American Journal of Software Engineering, *2017, Vol. 5, No. 1, 20-26*
Available online at http://pubs.sciepub.com/ajse/5/1/3
© Science and Education Publishing
DOI:10.12691/ajse-5-1-3

A Distributed Multi-facet Search Engine of Microblogs Based on SolrCloud

Lan Huang*, **Juan Zhou**

College of Computer Science, Yangtze University, Jingzhou, Hubei, China
*Corresponding author: lanhuang@yangtzeu.edu.cn

Abstract Microblog services, such as Twitter and Weibo in China, has become a new yet powerful information dissemination channel. More than 500 million tweets are sent every day. The extraordinary large number of messages brings new challenges to conventional search paradigms: a message might be relevant to the query in many aspects, for example the content, time and location of a message. Furthermore, there might be a large number of such relevant messages. In order to address these challenges, we designed a multi-facet distributed microblog search system using off-the-shelf open source frameworks including SolrCloud, Hadoop and Zookeeper. The system was then populated with real world messages collected from the most popular microblog website in China: Sina Weibo. We compared the performances of the standalone and the distributed version of the system. Empirical experimental results showed both effectiveness and efficiency of the proposed system in retrieving large scale microblog messages.

Keywords: *solr, SolrCloud, multi-facet retrieval, information retrieval, microblog*

Cite This Article: Lan Huang, and Juan Zhou, “A Distributed Multi-facet Search Engine of Microblogs Based on SolrCloud.” *American Journal of Software Engineering*, vol. 5, no. 1 (2017): 20-26. doi: 10.12691/ajse-5-1-3.

1. Introduction

Since Jack Dorsey, creator of Twitter, sent out the first tweet on March 21 2006, microblog has gradually grown into a major information dissemination channel. Nowadays, more than 500 million tweets are tweeted on Twitter every day [1], while the most popular microblog website in China, Sina Weibo, has around 300 million active users in late 2016 [2]. Millions of people broadcast their thoughts and opinions on a great variety of topics on microblogs. In recent years, it has been shown that microblogs can be a valuable resource for citizen-driven emergency responses [3], brand marketing and promoting [4], public opinion mining [5], rumor detection and tracking [6], and even for predicting trends in the stock markets [7].

However, the distinct features of microblogs as well as the sheer amount of them bring significant challenges to tools that aim to efficiently and effectively mine useful information from these messages. Microblog messages differ from conventional Web pages in various aspects [8]. Furthermore, people look for different things in microblogs and search in different ways [9]. To address these challenges, we proposed a multi-faceted microblog search system with distributed computing capabilities. The system provides exploratory search and browsing of microblogs. We employed off-the-shelf tools and platforms to build the system: the open source search engine framework SolrCloud and the Hadoop-related frameworks for distributed computing.

The rest of this paper is organized as following. Next we review related work on microblog retrieval. Section 3 introduces the core of the system: the search engine frameworks Solr and SolrCloud in detail. Section 4 describes the system design and implementation details. Section 5 presents experimental setup and discusses empirical experimental results. Section 6 concludes the study.

2. Related Work

Comparing to the conventional Web search engines, there is evidence showing that people search microblog messages for different types of information and in different ways. Teevan et al.’s study [9] on Twitter showed that people tend to search twitter to find dynamic information (e.g. temporally and spatially relevant information), whereas they search the Web for static information (e.g. basic facts and navigational content). Meanwhile, people tend to search microblog to monitor things, for example a particular topic, an event or a celebrity. A microblog search engine ought to incorporate such differences into its design and implementation.

Tapping into the rich content contained in microblogs is usually an obvious solution. Microblog messages differ from other types of information on the Web and on other social networking platforms, in that each message is very short (no more than 140 characters), its language is usually informal, yet it can contain diverse content features such as hashtagged topics, links and emojis, and associate to a bunch of social features such as followers, mentions, and retweets.

These different features present challenges as well as new opportunities to microblog search engines. Several previous studies have tried combining some of these features to form a better ranking algorithm for microblogs [8,10,11], usually with the help of machine learning. For example, Damak et al.'s study [8] found that tweet popularity, length, exact term matching, URL presence, URL popularity in the corpus, URL frequency in the microblog and the recency of the microblog were the best relevance indicators.

However, apart from coining these relevance indicators into a black-box ranking algorithm, explicitly providing them as options to the user might be a more flexible, user-friendly, and even more effective solution. So users can choose to explore microblogs that are relevant in the chosen aspect. In this direction, O'Conner et al. [12] developed TweetMotif, an exploratory search application for Twitter. Instead of simply listing matched messages, TweetMotif groups messages into topics and presents topics as facets. Quite a few subsequent studies have also exploited this multi-perspective nature of microblogs [13,14]. This motivates us to design and build a multi-facet search system for the Chinese microblogs.

