Hadoop

A distributed framework for Big Data

Outline

- Caratteristiche di Hadoop
- HDFS
- YARN
- MapReduce

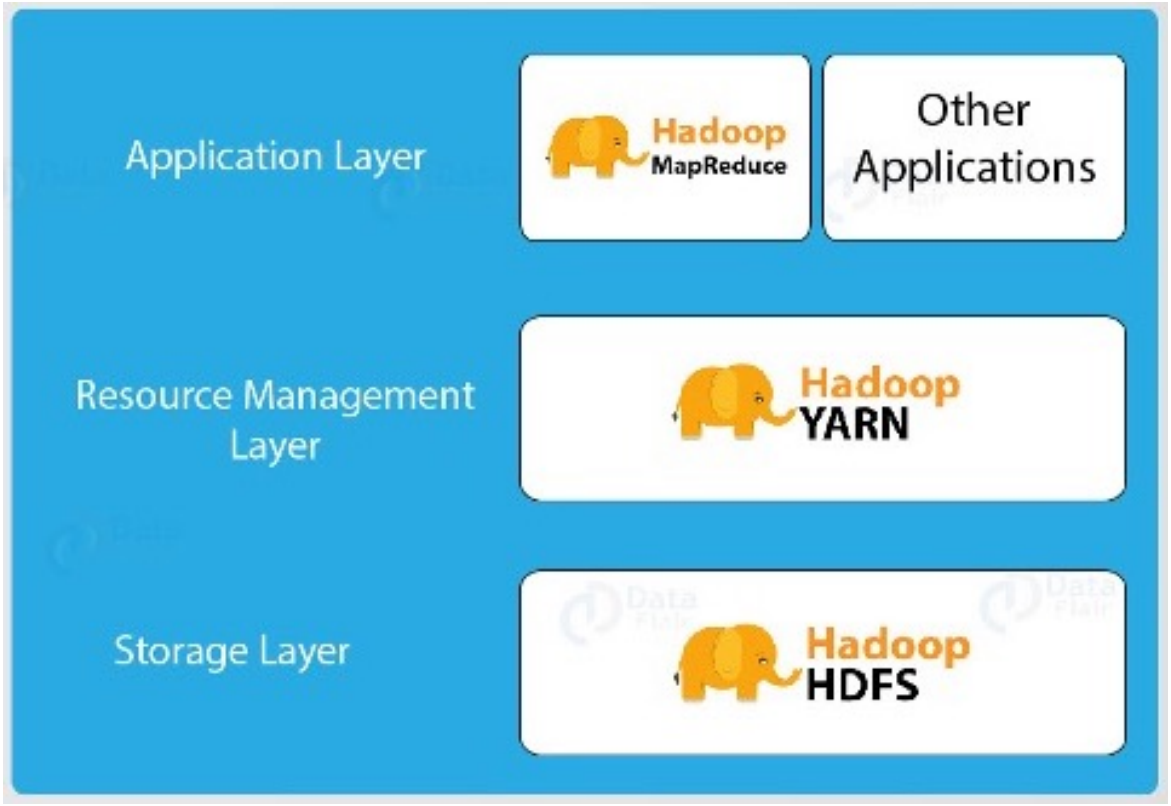
Cos’è Hadoop

- Progetto Apache: implementazione open source di un framework per elaborazione e archiviazione dati affidabile, scalabile e distribuito.
  - Apache Hadoop consente l'elaborazione distribuita di grandi set di dati su cluster di computer utilizzando semplici modelli di programmazione.
  - Progettato per scalare da singoli server a migliaia di macchine, ognuna delle quali offre calcolo e archiviazione locali.
    - Piuttosto che affidarsi all'hardware per fornire un'elevata disponibilità, la libreria stessa è progettata per rilevare e gestire i guasti a livello dell'applicazione, fornendo così un servizio ad alta disponibilità su un cluster di computer, ognuno dei quali può essere soggetto a guasti.

Caratteristiche di Hadoop

- Architettura software per i Big Data orientata nativamente all'elaborazione batch.
  - File system distribuito – HDFS
  - Resource manager – YARN
  - Batch processor – MapReduce
- Interfaccia con una serie di altri componenti software, sempre sviluppati da Apache, che ne estendono le funzionalità verso diversi campi di applicazione:
  - Stream processing
  - Workflow management
  - Interazione con RDBMS
  - Machine learning
  - …
- Si parla di «ecosistema Hadoop» intendendo l'insieme di tutte queste componenti

Architettura generale



Importanza di Hadoop

- **Capacità di archiviare e processare un enorme volume di dati di qualsiasi tipo, velocemente.** Caratteristica fondamentale per gestire il costante aumento dei volumi e la varietà dei dati, in particolare social media e IoT.

- **Computazione potente.** I modelli computazionali distribuiti di Hadoop processano i big data velocemente. Maggiori sono i computing nodes che si utilizzano e maggiore sarà la potenza di elaborazione.

- **Tolleranza ai guasti.** I dati e le loro elaborazioni sono protetti dai guasti hardware. Se un nodo si guasta, le attività vengono automaticamente indirizzate ad altri nodi per assicurare che il distributed computing funzioni. Ogni dato viene automaticamente salvato in più copie.

Importanza di Hadoop

- **Flessibilità**. A differenza dei database tradizionali, non c'è bisogno di elaborare i dati prima di memorizzarli. Puoi memorizzare tutti i dati che vuoi e decidere in seguito come utilizzarli. Inclusi i dati non strutturati, come testi, immagini e video.

- **Low cost**. Il framefork open-source è gratuito e utilizza commodity hardware per archiviare grandi volumi di dati.

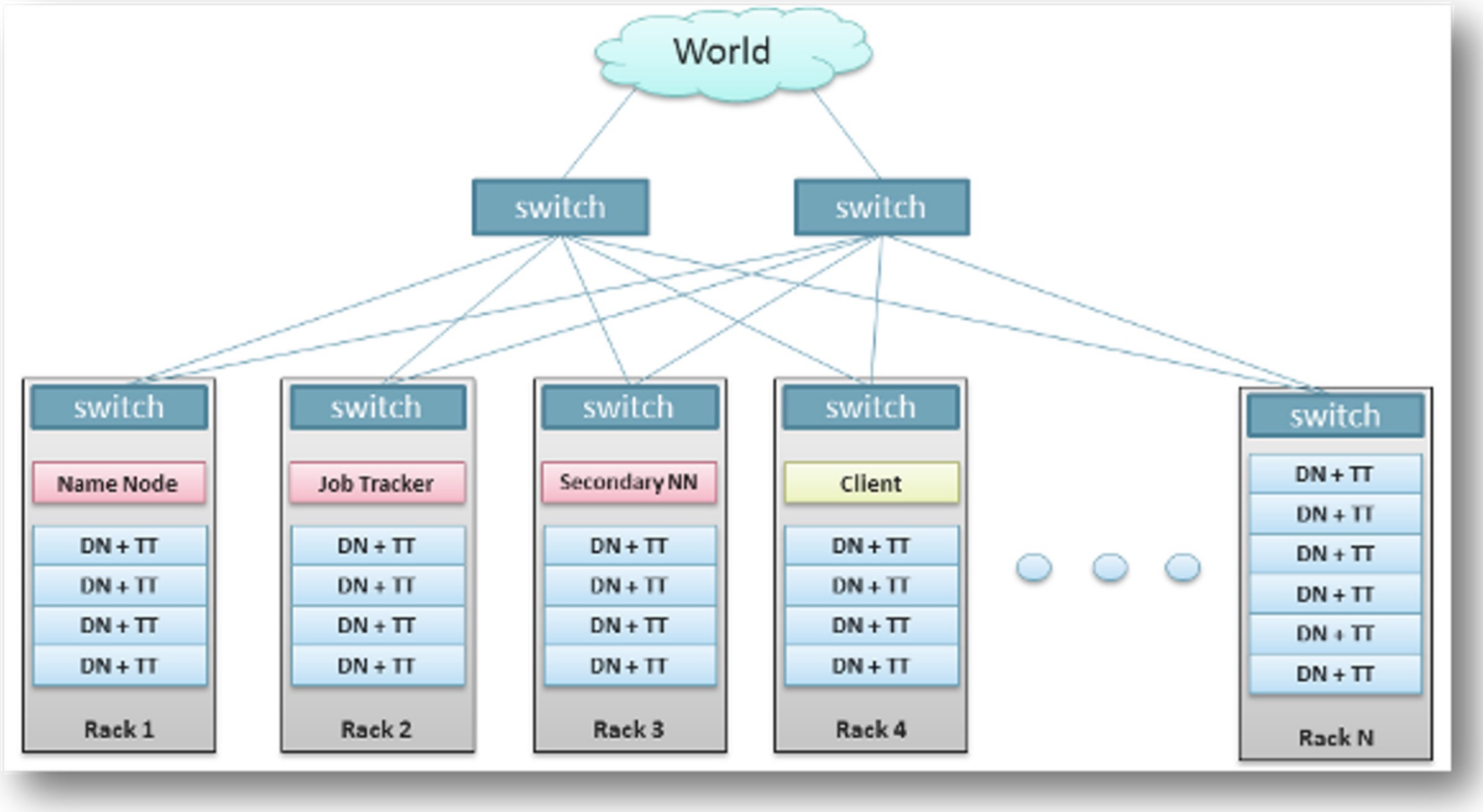
- **Scalabilità**. Puoi facilmente aumentare i dati gestiti dal tuo sistema solo aggiungendo dei nuovi nodi. L'amministrazione richiesta è davvero minima.

