Distributed parallel data structure of a traffic network simulation based on object-oriented programming

Z. Juan[1], L. Gao[2], A. Ni[2] & G. Zhang[2]
[1]*Institute of Transportation Studies, Shanghai Jiao Tong University, People's Republic of China*
[2]*Transportation College, Jilin University, People's Republic of China*

Abstract

Traffic simulation design ideas based on object-oriented programming and modelling theory are used to analyse the data structure of a simulation system for a traffic network. A traffic network simulation consists of vehicles, links, intersections and signal controls. By defining the class in C++ language, this paper establishes objects of traffic network units, describes the variables and functions of their members in detail and exactly expresses the relationship between nodes and links in a traffic network. It constructs the shared data of the traffic network simulation based on a standard library function of template and object-classes of the traffic network. It uses SQL database technology to access the parallel data structure. Thus it can reduce the occupation of memory resources and increase the speed of data access. Each simulation unit can access the network data expediently. Finally, by simulating a traffic network made of four intersections in Changchun city, results indicate that the simulation speed increases 2.5 times and the error rate is less than 10%. Hence, a distributed parallel data structure based on object-oriented programming is the foundation for improving speed and for benefiting the traffic network simulation.
Keywords: object-oriented programming, traffic network, parallel data structure, parallel simulation.

1 Introduction

A traffic network distributed parallel simulation system studies the content of a traffic field. The system keeps several computers of the same or different

WIT Transactions on The Built Environment, Vol 89,
www.witpress.com, ISSN 1743-3509 (on-line)
doi:10.2495/UT060331

architectures connected to one another and constitutes a traffic simulation platform. The system provides real-time traffic-inducing information to the travellers by parallel simulation of a large-scale traffic network. The distributed system has a great potential for improving the simulation speed for parallel, extension and module. By increasing the scale of the traffic network, the system extends the simulation platform and makes it more applicable, thus meeting the new criteria and configuring to the different traffic networks. It is important to study traffic networks in order to develop effective real-time traffic flow simulation models by making full use of the parallel resource of a distributed system. Meanwhile, distributed parallel traffic simulation is also an effective tool for traffic engineers to solve traffic overcrowding problems in cities. Studies of parallel traffic simulation systems are conducted on massive parallel processors abroad. Jilin University and Beijing Jiaotong University undertake in-depth research into the algorithm for distributed parallel simulation at home.

The program is composed of data structures and algorithms. Building a data structure is key to traffic network simulation, because it is directly related to the degree of difficulty when developing programs and the effectiveness of a program run. The role of data structure in the whole traffic network simulation platform is shown in Fig. 1. A data computing server is responsible for storing data structure for the whole traffic simulation. Other components deal with data through a switcher.

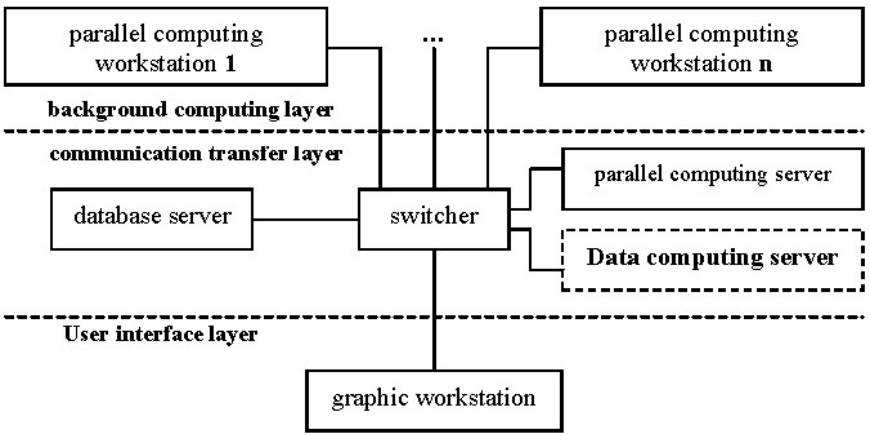


Figure 1: The status of data structure in distributed parallel simulating platform.

This paper builds an object-oriented data structure and simulates a traffic network that consists of four intersections in Changchun city. By analyzing the parallel benefit and comparing the simulation results with the actual results, the foundation is laid for studying and developing a traffic network distributed parallel simulation system.

