Practical Work 5: The Longest Path

Nguyen The Xuan

December 5, 2025

1 Introduction

The objective of this practical work is to solve the "Longest Path" problem using the MapReduce framework. Given a large set of file paths (simulating the output of `find /` across multiple machines), the system must identify the single path with the maximum character length.

2 Design Strategy

We implemented a custom MapReduce solution in Python. To optimize performance and reduce communication overhead, we employed a "Combiner-like" strategy within the Mapper.

2.1 Mapper Logic (Local Maximum)

Instead of emitting every single path to the Reducer (which would cause a massive I/O bottleneck), each Mapper processes a chunk of paths and calculates the **local maximum** for that chunk.

- **Input:** A list of strings (paths).
- **Process:** Iterate through lines, track the longest string seen so far in this chunk.
- **Output:** A single pair `('global_max', (length, path))`.

2.2 Reducer Logic (Global Maximum)

- **Input:** A list of local maximums from all Mappers.
- **Process:** Compare the local maximums to find the absolute longest path.
- **Output:** The final longest path string.

3 Implementation Code

3.1 Mapper Function

```python
def mapper(text_chunk):
    lines = text_chunk.strip().split('\n')
    local_max_path = ""
    local_max_len = -1

    for line in lines:
        path = line.strip()
        if len(path) > local_max_len:
            local_max_len = len(path)
            local_max_path = path

    # Key is constant to route all local maxes to one reducer
    return [('global_max', (local_max_len, local_max_path))]
```

Listing 1: Finding Local Max in Mapper

3.2 Reducer Function

```
def reducer(item):
    key, values = item
    # values = list of (length, path) tuples
    overall_max = max(values, key=lambda x: x[0])
    return (key, overall_max)
```

Listing 2: Finding Global Max in Reducer