# Exercise 1

**3436574 - Marcel Hasenbalg**

# Task 1

1. What information does the task description contain that the master gives to a parser?

The master tells the parser in the task description the split (subset of documents) he has to parse.

1. What information does the parser report back to the master upon completion of the task?

The parser tells the master that he is idle and ready to parse another split.

1. What information does the task description contain that the master gives to an inverter?

The master assigns a term partition of a parser to an inverter for which he has to collect (term, docId) pairs, e.g. parser 3, term-partition a-f.

1. What information does the inverter report back to the master upon completion of the task?

The inverter tells the master that he is idle and can be assigned to a parser/term partition again.

1. How would you specify the number of parsers? Can you estimate this somehow?

It depends on how much time the index creation should take, how much memory is available on the worker nodes and the average length of a document.

If a low runtime is desirable, we should use a high number of available worker nodes for parsing, however we want to keep a balance such that we don’t need to merge and sort postings list too often.

So depending on the collection size and available memory on parsers, 75% of memory might be a good utilization. So for a collection with 100GB size and worker nodes with 16GB of RAM we could assign 12GB splits to the parsers 🡪 100GB/12GB = 9 parser nodes.

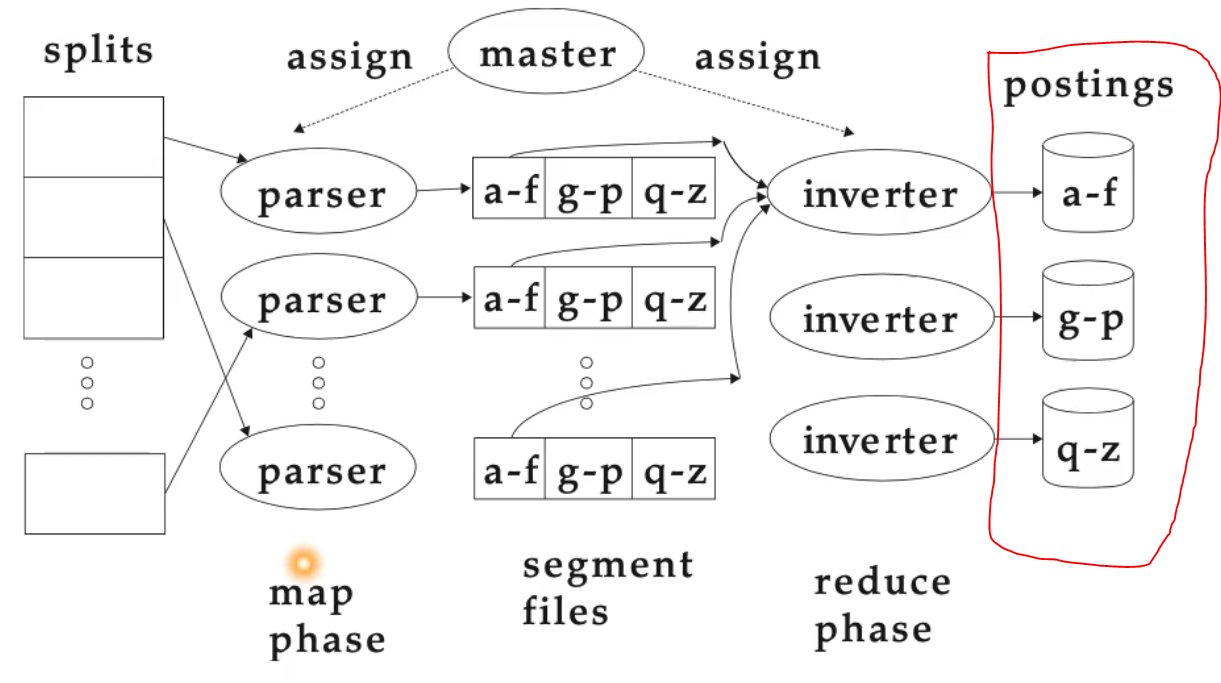
1. How would you specify the number of partitions from which the inverters put the index together?

If only one partition is used this might exceed the inverter nodes available memory.

If one partition per term is used, the inverters have to perform way too many merge and sort operations.

Again the number of partitions should be chosen in such a way, that the largest partition can still be handled by the available memory of the inverter.

# Task 2



To update the index we can load the already existing index into the partitioned postings of the distributed index algorithm (encircled in red).

Next we start indexing the new batch of documents, which will be parsed into the segment files and then merged and sorted with the old index.

When the process has finished we the new index which includes the new documents. The old index can be replaced by the new one.

We save all the time required to parse and invert the already indexed documents and only process the new ones.

However, the longer the postings list get, the longer the sorting process will take.

# Task 3

## Subtask 3.1

10 000 = k \* 1 000 000 ^b

3000 = k \* 100 000 ^b