3. Solr and SolrCloud

Solr, standing for Search On Lucene Replication, is an Apache open source project [15,16,17]. As its full name shows, it is a full-text search engine framework and is built upon the Lucene indexing libraries. Apart from the basic search functionality that comes with Lucene, Solr further expands it with a variety of enriched features such as advanced fulltext search and result highlighting, multi-facet indexing, matching and search result browsing, and index replication. Meanwhile, Solr also adds support for distributed computing.

Solr's architecture mainly consists of three layers, as shown in Figure 1. The outer layer is the user interface layer, which mainly provides HTTP accesses to the underlying Solr system. Major types of interactions include indexing new documents, executing queries and returning search results. Major formats of the exchanged data include XML and JSON.

The middle layer consists of Solr's core functionality components, such as document parsing, document indexing, index storage, and index updating. The parsing component performs most of the necessary document preprocesses. For example, for Chinese documents, word segmentation (i.e. segmenting consecutive Chinese characters into meaningful words with spaces) is usually the first preprocess conducted by the parsing component. Solr also provides flexible configuration of the preprocessing tools to be used in the system.

At the bottom is the search and index layer, who builds indexes and performs query-document matching, using Lucene's index and search modules. Besides, on the side of Figure 1 is Solr's independent index replication component, which mainly provides support for distributed data indexing and retrieval.

SolrCloud can be seen as a distributed version of Solr. By combining the distributed computing framework Hadoop and the distributed resource management framework ZooKeeper, SolrCloud can provide highly reliable and highly responsive search experiences. Figure 2 shows a typical SolrCloud structure.

SolrCloud usually consists of several Solr instances. Each Solr instance can have one or more Solr cores, where each core independently stores data and provides indexing and searching services. The multi-core mode is essential for ensuring the responsiveness of the retrieval services.

Shard is another distinct characteristic of SolrCloud. Shard can be considered as a logical organization unit of data and resources, as shown in Figure 2. Shards are disjunctive: there are no overlaps between different shards. The index collection is logically separated into different shards. As a result, data in each shard needs to be combined to form the complete index collection.

As for the structure within a shard, each shard usually connects to multiple Solr cores, with one being the leader node of the shard and the others being the replica. The leader node receives service requests from SolrCloud, and then dispatches the requests to all the replica nodes in the same shard. A replica node, as its name shows, is a replication of the leader node. When the leader node fails, SolrCloud automatically choose a new leader node from all the replica nodes in that shard, so as to ensure reliable services.

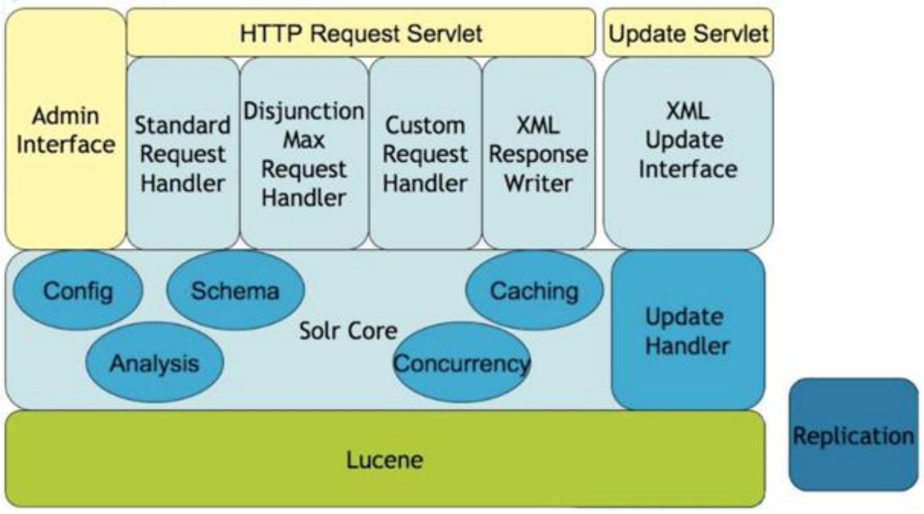


Figure 1. Architecture of Solr [18]

