ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO
LAB 1 – SCRAPING

Giảng viên hướng dẫn: Huỳnh Lâm Hải Đăng

Môn học: Nhập môn khoa học dữ liệu

Họ và tên sinh viên: Lê Hà Thanh Chương

Mã số sinh viên: 23120195

Thành phố Hồ Chí Minh, tháng 11 năm 2025

MỤC LỤC

A. TỔNG QUAN

Mục tiêu chính của bài lab này là ứng dụng kiến thức lý thuyết về thu thập dữ liệu (data scraping) vào một dự án kỹ thuật thực tiễn thông qua việc xây dựng một chương trình scraper để thu thập dữ liệu từ kho lưu trữ bài báo khoa học mở arXiv. Scraper cần thực hiện việc thu thập dữ liệu theo các phạm vi ID chỉ định, đồng thời thu thập tất cả các phiên bản của mỗi bài báo (ví dụ: v1, v2, v3...). Với mỗi bài báo, chương trình phải lưu trữ bốn loại dữ liệu chính: (1) Nguồn TeX chứa toàn bộ mã nguồn .tex; (2) BibTeX chứa thông tin trích dẫn nếu có; (3) Metadata của bài báo như tên, tác giả, ngày nộp, được lưu trong file metadata.json; và (4) Thông tin tham khảo thu thập từ Semantic Scholar API, lưu trong file references.json. Ngoài ra, bước xử lý dữ liệu cũng được yêu cầu, bao gồm loại bỏ tất cả các file hình ảnh trong source đã scrape nhằm giảm dung lượng lưu trữ và chuẩn hóa dữ liệu cho các phân tích tiếp theo.

B. NỘI DUNG CHI TIẾT

I. QUY TRÌNH TRIỂN KHAI

1. Kỹ thuật và công cụ sử dụng

Trong quá trình thu thập dữ liệu bài làm sử dụng đồng thời các thành phần từ Python Standard Library, các thư viện bên thứ ba và API bên ngoài nhằm bảo đảm tính ổn định, độ chính xác và khả năng mở rộng.

Trước hết, các module thuộc Python Standard Library được sử dụng rộng rãi cho các tác vụ nền tảng. Nhóm xử lý hệ thống tệp và thao tác hệ điều hành bao gồm *os, shutil, tarfile, gzip, tempfile, pathlib.Path, csv* và *json*, cho phép thao tác tệp nén, quản lý cấu trúc thư mục và ghi/đọc dữ liệu trung gian một cách tin cậy. Các tác vụ liên quan đến thời gian được hỗ trợ bởi *time* và *datetime*.

Để xử lý đa luồng và đồng thời, bài làm sử dụng *threading, ThreadPoolExecutor* từ *concurrent.futures* và *Lock*, giúp tối ưu tốc độ thu thập dữ liệu trong khi vẫn duy trì tính an toàn luồng. Các chức năng xử lý chuỗi và biểu thức chính quy được thực hiện bởi *re*, trong khi *typing* được sử dụng để định nghĩa rõ ràng các kiểu dữ liệu (*List, Dict, Tuple, Optional*).

Bên cạnh đó, bài làm cũng tích hợp các thư viện Python bên thứ ba nhằm mở rộng năng lực xử lý. Thư viện *requests* đảm nhiệm việc gửi các HTTP request đến arXiv và Semantic Scholar, được cấu hình bổ sung *HTTPAdapter* và cơ chế retry nhằm bảo đảm độ bền vững trước lỗi mạng. Module *urllib3.util.retry.Retry* hỗ trợ quản lý backoff và chiến lược thử lại tự động. Để phân tích cấu trúc HTML của các trang arXiv, bài làm sử dụng *BeautifulSoup* từ gói *bs4*. Đồng thời, *psutil* được triển

