

BY 8 DREAM (KELOMPOK 1)



LEARNING PROGRESS REVIEW “WEEK 10”

Data Realm Engineers And Maestros

ANGGOTA KELOMPOK

- 01** AFROH FAUZIAH
- 02** ALTHAF NAWADIR TAQIYYAH
- 03** ANDI ROSILALA
- 04** ANDREW BINTANG PRATAMA
- 05** ANDREW FORTINO MAHARDIKA
SUADNYA





MapReduce



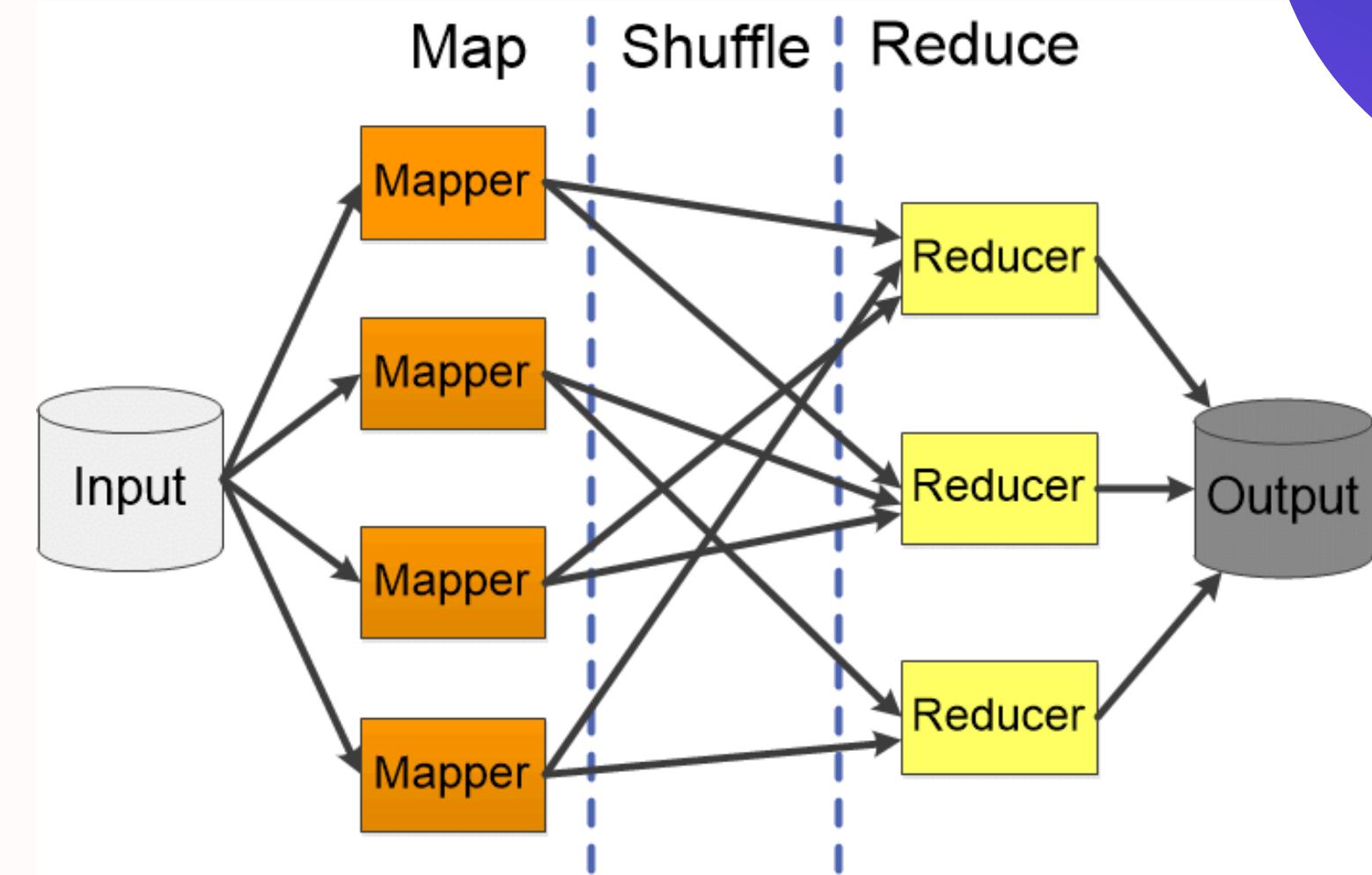
Apa itu MapReduce?

MapReduce adalah model pemrograman dan implementasi terkait yang dirancang untuk memproses dan menghasilkan kumpulan data besar secara paralel dan terdistribusi pada sebuah cluster. Sederhananya, MapReduce memecah data besar menjadi potongan-potongan kecil yang dapat diproses secara bersamaan oleh banyak komputer, kemudian menggabungkan hasilnya untuk menghasilkan keluaran yang diinginkan.

Cara Kerja MapReduce?

MapReduce bekerja dengan dua fungsi utama:

- Map: Fungsi ini memecah data besar menjadi potongan-potongan kecil dan memprosesnya secara independen. Setiap potongan data dipetakan ke kunci-nilai (key-value) yang sesuai.
- Reduce: Fungsi ini menggabungkan hasil dari fungsi map berdasarkan kunci yang sama. Nilai-nilai yang sesuai dengan kunci yang sama digabungkan untuk menghasilkan keluaran yang diinginkan.



Input dan Output (Java Perspective)

Kerangka kerja MapReduce memanfaatkan pasangan `<key, value>` sebagai format data input dan output. Ini memungkinkan pemrosesan data sebagai kumpulan pasangan `<key, value>` pada masukan dan keluaran. Kelas-kelas yang digunakan harus dapat diserialkan melalui implementasi antarmuka yang Dapat Ditulis, serta mengimplementasikan antarmuka WritableComparable untuk pengurutan data.

