Name: Chitwan Goel
Roll No. 210295

CS610 Assignment 2

1 Problem 1

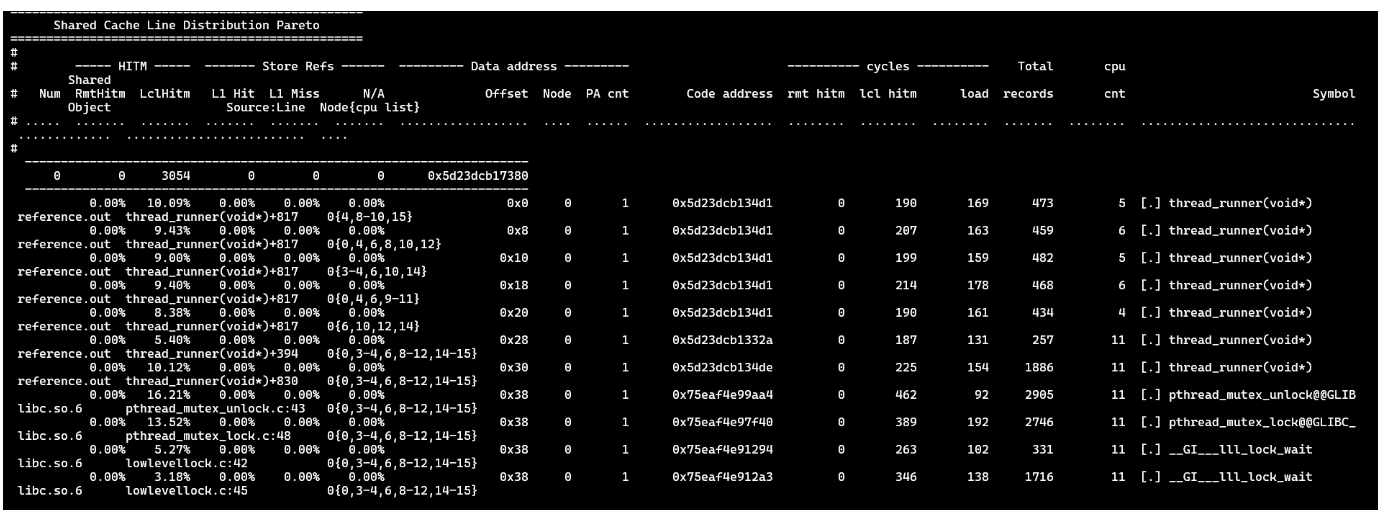
Performance Bugs Identified:

False Sharing

The `struct tracker` has `word_count` array whose size is equal to the number of threads. In the execution of the statement `tracker.word_count[thread_id]++;` (Line 142 in `reference.cpp`), false sharing is present which is clear from the code as well as from the generated PERF report.
To remove this false sharing, padding was done in the `word_count` array and its size was made equal to `NUM_THREADS` $\times$ `8` and instead of accessing the index `thread_id`, `thread_id*8` was accessed by thread having `tid` equal to `thread_id`. This multiplication by 8 ensured that the memory location accessed by each thread is on separate cache line (8 `uint_64` of 8 Bytes each take 64 Bytes, which is equal to the cache line size) and thus false sharing is minimised. The source file `padded.cpp` has these modifications.

