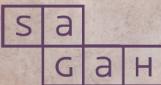


FUNDAMENTOS DE BIG DATA



SOLUÇÕES
EDUCACIONAIS
INTEGRADAS

MapReduce no Hadoop: arquitetura de análise distribuída

Roger Robson dos Santos

OBJETIVOS DE APRENDIZAGEM

- > Resumir o histórico do MapReduce no Hadoop.
 - > Descrever a arquitetura e componentes do MapReduce implementado no Hadoop.
 - > Explicar o processo de implementação do MapReduce.
-

Introdução

Grandes empresas enfrentam o desafio de lidar com imensos volumes de dados diariamente. Uma das tecnologias utilizadas para resolver esse problema é o Hadoop, que possibilita armazenar grandes volumes de dados sem muitos esforços computacionais. O Hadoop exige um profissional que tenha conhecimentos relativos à *application programming interface* (API) do MapReduce, tanto no âmbito de sua utilização quanto no âmbito de seu funcionamento.

Neste capítulo, você vai conhecer a história do MapReduce no Hadoop, bem como a arquitetura e o processo de funcionamento dessa ferramenta. Além disso, vai ver quais são os primeiros passos para a implementação dessa tecnologia.

História do MapReduce no Hadoop

A história do MapReduce se iniciou em 2008, com os cientistas da Google Jeffrey Dean e Sanjay Ghemawat. Esses cientistas popularizaram um novo modelo de programação, chamado “MapReduce”, a partir de seu artigo “MapReduce: simplified data processing on large clusters” (DEAN; GHEMAWAT, 2008). Nesse artigo, eles discutem a abordagem utilizada pela Google para otimizar a coleta e a análise de dados em seus sistemas de pesquisa. O MapReduce proprietário da Google era executado no Google File System (GFS). Logo a Apache começou a utilizar o MapReduce em seu projeto Nutch, considerado um mecanismo rastreador da web altamente extensível e escalável em código aberto, ativo até hoje.

Com o passar do tempo, a aplicação Apache Hadoop, como um subprojeto do Apache Lucene, tornou-se capaz de fornecer recursos de pesquisa de texto para grandes bancos de dados. Em 2006, Doug Cutting, um funcionário da Yahoo!, projetou o Hadoop como um subprojeto ainda em fase de testes. Porém, mesmo assim Cutting o lançou como um projeto oficial do Apache de código aberto em 2007 (APACHE HADOOP, 2020). Em 2008, foi criado um *cluster* experimental com 4 mil nós usando Hadoop, o que levou essa aplicação a um nível superior na Apache. Logo em 2009, o Hadoop se mostrou muito eficaz: durante um teste, ele foi capaz de classificar 1TB de dados em apenas 17 horas (BHANDARDKAR, 2020).

O Apache Hadoop logo se popularizou na indústria do comércio eletrônico. Hoje, diversas empresas utilizam o Hadoop como ecossistema para o processamento de grandes volumes de dados. É o caso de Amazon, Facebook e Yahoo!. O Hadoop utiliza a biblioteca MapReduce para processar análises de dados e determinar a intenção dos usuários a partir de pesquisas informadas em seus sites, por meio de análises estatísticas.

O MapReduce consegue processar dados de fluxos de cliques, pesquisas e atividades lucrativas, bem como armazenar e minerar com eficácia terabytes de dados que são coletados diariamente. Ele também atende a empresas como a The New York Times Company, que armazena dados de vídeos e imagens.



Fique atento

Para utilizar o MapReduce, não é necessário fazer um grande investimento em infraestrutura. Outro ponto interessante é que o Hadoop já está disponível em muitos provedores de nuvem pública com pagamento por uso e otimizações na plataforma.

O Apache Hadoop introduziu a ideia de realizar trabalhos com grandes volumes de dados em *hardware* comum. O Hadoop permite a utilização de *clusters* de computadores para ampliar seu *hardware*. Além disso, esse *software* possui ferramentas que detectam e tratam falhas, tornando o sistema altamente disponível para utilização de máquinas em *clusters*. O Hadoop é desenvolvido na linguagem Java.

O Hadoop é formado por diversos componentes. Três desses componentes são os principais. Veja a seguir.

