A New Grid Server

Kenji Arisawa
Aichi University, Nagoya, Japan
arisawa@aichi-u.ac.jp

Abstract

A new type of data server is proposed[1]. The server is designed for grid computing. The distinctive feature of the server is: that enables to execute programs of clients without allowing any byte to be written to the server. Therefore we need not allocate storage space for clients, which means the time and labor will be reduced greatly, and in addition, we can keep the server perfectly clean.

1 Introduction

Plan 9 is designed as a distributed operating system[2]. Therefore, the OS has many nice features for grid computing as shown in the paper by Mirtchovski et al.[2] and also with more detailed descriptions in the thesis by Mirtchovski[3]. Stimulated by these works, people in 9fans[3] devoted themselves to grid computing around the year 2005. From that exercise, they got some fruitful results and ideas to the future. These experiences are summarized on Bell-labs website[4]. However, the activity had ended up without making nice pieces of ideas into reality.

Recently Big Data is talked about. By "Big Data" I mean those data that is inappropriate to be transferred across network. We should note that programs are much smaller then data. Therefore, the data should be processed at the server side by transferring programs to the server. Then we need remote login. In such case, security does matter.

The server I proposed in this paper is a magical server for those who don't know Plan 9. Consider the requirement:

> *Clients need to execute their programs in the server, on the other hand, the server does not want any byte to be written into the server's storage.*

Is it possible to construct such a server that allows execution of the client programs without allowing any byte to be written? Yes possible! If we have such a server, we need not allocate user's storage space in the server side, which means the time and labor are reduced greatly. In addition, we can perfectly keep the server clean.

Remote execution command `ssh` is a common tool today in Unix world. Instead, Plan 9 uses `cpu` command for remote execution[4]. The command is different from `ssh` (or `telnet`) in that it is not only for command execution but also it mounts local file system to the remote side on the fly. The mount point is `/mnt/term` in the server's name space. The trick is in multiplexed communication channel between client and server. Files in local side are visible in remote side. This means we no longer need traditional tools such as `ftp` nor `scp` to transfer files from locale side to remote side, and furthermore, we need not edit files in remote side. They are editable in local side and the effect is immediately reflected on the remote side.

Grid computings today have been developed in Unix world and mainly based on the softwares from Globus[5]. They need distributed accounting with distributed file system so that users are permitted to login and so that storage space is allocated to the users for locating their programs, which however needs high level collaboration. The mechanism proposed here makes it

[1] This paper is based on the auther's webpage[1] and is rewritten for those who are unfamiliar to Plan 9, discarding some contents that are not essential to the subject matter.

[2] In this paper "Plan 9 from Bell Labs" is referred to as "Plan 9". Plan 9 is originally developed by the people in Computing Sciences Research Center at Bell Labs, the same research group that developed Unix operating system. Plan 9 has many innovative features and is now available under the GNU Public License. There was many documents on Plan 9 on the website `plan9.bell-labs.com`. Unfortunately the site is now closed after the Labs belongs to Nokia. These documents are still kept on some mirror sites in the Internet. It is recommended to read the concept outlined by original authors [7].

[3] Name of mailing list `9fans@9fans.net`.

[4] Look the reference [9] for the command.

needless to allocated users space, which means: grid systems may become greatly simplified[5]. The server is designed dreaming to be a base model for future grid computing that enables collaboration among wider range of people, and also that enables perfect cleanness to keep the computing service. The server is constructed on those nice pieces of ideas discussed in Plan 9 users group[4].

2 Login

In entering the grid server, you need to be registered as a user of `plan9.bell-labs.com`[6]. If you are registered as a user, then execute:

```
cpu -h grid.nyx.link -k 'dom=outside.plan9.bell-labs.com'
```

with `factotum`[7] key

```
key dom=outside.plan9.bell-labs.com proto=p9sk1 user=XXXXX
 !password=YYYYY
```

where `XXXXX` is your ID and `YYYYY` is your password. Attribute/value pairs that follow `key` must be in a single line[8]. If you succeed in login, you will see the prompt "`grid%`".

3 The Grid Server

Try first:

```
ps
```

then you will find some strings

```
XXXXX@outside.plan9.bell-labs.com
```

in the output, where `XXXXX` is your ID on the domain `outside.plan9.bell-labs.com`. The list below is the example.

```
grid% ps
arisawa           1    0:00   0:00      256K Await    bootrc
arisawa           2    0:00   0:00        0K Wakeme   mouse
...
none            369    0:00   0:00      132K Open     listen
none            370    0:00   0:00      132K Open     listen
arisawa@outside.plan9.bell-    20188    0:00   0:00      124K Await    gcpu
arisawa@outside.plan9.bell-    20195    0:00   0:00      240K Await    rc
arisawa@outside.plan9.bell-    20196    0:00   0:00      124K Pread    gcpu
arisawa@outside.plan9.bell-    20247    0:00   0:00      116K Pread    ramfs
arisawa@outside.plan9.bell-    20252    0:00   0:00       92K Pread    ps
grid%
```

The `XXXXX@outside.plan9.bell-labs.com` is a process owner's ID in the grid server. The ID is not registered to file system of the server. Then Plan 9 allows the process to use the file system as user name `none`. What if another user, say, `YYYYY@outside.plan9.bell-labs.com` is logging then? Both processes are playing as user `none` to the file system. If those grid user are allowed writing to the file server, they will interfere. However you need not worry about because writing to the shared file system is disallowed. Writing to private file system is allowed. One example is ramfs[9] mounted on `/tmp` (and bound to `/usr/none/tmp`). The disk is automatically provided to grid users for temporal use and automatically disappears as the user logged out. Another example is client file system mounted on `/mnt/term`. The grid user's process can accesses the client file system as if the user operates in local side.

