AN IN-DEPTH STUDY OF MAP REDUCE IN CLOUD ENVIRONMENT

Sistemas Distribuídos e Tolerância a Falhas

Nuno Garcia m6284, Tiago Carvalho m6294

Sobre o *paper*…

Novia Nurain, Hasan Sarwar, Md.Pervez Sajjad
United International University, Bangladesh

Moin Mostakim
Bangladesh University of Engineering and Technology

Advanced Computer Science Applications and Technologies (ACSAT), 2012 International Conference

Introdução

- MapReduce (Google)
- Cloud Computing
- Arquitectura da Google
- Arquitectura da MS (Azure)
- Arquitectura da Amazon (Cloud MR)

Motivação

- Comunidade científica depende cada vez mais do processamento de grandes quantidades de dados.
- Grande tolerância a falhas e escalabilidade.
- Vantagens conhecidas de Cloud Computing

MapReduce

- Introduzido pela Google em 2003
- Tolerância a falhas
- Paralelização automática
- Escalabilidade
- Optimizações a nível local de gestão de dados

Execução do MapReduce

- Pedaços de dados (chunks) são distribuídos para tarefas Maps através de um sistema de ficheiros distribuído.
- Chunks -> sequência de pares chave-valor; escrita para disco local, particionados em R (# tarefas Reduce) regiões.

Execução do MapReduce

- Master envia localização das regiões para as tarefas Reduce.
- Tarefas Reduce encarregam-se de combinar os valores de cada chave.

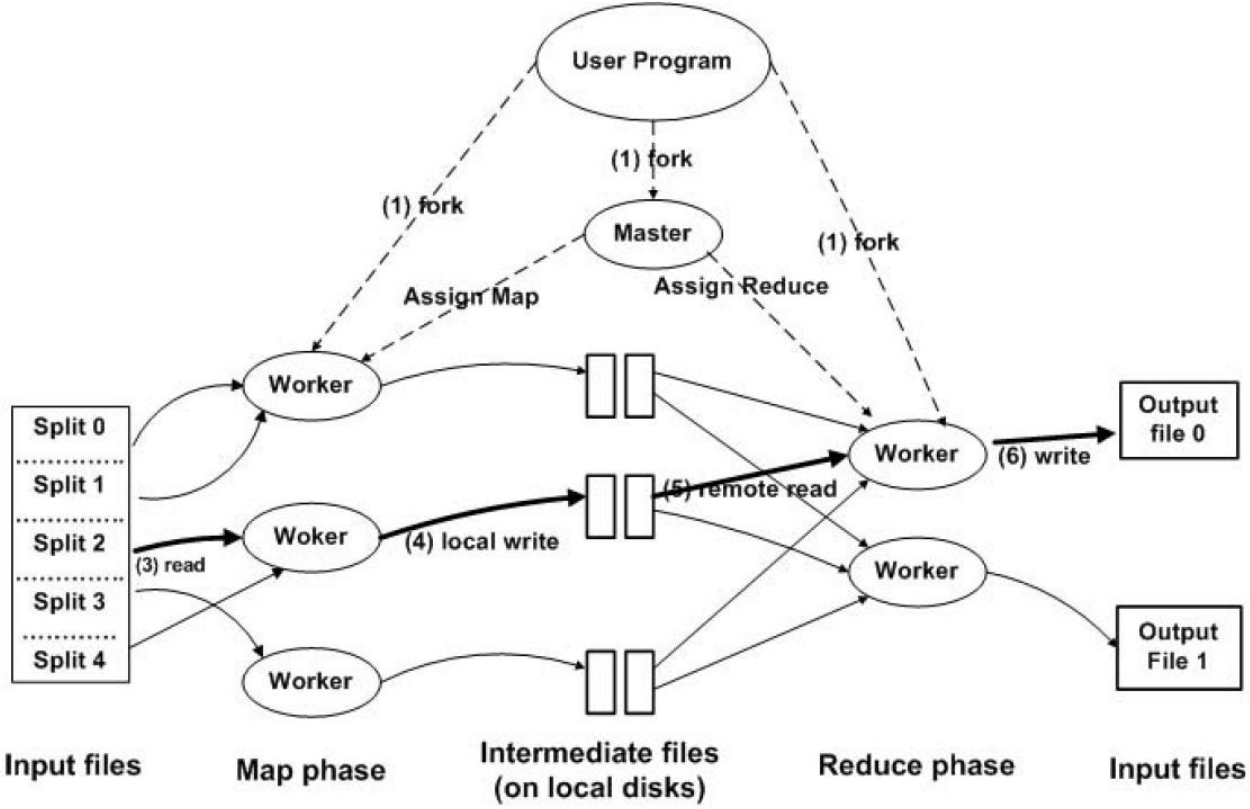
Desafios do MapReduce na Cloud

- Armazenamento de Dados e Metadados: a performance é afectada pela localização dos dados e pela largura de banda.
- Consistência da comunicação e escalabilidade: flutuação na performance de I/O (topologia desconhecida, etc…)
- Fiabilidade do Master: Toda a lógica depende da capacidade do Master não sofrer falhas.

