Capítulo 3

Introdução à Computação de Alto Desempenho na Nuvem Computacional

Edson Borin, Charles Boulhosa Rodamilans e Jeferson Rech Brunetta

Abstract

HPC systems are expensive and the access to them is limited to some communities, however, the recent advances in cloud computing technologies are allowing anyone to access HPC resources by paying only for the use of the system. The main cloud computing providers are offering high performance computing services, such as machines with GPUs and machines with high computational power interconnected by a high performance network to support the execution of tightly coupled applications. Regarding conventional HPC, performed on clusters of computers, the user's benefits by using cloud computing services are: (a) no wait on queues to use the system; (b) significant time reduction in system deployment; (c) access to high performance resources without a high initial investment; (d) use of specific resources for each application, such as the use of machines with GPUs. The main goal of this course is to provide an introduction to high performance computing on cloud computing resources and demonstrate for the participants how cloud computing can be used to execute high performance programs that make use of technologies such as GPU and MPI.

Resumo

Sistemas de HPC são custosos e o acesso aos mesmos é geralmente limitado a uma pequena parcela da comunidade, entretanto, avanços recentes em tecnologias de computação na nuvem estão permitindo que qualquer um possa acessar recursos de HPC pagando apenas pelo uso do sistema. Os principais provedores de computação em Nuvem estão fornecendo serviços voltados para a computação de alto desempenho, tais como máquinas com GPU e máquinas com alto poder computacional interligadas por uma rede de alto desempenho para dar suporte à execução de aplicações fortemente acopladas. Em relação à HPC convencional, realizada em aglomerados de computadores, ao utilizar estes serviços da Nuvem Computacional, o usuário se beneficia por: (a) não esperar na fila

para utilização do sistema; (b) haver uma redução significativa no tempo para implantação do sistema; (c) ter acesso aos recursos de alto desempenho sem um investimento inicial elevado; e (e) possibilitar o uso de recursos específicos para cada aplicação, como, por exemplo, o uso de máquinas com GPU. O principal objetivo deste minicurso é prover uma introdução à computação de alto desempenho em recursos na nuvem computacional e demonstrar aos participantes como a nuvem pode ser usada para executar programas de alto desempenho que fazem uso de tecnologias como GPU e MPI.

3.1. Introdução a HPC e tipos de aplicações

Desde a invenção dos primeiros computadores eletrônicos de propósito geral, na década de 40, até o início dos anos 90, os supercomputadores, utilizados na computação de alto desempenho, ou *High Performance Computing* (HPC), eram máquinas altamente especializadas, com processadores vetoriais projetados especificamente para executar rapidamente um nicho de aplicações científicas e de engenharia. Com a evolução e popularização dos processadores destinados a computadores pessoais e estações de trabalho, a partir da década de 90 os supercomputadores passaram a ser projetados como aglomerados de computadores com processadores de propósito geral, disponíveis no mercado.

Aglomerados, ou ***clusters***, de computadores são sistemas computacionais formados por computadores organizados em *racks*, sendo que cada computador possui memória própria. Ao longo dos anos surgiram diversas abordagens para facilitar a programação deste tipo de sistema, mas a que se tornou mais popular foi a ***Message Passing Interface* (MPI)**, que oferece funções para que o programador coordene manualmente o fluxo de dados entre os processos que executam em paralelo nos diferentes nós de processamento do *cluster*.

Nos anos 90, o desempenho dos processadores de propósito geral crescia exponencialmente, dirigido principalmente pelo aumento da frequência de operação dos processadores. Entretanto, no início dos anos 2000, a indústria de semicondutores encontrava cada vez mais dificuldades para aumentar a frequência de operação dos processadores e sinalizou que a maneira de continuar oferecendo processadores com desempenho cada vez melhor seria através da introdução de múltiplos núcleos computacionais em um mesmo processador, os processadores ***multi-core***. Dessa forma, a partir de 2005, os fabricantes de processadores começaram a produzir processadores *multi-core*, sendo que os núcleos computacionais compartilhavam a memória do computador. Nesta mesma época, os supercomputadores, que eram baseados em processadores de propósito geral, também começaram a fazer uso de processadores *multi-core*.

Os programas que eram codificados com MPI podiam tirar proveito naturalmente dos múltiplos núcleos computacionais dos supercomputadores através do lançamento de múltiplos processos no mesmo nó computacional (um processo para cada núcleo computacional, ou *core*). No entanto, esta abordagem consome mais memória e, em alguns casos, é mais lenta do que as alternativas que fazem uso direto da memória compartilhada entre os *cores* e o modelo de programação para memória compartilhada **OpenMP** se tornou mais popular para a programação paralela dos múltiplos núcleos dentro de cada computador do *cluster*. Dessa forma, os programas desenvolvidos para os supercomputadores passaram a fazer uso de MPI para a comunicação entre os nós do *cluster* e OpenMP

para a programação paralela dos múltiplos núcleos computacionais dentro de um mesmo nó.

Nos anos 2000 também houve uma evolução significativa no *hardware* das placas de processamento gráfico, ou **GPUs**[1], que deixaram de ter funcionalidade fixa e passaram a ser programáveis. Neste mesmo período, pesquisadores verificaram que era possível e vantajoso utilizar o poder computacional destes dispositivos para realizar processamento científico [Luebke et al. 2004]. Como consequência, a indústria evoluiu o *hardware* das GPUs e desenvolveu linguagens e ferramentas para facilitar a programação destes dispositivos para resolver problemas de propósito geral, dando origem ao termo ***General Purpose Graphics Processing Unit* (GPGPU)** [Owens et al. 2007]. A partir de 2009 as GPGPUs passaram a ser integradas nos supercomputadores para serem utilizadas como aceleradores de código. Em resposta a este movimento, a Intel desenvolveu o Xeon PHI, um acelerador em *hardware* feito para competir com as GPGPUs.

As GPGPUs possuem um *hardware* bem diferente dos processadores *multi-core* e no início não era possível converter automaticamente programas em C, C++ ou Fortran para serem executados de forma eficiente nas GPGPUs, dessa forma, novas linguagens de programação surgiram. Uma das primeiras linguagens foi o **CUDA**, da NVidia, no entanto, outras abordagens como **OpenCL** [Du et al. 2012], **OpenACC** e **OpenMP 4.0** surgiram ao longo do tempo. Neste novo ecossistema, os supercomputadores passaram a ser programados com MPI + X, onde X pode ser uma composição de OpenMP, CUDA, OpenCL, *etc*. A Figura 3.1 ilustra a evolução dos supercomputadores ao longo dos anos, incluindo os dispositivos computacionais e suas respectivas tecnologias de programação.

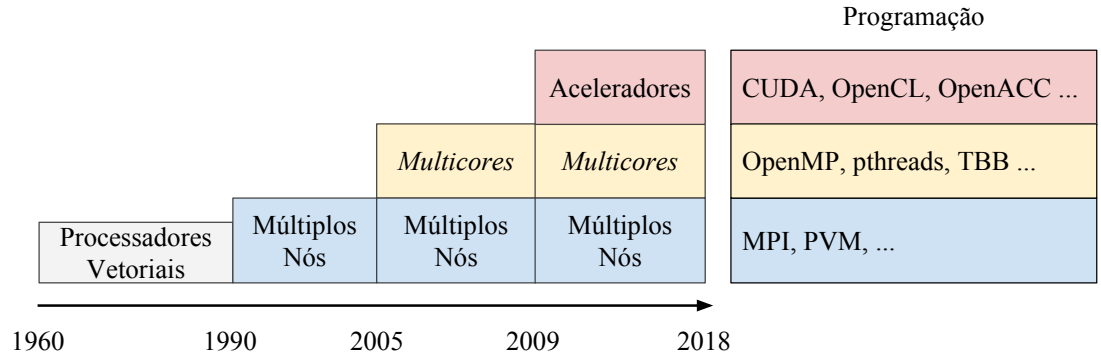


Figura 3.1: Evolução dos supercomputadores ao longo dos anos, incluindo os dispositivos computacionais e suas respectivas tecnologias de programação.

As aplicações que demandam processamento de alto desempenho podem ser classificadas em "aplicações fortemente acopladas" e "aplicações fracamente acopladas", descritas a seguir.

3.1.1. Aplicações fortemente acopladas vs fracamente acopladas

A evolução dos supercomputadores foi dirigida principalmente por aplicações caracterizadas como "**fortemente acopladas**", ou *Tightly-Coupled Application* (TC-A), que exigem uma rede de interconexão de alto desempenho. Muitas destas aplicações imple-

[1] do inglês: *Graphic Processing Unit*.

mentam algoritmos projetados com o modelo de programação ***Bulk Synchronous Parallel* (BSP)** [Valiant 1990], que trabalha em rodadas intercaladas de a) computação isolada nos nós computacionais e b) troca de informações entre os nós, como ilustrado na Figura 3.2. Neste modelo de programação, o algoritmo faz uso de barreiras de sincronização para fazer com que o programa aguarde até que todos os nós terminem uma rodada antes de iniciar a próxima.

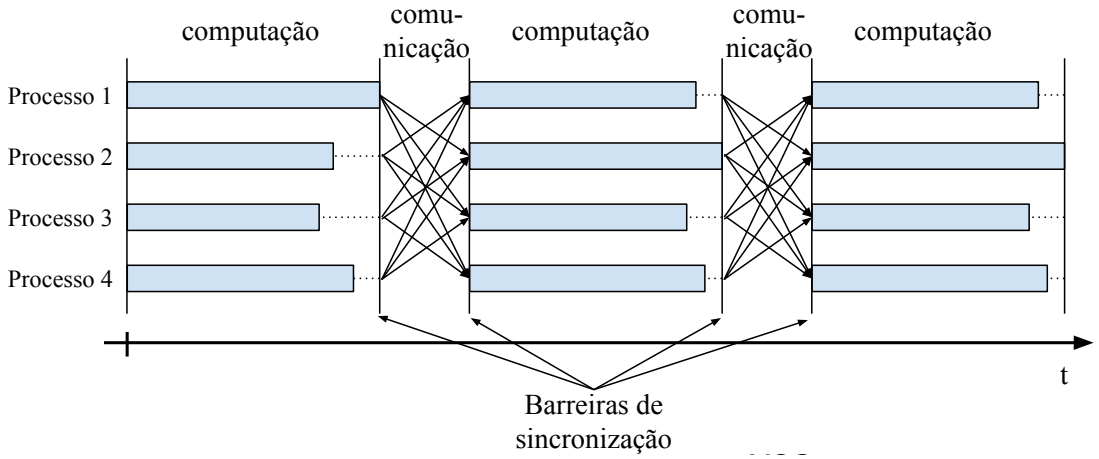


Figura 3.2: Execução de aplicações fortemente acopladas em rodadas de computação e comunicação.

Note que uma rede de interconexão lenta poderia fazer com que o tempo gasto nas rodadas de comunicação de aplicações fortemente acopladas (TC-A) aumentasse demasiadamente o tempo de execução da aplicação. Como consequência, os *clusters* de HPC são geralmente compostos por computadores de alto desempenho conectados por uma rede especializada de alto desempenho, com o padrão InfiniBand, por exemplo. A biblioteca de programação MPI oferece suporte à sincronização e comunicação entre os processos de aplicações que executam em *clusters* de HPC e tem sido amplamente utilizada para codificação de aplicações fortemente acopladas.

Existem diversos tipos de problemas que demandam um alto poder computacional mas que podem ser paralelizados sem que haja muita comunicação entre os diferentes nós de computação. Estes problemas, também conhecidos como "**embaraçosamente paralelos**", permitem a implementação de aplicações "**fracamente acopladas**", ou *Loosely-Coupled Application* (LC-A), que são facilmente paralelizáveis e podem ser executadas de forma eficaz em aglomerados de computadores conectados por uma rede de interconexão comum, sem garantias de desempenho.

Por não precisarem de comunicação de alto desempenho, as aplicações fracamente acopladas (LC-A) também podem ser executadas em federações de *clusters* e estações de trabalhos conectados através da Internet. Este tipo de computação, que pode fazer uso de sistemas ociosos e geograficamente dispersos, é conhecido como **computação em grade**[2]. Este modelo de computação paralela se tornou amplamente conhecido por aplicações de computação voluntária famosas como o projeto SETI@home, que faz uso

[2]do inglês: *grid computing*.

de computadores ociosos[3] para processar dados em busca de inteligência extraterrestre.

