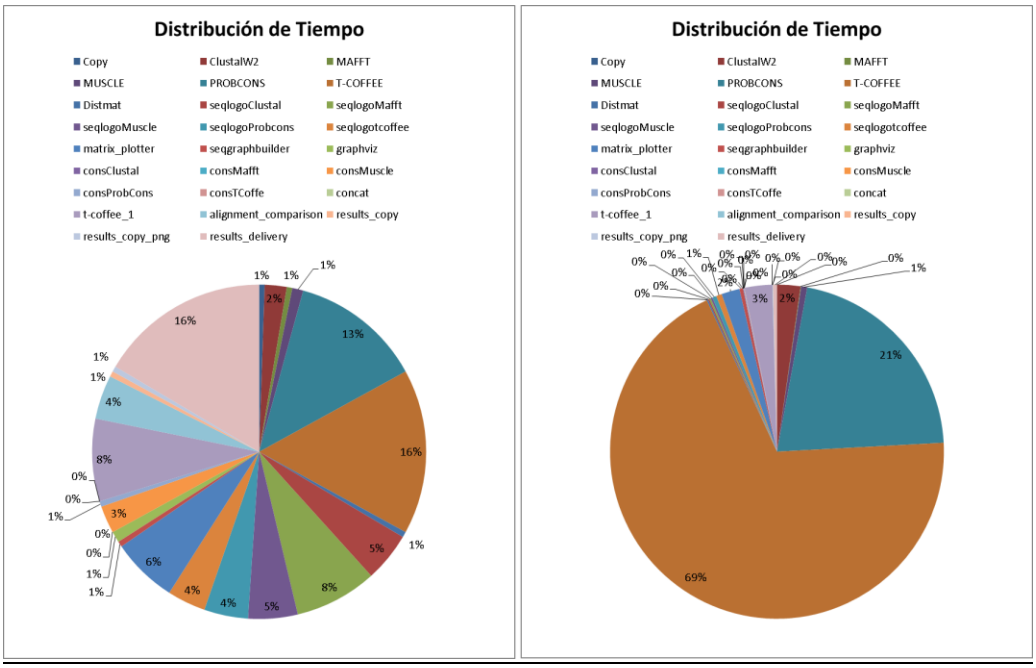


**Ilustración 54: Distribución del Tiempo para 10 y 100 secuencias**

En la gráfica 55 se observa cómo se comportan el tiempo total de ejecución del workflow y tres de las tareas que más tiempo consumen a medida que se incrementa el número delas secuencias a alinear. Se observa como el crecimiento no es lineal, al utilizar una aproximación con mínimos cuadrados con un polinomio de grado 2 se obtiene un valor de $R^2$ de 0.997 lo que indica un muy bajo nivel de error. Esta información nos lleva a concluir que el Workflow MSA no es lo suficientemente escalable en términos del número de secuencias, debido en su mayoría al componente de T-Coffee. Esto nos permite indicar que el workflow no es práctico de usar vía web puesto que para un archivo de 100 secuencias se obtiene un tiempo total de más de una hora lo que no va de acuerdo a la interacción usual con una aplicación web donde los resultados se esperan de manera casi inmediata. Sin embargo, como el Workflow MSA puede ser utilizado a través de la interface de LoniPipeline también, su escalabilidad pierde un poco de importancia puesto que en el cliente gráfico se pueden dejar corriendo Workflows que toman tiempos del orden de días sin ningún problema.