2 Characteristics of a traffic network simulation data structure

Traffic network simulation computation is based on a fixed time step, so with the time increment, computation needs to be done repeatedly with large amounts of computation. Defining a traffic network data structure effectively is a key to traffic network simulation programs. When organizing traffic network data, as little memory resources as possible should be used and the relationship between data is readable, easy to call, and maintains the program. Good data structure should not only be simple and clear but also be convenient to deal with. On the whole, from the point of view of parallel traffic simulation modelling, the data structure of a traffic network simulation system has three characteristics: (1) there is a large amount of data and there is a high level of access to the simulation; (2) once the network is defined, geometry and the relationship between node and link is decided; (3) the access of the simulation modules to the data structure is parallel.

According to the characteristics of the traffic network simulation system, basic data structures and sharing parallel data structures are built to satisfy the needs of parallel simulations.

3 Object-oriented data structure of traffic network simulation

3.1 The object-oriented programming idea

Programming consists of structured programming and object-oriented programming. The main idea behind structured programming is function decomposition and to make the result more precise step-by-step. By means of module decomposition and function abstraction, the design task of a complex program system is effectively separated into many sub-tasks that are easy to control and handle. Structured programming offers a strong means for handling complex problems. However, it is still a kind of data-oriented design method. It divides the course of data and handling data into independent objects. The data format must constantly be considered in the course of programming. Matching data and program course has become a difficult task for programmers.

The essence of object-oriented programming is to regard the data and the handling data as a single object. Object-oriented programming is a trend of software systematic design and realization. By increasing the expansion and repetition capabilities of software, object-oriented programming improves the efficiency of program compilation and controls the expense and complication of maintaining software. Object-oriented programming takes advantage of structured programming. It puts data and data operation together and makes them an interdependent and indivisible whole. Meanwhile, data abstraction and information encryption make object and object operation abstract a new data type, namely class. Member variables in a class can be accessed through member functions.

Object-oriented programming has four characteristics: encapsulation, derivation, inheritance and polymorphism. The C++ language supports object-oriented programming fully. User-defined type supports encapsulation and data encryption. The class becomes an encapsulated entity and can be used as a whole unit. The operation in the class is hidden. The user only knows how to use this class and does not need to know how the class works.

The basic units of a traffic network simulation system consist of a vehicle, a link, a lane, an intersection, a signal and a clock. Each unit of the traffic network and network structure can be regarded as a single object. The interaction between them constitutes the simulation process of the whole network. The method of object-oriented programming may define this traffic network. Hence, defining the network data structure is the way to define the member variables and the member functions of the basic objects of a traffic network simulation system and the relationship between those objects. This paper establishes the data structure of a traffic network simulation by defining the object-oriented class in C++ language.

3.2 Definition of object-oriented class

Basic units such as link, intersection and signal control can represent a traffic network. Microscopic traffic simulation simulates the behaviour of an individual vehicle run in the whole traffic network. The main task of a traffic simulation system is to analyse and study the rules of movement of the system entity and property, assess the performance of the system and assist in the management decision-making and system design. Traffic simulation systems possess complex entities and property. When an applicable study is conducted, the actual problem is simplified and hypothetical. A proper model is then built to solve the problem. Entity in traffic network consists of a vehicle, a link, a lane, an intersection, a signal and a clock etc. According to the relationship between all entities, these entities are mapped to object-oriented simulation space and fall into abstract classes: Tvehicle, Tlink, Tlane, Tintersection, Tsignal and Toclock. According to the characteristics of each class and the relationship between them, the member variables and member functions of each class are also defined. The attributes of each class are introduced as follows.

3.2.1 Vehicle class

Vehicle class describes the attributes of a vehicle entity and produces vehicles constantly to ensure a simulation run. The member variables and member functions in vehicle class reflect the dynamic nature of a vehicle from different aspects when running in a network. Vehicle attributes defined in this paper include private, protected and public attributes. These attributes prescribe the degree open to other classes. For detailed descriptions of the individual vehicle behaviour in a traffic network, private attributes in the vehicle class include: vehicle ID, length, width, type, state, location, speed, max speed, desired speed, average speed, acceleration, max acceleration, max deceleration, front vehicle ID, rear vehicle ID, turn direction and degree of driver skill and driver risk tendency etc.

3.2.2 Link class

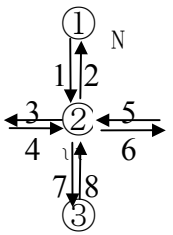
In a network graph, a link is a line with a direction that connects two intersections or nodes. A link is a basic unit in a traffic network. The definition of link class is similar to vehicle class. The main member variables include private and public attributes. Private variables include link ID, link length, the lane number, capacity, free flow speed, downstream node ID and upstream node ID etc. Member functions include set_current_link() and get_current_link(), which set and decide current link IDs and their attributes in the network, set_down_link() and get_down_link(), which set the next link ID the current vehicle is going into. Link class is indispensable in a network object-oriented simulation, which is used to describe the single link attribute and clearly show the interlink relationship between nodes or links in the network.

