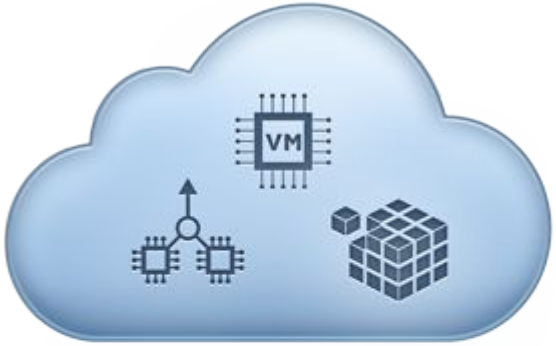
Computação em nuvem

Tecnologias de Suporte à Computação em Nuvem



Prof. Dr. Marcos A. Simplicio Jr.
Laboratório de Arquitetura e Redes de Computadores
Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo

Objetivos – Aula 20

- Compreender o conceito de "**big data**" e como a abordagem **MapReduce** pode resolver problemas nesse cenário

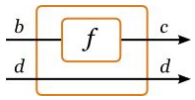
"Big Data": Contexto

- Mineração de dados em **bases de dados gigantescas**
  - Ex.: a Web inteira
- Dados **não estruturados**
  - Ex.: texto simples em páginas web; informações em imagens
- **Grande número de máquinas** disponíveis
  - Devem ser usadas de forma eficiente
- Há necessidade de **soluções específicas** de:
  - **Modelos de programação** para processamento paralelo
  - **Algoritmos, linguagens e estruturas de dados** para facilitar processamento paralelo

"Big Data": Abordagem geral

- **Minimizar movimentação de dados**
  - **Dados distribuídos** pelos diversos clusters de processadores

  - **Mover algoritmo** para onde estão dados: melhor desempenho, pois algoritmo é pequeno em relação aos dados em si
- Modelos de **programação em alto nível**



  - Orientação a **fluxo de dados**: ao invés de operações de baixo nível que dependem de máquina e localização de dados
- **Resistência a falhas**

  - Muitos discos/nós: **falhas são regra, não exceção**
  - Redundância, armazenamento de resultados intermediários, detecção e recuperação automática de falhas, ...
- **Escalabilidade e alocação dinâmica** de recursos

"Big Data": MapReduce

- Modelo de programação para processamento paralelo de grandes conjuntos de dados

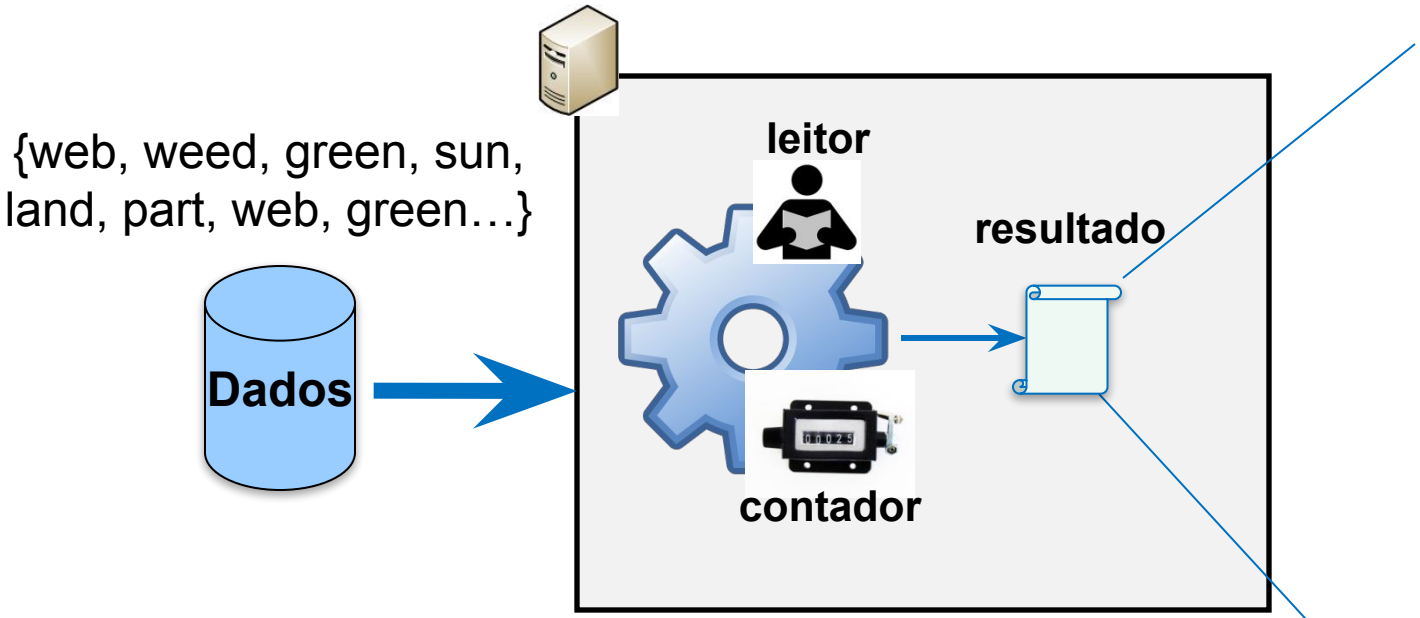
  - Computação definida por duas funções: **map** e **reduce**;
  - Sistema subjacente automaticamente **distribui** processamento paralelo entre diferentes máquinas
  - Sistema subjacente também lida com **falhas**, **comunicação** eficiente e questões de **desempenho**.

