UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL - UFMS

PROJETO MINI-CLUSTER

Aluno: Jonatas Santos Galvão

Tutor: Renato Porfírio Ishii

Campo grande – MS, 2022

Programa de Educação Tutorial

O que é Cluster ?

- Na computação, o termo define uma arquitetura de sistema capaz combinar vários computadores para trabalharem em conjunto ou pode denominar o grupo em si de computadores combinados.

- Cada estação é denominada “nó” e em conjunto formam o **cluster**. Em alguns casos, é possível ver referências como “supercomputadores” ou “computação em cluster” para o mesmo cenário, representando o hardware usado ou o software especialmente desenvolvido para conseguir combinar esses equipamentos.

Entenda as diferenças:

- cluster de alto desempenho: também conhecido como cluster de alta performance, ele funciona permitindo que ocorra uma grande carga de processamento com um volume baixo de gigaflops em computadores comuns e utilizando sistema operacional Linux, o que diminui seu custo drasticamente; São os mais usados em processamento pesado, como cálculos de genomas, sequenciamento de vírus, etc.

- cluster de alta disponibilidade: são clusters cujos sistemas conseguem permanecer ativos por um longo período de tempo e em plena condição de uso; sendo assim, podemos dizer que eles nunca param seu funcionamento; além disso, conseguem detectar erros se protegendo de possíveis falhas; São mais usados em servidores de mídias sociais que não podem sair do ar.

- cluster para balanceamento de carga: esse tipo de cluster tem como função controlar a distribuição equilibrada do processamento. Requer um monitoramento constante na sua comunicação e em seus mecanismos de redundância, pois, se ocorrer alguma falha, haverá uma interrupção no seu funcionamento. Ele mescla o melhor dos dois mundos: Redundância e alto desempenho.

O que vamos trabalhar ?

O tipo de cluster que o MPICH mais comumente gerencia é o de alto desempenho, então o objetivo será ter vários computadores com seus respectivos poderes computacionais somados – Aqui chamamos cada máquina de Nó da Rede, para isso vamos precisar de alguns requisitos, são eles:

-terminal, vamos usá-lo o tempo todo, sem qualquer interface gráfica.

-Raspberry Pi.

-Seu funcionamento, configurações básicas e seus principais componentes.

-Conhecimento básico em Redes LAN, DNS e DHCP.

-Configurar roteador ou fazer conexões em switchs, fixar IP's, definir gateways, etc.

-Por fim mas não menos importante: Requer que você tenha um conhecimento básico de alguma linguagem de programação entre Fortran, Python, C ou C++, as linguagens utilizadas pelo sistema de computação distribuída neste exemplo.

Materiais

-O Raspberry será nosso objeto de estudos, porém o que é ensinado aqui vale pra qualquer máquina! Apenas faça as adaptações necessárias, em vez do Raspbian, use Debian; em vez de ARM, use arquitetura x64.

-Switch ou um roteador de internet extra que faça o trabalho de distribuir portas de rede com a finalidade de criar uma segunda LAN

-PatchCords

-MicroSD Card 8GB ou maior.

-Fontes de alimentação para os rasberry

-Fonte de alimentação Switch

Procedimento

Pelo cerne do tutorial, conforme acima citado, espero que você já tenha uma prévia experiência em algumas áreas. Precisamos que você configure o básico em todos os Raspberries, instalando o Raspbian Lite (64-bit), sem desktop, essa versão tem o menor consumo de RAM, portanto você terá em torno de 900 Mb livres para usar nos serviços do MPICH essenciais.

De forma sintética, após baixar a ISO, gravar ela no SDCard e ligar o Raspberry, o sistema automaticamente irá realocar o espaço em todo o SDCard. Após isso, você poderá fazer login inicial nos Raspberries. Essa primeira configuração é interessante ter um monitor e teclado para certificar que está tudo em ordem.

