PANORAMICA SUI BIG DATA

Francesco Alaimo

datasciencefacile.it@gmail.com

https://www.datasciencefacile.it

Sommario

Sommario 2
Panoramica sui Big Data: concetti, architetture, applicazioni 3
Concetti 3
RFID 4
Web 2.0 4
GIS 4
Aspetti critici dei Big Data 4
Investire nel Big Data 5
Architetture 6
Tecnologie per Big Data 6
Soluzioni hardware 7
Soluzioni cloud 7
Hadoop 8
Database NoSQL 11
Applicazioni 13
MapReduce all'opera 13
Alcuni esempi reali di utilizzo dei Big Data 15
Conclusioni 17
Bibliografia 18

Panoramica sui Big Data: concetti, architetture, applicazioni

Concetti

Il crescente sviluppo di Internet, in particolare del Web 2.0, e l'avvento dell'IoT, ha generato una quantità di dati che, oramai, è arrivata a dimensioni dell'ordine dei PB[1], traguarda i ZB[2] e non accenna a diminuire. Si parla oggi, infatti, di Big Data, facendo riferimento alla massa di informazioni disponibile, che, rispetto al passato, è contraddistinta da tre fattori principali:

- Volume
- Velocità
- Varietà

Queste tre caratteristiche, hanno determinato la necessità di dover elaborare queste informazioni grazie all'utilizzo di nuovi strumenti e tecniche di elaborazione, che non appartengono più al dominio RDBMS[3], ma piuttosto ai database No-SQL[4] e ai sistemi che si basano sul framework Hadoop[5].

Col termine Big Data, si fa riferimento, in generale, a dati non strutturati (potremmo anche dire misti, strutturati/non strutturati), disponibili in enormi quantità (l'aspetto del volume accennato sopra), varietà (si immagini, ad esempio, alla tipologia di dati che può essere registrata da un sensore o a certe informazioni sulla navigazione di un utente su un sito) e a ritmi sempre più frenetici di produzione (si pensi a tutti i post che vengono scritti su Facebook, ogni ora).

Questa massa di informazioni è dirompente e nessun RDBMS potrebbe gestirla a costi sostenibili per qualsiasi azienda. È questo il motivo per cui, negli anni, le architetture e gli strumenti che sono stati sviluppati hanno avuto come leitmotiv l'utilizzo di hardware commerciale (commodity hardware) e di software Open Source.

Ma perché c'è tanto interesse? In realtà, la disponibilità di queste informazioni, se governata, pone nuove sfide, ma anche alcune opportunità:

- *Business*: aumenta il vantaggio competitivo delle aziende e permette di proporre nuovi modelli di business
- *Tecnologiche*: è necessario governare i dati, ma ciò non è possibile con i tradizionali strumenti di BA[6] e occorre introdurne di nuovi
- *Finanziarie*: implementare una infrastruttura per governare i Big Data è un impegno economico di rilievo e occorre pesarlo con un chiaro obiettivo di ROI[7], valutando se è più conveniente ricorrere ad una soluzione in house o ad una Cloud

Cominciamo con l'elencare le possibili fonti di dati e il loro ambito di applicazione.

---

[1] Petabytes, pari a 10^{15} byte
[2] Zettabytes, pari a 10^{24} byte
[3] RDBMS, Relational Database Management System, fa riferimento ai database relazionali che, prima dell'avvento dei Big Data, permettevano di estrarre conoscenza dai dati.
[4] No-SQL: strumenti di elaborazione di dati non strutturati che si contrappongono ai RDBMS
[5] Ne parleremo diffusamente nel seguito
[6] Business Analysis
[7] ROI, Return Of Investiment

RFID[8]

Si tratta di un dispositivo che attraverso un tag (etichetta elettronica) permette di identificare qualsiasi elemento che ne sia fornito (da un capo di vestiario ad una persona), identificabile attraverso lettori fissi o portatili a radiofrequenza. Nonostante il nome esso può essere letto e in alcune versione, può anche trasmettere dati. L'impiego è estremamente vario, si passa dalla gestione dell'inventario di un magazzino alla rilevazione del superamento di alcuni parametri di qualità (es. esposizione a temperature elevate per i prodotti deperibili). Si veda ad esempio il case-study di Walmart, leader nella grande distribuzione, sull'utilizzo degli RFID al link http://www.slideshare.net/AchchuthanSeetharan/wal-mart-45668917. Il numero di RFID mondiale ha superato il miliardo e non accenna a diminuire. È un ottimo esempio di sorgente di dati per il mondo Big Data.

Web 2.0

Rappresenta da solo il bacino di analisi più interessante e più variegato, tra quelli disponibili: miliardi e miliardi di documenti, post, pagine html, tweet e altro. Si tratta per lo più di dati non strutturati che raccolgono informazioni sui trend e il sentiment delle persone riguardo ad argomenti di discussione e, molto più interessante per le aziende, prodotti e brand. L'elaborazione di queste informazioni permette di attuare delle politiche di fidelizzazione verso i clienti o iniziative di *cross-selling* (si pensi ai suggerimenti di acquisto di Amazon), oppure di acquisire nuova clientela attraverso l'analisi di comportamenti di acquisto[9]. I precursori di questo universo sono stati Google, Apple (Siri permette l'interazione vocale con una persona utilizzando i Big Data) e Facebook. Solo quest'ultima, nel 2012 poteva contare su una 'base di dati' di circa 100 PB. Per queste aziende, l'elaborazione di grossi volumi di dati è stata possibile attraverso l'adozione di architetture sviluppate ad hoc (Google ha sviluppato in proprio una soluzione da cui è derivato poi il framework Hadoop, poi utilizzato da Facebook).

Il caso Barack Obama[10]

Il primo esempio di utilizzo dei Big Data è stato la conduzione della campagna politica di Barack Obama, nel 2012. Lo staff del presidente aveva contribuito ad alimentare un'unica base di dati che raccoglieva i risultati dei sondaggi, le anagrafiche dei sostenitori Democratici, le attività sul campo, le attività sui social, allo scopo di scovare potenziali votanti per poterli convincere a concedere il proprio voto in favore di Obama. Per mezzo di simulazioni predittive, cercavano di capire quali modifiche portare alla campagna politica del candidato per aumentare la probabilità di vincita su ciascuno Stato. Si può dire che, senza dubbio, la vittoria di Obama sarebbe stata difficile senza l'ausilio dei Big Data (presenza di dati + ricerca degli insight).

GIS[11]

I dati di localizzazione sono oramai disponibili su moltissimi dispositivi e fanno parte integrante dell'universo Big Data. Permettono di mostrare nuovi pattern di analisi basati sulla dislocazione delle sorgenti in un dato territorio o di mostrare l'origine geografica di alcuni messaggi provenienti dai social network. L'ambito in cui finora hanno trovato maggiori applicazioni è in caso di disastri naturali o eventi meteorologici avversi, ma questo tipo di dati viene adoperato anche dalle aziende che forniscono servizi (es. acqua o corrente elettrica), o dagli istituti che forniscono le carte di credito, per la verifica delle transazioni dei clienti e, in generale, nei sistemi di fraud detection.

