Crawler-Doc

Isaque Evangelista e Ezio Pacheco

Maio de 2025

1 Descrição Geral

Este projeto é um sistema de **crawler inteligente** que combina raspagem de dados web com *IA* para análise de conteúdo. Oferece múltiplas interfaces de uso e capacidades de busca avançada, tornando-se ideal para aplicações que exigem recuperação semântica de informações.

2 Funcionalidades

- **Web Scraping Paralelo:** Extração eficiente de conteúdo de múltiplas páginas web utilizando processamento paralelo.
- **Processamento com IA:** Utiliza embeddings e modelos de linguagem (LLM) para análise contextual e semântica do conteúdo extraído.
- **Banco de Dados Vetorial:** Armazena os embeddings usando `ChromaDB` para busca semântica.
- **Busca de Palavras:** Contagem e pesquisa por termos específicos no conteúdo coletado.
- **Modo Terminal:** Operação totalmente via linha de comando com diferentes modos de execução.

3 Modos de Execução

- **Scraping Puro:** Apenas coleta de dados, sem análise semântica.
- **Scraping + IA:** Coleta e indexação no banco vetorial com suporte a consultas semânticas.
- **Scraping + Busca:** Coleta e contagem de palavras para análise estatística.
- **Sair:** Encerra o programa.

4 Configuração

- **URL Base:** URL inicial para o processo de scraping.
- **Profundidade Máxima:** Número de níveis de links a seguir a partir da URL base.
- **Processos:** Número de processos em paralelo utilizados na coleta.
- **IA:** Ativa ou desativa o uso de IA para análise.

5 Lógica de Funcionamento

1. O usuário insere a URL e o nível de profundidade.
2. A URL é raspada (scraping).
3. 3.1 Os dados são adicionados ao banco vetorial para RAG (Retrieval-Augmented Generation).
   3.2 Os dados são adicionados ao banco SQL para contagem de palavras.
   3.3 As novas URLs são adicionadas à fila.
   3.4 O processo pai administra a distribuição para subprocessos.
4. Se a fila não estiver vazia, retornar ao passo 2.
5. Se a fila estiver vazia ou o nível de profundidade for atingido, encerrar.
6. Permitir que o usuário consulte o banco vetorial com um modelo de linguagem.

6 Especificações do Sistema de Teste

Os benchmarks foram executados no seguinte ambiente:

6.1 Hardware

- **CPU:** Intel Core i9-12900KF (12ª geração) - 24 núcleos (8P + 16E)
- **GPU:** NVIDIA GeForce RTX 3060
- **Memória RAM:** 48 GB DDR4
- **Armazenamento:** SSD NVMe

6.2 Software

- **Sistema Operacional:** Linux Mint 22.1
- **Kernel:** 6.8.0-60-generic
- **Desktop Environment:** Cinnamon 6.4.8
- **Shell:** Bash 5.2.21
- **Python:** 3.11+

7 Benchmark Detalhado

7.1 Resultados de Performance

Os testes foram executados com diferentes configurações de profundidade e número de processos paralelos. A tabela a seguir apresenta os resultados completos:

7.2 Análise dos Resultados

7.2.1 Escalabilidade

O sistema demonstra excelente escalabilidade, com melhorias significativas de performance conforme o aumento do número de processos paralelos:

- **Profundidade 2:** Speedup de 9.2x com 24 processos
- **Profundidade 3:** Speedup de 15.1x com 24 processos
- **Profundidade 4:** Speedup de 15.6x com 24 processos

Table 1: Resultados de Performance do Crawler

Prof.	Proc.	Páginas	Tempo (s)	Pág/s	Speedup	Eficiência (%)
2	1	37	17.41	2.12	1.00	100.0
	2	37	9.29	3.98	1.87	93.7
	4	37	4.94	7.48	3.52	88.1
	8	37	2.94	12.58	5.92	74.0
	12	37	2.33	15.88	7.47	62.3
	16	37	2.18	16.94	7.99	49.9
	24	37	1.89	19.61	9.21	38.4
3	1	184	85.65	2.15	1.00	100.0
	2	184	43.81	4.20	1.95	97.7
	4	184	22.43	8.20	3.82	95.4
	8	184	11.69	15.74	7.32	91.5
	12	184	9.15	20.10	9.36	78.0
	16	184	7.58	24.27	11.30	70.6
	24	184	5.66	32.49	15.13	63.0
4	1	325	151.13	2.15	1.00	100.0
	2	325	75.79	4.29	1.99	99.6
	4	325	38.83	8.37	3.89	97.3
	8	325	21.28	15.27	7.10	88.8
	12	325	15.05	21.59	10.04	83.7
	16	325	12.35	26.31	12.24	76.5
	24	325	9.68	33.58	15.61	65.0

