Prova: Inteligência na Web e Big Data

Eduardo Pinhata

May 4, 2018

Descreva em uma página os conceitos de programação funcional necessários para o desenvolvimento de algoritmos distribuídos para trabalhar com grande massa de dados.

O paradigma funcional é baseado em alguns conceitos, entre eles:

- **Imutabilidade:** nenhuma variável, estrutura de dados, função muda após ter sido criada. Caso seja necessário alterá-las, uma nova variável, estrutura de dados, funçãos, etc é criada.
- Uma mesma função com os mesmos argumentos, produzem o mesmo resultado: como não há mutabilidade das funções, variáveis, estruturas, uma função não pode produzir resultados diferentes dadas as mesmas entradas.
- Ausência de efeitos colaterais: Isto está associado à imutabilidade. Como nada é alterado, funções não geram efeitos como por exemplo, alterar variáveis globais, que poderiam gerar resultados diferentes.

No contexto de algoritmos distribuídos para trabalhar com grande massa de dados, essas três propriedades garantem que quando uma função é executada sobre um conjunto de dados, este não será alterado. Desta forma, outra função pode ser aplicada ao mesmo conjunto de dados paralelamente sem a preocupação de gerar inconsistência. Uma mesma função pode até ser aplicada a diversas porções do mesmo conjunto de dados. Essas propriedades fazem com que a paralelização em programação funcional seja algo natural.

Outra vantagem é que a programação funcional avalia funções ao invés de executar instruções. Isto associado à avaliação preguiçosa permite economizar processamento, principalmente quando a base de dados é muito grande. Um exemplo seria realizar uma grande quantidade de processamento para chegar em um valor e por fim multiplicá-lo por zero. Em uma avaliação preguiçosa, é possível não realizar a grande quantidade de processamento.

Um modelo de programação utilizado em programação paralela é o *MapReduce*. este modelo é utilizado para processar uma grande quantidade de dados e se baseia em aplicar uma mesma função em cada elemento de um conjunto de dados (*Map*) ou combinar os elementos do conjunto de dados utilizando outra função (Reduce), reduzindo esses elementos em um único elemento.

As funções *Map* e *Reduce* não dependem da ordem em que os elementos estão dispostos. A função utilizada com o *Reduce* precisa ser comutativa, pois não há uma ordem específica para agrupar os dados. Estas propriedades fazem com que *Map* e *Reduce* sejam altamente paralelizáveis.

References

[1] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, OSDI'04, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[2] Claudio Cesar de Sá and Márcio Ferreira da Silva. *Haskell Uma Abordagem Prática*. Editora Novatec, first edition, 2006.

[3] Haskell basics, jan 2013.