OpenMPIが実行可能なクラスタを組む！

Step.0 本解説に関して

本解説はUbuntuを用いてOpenMPIの実行環境（並列演算装置）のセットアップの手順を解説したものである。
複数のコンピュータを用いてOpenMPIを実行できる環境構築を目標とします。OpenMPIの書き方に関しては最小限しか表記しません。

最終構成
使用OS:Ubuntu Server16.04
親(Master)ノードにNFSサーバー,DHCPサーバ,DNSサーバーを構築し、子(Slave)ノードでの設定を最小限にする。

免責
本解説はhermemo.comが備忘録として作成した解説したものであるため、至らない点が多々あります。本解説に関して発生したいかなる損失・障害に関してhermemo.comは一切責任を負いません。

お願い
本解説を読んで不明な点、誤字、脱字等が有りましたら、ご連絡頂ければ幸いです。また、「本解説を参考にした！」等の連絡もお待ちしています。
ご連絡はhermemo.comのお問い合わせフォームからお願いします。

2017/11/12 hermemo.com: Sunaipu

Step.1 ネットワークに接続する

/etc/network/interface に設定を記述する
設定方法としてDHCPを用いる場合と用いずに静的に設定する方法がある。
DHCPサーバーを設けることで子ノード全てでIPを静的に設定する必要がなくなる。ただ、プログラムを実行する際にIPアドレスが固定されている必要があるため、DHCPを用いる場合は注意が必要である。
本解説中[Step.5]でDHCPサーバーを用いてIPを自動割当し、DNSサーバーを用いることで上記の問題を解決できるようにしてある。

1.1 インターフェース名の確認

$fdisk -a
[fdisk]のみの場合動作していないインターフェースが表示されないため、オプション[-a]を用いる
インターフェース名の例）enp, eno, enp0s3. eth, wlan　など

1.2 ネットワークの種類

ネットワークは大きく分けてグローバルネットワークとローカルネットワークに分けることができる。グローバルネットワークであれば、インターネットにもアクセスすることができ、ソフトウェアのインストールには必須である。ローカルネットワークは今回の場合SSH, OpenMPIの通信に利用する。
ローカルネットワークは同じスイッチングハブにLANケーブルを挿し、同じDHCPサーバーからIPを取得してくることで簡単に構築することができる。
問題はグローバルネットワークへの接続だが、グローバルとローカル両方に接続する場合ルーティングの設定が必要になり、多少面倒である。よって今回は親(Master)ノードのみをグローバルネットに接続し、子（Slave)ノードは親ノードを介してグローバルネットに接続させる。

1.3 名称を理解する

今回のネットワークの設定では下記の項目を設定する。(DHCPを用いる場合は無視してもらって構わない)

address	アドレス（いわゆるIPアドレス）
netmask	ネットマスクと呼ばれる。IPアドレスの設定領域などを表すもの。詳しくはgoogleに聞いてほしい。 また、特に問題がない場合（ローカル接続）は[255.255.255.0]で問題ない。（ノード台数が200台を超える場合は変更）

network	ネットワークのアドレス。netmaskおよびaddressによって変化する。 IP:192.168.10.1 Mask:255.255.255.0 Network[192.168.10.0] IP:192.168.10.6 Mask:255.255.255.0 Network[192.168.10.0] IP:192.168.10.1 Mask:255.255.0.0 Network[192.168.0.0] IP:192.168.20.6 Mask:255.255.0.0 Network[192.168.0.0] netmaskの0と同じ場所をIP でも0 にして設定する。
broadcast	networkと同様の方法で入力する。ただし、broadcastは0ではなく255に変換する。 IP:192.168.10.1 Mask:255.255.255.0 Broadcast[192.168.10.255]
gateway	外に出るときのアドレス。（グローバルの場合のみ設定） 固有値であるため、各機関によって異なる。
DNS	DNSサーバー。（グローバルの場合のみ設定） これも固有値であるため、各機関によって異なる。

1.4 ネットワークの設定（DHCP)

$sudo vi /etc/network/interface

auto [deviceName]

Iface [deviceName] inet dhcp

1.5 ネットワークの設定(Static)

$sudo vi /etc/network/interface

auto [deviceName]

Iface [deviceName] inet static

address [address(192.168.10.1]

netmask [nermask(255.255.255.0)]

network [network(192.168.10.0)]

broadcast [broadcast(192.168.10.255)]

//以下グローバルのみ

gateway [gateway]

dns-nameserver [DNS]

1.6 注意

ローカルとグローバルように複数のインターフェースを用意した場合sshなどを行うときにエラーを起こす場合がある。おそらく使用するインターフェースを自動で選択していて、間違った方を選んでしまっているからだと思われる。その為、実行するときにインターフェースを指定してあげることがベストではある。もしくは使用しないインターフェースを終了させると改善される。

