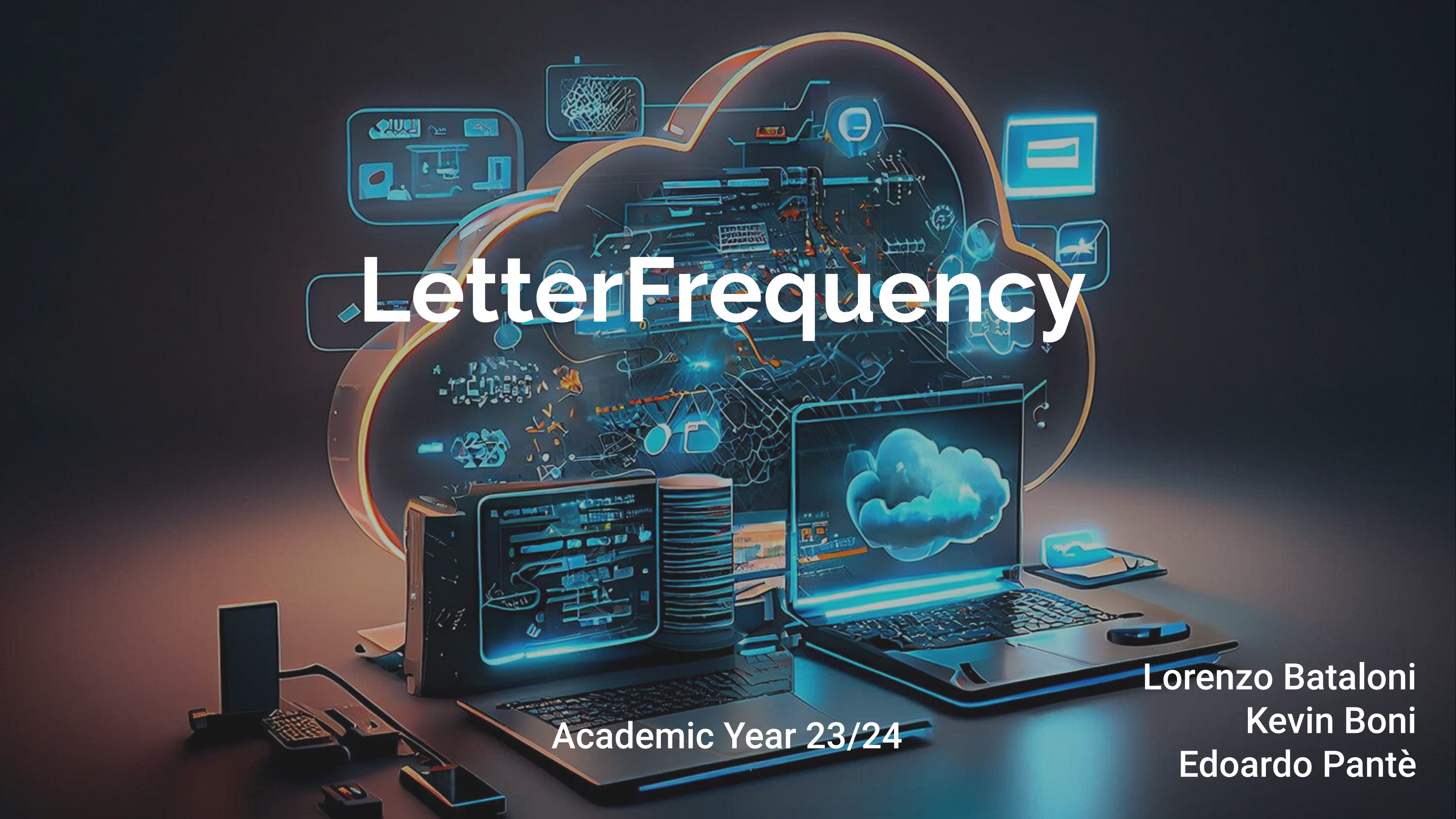


LetterFrequency

Academic Year 23/24

Lorenzo Bataloni
Kevin Boni
Edoardo Pantè

Timeline

Machines

- Configuration
- Script

Implementation

- Naive
- OptCombiner
- Opt

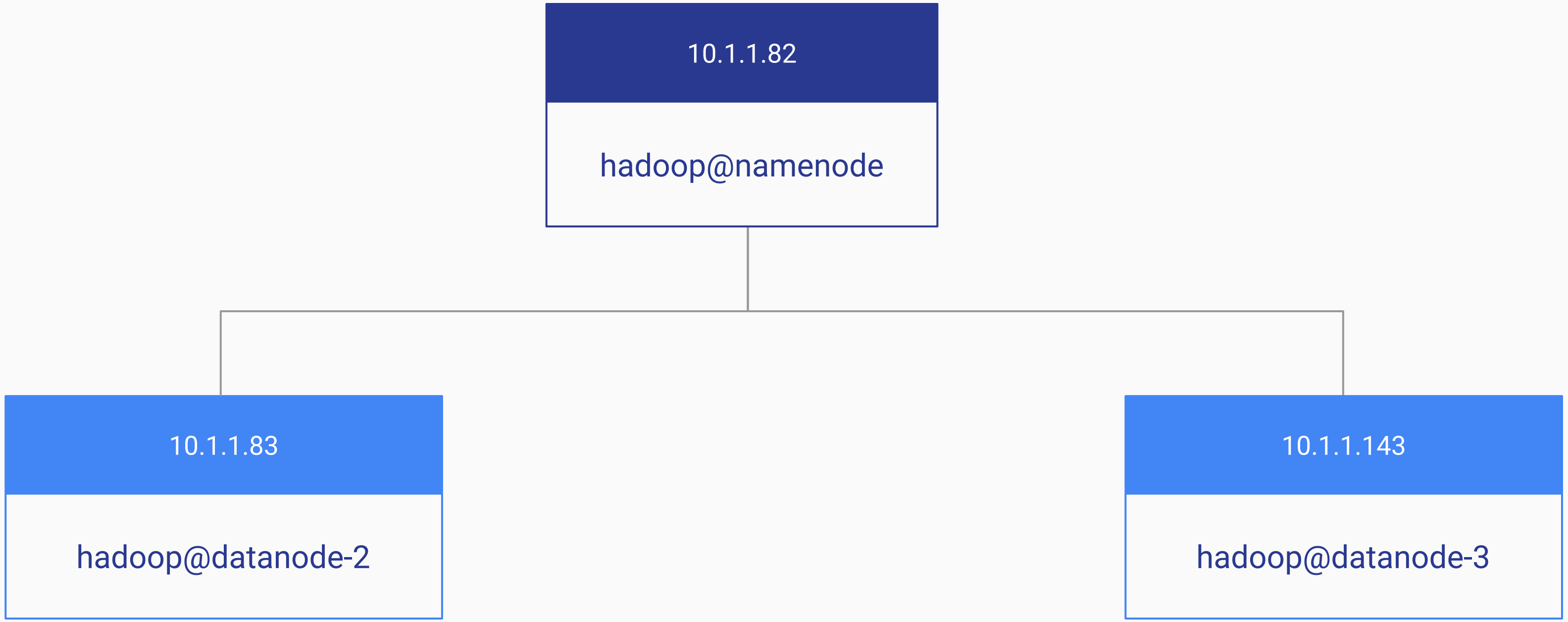
Results

- Frequency Analysis
- Execution Time
- Memory Occupation

Machines

- Architecture
- Script
- Files

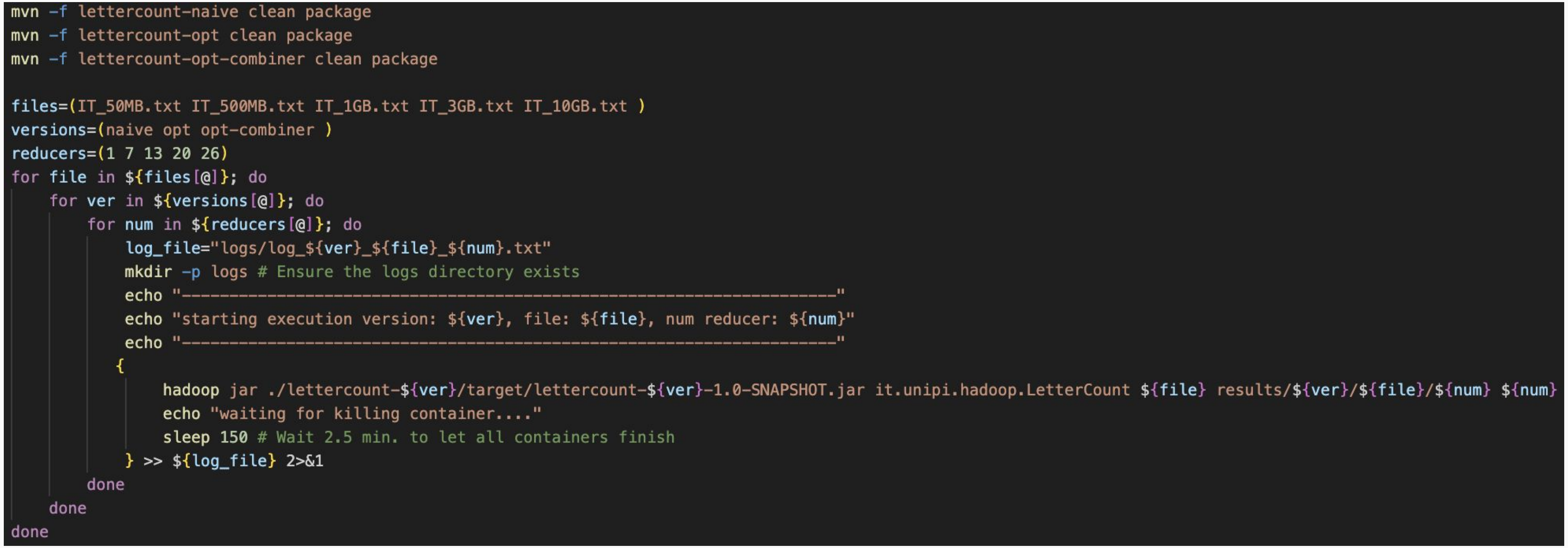
Architecture



Script

Bash Script

The following bash script was used to automate the execution of the implementations



```bash
mvn -f lettercount-naive clean package
mvn -f lettercount-opt clean package
mvn -f lettercount-opt-combiner clean package

files=(IT_50MB.txt IT_500MB.txt IT_1GB.txt IT_3GB.txt IT_10GB.txt )
versions=(naive opt opt-combiner )
reducers=(1 7 13 20 26)
for file in ${files[@]}; do
    for ver in ${versions[@]}; do
        for num in ${reducers[@]}; do
            log_file="logs/log_${ver}_${file}_${num}.txt"
            mkdir -p logs # Ensure the logs directory exists
            echo "---------------------------------------------------------------------"
            echo "starting execution version: ${ver}, file: ${file}, num reducer: ${num}"
            echo "---------------------------------------------------------------------"
