Yueying Cui
01/22/2013

Indexing Product Review

Section I: Main Thread

1. **Data Structure**: Inverted Index
   a. dictionary

string word	list<int brand_id, int frequence>
“love”	[1,2] → [5,3] → [21, 1] → ⋯
“green”	[18,5] → [53,2]
⋮	⋮
“soft”	[4,5] → [99,2] → [212, 1] → ⋯

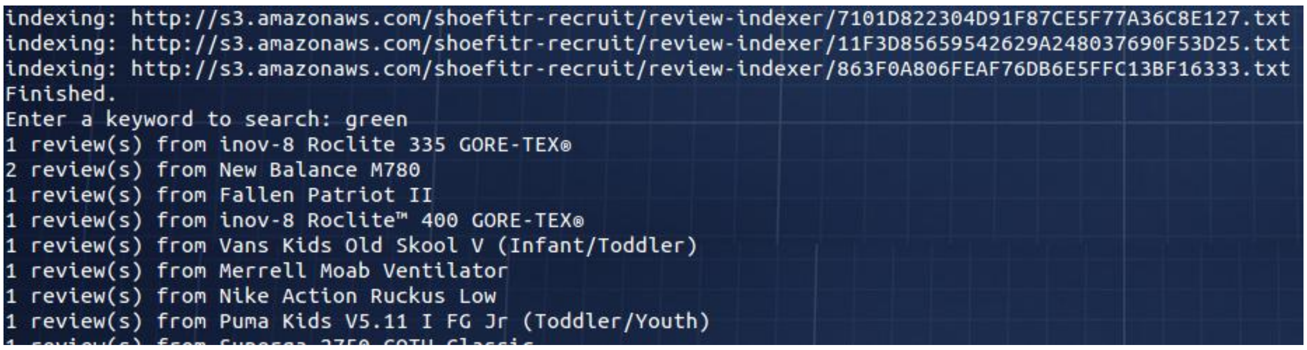
   b. brand_table

int brand_id	string brand_name
1	New Balance MR993
2	adidas Originals Kids AR 2.0 (Toddler/Youth)
⋮	⋮
n	The North Face Women's Hedgehog III

   Q: What will give you the biggest bang for your buck?
   A: “Inverted Index” saves more memory space than “M-d Trie”.

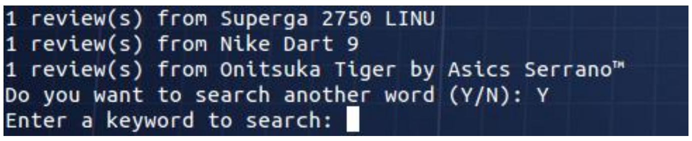
2. **Supported Features**:
   a. Case Insensitive (parse & search).
   b. Filter useless characters.
   c. Filter stop words.
   d. Enable the user to search many times “instantly” once the index is created.
   e. Scale for a data set of hundreds of reviews to a data set of tens of thousands of reviews.

3. **Results**:
   Searching result for keyword:



```
indexing: http://s3.amazonaws.com/shoefitr-recruit/review-indexer/7101D822304D91F87CE5F77A36C8E127.txt
indexing: http://s3.amazonaws.com/shoefitr-recruit/review-indexer/11F3D85659542629A248037690F53D25.txt
indexing: http://s3.amazonaws.com/shoefitr-recruit/review-indexer/863F0A806FEAF76DB6E5FFC13BF16333.txt
Finished.
Enter a keyword to search: green
1 review(s) from inov-8 Roclite 335 GORE-TEX®
2 review(s) from New Balance M780
1 review(s) from Fallen Patriot II
1 review(s) from inov-8 Roclite™ 400 GORE-TEX®
1 review(s) from Vans Kids Old Skool V (Infant/Toddler)
1 review(s) from Merrell Moab Ventilator
1 review(s) from Nike Action Ruckus Low
1 review(s) from Puma Kids V5.11 I FG Jr (Toddler/Youth)
1 review(s) from Superga 2750 COTU Classic
```

