




Aula 1 - Hadoop

Lucio Monteiro



Hadoop

Apache Hadoop

O que é?

Hadoop é um **framework** em código aberto para armazenamento e processamento distribuído (computação distribuída) de grandes conjuntos de dados em hardware simples.





Motivado a construir um buscador complexo, que funcionasse na escala da web indexando bilhões de páginas, Doug Cutting resolveu se dedicar ao desafio iniciando seu projeto Nutch junto a Mike Cafarella, mas enfrentou alguns problemas com escalabilidade.

Um artigo publicado em 2003 pelo Google abriu caminho para que a equipe do Nutch criasse uma implementação open source do GFS (Google File System).

Em 2004 o Google publica o clássico artigo descrevendo seu framework MapReduce para atender às necessidades de processamento de várias máquinas das tarefas de rastreamento e índice.

Em 2006 o projeto Hadoop (NDFS + MapReduce) é criado (Hadoop era o nome do elefante amarelo de pelúcia do filho de Doug).

Em 2008 Hadoop se tornou um projeto independente dentro da Apache.

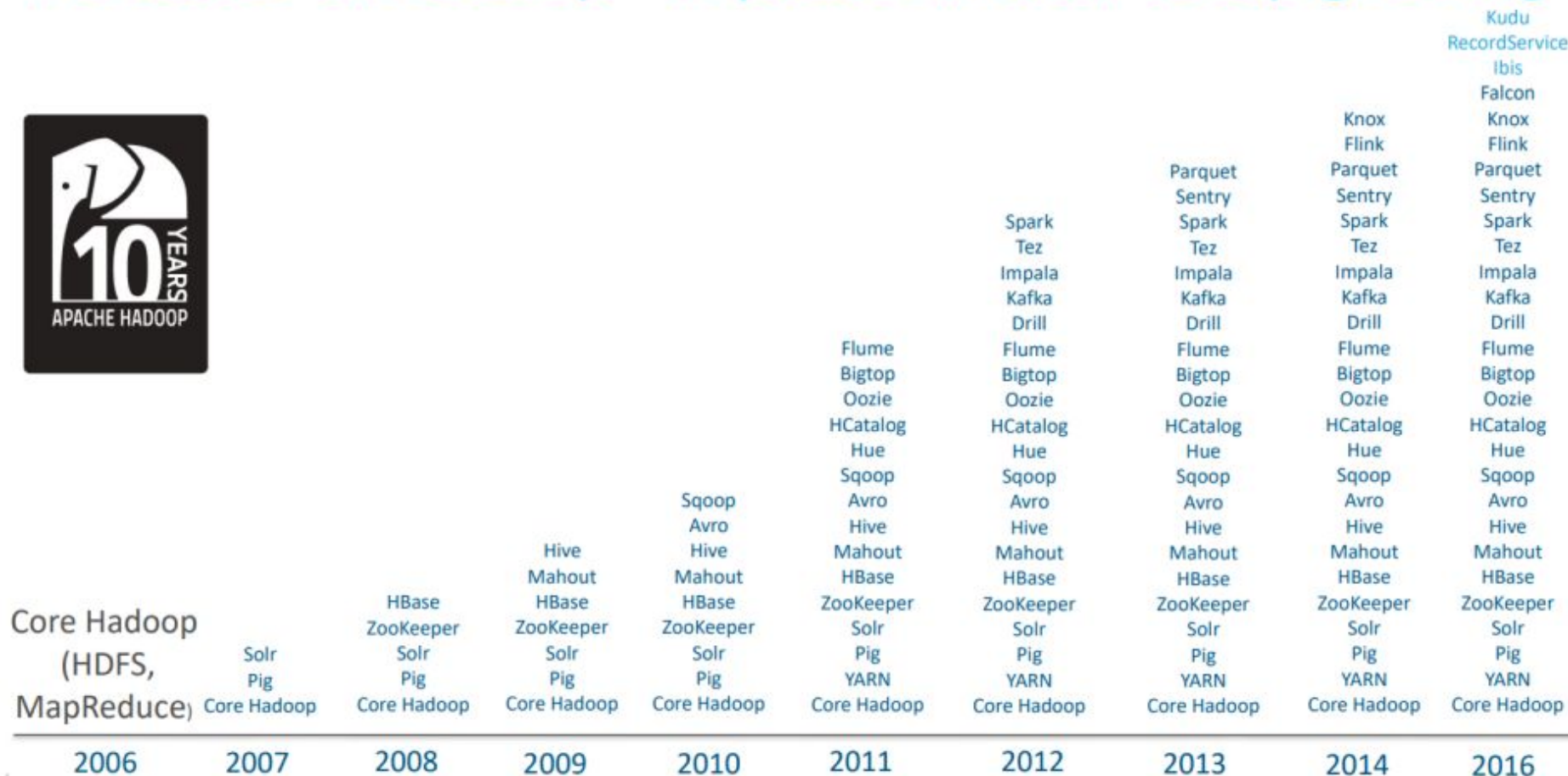
Em 2012 a primeira versão do Apache Hadoop foi disponibilizada.

Em 2013 a versão 2.2 já estava disponível.

Em 2018 o Hadoop 3.0 foi lançado.

Note como Hadoop foi desenvolvido de forma colaborativa, com um time dedicado ao projeto, mas fazendo uso de artigos e pesquisas de outras empresas.

