CORSO I.F.T.S

“TECNICHE PER LA PROGETTAZIONE E LA GESTIONE DI DATABASE ”

Ing. Mariano Di Claudio

Lezione del 15/10/2014

Indice

1. **Processo di analisi/elaborazione dei Big Data**

1. **Hadoop**
   - Caratteristiche chiave
   - Architettura
   - HDFS
   - MapReduce

2. **Installazione e configurazione del cluster**
   - Aspetti teorici e pratici
   - Esempio applicativo MapReduce

Processo di elaborazione dei Big Data

- Uno dei contesti in cui i **Big Data** trovano una forte applicazione è quello della ***Business Intelligence***, in cui *contribuiscono a creare dei sistemi di supporto alle decisioni*.

Big Data

Acquisizione

Analista

Immagazzinamento
Organizzazione
Integrazione/Arricchimento
Analisi

Processo Iterativo
Per estrapolare informazioni utili

Inserimento

DataWarehouse

Molte delle tecnologie utilizzate nelle 4 fasi di elaborazione sono legate ***all'ecosistema Hadoop*** (che insieme alle sue componenti è sviluppato in Java).

Acquisizione dei Big Data

- L'**acquisizione** delle diverse tipologie di dati può essere realizzate con mezzi differenti, le **principali categorie** sono:

  - ***API di chi fornisce i dati.*** Si parla principalmente di dati provenienti da Social Network o app di condivisione.

  - ***Importazione dei dati mediante strumenti ETL*** (Sqoop o PDI).

  - ***Utilizzo di software di web scraping.*** Raccolta automatica di dati dal web, ad esempio mediante parser HTML.

  - ***Lettura di stream di dati.*** Tecnologie per acquisire flussi di dati, come ad esempio le piattaforme CEP (dati provenienti dal web o da sensori).

Immagazzinamento e Organizzazione

- In questa fase bisogna considerare **due aspetti**:
  1. La **gestione di grandi moli di dati**.
  2. Dati **non strutturati o semi-strutturati**.

- La **principale tecnologia software** adatta a questo scopo è ***Hadoop***, piattaforma di calcolo distribuito che presenta o supporta diverse componenti:
  - **Hadoop common**: strato sw che fornisce funzioni di supporto ad altri moduli.
  - **HDFS**: file system distribuito che offre una elevata capacità di accesso ai dati.
  - **YARN**: sistema di scheduling e gestione delle risorse condivise.
  - **MapReduce**: sistema di parallel processing per grandi moli di dati.
  - **Hbase**: Database NoSQL di tipo Column Family.

Integrazione

Prima di passare alla fase di analisi è spesso necessario eseguire delle ***integrazioni/elaborazioni*** sui dati immagazzinati.

- **Sqoop**

permette di realizzare **l'integrazione con DB esterni**, spostando dati da e verso Hadoop.

- **Apache Tika**

Sw che permette di **trattare formati differenti** (PDF, HTML, XML, Word, Excel) nello stesso modo. Specie per dati provenienti dal Web o da repository di documenti.

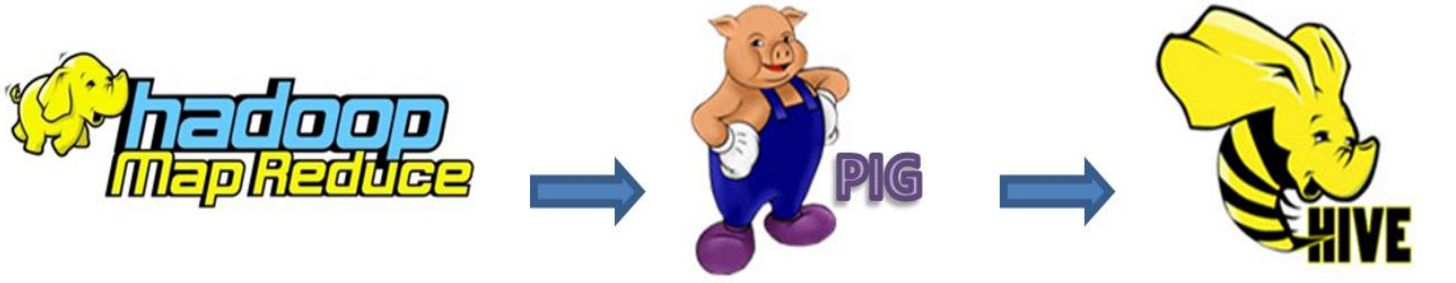
- **Hive**

Sistema di datawarehousing costruito su Hadoop, che permette l'aggregazione di dati, l'esecuzione di query e l'analisi di grandi dataset mediante **HiveQL (simile ad SQL)**. Così si nasconde la complessità di scrivere funzioni MapReduce.

Analisi

- ***L'analisi dei Big Data*** *nel mondo Hadoop può essere fatta con diversi strumenti che sfruttano* ***MapReduce e HDFS****.* Tra i diversi tool a disposizione, importanti sono **Pig** (ecosistema Hadoop), **R** (no Hadoop) e **Mahout** (ecosistema Hadoop - non integrerà i nuovi algoritmi di MapReduce).

- **Pig** con il suo ***linguaggio Pig Latin*** semplifica la scrittura di sequenze di operazioni rispetto a MapReduce; infatti ci colloca tra **MapReduce** e **Hive**, come dimostra il codice necessario per contare le occorrenze di ciascuna parola in un file testuale.



> 30 righe di codice

~ 15 righe di codice

~ 5 righe di codice

Hadoop

- Hadoop è un **framework open source** per **l'elaborazione, la memorizzazione e l'analisi** di grandi quantità di dati distribuiti e non strutturati.

- E' stato progettato con l'obiettivo di poter **scalare da un singolo server a migliaia di macchine** con un alto livello di **tolleranza ai guasti**.

Hadoop

- **Hadoop** è stato **ispirato da MapReduce**, una *funzione "user-defined" sviluppata da Google* nei primi anni del 2000 per l'indicizzazione del web.

- **Hadoop** è ora un **progetto della Apache Software Foundation**, dove centinaia di collaboratori lavorano per migliorare continuamente la tecnologia di base.

Idea chiave

Invece di processare un **enorme blocco di dati** con una **singola macchina**, ***Hadoop rompe i Big Data in più parti*** *in modo che ogni parte possa essere elaborata e analizzata allo stesso tempo, con un* ***approccio di calcolo parallelo e distribuito***.

Hadoop – Caratteristiche chiave

Hadoop cambia l'economia e le dinamiche del calcolo computazionale su larga scala, definendo una **soluzione di elaborazione che è**:

Scalabile

- Nuovi **nodi possono essere aggiunti**, se necessario, **senza dover modificare** i *formati di dati*, le *modalità di caricamento dei dati*, come vengono *scritti i job*, o le applicazioni ad un livello superiore.

Performante

- Hadoop porta un **calcolo parallelo e massivo su dei commodity server**. Il risultato è una *riduzione consistente del costo per terabyte* dello spazio di archiviazione.
- La **ripartizione dei dati sui nodi** di calcolo permette di *minimizzare i tempi di accesso*, eliminando onerosi trasferimenti di rete.

Hadoop – Caratteristiche chiave

Hadoop cambia l'economia e le dinamiche del calcolo computazionale su larga scala, definendo una **soluzione di elaborazione che è**:

Flessibile

- Hadoop è **schema-less**, e in grado di ***gestire dati strutturati e non, provenienti da più fonti***. I dati provenienti da sorgenti multiple possono essere uniti ed aggregati in modi arbitrari che permettono un'analisi più approfondita di rispetto ai sistemi tradizionali.

