SEQUENCE ALIGNMENT USING PARALLEL PROCESSING TECHNIQUES DEVELOPED ON A CLUSTER OF COMPUTERS

Wilson Castaño Galviz[1]
Universidad Pontificia Bolivariana Seccional Bucaramanga
wcastano@upbbga.edu.co
Lenin Javier Serrano Gil[2]
Universidad Pontificia Bolivariana Seccional Bucaramanga
lenin.serrano@correo.upbbga.edu.co
Miguel Mateus Marín[3]
Universidad Pontificia Bolivariana Seccional Bucaramanga
miguel.mateus@correo.upbbga.edu.co

Abstract — This paper presents the development of a computer algorithm for sequence alignment, implemented in a computer cluster at the Universidad Pontificia Bolivariana Bucaramanga branch using Linux operating system and OSCAR configuration tool for deploying applications in parallel, also shows the performance clusters with different sizes of files to compare, that using the cluster with different amounts of processing nodes.

Palabras clave — Cluster de Computadores, OSCAR, MPI, alineamientos.

I. INTRODUCCIÓN

Un área de aplicación de la Bioinformática[1] que actualmente se encuentra en auge es el alineamiento de secuencias (aplicados a genomas) [2], que permite detectar los niveles de similitud entre las diferentes genomas de distintas especies, permitiendo detectar los cambios entre los organismos vivos de nuestro planeta.

En la actualidad existen herramientas[3] para el alineamiento de secuencias que se encuentran disponibles en Internet y permiten a las personas enviar los datos correspondientes a procesar y esperar por un resultado, dichas herramientas realizan los procesos de forma secuencial las cuales tardan en realizarse el cálculo ya que equipos de altas prestaciones no son posible de tener en disponibilidad por el alto costo que esto conlleva.

Con el avance de la tecnología se ha llevado al uso de sistemas computacionales que mejoren el rendimiento de los procesos, por esto los computadores necesarios para desarrollo computacional de alto consumo de horas máquinas tienen un costo muy elevado lo que hace difícil el acceso a las personas que desean realizar investigación que requiera alta prestación computacional, por este motivo nace el procesamiento en paralelo usando cluster de computadores (aprovechando los computadores que permanecen en tiempo ocioso) para realizar en conjunto el cálculo de alto rendimiento que sea necesario.

Una de las áreas de la Ingeniería Informática es el desarrollo de aplicaciones que sean además de ser llevadas a cabo con toda la Ingeniería del Software pertinente, ser aplicables con la funcionalidad que se espera de dicho desarrollo, por este y por los motivos definidos anteriormente se considera necesario llevar a cabo aplicaciones que sean útiles al avance de la ciencia y que pueda ser usada por los miembros de la comunidad tanto académica como científica, que para este caso será la Bioinformática.

[1] Wilson Castaño Galviz, Ingeniero de Sistemas, Especialista en Docencia Universitaria y Magíster en Informática, actualmente labora como Docente e Investigador de tiempo completo en la Universidad Pontificia Bolivariana seccional Bucaramanga Colombia.

[2] Lenin Javier Serrano Gil, Tecnólogo en Electrónica, Estudiante de 10 semestre de Ingeniería Informática en la Universidad Pontificia Bolivariana seccional Bucaramanga Colombia.

[3] Miguel Mateus Marín, Estudiante de 5 semestre del Ingeniería Informática en la Universidad Pontificia Bolivariana seccional Bucaramanga Colombia.

Por esto se plantea, ¿es posible construir una aplicación de alto rendimiento para el proceso de alineamiento de secuencias que pueda realizar alineamientos en menos tiempo de lo que lo hace un computador?.

II. SISTEMAS DE CLUSTER DE COMPUTADORES

El término cluster de computadores[4,5] se definió al sistema de integración de varias CPU (Board, Memoria, Disco y procesador) interconectadas a través de una red de datos que permite la comunicación y que utilizando configuración de paso de mensajes a través de los nodos, es posible que todos (los nodos) trabajen como un sistema de multiprocesador con memoria distribuida.