A Decade of Hadoop – A platform won't stop growing

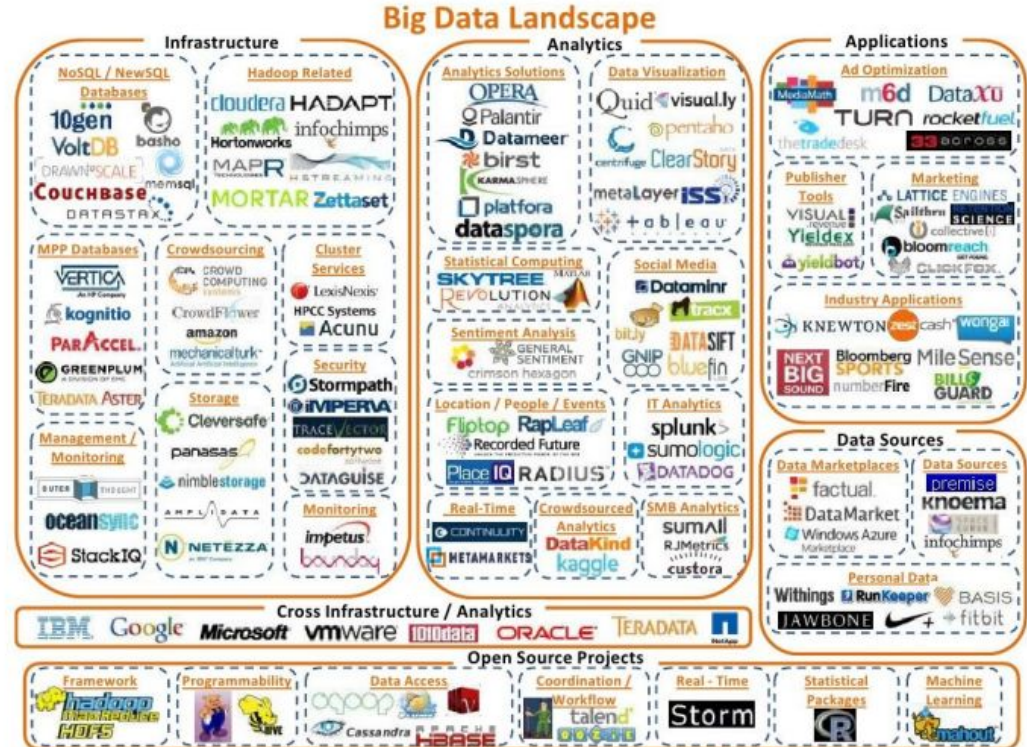
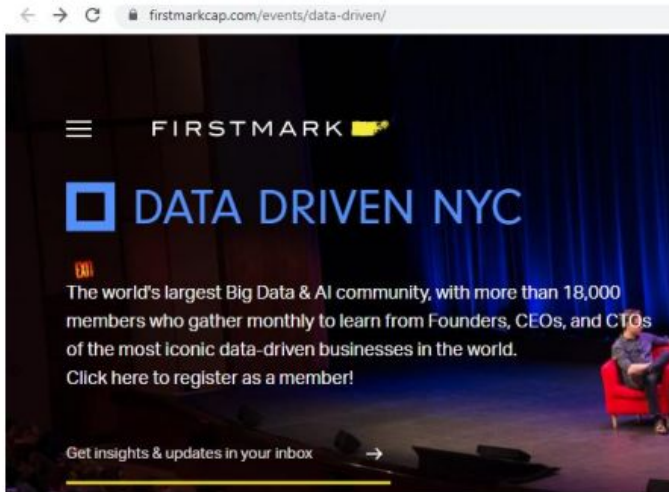


