Data Structures Access Model for Remote Shared Memory

TECHNICAL CODETHAN REPORT SUBMITTED TO

RAMAIAH INSTITUE OF TECHNOLOGY
(Autonomous Institute, Affiliated to VTU)

Bangalore – 560054

SUBMITTED BY

Mihika Dhariwal	**1MS21CS075**
Kavyasri R	**1MS21CS063**
Jyoti Yadav	**1MS21CS056**
Manoj V.L.	**1MS21CS072**

As part of the Course **Data Structures-CS33**

SUPERVISED BY

Faculty
Prof. Nandini S B

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAMAIAH INSTITUTE OF TECHNOLOGY

Dec 2022

Department of Computer Science and Engineering
Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)
Bangalore – 54

CERTIFICATE

This is to certify that **Mihika Dhariwal (1MS21CS075), Kavyasri R (1MS21CS063), Jyoti Yadav (1MS21CS056) and Manoj V.L. (1MS21CS072)** have completed the "**Data Structures Access Model for Remote Shared Memory"** as part of Technical Codethan. We declare that the entire content embodied in this B.E. 3rd Semester report contents are not copied.

Submitted by:

Name: Mihika Dhariwal USN: 1MS21CS075
Name: Kavyasri R USN: 1MS21CS063
Name: Jyoti Yadav USN: 1MS21CS056
Name: Manoj V.L USN: 1MS21CS072

Guided by:

Prof Nandini S B
(Dept. of CSE, RIT)

Department of Computer Science and Engineering
Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)
Bangalore – 54

Evaluation Sheet

Sl. No	USN	Name	Research Content understanding and Coding (10)	Demo & Report submission (10)	Total Marks (20)

Evaluated By:

Name: Prof Nandini S B
Designation: Associate Professor
Department: Computer Science and Engineering, RIT

HOD, CSE

Table of Contents

Topic	Page No.
1. Abstract	1
2. Introduction	2
3. Literature Survey	3
4. Abstract Data Type	4
5. Implementation	5
6. Results and Discussions	9
7. Conclusion	10
8. References	11

1. Abstract

Recent achievements in high-performance computing have significantly narrowed the performance gap between single and multi-node computing, and opened up opportunities for systems with remote shared memory. The combination of in-memory storage, remote direct memory access and remote calls requires rethinking how data can be organised, protected and queried in distributed systems. This report begins with introducing the concept of distributed computing, its various applications and the two traditional message passing models used to exchange data between different nodes. It then goes on to introduce Remote Direct Memory Access (RDMA) and its role in achieving high-bandwidth, low-latency communication between the nodes in a distributed system. It presents a data model for mixed memory access, by analysing the use of distributed hash tables to implement RDMA. Additionally, this report also includes a practical demonstration of the applications of hash tables through the use of a C program, that shows how they can be utilised in real-world scenarios, such as storing the Aadhaar Card details of individuals in a database.

2. Introduction

Distributed computing is used in a wide variety of fields including scientific research, and technical developments such as facial recognition, control systems and autopilots. It involves the use of multiple computers called nodes to perform a task, thereby increasing processing power, improving fault tolerance and reducing the time taken to solve complex problems. In distributed systems, there are two main models for how data is shared between nodes: the message passing model and the shared memory model.

Message Passing Model: The message passing model requires data to be transferred between nodes and processes so that it can be accessed by other nodes. In these networks, processes use sockets to request network resources from the operating system through an API.

Shared Memory Model: In the shared memory model, all nodes and processes read or write into any address in a global shared memory. This allows data to be shared and made accessible to all nodes in the system.

A message can either be a Read/Write operation or a Send/Receive operation.

With the growing popularity of distributed computing, the volume of traffic exchanged between individual nodes has been increasing. In such scenarios, it is challenging to exchange data using the traditional message passing interfaces mentioned above. To solve this problem, new tools called distributed data structure libraries have been developed which use a technology called Remote Direct Memory Access (RDMA) to ensure fast and efficient transfer of data between different nodes. RDMA enables two networked computers to exchange data in the main memory without involving either computer's processor, cache or operating system.