Factores de Performance

- I/O system: I/O directo (melhor) ou Streaming.
- Algoritmo de escalonamento: distribuir as tarefas Map pelos nós.
- Tolerância a Falhas.

MapReduce para Google Large Clusters



Google MR

- Dividir ficheiros de entrada.
- Master distribui cada chunk para um worker disponivel.
- Cada mapworker faz o parse do chunk para pares chave/valor e aplica a função Map definida.
- Os pares são divididos em regiões, cujas localizações são enviadas para o Master. O Master envia estas localizações para os reduce workers.

Google MR

- Os reduce workers lêem os dados dos discos dos map workers, e agrupam por chave.
- Os reduce workers enviam para a função Reduce os pares chave/valores intermediários.
- Quando todas as tarefas acabam, o Master acorda o programa.
- O output está disponível nos R ficheiros (# reduce workers).

Tolerância a Falhas

- O Master faz ping ao worker periodicamente. Se não receber resposta, marca o worker como “failed”.
- Quando os workers terminam as tarefas, são marcados como “idle” e disponiveis para novo escalonamento.
- Qualquer tarefa num worker “failed” é marcada como idle e disponivel para re-escalonamento.

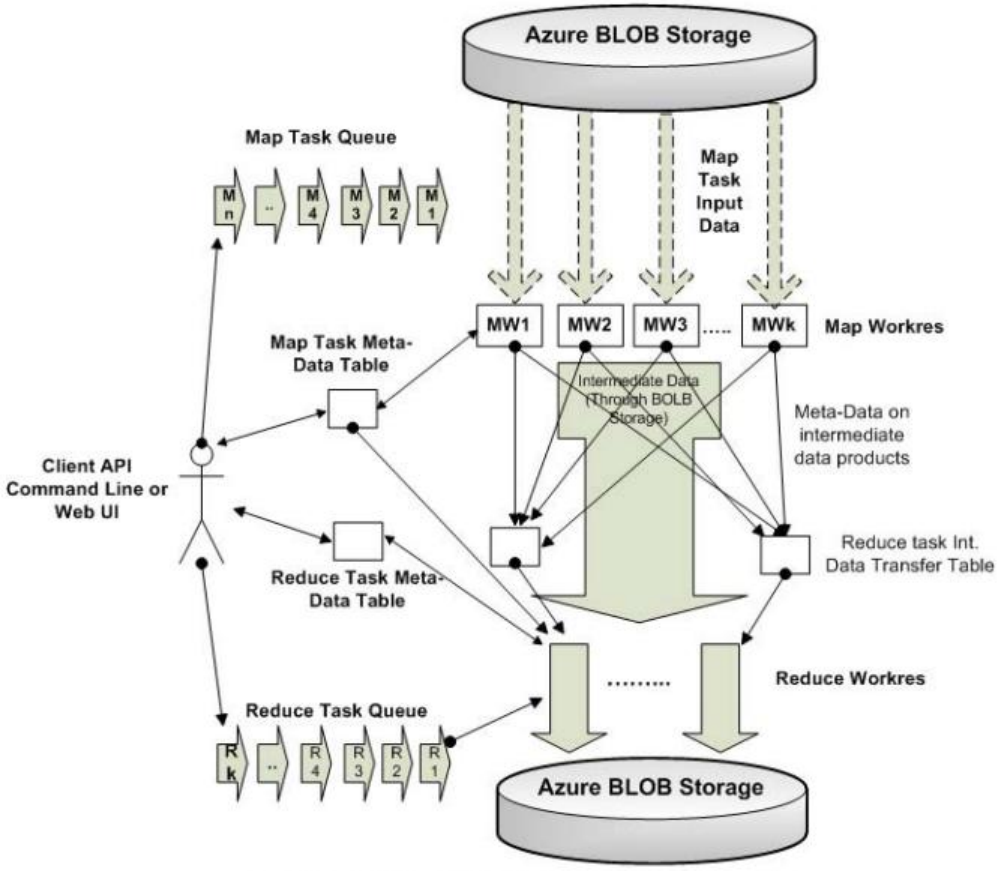
Perigos

- É uma tecnologia proprietária.
- Se o Master falha, tudo falha.
- Não há nenhuma garantia que seja a melhor.