cloudera



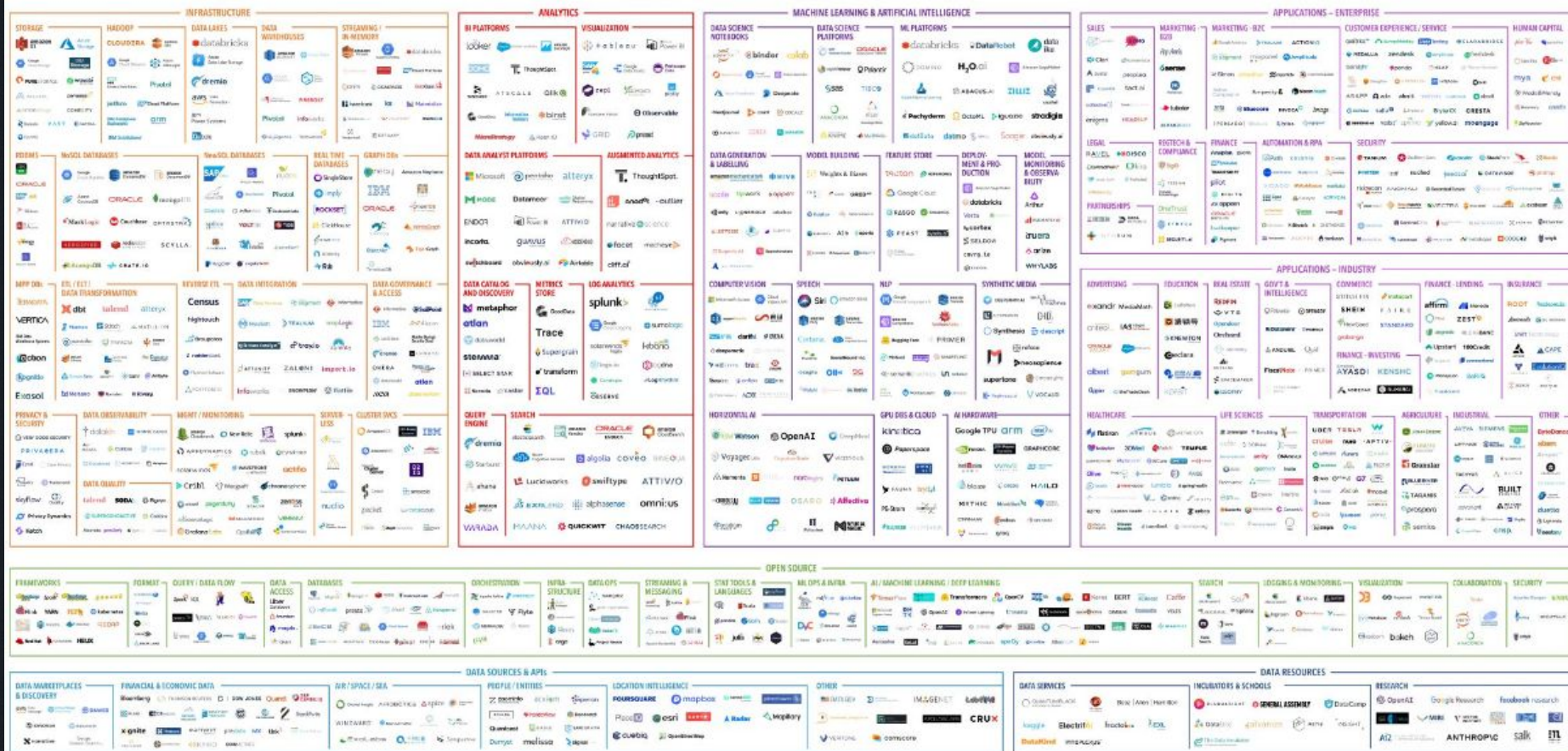
○ mattturck.com

○ Desde 2012



© Matt Turck (@mattturck) and ShivonZilis (@shivonz)

MACHINE LEARNING, ARTIFICIAL INTELLIGENCE, AND DATA (MAD) LANDSCAPE 2021



OPEN SOURCE

FRAMEWORKS



QUERY / DATA FLOW



DATA ACCESS & DATABASES



ORCHESTRATION & MGMT



STREAMING & MESSAGING



STAT TOOLS & LANGUAGES



AI OPS & INFRA



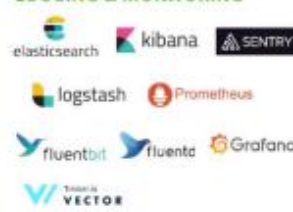
AI / MACHINE LEARNING / DEEP LEARNING



SEARCH



LOGGING & MONITORING



VISUALIZATION



COLLABORATION



SECURITY



HADOOP ON-PREMISE



HADOOP IN THE CLOUD



○ Outubro 2018 - Fusão da Cloudera e Hortonworks

○ Hortonworks - Gerenciamento de dados

○ Cloudera - Armazenamento de dados e aprendizado de máquina

○ Enterprise data cloud to AI

Onde Usar Hadoop?



- Análise de Dados
- Data Warehouse
- Data Lake
- Processamento de logs
- Muito mais!

Características?

- Baixo custo
- Flexibilidade de armazenamento
- OpenSource
- Tolerante a falha
- Permite complexas análises de dados
- Escalabilidade

Componentes?

- Hadoop Common
- HDFS (Hadoop File System)
- MapReduce
- Yarn



Em um **cluster Hadoop** os arquivos ali presentes são divididos em blocos (128MB) e estes blocos são replicados nos nós de acordo com o fator de replicação definido. Em outras palavras, se em um cenário hipotético um determinado cluster possuir fator de replicação três, cada bloco de arquivo terá três cópias espalhadas em diferentes nós do cluster (desde que o cluster tenha nós suficientes), garantindo então alta disponibilidade, pois se um ou dois nós ficarem inativos, nenhum dado será perdido. Portanto, quanto maior o fator de replicação, menores os riscos de indisponibilidade; entretanto, maior será o espaço em disco ocupado.



O Hadoop é baseado em uma arquitetura Master/Slave. Um cluster Hadoop possui um único nó Master e vários nós Slaves.

Master

É responsável por armazenar os metadados associados aos seus nós slaves no rack do qual faz parte.

O nó principal é responsável por manter o status de seus nós slaves, estabelecendo um deles como um nó passivo, que se tornará um nó principal se, por qualquer motivo, estiver bloqueado.

Slave

É o nó encarregado de armazenar e processar dados.

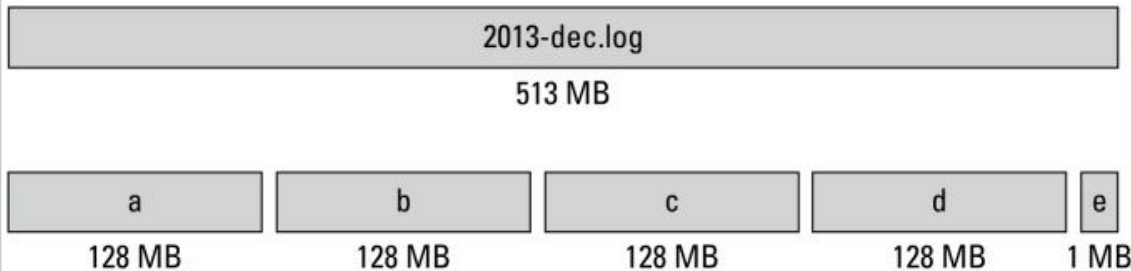


Hadoop Distributed File System (HDFS) é o sistema de armazenamento distribuído utilizado por aplicações Hadoop. O HDFS quebra os arquivos em blocos de dados (128 MB por padrão), cria réplicas (três por padrão) e as distribui no cluster, permitindo assim computações extremamente rápidas em arquivos pequenos e em máquinas distintas. HDFS permite escalabilidade e tolerância a falhas.

- Blocos de armazenamento

- Linux: 4KB

- Hadoop: 128MB



Hadoop For Dummies, 2014



NameNode Gerencia o namespace do sistema de arquivos do Hadoop.

O NameNode faz a gestão do HDFS em um nó: mantém metadados, logs, adiciona, encontra, exclui e copia arquivos.