Inicialmente, cada um dos Raspbarries precisamos atualizar o sistema operacional, para isso digite o seguinte comando no terminal:

$ sudo apt-get update

Logo que terminar você digita esse comando e aperta enter:

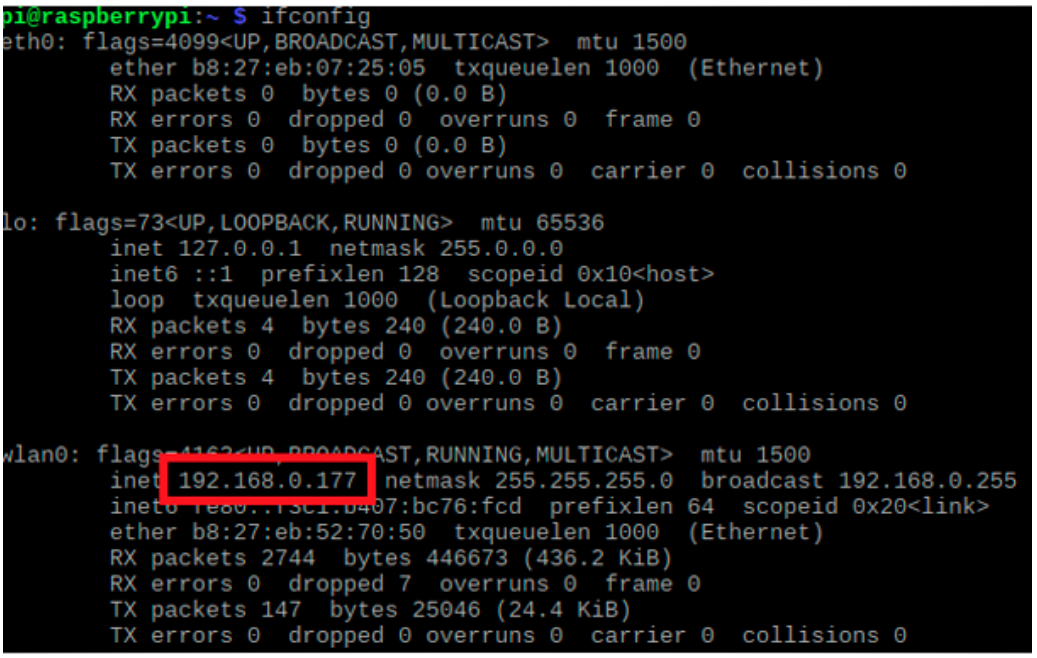
$ sudo apt-get upgrade

Frequentemente pode ser que algumas vezes a instalação pare, pois, para alguns pacotes será preciso dar permissão.

Finalizado, será necessário habilitar o SSH pelo rasp-config, para isso, Verifique na Raspberry o seu IP, digitando o comando no terminal:

$ ifconfig

O IP está destacado em vermelho

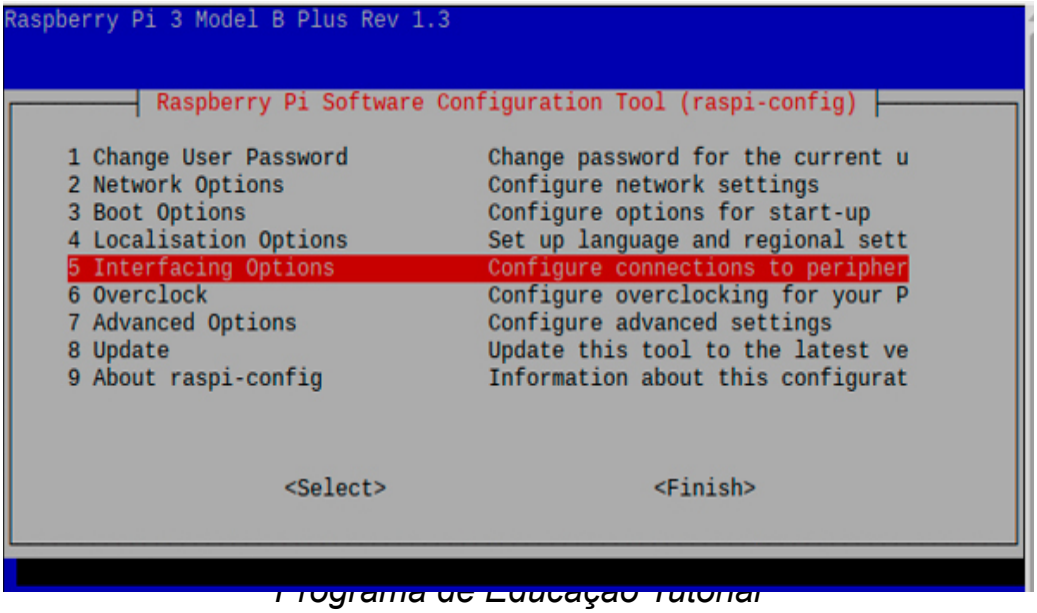


Anote o IP obtido da Raspberry Pi B+. Agora vamos entrar nas configurações para habilitar o SSH da Raspberry digitando o comando no terminal :

