Introduction to MapReduce

Concept and Programming

趨勢科技研發實驗室

Public 2009/5/13

TREND MICRO

Outline

- Overview and Brief History
- MapReduce Concept
- Introduction to MapReduce programming using Hadoop
- Reference

Copyright 2009 - Trend Micro Inc.

Brief history of MapReduce

- A demand for **large scale data processing** in Google
- The folks at Google discovered certain **common themes** for processing those large input sizes
  - Many machines are needed
  - Two basic operations on the input data: **Map and Reduce**
- Google's idea is inspired by ***map*** and ***reduce*** functions commonly used in **functional programming**.
- Functional programming: MapReduce

– Map
  - [1,2,3,4] – (*2) → [2,3,6,8]

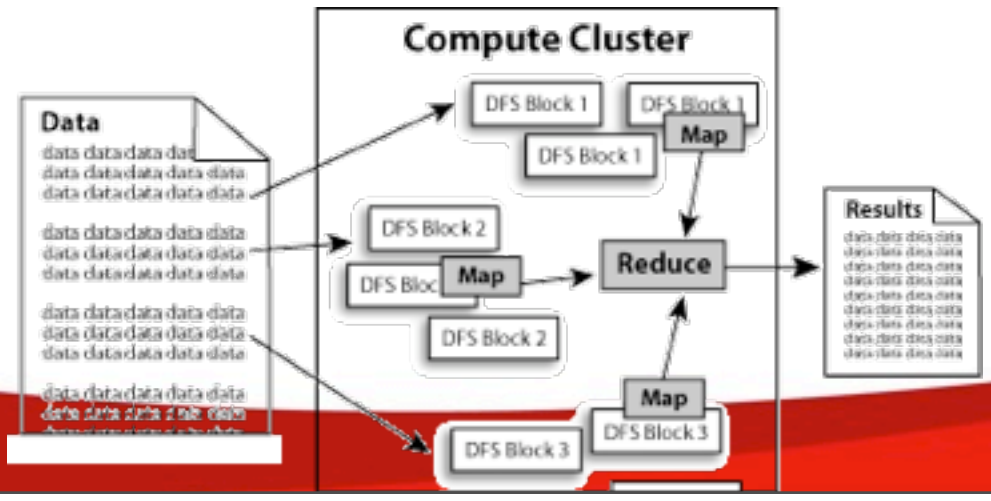
– Reduce
  - [1,2,3,4] – (sum) → 10

– **Divide and Conquer** paradigm

Copyright 2009 - Trend Micro Inc.

MapReduce

- **MapReduce** is a software **framework** introduced by **Google** to support **distributed computing** on **large data sets** on clusters of computers.
- Users specify
  - a ***map*** function that processes a key/value pair to generate a set of intermediate key/value pairs
  - a ***reduce*** function that merges all intermediate values associated with the same intermediate key ◦
- And MapReduce framework handles details of execution.



Copyright 2009 - Trend Micro Inc.

Application Area

- MapReduce framework is for computing certain kinds of **distributable problems** using **a large number of computers** (nodes), collectively referred to as a cluster.

- Examples
  - Large data processing such as search, indexing and sorting
  - Data mining and machine learning in large data set
  - Huge access logs analysis in large portals
- More applications

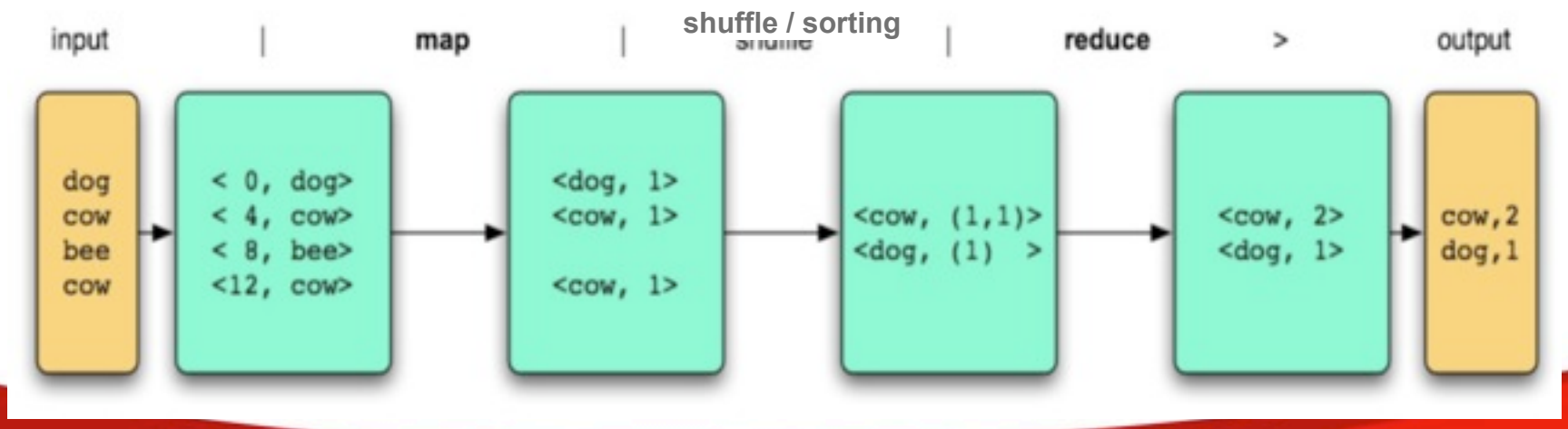
http://www.dbms2.com/2008/08/26/known-applications-of-mapreduce/

Copyright 2009 - Trend Micro Inc.

