

Elaboración de sistemas multifuncionales mediante el cómputo concurrente y paralelo

Alexis Paleta Osorio ¹, Dra. Meliza Contreras González ¹

¹ Facultad de ciencias de la computación – BUAP

Resumen

El correcto manejo de los recursos computacionales es una parte fundamental para la construcción de sistemas complejos, pues las acciones que desempeñan son cada vez más demandantes del poder que ofrece el hardware de una máquina. Es mediante el cómputo concurrente y paralelo que se puede llegar a la optimización de la utilización de dichos recursos para lograr los cometidos de cada sistema, aprovechando al máximo los componentes de la computadora que se tengan para poder trabajar, y así sobrellevar las limitantes que se lleguen a presentar. Otro aspecto por resaltar es la constante búsqueda de maneras con las que reducir el tiempo de ejecución de los programas construidos, la eficiencia siempre ha sido objetivo de la computación. A través de la elaboración de un sistema multifuncional se mostrará la importancia de la correcta implementación de la programación concurrente y paralela a la hora de construir programas computacionales para obtener los mejores comportamientos posibles de cada funcionalidad del sistema, mejorando y comparando los tiempos de ejecución de cada uno de estos, para la evaluación y comprensión de la importancia de este tipo de programación, mostrando ventajas que esta trae consigo en problemas de diferentes ámbitos.

Introducción

El comprender los conceptos de concurrencia y paralelismo, es una parte esencial para poder construir sistemas eficientes, es muy común que exista confusión entre estos dos conceptos debido a las similitudes que existen, sin embargo, es necesario el diferenciarlos, ya que la aplicación de estos depende del problema que esté resolviendo, no todo proceso puede ser paralelizado o enfocado a realizarse de manera concurrente. Primeramente, se identifican las formas en las que se puede diseñar la ejecución de los programas, siendo la secuencial, concurrente y la paralela, definiendo cada una de estas:

Programación secuencial, se refiere a la realización de tareas que “se ejecutan como una carrera de relevos, donde una no comienza hasta que su predecesora no haya finalizado” [1], siendo esta la manera más sencilla de realizar los procesos, el orden se establece desde que se escribe el código.

Programación concurrente, la RAE define a la concurrencia como “coincidencia, concurso simultáneo de varias circunstancias.” [2]. Es decir, existe un solapamiento entre las acciones a realizar, se entiende al aplicar este concepto a procesos computacionales como “dos procesos serán concurrentes cuando la primera instrucción de uno de ellos se ejecuta después de la primera instrucción del otro y antes de la última” [3].

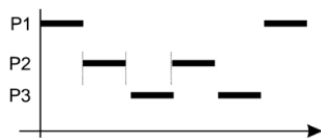


Figura 1) Ejemplo de ejecución de 3 procesos de forma concurrente [3]

La concurrencia da la impresión de que las tareas se ejecutan en un mismo tiempo, sin embargo, estas se van turnando. Un concepto fundamental en la concurrencia es el hecho de que exista la competencia o cooperación entre los procesos que se estén ejecutando, dichos conceptos traen consigo un grado de complejidad que debe comprenderse para evitar problemas con la realización de tareas.

La programación paralela hace referencia a la ejecución simultánea de tareas, lo que trae consigo un beneficio directo en cuanto al tiempo de ejecución, para poder usar el paralelismo es necesario contar con más de un procesador, por lo que, en el caso específico de este enfoque, existen requerimientos en el hardware [4].

Para establecer una clara diferencia entre la concurrencia y el paralelismo, una forma simple de verlo es: La concurrencia trata sobre manejar muchas cosas al mismo tiempo, mientras que el paralelismo trata sobre hacer muchas cosas al mismo tiempo. Claramente están relacionados, la concurrencia es más que nada sobre la estructura de una solución a un problema que puede, pero no necesariamente, ser paralela [5].

