UNSA

UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

FACULTAD DE INGENIERÍA Y PRODUCCIÓN DE SERVICIOS

ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACIÓN

"AÑO DEL BICENTENARIO DEL PERÚ: 200 AÑOS DE INDEPENDENCIA"

ESTADÍSTICA Y PROBABILIDADES

GRUPO: B SEMESTRE: 2021 - B

Analisis Teorico de la Complejidad Computacional

INTEGRANTES:

Sumare Uscca Josue Gabriel

Gutierrez Arratia Ronald Romario

Ituccayasi Umeres Marko Marcelo

Alvarez Astete Jheeremy Manuel

DOCENTE:

Ronny Ivan Gonzales Medina

Arequipa – Perú

2021

ÍNDICE

INTRODUCCIÓN

La teoría de la complejidad es una rama de la teoría de la computación que analiza a través de modelos matemáticos la “dificultad” de los algoritmos.

Dicha complejidad se refiere a la cantidad de pasos o líneas de código que necesita un algoritmo para funcionar, lo que trae consigo un consumo de recursos relacionados al hardware del computador proporcional a dicha cantidad.

El propósito principal de esta rama es el determinar los límites de aquello programable en una computadora , es decir aquello que una computadora puede hacer considerando los recursos disponibles.La eficiencia es un término muy importante a definir, hace referencia a una complejidad que hace uso de menos recursos.

Los lenguajes de programación tienden a ser evaluados también por diferentes parámetros con el fin de determinar el uso de recursos , la tendencia más común es que mientras dichos lenguajes sean más “fáciles” de programar el consumo de recursos es mayor , por el hecho que para que tengan estas características necesitan de pasar por varias “capas” que facilitan la generación de código .

A continuación presentaremos parámetros relacionados a el consumo de energía de diferentes lenguajes de programación para así poder un rango de aquellos lenguajes con mayor eficiencia en esta característica.

CAPÍTULO I

DEFINICIONES Y CONCEPTOS PRINCIPALES

La ciencia de la computación es un cuerpo sistematizado del conocimiento concerniente al cálculo, que se sostiene en dos áreas fundamentales: La Teoría de la Computabilidad, basada en las ideas y los modelos fundamentales subyacentes al cálculo, y las técnicas de la ingeniería para el diseño de algoritmos: algoritmos voraces (divide y vencerás), algoritmos aleatorizados, algoritmos con retroceso, etc

Un proceso computacional, también llamado proceso algorítmico o algoritmo, es fundamental para la ciencia de la computación, puesto que un computador no puede ejecutar un problema que no tenga una solución algorítmica. Así, cualquier limitación de las capacidades de los procesos computacionales constituye también una limitación de las capacidades de la computadora [1].

. Evaluar la eficiencia de los algoritmos tiene mucho que ver con evaluar la complejidad de los mismos. Aunque esta labor pueda resultar ardua, compensa el hecho de obtener programas con alta garantía de corrección y la satisfacción intelectual de haberlos construido bien desde el principio, sin esperar el veredicto del sistema computacional.

El tema de la computabilidad tiene mucho que ver con la búsqueda de las estructuras que se requieren en un lenguaje de programación, de tal forma que se asegure que un programa expresado en dicho lenguaje pueda resolver cualquier problema que tenga una solución algorítmica [2], por ende la Teoría de la Complejidad Computacional es la parte de la teoría de la computación que estudia los recursos requeridos durante el cálculo para resolver un problema.

Los recursos comúnmente estudiados son el tiempo (número de pasos de ejecución de un algoritmo para resolver un problema) y el espacio (cantidad de memoria utilizada para resolver un problema). Un algoritmo que resuelve un problema pero que tarda mucho en hacerlo, difícilmente será de utilidad. Igualmente un algoritmo que necesite un gigabyte de memoria no será probablemente utilizado. A estos recursos se les puede añadir otros, tales como el número de procesadores necesarios para resolver el problema en paralelo.

