Introduzione a MapReduce ed Hadoop

Piero Molino

CLUB SVILUPPATORI

BARI

Chi sono

Dottorando UniBa

Interessi di ricerca: Question Answering, Motori di Ricerca , Semantica, Elaborazione del Linguaggio, Machine Learning

Co-founder e CTO di QuestionCube

Internship agli Yahoo! Labs di Barcellona

Indice

Problema

Qualcuno ha detto Big Data?

Volume Variety Velocity

Nei termini di un database...

Righe Campi Tabelle Inserimenti/sec

Cattive Notizie

20+ miliardi di pagine web x 20 kB = 400+ TB

~200 hd da 2 TB

~40 MB/sec in lettura = ~4 mesi solo per leggere

Ancora di più per farci qualcosa

Buone Notizie

Stesso task su 1000 macchine = <3 ore

però...

comunicazione, coordinazione, recupero dalle failure, reporting, debugging, ottimizzazione, ...

soluzione ad hoc per ciascun problema

MapReduce

MapReduce

Modello computazionale per il calcolo parallelo e distribuito proposto da Google (Jeffrey Dean e Sanjay Ghemawat) nel 2004

Parallelizzazione e distribuzione automatica

Gestione delle failure

Ottimizzazione I/O

Status monitoring

Come funziona

Lettura di molti dati (record)

Map - estrazione di informazioni dai record

Mischia e Ordina

Reduce - aggrega, somma, filtra o trasforma

Scrittura dei risultati

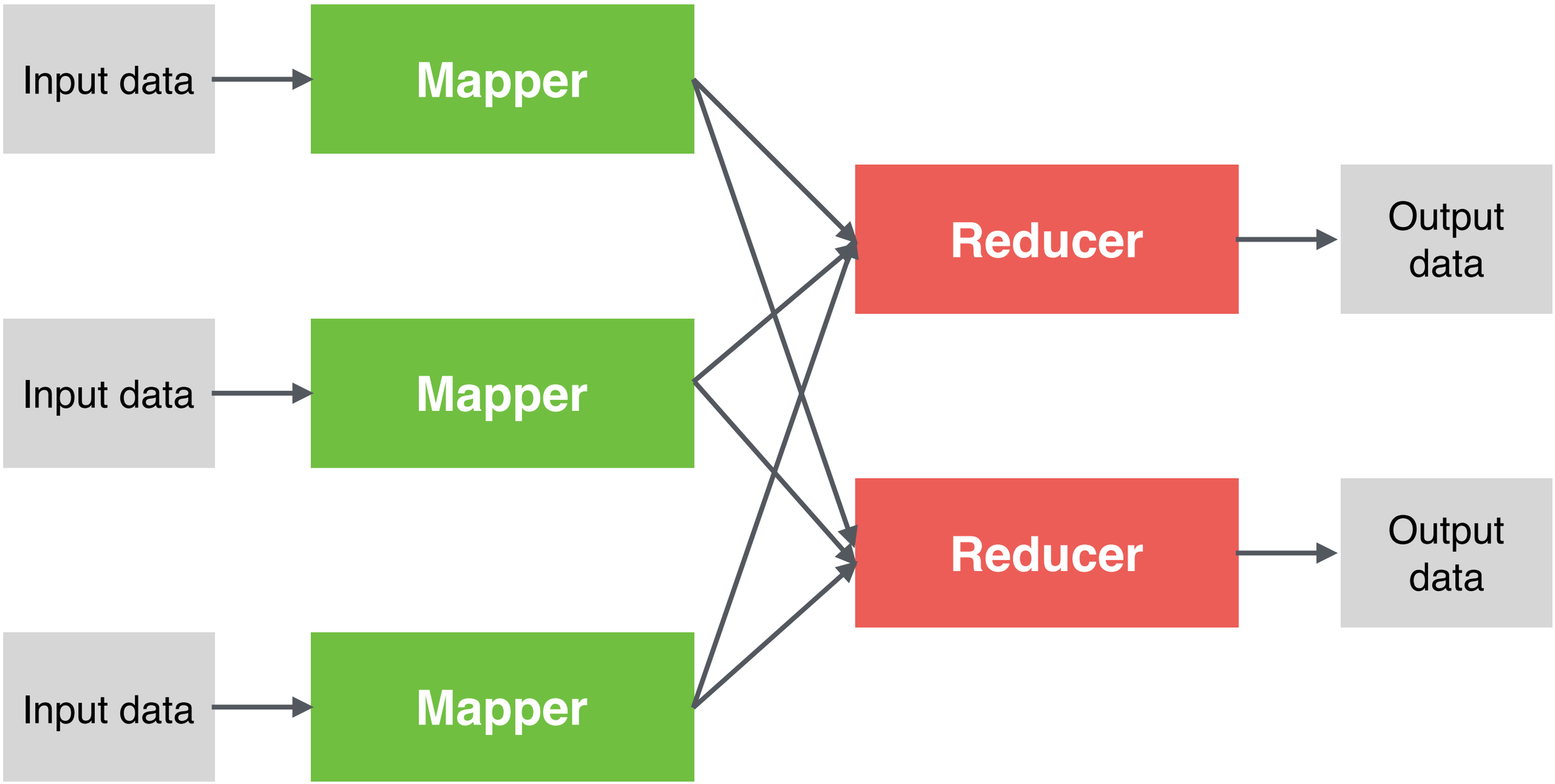
Formalmente (vista logica)

```
map(in_key, in_value) →
   list(<inter_key, inter_value>)

reduce(inter_key, list(inter_value)) →
   list(<out_key, out_value>)
```

