RESEARCH ARTICLE | APRIL 18 2018

Research on TCP/IP network communication based on Node.js

Jing Huang; Lixiong Cai

AIP Conf. Proc. 1955, 040115 (2018)
https://doi.org/10.1063/1.5033779

Research on TCP/IP Network Communication based on Node.js

Jing Huang, Lixiong Cai[a)]

School of Information, Zhejiang Sci-Tech University Hangzhou, China

[a)] superbear24@163.com

Abstract. In the face of big data, long connection and high synchronization, TCP/IP network communication will cause performance bottlenecks due to its blocking multi-threading service model. This paper presents a method of TCP/IP network communication protocol based on Node.js. On the basis of analyzing the characteristics of Node.js architecture and asynchronous non-blocking I/O model, the principle of its efficiency is discussed, and then compare and analyze the network communication model of TCP/IP protocol to expound the reasons why TCP/IP protocol stack is widely used in network communication. Finally, according to the large data and high concurrency in the large-scale grape growing environment monitoring process, a TCP server design based on Node.js is completed. The results show that the example runs stably and efficiently.

Key words: Node.js; TCP/IP; network communication; high concurrency; asynchronous I/O.

INTRODUCTION

With the rapid development of information science and technology and Internet of things technology, agriculture is also moving forward rapidly in the process of modernization. A small range of traditional agricultural planting outdoor planting in the greenhouse planting scale, the traditional manual data collection becomes the modern intelligent remote data acquisition. But with the expansion of the scale, increase the amount of data acquisition and frequency, multi thread blocking server has been unable to meet the data communication situation of this large data and high frequency.

Node. js [1] is a JavaScript platform based on Chrome V8 [2] engine. It has the characteristics of event driven, non-blocking, single thread and so on. It has good scalability, and it is very lightweight and efficient [3]. Node.js solves the performance problems of blocking multithreaded servers in the case of big data and high concurrency by changing the maximum number of connections to the response processing of a single system [4].

NODE.JS ASYNCHRONOUS NON BLOCKING MODEL

Node.js architecture

In additi on to using V8 engines to parse JavaScript outside [5], node.js uses the libev library to support event driven and use libeio libraries to asynchronously format I/O, and the architecture diagram of Node.js is depicted in Fig. 1.

Advances in Materials, Machinery, Electronics II
AIP Conf. Proc. 1955, 040115-1–040115-6; https://doi.org/10.1063/1.5033779
Published by AIP Publishing. 978-0-7354-1654-3/$30.00

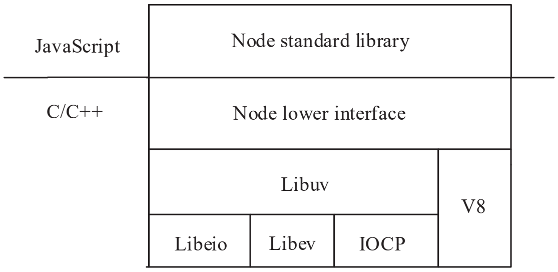

FIGURE 1. Node.js architecture

Because of the difference between Windows system and POSIX system, Node.js developers abstracted the libuv layer on the basis of libev and libeio. For the POSIX operating system, libuv uses epoll or kqueue [6] by encapsulating libev and libeio, while in Windows, libuv uses Windows's IOCP mechanism [7]. The Libuv layer ensures platform compatibility, enabling the upper layer of Node.js to be independent of the underlying libeio/libev and IOCP to achieve the same high performance on different platforms.

Asynchronous non-blocking I/O model

From the program execution effect, asynchronous and non-blocking both achieve the purpose of parallel I/O, but asynchronous / synchronous and blocking / nonblocking are two different things. The distinction between blocking and non-blocking is defined by the way the OS kernel handles the I/O differently, while asynchronous differentiation is required when the application calls the asynchronous API, and calls the synchronous API without waiting for [5].

The asynchronous I/O of Node.js mainly involves three objects, which are event loop, observer and request object [8].

Event cycle—When you start the Node.js system process, the system creates a loop to handle the events that are received during the execution of the process. This loop is called event loop. The system will always execute the event cycle to see if there is a new event registration, if there is, execute the event and call the callback function of the event object, and then continue to execute the event loop [1,9].

