KMeans MapReduce cho phân đoạn hình ảnh

Ứng dụng vào tìm kiếm khối u não từ ảnh chụp cắt lớp MRI

Mục tiêu

- Cài đặt thuật toán KMeans và triển khai trên kiến trúc MapReduce của Hadoop
- Sử dụng các kĩ thuật xử lý hình ảnh để tìm vị trí khối u

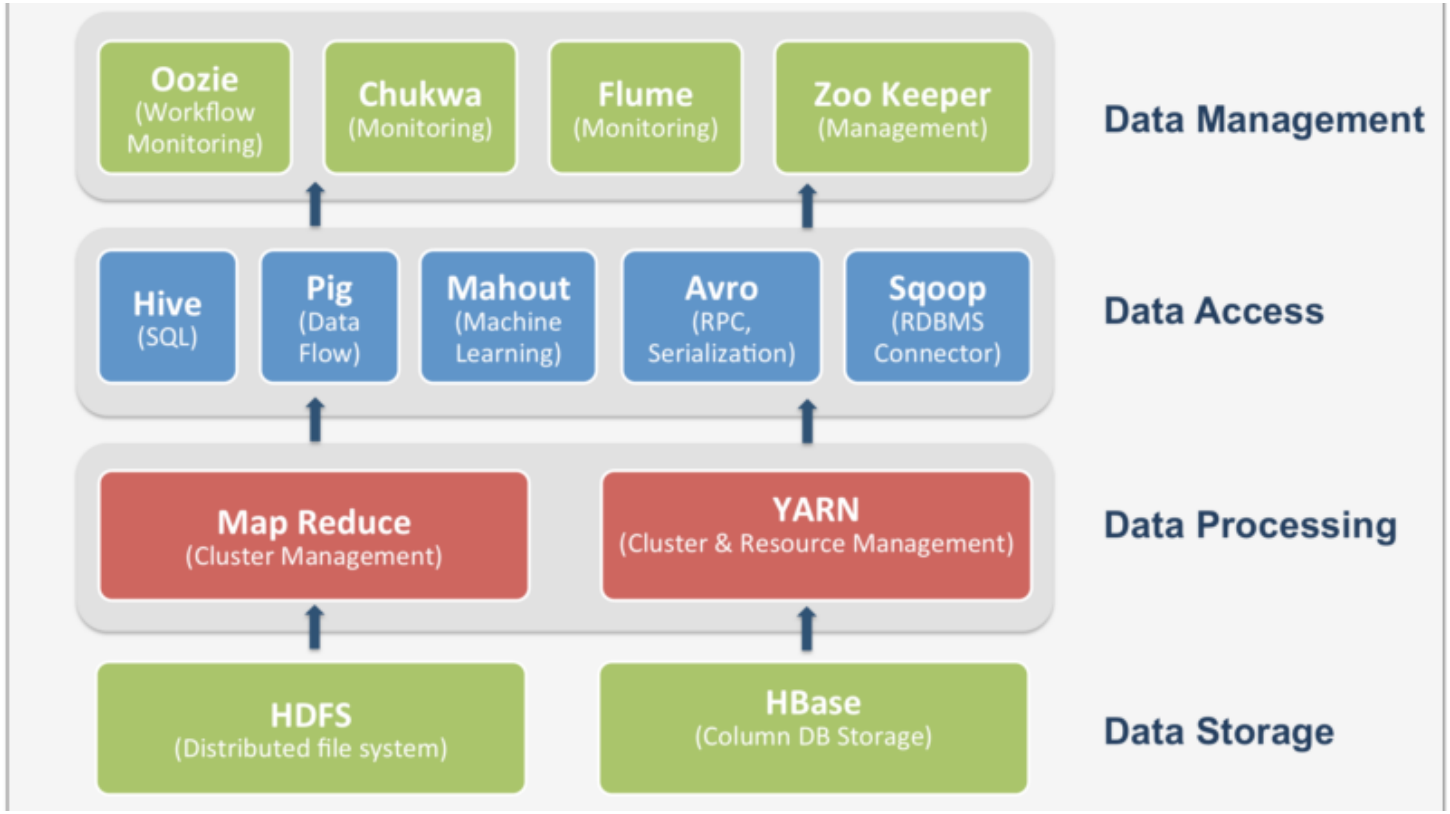
Mục lục

- 1. Tổng quan về Hadoop
- 2. Về kiến trúc MapReduce
- 3. Thuật toán KMeans và khởi tạo clusters bằng KMeans++
- 4. Triển khai trên MapReduce
- 5. Xử lý hình ảnh sau khi đã phân đoạn để định vị khối u
- 6. Đánh giá thuật toán.

1. Tổng quan về Hadoop

- **Theo Apache Hadoop** thì: Apache Hadoop là một framework dùng để chạy những ứng dụng trên 1 cluster lớn được xây dựng trên những phần cứng thông thường.

