Ambiente Livre
Tecnologia

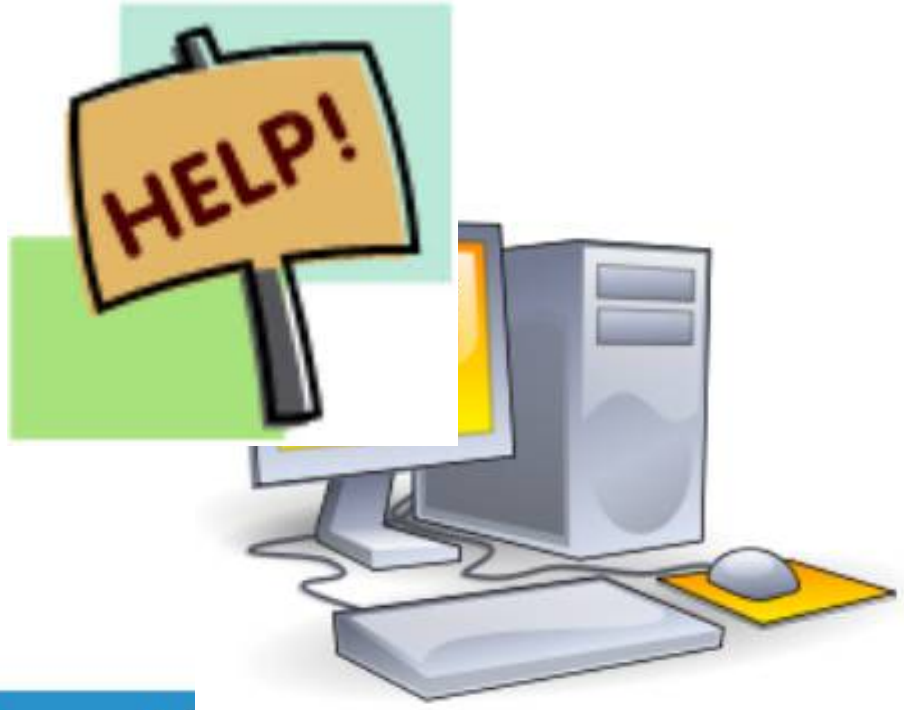
Hadoop

Treinamento Hadoop – Big Data Open Source - Fundamental.

Instrutor: Marcio Junior Vieira.
marcio@ambientelivre.com.br

MapReduce

- É um modelo de programação desenhado para processar grandes volumes de dados em paralelo, dividindo o trabalho em um conjunto de tarefas independentes



Ambiente Livre
Tecnologia

Programação Distribuída

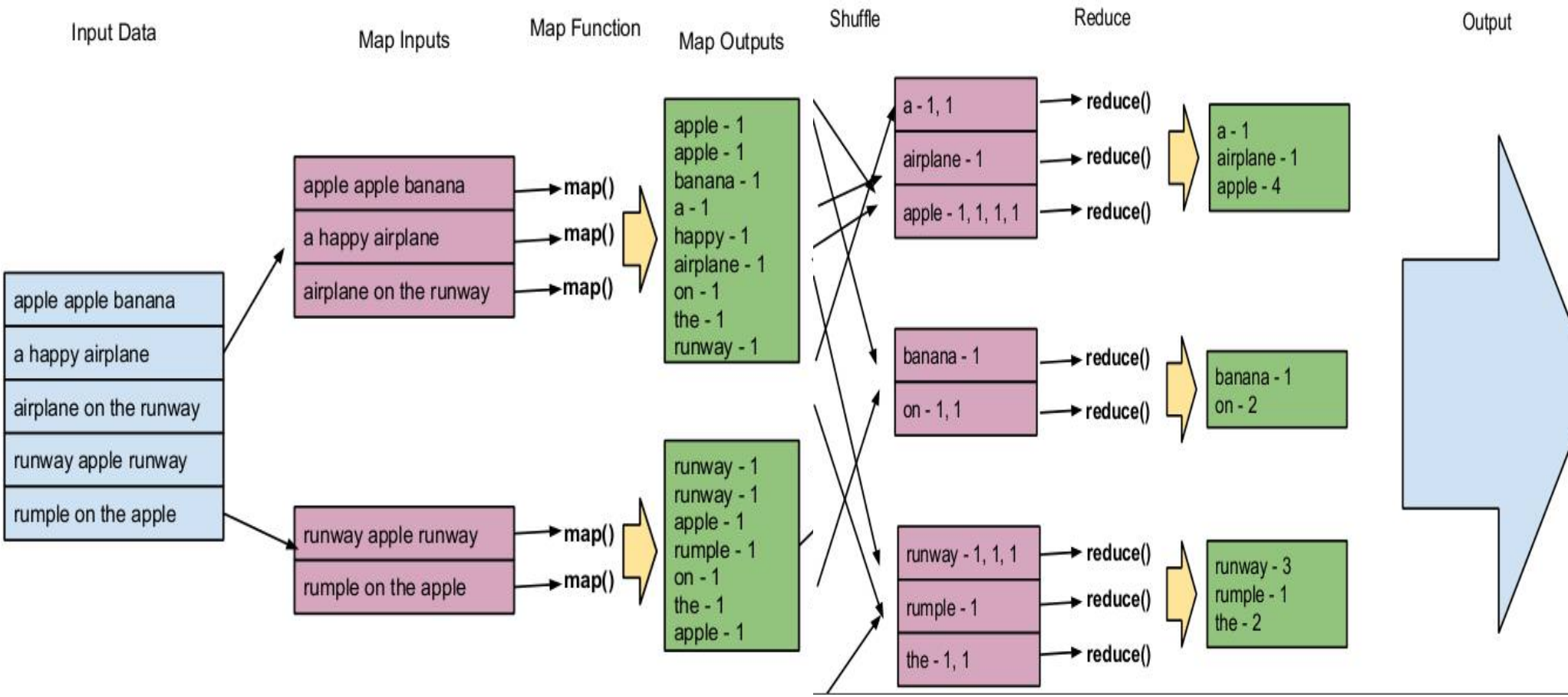
MapReduce

- MapReduce é um modelo de programação e implementação associado para o processamento e geração de grandes conjuntos de dados.
- Conceito Criado a mais de 40 anos
- Baseado em um modelo de programação funcional (como Lisp, Ml, etc)
- Processamento de dados base em batch
- abstração limpa para programadores
- Paralelização automática e distribuição
- Tolerância a falhas