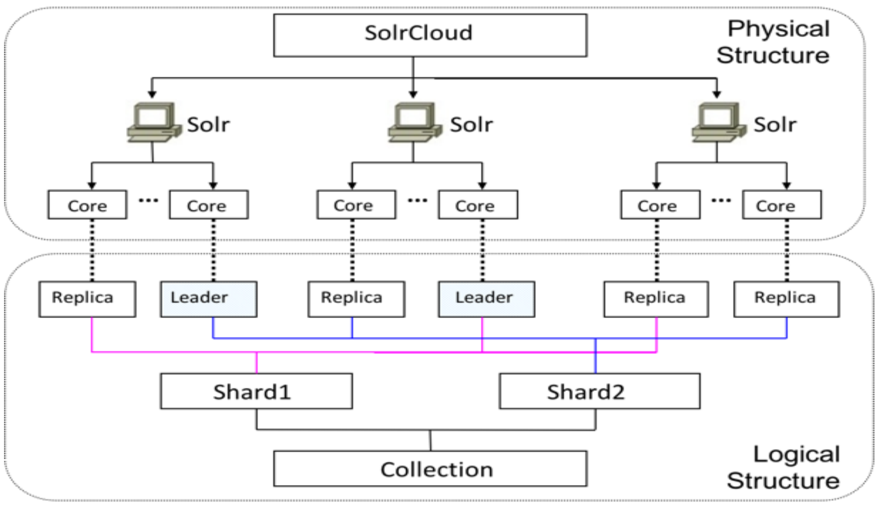


Figure 2. Architecture of SolrCloud

4. System Design and Implementation

Search engines usually consist of two parts: the backend crawlers that constantly collect and process information and the frontend retrieval components that provide search services. Following such a paradigm, we designed the frontend and the backend, and the components within each part. Figure 3 shows the overall architecture. The backend components mainly collect, filter, store and index microblog messages, whereas the frontend components mainly search, rank and present the retrieved messages. Furthermore, all the components showed in Figure 3 were built upon the Hadoop framework. The rest of this section describes each component in detail.

4.1. Data Collection and Preprocessing Module

We used the most popular microblog website in China—the Sina Weibo—in this study. Therefore, the first step of building our system was to collect microblog messages from this platform. We used Weibo platform's API [19] and collected public timeline messages for three months starting from January 1, 2016, resulting in over 2.1 million valid microblog messages.

Unfortunately, a considerable portion of the collected messages were spams, for example ads, usually with highly repetitive content. We then built a spam filter based on the multinomial Naïve Bayes classifier for text categorization [20]. To generate the training data for training the classifier, we firstly grouping high repetitive messages published in a randomly selected short time period (three days), and then manually labeled all distinct ones, resulting in around five thousand training examples. Messages were labeled as either negative instances (i.e. spam messages) or as positive instances (i.e. non-spam messages). Each instance is firstly segmented using the IK Chinese word segmenter [21] and then represented using the bag-of-words document representation. Words were weighted with TFIDF weighting scheme and stop words were removed.

When the classifier was trained with fine-tuned parameters, it is then applied to new incoming messages. Only messages that are classified as positive (i.e. non-spam) were indexed and stored.

4.2. Indexing Module

Indexing is the fundamental module of the system. The quality of the index structure directly impacts search effectiveness and efficiency. The indexing process works in two steps: locating and indexing. Locating means to find out the shard that the document's index belongs to. When a Solr server receives an indexing request, it will first check the type of the current Solr core. Indexing only happens in the leader node of a shard. Therefore, if the node is a replica, the document to be indexed will be transferred to the leader node in that shard. At the leader node, the document will be send to the shard according to the Hash value of its document ID. Then the leader node of that shard will generate the index entries of the input document.

The second step is updating and synchronizing indexes across different nodes. When a document is indexed, the leader node will update and synchronize the index data on the other replica nodes in the same shard. Therefore, the replica node can replace the leader node anytime in case the leader node fails. Scheduling and resource allocations in the above steps are performed by Zookeeper.

4.3. Search Module

Once a document is indexed, it can be searched through system's user interface. In the distributed search mode, every Solr server can provide search service in an independent fashion. As mentioned above, SolrCloud organizes index data by shards. Thus when a server receives a search request, SolrCloud will first disseminate the request to all the shards in the system. The leader node in each shard will perform search, i.e. document-query matching based on the index data stored in that shard. Results returned from each shard are then combined and ranked at the server that receives the request, and then returned to the frontend user.

The number of shards configured in a SolrCloud system thus clearly impacts system efficiency. Similar to the indexing module, all task scheduling and resource allocations are again performed by Zookeeper.

