[Volume 1 issue 2 March 2013]
Page No.69-72 ISSN :2320-7167
[International Journal Of Mathematics And Computer Research]

Low Cost Networked Storage

Vaibhav Jagannathan, Shounak Kamalapurkar, Ameya Mahabal, Gaurang Davda , Mukesh Tiwari,

Kuldip Wagh, Btech(Electronics) student , VJTI college of Engineering.

Abstract

Technology today is becoming increasingly mobile. It is now possible to all the work, which would require a large desktop computer a few years back, on a handheld tablet or a mobile phone. Today most people do not possess just one such device but a multiple devices. An individual may own laptops, tablets, cellular phones, desktops, any combination of such devices. Also the widespread popularity of wireless internet connections (wi-fi or mobile internet) has ensured that all these devices are connected to the 'World Wide Web' almost all the time. It is essential for the users to be able to share and synchronize data and files between these devices and the simplest way is to have a common repository of files on a server that can be accessed through the web. Existing services charge huge amounts for large amounts of storage and also the files are stored on servers that the user does not directly control and can be accessed by the company also making privacy and security matters of concern. This paper details the development of a low cost, personalized, network attached storage that will be accessible over the internet. The low cost Raspberry Pi board will be used as the hardware on which the server is hosted. The linux based Raspbian OS was used on the Raspberry Pi. The http server was implemented in server side javascript using node.js. The one time cost of the setup along with minimal recurring charges makes this solution feasible.

Keywords:

Cloud storage, Raspberry Pi, Raspbian, Network Protocols, Wireless Network.

I. INTRODUCTION

The Raspberry Pi is used as a remote http server. This server is used to host a website that allows the clients to access the storage attached to the server. This storage service can be used to upload any file from a remote location to the server. It can be used to browse through and download the files previously stored on the server.

II. THE SERVER

The assembly consists of a Raspberry Pi board which acts as the server. Raspberry Pi has a single core ARM processor and cannot efficiently handle multithread programming because of the large overhead associated with it and the limited computing power of the ARM processor. It is more efficient to implement a server using single thread asynchronous programming. Thus single threaded event driven programming can be efficiently executed on the single core processor of the Raspberry Pi.

A. Raspberry Pi

The Raspberry Pi is a low cost ($35), compact development board containing an ARM-11 System on Chip. It is possible to run operating systems on this device. It has a SD card slot. It contains an Ethernet port which can be used to connect the board to a local area network or the internet. It also contains 2 USB 2.0 ports which can be used for I/O devices like mice, keyboards, hubs, or storage devices. The operating system used in this project is the linux based Raspbian(wheezy) OS.

a. Raspbian OS

Raspbian is an operating system which is developed specifically to run on Raspberry Pi. Its operating system is based on debian so the name Raspbian. It is built with compilation settings adjusted to produce optimized 'hard float' code for the Raspberry Pi. The floating point arithmetic operations are significantly faster due to this. The advanced instructions of the ARMv6 CPU in the Raspberry Pi improve the performance of the many other applications.

b. Connecting to Internet

The Raspberry Pi can be connected to the internet using an Ethernet cable. One end is plugged into Raspberry Pi Ethernet port and the other into the back of the home network. The network LEDs will start to flicker. If the router is configured for DHCP (Dynamic host configuration protocol) then the IP address is provided to the Raspberry Pi and it is now connected to the internet. If DHCP is not turned on, on network router, then connect to its management console using a different computer that is already connected. Using tunneling the localhost can be made visible over the internet by associating the localhost with a domain name (DNS). A simpler approach could be to

assign a static IP address to the Raspberry Pi, using port forwarding and Dynamic DNS the localhost can be accessed through the web.

c. USB storage

There are 2 USB ports in the model B of Raspberry Pi. The uploaded data can be stored by the memory card put in the memory card slot of the Raspberry Pi. This capacity can be increased by inserting 'pen drives', 'flash drives', or 'hard disks' into the USB port provided on the Raspberry Pi. This way the data storage space can be altered as required. A USB hub may be used and multiple pen drives may be connected to increase the capacity. But a number of low-cost powered USB hubs are known to have caused problems. If a powered hub and the Raspberry Pi PSU together are powered from the same power bar with switch, they can turned on simultaneously, and if the HUB tries to feed the Raspberry Pi through the interconnect cable, due to the 100 mA limiting fuse in the Raspberry Pi, the Raspberry Pi will be partially powered which may cause unwanted write problems to the SD card. Another disadvantage of low-cost hubs is the speed limitation they impose. However connecting a portable hard disk requires some modifications since the drive is not self powered. So an external power supply is required to power the hard disk.

B. NODE.JS

Node.js is a server-side software system designed for writing scalable Internet applications, notably web servers. It allows the server-side execution of Javascript code i.e. the code is run in the backend, outside a browser. The Javascript code is interpreted as well as executed using Google's V8 VM (Google Chrome's Javascript Runtime Environment).

