



## Il data warehouse nell'era dei Big Data

a cura di:

Gino Farisano



# **Abstract**

Scopo di questo lavoro, nato come progetto d'esame di basi di dati 2 tenuto dalla prof.ssa G. Tortora e G. Polese, è di esplorare le possibilità offerte dai big data, dalle caratteristiche che li contraddistinguono agli strumenti attualmente disponibili per acquisirli, memorizzarli ed estrarne informazione utile. Si è volutamente scelto uno stile pragmatico, per cui, relativamente alla tematica affrontata, saranno presentati numerosi esempi e contesti di utilizzo degli strumenti oggetti di studio. Pertanto, l'elaborato è rivolto alle aziende che siano intenzionate a valutare l'utilizzo di tali strumenti per accrescere il proprio business o a chiunque sia interessato ad intraprendere la carriera del data scientist "informatico, statistico e narratore che estrae pepite d'oro nascoste sotto montagne di dati", il mestiere che The Economist definisce "the sexiest job of the 21st century".

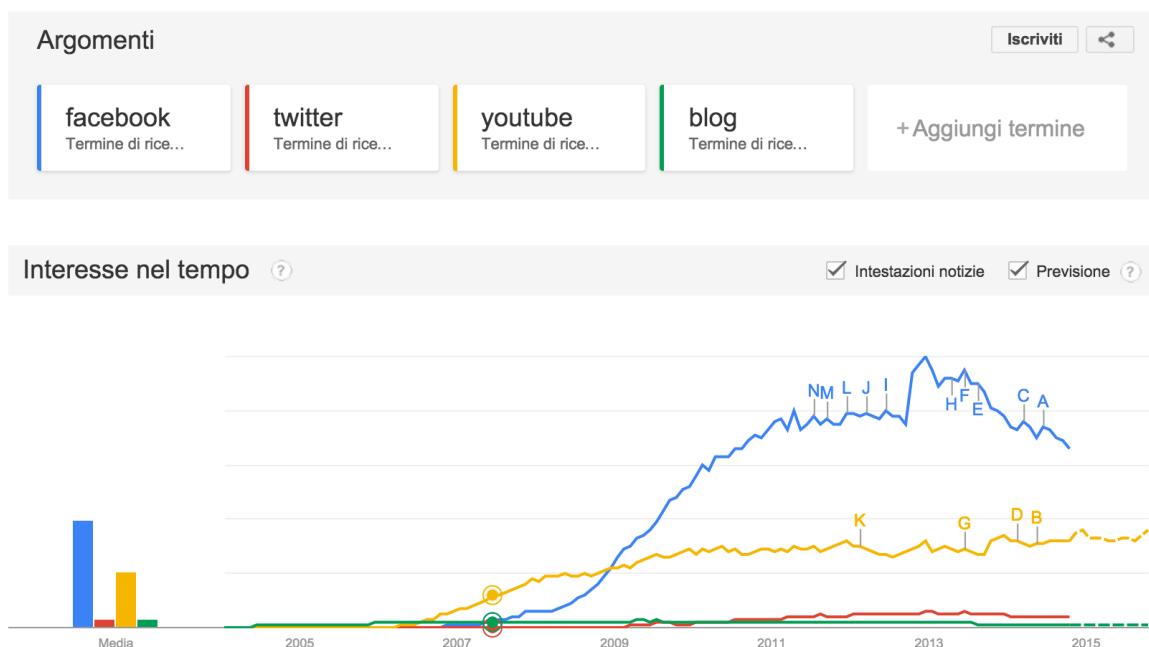
<b>Capitolo 1</b>	<b>6</b>
1.0 Introduzione	6
1.1 Contesto storico	8
1.2 I big data	9
1.3 Business intelligence	13
<b>Capitolo 2</b>	<b>16</b>
2.1 Le tecnologie di acquisizione per Big Data	17
2.2 API	17
2.2.1 API Twitter	18
2.3 Strumenti di web scraping	22
2.3.1 cURL	23
2.4 Apache Tika	24
2.5 Strumenti ETL e ELT	25
2.5.1 Sqoop	28
2.6 Apache Flume	30
<b>Capitolo 3</b>	<b>31</b>
3.2 HDFS	32
3.3 Limiti del modello relazionale	34
3.4 Database NoSQL	35
3.4.1 Proprietà ACID	36
3.4.2 Teorema CAP e proprietà BASE	37
3.4.3 Categorie di database NoSQL	38
3.5 Apache HBase	43
<b>Capitolo 4</b>	<b>46</b>
4.1 Perché MapReduce?	46
4.2 Il paradigma MapReduce	47
4.2.1 Hadoop MapReduceFramework	49
4.2.2 MapReduce, un esempio di utilizzo: WordCount	52

4.2.3 Hive e Pig	53
Conclusioni	57

# Capitolo 1

## 1.0 Introduzione

L'arrivo di Internet negli anni '90 ha permesso a molte aziende di ampliare il proprio bacino di utenza e di fornire online i propri servizi; pensiamo ad esempio ad Amazon nata nel 1994 e che oggi presenta un fatturato di 126 milioni di dollari [1]. Inoltre, con il Web 2.0, la quantità dei dati presenti sulla rete è ulteriormente aumentata attraverso gli user-generated content, cioè da contenuti prodotti in gran parte dagli utenti [2]: foto, video, blog post. Come si evince dal grafico sottostante [img. 1], generato dal sito [www.google.it/trend](http://www.google.it/trend)<sup>1</sup>, infatti, l'interesse nei confronti di termini quali facebook, twitter, ecc. è cresciuta vertiginosamente a partire dal 2007.



[img 1] Trends di termini ricorrenti nel www.

<sup>1</sup> Google Trends è uno strumento basato sulla ricerca di Google che mostra quanto spesso un termine sia stato ricercato nelle varie lingue e nelle varie regioni del mondo. Il trend mostrato risale al 10/10/2014 e comprende anche una previsione del 2015.

Questo ha fatto sì che i sistemi informativi elaborassero molte più informazioni e quindi maggiore potenza di calcolo richiesta; inoltre i dati sono diventati sempre più numerosi, eterogenei tra loro e sempre più difficili da gestire e organizzare. D'altro canto le aspettative dell'utente finale sono un servizio sempre disponibile, facilmente accessibile (sia come tempi di risposta che come usabilità) e probabilmente attendibile [3]. Pertanto, la tendenza è diventata quella di garantire più il servizio o il contenuto, rispetto alla correttezza delle informazioni o alla consistenza dei dati. Si pensi, ad esempio, alla classifica dei libri venduti su un sito di e-commerce (es. Amazon, Ibs): dal punto di vista dell'utente è importante avere quest'informazione sempre disponibile, magari non corretta al 100% e quindi approssimata, piuttosto che attendere lunghi tempi di elaborazione prima di ricevere una risposta alle proprie domande. In questo scenario, i DBMS relazionali possono rappresentare un collo di bottiglia per i sistemi che possono svincolarsi dall'affidabilità delle informazioni [3]. Nel prossimo paragrafo verrà brevemente presentato il contesto storico e le soluzioni proposte per risolvere le problematiche sopra citate. Lo stesso sarà approfondito nel terzo capitolo in cui si discuterà dei database NoSql. Al fine di porre maggiore accento ai prodotti utilizzabili in ambiente Hadoop, si analizzerà nello specifico il database column-oriented HBASE e non saranno trattate le altre tipologie di Database NoSql. Per ulteriori approfondimenti, vedere [6].

## 1.1 Contesto storico

Come anticipato nel paragrafo precedente, i database relazionali possono rappresentare un collo di bottiglia nell'elaborazione di grosse quantità di dati. In particolare sono stati riscontrati problemi nel garantire un'efficiente elaborazione, un efficace parallelizzazione, scalabilità e costi ridotti. Il primo esempio storico proviene da Google che ha dato vita ad un'infrastruttura estremamente scalabile per il suo motore di ricerca creando il Google File System (GFS), un filesystem distribuito che permette di "spalmare" file di grosse dimensioni su centinaia di macchine in shared-nothin cluster, creati usando commodity hardware<sup>2</sup>. Quest'ultimo fornisce centinaia di terabyte di storage attraverso migliaia di macchine, ed è contemporaneamente accessibile da centinaia di clienti [4] per applicazioni data-intensive. Inoltre l'infrastruttura google si compone di un database non relazionale (BigTable), un sistema di coordinazione distribuito ed un ambiente di esecuzione di algoritmi paralleli basati sul meccanismo MapReduce. Google ha pubblicato una serie di articoli ([4] e [5]) i quali hanno suscitato l'interesse di molti sviluppatori open-source. I creatori del motore di ricerca *Lucene* sono stati i primi a sviluppare una versione open-source che replica alcune caratteristiche dell'infrastruttura Google. Successivamente, i principali sviluppatori di *Lucene*, quelli di Yahoo e altri collaboratori hanno creato un universo parallelo a Google che imitava tutte le componenti della sua infrastruttura di calcolo distribuito [6]. Quest'alternativa open-source a Google è Hadoop [6]. Concludendo, nel 2007, anche Amazon ha presentato la propria proposta di database distribuito altamente disponibile ed eventualmente consistente:

---

<sup>2</sup> Computer a prezzi accessibili, tipicamente a basse performance; come vedremo in dettaglio nel terzo capitolo, in ambiente Hadoop, più computer di fascia bassa, raggruppati in rack (armadi di grosse dimensioni) "collaborano" raggiungendo capacità di calcolo estremamente elevate.

Dynamo. Successivamente, l'interesse nei confronti dei database NoSql è cresciuto ancora (MongoDB, CouchDB, FlockDB, Cassandra, ecc).

## 1.2 I big data

E' evidente l'esigenza da parte delle aziende di archiviare, gestire e trattare quantità di dati sempre crescenti. Il gruppo italiano Dedagroup, leader nel settore dell'information technology, sostiene che è difficile stimare la crescita del volume dei dati generati e da gestire, il fatto certo è che il volume crescerà, e in maniera cospicua [8]. Questa esplosione del quantitativo di dati giustifica il fatto che uno dei termini ricorrenti degli ultimi tempi, nel mondo dell'informatica, sia *big data*. Una visione interessante di cosa sono i Big Data è stata esposta da Alexander Jaimes, ricercatore presso Yahoo Research il quale ha affermato: "i big data siano noi". Sono proprio i contenuti generati dagli utenti, prodotti "volontariamente" da persone comuni che alimentano le basi di dati attraverso wiki, video, reating di vario genere (ristoranti, prodotti alimentari, ecc). Sebbene l'aggregazione degli user-generated content sia utile lo è ancor di più cercare dei pattern in quest'ultimi per capire "come funziona il mondo". Ad esempio, Dirk BrockMann e Fabian Theis hanno analizzato la circolazione di documenti attraverso il geocaching<sup>3</sup> per scoprire i pattern nei movimenti umani. Un altro articolo, "Digital Footprinting: Uncovering Tourists with User- Generated Content" , rileva come i turisti esplorano Roma in base alla geolocalizzazione delle loro foto e delle loro chiamate telefoniche [9].

---

<sup>3</sup> Il geocaching è un tipo di caccia al tesoro in cui i partecipanti, detti "geocacher", usano un ricevitore GPS per nascondere o trovare contenitori di differenti tipologie e dimensioni. Questi contenitori sono chiamati "geocache" o più semplicemente "cache".

Analizziamo ora singolarmente quali sono le caratteristiche che contraddistinguono i big data, conosciute come "le quattro V dei Big Data":

1. Volume: il volume si riferisce alla capacità di acquisire, memorizzare ed accedere a grandi volumi di dati; basti pensare che il 90% dei dati di tutto il mondo è stato generato negli ultimi due anni. Il McKinsey Global Institute afferma che il set di dati è superiore alle capacità che i tipici DBMS sono in grado di acquisire, memorizzare, gestire e analizzare, infatti, devono essere trattate informazioni che partono dai terabytes ai petabytes per entrare nel mondo degli exabyte e i volumi sono in continuo aumento [8].

kilobyte (KB)	$10^3$
megabyte (MB)	$10^6$
gigabyte (GB)	$10^9$
terabyte (TB)	$10^{12}$
petabyte (PB)	$10^{15}$
exabyte (EB)	$10^{18}$
zettabyte (ZB)	$10^{21}$
yottabyte (YB)	$10^{24}$

[img. 2] Multipli dei Bytes

2. Varietà: la varietà può essere intesa come molteplicità di fonti o come eterogeneità del formato dei dati [10]. Come mostrato in [img. 3], questi dati possono avere diversi formati che sono riconducibili a tre categorie: dati strutturati, semi-strutturati e non strutturati. I dati strutturati sono quelli che rispettano regole predefinite, quali: tipo di contenuto, lunghezza, formato, ecc. Questa tipologia di dati è di semplice archiviazione, interpretazione e

Social networks e social media (es. Facebook, Twitter, blogs, forum)	54%
Documenti cartacei digitalizzati	52%
Email	46%
Transazioni	40%
Immagini	34%
Registrazioni video	32%
Dati di geo-posizionamento (GPS)	25%
Dati generati da sensori o misuratori digitali (es. RFID, NFC, meters)	25%
Automazione processi produttivi	24%
Clickstream – Web Log	18%
M2M (Machine to Machine) data - Internet of Things	17%
Digitalizzazione dei processi di R&D (es. nella bioinformatica e biogenetica, chimica, climatologia)	15%
Registrazioni audio	12%
Altro	3%

[img. 3] Fonti dati associate ai Big Data

categorizzazione, come ad esempio la strutturazione utilizzata nei DB relazionali. I dati semi-strutturati presentano caratteristiche sia dei precedenti che dei dati non strutturati e vengono rappresentati, ad esempio, con il linguaggio XML. Infine, i dati non strutturati sono completamente privi di schema (immagini, porzioni di testo). Come possiamo estrarre, allora, informazioni da quest'ultimi? La disciplina che studia come manipolare i dati non strutturati è l'Information Retrieval la quale rispetto alla teoria classica delle basi di dati, pone l'enfasi non sulla ricerca di dati ma sulla ricerca di informazioni [12]. Per quanto riguarda l'analisi testuale, nonostante siano stati proposti linguaggi di interrogazione più complessi, nella maggior parte dei casi quest'ultimi sono solitamente composti da elenchi di parole chiave. La costruzione della risposta, a differenza di quanto accade per un'interrogazione relazionale (modello booleano, quindi tupla presente o non presente), può

rispondere più o meno bene ai requisiti espressi nell'interrogazione e dato il gap tra dati e informazioni, dovuto all'ambiguità dei dati, spesso non si riesce a determinare con precisione se un risultato sia completamente o per nulla rilevante. Per questo motivo i risultati vengono ordinati per gradi di rilevanza (vedi ad esempio ricerca Google) [12].