3. Literature Survey

The paper “Data Structures Access Model for Remote Shared Memory” discusses the use of high-performance computing to improve the efficiency of systems that allow multiple computers to access and manipulate the data stored in the memory of other devices. With the help of a data model, it explains how data structures can be optimised for use in such systems, and presents the results of experiments comparing the performance and limitations of different data structures under different conditions.

Through RDMA, one can easily access the memory of other remote nodes, and directly exchange messages without any OS/CPU intervention. RDMA is especially used in enterprise environments that use systems with up to 16 machines. However, one potential issue is its scalability. When a system needs to transfer large amounts of data between multiple computers, the number of RDMA connections and data transfers can become very large. Not only does this strain the resources of the computer and the network, but it also leads to decreased performance and possibly even system failures. It is hence important to optimise data structures used to implement RDMA, to improve performance and enable systems to be able to handle increased workloads.

The Berkeley Container Library (BCL) is a software library that provides a set of distributed data structures namely distributed hash tables, distributed graphs and distributed arrays for use in parallel and distributed computing applications.

Distributed hash tables (DHTs) are data structures that are used to store data in distributed systems. A DHT is organised as a hash table that is distributed across multiple nodes in the system, with each node responsible for storing a portion of the data. In the context of RDMA, DHTs are used to store the memory addresses of data that is stored in the memory of different nodes in the system. This allows a node to perform an RDMA operation by quickly looking up the memory address of the data it needs in the DHT, rather than having to communicate with the other node to obtain the address. This allows data to be quickly accessed, updated and transferred between nodes with minimal overhead and low latency.

4. Abstract Data Type

Objects:

A collection of key-value pairs. Each key in a hash table is unique and is used to access its corresponding value.

Functions:

- **bool insert(key, item):** inserts an item into the hash table with an associated key. Returns true if the item was successfully inserted, else returns false.
  Example: void insert(int key,int data) {
  struct DataItem *item = (struct DataItem*) malloc(sizeof(struct DataItem));
  item->data = data;
  item->key = key;
  }
  .
- **int delete(key)**: removes the item associated with the specified key from the hash table. Returns the deleted item back to the main function.
  Example: void delete(struct DataItem* item) {
  int key = item->key;
  }

.

- **bool search(key)**: finds the item associated with the specified key in the hash table. Returns true if a match is found, else returns false.
  Example: struct DataItem *search(int key) {
  int hashIndex = hashCode(key);
  while(hashArray[hashIndex] != NULL) {
  if(hashArray[hashIndex]->key == key)
  return hashArray[hashIndex];
  ++hashIndex;
  hashIndex %= SIZE;
  }
  return NULL;
  }

.

- **void print_table():** prints the contents of the hash table.
  Example: void print() {
  for(int i = 0; i < size; i++)
  print(arr[i]);
  }

