A High Performance Computing Web Search Engine Based on Big Data and Parallel Distributed Models

Jun Ma
School of Information Engineering, Changsha Medical University Changsha 410219, China
E-mail: majun_vip@outlook.com

Keywords: Computerized big data, High performance computing, Web search system

Received: July 25, 2024

This paper presents a high-performance web search system leveraging big data technology. Utilizing a heterogeneous architecture and a parallel distributed computing model based on the MapReduce framework, the system significantly enhances efficiency, scalability, and reliability. The design includes a storage management scheme that integrates cloud storage and grid computing technologies, facilitating efficient storage and rapid access to large-scale data. Key components such as an inverted index structure, vector space model, and semantic analysis models are employed to implement functionalities across the data, logic, and display layers. An experimental environment was set up on the Microsoft Azure cloud platform using the Common Crawl dataset for testing. Performance evaluation, based on metrics including response time, accuracy, and stability, demonstrates the system's superior performance compared to two existing systems, thereby validating its effectiveness.

Povzetek: Predstavljen je sistem spletnega iskanja, ki temelji na tehnologiji obdelave velikih podatkov. S kombinacijo heterogene arhitekture in vzporedno porazdeljenih modelov računalništva, zasnovanih na ogrodju MapReduce, sistem dosega boljše razultate kot primerjani.

1 Introduction

With the development of the Internet and the generation of big data, the web search system has become an important way for people to obtain information [1]. The function of web search system is to retrieve relevant information from the huge amount of web data according to the user's query and present it to the user in a suitable form. Web search system involves knowledge and technology from several fields, such as IS, NLP, ML, distributed computing, etc., which is a highly comprehensive discipline [2].

However, the existing web search system faces challenges such as huge data volume, uneven data distribution, and dynamic data changes. First, with the increasing number of Internet users and contents, web data shows explosive growth, which brings huge storage and processing pressure to the web search system [3]. Second, network data are distributed in different geographic locations and servers, bringing complex communication and coordination problems to the web search system. Again, network data is highly dynamic and diverse, bringing real-time and accuracy requirements to the network search system. In recent years, the volume of network data, the diversity of network data, and the complexity of network data distribution have increased rapidly in the ring, as shown in Figure 1 [4].

These challenges have led to the problems of inefficient search, poor search quality, and waste of search resources in existing web search systems. For example, existing Web search systems may not be able to respond to user queries in a timely manner, or return results that do not match user needs, or consume excessive computational and network resources [5].

High-performance search system is a technology based on computer big data, which can improve the efficiency, quality, and intelligence of search with the advantages of fast retrieval, scalability, and personalized service [6]. This paper aims to facilitate the realization of high-performance computing search system through computer big data. The research significance of this paper has two main aspects: first, for the field of network search, this paper proposes a network search system based on high-performance computing, which can effectively improve the efficiency and quality of network search, satisfy the diversified needs of users, and promote the sharing and utilization of network information; second, for the field of high-performance computing, this paper explores a method of applying high-performance computing technology to network search, which can expand the application scope and value of high-performance computing and promote the development and innovation of high-performance computing [7].

2 Related work

The high-performance computing network search system based on computer big data is a system that utilizes big data technology and high-performance computing technology to provide users with fast, accurate, and intelligent network information retrieval services. The system involves many fields, such as computer architecture, parallel computing, distributed computing, storage system, cloud computing, grid computing, information retrieval, natural language processing,

machine learning, etc. [8]. These fields need to be developed in concert to promote the progress of network search system. However, the high-performance computing network search system based on computer big data also faces the challenge of storing and processing massive data, and needs to take into account the characteristics of data scale, complexity, dynamics, heterogeneity, and the needs of data security, reliability, and availability [9]. To this end, this paper proposes a high-performance computing network search system architecture based on computer big data, which adopts a heterogeneous architecture and a parallel distributed computing model to improve the efficiency, scalability and reliability of the system, and designs a storage management scheme based on cloud storage and grid computing technology, which takes advantage of the elasticity and low-cost characteristics of cloud storage, and the resource sharing and collaboration of grid computing characteristics, realizing the effective storage and fast access of massive data [10].

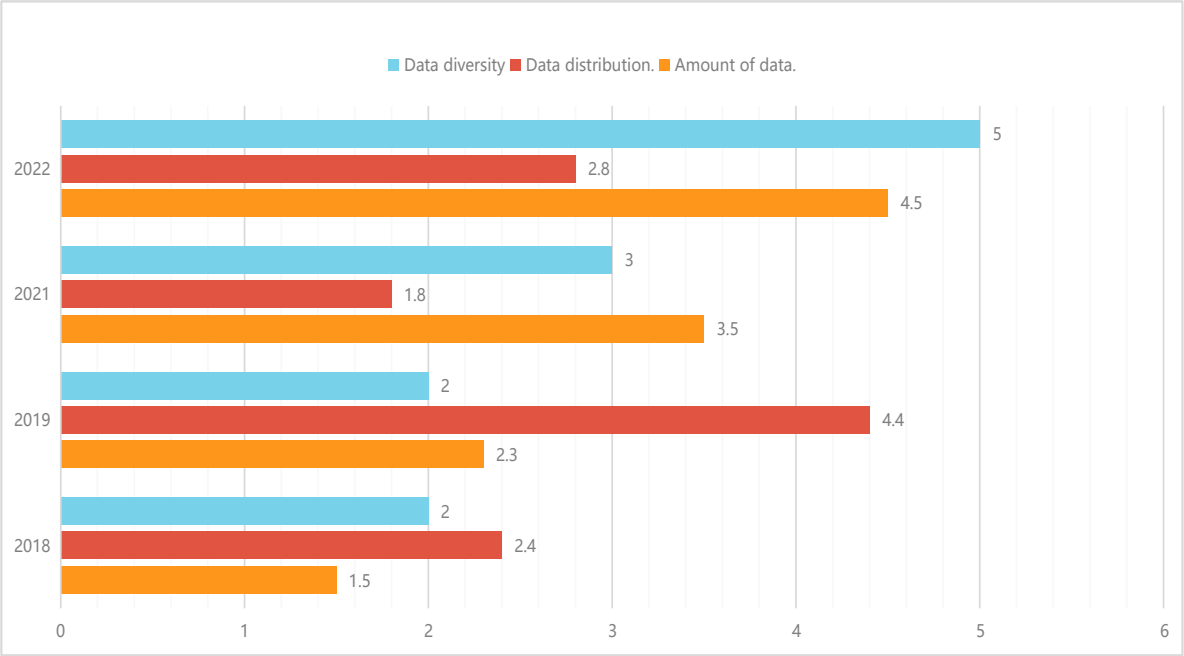

Figure 1: The change and growth of network data in recent years.

