Insper

Parsing de páginas web em C++

Super Computação 2018/2

Igor Montagner, Luciano Soares

Neste projeto trabalharemos com um problema em que concorrência tem um papel fundamental na obtenção de bom desempenho: download e análise de páginas web. Iremos criar um crawler que identifica páginas de produtos em um site de e-commerce e extrai as informações básicas dos produtos. Veja abaixo alguns exemplos de páginas a serem analisadas.

1. Bicicleta Caloi na Amazon;
2. iPhone 8 no Magazine Luiza;

O crawling de páginas é uma etapa do sistemas de comparação de preços de sites como Buscapé e Zoom. Após esta extração estes sistemas cruzam as informações dos sites de modo a identificar quais produtos estão presentes em ambos e fazer as páginas de comparação de preços.

Parte 1 - Descrição do projeto

Dada a página de exibição de um produto, seu web crawler deverá extrair as seguintes informações:

1. nome do produto
2. descrição do produto
3. url da foto do produto
4. preço à vista
5. preço parcelado
6. categoria do produto
7. url da página de exibição

A identificação de páginas de produto pode ser feita a partir de sua categoria, ou seja, você pode apontar seu crawler para uma página com todos os produtos de uma categoria e ele deve conseguir obter as páginas de produto, possivelmente lidando com algum tipo de paginação da listagem.

Seu programa deverá ser executado na linha de comando como

```
./crawler url_da_listagem_por_categoria
```

A lista abaixo contém exemplos de páginas de categorias de produto.

1. Brinquedos para bebê no Submarino;
2. Computadores Gamer na Kabum;
3. Bicicletas na Amazon;
4. Smartphones no Magazine Luiza.

O resultado do seu programa deverá ser escrito na saída padrão no formato `json`. Cada produto deverá ser um objeto com os seguintes campos:

```
{
    "nome": "",
    "descricao": "",
    "foto": "",
    "preco": 0,
    "preco_parcelado": 0,
    "preco_num_parcelas": 0,
    "categoria": "",
    "url": ""
}
```

Métricas de desempenho

Seu programa deverá calcular algumas métricas de desempenho para quantificar os ganhos de desempenho.

1. Tempo total por produto: tempo total gasto no download e análise de cada produto.
2. Tempo médio por produto: tempo total de execução do programa dividido pelo total de produtos analisados.
3. Tempo total ocioso: tempo total gasto esperando o download de páginas web.

Para quantificar os ganhos de desempenho ao usar concorrência você deve calculá-las para diferentes números de threads. Esta opção deverá ser configurável no programa sem a necessidade de recompilá-lo. Você deve também fazer uma versão puramente sequencial para evidenciar os ganhos obtidos pela sua paralelização.

Data de entrega

O projeto está com data de entrega para `10/10`. Como o projeto 3 envolverá a distribuição deste código em um cluster, a data pode ser extendida para `15/10`. A data final de entrega do projeto 3 é `31/10`.

Quanto mais cedo o trabalho for entregue mais tempo haverá para o projeto 3 e os trabalhos entregues na primeira data serão corrigidos antes do dia `15/10`.

Parte 2 - suporte tecnológico

Nesta seção do roteiro apresentamos algumas bibliotecas que podem ser usadas para o download das páginas e sua análise.

- cpr - C++ Request - biblioteca fácil de usar para fazer download de páginas web.
- Gumbo - Parser HTML em C
- Gumbo-Query - Implementa seletores CSS usando Gumbo
- GQ - Também implementa seletores CSS usando Gumbo. Parece mais atualizado, mas é menos popular.
- regex - Biblioteca padrão para expressões regulares. Mais simples de usar e potencialmente também mais simples de aplicar nas páginas.

Seu projeto deverá usar CMake para compilação e deverá checar se as bibliotecas usadas estão presentes.

Parte 3 - avaliação

A avaliação do projeto será dividida em três partes: técnicas de super computação (40%), qualidade de código (20%) e relatório de desempenho (30%). Os últimos 10% da nota estão reservados para a rubrica A+, descrita no fim desta seção.

Técnicas de super computação

Na primeira parte serão avaliados os conceitos apresentados no módulo de concorrência da disciplina. Em especial, será avaliada

- a divisão das tarefas em threads
- a utilização correta de mecanismos de sincronização para acesso a recursos compartilhados
- o grau de paralelismo da sua solução

Qualidade de código

Na segunda parte será avaliada a organização do seu código e a facilidade de compilação e uso. Seu projeto deverá estar acompanhado de um README explicando como compilar e rodar o crawler e listando quais sites são suportados pela ferramenta. Explique com clareza como obter os links de página de produto e liste algumas páginas testadas por sua solução.

Recomendado: faça seu código de maneira que seja possível adicionar outros sites de e-commerce sem modificar o programa de maneira extensa. Isto facilitará o trabalho na rubrica A+.

Relatório de desempenho

Será usada a mesma rubrica do projeto anterior. Neste projeto, além da comparação de desempenho com CPU é importante medir também o consumo de memória do programa. Você pode fazer isto usando o mprof.

Devido a fatores como rede ou número de cores da máquina, é especialmente importante criar um conjunto de testes de desempenho de fácil replicação. Sua comparação de desempenho deverá usar as métricas listadas na Seção 1 além das métricas de CPU e memória.

Dica: o projeto é em C++, mas vocês podem usar Python, por exemplo, para criar um conjunto de testes que rode automaticamente e já gere os dados usados no relatório. Isto pode ser feito de maneira prática usando o PWeave, que permite intercalar código e texto e gera um relatório com formato bonito.

Rubrica A+

Para ganhar o último ponto seu programa deverá suportar mais de um site de e-commerce de maneira transparente. Ou seja, dependendo da url passada na linha de comando ele deverá ser capaz de identificar o site, chamar o código correspondente e apresentar os resultados. Se a url passada não for identificada ele deverá indicar isto na saída do programa.

Esta rubrica só é válida se for obtida nota maior que 5 nos conceitos anteriores.