MapReduce

MAC5742/0219 Introdução à Programação Concorrente, Paralela e Distribuída

Felipe e Mateus

06/06/17

O que é?

- Modelo de programação
- Processamento de grande conjuntos de dados
- Algoritmo paralelo e distribuído
- Redundância e tolerância a falhas

História

- Inspirado nas operações de Map e Reduce
- Referência à tecnologia proprietária da Google
- Implementada em várias linguagens de programação
- Parte do Apache Hadoop (open source)
- Não é mais o modelo principal do Google desde 2014

Modelo

- Recebe um conjunto de pares chave-valor
- Mapeia para pares intermediários

map

- Agrupa pares com as mesmas chaves

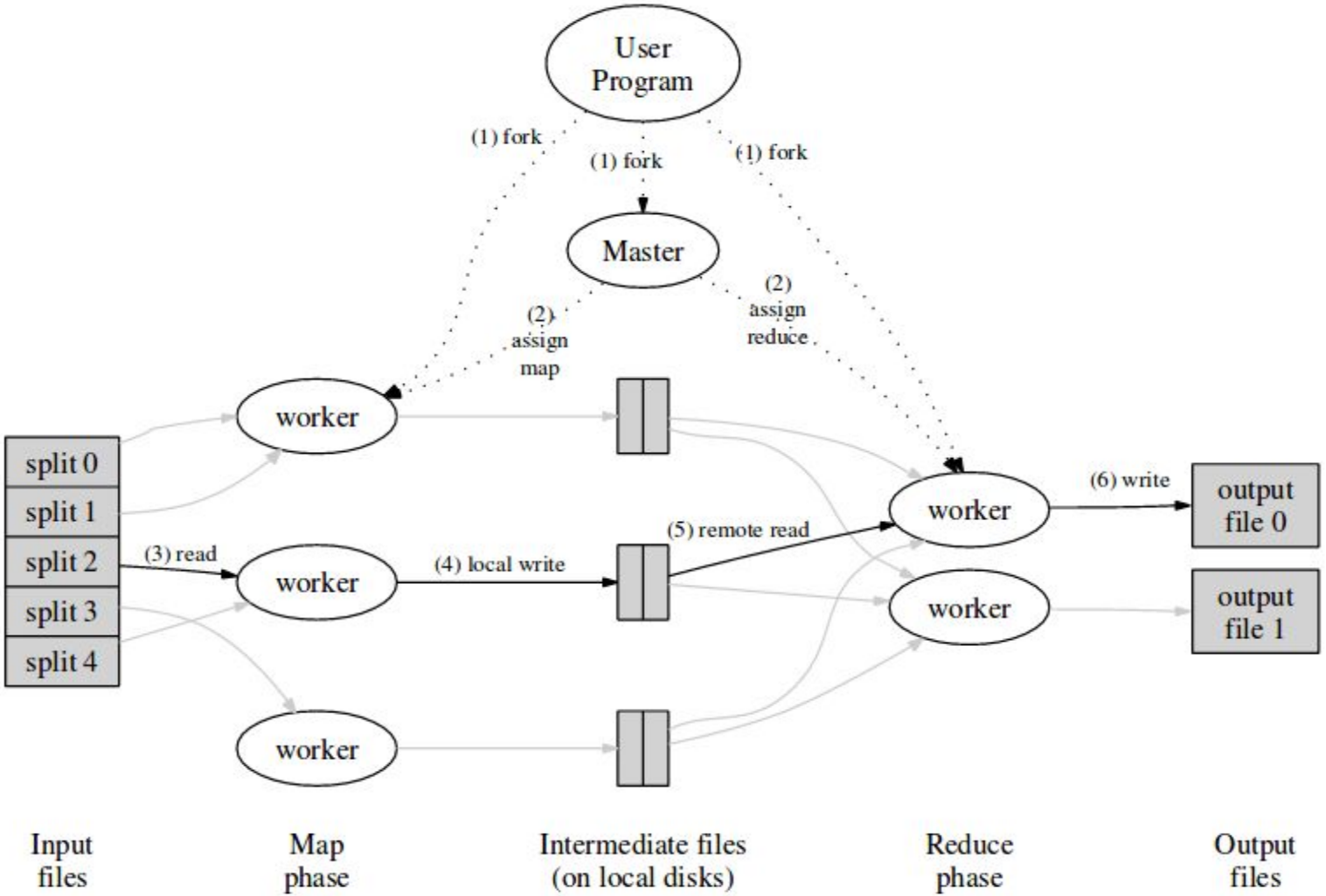
- Recebe uma chave e seus valores
- Processa os pares que possuem a mesma chave
- Usualmente retorna um único valor para cada chave

reduce

Modelo

```
function map(String name, String document):
  for each word w in document:
    emit (w, 1)

function reduce(String word, Iterator partialCounts):
  sum = 0
  for each pc in partialCounts:
    sum += pc
  emit (word, sum)
```



