CCM-206 - INTELIGÊNCIA NA WEB E BIG DATA

Bayard da Rocha – RA : 21201810004 – 2018.1

PROVA - Questão teórica: *"Conceitos de programação funcional necessários para o desenvolvimento de algoritmos distribuídos para trabalhar com grande massa de dados"*

Um dos desafios em projetos voltados à Ciência de Dados, Big Data e Inteligência Artificial é otimizar tempo do processamento de grande massa de dados; um ponto crítico nestes projetos. O desenvolvimento de lógica de programação no modelo procedural (paradigma da programação Imperativa) para o tratamento destes dados tornou-se inviável.

A estratégia adequada para processarmos grandes volumes de dados é utilizando a capacidade de processamento dos computadores via processamento paralelo que consiste em programas com lógica que permitem múltiplos fluxos de execução simultâneos. No processamento paralelo, cada processo tem seu espaço de endereçamento individual e vários fluxos de execução no mesmo espaço de endereçamento, que denominamos THREADS.

Programação Funcional, um dos paradigmas de programação, caracteriza-se por desenvolver estes fluxos para o tratamento dos dados através de funções matemáticas sem mudança de estado e dados mutáveis ao invés do encadeamento de comandos no tratamento de variáveis.

No código funcional o valor de saída de uma função depende apenas dos parâmetros (argumentos) que são passados ou seja, desde que não haja alteração dos valores destes parâmetros, o resultado da função será o mesmo independente da quantidade de vezes que esta função for executada; característica importante que possibilita o processamento em paralelo destas funções.

Relaciono a seguir outras características da Programação Funcional que considero importantes em se tratando de processamento massivo de dados (não que sejam apenas estas):

a) **Funções Puras**: não possuem efeitos colaterais (memória ou E / S). Significa que funções puras têm várias propriedades úteis, muitas das quais podem ser usadas para otimizar o código. Se o resultado de uma função não for usado, ele poderá ser removido sem afetar outras expressões; e se ela for acionada novamente, retornará o mesmo valor. Com isso, pode-se habilitar otimizações de cache. Não havendo dependência de dados, duas funções puras podem ser executas em qualquer ordem sem prejuízo para o resultado final;
b) **Funções de Alta Ordem**: São funções que podem receber uma ou mais função como argumento e poderem retornar uma função como resultado. Comumente teremos estas funções a nossa disposição ao lidarmos com uma API que propõe uma fundação para a programação;
c) Recursividade: as interações em looping via de regra são resolvidas por recursão; algumas recursões exigem o que chamamos de recursão de cauda (caudal) porém esta pode ser reconhecida e otimizada pelo compilador;
d) Avaliação Preguiçosa ***(Lazy Evaluation)***: é uma estratégia de avaliação que retarda da execução de uma expressão até que seu valor seja necessário; evita avalições repetitivas. Esta característica pode reduzir o tempo de execução de determinadas funções por um fator exponencial além de reduzir otimizar os recursos em memória uma vez que os valores são criados apenas quando necessários. A linguagem Haskell utiliza *Lazy Evaluation*; não temos a garantia de que a função será executada em ordem porque o Haskell só executa o código quando for necessário;
e) Estrutura de Dados Infinita: não podemos processar uma lista infinita em um período de tempo adequado (por exemplo, a lista de Fibonacci) e nem armazenar em memória. Em uma linguagem que trabalha com Avaliação Preguiçosa, podemos abstrair e simplesmente definir uma lista infinita pois partes da lista serão criadas apenas quando for necessário.

Estas características fazem com que um Programação Funcional esteja pronto para a simultaneidade sem qualquer modificação adicional. Nenhuma parte dos dados em um programa funcional é modificada duas vezes pelo mesmo encadeamento, quanto mais por dois encadeamentos diferentes. Isso significa que poderemps adicionar *threads* sem os riscos inerentes da Programação Imperativa.

São fatores que fazem da Programação Funcional a indicada para desenvolvimento de programas para tratamento de grande massa de dados.

Referencias:

1) https://en.wikibooks.org/wiki/Computer_Programming/Functional_programming
2) https://thesocietea.org/2016/12/core-functional-programming-concepts/
3) http://www.ppgia.pucpr.br/~fabricio/ftp/Aulas/Mestrado/Lisp/LISP-Aula1.PDF
4) http://www.ppgia.pucpr.br/~fabricio/ftp/Aulas/Mestrado/Lisp/LISP-Aula1.PDF
5) https://pt.wikibooks.org/wiki/Processamento_de_Dados_Massivos/Projeto_e_implementa%C3%A7%C3%A3o_de_aplica%C3%A7%C3%B5es_Big_Data
6) https://pt.wikibooks.org/wiki/Processamento_de_Dados_Massivos/Projeto_e_implementa%C3%A7%C3%A3o_de_aplica%C3%A7%C3%B5es_Big_Data
7) https://pt.slideshare.net/gabrielnepo/processamento-paralelo