Aspetti critici dei Big Data

I Big Data offrono grandi potenzialità, ma anche qualche rischio. Si parla, infatti, di *Qualità dei dati*, intendendo con questo termine, la presenza delle seguenti caratteristiche a supporto:

- *Completezza*: presenza nei dati di tutte i campi necessari ad un utilizzo efficace degli stessi
- *Consistenza*: i dati devono essere consistenti, cioè non devono esserci delle discontinuità tra un periodo e quello immediatamente successivo se non è giustificato da una operazione di carico/scarico

[8] Radio Frequency Identification, altre info al link: https://it.wikipedia.org/wiki/Radio-frequency_identification
[9] Detto anche 'funnel customer journey', http://onlinelibrary.wiley.com/doi/10.1002/bltj.21642/full
[10] http://www.infoworld.com/article/2613587/big-data/the-real-story-of-how-big-data-analytics-helped-obama-win.html
[11] Geographic Information System

- *Accuratezza*: i dati devono essere coerenti col modello di business
- *Mancanza di duplicazioni*: in un array di dati non devono esistere dati duplicati
- *Integrità*: questo termine, che deriva dai database relazionali, indica che i dati devono rispettare alcuni vincoli (un campo deve essere univoco, oppure rientrare in un certo range, o essere di un dato tipo)

In sostanza i dati non sono esenti da errori e se la loro qualità non è buona, non lo sono nemmeno le valutazioni e/o le predizioni che con essi intendiamo svolgere. Alcune volte, per la loro natura, essi possono essere particolarmente imprecisi, ad esempio quelli che provengono dai social o quelli caricati su un sistema da un operatore, perché essendo per lo più composti da testo sono soggetti ad errori o essere pieni di abbreviazioni, che possono cambiare il significato, a seconda del contesto. Altri aspetti rilevanti sono la *veridicità* delle informazioni, che non sempre può essere confermata, la presenza di informazioni che sembrano essere fuori dal contesto, gli *outlier*, ma che in alcuni ambiti (es. la frode) sono le più importanti e che è opportuno mantenere in fase di normalizzazione dei dati, per non parlare dei problemi legati alla *privacy*, in particolare a l'utilizzo che si può fare dei dati. Premesso che, per definizione, chi pubblica informazioni sul web, anche di carattere personale, deve essere consapevole, per le caratteristiche del media, che queste sono di dominio pubblico, è anche vero che tramite elaborazioni si può risalire ad informazioni sensibili sull'individuo, relative a suoi orientamenti religiosi o sessuali o semplicemente legati alla sua posizione geografica, per non parlare poi delle informazioni mantenute dagli ospedali e per le quali dovrebbe essere tenuto uno specifico livello di protezione.

Investire nel Big Data

Perché le aziende dovrebbero investire nei Big Data? Per spiegarlo si può fare riferimento ad un indice finanziario, il ROI, modificandolo allo scopo e così definito:

$$ROI_{BD} = \frac{minori\ c osti + maggiori\ r ricavi}{investimento\ i niziale}$$

La formula ha questo significato: l'adozione dei Big Data nei processi produttivi, può permettere di prevedere eventuali malfunzioni nella catena di produzione, quindi limitando i fermi e i costi di gestione delle anomalie *(minori costi),* mentre l'analisi delle opinioni dei consumatori permette di realizzare campagne mirate di acquisizione o di retention, aumentando i ricavi *(maggiori ricavi)*. Per valutare invece i *costi ricorrenti* e *l'investimento iniziale* è necessario ricorrere al concetto di TCO[12], che è un indicatore che bene esprime i costi del ciclo di vita dell'hardware e del software, all'interno di un'azienda. Il TCO tiene conto dei costi di acquisto, installazione, gestione, manutenzione e alienazione, comprendendo tra le voci il costo dell'hardware, le licenze software, i costi del personale (incluse di training e consulenze), i costi di gestione e manutenzione.

Rispetto al passato questi costi si sono molto ridotti grazie alla possibilità di utilizzare hardware commerciale[13] e software open source[14], ma non ultimo, grazie all'adozione di soluzioni di Cloud computing[15], che permettono di implementare il modello di utilizzo *pay-per-user*, cioè le aziende pagano per quello che utilizzano in termini di spazio su hard disk, tempo di utilizzo e potenza di calcolo. Le soluzioni Cloud hanno anche il vantaggio di spostare gli oneri di manutenzione e aggiornamento al fornitore del servizio.

Sulla base di questi 'calcoli' le aziende possono decidere se l'utilizzo dei Big Data può portare dei vantaggi al business e a quali costi.

Architetture

Tecnologie per Big Data

Il ciclo di vita dei Big Data prevede le seguenti fasi:

[12] Total Cost of Ownership, sviluppato da Gartner nel 1997
[13] Commodity hardware, si veda http://whatis.techtarget.com/definition/commodity-hardware
[14] Si veda https://www.talend.com/blog/2016/03/03/big-data-why-you-must-consider-open-source/
[15] Si veda https://pdfs.semanticscholar.org/8d8e/9fea7bf3a9367f5e555fdb954d9f85a50358.pdf

- *Acquisizione*
- *Immagazzinamento*
- *Organizzazione*
- *Integrazione* e arricchimento
- *Analisi*

A ciascuna fase è possibile associare una specifica tecnologia ed in particolare, molte appartengono al cosiddetto ecosistema Hadoop, che è un punto di riferimento nel mondo dei Big Data, utilizzato da moltissime aziende nel mondo.[16]

Acquisizione

L'acquisizione dei dati può avvenire attraverso diversi metodi, in funzione del tipo di dato che si vuole acquisire:

- *API*[17]: permettono, ad esempio, di importare le informazioni pubblicate in una piattaforma di social (facebook, twitter, etc.) o l'interfacciamento con un motore di ricerca
- *Import dei dati attraverso ETL*[18]: Hadoop, ad esempio prevede l'uso dello strumento Sqoop
- *Web scraping*[19]: è un processo di acquisizione automatica dei dati da un sito web; Hadoop prevede ad esempio, l'utilizzo di Apache Tika
- *Lettura di uno stream*: adatto, in particolare, per le sorgenti continue di dati che richiedono un trasferimento costante degli stessi per il loro processamento. Hadoop prevede l'utilizzo di Apache Flume, ma esistono altri strumenti allo scopo, come quello offerto da Microsoft denominato Stream Insight, che è a tutti gli effetti una piattaforma di CEP[20]

Immagazzinamento e organizzazione

È questo un ambito in cui differiscono i Big Data e le tecnologie di BI[21] tradizionali, determinato in buona misura dalla mole e dalla struttura dei dati. Fortunatamente negli ultimi anni si è sviluppato a diffuso Hadoop, un framework open source, affidabile e scalabile, che utilizza commodity hardware, denominati nodi, collegati in rete. Il suo funzionamento si basa sul concetto di '*Divide et impera*', cioè di dividere il problema in problemi più piccoli, da risolvere separatamente per poi risalire, dai singoli risultati, al risultato finale. In questo modo si può risolvere un problema anche complesso, che operi su una grande quantità di dati, non risolvibile da un unico elaboratore, distribuendolo su più elaboratori. Hadoop prevede i seguenti componenti:

- *Hadoop common*: è uno strato software che fornisce servizi agli altri moduli
- *HDFS*: è il file system distribuito di Hadoop
- *YARN*: è il sistema di scheduling e gestione delle risorse
- *MapReduce*: è il cuore del framework, riesce a processare, in parallelo, enormi quantità di dati

Hadoop è a tutti gli effetti un sistema di calcolo distribuito, che espone a tutti gli elaboratori un file system distribuito, HDFS[22], che attraverso un meccanismo di replica dei dati, sui vari nodi, li protegge da eventuali fault. Inoltre, la funzionalità *MapReduce* è quella che si occupa di elaborare i dati, già ridotti in singole porzioni, e di restituire e aggregare i risultati. La coppia, HDFS e MapReduce, non è assimilabile ad un database, almeno in senso tradizionale, ma è gestibile come se lo fosse per mezzo di alcuni strumenti dell'ecosistema Hadoop, tra i quali Apache HBase, che riesce ad effettuare interrogazioni su miliardi di righe

[16] Si veda https://wiki.apache.org/hadoop/PoweredBy
[17] Application Programming Interface
[18] È l'acronimo per Extract, Transform, Load, vedi https://it.wikipedia.org/wiki/Extract,_transform,_load
[19] Si veda https://it.wikipedia.org/wiki/Web_scraping
[20] Complex Event Processing
[21] Business Intelligence, si veda https://it.wikipedia.org/wiki/Business_intelligence
[22] Hadoop Distribuited File System

e milioni di colonne di dati anche non strutturati. Ad esso si affiancano altri database NoSQL[23] come Cassandra, MongoDB e Neo4J.

Integrazione

È una fase che opera sui dati caricati e organizzati per mezzo di Hadoop/NoSQL, preparatoria all'attività di analisi. È utile, in particolare, nei casi in cui le informazioni siano inserite in file di formati differenti, come PDF, Word, Excel, HTML, XML etc. ed è quindi necessario recuperarle e renderne uniforme il trattamento. Esistono allo scopo, diversi strumenti nell'ecosistema Hadoop, come Apache Tika e HiveQL.

Analisi

La fase di analisi utilizza strumenti dell'ecosistema Hadoop o esterni ad esso: questi sono Pig, R e Mahout. Pig[24], in particolare, permette di scrivere delle sequenze di operazioni che realizzano le fasi di lavorazione di MapReduce, superando la complessità di scriverle direttamente utilizzando funzioni di quest'ultimo. Con lo stesso scopo, ma più semplice nell'utilizzo è Hive[25], che addirittura permette di utilizzare un linguaggio simile a SQL (HiveQL). Infine, ad un livello di complessità più elevato, si pone R[26], che è un software per il calcolo statistico che, attraverso l'utilizzo di un plug-in, RHadoop, permette di utilizzare le funzioni MapReduce di Hadoop, potenziandone le caratteristiche di calcolo. Mahout[27], invece, permette di svolgere sui dati, tramite script e librerie, tecniche di Machine learning, Supervised e Unsupervised learning, per estrarre gli insight.

Soluzioni hardware

Le tecnologie hardware per Big Data si dividono in due soluzioni: *SMP*[28] e *MPP*[29]. La prima soluzione si riferisce a sistemi in cui diversi processori (CPU) condividono lo stesso Sistema Operativo (SO), la RAM e il bus di IO e sono anche detti *shared everything.* Presentano alcuni limiti al crescere della quantità di dati perché presentano un collo di bottiglia proprio nell'utilizzo di un unico bus di accesso alla RAM e ai dischi rigidi. Al contrario, nei sistemi MPP, ogni CPU accede con un BUS dedicato alla RAM e al disco rigido (*shared nothing*), ma riesce a dialogare con altre unità di elaborazione per mezzo di un apposito layer di comunicazione. Questi ultimi si prestano bene a gestire grandi quantità di dati.

Soluzioni cloud

Un'alternativa a quella di creare un'infrastruttura Big Data nella propria azienda, è quella di adoperare alcune soluzioni proposte da Google, Amazon, Microsoft, Cloudera, MapR giusto per citarne alcune. Come accennato in precedenza, i vantaggi di questa soluzione sono principalmente:

- Non è necessario definire alcun cluster di Hadoop
- Le operazioni di manutenzione hardware e software sono a carico del fornitore
- I costi iniziali di questa soluzione sono meno elevati di quelli che si avrebbero se si perseguisse una soluzione in house
- Modello di fatturazione basato su *pay-per-use*

Di contro, se i dati da elaborare si trovano a casa propria bisogna tenere conto dei tempi necessari per il loro caricamento nel Cloud, motivo per cui alcuni fornitori prevedono un servizio per spostare 'fisicamente' i dati presso i loro siti tramite la spedizione di hard disk.

Hadoop

Poiché Hadoop è un framework largamente adoperato in ambito Big Data, ne tratteremo diffusamente in questo paragrafo. Hadoop è una piattaforma software open source, affidabile e scalabile per eseguire calcolo distribuito, per mezzo di commodity hardware. Esso nasce all'interno di un progetto per la costruzione di un

[23] NoSQL è un acronimo che indica i database non relazionali, che operano sui Big Data e, in particolare, su dati non sempre strutturati
[24] Si veda https://pig.apache.org/
[25] Si veda https://hive.apache.org/
[26] Si veda https://cran.r-project.org/
[27] Si veda https://www.tutorialspoint.com/mahout/mahout_machine_learning.htm
[28] Symmetric MultiProcessing
[29] Massive Parallel Processing

motore di ricerca basato su Java, denominato Nutch e sviluppato da Doug Cutting e Mike Caffarella. Hadoop tenta di sopperire ad un problema di scalabilità di Nutch. Il suo sviluppo segue nel 2004, la decisione di Google di rendere pubblici i documenti tecnici che riguardavano Google File System[30] e Google Mapreduce[31], da cui Hadoop ha chiaramente attinto. Nel 2008, Hadoop viene rilasciato, nella sua prima release, come progetto indipendente di Apache. Oggi fa parte dei progetti di calcolo distribuito ospitati dalla Apache Software Foundation. Prima dell'affermazione di Hadoop, le elaborazioni su grandi quantitativi di dati venivano svolte utilizzando delle architetture *HPC* (*High Performance Computing*). Questa architettura permetteva di suddividere il lavoro su molti computer in cluster che accedevano ad un insieme di dischi condiviso, che si scambiavano informazioni attraverso un complesso protocollo di comunicazione di basso livello, denominato *MPI* (*Message Passing Interface*), che aveva il punto debole proprio nel fatto che se l'attività di calcolo richiedeva numerosi accessi ai dischi, la componente di storage poteva saturarsi creando delle fastidiose code di accesso. Hadoop, supera queste limitazioni, grazie all'utilizzo di HDFS e MapReduce, utilizzando delle librerie di alto livello, più semplici da implementare, e distribuendo i dati fra i vari nodi, rendendoli disponibili all'elaborazione, riducendo al minimo gli accessi alla rete. Hadoop, inoltre, ha un'elevata affidabilità, potendo gestire i fault a livello applicativo, senza necessità di introdurre dei meccanismi hardware, e scalabilità, potendo aumentare la capacità di elaborazione semplicemente aggiungendo dei nodi al cluster. Quest'ultimo punto permette dei costi molto più bassi di quelli che erano richiesti per aumentare le performance di un'architettura HPC.