Node.js makes use of an event driven, non-blocking I/O model that makes it lightweight and efficient, perfect for a data-intensive applications that runs on a single core processor with limited processing power, like the Raspberry Pi's single core ARM processor. The resulting Node web applications are both fast and scalable with minimum overhead for the processor.

Node.js is essentially comprised of a runtime environment and a library. The library is a collection of modules which are themselves Javascript code providing abstractions from the basic hardware to form a simpler Application Programming Interface (API).

a. Javascript

JavaScript (JS) is an interpreted computer programming language formalized in the ECMAScript language standard. It was originally developed and implemented as part of web browser so that client-side scripts may interact with the user to control the data displayed on a webpage. Over time it was standardized and became a "complete" programming language that could be used to achieve any output that other languages could produce. One such use of Javascript is as a server-side scripting language (e.g. Node.js, Netscape Enterprise Server).

b. Event Driven Programming

The "traditional" mode of web servers has always been one of the thread-based model. You launch Apache or any other web server and it starts receiving connections. When it receives a connection, it holds that connection open until it has performed the request for the page or whatever other transaction was sent. If it takes a few microseconds to retrieve a page from disk or write results to a database, the web server is blocking on that input/output operation. (This is referred to as "blocking I/O".) To scale this type of web server, additional copies of the server need to be launched (referred to as "thread-based" because each copy typically requires another operating system thread).

In contrast, Node.js uses an event-driven model where the web server accepts the request, spins it off to be handled, and then goes on to service the next web request. When the original request is completed, it gets back in the processing queue and when it reaches the front of the queue the results are sent back (or whatever the next action is). This model is highly efficient and scalable because the web server is basically always accepting requests because it's not waiting for any read or write operations. (This is referred to as "non-blocking I/O" or "event-driven I/O".)

c. Callbacks

The Node library consists of numerous asynchronous functions. At least one argument of these functions is another function called the "callback function".
For example "function foo(a,b, bar(){}); Here the function foo() takes two parameters a,b and on completion executes callback bar(). Such asynchronous functions run in the background and do not halt the execution of the program making it non-blocking. The Node runtime environment contains an Event Loop. Once the execution of a program is finished the execution shifts to the Event Loop where, Node waits for an event to occur. When an event occurs the associated callback function is executed. Thus Event Driven I/O is achieved through use of callbacks.

d. Modules

The Node library contains several modules which are Javascript codes providing API to the user. This makes building web applications in Node easier as the user does not have to start from scratch. Many modules come built-in with node but the user is not restricted to the use of only these modules as, a vast repository of user created external modules is also available if required. Node implements CommonJS Modules 1.0 which specifies the standard of all

node modules. Some important modules used for implementation of this project:

e. HTTP:

This module is used to implement a http server.
The "CreateServer" function takes a callback which is executed on the request event. The callback is passed two objects - request and response. The request object details the request made to the server and the response object is used to send the appropriate data to the client.

f. FS:

It stands for File System. This module contains API to read/write files, read directories, create/delete directories, rename files etc. Most of the functions have both synchronous and asynchronous versions. Synchronous versions are "blocking" and should be avoided if the operation will be time consuming.

g. Formidable

External module for handling file uploads. File is uploaded from an html page as a POST data multipart form. The data is parsed together to obtain the file which is saved to the disk. The upload directory and other such parameters can be set by the user.

C. User Interface

Provided the server is running, on accessing the Domain Name associated with the server through a browser, the user is taken to the file system of the Raspberry Pi. Folders are denoted by a [+] sign to the left of their name and files by [-] sign. All the folders present in the directory are shown one below the other with their date and time of creation or latest modification shown in the adjacent column. If there is a file, in addition to the date, the size of the file in bytes is also displayed in the third column. Clicking on a folder takes the user to that particular directory. To guide the user while browsing, the path of the current directory is also shown on the top of each page. The 'Back' button of the browser can be used to go to the previous directories. A 'Go Back' link is also provided on each page for the same purpose. To download a file from the server, the user simply has to click on the filename. He will be given options to either open the file directly or save it on his computer. For example, if a rar file is opened, only the extracted contents will be saved on the computer. To upload a file to the server, an option to browse files on his computer and upload a file is provided on each page. The user must first go to the destination directory and then use the 'Browse' button to select a file from his computer. On selecting a particular file and clicking 'Upload' button, the file will be uploaded to that directory. Provided the operation is successful, the user will be shown the name of the file uploaded and the destination directory.

