John the Ripper on a Ubuntu 10.04 MPI Cluster

Pétur Ingi Egilsson
petur [at] petur [.] eu

Table of Contents

Foreword

First of all I must state that I'm no expert on MPICH, this guide is written by an hobbyist.

The cluster implementation presented in this paper is not meant for environments where high availability and security is an issue.

History

I found myself in a situation where I had to break up an old password of mine. Having a couple of computers around I started looking for a way to get them all working together.

This paper is the result of extreme frustration, an overdose of coffee and a sleepless night.

Requirements

At least two networked computers running Ubuntu Linux version 10.04.

I used 802.11g (54Mbit WiFi), John does not require much bandwidth.

Configuring the Server

Requirements

A static IP address or a reserved IP in DHCP.

Required packages

The following packages are required:

- libmpich1.0-dev - mpich static libraries and development files
- libmpich-mpd1.0-dev - mpich static libraries and development files
- libmpich-shmem1.0-dev - mpich static libraries and development files

- mpich2 - Implementation of the MPI Message Passing Interface standard
- mpich2-doc - Documentation for MPICH2
- john - active password cracking tool
- openssh-server - secure shell (SSH) server, for secure access from remote machines
- build-essentials - Informational list of build-essential packages

```
petur@server:~$ sudo apt-get install libmpich1.0-dev libmpich-mpd1.0-dev libmpich-shmem1.0-dev mpich2 mpich2-doc john openssh-server build-essentials
```

Network configuration

By default the /etc/hosts file looks like this:

```
127.0.0.1       localhost
127.0.1.1       server.petur.eu server

# The following lines are desirable for IPv6 capable hosts
::1             localhost ip6-localhost ip6-loopback
fe00::0         ip6-localnet
ff00::0         ip6-mcastprefix
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters
```

You need to change the 127.0.1.1 to your IP address.
<server.petur.eu> should be your FQDN and <server> is your machines hostname.

Find your IP by executing

```
petur@server:~$ ifconfig|grep "inet addr"
          inet addr:10.0.0.1      Bcast:10.255.255.255          Mask:255.0.0.0
          inet addr: 127.0.0.1    Mask:255.0.0.0
```

/etc/hosts should look like this after you have changed it:

```
127.0.0.1       localhost
10.0.0.1        server.petur.eu server

# The following lines are desirable for IPv6 capable hosts
::1             localhost ip6-localhost ip6-loopback
fe00::0         ip6-localnet
ff00::0         ip6-mcastprefix
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters
```

User configuration

Create a new user called 'cluster' and add ~/bin/ to his path.

I find it most convenient to use the same pass for 'cluster' on every machine.

```
petur@server:~$ sudo useradd -m -s /bin/bash cluster
petur@server:~$ sudo passwd cluster
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
petur@server:~$ sudo su - cluster -c "mkdir ~/bin;export PATH=~/bin:$PATH"
```

Configuring MPICH

MPI makes use of the following configuration files:

- ~/.mpd.conf
  NOTE: The file begins with a .
  This file must be chmod 600
  The this file contains a single line “secretword=<password>” (replace <password> with your password, which must be the same in all ~/.mpd.conf in the cluster).
- ~/mpd.hosts
  Contains the list of all nodes in the cluster, including the server.
  The format of the file is “host:number-of-cpu-cores”, fx. 10.0.0.2:4 if 10.0.0.2 has 4 cores.
  The number of cores can be set lower than the actual core number if you only want MPICH to use a specific number of cores without regard to the number of available cores.
  For an example, if the server has a quad core processor you might want to set the value to :3 instead of :4 so the server can be used for something else.
  DO NOT use localhost or 127.0.0.1, you MUST use a network reachable IP.