Debido a que cada lenguaje es único, puede existir entre lenguajes compilados e interpretados:

COMPILADOS

Como la palabra lo dice un lenguaje que es compilado necesita un compilador si o si, este compilador podríamos decir que es el que se encarga de entender todo lo que estamos escribiendo. Los lenguajes que necesitan de un compilador trabajan de la siguiente forma: Se escribe un archivo y normalmente con una extensión dependiendo del lenguaje, por ejemplo: archivo.c, archivo.java, etc. Este archivo necesita parar por el compilador, el compilador se encargará de leer el archivo, las cantidades de veces que se lee el archivo varía.

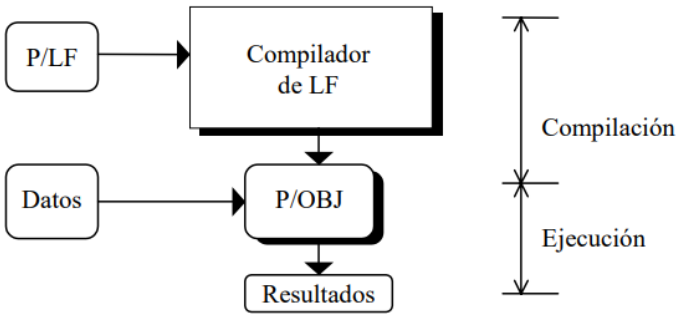

Figura 1: Esquema general de un compilador

INTERPRETADOS

El intérprete podemos decir que tiene un proceso mucho más sencillo, el intérprete lo que hace es leer el archivo y hacer la traducción(interpretar) lo que entiende, normalmente lo hace línea por línea. Con esto podemos decir que se comporta como un intérprete en verdad, las personas que están durante la transmisión de un evento deportivo, un debate o una entrega de premios y que escucha lo que dice la persona y va interpretando conforme lo va entiendo. Además los intérpretes no generan archivos para almacenar el resultado de lo que se traduce. Como podemos leer es mucho más sencillo de describir.

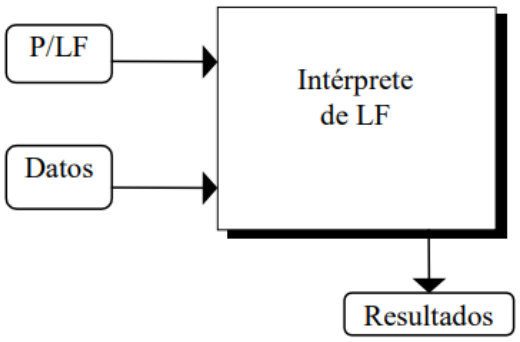

Figura 2: Esquema general de un intérprete

CAPÍTULO II

DESARROLLO DEL TEMA

¿Qué lenguajes de programación son los más usados?

Hay muchas preguntas que surgen cuando se intenta revisar el uso eficiente de energía de un lenguaje de programación. Sin embargo, comparar los lenguajes es difícil. El rendimiento de un lenguaje puede ser fácilmente mejorado sólo con la calidad del compilador o máquina virtual. El optimizar el código fuente es tan importante como las librerías mejoradas

Utilizando the Computer Benchmarks Game[3], el equipo de investigadores probaron varios lenguajes compilando/ejecutando dichos programas utilizando los mejores compiladores, máquinas virtuales, intérpretes y librerías. Luego analizaron el rendimiento de las diferentes implementaciones considerando tres variables:

- Tiempo de ejecución
- Consumo de memoria
- Consumo de energía

Dentro de los lenguajes llevados analizar estas pruebas se eligieron los más usados tanto en desarrolladores así como empresas.

TIEMPO DE EJECUCIÓN

Se conoce como tiempo de ejecución (Runtime) en programación al intervalo de tiempo que va desde que el sistema operativo comienza a ejecutar las instrucciones de un determinado programa, hasta que finaliza la ejecución del mismo, ya sea porque el programa concluyó exitosamente o porque fue finalizado por el sistema operativo a causa de un fallo en tiempo de ejecución.

