# Capítulo 6

# Ferramentas para Configuração e Gerenciamento de Cluster de Alto Desempenho em Nuvem Pública

Albino A. Aveleda (UFRJ) e Alvaro L.G.A. Coutinho (UFRJ)

***Abstract***

*The High-Performance Computing Center (NACAD) of COPPE / UFRJ is a laboratory specialized in applying high-performance computing to engineering and science problems in general. NACAD also has extensive experience in the administration, management, and tools development to support the Supercomputing Center and to develop and implement innovations in the machine management environment. The mini-course proposes sharing some of the best practices adopted by NACAD-COPPE/UFRJ for deploying an HPC cluster using a public cloud.*

***Resumo***

*O Núcleo Avançado de Computação de Alto Desempenho (NACAD) da COPPE/UFRJ é um laboratório especializado na aplicação de computação de alto desempenho a problemas de engenharia e ciências em geral. O NACAD também possui grande experiência na administração, gerencia e ferramentas de apoio ao Centro de Supercomputação e de desenvolver e implementar inovações no ambiente de administração e gerencia da máquina. O presente minicurso propõe compartilhar algumas dessas melhores práticas adotadas pelo NACAD-COPPE/UFRJ para a implantação de um cluster de Alto Desempenho usando uma nuvem pública.*

***Palavras-chave****: cluster, computação de alto desempenho, nuvem pública, computação em nuvem*

## 6.1. Introdução

O presente minicurso tem a proposta de compartilhar melhores práticas de como utilizar uma ferramenta para fazer a implantação de um cluster de Alto Desempenho usando uma Nuvem Pública, utilizando a ferramenta de gerenciamento de cluster de código aberto conhecida como *AWS Parallel Cluster*.

Apesar da ideia de computação em nuvem ser conhecida há décadas, ela de fato começou a ser oferecida comercialmente entre 2006 e 2008 através da *Amazon* quando lançou o seu produto *Amazon Web Service* (AWS). Aos poucos, ela foi se popularizando através de vários provedores, tais como: AWS, *Microsoft Azure*, *Google Cloud Platform* (GCP), *IBM Cloud*, *Oracle Cloud*, *Alibaba Cloud* e muitos outras ao redor do mundo. Os maiores provedores de nuvem pública possuem um alcance mundial. Basicamente, eles são divididos por regiões e zonas de disponibilidades. Por exemplo, na AWS pode-se citar algumas regiões, tais como: Oregon, São Paulo, Irlanda, Tóquio, etc. Sendo que cada região possui em média três zonas de disponibilidades. Cada zona de disponibilidade é um datacentro e tem pelo menos 100 Km de distância entre elas. Hoje em dia, grande parte da população mundial usa algum serviço na nuvem.

No caso particular do uso da nuvem para aplicações de computação de alto desempenho (HPC, do inglês, *High-Performance Computing*), o processo foi um pouco diferente. Desde o início houve uma certa resistência em se utilizar uma nuvem pública para fazer o processamento de HPC, já que as aplicações normalmente possuem requisitos bem específicos tais como: necessidade de um hardware de alto desempenho, várias conexões e topologias de rede, alta banda e baixa latência de comunicação, alta taxa de leitura e escrita dos dados, etc. Além disso, a configuração de cada nó de processamento é otimizada e configurada com diversas bibliotecas otimizadas. Consequentemente, todos esses requisitos dificultavam o uso da nuvem. Aos poucos esses itens foram sendo endereçados por alguns provedores, facilitando assim o uso de uma nuvem pública.

Em relação ao hardware, vários provedores fizeram investimentos a fim de atender a esta demanda. Seguem abaixo alguns exemplos:

- AWS *Elastic Fabric Adapter* (EFA) é uma interface de rede que pode ser conectada a algumas instâncias. Sua interface de hardware faz um desvio do sistema operacional (SO) e melhora o desempenho das comunicações entre instâncias. Com o EFA, os aplicativos de HPC que usam o MPI (*Message Passing Interface*) e os aplicativos de aprendizado de máquina (ML) usando NCLL (*NVIDIA Collective Communications Library*) podem ser dimensionados para milhares de CPUs ou GPUs.
- *Amazon FSx for Lustre* é o serviço totalmente gerenciado e compartilhado com escalabilidade e performance do sistema de arquivo paralelo Lustre [3], um dos mais comuns em ambientes de HPC.
- A *Microsoft Azure* possui algumas instâncias com acesso à rede *Infiniband* que possui alta largura de banda e baixa latência. A Microsoft e a HPE-Cray, possuem uma parceria que onde é possível disponibilizar um cluster da HPE-Cray dentro da infraestrutura da Microsoft Azure.

Além do hardware e bibliotecas otimizadas, não haviam ferramentas que auxiliassem a implementação de um cluster na nuvem. Uma das primeiras ferramentas neste sentido foi o *StarCluster* [ *1* ] desenvolvido pelo MIT (*Massachusetts Institute of*

*Technology*). O desenvolvimento do *StarCluster* parou depois de 2014. Logo depois a AWS lançou o seu software que era chamado de *CfnCluster*. Posteriormente, ele foi migrado para o *Parallel Cluster* [2], que será abordado neste minicurso em mais detalhes. Pode-se citar outro exemplo, o *Google HPC Toolkit*, que permite iniciar *jobs* rapidamente com desempenho previsível através de VMs (*virtual machines*) de HPC pré-configuradas.

Este minicurso foca no uso de ferramentas para implantar e gerenciar clusters de HPC na Nuvem AWS (*Amazon Web Services*). O cluster de HPC na nuvem tem uma estrutura semelhante à maioria dos clusters tradicionais, isto é, uma instância fazendo o papel de *master node* (com sistema de fila para receber solicitações de *jobs* e outras configurações) e as demais instâncias atuando como *compute node,* que são nós de processamento alocados de acordo com a demanda ou de acordo com o que foi definido na configuração. Essas ferramentas permitem usufruir dos recursos e das melhores características da nuvem, como por exemplo, alocação dinâmica da quantidade de instâncias de forma elástica à carga de trabalho do cluster, permitindo assim que o cluster aumente e diminua automaticamente o número de instâncias computacionais.

Sendo assim, de forma a conduzir esta exposição da forma mais clara possível, este texto está organizado da seguinte forma:

- A seção 7.2 apresenta uma introdução a computação em nuvem fazendo uma comparação entre um cluster tradicional versus um cluster numa nuvem e suas respectivas ferramentas;
- A seção 7.3 aborda as principais características dos serviços utilizados na nuvem para uma implementação de um cluster de HPC (VPC, instâncias, armazenamento, autenticação, etc.);
- A seção 7.4 mostra o uma visão geral da configuração de um cluster de forma manual ou com uso de ferramentas para implementação do cluster de HPC na nuvem pública;
- A seção 7.5 discute alguns exemplos de implementação do cluster de HPC na nuvem pública;
- A seção 7.6 faz as considerações finais.

## 6.2. Cluster tradicional versus Cluster na nuvem

### 6.2.1. Cluster Tradicional

Existem diversos desenhos de topologias possíveis e de como provisionar um cluster de HPC. A Figura 1 mostra, de forma simplificada, um cluster tradicional de HPC. Nesta figura pode-se ver um cluster formado por um *headnode*, responsável pela gerencia e acesso ao cluster, *nodes* que são os nós de processamento, e switches para a comunicação intra-cluster. Nesta figura não foi incluído um ou mais servidores de armazenamento a fim de simplificar o desenho. A área de armazenamento do cluster pode ser feita no próprio *headnode*, caso seja um cluster pequeno, ou de um ou mais servidores de armazenamento externo, e que normalmente são conectados na rede de alto desempenho através do switch HPC.

Neste exemplo, o *headnode* exerce algumas funções tais como: servidor de acesso e gerenciamento do cluster, *job scheduler*, entre outras. Entretanto, dependendo do tamanho do cluster, várias dessas funções podem ser distribuídas por vários servidores. Por exemplo, o *headnode* pode fazer o papel de gerenciamento e de *job scheduler*, enquanto um ou mais servidores podem servir como servidor de acesso aos usuários para compilação e submissão de *jobs* (tarefas).

Nesta topologia, o *headnode* se conecta à Internet e ao switch interno do cluster para se comunicar com os nós computacionais a fim de alocar ou monitorar os recursos disponíveis no cluster. Em alguns clusters também existem uma ou mais redes de alto desempenho, que possuem uma maior banda e uma menor latência de comunicação, como ilustrado na Figura 1 através do Switch HPC. Por exemplo, switches *Infiniband*, *Slingshot*, etc. Normalmente essa rede é utilizada para comunicações entre nós de processamento, tais como: MPI (*Message Passing Interface*), acesso ao servidor de armazenamento, etc.

