

NÃO PODE FALTAR

▲
Imprimir

CONCEITOS DE SISTEMAS DISTRIBUÍDOS

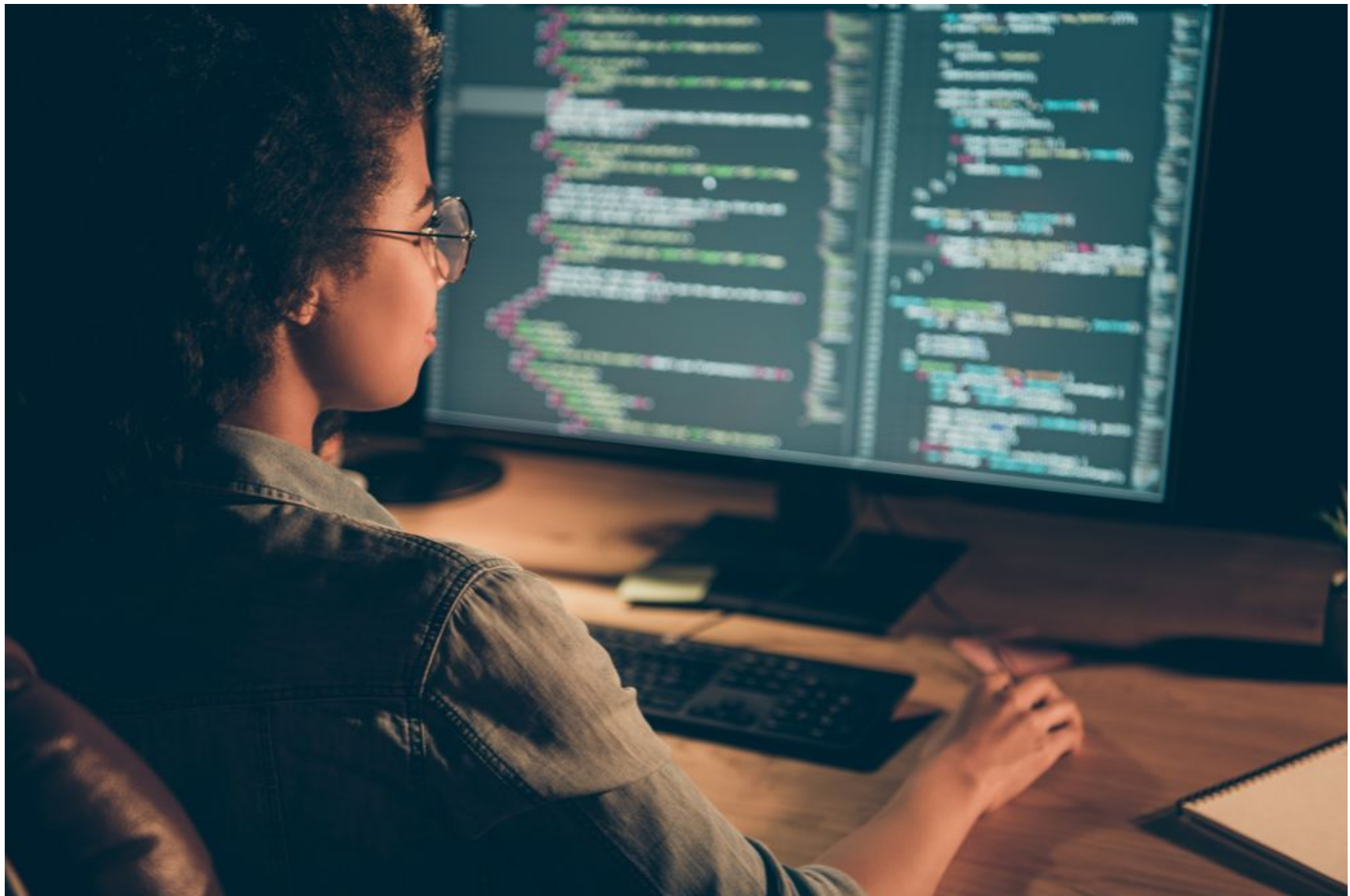
Caique Silva Pereira

0

Ver anotações

O QUE É UM SISTEMA DISTRIBUÍDO?

Um sistema distribuído é um conjunto de computadores que são interligados via rede, porém, para o usuário final das aplicações que são executadas através deles, aparenta ser um sistema único, como uma única máquina ou um único software (TANENBAUM; STEEN, 2017).



Fonte: Shutterstock.

Deseja ouvir este material?

Áudio disponível no material digital.

CONVITE AO ESTUDO

Caro aluno, você sabia que entender os conceitos de sistemas distribuídos é de extrema importância atualmente? Grande parte das aplicações mais populares utilizam esse modelo, em que várias máquinas são interligadas por meio de redes de computadores. Você já parou para pensar como os sistemas computacionais evoluíram até os dias atuais?

Nesta unidade, vamos compreender essa evolução, passando por todas as suas etapas até chegar nos sistemas distribuídos. Você sabe classificar os diferentes sistemas computacionais e distinguir suas características? Após completar esta Unidade, você será capaz de realizar essa tarefa, utilizando seu raciocínio crítico e criatividade para resolução dos problemas propostos. É fato que a demanda por profissionais com conhecimento em implantação de sistemas distribuídos tem aumentado principalmente em consultorias de TI que prestam serviços especializados para seus clientes nos mais diversos segmentos.

Ver anotações

Iniciaremos agora os estudos sobre os sistemas distribuídos, o que lhe proporcionará um leque de oportunidades no mercado de trabalho, assim como conhecimentos importantes em relação a tecnologias atuais. Importante frisar que quanto mais você se dedicar, mais poderá aproveitar os ensinamentos transmitidos neste material.

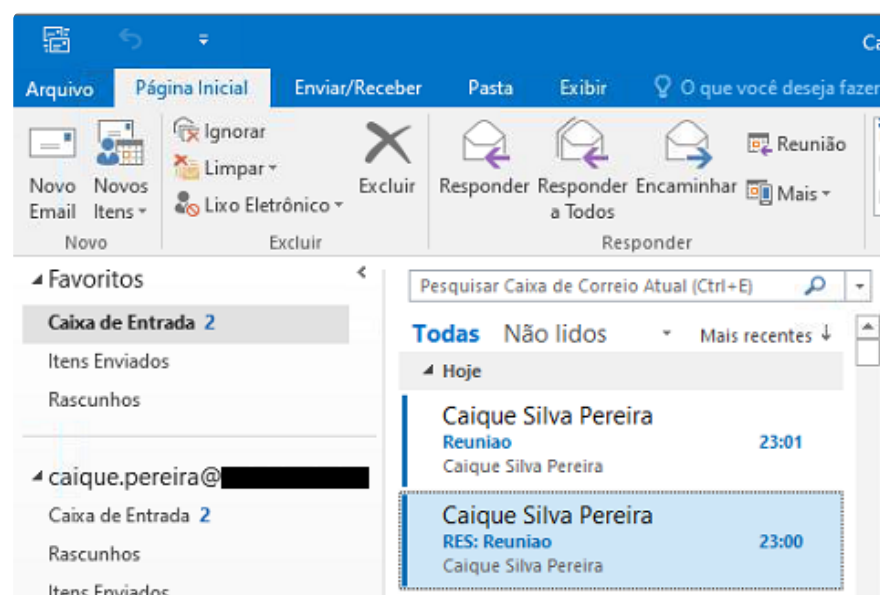
PRATICAR PARA APRENDER

Nesta seção, trabalharemos conceitos e definições de sistemas distribuídos e veremos as diferenças entre *clusters* e *grids*, além de exemplos de sistemas distribuídos e de sincronização de relógios no sistema operacional Windows.

Você sabia que, para o funcionamento correto de uma aplicação distribuída, os relógios de todas as máquinas devem estar sincronizados? Além disso, já ouviu falar em um servidor NTP, o qual serve como base para sincronizar os relógios das máquinas?

Caro aluno, você foi contratado por uma *startup* de desenvolvimento de sistemas bem-sucedida nos negócios, a qual iniciou projetos de sistemas para grandes empresas. Em seu primeiro dia, você recebeu um e-mail para uma reunião on-line de boas-vindas, mas percebeu que, ao responder e confirmar a reunião, ocorreu a indicação de horário equivocada, ou seja, o horário de recebimento do e-mail original apareceu como posterior ao horário do e-mail de resposta, conforme mostrado na Figura 3.1 a seguir.

Figura 3.1 | Exemplo de problema em uma caixa de e-mails



Fonte: captura de tela do Microsoft Outlook.

0
Ver anotações

A Figura 3.1 ilustra um problema de sincronização de relógios nos sistemas computacionais que se comunicam em rede.

Sua primeira missão como analista na *startup* é reportar um erro de sincronização de aplicações que rodam em diferentes máquinas de um cliente de lojas de varejo. Assim que começou sua análise, você percebeu que os relógios das máquinas que rodam a aplicação não estão sincronizados, então, para resolver isso, você deverá preparar um relatório mencionando o problema e como resolver esse ajuste de horas, para que um técnico de campo seja enviado até o local.

Após seus relatórios e apresentações entregues ao dono das lojas, este percebe que você sabe do que está falando e confessa que não entende como os computadores podem conversar entre si. Entre outras coisas, você fala da importância do sincronismo entre as máquinas e decide mostrar que sistemas em rede não são tão complicados assim.