1. Tổng quan về Hadoop

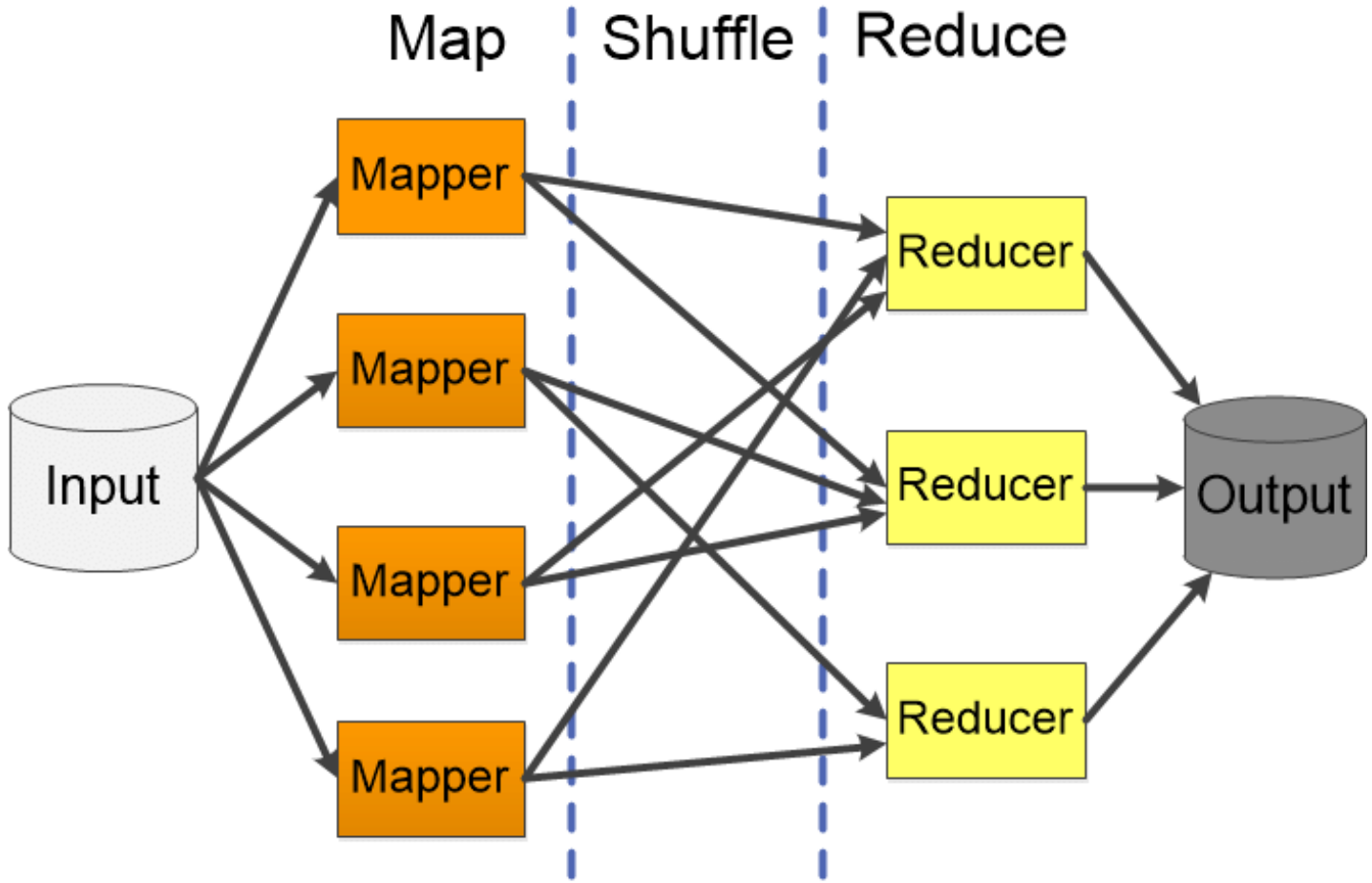


- Hệ sinh thái Hadoop gồm nhiều thành phần, nhưng trong chủ đề này chúng ta sẽ nói đến HDFS và Map Reduce

2. Về kiến trúc MapReduce

- **Theo Google**: MapReduce là mô hình dùng cho xử lý tính toán song song và phân tán trên hệ thống phân tán.

2. Về kiến trúc MapReduce



2. Về kiến trúc MapReduce

- **Hàm Map** nhận mảnh dữ liệu input, rút trích thông tin cần thiết các từng phần tử tạo kết quả trung gian
- **Hàm Reduce** tổng hợp kết quả trung gian, tính toán để cho kết quả cuối cùng.