Architettura

- Distribuita, con una certa centralizzazione
- I nodi principali del cluster sono dove risiede la maggior parte della potenza di calcolo e dell'archiviazione del sistema
- I nodi principali eseguono *TaskTracker* per accettare e rispondere alle attività di *MapReduce* e anche *DataNode* per memorizzare i blocchi necessari il più vicino possibile
- Il nodo di controllo centrale esegue *NameNode* per tenere traccia di directory e file HDFS e *JobTracker* per inviare attività di calcolo a *TaskTracker*

- Scritto in Java, supporta anche Python e Ruby

Architettura



HDFS

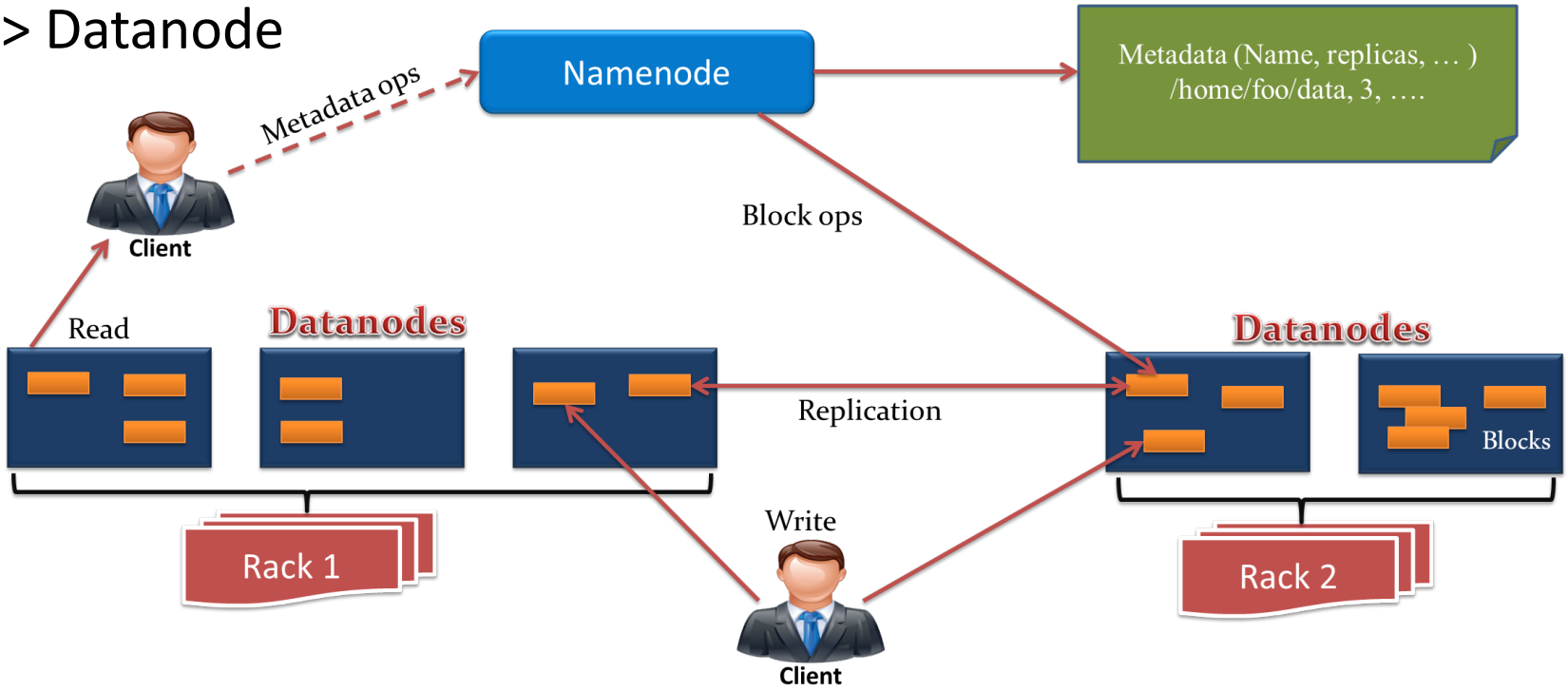
- **Hadoop Distributed File System** (in sigla HDFS) è un file system distribuito, portabile e scalabile scritto in Java per il framework Hadoop.
- Un cluster in Hadoop tipicamente possiede uno o più NameNode (su cui risiedono i metadati dei file) e un insieme di DataNode (su cui risiedono, in blocchi di dimensione fissa, i file dell'HDFS).
- Filosofia HDFS: conservare pochi grandi frammenti di un file nell'infrastruttura, piuttosto che molti frammenti piccoli.
- I formati più usati per i file su HDFS sono comma-separated values, Apache Avro, Apache ORC e Apache Parquet.

Caratteristiche HDFS

- Fault tolerance
- Elaborazione ad elevato thoroughput
- Grandi quantità di dati
- Write-once-read-many
- Computazione vicino ai dati
- Portabilità
- Su misura per le esigenze di MapReduce
- Alto grado di replica dei dati (3x per impostazione predefinita)
- Non c'è bisogno di RAID sui nodi normali
- Grande blocksize (64/128 MB)

Architettura HDFS

- Master -> Namenode
- Slave -> Datanode



NameNode

- Registro delle transazioni per l'eliminazione/aggiunta di file, ecc.
- Memorizza i metadati per i file, come la struttura di directory di un tipico FS.
  - **FsImage** – l'intero spazio dei nomi del file system è conservato come una sorta di «file immagine» nel file system locale del Namenode
    - Contiene una rappresentazione serializzata di tutte le cartelle e gli inode del file system dove stanno effettivamente i metadati
  - **EditLogs** – contiene tutte le modifiche recenti del file system sulla versione più recente di **FsImage**
    - In questo file sono registrate tutte le richieste di create/update/delete che vengono dai Datanode
- Regola l'accesso ai file da parte dei client
- Esegue le operazioni di file system: apertura, chiusura, ridenominazione di file e cartelle ...
- Gestisce le segnalazioni da parte dei datanode:
  - **Heartbeat** cioè la segnalazione di datanode attivo
  - Block report che contiene la lista dei datanode attivi
- Definisce il replication factor per tutti i blocchi
- Gestisce la creazione di più blocchi di replica quando necessario dopo un errore di DataNode

NameNode SPOF?