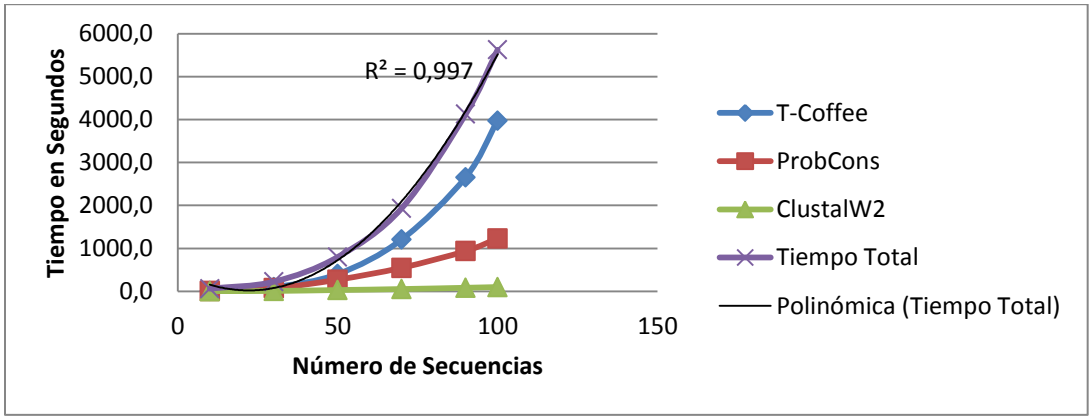


**Ilustración 55: Crecimiento de tiempos con respecto al número de secuencias**

## 7.6 Evaluación de OpportunisticDeployer

Este conjunto de pruebas buscaba evaluar el comportamiento del componente de OpportunisticDeployer y comparar las diferentes estrategias planteadas. Para la ejecución de la prueba se lanzaron 20 ejecuciones simultáneas del Workflow MSA con un archivo de 50 secuencias y se lanzó la ejecución de OpportunisticDeployer. Se buscó ejecutar un número de tareas mediano ya que si se lanzaba un número grande de tareas, todas las diferentes estrategias iban a terminar solicitando el despliegue de todas las máquinas virtuales, por lo que no se iban a poder apreciar las diferencias entre ellas.

Las máquinas virtuales desplegadas contaban con un 1 Gb de RAM y 2 core. En cada prueba se utilizó una estrategia distinta: JobCount, WeightedJobCount, AverageLoad con un valor de carga de 0,4, de 0,5 y de 0,75. Cada dos minutos el componente de monitoreo revisaba el estado del cluster y el componente de estrategia determinaba el número de máquinas virtuales a desplegar.

Para el WeightedJobCount se estableció un conjunto de pesos así: 0.5 para T-Coffee, 0.5 para ProbCons y 0.05 para todos los demás tipos de jobs. Estos números se sacaron a partir de los datos tomados a partir del estudió de la distribución de tiempo total de cada una de las tareas donde se aprecia que a medida que se incrementa el tamaño del archivo de secuencias, las dos tareas empiezan a consumir mayor tiempo de computación, por lo que éstas deberían tener un peso mayor a las otras tareas. Adicionalmente como se configuraron las máquinas con 2 cores de ejecución, entonces 1 máquina virtual desplegada, permite al tiempo la ejecución de una tarea de T-Coffee y de ProbCons, de ahí el valor de 0.5. El valor de 0.05 para las demás tareas viene del hecho de que en promedio 20 tareas (1/0.05) toman el mismo tiempo que una máquina virtual en desplegarse.

Se buscaba identificar qué estrategia ofrecía mejor desempeño en cuanto a tiempo total para terminar la ejecución de todas las tareas y el número de máquinas virtuales totales desplegadas. La figura 56 permite observar los diferentes momentos en el tiempo (Separados 2 minutos) y cuántas máquinas virtuales tenía el cluster levantado en ese momento.