Na computação em grade o sistema computacional pode aumentar ou diminuir de tamanho dinamicamente, durante a execução das aplicações. Dessa forma, é importante que as aplicações desenvolvidas para este tipo de sistema sejam **tolerantes a falhas**, para que a falha ou remoção de nós do sistema não cause falha ou interrompa a aplicação como um todo. Também é importante que as aplicações sejam capazes de fazer uso de novos recursos computacionais, integrados ao sistema após a aplicação ter sido iniciada.

3.1.2. Aplicações com acesso intensivo a dados

Aplicações com acesso intensivo a dados fazem acesso a arquivos de dados intensivamente durante a execução. Para este tipo de aplicação, é importante que o sistema computacional tenha mecanismos de alto desempenho para acesso aos arquivos. Caso contrário, o tempo de execução total da aplicação poderia ser afetado significativamente pelo atraso no acesso aos dados. As seguintes abordagens podem ser utilizadas para permitir o acesso com alto desempenho a arquivos:

1. Armazenamento dos arquivos em um sistema de arquivos centralizado e compartilhado. Neste caso, os arquivos são armazenados em um subsistema especializado e, à medida que os nós computacionais tentam acessar os arquivos, o conteúdo destes é transferido pela rede.

2. Armazenamento dos arquivos em um sistema de arquivos distribuído e compartilhado. Neste caso, os arquivos são armazenados de forma distribuída nos próprios nós computacionais. À medida que os processos que executam nos nós computacionais tentam acessar os arquivos, os dados são lidos da unidade de armazenamento local (caso haja uma cópia do arquivo na máquina) ou transferidos de outro nó computacional pela rede. A replicação parcial ou total de arquivos é comum nesta abordagem.

3. Armazenamento de cópias dos arquivos na unidade de armazenamento local do nó computacional. Neste caso, os dados devem ser copiados para as unidades de armazenamento dos nós computacionais antes da computação ser iniciada.

A primeira e a segunda abordagem são transparentes para o usuário, já que o mesmo acessa os arquivos como se eles existissem no nó computacional onde o processo é executado, mas elas colocam mais pressão na rede de comunicação do sistema, já que esta é utilizada para transmitir o conteúdo dos arquivos à medida que eles são lidos. A terceira abordagem, por outro lado, não utiliza a rede do sistema para transferência de arquivos durante a execução da tarefa e pode oferecer maior paralelismo, no entanto, o usuário deve coordenar a cópia dos dados de entrada (saída) para os (dos) nós computacionais do sistema.

A intensidade com que a aplicação acessa os dados é inversamente proporcional à "**intensidade operacional**" da aplicação, que corresponde ao número de operações realizadas para cada *byte* de dados lido. Quanto maior a "intensidade operacional", mais

[3] A ativação do protetor de tela do computador do usuário é muitas vezes usada como gatilho para indicar que o sistema está ocioso.

tempo o sistema passa processando e menos tempo acessando dados dos arquivos. Dessa forma, a intensidade de acesso aos dados não está ligada diretamente ao tamanho do dado de entrada. Observe, por exemplo, que uma aplicação que acessa dados grandes (p.ex. dados com *terabytes* ou *petabytes*) mas tem alta intensidade operacional não faz acesso intensivo aos dados.

Aplicações com baixa intensidade operacional, que fazem poucas operações de processamento para cada *byte* lido do arquivo, são sensíveis ao desempenho do subsistema de leitura e escrita de arquivos. Aplicações de *Big Data*, que fazem busca e mineração em grandes volumes de dados, são exemplos de aplicações com baixa intensidade operacional pois acessam um volume muito grande de dados e a quantidade de operações realizadas por *byte* lido é pequena. Este tipo de aplicação é comumente acelerado com sistemas que mantêm os dados armazenados de forma distribuída, em múltiplos nós de processamento, e movem o código para execução nos nós distribuídos. O sistema Hadoop é um exemplo de sistema que processa dados com esta abordagem.

3.1.3. Considerações sobre o desempenho de aplicações de alto desempenho

O desempenho de uma aplicação de alto desempenho depende das características da aplicação e do sistema computacional que será utilizado para a execução desta. Características como o número de núcleos computacionais, a frequência de operação da CPU, a quantidade de vias nas unidades vetoriais, o tamanho das *caches* do processador, a vazão e a latência da memória principal, dos dispositivos de armazenamento secundários e da rede de interconexão e até mesmo as características de aceleradores em *hardware* (p.ex. GPUs) podem afetar o desempenho de uma aplicação de alto desempenho. Entretanto, geralmente é um subconjunto pequeno destas características (muitas vezes apenas uma) que causa o maior impacto no desempenho da aplicação e este subconjunto depende das características da aplicação. Por exemplo, como discutido anteriormente, o desempenho de aplicações fortemente acopladas pode ser fortemente afetado pelo desempenho da rede de interconexão.

Entender como a aplicação interage com os diferentes recursos do sistema computacional, identificar os principais gargalos (limitantes de desempenho) e otimizar o código para o sistema em questão é geralmente uma tarefa que demanda bastante conhecimento da aplicação e do sistema computacional. Esse processo pode envolver o desenvolvimento de modelos de desempenho, a realização de experimentos de desempenho e a modificação do código da aplicação. Dessa forma adequar uma aplicação a um *cluster* de alto desempenho pode ser um desafio à parte.

Apesar do processo de otimização de aplicações para *clusters* específicos estar fora do escopo deste minicurso, podemos fazer as seguintes considerações gerais:

- Aplicações fortemente acopladas geralmente dependem de um bom desempenho da rede de interconexão. Dessa forma, a execução deste tipo de aplicação em sistemas com uma rede de interconexão de alto desempenho pode ser mais vantajosa. É importante notar que, em alguns casos, a rede só se torna o gargalo do sistema quando o número de nós computacionais utilizados para a computação é muito grande. Dessa forma, aplicações fortemente acopladas também podem funcionar

bem em sistemas com redes de interconexão simples quando a quantidade de nós computacionais não for muito grande.

- Aplicações fracamente acopladas geralmente não dependem muito do desempenho da rede de interconexão. Nestes casos, é mais vantajoso investir em outros tipos de recursos computacionais, como CPUs, memória e/ou dispositivos de armazenamento secundários de alto desempenho.

- Aplicações programadas com MPI podem tirar proveito de sistemas com múltiplos nós e de nós com múltiplos núcleos computacionais, ou *multi-core*. Para isso, basta executar um processo por núcleo computacional, ou *core*. Neste cenário, há aplicações que atingem melhor desempenho em sistemas com muitos nós com poucos núcleos computacionais e outras que executam de forma mais rápida em poucos nós com muitos núcleos computacionais. Vale à pena realizar testes para determinar qual é a melhor estratégia para a aplicação de interesse.

- Aplicações com baixa intensidade operacional colocam mais pressão no sistema de entrada e saída. Nestes casos, é importante otimizar o acesso aos arquivos; por exemplo, através da replicação destes nos dispositivos de armazenamento local dos nós computacionais. A aquisição de dispositivos de armazenamento secundários com melhor desempenho, como *Flash Drives*, pode melhorar ainda mais o desempenho nestes casos.

- Aplicações ***CPU-bound***, que dependem mais do desempenho da CPU do que dos outros componentes do sistema, geralmente podem ser aceleradas significativamente com o uso de aceleradores em *hardware*, como GPUs. No entanto, para isso, a aplicação deve ter sido codificada para fazer uso destes recursos com linguagens como CUDA ou OpenCL, por exemplo.

3.2. Computação na Nuvem

3.2.1. Definição e modelos de negócios

De acordo com a definição do NIST [Mell and Grance 2011], computação na Nuvem é um modelo para permitir acesso ubíquo, conveniente e sob demanda via rede a um conjunto de recursos computacionais configuráveis (p. ex: redes de computadores, servidores, armazenamento, aplicações, e serviços) que possam ser provisionados e descartados rapidamente e com pouco esforço de gerenciamento ou interação com o provedor de serviço. Este modelo é composto por cinco características essenciais:

- Autosserviço sob demanda: O cliente pode provisionar ou descartar recursos de computação, como armazenamento e processamento, sob demanda sem a necessidade de interagir com humanos durante o processo;

- Acesso amplo pela rede: O serviço deve estar disponível para uso através da rede de computadores por mecanismos padronizados;

- *Pool* de recursos: Os recursos computacionais do provedor devem atender a múltiplos clientes, com diferentes recursos físicos e virtuais atribuídos e re-atribuídos dinamicamente para os clientes sob demanda;

- Elasticidade rápida: Recursos devem ser provisionados e descartados de forma elástica e rápida, e até automaticamente em alguns casos, para permitir um ajuste dinâmico à demanda de serviço. Este tipo de característica permite que clientes adequem os recursos computacionais à sua necessidade ao longo do tempo. A Figura 3.3 ilustra como a elasticidade rápida pode trazer benefícios como redução de custo ou aumento da qualidade de serviço. A Figura 3.3a representa um cenário onde o sistema foi dimensionado para suprir a demanda de pico e nos momentos onde a demanda não é alta o recurso computacional é subutilizado, representado pela área escura. A Figura 3.3b ilustra as situações onde as projeções iniciais de demanda de recursos foram infladas e o sistema adquirido passa a ser sempre subutilizado. A Figura 3.3c ilustra o caso oposto, onde o sistema é subdimensionado. Neste caso, os usuários têm uma qualidade de serviço inferior durante os períodos de alta demanda, o que pode causar atrasos ou mesmo decréscimo na produtividade. Um sistema com elasticidade rápida é ilustrado na Figura 3.3d, onde a quantidade de recursos é ajustada rapidamente para se adequar à demanda do cliente;
- Serviços mensuráveis: Os serviços oferecidos pelo provedor devem ser mensuráveis, de forma a permitir o monitoramento e controle automático do uso dos recursos.

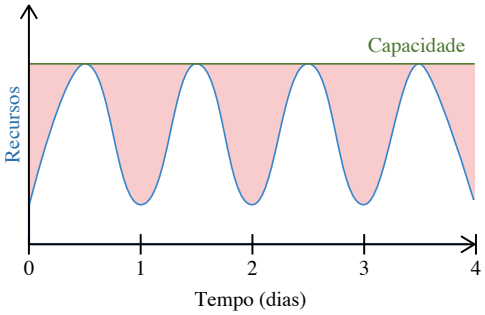
Em suma, estas características permitem que a Nuvem ofereça acesso sob demanda, conveniente e ubíquo a seus serviços de forma que a quantidade de recursos possa ser incrementada ou reduzida rapidamente com pouco esforço de gerenciamento e nenhuma interação humana com o provedor de serviços.

3.2.1.1. Modelos de serviço e implantação

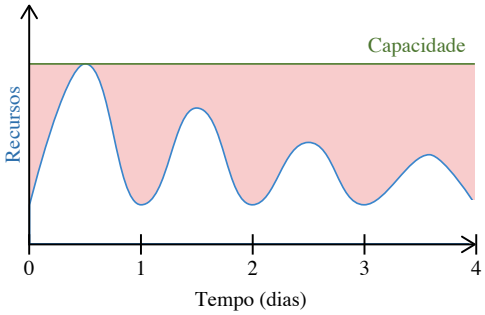
Os serviços oferecidos em uma Nuvem Computacional são classificados em três modelos principais [Mell and Grance 2011, Puthal et al. 2015]:

- *Software* como Serviço, ou *Software as a Service* (SaaS): neste modelo o serviço oferecido é um *software* que é executado pelo provedor da Nuvem. Gmail, Google Drive e Bing são exemplos de aplicações que empregam o modelo *Software* como Serviço.
- Plataforma como Serviço, ou *Platform as a Service* (PaaS): neste modelo o serviço oferecido é uma plataforma de *software* base para o desenvolvimento ou instalação de aplicações pelo cliente. Neste modelo, a maior parte da pilha de *software*, como o sistema operacional e as bibliotecas de *software*, é gerenciada pelo provedor e o cliente não tem que se preocupar com atividades como atualizações das bibliotecas e do sistema operacional. O Google App Engine[4] é um exemplo de plataforma como serviço.
- Infraestrutura como Serviço, ou *Infrastructure as a Service* (IaaS): neste modelo de serviço o cliente recebe a infraestrutura bruta, como máquinas virtuais e espaços

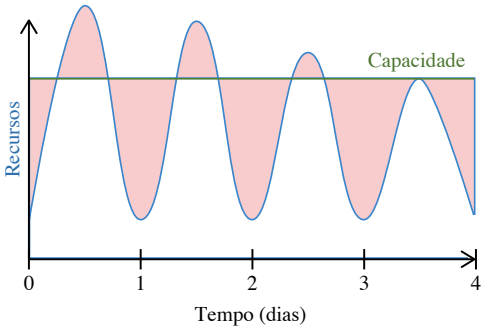
[4] `http://code.google.com/appengine`



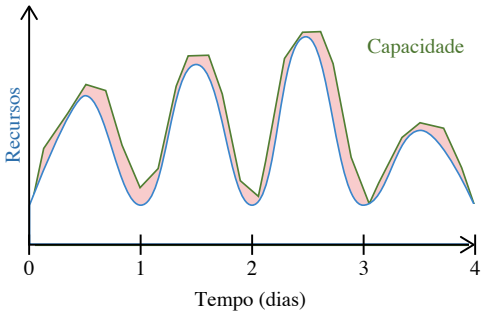
(a) Infraestrutura de computação adquirida com dimensionamento baseado na demanda de pico. Ocorre o desperdício de recursos durante períodos de baixo uso.