## STRUTTURATI

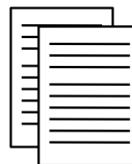
<i>id-pers</i>	<i>nome</i>	<i>cognome</i>
0000001	Mario	Rossi
0000002	Giorgio	Verdi

<i>id-pers</i>	<i>telefono</i>
0000001	051 1234
0000001	333 3333



## NON STRUTTURATI



3

[img. 4] Dati strutturati e dati non strutturati

3. Velocità: la velocità è riferita al fatto che l'analisi deve essere fatta in tempo reale. Riprendendo l'esempio relativo alla classifica dei libri più venduti da Amazon, è impensabile che l'utente debba attendere del tempo prima di conoscere questa informazione: la risposta deve essere real time.

4. Veridicità: i dati raccolti costituiscono un valore per l'azienda. E' dall'analisi di questi che si colgono le opportunità e si trae supporto per i processi decisionali; è, per questo, importante far riferimento solo a dati attendibili.

## 1.3 Business intelligence

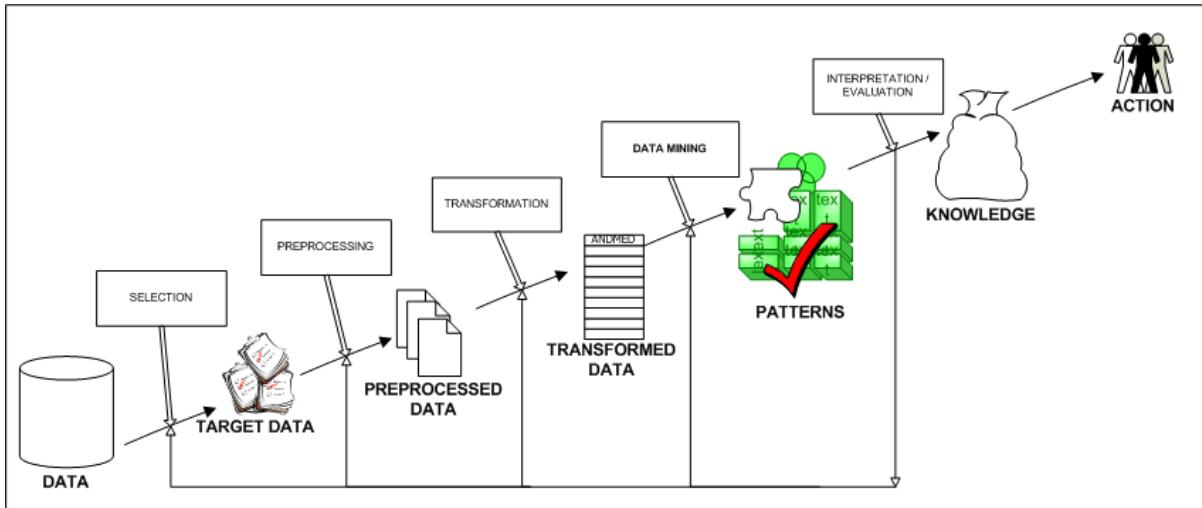
In questo paragrafo discuteremo dell'utilizzo dei big data a supporto dei processi decisionali e dal management. È fondamentale, però, ripercorrere brevemente l'evoluzione delle attività di analisi dai "classici" DBMS ai sistemi MOLAP.

Le basi di dati tradizionali sono di tipo OLTP (On Line Transaction Processing) e il modello dei dati è fortemente normalizzato<sup>4</sup>, per favorire, non tanto le letture e l'analisi di grandi quantità di record, quanto le cosiddette attività transazionali (inserimenti, cancellazioni, modifiche dei dati). Per questo motivo, sono necessarie operazioni di JOIN per denormalizzare le tabelle e rendere possibile la lettura dei dati da parte dell'operatore. A partire degli anni novanta sono emersi i data warehouse, database specializzati contenenti dati statici integrati, che riguardano una serie di fatti accaduti nel tempo e finalizzati al recupero di informazioni a supporto dei processi decisionali. Inizialmente, le interrogazioni al data warehouse erano effettuate tramite query SQL, le quali richiedevano conoscenza da parte dell'analista di tecniche relative ai database [2]. Si è poi passati dai sistemi ROLAP ai sistemi MOLAP la cui struttura base è un cubo multi-dimensionale che consente all'analista di fare a meno di conoscenze tecniche e concentrarsi sulle problematiche del business. Attraverso specifiche operazioni (OLAP), come quelle di slice, dice, roll up e drill down è possibile quindi "navigare all'interno dei dati". Proseguendo sul nostro asse temporale, a partire dai primi anni duemila, si è affermata la necessità di effettuare analisi previsionali per anticipare gli eventi e trarre vantaggio dal business; potremmo essere, ad esempio, interessati a "particolarità" (nuove tendenze), anomalie (frodi), ecc. Questa nuova disciplina,

---

<sup>4</sup> La normalizzazione è il procedimento volto all'eliminazione della ridondanza e del rischio di incoerenza nei database. Favorisce l'attività transazionale ma incrementa notevolmente il numero di tabelle utilizzate per contenerli

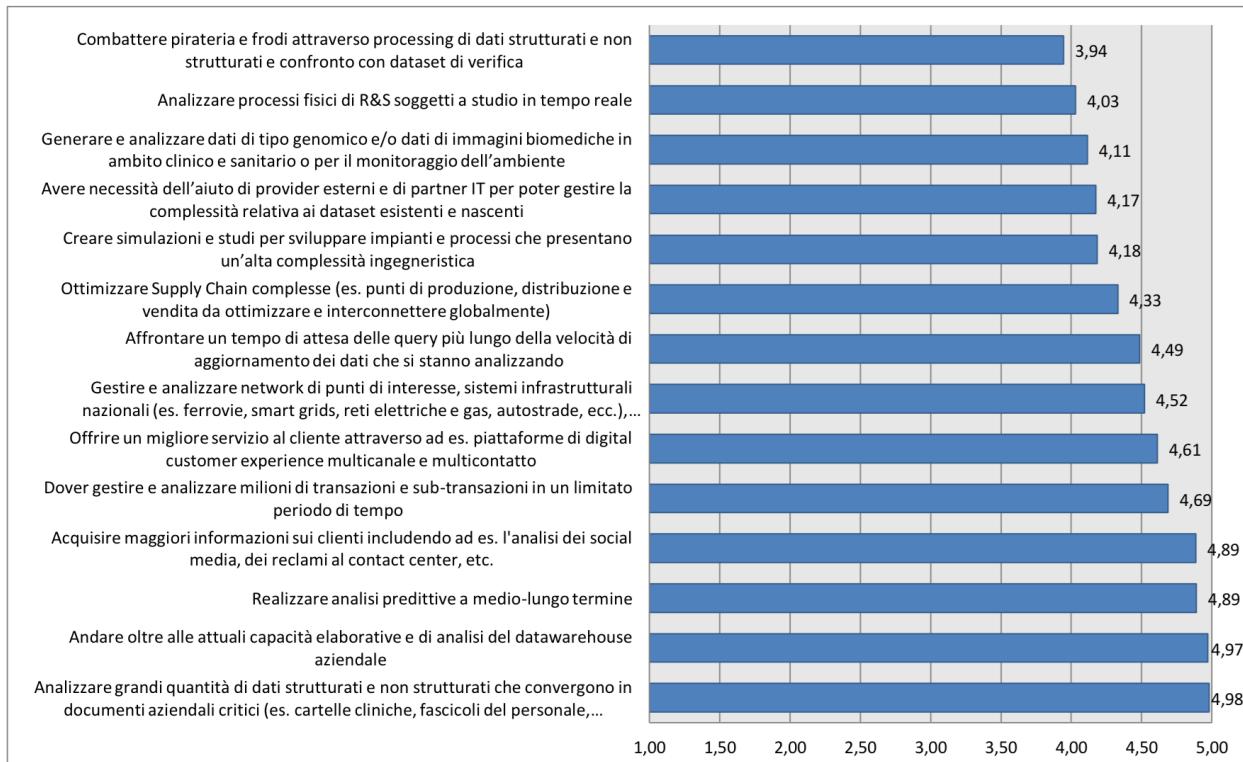
che prende il nome di *data mining* lega varie aree scientifiche: basi di dati, intelligenza artificiale (machine learning e pattern recognition) e statistica.



[img. 5] Processo di data mining

Concludendo questa breve parentesi storica e riallacciandoci alla tematica big data, vediamo realmente quali benefici le aziende vogliono trarre da quest'ultimi. L'SDA Bocconi in [37] sostiene che in termini di benefici e impatti, il valore dei Big Data si sostanzia soprattutto in "maggiore accuratezza delle analisi del comportamento del mercato e di maggiori informazioni sui clienti", "velocità delle insight a supporto delle decisioni strategiche future", "costruzione di una nuova piattaforma IT che superi i limiti del data warehousing, che permetta l'analisi di milioni di transazioni consentendo di analizzare grandi quantità di dati strutturati e non". A tal proposito anche il governo italiano [38] attraverso l'analisi dei dati di fonti informative eterogenee (catasto, adempimenti unici, anagrafe della popolazione, sistemi informativi dei tributi, dichiarazioni ICEF, ecc.) incrociate con informazioni prelevate dai social network, dai blog e attraverso l'uso della semantica e del GIS, potrà fornire una vista unificata ed integrata del cittadino dal

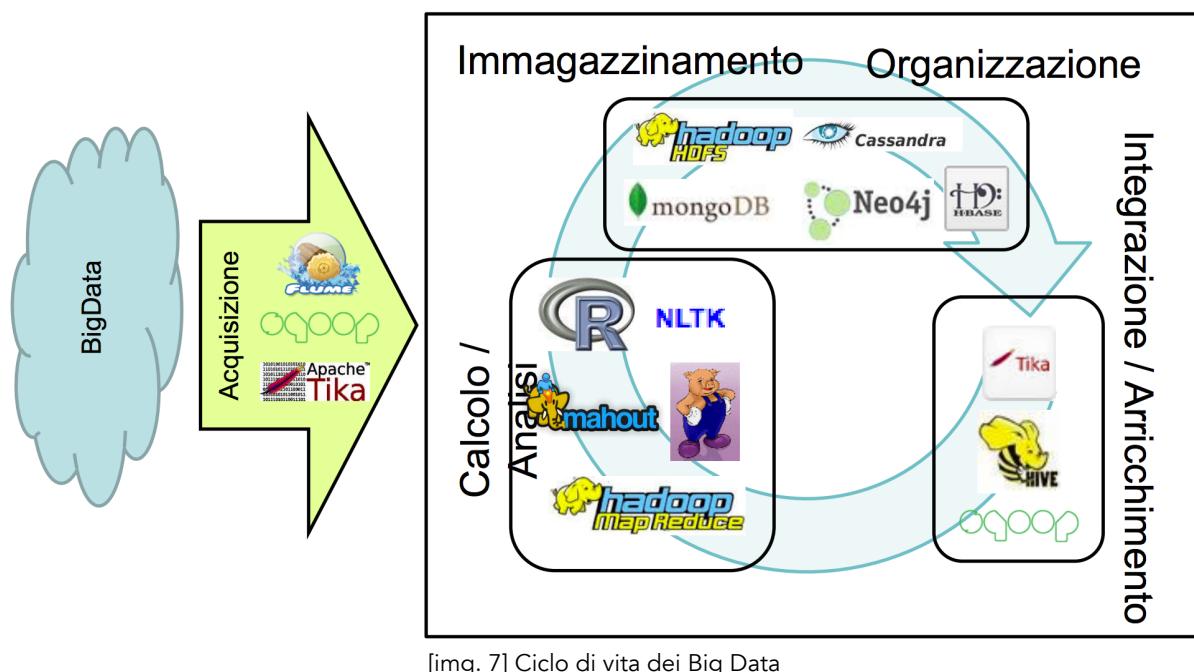
punto vista sociale e tributario. Ingegnerizzando opportunamente questi risultati, gli enti preposti all'accertamento fiscale e alla riscossione potranno raccogliere, aggregare e navigare informazioni in maniera più agevole, fruirne in maniera organica e centralizzata con notevole qualità ed affidabilità (ad esempio, si potranno incrociare i dati delle pratiche dei cittadini che usufruiscono di agevolazioni, i dati patrimoniali, nonché i dati della composizione del nucleo familiare al fine di rintracciare dichiarazioni infedeli) ed intraprendere opportune azioni mirate di lotta all'evasione fiscale.



[img 6.] Microbenefici derivanti dall'utilizzo dei Big Data

# Capitolo 2

Nel primo capitolo abbiamo visto cosa sono i big data e quali sono e i benefici che un'azienda può trarre da quest'ultimi. Vediamo adesso nel dettaglio quali sono gli strumenti per acquisirli, memorizzarli e analizzare le informazioni contenute al loro interno. Molti prodotti che presenteremo appartengono all'infrastruttura Hadoop che è lo strumento maggiormente utilizzato da parte delle aziende che lavorano con big data<sup>5</sup>. In particolare in questo capitolo ci concentreremo sugli strumenti utilizzabili per acquisire grosse moli di informazioni.



