Virtual Private Grid

A Command Shell for Utilizing Hundreds of Computers
for version 1.0, 30 June 2003

Phoenix Project `phoenix-discuss@logos.t.u-tokyo.ac.jp`

This manual is for Virtual Private Grid(version 1.0, 30 June 2003), which is a command shell for utilizing hundreds of computers

Short Contents

Table of Contents

1 Organization of This Manual

In this manual, we briefly describe how to set up and use Virtual Private Grid (VPG). Chapter 2 explains platforms supported by this version of the implementation. Chapter 3 shows required tools for VPG. Chapter 4 explains the simplest way for installing VPG. It includes how to configure SSH. Chapter 5 shows the basic usage of VPG.

2 Supported Platforms

This version of VPG is intended to run on Unix-like systems. In particular, we have test the library on the following platforms:

- Linux on x86 processors.
- Solaris on SPARC processors.

VPG should also run on other systems, provided GNU systems (e.g., `gcc`, `make`) and POSIX threads are available.

3 Required Tools

VPG is distributed as source code that must be compiled before use. To compile the library, you must install some GNU tools: GNU C Compiler, GNU Make, and GNU Libtool.

If you use only basic features of VPG, you need not install any other tools. To use full features of VPG, however, you must install additional tools. More specifically, VPG supports communication over SSH. To enable this feature, you must install Secure Shell (SSH). Most implementations of SSH (e.g., OpenSSH) are acceptable.

Here is the list of required tools.

- GNU C Compiler, GNU Make, and GNU Libtool (`http://gcc.gnu.org/`)
- OpenSSH (`http://www.openssh.org/`)

4 Setting up

This chapter describes set-up necessary to use VPG. First, you must obtain the source code from a web server. Second, you need to compile and install VPG using the previously mentioned GNU tools. Third, you must configure the authentication method of SSH to enable SSH communication.

4.1 Obtaining the Distribution

The source code of VPG can be dowloaded from the project web page:

`http://wwww.yl.is.s.u-tokyo.ac.jp/~kaneda/vpg/`

4.2 Building and Installing VPG

VPG is compiled and installed using the `configure` script and a pair of `make` commands. The simplest way to install the library is:

1. extract the file.

   ```
   % tar xfzv vpg_20031114.tar.gz
   ```

2. compile the Pheonix library that VPG uses.

   ```
   % cd source/phoenix
   % ./configure
   % make
   ```

   The compile succeeds if the libary is built on `source/phoenix/build/`.

3. compile VPG.

   ```
   % cd source/vpg
   % ./configure
   % make
   ```

   The compile succeeds if command files `vpg` and `vsh` are built on `source/vpg/build/`.

4.3 Setting up SSH Communication

After finishing the compile of VPG, you can use most of the functions provided by VPG. As previously mentioned, however, there still remain some set-up to enable VPG to communicate with one another via SSH.

This facility is very useful especially when you would like to run your programs on machines spread over multiple LANs. This is because administrative restrictions (e.g., firewall) are often imposed on these machines and may penetrate only SSH connections between the different LANs, Before explaining how to set up SSH communication in detail, let us look at how it works. Imagine that you would like to run VPG on host *X* and *Y* and that *Y* is protected by a firewall. The allowed connections from *X* to *Y* is only SSH (port 22). In this situation, VPG enables a process on host *X* to communicate with another process on host *Y* as follows:

1. VPG on *X* executes `ssh`, initiates a SSH connection to *Y*, and forks `sock` process on host *Y*. `sock` is a simple Unix utility which reads and writes data across network connections. It is created when you compile VPG.
2. The forked `sock` process on *Y* establishes a TCP connection to the VPG running on the same host.
3. The message forwarding route between the two hosts has been established via `ssh` process and `sock` process. The processes on *X* and *Y* deliver messages to each other via this route.

Then, we describe set-up that is required for the above SSH communication. You must configure SSH and install `sock` on proper hosts.

- **Authentication with no user interaction.** VPG requires establishing SSH connections automatically without a user's typing password. In the above example, when host *X* wants to communicate with host *Y* vis SSH, Thus the runtime on *X* needs to establish a SSH connection to *Y* with no user interaction. The simplest way to accomplish this is to use public-key authentication with an empty pass-phrase.
- **Sock Installation.** `sock` must be installed on a remote host to which VPG initiates a SSH connection. In the above example, Sock must be installed on the host `Y`. Because `sock` is created on the directory `source/phoenix/build/$UNAME/sock`, the directory must be added to environmental variable `PATH` of the remote host $emphY.

5 Getting Started

To use VPG, you needs to install daemons on available machines with a manual configuration. In this chapter we describe how to write a configuration file, boot daemons and utilize machines with VPG.

5.1 Writing a Configuration File

Each daemon reads a configuration file when it boot. In the configuration file, you needs to write a list of available machines. More specifically, you gives a daemon X a configuration file that lists contact points to which the daemon X can initiate connections. In the configuration file, you specify one contact point in one line as follows:

```
tuba.is.s.u-tokyo.ac.jp 30000 kaneda
pythagoras.is.s.u-tokyo.ac.jp 30000 kaneda
133.11.23.16 30000 kanedacc
```

Each line consists of a hostname, a port number at which VPG listens, an account name of the host. Each item is separated with either `TAB` or `SPACE`.

Note that VPG can work even if the configuration file includes some hosts to which a daemon actually cannot connect.

5.2 Booting Daemons

You boot daemons on all machines that you want to use. You log in hosts where you want to run VPG, and execute the command `vpgd`, which is created on `source/vpg/build/$UNAME/`.

You can give the following options to `vpgd`:

- `-n <nickname>` This option is used to specify a nickname of the daemon. The alphabets that the nickname can use is `[a-z][A-Z][0-9]`. And the length of the nickname must be less than or equal to eight.

  If this option is not given, the nickname becomes equal to the hostname.
- `-c <filename>` This option specifies the location of the configuration file. When this option is not given,
  - the environmental variable `VPG_CONF_PATH` is used to specify the location of the configuration if the variable is set.
  - the daemon cannot connect to any other hosts if the environmental variable `VPG_CONF_PATH` is not set.
- `-p <port>` This option specifies the port number at which the daemon listens and accepts connections. When this option is not given, the daemon listens at port 30000.
- `-t` The daemon runs as a test mode. This option is used for debug.

5.3 Booting a shell

After booting daemons, you need to boot a shell. You need to execute `vsh`, which is also created on `source/vpg/build/$UNAME/`.

Note that you need to boot a daemon `vpgd` on the host where the shell `vsh` runs before the shell starts.

This shell is based on `zsh-4.0.4`. If you type `command@nickname` on this shell, `command` is executed on the host whose nickname is `nickname`.

If you type `vpg show`, the shell prints a list of available machines on this shell.