https://github.com/letshpcorg?language=python

CS301-HPC Lab 1 (2024)

CPU architecture, Triad, Measuring Performance

Performance of a code is primarily characterized by a number of metrics:

- The rate at which floating-point operations can be executed (FLOPs/sec)
- The rate at which data can be moved between various levels of memory (GB/sec)
- The rate at which energy is used by your application (Watts/ Flops)

In determining hardware performance, we use a mixture of theoretical and empirical measurements. Theoretical value provides an upper bound to performance and the empirical confirming what can be achieved in a simplified kernel in close to actual operating conditions.

Actual data related to hardware performance can be found here for intel and amd:

https://ark.intel.com/content/www/us/en/ark.html

https://www.amd.com/en/products/ specifications/processors

Some other commands for probing hardware details are '**lscpu**' on Linux systems, '**wmic**' on Windows, and '**sysctl**' or '**system_profiler**' on Mac. The linux **lscpu** command outputs a consolidated report of the information from the */proc/cpuinfo* file. You can see the full information for every logical core by viewing */proc/cpuinfo* directly.

Measuring bandwidth using the Stream Benchmark

The stream benchmark measures the time to read and write a large array. Depending on the operations performed on the data by the CPU as it is being read, there are four variants. These are copy, scale, add and triad measurement. The copy does no floating point work, the scale and add do one arithmetic operation, and the triad does two. In this regime, the flop rate is limited by how fast memory can be loaded.

		Bytes (simplified view)	Arithmetic Ops
Copy	a(i)=b(i)	16	0
Scale	a(i)=q*b(i)	16	1
Sum	a(i)=b(i)+c(i)	24	1
Triad	a(i)=b(i)+q*c(i)	24	2

A. Understanding your CPU architectures (personal machine, LAB 207 and HPC Cluster) using `lscpu or data available in intel/amd website`

The goal of this exercise is to explore and understand the hardware architecture in a given Linux Computer.

1. Open the Terminal
2. Type the command **`lscpu`**
3. Several important parameters will be listed.

The following are some important parameters to take note of:

- Architecture
- Byte Order

· CPU(s) : Clock (GHz); Throughput (ops - generally 4); Peak FP rate MultAdd (FLOPS/Cycle)

· Model Name

· L1, L2, L3 cache

. Main Memory - Calculate Bandwidth (using the following data, will be discussed in tomorrow's lecture)

Channels/ Memory clock [MHz] / Bytes/ clock → Bandwidth [GB/s]

Make a table and compare the three architectures.

B. benchmarking

As seen in the previous exercise, the CPU memory is divided into 4 parts:

1. L1 cache
2. L2 cache
3. L3 cache
4. RAM Memory

These are listed in the order of the bandwidth/ latency at which they operate.
The goal of this exercise is to explore the impact of the memory hierarchy on the computation speed of a CPU. For this, we will use the **Stream Benchmark (copy, scale, sum, triad)**

Sample Vector Triad benchmarking is given below - rest can be designed accordingly.

This benchmark consists of performing a vector sum between 3 vectors and storing it in another vector. Consider 4 vectors **A**, **B**, **C**, **D** of size **N** each.

Pseudocode for Vector triad is as follows:

```
for i=1 to N
     A[i] = B[i] + C[i]*D[i]
```

For benchmarking purposes we will measure the time taken by the vector triad for different problem sizes **N**.

```
minSize = 2^8 maxSize
= 2^29 total = maxSize
for size =minSize; size<maxSize; size*=2
RUNS = total/size //initialise arrays
double A[size], B[size], C[size], D[size]
start_time = clock()
```

```
for j=1 to RUNS
    for i=1 to size
            A[i] = B[i] + C[i]*D[i]
    end_time = clock() - start_time
    throughput = (sizeof(double)*2*total)/end_time
    print size, throughput
```

Using python/matlab/gnuplot make the necessary plots and investigate the following:

1. Make the **necessary modifications in the code provided to you** to measure the run-time accurately. What are the couple of clock functions that can be used to measure the elapsed time? Understand how the clock works (in the code) and how it is used to measure runtime.
2. **Make Bandwidth (GB/s) vs <u>working data-set size (problem size)</u> plot for the four cases in STREAM benchmark.**
3. **Make Performance vs working data-set plot for the three cases in STREAM benchmark.**
4. Make modifications in the code to approximately measure the time taken by the computation part (in the Triad).
5. Make modifications in the code to approximately measure the time taken by the data access part (Triad).
6. How does your measured performance compare with the peak-theoretical performance of the computing system?