Ingeniería. Investigación y Tecnología
ISSN: 1405-7743
iit.revista@gmail.com
Universidad Nacional Autónoma de México
México

Orozco-Arias, Simon; Camargo-Forero, Leonardo; Correa, Juan Carlos; Guyot, Romain; Cristancho, Marco
BIOS-ParallelBlast: Paralelización optimizada de alineamiento de secuencias sobre Xeon Phi
Ingeniería. Investigación y Tecnología, vol. XVIII, núm. 4, octubre-diciembre, 2017, pp. 423-432
Universidad Nacional Autónoma de México
Distrito Federal, México

Ingeniería Investigación y Tecnología
volumen XVIII (número 4), octubre-diciembre 2017 423-432
ISSN 1405-7743 FI-UNAM artículo arbitrado
Información del artículo: recibido: noviembre de 2016, reevaluado: febrero de 2017, aceptado: marzo de 2017

BIOS-ParallelBlast: Paralelización optimizada de alineamiento de secuencias sobre Xeon Phi

BIOS-ParallelBlast: Optimized sequences alignment parallelization on Xeon Phi

Orozco-Arias Simon

Centro de bioinformática y biología computacional, BIOS, Colombia
Correo: simon.orozco@bios.co

Camargo-Forero Leonardo

Universidad Politécnica de Cataluña, Barcelona
Departamento de Física
Correo: leonardo.camargo@estudiant.upc.edu

Correa Juan Carlos

Centro de bioinformática y biología computacional, BIOS, Colombia
Correo: juan.correa@bios.co

Guyot Romain

Institut de Recherche pour le Développement
Centre IRD de Montpellier, UMR IPME, Francia
Correo: romain.guyot@ird.fr

Cristancho Marco

Centro de bioinformática y biología computacional, BIOS, Colombia
Correo: marco.cristancho@bios.co

Resumen

El uso de supercomputación en la ciencia es necesario, debido a las grandes cantidades de datos que se analizan para obtener resultados impactantes. Las tecnologías *many-core* como Intel Xeon Phi aparecen como una alternativa de acelerar estos análisis y su uso en la supercomputación, en especial, dentro de la bioinformática, donde es más común, pero los resultados no son los esperados al ejecutar *software* altamente demandante en clústeres heterogéneos. En la siguiente investigación se muestra un *wrapper* encargado de paralelizar óptimamente NCBI-Blast sobre varios nodos de CPUs y Xeon Phi, con el objetivo de reducir al máximo los tiempos de ejecución y demostrar la utilidad que pueden tener las tecnologías *many-core* en las ciencias aplicadas.

Descriptores: NCBI-Blast, Xeon Phi, tecnologías many-cores, clústeres heterogéneos, bioinformática, supercomputación heterogénea.

Abstract

Usage of supercomputing in science is more necessary every day, due to the large amount of data that researches have to analyze to obtain significant results. Many-cores technologies such as Intel Xeon Phi were developed as an alternative to accelerate these studies and its use in supercomputers and especially in bioinformatics is more common nowadays, but the results are not good enough when high-demand software is executed in heterogeneous clusters. This research shows a wrapper that achieves parallelization optimally using NCBI-Blast on CPUs and Xeon Phi nodes, the main goal here is to reduce execution times and to demonstrate the usefulness that many-cores technologies contribute in applied sciences.

***Keywords*:** *NCBI-Blast, Xeon Phi, many-cores technologies, heterogeneous clusters, bioinformatics, heterogeneous supercomputing.*

Introducción

En la actualidad, la supercomputación tiene una vital importancia a la hora de lograr avances significativos en ciencias tan diversas como la biología (Smith, 2014), la medicina en la simulación de respuesta a medicamentos (Kim *et al*., 2015) y la matemática, logrando conseguir en horas los resultados que en computadores personales puede tardar meses. Debido a esto, es común encontrar en distintas partes del mundo máquinas robustas que tienen como objetivo procesar grandes cantidades de datos en tiempos cortos, para que los investigadores puedan continuar con su labor y obtener avances en la ciencia mucho más rápido que en tiempos pasados.

La supercomputación se integra de distintas unidades de procesamiento; CPUs, GPUs[1] y Xeon PHI son las más empleadas en la actualidad. Gracias a las arquitecturas *multi-core* y *many-core*, las capacidades de los supercomputadores, que en la actualidad alcanzan mil billones de operaciones de punto flotante por segundo (FLOPS), conocido así por sus siglas en inglés, aumentan de gran manera, pero para aprovechar estas tecnologías se debe contar con *software* realmente paralelizable.

Computación de alto rendimiento

La computación de alto rendimiento o *High Perfomance Computing* (HPC) es un área que abarca varios paradigmas de programación paralela, incluyendo lenguajes e interfaces de programación, *software* especializado, conferencias internacionales, entre otras, se define desde el ámbito técnico y científico (Nielsen, 2016).

El uso de HPC en distintos ámbitos de la bioinformática es más frecuente, como en el análisis de *big data*, procesamiento de señales biomédicas, *deep learning*[2], *machine learning*, entre otras (Min *et al*., 2016) esto se debe a las ventajas que esta tecnología ofrece en cuanto a tiempo de ejecución y herramientas para que investigadores y científicos puedan optimizar el tiempo en sus investigaciones.