5. Implementation

```
#include <stdio.h>
#include<stdlib.h>
#include<stdbool.h>
#include<string.h>
#define MAX_NAME 256
#define TABLE_SIZE 10

typedef struct {
    char name[MAX_NAME];
    char address[100];
    char aadhaar_number[15];
    int age;
}person;

person p[TABLE_SIZE];
person * hash_table[TABLE_SIZE];

//hashing function
int hash(char * name){
    int length = strnlen(name, MAX_NAME);
    unsigned int hash_value= 0;
    for(int i=0; i<length; i++){
        hash_value+=name[i];
        hash_value=(hash_value*name[i])%TABLE_SIZE;
    }
    return hash_value;
}

void init_hash_table(){
    for(int i=0; i<TABLE_SIZE; i++){
        hash_table[i]=NULL;
    }
}

void print_table(){
    for(int i=0; i<TABLE_SIZE; i++){
        if(hash_table[i]==NULL){
            printf("%d\t---\n", i);
        }
        else{
```

```
            printf("%d\t%s\t\t\t", i, hash_table[i]->name);
            printf("\t%s\t\t\t", hash_table[i]->address);
            printf("\t%s\t\t",  hash_table[i]->aadhaar_number);
            printf("\t%d\t\t\t\n",  hash_table[i]->age);
        }
    }
}
//insert into hash table
bool insert(person * p){
    if(p==NULL)
    return false;
    int index = hash(p->name);
    {
    for(int i=0; i<TABLE_SIZE; i++){
        int try = (i+index)%TABLE_SIZE;
        if(hash_table[try]==NULL){
            hash_table[try]=p;
            return true;
        }
    }
    }
    return false;
}

//to find a person in the table by their name
person *lookup(char*name){
    int index = hash(name);
    for(int i=0; i<TABLE_SIZE; i++){
        int try = (index +i)%TABLE_SIZE;
        if(hash_table[try]!=NULL && strncmp(hash_table[try]->name, name,
TABLE_SIZE)==0){
            return hash_table[try];
        }
    }
    return NULL;
}

person* delete_name(char*name){
    int index= hash(name);
     for(int i=0; i<TABLE_SIZE; i++){
         int try = (index +i)%TABLE_SIZE;
         if(hash_table[try]!=NULL && strncmp(hash_table[try]->name, name,
TABLE_SIZE)==0){
            person * tmp = hash_table[try];
```