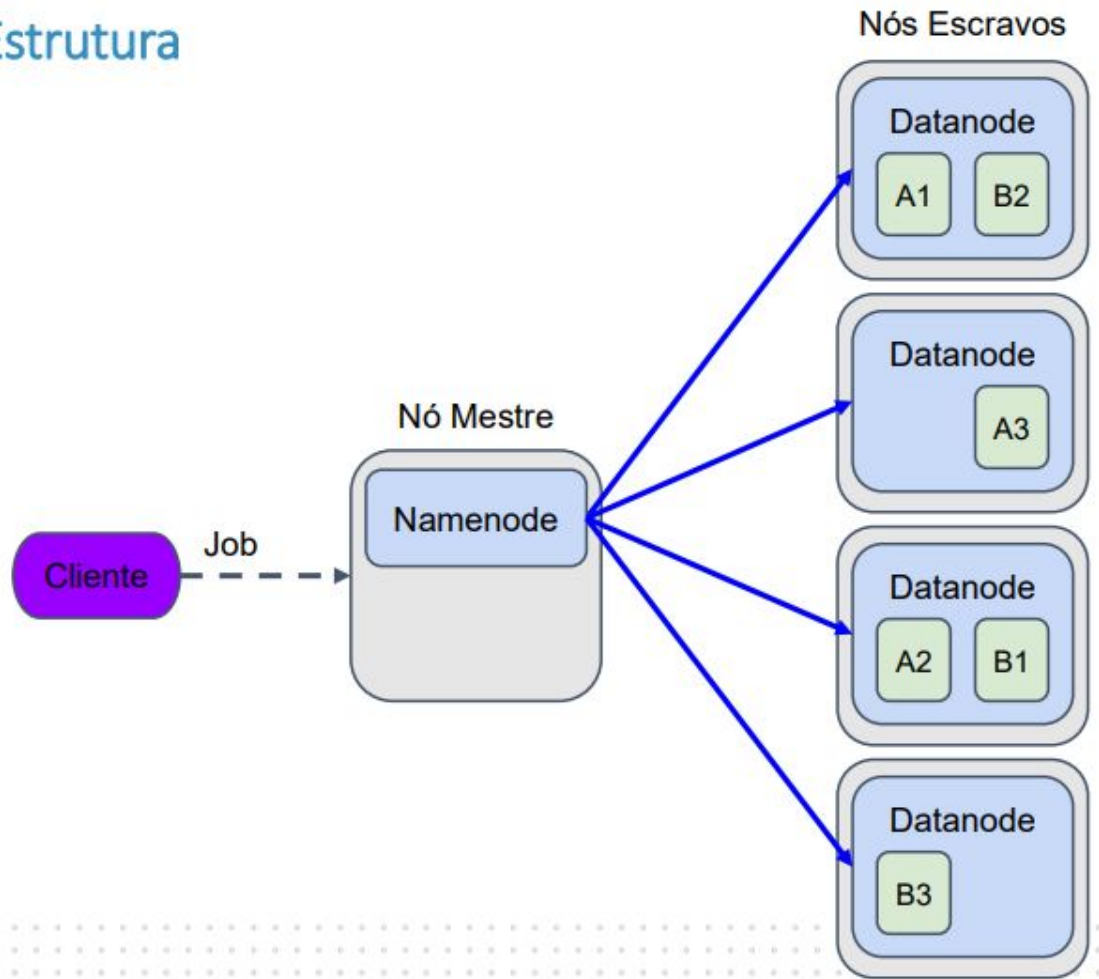
- Armazena metadados.
- Usa cache em RAM para acesso mais rápido ao metadado.
- Não armazena dados.
- Apenas 1 ativo por cluster.
- Ponto único de falha sem HA (Alta Disponibilidade).

DataNode Armazena os blocos de dados em um nó.

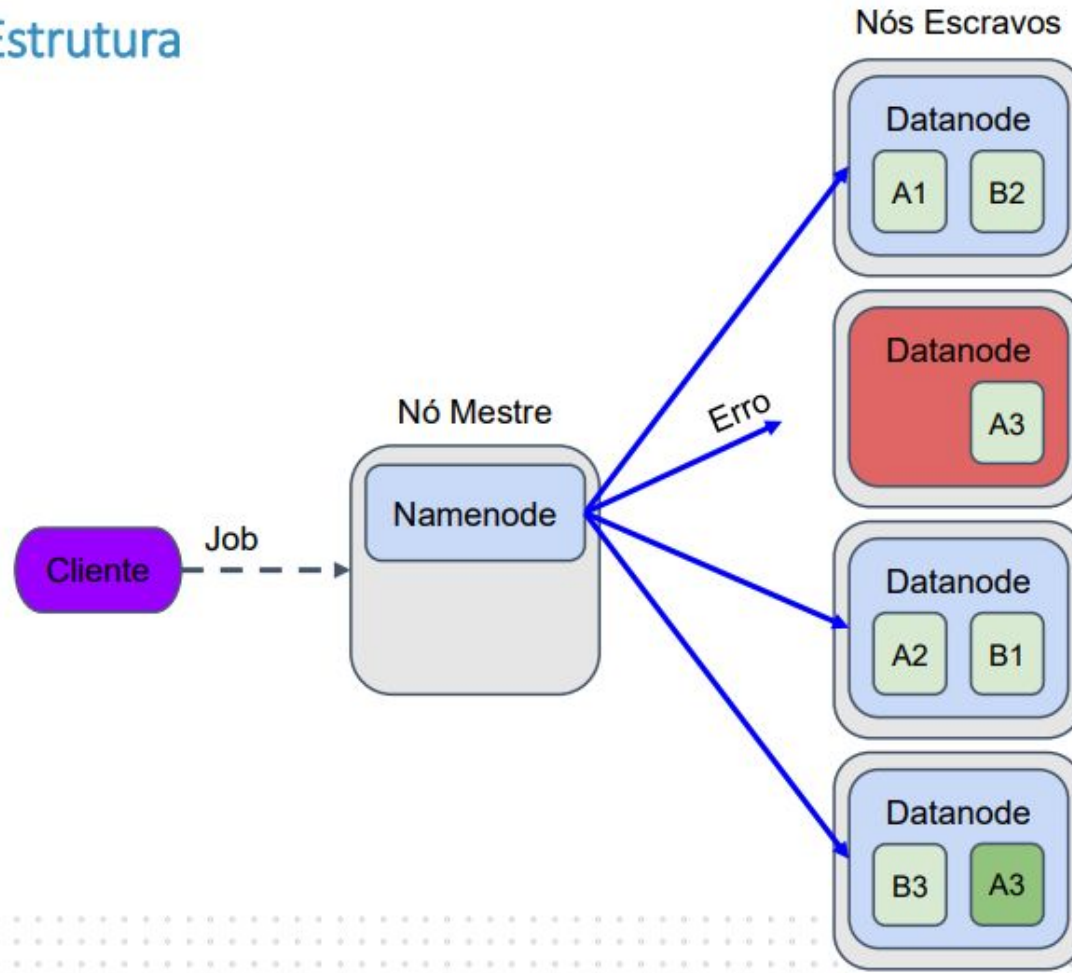
O DataNode mantém dados e replica blocos.