- **Hadoop Common:** é um componente que contém utilitários usados em toda a aplicação. Possui diversas bibliotecas para manipulação e serialização de arquivos. Esse componente também é utilizado para disponibilizar integrações de interface para outros sistemas, como CloudSource e AWS S3.
- **Hadoop Distributed File System (HDFS):** é um sistema de arquivo distribuído altamente tolerante a falhas que permite o armazenamento e a manipulação de grandes conjuntos de dados em *clusters* de máquinas de baixo custo.
- **Hadoop MapReduce:** é uma biblioteca especializada no processamento de grandes conjuntos de volumes de dados distribuídos, possuindo duas funções principais, a `Map` e a `Reduce`.

Arquitetura e componentes MapReduce e HDFS

A análise de dados realizada pelo Hadoop permite o processamento paralelo na aplicação. Entretanto, é necessário entender os dois módulos que realizam essa atividade: o HDFS é o módulo de armazenamento, enquanto o MapReduce é o módulo de processamento dos dados.

HDFS

O HDFS é um sistema escalável de arquivos distribuídos baseado no GFS. Essa aplicação tem a função de garantir a escalabilidade do sistema à medida que é necessário armazenar grandes volumes de dados em uma única máquina ou em um *cluster* (BORTHAKUR, 2019).

O HDFS possui blocos de dados normalmente de tamanhos fixos (64MB). Assim, um arquivo muito grande pode ter blocos armazenados em diversas máquinas dentro do *cluster*. Além disso, o HDFS possui réplicas que adicionam confiabilidade ao armazenamento de dados e melhoram o processamento paralelo. A seguir, veja quais são os componentes do HDFS.

- **NameNode:** conhecido como “*master*”, é o componente que armazena os metadados, monitora o *status* e a integridade dos dados e atribui tarefas do **DataNode**.
- **DataNode:** é o componente que armazena os blocos com os dados e retorna ao NameNode o *status* do seu funcionamento. Caso uma tarefa do DataNode falhe ou pare de responder, ela será reatribuída ao próximo DataNode.
- **SecondaryNameNode:** é um NameNode secundário que armazena as atualizações do NameNode em intervalos de tempo predefinidos. Por isso, é também conhecido como “*checkpoint*”. Ele garante a recuperação do estado em dado ponto.



Fique atento

NameNodes e DataNodes são componentes desenvolvidos em Java. Ou seja, eles não necessitam de um sistema operacional específico para executar o HDFS; exigem somente a instalação da aplicação Java Development Kit (JDK).

Na Figura 1, veja como o HDFS funciona. Na primeira etapa, o cliente HDFS abre o arquivo que deseja ler no *DistributedFileSystem*, que é uma instância do HDFS. Na segunda etapa, o *DistributedFileSystem* chama o **NameNode** para determinar as localizações dos primeiros blocos e retornar o endereço dos arquivos que estão nos **DataNodes**. Além disso, essa classificação é feita de acordo com a topologia de rede do *cluster* (GOLDMAN *et al.*, 2012).

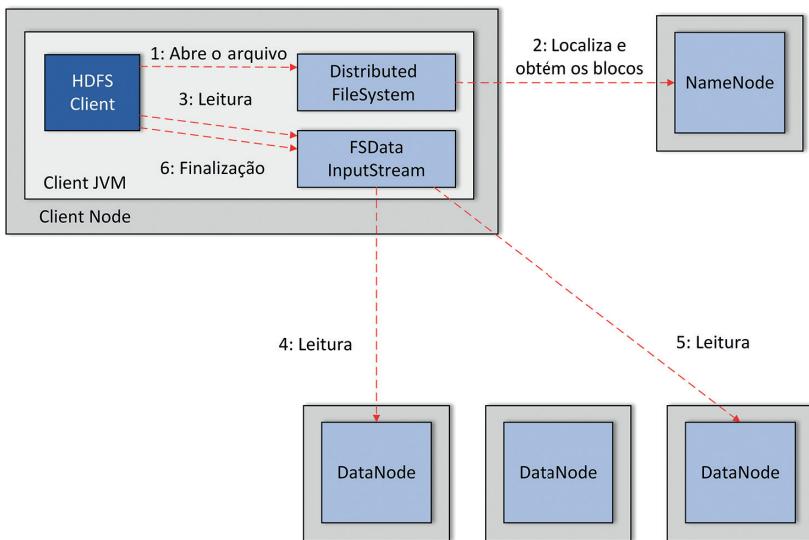


Figura 1. Arquitetura de funcionamento do HDFS.