En la actualidad la supercomputación es un campo ampliamente estudiado, de hecho se considera uno de los tres pilares de la investigación, gracias a que muchos de los avances en la ciencia y la ingeniería están basados en ella (González *et al*., 2015) en la literatura se encuentran implementaciones realizadas por investigadores en donde buscan solucionar problemas u optimizar recursos, incluso usando máquinas virtuales (Huang *et al*., 2006) y para dar solución a problemáticas en campos tan variados como las matemáticas (Behrends *et al*., 2016), diseño de turbinas de gas (Alhatim, 2016), secuenciación (Al-Ali *et al*., 2016), entre otras.

Bioinformática

La bioinformática es la aplicación de la estadística y las ciencias computacionales en el campo de la biología, donde el principal objetivo es mejorar el entendimiento de los procesos biológicos (Gobalan y John, 2016). La bioinformática ha aportado en muchos roles de la ciencia, por ejemplo, en el cambio climático y el problema del crecimiento de la población (Batley y Edwards, 2016), en la agricultura (Esposito *et al*., 2016), en la medicina, en el entendimiento de enfermedades congénitas, entre otras.

Gracias a los avances en las tecnologías genéticas como *Next Generation Sequencing (NGS)* y en el campo de la biología molecular, la comunidad científica ha generado grandes cantidades de información (Fatumo *et al*., 2014), por esta razón la bioinformática ha tomado mucha importancia en los últimos años, ya que gracias a ella, a los algoritmos y técnicas que aplica, se pueden analizar ágilmente los volúmenes de datos demasiado grandes y así obtener información relevante para las investigaciones.

Software bioinformático altamente paralelizable

Como se mencionó anteriormente, para aprovechar las ventajas de la supercomputación al máximo, se requiere hacer uso de todos los procesadores y esto se logra con *software* altamente paralelizable. En la bioinformática existen herramientas diseñadas para correr en máquinas de alto rendimiento como ABySS (Simpson *et al*., 2009) que es un ensamblador en paralelo, MAFFT (Katoh y Standley 2013) que puede hacer alineaciones múltiples sobre diferentes CPUs y BIOS-ParallelBlast. Este último se empleó en el siguiente estudio para optimizar la ejecución de NCBI-Blast [https://blast.ncbi.nlm.nih.gov/Blast.cgi] sobre Xeon Phi. NCBI-Blast es el software bioinformático más popular para hacer alineamientos locales sobre secuencias de nucleótidos o proteínas contra bases de datos previamente formateadas.

BIOS-ParallelBlast (PB) es un *wrapper* desarrollado en *Python* por el Centro de Bioinformática y Biología

1 GPU Unidad de procesamiento gráfico, en la actualidad se usa como un dispositivo de computación masivamente paralelo en donde una instrucción se ejecuta sobre múltiples datos en paralelo. (Endo *et al*., 2016).

2 Deep Learning o aprendizaje profundo, surgió de las bases de *big data*, la computación paralela y distribuida y algoritmos sofisticados, con aplicaciones en reconocimiento de imágenes y voz y el procesamiento de lenguaje natural (Min *et al*., 2016).

Computacional de Colombia BIOS [www.bios.co] el cual permite la ejecución en paralelo sobre un conjunto de recursos computacionales usando dos esquemas de paralelización – *Trivial Parallelization TP y by-sequence Parallelization bSP* –. En términos generales, un archivo de secuencias se dividió en pequeñas partes dependiendo de los esquemas de paralelización y de los algoritmos de división que se apliquen (BIOS-FastaSplitter, LongerSplitter, SorterSplitter o PhiSplitter). Una vez realizada la división se procesó cada archivo con un NCBI-Blast independiente. Al final se unen los resultados de los alineamientos en un archivo de salida único. En su versión actual soporta diferentes tipos de comparación (blastx, blastp y blastn). Esta herramienta se explica con más detalle en la sección de la metodología.

Importancia de Xeon PHI en la actualidad

En la actualidad, gracias a la gran eficiencia energética y al poder computacional que aportan los Xeon PHI a las arquitecturas heterogéneas, se emplea en diferentes ámbitos de la ciencia y la tecnología, por ejemplo, en la tecnología *cloud* y en especial en *big data* (Paranjape *et al*., 2012).

Cómo optimizar el alineamiento local en Xeon PHI

Actualmente se han planteado diferentes formas de mejorar el rendimiento del NCBI-Blast y de diferentes alineamientos locales en las arquitecturas *multi-core*. Una de estas alternativas es dividir la secuencia *query* y la base de datos en partes mucho más pequeñas, distribuir estas partes mediante MPI e hilos OpenMP y combinar los resultados de todos los procesos en una sola salida (Sawyer *et al*., 2015).

Metodología

Operación de BIOS-ParallelBlast (PB)

El flujo de trabajo general se muestra en la figura 1 de de BIOS-ParallelBlast.