3. Thuật toán KMeans và khởi tạo clusters với KMeans++

3.1 Thuật toán KMeans

- Cho một bộ dữ liệu gồm n điểm dữ liệu $(\overrightarrow{x_1}, \overrightarrow{x_2}, \ldots, \overrightarrow{x_n})$, với mỗi điểm là một vector có d chiều

- **Mục tiêu:** chia n điểm dữ liệu đã cho thành $k\ (k \leq n)$ tập con $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ nhằm giảm thiểu tổng bình phương bên trong cụm.

3.1 Thuật toán KMeans

- Mục tiêu chính sẽ phân cụm theo phương trình:

$$\underset{\mathcal{S}}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{\vec{x} \in \mathcal{S}_i} \|\vec{x} - \overrightarrow{\mu_i}\|^2$$

- Với $\overrightarrow{\mu_i} = \frac{1}{|S_i|} \sum_{\vec{x} \in S_i} \vec{x}$ là kì vọng (centroid) và $|S_i|$ là kích cỡ của S_i

3.2 Các bước hoạt động của thuật toán KMeans

3.2 Thuật toán

- Khởi tạo các centroids ngẫu nhiên

$$\mathcal{C}^{(0)} = \left\{ \vec{\mu}_1^{(0)}, \vec{\mu}_2^{(0)}, \dots, \vec{\mu}_{\mathrm{k}}^{(0)} \right\}$$

3.2 Thuật toán

- Gán các điểm vào các clusters
- Với mỗi điểm dữ liệu, tính khoảng cách của nó tới centroids và gán vào centroids gần nhất và tạo thành cụm:

$$\mathcal{S}_i^{(t)} = \left\{ \vec{x}_p : \left\| \vec{x}_p - \vec{\mu}_i^{(t)} \right\|^2 \leq \left\| \vec{x}_p - \vec{\mu}_j^{(t)} \right\|^2 \right\}, \forall j, 1 \leq j \leq k$$

3.2 Thuật toán

- Cập nhật centroids

$$\vec{\mu}_i^{(t+1)} = \frac{1}{\left|\mathcal{S}_i^{(t)}\right|} \sum_{\vec{x} \in \mathcal{S}_i^{(t)}} \vec{x}_j$$

- Lặp lại thuật toán cho đến khi chấp nhận được.

Hạn chế của KMeans

- Trong trường hợp xấu nhất, độ phức tạp trở thành superpolynomial
- Việc khởi tạo centroids ngẫu nhiên có thể khiến việc clustering trở nên khó khăn hơn và kết quả sẽ có sự sai khác sau mỗi lần chạy.

3.3 Cải thiện với thuật toán KMeans++

3.3 Thuật toán

- Thay vì khởi tạo k centroids một cách ngẫu nhiên, ta chỉ khởi tạo centroid đầu ngẫu nhiên
- Tìm $k - 1$ centroids còn lại bằng việc sử dụng xác suất

3.3 Thuật toán

- Với mỗi điểm dữ liệu, chúng ta tính khoảng cách từ $\vec{x}$ đến $\vec{\mu}$

$$\mathcal{D}_i(\vec{x}) = \min_{\vec{\mu} \in \{\vec{\mu}_1,\ldots,\vec{\mu}_i\}} \|\vec{x} - \vec{\mu}\|^2$$

- Tạo vector

$$\vec{\mathcal{D}}_{i+1} = [\mathcal{D}_1(\vec{x}_1) \quad \cdots \quad \mathcal{D}_1(\vec{x}_n)]$$

- Tính xác suất:

$$p(\vec{x}) = \frac{\mathcal{D}_i(\vec{x})}{\sum_{\vec{x'} \in X} \mathcal{D}_i\left(\vec{x'}\right)}, \qquad \forall\ \vec{x} \in X$$

