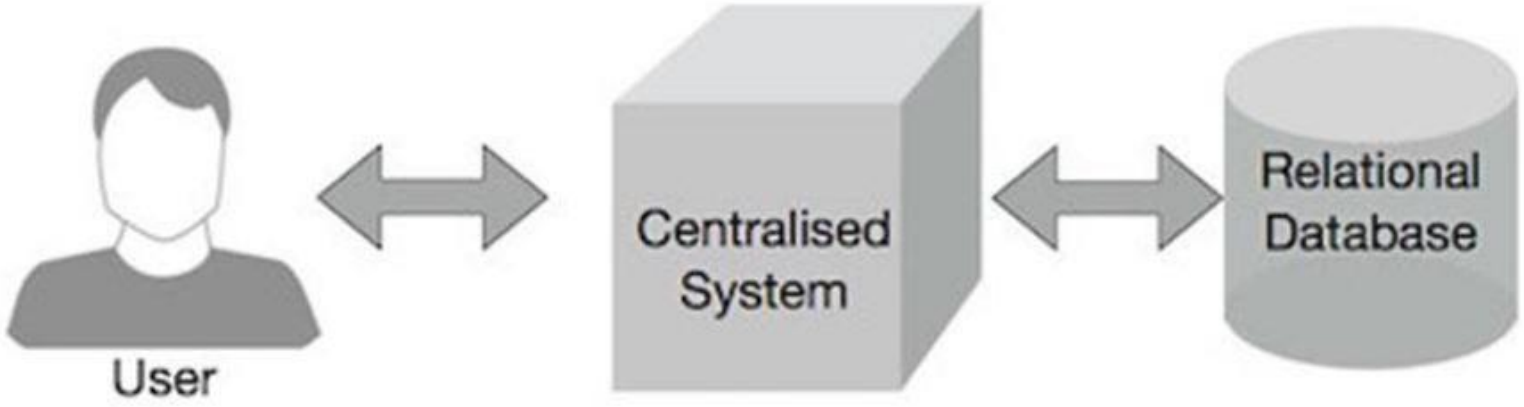
BIG DATA

Sirojul Munir | rojulman@nurulfikri.ac.id | @rojulman

Map Reduce

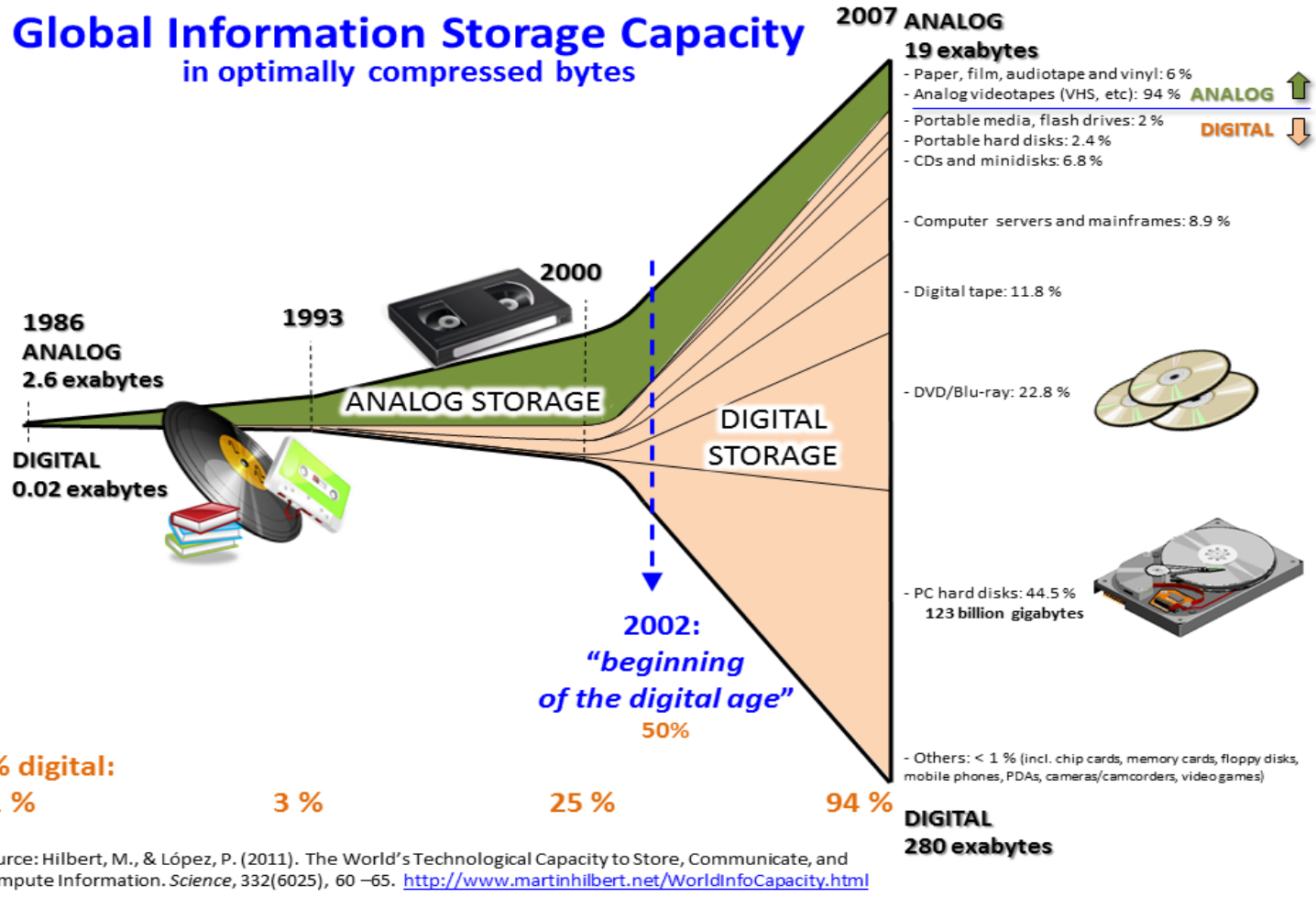
Sirojul Munir | rojulman@nurulfikri.ac.id | @rojulman

Latar Belakang



- Aplikasi tradisional melakukan proses datanya pada sistem terpusat
- Sistem terpusat tidak cocok untuk memproses data ber-skala besar karena tidak dapat di akomodir oleh sistem standard database (RDBMS)
- Sistem terpusat memiliki kelemahan (adanya bottleneck) ketika memproses multiple file secara simultan

Latar Belakang : Growing Data

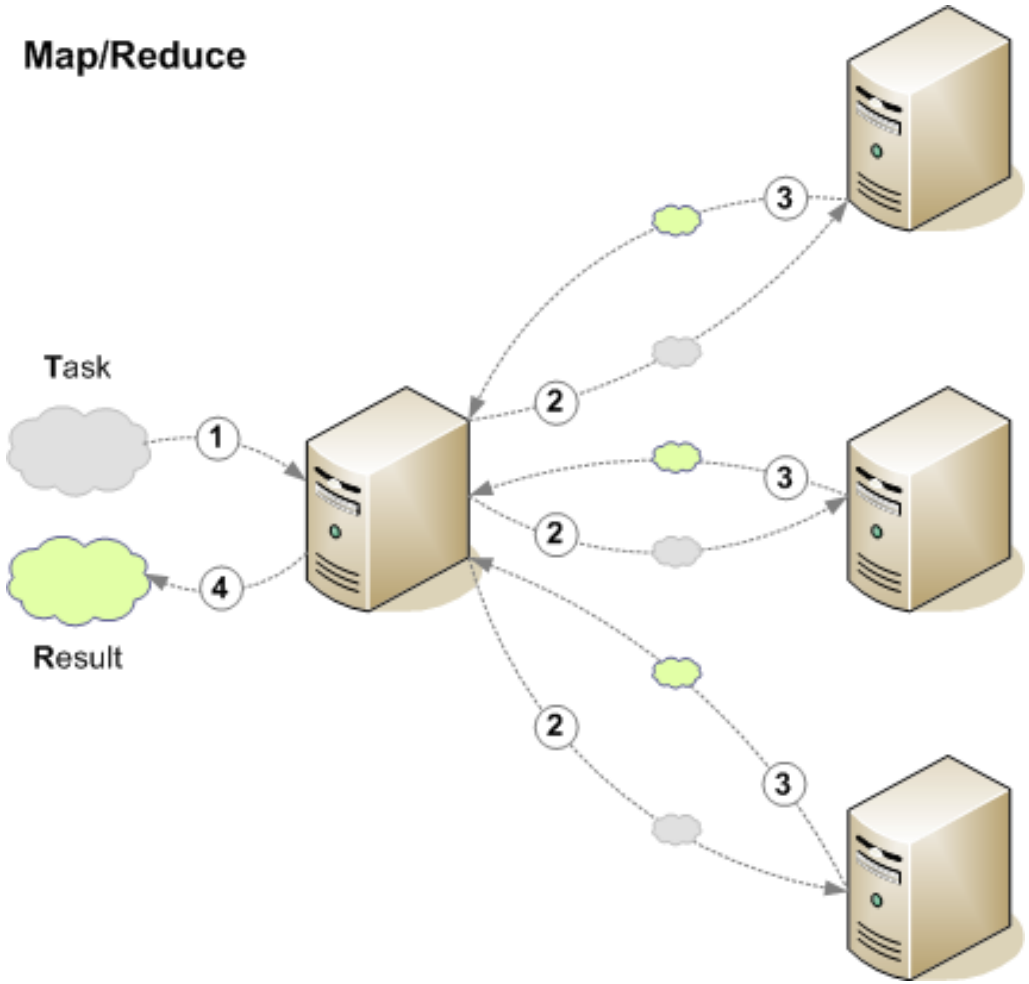


Source: Hilbert, M., & López, P. (2011). The World's Technological Capacity to Store, Communicate, and Compute Information. *Science*, 332(6025), 60 –65. http://www.martinhilbert.net/WorldInfoCapacity.html

