SO SÁNH HIỆU NĂNG VI XỬ LÝ ĐA NHÂN

THÔNG QUA CÁC CHIẾN LƯỢC ĐỒNG BỘ HÓA KHÁC NHAU

Sinh viên: Phùng Hữu Khoa
MSSV: 24022370

Ngày 27 tháng 11 năm 2025

Mục lục

1 Giới thiệu và Mục tiêu Nghiên cứu

1.1 Bối cảnh và Động lực

Trong kỷ nguyên tính toán hiện đại, định luật Moore đang dần chạm tới giới hạn vật lý, dẫn đến sự chuyển dịch xu hướng thiết kế vi xử lý từ việc tăng xung nhịp đơn nhân sang tăng số lượng nhân (Multi-core) và luồng (Multi-thread). Tuy nhiên, việc sở hữu phần cứng mạnh mẽ là chưa đủ; hiệu năng thực tế phụ thuộc chặt chẽ vào khả năng của phần mềm trong việc khai thác tài nguyên song song.

Bài toán **Word Count** (đếm tần suất từ) được lựa chọn làm đối tượng nghiên cứu vì đây là đại diện kinh điển cho mô hình lập trình *MapReduce* – nền tảng của xử lý dữ liệu lớn (Big Data). Mặc dù thoạt nhìn đơn giản, bài toán này lại bộc lộ rõ nét các thách thức cốt lõi của lập trình đa luồng: quản lý I/O, phân chia công việc (Load Balancing), và quan trọng nhất là chi phí đồng bộ hóa (Synchronization Overhead) khi truy cập bộ nhớ chia sẻ.

1.2 Mục tiêu báo cáo

Báo cáo này không chỉ dừng lại ở việc so sánh thời gian chạy, mà sử dụng Word Count như một công cụ đo lường (Benchmark tool) để "khám sức khỏe"và phân tích đặc tính kiến trúc của các vi xử lý hiện đại.

Các mục tiêu cụ thể bao gồm:

1. **Đánh giá Hiệu năng Đơn nhân (Single-core Performance):** Đo lường sức mạnh tính toán thô và chỉ số IPC (Instructions Per Clock) thông qua phiên bản đơn luồng. Đây là cơ sở (Baseline) để tính toán hiệu quả tăng tốc.

2. **Phân tích Chi phí Đồng bộ hóa (Synchronization Penalty):** Thông qua kịch bản *High Contention* (sử dụng Global Mutex), báo cáo sẽ làm rõ mức độ sụt giảm hiệu năng khi các luồng phải tranh chấp tài nguyên, từ đó đánh giá khả năng xử lý độ trễ (Latency) và băng thông liên kết (Interconnect Bandwidth) giữa các nhân CPU.

3. **Đo lường Khả năng Mở rộng (Scalability):** Thông qua kịch bản *High Parallelism* (tối ưu hóa với Local Aggregation), báo cáo đánh giá khả năng tận dụng đa nhân của CPU. Đặc biệt, nghiên cứu sẽ so sánh sự khác biệt giữa kiến trúc lai (Hybrid - P/E cores) của Intel và kiến trúc đồng nhất (Homogeneous) của AMD.

4. **Kiểm chứng lý thuyết Hệ điều hành:** Quan sát tác động của việc quản lý luồng (Thread Management) và chuyển ngữ cảnh (Context Switch) từ Hệ điều hành khi số lượng luồng vượt quá số lượng nhân vật lý (kịch bản Overhead Bound).

1.3 Phạm vi nghiên cứu

Thực nghiệm được tiến hành trên ba đại diện tiêu biểu của các thế hệ vi xử lý khác nhau: **Intel Core Ultra 9** (Arrow Lake - Desktop Flagship), **AMD Ryzen 9** (Zen 4 - Mobile High-end), và **Intel Core i7** (Tiger Lake - Mobile Legacy). Việc so sánh chéo giữa các hệ thống này sẽ cung cấp cái nhìn đa chiều về sự tiến hóa của kiến trúc máy tính trong những năm gần đây.

2 Thiết lập Thí nghiệm (Experimental Setup)

Để đảm bảo tính khách quan và khả năng tái lập (reproducibility), môi trường thử nghiệm được kiểm soát chặt chẽ theo các tiêu chuẩn dưới đây.