Azure MR

Arquitectura

- Azure queues
- Azure tables
- Azure blob storage



Azure MapReduce – Vantagens

- **Tolerância a falhas:** as tarefas são removidas das *task queues* apenas após serem concluídas;

- **Resiliente a falhas do *Master*:** devido à inexistência de um *Master* que distribuí tarefas, é impossível cancelar todo o processo de MapReduce.

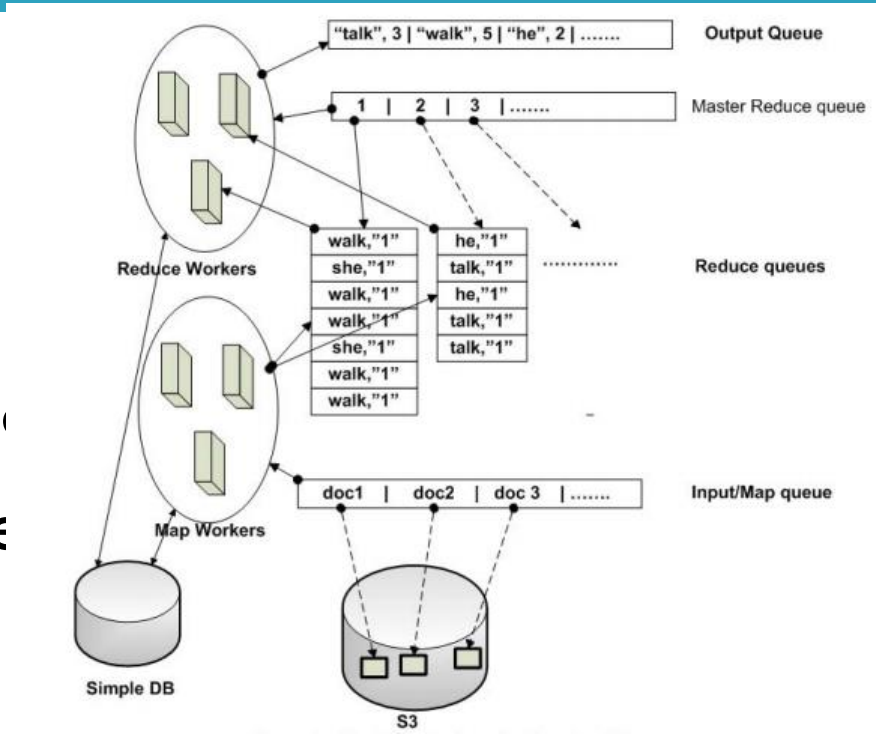
Azure MapReduce - Desvantagens

- **Tempos de *queue* fixos**: não permite alterar o limite de 2h para *timeout* das *queues*. O que se torna bastante limitador em grandes cálculos de MapReduce;
- **Falta de optimização local**: não há verificação se os dados existem localmente, é feito o download dos dados da *blob storage*, seguido pelo upload dos resultados;
- **Inexistência de uma forma directa para transmitir dados entre as tarefas de Map e Reduce.**
- **Dependência de tabelas de metadados**: permite remover o único mestre da arquitectura, mas fica dependente de guarder os dados fisicamente.

Cloud MapReduce (CMR)

Arquitectura

- Input Queue
- Output Queue
- Master Reduce Qu
- Várias Reduce Que



Cloud MapReduce - Vantagens

- **Escalabilidade Incremental:** possibilidade de adicionar servidores durante cálculos;
- **Simetria e descentralização:** todos os nós de computação têm o mesmo trabalho;
- **Nós heterogénios:** cada nó pode ter um desempenho diferente e ter uma localização geográfica diferente;
- **Troca de dados mais eficiente** devido ao uso de uma queue intermediária entre Map e Reduce.

Cloud MapReduce - Desvantagens

- **Falta de optimização local:** a rede é usada exclusivamente para I/O, ignorando qualquer cache local. Possível estrangulamento quando as ligações entre nós estão saturadas.

Comparação de Serviços

Name of Architecture	Pros	Cons
Generalize MapReduce [2]	Silmple and powerful interface Enables automatic parallelization Resilient to worker failure	Not resilient to master failure Patent Risk Lack of Architectural Exploration
AzureMapReduce [5]	Fault Tolarent Resilient to Master failure	Queue message timeout not enough Donnot allow for dynamic changes Lack of locality optimization Indirect data parsing Table dependacy of metadata
CloudMapReduce [7]	Incremental Scalability Symmetry and Decentralization Heterogeneity More efficient data shuffling between Map and Reduce	Donot employ locality optimization

Conclusão

- Ainda há espaço para optimizar o agendador de tarefas;

- Não há uma melhor solução implementada – cada solução tem os seus prós e contras, e cada uma tem o seu caso ideal de uso, que será diferente de outra solução