           {
                hadoop jar ./lettercount-${ver}/target/lettercount-${ver}-1.0-SNAPSHOT.jar it.unipi.hadoop.LetterCount ${file} results/${ver}/${file}/${num} ${num}
                echo "waiting for killing container...."
                sleep 150 # Wait 2.5 min. to let all containers finish
            } >> ${log_file} 2>&1
        done
    done
done
```

Files

Sizes

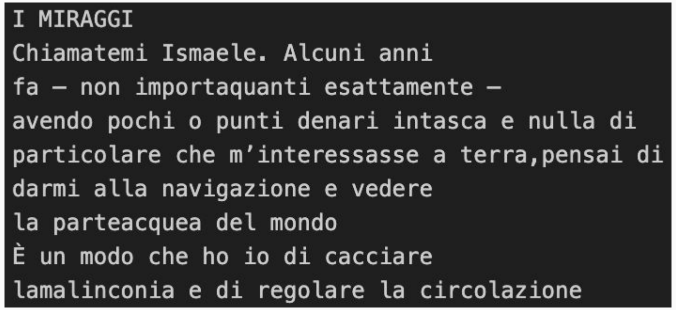
- 50 MB
- 500 MB
- 1 GB
- 3 GB
- 10 GB

Block Size

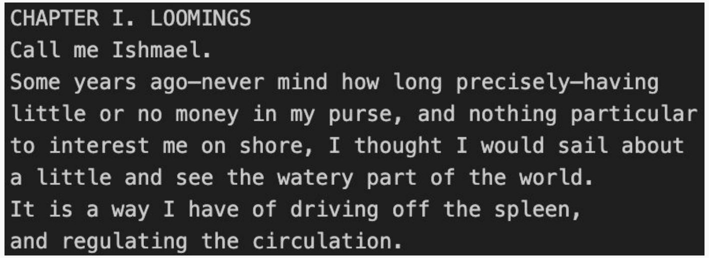
- 128 MB

Languages

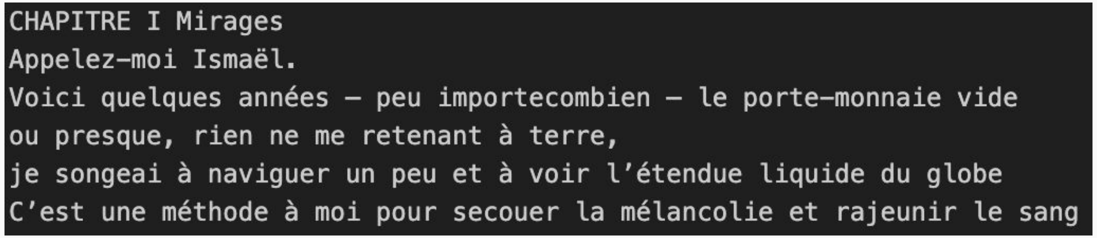
- Italian
- English
- French



```
I MIRAGGI
Chiamatemi Ismaele. Alcuni anni
fa – non importaquanti esattamente –
avendo pochi o punti denari intasca e nulla di
particolare che m’interessasse a terra,pensai di
darmi alla navigazione e vedere
la parteacquea del mondo
È un modo che ho io di cacciare
lamalinconia e di regolare la circolazione
```



```
CHAPTER I. LOOMINGS
Call me Ishmael.
Some years ago—never mind how long precisely—having
little or no money in my purse, and nothing particular
to interest me on shore, I thought I would sail about
a little and see the watery part of the world.
It is a way I have of driving off the spleen,
and regulating the circulation.
```