Observer—In the event that each event cycle is executed, it is necessary to introduce an object, called observer, to determine whether there is an event. The function of the observer is to determine whether there is an event that needs to be dealt with during the execution of the event cycle. There are usually multiple observers in the event loop, and different observers are responsible for monitoring and handling different events, called observer subscription events. The essence of event cycles is to cycle the viewer to subscribe events.

Request object—The request object is the intermediate product when executing asynchronous I/O. The request object keeps the state and intermediate product when executing asynchronous I/O. The event is the request object essentially in the event cycle, and the target object that the observer subscribes is also the request object.

Node.js asynchronous I/O flow is depicted in Fig. 2

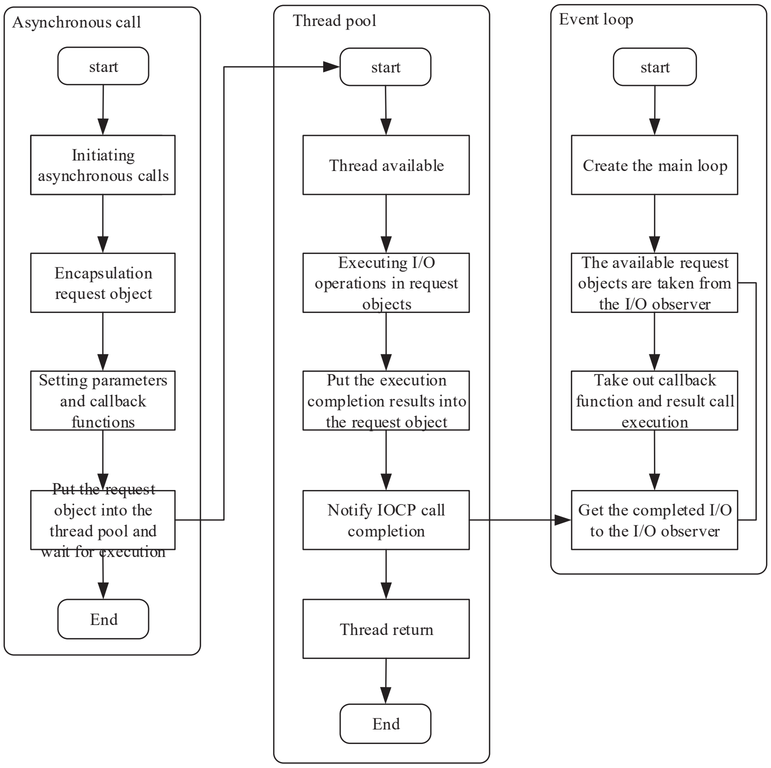
Asynchronous call
start
Initiating asynchronous calls
Encapsulation request object
Setting parameters and callback functions
Put the request object into the thread pool and wait for execution
End
Thread pool
start
Thread available
Executing I/O operations in request objects
Put the execution completion results into the request object
Notify IOCP call completion
Thread return
End
Event loop
start
Create the main loop
The available request objects are taken from the I/O observer
Take out callback function and result call execution
Get the completed I/O to the I/O observer

FIGURE 2. Node.js asynchronous I/O flow

The I/O call is sent to the system kernel through the IOCP under the Windows system. After the I/O operation is completed, the I/O is captured from the system kernel through IOCP. The entire asynchronous I/O process needs the event cycle to match with the call I/O and capture I/O. Under Linux, this process is realized by epoll, implemented by FreeBSD under kqueue, and implemented by Event ports under Solaris. The difference is that the thread pool is provided directly by IOCP under the Windows operating system, and is implemented by libuv under POSIX.

TCP/IP NETWORK COMMUNICATION MODEL

TCP/IP protocol, as a communication-oriented network protocol, aims to achieve reliable data transmission between two arbitrary computers. TCP can ensure data transmission from one system network to another system network, and the data can be delivered accurately and safely, which can ensure the safety of data transmission. The system structure of TCP/IP protocol is usually divided into four layers, from top to bottom are network interface layer, network layer, transport layer (host to host), and application layer [10], TCP/IP protocol four-layer model and OSI seven-layer model [11] contrast can be seen [12] in the Fig. 3.

