Department of Computer Science
The University of Hong Kong
COMP3230B: Principles of Operating Systems
Sample Solution of Assignment 2

Programming

Please refer to *agent_basic.c*, *agent_bonus.c* and *trs_bonus.c* for details.

Analysis

1. The reader_sem isn't used as a limit on the number of reading threads. It is used as mutex on variable reader_count.

   If reader_sem is initialized to 3, there may be data races on reader_count, which may lead to multiple invocations of sem_wait() and/or sem_post() on writer_sem. The program may run into deadlock or produce incorrect results.

2. A writer needs to compete against the reading threads and other writers for the writer_sem. It could happen that readers keep coming in, or the OS never assigns writer_sem to one of the writers.

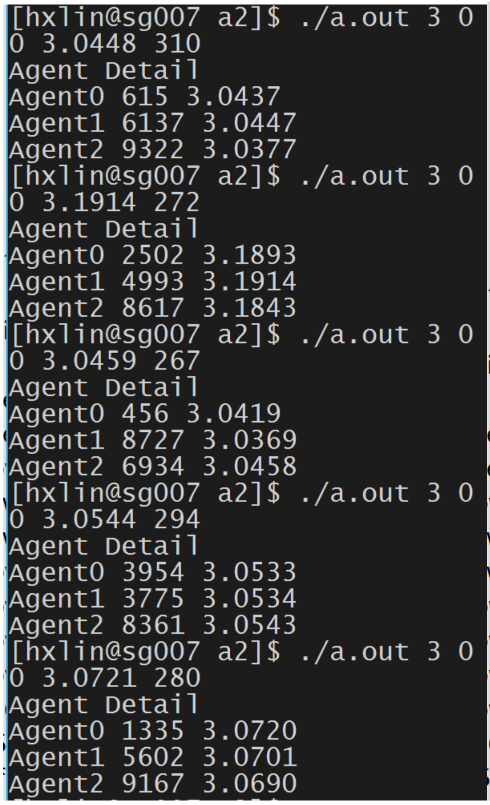
   The fundamental reason is that the scheme does not take writers' waiting time into consideration. More details can be found here:

   https://en.wikipedia.org/wiki/Readers%E2%80%93writers_problem

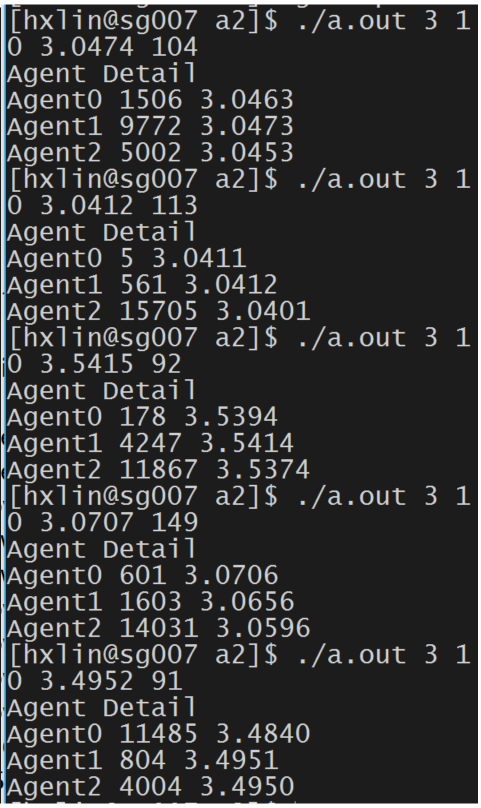
3. As shown in the figures below, best-fit leads to less vacancy. It is a greedy algorithm that minimizes the size of "bubbles" in the table, which proves quite effective.

   It is more likely that threads are targeting different seats than in the case of first-fit, so there will be less researching rounds.

   However, note that the increase in reading time affects load balancing.

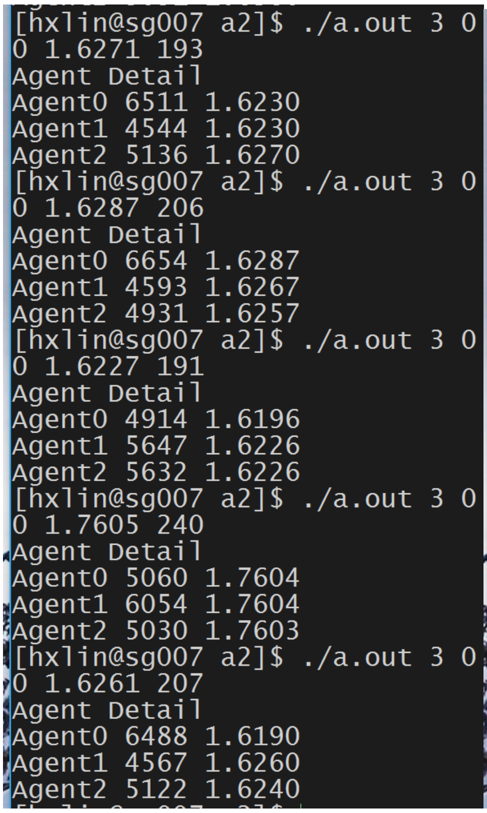


```
[hxlin@sg007 a2]$ ./a.out 3 0
0 3.0448 310
Agent Detail
Agent0 615 3.0437
Agent1 6137 3.0447
Agent2 9322 3.0377
[hxlin@sg007 a2]$ ./a.out 3 0
0 3.1914 272
Agent Detail
Agent0 2502 3.1893
Agent1 4993 3.1914
Agent2 8617 3.1843
[hxlin@sg007 a2]$ ./a.out 3 0
0 3.0459 267
Agent Detail
Agent0 456 3.0419
Agent1 8727 3.0369
Agent2 6934 3.0458
[hxlin@sg007 a2]$ ./a.out 3 0
0 3.0544 294
Agent Detail
Agent0 3954 3.0533
Agent1 3775 3.0534
Agent2 8361 3.0543
[hxlin@sg007 a2]$ ./a.out 3 0
0 3.0721 280
Agent Detail
Agent0 1335 3.0720
Agent1 5602 3.0701
Agent2 9167 3.0690
```