<sup>5</sup> Per un elenco completo delle aziende che hanno adottato Hadoop per analizzare i loro dati visitare il link <http://wiki.apache.org/hadoop/PoweredBy>

## 2.1 Le tecnologie di acquisizione per Big Data

Come abbiamo più volte ripetuto nel corso della trattazione, è importante per un'azienda acquisire informazioni circa la propria clientela (comportamenti d'acquisto, opinione dei prodotti) con l'obiettivo di aumentare la fidelizzazione e mettere in atto opportune politiche di cross-selling. Blog, tweet e post sui social network permettono d'altro canto alle persone di esprimere le proprie idee e percezioni sui prodotti o servizi ma bisogna essere in grado di acquisirle con le tecnologie adatte. Le tecnologie più utilizzate per acquisire i big data appartengono sostanzialmente a quattro categorie [2]:

1. API: Twitter API, Facebook API ed API di motori di ricerca (Google, Bing, ecc.)
2. Web scraping: cURL, Apache Tika
3. ETL: Sqoop
4. Stream e CEP: Apache Flume, Microsoft StreamInsight

## 2.2 API

Una miniera di informazioni (se si considera che ormai gli utenti web sono due miliardi e mezzo, gli iscritti a Facebook oltre un miliardo e 550 milioni quelli di Twitter) [17] possono essere utilizzate per capire come la pensano gli utenti relativamente ad una determinata tematica. Post su Facebook o tweet su Twitter, infatti, permettono all'azienda di accedere ad informazioni che può utilizzare per capire come si parla nel web dei propri prodotti, ma non solo. I big data in

questione possono essere utilizzati per capire chi vincerà le elezioni, cosa ne pensa la gente di Papa Francesco, ecc. Tuttavia le espressioni utilizzate nei social network sono poco comprensibili ad una macchina, per questo motivo, devono essere utilizzate degli algoritmi di intelligenza artificiale per tradurre abbreviazioni o storpature di parole (es. "beeeeeelloooo" dovrebbe essere tradotto come "molto bello", "bellissimo"), modi di dire, emoticon. Questa è la sentiment analysis, metodo di analisi applicata ai social network che sta iniziando a muovere i suoi primi passi anche in Italia. Nel prossimo paragrafo, vedremo brevemente, le API che Twitter ci mette a disposizione per poter acquisire dati utili per le nostre analisi.

## 2.2.1 API Twitter

Quadro Twitter ha lanciato nel 2009 le proprie API, ha ispirato migliaia di progetti di sviluppo, tra cui un certo numero di iniziative di ricerca. In particolare, il primo studio su Twitter è stato pubblicato nel 2010 da Kwak [7] che ha studiato le caratteristiche del social network e il suo potere come mezzo di condivisione delle informazioni; ancora, Java e altri [8], hanno presentato le loro osservazioni dei fenomeni di microblogging attraverso lo studio delle proprietà topologiche e geografiche di Twitter. Vediamo in questo paragrafo come è possibile integrare la nostra applicazione<sup>6</sup> con il social network; andare quindi alla pagina Twitter Applications Console, loggarsi attraverso le proprie credenziali e creare una nuova applicazione. Una volta creata l'applicazione nella sezione "Oauth tool" è

---

<sup>6</sup> Per poter stabilire un canale di comunicazione con Twitter e poter utilizzare le REST API, le Streaming API è necessario creare delle Twitter applications al sito <https://apps.twitter.com/>.

necessario annotare i valori Consumer Key, Consumer secret, Access token e Access token secret, utilizzabili per stabilire un canale di comunicazione con la Twitter application creata. Per i nostri test abbiamo utilizzato la PHP-Twitter-API [<https://github.com/timwhitlock/php-twitter-api>] che ci permette di astrarre quella che è l'autenticazione con il protocollo Oauth<sup>7</sup> ed abbiamo costruito un tool di sentiment analysis [img. 8] per classificare i tweet come positivi, negativi o neutrali individuando automaticamente la polarità, la soggettività e gli stati emotivi di un particolare documento o frase contenente una key di nostro interesse.

```
/*
 * Calls the Search/tweets method of the Twitter API for particular Twitter Search Parameters and returns the list of tweets that match the search criteria.
 * @param mixed $twitterSearchParams The Twitter Search Parameters that are passed to Twitter API. Read more here https://dev.twitter.com/docs/api/1.1/get/search/tweets
 * @return array $tweets
 */
protected function getTweets($twitterSearchParams) {
    $client = new TwitterApiClient(); //Use the TwitterApiClient
    $client->set_auth($this->consumer_key, $this->access_key, $this->access_secret);

    $tweets = $client->call('search/tweets', $twitterSearchParams, 'GET'); //call the service and get the list of tweets
    unset($client);
    return $tweets;
}

protected function findSentiment($tweets) {
    $datumboxAPI = new DatumboxAPI($this->datumbox_api_key); //initialize the DatumboxAPI client
    $results=array();
    foreach($tweets['statuses'] as $tweet) { //foreach of the tweets that we received
        if($tweet['metadata']['iso_language_code'] == 'en') { //perform sentiment analysis only for the English Tweets
            $sentiment=$datumboxAPI->twitterSentimentAnalysis($tweet['text']); //call Datumbox service to get the sentiment

            if($sentiment!=false) { //if the sentiment is not false, the API call was successful.
                $results[]=$tweet; //add the tweet message in the results
                'id'=>$tweet['id_str'],
                'user'=>$tweet['user']['name'],
                'text'=>$tweet['text'],
                'url'=>https://twitter.com/\$tweet\['user'\].'/status/\$tweet\['id\_str'\],
                'sentiment'=>$sentiment,
            };
        }
    }
    unset($tweets);
    unset($datumboxAPI);
    return $results;
}

```

[img. 8] Estratto di codice PHP per l'utilizzo delle API Twitter e della libreria Datumbox

Il problema in questione non è per niente banale in quanto sono richieste tecniche di machine learning e di natural language processing, per questo motivo, per i nostri test, abbiamo scelto di utilizzare la libreria Datumbox [<http://>

---

<sup>7</sup> OAuth è un protocollo aperto che permette l'autorizzazione di API di sicurezza con un metodo standard e semplice, consentendo agli sviluppatori di applicazioni di interagire con dati protetti. È possibile quindi accedere ad un service provider (es. Twitter) attraverso un consumer e, utilizzare i dati degli utenti proteggendo contemporaneamente le loro credenziali.

[www.datumbox.com/](http://www.datumbox.com/), un potente open-source framework scritto in Java che fornisce sofisticate tecniche di IA per il clustering di contenuti testuali.

### Datumbox Twitter Sentiment Analysis

Inserisci la parola sotto per eseguire l'analisi del sentimento:

Keyword: lollipop android

#### Results for "lollipop android"

Id	User	Text	Twitter Link	Sentiment
537640724771381248	Robonto	I made a wish and it didn't come true. @Android @googleNexus 4 #lollipop □ http://t.co/ZUkel4IPO	<a href="#">View</a>	negative
537640397770481664	SJT	Xiaomi's Hugo Barra: We might launch an Android One phone, MIUI with Lollipop to arrive Q1 2015 http://t.co/6aVft6alQp	<a href="#">View</a>	neutral
537640358663159808	Juan Ignacio Yarza	Android 5.0 Lollipop review: the biggest Android update. - http://t.co/sY6Q7qrRc http://t.co/01w07hmq1y	<a href="#">View</a>	neutral
537640248742645760	Domenico Loiacono	#Android Lollipop arriva su Android Wear - Wired.it http://t.co/La6Khxx2Fk	<a href="#">View</a>	neutral
537640109521526784	Israel_Banda	RT @chungo_mwila: 16 Things You Can Do in Android Lollipop That You Couldn't Do in KitKat http://t.co/wHgk7r8xBT	<a href="#">View</a>	positive
537640071772381184	Android Cdma	#AndroidCdma Xiaomi's Hugo Barra: We might launch an Android One phone, MIUI with Lollipop to arrive Q1 2015 http://t.co/ds1ErFj1x5	<a href="#">View</a>	neutral
537640071311011841	Android Pres News	#android Xiaomi's Hugo Barra: We might launch an Android One phone, MIUI with Lollipop to arrive Q1 2015 http://t.co/V8MPpbBjnfE	<a href="#">View</a>	neutral
537639952310599680	Reddit Bot	No more □ hairy heart emoji on Lollipop http://t.co/MqH4YbkUN	<a href="#">View</a>	neutral
537639408539680768	SIAM_MS	Waiting For Moto G 2nd Gen To Be In Stock Again □ #Flipkart #Android #Kitkat #Lollipop #Waiting #MotoG2ndGen #Google #Moto @Flipkart Moto	<a href="#">View</a>	positive

[img. 9] Classificazione dei tweet circa lollipop android (nuovo sistema operativo in casa Google)

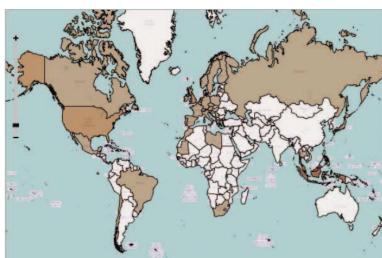
Vengono, inoltre, fornite delle REST API che permettono di utilizzare il framework senza installarlo sulla macchina. L'esempio appena descritto permette quindi di effettuare un'analisi qualitativa dei dati a disposizione, tuttavia, si può estrarre dai tweet molto di più, come ad esempio informazioni legate alla posizione geografica (potremmo voler sapere dove "si parla meno" dei nostri prodotti per intensificare campagne pubblicitarie in quei luoghi). Queste informazioni quindi, opportunamente integrate con i dati statici delle vendite, trasformate in una serie di cubi multidimensionali e il risultante data warehouse usato per una serie di social media analysis nella speranza di migliorare la commercializzazione, assistenza ai clienti e le relazioni pubbliche. Un esempio di utilizzo che sfrutta la geolocalizzazione dei tweet è discusso in [18] in cui viene descritto la potenza delle tecniche OLAP per studiare il ruolo di Twitter come mezzo di comunicazione in caso di emergenza. Gli autori hanno estratto i dati dai tweet che si estendono nelle tre ore successive al terremoto del 11 aprile 2012 in Indonesia, con almeno una menzione di un argomento presente in una tabella (vedi [img. 10])

No.	Trending Topic
1	#PrayForSumatera
2	#sumatra
3	#tsunami
4	#10favouritebands
5	Earthquake in Indonesia
6	Magnitude 8.9
7	Sumatra

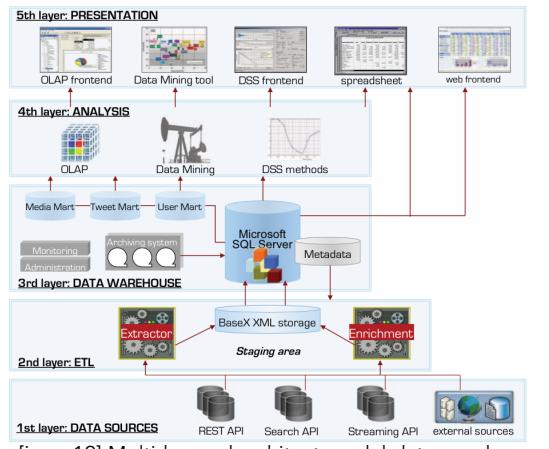
[img. 10] Parole chiave da ricercare nei tweets

Il DW twitter realizzato comprende 5 layers di base:

- il data source, in cui confluiscono i tweet.
- L'ETL (Extraction, Transformation, Loading) per bufferizzare, selezionare, arricchire (con dati provenienti da altre fonti come la geolocalizzazione) i dati semistrutturati Json<sup>8</sup> da caricare nel DW target.
- Il layer del data warehouse; gli autori hanno utilizzato a tale scopo Microsoft SQL Server che include servizi di analisi OLAP e data mining.
- Il layer di analisi (OLAP, Data Mining, DSS method).
- Il layer di presentazione.



[img. 11] Luoghi in cui si discute del maremoto in



[img. 12] Multi-layered architecture del data warehouse

<sup>8</sup> Le twitter applications forniscono le risposte alle interrogazioni nel formato JSON. Ogni oggetto contiene informazioni utili come hashtag, retweets, media, ecc. [<https://dev.twitter.com/overview/api/entities-in-twitter-objects>]

## 2.3 Strumenti di web scraping

Il web scraping (detto anche web harvesting o web data extraction) è una tecnica di estrazione dei dati da un sito web per mezzo di programmi software che simulano la navigazione umana nel World Wide Web attraverso l'implementazione a basso livello dell'Hypertext Transfer Protocol (HTTP); si può usare per confrontare prezzi online, monitorare dati meteorologici, rilevare modifiche in un sito Internet, nella ricerca scientifica, per il web mashup e il web data integration [20]. Mentre l'estrazione di tweet su Twitter o post su Facebook, come abbiamo visto nel paragrafo precedente, può essere effettuata utilizzando le API dei rispettivi social network, differente è l'estrazione di commenti su comuni pagine web come ad esempio forum e blog, infatti, questi non sono ovviamente rappresentati in un formato direttamente accessibile ai programmi ma bensì pensati per la visione ed esplorazione da parte degli utenti. Inoltre, nel www, esistono numerosissimi cicli e percorsi ridondanti, che rendono complessa la creazione di un attraversatore che visiti ogni nodo del grafo delle pagine web una e una sola volta [14]. Altre problematiche sono legate alla contestualizzazione dei commenti (ad esempio con il titolo del commento), al codice sporco, ecc. Vi sono tuttavia numerose soluzioni software che permettono, in maniera automatica o semiautomatica di sfogliare per noi le pagine web di nostro interesse e di estrarre solo le informazioni che riteniamo utili. Tuttavia non esiste una tecnica che vada bene sempre e che ci faccia sempre risparmiare tempo; sicuramente un elemento che contribuisce alla buona riuscita di uno scraper è la presenza di una sintassi utile a sfogliare la mappa di un sito tramite query. Ad esempio, il seguente link:

[www.sito.it/commenti\\_prodotti.html/prodotto=nexus4&regione=campania&comune=napoli](http://www.sito.it/commenti_prodotti.html/prodotto=nexus4&regione=campania&comune=napoli)

potrebbe essere l'URL per estrarre da un sito tutti i commenti su un determinato prodotto, in una data regione e provincia. Una struttura di questo tipo ha un'altra grande utilità: consente di individuare gli URL delle sole pagine da cui vogliamo estrarre informazioni, e non spendere tempo in processi automatici su sezioni del sito inutili per la nostra indagine. Vi sono, a tal proposito, molti strumenti che permettono l'attraversamento automatico del grafo degli ipertesti e l'utilizzo di espressioni regolari per navigare all'interno di un sito web, come lo strumento presentato nel paragrafo successivo.

### 2.3.1 cURL

cURL (Client for URLs) è un tool per trasferire dati da-a un server, utilizzando uno dei protocolli supportati (DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, ecc.) ed è stato progettato per lavorare senza l'interazione dell'utente. cURL offre una grossa quantità di trucchi utili come proxy, user interaction, upload FTP, HTTP post, connessioni SSL, cookies, file transfer resume, proxy tunneling, ecc [21].

