UNIPAC

UNIVERSIDADE PRESIDENTE ANTÔNIO CARLOS

CIÊNCIA DA COMPUTAÇÃO

Hadoop vs Spark

Lucas Santos

Matheus Barbosa

Rafael Sidnei

Trabalho teórico apresentado à disciplina de Banco de Dados do curso de Ciência da Computação da Universidade Presidente Antônio Carlos.

Barbacena

Dezembro de 2018

Sumário

1 Introdução

A história da computação é marcada por períodos de inovação disruptiva que muda completamente o cenário da tecnologia.E os gerenciadores de banco de dados pareciam imunes a mudanças de paradigma. Mas desde 2004, o cenário começou a mudar e agora a mudança parece inevitável. A prova disso, é que os grandes players do mercado, Oracle, IBM e Microsoft, estão revendo seus produtos, que juntos representam mais de 90% do mercado de bancos de dados.

Para empresas que estão selecionando produtos de banco de dados para um tipo específico de aplicativo, nosso conselho é determinar qual categoria de banco de dados eles precisam antes de pensar em quais produtos investigar. Embora, com o passar do tempo, possamos esperar que haja alguma racionalização entre essas categorias de bancos de dados, esperamos que a maioria deles persista com dois ou três produtos dominando cada categoria. Isso ocorre porque as categorias foram derivadas com base em diferentes tipos de carga de trabalho, e não esperamos que um mecanismo de banco de dados que seja excelente em uma dessas categorias tenha um desempenho particularmente bom em outras categorias.

Sistemas Gerenciadores de Bancos de dados são usados em muitas aplicações, enquanto atravessando virtualmente a gama inteira de software de computador. Os Sistemas Gerenciadores de Bancos de dados são o método preferido de armazenamento/recuperação de dados/informações para aplicações multiusuárias grandes onde a coordenação entre muitos usuários é necessária. Até mesmo usuários individuais os acham conveniente, entretanto, muitos programas de correio eletrônico e organizadores pessoais estão baseados em tecnologia de banco de dados standard.

2 Hadoop versus Spark

Em primeiro lugar, é importante notar que o Hadoop e Spark são tecnologias diferentes, com diferentes casos de uso. Berço das duas tecnologias, a própria Apache Software Foundation as aloca em categorias diferentes: Hadoop é um banco de dados, o Spark é uma ferramenta de Big Data.

A cada ano, parece haver cada vez mais sistemas distribuídos no mercado para gerenciar o volume, a variedade e a velocidade dos dados. Entre esses sistemas, o Hadoop e o Spark são os dois que continuam a obter o maior número de mentes.

O Spark tem melhor desempenho que o Hadoop quando:

- o tamanho dos dados varia de GBs a PBs;
- há uma complexidade algorítmica variada, do ETL ao SQL ao aprendizado de máquina;
- trabalhos de fluxo de baixa latência para trabalhos em lote longos processamento de dados, independentemente do meio de armazenamento, seja discos, SSDs ou memória;
- Além destes, o Hadoop ultrapassa o Spark.

Por exemplo, quando o tamanho dos dados é pequeno (100 MB). Quando os dados são classificados, às vezes pode ser mais rápido ao executar o mapeamento nos nós de dados.

O Hadoop é usado para processamento em lote, enquanto o Spark pode ser usado para ambos. A esse respeito, os usuários do Hadoop podem processar usando tarefas MapReduce em que o processamento em lote é necessário. Em teoria, o Spark pode executar tudo o que o Hadoop pode e muito mais. Assim, torna-se uma questão de conforto quando se trata de escolher o Hadoop ou o Spark.

2.1 Hadoop

O Hadoop é um framework open source, escalável e tolerante a falhas escrito em Java. Ele processa com eficiência grandes volumes de dados em um cluster de hardware de commodity. O Hadoop não é apenas um sistema de armazenamento, mas também uma plataforma para armazenamento de dados e processamento de dados.

O Hadoop começou como um projeto do Yahoo em 2006, tornando-se um projeto de código aberto do Apache de nível superior mais tarde. É uma forma geral de processamento distribuído que possui vários componentes: o Hadoop Distributed File System (HDFS), que armazena arquivos em um formato nativo do Hadoop e os paraleliza em um cluster; YARN, um cronograma que coordena os tempos de execução dos aplicativos; e MapReduce, o algoritmo que processa os dados em paralelo. O Hadoop é construído em Java e acessível através de muitas linguagens de programação, para escrever código MapReduce, incluindo Python, através de um cliente Thrift.

Além desses componentes básicos, o Hadoop também inclui o Sqoop, que move os dados relacionais para o HDFS; Hive, uma interface semelhante a SQL que permite aos usuários executar consultas no HDFS; e Mahout, para aprendizado de máquina. Além de usar o HDFS para armazenamento de arquivos, o Hadoop também pode agora ser configurado para usar buckets do S3 ou blobs do Azure como entrada.

