Assignment II - Valgrind & Pytorch profiler

1. Memcheck

Error 1 : Invalid write

```
Invalid write of size 4                          ##error 1 : invalid write
   at 0x1091C0: main (memleak.c:49)
 Address 0x4a96068 is 0 bytes after a block of size 40 alloc'd
   at 0x484884F: malloc (vg_replace_malloc.c:393)
   by 0x10919E: main (memleak.c:46)
```

這個錯誤是指程式想要對於 malloc 的記憶體以外的位置進行寫入的動作，這個例子是在 malloc 40 bytes 之後，在第 41 byte 的位置進行寫入。

Error 2 : Invalid read

```
Invalid read of size 4                           ##error 2 : invalid read
   at 0x1091ED: main (memleak.c:54)
 Address 0x4a96068 is 0 bytes after a block of size 40 alloc'd
   at 0x484884F: malloc (vg_replace_malloc.c:393)
   by 0x10919E: main (memleak.c:46)
```

這個錯誤是指程式想要對於 malloc 的記憶體以外的位置進行讀取的動作，這個例子是在 malloc 40 bytes 之後，在第 41 byte 的位置進行讀取。

Error 3 : Conditional jump or move depends on uninitialized

```
==93707== Conditional jump or move depends on uninitialised value(s)
##error 3 : Conditional jump or move depends on uninitialised value(s)
==93707==    at 0x48E0AD6: __vfprintf_internal (vfprintf-internal.c:1516)
==93707==    by 0x48CA79E: printf (printf.c:33)
==93707==    by 0x109214: main (memleak.c:57)
```

這個錯誤是指在條件判斷時使用了未初始化的數值，這個例子是在 printf 裡使用未初始化的變數或記憶體，且這個數值影響了條件的判斷。

Error 4 : Use of uninitialized value

```
Use of uninitialised value of size 8          ##error 4 : Use of uninitialised value
   at 0x48C42EB: _itoa_word (_itoa.c:177)
   by 0x48DFABD: __vfprintf_internal (vfprintf-internal.c:1516)
   by 0x48CA79E: printf (printf.c:33)
   by 0x109214: main (memleak.c:57)
```

這個錯誤是指程式想要讀取或使用操作一個未初始化的數值，這個例子可能是要 print 出一個變數，並且在 itoa 時用了一個未初始化的變數。

Error 5 : Argument ‘size’ of function malloc has a fishy value

```
==93707== Argument 'size' of function malloc has a fishy (possibly negative) value: -40
##error 5 : Argument 'size' of function malloc has a fishy (possibly negative) value
==93707==    at 0x484884F: malloc (vg_replace_malloc.c:393)
==93707==    by 0x109220: main (memleak.c:61)
```

這個錯誤是指在使用動態分配記憶體 malloc 時，給定的大小不是正數。這個例子是在呼叫 malloc 函數時給了-40 的大小。

Error 6 : Invalid free() / delete / delete[] / realloc()

```
==93707== Invalid free() / delete / delete[] / realloc()
##error 6 : Invalid free() / delete / delete[] / realloc()
==93707==    at 0x484B0C4: free (vg_replace_malloc.c:884)
==93707==    by 0x10924A: main (memleak.c:65)
==93707==  Address 0x4a964f0 is 0 bytes inside a block of size 40 free'd
==93707==    at 0x484B0C4: free (vg_replace_malloc.c:884)
==93707==    by 0x10923E: main (memleak.c:64)
==93707==  Block was alloc'd at
==93707==    at 0x484884F: malloc (vg_replace_malloc.c:393)
==93707==    by 0x10922E: main (memleak.c:63)
```

這個錯誤可能是指以下幾種錯誤，對同一個指標重複 free，對一個未分配記憶體的指標 free，free 一個 local variable。這個例子是對指向一塊大小為 40bytes 的記憶體的 pointer 呼叫了兩次 free()。

2. Cachegrind

Good:

```
==4418== Cachegrind, a cache and branch-prediction profiler
==4418== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote et al.
==4418== Using Valgrind-3.20.0 and LibVEX; rerun with -h for copyright info
==4418== Command: ./good
==4418== Parent PID: 3209
==4418==
--4418-- warning: L3 cache found, using its data for the LL simulation.
--4418-- warning: specified LL cache: line_size 64  assoc 16  total_size 12,582,912
--4418-- warning: simulated LL cache: line_size 64  assoc 24  total_size 12,582,912
==4418==
==4418== I   refs:      30,158,916
==4418== I1  misses:         1,069
==4418== LLi misses:         1,057
==4418== I1  miss rate:       0.00%
==4418== LLi miss rate:       0.00%
==4418==
==4418== D   refs:      14,053,786  (12,039,693 rd   + 2,014,093 wr)
==4418== D1  misses:       127,168  (    64,086 rd   +    63,082 wr)
==4418== LLd misses:        64,330  (     1,291 rd   +    63,039 wr)
==4418== D1  miss rate:        0.9% (       0.5%     +       3.1%  )
==4418== LLd miss rate:        0.5% (       0.0%     +       3.1%  )
==4418==
==4418== LL refs:          128,237  (    65,155 rd   +    63,082 wr)
==4418== LL misses:         65,387  (     2,348 rd   +    63,039 wr)
==4418== LL miss rate:         0.1% (       0.0%     +       3.1%  )
```