Input. PB recibe dos archivos como entrada:

1. **Sequences File:** Archivo en formato FASTA (secuencias identificadas por el caracter ">") [http://www.ncbi.nlm.nih.gov/BLAST/blastcgihelp.shtml]
2. **Machines file:** Una línea por cada nodo (hostname) que se usará por PB

PB core. El proceso descrito puede variar dependiendo del esquema de paralelización seleccionado (TP o bSP). Información sobre estos esquemas se presenta en las secciones siguientes. PB puede soportar blastx, blastn y blastp.

Input file fragmentation. Teniendo en cuenta el esquema de paralelización escogido, la entrada se divide en pequeños fragmentos, usando uno de los cuatro algoritmos de división de secuencias (BIOS-FastaSplitter, LongerSplitter, SorterSplitter o PhiSplitter), los cuales se presentan en las siguientes secciones.

Parallelization. *Puede ser TP o bSP.*

Monitor launching. Una vez que el segmento del archivo se envía al procesado por una máquina seleccionada, un proceso de monitoreo se lanza por PB. Esta fase de monitoreo permite a PB enviar un nuevo segmento solo una

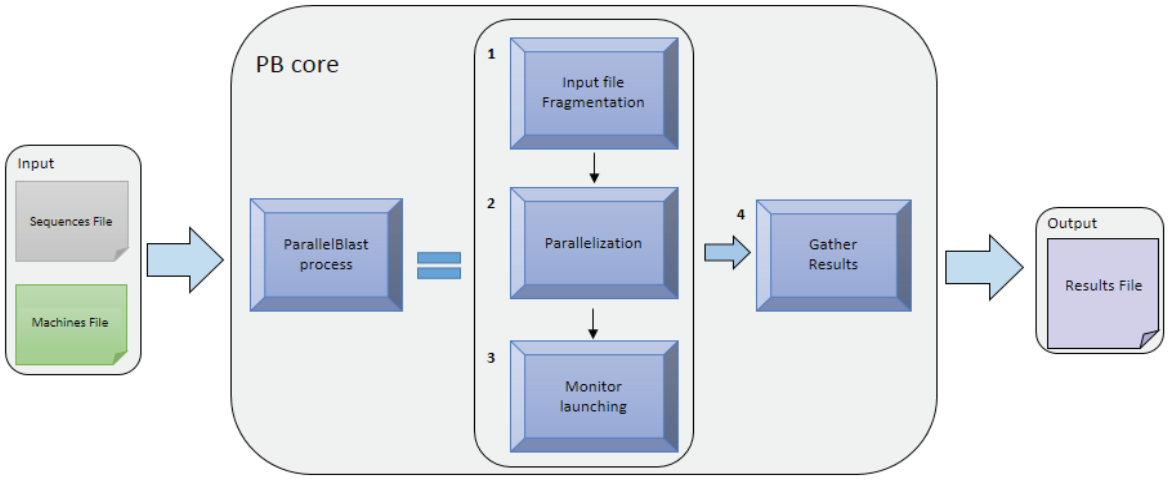

Figura 1. Flujo de trabajo general de BIOS-ParallelBlast. Autoría propia

vez, donde el proceso se termina en una máquina (bSP) y une las salidas de las alineaciones en un solo archivo (TP y bSP).

Trivial parallelization

El primer esquema de paralización disponible en PB es *Trivial Parallelization – TP,* sobre el cual pueden ejecutarse cuatro algoritmos de división.

BIOS-fastaSplitter

Este algoritmo divide las secuencias de entrada en pequeñas piezas con el número de secuencias calculadas basado en el número de máquinas incluidas en el archivo de hosts.

$$NumSeq = \frac{TotalNumSeqArchivoEntrada}{NumeroMaquinas} \quad (1)$$

donde

TotalNumSeqArchivoEntrada = Número total de secuencias a analizar
NumeroMaquinas = Número de nodos incluidos en el archivo de hosts

Este algoritmo no tiene en cuenta las longitudes de las secuencias, por lo que es ideal cuando se ejecuta sobre *Reads,* que tienen la misma longitud. Se recomienda el uso de este algoritmo cuando se lanza PB únicamente sobre nodos de CPU (no sobre Xeon Phi).

LongerSplitter

En primer lugar, calcula si la cantidad de secuencias y su longitud es apropiada, si cumple con las condiciones, calcula la longitud total de todas las secuencias contenidas dentro del archivo, excluyendo su encabezado y divide este número entre el número de máquinas donde se ejecutará PB, con el fin de tener una repartición equivalente cuando se cuenta con secuencias de diferentes longitudes dentro del archivo.

$$LongSeq = \frac{TotalLongitudSeqArchivoEntrada}{NumeroMaquinas} \quad (2)$$

donde

TotalLongitudSeqArchivoEntrada = Longitud total de las secuencias a analizar, excluyendo su encabezado
NúmeroMáquinas = Número de nodos incluidos en el archivo de hosts

Si no cumple con los requisitos ordena el archivo con secuencias por longitud, en orden descendente, y reparte las secuencias en el número de archivos solicitados por iteraciones. Si el número de máquinas es n, las primeras n-secuencias se copian en los n-archivos, una por archivo, continuando hasta terminar la totalidad de las secuencias. El resultado son n-archivos con la misma cantidad de secuencias ordenadas por longitud. El propósito de esta división es tener una longitud total de las secuencias de cada archivo lo más similar posible.