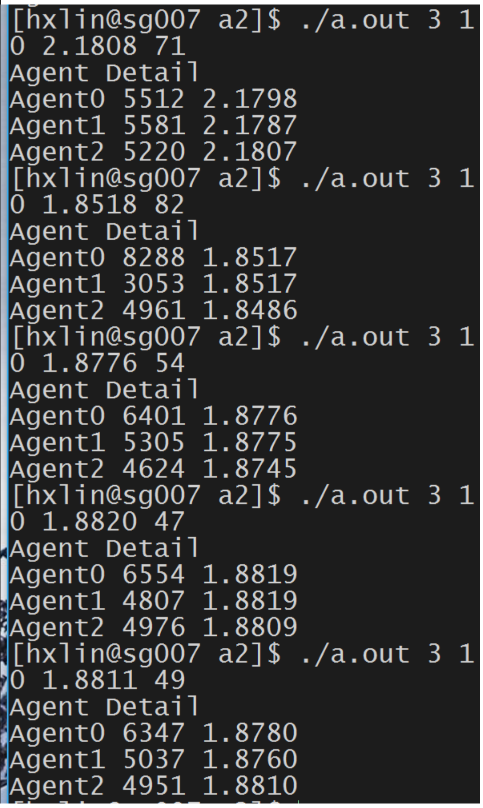


```
[hxlin@sg007 a2]$ ./a.out 3 1
0 3.0474 104
Agent Detail
Agent0 1506 3.0463
Agent1 9772 3.0473
Agent2 5002 3.0453
[hxlin@sg007 a2]$ ./a.out 3 1
0 3.0412 113
Agent Detail
Agent0 5 3.0411
Agent1 561 3.0412
Agent2 15705 3.0401
[hxlin@sg007 a2]$ ./a.out 3 1
0 3.5415 92
Agent Detail
Agent0 178 3.5394
Agent1 4247 3.5414
Agent2 11867 3.5374
[hxlin@sg007 a2]$ ./a.out 3 1
0 3.0707 149
Agent Detail
Agent0 601 3.0706
Agent1 1603 3.0656
Agent2 14031 3.0596
[hxlin@sg007 a2]$ ./a.out 3 1
0 3.4952 91
Agent Detail
Agent0 11485 3.4840
Agent1 804 3.4951
Agent2 4004 3.4950
```

Bonus



```
[hxlin@sg007 a2]$ ./a.out 3 0
0 1.6271 193
Agent Detail
Agent0 6511 1.6230
Agent1 4544 1.6230
Agent2 5136 1.6270
[hxlin@sg007 a2]$ ./a.out 3 0
0 1.6287 206
Agent Detail
Agent0 6654 1.6287
Agent1 4593 1.6267
Agent2 4931 1.6257
[hxlin@sg007 a2]$ ./a.out 3 0
0 1.6227 191
Agent Detail
Agent0 4914 1.6196
Agent1 5647 1.6226
Agent2 5632 1.6226
[hxlin@sg007 a2]$ ./a.out 3 0
0 1.7605 240
Agent Detail
Agent0 5060 1.7604
Agent1 6054 1.7604
Agent2 5030 1.7603
[hxlin@sg007 a2]$ ./a.out 3 0
0 1.6261 207
Agent Detail
Agent0 6488 1.6190
Agent1 4567 1.6260
Agent2 5122 1.6240
```



```
[hxlin@sg007 a2]$ ./a.out 3 1
0 2.1808 71
Agent Detail
Agent0 5512 2.1798
Agent1 5581 2.1787
Agent2 5220 2.1807
[hxlin@sg007 a2]$ ./a.out 3 1
0 1.8518 82
Agent Detail
Agent0 8288 1.8517
Agent1 3053 1.8517
Agent2 4961 1.8486
[hxlin@sg007 a2]$ ./a.out 3 1
0 1.8776 54
Agent Detail
Agent0 6401 1.8776
Agent1 5305 1.8775
Agent2 4624 1.8745
[hxlin@sg007 a2]$ ./a.out 3 1
0 1.8820 47
Agent Detail
Agent0 6554 1.8819
Agent1 4807 1.8819
Agent2 4976 1.8809
[hxlin@sg007 a2]$ ./a.out 3 1
0 1.8811 49
Agent Detail
Agent0 6347 1.8780
Agent1 5037 1.8760
Agent2 4951 1.8810
```

We can see that the load balancing is also improved greatly. It would also be interesting to look at the relationship between vacancy and load balancing.