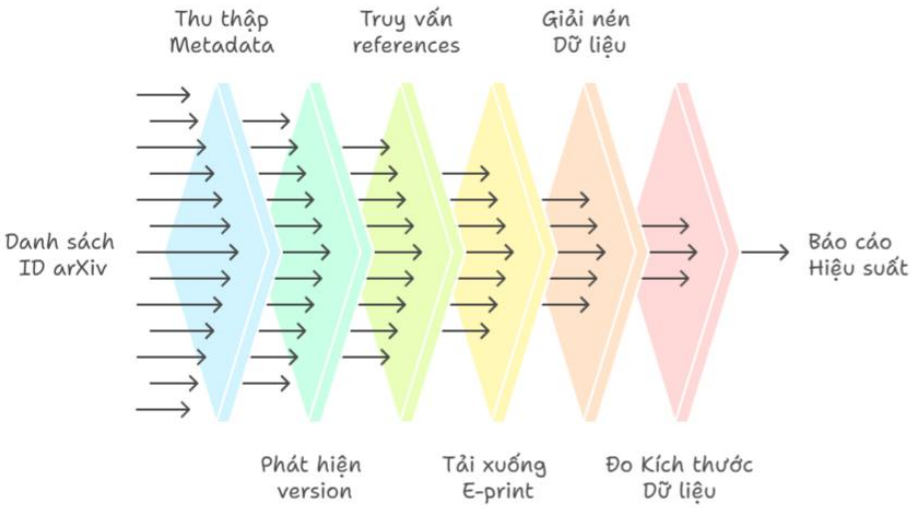
khai để giám sát mức sử dụng tài nguyên hệ thống, đặc biệt là RAM trong quá trình xử lý từng bài báo khoa học.

Chương trình cũng tương tác trực tiếp với các API bên ngoài, bao gồm nhóm ArXiv Web Scraping Interfaces và ArXiv E-Print Source Endpoint. Quá trình thu thập dựa trên các web endpoints như trang ABS (/abs/{id}) để lấy tiêu đề, danh sách tác giả, ngày nộp và lịch sử chỉnh sửa, cũng như endpoint tải source (/e-print/{id_with_version}) cho phép thu thập toàn bộ tệp LaTeX. Nội dung HTML thu được được phân tích nhằm trích xuất thông tin về lịch sử nộp bài, journal reference và phân loại chủ đề. Song song đó, Semantic Scholar Graph API (endpoint Paper API: /graph/v1/paper/arXiv:{id}) cung cấp dữ liệu liên quan đến venue, danh sách tài liệu tham khảo và thông tin meta khác.

2. Cơ chế hoạt động

Chương trình scraper được thiết kế để thu thập dữ liệu từ arXiv theo khoảng ID xác định, sử dụng cơ chế batch kết hợp đa luồng (ThreadPoolExecutor) nhằm tối ưu hóa hiệu năng và đảm bảo khả năng mở rộng. Toàn bộ quá trình bắt đầu bằng việc tạo danh sách các arXiv base ID theo khoảng (START_MONTH, START_ID → END_ID) và chia thành các batch. Với mỗi paper, scraper thực hiện tuần tự các bước:

i. Thu thập metadata từ trang abstract của arxiv;
ii. Phát hiện tất cả các version có sẵn;
iii. Truy vấn Semantic Scholar Graph API để thu thập thông tin references;
iv. Tải về source e-print;
v. Giải nén đệ quy các archive để trích xuất các file latex và bibtex;
vi. Đo kích thước dữ liệu trước và sau khi loại bỏ hình ảnh;
vii. Ghi kết quả và các thông số hiệu năng.



Trong quá trình này, scraper vận hành một session HTTP với cơ chế retry/backoff để đảm bảo độ bền mạng, áp dụng rate-limit thread-safe, ghi log chi tiết và sử dụng các fallback khi xảy ra lỗi (ví dụ thử tải e-print không theo version nếu phiên bản cụ thể không khả dụng).

Chi tiết từng bước thực thi được thực hiện như sau: đầu tiên, môi trường được khởi tạo với các hằng số cấu hình, session HTTP với adapter retry, và Lock cho throttling thread-safe. Danh sách ID được tạo và phân chia batch, mỗi batch được xử lý song song thông qua ThreadPoolExecutor, thu thập kết quả và đánh dấu trạng thái thất bại nếu có exception.

Trên mức độ từng paper, scraper khởi đo hiệu năng (thời gian, RAM), tạo thư mục lưu trữ, truy xuất metadata từ arXiv, xác định các version có sẵn, và truy vấn Semantic Scholar để trích references. Source e-print được tải về và giải nén đệ quy, đảm bảo các file LaTeX và BibTeX được giữ nguyên cấu trúc thư mục. Các bước đo kích thước dữ liệu, xóa file hình ảnh, và cleanup thư mục tạm được thực hiện để tránh rò rỉ bộ nhớ và disk. Metadata, references và thông số hiệu năng (thời gian, RAM, kích thước dữ liệu) được ghi lại cho từng paper.

