FUNDAMENTOS E TÉCNICAS EM CIÊNCIAS DE DADOS

PROF. JOSENALDE OLIVEIRA

josenalde@eaj.ufrn.br
https://github.com/josenalde/datascience

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - UFRN

SITUANDO A DATIFICAÇÃO...E O BIG DATA

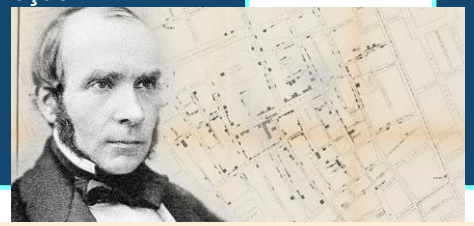
1750-1860
- Carvão, ferro, vapor, primórdios da automação

1860-1945
- Energia elétrica, aço
- Condições de trabalho

1945-90's
- +produtividade
- Comunicação
- tecnologia
- computação

1990-
- Atualização (sistemas de busca)
- Participação (redes sociais)
- Ação (solicitar serviços)
- Transformação

Tim Bernes-Lee (www, 1991)
Redes neurais (1990)



John Snow
surtos de cólera 1840-1850, Londres

- Aplicações 4.0 mais comuns: cidades, saúde, indústria, agricultura
- Associado ao termo INTELIGENTE, SMART
- Premissa: acesso a dados (infraestrutura de comunicação, dispositivos acessíveis)
- TECNOLOGIAS HABILITADORAS

- O termo 5.0 já está em discussão... (sistemas ciberfísicos, gêmeos digitais etc.)

SITUANDO A DATIFICAÇÃO...E O BIG DATA

PALAVRAS-CHAVE: PRECISÃO, PERSONALIZAÇÃO – Termo promovido pela IBM em 2011 (#IBMBigData); Google Aprendizagem Profunda (2012), uma máquina reconheceu um gato após ser apresentado a 10 M imagens de vídeos do Youtube por 03 dias - rede com 16 k computadores (distribuídos), com 01 bi de conexões.
Dica: o óbvio às vezes precisa ser explicado, não há problema. Se necessário DESENHE.
Entender a demanda/dificuldade do solicitante – **VISUALIZAR/DEMONSTRAR**

SITUANDO A DATIFICAÇÃO...E O BIG DATA

1954 – The first fully transistorised computer used all transistors and diodes and no vacuum tubes.
1964 – The IBM System/360 family of mainframe computer systems was launched.
1971 – Intel's 4004 became the first general purpose programmable processor.
1973 – Xerox unveiled the first desktop system to include a graphical user interface and internal memory storage.
1977 – ARCnet introduced the first LAN at Chase Manhattan Bank, connecting 255 computers.
1981 – The PC era began.
1983 – IBM released its first commercially available relational database, DB2.
1989 – Implementation of the Python programming language began.
1998 – Carlo Strozzi developed NoSQL, an open-source relational database.
1999 – VMware began selling VMware Workstation, allowing users to set up virtual machines.
2002 – Amazon Web Services (AWS) launched as a free service.
2006 – AWS started offering web-based computing infrastructure services, now known as cloud computing.
2007 – Apple launched the first iPhone, creating the mobile internet as we know it today.

2010 – The first solutions for 100 Gigabit Ethernet were introduced.
2011 – Facebook launched the Open Compute Project to share specifications for energy efficient data centers.
2013 – Docker introduced open-source OS container software.
2015 – Google and Microsoft lead massive build outs of data centers.
2017 – Huawei and Tencent joined Alibaba in major data centre build-outs in China.
2018 – Leading data center operators started the migration to 400G data speeds.
2018 – Silicon photonics technology started to positively impact data center networking architectures.
2020 – Edge computing will revise the role of the cloud in key sectors of the economy.
2021 – Data center speeds expected to exceed 1,000G.
2025 – Data centers will be increasingly on-device.

History of big data: Timeline (verdict.co.uk)

QUAL O "BACKEND" DO BIG DATA?