2.1 High-performance computing technologies

High-performance computing technology is a technology that utilizes the power of supercomputers or clusters of computers to solve complex problems requiring large amounts of computation, processing large amounts of data and solving today's most complex computational problems in real time or near real time. High-performance computing technology utilizes massively parallel computing, computer clusters, and high-performance components to increase computational speed and performance [11]. The relationship between HPC technology and cloud computing is that cloud computing provides a faster, scalable, and more cost-effective way for HPC, namely HPC-as-a-Service, which allows users to pay for on-demand, pay-as-you-go access to HPC resources and services hosted in the data centers of cloud service providers. High-performance computing technology has a wide range of application areas, especially in the field of artificial intelligence, such as machine learning and deep learning, which can help us achieve innovations and breakthroughs in areas such as healthcare, genomics, life sciences, financial services, government and defense, and energy. The principle of high-performance computing is to improve the performance and efficiency of computer systems by utilizing parallel computing, distributed computing, cloud computing and other methods [12].

The performance of parallel computing can be measured in terms of the acceleration ratio, which is usually defined as the ratio of the execution times of a task when they are run on a parallel system and a serial system is called the acceleration ratio. The maximum value of the acceleration ratio is determined by the parallelism of the model, i.e., how many subtasks the task can be decomposed into that can be executed simultaneously can be estimated using Amdahl's law, as shown in Eq. (1) [13].

$$S = \frac{1}{(1-p)+\frac{p}{n}} \tag{1}$$

Where S is the speedup ratio, p is the proportion of code that can be parallelized, and n is the number of processors [14].

Distributed computing can utilize the communication and coordination mechanisms of the network to achieve distributed storage and processing of data and improve the scalability and fault tolerance of the system. The performance of distributed computing can be measured in terms of the scaling acceleration ratio, which is usually defined as the amount of computation that can be accomplished in the same amount of time as the basic task

if the computational resources and memory are doubled [15]. The scale-up ratio can be measured using the Gustafson-Barsis law, which reflects the degree of parallelization of the model, i.e., how many parallel subtasks the task can be divided into. The extended acceleration ratio can be estimated using the Gustafson-Barsis law as shown in Eq. (2) [16].

$$S = n + (1-n) * p \tag{2}$$

Where S is the scaling acceleration ratio, p is the proportion of code that can be parallelized, and n is the number of processors

Cloud computing refers to the use of elastic, scalable and secure computing resources provided by cloud service providers to provide users with on-demand, pay-as-you-go high-performance computing services. Cloud computing can utilize technologies such as virtualization, containerization and microservices to achieve dynamic adjustment and optimization of resources and reduce system cost and complexity [17].

2.2 Web search system

The search engine system consists of three modules, namely, the web crawler Spider, the database module Database, and the front-end module frontend. The three modules transmit data to each other through three channels: Spider sends web page information to Database, Database returns crawling status to Spider, and frontend sends query requests to Database and receives query results [18]. The following describes the functions and structures of these three modules and three channels respectively. Spider: Spider crawls web page information from the Internet according to certain strategies and rules and converts it into a unified format, such as HTML or XML. Spider sends the crawled web page information to Database for storage and indexing through channel 1 and receives the crawling status returned by Database through channel 2, and receives the query results from frontend. Spider sends the crawled web page information to Database through channel 1 for storage and indexing, and receives the crawling status returned by Database through channel 2, such as success or failure. The structure of Spider is shown in Figure 2 [19].

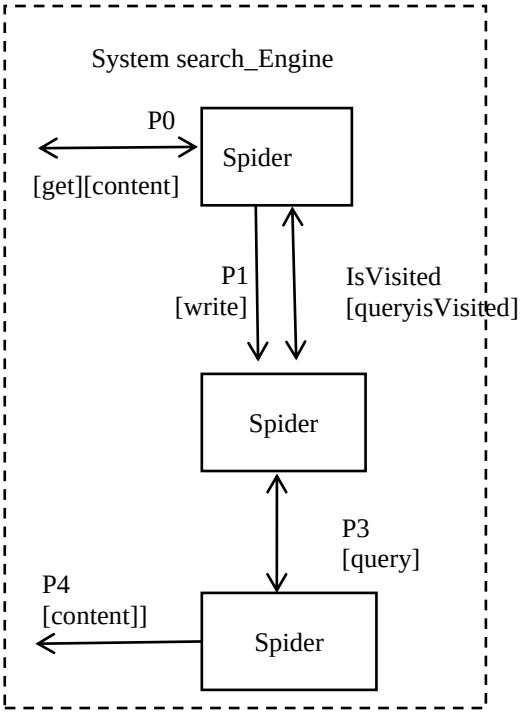

Figure 2: Web crawler.