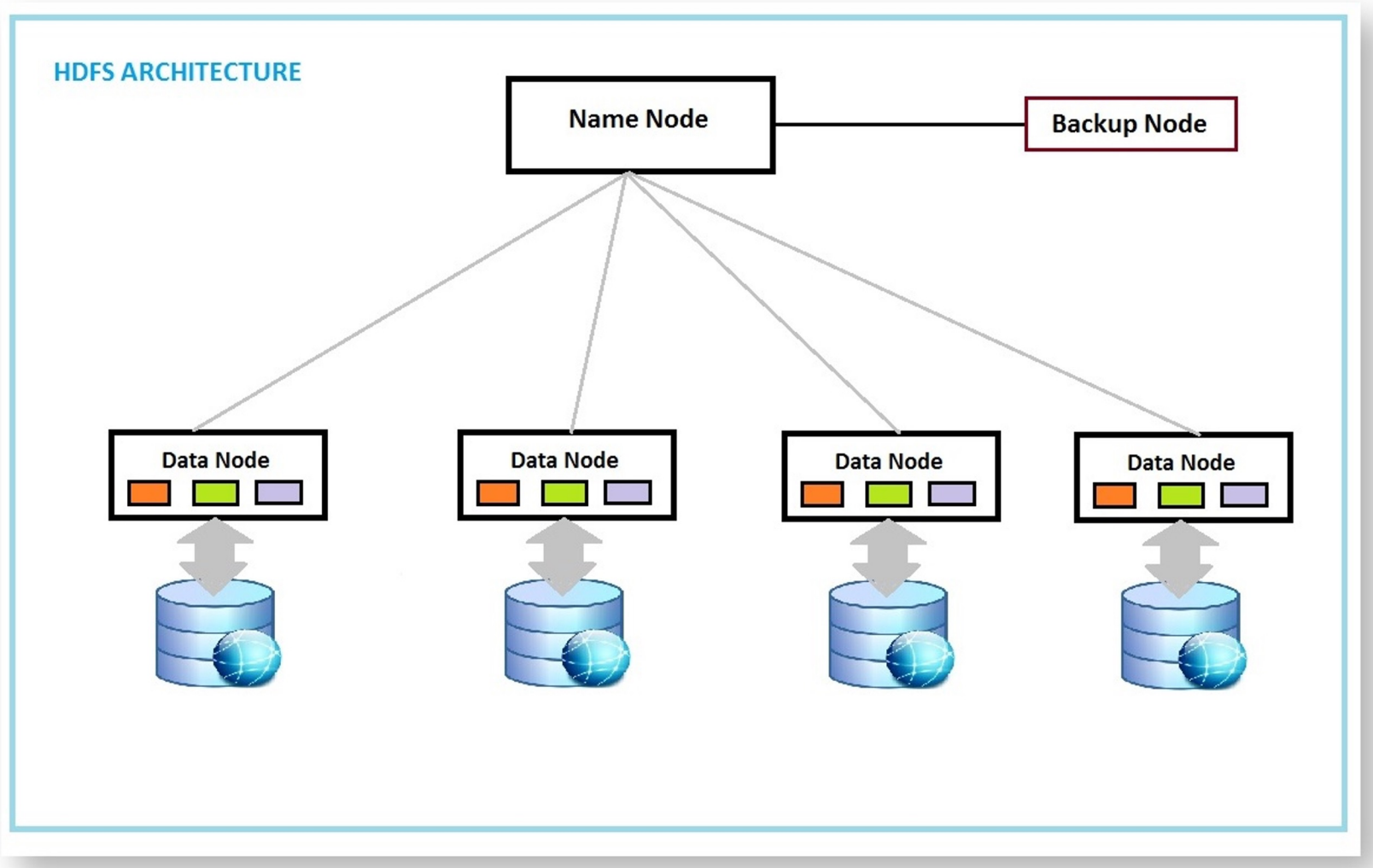
- Namenode secondario: si utilizza anche per accelerare i tempi di restart del Namenode, in caso di fallimento
  - A startup, il Namenode legge lo stato di HDFS da FsImage e poi applica gli edit contenuti in EditLogs
  - Esegue un merge dei due file che può essere molto lungo per effetto delle loro dimensioni
- Il Namenode secondario effettua il merge per suo conto, conservandolo su storage persistente
- Si sincronizza regolarmente con il Namenode

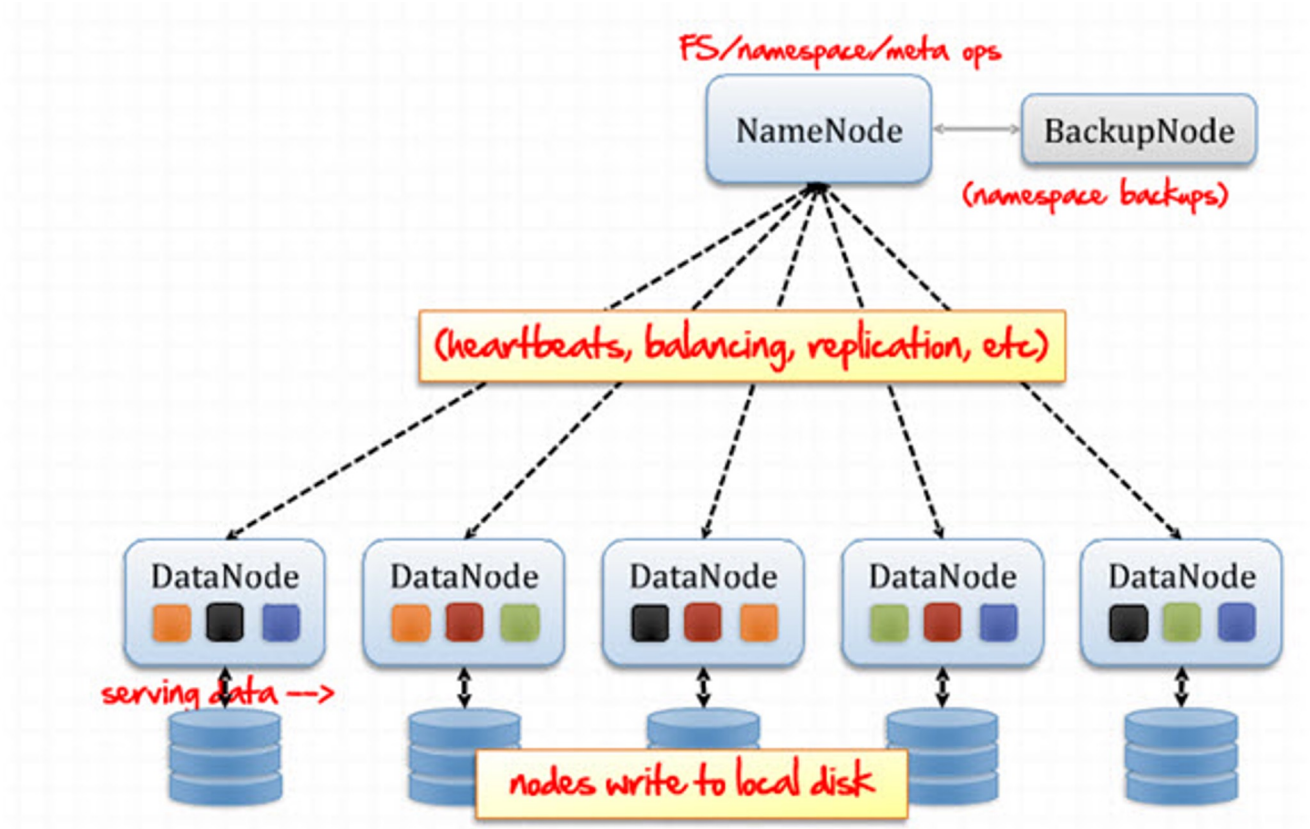
DataNode

- Lo slave dell'architettura
- Memorizza i dati effettivi in HDFS
- Può essere eseguito su qualsiasi filesystem sottostante (ext3/4, NTFS, ecc.)
- Notifica a NameNode quali blocchi gestisce
- Ogni 3 sec invia il segnale di heartbeat al Namenode
- Esegue le operazioni di read/write, creazione delle repliche dei blocchi e loro cancellazione