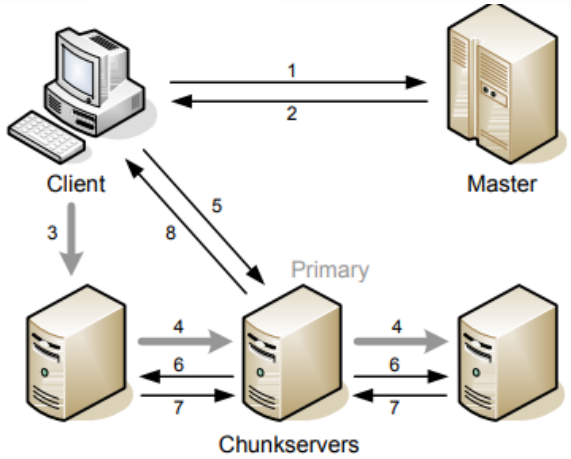
Apache Hadoop

Hadoop é uma plataforma de software de código aberto para o armazenamento e processamento distribuído de grandes conjuntos de dados, utilizando clusters de computadores com hardware commodity. Os serviços do Hadoop (Apache, 2006) fornecem armazenamento , processamento, acesso, governança, segurança e operações de Dados.

Baseado nos artigos Google File System (2003) e no modelo de programação de grandes volumes de dados em clusters MapReduce: https://static.usenix.org/publications/library/proceedings/osdi04/tech/full_papers/dean/dean.pdf

Armazenamento distribuído +

Processamento distribuído



Exemplos de aplicações:

Contar palavras
Frequência de acessos
Busca (grep) distribuída
Sort distribuído
Etc.

QUAL O "BACKEND" DO BIG DATA?

The primary role of MapReduce is to provide an infrastructure that allows development and execution of large-scale data processing jobs. As such, MapReduce aims at efficiently exploiting the processing capacity provided by computing clusters while at the same time offering a programming model that simplifies the development of such distributed applications. Moreover and similar to the requirements of GFS, MapReduce is designed to be resilient to failures such as machine crashes.

Google uses MapReduce to process data sets up to multiple terabytes in size for purposes such as indexing web content.

O principal objetivo do MapReduce é prover infraestrutura que permita o desenvolvimento e execução de Jobs de processamento de larga escala. Permite explorar de maneira eficiente a capacidade de processamento de clusters, ao mesmo tempo provendo um modelo simplificado para desenvolvimento de aplicações distribuídas. MapReduce também é construído para ser tolerante às falhas. Google usa MapReduce para processar múltiplos terabytes, por exemplo, para indexação de conteúdo web.

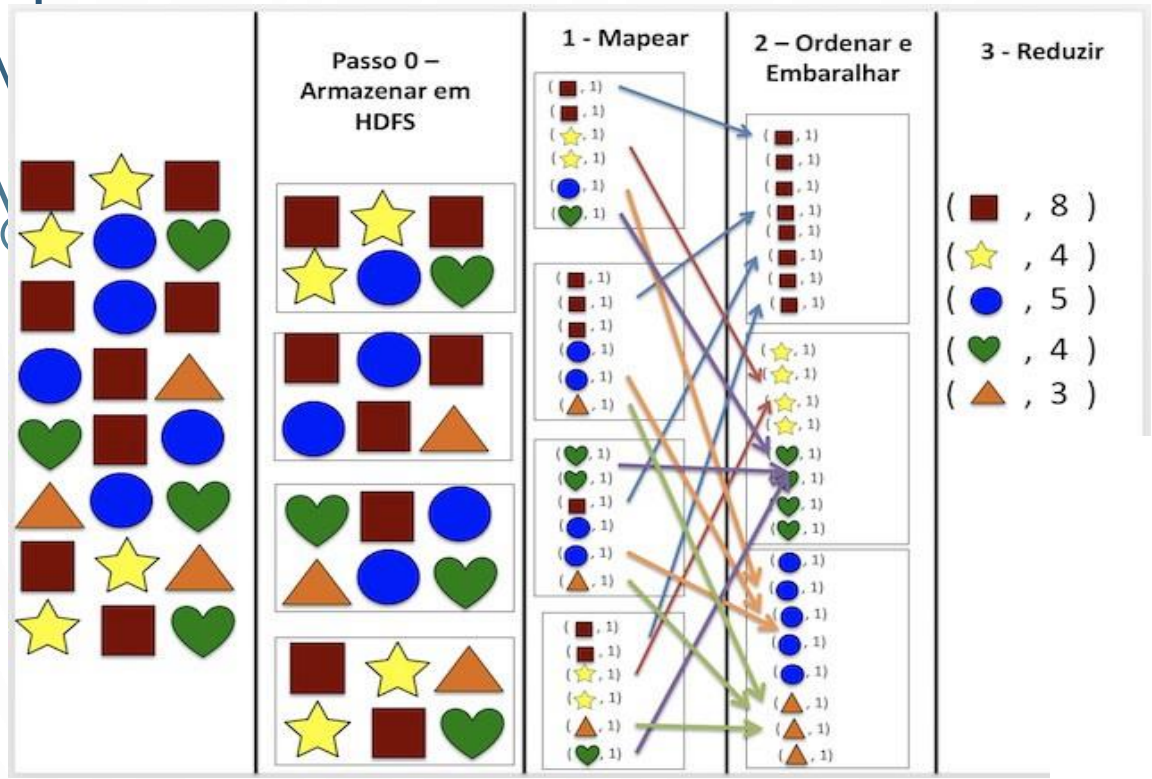
O modelo estabelece uma abstração que permite construir aplicações com operações simples, escondendo os detalhes da paralelização. Tais funções transformam um grande volume de dados de entrada em um conjunto resumido e agregado na saída.