2.1 Cấu hình Phần cứng (Hardware Specifications)

Thực nghiệm được triển khai trên ba hệ thống tính toán với các kiến trúc vi xử lý đại diện cho các phân khúc khác nhau của thị trường.

Thành phần	System 1 (Desktop)	System 2 (Laptop)	System 3 (Laptop)
CPU Model	**Intel Core Ultra 9 285K**	**Intel Core i7-11800H**	**AMD Ryzen 9 7940HS**
Codename	Arrow Lake-S	Tiger Lake-H	Phoenix (Zen 4)
Cores/Threads	24 (8P + 16E) / 24	8 / 16	8 / 16
Max Turbo Freq	5.70 GHz	4.60 GHz	5.20 GHz
Lithography	TSMC N3B (Compute Tile)	10 nm SuperFin	TSMC 4nm
RAM	64GB DDR5	16GB DDR4	16GB LPDDR5x
Storage	NVMe SSD Gen5	NVMe SSD Gen4	NVMe SSD Gen4

Bảng 1: Bảng so sánh thông số kỹ thuật phần cứng

2.2 Môi trường Phần mềm (Software Environment)

Do sự khác biệt về Hệ điều hành gốc (Windows/Linux), môi trường thực thi được chuẩn hóa thông qua **WSL 2 (Windows Subsystem for Linux)**.

- **Hệ điều hành:** Ubuntu 22.04.3 LTS (chạy trên nền WSL 2).
- **Hệ thống tệp (Filesystem):** Dữ liệu Benchmark được lưu trữ trực tiếp trên phân vùng gốc (`ext4`) của Linux. Điều này giúp loại bỏ nút thắt cổ chai về I/O (I/O Bottleneck) thường gặp khi truy xuất qua lại giữa Linux và phân vùng NTFS của Windows.
- **Trình biên dịch:** GCC 11.4.0 với các cờ tối ưu hóa:
  - `-O3`: Tối ưu hóa hiệu năng cao nhất.
  - `-pthread`: Hỗ trợ thư viện luồng POSIX.
  - `-std=gnu99`: Chuẩn C99 mở rộng.
- **Automation:** Python 3.10 dùng cho kịch bản chạy tự động.

3 Phương pháp Nghiên cứu (Methodology)

Để tách biệt và đánh giá từng khía cạnh hiệu năng của vi xử lý, tôi đã thiết kế ba phiên bản chương trình Word Count với các chiến lược đồng bộ hóa khác biệt, kết hợp với quy trình đo lường nghiêm ngặt.

3.1 Các phiên bản thuật toán

3.1.1 Baseline: Đơn luồng (lwords)

Phiên bản này thực hiện xử lý tuần tự truyền thống:

- **Cơ chế:** Một luồng duy nhất đọc lần lượt từng file và cập nhật vào một danh sách liên kết đơn.
- **Mục tiêu:** Thiết lập mốc thời gian chuẩn (T_{base}) để tính toán chỉ số tăng tốc.
- **Ý nghĩa phần cứng:** Phản ánh sức mạnh đơn nhân (Single-core Performance), phụ thuộc chủ yếu vào xung nhịp và chỉ số IPC.

3.1.2 High Contention: Đa luồng với Khóa mịn (pwords_v1)

Phiên bản này áp dụng chiến lược khóa đơn giản (Naive locking strategy):

- **Cơ chế:** Mỗi luồng xử lý một file, nhưng mọi thao tác thêm từ đều thực hiện trực tiếp trên **một danh sách toàn cục** (Global List).
- **Điểm nghẽn:** Mutex toàn cục bị khóa/mở hàng triệu lần, gây ra sự tranh chấp tài nguyên cực đoan.
- **Ý nghĩa phần cứng:** Stress test hệ thống liên kết nội bộ (Interconnects) và độ trễ của giao thức Cache Coherency (như MESI).

3.1.3 Optimized: Đa luồng với Gộp cục bộ (pwords_v2)

Phiên bản này tối ưu hóa dựa trên tư tưởng MapReduce:

- **Cơ chế:**
  1. *Phase 1 (Local Processing):* Mỗi luồng đếm từ vào một danh sách riêng (Lock-free).
  2. *Phase 2 (Global Merge):* Sau khi xong, luồng yêu cầu khóa Mutex toàn cục **đúng một lần** để gộp kết quả.
- **Ý nghĩa phần cứng:** Đo băng thông xử lý thực tế của đa nhân (Multi-core Throughput) khi loại bỏ chi phí đồng bộ hóa.