Se recomienda el uso cuando se cuenta con archivos de secuencias de diferentes longitudes y cuando se desea ejecutar PB únicamente sobre nodos de CPUs (no sobre Xeon Phi).

SorterSplitter

En primer lugar ordena las secuencias dentro del archivo por longitud en orden descendente y repite el proceso del algoritmo BIOS-fastaSplitter (dividir en archivos más pequeños, todos con la misma cantidad de secuencias). El propósito de este algoritmo es generar archivos de diferentes tamaños (en orden decreciente, el primer archivo será el más grande y el último el más pequeño) para que los nodos con CPUs analicen los archivos más grandes y los nodos con Xeon Phi los archivos más pequeños. Se requiere adicionalmente que el archivo de hosts diferencie de forma notoria los nodos con CPUs y los nodos con Xeon Phi (se recomienda usar el sufijo –mic para Xeon Phi).

PhiSplitter

En primer lugar, calcula dentro del archivo de hosts, cuántas máquinas son nodos de CPUs y cuántas son nodos de Xeon Phi, debido a que estos factores son necesarios para calcular la cantidad de secuencias que contendrán los archivos. Cabe aclarar que este algoritmo crea archivos 3 veces más grandes para los nodos de CPUs que los archivos para los nodos de Xeon Phi.

$$LongSeq = \frac{TotalLongitudSeqArchivoEntrada}{3 * TotalNodosCPU + TotalNodosPhi} \quad (3)$$

donde

TotalLongitudSeqArchivoEntrada = Longitud total de las secuencias a analizar, excluyendo su encabezado
TotalNodosCPU = Número de nodos de CPUs incluidos en el archivo de hosts

TotalNodosPhi = Número de nodos de Xeon Phi incluidos en el archivo de hosts

El factor diferenciado de tamaño (tres) se encontró luego de pruebas de comparación de tiempos de ejecución de alineamientos con NCBI-Blast, realizados con los mismos archivos de entrada y la misma base de datos sobre Nodos de CPUs y Nodos de Xeon Phi.

El uso de este algoritmo se recomienda cuando se ejecuta PB sobre nodos con CPUs y nodos de Xeon Phi (tabla 4, Comparación Xeon y Xeon Phi).

El flujo de trabajo de PB-TP, en general, se presenta en la figura 2, de flujo de trabajo PB-TP.

By-Sequence Parallelization

El segundo algoritmo de paralelización disponible en PB es *by-Sequence Parallelization–bSP* que consiste en dividir el archivo de secuencias de entrada en partes más pequeñas con el número de secuencias calculadas basado en el mínino número de hilos disponibles en el grupo de máquinas incluidas en el archivo de hosts. Por ejemplo, si el archivo contiene dos máquinas con 32 hilos disponibles en la primera y 64 hilos disponibles en el segundo, el archivo de entrada se divide en 32 partes. A continuación, estas piezas se procesan en los nodos en un orden creciente acorde con los nodos que están disponibles o que ya terminaron de procesar su archivo de entrada. El número de secuencias a ser procesadas en un ciclo de PB-bSP se obtiene por la siguiente ecuación

$$NumSeq = MinMaxParThrInMach \quad (4)$$

donde

MinMaxParThrInMach = Mínimo número de hilos paralelos a ser ejecutados por máquina en el archivo de hosts

El número total de ciclos se calcula (ciclo = procesamiento de una pieza más monitoreo de esa ejecución) de la siguiente forma

$$NumCycles = \frac{TotalNumSeqInputFile}{MinMaxParThrInMach} \quad (5)$$

El flujo de trabajo de PB-bSP se muestra en la figura 3.

Herramientas adicionales

BIOS-ParallelBlast provee actualmente una función adicional:

Creación de una base de datos formateada para NCBI-Blast

Donde PB soporta la creación de una base de datos formateada para **NCBI-Blast** basada en un archivo con secuencias FASTA (blastx, blastn, blastp).

Resultados

Las pruebas se realizaron sobre nodos de CPUs con Xeon E5-2670, cada uno con 32 *cores* (HT) a 2.60GHz y 256 GB de memoria RAM y con tarjetas Xeon Phi 5110p con 60 *cores* a 1.053 GHz, cada una con 8 GB de RAM y número total de 240 hilos. Es importante mencionar, que solo se incluyen los resultados del paradigma de paralelización TP, debido a que es el más efectivo. La versión de NCBI-Blast sobre la cual se ejecutaron las pruebas fue 2.2.31.