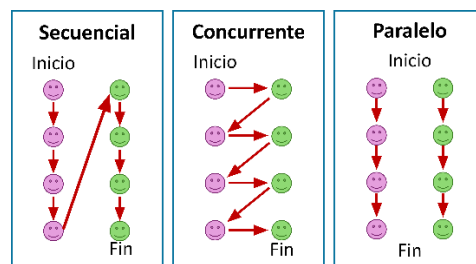


Figura 2) Orden de ejecuciones según su enfoque

Beneficios de la concurrencia y paralelismo

Entre las utilidades que existen de la aplicación de la programación concurrente y paralela, se destacan dos: la velocidad de ejecución y la solución de problemas que desde su concepción dan pie a un comportamiento concurrente.

Velocidad de ejecución: Cuando se aplica el concepto de concurrencia se está dividiendo un problema en distintos fragmentos, por lo que, en vez de tratar de resolverlo en un mismo momento, el hacer esta división permite que sea más sencillo el proceso tanto en su comprensión, como en su resolución. Pasando a su implementación directa en la computación, debido a la fragmentación del problema, si se posee un sistema con más de un procesador, “existe la posibilidad de asignar un proceso a cada procesador de tal forma que el programa se ejecuta de una forma más rápida” [3], pues en cada procesador se realizarían tareas independientes que no obstaculizan con el resto, permitiendo así que su finalización de paso a realizar más tareas de manera simultánea.

Problemas inherentemente concurrentes: Cuando se quiera proveer de un servicio, la situación es obligatoriamente concurrente, debido a que, por ejemplo, si en un restaurante se necesita atender a los clientes que están constantemente llegando, si se tuviera un enfoque secuencial, estaríamos diciendo que no podemos atender a un cliente hasta que este haya sido completamente atendido y que haya salido del local, la realidad es en el momento que llega un cliente, bien se le puede tomar la orden, pero inmediatamente después se le puede estar llevando la comida a un cliente distinto,

se podría estar cobrando a otro y después se regresa al cliente que se mencionó al inicio. Este constante cambio entre que cliente es atendido y que tarea se está realizando, tienen necesariamente un enfoque concurrente, con claro la posibilidad de ser paralelizado, pues al añadir empleados, se podrían atender clientes al mismo tiempo.

Riesgos y problemas de la programación concurrente y paralela

Si bien la idea de la división de tareas es muy atractiva en primera instancia, existen problemáticas que pueden llegar a surgir, como lo son las condiciones de carrera o los interbloqueos.

“Una condición de carrera es un comportamiento del software en el cual la salida depende de un orden de ejecución de eventos que no se encuentran bajo control y que puede provocar resultados incorrectos” [6]. Esto mismo puede llevar a un interbloqueo o deadlock, que son provocados debido a que los procesos están enviando y recibiendo datos, en caso de que los dos se queden esperando la llegada de información ninguno podrá continuar, ya que para hacerlo necesitan dicha información [7].

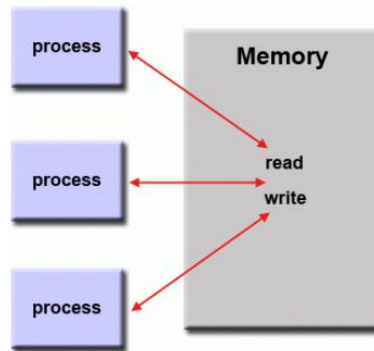


Figura 3) Procesos que leen y modifican variables compartidas [8]

Para resolver estos problemas existen los métodos de sincronización, ya que como la ejecución concurrente es no determinista (no se puede saber el orden de ejecución que adoptaran los procesos), será con estos métodos que se cuida el acceso a los recursos compartidos, entre ellos se encuentran [3]:

- Semáforos
- Monitores
- Definición y cuidado de región crítica

La división de los problemas para aplicar la concurrencia no es una tarea sencilla de abstraer, ya que, si bien existen problemas con una sencilla aplicación de concurrencia, otros dificultan este tipo de análisis, lo que puede provocar el hacer una mala fragmentación de los procesos de un problema, esto lleva al concepto de granularidad que se define como una medida cualitativa para medir el rango de comunicación. Se divide en la granularidad fina que se refiere a pequeños trabajos computacionales, pero con alta comunicación entre los procesos, y la de grano grueso, que posee trabajos con alta demanda computacional pero menor comunicación entre ellos [8]. El crear muchos procesos usando la idea de la granularidad fina puede ser perjudicial debido a la excesiva e innecesaria fragmentación del problema, perjudicando al rendimiento debido a la demanda de

creación de procesos, por otro lado, el usar indiscriminadamente la granularidad gruesa podría no estar aprovechando y limitando la oportunidad de mejora en la ejecución y eficiencia.

