

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO BÀI KIỂM TRA GIỮA KỲ BIGDATA

TÌM SỐ NGUYÊN TỐ BẰNG
GIẢI THUẬT SÀNG ERATOSTHENES

Giảng viên hướng dẫn: **THS. HỒ KHÔI**
Sinh viên thực hiện: **NGÔ CÔNG HUÂN**
MSSV: **2000002680**
Môn học: **Dữ liệu lớn**
Khoá: **2020**

Tp.HCM, tháng 8 năm 2023

MỤC LỤC

DANH MỤC BẢNG BIỂU	1
DANH MỤC HÌNH ẢNH	2
CHƯƠNG 1: TỔNG QUAN VỀ GIẢI THUẬT SÀNG ERATOSTHENES	3
1.1. Số nguyên tố	3
1.2. Giải thuật sàng Eratosthenes	3
1.3. Giải thuật song song Paralell Eratosthenes	4
CHƯƠNG 2: TĂNG TỐC (SPEED UP) VÀ ĐỘ HIỆU QUẢ (EFFICIENCY).....	5
2.1. Speedup (Tăng tốc)	5
2.2. Efficiency (Độ hiệu quả)	5
CHƯƠNG 3: THỰC NGHIỆM VÀ KẾT QUẢ	6
3.1. Thực nghiệm thuật toán tuần tự	6
3.2. Thực nghiệm thuật toán song song trên máy thật	6
3.3. Thực nghiệm thuật toán song song trên máy ảo	10
3.4. Phân tích Speedup (tăng tốc) và Efficiency (độ hiệu quả).....	12

DANH MỤC BẢNG BIỂU

Bảng 3.1. Bảng khảo sát Speedup và Efficiency	12
---	----

DANH MỤC HÌNH ẢNH

Hình 1.1. Minh họa thuật toán Sàng Eratosthenes	3
Hình 3.1. Hình ảnh kết quả thực thi thuật toán tuần tự Eratosthenes.....	6
Hình 3.2. Hình ảnh tệp ParallelEratosthenes.jar.....	7
Hình 3.3. Hình ảnh tệp đầu vào input.txt	7
Hình 3.4. Hình ảnh khởi động Hadoop	8
Hình 3.5. Hình ảnh thư mục input.txt trên HDFS	8
Hình 3.6. Hình ảnh chạy chương trình trên Hadoop	8
Hình 3.7. Hình ảnh kết quả tệp đầu ra trên HDFS	9
Hình 3.8. Hình ảnh thời gian thực thi chương trình	9
Hình 3.9. Hình ảnh thời gian thực thi trên máy ảo với 2 processors.....	10
Hình 3.10. Hình ảnh thời gian thực thi trên máy ảo với 4 processors.....	10
Hình 3.11. Hình ảnh thời gian thực thi trên máy ảo với 6 processors.....	11
Hình 3.12. Hình ảnh thời gian thực thi trên máy ảo với 8 processors.....	11

CHƯƠNG 1: TỔNG QUAN VỀ GIẢI THUẬT SÀNG ERATOSTHENES

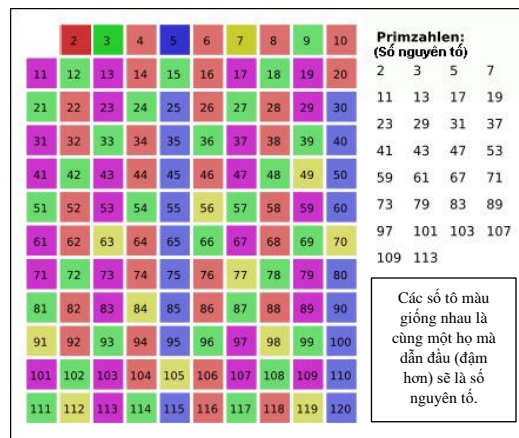
1.1. Số nguyên tố

Số nguyên tố là những số tự nhiên lớn hơn 1 chỉ chia hết cho 1 và chính nó.

- Ví dụ: 2, 3, 5, 7,...

1.2. Giải thuật sàng Eratosthenes

Sàng Eratosthenes là thuật toán tìm các số nguyên tố bé hơn một số n ($n \geq 2$).



