LABORATORY TUTORIAL #1

COMPILER OPTIONS AND IMPACT ON EXECUTION TIME, POWER DISSIPATION, AND ENERGY CONSUMPTION

V1.0, Sept. 2023, revised in Oct. 2024

Course: "Efficient Heterogeneous Computing"
Master Program in Informatics and Computing Engineering (M.EIC)
Faculty of Engineering of the University of Porto (FEUP), Porto, Portugal

SUMMARY

In these first lab experiments, we will evaluate the impact of some compiler optimizations on the execution time, power dissipation, and energy consumption of a given kernel in the context of a Windows OS.

MAIN GOALS

- analyze the impact of typical compiler options (including the parallel execution of multiple threads) on the execution time, power dissipation, and energy consumption of a given program

- understand some of the tools that can be used to support power and energy estimations

INVOLVES

- experiments with a provided C program, the use of the GNU gcc compiler, and the execution of the program in a desktop or laptop with Windows OS and an Intel CPU

- be aware of the impact on execution time, power dissipation, and energy consumption of the execution using the multiple CPU cores

PLANNED TIME:

Around 2 hours

EXPERIMENTS AND RESULTS

The 2mm kernel from the PolyBench benchmark repository [3], and its OpenMP version [4], is the computing kernel we are going to use. It includes 2 Matrix Multiplications (D=AxB; E=CxD) and computes: alpha * A * B * C + beta * D. For these experiments we will use the STANDARD_DATASET and the DATA_TYPE float.i

We focus on the use of GNU gcc, its main compiler optimizations reflected in -Ox flags [2], and estimations of power dissipation, execution time and operating clock frequency provided by the Intel Power Gadget tool [1].

Compile with GNU gcc, and with the different command line options indicated, and run each application binary with Power Gadget and collect and fill in the following data. Since Power Gadget provides power dissipation and energy

consumption at specific sampling resolutions[1] (although this can be modified with a tool option, we suggest using the default sampling rate).

Note that from the power dissipation provided by power dissipation estimations (or current and voltage) or measurements and from the execution time, one can estimate the energy consumption by using the equation E=PxT, where P represents the average power dissipation during the execution of the application and T represents the time required to execute the application.

COLLECT THE CHARACTERISTICS OF THE COMPUTING SYSTEM

First, let's describe the main characteristics of the machine used for the experiments. Fill in Table I. One possibility is to use the Windows command line (via **cmd** or **powershell**) and tools such as: **systeminfo**, **wmic**[2], **msinfo32**, and **dxdiag**.

Let's use wmic to obtain the computing system characteristics:

- **wmic** /? (list the options of the tool)
- **wmic** MEMCACHE /? (list the options regarding MEMCACHE)
- **wmic** MEMCACHE
- **wmic** COMPUTERSYSTEM /?
- **wmic** COMPUTERSYSTEM
- **wmic** memorychip list full
- **wmic** MEMORYCHIP
- **wmic** MEMORYCHIP /?
- **wmic** MEMPHYSICAL /?
- **wmic** MEMPHYSICAL
- **wmic** MEMPHYSICAL list full

Particularly:

- wmic CPU Get Caption,Name,DeviceID,NumberOfCores,NumberOfLogicalProcessors, ThreadCount,L2CacheSize,L3CacheSize,MaxClockSpeed,CurrentClockSpeed
- wmic memorychip get devicelocator,manufacturer,capacity,speed,banklabel,memorytype,partnumber

Table I. Characteristics of the machine

CPU used	No. physical cores	No. logical cores	MaxClockFrequency (GHz)	Sizes of caches {L1, L2, L3 and}, bytes (Bi)	Size of Physical Memory: RAM (MBi)

OPERATING SYSTEM AND POWER SCHEME

[1] The logging sampling resolution ranges from 1 ms to 1000 ms and the default set to 100 ms [1].
[2] **wmic** is a command line utility of the Windows Management Instrumentation Command (WMIC), https://learn.microsoft.com/en-us/windows/win32/wmisdk/wmic

The windows tool powercfg allows users to control power settings on a local system and to also list the existent power schemes and the one currently active.

One can check the options of the tool using: powercfg /? (or powercfg | more)

And list the power schemes defined and the one active: powercfg /list

Identify the version of the operating system of the machine where you will run the experiments and the active power scheme. Fill in Table II.

Table II. Operating system and power scheme used.

