MapReduce

Sains Data IKOPIN

https://github.com/sevinurafni/SD3204

Kamis, 7 April 2024

How to Use Data

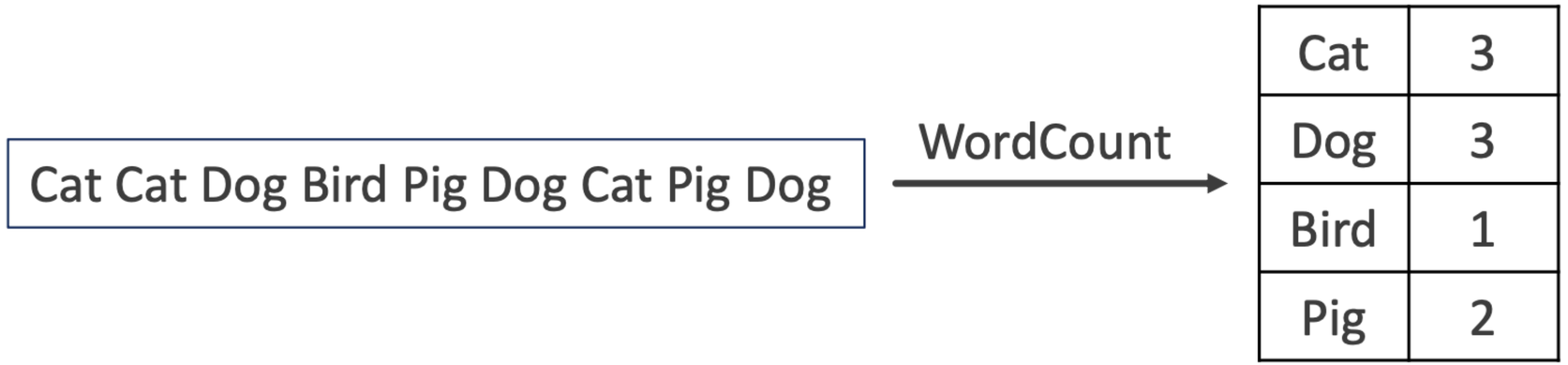
- Setelah menyimpan data di HDFS, pertanyaan pertama yang muncul di benak Anda adalah "Bagaimana cara menganalisis atau query data saya?"
  - Menyimpan data bukanlah tugas seorang analis.
- Mentransfer semua data ini ke central node untuk diproses tidak praktis.
  - Jika Anda dapat melakukan ini, mengapa Anda perlu menyimpan data secara paralel?
- HDFS memiliki sejumlah manfaat: biaya rendah, toleran terhadap kesalahan, dan mudah diskalakan,
- HDFS dirancang untuk menyimpan data secara paralel. Sekarang kita membutuhkan alat untuk memproses data secara paralel.
- MapReduce adalah solusinya!
- MapReduce terintegrasi dengan HDFS untuk memberikan manfaat yang sama persis untuk pemrosesan data.

MapReduce

- MapReduce adalah sebuah model pemrograman.
  - MapReduce sendiri bukanlah sebuah algoritma yang dapat Anda gunakan secara langsung.
  - Rancang aplikasi Anda di bawah kerangka kerja MapReduce.
- MapReduce memungkinkan programmer yang terampil untuk menulis aplikasi terdistribusi tanpa harus mengkhawatirkan infrastruktur komputasi terdistribusi yang mendasarinya.
  - Sama seperti HDFS, Anda tidak perlu khawatir tentang bagaimana data disimpan.
  - Sama halnya dengan MapReduce, Anda tidak perlu khawatir tentang bagaimana menulis algoritma terdistribusi setelah Anda mendefinisikan dua fungsi dengan baik: map dan reduce.

WordCount Example

- Aplikasi WordCount diperlukan untuk menghitung frekuensi setiap kata dalam dokumen.
  - Dokumen ini mungkin sangat besar.



