Spark
Aplicações Spark - Conceitos

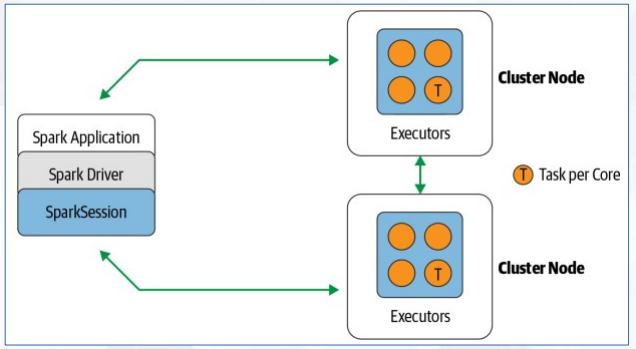
2023

<lapti>

Agenda

- Spark Application
- SparkSession
- Transformações
- Ações
- Lazy Evaluation
- Transformações narrow e wide
- Spark UI

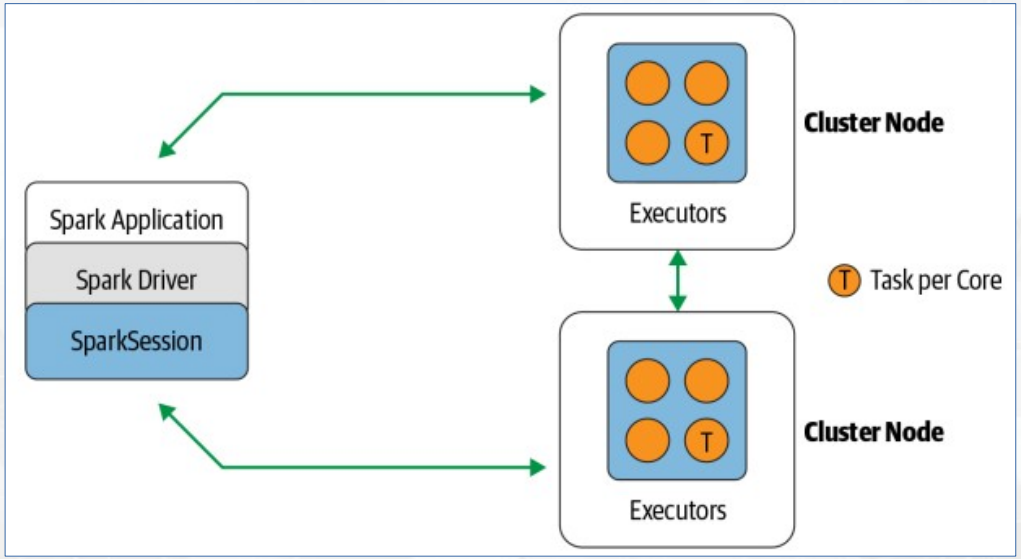
Conceitos principais



- Aplicação
  - Programa escrito usando APIs do Spark
  - É executado como um **driver program** e **executors** no cluster
- SparkSession
  - Objeto para interação com a funcionalidade do Spark
  - Permite que programadores acessem os recursos da API
  - Criado automaticamente em um “spark shell” ou explicitamente em um programa
- Job
  - Processamento paralelizado, consistindo de múltiplas tasks, que respondem a “Spark actions”
- Stage
  - Cada job é dividido em um conjunto menor de tasks que são interdependentes
- Task
  - Unidade única de trabalho ou execução que é submetida a um “Spark executor”

