CS4110: Computer System Design Lab

Lab1 - Using the perf tool

CS14B050 - Rachit Garg

Introduction

Perf tool was used to analyze the performance of two programs. The two programs were matrix multiplication programs, written such that one scenario had more cache hits than the other. The programs were run on two different system configurations.

The output of the programs and observations were as follows.

Experiment

A shell script was written in python to run the program multiple times and the average of statistics were calculated. This was done for two different system configurations.

Observations

System Configuration - 1

Architecture:	x86_64
CPU op-mode(s):	32-bit, 64-bit

Byte Order: Little Endian

CPU(s): 4

On-line CPU(s) list: 0-3

Thread(s) per core: 2

Core(s) per socket: 2

Socket(s): 1

NUMA node(s): 1

Vendor ID: GenuineIntel

CPU family: 6

Model: 69

Model name: Intel(R) Core(TM) i5-4200U CPU @ 1.60GHz

Stepping: 1

CPU MHz: 800.058

CPU max MHz: 2600.0000

CPU min MHz: 800.0000

BogoMIPS: 4589.22

Virtualization: VT-x

L1d cache: 32K

L1i cache: 32K

L2 cache: 256K

L3 cache: 3072K

NUMA node0 CPU(s): 0-3

Scenario #	1
Average-runtime (in milliseconds)	2673.5

Instructions	4636824819.1
Branche-instructions	293025173.9
cpu-cycles	5355750781.1
cache-references	270912750.8
cache-misses	112997479.3
L1-dcache-loads	855677277.3
L1-dcache-load-misses	269993457.4
dTLB-loads	827665815.3
dTLB-load-misses	47047.3
LLC-loads	268665455.3
LLC-load-misses	109506952.6

Scenario #	2
Average-runtime (in milliseconds)	522.5
Instructions	4158560043.2
Branche-instructions	291835238.7
cpu-cycles	1397646870.7
cache-references	20315467.4
cache-misses	2647181.2
L1-dcache-loads	1091539475.8

L1-dcache-load-misses	66361106.1
dTLB-loads	1089644897.0
dTLB-load-misses	1253.1
LLC-loads	19136315.2
LLC-load-misses	2063568.0

We observe that in this system configuration, the average-runtime of scenario 2 is much less than the average-runtime of scenario 1. The perf statistics that were observed showed that the percentage of cache-misses in scenario 2 were much lesser than those in scenario 1, and hence it could be the reason for the faster output of the program.

System Configuration - 2

Architecture: x86_64
CPU op-mode(s): 32-bit, 64-bit
Byte Order: Little Endian
CPU(s): 4
On-line CPU(s) list: 0-3
Thread(s) per core: 2
Core(s) per socket: 2
Socket(s): 1
NUMA node(s): 1
Vendor ID: GenuineIntel
CPU family: 6
Model: 78
Model name: Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz
Stepping: 3

CPU MHz: 421.781

CPU max MHz: 2800.0000

CPU min MHz: 400.0000

BogoMIPS: 4799.92

Virtualization: VT-x

L1d cache: 32K

L1i cache: 32K

L2 cache: 256K

L3 cache: 3072K

NUMA node0 CPU(s): 0-3

Flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch epb intel_pt tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt xsaveopt xsavec xgetbv1 dtherm ida arat pln pts hwp hwp_notify hwp_act_window hwp_epp

Scenario #	1
Average-runtime (in milliseconds)	4793.1
Instructions	4606216060.1
Branche-instructions	286467542.8
cpu-cycles	7699254576.6
cache-references	1079797113.4
cache-misses	471862578.1
L1-dcache-loads	866458787.2

L1-dcache-load-misses	384587028.3
dTLB-loads	832684883.0
dTLB-load-misses	5643.4
LLC-loads	271323634.2
LLC-load-misses	174640485.4

Scenario #	2
Average-runtime (in milliseconds)	1174.5
Instructions	4094022512.4
Branche-instructions	288825319.2
cpu-cycles	3273731836.2
cache-references	148083843.9
cache-misses	73603817.9
L1-dcache-loads	1124105172.1
L1-dcache-load-misses	68079524.5
dTLB-loads	1088629872.9
dTLB-load-misses	1568.6
LLC-loads	223573.5
LLC-load-misses	50783.3

We observe that in this system configuration too, the average-runtime of scenario 2 is much less than the average-runtime of scenario 1. The perf statistics that were observed showed that the percentage of cache-misses in scenario 2 were much lesser than those in scenario 1, and hence it could be the reason for the faster output of the program.

Since both the systems suggest the same observation, we conclude that more cache-hits can make the programs more faster since it is faster for the data to be loaded from the cache than from the main memory.
We also observe that there were a difference in the two system configurations. Statistics of system 1 were much faster than system 2.