Remote Administrative Suite for Unix-Based Servers

G.Rama Koteswara Rao
Professor, Department of Computer Science, Post-Graduate Centre, P.B.Siddhartha College, Vijayawada – 520 010
Email: koti_g @yahoo.com
G.Siva Nageswara Rao
Reader, Department of Computer Science, Post-Graduate Centre, P.B.Siddhartha College, Vijayawada – 520 010
Email: sivanags@india.com
B.V.Subba Rao
Associate Professor, Department of IT, PVP Siddhartha Institute of Technology, Vijayawada
Email: bvsrau@gmail.com

---ABSTRACT---
This paper deals with the methodologies that help in enhancing the capabilities of the server. An attempt is made to develop software that eases the burden of routine administrative functions. This results in increasing the overall throughput of the server.

Keywords: Server, throughput

Date of Submission: 05, December 2009 Revised: 22, April 2010 Date of Acceptance: 15, May 2010

1.INTRODUCTION

In this paper, we deal with client-server technology. We develop methods to enhance the capabilities of a client in accessing a server on static and dynamic administrative services. Generally, a server administrator has the privilege of capturing everything that is happening on the server side.

This paper discusses two processes, running one at server and another at selected client. The client side process sends an IP packet, with a request for desired service. The process running on the server side acts like a gateway and examines the incoming packet. This “gateway process” processes the request.

2. CLIENT SIDE SOFTWARE

Features that incorporated in developing client side software include the following among several others.

- User and Group Management
- Remote Script Execution with Feedback
- File System Monitoring
- Monitoring Paging and Swap Space
- Monitoring System Load
- Process Management
- File Locking
- Device Drivers
- Database Administration

3..ROLES OF CLIENTS

A main feature of the client is to give a convenient User interface, hiding the details of how the server 'talks' to the user. The client needs to first establish a connection with the server, given its address. After the connection is established, the client needs to be able to do two things

1. Receive commands from the user, translate them to the server's language (protocol) and send them to the server.
2. Receive messages from the server, translate them into human-readable form, and show them to the user. Some of the messages will be dealt-with by the client automatically, and hidden from the user. This is based on the Client designer's choice.

4. ALGORITHM DEVELOPED FOR CLIENT SIDE SOFTWARE FUNCTIONS:

```
1.1. get the server's address from a working address that
     can be used to talk over the Internet.
1.2 connect to the server
1.3 while (not finished) do:
1.3.1wait until there is information either from the server,
   or from the user.
1.3.2 If (information from server) do
1.3.2.1 parse information, show to user, update local state
   information, etc.
1.3.3 else {we've got a user command}
1.3.3.1 parse command, send to server, or deal with locally.
1.4 done
```

5. ROLES OF SERVERS

A server’s main feature is to accept requests from clients, handle them, and send the results back to the clients. The Server side process checks the 8-bit unused field of IP packet to confirm that the request is from a valid client. We discuss two kinds of servers: a single-client server, and a multi-client server.

5.1 Single Client Servers

Single client server responds only to one client at a given time. It acts as follows:

1. Accept connection requests from a Client.

2. Receive requests from the Client and return results.

3. Close the connection when done, or clear it if it's broken from some reason.

Following is the basic algorithm a Single-Client Server performs:

```
1.1 bind a port on the computer, so Clients will be able to connect
1.2.      forever do:
1.2.1.    listen on the port for connection requests.
1.2.2.    accept an incoming connection request
1.2.3.    if (this is an authorized Client)
1.2.3.    1while (connection still alive) do:
1.2.3.    2receive request from client
1.2.3.3   handle request
1.2.3.4   send results of request, or error messages.
1.2.3.5   done
1.2.4     else
1.2.4.1   abort the connection
1.3       done
```

5.2 Multi Client Servers

Multi-Client server responds to several clients at a given time. It acts as follows:

1. Accept new connection requests from Clients.

2. Receive requests from any Client and return results.

3. Close any connection that the client wants to end.

Following is the basic algorithm a Multi-Client Server performs:

```
1.1       bind a port on the computer, so Clients will be able to connect
1.2       listen on the port for connection requests.
1.3       forever do:
1.3.1     wait for either new connection requests, or requests from existing Clients.
1.3.2     if (this is a new connection request)
1.3.2.1   accept connection
1.3.2.2    if (this is an un-authorized Client)
1.3.2.2.1     close the connection
1.3.2.3   else if (this is a connection close request)
1.3.2.3.1     close the connection
1.3.2.4   end if
1.3.3         end if
1.3.4         else { this is a request from an existing Client
              connection}
1.3.4.1       receive request from client
1.3.4.2   handle request
1.3.4.3   send results of request, or error messages
1.3.5         end if
1.4           done
```

