Using MapReduce for PageRank — How Google Ranked the Web

Context:

Big Data / Distributed Systems / Hadoop Ecosystem

Learning Objectives

By the end of this lecture, you should be able to:

1. Explain how Google's PageRank algorithm measures the importance of web pages.
2. Understand how PageRank can be implemented using the MapReduce framework.
3. Identify the role of the Mapper and Reducer functions in iterative graph computation.
4. Write pseudocode (or simple code) to perform one iteration of PageRank using MapReduce.
5. Interpret convergence and scaling behavior in large-scale web graphs.

Lecture Outline (2–3 hours)

Segment	Duration	Content / Activities
1. Introduction to PageRank (Concepts)	20 min	- What is PageRank? - How hyperlinks represent votes of importance. - Intuition: "A page is important if many important pages link to it."
2. The Mathematics of PageRank	25 min	- Representing the web as a graph. - Adjacency matrix, stochastic transition matrix. - The PageRank formula: ($PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$) - Damping factor (typically 0.85).
3. Why MapReduce for PageRank?	15 min	- The web is massive → billions of nodes (pages). - Iterative computation with distributed data. - Each page's rank depends on the ranks of its incoming links. - Each iteration can be done with parallel map and reduce operations.
4. Implementing PageRank with MapReduce	45 min	- Concept of one iteration: 1️⃣ Mapper distributes rank contributions. 2️⃣ Reducer aggregates incoming contributions. - Maintaining the link graph across iterations. - Example pseudocode walkthrough (below).

Segment	Duration	Content / Activities
5. Example Walkthrough	30 min	- Sample dataset with 4 pages (A, B, C, D). - Show first iteration step-by-step. - Demonstrate how ranks propagate. - Discuss damping factor application.
6. Optimizations and Convergence	15 min	- Iterations until convergence (e.g., change < 0.001). - Dangling nodes and rank sinks. - Use of combiner and partitioner for performance.
7. Lab / Discussion	30 min	- Students simulate one iteration manually. - Discussion: How would Spark handle this differently? - Ethical reflection: “Should all links count equally?”

PageRank Algorithm Recap

Given:

- A set of pages: P = {A, B, C, D, ...}
- Each page has outgoing links L(p).
- Rank(PR) represents the probability that a random surfer lands on the page.

Formula: [PR(p_i) = \frac{1 - d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}] Where:

- (d) = damping factor (e.g., 0.85)
- (N) = total number of pages
- (M(p_i)) = set of pages linking to (p_i)

PageRank Using MapReduce — Pseudocode

Mapper Function

Input: (page_id, [list_of_outlinks, current_rank]) Output: (outlink, contribution)

```python
def mapper(page, page_data):
    links, rank = page_data
    num_links = len(links)
    for link in links:
        emit(link, rank / num_links)
    # Keep graph structure for next iteration
    emit(page, links)
```

Reducer Function

Input: (page, values) Output: (page, [list_of_outlinks, new_rank])

```
def reducer(page, values):
    total_rank = 0
    links = []
    for v in values:
        if type(v) == list:
            links = v
        else:
            total_rank += v
    new_rank = 0.15 / N + 0.85 * total_rank
    emit(page, [links, new_rank])
```

Iteration: Repeat mapper–reducer passes until ranks converge.

Example Walkthrough

Input graph:

- A → B, C
- B → C
- C → A
- D → C, A

Initial ranks: 1/N = 0.25 each

1️⃣ Mapper distributes fractional rank shares. 2️⃣ Reducer sums incoming contributions. 3️⃣ Apply damping factor. 4️⃣ Repeat 10–20 iterations.

Results and Insights

- Pages with many inbound links from high-ranked pages receive higher PageRank.
- Converges to a stable rank distribution.
- MapReduce makes it scalable to billions of pages.

Discussion Questions

1. Why does PageRank need multiple iterations?
2. How does the damping factor affect convergence?
3. What challenges occur with dangling nodes?
4. How might modern systems (like Spark) improve PageRank computation?
5. Should social media “influence” ranking follow similar ideas?

Suggested References

- Dean, J. & Ghemawat, S. (2004). *MapReduce: Simplified Data Processing on Large Clusters*. Google Research.

- Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). *The PageRank Citation Ranking: Bringing Order to the Web*. Stanford Technical Report.
- White, T. (2015). *Hadoop: The Definitive Guide* (O'Reilly).