POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale

Sistemi per l'elaborazione del flusso di dati nell'Industria 4.0

Studio di un framework Open Source per la progettazione di un sistema atto alla rilevazione delle anomalie usando il Machine Learning e il Complex Event Processing

Relatore:
prof. Giorgio Bruno

Candidato:
Andrea Bianchi
matricola: 230868

Tutore Aziendale
AEC Soluzioni s.r.l.
Ing. Davide Cecconi

Luglio 2018

"Chi controlla il passato controlla il futuro.
Chi controlla il presente controlla il passato."

George Orwell, 1984

Indice

Elenco delle figure

Listings

Introduzione

Il mondo dell'industria manifatturiera è ormai avviato alla ricerca di nuove tecnologie per massimizzare l'efficacia della produzione e della gestione delle proprie imprese. A tal proposito negli ultimi anni si è scelto di dare il nome di Industria 4.0 a questa rivoluzione nel settore. Appunto viene rivolta l'attenzione all'automatizzazione dei processi industriali e logistici e al trovare i mezzi necessari per raggiungere lo stato di Smart Factory, ovvero una fabbrica dove l'asset che la compone viene digitalizzato e gli viene permesso di comunicare all'interno del Fog Cloud. L'obiettivo di questo studio è quello di presentare tale iniziativa presa dalle diverse imprese nel settore di aggiornarsi alla digitalizzazione, proponendo un progetto che permetta attraverso l'utilizzo di tecnologie e materie come Stream Processing, Machine Learning e Big Data di realizzare un prodotto che si possa avvicinare allo stato di Cyber-Physical System. Viene quindi illustrata anche una componente importante per questo progetto, quale il Complex Event Processing, strumento che per eccellenza si distingue nella generazione delle notifiche. A tal proposito sarà fornito un prototipo di progetto, che sarà la pietra su cui saranno basate le fondamenta per costruire una funzionante Control-Room per l'Industria 4.0, la quale si possa interfacciare sia alle macchine che ai gestionali di azienda, come i sistemi MES e gli ERP.

La tesi è strutturata in quattro capitoli che verranno descritti brevemente di seguito.

Il **primo capitolo** servirà di introduzione a quelle materie e tecnologie delle scienze informatiche che hanno permesso lo studio del progetto e realizzare il prototipo. Verranno quindi discussi i problemi e le opportunità del Big Data, sarà fornita un'introduzione allo Stream Processing e al Complex Event Processing. Si farà poi riferimento al Machine Learning.

Il **secondo capitolo** si contestualizzerà sulla descrizione dell'Industria 4.0, come è nata e quale è il paradigma per definire una Smart Factory. Verrà poi fornito un piccolo résumée sul problema imposto che caratterizza il progetto e una breve descrizione dell'azienda dove si è svolto il tirocinio.

Il **terzo capitolo** prevederà, invece, la discussione del progetto, quindi dei framework e dei prodotti studiati e delle features di quest'ultimi. Per realizzare un primo prototipo di topologia per lo Stream Processing sono stati utilizzati Apache Kafka, Apache Storm, MongoDB e Esper. Viene fornita poi una breve descrizione sull'implementazione del prototipo svolto per l'esperimento.

Il **quarto capitolo** prevede invece l'interpretazione delle misurazioni prese dal testing del prototipo e quindi i passi futuri da prendere per riuscire a realizzare un prodotto utilizzabile in ambito Smart Factory.

La tesi verrà poi chiusa dalle formali conclusioni sul lavoro svolto.

Capitolo 1

Stato dell'arte e introduzione alle materie del progetto

In questo primo capitolo si procederà a presentare la definizione e alcune digressioni teoriche riguardo appunto le materie dello **Stream Processing**, del **Complex Event Processing** e delle tecniche di **Machine Learning** per il **Data Mining**. Saranno appunto riportate quindi fonti di vari autori esperti in materia, evidenziando le parole e i concetti chiave che serviranno poi al lettore per comprendere meglio il contesto su cui si basa il progetto di tesi. Ovviamente prima di queste digressioni, si discuterà anche a riguardo sull'utilizzo del **Big Data** e del processo di Data Analysis. Questo serve ad introdurre sia lo Stream Processing che il Machine Learning, dato che queste materie sono branche dello studio del Big Data, nate per fornire strumenti utili per l'esplosivo fenomeno dell'ultimo ventennio nell'uso di database non strutturati e dello scambio di diversi tipi di dati a dimensione variabile e ad una frequenza di transazioni molto grande. Quindi si discuterà anche delle sfide che ha fatto nascere questo fenomeno in continua evoluzione e cambiamento e delle opportunità e i vantaggi che può offrire. Tutto ciò è dovuto al fatto che il problema che si esaminerà nel proseguimento della tesi riguarderà appunto al caso d'uso del Big

Data nell'ambito industriale, specificamente nel contesto macchina-gestionale.

Una tecnica piuttosto difficile, ma comunque molto utilizzata in parallelo o assieme allo Stream Processing, è quella del **Complex Event Processing**, che permette l'elaborazione di notifiche da parte di varie sorgenti del sistema, per poter ottenere informazioni in real-time utili al monitoring del sistema. Sarà proposta una breve definizione della pratica che sarà poi estesa nel proseguimento della tesi, incentrata sulla presentazione degli attori principali dei framework per CEP e sulle architetture per il quale può essere utilizzato.

L'obiettivo finale di questo capitolo sarà quello di introdurre il lettore al progetto della tesi esponendo una breve paronamica delle materie e come si possono definire in ambito globale, oltre a piccoli cenni della storia che le compongono. Inoltre saranno inseriti alcuni dei problemi che affiorano dalla loro applicazione nel mondo reale e quali sono alcuni dei vantaggi che possono presentare in maniera globale.

1.1 Definizione di Big Data

Si introduce il capitolo con lo studio delle varie sfaccettature e dei problemi nati con l'utilizzo del Big Data.

Come introduce Madden nel suo articolo [1], storicamente la gestione dei dati è stata effettuata per molto tempo attraverso l'uso di Database Relazionali e del linguaggio SQL seguendo il paradigma Atomicity, Consistency, Isolation e Durability (*ACID*). Si suppone quindi l'uso di transazioni che soddisfino queste proprietà sopracitate. Ma nel caso di Big Data, quindi quando i dati diventano troppo grandi, frequenti o di difficile associazione e lettura, i normali DB relazionali o la gestione delle transazioni possono ancora garantire il rispetto dell'ACID? Appunto molte compagnie si sono trovati di fronte a gestire moli enormi di dati, che raggiungono l'ordine del petabye, che vengono ingestiti e devono essere processati in tempi molto

brevi e che molte volte contengono informazioni di difficile estrazione con i tool di elaborazione, implementati apposta per le diverse applicazioni che utilizzano questi dati.

Come riporta questo articolo [2], i Big Data sono utilizzati maggiormente per descrivere enormi dataset composti soprattutto da informazioni disorganizzate. Ciononostante, da tale garbuglio è possibile trarre deduzioni più complicate sul processo producente tali dati, quindi si garantisce la possibilità di evolvere e migliorare il sistema esaminato. Questo aspetto ha interessato molte aziende, compagnie ed industrie che si sono affacciate su questo tema. Inoltre, sono state create nuove figure professionali apposta per indagare sui dati prodotti, chiamati appunto **Data Scientist**. Però, oltre ad avere il problema della crescita di numero di questi larghi dataset, vi sono problemi che derivano dalla generazione di questi dati, della loro massività, della difficile integrazione negli storage e della loro provenienza, quindi sono problemi che riguardano l'organizzazione delle sorgenti di dato. Si pensi appunto al mondo del Cloud Computing e dell'Internet of Things, che si basano proprio sulla larga distribuzione del dato, che per quantità e per mutue relazioni tra le sorgenti si manifesta una saturazione della capacità di calcolo e di storage delle architetture esistenti nelle diverse aziende che ne fanno uso, soprattutto per quanto riguarda la possibilità di elaborare il dato in real-time. Considerando poi l'eterogeneità, la scalabilità e la privacy dei Big Data, occorre elaborare l'informazione in diverse fasi per fornire un'accurata analisi e il monitoring.

Dato che il concetto di Big Data è essenzialmente astratto e non descrive un paradigma in particolare, i vantaggi e gli svantaggi di questo incremento dei dati gestiti sono stati modellizzati per la prima volta dall'analista Doug Laney attraverso il modello a 3V. Nello scritto di Laney [3] si rappresenta quindi l'incremento della Velocità, Volume e Varietà del dato al crescere dei dataset:

- **Velocità:** con l'avvento dei Social Media e di tutte le applicazioni real-time

di digitalizzazione del dato sensoristico, è cresciuto anche la velocità del POI (Point-of-Interaction), quindi della frequenza con la quale i dati vengono generati dalle interazioni. Questo indice di performance viene spesso utilizzato come coefficiente per la concorrenza sul mercato. La gestione della velocità del dato riguarda molto di più rispetto a conoscere la larghezza di banda del mezzo sul quale passa il dato o di conoscere le specifiche di protocollo. Per ovviare all'alta frequenza di ingestione, le aziende hanno sviluppato diverse soluzioni come:

  - **Datastore operazionali (ODS):** periodicamente estraggono, integrano e riorganizzano i dati per ottimizzare il Data Mining o le operazioni di interrogazione.
  - **Caching:** sono piccole strutture che permettono di avere un accesso istantaneo alle transazioni e riducono il carico del back-end server attraverso il buffering dei record.
  - Miglioramento del trasferimento Point-to-Point del dato tra applicazione e database.
  - Architetture che bilanciano il carico e la latenza del dato attraverso politiche e configurazioni adattate all'applicazione d'uso e al suo ciclo di decisione. Non si suppone che l'intera catena dell'utilizzo del dato debba soddisfare le specifiche del real-time.

- **Volume:** si considerino i Social network (*a.e.* Youtube, Facebook...) e si pensi alla quantità di dati prodotti da questi in un breve periodo di tempo. Si parla di milioni di utenti che caricano immagini, video, etc. su tali piattaforme contemporaneamente. Appunto un problema di Big Data è rappresentabile anche dalla quantità di dato ingerita per una transazione. Dall'analisi di Laney risulta che, visto il basso costo dei canali di ingestione del dato e visto

l'interesse degli stakeholder di un'azienda a esaminarli, in media si raccolgono all'incirca dati di dimensione di dieci volte maggiori rispetto all'informazione effettiva che tali dati rappresenterebbero. In più l'azienda vede il dato come asset con un suo valore, perciò questa risulta riluttante ad eliminarlo. Perciò, come soluzione spesso si acquistano nuovi storage di dimensione maggiore per contenere tutto il raw data non elaborato. Comunque il valore del dato decresce con l'ammontare delle informazioni raccolte, ciò diventa un motivo per il quale acquistare nuovo storage diventa una soluzione inefficace. Delle soluzioni alternative possono essere:

  - **Implementazione di sistemi DB complessi:** questi vengono organizzati in modo da bilanciare la forbice costo-efficienza, basandosi sulla disponibilità e utilità del dato usando diversi mezzi e algoritmi.
  - **Filtraggio:** si limita il dato in ingresso, considerando solo le informazioni relative al processo corrente o futuro, oppure lo si può limitare basandosi solo su campioni di validità statistica.
  - **Identificazione della sorgente:** permette di identificare e quindi eliminare le ridondanze di dato.
  - **Monitoraggio del dato immagazzinato:** si possono rilevare le zone meno interrogate di un database e successivamente le si eliminano oppure se ne esegue un backup.
  - Appaltare la gestione del dato a compagnie che offrono servizi in base alle necessità.

- **Varietà:** con l'introduzione di sistemi complessi per le svariate applicazioni, spesso vengono integrate sorgenti di dato diverse fra loro. Ciò porta a una varietà del dato in ingresso. Ci sono diversi tipi di dato: quelli strutturati, i

semi-strutturati come i file log, i non strutturati e i dati ibridi. Ciò può portare a una gestione di dato per tipi incompatibili fra loro, su strutture disallineate e può portare anche a una semantica insignificante dell'informazione senza un contesto rigidamente strutturato attraverso entità e relazioni. Motivi della perdità di contesto possono derivare anche dalla realtà aziendale, sempre secondo Laney, dato il fatto che la maggior parte delle compagnie ora si estende con diverse partnership o a causa di acquisizioni o per le formazioni di holding. Per risolvere questo problema di solito si ricorre a:

- **Definizione di dizionari dei dati:** in modo da allineare le relazioni e togliere le inconsistenze che possono risultare avendo sorgenti di dato differenziate.
- **Uso del formato XML:** l'**eXtended Markup Language** è nato per essere un metalinguaggio universale usato apposta per essere direttamente utilizzabile su Internet e per essere usato dalla maggior parte delle applicazioni. È lo standard realizzato sotto la **W3C**, nato apposta per allineare la struttura del payload informativo delle applicazioni Web. Un esempio particolarmente rilevante è l'utilizzo di questo standard per realizzare le richieste SOAP ad un Web Server. Questo è il mezzo maggiormente usato per risolvere il problema della varietà. Un'altro metalinguaggio utile a strutturare il payload, soprattutto in IoT per le reti di sensori, è il JavaScript Object Notation (**JSON**).
- **EAI:** definite come Enterprise Application Integration, in parole povere sono API realizzate dalle aziende per permettere a loro di acquisire o trasferire il dato attraverso code di messaggi o connettori. Tali API possono essere anche distribuite agli interessati per realizzare dei driver, utili a standardizzare le chiamate alle applicazioni sviluppate dalla azienda distributrice.

- **Data Access Middleware:** per maggiore comprensione, il middleware è un pacchetto di software e librerie usate per l'interfacciamento tra strutture e sistemi che prevedono l'utilizzo di protocolli o sistemi operativi differenti. Nel caso dei Data Access Middleware, questi vengono utilizzati per l'accesso diretto tra applicazione e database. Assieme a questi si affiancano sistemi per la gestione distribuita delle query, per rendere più intelligente il middleware aggiungendo un routing del dato e per permettere un'elaborazione al passaggio del dato stesso.
- Soluzioni per la gestione del metadato: per definizione il **metadato** è quella struttura che serve per la descrizione di altri dati. Questi vengono utilizzati per la contestualizzazione dell'informazione "stupida" raccolta attraverso schemi definiti a priori (che ad esempio si possono realizzare grazie gli XML Schema Definition per quanto riguarda il payload in struttura XML).
- Sistemi avanzati per l'indicizzazione dei dati di vari tipi incompatibili fra loro.

Oltre a questa definizione, che è la più utilizzata nei testi, ve ne sono diverse altre. Questo tende a sottolineare che lo studio Big Data è una scienza in continua evoluzione data l'incessante attività nell'implementazione di nuove applicazioni e nella necessità di studiare il dato per le più recenti esigenze che il mercato dei servizi informatici produce.

Lo stesso modello delle 3V di Laney viene aggiornato, aggiungendo delle V a seconda della nuova caratteristica che descrive il problema, come afferma l'articolo [4]. Una delle più rilevanti è il **Valore**, ovvero l'utilità intrinseca del dato collezionato, cioè il valore aggiunto per la decision making dell'attività, oppure è definito anche come il suo valore nell'analisi predittiva. Altre V utilizzate possono essere: *Veridicità,* proprietà dell'informazione di essere veritiera anche dopo una possibile

manipolazione o accesso non autorizzato; *Volatilità,* periodo di tempo per il quale il dato può essere ritenuto interessante e valido, quindi il tempo di persistenza utile; *Variabilità,* consistenza del dato nel tempo. Esiste anche la **Complessità**, che riguarda il grado delle dipendenze e della connessione dei dati. Con questa si viene a misurare l'effetto che un cambiamento del sistema può portare a tutta la struttura del Big Data collezionat e quanto può essere onerosa la ristrutturazione di tutto l'asset informativo.

Grazie a queste caratteristiche, un'azienda può identificare il problema al quale si trova di fronte. Quindi può scegliere in maniera efficiente le pratiche e le tecnologie da poter utilizzare per gestire al meglio la catena di ingestione del dato e il suo immagazzinamento. Inoltre le varie branche della materia, si veda Batch o Stream Processing oppure studio di dati sparsi o alta densità etc., sono definite navigando appunto il modello delle 3V.

1.1.1 Le operazioni sul dato fini alla generazione dell'informazione, i problemi scaturiti e le sfide tecnologiche

Come espone l'articolo [5], generalmente un sistema che permetta l'ingestione e integrazione del dato in un database per il Big Data permette cinque operazioni principali per riuscire a raggiungere la decisione e ottenere l'applicazione atta allo sfruttamento del tesoro informativo. Da queste operazioni comunque si possono scaturire diversi problemi di carattere tecnologico e se non è possibile sormontarli la persistenza del dato potrebbe rilevarsi tanto inutile quanto inefficace.

- **Cattura del dato e storaging:** come già riportato, la crescita del volume e della velocità del dato nel Big Data porta le aziende a correre incessantemente all'acquisto di nuovi supporti hardware per garantire il salvataggio di tutti i dati raccolti. A volte però il costo sostenuto per correre ai ripari può motivare

gravi perdite, se non si esamina a priori il valore intrinseco del dato raccolto stesso. A fronte di questa corsa, i meccanismi di immagazzinamento e di accesso al dato sono cambiati drasticamente. Soprattutto per quanto riguarda l'accesso, che ha massima priorità appunto per lo studio e il monitoring del dato atto a ottenere la knowledge decision, deve essere reso facile e veloce. Quindi occorre scindere il paradigma tecnologico che è stato imposto nella produzione di hardware nell'ultimo ventennio, cioè architetture di elaborazione molto veloci per le operazioni di CPU ma molto povere in quelle di I/O. In passato la maggior parte dei dati persistenti era posta in hard disk magnetici, dove si aveva maggiore performance per le operazioni di I/O sequenziali piuttosto che per quelle ad accesso casuale. Ciò imponeva un vincolo molto importante sul formato del dato e sulla elaborazione delle query. Il passaggio poi ai dischi a stato solido (SSD) ha portato un bilanciamento maggiore per quanto riguarda le prestazioni d'accesso. Ciononostante questo ha portato ad un ulteriore ripensamento delle architetture per i sistemi dell'elaborazione dei Big Data.

I paradigmi di architettura odierni usati per lo server storaging aziendale sono spesso i DAS, NAS e i SAN. Comunque queste architetture hanno brutte aspettative e performance per quanto riguarda i sistemi distribuiti di larga estensione, dove throughput e concorrenza di alto livello diventano requisiti essenziali per sostenere le applicazioni di Big Data. Quindi ci si porta ad ottimizzare l'accesso con tecniche di replicazione, migrazione e distribuzione del dato, oltre a inserire certi gradi di parallelismo. Inoltre si punta a cambiare anche lo schema del sistema applicativo per garantire maggiori performance di storaging, passando da applicazioni server-centered, con alta concentrazione del dato persistente sul server master delle architetture, ad applicazioni distribuite, dove l'onere dello storage e suddiviso in parte anche sui client.

- **Trasmissione del dato:** con l'avvento delle tecnologie Cloud, lo storage su remoto sta diventando sempre di più largo utilizzo. Un probabile collo di bottiglia però può essere imposto dalla banda della rete, soprattutto con un largo volume delle informazioni distribuite. Inoltre possono incombere anche problemi di sicurezza, quindi occorre garantire integrità e autenticazione nella comunicazione, oltre a certificare la riservatezza. Diversi sono i sistemi utilizzati per garantire sicurezza sul canale (come TLS a livello applicativo, o IPsec con l'uso dell'ESP e dell'AH a livello trasporto), ciononostante l'uso di crittosistemi può portare a maggiore latenza di elaborazione e comunicazione, soprattutto in fase di handshaking e di cifratura/decifratura del pacchetto trasportato.

- **Data curation:** con tale termine si indicano tutte le tecniche e pratiche che mirano alla gestione del dato nel database, in modo tale da rendere il dato riutilizzabile nel knowledge discovery. Quindi si possono avere delle sotto-operazioni come autenticazione del dato, ovverosia l'accertazione che il dato in storage sia stato inserito da fonti autenticate, archiviazione, gestione, possibilità di prelevamento e rappresentazione. Per quanto riguarda l'approccio dei tradizionali DBMS, questi dividevano il compito in due parti, uno era implementare uno schema di storage del dataset e per il prelevamento si utilizzava lo schema relazionale. Per dataset strutturati di larga scala invece si procede con l'approccio del **Data Warehouse**, che altro non è che un database relazionale sempre basato su SQL con sistemi di analisi e stroage automatizzati e schedulati, atti a proporre report periodici all'utente. In questo caso, funzioni di preprocessing del dato, pulizia e trasformazione sono essenziali, soprattutto per ovviare a problemi come campi vacanti, sparsità e volume dello storage.

  Oggigiorno, dataset di largo volume sono spesso gestiti su database **NoSQL**, che ovviano ad alcuni problemi di rigidità che i tradizionali DBMS relazionali

possono presentare, ad esempio le tabelle con schema fisso. Presentando schemi per dati non strutturati, i NoSQL separano gli approcci per lo storage e per la gestione del dato, rendendoli indipendenti tra loro. Nella parte di storage, si usano approcci basati su strategie chiave-valore per permettere una scalabilità del dato ad alta prestazione. Invece, per quanto riguarda la gestione i NoSQL offrono meccanismi di accesso a basso livello e quindi driver specifici per scrivere applicazioni permettenti chiamate dirette al sistema di gestione senza dover tradurre in precedenza la logica in SQL. Forniscono quindi una modellizzazione dei dati flessibile e la possibilità di realizzare applicazioni facili da aggiornare e da portare in funzione.