Serial WordCount Algorithm

Kenyataannya adalah kita tidak akan bisa mengambil kode yang begitu sederhana dengan lancar dan menjalankannya dengan sukses pada data yang disimpan di sistem terdistribusi.

```
membuat array dua dimensi
    membuat baris untuk setiap hitungan
    kata
        isi kolom pertama dengan kata
        isi kolom kedua dengan angka nol

baca dokumen setiap kata
    Untuk setiap kata
    Temukan baris dalam array yang sesuai
    dengan kata.
    Tambahkan satu ke penghitung di
    kolom kedua.
```

Parallel WordCount Algorithm

1. Fase Map (perlu Anda implementasikan)

- Operasi pemetaan dijalankan terhadap setiap blok data dokumen secara individual.
- Kami mengekspor sepasang kunci / nilai, dengan kata sebagai kunci dan nilai berupa bilangan bulat satu.

2. Fase Shuffle dan Short (Anda tidak perlu peduli)

- Pastikan semua nilai (bilangan bulat) dikelompokkan bersama untuk setiap kunci.
- Disortir berdasarkan kunci.

3. Fase Reduce (perlu Anda implementasikan)

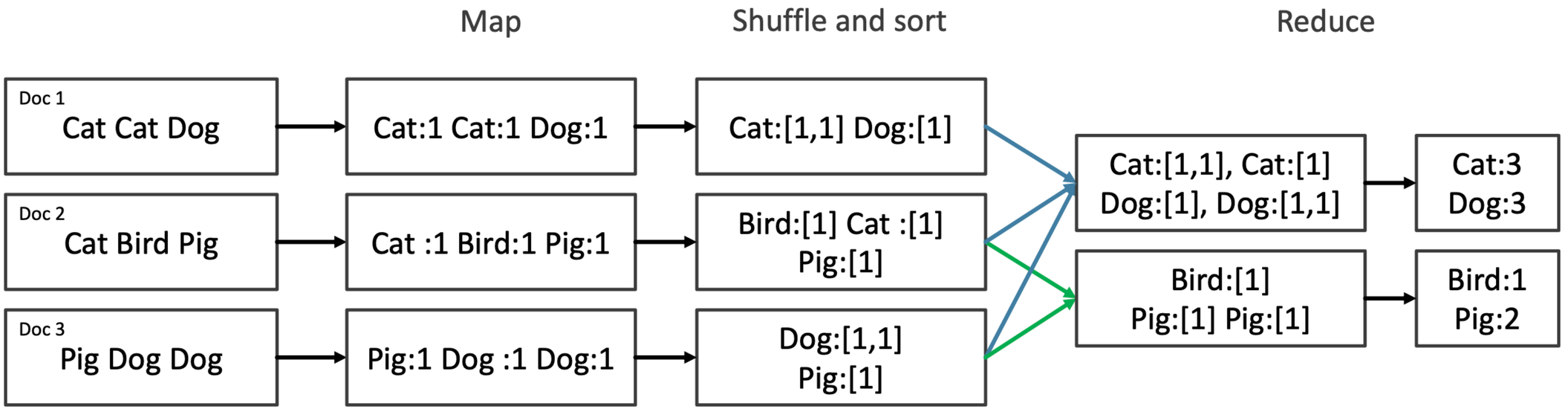
- Tambahkan total jumlah nilai satu untuk setiap kata.

Fase Pemetaan:
Baca blok data saat ini.
Keluaran kata dan bilangan bulat satu sebagai pasangan kunci/nilai.

Fase Pengacakan dan Penyortiran:
Baca daftar pasangan kunci/nilai dari fase pemetaan
kelompokkan semua nilai dengan kunci yang sama bersama-sama.
Setiap kunci memiliki sebuah array nilai yang sesuai
Urutkan data berdasarkan kunci.
Keluaran setiap kunci dan array nilai yang sesuai.

Fase Reduksi:
Baca daftar kata dan array nilai untuk setiap kata.
Tambahkan array nilai bersama-sama.