Você já parou para pensar quais são as configurações e os comandos necessários para que essa sincronização funcione adequadamente? Qualquer computador pode ser usado para realizar a configuração ou é necessário um servidor específico para essa tarefa? Se for necessário um servidor, qual é o mais utilizado? Para esclarecer essas dúvidas, você decide exemplificar o funcionamento de um sistema em rede de maneira simples e prática, acessando remotamente e sincronizando o relógio local dos computadores das três lojas com a internet, por meio do seu notebook, fazendo, assim, um acesso remoto, utilizando comandos específicos, relacionados aos servidores NTP. Para que o proprietário consiga fazer o

procedimento quando a sua loja adquirir uma nova máquina, crie um relatório técnico com os comandos necessários, passo a passo, para que ele mesmo sincronize a nova máquina com as demais.

Para completar esse desafio, nesta seção, você verá em detalhes o funcionamento desse processo de sincronização de relógios, incluindo comandos específicos a serem utilizados tanto para a configuração como para a constatação de que o serviço de sincronização está funcionando de maneira adequada. Ficou curioso? Espero que você se sinta motivado a dedicar seu tempo e seus esforços a um estudo que lhe proporcionará chances reais de assimilar os conceitos e as práticas que você utilizará na sua vida profissional.

0
Ver anotações

CONCEITO-CHAVE

DEFINIÇÃO E EXEMPLOS DE SISTEMAS DISTRIBUÍDOS

Mesmo que não saiba, hoje mesmo você já deve ter acessado um sistema distribuído. Ao abrir o navegador de sua preferência e acessar uma página de internet, você está usando um sistema distribuído. Essa simples ação rotineira em nosso dia a dia, por meio de um smartphone ou computador, utiliza um sistema distribuído. Mas, afinal, o que é um sistema distribuído?

Um sistema distribuído é um conjunto de computadores que são interligados via rede, porém, para o usuário final das aplicações que são executadas através deles, aparenta ser um sistema único, como uma única máquina ou um único software (TANENBAUM; STEEN, 2017).

Um de seus principais aspectos é que os computadores que fazem parte de sistemas distribuídos têm o funcionamento independente, ou seja, cada um age por si próprio e, muitas vezes, os sistemas e hardwares dessas máquinas são totalmente diferentes, mas aparentam ser uma coisa só para o usuário. Esses computadores estão ligados por meio de uma rede, e só assim é possível seu funcionamento de forma distribuída.

ASSIMILE

As principais aplicações e sistemas da atualidade utilizam modelos distribuídos. Esse conceito faz com que o sistema não dependa apenas de uma máquina, pois toda a carga do ambiente estará distribuída entre várias máquinas. Assim, os sistemas distribuídos ocultam o conjunto de máquinas que o fazem funcionar, aparentando ser algo único.

o

Ver anotações

Conforme Tanenbaum e Steen (2017, p. 2), “Os modelos de sistemas distribuídos servem para definir a forma como os diversos objetos distribuídos se comunicam e interagem entre si”. Os sistemas distribuídos seguem tendências e modelos para seu funcionamento. Por exemplo, se pensarmos em variados sistemas operacionais, a ação de abrir uma pasta é representada por dois cliques do mouse. Quando trabalhamos com sistemas distribuídos, temos objetivos claros a serem alcançados em nosso sistema, com sua criação, a saber:

- Disponibilidade alta e fácil acesso ao sistema e a todos os seus recursos, tanto pelas máquinas que fazem parte do sistema distribuído como pelo usuário final.
- Devemos também ocultar ao usuário que os recursos de nosso sistema são distribuídos; essa é uma característica muito importante.
- O sistema distribuído deve ser aberto, ou seja, deve facilitar a inclusão de novas máquinas e recursos no ambiente em funcionamento, sendo assim, esse sistema pode ser expandido facilmente.

REFLITA

Você conheceu alguns conceitos de sistemas distribuídos e percebeu o quanto eles são utilizados em aplicações modernas. Agora que você já os conhece, consegue identificar quais seguem esse modelo dentre os serviços que mais utiliza? Além disso, será que algum serviço que você utiliza pode ser otimizado aplicando esses conceitos? Reflita!

EXEMPLIFICANDO

Imaginaremos que queremos ver a página de notícias esportivas do nosso time do coração em um portal de notícias X, o qual segue o modelo de sistemas distribuídos. Sendo assim, a página acessada está sendo fornecida

por um conjunto de quatro servidores, os quais, para o usuário final, aparentarão ser algo único. Pensando nisso, uma das funções dos sistemas distribuídos é que, mesmo que algum dos quatro servidores que estão mantendo essa página no ar esteja desligado, os outros devem assumir sua função, dividindo a carga, e o conteúdo deve estar no ar, assim como o sistema precisa estar em funcionamento.

Ver anotações

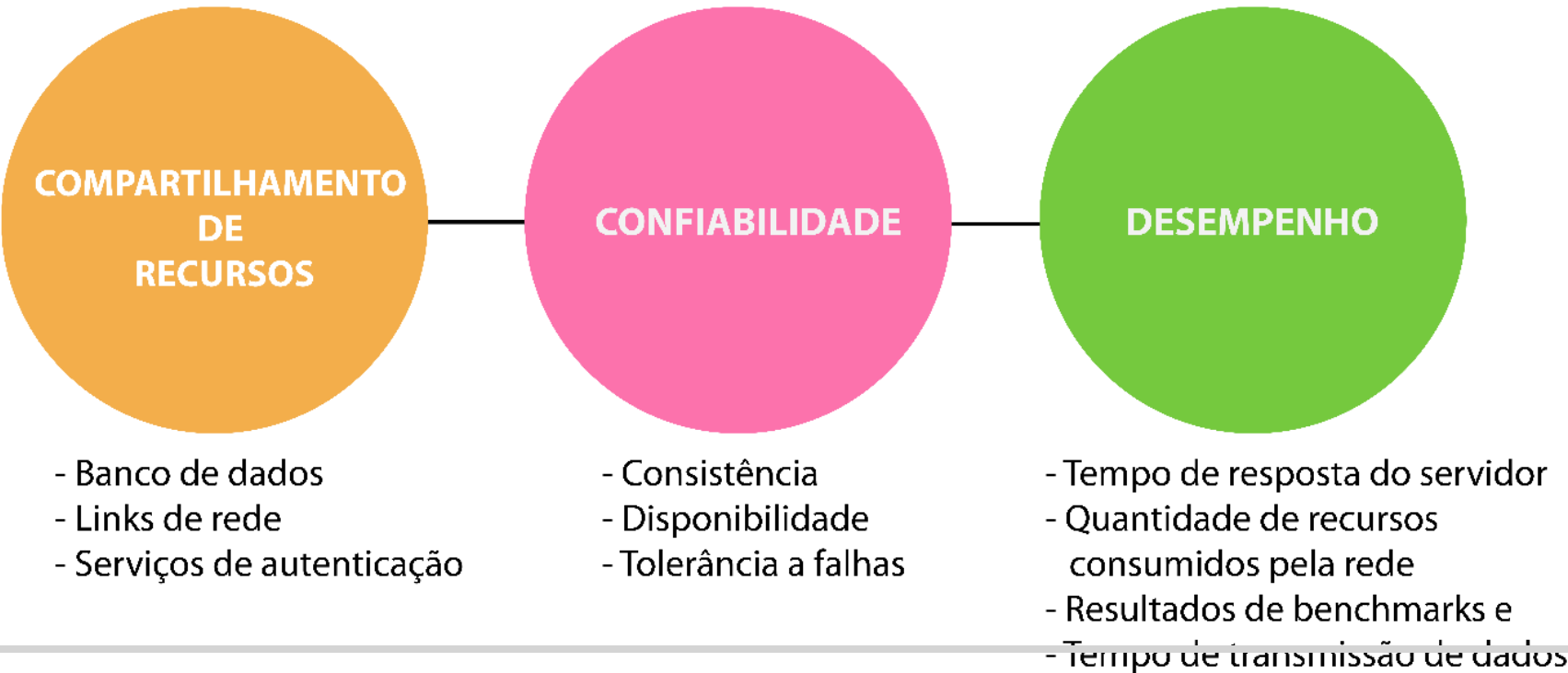
A principal motivação para construir e utilizar sistemas distribuídos é proveniente do desejo de compartilhar recursos. O termo “recurso” é bastante abstrato, mas caracteriza o conjunto de elementos que podem ser compartilhados de maneira útil em um sistema de computadores interligados em rede. Ele abrange desde componentes de hardware, como discos e impressoras, até entidades definidas pelo software, como arquivos, bancos de dados e objetos de dados de todos os tipos, conforme indica *Coulouris et al.* (2013).

ASSIMILE

Um sistema distribuído é formado por vários nós/máquinas que executam uma função dentro do sistema distribuído, de modo que, para o usuário final, aparentam ser uma única máquina.

Segundo Tanenbaum e Steen (2017), os sistemas distribuídos possuem três objetivos principais: compartilhamento de recursos, confiabilidade e desempenho, conforme ilustra a Figura 3.2.

Figura 3.2 | Exemplos dos principais objetivos de um sistema distribuído



Fonte: elaborada pelo autor.

COMPARTILHAMENTO DE RECURSOS

O compartilhamento de recursos refere-se à capacidade do sistema em compartilhar o acesso a quaisquer recursos utilizados por ele entre as máquinas que fazem parte da arquitetura (também chamadas de nós). Esses recursos são, na maioria das vezes, bancos de dados, links de rede que se conectam à internet, serviços de autenticação, entre outros. Apesar de não ser um objetivo exclusivo dos sistemas distribuídos – uma vez que também é um objetivo dos sistemas de rede –, é uma característica muito importante.

A vantagem de compartilhar recursos está na economia financeira, uma vez que, caso não haja tal possibilidade de compartilhamento, mais réplicas de um determinado recurso devem estar presentes em cada nó do sistema, o que impacta direta e indiretamente no custo. Entretanto, como aspecto negativo associado a esse compartilhamento de recursos, temos a questão da segurança, uma vez que o fato de mais máquinas terem acesso ao recurso implica que o sistema possui mais pontos de acesso, os quais podem ser explorados por hackers, tanto no sentido de rastreamento da comunicação quanto na própria questão de invasão de privacidade e integridade dos dados (COULOURIS *et al.*, 2013).