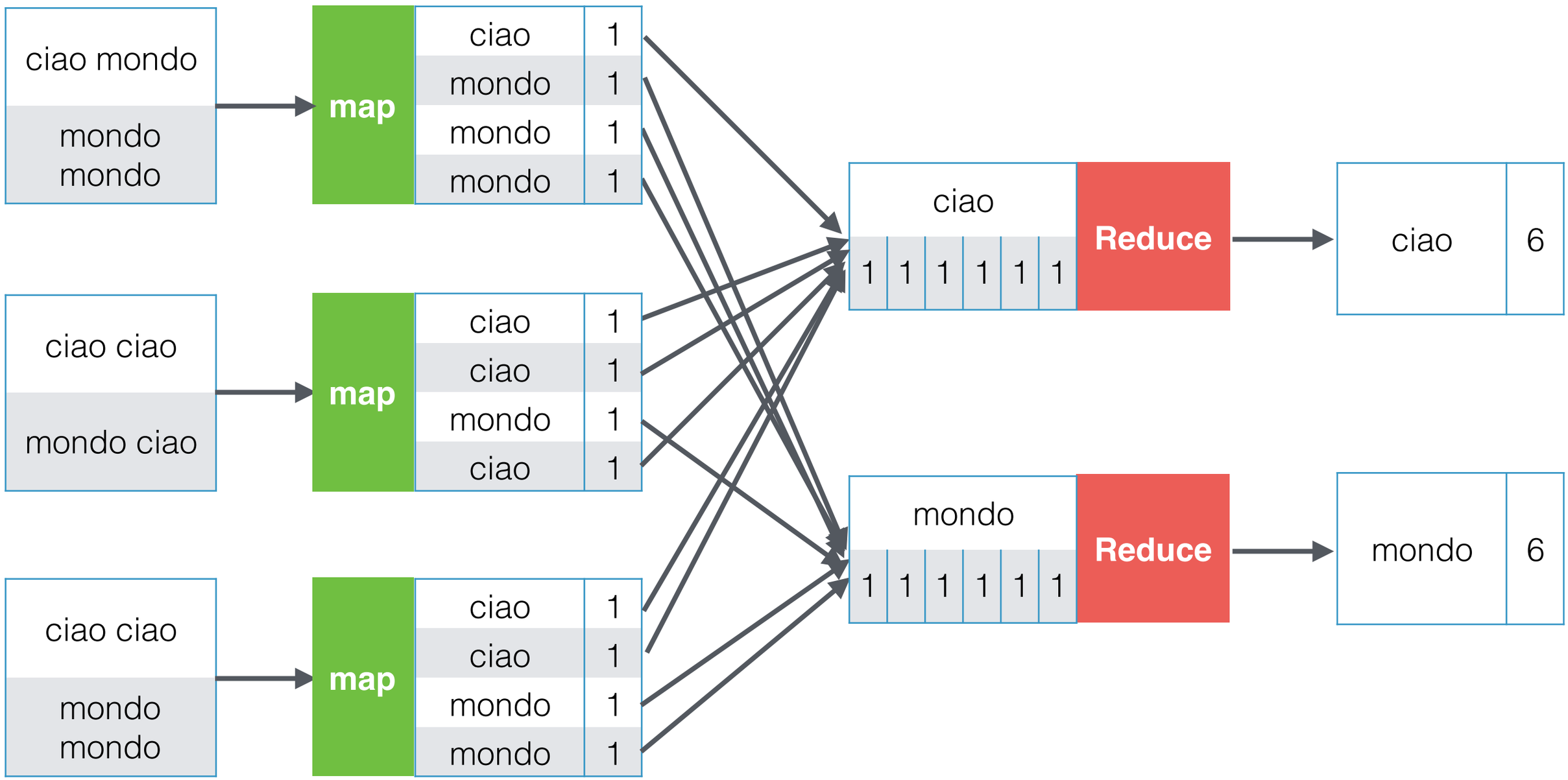
Esempio - Pseudo Codice

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

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

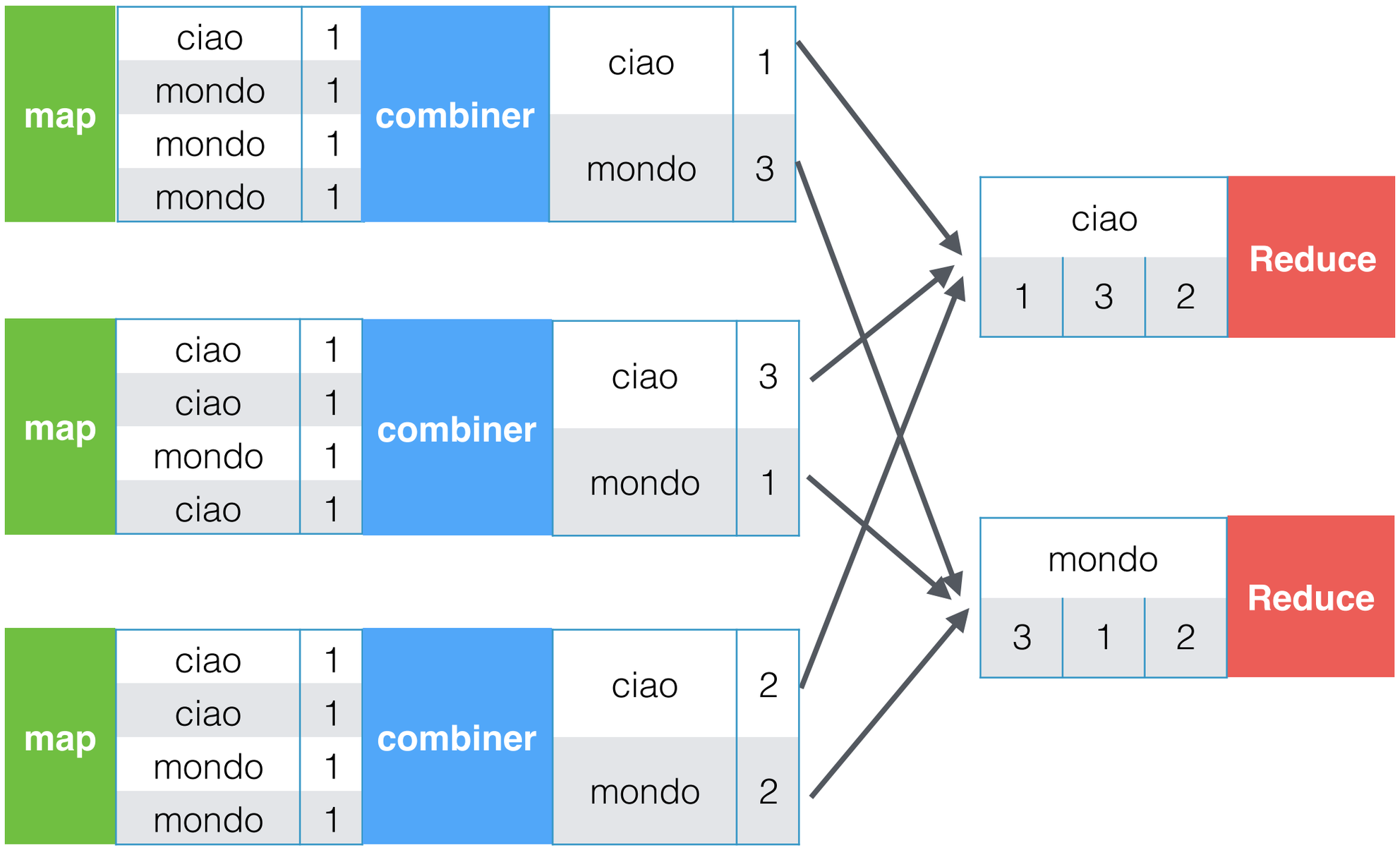
Schema di parallelizzazione



Esempio


ciao mondo
mondo mondo
map
ciao 1
mondo 1
mondo 1
mondo 1
ciao ciao
mondo ciao
map
ciao 1