$ **sudo raspi-config**

Na janela de configurações selecione a opção 5, *Interfacing Options*

Selecione a opção 2 SSH



seguir selecione a opção SIM para habilitar o SSH

feito isso, agora vamos gravar de forma definitiva o IP de cada Raspbarry, para isso, execute o seguinte comando no terminal:

$ sudo nano /etc/dhcpcd.conf

e remova o "#" em frente as linhas para definir os IP's, ficando:

Para

```
#interface eth0
#static ip_address=192.168.0.70/24
#static routers=192.168.0.1
#static domain_name_servers=192.168.0.1 8.8.8.8

#interface wlan0
#static ip_address=192.168.1.62/24
#static routers=192.168.1.1
#static domain_name_servers=192.168.1.1 8.8.8.8
```

retornar as configurações originais (default) basta colocar o "#" antes das linhas. Deixando a linha como comentário e recebendo o IP automaticamente.

Cada modem/roteador possui uma configuração e você deverá adaptar o seu á isso. Você pode fixar os IP's dentro do próprio Raspberry ou, como eu prefiro, fixar diretamente fora da faixa DHCP direto no modem/roteador. Assim você evita problemas.
Com a rede pronta, vamos configurar o MPICH.

MPICH

Nesta fase, use o ClusterSSH para configurar o MPICH em todos os Raspberries de uma vez. O MPICH, formalmente MPICH2, é uma implementação portátil e disponível gratuitamente baseado no MPI (Mesage Passing Interface), um padrão para passagem de mensagens para aplicativos de memória distribuída usados em computação paralela. O MPI é uma forma de comunicar computadores em rede de forma rápida e direta, com latência baixa, usando linguagens de baixo nível como Fortran e C.
O conceito de computação distribuída dita basicamente que a primeira máquina definirá os comandos e serão repassados ás próximas máquinas; Para essa tarefa, cada uma das máquinas terá uma cópia do MPICH, do MPI4PY (mais abaixo explico sobre ele) e dos scripts .py que executarão no Cluster. Assim você não sobrecarrega a rede e nem a primeira máquina, a rede vai servir para transacionar dados de cálculos e não diretamente scripts ou softwares brutos. O que é bom pra diminuir ao máximo a latência.

OBS: NÃO utilize o mpich do repositório do Raspbian!
Eu sei da tentação de tentar "atalhar" o tutorial e dar "apt install mpich" diretamente, não faça isso. O mpich vai dar erro de "segmentation fault" na hora de rodar códigos Python e você vai ficar sem o que fazer a respeito.
O ideal é compilar o MPICH para o Raspberry (e qualquer hardware) como mostraremos abaixo.

Siga exatamente o seguinte, no ClusterSSH aberto, acessando todos os Pi's ao mesmo tempo, digite:

$ cd ~

$ mkdir mpich2

$ cd mpich2

$ wget https://www.mpich.org/static/downloads/3.4.1/mpich-3.4.1.tar.gz

Até a data desta publicação a versão mais nova é a 3.4.1. Se houver uma mais nova baixe a mais nova. Não baixe as versões com **a b c d** no nome.

$ tar xfz mpich-3.4.1.tar.gz

$ sudo mkdir /home/rpimpi/

$ sudo mkdir /home/rpimpi/mpi-install

$ sudo mkdir /home/pi/mpi-build

$ sudo apt install gfortran

$ sudo /home/pi/mpich2/mpich-3.4.1/configure -prefix=/home/rpimpi/mpi-install

$ sudo make

Essa é a compilação do MPICH. Provavelmente a compilação irá demorar.

$ sudo make install

$ cd ..

$ nano .bashrc

Adicione esta linha no final do arquivo:

PATH=$PATH:/home/rpimpi/mpi-install/bin

Salve com CTRL+O, ENTER, feche com CTRL+X

$ sudo reboot

O MPICH é executado sempre com o binário mpiexec + parâmetro de quantos núcleos de processador você tem e gostaria de usar + comando a ser executado. Assim:

$ mpiexec -n 1 hostname

Ao fim do comando, se deu certo, você verá uma saída com o respectivo *hostname* de cada Raspberry:

Por enquanto, é apenas Localhost. Se isso aqui deu certo, seu MPICH2 está configurado corretamente!