3.3 Thuật toán

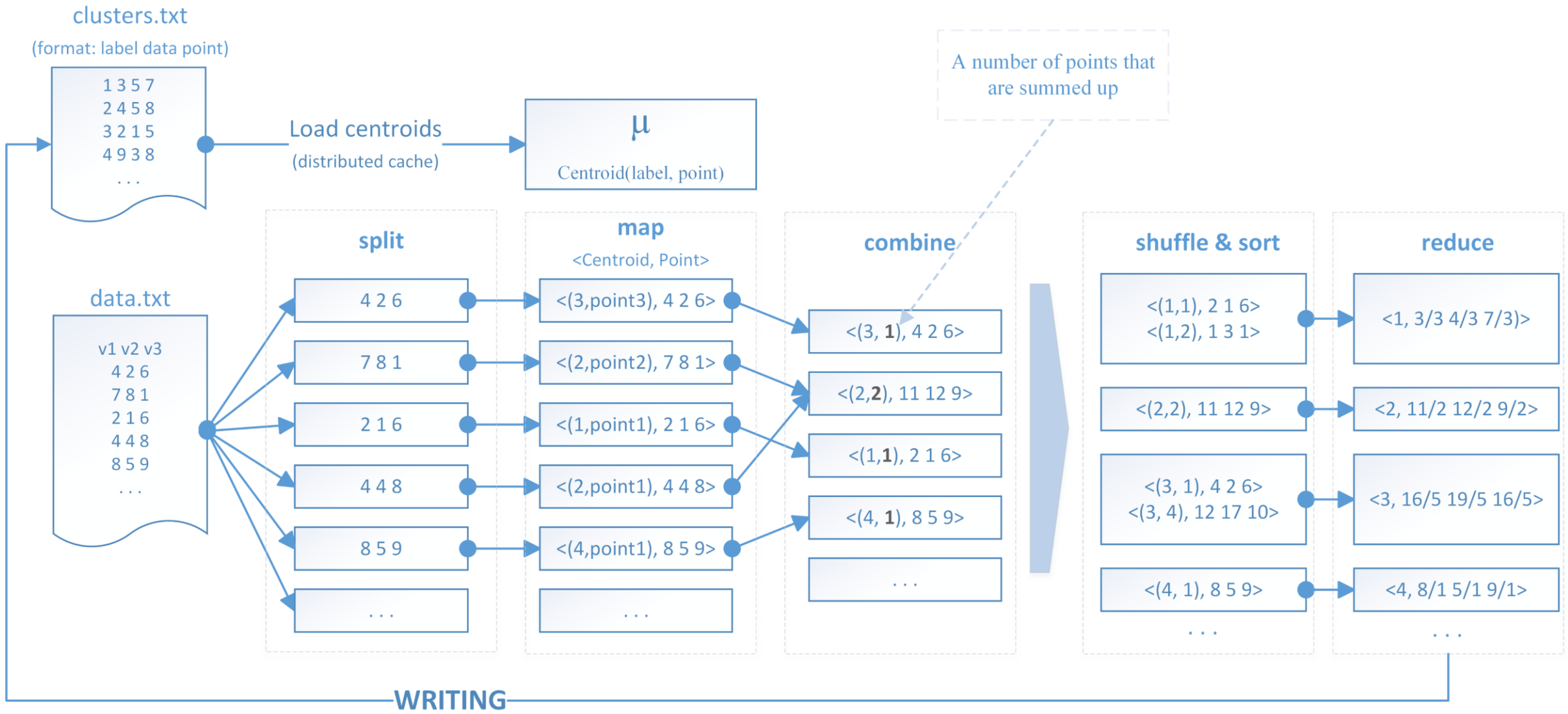
- Tính hàm phân phối tích lũy:

$$C(\vec{x}) = \sum_{i=1}^{j} \left(p(\vec{x}_i)\right), \qquad \vec{x}_j \leq \vec{x}$$