3.2 Quy trình Đo lường (Measurement Protocol)

Quy trình benchmark được tự động hóa bằng script `benchmark-full.py` với các nguyên tắc:

1. **Warm-up:** Chạy mồi 1 lần trước khi đo để nạp thư viện vào RAM và làm nóng bộ nhớ đệm (OS Page Cache).
2. **Lặp lại (Iterations):** Mỗi bài test được chạy lặp lại **3 lần**. Kết quả cuối cùng là giá trị trung bình cộng (Mean) để loại bỏ sai số ngẫu nhiên.
3. **Kiểm tra tính đúng đắn:** Output của chương trình đa luồng được so sánh (checksum) với đơn luồng để đảm bảo không xảy ra Race Condition.

3.3 Chỉ số Đánh giá (Metrics)

Hiệu năng được lượng hóa qua ba chỉ số chính:

- **Thời gian thực thi (T):** Đo bằng "Wall-clock time"(giây).
- **Hệ số tăng tốc (Speedup):** Tỷ lệ cải thiện tốc độ so với đơn luồng.

$$S = \frac{T_{single}}{T_{multi}}$$

- **Hiệu suất (Efficiency):** Mức độ tận dụng tài nguyên trên số luồng (P).

$$E = \frac{S}{P} \times 100\%$$

4 Đặc tả Bộ dữ liệu (Dataset Characteristics)

Để đảm bảo tính khách quan và phản ánh đúng các thách thức trong xử lý ngôn ngữ tự nhiên (NLP) cơ bản, dữ liệu đầu vào được chuẩn hóa về định dạng và dung lượng.

4.1 Nguồn gốc và Định dạng (Source & Format)

Toàn bộ dữ liệu được trích xuất từ thư viện **Project Gutenberg** - kho lưu trữ các tác phẩm văn học kinh điển thuộc phạm vi công cộng.

- **Định dạng:** Plain Text (UTF-8/ASCII). Đây là dữ liệu phi cấu trúc (Unstructured Data), đòi hỏi CPU phải thực hiện quét tuần tự, xử lý chuỗi và quản lý bộ nhớ động liên tục.
- **Nội dung:** Bao gồm các tiểu thuyết tiếng Anh, văn bản khoa học và các bài luận. Đặc điểm của loại dữ liệu này là độ dài từ và cấu trúc câu không đồng nhất, giúp tránh được việc CPU dự đoán nhánh (Branch Prediction) quá dễ dàng.
- **Tổng dung lượng:** Được cố định ở mức $\approx$ **62.5 MB** cho mỗi kịch bản kiểm thử. Kích thước này đủ lớn để phép đo thời gian có ý nghĩa (tránh sai số do độ phân giải đồng hồ hệ thống).

4.2 Các kịch bản Kiểm thử (Test Scenarios)

Dữ liệu 62.5 MB được chia thành 3 cấu hình (Dataset) khác nhau nhằm kích hoạt các hành vi riêng biệt của Vi xử lý và Hệ điều hành:

Bảng 2: Thông số chi tiết các bộ dữ liệu thử nghiệm

Dataset	Số File	Kích thước/File	Mục tiêu Stress Test
Set A	4	15.6 MB	**CPU Bound**: Tính toán nặng, ít overhead.
Set B	100	625 KB	**Overhead Bound**: Stress test OS scheduler.
Set C	18	Ngẫu nhiên	**Load Imbalance**: Cân bằng tải.

4.2.1 Set A - CPU Bound (Mô phỏng Xử lý Lô)

Dữ liệu được gộp thành 4 file lớn (tương ứng khoảng 15MB/file).

- **Đặc điểm:** Số lượng file ít hơn số lượng luồng của CPU.
- **Mục tiêu:** Tối đa hóa thời gian tính toán thuần túy và giảm thiểu thời gian mở/đóng file. Đây là kịch bản lý tưởng để đo băng thông xử lý (Throughput) tối đa của các nhân P-core và hiệu quả của bộ nhớ đệm L2/L3.