Fault Tolerant

- Quando si perde un nodo, il sistema **reindirizza il lavoro** su un'altra macchina che possiede i dati (copia), garantendo una **elaborazione continua senza tempi di attesa**.

Hadoop – Componenti chiave

Le **componenti chiave** che costituiscono il nucleo della piattaforma **Hadoop** sono:

Hadoop common

- Strato di sw comune che ***fornisce funzioni di supporto agli altri moduli***.

HDFS

- E' il **filesystem distribuito** in grado di ***gestire dati in diversi formati*** e di ***regolarne l'accesso*** in modo efficace. ***Garantisce che i dati siano ridondanti*** sui nodi realizzando così la tolleranza ai guasti

Hadoop – Componenti chiave

Le **componenti chiave** che costituiscono il nucleo della piattaforma **Hadoop** sono:

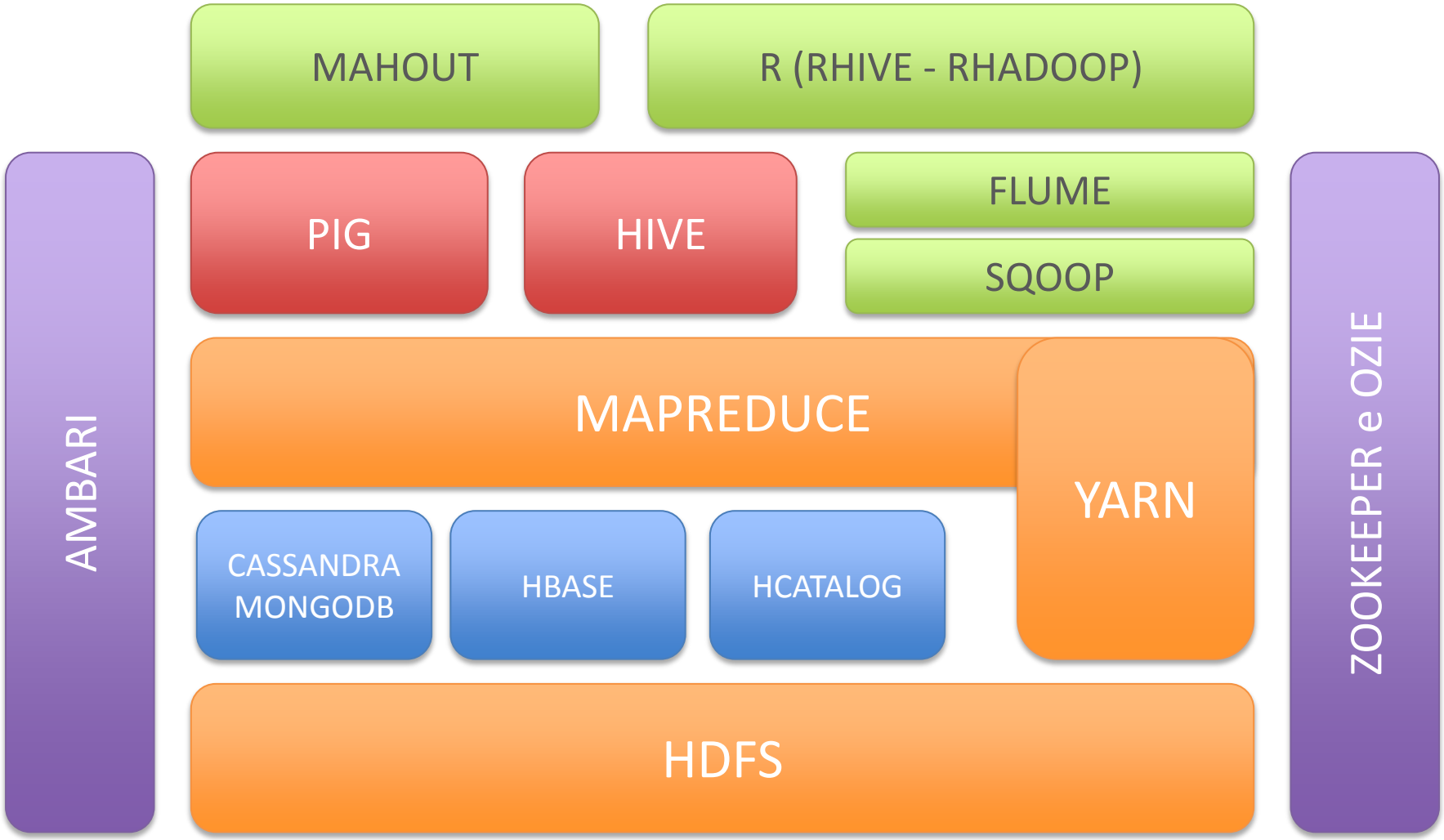
YARN

- Framework di supporto alla creazione di applicazioni o infrastrutture per il calcolo distribuito (presente nelle versioni 0.23.X e 2.X.X).

MapReduce

- E' il sistema di ***parallel processing*** di grandi quantità di dati. Lavora secondo il principio ***dividi et impera***, in cui un problema complesso viene suddiviso in sottoproblemi insieme ai dati, elaborati in modo separato.

Hadoop – Architettura (componenti aggiuntivi)



Hadoop – Architettura (componenti aggiuntivi)

PIG

- E' una **libreria** che semplifica la **creazione di job** per l'elaborazione dei dati. Fornisce il linguaggio ***Pig Latin***, e poi *Pig stesso opera la conversione in comandi MapReduce.*

Hive

- E' un sistema di data warehousing su Hadoop. Permette l'aggregazione e l'analisi di grandi dataset offrendo il linguaggio HiveQL (SQL-like).

Hcatalog

- Sistema di gestione di metadati che facilita la lettura/scrittura dei dati.

HBase

- Database *Column oriented* basato su HDFS

Hadoop – Architettura (componenti aggiuntivi)

Cassandra-MongoDB…

- Database NoSQL che comunicano o utilizzano Hadoop.

Zookeeper

- Strumento di coordinamento e sincronizzazione delle risorse.

Ozie

- Strumento di gestione del workflow delle operazioni.

Zookeeper

- Strumento di monitoraggio e amministrazione del cluster Hadoop.

Hadoop – Architettura (componenti aggiuntivi)

Flume

- Sistema per muovere grandi quantità di dati (come flussi), provenienti da fonti diverse verso HDFS o altre destinazioni.

Sqoop

- Libreria per trasferire dati tra database relazionali e Hadoop.

Mahout

- Libreria di *machine learning*.

R (con Rhive e RHadoop)

- Software statistico in grado di utilizzare Hadoop attraverso i plug-in Rhive e Rhadoop.

Hadoop

Uno dei principali obiettivi di Hadoop è quello di consentire la ***scansione di grandi dataset*** e produrre dei risultati attraverso un ***sistema di "batch-processing" distribuito*** e altamente scalabile.

- Apache Hadoop è caratterizzato da **due componenti chiave**:
  - ***HDFS*** (*Hadoop Distributed FileSystem*)

  - ***MapReduce***

Hadoop - HDFS

- ***HDFS*** è un **filesystem distribuito**, eseguito sul filesystem nativo.
- Nasce con l'obiettivo di ***soddisfare requisiti*** di **affidabilità** e **scalabilità**.
- Può gestire un **numero elevato di file**, anche di notevoli dimensioni (*nell'ordine di **GB** e **TB***), tramite **cluster** che possono contenere anche ***migliaia di nodi***.
- Può utilizzare ***commodity hardware.*** Ciò comporta la possibilità di *guasti frequenti e/o nodi non disponibili*.

Capacità di effettuare operazioni di recovery automatiche.