Sistema propuesto: Describiendo sus funciones

El sistema multifuncional creado para esta investigación tiene la arquitectura de una aplicación cliente – servidor, múltiples dispositivos pueden conectarse de manera simultánea para poder realizar diferentes solicitudes y que el servidor responda a cada uno de ellos de manera independiente, atendiendo las necesidades específicas que le fueron notificadas por cada cliente. El lenguaje de programación en el que fue desarrollado fue Java, en su versión 18, pero permitiendo conectarse con clientes que tengan versiones de al menos java 8. Las ejecuciones fueron realizadas en una computadora con un procesador AMD Ryzen 2700.

La vista del cliente presenta una tabla que contiene en su columna izquierda el nombre del programa/funcionalidad que se ejecutó por el cliente, y en la columna derecha, el tiempo de ejecución en milisegundos que le llevó al servidor poder realizar los cálculos necesarios de la operación especificada.

El servidor no tiene una vista, pues se encarga de únicamente atender a los clientes a través de sockets. Cada vez que un cliente realice una petición al servidor se crea un socket, permitiendo así que la carga en el servidor sea la “justa”, ya que un cliente va a estar únicamente conectado el tiempo necesario para poder realizar la petición y recibir la información necesaria.

El sistema contiene la siguiente lista de funciones:

- Cálculo y dibujo de un fractal (Mandelbrot), con ejecuciones de manera secuencial, concurrente y otra paralela, la lista de tiempos de cada ejecución se guarda en la tabla que está en la ventana del cliente, además de que el cliente recibe la imagen del fractal y la muestra en una ventana. Para ejecutar la función del fractal existen tres botones, uno cada tipo de ejecución.
- Juego de la vida (de John Horton Conway), se ejecuta con un botón específico que lleva el nombre de la función, el cliente abre una nueva ventana en la que recibirá de manera continua los valores necesarios para dibujar en esta a los miembros de la población. Si un cliente está visualizando esta ventana y otro cliente ejecuta esta misma función, entonces se verá reflejada esta “aparición” del nuevo cliente en la ventana del cliente original. El servidor es capaz de soportar la ejecución simultánea del juego de la vida en varios clientes y además enviar información de las demás funcionalidades, esto es posible debido a que la lógica del servidor se separa en dos partes, siendo una la que se dedica por completo al juego de la vida debido a su alta demanda de información solicitada, y la otra parte dedicada a realizar el resto de las funcionalidades del sistema.
- Realización de operaciones en matrices de tamaño $N \times N$, esta función permite operar una suma a través de todas las posiciones de una matriz de 10000×10000 , 15000×15000 y 20000×20000 , además de poder elegir si los cálculos se realizarán de manera secuencial, concurrente o paralela para cada uno de los tres tamaños disponibles a selección del usuario. Los resultados de los tiempos de ejecución se almacenan en la tabla de la ventana principal del cliente, junto con el nombre y especificación del tamaño de la matriz calculada.

- Petición a base de datos, permite que múltiples clientes puedan solicitar una lista de alumnos, que serán mostrados en una nueva ventana a través de una tabla, mostrando la matrícula, el nombre y el promedio de todos los alumnos registrados.
- Cálculo de N decimales de Pi, se le permite al usuario seleccionar la cantidad de dígitos de Pi que desea calcular, permitiendo mil, cinco mil y hasta diez mil decimales, se tienen tres formas de realizar los cálculos, teniendo un enfoque secuencial, concurrente y uno paralelo, los resultados de los tiempos de ejecución junto con el nombre de la función específica y el número de decimales deseado se guardan en la tabla de la vista principal del cliente.