Fonte: Adaptada de White (2015).

Com a posição do DataNode, o DistributedFileSystem retorna um FSDaInputStream (fluxo de entrada para suporte a buscas de arquivos) para que o cliente possa ler os dados. Em seguida, o FSDaInputStream envia os dados para o DFSInputStream, que fará o gerenciamento do DataNode e do NameNode. Assim se inicia a terceira etapa, na qual o DFSInputStream lê os endereços armazenados e então se conecta ao DataNode mais próximo para coletar os primeiros blocos de arquivos.

Na quarta etapa, os dados são transmitidos do DataNode de volta ao cliente; esse fluxo ocorre até a finalização da coleta desse bloco. Em seguida, o DFSInputStream fecha a conexão com o DataNode e vai para o próximo DataNode. Assim se inicia a quinta etapa, que se repete até o final dos blocos requisitados ser alcançado. Por fim, a finalização ocorre na sexta etapa, que chama o FSDaInputStream para fechar as conexões.



Fique atento

Caso ocorra algum erro durante o processo mostrado na Figura 1, o DFSInputStream seguirá para o próximo bloco e informará que houve falha em um bloco específico.

MapReduce

O MapReduce é um *framework* responsável por realizar funções de mapeamento e redução. Essas tarefas devem rodar em paralelo entre as máquinas do *cluster* e em conjunto com o sistema de armazenamento HDFS. A seguir, veja quais são as funções do MapReduce.

- **Map:** essa função recebe os dados de entrada e os coloca em uma coleção de pares de chave/valor. Esses pares devem ser definidos pelo programador em Java ou em linguagens suportadas pelo Hadoop.
- **Sort/Shuffle:** o Sort organiza os dados recebidos pela função Map e atribui entrada para cada Reduce associado à mesma chave. Já o Shuffle transfere as saídas da função Map para os Reduce como entrada inicial. Essa função é feita na biblioteca MapReduce.
- **Reduce:** essa função recebe os dados do Shuffle e retorna uma lista de chave/valor contendo os registros. Essa função também deve ser definida pelo programador, assim como a Map.
- **JobTracker:** é uma função que auxilia dando suporte para a realização de tarefas das funções Map e Reduce, por meio de coordenadas enviadas para os TaskTrackers. Essa função cria um processo JVM para cada TaskTracker e aloca nós responsáveis por processar as tarefas da aplicação e monitorá-las.
- **TaskTracker:** é o processo responsável por executar as tarefas do Map e do Reduce e informar os progressos ao JobTracker. Assim como os DataNodes descritos anteriormente, essa aplicação é composta por diversas instâncias de TaskTrackers. Dessa maneira, é possível utilizar diversas máquinas para aumentar os recursos computacionais disponíveis no cluster.

A Figura 2 apresenta a arquitetura do funcionamento do MapReduce com o HDFS. Nessa arquitetura do HDFS, cada nó roda em uma parte do *cluster*, em que os dados estão armazenados e replicados entre os DataNodes e são supervisionados pelo NameNode. Enquanto isso, o JobTracker acompanha a submissão das tarefas do MapReduce e efetua o monitoramento e o escalonamento das atividades nos TaskTrackers. Ou seja, o TaskTracker é responsável por executar as tarefas designadas pelo JobTracker.

Um trabalho do MapReduce funciona em duas fases, Map e Reduce. Cada fase trabalha com pares de valores e chaves como entrada e saída. Os tipos das chaves podem ser escolhidos pelo programador do sistema para criar as funções Map e Reduce.

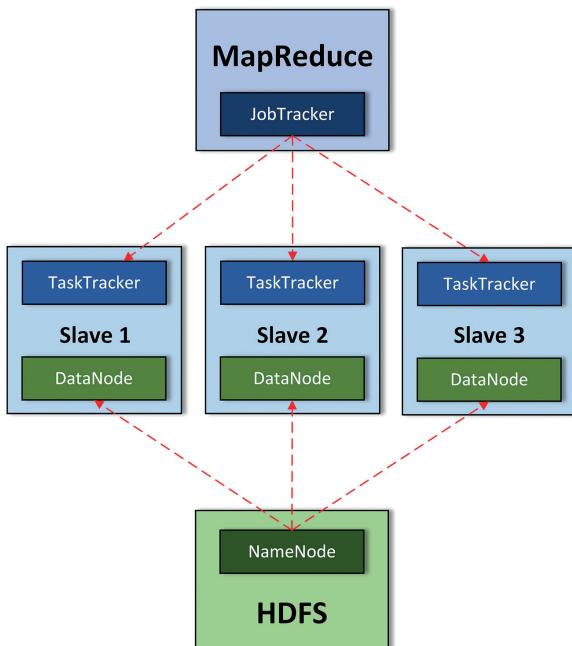


Figura 2. Arquitetura de funcionamento do MapReduce com HDFS.