MapReduce Processing Flow



Thinking in Parallel

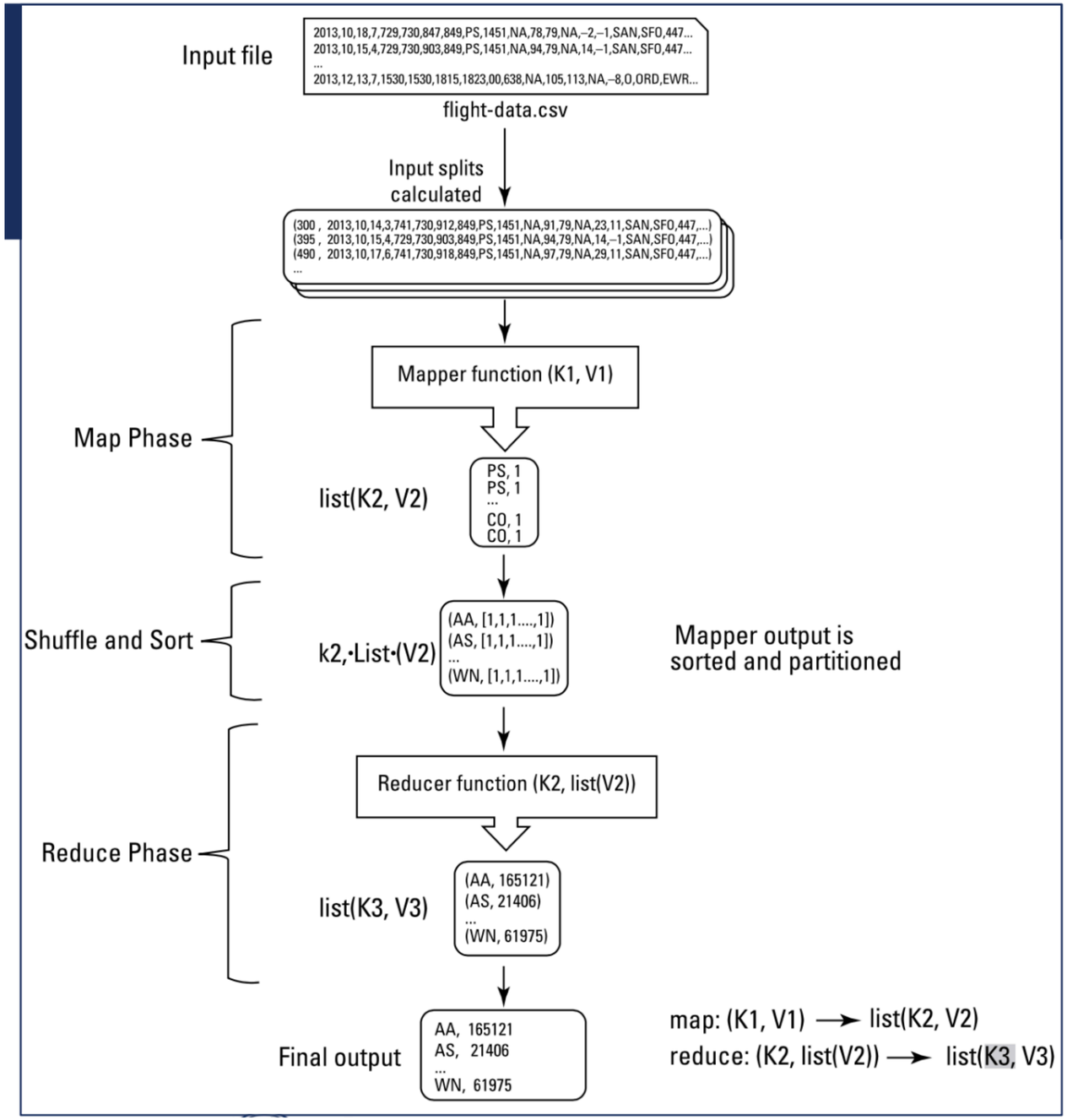
- Meskipun MapReduce menyembunyikan sejumlah kompleksitas yang besar, Anda tetap perlu tahu cara memprogram secara paralel.
- Untuk menjadi seorang MapReduce programmer dan memikirkan masalah secara paralel tidaklah mudah.

Map Phase

- Masukan ke fungsi Map diatur dalam pasangan kunci/nilai sebagai (K1, V1).
  - V1 adalah setiap record (misalnya, dokumen untuk WordCount) dan K1 adalah indeksnya.
- Fungsi pemetaan menghasilkan daftar (K2, V2).
  - K2 dan V2 benar-benar berbeda dari K1 dan V1.

