

UNIVERSITÀ DEGLI STUDI DI CAMERINO

**Scuola di Scienze e Tecnologie**

*Corso di Laurea in Informatica*



**Big Data**

**Tecniche e tool di analisi**

Elaborato Finale

*Laureando*

**Emanuele Gentiletti**

*Relatore*

**Prof. Diletta Romana Cacciagrano**

Matricola: **090150**

# Indice

<b>1</b>	<b>Introduzione</b>	<b>2</b>
	Definizione di Big Data . . . . .	2
	RDBMS . . . . .	3
<b>2</b>	<b>Hadoop</b>	<b>4</b>
	HDFS . . . . .	5
	Principi architetturali . . . . .	5
	Funzionamento . . . . .	6
	MapReduce . . . . .	6
	<b>Bibliografia</b>	<b>7</b>

# Capitolo 1

## Introduzione

Negli ultimi anni, i Big Data hanno preso piede in modo impetuoso in una grande varietà di ambiti. In parte per questa ragione, il fenomeno ha avuto un enorme impatto: settori come medicina, finanza, business analytics, istruzione e ... possono sfruttare le capacità analitiche avanzate nei Big Data per guidare lo sviluppo e l'innovazione in modi semplicemente non possibili prima.

La ragione per cui

L'innovazione che ha reso possibili questi risultati è guidata dal software molto più che dall'hardware. Ci sono stati dei grandi cambiamenti nel modo di pensare alla computazione e all'organizzazione dei suoi processi, che hanno portato a risultati notevoli primariamente nell'efficienza di elaborazione di grandi quantità di dati.

Il cambiamento è dovuto principalmente ad Hadoop, un framework open source il cui design è orientato alla computazione batch di dataset di grandi dimensioni. Utilizzando un'architettura ben congeniata, Hadoop ha permesso

Una delle conseguenze più importanti di Hadoop è stata una democratizzazione delle possibilità di analisi dei dati:

- Hadoop è sotto licenza Apache, permettendo a chiunque di utilizzarlo a scopi commerciali e non;
- Hadoop non richiede hardware costoso ad alta affidabilità, e incoraggia l'adozione di macchine più generiche e prone al fallimento per il suo uso, che possono essere ottenute a costi inferiori;
- Il design di Hadoop permette la sua esecuzione in cluster di macchine eterogenee nel software e nell'hardware, che possono essere acquisite da diversi rivenditori, un altro fattore che permette l'abbattimento dei costi;
- I vari modelli di programmazione in Hadoop hanno in comune l'astrazione della computazione distribuita e dei problemi intricati che questa comporta, abbassando la barriera di entrata in termini di conoscenze e lavoro richiesti per creare programmi che necessitano di un altro grado di parallelismo.

Questi fattori hanno spinto a una vasta adozione di Hadoop e dell'ecosistema software che lo circonda, in ambito aziendale e scientifico. L'adozione di Hadoop, secondo un sondaggio fatto a maggio 2015[1], si aggira al 26% delle imprese, e si prevede che il mercato di Hadoop attorno ad Hadoop sorpasserà i 16 miliardi di dollari nel 2020 [2].

<http://www.techrepublic.com/article/the-secret-ingredients-for-making-hadoop-an-enterprise-tool/>

## Definizione di Big Data

Per Big Data si intendono collezioni di dati non gestibili da tecnologie "tradizionali". La ragione per cui queste collezioni non sono gestibili è rilevabile in tre fattori

- Il **volume** della collezione;
- La **varietà**, intesa come la varietà di *fonti* e di *possibili strutturazioni* dell'informazione;
- La **velocità** dell'informazione, intesa come la velocità di produzione di nuova informazione.

Ognuno dei punti di questo modello deriva da esigenze relativamente recenti (a volte per la tipologia, a volte per la scala di necessità), in particolare:

- Il volume delle collezioni di dati è aumentato esponenzialmente in tempi recenti, con l'avvento dei Social Media, dell'IOT, e degli smartphone muniti di molti sensori diversi. Generalizzando, i fattori che hanno portato a un grande incremento del volume dei data set sono un aumento della generazione automatica di dati da parte di dispositivi (sensori a basso costo e smartphone), in opposizione all'inserimento manuale dei dati da parte di operatori, e di un grande incremento dei contenuti prodotti dagli utenti rispetto al passato.
- La varietà delle collezioni di dati è aumentata, perché ci sono più fonti rispetto che in passato da cui è desiderabile attingere dati, e molte fonti forniscono dati che non sono strutturati uniformemente rispetto alle altre. Le fonti possono differire in struttura, o possono essere non strutturate affatto, come nel caso dei documenti JSON o del linguaggio naturale. Una struttura uniforme è una condizione necessaria per l'elaborazione corretta dei dati, e a volte può non essere triviale giungere a questa condizione. Ci sono molti casi in cui le fonti di dati possono avere informazioni non corrette che richiedono di essere filtrate, o in cui è necessario applicare strategie difensive nei confronti dei dati ricevuti, per la possibilità che questi siano mal filtrati o provengano da una fonte non sicure.
- Si possono fare le stesse considerazioni fatte per il volume dei dati per quanto riguarda la velocità. I flussi di dati vengono generati dai dispositivi e dagli utenti, che li producono a velocità molto maggiori rispetto a degli operatori.