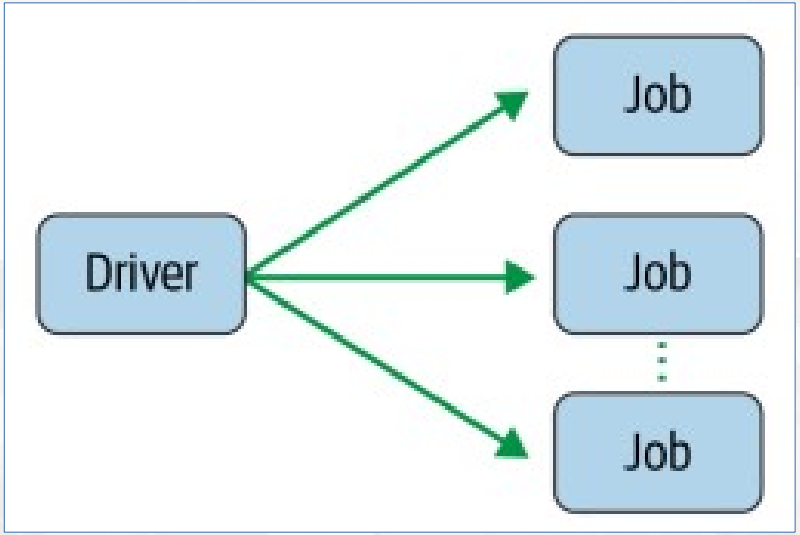
Spark Application e SparkSession

- Em cada aplicação Spark, é executado um driver program
  - Que cria a SparkSession
- Em um shell, SparkSession é criada implicitamente
- Em execuções locais, todos os processos rodam na mesma JVM
  - A criação da SparkSession pode apontar para outro cluster manager
  - Opções do shell podem também conectar a outros clusters
    - Parâmetros em spark-shell e pyspark



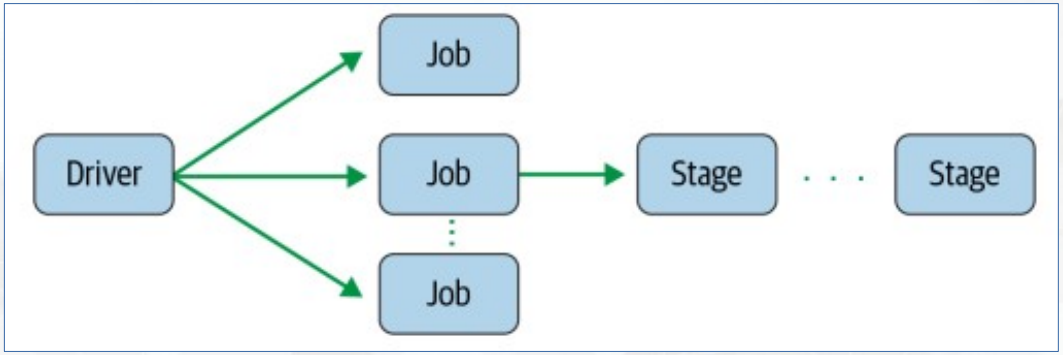
Spark Jobs

- Driver program converte uma aplicação em um ou mais *jobs*
  - Cada job então é transformado em um DAG
  - DAG é o equivalente do Spark a um plano de execução
    - Execution plan
  - Cada nó em um DAG pode ser um ou mais *stages*



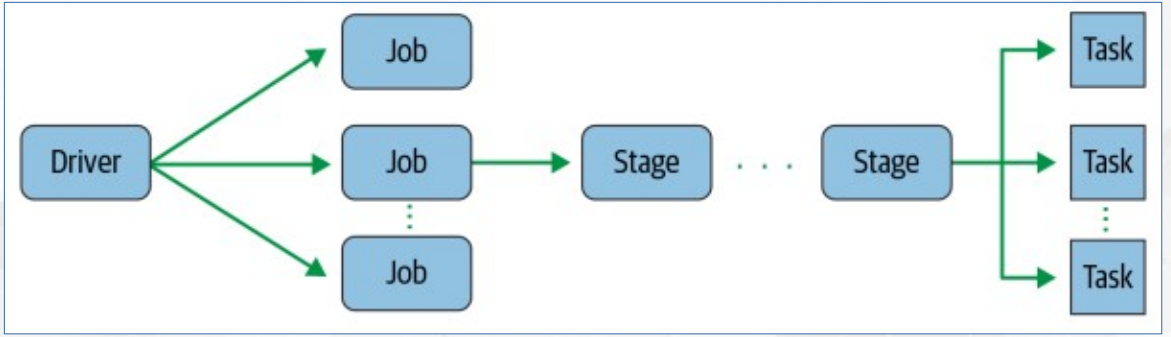
Spark Stages

- Stages são criados baseados em quais operações devem ser executadas
  - De maneira serial ou paralela
- Nem todas as operações podem executar em um único stage
  - Divisão de stages
- Geralmente definidos pelos limites do processamento de cada operador
  - O que define como os dados serão transferidos entre executors



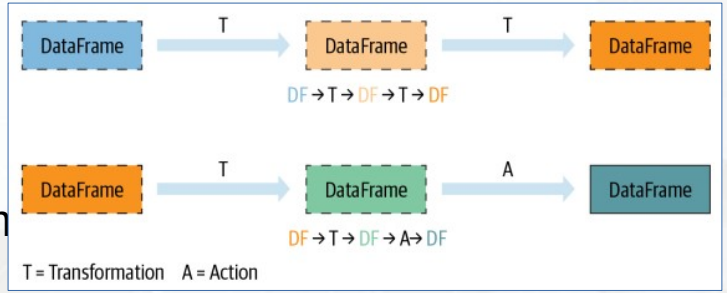
Spark Tasks

- Cada stage é dividido em tasks
  - Unidades de execução
- Que são então federadas para cada Spark executor
  - Cada task em um único core e trabalhando sobre uma única partição de dados
    - Um executor com 12 cores pode ter 12 ou mais tarefas trabalhando em 12 ou mais partições em paralelo