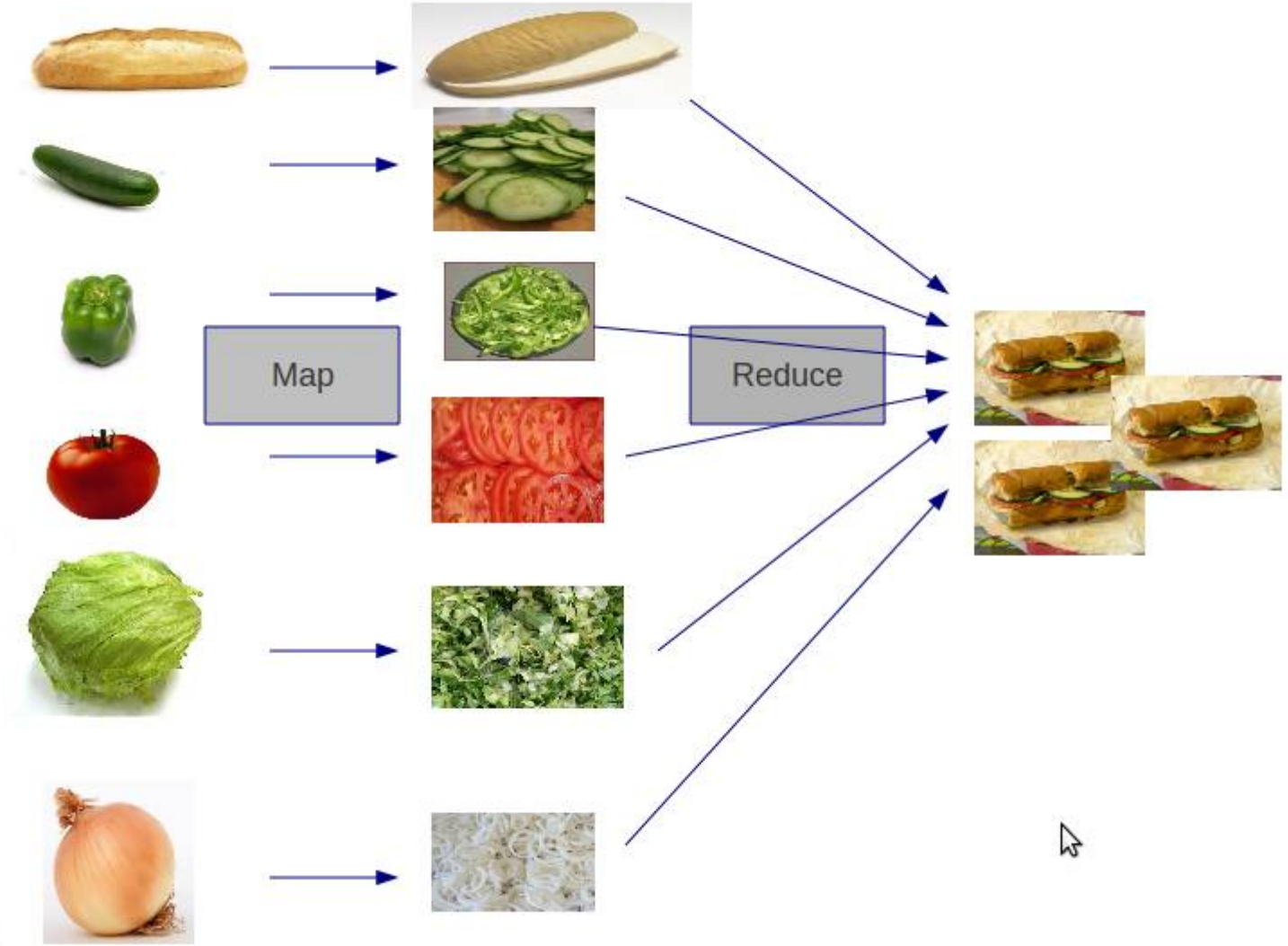
MapReduce

Map

Reduce



Map-Reduce-Digest



MapReduce

Você especifica o map (...) e reduce (...) funções

- map = (lista (k, v) -> lista (k, v))
- reduce = (k, lista (v) -> k, v)

