ΕΛΠ 221: Οργάνωση Υπολογιστών
Εργασία 6

BENCHMARK 508.namd_r

Βαλεντίνος Παρίζα

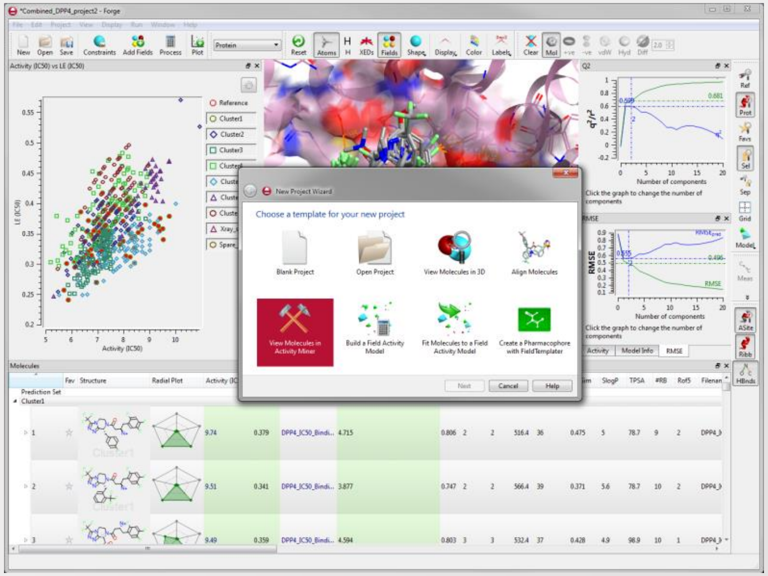
Μάριος Παφίτης

909759, 911719

Description of Benchmark:

- This Benchmark ,which is known as 508.namd_r is derived-taken from the program NAMD (Basically from the data layout and inner loop of NAMD) and is a program for simulating large biomolecular systems(more specific, simulating millions of atoms).
- NAMD is the abbreviation for Nanoscale Molecular Dynamics ,and it is a computer software for molecular dynamics simulation (δυναμική μοριακή προσομοίωση), build with Charm++. NAMD scales to over 200,000 cores for very large systems in order to simulate efficiently its biomolecular systems(using all the parallel capabilities that PCs offers).
- In this benchmark analysis the benchmark was tested in single performance. Because of the efficiency of this program, its maturing and of the importance of the operations that this program does this program was used as a compact benchmark for CPU2017.
- This program makes a lot calculations with floating point numbers.

This benchmark simulates proteins-atoms in biomolecular systems, and can be used to observe the interactions of the atoms inside these systems. But at this point it is used to examine the performance of a CPU at some situations.



Analysis/Association of time with `time` and `perf stat`

From the execution of the command “time” 10 times we took the averages :

Average Time:	
real	277.502
user	277.364
sys	0.052

Average IPC for 10 executions -> **IPC=2.57** .We know that **CPI=1/IPC** So **CPI= 0.389**

From the array of the next slide we took the averages of the statistics
which are mentioned-used below for calculating the execution time(USER):

Execution time with CYCLES statistic from (perf stat) :

For 3.3 GHz (Standard): Cycles_Statistic*(Cycle_Time) = 307.471s Cycle_time=1/(3.3E+10)

For 3.7 GHz (Turbo Boost): Cycles_Statistic*(Cycle_Time) = 274.231s Cycle_time=1/(3.7E+10)

Now calculating execution time with CYCLES=INSTRUCTIONS*CPI :

Cycles_Calculated = CPI*instructions = 0.389 *2,598,612,782,571 = 1,010,860,372,420.119

For 3.3 GHz (Standard): Cycles_Calculated*(Cycle_Time) = 306.321s Cycle_time=1/(3.3E+10)

For 3.7 GHz (Turbo Boost): Cycles_ Calculated *(Cycle_Time) = 273.206s Cycle_time=1/(3.7E+10)

CPU frequency→ (cycles / execution_time_from_command_time_User)/(10E+9) = 3.66GH

CPU frequency→ (cycles / execution_time_from_perf_stat)/(10E+9) = 3.62GH

=> The time multiplexing is a factor which changes (significantly) the statistics we estimate .