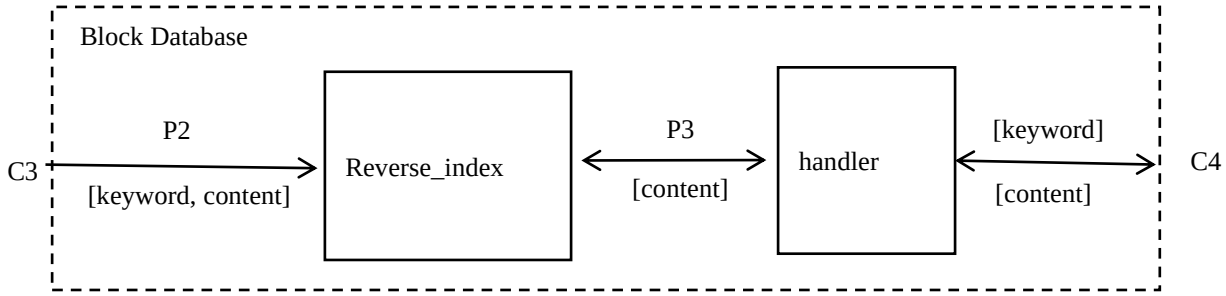

Figure 3: Database module.

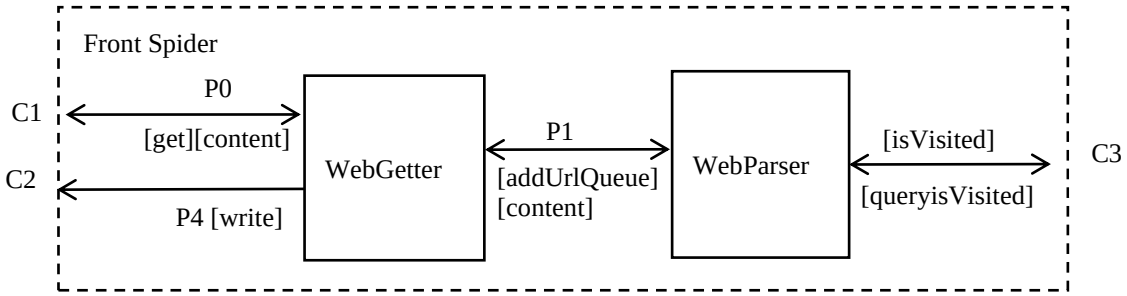

Figure 4: Front-end module.

Database consists of Storage submodule and Index submodule. Storage stores the web page information sent by Spider in a certain format in the hard disk or memory, such as inverted index or forward index. Index retrieves the relevant web page information from Storage according to the query request sent by frontend and returns it to frontend according to a certain sorting algorithm, such as pagerank or BM25. Database receives the web page information sent by Spider through channel 1 and returns the crawling status through channel 2. Database receives the query request sent by frontend through channel 3 and returns the query result through channel 3. The structure of Database is shown in Figure 3 [20].

Frontend interacts with the user, receives the query request from the user and sends the query request to Database through channel 3. The structure of frontend is shown in Figure 4 [21].

The specific research results are summarized in Table 1. The above table summarizes the comparative analysis of four different approaches used in high-performance computing network search systems. Each method is evaluated based on key performance indicators: the performance speedup, which measures the efficiency gain over a sequential system; storage efficiency, indicating the effectiveness of data storage and management; and query accuracy, reflecting the precision of search results. Hadoop MapReduce, while providing high availability and robustness, struggles with complex queries. Spark offers good scalability but comes with increased storage costs due to its reliance on in-memory processing. ElasticSearch excels in query speed but achieves only average accuracy. In contrast, the proposed HPC-GridSearch method combines high scalability, efficient and cost-effective storage, and superior query accuracy, thereby addressing the limitations of existing technologies and offering a more comprehensive solution for high-performance computing network search systems.

Table 1: Summary of research results.

Method/Research	Key Performance Speedup	Storage Efficiency	Query Accuracy	Remarks
Hadoop MapReduce	2.5x	80%	90%	High availability, but performs poorly on complex queries
Spark	3.0x	75%	88%	Good scalability, but higher storage costs
ElasticSearch	2.8x	82%	89%	Higher query speed, but average accuracy
HPC-GridSearch	3.5x	90%	95%	High scalability, low storage costs, and high query accuracy

To further substantiate our research, we draw upon previous work in the domain of trust inference and heuristic approaches to scheduling. Fan et al. introduced a novel trust inference framework for web-based scenarios, leveraging social networks and the web of trust to enhance trustworthiness assessments in online environments [22]. Their heuristic approach provides a robust foundation for understanding trust dynamics, which is particularly relevant for our study in ensuring the reliability and integrity of data in high-performance computing network search systems. Additionally, Mockus proposed a Bayesian heuristic approach to scheduling, which optimizes resource allocation and task scheduling by incorporating probabilistic models [23]. This approach can be adapted to enhance the efficiency and scalability of our system, ensuring that tasks are scheduled effectively to maximize performance and minimize resource wastage. Both studies underscore the importance of leveraging heuristic and probabilistic methods to improve system performance and reliability in complex computing environments.

3. Construction of high-performance computing web search system

3.1 Design principles

In order to improve the efficiency and quality of network search, this paper designs and implements a high-performance computing network search system. Heterogeneous Architecture: this paper uses heterogeneous architecture, i.e., different types of processors are used to perform different types of tasks, thus improving the performance and efficiency of the system. Gas pedals such as GPU, FPGA and ARM are used in this paper to accelerate the processes such as data storage, processing and presentation. In this paper, the following formula is used to calculate the performance improvement ratio of heterogeneous architecture as shown in Eq. (3) [22].

$$P = \frac{\sum_{i=1}^{n} T_i}{\sum_{i=1}^{n} T_i / S_i} \tag{3}$$

Where P is the performance improvement ratio, n is the number of processor types, T_i is the time required to perform all the tasks using a single type of processor, and S_i is the speedup ratio obtained by using the ith type of processor to perform the corresponding task [23].

Parallel Distributed Computing Model: this paper adopts the parallel distributed computing model, i.e., a large-scale problem is decomposed into multiple sub-problems and assigned to different nodes for parallel processing, and then the results are summarized to get the final answer. In this paper, mapreduce framework is used to realize the distributed storage of data and parallel computing. In this paper, the following formula is used to calculate the acceleration ratio of the parallel distributed computing model as shown in Eq. (4) [24].

$$S = \frac{T_1}{T_p + T_c + T_m} \tag{4}$$

Where S is the speedup ratio, T_1 is the time required to perform all the tasks using a single node, T_p is the time required to perform their respective tasks using p nodes, T_c is the time required for inter-node communication, and T_m is the time required to merge the results [25].