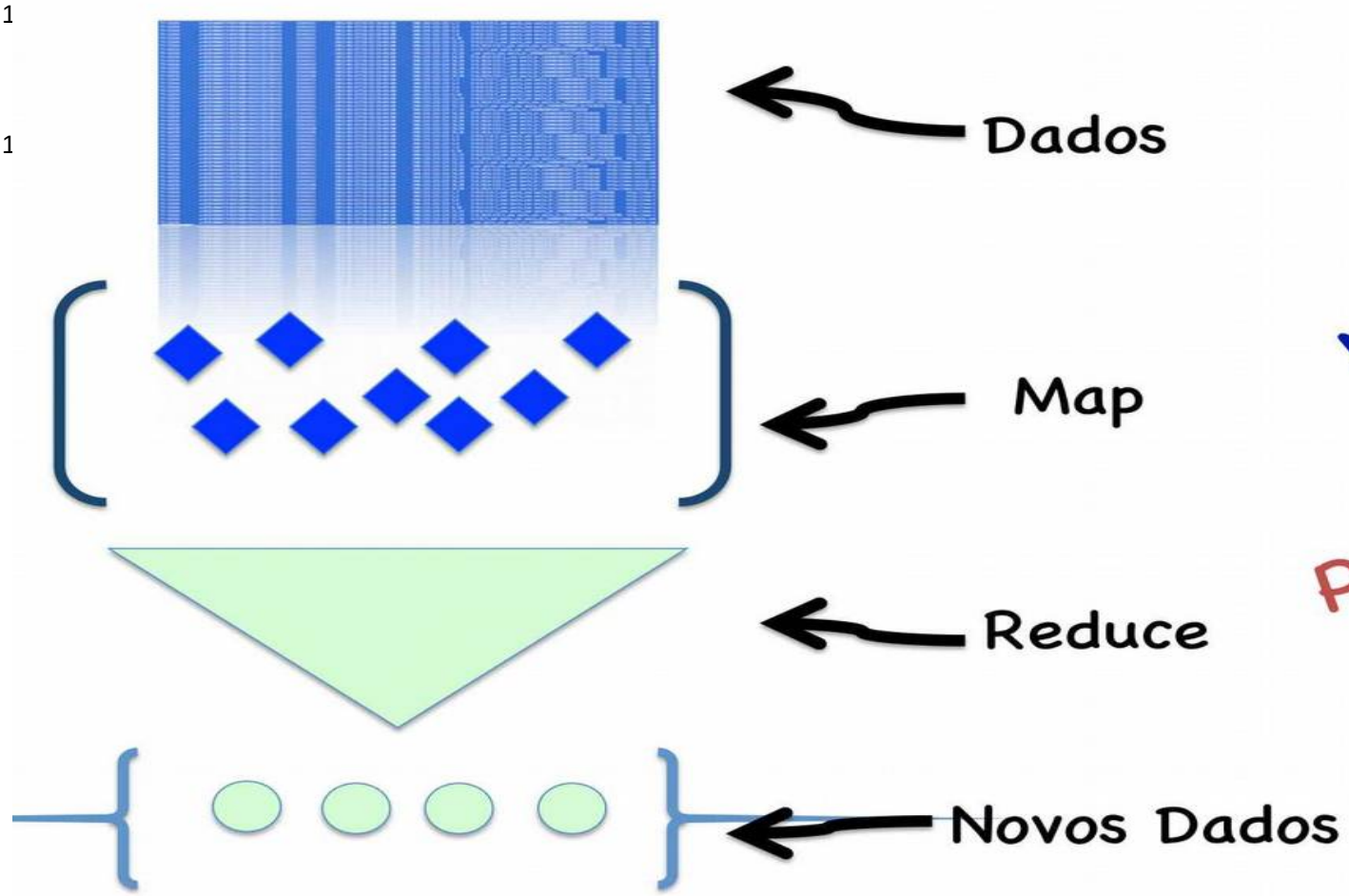
O Framework faz o resto

- Dividir os dados
- Executa vários mappers sobre as divisões
- Embaralhar os dados para os redutores
- Executa vários redutores
- Guarda os resultados finais

MapReduce

1

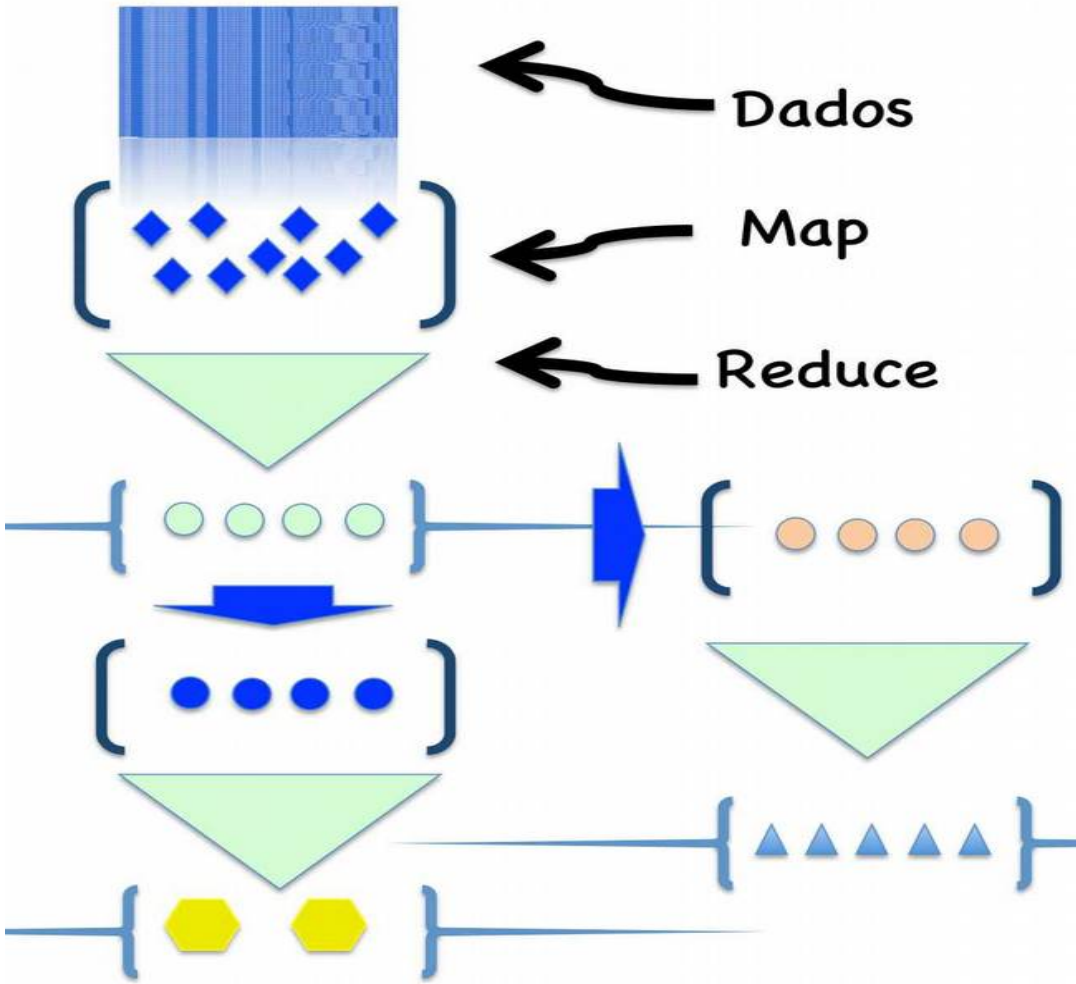
1



MapReduce

- A abordagem feita pelo MapReduce pode parecer uma abordagem de força bruta.
- A premissa é que todo o conjunto de dados, ou pelo menos uma boa parte dele é processada para cada consulta. Mas este é o seu poder.
- MapReduce é um processador de consulta de lote, e a capacidade de executar uma consulta ad hoc contra todo o conjunto de dados e obter os resultados em um tempo razoável é transformadora.
- Isso muda a maneira que você pensa sobre os dados, e desbloqueia dados que foram previamente arquivados em fita ou disco. Ela oferece às pessoas a oportunidade de inovar com os dados. Perguntas que levou muito tempo para ser respondida antes já pode ser respondida, que por sua vez leva a novas questões e novas perspectivas.

MapReduce



MapReduce
PIPELINE

Quando MapReduce é Necessário

- Por que não podemos usar bancos de dados com muitos discos para fazer análise de lote em larga escala?
- A resposta a estas perguntas vem de outra tendência em unidades de disco:
- O tempo de busca é mais lentamente do que a taxa de transferência.
- Buscando é o processo que mover a cabeça do disco para um lugar especial no disco para ler ou gravar dados.
- Ele caracteriza a latência de uma operação de disco, ao passo que a taxa de transferência corresponde à largura de banda de um disco.