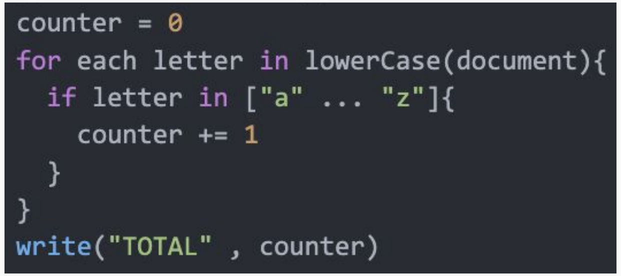


```
CHAPITRE I Mirages
Appelez-moi Ismaël.
Voici quelques années – peu importecombien – le porte-monnaie vide
ou presque, rien ne me retenant à terre,
je songeai à naviguer un peu et à voir l’étendue liquide du globe
C’est une méthode à moi pour secouer la mélancolie et rajeunir le sang
```

Implementation

- Naive
- OptCombiner
- Opt

Naive - Count Pipeline



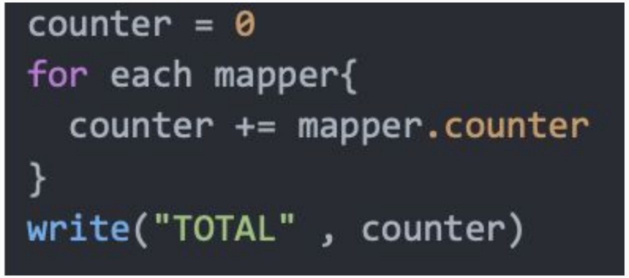
```
counter = 0
for each letter in lowerCase(document){
  if letter in ["a" ... "z"]{
    counter += 1
  }
}
write("TOTAL" , counter)
```

Mapper

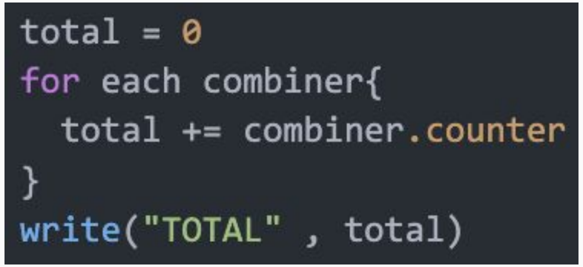
Count the total number of letter in the document.

Combiner

Pre-aggregate data for the reducer summing the data received from the mappers.



```
counter = 0
for each mapper{
  counter += mapper.counter
}
write("TOTAL" , counter)
```



```
total = 0
for each combiner{
  total += combiner.counter
}
write("TOTAL" , total)
```

Reducer

Calculate the total number of letters.

Naive - Frequency Pipeline

```
occurrency = {}
for each letter in lowerCase(document){
  if letter in ["a" ... "z"]{
    if letter not in occurrency{
      occurrency[letter] = 1
    }else{
      occurrency[letter] =+ 1
    }
  }
}
for each key in occurrency{
  write(key , occurrency[key])
}
```

Mapper

For each letter encountered in the document, create a map containing the number of occurrences.

Combiner

For each letter Received from the mapper, sum the occurrences to get the total for each letter.

```
for each key in keys{
  sum = 0
  for each mapper {
    sum += mapper.occurrency[key]
  }
  write(key , sum)
}
```

```
total_letters = read("TOTAL")
for each key in keys{
  sum = 0
  for each value in combiner[key].values{
    sum += value
  }
  freq = sum / total_letters
  write(key , freq)
}
```

Reducer

For each letter, calculate the frequency based on the total number of letter and the occurrences of each of them.

OptCombiner

```
counter = 0
for each letter in lowerCase(document){
  if letter in ["a" ... "z"]{
    counter += 1
  }
  write(letter , 1)
}
write(TOTAL_LETTERS , counter)
```

Mapper

Count the total number of letter in the document and for each letter encountered write it in the context.

Combiner

Sum the partial total received from the mapper, both for the number of characters and for the single character.

```
for each mapper{
  sum = 0
  for each key in mapper.keys{
    sum += read(mapper[key])
  }
  write(key , sum)
}
```

```
total_letters = read(TOTAL_LETTERS)
for each combiner{
  sum = 0
  for each key in combiner.keys{
    sum += read(mapper[key])
  }
  freq = sum / total_letters
  write(key , freq)
}
```

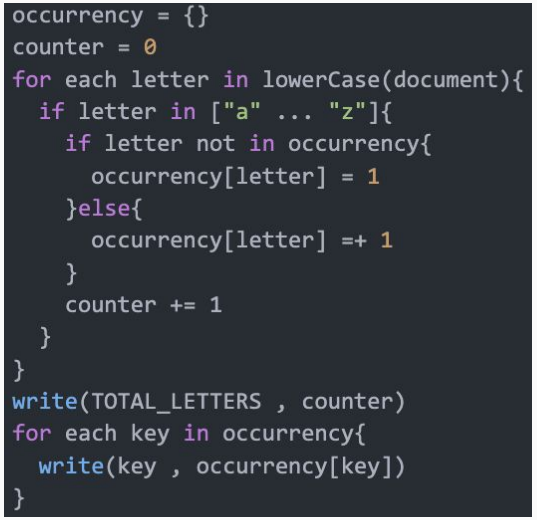
Reducer

Read the total from the context and sum the data received from the combiner to obtain the frequency of each letter.

Opt

Mapper

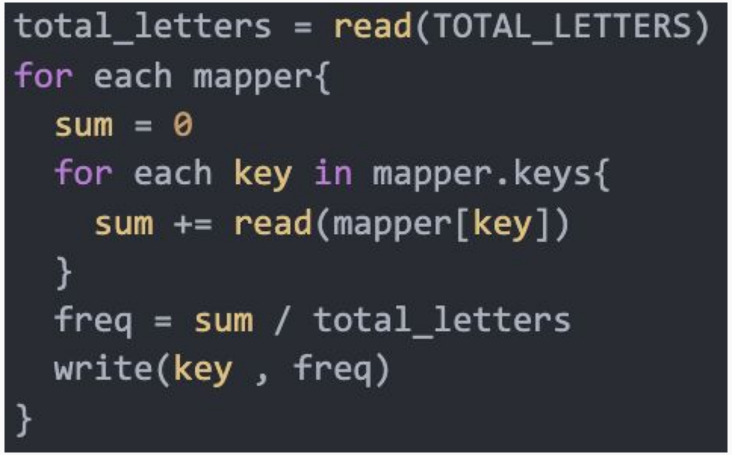
Writes in context for each letter the number of occurrences and increments TOTAL_LETTERS by the number of letters present



```
occurrency = {}
counter = 0
for each letter in lowerCase(document){
  if letter in ["a" ... "z"]{
    if letter not in occurrency{
      occurrency[letter] = 1
    }else{
      occurrency[letter] =+ 1
    }
    counter += 1
  }
}
write(TOTAL_LETTERS , counter)
for each key in occurrency{
  write(key , occurrency[key])
}
```

Reducer

Read the total from the context and sum the data received from the combiner to obtain the frequency of each letter.