C'è da chiarire un punto. Hadoop nasce per gestire grosse moli di dati mentre un RDBMS opera su una serie di record più ridotta. Se la nostra necessità non è quella di operare sui Big Data, allora la soluzione più efficiente rimane l'utilizzo di un RDBMS.

Come già detto, Hadoop è costituito dai seguenti componenti:

- Hadoop common
- HDFS
- YARN
- MapReduce

Nel seguito le analizzeremo una per una.

HDFS

È un file system distribuito che può adoperare commodity hardware. Può gestire un numero molto elevato di file di dimensione variabile dal GB al TB, distribuiti su un cluster che può essere composto da migliaia di nodi. Può identificare automaticamente il fault di uno dei nodi e attivare le operazioni di recovery. L'architettura del cluster è composta dalle seguenti tipologie di nodi:

- *NameNode*: è il nodo principale del cluster, che gestisce la struttura del file system e il *namespace*, cioè l'elenco dei nomi dei file e dei blocchi, porzioni della dimensione di 64 o 128MB che compongono i file. Controlla tutti gli accessi ai file, in lettura e scrittura, gestisce il meccanismo delle repliche e in caso di fault di uno o più nodi, si occupa di riallocare i blocchi. Il NameNode memorizza il namespace in due file: il primo è l'*fsimage* e il secondo è un log dei cambiamenti che prende il nome di *journal*. All'avvio dei NameNode, questo unisce il contenuto di *fsimage* e *journal* e lo salva in uno *snapshot*
- *DataNode*: è il nome delle applicazioni che girano sui nodi del cluster, che realizzano fisicamente le operazioni di lettura/scrittura richieste dai client, incluse le repliche richieste dal NameNode
- *SecondaryNameNode*: a dispetto del nome, non si tratta di un backup del NameNode, anche se svolge funzioni di ausilio nel mantenimento di copie di *fsimage* e *journal,* che restituisce al NameNode in forma di *snapshot*. Nelle ultime versioni di Hadoop è stato rinominato CheckPointNode.

[30] Si veda https://static.googleusercontent.com/media/research.google.com/it//archive/gfs-sosp2003.pdf
[31] Si veda https://static.googleusercontent.com/media/research.google.com/it//archive/mapreduce-osdi04.pdf

- *BackupNode*: poiché il NameNode è un Single Point of Failure, nelle ultime versioni è stato introdotto un nodo di backup del NameNode, simile al CheckPointNode, che rimane sempre sincronizzato col NameNode e permette di il failover automatico in caso di fault

La dimensione dei blocchi e il numero di repliche di ogni file può essere configurato singolarmente. L'HDFS recupera i dati sulla base della loro prossimità, quindi nella realizzazione delle repliche viene utilizzata una tecnica che tiene conto della posizione dei blocchi all'interno dei rack, prediligendo quelli che sono più vicini, col vantaggio di avere sempre i dati disponibili nel caso di fault di un nodo o di un intero rack.

HDFS è molto efficiente se lavora con file di grandi dimensioni, mentre non lo è al contrario. Nel caso fossero presenti molti file piccoli, HDFS li compatta in blocchi più grandi per mantenere efficiente l'utilizzo del namespace. Naturalmente, come tutti i file system, è dotato di meccanismo di sicurezza di accesso ai file, secondo il modello POSIX (*Portable Operating System Interface*), che permette di associare ai file permessi di lettura, scrittura, esecuzione (rwx) e gruppi di appartenenza definiti come *owner*, *group* e *superuser*. È possibile, tramite comando, cambiare i diritti di un file e l'appartenenza ad un gruppo.

MapReduce

Mentre HDFS si occupa di gestire i file e il loro accesso, l'elaborazione è deputata alla componente MapReduce. Si tratta di un framework che permette di creare delle applicazioni in grado di elaborare grandi quantità di dati in parallelo, che utilizza un approccio denominato *functional programming*. A differenza della programmazione basata su *multithreading*, dove diversi processi, utilizzando complessi meccanismi di coordinamento, condividono la stessa risorsa, in questo caso la condivisione è eliminata, passando porzioni di dati alle funzioni distribuite nei vari nodi, come parametri o valori di ritorno. Come detto la soluzione segue il principio 'divide et impera', cioè un'operazione di calcolo per quanto complessa viene suddivisa in un certo numero di parti che vengono processate autonomamente; successivamente i singoli risultati saranno 'ridotti', cioè ricomposti, nel risultato finale. MapReduce è composto da quattro elementi:

- Dati di ingresso su HDFS
- Una funzione di *map* dei dati, che trasforma i dati di input in una serie ordinata di coppie chiave/valore
- Una funzione di *reduce* che raggruppa le coppie con la stessa chiave per applicare una funzione di calcolo ai valori (media, conteggio, somma, etc.), restituendo ancora delle coppie chiave/valore
- Dati di uscita su HDFS

Se i problemi sono particolarmente complessi è possibile che vengano eseguiti sui dati più fasi di MapReduce.

Entrando più nel dettaglio del suo funzionamento, MapReduce utilizza due componenti funzionali:

- Il *JobTracker*: si occupa di presiedere alla gestione delle risorse di elaborazione (CPU e memoria) e controlla il ciclo di vita di un job MapReduce. Ogni job viene assegnato ai nodi che possiedono i dati da elaborare o a quelli, nello stesso rack, che ne possiedono una replica (se possibile). Se un task di un job fallisce, il JobTracker si occupa di riassegnarlo ad un altro nodo, con lo stesso criterio descritto prima.
- Il *TaskTracker*: indica i processi che sui nodi eseguono i task sotto la direzione del JobTracker

È il JobTracker che determina il numero di porzioni in cui devono essere suddivisi i dati di input e attiva i TaskTracker sui nodi che sono più vicini alle porzioni di dati da elaborare. I TaskTracker avviano la funzione di *map* sui dati e, appena terminato, avvisano il JobTracker che può attivare la fase di *reduce*, il cui output viene salvato in un file diverso per ciascun TaskTracker.

Hadoop streaming

Accenneremo a questa utility, inclusa in Hadoop, perché permette ad un qualsiasi linguaggio di interfacciarsi con le funzioni di *map* e *reduce* utilizzando lo standard input e lo standard output del sistema. I dati in ingresso dell'utility sono riconosciuti come coppia chiave/valore sulla base della presenza del separatore *Tab*; in tal caso, la porzione della riga di input a sinistra del *Tab* sarà considerata chiave, mentre quella a destra sarà il

valore. Se il *Tab* non è presente, tutta la riga sarà considerata come chiave e il valore sarà nullo. Questo comportamento può essere variato agendo su alcuni parametri dell'utility.

YARN[32]

Le versioni più recenti di Hadoop, introducono questo nuovo framework che si prende carico della gestione delle risorse, quali la CPU e la memoria e che monitora l'esecuzione delle applicazioni. Rispetto al MapReduce detto di prima versione (MRv1), costituito dal JobTracker e dal TaskTracker, questo nuovo MapReduce di seconda versione (MRv2), è un'applicazione che viene monitorata da YARN, costituita da un *ResourceManager* (RM), che si occupa dell'assegnazione delle risorse alle applicazioni e da uno o più *ApplicationMaster* (AM), che si occupa di gestire la singola applicazione, il suo avvio e la ripartenza in caso di errore. Ciascun nodo fisico costituisce un *NodeManager* (NM), su cui girano i processi. In questo caso è possibile fare girare, sotto la supervisione del *RM*, più applicazioni, ognuna collegata ad un *AM*, consistenti in un certo numero di task, detti *Container*, che sono stati richiesti dal *RM* (Figura 1).