CONFIABILIDADE

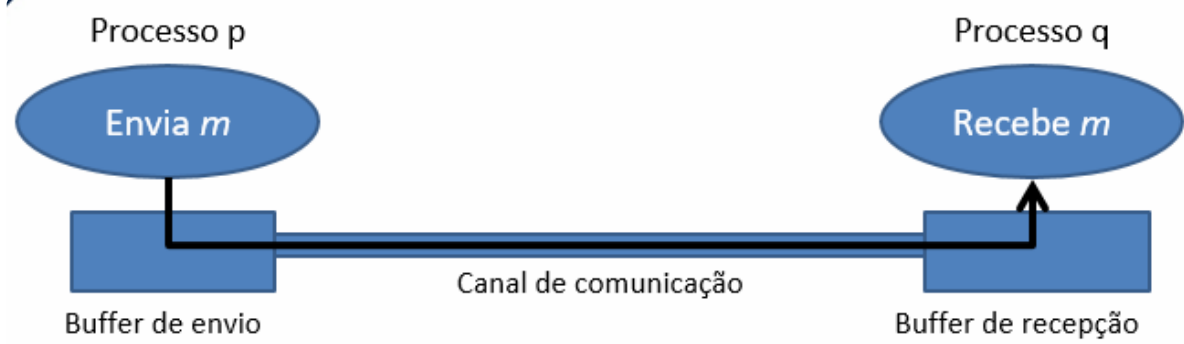
A análise morfológica da palavra confiabilidade nos mostra que ela se refere à probabilidade de um produto executar a sua função prevista de forma que atenda ou exceda às expectativas, ou seja, está relacionada ao funcionamento adequado, conforme foi planejado. Podemos confundir a confiabilidade como algo relacionado à segurança do sistema, porém não tem relação alguma, conforme observam Colouris *et al.* (2013).

A confiabilidade nos sistemas distribuídos é maior que nos sistemas centralizados, no entanto qualquer problema relacionado aos processos ou ao canal de comunicação/transmissão pode surtir efeitos diretos sobre a execução do sistema. Podemos observar, na Figura 3.3, como ocorre a comunicação entre processos nos sistemas distribuídos, nos quais são aplicados os conceitos de confiabilidade:

o

Ver anotações

Figura 3.3 | Comunicação entre processos em um sistema distribuído



Fonte: elaborada pelo autor.

Ver anotações

Podemos observar, na Figura 3.3, que, se ocorrer algum problema no canal de comunicação, isso será refletido em todo o processo de comunicação e funcionamento do sistema. A confiabilidade de um sistema é baseada em três pilares: consistência, disponibilidade e resiliência (tolerância a falhas), conforme o Teorema CAP. Esse teorema é baseado nesses pilares, e sua sigla vem das palavras consistência (*consistency*), disponibilidade (*availability*) e tolerância a falhas (*partition tolerance*) (GREINER, 2014). Podemos observar, na Figura 3.4, a representação desse teorema.

Figura 3.4 | Representação do Teorema CAP



Fonte: adaptada de Greiner (2014, [s.p.]).

Uma das características importantes do Teorema CAP, a qual pode ser observada na representação da Figura 3.4, é que nunca podemos atingir os três pilares em sua totalidade dentro de um sistema. A forma como foi elaborado permite que você tenha apenas dois dos pilares em evidência em seu sistema, ou seja, caso selecione dois pilares em seu sistema, o terceiro ficará enfraquecido.

Figura 3.5 | Disponibilidade e tolerância de partição

Disponibilidade e tolerância de partição



Fonte: elaborada pelo autor.

ASSIMILE

O Teorema CAP consiste em três pilares: consistência, disponibilidade e resiliência (tolerância a falhas). Seu sistema só poderá ter foco em dois deles, e o terceiro será enfraquecido. Por exemplo: se optar por consistência e disponibilidade, seu sistema terá menor resiliência. E assim ocorre em todas as suas combinações e possibilidades.

DESEMPENHO

Aumentar o desempenho de um sistema também é um objetivo dos sistemas distribuídos. Se fizermos uma comparação, os sistemas distribuídos, na maioria dos casos, apresentam melhor desempenho do que os sistemas centralizados. Isto ocorre porque, em um sistema distribuído, temos múltiplas instâncias, tanto de hardware quanto de software, para realizar o processamento necessário. Como podemos medir esse desempenho?

Conforme Coulouris *et al.* (2013, p. 83), podemos utilizar como parâmetros:

- Tempo de resposta do servidor.
- *Throughput* (taxa de transferência).
- Quantidade de recursos consumidos pela rede.
- Resultados de *benchmarks* (execução do sistema).
- Tempo de transmissão dos dados.

Com os resultados das medições dos parâmetros listados, podemos identificar se o sistema distribuído é satisfatório.

Importa destacar que os sistemas distribuídos são mais utilizados em arquiteturas do tipo cliente-servidor, as quais apresentam recursos compartilhados (tanto a nível de hardware quanto a nível de software), a fim de permitir que milhares de clientes tenham acesso a esses recursos e possam utilizá-los como se houvesse uma comunicação direta entre as máquinas e o cliente.

0
Ver anotações

EXEMPLIFICANDO

A maioria dos jogos multijogadores on-line atuais utiliza a arquitetura cliente-servidor para seu funcionamento. Esse tipo de jogo também é um exemplo de sistema distribuído.

Imagine um jogo on-line de guerra em que você é o jogador X e está enfrentando seu amigo, que é o jogador Y. Você, na verdade, tem três avatares dentro de uma sessão de jogo: o primeiro é o avatar que você está controlando e visualizando em seu dispositivo; o segundo é chamado de imagem autoritária, isto é, a cópia de seu avatar feita pelo servidor do jogo, a qual será enviada para que os outros jogadores possam te “enxergar” dentro do jogo; esse segundo avatar, ao ser enviado aos outros jogadores, funciona como o terceiro avatar, que será visualizado, por exemplo, no jogo do seu amigo. Logo, você movimenta o seu avatar e a sua posição é enviada ao servidor, o qual gera uma cópia com seu posicionamento e transfere-a para o jogador Y. Com isso, temos a arquitetura cliente-servidor em funcionamento.

Figura 3.6 | Exemplo de jogo multijogador on-line



Fonte: Shutterstock.

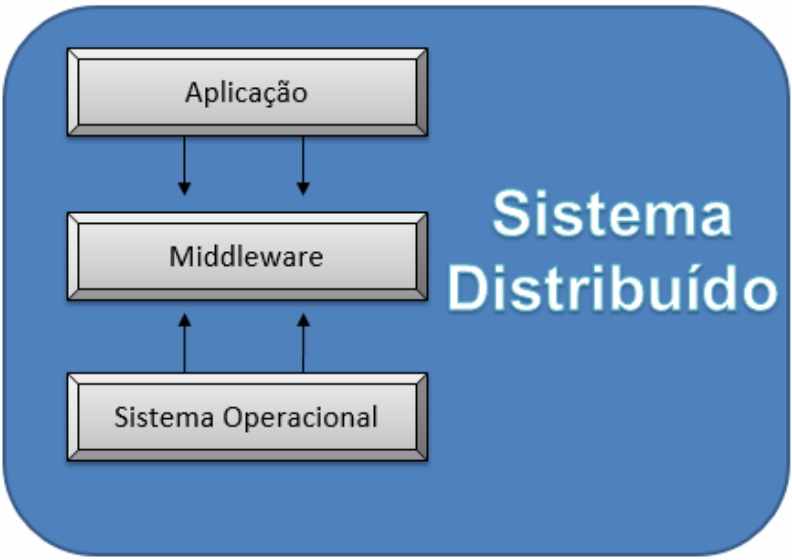
| MIDDLEWARE

As redes estão por toda parte e servem de base para muitos serviços cotidianos que agora consideramos naturais, por exemplo, a internet e a *World Wide Web* associada a ela, a pesquisa na web, os jogos on-line, os e-mails, as redes sociais, o e-commerce, etc. (COULOURIS *et al.*, 2013). Os sistemas distribuídos podem ser considerados como uma solução mais robusta em resposta aos sistemas puramente de rede, isso graças ao componente conhecido como *middleware*. Ele é um dos fatores principais para o bom funcionamento de aplicações distribuídas.

Middleware é uma camada oculta, um software, o qual se encontra entre os sistemas operacionais e os aplicativos (programas criados) que são executados neles. Portanto, é uma camada central, que permite o gerenciamento e a comunicação de dados para o funcionamento de aplicativos distribuídos. O *middleware* funciona como uma camada de tradução para interligar o sistema operacional com os programas (COULOURIS *et al.*, 2013).

A Figura 3.7 ilustra as camadas que compõem um sistema distribuído.

Figura 3.7 | Camadas que compõem um sistema distribuído



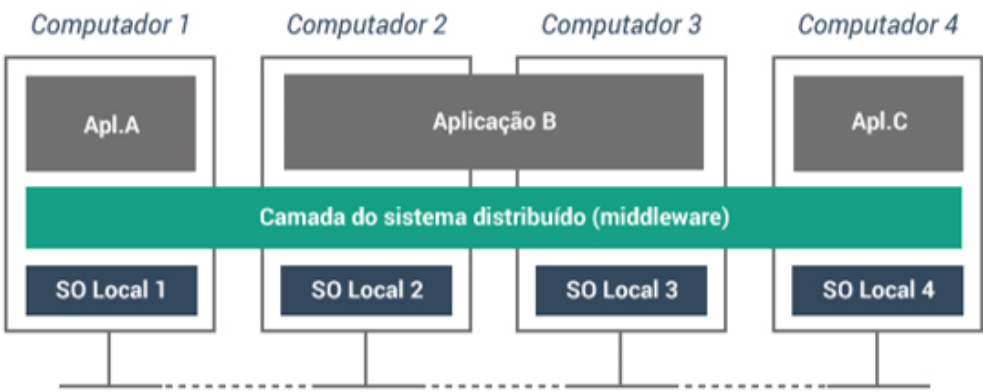
Ver anotações

Fonte: elaborada pelo autor.