MPI4PY

O MPI4PY é um tradutor de Python para MPICH, permitindo que o MPICH execute scripts Python 2.7 e 3.0+. Tal qual o MPICH anteriormente, o MPI4PY também requer ser compilado para que funcione adequadamente.

Siga exatamente o seguinte, no ClusterSSH aberto, acessando todos os Pi's ao mesmo tempo, digite:

$ wget https://bitbucket.org/mpi4py/mpi4py/downloads/mpi4py-3.0.3.tar.gz

Se houver uma mais nova, baixe a mais nova!

$ tar -zxf mpi4py-3.0.3.tar.gz

$ cd mpi4py-3.0.3

$ sudo apt install python-dev

$ python setup.py build

$ sudo python setup.py install

$ cd ~

$ nano .bashrc

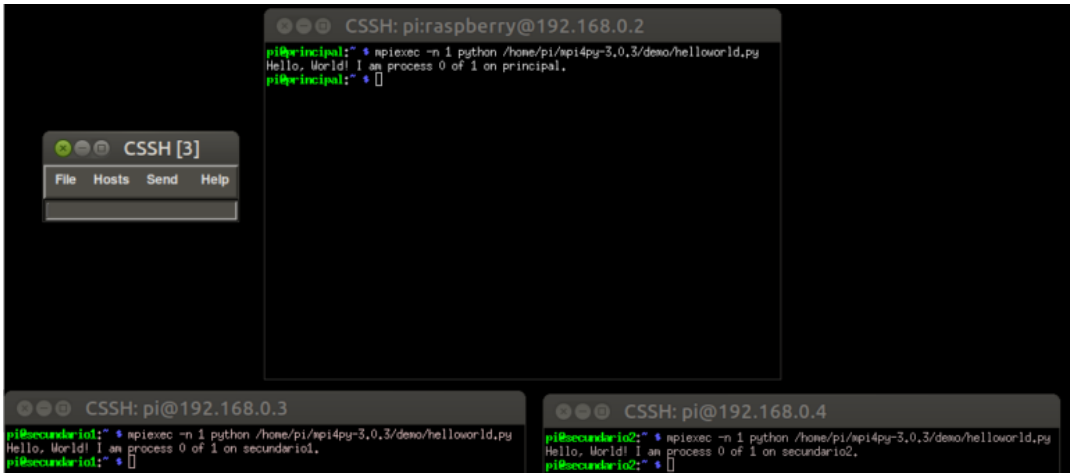
Adicione esta linha no final do arquivo:

export PYTHONPATH=/home/pi/mpi4py-3.0.3

Por fim, teste:

$ mpiexec -n 1 *python* /home/pi/mpi4py-3.0.3/demo/helloworld.py

Se tudo deu certo, após o último comando você verá o script executando corretamente! Observe a adição do comando **python** dentro da sintaxe de comando do mpiexec.



Por enquanto, é apenas Localhost. Se isso aqui deu certo e exibiu uma linha de Hello World em cada Raspberry, seu MPICH2 está configurado corretamente.

O MPICH e o MPI4PY estão compilados e executam bem localmente em todos os nós. Agora, vamos configura-los para se comunicarem

Definindo o ClusterSSH

O protocolo MPICH utiliza o SSH para se comunicar com os nós. Assim, todos os cálculos e comando enviados de um para outro passarão pelo túnel do SSH, o que trará segurança e também bom desempenho ao sistema.
Para que o MPICH opere adequadamente, todos os nós devem ter a chave PublicKey SSH de todos, salva, em authotized_keys. Assim, quando o Principal for contactar o Segundário1 para rodar algum comando, este deverá fazer diretamente, sem que peça senha.
Ainda no ClusterSSH, vamos gerar as chaves públicas pra cada Pi:

$ ssh-keygen

Quando perguntar “Enter file in which to save the key” apenas tecle Enter; quando perguntar “Enter passphrase” também aperte apenas Enter, sem uma senha.

Daqui em diante, os comandos não podem ser feitos no ClusterSSH, porque diferem de Pi para Pi.

