Proiect de Testare a Performanțelor Hardware ale unui PC

Autor: Toma-Nan George-Alexandru
Universitate: Universitatea Tehnică din Cluj-Napoca
Catedra: Calculatoare
Numar grupa: 30238
Coordonator Științific: Andrei Mihai Sopterean

19 Decembrie 2024

Cuprins

1 Rezumat

Acest proiect are ca scop testarea performanțelor hardware ale unui PC prin analizarea parametrilor esențiali precum viteza procesorului, dimensiunea memoriei RAM, viteza de transfer a datelor, precum și capacitatea de a executa operații aritmetice și logice complexe. Proiectul implementează un program Python modular, care folosește biblioteci precum ‘psutil‘ și ‘numpy‘ pentru a măsura parametrii hardware și pentru a evalua performanțele procesorului în termeni de TFLOPS (trilioane de operații floating-point pe secundă). Rezultatele obținute evidențiază limitele hardware și oferă o bază comparativă pentru optimizarea performanțelor sistemului.

2 Introducere

Performanțele hardware reprezintă un criteriu esențial în evaluarea eficienței unui PC, fie că este utilizat în scop personal, academic sau industrial. Odată cu creșterea complexității aplicațiilor moderne, devine imperativ să înțelegem capacitatea sistemului de a gestiona sarcini intensive.

Acest proiect își propune să analizeze performanțele unui PC prin:

- Determinarea caracteristicilor procesorului (tip, frecvență).
- Calcularea dimensiunii totale a memoriei RAM.
- Măsurarea vitezei de transfer a blocurilor de date.
- Testarea execuției operațiilor aritmetice și logice.
- Evaluarea TFLOPS pentru sarcini complexe.

Proiectul folosește limbajul Python datorită flexibilității sale și a ecosistemului larg de biblioteci dedicate măsurătorilor hardware și performanței.

3 Fundamentare Teoretică

În cadrul acestui proiect, sunt utilizați următorii parametri hardware pentru evaluare:

1. Procesorul (CPU): Procesorul reprezintă unitatea principală de calcul. Se analizează frecvența acestuia (în MHz) și tipul, folosind biblioteca 'psutil'.

2. Memoria RAM: Dimensiunea memoriei RAM este un factor esențial pentru rularea aplicațiilor intensive. Măsurătorile sunt efectuate în GB, convertind valoarea înregistrată în bytes.

3. Viteza de transfer: Transferul blocurilor de date este testat prin copierea acestora în memorie și măsurarea timpului necesar.

4. TFLOPS (Trilioane de operații floating-point pe secundă): Performanța procesorului este cuantificată folosind TFLOPS, care indică capacitatea de procesare în calcule complexe.

Folosirea limbajului Python permite accesarea directă a acestor informații prin pachete precum:

- psutil: Acces la informațiile hardware și utilizarea resurselor.
- numpy: Operații vectoriale rapide și eficiente.
- multiprocessing: Paralelizare pentru evaluarea TFLOPS.

4 Proiectare și Implementare

4.1 Arhitectura Generală

Programul este structurat modular pentru a permite testarea diverselor componente hardware și colectarea rezultatelor. Principalele module sunt:

1. Determinarea parametrilor hardware de bază:

- Funcțiile `get_cpu_info` și `get_memory_info` colectează informații despre procesor și dimensiunea memoriei RAM.

2. Măsurarea vitezei de transfer a datelor:

- Funcția `test_transfer_speed` evaluează viteza de transfer a blocurilor de date fara "goluri" de memorie. Functia `test_transfer_speed_multiple_b` evaluează viteza de transfer a blocurilor de date cu "goluri" de memorie.

3. Testarea performanțelor operațiilor aritmetice și logice:

- Funcția `test_arithmetic_operations` folosește biblioteca `timeit` pentru a măsura timpul de execuție al operațiilor simple pe vectori.