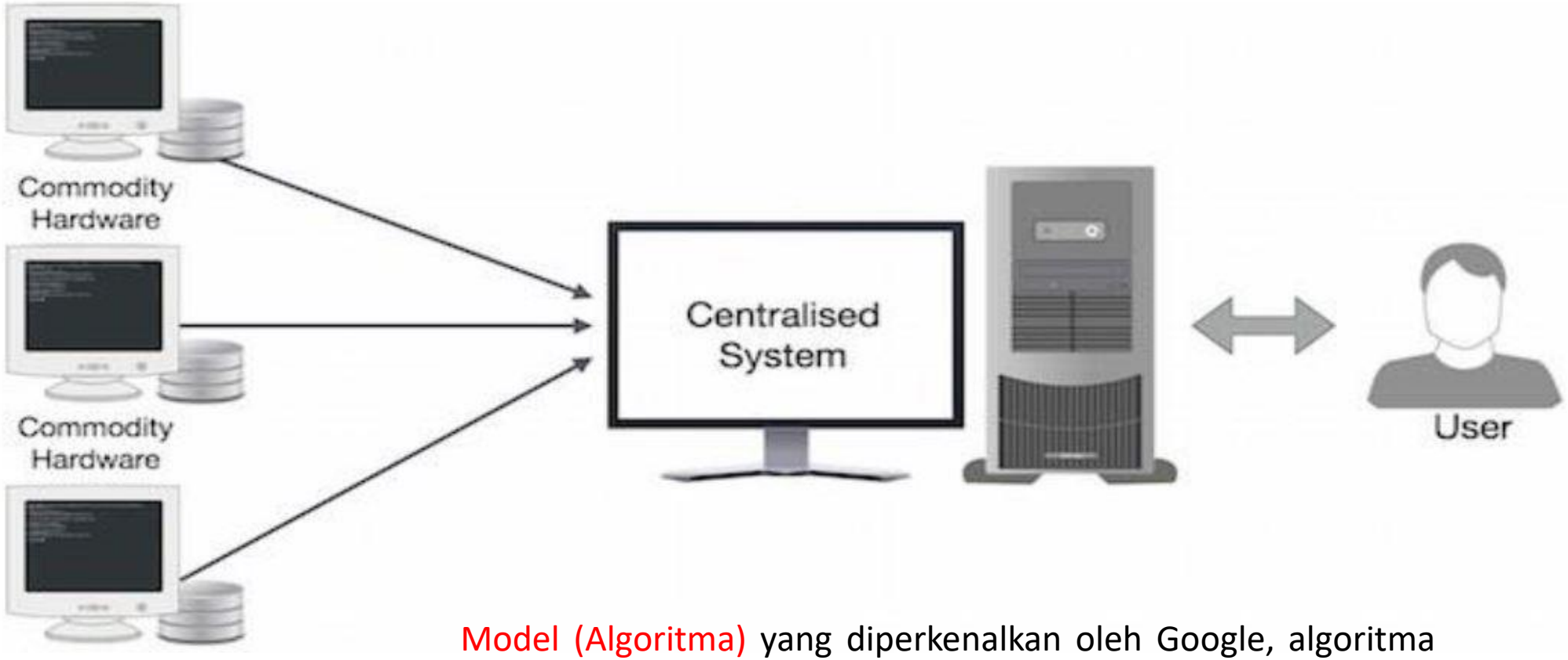
- Tahun 2009, Google memproses data per hari mencapai **24 Petabytes**
- Satu mesin server tidak dapat menangani seluruh proses data, karenanya dibutuhkan **system yang terdistribusi** dan **proses data yang paralel**

Apa itu Map Reduce ?

- Model (Algoritma) yang diperkenalkan oleh Google, algoritma membagi pekerjaan (task) menjadi beberapa bagian pekerjaan dan dikerjakan oleh beberapa komputer
- Algoritma Map Reduce memecahkan masalah bottleneck yang terjadi pada pemrosesan data
- Proses data berskala besar:
  - Membutuhkan perangkat computer spek besar/tinggi
  - Eksekusi proses data secara terdistribusi
  - Menawarkan tingkat ketersediaan yang tinggi (high avaibility)

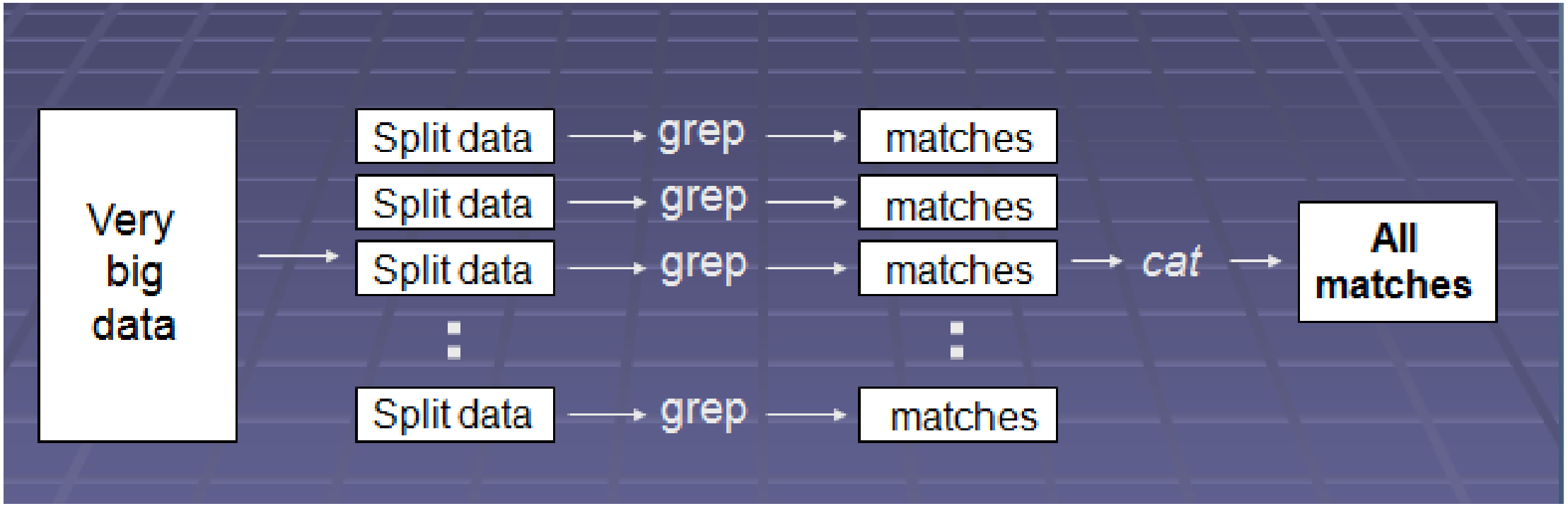


Algoritma Mapreduce

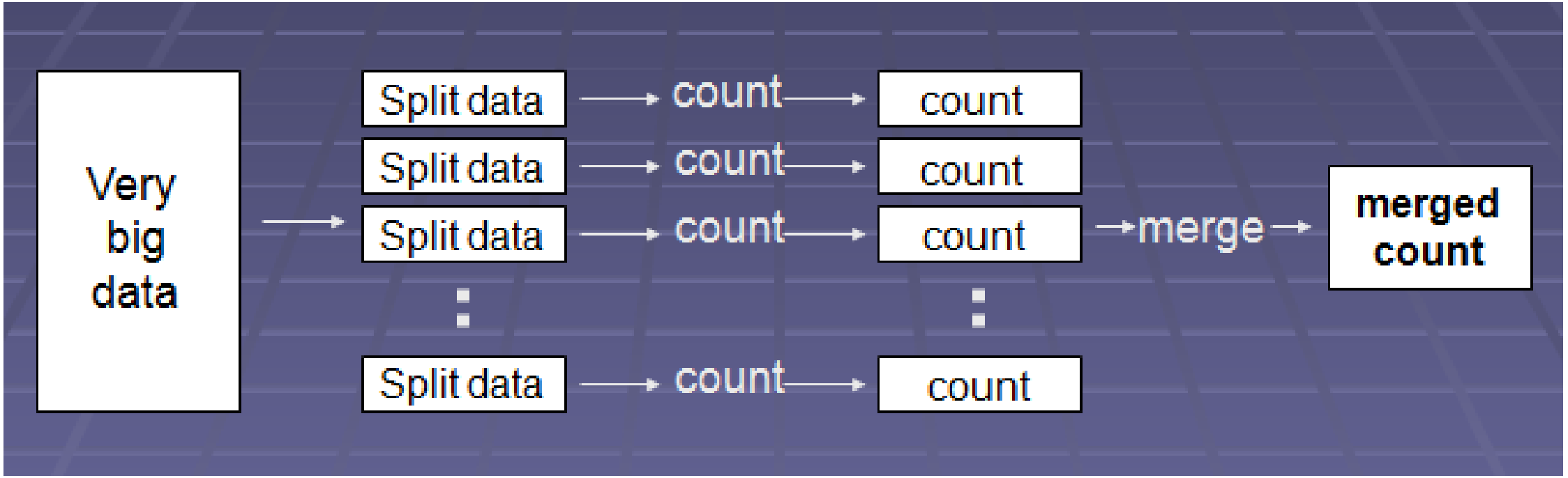


Model (Algoritma) yang diperkenalkan oleh Google, algoritma membagi pekerjaan (task) menjadi beberapa bagian pekerjaan dan dikerjakan oleh beberapa komputer