Hình 1.1. Minh họa thuật toán Sàng Eratosthenes

Mô tả thuật toán:

- Bước 1: Tạo 1 danh sách các số tự nhiên liên tiếp từ 2 đến n : (2, 3, 4,..., n).
- Bước 2: Giả sử tất cả các số trong danh sách đều là số nguyên tố. Trong đó, $p = 2$ là số nguyên tố đầu tiên.
- Bước 3: Tất cả các bội số của p : $2p, 3p, 4p, \dots$ sẽ bị đánh dấu vì không phải là số nguyên tố.
- Bước 4: Tìm các số còn lại trong danh sách mà chưa bị đánh dấu và phải lớn hơn p và nhỏ hơn hoặc bằng căn bậc 2 của n . Nếu không còn số nào, dừng tìm kiếm. Ngược lại, gán cho p giá trị bằng số nguyên tố tiếp theo và quay lại bước 3.

Khi giải thuật kết thúc, tất các số chưa bị đánh dấu trong danh sách là các số nguyên tố cần tìm. (Xem Source code đính kèm: **Eratosthenes.java**)

1.3. Giải thuật song song Paralell Eratosthenes

Thuật toán Sàng Eratosthenes nhanh hơn thuật toán tuần tự, nhưng nó vẫn có thể được cải thiện bằng cách song song hóa tính toán. Để song song hóa thuật toán Sàng Eratosthenes sử dụng Hadoop MapReduce, chúng ta có thể chia khoảng các số nguyên thành các phạm vi con nhỏ hơn và thực hiện phép sàng trên mỗi phạm vi con một cách song song.

Dưới đây là một cái nhìn tổng quan về thuật toán song song sử dụng Hadoop MapReduce:

- Giai đoạn Map (Map Phase):
 - Tạo một công việc MapReduce với số lượng node worker tùy ý và chia khoảng các số nguyên từ 2 đến n thành các phạm vi con nhỏ hơn.
 - Giao mỗi phạm vi con cho một node worker, tại đây node worker sẽ thực hiện phép sàng Eratosthenes trên phạm vi con được giao. Node worker sẽ tạo một danh sách các số nguyên tố trong phạm vi con và đưa ra danh sách này như một cặp key-value, với key là null (hoặc một giá trị duy nhất) và value là danh sách các số nguyên tố.
 - Các node worker sẽ gửi các cặp key-value này về master node (node chủ).
- Giai đoạn Reduce (Reduce Phase):
 - Master node sẽ nhận các cặp key-value từ các node worker.
 - Master node sẽ gộp danh sách các số nguyên tố từ các cặp key-value vào một danh sách duy nhất các số nguyên tố.
 - Danh sách các số nguyên tố đã gộp được trả về là kết quả cuối cùng của thuật toán.

Thuật toán song song này sẽ cải thiện hiệu suất của thuật toán Sàng Eratosthenes khi sử dụng nhiều node worker để xử lý phạm vi con một cách song song. Việc song song hóa giúp chia nhỏ vấn đề thành các phần nhỏ hơn và xử lý chúng cùng một lúc, làm tăng hiệu suất tổng thể của thuật toán.

(Xem Source code đính kèm: **GeneratePrimesMR.java**)

CHƯƠNG 2: TĂNG TỐC (SPEED UP) VÀ ĐỘ HIỆU QUẢ (EFFICIENCY)

Trong thuật toán song song, "tăng tốc" (speedup) và "hiệu quả" (efficiency) là hai thước đo hiệu năng quan trọng giúp đánh giá hiệu quả của việc song song hóa. Những thước đo này thường được sử dụng để đánh giá mức độ cải thiện thời gian thực thi của thuật toán song song so với phiên bản tuần tự và cách mà tài nguyên song song được tận dụng một cách hiệu quả.