Ecco alcune operazioni non banali che possiamo eseguire con cURL:

- specificare più URLs: `http://site.{one,two,three}.com`
- specificare sequenze: `ftp://ftp.numericals.com/file[1-100].txt`
- combinazioni delle precedenti: `http://any.org/archive[1996-1999]/vol[1-4]/part{a,b,c}.html`
- pagine che richiedono l'utilizzo di cookie: `curl -b "name=Gino" www.sillypage.com`

- recuperare documenti utilizzando il protocollo HTTPS: curl <https://www.secure-site.com>.

## 2.4 Apache Tika

È ben nota l'esistenza di un enorme divario tra dati salvati (uno studio ha previsto che la quantità di dati in tutto il mondo era di circa 1.800 Exabyte nel 2011 [10]) e informazioni riutilizzate [23]; quindi, è stato necessario creare strumenti per trovare informazioni ed estrarre nuova conoscenza da quest'ultime.

Uno degli strumenti utilizzabili in questo contesto è Apache Tika che estrae metadati e contenuti testuali da varie tipologie di documenti (come PPT, CSV, PDF) utilizzando librerie di parsing del testo [22]. In altre parole, uno strumento di knowledge management (KM) per reperire informazioni da documenti eterogenei tra loro.

Affinchè questo sia possibile è molto utile che i vari files contengano una serie di metadati, letteralmente "(dato) oltre un (altro) dato" [24] come ad esempio la scheda del catalogo di una biblioteca, la quale contiene informazioni circa il contenuto e la posizione di un libro, cioè dati riguardanti i dati che si riferiscono al singolo libro. Un altro contenuto tipico dei metadati può essere la fonte o l'autore dell'insieme di dati descritto oppure le modalità d'accesso [24]. Vediamo come, con poche linee di codice [img. 13] con Apache Tika è possibile estrarre metadati da un documento:

```
Metadata metadata = new Metadata(); ContentHandler handler = new
BodyContentHandler();
ParseContext context = new ParseContext();
InputStream test;
test = new FileInputStream("ianno-m-(cur-ic).pdf");
new PDFParser().parse(test, handler, metadata, context);

System.out.println(metadata);
```

[img. 13] Estratto di codice in java utilizzando la libreria Tika

In [23] Apache Tika è stato utilizzato per estrarre metadati da documenti e inserirli in un'ontologia. Quindi, dopo tale processo, i documenti strutturati potevano essere trovati nella suddetta antologìa con l'aiuto di una ricerca semantica, ad esempio con l'utilizzo di query SPARQL. Come si evince da [23] possono esistere dei documenti in cui i metadati sono assenti o non rispettano determinate misure di qualità, come quelle definite dal Dublin Core Metadata<sup>9</sup>, pertanto, bisogna utilizzare tecniche di text mining per estrarre i metadati direttamente dal contenuto dei documenti e procedere alla classificazione utilizzando tecniche di clustering quali il KMeans che, come presentato in [57] può essere parallelizzato utilizzando il paradigma map/reduce e quindi utilizzato in ambiente Hadoop. Apache Tika è quindi un prodotto ormai maturo utilizzato in molti lavori scientifici come ad esempio in [26] in cui è stato realizzata una banca dati on-line per migliorare il processo decisionale in unità di terapia intensiva pediatrica.

## 2.5 Strumenti ETL e ELT

In contesti di analisi di grosse quantità di informazioni, come nel caso dei data warehouse, è importante considerare il carico di lavoro necessario per manipolare, allineare ed integrare i dati provenienti da una o più sorgenti eterogenee. Un processo di questo tipo può essere più o meno pesante a seconda che i dati nelle sorgenti siano strutturati o meno, ad esempio, l'assimilazione di un file CSV in un database SQL richiede minor tempo rispetto all'estrazione di informazioni da uno o più file di log; nel secondo caso, infatti, è necessario esaminare ricorsivamente

---

<sup>9</sup> Organizzazione aperta a sostenere l'innovazione nel design e le migliori pratiche in tutta l'ecologia dei metadati [25].

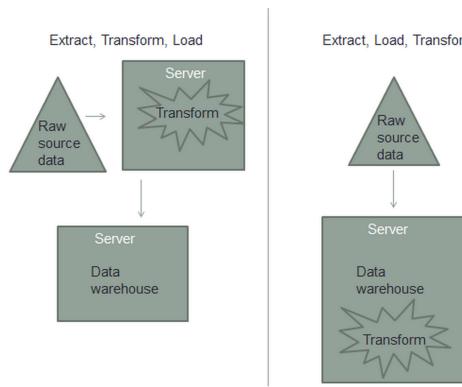
tutti i file nelle cartelle di log, estrarne il contenuto e contestualizzare le infomazioni estratte (i log ad esempio riportano solo l'ora ma non il giorno che è contenuto nel nome del file). Inoltre, anche se i dati sono strutturati o semistrutturati possono essere "sporchi" come ad esempio in una pagina web, dove è estremamente raro che risulti "chiara e ordinata", poiché è stata realizzata per dover essere letta da un browser piuttosto che da programmi che effettuano il parsing del testo. Altre problematiche, relative all'estrazione di dati dalle pagine web, potrebbero sorgere nel caso in cui il testo venga caricato in maniera asincrona (pensiamo ad esempio alla tecnologia AJAX) rendendo difficile per uno scraper, che non esegue il codice JavaScript, determinare il contenuto visto effettivamente dall'utente. Inoltre, i dati potrebbero essere ridondanti, la struttura del documento potrebbe essere in un formato particolare (ad esempio .doc), la codifica non nota, ecc.

In generale, il processo di estrazione, trasformazione e caricamento dei dati in un sistema di sintesi (ad esempio il data warehouse) prende il nome di Extract, Transform, Load (ETL) [13]. Il nome è giustificato dalla presenza di tre fasi nel processo di alimentazione del data warehouse:

- estrazione dei dati grezzi delle fonti;
- trasformazione e integrazione dei dati provenienti da fonti differenti ed eterogenee;
- caricamento dei dati puliti e integrati all'interno del data warehouse.

Nel contesto dei big data l'assimilazione può essere continua, cioè avvenire in tempo reale su dati prodotti in maniera irregolare quindi è possibile eseguire la fase di trasformazione dei dati in un secondo momento, sfruttando la nuova possibilità di memorizzare dati non strutturati come testi e file grezzi [13]. Questa soluzione definita ELT ovvero Extract, Load, Trasform prevede che la trasformazione dei dati avvenga su un "server intermediario" prima che

quest'ultimi vengano caricati nel data warehouse [15]. Quindi anziché trasformare tutti i dati prima di memorizzarli nel data warehouse (ricordiamo che gli strumenti ETL sono nati in un era in cui la capacità di memorizzazione era inferiore, più lenta e costosa), la tendenza oggi giorno è di memorizzare quanti più dati possibili, nelle forme più svariate e successivamente permettere l'analisi da parte di applicazioni di consumo differenti [16]. Per questo motivo i vendor di sistemi di gestione dei dati si sono mossi nella direzione di fornire soluzioni in grado di rendere utilizzabili i big data ai fini di estrarne informazioni; è quindi possibile utilizzare soluzioni cloud che rendono lo storage e l'analisi dei big data accessibili a tutti. Pensiamo ad esempio a Terveda ([www.tervela.com](http://www.tervela.com)) che permette un trasferimento di big data di tipo cloud-to-cloud e on-premise-to-cloud. Occorre chiarire che, in certi casi, potrebbe anche essere realizzabile l'immagazzinamento dei dati in un normale RDBMS, al costo di investire cifre elevatissime per lo storage, per ottenere la capacità di calcolo necessaria ad elaborare i dati. Tuttavia, gli investimenti potrebbero rilevarsi non giustificabili dai risultati ottenuti in termini di performance. È per questo che grandi aziende utilizzano strumenti proprietari o open source, diversi dagli RDBMS, per l'analisi della grandissima mole di dati di cui dispongono (es. Hadoop). La sfida per i sistemi di Business Intelligence è duplice: da un lato si tratta di sfruttare i big data integrandoli con il data warehouse, dell'altro è necessario individuare quali sono i dati rilevanti, separandoli dal rumore [2].



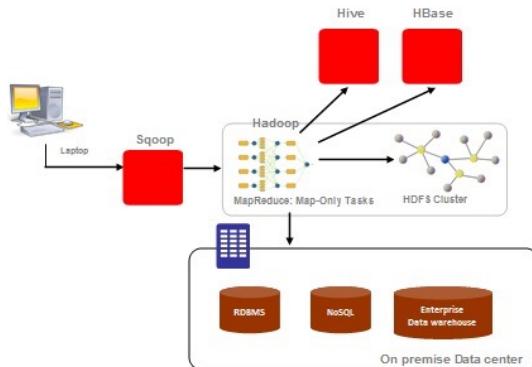
[img. 14] ETL VS ELT

Vediamo nel prossimo paragrafo lo strumento Sqoop che permette di trasferire i dati memorizzati in un database o in un data warehouse così da poter integrare le informazioni aziendali nell'infrastruttura Hadoop, che sta rapidamente diventando lo standard per l'elaborazione dei big data.

## 2.5.1 Sqoop

All'origine di Sqoop, una delle principali preoccupazioni era lo "sblocco" dei dati memorizzati negli RDBMS di un'organizzazione e il trasferimento di quest'ultimi in Hadoop [30]. Ciò ha fatto sì che molti vendor SQL fornissero plug-in e che Sqoop diventasse uno strumento ETL fondamentale per integrare i dati aziendali (conservati negli RDBMS) in Hadoop.

Sqoop, progetto top-level della Apache Software Foundation, è, in particolare, uno strumento progettato per trasferire dati tra Hadoop e database di differenti tipologie [28]. Gli utenti Sqoop hanno importato di tutto, da dataset di modeste dimensioni a data warehouse MammothDB<sup>10</sup> [30].



[img. 15] Utilizzo di Sqoop come

---

<sup>10</sup> MammothDB è un database di analisi utilizzato per costruire soluzioni "business intelligence" o "business analytics", e più frequentemente utilizzato come spina dorsale per lo sviluppo di una "Data Warehouse" (DWH) [31].

È possibile utilizzare tale strumento, ad esempio, per importare dati da un sistema di gestione di database relazionali (RDBMS) come MySQL o Oracle nel file system distribuito Hadoop (HDFS), trasformare i dati in Hadoop MapReduce, e quindi esportare i dati di nuovo in un RDBMS [28].

L'input per il processo di importazione è quindi una tabella del database. Sqoop leggerà la tabella riga per riga salvandola sull'HDFS e il processo di importazione sarà eseguito in parallelo generando più file di output (semplici file testuali, binari Avro o SequenceFiles contenenti record di dati serializzati). In [29] sono mostrati molti esempi di utilizzo del tool; la figura sottostante, ad esempio, mostra il codice necessario per importare, a partire da un database MySQL, la tabella "user\_data" in una specifica locazione dell'HDFS.

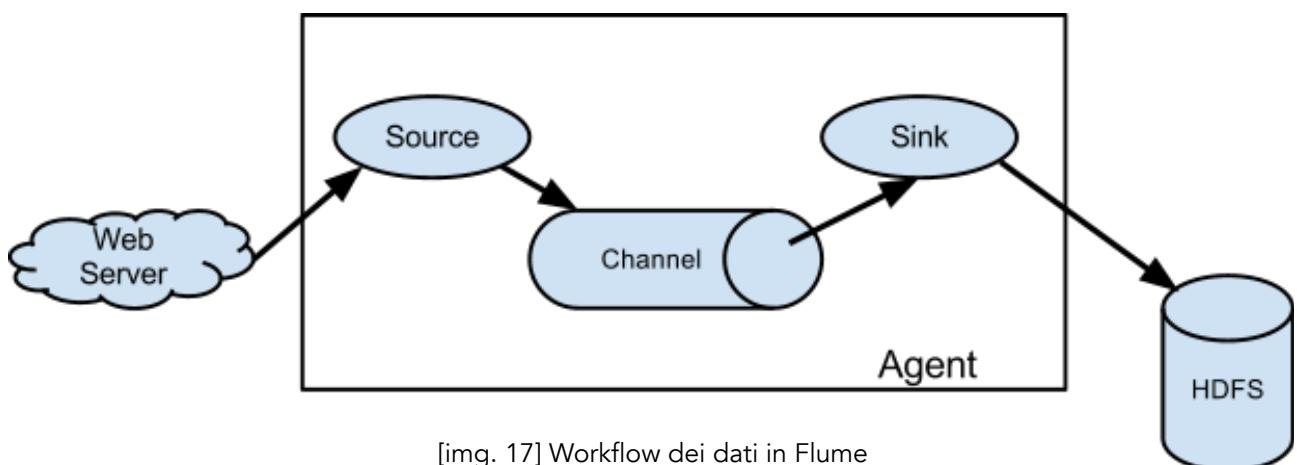
```
$ sqoop import --connect jdbc:mysql://localhost/test_db --table user_data -m 2 --target-dir /tables/userdata/
```

[img. 16] Utilizzo di Sqoop per importare un database SQL

Questo processo di importazione causerà, inoltre, la creazione di una class Java che consentirà la serializzazione e deserializzazione in e da SequenceFile in modo tale da sviluppare rapidamente applicazioni MapReduce che utilizzano i record memorizzati sull'HDFS. Dopo aver lavorato con i record importati (per esempio, con MapReduce o Hive) è possibile effettuare l'esportazione del risultato in un DBMS relazionale, per il consumo da parte di applicazioni esterne [28].

## 2.6 Apache Flume

Molte importanti applicazioni per i "big data" devono elaborare dati in arrivo in tempo reale. Apache Flume è un servizio distribuito per la raccolta, l'aggregazione e lo spostamento di grandi flussi di dati nel file system distribuito di Hadoop (HDFS) [40]. È stato progettato per raccogliere log web in tempo reale ma la fonte dati a cui attinge può essere personalizzabile: social-media-generated data (twitter, facebook), email e qualsiasi altra fonte che genera dati in maniera continua [41] in modo tale da alimentare la base di dati automaticamente.



