OPTIMIZACIÓN DE ALGORITMOS Y REDUCCIÓN DE LA HUELLA DE CARBONO MEDIANTE EL USO DE ESTRUCTURAS DE DATOS COMPACTAS Y TÉCNICAS DE COMPUTACIÓN DE ALTO RENDIMIENTO PARA BIG DATA

Un Algoritmo es, en palabras sencillas, la receta o el camino con las instrucciones concretas y sin ambigüedad para resolver una tarea especifica dentro de un modelo de cómputo bien definido. El modelo es la abstracción que describe al escenario y las reglas físicas y matemáticas que se deben respetar en el proceso. En las áreas de las Ciencias de la Computación, Informática y Tecnología, los algoritmos son la base que sustenta todas sus contribuciones científicas, y por ende lo es también en la industria relacionada. Prácticamente todo el mundo gira en torno a los algoritmos, ya que en el corazón de toda función que se ejecuta en todo proceso productivo, hay siempre un algoritmo que previamente fue diseñado para dicho propósito. De aquí la gran relevancia que tiene el área del Diseño y Análisis de Algoritmos, en el estudio del *performance* y de los recursos que estos consumen.

Dentro de las Ciencias de la Computación e Informática, la Computación de Alto Rendimiento (HPC) busca potenciar los cálculos dentro de sistemas informáticos que involucran cómputos complejos. Su principal objetivo es lograr una considerable reducción en el tiempo de cómputo de un complicado proceso computacional, siendo la computación en paralelo la técnica más utilizada para este fin. El paralelismo es posible tanto en arquitecturas de CPU —con procesadores que cuentan con varios núcleos— o en arquitecturas de GPU (unidad de procesamiento gráfico), las cuales disponen de miles de núcleos optimizados para el procesamiento en paralelo [NHM14]. De este modo, las tarjetas gráficas han despertado gran interes dentro de la computación de alto rendimiento. HPC debe su característica de informática de alta velocidad a su gran capacidad para procesar información [EGC11]. Este es un tema crucial para esta propuesta de investigación, ya que si bien hoy en día hablar de HPC es prácticamente un sinónimo de paralelismo, hay otras técnicas con gran potencial para acelerar los cálculos en sistemas complejos, y en particular en aquellos que operan sobre grandes volúmenes de datos. Considere a un veloz algoritmo ***A*** que resuelve un cierto problema ***P***, este también requiere de una cierta cantidad de memoria principal en donde operar, la que depende del tamaño de su entrada, digamos ***n***. Cuando los datos de entradas son de pequeñas a medianas escalas, muchas veces la memoria requerida no es muy significativa y de hecho no hace necesario ningún tipo de estudio. Sin embargo, cuando se procesan grandes cantidades de datos, como en *Big Data* —que por lo demás, la tendencia es realizar análisis de grandes magnitudes de datos digitales— es cuando se hace necesario de algoritmos que construyan sus estructuras de datos de modo eficiente, disminuyendo los requerimientos de memoria principal.

Considere además que los discos modernos de estado sólido (SSD) son ordenes de magnitud más lentos que el acceso directo a la memoria principal, por tanto para lograr mayor velocidad en los cómputos complejos, se deben diseñar algoritmos que, en lo posible, construyan la totalidad de sus estructuras de datos en RAM. Todo esto indica que la compresión de datos es una herramienta a considerar en el diseño de algoritmos cuando trabajamos en *Big Data*. Considere también que no toda institución posee la infraestructura de las llamadas “gigantes” de la informática, como Google o Facebook, que poseen decenas de supercomputadores que operan sobre enormes memorias distribuidas. Por lo demás, estás grandes compañías tampoco trabajan con energía totalmente limpias, ni en el almacenamiento de sus enormes cantidades de datos digitales, ni en la implementación y puesta en producción de sus algoritmos más complejos. Dejando con todo esto una contundente huella de carbono para nuestro ambiente.

La realidad es que lo recursos computacionales siempre son limitados y por tanto se hace necesario un uso eficiente de ellos, lo que conlleva a la búsqueda de nuevos diseños de algoritmos que, en lo posible, evadan trabajar con sus datos en memoria secundaria. Salomon [108] define la Compresión de Datos, o *Data Compression* (DC), como el proceso de convertir un flujo de datos de entrada (los datos originales) en otro flujo de datos (la salida o el *compressed stream*) que es de menor tamaño. Para nuestro escenario, DC por si solo tampoco es una opción, ya que si bien existen muy buenos