   Searching for another keyword:



```
1 review(s) from Superga 2750 LINU
1 review(s) from Nike Dart 9
1 review(s) from Onitsuka Tiger by Asics Serrano™
Do you want to search another word (Y/N): Y
Enter a keyword to search: █
```

Section II: Multiple Threads

1. **Library for multiple threads**: pthread.h

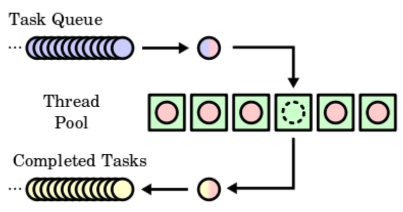
2. **Algorithm**: Thread-safe Queue

For any idle thread, if Task Queue is not empty,

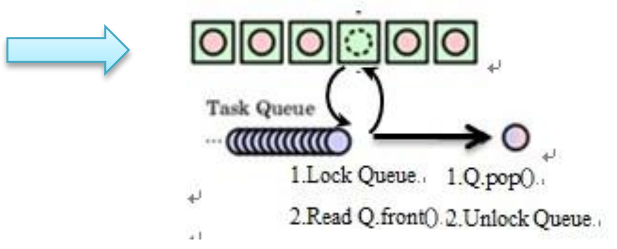
Step 1: Lock Task Queue → Read Q.front() [Other idle thread can't read data from Queue]

Step 2: Q.pop() → Unlock Task Queue [Other idle thread can read data from Queue]

Step 3: Process Inverted Index functions.



Typical Thread Pool



Thread-safe Queue

Q: What will give you the biggest bang for your buck?

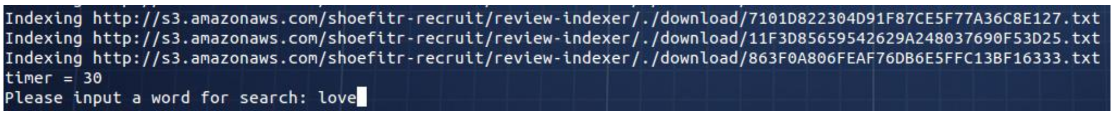
A: "Tread-safe Queue" is better than "Evenly assigns tasks to multiple threads" and "Typical Thread Pool" method.

1. "Thread-safe Queue" is much efficiency than "Evenly assign tasks to multiple threads". Since the efficiency of thread may vary widely to each other, it is not fair to assign the same amount of tasks to all threads. If so, the fast threads will complete their task early, and will be idle for a long time before the slow threads completed their task. Letting some threads be idle during the process really waste resources.
2. "Thread-safe Queue" is much simpler than the "Typical Thread Pool" method. In the "Typical Thread Pool", we need a special "guard thread" to control the occasions for Task Queue to assign tasks. It is so complicated to generate multiple kinds of locks and threads for this small project.

3. **Results**:

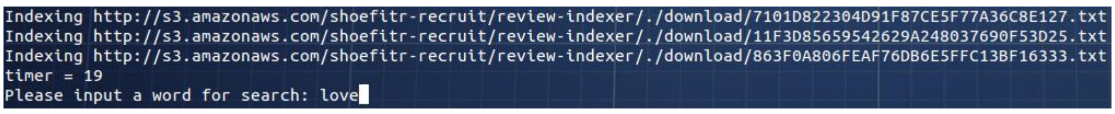
Since my PC is Core i5, it supports 2 CPUs and 4 virtual threads (virtual 4 CPUs). By changing the parameter "THREAD_NUM" in ".code/parallel/main.cpp", we can set different number of threads to leverage concurrency.