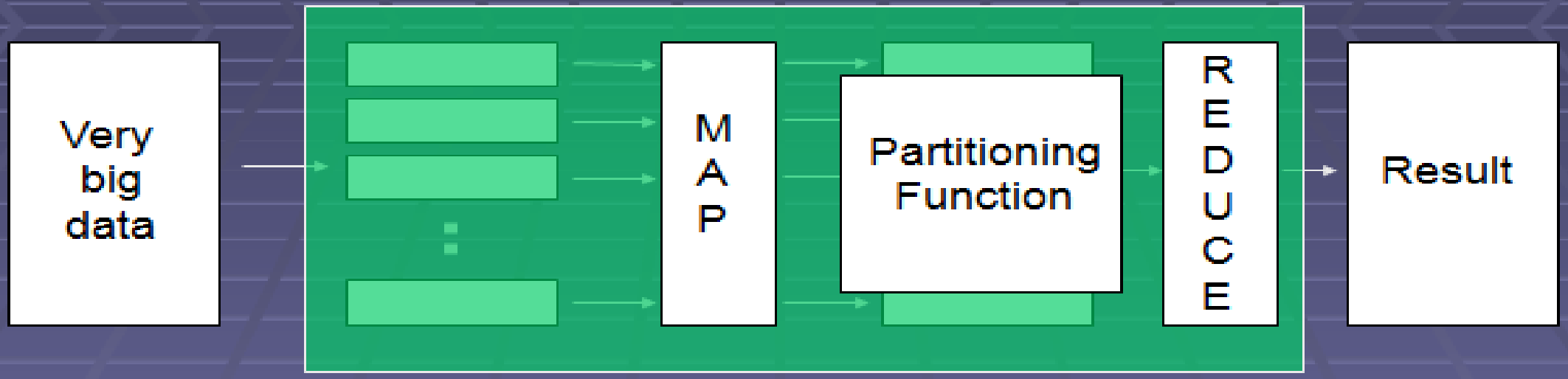
Contoh : Proses data - Distributed Grep



Contoh : Proses data - Distributed Word Count

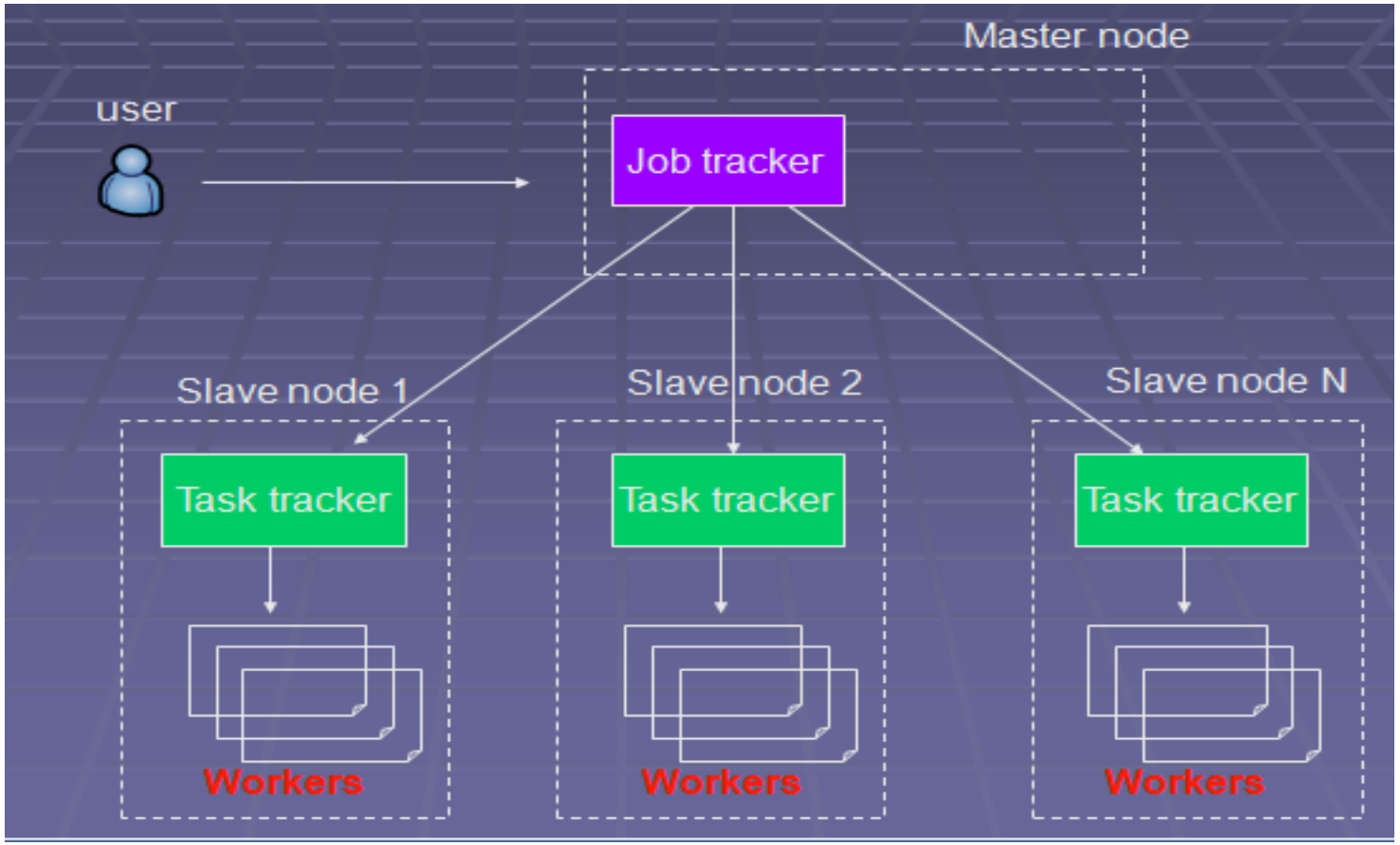


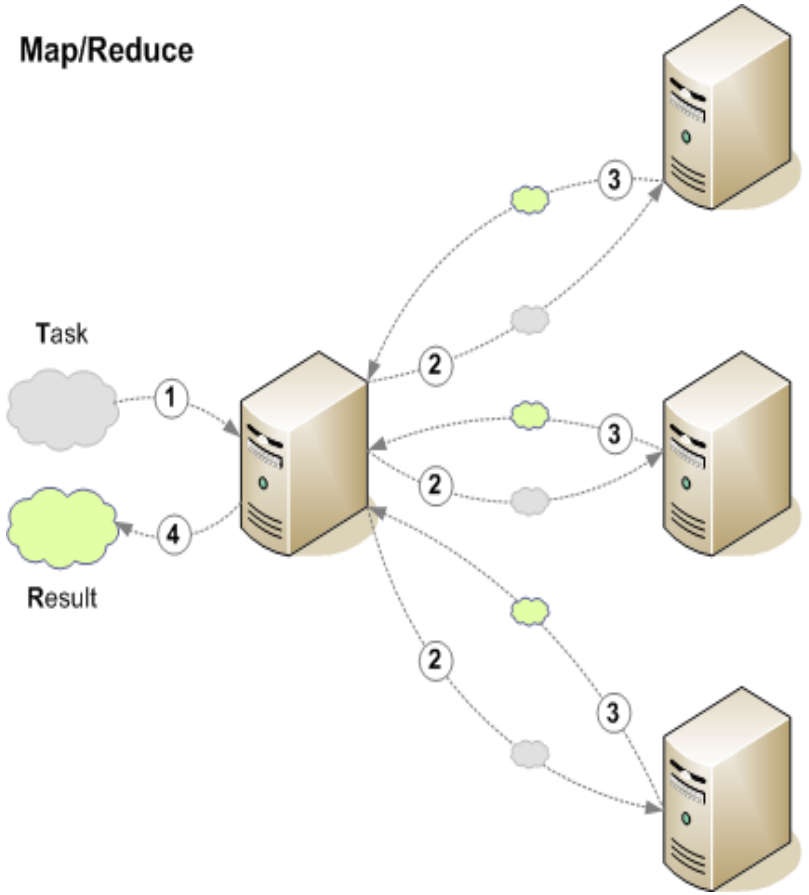
Map + Reduce



- Map:
  - Accepts *input* key/value pair
  - Emits *intermediate* key/value pair
- Reduce :
  - Accepts *intermediate* key/value* pair
  - Emits *output* key/value pair

