Fourieranalyse - Parallel

Bildquelle: Bild wurde von Copilot (DALL-E3) generiert:
„Ein Nerd, der versucht eine Fourieranalyse parallel laufen zu lassen “

Jan Niclas Ruppenthal
1481198

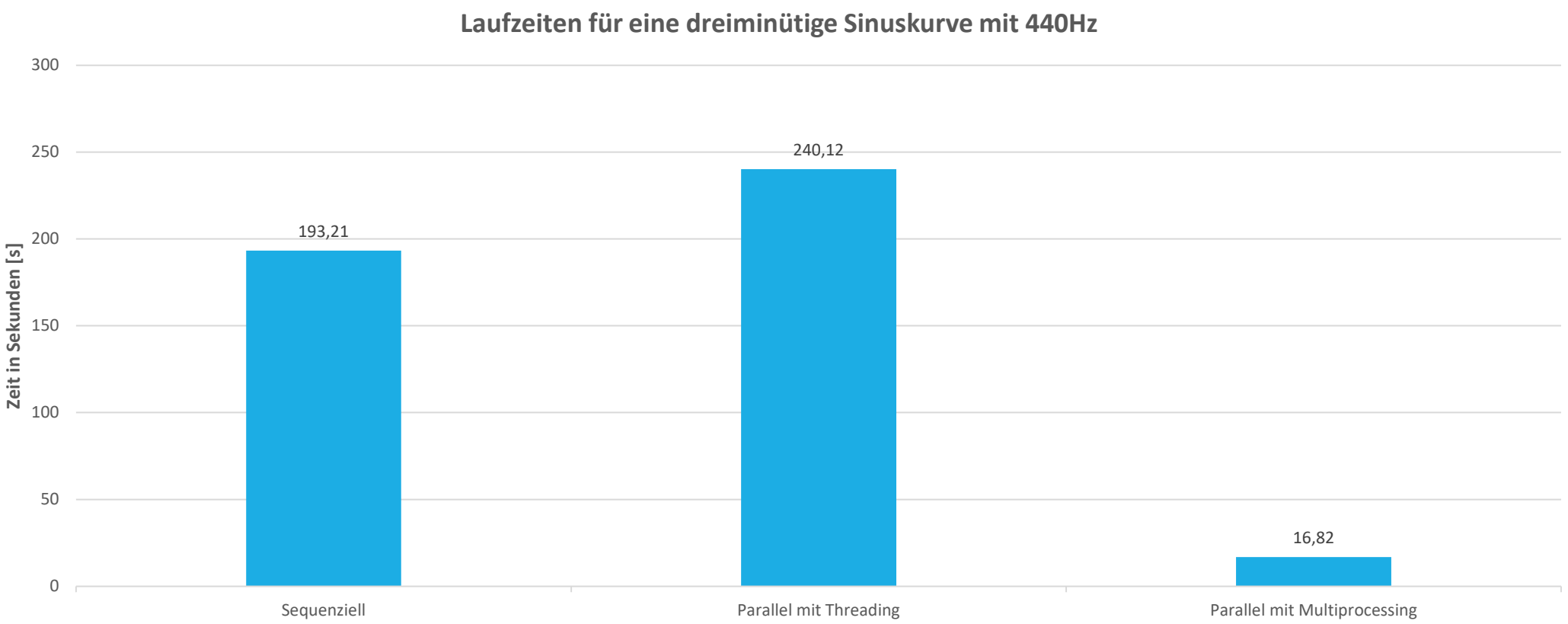
Parallel auf der CPU - I

```
def analyze(data, block_size, shift_size):
    […]

    aggregated_fft = mp.Array('d', block_size//2)

    for id in range(cpu_count):
        process = mp.Process(target=fft_process, args=(id, lock, cpu_count, data, block_size, shift_size,
aggregated_fft))
        processes.append(process)
        process.start()

    for process in processes:
        process.join()

    aggregated_fft = np.frombuffer(aggregated_fft.get_obj())
    aggregated_fft /= num_blocks

    return aggregated_fft
```

Parallel auf der CPU - II

```
def fft_process(id, lock, cpu_count, data, block_size, shift_size, aggregated_fft):
    local_fft = np.zeros(block_size//2)
    for i in range(id * shift_size, len(data) - block_size + 1, cpu_count * shift_size):
        block = data[i:i+block_size]
        fft_result = np.fft.fft(block)
        local_fft += np.abs(fft_result[:block_size//2])


    with lock:
        for i in range(block_size//2):
            aggregated_fft[i] += local_fft[i]
```

