MPI備忘録（環境構築編）

・実行環境

フロントマシン

CPU：Ryzen 7 3700X

GPU：Geforce RTX 3070

メモリ：DDR4-3200 64GB

ドライブ：USB3.2 64GB

バックマシン

CPU：Core i5 3320M

GPU：Intel内蔵グラフィックス

メモリ：8GB

ドライブ：SSD 128GB

OS：Ubuntu 22.04

基本的には「高校生のためのクラスター入門」に書いてある通りです。そこに書いていないけど必要だったことや、出てきたエラーについてをここに書いておきます。

ちなみに、「MPICH」は「エムピッチ」と読むらしいです。

1. **ダウンロードとインストール**

・OSのインストールは、「高校生…」にならいます。

・MPIを入れる前に

MPIのインストールには、「gcc（Cコンパイラ）」、「gfortran（Fortranコンパイラ）」、「make（ビルドツール）」が必要になります。コマンドで

＜フロント＞

sudo apt install build-essential //gccとmakeを一緒にインストール

sudo apt install gfortran

を打ちます。ここで、「高校生…」ではインストールコマンドが「yum」ですが、ここでは「apt」となっています。これは、Linuxのなかで、UbuntuなどのDebian系OSは「apt」、CentOSなどのRedHat系は「yum」となっています。

「高校生…」にある通り、ダウンロードして、展開して、ビルドして…とやってもいいですが、コマンドから直接インストールすることもできます。というかこっちのほうが簡単です。

＜フロント＞

sudo apt install mpich

sudo apt install mpich-devel

1. **MPIがインストールされているか確認**

まず、フロント１台だけでMPIを動かしてみます。いきなり並列環境を作ると、エラーが出たときにネットワークによるものなのかMPIによるものなのかわからないからです。

以下のコマンドでコンパイルします。パスやファイル名は適宜変更してください。

**<プログラム①：sample1.c>**

#include <stdio.h>

#include <mpi.h>

int main(int argc, char \*argv[]){

int rank; //**ランク（自分が何プロセス目か）**

int proc; //**全プロセス数**

int namelen = 10;

char \*name[namelen]; //**プロセスを行うマシンの名前**

MPI\_Init(&argc, &argv); //MPI**の初期化**

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank); //**ランクの取得**

MPI\_Comm\_size(MPI\_COMM\_WORLD, &proc); //**全プロセス数の取得**

MPI\_Get\_processor\_name(name, &namelen); //**名前の取得**

printf(“Hello, World! From process %d of %d, %s\n”, rank, proc, name);

MPI\_Finalize(); //MPI**を終了**

return 0;

}

＜フロント＞

cd ＜ファイルパス＞ //sample1.cを保存した場所 ~/program/MPI など

mpicc -o sample1 sample1.c

すると、なんか色々出てくるかもしれませんが、「error」と書いていなければ（「warning」だけなら）たぶん大丈夫です。コンパイルに成功すると、該当フォルダにsample1という実行ファイルができます。

＜フロント＞

ls

sample1.c sample1 //表示結果

つづいて、以下のコマンドで実行します。

＜フロント＞

mpirun -np 4 ./sample1

-npの後に続く4という数字は、sample1を並列実行するプロセスの数です。上の例では、sample1を４並列で実行することを表します。

そうすると、たぶんこんな感じに出力されると思います。

Hello, World! From process 0 of 4, node01

Hello, World! From process 2 of 4, node01

Hello, World! From process 1 of 4, node01

Hello, World! From process 3 of 4, node01

※１個目の数字（１から４）は順番が変わることがあります。

こう表示されれば、MPIのインストールは成功しています。

◎**WindowsでMPIを使う**

MPIのプログラムを書く時などはWindowsのほうが都合がいいこともあるので、MPIのWindowsへのダウンロード方法も書いておきます。

下のサイトの「MS-MPIのダウンロード」というところから、MS-MPIをダウンロードします。

<https://learn.microsoft.com/ja-jp/message-passing-interface/microsoft-mpi>

グラフィカル ユーザー インターフェイス

自動的に生成された説明ダウンロードに移って、赤の「Download」をクリックすると、下のような画面になります。

２つにチェックを入れて、Nextを押すと、ダウンロードが始まります。

ダウンロードした２つのインストーラーを実行して、インストールします。

デフォルトのインストール場所は、

mpiexec.exe（実行ファイル）: C:\Program Files\Microsoft MPI\Bin

mpi.h（ヘッダーファイル）: C:\Program Files(x86)\Microsoft SDKs\MPI\Include

msmpi.lib（ライブラリ）: C:\Program Files(x86)\Microsoft SDKs\MPI\Lib\x64

です。

それでは、プログラムを実行します。

まず、gccなどのCコンパイラをダウンロードしておき、２ページのsample1.cを、以下のコマンドでコンパイルします。Windowsでは、コマンドを打つときはコマンドプロンプトかWindows PowerShellを使います。