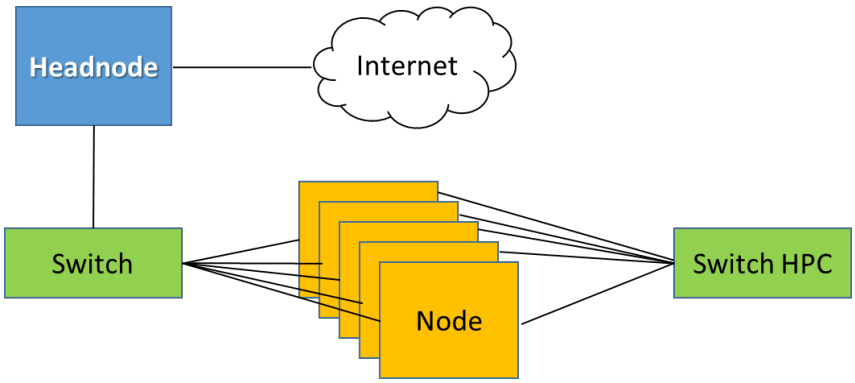


*Figura 1:* Topologia simplificada de um cluster de HPC

### 6.2.2. Cluster na Nuvem

Um cluster em uma nuvem pública também pode utilizar uma topologia semelhante à mostrada na Figura 1, porém a nuvem possui algumas peculiaridades que a tornam diferente de um cluster tradicional. Enquanto que um cluster tradicional possui seu custo de aquisição, manutenção e operação, no cluster na nuvem você paga pelo que consome e pode provisionar a qualquer momento. A Tabela 1 mostra as principais diferenças entre um cluster tradicional e um cluster em uma nuvem pública.

| | Cluster tradicional | Cluster na nuvem |
|---|---|---|
| **Quantidade de nós de processamento** | Fixo | Variável |
| **Responsabilidade sobre segurança** | Total | Compartilhada |
| **Redes** | LANs e VLANs | VPCs |
| **Rede de alto desempenho** | Pode ter ou não | Depende do provedor |
| **Armazenamento (capacidade)** | Fixo | Variável |
| **Custo** | Parcela fixa + variável (consumo energético) | Paga apenas pelo que consome |

*Tabela 1: Resumo das principais diferenças entre um cluster tradicional e um cluster na nuvem pública*

Em geral, os provedores possuem um limite em função do tipo de instância e da quantidade que pode ser alocada para evitar supressas na conta do usuário. No caso da AWS, esse limite costuma ser 20 instâncias, dependendo da capacidade de processamento com base no número de unidades de processamento central virtuais (vCPUs). Entretanto, esse limite pode ser aumentado facilmente, bastando fazer uma solicitação ao suporte da AWS.

Para exemplificar a diferença entre os clusters, suponha que para resolver um problema é necessário fazer 8 execuções de um programa MPI que utiliza 5 nós de processamento em cada rodada. Sendo que cada execução demora em média 1 hora e meia. Por parte do usuário, ele simplesmente submete os 8 jobs MPI na fila do cluster.

- Usando um cluster tradicional com 10 nós de processamento: Neste caso, serão executados dois *jobs* ao mesmo tempo. Logo, para fazer a execução dos 8 casos seriam necessárias 6 horas de processamento (8 jobs /2 jobs por vez x 1,5h=6h).
- Usando um cluster na nuvem: como o cluster pode alocar dinamicamente os nós de processamento (instâncias), pode-se executar os 8 *jobs* ao mesmo tempo e ter o resultado em apenas 1,5 hora. Em outras palavras, pode-se alocar 40 instâncias (8 jobs x 5 instâncias) de uma vez. Em relação ao custo na nuvem é indiferente usar 10 instâncias por 6 horas ou usar 40 instâncias por 1,5 hora, já que atualmente a unidade de consumo é medida em segundos.

Cabe aqui lembrar que no início da AWS o cálculo não era tão simples assim. A unidade de custo de cada instância era por hora o que complicava um pouco mais a estimativa de custo. Na época, o *StarCluster* também endereçava esse problema de otimização de custos. Para facilitar o entendimento, suponha o mesmo problema anterior e que o custo da instância fosse $0,10/h. Ao usar as mesmas 10 instâncias fixas, semelhante ao cluster tradicional, se poderia ter uma vantagem em relação a alocação de todas as instâncias. Neste caso, usando 10 instâncias durante 6 horas o custo seria de $6,00. Porém, ao usar 40 instâncias por 1,5 hora, o custo seria de 40 instâncias por 2 horas, já que a unidade mínima era de 1 hora, o que daria $8,00 de custo. A evolução do tempo e concorrência fizeram que a AWS mudasse a unidade de cobrança, beneficiando assim seus clientes.

## 6.3. Serviços utilizados no AWS *ParallelCluster*

Atualmente existem centenas de serviços disponíveis em uma nuvem pública e no caso particular da AWS, além de ser a mais adotada no mundo, é a que oferece mais serviços aos seus clientes. O *AWS ParallelCluster* em si é gratuito, o que é cobrado são os recursos alocados por ele. Em particular o *AWS ParallelCluster* [ *2 ]* utiliza ou pode utilizar os seguintes serviços descritos abaixo em ordem alfabética.

- *Batch*
  Serviço gerenciado de *job scheduler*, função semelhante ao SLURM (*Simple Linux Utility for Resource Management*). Ele provisiona dinamicamente a quantidade e o tipo de recursos de computação (por exemplo, CPU, GPU, instâncias otimizadas para memória, etc.). Com *AWS Batch* não é necessário instalar ou gerenciar softwares de computação em lote, como o SLURM, ou clusters de servidor adicionais para executar seus trabalhos de maneira eficaz.

- *CloudFormation*
  Serviço de infraestrutura como código que permite provisionar recursos na AWS e de terceiros no ambiente de nuvem. É o principal serviço usado pelo *ParallelCluster*. Cada cluster é representado como uma pilha e todos os recursos necessários para cada cluster são definidos dentro do *ParallelCluster template*. As instâncias que são executadas em um cluster fazem chamadas de HTTPS para o *CloudFormation endpoint* na região onde o cluster é alocado.

- *CloudWatch*
  Serviço de monitoramento e observabilidade que fornece dados e insights acionáveis. Esses insights podem ser usados para monitorar os aplicativos, responder a alterações de desempenho e exceções de serviço, e otimizar a utilização de recursos. O *CloudWatch* é usado como um painel, para monitorar e registrar em logs as etapas de criação da imagem em *Docker* e a saída dos *jobs* da *AWS Batch*.

- *CloudWatch Logs*
  É um dos principais recursos do *CloudWatch*. Utilizado para monitorar, armazenar, exibir e pesquisar os arquivos de log de muitos dos componentes usados pelo *ParallelCluster*.

- *CodeBuild*
  Serviço de integração contínuo gerenciado que compila o código-fonte, executa testes e produz pacotes que estão prontos para implantação. O *CodeBuild* é usado para criar imagens do *Docker* de forma automática e transparente quando os clusters são criados e é usado somente com *AWS Batch*.

- *DynamoDB*
  Serviço de banco de dados *NoSQL* rápido e flexível. Ele é usado para armazenar as informações mínimas do estado do cluster. O *headnode* principal rastreia instâncias provisionadas em uma tabela do *DynamoDB*.

- *Elastic Compute Cloud* (EC2)
  Serviço que oferece a capacidade de computação para o cluster. O nó *headnode* e os nós de computação são instâncias do EC2. Qualquer tipo de instância que ofereça suporte ao HVM pode ser selecionada. O *headnode* e os nós de computação podem ser de diferentes tipos de instância. Entretanto, cabe aqui uma observação, caso o *headnode* também faça o serviço de NFS (*Network File System)* para os nós de processamento, deve-se atentar para verificar se a instância selecionada também possui a mesma banda de comunicação de rede. Dependendo do tamanho da instância, ela pode ter uma banda de rede diferente. Além disso, se forem usadas várias filas, alguns ou todos os nós de computação também poderão ser executados como uma instância *spot*. As instâncias *spot* do EC2 permitem aproveitar a capacidade não utilizada do EC2 na nuvem AWS. Em comparação com a definição de preço sob demanda, as instâncias *spot* oferecem descontos de até 90%.

- *Elastic Container Registry* (ECR)
  Serviço de registro de contêiner do *Docker* totalmente gerenciado que facilita o armazenamento, o gerenciamento e a implantação de imagens de contêiner do *Docker*. O ECR armazena as imagens do *Docker* que são criadas quando os clusters são criados. As imagens do *Docker* são então usadas pelo *AWS Batch* para executar os contêineres para os trabalhos enviados. O ECR é usado somente com o *AWS Batch*.

- *Identity and Access Management* (IAM)
  Serviço usado para fornecer um papel do IAM menos privilegiado para as instâncias EC2 de cada cluster. As instâncias recebem acesso apenas às chamadas de API específicas que são necessárias para implantar e gerenciar o cluster. As funções do IAM também são criadas para os componentes envolvidos no processo de criação de imagem do *Docker* quando os clusters são criados. Esses componentes incluem as funções do *Lambda* que têm permissão para adicionar imagens do *Docker* ao repositório do ECR. Eles também incluem as funções que permitem excluir o *bucket* do S3 que é criado para o cluster e o projeto *CodeBuild*. Também há funções para recursos, instâncias e tarefas do *AWS Batch*.

- *Lambda*
  Serviço de computação sem servidor e orientado a eventos que permite executar as funções que orquestram a criação de imagens do *Docker*. O *Lambda* também gerencia a limpeza de recursos de cluster personalizado, como imagens do *Docker* armazenadas no repositório do ECR e no S3.

- NICE DCV
  O NICE DCV é um protocolo de exibição remoto de alto desempenho que fornece uma maneira segura de fornecer desktops remotos e streaming de aplicativos para qualquer dispositivo em diferentes condições de rede.

- *Route 53*

Serviço de *Domain Name System* (DNS) na nuvem altamente disponível e escalável. Ele é usado para criar zonas hospedadas com *hostnames* e *fully qualified domain* para cada um dos nós de computação.

- *Virtual Private Cloud* (VPC)
  É um serviço que permite criar uma nuvem privada localizada dentro de uma nuvem pública. A VPC define uma rede usada pelos nós do cluster.

Armazenamento:

Existem quatro tipos de armazenamento, que podem ser usados no *ParallelCluster*, que estão descritos a seguir.