Cuối cùng, sau khi tất cả batch được xử lý, scraper tổng hợp các kết quả và thống kê: số paper xử lý thành công, thất bại, kích thước trung bình trước/sau xử lý, số references trung bình, thời gian xử lý, và mức sử dụng bộ nhớ, từ đó tạo ra báo cáo toàn diện (performance_report.json) và log tóm tắt hiệu năng. Toàn bộ quá trình minh họa một pipeline thu thập dữ liệu khoa học có cấu trúc, vừa đảm bảo độ tin cậy cao vừa tối ưu hóa hiệu năng thông qua xử lý song song và quản lý tài nguyên mạng/bộ nhớ chặt chẽ.

II. THỐNG KÊ VÀ PHÂN TÍCH KẾT QUẢ

Các thí nghiệm được tiến hành trên môi trường Google Colab nhằm đánh giá độ tin cậy và đặc trưng dữ liệu của hệ thống thu thập thông tin từ arXiv.

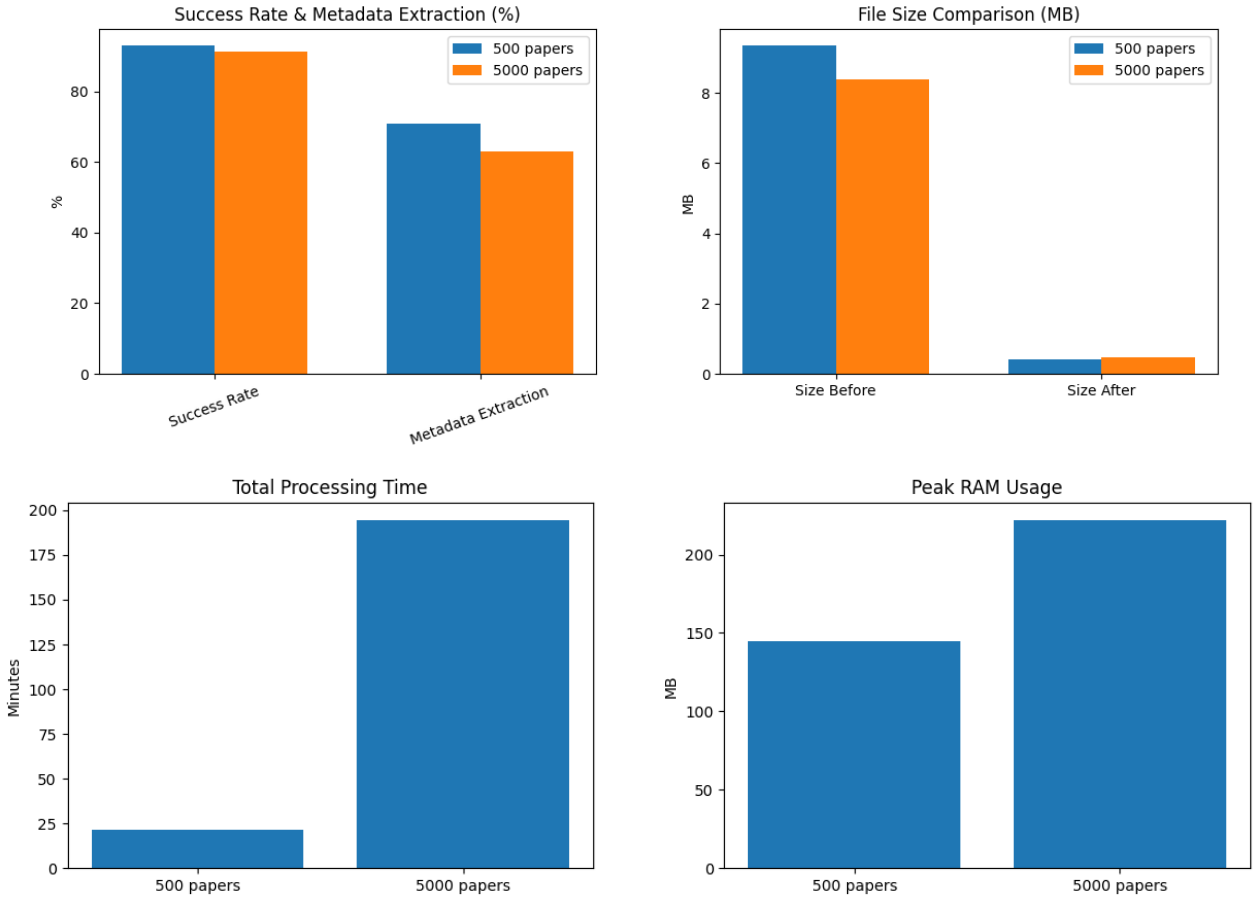
Với mẫu đầu tiên gồm 500 bài báo, chương trình đạt 465 bài scrape thành công, tương ứng tỷ lệ 93.0%. Kích thước trung bình của nguồn dữ liệu LaTeX trước khi loại bỏ hình ảnh đạt 9.34 MB, trong khi kích thước trung bình của phần nội dung thuần túy (chỉ bao gồm tệp .tex và .bib) giảm xuống còn 0.42 MB, phản ánh mức độ đóng góp rất lớn của thành phần hình ảnh vào tổng dung lượng e-print. Trung bình mỗi bài báo có 16.33 tài liệu tham khảo, và chương trình trích xuất thành công metadata của references với tỷ lệ 70.8%. Tổng thời gian xử lý 500 bài là 21.5 phút, với mức tiêu thụ bộ nhớ cực đại đo được là 144.492 MB.

```
[14:49:33] --- SCRAPING COMPLETED ---
[14:49:33] Successful papers: 465/500 (93.0%)
[14:49:33] Average paper size before removing figures: 9792039.5 bytes
[14:49:33] Average paper size after removing figures (tex/bib only): 444810.0 bytes
[14:49:33] Average references per successful paper: 16.33
[14:49:33] Reference metadata success rate: 70.8%
[14:49:33] Total time: 21.5 minutes
[14:49:33] Total memory usage by all papers: 69984.438 MB
[14:49:33] Max ram used: 144.492 MB
```