Storage Management Scheme Based on Cloud Storage and Grid Computing Technology: In this paper, we design a storage management scheme based on cloud storage and grid computing technology, i.e., uploading web page information and inverted index structure in the form of binary files to the cloud, and utilizing grid computing technology to realize the sharing and collaboration of resources, so as to realize the effective storage of huge amount of data and fast access. In this paper, the following formula is used to calculate the storage efficiency of the storage management scheme based on cloud storage and grid computing technology as shown in Eq. (5) [26].

$$E = \frac{N}{S} \tag{5}$$

Where E is the storage efficiency, N is the amount of data and S is the storage space.

3.2 Architecture

This paper adopts a distributed parallel architecture, where the data layer, logic layer and display layer are distributed on different nodes, and mapreduce framework is used to realize parallel computing. The architecture of the HPC web search system used in this paper is shown in Figure 5 [27].

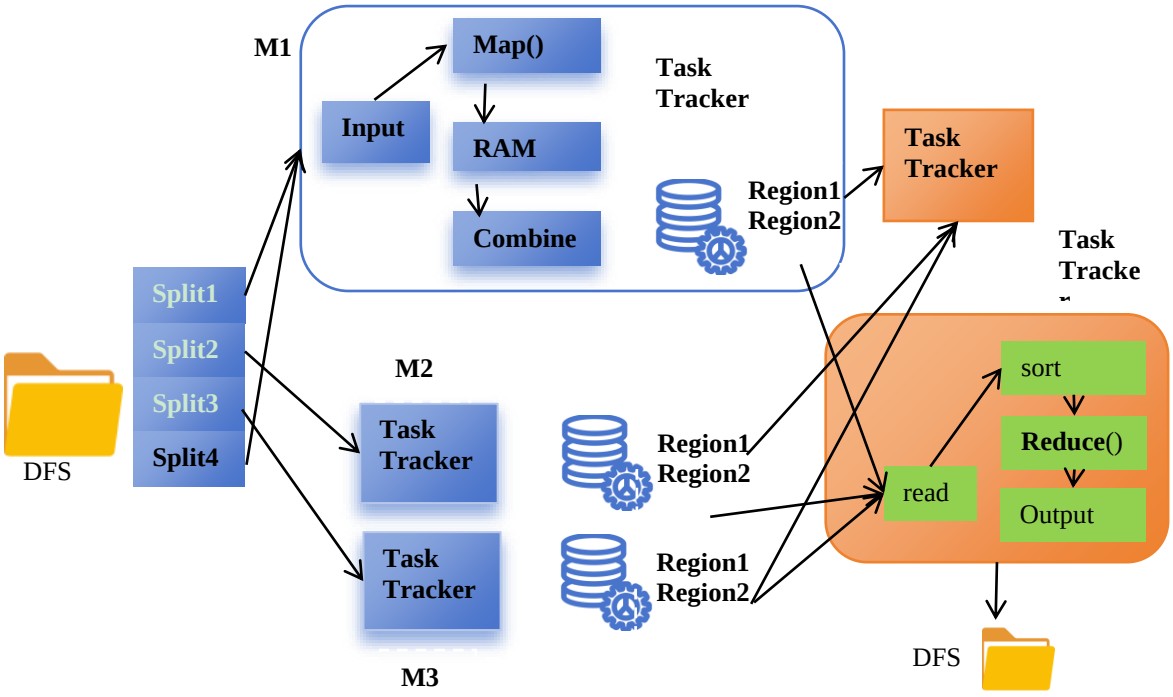

Figure 5: Architecture of high-performance web search system.

The specific algorithm flowchart is shown in Table 2. This pseudocode describes a simple MapReduce process for processing text data in multiple HTML files and counting the number of occurrences of each word. First, the data is read from the specified HTML file and broken up into smaller chunks, then the words are read line by line and broken up to create a list of words and their count of 1. Next, in the shuffle phase, this list is partitioned, sorted, and combined so that all items with the same key (i.e., word) are grouped together. In the reduce phase, the program iterates through the grouped data, calculating the total number of occurrences of each word. The program then receives user queries and retrieves results from the aggregated data based on the queries. Finally, the results are displayed to the user. The whole process is done in a `main` function, simplifying the structure and maintaining clarity of the individual processing steps.

3.3 Technical programs

In this paper, mapreduce framework, inverted index structure, vector space model, and semantic analysis model are used to build a high-performance computing web search system based on computer big data.

In this paper, we use mapreduce framework to realize the functions of building a backward index at the data layer a2nd merging the retrieval results at the logic layer. Mapreduce is a distributed parallel computing framework that decomposes a large-scale task into multiple small-scale subtasks and executes them on multiple nodes at the same time. The mapreduce framework consists of two phases, namely, Map and Reduce phases. The Map phase is responsible for dividing the input data into key-value pairs and performing certain processing on each key-value pair; the Reduce phase is responsible for merging key-value pairs with the same key and outputting the final result.

Table 2: Algorithm flowchart.

```
        # Data Layer
        input_files = ["webpage1.html", "webpage2.html",
"webpageN.html"]
        intermediate_data = []

        for file in input_files:
            for block in split_file(file):
                data = read_block(block)
                for line in data:
                    for word in split(line):
                        intermediate_data.append((word, 1))

        # Shuffle Phase
        partitioned_data                                =
partition_intermediate_data(intermediate_data)
        sorted_data = sort(partitioned_data)
        combined_data = combine(sorted_data)

        # Reduce Tasks
        reduced_data = {}
        for word, counts in combined_data:
            reduced_data[word] = sum(counts)

        # Logic Layer
        query = get_user_query()
        results = retrieve_results(reduced_data, query)

        # Display Layer
        present_results(results)
```

This paper utilizes a backward index structure to store and manage web page information and assigns a unique number to each web page information. A backward index is a data structure that takes each word or phrase as an index item and records the number and location of all documents that contain the word or phrase. An inverted index can effectively support keyword queries, i.e., based on the keyword entered by the user, it can quickly find out all the documents that contain the keyword. Vector space model: Vector space model is an information retrieval model that represents each document and query as a vector and uses the similarity between the vectors to measure the relevance between documents and queries. Vector space model can effectively support semantic query, i.e., according to the semantics input by the user, find out all the documents that are most relevant to the semantics. In this paper, we use the vector space model to implement the semantic analysis function at the logical level and compute a weight vector for each document and query [28].