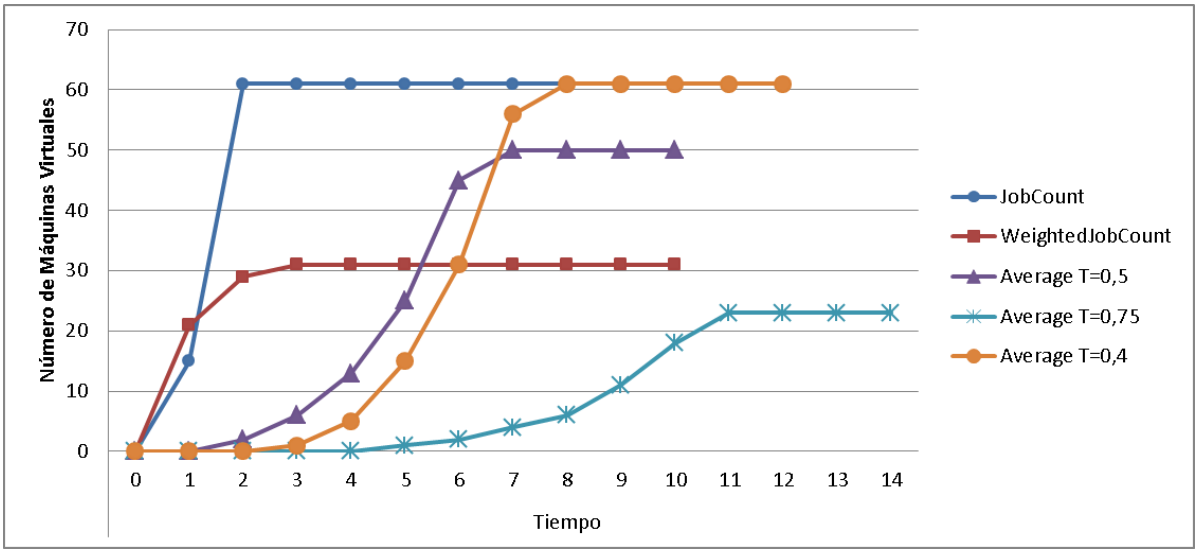


**Ilustración 56: Resultados de OpportunisticDeployer**

| Estrategia | Tiempo total | Número Final de MV | Número Promedio de MV |
|---|---|---|---|
| JobCount | 1254 | 61 | 52,1 |
| WeightedJobCount | 1171 | 31 | 27,09 |
| AverageLoad T=0,4 | 1405 | 61 | 31,76 |
| AverageLoad T=0,5 | 1180 | 50 | 26,45 |
| AverageLoad T=0,75 | 1970 | 23 | 10,59 |

**Tabla 8: Resultados de las pruebas del componente de Opportunistic Deployer**

A partir de los resultados obtenidos se pueden resaltar las siguientes conclusiones:
Las estrategias de WeightedJobCount, Average Load con T=0,5 y JobCount terminaron sus ejecuciones en menor tiempo, las otras dos estrategias demoraron un tiempo considerablemente más largo.
De las 3 estrategias que brindaron el mejor tiempo WeightedJobCount utilizó la menor cantidad de máquinas virtuales, por lo que se clasifica como la mejor estrategia con respecto a tiempo empleado y máquinas virtuales desplegadas.
Se observa también que aunque la estrategia de JobCount rápidamente encendió todas las máquinas virtuales disponibles en ese momento, esto no le garantizó un tiempo de terminación mejor que otras estrategias. Esto se debe a que como no todos los jobs del workflow necesariamente duraban lo mismo, entonces algunos procesadores quedaban en reposo esperando a que otros jobs terminaran y así poder seguir la ejecución del workflow.
También se aprecia que la variación del parámetro de factor de carga en la estrategia AverageLoad juega un papel importante en los resultados. Con un factor de carga más alto, se obtiene un menor número de máquinas a desplegar pero no necesariamente el número suficiente para terminar la tarea con un muy buen tiempo. A medida que se disminuye la carga límite, AverageLoad tiene a prender más máquinas y a terminar las tareas con mayor prontitud.

# 8 Conclusiones

A partir del objetivo general de UnaCloud MSA se puede concluir que se logró desarrollar una plataforma para la generación y análisis de alineamientos múltiples que integrara diferentes programas computacionales con el ánimo de brindar mayor información sobre las secuencias que se poseen. UnaCloud MSA logra brindar su funcionalidad, integrando de manera transparente una infraestructura dedicada con una infraestructura oportunista a través de un modelo de despliegue dinámico que permite un mejor aprovechamiento de los recursos computacionales.

Como conclusiones puntuales se destacan las siguientes:

- La integración de diferentes herramientas computacionales para abordar un problema en específico (En el caso de UnaCloud MSA, el problema de alineamientos múltiples) a través de la integración de un motor de workflows y una infraestructura oportunista, brinda una excelente manera de apoyar diferentes proyectos de investigación, brindando un alto nivel computacional a un costo mínimo.
- La infraestructura oportunista UnaCloud, logra aprovechar los recursos computacionales ociosos de las salas de computadores, que en promedio tienen un porcentaje de uso de CPU menor al 10% en la actualidad.
- Para aplicaciones de carga variable, un modelo de despliegue oportunista permite aprovechar recursos computacionales ociosos y un modelo de despliegue dinámico permite que no se soliciten recursos innecesarios. Así se regula la infraestructura, solicitando máquinas virtuales cuando el nivel de carga de la aplicación lo requiera.
- Al desarrollar un workflow, tener un conocimiento del comportamiento de las diferentes tareas, permite asignar de manera informada los pesos necesarios, con lo cual se logra optimizar el modelo de despliegue dinámico, garantizando que con la menos cantidad de recursos solicitados, se obtengan los mejores resultados. Aunque las otras dos estrategias no requieren tener información adicional sobre el comportamiento del workflow, a partir de los resultados se observa que el mejor comportamiento lo tiene la estrategia de WeightedJobs, por lo que se justifica hacer un estudio previo de los tiempos de las tareas que componen el workflow.
- UnaCloud MSA por sí solo, es un proyecto que brinda valor a la comunidad biológica por permitir la integración de herramientas relevantes para el análisis de alineamientos múltiples. Un aporte adicional del proyecto, es el de presentar el proceso de desarrollo de un workflow para realizar una tarea específica, como integrarlo con una plataforma de grid oportunista como UnaCloud, y ofrecer una interface web con el fin de facilitar la interacción con la aplicación.