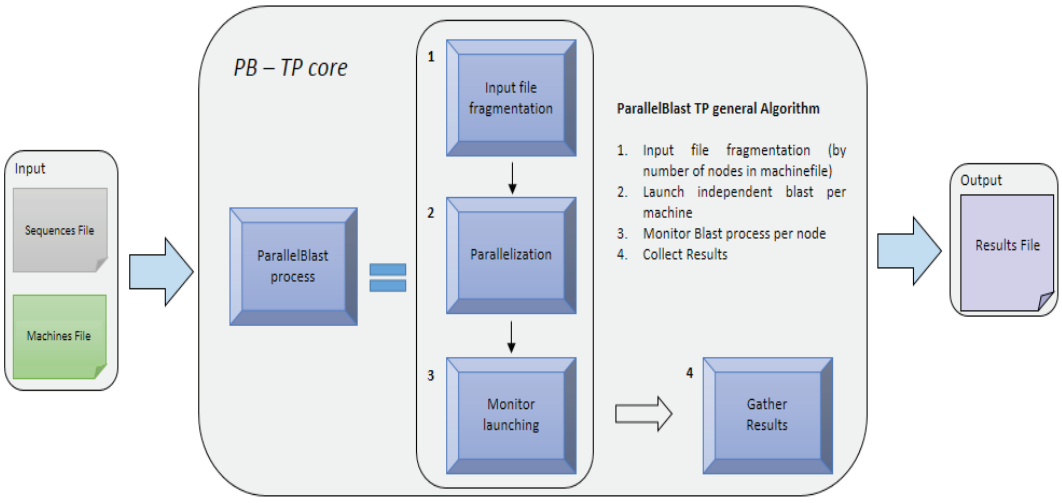

Figura 2. Flujo de trabajo PB-TP. Autoría propia

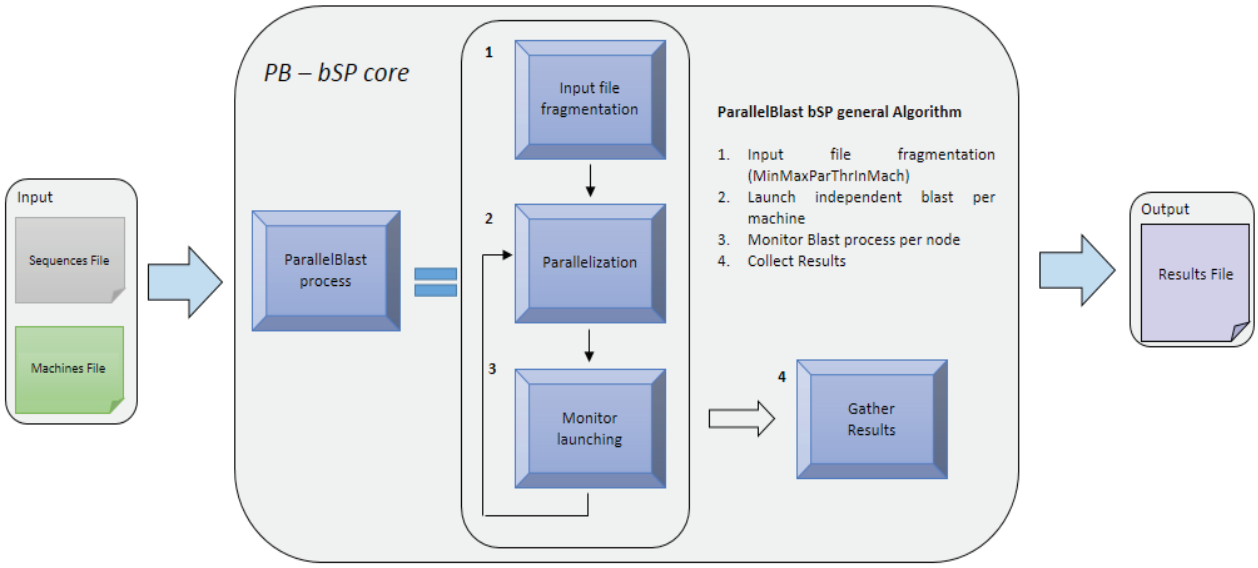

Figura 3. Flujo de trabajo PB-bSP. Autoría propia

Los archivos se comparten bajo el protocolo NFS. En la tabla 1 se presentan los archivos utilizados en las ejecuciones. Todos los archivos contienen bases de nucleótidos, además se usó la opción *blastx* de PB. El archivo de referencia fue *env_nr* [ftp://ftp.ncbi.nlm.nih.gov/blast/db/FASTA/env_nr.gz], que contiene 6.907.428 secuencias de aminoácidos, previamente formateado para NCBI-Blast.

Resultados algoritmos de división

A continuación se muestran los resultados correspondientes a las salidas de los algoritmos de división utilizados por PB. Cabe aclarar que para el algoritmo de división PhiSplitter se emplearon 1 nodo de CPUs + 1 tarjeta Xeon Phi y 1 nodo de CPUs + 2 tarjetas Xeon Phi, respectivamente.
Los valores corresponden al porcentaje de peso en cada una de las partes en que se divide el archivo.