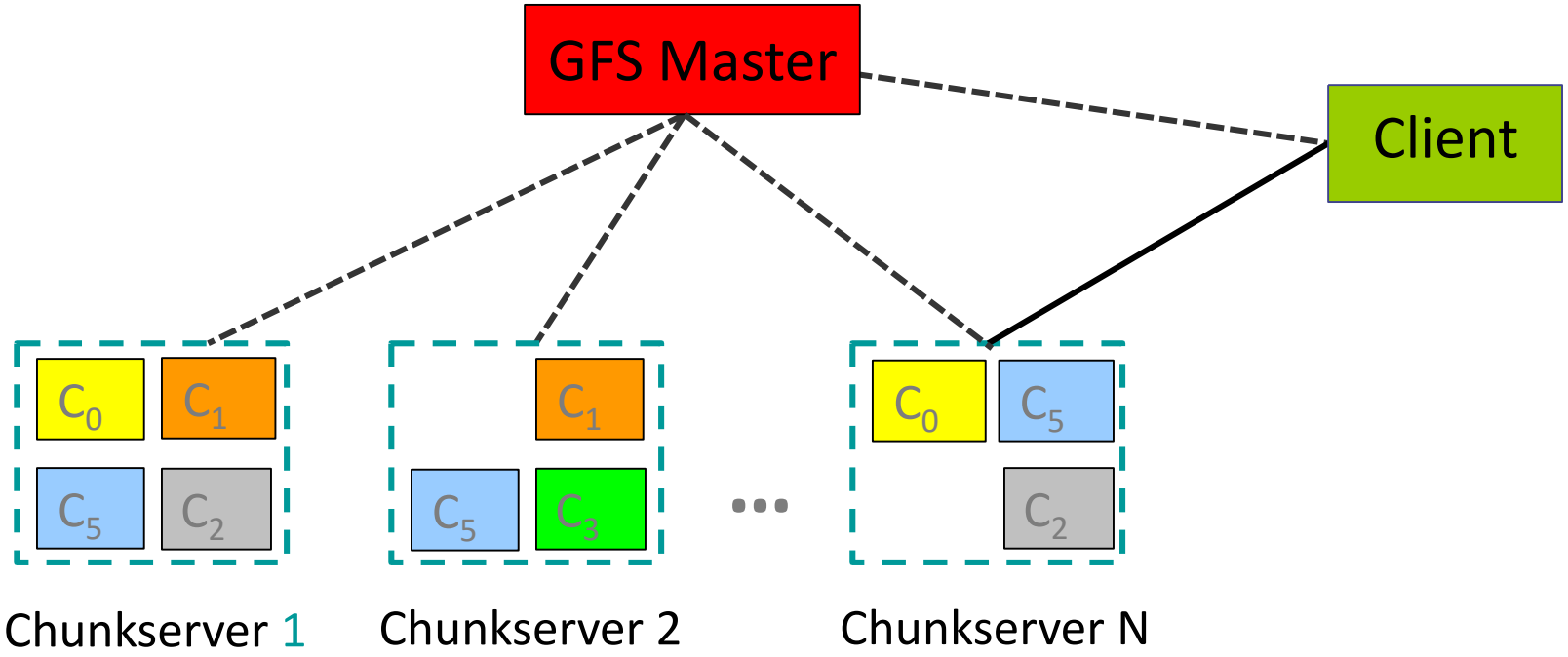
Design System : Arsitektur view





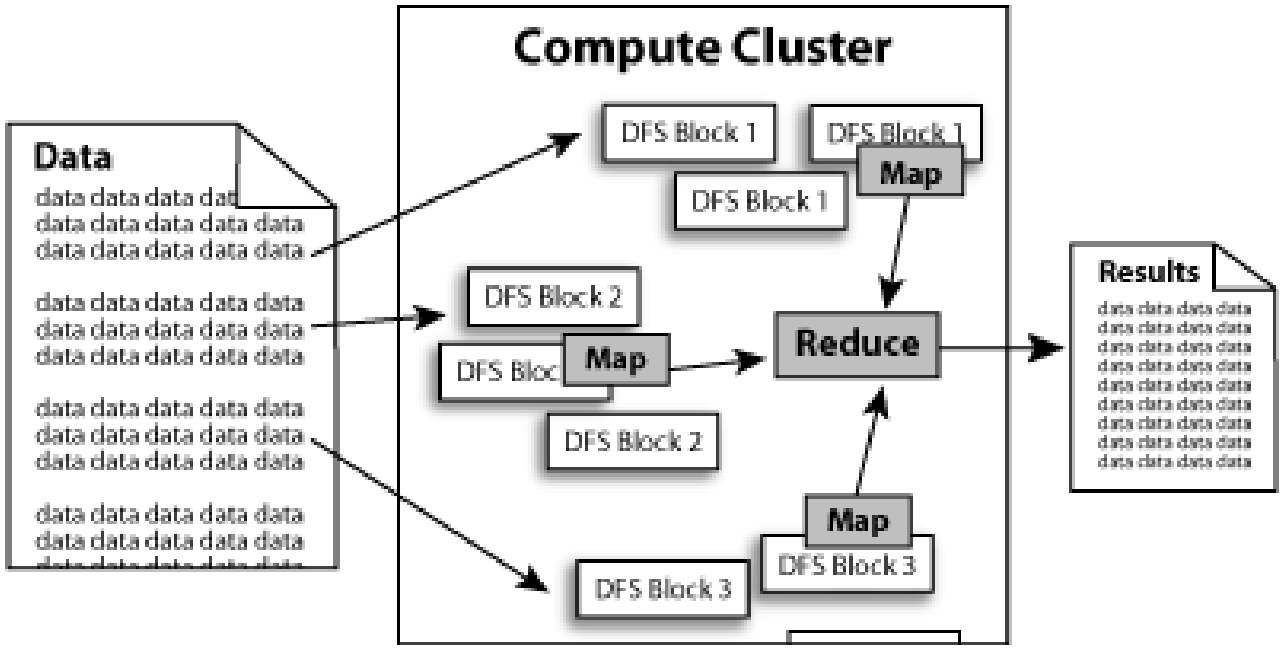
Google File System (GFS)

- Goal
  - global view
  - make huge files available in the face of node failures
- Master Node (meta server)
  - Centralized, index all chunks on data servers
- Chunk server (data server)
  - File is split into contiguous chunks, typically 16-64MB.
  - Each chunk replicated (usually 2x or 3x).
    - Try to keep replicas in different racks.



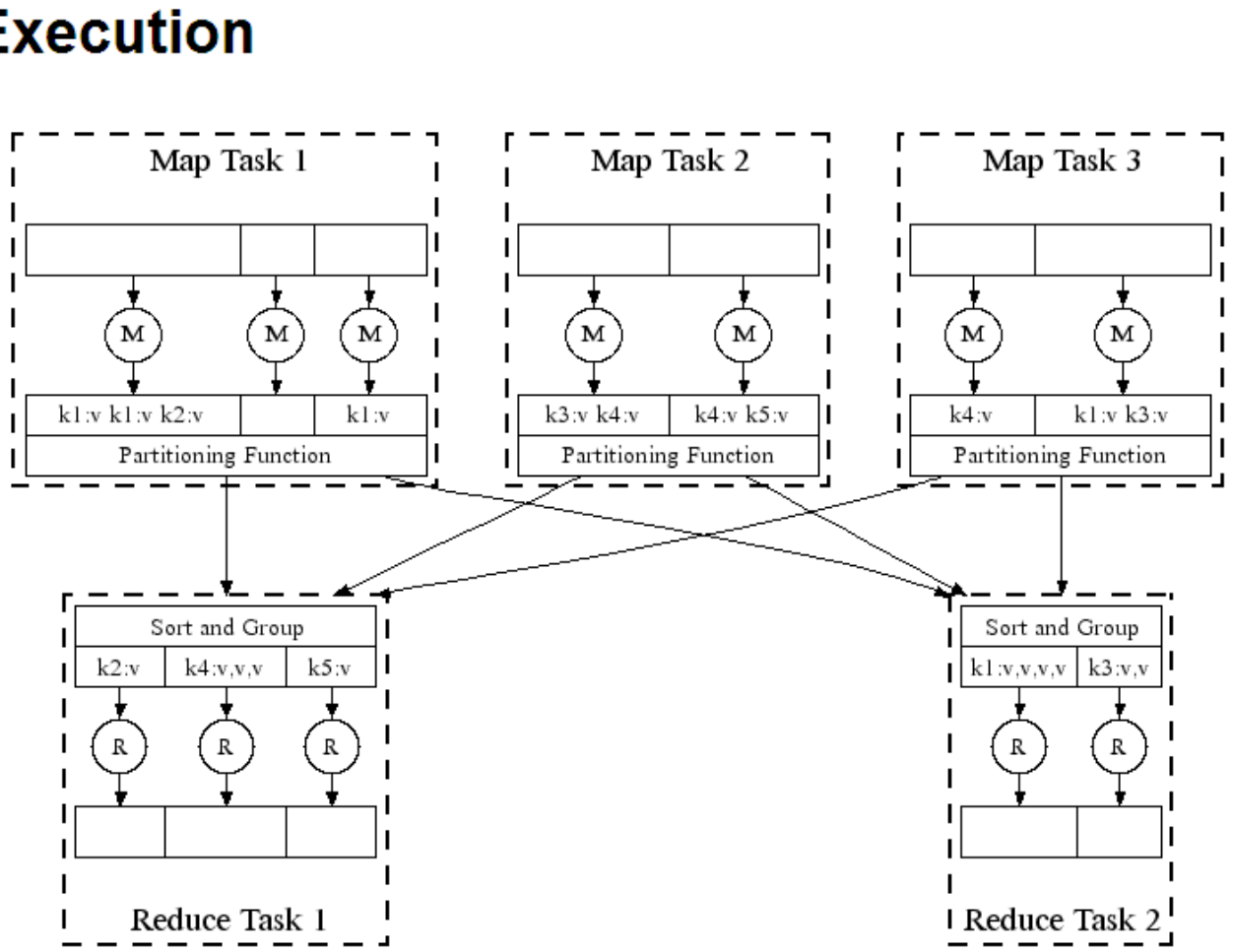
Model : Map Reduce

- Map
  - Process a key/value pair to generate intermediate key/value pairs
- Reduce
  - Merge all intermediate values associated with the same key
- Partition
  - By default : `hash(key) mod R`
  - Well balanced



