Instalação e Configuração do MASA-CUDAlign no Amazon Web Services ParallelCluster

Autores: Filipe Maia & Walisson Sousa

Data: Out. 20, 2021

Versão: 1.0

Bio: Information

Sumário

Capítulo 1 Instalação e Configuração do AWS ParallelCluster

O *ParallelCluster* é uma ferramenta que utiliza recursos do *Elastic Computing Cloud (EC2)* e de outros serviços da *Amazon Web Services (AWS)* para criar e administrar *clusters* de alta performance. Os principais recursos que esta ferramenta utiliza são: Instâncias, *Elastic Block Store (EBS)* e *Virtual Private Cloud (VPC)*. Sua estrutura é composta por um mestre e nenhum ou mais nós computacionais (escravos). O mestre é responsável por delegar tarefas aos nós, que têm a função de executá-las.

Este tutorial foi criado utilizando sistema operacional *Linux Ubuntu* na distribuição 18.04 disponível no *Amazon EC2*. Para criação e gerenciamento do cluster, utilizamos a versão 2.11.2 do *AWS ParallelCluster* e usamos o *Simple Linux Utility for Resource Management (SLURM)* versão 20.02.7 para gerenciamento das cargas de trabalho.

1.1 Dependências

Para utilização do ParallelCluster, é necessário primeiramente instalar as dependências a seguir:

1. Instalar o python 3 e pip

```
sudo apt-get install python3
sudo apt-get install python3-pip
```

2. Instalar unzip (caso não esteja instalado)

```
sudo apt install unzip -y
```

1.2 AWS CLI

Deve-se então instalar e configurar o *AWS CLI* (Interface da Linha de Comando: ferramenta para o gerenciamento dos serviços da *AWS*) com as informações do usuário.

1. Baixar o AWS CLI

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

2. Descompactar e instalar o AWS CLI

```
unzip awscliv2.zip
sudo ./aws/install
```

3. Configurar credenciais AWS

Preencher com suas informações de acesso. A região *us-east-1* corresponde ao Leste dos Estados Unidos, mais especificamente, no Norte da Virgínia, porém pode ser escolhida qualquer outra. Em nossos testes, essa foi a região cujas instâncias apresentaram menores preços.

```
aws configure

AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [us-east-1]: us-east-1
Default output format [None]: json
```

1.3 Ambiente Virtual e Instalação do Cluster

Para dar prosseguimento à instalação do *ParallelCluster*, é fundamental utilizar um ambiente virtual, uma vez que este proporciona isolamento entre a máquina do usuário e o terminal utilizado para realizar as configurações do *cluster*. O ambiente virtual adotado neste trabalho foi o *virtualenv*, uma vez que este também é utilizado no *ParallelCluster user guide* da *AWS* [1].

1. Instalar virtualenv

```
python3 -m pip install --upgrade pip
python3 -m pip install --user --upgrade virtualenv
```

2. Criar ambiente virtual

```
python3 -m virtualenv ~/nome_do_ambiente
```

3. Ativar o ambiente virtual criado

```
source ~/nome_do_ambiente/bin/activate
```

4. Instalar AWS ParallelCluster dentro do ambiente virtual criado.

O comando abaixo especifica a versão de instalação do *ParallelCluster*. Neste trabalho foi utilizada a versão 2.11.2.

```
python3 -m pip install --upgrade "aws-parallelcluster<2.11.3"
```

Caso queira instalar a versão mais recente, utilize o comando a seguir:

```
python3 -m pip install --upgrade aws-parallelcluster
```

5. Verificar instalação do ParallelCluster

```
pcluster version
```

1.4 Arquivo de Configuração

Nesta etapa, é criado um arquivo com as configurações básicas do cluster. No entanto, é possível adicionar outras especificações no arquivo gerado. Todas as configurações possíveis estão disponíveis na seção de configuração do AWS ParallelCluster .