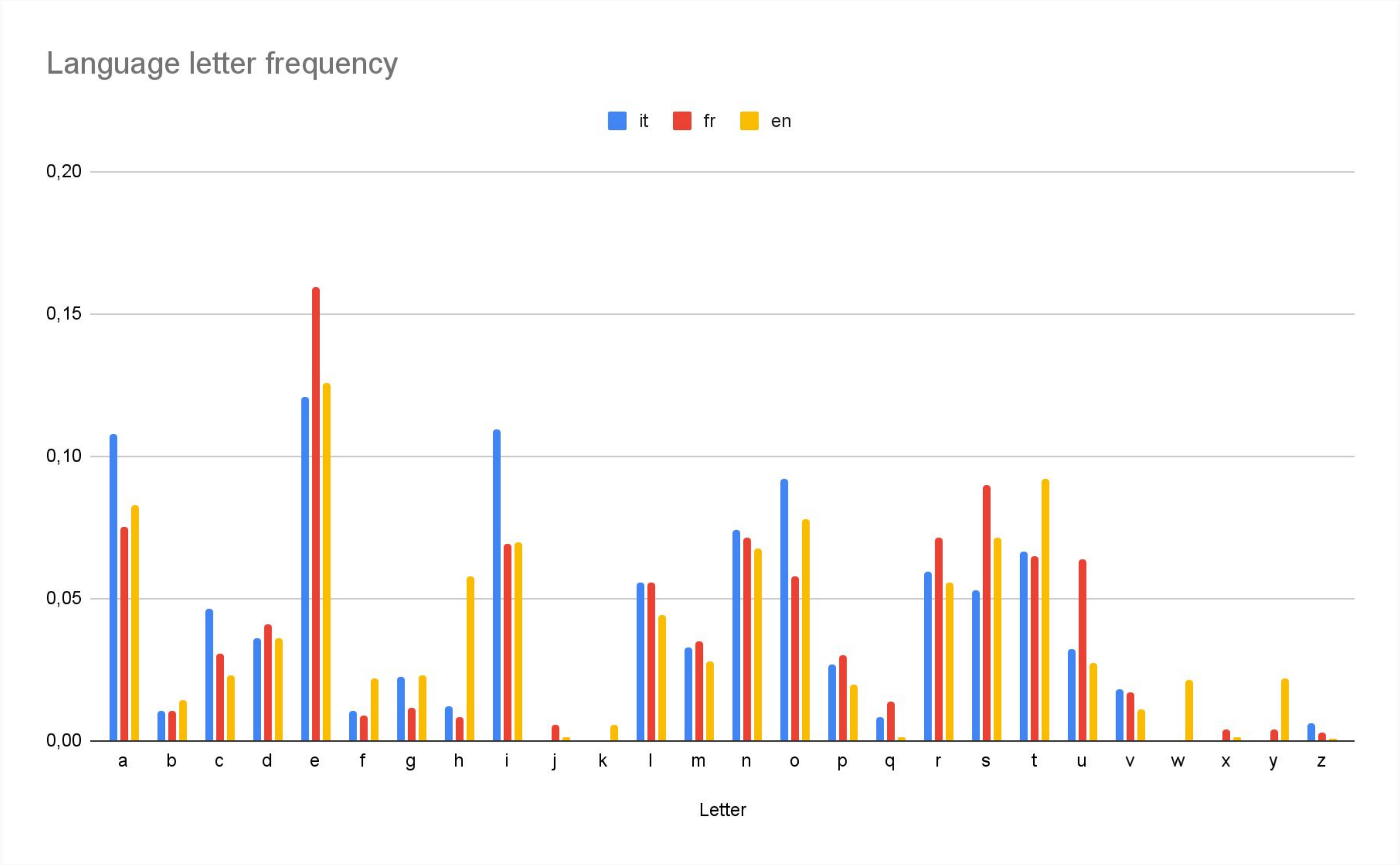


```
total_letters = read(TOTAL_LETTERS)
for each mapper{
  sum = 0
  for each key in mapper.keys{
    sum += read(mapper[key])
  }
  freq = sum / total_letters
  write(key , freq)
}
```