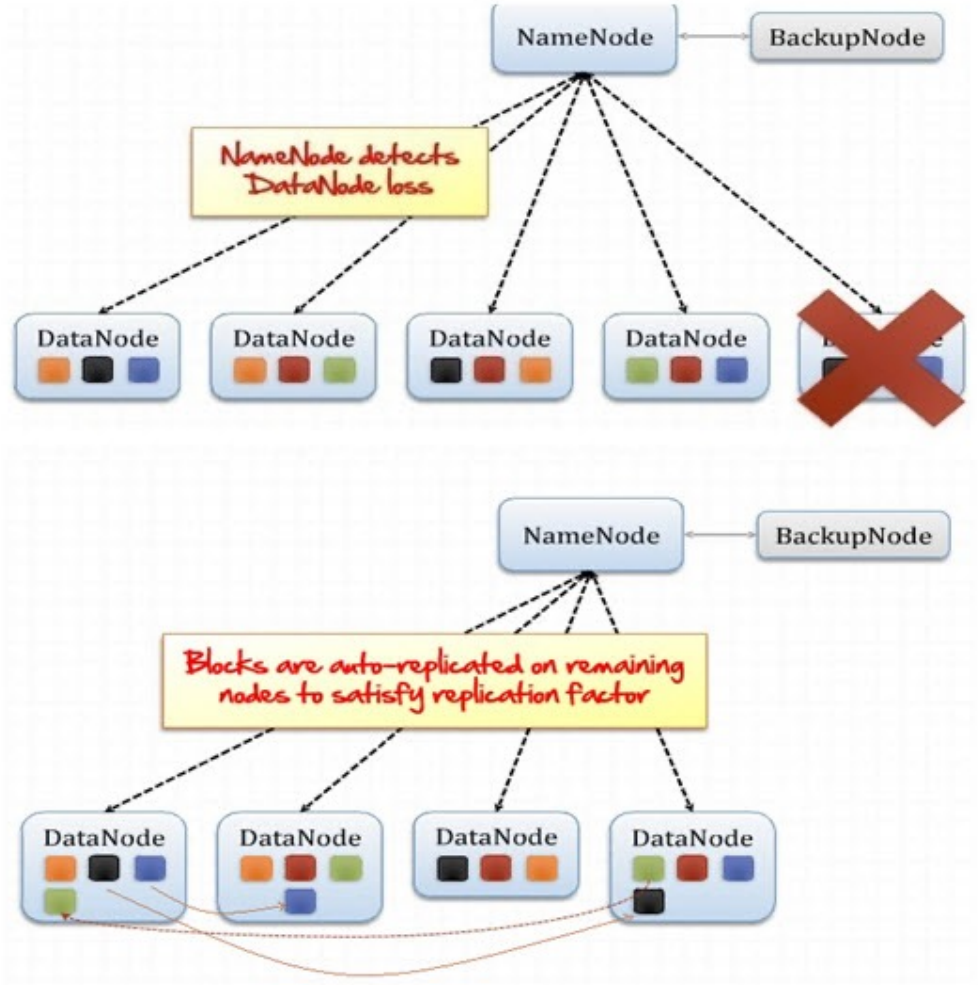
Rack awareness

- Un rack è un cluster di macchine fisiche su cui è fatto il deploy di un insieme di datanode
- L'algoritmo di rack awareness è un criterio di scelta del datanode più vicino da parte del Namenode per creare un blocco
- Serve a limitare l'impegno di banda passante sulla rete
- Riduce i costi delle operazioni di read/write
- La prima replica va sul nodo locale che l'ha creata, la seconda su un altro datanode del rack e la terza su un datanode di un altro rack

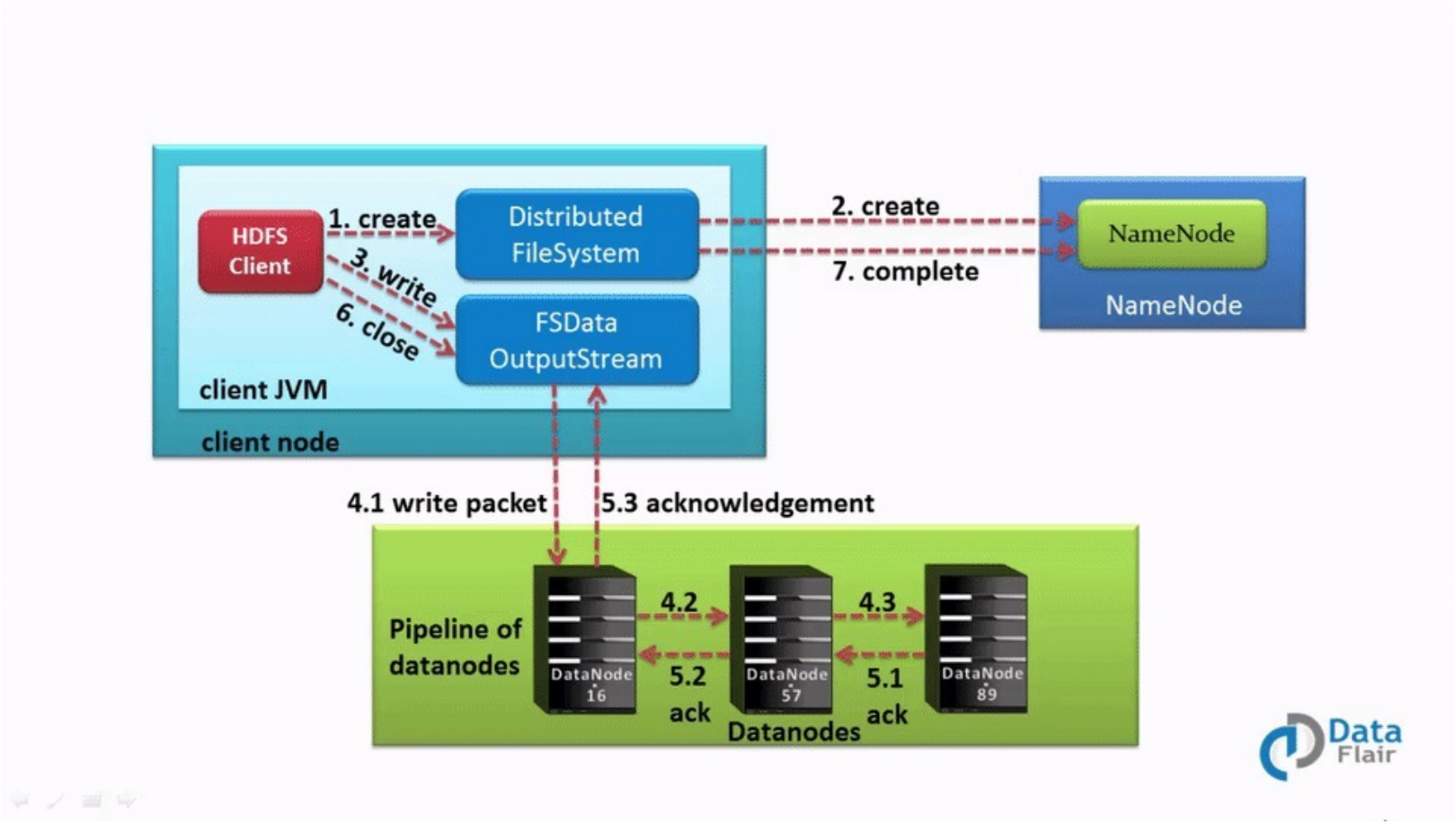

HDFS ARCHITECTURE
Name Node
Backup Node
Data Node
Data Node
Data Node
Data Node


FS/namespace/meta ops
NameNode
BackupNode
(namespace backups)
(heartbeats, balancing, replication, etc)
DataNode
DataNode
DataNode
DataNode
DataNode
serving data -->
nodes write to local disk