Dal punto di vista dell'architettura un *cluster Hadoop* è composto da diverse ***tipologie di nodi*** (processi che girano sui vari server)

- ***NameNode***
- ***DataNode***
- ***SecondaryNameNode***

HDFS - NameNode

- E' ***l'applicazione in esecuzione sul server principale***.
- Si occupa della **gestione del filesystem** e controlla l'accesso ai file.
- Gestisce il ***namespace,*** cioè l'*elenco* dei nomi **dei file** e **dei blocchi** (da *64 o 128 MB*) in cui sono suddivisi.
- Determina la **distribuzione dei blocchi sui vari DataNode** e le **strategia di replicazione dei file** in modo da garantire l'affidabilità del sistema.
- Controlla **lo stato e l'esecuzione dei vari nodi** del cluster.

HDFS - NameNode

Il NameNode salva il ***namespace*** su due file:

1. ***fsimage –*** è l'ultima immagine del namespace

2. ***journal –*** è il log dei cambiamenti avvenuti nel namespace dall'ultimo aggiornamento del fsimage.

- All'**avvio** il NameNode **unisce il file fsimage con il log dei cambiamenti**, in modo da *ottenere una fotografia aggiornata dello stato del cluster*.

- Questo **snapshot va poi a sovrascrivere l'fsimage** esistente.

HDFS - DataNode

I **DataNode** sono i **processi che girano sui vari nodi del cluster**.

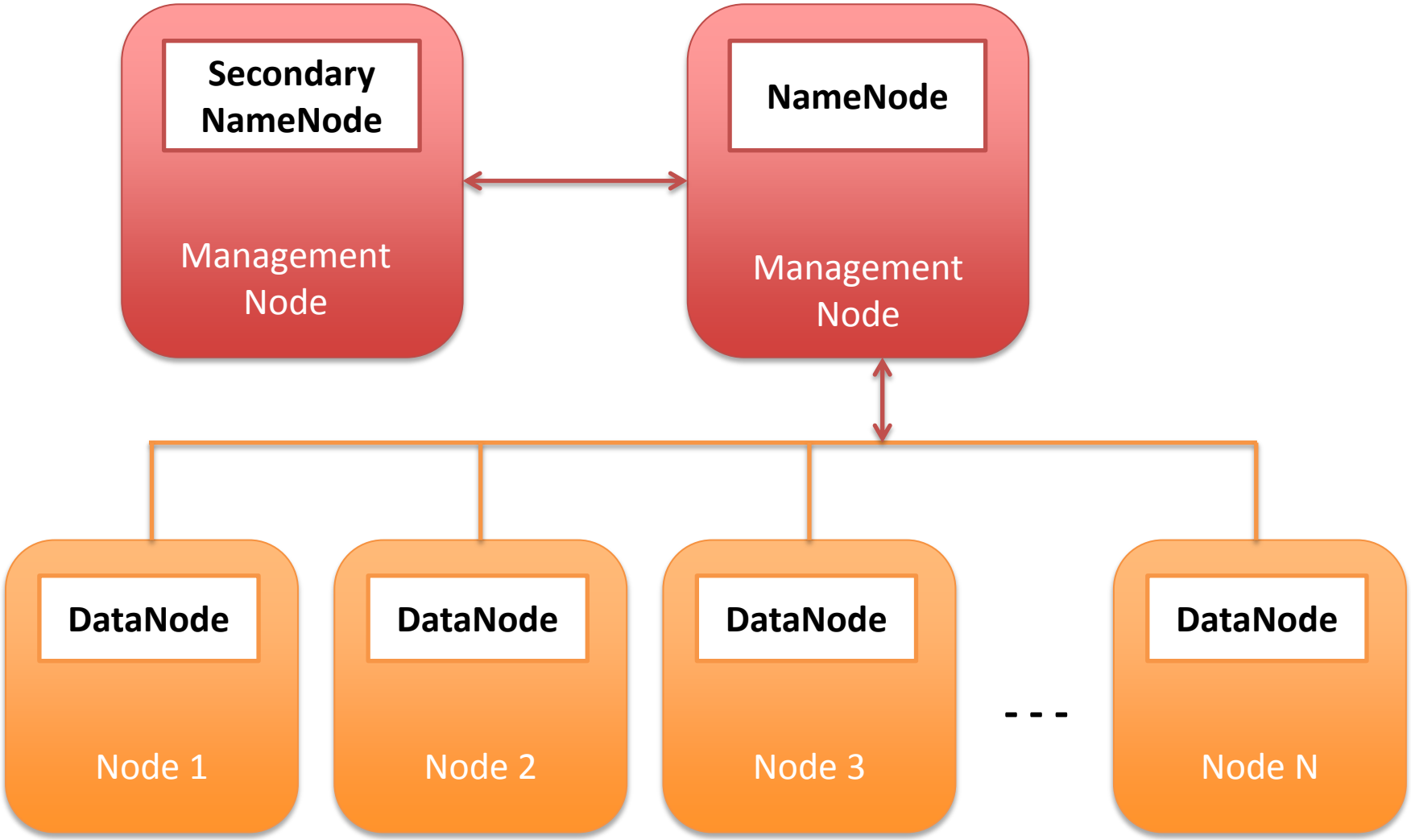
- Solitamente si ha **un DataNode per ogni nodo**.
- Gestiscono fisicamente lo storage di ogni nodo.
- **Eseguono le operazioni di lettura e scrittura richieste dai client**, occupandosi della *creazione, cancellazione e replica dei vari blocchi di dati*.

HDFS - SecondaryNameNode

Non deve essere confuso come un ***nodo di failover per il NameNode***.

- E' infatti un ***servizio di supporto*** **al NameNode** per far si che esso lavori in modo più efficiente.
- Questo processo **scarica periodicamente** il file *fsimage* e il *journal* (dal NameNode), li **unisce in un unico snapshot** che poi **restituisce al NameNode**.
- In alcune versioni questo processo è chiamato ***CheckPointNode***.

HDFS - Architettura



HDFS - Approfondimento

- Nelle ***versioni precedenti la 2.X.X***. il *NameNode era un'applicazione che girava su un solo server*. Ciò poteva portare a **problemi di indisponibilità**.

Soluzione: *due diversi server configurati come NameNode*, **uno attivo** e **l'altro in standby**. Come?

1. Utilizzo del ***Quorum Journal Manager,*** cioè un insieme di processi (JournalNode) a cui il NameNode comunica le modifiche. Il nodo in standby le legge e si mantiene sincronizzato.

2. Utilizzo di uno ***storage condiviso,*** su cui il nodo attivo salva le modifiche del HDFS. Il nodo in standby le legge mantenendosi sincronizzato.

HDFS – File e blocchi

- HDFS organizza i **file come sequenze di blocchi** di uguale **dimensione,** di solito ***64 o 128 MB***, *ridondanti su più nodi*.

- Per ciascun singolo file **è possibile configurare** sia il ***numero di repliche*** che la ***dimensione dei blocchi***.

- Il **meccanismo di replicazione** serve sia a ***garantire la disponibilità*** in caso di guasti, ma anche per ***recuperare i dati in modo più efficiente***.

- Infatti in HDFS per **rispondere ad una richiesta di lettura dati**, vengono **selezionate le repliche presenti sui nodi più vicini** al client che ha effettuato la richiesta.