MapReduce Programming Model

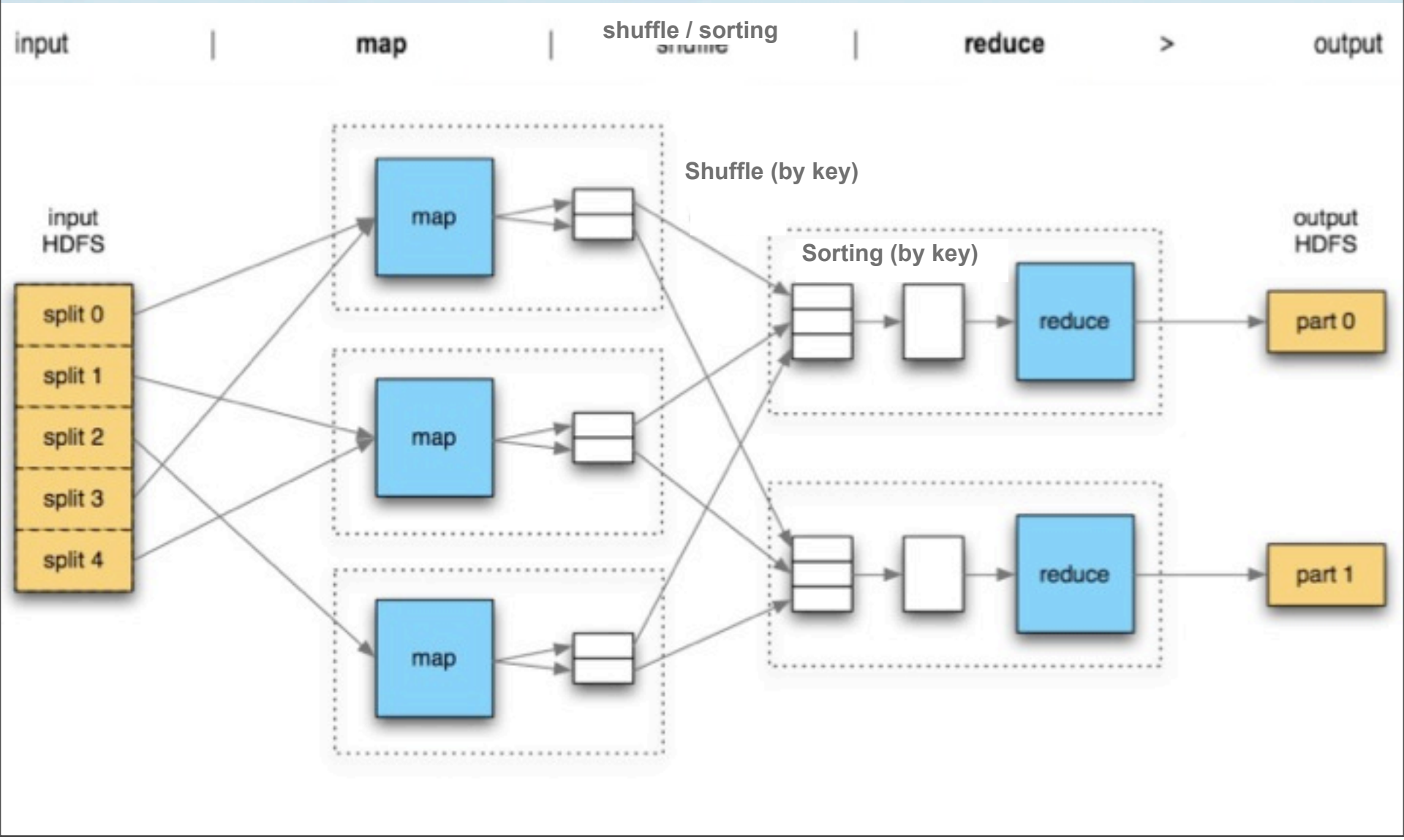
- User define two functions
  - **map (K1, V1) → list(K2, V2)**
    - takes an input pair and produces a set of intermediate key/ value
  - **reduce (K2, list(V2)) → list(K3, V3)**
    - accepts an intermediate key I and a set of values for that key. It merges together these values to form a possibly smaller set of values

- MapReduce Logical Flow



Copyright 2009 - Trend Micro Inc.