Try second:

```
ls /usr
```

then the command shows the list of home directories of users: `/usr/none`, `/usr/arisawa` and in addition, some other directories, say, `/usr/glenda` and etc. In the list, `/usr/none` and `/usr/arisawa` are directories of the server[10]. On the other hand, `/usr/glenda` and etc are directories of the client, which are produced by the command

```
bind -a /mnt/term/usr /usr
```

where `bind` is one of commands that is used in configuring name space[10]. Plan 9 name space can be configured very flexibly. The name space under `/usr` is private to the client and is hidden to other users; and note: the name space that is visible by grid users is only a small portion of system name space[11].

Try third:

```
acme
```

You can browse files on the grid server using Plan 9 text editor `acme`[8]. The editor supports mouse-operation and multi-windows. Regular users can run commands in the window of `acme`. However grid users are disabled this functionality[12]. They need to run commands outside of `acme`. Any user has privilege to access his local files. Hence you can edit your files using `acme` on grid server. (And of course also on your local machine.) In processing data in grid server, you probably need your own programs. Except a few operations, the grid server allows executing your programs even if that are binary executables compiled on the local side. Your commands are in `/mnt/term/bin` or somewhere else under `/mnt/term`[13]. If you want to save something, you can write it to your own storage through `/mnt/term/usr`.

[5] In Plan 9, permitting to enter the server does not mean giving an account on the file system. The proposed grid server needs only authentication for a user to enter the server so that process owner on the server are guaranteed to be unique.

[6] If you don't have a user account of outside.plan9.bell-labs.com, please email to me. I can offer you a user account to my grid server. It seems that new bell-labs account is now closed.

[7] Name of authentication proxy for Plan 9. Look the reference [6].

[8] Note that `factotum` accepts ticket from one or more authentication domains.

[9] Look man page RAMFS(4) [11].

[10] It is not nice to expose `/usr/arisawa`. The directory is required for some special services.

[11] When you enter the grid server, look `/usr/none/lib/profile` for the name space configuration and look `/usr/arisawa/src/grid` for the grid patches.

[12] This inconvenience comes from: grid users are disallowed mount operation.

[13] If object code type is different between client and server, the situation is somewhat complicated. Then we need to have the executable for the server.

4 Security

In accessing grid servers, we use cpu command. Then your processes on the grid server have ability to access your local machine. This means you have a potential security risk when you are compromised on the server side[14]. Therefore it is safe to export only a portion of namespace of your file system to grid server. Plan 9 cpu command has -P option for this purpose. However, unfortunately, this option does not work well. Another way is to construct minimum namespace to export in execution cpu command. To do this, create the following files in somewhere, for example in /usr/none/lib.

```
term% pwd
/usr/none/lib
term% lr -l grid
d-rwxrwxr-x arisawa  sys      0 2015/12/24 13:29:26 grid
d-rwxrwxr-x arisawa  sys      0 2015/12/24 13:26:44 grid/ns
d-rwxrwxr-x arisawa  sys      0 2015/12/16 23:05:16 grid/ns/bin
d-rwxrwxr-x arisawa  sys      0 2015/12/24 12:54:23 grid/ns/dev
--rw-rw-r-- arisawa  sys      0 2015/12/24 12:54:11 grid/ns/dev/cons
--rw-rw-r-- arisawa  sys      0 2015/12/24 12:54:11 grid/ns/dev/consctl
d-rwxrwxr-x arisawa  sys      0 2015/12/24 12:54:23 grid/ns/dev/draw
--rw-rw-r-- arisawa  sys      0 2015/12/24 12:54:11 grid/ns/dev/random
d-rwxrwxr-x arisawa  sys      0 2015/12/25 09:13:14 grid/ns/env
d-rwxrwxr-x arisawa  sys      0 2015/12/24 02:07:26 grid/ns/mnt
d-rwxrwxr-x arisawa  sys      0 2015/12/17 05:41:17 grid/ns/mnt/factotum
--rw-rw-r-- arisawa  sys      0 2015/12/17 05:32:11 grid/ns/mnt/factotum/ctl
--rw-rw-r-- arisawa  sys      0 2015/12/17 05:41:17 grid/ns/mnt/factotum/log
d-rwxrwxr-x arisawa  sys      0 2015/12/24 02:07:26 grid/ns/mnt/wsys
d-rwxrwxr-x arisawa  sys      0 2015/12/16 23:15:39 grid/ns/net
d-rwxrwxr-x arisawa  sys      0 2015/12/24 12:52:41 grid/ns/proc
d-rwxrwxr-x arisawa  sys      0 2015/12/25 09:25:52 grid/ns/usr
d-rwxrwxr-x arisawa  sys      0 2015/12/25 09:25:52 grid/ns/usr/glenda
d-rwxrwxr-x arisawa  sys      0 2015/12/25 09:25:52 grid/ns/usr/none
--rw-rw-r-- arisawa  sys     89 2015/12/24 13:29:45 grid/patt
term%
```