RBMS x MapReduce

- Se o padrão de acesso a dados é dominado pela procura, vai demorar mais tempo para ler ou escrever grande porções do conjunto de dados que flui através dele, que opera na taxa de transferência.
- Para atualizar a maioria de um banco de dados, uma B-Tree é menos eficiente do que o MapReduce, que usa Sort / Merge para reconstruir o banco de dados.

RBMS x MapReduce

- Em muitos aspectos, o MapReduce pode ser visto como um complemento para um RDBMS.
- MapReduce é um bom ajuste para problemas , que precisa de analisar todo o conjunto de dados, de uma forma de lote, em particular para análise ad hoc.
- Um RDBMS é bom para consultas pontuais ou atualizações, onde o conjunto de dados foi indexado para fornecer recuperação de baixa latência e os tempos de atualização de uma quantidade relativamente pequena de dados.
- MapReduce se adapte às aplicações onde os dados são gravados uma vez, e li muitos vezes,
- Enquanto um banco de dados relacional é bom para conjuntos de dados que são continuamente atualizados.

RBMS x MapReduce

	Traditional RDBMS	MapReduce
Data size	Gigabytes	Petabytes
Access	Interactive and batch	Batch
Updates	Read and write many times	Write once, read many times
Structure	Static schema	Dynamic schema
Integrity	High	Low
Scaling	Nonlinear	Linear

Diferença MapReduce /RDBMS

- **RDMS:**
- Dados estruturados são dados que são organizados em entidades que tem um formato definido , como documentos XML ou tabelas de banco de dados que se conformam a um esquema pré-definido particular.
- **MapReduce:**
- Semi-estruturado de dados, por outro lado, é mais flexível e, embora possa haver um esquema, que é muitas vezes ignoradas por isso, só pode ser usada como um guia para a estrutura de dados por exemplo, uma folha de cálculo, em que a estrutura é a grade de células, embora as próprias células podem conter qualquer forma de dados.
- Dados não estruturados não tem qualquer estrutura interna especial: para exemplo, texto ou dados de imagem. MapReduce funciona bem em não estruturados ou semi-estruturados dados, uma vez que ele é projetado para interpretar os dados de tempo de processamento.

A distribuição dos dados

- Os dados relacionais são muitas vezes normalizados para reter a sua integridade e remover redundância.
- Normalização coloca problemas para MapReduce, uma vez que torna a leitura de um registro de uma operação não-local, e um dos pressupostos centrais que MapReduce é que ele faz é possível realizar (alta velocidade) de streaming lê e escreve.

Modelo de Programação

- **MapReduce** é um modelo de programação linear escalável.
- O programador escreve duas funções de mapa e um reduzem a função,
- cada um dos quais um funciona-define um mapeamento a partir de um conjunto de pares de valores-chave para outro. Estas funções são indiferentes ao tamanho da os dados ou a fragmentação que estão a funcionando
- Mais importante, se você dobrar o tamanho da entrada de dados, um trabalho será executado duas vezes mais lento. Mas se você também o dobro do tamanho do cluster, um trabalho vai correr tão rápido quanto o original. Isso geralmente não é verdadeiro de consultas SQL.

Função MAP

- **O Objetivo e estrais alguma informação de Valor**
- **Fase Intermediária: Shuffle & Sort**
- **Input:**

  Registros de alguma fonte de dados (por exemplo, linhas de arquivos, linhas de um banco de dados,etc) estão associados no par (chave, valor)
- **Output:**

  Um ou mais valores intermediários no formato (chave, valor)

- Exemplo: (palavra, numero_de_ocorrencias)

MAP em Funcional

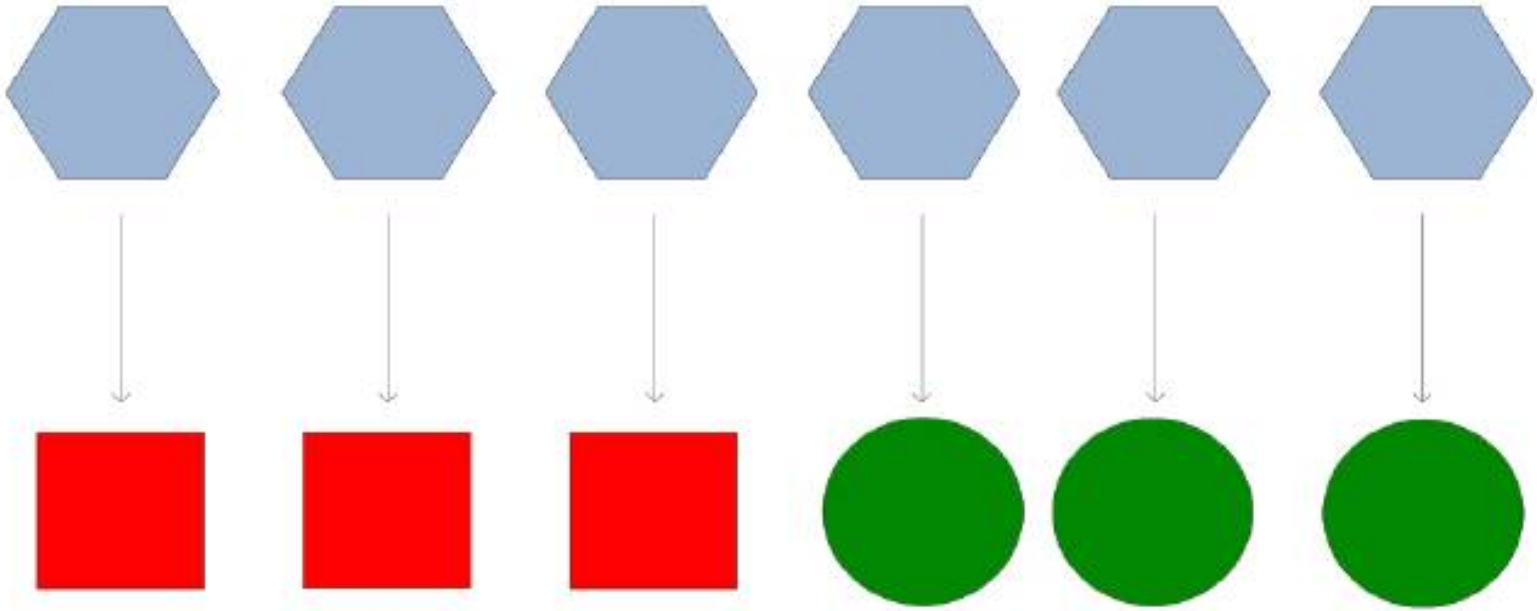
- Map em Programação Funcional

  map({1,2,3,4}, {x2} -> {2,4,6,8}

- Todos os elementos são processados por um método e os elementos não afetam um ao outro.