	Memasukkan	Keluaran
Peta	<code><k1, v1></code>	daftar (<code><k2, v2></code>)
Mengurangi	<code><k2, daftar(v2)></code>	daftar (<code><k3, v3></code>)

Secara umum, pola pemrosesan data dalam MapReduce adalah: (Input) `<k1, v1>` → map → `<k2, v2>` → reduce → `<k3, v3>` (Output). Data masukan diubah oleh fungsi map menjadi pasangan `<key, value>` `<k2, v2>`, yang kemudian disortir dan digabungkan oleh fungsi reduce untuk menghasilkan keluaran akhir `<key, value>` `<k3, v3>`.

5 penggunaan utama MapReduce

USES OF MAPREDUCE



- Entertainment: Menganalisis preferensi pengguna dan riwayat penontonan untuk merekomendasikan film dan serial populer di platform seperti Netflix.
- E-commerce: Meng evaluasi pola pembelian konsumen untuk memberikan rekomendasi produk dan meningkatkan manajemen inventaris bagi perusahaan seperti Flipkart dan Amazon.
- Social Media: Memproses jumlah data besar dari platform seperti Twitter untuk melakukan tugas seperti tokenisasi, penyaringan, dan perhitungan.
- Data Warehouse: Menangani volume data besar dan mengoptimalkan kecepatan eksekusi kueri dalam sistem gudang data menggunakan logika bisnis khusus.
- Fraud Detection: Memanfaatkan pemrosesan data berskala besar untuk mengidentifikasi pola dan anomali untuk deteksi kecurangan di institusi keuangan seperti bank dan perusahaan asuransi.

~DEFINITION~

- “MapReduce is a programming model and an associated implementation for processing and generating large data sets”
- Programming model : Abstractions to express simple computations
- Library

Takes care of the gory stuff:

Parallelization, Fault Tolerance, Data Distribution and Load Balancing

- Terms are borrowed from Functional Language (e.g., Lisp)

Sum of squares:

- (map square '(1 2 3 4))
 - Output: (1 4 9 16)

[processes each record sequentially and independently]

- (reduce + '(1 4 9 16))
 - (+ 16 (+ 9 (+ 4 1)))
 - Output: 30

[processes set of all records in batches]



~Programming Model~

- To generate a set of output key-value pairs from a set of input key-value pairs
 - $\{ \langle ki, vi \rangle \} \rightarrow \{ \langle ko, vo \rangle \}$
- Expressed using two abstractions:
 - Map task : $\langle ki, vi \rangle \rightarrow \{ \langle kint, vint \rangle \}$
 - Reduce task : $\langle kint, \{ vint \} \rangle \rightarrow \langle ko, vo \rangle$

- Library

aggregates all the intermediate values associated with the same intermediate key

passes the intermediate key-value pairs to reduce function

- ‘map’ & ‘reduce/fold’ of functional programming languages
 - (map *f list [list2 list3 ...]*)
 - (map square (1 2 3 4)) à (1 4 9 16)
 - (reduce *f list [...]*)
 - (reduce + (1 4 9 16)) à 30
 - (reduce + (map square (map – l1 l2))))

~EXAMPLE OF WORD COUNT~

```
map(String input_key, String input_value):
    // input_key: document name
    // input_value: document contents
    for each word w in input_value:
        EmitIntermediate(w, "1");
```

```
<“Sam”, “1”>, <“Apple”, “1”>, <“Sam”, “1”>, <“Mom”, “1”>,
<“Sam”, “1”>, <“Mom”, “1”>,
```

```
reduce(String output_key, Iterator intermediate_values):
    // output_key: a word
    // output_values: a list of counts
    int result = 0;
    for each v in intermediate_values:
        result += ParseInt(v);
    EmitAsString(result);
```

```
<“Sam” , [“1”,“1”,“1”]>, <“Apple” , [“1”]>, <“Mom” , [“1”, “1”]>
“3”
“1”
“2”
```

01

Distributed Grep

- Input: large set of files
- Output: lines that match pattern
- Map – Emits a line if it matches the supplied pattern
- Reduce – Copies the intermediate data to output

02

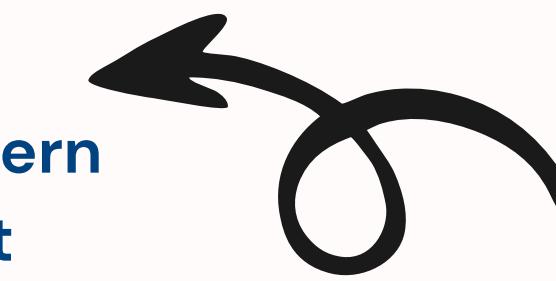
Reverse Web-Link Graph

- Input: Web graph: tuples (a, b) where (page a à page b)
- Output: For each page, list of pages that link to it
- Map – process web log and for each input <source, target>, it outputs <target, source>
- Reduce – emits <target, list(source)>

03

Count of URL access frequency

- Input: Log of accessed URLs, e.g., from proxy server
- Output: For each URL, % of total accesses for that URL
- Map – Process web log and outputs <URL, 1>
- Multiple Reducers – Emits <URL, URL_count>
(So far, like Wordcount. But still need %)
- Chain another MapReduce job after above one
- Map – Processes <URL, URL_count> and outputs <1, (<URL, URL_count>)>
- 1 Reducer – Sums up URL_count's to calculate overall_count.
Emits multiple <URL, URL_count/overall_count>



~SOME
APPLICATIONS~

04

- Map task's output is sorted (e.g., quicksort)
- Reduce task's input is sorted (e.g., mergesort)