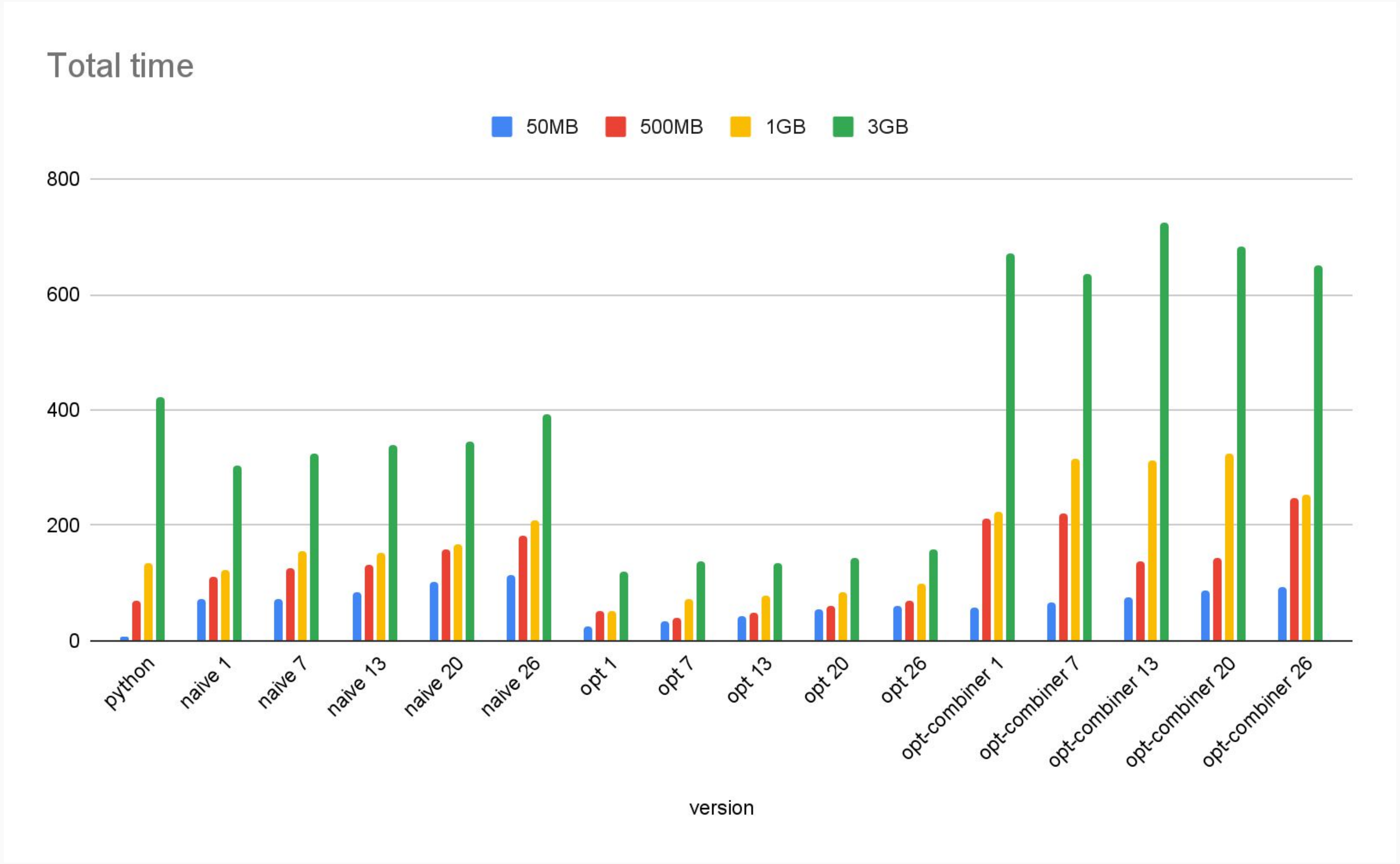
Results

- Frequency Analysis
- Execution Time
- Memory Consumptions

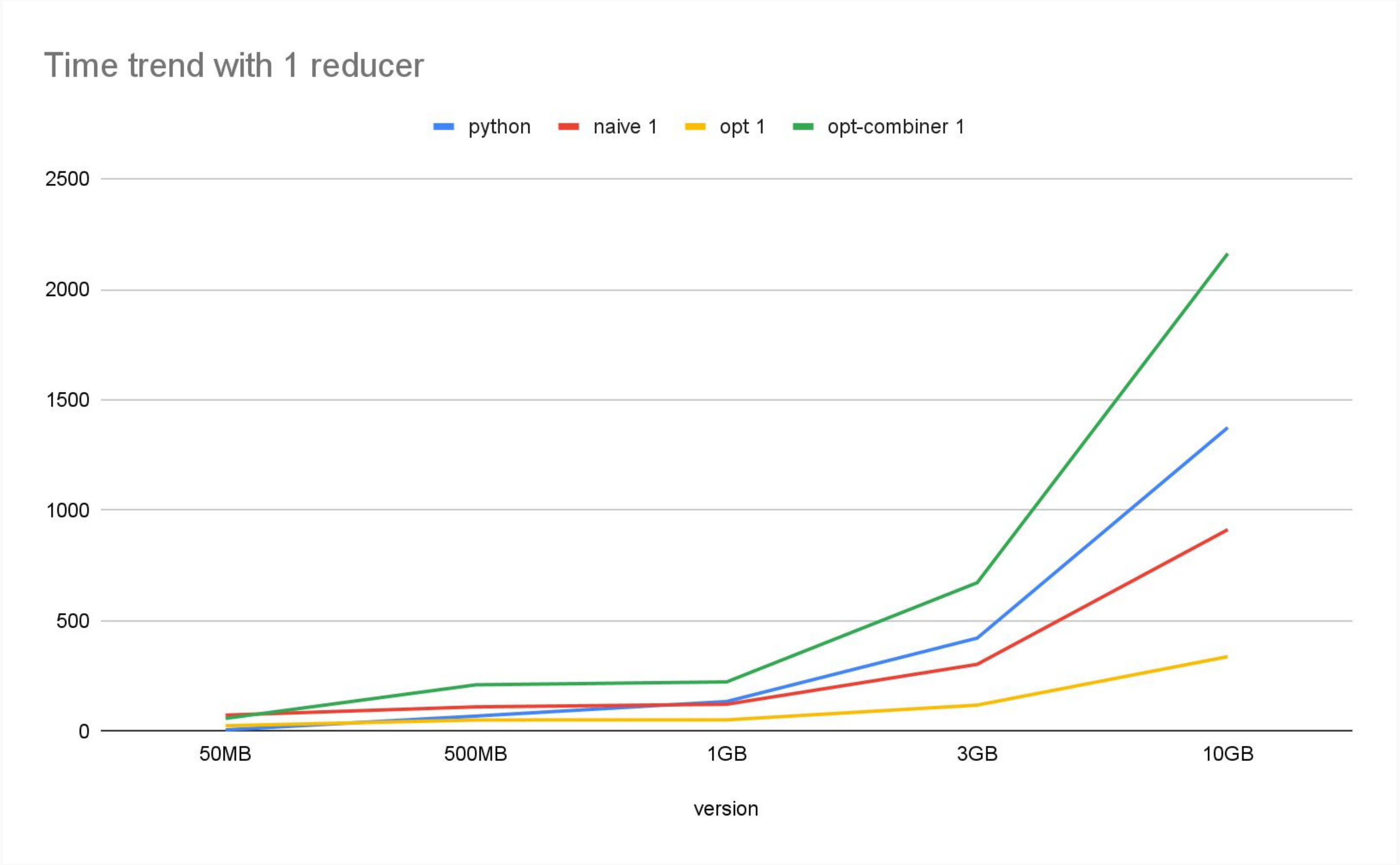