3.2.3 Lane class

Since vehicles are moving in lanes, it is important to investigate the queue of vehicles in that lane. As a factor that is closely related to the link, the lane can form a single class. Lane variables include lane width, turn direction, vehicle queue, max queue length, type of lane control etc. Lane class is a basic unit in the microscopic simulation.

3.2.4 Intersection class

Different variables in the intersection class describe different attributes of the intersection, such as intersection ID, ID of the upstream and downstream intersections, ID of each directional link, and each entry link attribute. The intersection class is a basic unit in the microscopic simulation. The following example explains the basic structure of the intersection class as shown in Fig. 2.



Class object node 2 is defined as follows:

North entrance entry/exit link= (1,2)
South entrance entry /exit link = (7,8)
East entrance entry /exit link = (5,6)
West entrance entry /exit link = (3,4)

If link 1 is the entrance to node 2 and the running direction of the current vehicle is through, the upstream node of node 2 is 1; the downstream node is 3 and next entry link is 7.

Figure 2: The definition of the intersection class structure.

3.2.5 Signal class

Signal class describes the signal control parameters such as red and green intervals. It controls the right of way of the vehicles in each lane. Its member variables include signal ID, cycle length, the colour of the signal, red time, green time, yellow time, phase sequence, split etc.

3.2.6 Clock class

Clock class controls the clock in advance in the simulation and is updated as events take place. It controls the finishing time of the simulation and is often used in the timing module. The clock class variables include: current time, simulation time step, and simulation time.

All classes are defined as the following codes:

```
Class T_name      // name of entity class
{
public:
      define member function;
private:
      define member variable;
};
```

We can learn that are generally independent and can only be called by relevant functions from the above codes.

After the entity classes of the traffic network are defined, we declare class objects in the program by the following codes:

```
T_name name;      //class object
```

Name is a declared object and it controls each object by programming.

3.3 Deque queue structure based on C++ programming

After the class of each entity in a traffic network is defined, each object is only a unit of a traffic network simulation system object set. For example, a vehicle object is only a unit of all vehicle objects in the whole network. Sharing the data structure is built to facilitate parallel access to the data and improve the speed of access to the data. Sharing the data structure defines the queue structure of objects or pointers in the form of <deque> module function. This is similar to a chain table. This paper defines four deque queue structures:

```
typedef deque<TVehicle> TVEHICLELIST;      //define vehicle queue
typedef deque<TLink> TLINKLIST;            //define link queue
typedef deque<TIntersection> TINTERSECTIONLIST;
//define intersection queue
typedef deque<TSignal> TSIGNALLIST;        //define signal queue
```

TVEHICLELIST: vehicle deque queue structure, used to save class object of all vehicles produced in network

TLINKLIST: link deque queue structure, used to save class object of all links in network

TINTERSECTIONLIST: intersection deque queue structure, used to save class object of all intersections in network

TSIGNALLIST: signal deque queue structure, used to save class object of all signals in network.

Then building queue objects through the following sentences:

```
TVEHICLELIST vehiclelist;                  //define vehicle queue object
TLINKLIST linklist;                        //define link queue object
TINTERSECTIONLIST intersectionlist;  //define intersection queue object
TSIGNALLIST signallist;                    //define signal lamp queue object
```

All simulation modules can access the shared data at the same time and search, add, modify and delete things in the traffic network simulation by defining the sharing data structure. The parallel data structure is the foundation of increased speed and efficiency of traffic simulation.