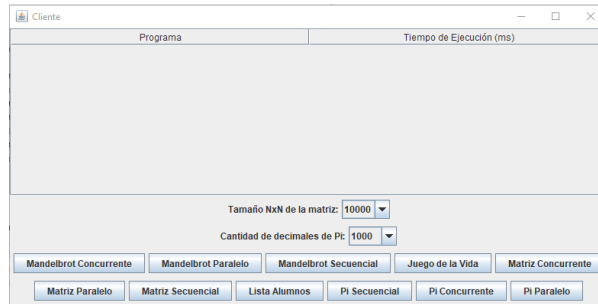


Figura 4) Ventana principal del cliente recién iniciado

Para poder tener los resultados más “justos” posibles, en los enfoques concurrentes y paralelos, la división de tareas se hizo en las mismas cantidades, es decir, si la ejecución concurrente crea 4 hilos, la paralela realiza 4 divisiones. De esta manera los resultados pertenecen a situaciones equivalentes, permitiendo así una mejor interpretación y comparación de los tiempos obtenidos.

Fractal Mandelbrot

Se realizaron nueve ejecuciones en total, tres por cada implementación. Los tiempos obtenidos por la ejecución secuencial son considerablemente más altos que el resto de las ejecuciones, pues la manera en la que se realiza el coloreado de la ventana la realiza solo un hilo. Para el caso de las ejecuciones concurrente y paralela, la ventana se divide en ocho partes, haciendo el tiempo se reduzca de manera considerable al compararla con la secuencial, la versión paralela resulta ser la más eficiente, dando como resultado las ejecuciones más veloces, siendo la ejecución paralela la segunda mejor, pero no quedándose tan atrás de la paralela.

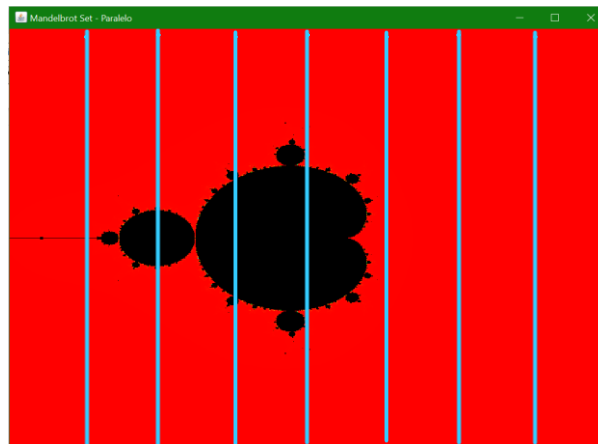


Figura 5) Imagen generada del fractal, junto con las divisiones realizadas a la ventana de las ejecuciones concurrente y paralela

Cliente	
Programa	Tiempo de Ejecución (ms)
mandelbrot_concurrente	349
mandelbrot_concurrente	336
mandelbrot_secuencial	422
mandelbrot_secuencial	681
mandelbrot_paralelo	214
mandelbrot_paralelo	212
mandelbrot_secuencial	719
mandelbrot_paralelo	207
mandelbrot_concurrente	338

Figura 6) Tiempos de ejecución para las diferentes implementaciones del dibujado y cálculo del fractal

Juego de la vida

Al momento de presionar el botón de la función del juego de la vida se crea una nueva ventana, que estará en constante comunicación con el servidor, de manera indefinida hasta que el usuario cierre esta nueva ventana, la mejor forma de apreciar esta funcionalidad es cuando se ha entrado varias veces en ella, ya que en cada entrada se crea un nuevo “miembro” del juego, permitiendo interacciones de lo más interesantes. Debido a que el servidor contiene en dos hilos distintos las lógicas del manejo de las peticiones, se pueden realizar peticiones aún mientras el Juego de la Vida está en plena ejecución.



Figura 7) Ventana desplegada al ejecutar el Juego de la Vida después de varias entradas

Operaciones con Matrices de N x N

Se realizan las ejecuciones de la suma de valores contenidos en una matriz de un tamaño especificado por el usuario, las ejecuciones paralelas y concurrentes se realizan en cuatro divisiones para realizar los cálculos, como resultado la versión paralela obtiene los tiempos de ejecución más bajos, siendo la versión secuencial nuevamente la que presenta tiempos de en promedio casi el doble que la versión concurrente. Nótese que, para el caso de las últimas 6 ejecuciones, los tiempos son bastante diferentes a pesar de que los programas ejecutados fueron los mismos, esto es debido al como el propio sistema operativo está realizando la asignación de tareas, además de que depende también de si se tienen más aplicaciones ejecutándose mientras se realizan estas pruebas.