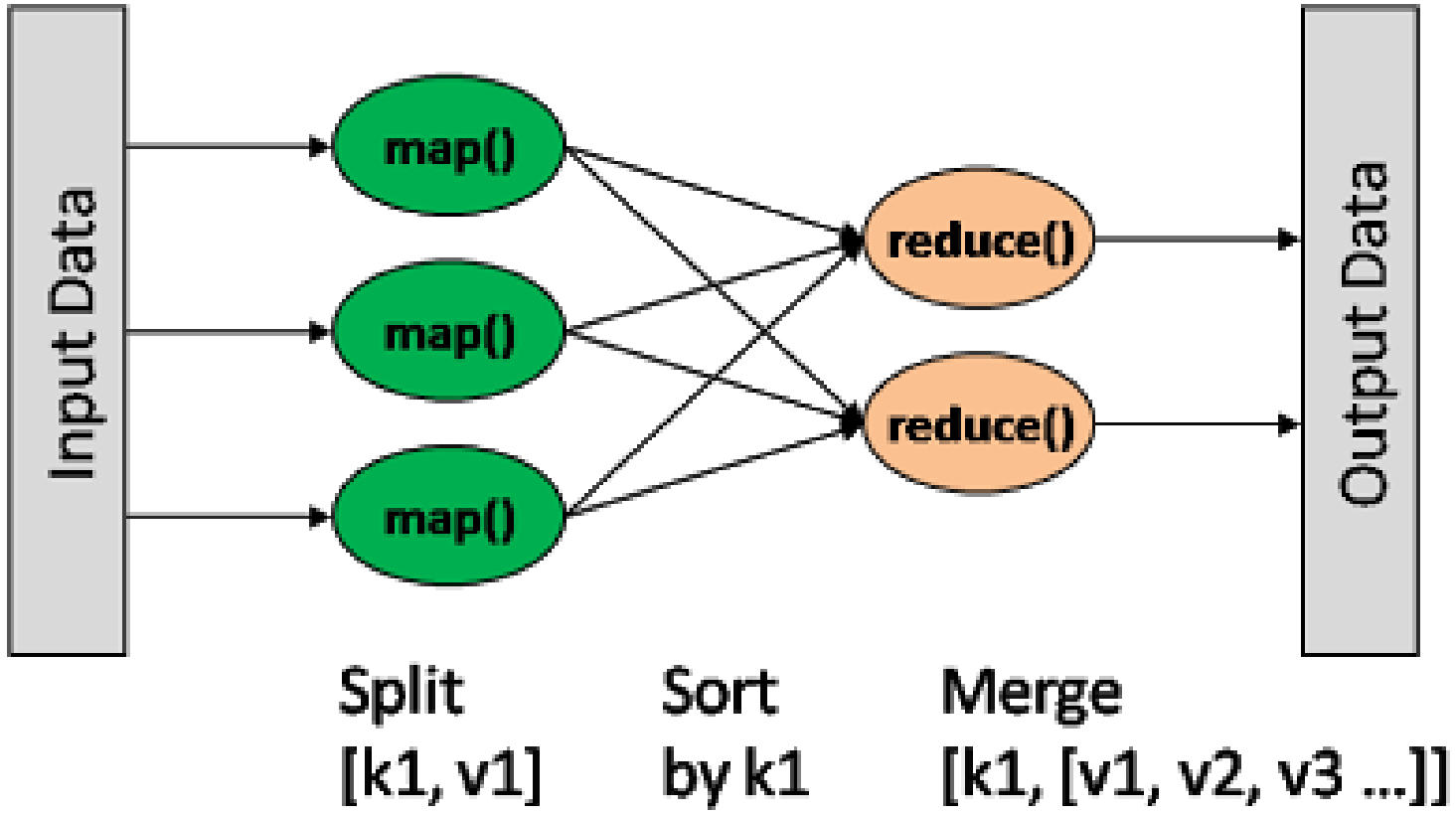
Parallel Execution

- Map
  - Process a key/value pair to generate intermediate key/value pairs
- Reduce
  - Merge all intermediate values associated with the same key
- Partition
  - By default : `hash(key) mod R`
  - Well balanced



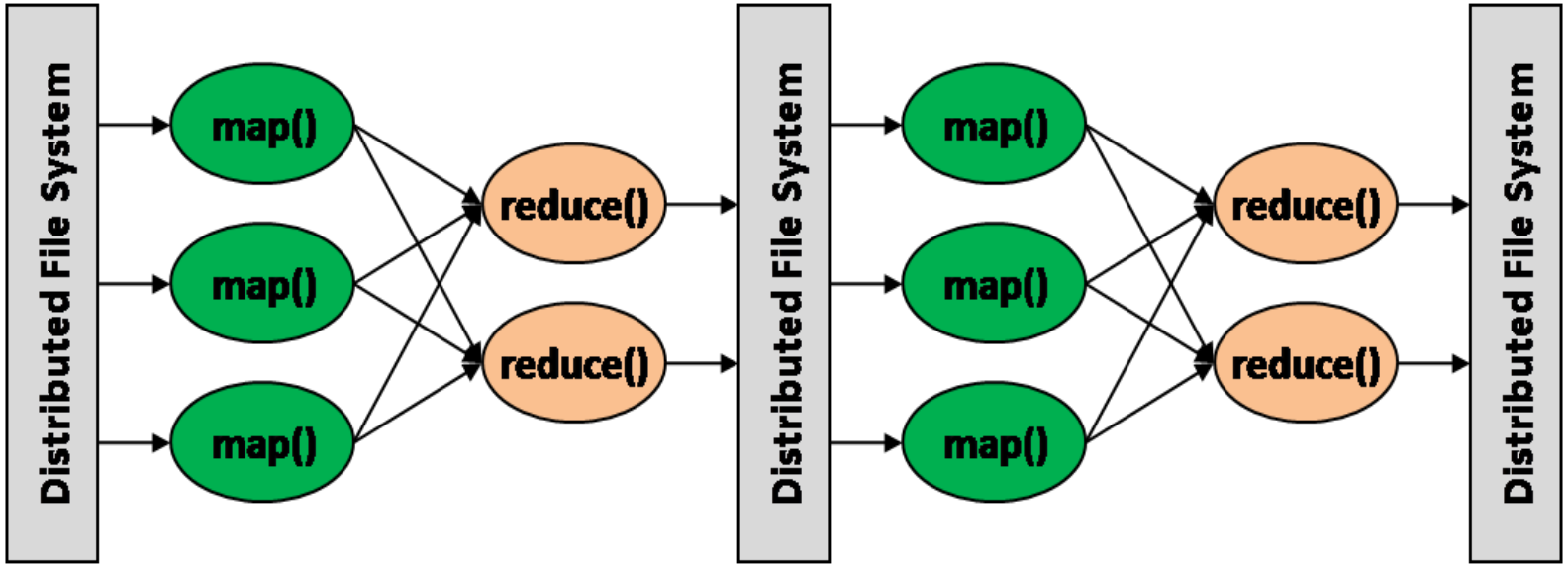
Bagaimana Mapreduce bekerja ?

- Map – Membagi himpunan data menjadi beberapa himpunan data yang elemennya terdiri atas elemen key berpasangan dengan elemen value
- Reduce – Mengambil output dari map sebagai inputan dan melakukan kombinasi dari data tuple(key-value) menjadi bagian-bagian kecil data tuple



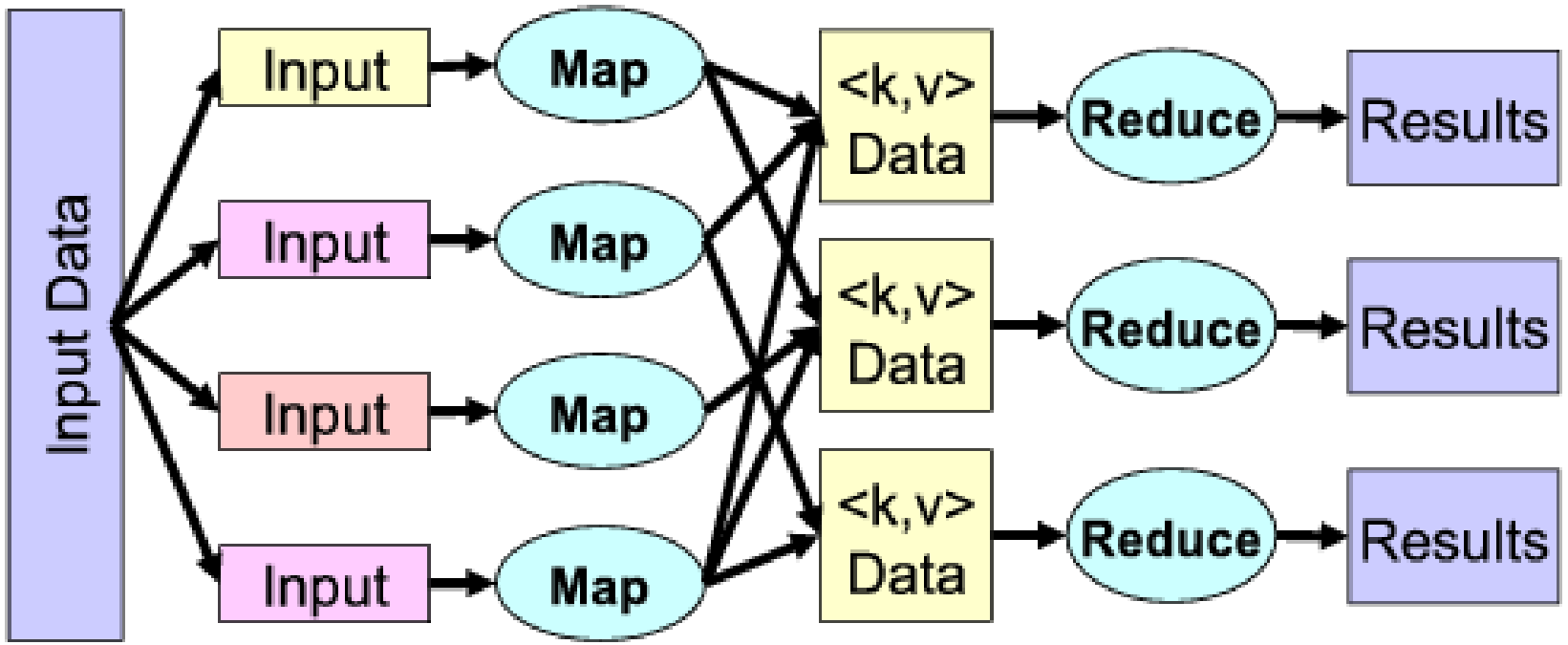
Bagaimana Mapreduce bekerja ?

- Map – Membagi himpunan data menjadi beberapa himpunan data yang elemennya terdiri atas elemen key berpasangan dengan elemen value
- Reduce – Mengambil output dari map sebagai inputan dan melakukan kombinasi dari data tuple(key-value) menjadi bagian-bagian kecil data tuple