MapReduce Execution Flow



Word Count Example

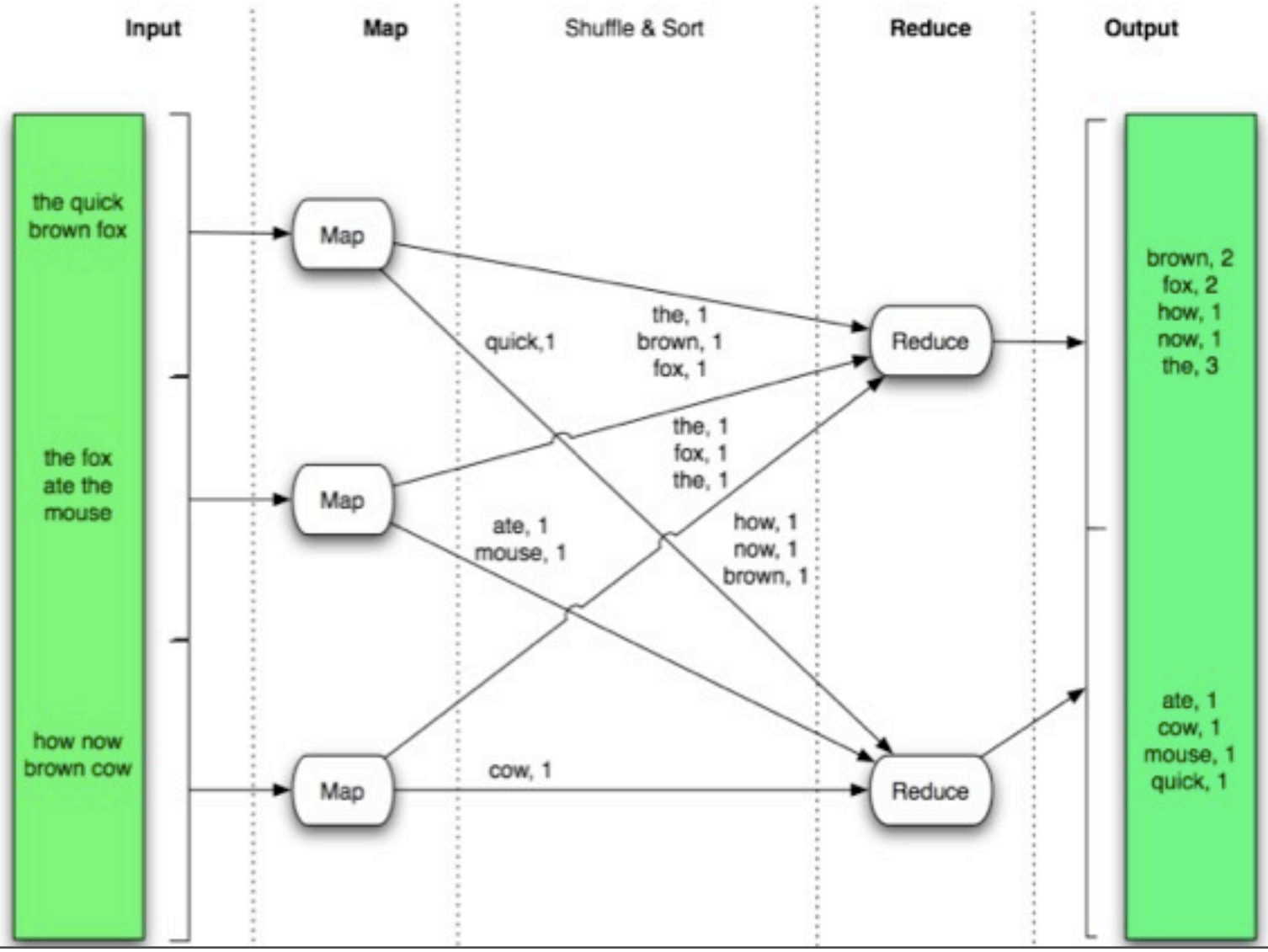
- Word count problem :
  - Consider the problem of **counting the number of occurrences of each word** in a large collection of documents.
- Input:
  - A collection of (document name, document content)
- Expected output
  - A collection of (word, count)
- User define:
  - Map: **(offset, line) → [(word1, 1), (word2, 1), ...]**
  - Reduce: **(word, [1, 1, ...]) → [(word, count)]**

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

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

Copyright 2009 - Trend Micro Inc.