Frequency Analysis



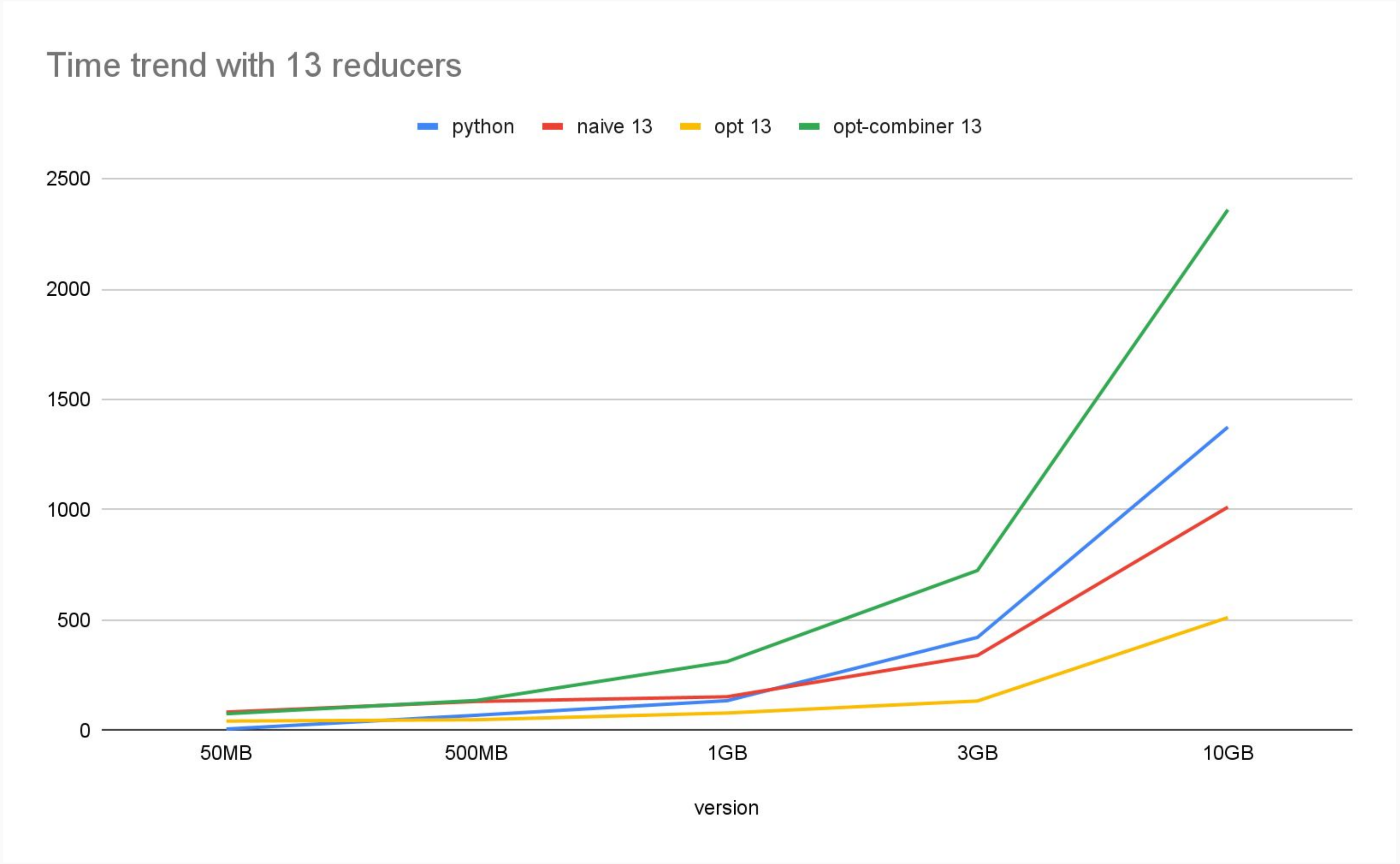
Execution Time



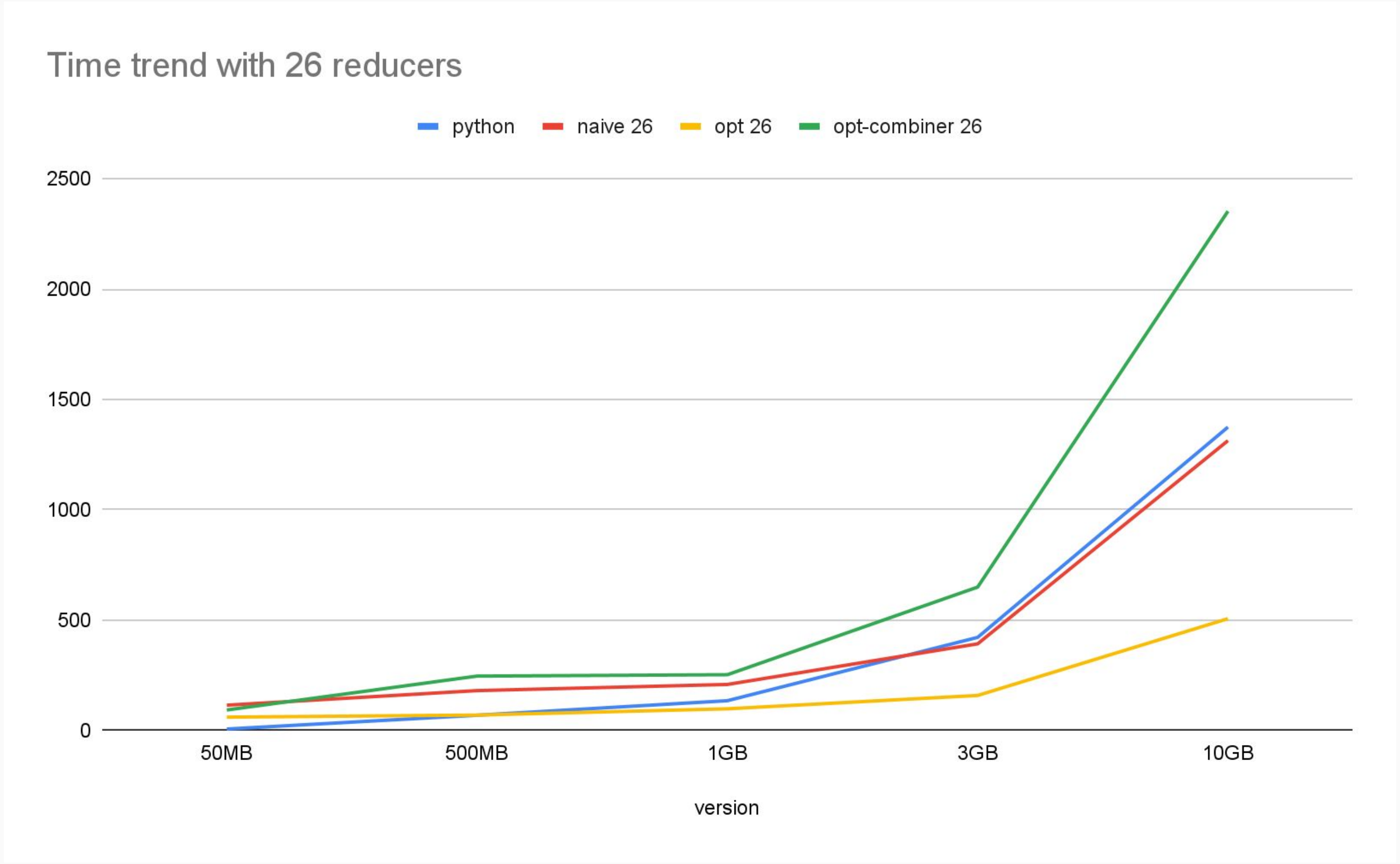
Execution Time