Alguns exemplos são: *middlewares* de banco de dados e servidores WEB e ferramentas de mensageria. As mais diversas redes sociais que utilizamos hoje em dia, com conteúdos multimídias, são exemplos de sistemas distribuídos, assim como sites de pesquisas e plataformas de vídeos on line.

A Figura 3.8 ilustra um exemplo da atuação da camada *middleware* perante diferentes computadores (máquinas), aplicações e sistemas operacionais. Podemos observar que ela liga aplicações A, B e C com os sistemas operacionais 1, 2, 3 e 4. Essas aplicações e sistemas operacionais estão em diferentes computadores (1, 2, 3 e 4). Eles interagirão entre si para a execução de um sistema distribuído, e isso só é possível devido à camada do sistema distribuído, chamada *middleware*.

Figura 3.8 | Middleware e sistemas distribuídos



Fonte: elaborada pelo autor.

ASSIMILE

Em uma aplicação escrita em diferentes máquinas, com sistemas operacionais diferentes, é necessária a comunicação entre essas máquinas. Cada uma delas fornece a própria representação de dados e formas de

comunicação e, nesse caso, o *middleware* atua como uma camada de tradução para que seja possível a comunicação correta entre as máquinas para o funcionamento do sistema.

COMPUTAÇÃO EM *CLUSTER* E COMPUTAÇÃO EM *GRID*

Os sistemas distribuídos podem ser classificados em diferentes categorias, de acordo com sua arquitetura e finalidade, sendo os mais comuns: computação em cluster e computação em grid.

CLUSTER

Agora, falaremos um pouco sobre algumas características da computação em *cluster*. Esse tipo de computação é formado por um conjunto de máquinas com hardwares semelhantes, ou seja, as máquinas que compõem o *cluster* possuem características homogêneas (TANENBAUM; STEEN, 2017). O conjunto de máquinas que compõem o *cluster* são ligadas por rede local (LAN).

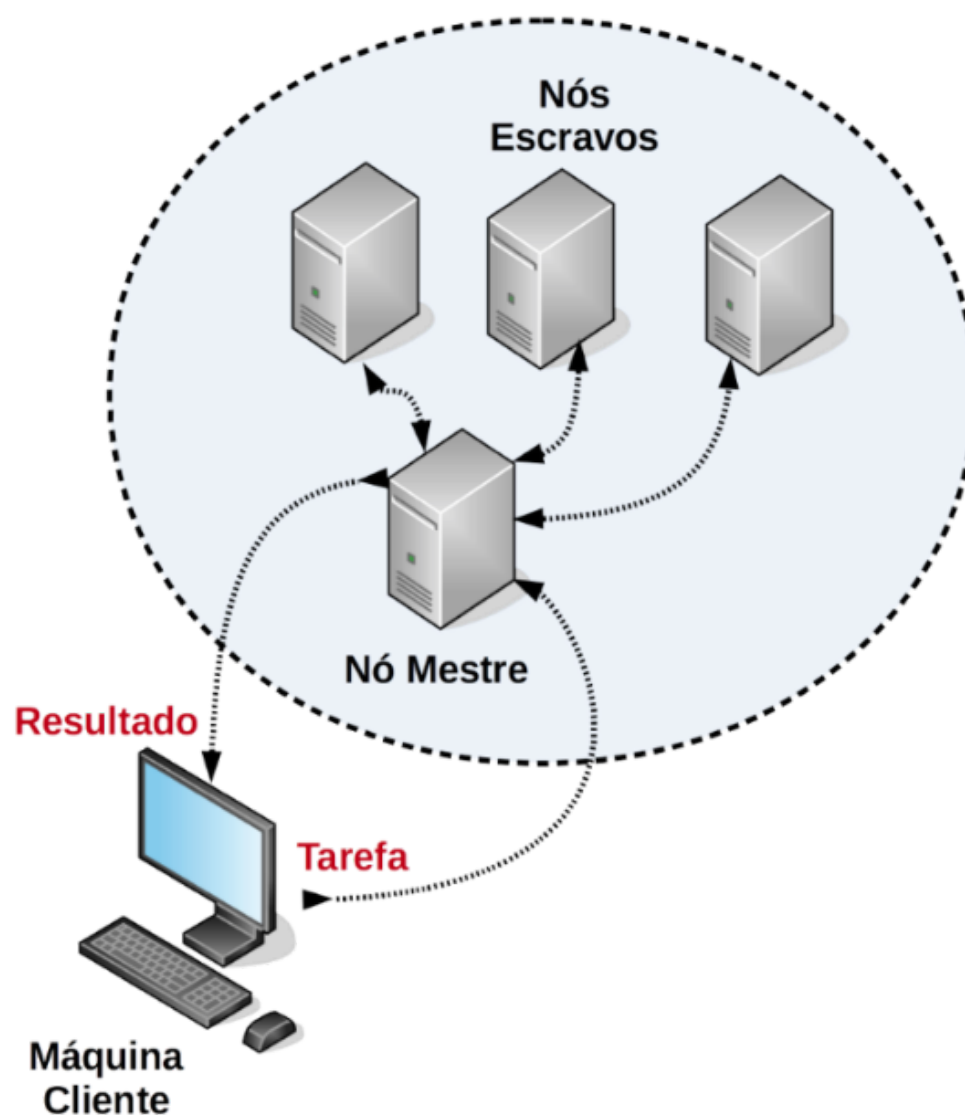
Quando falamos da parte de software da computação em *cluster*, temos algumas características importantes. Na maioria das vezes, o sistema operacional entre as máquinas que formam o *cluster* é equivalente. Além disso, é frequente que um único programa funcione de forma paralela, ou seja, um programa é subdividido em partes menores, e cada parte é executada em uma máquina (ou nó) desse *cluster*, de forma distribuída, a fim de obter um aumento significativo de desempenho e, conseqüentemente, executar determinada tarefa em menos tempo. Geralmente, as máquinas desse tipo de sistema são fortemente acopladas em suas ligações e, muitas vezes, podem até compartilhar a mesma memória RAM entre várias máquinas.

Há sempre uma das máquinas que chamamos de **nó mestre**, isto é, a máquina principal que gerencia o funcionamento da aplicação entre todos os nós. O nó mestre faz a interface com o usuário, aloca tarefas e administra a fila de tarefas. A Figura 3.9 ilustra um *cluster*, que se encontra dentro do círculo pontilhado.

Figura 3.9 | Exemplo de *cluster*

0

Ver anotações



Fonte: elaborada pelo autor.

GRID

Agora que já conhecemos a computação em *cluster*, falaremos da computação em *grid*, ou grades, e apontaremos algumas características. Conforme Tanenbaum e Steen (2008), esse tipo de computação é formado por um conjunto de máquinas com características diferentes, podendo o hardware e os sistemas operacionais serem de fabricantes diferentes. Com isso, temos uma característica heterogênea na computação em grid: essencialmente, um sistema de computação em *grid* interliga vários *clusters*. Um exemplo de *grid* é o CINEGRID, que trabalha no desenvolvimento de ferramentas colaborativas multimídia (CINEGRID BRASIL, [s.d.]). Na Figura 3.10, vemos uma parte desse *grid*, com as ligações entre *clusters* no Brasil; mas saiba que o CINEGRID interliga outros centros de pesquisa, em várias partes do mundo.

Figura 3.10 | Exemplo de grid - CINEGRID



Ver anotações

Fonte: CINEGRID BRASIL ([s.d.], [s.p.]).

DIFERENÇAS ENTRE *CLUSTERS* E *GRIDS*

Muitas vezes, pode parecer que *clusters* e *grids* são a mesma coisa, mas existe uma característica fundamental que difere esses dois tipos de sistemas distribuídos. Para facilitar o entendimento das diferenças entre eles, podemos pensar que *clusters* são sistemas homogêneos, ou seja, são criados para executarem alguma tarefa específica que, em geral, necessita de um alto poder de processamento e, portanto, levaria muito tempo para ser executada em um computador convencional.

EXEMPLIFICANDO

Um exemplo de tarefa que leva muito tempo para ser executada é o treinamento de redes neurais artificiais de aprendizagem profunda (*deep learning*) para uso em ferramentas de chat on-line. Nesse tipo de aplicação, os chamados *bots* – que, nesse caso, são o resultado da rede neural treinada – conversam com e respondem a questionamentos dos clientes (usuários) do serviço de chat on-line de uma determinada empresa. Recentemente, muitos sistemas bancários têm utilizado esse tipo de recurso. Supondo que para executar o treinamento desse tipo de rede em computador convencional seriam necessários dois dias, ao executarmos essa tarefa em um *cluster*, esse treinamento poderia ser realizado em questão de minutos.