- *Elastic Block Store* (EBS)
  Serviço que disponibiliza volumes de armazenamento persistente em blocos de alto desempenho projetado para o EC2, isto é, normalmente utilizado para o sistema operacional e outras aplicações.
- *Elastic File System* (EFS)
  Serviço que oferece um sistema de arquivos NFS elástico, simples, escalável e totalmente gerenciado para uso na nuvem.
- *FSx for Lustre*
  O FSx for Lustre fornece um sistema de arquivos distribuído, paralelo e de alto desempenho que usa o sistema de arquivos Lustre [ 3 ] de código aberto.
- *Simple Storage Service* (S3)
  Serviço de armazenamento de objetos que oferece escalabilidade, disponibilidade de dados, segurança e performance. Com classes de armazenamento econômicas e recursos de gerenciamento fáceis de usar, pode-se otimizar custos, organizar dados e configurar controles de acesso ajustados para atender a requisitos específicos de negócios, organizacionais e de conformidade.

Serviços que foram removidos a partir da versão 2.11.5 do *ParallelCluster*:

- *Auto Scaling*
  Serviço que monitora os aplicativos e ajusta automaticamente a capacidade para manter um desempenho constante e previsível pelo menor custo possível.
- *Simple Notification Service* (SNS)
  Serviço de mensagens totalmente gerenciado para a comunicação de aplicação para aplicação (A2A) e de aplicação para pessoa (A2P).
- *Simple Queue Service* (SQS)
  É um serviço de filas de mensagens gerenciado que permite o desacoplamento e a escalabilidade de micros serviços, sistemas distribuídos e aplicações sem servidor.

## 6.4. Configuração do cluster

### 6.4.1. Visão Geral

A implementação de forma manual de um cluster, apesar de ser possível, é trabalhosa e podem ocorrer em alguns erros durante o processo da implantação. No caso de se usar uma infraestrutura na nuvem, mesmo sem um software de gerenciamento de cluster (o que acontece em alguns provedores), pode-se otimizar e minimizar a possibilidade de erros durante a implantação. Basta usar a infraestrutura como código (IaC - *Infrastructure as Code*), que é um dos recursos disponíveis na nuvem, e ferramentas DevOps (*Development and Operations*).

Além das ferramentas de infraestrutura fornecida pelos próprios provedores tais como: *AWS CloudFormation*, *Azure Resource Manager*, *Google Cloud Deployment Manager*, etc. Existem diversas outras ferramentas que podem ser usadas em clusters na nuvem ou *on-premise*. A seguir são listadas algumas delas.

- Terraform [ 4 ]: É uma das ferramentas de IaC mais populares do mercado. É um projeto de código aberto com flexibilidade, suportando as principais plataformas de nuvem, incluindo; AWS, GCP, Azure, DigitalOcean, GitHub, Cloudflare, OpenStack e muitos outros. Além disso, o Terraform também permite a manipulação de recursos por meio do controle de origem. Esse recurso é essencial ao manipular nuvens híbridas, onde os planos podem ser feitos em vários provedores e infraestruturas de nuvem, tudo isso usando o mesmo fluxo de trabalho.
- Pulumi [ 5 ]: É uma ferramenta de IaC que se diferencia das demais plataformas por oferecer maior flexibilidade. Ele suporta várias linguagens de programação, como Python, JavaScript, C#, Go e TypeScript. Não possui a mesma portabilidade que o Terraform, porém oferece suporte para os principais provedores da nuvem, como: AWS, GCP e Azure.
- Ansible [ 6 ]: É uma ferramenta de orquestração e configuração da Red Hat. O *Ansible* usa módulos de configuração chamados "*Playbooks*" escritos em YAML (*YAML Ain't Markup Language*), onde você pode configurar o estado final desejado de sua infraestrutura. O *Ansible* melhora o desenvolvimento automatizando muitas tarefas repetitivas e complexas, economizando muito tempo ao instalar pacotes ou configurar um grande número de servidores.
- Chef [ 7 ]: É uma das ferramentas de infraestrutura como código mais populares no mercado. Esta ferramenta usa "*recipes*" e " *cookbooks*" contando com uma linguagem específica de domínio (DSL) baseada em Ruby. O *Chef* é independente da nuvem, trabalhando com grandes provedores de nuvem, como AWS, GCP e Azure Cloud e também suporta APIs de provisionamento permitindo ser usada em conjunto com o *Terraform*. Sua flexibilidade é escalável e aplicável em qualquer pipeline de CI/CD (*Continuous Integration/Continuous Delivery*) existente.
- Puppet [ 8 ]: O Puppet tem muitas semelhanças com o Chef e faz parte da base de muitos pipelines de CI/CD criados por engenheiros de DevOps. Ele também usa uma DSL baseada em Ruby, onde você pode declarar o estado final de sua infraestrutura e o que deseja que ela faça. Se ocorrer algum desvio de configuração

após esse ponto, o Puppet monitora e corrige automaticamente quaisquer alterações incorretas. Atualmente, esse projeto de código aberto oferece suporte a todas as plataformas de nuvem proeminentes, como AWS, GCP, Azure Cloud, permitindo a automação em vários provedores.

- Crossplane [ 9 ]: É uma ferramenta de código aberto *Kubernetes Infrastructure as Code* que oferece suporte a todos os principais provedores de nuvem. Ele visa gerenciar e provisionar infraestruturas e serviços em nuvem usando o *kubectl*.

Além das ferramentas acima alguns provedores oferecem alguns serviços que se integram com as ferramentas descritas anteriormente, como por exemplo o *AWS OpsWorks*, que é um serviço de gerenciamento de configurações que permite as instâncias sejam gerenciadas tanto pelo *Chef* como pelo *Puppet*. Outros provedores oferecem essas ferramentas em seus *marketplaces*.

### 6.4.2. Boas práticas para rodar HPC na AWS

A seguir são descritas as melhores práticas para a execução de uma aplicação de HPC na infraestrutura da AWS.

- Gerenciar clusters com o *AWS Parallel Cluster*. Esta ferramenta de código aberto é totalmente mantida e suportada pela AWS, tornando mais fácil gerenciar e implementar clusters de HPC na AWS.
- Usar um *Placement Group* que são agrupamentos lógicos de instâncias na mesma zona de disponibilidade (AZ). O uso do *Placement Group* oferece conectividade de alta taxa de transferência e baixa latência entre instâncias do mesmo grupo, especialmente quando a maior parte do tráfego de rede está limitado às instâncias do grupo, como é o caso de um cluster.
- Evite usar *Hyper-Threading* (HT), a tecnologia Intel HT permite que várias *threads* sejam executadas simultaneamente no mesmo núcleo de CPU Intel Xeon, com cada *thread* atua como uma vCPU em uma instância do EC2. Para trabalhos de HPC, o *hyper-threading* pode ser lento, especialmente se o trabalho exigir cálculos de ponto flutuante. Cada núcleo possui dois threads que compartilham a mesma unidade de processamento (Unidade Lógica e Aritmética, cache, etc) e podem, assim, bloquear um ao outro. Certifique-se de que o HT esteja desabilitado nas instâncias do EC2.
- Use uma conexão rápida entre nós. Algumas instâncias do EC2 podem fornecer uma taxa de transferência de rede de até 100 Gbps. Estas instâncias podem beneficiar aplicações que dependem de uma comunicação rápida, *data lakes* e caches de memória.
- Usar instâncias do EC2 com GPU para tarefas gráficas ou de Machine Learning (ML). As instâncias P3 fornecem HPC baseado em nuvem com taxa de transferência de rede de até 100 Gbps e oito GPUs NVIDIA V100 Tensor Core para aplicativos de HPC e ML.
- Use o *Elastic Fabric Adapter* (EFA), esta interface de rede para instâncias do EC2 permite que os clientes executem aplicativos HPC que exigem comunicação intensa entre instâncias. Esta interface possui uma latência menor e mais

consistente e maior *throughput* do que os canais tradicionais de TCP, permitindo sua melhor escalabilidade. Entretanto apenas alguns tipos de instâncias suportam o EFA. Recomenda-se o uso junto com o *Placement Group*.

- Usar o *Amazon FSx for Lustre*, sistema de arquivos paralelo, para cargas de trabalho de ML e HPC. Esse sistema de arquivos totalmente gerenciado permite que você inicie um sistema de arquivos *Lustre* para processar grandes volumes de dados com uma taxa de transferência de centenas de GB/s e milhões de IOPS, garantindo latência abaixo de milissegundos.
- Escolha o tipo de instância certo para sua aplicação. Use instâncias otimizadas para computação ou de uso geral para aplicativos vinculados à CPU que exigem processadores de alto desempenho.

## 6.5. Implementação do cluster de HPC

Este minicurso irá focar na versão 3.x do *ParallelCluster*. A versão 2.x possui arquivos de configuração diferentes e estes não serão abordados aqui. Caso deseje utilizar as duas versões, recomenda-se o uso de um ambiente virtual para isolar ambos pacotes.

### 6.5.1 Instalação do *ParallelCluster*

Esta instalação no Linux pode ser feita no notebook, servidor, máquina virtual, na nuvem e praticamente em qualquer lugar. Recomenda-se o uso de um ambiente virtual para a instalação. Caso o *virtualenv* não esteja instalado execute os comandos abaixo.

```
[linux@ip ~]$ python3 -m pip install --upgrade pip
Successfully installed pip-22.2.2
[linux@ip ~]$ python3 -m pip install --user --upgrade virtualenv
```

Crie um ambiente virtual chamado, por exemplo, *pcluster* e ative ele, conforme mostrado abaixo. Repare que o *prompt* do comando mudou em função da ativação do ambiente virtual.

```
[linux@ip ~]$ python3 -m virtualenv ~/env/pcluster
[linux@ip ~]$ source ~/env/pcluster/bin/activate
(pcluster)[linux@ip ~]$
```

Caso ainda não esteja instalado, instale o AWS CLI, com os comandos a seguir.

```
(pcluster)[linux@ip ~]$ curl "https://awscli.amazonaws.com/awscli-exe-
linux-x86_64.zip" -o "awscliv2.zip"
(pcluster)[linux@ip ~]$ unzip awscliv2.zip
(pcluster)[linux@ip ~]$ sudo ./aws/install
```

Configure o AWS CLI para permitir acesso a nuvem da AWS, mas primeiro é necessário se entrar na console da AWS (Figura 2) e criar as chaves de acesso. Entre no serviço IAM selecionando o ícone de matriz ou digitando o serviço, como mostrado na Figura 3, selecione seu usuário, depois selecione a aba "*Credenciais de segurança*", e por fim clique no botão "*Criar chave de acesso*", conforme ilustrado na Figura 4.