- **Analisi del dato:** con l'avanzare del tempo, gli algoritmi di analisi sono stati migliorati per far fronte all'aumento del volume dei dati e per essere maggiormente scalabili. Si è ottenuto anche un netto miglioramento nella velocità di esecuzione di tali algoritmi, grazie alla produzione di processori più potenti. Diventa necessario quindi introdurre anche metodi di analisi che funzionano in real-time e che permettano di campionare il dato in entrata. Per quanto riguarda gli algoritmi di Machine Learning, solo l'Incremental Computing, ovverosia quella tecnica che permette di prevedere l'output in base al cambiamento che si è osservato sull'input al sistema, è quello che risalta di più per termini di scalabilità. Per quanto riguarda invece la potenza di calcolo del processore, si registra un aumento della dimensione del dato maggiore rispetto a quanto aumenta la frequenza del clock. Appunto si sono sviluppate e sono state migliorate le tecniche di calcolo in parallelo.

  Effettivamente, un analisi che non presenta ritardi e sia altresì preventiva ha un'alta priorità per le applicazioni che lavorano in real-time, e la possibilità di elaborare dati di volume sempre crescente è molto importante per sviluppare

un'applicazione competitiva per il mercato. Questa è appunto una sfida che lo Stream Processing può eliminare.

Un'altra sfida viene presentata dalla necessità di rendere sicura e protetta l'analisi del dato. Oltre a questo si pensa come il dato sia immagazzinato in maniera distribuita e quindi l'analisi può essere rallentata per la sicurezza dello scambio dei pacchetti di informazioni sulla rete di cluster.

- **Rappresentazione del dato:** l'obiettivo della rappresentazione è quello di visualizzare le informazioni che emergono dal dataset attraverso grafici e report. L'informazione che viene visualizzata diventa poi anche molto importante per l'analisi. Questo può essere molto difficile se si pensa alla larghezza che il dataset Big Data può raggiungere. Ciononostante i tool grafici più distribuiti per questo compito hanno performance abbastanza basse, sia per funzionalità, scalabilità e prontezza. Un'altra sfida è data dall'incertezza del dato e nuovi framework per modellizzarla sono necessari per il processo di analisi.

1.1.2 L'analisi di Big Data e le prospettive per il futuro

Per quanto riportato dalla fonte [6], ultimamente l'analisi di Big Data (Big Data Analytics) come termine viene spesso incluso in Big Data, e definendo quest'ultimo infatti a volte si và a descrivere quelle che sono le tecniche analitiche atte all'estrazione di informazioni da dataset enormi che necessitano tecnologie avanzate per lo storage, gestione e rappresentazione. Tali tecniche provengono da un vasto numero di discipline come la statistica, il Data Mining, il Machine Learning, etc. e il confine tra queste a volte può essere anche molto sottile. Sono tecniche tutte molto utili che possono essere utilizzate in svariate applicazioni.

Si può constatare che l'approccio analitico in precedenza si soffermava solo a fornire risultati descrittivi e diagnostici, quindi si basava su dati storici e correnti come

sorgente per identificare dei pattern e correlazioni tra le informazioni raccolte, usare semplici indicatori statistici come media o deviazione standard oppure trovare la radice di un problema che è sorto per risolverlo e prevenirne un nuovo accadimento. Quello a cui puntano ora le scienze della BDA sono le analisi predittive e prescrittive. Si tratta appunto di utilizzare le nuove tecniche di Data Mining, Machine learning etc. per ottenere appunto una previsione del risultato futuro, estraendo dalla stessa le probabili opportunità e i rischi di una particolare decisione strategica. La particolarità dell'analisi predittiva è quella appunto di utilizzare i pattern nascosti e le possibili relazioni tra i dati collezionati per formulare una previsione. Invece l'analisi prescrittiva permette di realizzare diverse previsioni in base all'impatto di possibili Business Decision future. Questa analisi fa uso di tool avanzati per la modellizzazione dell'ambiente osservato, per permettere di prendere le decisioni più furbe e veloci con bassi costi e rischi e per realizzare soluzioni ottimizzate alla gestione dell'asset aziendale. L'utilizzo dell'analisi predittiva e prescrittiva può giocare molto a favore della strategia di impresa, risolvendo problemi per lo sviluppo e vendita di prodotti e servizi e quelli che riguardano l'organizzazione della struttura.

Gli esperti suddividono la storia della BDA in quattro fasi, dove la maggiore evoluzione di questa scienza si è registrata in questo ultimo ventennio. Le prime analisi statistiche dei dati di processo e di gestione nascono nella metà degli anni 50, anche se in questo periodo non si può proprio parlare di Big Data. Questa viene sancita come l'era degli **Analytics 1.0**, dove nascono i primi tools per la cattura di dati e l'implementazione dei primi algoritmi di pattern mining, che permettono strategie meglio strutturate e che esulano dalla semplice intuizione basata sull'esperienza del manager. Con l'esplosione negli anni 2000 del numero di dati generati e la nascita dei Big Data, ci si apre all'era degli **Analytics 2.0**, dove strumenti più avanzati sono stati implementati per lo studio dei dati strutturati e si è registrata la prima

apertura all'uso dei database NoSQL. Le imprese subito si sono lanciate alla scoperta di queste nuovi tecnologie per l'analisi e, con l'aumentare degli strumenti e delle tecniche, maggiore è diventata la loro complessità ed anche la loro comprensione, richiedendo competenze specializzate nel settore Data Science. Nell'era successiva del **Analytics 3.0** o anche detta era di "Data Economy", la BDA diventa quasi una parte integrante e necessaria per l'asset delle imprese di medio-grande dimensione. Soprattutto in questa era vi è stata l'esplosione della commercializzazione di applicazioni IoT, che, con miliardi di dispositivi connessi, l'aumento del dato generato è diventato esponenziale. Al momento siamo arrivati alla fase **Analytics 4.0** che introduce le tecnologie cognitive come il Machine Learning. L'analisi diventa essenziale, tale per cui vengono prodotte tecnologie hardware ottimizzate alla BDA, le quali in maniera embedded includono molte delle operazioni atte al compito di analisi. Le analisi vengono poi svolte in maniera automatica, introducendo un grado di dinamicità per essere sincronizzate con un certo cambio di contesto.

Ritornando al modello 3V di Laney, è impossibile pensare che i problemi dovuti a volume, velocità e varietà del dato ad un certo punto si blocchino, anzi, con la nascita di nuove applicazioni e tecnologie, ci si troverà sempre di fronte a nuove sfide. Per trovare informazioni migliori, si prevede che per il 2020 verranno effettuati investimenti per più di 203 milioni di dollari, con imprese bancarie, manifatturiere e governative come top investitrici, secondo la International Data Corporation. Si punta quindi a nuove materie, come le già dette analisi prescrittive e predittive, assieme alle nuove tecnologie di analisi di testo e voce, assieme all'Image Recognition, alla Social Media Analysis e alla simulazione di processo. Queste portano alla nascita di nuovi mezzi per permettere alle diverse compagnie di rendersi competitive sul mercato e ottenere continui profitti dalle attività.

1.2 Stream Processing

Generalmente il concetto di Stream Processing riguarda la contestualizzazione di un programma in un insieme di moduli connessi staticamente da una rete di collegamenti. Tali moduli hanno il basilare compito di prendere in input una sequenza di dati, elaborali e di rendere disponibile in output la sequenza di dati trasformata. Come dice [7], questo concetto ha origine dalle reti di Robert Kahn, le quali appunto rientrano nel design delle reti TCP, delle quali a Kahn è stata attribuita la creazione. Storicamente il primo linguaggio che permettesse di implementare tali reti fu il Lucid, nato nel 1974 all'università di Waterloo (Canada). L'obiettivo principale di Wadge e Ashcroft, i loro programmatori, era quello di rivoluzionare il linguaggio dell'epoca, concentrato più sulla programmazione di algoritmi iterativi e strutturali. Difatti Lucid permetteva di tradurre quegli algoritmi in relazioni algebriche utilizzando specifici operatori sulle variabili inizializzate, le quali rappresentavano appunto delle sequenze di dati che venivano elaborate da tale relazioni. In altre parole si tratta di trasformare dei flussi di informazioni, quindi si rientra nella materia dello Stream Processing. Altri linguaggi per il dataflow, come Lustre ed Esterel, sono poi stati pubblicati, ed introdussero i concetti di relazioni ricorrenti tra i flussi di dati e lo sviluppo di reti sincrone per l'elaborazione. Tutt'oggi tali linguaggi vengono usati per la programmazione di sistemi reattivi e di algoritmi per la misura di segnali, che vengono usati nelle applicazioni industriali e nel loro controllo.

Molte applicazioni che sfruttano il Big Data necessitano di elaborare il dato in arrivo in tempo reale, anche per un discorso di valore, ove il dato acquisisce più importanza ed ha informazioni più utili secondo un certo concetto di spazialità temporale. La potenzialità di fornire un'informazione in real-time è ricercata in molti campi e soprattutto permette di far fronte a quella che è lo storaging di tutto il raw-data prodotto da tutte quelle che sono le diverse sorgenti distribuite. Ciononostante, è risaputo che più l'esecuzione di una determinata applicazione deve fornire risultati

in tempo reale, più il sistema che implementa tale applicazione diventa complesso. Le tre caratteristiche princiapli di un sistema di Stream Processing sono:

- Velocità: si tratta appunto di collezionare dati anche da un largo numero di sorgenti che generano flussi a frequenza molto alta, all'incirca si stimano flussi di milioni di eventi al secondo.

- Parallelismo: l'elaborazione parallela è fondamentale, se non altro per ottenere informazioni consistenti e garantire il rispetto del flusso logico dell'elaborazione dei diversi pacchetti che vengono smistati all'interno della rete di processing.

- Correlazione: la logica deve essere distribuita in modo tale da permettere l'estrazione da più eventi arrivati contemporaneamente.

Questi tre punti principali devono essere integrati in un'architettura che permetta un'elaborazione in maniera distribuita e scalabile orizzontalmente e che garantisca la Fault Tolerance, ovvero che gestisca le situazioni di errore senza provocare discontinuità nel processo.

La maggior parte delle aziende tutt'oggi sfrutta la potenzialità di elaborare analisi sul flusso di dato in tempo reale. Questa affermazione è più che sbagliata, in quanto in molti casi si tende a semplificare il sistema pur di applicare strategie che minimizzano gli sprechi. Si tratta quindi di non applicare un'azione che corregga lo sbandamento del processo nel brevissimo termine, ma di prendere un'iniziativa dopo un certo periodo di tempo e su una base di dati già stipata in storage. Si parla quindi di effettuare il **Batch Processing**, che per definizione si tratta di effettuare l'analisi e il processing sul dato in sosta. Il costo dei tool è molto basso e molti sono anche Open Source, ed ormai la conoscenza nell'uso di questi mezzi è più che estesa, oltre che ad avere maggiore documentazione. Tecnologie come Hadoop, Spark (in parte, un modulo è anche adatto alla Stream Analysis) etc. sono utilizzate in molte

applicazioni, ciononostante non sono appropriati in caso di dati non deterministicamente generati dal sistema o real-time dinamici. La tendenza di spostarsi dal Batch Processing allo studio di soluzioni per il tempo reale, si è registrata all'incirca dal 2013, dove sondaggi registrano che la maggior parte delle aziende intervistate sono affascinate o hanno bisogno dell'analisi in diretta. Difatti da quel momento le community Open Source si sono attivate nella realizzazione di framework e tecnologie adatti allo scopo.

Uno degli aspetti più affascinanti delle reti di Stream Processing, è la possibilità che, con un buono studio della scienza e con una certa destrezza tra i meccanismi di Machine Learning, si può realizzare un sistema di analisi con una reattività tempestiva. L'obiettivo è quello di realizzare un sistema che permetta di reagire ai cambi di contesto su cui giace l'applicazione, anche in casi inaspettati. Diversi algoritmi che funzionano online permettono appunto di classificare il dato in arrivo, che se inseriti in un framework di Stream Processing poi è possibile rilevare una situazione o un pattern straordinario e che possa essere utilizzato per uno studio approfondito successivamente.

Col proposito del Big Data, effettivamente i normali DBMS possono non essere necessari per il compito da svolgere, dato che risulta impossibile salvare tutti i dati che vengono generati dal sistema in esame. Il processo di tali dati viene effettuato solo su richiesta, senza la possibilità di inserire la temporizzazione nell'automatizzazione dell'elaborazione dei dati. Per l'obiettivo di eseguire lo Stream Proccessing sono stati appunto evoluti i DBMS per ottenere i **Data Stream Management Systems** (DSMS), che appunto con il processamento di query continue (continuos query) con sintassi SQL-like è possibile effettuare un'elaborazione dei dati preventiva alla loro memorizzazione nel dominio DBMS.

1.3 Complex Event Processing

Introduco l'argomento attraverso gli scritti [8] e [9], i quali a loro volta traggono riferimento dai concetti formulati nel "The Power of Events" di Luckham, luminare teorista appunto dell'Event Processing. Il **Complex Event Processing** è quella scienza che ha formalizzato tecniche e metodi che permettono di visualizzare e reagire ad **eventi complessi** in tempo reale, in modo tale da scaturire subito una notifica di informazione verso gli organi interessati del sistema informatico implementato. Come tecnica la si può definire come specializzazione del Data Stream Processing, dove in quel caso si può lavorare su qualsiasi tipo di dato in entrata per la produzione di nuovi flussi di dati in uscita. Il CEP invece si può definire come un evoluzione dei sistemi di gestione per le code dei messaggi publish/subscribe.

Il punto cardine del CEP, appunto è l'utilizzo di **eventi** o notifiche di eventi. Questi eventi non sono altro che oggetti che rappresentano un record prodotto da una attività del sistema e rappresenta qualcosa che è successo. Tale record è simile ad una tupla di un normale database, sul quale viene utilizzato ed è collegabile ad altri eventi attraverso il tempo, la causa e l'aggregazione. Il tempo appunto è uno degli elementi indispensabili alle architetture CEP, che di solito viene inserito sotto forma di timestamp. Per questo proposito occorre tenere un occhio di riguardo all'utilizzo di temporizzatori differenti all'interno delle macchine che compongono il sistema, dato che la sincronizzazione è molto importante per non arrivare a risultati di analisi errata. Esiste a tal proposito l'**Assioma di Causa-Tempo** di Luckham, il quale afferma appunto che se l'evento A causa B ($A \rightarrow B$), nessun temporizzatore nel sistema può generare un timestamp per B che sia precedente a quello di A. Per quanto riguarda l'aggregazione degli eventi, si può definire l'**evento complesso** come quell'evento A risultante dall'unione delle attività che l'inseme $\{B_i\}$ di eventi rappresenta. Le tre relazioni di tempo, causa e aggregazione sono proprietà transitive e asimmetriche dei sistemi CEP.

Grazie all'aggregazione si possono definire gerarchie di eventi complessi, quindi è possibile partire anche dal basso livello, cioè eventi scaturiti dall'hardware, per poi arrivare a costruire notifiche per il livello applicativo che rientrano nel contesto del sistema. Questo è molto utile anche nel **cause recognition**, ovvero si possono aggregare eventi, quindi formare un pattern, in modo tale da riconoscere la causa della notifica arrivata all'alto livello; questo caso serve soprattutto quando l'anomalia o la causa è straordinaria per il sistema.

Le strutture degli eventi si possono definire in due step: prima occorre osservare qualsiasi livello del sistema sul quale il CEP deve essere utilizzato, senza modificare in alcun modo il funzionamento e il suo comportamento, dopodiché si trasformano tali osservazioni in oggetti evento, quindi si determinano le strutture degli oggetti che il CEP andrà a gestire. Quindi si possono definire tre sorgenti di eventi in un sistema che utilizza il CEP, quali: i componenti informatici dell'applicazione, cioè tutti i componenti utilizzati per mettere in piedi un sistema informatico come database, connettori, il middleware e il processo informatico stesso; la strumentazione, quindi tutti i dispositivi hardware e il software che si occupa di monitorare il sistema o che producono dati per esso; il CEP stesso, dato che si definisce la gerarchia di eventi e quelli che sono stati aggregati, filtrati etc. in precedenza possono essere riutilizzati da altri processi CEP.

I processi CEP si possono separare in due categorie: quelli che reagiscono agli eventi ed eseguono una certa azione e i connettori che trasportano gli eventi tra i processi che eseguono le attività. Quindi questi processi CEP si possono definire all'interno di un'architettura che consiste da:

- **Diagramma dell'architettura:** modello rappresentativo del sistema CEP. Occorre per mostrare il flusso di eventi che viene scambiato fra i processi e i loro connettori.

- **Specifica del comportamento:** definite dalle **regole**, codici molto importanti per il CEP, senza quelle appunto il sistema non sà come gestire il flusso di eventi.

- **Vincoli di progettazione:** sono vincoli sulle regole e gli eventi del processo oltre a definire in che modo avviene la comunicazione all'interno dei connettori.

Come scritto sopra, le regole sono forse la parte più importante del sistema CEP. Appunto svolgono il ruolo di descrivere i processi e i connettori. Si possono avere processi che appunto aggregano, filtrano, eseguono pattern-matching di eventi a basso livello etc. Per descrivere tali regole esistono differenti linguaggi che cambiano a seconda del prodotto per il CEP utilizzato, di cui molti sono SQL-like. Ciononostante, il fulcro del CEP è proprio il pattern-matching, il quale non solo permette di filtrare solo gli eventi interessanti (utile soprattutto nei sistemi ad alta velocità e volume di dato), ma permette l'aggregazione negli eventi complessi. Questa parte non si differenzia tanto dalle espressioni regolari, inclusa la parte di cattura dell'evento per essere riutilizzato per scopi futuri.

Il CEP di per sè non è un'applicazione a se stante, in commercio esiste una varietà di prodotti che permettono di implementarlo a seconda dell'uso che se ne vuole fare. Le applicazioni più studiate che utilizzano il CEP sono:

- **Event Stream Processing:** si focalizza principalmente sull'elaborazione dei flussi di eventi a basso livello. Dato che il CEP è la specializzazione del Data Stream Processing, questo è il caso a cui ci si avvicina di più, andando ad elaborare dati più omogenei, con un contesto ben più limitato. Quindi una applicazione ESP si applica solo a offrire calcoli statistici e matematici sull'evento ingestito piuttosto che operare con pattern-matching, aggregazione e giungere a rappresentare una gerarchia di eventi.

- **Event Driven Architecture:** il CEP può far parte di una EDA, la quale è una architettura che si permette generalmente di produrre, rilevare e consumare e di reagire agli eventi. Il CEP ha obiettivi più specifici e semplici, dato che vuole solo filtrare, combinare e aggregare eventi per formarne uno più complesso che sarà poi consumato da altri organi interessati, che possono essere altri motori per il CEP.
- **Service Oriented Architecture:** alcuni autori suggeriscono di utilizzare il CEP come una Web API di tipo RESTful, per appunto comunicare in tempo reale con altre Web Application e fornire loro informazioni sui risultati ottenuti dal processing.

Utilizzato in queste architetture, il CEP permette di reagire a problemi e possibili opportunità in tempo reale, dopo averli infatti notificati ad altri organi del sistema. Questi poi possono gestirli ed effettuare la corretta reazione ad essi. In alternativa ciò che si ottiene è solo storia: il problema o il vantaggio di prendere una certa decisione svaniscono e sono solo individuabili attraverso i dati storici salvati in storage. Si rivela quindi uno strumento molto potente e soprattutto flessibile, dato che può essere utilizzato in molti contesti, permettendo non solo di monitorare l'ambiente nel quale viene utilizzato, ma anche di automatizzarlo e di renderlo robusto davanti ad eventi inaspettati. Uno dei maggiori aspetti che ha permesso l'uso e la vendita di molti prodotti CEP è la facile integrazione nel **Business Process Management**, ovvero tutte quelle attività e gli asset che permettono di interfacciare il mondo informatico a quello gestionale delle aziende. Appunto uno degli strumenti di supporto al BPM è il Business Activity Monitoring (BAM), che permette il monitoraggio in tempo reale dei processi aziendali ed individuare dei possibili colli di bottiglia o problemi nelle fasi osservabili dei processi. Dato appunto che il BAM si occupa soprattutto di collezionare e processare dati per poi effettuarne una visualizzazione, il CEP può essere di grande aiuto, dato che può identificare situazioni complesse che

possono accadere all'interno di una azienda per poi notificarle al BAM per effettuare un report efficace.