A Figura 3 apresenta as fases das funções **Shuffle** e **Sort** aplicadas durante as tarefas **Map** e **Reduce**. A função **Map** coleta os dados de entrada e os divide em blocos para as tarefas **Reduce**. Esses dados ficam ordenados conforme as chaves criadas, predefinidas e alocadas em uma fila para envio ao armazenamento. Se o limite de *buffer* da memória é atingido, a tarefa é suspensa e aguarda até que as operações sejam escritas em disco. O mecanismo **Shuffle** permite que as saídas da tarefa **Map** sejam comprimidas com as ferramentas disponíveis do sistema operacional, como bzip2 e gzip (WHITE, 2015).

Ainda na fase **Shuffle**, as tarefas em execução pela função **Map** são alocadas nos discos e nos nós do *cluster* Hadoop. Na fase **Reduce**, esses dados são distribuídos de acordo com tarefas. Os gerenciadores de tarefa sabem organizar as tarefas das funções **Map** e **Reduce** à medida que uma execução inicia e finaliza. Assim que todos os dados ficam disponíveis no **Reduce**, é realizada a tarefa **Sort**, para ordenação das junções dos dados. Ao final, quando os dados já foram processados pelas funções **Map** e **Reduce**, eles são escritos no sistema de arquivos HDFS e ficam disponíveis para o usuário.

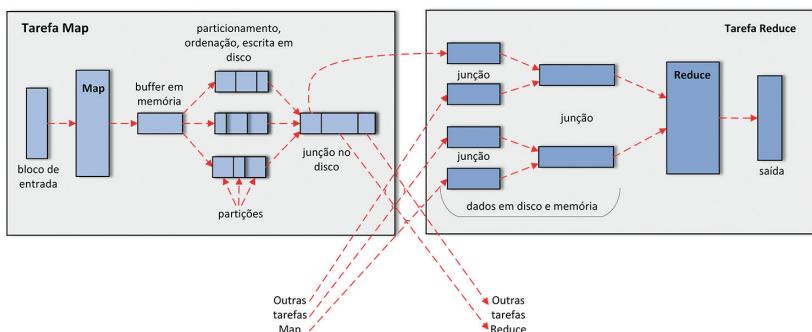


Figura 3. Funções **Shuffle** e **Sort** aplicadas durante as tarefas **Map** e **Reduce** no Hadoop.
Fonte: Adaptada de White (2015).

A seguir, você vai ver um exemplo que utiliza dados de temperatura (ano, temperatura). Observe a Figura 4. A função **Map** é a fase de preparação dos dados para utilização na função **Reduce**. A **Map** prepara os dados do ano e da temperatura obtidos do HDFS. Além disso, permite incluir dados considerados registros ruins, que posteriormente serão filtrados (WHITE, 2015).

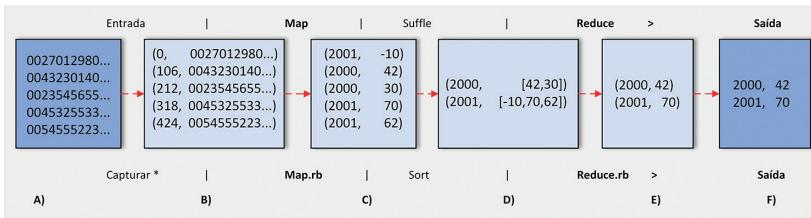


Figura 4. Fluxo de dados lógico do MapReduce.

Fonte: Adaptada de White (2015).

O fluxo de dados lógico do processo Map é apresentado inicialmente na Figura 4a, que mostra uma parte dos dados utilizados para entrada no Map. A Figura 4b apresenta os dados de entrada já com os pares de chaves adicionados a cada valor. Em seguida, na Figura 4c, note que as chaves são deslocadas para dentro da linha do arquivo, enquanto a função Map extrai os dados e emite uma saída com conjunto que será utilizada na próxima fase.