6. File System Monitoring

Monitoring complete file systems is the most common monitoring task. On different flavors of Unix the monitoring techniques are the same, but the commands and fields in the output vary slightly. This difference is due to the fact that command syntax and the output columns vary depending on the flavor of the Unix system being used.
We have developed software script for monitoring the file system usage.
The outcome of our software that is developed using several methods are as follows:

6.1 Percentage of used space method.
Example:
```
/dev/hda2 mounted on /boot  is   11%
```

6.2 Megabytes of free space method.
Example:
```
Full File System on pbscpg55046.pbscpg
/dev/hda3 mounted on / only as 9295 MB Free Space
/dev/hda2 mounted on /boot only as    79 MB Free Space
```

6.3 Combining percentage used 6.1 and megabytes of free space 6.2.
6.4 Enabling the combined script to execute on AIX, HP_UX, Linux and Solaris.

7. Monitoring Paging and Swap Space

Every Systems Administrator attaches more importance to paging and swap space because they are supposed to be the key parameters to fix a system that does not have enough memory. This misconception is thought to be true by many people, at various levels, in a lot of organizations. The fact is that if the system does not have enough real memory to run the applications, adding more paging and swap space is not going to help. Depending on the applications running on the system, swap space should start at least 1.5 times physical memory. Many high-performance applications require 4 to 6 times real memory so the actual amount of paging and swap space is variable, but 1.5 times is a good place to start.

A page fault happens when a memory segment, or page, is needed in memory but is not currently resident in memory. When a page fault occurs, the system attempts to load the needed data into memory, this is called paging or swapping, depending on the Unix system being used. When the system is doing a lot of paging in and out of memory, this activity needs monitoring. If the system runs out of paging space or is in a state of continuous swapping, such that as soon as a segment is paged out of memory it is immediately needed again, the system is thrashing. If this thrashing condition continues for very long, there is a possible risk of the system crashing. One of the goals of the developed software is to minimize the page faults.

Each of four Unix flavors, AIX, HP-UX, Linux, and Solaris, use different commands to list the swap space usage, the output for each command and OS varies also. The goal of this paper is to create all-in-one shell script that will run on any of our four Unix flavors. A sample output of the script is presented below.

```
Paging Space Report for GRKRAO
Thu Oct 25 14:48:16 EDT 2007
Total MB of Paging Space          :       33MB
Total MB of Paging Space Used     :        33MB
Total MB of Paging Space Free     :       303MB
Percent of Paging Space   Used  :       10%
Percent of Paging Space    Free  :       90%
```

8. Monitoring System Load

There are three basic things to look at when monitoring the load on the system.

1. First is to look at the load statistics produced.

2. Second one is to look at the percentages of CPU usage for system/kernel, user/applications, I/O wait state and idle time.

3. The final step in monitoring the CPU load to find hogs. Most systems have a top like monitoring tool that shows the CPUs, processes, users in descending order of CPU usage.

9. File Locking

File locking allows multiple programs to cooperate in their access to data. This paper looks at the following two schemes of file locking.

1. A simple binary semaphore scheme
2. A more complex file locking scheme of locking different parts of a file for either shared or exclusive access

10. Device Drivers

Device Drivers are needed to control any peripherals connected to a server. This paper focuses on the following aspects of device drivers where an authorized client can control devices connected to the server.

1. Registering the device
2. Reading from a device and Writing to a device
3. Getting memory in device driver

11. Database Administration

C" Language is used to access MySQL. In this paper, the following database administrative features are implemented to be run at an authorized client

1. Create a new database
2. Delete a database
3. Change a password
4. Reload the grant tables that control permissions
5. Provide the status of the database server
6. Repair any data tables
7. Create users with permissions