Word Count Execution Flow



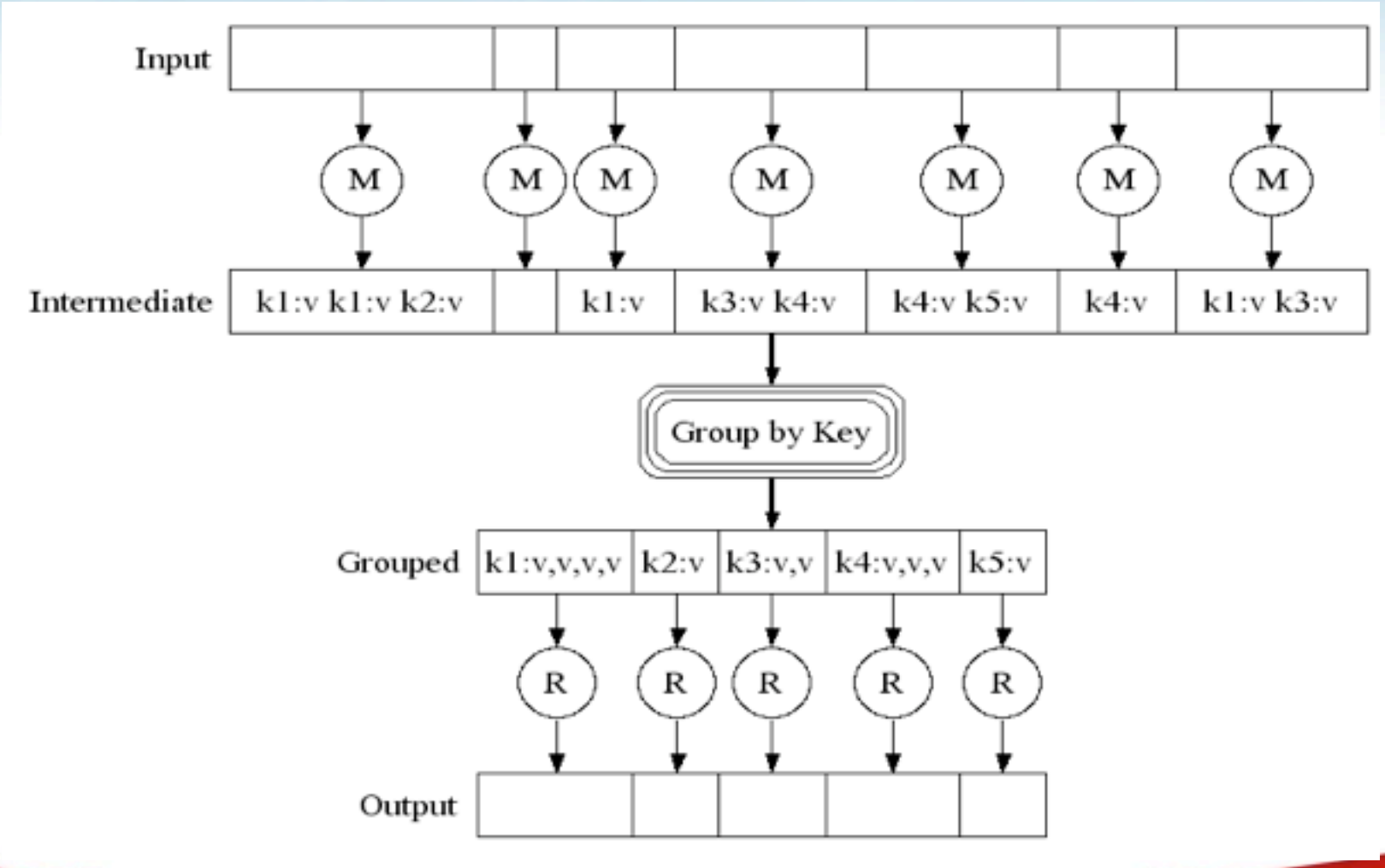
More MapReduce Examples

- Distributed Sort
  - Sorting large number of values
    - ***Map*:**
      - **extracts the key from each record, and emits a <key, record> pair.**
    - ***Reduce***
      - **emits all pairs unchanged.**

- Count of URL Access Frequency
  - Calculate the access frequency of URLs in web request logs especially when log file size and log file number are huge
    - ***Map*:**
      - **processes logs and output <URL,1>**
    - ***Reduce*:**
      - **adds together all values for the same URL and emits a <URL, total count> pair**

