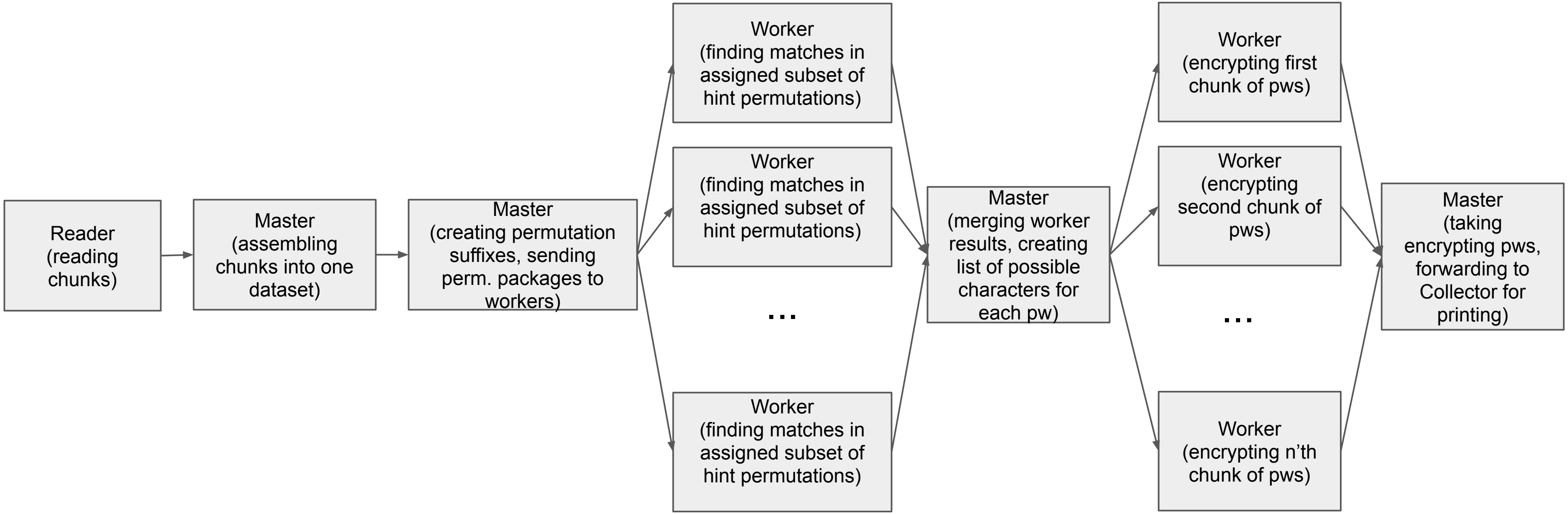
Distributed Data Exercise 1

Task 3, Tobias Pietz

General Idea



Reader (reading chunks)

Master (assembling chunks into one dataset)

Master (creating permutation suffixes, sending perm. packages to workers)

Worker (finding matches in assigned subset of hint permutations)

Worker (finding matches in assigned subset of hint permutations)

...

Worker (finding matches in assigned subset of hint permutations)

Master (merging worker results, creating list of possible characters for each pw)

Worker (encrypting first chunk of pws)

Worker (encrypting second chunk of pws)

...

Worker (encrypting n'th chunk of pws)

Master (taking encrypting pws, forwarding to Collector for printing)

General Idea notes

- Generate permutations suffixes on master (fast, not that many data to send)
- Send suffixes to workers
- Workers create all prefix permutations
- Workers try permutations
- Master merging results -> creating list of possible characters per password
- Workers try all combinations per password, sind result to master

Some thoughts

- Hints are permutations >> there are not that many hints possible
  - Generating all possible hints, hashing them and comparing with available hints using a Hashmap is faster than decrypting each hint
- First decrypting all hints before starting to decrypt passwords because hints make pw problem much easier
- Parallelising hint decryption by iterating through all permutation suffixes and giving batches of permutations to workers for trying them out (hashing them and comparing to hashed hints in Hashmap)
  - Generating permutations is cheap compared to hashing them -> generating permutations at master and doing the hashing in parallel on the workers
- Assuming that there are not many duplicates in the passwords, finally constrained passwords (some of the characters were eliminated because of hints) are decrypted in parallel by assigning a small batches of pws to workers