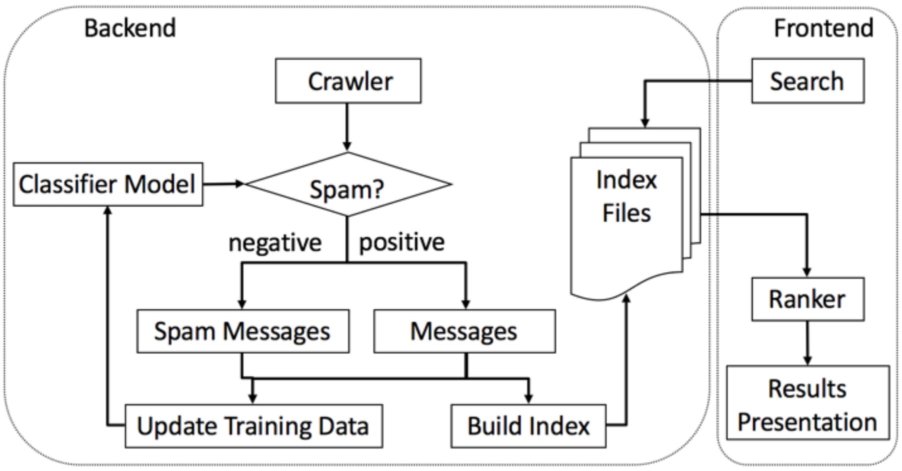


Figure 3. System Architecture

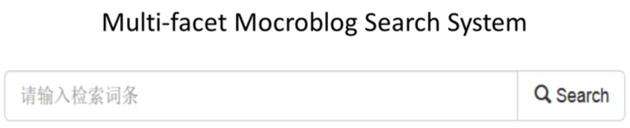
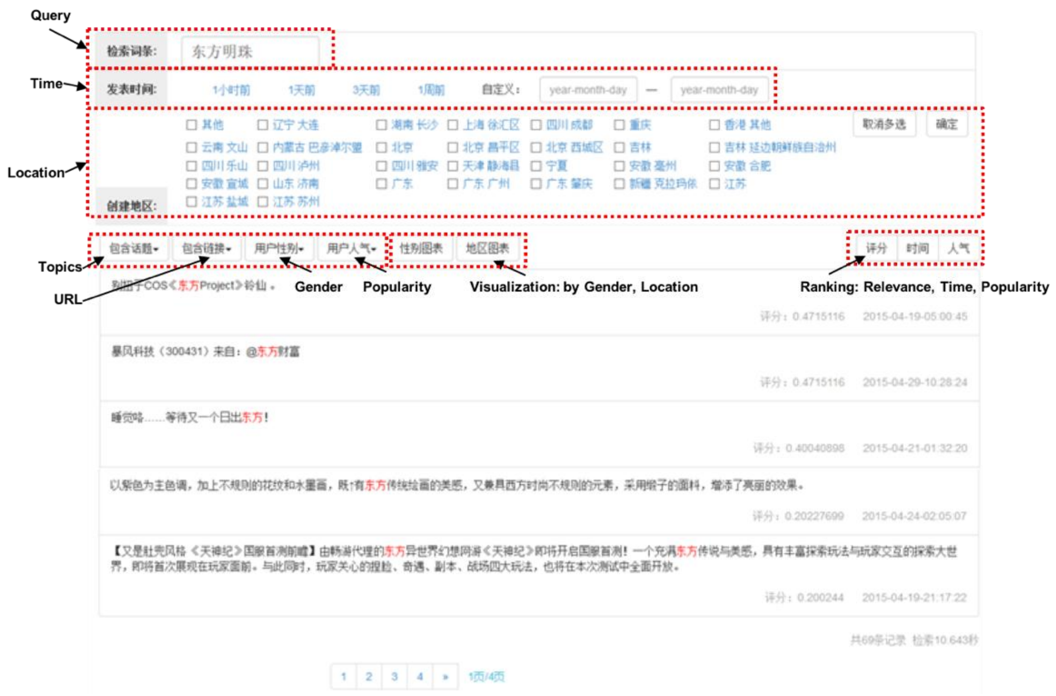


Figure 4. Search Interface



别扭于COS《东方Project》铃仙。

评分：0.4715116 2015-04-19-05:00:45

暴风科技（300431）来自：@东方财富

评分：0.4715116 2015-04-29-10:28:24

睡觉咯……等待又一个日出东方！

评分：0.40040898 2015-04-21-01:32:20

以紫色为主色调，加上不规则的花纹和水墨画，既有东方传统绘画的美感，又兼具西方时尚不规则的元素，采用缎子的面料，增添了亮丽的效果。

评分：0.20227699 2015-04-24-02:05:07

【又是杜亮风格《天神纪》国服首测前瞻】由畅游代理的东方异世界幻想网游《天神纪》即将开启国服首测！一个充满东方传说与美感，具有丰富探索玩法与玩家交互的探索大世界，即将首次展现在玩家面前。与此同时，玩家关心的捏脸、奇遇、副本、战场四大玩法，也将在本次测试中全面开放。