Problem mit threading



Parallel mit OpenCL

```
def analyze(data, block_size, shift_size):
    […]
    while num_blocks_temp > 0:
        current_limit = min(max_limit, num_blocks_temp)
        […]

        fft_kernel.set_args(data_buffer, result_buffer, np.int32(block_size),
            np.int32(shift_size), np.int32(len(data[start:start + current_limit * shift_size + block_size])))

        […]

        aggregated_fft += np.sum(fft_result, axis=0)

        […]

    aggregated_fft /= num_blocks

    return aggregated_fft
```

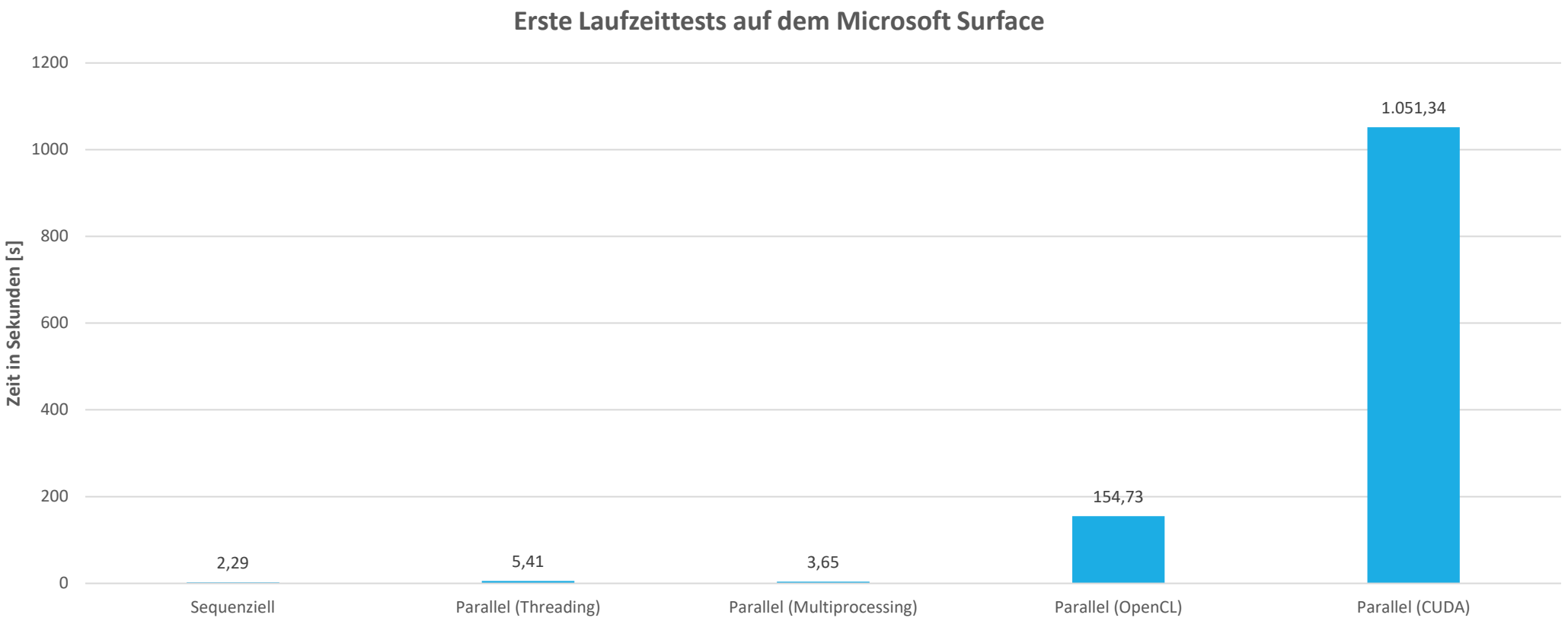
Parallel mit OpenCL - Kernel

```
__kernel void fft_kernel(__global const float *data, __global float *result, int block_size, int shift_size, int data_length) {
    int gid = get_global_id(0);
    int block_start = gid * shift_size;
    if (block_start + block_size > data_length) return;

    for (int k = 0; k < block_size/2; k++) {
        float sum_real = 0.0f;
        float sum_img = 0.0f;
        for (int n = 0; n < block_size; n++) {
            float angle = -2.0f * M_PI * k * n / block_size;
            sum_real += data[block_start + n] * cos(angle);
            sum_img -= data[block_start + n] * sin(angle);
        }
        result[gid * block_size/2 + k] = sqrt(sum_real * sum_real + sum_img * sum_img);
    }
}
```