インターフェースの終了

$sudo ifdown [deviceName]

追記：[Step.5]でこの問題は解決します。

Step.2 アップデートとインストール

このステップではリポジトリ、カーネルのアップデートと必要になるソフトウェアのインストールを行います。
なお、ubuntu導入時にOpenSSHServer を導入していることを前提としているので含んでいない。
一覧
・gcc
・g++
・OpenMPI
・NFS
・Vim
・dnsmasq

※このステップはExtra.1を用いることで簡略化できます。

2.1 アップデート

カーネルやリポジトリを最新のものにする。

$sudo apt-get update
パッケージリストをアップデートします。
$sudo apt-get -y upgrade
すでにインストールされているパッケージを更新します。
$sudo apt-get -y dist-upgrade
カーネルの更新
$sudo apt-get -y autoreemove
使っていないパッケージを自動削除
$sudo apt-get -y autoclean
使っていないパッケージのアーカイブファイルを削除

2.2 gcc,g++

gcc , g++ はC言語, C++用のコンパイラです。とりあえず入れておきましょう。
$sudo apt-get install gcc
$sudo apt-get install g++

2.3OpenMPI

メインのOpenMPIである。OpenMPIはバイナリとライブラリに分かれているため二種類インストールが必要である。
$sudo apt-get install openmpi-bin
$sudo apt-get install libopenmpi-dev

//確認
$mpirun --version
$mpicc --version
バージョンが表示されればインストールできている。

2.4 NFSとは

必要に応じてNFSをインストールする。これはファイルサーバーの一種でOpenMPIはその特性上計算ノード全てに実行ファイルがなくては並列演算できない。よって全てのノードにファイルをコピーする必要がある。しかし、NFSをインストールしておくとその手間がなくなる。
理由としては親ノードのディレクトリを子ノードでマウントするからである。
その為、親ノードと子ノードで設定等が異なる。設定は後述。

2.5 NFS(親ノード)

親ノードには[nfs-server]をインストールする。
$sudo apt-get install nfs-kernel-server

2.6 NFS(子ノード)

子ノードには[nfs-common]をインストールする。
$sudo apt-get install nfs-common

2.7 vim

UbuntuはおそらくVimが初期から入っていますが、他のOSの場合Viが入っていて多少操作が異なるため、一応インストールしておきます。
$sudo apt-get install vim

2.8 DHCP, DNSサーバー

DHCPサーバー、DNSサーバーを構築するために[dnsmasq]を使います。
$sudo apt-get install dnsmasq

Dnsmsqを停止
インストールすると自動で起動してしまうので、停止しておきましょう。
$sudo service dnsmasq stop

Step.3 NFSサーバーの構築

OpenMPIを実行する時、全てのノードに実行ファイルを用意する必要がある。また、それらの実行ファイルは同じPathを持っていなければならない。それぞれのノードの同じ場所にディレクトリを制作し、コピーする手間を省くため、マスターノードの一部のディレクトリを共有する。
それによって面倒な作業がなくなる。
今回は[/home/share],[/usr/local]を共有する。

3.1 概念的な話

LINUXにおけるマウントと言う動作は既存のディレクトリに対して、サーバーやデバイスのディレクトリをリンクさせる行為のことである。
今回の場合サーバー側のディレクトリをクライアント側のディレクトリにマウントさせるが、pathは同じ場所である必要がある。
例）
[masternodeIP]:~/home
[node1IP]:~/home

3.2 親（サーバー）側

前述の通り[/home]と[/usr/local]をネットワーク上に共有する。重要なのは公開するのはネットワーク上であること。IPではなくネットワークアドレスを用いる。

```
$sudo vi /etc/exports
/home/share        [network]/[netmask](rw,async,no_root_squash)
/usr/local         [network]/[netmask](rw,async,no_root_squash)
```

※/home/share, /usr/localの後ろ以外にスペースはない。[netmask]などの後ろにはスペースを入れないことに注意せよ。

3.3 子（クライアント）側

クライアントの設定はfstabというファイルを編集します。記入ミスをするとOSが起動しない（シングルブート状態）になる場合があるので注意して行うこと。

```
$sudo vi /etc/fstab
（一番下に追加）
[サーバーIP]:/home/share        /usr/local    nfs    defaults    0    0
[サーバーIP]:/usr/local         /usr/local    nfs    defaults    0    0
```

[サーバーIP]は先程設定した親ノード、サーバーのIPアドレスを示します。

3.4 確認

子ノード再起動時に親ノードのディレクトリがマウントされているかどうか確認します。
$df -h
結果）
下の方にIPアドレスを含むディレクトリが表示されていれば成功です。