评分：0.200244 2015-04-19-21:17:22

共69条记录 检索10.643秒

Figure 5. Multi-faceted Search Result Presentation

4.4. User Interface

User interface (UI) is an essential component of a search system. We considered both effectiveness and user-friendliness while designing the system's UI. Figure 4 and Figure 5 show the search interface and the result page respectively. We used the BootStrap framework to design system UI.

Information facets refer to the multiple properties that an object might have. For example, the facets of a microblog message can contain its publishing time and location, content features such as whether it contains topics or hyperlinks, and properties of its sender such as gender and popularity. Multi-faceted browsing combines the different aspects of information in a unified manner, and is thus particularly effective for exploring diversified data objects such as microblogs.

Accordingly, our search system provides the above mentioned six facets: time, location, author's gender, author's popularity in terms of the number of followers the author has, containing topics and containing hyperlinks. Retrieved microblog messages can be ranked in three ways: relevance, publishing time and author's popularity. In comparison with previous studies, our system provides a rich set of options. Readers are referred to references [12,13,14] for similar systems in previous research.

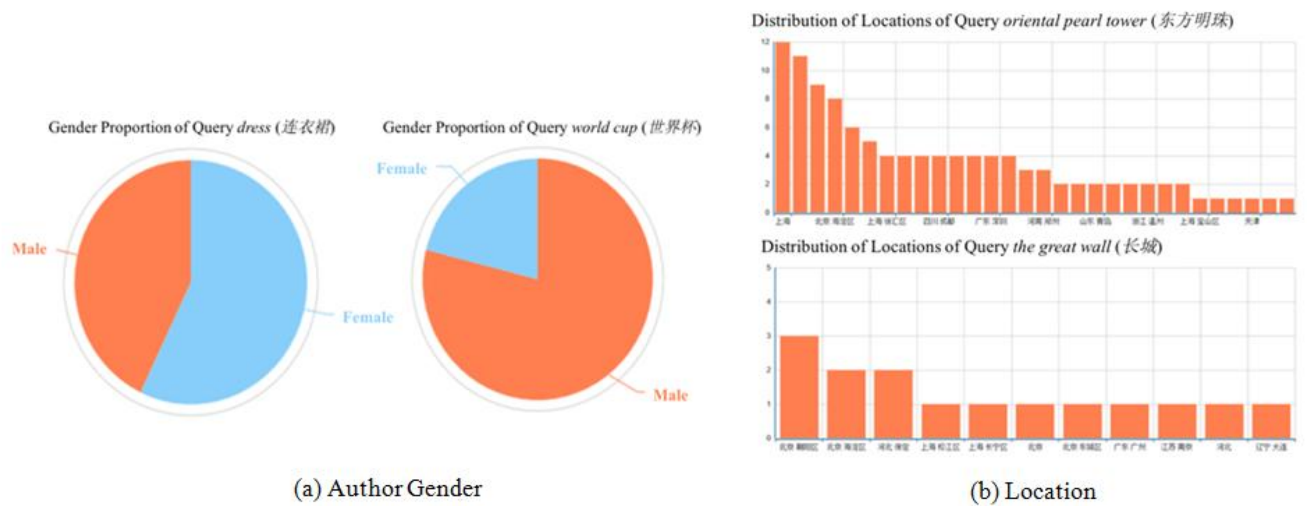


Figure 6. Visualization of Different Aspects of the Retrieved Messages

We also provide search results visualization to enhance user-friendliness. Figure 6 shows the visualization of author's gender and location of the retrieved messages. Compared to the plain ranked list of relevant messages in many mainstream microblog platforms, our system provides multiple angles in search, filtering, browsing and visualization, and thus allows users to explore the vast message collection in more effective ways. This is particularly helpful for applications such as opinion mining, marketing and personalized recommendation. For example, Figure 6(b) showed clear spatial clustering of messages relating to two landmark buildings of Shanghai and Beijing respectively. For the landmark of Shanghai (oriental pearl tower), three of the top ten locations came from Shanghai. Similarly, six of the top ten locations were either in Beijing or neighboring cities. This suggests that people tend to send location-aware messages, providing valuable information for mining places of interests.