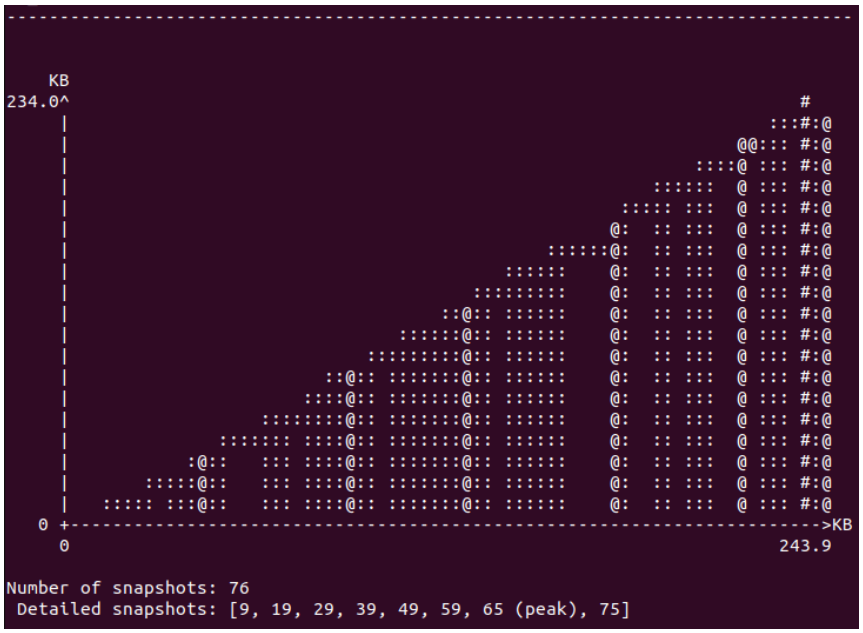
Bad:

```
==4436== Cachegrind, a cache and branch-prediction profiler
==4436== Copyright (C) 2002-2017, and GNU GPL'd, by Nicholas Nethercote et al.
==4436== Using Valgrind-3.20.0 and LibVEX; rerun with -h for copyright info
==4436== Command: ./bad
==4436== Parent PID: 3209
==4436==
--4436-- warning: L3 cache found, using its data for the LL simulation.
--4436-- warning: specified LL cache: line_size 64  assoc 16  total_size 12,582,912
--4436-- warning: simulated LL cache: line_size 64  assoc 24  total_size 12,582,912
==4436==
==4436== I   refs:      30,158,921
==4436== I1  misses:         1,070
==4436== LLi misses:         1,057
==4436== I1  miss rate:       0.00%
==4436== LLi miss rate:       0.00%
==4436==
==4436== D   refs:      14,053,786  (12,039,693 rd   + 2,014,093 wr)
==4436== D1  misses:     2,002,168  ( 1,001,586 rd   + 1,000,582 wr)
==4436== LLd misses:        64,330  (     1,291 rd   +    63,039 wr)
==4436== D1  miss rate:       14.2% (       8.3%     +      49.7%  )
==4436== LLd miss rate:        0.5% (       0.0%     +       3.1%  )
==4436==
==4436== LL refs:        2,003,238  ( 1,002,656 rd   + 1,000,582 wr)
==4436== LL misses:         65,387  (     2,348 rd   +    63,039 wr)
==4436== LL miss rate:         0.1% (       0.0%     +       3.1%  )
```

Good 和 Bad 的差別在於 D1 misses，這代表 bad 在 level 1 data cache 有很多的 miss。這有可能是因為 data access pattern 不同的緣故，bad 很可能是以 strided 或 random 的取法，導致 cache miss 一直發生。除此之外，也有可能是 good 的 data locality 比較好。而 D1 misses 也影響到了 LL refs，因為 cache 是從上層取到下層，而前面層數越多 miss，last level 也就有可能會有越多的 reference。

3. Massif

(1)



(2)