Na Figura 4d, a função Sort organiza os dados recebidos pela função Map e atribui entrada para cada Reduce associado à mesma chave. Já o Shuffle transfere as saídas do Map para os Reduce como entrada inicial. Essa função é feita na biblioteca MapReduce. Na Figura 4e, a função Reduce classifica e agrupa os pares de chaves divididos entre os anos e as maiores temperaturas encontradas. Por fim, a Figura 4f apresenta a saída completa dos dados após o processo.

Processo de implementação do ambiente Hadoop

O Apache Hadoop (2020) é uma aplicação conhecida por executar em um conjunto de máquinas chamadas “cluster”. Entretanto, essa aplicação também pode ser executada em apenas uma máquina. Dessa maneira, a aplicação roda com configurações simplificadas nas fases iniciais de implementação e testes. É importante lembrar que não é obrigatório executar aplicações de forma distribuída; entretanto, se você quiser obter desempenhos superiores, precisa utilizar ao menos duas máquinas.

O Hadoop possui três modos de execução durante sua implementação, configurados em três arquivos: `core-site.xml`, `hdfs-site.xml` e `mapred-site.xml`. Esses arquivos permitem a execução do Hadoop em

diferentes modos: modo local (*standalone mode*), modo pseudodistribuído (*pseudo-distributed mode*) e modo completamente distribuído (*fully distributed mode*) (LAM, 2010). A seguir, veja como o ambiente Hadoop pode ser configurado com poucas modificações em pequenos arquivos.

Modo local

Nessa configuração, o Hadoop utiliza apenas um computador, ou seja, é configurado para rodar em modo local. Para utilizar esse modo, é necessário alterar os parâmetros dos arquivos mencionados anteriormente. Nessa configuração, o Hadoop utiliza somente os recursos da máquina local e não é necessário utilizar o HDFS, já que os dados não estão distribuídos entre outras máquinas. Esse modo é recomendado para usuários iniciantes e durante a fase de testes, que não necessita de tanto processamento para a execução dos códigos.

Modo pseudodistribuído

Nesse modo, o Hadoop aplica configurações semelhantes às utilizadas em um *cluster*, porém ele executa a aplicação em modo local. Esse modo é conhecido como “*cluster* de uma máquina só”. A principal diferença em relação ao modo local com o modo pseudodistribuído, é que no modo pseudodistribuído, apesar de a aplicação ser local, ela permite a execução paralela em uma única máquina utilizando NameNode, DataNode, JobTracker, TaskTracker e SecondaryNameNode.

No exemplo a seguir, veja a configuração padrão necessária do arquivo `core-site.xml` para execução em modo pseudodistribuído. A linha 5 especifica o nome da variável de sistema, e a linha 6 especifica a localização como HDFS dentro do *localhost*, que é máquina local, na porta 9000. As demais configurações são padrão do arquivo.

```
1.      <!-- core-site.xml -->
2.      <?xml version="1.0"?>
3.      <configuration>
4.          <property>
5.              <name>fs.default.name</name>
6.              <value>hdfs://localhost:9000</value>
7.              <description>The name of the default file
system. A URI whose scheme and authority determine the
FileSystem implementation.</description>
8.          </property>
9.      </configuration>
```

O exemplo a seguir mostra a configuração do arquivo `mapred-site.xml`. Como você já viu, a linha 5 define a variável de sistema, e a linha 6 aplica a localização do servidor com a porta 9001. Nesse arquivo, não é necessário inserir `hdfs` antes (como no arquivo `core-site.xml`).

```
1.      <!-- mapred-site.xml -->
2.      <?xml version="1.0"?>
3.      <configuration>
4.          <property>
5.              <name>mapred.job.tracker</name>
6.              <value>localhost:9001</value>
7.              <description>The host and port that the Ma-
pReduce job tracker runs at.</description>
8.          </property>
9.      </configuration>
```

A seguir, veja a configuração do arquivo `hdfs-site.xml`. Nesse arquivo, é definido o número de réplicas do HDFS. Por padrão, a linha 6 vem configurada com valor 3; entretanto, como a utilização da máquina é local, é necessário mudar para 1, já que não haverá réplicas para outras máquinas.

```
1.      <!-- hdfs-site.xml -->
2.      <?xml version="1.0"?>
3.      <configuration>
4.          <property>
5.              <name>dfs.replication</name>
6.              <value>1</value>
7.              <description>The actual number of replications
can be specified when the file is created.</description>
8.          </property>
9.      </configuration>
```

Por fim, as últimas configurações estão localizadas nos arquivos *master* e *slave*, local onde deve conter somente o apontamento como `localhost`. Isso ocorre porque a aplicação está sendo utilizada apenas em uma máquina.