In [img. 17] vediamo come i dati fluiscano da un client (nell'esempio è mostrato un Web Server) ad una sorgente (Source). Da questa, i dati passano attraverso uno "channel" ad un "Sink" che si occupa di trasmettere le informazioni a destinazione (ad esempio nell'HDFS).

# Capitolo 3

Abbiamo visto nel capitolo precedente quali sono le fonti e gli strumenti per acquisire i big data; vediamo in questo capitolo quali sono gli strumenti utilizzabili per memorizzarli. In prima istanza, è importante però capire perché gli strumenti tradizionali non sono adatti a manipolare grosse moli di dati e perché colossi come Facebook, Yahoo, Microsoft hanno adottato a tal proposito metodologie differenti.

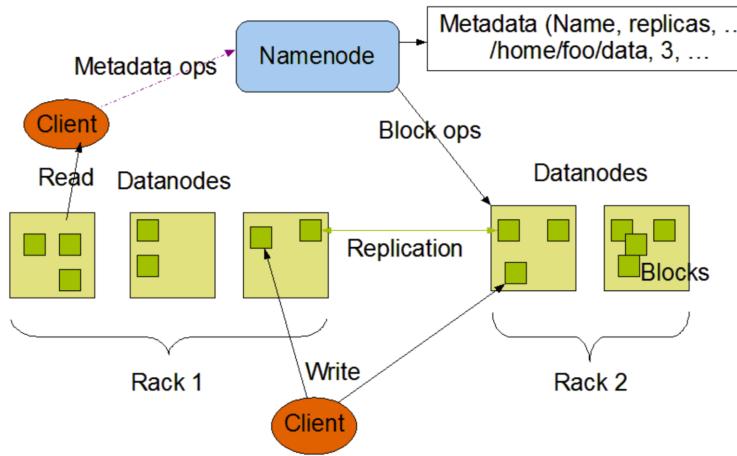
In [7] il problema viene così esemplificato: "sebbene le capacità dei dischi siano cresciute molto nel corso degli anni, il tempo di accesso (tempo per leggere i dati) non è cresciuto "abbastanza". Ad esempio un tipico disco degli anni 90 poteva salvare 1.370 MB ad una velocità di trasferimento di 4.4 MB/s; quindi poteva leggere l'intero disco in circa cinque minuti. Più di venti anni dopo, dove un terabyte è la norma, la capacità di trasferimento è di circa 100 MB/s, quindi per leggere l'intero disco ci vogliono circa 2 ore e mezza!".

E' subito evidente, alla luce di questo esempio che, avendo molti dati a disposizione, i tempi di computazione sarebbero comunque vincolati dallo spread rate del disco. Il modo ovvio per ridurre il tempo è quindi di leggere da più dischi contemporaneamente quindi, se avessimo 100 dischi, ciascuno con un centesimo di terabyte del disco iniziale, lavorando in parallelo, potremmo leggere i dati in meno di due minuti. Utilizzare solo un centesimo di disco può sembrare uno spreco ma in questo modo siamo in grado di memorizzare un centinaio di set di dati, ognuno dei quali è un terabyte, e fornire l'accesso condiviso ad essi [7]. In maniera molto semplicistica, è proprio questo il principio su cui si fonda Hadoop, ma vi è molto di più. Infatti, quando i dati sono distribuiti su più nodi della rete è molto probabile che uno di essi possa rompersi quindi è importante in queste

situazioni adottare delle politiche di fault tollerance; in sostanza i dati vengono replicati sulle varie macchine in modo tale che se una di esse si rompe la computazione non viene interrotta perché una copia dei dati è presente su un'altro nodo. Infine, a completare il kernel di Hadoop c'è il paradigma MapReduce che permette di combinare le computazioni dei singoli nodi in un unico risultato finale.

## 3.2 HDFS

L'Hadoop Distributed File System (HDFS) è un file system distribuito costruito per archiviare grandi dataset in modo affidabile e per effettuare lo streaming di quest'ultimi ad applicazioni utente [32]. Ispirato al Google File System (GFS) [4] [34] e progettato per essere eseguito su commodity hardware, è un block-structured file system gestito da un nodo particolare chiamato NameNode che mantiene i metadati circa le dimensioni, la posizione e le repliche dei blocchi contenuti in altri nodi chiamati DataNode [33]. Un client che vuole leggere un file quindi contatta il NameNode per conoscere le posizioni dei blocchi che lo compongono e legge i dati richiesti dal DataNode a lui più vicino. Nella fattispecie ogni blocco ha una dimensione di default di 64 MB per Hadoop v1.x e 128 MB per Hadoop v2.x (in modo tale da diminuire la quantità di metadati necessari per linkare i files e permettere uno streaming più veloce per la lettura) che viene replicato per fault tolerance secondo un fattore di replica configurabile dall'utente.



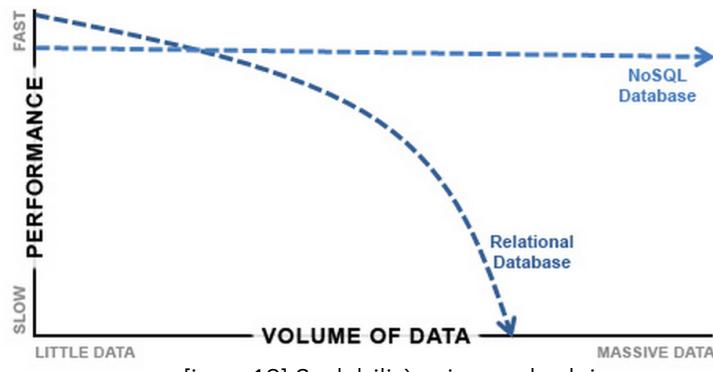
[img. 18] Architettura dell'HDFS

Alla luce di quanto detto e senza soffermarci molto su dettagli tecnici è possibile riassumere i goals fondamentali dell'HDFS nei seguenti punti [34]:

1. Hardware Failure: individuazione e ripristino automatico degli errori.
2. Streaming Data Access: l'HDFS è progettato per processare in modalità batch piuttosto che in interactive user access.
3. Large data Sets: adatto per applicazioni con grandi datasets con file che vanno dai gigabyte ai terabyte
4. Simple Coherency Model: un file, una volta creato, scritto e chiuso non ha bisogno di essere modificato. Questo assunto semplifica i problemi di coerenza dei dati.
5. Moving Computation is Cheaper than Moving Data: al fine di minimizzare la congestione della rete e aumentare la produttività è più efficiente eseguire un calcolo nel nodo in cui sono presenti i dati. Questo è particolarmente vero quando la dimensione del data set è molto grande.
6. Portability Across Heterogeneous Hardware and software Platform: HDFS è stato progettato per essere facilmente trasportabile da una piattaforma all'altra.
7. Highly fault-tolerant.

### 3.3 Limiti del modello relazionale

Nel capitolo 1 di questo survey abbiamo appreso che i dati possono essere essenzialmente di tre tipologie: strutturati, semistrutturati e non strutturati. Il modello relazionale, pensato per la prima volta da Edgar Codd nel 1970 [35], funziona bene con i dati strutturati perché è possibile convertire quest'ultimi in tabella ma, come abbiamo detto, i big data sono il più delle volte in forma non tabellare. Inoltre, [6] pur riuscendo a trasformare quest'ultimi in dati strutturati, l'unico modo per "far scalare"<sup>11</sup> un database relazione è quello di eseguirlo su un computer più potente (scalabilità verticale) ma per andare oltre è importante che la memorizzazione sia distribuita su più server. I database relazionali, però, non scalano bene orizzontalmente perché il join delle tabelle risulta essere molto difficoltoso in situazioni distribuite; i database NoSQL, invece, in cui i dati sono non strutturati, scalano bene orizzontalmente e quando i dati aumentano basta aggiungere ulteriori nodi, senza alcun rallentamento nelle prestazioni. L'infrastruttura NoSQL, infatti, è stata la soluzione per gestire alcuni dei più grandi data warehouse del pianeta del calibro di Google, Amazon e della CIA [36].



[img. 19] Scalabilità orizzontale dei

<sup>11</sup> La scalabilità di un sistema è la sua proprietà di avere un incremento di prestazioni in qualche modo direttamente proporzionale al suo aumento di risorse.

## 3.4 Database NoSQL

NoSQL ("Not Only SQL") si riferisce a varie tecnologie di persistenza dei dati differenti tra di loro che hanno come comune denominatore il fatto di non utilizzare il modello relazionale. Benché non esista una definizione formale del termine i caratteri distintivi dei database NoSql sono [7 ][36] :

- Non relazionale: la struttura dei dati non è tabellare.
- Distribuito: i dati sono replicati su più macchine
- Open source: gli autori del software permettono lo studio e le modifiche da parte di altri programmatore. Una delle conclusioni in [6], pienamente condivise dall'autore di questo survey è che nonostante molti database NoSql siano open source tale requisito non rappresenta una caratteristica distintiva di quest'ultimi. Ciò, infatti, escluderebbe prodotti come BigTable di Google e Dynamo di Amazon che hanno dato origine al movimento NoSql.
- Scalabili in orizzontale: è possibile aggiungere dei database server per ottenere un incremento delle prestazioni, in modo quasi lineare. Come abbiamo detto nel paragrafo precedente i database relazionali non scalano bene in orizzontale perché le operazioni di join non funzionano bene in ambienti distribuiti.

Prima di fornire una classificazione dei database NoSQL è bene introdurre alcuni concetti che ci consentiranno di comprendere le differenze tra i database SQL e i database NoSQL, infatti, se la caratteristica principale di una base di dati relazionale è quella di avere a disposizione sempre i dati consistenti, i database NoSql garantiscono alti livelli di disponibilità di quest'ultimi nonché della loro velocità di recupero a scapito della consistenza dell'informazione in ogni circostanza. Per tale motivo, per i database relazionali si parla di proprietà acide

(ACID) mentre per i database NoSql si comincia a parlare di proprietà base (BASE) [3].

### 3.4.1 Proprietà ACID

Uno dei termini ricorrenti nell'ambito dei database relazionali è il concetto di transizione, intesa come unità logica di elaborazione composta da operazioni fisiche che agiscono sulla base di dati [42]. Le proprietà di cui deve godere una transizione possono essere riassunte sotto l'acronimo ACID (Atomicità, Consistenza, Isolamento, Durabilità).

- Atomicità: la proprietà di atomicità garantisce che la transazione venga eseguita in tutta la sua interezza quindi, se per qualche motivo la transizione non termina (ad esempio si verifica un crash) il gestore dell'affidabilità del DBMS deve poter disfare qualunque effetto della transazione sulla base di dati.
- Consistenza: il DBMS garantisce che nessuno dei vincoli di integrità venga violato dopo l'esecuzione di una transizione.
- Isolamento: se più transazioni vengono eseguite concomitamente, il DBMS garantisce che ognuna di essa sia eseguita indipendentemente dalle altre.
- Durabilità: gli effetti di una transazione che ha terminato correttamente la sua esecuzione devono essere persistenti nel tempo ovvero non possono andare perse a causa di qualche errore successivo.

Alla luce di tali proprietà e considerando l'enorme quantità di dati che oggi viene generata tali requisiti risultano essere "eccessivamente stringenti" in applicazioni che per garantire l'affidabilità presuppongono la ridondanza dei dati. Per questo motivo nel 2000 Eric Brewer [3][6], in contrapposizione alle proprietà ACID e alla luce del teorema CAP (da lui stesso teorizzato) introdusse le proprietà BASE.

### 3.4.2 Teorema CAP e proprietà BASE

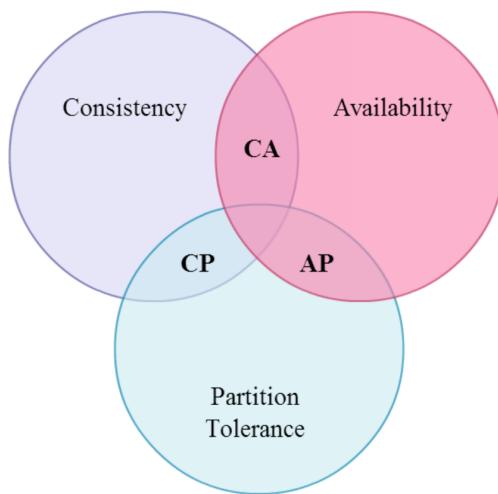
In un sistema distribuito e alla luce delle seguenti tre proprietà:

- Consistency (Consistenza): tutti i nodi, in seguito ad una scrittura, vedono gli stessi dati nello stesso momento.
- Availability (Disponibilità): il sistema risponde a tutte le richieste e quindi i dati risultano sempre accessibili.
- Partition tolerance (Tolleranza al Partizionamento): il sistema continua ad operare anche se alcune sue parti rimangono isolate dalle altre (ovvero se il grafo che rappresenta il sistema è disconnesso);

il teorema CAP asserisce che:

"possono essere soddisfatte contemporaneamente solo due di queste proprietà.

*Bisogna, quindi, scegliere quali prediligere in base alle proprie necessità"*



[img. 20] Rappresentazione grafica del teorema

E' quindi possibile classificare i database in tre categorie [img. 20]: CA, CP e AP. [6]

A fronte del teorema CAP quindi i database di tipo NoSQL, per poter essere scalabili come richiesto, devono sacrificare delle proprietà e dunque non possono in alcun modo aderire strettamente al modello ACID [43]. Per questo motivo, la maggioranza di essi segue una logica operazionale denominata BASE:

- Basically Available: garantire sempre la disponibilità dei dati.
- Soft-state: il sistema può cambiare lo stato anche se non si verificano scritture o letture.
- Eventual consistency: la consistenza può essere raggiunta nel tempo.

Si pongono quindi in primo piano le prestazioni e la disponibilità dei dati, a fronte di una minore consistenza. Quest'ultima viene definita "eventuale" in quanto non è possibile garantirla sempre a priori, ma è comunque presente in una forma più debole ovvero "prima o poi la base di dati diventerà consistente" [6].