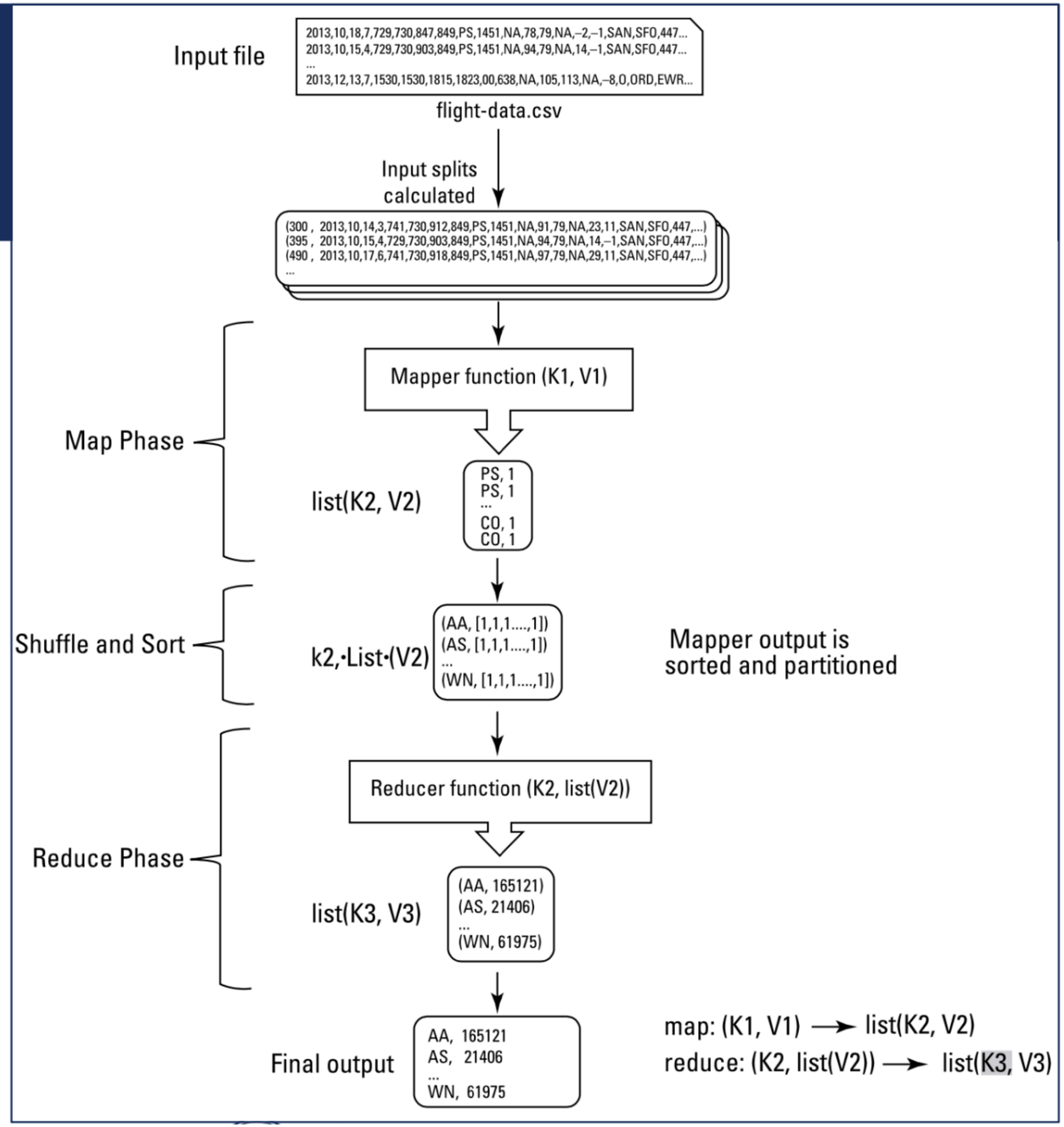
Apa yang perlu Anda lakukan dalam fase ini:

- Dengan hati-hati merancang fungsi pemetaan Anda dan menentukan apa K2 dan apa V2 seharusnya untuk setiap K2 untuk aplikasi Anda.