~LIFECYCLE OF A MAPREDUCE JOB~



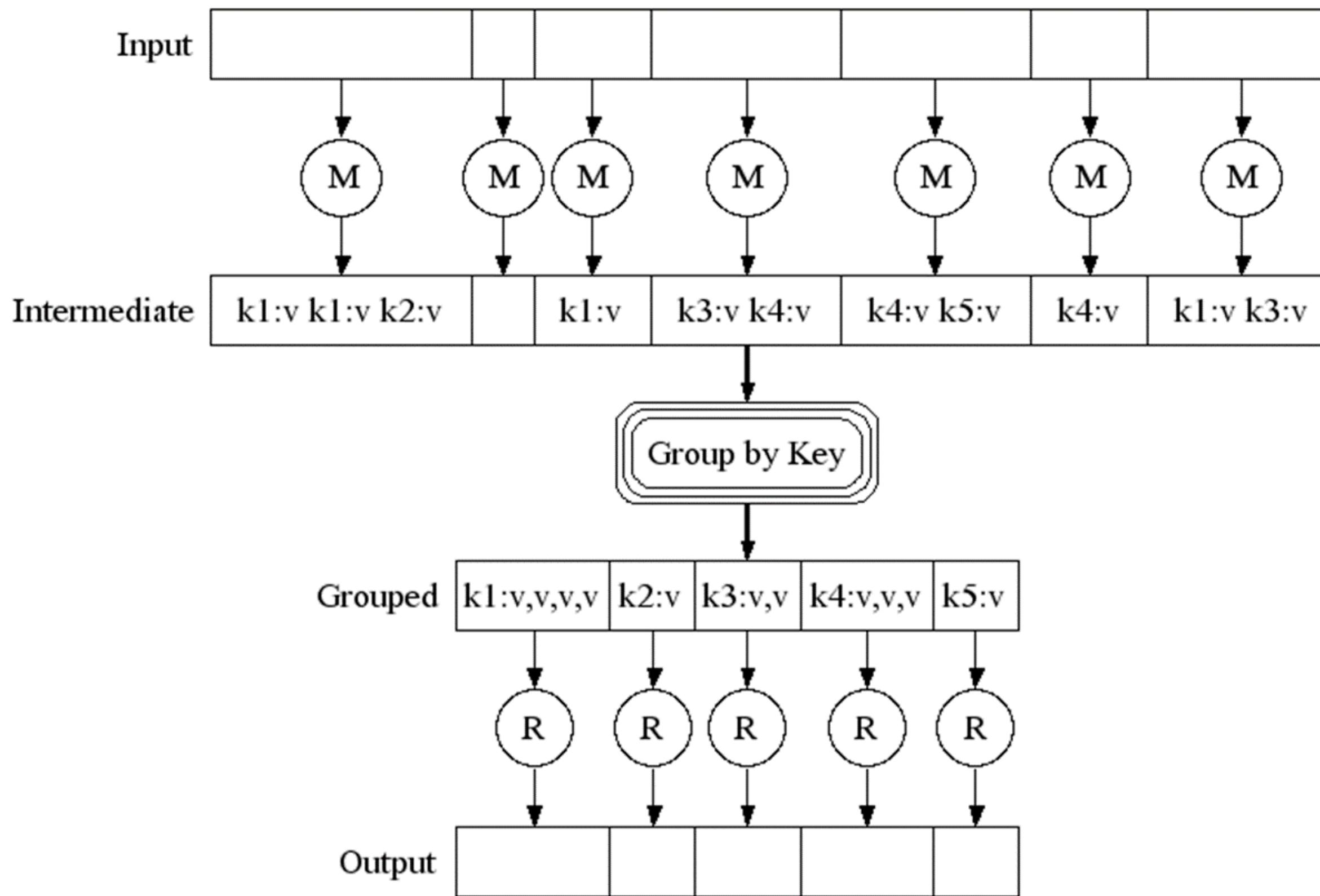
```
File Edit Options Buffers Tools Java Help  
public class WordCount {  
    public static class Map extends MapReduceBase implements  
        Mapper<LongWritable, Text, Text, IntWritable> {  
        private final static IntWritable one = new IntWritable(1);  
        private Text word = new Text();  
  
        public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable>  
            output, Reporter reporter) throws IOException {  
            String line = value.toString();  
            StringTokenizer tokenizer = new StringTokenizer(line);  
            while (tokenizer.hasMoreTokens()) {  
                word.set(tokenizer.nextToken());  
                output.collect(word, one);  
            }  
        }  
  
        public static class Reduce extends MapReduceBase implements  
            Reducer<Text, IntWritable, Text, IntWritable> {  
            public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text,  
                IntWritable> output, Reporter reporter) throws IOException {  
                int sum = 0;  
                while (values.hasNext()) { sum += values.next().get(); }  
                output.collect(key, new IntWritable(sum));  
            }  
  
            public static void main(String[] args) throws Exception {  
                JobConf conf = new JobConf(WordCount.class);  
                conf.setJobName("wordcount");  
                conf.setOutputKeyClass(Text.class);  
                conf.setOutputValueClass(IntWritable.class);  
                conf.setMapperClass(Map.class);  
                conf.setCombinerClass(Reduce.class);  
                conf.setReducerClass(Reduce.class);  
                conf.setInputFormat(TextInputFormat.class);  
                conf.setOutputFormat(TextOutputFormat.class);  
                FileInputFormat.setInputPaths(conf, new Path(args[0]));  
                FileOutputFormat.setOutputPath(conf, new Path(args[1]));  
  
                JobClient.runJob(conf);  
            }  
        }  
    }  
}  
---- mapreduce.java All L9 (Java/l Abbrev)-----  
Wrote /home/shivnath/Desktop/mapreduce.java
```