1.4 Data Mining

Per descrivere l'argomento trattato in questa sezione è stata utilizzata la fonte [10]. Essenzialmente il Data Mining è quel processo iterativo il quale obiettivo è quello di raggiungere una scoperta e identificare le relazioni nel dataset o nel flusso di dati preso in esame attraverso metodi manuali od automatici. É la materia regina della Big Data Analysis ed è la scienza più efficace quando non si hanno informazioni a priori sul contesto da studiare, andando a bilanciare la conoscenza degli esperti dell'ambiente del dato e la potenza di calcolo degli elaboratori. Come detto prima, l'analisi del dato permette di effettuare due tipi di attività, ovvero l'analisi predittiva e descrittiva. Effettivamente, non è giusto pensare che una teoria sia meglio dell'altra, anzi, i risultati delle due attività sono complementari al raggiungimento dello scopo. Dove l'analisi descrittiva si ferma a trovare i pattern e nuove informazioni che l'esperto può trarre per effettuare degli studi, la predittiva invece permette di produrre un modello eseguibile sotto forma di codice, utile alla predizione, estimazione e identificazione di un processo. Le principali tecniche del Data Mining sono:

- **Classificazione:** scoprire la funzione predittiva che permette di etichettare il dato ad una classe. Alcuni algoritmi sono la classificazione Bayesiana, quella statistica o la foresta casuale.

- **Regressione:** si tratta di trovare la funzione che associa una variabile dipendente con una o più indipendenti. Tale variabile dipendente è funzione, lineare o di più gradi polinomiali, delle indipendenti più il valore di errore. Il metodo più usato è quello dei minimi quadrati.

- **Clustering:** attività che permette di identificare una serie di categorie, o appunto cluster, che dividono il dataset.

Uno dei punti di forza del Data Mining è appunto la vasta varietà delle metodologie e tecniche che permettono di affrontare particolari casi, anche quelli straordinari, tecniche che poi sono sempre più di numero crescente data la grande comunità di studiosi ed entusiasti che vi lavora. Infatti storicamente questa scienza nasce solo in ambito informatico e statistico, fino appunto a diventare un campo di studio indipendente.

Difficile è invece trovare una definizione comune tra gli studiosi di cos'è il Data Mining. Essenzialmente in origine trae le sue radici dalla BDA classica, soprattutto per le tecniche che portano all'analisi descrittiva e diagniostica. Per lo stato dell'arte presente, si può definire questa scienza come unione di altre due, le scienze statistiche e il **Machine Learning**. Le scienze statistiche hanno origini fondate nella matematica, quindi è stato ovvio applicare anche al Data Mining un approccio teorico per poi essere implementato in pratica. Mentre il Machine Learning conferisce al Data Mining l'imprinting informatico, con un metodo più pratico rispetto alle scienze statistiche, quindi verificando prima le performance delle tecniche con un approccio mirato al trial-and-error, senza avere una prova formale dell'efficienza. Oltre alla caratteristica dell'approccio, le due materie conferiscono enfasi differenti ai modelli e gli algoritmi prodotti. La parte statistica infatti è più mirata alla modellizzazione del sistema, quindi ad ottenere una struttura o una approssimazione di tale struttura che possa portare a produrre gli stessi dati di quelli studiati. Invece la parte di Machine Learning si spinge sulla produzione degli algoritmi.

La modellizzazione nasce dalla **Teoria dei Controlli**, applicata principalmente ai processi industriali e ai sistemi ingegneristici. Si arriva appunto all'*identificazione del sistema* attraverso l'osservazione degli input e degli output prodotti di un sistema sconosciuto (approccio Black Box). Gli scopi di tale identificazione sono molteplici,

uno tra i più importanti è quello appunto di predizione del funzionamento di un sistema e di trovare le relazioni tra le variabili che lo descrivono. Senza conoscenze a priori del sistema da studiare, l'identificazione diventa difficile, andando ad associare modelli che già sono stati definiti a quello in esame. Il più dei processi industriali e dei sistemi ingegneristici è stato descritto attraverso un modello, una cosa più complicata riguarda a fornire un modello per altre applicazioni, come quelle del settore business, social etc. In questi casi le strutture sono assolutamente sconosciute, in quanto molte non hanno avuto leggi matematiche che potevano descrivere i fenomeni riguardanti, perciò diventa più difficile apporre a loro un certo modello matematico adeguato ottenibile con le tecniche di identificazione. Quindi nuove metodologie per il Data Mining vengono sviluppate per l'identificazione dei parametri, per proporre leggi matematiche che possano descrivere fenomeni non fisici non studiati fino ad ora.

Il Data Mining è definibile quindi come un processo, che permette di scovare modelli e descrizioni a partire da un dataset. Tale processo non può essere un applicazione di metodi di Machine Learning e tool statistici presi a caso, anzi, deve essere un processo ben pianificato e strutturato in modo tale da risultare utile e pienamente descrittivo del sistema preso in esame. Tale piano di estrazione delle informazioni, di solito segue una procedura sperimentale di cinque passi solitamente:

1. **Definizione del problema e formulazione delle ipotesi:** l'identificazione del modello è più efficiente se il contesto su cui l'applicazione studiata funziona è ben definito, perciò un'ottima conoscenza e esperienza degli esperti in tale contesto è necessaria per definire al meglio il problema su cui il Data Miner deve lavorare. Purtroppo, molti tendono a pianificare il lavoro partendo subito a studiare quale algoritmo utilizzare, piuttosto che studiare al meglio il contesto d'applicazione. Quindi in questa fase il modellista si sofferma nella formulazione di diverse ipotesi assieme agli esperti, con una collaborazione che poi continuerà per tutto il processo.

2. **Collezione dei dati:** in questa fase ci si sofferma su come il dato viene generato e poi come lo si può collezionare. In genere ci si può agganciare a due possibili approcci. Il primo è quello di *design of experiment*, dove l'esperto ha controllo sulla generazione dei dati, infatti si va ad influenzare il sistema in modo da poi studiare l'effetto scaturito in maniera isolate in una variabile di risposta. La seconda invece, prevede che non è possibile influenzare il sistema, tale approccio è definito come **observational study**, dove ci si limita ad osservare il dato senza conoscerne la causa generatrice, come la distribuzione dei valori del dato, la velocità di campionamento e di arrivo del dato, che di solito può essere casuale. É molto importante conoscere come la collezione possa influire sulla teorica distribuzione del dato, dato che tale conoscenza può essere molto utile per la modellizzazione e per l'estrazione di informazioni. É importante anche che i dati che vengono utilizzati per il testing e per l'estimazione provengano dalla stessa sorgente con distribuzione di campionamento sconosciuta, altrimenti il modello può non essere utile all'applicazione.

3. **Pre-elaborazione del dato:** di solito, con l'approccio di osservazione, il dato deriva da database e altri storage. Il data prepprocessing quindi include almeno queste due attività: *rilevazione e rimozione degli outlier*, gli outlier sono dati anomali per il contesto e incorrelati agli altri dati osservati, comunemente sono la rappresentazione di errori sensoristici o di trasmissione, oppure anche di valori naturalmente anormali. Tali dati sono dannosi per la costruzione del modello, perciò vengono rimossi oppure l'algoritmo di costruzione del modello e tale da essere robusto e insensibile a tali outlier. L'altra attività è quella di *uniformare* la distribuzione e le soglie dei valori, andando a scalare tali soglie e i valori stessi in modo tale che tutti le variabili possano avere lo stesso peso. Molte altre metodologie sono previste per il data preprocessing. La conoscenza dei dati a priori comunque è importante per gli algoritmi di preprocessing in

modo da avere una buon modello rappresentativo, dato che tali conoscenze comunque influenzano l'attività di conformazione del dato. Le attività di preelaborazione non sono indipendenti fra loro e neanche con le altre fasi del Data Mining, dato che ad ogni iterazione del processo di modellizzazione un nuovo dataset rielaborato può portare a nuove informazioni.

4. **Estimazione del modello:** in questa fase si selezione la metodologia che possa fornire il miglior modello che possa rappresentare il caso studiato.

5. **Interpretazione del modello e studio delle prestazioni:** si deve notare che la possibilità di interpretare un modello non dipende dalla sua accuratezza, difatti i modelli più semplici sono i più interpretabili, ma sono anche quelli meno accurati. Questo perché un modello costruito con il Data Mining, a volte deve essere interpretato per facilitare l'utenza umana ad attuare strategie. Oggigiorno,le tecniche possono creare modelli molto accurati, ma comunque molto complessi e a volte multidimensionali. Si possono sì ottenere molti risultati da questi, ma essenzialmente a un utente normale possono anche non essere utili o non può capirle per definire una strategia o una decisione. Risulta quindi necessario ottimizzare e rendere capibili questi risultati in modo da migliorare l'user experience.

Come detto prima, queste fasi non sono tutte indipendenti fra loro. Il processo di Data Mining prevede appunto un approccio iterativo che, osservando i risultati ottenuti da una certa fase, rielabori il dataset in modo da raggiungere l'obiettivo per risolvere il problema preposto.

1.4.1 L'apprendimento automatico induttivo per l'estimazione del modello: il Machine Learning

Dopo aver descritto l'importanza della modellizzazione del sistema per lo studio del Data Mining, occorre capire come è possibile estimare tale modello del sistema studiato per effettuare delle possibili previsioni. Andando a prendere il concetto di inferenza dalla filosofia classica, si può dire che un processo di apprendimento predittivo è composto da due fasi: **induzione**, cioè il processo di ottenere il modello dalle relazioni sconosciute che intercorrono fra i dati di training, vengono aggiunte anche le eventuali considerazioni degli esperti di contesto; **deduzione**, col modello e determinati dati di input, è possibile predire un output. Dato che queste due fasi sono sviluppate sopra un modello globale del sistema, a volte può risultare inutile dato che diverse applicazioni sono interessate solo alla risposta a partire da un dominio ristretto di input. Esistono quindi tecniche che permettono di effettuare un'estimazione locale per poter predire un solo output, tecniche del genere sono definite come **trasduttive**. Tali tecniche sono spesso utilizzate per sviluppare regole di associazione.

Il processo di apprendimento induttivo e di estimazione può essere formalizzato e implementato utilizzando metodi differenti di apprendimento, cioè gli algoritmi che estraggono una mappatura a posteriori tra gli input e gli output, questi ultimi conosciuti invece a priori traendoli dal dataset. Al termine di tale mappatura è possibile poi predire gli output futuri a partire dagli input conosciuti. La formalizzazione di tali metodi e la loro descrizione matematica vengono studiate dagli esperti in Intelligenza Artificiale e di Teoria dell'Apprendimento Statistico, che fusi insieme han dato origine al Machine Learning.

Il Machine Learning comprende appunto lo studio di tutti gli algoritmi usati per l'apprendimento induttivo. Tali algoritmi variano spesso, a causa del fatto che alcuni sono adatti per determinati obiettivi, a seconda del dataset di training disponibili,

alle loro strategie di apprendimento oppure per la rappresentazione del dato. Tutti però sono costruiti per esaminare dataset n-dimensionali usati per il training, in quanto occorre comunque soddisfare e trovare una generalizzazione d'utilizzo.

Uno schema generale di apprendimento comprende almeno tre componenti, quali:

- **Generatore di input:** tale generatore produce un vettore di variabili stocastiche, definito X, progettato per essere indipendente da qualsiasi distribuzione e tempo di campionamento. Si rientra quindi nella situazione di *observational study*. Nel caso invece di *designed experiment* gli input erano definiti a priori e con un tempo di campionamento deterministico, ottimizzato per il DoE (Design of Experiment). Il Machine Learning comunque prevede che la mappatura non sia eseguita con il controllo sugli input.

- **Sistema osservato:** il secondo componente è il sistema stesso preso in esame. Per tale sistema si ipotizza che per ogni X venga prodotto un output Y in base ad una probabilità condizionale $p(Y/X)$. Bisogna dire che tale ipotesi prevede che il sistema sia deterministico, quindi vale $Y = f(X)$, ciò non è vero però nella realtà, dato che quasi tutti i sistemi non producono valori di output random, però ci possono essere comunque degli input che è possibile osservare. Questi input saranno quindi definiti come random e rappresentati attraverso una funzione di distribuzione delle probabilità.

- **Learning Machine:** tale componente è il cuore del Machine Learning, che prevede di costruire la mappa delle dipendenze dopo che gli algoritmi che la compongono vengono istruiti con il dataset di training. Tali dipendenze sono formalizzabili attraverso funzioni che approssimano il comportamento del sistema a fronte di un certo margine di errore. Tutti i metodi di inductive-learning prevedono una conoscenza a priori della classe di funzioni approssimanti, dove nel caso generale sono del tipo $f(X, w), w \in W$ dove w sono i parametri della

funzione e W è l'insieme di tali parametri usato per indicizzare una particolare funzione. Tale set di funzioni che viene estratto con il generico algoritmo di Machine Learning viene definito *insieme delle ipotesi*. Dato che tali ipotesi approssimano e non rappresentano in tutto per tutto il comportamento del sistema esaminato, vengono utilizzati particolari criteri che permettono di trovare l'ipotesi che meglio descrive il sistema, come la **distanza euclidea**, **indici di Akaine** etc. Tali criteri possono essere utilizzati nei particolari algoritmi di **ottimizzazione del costo**, che vanno ad identificare quale ipotesi permette di predire al meglio l'output generato dal sistema iterando una particolare simulazione cambiandone i parametri ad ogni ciclo, in questo caso si spazia nel mondo delle euristiche di ottimizzazione per la ricerca del minimo globale.

I metodi induttivi vengono classificati principalmente in due categorie: metodi per l'**apprendimento supervisionato** e quelli per l'**apprendimento non supervisionato**. I primi, spesso sono metodi che rientrano nei campi della regressione o della classificazione, tendono a estrarre le funzioni sconosciute utilizzando coppie di input ed output conosciuti e prevedono l'esistenza di un altro componente che sarebbe "l'insegnante", cioè le particolari *funzioni di fitness* o le euristiche di ottimizzazione che permettono di trovare il modello più accurato del sistema, oltre a contenere la conoscenza del contesto sapendo a priori la mappatura degli input ed output. La Learning Machine setterà quindi i parametri del set di funzioni attraverso gli input e il **segnale di errore**, ovvero la differenza dell'output desiderato meno il risultato dell'apprendimento. Tutto ciò serve per ottenere feedback nel sistema e quindi permette di trovare il modello più accurato dopo una serie di iterazioni, andando a trovare il minimo locale nella superficie della funzione di costo. I metodi per l'apprendimento non supervisionato usano solo il vettore di input p conosciuto e non vi sarà alcuna definizione dell'output durante il procedimento di

apprendimento. Quindi viene eliminato il blocco dell'"insegnante" e prevede che la Learning Machine svolga la modellizzazione e la valuti senza alcun interesse verso l'output e l'errore. L'obiettivo di questi metodi è quello di estrarre la struttura dei dati di input e di misurare la qualità della loro rappresentazione. Tale misura di qualità permette di ottimizzare i parametri w del sistema di apprendimento. Appena il sistema viene preparato e settato nel miglior modo con questi parametri, si può ottenere una rappresentazione globale ed applicabile per tutto il dataset in entrata. Tale rappresentazione di solito è ottenibile con il clustering o con le reti neurali (ANN). Altrimenti si può ottenere una rappresentazione locale, tramite, per esempio, algoritmi che utilizzano le regole di associazione.

Capitolo 2

Industria 4.0 e introduzione al problema del progetto

Questo capitolo si addentrerà nel contesto di **Industria 4.0**, ambiente di applicazioni e soluzioni hardware e software per il quale il mondo dell'informatica ed automazione industriale si sta aggiornando. Verrà data importanza soprattutto a come le tecnologie presentate nel capitolo precedente vengono utilizzate per sviluppare progetti e casi d'uso che riguardano appunto il settore industriale. Verrà quindi data una definizione di questo fenomeno della digitalizzazione del processo industriale e anche come è interfacciata la parte gestionale con il mondo degli impianti per permettere all'azienda di essere competitiva grazie all'ulteriore grado di automatizzazione implementato al suo interno. Inoltre verrà concessa qualche parola su come il mercato e la politica considera questa corsa alla digitalizzazione e quali aiuti sta garantendo al mondo delle imprese per portare avanti questo progresso in un ottica di salvaguardia degli sprechi e di massimizzazione dei profitti. Come scritto nel capitolo precedente, un'ottima attività di notifica può rendere possibile la prevenzione a particolari tipi di problemi all'interno del processo aziendale, in ottica industriale però occorrerà tenere un occhio di riguardo anche alla gestione delle macchine e

degli impianti, occorrerà quindi definire soluzioni apposite.

Identificato il contesto, si discuterà poi del problema che mi è stato posto per essere risolto durante l'esperienza di tirocinio in **AEC Soluzioni S.R.L**, quale è stata la mia linea di pensiero per iniziare a svolgere il progetto di tesi e di come la mia soluzione può essere integrata nel contesto in cui ho lavorato.

2.1 Il fenomeno dell'Industria 4.0

Come dice la fonte [11], il concetto di Industria 4.0 nasce nel 2011 per dare il nome ad un'iniziativa del governo tedesco facente parte del *High-Tech Strategy 2020 Action Plan* pubblicato da Kagermann. A partire dalla Germania, altre nazioni hanno incominciato a proporre direttive e discutere sul tema, a partire dagli Stati Uniti e la Corea del Sud. L'idea era proprio quello di contestualizzare il nuovo fenomeno in una prossima rivoluzione industriale manifatturiera dove vengono introdotti i temi di **Industrial Internet of Things**, Big Data e di **Sistemi Cyber-Fisici**, o CPS.

Mentre i primi due sono concetti molto approfonditi in letteratura, ma che comunque sono le tecnologie che permettono di implementare i CPS. Gli ultimi sono proprio il punto chiave di questa neo-rivoluzione industriale. Per dare una definizione, i CPS sono quegli insiemi di metodi e tecnologie che permettono di avere un sistema autonomo, interconnesso e che possieda la facoltà di integrarsi facilmente con altri sistemi diversi e distanti fisicamente. Oltre a visualizzare i CPS come insieme di oggetti interconnessi fra loro, come sensori, attuatori, software gestionali etc. e che siano in grado di scambiare grosse moli di dati ad alta frequenza nella connessione di campo, subentra anche il concetto di **Digital Twin**, ovvero la rappresentazione dell'oggetto fisico, quale un processo, un prodotto o il sistema stesso, in un oggetto digitale dal quale si possa ottenere ulteriori informazioni. A questo proposito vengono definiti i tre punti chiave per avere una fabbrica che lavora in

ambito Industria 4.0:

- **Smart Product:** il concetto di prodotto come pezzo di lavorazione viene esteso a parte attiva del sistema. Infatti esso viene dotato di memoria sul quale dati operativi e requisiti vengono caricati direttamente come se fosse una ricetta, in gergo industriale si intende come l'insieme delle materie prime e le istruzioni operative che occorrono per ottenere un prodotto finito o un semilavorato. A tal punto è il prodotto stesso a richiedere le risorse necessarie e le operazioni di trasformazione in modo da arrivare al suo completamento.

- **Smart Machine:** integrate nel CPS, questi impianti assumeranno un'altra visione di Sistemi Cyber-Fisici per la Produzione (CPPS). Diventano quindi organismi decentralizzati e modulari all'interno dei CPS. Anche qua rientra il concetto di interconnessione, infatti i componenti di queste macchine avranno una loro intelligenza locale che comunicherà con tutti gli altri agenti all'interno del campo di produzione. Questo renderà le catene di montaggio flessibili e modulari. Occorre però sviluppare componenti plug-and-play che garantiscono una veloce riconfigurazione in caso di cambio del piano produttivo.

- **Augmented Operator:** si premette che il fine ultimo dell'Industria 4.0 non è quello di soppiantare l'operaio a favore di processi completamente automatizzati o robotizzati. Anzi, l'operaio è definibile come l'agente più flessibile ed il più adattabile all'interno della produzione. Difatti si tende a studiare quali sono le tecnologie che possono offrire una guida all'operaio all'uso di apparecchiature complesse o di fornirgli strumenti che lo possano aiutare nel monitoraggio del processo. Questo permette loro di esprimere il massimo potenziale e di gestire la complessità tecnica che queste nuove tecnologie possono portare all'interno della fabbrica. Oltre a questo, l'operaio diventa anch'esso un oggetto integrato nel sistema, dove le azioni e i movimenti all'interno dello

stabilimento saranno monitorati, in modo da fornire informazioni aggiuntive all'ottimizzazione dei processi industriali.

Seguendo questi tre punti principali, la fabbrica raggiunge lo stato di **Smart Factory** ed appunto si serve di tecnologie come IoT, Cloud Computing, Big Data e Robotica per stabilire una catena di produzione e di distribuzione connessa e data-driven. Oltre ad avere scopi come la produzione sostenibile od altro, il fine ultimo di una Smart Factory è il raggiungimento di massima flessibilità nell'automazione del processo che permette di soddisfare ordini da cliente di dimensione veramente piccola. La possibilità quindi di avere dati di un certo valore in tempo reale dal processo produttivo garantisce quindi la certezza di sostenere tutti gli ordini da cliente senza perdere tempo nella riconfigurazione delle linee di assemblaggio. Questo viene realizzato attraverso l'uso di algoritmi dinamici per l'elaborazione di processi ingegneristici e gestionali, come gli algoritmi di scheduling per il setup, manutenzione macchina o per la gestione del floor-shop delle operazioni negli impianti.