El hecho que el tiempo de ejecución dependa de la entrada indica que el tiempo de ejecución de un programa debe estar definido como una función de los datos de entrada, que puede especificarse con el “tamaño” de la entrada. Así, por ejemplo, en un programa de

ordenamiento el tamaño natural de medida para la entrada es el número de elementos a ordenar. Es usual indicar con T(n) el tiempo de ejecución de un programa para una entrada de tamaño n. Las unidades de T(n) pueden dejarse sin especificar, pero puede pensarse que es el número de instrucciones ejecutadas en un ordenador ideal.

Las pruebas hechas de tiempo entre distintos lenguajes de programación por excelencia entre los desarrolladores, arrojaron los siguientes resultados.

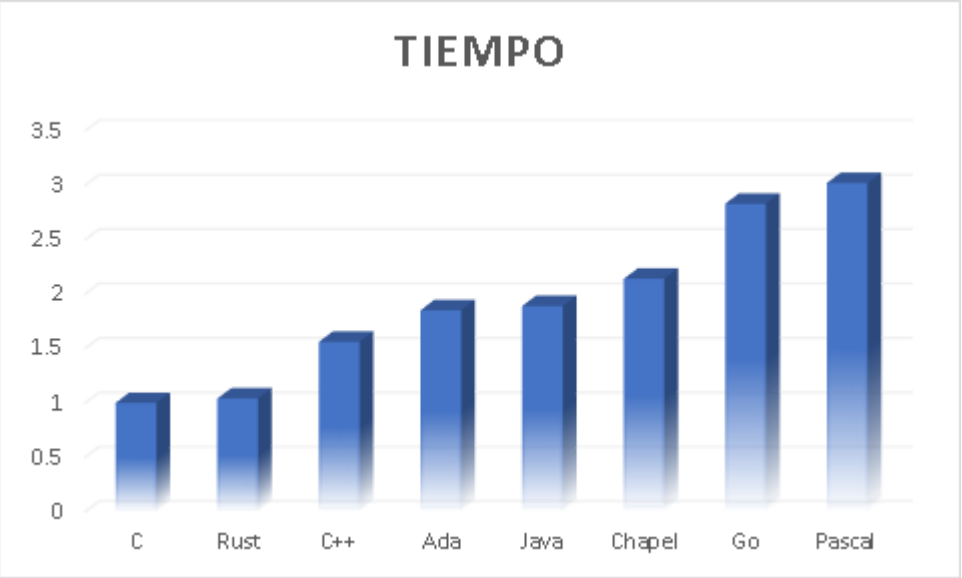

Figura 3: Esquema de tiempo de ejecución de LP

Como podemos observar en la figura 3, dentro de los lenguajes de programación mas usados, el lenguaje de programación C es el que ha tenido mejores resultados en cuando al tiempo , ya que al ser un lenguaje de programación compilado, es más fácil para este poder analizar diferentes algoritmos con un tiempo inmediato al devolver el resultado.

CONSUMO DE MEMORIA

Nuestro sistema operativo utiliza la memoria para almacenar los datos y archivos de los programas que utilizamos para poder acceder a ellos de una manera más rápida, por eso la administración de memoria de una computadora es una tarea fundamental debido a que la cantidad de memoria es limitada.El sistema operativo es el encargado de administrar la memoria del sistema y compartirla entre distintos usuarios y/o aplicaciones.El RTS (Run Time System) de un lenguaje de programación administra la memoria para cada programa en ejecución.

La ejecución de un programa requiere que diversos elementos se almacenan en la memoria: Código del programa (instrucciones) Datos Permanentes Temporales Direcciones para controlar de flujo del ejecución del programa

La asignación de memoria para algunos elementos fijos del programa que es controlada por el compilador se le llama asignación de memoria estática.A la asignación y posible recuperación de memoria durante la ejecución de un programa y bajo su control, se le llama asignación de memoria dinámica.