Step.4 Hostname

ホスト名とはIPと対応させることのできる名前です。名前解決(IPとHostNameを変換)することができれば、IPアドレスを毎回指定しなくでもHostNameを指定することで、そのパソコンを指定することができます。

ホスト名は各ノードがそれぞれのIPとHostNameを知っている必要があります。今回は手動で設定する方法とDHCPサーバ, DNSサーバを用いて自動設定する方法を紹介します。どちらかをしておきましょう。

ホスト名とIPアドレスの対応表を作成します。これをしておくことで、IPを入力しなくてもホスト名を入力することでSSH等が可能になります。
例）$ssh node@192.168.10.1 ==> $ssh node@node1
$sudo vi /etc/hosts
[IP] {hostName]

書式は[IP]を記入してスペース後[hostName]を記入します。
この場合[IP]は[hostName]に等しいことになり、逆でも参照可能です。

Step.5 DHCP, DNSサーバー

Step.1でネットワークに接続する際にも触れたが、OpenMPIでプログラムを実行する際にはコンピュータのIPが固定されている必要がある。正確にはマスターノードが全てのコンピュータのIPアドレスを知っている必要があり、静的に設定するかDHCP,DNSサーバーを構築する必要がある。

5.1 DHCPサーバーとは

DHCPサーバーがネットワーク内に有り、IPの設定がDHCPサーバーから取得するように設定されている場合、ネットワーク内のDHCPサーバーを検索して、基本的にはランダムにDHCPサーバーがIPを定めます。（Macアドレスを使って固定IPを与えることも可能）

再三言ってきましたが、プログラムの実行のためにはIPが固定されている必要があります。起動毎にIPが変わってしまうと、 マスターノードからアクセスできなくなります。DHCP サーバーは基本的にランダムに割り当ててしまうため、これだけでは静的には設定されません。

5.2 DNSサーバーとは

DNSサーバーはDomain Name Systemの略称で、IPアドレスとHostNameを関連付けるサーバーです。インターネットで[google.com]と打つとGoogleのページに移動すると思います。それはDNSサーバーがgoogle.comという名前をIPに変換してくれているからです。

今回、DNSサーバーを構築し、ネットワーク内にに設置することで、DHCPサーバーによってランダムに割り当てられたIPアドレスをDNSサーバーに集約し、HostNameで参照することでIPアドレスが動的であっても、参照する方法は静的になります。

ここでポイントなのはHostNameで参照することで静的になるということです。

5.3 dnsmasqを使う

DHCP,DNSサーバーを構築するためにdnsmasqを用います。基本的にはdnsサーバーですが、dhcpサーバーも構築できます。セットで構築しましょう。

5.4 設定

初期にの設定ファイルは項目が非常に多くて、それを編集するのはかなり面倒です。そこで、そのせてファイルともう一つ設定ファイルを作ってそれ両方を利用します。そのために[/etc/dnsmasq.conf]内の[conf-file=/etc/dnsmasq.more.conf]のコメントアウトを外します。
本環境では659行目

```
$sudo vi /etc/dnsmasq.conf
…
# Include another lot of configuration options
conf-file=/etc/dnsmasq.more.conf
…
```

以降詳細な設定は[/etc/dnsmasq.more.conf]に記述していきます。

```
$vi /etc/dnsmasq.more.conf
local = /local/
domain = [Domain]
expand-hosts
dhcp-range = 192.168.20.2, 192.168.20.255, 12h
dhcp-option = option:router. 192.168.20.1
dhcp-option = option:dns-server, 192.168.20.1
```

設定の基本はDHCPサーバーにIPを要求してきたコンピューターにどのような情報を返すかという設定です。

[dhcp-range = 192.168.20.2, 192.168.20.255, 12h]
: [192.168.20.2] ～ [192.168.20.255]のIPが12時間の有効期限で割り当てられます。

[dhcp-option = option:router. 192.168.20.1]
: ルーター（DHCPサーバー）のIPを指定します。

[dhcp-option = option:dns-server, 192.168.20.1]
: DNSサーバーを指定します。名前解決のために問い合わせるサーバです。複数しているすることも可能で、ローカルネットワーク外のDNSを指定するとインターネットにも接続できるようになります。

5.5 注意

DHCPサーバー自身はDHCPサーバーからIP等を取得してくる事ができないので、静的に割り当てる必要があります。方法に関しては[Step1]を参照。

Stop.6 SSH 公開鍵認証

SSHを行う際にパスワードの入力を求められる設定の場合、MPI を実行する場合も求められことになる。その時全てのノードのパスワード入力が必要になり、ノード数が増えると非常に手間である。そこで、公開鍵認証を設定して自動で認証できるように設定する。