Ở quy mô lớn hơn với 5,000 bài báo, xu hướng thống kê vẫn nhất quán nhưng thể hiện sự thay đổi đáng chú ý theo kích thước mẫu. Tỷ lệ thành công đạt 91.3% (4,566 bài), giảm nhẹ so với mẫu nhỏ do gia tăng các trường hợp bài không có source hợp lệ hoặc cấu trúc LaTeX không chuẩn. Kích thước e-print ban đầu trung bình là 8.39 MB, thấp hơn nhóm 500 bài, trong khi kích thước nội dung .tex/.bib tăng lên mức 0.485 MB, gợi ý rằng đối với tập dữ liệu lớn hơn, tỉ lệ phần văn bản trong source LaTeX có xu hướng cao hơn tương đối so với thành phần hình ảnh. Số lượng tài liệu tham khảo trung bình giảm còn 14.69, và tỷ lệ trích xuất metadata references cũng giảm xuống 63.0%. Tổng thời gian xử lý mẫu 5,000 bài đạt 194.1 phút, với mức RAM đỉnh là 221.6 MB.

```
[18:27:07] --- SCRAPING COMPLETED ---
[18:27:07] Successful papers: 4566/5000 (91.3%)
[18:27:07] Average paper size before removing figures: 8802020.7 bytes
[18:27:07] Average paper size after removing figures (tex/bib only): 508560.6 bytes
[18:27:07] Average references per successful paper: 14.69
[18:27:07] Reference metadata success rate: 63.0%
[18:27:07] Total time: 194.1 minutes
[18:27:07] Total memory usage by all papers: 943845.961 MB
[18:27:07] Max ram used: 221.621 MB
```

Nhìn chung, các thống kê này cho thấy chương trình scraper đạt độ ổn định và khả năng mở rộng tốt khi chuyển từ quy mô nhỏ (500 bài) sang quy mô lớn (5,000 bài). Sự suy giảm nhẹ về tỷ lệ thành công và tỷ lệ trích xuất metadata phản ánh bản chất không đồng nhất của tập bài báo arXiv theo từng tháng và theo từng lĩnh vực.