1. Configurar o ambiente

```
pcluster configure
```

2. Escolher a região em que o cluster irá residir

```
1. af-south-1
2. ap-east-1
3. ap-northeast-1
4. ap-northeast-2
5. ap-south-1
6. ap-southeast-1
7. ap-southeast-2
8. ca-central-1
```

```
9. eu-central-1
10. eu-north-1
11. eu-south-1
12. eu-west-1
13. eu-west-2
14. eu-west-3
15. me-south-1
16. sa-east-1
17.  us-east-1
18. us-east-2
19. us-west-1
20. us-west-2
```

3. Escolher escalonador

Utilize o *awsbatch* ou *slurm*. Os demais deixarão de ter suporte da AWS ao final de 2021. Para o *MASA-CUDAlign*, deverá ser escolhida a opção 3. *slurm*.

```
1. sge
2. torque
3.  slurm
4. awsbatch
```

4. Selecionar sistema operacional

Em nossos experimentos, utilizamos a opção 5, *ubuntu1804*, porém é passível de funcionamento na distribuição *Linux Ubuntu 20.04* (opção 6).

```
1. alinux
2. alinux2
3. centos7
4. centos8
5.  ubuntu1804
6. ubuntu2004
```

5. Escolher a quantidade de nós computacionais (escravos)

O modelo de execução do *MASA-CUDAlign* requer um número fixo de instâncias. Logo, é necessário selecionar a mesma quantidade de instâncias em ambos os campos de mínimo e máximo:

```
Minimum cluster size (instances) [0]: 8
Maximum cluster size (instances) [10]: 8
```

6. Escolher os tipos de instância do mestre e dos escravos

Utilizamos a instância g4dn.xlarge por apresentar melhor custo-benefício entre as instâncias analisadas em [2], porém há outras famílias de instâncias com configurações distinas, que podem ser encontradas na Seção de instâncias no site da AWS. Observa-se que, por mais que seja permitido criar mestre e escravos com instâncias diferentes, recomenda-se utilizar instâncias do mesmo tipo para a criação de clusters:

```
Master instance type [t2.micro]: g4dn.xlarge
Compute instance type [t2.micro]: g4dn.xlarge
```

7. Escolha ou criação da VPC (Virtual Private Cloud)

Nos testes efetuados, observamos que é preciso criar uma nova VPC a cada configuração de um novo cluster. Isso é necessário porque é associado um IP Elástico à VPC criada. Caso, não seja feito dessa maneira, ao excluir o cluster, os IPs elásticos não são desassociados das respectivas VPCs:

```
Automate VPC creation? (y/n) [n]: y
```

8. Configuração da rede

Nesta etapa, deve-se escolher entre duas opções. A opção 1 cria duas subredes, sendo uma pública apenas para o mestre e outra privada para comunicação entre os nós computacionais. Já a opção 2 cria somente uma subrede pública, na qual os nós computacionais e o mestre são alocados. Realizamos testes com as duas opções e não observamos diferenças na execução do *MASA-CUDAlign*:

```
Allowed values for Network Configuration:

    1. Master in a public subnet and compute fleet in a private subnet
    2. Master and compute fleet in the same public subnet

Network Configuration [Master in a public subnet and compute fleet in a private subnet]: 1

    Beginning VPC creation. Please do not leave the terminal until the creation is finalized
        .
```

Caso a comunicação entre os hosts não funcione, verifique se o tráfego das portas está habilitado, por meio do serviço VPC na console da AWS.

9. criação de subredes automáticas

```
Automate Subnet creation? (y/n) [y]: y
```

Os passos anteriores culminam na criação do arquivo de configuração utilizado pelo cluster, cujo conteúdo refere-se às configurações de um cluster básico. Outras opções podem ser adicionadas nesse arquivo, conforme mostrado no Guia do Usuário do ParallelCluster. Para adicionar ou modificar os parâmetros do arquivo, basta acessá-lo em:

```
\home\nome_do_usuario\.parallelcluster\config
```

Caso seja necessário configurar outro cluster no mesmo ambiente, é preciso excluir o arquivo de configuração do diretório padrão.