When THREAD_NUM = 1,



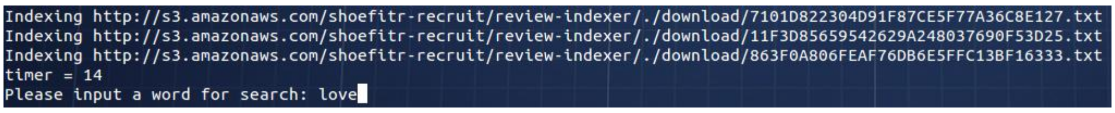
```
Indexing http://s3.amazonaws.com/shoefitr-recruit/review-indexer/./download/7101D822304D91F87CE5F77A36C8E127.txt
Indexing http://s3.amazonaws.com/shoefitr-recruit/review-indexer/./download/11F3D85659542629A248037690F53D25.txt
Indexing http://s3.amazonaws.com/shoefitr-recruit/review-indexer/./download/863F0A806FEAF76DB6E5FFC13BF16333.txt
timer = 30
Please input a word for search: love
```

When THREAD_NUM = 2,



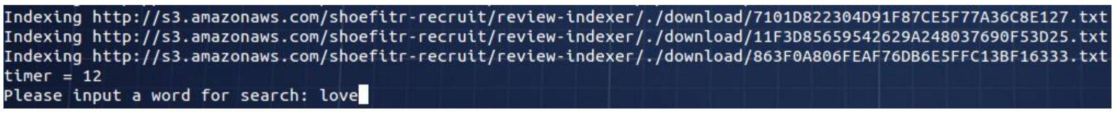
```
Indexing http://s3.amazonaws.com/shoefitr-recruit/review-indexer/./download/7101D822304D91F87CE5F77A36C8E127.txt
Indexing http://s3.amazonaws.com/shoefitr-recruit/review-indexer/./download/11F3D85659542629A248037690F53D25.txt
Indexing http://s3.amazonaws.com/shoefitr-recruit/review-indexer/./download/863F0A806FEAF76DB6E5FFC13BF16333.txt
timer = 19
Please input a word for search: love
```

When THREAD_NUM = 3,



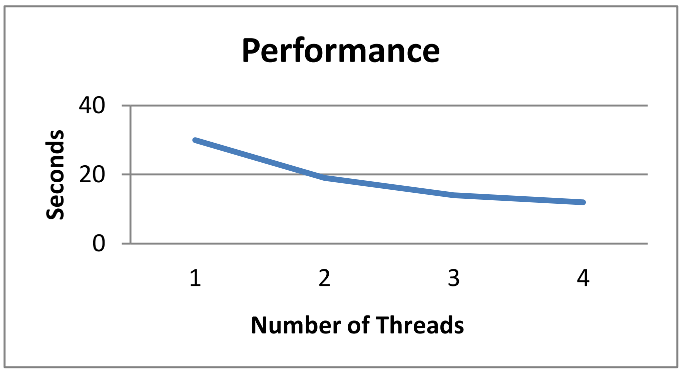
```
Indexing http://s3.amazonaws.com/shoefitr-recruit/review-indexer/./download/7101D822304D91F87CE5F77A36C8E127.txt
Indexing http://s3.amazonaws.com/shoefitr-recruit/review-indexer/./download/11F3D85659542629A248037690F53D25.txt
Indexing http://s3.amazonaws.com/shoefitr-recruit/review-indexer/./download/863F0A806FEAF76DB6E5FFC13BF16333.txt
timer = 14
Please input a word for search: love
```

When THREAD_NUM = 4,



```
Indexing http://s3.amazonaws.com/shoefitr-recruit/review-indexer/./download/7101D822304D91F87CE5F77A36C8E127.txt
Indexing http://s3.amazonaws.com/shoefitr-recruit/review-indexer/./download/11F3D85659542629A248037690F53D25.txt
Indexing http://s3.amazonaws.com/shoefitr-recruit/review-indexer/./download/863F0A806FEAF76DB6E5FFC13BF16333.txt
timer = 12
Please input a word for search: love
```

From the above result, I get



The above figure shows:

a. Using multiple threads speed up the indexing process doubly.

b. Only using one additional thread can speed up the indexing process by 1/3.

c. Generating the 3rd and 4th virtual threads is based on "Time round-robin scheduling method", thus these two virtual threads can't provide great efficiency to the indexing process as the real threads.

Q: How would your program perform for thousands of review files instead of just hundreds? What about hundreds of thousands?