gcc sample1.c -o sample1.exe -I” C:\Program Files(x86)\Microsoft SDKs\MPI\Include” -L” C:\Program Files(x86)\Microsoft SDKs\MPI\Lib\x64”

-oは出力先を指定するオプション、-Iがインクルードするヘッダーファイル（sample1.cの３行目でmpi.hをインクルードしている）のパスを指定するオプション、-Lがライブラリファイルを指定するオプションです。

コンパイルに成功したら、sample1.exeという実行ファイルができるはずです。そうしたら、以下のコマンドで実行します。

mpiexec -np 4 sample1.exe

Linuxでやったときと同じく、-npオプションの数値は並列実行するプロセス数です。成功したら、３ページにあったものと同じような結果が出力されると思います。

＜エラー対策＞

Q用語“mpiexec”は、コマンドパレット……として認識されませんでした。名前が正しく…

といわれたら

A mpiexec.exeのパスが環境変数に通っていないと思われます。「コントロールパネル」を開き、「環境変数」で検索して「システム環境変数の編集」を開いてください。出てきたウィンドウの下のほうの表にある「Path」をクリックし、C:\Program Files\Microsoft MPI\Binのパスを追加します。そして、コマンドプロンプトを再起動すれば行けると思います。環境変数の追加方法はいろんなサイトに記載されているのでそれを参考にしてください。

1. **ネットワークの構築**

並列処理を複数のマシンで行うため、ネットワーク環境を作ります。

まず、フロントマシンとバックマシンを、LANケーブルでスイッチングハブにつなぎます。ここでのハブは１番安いやつでいいです。

**3-1 IPアドレスの設定**

まずIPアドレスを設定します。以下の設定はフロント・バック共通です。

ターミナルを開き、以下のコマンドを打ちます。

ip -a

すると、こんなかんじに表示されると思います。

1: lo: <LOOPBACK>…

link/loopback …

inet …

…

2: **ens4s0**: …

Link/ether …

inet …

ここで、下線を引いた「ens4s0」という文字列を覚えておきます。（マシンによって異なります）

Ubuntuで固定IPアドレスを設定するには、/etc/netplan/\*\*-cloud-init.yaml（\*\*は数字２桁）を編集します。ここで、netplanが読み込むこのyamlファイルには、アルファベット順で参照し、最後に参照したものが優先されるという特徴があります。そのため、たとえば10-cloud-init.yamlと20-cloud-init.yamlというファイルがあったとすると、後者の設定のほうが優先されます。

今回は、デフォルトの設定も残すため、\*\*-cloud-init.yamlをコピーして編集します。ここでは、\*\*が01という数値であったとします。

cd /etc/netplan

cp 01-cloud-init.yaml 10-cloud-init.yaml //ファイルのコピー

vi 12-cloud-init.yaml または sudoedit 12-cloud-init.yaml

個人的にsudoeditのほうが使いやすくおすすめです。

そうしたら、以下のように書き込みます。

network:

ethernets:

**ens4s0** //上で覚えた文字列

dhcp4: false

dhcp6: false

addresses: [192.168.1.30/24]　//末尾の30は2~254までの任意の数

gateway4: 192.168.1.1

nameservers:

addresses: [8.8.8.8]