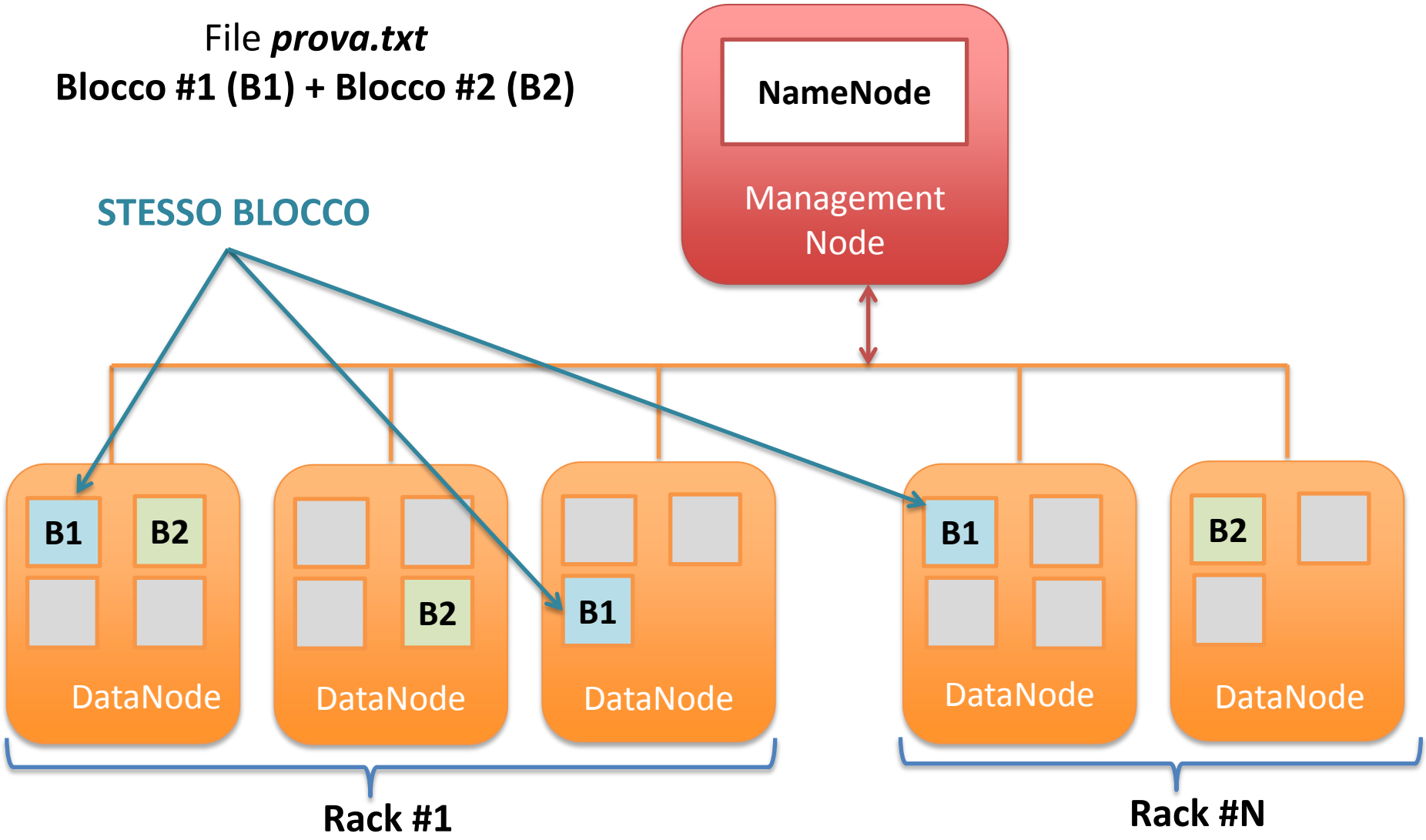
HDFS – File e blocchi

I **file *sono replicati su più macchine*** al momento del caricamento.

- Uno **stesso blocco è memorizzato** su un **numero multiplo di nodi**.
- Ciò ***favorisce*** le **operazioni di lettura** e **garantisce tolleranza ai guasti** (*disponibilità dei dati*).
- Il **valore di *"replicazione"*** di default è pari a **3**.
  - *Prima replica* sul **rack locale**
  - *Seconda replica* sempre sul **rack locale, ma su un nodo differente**.
  - *Terza replica* su un **rack differente**.

HDFS – File e blocchi

File ***prova.txt***
Blocco #1 (B1) + Blocco #2 (B2)



HDFS – Creazione dei file

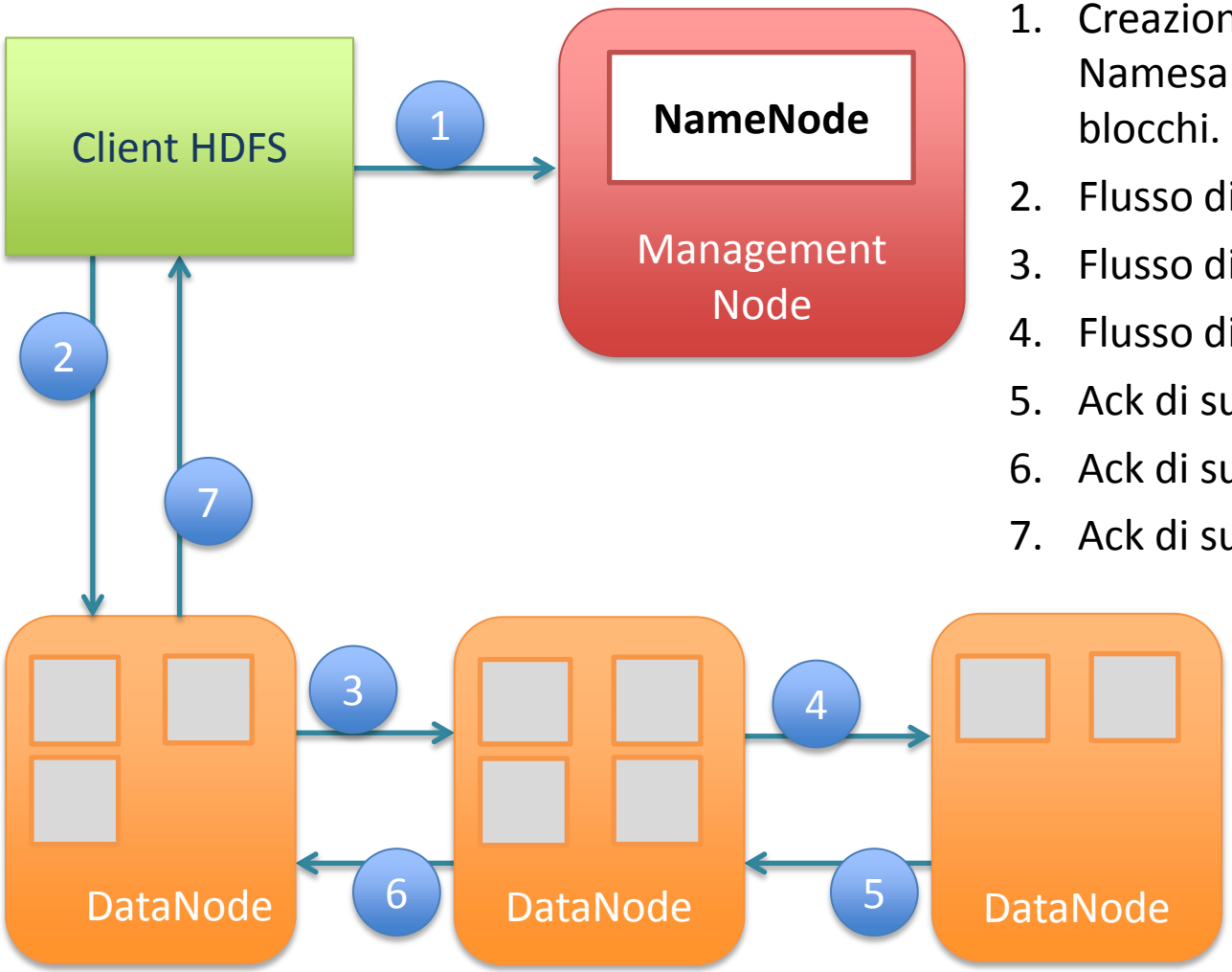
Un file *non viene creato* direttamente **attraverso il NameNode**.