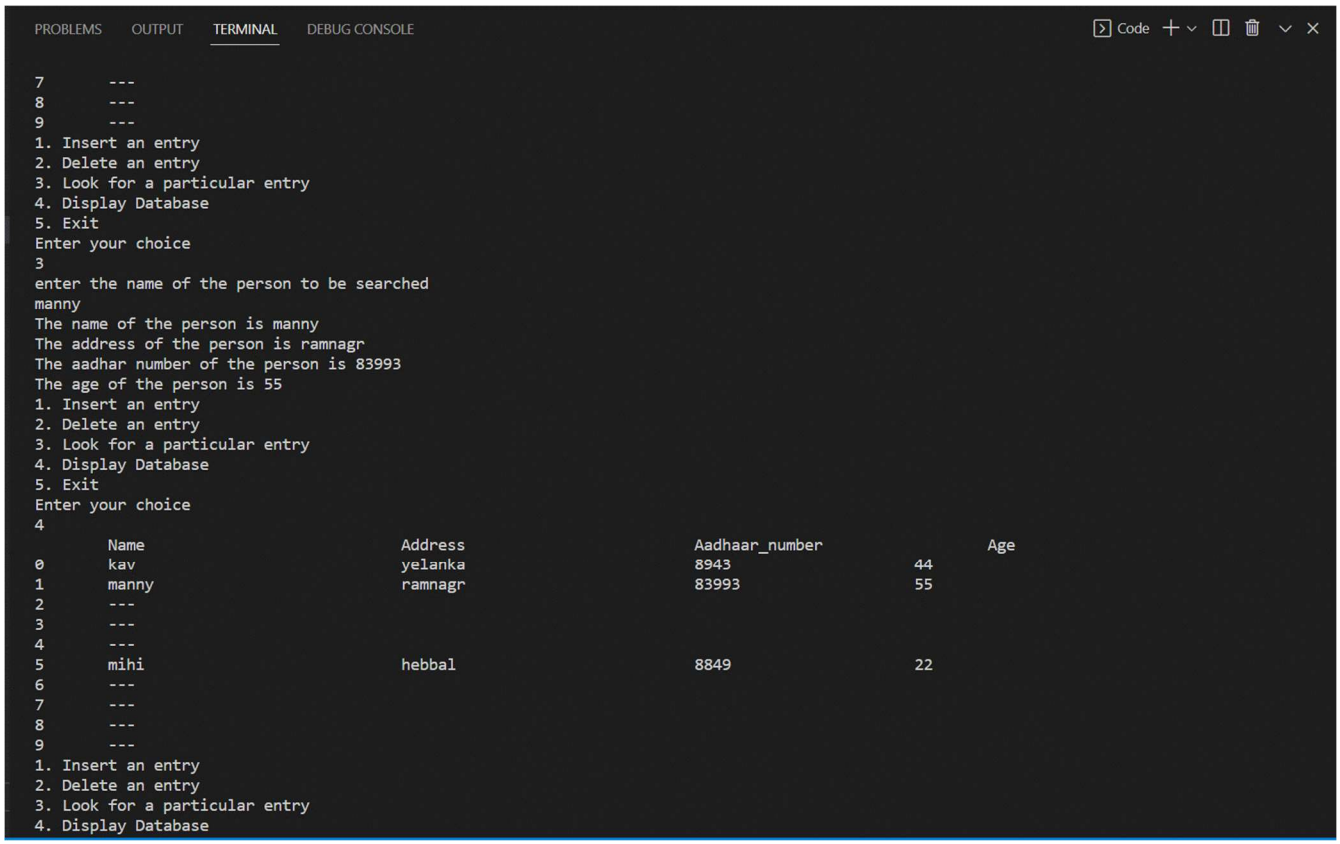
```
            hash_table[try]=NULL;
            return tmp;
          }
      }
   return NULL;
}

int main() {
    int i=0;
    person * pointer;
    person* deleted_name;
    char name[TABLE_SIZE];
    int choice, ch=0;
    char c;
    init_hash_table();
    printf("\t\t\t---------Aadhaar card details database!--------- \n\n");
     while(ch==0){
    printf("1. Insert an entry \n");
    printf("2. Delete an entry \n");
    printf("3. Look for a particular entry \n");
    printf("4. Display Database\n");
    printf("5. Exit\n");
    printf("Enter your choice \n");
    scanf("%d", &choice);

    switch(choice){
        case 1:
        for(int i=0; i<10; i++){
        printf("Enter name: ");
        scanf(" %[^\n]" , p[i].name);
         printf("Enter address: ");
        scanf(" %[^\n]", p[i].address);
         printf("Enter Aadhaar card number: ");
        scanf(" %[^\n]", p[i].aadhaar_number);
         printf("Enter age: ");
        scanf("%d", &p[i].age);
        insert(&p[i]);
        printf("Enter another person's details? Enter Y/N \t");
        scanf(" %c", &c);
        if(c=='y'|| c=='Y'){
            continue;
        }
        else{
            break;
```

```
        }
        }
       break;

        case 2: {
           printf("Enter the person name to be deleted\n");
           scanf("%s",&name);
           deleted_name=delete_name(name);
           if(deleted_name==NULL)
           printf("Person not found in database! \n");
           else
           printf("The info of %s has been deleted\n",deleted_name->name);
           break;
        }

       case 3:
       {
        printf("enter the name of the person to be searched\n");
        scanf("%s",&name);
        pointer=lookup(name);
        if(pointer==NULL)
        printf("Person not found in database! \n");
        else{
        printf("The name of the person is %s\n",pointer->name);
          printf("The address of the person is %s\n",pointer->address);
         // printf("The date of birth of the person is %s\n",pointer->dob);
      printf("The aadhar number of the person is %s\n",pointer->aadhaar_number);
      printf("The age of the person is %d\n",pointer->age);
        }
        break;
        }
       case 4 :
       {
        print_table();
        break;
       }
       case 5:
        exit(0);
        break;
    }
    }
   return 0;
}
```

6. Results and Discussions

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE                                  Code

                    ---------Aadhaar card details database!---------

1. Insert an entry
2. Delete an entry
3. Look for a particular entry
4. Display Database
5. Exit
Enter your choice
1
Enter name: mihi
Enter address: hebbal
Enter Aadhaar card number: 8849
Enter age: 22