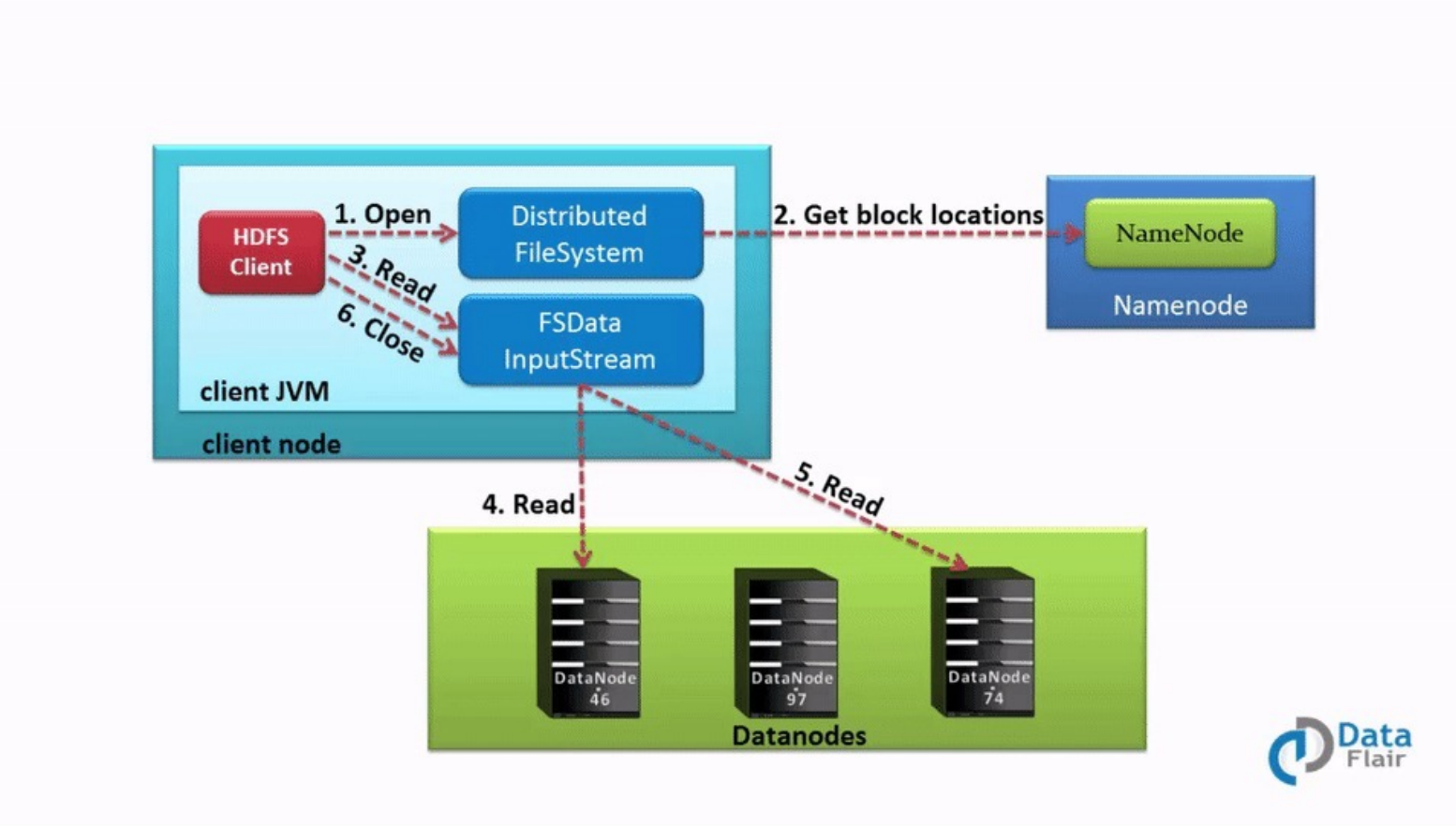
Tolleranza ai guasti



Operazioni di write



Operazioni di read



YARN

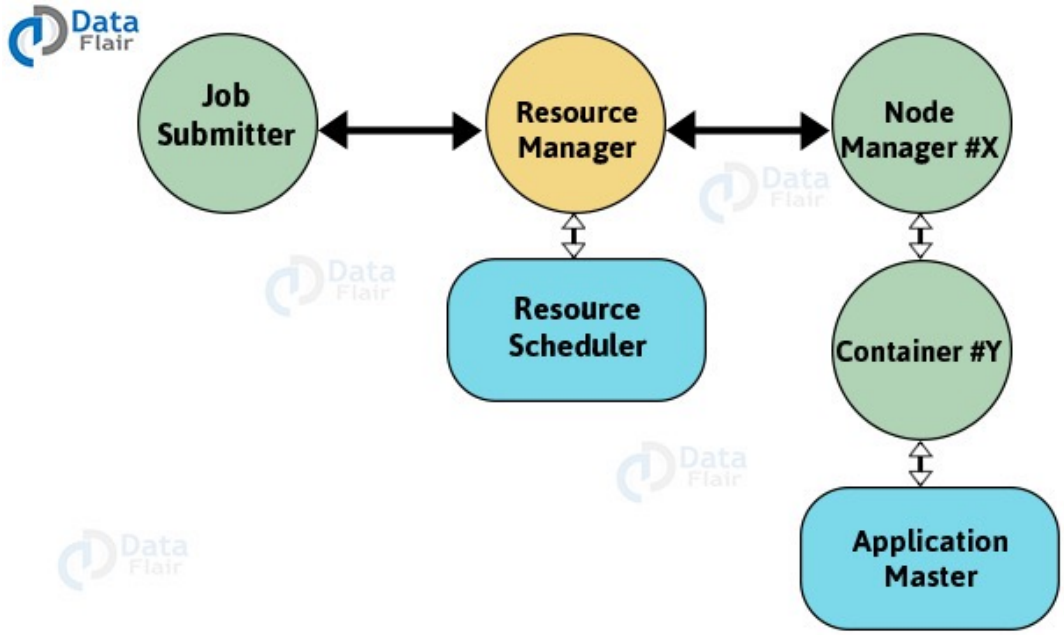
- YARN – Yet Another Resource Negotiator
- Tre componenti principali
  - Resource manager – gestore delle risorse del cluster
  - Node manager – gestore delle risorse del singolo nodo
  - Job submitter – esecutore del job

YARN

- Resource manager
  - Gestisce la rack awareness
  - Conosce le effettive risorse di ogni slave
  - Gestisce le risorse dei nodi slave (Res. Scheduler)
  - Il suo servizo di Application Manager negozia il primo container di ogni applicazione con i Node manager
- Node manager
  - Esegue su ogni nodo
  - Gestisce i container: frazioni delle risorse computazionali del nodo
  - Controlla l'utilizzo delle risorse di ogni container
  - Invia il segnale di heartbeat al Resource manager

Job Submitter

- Sottomette il job al Resource manager
- Resource manager e Resource scheduler negoziano il primo container per eseguire l'Application Master
- Il Resource Manager contatta il nodo interessato per lanciare il container

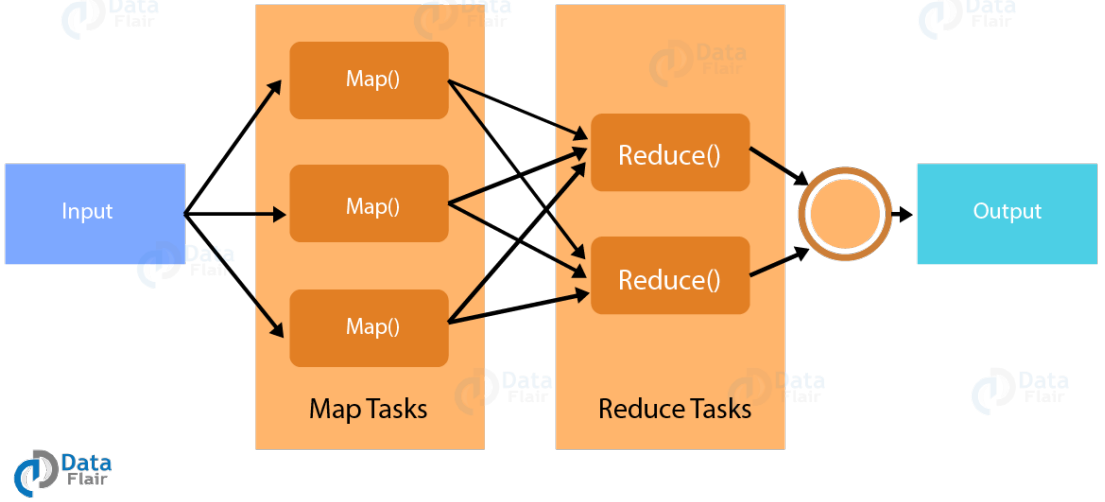


MapReduce

- MapReduce gestisce l'esecuzione di programmi sui dati presenti in HDFS e salva i risultati di nuovo in HDFS
- Ogni programma è costituito da due componenti
  - Mapper
  - Reducer
- Un task è l'esecuzione del mapper o del reducer su una partizione dei dati presenti su un nodo nel rispetto del principio di data locality