4. Evaluarea performanței TFLOPS: -

- Funcțiile `test_tflops` și `compute_max_tflops_optimized` evaluează performanța procesorului în termeni de TFLOPS, utilizând operații vectoriale și paralelizare.

5. Funcția principală (`main`):

- Integrează toate funcționalitățile și afișează rezultatele într-un format ușor de citit.

4.2 Implementare Detaliată

4.2.1 Colectarea Informațiilor de Bază

```
def get_cpu_info():
    cpu_name = platform.processor()
    cpu_freq = psutil.cpu_freq().current
    return cpu_name, cpu_freq
```

Această funcție returnează denumirea procesorului și frecvența sa actuală, folosind biblioteca 'psutil'.

```
def get_memory_info():
    total_memory = psutil.virtual_memory().total / (1024 ** 3)
    return total_memory
```

Această funcție returnează memoria RAM a procesorului.

4.2.2 Măsurarea Vitezei de Transfer

```
def test_transfer_speed():
        main_block = []
        block_size = 10000000
        transfer_time = 0.0
        for i in range(4):
            data_block = np.random.rand(block_size)
            main_block.append(data_block)
        for block in main_block:
            start_time = time.time()
            transfer = block.copy()
            end_time = time.time()
            transfer_time += end_time - start_time
        return transfer_time
```

Această funcție măsoară timpul necesar pentru transferul unor blocuri de date (de dimensiuni mari) prin copierea lor în memorie.

```
def test_transfer_speed_multiple_blocks_with_gaps(
block_size=10000000, gap_size=1000000):
    main_blocks = []

    # 4 main data blocks with gaps in between
    for i in range(4):
        # main memory block
        main_blocks.append(np.random.rand(block_size))

        # Add a gap block
        if i < 3:  # Only add gaps between main blocks
            main_blocks.append(np.zeros(gap_size))

    # Measure time
    total_transfer_time = 0.0

    i = 0
```

```
    for block in main_blocks:
        if i % 2 == 0:
            start_time = time.time()
            copied_block = block.copy()  # Copy operation
            end_time = time.time()
            total_transfer_time += (end_time - start_time)
        i += 1

        # Accumulate the time taken for copying each main bloc
        # total_transfer_time += (end_time - start_time)

    return total_transfer_time
```

Această funcție măsoară timpul necesar pentru transferul unor blocuri de date (de dimensiuni mari) cu spatii intre ele prin copierea lor în memorie.

4.2.3 Testarea Operațiilor Aritmetice:

```
def test_arithmetic_operations():
    setup_code = """
import numpy as np
a = np.random.rand(1000000)
"""
    test_code = """
b = a * 2
"""
    exec_time = timeit.timeit(stmt=test_code,
    setup=setup_code, number=100)
    return exec_time
```

Funcția utilizează 'timeit' pentru a măsura timpul necesar execuției unor operații aritmetice pe un vector de dimensiuni mari.

4.2.4 Evaluarea TFLOPS

```
def test_tflops():
    n = 10 ** 7  # Size of arrays (10 million elements)
    iterations = 100  # Number of iterations

    # Initialize large arrays with random values
    a = np.random.rand(n)
```

```
    b = np.random.rand(n)
    c = np.zeros(n)  # Output array

    total_operations = 2 * n * iterations  # Each iteration do

    # Start timing
    start_time = time.time()
    for _ in range(iterations):
        c = a * b + c  # Floating-point operations: multiplica
    end_time = time.time()

    # Calculate FLOPS
    execution_time = end_time - start_time
    flops = total_operations / execution_time
    tflops = flops / 1e12  # 10^12

    return tflops, execution_time
```

Această funcție măsoară performanța procesorului în TFLOPS utilizând operații matematice de inmultire si adunare pe vectori de dimensiuni mari.