*Figura 2: Pagina http://aws.amazon.com para acesso à console*

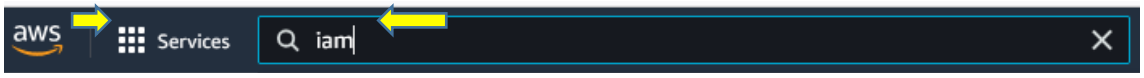


*Figura 3: Selecionando o serviço*

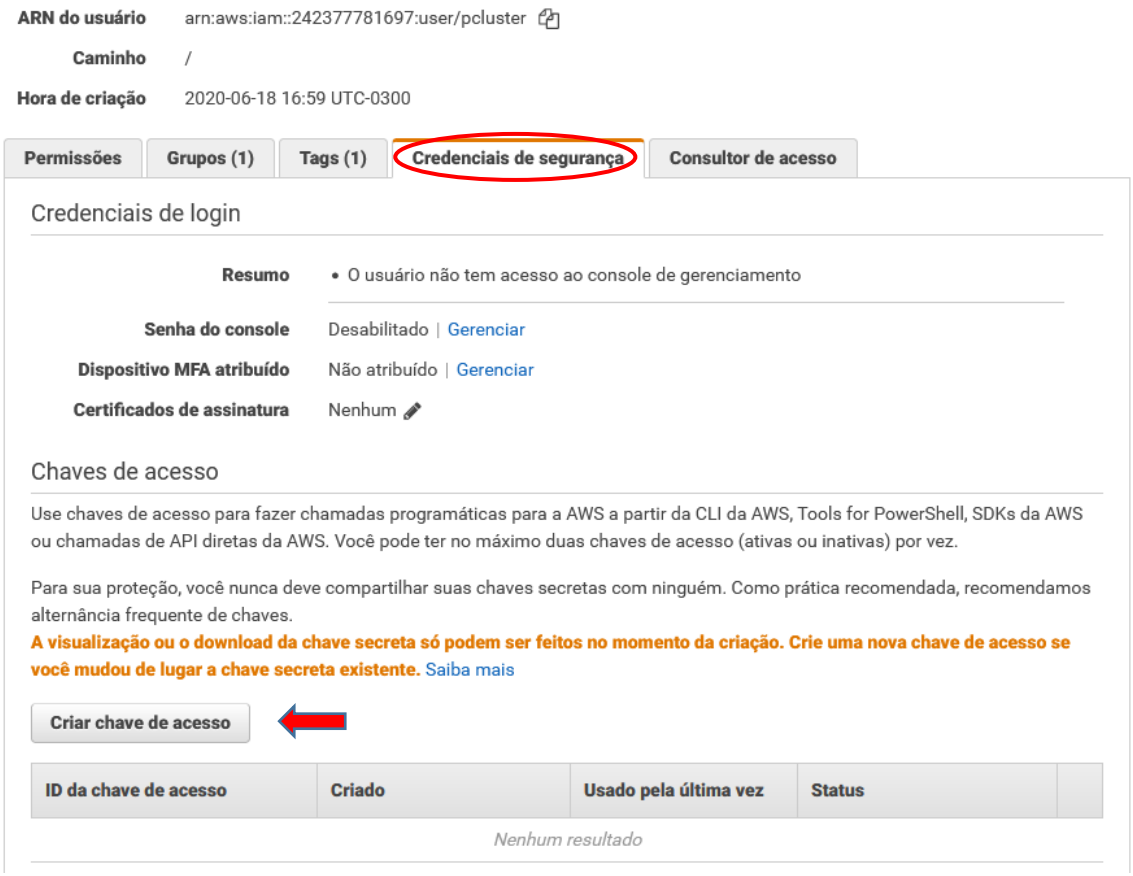


*Figura 4: Serviço IAM para criação da chave*

O acesso ao par de chave é somente neste momento. Pode-se visualizar ou fazer o download das chaves, como mostrado na Figura 5.

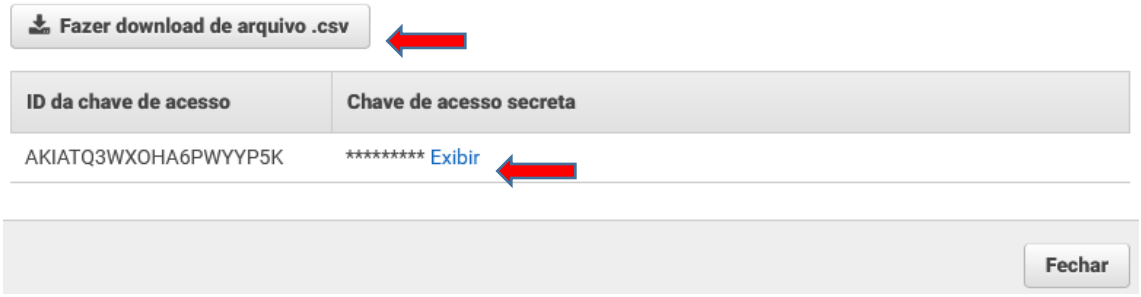


*Figura 5: Acesso ao par de chaves*

Dispondo do par de chaves pode-se configurar o AWS CLI através do comando.

```
(pcluster)[linux@ip ~]$ aws configure
AWS Access Key ID [None]: AKIATQ3WXOHA6PWYYP5K
AWS Secret Access Key [None]: yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
Default region name [None]: us-east-1
Default output format [None]:
```

Depois instale o *ParallelCluster* e verifique a versão instalada com os comandos abaixo.

```
(pcluster)[linux@ip ~]$ python3 -m pip install --upgrade \
                        aws-parallelcluster
(pcluster)[linux@ip ~]$ pcluster version
{
  "version": "3.2.0"
}
```

Instale o *Node Version Manager* e o *Node.js* dentro do ambiente virtual. Ele é necessário devido ao uso do *AWS Cloud Development Kit* (CDK) para a geração de *templates*.

```
(pcluster)[linux@ip ~]$ curl -o- https://raw.githubusercontent.com/nvm-
sh/nvm/v0.38.0/install.sh | bash
(pcluster)[linux@ip ~]$ chmod ug+x ~/.nvm/nvm.sh
(pcluster)[linux@ip ~]$ source ~/.nvm/nvm.sh
(pcluster)[linux@ip ~]$ nvm install --lts
(pcluster)[linux@ip ~]$ node --version
v16.17.0
```

Antes de configurar o *ParalleCluster*, recomenda-se gerar um par de chaves SSH para o cluster, conforme ilustrado na Figura 6. Escolha um nome para a chave e selecione o formato desejado. O formato PEM normalmente é usado para o Linux, Mac OS X e alguns softwares do Windows, enquanto que o formato PPK costuma ser usado para o programa PuTTY no Windows. Crie as chaves usando o botão “Criar par de chaves“ e o download delas será feito automaticamente.

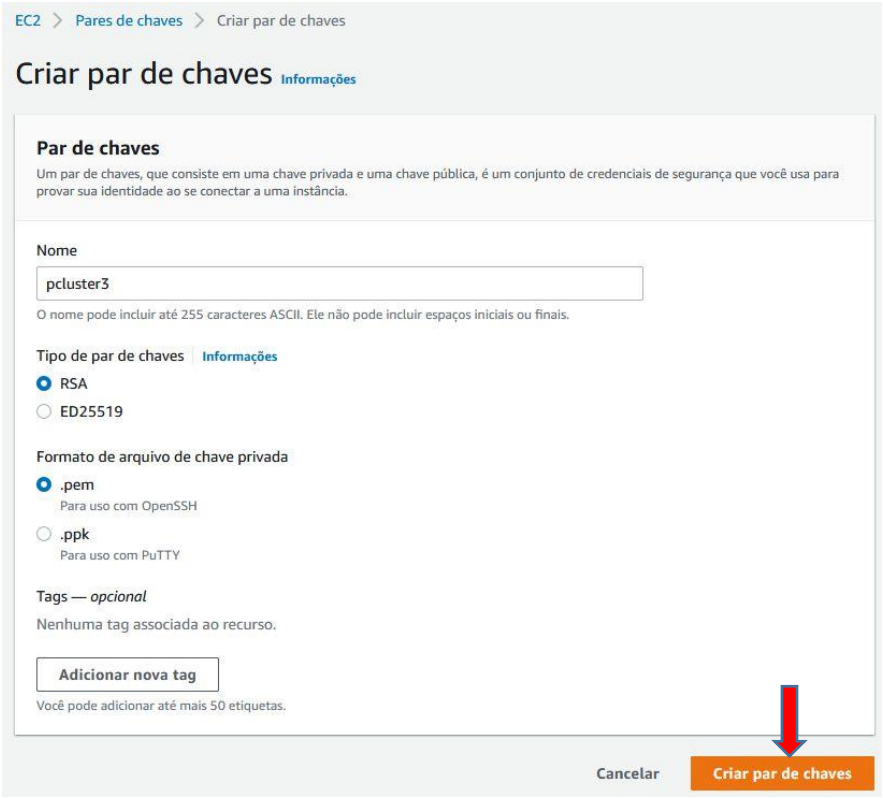


*Figura 6: Gerar par de chaves do cluster para acesso via SSH*

Então finalmente pode-se configurar o *ParallelCluster*. Ao executar o comando abaixo e gerar o arquivo de configuração no formato YAML serão feitas diversas perguntas sobre o cluster. Segue a seguir um exemplo das possíveis respostas.