# 9 Trabajo Futuro

## 9.1 Workflow MSA

Por un lado el Workflow MSA se puede expandir fácil y naturalmente con otro tipo de tareas que permiten tener un mayor conocimiento de las secuencias iniciales con las que cuenta el investigador y de los alineamientos generados. Diferentes programas de la suite de EMBOSS permiten profundizar en estos análisis detectando diferentes características. También la posibilidad de añadir búsquedas de dominios con PFAM e incorporar un mapeo a las secuencias originales, permitiría obtener información más detallada sobre la funcionalidad de las secuencias a alinear.

## 9.2 Interface Web

Así como se pueden incorporar más tareas de análisis en el workflow MSA, también la aplicación web de UnaCloud MSA permite expansión a través de diferentes opciones que facilitarían el trabajo del investigador. Por un lado la personalización del workflow permitiendo establecer los parámetros de los diferentes jobs, le daría más flexibilidad al biólogo para hacer análisis más alineados con sus intereses. También la incorporación de un sistema de sesiones, permitiría a los usuarios guardar sus resultados y consultarlos en cualquier momento. Este tipo de características no se incluyeron en UnaCloud MSA puesto que correspondían más a una labor de desarrollo de software y no aportaban a la propuesta de investigación aquí presentada.

## 9.3 Mecanismo de Despliegue Dinámico

Opportunistic Deployer es el primer esfuerzo por incorporar a UnaCloud un mecanismo elástico de despliegue de máquinas virtuales. Sería interesante un trabajo más profundo sobre la selección del número de máquinas a levantar óptimo. Este problema se puede ver como uno de optimización objetivo donde se desea minimizar el número de máquinas virtuales a desplegar, minimizar el tiempo de ejecución de las tareas que hay pendientes y a su vez maximizar el uso de los recursos de las máquinas que se despliegan. Adicionalmente sería interesante proponer un modelo predictivo para estimar la cantidad de jobs que se van a tener en cierto momento para anticiparse a estos eventos y desplegar la infraestructura de tal manera que se garantice siempre un alto nivel de rendimiento de la aplicación.

## 9.4 Análisis de disponibilidad de la infraestructura

Si bien se tomaron datos de la disponibilidad de los computadores durante un mes de trabajo promedio. Este modelo no necesariamente refleja la información más reciente que se tiene sobre el estado de los computadores de los laboratorios. Un sistema en el que la base de datos sea actualizada constantemente y que las decisiones tomadas por UnaCloud sean tomadas a partir de la nueva información que se va registrando, permitiría incorporar al sistema no sólo datos históricos sino datos en tiempo real de las máquinas lo que podría mejorar el nivel de disponibilidad general de UnaCloud.

## 9.5 UnaCloud

Al momento de realizar las diferentes pruebas, se detectaron algunos problemas o aspectos a mejorar en UnaCloud. Por un lado, no se contaba con una manera amigable de monitorear la infraestructura, y por lo tanto no era fácil determinar si en una máquina física se encontraba ejecutándose o no una máquina virtual y mucho menos determinar qué máquina se estaba ejecutando. Otro aspecto a mejorar en UnaCloud es incorporar la posibilidad de levantar más de una máquina virtual en una máquina física. Esto permitiría un mejor aprovechamiento de los recursos ya que una máquina virtual desplegada no siempre está ejecutando trabajo.

# 10 Referencias


[Alg01] Hans-Joachim Böckenhauer and Dirk Bongartz. Algorithmic Aspects of Bioinformatics. Publicado por Springer 2007.