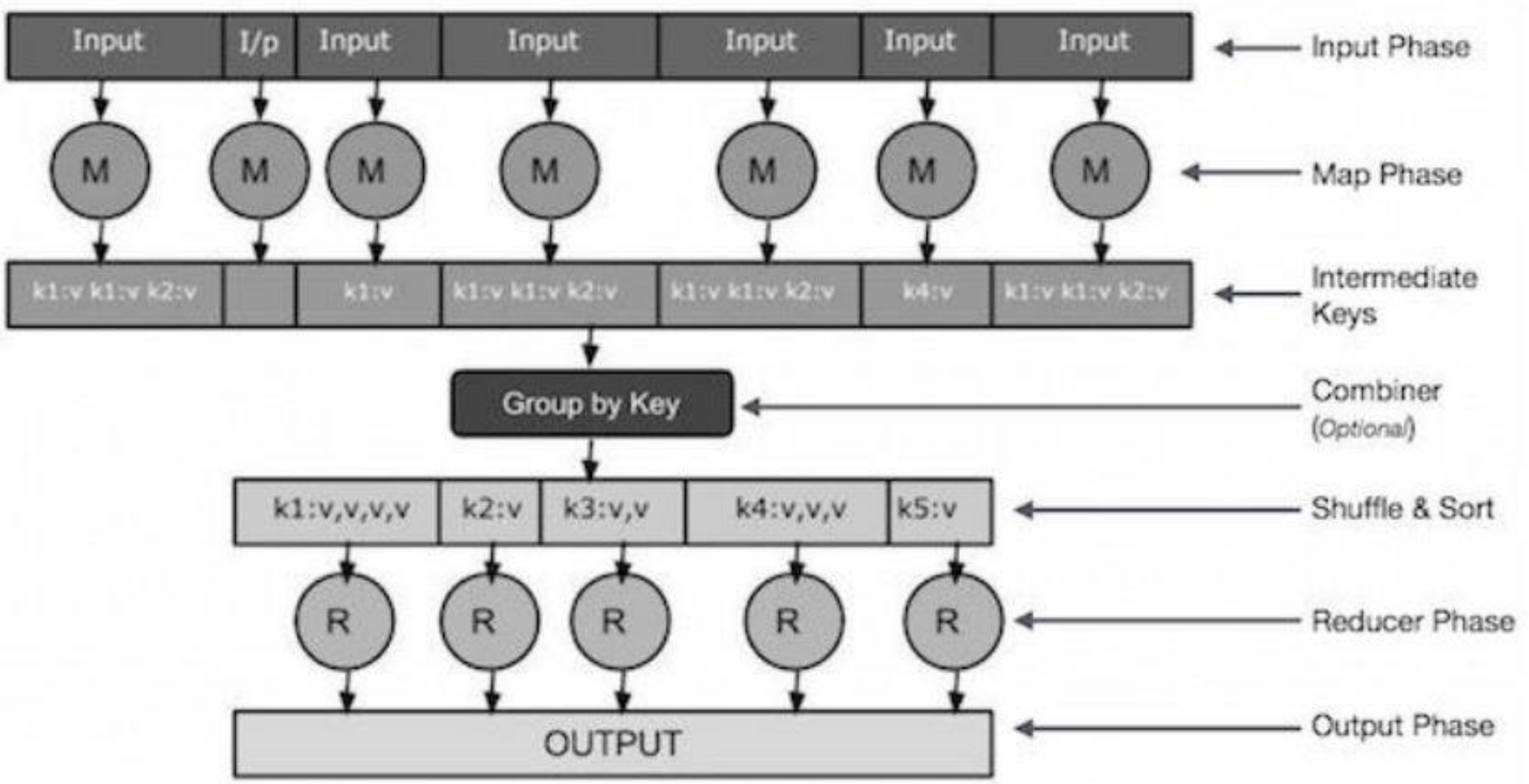


Map - Reduce

	Input	Output
Map	<k1, v1>	list (<k2, v2>)
Reduce	<k2, list(v2)>	list (<k3, v3>)



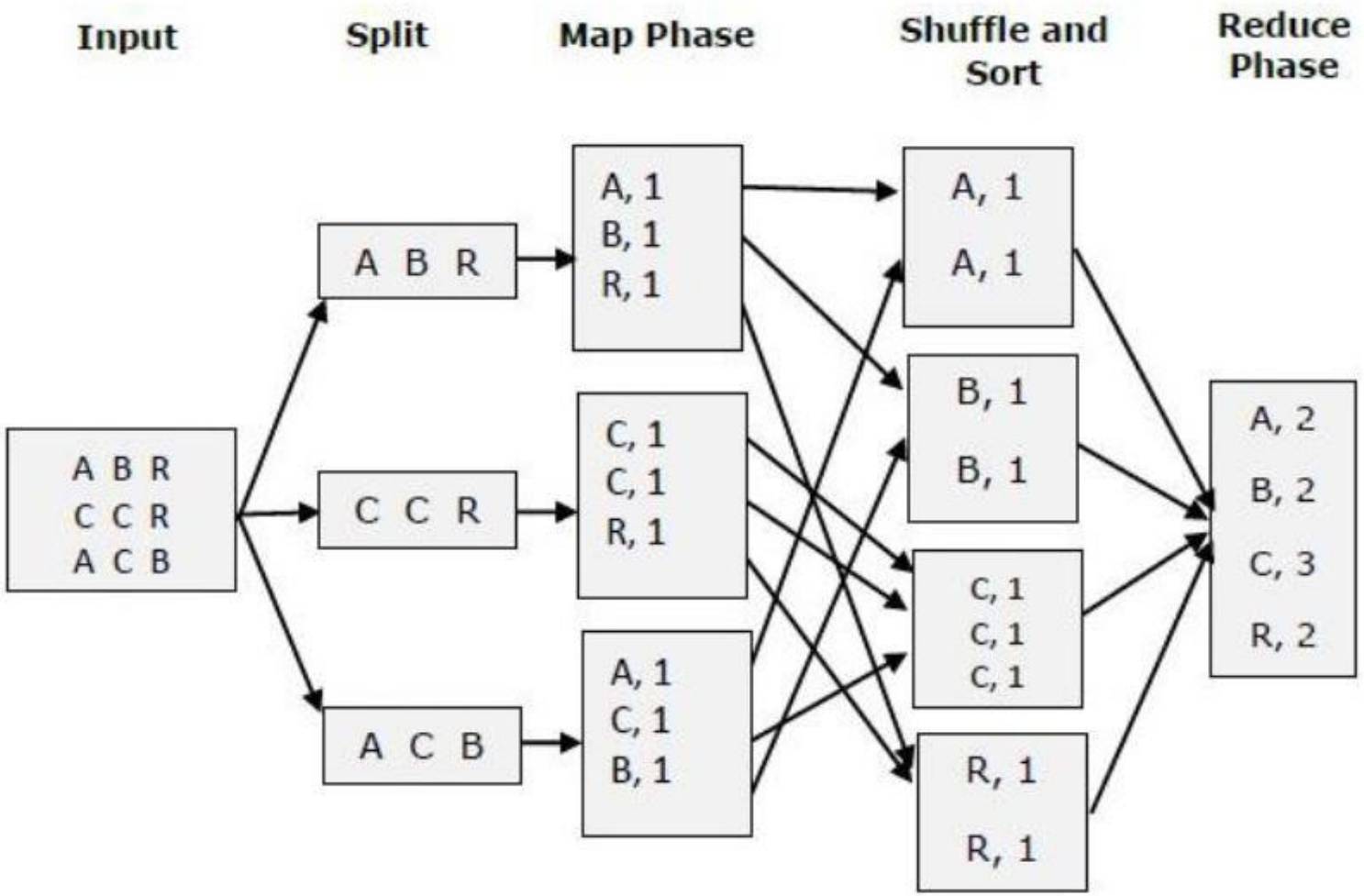
Fase proses MapReduce



Phase MapReduce

- Input Phase
- Map
- Intermediate Keys
- Combiner
- Shuffle and Sort
- Reducer
- Output Phase

Ilustrasi proses MapReduce

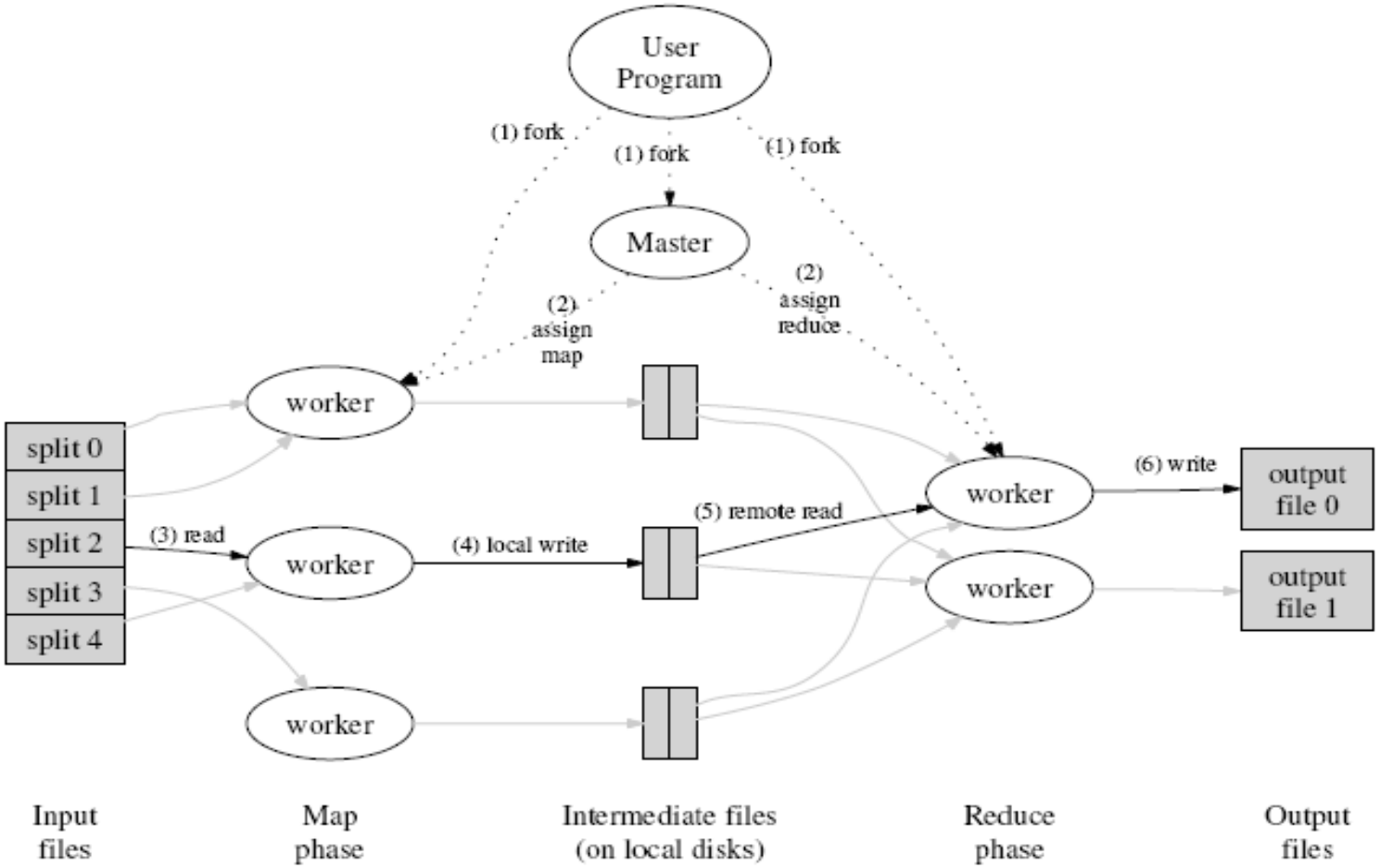


Contoh sederhana lain : pseudo code

- **Counting words in a large set of documents**

```
map(string value)
        //key: document name
        //value: document contents
        for each word w in value
                EmitIntermediate(w, "1");

reduce(string key, iterator values)
        //key: word
        //values: list of counts
        int results = 0;
        for each v in values
                result += ParseInt(v);
        Emit(AsString(result));
```