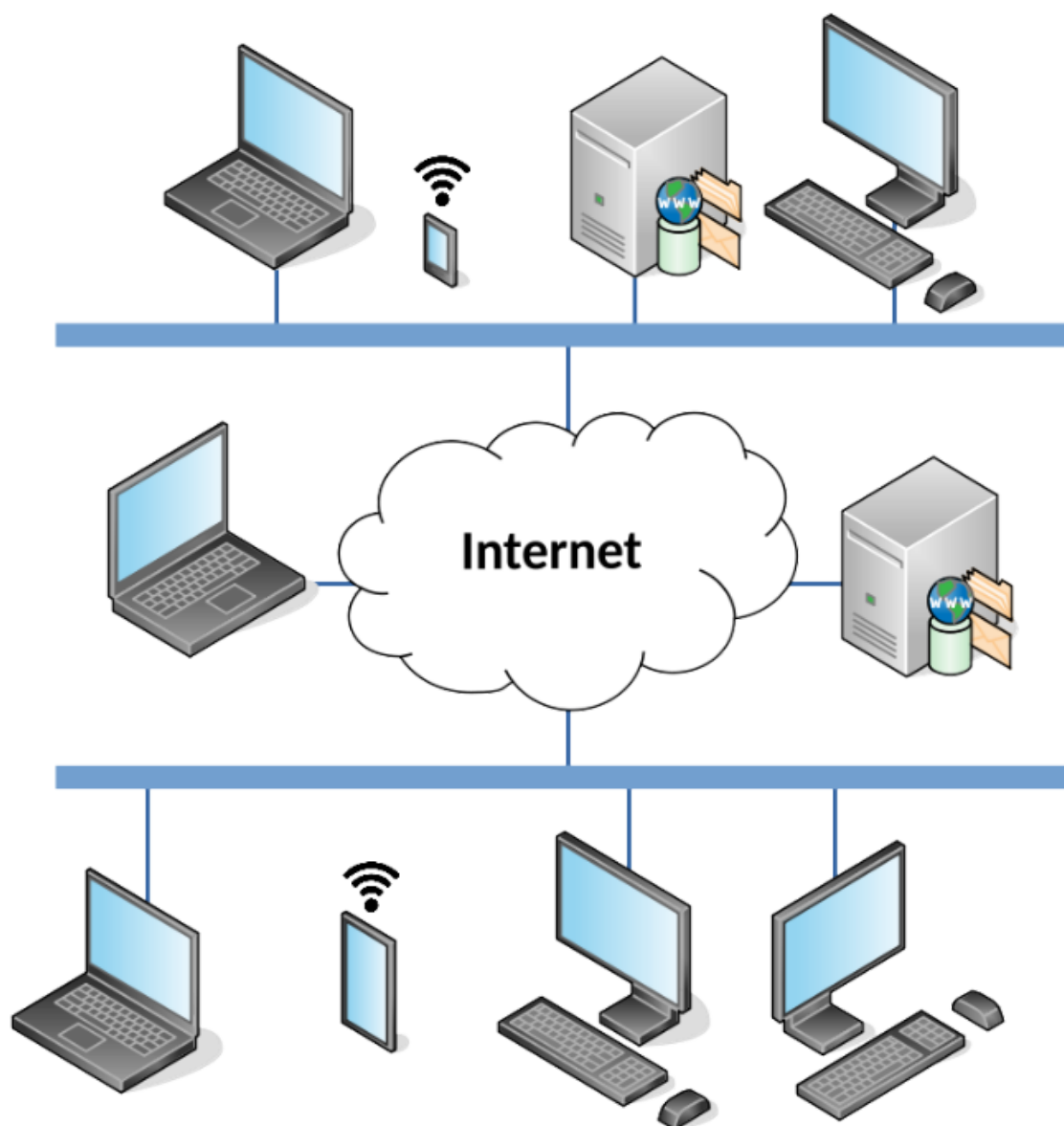
Para diminuirmos o tempo de processamento, poderíamos executar esse tipo de tarefa (treinamento de redes neurais artificiais de aprendizagem profunda) em um computador com alto poder de processamento, porém o custo desse tipo de computador pode tornar essa opção inviável. Segundo o *Lawrence Livermore National Laboratory* (2018), o sequoia é utilizado para realizar simulações numéricas referentes à física de armas nucleares. Assim sendo, este computador é um *cluster* homogêneo, visto que executa uma pesquisa de finalidade específica.

Por sua vez, podemos pensar que os *grids* têm uma abordagem heterogênea, ou seja, são criados para executarem diferentes tarefas, de certa maneira relacionadas entre si, formando um centro de pesquisas de caráter multidisciplinar. Uma maneira ainda mais simples de entender essa característica é “pensar” em um *grid* como um conjunto de dois ou mais *clusters*, cada um deles responsável por um certo tipo de pesquisa.

ARQUITETURA DESCENTRALIZADA

Na arquitetura descentralizada, os computadores são os próprios servidores da aplicação (serviço ou recurso a ser compartilhado), o que se assemelha à arquitetura ponto a ponto. Entretanto, diferentemente do que ocorre na arquitetura ponto a ponto, o estado da aplicação (por exemplo, os valores atuais das variáveis utilizadas na tal aplicação) é replicado entre todos os computadores na rede, de maneira que exista um chamado consenso entre os computadores nessa rede.

Figura 3.11 | Arquitetura descentralizada



Fonte: elaborada pelo autor.

Há, ainda, a possibilidade de que alguns computadores simplesmente utilizem a aplicação (em vez de serem servidores desta), tendo o papel puramente de “clientes” nesse caso, o que remete à arquitetura cliente-servidor.

Essa é a arquitetura utilizada pelas plataformas baseadas em *blockchain*, e tem se tornado mais popular após o advento do bitcoin. Aplicações que funcionam sobre esse tipo de plataforma são chamadas de dApps (do inglês, *decentralized application*). Uma das principais vantagens ao utilizar esse tipo de arquitetura é a de que não há uma entidade que controle sua aplicação, como tipicamente ocorre nas arquiteturas cliente-servidor. Por exemplo, seus e-mails do Gmail são controlados pelo Google, que armazena os dados e poderia, hipoteticamente, acessar dados privados ou decidir encerrar seus serviços – situação na qual você estaria impossibilitado de optar por decisão contrária, caso não seja um acionista da empresa. Na arquitetura descentralizada, há uma garantia de transparência aos usuários de uma determinada aplicação, uma vez que a aplicação e seus dados são armazenados por computadores dos próprios usuários que participam da rede, e não por uma empresa somente.

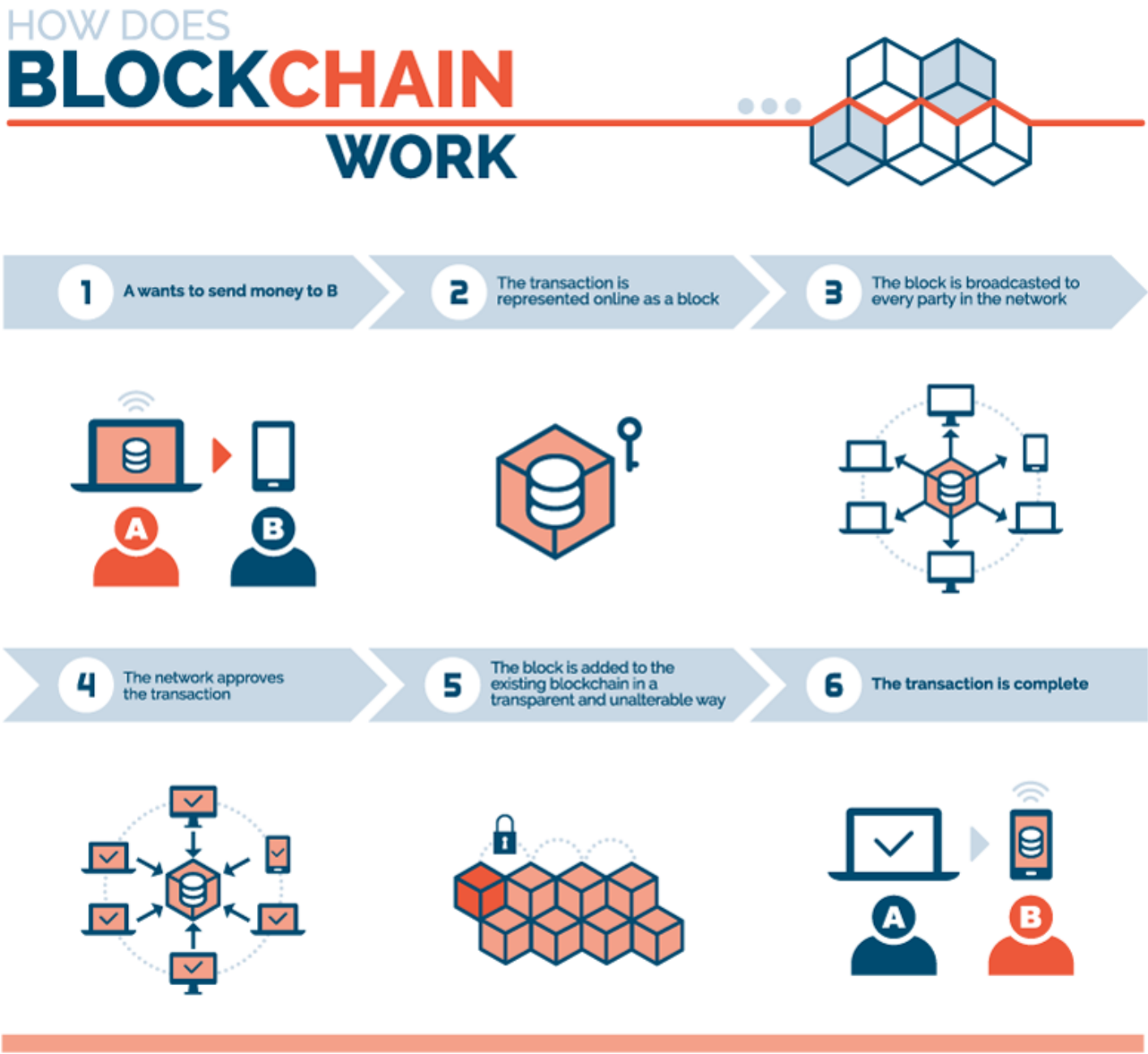
Exemplos desse tipo de arquitetura podem ser vistos em qualquer sistema de transações financeiras das chamadas criptomoedas (como o próprio bitcoin), mas não se limita apenas a esse tipo de aplicação. Um exemplo é a Steemit, aplicação de rede social similar ao Twitter, porém faz uso da arquitetura descentralizada.

o

EXEMPLIFICANDO

Figura 3.12 | Como funciona a BlockChain

Ver anotações



Fonte: Shutterstock.