Check the number of cores available, and create the configuration files.

```
cluster@server:~$ touch ~/.mpd.conf
cluster@server:~$ chmod 600 ~/.mpd.conf
cluster@server:~$ echo secretword=pass>~/.mpd.conf
cluster@server:~$ /sbin/ifconfig|grep "inet addr"
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
          inet addr:127.0.0.1  Mask:255.0.0.0
cluster@server:~$ cat /proc/cpuinfo|grep processor|wc -l
1
cluster@server:~$ echo 10.0.0.1:1>~/mpd.hosts
```

Check if everything is in order by executing the following commands:

- mpdboot – start the cluster
- mpdtrace - list all nodes in the cluster
- mpdallexit – shut down the cluster

```
cluster@server:~$ mpdboot
cluster@server:~$ mpdtrace
server
cluster@server:~$ mpdallexit
```

If the commands executed without any errors then your all set, if not then review the permissions on your ~/.mpd.conf, review your /etc/hosts and the contents of your ~/mpd.hosts

Installing John the Ripper

An MPI patched John the Ripper version can be found at www.bindshell.net/tools/johntheripper

```
cluster@server:~$ mkdir source
cluster@server:~$ cd source
cluster@server:~/source$ wget http://www.bindshell.net/tools/johntheripper/john-1.7.2-bp17-mpi8.tar.gz
```

Unpack it and run make from the src directory (use make|less if your resolution is too small)

```
cluster@server:~/source$ tar -zxf john-1.7.2-bp17-mpi8.tar.gz
cluster@server:~/source$ cd john-1.7.2-bp17-mpi8/src/
cluster@server:~/source/john-1.7.2-bp17-mpi8/src$ make
```

You will be presented with a list of options.

```
To build John the Ripper, type:
        make clean SYSTEM
where SYSTEM can be one of the following:
linux-x86-mmx           Linux, x86 with MMX
linux-x86-sse           Linux, x86 with SSE2 (best)
linux-x86-any           Linux, x86
linux-x86-64            Linux, AMD x86-64, 64-bit native w/SSE2 (best)
linux-x86-64-mmx        Linux, AMD x86-64, 32-bit with MMX
```

I've found linux-x86-sse2 to give the best performance on my intel based workstation.

```
cluster@server:~/source/john-1.7.2-bp17-mpi8/src$ make clean linux-x86-sse2
```

If the programs compiles OK check if it works

```
cluster@server:~/source/john-1.7.2-bp17-mpi8/src$ ../run/john -format=DES -test
Benchmarking: Traditional DES [128/128 BS SSE2]... DONE
Many salts:     1994K c/s real, 1994K c/s virtual

Only one salt:  1658 c/s real, 1654K c/s virtual
```

Move your newly compiled executables to ~/bin

```
cluster@server:~/source/john-1.7.2-bp17-mpi8/src$ mv ../run/* ~/bin
```

Run john and make sure you have the _mpi version.
If it does not then something is wrong with your PATH.

```
cluster@server:~/source/john-1.7.2-bp17-mpi8/src$ john|grep version
John the Ripper password cracker, version 1.7.2_bp17_mpi
```

Configuring extra nodes

Do the following each time you add a new node to the cluster.

This particular node has the IP address 10.0.0.2

Pre requirements

A static IP address or a reserved IP in DHCP.

Network configuration

Follow the same instructions as for the server, but of course use the correct IP address.

Required packages

Same as on the server.

User configuration

Choose the same password for 'cluster' as you did on the server.

```
petur@node1:~$ sudo useradd -m -s /bin/bash cluster
petur@node1:~$ sudo passwd cluster
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
petur@node1:~$ sudo su - cluster -c "mkdir ~/bin;export PATH=~/bin:$PATH"
```

Configuring MPICH

The following commands are to be executed from the server and **not on the new node.**