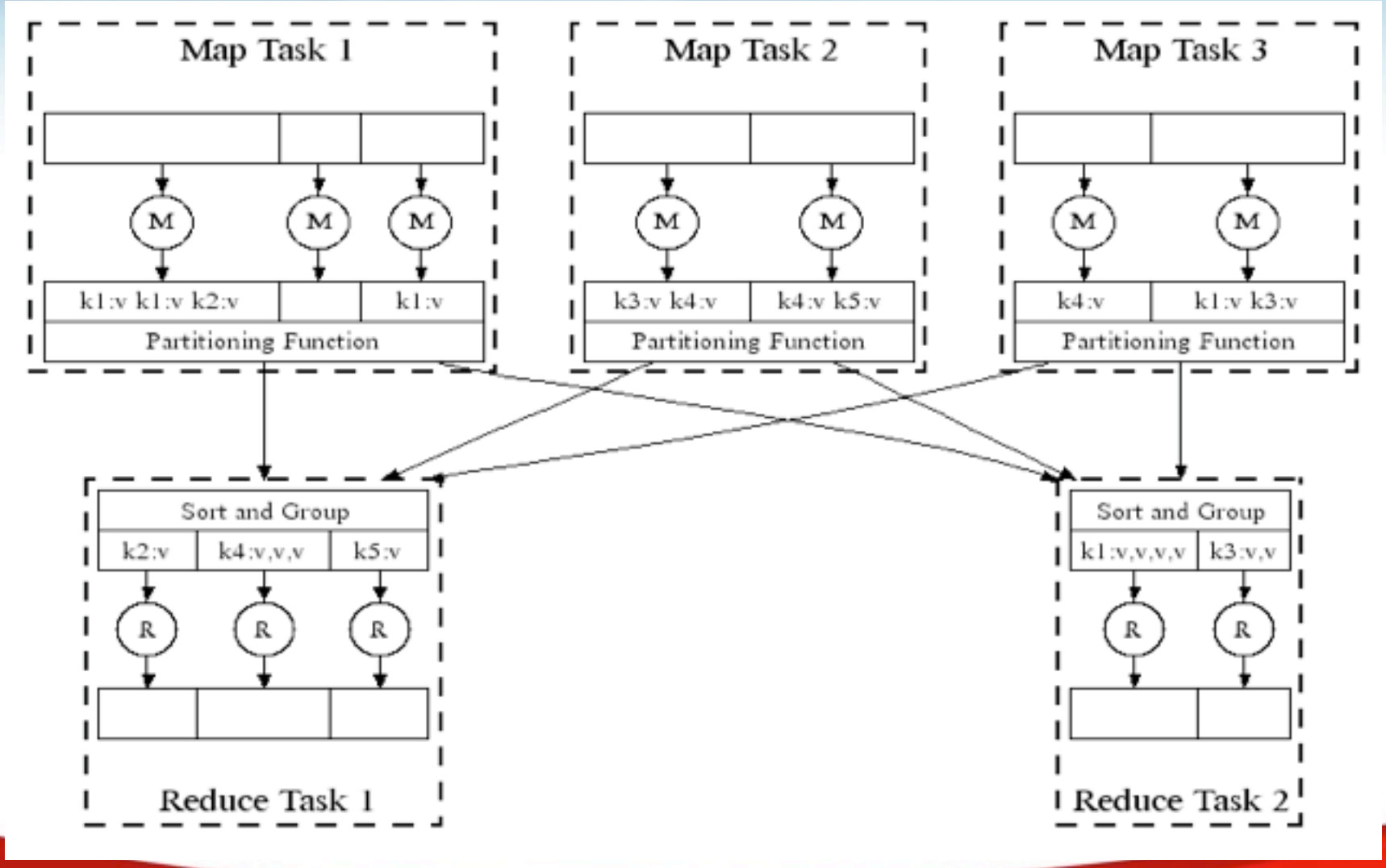
Copyright 2009 - Trend Micro Inc.

The logical flow of a MapReduce Job in Hadoop



Copyright 2009 - Trend Micro Inc.

The execution flow of a MapReduce Job in Hadoop



Copyright 2009 - Trend Micro Inc.

Introduction to

MapReduce Programming using Hadoop

Copyright 2009 - Trend Micro Inc.