Cliente	
Programa	Tiempo de Ejecución (ms)
matriz_concurrente_10000	76
matriz_paralelo_10000	37
matriz_secuencial_10000	113
matriz_concurrente_15000	141
matriz_paralelo_15000	101
matriz_secuencial_15000	225
matriz_concurrente_20000	50
matriz_paralelo_20000	59
matriz_secuencial_20000	134
matriz_concurrente_20000	261
matriz_paralelo_20000	111
matriz_secuencial_20000	424

Figura 8) Tiempos de ejecución de la función de las matrices en sus tres versiones

Petición a base de datos

La idea detrás de esta funcionalidad a pesar de su simpleza es el destacar la importancia de otorgar a los clientes la oportunidad de poder acceder a un recurso compartido de manera controlada, en la que con el uso de un banco de conexiones y la ejecución en hilos distintos, se le permiten a varios dispositivos realizar solicitudes al mismo tiempo, ya que si no se tuviese este enfoque se le estaría negando al acceso a los clientes que lleguen después de otros, provocando un problema de retardo para atender a las solicitudes entrantes y creando un sistema que no es eficiente para poder satisfacer las necesidades de múltiples usuarios. La función creada permite la consulta continua de los datos de diferentes clientes al mismo tiempo, atendiéndolos de manera eficiente.

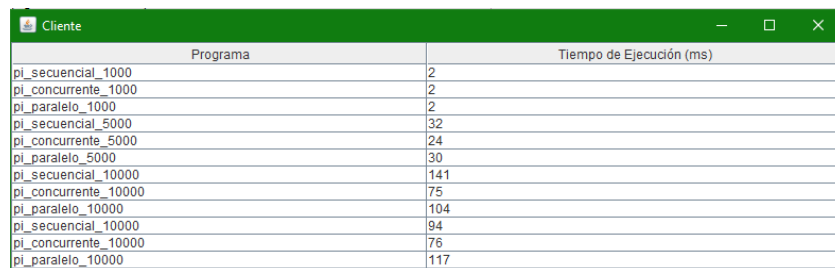


Matricula	Nombre	Promedio
1	Alexis Paleta	9.8
2	Juan Luna	9.1
3	Pedro Paredes	8.0
4	Rosa Munguia	8.9
5	Sergio Rivera	9.2
6	Luz Puentes	9.4
7	Diego Salazar	8.5
8	Karol Claudio	9.3

Figura 9) Listado de alumnos en una nueva ventana

Cálculo de N decimales de Pi

Se realizan las ejecuciones de las tres versiones del cálculo de decimales de Pi, siendo estas las versiones con un enfoque secuencia, concurrente y paralelo. Los resultados de los tiempos en este caso resultan en unos diferentes a los que en un inicio se pueden pensar, ya que la que menores tiempos de ejecución tiene es la versión concurrente, seguida de la paralela y luego la secuencial, incluso hay una serie de ejecuciones que muestra que la ejecución secuencial terminó siendo más rápida que la paralela, eso puede deberse a varios motivos, desde el hardware que se está utilizando, entre mejor se tenga mayor oportunidad de mejora existe para las versiones paralelas, debido a la velocidad y cantidad de procesadores que se posean, en caso de que el equipo de cómputo no posea las mejores especificaciones bien se pueden obtener resultados peores al intentar paralelizar las tareas. En este caso debido a que la división de tareas se limitó a dos, parece ser que a la versión paralela le beneficiaría el tener más divisiones, ya que su enfoque es el de dividir las tareas hasta que se llegue un punto de que cada división sea lo más “procesable” posible, reduciendo la carga de trabajo que se debe tener en cada división, en esta función al tener en cada tarea una carga de trabajo considerable, la comunicación necesaria, junto con la cantidad de operaciones que se deben realizar provoca que la versión paralela no sea la mejor de las presentadas, pero claro realizando más divisiones es muy posible que el manejo mejores de tal manera que los tiempos se vean mucho más reducidos.