version: 2 //これはたぶん最初から書いてある

ちなみに、４行目と５行目の「DHCP」は、IPアドレスを自動で割り振る仕組みです。ここでは自分でIPアドレスを設定するのでオフにしておきます。

終わったら、以下のコマンドで設定を更新します。

sudo netplan apply

これで、IPアドレスが変わっているはずです。ip aコマンドで調べてみてください。

ここで、フロントとバックがちゃんとつながっているか確認します。以下のコマンドを打ってみてください。

ping 192.168.1.32　//さっき設定した、自分のではないIPアドレス

１秒に１行出力されるので、適当なところでCtrl + Cを押します。そうすると、以下のように表示されます。

PING 192.168.1.32 …… 56 data bytes

64 bytes from 192.168.1.32 …… : …… time=0.501 ms

64 bytes from 192.168.1.32 …… : …… time=0.499 ms

64 bytes from 192.168.1.32 …… : …… time=0.481 ms

64 bytes from 192.168.1.32 …… : …… time=0.519 ms

64 bytes from 192.168.1.32 …… : …… time=0.522 ms

64 bytes from 192.168.1.32 …… : …… time=0.514 ms

--192.168.1.32 ping statistics—

6 packets transmitted, 6 received **0% packet loss**, time 3.036ms

……

ここで注目するのは、下線を引いた〇％ packet lossというところです。これが0%ならば、通信は成功しています。一方、通信に失敗している場合は、100% lossと表示されます。

＜通信に失敗する場合＞

上から順に試してみてください。

・pingで指定したIPアドレスは正しいですか？

・sudo netplan applyをしましたか？

・LANケーブルはちゃんと刺さっていますか？ハブの電源は入っていますか？

・ここまででできなかったら、3-1の最初に戻って、もう１回IPアドレスを設定しなおしましょう。

・それでもできなければ、設定するIPアドレスを変えてみてください。

**3-2 SSHの設定**

次に、SSH通信の設定をします。

ちなみに、SSHというのはコンピューターどうしをリモートアクセス（有線・無線にかかわらず）する際に通信を暗号化してやり取りするためのプロトコルです。Secure Shellの略です。昔は、リモートアクセスにはTELNETというプロトコルを使っていたようなのですが、TELNETは通信がすべて平文で行われており、セキュリティ上問題があることから、現在ではSSH通信が主流となっています。

次に、ssh関連のパッケージをインストールします。

＜フロント＞

sudo apt install openssh-server

sudo apt install openssh-clients

＜バック＞

sudo apt install openssh-server

そうしたら、SSH関連の設定を変えます。/etc/ssh/sshd\_configを開きます。

＜フロント・バック＞

cd /etc/ssh

sudoedit sshd\_config

以下の部分を変えます。

#PubkeyAuthentication yes

→PubkeyAuthentication yes （#を削除）

編集が終わったら、sshdを再起動し、設定を反映させます。

＜フロント・バック＞

sudo systemctl restart sshd

これでSSHの初期設定が完了しました。

**3-3 SSH通信**

では、SSHを使って通信してみます。

SSHの暗号化には、パスワード認証方式と公開鍵認証方式の２つがありますが、MPIで通信する際にいちいちパスワードを入力していられないので、公開鍵方式を使います。公開鍵暗号に関しては、結構複雑なので、いろいろ読んで頑張って理解してください。

まず、公開鍵を作ります。SSH通信では、~/.sshディレクトリ（~はDesktopとかDocumentsなどがある、いわゆるホームディレクトリ）を使うので、それを作っておきます。

＜フロント＞

cd ~

mkdir .ssh

ssh-keygen -t rsa -N “” -f ~/.ssh/key1\_rsa

ssh-keygenが鍵を生成するコマンドです。-tオプションは暗号の方式を決めるものです。ここではRSA暗号を採用しています。-Nはパスフレーズを指定するものです。パスフレーズとは、パスワードの強い版みたいなやつです（「ワード」じゃなく「フレーズ」なので長い）。上述のとおりパスワードなんていらないのでパスフレーズなしを指定しています。-fオプションは鍵を生成する場所を指定するものです。「高校生…」にある方式と違って、ここではオプションで一括指定しています。どっちでもいいです。

終わったら、lsコマンドで鍵が生成されているか見てみましょう。key1\_rsaとkey1\_rsa.pubというのがあるはずです。

次に、作った公開鍵をバックマシンに渡します。

＜バック＞

cd ~

mkdir .ssh

＜フロント＞

ssh-copy-id -i ~/.ssh/key1\_rsa [user@192.168.1.32](mailto:user@192.168.1.32)

ここで、userは６ページの下のところで宛先のマシンにあるユーザー名（管理者アカウントでOK）を、192.168.1.32は送信先のバックマシンのIPアドレスを入力してください。IPアドレスのところはホスト名（PC名、たとえば学校から貸与されたものならSF11223344とか　コマンドhostnameで調べられる）

これが終わったら、鍵交換に成功しているはずです。コマンドラインにも「”ssh” ”192.168.1.32”を試してみて！」みたいな内容が表示されていると思います。ということで、つないでみます。