Função MAP

map (in_key, in_value) → (out_key, intermediate_value) list



Source: (Cloudera, 2010)

Funcão Reduce

- Depois que a fase de mapeamento terminar, todos os valores intermediários vão para uma chave de saída, estes são combinadas em uma lista (reúne compila , filtra e transforma)
- **Input**
- Valores Intermediários :

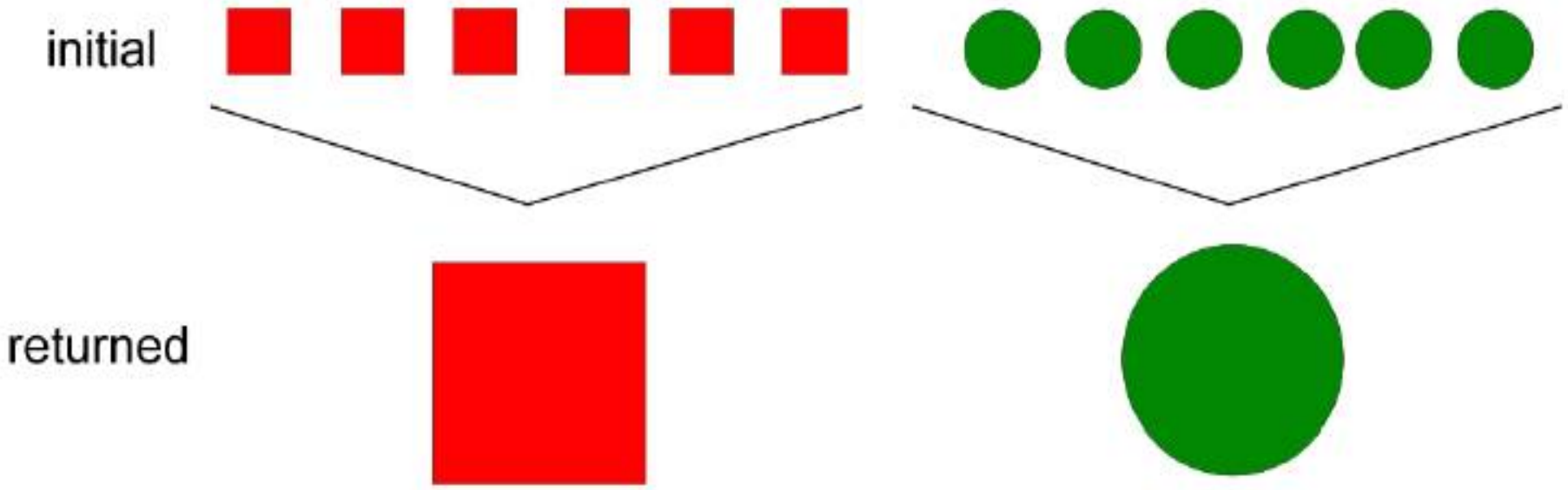
  Exemplo: (“A”, [42, 100, 312])
- **Output**

  Normalmente, apenas um valor final por chave

  Exemplo (“A”, 454)

Função Reduce

reduce (out_key, intermediate_value list) → out_value list

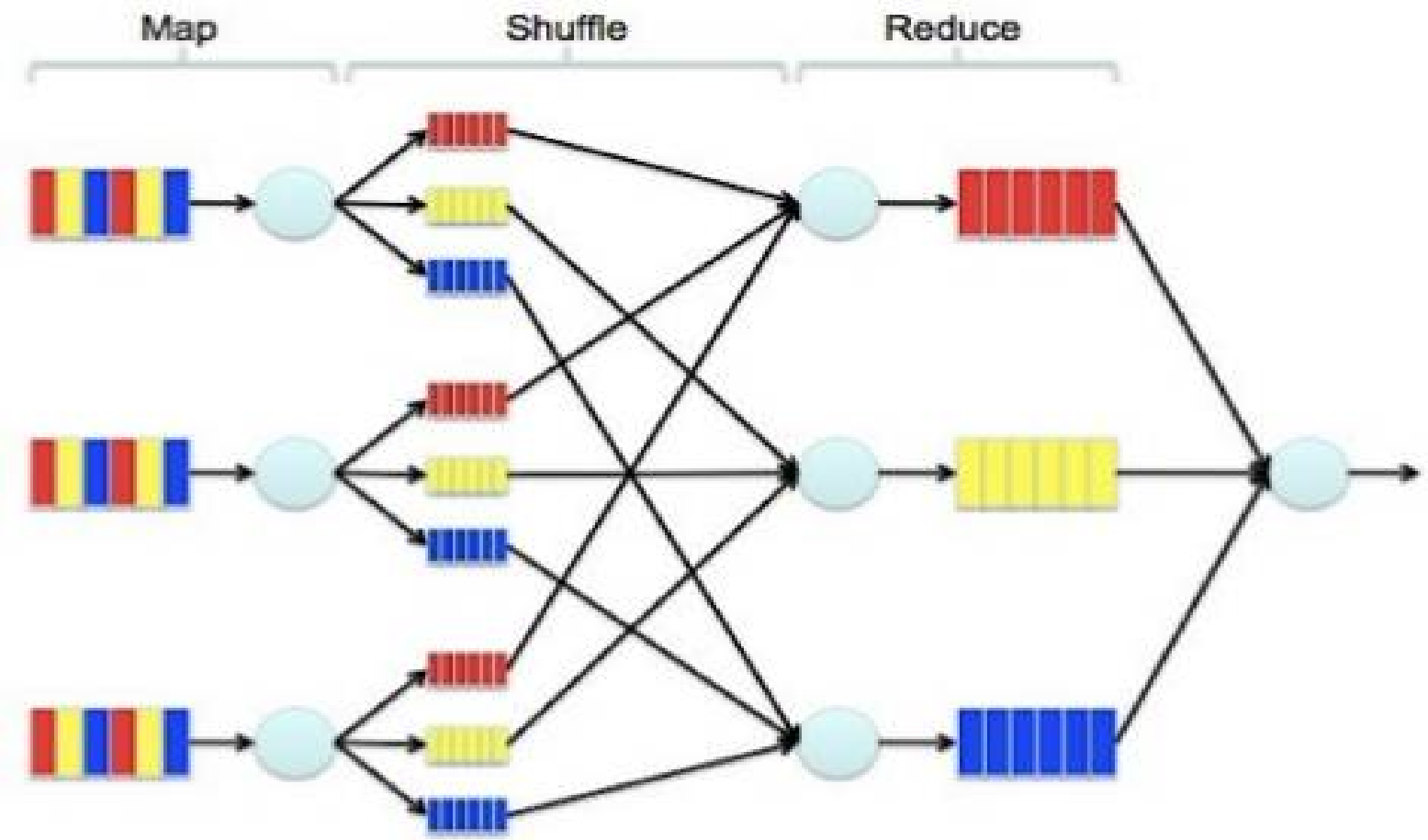


Source: (Cloudera, 2010)

