Measuring the performance of computer systems using benchmarks

1 Purpose

This laboratory work introduces basic theoretical concepts about measuring the performance of computer systems using benchmarks and practical information on how to use benchmark applications in order to evaluate the performance of a computer.

2 Introduction to benchmarking

The easiest way to compare the performance of computer systems (or computer hardware) is to look at the specifications. This is not the most relevant method for assessing computer hardware performance, because, for example, two CPUs with the same specifications (but different architectures) can have different performances.

Efficiently measuring and comparing the performance of computer systems involves:

- Identifying good performance metrics (e.g. response time, bandwidth, resource utilization)
- Defining a method to measure the performance metrics
- Aggregating and interpreting the results

Good performance metrics are proportional to the performance of the system. They are deterministic, consistent, and easy to measure. Benchmarks are commonly used to measure performance metrics.

Benchmarking is a method used for evaluating the performance characteristics of computer hardware or software. In this context, the term "benchmark" has a double meaning. It refers to **executing a set of programs in order to evaluate the performance of hardware or software**, but it also refers to the **application used for benchmarking purpose**.

Many benchmark applications focus on measuring the performance of individual hardware components, such as processor execution time, memory speed and bandwidth, graphic processing and so on. To measure the overall performance of a system, there are collections of benchmark applications called benchmark suites. Each benchmark application in the suite addresses a specific component and performance parameter. The results are aggregated to express the performance of the entire system. Benchmark suites can offer facilities of result interpretation (benchmark scores) and they can also compare the obtained results with the results of reference computer systems (or computer hardware).

A good benchmark should:

- Provide relevant performance measures
- Present unbiased results that are accepted by users and vendors
- Be repeatable in the sense that several runs of the benchmark on the same machine produce the same results, with an acceptable level of variance
- Be portable

It is a common practice for vendors to tune their products for industry-standard benchmarks. Consequently, the benchmark scores for these products are not relevant. There have been cases when vendors used some strategies that allowed their products to receive good scores at benchmarks, while for real-life workloads the products performed a lot worse.

3 Examples of performance metrics

CPU performance metrics:

- Clock frequency (Hz)
- IPS – instructions per second (integer arithmetic operations)
- FLOPS – floating point operations per second
- Performance vs. Power – performance to energy ratio
- Performance vs. Speed – performance to speed ratio

Memory performance metrics:

- Bandwidth (B/s)
- Latency (response time)
- Capacity vs. Power – capacity to energy ratio
- Performance vs. Power – performance to energy ratio
- Performance vs. Speed – performance to speed ratio

Storage device performance metrics:

- Speed (B/s)
- Access time - average time to read a random sector on the disk

4 Types of benchmarks

The type of application used to measure the computer's performance influences the relevancy of the obtained results. There are several types of benchmark applications, such as:

- **Real applications.** The most relevant results are obtained by using real applications like compilers, file/video compression programs, video games, image-processing programs and so on. These applications expose the assessed systems to real workloads.
- **Kernels.** Kernels are small, key parts of real applications used as benchmarks. These benchmarks are not as relevant as complete real applications.
- **Component benchmarks.** Component benchmarks are programs designed to measure the performance of specific computer components. They are used for automatic detection of computer hardware parameters such as CPU clock frequency, cache size and architecture, etc.
- **Synthetic benchmarks.** Synthetic benchmarks are programs written for the specific purpose of benchmarking. These programs contain operations that statistically match the behavior of real applications. Dhrystone and Whetstone are popular synthetic benchmarks used for measuring the performance of integer and, respectively, floating point operations execution on CPUs. Synthetic benchmarks can produce unrealistic results, as they only mimic real workloads.
- **Parallel benchmarks.** Parallel benchmarks are used for systems with multiple cores, multiple processors or for systems consisting of multiple computers.

- **Micro-benchmarks.** Micro-benchmarks are intended to measure the performance of a very small portion of code.
- **I/O benchmarks.** I/O benchmarks are used for measuring the performance of I/O operations such as network communication.