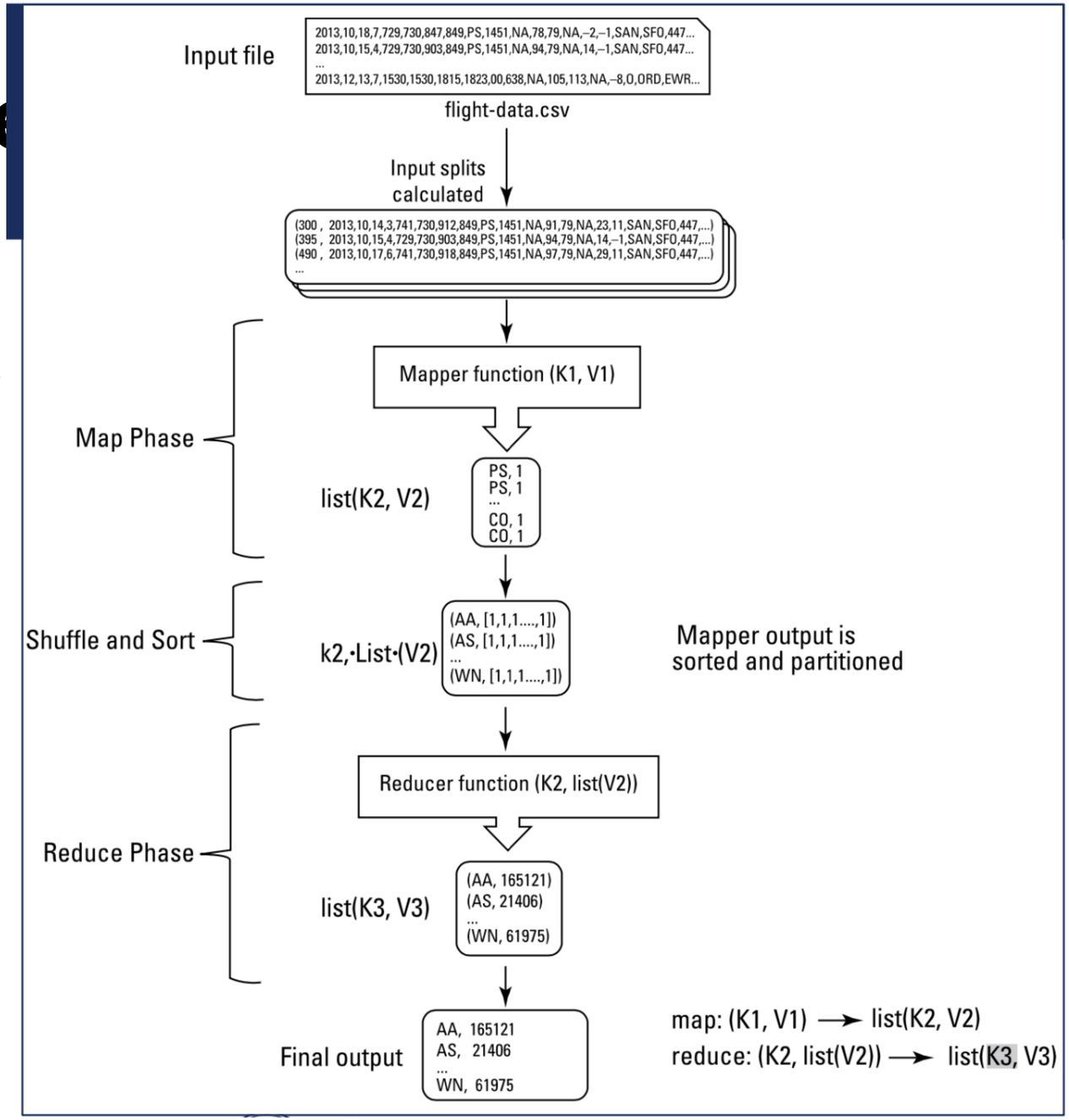
Shuffle and Short Phase

- Masih pada node map, semua nilai untuk setiap K2 dikelompokkan bersama, menghasilkan: K2, daftar(V2).
- K2, daftar(V2) kemudian ditulis ke disk node pemeta sesuai dengan K2.
  - Jumlah file yang harus ditulis tergantung pada jumlah node reduksi yang ditugaskan.
  - Setiap file berisi seperangkat kunci.
- Apa yang perlu Anda lakukan dalam fase ini:
  - Tidak ada. Ini dilakukan secara otomatis setelah Anda mendefinisikan kunci Anda dalam fungsi pemetaan.