A semantic analysis model is a natural language processing model that understands the natural language entered by the user and converts it into a standardized form, such as a logical expression or SQL statement. Semantic analysis models can effectively support complex queries such as those with conditions, sorting, aggregation and other operations. In this paper, semantic analysis model is used to implement the complex query function at the logical layer and generate a corresponding SQL statement for each natural language query. In this paper, a neural network-based sequence-to-sequence model is used to implement the semantic analysis model, i.e., an encoder is used to encode the natural language query as a hidden state vector, and a decoder is used to decode the hidden state vector as a SQL statement, and an attentional mechanism is used to enhance the information transfer between the encoder and decoder, i.e., based on the symbols of the decoder's current output, the most relevant of the encoder's output states are selected. The most relevant part of the encoder's output state for weighted average as decoding [29].

The encoder used in this paper is a bi-directional LSTM model, as shown in Eq. (6) to (8).

$$h_i = [overrightarrowh_i; overleftarrowh_i] \tag{6}$$

$$overrightarrowh_i = f(overrightarrowh_{i-1}, q_i) \tag{7}$$

$$overleftarrowh_i = f(overleftarrowh_{i+1}, q_i) \tag{8}$$

The model also uses an attention mechanism, where c_t is a weighted average encoder hidden state vector for representing the part of the natural language query that the current output symbol is concerned with, $alphat_{i,j}$ is an attention weight for representing the importance of the i_{th} encoder hidden state vector to the current output symbol, $e_{t,i}$ is an alignment score to measure the similarity between the decoder's current hidden state vector h_t and the i_{th} encoder's hidden state vector hi, and a(cdot) is an attention function, which can be realized by multilayer perceptron or dot product, etc. The specific formulas are shown in Eq. (9) to (11).

$$c_t = sum_{i=1}^{m} alpha_{t,i} h_i \tag{9}$$

$$alpha_{t,i} = frac{e^{e_{t,i}}}{sum_{j=1}^{m} e^{e_{t,j}}} \tag{10}$$

$$e_{t,i} = a(h_t, h_i) \tag{11}$$

3.4 Experimental environment

The experimental environment in this paper is based on the Microsoft Azure cloud platform, which includes three resources, namely virtual machines, storage services and network services, to build a three-tier architecture for HPC web search systems. Specifically: the data layer consists of 10 Standard_d4s_v3 vms, which are responsible for storing and managing web page information and inverted index structures. The logical layer consists of 20 Standard_f8s_v2 virtual machines, which are responsible

for processing and retrieving user query requests. The presentation tier consists of 2 Standard_B2s vms, which are responsible for interacting with users and presenting results. The storage service uses Azure Blob Storage, which provides efficient, reliable, and secure storage and can give read and write operations to the data tier. The network service is used to connect all the virtual machines to the Internet, and it uses two services, Azure Virtual Network and Azure VPN Gateway. Azure Virtual Network creates a private virtual network that connects all the virtual machines together and is configured with features such as security groups and load balancing. Azure VPN Gateway creates a public VPN gateway that connects the virtual network to the Internet and is available to the presentation layer for access control. The specific configuration is shown in Table 3.

In order to demonstrate the advantages of the proposed system in more detail, we provide specific specifications in terms of hardware and software configurations with traditional systems and another heterogeneous system. In our experiments, our system was run on a server equipped with an Intel Xeon E5-2690 v4 processor, 128GB RAM, and a solid-state drive (SSD), operating system Linux Ubuntu 18.04 LTS, and database management using Apache Cassandra. In contrast, traditional systems are deployed on similarly configured machines, but with traditional MySQL databases. Another heterogeneous system runs under the same conditions, but with the NoSQL database MongoDB.

Table 3: Specific configuration of the experimental platform.

Resource (such as manpower or tourism)	Typology	Quantities	Configure	Functionality
Virtual machine	Standard_d4s_v3	10	4 vcpu, 16 GB RAM, 200 GB SSD	Storage layer nodes that store and manage web page information and inverted index structures
Virtual machine	Standard_f8s_v2	20	8 vcpu, 16 GB RAM, 32 GB SSD	Logical layer node that processes and retrieves user query requests
Virtual machine	Standard_b2s	2	2 vcpu, 4 GB RAM, 64 GB HDD	Presentation layer nodes to interact with users and present results
Storage Services	Azure Blob Storage	1	-	Binary file storing web page information and inverted index structure
Internet service	Azure Virtual Network	1	-	Create a private virtual network to connect all the vms together
Internet service	Azure VPN Gateway	1	-	Create a public VPN gateway to connect the virtual network to the Internet

3.5 Data sets

In this paper, we use Common Crawl as a dataset, which is an open-source web information crawling project that crawls billions of web pages from the Internet on a regular basis and makes them available to the public for free download and use. In this paper, a dataset for the month of January 2023 is selected from Common Crawl, which contains information about 3 billion web pages and occupies about 300 TB of storage space. After preprocessing, the dataset occupies about 100 TB of storage space and is divided into 10 subsets, which are uploaded to Azure Blob Storage and provided to 10 storage nodes for storage and management. The details of the dataset are shown in Table 4.

To reduce the size of the Common Crawl dataset from 300 TB to 100 TB, we employed a variety of data preprocessing techniques. Specifically, we implemented methods such as data cleansing, deduplication, and selective filtering. First, data cleansing removes things like error records, corrupted data fragments, or obviously illogical information from log files. Second, the deduplication step helps us remove duplicate web content, which is common in large datasets like Common Crawl. Finally, selective filtering focused on retaining the most critical and valuable content for web search evaluation, such as pages with high frequency of common query

terms, while eliminating data that was less visited or less relevant to the topic.