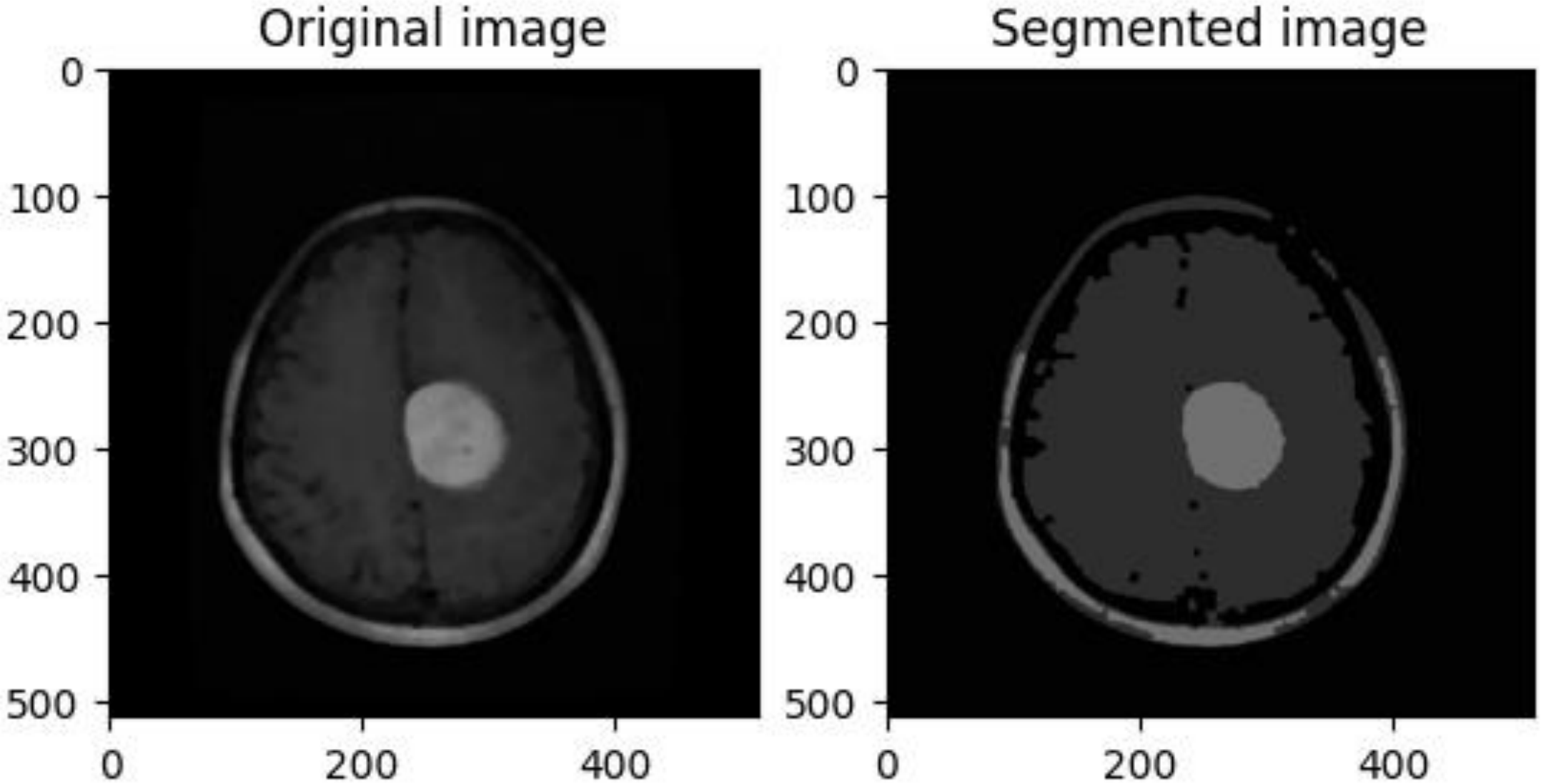
- Chọn ngẫu nhiên 1 điểm $r \in [0,1]$
- Dùng binary search để tìm khoảng $C(\vec{x})$ có chứa r
- Chọn $\vec{x}$ làm một centroid mới