2.2 Spark

O Spark é um projeto mais recente, desenvolvido inicialmente em 2012, no AMPLab, na UC Berkeley. É também um projeto Apache de nível superior focado no processamento de dados em paralelo em um cluster, mas a maior diferença é que ele funciona na memória.

Enquanto o Hadoop lê e grava arquivos no HDFS, o Spark processa dados na RAM usando um conceito conhecido como RDD, Conjunto de Dados Distribuídos Resiliente. O Spark pode ser executado no modo autônomo, com um cluster do Hadoop servindo como fonte de dados ou em conjunto com o Mesos. No segundo cenário, o mestre Mesos substitui o mestre Spark ou o YARN para fins de agendamento.

O Spark é estruturado em torno do Spark Core, o mecanismo que impulsiona o agendamento, as otimizações e a abstração do RDD, além de conectar o Spark ao sistema de arquivos correto (HDFS, S3, RDBMs ou Elasticsearch). Existem várias bibliotecas que operam sobre o Spark Core, incluindo o Spark SQL, que permite executar comandos semelhantes a SQL em conjuntos de dados distribuídos, MLLib para aprendizado de máquina,

GraphX para problemas de gráficos e streaming, que permite a entrada contínua de fluxo dados de registro.

O Spark tem várias APIs. A interface original foi escrita em Scala e, com base no uso pesado por cientistas de dados, os pontos finais de Python e R também foram adicionados. Java é outra opção para gravar trabalhos do Spark.

3 Diferenças

3.1 Arquitetura

Hadoop

Para começar, todos os arquivos passados para o HDFS são divididos em blocos. Cada bloco é replicado um número especificado de vezes no cluster com base em um tamanho de bloco configurado e em um fator de replicação. Essa informação é passada para o NameNode, que controla tudo através do cluster. O NameNode atribui os arquivos a um número de nós de dados no qual eles são gravados. A alta disponibilidade foi implementada em 2012 , permitindo que o NameNode faça failover em um nó de backup para controlar todos os arquivos em um cluster.

O algoritmo MapReduce fica no topo do HDFS e consiste em um JobTracker. Depois que um aplicativo é gravado em um dos idiomas, o Hadoop aceita o JobTracker, o seleciona e aloca o trabalho (que pode incluir desde contar palavras e limpar arquivos de log até executar uma consulta HiveQL sobre os dados armazenados no warehouse Hive) para TaskTrackers ouvindo em outros nós.

O YARN aloca recursos que o JobTracker gera e monitora, movendo os processos para maior eficiência. Todos os resultados do estágio MapReduce são então agregados e gravados de volta no disco no HDFS.

Spark

As alças do Spark funcionam de maneira semelhante ao Hadoop, exceto pelo fato de que as computações são realizadas na memória e armazenadas lá, até que o usuário as persista ativamente. Inicialmente, o Spark lê de um arquivo no HDFS, S3 ou outro diretório, em um mecanismo estabelecido chamado SparkContext. Fora desse contexto, o Spark cria uma estrutura chamada RDD, ou Conjunto de Dados Distribuído Resiliente, que representa uma coleção imutável de elementos que podem ser operados em paralelo.

Conforme o RDD e as ações relacionadas estão sendo criados, o Spark também cria um DAG, ou Directed Acyclic Graph, para visualizar a ordem das operações e o relacionamento entre as operações no DAG. Cada DAG tem etapas e etapas. Dessa maneira, é semelhante a um plano de explicação no SQL.

Você pode executar transformações, etapas intermediárias, ações ou etapas finais em RDDs. O resultado de uma determinada transformação entra no DAG, mas não persiste no disco, mas o resultado de uma ação persiste todos os dados na memória para o disco.

Uma nova abstração no Spark é o DataFrames, que foi desenvolvido no Spark 2.0 como uma interface complementar aos RDDs. Os dois são extremamente semelhantes, mas os DataFrames organizam os dados em colunas nomeadas, semelhantes aos pandas ou pacotes R do Python. Isso os torna mais fáceis de usar do que os RDDs, que não

têm um conjunto semelhante de referências de cabeçalho no nível da coluna. O SparkSQL também permite que os usuários consultem DataFrames de maneira semelhante às tabelas SQL em armazenamentos de dados relacionais.