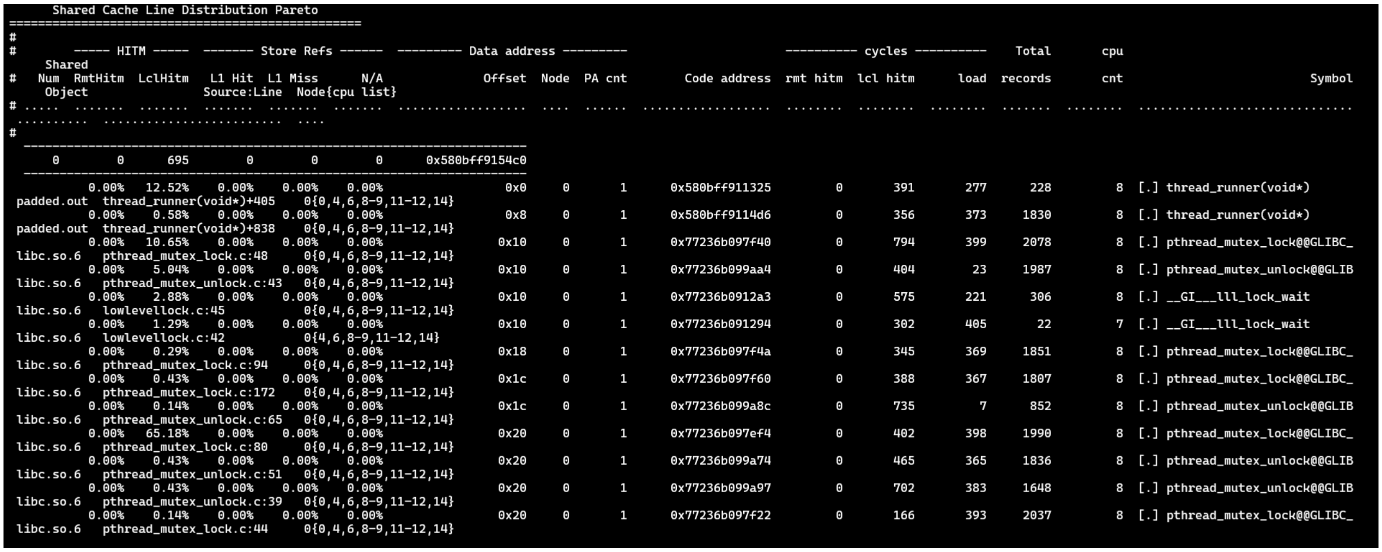
True Sharing

Taking and releasing lock for every word and line of the input file caused true sharing and was the cause of large number of HITMs (Lines 144-146 and 133-135 in `reference.cpp`). This was reduced by keeping thread-private variables for the words and line counts of each thread, and then later reducing them with the count in `tracker` struct, thus taking lock only once. This reduced the HITMs to nearly zero. The source file `improved.cpp` has these modifications. Also we have removed the redundant mutex `line_count_mutex` since the mutex in `struct tracker` is sufficient for the synchronisation needed.



```
=================================================
      Shared Cache Line Distribution Pareto
=================================================
#
#        ----- HITM -----  ------- Store Refs ------  --------- Data address ---------                      ---------- cycles ----------    Total       cpu
      Shared
#   Num  RmtHitm  LclHitm   L1 Hit  L1 Miss      N/A            Offset  Node  PA cnt        Code address  rmt hitm  lcl hitm      load  records       cnt                           Symbol
      Object                 Source:Line  Node{cpu list}
# .....  .......  .......  .......  .......  .......  .................  ....  ......  .................  ........  ........  ........  .......  ........  ............................
 .............  ........................  ....
#
  -----------------------------------------------------------------------
      0        0     3054        0        0        0     0x5d23dcb17380
  -----------------------------------------------------------------------
           0.00%   10.09%    0.00%    0.00%    0.00%               0x0     0       1      0x5d23dcb134d1         0       190       169      473         5  [.] thread_runner(void*)
 reference.out  thread_runner(void*)+817    0{4,8-10,15}
           0.00%    9.43%    0.00%    0.00%    0.00%               0x8     0       1      0x5d23dcb134d1         0       207       163      459         6  [.] thread_runner(void*)
 reference.out  thread_runner(void*)+817    0{0,4,6,8,10,12}
           0.00%    9.00%    0.00%    0.00%    0.00%              0x10     0       1      0x5d23dcb134d1         0       199       159      482         5  [.] thread_runner(void*)
 reference.out  thread_runner(void*)+817    0{3-4,6,10,14}
           0.00%    9.40%    0.00%    0.00%    0.00%              0x18     0       1      0x5d23dcb134d1         0       214       178      468         6  [.] thread_runner(void*)
 reference.out  thread_runner(void*)+817    0{0,4,6,9-11}
           0.00%    8.38%    0.00%    0.00%    0.00%              0x20     0       1      0x5d23dcb134d1         0       190       161      434         4  [.] thread_runner(void*)
 reference.out  thread_runner(void*)+817    0{6,10,12,14}
           0.00%    5.40%    0.00%    0.00%    0.00%              0x28     0       1      0x5d23dcb1332a         0       187       131      257        11  [.] thread_runner(void*)
 reference.out  thread_runner(void*)+394    0{0,3-4,6,8-12,14-15}
           0.00%   10.12%    0.00%    0.00%    0.00%              0x30     0       1      0x5d23dcb134de         0       225       154     1886        11  [.] thread_runner(void*)
 reference.out  thread_runner(void*)+830    0{0,3-4,6,8-12,14-15}
           0.00%   16.21%    0.00%    0.00%    0.00%              0x38     0       1      0x75eaf4e99aa4         0       462        92     2905        11  [.] pthread_mutex_unlock@@GLIB
 libc.so.6      pthread_mutex_unlock.c:43   0{0,3-4,6,8-12,14-15}
           0.00%   13.52%    0.00%    0.00%    0.00%              0x38     0       1      0x75eaf4e97f40         0       389       192     2746        11  [.] pthread_mutex_lock@@GLIBC_
 libc.so.6      pthread_mutex_lock.c:48     0{0,3-4,6,8-12,14-15}
           0.00%    5.27%    0.00%    0.00%    0.00%              0x38     0       1      0x75eaf4e91294         0       263       102      331        11  [.] __GI___lll_lock_wait
 libc.so.6      lowlevellock.c:42           0{0,3-4,6,8-12,14-15}
           0.00%    3.18%    0.00%    0.00%    0.00%              0x38     0       1      0x75eaf4e912a3         0       346       138     1716        11  [.] __GI___lll_lock_wait
 libc.so.6      lowlevellock.c:45           0{0,3-4,6,8-12,14-15}
```