Come dice la fonte [12], nell'impresa di ottenere una Smart Factory a tutti gli effetti occorre focalizzare le ricerce e le nuove tecnologie verso otto campi in particolare che rappresentano le caratteristiche principale di un impresa del settore industriale:

1. **Architettura di riferimento e definizione degli standard:** questo permette di migliorare le partnership e la comunicazione con aziende della stessa area produttiva e tra gli stessi reparti di stabilimento. Oltre a permettere uno scambio di dati orizzontalmente tra i reparti degli stabilimenti, esistono standard che permettono di interfacciare i diversi componenti della piramide di software gestionale che governa l'azienda. Molte architetture puntano ad utilizzare il protocollo standard **OPC-UA** per implementare le API driver di interfaccia tra i software gestionali e per rendere possibile la comunicazione **Machine-to-Machine** (M2M) tra macchine. Si prendano in esempio gli standard EUROMAP, che permettono di definire su OPC-UA le strutture dati

rappresentanti gli oggetti chiave di un gestionale (ordine, prodotto etc.) e standardizzarle per la comunicazione nel settore della manifattura della plastica e della gomma.

2. **Gestione di sistemi complessi:** ovviamente si tratta di un'architettura complessa e modulare. Occorrono quindi strumenti e progettisti software preparati nel modellizzare il contesto della fabbrica e per rendere implementabili le soluzioni software appropriate.

3. **Infrastrutture per la connessione:** l'Industrial IoT necessità di un'infrastruttura di rete che permetta una comunicazione affidabile e veloce. Occorre quindi uno studio del mezzo fisico sul quale avviene la comunicazione e sui protocolli più appropriati per l'integrazione nel campo.

4. **Sicurezza:** con lo stesso tema ci si riferisce a due concetti differenti: il primo riguarda la **sicurezza informatica**, quindi tutte le soluzioni software, hardware e sociali che permettono di garantire le proprietà della sicurezza informatica nel sistema, come gli accessi autorizzati alle risorse e la protezione dei dati e della comunicazione. Si evita quindi di diffondere dati sensibili della produzione che possano permettere di risalire al know-how e ai brevetti aziendali tramite il reverse engineering. Con il secondo concetto di sicurezza ci si riferisce alla **sicurezza dell'operatore**, cioè i mezzi e le normative che permettono all'operaio di lavorare nel campo di produzione senza essere posto a rischi per la sua salute.

5. **Progetto e organizzazione del lavoro:** oltre a gestire e a riprogettare il campo macchine e tutta l'architettura software della Smart Factory, occorre

rivedere anche come sono strutturati i processi industriali e logistici. Ottimisticamente parlando, secondo il paradigma dell'Augmented Operator il lavoratore dovrà svolgere un lavoro di maggiore responsabilità, pur garantendogli flessibilità nel muoversi nell'ambiente automatizzato.

6. **Addestramento e crescita professionale continua:** come già espresso, più il sistema diventa complesso, più all'operaio vengono date responsabilità sul controllo degli impianti. Il suo know-how e esperienza nel settore è importante, ma occorre comunque aggiornarla in modo da rispecchiare i continui cambiamenti che l'ambiente di lavoro subisce.

7. **Regolamento e legislazione:** ovviamente si deve regolamentare quest'innovazione, introducendo nuove leggi che possono tutelare la sicurezza e la privacy dell'operatore e i margini della acquisita responsabilità. Si deve discutere sulla riservatezza dei dati prodotti e la certificazione delle tecnologie utilizzate per la Smart Factory. Dal punto di vista finanziario ed economico occorrerebbe anche aggiornare le materie di bilancio per quanto riguardano gli incentivi e gli iperammortamenti, oltre a discutere su come vengono regolamentate le immobilizzazioni per quanto riguarda l'acquisto, vendita e utilizzo delle soluzioni hardware e software.

8. **Consumo sostenibile delle risorse:** si studiano le materie che permettono l'aumento della produttività, ponendo sempre attenzione a come le materie prima e l'energia vengano consumate senza sprechi. Soluzioni di controllo per la Smart Factory vengono implementate in modo da ridurre gli sprechi di consumo, l'inquinamento atmosferico e acustico permettendo una produzione sostenibile a salvaguardia dell'ambiente.

Diverse aziende si sono date da fare per apportare le innovazioni tecnologiche

previste dall'iniziativa, ciononostante per molte rimane comunque una visione piuttosto ottimista. Tolte le piccole industrie artigianali che magari non hanno bisogno di lavorare nel contesto di Smart Factoring, molte sono ostacolate dai problemi che possono sorgere per motivi economici o di progetto. Per quanto riguarda la parte di progetto, prima di tutto sono necessari dei *modelli di riferimento*. Effettivamente c'è la necessità di utilizzare degli standard tecnologici già esistenti per costruire l'architettura del sistema di gestione e di definire dei metodi e dei connettori per allinearli all'interno di tale architettura. Occorre quindi una definizione di partenza per la modellistica di progetto e una struttura descrittiva per casi d'uso che riguardino il contesto di Smart Factory. Molto importante è anche il rapporto con gli stakeholder, perciò la standardizzazione dei modelli è fondamentale se si vuole sfruttare la rappresentazione delle informazioni di processo. Questi sono i problemi che derivano dalla ricerca sulla *visualizzazione* delle informazioni, e bisogna tenere conto che come lavoro di ricerca diventa piuttosto complesso, data l'eterogeneità dei bisogni degli stakeholder che la richiedono. Molto importante è anche la stesura dei *requisiti di processo*, la cosiddetta Requirement Engineering, che se non viene utilizzata in maniera corretta diventa la causa principale per il fallimento dello sviluppo del progetto di Smart Factory e CPS. Questo può portare anche a buchi di bilancio, funzionalità mancanti o la terminazione stessa del piano di innovazione. Oltretutto un approccio sbagliato è vedere il CPS come un sistema di sistemi, questo andrebbe a infliggere in maniera negativa sulla definizione di requisiti e la visione del CPS formato da elementi interdipendenti diventa molto difficile da realizzare. Questo perché i diversi sviluppatori provenienti dallo studio di materie differenti lavorerebbero in maniera indipendente l'un dall'altro e solo su una parte che compone il CPS (approccio Divide-et-Impera). Qualora si mettesse in piedi il sistema, sorgerebbero i primi problemi di interconnessione tra i diversi componenti e di non continuità del data-flow del processo, dato che ciascun sviluppatore ha svolto la RE da solo.

Includendo requisiti che quindi risultano instabili, non pienamente definiti o che riguardano solo la disciplina per il quale lo sviluppatore lavora non è identificabile nel contesto Industry 4.0, dove si ricerca uno sviluppo smart e condiviso. Quindi devono venire definiti gli strumenti e i metodi necessari alla RE per prevenire questi casi malevoli nello sviluppo dell'architettura.

Da un punto di vista economico, gli *investimenti* per iniziare i lavori di aggiornamento della fabbrica possono essere veramente onerosi, e potrebbero essere una barriera per l'accesso a queste tecnologie da parte di piccole-medie industrie. L'implementazione di tali tecnologie in una piccola-media industria richiede un significativo investimento ed una previsione del Return on Investment (indice economico che misura il rapporto fra i ricavi totali e le immobilizzazioni). Tale previsione è piuttosto complicata da effettuare in quanto occorrerebbe fidarsi delle misure a priori delle opportunità apportate dalla nuova tecnologia installata. Questo problema comunque potrebbe essere risolto eseguendo prove e benchmark sulla piattaforma manifatturiera in anticipo rispetto all'investimento vero e proprio. Occorrono poi studi sulla *revisione del Business Model* aziendale. Un BM tradizionale per il manifatturiero prevede la focalizzazione dell'obiettivo principale dell'azienda sulla trasformazione di materie in un prodotto più o meno costumizzato per l'esigenza del cliente e sui successivi ricavi della vendita dello stesso. Le macchine, le materie e il personale rappresenterebbero dei costi fissi nel manifatturiero, quindi l'unico punto di competitività si avrebbe nella catena di distribuzione. L'apertura del mercato a standard tecnologici e la scomparsa delle barriere dei commerci ha messo in crisi questi BM, perciò gli studiosi suggeriscono di estendere l'obiettivo aziendale sull'offerta ai clienti di prodotti in bundle con servizi adattati alle esigenze. Tale concetto si sposa appieno con i concetti di Smart Factoring e CPS, dove la flessibilità nel servire anche il più piccolo degli ordini verso un cliente deve essere soddisfatta. Oltre a questo il concetto di CPS è quello di creare connessioni con altri sistemi ed il loro contesto,

quindi la fornitura di un servizio duraturo verso un cliente permette di estendere il processo industriale e l'estrazione di informazioni grazie a questa collaborazione che si è venuta a formare.

Risulta interessante analizzare un sondaggio globale effettuato dalla PwC nel 2016 [13] in merito all'Industria 4.0 e ai benefit che la digitalizzazione del settore industriale può portare. Circa 2000 aziende provenienti da 26 paesi, che si sono aperte al mondo dell'Industria 4.0, sono state intervistate. Lo studio riporta che entro il 2020 circa 907 miliardi USD all'anno verranno investiti in nuove tecnologie e al cambiamento del BM e dell'organizzazione del personale e all'addestramento delle risorse umane, con un trend del 20% delle aziende che intende investire circa il 10% dei ricavi annui nella digitalizzazione. Alcune però sono ancora scettiche a riguardo e cercano di rinviare i lavori in attesa di una tecnologia che permetta in un colpo solo di arrivare allo stato di Smart Factory. Questa è una veduta molto sbagliata, dato che l'implementazione di nuove macchine e gestionali è la sfida minore, invece è il cambio del BM che porta a spendere risorse e tempo. Il ritardo negli investimenti poi può portare a minore competitività nel mercato, che sarà comandato da quelle aziende che già si sono portate avanti nel lavoro di digitalizzazione. Risulta anche che le diverse imprese hanno una visione ottimistica sul ROI, dove molte prevedono di avere un coefficiente superiore all'unità entro due anni dall'investimento. Giappone, Germania e Sud Corea sono le nazioni che più di tutte si sono portate all'avanzamento nell'Industria Digitale, interessate agli aspetti di guadagni in efficienza di processo, riduzione dei costi, migliori condizioni di lavoro e aspettative di qualità del prodotto alte. Gli Stati Uniti invece si concentrano più sull'aggiornamento dei BM per ottenere maggiori profitti rispetto a guadagni di efficienza complessiva. La nazione che è più toccata dalla rivoluzione digitale è la Cina, che per il 2020 si aspetta guadagni sia in termini di ricavi che di efficienza nella produzione, avendo però la sfida di rivoluzionare le proprie industrie che si focalizzano sull'uso massivo di processi

industriali focalizzati sulla manodopera. Ciononostante le aziende cinesi sono quelle che più si distinguono in termini di flessibilità e di apertura alla rivoluzione digitale, con un personale che già è qualificato per l'utilizzo delle nuove tecnologie. Considerando le tre macroregioni di Americhe, Europa-Medio Oriente-Africa (EMEA) e Asia-Oceania, circa il 36% delle aziende di ciascuna regione nel 2016 aveva già fatto passi avanti nella trasformazione e si prevede che questa percentuale avanzerà del 40%. Questa rivoluzione porterà a grandi benefit per quanto riguarda la riduzione dei costi e degli sprechi, dove si stima che circa 403 miliardi di USD all'anno verranno risparmiati fino al 2020 e circa 493 miliardi di USD all'anno saranno composti dai ricavi provenienti dalla digitalizzazione. I settori industriali che sono più interessati in tutti questi aspetti sono quelli dell'elettronica, dell'edilizia e del manifatturiero.

Per quanto riguarda la situazione italiana, nel 2016 il Ministero dello Sviluppo Economico della XVII Legislatura ha avviato il **Piano Nazionale Industria 4.0** [14]. Si tratta di un'iniziativa che ha richiesto la collaborazione di diversi dicasteri come quello dell'economia, del lavoro e dell'istruzione, oltre alla partecipazione dei centri di ricerca, di Confindustria, delle Camere del Lavoro e delle più importanti università d'ingegneria, tra le quali anche il Politecnico di Torino. Considerando che in Italia vi è una mancanza di top player di larga distribuzione nel settore industriale manifatturiero e informatico che potrebbero portare le altre imprese alla digitalizzazione, il Governo ha rilasciato comunque delle linee guida che cercano di portare le imprese ad attuare questa trasformazione, come orientarle verso l'utilizzo di strumenti esistenti ed a lavorare sulle tecnologie abilitanti dell'Industria 4.0. Soprattutto deve essere un piano che deve intervenire in maniera generale, senza presentare iniziative ad hoc per alcuni settori dell'industria. Inoltre il mondo italiano dell'industria manifatturiera è fortemente basato sulle piccole-medie industrie, quindi lo Stato deve operare per fornire loro dei piani di finanziamento che possono supportare gli investimenti alla digitalizzazione. L'introduzione quindi dei piani di

iperammortamento e superammortamento ha permesso di registrare nel 2017 un incremento di 10 miliardi di euro negli investimenti e si prevede un incremento di 11.3 miliardi nel triennio 2017-2020 per investimenti focalizzati all'Industria 4.0. Oltre a migliorare le infrastrutture di rete, con la previsione del 2020 di avere tutte le aziende italiane coperte dalla fibra a 30Mbps, ci si è tenuto in particolare a definire le direttrici per quanto riguardano le competenze in materia di Industria 4.0. Infatti si è investito sulla specializzazione di molti studenti universitari e di istituti tecnici sui temi della digitalizzazione e si procede a creare dei **Competence Center** nazionali, che permettono di coinvolgere i poli accademici e i player privati per attività di formazione e di R&D sui temi di Industria 4.0.

2.1.1 Tecnologie a supporto dell'Industria 4.0

L'Industria 4.0 ha portato molte aziende e sviluppatori di software industriale a immedesimarsi in materie che solo negli anni precedenti hanno avuto rilevanza solo accademica o scientifica. Analisi di Big Data, Cloud Computing, Internet of Things sono parole che in questo momento sono sulla bocca degli entusiasti industriali ed in questi ultimi anni le assunzioni di informatici e data scientist sono aumentate esponenzialmente.

In primis, si possono contestualizzare tutte queste materie per produrre tecnologie che permettano di realizzare la Smart Factory e cercando di incastrarle e connetterle fra loro si può arrivare a costruire un'architettura gerarchica di mezzi che rappresenta la topologia di rete delle applicazioni, come esposto nella fonte [15].

I livelli più bassi dell'architettura fanno parte dell'**Industrial Internet of Things**, di cui fan parte tutte le attrezzature, le macchine e le risorse integrate in un sistema che permetta percezione, interconnessione e integrazione del dato. Il primo livello è appunto il **physical resource layer** che include tutte le risorse utilizzate nell'intero ciclo di manifattura. Per rientrare nel contesto di Smart Factory, l'impianto o

unità per la manifattura, deve essere modulare, composta quindi da attrezzatura come manipolatori industriali o macchine utensili, e riconfigurabile in tempi brevi per adattarsi allo scheduling dinamico degli ordini. Quindi il controllo dell'impianto (*a.e.* PLC hardware o software, ROS per i robot o controlli per macchine CNC) deve permettere la riconfigurabilità anche on-the-fly e deve essere estensibile per le varie attrezzature e le funzioni manifatturiere. In questi casi la robotica ha dato un contributo piuttosto alto e ci sono esempi di applicazione nel quale si parlano di isole di robot cognitivi e auto-configurabili, dei quali i controlli vengono verticalmente integrati con il sistema MES (Manufacturing Execution Systems) dell'azienda, il quale controlla e schedula dinamicamente l'esecuzione degli ordini. La Smart Factory richiede poi un'acquisizione di dati intelligente da parte delle macchine per salire di livello nell'architettura. In questo caso entra in gioco l'IoT, che permette di mettere a disposizione le tecnologie utili a costruire reti per la comunicazione dei dati da parte delle macchine. Una soluzione particolarmente utilizzata nel campo industriale è la Wireless Sensor Network (WSN), dove i dati per il monitoraggio, la storicizzazione e il logging vengono presi dai sensori delle macchine collegati alla rete. Grazie all'acquisizione dei dati poi è possibile eseguire tutte le funzioni di scheduling automatico delle attività industriali come produzione e manutenzione. Le tecnologie IoT più utilizzate in questo livello sono la Radio Frequency Identification (RFID), 802.11g, ZigBee, WM-Bus e Bluetooth, le quali sono tutte standardizzate e ottimizzate per un basso consumo di banda e di energia e per ottenere una coordinazione nella rete senza che questa venga mai interrotta a causa di guasti tecnici. Oltretutto sono soluzioni veramente poco costose che utilizzano driver e device di trasmissione vendibili a basso prezzo. Ovviamente devono essere anche compatibili con i protocolli necessari per portare i dati su dispositivi SCADA (Supervisory Control and Data Acquisition) o PCS (Process Control System). Ciononostante occorre

uno studio accurato sulla topologia di rete da utilizzare con i device per l'acquisizione dei dati e nel caso di considerazioni sbagliate si può ottenere un collo di bottiglia per il processo di gestione intelligente.

Il livello successivo è il **network layer**, ove gioca un ruolo molto importante se si vuole un sistema che lavora in tempo reale e sia completamente affidabile e robusto ai guasti. In questo livello si posizionano tutte le tecnologie che permettono di effettuare la comunicazione tra le risorse e i livelli applicativi di gestione superiori. Tradizionalmente sono stati usati spesso i *bus di campo* (si vedano Foundation Fieldbus, Profibus, Industrial Ethernet) che tramite la loro standardizzazione già hanno permesso alle aziende di soddisfare i requisiti di reti compatibili, aperte ed universali. L'innovazione in questo campo sono le **Industrial Wireless Sensor Network** e rappresentano l'estensioni delle tecnologie wireless per applicazioni general-purpose verso il mondo applicativo industriale. Esistono già versioni standardizzate di protocolli come HART, WIA-AP e ISA100.11a e questi standard permettono di avere bassa latenza, alta affidabilità e alta sincronizzazione per permettere un controllo altamente accurato e privo di pericoli in eventuali sovraelongazione dei segnali. Inoltre permettono di garantire alta densità di accessi, utili ai servizi interattivi per le applicazioni, e per l'acquisizione dei dati devono lavorare in regime di basso consumo. Tutto ciò deve funzionare in un ambiente che, come i reparti della produzione, presentano un alto numero di onde elettromagnetiche e di fattori di interferenza, molto dannose per delle tecnologie wireless. Questi problemi possono portare a problemi di disincronizzazione e calo della garanzia del Quality of Service, con potenziale perdita dei pacchetti di dato e tale problema potrebbe essere piuttosto fastidioso per delle applicazioni data-driven, oltre a essere dannose per la criticità dei processi delle macchine. Inoltre, avendo risorse computazionali limitate e necessità di avere bassa latenza, è impossibile implementare protocolli di sicurezza sulle reti IWSN, in quanto le operazioni di cifratura sono piuttosto onerose in termini di calcolo. Rientrano nelle

tecnologie per il network layer anche le applicazioni di **Edge Computing**. Come si può capire dal nome, l'Edge Computing viene processato su appositi elaboratori al confine della rete degli impianti e fornisce servizi intelligenti per la connessione agile e flessibile, elaborazione e pulizia dei flussi di dati in tempo reale e fornisce anche misure di sicurezza per la riservatezza del dato. Sono dei nodi piuttosto importanti e fungono da middleware anche per i livelli superiori e le logiche che vengono caricate al loro interno permettono di decentralizzare il carico di lavoro e permettere una comunicazione distribuita P2P. L'utilizzo degli Edge Computer è fondamentale per il monitoraggio dinamico e il controllo del processo manifatturiero, garantendo sicurezza, autonomia e robustezza nel sistema. Ovviamente poi non si può parlare di interconnessione senza protocollo di comunicazione. In questo caso si sceglie spesso l'utilizzo del protocollo Machine-to-Machine **OPC-UA**, che grazie alla sua caratteristica di essere cross-platform permette di fornire comunicazione tra le attrezzature e i dispositivi costruendo una SOA. Oltre alla comunicazione, OPC-UA permette di integrare dati di semantica strettamente industriale in un'architettura di rete e di ottimizzare quest'ultima al processo manifatturiero.