4.2.2 Set B - Overhead Bound (Mô phỏng Máy chủ Web)

Dữ liệu được chia nhỏ thành 100 file (khoảng 625KB/file).

- **Đặc điểm:** Số lượng tác vụ (files) lớn gấp nhiều lần số lượng nhân CPU.
- **Mục tiêu:** Tạo áp lực lên bộ lập lịch (Scheduler) của Hệ điều hành. CPU buộc phải thực hiện chuyển đổi ngữ cảnh (Context Switch) liên tục. Kịch bản này kiểm tra khả năng quản lý luồng và độ trễ (Latency) của kiến trúc vi xử lý.

4.2.3 Set C - Load Imbalance (Mô phỏng Thực tế)

Dữ liệu bao gồm các file có kích thước ngẫu nhiên: từ các bài thơ ngắn (vài KB) đến các tiểu thuyết dài (vài MB).

- **Đặc điểm:** Phân bố dữ liệu không đồng đều (Skewed distribution).
- **Mục tiêu:** Gây ra hiện tượng "Straggler một số luồng hoàn thành sớm và phải chờ đợi các luồng xử lý file lớn. Đây là bài kiểm tra khắc nghiệt nhất đối với kiến trúc lai (Hybrid Architecture) của Intel, nơi việc phân chia nhầm một file lớn vào nhân E-core sẽ gây sụt giảm hiệu năng nghiêm trọng.

5 Kết quả Thực nghiệm và Đánh giá

Phần này trình bày các số liệu đo đạc thực tế và phân tích sâu về hành vi của kiến trúc vi xử lý dưới các áp lực tính toán khác nhau.

5.1 Đánh giá Sức mạnh Đơn nhân (Single-core Performance)

Thử nghiệm trên tập dữ liệu **Set A** với phiên bản `lwords`. Đây là thước đo sức mạnh xử lý thô của một nhân CPU (Raw Compute Power) khi loại bỏ hoàn toàn yếu tố đa luồng.

CPU Model	Kiến trúc	Thời gian (s)	So sánh
Intel Core Ultra 9 285K	Arrow Lake	**189.37**	Chuẩn (1.00x)
Intel Core i7-11800H	Tiger Lake	315.63	Chậm hơn 1.67 lần
AMD Ryzen 9 7940HS	Zen 4	355.54	Chậm hơn 1.88 lần

Bảng 3: Thời gian thực thi đơn luồng (Thấp hơn là tốt hơn)

Nhận xét: Intel Core Ultra 9 285K thể hiện sự vượt trội tuyệt đối về sức mạnh đơn nhân, nhanh hơn gần gấp đôi so với AMD Ryzen 9 7940HS.