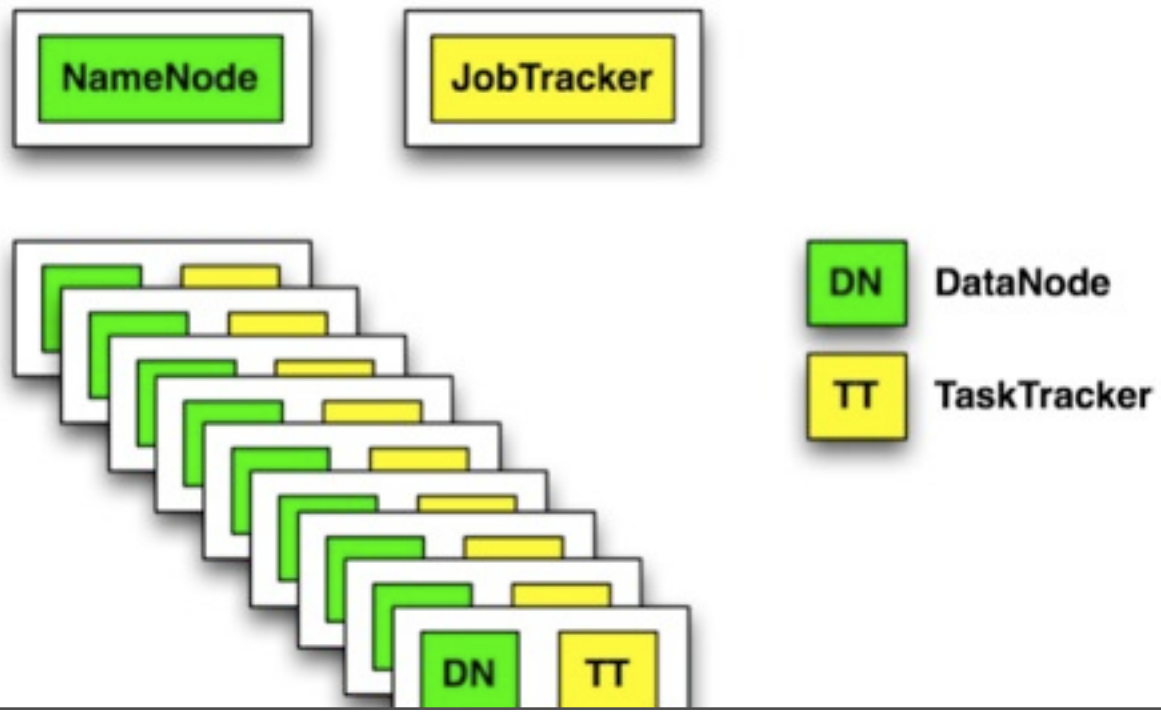
MapReduce from Hadoop

- Apache Hadoop project implements Google's MapReduce
  - Provide an open-source MapReduce framework
  - Using Java as native language.
  - Using Hadoop distributed file system (HDFS) as backend.
- Yahoo! Is the main sponsor.
- Yahoo!, Google, IBM, Amazon.. use Hadoop.
- Trend Micro use Hadoop MapReduce for threat solution research.

Copyright 2009 - Trend Micro Inc.

The Architecture of Hadoop MapReduce Framework

- Map/Reduce framework consists of
  - one single master **JobTracker**
  - one slave **TaskTracker** per cluster node.
- **JobTracker** is responsible for
  - scheduling the jobs' component tasks on the slaves
  - monitoring them and re-executing the failed tasks.
- **TaskTrackers** execute the tasks



pyright 2009 - Trend Micro Inc.

Minimal steps for MapReduce programming

1. Supply ***map*** *and* ***reduce*** *functions via implementations of appropriate interfaces and/or abstract-classes*
2. Comprise the ***job configuration***
3. Applications specify the ***input/output locations***
4. ***Job client*** *submits the job*
   1. JobTracker which then assumes the responsibility of distributing the software/configuration to the slaves
   2. scheduling tasks and monitoring them
   3. providing status and diagnostic information to the job-client.
5. Application continue handling the output

Copyright 2009 - Trend Micro Inc.

Simple skeleton of a MapReduce job

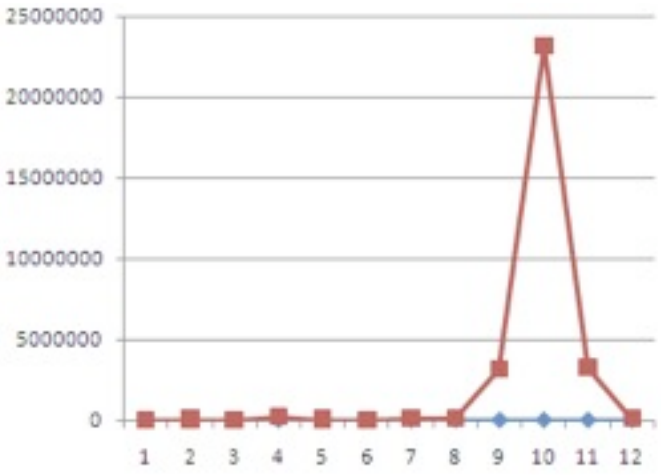
```
class MyJob {
        class Map {             // define Map function
        }
        class Reduce {          // define Reduce function
        }
        main() {
                // prepare job configuration and run job
                JobConf conf = new JobConf("MyJob.class");
                conf.setInputPath(…);
                conf.setOutputPath(…);
                conf.setMapperClass(Map.class);
                conf.setReduceClass(Reduce.class)
                // prepare job configuration and run job
                JobClient.runJob(conf);
                // continue handling the output
                …
        }
```

Get started

- Background
  - In Trend Micro, threat research department receives large amount of users' feedback log every day.
  - A feedback contain information of threat-related events detected by productions of Trend Micro.
  - Thread research folks analyze the feedback logs using data mining, correlation rules, and machine learning for detect more new viruses.
  - The size of total feedback logs is possible to reach several peta bytes in a year, so we take advantage of HDFS to keep such a large amount of data and process them using MapReduce.

- Problem
  - Calculate the distribution of feedback logs number in a day
  - Log format
    - Each line contains a log record
    - Log format

No.,GUID, timestamp,module, behavior
1, 123, 131231231, VSAPI, open file
2, 456, 123123123, VSAPI, connect internet



Copyright 2009 - Trend Micro Inc.

Define Map function (0.19)

- Implements Mapper interface and define map() method

org.apache.hadoop.mapred

Interface Mapper<K1,V1,K2,V2>

All Superinterfaces:

Closeable, JobConfigurable

- Input of map method : <key, value>

```
void map(K1 key,
         V1 value,
         OutputCollector<K2,V2> output,
         Reporter reporter)
         throws IOException
```

- K1, K2 should implement ***WritableComparable***
- V1, V2 should implement ***Writable***
- For each input key pair, the map() function will be called once to process it
- The OutputCollector has collect() method to collect the immediate result
- Applications can use the Reporter to report progress

Copyright 2009 - Trend Micro Inc.