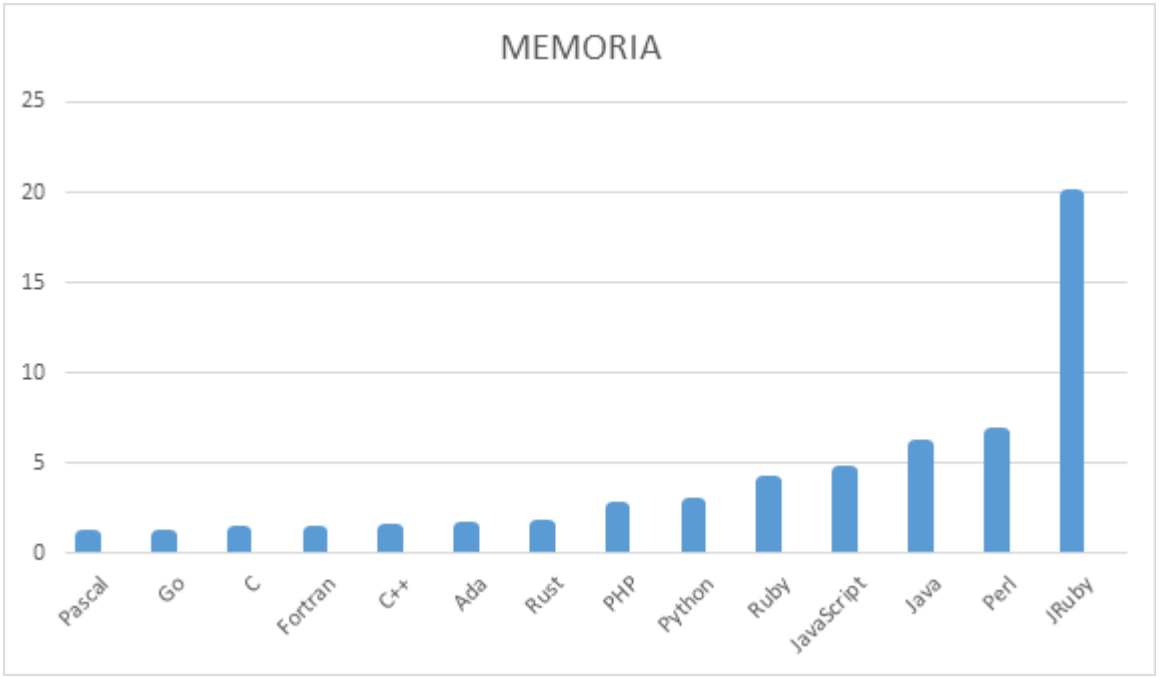

Figura 4: Esquema de consumo de Memoria en LP

CONSUMO DE ENERGÍA

Los dispositivos móviles e integrados actuales ejecutan programas escritos con diferentes lenguajes de programación. Por esta razón, también es importante considerar la eficiencia energética de un software escrito con un lenguaje de programación particular. Por eso es importante analizar el algoritmo en eficiencia energética , y su implementación eficiencia energética porque hoy en día sólo conocemos el impacto en términos de uso de memoria y tiempo de ejecución.

El siguiente diagrama de barras muestra la distribución de los lenguajes de programación más usados en cuanto a consumo energético que no solo van a depender del tiempo de compilación, sino también del consumo de memoria.