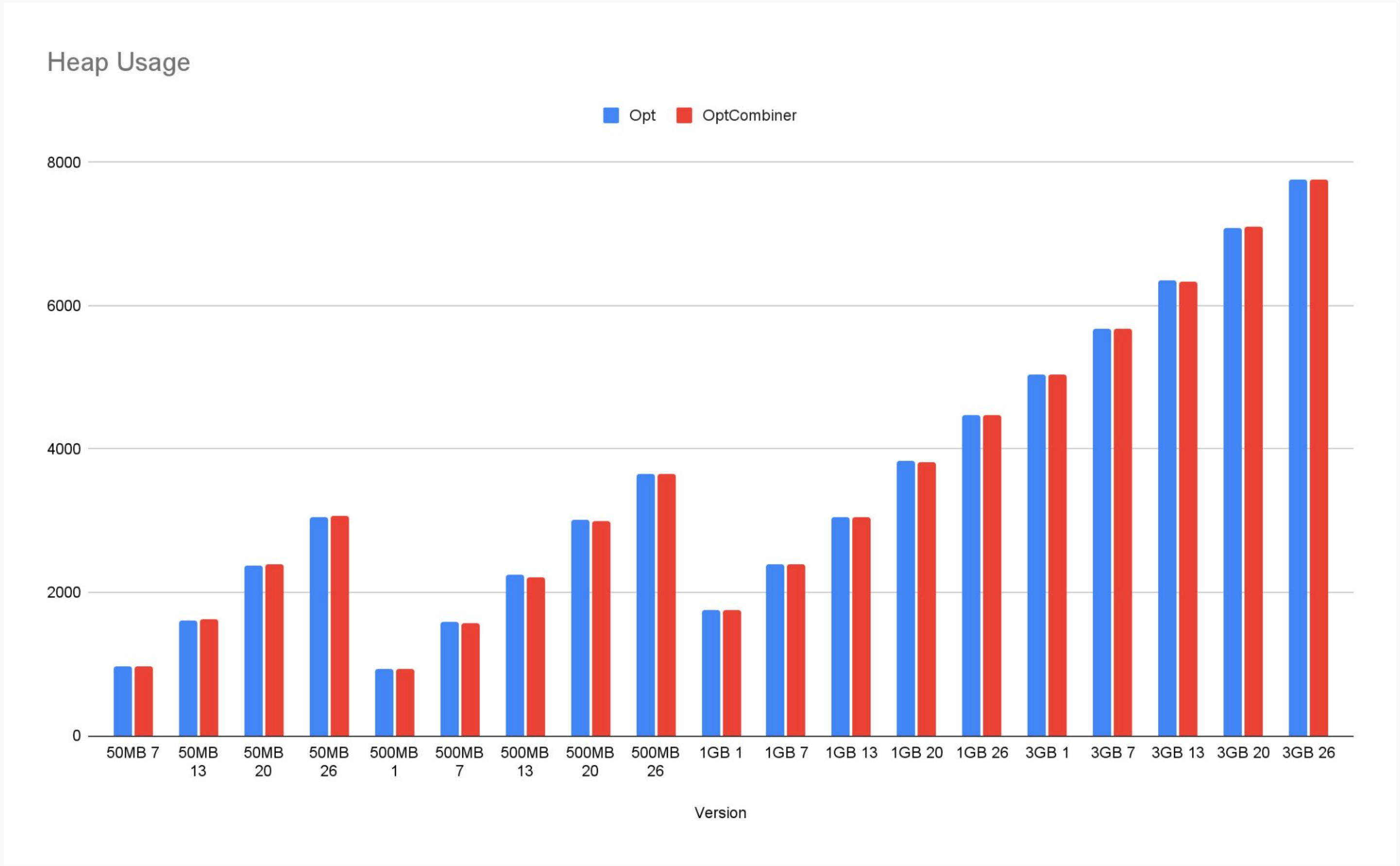
Execution Time



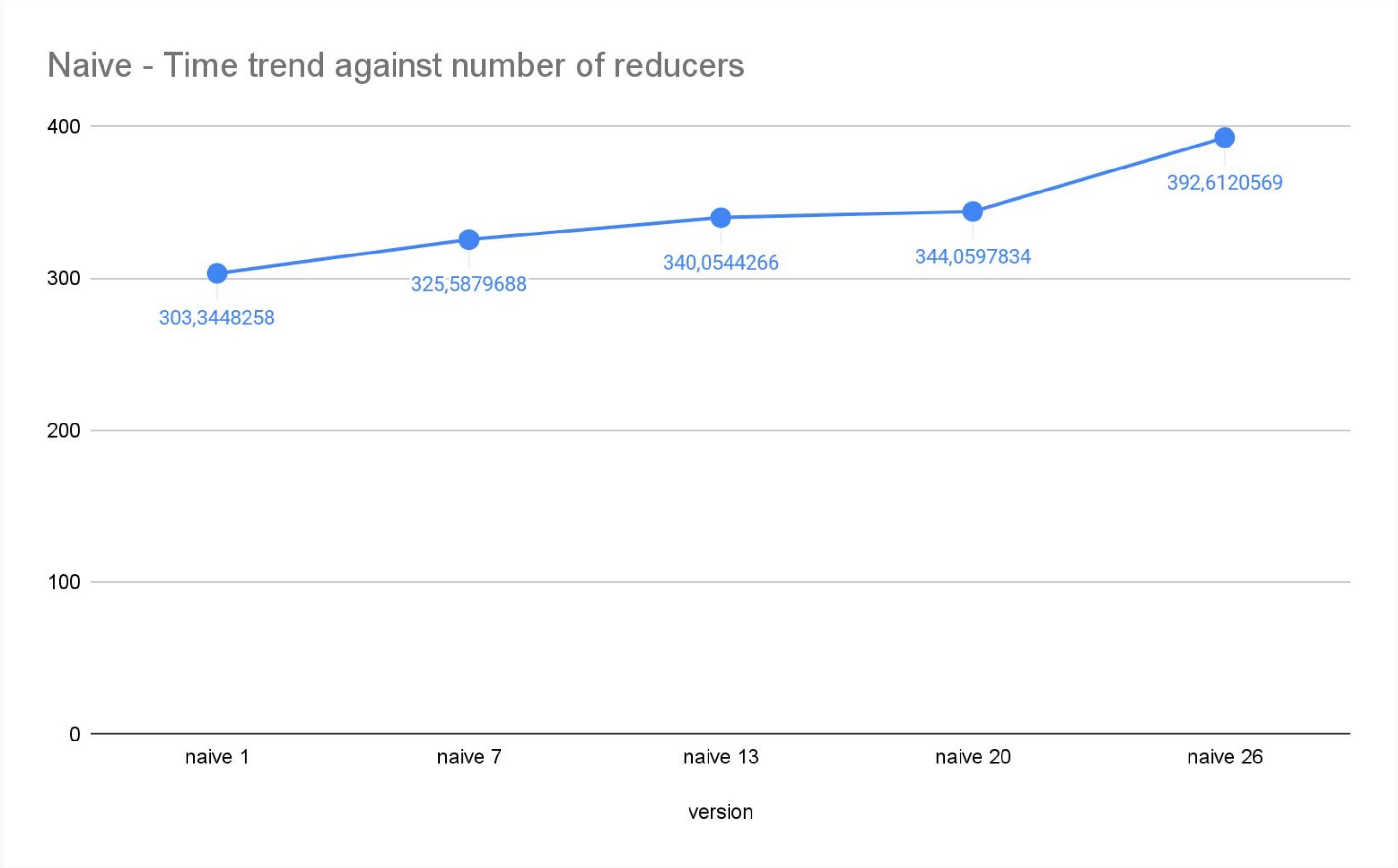
Execution Time



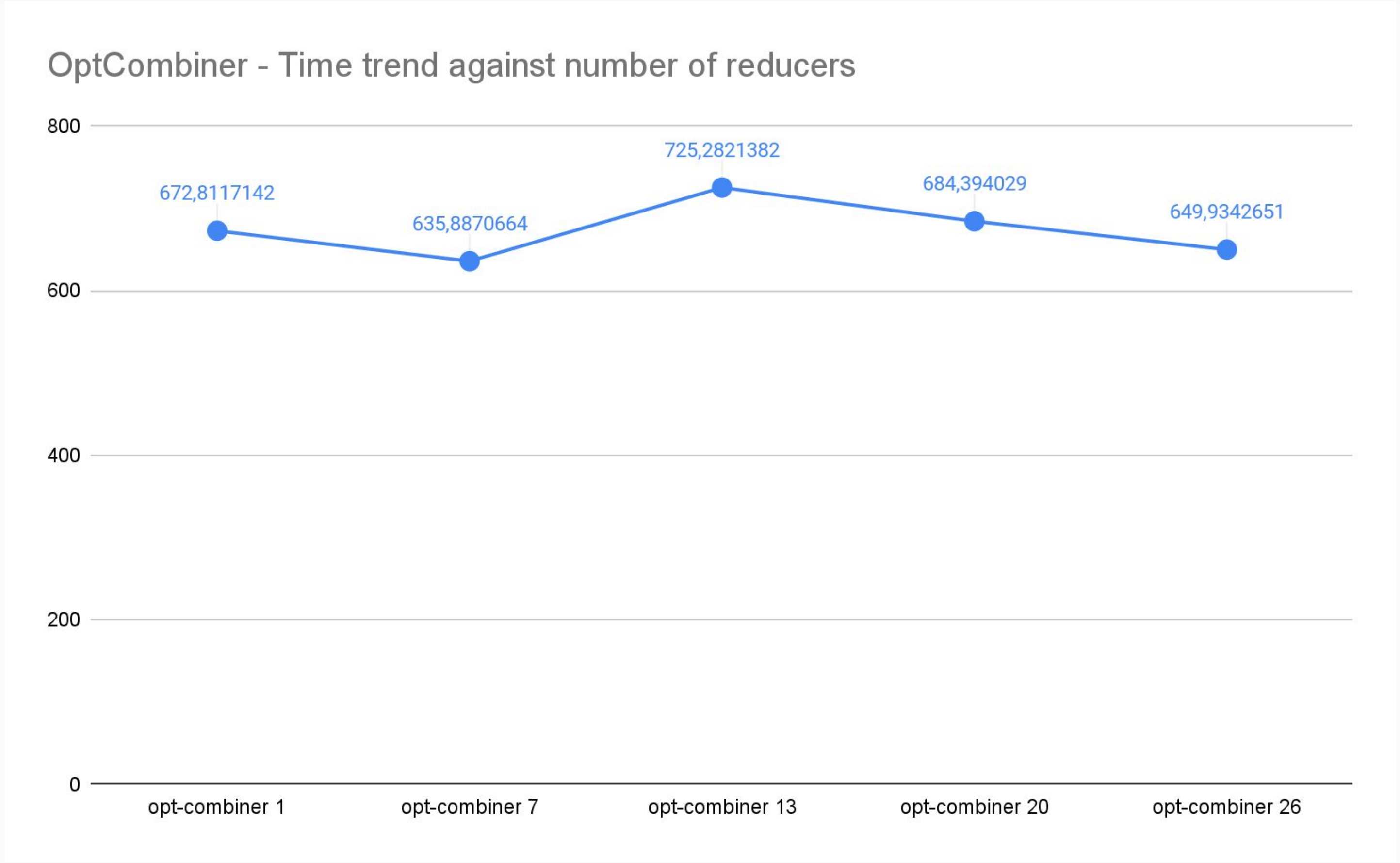
