實驗三　Intel VTune Performance Profiler

9617145 資工 4C 許晏峻

◎ **實驗目的：**

熟悉 VTune profiler，並且以此工具找出程式的瓶頸。

◎ **步驟過程：**

1. 根據附檔 Intel(R) VTune(TM) Amplifier XE 2011 Getting Started Tutorials.pdf (page 18~32)，將 VTune 所附的範例 (tachyon_find_hotspots) 操作一次，並分析 GCC 在不同最佳化等級的 (-O0、-O2) 差別。
2. 根據附檔 org 操作一遍，分析四種編譯執行狀況：原始狀態、auto_unroll、manual_8、manual_32。
   auto_unroll 為在編譯時加入自動執行 loop unrolling 選項 -funroll-all-loops。
   manual_8 為手動修改原始碼，將 work() loop 展開 8 次。
   manual_32 為手動修改原始碼，將 work() loop 展開 32 次。
3. 根據附檔 org 操作一遍，分析三種編譯執行狀況：原始狀態、prefetch、自由修改。
   prefetch 為修改原始碼，加入 prefetch 指令。

◎ **數據結果：**

1. 步驟 1 數據結果：
   CPU：Intel(R) Core(TM)2 Quad CPU　Q8200　2.336 GHZ

	-O0	**-O2**
CPU_CLK_UNHALTED.CORE	98 074 000 000	56 084 000 000
INST_RETIRED.ANY	153 296 000 000	93 198 000 000
BR_INST_EXEC	11 760 000 000	10 775 000 000
BR_MISSP_EXEC	258 000 000	311 500 000
BUS_TRANS_ANY.ALL_AGENTS	392 000 000	20 500 000
L1D_ALL_CACHE_REF	89 525 000 000	32 730 000 000
L1D_PEND_MISS	8 920 000 000	8 875 000 000
L1D_PREFETCH.REQUESTS	297 000 000	228 000 000
L1I_MISSES	3 500 000	500 000
L1I_READS	96 720 000 000	55 500 000 000
L2_LD.SELF.DEMAND.MESI	890 000 000	869 000 000
L2_RQSTS.BOTH_CORES.PREFETCH.MESI	2 250 500 000	981 500 000

	-O0	-O2
Execution time	40.810 s	25.856 s
Execution time reduction (%)	-	36.6429797 %
CPI	0.639768813 c/i	0.60177257 c/i
Branch prediction hit rate	97.80611224 %	97.1090487 %
L1 I$ miss rate	0.036187 %	0.0009009 %
L1 D$ miss rate	9.9636973 %	27.1157959 %
L2 cache miss rate	60.4532326 %	11.4620479 %
Bus request count	392000000	20500000
Memory bandwidth	153687821.612349914 bytes/sec	12685643.564356436 bytes/sec

2. 步驟 2 數據結果：

CPU：Intel(R) Core(TM)2 Quad CPU Q8200 2.527 GHZ

	org	auto_unroll	manual_8	manual_32
CPU_CLK_UNHALTED.CORE	14 896 000 000	14 922 000 000	12 178 000 000	12 100 000 000
INST_RETIRED.ANY	25 064 000 000	25 076 000 000	25 074 000 000	25 036 000 000
BR_INST_EXEC	1 045 000 000	1 040 000 000	145 000 000	60 000 000
BR_MISSP_EXEC	1 000 000	1 000 000	1 000 000	< 500 000
BUS_TRANS_ANY.ALL_AGENTS	72 500 000	88 500 000	64 500 000	82 500 000
L1D_ALL_CACHE_REF	16 655 000 000	17 325 000 000	13 200 000 000	12 400 000 000
L1D_PEND_MISS	210 000 000	215 000 000	190 000 000	675 000 000
L1D_PREFETCH.REQUESTS	61 000 000	60 500 000	60 000 000	45 500 000
L1I_MISSES	< 500 000	< 500 000	< 500 000	500 000
L1I_READS	14 390 000 000	14 190 000 000	11 910 000 000	11 880 000 000
L2_LD.SELF.DEMAND.MESI	62 000 000	61 500 000	62 000 000	61 500 000
L2_RQSTS.BOTH_CORES.PREFETCH.MESI	148 500 000	171 000 000	144 500 000	121 500 000