MapReduce task

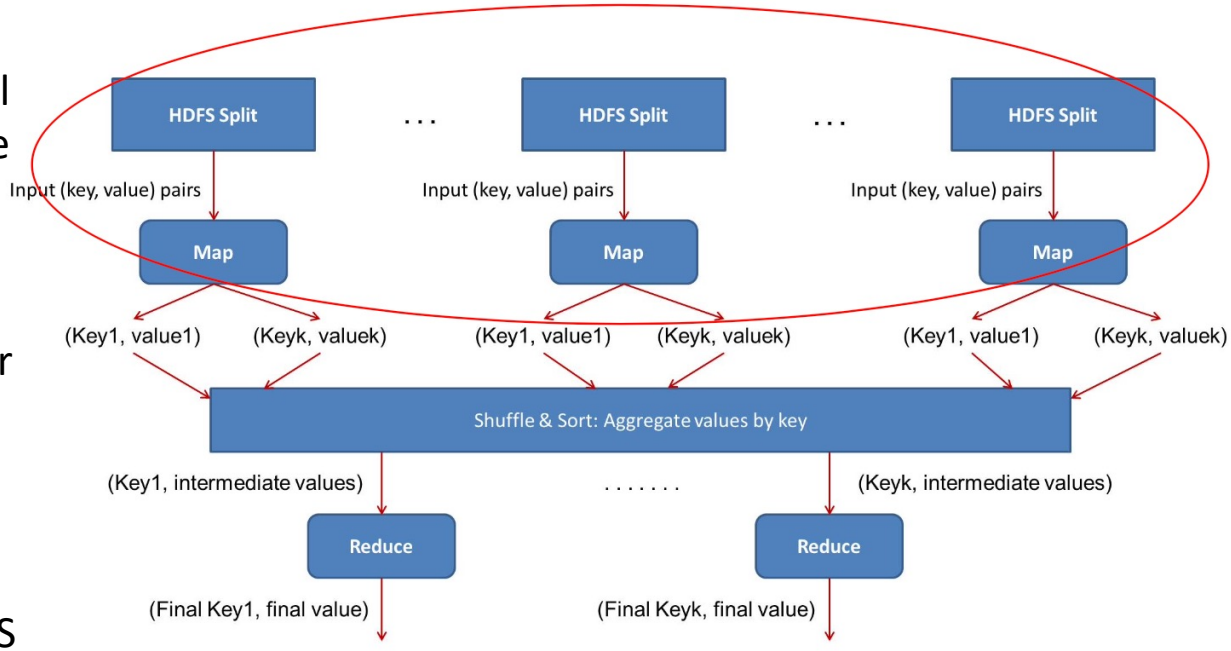
- I singoli task si generano perché il codice MapReduce viene fatto migrare sui vari nodi slave in cui si trovano i dati
- L'esecuzione di un task viene tentata (TaskAttempt) al più 4 volte e, se fallisce, il job viene considerato fallito
- Al fine di incrementare la performance, l'esecuzione speculativa rileva i task lenti e genera dei task di backup
- Quando il più veloce dei due termina l'altro viene cancellato



- I task di map e di reduce sono in genere su nodi distinti
- L'intero processo si svolge elaborando coppie chiave valore

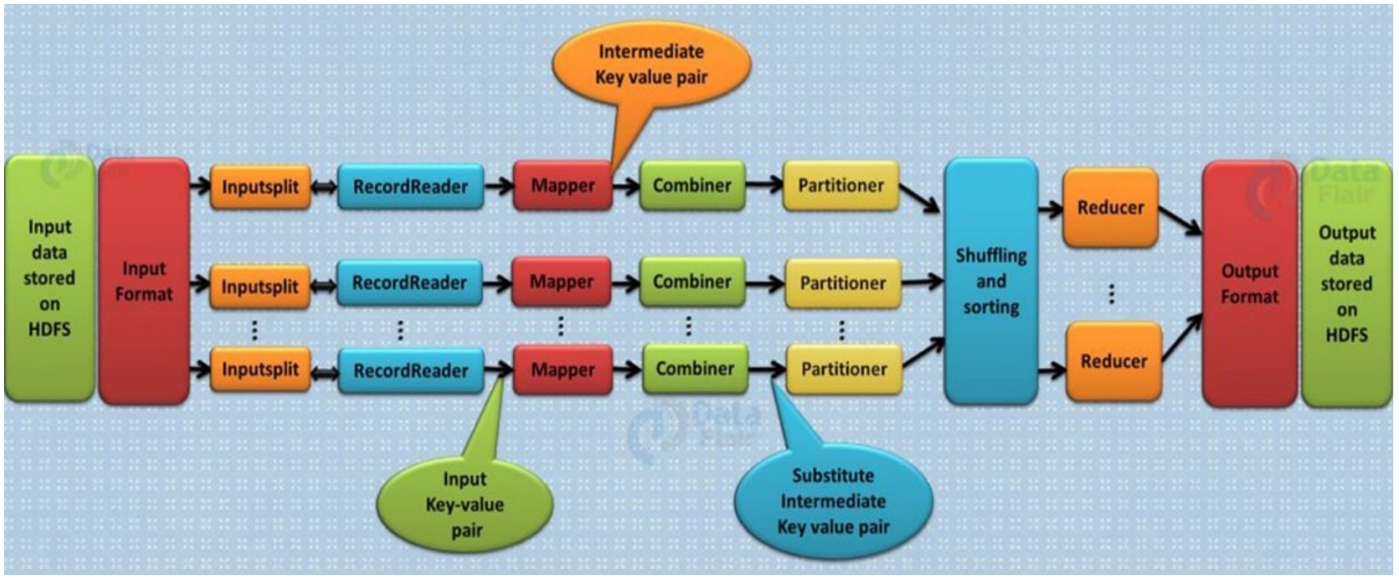
MapReduce flow

- HDFS esegue lo split dei dati, un blocco da 128MB per volta
- Ogni task di Map lavora su tutto il proprio input costituito da coppie chiave-valore
- Coppie in uscita da tutti i task di Map processate da un layer di shuffle/sort che aggrega i dati per chiave
- Ogni task di Reduce genera una nuova lista di coppie secondo la propria logica programmata
- Risultato salvato di nuovo in HDFS

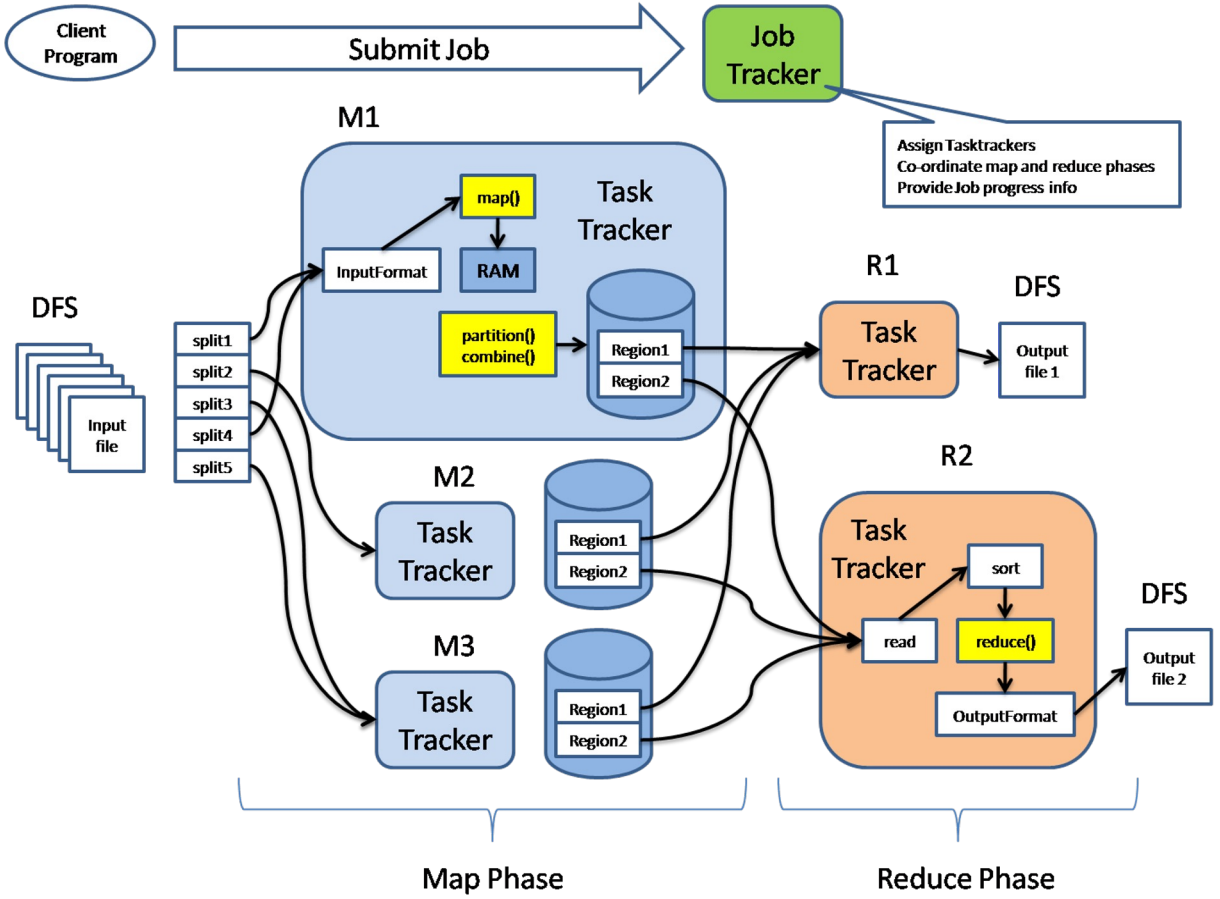


MapReduce software architecture

- Pipeline di esecuzione mappata in un serie di classi Java
- Un programma MapReduce ha una struttura fissa all'interno della quale si introduce la logica vera e propria