Conforme podemos observar na Figura 3.12, a *BlockChain* validadora de transações é a principal tecnologia por trás do universo das criptomoedas, que tem o bitcoin como sua principal moeda. De uma maneira simples, a *BlockChain* pode ser considerada um livro compartilhado e distribuído, no

qual todos os usuários dessa rede mantêm uma cópia e todas as transações de moedas ficam registradas nessa rede. Portanto, esses registros são imutáveis e confiáveis.

A *BlockChain* armazena essas informações, esse conjunto de transações, em blocos, carimbando cada um deles com um registro de tempo e data. A cada período de 10 minutos é formado um novo bloco de transações, o qual se liga ao bloco anterior. Os blocos são dependentes uns dos outros e formam uma cadeia de blocos. Isso torna a tecnologia perfeita para o registro de informações que necessitam de confiança, como no caso de uma transação de moeda.

A rede do *BlockChain* é formada por mineradores que verificam e registram as transações no bloco.

Ao invés dessas informações ficarem armazenadas em um computador central, no *BlockChain*, essa mesma informação fica armazenada em milhares de computadores espalhados pelo mundo inteiro, um claro exemplo de aplicação distribuída.

Cada computador da rede fica com uma cópia integral do banco de dados, o que torna as informações inseridas nele extremamente seguras e confiáveis, porque não há um ponto único de ataque. Logo, não podemos ir a um computador central do *BlockChain* e roubar os registros de transações e modificá-los, porque cada computador dessa rede possui um registro dessas informações. Se você tentar modificar o banco de dados de um computador, ele será expulso pelo restante da rede (BASSOTTO, 2018).

COMPUTAÇÃO EM NUVEM

O termo computação em nuvem (do inglês, *cloud computing*) se refere a uma tecnologia que possibilita acessar recursos e serviços via internet, sem a necessidade de instalações de softwares em seu computador. Dessa forma, é permitido que os usuários façam acessos por meio de qualquer dispositivo, seja ele um computador ou telefone celular.

EXEMPLIFICANDO

Um exemplo de utilização de serviços em nuvem é quando você edita um trabalho da universidade utilizando o Google Docs e esse trabalho fica armazenado na nuvem. Outro exemplo é quando você escuta uma música no Spotify, ou até mesmo assiste a um filme no Netflix. Todas essas situações trazem um contato com a computação em nuvem. Elas têm em comum o fato de não exigirem nenhuma instalação nem consumirem recursos do seu dispositivo, todas são serviços on-line. Para acessá-los, só precisamos de um navegador e ter conexão com a internet.

Ver anotações

A Figura 3.13 ilustra múltiplos dispositivos acessando a nuvem, a qual contém diversos tipos de dados.

Figura 3.13 | *Cloud computing*



Fonte: Shutterstock.

1 O PAPEL DA VIRTUALIZAÇÃO EM SISTEMAS DISTRIBUÍDOS

Dois tipos de virtualização são muito úteis no contexto de sistemas distribuídos, conforme Coulouris *et al.* (2013):

- Virtualização de redes.
- Virtualização de sistemas.

Os autores observam, muito adequadamente, que a vantagem da criação e utilização de redes virtuais advém do fato de que uma rede virtual específica para um determinado tipo de aplicação pode ser criada sobre uma rede física real, de forma que a rede virtual possa ser otimizada para aquela aplicação em particular, sem a necessidade de alterar as características da rede física.

o

Ver anotações

EXEMPLIFICANDO

Imagine que você está desenvolvendo um sistema distribuído de um serviço de *streaming* de vídeo que, obviamente, é composto por vários elementos, como banco de dados, servidor web, servidor de e-mail, servidor de autenticação, etc. Supondo que o arquiteto de sistemas da empresa optou por utilizar dois bancos de dados diferentes: um para armazenar informações gerais, como dados dos assinantes, datas de vencimento de assinaturas, etc., do tipo SQL; e outro para armazenar os vídeos em si, com características mais adequadas para otimizar a troca de informações, do tipo NoSQL. Esses dois bancos de dados não precisam (e, por questões de segurança, nem devem) saber da existência um do outro. Para atingir esse objetivo, tipicamente as empresas criam duas redes virtuais dedicadas, de forma que, além de estarem isoladas entre si, podem ser otimizadas de acordo com a natureza do banco de dados, prevalecendo a comunicação de segmentos UDP para o banco de dados NoSQL, por exemplo, em detrimento aos segmentos TCP.

Para Coulouris et al. (2013), a virtualização de sistemas é uma alternativa interessante por permitir emular o hardware de uma máquina física, permitindo que várias máquinas virtuais, cada uma com um sistema operacional, possam coexistir e se comunicar. Os autores ainda salientam que a principal vantagem da virtualização de sistemas está no fato de que aplicações já escritas e validadas, as quais dependem de um sistema operacional em específico, que necessitam se comunicar e interagir com outra aplicação em um sistema operacional diferente, podem assim fazê-lo através da virtualização dos sistemas operacionais, sem a necessidade de que a aplicação seja reescrita novamente ou recompilada.

REFLITA

Pense na seguinte situação: na empresa que você trabalha, o sistema de planejamento de recursos, conhecido simplesmente por ERP (do inglês, *Enterprise Resource Planning*), utiliza um módulo de controle de estoque escrito pelo próprio time de desenvolvedores de sua empresa, na linguagem Asp.NET, que, por sua vez, utiliza funcionalidades específicas da plataforma Windows, não sendo 100% compatível com alternativas, como o .NET Core (recentemente aberto para a comunidade). Para economizar os gastos e, principalmente, aumentar a disponibilidade do sistema ERP utilizado pelos colaboradores da empresa, suas filiais e seus representantes comerciais, o Diretor Executivo, ou CEO (do inglês, *Chief Executive Officer*), decide migrar para uma solução em nuvem, de algum provedor de *cloud computing* de mercado. Por questões financeiras, o Diretor de TI, ou CIO (do inglês, *Chief Information Officer*), opta por utilizar um sistema operacional GNU/Linux. Como fazer para adaptar o módulo de controle de estoque? Reescrever o código seria uma opção, mas os desenvolvedores que fizeram esse módulo já não trabalham mais na empresa, e o código é muito extenso e complexo, o que significa que sua reescrita impactaria em um aumento significativo de tempo até que o “novo” ERP esteja disponível. Qual seria sua solução para esse problema?

0
Ver anotações

Se você já utilizou ou leu a respeito de computação em nuvem, deve saber que, independentemente do tipo de serviço que você contrata e do provedor desse serviço, você já estará utilizando a virtualização em algum nível. Esses serviços são tipicamente categorizados como IaaS (do inglês, *Infrastructure as a Service*), PaaS (do inglês, *Platform as a Service*) e SaaS (do inglês, *Software as a Service*). Para entender melhor essa ideia, veja a Figura 3.14.

Figura 3.14 | Serviços em nuvem e níveis de virtualização

IaaS	PaaS	SaaS
Você gerencia/controla: <ul style="list-style-type: none">Sistema OperacionalAplicação(ões)Bibliotecas e componentes necessários à aplicação (ex. JDK, JRE, etc.)	Você gerencia/controla: <ul style="list-style-type: none">Aplicação(ões)	Você gerencia/controla: <ul style="list-style-type: none">Nada: apenas utiliza a aplicação (ex. E-mail)
Recursos virtualizados: <ul style="list-style-type: none">ServidoresArmazenamentoRede	Recursos virtualizados: <ul style="list-style-type: none">ServidoresArmazenamentoRedeSistema OperacionalBibliotecas e componentes necessários à aplicação (ex. JDK, JRE, etc.)	Recursos virtualizados: <ul style="list-style-type: none">ServidoresArmazenamentoRedeSistema OperacionalBibliotecas e componentes necessários à aplicação (ex. JDK, JRE, etc.)Aplicação(ões)