7.2.2 Eficiência de Paralelização

A eficiência de paralelização permanece alta até 8 processos (90%), começando a declinar gradualmente com mais processos devido ao overhead de coordenação e limitações de I/O.

7.2.3 Throughput Máximo

O maior throughput foi alcançado com profundidade 4 e 24 processos: **33.58 páginas/segundo**, processando 325 páginas em apenas 9.68 segundos.

7.2.4 Relação Profundidade-Performance

Profundidades maiores apresentam melhor utilização de recursos paralelos, pois o maior volume de trabalho permite distribuição mais eficiente entre os processos.

7.3 Gráficos de Performance

7.3.1 Speedup vs Número de Processos

7.3.2 Eficiência de Paralelização

7.3.3 Taxa de Processamento (Páginas/Segundo)

8 Problemas Encontrados

8.1 Repetição de Palavras

Devido à sobreposição de chunks (*overlap*) para manter o contexto, algumas palavras são repetidas, afetando a contagem estatística.

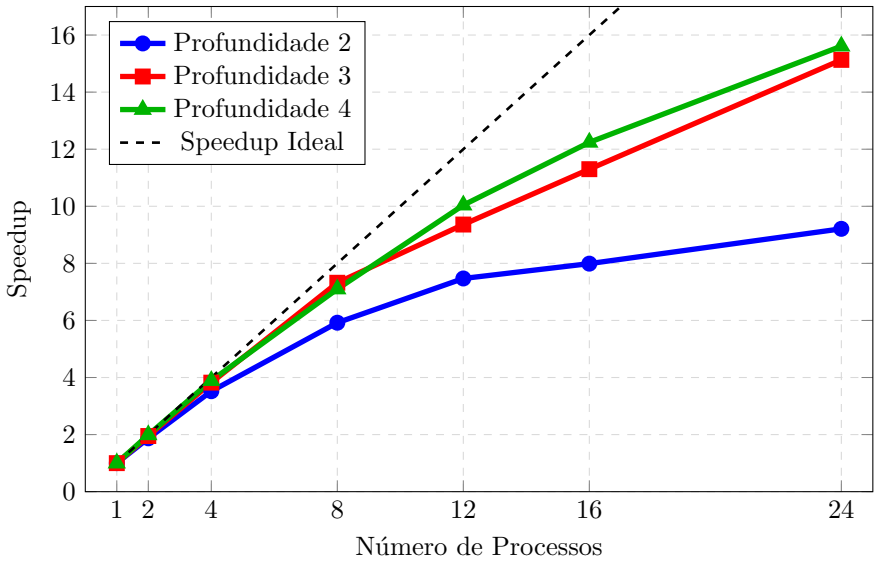

Figure 1: Speedup relativo ao processamento com 1 processo

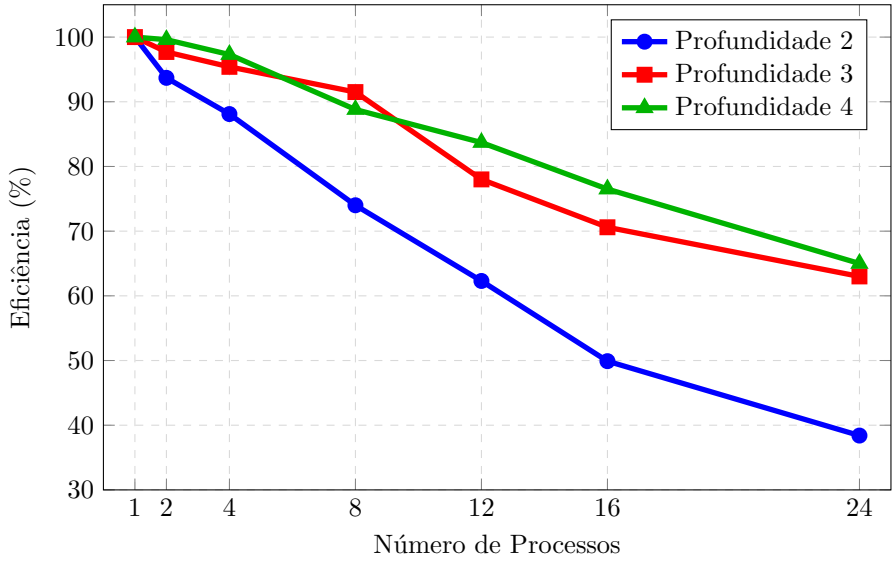

Figure 2: Eficiência de paralelização por número de processos

8.2 Interface de Usuário

Havia a intenção de desenvolver uma interface usando Streamlit, mas essa funcionalidade foi descartada por limitações de tempo.

8.3 Alta Demanda de Processamento