Map é uma função de mapeamento que distribui os dados em diversos nós para processamento e armazenamento.

Reduce uma função que agrega e sumariza os resultados obtidos no mapeamento, para gerar um resultado final.

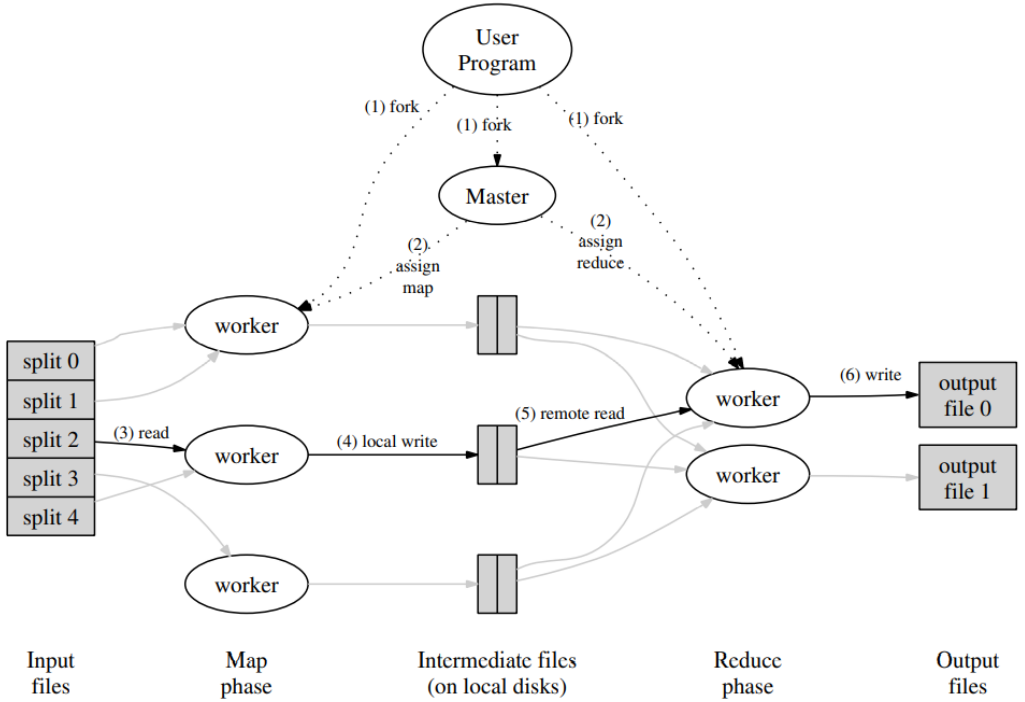
Twitter, Facebook, Google, Amazon e muitas empresas que utilizam Hadoop se valem do MapReduce em suas aplicações.