ciao 1
mondo 1
ciao 1
ciao ciao
mondo mondo
map
ciao 1
ciao 1
mondo 1
mondo 1
ciao
1 1 1 1 1 1
Reduce
ciao 6
mondo
1 1 1 1 1 1
Reduce
mondo 6

Combiner


map
ciao 1
mondo 1
mondo 1
mondo 1
combiner
ciao 1
mondo 3
map
ciao 1
ciao 1
mondo 1
ciao 1
combiner
ciao 3
mondo 1
map
ciao 1
ciao 1
mondo 1
mondo 1
combiner
ciao 2
mondo 2
ciao
1 3 2
Reduce
mondo
3 1 2
Reduce

Partizionamento delle chiavi

Ci sono meno Reducer che chiavi

Funzione di partizionamento che divide le chiavi in n = #reducer blocchi

Si può specificare esplicitamente, ma spesso è inutile

Es. Se key = stringa, n = 2

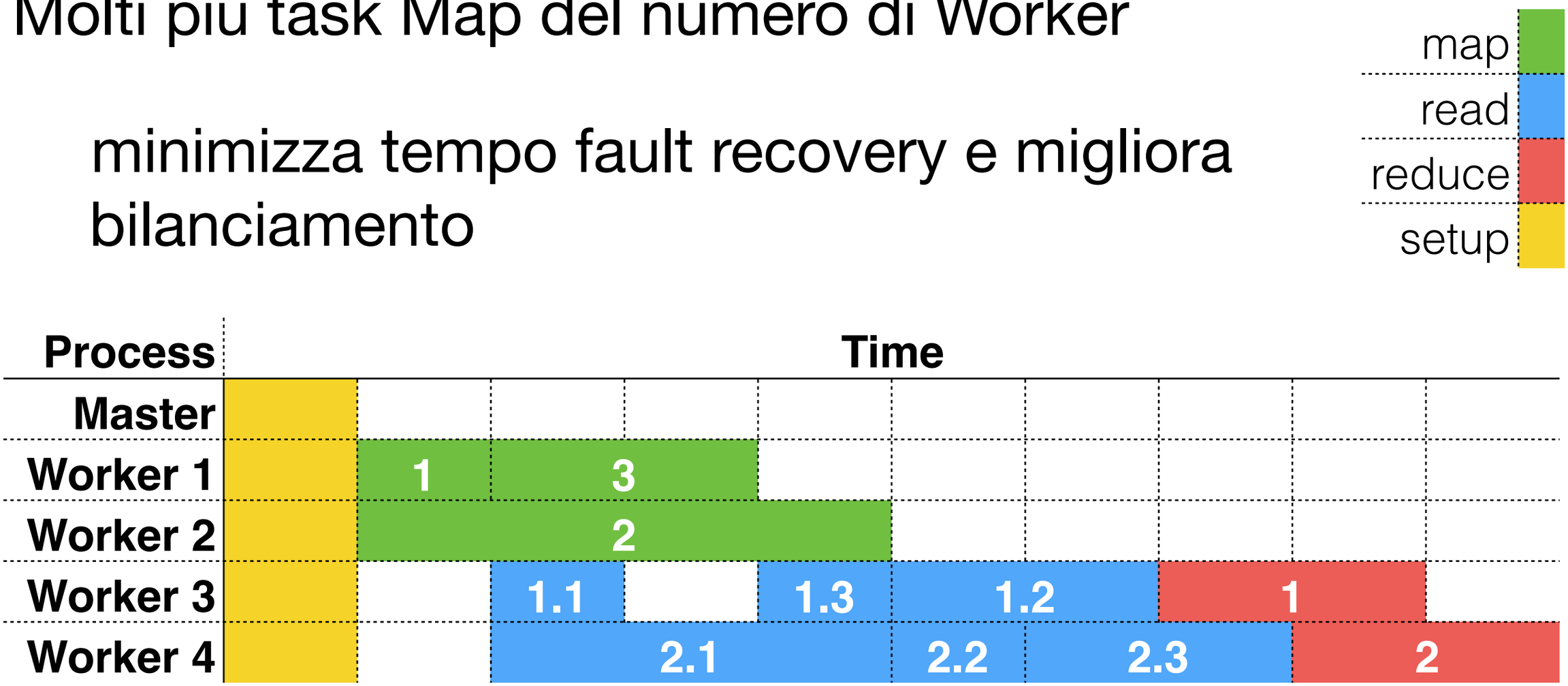
```
IF firstChar(key) <  "n" THEN
partizione 1 ELSE partizione 2
```