Proses Word Count



Locality issue

- Master scheduling policy
  - Asks GFS for locations of replicas of input file blocks
  - Map tasks typically split into 64MB (== GFS block size)
  - Map tasks scheduled so GFS input block replica are on same machine or same rack
- Effect
  - Thousands of machines read input at local disk speed
  - Without this, rack switches limit read rate

Fault Tolerance

- Reactive way
  - Worker failure
    - Heartbeat, Workers are periodically pinged by master
      - NO response = failed worker
    - If the processor of a worker fails, the tasks of that worker are reassigned to another worker.
  - Master failure
    - Master writes periodic checkpoints
    - Another master can be started from the last checkpointed state
    - If eventually the master dies, the job
- Input error: bad records
  - Map/Reduce functions sometimes fail for particular inputs
  - Best solution is to debug & fix, but not always possible
  - On segment fault
    - Send UDP packet to master from signal handler
    - Include sequence number of record being processed
  - Skip bad records
    - If master sees two failures for same record, next worker is told to skip the record

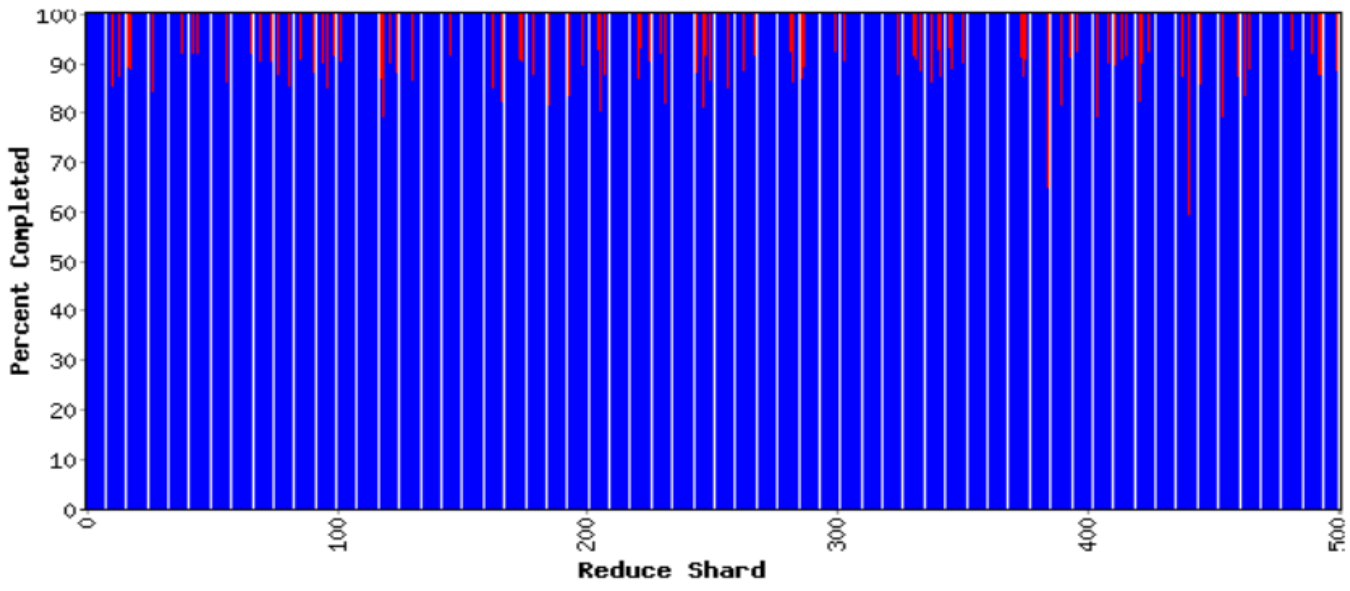
Lab research Map Reduce

MapReduce status: MR_Indexer-beta6-large-2003_10_28_00_03

Started: Fri Nov 7 09:51:07 2003 -- up 0 hr 37 min 01 sec

1707 workers; 1 deaths

Type	Shards	Done	Active	Input(MB)	Done(MB)	Output(MB)
Map	13853	13853	0	878934.6	878934.6	523499.2
Shuffle	500	500	0	523499.2	520468.6	520468.6
Reduce	500	406	94	520468.6	512265.2	514373.3



Counters

Variable	Minute
Mapped (MB/s)	0.0
Shuffle (MB/s)	0.0
Output (MB/s)	849.5
doc-index-hits	0
docs-indexed	0
dups-in-index-merge	0
mr-merge-calls	35083350
mr-merge-outputs	35083350

Lab Research : Perbaikan2

Refinement

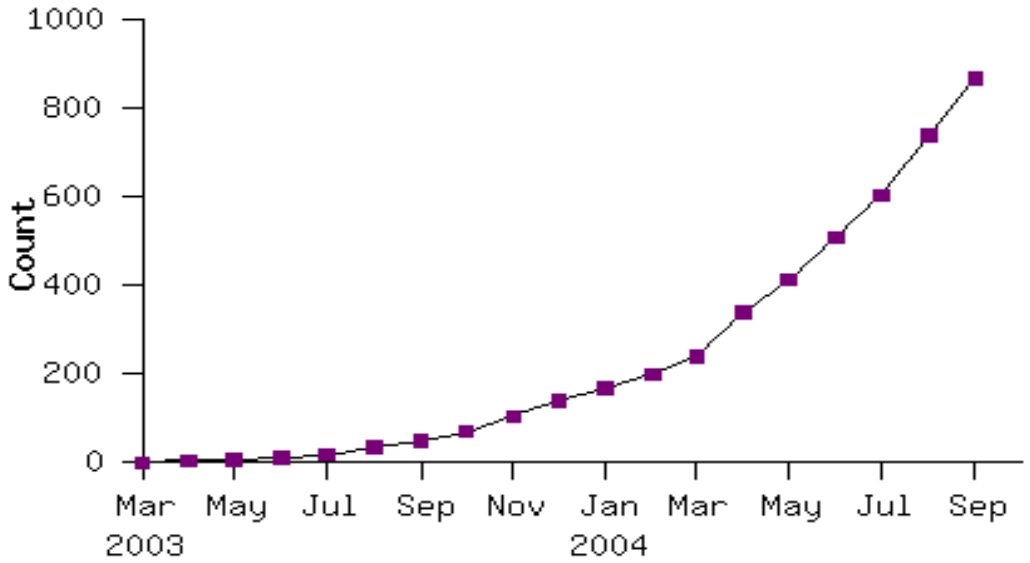
- Task Granularity
  - Minimizes time for fault recovery
  - load balancing
- Local execution for debugging/testing
- Compression of intermediate data

Notes to emphasized

- No *reduce* can begin until *map* is complete
- Master must communicate locations of intermediate files
- Tasks scheduled based on location of data
- If *map* worker fails any time before *reduce* finishes, task must be completely rerun
- MapReduce library does most of the hard work for us!

Model Map Reduce : implementasi kasus

- **MapReduce Programs In Google Source Tree**



Contoh implementasi dalam beberapa kasus lain:

distributed grep	distributed sort	web link-graph reversal
term-vector / host	web access log stats	inverted index construction
document clustering	machine learning	statistical machine translation
…	…	…

Contoh program word count (1)

- Map :

```
#include "mapreduce/mapreduce.h"

// User's map function
class WordCounter : public Mapper {
 public:
  virtual void Map(const MapInput& input) {
    const string& text = input.value();
    const int n = text.size();
    for (int i = 0; i < n; ) {
      // Skip past leading whitespace
      while ((i < n) && isspace(text[i]))
        i++;

      // Find word end
      int start = i;
      while ((i < n) && !isspace(text[i]))
        i++;

     if (start < i)
       Emit(text.substr(start,i-start),"1");
    }
  }
};
REGISTER_MAPPER(WordCounter);
```

Contoh program word count (2)

- Reduce :