Comparación entre ejecución de NCBI-Blast en un nodo con CPUs y con una tarjeta Xeon Phi

A continuación se muestran los resultados de la ejecución de PB sobre 1 nodo de CPUs y una tarjeta Xeo Phi. El tiempo se muestra en segundos. La base de datos de

Tabla 1. Archivos de secuencias

Nombre del Archivo*	Tamaño en MB	Cantidad de Secuencias	Longitud total de Secuencias	Secuencias con diferentes longitudes
Cafe-ensamblado.fa	0.156	14	163.667	Si
Pdbnt.fa	3.1	7.722	769.828	Si
Cafe-illumina.fa	1.6	10.000	1.020.000	No

* Estas secuencias se pueden descargar en el siguiente enlace: (https://github.com/simonorozcoarias/ncbi-blast_sobre_xeon_phi)

Tabla 2. Algoritmos de división en dos partes

Programa	Cafe-ensamblado.fa		Pdbnt.fa		Cafe-illumina.fa	
	Parte 1	Parte 2	Parte 1	Parte 2	Parte 1	Parte 2
BIOS-fastaSplitter	**0.68**	**99.32**	**38.46**	**61.54**	**50.00**	**50.00**
LongerSplitter	**55.96**	44.04	51.36	48.64	51.00	49.00
SorterSplitter	**99.39**	0.61	93.84	6.16	50.00	50.00
PhiSplitter	**99.48**	0.52	76.52	23.48	76.50	23.50

Tabla 3. Algoritmos de división en tres partes

Programa	Cafe-ensamblado.fa			Pdbnt.fa			Cafe-illumina.fa		
	Parte 1	Parte 2	Parte 3	Parte 1	Parte 2	Parte 3	Parte 1	Parte 2	Parte 3
BIOS-fastaSplitter	0.36	0.49	99.15	16.45	44.85	38.70	33.33	33.34	33.33
LongerSplitter	47.04	43.48	9.48	34.19	33.96	31.84	33.55	33.55	31.58
SorterSplitter	99.15	0.51	0.34	90.47	5.95	3.57	33.79	33.79	33.77
PhiSplitter	99.15	0.51	0.34	61.25	20.72	18.03	53.25	17.75	16.01

Tabla 4. Comparación Xeon y Xeon Phi

Archivo	Xeon	Xeon Phi	Relación
Cafe-ensamblado.fa	257.54	651.81	2.53
Pdbnt.fa	1099.43	2830.43	2.57
Cafe-illumina.fa	1219.22	3263.34	2.68
		PROMEDIO	2.59

Comparación en la ejecución con Xeon Phi

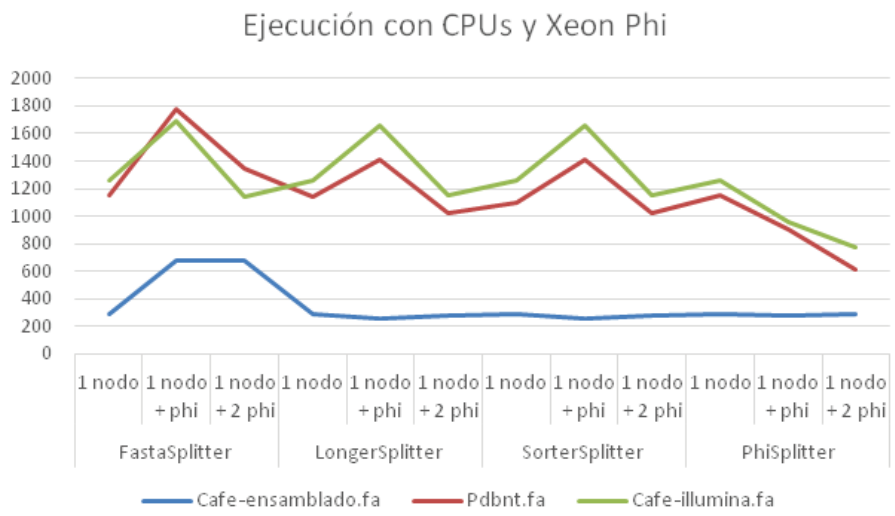

Figura 4. Ejecución con CPUs y Xeon Phi

Comparación en la ejecución sobre nodos de CPUs

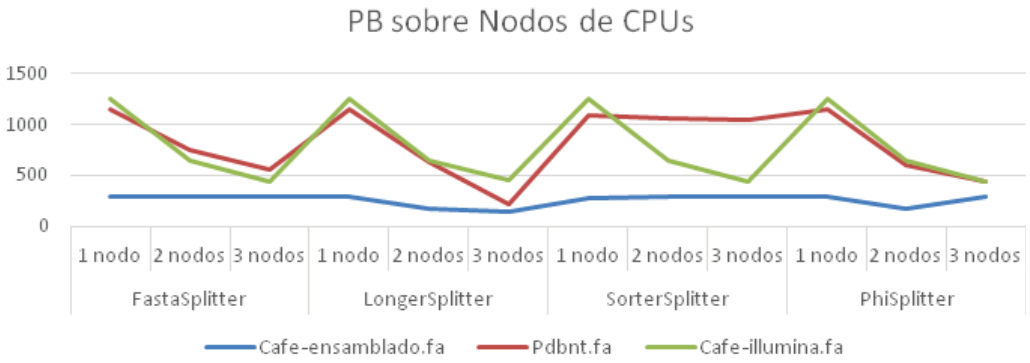

Figura 5. Ejecución de PB sobre Nodos de CPUs

referencia fue la misma que para los demás análisis (env_nr).

Ejecución de PB sobre nodos con CPUs y Xeon Phi al mismo tiempo

A continuación se presentan los resultados de la ejecución sobre nodos con CPUs y nodos con CPUs y tarjetas Xeon Phi al mismo tiempo. El tiempo se muestra en segundos.