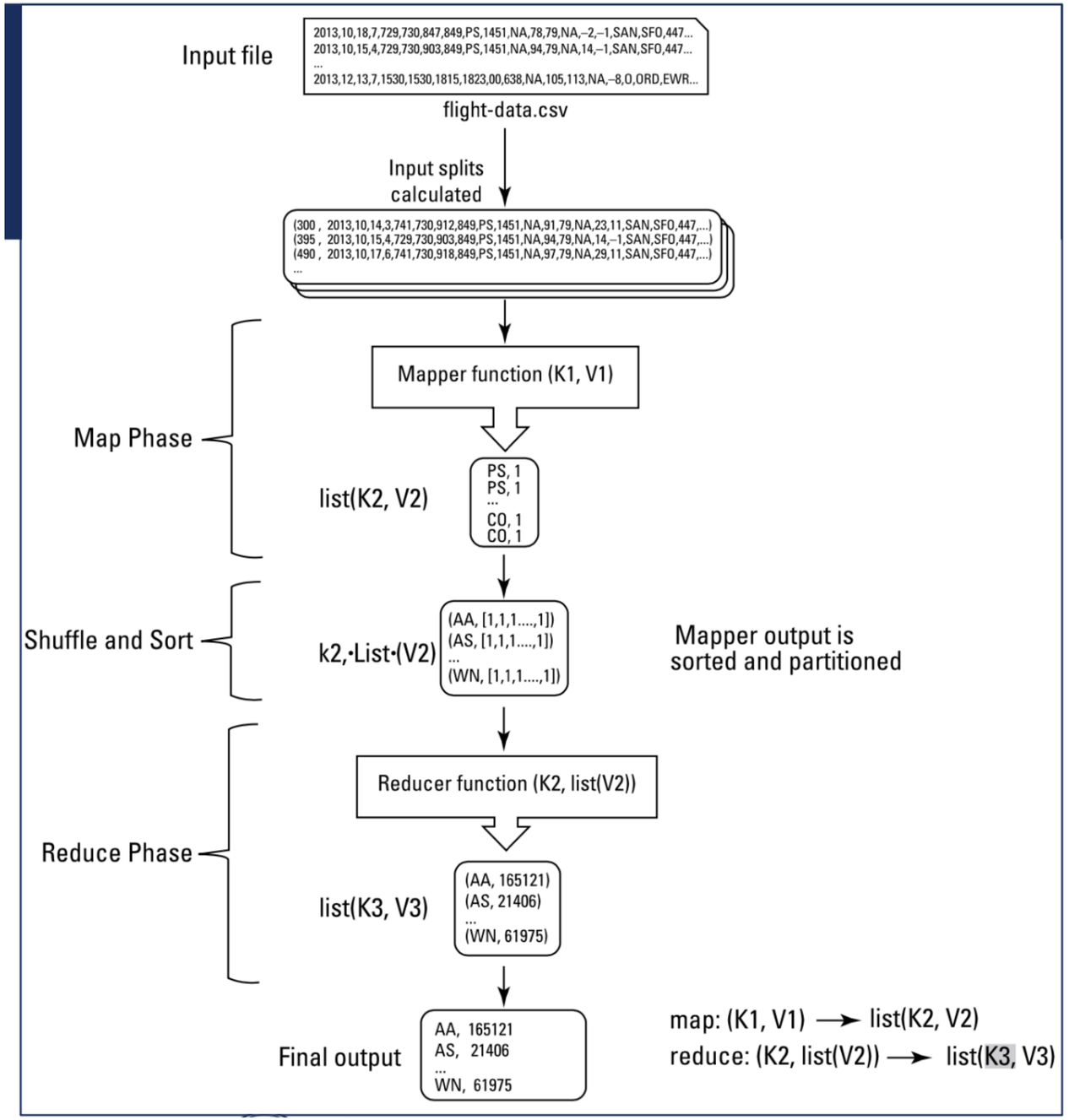
Shuffle and Short Phase

- Output dari tugas pemetaan tidak ditulis ke HDFS, melainkan ke disk lokal pada node slave tempat tugas pemetaan dijalankan.
- Sebagai hasilnya, output tersebut tidak direplikasi di seluruh klaster Hadoop.
- Mengapa?
  - Outputnya diproses oleh tugas reduksi untuk menghasilkan output akhir, dan setelah pekerjaan selesai, output pemetaan dapat dibuang.
  - Menyimpannya di HDFS dengan replikasi akan terlalu berlebihan.



