Projeto 2 – WebCrawler

André De Marco Toyama

https://github.com/AToyama/WebCrawler

INTRODUÇÃO

O objetivo desse projeto foi criar um programa que realiza o download e análise de páginas de produto de um site de e-commerce. Foram desenvolvidas duas versões diferentes do crawler, uma sequencial e outra paralela, com o intuito de realizar uma comparação de desempenho entre as duas versões.

FUNCIONAMENTO

Esse projeto foi feito baseado no e-commerce https://www.submarino.com.br, portanto o uso de qualquer outro site resultara no mau funcionamento do projeto. O código recebe como entrada uma URL apontando alguma categoria de produtos do site, então lista a url de todas as páginas de produtos dessa categoria. Por fim ele faz o download e análise de cada uma das páginas de produto da respectiva categoria.

O processo paralelizado é divido em threads que realizam tarefas distintas ao mesmo tempo. As threads são dividas em produtoras e consumidoras, as produtoras geram as urls das paáginas de produto e mandam para as consumidoras que realizam o download e análise das mesmas.

DADOS

Para a comparação final foram tiradas as seguintes medidas de tempo:

- Tempo para download e análise de cada produto
- Tempo médio para download e análise de produto
- Tempo total ocioso realizando download de páginas
- Tempo Total para rodas o programa

Foram executados os teste em 4 formatos diferentes do programa:

- Sequencial
- Paralelo com 2 threads
- Paralelo com 4 threads
- Paralelo com 8 threads

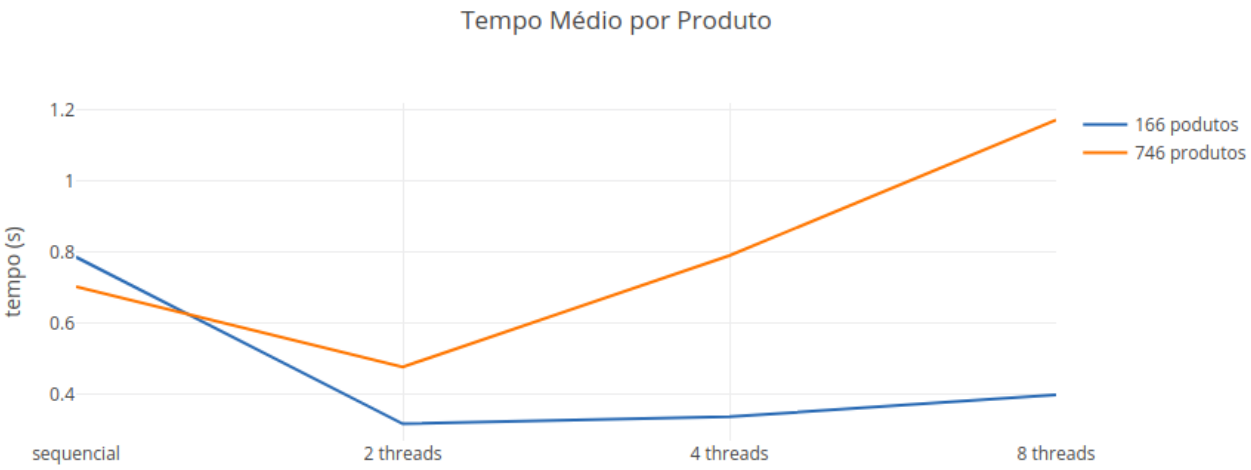
E também foram utilizadas duas URLs diferentes apresentando quantidades diferentes de produtos a serem análisados:

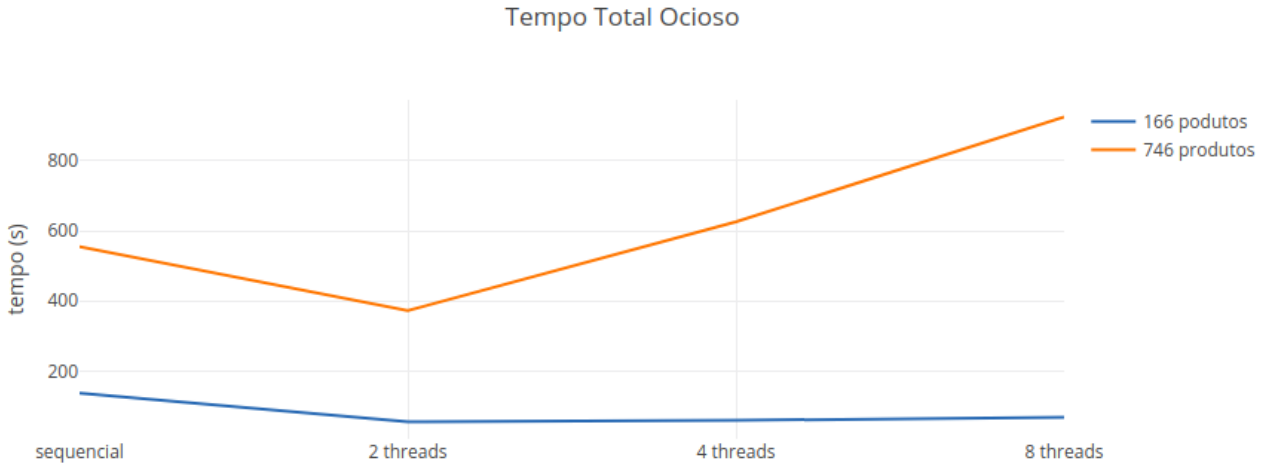
- https://www.submarino.com.br/categoria/acessorios-de-informatica/fitas (166 produtos)
- https://www.submarino.com.br/categoria/games/nintendo-ds (746 produtos)

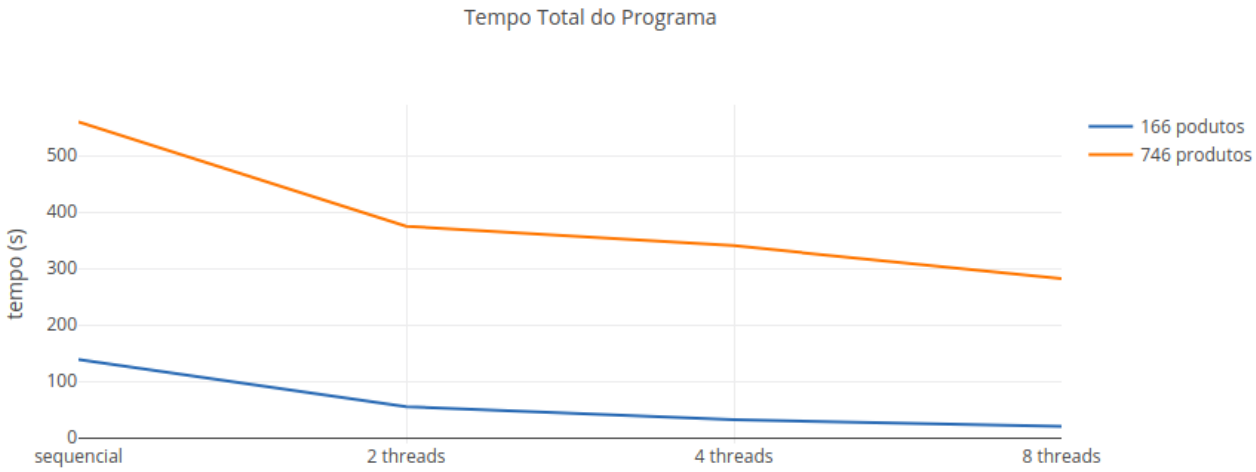
SISTEMA

CPU: Intel Core i5-4210U CPU @ 2.7GHz
GPU: Mesa DRI Intel(R) Haswell Mobile
RAM: 8GB DDR3