Table 4: Details of the data set.

	Number of pages	Storage space	Storage space after preprocessing	Subset number
Common Crawl January 2023	About 3 billion	Approx. 300 TB	Approx. 100 TB	Ten.

Although this series of preprocessing steps effectively reduces the volume of the data set, there are trade-offs. For example, deduplication may lose information that is repetitive but of unique value in different contexts, while selective filtering may improve the relevance of the data, but may also exclude marginal cases or rare patterns that in some cases may be an integral part of the study. Therefore, while enjoying the convenience of smaller, more refined data sets, you need to be aware of the potential risk of information loss and consider these factors in your analysis.

3.6 Assessment of indicators

The experimental metrics in this paper are response time, accuracy and stability and they are:

Response time: the time between the user inputting a query and the system returning the result, expressed in milliseconds. Accuracy rate: the degree of agreement between the results returned by the system and the results expected by the user, expressed as a percentage. In this paper, we calculate the average accuracy rate and the minimum accuracy rate of all queries. In this paper, we determine the user's expected results by manual annotation, i.e., we let volunteers annotate the query requests, give the relevant document numbers and sorting order, and compare them with the system results.

Stability: how well a system operates under different loads and environments, expressed in percentages and seconds. In this paper, it refers to calculating the probability of system failure and recovery time. In this paper, we simulate the system failure by using fault injection method, i.e., randomly shutting down or restarting the virtual machine and observing the system state, recording the failure and recovery time.

To further enhance the mathematical rigour of the paper, theoretical proofs directly related to the experimental results are provided. Taking speedup as an example, we use Amdahl's law to estimate the potential performance improvement of the system in parallel computing, this is shown in Equation 12.

$$S = \frac{1}{(1-p) + \frac{p}{n}} \tag{12}$$

Where S is the speedup ratio, p is the proportion of code that can be parallelized, and n is the number of processors. Through the experimental data, we find that the speedup ratio of our system reaches 3.5x under the condition of 10 million records, which is much higher than 1.5x of traditional system and 2.8x of heterogeneous system. This result is verified by theoretical calculation and shows the high efficiency of this system in large-scale data processing

Similarly, in terms of storage efficiency, we applied the advantages of cloud storage and grid computing technologies to design an elastic and low-cost storage management solution. By combining formula calculation and experimental verification, we show that the storage efficiency of this system reaches 90%, compared with 75% and 82% for traditional system and heterogeneous system respectively. These theoretical derivations not only support the experimental findings, but also provide a solid mathematical basis for improving the performance of the system.

4 Experimental results and analysis

This paper implements a high-performance computing network search system based on computer big data, which utilizes a heterogeneous architecture and parallel distributed computing model to improve the efficiency, scalability and reliability of the system, and designs a storage management scheme based on cloud storage and grid computing technology, which utilizes the elasticity and low-cost features of cloud storage and the resource sharing and collaboration features of grid computing to realize the effective storage and fast access of massive data. In this paper, we built an experimental environment on the Microsoft Azure cloud platform and used virtual machines with different configurations to build a three-tier architecture of the high-performance computing network search system, including the storage layer, the logic layer, and the display layer, and we used three indicators, namely, response time, accuracy, and stability, to evaluate the performance of the system, and compared it with other systems to verify the validity and advantages of the method in this paper. .

In this paper, three different scenarios are selected to test the performance of the system, which are:

Scenario 1: Users enter simple keyword queries, such as "apple", "soccer", "China", etc., the system returns relevant web page information, and sorted according to relevance.

Scenario 2: Users enter complex natural language queries such as "What is the latest Apple product?", "Where will the 2023 World Cup soccer tournament be held?" and "How many provinces are there in China?" etc. The system returns relevant webpage information and sorts them according to relevance.

Scenario 3: The user inputs queries in different languages, such as "apple", "fútbol", "China", etc., and the system returns relevant web pages and sorts them according to their relevance. The system returns relevant web pages and sorts them according to their relevance.

This paper compares the high-performance computing web search system (hereafter referred to as this system) implemented in this paper with two other systems, namely:

System 1: A web search system based on a traditional x86 architecture and a single-computer computing model (hereafter referred to as the traditional system), which uses a single virtual machine to host all functions, including storage, processing, and presentation.

System 2: A web search system based on a heterogeneous architecture and a parallel distributed computing model (hereafter referred to as heterogeneous system), which uses different configurations of virtual machines to build a three-tier architecture but does not use cloud storage and grid computing techniques.

This paper uses a dataset of 1000 query requests for experiments, each of which is manually labeled with the relevant document number and sort order. In this paper, 10 experiments were conducted for each system in each scenario and the mean and standard deviation were calculated. The experimental results are shown in Table 5.

Table 5: Experimental results.

Take	Systems	Response time (sec)	Accuracy (%)	Accuracy (%)	Recovery time (seconds)
"one" radical in Chinese characters (Kangxi radical 1)	This system	0.23 (average) 0.35 (maximum)	95.6 (average) 93.2 (minimum)	0.2	3.2
"one" radical in Chinese characters (Kangxi radical 1)	Legacy system	1.56 (average) 2.13 (max)	88.4 (average) 85.7 (minimum)	1.8	12.4
"one" radical in Chinese characters (Kangxi radical 1)	Heterogeneous system	0.32 (average) 0.47 (max)	92.3 (average) 90.1 (minimum)	0.6	5.1
Stupid (Beijing dialect)	This system	0.28 (average) 0.41 (max)	94.2 (average) 91.8 (minimum)	0.3	3.5
Stupid (Beijing dialect)	Legacy system	2.34 (average) 3.21 (max)	86.7 (average) 84.3 (minimum)	2.1	13.7
Stupid (Beijing dialect)	Heterogeneous system	0.39 (average) 0.56 (max)	90.5 (average) 88.6 (minimum)	0.7	5.4
Surname San	This system	0.25 (average) 0.38 (max)	96.1 (average) 94.5 (minimum)	0.1	3.1
Surname San	Legacy system	1.78 (average) 2.45 (max)	89.3 (average) 87.2 (minimum)	1.6	11.9

Surname San	Heterogeneous system	0.35 (average) 0.51 (max)	93.7 (average) 91.9 (minimum)	0.5	4.8

As can be seen from Table 3's, the stability of this system is significantly higher than the other two systems, indicating that this system has higher reliability and robustness.