2.1. Speedup (Tăng tốc)

Tăng tốc đo lường sự cải thiện hiệu năng tương đối bằng cách thực thi một thuật toán song song trên nhiều bộ xử lý (hoặc lõi) so với việc thực thi cùng một thuật toán tuần tự trên một bộ xử lý duy nhất. Tăng tốc được định nghĩa là tỷ lệ giữa thời gian thực thi tuần tự và thời gian thực thi song song. Độ tăng tốc (S) được tính như sau:

$$S = \frac{T_S}{T_P}$$

Trong đó:

- T_S : Thời gian thực hiện thuật toán tuần tự
- T_P : Thời gian thực hiện thuật toán song song

2.2. Efficiency (Độ hiệu quả)

Độ hiệu quả là tỷ lệ giữa tăng tốc đạt được và số lượng bộ xử lý được sử dụng. Độ hiệu quả (E) được tính như sau:

$$E = \frac{S}{p}$$

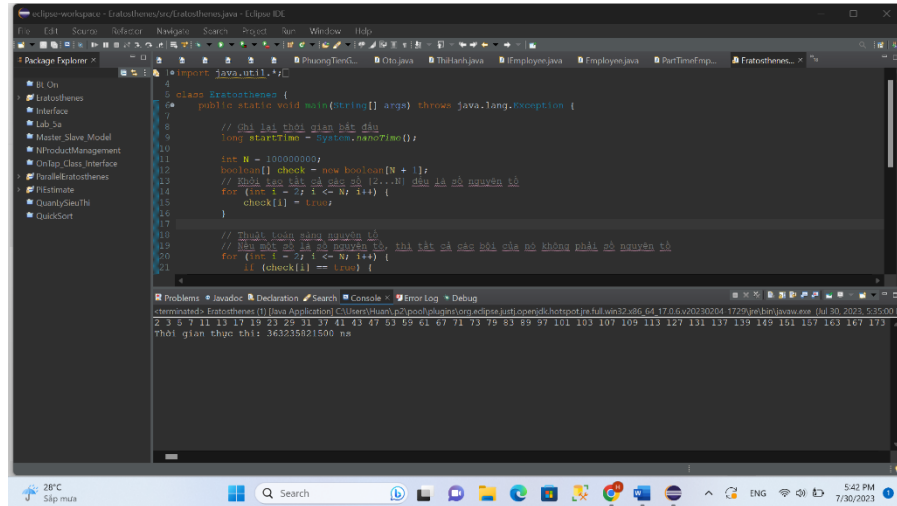
Trong đó:

- S : Độ tăng tốc (Speedup)
- p : Số bộ xử lý tham gia tính toán

CHƯƠNG 3: THỰC NGHIỆM VÀ KẾT QUẢ

3.1. Thực nghiệm thuật toán tuần tự

Thực nghiệm thuật toán tuần tự Sàng Eratosthenes bằng ngôn ngữ Java trên Eclipse.



```
import java.util.*;

class Eratosthenes {
    public static void main(String[] args) throws java.lang.Exception {
        // Ghi lại thời gian bắt đầu
        long startTime = System.nanoTime();

        int N = 100000000;
        boolean[] check = new boolean[N + 1];
        // Khởi tạo tất cả các số từ 2 đến N đều là số nguyên tố
        for (int i = 2; i <= N; i++) {
            check[i] = true;
        }

        // Thuật toán sàng nguyên tố
        // Nếu một số là số nguyên tố, thì tất cả các số bội của nó không phải là số nguyên tố
        for (int i = 2; i <= N; i++) {
            if (check[i] == true) {
                for (int j = i * i; j <= N; j += i) {
                    check[j] = false;
                }
            }
        }

        // In ra các số nguyên tố
        for (int i = 2; i <= N; i++) {
            if (check[i] == true) {
                System.out.print(i + " ");
            }
        }
    }
}

// Kết quả thực thi:
// Thời gian thực thi: 363235821500 ns
```