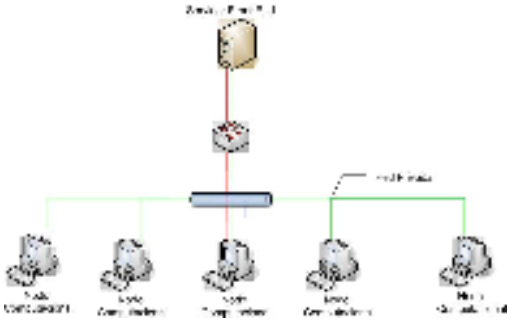

Figura 1. Modelo de un sistema de cluster de computadores.

Para el trabajo que se llevó a cabo se utilizó un cluster de 8 Nodos, están instalados con sistema operativo Linux Fedora en el equipo Front-end y para colocarlos a funcionar como cluster se utilizó OSCAR[6] (Open Source Cluster Application resource).

El cluster funciona de la siguiente manera:

1. Se instaló Linux Fedora[7] en el equipo principal que se llama Front-end
2. Se configuró OSCAR en el equipo Front-end y se inició el proceso de reconocimiento de cada uno de los nodos, para esto se utilizó la interconexión a través de PXE de cada nodo, con el cual esperaba un servidor DHCP para que cada equipo tomara su configuración desde el Front-end
3. Para cada equipo se asignó la IP con la respectiva MAC , por esta razón cada nodo queda completamente ligado al sistema de cluster para que en las posteriores "corridas" del código, cada máquina (pc) fuese reconocida como un elemento más del cluster.

III. ANÁLISIS DEL CÓDIGO DESARROLLADO

El nombre del archivo desarrollado se llamó EqualParallel_MPI.py, al cual se le hace una breve descripción de su funcionalidad.

EqualParallel es un algoritmo dedicado a encontrar las similitudes entre dos cadenas de caracteres basado en el paradigma de procesos en paralelo[8] para optimizar la búsqueda de coincidencias. Desarrollado en Python 2.4con uso de bibliotecas MPI[9].

Para analizar las coincidencias definidas por la igualdad de caracteres entre las cadenas, EqualParallel recibe dos cadenas de caracteres denominadas Source y Target bajo la restricción de longitud LEN(Source) >= LEN(Target) para conformar una matriz, Ejemplo:

Si Source = "atccatattgg" y Target = "gataca" se obtiene:

	A	T	C	C	A	T	A	T	T	G	G
G											
A											
T											
A											
C											
A											

Figura 2. Inicio del Análisis de comparación de las secuencias

Para procesar las coincidencias sobre la matriz se realiza un barrido diagonal como se referencia en la figura 3.

	A	T	C	C	A	T	A	T	T	G	G
G	C	C	C	C	C	C	U	U	U	U	U
A	B	C	C	C	C	C	C	U	U	U	U
T	B	B	C	C	C	C	C	C	U	U	U
A	B	B	B	C	C	C	C	C	C	U	U
C	B	B	B	B	C	C	C	C	C	C	U
A	B	B	B	B	B	C	C	C	C	C	C

Figura 3. Análisis de comparación de las secuencias con barrido diagonal

C = Diagonal Central.
U = Diagonal Superior.
B = Diagonal Inferior.

Definido para cada barrido diagonal el término de “recorrido”. Los recorridos se caracterizan por el tipo de diagonal, es decir, existen para una matriz 3 tipos: recorridos sobre diagonales centrales, recorridos sobre diagonales superiores y recorridos sobre diagonales inferiores. A partir de los recorridos EqualParallel asigna a cada procesador una tarea.

Cada tarea esta definida entonces por la función de identificación entre caracteres sobre el recorrido fijado. La función se encarga de identificar las igualdades agregando un peso valorado en 1 que a su vez es sumado al peso de acarreo en el recorrido. Con un valor neutro cero para la NO igualdad al finalizar el recorrido el peso total indica la cantidad de coincidencias. Para el ejemplo antes descrito el resultado se visualiza en la figura 4.