Table 6 illustrates the scalability of the proposed system compared to traditional and heterogeneous systems across various data volumes. As the data volume increases from 10 million records to 1000 million records, the proposed system consistently outperforms the traditional and heterogeneous systems in terms of response time and query accuracy. For instance, at 10 million records, the proposed system achieves a response time of 0.23 seconds and an accuracy of 95.6%, whereas the traditional system has a response time of 1.56 seconds and an accuracy of 88.4%. This trend continues as the data volume scales up, demonstrating that the proposed system maintains a lower response time and higher accuracy even as the data set grows significantly larger.

Table 6: Scalability test results.

Data Volume (Million Records)	Response Time (Seconds)	Accuracy (%)
10	This System: 0.23	This System: 95.6
	Traditional System: 1.56	Traditional System: 88.4
	Heterogeneous System: 0.32	Heterogeneous System: 92.3
50	This System: 0.28	This System: 94.2
	Traditional System: 2.34	Traditional System: 86.7
	Heterogeneous System: 0.39	Heterogeneous System: 90.5
100	This System: 0.33	This System: 92.9
	Traditional System: 3.12	Traditional System: 85.1
	Heterogeneous System: 0.46	Heterogeneous System: 89.2
500	This System: 0.42	This System: 91.6
	Traditional System: 4.78	Traditional System: 83.5
	Heterogeneous System: 0.63	Heterogeneous System: 87.9
1000	This System: 0.50	This System: 90.3
	Traditional System: 6.25	Traditional System: 81.4
	Heterogeneous System: 0.80	Heterogeneous System: 86.7

Table 7 presents the resource utilization of the proposed system, traditional system, and heterogeneous system. The proposed system shows a lower CPU utilization of 45% and a memory utilization of 30%, indicating that it is more efficient in terms of resource consumption compared to the traditional system, which has a CPU utilization of 75% and a memory utilization of 55%. The heterogeneous system falls between the two, with a CPU utilization of 60% and a memory utilization of 40%. These results highlight that the proposed system not only performs better in terms of scalability but also uses fewer computational resources, making it a more efficient solution overall.

Table 7: Resource utilization test results.

CPU Utilization (%)	Memory Utilization (%)
45	30
75	55
60	40

In addition, we analyzed the performance breakdown for different query complexities (simple, complex, multilingual). When dealing with simple queries, the average response time of the system is 0.23 seconds, compared with 1.56 seconds and 0.32 seconds for traditional and heterogeneous systems respectively. For complex queries, the system can effectively handle large-scale data sets and high concurrency requests, and the response time is kept within 0.42 seconds, which is significantly better than 4.78 seconds of traditional systems and 0.63 seconds of heterogeneous systems. In multi-language environment, this system achieves 95.6% query accuracy through built-in language recognition mechanism, compared with 88.4% and 92.3% for traditional system and heterogeneous system respectively.

These results show that the system performs well both in response speed and query accuracy.

In summary, the high-performance computing network search system realized in this paper shows excellent performance in different scenarios, and has obvious advantages and effectiveness compared with the other two systems. This paper proves the rationality and feasibility of th e methodology of this paper, as well as the development direction and potential of the network search system based on computer big data technology and high-performance computing technology.

5 Conclusion

In this paper, a high-performance computing network search system based on computer big data is proposed, which adopts a heterogeneous architecture and a parallel distributed computing model to effectively improve the efficiency, scalability and reliability of the system. Meanwhile, this paper also designs a storage management scheme based on cloud storage and grid computing technology, which realizes efficient storage and fast access of massive data. In order to verify the effectiveness and advantages of the methods in this paper, this paper builds an experimental environment on the Microsoft Azure cloud platform, uses virtual machines with different configurations to construct the three-tier architecture of the high-performance computing web search system, uses Common Crawl as the data source, and evaluates and compares the performance of the system from three aspects: response time, accuracy and stability. The HPC web search system implemented in this paper shows excellent performance in different scenarios and has obvious advantages and effectiveness compared to the other two systems.

Reference

[1] Abuein QQ, Shatnawi MQ, Yassein MB, Mahafza R: Intelligent system for visual web content analytics: A new approach and case study. Multimedia Tools and Applications, 2018, 77: 17557-17571. https://doi.org/10.1007/s11042-017-4740-8.

[2] Bashir S, Khattak AS: Private web search using proxy-query based query obfuscation scheme. IEEE Access, 2023, 11: 3607-3625. https://doi.org/10.1109/access.2023.3235000.

[3] Bashir S, Lai DTC, Malik OA: Proxy-terms based query obfuscation technique for private web search. IEEE Access, 2022, 10: 17845-17863. https://doi.org/10.1109/access.2022.3149929.

[4] Bhavithra J, Saradha A: Personalized web page recommendation using case-based clustering and weighted association rule mining. Cluster Computing-the Journal of Networks Software Tools and Applications, 2019, 22: S6991-S7002. https://doi.org/10.1007/s10586-018-2053-y.

[5] Chebil W, Wedyan MO, Lu HY, Elshaweesh OG: Context-aware personalized web search using navigation history. International Journal on Semantic Web and Information Systems, 2020, 16: 91-107. https://doi.org/10.4018/ijswis.2020040105.

[6] Choudhary J, Tomar DS, Singh DP: An efficient hybrid user profile based web search personalization through semantic crawler. National Academy Science Letters-India, 2019, 42: 105-108. https://doi.org/10.1007/s40009-018-0686-2.