Commonly used benchmark suites contain a mix of these types of benchmarks: synthetic, kernels, real applications, I/O and aggregate the results to obtain relevant information about the performance of the computer system.

5 Benchmark applications

There are organizations that produce and maintain standard sets of benchmarks for computers such as:

- Standard Performance Evaluation Corporation (SPEC)
- Embedded Microprocessor Benchmark Consortium (EEMBC)
- Transaction Processing Performance Council (TPC)

5.1 SPEC benchmarks

SPEC benchmarks are widely used to assess the performance of computer systems. An example of SPEC benchmark suite is CPU2006 used to evaluate the combined performance of the CPU and memory. The CPU2006 suite includes benchmarks for integer and floating point operations. The benchmarks are real applications or parts of real applications. Some examples of CPU2006 benchmarks are:

- Integer benchmarks
  - B2zip compression
  - Gcc compiler
  - Omnet discrete event simulator
  - XML processing
  - Chess game
  - Video compression
  - And others...
- Floating point benchmarks
  - Bwaves – Fluid dynamics
  - Gamess – Quantum chemistry
  - Povray – Image ray-tracing
  - Sphinx3 – Speech recognition
  - Wrf – Weather prediction
  - And others...

There are a lot of non-standard benchmark applications and benchmark suites. A small selection of free-ofcharge benchmarks is presented below.

5.2 CPU-Z

CPU-Z (Fig. 1) is a program that detects the computer hardware configuration and some performance parameters (like CPU clock rate, DRAM frequency, etc.). It offers information about the CPU, cache memory, main memory, about the mainboard (motherboard) and the GPU.

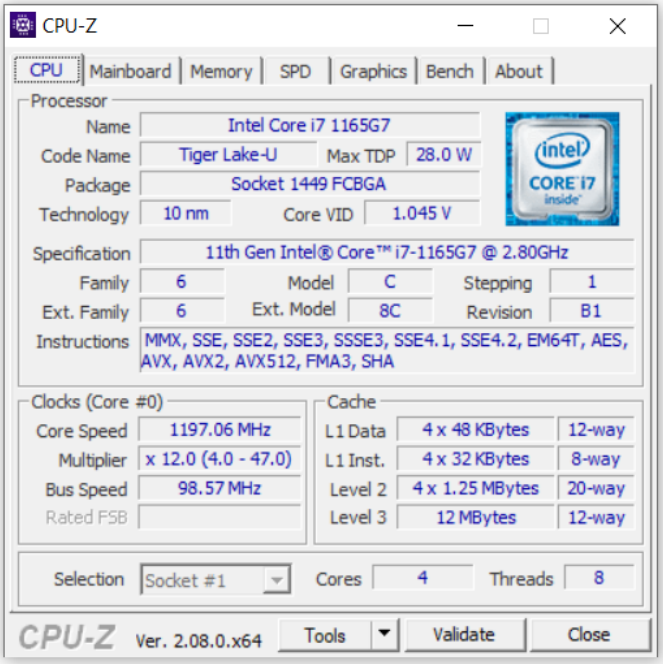

Figure 1: CPU-Z user interface

5.3 SuperPI

SuperPI (Fig. 2) is a simple benchmark that measures the execution time of a program that calculates the decimal digits of PI. The user has to select the number of digits to be calculated, and start the execution. The result is the time measured from the beginning to the end of the execution, in seconds.