1. Il ***client HDFS crea un file temporaneo in locale***, e solo quando la sua *dimensione è maggiore di quella di un blocco* sarà preso in carico dal **NameNode**.

2. Il **NameNode crea il file** nella gerarchia del filesystem, **identifica DataNode e i blocchi** in cui memorizzare il file.

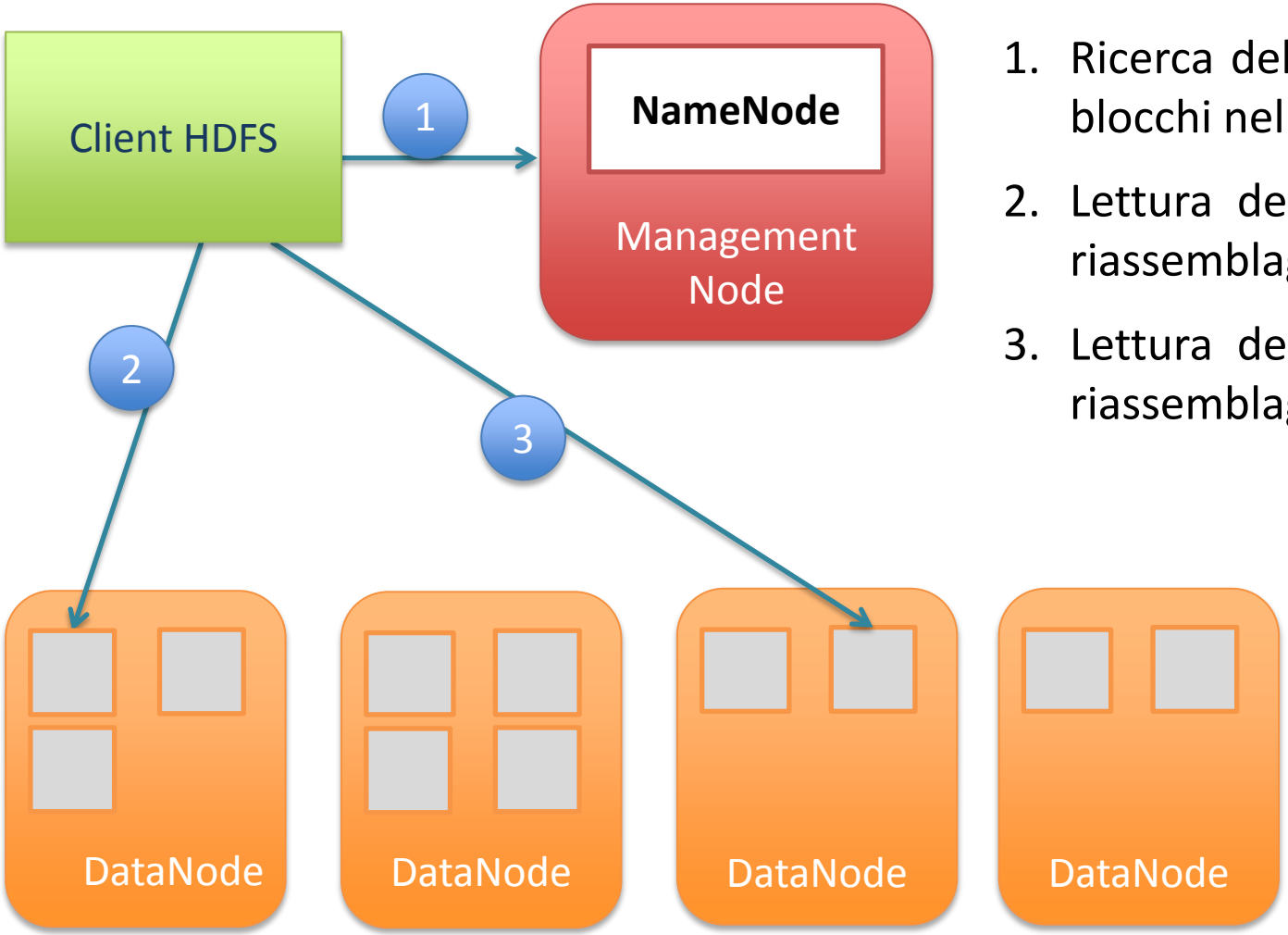
3. Il **client HDFS** riceve queste informazioni e ***provvede a copiare il file dalla cache locale alla locazione finale***.

HDFS – Scrittura di un file



1. Creazione di un nuovo file nel Namesapce, e individuazione dei blocchi.
2. Flusso di dati al primo Nodo
3. Flusso di dati al secondo Nodo
4. Flusso di dati al terzo Nodo
5. Ack di successo/fallimento
6. Ack di successo/fallimento
7. Ack di successo/fallimento

HDFS – Lettura di un file



1. Ricerca della posizione dei blocchi nel cluster.
2. Lettura dei vari blocchi e riassemblaggio del file
3. Lettura dei vari blocchi e riassemblaggio del file

Hadoop – Gli Archivi

Va ricordato che **HDFS ha dei problemi nel trattare file di piccole dimensioni** (inferiore a quella di un blocco).

- I **file utilizzano spazio nel Namespace**, cioè l'elenco di file gestito dal NameNode.
- **Namespace ha un limite di memoria** dovuto al server che ospita il NameNode.
- ***Problema –*** troppi file di dimensione inferiore al singolo blocco occuperebbero il Namespace, ma lo spazio del disco rimarrebbe parzialmente libero.

Hadoop – Gli Archivi

Soluzione – L'uso di **archivi Hadoop** risolve il problema. In questo modo **file di piccole dimensioni sono compattati in file più grandi** che possono essere acceduti in parallelo.