Map function

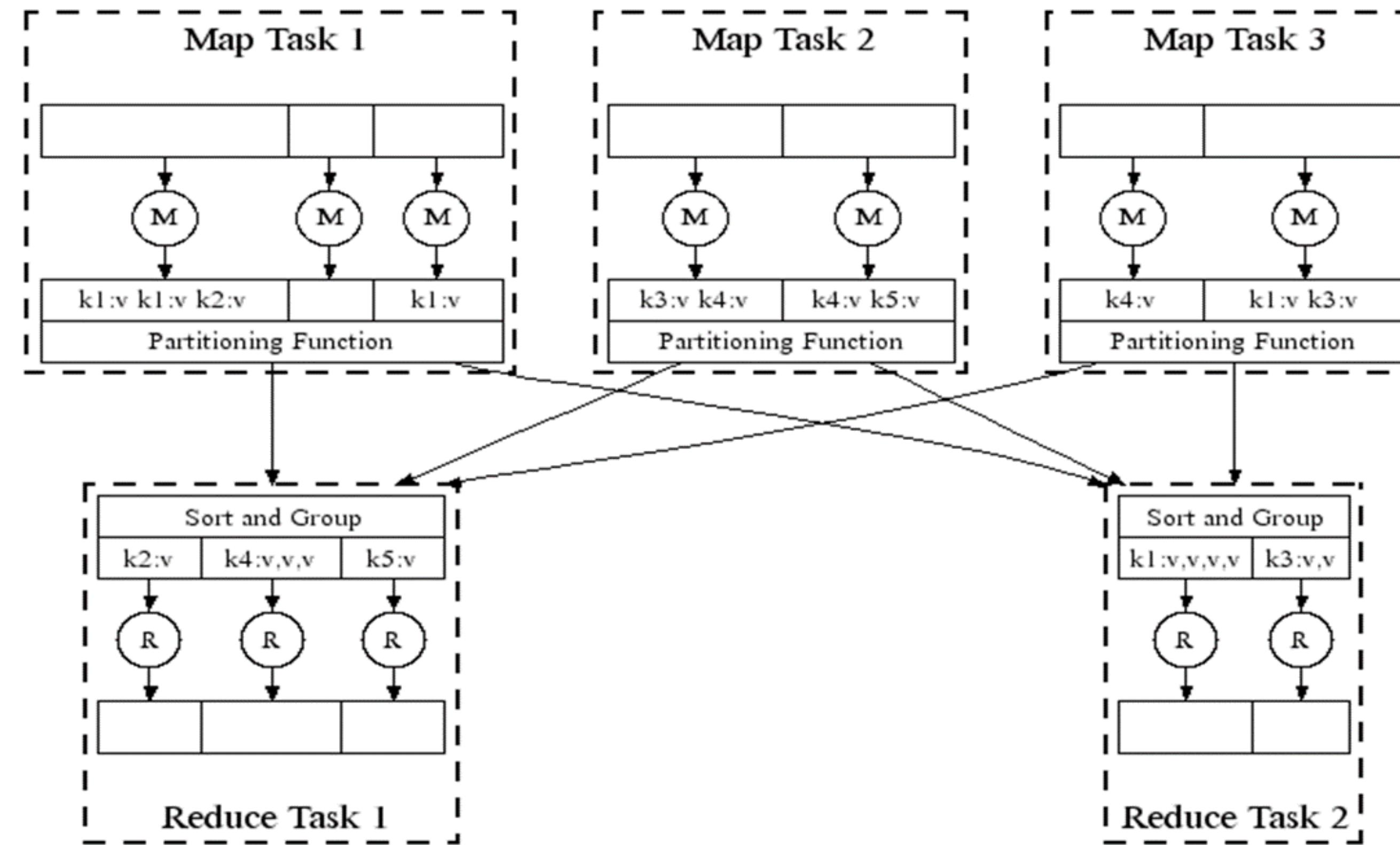
Reduce function

Run this program as a
MapReduce job

~THE MAPREDUCE FRAMEWORK (PIONEERED BY GOOGLE)~



~AUTOMATIC PARALLEL EXECUTION IN MAPREDUCE (GOOGLE)~



Handles failures automatically,
e.g., restarts tasks if a node fails;
runs multiples copies of the
same task to avoid a slow task
slowing down the whole job

LEADERSHIP SKILL





~KONSEP~

“Leader” > seseorang yang memiliki otoritas, pengaruh, kemampuan untuk memimpin atau mengarahkan orang lain dalam mencapai tujuan tertentu secara bersama. Pemimpin memiliki kualitas kepemimpinan yang dapat menginspirasi, memotivasi, memiliki visi yang jelas, serta mampu membimbing anggota tim atau organisasi menuju visi atau tujuan bersama.

“Leadership” > kegiatan atau aksi seorang leader dalam mempengaruhi, membimbing dan menginspirasi orang lain untuk mencapai tujuan bersama. Leadership melibatkan kemampuan atau kualitas pemimpin untuk mengelola tim secara efektif, membuat strategi, mengambil keputusan yang tepat, pengembangan visi, pembuatan strategi, komunikasi efektif, pemberian contoh yang baik, kemampuan memecahkan masalah serta memotivasi tim mencapai hasil yang diinginkan.



~KONSEP~

- Kepemimpinan adalah kombinasi antara faktor bawaan / lahir (born) dan pengembangan (made).
- Meskipun beberapa orang memiliki kecenderungan alami untuk menjadi leader, keterampilan kepemimpinan dapat dan harus diperkuat melalui pembelajaran dan pengalaman yang dapat dikembangkan.
- Baik faktor bawaan maupun pengembangan berperan penting dalam membentuk seorang leader yang efektif.
- Perlu adanya kesadaran tentang kemampuan diri terutama untuk dapat memimpin diri sendiri dan meningkatkan kemampuan sehingga mampu memimpin orang lain.

~KOMPONEN PENTING DARI KEPEMIMPINAN~

Self Awareness (Kesadaran Diri)

Kesadaran diri yang baik mampu memahami pikiran, kekuatan, kelemahan, nilai-nilai, dan dampak emosional dari perilaku mereka terhadap orang lain. Sehingga dapat mengenali bagaimana tindakan dan keputusan yang akan mempengaruhi tim atau organisasi.

Self Management (Manajemen Diri)

Kemampuan untuk mengelola pikiran, emosi, stres, waktu secara efektif dan perilaku dalam berbagai situasi kepemimpinan. Seorang pemimpin yang baik dapat mengendalikan impuls, menyesuaikan diri dengan perubahan, dan mengatasi tantangan dengan baik.