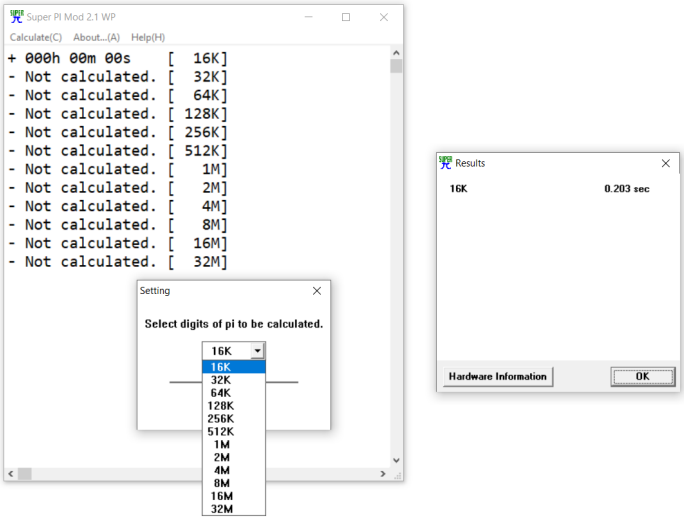

Figure 2: SuperPi user interface

5.4 NovaBench

NovaBench (Fig. 3) is a benchmark suite that measures the processing speed, 2D graphics performance and hard-drive performance.

It has a function for overall evaluation of the computer system and for the evaluation of the main

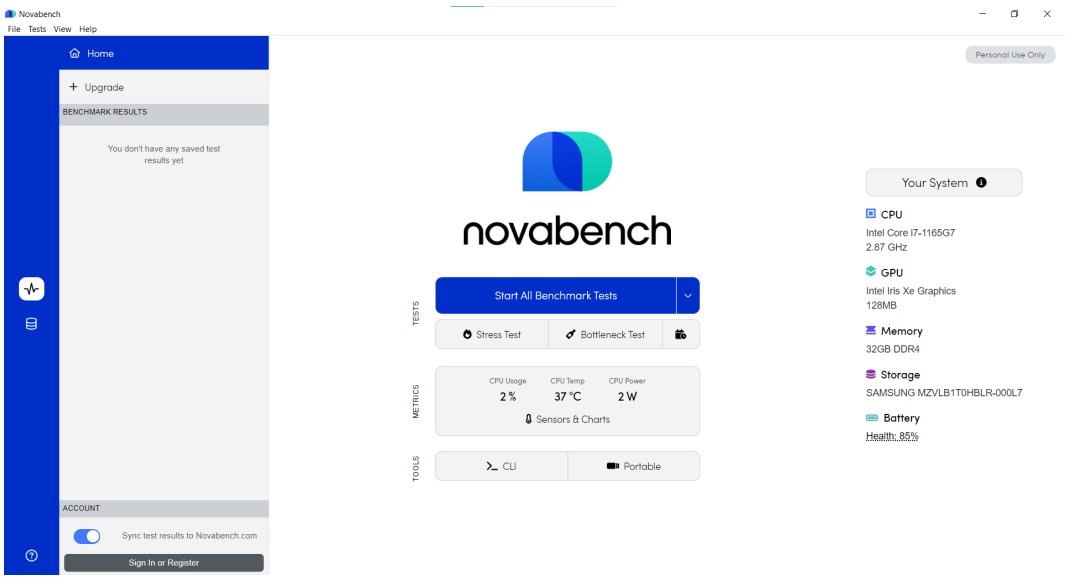

Figure 3: NovaBench user interface

hardware components: CPU, memory, hard-drive and GPU. Each component receives a score that is based on the measurements. The system's score is based on the component scores.

5.5 SiSoftware Sandra

SiSoftware Sandra is a very complex benchmark suite. It offers facilities for benchmarking the CPU (including multi-cores), memory, GPU, storage devices, network and both .net and java virtual machines.