Configure password less SSH from the server to the node.

```
cluster@server:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/cluster/.ssh/id_rsa):
Created directory '/home/cluster/.ssh'
Enter passpharse (empty for no passpharse):
Enter same passpharse again:
```

```
Your identification has been saved in /home/cluster/.ssh/id_rsa.
Your public key has been saved in /home/cluster/.ssh/id_rsa.pub.
The key fingerprint is:
0f:d7:c4:14:cf:06:11:d5:80:ec:1f:c3:f3:3b:7f:22 cluster@server
The key's randomart image is:
[picture omitted]
```

```
cluster@server:~$ ssh cluster@10.0.0.2 mkdir -p .ssh
cluster@10.0.0.2's password:
cluster@server:~$ cat .ssh/id_rsa.pub | ssh cluster@10.0.0.2 'cat>>.ssh/authorized_keys'
cluster@10.0.0.2's password:
cluster@server:~$ ssh cluster@10.0.0.2 'cat /proc/cpuinfo|grep processor|wc -l'
2
cluster@server:~$ echo 10.0.0.2:2 >> ~/mpd.hosts
```

```
cluster@server:~$  for i in `cut --delimiter=: -f1 ~/mpd.hosts`;do scp ~/.mpd.conf cluster@$i:~;scp
~/mpd.hosts cluster@$i:~;done
The authenticity of host '10.0.0.1 (10.0.0.1)' can't be established.
RSA key fingerprint is 2d:94:c6:40:b0:02:04:d9:86:c8:16:f3:e6:a7:9f:35.
Are you sure you want to countinue connecting (yes/no)? Yes
Warning: Permanently added '10.0.0.1' (RSA) to the list of known hosts.
cluster@10.0.0.1's password:
.mpd.conf       100%  16       0.0KB/s         00:00
cluster@10.0.0.1's password:
mpd.hosts       100%  22       0.0KB/s         00:00
.mpd.conf       100%  16       0.0KB/s         00:00
mpd.hosts       100%  22       0.0KB/s         00:00
```

Now add the line:
10.0.0.2 node1

to the /etc/hosts file on the server.

The final step is to replace the /etc/hosts on ALL of the node machines with the new /etc/hosts from the server. If this is not done then the following error will be shown when you try to boot the cluster **mpdboot_server (handle_mpd_output 407): failed to handshake with mpd on 10.0.0.2; recvd output={}**

Installing John the Ripper

Same as on the server.

Basic commands

Boot up the cluster using “mpdboot –verbose –ncpus=1 -n 2”

- --verbose :: gives us better overview of what's going on in case of a failure.
- --ncpus=1 :: tells the server machine to assign 1 core to the cluster.
- -n 2 :: use 2 computer (server + 1 node).

```
cluster@server:~$ mpdboot --verbose --ncpus=1 -n 2
running mpdallexit on server
LAUNCHED mpd on server via
RUNNING: mpd on server
LAUCNHED mpd on 10.0.0.2 via server
RUNNING: mpd on 10.0.0.2
```

Check if the cluster is working:

mpdtrace - lists all the nodes in the cluster

mpiexec -np 3 hostname, means “run the hostname command using three cores”

```
cluster@server:~$ mpdtrace
server
node1
cluster@server:~$ mpiexec -np 3 hostname
server
node1
node1
```

mpdallexit - Shuts down the cluster

```
cluster@server:~$ mpdallexit
```

Using the MPI cluster to crack passwords.

I'll use a simple MD5 hash as an example.

```
cluster@server:~$ echo user:47584a15f1ba6c65da3a2ef8e43e606b > crackme1.md5
cluster@server:~$ mpdboot --ncpus=2 -n 2
```

```
cluster@server:~$ for i in `cut --delimiter=: -f1 ~/mpd.hosts`;do scp ~/crackme1.mp5
cluster@$1:~;done
```

The above command used to distribute files to the cluster can be easily scripted.

```
--- distributer.sh begins ---

#!/bin/bash
# usage: ./distributer.sh filename
for x in `cut –delimiter=: -f1 ~/mpd.hosts`;do scp $1 cluster@$i:~;done

--- distributer.sh ends ---
```

Use ctrl+c once the password has been found.

```
cluster@server:~$ mpiexec -np 3 john --format:raw-MD5 crackme1.md5
Loaded 1 password hash (Raw MD5 [raw-md5 SSE2])
Loaded 1 password hash (Raw MD5 [raw-md5 SSE2])
Loaded 1 password hash (Raw MD5 [raw-md5 SSE2])
petur1           (user)
Process 2 completed loop.
Threat: 2 guesses: 1    time     0:00:00:02 (3)  c/s: 5616K trying: petciL – petusc
^Ccluster@server:~$
```

Remember to shutdown the cluster after use

```
cluster@server:~$ mpdallexit
```

If you have any questions, comments or would like to contribute to this document please send me an email to petur [at] petur [dot] eu