	A	T	C	C	A	T	A	T	T	G	G
G	0	0	0	0	0	0	0	0	0	1	1
A	1	0	0	0	1	0	1	0	0	0	1
T	0	2	0	0	0	2	0	2	1	0	0
A	1	0	2	0	1	0	3	0	2	1	0
C	0	1	1	3	0	1	0	3	0	2	1
A	1	0	1	1	4	0	2	0	3	0	2

Figura 4. Análisis de comparación de las secuencias con barrido diagonal y valores encontrados para lograr la mayor coincidencia

Los resultados al proceso se reflejan en la última fila para recorridos centrales e inferiores y en la última columna para recorridos superiores.

A	1	0	1	1	4	0	2	0	3	0	2

G
1
1
0
0
1
2

Figura 5. Análisis de comparación con resultados de recorridos

Los datos obtenidos sufren un proceso de ordenamiento que retorna una lista ordenada de mayor a menor de los pesos para cada recorrido, en consecuencia se identifica la mayor coincidencia para una posición de Target vs Source. Retomando el ejemplo para el peso de mayor valor 4 el alineamiento esta dado en la siguiente secuencia:

```
A T C C A T A T T G G
| |   | |
G A T A C A
```

Figura 6. Alineamientos por coincidencias en el programa desarrollado

Paralelización.

EqualParallel entrega a cada procesador existente los recorridos a ejecutar. La distribución de carga responde al tipo de recorrido, en caso que sea central se distribuye en cada procesador por igual. Para el caso de los recorridos superiores e inferiores a cada procesador se le asigna los recorridos de forma tal que las longitudes sean equivalentes. Para la matriz de ejemplo se describe el proceso en la figura 7, 8 y 9.

- Para un total de 3 procesadores recorridos centrales.

P		0	1	2	2	1	0				
		A	T	C	C	A	T	A	T	T	G	G
	G	0	0	0	0	0	0					
	A		0	0	0	1	0	1				
	T			0	0	0	2	0	2			
	A				0	1	0	3	0	2		
	C					0	1	0	3	0	2	
	A						0	2	0	3	0	2

Figura 7. Análisis de recorridos en paralelo usando tres procesadores para recorrido central

Procesador 0 = 6 + 6 = 12.
Procesador 1 = 6 + 6 = 12.
Procesador 2 = 6 + 6 = 12.

Carga total del proceso = 36.

Procesador 0 = (12/36) * 100% = 33.33%.
Procesador 1 = (12/36) * 100% = 33.33%.
Procesador 2 = (12/36) * 100% = 33.33%

- Para un total de 3 procesadores recorridos superiores.

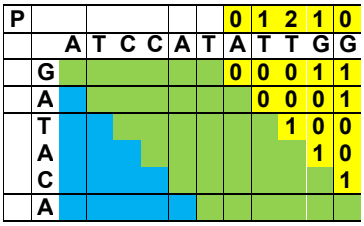

Figura 8. Análisis de recorridos en paralelo usando tres procesadores para recorrido superior

Procesador 0 = 5 + 1 = 6.
Procesador 1 = 4 + 2 = 6.
Procesador 2 = 3 = 3.
Carga total del proceso = 15.

Procesador 0 = (6/15) * 100% = 40%.
Procesador 1 = (6/15) * 100% = 40%.
Procesador 2 = (3/15) * 100% = 20%.

- Para un total de 3 procesadores recorridos inferiores.

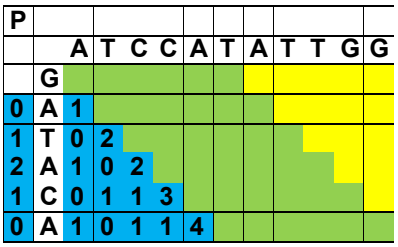

Figura 9. Análisis de recorridos en paralelo usando tres procesadores para recorrido inferior

Procesador 0 = 5 + 1 = 6.
Procesador 1 = 4 + 2 = 6.
Procesador 2 = 3 = 3.

Carga total del proceso = 15.

Procesador 0 = (6/15) * 100% = 40%.
Procesador 1 = (6/15) * 100% = 40%.
Procesador 2 = (3/15) * 100% = 20%.