  - Ref: Dean, J. and Ghemawat, S. 2008. MapReduce: simplified data processing on large clusters. Communication of ACM 51, 1 (Jan. 2008), 107-113.

"Big Data": MapReduce

- Para entender a ideia por trás do mecanismo, é mais fácil analisar um **exemplo**
- Cenário: considere um grande conjunto de dados:
  - Palavras: {web, casa, verde, sol, lua, hoje, web, verde,...}
- Problema: **contar a ocorrência** de cada palavra no conjunto
- Vamos começar com algumas solução simples...

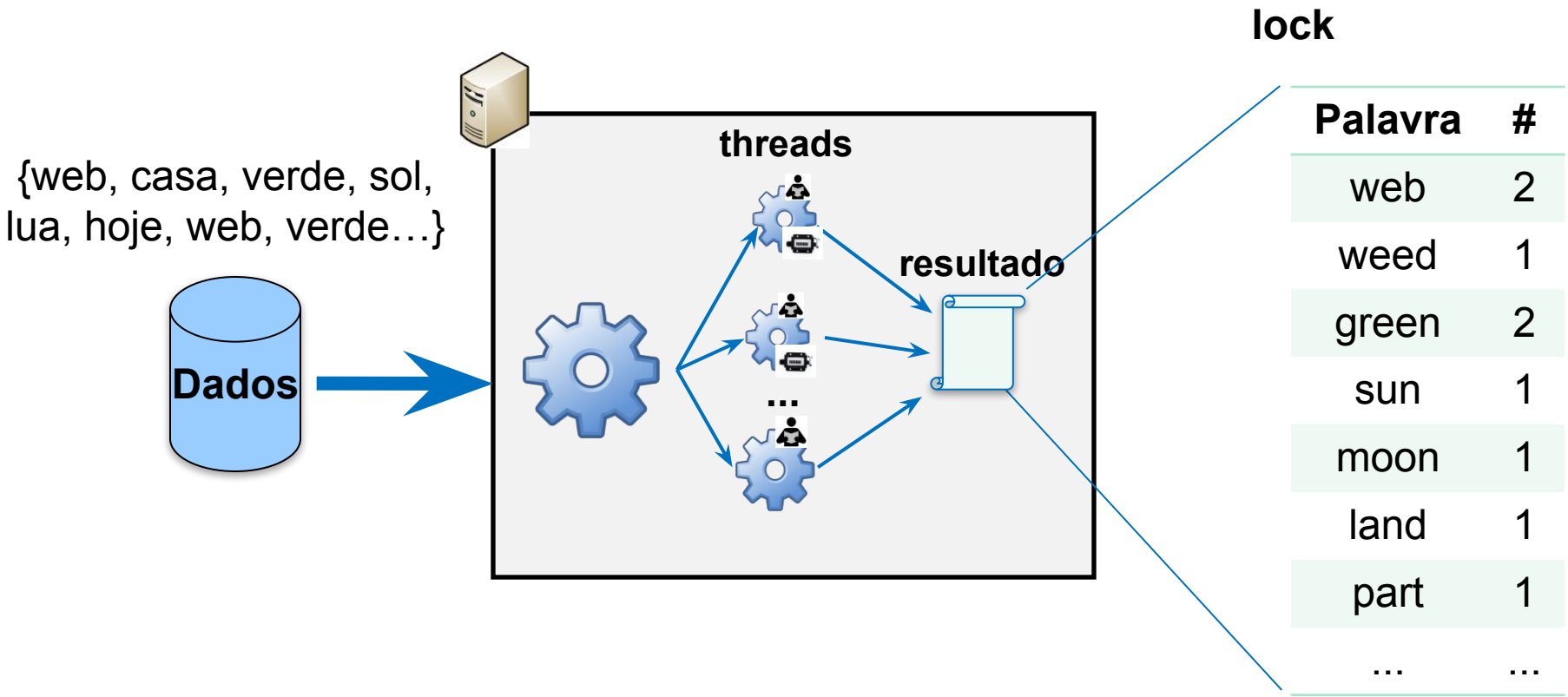
"Big Data": Solução simples



Palavra	#
web	2
weed	1
green	2
sun	1
moon	1
land	1
part	1
...	...