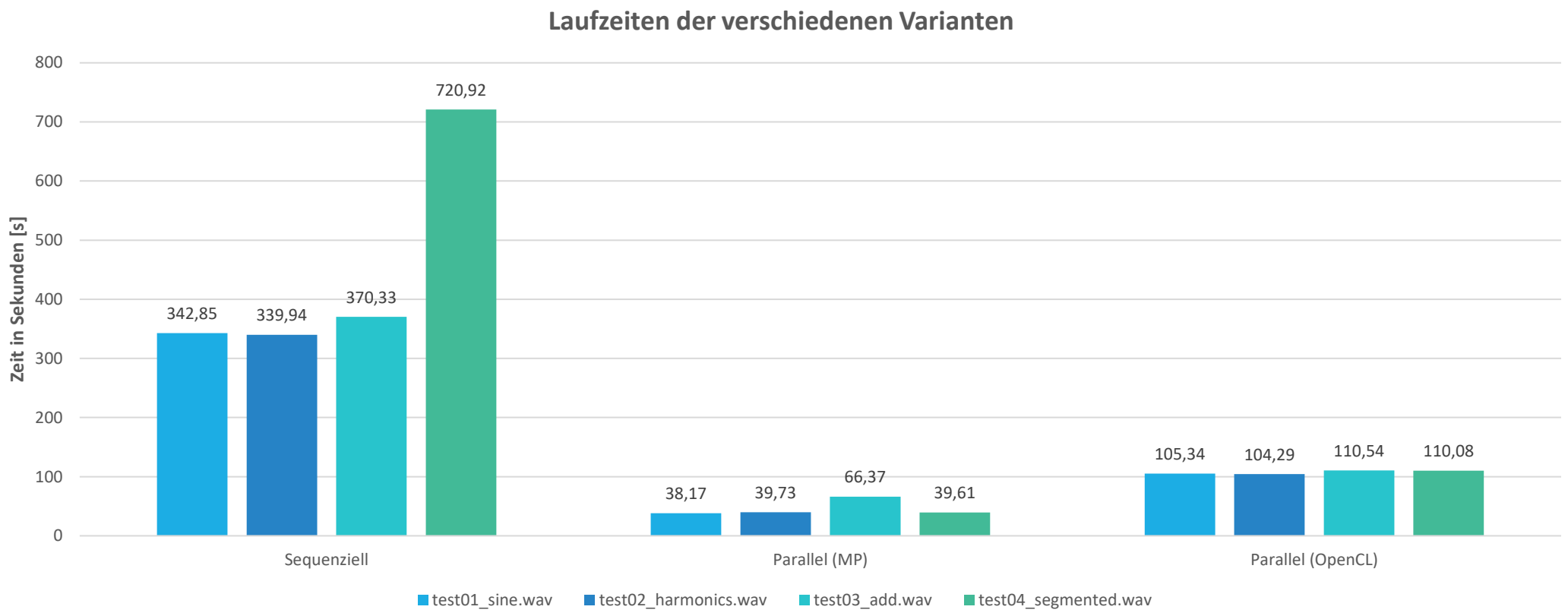
Parallel mit CUDA

- Prozessor: Intel(R) Core(TM) i7-8560U @ 1.90 GHz
- Installierter RAM: 16,0 GB
- Integrierte GPU: Intel(R) UHD Graphics 620
- Nvidia GPU: NVIDIA GeForce GTX 1050
- Systemtyp: 64-Bit-Betriebssystem, x64-basierter Prozessor
- Edition: Windows 10 Pro