MAPREDUCE

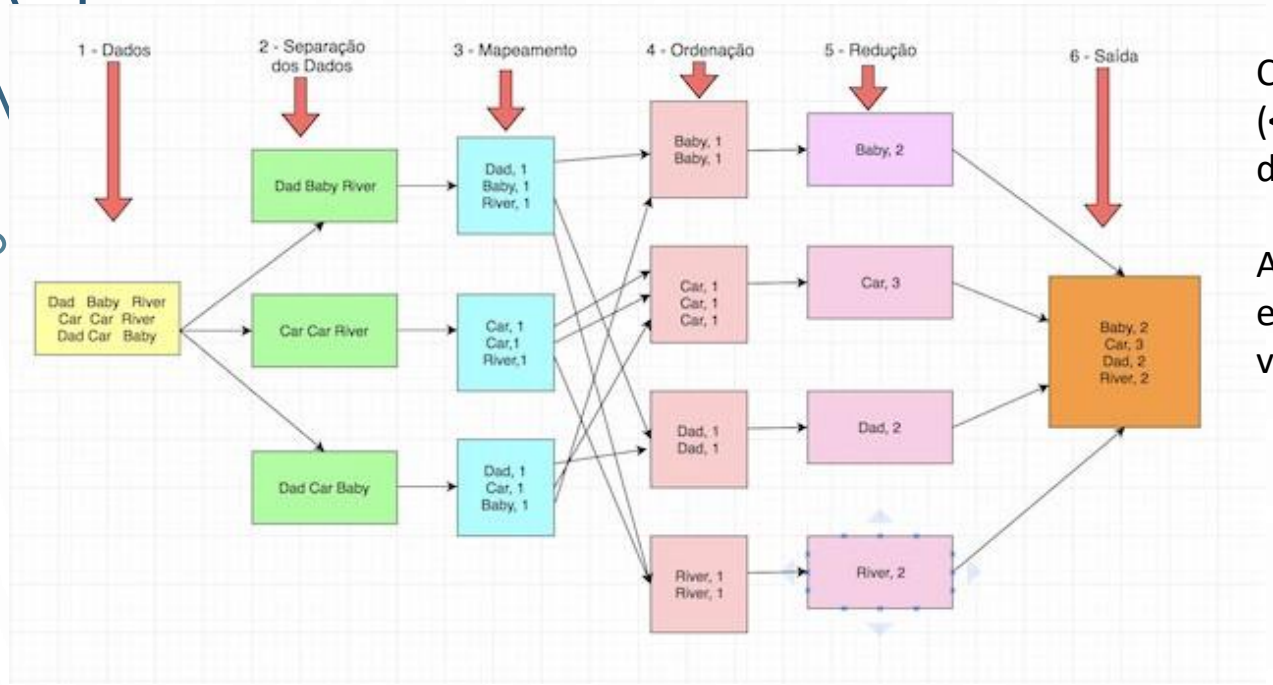



```
map(String key, String value):
  // key: document name
  // value: document contents
  for each word w in value:
    EmitIntermediate(w, "1");

reduce(String key, Iterator values):
  // key: a word
  // values: a list of counts
  int result = 0;
  for each v in values:
    result += ParseInt(v);
```