[Alt01] B. Morgenstern, S. Goel, A. Sczyrba, A. Dress. AltAVisT: A WWW tool for comparison of alternative multiple alignments, 2003. Volumen 19, páginas 425-426.

[Alt02] AltaVist: Alternative Alignment Visualization Tool. En línea: http://bibiserv.techfak.uni-bielefeld.de/altavist/

[Ama01] Amazon. Amazon Elastic Compute Cloud (Amazon EC2). En línea: http://aws.amazon.com/ec2/

[Bac01] A. González, H. Castro, M. Villamizar, N. Cuervo, G. Lozano, S. Orduz, S. Restrepo. Mesoscale Modeling of the Bacillus thuringiensis Sporulation Network Based on Stochastic Kinetics and Its Application for in Silico Scale-down. High Perfomance Computational Systems Biology, 2009.

[Bal01] Julie D. Thompson, Fréderic Plewniak and Olivier Poch. BAliBASE: A benchmark alignment databse for the evaluation of multiple alignment programs, 1999. Bioinformatic Oxford Journals, volumen 15, número 1, páginas 87-88.

[Bal02] Anne Bahr, Julie D Thompson, J C Thierry, Olivier Poch. BAliBASE (Benchmark Alignment dataBASE): enhancements for repeats, transmembrane sequences and circular permutations, 2001. Nucleic Acids Research, volumen 29, número 1, páginas 323, 326.

[Bal03] Julie D Thompson, Patrice Koehl, Raymond Ripp, Olivier Poch. BAliBASE 3.0: latest developments of the multiple sequence alignment benchmark, 2005. Proteins, volumen 61, número 1, páginas 127, 136.

[Bio01] National Center for Biotechnology Information. Just the Facts: A Basic Introduction to the Science Underlying NCBI Resources. En línea: http://www.ncbi.nlm.nih.gov/About/primer/bioinformatics.html

[Bio02] Mario Villamizar, Harold Castro, David Mendez, Silvia Restrepo, Luis Rodriguez. Bio-UnaGrid: Easing Bioinformatics Workflow Execution Using LONI Pipeline and a Virtual Desktop Grid, 2011. BIOTECHNO 2011, The Third International Conference on Bioinformatics, Biocomputational Systems and Biotechnologies.

[Bla01] National Center for BiotechnologyInformation NCBI. *BLAST Homepage*. En línea: http://blast.ncbi.nlm.nih.gov/Blast.cgi.

[Blo01] Steven Henikoff and Jorja G. Henikoff. Amino acid substitution matrices from protein blocks. Proceedings of the National Academy of Sciences of the United States of America, 1992. Volumen 89, páginas 10915-10919.

[Cam01] Harold Castro and Danilo Pérez. Campus-grid UniAndes. Conferencia Latinoamericana de Computación de Alto Rendimiento, CLCAR. 2007, páginas 18-24.

[Clo01] Paolo Di Timaso and Miquel Orobitg and Fernando Guiradi and Fernando Cores and Toni Espinosa and Cedric Notredame. Cloud-Coffee: Implementation of a parallel consistency-based multiple algorithm in the T-Coffee package and its benchmarking on the Amazon Elastic-Cloud, 2010. Bioinformatics, Oxford University Press, volumen 26, número 15, 1903-1904.

[Clo02] Amazon, Amazon CloudWatch. En línea:
http://docs.amazonwebservices.com/AmazonCloudWatch/latest/DeveloperGuide/index.html?Welcome.html#

[Clu01] Thompson, J. and Higgins, D. and Gibson, T. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice, 1994. Nucleic Acids Research, volumen 22, 4673-4680.

[Clu02] Desmond G. Higgins and Paul M. Sharp. CLUSTAL: a package for performing multiple sequence alignment on a microcomputer, 1988. Trends in Biochemical Sciences, volumen 23, número 10, páginas 403, 405.

[Clu03] M.A. Larkin, G. Blackshields, N.P. Brown, R. Chenna, P.A. McGettigan, H. McWilliam, F. Valentin, I.M. Wallace, A. Wilm, R. Lopez, J.D. Thompson, T.J. Gibson and D.G. Higgins. Clustal W and Clustal X version 2.0, 2007. Bioinformatics, volumen 23, número 21, páginas 2947-2948.