Figure 1: PERF Report of Reference Version - Large number of HITMs in the `thread_runner` function



```
      Shared Cache Line Distribution Pareto
=================================================
#
#        ----- HITM -----  ------- Store Refs ------  --------- Data address ---------                      ---------- cycles ----------    Total       cpu
     Shared
#   Num  RmtHitm  LclHitm   L1 Hit  L1 Miss      N/A              Offset  Node  PA cnt        Code address  rmt hitm  lcl hitm      load  records       cnt                          Symbol
     Object                Source:Line  Node{cpu list}
# .....  .......  .......  .......  .......  .......  ..................  ....  ......  ..................  ........  ........  ........  .......  ........  ............................
 ..........  .........................  ....
#
  ----------------------------------------------------------------------
      0        0      695        0        0        0      0x580bff9154c0
  ----------------------------------------------------------------------
           0.00%   12.52%    0.00%    0.00%    0.00%                 0x0     0       1      0x580bff911325         0       391       277      228         8  [.] thread_runner(void*)
 padded.out  thread_runner(void*)+405    0{0,4,6,8-9,11-12,14}
           0.00%    0.58%    0.00%    0.00%    0.00%                 0x8     0       1      0x580bff9114d6         0       356       373     1830         8  [.] thread_runner(void*)
 padded.out  thread_runner(void*)+838    0{0,4,6,8-9,11-12,14}
           0.00%   10.65%    0.00%    0.00%    0.00%                0x10     0       1      0x77236b097f40         0       794       399     2078         8  [.] pthread_mutex_lock@@GLIBC_
 libc.so.6   pthread_mutex_lock.c:48     0{0,4,6,8-9,11-12,14}
           0.00%    5.04%    0.00%    0.00%    0.00%                0x10     0       1      0x77236b099aa4         0       404        23     1987         8  [.] pthread_mutex_unlock@@GLIB
 libc.so.6   pthread_mutex_unlock.c:43   0{0,4,6,8-9,11-12,14}
           0.00%    2.88%    0.00%    0.00%    0.00%                0x10     0       1      0x77236b0912a3         0       575       221      306         8  [.] __GI___lll_lock_wait
 libc.so.6   lowlevellock.c:45           0{0,4,6,8-9,11-12,14}
           0.00%    1.29%    0.00%    0.00%    0.00%                0x10     0       1      0x77236b091294         0       302       405       22         7  [.] __GI___lll_lock_wait
 libc.so.6   lowlevellock.c:42           0{4,6,8-9,11-12,14}
           0.00%    0.29%    0.00%    0.00%    0.00%                0x18     0       1      0x77236b097f4a         0       345       369     1851         8  [.] pthread_mutex_lock@@GLIBC_
 libc.so.6   pthread_mutex_lock.c:94     0{0,4,6,8-9,11-12,14}
           0.00%    0.43%    0.00%    0.00%    0.00%                0x1c     0       1      0x77236b097f60         0       388       367     1807         8  [.] pthread_mutex_lock@@GLIBC_
 libc.so.6   pthread_mutex_lock.c:172    0{0,4,6,8-9,11-12,14}
           0.00%    0.14%    0.00%    0.00%    0.00%                0x1c     0       1      0x77236b099a8c         0       735         7      852         8  [.] pthread_mutex_unlock@@GLIB
 libc.so.6   pthread_mutex_unlock.c:65   0{0,4,6,8-9,11-12,14}
           0.00%   65.18%    0.00%    0.00%    0.00%                0x20     0       1      0x77236b097ef4         0       402       398     1990         8  [.] pthread_mutex_lock@@GLIBC_
 libc.so.6   pthread_mutex_lock.c:80     0{0,4,6,8-9,11-12,14}
           0.00%    0.43%    0.00%    0.00%    0.00%                0x20     0       1      0x77236b099a74         0       465       365     1836         8  [.] pthread_mutex_unlock@@GLIB
 libc.so.6   pthread_mutex_unlock.c:51   0{0,4,6,8-9,11-12,14}
           0.00%    0.43%    0.00%    0.00%    0.00%                0x20     0       1      0x77236b099a97         0       702       383     1648         8  [.] pthread_mutex_unlock@@GLIB
 libc.so.6   pthread_mutex_unlock.c:39   0{0,4,6,8-9,11-12,14}
           0.00%    0.14%    0.00%    0.00%    0.00%                0x20     0       1      0x77236b097f22         0       166       393     2037         8  [.] pthread_mutex_lock@@GLIBC_
 libc.so.6   pthread_mutex_lock.c:44     0{0,4,6,8-9,11-12,14}
```