Primeiro, escolha uma das regiões da AWS. A região *sa-east-1* corresponde a região de São Paulo. Esta região é a que possui a menor latência de comunicação, pois fica no Brasil. Outra região com baixa latência e a `us-east-1`.

```
(pcluster)[linux@ip ~]$ pcluster configure --config pcluster-config.yaml
INFO: Configuration file pcluster-config.yaml will be written.
Press CTRL-C to interrupt the procedure.
Allowed values for AWS Region ID:
1. ap-northeast-1
2. ap-northeast-2
3. ap-south-1
. . .
16. sa-east-1
. . .
AWS Region ID [us-east-1]:
```

Depois selecione o par de chaves gerado anteriormente na mesma região, vide Figura 6.

```
Allowed values for EC2 Key Pair Name:
1. pcluster3
EC2 Key Pair Name [pcluster3]: 1
```

Selecione o SLURM [ 10 ] como o *job scheduler*, muito comum em clusters de HPC. O AWS Batch não será abordado neste minicurso.

```
Allowed values for Scheduler:
1. slurm
2. awsbatch
Scheduler [slurm]: 1
```

Escolha qual o Sistema operacional do cluster. O *alinux2* (*Amazon Linux*) é uma distribuição Linux do AWS, originalmente baseada no Red Hat, simplificada otimizada para execução em instâncias do EC2. Ele fornece várias ferramentas para integrar-se perfeitamente aos serviços do EC2. Não é uma distribuição Linux independente e está disponível apenas para uso em um ambiente EC2.

```
Allowed values for Operating System:
1. alinux2
2. centos7
3. ubuntu1804
4. ubuntu2004
Operating System [alinux2]:
```

Escolha o tipo de instância, lembrando que o *headnode* pode ter uma instância diferente dos nós computacionais. Entretanto, recomenda-se que a rede tenha a mesma banda de comunicação entre eles, pois o *headnode* faz serviço de NFS com os nós computacionais e se as instâncias tiverem redes de velocidade diferentes pode impactar no desempenho. Neste exemplo, foi escolhido a *t3.micro* por, atualmente, ter melhor desempenho que a *t2.micro* e um custo menor.

```
Head node instance type [t2.micro]: t3.micro
```

Definir quantidade e as características de cada fila.

```
Number of queues [1]: 1
Name of queue 1 [queue1]: work
Number of compute resources for work [1]:
Compute instance type for compute resource 1 in work [t2.micro]: t3.micro
Maximum instance count [10]: 5
```

Depois que as etapas anteriores forem concluídas, decida se deseja usar uma VPC existente ou deixar que o AWS *ParallelCluster* crie uma VPC automaticamente, incluindo informações sobre a zona de disponibilidade. Pode-se usar o *headnode* e os nós de computação na mesma sub-rede pública ou somente o *headnode* em uma sub-rede pública com todos os nós em uma sub-rede privada. Em geral usa-se a segunda opção, porém se o nó de computação precisar acessar a Internet pode ser que a primeira opção seja mais fácil de configurar. Outra observação é que pode atingir o limite de VPCs permitidos em uma região. A cota padrão é cinco VPCs para uma região, mas caso seja necessário pode-se solicitar um aumento de cota no suporte da AWS.

```
Automate VPC creation? (y/n) [n]: y
Allowed values for Availability Zone:
1. us-east-1a
2. us-east-1b
3. us-east-1c
4. us-east-1d
5. us-east-1e
6. us-east-1f
Availability Zone [us-east-1a]:
Allowed values for Network Configuration:
1. Head node in a public subnet and compute fleet in a private subnet
```

```
2. Head node and compute fleet in the same public subnet
Network Configuration [Head node in a public subnet and compute fleet
in a private subnet]: 1
```

Finalmente o script gera toda a infraestrutura selecionada e gera o arquivo de configuração, neste caso o *pcluster-config.yaml*.

```
Beginning VPC creation. Please do not leave the terminal until the
creation is finalized
Creating CloudFormation stack...
Do not leave the terminal until the process has finished.
Stack Name: parallelclusternetworking-pubpriv-20220818134347 (id:
arn:aws:cloudformation:us-east-
1:242377781697:stack/parallelclusternetworking-pubpriv-
20220818134347/e814cd70-1efd-11ed-b0d6-0a60bc1e437f)
Status: parallelclusternetworking-pubpriv-20220818134347 -
CREATE_COMPLETE
The stack has been created.
Configuration file written to pcluster-config.yaml
You can edit your configuration file or simply run 'pcluster create-
cluster --cluster-configuration pcluster-config.yaml --cluster-name
cluster-name --region us-east-1' to create your cluster.
```

A seguir é mostrado o arquivo de configuração *pcluster-config.yaml.*

```
Region: us-east-1
Image:
  Os: alinux2
HeadNode:
  InstanceType: t3.micro
  Networking:
    SubnetId: subnet-0caf3554d75f42d7f
  Ssh:
    KeyName: pcluster3
Scheduling:
  Scheduler: slurm
  SlurmQueues:
  - Name: work
    ComputeResources:
    - Name: t3micro
      InstanceType: t3.micro
      MinCount: 0
      MaxCount: 5
    Networking:
      SubnetIds:
      - subnet-02b21b350c1169c8e
```

Feita a configuração básica pode-se fazer uma customização dos serviços que serão utilizados pelo cluster, alguns deles serão descritos no próximo item.

Recomenda-se que depois de qualquer alteração no arquivo de configuração, faça um teste na nova configuração. Para isto acrescente o parâmetro *"`--dryrun True`"* para não gerar o cluster e apenas verificar se a configuração possui algum erro. Se não houver

erro, a saída do comando de criar cluster será semelhante ao mostrado a seguir, onde o parâmetro "`-n`" especifica o nome do cluster e o "`-c`" o arquivo de configuração.

```
(pcluster)[linux@ip ~]$ pcluster create-cluster --dryrun True -n wscad
-c pcluster-config.yaml
{
  "message": "Request would have succeeded, but DryRun flag is set."
}
```

### 6.5.2. Customização na configuração e boas práticas

#### *6.5.2.1. S3*

Um dos serviços mais utilizados na AWS é o S3. Ele é o responsável por fazer o armazenamento de objetos. Para usar este serviço com o cluster recomenda-se primeiro criar os *buckets* e depois editar o arquivo de configuração YAML. O *bucket* S3 tem que ter um nome único dentro do S3 e não apenas na sua conta ou região. Neste minicurso, serão criados dois *buckets* chamados de *aws-pcluster3-dados* e *aws-pcluster3*, sendo que o primeiro será usado pelo cluster apenas como leitura e o segundo como leitura e escrita. Esse recurso é muito interessante quando se quer preservar e proteger os dados originais de algum problema durante a execução. Além de restringir o acesso aos demais *buckets* disponíveis na conta, pois apenas esses dois serão visualizados pelo cluster.

Selecione o serviço S3 de forma semelhante ao mostrado na Figura 3 e clique no botão "*Criar bucket*". Coloque o nome desejado e preste atenção para usar a mesma região em que o cluster foi ou será criado. As demais configurações podem ser deixadas como padrão. Depois vá ao final da página e clique no botão "*Criar bucket*". A Figura 7 ilustra parte do processo.

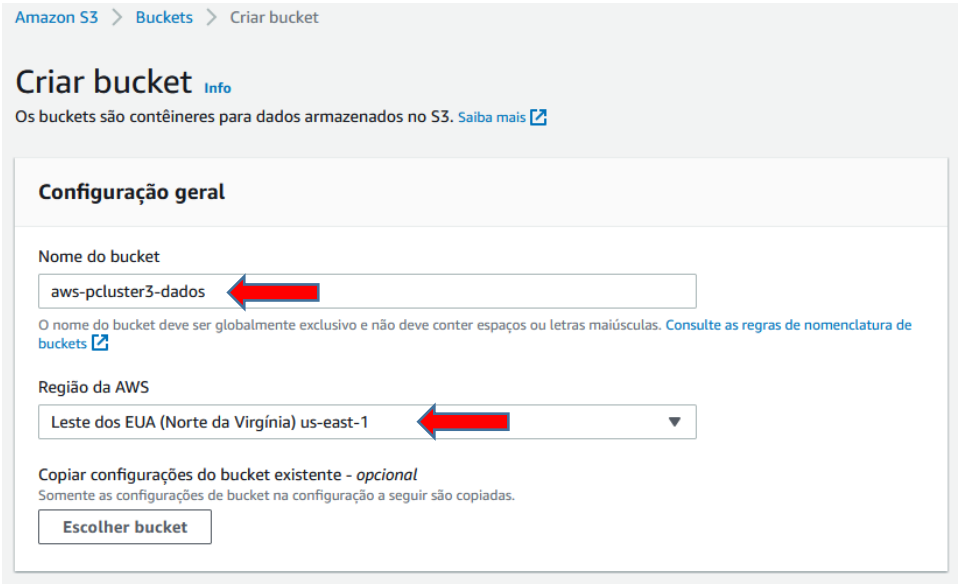


*Figura 7: Criando um bucket S3*

Adicione o trecho a seguir na configuração do YAML. Esta configuração dará acesso apenas de leitura ao `s3://aws-pcluster3-dados/dados_ro/` e acesso de leitura e escrita no `s3://aws-pcluster3/dados/`.

```
...
HeadNode:
...
Iam:
  S3Access:
    - BucketName: aws-pcluster3-dados
      KeyName: dados_ro/*
      EnableWriteAccess: false
    - BucketName: aws-pcluster3
      KeyName: dados/*
      EnableWriteAccess: true
...
```