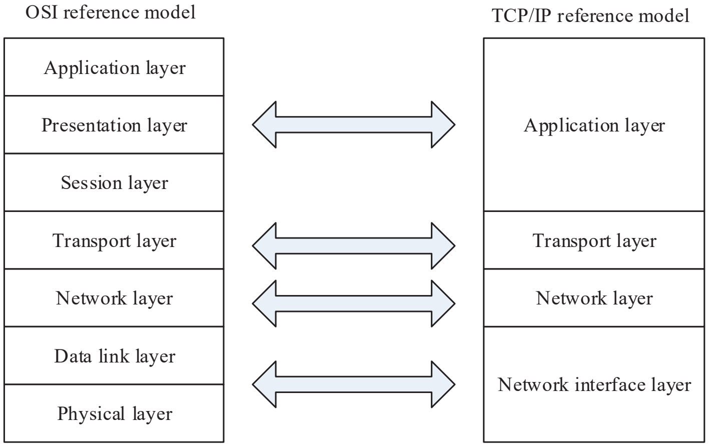

FIGURE 3. Comparison between OSI reference model and TCP/IP reference model

Application layer—The application layer provides a lot of services. Such as FTP, Telnet, DNS, SMTP and so on. The application layer of TCP/IP integrates the presentation layer and session layer function of OSI into the application layer.

Transport layer—The transport layer corresponds to the transport layer of the OSI model, usually dealing with problems such as reliability and traffic control. Two important agreements are defined in this layer: transmission control protocol (TCP) and user datagram protocol (UDP), transmission control protocol (TCP) to provide data transmission is reliable; and the user datagram protocol (UDP) data transmission service is not the guarantee of reliability.

Network layer—The main problem to be solved in the network layer is the communication between the specific computer and the target computer. The Internet Protocol (IP) is the most important protocol in this layer. It allows each computer to have a unique identification number and allows different computers and network devices to communicate with each other by assigning unique addresses to each computer. This layer corresponds to the network layer of OSI.

Network access layer—The network access layer corresponds to the the physical layer and the The data link layerin the OSI reference model. The main function of the network is to send datagram transmission control, is responsible for the encapsulated data sent to the target network computer [13].

Compared with TCP/IP network model and OSI network model, the designer of OSI network model is too perfect, which makes the whole model too complex and lacks flexibility, and the execution efficiency is low. The TCP/IP protocol discards the rarely used presentation layer and session layer in the OSI protocol. In the application layer to implement the representation layer and session layer function, safety and reliability and at the beginning of design on data transmission, while focusing on the universal connection, make use of the protocol in any network interconnection equipment can be set for communication in the network communication more vitality, more efficient and safer. Therefore, TCP/IP becomes the core of Internet network architecture.

AN APPLICATION OF TCP/IP NETWORK COMMUNICATION BASED ON NODE.JS IN THE MONITORING OF GRAPE GROWTH ENVIRONMENT

The growth environment of grape is strict with temperature, moisture and illumination. Suitable environment condition is the precondition of high quality grape growth. Therefore, it is necessary to record the growth environment parameters of grape with high density and high frequency, Through the analysis of environmental parameters to take external measures, so that the environmental parameters consistent with the optimal growth environment of grapes. In view of the large area grape planting in agricultural greenhouse. In this paper, a TCP network server based on Node.js TCP/IP network model is designed to accept the grape growth environment parameters transmitted by the bottom client under the condition of high frequency, big data and long connection, and the data is checked and stored after analysis.

In this system, the TCP network server created by Node.js will detect whether there is a client connection after startup, and when the client connection is detected, the server sends the data request instruction to the client, The client uses the W5100 monolithic network interface chip, W5100 module is connected with the Arduino UNO development board, the client receives the TCP server request instruction, Arduino UNO development board reads sensor data, then the data sent by the network chip W5100.The server receives the data sent by the client, verifies the received sensor data, analyzes the data after verification, and stores the parsed data to the MySQL database.