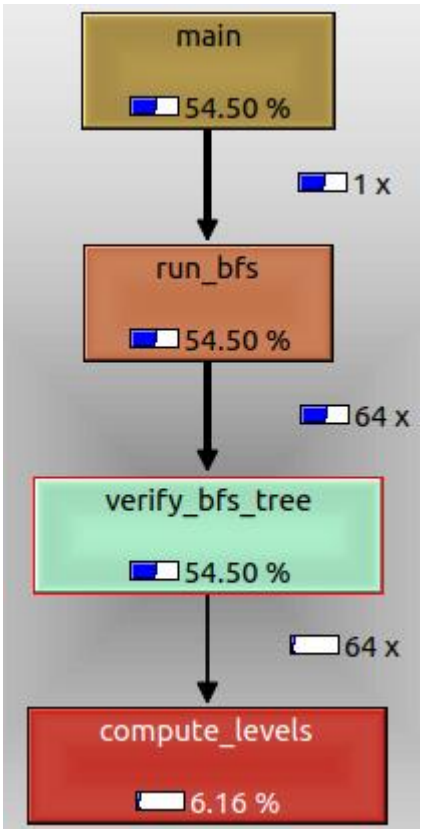
n	time(B)	total(B)	useful-heap(B)	extra-heap(B)	stacks(B)
60	225,568	225,568	225,000	568	0
61	229,576	229,576	229,000	576	0
62	231,584	231,584	231,000	584	0
63	235,592	235,592	235,000	592	0
64	239,600	239,600	239,000	600	0
65	239,600	239,600	239,000	600	0

Allocated: 239600 bytes Used: 239000 bytes

4. Callgrind

(1)

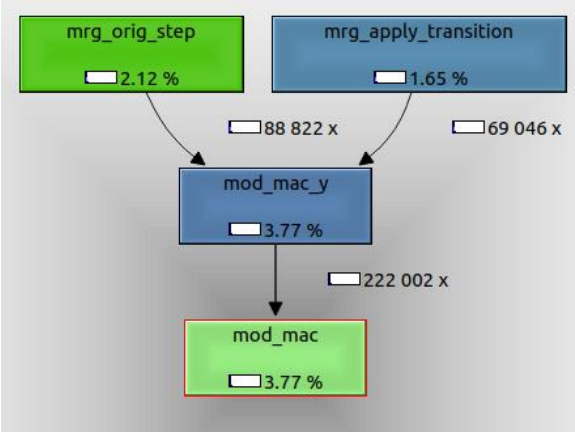
Incl.	Self	Called	Function
54.50	48.33	64	verify_bfs_tree
11.34	11.33	64	make_bfs_tree
6.16	6.16	64	compute_levels
3.77	3.77	222 002	mod_mac
16.43	3.13	12 288	make_one_edge
10.76	2.80	24 274	mrg_apply_transition
11.31	2.64	122 880	generate_4way_bernoulli
8.02	2.58	124 906	mrg_orig_step
2.49	2.49	173 454	mod_mul
1.89	1.89	72 822	mod_mac3
1.58	1.58	72 822	mod_mac2



當依照 self 來排序時，可以看到 verify_bfs_tree 有最高的 cost。

(2)

Incl.	Self	Called	Function
3.77	3.77	222 002	mod_mac
5.30	1.53	222 002	mod_mac_y
2.49	2.49	173 454	mod_mul
1.38	1.38	132 383	i64cmp
8.02	2.58	124 906	mrg_orig_step



用 called 來排序，可以看到 mod_mac 和 mod_mac_y 被呼叫最多次，mod_mac 的 caller 是 mod_mac_y，mod_mac_y 的 caller 是 mrg_orig_step 和 mrg_apply_transition。