Reduce Phase

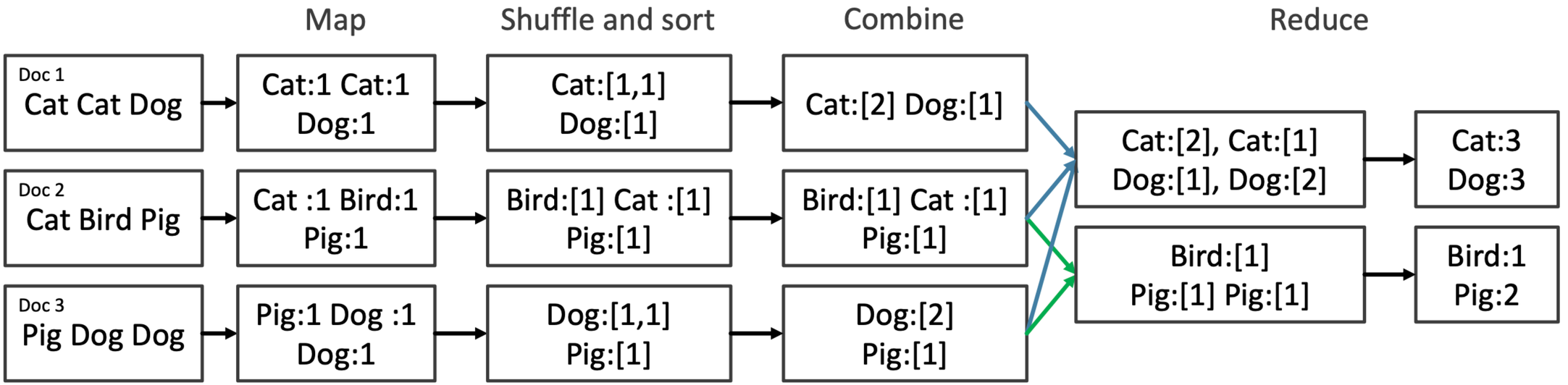
- Gabungkan semua K2, daftar(V2) dari pemeta dan kembalikan daftar(K3,V3).
  - Anda tidak perlu khawatir tentang kunci mana yang berada di node pemeta mana.
- Pemrosesan tugas reduksi tidak dapat dimulai sampai semua tugas pemeta selesai.
- Apa yang perlu Anda lakukan dalam fase ini:
  - Dengan hati-hati merancang fungsi reduksi Anda dan tentukan bagaimana membangun V3 dari V2 untuk aplikasi Anda.