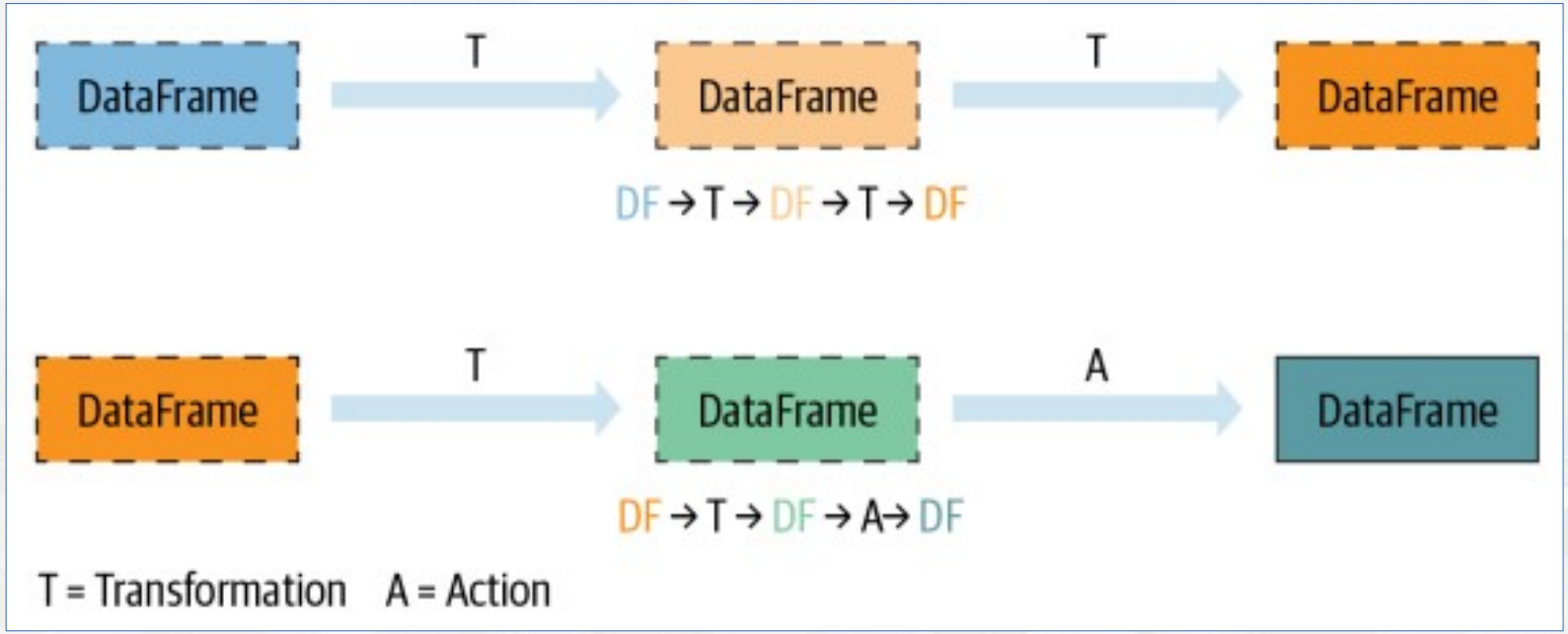


Operações e *Lazy Evaluation*

- Operações em dados distribuídos podem ser de dois tipos
  - Transformações
  - Ações
- Transformações
  - Transformam um Spark DataFrame em um novo DataFram
    - Sem alterar os dados originais
    - Princípio de imutabilidade
    - Operações como select(), filter() ou where() não alteram o DataFrame original
      - Retorno são os resultados transformados da operação como um novo DataFrame
- Todas as transformações são avaliadas de maneira tardia
  - Lazy evaluation
  - Não são executadas imediatamente, mas encadeadas em uma sequência