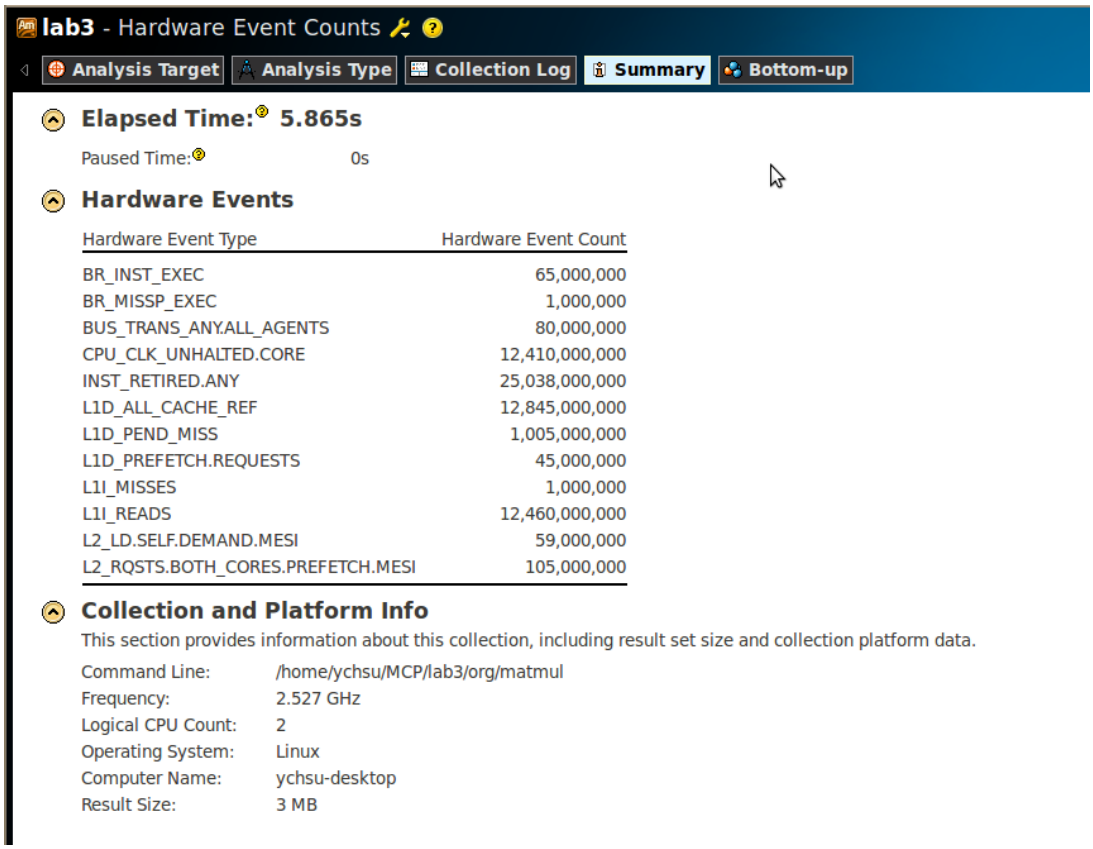
	org	auto_unroll	manual_8	manual_32
Execution time	8.810 s	8.989 s	7.610 s	7.466 s
Execution time reduction (%)	-	- 1.9913227 %	13.6208854 %	15.2553916 %
CPI	0.5943185445 c/i	0.5950709842 c/i	0.4856823802 c/i	0.4833040422 c/i
Branch prediction hit rate	99.90430622 %	99.903846154 %	99.310344828 %	> 99.166666667 %
L1 I$ miss rate	< 0.003474635 %	< 0.003523608 %	< 0.004198153 %	0.004208754 %
L1 D$ miss rate	1.1260888262 %	1.240981241 %	1.4393939394 %	5.4435483871 %
L2 cache miss rate	41.750841751 %	35.964912281 %	42.9065743945 %	50.6172839501 %
Bus request count	72500000	88500000	64500000	82500000
Memory bandwidth	131668558.4562996595 bytes/sec	157525864.9460451663 bytes/sec	135611038.1077529566 bytes/sec	176801500.1339405304 bytes/sec
Improvement (%)	-	- 1.9913227 %	13.6208854 %	15.2553916 %