4 Method of data structure storing

As the amount of data of the traffic network simulation becomes larger, storing the data structure by simple array will occupy a great deal of memory resources. This reduces the speed and efficiency of the simulation. To solve this problem, this paper saves the data structure by means of an SQL database. The simulation process is made possible by accessing a database. This method of database storing provides technical means for a distributed simulation of a large-scale traffic network. By setting an ODBC data source, the database is accessed. Detailed codes are as follows:

```
BOOL CDB::Connect(SQLCHAR* dsn,SQLCHAR* uid,SQLCHAR* pwd)
{
  long              V_OD_erg;        //define SQL RETURN
  char              V_OD_stat[10];          //state SQL
  SQLINTEGER            V_OD_err;
  SQLSMALLINT   V_OD_mlen;
  char                  V_OD_msg[200];
  // 1. Allocate Environment handle
  V_OD_erg=SQLAllocHandle(SQL_HANDLE_ENV,SQL_NULL_HANDLE,
    &m_V_OD_Env);
  V_OD_erg=SQLSetEnvAttr(m_V_OD_Env,SQL_ATTR_ODBC_VERSION,
(void*)SQL_OV_ODBC3, 0);                    //set application ODBC version
  // 2. Allocate connection handle
  V_OD_erg=SQLAllocHandle(SQL_HANDLE_DBC,m_V_OD_Env,
    m_V_OD_hdbc);
  SQLSetConnectAttr(m_V_OD_hdbc,SQL_LOGIN_TIMEOUT,
(SQLPOINTER *)5, 0);
  // 3. Connect to the data source "web"
  V_OD_erg=SQLConnect(m_V_OD_hdbc,dsn,SQL_NTS,uid,SQL_NTS,pwd,
    SQL_NTS);
  // 4. Band rows
  SQLBindCol(V_OD_hstmt,1,SQL_C_SHORT,&project_id,150,&V_OD_err);
```

```
   V_OD_erg=SQLExecDirect(V_OD_hstmt,(unsigned char *)"SELECT *
FROM tb_vehicle order by project_id,id ",SQL_NTS);  //execute SQL sentence
   V_OD_erg=SQLFetch(V_OD_hstmt);                   //fetch a record
   while(V_OD_erg != SQL_NO_DATA)
        {
            project_id1=project_id2;
              V_OD_erg=SQLFetch(V_OD_hstmt);
        }
   SQLFreeHandle(SQL_HANDLE_STMT,V_OD_hstmt);
   //release SQL sentence handle
   return TRUE;
}

BOOL CDB::Diconnect()
{
   SQLDisconnect(m_V_OD_hdbc);              //cut connection with data source
   SQLFreeHandle(SQL_HANDLE_DBC,m_V_OD_hdbc);
   //release connection handle
   SQLFreeHandle(SQL_HANDLE_ENV, m_V_OD_Env);
   //release environment handle
   return TRUE;
}
```

5 Application example

This paper tests the parallel simulation benefit of a program with a simulation time and the number of processors varying in the actual network in Changchun city based on an object-oriented data structure. The network consists of four intersections. The number of processors in a parallel environment is 1,2 and 4 respectively. Configuration is legend terminals (P4/2.4G/256M). Table 1 shows a comparison between the parallel and serial program execution times. The simulated average delay time and maximum queue are compared with the measured results in Table 2.

From these two tables, we can learn that an object-oriented data structure has been applied successfully.

Table 1: Comparison between parallel and serial program execution time.

simulation time(s)	serial program	parallel program		
		1	2	4
1000	76.49	274.87	251.08	111.84
2000	383.60	900.53	656.59	352.80
3600	1314.87	2439.73	1701.64	872.78
5400	2996.90	4332.85	3239.25	1621.70
7200	5342.98	7589.68	5024.53	2723.85
10800	12031.80	14805.93	9472.67	5053.22

Table 2: Comparison between measured and simulated results.

intersection ID	average delay(s)			maximum queue(veh)		
	measured	simulated	error(%)	measured	simulated	error(%)
1	42.36	40.52	4.3	37	34	8.1
2	44.14	42.38	4.0	34	36	5.9
3	40.38	38.26	5.3	25	23	8.0
4	40.79	37.64	7.7	26	27	3.8

6 Conclusion

Parallel data structure is fundamental to improving the speed and efficiency of a traffic network simulation. According to the complexity of a traffic network, this paper establishes a parallel data structure of a traffic network based on object-oriented programming. It can distinctly express the relationship between nodes and links in a traffic network and reduce the occupation of memory resources using database technology. It can also increase the speed and efficiency of a traffic simulation. Consequently, a parallel data structure is the foundation for developing distributed parallel traffic simulation systems.

Acknowledgements

This research is being funded by grant number 50378042 and 50338030 from the National Natural Science Foundation of China (NSFC). The authors would like to thank NSFC for their support.

References

[1] Thanavat Junchaya and Gang-Len Chang. Exploring Real-Time Traffic Simulation with Massively Parallel Computing Architecture[J]. Transportation Research , 1993 , 57-76.

[2] Wei Liying. Study on Modeling Distributed Microscopic Traffic Network Simulation [D]. Transportation college, Jilin University, 2002.

[3] Zhang Kunlin, Shao Chunfu, Wang Lishao. Study on Dynamic Traffic Assignment Based on Distributed Parallel Arithmetic [J]. Journal of Northern Jiaotong University, 2002,26(5): 57-61.

[4] Qian Neng. C++ programming tutorial [M]. Tsinghua University Press, 1999. 4-11.

[5] Wang Yan. object-oriented theory and C++ practice [M].Tsinghua University Press, 1997. 3-25.

[6] Guan Zhangquan. Standard C++ library. Publishing House of Electronics Industry, 2002. 56-84.