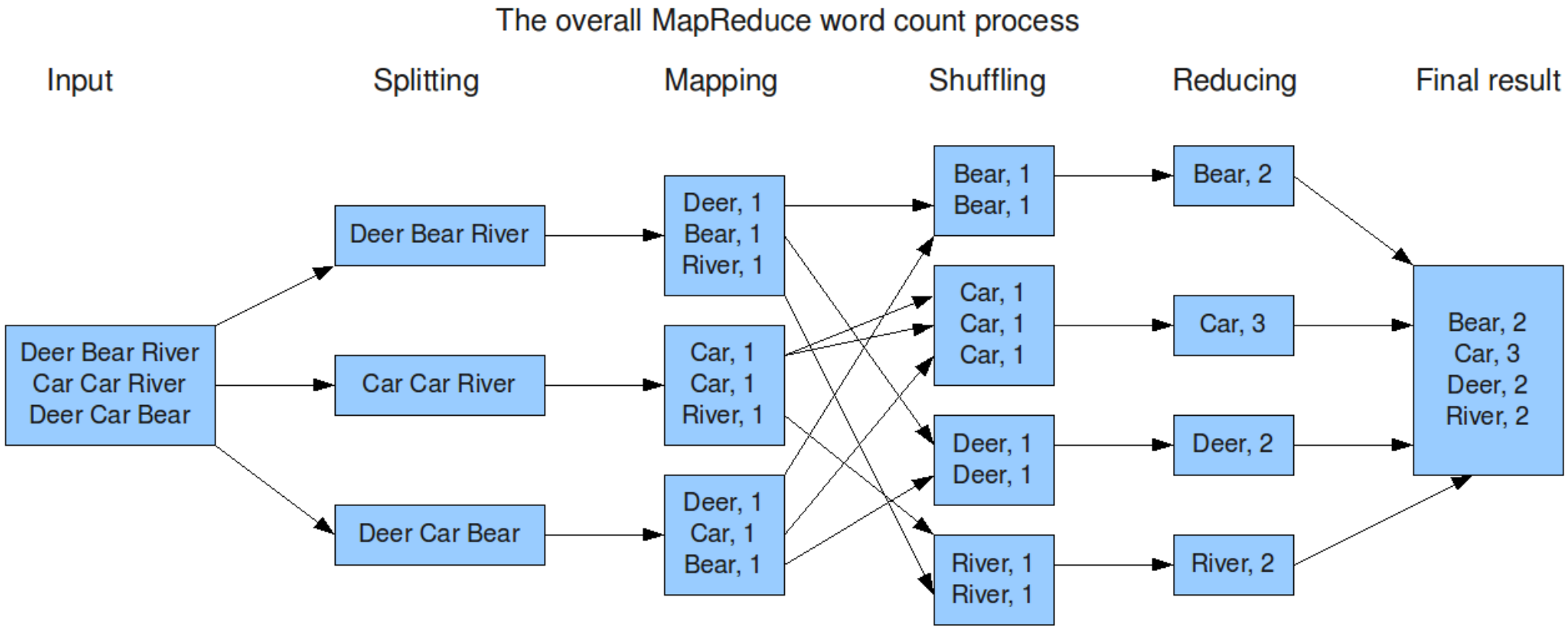
```
// User's reduce function
class Adder : public Reducer {
  virtual void Reduce(ReduceInput* input) {
    // Iterate over all entries with the
    // same key and add the values
    int64 value = 0;
    while (!input->done()) {
      value += StringToInt(input->value());
      input->NextValue();
    }

    // Emit sum for input->key()
    Emit(IntToString(value));
  }
};
REGISTER_REDUCER(Adder);
```

Contoh program word count (3)

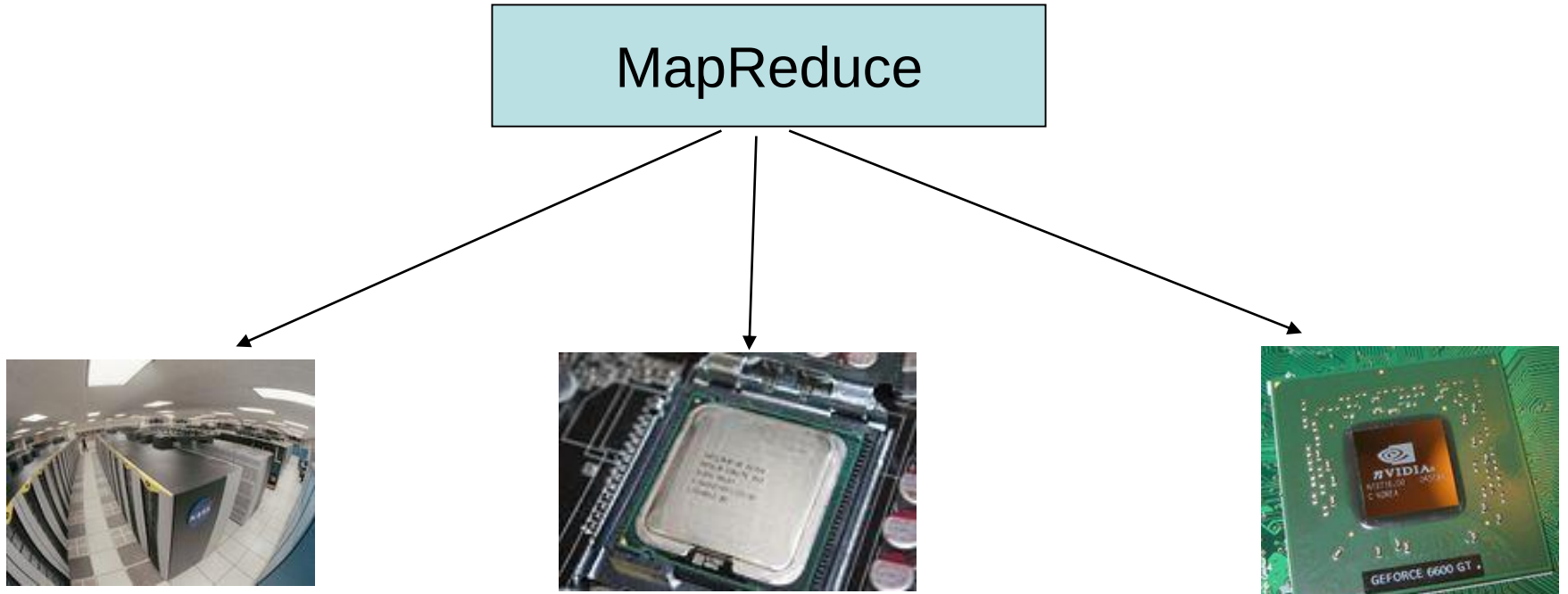
- Main :

```
int main(int argc, char** argv) {
  ParseCommandLineFlags(argc, argv);

  MapReduceSpecification spec;

  // Store list of input files into "spec"
  for (int i = 1; i < argc; i++) {
    MapReduceInput* input = spec.add_input();
    input->set_format("text");
    input->set_filepattern(argv[i]);
    input->set_mapper_class("WordCounter");
  }

  // Specify the output files:
  //    /gfs/test/freq-00000-of-00100
  //    /gfs/test/freq-00001-of-00100
  //    ...
  MapReduceOutput* out = spec.output();
  out->set_filebase("/gfs/test/freq");
  out->set_num_tasks(100);
  out->set_format("text");
  out->set_reducer_class("Adder");

  // Optional: do partial sums within map
  // tasks to save network bandwidth
  out->set_combiner_class("Adder");

  // Tuning parameters: use at most 2000
  // machines and 100 MB of memory per task
  spec.set_machines(2000);
  spec.set_map_megabytes(100);
  spec.set_reduce_megabytes(100);

  // Now run it
  MapReduceResult result;
  if (!MapReduce(spec, &result)) abort();

  // Done: 'result' structure contains info
  // about counters, time taken, number of
  // machines used, etc.

  return 0;
}
```

Map Reduce Process : Word Count

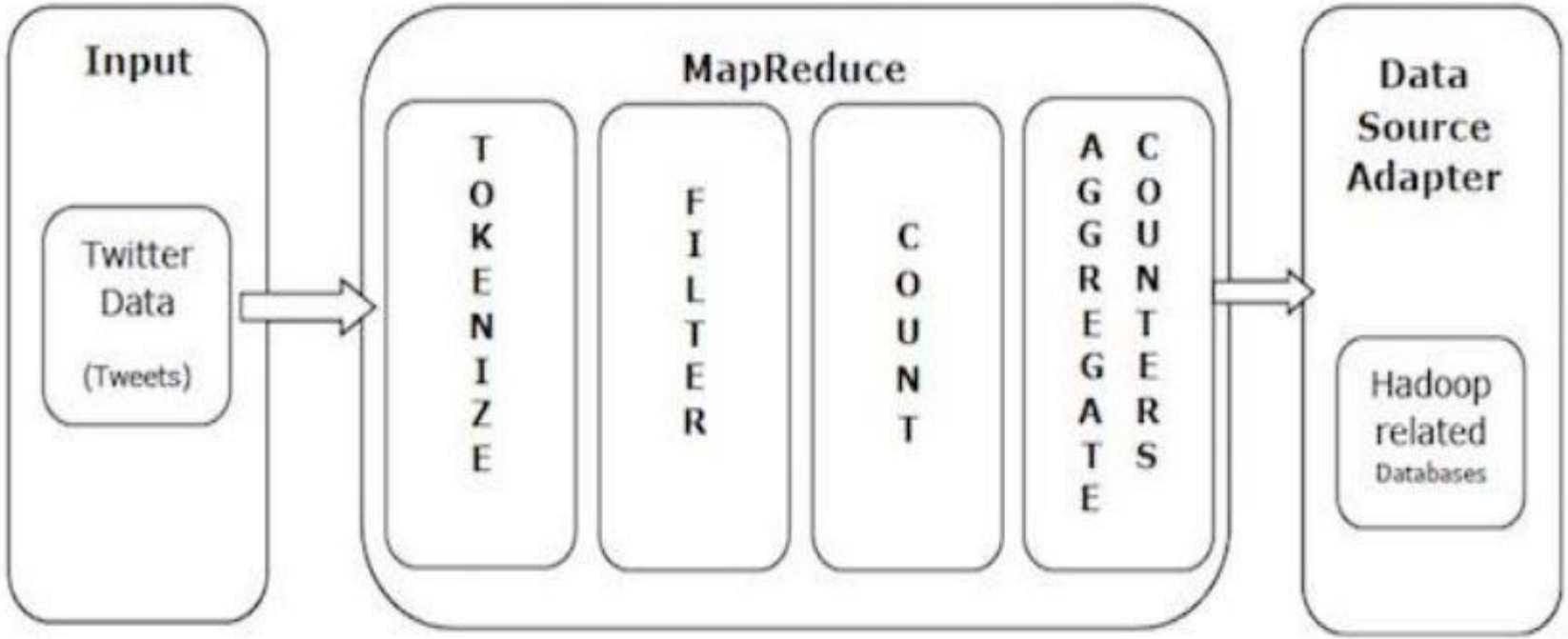


Implementasi Map Reduce



Contoh : MapReduce Twitter

- Twitter menerima data setiap hari sekitar 500 juta tweet , 3000 tweet per detik



Next Hadoop !

Google	Yahoo / Apache
MapReduce	Hadoop
GFS	HDFS
Bigtable	HBase
Chubby (distributed lock service)	Apache Zookeeper

Kuis !!

- Jelaskan apa Map Reduce ?
- Mengapa Map Reduce dibutuhkan untuk proses data skala besar ?