Enter another person's details? Enter Y/N       y
Enter name: joe
Enter address: rtnagar
Enter Aadhaar card number: 7388
Enter age: 33

Enter another person's details? Enter Y/N       y
Enter name: kav
Enter address: yelanka
Enter Aadhaar card number: 8943
Enter age: 44

Enter another person's details? Enter Y/N       y
Enter name: manny
Enter address: ramnagr
Enter Aadhaar card number: 83993
Enter age: 55

Enter another person's details? Enter Y/N       n
1. Insert an entry
2. Delete an entry
3. Look for a particular entry
4. Display Database
5. Exit
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE                                  Code

Enter your choice
4
       Name                    Address                 Aadhaar_number          Age
0      kav                     yelanka                 8943                    44
1      manny                   ramnagr                 83993                   55
2      ---
3      ---
4      ---
5      mihi                    hebbal                  8849                    22
6      ---
7      ---
8      joe                     rtnagar                 7388                    33
9      ---
1. Insert an entry
2. Delete an entry
3. Look for a particular entry
4. Display Database
5. Exit
Enter your choice
2
Enter the person name to be deleted
joe
The info of joe has been deleted
1. Insert an entry
2. Delete an entry
3. Look for a particular entry
4. Display Database
5. Exit
Enter your choice
4
       Name                    Address                 Aadhaar_number          Age
0      kav                     yelanka                 8943                    44
1      manny                   ramnagr                 83993                   55
2      ---
3      ---
4      ---
5      mihi                    hebbal                  8849                    22
6      ---
```


PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
Code
7 ---
8 ---
9 ---
1. Insert an entry
2. Delete an entry
3. Look for a particular entry
4. Display Database
5. Exit
Enter your choice
3
enter the name of the person to be searched
manny
The name of the person is manny
The address of the person is ramnagr
The aadhar number of the person is 83993
The age of the person is 55
1. Insert an entry
2. Delete an entry
3. Look for a particular entry
4. Display Database
5. Exit
Enter your choice
4
Name Address Aadhaar_number Age
0 kav yelanka 8943 44
1 manny ramnagr 83993 55
2 ---
3 ---
4 ---
5 mihi hebbal 8849 22
6 ---
7 ---
8 ---
9 ---
1. Insert an entry
2. Delete an entry
3. Look for a particular entry
4. Display Database

7. Conclusion

Distributed computing and RDMA have had a significant impact on the way systems are designed and used, and are likely to continue playing a vital role in the future of computing. They have greatly enhanced the performance of distributed systems by accelerating the processing of large amounts of data, and allowing its fast and efficient transfer between nodes. This has in turn enabled the development of larger and more sophisticated applications capable of tackling a wider range of problems and workloads.

One of the many ways that RDMA and distributed computing can be used, is in the implementation of databases using distributed hash tables (DHTs). DHTs can be effectively used to implement databases by storing database records as key-value pairs. DHTs offer several benefits for implementing databases, including the ability to horizontally scale the database by adding more nodes to the network, and a high level of availability and resilience to help ensure that the database remains accessible even if some nodes go offline or experience problems. As a result, DHTs have become a popular choice for implementing distributed database systems that need to handle a large volume of data or a high number of requests.

This report describes the implementation of an Aadhaar card details database on a single node using hash tables. In a multi-node system, this data would be distributed across multiple nodes in the network and would be accessible by clients from any node. One can perform various operations such as inserting, deleting, searching for, and displaying entries from the database.

8. References

Paper title: Data Structures Access Model for Remote Shared Memory

Author Names: Anatoliy Nyrkov, Konstantin Ianiushkin, Andrey Nyrkov, Yulia Romanova, and Vagiz Gaskarov.

Year of publication online: 2021

Where it has been published: E3S Web Conf. XXII International Scientific Conference Energy Management of Municipal Facilities and Sustainable Energy Technologies (EMMFT-2020)

Volume and article Number: Volume 244, Article 07001