Il prossimo livello rappresenta il ponte tra quello che è stata la parte di IIoT e quella che sarà la parte di **Internet of Service**, ovvero quella parte di architettura che permette di virtualizzare le risorse dei processi industriali e che tramite sistemi e prodotti che richiedono l'interazione umana, permettono di costruire il processo industriale a partire dagli ordini logistici inseriti. Questo livello, che è anche il nodo centrale per l'integrazione sia verticale che orizzontale del processo manifatturiero, è il **Cloud Application Layer** ed è anche il cuore dove risiede il CPS. In questo livello risiedono tutte le logiche applicative di back-end che fanno da cuore pulsante per tutto il sistema di Cloud Manufacturing. É definito livello Cloud in quanto è una soluzione recente fare riferimento al **Cloud Computing** per instanziare le logiche di back-end e lo storage, senza dover avere in azienda le risorse hardware per

instanziare il nodo centrale di logica. Il Cloud Computing è appunto un sistema di fornitura di risorse di elaborazione, storage e anche microservizi on-demand tramite una piattaforma cloud con tariffe a consumo. Le piattaforme leader al momento sono Amazon Web Services, Google Cloud Platform e Microsoft Azure. Questo permette comunque un accesso rapido alle risorse IT, a basso costo e senza dover investire su infrastrutture hardware o dedicarvi troppo tempo alla loro gestione. Permette anche di abbandonare in parte la fase di testing delle applicazioni sulla quantità di risorsa di cui il software ha bisogno, dato che ci pensa la piattaforma Cloud a fornire instanze di macchine con l'adeguata quantità. É necessario tenere conto però che la piattaforma risiede in remoto, bisogna quindi fare attenzione ai casi in cui la connessione ad Internet possa mancare e a instaurare una connessione sicura con la piattaforma, ma comunque si ottengono buoni risultati per quanto riguarda la rete interna dato che è prevista meno occupazione di banda. Per la sicurezza poi sono previste delle soluzioni già offerte a seconda del tipo di applicazione. In questo livello risiedono anche tutte le logiche data-driven e per la gestione degli storage. Per quanto riguarda la privacy e la tutela del dato, sono fornite dalla piattaforma Cloud tutti i mezzi per garantirla e viene offerta anche un'automatizzazione per il backup e il ripristino dei dati. Inoltre le logiche data-driven sono proprio quelle che si occupano di BDA e Data Mining ed utilizzando sistemi per il Batch Processing possono fornire un'aiuto nel decision-making o per effettuare uno studio sull'ottimizzazione del processo. A livello back-end inoltre occorre sviluppare applicazioni che siano aderenti al contesto e per questo motivo sono nate le applicazioni **ontology-based**. L'ontologia appunto è una descrizione semantica del contesto di riferimento ed è piuttosto utile per quanto riguarda la condivisione di informazioni ed il ragionamento. Appunto un'applicazione basata sulle ontologie permette di ottenere interoperabilità nel sistema manifatturiero attraverso le diverse applicazioni server-side.

L'ultimo livello, quello più in superficie, è il **terminal layer**, dove sono riposte

tutte le tecnologie atte all'interazione umana con il sistema. In questo livello si definiscono le tecnologie per il monitoraggio degli impianti e per la visualizzazione delle informazioni ottenute dalle analisi. In questo frangente ottengono importanza i sistemi a code di messaggi o di distribuzione delle notifiche, dove i subscriber a tali servizi possono ottenere in tempo reale importanti notifiche, come allarmi o avanzamenti nel processo industriale. Tecnologie mobile per la visualizzazione stanno prendendo sempre più piede, per permettere di ottenere una visione dello stato completo del reparto degli impianti anche lontano dalla fabbrica. In questo livello viene sviluppata la parte di front-end nel sistema, come i client dei software gestionali.

Tra le altre tecnologie che non sono state menzionate in questo paragrafo, ma che comunque stanno prendendo piede all'interno del settore manifatturiero, sono la stampa 3D, la realtà aumentata e la robotica mobile, come AMV, per l'aiuto nella logistica di magazzino e in produzione.

2.2 Progettazione di una Control Room in ottica Industria 4.0, introduzione all'esperienza di tirocinio

In questa sezione della tesi e nei successivi capitoli si discuterà a proposito dell'esperienza svolta per risolvere il problema proposto dal tema di tirocinio. Il problema appunto prevedeva di fornire all'azienda i mezzi e le soluzioni per costruire un sistema che aggiornasse la situazione delle Control Room tutt'ora in funzione nella piccola-media industria manifatturiera. Per Control Room industriale si intende il centro operativo di controllo e monitoraggio degli impianti e delle macchine della fabbrica. Sono fornite quindi di SCADA e di pannelli operatori per impartire gli

ordini necessari al processo per proseguire e controllare ed essere segnalati riguardo ad alcune anomalie del sistema. Il capo reparto quindi può facilmente visualizzare lo stato attuale delle macchine o consultare gli storici della produzione per costruire report da far risalire ai piani gestionali della produzione. In un ottica tradizionale però non c'è alcuna automatizzazione e gestione intelligente delle risorse di produzione, difatti il capo reparto può venire informato degli avanzamenti o di eventuali problemi, ma deve poi arrangiarsi con l'esperienza acquisita con il lavoro per indirizzare il reparto verso gli obiettivi della produzione.

Quello che si vuole progettare è quindi una soluzione di back-end per aggregare diverse tecnologie in modo da gestire automaticamente il campo macchine a seconda dei dati che vengono acquisiti da queste ultime. Si può ipotizzare che questa sia un'applicazione del cloud application layer e permetta di indirizzare i risultati delle analisi a degli utilizzatori finali ai gradi superiori o altrimenti fossero rimandati indietro alle macchine, così da ottenere un anello di feedback. Prima di tutto occorre identificare il mezzo con il quale i dati vengono acquisiti e si è ipotizzato di interfacciare la logica con il reparto macchine grazie ad un gestore di messaggi publish/subscribe, ipotizzando che possiedano i mezzi necessari per essere riprogrammate secondo quel paradigma. Dopodiché il problema risiede in come gestire tutti i problemi che riguardano i Big Data. Appunto si parla di macchine che in un secondo possono inviare migliaia di valori dei sensori o eventi di processo ed il numero di dati cresce con l'aumentare del parco impianti. Inoltre occorre gestire in tempo reale questi dati e fornire una logica che tempestivamente possa rispondere alle eventuali eccitazioni che arrivano dal campo. Si è pensato quindi di procedere progettando un'architettura che implementasse soluzioni di Data Stream Processing e di tecnologie che rispondono ai bisogni delle criticità degli impianti industriali e che siano robuste ai guasti di comunicazione e tolleranti per grandi carichi di lavoro. Si è pensato di inserire framework di Complex Event Processing che permettono

di facilitare l'implementazione di regole che siano aderenti al contesto e che siano eccitabili per fornire in tempo reale la risposta all'evento scaturito. Oltre a questo occorre definire delle soluzioni che permettano di risolvere i problemi di storage dei dati sensoristici, si parla appunto di grandi moli di dati al secondo e che quindi occorre selezionare a priori i dati che sono interessanti per la storicizzazione e come strutturare lo storage che deve ospitarli. Oltre a questo occorre anche capire quali sono gli algoritmi di analisi e di Machine Learning adatti allo scopo per ottenere informazioni da questi dati a riposo oppure direttamente grazie alle logiche di Stream Processing. Il tutto effettuabile ed esaminando tecnologie e framework open-source.

2.2.1 Esperienza in AEC Soluzioni S.R.L

L'esperienza di tirocinio aziendale è iniziata il mese di Novembre 2017 nell'azienda sopracitata. AEC Soluzioni S.R.L. [16] è un agenzia di consulenza informatica nata nell'Aprile 2013 e fondata dai soci Antonio Tripodi e Davide Cecconi, ingegneri con esperienza multidecennale nella produzione di soluzioni software per l'industria. L'azienda è specializzata nell'implementazione e integrazione di software MES, in particolare è focalizzata sulla vendita della piattaforma gestionale **jpiano** a quelli che sono i maggiori player sia della piccola-media che della grande industria manifatturiera. Tale piattaforma è in continua evoluzione, alla quale i dipendenti, esperti del contesto manifatturiero, aggiungono sempre più nuove funzionalità richieste dalla clientela. Oltre ad utilizzare tecnologie del tutto consolidate per l'integrazione della piattaforma MES, AEC Soluzioni spende molto nella ricerca su tecnologie innovative, come l'implementazione di soluzioni IoT per l'acquisizione dei dati di processo. Grande punto di forza dell'azienda è il continuo interesse nelle iniziative proposte a livello nazionale nell'ottica dell'Industria 4.0, il quale le ha permesso di stringere collaborazioni con i più grandi stakeholders del settore. É anche una delle aziende aderenti al MESAP. primo Competence Center della Regione Piemonte per quanto

riguarda la meccatronica e l'automazione industriale.

L'esperienza quindi consisteva nell'attività di ricerca delle tecnologie Open Source che permettessero di implementare logiche per l'automazione del lavoro delle macchine e per lo storage dei Big Data di processo. Il lavoro è stato svolto in autonomia, aiutato dalla guida esperta dei tutori del tirocinio. Per svolgere l'attività mi sono state fornite le risorse informatiche aziendali e il dataset di un'azienda cliente, della quale per motivi di privacy è escluso discutere della sua attività del processo che ha generato i dati, riferendomi ad esso come lavorazione meccanica generica. Il lavoro ha permesso al sottoscritto di ampliare le conoscenze delle materie della Big Data Analysis e scoprire il mondo dell'Industria 4.0 e del Cloud Computing, partecipando anche a conferenze sulle materie in questione. L'accordo di tirocinio è poi sfociato in un contratto di collaborazione con l'azienda, permettendo di continuare l'esperienza lavorativa all'interno di un'azienda di ricerca e sviluppo. Il lavoro di tesi è stato comunque di aiuto per l'azienda stessa, dimostrandosi ancora di più interessata alle tecnologie per lo Stream Processing e CEP e il progetto che viene discusso nella tesi verrà continuato e portato avanti.

Capitolo 3

Progettazione di un sistema open-source per il data processing e il monitoring in tempo reale

Questo capitolo descrive il progetto del prototipo realizzato per la tesi. Il progetto appunto riguarda la costruzione di una logica per il Cloud Application Layer, dove appunto risiedono tutte le logiche di back-end atte al data processing e analysis. Verrà quindi dato lo schema di come le tecnologie e i framework usati comunicano tra loro per fornire il servizio finale. Queste tecnologie verranno poi descritte per quanto riguarda il loro potenziale funzionamento e verranno date le motivazioni al perché sono state utilizzate all'interno del progetto. Verranno poi percorse le mete significative per arrivare alla soluzione finale e quali sono stati gli esperimenti per ottenere lo schema finale. Dopodiché si daranno maggiori informazioni sul caso d'uso della rilevazione anomalie e un piccolo excursus sugli algoritmi studiati durante la progettazione. Verrà poi dato il metodo utilizzato nel prototipo per effettuare una rilevazione delle anomalie basate sul controllo statistico di processo. Dopo tutti questi passi verrà fornito appunto il prototipo di progetto finale e di come può

essere configurato per ottenere il servizio di rilevazione delle anomalie sul processo di lavorazione meccanica.

3.1 Introduzione al progetto

Lo schema generale del progetto si è ispirato a questa fonte [17]. Il progetto citato propone la configurazione di un'architettura per il Data Processing in tempo reale e alla storicizzazione dei dati per la produzione sostenibile nel settore manifatturiero. Questa architettura appunto propone di usare tecnologie Open Source e l'utilizzo di un algoritmo di Data Mining per il Machine Learning per ottenere un controllo di qualità del prodotto ottenuto da un processo industriale, in particolare si è studiata una pressa a iniezione per la plastica. La configurazione prevede l'utilizzo del framework per messaggi publish/subscribe **Apache Kafka** per l'acquisizione dei dati dalle macchine, **Apache Storm** come sistema per lo Stream Processing e **MongoDB** per la storicizzazione in un database NoSQL, presenta anche il supplemento di Apache Ambari per il monitoraggio degli allarmi sulle anomalie rilevate sulla qualità del prodotto. Il vantaggio di utilizzare questi prodotti risiede, oltre al fatto che sono gratis, flessibili e di facile configurazione, nella loro **scalabilità orizzontale**. Questo concetto prevede che tali tecnologie coordinano il carico di lavoro e viene poi sostenuto da più macchine server all'interno di uno stesso cluster. Tali elaboratori poi possono anche avere poche risorse e basse prestazioni, ma hanno un basso costo e questo permette di acquistarne in serie e costruire una *server farm* e si riesce comunque a implementare un'applicazione che potrebbe anche essere installata su un *large server*. La scalabilità orizzontale si distingue dalla *scalabilità verticale*, dove si vanno ad aggiungere risorse e componenti ad un'unica macchina che ospita le logiche per aumentarne la memoria e migliorare le prestazioni. Con la scalabilità orizzontale aumenta anche la *Fault Tolerance* e la disponibilità del servizio, in quanto se una

macchina server smettesse di funzionare, comunque un'altra potrebbe prenderne il posto. Ciononostante è più difficile da gestire e vi sono veramente pochi software che permettono la distribuzione e l'elaborazione del dato in parallelo, inoltre si tratta di alimentare più macchine e quindi si potrebbero avere maggiori consumi di energia. Comunque la scalabilità orizzontale è un approccio che definisce uno degli aspetti più importanti del Cloud Computing. Appunto, le varie piattaforme atte alla vendita dei servizi Cloud, garantiscono la possibilità di installare su varie istanze di macchine remote una propria applicazione e si applica un prezzo a consumo in base alle prestazioni delle macchine e per quanto le loro risorse vengono utilizzate. Oltre a queste tecnologie, gli autori di [17] utilizzano anche un algoritmo di Data Mining a Random Forest sul database NoSQL per effettuare il Machine Learning e aggiornare le logiche del Data Stream Processing in maniera offline.

Su suggerimento del relatore, all'inizio del progetto è stato consigliato di utilizzare il framework per il Complex Event Processing **Esper** di Espertech per la gestione della logica dell'applicazione. Infatti questa è venuta in aiuto per poter considerare una soluzione che permetta all'applicazione di esulare dal contesto e renderla più generica. Purtroppo per come si è svolto il lavoro non è stato possibile implementare tale soluzione, però è piuttosto interessante considerarla per effettuare un cambio di ontologia, basata sulla definizione delle regole, degli eventi e delle azioni in reazione a tali eventi. Tale soluzione ancora in fase di studio viene spiegata meglio nel prossimo capitolo della tesi nella sezione delle considerazioni future. Questo è importante, in quanto Storm può avere dei vincoli insidiosi per la definizione della sua logica, perciò Esper può venire in aiuto ad aumentare la sua flessibilità, oltre ad una velocità di aggiornamento migliore. Si è quindi scelto di produrre un'applicazione che dipende dal contesto, ma a scopo di dimostrazione utilizza comunque il CEP. Vi è stato anche uno studio su vari algoritmi di Data Mining offline per permettere di costruire in maniera automatica delle regole per il CEP, come autoCEP ad

esempio. Purtroppo il dataset fornito, seppur abbastanza interessante, mancava dei dati sul controllo qualità del prodotto, quindi è stato poco possibile costruire delle associazioni tra input ed output che il Machine Learning induttivo richiede. Sono state quindi utilizzate alcune delle tecniche del **controllo statistico di processo** per monitorare l'andamento della lavorazione ed utilizzarle per le regole del CEP. Utilizzare il Data Mining sullo storico e la generazione automatica delle regole per il CEP permette di aggiungere un altro grado di genericità all'applicazione pensata.

Inoltre occorre pensare che l'architettura dell'applicazione ideata non è applicabile a tutte le macchine, dipende appunto a come queste campionano il dato di sensore e quando ne permettono l'acquisizione. Si pensi che la maggior parte delle macchine appena rilevano un dato sensoristico, non inviano immediatamente un messaggio o un evento, ma sono dotate di datalogger o buffer circolari e quindi i dati vengono acquisiti in blocco. Diventa quasi un'applicazione di Batch Processing, perciò sono state fatti dei cambiamenti al dataset fornito, in modo tale da renderlo adattabile al real-time. A tal proposito si considera la macchina presa in esame per questo studio. Infatti la macchina invia un file CSV ogni 5 minuti ed esso contiene tutti i cicli di lavorazione registrati in quel periodo di tempo. Ci si troverebbe quindi a rilevare un anomalia molto tempo dopo rispetto a quando questa si è verificata, nel caso peggiore l'anomalia può essersi manifestata proprio nel primo ciclo di quei 5 minuti. Si è scelto quindi di ipotizzare che la scrittura del CSV venga bypassata e la macchina invia un ciclo al completamento di quest'ultimo.

3.2 Tecnologie utilizzate per il progetto

Qui di seguito viene riportata una breve spiegazione delle tecnologie utilizzate per lo svolgimento del progetto. Vengono quindi descritte le potenzialità e le opportunità che possono fornire oltre al loro funzionamento. Si dà quindi un'anticipazione a

come queste vengono usate nel progetto e che ruolo possono coprire all'interno di una Smart Factory.

3.2.1 Framework per messaggistica publisher/subscriber - Apache Kafka

Per questa sezione sono state utilizzate le fonti [18], [19] e [20]. Il publish/subscribe è un pattern architetturale dove si ha un disaccoppiamento strutturale fra il publisher, colui che presenta un messaggio a un broker o gestore, e lo subscriber, cioè colui che usa il messaggio presentato sul broker. Tali publisher e subscriber sono completamente indipendenti e ignari l'uno dell'altro, infatti è il framework per la messaggistica a terminare l'interazione fra i due. Per tali sistemi non è necessaria neanche la sincronizzazione temporale, anzi uno dei due peer può anche essere disconnesso. Ad esempio un subscriber se spento può successivamente usufruire del dato pubblicato qualora si riaccendesse. Una delle caratteristiche che distingue i diversi sistemi publisher/subscriber è la *logica di instradamento*, quindi i metodi e gli algoritmi per decidere dove e quando il messaggio verrà inviato al consumatore. I due principali approcci di instradamento sono il *topic-based*, dove il publisher etichetta un messaggio con un path di topic (*a.e.* /macchina1/piano1/) e questo può essere efficiente per operazioni di filtraggio e instradamento verso il consumatore iscritto al topic. In alcuni sistemi è presente un supporto a wildcard. Il secondo approccio è *content-based* dove il produttore non etichetta il suo messaggio, ma i campi e i metadati che lo compongono vengono usati per le operazioni di instradamento e filtraggio, il consumatore quindi si iscrive alla ricezione specificando i filtri con un linguaggio ad-hoc. Questi ultimi sistemi di solito prevedono costi computazionali maggiori, anche se permettono migliore flessibilità. Un altro aspetto che permette di valutare un broker è la garanzia della *Quality of Service*. Tale concetto prevede che un sistema publish/subscribe debba garantire queste proprietà:

- *Correttezza:* il sistema non deve garantire l'integrità della comunicazione, quindi niente perdità, duplicazione o disordine del messaggio.

- *Disponibilità:* il sistema deve minimizzare i tempi nel quale non fornisce il servizio.

- *Atomicità delle transazioni:* nel caso si avessero gruppi di messaggi che formano tali transazioni, o sono inviati tutti correttamente, oppure l'invio non deve essere mai avvenuto. Ad esempio un gruppo di messaggi che soddisfa una stessa semantica non deve risultare inconsistente alla ricezione.

- *Scalabilità:* è l'abilità con la quale questi sistemi permettono di evolversi, aggiungendo più attività da svolgere in parallelo.

- *Efficienza:* i sistemi devono garantire buone prestazioni per quanto riguarda la latenza e il throughput nella comunicazione.

Data la comodità di configurazione dei sistemi topic-based e la loro efficienza in trasmissione, questi sono largamente utilizzati nel settore industriale per l'acquisizione dei dati. Oltre a Kafka che è l'oggetto di studio, tra i più usati si ricordano RabbitMQ, Mosquitto e Moquette; questi ultimi due sfruttano il protocollo MQTT, famoso per la semplicità d'uso e per i risultati ottenuti nelle performance.

Apache Kafka nasce come software proprietario di Linkedin ed ottiene la licenza Apache nel 2012. É stato progettato per fornire alte prestazioni in throughput e per l'efficienza nel gestire l'organizzazione di più consumatori di uno stesso stream, i quali possono leggere dalla coda di messaggi a velocità differenti. Descrivere questo prodotto solo come sistema di messaggistica è piuttosto riduttivo, infatti questo utilizzo è un'astrazione del reale funzionamento. Kafka non utilizza un sistema vero e proprio di code di messaggi, piuttosto utilizza un sistema di log nella memoria non volatile della macchina per il salvataggio dei record trasmessi, considerando un

sistema di caching che permette buone prestazioni anche se sfrutta maggiormente trasferimenti in I/O. Questo influisce anche nella gestione del topic, infatti per ciascuno di questi si tengono dei log che vengono partizionati sulle diverse macchine che formano il cluster su cui Kafka funziona. A questi log vengono continuamente aggiunti i record, composti da chiave, valore e un timestamp che viene aggiunto dal broker. I record arrivano al server di Kafka e vi viene assegnato un id sequenziale chiamato *offset*, che lo identifica univocamente all'interno della partizione. I record salvati rimangono persistenti per quello che si dice *retention period*, il quale è configurabile per ciascun topic. Questo permette un consumo ritardato del record ed in caso di errori di trasmissione o guasti nei consumer, questi possono riprendere a leggere record a partire dall'ultimo letto anche dopo diverse settimane. La salvaguardia del record è resa anche disponibile dalla possibilità di replica dei log per la tolleranza ai guasti. Per quanto riguarda la lettura dei record da parte dei consumer, è compito loro controllare l'offset del record da leggere. L'offset per l'applicazione consumer è flessibile, quindi può saltare record a suo piacimento o riprendere da uno precedente per rimediare ad un errore. Questo permette anche di realizzare dei client veramente basilari senza preoccuparsi della concorrenza, dato che i record non vengono modificati quando sono scritti sui log e sono solo aggiunti.