A geração de embeddings e uso de modelos locais demanda alta capacidade de hardware. As soluções consideradas foram:

- Tornar o uso de IA opcional.
- Utilizar uma API externa como a OpenAI para offload do processamento.

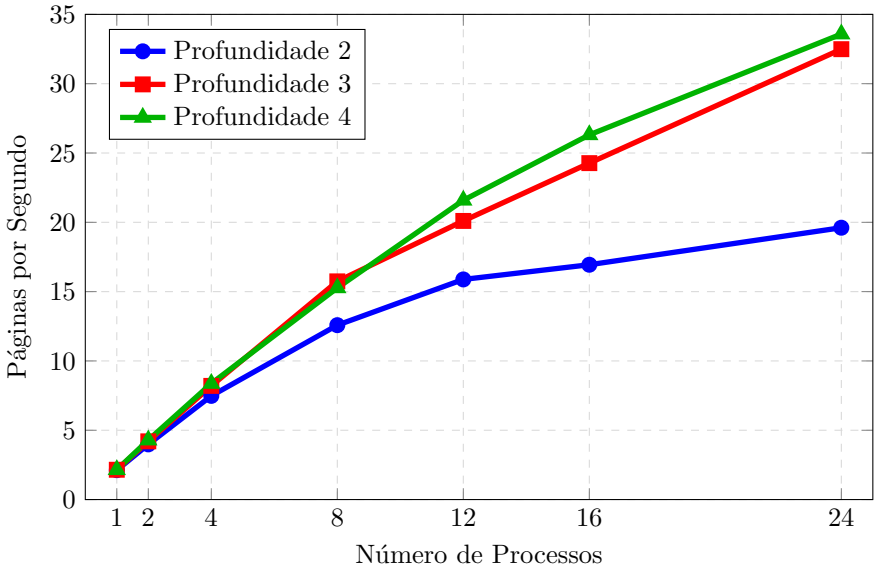

Figure 3: Taxa de processamento por número de processos

8.4 Diminuição da Eficiência com Muitos Processos

Observou-se que com mais de 16 processos, a eficiência de paralelização diminui significativamente devido a:

- Overhead de coordenação entre processos
- Contenção de recursos de I/O de rede
- Limitações do servidor web alvo

9 Recomendações

Com base nos resultados dos benchmarks, recomenda-se:

- **Configuração Ótima:** 12-16 processos paralelos para melhor relação performance/eficiência
- **Workloads Pequenos:** Para profundidade 2, usar até 8 processos
- **Workloads Grandes:** Para profundidades 3-4, usar até 24 processos se a performance máxima for prioritária
- **Recursos Limitados:** 4-8 processos oferecem bom speedup com menor uso de recursos

10 Dependências Principais

- `chromadb` – Banco de dados vetorial.
- `urllib3` – Requisições HTTP.
- `concurrent.futures` – Execução paralela.
- `langchain`, `ollama` – Integração com LLMs locais e processamento semântico.
- `beautifulsoup4` – Parsing de HTML.

11 Autores

- Isaque Evangelista
- Ezio Pacheco