5. System Evaluation and Discussion

5.1. Experimental Setup

We built a SolrCloud instance with three Linux servers in the same LAN. Table 1 showed the configuration of these servers. Moreover, each server was installed with Solr, ZooKeeper and Tomcat.

Table 1. Configuration of the SolrCloud Servers

ID	Role	Disk	OS	CPU	Memory
cloud001	master	20G	CentOS 6.5	Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz	1G
cloud002	slave	10G			
cloud003	slave	10G			

5.2. Indexing Efficiency

Indexing efficiency is usually measured in terms of throughput, i.e. the number of documents being indexed per second. This involves both creating and updating index. Many factors can impact indexing efficiency, such as hardware, bandwidth, document size and language. For example, for Chinese document, efficiency of the Chinese word segmenter directly impact the overall indexing speed.

We compared the standalone and the distributed versions of the system with the same experimental dataset. Figure 7 shows the indexing throughput in both cases. As Figure 7 shows, both time cost and indexing throughput increases as the number of documents grows. Distributed indexing demonstrated obvious advantage over standalone indexing, with a much smaller growth rate in time cost and a much greater growth rate in indexing throughput. Such advantage also expands as the number of documents grows, illustrating the strength of distributed computing in handling large datasets.

5.3. Retrieval Efficiency

The retrieval throughput, i.e. the number of relevant documents being retrieved per second, is usually the measure of retrieval efficiency. Apart from the matching and ranking algorithm, factors like bandwidth, complexity of the input query and format of the results can all impact retrieval efficiency. When examining system's retrieval efficiency, Figure 8 shows a similar trend to that in Figure 7. Figure 8(a) shows that distributed retrieval is constantly faster than standalone search, and this advantage grows as the number of documents grows. However, a closer look shows that such advantage of distributed computing is less obvious in the retrieval stage than in the indexing stage. SolrCloud's architecture of organizing index data by shards might be a possible reason. Because of this, the indexing operations, including both creating a new index entry and updating existing entries, are usually restricted to only one Solr core. In contrast, despite distributed computing is used or not, the search stage usually involves checking all index files. Although different shards are checked in parallel, results returned by each shard still needs to be combined to form the final result set. The combining step is performed on a single server, which means distributed computing does not reduce the time cost of this step. This also explains the similar performances in terms of system throughput of both systems on small datasets in Figure 8(b). When the dataset scales up, distributed system again obtained considerable advantage over the standalone system.

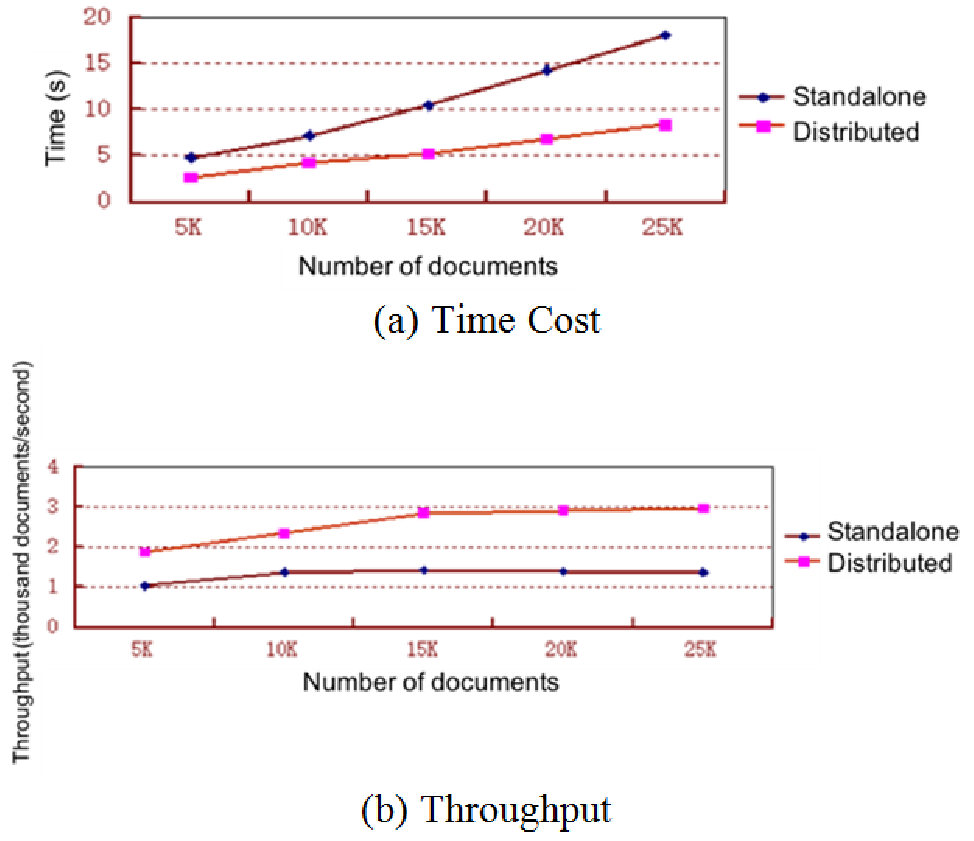


Figure 7. Comparison of (a) Time Cost and (b) Throughput Between Distributed Indexing and Standalone Indexing