Figura 10. Análisis de distribución de carga en los procesadores

Al final se usaron 5 procesadores para analizar el comportamiento de la carga, tal como se describirá en los resultados

Datos obtenidos

A continuación se muestran las diferentes corridas de comparaciones entre genomas con el mismo número de pares de bases, en las corridas se probaron con: 1, 2, 3 y 5 procesadores en el cluster, y en cada corrida se comprobó con 10,100, 1000 y 10000 pares de bases en cada genoma.

Cada una de las figuras desde la 11 hasta la 14 es la comprobación, en el eje de la abcisa se muestra el número de procesadores que se utiliza y el el eje de la ordenada se muestra el tiempo consumido por todo el cluster mientras realiza el proceso de alineamiento.

Utilizando 10 pares de bases

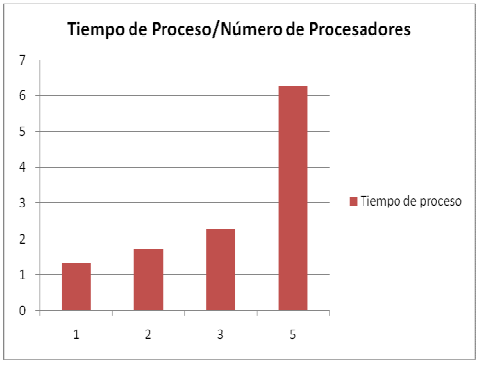

Figura 11. Tiempo de proceso concumido por el cluster usando 10 pares de bases

Utilizando 100 pares de bases

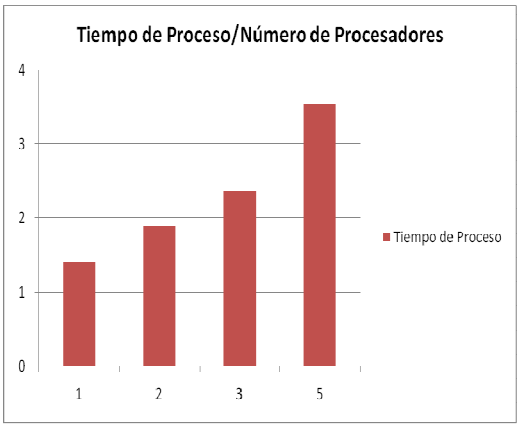

Figura 12. Tiempo de proceso consumido por el cluster usando 100 pares de bases

1000 pares de bases

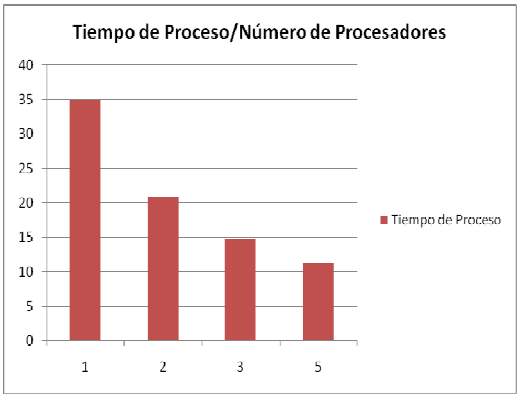

Figura 13. Tiempo de proceso consumido por el cluster usando 1000 pares de bases

Utilizando 10000 pares de bases

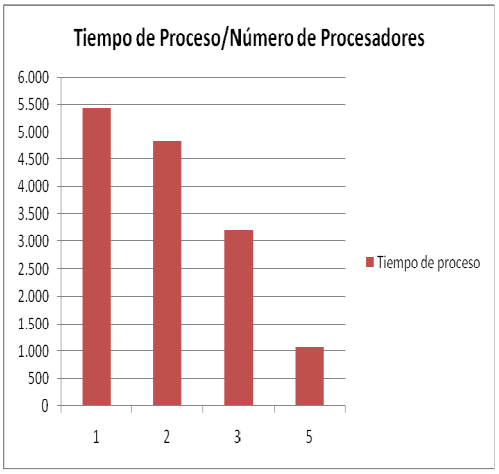

Figura 14. Tiempo de proceso consumido por el cluster usando 10000 pares de bases

En el caso de los procesos que se corren con cantidad diferente de numéros de pares de bases, se determinó que cuando se tienen pocos pares de bases, el cluster no es funcional sino que por el contrario se tarda mas tiempo ya que se presenta el retardo causado por la comunicación entre los nodos, a diferencia de las corridas en las cuales se tienen mas de 100 pares de bases a comparar ahí si es funcional el cluster, ya que los tiempos se disminuyen hasta en un 20% del tiempo que se tarda en una sola máquina.

