**Lương Tường Vy – 21133093**

**Task 1: Paper Reading**

**a) Khóa/giá trị đầu vào, khóa/giá trị trung gian và khóa/giá trị đầu ra liên quan như thế nào?**  
Các khóa/giá trị đầu vào, khóa/giá trị trung gian và khóa/giá trị đầu ra liên quan với nhau như sau trong quy trình xử lý dữ liệu:

- Khóa/giá trị đầu vào: Đây là dữ liệu/thông tin ban đầu được cung cấp làm nguồn cho quá trình xử lý.

***Ví dụ:*** Số nguyên được cung cấp đầu vào để tính giai thừa.

- Khóa/giá trị trung gian: Đây là kết quả tạm thời của các bước xử lý trung gian. Chúng phục vụ cho quá trình xử lý tiếp theo để đạt được kết quả cuối cùng.

***Ví dụ:*** Khi tính giai thừa, các giá trị trung gian là tích các số từ 1 đến số đang được tính toán.

- Khóa/giá trị đầu ra: Đây là kết quả/đáp án cuối cùng mà quá trình xử lý hướng đến.

***Ví dụ:*** Giai thừa của số đầu vào.

=> Ba loại khóa/giá trị này liên quan chặt chẽ theo trình tự: Giá trị đầu vào cung cấp nguồn xử lý, các giá trị trung gian phục vụ quá trình xử lý tiếp để cuối cùng đạt được giá trị đầu ra mong muốn.

**b) MapReduce xử lý các lỗi nút như thế nào?**

MapReduce có một số cơ chế tích hợp để xử lý và phục hồi từ lỗi nút:

* **Sao chép dữ liệu** - Mỗi khối dữ liệu đầu vào/trung gian được sao chép trên nhiều nút (thường là 3 bản sao). Điều này cho phép các tác vụ tiếp tục nếu một trong các nút bị lỗi.
* **Theo dõi tiến trình tác vụ** - Dịch vụ JobTracker theo dõi tiến trình các tác vụ (Map và Reduce) và báo cáo vòng đời. Nếu một tác vụ không heartbeart trong khoảng thời gian được cấu hình, nó được coi là lỗi.
* **Lập lại các tác vụ bị lỗi** - Khi phát hiện tác vụ lỗi, JobTracker lập lại tác vụ đó trên nút khác. Nó sử dụng sao chép để truy cập các khối dữ liệu cần thiết trên nút khác.
* **Phục hồi tự động tác vụ** - MapReduce có thể phục hồi từ lỗi nút và tiếp tục các tác vụ mà không cần khởi động lại toàn bộ công việc. Chỉ các tác vụ lỗi/quá thời hạn mới được thực thi lại.
* **Thực thi suy đoán** - JobTracker có thể lên lịch thực thi suy đoán đối với các tác vụ chậm để xử lý các tác vụ chậm. Nếu tác vụ suy đoán hoàn thành trước, kết quả của nó sẽ được ưu tiên so với tác vụ chậm hơn.

=> Tóm lại, thông qua sao chép dữ liệu, theo dõi tác vụ, lập lại tác vụ và thực thi suy đoán, MapReduce có thể phát hiện và phục hồi từ lỗi nút một cách mượt mà mà không cần khởi động lại toàn bộ công việc xử lý.

**c) Ý nghĩa và hàm ý của địa phương tính (locality) là gì? Nó được sử dụng như thế nào?**

Ý nghĩa và hàm ý của địa phương tính (locality) trong MapReduce:

Địa phương tính là khái niệm về việc đặt dữ liệu và tính toán càng gần với nhau càng tốt, trong một hệ thống phân tán dữ liệu và tính toán trên nhiều nút.

Ý hàm ẩn sau khái niệm này là giảm thiểu chi phí truyền dữ liệu giữa các nút, đặc biệt là chi phí truyền dữ liệu qua mạng. Bởi mạng là yếu tố hạn chế tốc độ xử lý chủ yếu trong mô hình phân tán.

MapReduce sử dụng địa phương tính bằng cách lên lịch các task Map và Reduce trên các nút có sẵn dữ liệu cần xử lý. Giảm việc phải di chuyển dữ liệu giữa các nút qua mạng.

Điều này góp phần tối ưu hoá hiệu suất tổng thể của công việc xử lý, bằng cách giảm thiểu chi phí truyền dữ liệu qua mạng.

**d) Vấn đề nào được giải quyết bằng cách đưa hàm kết hợp (Combiner function) vào mô hình MapReduce?**

Vấn đề được giải quyết bằng cách đưa hàm kết hợp (Combiner function) vào mô hình MapReduce là hạn chế băng thông mạng, cụ thể:

Trong MapReduce, kết quả của các task Map sẽ được gửi qua mạng tới các task Reduce. Điều này tạo ra lượng dữ liệu trung gian lớn trên mạng.

Bằng cách sử dụng hàm kết hợp, một số phép tổng hợp có thể được thực hiện ngay trên nút chạy task Map, trước khi dữ liệu được chuyển tới Reduce.

Qua đó giảm lượng dữ liệu trung gian phải truyền qua mạng, tiết kiệm băng thông và tăng tốc độ xử lý chung.

Do đó, vấn đề về hạn chế băng thông mạng trong truyền dữ liệu trung gian được giải quyết một phần nhờ sử dụng hàm kết hợp.