＜フロント＞

ssh user@192.168.1.32

これで、パスワード入力が求められずにつなげられたら成功です。「Last login: なんとかかんとか」みたいに表示されると思います。

ちなみに、生成する鍵の名前をid\_rsaにすると-iオプションをつけずに済みます。

＜パスワードが必要になる場合＞

・バックマシンを見て、鍵が渡されているか確認（~/.sshにauthorized\_keyがあればよい）

・そもそもフロントマシンにリモートアクセスしていないか（リモートアクセスした状態でのパスワードなしリモートアクセスはできないっぽい）

・鍵を作り直してみる

**3-4 NFSの設定**

NFSは、Network File Systemの略で、あるマシンにあるファイルを、同じネットワーク上にあるほかのマシンからもアクセスできるようにするシステム（プロトコル）です。Googleドライブをパソコンと同期させるとWindowsのエクスプローラーからもGoogleドライブのフォルダが見えるようになる、みたいな感じだと思います。

まずインストールします。ここらへんはOSによって異なります。

＜フロント＞

sudo apt install nfs-kernel-server

＜バック＞

sudo apt install nfs-common

次に、NFSとして公開するディレクトリを/mpiとします。

NFSとして外部のマシンからのアクセスを許可する設定をします。

＜フロント＞

cd /etc

sudoedit exports

これに、以下を書き込みます。

/mpi 192.168.1.32(rw,no\_root\_squash)

構文は、指定されているIPアドレスですが、192.168.1.0/24のようにCIDR表記で書くこともできます。この場合、192.168.1.1 ~ 192.168.1.254までを許可するということと同義です。

引数のrwは、読み書き(Read, Write)可能であることを示します。

できたら、NFSサーバを再起動します。

＜フロント＞

sudo systemctl enable nfs-blkmap.service --now

そのあとは、「高校生…」にあるようにバックの/etc/fstabを編集すれば、NFSが有効化されるはずです。フロントの共有ディレクトリにファイルを追加したり編集したりして、ちゃんと同期できているか確認してください。

今後、プログラムファイルはそのディレクトリに入れることになります。

*――私はこのやり方でできませんでしたー－*

マウントしたら、……not parmittedみたいな表示が出ました。

**3-4番外編　公式ドキュメントを読もう**

上に書いてある方法で2回やって2回ともダメだったのでおとなしく公式ドキュメントを見ることにします。

URLはこれ→ <https://ubuntu.com/server/docs/service-nfs>

（つながらないときはwayback machineをつかうとできます）

以下、””内はすべて上サイトからの引用です。

・introduction

“NFS allows a system to share directries and files with other over a network”

そのままです。その下にはNFSを使えばディスク容量を節約できるよーみたいなことが書いてあります。

・インストール

サーバー向け（フロントマシン）のインストールコマンドは、

sudo apt install nfs-kernel-server

のようです。

また、NFSを開始するには

sudo systemctl start nfs-kernel-server.sevice

だそうです。上にあるsudo systemctl enable nfs-blkmap.serviceは、このコマンドに書き換えたほうがよさそうです。

・設定

先ほどと同じように、/etc/exportsを編集します。

/mpi 192.168.1.32(rw,no\_root\_squash,no\_subtree\_rock)

ここで、no\_subtrr\_rockはセキュリティ認証をするかしないかの設定だそうです。そんなものあったらMPIが通らないのでnoを設定します。

適用するには次のコマンドを打ちます。

sudo exportfs -a

次はクライアント（バック）の設定です。

先ほどと同じように、nfs-commonをインストールしたら、コマンドからマウントします。

sudo mount 192.168.1.30:/mpi /mpi

このとき、何も表示が出ずにコマンドが終わったら、次のコマンドでNFSの疎通確認をします。

df -h

これで、最下行に192.168.1.30:/mpiと表示されていたら成功です。あとは、fstabの設定もすれば終了です。

1. **MPIを動かす**

いよいよ動かす、その前に、１つインストールするものがあります。

＜バック＞

sudo apt install mpich

MPICHのディレクトリもNFSでアクセスしてしまえばバックにインストールする必要はありません。「高校生…」でやっていることはこれです。ただ、私の環境ではNFSがうまくいっているか微妙な感じだったので、個別インストールすることにしました。