Combine Function

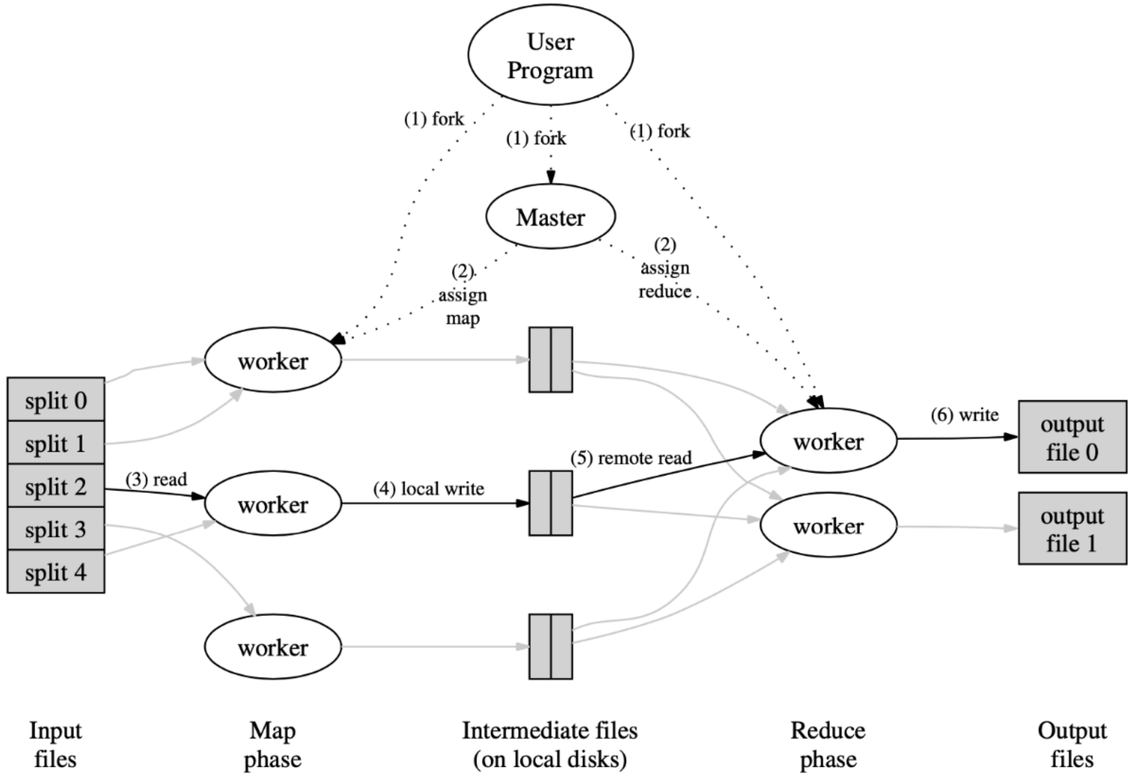
- Untuk meminimalkan transfer data antara tugas pemetaan dan reduksi, Hadoop memungkinkan pengguna untuk menentukan fungsi gabungan (combiner function) yang akan dijalankan pada output pemetaan, dan output dari fungsi gabungan tersebut menjadi masukan untuk fungsi reduksi.
- Ini melakukan penggabungan sebagian dari data ini sebelum dikirim melalui jaringan.
- Fungsi ini bersifat opsional antara pemetaan dan reduksi, tetapi tidak dapat menggantikan salah satunya.

Combine Function



Master Data Structures

- Master menyimpan beberapa struktur data. Untuk setiap tugas pemetaan dan tugas reduksi, ia menyimpan statusnya (idle, sedang berlangsung, atau selesai), dan identitas mesin pekerja (untuk tugas yang tidak idle).
- Master menugaskan M tugas pemetaan dan R tugas reduksi.
- Untuk setiap tugas pemetaan yang selesai, master menyimpan lokasi dan ukuran dari R file intermediate yang dihasilkan oleh tugas pemetaan tersebut.
- Pembaruan terhadap informasi lokasi dan ukuran ini diterima saat tugas pemetaan selesai.



Fault Tolerance

Karena pustaka MapReduce dirancang untuk membantu memproses jumlah data yang sangat besar menggunakan ratusan atau ribuan mesin, pustaka ini harus dapat menangani kegagalan mesin dengan baik.

- Kegagalan Pekerja: Jika salah satu pekerja (worker) gagal dalam menjalankan tugasnya, sistem harus dapat mendeteksinya dan secara otomatis menugaskan tugas tersebut ke pekerja lain yang masih berfungsi.
- Kegagalan Master: Jika node master mengalami kegagalan, sistem harus dapat dengan cepat mengalihkan kendali ke node master cadangan atau menciptakan kembali node master baru agar pemrosesan dapat dilanjutkan tanpa kehilangan data atau kinerja.

TUGAS

Mengapa kita membutuhkan MapReduce?

Apa saja langkah-langkah utama dari MapReduce?

THANK YOU!

Reference:

- **The official guide**: https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html#Overview
- **The paper**: Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." *Communications of the ACM* 51, no. 1 (2008): 107-113.
- **The book**: Chapter 6&7, DeRoos, Dirk. *Hadoop for dummies*. John Wiley & Sons, 2014