3. 步驟 3 數據結果：

CPU：Intel(R) Core(TM)2 Quad CPU Q8200 2.527 GHZ

	org	prefetch	自由修改
CPU_CLK_UNHALTED.CORE	14 896 000 000	14 930 000 000	12 410 000 000
INST_RETIRED.ANY	25 064 000 000	25 086 000 000	25 038 000 000
BR_INST_EXEC	1 045 000 000	1 035 000 000	65 000 000
BR_MISSP_EXEC	1 000 000	2 000 000	1 000 000
BUS_TRANS_ANY.ALL_AGENTS	72 500 000	86 500 000	80 000 000
L1D_ALL_CACHE_REF	16 655 000 000	17 100 000 000	12 845 000 000
L1D_PEND_MISS	210 000 000	190 000 000	1 005 000 000
L1D_PREFETCH.REQUESTS	61 000 000	61 500 000	45 000 000
L1I_MISSES	< 500 000	500 000	1 000 000
L1I_READS	14 390 000 000	14 220 000 000	12 460 000 000
L2_LD.SELF.DEMAND.MESI	62 000 000	62 000 000	59 000 000
L2_RQSTS.BOTH_CORES.PREFETCH.MESI	148 500 000	180 000 000	105 000 000

	org	prefetch	自由修改
Execution time	8.810 s	8.757 s	5.865
Execution time reduction (%)	-	0.601589103 %	33.427922815 %
CPI	0.5943185445 c/i	0.5951526748 c/i	0.4956466171 c/i
Branch prediction hit rate	99.90430622 %	99.80676329 %	98.46153846 %
L1 I$ miss rate	< 0.003474635 %	0.003516174 %	0.008025682 %
L1 D$ miss rate	1.1260888262 %	1.1111111111%	7.824056053 %
L2 cache miss rate	41.750841751 %	34.44444444 %	56.19047619 %
Bus request count	72500000	86500000	80000000
Memory bandwidth	131668558.4562996595 bytes/sec	158044992.5773666678 bytes/sec	218243819.2668371697 bytes/sec
Improvement (%)	-	0.601589103 %	33.427922815 %



↑ lab3 由修改測試結果

◎ **討論 & 結論心得：**

Q1：在步驟 2 中：在這四種情況下，效能差意是否明顯，如果為是，請問是何種原因造成？校能改善多少？如果為否，請問為何校能改善幅度不大？

從數據可看出，auto-unroll 改善不明顯，可能的原因是因為現在的 CPU 對於編譯都有做某些最佳化，而 auto-unroll 就是其中一項，因此 gcc 並沒有確實做 auto-unroll，而是預期 CPU 會自行最佳化，所以經過多次測試可發現，平均表現是差不多的。而手動展開 8 次和 32 次，從數據中可看到 BR_INST_EXEC counter 和前兩種狀況有相當大的不同，branch 的次數大幅減少了，因此效能有些許提升，主要是減少 branch 的 overhead 所造成的，改善效能大約 10~15%。

Q2：在步驟 3 中：在這三種情況下，效能差意是否明顯，如果為是，請問是何種原因造成？校能改善多少？如果為否，請問為何校能改善幅度不大？

從數據可看出原始版本和 prefetch 版本效能都大同小異，個人認為是因為這個程式在執行的過程，work() 所需要的變數集中性很大，因此大部分其實都在 L2 cache 中，甚至是 L1 cache，因此若做 prefetch 也只是將變數先丟進 L2 cache，這樣其實就沒有差別。

而在自由修改版本我將 work() 迴圈展開 32 次，預期能達到步驟 2 中展開 32 次的效果，之後在適當地方也加入 prefetch 功能希望能提升 L2 cache miss 的機

會，解果比預期還要好，大部分執行時間在 5.8~7 秒之間，我想大部分是因為 branch 次數減少，也許 prefetch 在這個 case 也有發揮作用。

心得：
研究過程式碼後，覺得 work() 的演算法已經很精簡，沒有想到更好的算法來從方法面提升效能，除非利用多執行緒 (pthread.h 或 thread.h) 來以平行化增加整體運算效率，否則大概只有在編譯時讓編譯器做某些選項來提升 instruction schedule 方面的最佳化。