algoritmos de compresión, como los basados en compresión *Lempel and Ziv* [ZL77, ZL78], en muchos casos sus estructuras deben descomprimirse casi en su totalidad, incluso para trabajar con una pequeña porción de ellos. En las recientes décadas, ha surgido una técnica llamada Diseño de Estructuras de Datos Compactas [Nav16, NM07], la cuál busca construir estructuras de datos sucintas o compactas. Estas requieren de menos espacio que los datos explícitos, pero aún requieren de mayor espacio que una totalmente comprimida y a su vez ofrecen rápidas operaciones de consulta sobre los datos originales. Algoritmos que construyen dichas estructuras son de gran aceptación hoy en día, un claro ejemplo son los llamados Índices Comprimidos, siendo alguno de los más populares los de la familia FMI [FM00, FMMN07], o los basados en Arreglos de Sufijos (SA) [NM07]. Por otro lado, si bien existen populares técnicas de compresión que no pueden recuperar toda la información original —esta es la llamada compresión con pérdida, usada primordialmente sobre datos multimedias— en esta propuesta se enfoca en compresión pura y sin pérdida.

Algoritmos optimizados en base a técnicas de HPC y estructuras de datos más pequeñas como resultado de eliminar la redundancia en los datos, conllevará a implementaciones con un mejor *performance* empírico. Todo esto a punta a la construcción de software que reduzca en consumo de tres recursos vitales: tiempo, memoria principal y energía. Consecuentemente, en este proyecto nos enfocamos en el diseño de este tipo de algoritmos, los cuales pueden aplicarse a diversos problemas computacionales y de optimización. Finalmente, esta propuesta busca colaborar en el cumplimiento de la primera competencia sello de nuestra universidad, cuyo compromiso es con el conocimiento, a través de buenos y eficientes algoritmos; la naturaleza y la protección ambiental, a través de la reducción en la huella de carbono en algoritmos que operan sobre *big data;* y el desarrollo sustentable, a través de mejoras en todo el proceso productivo y puesta en funcionamiento del software resultante.

HIPÓTESIS

El diseño de algoritmos en base a estructuras de datos compactas y técnicas de HPC mejora significativamente el rendimiento en procesos de cómputos complejos con grandes volúmenes de datos, minimizando el consumo de memoria, de energía y del tiempo requerido en los cálculos más complejos necesarios para operar en tiempo real.

OBJETIVOS

- Diseñar algoritmos que construyan estructuras de datos compactas con funcionalidades agregadas para la recuperación y consulta de datos, a través de técnicas de HPC.
- Construir algoritmos cuya implementación requieran de: i) un menor consumo de energía, ii) una disminución en su uso de la memoria y iii) una disminución del tiempo de construcción y del requerido en los cálculos más importantes requeridos por los usuarios.
- Ofrecer técnicas de diseño de algoritmos eficientes que puedan homologarse en la industria asociada, minimizando su negativo impacto ambiental.

METODOLOGÍA Y RESULTADOS ESPERADOS

Esta será en base a una colaboración mixta entre alumnos de pre y postgrado (Magister en Inf.) y profesores. El responsables del proyecto capacitarán a los alumnos participantes en las competencias relacionadas con: el diseño de algoritmos eficientes, técnicas de HPC, compresión en Big Data; y brindar la guía y soporte requerido.
Se espera enviar 2 publicaciones científicas en journals del área y/o congresos.

Bibliografía:

[EGC11]. V. Eijkhout, R. van de Geijn and E. Chow. lulu.com 2011. Ver http://pages.tacc.utexas.edu/ eijkhout/istc/istc.html

[FM00] P. Ferragina and G. Manzini. Opportunistic data structures with applications. In Proc. 41st Annual Symposium on Foundations of Computer Science (FOCS), pages 390–398, 2000.

[FMMN07] P. Ferragina, G. Manzini, V. Mäkinen, and G. Navarro. Compressed representations of sequences and full-text indexes. ACM Transactions on Algorithms, 3(2), 2007.

[NM07] G. Navarro and V. Mäkinen. Compressed full-text indexes. ACM Computing Surveys, 39(1):article 2, 2007.

[Nav16] G. Navarro, Compact Data Structures: A Practical Approach. Cambridge: Cambridge University Press. doi:10.1017/CBO9781316588284, 2016.

[NHM14] C. Navarro, N. Hitschfeld-Kahler and L. Mateu. A survey on parallel computing and its applications in data-parallel problems using GPU architectures. Commun. Comput. Phys., 15, pp. 285-329, 2014.

[Sa06] D. Salomon. Data Compression: The Complete Reference. Springer-Verlag New York, Inc., 2006.

[ZL78] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. IEEE Transactions on Information Theory, 24(5):530 – 536, 1978.

[ZL77] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. IEEE Transactions on Information Theory, 23(3):337–343, 1977.