(b) Infraestrutura de computação adquirida com superdimensionamento causado por projeções infladas de demanda.



(c) Infraestrutura subdimensionada. A qualidade de serviço inferior durante os períodos de alta demanda pode causar atrasos ou mesmo decréscimo na produtividade.



(d) Infraestrutura com ajuste elástico rápido, que permite o ajuste dinâmico dos recursos de acordo com a demanda.

Figura 3.3: Cenários de provisionamento de recursos. A área escura representa recursos não utilizados ou sub-provisionados. Figuras (a), (b) e (c) mostram casos onde não há elasticidade, fazendo com que os recursos sejam sub ou superdimensionados. A Figura (d) mostra um caso onde é possível realocar recursos ao longo do tempo de acordo com a demanda real.

para armazenamento, e é responsável pela instalação e gerenciamento da pilha de *software* que será executada no sistema. Este modelo é geralmente mais flexível e possui um custo de uso menor, entretanto, os custos associados ao gerenciamento da pilha de *software* do sistema ficam a cargo do cliente.

A Figura 3.4 compara a responsabilidade do gerenciamento dos recursos em um *cluster* privado e nos diferentes modelos de serviços na Nuvem Computacional. Note que no *cluster* privado a responsabilidade de gerenciamento dos recursos (incluindo *software* e *hardware*) é integralmente do usuário (proprietário) enquanto que no modelo Infraestrutura como Serviço na Nuvem Computacional o usuário é responsável por gerenciar apenas parte do *software* (Sistema Operacional - S.O., *Runtime*, Dados e Aplicações).

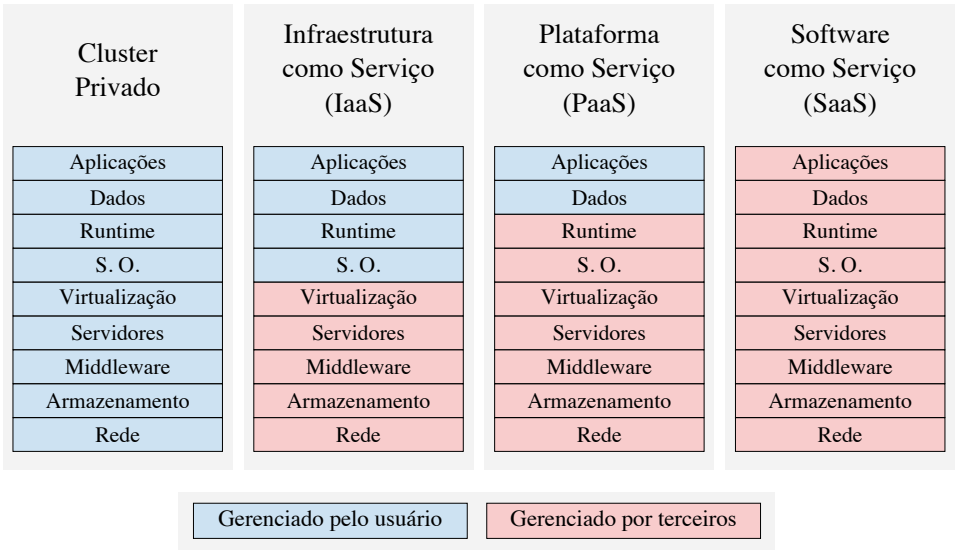


Figura 3.4: Gerenciamento de recursos em um *cluster* privado e nos diferentes modelos de serviço na nuvem.

As nuvens computacionais também podem ser classificadas com relação ao modelo de implantação. No modelo de implantação "Privado", a Nuvem construída sobre uma infraestrutura privada e é acessada exclusivamente pelos proprietários da infraestrutura. Uma empresa grande com diversos setores que fazem uso de recursos computacionais pode fazer uso do modelo "Privado" de Nuvem Computacional para o compartilhamento dos recursos computacionais entre os diferentes setores. No modelo "Público", os serviços são oferecidos ao público em geral e tipicamente cobrado com base nas métricas de uso. A Amazon AWS[5] e a Microsoft Azure [6] são exemplos de nuvens públicas que cobram em função do uso. No modelo de implantação "Híbrido" há a junção de nuvens privadas com nuvens públicas de forma que a Nuvem privada possa ter seus recursos computacionais ampliados em momentos de maior demanda a partir da contratação de recursos da Nuvem pública.

Historicamente, a nuvem computacional tem sido usada amplamente para aplicações *web*, onde *sites* se aproveitam da alta disponibilidade e dos mecanismos de auto dimensionamento, que provedores de nuvem fornecem, para manter os sistemas operantes e com cargas de trabalho distribuída entre diversos nós de computação. Nos primórdios, as tecnologias de emulação de máquina virtual não eram tão eficientes e a rede de intercomunicação entre os nós era rápida para aplicações da Internet, mas não o suficiente para aplicações de alto desempenho. Com os avanços nas tecnologias de virtualização e as vantagens da nuvem computacional, aplicações de alto desempenho, que não requerem uma rede de alta velocidade, começaram a tirar proveito do uso de recursos da nuvem

[5] `http://aws.amazon.com/`
[6] `http://www.microsoft.com/windowsazure/`

computacional. Do outro lado, com a demanda, os provedores destes serviços passaram a disponibilizar infraestruturas de rede de alta velocidade - como InfiniBand.

3.2.2. Considerações de custo e desempenho na Nuvem Computacional

Como discutido na Seção 3.1.3, adequar uma aplicação a um *cluster* de alto desempenho pode ser um desafio à parte. Entretanto, o modelo de serviço da Nuvem Computacional permite que o usuário realize o processo inverso, ou seja, adeque o *cluster* às necessidades da aplicação. Neste contexto, em vez do usuário tentar adequar a aplicação para executar de forma eficaz no *cluster*, o usuário pode testar a aplicação em diferentes configurações de máquinas virtuais na Nuvem Computacional e escolher o "*cluster*" que melhor satisfaz as necessidades da aplicação.

No modelo de Infraestrutura como Serviço, ou *Infrastructure as a Service* (IaaS), o cliente geralmente possui à disposição uma ampla gama de tipos de máquinas virtuais para contratar. Cada tipo possui características de recursos (p. ex.: tipo e número de núcleos computacionais, quantidade de memória, *etc*) e custos diferentes. Sendo que o custo pode variar de poucos centavos até dezenas de dólares por hora de uso. Dessa forma, é muito importante que o usuário escolha máquinas apropriadas para a aplicação que deseja executar. As principais variáveis que afetam o custo das máquinas virtuais são:

- O modelo do processador;
- A quantidade de núcleos (*cores*) do processador;
- A quantidade de memória RAM;
- A quantidade e o modelo dos aceleradores (p.ex. GPUs e FPGAs);

Para a escolha do tipo de máquina virtual com o melhor custo-benefício, o usuário deve analisar as características da aplicação de interesse. Por exemplo, analisar se a aplicação foi codificada para tirar proveito de múltiplos núcleos computacionais ou de GPUs. Note que executar uma aplicação que não foi codificada para usar GPUs em uma máquina virtual que possui GPUs não oferecerá melhor desempenho enquanto que o custo será maior. Alternativamente, em vez de analisar a aplicação, o usuário pode simplesmente "testar" o desempenho da aplicação em diferentes configurações e selecionar a configuração que oferece o melhor custo-benefício.

Além das características mencionadas anteriormente, o desempenho do sistema de armazenamento e o desempenho da rede (ambos contratados pelo cliente), também afetam o custo do processamento. Em particular, o desempenho da rede era uma das características que dificultavam bastante a execução de aplicações de alto desempenho na Nuvem Computacional. De fato, há alguns anos atrás, quando os provedores públicos de serviço de Nuvem Computacional estavam se consolidando, era difícil utilizar serviços da Nuvem Computacional para computação de alto desempenho com aplicações fortemente acopladas por causa da latência da rede. Naquela época, diversos estudos indicavam que as tecnologias de virtualização afetavam significativamente a latência de comunicação entre as máquinas virtuais [He et al. 2010, Mag 2011, Gupta et al. 2013]. O relatório desenvolvido pela Magellan [Mag 2011], por exemplo, apontava que a Nuvem Computacional

poderia ser até 50 vezes mais lenta para um determinado tipo de padrão de comunicação e que as aplicações mais penalizadas eram aquelas que dependiam de baixa latência de rede (fortemente acopladas) enquanto que aplicações que dependiam de vazão da rede, mas não muito da latência (poucas trocas de mensagens grandes), não eram tão afetadas.

Atualmente, o serviço de rede na Nuvem Computacional melhorou e os principais provedores de serviço já oferecem opções especializadas para melhorar o desempenho de aplicações de alto desempenho. Por exemplo, a AWS oferece o Elastic Network Adapter (ENA) [Barr 2016, AWS 2018], um serviço de rede que pode ser configurado para trabalhar com 10 ou 25 Gbps. A Azure oferece o serviço RDMA (Remote Direct Memory Access)[Karmarkar 2015, Azure 2018], que implementa uma rede Infiniband com desempenho de 32 Gbit/s Estes serviços mostram que os provedores estão oferecendo serviços cada vez mais compatíveis com as demandas de aplicações fortemente acopladas e a tendência é que a Nuvem Computacional poderá ser utilizada para qualquer tipo de aplicação de alto desempenho.

Estes são aspectos importantes que devem ser levados em consideração pelo usuário quando estiver contratando serviços na Nuvem Computacional para a execução de aplicações de alto desempenho.

3.3. Executando aplicações de HPC na nuvem

Nesta seção iniciaremos a parte prática do minicurso, entrando em contato com um provedor de nuvem, inicializando e explorando a utilização de sua infraestrutura no modelo IaaS. Atualmente os dois principais fornecedores de serviços de Nuvem Computacional são a Microsoft, com a plataforma *Azure*, e a Amazon com a plataforma *Amazon Web Services* (AWS). O conceito de uso de ambas plataformas é semelhante em muitos aspectos e uma noção do uso em uma delas pode guiar para o uso na outra. Dessa forma, concentraremos as atividades práticas deste minicurso apenas na AWS. Ressalta-se que o processo de escolha do melhor tipo de serviço (incluindo tipos de máquina virtual) para cada aplicação não faz parte do escopo deste minicurso.

3.3.1. Fluxo de trabalho no modelo IaaS

Para se executar aplicações em máquinas virtuais na Nuvem Computacional, o usuário deve realizar uma sequência de passos. Estas atividades estão representadas na Figura 3.5 e incluem:

1. Criar uma conta no *website* do provedor de serviço na Nuvem Computacional;

2. Selecionar a **imagem de máquina virtual** para ser instanciada. A imagem é um arquivo que contém o sistema operacional e pacotes de *software*. Geralmente os provedores oferecem diversas opções de imagens, incluindo imagens com o S.O. Linux e o S.O. Windows. Além disso, algumas dessas imagens possuem *software* pré-instalado. Nestes casos, o provedor pode cobrar a mais pelo uso da imagem em função do conjunto de *software* instalados;

3. Selecionar o **tipo de máquina virtual** que executará a imagem escolhida no passo anterior. O tipo de máquina virtual define os recursos em *hardware*, incluindo o

número de núcleos computacionais, a quantidade de memória, as interfaces de rede, os aceleradores (e.g. GPUs), entre outros;

4. Configurar as interfaces de rede, chaves de acesso, usuários, *etc*;
5. Ligar a máquina virtual, ou VM[7];
6. Conectar-se e utilizar a VM com *software* de acesso remoto, por exemplo, *secure shell* (SSH) ou Remote Desktop;
7. Desconectar e desligar a VM;
8. Destruir a VM.