5.2 Phân tích Chi phí Đồng bộ hóa (Synchronization Overhead)

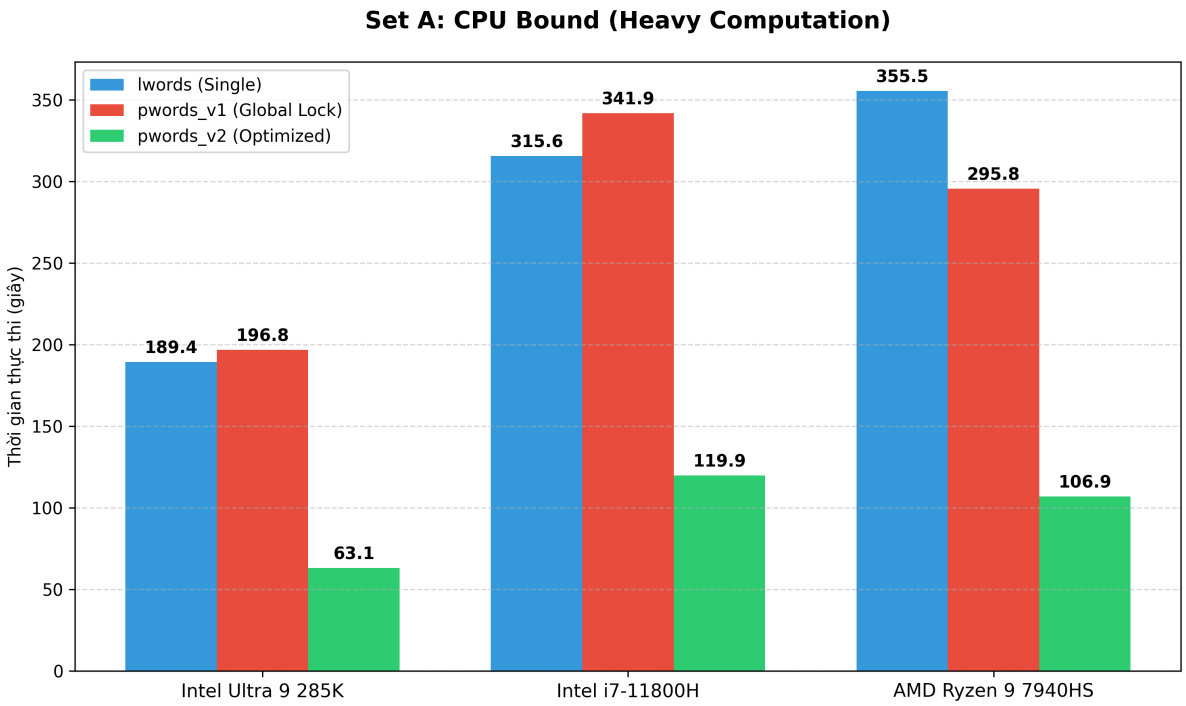
Thử nghiệm với phiên bản `pwords_v1` (dùng Global Mutex) trên **Set A**. Mục tiêu là kiểm tra khả năng chịu đựng của hệ thống Bus/Interconnect khi xảy ra tranh chấp tài nguyên cực đoan.

CPU Model	Số luồng	Thời gian (s)	Speedup	Trạng thái
Intel Core Ultra 9 285K	24	196.83	0.96x	*Chậm đi*
Intel Core i7-11800H	16	341.92	0.92x	*Chậm đi*
AMD Ryzen 9 7940HS	16	**295.75**	**1.20x**	*Tăng tốc*

Bảng 4: Hiệu năng trong môi trường tranh chấp khóa (pwords_v1)

5.3 Đánh giá Khả năng Mở rộng (Scalability - Set A)

Sử dụng phiên bản tối ưu `pwords_v2` trên **Set A** (các file kích thước lớn, đồng đều). Đây là kịch bản lý tưởng để đo băng thông tính toán tối đa.



Hình 1: Biểu đồ so sánh hiệu năng trên tập dữ liệu Set A (CPU Bound)

Từ Hình 1, ta thấy rõ sự sụt giảm thời gian đáng kể (cột màu xanh lá) khi áp dụng thuật toán tối ưu.

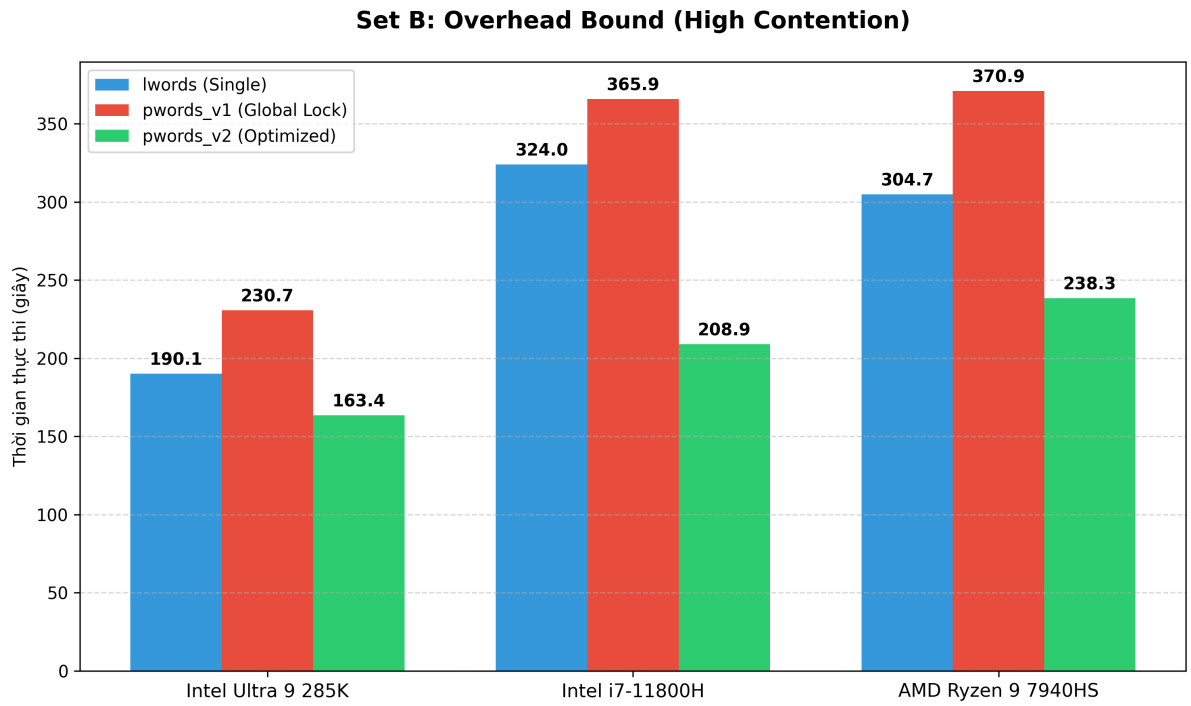
CPU Model	Luồng (P)	Time v2 (s)	Speedup (S)	Hiệu suất (E)	Đánh giá
Intel Core Ultra 9	24	**63.06**	3.00x	**12.5%**	Nhanh nhưng lãng phí
Intel Core i7-11800H	16	119.88	2.63x	16.4%	Trung bình
AMD Ryzen 9	16	106.91	**3.33x**	**20.8%**	Hiệu quả nhất

