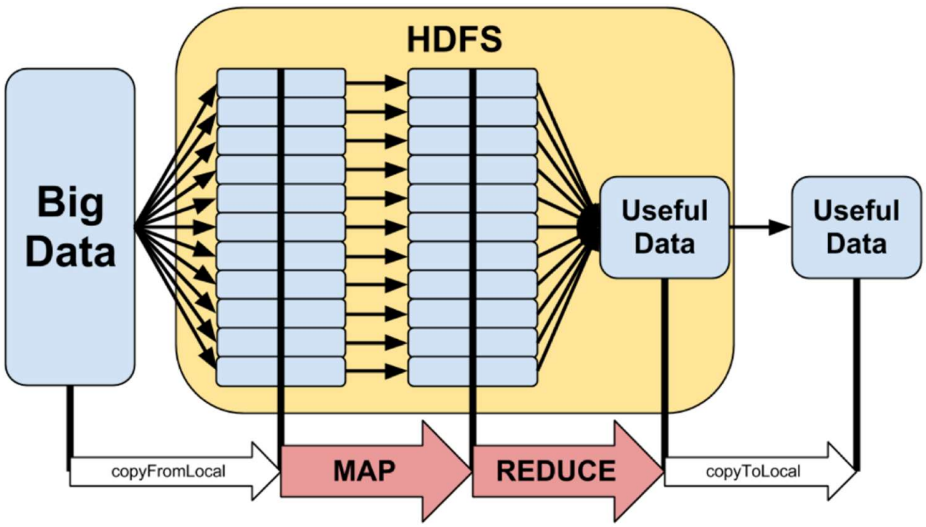
Machine Learning e IA em Ambientes Distribuídos Versão 2.0

Como Funciona o MapReduce

O Framework Hadoop MapReduce programa automaticamente a execução de um aplicativo em um conjunto de máquinas em um cluster. Ele lida com balanceamento de carga, falhas dos nodes, e comunicação entre nodes. O MapReduce cuida dos detalhes complicados de computação distribuída e permite que um Cientista de Dados possa se concentrar na lógica de processamento de dados. O Engenheiro de Dados é responsável por configurar e otimizar o ambiente do MapReduce.



Os blocos de construção básicos de um aplicativo MapReduce são duas funções: Mapeamento e Redução. No Mapeamento, os dados são separados em grupos de chave-valor e geram um resultado que serve como entrada para a função de redução. A função reduce() agrega esses valores e gera um valor agregado como resultado, a partir das regras de redução que foram aplicadas (e que são definidas via programação, pelo Cientista de Dados).

MapReduce é um paradigma de programação, sendo um dos mais antigos e mais conhecidos frameworks para clusterização. Ele segue o modelo de programação funcional, e executa a sincronização explícita por meio de etapas computacionais. A simplicidade do MapReduce é atrativa para os usuários, porém, é importante destacar que possui algumas limitações. Aplicações, por exemplo, de aprendizado de máquina e análise de grafos processam dados de forma iterativa o que significa que muitos processamentos são realizados com os mesmos dados. Para algoritmos iterativos, que realizam a leitura dos dados uma vez, mas interagem sobre eles muitas vezes, o modelo MapReduce representa uma sobrecarga significativa.

O Spark é um framework para clusterização que executa processamento em memória - sem utilização de escrita e leitura em disco rígido - com o objetivo de ser superior aos engines baseados em disco como o MapReduce.

O Spark realiza um processo semelhante ao Hadoop MapReduce, mas para superar as limitações do MapReduce descritas anteriormente, o Spark faz uso de Resilient Distributed Datasets (RDDs), e mais recentemente de dataframes, o qual implementa estruturas de dados em memória e que são utilizadas para armazenar em cache os dados existentes entre os nós de um cluster. Uma vez que as RDDs ficam em memória, os algoritmos podem interagir nesta área várias vezes de forma eficiente. Embora o MapReduce seja projetado para trabalhos em lote, ele é amplamente utilizado para realizar trabalhos iterativos. Por outro lado, o Spark foi concebido principalmente para trabalhos iterativos, mas também pode ser utilizado para realizar trabalhos em lotes. Isso ocorre porque as arquiteturas para análise de Big Data são compostas por várias etapas que trabalham em conjunto sobre os mesmos dados, e que normalmente encontram-se armazenados no HDFS.

O Spark detém resultados intermediários na memória, em vez de escrevê-los no disco, o que é muito útil quando se precisa processar os mesmos conjuntos de dados muitas vezes. Seu projeto teve por objetivo torná-lo um mecanismo de execução que funciona tanto na memória como em disco e, por isso, o Spark executa operações em disco quando os dados não cabem mais na memória. Assim, é possível usá-lo para o processamento de conjuntos de dados maiores que a memória agregada em um cluster. O Spark armazenará a maior quantidade possível de dados na memória e, em seguida, irá persisti-los em disco. Cabe ao Engenheiro de Dados olhar para os seus dados e casos de uso para avaliar os requisitos de memória. Com esse mecanismo de armazenamento de dados em memória, o uso do Spark traz vantagens de desempenho consideráveis. Portanto:

- Spark suporta mais do que apenas as funções de Map e Reduce.
- Hadoop MapReduce grava os resultados intermediários em disco, enquanto o Spark grava os resultados intermediários em memória, o que é muito mais rápido.
- Spark fornece APIs concisas e consistentes em Scala, Java e Python (e mais recentemente em R).
- Spark oferece shell interativo para Scala, Python e R.
- Spark pode utilizar o HDFS como uma de suas fontes de dados.