4.2.5 Evaluarea TFLOPS maximizata

```
def compute_max_tflops_optimized(process_id, duration,
result_dict):
    n = 10 ** 8  # Increase array size for larger workload
    a = np.random.rand(n).astype(np.float32)
    b = np.random.rand(n).astype(np.float32)
    c = np.zeros(n, dtype=np.float32)

    total_operations = 0
    start_time = time.time()

    while time.time() - start_time < duration:
        # Perform multiple operations to maximize FLOPS
        c = a * b + c * a - b * a + c * b
        total_operations += 6 * n  # 6 operations per element

    execution_time = time.time() - start_time
    flops = total_operations / execution_time
```

```
tflops = flops / 1e12  # Convert FLOPS to TFLOPS
timp = execution_time / total_operations  # Compute 'timp'
frequency_hz = 1.0 / timp if timp > 0 else 0  # Compute fre
frequency_mhz = frequency_hz / 1e6  # Convert Hz to MHz

print(f"Process {process_id} executed
with timp={timp:.10f} seconds
and frequency={frequency_mhz:.2f} MHz.")

# Store the results in the shared dictionary
result_dict[process_id] = (tflops, execution_time,
timp, frequency_mhz)
```

Această funcție măsoară performanța procesorului în TFLOPS utilizând paralelizare (prin 'multiprocessing') și operații matematice intensive.

5 Rezultate Experimentale

5.1 Rezultate PC actual

```
D:\anu3\SSC\PROIECT_SSC\testare_param_performance\.venv\Scripts\python.exe D:\anu3\SSC\PROIECT_SSC\testare_param_
Procesor: Intel64 Family 6 Model 140 Stepping 1, GenuineIntel
Frecvența procesorului: 2419.0 MHz
Memorie RAM totală: 15.70 GB
Timp transfer bloc date: 0.061103 secunde
Timp transfer bloc date cu goluri: 0.067391 secunde
Timp execuție operații aritmetice: 0.213521 secunde
TFLOPS measured: 0.0004 TFLOPS
Execution time: 4.763659 seconds
Process 3 executed with timp=0.0000000033 seconds and frequency=3036.97 MHz.
Process 2 executed with timp=0.0000000034 seconds and frequency=2952.72 MHz.
Process 0 executed with timp=0.0000000035 seconds and frequency=2892.52 MHz.
Process 1 executed with timp=0.0000000034 seconds and frequency=2898.73 MHz.

Maximized TFLOP Results for Each Process:
Process 3: TFLOPS=1.0370, Execution Time=6.926958 seconds, Timp=0.0000000033 seconds, Frequency=3036.97 MHz
Process 2: TFLOPS=0.9527, Execution Time=6.209623 seconds, Timp=0.0000000034 seconds, Frequency=2952.72 MHz
Process 0: TFLOPS=0.9925, Execution Time=6.222942 seconds, Timp=0.0000000035 seconds, Frequency=2892.52 MHz
Process 1: TFLOPS=0.9387, Execution Time=6.096075 seconds, Timp=0.0000000034 seconds, Frequency=2898.73 MHz

Total TFLOPS across all processes: 3.9209 TFLOPS

Process finished with exit code 0
```

În cadrul testelor efectuate, au fost rulate diverse scenarii pentru a evalua performanțele hardware ale unui PC. Datele obținute sunt prezentate mai jos.

5.2 Analiza Rezultatelor

5.2.1 Performanța Procesorului

- Frecvența raportată de 2419.0 MHz este în concordanță cu specificațiile hardware.
- Performanța TFLOPS măsurată pe un singur fir de execuție este limitată la 0.0004 TFLOPS, însă în regim multi-thread aceasta crește semnificativ la un total de 3.9209 TFLOPS.

5.2.2 Memoria RAM

- Dimensiunea totală de 15.70 GB oferă suficient spațiu pentru rularea programului și gestionarea blocurilor mari de date.

5.2.3 Transferul de Date

- Viteza de transfer a blocurilor de date este rapidă, cu un timp de 0.061103 secunde pentru transferuri simple și 0.067391 secunde pentru transferuri cu goluri.

5.2.4 TFLOPS Multi-Thread

- Performanța totală de 3.9209 TFLOPS evidențiază eficiența paralelizării, fiecare proces rulând în mod optim pentru a utiliza resursele hardware disponibile.