Implements Mapper

```
class MapClass extends MapReduceBase
implements Mapper<LongWritable, Text, Text, IntWritable> {
        private final static IntWritable one = new IntWritable(1);
        private Text hour = new Text();
        public void map( LongWritable key, Text value,
OutputCollector<Text,IntWritable> output, Reporter reporter) throws
IOException {
              String line = ((Text) value).toString();
                String[] token = line.split(",");
                String timestamp = token[1];
                Calendar c = Calendar.getInstance();
                c.setTimeInMillis(Long.parseLong(timestamp));
                Integer h = c.get(Calendar.HOUR);
                hour.set(h.toString());
                output.collect(hour, one)
}}}
```

Copyright 2009 - Trend Micro Inc.

Define *Reduce* function

- Implements Reducer interface and define reduce() method

org.apache.hadoop.mapred

Interface Reducer<K2,V2,K3,V3>

All Superinterfaces:

Closeable, JobConfigurable

- Input of reduce method : <key, a list of value>

```
void reduce(K2 key,
            Iterator<V2> values,
            OutputCollector<K3,V3> output,
            Reporter reporter)
            throws IOException
```

- K2, K3 should implement ***WritableComparable***
- V2, V3 should implement ***Writable***
- The OutputCollector has collect() method to collect the final result
- The output of the Reducer is **not re-sorted.**
- Applications can use the Reporter to report progress

Copyright 2009 - Trend Micro Inc.

Implements Reducer

```
class ReduceClass extends MapReduceBase implements Reducer< Text,
IntWritable, Text, IntWritable> {
        IntWritable SumValue = new 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();
                SumValue.set(sum);
                output.collect(key, SumValue);
}}
```

Copyright 2009 - Trend Micro Inc.

Configuring and running your jobs

- `JobConf` class is used for setting
  - `Mapper, Reducer, Inputformat, OutputFormat, Combiner, Partitioner` and etc. .
  - input path
  - output path
  - Other job configurations
    - Failure percentage of map and reduce tasks
    - Retry number when errors arise
- `JobClient` class takes `JobConf` as configuration when running your job
  - `JobClient.runJob(conf)`
    - Submits the job and returns only after the job has completed.
  - `JobClient.submitJob(conf)`
    - Only submits the job, then poll the returned handle to the RunningJob to query status and make scheduling decisions.
    - JobClient.setJobEndNotificationURI(URI)
      - Sets up a notification upon job-completion, thus avoiding polling.

Copyright 2009 - Trend Micro Inc.

More …

- ***Partitioner***
  - Partitioner partitions the key space for “shuffle” phase

org.apache.hadoop.mapred

Interface Partitioner<K2,V2>

All Superinterfaces:

JobConfigurable

getPartition

```
int getPartition(K2 key,
                 V2 value,
                 int numPartitions)
```

  - the total number of partitions i.e. number of reduce-tasks for the job

- ***HashPartitioner*** is the default Partitioner.

Copyright 2009 - Trend Micro Inc.

Main program

```
Class MyJob{
public static void main(String[] args) {
        JobConf conf = new JobConf(MyJob.class);
        conf.setJobName(”Caculate feedback log time distribution");
        // set path
        conf.setInputPath(new Path(args[0]));
        conf.setOutputPath(new Path(args[1]));
        // set map reduce
        conf.setOutputKeyClass(Text.class);  // set every word as key
        conf.setOutputValueClass(IntWritable.class); // set 1 as value
        conf.setMapperClass(MapClass.class);
        conf.setCombinerClass(Reduce.class);
        conf.setReducerClass(ReduceClass.class);
        onf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);
        // run
        JobClient.runJob(conf);
}}
```

Copyright 2009 - Trend Micro Inc.

Build

1. compile
   - javac -classpath hadoop-*-core.jar -d MyJava MyJob.java
- package
   - jar –cvf MyJob.jar -C MyJava .
- execution
   - bin/hadoop jar MyJob.jar MyJob input/ output/

Copyright 2009 - Trend Micro Inc.