1.5 Criação e Conexão ao Cluster

1. Criar o cluster

Esse procedimento leva, aproximadamente, 13 minutos:

```
pcluster create nome_do_cluster
```

2. Utilizar o nome e sua chave de acesso para se conectar ao mestre via SSH:

```
pcluster ssh nome_do_cluster -i sua-chave.pem
```

Percorridos tais passos, o cluster encontra-se pronto para utilização. É necessário, agora, realizar a instalação e configuração do MASA-CUDAling no ambiente.

Capítulo 2 Instalação e Configuração do MASA-CUDAlign no AWS ParallelCluster

2.1 Instalação do MASA-CUDAlign no Cluster

Para instalar e configurar o *MASA-CUDAlign* é necessário transferir os arquivos do usuário para o mestre:

1. Fazer o *download* do *MASA-CUDAlign*

Uma versão encontra-se disponível no Github. Nos testes, utilizamos a static-multibp.zip.

2. Abrir uma outra janela do terminal, pela qual os arquivos serão transferidos

3. Alterar as permissões do arquivo para possibilitar a sua transferência:

```
chmod 777 static-multibp.zip
```

4. Utilizar o Secure Copy Protocol (SCP) para realizar a transferência de arquivos

Para tanto são necessários: a chave do usuário, nome do arquivo a ser transferido, o nome do mestre e do respectivo endereço DNS IPV4 público:

```
scp -i caminho_da_chave/suachave.pem caminho_do_arquivo/static-multibp.zip
    ubuntu@nome_do_comp_destino.compute-1.amazonaws.com:
```

2.2 Configuração do MASA-CUDAlign

1. Descompactar arquivos do *MASA-CUDAlign*

```
unzip static-multibp.zip
```

2. Acessar a pasta que contém os arquivos de configuração da ferramenta

```
cd multibp
```

3. Conceder permissões ao arquivo *configure* para possibilitar a configuração das variáveis de ambiente

```
chmod 777 configure
```

4. Configurar compilação com a arquitetura da placa gráfica utilizada

A instância utilizada nos testes, g4dn.xlarge, é equipada com GPU *Tesla T4* e possui arquitetura *Turing (sm_75)*. Para obter o melhor desempenho, é necessário compilar o *MASA-CUDAlign* com a arquitetura correspondente:

```
./configure --with-cuda-arch=sm_75
make
```

Caso utilize outro modelo de GPU, consulte o guia que informa a arquitetura de algumas placas gráficas da NVIDIA.

5. Visualizar a lista de GPUs identificadas pela ferramenta MASA:

```
./cudalign --list-gpus
```

6. Obter e descompactar pasta contendo as sequências

O *MASA-CUDAlign* recebe dois arquivos .fasta como entrada para realizar a comparação. Neste link, é possível obter algumas das sequências utilizadas em nossos experimentos. A Tabela 2.1 apresenta estas sequências, juntamente com os tamanhos e as comparações realizadas.

Tabela 2.1: Sequências genômicas utilizadas nos testes

Comp.	Acesso	Nome	Tamanho	Acesso	Nome	Tamanho
3m	BA000035.2	*Corynebacterium efficiens YS-314*	3.147.090	BX927147.1	*Corynebacterium glutamicum ATCC 13032*	3.282.708
5m	AE016879.1	*Bacillus anthracis str. Ames*	5.227.293	AE017225.1	*Bacillus anthracis str. Sterne*	5.228.663
7m	NC_003997.3	*Bacillus anthracis str. Ames*	5.227.293	NC_005027.1	*Rhodopirellula baltica SH 1 chromosome*	7.145.576
10m	NC_014318.1	*Amycolatopsis mediterranei U32 chromosome*	10.236.715	NC_017186.1	*Amycolatopsis mediterranei S699 chromosome*	10.236.779
23m	NT_033779.4	*Drosophila melanogaster chromosome 2L*	23.011.544	NT_037436.3	*Drosophila melanogaster chromosome 3L*	24.543.557
47m	BA000046.3	*Pan troglodytes DNA chromosome 22*	32.799.110	NC_000021.7	*Homo sapiens chromosome 21*	46.944.323
chr17	NC_000017.11	*Homo sapiens chromosome 17*	83.257.441	NC_006484.4	*Pan troglodytes isolate Yerkes chimp pedigree chromosome 17*	83.181.570
chr21	NC_000021.9	*Homo sapiens chromosome 21*	46.709.983	NC_006488.4	*Pan troglodytes chromosome 21*	33.445.071
chr22	NC_000022.11	*Homo sapiens chromosome 22*	50.818.468	NC_006489.4	*Pan troglodytes chromosome 22*	37.823.149
chrY	NC_000024.10	*Homo sapiens chromosome Y*	57.227.415	NC_006492.4	*Pan troglodytes chromosome Y*	26.350.515