Esecuzione

Un unico nodo Master e tanti Worker

Molti più task Map del numero di Worker

minimizza tempo fault recovery e migliora bilanciamento



Fault Tolerance

Master

Single Point of Failure, ma l'esecuzione può essere ripresa

Worker:

heartbeat periodici

ri-esecuzione di task di map e reduce interrotti o troppo lenti (chi finisce per primo "vince")

Locality Optimization

Dati in input separati (per riga) in blocchi da 64 MB

Blocchi replicati dal file system distribuito (GFS, Hadoop) sulle macchine che eseguiranno il map

Ogni task eseguito su dati presenti sulla macchina per ottimizzare la velocità di lettura

I combiner utili per ridurre l'uso di banda di rete

I dati scambiati da mapper a reducer sono compressi

Utilizzi

grep distribuito

sort distribuito

statistiche sui log

calcolo inverso dei link web

creazione di indici invertiti

algoritmi su grafi

calcolo matriciale

count per i language model

clustering di documenti

machine learning

Hadoop

Hadoop

Google ha un'implementazione proprietaria di MapReduce (chiamata, per evitare confusione, MapReduce)

Hadoop è un'implementazione open source di MapReduce in Java, sviluppata principalmente da Yahoo!

MapReduce (l'implementazione) usa GoogleFS per la gestione distribuita dei dati, Hadoop utilizza l'analogo HDFS

Ecosistema

Basandosi su Hadoop si sono sviluppate una serie di tecnologie che ne colmano alcuni limiti e ne semplificano l'utilizzo

Fondamentalmente sono strati che si sovrappongono ad Hadoop per permettere di usarlo ad un più alto livello di astrazione

Pig, HBase, Hive, Mahout, Flume, ZooKeeper, Sqoop, Chukwa, Hama, Avrol, Ambari, Cassandra, Spark

Esempio Count

Tips and Tricks

In caso di file in input di diverso formato (tabelle diverse):

```
FileSplit f = (FileSplit)
context.getInputSplit();

String filePath = f.getPath().toString();

String[] pathComponents = filePath.split("/");

String originalFile =
pathComponents[pathComponents.length - 1];
```

Tips and Tricks

Se il reducer ha bisogno di sapere da quale file/tabella proviene il record

```
outputVal.set("$C$" + fields.get(2));
```