In this list, /usr/glenda is assumed to be exported, and grid/patt is a pattern file for cpu command. The content is:

```
- /mnt/factotum
- /mnt/wsys/(.*/)?(text|screen|window)
```

And have a new cpu command, for example sgcpu (safe guard cpu) with the contents:

```
#!/bin/rc
rfork ne
cd /usr/none/lib/grid
for(f in cons consctl draw random)
bind /dev/$f ns/dev/$f
for(f in bin net proc)
bind /$f ns/$f
bind -c '#e' ns/env
bind -a /mnt/factotum ns/mnt/factotum
bind /mnt/wsys ns/mnt/wsys
bind -a /usr/none ns/usr/none
bind -a /usr/glenda ns/usr/glenda  ## example
bind ns /
cd /usr/none
/bin/cpu -P lib/grid/patt $*
```

Finally, using this new command, execute:

```
sgcpu -h grid.nyx.link -k 'dom=outside.plan9.bell-labs.com'
```

[14]The problem happens if the server itself is a honeypot.

5 Disabled Operations

The following operations are disabled:

- networking
- writing to permanent storage.
- looking private information.

There was an argument in 9fans whether networking should be disabled or not. In some cases, it would be better to allow networking so that the grid users can import external data to the server. Then, however, we have risk that the server is used for undesirable purpose. The proposed grid server disables networking. Even if a user needs external data on the grid server, he can do as follows:
import the data to the client and then execute cpu command.

6 Technical Notes

Host owner[15] and user none are special users in Plan 9. Other users are devided in two categories in this grid server:

- regular users
- grid users

Host owner can do everything. User none is for network services. Grid users are restricted in small subset of namespace. For any users, mount/unmount operation is disabled after "rfork m"[16] (by kernel patch).

The following operations are disabled for regular users:

- networking (by kernel patch)
- becoming user none (by kernel patch)

The following operations are disabled for grid users:

- networking (by kernel patch)
- becoming user none (by kernel patch)
- mounting/unmounting (by "rfork m")
- writing to permanent storage. (by Plan 9 commands)

Ramfs is provided for grid users. Any user (including grid user) can read/write files in client side.

Known problem with process creation is fixed by kernel patch as follows:

- new process creation by kernel, host owner and user none are kept unchanged.
- others may fail in rfork().

This strategy is not perfect, but working reasonably. We need more work to make more robust kernel.

[15]The user who started the server.

[16]The command "rfork m" is used to disallow mount or bind operation. However unmount is allowed, which makes encapsulations of some sort of applications difficult. This problem is fixed in the grid kernel.

References

[1] Kenji Arisawa, “A New Grid Server”
http://plan9.aichi-u.ac.jp/9grid2/9grid.html

[2] Andrey Mirtchovski, Rob Simmonds and Ron Minnich, “Plan 9 an Integrated Approach to Grid Computing”
Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International

[3] Andrey A. Mirtchovski, “Grid Computing with Plan 9 an Alternative Solution for Grid Computing”
http://mirtchovski.com/p9/thesis.pdf (2005)

[4] Plan 9 Wiki, “9grid”
http://p9.nyx.link/wiki/9grid/

[5] Globus, “Research data management simplified”
https://www.globus.org/

[6] Russ Cox, Eric Grosse, Rob Pike, Dave Presotto and Sean Quinlan, “Security in Plan 9”
Proceedings of the 11th USENIX Security Symposium, 2002
https://www.usenix.org/legacy/event/sec02/cox/cox.pdf

[7] Rob Pike, Dave Presotto, Sean Dorward, Bob Flandrena, Ken Thompson, Howard Trickey and Phil Winterbottom, “Plan 9 from Bell Labs”
Plan 9 Programmer’s Manual, Volume 2, 1995
http://doc.cat-v.org/plan_9/4th_edition/papers/9

[8] Rob Pike, “Acme: A User Interface for Programmers”
Plan 9 Programmer’s Manual, Volume 2, 1995
http://doc.cat-v.org/plan_9/4th_edition/papers/acme/

[9] Plan9, “CPU(1)”
Plan 9 Programmer’s Manual, Volume 1, 1995
http://man.cat-v.org/plan_9/1/cpu

[10] Plan9, “BIND(1)”
Plan 9 Programmer’s Manual, Volume 1, 1995
http://man.cat-v.org/plan_9/1/bind

[11] Plan9, “RAMFS(4)”
Plan 9 Programmer’s Manual, Volume 1, 1995
http://man.cat-v.org/plan_9/4/ramfs