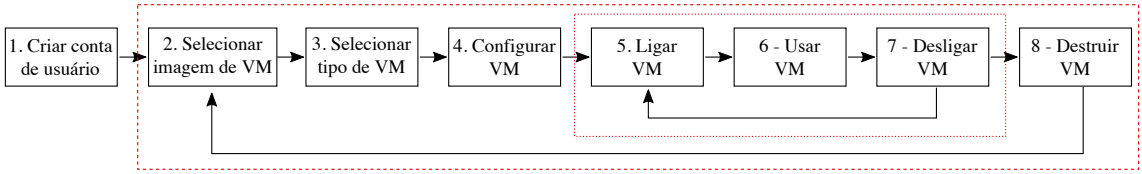


Figura 3.5: Diagrama do fluxo de trabalho no modelo IaaS.

3.3.2. Passos na AWS

3.3.2.1. Criando uma conta no provedor

Como descrito na Seção 3.3.1, o primeiro passo para a execução de código é a criação de uma conta de usuário no *website* do provedor de serviço na Nuvem Computacional. Como este processo pode diferir significativamente para cada provedor, não detalharemos o mesmo aqui.

3.3.2.2. Selecionando um imagem na AWS

Para selecionar uma imagem de VM na AWS, o primeiro passo é acessar os serviços EC2[8] no menu do provedor, como indicado na Figura 3.6.

Após acessar o serviço EC2, você deve acessar a opção "*Launch Instance*" para dar início ao processo de seleção de imagem e tipo de VM, como indicado na Figura 3.7.

Uma vez que a opção "*Launch Instance*" foi acessada, o provedor mostrará uma lista de imagens (chamadas AMIs[9] na AWS) com diferentes configurações de sistemas operacionais e *softwares*, como ilustrado na Figura 3.8.

Selecione a imagem denominada "**Deep Learning AMI (Ubuntu)**" na versão mais recente. A imagem na versão 13 possui o sistema operacional Linux Ubuntu na versão 16.04 LTS e os drivers de CUDA e OpenCL para GPUs NVidia previamente instalados.

[7] do inglês: *Virtual Machine*
[8] do inglês: *Elastic Cloud Computing*.
[9] do inglês: *Amazon Machine Images*.

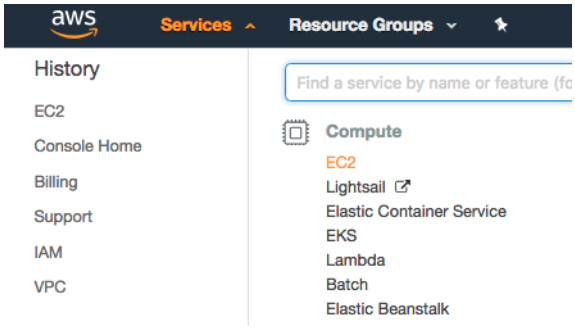


Figura 3.6: Seleção do serviço EC2.

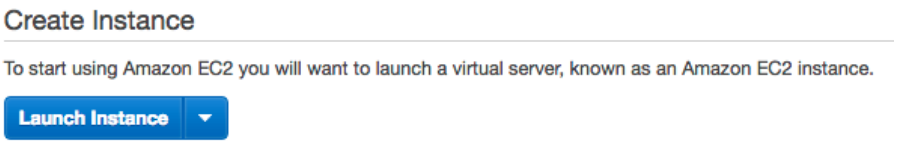


Figura 3.7: Criação de máquina virtual.

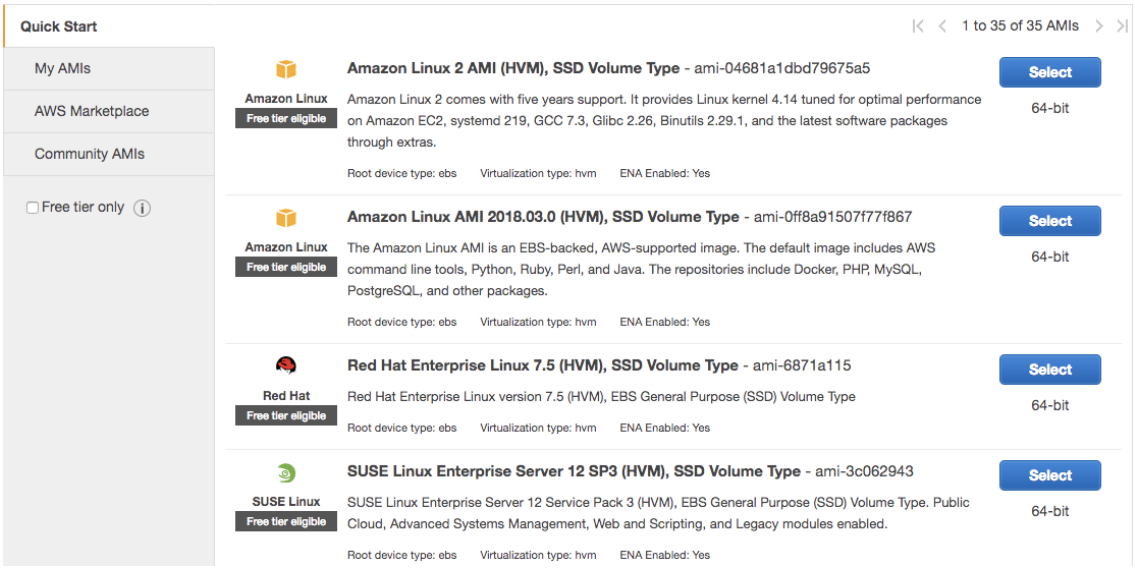


Figura 3.8: Lista de imagens (AMIs) disponíveis.

3.3.2.3. Selecionando um tipo de VM na AWS

Após selecionar a imagem, o provedor exibirá uma tela com múltiplas opções onde você poderá selecionar o tipo de máquina virtual que deseja instanciar, como mostrado na Figura 3.9

Selecione o tipo de máquina virtual intitulada "p2.xlarge", que possui uma GPU NVidia K80, e clique no botão "*Next: Configure Instance Details*" para prosseguir para a etapa de configuração da VM. (**NOTA: cuidado para não clicar em "*Review and Launch*", pois pretendemos configurar a VM antes**)

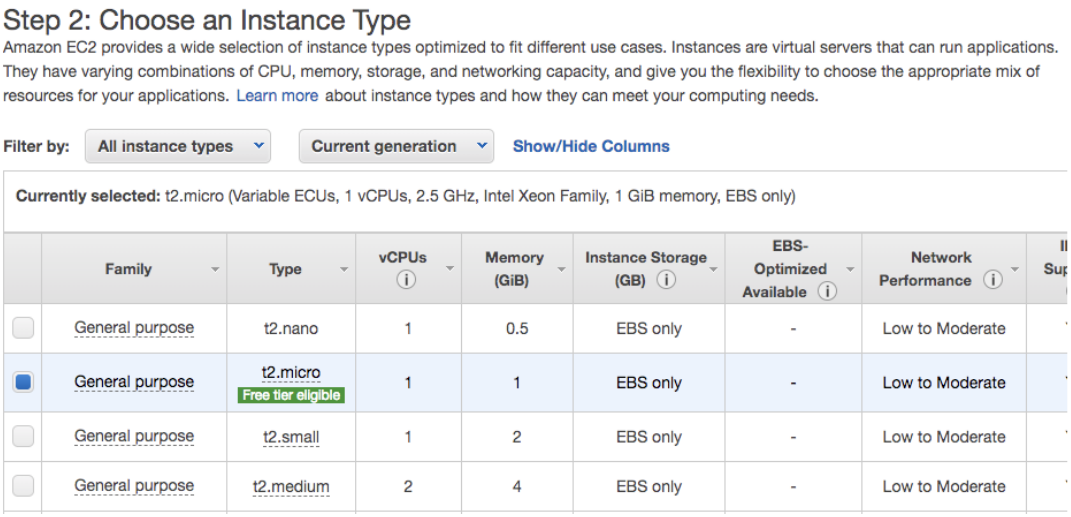


Figura 3.9: Lista de tipos de máquinas virtuais disponíveis.

3.3.2.4. Configurando a VM selecionada

Após selecionar o tipo de máquina virtual, o provedor mostrará um conjunto de telas com parâmetros para configuração da máquina virtual. A Figura 3.10 mostra a primeira tela deste conjunto.

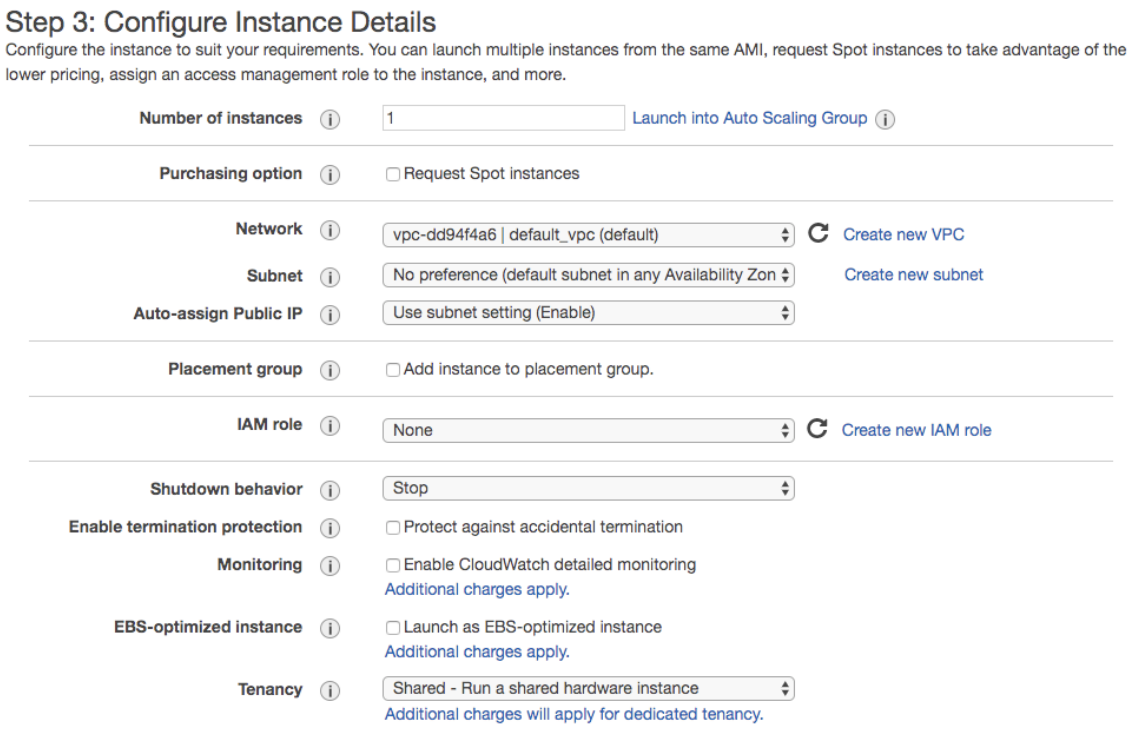


Figura 3.10: Configuração de detalhes da instância.

Para nosso experimento, você não precisará mudar os parâmetros desta primeira tela, que devem se parecer com os da Figura 3.10. Neste ponto, você deve clicar em "*Next:*

Add Storage" (este botão foi omitido da Figura 3.10) para configurarmos o dispositivo de armazenamento da máquina virtual.

A tela seguinte, intitulada "*Add Storage*" permite que o usuário selecione e configure os dispositivos de armazenamento da VM. Para nossos experimentos, utilizaremos o dispositivo padrão, como ilustrado na Figura 3.11. Para prosseguirmos com a configuração selecione a opção "*Next: Add Tags*".

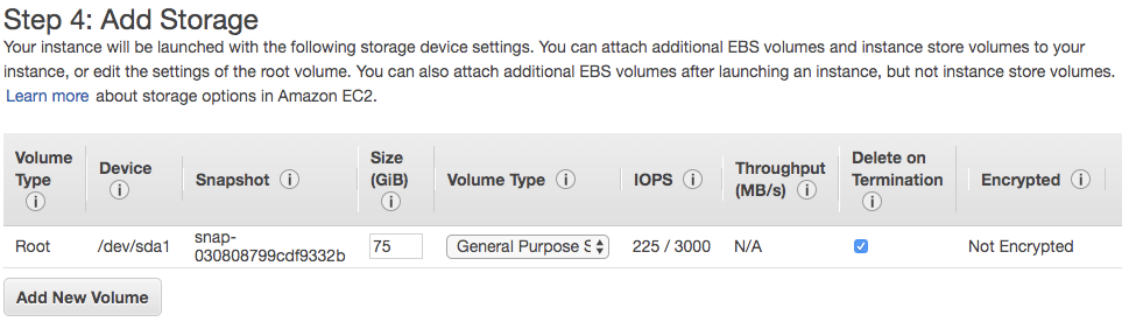


Figura 3.11: Configuração de armazenamento.