5.3 Observatii

Rezultatele oferă o imagine clară asupra performanțelor hardware și demonstrează eficiența procesorului în sarcini intensive.

5.4 Rezultate PC mai performant

```
C:\Users\atoma\OneDrive\Desktop\PROIECT_SSC\.venv\Scripts\python.exe C:\Users\atoma\OneDrive\Desktop\PROIECT_SSC\MAIN.py
Procesor: Intel64 Family 6 Model 191 Stepping 2, GenuineIntel
Frecvența procesorului: 2500.0 MHz
Memorie RAM totală: 15.84 GB
Timp transfer bloc date: 0.055806 secunde
Timp transfer bloc date cu goluri: 0.059643 secunde
Timp execuție operații aritmetice: 0.157596 secunde
TFLOPS measured: 0.0006 TFLOPS
Execution time: 3.621258 seconds
Process 1 executed with timp=0.0000000031 seconds and frequency=3048.79 MHz.
Process 3 executed with timp=0.0000000032 seconds and frequency=3072.05 MHz.
Process 0 executed with timp=0.0000000030 seconds and frequency=3136.12 MHz.
Process 2 executed with timp=0.0000000032 seconds and frequency=3267.16 MHz.

Maximized TFLOP Results for Each Process:
Process 1: TFLOPS=1.1110, Execution Time=5.508305 seconds, Timp=0.0000000031 seconds, Frequency=3048.79 MHz
Process 3: TFLOPS=1.1923, Execution Time=5.687058 seconds, Timp=0.0000000047 seconds, Frequency=3072.05 MHz
Process 0: TFLOPS=1.2847, Execution Time=6.239868 seconds, Timp=0.0000000052 seconds, Frequency=3136.12 MHz
Process 2: TFLOPS=1.3268, Execution Time=6.322090 seconds, Timp=0.0000000035 seconds, Frequency=3267.16 MHz

Total TFLOPS across all processes: 4.9148 TFLOPS

Process finished with exit code 0
```

5.5 Rezultate PC mai putin performant

```
Procesor: Intel64 Family 6 Model 142 Stepping 12, GenuineIntel
Frecvența procesorului: 1494.0 MHz
Memorie RAM totală: 7.85 GB
Timp transfer bloc date: 0.168882 secunde
Timp transfer bloc date cu goluri: 0.268624 secunde
Timp execuție operații aritmetice: 0.358528 secunde
TFLOPS measured: 0.0002 TFLOPS
Execution time: 8.831098 seconds
Process 3 executed with timp=0.0000000358 seconds and frequency=2790.28 MHz.
Process 2 executed with timp=0.0000000345 seconds and frequency=2890.84 MHz.
Process 1 executed with timp=0.0000000417 seconds and frequency=2400.08 MHz.
Process 0 executed with timp=0.0000000116 seconds and frequency=2862.77 MHz.

Maximized TFLOP Results for Each Process:
Process 3: TFLOPS=0.4579, Execution Time=21.483970 seconds, Timp=0.0000000358 seconds, Frequency=2790.28 MHz
Process 2: TFLOPS=0.4290, Execution Time=20.700951 seconds, Timp=0.0000000345 seconds, Frequency=2890.84 MHz
Process 1: TFLOPS=0.5240, Execution Time=24.992105 seconds, Timp=0.0000000417 seconds, Frequency=2400.08 MHz
Process 0: TFLOPS=0.3863, Execution Time=6.954305 seconds, Timp=0.0000000116 seconds, Frequency=2862.77 MHz

Total TFLOPS across all processes: 1.7972 TFLOPS
```

6 Concluzii

Proiectul realizat a avut ca scop evaluarea performanțelor hardware ale unui PC prin intermediul unor teste specifice pentru procesor, memorie RAM și viteza de transfer a datelor. În urma experimentelor efectuate pe trei procesoare distincte, s-au putut trasa concluzii clare cu privire la capacitățile și limitările acestora.