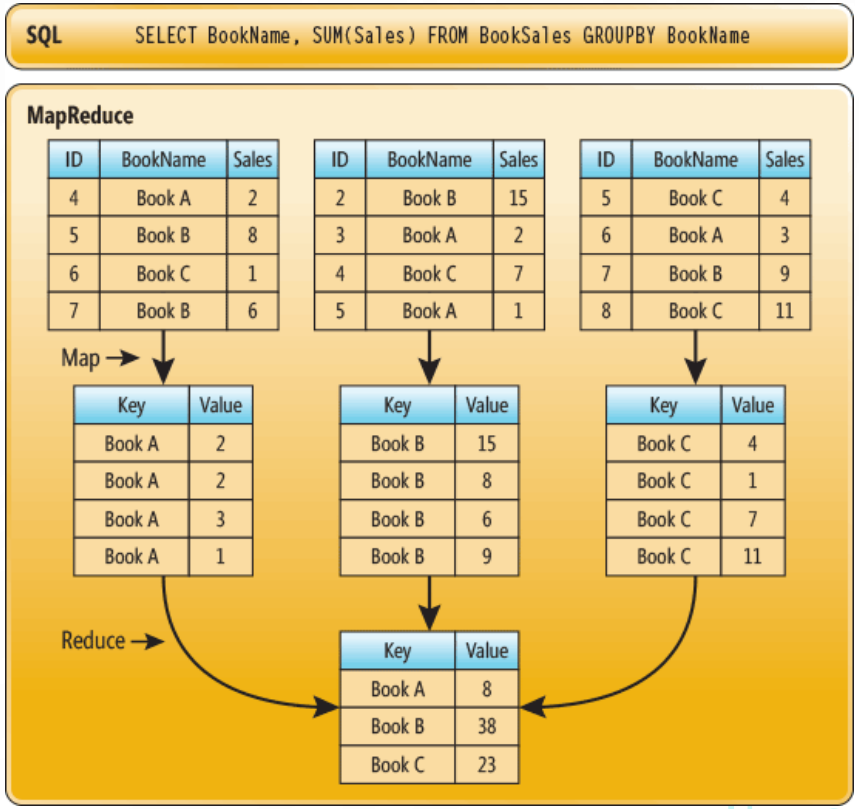


MAPREDUCE



Como padrão de MapReduce, a função Map recebe uma tupla (**<chave><valor>**) como entrada e gera um conjunto intermediário de dados, também no formato (**<chave><valor>**).

A função Reduce recebe como entrada as tuplas de (**<chave><valor**>) executadas para cada chave intermediária, com todos os conjuntos de valores associados àquela chave.



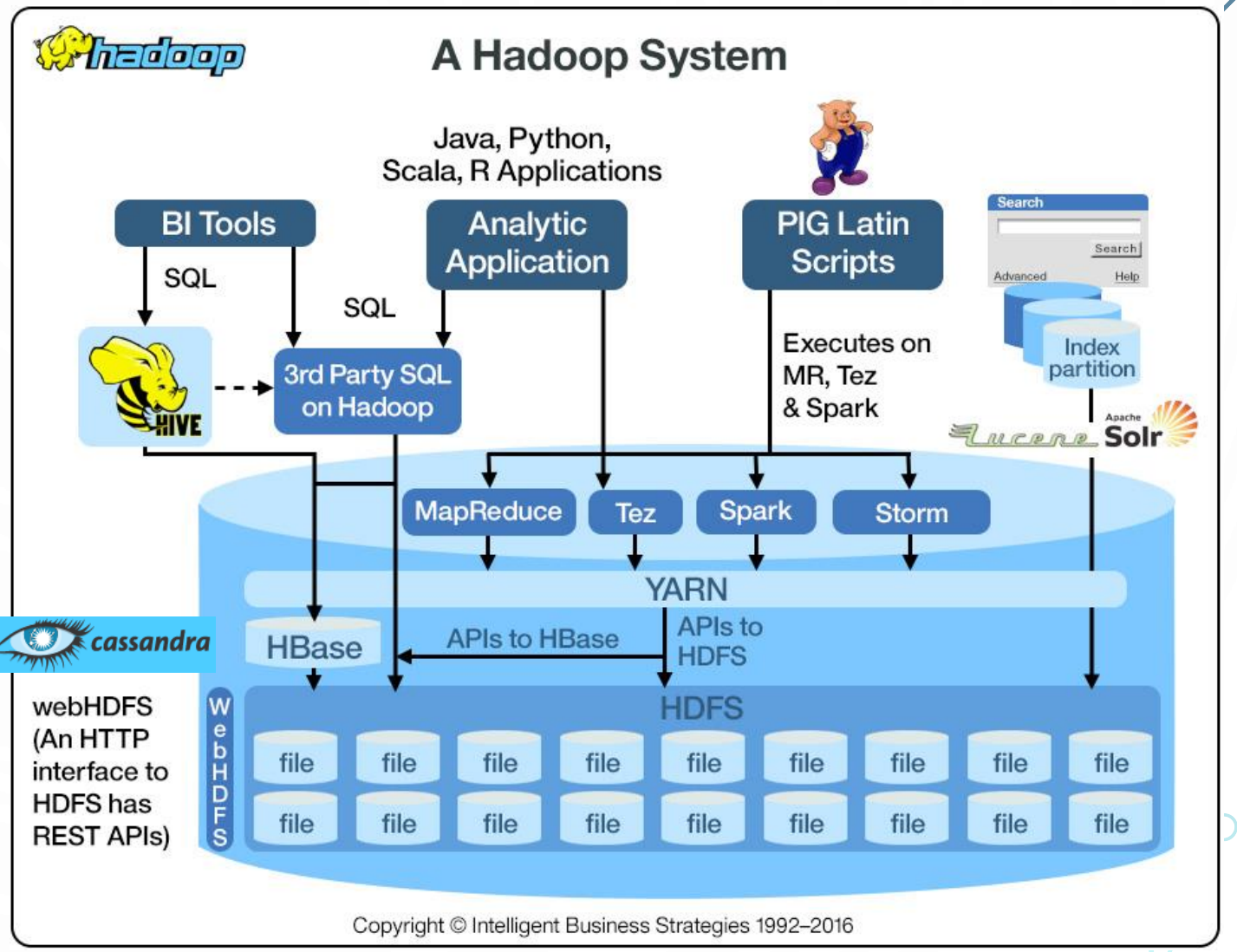
Neste exemplo, os dados em formato de texto foram separados, e mapeados, com a chave **<palavra><valor>** (**<Dad><1>**), em seguida ordenados, e depois reduzidos como **<palavra><valor>** (**<Dad> <2>**) e ao final temos então como resultado que a palavra **Dad** ocorre 2 vezes, **Car** três vezes, **Bebê** duas vezes e **Rio** duas vezes.

