**Asignación de bottlenecks con profiler + uso de copilot**

* **Introducción:**

Este documento describe el análisis, optimización y resultados del uso del profiler de Visual Studio junto con GitHub Copilot para mejorar el rendimiento de una operación pesada (multiplicación de matrices) en una aplicación ASP.NET Web Forms.

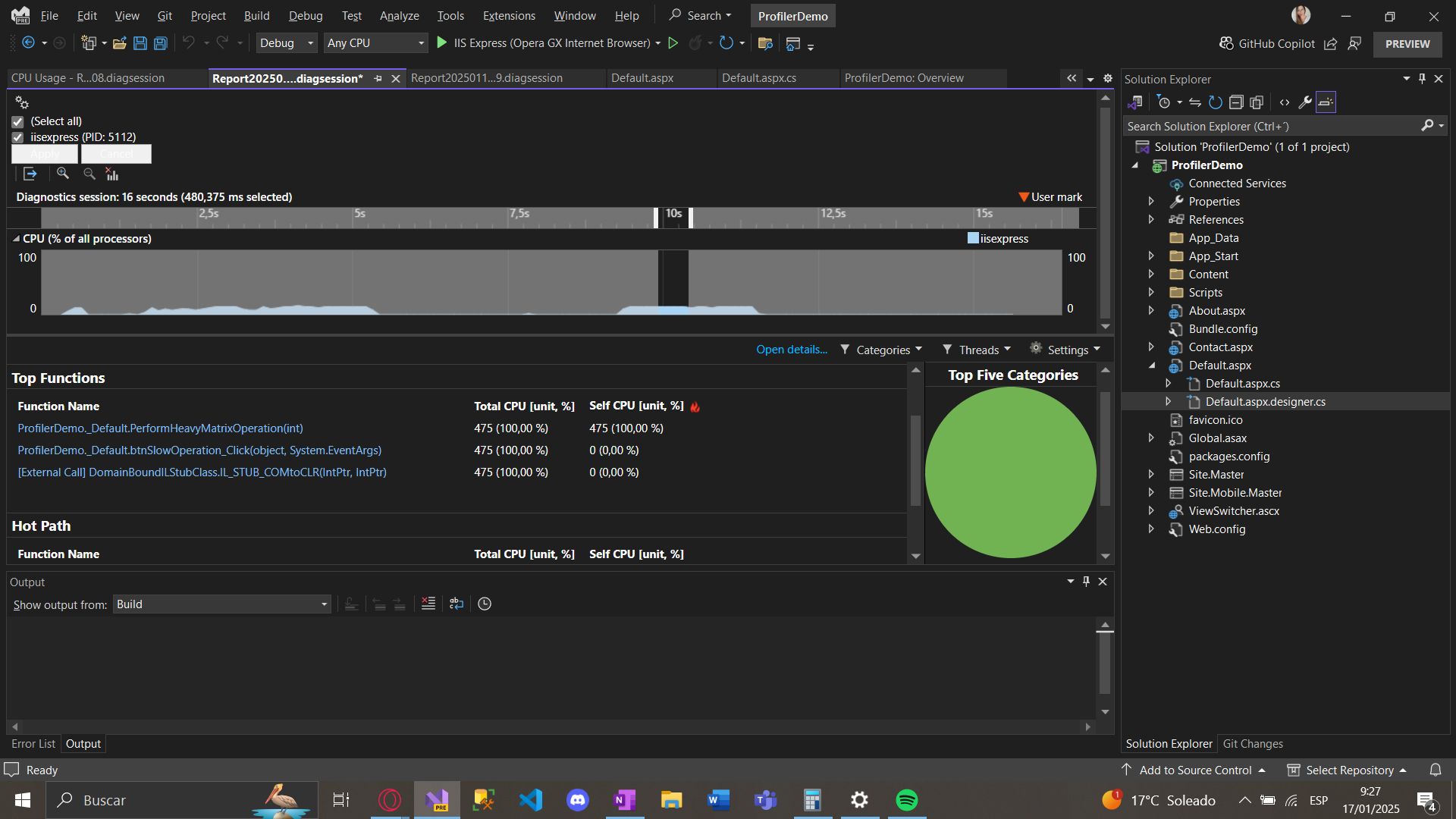
* **Descripción del problema:**

El proyecto original incluye un método *PerformHeavyMatrixOperation* que realiza una multiplicación de matrices de tamaño 500x500. Este método utiliza una implementación básica de bucles anidados con una complejidad O(n³).