VI. CONCLUSIONES

En la selección entre MPI y PVM, se tuvieron en cuenta las ventajas entre las cuales sobresale que PVM se encarga de asignar las tareas de forma automática mientras que MPI no, en contra a esto, MPI controla el envío desde el front-end hacia los nodos, mientras que con PVM no se puede controlar, por esta razón se seleccionó MPI como base de trabajo para el procesamiento en paralelo.

El cluster que se implementó fue un Beowulf homogéneo, ya que la Universidad facilitó 8 equipos de cómputo con las mismas características y teniendo en cuenta que se probó ROCKS, OSCAR y MPICH, se decidió por OSCAR, ya que brindó tanto la configuración como la puesta en marcha en una forma muy amigable, y su funcionalidad quedó demostrada.

De los sistemas de alineamiento que se analizaron se hizo la selección entre Alineamiento glogal y Alineamiento local, se tuvieron en cuenta las ventajas y desventajas de cada uno y dio un mejor rendimiento el uso de alineamiento global.

Utilizar un cluster de computadores para realizar alineamiento de genomas es útil cuando las cadenas tienen un número mayor a 1000 pares de bases, debido a que los tiempos de proceso en cada nodo del cluster se incrementan y el tiempo que se tarda en repartir las tareas es muy pequeño, en estos momentos cada vez que se agregan nodos al cluster, éste va a tener un mejor rendimiento en el procesamiento de las cadenas.

De los alineamientos probados en el cluster, se logró una disminución hasta del 80% de tiempo

cuando se utilizaron 5 nodos que fue la mayor cantidad de elementos utilizados.

VII. TRABAJOS FUTUROS

Cabe destacar que la implementación de un Cluster de computadores es el paso inicial para realizar procesos paralelos y se debe continuar con los trabajos de análisis de rendimientos, lo mismo que para ser utilizados en el procesamiento de imágenes, análisis de fluídos, control de alta velocidad, que requieran alta capacidad de cómputo del sistema informático.

Para los procesos que se desarrollan al interior del grupo de Investigación GIINFO (Inscrito en Colciencias) y especialmente del Semillero Especializado en Computación de Alto Rendimiento (SIECAR) se trabaja con los lenguajes de programación Python y C++, esto ha permitido implementar las aplicaciones que funcionan en paralelo en nuestro cluster computacional, permitiendo que algunas aplicaciones exigentes de cómputo puedan ser procesadas no en forma secuencial sino en forma paralela.

Agradecimientos: El autor expresa sus agradecimientos a la Universidad Pontificia Bolivariana Seccional Bucaramanga por asignar los computadores para la realización de este proyecto, lo mismo que los estudiantes del Semillero Especializado en Computación de Alto Rendimiento. http://siecar.upbbga.edu.co

VIII. REFERENCIAS

[1] BERGERON, Bryan. Bioinformatics Computing. Prentice Hall, Pearson Education. 2003.

[2] MOUNT, David. Bioinformatics: Secuence and Genome Analysis. CSHL Press. 2004.

[3] PEVZNER, Pavel. TESLER, Glenn. Genome Rearrangement in Mammalian evolution: Lesson from human and mouse genomes. Genome research. www.genome.org.

[4] LIZÁRRAGA CELAYA, Carlos. clústeres computacionales sobre Linux. www.fisica.uson.mx/carlos/LinuxClusters/cluster-computing2000.pdf. 2005.

[5] http://www.beowulf.org

[6] http://oscar.openclustergroup.org

[7] http://www.fedoraproject.org

[8] GARCÍA LÓPEZ, Areli. DELGADO, Julián J. y CASTAÑEDA, Salvador, Metodologías de paralelizacion en supercomputadoras. www.telematica.cicese.mx/computo/super/cicese2000/paralelo/Part3.html. 2005.

[9] http://www.python.org/