Retirada de: “MapReduce: Simplified Data Processing on Large Clusters”

Tolerância a falhas

Falha no *worker:*

- Verifica estado do *worker*
- Re-executa em outro *worker*

Falha no *master:*

- Pouco provável
- Aborta toda a execução

Localidade

- Divide arquivo de entrada em blocos (GFS)
- Armazena cópias dos blocos em diferentes máquinas
- Agenda a execução na máquina que contém o bloco
  - Ou em uma máquina da mesma rede local

Localidade

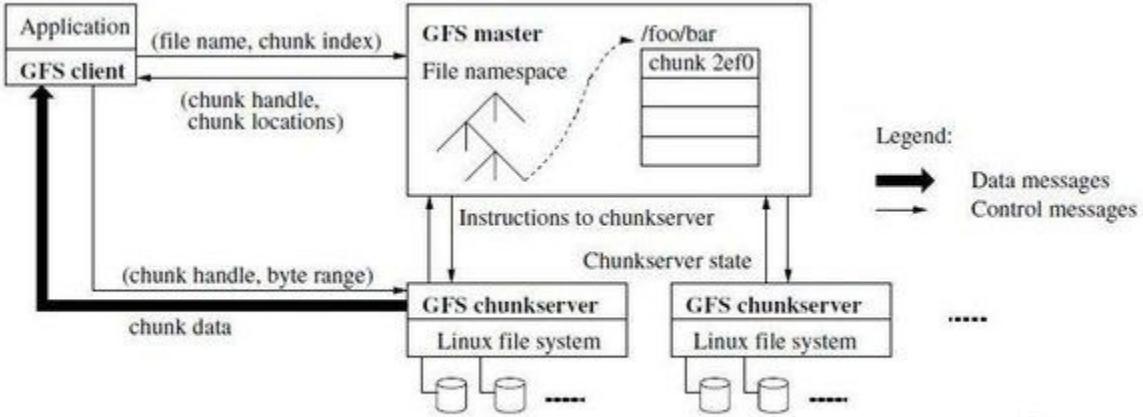
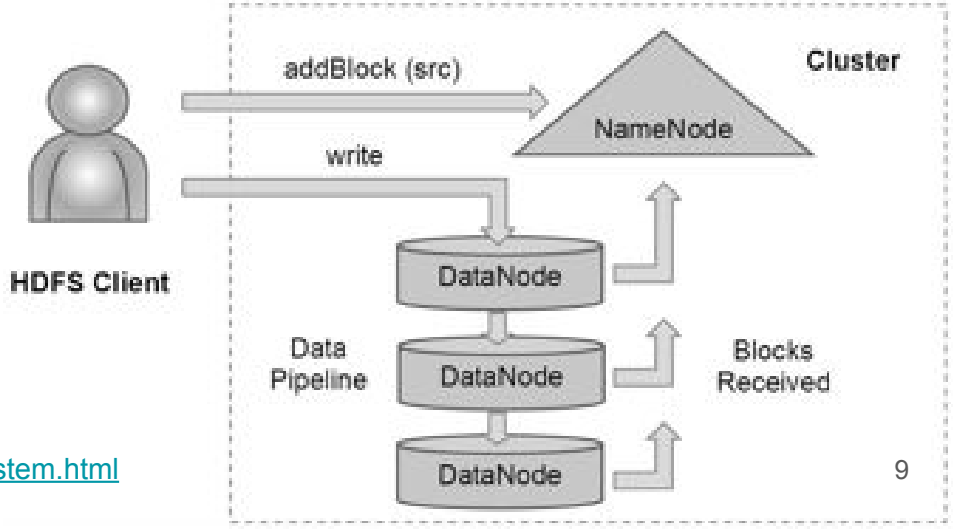


Figure 1: GFS Architecture



Retiradas de:
http://cs.uwec.edu/~buipj/teaching/cs.491.s13/lecture_06_google_file_system.html
http://www.aosabook.org/en/hdfs.html

Granularidade

- Divisão do arquivo de entrada em M partes no Map
- Mapeamento em R partes no Reduce
- Quanto maior o valor de M e R:
  - Maior granularidade das tarefas
  - Menos trabalho é perdido em caso de falha
  - Maior complexidade no master para agendar os M+R trabalhos
- Map >> Reduce > Workers

Tarefas de backup

- Podem ocorrer atrasos quando tiver perto do fim
- Execuções reservas das tarefas em progresso
- Quem acabar primeiro é escolhido