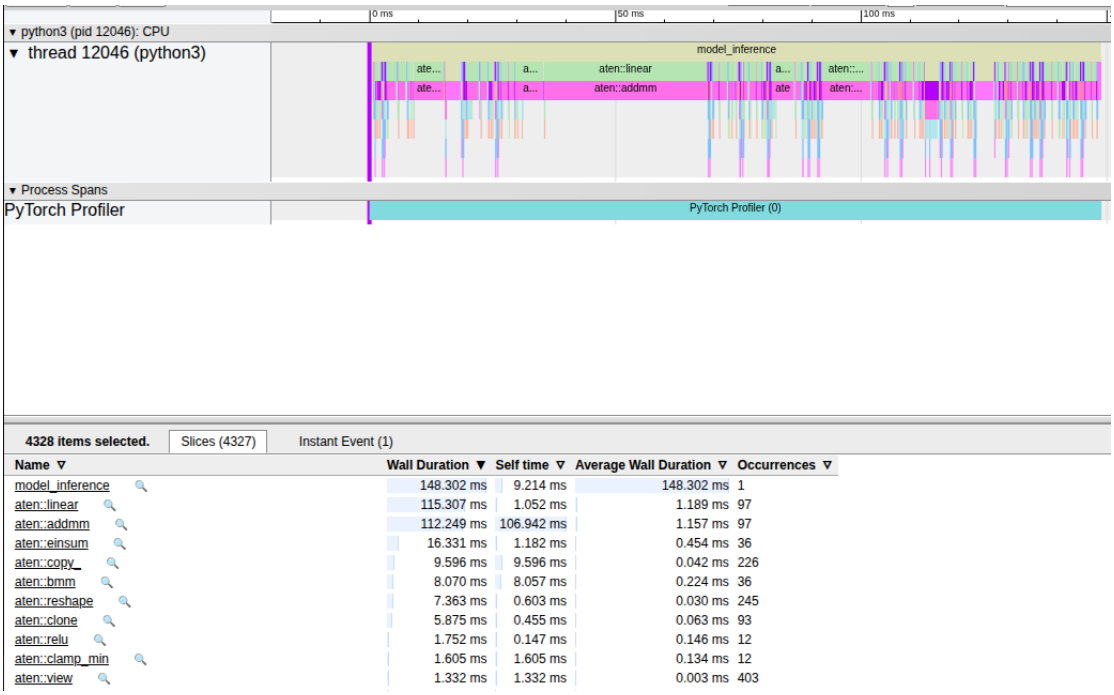
5. Pytorch profiler

(1)

```
karrs@MSS:~$ python3 test.py
cpu
torch.Size([2, 7, 10])
STAGE:2025-05-13 20:13:31 9172:9172 ActivityProfilerController.cpp:335] Completed Stage: Warm Up
STAGE:2025-05-13 20:13:31 9172:9172 ActivityProfilerController.cpp:342] Completed Stage: Collection
STAGE:2025-05-13 20:13:31 9172:9172 ActivityProfilerController.cpp:347] Completed Stage: Post Processing
-------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
                     Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg       CPU Mem  Self CPU Mem    # of Calls
-------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
              aten::addmm        65.38%      60.530ms        67.26%      62.273ms     641.990us       4.13 Mb       4.13 Mb            97
         aten::clamp_min         8.66%       8.021ms         8.66%       8.021ms     668.436us       1.50 Mb       1.50 Mb            12
          model_inference         7.11%       6.582ms       100.00%      92.582ms      92.582ms           0 b     -10.72 Mb             1
  aten::native_layer_norm         5.50%       5.088ms         5.71%       5.287ms     176.229us     939.66 Kb          72 b            30
             aten::copy_         2.89%       2.676ms         2.89%       2.676ms      11.840us           0 b           0 b           226
              aten::view         0.99%     917.073us         0.99%     917.073us       2.276us           0 b           0 b           403
               aten::add         0.99%     916.528us         0.99%     916.528us      28.642us    1000.00 Kb    1000.00 Kb            32
               aten::bmm         0.93%     864.130us         0.94%     873.837us      24.273us     624.38 Kb     624.38 Kb            36
            aten::einsum         0.79%     734.221us         5.05%       4.673ms     129.815us       2.32 Mb           0 b            36
            aten::linear         0.68%     632.436us        69.49%      64.331ms     663.203us       4.13 Mb           0 b            97
-------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 92.582ms
```

前三名分別是 aten::addmm、aten::clamp_min、aten::native_layer_norm，aten::addmm 是做矩陣乘法後加法；aten::clamp_min 是設定一個 lower bound，將小於該 lower bound 的資料回傳 lower bound；aten::native_layer_norm 是在實作 layer normalization。

(2)



4328 items selected. Slices (4327) Instant Event (1)

Name	Wall Duration	Self time	Average Wall Duration	Occurrences
model_inference	148.302 ms	9.214 ms	148.302 ms	1
aten::linear	115.307 ms	1.052 ms	1.189 ms	97
aten::addmm	112.249 ms	106.942 ms	1.157 ms	97
aten::einsum	16.331 ms	1.182 ms	0.454 ms	36
aten::copy_	9.596 ms	9.596 ms	0.042 ms	226
aten::bmm	8.070 ms	8.057 ms	0.224 ms	36
aten::reshape	7.363 ms	0.603 ms	0.030 ms	245
aten::clone	5.875 ms	0.455 ms	0.063 ms	93
aten::relu	1.752 ms	0.147 ms	0.146 ms	12
aten::clamp_min	1.605 ms	1.605 ms	0.134 ms	12
aten::view	1.332 ms	1.332 ms	0.003 ms	403

可以看到出現最多的是 aten::linear 和 aten::addmm，aten::linear 是用來實作全連接層；aten::addmm 是做矩陣乘法後加法。