Hình 3.1. Hình ảnh kết quả thực thi thuật toán tuần tự Eratosthenes

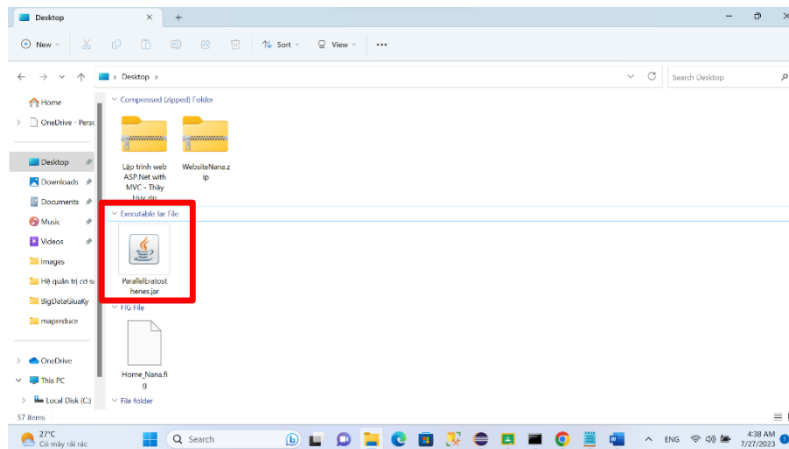
Dựa vào kết quả trên, ta thấy được nếu thực thi thuật toán tuần tự Sàng Eratosthenes để tìm số nguyên tố bé hơn 100.000.000 thì thời gian để chạy chương trình là 363235821500 nanoseconds = 363.24 sec.

3.2. Thực nghiệm thuật toán song song trên máy thật

Quy trình thực nghiệm thuật toán song song Paralell Eratosthenes trên máy thật với số lượng processor = 12.

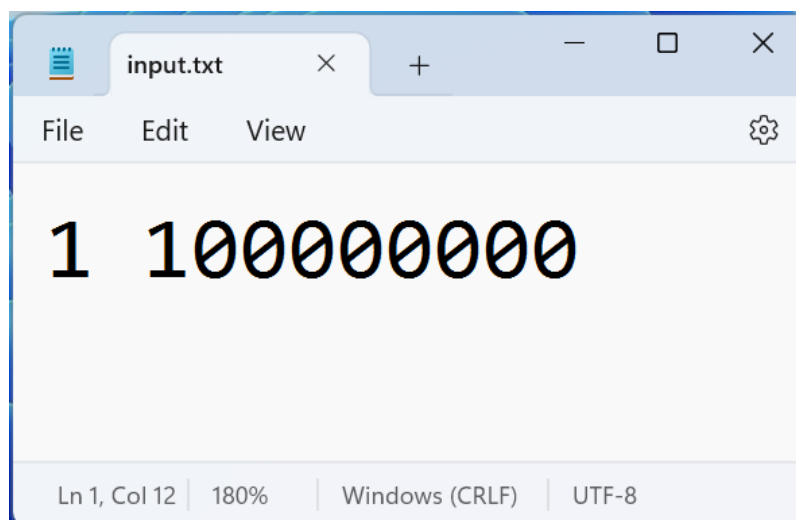
- **Bước 1:** Tạo source code thuật toán song song Paralell Eratosthenes bằng ngôn ngữ Java (xem file đính kèm **GeneratePrimesMR.java**).

- **Bước 2:** Chuyển đổi thành tệp Jar: **ParallelEratosthenes.jar**.



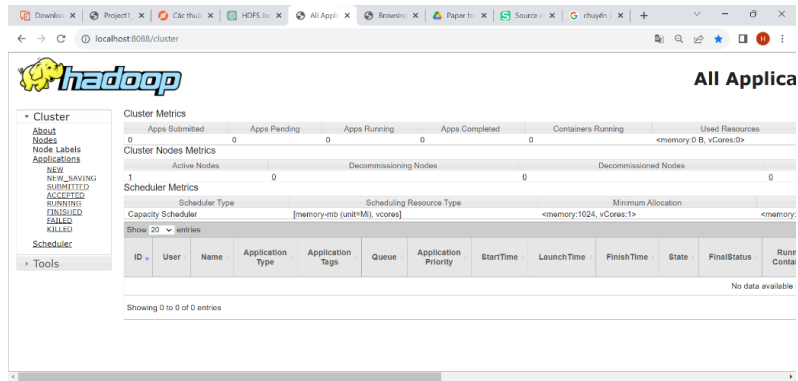
Hình 3.2. Hình ảnh tệp ParallelEratosthenes.jar

- **Bước 3:** Tạo tệp **input.txt** gồm điểm bắt đầu và điểm kết thúc.



