Procesos, hilos y corrutinas

* Un caso en el que usarías procesos para resolver un problema y por qué Una aplicación que envía correos electrónicos masivos a los usuarios, en esta aplicación podemos usar el proceso para enviar correos electrónicos masivos en diferentes procesos sin bloquear el hilo principal del nodo js. Esto con el fin de evitar la sobresaturación del hilo principal de nodejs
* Un caso en el que usarías threads para resolver un problema y por qué En procesamiento de imagenes o visualización de stream de una camara, ya que son procesos que consumen un alto recurso de cpu y al tener varias fuentes en 1 solo hilo su nivel de procesamiento seria elevado a demas de poder ocacionar una perdia del stream Se usaria un hilo por cada visualizacion del stream de una camara para poder realizar procesamiento de video y optimizacion del stream en cada uno de los hilos y con ello generar una mejor calidad de video en el stream
* Un caso en el que usarías corrutinas para resolver un problema y por qué. Una corrutina se puede usar para ejecutar multiples request, ejecutandose en paralelo, al final de la corrutina se recopila las respuesta. esto permite optimizar los tiempos de ejecución de la aplicación.

Optimización de recursos del sistema operativo

* Se puede utilizar una cola de solicitudes para procesar en segundo plano, anexo a eso utilizara child\_process para poder aprovechar mejor los recursos de la cpu al dividir el proceso en varios subprocesos y asi poder manejar varias solicitudes en simultaneo, usaria una base de datos no relacional como mongo db por su escalabilidad horizontal, el cual permite el manejo de volumenes de datos de manera mas eficiente.

Análisis de complejidad

Primer Ejercicio

* Principalmente me enfocaria en el algoritmo D su complejidad O(n log n) lo que significa que su tiempo de ejecucion crece en tamaño del problema, es decir a medida que el volumen de datos aumenta el tiempo de ejecución pero a un ritmo mas lento que los otros algoritmos.
* Por otro lado descartaria el algorimo C de complejidad O(2^n) el que difiere que el tiempo de ejecución crecera exponencial con el tamaña del volumen de datos que valla a procesar, esto lo hace ineficiente para el manejo de grandes volumenes de datos
* Los algoritmos A y B son cuadraticos y cubicos, lo que refiere que sus tiempos de ejecucion seran aceptables en procesamiento de mediamos volumenes de datos pero en el incremento de los volumenes de datos sus tiempos no seran aceptables.

Segundo Ejercicio

* AlfaDB: Esta base de datos seria adecuada para procesos que requieran una gran cantidad de lectura de datos pero un baja cantidad de escritura ya que su complejidad algorimica en el proceso de escritura hace que se demore escribiendo grandes volumenes de datos, pero su complejidad algorimica en lectura es baja lo que significa que podra leer datos rapidamente.

Un ejemplo para esta base de datos es una aplicacion para el control de inventario de una tienda en el cual su nivel de lectura es mayor el nivel de escritura. ya que apesar de que constantemente este agregando elementos al inventarios, son mayor las solicitudes de lectura para indicar el stock que existe en la tienda

* BetaDB: Esta base de datos seria adecuada para casos en los cuales sea necesario un nivel alto de escritura como de lectura, ya que su complejidad algoritmica es baja,a unque puede tardar mas que AlfaDB en escribir datos sigue siendo bastante eficiente.

Un ejemplo para esta base de datos puede ser una aplicacion de chat de una red social. los mensajes de los usuarios son almacenados en la base de datos y cada ves que un usuario abre un chat este requiere que se lean todos los mensajes de esta conversación.