[Clu04] Kuo-Bin Li. ClustalW-MPI: ClustalW analysis using distributed and parallel computing, 2003. Bioinformatics, volumen 19, número 12, páginas 1585-1586.

[Con01] Condor High Throughput Computing. En línea: http://www.cs.wisc.edu/condor/
[Con02] EMBOSS: cons. En línea: http://emboss.sourceforge.net/apps/cvs/emboss/apps/cons.html
[Con03] Sim, Jeong Seop and Park, Kunsoo. The consensus string problem for a metric is NP-complete, 2003. Journal of Discrete Algorithms, volumen 1, número 1, páginas 111 – 117.
[Deb01] Debian. Debian 5. En línea: www.debian.org
[Dis01] EMBOSS: distmat. En línea: http://emboss.sourceforge.net/apps/cvs/emboss/apps/distmat.html.
[Drm01] Open Grid Forum. DRMAA: Distributed Resource Management Application API. En línea: http://www.drmaa.org/
[Emb01] EMBOSS: European Molecular Biology Open Software Suite. En línea: http://emboss.sourceforge.net/
[Fpg01] Tim Oliver, Bertil Schmidt, Darran Nathan, Ralf Clemens and Douglas Maskell. Using reconfigurable hardware to accelerate multiple sequence alignment with ClustalW, 2005. Bioinformatics, volumen 21, número 16, páginas 3431,3432.
[Fun01] J. Pevsner. Bioinformatics and Functional Genomics. Publicado por Wiley-Liss, 2003.
[Gal01] Goecks, J.; Nekrutenko, A.; Taylor, J.; Galaxy Team, T. "Galaxy: A comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences". Genome Biology, 2010.
[Gen01] National Center for Biotechnology Information. Growth of Genbank. En línea. http://www.ncbi.nlm.nih.gov/genbank/genbankstats.html
[Gen02] Biomatters Ltd. Geneious. En línea: http://www.geneious.com/default,213,Alignment.sm
[Gla01] Oracle. GlassFish Application Server. En línea: http://glassfish.java.net/
[Gli01] EGEE Project. gLite: Lightweight Middleware for Grid Computing. En línea: http://glite.cern.ch/
[Gls01] The OpenGL Shading Language. En línea: http://www.opengl.org/registry/doc/GLSLangSpec.4.10.6.clean.pdf.
[Got01] Osamu Gotoh. An improved algorithm for matching biological sequences. Journal of Molecular Biology, 1982. Volumen 162, número 3, páginas 705-708.
[Gpu01] Liu, Weiguo and Schmidt, Bertil and Voss, Gerrit and Müller-Wittig, Wolfgang. GPU-ClustalW: Using Graphics Hardware to Accelerate Multiple Sequence Alignment, 2006. High Performance Computing - HiPC 2006, páginas 363-374.
[Gra01] Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, Kiem-Phong Vo. A Technique for Drawing Directed Graphs. AT&T Bell Laboratories. En línea: http://www.graphviz.org/Documentation/TSE93.pdf
[Gra02] Graphviz: Graph Visualization Software. En línea: http://www.graphviz.org/
[Gri01] Talbi, El-Ghazabi and Zomaya, Albert. Grid Computing for Bioninformatics and Computational Biology. Publicado por Wiley, 2007.
[Gri02] Omii-Uk. GridSAM. En línea: http://www.omii.ac.uk/wiki/GridSAM
[Hir01] D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. Communications of the ACM, 1975. Volumen 18.
[Hom01] Mizuguchi K, Deane CM, Blundell TL, Overington JP. HOMSTRAD (Homologous Structure Alignment Database): A database of protein structure alignments for homologous families. Protein Science volumen 7, páginas 2469-2471.
[Jal01] Waterhouse, A.M., Procter, J.B., Martin, D.M.A, Clamp, M., Barton, G.J. Jalview version 2: A Multiple Sequence Alignment and Analysis Workbench, 2009. Bioinformatics .
[Jav01] Oracle. Java. En línea: http://www.java.com/en/
[Jga01] A. Bernal, H. Castro, A. Medaglia, J.L. Walteros. JG2A: A Grid-Enabled Object-Oriented Framework for Developing Genetic Algorithms. IEEE Systems and Information Engineering Design Symposium, SIEDS 2009.
[Jgd01] Loni Pipeline. JGDI: Java grid engine Database Interface. En línea:
http://pipeline.loni.ucla.edu/support/server-guide/pipeline-grid-plugin-api-developers-guide/
[Jsf01] Oracle. Java Server Faces Technology. En línea: http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html