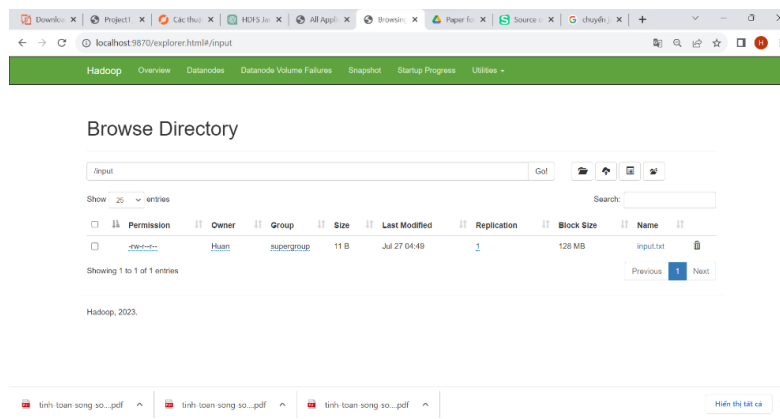
Hình 3.3. Hình ảnh tệp đầu vào input.txt

- **Bước 4:** Khởi động Hadoop (chạy với quyền quản trị viên).



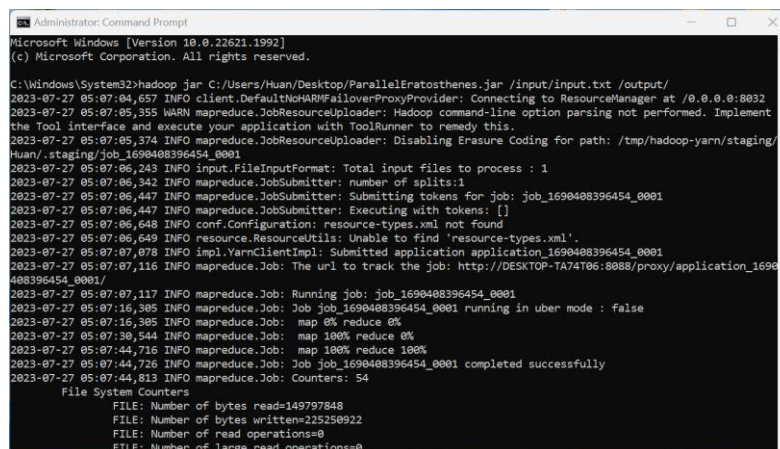
Hình 3.4. Hình ảnh khởi động Hadoop

- **Bước 5:** Tạo thư mục input trên HDFS và copy tệp input.txt vào.



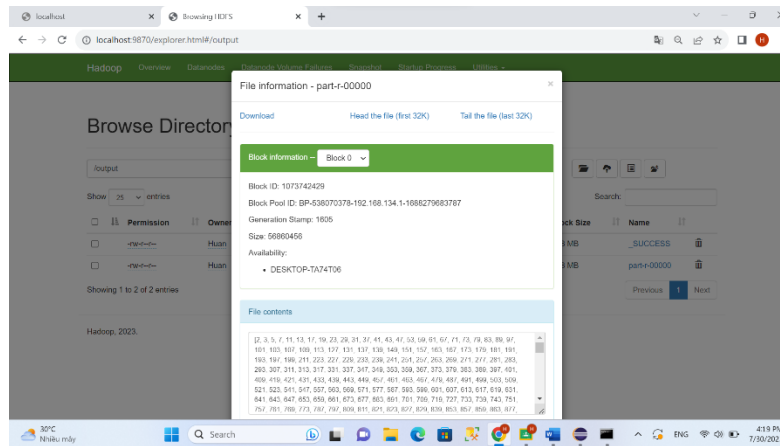
Hình 3.5. Hình ảnh thư mục input.txt trên HDFS

- **Bước 6:** Mở command prompt (chạy với quyền quản trị viên) và chạy chương trình.