Memory Occupation



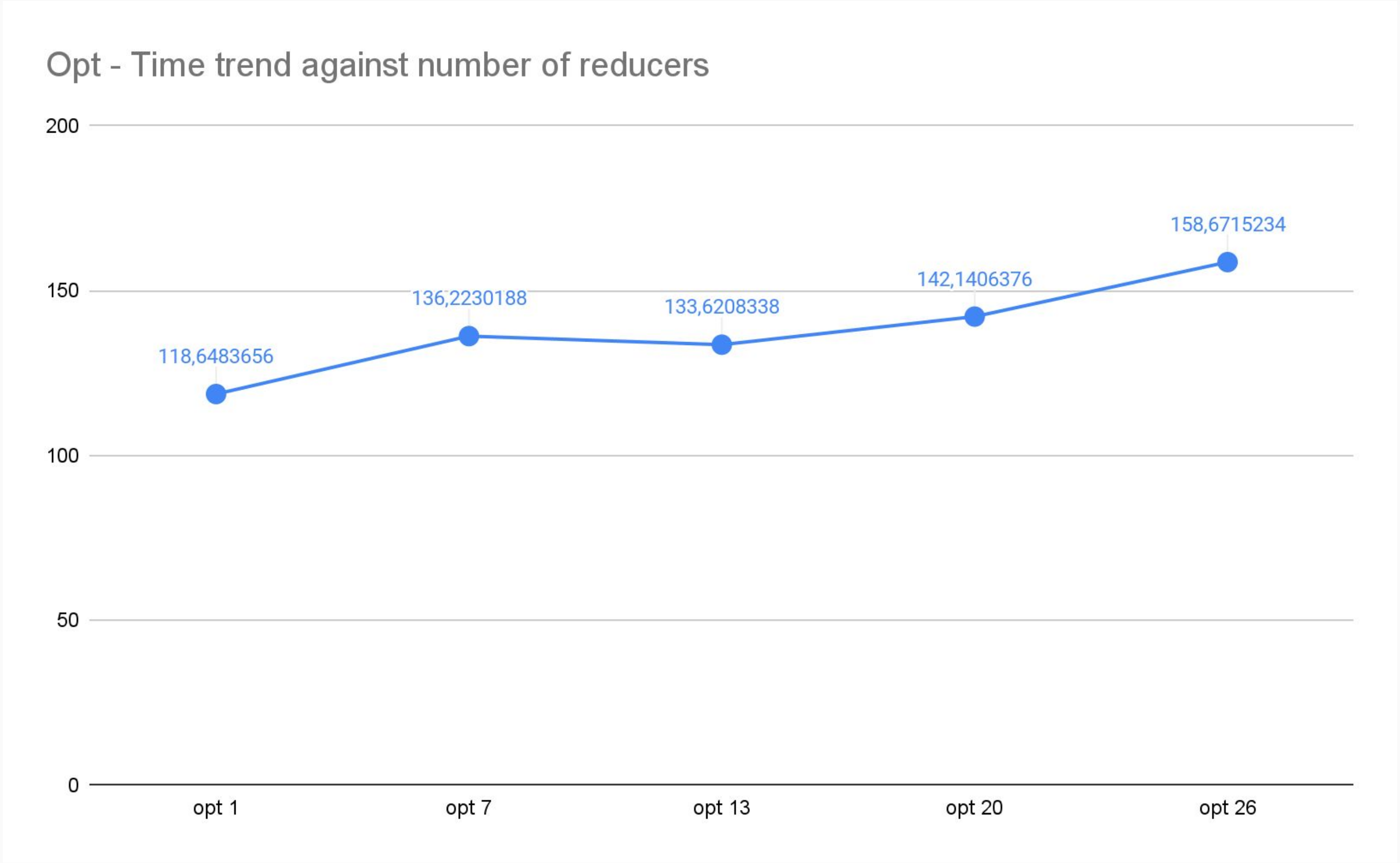
Time against reducer number



Time against reducer number



Time against reducer number



Thanks