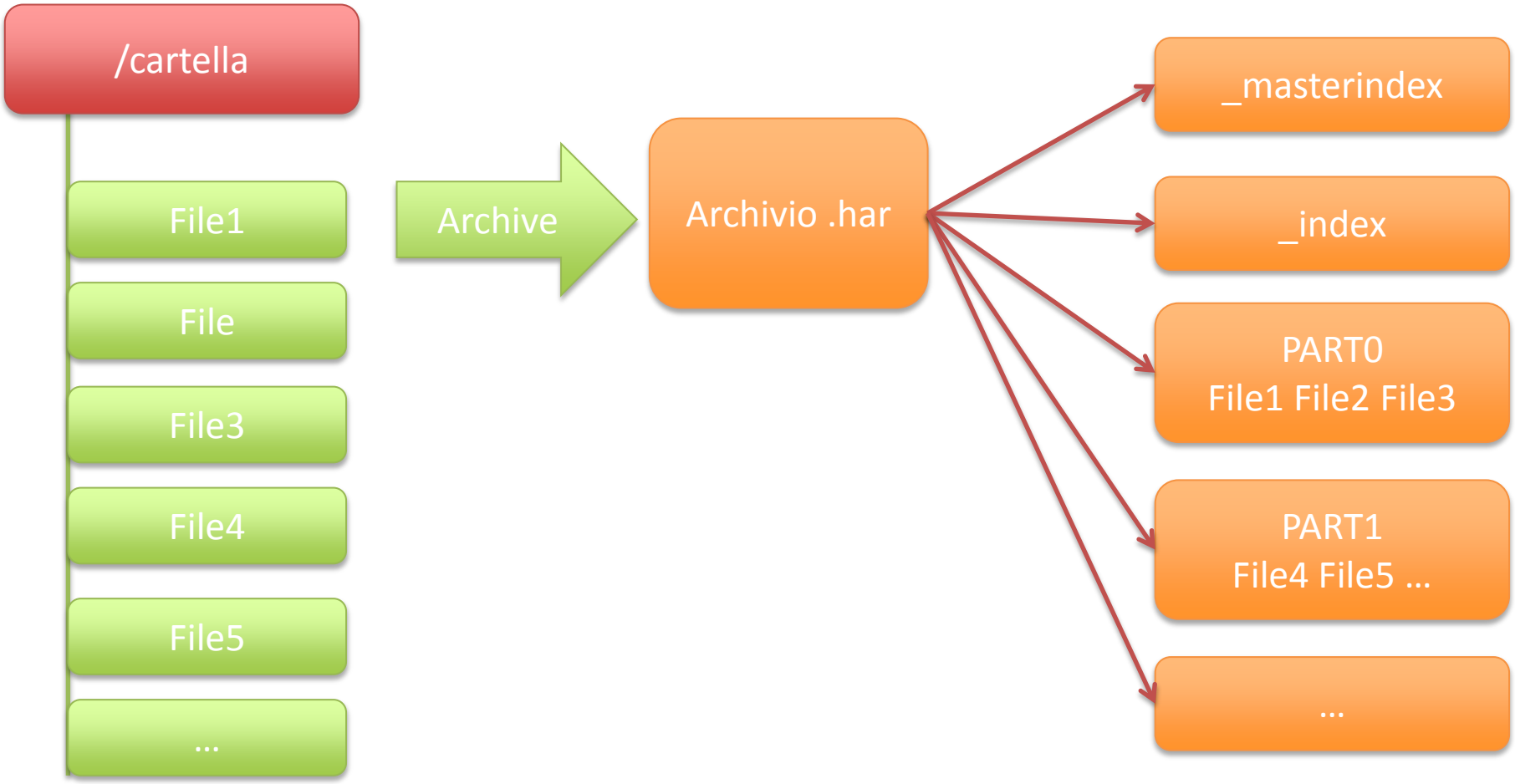
Gli archivi Hadoop si creano attraverso il comando `hadoop archive`, hanno estensione **.har**, e si compongono di tre parti:

1. ***Master index***, che contiene la posizione dei file originali nell'archivio.

2. ***Index,*** che contiene lo stato dei file.

3. ***Le parti,*** che contengono i dati.

Hadoop – Gli Archivi



HDFS – Accesso al Filesystem

L'accesso al filesystem e la sua gestione può avvenire in **due modi**:

- ***Tramite la shell dei comandi.***
- ***Tramite la Web API.***

Comando	Descrizione	Esempio
mkdir	Crea una cartella in uno specifico percorso.	hdfs dfs –mkdir hdfs://namenode/data/test_folder
rmr	Elimina le cartelle e le sottocartelle	hdfs dfs hdfs://namenode/data/test
copyFromLocal	Copia i file dal filesystem locale	hdfs dfs –copyFromLocal file_locale hdfs://namenode/data/test

Hadoop - MapReduce

- *MapReduce* **è l'elemento chiave del sistema di calcolo distribuito Hadoop**, *è il responsabile dell'elaborazione dei dati*.
- Framework per la creazione di **applicazioni che elaborano dati in parallelo** secondo il principio di ***functional programming***.
- MapReduce lavora secondo il principio ***dividi et impera***.

Un'operazione di calcolo è **suddivisa in sottoparti** ognuna delle quali è **processata in modo autonomo**. Quando ogni parte del problema è stata calcolata ***i risultati parziali vengono riassemblati o "ridotti".***

Hadoop - MapReduce

- *MapReduce* si occupa automaticamente della ***suddivisione dei vari task***, del ***monitoraggio*** e della loro ***ri-esecuzione*** in caso di malfunzionamenti.

- Questo framework **lavora in congiunzione con HDFS**, infatti i *nodi di calcolo si trovano in corrispondenza dei DataNode*.

- I vari *task sono quindi eseguiti li dove fisicamente sono memorizzati i dati*.

Aumento dell'efficienza di calcolo

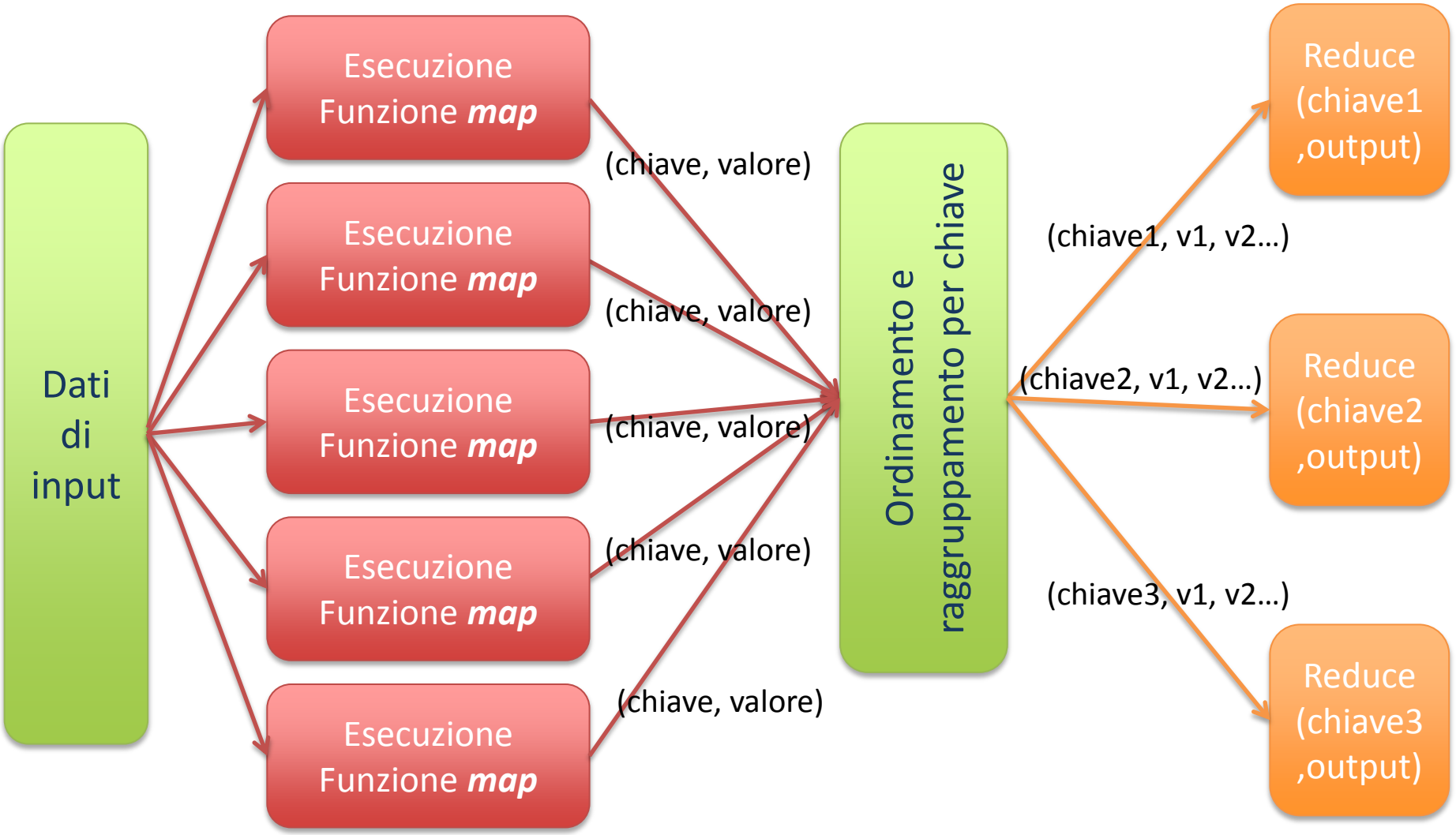
MapReduce - Job

In MapReduce le elaborazioni sono definite ***"job"***, ognuno dei quali è composto da:

- I ***dati di ingresso***, presenti du HDFS.
- *Una funzione* ***map***, che converte i dati in un insieme di coppie *chiave/valore.*
- *Una funzione* ***reduce***, che elabora i valori associati a ciascuna chiave e crea in output una o più coppie chiave valore.
- Il **risultato**, che viene scritto su un file su HDFS.