La definizione di Big Data fornita parla di collezioni di dati non gestibili da tecnologie tradizionali. Definite le caratteristiche di queste collezioni, le domande consequenziali a questa definizione sono, *quali sono le tecnologie tradizionali, e perché non sono adeguate?*

## RDBMS

I database relazionali sono stati in grado di gestire consistentemente e affidabilmente

## Capitolo 2

# Hadoop

La documentazione ufficiale di Hadoop lo descrive come:

...un framework che abilita l'elaborazione distribuita di grandi dataset in cluster di computer utilizzando semplici modelli di programmazione. **Hadoop** è progettato per essere scalato da server singoli a migliaia di macchine, dove ognuna di queste offre computazione e storage locale. Invece di affidarsi all'hardware per fornire un'alta affidabilità, **Hadoop** è progettato per rilevare e gestire i fallimenti [delle computazioni] a livello applicativo, mettendo a disposizione un servizio ad alta affidabilità su cluster di computer pronti al fallimento.

In questa definizione sono racchiusi dei punti molto importanti:

- **Semplici modelli di programmazione**

Hadoop raggiunge molti dei suoi obiettivi fornendo un'interfaccia di livello molto alto al programmatore, in modo di potersi assumere la responsabilità di concetti complessi e necessari all'efficienza nella computazione distribuita, ma che hanno poco a che fare con il problema da risolvere in sé (ad esempio, la sincronizzazione di task paralleli e lo scambio dei dati tra nodi del sistema distribuito). Questo modello **pone dei limiti alla libertà del programmatore**, che deve adeguare la codifica della risoluzione del problema al modello di programmazione fornito.

- **Computazione e storage locale**

L'ottimizzazione più importante che Hadoop fornisce rispetto all'elaborazione dei dati è il risultato dell'unione di due concetti: **distribuzione dello storage e distribuzione della computazione**.

Entrambi sono importanti a prescindere dell'uso particolare che ne fa Hadoop: la distribuzione dello storage permette di combinare lo spazio fornito da più dispositivi e di farne uso tramite un'unica interfaccia logica, e di replicare i dati in modo da poter tollerare guasti nei dispositivi. La distribuzione della computazione permette di aumentare il grado di parallelizzazione nell'esecuzione dei programmi.

Hadoop unisce i due concetti utilizzando cluster di macchine che hanno sia lo scopo di mantenere lo storage, che quello di elaborare i dati. Quando Hadoop esegue un lavoro, **quante più possibili delle computazioni richieste vengono eseguite nei nodi che contengono i dati da elaborare**. Questo permette di ridurre la latenza di rete, minimizzando la quantità di dati che devono essere scambiati tra i nodi del cluster. Il meccanismo è trasparente all'utente, a cui basta persistere i dati da elaborare nel cluster per usufruirne. Questo principio viene definito **data locality**.

- **Scalabilità**

|||

- **Hardware non necessariamente affidabile**

I cluster di macchine che eseguono Hadoop non hanno particolari requisiti di affidabilità rispetto ad hardware consumer. Il framework è progettato per tenere in conto dell'alta probabilità di fallimento dell'hardware, e per attenuarne le conseguenze, sia dal punto di vista dello storage e della potenziale perdita di dati,

che da quello della perdita di risultati intermedi e parziali nel corso dell'esecuzione di lavori computazionalmente costosi. In questo modo l'utente è sgravato dal compito generalmente difficile di gestire fallimenti parziali nel corso delle computazioni.

Hadoop è composto da diversi moduli:

- **HDFS**, un filesystem distribuito ad alta affidabilità, che fornisce replicazione automatica all'interno dei cluster e accesso ad alto throughput ai dati
- **YARN**, un framework per la schedulazione di lavori e per la gestione delle risorse all'interno del cluster
- **MapReduce**, un framework e un modello di programmazione fornito da Hadoop per la scrittura di programmi paralleli che processano grandi dataset.

## HDFS

Come accennato, HDFS è un filesystem distribuito, che permette l'accesso ad alto throughput ai dati. HDFS è scritto in Java, e viene eseguito nello userspace. Lo storage dei dati passa per il filesystem del sistema che lo esegue.