Reduce Funcional

- Reduce({1,2,3,4} (x)) -> {24}
- Todos os elementos são processados juntos
- Tando em Map quando em Reduce a entrada é fixa (imutável) e a saída é uma nova lista (em gerak)

Lógica do MapReduce



Um Exemplo

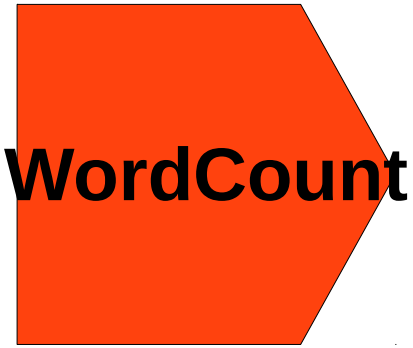
- **WordCount**(Conta Palavras)
  Gera uma lista de frequência das palavras em um conjunto de arquivos.

arquivo1.txt

treinamento Hadoop no Brasil
é na Ambiente Livre tecnologia
Brasil

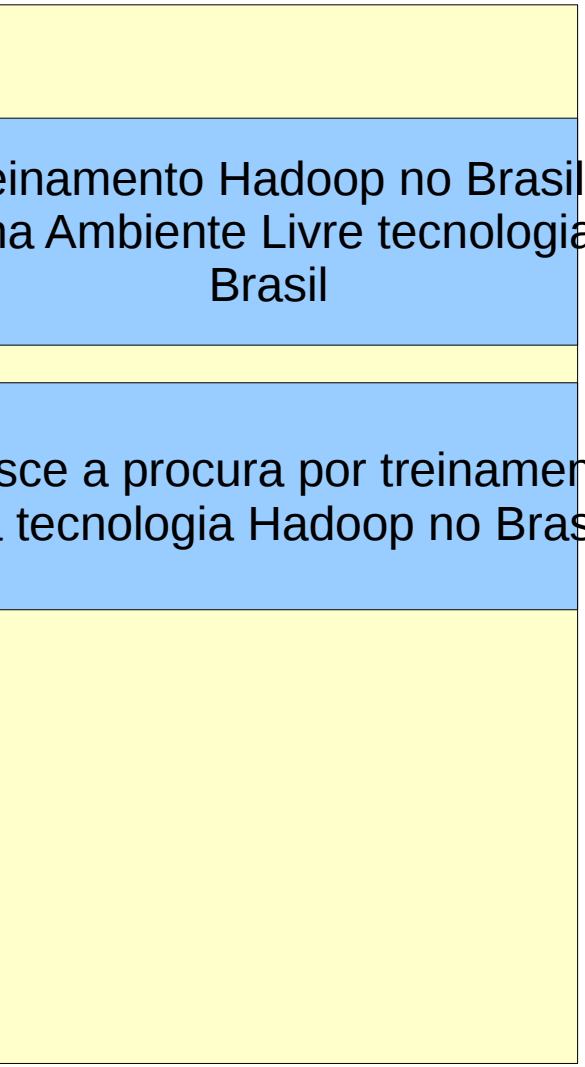
arquivo2.txt

Cresce a procura por treinamento
na tecnologia Hadoop no Brasil



Treinamento,2
Hadoop,2
tecnologia,2
Ambiente,1
Livre,1
na,2
é,1
no,1
Cresce,1
A,1
Procura,1
Por,1
Brasil,3

