

Complementos de Base de Dados

Resumos
2016/2017

João Alegria | 68661

Capítulo 1

Hadoop / HDFS

Hadoop

O que é o Hadoop?

O Hadoop é um software *open-source* desenvolvido pela a Apache, com alta escalabilidade, confiável, projectado para computação distribuída sobre o paradigma de programação map-reduce.

- **É útil em:**

- Processamento distribuído de grandes conjuntos de dados em clusters
(um cluster - ou agregado de computadores - é formado por um conjunto de computadores, que utiliza um tipo especial de sistema operativo classificado como sistema distribuído. Muitas vezes é construído a partir de computadores convencionais que estão ligados em rede e comunicam através do sistema, trabalhando como se fossem uma máquina única de grande parte)
- “scale-up” de um para milhares de servidores

- **Quem utiliza o Hadoop atualmente e seus exemplos de utilização:**

- Adobe
 - Facebook
 - Yahoo
 - Amazon
 - eBay
 - Spotify
- Os exemplos de uso do Hadoop são analisar padrões dos utilizadores em sites de e-commerce (vendas online) e sugerir novos produtos para que eles possam comprar.
- O Hadoop simplificou o processo paralelo uma vez que o desenvolvedor não precisa de se preocupar com problemas relativos ao processamento paralelo.
- Escreve apenas as funções de como quer que os dados sejam processados.

- **Quando se deve usar o Hadoop:**

- Em ficheiros muito grandes - ficheiros “muito grandes” neste contexto são arquivos com centenas de megas, gigas ou terabytes de tamanho.
- Transmissão de acesso de dados - o HDFS foi construído em torno da ideia de que o padrão de processamento de dados mais eficiente é a “*write-once*” - escrita única - e “*read-many-times pattern*” - padrão de várias leituras.
- Cada análise envolve uma grande proporção, se não toda, dos conjuntos de dados, portanto o tempo de ler todo o conjunto de dados é mais importante que a latência na leitura do primeiro registo (*record*).

- **Componentes do Hadoop**

- O framework do Hadoop é formado por dois componentes principais:
armazenamento e processamento

O primeiro é o HDFS (Hadoop Distributed File System), que manipula o armazenamento de dados entre todas as máquinas na qual o cluster do Hadoop está a ser executado.

O segundo, o Map-Reduce, manipula a parte do processamento da framework.

HDFS - Hadoop Distributed File System

O HDFS é um sistema de ficheiros escalável e distribuído, cuja arquitetura é fortemente baseada na GFS (Google File System), que também é um sistema de ficheiros distribuídos. Os sistemas de ficheiros distribuídos são assim necessários, uma vez que os dados tornam-se demasiados grandes para serem armazenados numa máquina só.

O HDFS armazena todos os ficheiros em blocos. O tamanho do bloco padrão é 64MB. Todos os ficheiros no HDFS possuem múltiplas réplicas, o que auxilia o processamento paralelo.

- **Os clusters HDFS possuem dois tipos de nós:**

- **Namenode** - Administra o namespace do sistema de ficheiros. Ele gere todos os ficheiros e diretórios. Namenodes possuem o mapeamento de ficheiros e os blocos nos quais estão armazenados. Todos os arquivos são acedidos usando esses namenodes e datanodes.

- **Namenodes secundários** - Este node é responsável por verificar a informação do namenode. Em caso de falha, podemos usar esse nó para reiniciar o sistema.


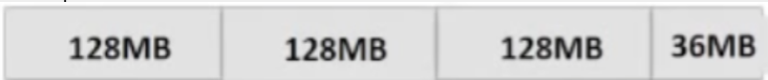
- **Datanode** - Armazena os dados em forma de blocos. Os datanodes comunicam aos namenodes sobre os ficheiros que possuem armazenados para que o namenode esteja ciente e os dados possam ser processados.

➡ Namenode é talvez o principal ponto crucial de falha no sistema.