I dati contenuti in HDFS sono organizzati in unità logiche chiamate *blocchi*, come è comune nei filesystem. Rispetto a questi, tuttavia, la loro dimensione è molto più grande, 128 MB di default. La ragione per cui HDFS utilizza blocchi così grandi è minimizzare il costo delle operazioni di seek, dato il fatto che se i file sono composti da meno blocchi, si rende necessario trovare l'inizio di un blocco un minor numero di volte.

Il blocco, inoltre, è un'astrazione che si presta bene alla replicazione dei dati nel filesystem all'interno del cluster: per replicare i dati, come si vedrà, si mette uno stesso blocco all'interno di più macchine nel cluster.

HDFS è basato sulla specifica POSIX, ma non la implementa in modo rigido: tralasciare alcuni requisiti di conformità alla specifica permette ad HDFS di ottenere prestazioni e affidabilità migliori, come verrà descritto in seguito.

## Principi architetturali

La documentazione di Hadoop descrive i seguenti come i principi architetturali alla base della progettazione di HDFS:

- **Fallimento hardware come regola invece che come eccezione**

Un sistema che esegue HDFS è composto da molti componenti, con probabilità di fallimento non triviale. Sulla base di questo principio, HDFS dà per scontato che **ci sia sempre un numero di componenti non funzionanti**, e si pone di rilevare errori e guasti e di fornire un recupero rapido e automatico da questi.

Il meccanismo principale con cui HDFS raggiunge questo obiettivo è la replicazione: in un cluster, ogni blocco di cui un file è composto è replicato in più macchine (3 di default). Se un blocco non è disponibile in una macchina, o se non supera i controlli di integrità, una sua copia può essere letta da un'altra macchina in modo trasparente per il client.

Il numero di repliche per ogni blocco è configurabile, e ci sono più criteri con cui viene deciso in quali macchine il blocco viene replicato.

- **Modello di coerenza semplice**

Per semplificare l'architettura generale, HDFS fa delle assunzioni specifiche sul tipo di dati che vengono salvati in HDFS e pone dei limiti su come l'utente possa lavorare sui file. In particolare, **non è possibile modificare arbitrariamente file già esistenti**, e le modifiche devono limitarsi a operazioni di troncamento e di append (aggiunta a fine file). Queste supposizioni permettono di semplificare il modello di coerenza, perché i blocchi di dati, una volta scritti, possono essere considerati immutabili, evitando una considerevole quantità di problemi in un ambiente dove i blocchi di dati sono replicati in più posti:

- Per ogni modifica a un blocco di dati, bisognerebbe verificare quali altre macchine contengono il blocco, e rieseguire la modifica (o rireplicare il blocco modificato) in ognuna di queste.

- Queste modifiche dovrebbero essere fatte in modo atomico, o richieste di lettura su una determinata replica di un blocco invece che in un'altra potrebbe portare a risultati inconsistenti o non aggiornati.

Le limitazioni che Hadoop impone sono ragionevoli per lo use-case per cui HDFS è progettato, caratterizzato da grandi dataset che vengono copiati nel filesystem e letti in blocco.

- **Accesso in streaming**

HDFS predilige l'accesso ai dati in streaming, per permettere ai lavori batch di essere eseguiti con grande efficienza. Questo approccio va a discapito del tempo di latenza della lettura dei file, ma permette di avere un throughput in lettura molto vicino ai tempi di lettura del disco.

- **Dataset di grandi dimensioni**

- **Portabilità su piattaforme software e hardware eterogenee**

HDFS è scritto in Java, ed è portabile in tutti i sistemi che ne supportano il runtime.

## **Funzionamento**

L'architettura di HDFS è di tipo master/slave, dove un nodo centrale, chiamato NameNode, gestisce i metadati e la struttura del filesystem, mentre i nodi slave, chiamati DataNode, contengono i blocchi di cui file sono composti. Tipicamente, viene eseguita un'istanza del software del DataNode per macchina del cluster, e una macchina dedicata esegue il NameNode.

## **MapReduce**

Per dare un'idea concreta di come funziona la programmazione in un cluster Hadoop

MapReduce è il primo importante modello di programmazione a cui Hadoop fa riferimento per l'esecuzione di applicazioni distribuite. Hadoop è stato scritto e pensato per l'esecuzione di lavori MapReduce, e nelle prime versioni era il solo modello di programmazione disponibile.

I lavori MapReduce

# Bibliografia

1. Gartner Survey Highlights Challenges to Hadoop Adoption, <http://www.gartner.com/newsroom/id/3051717>
2. Hadoop Market Forecast 2017-2022, <https://www.marketanalysis.com/?p=279>