Em muitas situações é útil você atribuir *Tags* (etiquetas) às máquinas virtuais. Nesta atividade, você criará uma *Tag* que nos ajudará a identificar os usuários das máquinas virtuais durante o minicurso. Para isso, escreva "*Student*" na caixa "*Key*" e o seu nome (*e.g.* Edson) na caixa "*Value*", como indicado na Figura 3.12. Por fim, clique em "*Next: Configure Security Group*" para avançar para a próxima tela de configuração.

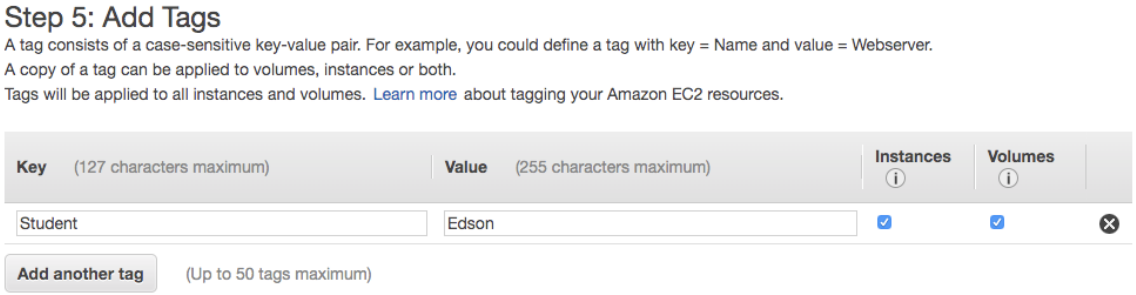


Figura 3.12: Definição de rótulos.

Na tela "*Configure Security Group*", você terá a oportunidade de configurar regras de acesso à máquina, como mostrado na Figura 3.13. Por padrão, o protocolo SSH já está habilitado na porta 22. Para nossos experimentos, não há necessidade de modificar estas configurações. Dessa forma, basta clicar em "*Review and Launch*" para exibir um resumo geral das configurações.

A tela intitulada "*Review Instance Launch*" mostra um resumo das configurações, como ilustrado na Figura 3.14. Você pode revisar as opções selecionadas e, por fim, iniciar a execução da máquina virtual clicando no botão "*Launch*".

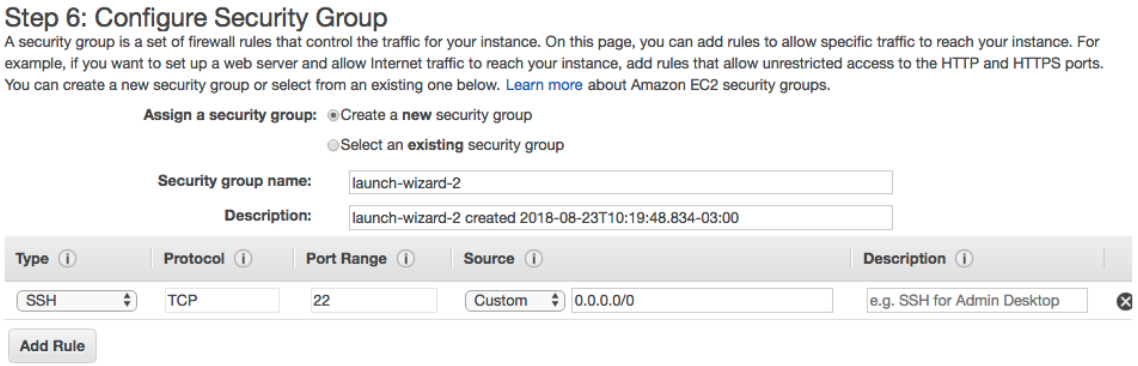


Figura 3.13: Configuração das regras de acesso à rede.

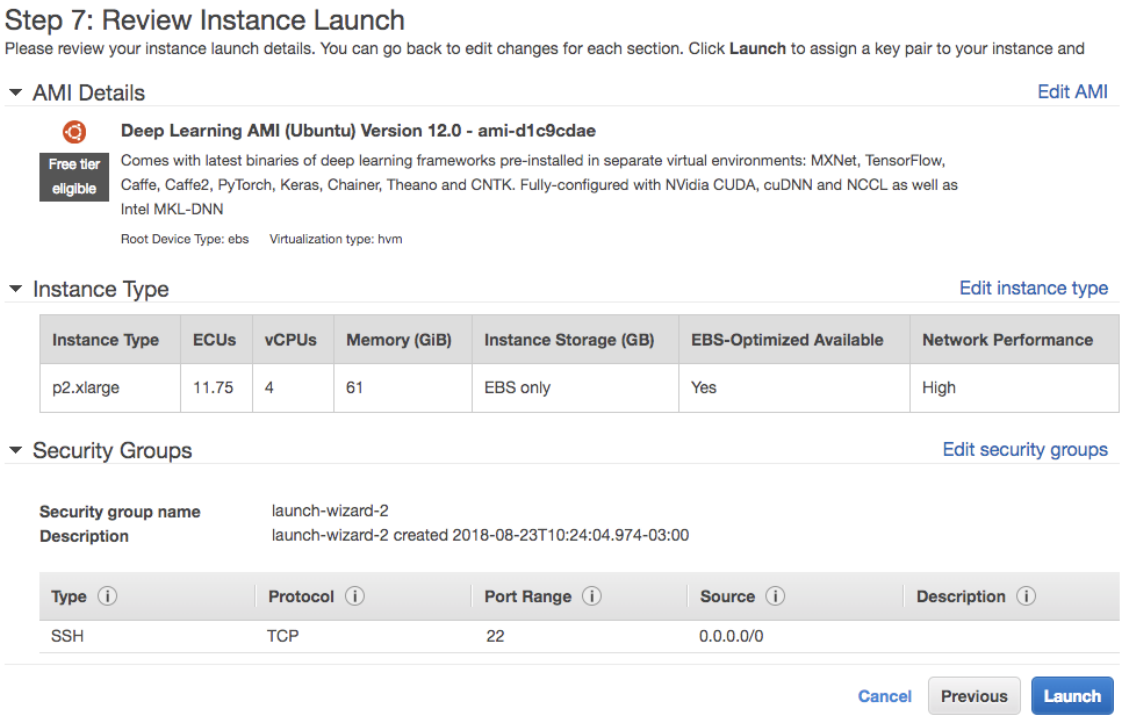


Figura 3.14: Resumo das configurações selecionadas.

3.3.2.5. Ligando a VM

Ao clicar em "*Launch*", antes de iniciar a máquina virtual, o provedor solicitará que você forneça a chave de acesso. Esta chave é importante para garantir que apenas você tenha acesso a esta máquina. Caso você não tenha uma chave, você pode criar selecionando a opção "*Create a new key pair*" e digitando um nome para a sua nova chave, como indicado na Figura 3.15. Após dar um nome à nova chave, você deve realizar o "*download*" desta chave para o seu computador clicando em "*Download Key Pair*". Dica: evite o uso de caracteres especiais como nome de chave.

Após realizar o *download* da chave, clique em "*Launch Instances*" para iniciar a máquina virtual. Neste momento, a máquina virtual será iniciada e o provedor exibirá

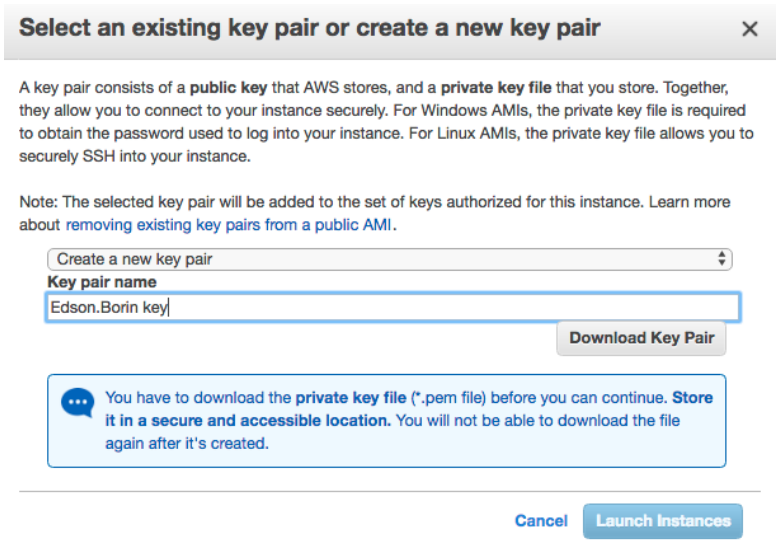


Figura 3.15: Definição da chave de acesso para a máquina virtual.

uma tela com informações sobre o processo, como mostrado na Figura 3.16. Por fins de segurança, esta é a única oportunidade que você terá para realizar o *download* desta chave.

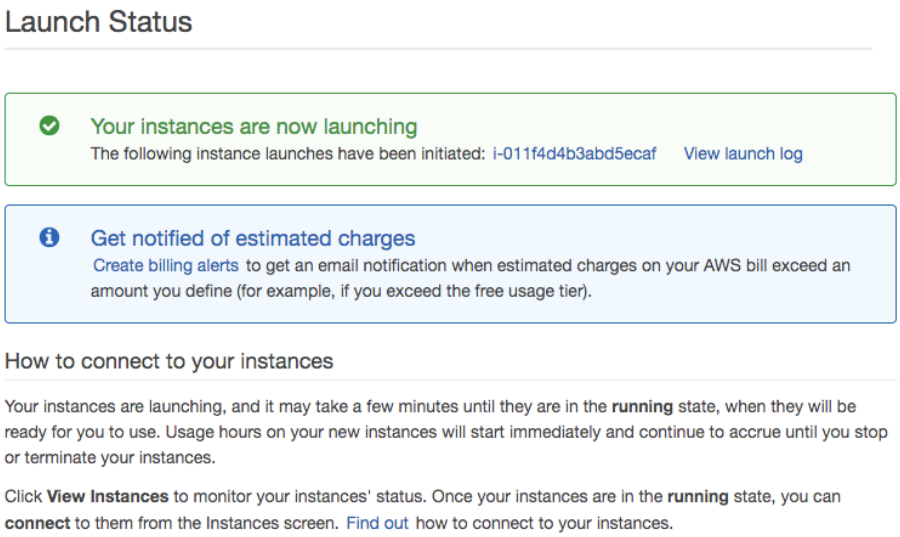


Figura 3.16: Informações da máquina virtual em criação.

Note que enquanto o sistema operacional estiver sendo iniciado (processo de *boot*), a máquina ainda não é acessível. Você pode clicar em "*View Instances*" para visualizar a máquina virtual criada e seu estado. Após o campo "*Status Checks*" exibir 2/2 *checks*, a máquina virtual estará pronta para o acesso remoto.

3.3.2.6. Acessando e utilizando a VM remotamente

A tela "Instances", exibida após você clicar em "*View Instances*" no passo anterior, também pode ser acessada pelo menu "*Instances*", à esquerda do *Dashboard* exibido após a escolha do serviço EC2. Para se conectar remotamente a uma instância, selecione a instância e clique em "*Connect*", como indicado na Figura 3.17

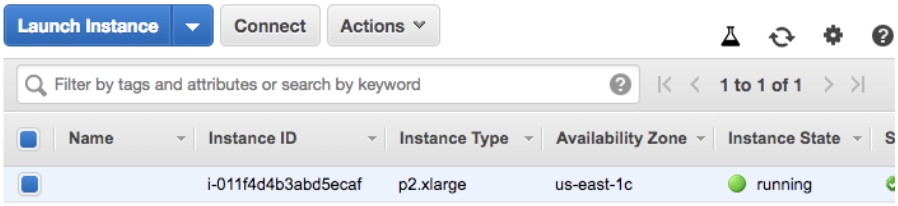


Figura 3.17: Instâncias criadas.