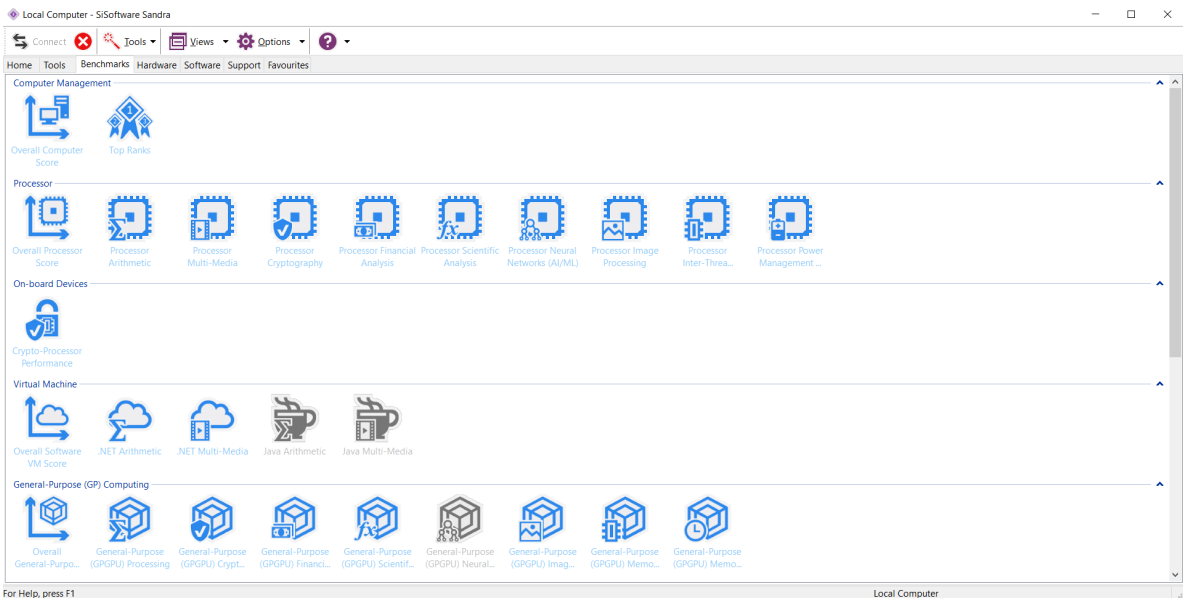

Figure 4: SiSoftware Sandra user interface

The components' benchmark results are aggregated to obtain a score for the system's performance. Sandra produces very detailed benchmarking results, which are analyzed and compared with the results obtained for other similar components and systems.

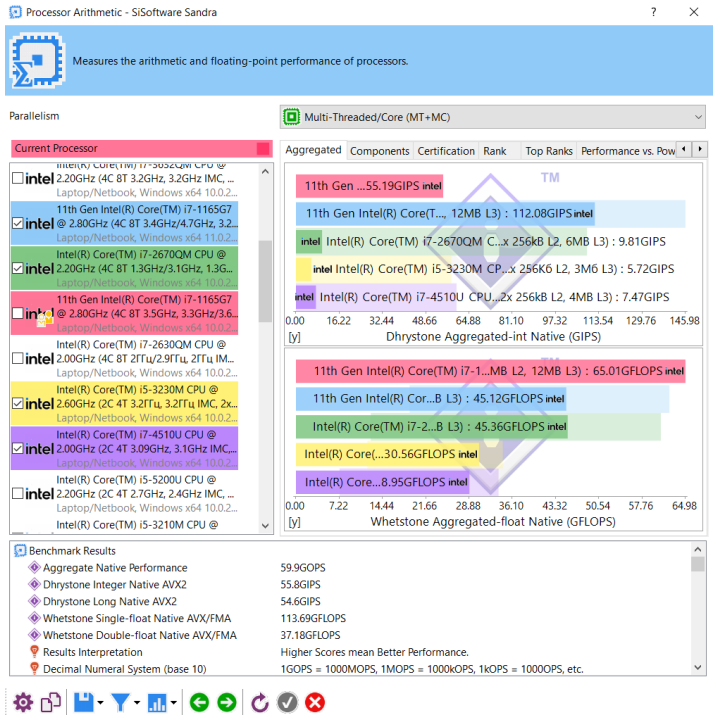

Figure 5: SiSoftware Sandra analysis results for processor arithmetic

6 Exercises

1. Use benchmark applications to evaluate the performance of a desktop computer.

Step 1. Identify the configuration of a computer using CPU-Z. Asses the Single thread performance and Multi-thread performance using CPU-Z. Go to **Bench** tab and select **Bench CPU**.