Características do HDFS

- Sistema de ficheiros distribuídos
- Alta performance
- Alta disponibilidade
- Confiança
- HDFS corre em cima do sistema de ficheiros existente
- Projectado para tratar arquivos muito grandes, como padrões de acesso de dados de streaming
- HDFS utiliza blocos para armazenar ficheiros ou parte deles

Blocos

Ficheiros Blocos:	<ul style="list-style-type: none">- Tamanho: 64MB (padrão), 128MB (recomendado)- 1 bloco HDFS é suportado por múltiplos "OS blocks"  <p>The diagram illustrates that one HDFS Block, labeled '128 MB', is composed of several smaller 'OS Blocks'. A dashed line separates the HDFS Block from the individual OS Blocks below it.</p>
Vantagens dos Blocos:	<ul style="list-style-type: none">- Tamanho fixo: Fácil de calcular quantos são armazenados num disco- Um ficheiro pode ser maior do que um disco- Se um ficheiro, ou parte dele, é menor que o tamanho do bloco, apenas o espaço necessário é utilizado. <p>Exemplo da divisão de um ficheiro com 420MB:</p>  <p>The diagram shows a horizontal bar representing a 420MB file, divided into four segments. The first three segments are labeled '128MB' and the last segment is labeled '36MB'.</p>

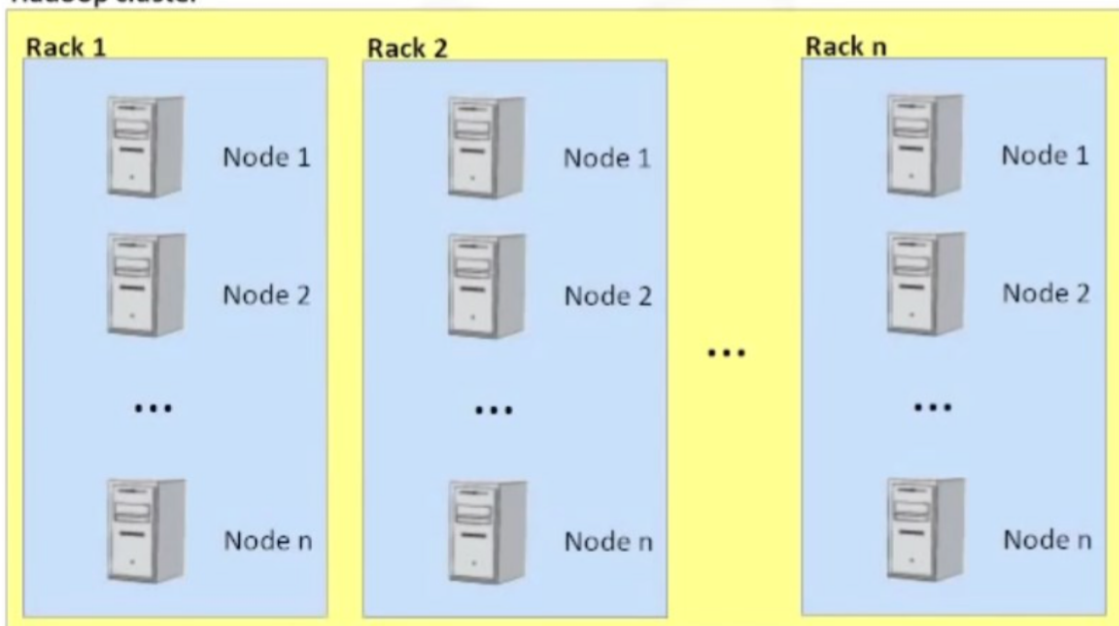
Vantagens do HDFS

- **Escalável**
 - apenas é necessário adicionar nós
 - evita “network bottleneck”
- **Flexível**
 - todos os tipos de dados (documentos, registos, etc)
 - em todos os formatos (estruturado, semi-estruturado, não estruturado)
 - armazena qualquer informação
- **Eficiente**
 - eficiência de custo (< \$1000 por Terabyte)

Principais Entidades

Node	Rack	Hadoop Cluster
<ul style="list-style-type: none">- Namenode (NN) - nó mestre (único)- Datanode (DN) - nó escravo (vários)	<ul style="list-style-type: none">- Coleção de nodes conetados ao mesmo switch de rede- Pode existir mais que um rack- largura de banda entre racks > largura de banda entre nodes	<ul style="list-style-type: none">- Coleção de racks

Hadoop cluster



Map-Reduce

Map-Reduce é um paradigma de programação introduzido pelo Google para processar e analisar grandes conjuntos de dados

Neste paradigma, cada tarefa é especificada em termos de funções de mapeamento e redução. Ambas as tarefas rodam paralelamente no cluster. O armazenamento necessário para essa funcionalidade é fornecido pelo HDFS.

A razão para a escalabilidade desse paradigma é a sua natureza distribuída do funcionamento da solução. Uma grande tarefa é dividida em várias tarefas pequenas que são então executadas em paralelo em máquinas diferentes e então combinadas para solução que deu início.