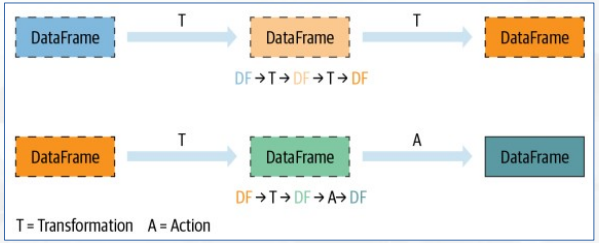


Operações e Lazy Evaluation



Operações e Lazy Evaluation

- Uma sequência de operações definida e não executada tem vantagens para otimização
  - Permite que o Spark defina o plano de execução posteriormente
    - Possivelmente rearranjando determinadas transformações, combinando algumas, otimizando em stages para execução mais eficiente



- Lazy evaluation
  - Estratégia de postergar a execução real até que uma ação seja invocada ou que os dados sejam “tocados” (lidos ou gravados)
  - Ação dispara a execução da sequência de transformações

Operações e Lazy Evaluation

- Lazy evaluation
  - Permite alto grau de otimizações nas queries
    - Cadeia de transformações
  - Nenhum query plan é executado até uma ação ser invocada
- Imutabilidade
  - Permite tolerância à falhas
    - Todas as transformações e sequências são armazenadas em log
    - Se nós ou dados são perdidos, o Spark pode reproduzir o estado original simplesmente pela repetição das sequências logadas
      - Resiliência em falhas

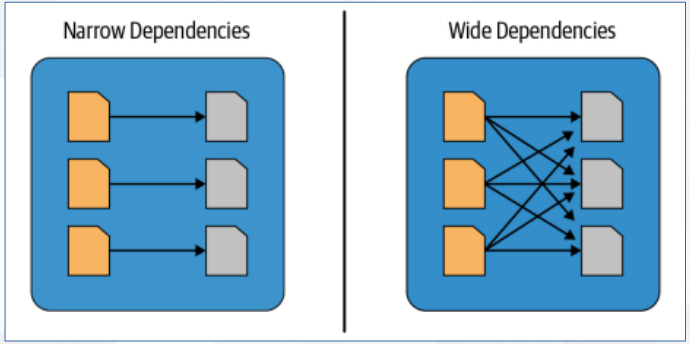
Operações e Lazy Evaluation

- Exemplos de transformações e ações

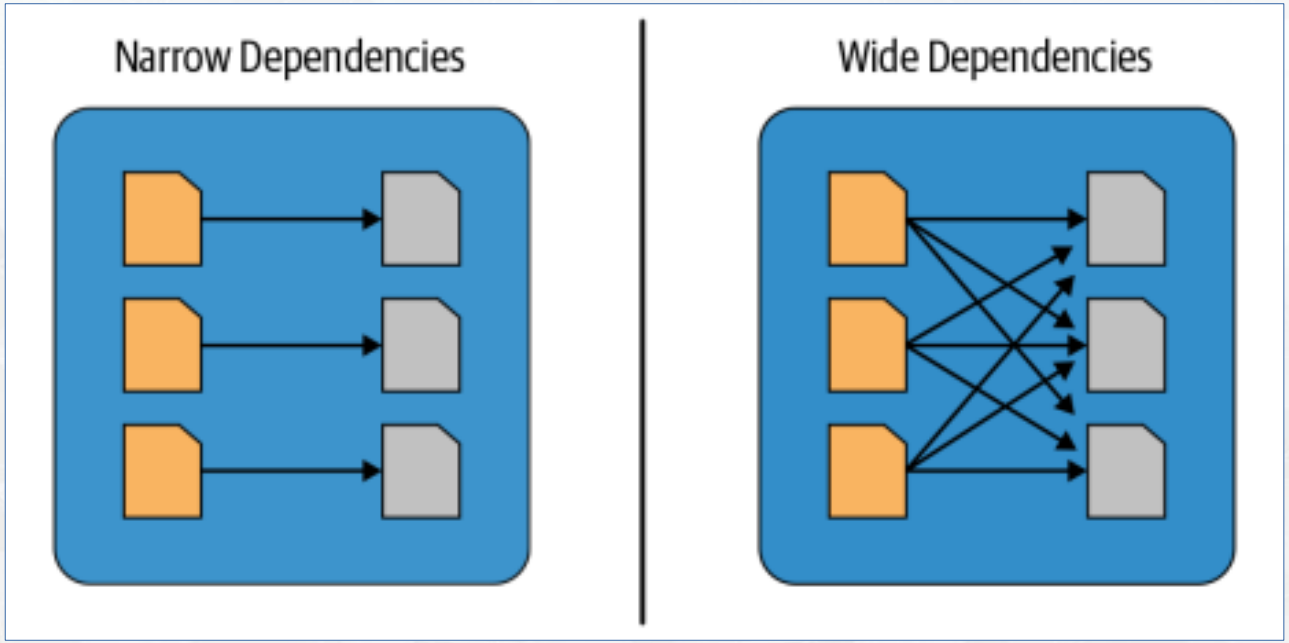
Transformações	Ações
`orderBy()`	`show()`
`groupBy()`	`take()`
`filter()`	`count()`
`select()`	`collect()`
`join()`	`save()`