7. Fazer download das sequências

```
git clone https://github.com/walissongpi/sequences
```

8. Descompactar as sequências

```
cd sequences
unzip 1-3M.zip
unzip 2-5M.zip
unzip 3-7M.zip
unzip 4-10M.zip
unzip 5-23M.zip
unzip 6-47M.zip
unzip chr21.zip
unzip chr22.zip
unzip chrY.zip
```

9. Adicionar o *MASA-CUDAlign* ao PATH para permitir a execução em qualquer diretório

```
export PATH=$PATH:/home/ubuntu/multibp
```

Para gerenciamento das cargas de trabalho no cluster, utilizamos o Simple Linux Utility for Resource Management (SLURM). No AWS Parallel Cluster, o SLURM já vem instalado e configurado, necessitando apenas submeter as execuções. A maioria dos tutoriais encontrada mostra exemplos de utilização do SLURM através da biblioteca MPI. No entanto, o *MASA-CUDAlign* realiza a comunicação entre os nós computacionais via *socket*. Assim, utiliza-se um *script* para realizar a ligação entre os computadores do cluster.

2.3 Scripts para Comunicação entre os Hosts do ParallelCluster

Utilizamos 2 *scripts* para submissão das execuções, escritos por Figueiredo et al.[3]. No primeiro, descrevemos as configurações utilizadas no processo, conforme apresentado na Figura 2.1 a seguir:

```
1  #!/bin/bash
2  #
3  #SBATCH --job-name=cudalign2
4  #SBATCH --ntasks=2
5  #SBATCH --nodes=2
6  #SBATCH --time=09:59:00
7  #
8
9  gpus=1
10
11 srun scriptslurm.sh $1 $gpus
12
```

Figura 2.1: script

Os comandos apresentados na Figura 2.1 são explicados na lista abaixo:

- **#SBATCH –job-name=cudalign2:** Atribui um nome ao processo;
- **#SBATCH –ntasks=2:** Define a quantidade de tarefas que serão designadas para executar o job referente ao script submetido;
- **#SBATCH –nodes=2:** Define a quantidade de nós computacionais que serão utilizados na execução;
- **#SBATCH –time=09:59:00:** Tempo para finalização da tarefa;
- **#gpus=1:** Define a quantidade de GPUs por instância;
- **srun scriptslurm.sh $1 $gpus:** Submete o script scriptslurm para execução. Também é necessário informar dois parâmetros, que neste caso são: nome que designa as sequências a serem comparadas e quantidade de GPUs por instância.

O segundo arquivo, o *scriptslurm*, contém os comandos necessários para interconexão e execução do *MASA-CUDAlign* em mais de um *host*. Realizamos alterações neste *script* de maneira a permitir a execução em computadores equipados com somente uma GPU. Ademais, é necessário modificar no script os seguintes parâmetros (Figura 2.2):

```
17 baseport=2000        # Porta base
18 sol=multibp
19 instancia=g4dnxlarge
20 nos=4
```

Figura 2.2: porta

- ***baseport***: porta inicial utilizada para comunicação via socket. Ela é incrementada de acordo com a quantidade de nós computacionais.
- ***sol***:diretório da solução do *MASA-CUDAlign* escolhida.
- ***instancia***: nome da instância AWS utilizada no *cluster*. Essa variável é utilizada para encontrar os nomes dos *hosts* de determinado tipo.
- ***nos***: quantidade de nós computacionais utilizados no *cluster*, excluindo o mestre.