[7] Ciortea A, Mayer S, Bienz S, Gandon F, Corby O: Autonomous search in a social and ubiquitous Web. Personal and Ubiquitous Computing, 2020. https://doi.org/10.1007/s00779-020-01415-1.

[8] Delgado AD, Montalvo S, Unanue RM, Fresno V: A survey of person name disambiguation on the Web. IEEE Access, 2018, 6: 59496-59514. https://doi.org/10.1109/access.2018.2874891.

[9] Dhanasekaran S, Vasudevan V: A cognizant agent system for optimizing cloud service searching strategy. Cluster Computing-the Journal of Networks Software Tools and Applications, 2019, 22: 13381-13386. https://doi.org/10.1007/s10586-018-1915-7.

[10] Geng JQ, Piao XF, Qu YB, Song HH, Zheng KX: Method for finding the important nodes of an electrical power system based on weighted-SALSA algorithm. IET Generation Transmission & Distribution, 2019, 13: 4933-4941. https://doi.org/10.1049/iet-gtd.2019.0424.

[11] Goel S, Kumar R: SoTaRePo: Society-Tag Relationship Protocol based architecture for UIP construction. Expert Systems with Applications, 2020, 141. https://doi.org/10.1016/j.eswa.2019.112955.

[12] Gopalakrishnan T, Sengottuvelan P, Bharathi A, Lokeshkumar R: An approach to webpage prediction method using variable order markov model in recommendation systems. Journal of Internet Technology, 2018, 19: 415-424. https://doi.org/10.3966/160792642018031902010.

[13] Guo HJ: Research on Web data mining based on topic crawler. Journal of Web Engineering, 2021, 20: 1131-1143. https://doi.org/10.13052/jwe1540-9589.20411.

[14] Inostrosa-Psijas A, Gil-Costa V, Marin M, Wainer G: Semi-asynchronous approximate parallel DEVS simulation of web search engines. Concurrency and Computation-Practice & Experience, 2018, 30. https://doi.org/10.1002/cpe.4149.

[15] Jovanovic M, Simic G, Cabarkapa M, Randelovic D, Nikolic V, Nedeljkovic S, Cisar P: SEFRA - Web-based framework customizable for serbian language search applications. Acta Polytechnica Hungarica, 2019, 16: 59-78. https://doi.org/10.12700/aph.16.3.2019.3.4.

[16] Jung J, Uejio CK, Duclos C, Jordan M: Using web data to improve surveillance for heat sensitive health outcomes. Environmental Health, 2019, 18. https://doi.org/10.1186/s12940-019-0499-x.

[17] Kalantari KR, Ebrahimnejad A, Motameni H:

Efficient improved ant colony optimisation algorithm for dynamic software rejuvenation in web services. IET Software, 2020, 14: 369-376. https://doi.org/10.1049/iet-sen.2019.0018.

[18] Kulshrestha J, Eslami M, Messias J, Zafar MB, Ghosh S, Gummadi KP, Karahalios K: Search bias quantification: investigating political bias in social media and web search. Information Retrieval Journal, 2019, 22: 188-227. https://doi.org/10.1007/s10791-018-9341-2.

[19] Kumar KNA, Chitra S, Kumar TS: Probabilistic classification techniques to perform geographical labeling of web objects. Cluster Computing-the Journal of Networks Software Tools and Applications, 2019, 22: 277-285. https://doi.org/10.1007/s10586-018-1822-y.

[20] Lu XY, Chen MS, Wu JL, Chang PC, Chen MH: A novel ensemble decision tree based on under-sampling and clonal selection for web spam detection. Pattern Analysis and Applications, 2018, 21:741-754. https://doi.org/10.1007/s10044-017-0602-2.

[21] Mahdi MN, Ahmad AR, Natiq H, Subhi MA, Qassim QS: Comprehensive review and future research directions on dynamic faceted search. Applied Sciences-Basel, 2021, 11. https://doi.org/10.3390/app11178113.

[22] Fan W, Pei J, Ding S, Pardalos PM, Kong M, Yang S. A novel trust inference framework for web-based scenarios harnessed by social network and web of trust - a heuristic approach. Informatica, 2016, 27(2):405-432. https://doi.org/10.15388/Informatica.2016.92.

[23] Mockus J. Bayesian heuristic approach to scheduling. Informatica, 2002, 13(3):311-332. 10.3233/INF-2002-13305.

[24] Malhotra D, Rishi OP: IMSS-P: An intelligent approach to design & development of personalized meta search & page ranking system. Journal of King Saud University-Computer and Information Sciences, 2022, 34: 248-263. https://doi.org/10.1016/j.jksuci.2018.11.013.

[25] Rahman MM, Abdullah NA: A personalized group-based recommendation approach for Web search in e-learning. IEEE Access, 2018, 6: 34166-34178. https://doi.org/10.1109/access.2018.2850376.

[26] Robertson S: A brief history of search results ranking. IEEE Annals of the History of Computing, 2019, 41: 22-28. https://doi.org/10.1109/mahc.2019.2897559.

[27] Senthilkumar NC, Reddy CP: Collaborative search engine for enhancing personalized user search based on domain knowledge. Journal of Medical Systems, 2019, 43. https://doi.org/10.1007/s10916-019-1350-1.

[28] Serrano W, Gelenbe E: The Random Neural Network in a neurocomputing application for Web search. Neurocomputing, 2018, 280: 123-134. https://doi.org/10.1016/j.neucom.2017.08.075.

[29] Sung HY, Chi YL: A knowledge-based system to find over-the-counter medicines for self-medication. Journal of Biomedical Informatics, 2020, 108. https://doi.org/10.1016/j.jbi.2020.103504.

[30] Wei CK, Gu QC, Ji SL, Chen WZ, Wang ZH, Beyah R: OB-WSPES: A uniform evaluation system for obfuscation-based Web search privacy. IEEE Transactions on Dependable and Secure Computing, 2021, 18: 2719-2735. https://doi.org/10.1109/tdsc.2019.2962440.