Self Motivation (Motivasi Diri)

Seorang pemimpin yang termotivasi secara internal memiliki tekad tinggi demi mencapai keberhasilan, bahkan dalam menghadapi rintangan atau kesulitan. Sehingga dapat mendorong dirinya maju mencapai tujuan dan memimpin orang lain dengan contoh yang baik.

~TIPE PARENTING STYLE~

Otoriter (Authoritarian)

Ditandai dengan aturan yang ketat, kepatuhan, kedisiplinan dan pengendalian yang kuat.



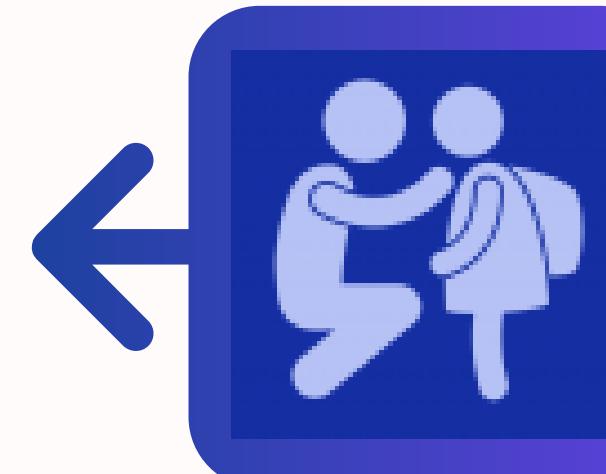
Permisif (Permissive)

Ditandai dengan kurangnya aturan yang ketat dan konsistensi dalam menerapkan batasan, lebih bersahabat, juga terbuka.



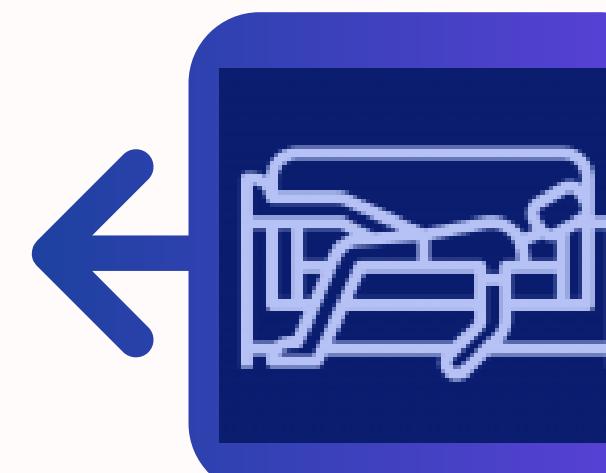
Otoritatif (Authoritative)

Ditandai dengan penciptaan keseimbangan aturan yang jelas, hangat, terbuka, konsekuensi dari tindakan dan dukungan emosional.



Abai (Uninvolved / Neglectful)

Ditandai dengan ketidak terlibatan dan kurang peduli kebutuhan emosional maupun fisik.



Providing (Memberikan)

Seorang pemimpin harus memberikan segala hal yang diperlukan untuk mendukung pertumbuhan dan perkembangan seseorang, baik secara fisik maupun emosional, sumber daya, bimbingan, dan dukungan yang dibutuhkan oleh timnya untuk mencapai tujuan.

Motivating (Memotivasi)

Seorang pemimpin harus mampu memotivasi dan menginspirasi untuk mencapai hasil yang lebih baik dan mengetahui nilai-nilai tim untuk merangsang minat, semangat, dan dorongan internal agar berkembang.

Empowering (Memberdayakan)

Seorang pemimpin harus memberikan kekuatan atau keterampilan untuk memberdayakan tim dengan memberi otonomi / kontrol untuk bertanggung jawab dan kepercayaan mengambil inisiatif juga keputusan.

**-ELEMEN KUNCI
PENDEKATAN
PARENTING
STYLE YANG
EFEKTIF-**



Supportive Communication (Komunikasi yang Mendukung)

Melibatkan berkomunikasi dengan cara yang positif, membangun, dan mendukung, termasuk memberikan umpan balik yang konstruktif dan saling pengertian di antara anggota tim.

01

Transparency (Transparansi)

Melibatkan kejelasan dan keterbukaan, berbagi informasi yang relevan dan jujur, membangun kepercayaan, memastikan keselarasan dan komitmen, serta pertanggungjawaban atas keputusan pemimpin.

02

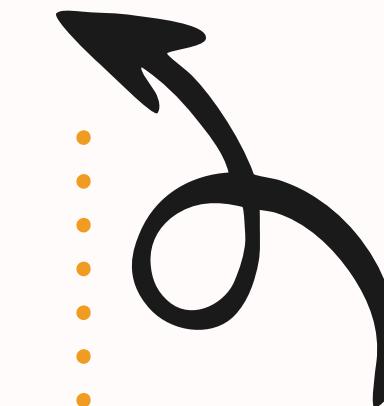
Active Listening (Pendengaran Aktif)

Melibatkan keterampilan mendengarkan dengan penuh perhatian dan empati, memahami perspektif dan perasaan, dan memberikan umpan balik, serta mengidentifikasi masalah dan cara menyelesaiakannya.

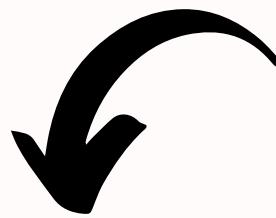
03

Supportive (Pendukung)

Melibatkan memberikan dukungan emosional, moral, dan praktis yang konstruktif dalam mencapai tujuan, serta memberikan bantuan dan bimbingan kepada tim guna meningkatkan kinerja / perilaku.