WordCount Exemplo



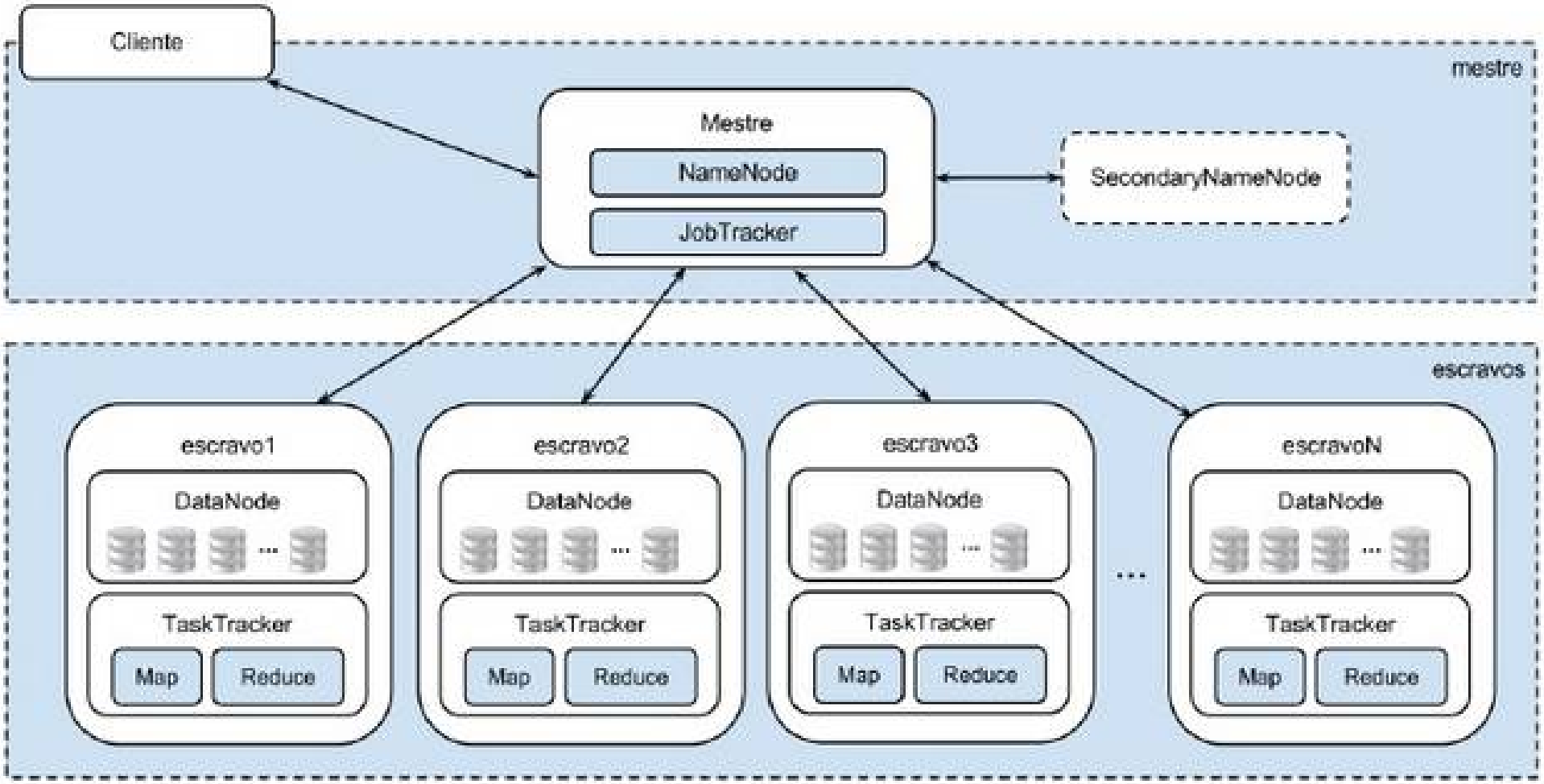
Componente JobTracker

- Gerencia o plano de execução de Tarefas do MapReduce
- Designa as tarefas aos nós escravos
- Monitora a execução das tarefas , para agir em caso de falhas.

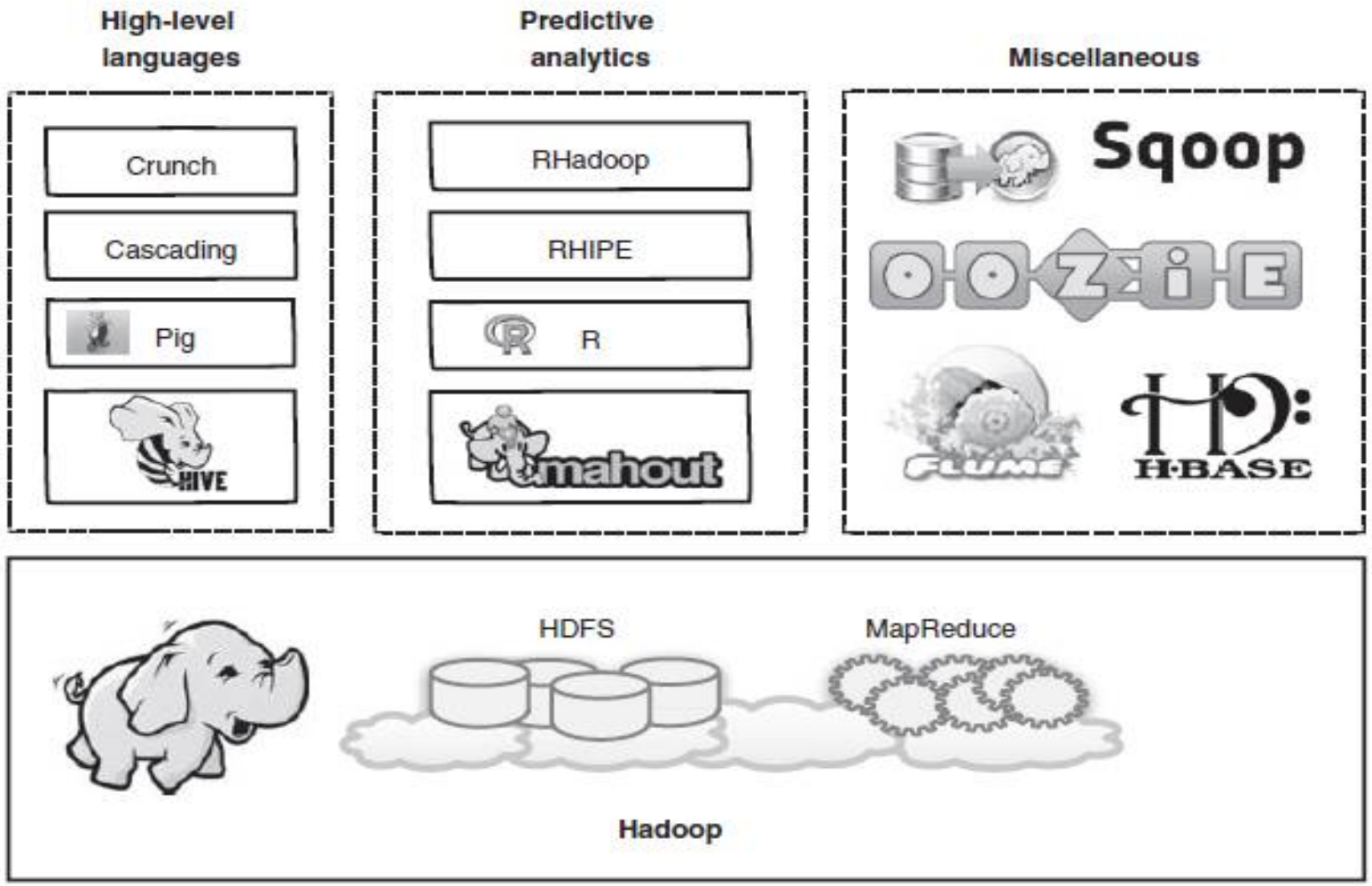
Componente TaskTracker

- Realiza o processamentos da tarefas MapReduce
- Uma instência em cada nó escravo.