3.2 Custos

O Spark e o Hadoop estão disponíveis gratuitamente como projetos Apache de código aberto, o que significa que você poderia executá-lo com custos de instalação zero. No entanto, é importante considerar o custo total de propriedade, que inclui manutenção, compras de hardware e software e a contratação de uma equipe que entenda a administração de cluster. A regra geral para instalações no local é que o Hadoop requer mais memória no disco e o Spark requer mais RAM, o que significa que a configuração de clusters do Spark pode ser mais cara. Além disso, como o Spark é o sistema mais novo, os especialistas nele são mais raros e mais caros. Outra opção é instalar usando um fornecedor como Cloudera for Hadoop ou Spark for DataBricks, ou executar processos EMR / MapReduce na nuvem com a AWS.

As comparações de preços de extração podem ser complicadas para serem divididas, pois o Hadoop e o Spark são executados em conjunto, mesmo em instâncias do EMR, que são configuradas para serem executadas com o Spark instalado. Para um ponto de comparação de alto nível, supondo que você escolha um cluster EMR otimizado para computação para o Hadoop, o custo para a instância mais pequena, c4.large, é de US $0,026 por hora. O menor cluster otimizado para memória do Spark custaria US $0,067 por hora. Portanto, em uma base por hora, o Spark é mais caro, mas otimizado para o tempo de computação, tarefas semelhantes devem levar menos tempo em um cluster do Spark.

3.3 Tolerância e Falha de Segurança

O Hadoop é altamente tolerante a falhas porque foi projetado para replicar dados em muitos nós. Cada arquivo é dividido em blocos e replicado inúmeras vezes em várias máquinas, garantindo que, se uma única máquina ficar inativa, o arquivo possa ser reconstruído a partir de outros blocos em outro lugar.

A tolerância a falhas do Spark é obtida principalmente por meio de operações de RDD. Inicialmente, o data-at-rest é armazenado no HDFS, que é tolerante a falhas por meio da arquitetura do Hadoop. Como um RDD é construído, também é uma linhagem, que lembra como o conjunto de dados foi construído e, como é imutável, pode reconstruí-lo do zero, se necessário. Os dados nas partições do Spark também podem ser recriados nos nós de dados com base no DAG. Os dados são replicados nos nós do executor e geralmente podem ser corrompidos se o nó ou a comunicação entre executores e drivers falhar.

O Spark e o Hadoop têm acesso ao suporte à autenticação Kerberos , mas o Hadoop tem controles de segurança mais refinados para o HDFS. O Apache Sentry, um sistema para impor acesso a metadados de baixa granularidade, é outro projeto disponível especificamente para segurança no nível do HDFS.

O modelo de segurança do Spark é atualmente escasso, mas permite autenticação via segredo compartilhado.

4 Semelhanças

Os componentes do Hadoop podem ser usados juntamente com o Spark das seguintes maneiras:

- **HDFS**: O Spark pode ser executado sobre o HDFS para aproveitar o armazenamento replicado distribuído.
- **MapReduce**: O Spark pode ser usado junto com o MapReduce no mesmo cluster do Hadoop ou separadamente como uma estrutura de processamento.
- **YARN**: Os aplicativos Spark podem ser executados no YARN (Hadoop NextGen).
- **Processamento em lote e em tempo real**: MapReduce e Spark são usados juntos, onde MapReduce é usado para processamento em lote e Spark para processamento em tempo real.

Para diferentes configurações experimentais, descobrimos que, embora O Spark é geralmente mais rápido que o Hadoop, é ao custo de consumo significativo de memória. Se a velocidade não é uma exigência e não temos memória abundante, é melhor Não escolha Spark. Neste caso, desde que tenhamos disco suficiente espaço para acomodar o conjunto de dados original e intermediário resultados, o Hadoop é uma boa escolha.

Para uma operação iterativa específica, se a aplicação for hora sensível, Spark deve ser considerado. Mas memória suficiente é uma condição necessária para a operação, a fim de garantir Vantagem de desempenho do Spark. O problema é que é difícil para determinar a quantidade exata de memória para iterativo operações em execução no Spark. Exatamente quanta memória é o suficiente depende do algoritmo iterativo particular e do tamanho do conjunto de dados que processa. Resultados intermediários durante as iterações são armazenadas na memória como RDDs. Por causa do readonly natureza do RDD, os novos RDD serão sempre criados em cada iteração. Se o tamanho dos resultados intermediários é constante nível de memória, o aumento do uso de memória entre dois as iterações não são significativas. Se o tamanho do intermediário os resultados são proporcionais ao tamanho do conjunto de dados de entrada (como PageRank), o aumento do uso de memória entre duas iterações consecutivas é significativo.

Referências

[1] Lei Gu and Huan Li. Memory or time: Performance evaluation for iterative operation on hadoop and spark. In *High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on*, pages 721–727. IEEE, 2013.

[2] Afshan K. What is the difference between hadoop and spark?, 2017.

[3] Amir K. How do hadoop and spark stack up?, 2018.

[1] [3] [2]