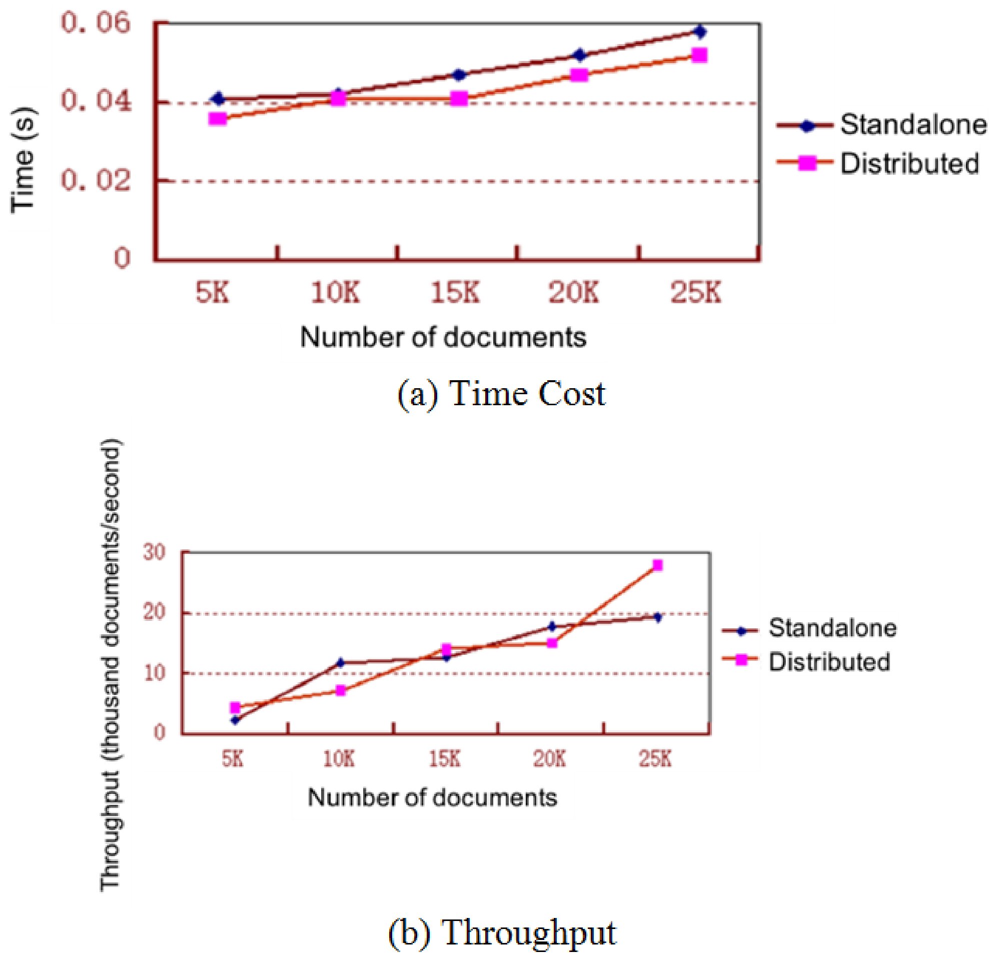


Figure 8. Comparison of Time Cost and Throughput Between Distributed Retrieval and Standalone Retrieval

6. Conclusions

Microblog platforms, such as Twitter and Sina Weibo, have become essential real-time information dissemination channels with an extraordinary large number of users and messages. They provide valuable information in a timely fashion and thus is helpful in many real world scenarios, including disaster response, brand marketing, opinion mining and rumor tracking. Yet, the large amount of microblog messages sent out every second brings pressing demand for effective and efficient retrieval tools. Accordingly, we designed and implemented a distributed multi-facet microblog retrieval prototype system. Mainstream frameworks were adopted, including Hadoop for distributed computing and Solr for information retrieval. Based on real word microblog messages from the most popular microblog website in China Sina Weibo, the experimental results showed the effectiveness of the proposed system. Users could explore relevant messages from different perspectives. Meanwhile, distributed computing successfully brought significant increase in system responsiveness as compared to the standalone system.