Transformações Narrow e Wide

- Grande vantagem de Lazy Evaluation
  - Permitir a inspeção da sequência de passos da query
    - E otimizar de acordo
  - Otimização por agregar ou colocar em sequência operações
- Dependências de operações e dados
  - Narrow (estreita)
    - Processada a partir de uma única partição, para saída de uma única partição
    - Ex.: filter(), contains()
  - Wide (larga)
    - Usa dados de outras partições
    - Força uma busca de dados de partições de outros executors pelo cluster
    - Ex.: groupBy(), orderBy()

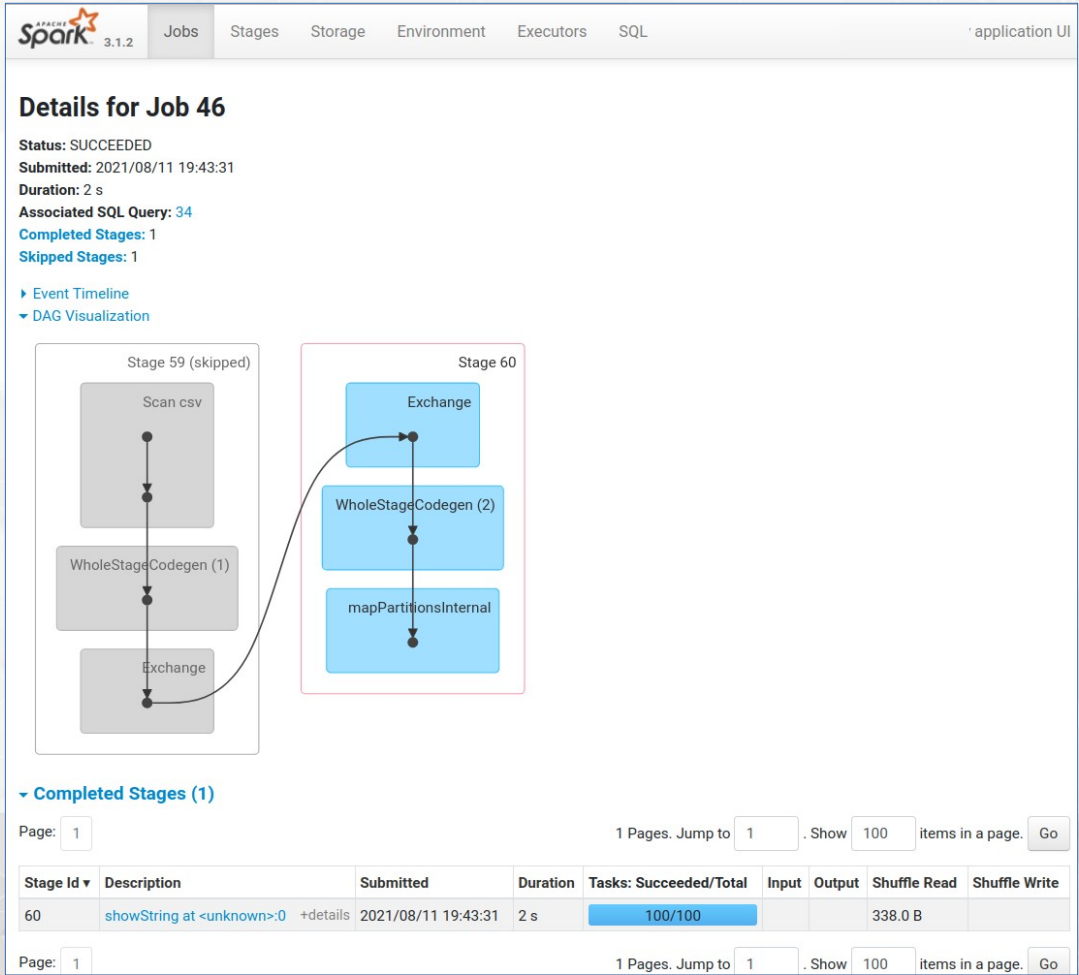


Transformações Narrow e Wide



Spark UI

- Monitoramento de aplicações
  - Decomposição em jobs, stages e tasks
  - Em modo local, http://localhost:4040
    - Ou em porta anunciada nos logs, se indisponível



Obrigado

leandro@utfpr.edu.br

<lapti>