0
Ver anotações

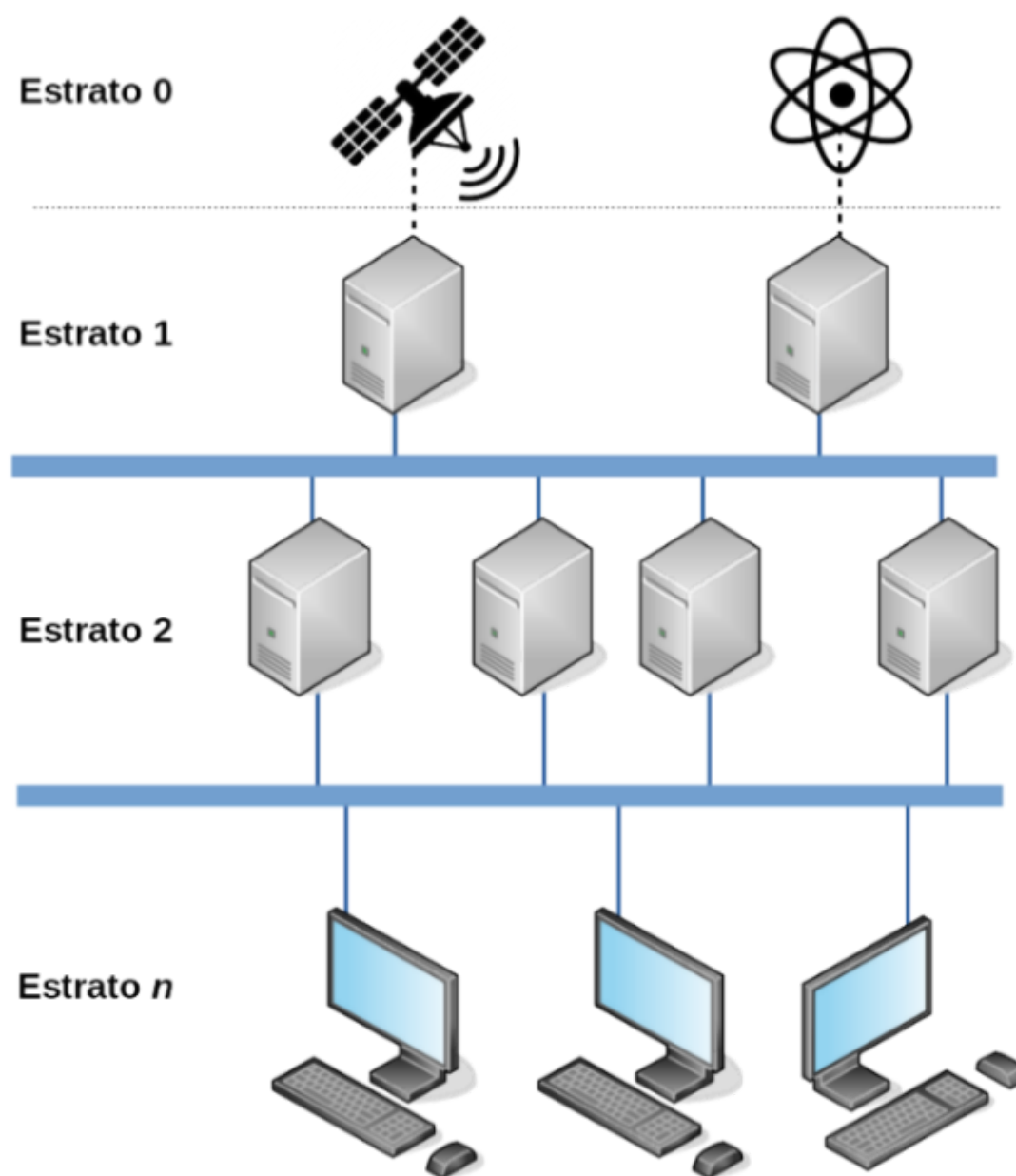
Fonte: elaborada pelo autor.

SINCRONIZAÇÃO DE RELÓGIOS

Sistemas formados por múltiplos computadores necessitam sincronizar suas ações entre si, e uma das maneiras mais utilizadas, dada sua simplicidade e popularidade, é A sincronização horária, por meio do protocolo conhecido como *Network Time Protocol* (NTP) (NTP, 2018). Esse protocolo, por sua vez, utiliza o protocolo de transporte de dados *User Datagram Protocol* (UDP), operando na porta 123. Essencialmente, esse protocolo é utilizado para sincronização do relógio das máquinas locais (desktops, notebooks, servidores) e demais dispositivos de rede.

A referência horária é dada por sistemas de altíssima precisão, como os relógios atômicos (NTP, 2018). Dada a precisão desses sistemas, computadores conectados a eles pertencem a uma camada de servidores chamada de estrato 1 (os sistemas de alta precisão em si pertencem a uma camada topológica chamada de estrato 0). Segundo NTP (2018), como não existem muitos servidores no mundo conectados diretamente a relógios atômicos, outros servidores são conectados aos de estrato 1, os quais, por sua vez, formam uma segunda camada de servidores de horário, chamada de estrato 2, e essa hierarquia se estende até os servidores de estrato 15, conforme podemos ver na Figura 3.15. Os computadores dos usuários são configurados para atualizarem a informação horária por meio da rede, consultando servidores de estratos com valores mais altos.

Figura 3.15 | Sistema de servidores NTP



Fonte: elaborada pelo autor.

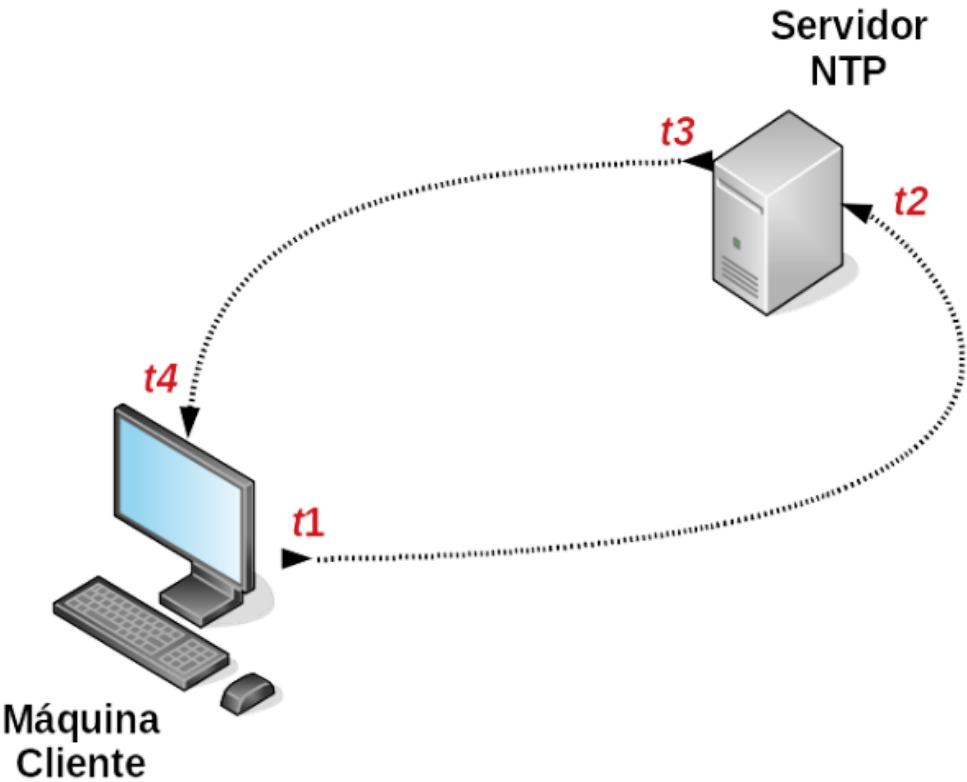
Aqui no Brasil, por exemplo, temos o Observatório Nacional (ON), que é o órgão oficial responsável pela geração, conservação e disseminação da Hora Legal Brasileira.

Um aspecto muito interessante do protocolo NTP é que ele é projetado para verificar a latência (atraso, *delay*) entre a máquina cliente e a máquina servidora, e a implementação disso é muito simples: essencialmente, de tempos em tempos, a máquina cliente faz uma consulta a um servidor NTP para verificar em quanto seu relógio está atrasado (ou adiantado) em relação ao servidor horário de referência, processando a seguinte operação: $(t_4 - t_1) - (t_3 - t_2)$, em que t_1 é a hora, minuto, segundo e milésimos de segundo (também chamada de *timestamp*) da máquina cliente ao enviar uma requisição (através de um pacote, no contexto de redes de computadores) para o servidor NTP; t_2 é o *timestamp* do servidor ao receber essa requisição; t_3 refere-se ao *timestamp* em que um pacote de resposta

a essa requisição é enviado ao cliente; e t_4 é o *timestamp* em que o cliente recebe a resposta do servidor NTP. Esse cálculo, portanto, resulta em quanto o sistema operacional deverá atrasar (ou adiantar) o relógio da máquina local para que ela esteja sincronizada com a referência horária em questão, por exemplo, a Hora Legal Brasileira, em nosso caso. Um exemplo desse processo é ilustrado na Figura 3.16.

Ver anotações

Figura 3.16 | Exemplo de sincronização através do protocolo NTP



Fonte: elaborada pelo autor.

Alguns serviços, como os de acesso remoto e de autenticação de usuários, podem não funcionar adequadamente, caso haja uma diferença muito grande no horário da máquina cliente (solicitante) em relação à máquina servidora (que roda e disponibiliza o serviço). Por esse motivo, é muito importante saber como habilitar a sincronização horária das máquinas utilizando o NTP. A configuração, bem como suas etapas, varia entre os sistemas operacionais e entre suas versões.

