MapReduce gồm 3 bước chính:

1. **Map (Ánh xạ)**
   * Chia nhỏ dữ liệu đầu vào thành các phần nhỏ.
   * Mỗi phần được xử lý song song bằng các hàm map(), tạo ra cặp **key-value**.
2. **Shuffle & Sort (Trao đổi và sắp xếp)**
   * Gom nhóm các cặp **key-value** có cùng **key** lại với nhau.
   * Sắp xếp dữ liệu theo **key** để chuẩn bị cho bước Reduce.
3. **Reduce (Giảm thiểu)**
   * Áp dụng hàm reduce() lên từng nhóm **key-value** để tổng hợp kết quả.
   * Xuất dữ liệu cuối cùng sau khi xử lý.

Ví dụ: Nếu muốn đếm số lần xuất hiện của mỗi từ trong một văn bản:

* **Map**: "hello world" → [(hello,1), (world,1)]
* **Shuffle & Sort**: [(hello, [1,1,1]), (world, [1,1])]
* **Reduce**: [(hello, 3), (world, 2)]

**Môi trường MapReduce đảm nhiệm:**

✔ **Phân chia dữ liệu đầu vào** → Chia nhỏ dữ liệu thành các phần (partitions) để xử lý song song.  
✔ **Lập lịch thực thi chương trình** → Phân phối các nhiệm vụ (tasks) cho nhiều máy.  
✔ **Nhóm dữ liệu theo key (group by key)** → Gom nhóm dữ liệu sau bước Map, nhưng đây thường là **nút thắt cổ chai** (bottleneck) do lượng dữ liệu lớn.  
✔ **Xử lý lỗi máy chủ** → Nếu một máy bị lỗi, MapReduce sẽ tự động điều chỉnh.  
✔ **Quản lý giao tiếp giữa các máy** → Đảm bảo dữ liệu được truyền tải đúng trong hệ thống phân tán.

**Xử lý lỗi trong MapReduce**

**1. Lỗi ở Map Worker (Máy chạy tác vụ Map bị lỗi)**

* Tất cả các Map tasks đã hoàn thành hoặc đang chạy trên máy bị lỗi **sẽ được đặt lại thành trạng thái chờ (idle)**.
* Các tác vụ Map này sẽ được **lên lịch lại (rescheduled) trên một máy khác**.
* Các Reduce workers sẽ được **thông báo** về sự thay đổi này để lấy dữ liệu từ Map mới.

**2. Lỗi ở Reduce Worker (Máy chạy tác vụ Reduce bị lỗi)**

* Chỉ các Reduce tasks **đang chạy** trên máy bị lỗi mới bị ảnh hưởng.
* Những tasks này sẽ **được đặt lại trạng thái idle và khởi động lại trên một máy khác**.
* Các Map tasks đã hoàn thành **không bị ảnh hưởng** vì đầu ra của Map đã được lưu trữ.