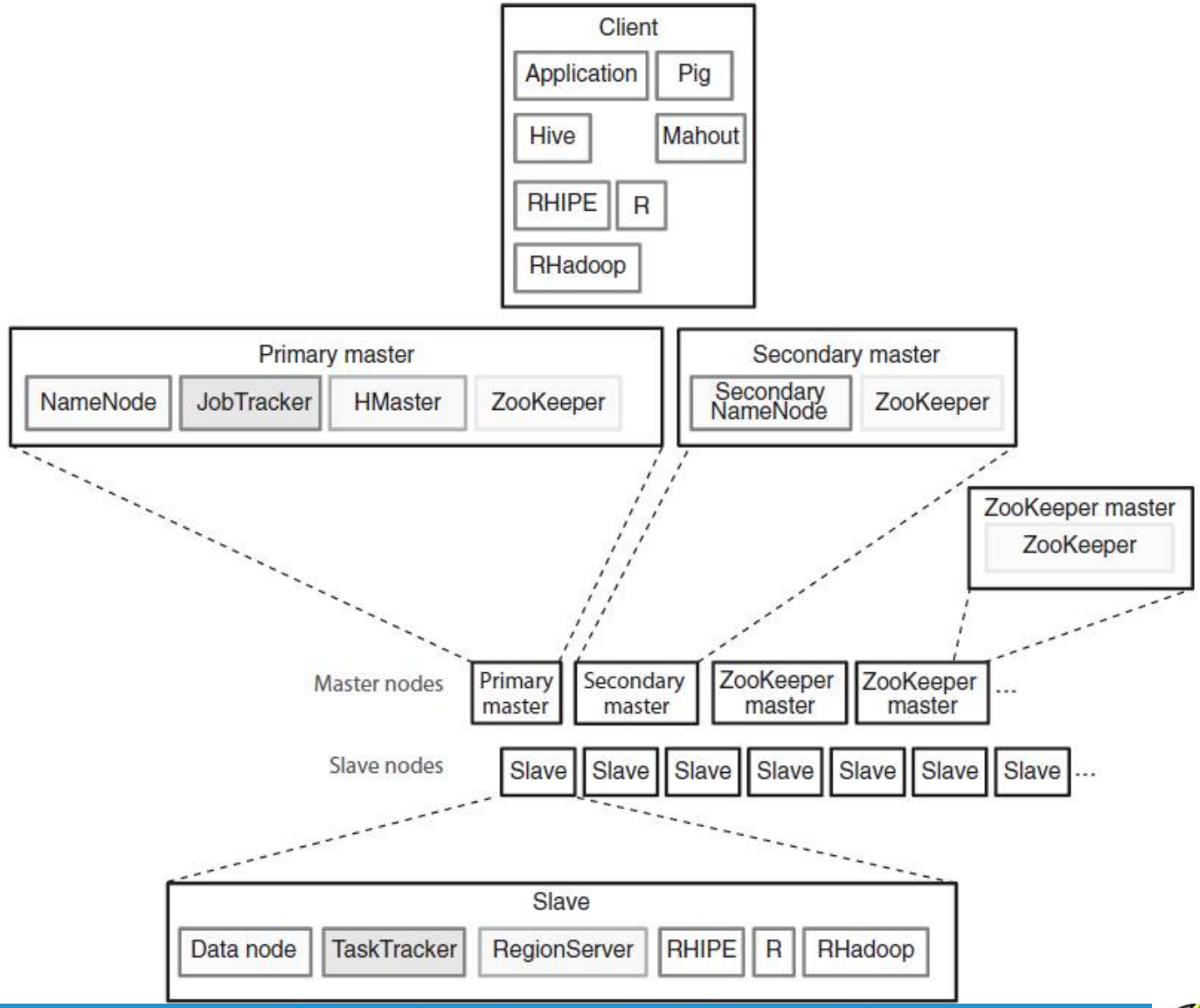
Ambiente


Cliente
mestre
Mestre
NameNode
JobTracker
SecondaryNameNode
escravos
escravo1
DataNode
TaskTracker
Map
Reduce
escravo2
DataNode
TaskTracker
Map
Reduce
escravo3
DataNode
TaskTracker
Map
Reduce
escravoN
DataNode
TaskTracker
Map
Reduce

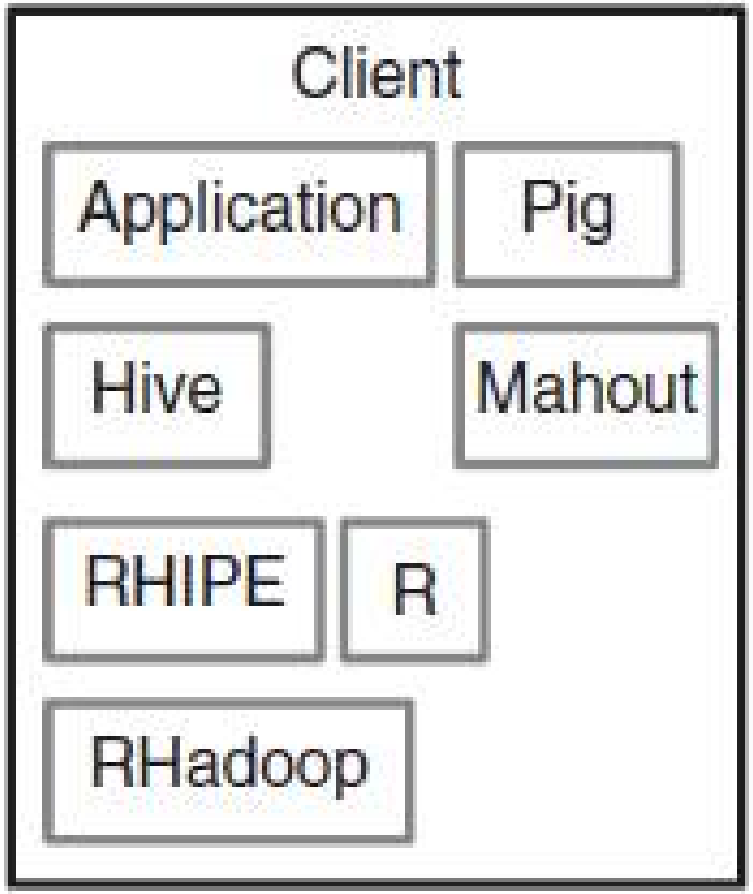
Tecnologias



Arquitetura Física



Clients



Hosts clientes executam código do aplicativo em conjunto com os projetos do ecossistema Hadoop.

Pig, Hive, e Mahout são projetos do lado do cliente, que não precisa ser instalado em seu cluster Hadoop real.

Limitações HDFS

- A fraqueza do HDFS é principalmente em torno de sua falta de alta disponibilidade, o seu manuseio ineficiente de arquivos pequenos, e sua falta de compressão transparente.
- HDFS não foi projetado para trabalhar bem com leituras aleatórias sobre pequenos arquivos devido a sua otimização para sustentado rendimento.

Limitações MapReduce

- MapReduce é uma arquitetura baseada em lotes, o que significa que não se presta a usar casos que necessitam de acesso a dados em tempo real.
- Tarefas que requerem sincronização global ou compartilhamento de dados mutáveis não são uma boa opção para MapReduce, porque é um shared-nothing arquitetura, que podem representar desafios para alguns algoritmos.