- Sem paralelismo...
  - Mas e se usarmos vários threads?

"Big Data": Multithread e locks

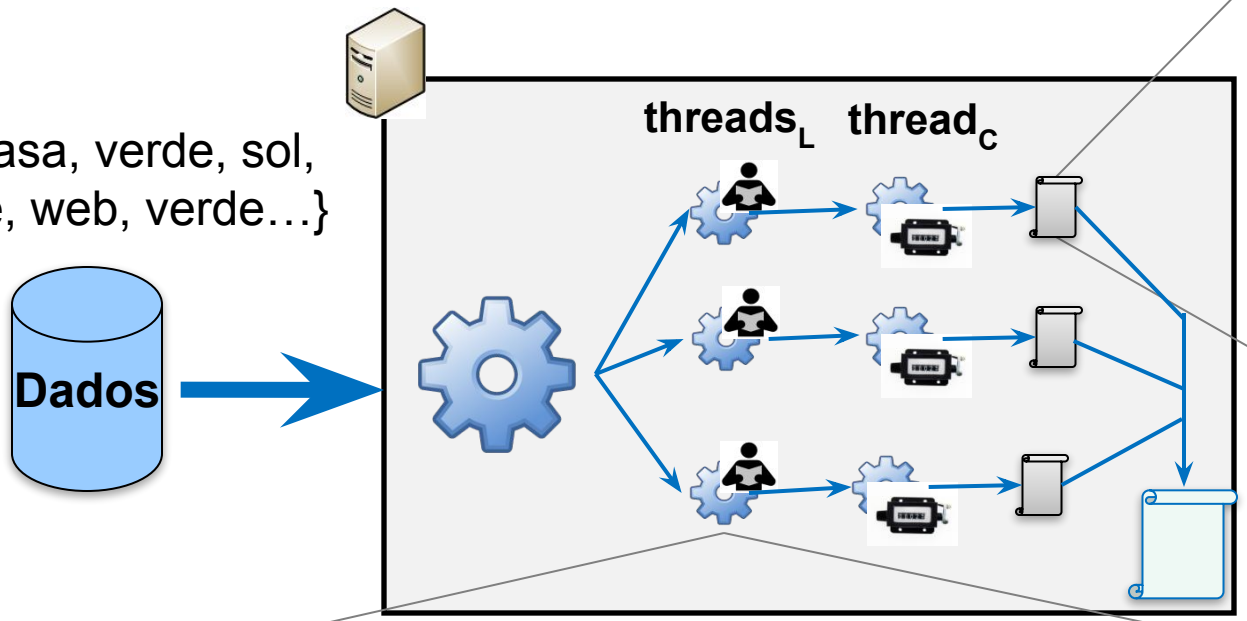


Palavra	#
web	2
weed	1
green	2
sun	1
moon	1
land	1
part	1
...	...

- Vários threads acessando mesmo conjunto de dados:
  - É necessário algum mecanismos para evitar inconsistências (ex.: locks).
  - Não daria para fazer algo melhor...?

"Big Data": Multithread sem locks

{web, casa, verde, sol, lua, hoje, web, verde…}



Palavra	#
web	2
casa	1
verde	2
sol	1
lua	1
hoje	1
...	...