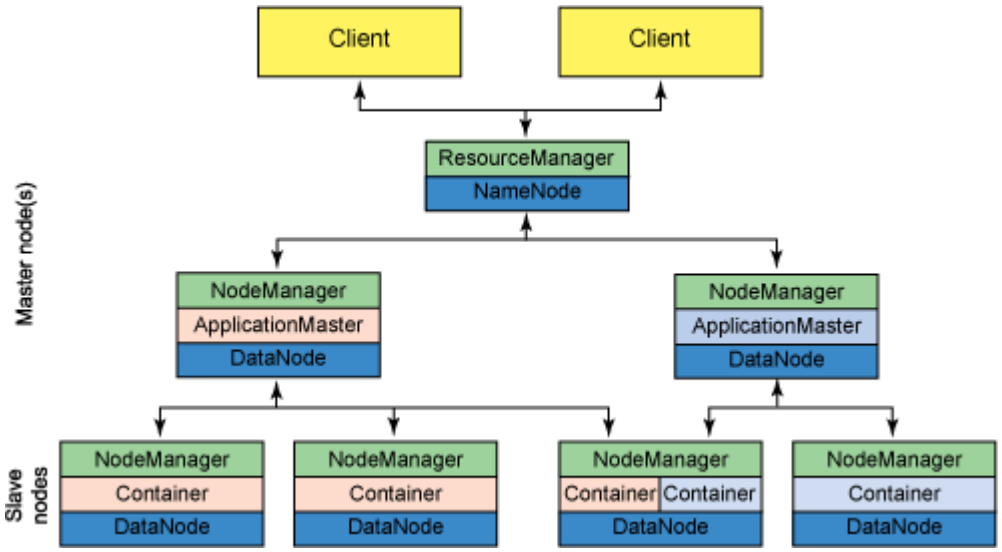

Figura 1 YARN (MRv2), fonte https://www.ibm.com/developerworks/library/bd-hadoopyarn/figure2.png

È importante notare che YARN può supportare altri framework, oltre a MapReduce, e che i programmi creati per il framework MRv1 sono compatibile con MRv2, perché le interfacce non sono state variate.

Altri strumenti dell'ecosistema

HCatalog

Hadoop lavora prevalentemente su dati non strutturati, che possono essere ricondotti ad uno schema nel momento i cui questi vengono letti. Questo schema non è rigido, ma è costituito da metadati associati ai dati, sui quali si possono eseguire delle interrogazioni utilizzando un linguaggio simile a SQL, come quello offerto da Hive. HCatalog è uno strumento che semplifica la creazione e la modifica di questi metadati, permettendo di visualizzare in formato tabellare dei dati registrati in HDFS, indipendentemente dal formato di origine. Si interpone tra i dati su HDFS e gli strumenti con cui è possibile interagire con gli stessi, quali ad esempio, MapReduce, Pig, Hive. HCatalog presenta i dati come se questi appartenessero ad un database relazionale, distribuiti in tabelle e organizzati in database.

Oozie

È un motore di workflow specializzato nell'esecuzione di job MapReduce, Pig Hive ed altre operazioni HDFS. Le azioni sono organizzate in un grafo di tipo *Direct Acyclic Graph*, che impone che un'azione non può eseguita se la precedente non è terminata. È il sistema controllato che notifica ad Oozie il termine di un'azione per fare in modo che le azioni del workflow possa essere eseguite in successione. Il workflow viene scritto utilizzando il linguaggio hPDL (*Process Definition Language*).

[32] È un acronimo che sta per Yet Another Resource Negotiator

Ambari

È uno strumento per la gestione e il monitoraggio di un cluster Hadoop tramite un'interfaccia web. Al momento supporta i seguenti componenti di Hadoop:

- HDFS
- MapReduce
- Hive
- HCatalog
- HBase
- Zookeeper
- Oozie
- Sqoop

Per ciascun componente, Ambari consente di utilizzare un wizard per l'installazione, gestire il cluster fornendo gli strumenti per arrestare e far ripartire i servizi su ogni nodo, effettuare il monitoraggio, per mezzo una dashbord, segnalando tramite email eventuali situazioni critiche.

Database NoSQL

Con questo termine si indicano quei database, che si discostano dai RDBMS, ma con in comune il fatto di operare su grandi quantità di dati, di avere un'alta scalabilità, di poter operare su strutture dati non fisse, anzi, con elevata variabilità. Il sito http://nosql-databases.org fornisce una definizione di questo tipo di database: "Next Generation Databases mostly addressing some of the points: being non-relational, distributed, open-source and horizontally scalable.", oltre a riportare una lista di circa 150 diverse implementazioni. In sintesi, i database NoSQL sono:

- Database distribuiti e, generalmente, open source
- Sono scalabili orizzontalmente aggiungendo nuovi nodi
- Non hanno uno schema
- I dati sono facilmente replicabili
- Utilizzano della API semplici
- Non supportano la proprietà ACID[33] delle transazioni
- Gestiscono grandi quantità di dati (appunto, Big Data)

Di quelle menzionate, la principale caratteristica che appartiene ai database relazionali e non presente in quelli NoSQL è proprio la proprietà ACID. Si ammette, cioè che i dati non siano consistenti, ad un dato momento. Si parla, in questo caso, di proprietà BASE (*Basically Available, Soft State, Eventual consistency*), ad indicare che un'applicazione deve funzionare sempre (*Basically Available*), tuttavia non deve garantire la consistenza ad ogni istante (*Soft State*), ma il dato può diventare consistente se cessano le attività di modifica dello stesso (Eventual consistency). Il modello BASE prevede tre modalità di funzionamento:

- *Casual consistency*: ad ogni modifica l'applicazione invia un segnale alle altre sessioni, che da quel momento vedranno il dato aggiornato
- *Read your own writes*: la sessione che modifica i dati vedrà subito i cambiamenti, mentre le altre sessioni potranno vederli dopo un certo ritardo

[33] Acronimo che sta per Atomicity, Consistency, Isolation, e Durability (Atomicità, Coerenza, Isolamento e Durabilità)

- *Monotonic consistency*: una sessione non potrà mai vedere dati con versioni antecedenti a quella letta

Uno dei sistemi per assicurare la consistenza è il *vector clock*. Questo funziona a partire da un contatore dei cambiamenti (*change number*), presente su ogni nodo, che viene caricato in un vettore, insieme ai change number degli altri nodi. Tale vettore assieme agli aggiornamenti, viene scambiato tra i nodi in occasione di ogni modifica, ed ogni nodo che lo riceve esamina il change number per verificare che l'aggiornamento sia in sequenza. L'aggiornamento non viene applicato subito, ma il nodo attende, nel caso, che arrivino gli altri aggiornamenti nell'ordine corretto.