Operating system (use, e.g., the windows command: winver)	**Power scheme active**	**Details about the power scheme active**

ESTIMATING THE IMPACT IN EXECUTION TIME, POWER DISSIPATION AND ENERGY CONSUMPTION

The estimations for Table III below can be performed using the Power Gadget tool[3] [1] for Windows OS and Intel CPUs (see the applied restrictions regarding Windows OS versions and CPUs). Power Gadget includes a command line tool named PowerLog:

```
PowerLog3.0.exe -file <results>.csv -cmd <program>.exe
```

The following are the metrics reported by PowerLog:

- System Time
- RDTSC
- Elapsed Time (sec)
- CPU Utilization(%)
- CPU Frequency_0(MHz)
- Processor Power_0(Watt)
- Cumulative Processor Energy_0(Joules)
- Cumulative Processor Energy_0(mWh)
- IA Power_0(Watt)
- Cumulative IA Energy_0(Joules)
- Cumulative IA Energy_0(mWh)
- Package Temperature_0(C)
- Package Hot_0
- DRAM Power_0(Watt)
- Cumulative DRAM Energy_0(Joules)
- Cumulative DRAM Energy_0(mWh)
- GT Power_0(Watt)

[3] Although for the goals of these lab experiments we use Power Gadget, we note that: "Intel® Power Gadget has been discontinued. As a replacement product, Intel recommends Intel® Performance Counter Monitor (Intel® PCM), which is an API used to monitor performance and energy metrics of various Intel® processors." (https://www.intel.com/content/www/us/en/developer/archive/tools/power-gadget.html)

- Cumulative GT Energy_0(Joules)
- Cumulative GT Energy_0(mWh)
- Package PL1_0(Watt)
- Package PL2_0(Watt)
- Package PL4_0(Watt)
- Platform PsysPL1_0(Watt)
- Platform PsysPL2_0(Watt)
- GT Frequency(MHz)
- GT Utilization(%)

The following is an example of the data in the csv file generated by PowerLog when executed for a specific program:

System Time	*RDTSC*	*Elapsed Time (sec)*	*CPU Utilization (%)*	*CPU Frequency_0 (MHz)*	*Processor Power_0 (Watt)*	*Cumulative Processor Energy_0 (Joules)*	*Cumulative Processor Energy_0 (mWh)*	...	*GT Utilization (%)*
07:10:21:934	*6,6E+12*	*0.109*	*23.000*	*1700*	*3.688*	*0.402*	*0.112*	...	*0.000*
07:10:22:059	*6,6E+12*	*0.234*	*17.000*	*1800*	*6.883*	*1.262*	*0.350*	...	*0.000*
...									
07:10:34:101	*6,63E+12*	*12.277*	*57.000*	*1500*	*7.733*	*67.240*	*18.678*	...	*0.000*

Total Elapsed Time (sec) = 12.276538
Measured RDTSC Frequency (GHz) = 1.992

Cumulative Processor Energy_0 (Joules) = 67.239502
Cumulative Processor Energy_0 (mWh) = 18.677639
Average Processor Power_0 (Watt) = 5.477074

Cumulative IA Energy_0 (Joules) = 54.370605
Cumulative IA Energy_0 (mWh) = 15.102946
Average IA Power_0 (Watt) = 4.428822

Cumulative DRAM Energy_0 (Joules) = 0.000000
Cumulative DRAM Energy_0 (mWh) = 0.000000
Average DRAM Power_0 (Watt) = 0.000000

Cumulative GT Energy_0 (Joules) = 0.395630
Cumulative GT Energy_0 (mWh) = 0.109897
Average GT Power_0 (Watt) = 0.032227

At the bottom of the csv file there is a summary of the global results which include:

- Total Elapsed Time (sec) = 12.276538
- Measured RDTSC Frequency (GHz) = 1.992
- Cumulative Processor Energy_0 (Joules) = 67.239502
- Average Processor Power_0 (Watt) = 5.477074
- Cumulative DRAM Energy_0 (Joules) = 0.000000
- Average DRAM Power_0 (Watt) = 0.000000

In order to calculate the speedup of the implementation with option i vs. an option selected as baseline (option A), we can use the following equation:

$$\text{Speedup} = \text{Texec(baseline)} / \text{Texec(option i)} \quad (1)$$

Note that speedup values greater than 1 represent an acceleration of the option i over the baseline and speedup values lower than 1 represent a slowdown of the option i over the baseline.

To calculate the % of Energy and Power reductions of option i vs baseline use:

$$\text{Energy-Reduction} = 100x(1-\text{Energy(baseline)} / \text{Energy(option-i)})\ \% \quad (2)$$

$$\text{Power-Reduction} = 100x(1-\text{Power(baseline)} / \text{Power(option-i)})\ \% \quad (3)$$

Note that negative values represent an increase in Energy/Power.