Conclusiones

Se logró demostrar que por medio de la división de los archivos de entrada fue posible optimizar el tiempo de ejecución del programa NCBI-BLAST para el análisis de secuencias de nucleótidos usando CPUs en conjunto con Xeon Phi.

Un nodo con CPUs (Xeon E5-2670) con 32 *cores* en HT y 256 GB de memoria RAM, ejecutan en promedio 2.6 veces más rápido un alineamiento con NCBI-Blast que un nodo de Xeon Phi (5110p) accedido por SSH, con 60 *cores*, 240 hilos y 8 GB de memoria RAM. Sin embargo, implementado el esquema de paralelización TP combinado con el algoritmo de divisón PhiSplitter se logra usar de forma óptima PB sobre nodos de CPU y Xeon Phi al mismo tiempo. Esto se debe a que el algoritmo tiene en cuenta el factor de ejecución de los nodos con CPUs, eliminando así el tiempo de diferencia entre las múltiples ejecuciones y reduciendo considerablemente los tiempos muertos de los recursos.

Cuando el archivo de secuencias de entradas tiene muy pocas secuencias y estas tienen longitudes variantes, su paralelización se complejiza, afectado el tiempo de ejecución. No se logra una disminución considerable al usar las tarjetas Xeon Phi en comparación con su ejecución en una sola máquina.

Si el conjunto de nodos donde será ejecutado PB está compuesto únicamente por nodos con CPUs y el archivo de entrada consiste en secuencias de diferentes longitudes, es más óptimo usar el esquema de paralelización TP combinado con el algoritmo de división LongerSplitter, ya que tiene en cuenta las longitudes de las secuencias y crea archivos más uniformes.

En caso de que los nodos donde se ejecutará PB sean exclusivamente nodos de CPUs y el archivo de secuencias comprenda solo secuencias de la misma longitud, es recomendable el uso del esquema de paralelización TP y el algoritmo de división BIOS-fastaSplitter.

Referencias

Al-Ali R., Kathiresan N., El Anbari M., Schendel E.R., Zaid T.A. Workflow optimization of performance and quality of service for bioinformatics application in high performance computing. *Journal of Computational Science*, 2016.

Alhatim O. Current status and prospects of supercomputing used for gas turbine engines design, 2016.

Batley J. y Edwards D. The application of genomics and bioinformatics to accelerate crop improvement in a changing climate. *Current opinion in plant biology,* volumen 30, 2016: 78-81.

Behrends R., Hammond K., Janjic V., Konovalov A., Linton S., Loidl H.W., Trinder P. HPC-GAP: engineering a 21st-century high-performance computer algebra system. *Concurrency and Computation: Practice and Experience*, 2016.

Endo Y., Shimobaba T., Kakue T., Ito T. GPU-accelerated compressive holography. *Optics Express,* volumen 24 (número 8), 2016: 8437-8445.

Esposito A., Colantuono C., Ruggieri V., Chiusano M.L. Bioinformatics for agriculture in the Next-Generation sequencing era. *Chemical and Biological Technologies in Agriculture,* volumen 3 (número 1), 2016: 1.

Fatumo S.A., Adoga M.P., Ojo O.O., Oluwagbemi O., Adeoye T., Ewejobi I., Nashiru O. Computational biology and bioinformatics in Nigeria. *PLoS Comput Biol,* volumen 10 (número 4), 2014: e1003516.

Gobalan K. y John A. Applications of bioinformatics in genomics and proteomics. *Journal of Advanced Applied Scientific Research,*volumen 1 (número 3), 2016: 29-42.

González Á.F., Rosillo R., Dávila J.Á.M., Olivera V.M. Historical review and future challenges in Supercomputing and Networks of Scientific Communication, 2015.

Huang W., Liu J., Abali B., Panda D.K. *A case for high performance computing with virtual machines.* Paper presented at the Proceedings of the 20th annual international conference on Supercomputing, 2006.

Kim J.S., Rho S., Lee M., Kim S., Kim S., Hwang S. A case study of drug repositioning simulation based on distributed supercomputing technology. *Journal of KIISE,* volumen 42 (número 1), 2015: 15-22.

Min S., Lee B., Yoon S. Deep learning in bioinformatics. *arXiv preprint arXiv:1603.06430*, 2016.

Nielsen F. *Introduction to HPC with MPI for Data Science*, Springer, 2016.

Paranjape K., Hebert S., Masson B. Heterogeneous computing in the cloud: crunching big data and democratizing hpc access for the life sciences. *Intel White Paper*, 2012.

Sawyer S.E., Rekepalli B., Horton M.D., Brook R.G. hpc-blast: distributed blast for xeon phi clusters. Paper presented at the Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics, 2015.

Smith J.C. Progress and prospects in extreme scale supercomputing in biology, bioenergy, and medicine. Paper presented at the abstracts of papers of the american chemical society, 2014.