Rispetto ad RDBMS, un database NoSQL può scalare con la semplice aggiunta di un nuovo nodo, utilizzando commodity hardware. La diffusione dei database NoSQL è andata di pari passo con quella di Hadoop. Alcune implementazioni, in particolare, lo supportano e traggono vantaggio dalle sue caratteristiche, altre, come HBase, sono completamente integrate con Hadoop.

I database NoSQL si distinguono in quattro categorie:

- *Key/Value database*: sono basati sul concetto di *associative array*, cioè una struttura che contiene delle coppie chiave/valore. Per definizione, la chiave è un valore univoco con il quale è possibile identificare e ricerca i valori nel database. Gli *associative array* sono spesso implementati tramite una *hash table*, supportano le operazioni di Add, Remove, Modify e Find degli elementi. Non tutte le operazioni disponibili su un RDBMS sono possibili su un database Key/Value (ad esempio, join o filtri complessi). Si dimostrano molto flessibili perché riescono a fare fronte all'aumento dei dati con l'aggiunta di nodi al cluster. Tra i database di tipo Key/Value troviamo Berkley DB, Project Voldemort

- *Document-oriented database:* sono strumenti per la gestione di dati semi-strutturati, in cui delle coppie chiave/valore sono organizzate utilizzando il formato JSON o XML (i documenti). Sono quindi simili al tipo di database precedente, ma presentano una struttura che comunque non pone limiti allo schema dei documenti. L'utilizzo di questo tipo di database è conveniente nelle situazioni in cui i dati hanno una struttura dinamica o presentano un gran numero di campi opzionali. Tra i più conosciuti, citiamo MongoDB, utilizzato da Amazon, e CouchDB

- *Column-oriented database:* è un modello che organizza i dati per colonne, invece che per righe. Hanno il vantaggio di poter comprimere i dati, quando un campo di una colonna non esiste. In ambito NoSQL, i database presentano una struttura mista, righe/colonne, simile a quella di Google Big Data, che è il riferimento per la realizzazione di una serie di strumenti NoSQL. In generale il modello di archiviazione prevede la presenza di coppie chiave/valore (o colonna/valore), in cui è presente una chiave univoca, che prende il nome di *chiave primaria*, cui afferiscono le unità dati. ciascuna riga può contenere una o più colonne (anche tutte). Riprendendo Google Big Data, che è il precursore di questo modello, esso è utilizzato in molte delle applicazioni di questa azienda: indicizzazione web, Google Earth, Google analytics e così via. Big Table è descritto da Google, come "a sparse, distribuited, persistent multidimensional sorted map", nel quale i valori sono salvati come array di byte e sono identificati da un indirizzo composto da una chiave di riga (*row key*), chiave di colonna (*column key*) e *timestamp*. Big Table utilizza il file system distribuito GFS (*Google File System*). Il timestamp serve a differenziare diverse versioni della stessa cella (intesa come dato caricato all'incrocio delle coordinate riga/colonna). Le colonne sono organizzate in *column family*, che devono essere predefinite prima della creazione delle colonne sottostanti. Oltre a Google Big Data, troviamo HBase, che è altamente integrato con Hadoop e HDFS e ha diversi punti in comune con Big Table, da cui è derivato, e Cassandra, utilizzato da Twitter ed eBay. Cassandra è un database *fault tolerant (*proprietà *Partition tolerance)*, *eventually consistent* (proprietà *Consistency*) che risponde a due dei requisiti del teorema CAP[34], su cui si basano i database NoSQL:

    - *Consistency: tutti i nodi del sistema vedono gli stessi dati allo stesso istante*
    - *Availability: ogni richiesta ha sempre una risposta sull'esito dell'operazione*

[34] Consistency, Avalability, Partition tolerance

- *Partition tolerance: il sistema rimane operativo in caso di perdita di messaggi tra i nodi o la perdita di un nodo stesso*

- *Graph database:* è un tipo di database che si basa sul concetto di grafo, che consiste in una serie di nodi (vertici o vertex), collegati tra loro da archi (lati o edge). I grafi sono utilizzati in numerosi campi della matematica. In informatica, un grafo è una struttura dati costituita da un insieme finito di coppie ordinate (gli archi) e di oggetti (i nodi) che possono far parte della struttura del grafo o fare riferimento a oggetti esterni. Questa struttura può associare agli archi un valore. I grafi si prestano molto bene a rappresentare dati semi-strutturati e con molte interconnessioni, come le pagine Web, i legami nei social network, le reti di computer e altri scenari. I *Graph database* nascono allo scopo di gestire questo tipo di rappresentazioni, permettendo in maniera agevole di calcolare il percorso più breve tra due nodi, o il conteggio dei collegamenti diretti e indiretti con altri nodi. Uno dei modelli dati più utilizzato per l'implementazione di questo tipo di database è basato sui *property graph*, costituito da tre elementi:

  - *I nodi: sono i contenitori di proprietà*
  - *Le proprietà: si tratta di coppie chiave/valore*
  - *Le relazioni: sono i collegamenti tra i nodi che, in alcune implementazioni, possono contenere proprietà*

  Tra i database di questo tipo citiamo Neo4J.

Applicazioni

MapReduce all'opera

Anche se il lavoro svolto dal framework MapReduce, può apparire piuttosto semplice, in realtà esso permette di svolgere svariate operazioni sui dati. Identifichiamo le principali:

- *Indicizzazione di contenuti:*[35] è il componente principale su cui si basano tutti i motori di ricerca (non a caso Hadoop si è sviluppato su un lavoro di Google)
- *Analisi dei grafi*: si tratta di strutture complesse, molto in uso ad esempio nei social network
- *Data Mining e machine learning*: permettono di risolvere problemi di classificazione e di clustering che, ad esempio, sono alla base delle tecniche di riconoscimento del linguaggio naturale

Analizziamo nel seguito alcuni algoritmi che possono essere sviluppati con l'utilizzo del framework MapReduce.

Ricerca di testo

La ricerca di una parola o gruppo all'interno di un testo è un'operazione che si può svolgere utilizzando la funzione di *map*, senza quella di *reduce*. In tal caso, la prima si occuperà di applicare le condizioni di ricerca sul testo, fornendo in uscita le righe corrispondenti, senza passare per la funzione *reduce*. Questa condizione è realizzabile impostando, a cura del JobTracker, dei job *map-only*, indicando un numero pari a zero di task *reduce*; il JobTracker, dopo l'esecuzione dei task di *map*, comanderà ai task di *map*, di scrivere i risultati su HDFS.

Ordinamento

È questa un'operazione molto semplice per il framework MapReduce, perché come detto in precedenza, le coppie chiave/valore prodotte durante la fase di *map* sono già ordinate. In tal caso, la funzione di *reduce* si limita a comporre i risultati in uscita, senza ulteriori manipolazioni.

Calcoli statistici

Anche questo compito risulta abbastanza semplice per MapReduce. La funzione di *map* in questo caso esegue dei calcoli in locale (es. calcolo del minimo, medio, massimo, conteggio, somme), sulle coppie chiave/valore,

[35] Questa tecnica prende anche il nome di inverted index, perché permette, da una parola, risalire ai documenti che la contengono, che è quanto fanno i motori di ricerca

che verranno poi inviate alla funzione *reduce* per il calcolo globale. Nel calcolo della media, ad esempio, la funzione *map* si occuperà di determinare il conteggio degli elementi e la loro somma, che verrà poi ulteriormente elaborata da *reduce* per determinare il conteggio e la somma globali per procedere col calcolo della media.