Flow chart is depicted in Fig. 4

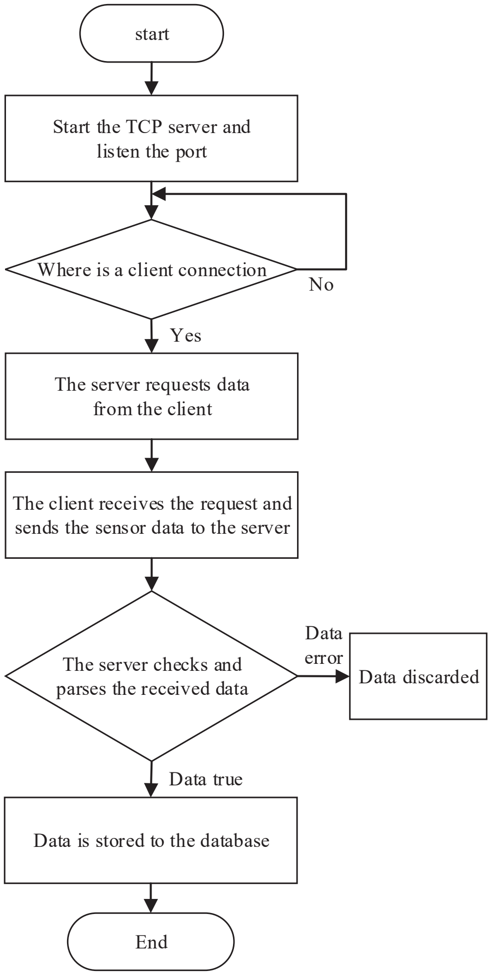
start
Start the TCP server and listen the port
Where is a client connection
No
Yes
The server requests data from the client
The client receives the request and sends the sensor data to the server
The server checks and parses the received data
Data error
Data discarded
Data true
Data is stored to the database
End

FIGURE 4. The example flow chart

Create server

The Net module is the core module of Node.js, mainly used to create TCP server, the Net module provides create Server method is used to create a server. Create a TCP server part of the code as follows:

```
const net = require('net');
const server = net. Create Server((client) => {
// 'connection' listener
console. Log ('client connected');
});
server. On ('error', (err) => {
throw err;
});
```

```
server. Listen (2017, () => {
console.log ('server bound');
});
```

Data acceptance and processing

The Net Module Socket is an object, which is a TCP or UNIX Socket abstract and be created by Node.js, and through the "server connection events" transfer to the user. Through the server to register 'data' event to realize the data monitoring, for the 'data' event to increase the callback function to achieve data processing.

TABLE 1. Some data received by the TCP server can be seen in the

Node number	Air temperature and humidity	Time
1	57.2/26.2	2016-6-19 01:19:28
2	57.6/25.9	2016-6-19 02:19:28
3	58.2/26.2	2016-6-19 02:19:28
4	59.2/26.0	2016-6-19 04:19:28

Experiments show that when large amounts of data are transmitted at high frequencies, the TCP server based on Node.js can stably and continuously receive the big data with high frequency.

CONCLUSION

Because of the single thread non-blocking model, Node.js can deal with big data and high concurrency in network communication rapidly and efficiently when facing big data and high frequency situation. This paper first analyzes the characteristics of Node.js architecture and the reasons for the high efficiency, and then expounds the advantages of TCP/IP model in the communication process, Finally, a TCP/IP network server based on Node.js is implemented, and the application of this server in the monitoring of greenhouse grape planting is realized. Under the condition of big data and high concurrency, the example is stable and reliable, and has good universality.

REFERENCES

1. BYVoid. Node.js Development Guide[M]. Posts and Telecommunications Press,1st.2012.
2. Gray J. Google Chrome: the making of a cross-platform browser [J]. Linux Journal, 2009.
3. Evented I/O for Javascript with Google V8-based Node.js [EB/OL].https://www.infoq.com/news/2009/11/nodejs-evented-io.2009-11-24.
4. LIU ShengJian, YUAN Guanyuan, YANG yan. Development of Matching System Based on Node.js [J]. Software Guide.Vol.13No.8, Aug.2014.
5. PIAO ling. Easy to understand node.js[M]. Beijing: Posts and Telecommunications Press,2013:1-12.
6. Cantelon M, Holowaychuk T J, Rajlich N, et al. Node. js in Action[M]. Manning Publications,2014.
7. DU Xiang, LEI Yueming. Server-Side Application Based on IOCP[J]. Computer Systems & Applications,2009.
8. XU Huiyuan, HE Lili. Asynchronous and Non－blocking I/O of NodeJS[J]. Industrial Control Computer,2015.
9. George Ornbo. Introduction to Node.js classics[M]. Post and Telecommunications Press,2013.
10. Behrouz A, Forouzan. Data Communication and Networking. China Machine Press,4th.2007.
11. Xie Xiren. Computer Networks,6th. Publishing House of Electronics Industry,6th.2013.
12. Kevin R. Fall, W. Richard Stevens. TCP/IP Illustrated Volume I:The Protocols. China Machine Press,2016.