### 3.4.3 Categorie di database NoSQL

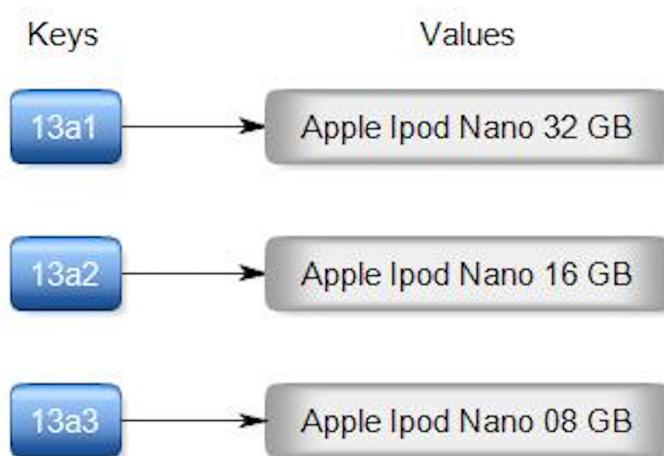
In questo paragrafo presenteremo le varie tipologie di database NoSQL. Benché ne esistano numerose in accordo a [6] e [44] abbiamo deciso di suddividerle in quattro categorie di base, ognuna adatta ad un particolare tipo di attività.

- (1) Key-Value stores; (2) Document databases; (3) Column oriented databases; (4) Graph databases.

Per ogni categoria forniremo un esempio anche se alcuni database potrebbero essere collocati in più di una categoria. Infatti, spesso un database NoSQL non utilizza un solo modello di dati in modo rigoroso: in molti casi quello che succede

è che vengono utilizzati dei modelli composti che assomigliano di volta in volta all'uno o a all'altro modello [6].

1. Key-Value stores: in tali DBMS gli *items* si presentano come delle coppie chiave/valore. Le chiavi, la cui unicità è realizzata con funzioni di *hash*, sono generalmente di tipo alfanumerico mentre i valori possono essere di diversi tipi; è importante sottolineare a tal proposito che alcuni database Key-Value oriented non prendono in considerazione il tipo e l'input viene spesso chiamato *blob (binary large object)*, per indicare un oggetto arbitrario, che il database non ha bisogno di interpretare [6]. La semplicità di tale tipologia di DBMS li rende estremamente scalabili (le coppie chiave valore possono essere distribuite tra i vari servers) e per tanto ideali per la gestione dei profili utente e il recupero dei nomi di prodotti. Per questo motivo, Amazon, uno dei più grandi e-commerce al mondo ha realizzato Dynano [45], un KV-DBMS per la gestione del corretto dei propri prodotti. Quest'ultimo è in grado di scalare, fino a picchi di carico estremi : 3 milioni di casse attive in un unico giorno e migliaia di sessioni contemporaneamente attive. Altri esempi di database Key/value sono: Voldemort (LinkedIn); Redis; BerkeleyDB; Riak.



[img. 21] Tipiche coppie chiave-valore

2. Document databases: ispirato a Lotus Notes<sup>12</sup> tale tipologia di database è stata progettata per gestire ed archiviare documenti [6][35][47]. Questo modello di dati può essere interpretato come un caso particolare del modello chiave valore infatti è possibile aggiungere ad ogni documento nuovi attributi (colonne) alla collezione di coppie chiave valore. Per la presentazione di tali strutture è molto utilizzato il formato JavaScript Object Notation (JSON) anche se altri linguaggi come di markup quali XML, YAML possono essere utilizzati. Il punto di forza dei document database è quindi la mancanza di uno schema fisso nella struttura che permette ai documenti di avere strutture e quindi attributi arbitrari; una singola colonna può contenere centinaia di attributi, e il numero e il tipo di attributi salvati possono variare da riga a riga. Inoltre, a differenza di semplici database chiave-valore, sia le chiavi che i valori (nei primi le ricerche possono riguardare solo le chiavi) sono ricercabili nel documenti.



[img. 22] Tipica struttura di un database  
orientato ai documenti

<sup>12</sup> Lotus Notes è il client applicativo ed e-mail di Domino, un software collaborativo client/server prodotto dalla divisione Lotus di IBM. Oltre ad essere utilizzabile come client di posta elettronica, browser internet, calendario/gestore di prenotazioni risorse può essere utilizzato per lo sviluppo di database composto da moduli [46] .

Alla luce di quanto detto, risultano evidenti i vantaggi di una tale struttura rispetto a quella utilizzata in un database SQL: per esempio, consideriamo un documento che contiene i dati relativi ad una persona ed i cui attributi sono: name = "John Doe" e webpage = "www.example.net". Consideriamo poi un altro documento con i seguenti attributi: name = "Jane Doe", e-mail = jane@example.net e children= ("Sarah", "Catherine", "Marcus"). In un database SQL, ognuno di questi attributi dovrebbe essere definito in una tabella con i seguenti campi: name, webpage, e-mail, child 1, child 2, child 3. Ovviamente per John, i campi e-mail e figli sono vuoti (NULL), mentre per Jane, il campo webpage è vuoto. Questo comporta un overhead inutile [6]. Concludendo i database orientati ai documenti sono quindi l'ideale per gestire collezioni di big data quali documenti semistrutturati de-normalizzati che richiederebbero molti campi nulls come ad esempio entità prodotti o clienti. Esempi di Document Database sono: CouchDB (JSON); MongoDB (BSON) [44].

3. Column oriented databases: i database che appartengono a questa categoria, tutti ispirati a Big Table [5] di Google introducono il concetto di column-family; ad ogni famiglia di colonne in particolare, possono essere associate un numero arbitrario di colonne in relazione ad una determinata row key (che identifica l'aggregato) . Per quanto possa sembrare che questa tipologia di database ricordi le tabelle relazionali, in realtà c'è una differenza sostanziale: nel column-family, le righe non hanno le stesse colonne e le colonne possono essere

**CF : anagrafica**

*nome → Sara / cognome → Piccolo / genere → Femminile*

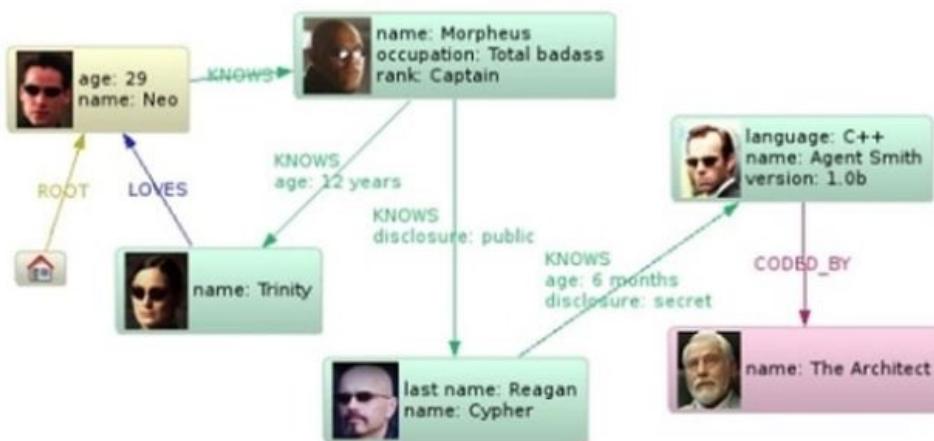
**CF : anagrafica**

*nome → Arturo / cognome → Collodi / genere → Maschile / eta → 39*

[img. 23] Columns family

aggiunte alle righe in qualsiasi momento, senza doverle aggiungere alle altre righe della stessa famiglia. A tal proposito in [img. 23] vediamo come ogni record contenga informazioni diverse per la column-family “anagrafica” [3]. Il vantaggio di tale struttura è evidente: quando un determinato valore non esiste per una certa colonna, a differenze di quello che accade per i RDBMS, possiamo evitare di sprecare spazio con valori *null*. In generale, quindi il modello di archiviazione prevede l’utilizzo di coppie (colonna/valore) e possiamo quindi vederlo come un’evoluzione dei Key-Value store, i dati infatti vengono organizzati anche in questo caso in un hash table [36]. Esempi di database column-oriented sono Cassandra utilizzato dal social network Facebook e HBase, database open source sviluppato dalla Apache Software Foundation che emula Big Table di Google. Quest’ultimo rappresenta un tassello fondamentale dell’architettura Hadoop e per questo, verrà approfondito nel prossimo paragrafo.

4. Graph databases: tale tipologia di database è specializzata nella gestione efficiente di dati fortemente legati tra di loro poiché la struttura a grafo ben si presta ad applicazioni in cui vi sono molte relazioni. Le entità sono rappresentate come nodi a cui sono associate delle proprietà; le relazioni, a loro volta, collegano ogni nodo e godono di specifiche proprietà [6]. Strutture di questo tipo permettono il passaggio tra un nodo ad un altro (graph traversal), meno dispendiose di rispetto alle join ricorsivi degli RDBMS e sono quindi molto utilizzati nel campo del Social Network: Twitter ad esempio, attraverso FlockDB [47], ottimizzato per liste di adiacenza di grandi dimensioni, gestisce le relazione tra gli utenti e il servizio che permette di seguire un particolare tweet. Altri Graph databases sono Neo4j e GraphDB entrambi basati su grafi diretti e multirelazionali.

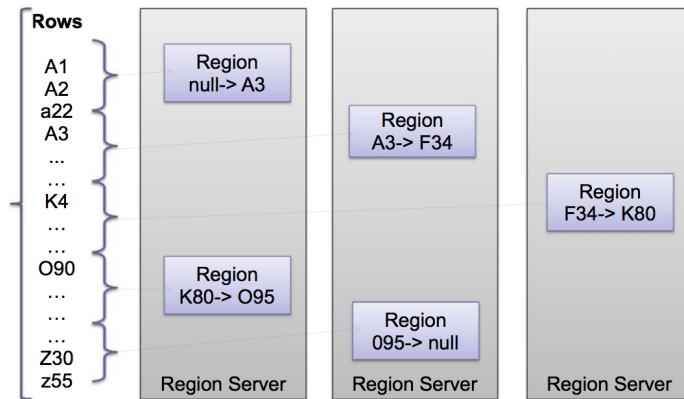


[img. 24] Rappresentazione del grafo di un social Network

### 3.5 Apache HBase

Apache HBase è il database distribuito di Hadoop, altamente scalabile e costruito appositamente per memorizzare big data [48]. E' quindi l'ideale quando si ha bisogno di accedere in modo casuale, realitime (sia in lettura che in scrittura) a grosse moli di dati memorizzati in grosse tabelle contenenti bilioni di righe e milioni di colonne; è infatti utilizzato da Twitter, Yahoo!, Adobe, ed anche Facebook, nel 2010, lo ha adottato per servire la sua piattaforma di messaggi [7]. Come accennato nel paragrafo precedente, HBase è un database non relazionale di tipo colonna che sfrutta l'HDFS per memorizzare le proprie informazioni; i dati sono quindi distribuiti su vari nodi (minimo 5 altrimenti non si ottengono buoni risultati) in un cluster composto da commodity hardware [32]. E' infatti la versione open source del database Big Table, costruito su misura da Google per il Google File System [5]. La struttura di una tabella di HBase è basata su un insieme di famiglie di colonne (vedi Column oriented databases al paragrafo 3.4.3), definite

alla sua creazione e modificate raramente, ognuna contenente diverse colonne che possono essere aggiunte in tempo reale durante il normale funzionamento del database.



[img. 25] Distribuzione dei dati tra i vari servers

L'esistenza delle famiglie di colonne è dovuta a motivi di efficienza: diverse famiglie di colonne vengono gestite da diversi server e quindi, se un'applicazione tende ad usare solo una certa famiglia alla volta , è possibile ridurre notevolmente il numero di richieste ai nodi [14]. Ciò rende HBase una cattiva soluzione per operazioni transazionali tipiche degli RDBMS in cui sono necessarie opzioni di "group by", "join" ecc. mentre un'ottima soluzione quando vi è necessità di iterare su valori random che presentano tutti lo stesso tipo [7]. I dati quindi sono organizzati in righe (spezzate sui vari nodi) dove la chiave è un array di byte e le colonne sono raggruppate in famiglie; è inoltre presente un sistema di versionamento delle celle basato sui timestamp. A patto che tutti i nodi sui cui è distribuito il database abbiano gli orologi del sistema sincronizzati tra loro (risultato ottenibile sfruttando un time server, servizio già disponibile a livello del kernel Linux), è possibile determinare con esattezza l'ordine di creazione di varie versioni e eliminazioni di ogni cella da parte di più nodi e il suo stato nel corso del tempo [14].

Row Key	Time stamp	Name Family		Address Family	
		first_name	last_name	number	address
row1	t1	<b><u>Bob</u></b>	<b><u>Smith</u></b>		
	t5			10	First Lane
	t10			30	Other Lane
	t15			7	<u>Last Street</u>
row2	t20	<b><u>Mary</u></b>	Tompson		
	t22			77	<b>One Street</b>
	t30		<b><u>Thompson</u></b>		

[img. 24] Rappresentazione logica di come i valori sono memorizzati

Le query al database possono quindi avvenire sia specificando la riga ed eventualmente quali famiglie di colonne o singole colonne recuperare (non si possono filtrare i contenuti in base a query sulle colonne) oppure filtrando i risultati in base al timestamp di ogni riga (potremmo essere ad esempio interessati solamente alla riga “più recente”). Concludiamo infine questo capitolo introducendo gli argomenti oggetto di analisi del successivo: MapReduce, che può utilizzare HBase sia come sorgente, sia come destinazione per i suoi calcoli e Hive/Pig che possono essere utilizzati per l’interazione con il NoSQL DB in questione.

# Capitolo 4

Nei capitoli precedenti abbiamo visto quali sono gli strumenti utilizzabili per acquisire i big data e come memorizzarli efficientemente in strutture diverse dagli RDBMS. Abbiamo appreso, infatti, che a fronte della loro natura non strutturata, quest'ultimi mal si prestano ad essere memorizzati nei database relazionali; a tal proposito abbiamo parlato dell'Hadoop Distribuite File System (HDFS) e dei database NoSQL. Vediamo in questo capitolo quali sono gli strumenti per processare ed analizzare grosse moli di dati utilizzando il paradigma MapReduce e linguaggi di più alto livello quali Hive e Pig.

