ResilientDistributedDatasets:AFault-TolerantAbstractionfor In-MemoryClusterComputing

Resilient Distributed Datasets (RDDs) são uma abstração de memória distribuída que permite que programadores realizem cálculos em memória em grandes clusters de forma tolerante a falhas. RDDs podem expressar uma ampla gama de aplicações paralelas, incluindo muitos modelos de programação especializados que foram propostos para computação interativa, e novas aplicações que esses modelos não capturam. Ao contrário das abstrações de armazenamento existentes para clusters, que exigem replicação de dados para tolerância a falhas, RDDs oferecem uma API baseada em transformações de granularidade grosseira que permite que eles recuperem dados de forma eficiente usando a linhagem. Spark é um sistema que implementa RDDs e supera o Hadoop em até 20 vezes em aplicações interativas e pode ser usado interativamente para consultar centenas de gigabytes de dados. RDDs podem ser gravados em segundo plano sem exigir pausas no programa ou esquemas de snapshot distribuídos. RDDs podem se recuperar rapidamente reconstruindo apenas as partições RDD perdidas quando os nós falham. Spark pode ser usado para consultar um conjunto de dados de 1 TB interativamente com latências de 5 a 7 segundos. RDDs capturam vários modelos de programação de cluster existentes, incluindo os modelos de programação Pregel e HaLoop.

1) Spark é um sistema que implementa RDDs e supera o Hadoop em até 20 vezes em aplicações interativas e pode ser usado interativamente para consultar centenas de gigabytes de dados.
2) RDDs são uma abstração de memória distribuída que permite que programadores realizem cálculos em memória em grandes clusters de forma tolerante a falhas. RDDs podem expressar uma ampla gama de aplicações paralelas, incluindo muitos modelos de programação especializados que foram propostos para computação interativa, e novas aplicações que esses modelos não capturam.
4) RDD/Spark pode ser usado para uma ampla gama de aplicações paralelas, incluindo muitos modelos de programação especializados que foram propostos para computação interativa, e novas aplicações que esses modelos não

MapReduce: A major step backwards

O artigo "MapReduce: A major step backwards" discute as limitações do framework MapReduce para processamento de dados em larga escala. Os autores, David DeWitt e Michael Stonebraker, argumentam que o MapReduce é uma abordagem subótima para processamento de dados em larga escala, pois usa força bruta em vez de indexação e não oferece muitos recursos que são rotineiramente incluídos em sistemas de gerenciamento de banco de dados (DBMS). Eles também afirmam que o MapReduce não é uma ideia nova, mas sim uma implementação específica de técnicas conhecidas há quase 25 anos. Os autores concluem que, embora o MapReduce possa ser uma boa ideia para escrever certos tipos de cálculos de propósito geral, ele não representa uma mudança de paradigma no desenvolvimento de aplicativos de dados intensivos em escala. Em vez disso, eles sugerem que outras abordagens, como o processamento em memória do Spark, podem ser mais eficientes e flexíveis para processamento de dados em larga escala.

3) Trocar MapReduce por Spark é uma escolha comum devido a várias vantagens que o Spark oferece em relação ao MapReduce. Aqui estão algumas razões pelas quais muitas organizações optam por usar o Apache Spark em vez do Apache MapReduce:

- **Desempenho:**
  O Spark é conhecido por ser significativamente mais rápido do que o MapReduce. Isso ocorre porque o Spark utiliza a execução em memória para processamento de dados, minimizando a necessidade de leituras e gravações frequentes em disco, que eram uma característica do MapReduce. O processamento em memória do Spark é especialmente vantajoso para algoritmos iterativos e operações de machine learning.

- **Modelo de Programação mais Expressivo:**
  O Spark oferece APIs mais expressivas em comparação com o MapReduce, tornando o desenvolvimento de aplicações mais fácil e produtivo. Ele fornece APIs em Java, Scala, Python e R, o que permite que os desenvolvedores usem a linguagem com a qual estão mais familiarizados.

- **Suporte a Diversos Tipos de Workloads:**
  Além do processamento batch, o Spark suporta também streaming, SQL interativo e processamento de machine learning em um único framework, o que elimina a necessidade de usar diferentes sistemas para cada tipo de carga de trabalho.

- **Tolerância a Falhas Melhorada:**
  O Spark oferece melhor tolerância a falhas do que o MapReduce. Ele faz uso de informações lineage (linha de execução) para reconstruir dados perdidos em caso de falha de um nó.

- **Suporte a Dados em Lote e em Tempo Real:**
  Enquanto o MapReduce é principalmente voltado para processamento em lote, o Spark suporta processamento em tempo real, tornando-o mais versátil para uma variedade de cenários de processamento de dados.

- **Ecossistema Rico:**
  O ecossistema Spark é mais abrangente e integrado do que o ecossistema MapReduce. Ele inclui módulos para SQL (Spark SQL), machine learning (MLlib), processamento de streams (Spark Streaming) e graph processing (GraphX), tornando-o uma escolha mais completa para diversas aplicações.

- **Integração com Hadoop e Outras Fontes de Dados:**
  O Spark é compatível com o Hadoop Distributed File System (HDFS) e pode ler dados diretamente do Hadoop, além de suportar várias fontes de dados, como bancos de dados JDBC, Apache Cassandra, Apache HBase, entre outros.

- **Economia de Recursos:**
  O Spark é mais eficiente em termos de uso de recursos, o que significa que pode realizar tarefas equivalentes com menos hardware em comparação com o MapReduce.