Launch the program

- `bin/hadoop jar MyJob.jar MyJob input/ output/`

```
09/05/16 17:39:58 INFO mapred.FileInputFormat: Total input paths to process : 13
09/05/16 17:39:58 INFO mapred.JobClient: Running job: job_200905161736_0001
09/05/16 17:39:59 INFO mapred.JobClient:  map 0% reduce 0%
09/05/16 17:40:09 INFO mapred.JobClient:  map 7% reduce 0%
09/05/16 17:40:19 INFO mapred.JobClient:  map 23% reduce 0%
09/05/16 17:40:37 INFO mapred.JobClient:  map 38% reduce 0%
09/05/16 17:40:39 INFO mapred.JobClient:  map 53% reduce 0%
09/05/16 17:40:40 INFO mapred.JobClient:  map 61% reduce 0%
09/05/16 17:40:42 INFO mapred.JobClient:  map 61% reduce 7%
09/05/16 17:40:43 INFO mapred.JobClient:  map 76% reduce 7%
09/05/16 17:40:45 INFO mapred.JobClient:  map 92% reduce 7%
09/05/16 17:40:46 INFO mapred.JobClient:  map 100% reduce 7%
09/05/16 17:40:48 INFO mapred.JobClient:  map 100% reduce 12%
09/05/16 17:40:58 INFO mapred.JobClient:  map 100% reduce 100%
09/05/16 17:41:00 INFO mapred.JobClient: Job complete: job_200905161736_0001
09/05/16 17:41:00 INFO mapred.JobClient: Counters: 19
09/05/16 17:41:00 INFO mapred.JobClient:   Job Counters
09/05/16 17:41:00 INFO mapred.JobClient:     Launched reduce tasks=1
09/05/16 17:41:00 INFO mapred.JobClient:     Rack-local map tasks=5
09/05/16 17:41:00 INFO mapred.JobClient:     Launched map tasks=13
09/05/16 17:41:00 INFO mapred.JobClient:     Data-local map tasks=8
09/05/16 17:41:00 INFO mapred.JobClient:   FileSystemCounters
09/05/16 17:41:00 INFO mapred.JobClient:     FILE_BYTES_READ=177
09/05/16 17:41:00 INFO mapred.JobClient:     HDFS_BYTES_READ=20635
09/05/16 17:41:00 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=842
09/05/16 17:41:00 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=111
09/05/16 17:41:00 INFO mapred.JobClient:   Map-Reduce Framework
09/05/16 17:41:00 INFO mapred.JobClient:     Reduce input groups=1
09/05/16 17:41:00 INFO mapred.JobClient:     Combine output records=9
09/05/16 17:41:00 INFO mapred.JobClient:     Map input records=613
09/05/16 17:41:00 INFO mapred.JobClient:     Reduce shuffle bytes=249
09/05/16 17:41:00 INFO mapred.JobClient:     Reduce output records=1
09/05/16 17:41:00 INFO mapred.JobClient:     Spilled Records=18
09/05/16 17:41:00 INFO mapred.JobClient:     Map output bytes=1564
09/05/16 17:41:00 INFO mapred.JobClient:     Map input bytes=20635
09/05/16 17:41:00 INFO mapred.JobClient:     Combine input records=92
09/05/16 17:41:00 INFO mapred.JobClient:     Map output records=92
09/05/16 17:41:00 INFO mapred.JobClient:     Reduce input records=9
```

Copyright 2009 - Trend Micro Inc.

Monitoring the execution from web console

http://172.16.203.132:50030/

State: RUNNING
Started: Sat May 16 17:36:12 CST 2009
Version: 0.20.0, r763504
Compiled: Thu Apr 9 05:18:40 UTC 2009 by ndaley
Identifier: 200905161736

Cluster Summary (Heap Size is 4.94 MB/198.5 MB)

Maps	Reduces	Total Submissions	Nodes	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node
0	1	4	2	4	4	4.00

Scheduling Information

Queue Name	Scheduling Information
default	N/A

Filter (Jobid, Priority, User, Name)
Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Running Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information
job_200905161736_0004	NORMAL	root	grep-search	100.00%	13	13	25.64%	1	0	NA

Completed Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information
job_200905161736_0001	NORMAL	root	grep-search	100.00%	13	13	100.00%	1	1	NA
job_200905161736_0002	NORMAL	root	grep-sort	100.00%	1	1	100.00%	1	1	NA
job_200905161736_0003	NORMAL	root	grep-search	100.00%	13	13	100.00%	1	1	NA

Failed Jobs

none

Local Logs

How many mappers and reducers?

- How many maps?
  - driven by the total size of the inputs, that is, the total number of blocks of the input files.
  - Can use JobConf's ***setNumMapTasks(int)*** to provides a hint to the framework.

- How many reducers?
  - The number of reduces for the job is set by the user via ***JobConf.setNumReduceTasks(int)***
  - Increasing the number of reduces increases the framework overhead, but increases load balancing and lowers the cost of failures.

Copyright 2009 - Trend Micro Inc.

Non-Java Interfaces from Hadoop

- Hadoop Pipes
  - Provide C++ API and library for MapReduce
  - The C++ application runs as a sub-process of java task
- Hadoop Streaming
  - A utility which allows users to create and run jobs with any executables (e.g. shell utilities) as the mapper and/or the reducer.

Copyright 2009 - Trend Micro Inc.

Reference

- Google MapReduce paper
  - http://labs.google.com/papers/mapreduce.html
- Google: MapReduce in a Week
  - http://code.google.com/edu/submissions/mapreduce/listing.html
- Google: Cluster Computing and MapReduce
  - http://code.google.com/edu/submissions/mapreduce-minilecture/listing.html
- Hadoop project official site
  - http://hadoop.apache.org/core/

Copyright 2009 - Trend Micro Inc.

Reference

- **MapReduce Tools for Eclipse Plug-In (by IBM)**
  - A robust plug-in that brings Hadoop support to the Eclipse platform
  - Download
- **Hadoop Virtual Image (by Google)**
  - his VMware image contains a preconfigured single node instance of Hadoop. This provides the same interface as a full cluster without any of the overhead. It is suitable for educators exploring the platform and students working independently. The following Download and VMware Player links point to websites external to Google.
  - **Document link**

Copyright 2009 - Trend Micro Inc.