Após clicar em "*Connect*", o *website* exibirá uma tela com instruções de conexão, como mostrado na Figura 13. Observe neste exemplo, que a opção padrão é "*A standalone SSH client*", que pode ser o programa ssh em linha de comando disponível nas distribuições Linux e no sistema OSX, ou o programa PuTTY, comumente utilizado nos sistemas Windows.

Caso você esteja usando o Linux ou o OSX, você deve ajustar as permissões do arquivo de chaves que você obteve nos passos anteriores (Veja os passos 2 e 3 na Figura 3.18). Uma vez que você ajustou as permissões, basta acessar a máquina virtual com o comando sugerido. No caso abaixo foi o comando:

```
ssh -i "Edson.Borin key.pem" ubuntu@ec2-34-227-227-217.compute...
```

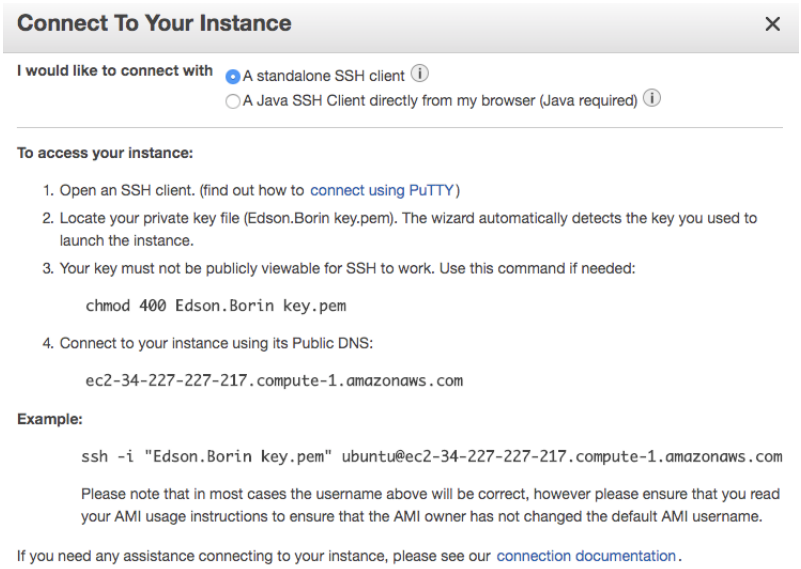


Figura 3.18: Instruções de conexão com a máquina.

Os próximos acessos a esta máquina virtual podem ser realizados repetindo-se este passo. Neste ponto, você pode instalar ou utilizar o seu *software* na máquina virtual.

3.3.2.7. Desligando a VM

Enquanto a máquina virtual estiver ligada, o provedor cobrará pelo serviço. Isso acontece mesmo que você não esteja executando programas dentro da máquina virtual. Por isso, sempre que finalizar o trabalho, é importante desligar a máquina virtual. Para isso, você pode ir na tela "Instances", selecionar a máquina virtual de interesse, clicar em "Actions" e selecionar as opções "Instance State" e "Stop", como ilustrado na Figura 3.19.

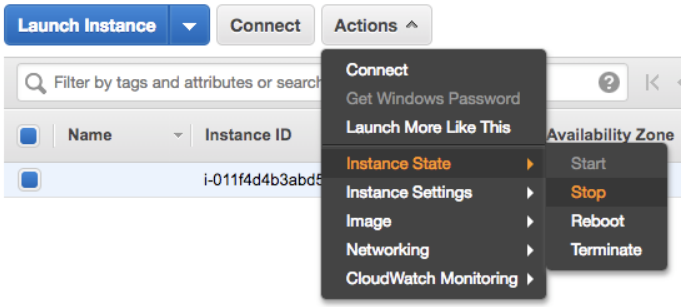


Figura 3.19: Interrupção da máquina virtual.

Uma vez que a máquina virtual foi desligada, o provedor cobra apenas pelo espaço de armazenamento necessário para armazenar o disco da máquina virtual. Observe, que você pode religar a máquina virtual posteriormente que seus dados e programas instalados estarão preservados.

3.3.2.8. Destruindo a VM

Quando a máquina virtual não tem mais utilidade, é importante destruí-la para que o armazenamento da mesma não gere custos. Este processo é muito similar ao processo descrito na Seção 3.3.2.7 ("Desligando a VM"). Neste caso, basta você selecionar a opção "*Terminate*" em vez de "*Stop*". Ainda não destrua esta máquina virtual, a utilizaremos para os demais passos. Note que, uma vez destruída, o conteúdo da máquina virtual não pode ser recuperado.

3.3.3. Estudo de caso 1: Executando a aplicação CMP em máquinas com GPU

Neste estudo de caso, executaremos uma aplicação que faz uso de GPUs em uma máquina virtual. Inicialmente, você deve selecionar a imagem e o tipo de máquina virtual, configurar, ligar e acessar remotamente a máquina virtual, como descrito na Seção 3.3.2. Para este experimento utilizaremos a imagem intitulada "**Deep Learning AMI (Ubuntu)**" na versão mais recente e o tipo de máquina virtual "p2.xlarge".

3.3.3.1. Transferindo e compilando a aplicação CMP

Após acessar a máquina virtual, faça clone do repositório "GPU Seismic Processing" usando o comando:

```
git clone https://github.com/hpg-cepetro/IPDPS-CRS-CMP-code.git
```

Este repositório contém código de duas aplicações de processamento sísmico que funcionam com OpenMP, OpenACC, CUDA e OpenCL [Gimenes et al. 2018]. Estas aplicações foram desenvolvidas por pesquisadores do *High Performance Geophysics lab* (HPG), que se localiza no Centro de Estudos de Petróleo (CEPETRO) da Universidade Estadual de Campinas (Unicamp). Neste minicurso nós executaremos a versão OpenCL/OpenMP da aplicação CMP na GPU/CPU.

Utilizaremos um dado de entrada chamado "simple-synthetic", produzido pelos pesquisadores do HPG para a validação de métodos de processamento sísmico. O dado pode ser copiado com o seguinte comando:

```
wget www.ic.unicamp.br/~edson/minicurso-hpc/simple-synthetic.su
```

Após copiar o repositório, você deve compilar a aplicação. Em nossos experimentos utilizaremos o compilador GCC, disponível na imagem de VM selecionada. Para compilar a aplicação, primeiro você deve entrar no diretório que contém a implementação do CMP em OpenCL utilizando o comando:

```
cd ~/IPDPS-CRS-CMP-code/CMP/OpenCL/
```

Depois, você deve criar um subdiretório intitulado "`bin`", entrar neste diretório, realizar a configuração do sistema de *build* e disparar a compilação com os seguintes comandos:

```
mkdir bin
cd bin
cmake ../
make -j
```

3.3.3.2. Executando a aplicação CMP

Para manter o diretório organizado, criaremos um subdiretório intitulado "`results`" e executaremos a aplicação a partir deste local. Para isso, execute os seguintes comandos:

```
mkdir ../results
cd ../results
```

A aplicação CMP, a ser utilizada em nossos experimentos, requer um conjunto de parâmetros de entrada para configurar o processamento. Estes parâmetros informam propriedades do dado de entrada e definem o espaço de busca do problema a ser resolvido pela ferramenta. Caso seja de interesse, o artigo de Gimenes e outros [Gimenes et al. 2018] apresenta mais detalhes sobre o funcionamento e a implementação desta ferramenta. Em nossos experimentos utilizaremos o dado `simple-synthetic.su` com os parâmetros listados na Tabela 3.1.

Tabela 3.1: Parâmetros de execução da aplicação CMP com o dado `simple-synthetic.su`.

Parâmetro	Valor
`-aph`	600
`-c0`	1.98e-7
`-c1`	1.77e-6
`-d`	1
`-i`	simple-synthetic.su
`-nc`	300
`-tau`	0.002
`-v`	3

Para executar a ferramenta CMP com os parâmetros listados na Tabela 3.1 execute o seguinte comando:

```
time ../bin/cmp-ocl2 -aph 600 -c0 1.98e-7 -c1 1.77e-6 -d 1 \
      -i ~/simple-synthetic.su -nc 15 -tau 0.002 -v 3
```

Note que o caractere '\' é um caractere especial para escapar a quebra de linha e não precisa ser digitado se o comando estiver em apenas uma linha.

O comando `time` mede o tempo de execução da ferramenta e o reporta no terminal como indicado abaixo:

```
real    0m1.753s
user    0m0.928s
sys     0m0.524s
```

Além do tempo reportado pela ferramenta `time`, você observará que o program CMP reporta outras métricas de desempenho a respeito do principal ponto da aplicação, como indicado abaixo:

```
[INFO]: ~/IPDPS-CRS-CMP-code/CMP/OpenCL/src/main.cpp:281:
Total Execution Time: 0.981616:
Giga-Semblances-Trace/s: 7.646577:
Kernel Execution Time: 0.912309:
Kernel Giga-Semblances-Trace/s: 8.227475
```

Podemos comparar essa implementação com a sua versão OpenMP executada na CPU da máquina. O processo de compilação e execução é similar, a diferença é que a execução é feita sem o uso do parâmetro '`-d`', que determina o dispositivo OpenCL.

```
cd ~/IPDPS-CRS-CMP-code/CMP/OpenMP/
mkdir bin
cd bin
cmake ../
make -j
mkdir ../results
cd ../results
time ../bin/cmp-omp2 -aph 600 -c0 1.98e-7 -c1 1.77e-6 \
      -i ~/simple-synthetic.su -nc 15 -tau 0.002 -v 3
```

3.3.3.3. Desligando a máquina virtual da aplicação CMP

Após realizar este experimento e antes de ir para o próximo estudo de caso, desligue a máquina virtual, conforme apresentado na subseção 3.3.2.7.

3.3.4. Estudo de caso 2: Executando o NPB em múltiplas VMs com MPI

Neste estudo de caso, será descrito o processo de criação de uma máquina virtual, sua configuração para execução de um *benchmark* paralelo, a generalização desta imagem e a execução deste tal *benchmark* em um *cluster* na Nuvem Computacional composto por múltiplas VMs. O *benchmark* é o *NAS Parallel Benchmarks* (NPB) [Bailey et al. 1991], composto por um conjunto de programas paralelos. Nosso foco será a utilização de sua versão MPI.

Neste estudo de caso, teremos dois nós: um nó Mestre/Trabalhador e o outro nó somente como Trabalhador. A Figura 3.20 apresenta o diagrama do experimento na AWS e será detalhado nas próximas subseções.

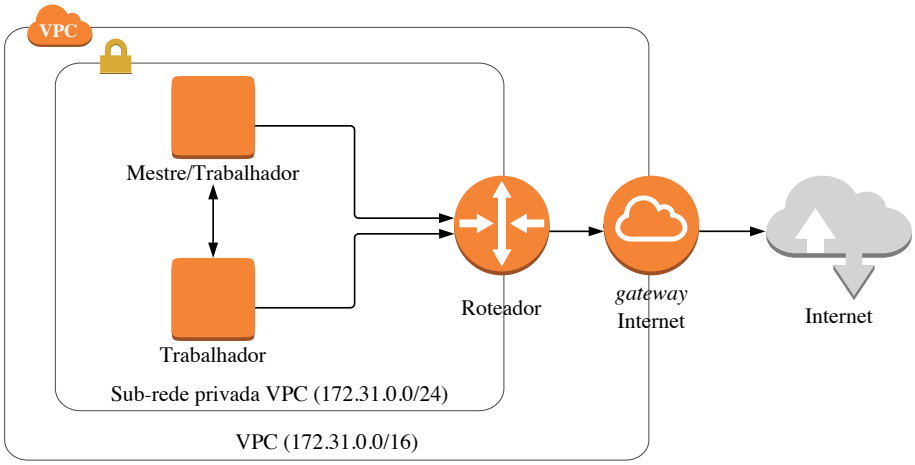


Figura 3.20: Diagrama do experimento na AWS.

3.3.4.1. Configurando uma máquina para geração de imagem

Para a etapa de configuração, uma máquina simples é suficiente. Para este experimento utilizaremos a imagem intitulada "**Ubuntu Server 16.04 LTS**" e o tipo de máquina virtual "t2.micro".

