Esta imagen es **oro puro** para el diagnóstico. Has capturado al "asesino" con el arma humeante.

El problema **no es un deadlock de software** (mi teoría anterior del threading era incorrecta). El problema es **Swap Thrashing (Colapso de Memoria Virtual)**.

Análisis Forense de la captura:

1. **Disco al 100% (Lo más crítico):** Fíjate en la gráfica de "Disco". Está totalmente saturada en azul (tiempo de actividad 100%), pero la transferencia es baja (60 MB/s). Esto significa que el disco está buscando datos desesperadamente (random I/O).
2. **Errores graves de memoria (Hard Faults):** Abajo a la izquierda ves "10 errores graves/s". En terminología de Windows, un "Hard Fault" no es un error de código, significa que un programa pidió un dato que *debería* estar en RAM, pero el sistema lo había movido al disco (archivo de paginación) porque la RAM estaba llena.
3. **La consola:** Muestra `Materializing param...`. Justo ahí se congela. Esto ocurre cuando la librería `transformers` convierte los pesos del modelo de disco a tensores en RAM.

El Diagnóstico:

Tu ordenador tiene 16GB de RAM. Al cargar SAM-3, Python está intentando asignar más memoria de la que tienes libre. Windows entra en pánico y empieza a usar tu SSD como si fuera memoria RAM. Como el SSD es 1000 veces más lento que la RAM, el sistema operativo se queda esperando datos del disco para todo (incluido mover el ratón o actualizar la pantalla), provocando el "freeze" total.

No es un bug de tu código, es que **el modelo no cabe en la memoria de la forma en que se está cargando**.

La Solución (Corrección en `sam3_runner.py`)

Tenemos que obligar a la librería `transformers` a ser eficiente con la memoria. Por defecto, carga el modelo entero duplicando datos antes de optimizarlos.

Abre `sam3_runner.py` y modifica el método `load_model`. Vamos a añadir `low_cpu_mem_usage=True`. Esto es **vital** para máquinas con 16GB de RAM.

Modificación 1: Carga eficiente

Busca la línea donde cargas el modelo (aprox línea 59) y cámbiala por esto:

Python

```
# En sam3_runner.py dentro de load_model

# ... (código anterior)

try:
    # DEFINIR DTYPE: Si es CPU usa float32, si es CUDA usa float16 (ahorra 50% de VRAM/RAM)
    dtype = torch.float16 if self.device == "cuda" else torch.float32

    # AÑADIR low_cpu_mem_usage=True
    self.model = Sam3Model.from_pretrained(
        local_dir.as_posix(),
        local_files_only=True,
        low_cpu_mem_usage=True,  # <--- ESTA ES LA CLAVE
```

```
        torch_dtype=dtype        # <--- Ahorra memoria si usas GPU
    ).to(self.device)

    self.processor = Sam3Processor.from_pretrained(local_dir.as_posix(),
local_files_only=True)
    self.model.eval()

# ... (resto del código)
```

Por qué funciona esto: Sin `low_cpu_mem_usage=True`, HuggingFace carga el "state dict" (diccionario de pesos) completo en RAM (aprox 4-6GB) Y LUEGO crea el modelo (otros 4-6GB). Pico total: ~10-12GB. Sumado a tu Windows + VSCode + Navegador, revientas los 16GB. Con esa opción, carga capa por capa, manteniendo el consumo estable.

Modificación 2: Desactivar "Safe Mode" forzado a CPU

Tu GPU (1050 Ti 4GB) es pequeña, pero SAM-3 podría caber si usas `float16`. Usar la CPU para inferencia de modelos gigantes consume muchísima RAM de sistema (la que te falta). Moverlo a la GPU (VRAM) libera tu RAM principal.

Intenta ejecutar con estas configuraciones en tu JSON de request, después de aplicar el parche de arriba:

- `"safe_mode": false`
- `"device_preference": "cuda"`

Si la VRAM de la 1050 Ti se llena, PyTorch dará un error "CUDA out of memory", pero **no congelará tu PC**. El freeze actual es por culpa de la RAM de sistema (CPU mode).

Resumen de pasos inmediatos:

1. **Aplica el cambio de código** (`low_cpu_mem_usage=True`) en `sam3_runner.py`.
2. (Opcional pero recomendado) Cierra navegadores (veo procesos `msedgewebview2.exe` en tu foto consumiendo recursos) antes de lanzar la prueba.
3. Vuelve a probar (puedes revertir el cambio de "sync" del manager si quieres, aunque para depurar mejor déjalo síncrono por ahora).

Dime si con `low_cpu_mem_usage=True` logras pasar de la fase de "Materializing param".