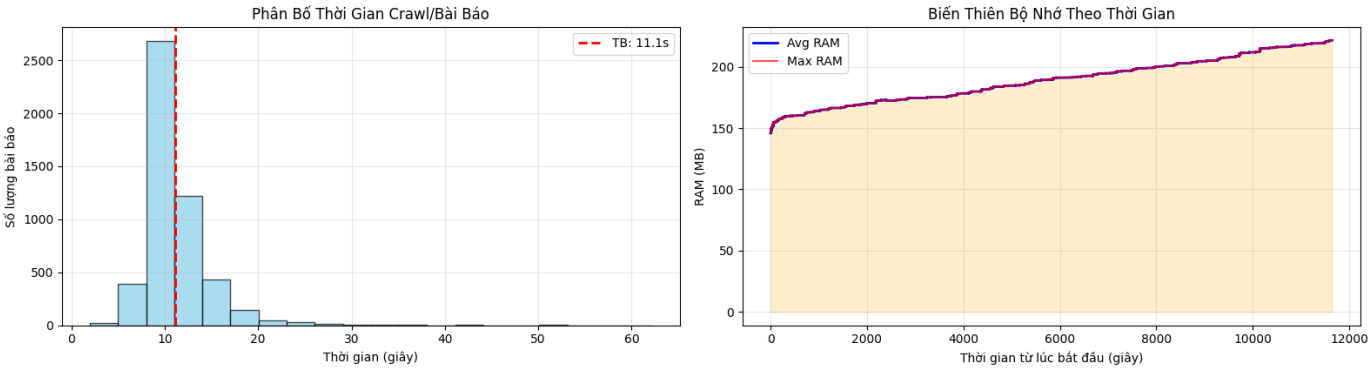


III. ĐÁNH GIÁ HIỆU NĂNG

Để đánh giá hiệu năng của pipeline crawl và xử lý dữ liệu, bài làm tiến hành thử nghiệm trên một tập mẫu gồm 500 và 5000 bài báo, tập trung vào ba khía cạnh: thời gian xử lý, sử dụng bộ nhớ (RAM) và hiệu quả nén/chiết xuất dữ liệu đầu ra.

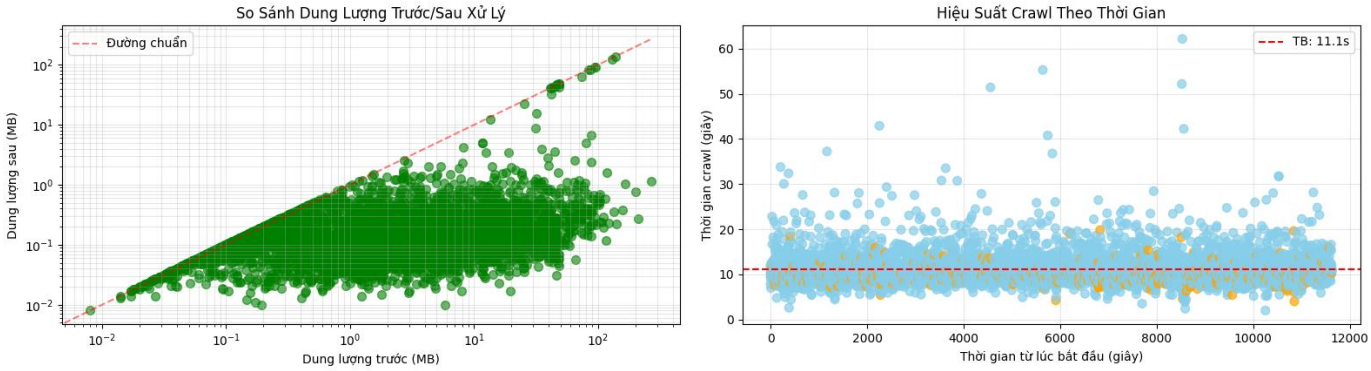
1. Mẫu 5000 bài báo

Về hiệu suất thời gian, chương trình đạt tốc độ xử lý trung bình 11.09 giây/bài báo, với độ lệch chuẩn 3.52 giây. Phân phối thời gian xử lý (Hình 1, trái) thể hiện lệch phải (right-skewed), với phần lớn tác vụ (khoảng 2,700 bài) tập trung quanh 10 giây.