Osservazione: prima di eseguire la funzione *reduce*, c'è una fase in cui ***le coppie chiave valore sono ordinate*** per chiave e ***i valori appartenenti alla stessa chiave, sono raggruppati***.

MapReduce - Job



MapReduce - Componenti

Il *framework MapReduce* è costituito da **due componenti chiave**:

1. ***JobTracker –*** l'elemento che si occupa di **gestire le risorse** (CPU e memoria) e il **ciclo di vita del job**. Si occupa di **distribuire i vari task** tra i nodi (secondo un criterio di vicinanza dei dati) e di **rieseguirli nel caso in cui si verificassero degli errori**.

1. ***TaskTracker –*** sono gli elementi che **girano sui vari nodi** del cluster **responsabili dell'esecuzione dei task** sotto le direttive del JobTracker.

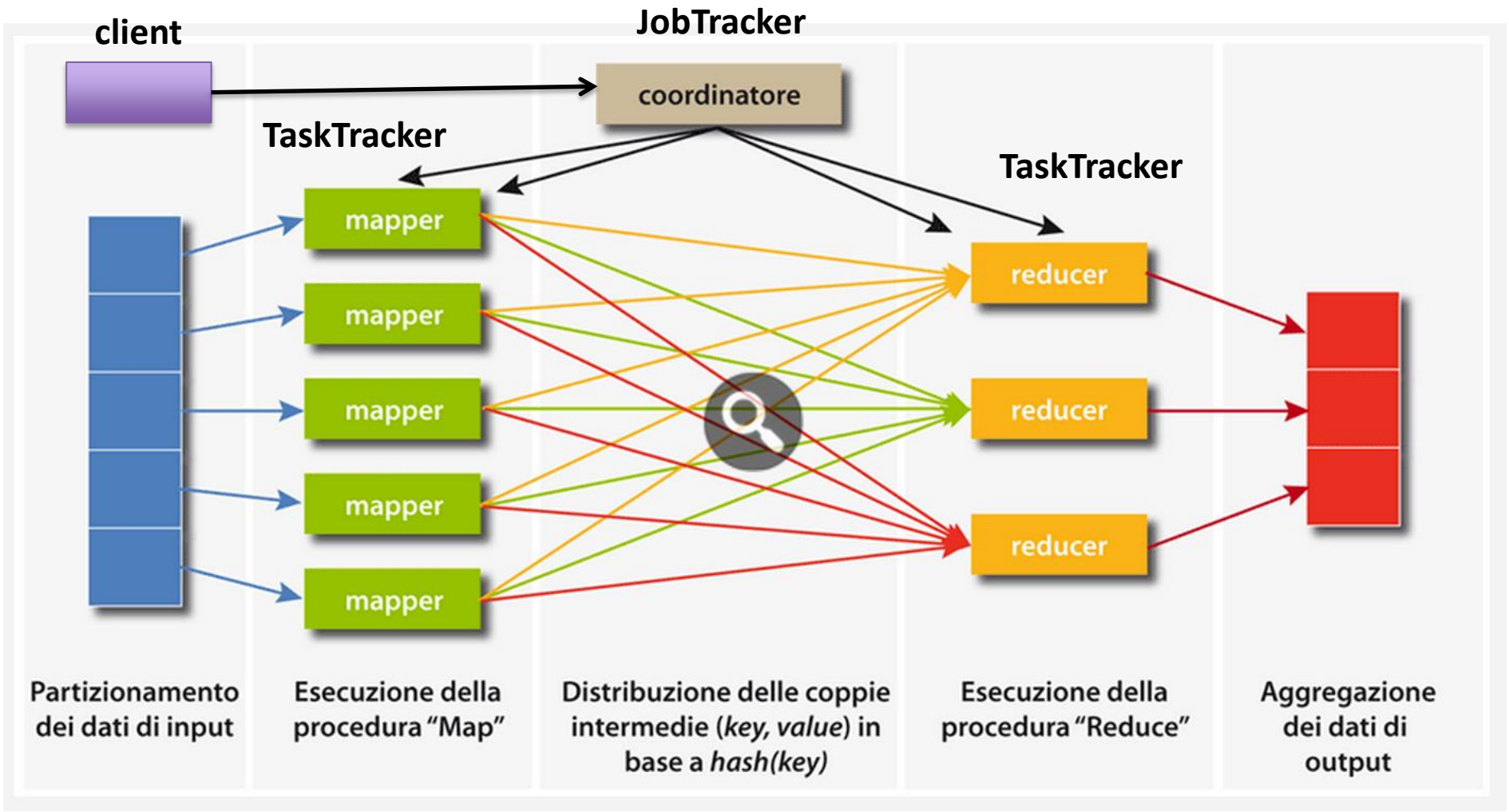
MapReduce - Funzionamento

Le applicazioni che sfruttano MR devono specificare: ***file di input***, ***file di output***, ***funzioni di map*** e ***reduce*** (contenute nel job).

Il client Hadoop comunica il job (come archivio .jar) al JobTracker, che si occupa di distribuire l'esecuzione sui vari nodi.

- Il **JobTracker** determina il **numero di parti in cui è suddiviso l'input**, attivando un certo **numero di TaskTracker** in base alla vicinanza.
- I **TaskTracker estraggono i dati e attivano la funziona *map*** che genera ***coppie chiave/valore***.
- Finita la fase di map, il **JobTracker attiva la fase di reduce** in cui i TaskTracker prima ordinano i risultati per chiave e poi li "riducono".
- Infine i vari **file di output prodotti saranno aggregati** in un unico risultato.

MapReduce - Funzionamento



MapReduce – Scrivere un Job

Scrivere un **Job MapReduce** non è un'operazione banale; consiste infatti nella creazione di **tre classi Java**: ***mapper***, ***reducer*** e ***driver***.

mapper

- Questa classe estende la classe base *MapReduceBase* e implementa l'interfaccia *Mapper.*
- La maggior parte del lavoro è svolto dal metodo ***map.***

reducer

- Questa classe estende la classe base *MapReduceBase* e implementa ovviamente l'interfaccia *Reducer.*
- La maggior parte del lavoro è svolto dal metodo ***reduce.***

MapReduce – Scrivere un Job

Scrivere un **Job MapReduce** non è un'operazione banale; consiste infatti nella creazione di **tre classi Java**: ***mapper***, ***reducer*** e ***driver***.

driver

- Questa classe ha il **compito di inizializzare il job**, di **passare i parametri di input** e **definire la posizione in cui sarà memorizzato l'output**.