Mean/Standard Deviation of Statistics for 10 executions

Standard Deviation/ Average	*Average*	*Standard Deviation*	*Diversion Percentage (%)*
Time:	279.70	1.84	**0.66**
Instructions:	2598612782571	520287031	**0.02**
Cycles:	1014653654177	2808275224	**0.28**
Cache-References:	449971763	36352832	**8.08**
Cache Misses	94312208	8367359	**8.87**
Branch-Instructions:	42013192364	42036526	**0.10**
Branch-Misses:	1867356594	1295047	**0.07**
L1-dcache-loads	707313493358	91651835	**0.01**
L1-dcache-load-misses	31907089229	49679082	**0.16**
L1-dcache-stores:	242144245126	40051837	**0.02**
L1-icache-load-misses:	18399756	336468	**1.83**
LLC-loads:	381215876	36698914	**9.63**
LLC-load-misses:	76630123	4205048	**5.49**
LLC-stores:	39202624	4154680	**10.60**
LLC-store-misses:	16277009	2348003	**14.43**
dTLB-loads:	706276858458	203576469	**0.03**
dTLB-load-misses:	6467694	717839	**11.10**
dTLB-stores:	241604005925	47710291	**0.02**
dTLB-store-misses:	2118082	133914	**6.32**
iTLB-loads:	38456	7586	**19.73**
iTLB-load-misses:	408499	76830	**18.81**

Branch_Instructions appear 1.6% of the instructions

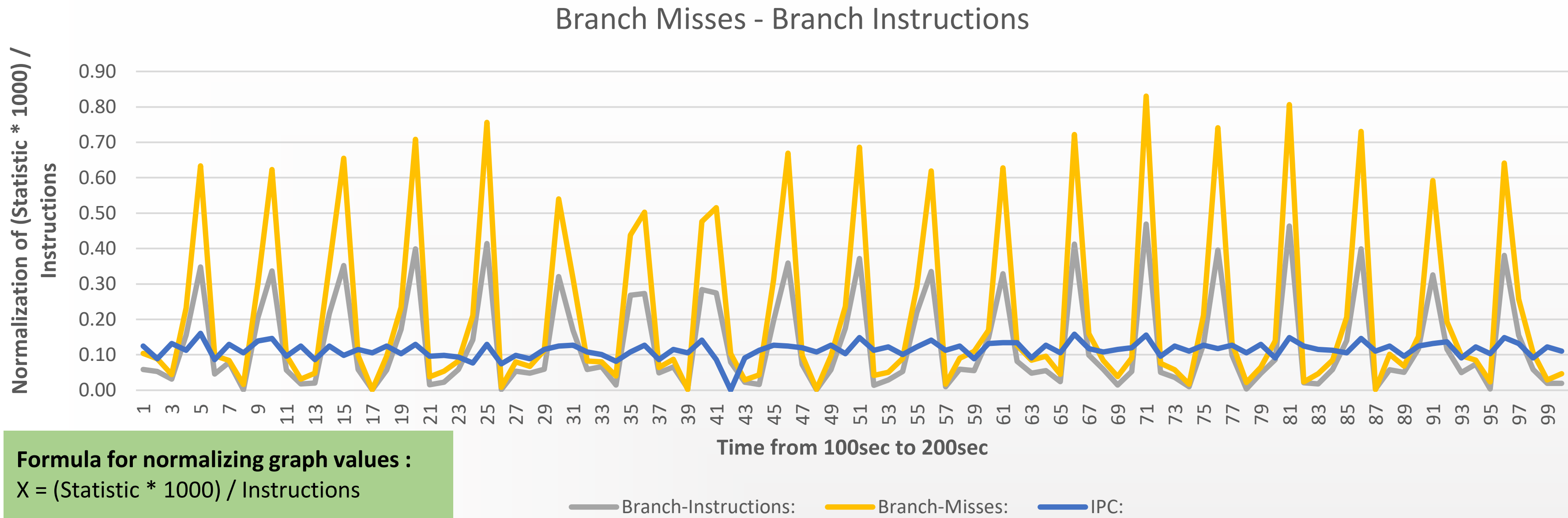
Branch_Misses appear 4.44% of Branch_Instructions

L1 dcache-loads appear 27.2% of the instructions

L1-dcache-load misses appear 4.5% of L1-cache-loads

Depends every time from the processes that run on PC and also from time multiplexing