Esempio Join

Esempio Prodotto Matriciale

$C = A \times B$

$A \in R^{LxM}$

$B \in R^{MxN}$

Mapper

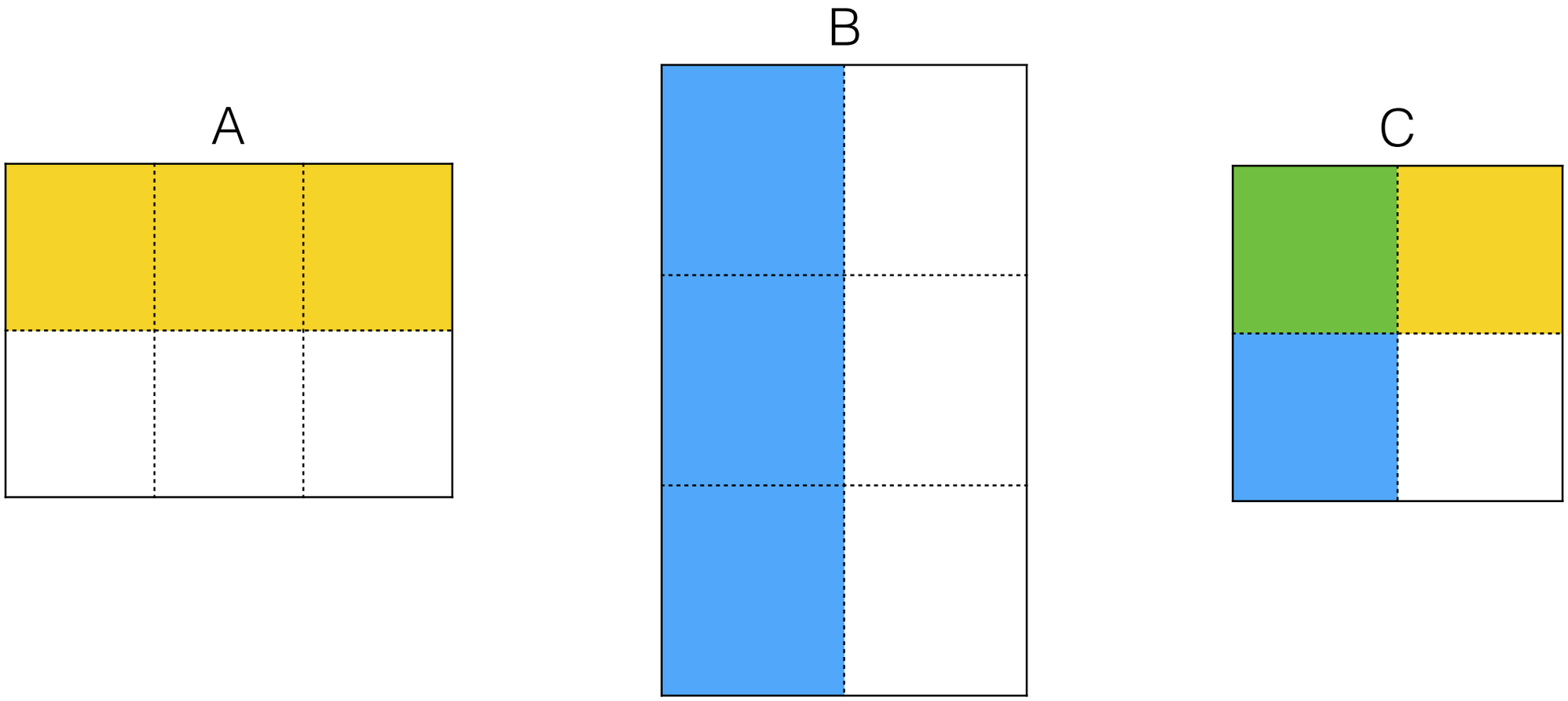
$\forall (i,j) \in A$ emettere $<ik, a_{ij}>$ per $k \in [1, N]$

$\forall (j,k) \in B$ emettere $<ik, b_{jk}>$ per $i \in [1, L]$

Reducer

$<ik, \sum a_{ij} \cdot b_{jk}>$

Esempio Prodotto Matriciale



MRUnit

Debugging Hadoop

Debuggare Hadoop è un processo doloroso

Si procede a tentativi, si analizzano i log di errore, si esegue più volte il processo su sottoinsiemi di dati

Ogni esecuzione impiega molto tempo per il setup e la replica dei dati

Principale cause di mal di testa a Yahoo! Lab Barcelona (e credo anche in Google)

MRUnit

Framework per l'esecuzione di test di unità per Hadoop

Permette di testare la funzione Map, la funzione Reduce ed entrambe in cascata

Si controlla l'output tramite assert in stile JUnit

Esempio Test Join

mrjob

mrjob

Yelp ha tutta la sua codebase in Python e decide di scrivere una libreria open source in Python che permette di

simulare in locale l'esecuzione di Hadoop

lanciare in modo semplice l'esecuzione su di un cluster Hadoop

lanciare l'esecuzione altrettanto semplicemente su Amazon Elastic MapReduce

Esempio Count Python

Esecuzione con mrjob

Locale

```
$ python my_job.py [-r local] input.txt
```

Hadoop

```
$ python my_job.py -r hadoop hdfs://my_home/
input.txt
```

Amazon Elastic MapReduce

```
$ python my_job.py -r emr s3://my-inputs/
input.txt
```