A estrutura de comandos será essa, adapte conforme o Raspberry configurado (faça isso individualmente com todos):

No Pi Principal:

$ cd ~/.ssh

$ cp id_rsa.pub Principal

Dê o mesmo nome que o seu hostname. Aqui no exemplo, Principal. Repita para os demais.
Agora que você tem a chave pública com o respectivo nome de cada Pi, transfira essas chaves para dentro de cada Pi. De forma didática, a configuração ficará assim:

Principal < recebe as chaves públicas SSH dos Pi’s Secundário1 e Secundário2

Secundário1 < recebe as chaves públicas SSH dos Pi's Principal e Secundário2

Secundário2 < recebe as chaves públicas SSH dos Pi's Secundário1 e Principal

Transfira entre as máquinas da forma que achar melhor: SFTP, SCP, SMB, FTP, etc.
Deixe elas dentro de:

$ ~/.ssh

Uma vez que você tem as chaves de todas as máquinas em todas as máquinas, adicione elas ao final do arquivo authotized_keys.
No exemplo, dentro do Pi cujo hostname é **Principal** você deve ter as chaves já copiadas, então faça:

$ cd ~/.ssh

$ cat secundario1 >> authorized_keys

$ cat secundario2 >> authorized_keys

Repita esse procedimento seguindo a mesma lógica nos demais Pis.

Machine File

Para se comunicar com os demais Raspberries, todos devem ter o "machinefile", um pequeno arquivo de texto com os IP's das máquinas. No exemplo, eu havia definido os IP's 192.168.0.2, 192.168.0.3 e 192.168.0.4 para meu Cluster. Então:

$ cd ~

$ nano machinefile

Digite de forma direta os IP's, 1 por linha:

192.168.0.2
192.168.0.3
192.168.0.4

Faça conforme sua configuração!
Salve com CTRL+O, Enter, feche com CTRL+X.

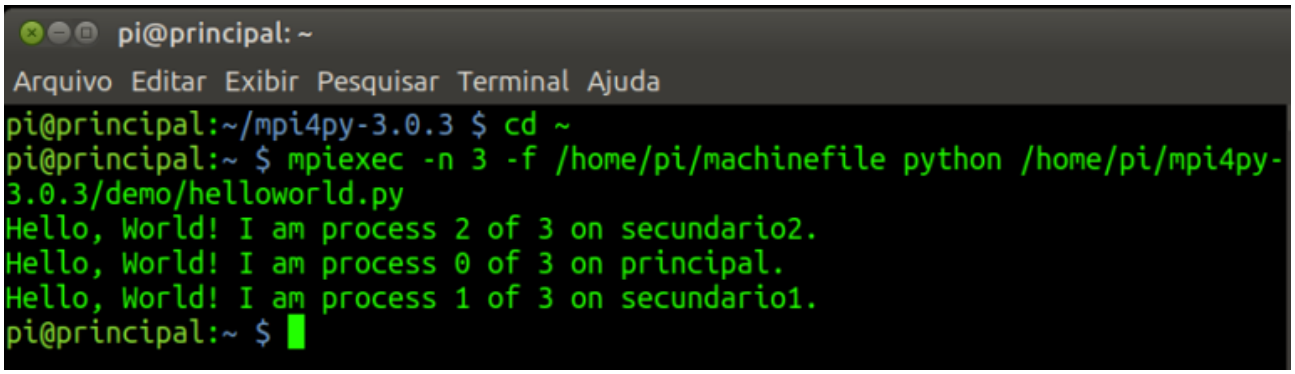
Testando

Aqui você pode acessar o Principal com SSH diretamente. O Hello World no meu exemplo é o seguinte:

$ cd ~

$ mpiexec -n 3 -f /home/pi/machinefile *python* /home/pi/mpi4py-3.0.3/demo/helloworld.py

Se tudo correr bem você será contemplado com a seguinte saída de comando:



Oficialmente o Raspberry “principal” deu um comando HelloWorld para todos os nós inclusive o seu próprio, os Raspberries devolveram o resultado ao Principal, que exibiu tudo no terminal.

Explicando e fazendo observações pertinentes:

-n 3

Número de núcleos de processamento, proporcional ao número de nós.

O MPICH sempre pega 1 núcleo por vez, distribuindo entre todos os nós. Por exemplo, se definir número = 3, ele criará 1 processo pra cada Pi. Se você definir número = 6, serão 2 processos para cada Pi; e se definir número = 7, Os processos serão distribuidos Pi a Pi, até que o primeiro Pi ficará com 1 a mais que os demais. Para usar tudo, potência máxima, no exemplo, use número = 12, assim ele usará 12 núcleos dos 3 processadores quad-core de cada Pi do Cluster.