- Armazena os dados no HDFS.
- Atende solicitações de leitura e gravação dos clientes ou NameNode.
- Responsável por criar, excluir e replicar blocos de dados.
- Reportar status para o NameNode (heartbeat).
- Em caso de falta de report o nó é desativado pelo NameNode.
- Totalmente dependente do NameNode.

HDFS Estrutura



HDFS Estrutura





Comandos

- Criar um diretório no *HDFS*:

```
hdfs dfs -mkdir <caminho até o diretório no HDFS>
```

- Listar conteúdo de um diretório do *HDFS*:

```
hdfs dfs -ls <caminho até o diretório no HDFS>
```

- Inserir arquivo no HDFS a partir do File System:

```
hdfs dfs -put <caminho do arquivo no File System> <caminho do diretório no HDFS>
```

- Visualizar conteúdo de arquivos no *HDFS*:

```
hdfs dfs -cat <caminho até o arquivo no HDFS>
```

```
hdfs dfs -tail <caminho até o arquivo no HDFS>
```

- Mover arquivo dentro do *HDFS*:

```
hdfs dfs -mv <caminho até o arquivo no HDFS> <caminho do diretório destino no HDFS>
```

- Copiar arquivo dentro do *HDFS*:

```
hdfs dfs -cp <caminho até o arquivo no HDFS> <caminho do diretório destino no HDFS>
```



Comandos

- Remover arquivo do *HDFS*:

```
hdfs dfs -rm <caminho até o arquivo no HDFS>
```

- Copiar arquivo do *HDFS* para o *File System*:

```
hdfs dfs -get <caminho do arquivo no HDFS> <caminho do diretório no File System>
```

- Verifica a integridade do *HDFS*:

```
hdfs fsck <caminho do arquivo no HDFS> -blocks -files -locations
```

- Aumentar quantidade de réplicas:

```
hdfs dfs -setrep 2 <caminho do arquivo no HDFS>
```




O Hadoop **MapReduce** é um modelo de programação para criação de aplicações que processam rapidamente vastas quantidades de dados paralelamente através de grandes clusters de computadores comuns.

O código ou programa a ser executado é transportado até o local do dado, para que tarefas independentes sejam executadas em cada bloco de dado (Map), e depois sejam consolidadas gerando a resposta do processamento (Reduce).

O processo de forma simplificada:

- Dados são divididos em blocos.
- Divisão de problemas grandes e/ou complexos em pequenas tarefas.
- Mapeamento é executado em paralelo nos nós.
- Apenas quando o Mapeamento é encerrado, redução inicia, também em paralelo.
- Fase intermediária: Shuffle (distribui as saídas dos mappers para a execução do reducer).



Map

Atua exclusivamente sobre um conjunto de entrada com **chaves e valores**, gerando uma lista.

Características:

- Ponto de partida.
- Recebe cada registro dos dados de entrada como pares de chave/valor.
- Cada Mapper é independente um do outro, permitindo paralelismo e reexecuções de tarefas.
- Hadoop cria tarefas de Mapper para cada bloco de dados HDFS dos dados de entrada.
- Produz uma lista de chave/valor.

Reduce

Atua sobre os valores intermediários produzidos pelo map para, normalmente, agrupar os valores e produzir uma saída.

Características:

- Responsável por agregações, filtros e combinações diversas nos dados de entrada.
- Executa uma função de reduce por vez.
- Shuffle: Distribui as saídas dos mappers para a execução do reducer.
- Sort: Ordena os registros chave/valor, agrupando pela chave.
- Reduce: Envia os conjuntos chave/valor agrupados, filtrados ou combinados o formato de saída.