## 4.1 Perché MapReduce?

A partire dagli anni 2000, Google iniziava ad affrontare una sfida molto ardua: Internet diventava sempre più popolare, i siti web sempre più numerosi e la sua missione di organizzare le informazioni a livello mondiale sempre più difficile. Nessun software disponibile in commercio era in grado di gestire il volume di dati da elaborare e presto, le infrastrutture esistenti, stavano per raggiungere i loro limiti. Gli ingegneri di Google così hanno progettato e costruito un'infrastruttura di elaborazione per risolvere questo problema: il Google File System [32] (a cui è ispirato l'HDFS, vedi paragrafo 3.2) per un'archiviazione affidabile e scalabile e MapReduce, per dividere il lavoro tra un gran numero di server in parallelo. Per quanto concerne MapReduce, benché negli anni diversi sviluppatori avessero lavorato all'implementazione di modelli di computazione special-purpose per grosse moli di dati [49] c'era bisogno di un modello che astraesse i dettagli scabrosi della programmazione parallela, della distribuzione dei dati e della

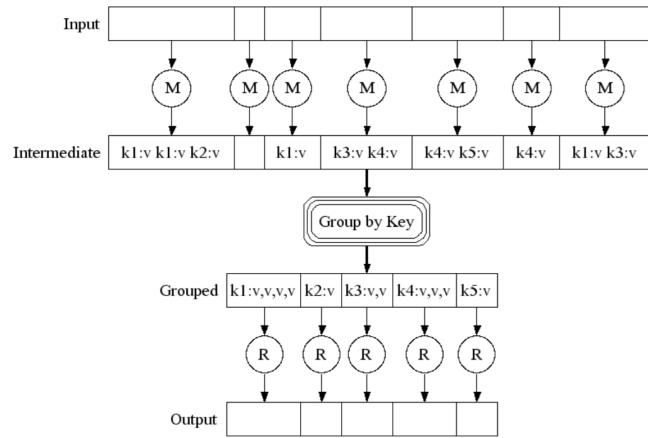
gestione delle failures. MapReduce, a tal proposito, consente a programmatori senza alcuna esperienza di parallel-computing di utilizzare le risorse di un grosso sistema distribuito (composto da un cluster di commodity hardware) e di processare molti terabyte di dati attraverso migliaia di macchine ogni giorno. Nel 2004, l’azienda di Mountain View pubblicò un documento accademico descrivendo il suo lavoro; poco prima, nel 2002 [7], un noto sviluppatore di software open source di nome Doug Cutting aveva sviluppato la prima versione di Nutch, applicazione da usare su tutta la rete per fare crawling, però, quest’ultima presentava dei problemi di scalabilità. A tal proposito Cutting modificò il codice del crawler relativamente alla raccolta dei dati, basando la sua nuova implementazione su MapReduce proposta da Google, chiamando il nuovo sistema Hadoop riferendosi al pupazzo-elefante che apparteneva al suo figlio. Da lì a poco “l’elefante” divenne molto popolare (è utilizzato ad esempio da Yahoo!, Facebook, Adobe, ecc.) e il framework MapReduce lo standard più diffuso per le elaborazioni massive di grosse moli di dati. Vediamo a tal proposito nel prossimo paragrafo i dettagli implementativi di tale paradigma [49] e successivamente nel paragrafo 4.3 il funzionamento di quest’ultimo con l’HDFS presentato nel capitolo precedente.

## 4.2 Il paradigma MapReduce

L’implementazione di MapReduce proposta da Google [49], in accordo alle primitive map e reduce, presenti in linguaggi funzionali quali Lisp [50], si compone di due funzioni: una di map e una di reduce. In particolare:

- Map prende in input una coppia  $\langle K, V \rangle$  e restituisce in output una lista di coppie intermedie  $\langle K', V' \rangle$ .

- Reduce prende in input una lista di coppie intermedie  $\langle K, V \rangle$  con la stessa key e restituisce in output una lista di coppie  $\langle K', V' \rangle$ .

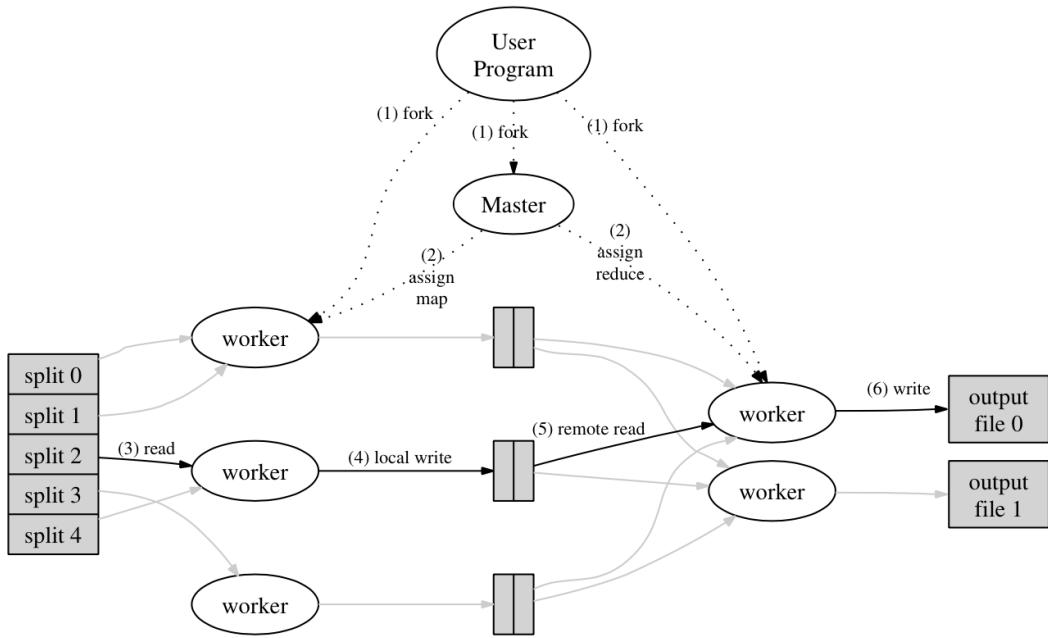


[img. 26] Execution overview

Il programma quindi, tramite la libreria MapReduce divide i dati in input e vengono attivate varie istanze del programma su un cluster di macchine [img. 26]; una di queste istanze, detta master, assegna i task alle altre macchine dette slaves o workers. Un worker che svolge un task di map in particolare, legge il blocco di input assegnato, estrae da esso le coppie  $\langle K, V \rangle$  e le passa una alla volta alla funzione `map()` (definita dal programmatore) la quale, come risultato della computazione produce delle coppie intermedie  $\langle K', V' \rangle$ . Il master successivamente, alla luce dei dati prodotti dai map worker, assegna ad un reduce worker una regione di dati da ridurre. Un reduce task quindi raggruppa tutte le coppie  $\langle K, V \rangle$  in modo tale che le chiavi siano uniche:

$(K1',v1',v2',\dots,vn'), \dots, (K1'',v1'',v2'',\dots,vm'')$ .

Ogni gruppo quindi viene passato alla funzione `reduce()` definita dal programmatore.



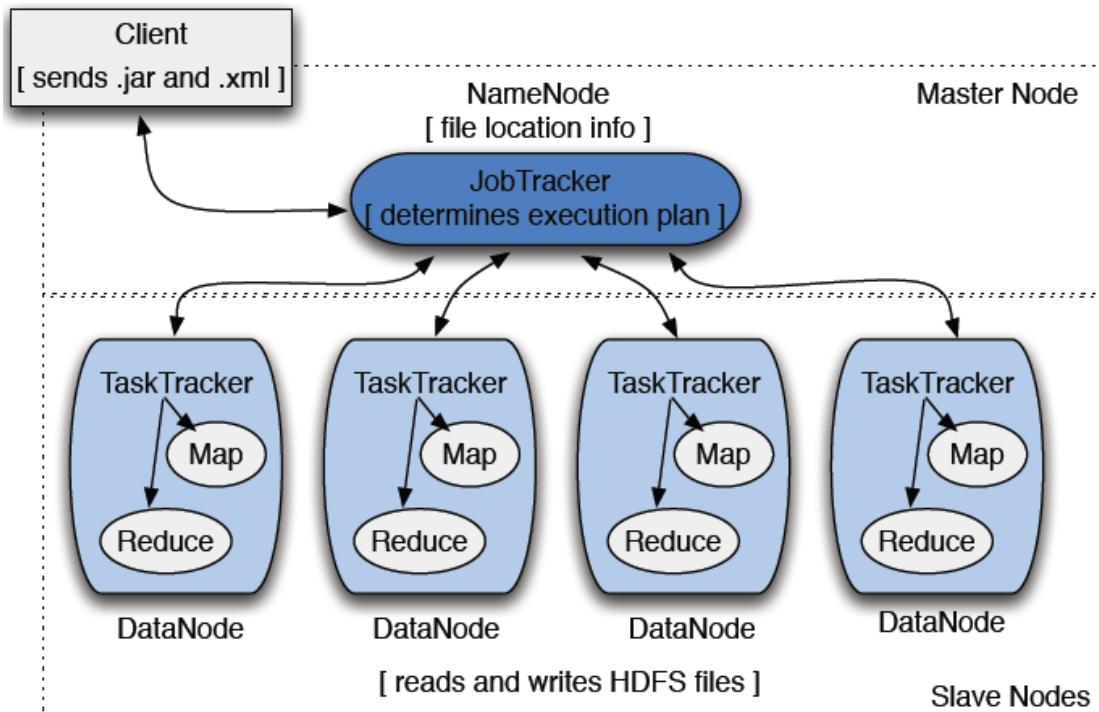
## 4.2.1 Hadoop MapReduceFramework

Nel capitolo precedente abbiamo visto come l'HDFS di Hadoop sia formato essenzialmente da due daemons: il Name Node e il Data Node. Nel paragrafo precedente, inoltre, abbiamo introdotto il paradigma MapReduce ideato da Google e abbiamo detto che quest'ultimo è stato "copiato" e adottato dall'Apache Software Foundation ed è parte integrante del progetto Hadoop. Vediamo in questo paragrafo come l'Hadoop MapReduceFramework [7] interagisca con gli HDFS demons. Senza entrare molte nel dettaglio ciò che accade può essere sintetizzato nei seguenti punti:

- Un client sottomette dei jobs MapReduce al JobTracker
- Il JobTracker che risiede nel NameNode orchestra il lavoro assegnando i vari Job map e reduce ai TaskTracker. D'altronde, il JobTracker risiede nel NameNode

che, come abbiamo detto nel paragrafo 3.2, contiene "la mappa" di come i dati sono distribuiti all'interno del cluster.

- Il TaskTracker, responsabile di eseguire i jobs di map e reduce, comunica al JobTracker il progresso riguardante i jobs che gli sono stati assegnati.



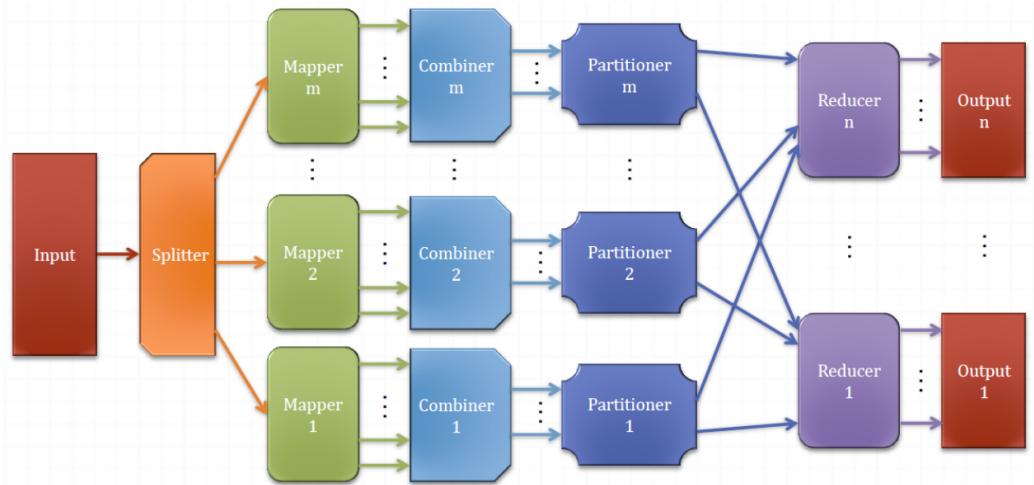
[img. 28] Hadoop's demons

Alla luce del modello presentato nel paragrafo precedente anche MapReduce di Hadoop processa coppie chiave valore. In particolare il componente Splitter determina come dividere l'input in più parti; l'input diviso quindi, viene dato in

[img. 29] Anatomia di un programma  
MapReduce in Hadoop

pasto ai vari map task che, come abbiamo detto, producono delle coppie intermedie chiave-valore. A questo punto, in ogni nodo (prima che quindi i dati vengano propagati sulla rete per essere inviati al reduce) le coppie intermedie chiave-valore possono essere inviate ad un componente opzionale chiamato

Combiner. Tale componente, definito "local reducer" [7], perché si comporta come un reducer locale, viene eseguito sulla stessa macchina che ha effettuato il map task. Il suo obiettivo è quello di effettuare delle operazioni intermedie ad ogni gruppo di record aventi la stessa chiave quindi, è importante ricorrere a tale tipo di componente opzionale per non prorogare troppi dati sulla rete, per incentivare la località e risparmiare bandwith. A questo punto i dati (sia che essi provengano dalla fase di map o dal combine), vengono inviati ad un componente chiamato Partitioner il quale stabilisce il criterio con cui i record verranno partizionati tra i vari Reducer. Il Reducer, infine, dopo aver ricevuto la propria porzione di dati dal Partitioner e aver ordinato i records ricevuti in base alla chiave, sintetizza i gruppi di record aventi la stessa chiave:  $\langle K, \text{list}(V) \rangle \rightarrow \text{list}(\langle K', V' \rangle)$ . I record prodotti vengono quindi salvati sull'HDFS.



## 4.2.2 MapReduce, un esempio di utilizzo: WordCount

WordCount rappresenta sicuramente l'esempio di utilizzo più popolare del framework MapReduce, è infatti presentato sia sul sito ufficiale [52] che su altri; il programma in questione, infatti, ben si presta ad essere utilizzato con il paradigma oggetto di analisi (ricordiamo a tal proposito che non tutti gli algoritmi sono "parallelizzabili" con MapReduce). In particolare WordCount legge più file di testo e conta le occorrenze di ogni parola; vediamo al tal proposito il codice cercando di comprendere come sono organizzate le funzioni di map e di reduce.