12. Using the algorithm that is described in serial number 4 above, we developed the following C program code:

<u>Sample Client Program</u>

```
#include <sys/socket.h>
#include<netinet/in.h>
#include<arpa/inet.h>
#include<stdio.h>

int main(int argc,char **argv)
{
       int sockfd,n,len;
       char buf[10240];
       struct sockaddr_in servaddr;
       if(argc!=2)       perror("invalid IP");
       if((sockfd=socket(AF_INET,SOCK_STREAM,0))<0)
perror("socket error");
       bzero(&servaddr,sizeof(servaddr));
       servaddr.sin_family = AF_INET;
       servaddr.sin_port = htons(13);
       if(inet_pton(AF_INET,argv[1],&servaddr.sin_addr)
        <= 0)     perror("SERVER ADDR ");
       if(connect(sockfd,(struct
sockaddr*)&servaddr,sizeof(servaddr))<0)
perror("connect error");

   buf[0] = '\0';
   printf("Enter the Directory name \n");
             scanf("%s",buf);
    if(write(sockfd,buf,100) < 0)      {printf("write error ");
   exit(1);  }
              if((len = read(sockfd,buf,100)) < 0) {
   printf("read error \n");   exit(1);         }
              else {   printf(" Inode Number  = %s\n", buf);
   }
   if((len = read(sockfd,buf,100)) < 0) {  printf("read error
   \n");  exit(1);       }
             else {   printf(" No of links  = %s\n", buf);      }
if((len = read(sockfd,buf,100)) < 0)    {    printf("read error
\n");   exit(1);    }
             else  {    printf("Size of file in bytes = %s\n",
buf);                              }
if((len = read(sockfd,buf,100)) < 0)    {    printf("read error
\n");     exit(1);         }
             else {   printf("UID   = %s\n", buf);    }
if((len = read(sockfd,buf,100)) < 0)  {     printf("read error
\n");      exit(1);    }
  else    {         printf("GID   = %s\n", buf);       }
if((len = read(sockfd,buf,100)) < 0)    {
printf("read error \n");                exit(1);      }
  else    {      printf("Type and Permissions   = %s\n", buf);
}
if((len = read(sockfd,buf,100)) < 0)      {         printf("read
error \n");                exit(1);     }
  else {    printf("Last Modification Time   = %s\n", buf); }
if((len = read(sockfd,buf,100)) < 0)  {       printf("read
error \n");         exit(1);    }
   else   {     printf("Last Access Time   = %s\n", buf);       }
   exit(1);
}
```

13. Using the algorithm that is described in serial number 5 above, we developed the following C program code :

<u>Sample Server Program</u>

```
#include<sys/socket.h>
#include<arpa/inet.h>
#include<stdio.h>
```

```
#define MAXLINE 10024
#define LISTENQ 10

int main(int argc,char **argv)
{
   int listenfd,connfd,len,i;
      struct sockaddr_in servaddr;
   struct stat statbuf;
      char buff[MAXLINE],buff1[MAXLINE];
   DIR *dir;
   struct dirent *direntry;
      listenfd = socket(AF_INET,SOCK_STREAM,0);
      bzero(&servaddr,sizeof(servaddr));
      servaddr.sin_family = AF_INET;
      servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
      servaddr.sin_port = htons(13);
      bind(listenfd,(struct
sockaddr*)&servaddr,sizeof(servaddr));
      listen(listenfd,LISTENQ);
   connfd = accept(listenfd,(struct
sockaddr*)NULL,NULL);
   if((len = read(connfd,buff,100)) < 0)       {
printf("read error \n");    exit(1);           }
else printf("%s\n",buff);
    lstat(buff,&statbuf);
   sprintf(buff, "%d", statbuf.st_ino);
            if(write(connfd,buff,100) < 0)   {    printf("write
error ");      exit(1);       }
   sprintf(buff, "%d", statbuf.st_nlink);
            if(write(connfd,buff,100) < 0)    {    printf("write
error ");      exit(1);       }
   sprintf(buff, "%d", statbuf.st_size);
 if(write(connfd,buff,100) < 0)  {      printf("write error ");
exit(1);  }
sprintf(buff,"%d", statbuf.st_uid);
if(write(connfd,buff,100) < 0)   {      printf("write error ");
exit(1);   }
sprintf(buff,"%d", statbuf.st_gid);
 if(write(connfd,buff,100) < 0)  {      printf("write error ");
exit(1);  }
sprintf(buff,"%o", statbuf.st_mode);
if(write(connfd,buff,100) < 0)  {         printf("write error ");
exit(1);  }
sprintf(buff,"%s", ctime(&statbuf.st_mtime));
 if(write(connfd,buff,100) < 0) {      printf("write error ");
exit(1);   }
sprintf(buff,"%s", ctime(&statbuf.st_atime));
  if(write(connfd,buff,100) < 0) {     printf("write error ");
exit(1);  }
   close(connfd);
   exit(1);
}
```