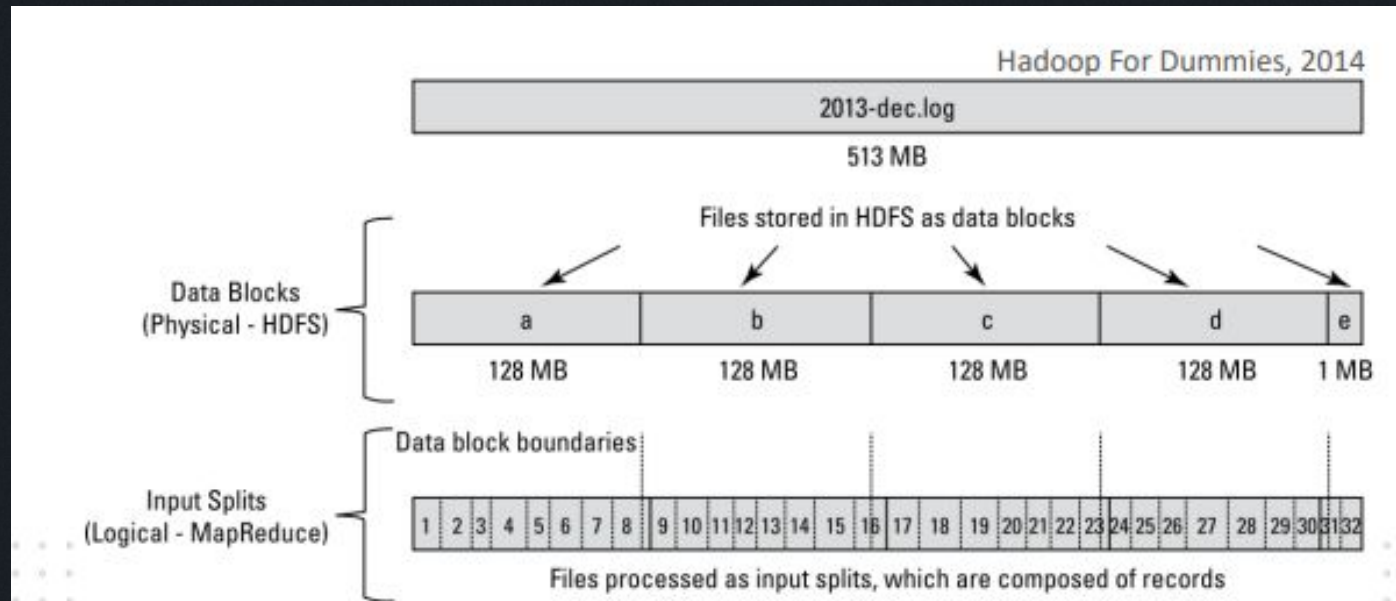
Pontos Positivos

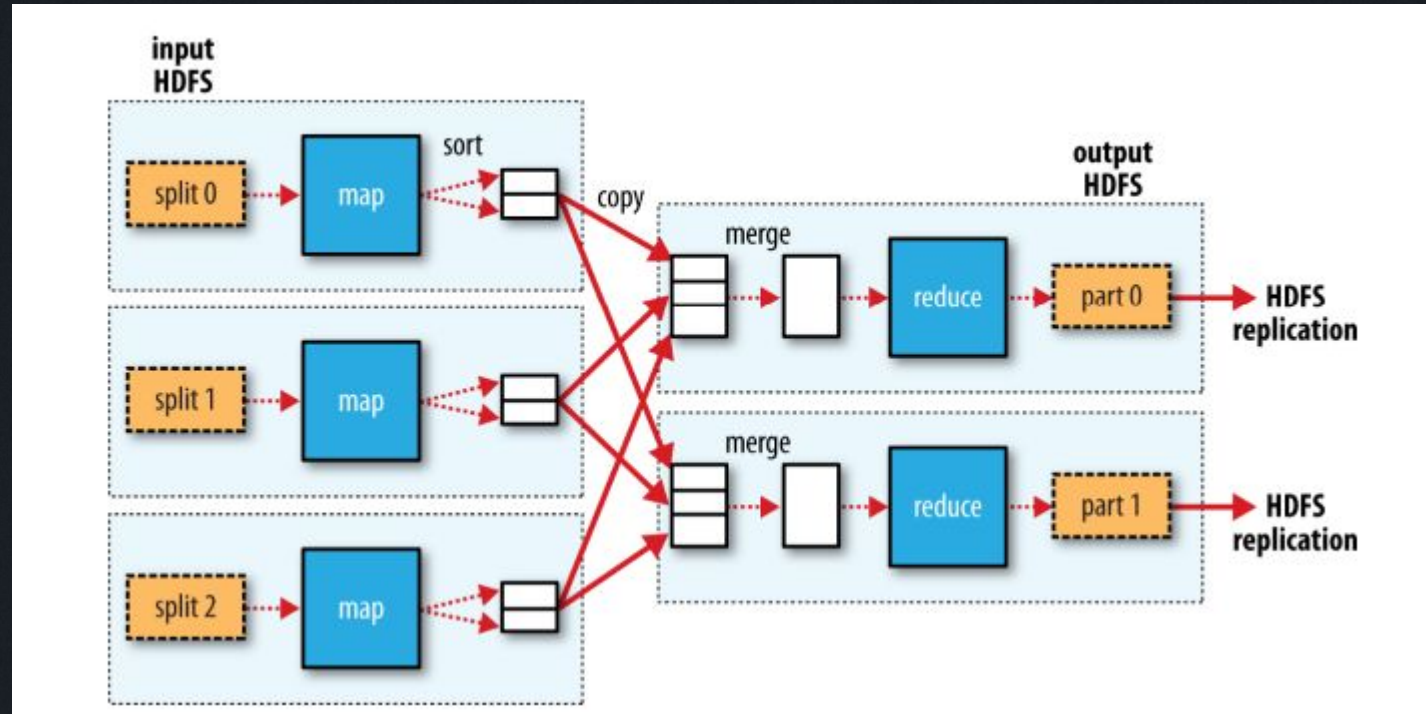
- Escalável.
- Tolerante a falhas.
- Disponibilidade.
- Confiável.
- Usa conceito de chave/valor.
- Não cria gargalos na rede pois os dados não trafegam (processamento no nó).