Simpson J.T., Wong K., Jackman S.D., Schein J.E., Jones S.J.M., Birol İ. ABySS: A parallel assembler for short read sequence data. *Genome Research*, volumen 19 (número 6), 2009: 1117-1123 [en línea]. Disponible en: http://doi.org/10.1101/gr.089532.108

Katoh K. y Standley D.M. MAFFT Multiple sequence alignment software version 7: Improvements in performance and usability. *Molecular Biology and Evolution*, volumen 30 (número 4), 2013: 772-780 [en línea]. Disponible en: http://doi.org/10.1093/molbev/mst010

Bibliografía

Gálvez S., Ferusic A., Esteban F.J., Hernández P., Caballero J.A., Dorado G. Speeding-up bioinformatics algorithms with heterogeneous architectures: highly heterogeneous smith-waterman (HHeterSW). *Journal of Computational Biology*, 2016.

Liu Y., Huang W., Johnson J., Vaidya S. Gpu accelerated smith-waterman Computational Science–ICCS 2006, Springer, 2006, pp. 188-195.

Liu Y. y Schmidt B. SWAPHI: Smith-waterman protein database search on Xeon Phi coprocessors. Paper presented at the Application-specific Systems, Architectures and Processors (ASAP), 2014 IEEE 25th International Conference, 2014.

Liu Y., Tran T.T., Lauenroth F., Schmidt B. SWAPHI-LS: Smith-Waterman Algorithm on Xeon Phi Coprocessors for Long DNA Sequences. Paper presented at the Cluster Computing (CLUSTER), 2014 IEEE International Conference, 2014.

Liu Y., Wirawan A., Schmidt B. CUDASW++ 3.0: accelerating Smith-Waterman protein database search by coupling CPU and GPU SIMD instructions. *BMC bioinformatics*, volumen 14 (número 1), 2013: 117.

Ramstad J. Protein alignment on the Intel Xeon Phi Coprocessor, 2015.

Rucci E., De Giusti A., Naiouf M., Botella G., Garcia C., Prieto-Matias M. Smith-Waterman algorithm on heterogeneous systems: A case study. Paper presented at the Cluster Computing (CLUSTER), 2014 IEEE International Conference, 2014.

Citación sugerida:

Citación estilo Chicago

Orozco-Arias, Simon, Leonardo Camargo-Forero, Juan Carlos Correa, Romain Guyot, Marco Cristancho. BIOS-ParallelBlast: Paralelización optimizada de alineamiento de secuencias sobre Xeon Phi. *Ingeniería Investigación y Tecnología*, XVIII, 04 (2017): 423-432.

Citación estilo ISO 690:

Orozco-Arias S., Camargo-Forero L., Correa J.C., Guyot R., Cristancho M.A. BIOS-ParallelBlast: Paralelización optimizada de alineamiento de secuencias sobre Xeon Phi. *Ingeniería Investigación y Tecnología*, volumen XVIII (número 4), octubre-diciembre 2017: 423-432.

Semblanzas de los autores

Orozco-Arias Simon. Ingeniero en sistemas por la Universidad de Caldas. Actualmente labora en el área de bioingeniería del Centro de Bioinformática y Biología Computacional de Colombia BIOS, tiene experiencia en supercomputación e hizo parte del semillero de investigación en HPC con énfasis en arquitecturas many-cores y multi-cores. Ha participado en investigaciones con arquitecturas Grid y arquitecturas many-cores como Xeon phi optimizando la respuesta de programas bioinformáticos.

Camargo-Forero Leonardo. Ingeniero en sistemas por la Universidad Industrial de Santander en Bucaramanga, Colombia, 2010. Posee un título de magister en redes y computación ubicua por la Université Nice Sophia Antipolis en Niza, Francia, 2012. Actualmente es candidato al título doctoral en ciencia y tecnología aeroespacial en la Universitat Politècnica de Catalunya en Barcelona, España. Ha trabajado en computación de altas prestaciones durante los últimos 7 años en diferentes roles tales como administrador de plataforma en centros de supercomputación, desarrollador de software paralelo y arquitecto de sistemas para campos tales como óptica, exploración espacial, bioinformática y robótica.

Correa Juan Carlos. Ingeniero telemático con 11 años de experiencia en implementación y administración de plataformas Linux y networking. Ha trabajado en los últimos años en temas de computación científica y HPC. Su interés actual comprende el análisis de datos aplicado a la seguridad informática.

Guyot Romain. Investigador del IRD (Institut de Recherche pour le Développement, Francia) desde hace once años. Realizó su tesis de doctorado en la Universidad de Zúrich en Suiza, estudiando la evolución del genoma de trigo y su HDR (Habilitation à diriger des recherches) en la Universidad de Montpellier, Francia, sobre la evolución de los genomas de plantas. Ha participado en el análisis de numerosos genomas de especies vegetales, utilizando estrategias de genética, genómica, biotecnología y bioinformática. Su principal interés es comprender la evolución y la estructura de los genomas y el impacto que tienen sobre ellas las secuencias repetidas.

Cristancho Marco. Investigador científico con más de 13 años de experiencia en bioinformática y más de 20 años de experiencia en las áreas de la genómica, la biología molecular y el mejoramiento del cultivo del café. Ha participado en un gran número de iniciativas de colaboraciones nacionales e internacionales y es asesor en las áreas de genómica, bioinformática y estrategias de control de patógenos de plantas, para centros de investigación públicos y privados en Colombia y otros países.