Essa lógica vale também para qualquer arquitetura e computador usado. Se você fizer um cluster com 2 computadores octa-core, os processos serão distribuídos núcleo a núcleo alternando entre os 2 PC's; Ao fim, número = 16 fará o Cluster operar em potência máxima.

-f machinefile

É o arquivo que define quais os IP's dos Nós.

Do jeito que configuramos tecnicamente você pode usar qualquer Pi como Principal, depende apenas de você e do que desejar realizar. O MPICH não faz nada se o usuário não ordenou e não executa nada em segundo plano até que o comando seja dado manualmente ou programado no Cron.

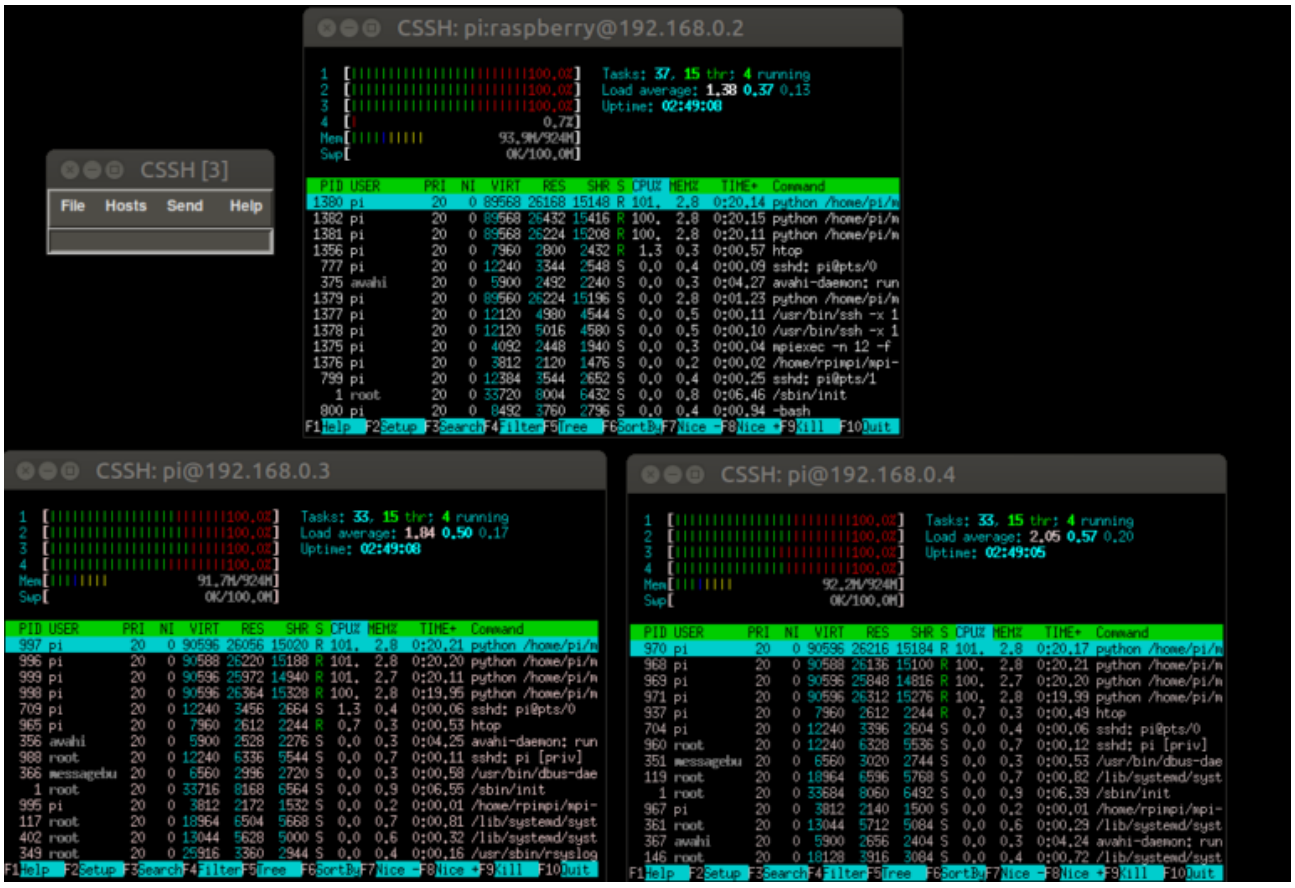
O mpiexec gerou 3 processos, 1 para cada nó e pelo script apontou qual computador fez qual processo. O Principal por sua vez coleta os resultados de todos, exibindo-os para você.
Ou seja: O Secundário2 é meu Pi (3b+ de 1.4 Ghz), mais poderoso que o Principal (3B de 1.2 Ghz) e o Secundário1 (também 3B de 1.2 Ghz). Isso afeta a ordem de exibição dos resultados. O Secundário2 vai responder rapidamente, sendo o primeiro da lista, o Principal será o segundo, porque apesar de lento é o Localhost; e o Secundário1 responderá por último, porque é um Nó na rede e é igualmente lento. – Aqui o delay do Switch vai contar pontos de desempenho.
O fato dos resultados serem exibidos fora de ordem, por ordem de "quem processou e terminou primeiro" é uma prova natural de que o Cluster está operando, dentro dos limites de cada hardware que o compõe.