Se for necessário ter acesso a todos os *buckets* basta mudar o parâmetro `BucketName` como mostrado a seguir.

```
...
Iam:
  S3Access:
    - BucketName: *
...
```

Dependendo de como vai ser usado o cluster pode-se incluir esta configuração também para os nós computacionais que estão definidos nas filas. Porém, cabe lembrar que o S3 é um sistema de armazenamento de objetos e não de blocos. Normalmente, o processamento de um programa deve usar o armazenamento em blocos e ao terminar e for necessário, pode-se copiar os arquivos para um sistema de armazenamento em objetos como o S3.

```
...
Scheduling:
  Scheduler: slurm
  SlurmQueues:
  - Name: work
  ...
    Iam:
      S3Access:
        - BucketName: aws-pcluster3
          EnableWriteAccess: true
        - BucketName: aws-pcluster3-dados
          KeyName: dados_ro/*
          EnableWriteAccess: false
  ...
```

#### *6.5.2.2. Placement Group*

Para configurar o *placement group* basta acrescentar o trecho abaixo no item relativo a rede da fila como mostrado abaixo.

```
…
Scheduling:
  Scheduler: slurm
  SlurmQueues:
  - Name: work
  ...
    Networking:
      SubnetIds:
      - subnet-02b21b350c1169c8e
      PlacementGroup:
        Enabled: true
  ...
```

#### *6.5.2.3. Minimização de custos*

A fim de minimizar o custo financeiro pode-se utilizar instâncias Spots no cluster. Em geral, as instâncias Spots possuem um preço menor que a instância sob demanda e às vezes o desconto pode chegar até 90%. Na configuração do cluster pode-se definir o preço máximo que será pago por cada instância Spot do EC2 antes que elas sejam executadas. O valor padrão é o preço sob demanda. Cabe aqui lembrar que se a AWS receber uma oferta maior pela sua instância que está sendo usada para processar o seu *job*, ele pode a qualquer momento desligar ela, mesmo que o seu *job* ainda não tenha terminado. Apesar deste inconveniente, este tipo de instância é muito utilizado em função do retorno financeiro e pela pouca frequência de interrupção, como pode ser vista pelo *Spot Instance Advisor* [ *11* ] que está mostrado na Figura 8. Nesta figura pode-se ver uma economia alta, em média acima de 70%, e uma frequência de interrupção menor que 5%.

Para exemplificar com números, suponha que o preço da instância *t3.micro* sob demanda esteja $0,0104. Pode-se definir que o preço máximo a ser pago por cada instância seja de $0,008. Neste caso, a configuração do arquivo YAML deve incluir também a linha com o preço máximo além de definir o tipo como SPOT.

```
...
Scheduling:
  Scheduler: slurm
  SlurmQueues:
  - Name: work
    ComputeResources:
    - Name: t3micro
      InstanceType: t3.micro
      MinCount: 0
      MaxCount: 5
      SpotPrice: 0.008
    CapacityType: SPOT
    ...
```

| Tipo de instância | vCPU | Memória GiB | Economia em relação aos preços sob demanda* | Frequência da interrupção |
|---|---|---|---|---|
| c5ad.4xlarge | 16 | 32 | 78% | <5% |
| c6g.xlarge | 4 | 8 | 71% | <5% |
| c6g.2xlarge | 8 | 16 | 71% | <5% |
| c5.large | 2 | 4 | 78% | <5% |
| t3a.nano | 2 | 0.5 | 66% | <5% |
| r5b.2xlarge | 8 | 64 | 86% | <5% |
| m6g.12xlarge | 48 | 192 | 73% | <5% |
| r6id.2xlarge | 8 | 64 | 85% | <5% |
| r5.metal | 96 | 768 | 83% | <5% |
| m5d.16xlarge | 64 | 256 | 82% | <5% |

Exibir todos os tipos de instância 473

*Figura 8: Spot Instance Advisor na região de Ohio*

#### *6.5.2.4. Hyperthreading*

Normalmente recomenda-se que o *hyperthreading* esteja desativado para o processamento de aplicações de HPC por questões de desempenho. Este parâmetro vem ativado por padrão nas instâncias EC2. Costuma-se desabilitar, caso seja necessário, apenas nos nós computacionais. Entretanto, nem todos os tipos de instância é possível desabilitar o *hyperthreading*. A configuração do parâmetro `DisableSimultaneousMultithreading` deve ser feito na fila correspondente. A mudança deste parâmetro não gera nenhum custo adicional.

```
...
Scheduling:
  Scheduler: slurm
  SlurmQueues:
  - Name: work
    ComputeResources:
    - Name: t3micro
      InstanceType: t3.micro
      MinCount: 0
      MaxCount: 5
      DisableSimultaneousMultithreading: true
    ...
```

#### *6.5.2.5. AMI customizada*

As aplicações de alto desempenho, normalmente, requerem algumas características diferentes das aplicações tradicionais. E para tanto, as vezes é necessário criar um ambiente bem específico que não são inclusas nas imagens tradicionais da AWS. Para endereçar a este tipo de problema é possível criar uma AMI (*Amazon Machine Image*) customizada.

A forma mais prática de criar uma imagem seria levantar uma instância com o sistema operacional desejado (*Amazon Linux, Rocky Linux, Ubuntu*, etc) e fazer todas as instalações e configurações necessárias para o ambiente da aplicação. Outra forma seria, se já tivesse algum script otimizado que fizesse todas as instalações e configurações. Este script poderia ser executado na hora de iniciar a instancia e depois gerar uma nova AMI, por exemplo, `ami-12345678`.

Estando de posse desta AMI pode-se usa-la como uma imagem, normalmente, dos nós computacionais do cluster conforme mostrado no trecho da configuração abaixo.

```
...
Scheduling:
  Scheduler: slurm
  SlurmQueues:
    - Name: work
      Image:
        CustomAmi: ami-12345678
      ...
```

#### *6.5.2.6. Tempo de espera*

A alocação do nó computacional se dá de acordo com a demanda e a definição do tamanho do cluster. Ao submeter um *job, uma ou mais* instâncias são inicializadas para o processamento, porém este processo demora poucos minutos. Entretanto, pode-se definir o tempo máximo que um nó computacional fique aguardando um novo *job,* evitando assim uma nova espera do provisionamento. Se esse tempo for ultrapassado o nó em questão será terminado e removido do cluster, desta forma parando de gerar custo. Por padrão esse tempo é de 10 minutos. O parâmetro que permite esse ajuste é o `ScaledownIdletime`. No trecho de configuração a seguir esse limite foi modificado para 5 minutos.

```
...
Scheduling:
  Scheduler: slurm
  SlurmSettings:
    ScaledownIdletime: 5
        ...
```

### 6.5.3. Usando o *ParallelCluster*

#### *6.5.3.1. Criando o cluster*

Depois da visão geral das principais características do *ParallelCluster* pode-se fazer a implementação do cluster. Neste exemplo será usado a seguinte configuração do YAML. Esta configuração exemplifica a alguns dos parâmetros descritos anteriormente, tais como: acesso ao S3 tanto de leitura e escrita como apenas de leitura, configuração do *idle time*, uso de instâncias spots, etc.

```
Region: us-east-1
Image:
  Os: alinux2
HeadNode:
  InstanceType: t3.micro
  Networking:
    SubnetId: subnet-00af4837d78f42d7f
  Ssh:
    KeyName: pcluster3
  Iam:
    S3Access:
      - BucketName: aws-pcluster3
        EnableWriteAccess: true
      - BucketName: aws-pcluster3-dados
        EnableWriteAccess: false
Scheduling:
  Scheduler: slurm
  SlurmSettings:
    ScaledownIdletime: 5
  SlurmQueues:
  - Name: work
    ComputeResources:
    - Name: t3micro
      InstanceType: t3.micro
      MinCount: 0
      MaxCount: 5
    CapacityType: SPOT
    Networking:
      SubnetIds:
      - subnet-02b23b87c1165c8e
    Iam:
      S3Access:
        - BucketName: aws-pcluster3
          EnableWriteAccess: true
        - BucketName: aws-pcluster3-dados
          EnableWriteAccess: false
```

Para criar um cluster utilize o comando a seguir e aguarde alguns minutos até que o serviço *CloudFormation* faça todas as configurações de rede, segurança e inicialize o *headnode*. Veja o status inicial do cluster `CREATE_IN_PROGRESS`.

```
(pcluster)[ec2-user@ip ~]$ pcluster create-cluster -n wscad \
                           -c pcluster-config.yaml
{
  "cluster": {
    "clusterName": "wscad",
    "cloudformationStackStatus": "CREATE_IN_PROGRESS",
    "cloudformationStackArn": "arn:aws:cloudformation:us-east-1:...",
    "region": "us-east-1",
    "version": "3.2.0",
    "clusterStatus": "CREATE_IN_PROGRESS",
    "scheduler": {
```

```
      "type": "slurm"
    }
  }
}
```