Bảng 5: Tốc độ và Hiệu suất lý tưởng (Set A - pwords_v2)

Phân tích kiến trúc: Mặc dù Intel Core Ultra 9 nhanh nhất (63s), hiệu suất chỉ đạt 12.5

5.4 Tác động của Overhead Hệ điều hành (Set B)

Thử nghiệm với **Set B** (100 file nhỏ) để kiểm tra chi phí quản lý luồng của OS.



Hình 2: Biểu đồ hiệu năng trên Set B: Tác động của Overhead quản lý luồng

Như quan sát trong Hình 2, thời gian xử lý (cột xanh lá) tăng lên đáng kể so với Set A dù tổng dung lượng dữ liệu tương đương.

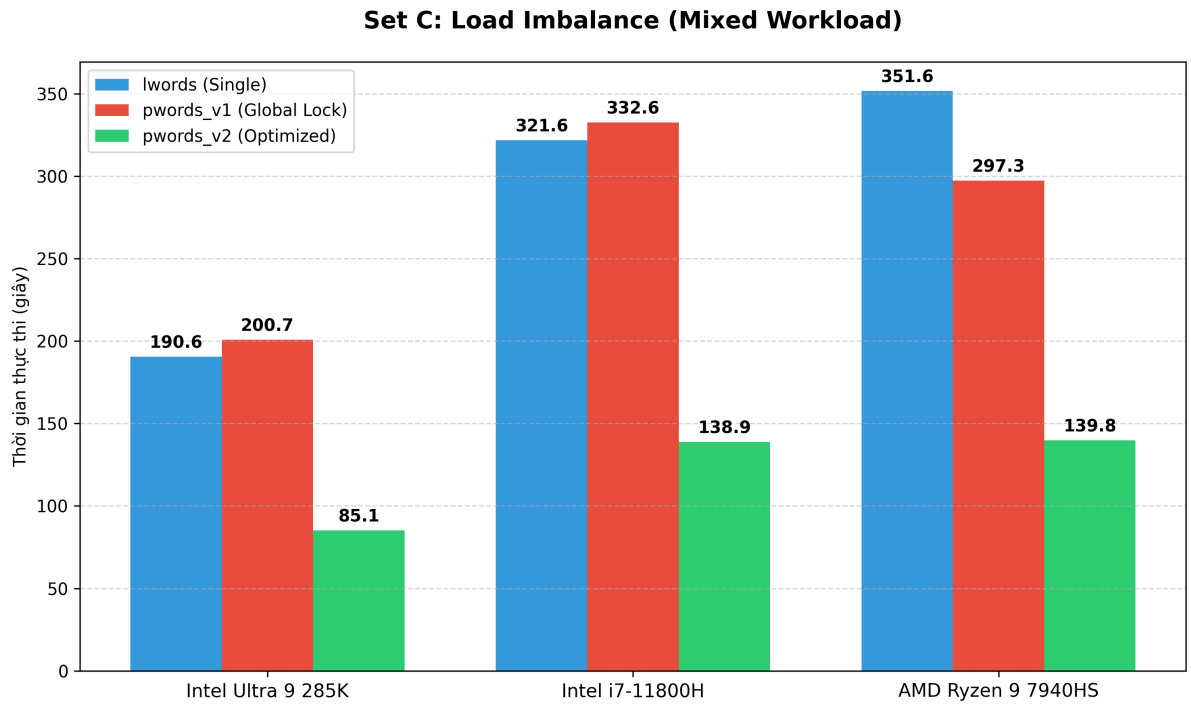
CPU Model	Speedup (Set A)	Speedup (Set B)	Mức sụt giảm
Intel Core Ultra 9	3.00x	1.16x	-61.3%
AMD Ryzen 9	3.33x	1.28x	-61.5%
Intel Core i7	2.63x	1.55x	-41.0%

