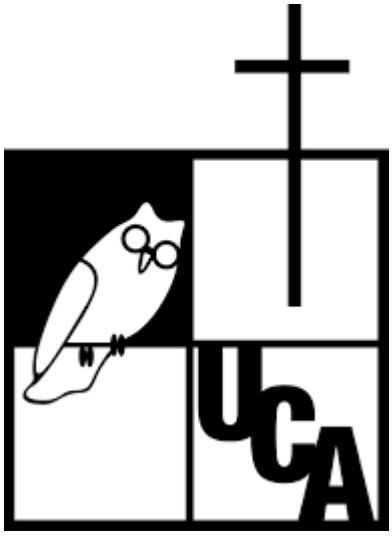
UNIVERSIDAD CENTROAMERICANAS

“JOSÉ SIMEÓN CAÑAS”



ARQUITECTURA DE COMPUTADORAS

FACULTAD DE INGENIERÍA Y ARQUITECTURA

INGENIERO:

JAVIER HERNANDEZ

INTEGRANTE:

JULIO EDGARDO FLORES GIRON 00081817

JULIO DE 2020

ANTIGUO CUSCATLÁN, EL SALVADOR, C.A.

En primer lugar, decidí escoger WebAssembly porque es una tecnología bastante innovadora y prometedora, ya que hace muchísimo mas alta la eficiencia de una aplicación web, si y solo si, esta se sabe utilizar y aplicar correctamente donde se necesite.

Por ejemplo, si se necesita realizar un cálculo matemático, probabilístico o cualquier otro proceso que implique números y además se requiere que sea aplicado para cada usuario dentro de la app, el realizarlo tomaría mucho tiempo y perdidas en recursos que quizás al inicio cuando el sistema es pequeño y no maneja data tan inmensa, les aseguro no será notado. ¿Pero qué pasa si el sistema crece y ahora la cantidad de información que se manipula ya se clasifica como BigData?

Se comenzará a notar el déficit en ciertas áreas de la app, por lo que, se necesitaría buscar soluciones que optimicen y agilicen las funciones y procesos dentro de la app… sería el momento indicado para utilizar WebAssembly para lograr que todo calculo que antes se realizaba dentro de la app ahora sea realizado aparte con WebAssembly y solo se comunique con él para mandar y recibir información.

Así que, para empezar…

¿Qué es WebAssembly?

Es un nuevo tipo de código que se puede ejecutar en los navegadores web modernos (un lenguaje de tipo ensamblador) de bajo nivel con un formato binario compacto que se ejecuta con un rendimiento casi nativo y proporciona lenguajes como C / C++ y Rust con un objetivo de compilación para que puedan ejecutarse en la web con mucha más eficiencia ya que utiliza una pila, está diseñado también para ejecutarse junto con JavaScript, lo que permite que ambos trabajen juntos y se logre identificar en que áreas necesita ser usado.

¿Cómo usarlo?

Puede ser usado de distintas maneras:

- WebAssembly.Global()
  Un WebAssembly.Global es un objeto que representa una instancia de variable global, accesible tanto desde JavaScript como importable/exportable a través de una o más WebAssembly.ModuleInstance. Esto permite la vinculación dinámica de múltiples módulos.

- WebAssembly.Module()
  Un WebAssembly.ModuleObject contiene código de WebAssembly sin estado que ya ha sido compilado por el navegador y puede compartirse de manera eficiente con los Trabajadores y crear instancias varias veces.

- WebAssembly.Instance()
  Un WebAssembly.InstanceObject es una instancia con estado y ejecutable de a Module.Instance, los objetos contienen todas las funciones de WebAssembly exportadas que permiten llamar al código de WebAssembly desde JavaScript.

Y así muchas más funcionalidades y áreas donde puede ser aplicado, incluso si se maneja una API. Acá un ejemplo matemático donde se necesitó utilizar WebAssembly: https://github.com/JasonWeathersby/WASMSobel

Conclusión

La idea que quiero dejar plantada aquí es que WebAssembly puede ser usada no solo para procesos que impliquen cálculos matemáticos sino para problemas que se tenga dentro de una aplicación web que necesite de optimización y alta eficiencia (claro esto dependerá del tipo de problema) pero para este caso que son cálculos numéricos, WebAssembly podría ser la solución.