RESULTADOS







A partir dos gráficos feitos nós podemos reparar que o tempo médio por produto e o tempo total ocioso aumentam conforme mais threads, mesmo o esperado sendo uma redução de tempo. Mas o que temos que entender é que mesmo que estes valores estejam maiores, o número de produtos sendo processados ao mesmo tempo é maior. Então se tivermos como exemplo um tempo médio de 1 segundo por produto, se tivermos 8 threads, teremos 4 produtos sendo processados ao mesmo tempo, ou seja em 1 segundo. Para ajudar no entendimento basta olhar o terceiro gráfico, onde podemos ver que embora os outros valores possam estar maiores, o tempo total do programa é reduzido proporcionalmente ao número de threads.

TESTE DE MEMÓRIA

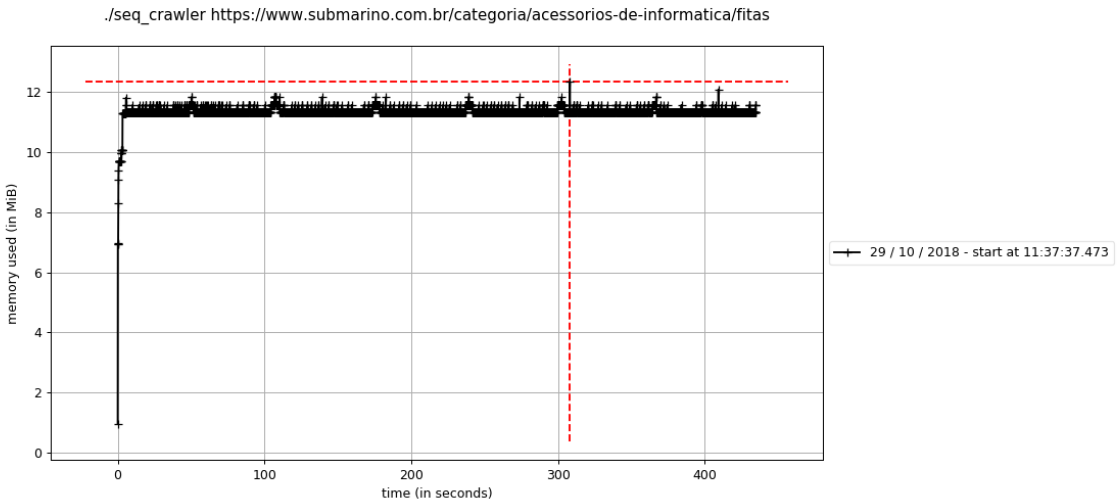


Illustration 1: teste de memória mprof - sequencial