- Funzione di map [img. 30]: prende in input una linea di testo (ricordiamo che è lo Splitter a dividere equamente il testo tra i vari mapper) e la divide in parole: emette quindi una coppia <word,1> per ogni parola analizzata. In particolare sia "word" che "1" non possono essere rispettivamente string e int ma devono utilizzare dei wrapper tipici del framework, che permettono in primis di serializzare e deserializzare gli oggetti che incapsulano poiché, ricordiamo, MapReduce deve far "fluire" i dati tra i vari nodi nella rete.

```
public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    @Override
    public void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken());
            context.write(word, one);
        }
    }
}
```

[img. 30] Funzione di map in WordCount

- Funzione di reduce [img. 31]: somma i contatori di ogni parola ed emette una coppia <word,sum>.

```

public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {

    @Override
    public void reduce(Text key, Iterable<IntWritable> values, Context context)
        throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable value : values) {
            sum += value.get();
        }

        context.write(key, new IntWritable(sum));
    }
}

```

[img. 31] Funzione di reduce in  
WordCount

Come evidenziato nel paragrafo precedente, oltre alla funzione di map e reduce può essere utilizzato la funzione combine; nell'esempio appena mostrato, infatti, sarebbe stato più efficiente far fluire sulla rete anzichè, ad esempio, le coppie <word,1>, <word,1>, ..., <word,1> semplicemente <word,3>. Ciò avrebbe ridotto il consumo di bandwitch senza far fluire "troppi dati sulla rete".

## 4.2.3 Hive e Pig

Benchè l'esempio presentato nel paragrafo precedente possa sembrare molto semplice, alcuni jobs MapReduce richiedono molto codice di media complessità (sicuramente molta di meno di quanto ne richiederebbe un programma parallelo scritto ad esempio con RMI); per tale motivo possiamo ricorrere a linguaggi di più alto livello quali Hive e Pig che ci permettono di astrarre i jobs map/reduce utilizzando linguaggi query-based sicuramente più familiari ai non addetti ai lavori.

In particolare, Hive nacque come strumento interno a Facebook [54] per

operazioni di riepilogo, business intelligence, machine learning poiché gli strumenti data warehouse tradizionali costruiti sugli RDBMS cominciavano a mostrare i propri limiti (alcuni job di analisi dei dati richiedevano più giorni per l'elaborazione). Quindi, alla luce della crescente mole di dati prodotta dal social network (da 15TB nel 2007 a 2PB oggi!) e sfruttando le potenzialità del framework Hadoop MapReduce gli ingegneri di Facebook costruirono un tool ad-hoc per processare i propri dati. Successivamente, Hive fu donato alla ASF ed oggi è parte integrante del framework Hadoop. È in particolare [53] un software data warehouse che facilita l'esecuzione di query permettendo di definire una struttura sui dati non strutturati e interrogarli utilizzando un linguaggio SQL-like chiamato HiveSQL; allo stesso tempo, inoltre, quando è inefficiente o inopportuno esprimere tali job utilizzando la logica di HiveQL il linguaggio permette di inserire job map/reduce personalizzati utilizzando il paradigma illustrato nel paragrafo precedente. Essendo quindi molto simile a un linguaggio per l'interrogazione di database relazionali, come per l'SQL è possibile creare delle tabelle con il comando CREATE TABLE ed eseguire delle query su di esse memorizzando i risultati in file sull'HDFS o come tabelle nel database HBase. Tuttavia a differenza di quello che accade per i tipici RDBMS che, come abbiamo detto lavorano solo su dati strutturati e quindi "prendono in input" dati con una specifica estensione, Hive non ha bisogno di leggere o scrivere dati in "Hive format" ma lavora bene con qualsiasi tipo formato in quanto siamo noi ad imporre una struttura ai dati oggetto di analisi. Ad esempio con sole tre linee di codice [img. 32] è possibile costruire il programma WordCount illustrato nel paragrafo precedente e sarà compito di Hive costruire i jobs map/reduce responsabili di spartire l'input e assegnarlo ai jobs di map e di reduce.

```
SELECT word , COUNT(*) FROM input  
LATERAL VIEW explode(split(text , ' ')) lTable as word  
GROUP BY word;
```

[img. 32] WordCount Hive based

Similmente a Hive, Pig [56] è una piattaforma sviluppata da Yahoo! e mantenuta dalla fondazione Apache per definire delle operazioni MapReduce da eseguire su Hadoop con un linguaggio di alto livello ispirato all'SQL chiamato *Pig Latin*. A differenza di Hive però, che è più orientato all'interrogazione dei dati [55], la così detta presentation phase (il team di HIVE a tal proposito ha iniziato a lavorare all'integrazione di quest'ultimo con strumenti di BI tramite interfacce come ODBC), Pig è, invece, uno strumento ETL e quindi di data preparation. A tal proposito, Pig è lo strumento ideale che, a partire da dati grezzi, può essere utilizzato per ripulire, legare (operazioni di join) il nostro dataset con altre fonti. Per questo motivo utilizzando delle UDF<sup>13</sup> è possibile specificare una sequenza di "data transformations" come merging di dataset, filtraggio di informazioni e altre funzioni specifiche definite dall'utente. Anche quest'applicativo, come Hive, utilizza il framework MapReduce come linguaggio di basso livello e quindi automaticamente le query in *Pig Latin* saranno convertite in job map/reduce. A tal proposito è importante sottolineare che essendo *Pig Latin* un linguaggio imperativo e quindi le operazioni sono definite in sequenza, Pig, internamente può alterare l'ordine dei nostri job e parallelizzarli in maniera tale che diversi nodi lavorino a diverse fasi dell'esecuzione migliorando le prestazioni [14]. I programmatore, inoltre, possono espandere le funzioni scrivendo i propri plug-in in Java, Python o Javascript [56] e chiamandoli all'interno di un codice Pig Latin. Concludendo, come sottolineato in [55], benché i due strumenti descritti in questo

---

<sup>13</sup> User Defined Functions

paragrafo possano sembrare uguali (al di là della sintassi) ognuno di loro "è più adatto" ad implementare le funzionalità di un particolare layer della data warehouse architecture.

# Conclusioni

Nel corso della trattazione abbiamo in prima istanza fornito le nozioni di base che contraddistinguono i big data: velocità, varietà, volume che spesso supera la reale capacità delle aziende di gestirli ed elaborarli con efficacia nei tempi utili. Una montagna di informazioni che costituiscono un problema se non utilizzati o usati poco o male, ma che possono trasformarsi in una formidabile opportunità quando vengono sfruttati nel modo corretto, ad esempio per sondare gli "umori" dei mercati e del commercio, e quindi del trend complessivo della società. Nel secondo capitolo abbiamo quindi appreso quali sono gli strumenti più idonei per "acquisire informazione", dalle API dei social network ai web scraper, cominciando ad introdurre Hadoop: un ecosistema di prodotti adottato da vendor del calibro di Facebook, Twitter durante l'intero life cycle dei big data. Nel terzo capitolo abbiamo inoltre appreso che, data la natura non strutturata dei big data quest'ultimi mal si prestano ad essere "trattati" con gli strumenti tradizionali quali gli RDBMS introducendo la tecnologia NoSQL e confrontando di volta in volta i vantaggi di taluna implementazione con le altre, compreso il modello relazionale. Abbiamo evidenziato come strumenti tradizionali non siano obsoleti, sottolineando come i database relazionali rappresentino la scelta migliore per i sistemi OLTP, sia per quanto riguarda la "potenza nelle interrogazioni" che la loro capacità di garantire la consistenza nei dati. Gli strumenti presentati in questo survay quindi, completano il data warehouse, raramente lo sostituiscono. La maggior parte delle organizzazioni, infatti, ha progettato il proprio DW per dati strutturati e relazionali, il che rende difficile ricavare valore dalla BI con dati non strutturati e semistrutturati. Infine, nell'ultimo capitolo abbiamo presentato il framework MapReduce una potente architettura che nasconde i dettagli di parallelizzazione e

bilanciamento del carico, fault-tolerance, località dei dati, affinché, l'analisi dei big data, proibitiva fino a qualche anno fa, possa diventare una realtà per molte organizzazioni.

[1] www.businessmagazine.it

[2] BIG Data - Architettura, tecnologie e metodi per l'utilizzo di grandi moli di dati

[3] NoSQL Databases Massimo Carro - Politecnico di Milano

[4] The Google File System - Sanjay Ghemawat, Howard Gobioff and Shun-Tak Leung - Google

[5] BigTable: A Distributed Storage System for Structured Data - FAY CHANG, JEFFREY DEAN, SANJAY GHEMAWAT, WILSON C. HSIEH, DEBORAH A. WALLACH, MIKE BURROWS, TUSHAR CHANDRA, ANDREW FIKES, e ROBERT E. GRUBER - Google, Inc.

[6] No SQL Database - Francesca Tranzillo

[7] Hadoop: The Definitive Guide, 3rd Edition Storage and Analysis at Internet Scale By Tom White, Publisher: O'Reilly Media / Yahoo Press

[8] Big Data: riconoscerli, gestirli, analizzarli - Dedagroup Highlights

[9] User-Generated Content - John Krumm, Microsoft Research - Nigel Davies, Lancaster University - Chandra Narayanaswami, IBM T.J Watson Research Center

[10] Rapporto di Ricerca per IBM - Big Data: nuove fonti di conoscenza aziendale e nuovi modelli di management - Paolo Pasini, Angela Perego

[11] Dati Non Strutturati: Information Retrieval - Alessandro Logheu

[12] <http://www.cs.unibo.it/~montesi/CBD/01IntroModelli.pdf>

[13] [http://it.wikipedia.org/wiki/Extract,\\_transform,\\_load](http://it.wikipedia.org/wiki/Extract,_transform,_load)

- [14] Estrazione, sentimenti analysis e rappresentazione di grandi quantità di messaggi pubblici tramite le tecnologie Big Data - Jacopo Farina
- [15] <http://searchdatamanagement.techtarget.com/definition/Extract-Load-Transform-ELT>
- [16] <http://www.tervela.com/hadoop--etl--and-elt>
- [17] [http://www.corriere.it/tecnologia/social/14\\_gennaio\\_28/social-network-big-data-sentiment-analysis-spiegata-profani-cce184b0-884a-11e3-bbc9-00f424b3d399.shtml](http://www.corriere.it/tecnologia/social/14_gennaio_28/social-network-big-data-sentiment-analysis-spiegata-profani-cce184b0-884a-11e3-bbc9-00f424b3d399.shtml)
- [18] Building a Data Warehouse for Twitter Stream Exploration Nafees Ur Rehman\*, Svetlana Mansmann, Andreas Weiler, Marc H. Scholl University of Konstanz, Germany
- [19] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?"
- [20] [http://it.wikipedia.org/wiki/Web\\_scraping](http://it.wikipedia.org/wiki/Web_scraping)
- [21] <http://curl.haxx.se/>
- [22] <http://tika.apache.org/>
- [23] Ontology-based Classification of Unstructured Information - Stefan Burger  
Eastern Michigan University Department of Computer Science Ypsilanti, Michigan
- [24] <http://it.wikipedia.org/wiki/Metadato>
- [25] <http://dublincore.org/>

[26] An Informatics Architecture for the Virtual Pediatric Intensive Care Unit - Daniel J. Crichton, Chris A. Mattmann, Andrew F. Hart, David Kale, Robinder G. Khemani, Patrick Ross, Sarah Rubin, Paul Veeravatanayothin, Amy Braverman, Cameron Goodale,

[28] <http://sqoop.apache.org/>

[29] <http://hadooptutorial.info/sqoop-importing-mysql-data-into-hdfs/>

[30] Apache Sqoop Cookbook, Kathleen Ting, Jarek Jarcec Cecho, "O'Reilly Media, Inc.

[31] <http://www.mammothdb.com>

[32] The Hadoop Distributed File System, Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler, Yahoo!

[33] HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads, Azza Abouzeid, Kamil Bajda-Pawlikowski, Daniel Abadi<sup>1</sup>, Avi Silberschatz, Alexander Rasin<sup>2</sup>, Yale University, <sup>2</sup>Brown University

[34] HDFS Architecture Guide, Dhruba Borthakur

[35] Will NoSQL Databases Live Up to Their Promise?, Leavitt, N., Leavitt Communications

[36] Database relazionali e NoSQL a confronto, Dimitri De Franciscis

[37] Big Data: nuove fonti di conoscenza aziendale e nuovi modelli di management, Paolo Pasini Angela Perego

[38] WEB SEMANTICO E GIS: LA TECNOLOGIA IN AIUTO DELLE POLITICHE TRIBUTARIE DEI COMUNI

- [39] <http://www.datumbox.com/machine-learning-framework/>
- [40] <http://flume.apache.org/>
- [41] Apache Flume: Distributed Log Collection for Hadoop, Steve Hoffman
- [42] Il linguaggio SQL: transazioni, Università di Bologna
- [43] MongoDB: analisi e prototipazione su applicazioni di Social Business Intelligence, Manuel Bianchi
- [44] NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison. A B M Moniruzzaman and Syed Akhter Hossain, Department of Computer Science and Engineering, Daffodil International University
- [45] Dynamo: Amazon's Highly Available Key-value Store, Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall e Werner Vogels - [amazon.com](http://amazon.com)
- [46] [http://it.wikipedia.org/wiki/Lotus\\_Notes](http://it.wikipedia.org/wiki/Lotus_Notes)
- [47] NoSQL Evaluation, A Use Case Oriented Survey, Robin Hecht e Stefan Jablonski - Chair of Applied Computer Science IV
- [48] <http://hbase.apache.org/>
- [49] MapReduce: Simplifies Data Processing on Large Clusters - Jeffrey Dean e Sanjay Ghemawat
- [50] <http://it.wikipedia.org/wiki/MapReduce>

[51] <http://hortonworks.com/>

[52] <http://hadoop.apache.org/>

[53] <https://hive.apache.org/>

[54] <https://www.facebook.com/notes/facebook-engineering/hive-a-petabyte-scale-data-warehouse-using-hadoop/89508453919>

[55] <https://developer.yahoo.com/blogs/hadoop/pig-hive-yahoo-464.html>

[56] <http://pig.apache.org/>

[57] Parallel K-Means Clustering Based on MapReduce - Weizhong Zhao, Huifang Ma e Qing He - The Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences - Graduate University of Chinese Academy of Sciences