Acknowledgements

This study was funded by Yangtze University (grant no. JY2014032 and 2015cqn52).

References

[1] Twitter Usage Statistics. [Online]. Available: http://www.internetlivestats.com/twitter-statistics/#trend. [Accessed Aug. 20, 2017].

[2] 2016 Sina Weibo User Statistics Report. [Online]. Available: http://data.weibo.com/report/reportDetail?id=346. [Accessed Aug. 20, 2017].

[3] Qu, Y., Huang, C., Zhang, P., Zhang, J.. Microblogging after a Major Disaster in China: A Case Study of the 2010 Yushu Earthquake. In: ACM 2011 Conference on Computer Supported Cooperative Work, 2011, ACM, pp. 25-34.

[4] Sui, Y., Yang, X.. The Potential Marketing Power of Microblog. In: 2nd International Conference on Communication Systems, Networks and Applications (ICCSNA), 2010, pp. 164-167.

[5] O'Connor, B., Balasubramanyan, R., Routledge, B. R., Smith, N. A.. From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series. In: 4th International AAAI Conference on Weblogs and Social Media, 2010, ACL, pp. 122-129.

[6] Qazvinian, V., Rosengren, E., Radev, D. R., Mei, Q.. Rumor Has It: Identifying Misinformation in Microblogs. In: 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2011, pp. 1589-1599.

[7] Chen, C., Wu, D., Hou, C., Yuan, X.. Exploiting Social Media for Stock Market Prediction with Factorization Machine. In: 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014, IEEE, pp. 142-149.

[8] Damak, F., Pinel-Sauvagnat, K., Cabanac, G., Boughanem, M.. Effectiveness of State-of-the-art Features for Microblog Search. In: 28th Annual ACM Symposium on Applied Computing, 2013, ACM, pp. 914-919.

[9] Teevan, J., Ramage, D., Morris, M. R.. #TwitterSearch: A Comparison of Microblog Search and Web Search. In: 4th ACM International Conference on Web Search and Data Mining, 2011, ACM, pp. 35-44.

[10] Duan, Y., Jiang, L., Qin, T., Zhou, M., Shum, H. Y.. An Empirical Study on Learning to Rank of Tweets. In: 23rd International Conference on Computational Linguistics, AAAI, 2010, pp. 295-303.

[11] Massoudi, K., Tsagkias, M., de Rijke, M., Weerkamp, W.. Incorporating Query Expansion and Quality Indicators in Searching Microblog Posts. In: 33rd European Conference on Advances in Information Retrieval, 2011, Springer-Verlag, pp. 362-367.

[12] O'Connor, B., Krieger, M., Ahn, D.. TweetMotif: Exploratory Search and Topic Summarization for Twitter. In: 4th International AAAI Conference on Weblogs and Social Media, 2010, AAAI, pp. 384-385.

[13] Itokawa, S., Shiramatsu, S., Ozono, T., Shintani, T.. Estimating Feature Terms for Supporting Exploratory Browsing of Twitter Timelines. In: IIAI International Conference on Advanced Applied Informatics, 2013, IEEE, pp. 62-67.

[14] Zilincik, M., Navrat, P., Koskova, G.. Exploratory Search on Twitter Utilizing User Feedback and Multi-Perspective Microblog Analysis. PLoS ONE 8(11): e78857.

[15] Grainger, T., Potter, T.. *Solr in Action*. Manning Publications, New York, 2014.

[16] Apache Solr. [Online]. Available: https://lucene.apache.org/solr/. [Accessed Aug. 20, 2017].

[17] SolrCloud. [Online]. Available: https://lucene.apache.org/solr/guide/6_6/solrcloud.html. [Accessed Aug. 20, 2017].

[18] Goeschl, S.. Solr: An Open Source Enterprise Search. [Online]. Available: http://people.apache.org/~sgoeschl/presentations/solr/index.html. [Accessed Aug. 20, 2017].

[19] Weibo API. [Online]. Available: http://open.weibo.com/wiki. [Accessed Aug. 20, 2017].

[20] Kibriya, A. M., Frank, E., Pfahringer, B., Holmes, G.. Multinomial Naive Bayes for Text Categorization Revisited. In: 17th Australian Joint Conference on Artificial Intelligence, 2004, Springer, pp. 488-499.

[21] IK-analyzer. [Online]. Available: https://code.google.com/archive/p/ik-analyzer/downloads. [Accessed Aug. 20, 2017].