Principais Componentes do Map-Reduce

• *Job Tracker Node*

- Todas as tarefas de Map-Reduce são submetidas ao Job Tracker.
- O Job Tracker precisa de comunicar com o Namenode para conseguir os dados.
- O Job Tracker submete a tarefa para os nós Task Trackers.
- Esses Task Trackers precisam reportar-se ao Job Tracker em intervalos regulares, dizendo que estão “vivos” e a efetuar as suas tarefas.
- Caso o Task Tracker não se reporte, então o nó é considerado “morto” e o seu trabalho é direcionado para outro Task Tracker.
- O Job Tracker é um ponto crucial de falhas. Se o Job Tracker falhar, não é possível rastrear as tarefas.

• *Task Tracker Node*

- O Task Tracker aceita as tarefas do Job Tracker. Essas tarefas são tanto de map, reduce ou ambas (*shuffle*).
- O Task Tracker cria um processo JVM separado para cada tarefa a fim de se verificar que uma falha no processo não resulta numa falha de Task Tracker.
- Task Trackers também reportam-se ao Job Tracker continuamente para que este possa manter os registos de tarefa bem ou mal sucedidos.

Vantagens	Limitações
<ul style="list-style-type: none">• Barato<ul style="list-style-type: none">- Dimensiona até Petabytes, ou mais...• Rápido<ul style="list-style-type: none">- Processamento paralelo de dados• Melhor<ul style="list-style-type: none">- Adequado para problemas de BigData	<ul style="list-style-type: none">• Escalável<ul style="list-style-type: none">- Tamanho máximo do cluster: 4000 nós- Tamanho máximo de tarefas em simultâneo: 40 000 nós- Sincronização bruta no Job Tracker• Namenode é o único ponto de falha<ul style="list-style-type: none">- A falha acaba com todos os trabalhos em execução na fila- Os trabalhos devem ser reenviados por utilizadores• Baixa utilização de recursos<ul style="list-style-type: none">- Partição de recursos nos slots de map-reduce• Não suporta outros paradigmas<ul style="list-style-type: none">- Suporta apenas map-reduce- Aplicações interativas que implementam o map-reduce são 10x mais lentas• Falta de protocolos compatíveis<ul style="list-style-type: none">- O cliente e o cluster devem ser da mesma versão

Yarn - Yet Another Resource Negotiator

O Yarn foi um grande avanço na versão 2 do Hadoop. Trouxe melhorias significativas de desempenho para algumas aplicações, oferece suporte a modelos de processamento adicional e implementa um mecanismo de execução flexível.

O Yarn é então um gestor de recursos que foi criado separando os mecanismos de processamento e recursos de gestão de map-reduce (que foi criado na versão 1 do Hadoop).

O Yarn é frequentemente chamado de sistema operativo do Hadoop porque é responsável pela gestão e monitoramento de cargas de trabalho, mantendo um ambiente de vários vizinhos, implementar controlos de segurança e gestão de recursos de alta disponibilidade do Hadoop.

Assim como um sistema operativo num servidor, o Yarn é projectado para permitir múltiplas e diferentes aplicações a correr numa plataforma de vários vizinhos.

No Hadoop 1, os utilizadores tinham a opção de escrever programas map-reduce em *Java*, *Python*, *Ruby* ou outras linguagens de script usando streaming ou usando o *Pig*, uma linguagem de transformação de dados. Independentemente do método que foi usado, esta fundamentalmente se baseou no modelo de processamento map-reduce para executar.

O Yarn suporta vários modelos de processamento para além do map-reduce. Um dos benefícios mais significativos é não estarmos limitados a nível de I/O, grande fonte de latência do map-reduce. Este avanço significa que os utilizadores de Hadoop devem estar familiarizados com os prós e contras dos novos modelos de processamento e entender quando aplicá-los aos casos de estudo particulares.

O Yarn executa vários tipos de trabalhos

- Processamento em tempo real
- ad-hoc OLAP
- map-reduce

O Yarn não é apenas um Hadoop 2.0 mas também:

- Framework para desenvolver e/ou executar aplicações de processamento distribuído

Requisitos atuais de Big Data

- Disponibilidade
- Confiança
- Utilização
- Compatibilidade “wire”
- Agilidade e evolução
 - Capacidade para os clientes controlarem as atualizações
- Escalabilidade
 - Clusters de 6000 a 10 000 máquinas