Hình 1

Tuy nhiên, biểu đồ hiệu suất theo thời gian (Hình 2, phải) cho thấy mức biến động cao, với các giá trị ngoại lai lên tới 62.14 giây. Sự xuất hiện ngẫu nhiên của các ngoại lai này được giải thích phần nào bởi mối quan hệ dương rõ rệt giữa thời gian xử lý và dung lượng gốc của bài báo (Hình 1, phải): thời gian xử lý có xu hướng tăng lên khi dung lượng gốc của bài báo tăng, lý giải một phần cho sự xuất hiện của các điểm ngoại lai. Do chương trình cài đặt có khoảng thời gian nghỉ giữa các request để điều phối tránh vượt giới hạn rate limit nên tổng thời gian chạy thực tế cao hơn nhiều so với tổng thời gian xử lý.

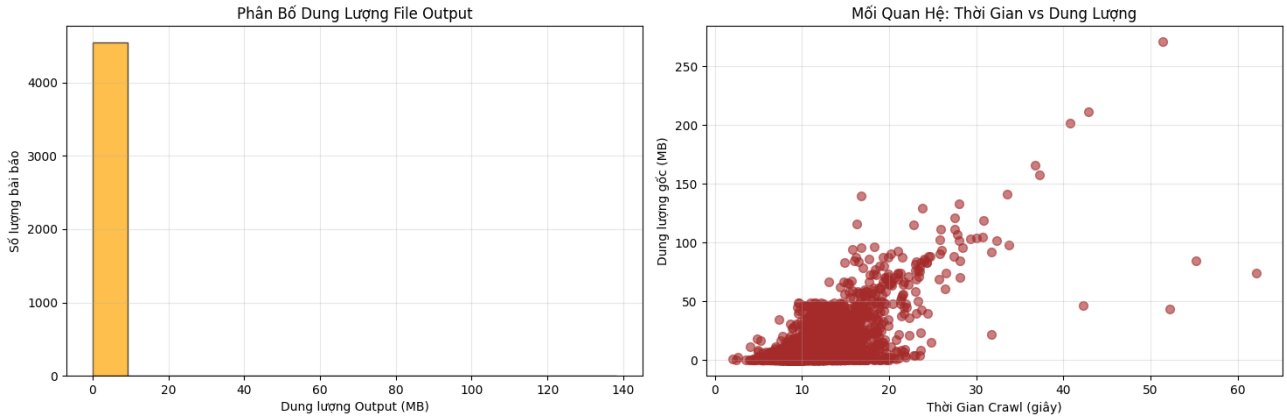


Hình 2

Về sử dụng bộ nhớ, mức RAM trung bình duy trì ở 189.15 MB, với mức tối đa ghi nhận là 221.62 MB. Biểu đồ biến thiên RAM (Hình 1, phải) cho thấy xu hướng tăng tuyến tính theo thời gian, từ khoảng 150 MB lên mức tối đa khi quá trình crawl kết thúc.

Về hiệu quả xử lý dữ liệu đầu ra, pipeline chứng tỏ khả năng giảm dung lượng dữ liệu rất cao. Dung lượng trung bình của output chỉ là 0.49 MB, mặc dù tồn tại một ngoại lệ với dung lượng tối đa 138.28 MB (Hình 3, trái). Gần như toàn bộ các bài báo (khoảng 4500 bài) có kích thước file nhỏ, gần bằng 0 MB. So sánh dung lượng trước và sau xử lý (Hình 2, trái) cho thấy tất cả các điểm dữ liệu nằm dưới

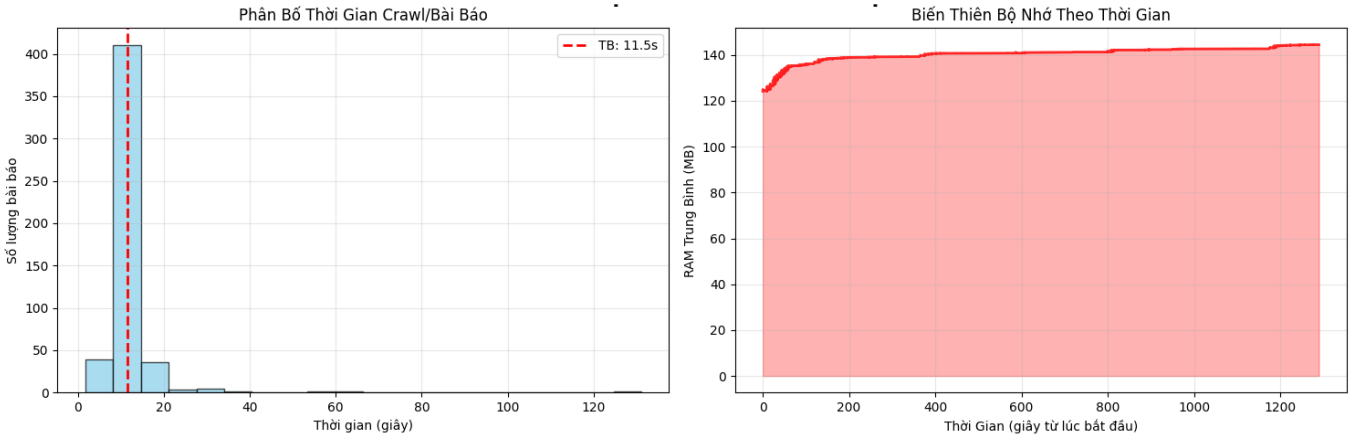
đường chuẩn (y = x), tương ứng với tỷ lệ nén trung bình 24.17%, tức output chỉ chiếm khoảng 24% dung lượng gốc.