MapReduce recebe uma entrada, divide-a em partes menores, executa o código do mapeado em todas as partes e, em seguida, fornece todos os resultados para um ou mais redutores que mesclam todos os resultados em um único.

ECOSSISTEMA HADOOP

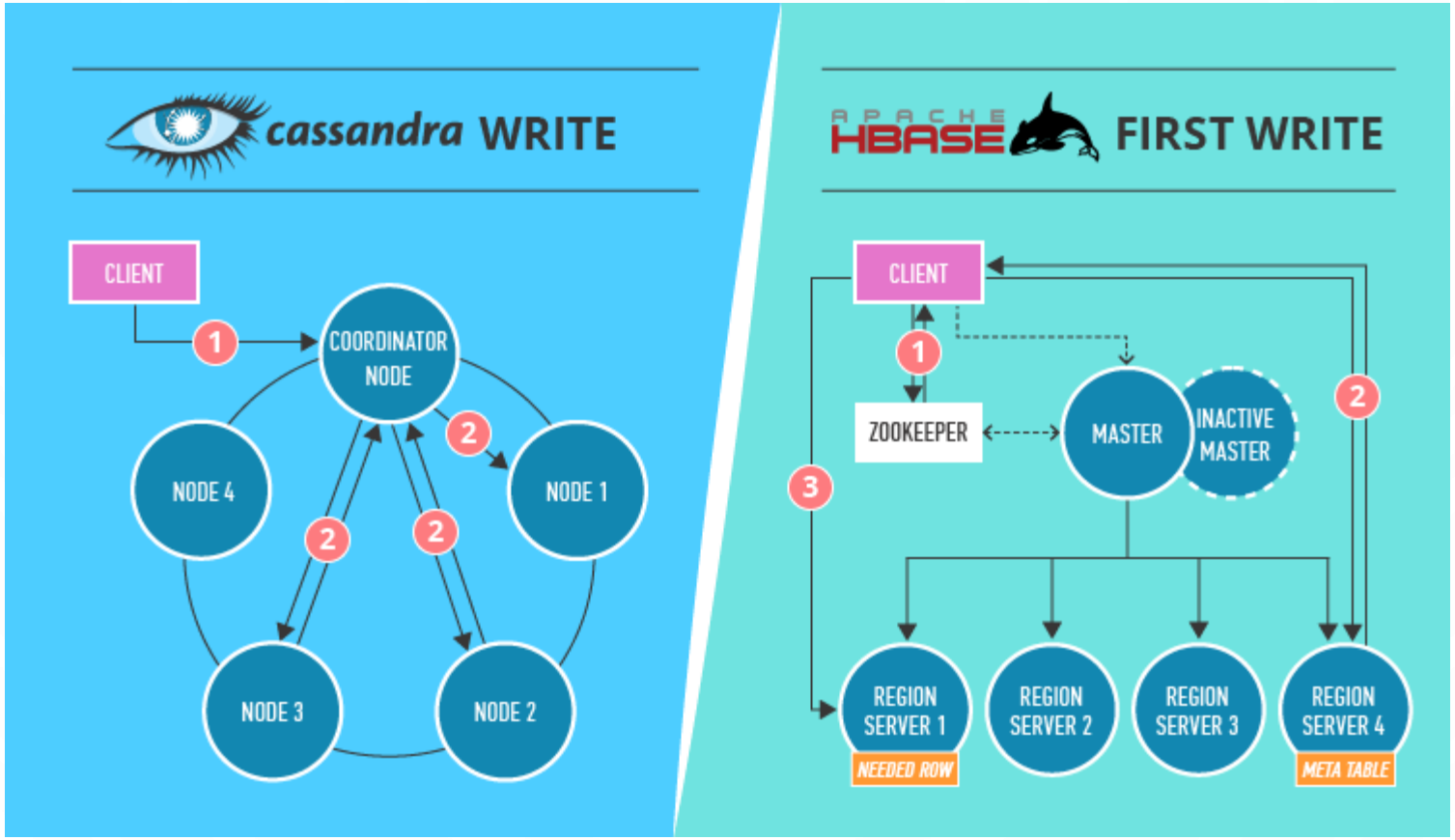
Hive: tradutor sql->java (mapreduce)
Pig: tradutor para o mapreduce, mas é linguagem específica, não muito usado
Apache zookeeper – controla aplicações
Ambari – admin do Hadoop (permissões)
Sqoop – transferência hadoop para outras bases
Spark – ideia do hadoop, mas funciona em memória