[Kep01] The Kepler Project. En línea: https://kepler-project.org/
[Kim01] Motoo Kimura. The Neutral Theory of Molecular Evolution, 1983. Cambridge University Press
[Lev01] Vladimir Iosifovich Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady, 1966. Volumen 10, páginas 707,710.
[Lon01] LONI Pipeline. En línea: http://pipeline.loni.ucla.edu/
[Maf01] Katoh, Asimenos, Toh. Multiple Alignment of DNA Sequences with MAFFT, 2009. Methods in Molecular Biology, volumen 537, páginas 39-64.
[Maf02] Mafft: Multiple Alignment using Fast Fourier Transform. En línea. http://mafft.cbrc.jp/alignment/software/
[Mat01] MathWorks. MATLAB - The Language Of Technical Computing. En línea: http://www.mathworks.com/products/matlab/
[Mil01] Eugene W. Myers and Webb Miller. Optimal alignments in linear space. Computer Applications in the Biosciences, 1988. Volumen 4, páginas 11-17.
[Moj01] Nihilogic Labs. MojoZoom: Free JavaSCript Image Zoom. En línea: http://www.nihilogic.dk/labs/mojozoom/
[Msa01] Liu, Y., Schmidt B. and Maskell, D. MSA-CUDA: Mutiple Sequence Alignment on Graphics Processing Units with CUDA, 2009. 20th IEEE International Conference on Application-specific Systems, Architectures and Processors.
[Mul01] Robert C. Edgar, and Serafim Batzoglou. Multiple sequence alignment. Current Opinion in Structural Biology, 2006. Volumen 16, número 3, páginas 368, 373.
[Mus01] Robert, C. Edgar. MUSCLE: Multiple Sequence Alignment with High Accuracy and High Throughput, 2004. Nucleic Acids Research, volumen 32, número 5, 1792-1797.
[Mus02] Xi Deng and Eric Li and Jiulong Shan and Wenguang Chen. Parallel implementation and performance characterization of MUSCLE, 2006. Parallel and Distributed Processing Symposium, página 7.
[Nas01] Gibson, G. A. and Meter, R. Network Attached Storage Architecture, 2000. ACM Communications, volumen 43, páginas 37-45.
[Nea01] Stephen C. North. Drawing graphs with NEATO, 2004. Publicado por Springer, Volumen 22, páginas 1-11.
[Ned01] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology, 1970. Volumen 48, número 3, páginas 443 – 453.
[Ope01] Xinmin Tian, Aart Bik, Milind Girkar, Paul Grey, Hideki Saito, Ernesto Su. Intel OpenMP C++/Fortran Compiler for Hyper-Threading Technology: Implementation and Performance. En línea: http://download.intel.com/technology/itj/2002/volume06issue01/art04_fortrancompiler/vol6iss1_art04.pdf
[Ope02] Intel Corporation. Getting Started with OpenMP. En línea: http://software.intel.com/en-us/articles/getting-started-with-openmp/
[Pal01] A. Bernal, H. Castro. pALS: An object-oriented framework for developing parallel cooperative metaheuristics. Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium, 2010, páginas 1-8.
[Pam01] M. O. Dayhoff , R. M. Schwartz and B. C. Orcutt. A model of evolutionary change in proteins. Bioinformatic Oxford Journals, 1978. Volumen 5, páginas 345, 351.
[Par01] Jaroslaw Zola, Xiao Yang, Adrian Rospondek and Strivas Aluru. Parallel-TCoffee: A Parallel Multiple Sequence Aligner, 2004. Genome Research, volumen 14, número 6, páginas 1188-1190.
[Par02] Aluru HPC Group. Parallel T-Coffee. En línea: http://aluru-sun.ece.iastate.edu/doku.php?id=parallel-tcoffee
[Pbs01] Portable Batch System. En línea: http://hpc.sissa.it/pbs/pbs.html
[Phy01] Phylip: PHYLogeny Inference Package. En línea: http://evolution.genetics.washington.edu/phylip.html