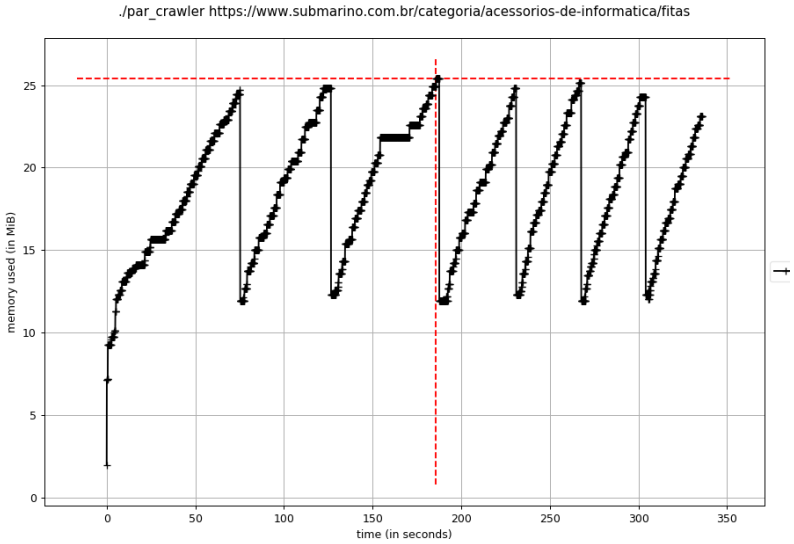


Illustration 2: teste de memória mprof - paralelo 2 threads

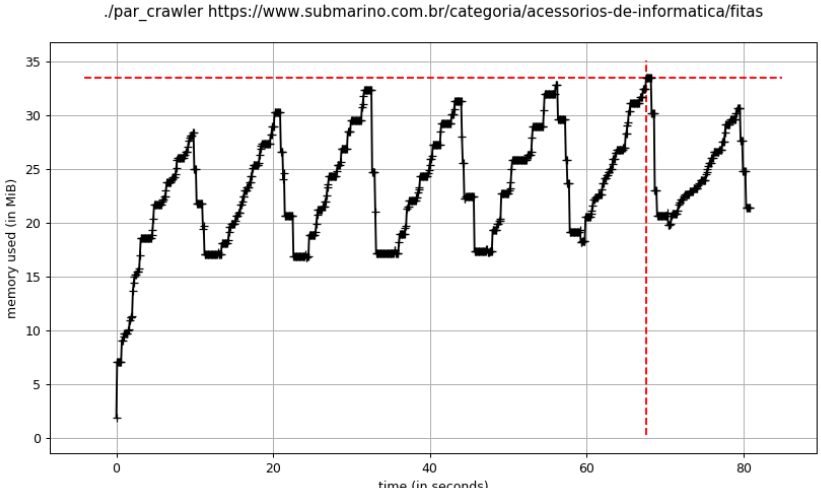


Illustration 3: teste de memória mprof - paralelo 4 threads]

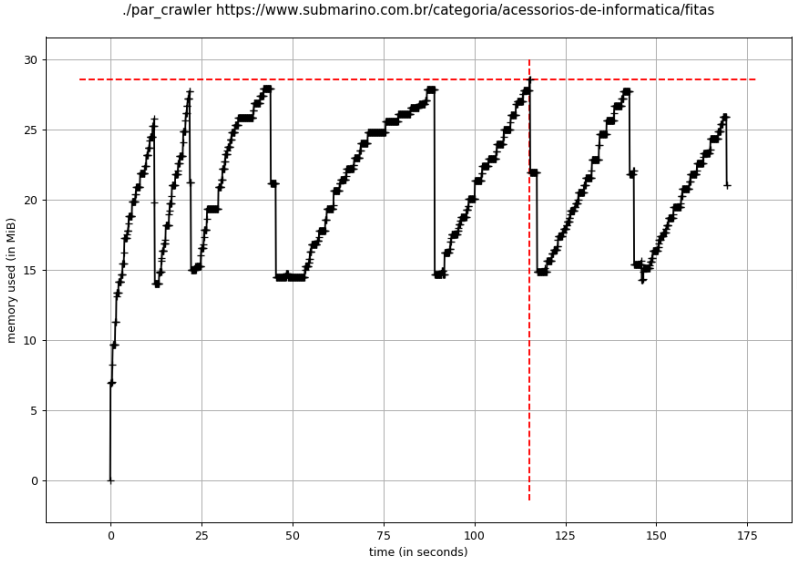


Illustration 4: teste de memória mprof - paralelo 8 threads

Podemos ver que o teste de memória não apresentou muita diferença em relação ao número de threads, porém é nitida a diferença entre a versão sequencial e as paralelas em geral. As versões paralelas consomem mais memória no geral, além de apresentar um formato com vários picos de consumo de memória, equivalentes a atividade das threads.

REPRODUTIBILIDADE

No README do repositório do projeto tem as instruções necessárias para rodar e realizar os teste por conta própria, basta acessar o link disponível no começo do relatório..