MapReduce: simples, atrativo; porém), este modelo apresenta sobrecarga – busca baseada em DISCO para Aprendizagem de máquina e análise de grafos, que processam
de modo iterativo, ou seja, muitos processamentos no mesmo conjunto de dados (lê uma vez, mas faz vários processamentos sobre os mesmos.
Alternativa **SPARK**.



ECOSSISTEMA HADOOP

Exemplo de implementações bancos de dados distribuídos



ECOSSISTEMA HADOOP - SPARK

Spark: O Spark é um framework para clusterização que executa processamento em memória - sem utilização de escrita e leitura em disco rígido - com o objetivo de ser superior aos motores de busca baseados em disco como o MapReduce.
Para superar as limitações do MapReduce descritas anteriormente, o Spark faz uso de Datasets Resilientes Distribuídos (RDDs) o qual implementa estruturas de dados em memória e que são utilizadas para armazenar em cache os dados existentes entre os nós de um cluster.
Uma vez que as RDDs ficam em memória, os algoritmos podem interagir nesta área de RDD várias vezes de forma eficiente.

Ver estudo em Um comparativo entre MapReduce e Spark para analise de Big Data (infoq.com) para tarefa de contagem de palavras e de ordenação

Table 3: Overall Results: Word Count

Platform	Spark	MR	Spark	MR	Spark	MR
Input size (GB)	1	1	40	40	200	200
Number of map tasks	9	9	360	360	1800	1800
Number of reduce tasks	8	8	120	120	120	120
Job time (Sec)	30	64	70	180	232	630
Median time of map tasks (Sec)	6	34	9	40	9	40
Median time of reduce tasks (Sec)	4	4	8	15	33	50
Map Output on disk (GB)	0.03	0.015	1.15	0.7	5.8	3.5

Table 7: Overall Results: K-Means

Platform	Spark	MR	Spark	MR	Spark	MR
Input size (million records)	1	1	200	200	1000	1000
Iteration time 1st	13s	20s	1.6m	2.3m	8.4m	9.4m
Iteration time Subseq.	3s	20s	26s	2.3m	2.1m	10.6m
Median map task time 1st	11s	19s	15s	46s	15s	46s
Median reduce task time 1st	1s	1s	1s	1s	8s	1s
Median map task time Subseq.	2s	19s	4s	46s	4s	50s
Median reduce task time Subseq.	1s	1s	1s	1s	3s	1s
Cached input data (GB)	0.2	-	41.0	-	204.9	-

Table 5: Overall Results: Sort

Platform	Spark	MR	Spark	MR	Spark	MR
Input size (GB)	1	1	100	100	500	500
Number of map tasks	9	9	745	745	4000	4000
Number of reduce tasks	8	8	248	60	2000	60
Job time	32s	35s	4.8m	3.3m	44m	24m
Sampling stage time	3s	1s	1.1m	1s	5.2m	1s
Map stage time	7s	11s	1.0m	2.5m	12m	13.9m
Reduce stage time	11s	24s	2.5m	45s	26m	9.2m
Map output on disk (GB)	0.63	0.44	62.9	41.3	317.0	227.2

Após nosso estudo de pandas, retorna-se ao Apache Spark (pySpark), HDFS, Hive, Hadoop e exemplos de uso em python