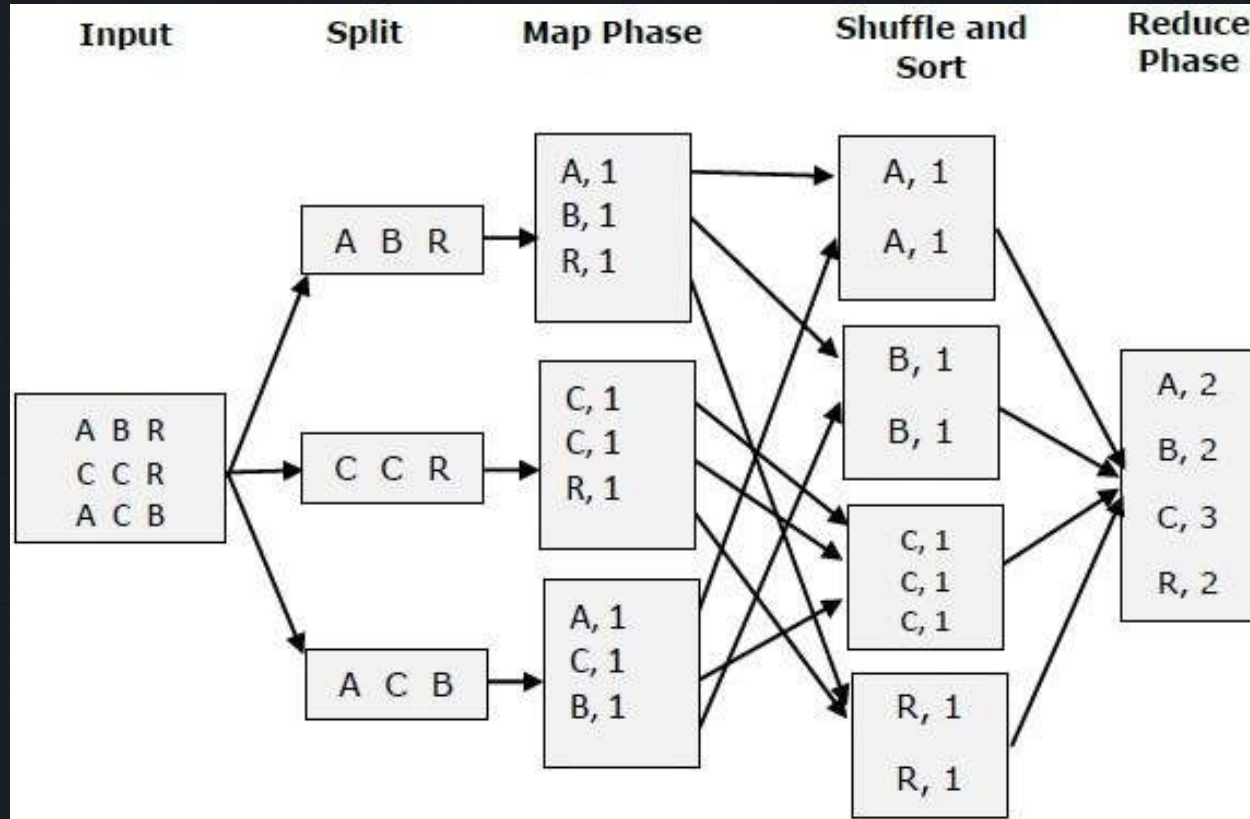
Pontos Negativos

MapReduce não é indicado para muitos casos, tais quais:

- Consultas que necessitam de baixa latência.
- Sistemas real-time.
- Consultas em um website.
- Processamento de pequenas tarefas.
- Overhead para gerenciamento das tarefas.









O **YARN** foi introduzido no Hadoop versão 2.0 no ano de 2012 pelo Yahoo e Hortonworks. A premissa por trás do **YARN** é aliviar o *MapReduce*, assumindo a responsabilidade de gerir recursos e de agendar tarefas. O **YARN** começou a dar ao *Hadoop* a capacidade de executar tarefas sem *MapReduce* na estrutura do *Hadoop*.

Características:

- Permite que vários aplicativos sejam executados simultaneamente no mesmo cluster compartilhado.
- Permite que os aplicativos negociem recursos com base na necessidade.

Arquitetura

Componentes:

- ResourceManager: um por cluster (orquestrador).
 - Application Manager: gerencia atividades, otimização, distribuição de recursos.
- NodeManager: um por nó, responsável pela execução dos Jobs.
- Application Master: distribui tarefas aos containers.
- Container: mantém as tarefas.



ResourceManager

- Possui um agendador de nível de cluster que tem responsabilidade pela alocação de recursos para todas as tarefas em execução, de acordo com as solicitações do *Application Manager*.
- A principal responsabilidade do *ResourceManager* é alocar recursos para os aplicativos.
- Não é responsável pelo rastreamento do status de uma aplicação ou tarefas de monitoramento.
- Não garante o reinício/balanceamento de tarefas no caso de falha no aplicativo ou no hardware.

NodeManager

- Nó Slave, é executado nos worker nodes.
- Gerencia o ciclo de vida do container e monitora o uso de recursos.
- Executa o container com base na capacidade do nó, que é calculada com base na memória instalada e no número de núcleos da CPU.
- Envia um sinal ao Resource Manager para atualizar seu status de integridade.
- Envia o status para ResourceManager, que pode ser o status do nó ou o status das tarefas executadas.



Application Master

- Biblioteca de aplicativos que gerencia cada instância de um aplicativo que é executado dentro de YARN.
- Responsável por negociar recursos do ResourceManager na submissão do aplicativo, como memória e CPU.
- Responsável por monitorar o status de um aplicativo e monitorar os processos de aplicativos em coordenação com o NodeManager.