Sample Outputs

```
[root@grkrao 01Oct]# ./a.out
Message From Client : /etc/passwd
[grkrao@grkraoclient 01Oct]# ./a.out 123.0.57.44
Enter the File  name : /etc/passwd
Message From Server :
 Inode Number  = 1798355
 No of links  = 1
Size of file in bytes = 3263
UID   = 0
GID   = 0
Type and Permissions   = 100644
Last Modification Time   = Fri Apr 20 16:12:06 2007
Last Access Time   = Tue Apr 24 11:46:33 2007
```

14. CONCLUSION

This paper attempted to enhance the capabilities of the server by providing software tools that reduce the burden of routine administrative functions of the administrator.

Through the use of the developed software, it can be concluded that Remote Administrative Suite for Unix Based Servers would prove very useful to administrators and such use would result in increasing the overall throughput of the server.

REFERENCES

[1] W.Richard Stevens, *Advanced Programming in Unix Environment*, Pearson Education, 91-136

[2] W.Richard Stevens, *Unix N/W Programming – Vol-1: Networking APIs: Socket and XTI*, Pearson Education, 3-140

[3] Uresh Vahalia, *Unix Internals: New Frontiers,* Pearson Education, 43-50

[4] W.Richard Stevens, *Unix Network Programming*, PHI, 258-3

[5] Eric S.Raymond, *The Art of Unix Programming*, Addison-Wesley Professional Computing Series.

[6] W.Richard Steven, *Unix Network Programming: Interprocess Communication,* Pearson Education.

[7] William Stalling, *Operating Systems, Internals & Design Principles*, Pearson Education.

[8] Terrence Chan, *Unix System Programming Using C++*, PHI

[9] AEleen Frisch, *Essential System Administration*, O'Reilly.

[10] Christian Benvenuti, *Understanding Linux Network Internals,* O'Reilly.

[11] Frank Mayer,Karl MacMillan, David Caplan, *SeLinux By Example: Using Security Enhanced Linux*, Prentice Hall.

[12] Robert Love, *Linux Kernel Development*, Sams Publishing

[13] Roderick W.Smith, *Linux Power Tools*, Sybex.

[14] Christopher Negus & Thomas Weeks, *Linux Trouble Shooting Bible*, John Wiley & Sons.

[15] Richard L.Petersen, *Redhat : The Complete Reference Enterprise Linux & Fedora Edition : The Complete Reference*,McGraw-Hill.

Authors Biography

Mr.Rama Koteswara Rao.*G* presently working as a Professor in P.B. Siddhartha College Vijayawada, affiliated to Acharya Nagarjuna University, Guntur. He received his M.Tech degree in Computer Science and Engineering from MGR University, Chennai. He is pursuing Ph.D in Computer Science and Engineering at Vinayaka Missions University, Chennai. He presented 3 papers in National /International Conference Proceedings. He is a member and secretary of Computer Society of India (CSI), Vijayawada Chapter. His current research interests are in the areas of Client Server Architecture, Operating Systems.

Mr.G.Siva Nageswara Rao presently working as a Reader in P.B. Siddhartha College Vijayawada, affiliated to Acharya Nagarjuna University, Guntur.. He pursued his M.Tech Acharya Nagarjuna University, Guntur. He has a total of 12 years of rich experience comprising teaching, research and industry. He has guided 65 Masters level projects. He is the life member of 3 professional bodies. His current research interests are in the areas of Data warehousing and Data mining and image processing.

B.V.Subba Rao, presently working as Associate Professor in P.V.P Siddhartha Institute of Technology Vijayawada, affiliated to Jawaharlal Nehru Technological University. He received his M.Tech degree with distinction in Computer Science and Engineering from Acharya Nagarjuna University. He is pursuing Ph.D in Computer Science and Engineering at Acharya Nagarjuna University, Guntur. He received Gold Medal from Andhra University in his Post Graduate Studies. He has guided 30 post Graduated and 40 graduate projects. He has published 4 papers in International Journals and 3 papers in national Journals and presented 4 papers in National /International Conference Proceedings. He has Academic participation in 24 International / National Seminars / workshops and Conferences. He is a member of Computer Society of India (CSI), Association for Computing Machinery (ACM), and Indian Society for Technical Education (ISTE). His current research interests are in the areas of Artificial Intelligence, Natural Language Processing and Information Retrieval systems.