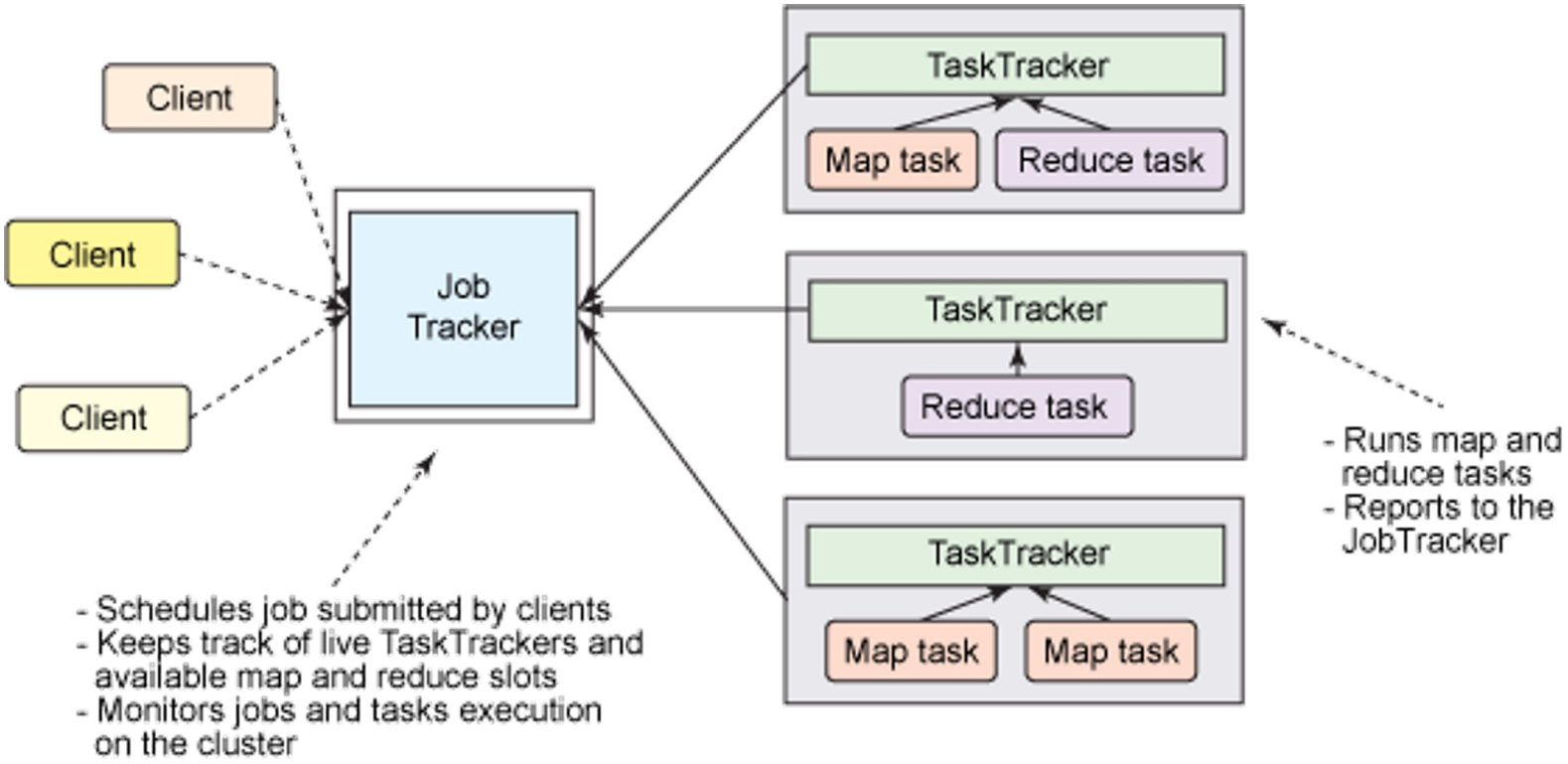
MapReduce Engine



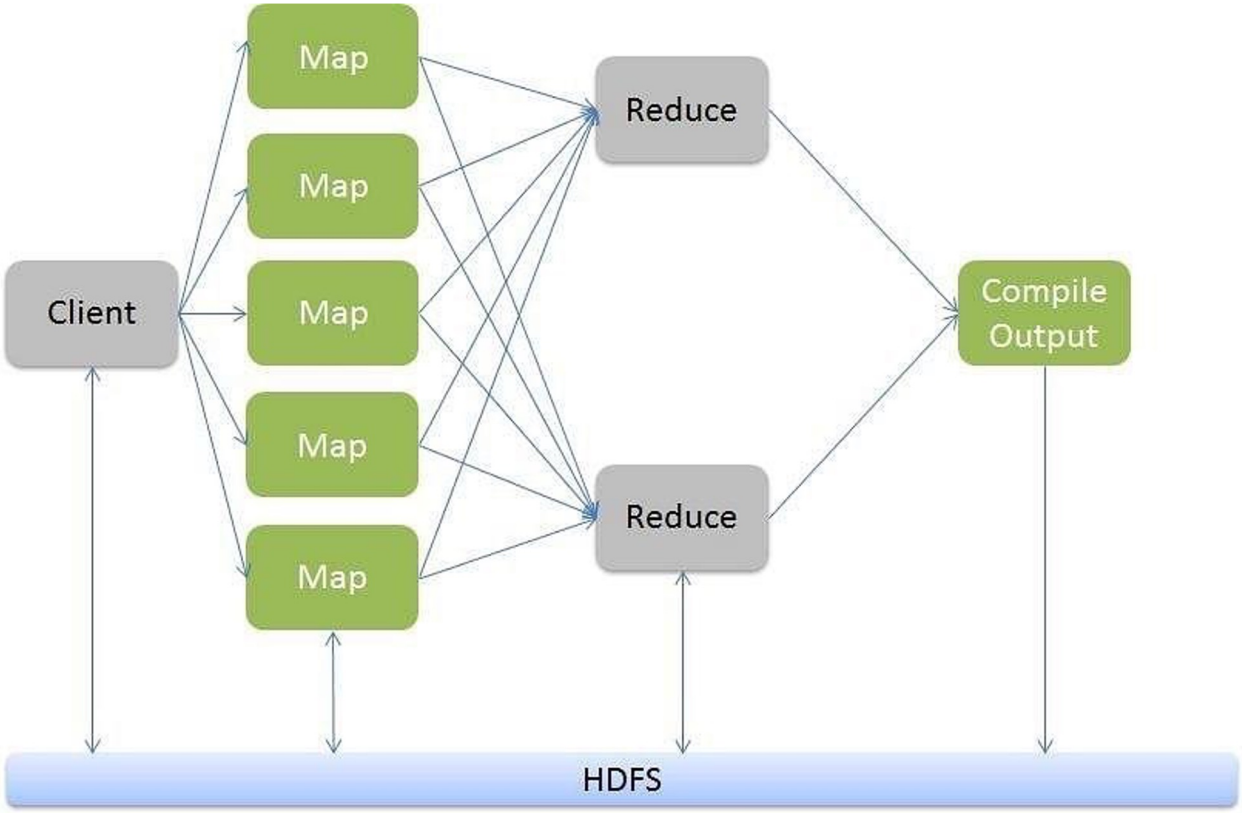
MapReduce Engine

- **Job**: questo è il lavoro effettivo che deve essere eseguito o elaborato
- **Task**: questa è una parte del lavoro effettivo che deve essere eseguito o elaborato. Un job MapReduce comprende molti piccoli task che devono essere eseguiti.
- **JobTracker**: questo tracker svolge il ruolo di job scheduling e tenere traccia di tutti i job assegnati ai TaskTracker.
- **TaskTracker**: questo tracker svolge il ruolo di task tracker e segnala lo stato dei task al JobTracker.
- **Client**: è un programma o API (Application Programming Interface) che invia lavori a MapReduce. MapReduce può accettare lavori da diversi client.