6.1 概念的な話

SSH の公開鍵認証はクライアント側（接続する側）で鍵を発行して、サーバー側（接続される側）で認証を行う。今回の場合**親ノードがクライアント側**になり、**子ノードがサーバー側**になる点に注意せよ。

6.2 親ノードで鍵を作成する

親ノードで公開鍵と非公開鍵のセットを制作する。
$ssh-keygen
その後出力するファイル名やパスフレーズなどを聞かれるが特に設定必要はないので未入力のままEntertを三回ほど押す。
すると各ユーザーのホームディレクトリ直下に[.ssh]という隠しディレクトリが生成され、中に鍵は収納されている。
生成されたことを確認するには下記のコマンドを実行すれば良い。
$ls /home/[username]/.ssh/
[id_rsa], [id_rsa.pub]が生成されていると成功。
[id_rsa] が秘密鍵で、[id_rsa.pub]が秘密鍵である。

6.3 親ノードのディレクトリのアクセス権限

SSHはディレクトリのアクセス権限が非常に重要である。
$sudo chmod 700 /home/[username]/.ssh/

6.4 子ノードのディレクトリに鍵を移動

[5.3]で作成した公開鍵を子ノード（サーバー側）に登録します。そのために、鍵を移動します。
方法はいくつかありますが、今回はNFSとscpの二通りの方法を記します。

NFS
NFSを利用する場合非常に簡単に移動が可能です。共有されているディレクトリ上にファイルを移動させるだけです。
$cp /home/[userneme]/.ssh/id_rsa.pub /home/share/

Scp
scpに関しては[Extra.2]を参照のこと。
$sudo scp /home/[username]/.ssh/id_rsa.pub [userName]@[IP]:[path]

6.5 子ノードでパスワードを登録する

まず、[/home/[username]/.ssh/]に[authorized_keys]というファイルが有るか確認します。なお、[authorized_keys]は自動認証の登録ファイルです。ここに登録されているパスワードは自動認証されます。[authorized_keys]はデフォルトの名前なので、変更しても構いません。その場合[6.7]で行う[SSHサーバーの設定]の際に注意てください。

[authorized_keys]がない場合
$touch authorized_keys

パスワードを設定する。
[5.4]で移動してきた鍵をauthorized_keyaに追加します。
$cat id_rsa.pub >> authorized_keys

※[.ssh]のディレクトリがない場合
手動でディレクトリを作成するか、公開鍵を発行すると自動で作成されます。
手動でディレクトリを作る
$mkdir /home/[username]/.ssh/

公開鍵を発行して自動で作成する。
$ssh-keygen
発行される公開鍵は無視して構いません。

6.6 子ノード上のアクセス権限

$sudo chmod 700 /home/[username]/.ssh/
$sudo chmod 600 /home/[username]/.ssh/authorized_keys

6.7 SSHサーバーの設定

[/etc/ssh/sshd_config]がSSHサーバーの設定ファイルです。ここで公開鍵認証の許可とその際に使用する鍵ファイルの設定を行います。
$sudo vi /etc/ssh/sshd_conf
31行目: PubkeyAuthentication yes
32行目: AuthorizedKeysFile %h/.ssh/authorized_keys

この二箇所がコメントアウトされている場合があるので有効化しておく。また、32行目のpathはデフォルトの場合なので別の名前を使用する場合はpathも変更する。

Extra

Extra.1 Step2を簡略化

アップデート及びインストールはシェルスクリプトをネット上に公開しています。その為、面倒な方はそちらを使用してください。
公開場所: hermemo.com/publicData/setup.sh (2017/04/23/925B)
利用方法）
1)$wget hermemo.com/publicData/setup.sh
2)$bash setup.sh
以上、放置するだけでStep2は終了します。なお、Step3.3記載のNFSに関しては子ノード用のライブラリのみイントールされます。親ノードの場合はStep3.4を別途実行してください。

Extra.2 scpコマンド

SSH経由でファイルを送信するコマンド。SSH接続ができる環境であればそれ以外は必要としない。
$sudo scp [fileName] [userName]@[IP]:[path]
送信側でコマンドを実行する。
[fileName]で送信するファイルを指定
[userName]で接続される側で使用するユーザー名を指定（未指定の場合はroot）
[IP]これは名前問題が解決できれいる場合はホストネームでも問題ない。それ以外は受け取る側のIPを入力する。
[path]送信先のパスを指定する。例）~/ (ホーム）
※IPとPathの間に":"（コロン）があることに注意せよ。