Kafka fornisce quattro API principali per le applicazioni che prevedono di connettersi a questo servizio. Le prime due sono quelle riservate alle operazioni per il producer e il consumer, quindi permettono alle applicazioni che fanno uso di tali API di pubblicare lo stream dei record sui topic o di iscriversi a tali topic per andare a leggerli. Un'altra API recentemente aggiunta è quella per Kafka Streams, che permette di usare Kafka come se fosse un framework per lo Stream Processing. Un'applicazione che si interfaccia a Kafka con tale API infatti va a consumare uno stream pubblicato su uno o più topic, per poi processarlo e produrre uno stream

in uscita verso uno od altri topic. L'ultima API è riservata ai connettori per le applicazioni già esistenti, come ad esempio l'interfaccia con DBMS. Oltre a fare da intermediario per code di messaggi o un sistema di Stream Processing, la peristenza dei record permette a Kafka di diventare a tutti gli effetti anche un sistema di storage, applicando anche misure per la consistenza delle scritture e della commit sui log. Risulta quindi un sistema generalmente adatto a molti tipi di applicazione e casi d'uso, grazie a questa architettura e alle trasmissioni che effettua con bassa latenza grazie al suo protocollo proprietario, diventando quindi uno dei middleware più usati.

Forse uno dei punti deboli del sistema è che per essere utilizzato e per garantire la sua scalabilità occorre la necessaria interfaccia con **Apache Zookeeper**. Questo è un servizio per la gestione delle informazioni di configurazione e della sincronizzazione tra le applicazioni che vi si appoggiano. É anche il framework che permette la scalabilità di molti servizi Apache e consente la loro integrazione per formare una maxi applicazione composta da più microservizi, fornendo loro la possibilità di fare distribuzione del carico di lavoro su più istanze in parallelo. All'inizio Kafka era completamente dipendente da questa infrastruttura, dovendo addirittura delegarle il compito della gestione dei consumer. Con l'avanzare del tempo, lo sviluppo ha reso Kafka più indipendente, lasciando a Zookeeper il compito di essere solo il controllore e gestore del cluster e di fornire solo i servizi di replicazione e partizione dei log persistenti.

In letteratura vi sono veramente pochi casi che utilizzano Kafka per le potenzialità di storage o di Stream Processing, soprattutto per quanto riguarda l'Industria 4.0. Molto spesso viene nominato quando si parla di applicazioni IoT e in questi casi viene utilizzato come middleware per l'acquisizione del dato dalla rete di sensori. Kafka viene utilizzato per lo stesso compito anche nell'applicazione svolta per il progetto, poggiando su Zookeeper. Non è detto che si possano utilizzare anche le

altre funzionalità di questo sistema, ma per il momento non hanno grande rilevanza sugli scopi finali.

3.2.2 Sistemi per lo Stream Processing - Apache Storm

Per questa sezione sono state utilizzate le fonti [21] e [22]. Come riportato nel capitolo introduttivo, un sistema di Stream Processing è formato da più moduli collegati tra loro. Questi moduli ricevono nelle loro code di input lo stream di dati per processarli e producono dell'output spedendolo attraverso le loro code in output. Gli stream di dati possono essere contestualizzati in **tuple**, degli oggetti composti da attributi generati continuamente nel tempo. Un motore per lo Stream Processing crea quindi una rete logica con questi moduli connessi in uno schema che è riconducibile a un grafo aciclico. I nodi rappresentano gli elementi di elaborazione e i rami sono le code di input ed output per questi nodi di elaborazione. Le tuple vengono quindi scambiate all'interno di questo grafo, dove i moduli di elaborazione le consumano per poi produrne altre. I nodi eseguono le loro logiche di trasformazione tuple in maniera indipendente e possono trasmettere tali tuple solo attraverso una comunicazione di tipo push o pull. Generalmente l'elaborazione di tali nodi in parallelo non è mai sincronizzata, quindi occorre considerare anche i problemi di concorrenza tra questi. Anche in questo caso, i requisiti fondamentali sono molto simili ai sistemi di messaggistica, o almeno si concentrano molto sulle prestazioni, quindi la riduzione dei tempi di latenza, e sulla disponibilità e la resistenza ai guasti del sistema. Questi sono i punti fondamentali di cosa ci si aspetta da un sistema per lo Stream Processing:

- *Mobilità del dato:* si identifica come il dato si muove attraverso i moduli di elaborazione. Maggiore è la mobilità, minore sarà la latenza. Operazioni bloccanti da parte dei moduli di elaborazione possono portare a un deterioramento della mobilità.

- *Alta disponibilità e garanzie sul QoS:* dati i requisiti piuttosto influenti della bassa latenza, i tempi di ripristino del sistema devono essere piccoli e tali da permettere comunque efficacia nell'operazione.

- *Partizione dei dati:* i dati dovrebbero essere partizionati e gestiti in parallelo quando il volume e la velocità raggiungono livelli alti. Gli algoritmi di partizionamento influiscono sulla scalabilità e sul parallelismo dell'elaborazione.

- *Elaborazione deterministica o non deterministica:* le reti di elaborazione deterministiche producono lo stesso output qualora si utilizzasse sempre lo stesso input, indipendentemente da altri fattori, limitando però la flessibilità e il ventaglio di possibili potenzialità dello Stream Processing. Nel caso del "non determinismo" però il ripristino dai guasti può diventare più complicato.

- *Gestione delle imperfezioni sullo stream:* le tuple degli stream possono essere ritardate, arrivare non in sequenza o non soddisfano la proprietà di integrità. I sistemi per lo Stream Processing possono gestire tali imperfezioni fornendo un elaborazione ritardata al momento dell'immagazzinamento.

Per il progetto della tesi, è stato scelto il framework Apache Storm. Tale framework è scritto prevalentemente in Clojure, linguaggio di programmazione che incoraggia l'immutabilità per implementare facilmente applicazioni robuste, soprattutto per evitare problemi di concorrenza. Storicamente è stato sviluppato da Twitter, difatti in letteratura si trovano molte applicazioni che eseguono controlli semantici sul flusso di dati che questo social network produce. Viene rilasciato nel 2011 per il mondo Open Source con licenza Apache e la prima versione stabile è stata confermata a Settembre 2017.

Un'applicazione di Stream Processing costruita su Storm si definisce una **topologia**, che essenzialmente è il modello di grafo aciclico comune per i vari sistemi di Stream Processing. Prevede l'utilizzo di due tipi di nodi per l'elaborazione: gli

Spout, sono i nodi di entrata nella topologia che si interfacciano con il mondo esterno per acquisire e poi instradare i dati all'interno della rete, oppure tali dati vengono generati direttamente da questi; i **Bolt** che sono i nodi di processo, all'interno di questi vengono costruite le logiche di processing. I rami del grafo sono rappresentati dagli Stream, che sono i flussi di tuple che vengono scambiati tra gli Spout e i Bolt. La fama di Storm deriva appunto da queste e altre poche semplici primitive che fornisce. Questo piccolo insieme di funzionalità permette a Storm di essere un framework potenzialmente generico ed utilizzabile in un vasto insieme di casi d'uso, che sia appunto il Data Stream Processing o il controllo su un flusso di dati, ma può anche essere in grado di parallelizzare diverse attività, come ad esempio distribuire l'elaborazione di una query di ricerca piuttosto intensa. L'architettura che vi sta sotto alla piattaforma, permette ottime prestazioni in scalabilità su cluster, si pensi che inizialmente era in grado di processare un milione di messaggi al secondo su un cluster di 10 macchine. Dopodiché è piuttosto diffuso per essere agnostico nei confronti del linguaggio di programmazione. Esistono appunto diverse API per la maggior parte dei linguaggi di programmazione che permettono di definire le topologie e i loro componenti per qualsivoglia piattaforma. Oltre a queste esistono anche diverse API per i driver che permettono di integrare la maggior parte dei database e dei broker per il consumo dei messaggi agli Spout in entrata delle topologie. Esistono quindi, per esempio, Spout preconfigurati per il consumo dei record pubblicati su un topic Kafka. Esistono anche Bolt preconfigurati per l'uscita dal sistema, ad esempio che inviano un record a un topic Kafka, scrivono su una tabella di database o eseguono una richiesta verso un Web Server RESTful.

Per quanto riguarda la garanzia della Fault Tolerance, la tupla nelle code di uscita di uno Spout o di un Bolt persiste fino all'acknowledgment proveniente da chi deve ricevere tale tupla. Tale acknowledgment avviene alla conclusione dell'esecuzione di una logica sul nodo processore e nel caso su tale logica previene un eventuale errore,

allora gli Spout o i Bolt che vengono prima ripropongono la stessa tupla dopo un periodo di timeout. Questo meccanismo assieme all'approccio pull su cui si basa permette che qualsiasi tupla all'interno di una topologia venga processata almeno una volta.

I Bolt possono avere più task in parallelo che esegue la loro logica. Quando due Bolt sono connessi e quindi il flusso di tuple parte da uno e arriva all'altro, il modo con cui questi messaggi vengono inviati ai task che appartengono a tali Bolt dipende dalle regole di Grouping utilizzate. Le più semplice fra queste regole è la *Shuffle Grouping*, dove le tuple vengono distribuite equamente a tutti i task del Bolt successivo. Esiste anche il *Fields Grouping*, dove viene utilizzata la chiave di una tupla per l'instradamento verso il task che richiede tale chiave, oppure l'*All Grouping* che invia a tutti i task del Bolt ricevente tutti i messaggi che escono dalla coda del Bolt mittente.

L'architettura che sostiene la piattaforma è composta da tre servizi agenti. Occorre anche in questo caso l'ausilio di Zookeeper per la coordinazione del parallelismo e della scalabilità su cluster. Quindi vi è il nodo Nimbus, il quale è il server principale della piattaforma, dove viene caricato il codice della topologia e poi viene distribuito a tutti i nodi worker del cluster per l'esecuzione. Inoltre Nimbus tiene traccia dei log sugli avanzamenti dei worker che servono per il ripristino da un errore oppure reindirizzare il lavoro ad un altro worker quando quello che stava lavorando la logica si guasta. I nodi worker di un cluster Storm sono coordinati con Nimbus attraverso il demone Supervisor, Zookeeper quindi fa da passacarte fra i due servizi.

Un'altra feature di Storm, è l'astrazione Trident. Tale linguaggio permette di aumentare il livello della definizione di una topologia Storm, fornendo le primitve che poi vanno a trasformarsi nei componenti base della topologia. Le topologie scritte con Trident quindi utilizzano Spout e Bolt configurabili automaticamente che forniscono la possibilità di implementare join, aggregazioni, funzioni e filtri, eseguendo, come

esempio, lo stesso compito di astrazione che Apache Pig svolge per Apache Hadoop, cioè fornire la possibilità di scrivere SQL per il MapReduce, però in questo caso riguarda lo Stream Processing.

Nel contesto di Industria 4.0, Storm potrebbe essere realizzato per il processing della messaggistica tra i diversi agenti di un CPS ed essere coordinatore all'interno della Smart Factory per garantire un alto livello sia di integrazione orizzontale che verticale. La possibilità di definire nodi worker può permettere di integrare delle logiche di preelaborazione anche all'interno dei computer a bordo delle macchine nei reparti di produzione. L'utilizzo poi di Zookeeper permette di integrare facilmente la piattaforma Storm con il broker Kafka, utilizzando un solo coordinatore per eseguire il bilancio del carico di lavoro per entrambi i sistemi.

Viene utilizzato all'interno del progetto soprattutto per la facilità di implementazione delle logiche, utilizzando le interfacce fornite dal pacchetto della piattaforma e per la possibilità di integrazione con più o meno la maggioranza delle applicazioni. Quello che mi interessava maggiormente di Storm, è anche l'esistenza per questa piattaforma di un Bolt che ha già al suo interno la configurazione del motore CEP di Esper, ovvero l'**EsperBolt**.

3.2.3 Framework CEP - Esper e query EPL

Per questa sezione sono state utilizzate le fonti [23] e [24]. É piuttosto complicato definire un modello generale per i framework atti al Complex Event Processing, in quanto è una tecnologia che ancora rimane sfruttata dalla nicchia e in quanto non è largamente commercializzata. Ancora vi sono difficoltà a definire uno standard vero e proprio di tecnologia CEP e valutare quali sono le prestazioni influenti nel suo utilizzo. Si può dire però che esiste più o meno un modello generale della tecnica CEP e come questa viene proposta per i diversi prodotti distribuiti. Ovviamente si parte dall'oggetto principale che viene scambiato all'interno di una rete CEP, che è

il semplice evento prodotto da una sorgente nel mondo reale e più semplici eventi possono comporre un evento complesso, come riferito nel primo capitolo della tesi. Tali oggetti possono essere organizzati in flussi, qui si rientra nello studio dell'**Event Stream Processing**, il quale è anche l'approccio utilizzato per il progetto della tesi. Altrimenti si organizzano in bucket o cloud e allora vi si applica l'approccio del Complex Event Processing puro. Ciononostante il prodotto applichi o uno o l'altro approccio, dopo la raccolta degli eventi subentra un motore per l'event processing (*event processing engine*). Gli eventi vengono forniti a quest'ultimo dagli *Event Processing Agent* (EPA) che eseguono delle operazioni di preprocessing per il motore che elabora gli eventi, tali operazioni possono essere aggregazioni, filtraggi etc. Dopodiché il motore elabora gli eventi e se un pattern o una regola inserita nel motore viene soddisfatta, allora il motore inoltra la notifica. Tali regole e pattern vengono definite tramite un apposito *Event Query Language*, i quali cambiano di sintassi a seconda del prodotto utilizzato. La reazione a tale notifica può essere automatizzata se viene configurata a priori un *Event Condition Action*.

I prodotti CEP in fin dei conti si possono comparare con le tecnologie per lo Stream Processing, in quanto per nome possono presentare particolari similitudini, ma effettivamente sono piuttosto differenti. Ad esempio Storm è una piattaforma per container, infatti al suo interno viene caricata una topologia configurata a priori e rimane immutabile nel corso dell'esecuzione. Inoltre per lo Stream Processing operazioni come matching di pattern, filtraggio, aggregazione, gerarchia di eventi etc. non sono l'elemento sostanziale di tali piattaforme, mentre per il CEP lo sono. Quello che identifica in maniera centrale lo Stream Processing è invece il trasporto di eventi tra processi e interessati. Anche la Fault Tolerance è di maggiore importanza per lo Stream Processing dato che appunto si applica nel trasporto dei dati in maniera maggiore e quindi deve garantire l'integrità della comunicazione. Oltre a questo, il tipo di oggetto che viene scambiato nelle due tecnologie ha più probabilità di essere

non strutturato nelle piattaforme di Stream Processing, per il CEP invece si usa di più il dato strutturato. Il CEP poi è caratterizzato dall'utilizzo delle query, mentre l'altra tecnologia confrontata si basa di più sulle logiche di elaborazione. Questo perché il CEP è più mirato all'analisi e al matching di pattern rispetto allo Stream Processing, che è più calato sulla distribuzione dell'informazione di largo volume in maniera automatica.

Esper è uno dei prodotti open-source più utilizzati per il CEP ed è nato appunto per scrivere tali logiche in Java. EsperTech, l'azienda sviluppatrice, ha proposto anche un gemello scritto per il C# chiamato NEsper. Tra i vantaggi che offre vi è la sua caratteristica di poter essere inserito sia in un ambiente standalone che in un applicazione già costruita, con poche operazioni per l'integrazione. Si tratta infatti di realizzare alcune righe di codice per utilizzare l'API offerta dal modulo. La sua semplicità e il suo poco peso che occupa in un elaboratore, permette anche di processare sui dispositivi che emettono gli eventi, senza il bisogno di trasportare i dati su un server remoto dove vi è sita la logica centrale del sistema. Ciò non toglie che effettivamente occorre installare la logica di CEP su tutti i dispositivi e tale pratica può essere debilitante per la configurazione dell'intero caso d'uso. Comunque garantisce una gran flessibilità nell'implementazione della logica, permettendo di lavorare con tutti gli aspetti dei linguaggi ad oggetti oltre a supportare la tipizzazione dinamica. Un punto a sfavore per Esper è appunto la mancanza della possibilità di scalare il motore CEP nella versione Open Source, feature invece presente nella versione premium del prodotto. Ciononostante un utilizzo single-threaded del prodotto base permette di raggiungere un throughput variabile tra i diecimila e i duecentomila eventi al secondo. Inoltre il suo core é resistente ai problemi di concorrenza in un contesto multi-threaded.

Il sistema offre un'esagerazione di gradi di libertà per lo sviluppatore e tale

flessibilità si rispecchia soprattutto nell'**Event Processing Language**. Questo linguaggio permette di creare query sugli stream di eventi con una semantica uguale al SQL, anche se è più orientato agli oggetti piuttosto che alle tabelle. Le query vengono chiamate *statement* ed essenzialmente rappresentano le regole per il matching o pattern matching. Gli statement EPL influiscono molto sull'occupazione della memoria RAM del sistema, si ricorda poi che il motore Esper versione base risiede solo in memoria volatile senza alcuna persistenza degli eventi. Si pensi che la quantità di eventi che permane in memoria dipende appunto da come lo statement EPL viene formulata, ad esempio se si vuole fare un matching con gli ultimi eventi entrati all'interno di uno stream nell'ultima ora, allora tutti gli eventi ricevuti in quell'ora permangono nella memoria del sistema. Si ricorda inoltre che gli eventi sono per default dei POJO.

Viene riportato un esempio di statement EPL, dopodiché si descriverà il suo funzionamento e verrà spiegata la semantica di tale sintassi:

```
1 INSERT INTO ExitEvent
2 SELECT itemId, count(*) AS cnt
3 FROM OrderEvent.win:time(30min)
4 GROUP BY itemId
```

Listing 3.1: Statement EPL d'esempio

In questo facile esempio si contano le occorrenze di eventi OrderEvent nell'ultima mezz'ora. La INSERT INTO prevede che gli eventi prodotti con gli attributi della SELECT vengano inseriti nel flusso ExitEvent per una successiva rielaborazione. La FROM invece espone il concetto di "finestra". In SQL ci si ricondurrebbe a una vista, qui invece con le keyword *win:time(x)* si identifica proprio una finestra di tempo che scorre con l'arrivo di un evento all'interno, in questo caso, dello stream OrderEvent. Per definire una finestra si può anche inserire la keyword *last(y)*, dove si considera

una finestra formata solo gli ultimi y eventi arrivati all'interno dello stream. Per necessità del motore CEP di Esper, tutti gli eventi riferiti in tali finestre risiedono in memoria RAM fino a quando sono usciti da tali finestre oppure non vengono neanche salvati se nessun statement vi fa riferimento.

Come si può ben vedere, l'EPL rende molto facile scrivere una logica per il Complex Event Processing ed effettivamente si può trasferire la conoscenza di un sistemista di database relazionali alla gestione dei flussi di eventi. Se si vuole sfruttare la potenzialità completa di Esper, allora occorre fare un passo avanti e studiare i concetti portati dalle macro *pattern* e *match_recognize*. Entrambe permettono di eseguire il pattern matching per produrre un evento a seguito di un certo pattern di eventi che si è catturato all'interno di una finestra.

All'interno del progetto di rilevazione anomalie, ci si potrà domandare perché usare anche un modulo CEP all'interno della piattaforma di Stream Processing. Per il compito proposto, l'apporto che queste due tecnologie possono garantire è lo stesso. Però, come è stato scritto in precedenza, le logiche delle topologie Storm sono un po' rigide e nel caso si volessero cambiare tali logiche, si deve reinstallare sulla piattaforma la nuova topologia. Esper può invece venire in aiuto grazie alla sua flessibilità, dato che il core funziona tramite gli statement che sono in fin dei conti delle stringhe. Questo permette di decontestualizzare la logica nello Stream Processing e utilizzare le regole necessarie a seconda dell'ontologia desiderata.

3.2.4 Database NoSQL - MongoDB

Per questa sezione sono state utilizzate le fonti [25], [26] e [27]. Il concetto di database NoSQL, vede la sua nascita già dal 1998, dato per un progetto che prevedeva un database basato su file che ometteva l'uso di SQL. Dal 2009, con l'avvento del Web 2.0 e la nascita dei primi concetti di Big Data, il problema delle 3V di Laney ha messo a dura prova i database relazionali. Molte delle grandi aziende come Yahoo, Facebook

etc. hanno quindi sviluppato diversi prodotti di database non relazionali, che saranno poi riferiti come i NoSQL, per far fronte al problema dei Big Data e per realizzare storage distribuiti con attenzione alla parallelizzazione e alla sparsità geografica. Mentre i fondamentali dei database relazionali si racchiudono nel paradigma ACID, per i NoSQL è diverso e molti dei prodotti per lo storage non strutturato si trovano ad essere degli ibridi tra l'ACID e il paradigma **BASE**, che sta per: *Basically avaiable*, dove il punto centrale di un database è quello di essere sempre disponibile e resistente ai guasti; *Soft state*, si intende che la consistenza non è più rigida, ma lo stato del database può cambiare anche senza input; *Eventual consistency*, indica che il sistema diventa consistente non subito dopo una transazione, ma lo diventerà poi, quando magari possono accadere delle scritture.

Dato che BASE e ACID sono completamente opposti, è stato coniato un altro paradigma che potrebbe descrivere i database NoSQL meglio, questo è il **CAP**: *Consistency*, il dato rimane uguale in tutti i server di replicazione; *Availability*, il dato deve essere sempre accessibile; *Partition Tolerance:* il database deve lavorare fornendo lo stesso il servizio garantendo la Fault Tolerance. Il CAP comunque prevede che per un database non strutturato solo due tra i tre aspetti possono essere rispettati in un sistema distribuito.