~STRATEGI KOMUNIKASI~



Decision Making

(Pembuatan Keputusan)

Keterampilan penting pemimpin, memberikan kesempatan untuk berpartisipasi, berkontribusi, dan memahami alasan di balik keputusan yang dibuat, harus cepat dan tegas, juga bertanggung jawab atas konsekuensinya.

— — — — — — — — — —

keputusan yang dibuat, harus cepat dan tegas, juga bertanggung jawab atas konsekuensinya.

~KETERAMPILAN KEPEMIMPINAN~

Creativity (Kreativitas)

Kemampuan menghasilkan ide baru, solusi inovatif, dan pendekatan yang segar untuk menyelesaikan masalah, menetapkan arah dan tujuan tim secara keseluruhan seperti visi misi.

01

Gathering Info (Mengumpulkan Informasi)

Mengidentifikasi sumber informasi, mengevaluasinya dengan kritis, mempertimbangkan beragam perspektif dan pendapat, menggunakan data untuk keputusan / tindakan.

02

Making Tough Calls (Mengambil Keputusan Sulit)

Keberanian mengambil resiko, mempertimbangkan konsekuensi setiap pilihan dan manfaat, menilai potensi hasil, dan memilih tindakan terbaik dengan keyakinan untuk kesuksesan jangka panjang.

03

Taking Responsibilities

(Bertanggung Jawab)

Atas tindakan dan keputusan, hasil yang dicapai positif maupun negatif, belajar dari keberhasilan dan kegagalan. Seperti mengakui kesalahan dan mengambil langkah untuk memperbaiki atau menyelesaikan masalah.

04

~ASPEK KRITIS YANG HARUS DIPAHAMI DAN DIKELOLA DENGAN BAIK OLEH PEMIMPIN~

O1 Ethics (Etika)

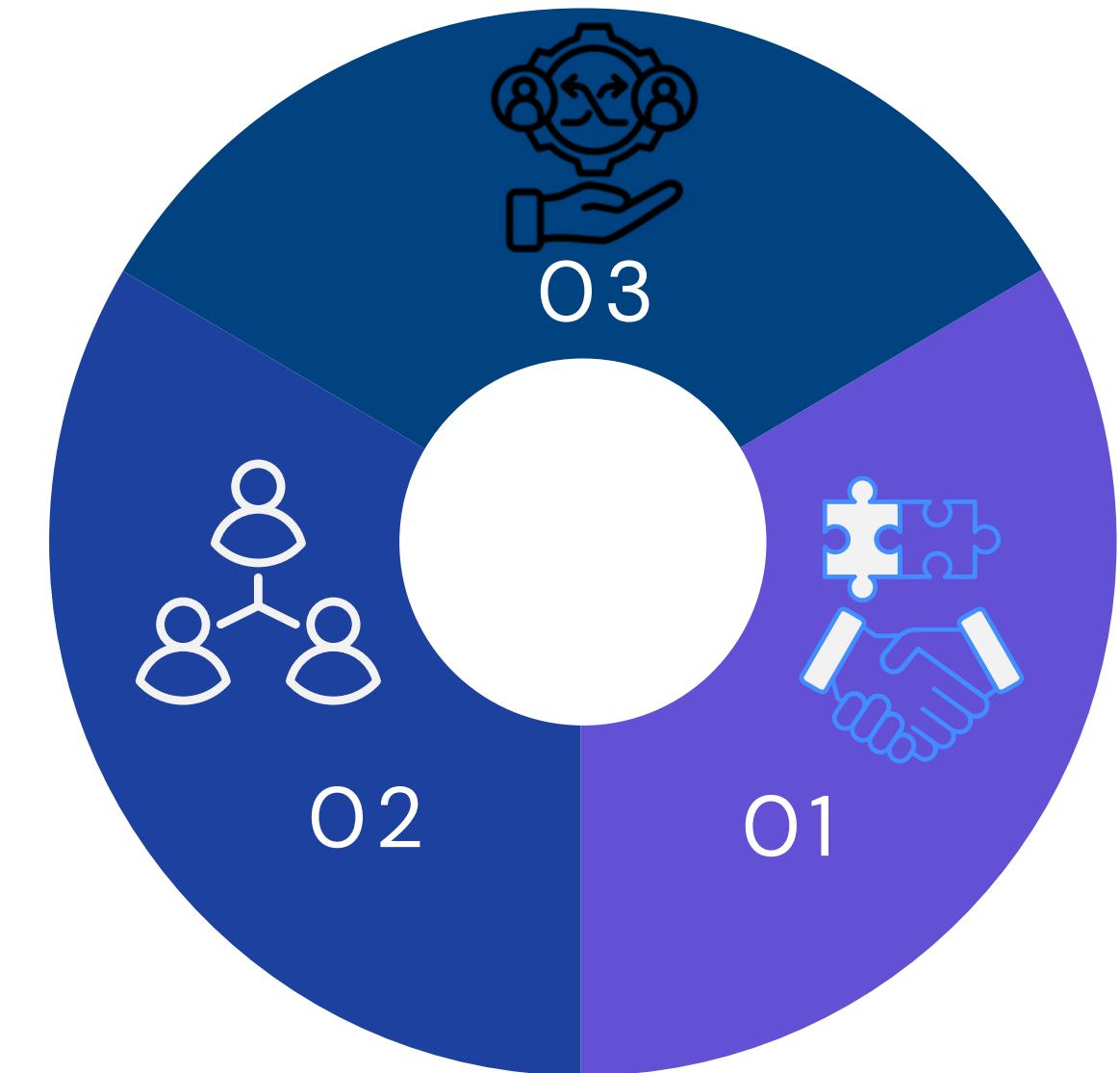
Mengacu pada prinsip-prinsip moral dan nilai-nilai yang membimbing perilaku dan keputusan, mengatasi tantangan, juga memahami dan menerapkan standar etika.

O2 Culture (Budaya)

Merujuk pada keyakinan dan praktik dalam tim, memahami ragam budaya timnya dan berperan dalam membentuk dan memelihara budaya yang positif dan produktif.