4. Triển khai trên MapReduce

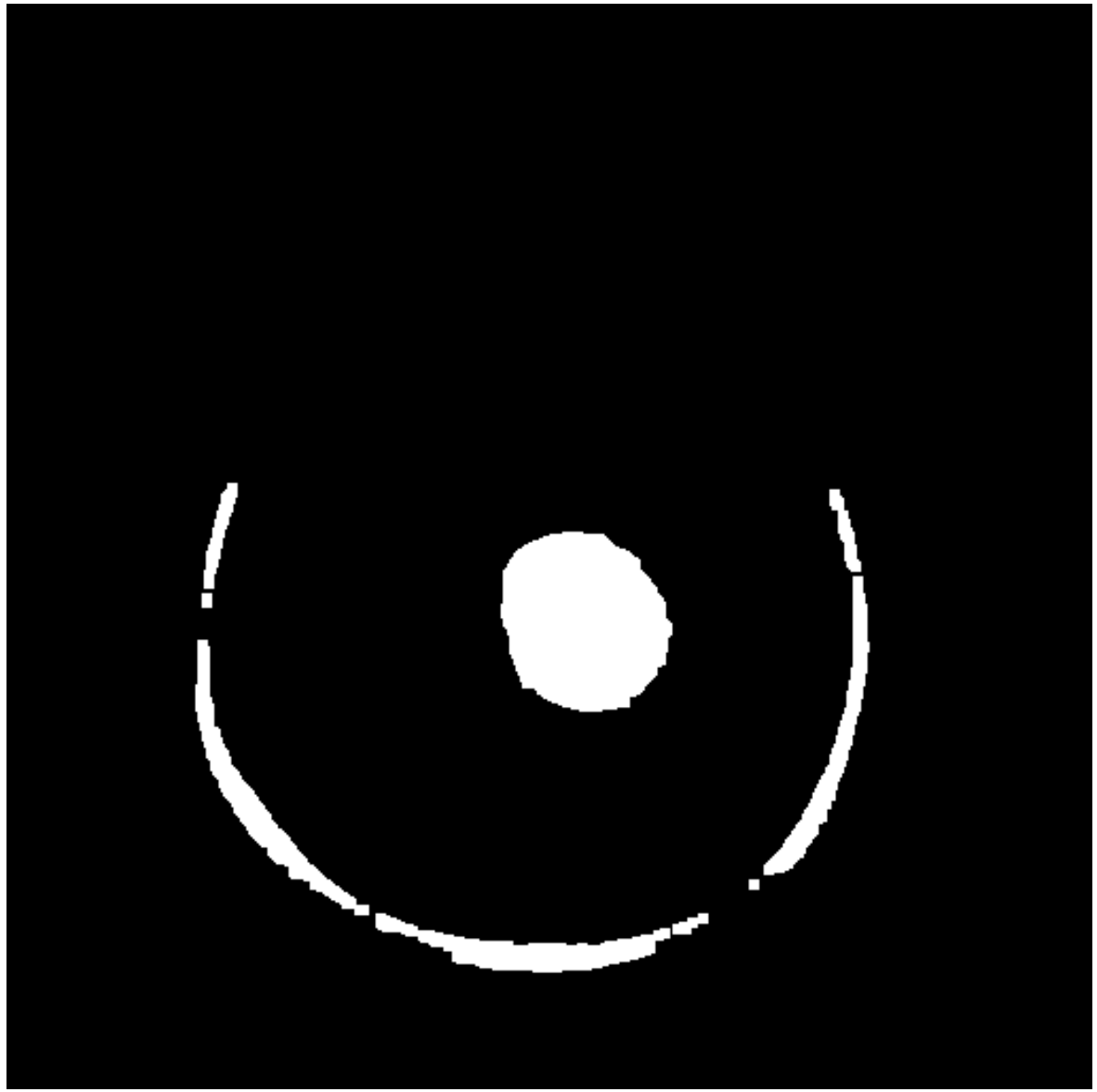
4. Triển khai trên MapReduce



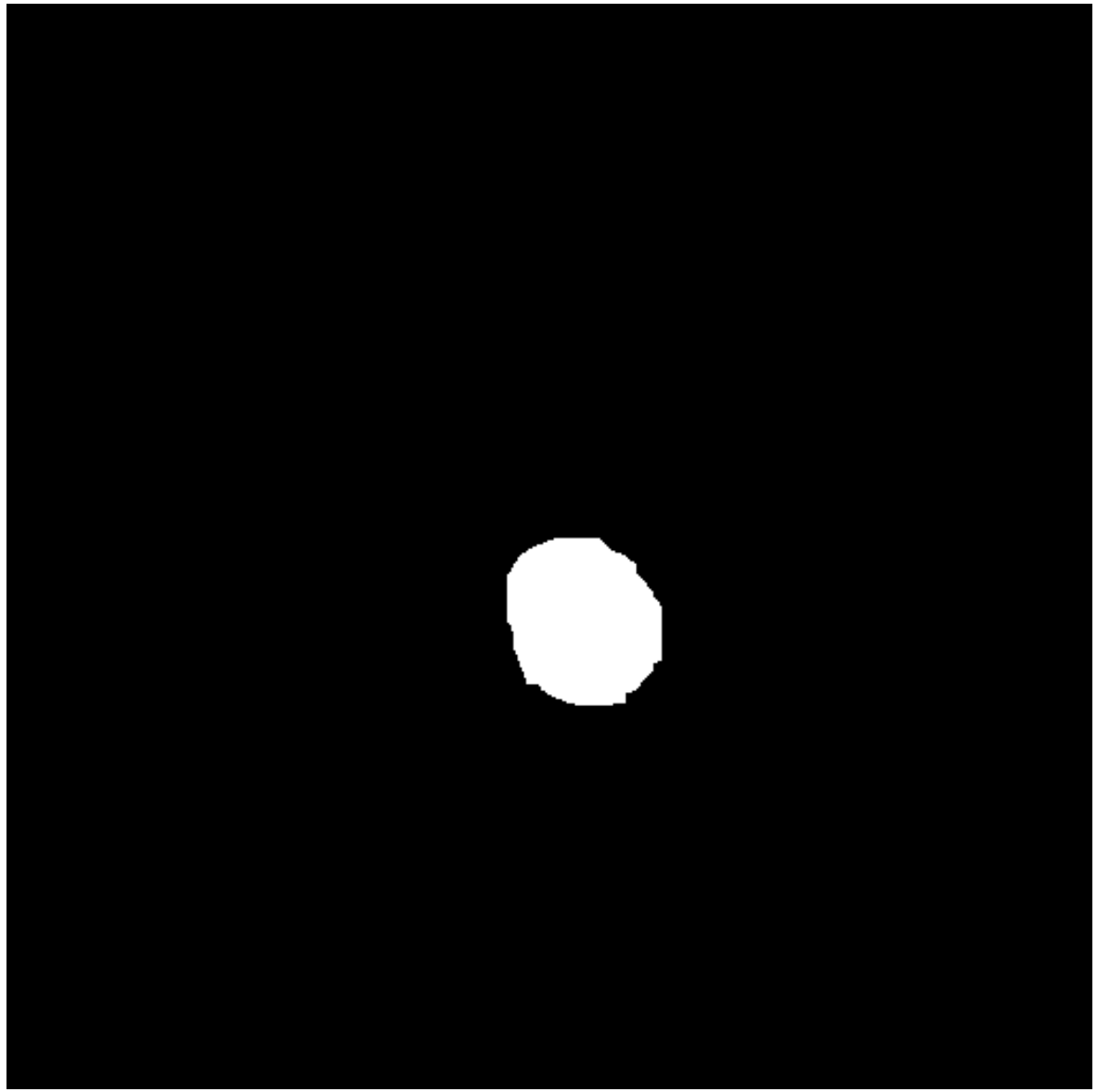
5. Xử lý hình ảnh sau khi đã phân cụm


Original image
Segmented image
0
100
200
300
400
500
0
200
400

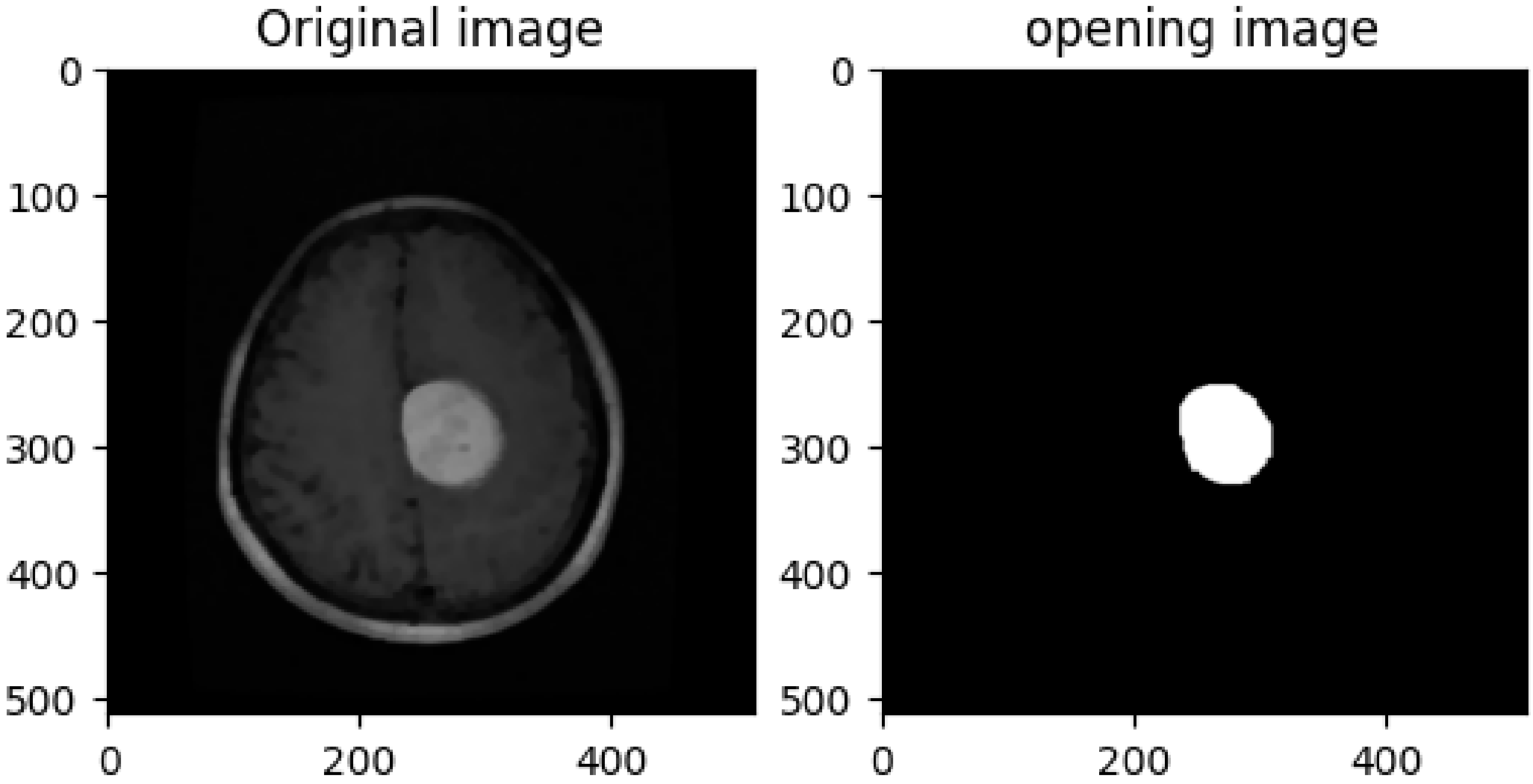
Nhị phân hóa hình ảnh



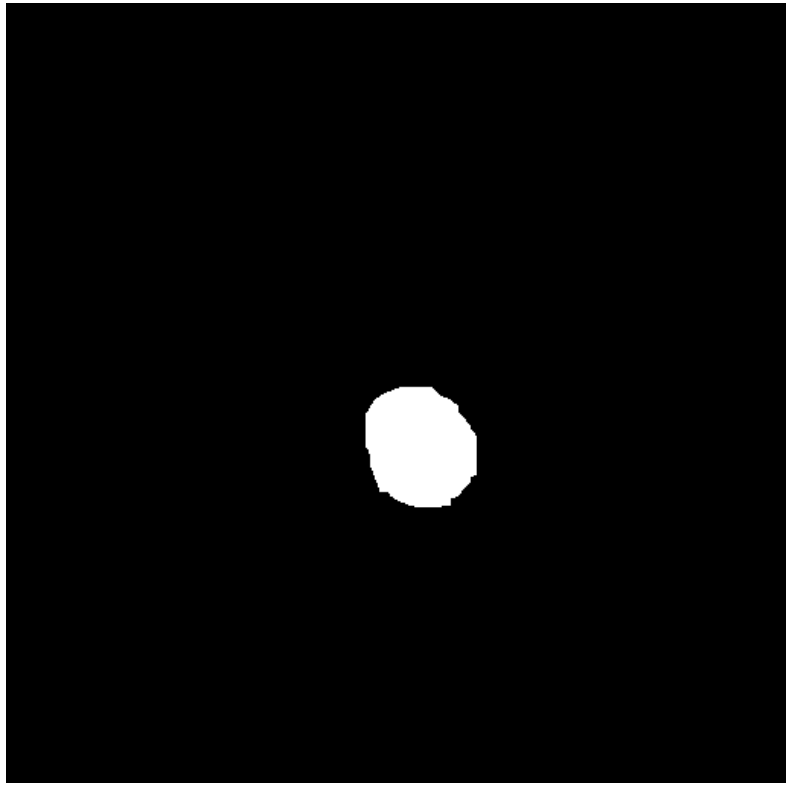
Erode và Dilate hình ảnh sau nhị phân hóa