El objetivo es optimizar el rendimiento de este método y reducir el tiempo de ejecución utilizando GitHub Copilot para explorar alternativas.

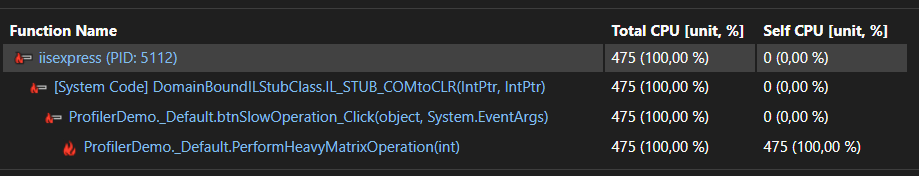
* **Análisis inicial:** 
  + Herramientas utilizadas:
* Visual Studio Profiler: para identificar cuellos de botella.
* GitHub Copilot: para sugerencias de optimización.
  + Resultados del Profiler:

El profiler muestra que el método *PerformHeavyMatrixOperation* consume el 99.48% del tiempo total de CPU durante la ejecución de la aplicación.



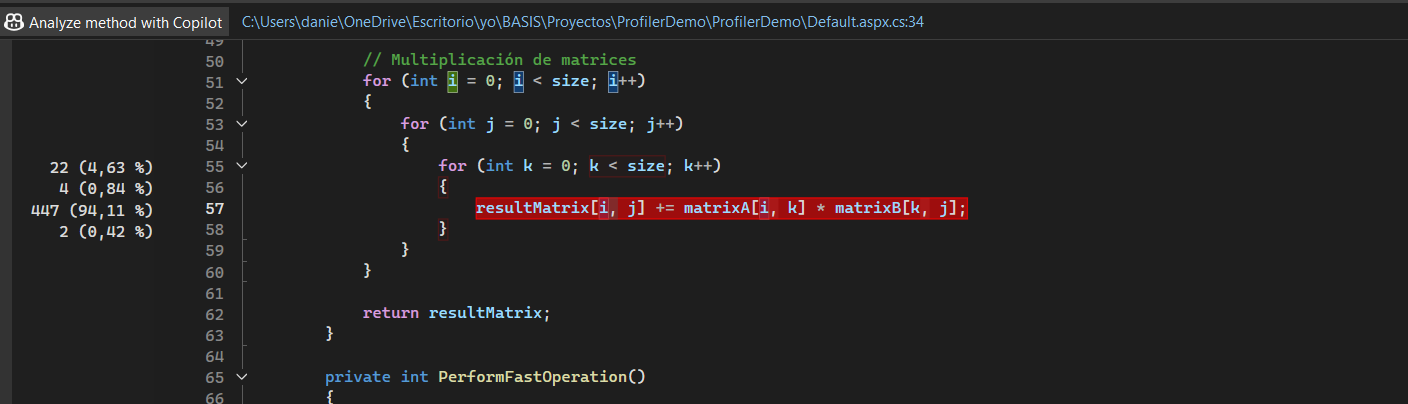
* **Proceso de optimización:**
  + Paso 1: identificación del problema

Profiler identificó que el cuello de botella se encuentra en el bucle de la multiplicación:

*Al hacer click en el método obtenemos más detalles*

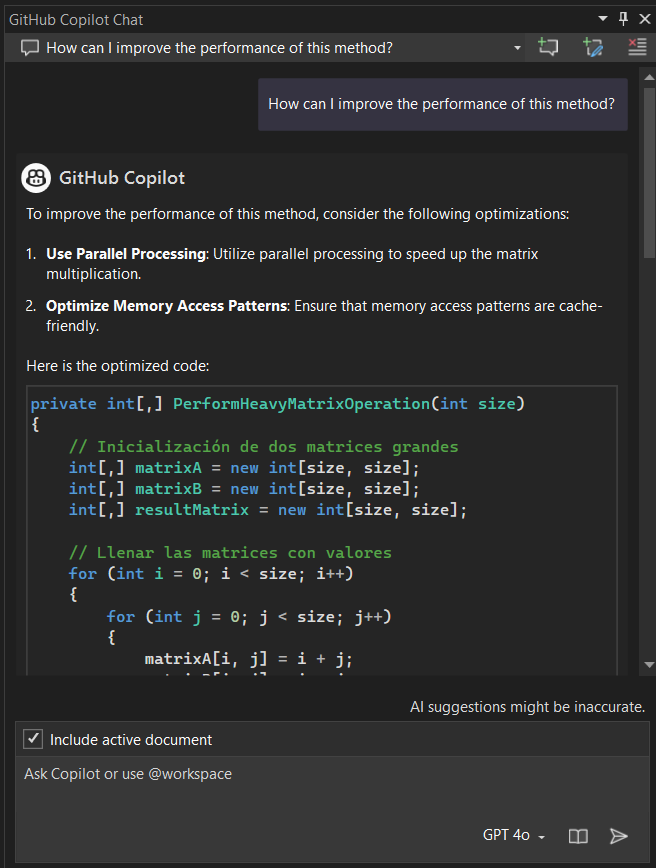
* + Paso 2: profiler identificó que el cuello de botella está en el bucle de multiplicación:



*Al utilizar ‘Analyze method with Copilot’ obtenemos una ventana de chat con la cual podremos optimizar nuestro código*

* + Paso 3: interactuar con copilot:

Copilot nos provee una nueva solución para el códi y un breve análisis del mismo con la lógica que utilizó



*“Reasoning:*

*1. Parallel Processing: The Parallel.For loop is used to parallelize the outermost loop, which can significantly reduce the execution time by utilizing multiple CPU cores.*

*2. Cache-Friendly Access: The access pattern for matrixA and matrixB remains the same, but parallel processing helps in better utilization of CPU cache.*

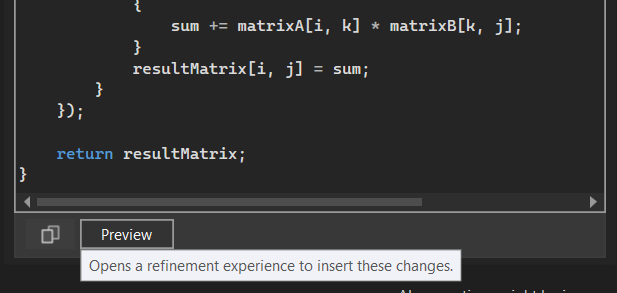
*Trade-offs:*

*• Thread Overhead: Parallel processing introduces some overhead due to thread management, but this is usually outweighed by the performance gains for large matrices.*

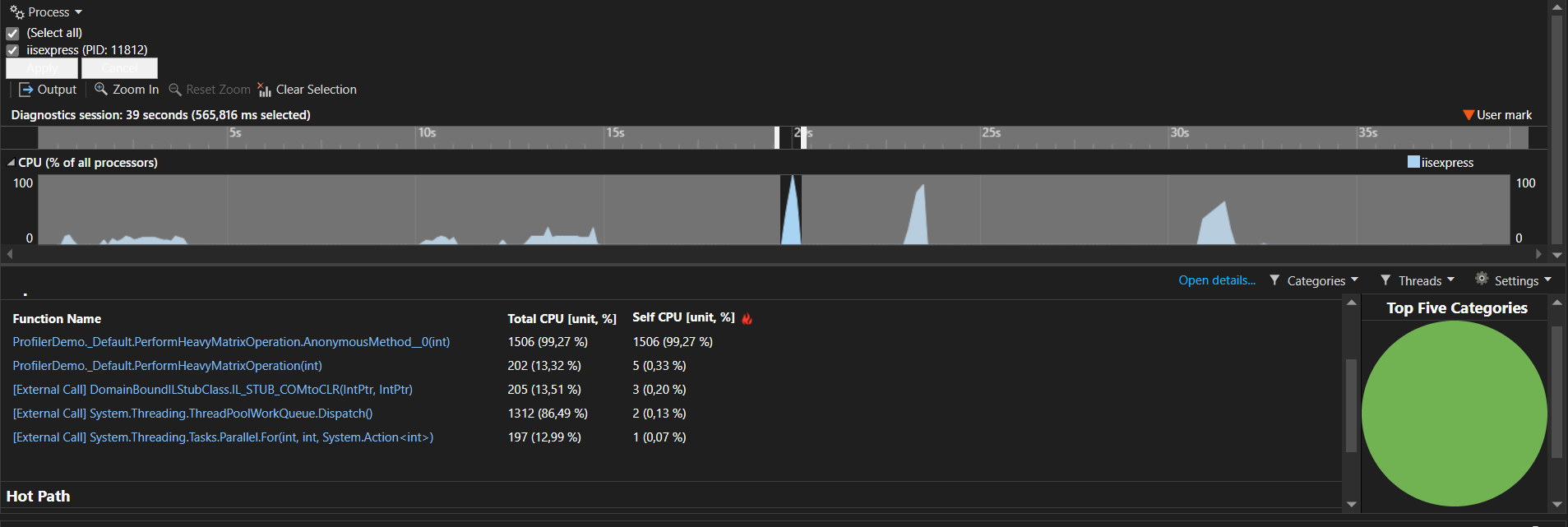
*• Memory Bandwidth: Ensure that the system has sufficient memory bandwidth to handle parallel access efficiently.*