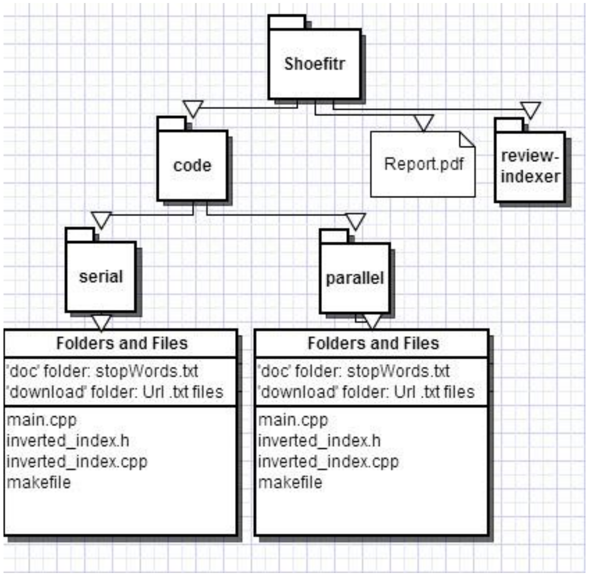
A:

1. From the above results, my program performs efficiently for thousands of review files.

2. If the PC's memory is not big enough for the much larger data set, I would like to use "External Merge" method to improve my program, make some exchange with disks.

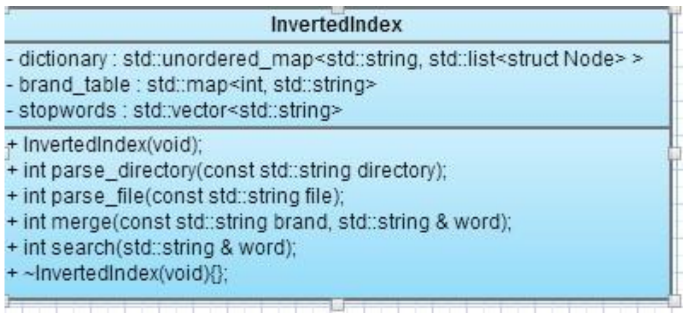
Section III: Code Structure (C++, Linux)

1. File Structure



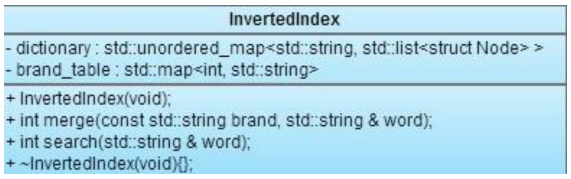
2. **Class Structure**

For this simple project, I generate an 'InvertedIndex' class.

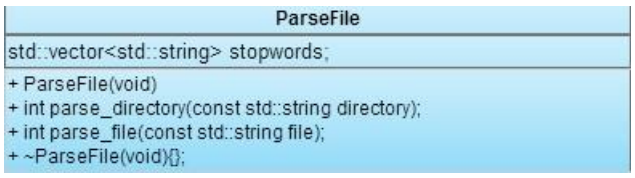


```
InvertedIndex
- dictionary : std::unordered_map<std::string, std::list<struct Node> >
- brand_table : std::map<int, std::string>
- stopwords : std::vector<std::string>
+ InvertedIndex(void);
+ int parse_directory(const std::string directory);
+ int parse_file(const std::string file);
+ int merge(const std::string brand, std::string & word);
+ int search(std::string & word);
+ ~InvertedIndex(void){};
```

For the intension of modularization in a large project, I would like to separate 'ParseFile' class from 'InvertedIndex' class.



```
InvertedIndex
- dictionary : std::unordered_map<std::string, std::list<struct Node> >
- brand_table : std::map<int, std::string>
+ InvertedIndex(void);
+ int merge(const std::string brand, std::string & word);
+ int search(std::string & word);
+ ~InvertedIndex(void){};
```



```
ParseFile
std::vector<std::string> stopwords;
+ ParseFile(void)
+ int parse_directory(const std::string directory);
+ int parse_file(const std::string file);
+ ~ParseFile(void){};
```

Thank you for reading my report. Welcome any suggestions for my algorithms and codes.

Yueying