Na próxima etapa, em "*Advanced Details*", na seção "*user data*" pode ser configurado um *script* a ser executado em sua primeira inicialização, onde pacotes necessários podem ser instalados ou atualizados. Selecione a opção "*As text*" e cole as seguintes linhas de comando:

```
#!/bin/bash
sudo apt-get update
sudo apt-get install -y wget make gcc libgfortran3 \
   sysstat libibnetdisc-dev openmpi-bin libopenmpi-dev \
   libhdf5-openmpi-dev gfortran build-essential git
```

Nenhuma outra configuração especial é necessária nesta esta etapa. Dessa forma, avance para a próxima etapa clicando no botão "*Next: Add Storage*" e então avance novamente clicando em "*Next: Add Tags*". Defina os rótulos assim como anteriormente e conclua a inicialização da máquina. Como uma chave de acesso já foi criada na Seção 3.3.2.5, ao lançar a máquina, você pode reutilizar esta mesma chave.

3.3.4.2. Configurando a chave RSA

Após a criação da máquina o acesso pode ser feito por SSH, conforme descrito nas opções do botão conectar. A fim de permitir o acesso por SSH livre entre as máquinas, pode-se criar uma chave de RSA e compartilhá-la entre as demais máquinas. Para tanto, podemos criar a chave utilizando o seguinte comando na máquina virtual criada.

```
ssh-keygen
```

Você não precisa preencher nenhum dos campos solicitados, basta prosseguir pressionando a tecla *Enter*.

Como esta será uma imagem base para todas as máquinas virtuais podemos liberar o acesso dela para ela mesma utilizando o comando:

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

3.3.4.3. Transferindo, compilando e executando o NPB

Para testes de aplicações MPI utilizaremos o *benchmark* paralelo NAS. A primeira etapa é a transferência e a descompressão do *benchmark* na máquina virtual. Estas são feitas utilizando as seguintes linhas de comando:

```
wget https://www.nas.nasa.gov/assets/npb/NPB3.3.1.tar.gz
tar -zxf NPB3.3.1.tar.gz
cd NPB3.3.1/NPB3.3-MPI/
```

Os *benchmarks* serão compilados utilizando um arquivo de descrição da suíte localizado no diretório "*config*". Uma cópia do modelo pode ser feita usando o comando:

```
cp config/make.def.template config/make.def
```

As duas únicas alterações que serão feitas neste arquivo são: a) a troca do compilador Fortran para o mpif90 e b) a troca do compilador C para o mpicc. Estas alterações podem ser realizadas com os seguintes comandos:

```
cat config/make.def.template | \
sed "s/MPIF77 = f77/MPIF77 = mpif90/g" | \
sed "s/MPICC = cc/MPICC = mpicc/g" > config/make.def
```

A suíte de aplicativos a ser compilada também é definida em um arquivo de configuração no mesmo diretório. Uma cópia do *template* pode ser feita usando o comando:

```
cp config/suite.def.template config/suite.def
```

Cada linha deste arquivo descreve uma das opções de compilação, combinando tamanhos de execução (A, B, C e D), com os *benchmarks* e a quantidade de processos que serão lançados durante a execução. Altere o arquivo copiado para gerar executáveis para múltiplos tamanhos de entrada de todas as aplicações com 4 processos. Um editor de texto pode ser utilizado (p.ex. `vim` ou `nano`) para editar o arquivo. Alternativamente, você pode executar o *script* seguinte para realizar esta edição:

```
np=4
for bench in bt cg ep ft is lu mg sp; do
    for size in A B C; do
        echo "$bench    $size    $np" >> config/suite.def
    done
done
```

A compilação pode ser feita utilizando-se o comando:

```
make suite
```

Para um teste simples de execução, execute o comando:

```
mpirun -n 1 bin/cg.S.1
```

Um relatório da execução deve ser exibido no terminal, incluindo informações como tempo de execução e checagem de erros.

3.3.4.4. Criando a imagem base

Uma vez com tudo configurado, está máquina será utilizada como base para a criação de futuras máquinas virtuais. Para isso, selecione a máquina virtual e escolha a opção "*Actions*", "*Image*", "*Create Image*", conforme a Figura 3.21.

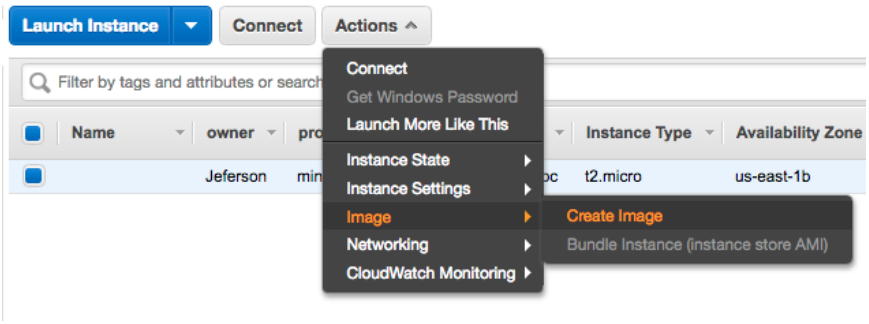


Figura 3.21: Criação de AMI.

Na janela de criação, preencha o campo "*Image name*" com o nome da imagem desejado e o campo "*Image Description*" com uma pequena descrição de sua AMI, conforme apresentado na Figura 3.22. Logo após esta etapa, a instância que estava em uso já pode ser desalocada (*Teminate*).

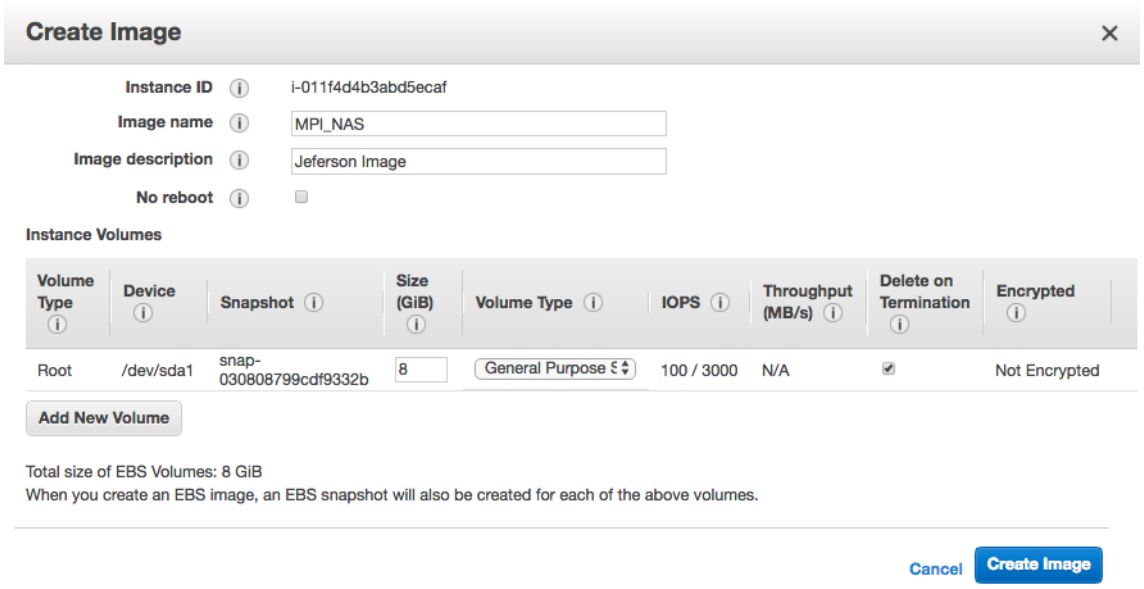


Figura 3.22: Detalhes da criação da AMI.

3.3.4.5. Configurando a rede de interconexão

Para a distribuição da aplicação MPI, todas as máquinas virtuais da rede devem ter livre acesso entre elas. Tal liberação é feita através dos *Security Groups*. A configuração pode

ser feita na parte de "*Security Groups*" do *Dashboard* EC2. Então criamos um novo grupo clicando no botão "*Create Security Group*".

Na criação de um grupo, solicite a liberação apenas da porta 22 a partir de qualquer lugar, conforme ilustrado na Figura 3.23. Se atente à VPC que usará para criar o *Create Security Group*, a mesma VPC deve ser selecionada no momento da criação da máquina.

Figura 3.23: Criação de um *Security Group*.

Com o "*Security Group*" criado, selecione o grupo, no botão "*Actions*" e clique m "*Edite inbound rules*", como mostrado na Figura 3.24.

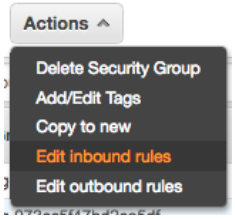


Figura 3.24: Menu de edição das regras do *Security Group*.

Agora você deve criar uma regra para liberar todas as portas entre as máquinas que pertencentes a aquele "*Security Group*". Selecione "*Add Rule*", preencha os campos conforme ilustrado na Figura 3.25. Enquanto você estiver digitando o nome do seu "*Security Group*", aparecerá uma caixa de texto com o identificador, apenas clique na caixa e o campo se preencherá. Em seguida, salve as alterações clicando no botão "*Save*"

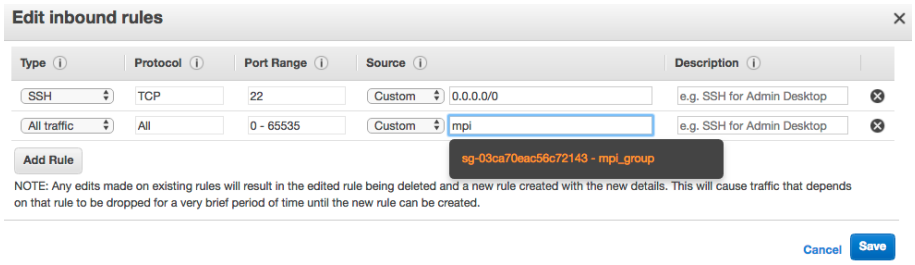


Figura 3.25: Edição de regras do *Security Group*.

3.3.4.6. Selecionando e configurando múltiplas instâncias

Agora, criaremos duas VMs do tipo "M5.large" para executar o *benchmark* NAS de forma paralela. Inicie o processo de criação de uma Máquina Virtual e, para selecionar uma AMI criada por você, escolha no menu lateral a opção "*My AMIs*", conforme a Figura 3.26. Para a criação de um *cluster*(*Cluster Placement Group*), utilizaremos máquinas virtuais otimizadas para Computação, do tipo C5. Os outros tipos de máquinas virtuais que podem ser utilizadas para o *cluster* são:

- Propósito Geral: M4, M5, M5d;
- Otimizada para Computação: C3, C4, C5, C5d, cc2.8xlarge;
- Otimizada para Memória: cr1.8xlarge, R3, R4, R5, R5d, X1, X1e, z1d;
- Otimizada para Armazenamento:: D2, H1, hs1.8xlarge, I2, I3, i3.metal;
- Accelerated computing: F1, G2, G3, P2, P3.

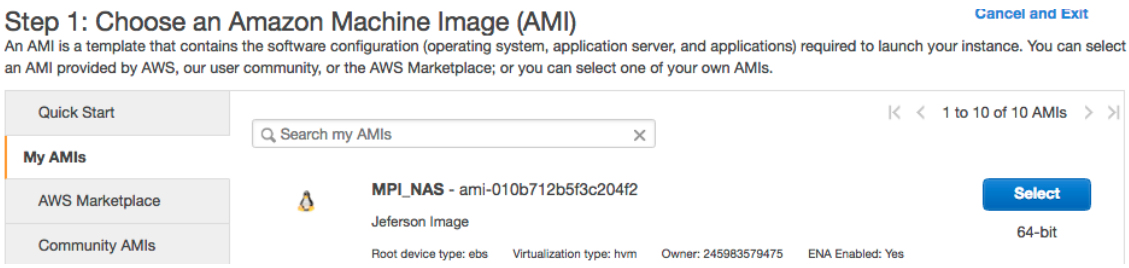


Figura 3.26: Lista de AMIs customizadas.

Na tela de configuração, defina o campo "*Number of instances*" para 2, assim podemos criar as duas máquinas com a mesma configuração de forma simultânea. Marque a caixa "*Add instance to placement group*", para que as máquinas sejam inicializadas no mesmo "*Placement Group*" e possam tirar proveito de uma rede com desempenho melhor. Um grupo pode ser criado selecionando-se a opção "*Add a new placement group*" com o nome definido na caixa de texto abaixo. Assim como apresentado na Figura 3.27.

Ainda nesta etapa, você deve selecionar a mesma VPC de seu "*Security Group*". Então, avance até a etapa de seleção do "*Security Group*" e marque a opção "*Select an existing security group*". Em seguida, selecione o grupo que você criou na Seção 3.3.4.5, como indicado na Figura 3.27.