Bảng 6: Sự sụt giảm hiệu năng với nhiều file nhỏ (Set B)

Nhận xét: Định luật Amdahl phát huy tác dụng rõ rệt. Khả năng tăng tốc bị giới hạn bởi thời gian khởi tạo luồng và gộp kết quả.

5.5 Mô phỏng Thực tế & Mất cân bằng tải (Set C)

Đây là kịch bản thực tế nhất: các file có kích thước ngẫu nhiên, gây ra hiện tượng mất cân bằng tải (Load Imbalance).



Hình 3: Biểu đồ Set C: Mất cân bằng tải và hiện tượng Straggler

CPU Model	Speedup (Set A)	Speedup (Set C)	Hiệu suất (Set C)	Sụt giảm
Intel Core Ultra 9	3.00x	2.24x	**9.3%**	-25.3%
AMD Ryzen 9	3.33x	2.51x	**15.7%**	-24.6%
Intel Core i7	2.63x	2.32x	14.5%	-11.8%

Bảng 7: Ảnh hưởng của mất cân bằng tải lên hiệu suất (Set C)

Đánh giá chuyên sâu: Trong Hình 3, ta thấy Intel Ultra 9 (Arrow Lake) bị ảnh hưởng nặng nề bởi kiến trúc lai. Nếu file lớn rơi vào E-core, toàn bộ hệ thống bị kéo chậm, khiến hiệu suất tụt xuống dưới 10%. Ngược lại, AMD Ryzen 9 duy trì sự ổn định tốt nhất nhờ các nhân đồng đều.

6 Thảo luận và Phân tích Chuyên sâu

Dựa trên các dữ liệu về Hiệu suất (E) và khả năng chịu tải trong các kịch bản thực tế (Set C), chúng ta có thể vẽ nên bức tranh toàn cảnh về hành vi của các vi xử lý hiện đại.

6.1 Nghịch lý Hiệu năng trên Kiến trúc Lai (The Hybrid Paradox)

Intel Core Ultra 9 285K đại diện cho xu hướng thiết kế "Big.LITTLE"(P-cores và E-cores).

- **Sự đánh đổi về mặt Năng lượng (Power Budget):** Cần lưu ý một yếu tố khách quan: Intel Core Ultra 9 là vi xử lý Desktop với ngân sách năng lượng (TDP > 125W) lớn gấp nhiều lần hai đối thủ Mobile (TDP $\approx$ 45W). Tuy nhiên, điều này càng làm nổi bật vấn đề về *Hiệu suất thực tế*: dù tiêu thụ nhiều điện năng hơn và sở hữu số lượng nhân vượt trội, hiệu quả tính toán trên mỗi Watt và mỗi nhân lại thấp hơn đáng kể trong tác vụ này.
- **Vấn đề lập lịch (Scheduling Penalty):** Trong kịch bản Set C (file lớn nhỏ lộn xộn), hiệu suất sử dụng tài nguyên của Ultra 9 tụt xuống mức kỷ lục (**9.3%**). Nguyên nhân là do Hệ

điều hành gặp khó khăn trong việc phân phối "đúng người đúng việc". Nếu một file lớn (heavy task) vô tình bị gán cho E-core, trong khi P-core đã xử lý xong và rảnh rỗi, toàn bộ hệ thống bị kéo chậm lại bởi nhân yếu nhất (Straggler problem).