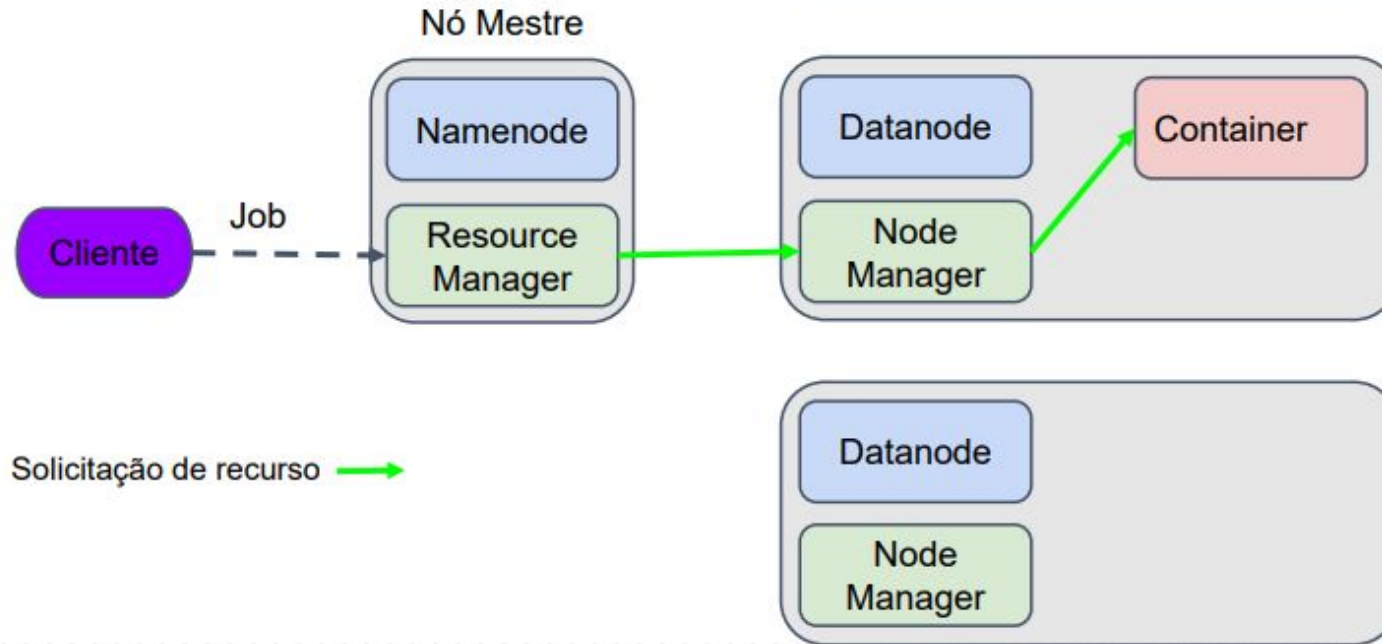
Container

- Pacote lógico de recursos em termos de memória, CPU, disco etc.
- Vinculado a um nó específico.
- ResourceManager aloca dinamicamente recursos como containeres.
- Um container concede direitos a um Application Master para usar uma quantidade específica de recursos de um host específico.
- Application Master é considerado como o primeiro container de um aplicativo e gerência a execução da lógica do aplicativo em containers alocados.

YARN Arquitetura

Enviar Job para RM

Resource Manager solicita um container para NM



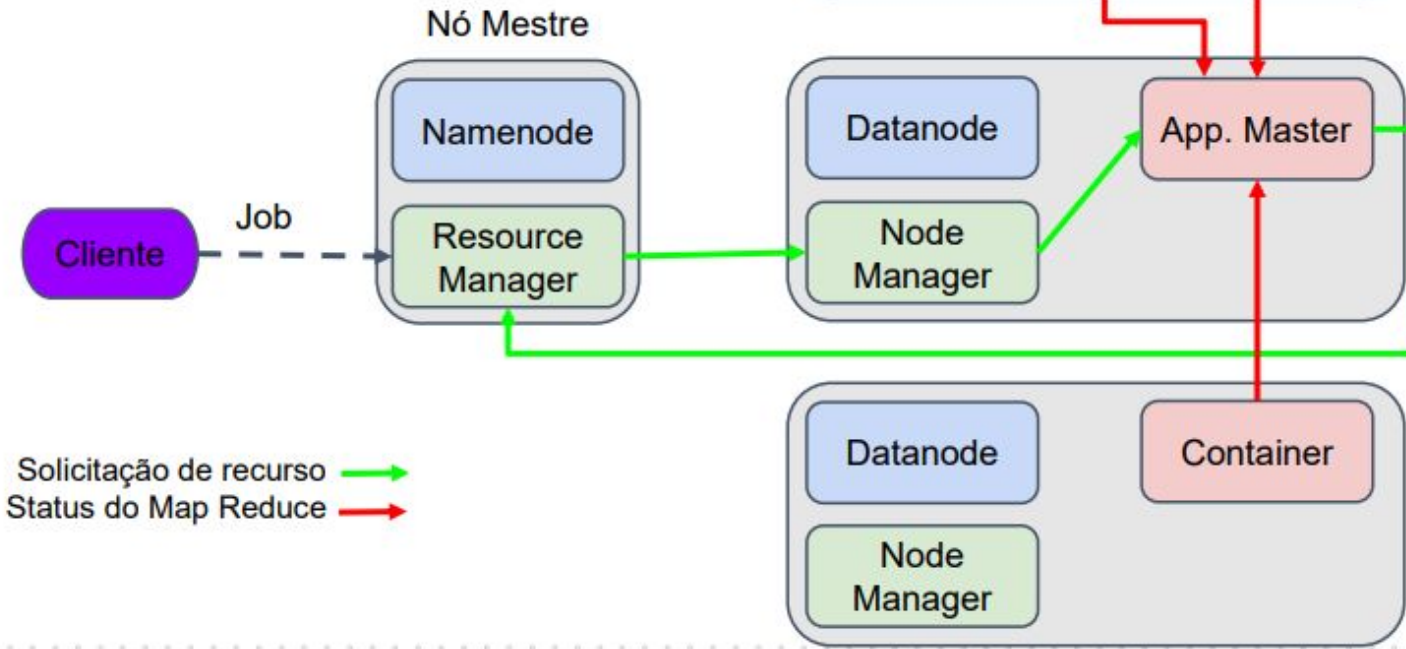
YARN Arquitetura

Enviar Job para RM

Resource Manager solicita um container para NM

App. Master começa a execução no container

App. Master solicita ao RM mais containers para Job





Comandos

Os comandos YARN são invocados usando o script `bin/yarn` no pacote *Hadoop*.

A sintaxe básica para o comando:

```
yarn [--config confdir] COMMAND COMMAND_OPTIONS
```

Application: lista, obter status e mata um aplicativo:

```
yarn application -list
```

```
yarn application -status <job>
```

```
yarn application -kill <job>
```

Node: lista e obter status dos nós

```
yarn node -list
```

```
yarn node -all -list
```

```
yarn node -status quickstart.cloudera:47512
```




Comandos

Logs: obtém logs de um aplicativo já finalizado

```
yarn logs -applicationId <job>
```

Classpath: retorna o valor do classpath atual

```
yarn classpath
```

Version: retorna a versão atual do Cluster Yarn

```
yarn version
```

Top: fornece um resumo de informações:

```
yarn top
```

Indicações e Bibliografia

[Site oficial Apache Hadoop](#)

[Download Apache Hadoop](#)

[Documentação e Getting Started](#)

[O que é Hadoop?](#)

[Livro - Hadoop: The Definitive Guide](#)

Obrig.ada