Bloom filter[36]

Si tratta di un algoritmo statistico, ideato da Burton Howard nel 1970, che permette di stabilire se un elemento appartenga o no ad un determinato insieme. Permette un uso molto efficiente dello spazio necessario a memorizzare le informazioni e, per ogni elemento, di sapere con sicurezza se esso non è presente nell'insieme (assenza di falsi negativi), mentre nel caso la ricerca dia esito positivo, allora esiste una qualche probabilità che invece esso non sia presente (presenza di falsi positivi). Uno dei suoi utilizzi è, ad esempio, durante l'esecuzione di un'operazione di JOIN di due input, all'interno di Hadoop, qualora ci sia corrispondenza sulla base di una chiave. Il bloom filter è un array di bit ognuno dei quali indica se un elemento appartiene all'insieme che l'array rappresenta. Il suo funzionamento è il seguente:

- Durante l'inizializzazione viene definita la dimensione del filtro (*n*), in termini di numero di bit e il numero di funzioni di hash (*h*) che costituisce l'indice dell'array di bit
- Per ogni elemento del set, le *h* funzioni di hash producono ciascuna un numero di indice dell'array e i bit corrispondenti sono posti pari a 1
- Per verificare se un elemento appartiene al set, si determinano con le funzioni di hash, gli indici dell'array e si verifica che il valore del bit sia pari a 1 per tutti; se uno dei bit è a 0, allora l'elemento non fa parte del set

Se definiamo con *x* il numero di elementi del set, allora la probabilità che vi siano dei falsi positivi è data dalla seguente formula:

$$P = (1 - \frac{e^{-hx}}{n})^h$$

In questa formula *x* è noto, mentre *n* dipende dalla quantità di memoria che si vuole adoperare; a questo punto *h* è ricavabile tramite la formula:

$$h = \frac{n}{x} \ln\ 2$$

se ipotizziamo di aver 10^8 elementi nel set e di utilizzare 8 bit per elemento avremo:

$$x = 100000000$$

$$n = 100000000*8$$

$$h = 8*\ln\ 2 = 5.545177444 \approx 6$$

$$P = (1 - \frac{e^{-6}}{8})^6 = 2.158\%$$

Quindi esiste una probabilità di poco superiore al 2% che si abbiano dei falsi positivi, utilizzando 6 funzioni di hash. Lo spazio occupato dal filtro sarà di circa 96MB; ipotizzando che l'insieme contenga stringhe di 50 byte, l'occupazione del set sarebbe di circa 4,6GB, mentre con l'utilizzo del filtro tale spazio si riduce a circa il 2%. Hadoop, nelle ultime versioni, implementa nativamente questo filtro.

Operazioni SQL-like

Questo tipo di operazioni fa riferimento a quelle che possono essere effettuate, nei RDBMS, utilizzando il linguaggio SQL, tipicamente filtri, aggregazioni o operazioni sui dati, come somma, conteggio, etc., ma anche operazioni di JOIN tra tabelle che abbiano una chiave in comune. In MapReduce, alcune operazioni di aggregazione possono essere svolte dalla funzione *map*, mentre altre da *reduce*. Le operazioni di filtro vengono svolte direttamente da *map*, mentre *reduce* può svolgerle su dati aggregati. L'operazione di JOIN,

[36] Si veda, https://en.wikipedia.org/wiki/Bloom_filter

invece, è possibile, ad esempio, marcando con dei tag i file in ingresso, a cura della funzione di map, per tipologia, salvando i dataset così creati in uscita. A questo punto la funzione di reduce, può sulla base dei tag e delle chiavi, effettuare delle operazioni di unione tra i dataset. Il limite di questa soluzione è che occorre scrivere una funzione di map specifica per ogni tipologia di file in ingresso. Esiste un'altra soluzione che utilizza delle classi chiamate *datajoin* che è un framework generico per l'esecuzione di JOIN in Hadoop. In sintesi queste classi permettono di specificare per ogni *datasource* (file), un *tag* e una *group-key*, che è la chiave da utilizzare per il JOIN tra i datasource. In tal caso la funzione di map ha il compito di identificare tag, group-key e valori nei datasource, mentre reduce di occuperà di effettuare il join. Il limite di questo metodo è che l'operazione di join carica la fase finale del job di MapReduce; è possibile anticipare questo lavoro nella funzione di *map*, se è possibile caricare in memoria la porzione dei file su cui deve essere eseguito il join parziale. Operazioni SQL-like su Hadoop, come quelle descritte, sono semplificate utilizzando lo strumento Hive, come accennato nella parte introduttiva.

Alcuni esempi reali di utilizzo dei Big Data

Le informazioni che seguono riferiscono di casi reali di utilizzo dei Big Data, da parte di aziende per lo più americane. L'analisi completa, condotta Douglas Laney, analista di Gartner[37], riporta ben 55 casi, ma nel seguito ne verranno elencati solo 10:

1. **Macy**: l'azienda, famosa nella grande distribuzione organizzata, utilizza la tecnologia Big Data di SAS Institute per cambiare quasi in tempo reale i prezzi dei suoi prodotti per circa 73 milioni di oggetti. La variazione di prezzo avviene in base alla domanda e alla quantità di prodotti presenti in magazzino, in modo da ottimizzare sempre i costi e i guadagni

2. **Tipp24 AG**: è una piattaforma di scommesse e lotterie europee e usa l'algoritmo Big Data KXEN per analizzare miliardi di transazione e centinaia di attributi e sviluppare così dei modelli predittivi per catturare i clienti e personalizzare i messaggi marketing on the fly, con una diminuzione del 90 percento del tempo necessario alla costruzione dei modelli predittivi ai fini marketing.

3. **Wal-Mart Stores**:[38] è una delle catene di acquisti americane più famose nel mondo, ha sviluppato internamente la tecnologia Polaris, una piattaforma Big Data basata sulle analisi testuali, che tramite il Machine learning e l'analisi semantica dei dati, ha migliorato il sistema di ricerca interno allo store Walmart.com con un aumento delle vendite online fino al 15 percento.

4. **azienda di fast-food**: una nota azienda, di cui Laney non rivela il nome, utilizzando i dati di alcune telecamere sulle corsie dei suoi drive-in, è riuscita a modificare il menù da visualizzare sui display per le ordinazioni, in base lunghezza della fila di automobili. Se la fila è lunga vengono proposti menù veloci da preparare, mentre se la fila è corta, possono essere proposti menù più elaborati. In questo modo, vengono ottimizzati i tempi di attesa dei clienti, evitando che i clienti abbiano un ricordo negativo dell'esperienza di consumo.

5. **Morton's The Steakhouse:**[39] questo utilizzo è a metà tra lo scherzo e la realtà. Un cliente invia un tweet ad una steakhouse di Chicago, scherzando sul fatto che siccome il suo aereo farà tardi, avrebbe gradito ricevere la cena al Newark airport. La cosa non poteva essere più seria per Morton, che con l'utilizzo dei Big Data è riuscito a risalire al fatto che si trattasse di un cliente abituale, che erano noti i suoi ordini passati riuscendo, per tempo, ad inviare un cameriere ad aspettare lo stupito cliente. Si trattava di una trovata pubblicitaria che è poi diventata virale sui social, ma non sarebbe stata possibile senza il supporto delle tecnologie del Big Data.