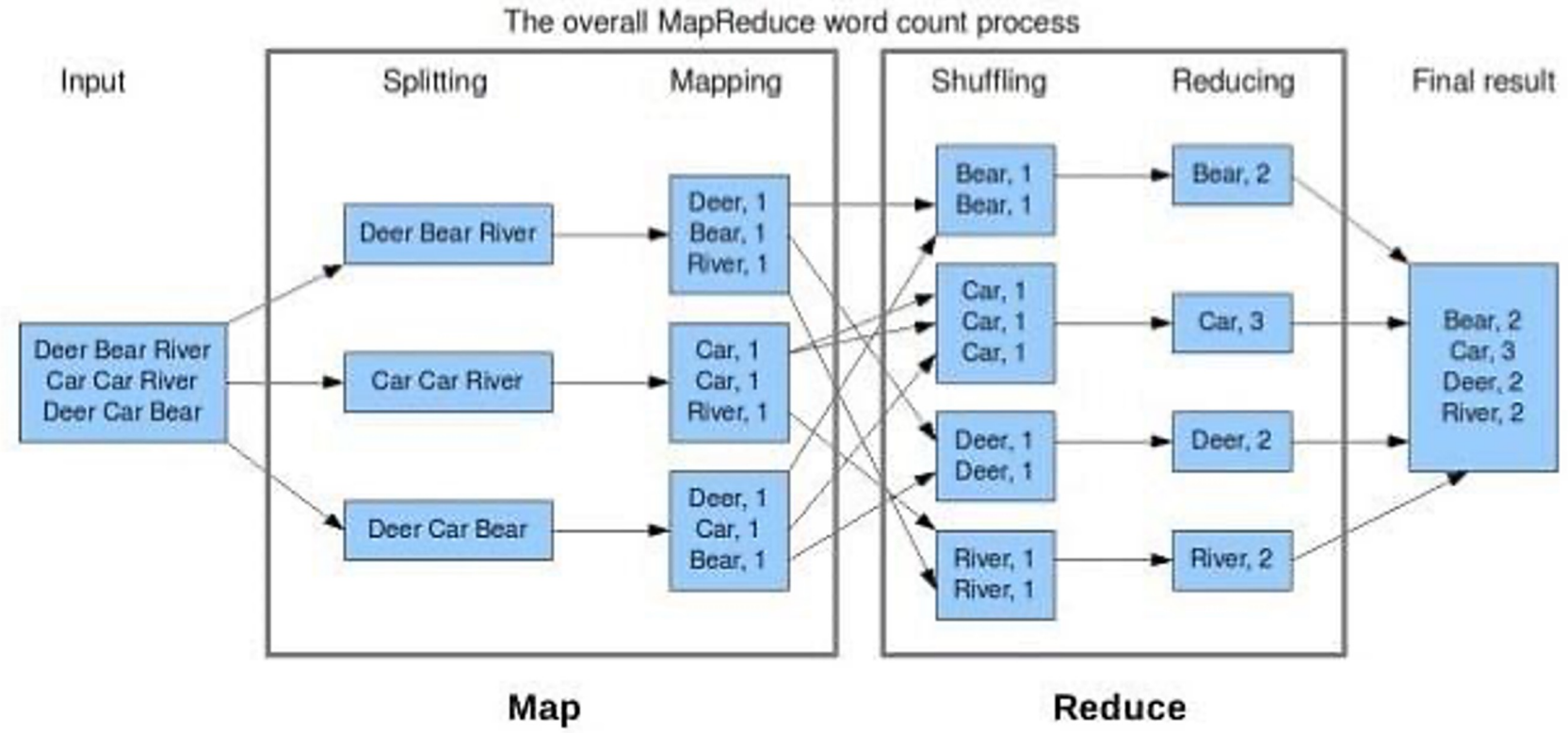
JobTracker e TaskTracker



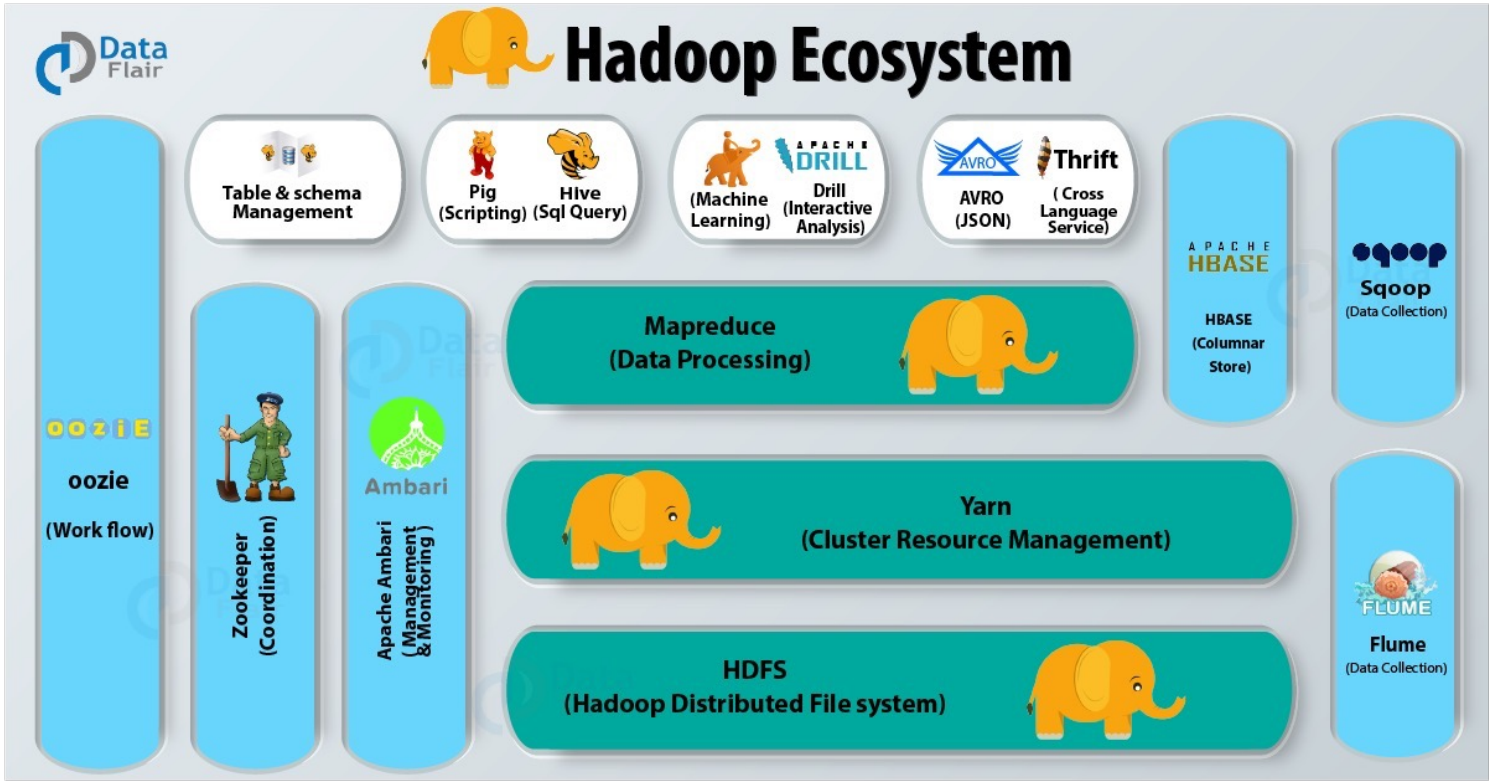
Esempio MapReduce



Word count in MapReduce



Ecosistema Hadoop



Uso di Hadoop

- Pubblicità (analisi del comportamento dell'utente per generare consigli mirati)
- Ricerche (relazioni tra gruppi di documenti)
- Sicurezza (ricerca di patterns non comuni)
- Elaborazione di set di dati di grandi dimensioni non in tempo reale:
  - Il NY Times generava dinamicamente PDF di articoli dal 1851 al 1922
  - Voleva pre-generare e pubblicare staticamente articoli per migliorare le prestazioni
  - Utilizzando Hadoop + MapReduce in esecuzione su EC2/S3, ha convertito 4 TB di TIFF in 11 milioni di articoli PDF in 24 ore

Riepilogo

- Caratteristiche di Hadoop
- HDFS
- YARN
- MapReduce

- Approfondimento
  - https://hadoop.apache.org/docs/stable/