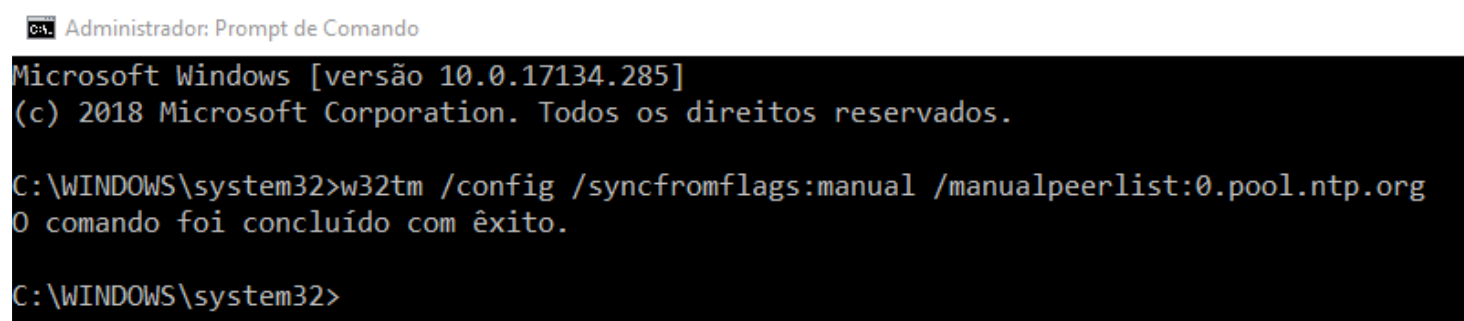
EXEMPLIFICANDO

Um exemplo utilizando um sistema operacional Windows 10 a partir do procedimento a seguir:

1. Abra o Prompt de Comando (CMD), que pode ser acessado pelo menu iniciar, ou pelo campo de busca, digitando prompt ou cmd.
2. Na janela do CMD, insira o código a seguir e pressione a tecla “Enter” (Figura 3.17):

```
w32tm /config /syncfromflags:manual /manualpeerlist:0.pool.ntp.org
```

Figura 3.17 | Apontando para o servidor NTP

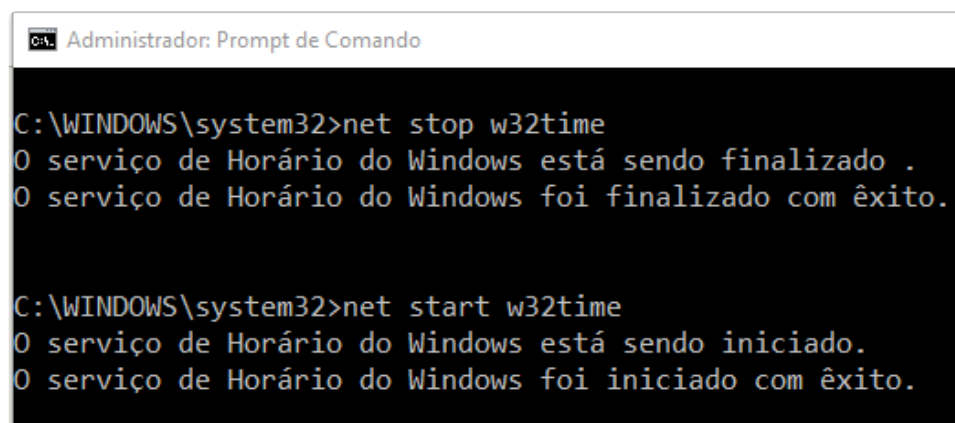


Fonte: captura de tela do Prompt de Comando do Windows 10.

Na Figura 3.17, podemos observar que, ao final do comando, aparece a mensagem “O comando foi concluído com êxito”. Lembre-se: para executar os comandos, você deve estar como um usuário Administrador ou executar o CMD como Administrador. Agora que fizemos o apontamento para o servidor NTP que utilizaremos, devemos reiniciar o serviço de data e hora para aplicar as alterações:

3. Utilize os comandos `net stop w32time` e `net start w32time` para parar o serviço e iniciar, reiniciando-o, conforme podemos observar na Figura 3.18:

Figura 3.18 | Reiniciando o serviço de data e hora

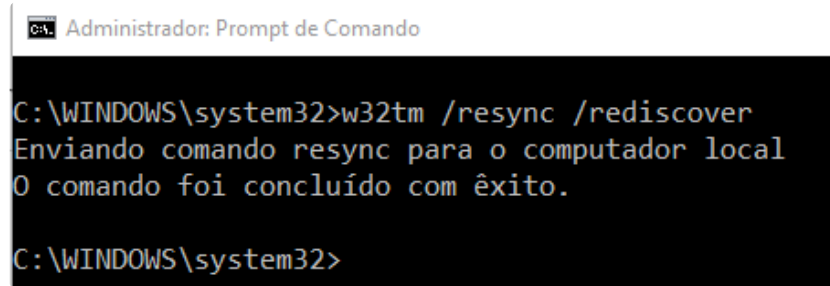


Fonte: captura de tela do Prompt de Comando do Windows 10.

Após isso, forçaremos uma sincronização de data e hora para verificarmos se está tudo funcionando corretamente.

4. Force uma sincronização por meio do comando `w32tm /resync /rediscover`, conforme observamos na Figura 3.19.

Figura 3.19 | Sincronizando data e hora



```
C:\WINDOWS\system32>w32tm /resync /rediscover
Enviando comando resync para o computador local
O comando foi concluído com êxito.

C:\WINDOWS\system32>
```

Fonte: captura de tela do Prompt de Comando do Windows 10.

Por meio da saída da Figura 3.19, podemos observar a mensagem que será dada, informando que nossa configuração foi feita com sucesso.

Continue firme em seus estudos, pois ainda vem muita coisa interessante e útil pela frente. Assim finalizamos esta seção. Esperamos que o conhecimento adquirido seja de importância para seu crescimento profissional. Bons estudos e até a próxima seção!

FAÇA VALER A PENA

Questão 1

Os sistemas distribuídos têm muitas características semelhantes a quaisquer outros sistemas de rede. Entretanto, uma diferença principal está no fato de que, nos sistemas distribuídos, a aplicação é replicada (ou distribuída) entre as diferentes máquinas, comportando-se como se estivesse rodando em uma única máquina.

Considerando os conceitos inerentes aos sistemas distribuídos, analise as afirmativas a seguir:

- I. As aplicações se comportam como se estivessem rodando em uma única máquina.
- II. Os sistemas distribuídos possuem um elemento central (intermediário), conhecido como *middleware*, entre o sistema operacional e a aplicação.
- III. Sistemas distribuídos são, necessariamente, sistemas que possuem sistemas operacionais especiais.

0

Ver anotações

- IV. Os sistemas distribuídos são utilizados em aplicações modernas de grande porte.
- V. Uma das características dos sistemas distribuídos é a facilidade para incluir novas máquinas em um sistema em funcionamento.

É correto o que se afirma em:

- a. I, apenas.
- b. II e III, apenas.
- c. III, apenas.
- d. I, II e IV, apenas.
- e. I, II, III, IV e V.

0
Ver anotações

Questão 2

Sistemas formados por múltiplos computadores, como os sistemas distribuídos, necessitam sincronizar suas ações entre si, e uma das maneiras mais utilizadas, dada sua simplicidade e popularidade, é a sincronização horária, a qual é necessária para o funcionamento da maioria das aplicações.

Para fazer a sincronização horária entre os computadores que foram os nossos sistemas distribuídos, utilizamos um protocolo muito popular. Assinale a alternativa que corresponde ao protocolo utilizado para sincronização horária:

- a. Transmission Control Protocol(TCP).
- b. Network Time Protocol(NTP).
- c. User Datagram Protocol(UDP).
- d. HyperText Transfer Protocol(HTTP).
- e. SSH Remote Protocol(SSH).

Questão 3

Os sistemas distribuídos estão presentes em quase todas as aplicações que utilizamos atualmente. Além disso, podemos apontar vários tipos de sistemas distribuídos, como computação em *cluster* e em *grid*, que são muito populares

quando falamos em sistemas distribuídos.

Considerando as características da computação em *cluster* e *grid*, analise as afirmativas a seguir:

- I. As máquinas que compõem um sistema com computação em *grid* têm características homogêneas.
- II. As máquinas que compõem um sistema com computação em *cluster* têm características heterogêneas.
- III. Entre a computação em *cluster* e *grid*, não conseguimos apontar diferenças.
- IV. Geralmente, o sistema operacional e o hardware das máquinas que compõem um sistema em computação em *cluster* são equivalentes.
- V. Geralmente, o sistema operacional e o hardware das máquinas que compõem um sistema em computação em *cluster* são de fabricantes e configurações diferentes.

É correto o que se afirma em:

a. I, apenas.

b. II e IV, apenas.

c. IV, apenas.

d. V, apenas.

e. I e II, apenas.

REFERÊNCIAS

COMER, D. E. **Redes de computadores e internet**. Porto Alegre: Bookman, 2016.
Disponível em: <http://bit.ly/3cKdZp2>. Acesso em: 11 nov. 2020.

DEITEL, H. M.; DEITEL, P. J.; CHOFFNES, D. R. **Sistemas operacionais**. 3. ed. São Paulo: Pearson Prantice Hall, 2005.

DIAMANDIS, P. H.; KOTLER, S. **Bold**: oportunidades exponenciais: um manual prático para transformar os maiores problemas do mundo nas maiores oportunidades de negócio. Rio de Janeiro: Alta Books, 2018.

FOROUZAN, B. A. **Comunicação de dados e redes de computadores**. 4. ed. Porto Alegre: AMGH, 2010.

KUROSE, J. F.; ROSS, K. W. **Redes de computadores e a internet**: uma abordagem top-down. 6. ed. São Paulo: Pearson Education do Brasil, 2013.

LAUDON, K. **Sistemas de informações gerenciais**. 11. ed. São Paulo: Pearson Prentice Hall, 2014.

LOPER, A. A.; SILVA, N. S.; LOPES, G. M. B. **Projeto de redes e sistemas distribuídos**. Londrina: Editora e Distribuidora Educacional S.A., 2019.

MEDEIROS, J. C. O. **Princípios de telecomunicações**: teoria e prática. 5. ed. São Paulo: Érica, 2015.

ROBERTS, M. J. **Fundamentos de sinais e Sistemas**. Porto Alegre: AMGH Editora, 2009.

SIQUEIRA, E. **Revolução digital**: história e tecnologia no século 20. São Paulo: Saraiva, 2007.

STALLINGS, W. **Redes e sistemas de comunicação de dados**. Rio de Janeiro: Elsevier, 2016. Disponível em: <http://bit.ly/2OLEzpS>. Acesso em: 11 nov. 2020

TANENBAUM, A. S. **Redes de computadores**. 5. ed. São Paulo : Pearson Prentice Hall, 2011.

0

Ver anotações