OSSERVAZIONE – vi è un'ulteriore classe che si va a collocare nel workflow tra i *mapper* e i *reducer*, la classe ***combiner.***

- Questa classe è **simile a reducer**, ma limitatamente ad un solo nodo; cioè **si occupa di aggregare i dati elaborati su un nodo prima di passarli ad un altro** con l'obiettivo di ridurre il traffico di rete.

Hadoop – Installazione

Un cluster Hadoop può essere installato in **tre differenti modalità**:

1. **Modalità Standalone (Locale)**

è la configurazione di default, Hadoop consiste in un singolo processo Java in esecuzione.

2. **Modalità Pseudo-Distribuita**

Il cluster è installato su un singolo nodo su cui sono eseguiti più daemon Hadoop, ognuno su uno specifico processo Java.

3. **Modalità Fully-Distributed**

E' la configurazione più interessante, il cluster è costituito da un insieme di nodi fisici (collegati in rete) su ciascuno dei quali è eseguito un processo Hadoop.

Hadoop – Installazione Fully-Distributed

Configurazione preliminare della macchina

- Su ciascun macchina bisogna modificare il file ***hosts*** in modo creare un *mapping tra gli indirizzi IP e il rispettivo hostname* di ogni nodo del cluster.

```
sudo gedit /etc/hosts
```

- Esempio di codice da inserire

```
192.168.0.72   hadoopmaster
192.168.0.71   hadoopnode01
192.168.0.69   hadoopnode02
……………………………………………………
```

Configurazioni comuni *Master e Slaves*

Per poter funzionare correttamente Hadoop richiede che siano installati e/o configurati i seguenti componenti:

- **Installazione di Java (la versione 1.7.065 è OK)**
- **Installazione e configurazione di SSH**
- **Disabilitazione di IPV6**

Installazione di Java v1.7

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java7-installer
```

- Controllo della versione installata

```
java -version
```

- Settaggio automatico delle variabili d'ambiente

```
sudo apt-get install oracle-java7-set-default
```

Configurazione accesso SSH

- Verificare se openssh-server è installato

```
dpkg –l | grep openssh-server
```

- Se non è installato, eseguire l'installazione con il comando

```
sudo apt-get install openssh-server
```

- Generare la chiave SSH

```
ssh-keygen –t rsa –P “”
```

- Abilitare l'accesso SSHcon la nuova chiave creata

```
cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

- Testare l'accesso con il comando -> ***ssh localhost***

Disabilitazione di ipv6

- Utilizzando l'account di root eseguire il comando

```
sudo gedit /etc/sysctl.conf
```

- Aggiungere le seguenti linee di codice e riavviare la macchina per fare in modo che i cambiamenti abbiano effetto.

```
#disable ipv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

Hadoop – Installazione Fully-Distributed

Installazione Hadoop

- Scaricare la versione di Hadoop desiderata, estrarla nella cartella /home/bigdata e rinominarla con il nome hadoop

- Al suo interno creare le seguenti cartelle

tmp
hdfs
hdfs/name
hdfs/data
hdfs/namesecondary

- Passare poi alla configurazione specifica dei vari file per il nodo Master e per i vari slave.

Hadoop – Installazione Fully-Distributed

Configurazione nodo Master

core-site.xml

- Contiene le opzioni di configurazione principali, come le proprietà I/O

mapred-site.xml

- Contiene le impostazioni dei processi JobTracker e TaskTracker

hdfs-site.xml

- Contiene le impostazioni per i processi NameNode, DataNode e Secondary NameNode

hadoop-env.sh

- Specifica le variabili d'ambiente (percorso Java e Log)
- E' utilizzato dagli script che eseguono e gestiscono Hadoop

Configurazione nodo Master

$HOME/.bashrc

- Contiene le variabili d'ambiente definite per Java e per i vari componenti/processi di Hadoop

masters

- Contiene l'indirizzo IP o l'Hostname del nodo designato come master

slaves

- Contiene gli indirizzi IP o gliHostname dei vari nodi designati come slave del cluster.

Hadoop – Installazione Fully-Distributed

Configurazione nodo Slave

core-site. xml

- Contiene le opzioni di configurazione principali, come le proprietà I/O

mapred-site.xml

- Contiene le impostazioni dei processi JobTracker e TaskTracker

hdfs-site.xml

- Contiene le impostazioni per i processi NameNode, DataNode e Secondary NameNode

hadoop-env.sh

- Specifica le variabili d'ambiente (percorso Java e Log)
- E' utilizzato dagli script che eseguono e gestiscono Hadoop

Configurazione nodo Slave

$HOME/.bashrc

- Contiene le variabili d'ambiente definite per Java e per i vari componenti/processi di Hadoop

masters

- Contiene l'indirizzo IP o l'Hostname del nodo designato come master

Hadoop – Avvio del Cluster

Esecuzione di Hadoop (intero cluster)

- Alla prima esecuzione l'HDFS deve essere formattato, con il comando ***hadoop namenode –format***

  - Avvio del filesystem -> ***start-dfs.sh***
  - Arresto del filesystem -> ***stop-dfs.sh***
  - Verifica dell'esecuzione -> ***jps (da shell)***

  - Avvio mapreduce -> ***start-mapred.sh***
  - Arresto mapreduce -> ***stop-mapred.sh***

Hadoop – Accesso Web Interface

Interfaccia Web di Hadoop

- Accedendo a ***http://nomemaster:50070*** si arriva alla user interface del NameNode da cui si possono ricavare alcune informazioni sullo stato del cluster.

- Accedendo a ***http://nomemaster:50030*** si raggiunge l'interfaccia per la visualizzazione dei vari Job in esecuzione/completati/falliti con i relativi dettagli.

Hadoop – NameNode Web Interface

NameNode 'node0:54310'

Started:	Mon Mar 11 08:19:57 PDT 2013
Version:	1.0.4, r1393290
Compiled:	Wed Oct 3 05:13:58 UTC 2012 by hortonfo
Upgrades:	There are no upgrades in progress.

Browse the filesystem
Namenode Logs

Cluster Summary

791 files and directories, 706 blocks = 1497 total. Heap Size is 26.38 MB / 966.69 MB (2%)

Configured Capacity	:	56.81 GB
DFS Used	:	775.55 MB
Non DFS Used	:	12.48 GB
DFS Remaining	:	43.57 GB
DFS Used%	:	1.33 %
DFS Remaining%	:	76.69 %
Live Nodes	:	3
Dead Nodes	:	0
Decommissioning Nodes	:	0
Number of Under-Replicated Blocks	:	1

Hadoop – NameNode Web Interface

Dettagli

- ***Configured Capacity:*** Mostra lo spazio totale a disposizione.
- ***DFS Used:*** è lo spazio occupato per i dati presenti nel filesystem distribuito.
- ***Non DFS Used:*** è lo spazio in locale delle varie macchine occupato da hadoop, dati però non accessibili dall'HDFS.
- ***Live Nodes:*** Numero di nodi attivi e funzionanti al momento.
- ***Dead Nodes:*** Numero di nodi non attivi (spenti o non ben configurati), non raggiungibili da Hadoop
- ***Decommissioning Nodes:*** Numero di nodi impostati come non attivi per nostra scelta.

Hadoop – Avvio singolo nodo

Esecuzione di Hadoop (singolo nodo)

- Avvio del filesystem datanode-> ***hadoop-daemon.sh start datanode***
- Arresto del filesystem datanode -> ***hadoop-daemon.sh stop datanode***
- Verifica dell'esecuzione -> ***jps (da shell)***

- Avvio mapreduce tasktracker-> ***hadoop-daemon.sh start tasktracker***
- Arresto mapreduce tasktracker-> ***hadoop-daemon.sh stop tasktracker***