Appunto i database NoSQL centralizzano molto le loro funzionalità per essere scalabili orizzontalmente e questo ha esaltato molto gli interessati. Inoltre la varietà dei prodotti per lo storage non strutturato permette di sceglierne uno fra i tanti disponibili che soddisfi al meglio le richieste dell'applicazione. Tra i tanti per questo progetto è stato scelto MongoDB.

MongoDB è un database Open Source orientato al **Documento**, dove ogni record viene memorizzato e incapsulato ad uno standard che può essere un XML, JSON o YAML etc. Per MongoDB il Documento è un oggetto *BSON*, un formato JSON-like ideato per MongoDB, composto da più coppie chiave-valore. Dato che è un

database schema-free, si possono salvare all'interno documenti BSON di diverso tipo e struttura. Il formato BSON è un'evoluzione del JSON, in quanto è progettato per essere efficiente sia in spazio consumato che per la velocità di ricerca. Tutti i documenti vengono salvati all'interno delle *Collection*, che sono dei maxi-documenti e ciascun record viene identificato univocamente dalla coppia chiave *"_id"* e dal valore ObjectId.

```
{
 _id: ObjectId("123456789101112131415 16"),
 "keyA" : "value_A",
 "keyB" : {
            keyB_a: "valueB_a",
            keyB_b: new Date('25/05/1993')
          }
}
```

Listing 3.2: Esempio di Documento BSON

Tra le feature più importanti si ha il pieno supporto all'indicizzazione per ogni attributo. MongoDB sfrutta molto questa possibilità per l'efficienza di esecuzione delle query, senza gli indici il sistema deve leggere qualsiasi documento della collezione per trovare i dati interrogati. Gli indici sono dei B-tree e queste strutture contengono gli insiemi dei campi ordinati per il loro valore. Tutte le Collezioni hanno un indice con il campo "_id" per default. Per quanto riguarda la replicazione, MongoDB gestisce diverse copie del database nel cluster di server, per fornire la Basically Availability. Tra i server se ne definisce uno leader chiamato Primary, sul quale vengono svolte tutte le operazioni di scrittura che poi verranno riflesse sui server Secondary. Le operazioni di lettura possono essere svolte su tutti i server. Nel caso il Primary server subisce un guasto, può esistere un Arbiter server che definisce

chi tra i Secondary prende il suo posto oppure si procede a votazione tra tutti i Secondary. Un'altra feature importante è l'*Auto Sharding* dove appunto il DBMS di MongoDB automaticamente esegue la scalabilità orizzontale del database su tutti i server del cluster. Questo processo viene svolto attraverso i Query Router che impongono lo instradamento della query verso la esecuzione su uno specifico server e i Config Server che contengono i metadati e i mapping delle informazioni sui shard, ovvero le partizioni di database. MongoDB implementa anche una sua versione del *Map/Reduce* per l'aggregazione dei documenti di una collezione per fornirne altri in output, che però non possono superare i 16MB di dimensione. GridFS invece è la specifica di storing dei file che superano i 16MB su MongoDB e permette di dividerli in chunk posti in speciali collezioni.

MongoDB è stato utilizzato nel progetto per la storicizzazione dei dati che arrivano dalla macchina, grazie ai suoi driver per Java ha reso molto facile la consultazione dei documenti e permette di essere integrabile con l'ambiente utilizzato in azienda. Le feature poi sono un punto di forza di questo prodotto, che l'ha reso uno fra i più utilizzati fra i major player nel settore industriale. Risulta essere anche uno dei database che garantisce un buon grado di compressione, utilizzando il sistema WiredTiger che è molto performante a livello multithread e permette anche la configurazione per la cifratura del Data at Rest, con buoni risultati poi per quanto riguardano velocità di scrittura e lettura del dato cifrato. Apache Cassandra e PostgreSQL sono stati considerati per l'integrazione nel sistema, ma il primo rende difficile l'utilizzo dei driver e occorre organizzare a priori le operazioni di lettura e scrittura attraverso query CQL (Cassandra Query Language) generiche che l'utente deve fornire. Il secondo è ancora sotto fase di studio, potrebbe essere utilizzato dato il fatto che nell'azienda dove si è svolto il tirocinio il DBMS utilizzato è PostgreSQL. Inoltre è stata sviluppata un'estensione per quest'ultimo che permette la storicizzazione con buone performance dei dati di timeserie, chiamata TimeScale.

3.3 Rilevazione anomalie

La rilevazione delle anomalie, come riportato dalla fonte [28] è una tecnica usata per monitorare un sistema e identificare dei pattern inusuali nel processo. Questa tecnica ha largo utilizzo in diversi casi d'uso, i più famosi sono quelli per la fraud e intrusion detection nel settore bancario, dove si vanno rilevare comportamenti anomali sul flusso di transazioni bancarie che potrebbero poi rilevarsi tentativi di frode, oppure quelli per il monitoraggio delle condizioni vitali nel settore medico, quindi rilevare ad esempio tumori in base ad una TAC o comportamenti anomali del battito cardiaco. Tali anomalie possono essere categorizzate come:

- **Singole anomalie:** definiti outliers, sono singole istanze di dati che si rilevano piuttosto lontani dal resto. Tale lontananza si può calcolare, in base alla dimensionalità e il tipo di dato, ad esempio con le distanze euclidee.
- **Anomalie di contesto:** comuni nelle timeseries, occorrono qualora si rilevi un pattern anomalo nelle misurazioni susseguenti oppure si registra un valore al di fuori delle soglie considerate.
- **Anomalie collettive:** un insieme di dati può dare una semantica alle misurazioni o da una successione di eventi che possono essere sconnessi si può risalire ad una anomalia.

Dal monitoraggio di una macchina si acquisisce una serie di valori da sensori nel tempo, quindi si vengono a formare diverse timeserie. Diventa quindi ineluttabile che le anomalie da rilevare in questo caso sono quelle singole o di contesto. Pensandoci però si potrebbero rilevare anche le anomalie collettive, infatti si può risalire ad una certa causa a partire, ad esempio, da una serie di allarmi che arrivano dalla macchina. Questo potrebbe essere fatto con il Complex Event Processing puro ad esempio. Ciononostante esistono diversi approcci per ottenere la rilevazione delle anomalie.

Vi sono degli algoritmi e tecniche del Machine Learning che lo permettono. Nella ricerca svolta sono state studiate due tecniche in particolare:

- **Regressione lineare o polinomiale:** tecnica induttiva dell'apprendimento, si costruisce un modello del processo a partire dai dati di input per relazionarli agli output, da queste associazioni si possono estrarre i parametri che permettono di ottenere l'equazione del modello. L'equazione che approssima di più dipende dall'errore tra output predetto e output ottenuto. L'equazione può essere lineare, quindi interpola con una retta tutti i punti campionati, oppure polinomiale, dove l'equazione interpolante può avere polinomi di grado superiore. A seconda che sia polinomiale o lineare, oppure con solo una variabile di input o multivariata, il tempo di calcolo per trovare tale equazione dipende dalla complessità del sistema e se l'output è impredicibile, l'errore diventa piuttosto alto. Ottenendo il modello dai nuovi campionamenti rilevati, è possibile verificare quanto tale modello si avvicina a quello ottenuto con i campionamenti di test derivanti dal processo in stato normale, nel caso tra i due modelli non vi sia una corrispondenza soddisfacente, allora si è rilevata un'anomalia.

- **K-means:** tecnica di data mining, propone di minimizzare la varianza tra cluster di punti di campionamento, identificati da un centroide. Eseguendo un'operazione iterativa, si vengono a creare K partizioni di campioni. Nella rilevazione anomalie si possono usare per identificare gli outlier dei cluster oppure fornire delle regole euristiche per allarmare quando il campione rilevato sta per uscire dal cluster ottenuto con i dati di processo in stato normale.

Per ottenere i risultati di Machine Learning è stato utilizzato il pacchetto Apache Spark sui dati del dataset dato in esame storicizzati su MongoDB. Purtroppo non si sono avuti dati a disposizione che corrispondessero ad un output, come il controllo

della qualità del prodotto della lavorazione. Ciò ha provocato parecchi problemi per ottenere tale soluzione ed infatti è ancora in fase di studio per il progetto. Ciononostante questi due algoritmi possono essere abbastanza efficaci per la rilevazione anomalie. Si è studiato Rule Learning offline che **autoCEP** svolge, algoritmo basato sul rilevamento di pattern nelle curve delle timeserie, per poi adattarlo a query EPL. Ciononostante tale algoritmo prevede l'utilizzo di eventi semplici, quindi rilevamento della singola misurazione di sensore e poi seguente elaborazione degli eventi ricevuti. Nel caso di questo progetto, con il dataset fornito risulta impossibile, in quanto venivano salvati tutti i campionamenti effettuati nel ciclo macchina, però viene fornito solo il timestamp di inizio ciclo senza il tempo di campionamento. Ottenere singoli eventi per la singola misurazione quindi risulta impossibile ed il tempo che intercorre tra i due cicli non è suddivisibile per il numero di campioni raccolti, dato che in mezzo a questi due tempi possono essere svolte altre operazioni, come homing dell'utensile, ripristino dell'operazione etc. Si è scelto di studiare algoritmi per il Rule Learning offline in quanto algoritmi per il Machine Learning online potrebbero portare a ritardi nell'elaborazione dei cicli di lavorazione, oltre ad avere un calo della generalità che la topologia studiata per la rilevazione delle anomalie può ottenere, portandola ad essere troppo vincolata al contesto dei dati in esame. Quindi si effettua uno studio sulle strategie di decisione, le si trasformano in regole per poi destinare le operazioni di decisione al modulo CEP all'interno della topologia.

Ciononostante per una prima fase del progetto si è scelto di contestualizzare la topologia ai dati senza effettuare l'operazione di Rule Learning offline, utilizzando altre tecniche per il controllo del processo.

3.3.1 Controllo statistico di processo

Per questo paragrafo, si usa il manuale [29]. Come detto nella sezione precedente, per fornire un esempio del primo stadio del progetto che verrà continuato in futuro,

sono state utilizzate delle tecniche per il controllo statistico del processo (SPC - **Statistical Process Control**). Sono tecniche semplici e basilari che permettono di rilevare se il processo esaminato sta avanzando correttamente utilizzando le proprietà statistiche più comuni di una distribuzione di campioni, come moda, media, mediana e deviazione standard. Sono tecniche utilizzate nel settore business e industriale, ma comunque abbastanza obsolete e si basano su regole ottenute più per l'esperienza degli esperti del settore e che sono piuttosto "greedy". Queste tecniche poi presentano delle debolezze, ad esempio per la media mobile, in caso di misurazioni che presentano rumore la media può risultare errata fino a quando il segnale non esce dalla finestra dei campionamenti.

A questo proposito sono state utilizzate le tecniche delle **carte di controllo X e S** con numeri di campionamenti variabili, dato che i cicli forniti nel dataset contengono un array di valori che può variare. In queste carte si calcolano media e deviazione standard ponderate su tutti i campionamenti presi dal passato per poi andare a definire i confini nel quale la media e la deviazione standard del campionamento presente devono stare per avere il controllo statistico del processo, come regola generale. Si applicano quindi le seguenti formule:

$$\overline{\overline{x}} = \frac{\sum_{i=1}^{m} n_i \overline{x}_i}{\sum_{i=1}^{m} n_i} \quad \overline{s} = \left[\frac{\sum_{i=1}^{m} (n_i - 1) s_i^2}{\sum_{i=1}^{m} n_i - m} \right]^{1/2}$$

Dove: $\overline{\overline{x}}$ e $\overline{s}$ sono la media ponderata e la deviazione standard ponderata su tutto il periodo dei campionamenti; n_i è il numero di valori del campionamento i-esimo; m è il numero di campionamenti effettuati. A partire quindi da $\overline{\overline{x}}$ e $\overline{s}$ è possibile ottenere le soglie di controllo massime e minime per la media e la deviazione calcolati con opportuni coefficienti descritti nelle tabelle del controllo statistico. Tali valori dipendono dal numero di misure per campionamento. Ricordo che questa tecnica non è il massimo per il controllo di anomalie, come scritto sopra, ma comunque potrebbe essere utilizzabile per presentare un primo prototipo della topologia in esame e per definire qualche regola di matching per il motore CEP.

3.4 Prototipo di sistema open-source per la rilevazione anomalie

In questa sezione viene presentato il prototipo del sistema per la rilevazione anomalie basata su controllo statistico e che utilizza il CEP. Si fornisce quindi lo schema che organizza le componenti del sistema, ricordando che per il momento è fortemente contestualizzato:

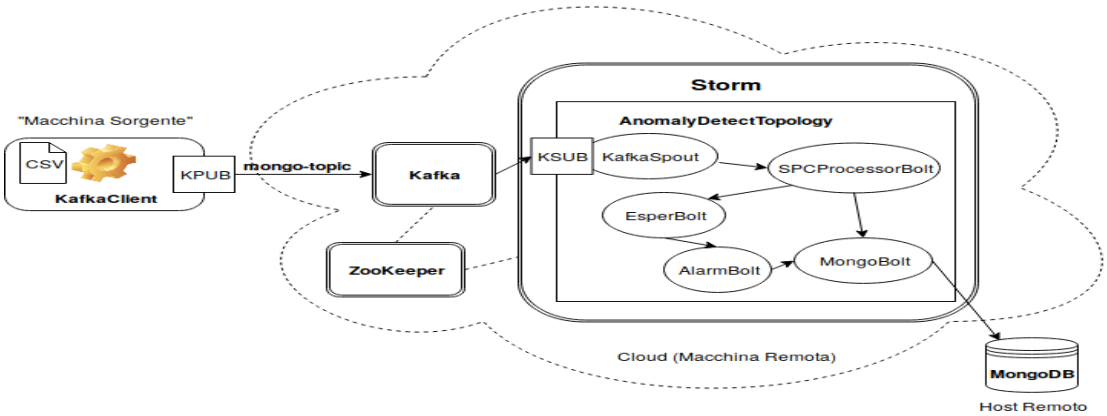

Figura 3.1: Schema principale del prototipo

Di seguito si descriveranno le componenti e le loro principali configurazioni. In primis occorre conoscere come il dataset viene fornito. Dopo che la macchina produce per una serie di cicli, i dati raccolti dai sensori vengono inseriti in blocco all'interno di un file CSV, al quale gli viene dato un nome che identifica la macchina, la parte della macchina che ha prodotto i dati e il timestamp (del tipo MMX_TX_YYYY_MM_DD_HH_mm_ss). Dato che il CSV viene rilasciato dopo che i cicli macchina sono già finiti, non è facile mandarli alla logica su Storm per eseguire il parsing e ottenere informazioni e controllo in tempo reale, questo diventerebbe utile solo per il Batch Processing. Perciò ci si è presi la libertà di modificare il problema e ipotizzare di avere un tipo di macchina sorgente che inviasse le

misurazioni del ciclo non appena questo fosse compiuto. Il ciclo all'interno del CSV è composto da questi campi (i nomi fra parentesi saranno quelli usati poi all'interno del sistema):

- Nome della macchina ("MACHINE")
- Nome della parte della macchina che produce i dati ("IDPART")
- Timestamp del ciclo ("TIMESTAMP")
- Numero sequenziale del ciclo ("NUMCYCLE")
- Numero dei valori di coppia registrati nel ciclo ("NUMVALUES")
- Temperatura della parte ("HEADTEMP")
- Temperatura dell'olio motore che raffredda la parte ("OILTEMP")
- Array dei valori percentuali della coppia del motore in rapporto alla coppia nominale ("TORQUE")

```
{
        "MACHINE" : "MMX",
        "IDPART" : "TX",
        "TIMESTAMP" : "YYYY-MM-XX HH:mm:ss"
        "NUMCYCLE" : 9999,
        "NUMVALUES" : 550,
        "HEADTEMP" : 26.0,
        "OILTEMP" : 25.0,
        "TORQUE" : [85.0 56.0 ....]
}
```

Listing 3.3: Rappresentazione in JSON del ciclo macchina

KafkaClient permette di fare l'astrazione e di inviare i dati al Cloud, garantendo condizioni per il testing migliore.

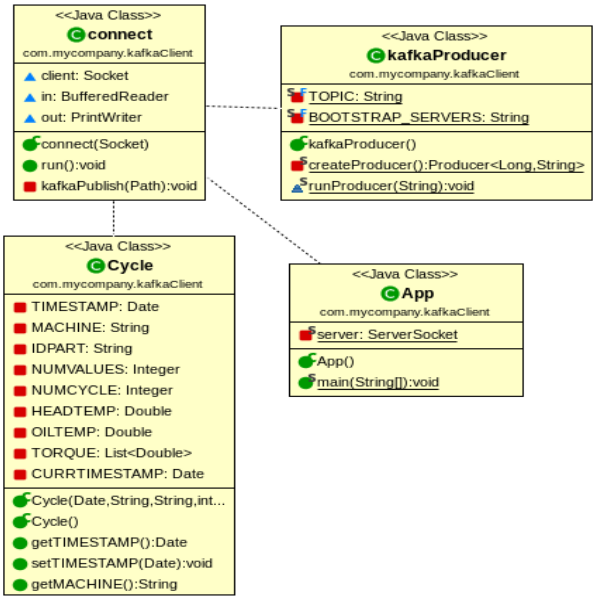

Figura 3.2: Diagramma UML delle classi di KafkaClient

Questo è un programma standalone scritto in Java come progetto Maven, che permette di caricare i file CSV, parsare i cicli ordinandoli per il timestamp e partizionandoli con il nome della parte della macchina, permettendo loro di inviare il ciclo in concorrenza. Occorre premettere che in tutto il sistema le parti e le macchine vengono trattate come indipendenti fra loro, quindi non vi sono dati in comune e non accade la comunicazione M2M. KafkaClient inoltre invia i cicli attendendo il tempo che intercorre tra uno e l'altro, permettendo di testare il sistema come se la macchina invia in tempo reale il ciclo appena compiuto. Dopo aver parsato i dati del ciclo, inizializza un istanza della classe *Cycle* settando le sue proprietà con i dati raccolti. Oltre ai dati scritti sopra, viene aggiunto anche un "CURRTIMESTAMP" contenente il timestamp che identifica l'inizio del ciclo di vita dell'oggetto da inviare, utile poi per eseguire i test di latenza prendendo anche in considerazione i millisecondi. Tale oggetto Cycle viene trasformato in JSON per diventare il payload

informativo da inviare tramite la classe KafkaProducer, che si occupa della pubblicazione del dato sul server Kafka in remoto. Per poter iniziare il publish, basta chiamare il metodo *runProducer()* con all'interno la stringa di JSON da inviare, quindi il metodo andrà a configurare un Producer (classe della libreria Kafka) con il BOOTSTRAP_SERVER e il TOPIC, ovverosia l'indirizzo IP e porta dove il broker Kafka rimane in listening e il topic dove viene pubblicato. Nel prototipo si utilizza il solo topic *mongo-topic*. Questo progetto Java utilizza le seguenti librerie come dipendenze Maven:

- openCSV 4.1
- Google GSON 2.2.4
- Apache Kafka 2.11-1.0.0

Ci si addentra poi all'interno del reame Cloud dove vengono configurati le componenti per lo Stream Processing. Prima di avviare il prototipo occorre far partire i componenti Apache. Quindi viene avviato prima il server Zookeeper, che si mette in ascolto sulla porta 2181 per default. Quindi vengono avviati il broker Kafka e vi si imposta anche i topic, nel caso di questo prototipo solo "mongo-topic". Infine si avvia la piattaforma Storm, quindi il server Nimbus e i vari Supervisor, che nel caso del prototipo sarà solo uno. Storm propone anche una User Interface accessibile su porta 8080, questa è una dashboard che permette di consultare lo stato della piattaforma e le metriche o gli errori che risultano all'interno della topologia. Tutte e tre le componenti sono in funzione su un'istanza remota di una macchina virtuale sulla piattaforma Aruba Cloud, le specifiche di tale istanza saranno riportate nei test. A questo punto, attraverso riga di comando, sarà possibile avviare la topologia **AnomalyDetectTopology** sulla piattaforma Storm.