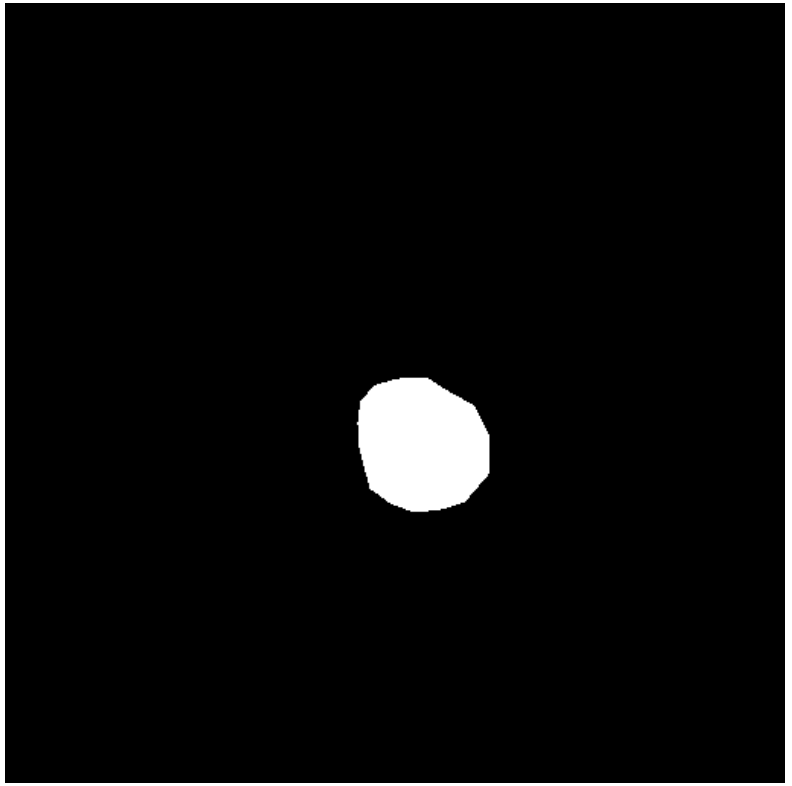
Kết quả:



So sánh kết quả thu được và thực tế



Thu được



Thực tế

6. Đánh giá thuật toán

6.1 Dice

$$\text{Dice} = \frac{2|A \cap B|}{|A| + |B|}$$

- Đánh giá độ giống nhau giữa hai tập hợp, =1 nếu hai tập trùng nhau, =0 nếu hai tập không hề trùng nhau.
- Chúng ta sẽ sử dụng hệ số này để đánh giá độ tương đồng (chính xác) giữa 2 hình ảnh thu được và kết quả thực tế.

6.2 Đánh giá

- Chương trình được chạy trên tổng cộng 59 hình ảnh để so sánh với các kết quả đề ra.
- Độ chính xác cao nhất là: 84.74%
- Độ chính xác trung bình là: 40.19%

6.2 Đánh giá

- Tuy độ chính xác rõ ràng kém hơn hẳn so với mô hình phức tạp như CNN, CapsNet, U-Net,… Nhưng có thể được dùng để đánh giá bộ dữ liệu.