6.2 Sự vượt trội về tính Nhất quán của AMD Zen 4

AMD Ryzen 9 7940HS với thiết kế nhân đồng nhất (Homogeneous Cores) đã chứng minh sự ưu việt trong bài toán MapReduce:

- **Khả năng mở rộng tuyến tính:** Với Speedup đạt **3.33x** và Hiệu suất **20.8%**, AMD cho thấy khả năng chia tải tự nhiên (Natural Load Balancing) tốt hơn. Mọi nhân đều có sức mạnh ngang nhau, giúp các luồng kết thúc gần như cùng lúc, giảm thiểu thời gian chờ (Idle time).
- **Hiệu quả Interconnect:** Trong thử nghiệm `pwords_v1` (tranh chấp khóa cao), AMD là chip duy nhất tăng tốc (1.20x). Điều này gợi ý rằng kiến trúc Infinity Fabric xử lý các tín hiệu đồng bộ hóa (Cache Coherency traffic) hiệu quả hơn so với đối thủ trong bối cảnh này.

6.3 Định luật Amdahl và Giới hạn của Phần mềm

Kết quả từ **Set B** (100 file nhỏ) là minh chứng sống động cho định luật Amdahl:

- Khi phần công việc tuần tự (mở file, khởi tạo luồng, gộp kết quả) chiếm tỷ trọng lớn, việc thêm nhân (từ 16 lên 24) trở nên vô nghĩa.
- Sự sụt giảm hiệu năng hơn **60%** ở cả hai chip đời mới nhất cho thấy: Để tăng tốc ứng dụng, việc tối ưu hóa thuật toán (chuyển từ Global Lock sang Local Aggregation) mang lại lợi ích gấp hàng chục lần so với việc chỉ đơn thuần nâng cấp phần cứng.

7 Kết luận

Từ quá trình thực nghiệm, báo cáo rút ra 4 kết luận cốt lõi:

1. **Tốc độ $\neq$ Hiệu quả:** Intel Core Ultra 9 là CPU *nhanh nhất* (nhờ sức mạnh phần cứng thô và điện năng tiêu thụ cao), nhưng AMD Ryzen 9 là CPU *hiệu quả nhất* (về khả năng cân bằng tải và tận dụng tài nguyên).
2. **Kiến trúc Đồng nhất ổn định hơn cho HPC:** Đối với các thuật toán song song tổng quát chưa được tối ưu riêng, kiến trúc đồng nhất (như Zen 4) cung cấp hiệu năng dự đoán được (predictable performance) và ít bị ảnh hưởng bởi lỗi lập lịch hơn kiến trúc lai.
3. **Chiến lược phần mềm là yếu tố quyết định:** Sự chênh lệch giữa phiên bản `v1` (chậm hơn đơn luồng) và `v2` (nhanh gấp 3 lần) khẳng định rằng: Trong lập trình đa luồng, cách quản lý dữ liệu và đồng bộ hóa quan trọng hơn nhiều so với số lượng nhân của CPU.
4. **Hướng phát triển tiếp theo:** Kết quả nghiên cứu gợi mở nhu cầu cấp thiết về các bộ lập lịch (Schedulers) thông minh hơn trong Hệ điều hành, có khả năng nhận diện đặc tính luồng (Thread Characterization) để phân phối việc nặng vào P-cores một cách chính xác, thay vì phó mặc cho các thuật toán lập lịch "mù"(blind scheduling) hiện tại.

8 Mã nguồn tham khảo

Toàn bộ mã nguồn dự án, bao gồm chương trình C, các kịch bản kiểm thử tự động (Python/Bash scripts) và dữ liệu benchmark thô (file JSON) từ 3 máy tính thử nghiệm đều được lưu trữ công khai và minh bạch.

Giảng viên và người đọc quan tâm có thể truy cập để tải về, kiểm tra mã nguồn hoặc tái lập (reproduce) các kết quả thực nghiệm tại địa chỉ GitHub sau:

https://github.com/Khoaph1709/CPU-Benchmark.git