Pode-se acompanhar a implementação do cluster com o comando abaixo até dar por concluído todo o processo, com o status `CREATE_COMPLETE`.

```
(pcluster)[ec2-user@ip ~]$ pcluster describe-cluster -n wscad
{
  "creationTime": "2022-08-26T19:09:50.252Z",
  "headNode": {...},
  "version": "3.2.0",
  "clusterConfiguration": {...},
  "tags": [ {...}, {...} ],
  "cloudFormationStackStatus": "CREATE_COMPLETE",
  "clusterName": "wscad",
  "computeFleetStatus": "RUNNING",
  "cloudformationStackArn": "arn:aws:cloudformation:...",
  "lastUpdatedTime": "2022-08-26T19:09:50.252Z",
  "region": "us-east-1",
  "clusterStatus": "CREATE_COMPLETE",
  "scheduler": {
    "type": "slurm"
  }
}
```

Também é possível listar todos os clusters com o comando a seguir.

```
 (pcluster)[ec2-user@ip ~]$ pcluster list-clusters
{
  "clusters": [
    {
      "clusterName": "wscad",
      "cloudformationStackStatus": "CREATE_COMPLETE",
      "cloudformationStackArn": "arn:aws:cloudformation:us-east-1:...",
      "region": "us-east-1",
      "version": "3.2.0",
      "clusterStatus": "CREATE_COMPLETE",
      "scheduler": { "type": "slurm" }
    }
  ]
}
```

Caso haja interesse em examinar os eventos do *CloudFormation* use o comando a seguir ou consulte diretamente o serviço através da console da AWS.

```
(pcluster)[ec2-user@ip ~]$ pcluster get-cluster-stack-events -n wscad
```

Inicialmente só havia a instancia chamada *Bastion*, onde foi instalado o ambiente virtual com o *ParallelCluster*. Ao executar o comando para criar o cluster foi inicializado o *headnode* do cluster sem nenhum nó computacional, conforme mostra a Figura 9.

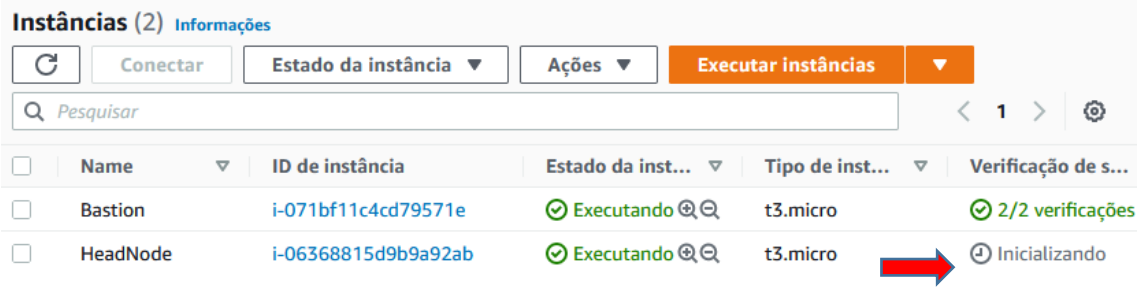


*Figura 9: Instâncias na AWS*

### *6.5.3.2. Acessando o cluster*

Para acessar o cluster basta usar a chave gerada anteriormente, vide Figura 6.

```
(pcluster)[ec2-user@ip ~]$ pcluster ssh -n wscad -i ~/.ssh/pcluster3.pem
Last login: Fri Aug 26 19:14:53 2022

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/

[ec2-user@ip-10-0-0-201 ~]$
```

Coletando informações da fila, neste acaso apenas uma fila com cinco nós computacionais.

```
[ec2-user@ip-10-0-0-201 ~]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
work*        up   infinite      5  idle~ work-dy-t3micro-[1-5]
```

Consulte as áreas compartilhadas pelo *headnode* para os nós computacionais através do serviço NFS.

```
[ec2-user@ip-10-0-0-201 ~]$ sudo exportfs -v
/home                         10.0.0.0/16(rw,sync,wdelay,hide,
    no_subtree_check,sec=sys,secure,no_root_squash,no_all_squash)
/opt/parallelcluster/shared   10.0.0.0/16(rw,sync,wdelay,hide,
    no_subtree_check,sec=sys,secure,no_root_squash,no_all_squash)
/opt/intel                    10.0.0.0/16(rw,sync,wdelay,hide,
    no_subtree_check,sec=sys,secure,no_root_squash,no_all_squash)
/opt/slurm                    10.0.0.0/16(rw,sync,wdelay,hide,
    no_subtree_check,sec=sys,secure,no_root_squash,no_all_squash)
```

A versão do MPI que já vem instalado por padrão no *ParallelCluster* é o OpenMPI [ 12 ], vide comando abaixo, e ele tem suporte para a interface EFA. Entretanto para usar o EFA é necessário instalar o software EFA, vide o *ParallelCuster User Guide* [ 13 ]. Também é possível usar o Intel MPI e o MVAPICH2-X-AWS [ 14 ]. Estas instalações e configurações não serão abordados neste minicurso.

```
[ec2-user@ip-10-0-0-189 ~]$ which mpirun
/opt/amazon/openmpi/bin/mpirun
```

Submetendo um *job* MPI que usará dois nós computacionais. Depois verifica-se o estado da fila. Repare que o status (ST) está como CF, isto quer dizer que o nó ainda está sendo configurado.

```
[ec2-user@ip-10-0-0-201 ~]$ sbatch openmpi.job
Submitted batch job 1
[ec2-user@ip-10-0-0-201 ~]$ squeue
JOBID PARTITION     NAME     USER ST TIME  NODES NODELIST(REASON)
    1      work  openmpi ec2-user CF 0:50      2 work-dy-t3micro-[1-2]
```

Repare os nós de processamento sendo alocados na

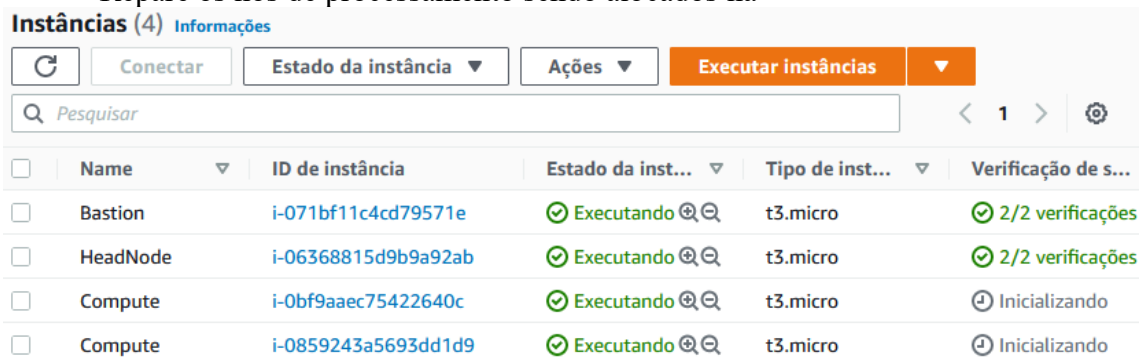


Figura 10.

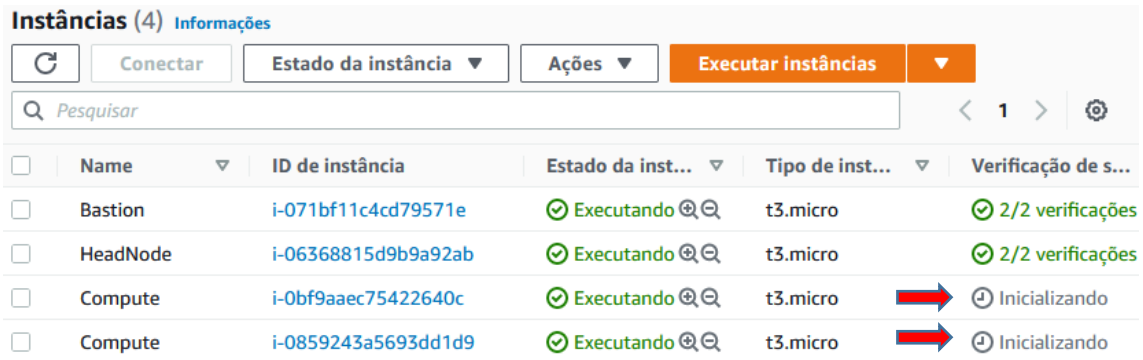


*Figura 10: Alocação dos nós de computação*

Submetendo um *job* serial e observando que após uns minutos o primeiro *job* já sendo executado enquanto que o *job* serial ainda está sendo alocado.

```
[ec2-user@ip-10-0-0-201 ~]$ sbatch test.job
Submitted batch job 2
[ec2-user@ip-10-0-0-201 ~]$ squeue
JOBID PARTITION     NAME     USER  ST TIME  NODES NODELIST(REASON)
    1      work openmpi. ec2-user   R 4:33      2 work-dy-t3micro-[1-2]
    2      work teste.sh ec2-user  CF 0:12      1 work-dy-t3micro-3
```

Observando novamente a fila, pode-se identificar o nome de DNS dos nós alocados e o mesmo responde pelo seu nome de DNS conforme ilustrado a seguir.

```
[ec2-user@ip-10-0-0-201 ~]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
work*        up   infinite      2  idle~ work-dy-t3micro-[4-5]
work*        up   infinite      1    mix work-dy-t3micro-3
work*        up   infinite      2  alloc work-dy-t3micro-[1-2]
```

```
[ec2-user@ip-10-0-0-201 ~]$ ping work-dy-t3micro-1
PING work-dy-t3micro-1.wscad.pcluster (10.0.24.168) 56(84) bytes of
data.
64 bytes from ip-10-0-24-168.ec2.internal (10.0.24.168): icmp_seq=1
ttl=255 time=0.201 ms
64 bytes from ip-10-0-24-168.ec2.internal (10.0.24.168): icmp_seq=2
ttl=255 time=0.203 ms
64 bytes from ip-10-0-24-168.ec2.internal (10.0.24.168): icmp_seq=3
ttl=255 time=0.192 ms
^C
--- work-dy-t3micro-1.wscad.pcluster ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 4054ms
rtt min/avg/max/mdev = 0.192/0.200/0.207/0.018 ms
```

Depois que os *jobs* terminaram e se passaram os 5 minutos definidos no arquivo YAML, os nós de processamento foram encerrados automaticamente, como mostrado na Figura 11.