Programa	Tiempo de Ejecución (ms)
pi_secuencial_1000	2
pi_concurrente_1000	2
pi_paralelo_1000	2
pi_secuencial_5000	32
pi_concurrente_5000	24
pi_paralelo_5000	30
pi_secuencial_10000	141
pi_concurrente_10000	75
pi_paralelo_10000	104
pi_secuencial_100000	94
pi_concurrente_100000	76
pi_paralelo_100000	117

Figura 10) Tiempos de ejecución para el cálculo de N decimales de Pi en sus tres versiones

Conclusión

Realizar sistemas eficientes implica identificar el enfoque correcto que debe seguir el problema, plantearse preguntas como ¿varias personas realizan tareas al mismo tiempo?, ¿con qué poder computacional cuenta el equipo en el que deseo implementar el sistema?, ¿el tiempo de mejora es lo suficientemente valioso como para cambiar la implementación de ciertas funciones?, ¿cómo puedo mejorar mi sistema?, es necesario para poder tener una idea de las necesidades a resolver. La concurrencia y paralelismo traen muchísimos beneficios, sin embargo el poder implementarlos correctamente implica comprender bien las interacciones con las que se realizan las tareas implicadas en un problema, resolviendo situaciones que no son tan sencillas de ver a comparación de una implementación secuencial, en la elaboración del sistema presentado se presentaron algunos inconvenientes a la hora de crear las funciones, pero al analizar detenidamente la problemática y proponiendo soluciones óptimas se llegó a buenos resultados. El aplicar el cómputo concurrente y paralelo se ha convertido en una necesidad fundamental para poder realizar muchas tareas de hoy en día, por lo que es muy importante el conocer su aplicación, implementación y problemas que pueden surgir al implementarlo, de no contar con el ni si quiera se podrían realizar acciones tan comunes en nuestro día a día como tener una aplicación de música abierta mientras trabajamos en un documento, o se daría el caso en el que se tenga que esperar a que todos los usuarios terminen de realizar actividades en una aplicación web para que se nos otorgue acceso, a través de este trabajo se revisaron diversas implementaciones para poder ejemplificar las oportunidades que ofrece esta forma de resolver problemas y poder así crear sistemas multifuncionales eficientes.

Agradecimientos

A la Dra. Meliza por el apoyo otorgado a lo largo del desarrollo de este trabajo.

Referencias

- [1] MENDOZA, D. E., ANGEL, J. C., BARRIOS, W. J., LEGUIZAMON, G., & GARCIA, R. G. (2017). Análisis comparativo de los costos por actividades: diseño secuencial vs diseño concurrente. Estudio de caso para una empresa productora de perfiles de aluminio. *Revista Espacios*, 38(24).
- [2] REAL ACADEMIA ESPAÑOLA. (s.f.). Concurrencia. En el *Diccionario de la lengua española*. Recuperado el 11 de octubre, 2024, en <https://dle.rae.es/concurrencia>
- [3] Palma Méndez, J. T., GARRIDO CARRERA, M. C., Sánchez Figueroa, F., & Quesada Arencibia, A. (2003). Programación concurrente. Ediciones Paraninfo, SA.
- [4] Terrell, R. (2018). *Concurrency in. NET: Modern patterns of concurrent and parallel programming*. Simon and Schuster.
- [5] Rob Pike. (2012). Concurrency is not Parallelism. Recuperado el 11 de octubre, 2024, en <https://go.dev/talks/2012/waza.slide#1>
- [6] Guillermo Román Díez. (2019). Condiciones de Carrera. Universidad Politécnica de Madrid. Recuperado el 12 de octubre, 2024, en https://babel.upm.es/teaching/concurrencia/material/slides/groman/CC_CondCarrera.pdf
- [7] Wilkinson, B., & Allen, M. (2005). *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers* (2da ed.). Prentice Hall.
- [8] Barney, B. (2020). Introduction to parallel computing. Lawrence Livermore National Laboratory. Recuperado el 19 de octubre, 2024, disponible en: https://computing.llnl.gov/tutorials/parallel_comp/