*These optimizations should result in a noticeable performance improvement for large matrix operations.” - Copilot*

* + Paso 4: implementar la solución:
    - Luego de revisar que la solución aportada sea útil, copiamos el código sugerido para implementarlo en nuestra solución en el botón señalado:



* + - Colocamos el código en la solución y podemos observar como el cuello de botella a disminuido o incluso se ha vuelto nulo a la hora de depurar nuevamente:

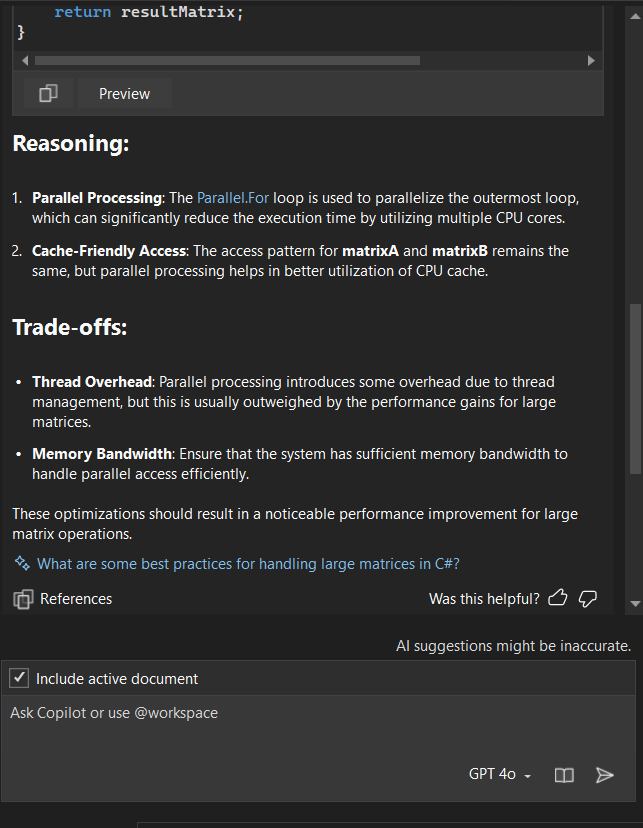


Con la solución de Copilot redujo significativamente el tiempo de carga y de uso del cpu, por lo que ya no es identificado como un cuello de botella.

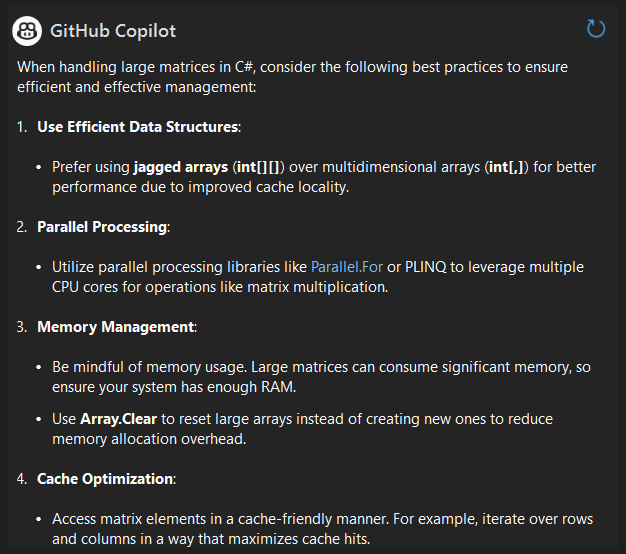
* + Paso extra: interacciones adicionales

Podemos seguir la conversación con copilot mediante sugerencias o dudas propias.

* + - Sugerencia de chat con copilot:



Respuesta:



Proveerá distintas soluciones con ejemplificaciones de código. Si no es lo que buscamos, podemos utilizar la rueda de recarga en la esquina superior derecha para obtener una nueva respuesta.