[Pla01] Plattform Computing. Plattform LSF. En línea: http://www.platform.com/workload-management/high-performance-computing
[Pre01] Edgar, Robert C. Prefab: Protein Reference Alignment Benchmark. En línea: http://www.drive5.com/muscle/prefab.htm
[Pro01] ProbCons Probabilistic Consistency-based Multiple Alignment of Amino Acid Sequences. En línea: http://probcons.stanford.edu/index.html
[Pro02] Do, C.B., Mahabhashyam, M.S.P., Brudno, M., and Batzoglou, S.. PROBCONS: Probabilistic Consistency-based Multiple Sequence Alignment, 2005. Genome Research , volumen 15, páginas 330-340.
[Pto01] Christopher Brooks, Edward A. Lee. Ptolemy II - Heterogeneous Concurrent Modeling and Design in Java, 2010. 2010 Berkeley EECS Annual Research Symposium (BEARS).
[R01] The R project for statistical computing. En línea: http://www.r-project.org/
[Ric01] JBoss. Richfaces Project. En línea: http://www.jboss.org/richfaces
[Sab01] Van Walle I, Lasters I, Wyns L. SABmark: A benchmark for sequence alignment that covers the entire known fold space, 2005. Bioinformatics volumen 21 páginas 1267-1268.
[Ser01] Luis Cabellos, Isabel Campos, Enol Fernandez-del-Castillo, Michal Owsiak, Bartek Palak, Marcin Plociennik. Scientific workflow orchestration interoperating HTC and HPC resources, 2011. Computer Physics Communications, volumen 182, número 4, páginas 890-897.
[Sge01] Oracle. SunGrid Engine. En línea: http://wikis.sun.com/display/GridEngine/Home
[Sgi01] Mikhailov,D., Cofer,H. and Gomperts,R. Performance optimization of Clustal W: parallel Clustal W, HT Clustal, and MULTICLUSTAL, 2001. White papers, Silicon Graphics CA.
[Sig01] Hyperic Inc. SIGAR: System Information Gatherer and Reporter. En línea: http://www.hyperic.com/products/sigar
[Smi01] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. Journal of Molecular Biology, 1981.Volumen 147, número 3, páginas 195-197.
[Sql01] Microsoft Corporation. Microsoft SQL Server. En línea. http://www.microsoft.com/sqlserver/en/us/default.aspx
[Sum01] Iaas Elian. Settling the Intractability of Multiple Alignment, 2006. Journal of Computational Biology, volumen 13, número 7, páginas 1323, 1339.
[Tav01] MyGrid. Taverna Workflow Management System. En línea: http://www.taverna.org.uk/.
[Tco01] Cédric Notredame, Desmond G. Higgins and Jaap Heringa. T-coffee: a novel method for fast and accurate multiple sequence alignment, 2000. Journal of Molecular Biology, volumen 302, páginas 205 – 217.
[Uge01] Unipro UGENE: Integrated Bioinformatics Tools. En línea: http://ugene.unipro.ru/
[Una01] Eduardo Rosales, Mario Villamizar and Harold Castro. UnaCloud: A Desktop Grid and Cloud Computing Solution, 2011. The Second International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering (PARENG2011).
[Una02] Castro, H., Rosales, E., Villamizar, M. and Miller, A. UnaGrid - On Demand Opportunistic Desktop Grid, 2010. 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing.
[Uni01] Jülich Supercomputing Centre. UNICORE: Uniform Inteferface to Computing Resources. En línea: http://unicore.eu/
[Vmw01] VMWare Inc. VMWare Workstation. http://www.vmware.com/products/workstation/
[Vmw02] VMWare Inc. VMware ESX. En línea: http://www.vmware.com/products/vsphere/esxi-and-esx/index.html
[Web01] Gavin Crooks, Gary Hon, John-Marc Chandonia and Steven E. Braner. WebLogo: A Sequence Logo Generator, 2004. Genome Research. Volumen 14, número 6, páginas 1188-1190.
[Web02] WebLogo 3. En línea: http://weblogo.threeplusone.com/
[Wps01] Mariana B Monteiro, Manuela E Pintado, Francisco X Malcata, Conrad Bessant, Patrícia R Moreira. Development of a Workflow for Protein Sequence Analysis Based on the Taverna Workbench Software, 2009. Distributed Computing Artificial Intelligence Bioinformatics Soft Computing and Ambient Assisted Living (2009)

[Wps02] MyExperiments. Workflow Entry: Workflow for Protein Sequence Analysis. En línea: http://www.myexperiment.org/workflows/124.html