Refinamentos

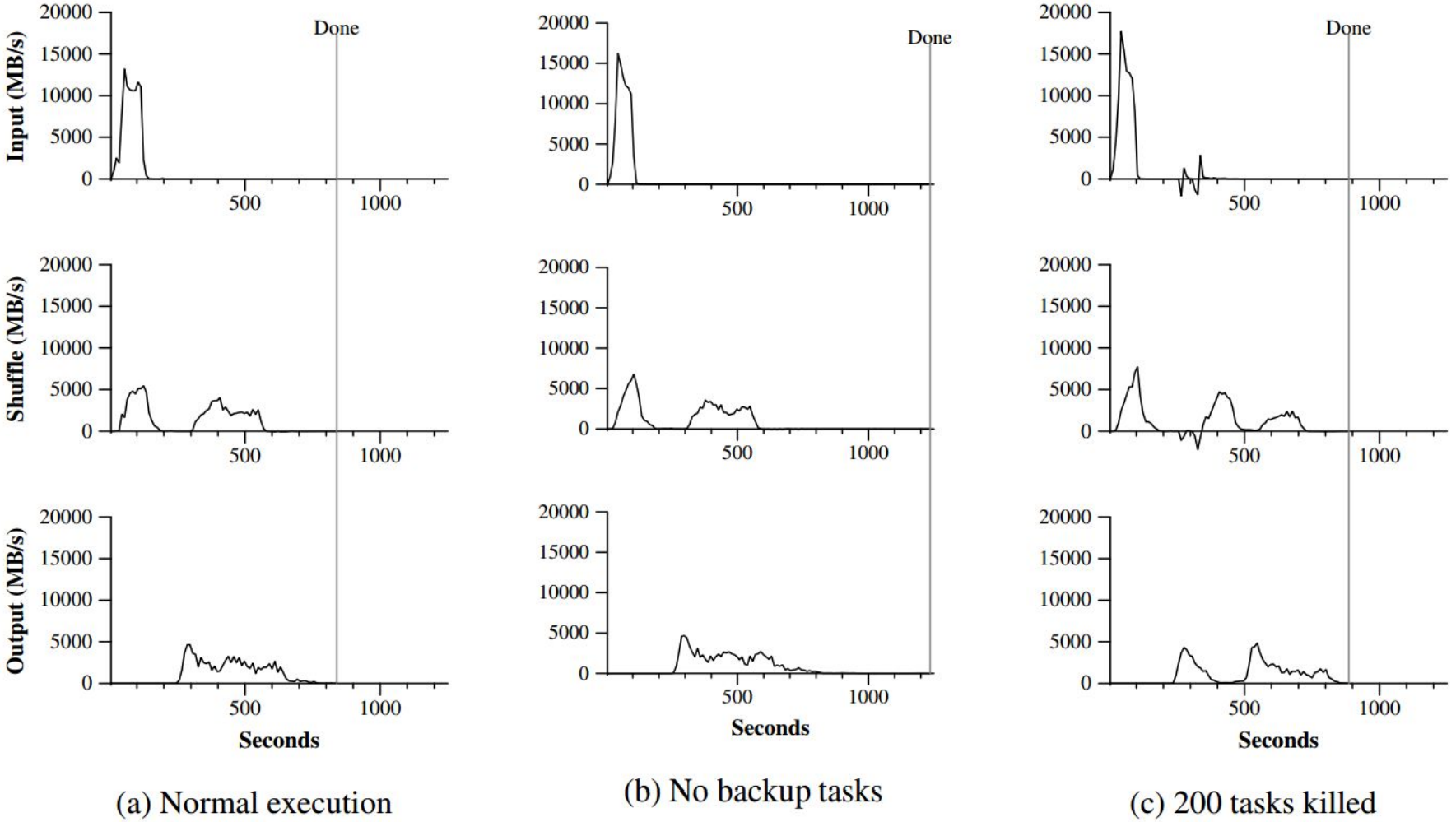
- Utilização de hashes
  - Partições equilibradas
- Garantia de ordem
  - Saída é ordenada
- Função de combinação
  - Função intermediária para reduzir o número de pares

Refinamentos

- Entrada e saída flexíveis
- Possibilidade de ignorar registros ruins
- Monitoramento da execução
- Execução local
  - Opção de executar sequencialmente

Desempenho

- 1800 Máquinas
  - 2GHz Intel Xeon Dual Core com Hyper-Threading
  - 4GB de memória
  - 160GB de disco
- 100-200Gbps de largura de banda agregada
- Experimento realizando a ordenação de 10^{10} registros de 100 bytes (1 TB aproximadamente)



(a) Normal execution

(b) No backup tasks

(c) 200 tasks killed

Retirada de: MapReduce: Simplified Data Processing on Large Clusters

Críticas

- DeWitt e Stonebraker's "MapReduce: A major step backwards"
  - “MapReduce is a step backwards in database access”
  - “MapReduce is a poor implementation”
  - “MapReduce is not novel”
  - “MapReduce is missing features”
- Em resposta Greg Jorgensen disse que o MapReduce nunca foi feito para ser um banco de dados.

Spark

- Trabalhos iterativos
- Resilient Distributed Datasets
- Agregação baseada em hash
- Map dependente de CPU
- Reduce dependente de rede

Referências

- MapReduce: Simplified Data Processing on Large Clusters
- MapReduce: A major step backwards
- Um comparativo entre MapReduce e Spark para analise de Big Data