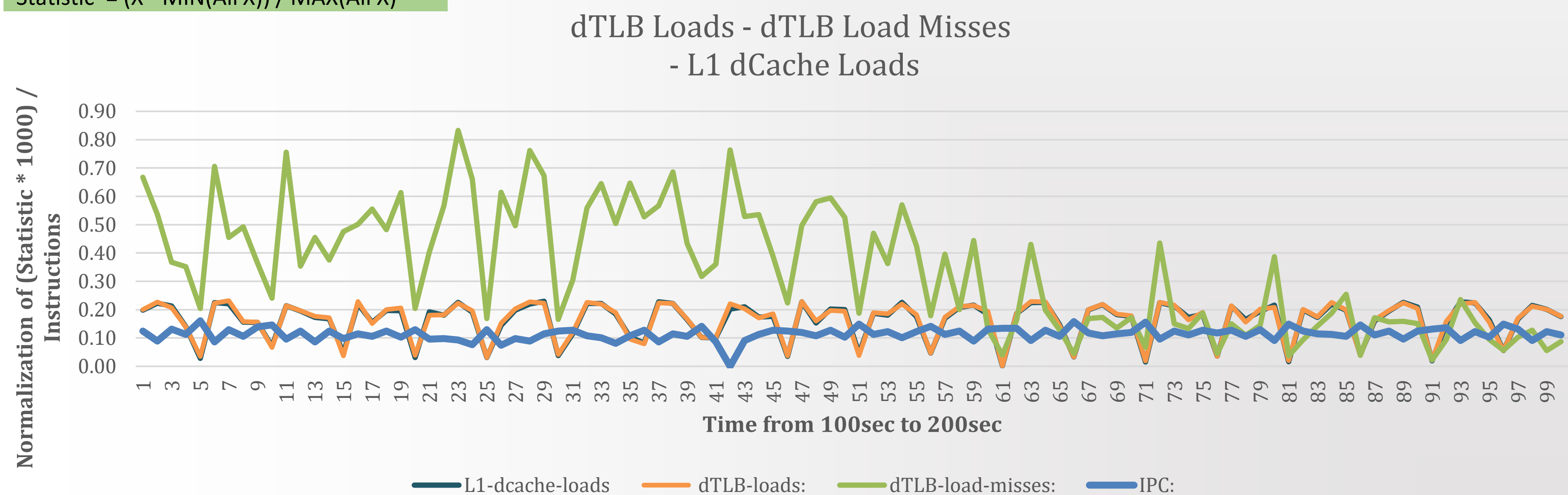
IPC Correlation Statistics



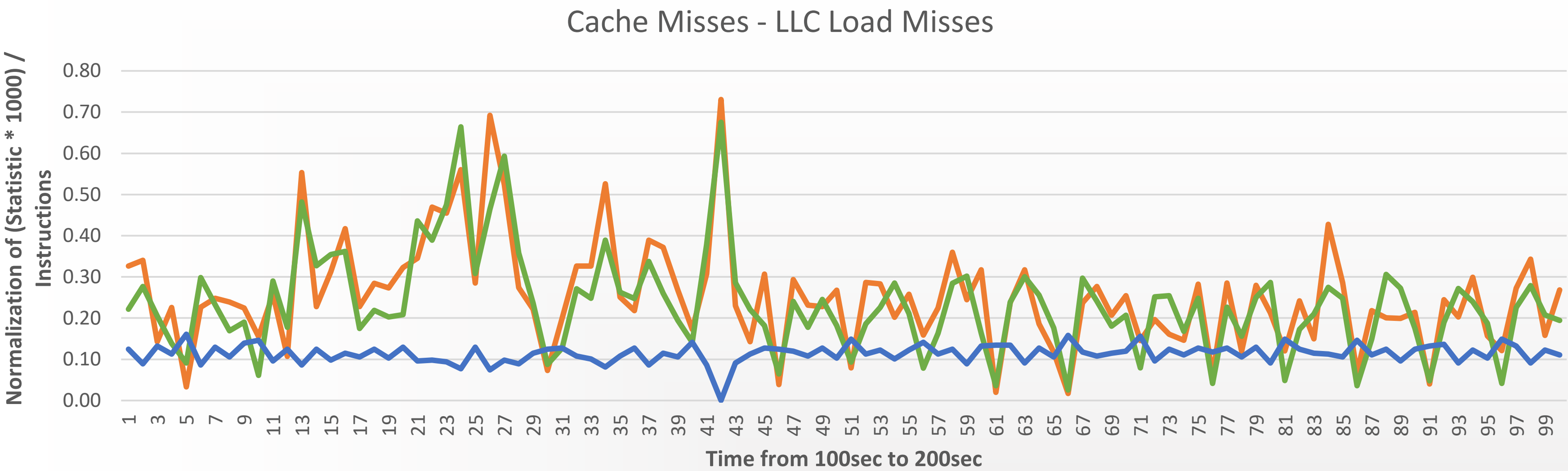
Formula for normalizing graph values :