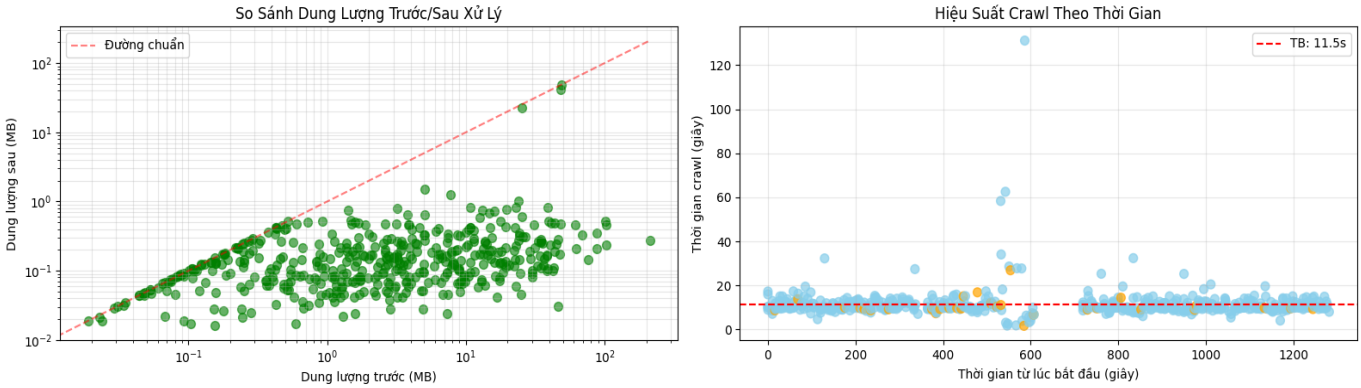


Hình 3

1. Mẫu 500 bài báo

Về hiệu suất thời gian, chương trình đạt tốc độ xử lý trung bình 11.5 giây/bài báo. Tuy nhiên, chỉ số này che giấu một mức độ biến động cao, được phản ánh qua độ lệch chuẩn lớn (7.18 giây) và một khoảng chênh lệch giữa thời gian nhanh nhất (1.70 giây) và chậm nhất (131.11 giây). Phân phối thời gian xử lý (Hình 4, trái) thể hiện lệch phải (right-skewed), với phần lớn tác vụ (khoảng 400 bài) tập trung quanh 11.5 giây.





Hình 4

Về sử dụng bộ nhớ, mức RAM trung bình duy trì ở 140.43 MB gần như tiệm cận mức RAM tối đa là 144.49 MB. Biểu đồ biến thiên RAM (Hình 4, phải) cho thấy xu hướng tăng tuyến tính theo thời gian. Chương trình thực hiện một sự gia tăng bộ nhớ cực kỳ nhanh trong vài giây đầu tiên, sau đó đạt đến trạng thái ổn định, duy trì mức RAM ở khoảng 140-145 MB trong suốt phần còn lại của quá trình thực thi.

C. THỰC HÀNH – DEMO

Link video: VIDEO DEMO LAB 1 SCRAPING

D. TÀI LIỆU THAM KHẢO

[1] Waleed Ammar u.a. The Semantic Scholar Open Research Corpus. https://arxiv.org/abs/1805.02234. Accessed: 2025-11-8.

[2] SemanticScholar Academic Graph API. https://api.semanticscholar.org/api-docs/graph. Accessed: 2025-11-10.