Pig

Pig

Una delle tecnologie dell'ecosistema Hadoop

Permette di eseguire query SQL-like tramite Hadoop

Velocizza (di molto) la scrittura di Join, Select e Projection rispetto ad Hadoop "liscio"

Esempio Pig

```
students = LOAD 'student' USING PigStorage()
AS (id:chararray, name:chararray, age:int,
gpa:float);

courses = LOAD 'course' USING PigStorage() AS
(id:chararray, name:chararray, credits:int,
student_id:chararray);

student_course = join students by id, courses
by student_id;

dump student_course;
```

Altre tecnologie

Nuovi sviluppi

Scrivere un workflow map-reduce in java richiede la scrittura di molto codice

Non tutti i problemi sono rappresentabili in catene di map → reduce → ... → map → reduce

Tecnologie costruite attorno ad Hadoop:

Scalding - Scoobi - Scrunch - Cascalog - Hive - Pig

Spark - Shark - Impala - PrestoDB

Framework per semplificare la scrittura di workflow

Pig (https://pig.apache.org)

Scalding (https://github.com/twitter/scalding)

Scoobi (https://github.com/NICTA/scoobi)

Scrunch (https://github.com/cloudera/crunch/tree/master/scrunch)

Cascalog (https://github.com/nathanmarz/cascalog)

Hive (http://hive.apache.com)

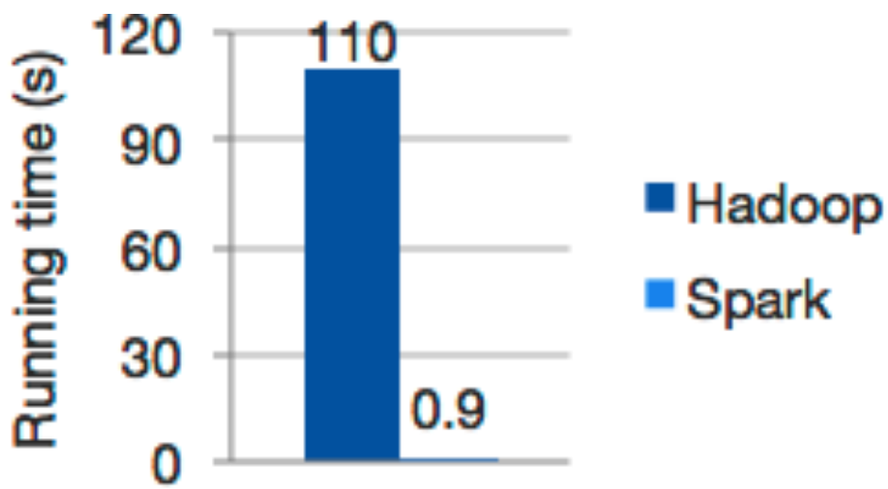
Real time processing

Hadoop è adatto al processing batch di dati

Spark (http://spark.incubator.apache.org), engine generico e veloce per il processing real time di dati (100x faster than Hadoop MapReduce).

Shark, estensione di spark per supportare Hive SQL

```
file =
spark.textFile("hdfs://...")

file.flatMap(line =>
line.split(" "))
     .map(word => (word, 1))
     .reduceByKey(_ + _)
```



Engine SQL

Cloudera Impala (http://impala.io/), Engine SQL distribuito costruito su Hadoop per l'esecuzione di query in real time

PrestoDB (http://prestodb.io/), Engine SQL distribuito per l'esecuzione di query real time su Big Data (Hive, HBase, database relazionali, etc.)

Discussione

Discussione

Le tecnologie dell'ecosistema Hadoop sono sempre più utilizzate per velocizzare lo sviluppo e per il processing real time

Quando ha senso usare Hadoop al posto di tecnologie alternative? Bisogna avere TANTI dati

Rule of thumb: decine milioni di righe/entry, centinaia di GB sono il MINIMO perché valga la pena

Quanti di voi gestiscono così tanti dati?

Grazie per l'attenzione