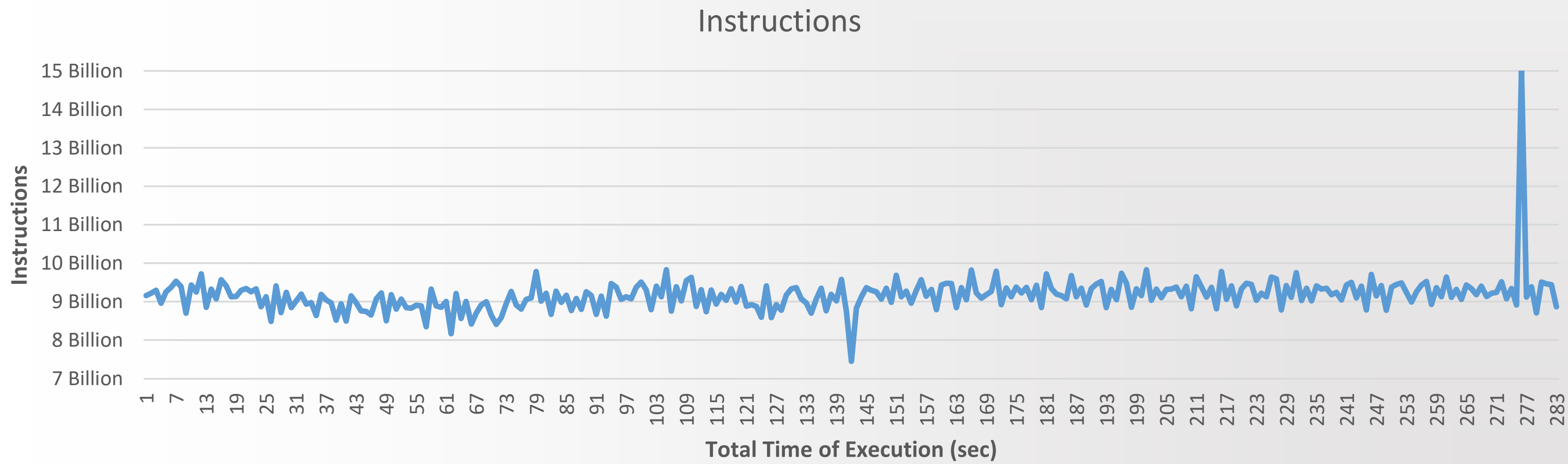
X = (Statistic * 1000) / Instructions

Statistic' = (X - MIN(All X)) / MAX(All X)



IPC Correlation Statistics





IPC Correlation Statistics

Correlation	IPC:
Time:	0.144
Instructions:	0.996
Cycles:	0.907
Cache-References:	-0.019
Cache Misses	-0.286
Branch-Instructions:	0.444
Branch-Misses:	0.432
L1-dcache-loads	0.518
L1-dcache-load-misses	0.177
L1-dcache-stores:	0.009
L1-icache-load-misses:	0.091
LLC-loads:	-0.077
LLC-load-misses:	-0.359
LLC-stores:	-0.023
LLC-store-misses:	0.039
dTLB-loads:	0.518
dTLB-load-misses:	-0.292
dTLB-stores:	0.010
dTLB-store-misses:	0.023
iTLB-loads:	0.052
iTLB-load-misses:	0.089

IPC=Instructions/Cycles

Processor operates in different frequencies as we can see from the cycles per seconds. So As the cycles increases-> the instructions increases respectively making changing respectively the IPC

Conclusions

The programs depends not only on software, but there are many factors that can affect significantly the performance of a program

- We have observed and realized that we can recognize some phases of the program (like functions with many branches , phases handling a lot of data) by observing how the program behaves during the execution .By observing we can derive many information about the behavior of a program
- Last we have realized that many factors that we think affect the hardware , at the end might not affect it, as we think, but with a different way
- We have seen how different behaviors ,(like the number of branch-misses doesn't affect directly IPC) of a program can significantly affect other parts.

We should observe the behavior of hardware during the execution of a program. We should look deeper than the abstraction of the code of the software.