resultado parcial da contagem

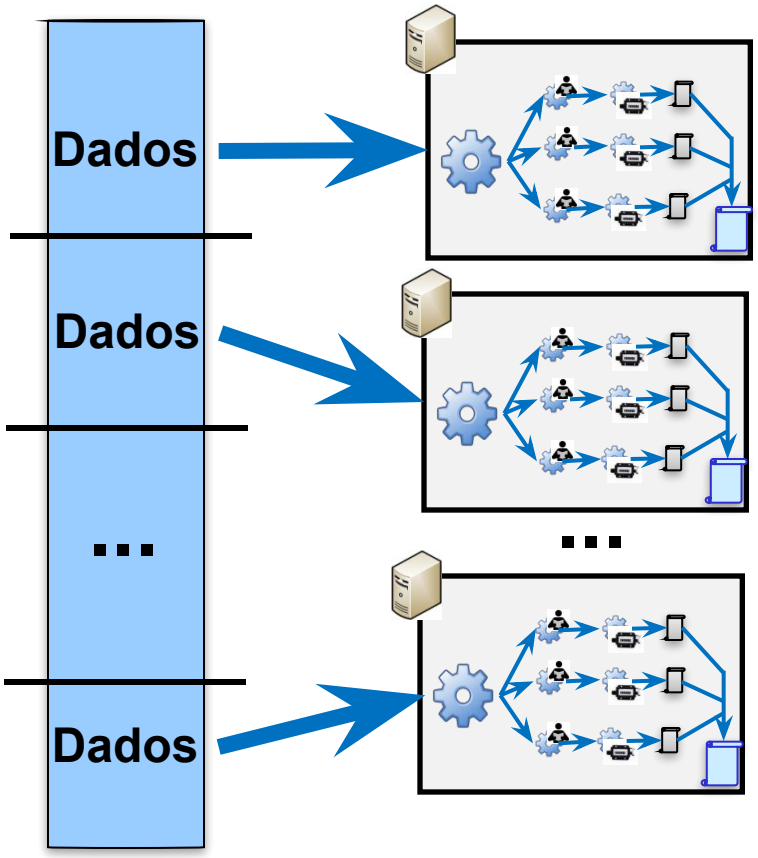
Chave	web	casa	verde	sol	lua	hoje	web	verde	...
Valor	1	1	1	1	1	1	1	1	

Mapa de palavras lidas

Valor:
"1 palavra lida"

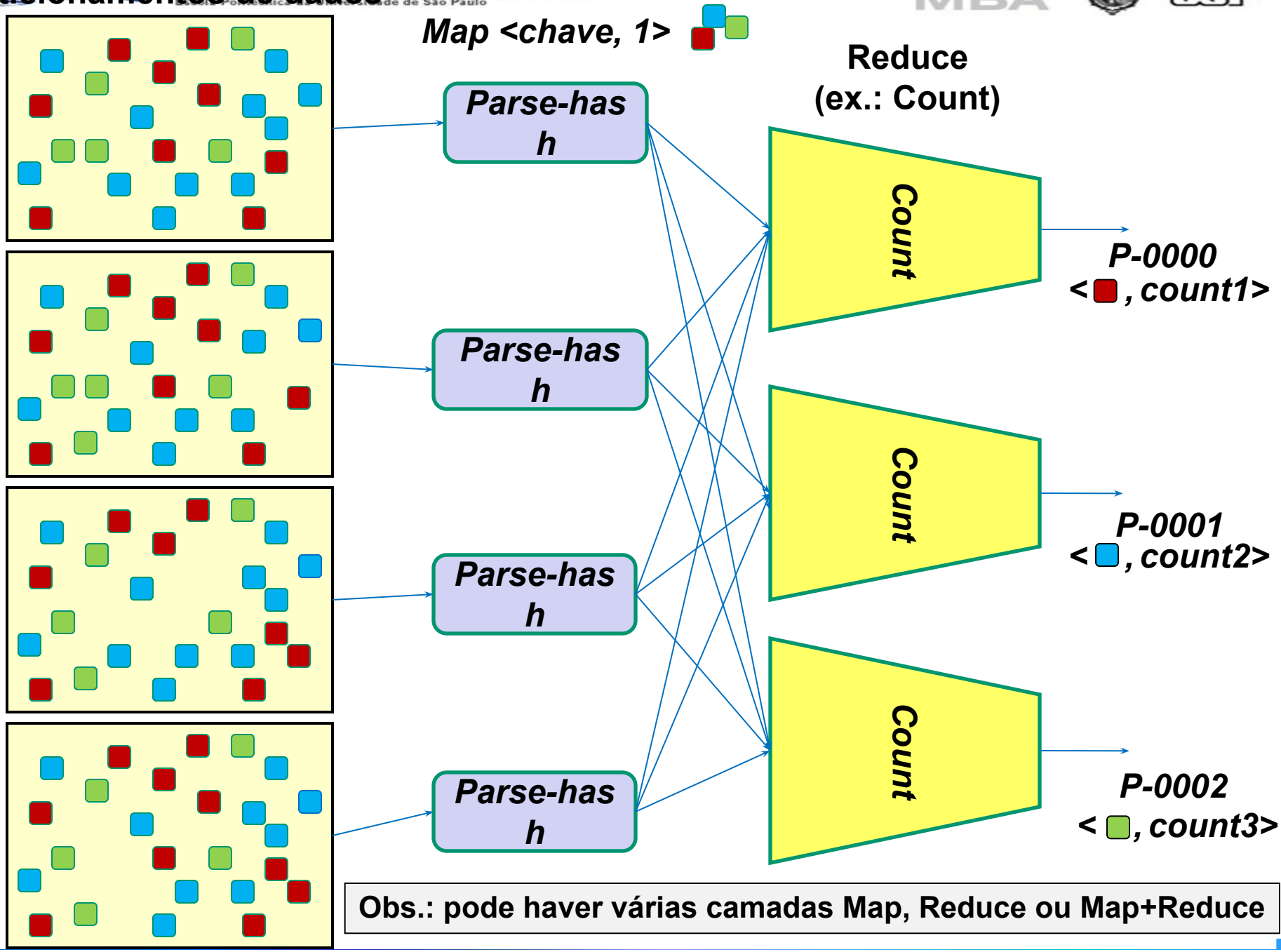
- E se a quantidade de dados é gigantesca?