6. **PredPol:**[40] l'idea è quasi bislacca; utilizzare un algoritmo per la predizione dei terremoti, in uso alla città di Los Angeles, per sviluppare, con il dipartimento di polizia di Santa Cruz, un team di scienziati e una società di nome PredPol un software che è in grado di prevedere dove i crimini sono maggiori con una precisione di circa 50 metri quadrati. La cosa bella è che nelle zone della città dove viene

[37] Fonte: http://searchcio.techtarget.com/opinion/Ten-big-data-case-studies-in-a-nutshell
[38] Si veda https://www.dezyre.com/article/how-big-data-analysis-helped-increase-walmarts-sales-turnover/109
[39] Si veda http://shankman.com/the-greatest-customer-service-story-ever-told-starring-mortons-steakhouse/
[40] Si veda http://www.predpol.com/data-mining-crime-predictions/

usato questo algoritmo Big Data, c'è stata una riduzione dei furti del 33 percento e del 21 percento per i crimini violenti!

7. **Tesco PLC**:[41] una delle più grandi catene alimentari al mondo, la Tesco PLC, ha raccolto in un data warehouse tutte le informazioni provenienti dai punti frigo dei propri supermercati e con strumenti di Big Data analizza ogni singolo dato, per valutare l'efficienza del frigoriferi, verificarne le prestazioni e predire l'eventuale necessità di assistenza e manutenzione, con un decremento dei costi di chiamata ed energetici.

8. **American Express Co.**:[42] usa i Big Data per cercare tutti quegli indicatori che servono a predire la chiusura anticipate dei conti finanziari. La società ha sviluppato sofisticati modelli predittivi per analizzare le transazioni storiche e 115 variabili differenti, con cui AmEx è ora capace di individuare circa il 24 percento dei conti che verranno chiusi nei quattro mesi successivi.

9. **Express Scripts Holding Co.**: una delle associazioni di farmacie più grandi degli USA, ha utilizzato i Big Data per ricordare ai pazienti, con una chiamata automatica, quando è il momento di assumere un farmaco, sulla scorta del fatto che i pazienti tendono a dimenticare di prendere le medicine

10. **Infinity Property & Casualty Corp.**:[43] in questo caso, l'azienda ha utilizzato dei dati 'trascurati' (dark data), ossia quelle informazioni sottoutilizzate, raccolte per singoli fini e poi archiviate. L'azienda ha pensato che questi dati potessero essere recuperati dagli archivi per ottenere dei rapporti specifici sui casi di frode. In questo modo, ha costruito un algoritmo che ha permesso di raccogliere circa 12 milioni di dollari in recuperi surroga.

Infine riporto un utilizzo di Big Data, svolto in Italia da TIM nel luglio del 2016[44], in occasione del Giubileo. In pratica si sono adoperate, in forma anonima e aggregata, le informazioni delle celle e dei terminali mobili TIM registrati, per tracciare una 'mappa di calore', così è stata chiamata, dei propri clienti sovrapposta alla toponomastica della città di Roma, allo scopo di visualizzare la concentrazione delle folle, anche in mobilità e di predisporre dei piani di gestione del traffico e della sicurezza.

Conclusioni

Nel 2011, Secondo la TDWI[45], i Big Data negli Stati Uniti e nei paesi esteri, erano visti come un'opportunità. Presentavano, tuttavia, alcune problematiche legate alla mancanza di skill e, a seguire, dei costi elevati. Dallo studio emerge che Hadoop/MapReduce e i database NoSQL sono tecnologie con ampio potenziale di crescita, ma sulle quali non c'è un impegno forte, contrariamente al data mining, analisi predittiva e avanzata, visualizzazione dei dati, che presentano sia un ampio potenziale di crescita, sia un forte interesse ad implementarle da parte delle aziende. Questo studio è superato oggi, grazie alla riduzione dei costi, dovuta anche alle soluzioni cloud e al crescere delle persone con skill adeguato, avvenuta negli ultimi anni. Basta controllare su sito di Hadoop quanto sia diventato elevato il numero delle aziende che lo utilizzano. In Italia, invece, la situazione è descritta in uno studio effettuato dalla SDA Bocconi School of Management realizzato in collaborazione con IBM[46], che riguarda 202 imprese a dimensione media o medio grande, appartenenti a diversi settori industriali, incluso la PA e la Sanità. Lo studio rileva i seguenti risultati:

- Le fonti di Big Data che sono ritenute più importanti sono i social network e social media, documenti digitalizzati, mail, transizioni, immagini e video, dati GPS, dati di sensori o misuratori digitali

- Gli ambiti che queste aziende intendono analizzare con l'utilizzo dei Big Data sono i comportamenti di consumo dei propri clienti allo scopo di analizzarli per creare prodotti/servizi che abbiano un maggiore appeal

[41] Si veda http://www-03.ibm.com/software/businesscasestudies/mx/es/corp?synkey=Y899603V42608M66
[42] Si veda https://www.datanami.com/2015/04/15/amex-adopts-machine-learning-to-crunch-more-data/
[43] Si veda http://www.insurancetech.com/infinity-property-and-casualty-builds-a-smarter-system-for-fraud/a/d-id/1313275?
[44] Si veda http://www.telecomitalia.com/tit/it/innovazione/giubileo-big-data.html
[45] The Data Warehouse Institute, https://tdwi.org/Home.aspx
[46] Il report è disponibile al link www.sdabocconi.it/sites/default/files/upload/pdf/Report-BigData_final.pdf

- Il 57% delle aziende sta analizzando il fenomeno e i possibili benefici per il proprio business, il 24%, pur comprendendone i benefici ha altre priorità di investimento in ambito IT, il 4% sta preparando una strategia per affrontare i Big Data, il 5% ha già lanciato un progetto in tal senso, il 2% sta cercando un business partner per affrontare la problematica. L'8% ritiene di avere già strumenti sufficienti per le proprie esigenze, il 5% non esprime un'opinione, il 25% ritiene di non avere bisogno di gestire i Big Data
- Fattori critici che favoriscono o impediscono l'adozione dei Big Data sono:
    - il budget disponibile
    - la valutazione del ROI
    - la volontà di realizzazione del management
    - la presenza di competenze

non mancano di detrattori, convinti che Big Data corrisponda ad una moda del momento, o chi è convinto che bastino gli RDBMS per gestire i dati aziendali. È chiaro a questo punto come queste posizioni non siano conciliabili, per il semplice fatto che nessun RDBMS può svolgere il lavoro di una piattaforma di Big Data, anzi come sia possibile, attraverso l'uso dei Big Data, fornire dei dati aggiuntivi alle normali analisi tramite BI o Data Warehouse.

Abbiamo riportato esempi di aziende che hanno utilizzato i Big Data per percorrere strade mai battute: chi vuole osare è avvisato.

Bibliografia

Rezzani Alessandro Big Data, Architettura, tecnologie e metodi di utilizzo di grandi basi di dati [Libro]. - Milano : Maggioli Editore, 2013.