D. Approximate cost:

The prime advantage of this project is the low cost involved in setting up the server. The cost of the Raspberry Pi board is $35(Rs. 1925). The concerned accessories like the charger, an 8GB SD card can be estimated to collectively cost Rs. 500. Hence, the minimalistic cost of the entire system with approximately 6 GB storage is Rs. 2425. This cost is incurred only once and hence there are no recurring costs. Additional space, if needed, may be obtained by using a USB hub to connect as many hard drives or flash drives as needed. Alternatives like 'Dropbox', Sky Drive, iCloud and Google Drive cost more with their monthly or yearly billing cycles and also offer lesser options in terms of space. Shown below is a comparison of the pricing of various products offered for web based storage of 500 GB.

Product	Raspberry Pi	Sky Drive	Google Drive	Dropbox	iCloud
Pricing (INR)	5425 (including the cost of a 500GB HDD)	12500	15000	25000	50000

It should be noted that the expenditure mentioned for Raspberry Pi is one time whereas that for all others is annual.

E. Speed results:

Speed is an important criterion for a server. The Pi supports USB 2.0 which allows maximum transfer speeds of 60 Mbytes/sec. Also the Ethernet standard on the Pi uses 100 Mbits/sec. When Pi is connected in home network via Ethernet, transfer speeds of up to 8 Mbytes/sec are obtained. The Pi boots from SDcard. The minimum transfer speeds from SDcard to processor varies from 2 Mbytes/sec to 10 Mbytes/sec.

F. Factors affecting speed:

Though the maximum USB transfer speed is 60 Mbytes/sec, this speed is never achieved due to factors like NAND limitations on USB flash drives and so the maximum speeds achieved range between 30 – 35 Mbytes/sec whereas external HD and other magnetic/optical storage lower the maximum transfer speeds to 25 Mbytes/sec. These are serial access speeds. For better random access speeds, external HD with rootfs file system on it is preferred over flash drives. Also the Class of flash drive or SDcard being used contributes to the transfer speed. Class 2, 4, 6 and 10 have their minimum transfer speeds of 2, 4, 6 and 10 Mbytes/sec when the SDcard or flash drive is empty. The speed varies depending on the amount of free space left and also on the file system being used for that drive. The file systems used for USB flash drives i.e. NTFS, exFAT are faster and efficient than ext1, ext2, ext3, ext4, FAT32 etc used for

SDcard. For optical/magnetic devices like external HD, "x" rating is used to indicate transfer speed like x100, x125, x150. This rating is multiplied with standard CDRom drive speed of 150 Kbytes/sec to obtain the transfer speed. Higher the rating better is the speed. So, depending on the speed and capacity requirement, an External HD or USB flash drive can be chosen. When an user outside the home network tries to upload or download a file from the Pi, the Pi provides 100 Mbits/sec transfer speed but the bandwidth is limited depending on the user's ISP. Smaller the bandwidth lesser is the transfer speed of the user.

.

III. Conclusion

The purpose of designing and implementing a low cost networked storage is achieved by using ARM-11 based Raspberry Pi board running the Linux based Raspbian OS. The HTTP server is implemented with the use of Node.js. The system can be used to store, browse through and retrieve data from the server to any remote location through the web. The prime advantage of the system is low and non-recurring cost for the setting up of the system. Also storage can be added and removed as and when required. This design can be widely used for personal as well as industrial applications.

Acknowledgment

All the authors would like to thank Dr. R.D. Daruwala for the valuable guidance throughout the implementation of the project.

References

[1] HTTP://NODEJS.ORG/

[2] HTTP://NODEJS.ORG/ABOUT/

[3] HTTP://WWW.NODEBEGINNER.ORG/

[4] HTTP://EN.WIKIPEDIA.ORG/WIKI/NODEJS

[5] HTTP://CODE.DANYORK.COM/2011/01/25/NODE-JS-DOCTORS-OFFICES-AND-FAST-FOOD-RESTAURANTS-UNDERSTANDING-EVENT-DRIVEN-PROGRAMMING/

[6] HTTP://NODEJS.ORG/API/

[7] HTTP://EN.WIKIPEDIA.ORG/WIKI/JAVASCRIPT

[8] HTTPS://GITHUB.COM/FELIXGE/NODE-FORMIDABLE

[9] HTTP://NODEJS.ORG/API/FS.HTML

[10] HTTP://NODEJS.ORG/API/HTTP.HTML

[11] http://www.raspberrypi.org/faqs

[12] https://support.google.com/drive/bin/answer.py?hl=en&answer=2375123

[13] http://windows.microsoft.com/en-us/skydrive/compare

[14] https://www.dropbox.com/upgrade

[15] http://support.apple.com/kb/ht4874

[16] HTTP://WWW.RASPBIAN.ORG/RASPBIANABOUT

[17] http://www.raspberrypi.org/phpBB3/viewtopic.php?f=46&t=32454

[18] http://en.wikipedia.org/wiki/Secure_Digital

[19] http://en.wikipedia.org/wiki/Flash_memory

[20] http://www.tomshardware.com/forum/248184-32-best-drives