"Big Data": WORM e MapReduce



- WORM: **Write Once Read Many**
  - Fácil de paralelizar
  - Se dados independentes: podem ser processados em qualquer ordem
- No exemplo:
  - #1: Leitura (parse) em paralelo
  - #2: Contagem em paralelo
- Generalizando: **MapReduce**
  - #1: Parse = **mapear**
    - MAP: entrada → pares <chave, valor>
  - #2: Contagem = **reduzir**
    - REDUCE: pares <chave, valor> reduzidos

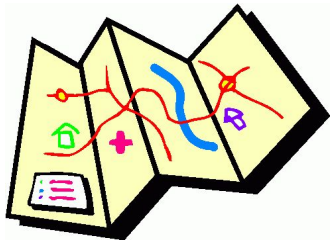
Particionamentos de dados



Obs.: pode haver várias camadas Map, Reduce ou Map+Reduce

"Big Data": Exemplo de MapReduce

- **Entrada do Map**: processar as seguintes páginas
  - Página 1: "o dia é bom"
  - Página 2: "hoje é bom"
  - Página 3: "dia bom é bom"



- **Saída do Map:**
  - $Thread1_M$: (o 1), (dia 1), (é 1), (bom 1).
  - $Thread2_M$: (hoje 1), (é 1), (bom 1).
  - $Thread3_M$: (dia 1), (bom 1), (é 1), (bom 1).

"Big Data": Exemplo de MapReduce (cont.)

- **Entrada do Reduce:** saídas do Map, já distribuídas
  - $Thread1_R$: (o 1)
  - $Thread2_R$: (dia 1), (dia 1)
  - $Thread3_R$: (é 1), (é 1), (é 1)
  - $Thread4_R$: (hoje 1)
  - $Thread5_R$: (bom 1), (bom 1), (bom 1), (bom 1)
- **Saída do Reduce:**
  - $Thread1_R$: (o 1)
  - $Thread2_R$: (dia 2)
  - $Thread3_R$: (é 3)
  - $Thread4_R$: (hoje 1)
  - $Thread5_R$: (bom 4)

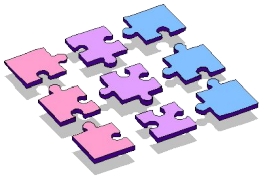
"Big Data": Uso do MapReduce

- Usado pelo **Google** para **contagem** de pala… **ranking** de páginas, **indexação** de dados, ...
- Usado pelo **Facebook** para várias operações, como **estatísticas demográficas**
- Construção de algoritmos simples de **edição de texto**

  - Ex.: **contagem/busca/substituição** de palavras/caracteres, geração de **índice remissivo**, etc.
- **Mineração** de dados (ex.: análise de dados financeiros ou busca de **objetos em fotos**)
  - Espera-se que tal operação permita **buscas semânticas** (significado em vez de palavra) na Web: a chamada "Web 3.0"

Resumo

- Compreender o conceito de "big data" e como a abordagem MapReduce pode resolver problemas nesse cenário
  - Big data: grande quantidade de dados pouco estruturados ➔ como extrair informações úteis de forma eficiente?
  - MapReduce: permite paralelizar processamento de dados
    1. **Distribuir** dados entre processos
    2. **Map**: entrada → pares <chave, valor>
    3. **Reduce**: processar localmente pares <chave, valor>, retornando resultados
    4. **Agregar** saídas dos processos de *reduce*



- Próxima aula: Google File System & Hadoop