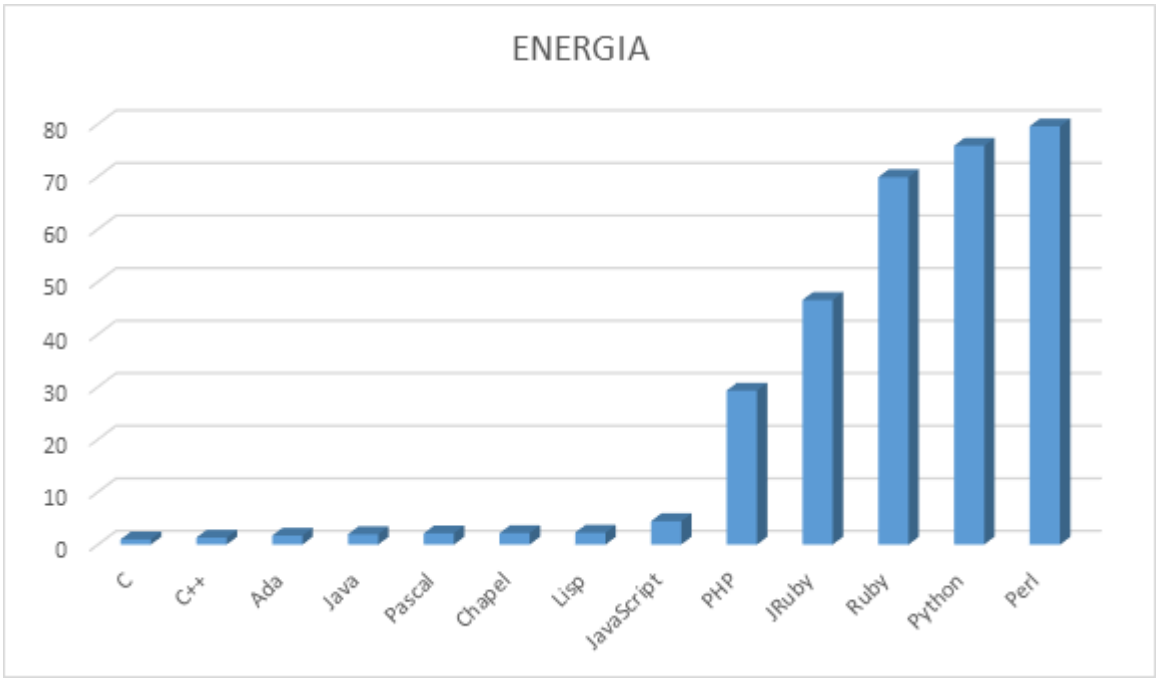

Figura 5: Esquema de consumo energético en LP

Como resultado los lenguajes C, C++ y Ada son los más eficientes energéticamente, aunque no necesariamente los que menos memoria consumen. Además el paradigma de programación utilizado influye directamente en la eficiencia energética, siendo los más eficientes, los lenguajes imperativos, los orientados a objetos, los funcionales y finalmente los lenguajes interpretados. Pero lo más importante, en el consumo de energía lo que más influye, es el uso del procesador.

CONCLUSIONES

Gran parte de la eficiencia está determinada por el tiempo de ejecución, es decir, la complejidad computacional de la solución, aunque aproximadamente una mínima parte está determinada por una patrón de consumo de energía.

El análisis en los distintos lenguajes proporciona la información básica para elegir una implementación de clasificación para hacer más eficiente un programa o acomodarnos más a un lenguaje.

¿Se puede elegir un lenguaje de programación basado en la energía, tiempo y memoria? Si. “C” es el lenguaje que más destaca analizando los 3 campos.

Pero no solo debemos de tener una perspectiva de los lenguajes, unos cumplen una función mejor que otras, en cuanto a paradigmas como POO con java en el apogeo por su eficiencia energética y en rapidez.

Como trabajo adicional se planea investigar con más detalle los factores de hardware que afectan el consumo de energía, memoria y complejidad ,cómo están relacionados con distintos lenguajes de programación..

BIBLIOGRAFÍA

- [1] BROOKSHEAR J. Glean. Teoría de la computación. Addisson Wesley Iberoamericana Wilmington Delaware 1993.
- [2] PeñA M., Ricardo. Diseño de programas. Formalismo y abstracción. Prentice Hall Madrid 1998.
- [3] Colaboradores de Wikipedia. El juego de comparativas del lenguaje informático Wikipedia, La enciclopedia libre, 10 de octubre de 2021. Disponible en: https://en.wikipedia.org/wiki/The_Computer_Language_Benchmarks_Game
- Jose D. L.*Lenguajes compilados y lenguajes interpretados. (2017).*ED Team .https://ed.team/blog/lenguajes-compilados-vs-lenguajes-interpretados

ANEXOS

- [1] Mohammad Rashid, Luca Ardito, Marco Torchiano. Energy Consumption Analysis of Algorithms Implementations. 2015.