6.1 Concluzii Generale

Dupa cum se poate observa, am comparat programul de benchmark pe 3 tipuri de procesoare, cel standard aflat pe pc-ul meu, pe un calculator cu un procesor mai performant, cel al fratelui meu si pe un calculator cu un procesor mai putin performant, pe pc-ul colegului de camera.

Pc-ul meu are un procesor destul de performant, de aceea rezultatele sunt destul de bune. Putem observa ca timpul de transfer al datelor si cel pentru operatii este destul de scazut, iar operatiile pe TFLOPS ridica procesorul la o frecventa destul de mare.

In comparatie, pe procesorul mai performant, se poate vedea o diferenta destul de majora din punct de vedere al timpului de executie pentru transferul de date si operatii, iar pentru TFLOPS frecventa este mai ridicata si timpul pentru ele mai scazut.

In ulima rulare, cea pe procesorul mai slab, se poate observa diferenta de performanta, timpul de transfer si operatii este mult mai ridicat, iar nivelul de TFLOPS scazut, la o frecventa mai redusa.

In concluzie, putem afirma ca nivelul de performanta al PC-ului depinde de tipul procesorului.

6.2 Limitări ale Experimentului

6.2.1 Lipsa Utilizării GPU-ului:

- Testele s-au concentrat exclusiv pe performanțele CPU-ului. Integrarea GPU-ului ar putea duce la o creștere semnificativă a TFLOPS, în special pentru sarcini intensive de calcule floating-point.

6.2.2 Configurații Statice:

- Dimensiunile blocurilor de date, durata execuției și numărul de procese au fost fixe. Configurații mai dinamice ar permite o evaluare mai detaliată.

6.2.3 Impactul Temperaturii:

- Performanța procesoarelor ar putea varia în funcție de temperatură și de mecanismele de throttling, dar acest aspect nu a fost luat în considerare.

6.3 Perspective de Îmbunătățire

6.3.1 Adăugarea Testelor GPU:

- Extinderea testelor pentru a include performanțele GPU-ului ar oferi o imagine completă asupra capacităților hardware ale sistemului.

6.3.2 Configurabilitate Sporită:

- Dezvoltarea unei interfețe de utilizator pentru configurarea parametrilor testelor (dimensiuni, procese, durate) ar crește flexibilitatea programului.

6.3.3 Automatizarea Raportării:

- Integrarea unui modul care generează rapoarte detaliate în format PDF, inclusiv grafice și analize comparative, ar îmbunătăți experiența utilizatorului.

6.4 Concluzie finala

Proiectul și-a atins scopul de a analiza și compara performanțele hardware a trei procesoare diferite, oferind o metodă clară și eficientă de testare. Rezultatele obținute subliniază diferențele arhitecturale dintre procesoare și evidențiază oportunități de optimizare pentru aplicațiile intensive în resurse hardware.

7 Bibliografie

Bibliografie

[1] Python Software Foundation. *platform — Access to underlying platform's identifying data*. Disponibil la: `https://docs.python.org/3/library/platform.html`.

[2] *psutil documentation*. Disponibil la: `https://psutil.readthedocs.io/en/latest/`.

[3] Python Software Foundation. *timeit — Measure execution time of small code snippets*. Disponibil la: `https://docs.python.org/3/library/timeit.html`.

[4] Python Software Foundation. *time — Time access and conversions*. Disponibil la: `https://docs.python.org/3/library/time.html`.

[5] Python Software Foundation. *multiprocessing — Process-based parallelism*. Disponibil la: `https://docs.python.org/3/library/multiprocessing.html`.

[6] Codedamn. *How to: Modeling and Simulation in Python (with an example)*. Disponibil la: `https://codedamn.com/news/python/how-to-modeling-and-simulation-in-python-with-an-example`.

[7] GeeksforGeeks. *Processes in Linux/Unix*. Disponibil la: `https://www.geeksforgeeks.org/processes-in-linuxunix/`.