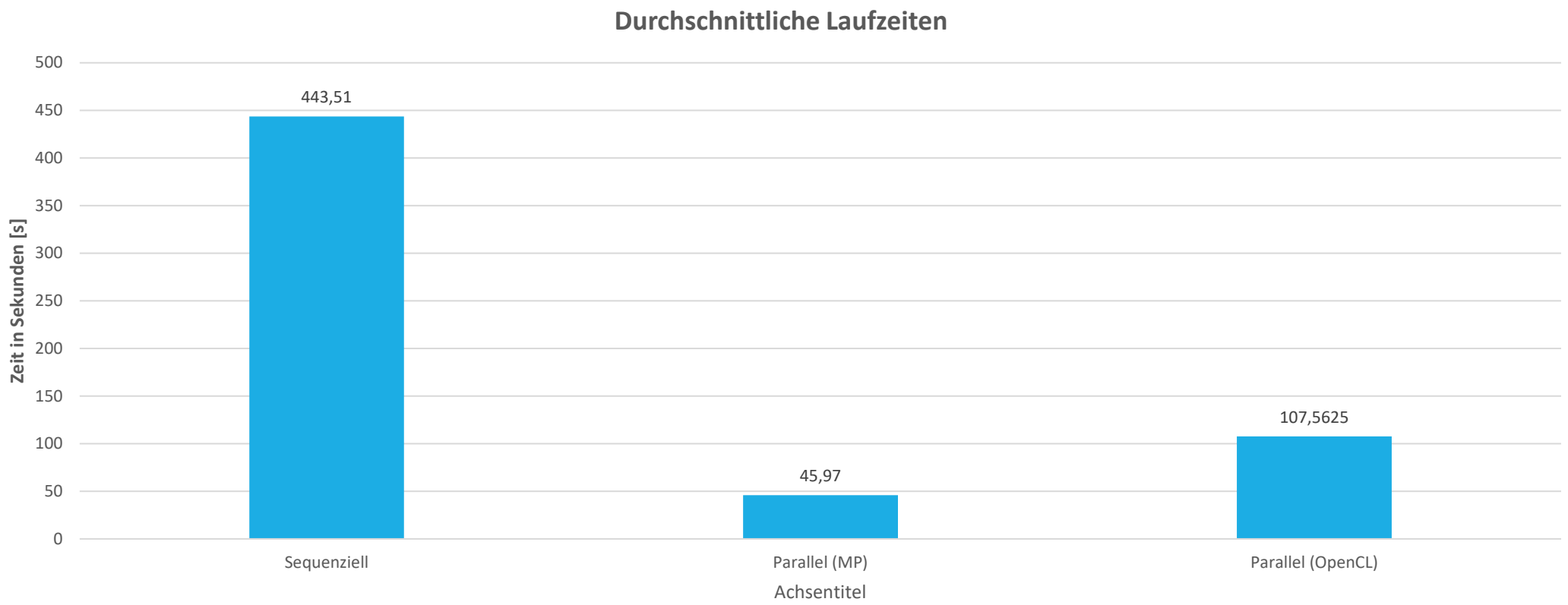
Problem mit CUDA



Vergleich der Laufzeiten - I

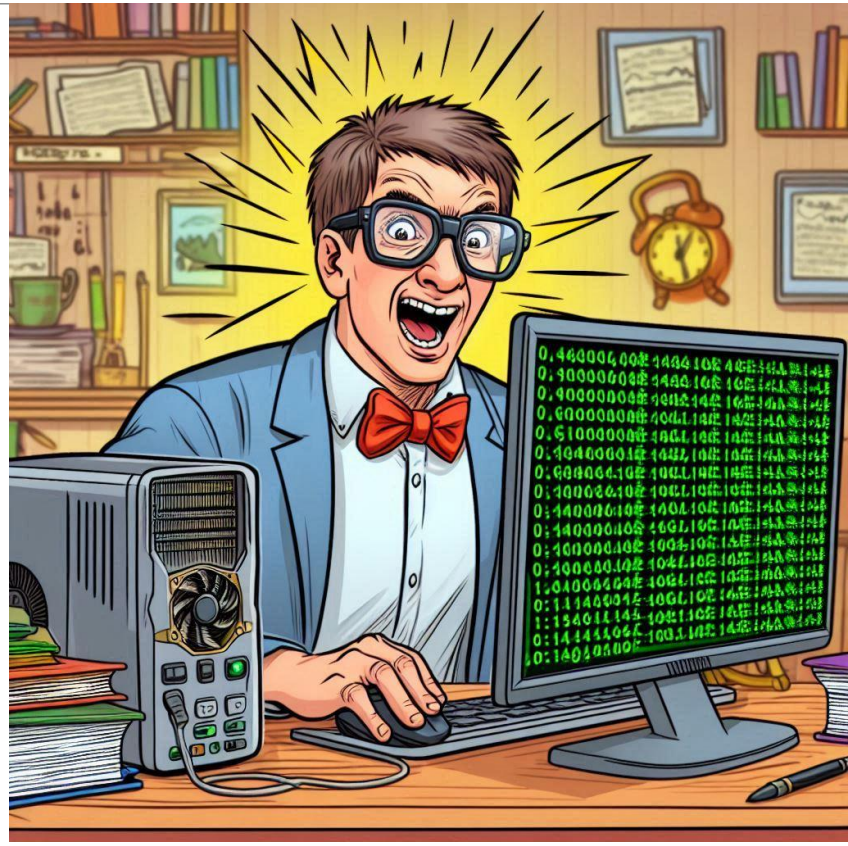


Vergleich der Laufzeiten - II



Diskussion

- Maximalen Speedup auf der CPU nicht erreicht
  - Notwendigkeit der Synchronisierung
- Maximalen Speedup auf der GPU nicht erreicht
  - Aufteilen der Daten
  - Mehrmaliges Kopieren der Daten auf die GPU
  - Zwischenergebnisse werden von der CPU miteinander addiert
  - Einfache Implementierung einer Fourieranalyse
- Ungeeignete Wahl der Programmiersprache
  - Java
  - Rust



Bildquelle: Bild wurde von Copilot (DALL-E3) generiert: „Nerd, der versucht eine Fourieranalyse auf der GPU parallel laufen zu lassen. “