Figure 2: PERF Report of Padded Version - Small number of HITMs in the `thread_runner` function, majority of them are due to locking

```
=================================================
      Shared Cache Line Distribution Pareto
=================================================
#
                                                                                                  81,1          Bot
```

Figure 3: PERF Report of Improved Version - No HITMs

Version	HITMs	Time (s)
Reference	5904	1.651
Padded	1291	1.009
Improved	0	0.070

Table 1: Performance Comparison of three versions (Data taken on `csews172` with 11 MB files)

Running the Code

There is a Makefile and a bash script in the code. `make` generates the executables for three versions. `run.sh` generates the PERF report for all three versions.

```
make
./run.sh <path to the file which has the names of input files>
```

The three PERF reports will be generated as `perf_report_reference.out`, `perf_report_padded.out` and `perf_report_improved.out`.

Also, there is a file `generate_input.cpp` which takes an integer `i` as CLI and generates a large random text file `inp<i>.txt` in the `test1` directory. `generate_input.out` is also generated by the `make` command. Ensure that `test1` directory is present before running `generate_input.out`.

2 Problem 2

The directory `problem2-dir` contains the source files. Below is the compilation and execution commands -

```
g++ problem2.cpp -lpthread
./a.out -inp=<input_path> -thr=<num_producers> -lns=<lines_per_thread> -buf=<shared_buffer_size>
-out=<output_path>
```

3 Problem 3

```
for i = 1 , N - 2
    for j = i + 1 , N
        A (i , j - i ) = A (i , j - i - 1) - A ( i + 1 , j - i ) + A (i - 1 , i + j - 1)
```

We consider the following pairs and type of dependencies:

1. $A(i, j-i)$ and $A(i, j-i-1)$

$$i_0 = i_0 + \Delta i \implies \boxed{\Delta i = 0}$$
$$j_0 - i_0 = j_0 + \Delta j - i_0 - \Delta i - 1 \implies \Delta j = \Delta i + 1 \implies \boxed{\Delta j = 1}$$

   Thus the flow dependency is $(0, 1)$ or $(0, +)$, which is a valid flow dependency. Since, we have a valid flow dependency, the corresponding anti dependency would be invalid.

2. $A(i, j-i)$ and $A(i+1, j-i)$

$$i_0 = i_0 + \Delta i + 1 \implies \boxed{\Delta i = -1}$$
$$j_0 - i_0 = j_0 + \Delta j - i_0 - \Delta i \implies \Delta j = \Delta i \implies \boxed{\Delta j = -1}$$

   The flow dependency is $(-1, -1)$, but it is an invalid dependency since the first non-zero component of the vector is negative. However, there will be an anti dependency with the vector equal to $(1, 1)$.

3. $A(i, j-i)$ and $A(i-1, i+j-1)$

$$i_0 = i_0 + \Delta i - 1 \implies \boxed{\Delta i = 1}$$
$$j_0 - i_0 = i_0 + \Delta i + j_0 + \Delta j - 1 \implies \boxed{\Delta j = -2i_0}$$

   In this case, Δi is positive and $\Delta j = -2i_0$. Again since i_0 is always positive, Δj will be always negative. Thus the dependency vector will be $(+, -)$. It is a valid flow dependency and so there is no anti dependency.

There is no output dependency for any of the memory access pairs.