1台でやった時と同じようにmpiccでコンパイルします。

mpicc sample1.c -o a.out

次に、ホストファイルを書きます。これは、どのマシンで何プロセス動かすかを決めたファイルのことです。共有ディレクトリ内で作成し、編集します。

＜フロント＞

cd /mpi

sudo touch hosts

sudoedit hosts

＜ホストファイル書式＞

IPアドレス:プロセス数 user=ユーザー名

IPアドレス:プロセス数 user=ユーザー名

IPアドレス:プロセス数 user=ユーザー名

IPアドレス:プロセス数 user=ユーザー名

…

＜ホストファイル例＞

192.168.1.30:2 user=front

192.168.1.32:3 user=backnode01

192.168.1.33:3 user=backnode11

ここで、ユーザー名は、各マシンに渡した鍵を持っているユーザーの名前を入れてください。**フロントマシンも含め**同一のユーザー名の場合は、ここを省略することが可能です。プロセス数は、そのマシンで実行したいプロセスの数です。この指定では、上から順にプロセス0、プロセス1、プロセス2、となるようです。実際に実行されるプロセス数はmpirunのオプションとして指定されます。そのオプションの値がhostsに書かれたプロセス数より少なければ該当プロセスまでが割り当て（例えば、上に書いたhostsで4プロセスを実行すると、プロセス0,1がuser=frontで、プロセス2,3がuser=backnode01、user=backnode11はおそらく仕事をしない）られます。また、オプション地のほうが多ければ、hostsに書かれた数を超過したプロセスは各マシンに均等に割り振られるようです。

長々と書きましたが、ようやく実行です。

＜フロント＞

mpirun -f hosts -np 8 ./a.out

これで、以下のように表示されると思います。

Hello, World! From process 2 of 8, back01

Hello, World! From process 4 of 8, back01

Hello, World! From process 5 of 8, back02

Hello, World! From process 7 of 8, back02

Hello, World! From process 6 of 8, back02

Hello, World! From process 3 of 8, back01

Hello, World! From process 0 of 8, frontmachine

Hello, World! From process 1 of 8, frontmachine

表示されたら成功です。

◎ssh\_askpassと表示される場合

意味：ssh通信でパスワード要求が発生しています。

原因1：hostsに鍵を渡したユーザー名が記載されていない（ユーザー名を書かないと、おそらくシステムはフロントマシン内のユーザー名で探しに行く。例えば、上のhostsではフロントのユーザー名がfront、バックがbacknode01、backnode11となっているが、ユーザー名を書かないとシステムはバックマシン内のfrontという名前のユーザーにアクセスを求める。当然そんなユーザーはいないので接続は成立しない。）

私はこれで詰まってました。

原因2：鍵が正しく渡されていない

ssh IPアドレス コマンドで、パスワード入力なしで接続できるかどうか確かめてください。

・参考サイト

公式ドキュメントを見るのが一番ですが、それだけを見てもわからないところは多いので、ある程度説明しているサイトを載せておきます。

追記　でもやっぱり最後は公式ドキュメントに助けられました。

MPIの環境構築や基本コマンドまとめ

<https://qiita.com/kkk627/items/49c9c35301465f6780fa>

ssh-copy-idで公開鍵を渡す

<https://qiita.com/kentarosasaki/items/aa319e735a0b9660f1f0>

CentOS 7でNFSサーバを構築してみた

<https://qiita.com/tanuki_mujina/items/5c706b2eab6e7eed71fd>

MPICH2のインストール

<https://www.nslabs.jp/mpi-setup.rhtml>

MPI超入門（C言語編）

<https://www.cc.u-tokyo.ac.jp/events/lectures/13/MPIprog.pdf>

NFSサーバの構築手順（Ubuntu 20.4編）

<https://changineer.info/vmware/hypervisor/vmware_ubuntu_nfs.html>

【Ubuntu】固定IPアドレスの設定（無線LAN、Wi-Fi対応）：netplan

<https://office54.net/iot/linux/ubuntu-ipaddress-netplan>

MPICH公式ドキュメント

インストールガイド

<https://www.mpich.org/static/downloads/4.0.3/mpich-4.0.3-installguide.pdf>

ユーザーガイド

<https://www.mpich.org/static/downloads/4.0.3/mpich-4.0.3-userguide.pdf>

公式Wiki

<https://github.com/pmodels/mpich/blob/main/doc/wiki/Index.md>

Wiki内の実行コマンドに関するヘルプ

<https://github.com/pmodels/mpich/blob/main/doc/wiki/how_to/Using_the_Hydra_Process_Manager.md>

補足

・間違ってるところや質問等あったら言ってください

・数日で作ったものなのでもっといいやり方があると思います