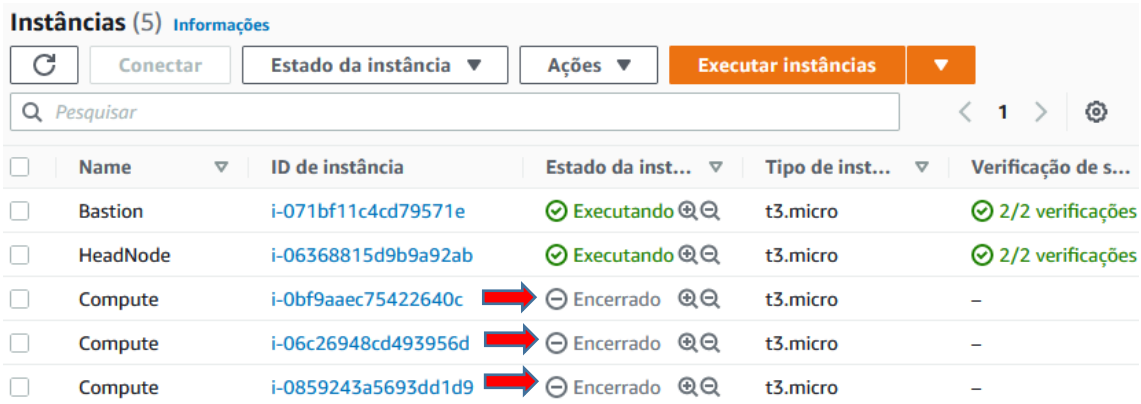


*Figura 11: Nós de processamento encerrados*

### *6.5.3.3. S3*

Verificando a política do S3 definida no arquivo YAML, que tem permissão de leitura e escrita no *bucket* `aws-pcluster` e só leitura no *bucket* `aws-pcluster-dados`, se está funcionando.

Primeiro tentando listar todos os *buckets* através do AWS CLI e não conseguindo porque só tem acesso aos dois *buckets* definidos anteriormente.

```
[ec2-user@ip-10-0-0-189 ~]$ aws s3 ls
An error occurred (AccessDenied) when calling the ListBuckets operation:
Access Denied

[ec2-user@ip-10-0-0-189 ~]$ aws s3 ls s3://aws-pcluster3/
2022-08-26 19:41:47      28820 pcluster-manager.yaml

[ec2-user@ip-10-0-0-189 ~]$ aws s3 ls s3://aws-pcluster3-dados
```

```
                           PRE dados_ro/
```

Copiando a saída do primeiro *job* para um novo diretório chamado output. Se o diretório não existir o S3 criará ele automaticamente, conforme mostrado a seguir.

```
[ec2-user@ip-10-0-0-189 ~]$ aws s3 cp slurm-1.out \
                            s3://aws-pcluster3/output/
upload: ./slurm-1.out to s3://aws-pcluster3/output/slurm-1.out

[ec2-user@ip-10-0-0-189 ~]$ aws s3 ls s3://aws-pcluster3/
                            PRE output/
2022-08-26 19:41:47      28820 pcluster-manager.yaml

[ec2-user@ip-10-0-0-189 ~]$ aws s3 ls s3://aws-pcluster3/output/
2022-08-28 18:42:04      81723 slurm-1.out
```

Tentando fazer o mesmo no *bucket* que tem permissão apenas de leitura.

```
[ec2-user@ip-10-0-0-189 ~]$ aws s3 cp slurm-1.out \
                            s3://aws-pcluster3-dados/output/
upload failed: ./slurm-1.out to s3://aws-pcluster3-dados/output/slurm-
1.out An error occurred (AccessDenied) when calling the PutObject
operation: Access Denied
```

É possível remover um diretório de forma recursiva com o comando abaixo.

```
[ec2-user@ip-10-0-0-189 ~]$ aws s3 rm s3://aws-pcluster3/ --recursive \
                            --exclude "*" --include "output/*"
delete: s3://aws-pcluster3/output/slurm-1.out
```

Porém é importante prestar atenção na sequência `--exclude` e depois `--include`, se esta ordem for invertida não irá funcionar. Por exemplo, no *bucket* abaixo ainda tem uma cópia do arquivo de configuração YAML. Se colocarmos primeiro o `--include` e depois `--exclude` veja que não funciona. Depois colocando na ordem correta o arquivo foi removido.

```
[ec2-user@ip-10-0-0-189 ~]$ aws s3 ls s3://aws-pcluster3
2022-08-26 19:41:47      28820 pcluster-manager.yaml

[ec2-user@ip-10-0-0-189 ~]$ aws s3 rm s3://aws-pcluster3/ --recursive \
                            --include "*.yaml" --exclude "*"
[ec2-user@ip-10-0-0-189 ~]$ aws s3 ls s3://aws-pcluster3
2022-08-26 19:41:47      28820 pcluster-manager.yaml
[ec2-user@ip-10-0-0-189 ~]$ aws s3 rm s3://aws-pcluster3/ --recursive \
                            --exclude "*" --include "*.yaml"
delete: s3://aws-pcluster3/pcluster-manager.yaml
```

#### *6.5.3.4. Removendo o cluster*

Depois de fazer todo o processamento necessário deve-se remover o cluster para não ficar gerando custo. A ideia da nuvem é que você paga pelo que consome, logo se terminou as analises, remova o cluster. O comando para remover o cluster está mostrado a seguir.

```
(pcluster)[ec2-user@ip ~]$ pcluster delete-cluster -n wscad
{
  "cluster": {
    "clusterName": "wscad",
    "cloudformationStackStatus": "DELETE_IN_PROGRESS",
    "cloudformationStackArn": "arn:aws:cloudformation:us-east-1:...",
    "region": "us-east-1",
    "version": "3.2.0",
    "clusterStatus": "DELETE_IN_PROGRESS",
    "scheduler": {
      "type": "slurm"
    }
  }
}
```

Após algum tempo pode-se confirmar que o cluster foi removido pelo comando abaixo.

```
(pcluster)[ec2-user@ip ~]$ pcluster list-clusters
{
  "clusters": []
}
```

## 6.6. Considerações Finais

Este minicurso mostrou as principais características do *ParallelCluster* para facilitar a implementação de um cluster na nuvem pública da AWS. Entretanto, não foi possível cobrir todos os recursos oferecidos pela ferramenta, nem entrar nos detalhes das configurações. A seguir uma lista de recursos que podem ser interessantes em se aprofundar: interface EFA incluindo o uso com o Intel MPI e o MVAPICH-X, EFS, *FSx for Lustre*, verificar as opções dos comandos, criação de AMIs personalizadas, múltiplas filas, múltiplos usuários, políticas de segurança da AWS, ajustes no SLURM para contabilidade e muitos outros.

Neste minicurso foi feito algumas ações via a console da AWS e outros por linha de comando, porém com o AWS CLI tudo poderia ser feito por linha de comando. O que dependendo da aplicação pode facilitar a automação de processos.

Existe também um *front-end* via web para o *AWS ParallelCluster*, conhecido como *Parallel Cluster Manager* [ 15 ]. Basicamente é UI (*User Interface*) que ajuda a montar o arquivo de configuração YAML e gerenciar o cluster.

## 6.7. Referências

[ 1 ] Star Cluster. Disponível em: http://star.mit.edu/cluster/

[ 2 ] AWS services used by AWS ParallelCluster. Disponível em: https://docs.aws.amazon.com/parallelcluster/latest/ug/aws-services.html. Acesso em: 15/08/2022.

[ 3 ] Lustre. Página inicial. Disponível em: https://www.lustre.org. Acesso em: 15/08/2022.

[ 4 ] Terraform. Página inicial. Disponível em: https://www.terraform.io/. Acesso em: 15/08/2022.

[ 5 ] Pulumi. Página inicial. Disponível em: https://www.pulumi.com/. Acesso em: 15/08/2022.

[ 6 ] Ansible Red Hat. Página inicial. Disponível em: https://www.ansible.com/. Acesso em: 15/08/2022.

[ 7 ] Chef. Página inicial. Disponível em: https://www.chef.io/. Acesso em: 15/08/2022.

[ 8 ] Puppet. Página inicial. Disponível em: https://puppet.com/. Acesso em: 15/08/2022.

[ 9 ] Crossplane. Página inicial. Disponível em: https://crossplane.io/. Acesso em: 15/08/2022.

[ 10 ] SLURM. Disponível em: https://slurm.schedmd.com/. Acesso em: 15/08/2022.

[ 11 ] Spot Instance Advisor. Disponível em: https://aws.amazon.com/pt/ec2/spot/instance-advisor/. Acesso em: 15/08/2022.

[ 12 ] Open MPI: Open Source High Performance Computing. Disponível em: https://www.open-mpi.org/. Acesso em: 15/08/2022.

[ 13 ] Amazon Web Service, “AWS ParallelCluster: AWS ParallelCluster User Guide”, July 6, 2022.

[ 14 ] MVAPICH: MPI over InfiniBand, Omni-Path, Ethernet/iWARP, and RoCE. Disponível em: https://mvapich.cse.ohio-state.edu/downloads/. Acesso em: 15/08/2022.

[ 15 ] ParallelCluster Manager - Make HPC Easy. Disponível em: https://github.com/aws-samples/pcluster-manager. Acesso em: 15/08/2022