O3 Change Management (Manajemen Perubahan)

Proses merencanakan, mengimplementasikan, dan mengevaluasi perubahan, merencanakan, mendorong dan memantau pelaksanaan perubahan berkelanjutan agar sesuai dengan harapan semua pihak.



SPARK I

spark³

WHAT IS SPARK?

Apache Spark merupakan sebuah unified computing engine and set of libraries for parallel data processing pada suatu computer clusters. Sistem ini memanfaatkan caching dalam memori dan eksekusi kueri yang dioptimalkan untuk kueri analitik cepat terhadap data dengan segala ukuran.

Spark supports beberapa programming language terkenal, seperti Python, Java, Scala, R. Task yang dilakukan meliputi dari SQL, streaming, bahkan machine learning (Big Data).

WHY SPARK?

- Spark dibuat untuk mengatasi keterbatasan MapReduce dengan melakukan pemrosesan dalam memori, yang mengurangi jumlah langkah dalam suatu tugas, dan dengan menggunakan kembali data di banyak operasi paralel.
- Spark hanya memerlukan satu langkah, yaitu data dibaca ke dalam memori, operasi dilakukan, dan hasilnya ditulis kembali—menghasilkan eksekusi yang jauh lebih cepat.

SPARK ARCHITECTURE

Driver processes

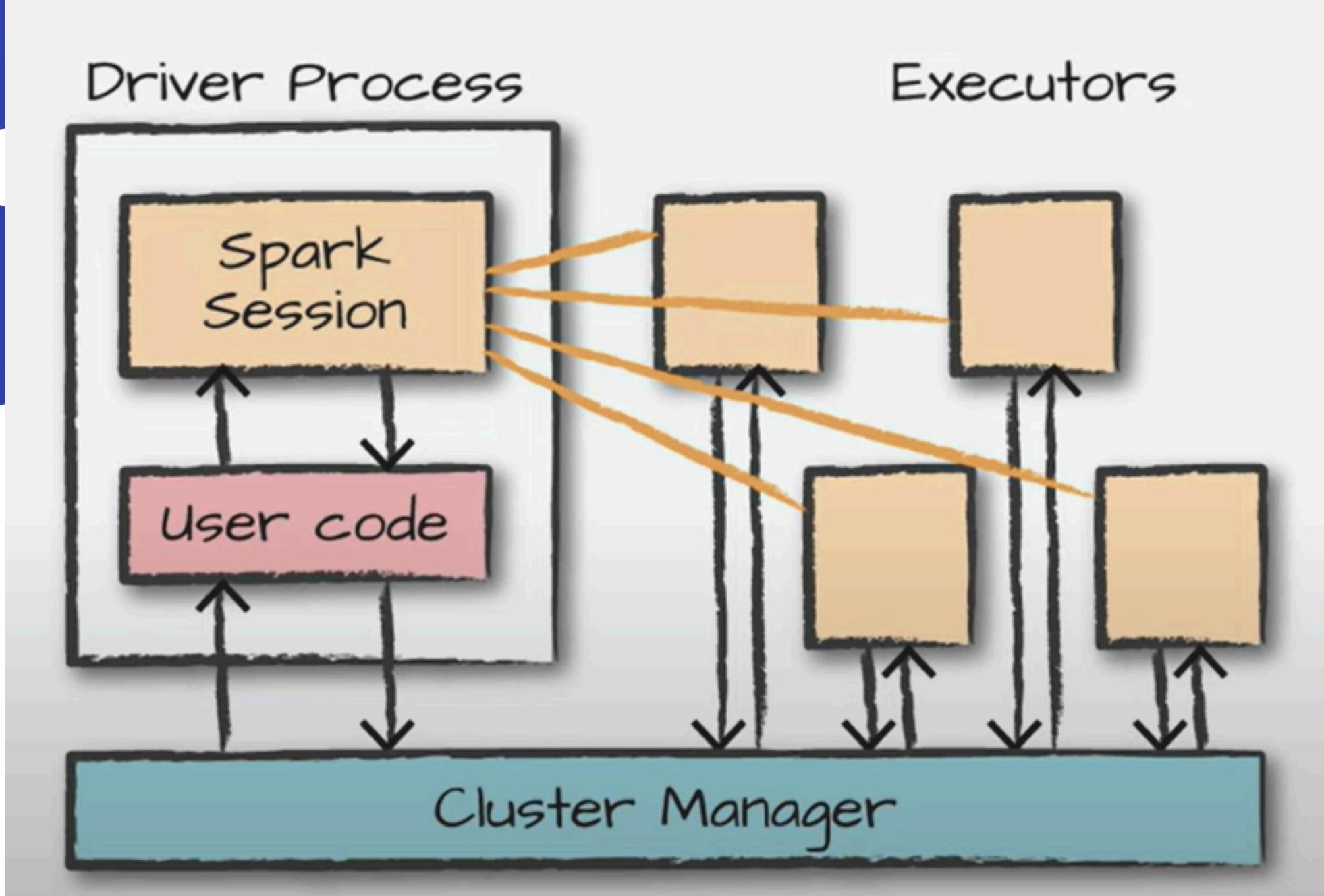
- Maintain informasi terkait spark app
- Responsif terhadap user dan program input
- Melakukan *Analyzing, distributing, scheduling work across executors*

Executors processes

- Execute code yang telah diassign oleh driver
- melaporkan state dari komputasi kembali ke driver



SPARK ARCHITECTURE



- Spark menggunakan manajer kluster untuk melacak sumber daya yang tersedia. Manajer kluster ini dapat berupa standalone atau dapat berintegrasi dengan manajer kluster eksternal seperti Mesos atau YARN.
- Proses driver bertanggung jawab untuk menjalankan perintah program driver di seluruh executor. Proses ini mengkoordinasikan proses distribusi dan menjalankan tugas di kluster.