Table III. Results including estimations for execution time, power dissipation and energy consumption.

Type of implementation	Option	GNU gcc compiler flags (add -march=native)	Execution time (s)	Power dissipation (W)	Energy consumption (J)	Speedup (vs. single-thread -O0)	Power dissipation reduction (%) (vs. single-thread -O0)	Energy consumption reduction (%) (vs. single-thread -O0)
Single-thread	A	-O0				1.00	0.00	0.00
	B	-O1						
	C	-O2						
	D	-O3						
	E	-Ofast						
	F	-Os						
Multi-thread	G	-O0 -fopenmp						
	H	-O1 -fopenmp						
	I	-O2 -fopenmp						
	J	-O3 -fopenmp						
	L	-Ofast -fopenmp						
	K	-Os -fopenmp						

Calculate the EDP (Energy-Delay Product), ED^2P and ED^3P metrics and fill in the table below and highlight the highest and the lowest value for each metric. Note that E represents the Energy consumption and D (delay) represents the execution time.

Table IV. Energy-Delay Product (EDP)

Option	EDP (ExD)	ED^2P (ExDxD)	ED^3P (ExDxDxD)
A			
B			
C			
D			
E			
F			
G			
H			
I			
J			
L			
K			

Note 1: an interesting experiment would be to change the power scheme, mainly the maximum and minimum operating clock frequency, and check the impact of that on the execution time, power dissipation, and energy consumption.

Note 2: an interesting experiment would be to verify the impact of not compiling for the native ISA of the target CPU (i.e., without using GCC option **march=native** and thus generating possibly non-portable binaries).

Note 3: additional experiments can include the analysis of the impact of bigger data sizes and double precision floating-point (double data type)-

Note 3: the impact on the results and conclusions of using higher logging sampling resolution, e.g., instead of 100 ms to use 1 ms and 10 ms.

ANALYSIS OF THE RESULTS

Analyze the results reported and draw conclusions about them and considering the following:

1) Report the best option for each of the following goals:
   a) lowest energy consumption;
   b) lowest power dissipation;
   c) lowest execution time.

Think and try to understand the options provided for each of these goals and analyze the impact of the execution of more than one CPU core (OpenMP-based multithreaded implementations).

2) Regarding the EDP, ED^2P, and ED^3P metrics and their highest and lowest values, why do you think these metrics can be helpful?
3) Share your results with other groups of colleagues and compare them. Are the options for each goal the same? Justify your answer.
4) Let's assume the following scenarios:
   a) A supercomputer system and supposing that one needs to minimize the speed or even avoid as much as possible running the cooling fan;

b) An embedded system where one wants to keep battery life as long as possible;
c) A cloud center where one needs to provide response times as fast as possible;

Which option would you suggest for each scenario above?

NOTES

As the Intel Power Gadget tool [1] for Windows OS and Intel CPUs has been discontinued, Intel recommends the Performance Counter Monitor (PCM), which includes a power utility [6].

TOOLS AND BENCHMARKS USED

[1] Intel® Power Gadget, 2019. https://www.intel.com/content/www/us/en/developer/articles/tool/power-gadget.html

[2] GNU GCC compiler: "3.11 Options That Control Optimization," https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

[3] Louis-Noel Pouchet, PolyBench/C, the Polyhedral Benchmark suite, © Ohio State University, USA. https://web.cse.ohio-state.edu/~pouchet.2/software/polybench/

[4] PolyBench-ACC, Copyright (c) 2012-2014 University of Delaware, USA. https://github.com/cavazos-lab/PolyBench-ACC

[5] "Powercfg command-line options." Microsoft. https://learn.microsoft.com/en-us/windows-hardware/design/device-experiences/powercfg-command-line-options

[6] "Intel® Performance Counter Monitor - A Better Way to Measure CPU Utilization," https://www.intel.com/content/www/us/en/developer/articles/tool/performance-counter-monitor.html

BIBLIOGRAPHY

[7] João M.P. Cardoso, José Gabriel F. Coutinho, and Pedro C. Diniz, Embedded Computing for High Performance: Efficient Mapping of Computations Using Customization, Code Transformations and Compilation, Morgan Kauphman, 2017. ISBN: 978-0-12-804189-5 . https://doi.org/10.1016/C2015-0-00283-0 **[Chapter 2 and especially Section 2.6] Online resources: https://web.fe.up.pt/~jmpc/books/hpec/**