Exercício Programa 2 - Fase I
MAC0431 - Introdução à Computação Paralela e Distribuida

Fernando Omar Aluani 6797226
Jefferson Serafim Ascaneo 6431284

1 Descrição do Problema

Hoje em dia muitos desenvolvedores usam sistemas de controle de versão para gerenciar seus projetos, e entre esses sistemas, um que é amplamente utilizado é o *Git*. Uma das funcionalidades do Git é a possibilidade de exibir um relatório de informações de um repositório, listando os commits feitos, quem é o autor de cada commit, as alterações realizadas, e mais.

Como pode existir grande variabilidade dos métodos e locais de trabalho dos desenvolvedores, notar quais pessoas estão trabalhando mais ou menos torna-se um desafio. No entanto, o Git pode nos ajudar fornecendo informações detalhadas sobre o repositório, possibilitando a análise do trabalho de cada desenvolvedor.

Além disso, em certas metodologias de desenvolvimento de software é importante que os commits tenham um número reduzido de modificações, permitindo a integração contínua do software sendo desenvolvido e um menor número de conflitos. Um outro ponto interessante é a análise da estabilidade do código, e um indicador para isso é a quantidade de linhas de um arquivo modificadas por commit. Um arquivo que sofre, em média, uma grande quantidade de modificações é mais propenso a erros de programação por conter código mais recente e que ainda não foi executado em um grande número de cenários. Todas estas informações podem ser extraídas dos relatórios fornecidos pelo Git.

1.1 Formato da Entrada de Dados

O relatório do Git é um arquivo texto contendo a descrição de cada commit feito no repositório de forma ordenada, do commit mais recente para o mais antigo.

A descrição de cada commit em si segue o formato:

```
commit <hash code>
Author: <nome> <email>
Date:   <data e hora>

    <texto descrevendo o commit>

<número de linhas adicionadas>      <número de linhas removidas>      <caminho/nome do arquivo modificado>
<número de linhas adicionadas>      <número de linhas removidas>      <caminho/nome do arquivo modificado>
<número de linhas adicionadas>      <número de linhas removidas>      <caminho/nome do arquivo modificado>
...
```

2 Abordagem para Solução

Processar um relatório do Git recebido como entrada, extraindo dele diversas estatísticas, como:

- Número de commits por desenvolvedor;

- Média, desvio padrão e total de linhas modificadas por desenvolvedor;
- Média e desvio padrão de linhas modificadas por commit;
- Média e desvio padrão de linhas modificadas por arquivo.

A função Map irá receber uma chave identificando o commit, que é um hash único gerado pelo Git, e o valor é todo o texto descrevendo o commit. A saída será uma série de chaves e valores, contendo, por exemplo, o par <"lindev_"+ desenvolvedor, número de linhas modificadas neste commit>, ou o par <"numcommits_"+ desenvolvedor, 1>.

```
map(chave, texto):
    desenvolvedor = parseNomeDesenvolvedor(texto)
    // Commits por desenvolvedor
    collect("numcommits_" + desenvolvedor, 1)
    total_linhas = 0
    listaArquivos = parseArquivos(texto)
    for arquivo in listaArquivos:
        total_linhas += arquivo.linhas_modificadas
        // Linhas modificadas por arquivo
        collect("arq_" + arquivo.nome, arquivo.linhas_modificadas)
    // Linhas modificadas por desenvolvedor
    collect("lindev_" + desenvolvedor, total_linhas)
    // Linhas modificadas por commit
    collect("linhas_mod_por_commit", total_linhas)
```

A função Reduce irá receber os pares produzidos pela função Map e, a partir deles, irá calcular a média, desvio padrão e total de cada tipo de chave.

```
reduce(chave, valores):
    soma=0
    somaQuadrado = 0
    c = 0
    for v in valores:
        soma += v
        somaQuadrado += v*v
        c++

    media = soma/c
    collect(chave + "_media", media)
    collect(chave + "_desvpadr",
            sqrt(abs(somaQuadrado - media * soma) / c))
    collect(chave + "_total", soma)
```