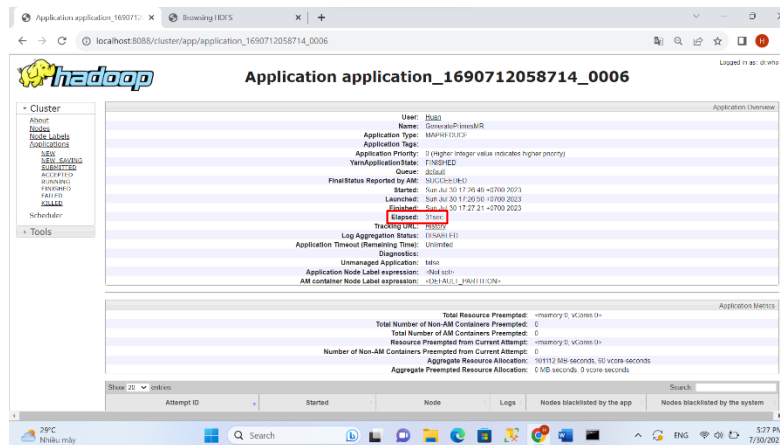
Hình 3.6. Hình ảnh chạy chương trình trên Hadoop

- **Bước 7: Xem kết quả của tệp đầu ra trên HDFS.**



Hình 3.7. Hình ảnh kết quả tệp đầu ra trên HDFS

- **Bước 8: Xem thời gian thực thi chương trình (31 sec).**

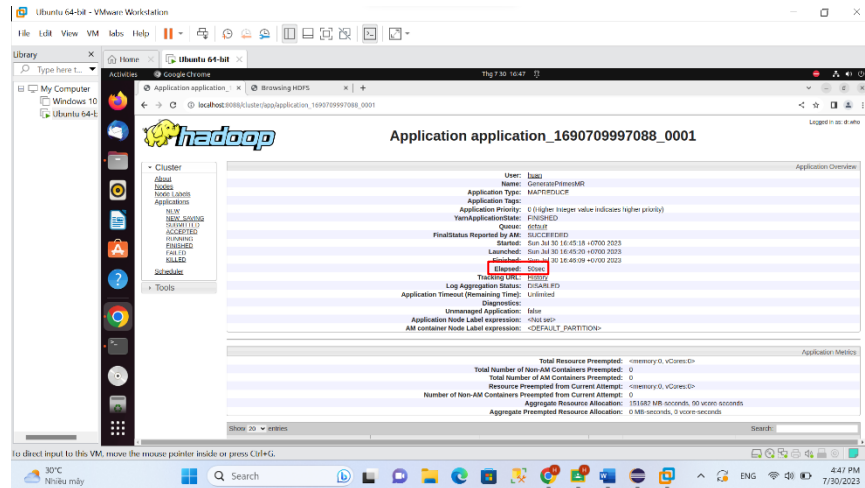


Hình 3.8. Hình ảnh thời gian thực thi chương trình

3.3. Thực nghiệm thuật toán song song trên máy ảo

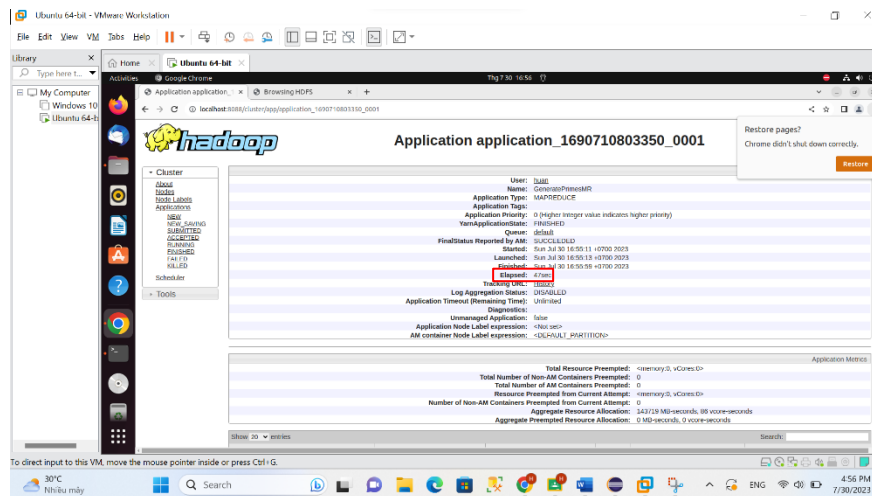
Làm tương tự như máy thật nhưng thay đổi số lượng processor lần lượt là 2, 4, 6, 8.

- Thực thi chương trình trên máy ảo với số lượng processors = 2 (50 sec).