Exemplo 2:

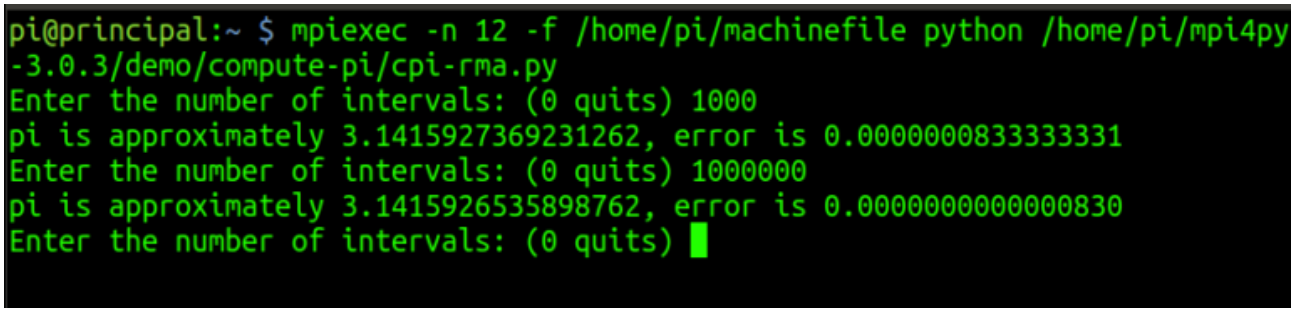
Dentro da pasta Demo tem um cálculo que usa todos os núcleos do Cluster para chegar ao valor mais próximo de Pi.
O comando aqui será dado com -n = 12 para termos máximo desempenho.

$ mpiexec -n 12 -f /home/pi/machinefile python /home/pi/mpi4py-3.0.3/demo/compute-pi/cpi-rma.py

O comando HTOP, com o ClusterSSH, nos permitirá ver que todos os Raspberries estão operando com os 4 núcleos em 100%:



Ao fazer o comando, ele perguntará quantos intervalos você deseja para o cálculo. Quanto maior o valor, mais próximo de PI o resultado será. Considerando que Pi vale 3,14159 26535 89793 23846 26433 83279, qualquer valor próximo disso é interessante.



Observe que com apenas 1000 intervalos, a margem de erro é de 0.0000000833333331 gerando o valor de PI de “3.1415927369231262”, enquanto que com 1000000, a margem de erro é de 0.0000000000000830 e o valor encontrado é “3.1415926535898762”, bem próximo do valor exato.

Os Raspberries não toleram um valor absurdo demais como “1.000.000.000” no Intervalo e vai dar erro de calculo por inanição de recursos. Porém se o Cluster for com mais Nós e/ou com nós mais potentes, tipo Intel Core i3 ou mesmo i5, o resultado será dado satisfatoriamente.
E mais: Se você fizer o mesmo calculo com número = 1 (1 processo de 1 Pi) o resultado vai demorar vários minutos pra sair, coisa que aparece em segundos com número = 12.

Referências bibliográficas

[1.1] https://www.southampton.ac.uk/~sjc/raspberrypi/pi_supercomputer_southampton.htm

https://blog.smartkits.com.br/atualizar-a-raspberry-pi/

[1.2]
https://b2midia.freshdesk.com/support/solutions/articles/1000266606-configurando-ip-fixo-no-raspberry