Step 2. Assess the performance of the CPU, Memory, HDD with SuperPI, NovaBench and SiSoftware Sandra benchmarks.

Step 3. Compare the results obtained for the same computer with the benchmarks at Step 2.

Step 4. Compare the results obtained for 2-3 computers with the benchmarks at Step 2.

2. Create a benchmark using a compute-intensive program.

Step 1. Given the program in `Prime.cpp` (for Windows users) or `Primes_linux.c` (for Linux users) that calculates the prime numbers up to a limit, create a benchmark program that measures its execution time. The `Prime` program listed below takes two arguments. The first argument is the upper limit to the prime numbers interval. The second argument is the number of threads which are created to find prime numbers in parallel. Use `clock()` function from <`time.h`> to measure the execution time. Modify the program to compute the prime numbers without using threads (remove the threading part) and measure the execution time.

Step 2. Measure the computation time in case of no threads, a single thread and in case of multiple threads. Perform at least 5 experiments using different values for the upper limit parameter. For each value of the upper limit measure the execution time in the case of no threads, 1 thread, 2, 4, 8, 16, 32 and 64 threads. Record the time measurements as shown in Table 1.

Step 3. Based on the time measurements recorded, plot a line graph which shows the variation of execution time (y-axis) with the number of threads used (x-axis) for all experiments. Plot an individual line for each different value of the upper limit. All lines should be plotted within the same graph.

Step 4. Compare the computation times. Is the computation 2/4/8/16/32/64 times faster for 2/4/8/16/32/64 threads compared to a single thread? Does the execution using no threads result in the same execution time as using a single thread? Discuss on the results.

Upper limit value	Execution time (clock cycles)							
	No threads	1 thr.	2 thrs.	4 thrs.	8 thrs.	16 thrs.	32 thrs.	64 thrs.
L_1								
L_2								
L_3								
L_4								
L_5								

Table 1: Execution time measurements table

Note: For Linux users, please use the `-lpthread` flag when compiling the code in `Primes_linux.c` as shown below.

```
gcc Primes_linux.c -o <exec_name> -lpthread
```

References

[1] SPEC's Benchmarks and Tools [accessed Sept. 2024], https://www.spec.org/benchmarks.html

[2] CPUID [accessed Sept. 2024], https://www.cpuid.com/softwares/cpu-z.html

[3] Super Pi [accessed Sept. 2024], https://www.techpowerup.com/download/super-pi/

[4] NovaBench [accessed Sept. 2024], https://novabench.com/

[5] SiSoftware Sandra [accessed Sept. 2024], https://www.sisoftware.co.uk/

[6] AnandTech [accessed Sept. 2024], https://www.anandtech.com/

[7] The Vishera Review: AMD FX-8350, FX-8320, FX-6300 and FX-4300 Tested [accessed Sept. 2024], https://www.anandtech.com/show/6396/the-vishera-review-amd-fx8350-fx8320-fx6300-and-fx4300-tested

[8] The Clarkdale Review: Intel's Core i5 661, i3 540 & i3 530 [accessed Sept. 2024], https://www.anandtech.com/show/2901

[9] Legacy CPU Benchmarks [accessed Sept. 2024], https://www.anandtech.com/bench/CPU/2

[10] Mushkin XP2 PC2-5300 DDR2 – Xtreme Performance Memory [accessed Sept. 2024], https://www.anandtech.com/show/1979

[11] Western Digital Caviar Green 3TB and My Book Essential 3TB Drives Reviewed [accessed Sept. 2024], https://www.anandtech.com/show/3981/western-digital-caviar-green-3tb-and-my-book-essential-3tb-drives-reviewed