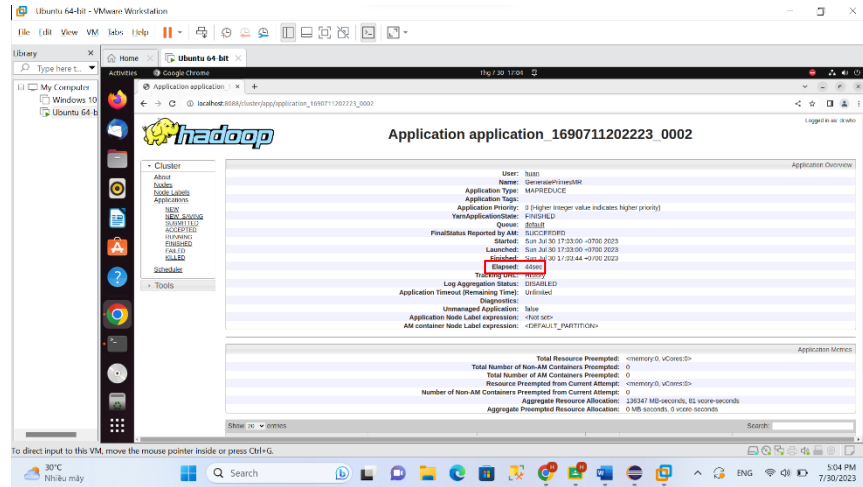
Hình 3.9. Hình ảnh thời gian thực thi trên máy ảo với 2 processors

- Thực thi chương trình trên máy ảo với số lượng processors = 4 (47 sec).



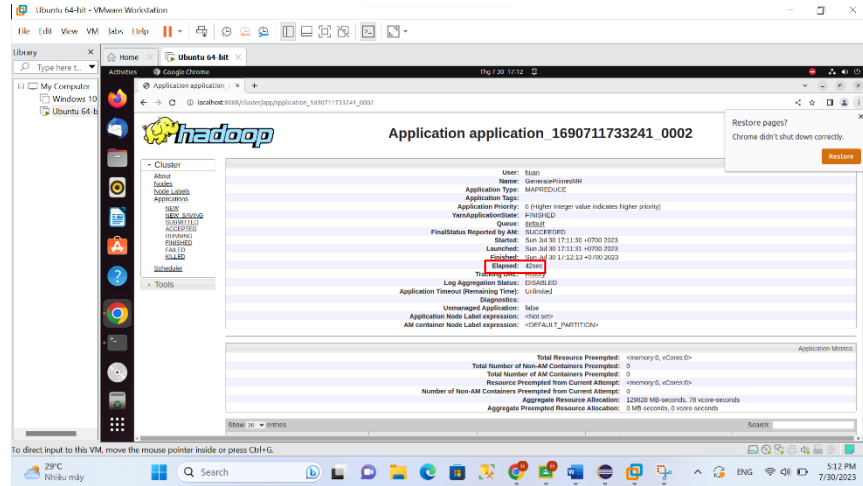
Hình 3.10. Hình ảnh thời gian thực thi trên máy ảo với 4 processors

- Thực thi chương trình trên máy ảo với số lượng processors = 6 (44 sec).



Hình 3.11. Hình ảnh thời gian thực thi trên máy ảo với 6 processors

- Thực thi chương trình trên máy ảo với số lượng processors = 8 (42 sec).



Hình 3.12. Hình ảnh thời gian thực thi trên máy ảo với 8 processors

3.4. Phân tích Speedup (tăng tốc) và Efficiency (độ hiệu quả)

Áp dụng công thức tính Speedup và Efficiency ở chương 2, ta có bảng kết quả:

Bảng 3.1. Bảng khảo sát Speedup và Efficiency

	T_S	T_P	S	E
2	363.24	50	7.2648	3.6324
4	363.24	47	7.7285	1.9321
6	363.24	44	8.2555	1.3759
8	363.24	42	8.6486	1.0810
12	363.24	31	11.7174	0.9765

Dựa vào bảng kết quả trên, ta thấy được thuật toán song song hiệu quả hơn thuật toán tuần tự và nếu số lượng processors càng nhiều thì thời gian thực hiện thuật toán song song Sàng Eratosthenes càng nhanh.