Modo completamente distribuído

Por fim, o modo completamente distribuído é o terceiro modo utilizado para execução distribuída em um *cluster* de dois ou mais computadores. Nesse modo, as configurações são as mesmas dos dois primeiros exemplos; na configuração do terceiro exemplo, será alterado apenas o valor 1 da linha 6 para o valor padrão 3.

As principais configurações desse modo estão nos arquivos *masters* e *slaves*. Nestes arquivos deverão ser configurados os IPs ou os nomes das máquinas caso estejam adicionados no arquivo */etc/hosts*, que permite definir um nome para um IP específico, como mostra o exemplo a seguir.

maquina _ mestre	192.168.1.20
Maquina _ slave _ 1	192.168.1.21
Maquina _ slave _ 2	192.168.1.22
Maquina _ slave _ 3	192.168.1.23
Maquina _ slave _ 4	192.168.1.24

A seguir, veja a configuração do arquivo *masters*.

```
maquina _ mestre
```

Por fim, veja a configuração do arquivo *slaves*.

maquina _ slave _ 1
maquina _ slave _ 2
maquina _ slave _ 3
maquina _ slave _ 4

Referências

APACHE HADOOP. *MapReduce tutorial*. California: Apache Software Foundation, 2020. Disponível em: <https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/>. Acesso em: 17 out. 2020.

BHANDARKAR, M. MapReduce programming with apache Hadoop. In: IEEE INTERNATIONAL SYMPOSIUM ON PARALLEL & DISTRIBUTED PROCESSING, 2010. Proceedings [...]. Atlanta, 2010. Disponível em: <https://www.computer.org/csdl/pds/api/csdl/proceedings/download-article/12OmNApLGAg/pdf>. Acesso em: 17 out. 2020.

BORTHAKUR, D. *HDFS architecture guide*. California: Apache Software Foundation, 2019. Disponível em: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html. Acesso em: 16 out. 2020.

DEAN, J.; GHEMAWAT, S. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, v. 51, n. 1, 2008. Disponível em: <https://www.alexdelis.eu/M125/Papers/p107-dean.pdf>. Acesso em: 16 out. 2020.

GOLDMAN, A. et al. Apache Hadoop: conceitos teóricos e práticos, evolução e novas possibilidades. In: JORNADAS DE ATUALIZAÇÃO EM INFORMÁTICA, 31., 2012. Anais [...]. Curitiba, 2012. Disponível em: http://www.inf.ufsc.br/~bosco.sobral/ensino/ine5645/JAI_2012_Cap%203_Apache%20Hadoop.pdf. Acesso em: 16 out. 2020.

LAM, C. *Hadoop in action*. New York: Manning Publications, 2010.

WHITE, T. *Hadoop: the definitive guide*. 4th. ed. Massachusetts: O'Reilly, 2015.

Leituras recomendadas

BENGFORT, B.; KIM, J. *Data analytics with Hadoop: an introduction for data scientists.* Massachusetts: O'Reilly, 2016.

CAPRIOLI, E.; WAMPLER, D.; RUTHERGLEN, J. *Programming Hive: data warehouse and query language for Hadoop.* Massachusetts: O'Reilly, 2012.

MINER, D.; SHOOK, A. *MapReduce design patterns: building effective algorithms and analytics for Hadoop and other systems.* Massachusetts: O'Reilly, 2012.



Fique atento

Os *links* para sites da web fornecidos neste capítulo foram todos testados, e seu funcionamento foi comprovado no momento da publicação do material. No entanto, a rede é extremamente dinâmica; suas páginas estão constantemente mudando de local e conteúdo. Assim, os editores declaram não ter qualquer responsabilidade sobre qualidade, precisão ou integralidade das informações referidas em tais *links*.

Conteúdo:



SOLUÇÕES
EDUCACIONAIS
INTEGRADAS