Outra alteração necessária foi a maneira como obter os nomes ou ips das instâncias do cluster. Como a atribuição dos nomes e distribuição dos endereços IPs dos *hosts* é realizada automaticamente, utilizamos a função *scontrol* do *SLURM* para retornar a lista de instâncias ativas no cluster. Note que o comando *scontrol* necessita especificar o tipo de instância utilizada. Para tanto, criamos a variável instancia, que armazena uma *string* contendo o tipo de instância adotada no cluster.

```
22 # Função que recupera os nomes dos hosts
23 function getips ()
24 {
25     nodelist=`scontrol show hostnames compute-st-$instancia-[1-$nos]`
26     nodeips=$nodelist
27     echo $nodeips
28 }
```

Figura 2.3: Função que retorna o nome dos hosts do cluster. Também pode ser utilizado comando para retornar os endereços IPs. Alguns comandos úteis são apresentados na seção 2.4

No *AWS Parallel Cluster*, os nomes dos *hosts* são atribuídos utilizando o nome da instância e uma numeração. Dessa maneira, ao iniciar 8 nós computacionais do tipo g4dn.xlarge, os nomes atribuídos serão os seguintes:

```
compute-st-g4dnxlarge-1
compute-st-g4dnxlarge-2
compute-st-g4dnxlarge-3
compute-st-g4dnxlarge-4
compute-st-g4dnxlarge-5
compute-st-g4dnxlarge-6
compute-st-g4dnxlarge-7
compute-st-g4dnxlarge-8
```

O *scriptslurm* utiliza o nome ou ip das instâncias para comunicação, sendo necessário identificá-las no script. De maneira análoga, os nós computacionais ativos do tipo t2.micro, terão o nome iniciado em compute-st-t2micro-.

1. Executar o *Static-MultiBP*

Para tanto é necessário inserir um *job* na fila do escalonador. Este *job* é associado a um *script* que estabelece a comunicação entre as GPUs e inicia a execução da ferramenta. É necessário informar na linha de comando dois parâmetros de execução: o nome do *script batch* e o identificador da comparação.

```
sbatch  nome_arquivo parâmetros
ex.:
sbatch sbatch-nvidia-2 1-3m
```

2.4 Comandos úteis

Separamos alguns comandos que podem ser úteis na identificação dos nomes e endereços IPs dos *hosts*. São eles:

- **descobrir o ip do nó computacional:**

```
echo `hostname -i`
```

- **descobrir o nome do nó computacional:**

```
echo `hostname`
```

- **apresentar a lista de nós computacionais ativos e inativos:**

```
scontrol show hostnames
```

- **apresentar todos os nós computacionais ativos:**

```
echo `scontrol show hostnames compute-st-g4dnxlarge-[1-8]`
```

- **retornar endereço ip do nó através do nome:**

```
sinfo -n compute-st-g4dnxlarge-1 -o %o -h
```

- **retornar lista dos ips das instâncias ativas:**

```
sinfo compute-st-g4dnxlarge-[1-8] -o %o -h
sinfo -n compute-st-g4dnxlarge-[1-8] -o %o -h
```

Bibliografia

[1] AWS. *AWS ParallelCluster User Guide*. https://docs.aws.amazon.com/parallelcluster/latest/ug/configuration.html/. Acesso em 22/08/2021.

[2] Rafaela C Brum et al. "A Fault Tolerant and Deadline Constrained Sequence Alignment Application on Cloud-Based Spot GPU Instances". Em: *European Conference on Parallel Processing*. Springer. 2021, pp. 317–333.

[3] Marco Figueiredo et al. "Parallel Fine-Grained Comparison of Long DNA Sequences in Homogeneous and Heterogeneous GPU Platforms With Pruning". Em: *IEEE Transactions on Parallel and Distributed Systems* 32.12 (2021), pp. 3053–3065. DOI: 10.1109/TPDS.2021.3084069.