SPARK API

Low-level unstructured APIs

- Memberikan akses langsung ke fungsi inti Spark seperti RDDs (Resilient Distributed Datasets).
- Memungkinkan kontrol yang lebih besar tapi memerlukan kode lebih banyak.

High-level structured APIs

- Menyediakan abstraksi tinggi dengan DataFrames dan DataSets.
- Memungkinkan penulisan kode yang lebih ringkas dan ekspresif dengan banyak fungsi pemrosesan data siap pakai.

STRUCTURED API

Structured API Apache Spark adalah kumpulan alat yang dirancang untuk memanipulasi data dalam berbagai format, mulai dari file log yang tidak terstruktur hingga file CSV yang semi-terstruktur hingga file Parquet yang sangat terstruktur.

Di dalam Spark, ada tiga jenis utama API koleksi terdistribusi yang membentuk bagian dari Structured API, yaitu:

1. Datasets
2. Dataframes
3. SQL tables and views

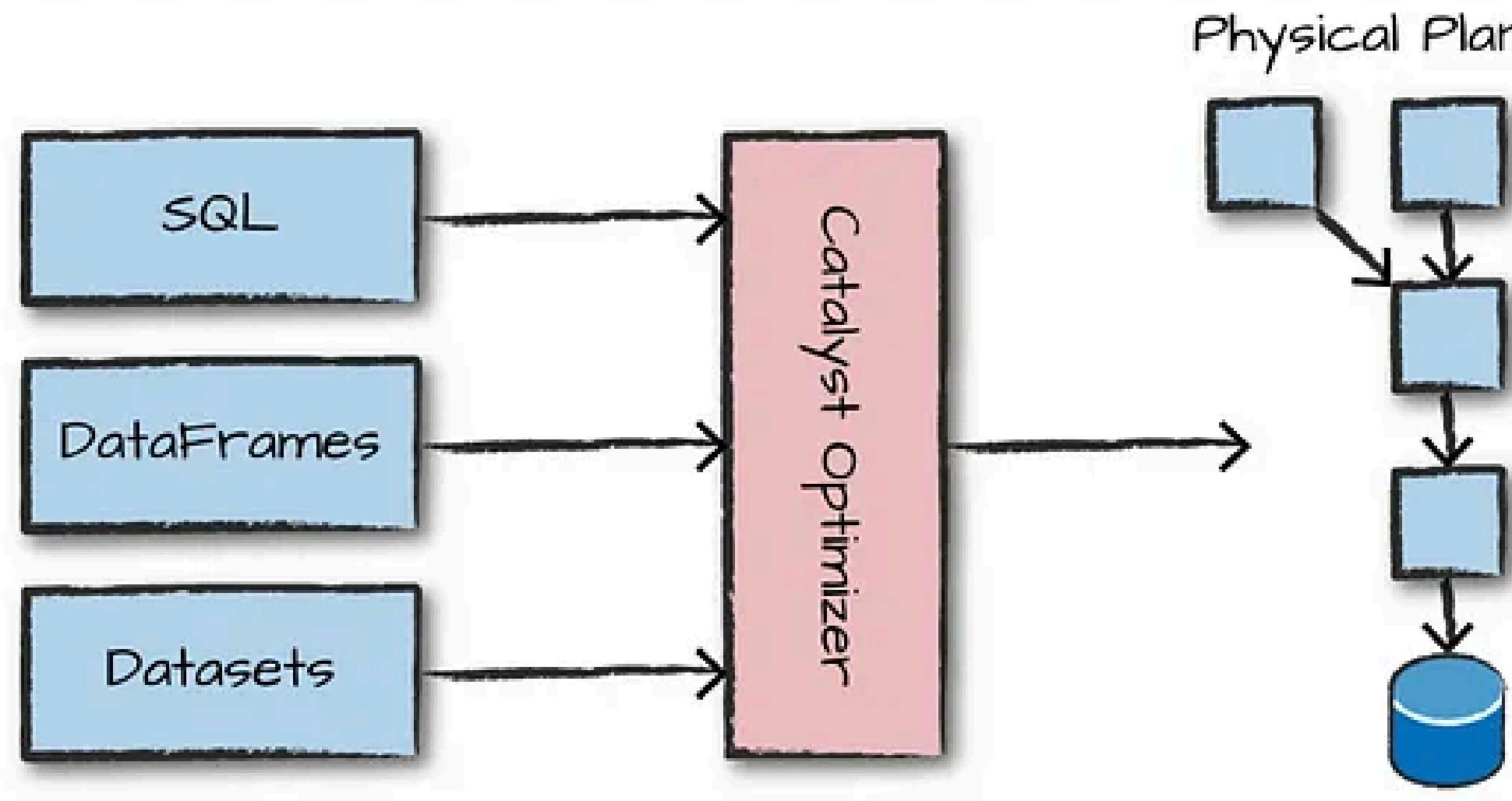
STRUCTURED API - DATASETS & DATAFRAMES

- Tipe data dari column
DataFrame fully maintained by Spark, check at runtime
- Dibalik layar, DataFrame adalah Dataset dengan type Row
- Row merupakan tipe data dari Spark core engine dan optimized in-memory format R dan Python hanya memiliki DataFrame
- **Datasets adalah tipe data terstruktur yang menyediakan tipe data terencana untuk setiap baris data.**
- Dataset menggabungkan keunggulan dari RDDs (Resilient Distributed Datasets) dan DataFrames sehingga memungkinkan kompilasi tipe statis dan optimasi kueri.
- Tipe data dari Dataset dicek oleh bahasa pemrograman at compile time dan hanya ada di JVM language

STRUCTURED API EXECUTION

Step dari sebuah structured API query dari user code to executed code:

- Menulis kode DataFrame/ Dataset/SQL
- Jika kode valid, Spark konversikan ke Logical Plan
- Spark transforms Logical Plan ke Physical Plan, checking for optimization
- Spark execute Physical Plan (RDD Manipulations) pada cluster





**TERIMA
KASIH**