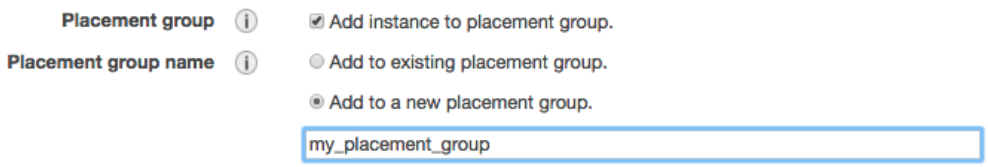


Figura 3.27: Seleção de *Placement Group*.

Após esta etapa, você já pode finalizar a criação das máquinas virtuais, utilizando a sua chave de acesso criada na Seção 3.3.2.5.

Faça o acesso remoto em uma das máquinas, assim como na Seção 3.3.2.6. Como as devidas permissões já foram dadas a chave, basta apenas se conectar a máquina da mesma forma, apenas alterando o endereço. Note que você deve usar o mesmo usuário anterior, `ubuntu`.

Para executar a aplicação nas múltiplas máquinas, primeiro você deve coletar os endereços de IP privados das máquinas. Faça isso selecionando a máquina no *Dashboard* EC2 e no menu inferior (na aba "*Description*" o campo "*Private IPs*" exibe endereços da rede privada). Agora, execute o *benchmark* substituindo os endereços coletados nos campos <IPx>. Note que cada uma das duas máquinas aparece duas vezes na lista, isso ocorre pois cada nó de processamento tem dois núcleos:

```
mpirun -np 4 --host <IP1>,<IP1>,<IP2>,<IP2> \
NPB3.3.1/NPB3.3-MPI/bin/cg.A.4
```

Para não inserir os endereços de IP diretamente na linha de comando, você poderia criar um arquivo para armazena-los, indicando a quantidade de núcleos de processamento de cada nó, através do comando:

```
echo "<IP1> slots=2" > hostfile
echo "<IP2> slots=2" >> hostfile
```

Assim, é possível executar a aplicação, indicando o arquivo de endereços recém criado, usando-se o comando:

```
mpirun -np 4 --hostfile hostfile NPB3.3.1/NPB3.3-MPI/bin/cg.A.4
```

3.3.4.7. Criando o sistema de arquivo compartilhado com o EFS

Quando trabalhamos com sistemas distribuídos, muitas vezes precisamos compartilhar arquivos entre os nós de processamento. O *Elastic File System* (EFS) é um serviço de sistema de de arquivos pela rede fornecido pela AWS. Este serviço possui a vantagem da elasticidade, não sendo necessário dimensionar o tamanho total a ser utilizado.

Para criar um EFS, no menu superior esquerdo, selecione "*Services*" e busque pelo serviço EFS, conforme ilustrado na Figura 3.28.

Em seguida, selecione a opção "*Create File System*", conforme mostrado na Figura 3.29

Selecione a VPC em que você criou o grupo. No destino de montagem, modifique o "*Security Group*" para utilizar o que foi criado anteriormente, no nosso caso "*mpi_group*", e então avance clicando no botão "*Next Step*". Você pode deixar o EFS disponível apenas na mesma zona em que suas máquinas estão executando. Continue a

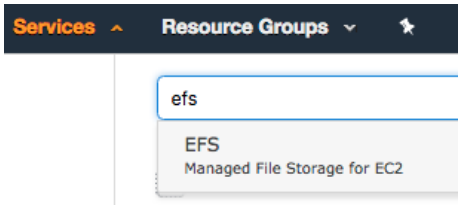


Figura 3.28: Seleção do serviço EFS.

Figura 3.29: Criação de um *Elastic File System*.

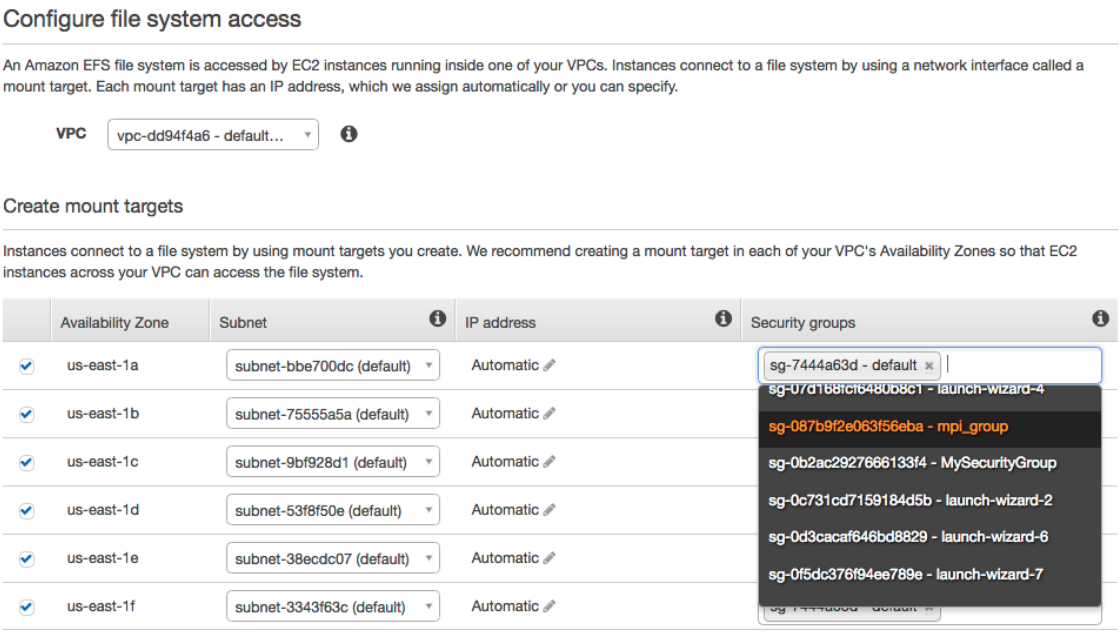


Figura 3.30: Configuração do *Elastic File System*.

criação do sistema de arquivos adicionando etiquetas como na Seção 3.3.2.4 e avance até concluir a criação do EFS.

Depois de analisar e criar o EFS, certifique-se de que o estado do ciclo de vida do destino de montagem esteja disponível, assim como na Figura 3.31.

Após a verificação, siga as Instruções de montagem no Amazon EC2 clicando em "*Amazon EC2 mount instructions*", conforme apresentado na Figura 3.32.

Os comandos utilizados para instalação dos pacotes e montagem foram:

VPC	Availability Zone	Subnet	IP address	Mount target ID	Network interface ID	Security groups	Life cycle state
	us-east-1a	subnet-bbe700dc (default)	172.31.9.104	fsmt-53ae9f1b	eni-0bbf0068fbdc1b2f8	sg-087b9f2e063f56eba - mpi_group	Available

Figura 3.31: Ciclo de vida do EFS.

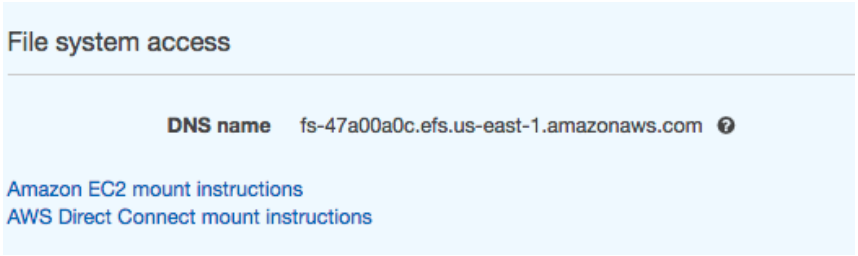


Figura 3.32: Instruções de montagem do EFS.

```
sudo apt-get install nfs-common -y
sudo mkdir ~/efs
sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,\
wsize=1048576,hard,timeo=600,retrans=2,noresvport\
fs-aebf63d7.efs.us-east-2.amazonaws.com:/ ~/efs
```

Agora você pode montar este mesmo EFS em suas duas máquinas e observar que que o sistema compartilhado funcionando. Isso pode ser feito criando-se uma conexão para uma máquina e executando-se o comando

```
sudo sh -c "echo 'Ola mundo' > ~/efs/meu_arquivo.txt"
```

Agora, conecte na segunda máquina e, depois de montar o EFS criado, veja o conteúdo do seu arquivo com o seguinte comando:

```
cat ~/efs/meu_arquivo.txt
```

3.3.4.8. Desligando as máquinas virtuais com MPI

Após realizar este experimento, desligue as máquinas virtuais, conforme apresentado na subseção 3.3.2.7.

3.4. Considerações finais

Este minicurso apresentou uma introdução aos conceitos de computação de alto desempenho, uma visão geral dos serviços oferecidos na Nuvem Computacional e como estes serviços podem ser utilizados para realizar computação de alto desempenho. Além disso, o minicurso apresentou dois casos de uso no provedor de serviços Amazon Web Services (AWS): o primeiro mostrando a execução de uma aplicação de alto desempenho em uma máquina virtual com GPU e o segundo executando uma aplicação paralela com MPI em um *cluster* formado por um grupo de máquinas virtuais.

Referências

[Mag 2011] (2011). The Magellan Report on Cloud Computing for Science. Technical report, U.S. Department of Energy Office of Science Office of Advanced Scientific Computing Research (ASCR).

[AWS 2018] AWS, A. W. S. (2018). Enhanced networking on linux. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/enhanced-networking.html. Acessado em 29 agosto de 2018.

[Azure 2018] Azure, M. (2018). Azure - cloud services pricing. https://azure.microsoft.com/en-us/pricing/details/cloud-services/. Acessado em 29 agosto de 2018.

[Bailey et al. 1991] Bailey, D. H., Barszcz, E., Barton, J. T., Browning, D. S., Carter, R. L., Dagum, L., Fatoohi, R. A., Frederickson, P. O., Lasinski, T. A., Schreiber, R. S., et al. (1991). The nas parallel benchmarks. *The International Journal of Supercomputing Applications*, 5(3):63–73.

[Barr 2016] Barr, J. (2016). Elastic network adapter – high performance network interface for amazon ec2. https://aws.amazon.com/pt/blogs/aws/elastic-network-adapter-high-performance-network-interface-for-amazon-ec2/. Acessado em 29 agosto de 2018.

[Du et al. 2012] Du, P., Weber, R., Luszczek, P., Tomov, S., Peterson, G., and Dongarra, J. (2012). From cuda to opencl: Towards a performance-portable solution for multi-platform gpu programming. *Parallel Computing*, 38(8):391–407.

[Gimenes et al. 2018] Gimenes, T. L., Pisani, F., and Borin, E. (2018). Evaluating the performance and cost of accelerating seismic processing with cuda, opencl, openacc, and openmp. In *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 399–408. IEEE.

[Gupta et al. 2013] Gupta, A., Kale, L. V., Gioachin, F., March, V., Suen, C. H., Lee, B.-S., Faraboschi, P., Kaufmann, R., and Milojicic, D. (2013). The who, what, why, and how of high performance computing in the cloud. In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, volume 1, pages 306–314. IEEE.

[He et al. 2010] He, Q., Zhou, S., Kobler, B., Duffy, D., and McGlynn, T. (2010). Case study for running hpc applications in public clouds. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 395–401. ACM.

[Karmarkar 2015] Karmarkar, T. (2015). Availability of linux rdma on microsoft azure. https://azure.microsoft.com/en-us/blog/azure-linux-rdma-hpc-available/. Acessado em 29 agosto de 2018.

[Luebke et al. 2004] Luebke, D., Harris, M., Krüger, J., Purcell, T., Govindaraju, N., Buck, I., Woolley, C., and Lefohn, A. (2004). Gpgpu: General purpose computation

on graphics hardware. In *ACM SIGGRAPH 2004 Course Notes*, SIGGRAPH '04, New York, NY, USA. ACM.

[Mell and Grance 2011] Mell, P. and Grance, T. (2011). The NIST definition of cloud computing. Technical report, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg.

[Owens et al. 2007] Owens, J. D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A. E., and Purcell, T. J. (2007). A survey of general-purpose computation on graphics hardware. 26(1):80–113.

[Puthal et al. 2015] Puthal, D., Sahoo, B., Mishra, S., and Swain, S. (2015). Cloud Computing Features, Issues, and Challenges: A Big Picture. In *Computational Intelligence and Networks (CINE), 2015 International Conference on*, pages 116–123.

[Valiant 1990] Valiant, L. G. (1990). A bridging model for parallel computation. *Communications of the ACM*, 33.