La topologia è anch'essa un'applicazione Java che viene compilata con Apache Maven che poi viene eseguita sulla piattaforma. Nel *main()* della classe principale

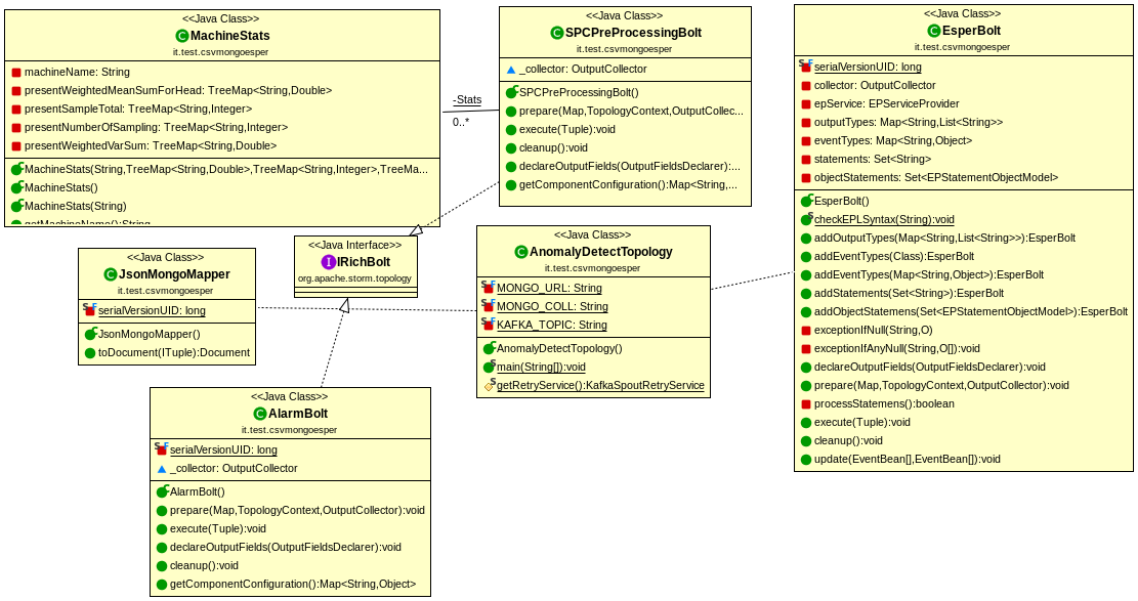

Figura 3.3: Diagramma UML delle classi di AnomalyDetectTopology

AnomalyDetectionTopology è possibile configurare gli Spout e i Bolt che vengono già forniti dalle dipendenze pronti per essere utilizzati (in questo caso il KafkaSpout, l'EsperBolt e il MongoBolt). Dopodiché viene configurata l'intera topologia definendo i connettori tra i diversi elementi processori.

Il **KafkaSpout** fornito dal pacchetto *Storm-Kafka* viene configurato per sottoscriversi al topic "mongo-topic" che sta sul broker Kafka nella macchina virtuale remota. In uscita emetterà una tupla con un singolo campo che conterrà tutto il payload informativo del record arrivato su Kafka, quindi la stringa JSON del ciclo di lavorazione.

Tale payload arriva quindi all'**SPCProcessorBolt** che implementa la classe *IRichBolt* del pacchetto Storm. *IRichBolt* è l'interfaccia sulla quale lo sviluppatore può implementare un suo Bolt, andando a fare l'override dei metodi già forniti dall'interfaccia. Tali metodi sono: *prepare()* che permette di inizializzare il collettore delle tuple del Bolt; *execute()*, che esegue la tupla non appena questa arrivi al Bolt; *declareOutputFields()*, dove si configurano i campi e i flussi delle tuple in uscita.

Per permettere di avere la Fault Tolerance tra i nodi Storm, occorre eseguire l'acknowledgment della tupla qualora questa viene processata completamente dal Bolt, è quindi necessario utilizzare il metodo *ack()* dell'oggetto Collector che identifica il collettore. In questo modo tutti i nodi precedenti sono informati dell'avvenuta transazione e quindi tolgono dalle loro code in uscita la tupla che è stata eseguita. SPCProcessorBolt quindi esegue i calcoli della media e deviazione standard ponderata con variabilità del numero di valori sull'array "TORQUE" e definisce i flussi in uscita: uno verrà inviato all'EsperBolt per la fase di controllo e l'altro al MongoBolt per la storicizzazione, inserendo un campo aggiuntivo al JSON con chiave "TYPE" e valore "MEASURE". All'interno della classe si definisce un istanza statica *Stats* di *MachineStats*, per contenere i valori utili per tutto il processo per elaborare la media e la deviazione standard ponderate.

EsperBolt è un Bolt preconfigurato fornito dalla repository GitHub https://github.com/miguelantonio/storm-esper-bolt, che permette appunto di avere un processore CEP sulla topologia Storm. Usando questo Bolt si devono definire i tipi degli eventi, gli statement EPL e i flussi in uscita. Lo statement dovrà utilizzare dei nomi particolari per definire le finestre che sono nient'altro che i nomi dei flussi di tuple in entrata al Bolt oltre a inserire i nomi dei flussi in uscita. Ad esempio:

```
1 INSERT INTO control
2 SELECT 'ALR:FUORI_DAI_RANGE_MEDIA_PER_MM8_T1' as alarm
3 FROM spcProcessing_values(head = 'T1')#length(1) as t1,
4 HAVING (t1.mean not between t1.lclx and t1.uclx)
```

Listing 3.4: Statement EPL d'esempio inserito su EsperBolt

L'"INSERT INTO" permette appunto di definire un flusso di tuple con i campi definiti nella "SELECT", mentre "*kafka2esper*" è il nome del flusso di tuple in entrata all'EsperBolt.

AlarmBolt prende invece il flusso "control" dall'EsperBolt, per mettere un campo ulteriore quale il tipo di messaggio, in questo caso "TYPE" avrà come valore "ALARM", che viene poi spedito al MongoBolt. Questo Bolt può essere potenzialmente usato per smistare l'allarme a seconda del tipo di azione che si deve intraprendere a seconda del tipo di allarme.

Tutte le tuple che arrivano al **MongoBolt** devono essere mappate per diventare documenti, quindi si configura il Bolt con la possibilità di usare la classe *JsonMongoMapper* che prende i campi delle tuple che gli arrivano e li inserisce tutti in un documento BSON. Per il calcolo della latenza di questo esperimento, appende anche un timestamp "TSEXIT" con il tempo di uscita dalla topologia. Le dipendenze usate per questo progetto sono:

- Apache Storm-Core 1.1.1 (deve avere scope "provided", dato che la piattaforma poi contiene tutti i pacchetti e i configuratori YAML già al suo interno)
- Apache Storm-MongoDB 1.1.1
- Apache Strom-KafkaClient 1.1.1
- Espertech Esper 7.0.0
- JSON 30/01/2018

Tutti i documenti poi inviati dal MongoBolt saranno raccolti all'interno della collezione "test" del database "db1" sullo storage su host remoto di MongoDB, tale spazio di storage viene fornito da MLab alla registrazione di un account.

Con questo si è definito un primo prototipo del progetto. Si nota che la topologia ancora permette solo il processing per questo tipo di dato fornito dal dataset, ma potrebbe essere utilizzata per effettuare un testing e vedere se una architettura complessa che utilizza varie tecnologie possa venire retta anche da elaboratori con a disposizioni poche risorse computazionali.

Capitolo 4

Risultati e considerazioni per il lavoro futuro

Nell'ultimo capitolo principale della tesi, verrà discusso il test effettuato sul prototipo e i risultati ottenuti, come occupazione delle risorse della macchina che ospita le tecnologie per lo Stream Processing, la latenza ottenuta a partire da KafkaClient fino ad uscire dalla topologia e il risultato del controllo statistico sulle carte X e S ponderate. Questo permette di interpretare la situazione attuale del progetto. A partire dal test si possono trarre idee per lo sviluppo di implementazioni future sul sistema. Quindi verrà fornito uno schema che rappresenta l'obiettivo finale del progetto che verrà continuato in azienda. Saranno fornite poi particolari attenzioni a quegli aspetti che non sono stati toccati dal prototipo, in modo da arrivare a produrre un sistema che sia sicuro, sempre disponibile e robusto agli errori di elaborazione. Soprattutto tale sistema dovrà essere non contestualizzato al dato e alla modalità con qui questo viene fornito dalla macchina, quindi occorra implementare diversi connettori per la comunicazione con il broker Kafka in entrata e la topologia dovrà essere la più generica e performante possibile, in modo da garantire il monitoring in tempo reale ed eventualmente permettere una retroazione sulla macchina da cui

sono stati acquisiti i dati.

4.1 Test del prototipo

Il test è stato realizzato utilizzando come dataset i 600 cicli macchina forniti dal cliente nei file CSV, che appunto sono stati tradotti e scompattati nei singoli cicli, inviati con un distacco tra uno e un altro a seconda del timestamp salvato su tale ciclo con KafkaClient. I cicli riguardano una sola macchina e sono stati salvati 150 cicli per ciascuna delle parti della macchina presa in esame. Tali cicli sono stati salvati in un periodo di tempo pari a 10 minuti, pari quindi al tempo di esecuzione del test. Per testare il controllo sul flusso dell'EsperBolt, sono state utilizzate delle semplici regole EPL che permettono di confrontare la media e la deviazione standard del ciclo con le soglie ottenute dalla tecnica del controllo statistico delle carte X e S. In tale calcolo, ciascuna parte della macchina in esame rimane indipendente dalle altre. La regola utilizzata è proprio quella fornita dal listing (3.4) adattata poi anche alla deviazione standard e alle diverse parti della macchina. Vengono poi salvati sullo storage MongoDB remoto i documenti di misura e di allarme, per l'esempio ne vengono forniti uno di un tipo e uno dell'altro:

```
{
 "_id": {"$oid": "5b4bc4e49fc077b0f99d92b6"},
 "ALARM": "ALR: FUORI DAI RANGE MEDIA PER MM8_T1",
 "TYPE": "ALARM",
 "IDPART": "T1",
 "NUMCYCLE": 14462,
 "TSEXIT": "2018-07-14 00:04:20.315"
}
```

Listing 4.1: Documento BSON dell'allarme memorizzato in MongoDB

```
{
 "_id": {"$oid": "5b4bc2709fc077b0f99d906a"},
 "MACHINE": "MM8",
 "IDPART": "T1",
 "OILTEMP": 27.6,
 "HEADTEMP": 51.7,
 "NUMCYCLE": 14313,
 "TIMESTAMP": "Feb 6, 2018 2:17:29 PM",
 "MEAN": 24.063229571984394,
 "STD": 92.77500036153945,
 "CURRTIMESTAMP": "2018-07-14 23:53:50.117",
 "UCLX": 36.34560174193815,
 "LCLX": 11.780857402030641,
 "CLX": 24.063229571984394,
 "UCLS": 101.46840975304228,
 "LCLS": 84.08159097003661,
 "CLS": 92.77500036153945,
 "TYPE": "MEASURE",
 "TSEXIT": "2018-07-14 23:53:52.586"
}
```


Listing 4.2: Documento BSON della misura memorizzato in MongoDB

Questi documenti poi vengono consultati con l'add-on MATLAB "Database Toolbox Interface for MongoDB" che permette di connettersi alla collezione remota e scaricare i documenti in una tabella MATLAB. Dai documenti scaricati quindi vengono estratti le informazioni utili per elaborare i grafici dei risultati.

Si parte quindi con i risultati della **latenza**, ovvero quanto tempo ci mette un

ciclo ad essere pubblicato su Kafka e quanto a perseguire su tutta la topologia Storm, per misurare quanto può essere reattivo il sistema:

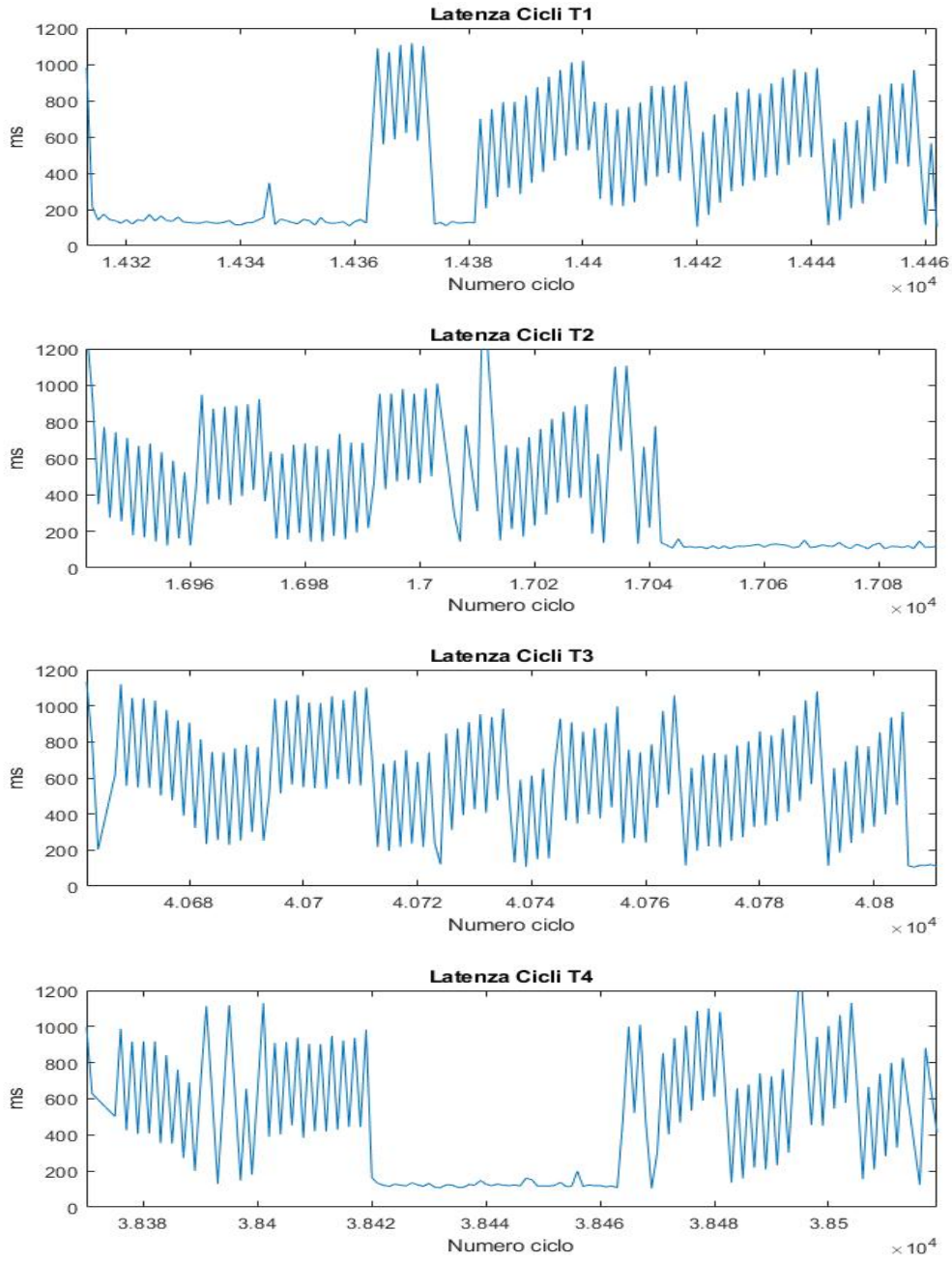

Figura 4.1: Misure della latenza per ciclo macchina

Si può notare un andamento non lineare della latenza, questo è dovuto al fatto

che i nodi della topologia non hanno task replicati. Infatti SPCProcessingBolt che è il nodo che richiede più tempo, dato che al suo interno contiene due cicli eseguiti su 500 valori in media, non è parallelizato per le diverse parti delle macchine, quindi qualora il nodo fosse già occupato da una tupla di un ciclo, la seguente deve aspettare l'acknowledge del bolt per essere processata e il dataset presenta molti cicli che vengono completati contemporaneamente. Ciononostante si può notare dai grafici che la latenza minima è di circa 150ms, negativamente però si visualizzano picchi di 1s a ciclo e questo, nel caso si volesse eseguire una logica reattiva a un probabile guasto della macchina, potrebbe causare criticità. Occorre quindi fare una distinzione delle priorità del dato oppure replicarlo in più tuple per far si che sia inviato per diverse strade della topologia che prevedano priorità differenti, perciò ad esempio se si invia un segnale critico, questo deve essere subito inoltrato al nodo EsperBolt che esegue il controllo della regola per fornire un'effettiva reazione senza che questo venga processato a priori. Bisogna anche dire che le capacità e le risorse di elaborazione della macchina sul quale risiede la topologia sono anche abbastanza limitate e magari in un ambiente dove è richiesto solo il monitoraggio senza reazione, si ottengono già dei buoni tempi, con una latenza media di 478ms, quindi è il fattore principale dei ritardi rimane solo la logica innestata all'interno della topologia.

La seconda misurazione riguarda la visualizzazione dei risultati dell'elaborazione di SPCProcessingBolt, quindi le carte di controllo X ed S ottenute. Questo occorre per realizzare se il processo è in buono stato utilizzando le classiche regole del SPC e per controllare se vi può essere l'occasione, con i cicli dati per il test, dello scattare di un allarme. Si ricorda che le regole EPL utilizzate nell'EsperBolt notificano solo se la media e la deviazione standard fuoriescono dai limiti calcolati con i metodi forniti dall'SPC. I grafici sono divisi per la parte della macchina e come asse delle ordinate è presente il numero di ciclo.

Come si può notare, l'andamento del processo è corretto secondo le regole SPC

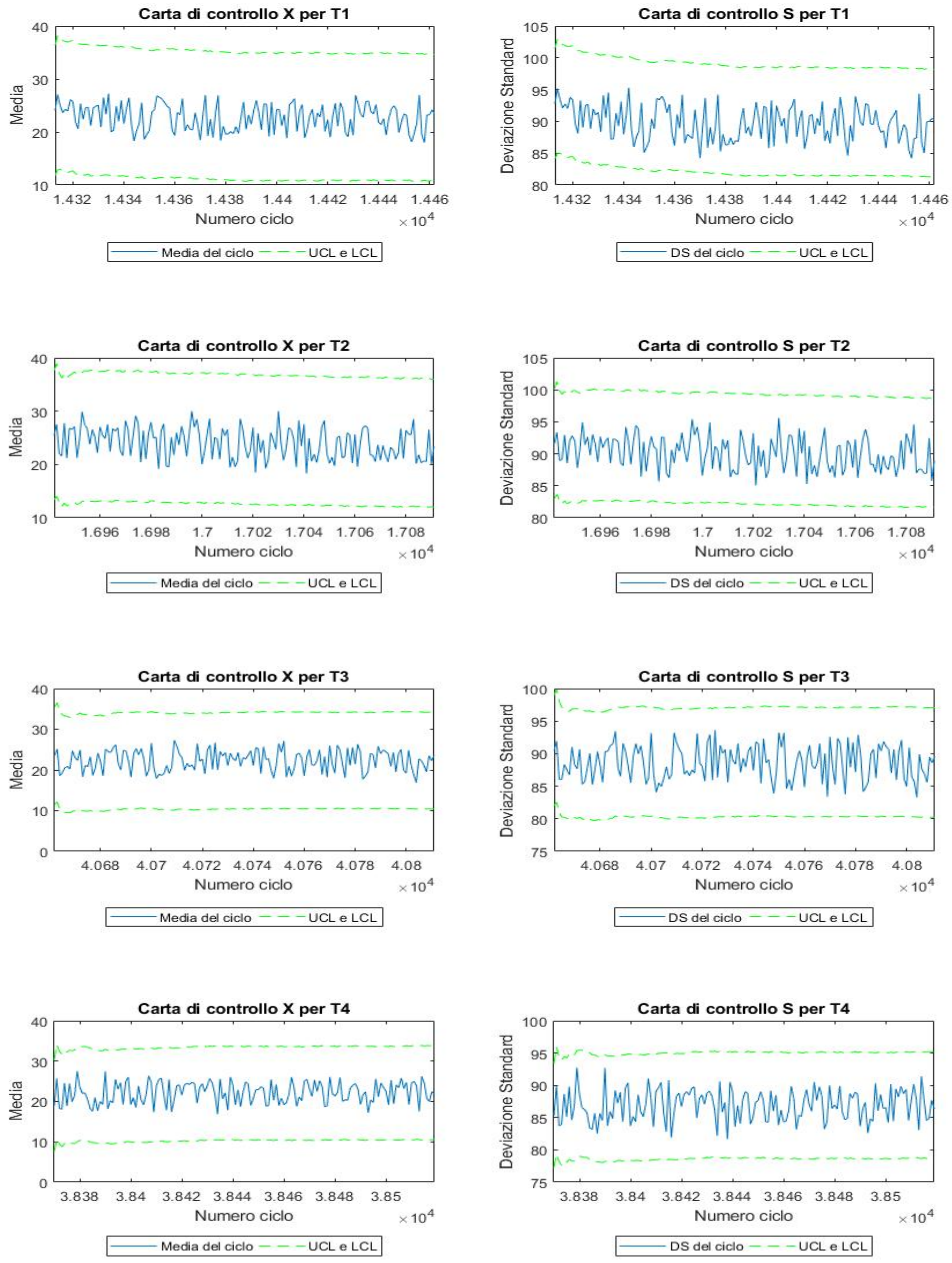

Figura 4.2: Carte di controllo X e S per le parti macchina

impostate, difatti la media e la deviazione standard rimangono sempre all'interno dei limiti imposti dal ciclo. Utilizzando altre regole sulle carte di controllo che non sono state inserite comunque, si può notare che il processo non presenta disfunzioni.

Altre regole ad esempio sono identificare pattern di discesa e di salita delle due unità statistiche, oppure ottenere una statistica che rimane per molto tempo al di sopra o al di sotto della mediana data dalle due soglie, questo può portare ad un cattivo comportamento statistico del sistema. Quindi non si è ottenuto nessun tipo di allarme arrivato come documento BSON, perciò questo test non è rilevante ai fini del corretto funzionamento di EsperBolt o per verificare la sua corretta configurazione. Perciò si è scelto di iniettare un'anomalia nel dataset, quindi di inserire un ciclo con valori esagerati per ottenere un altro tipo di riscontro. Si è quindi inserito un ciclo sbagliato come ultimo prodotto dalla parte T1 per verificare se il sistema produce allarmi oppure meno. Si è quindi creata una collezione "testAnomaly" per raccogliere anche gli allarmi.

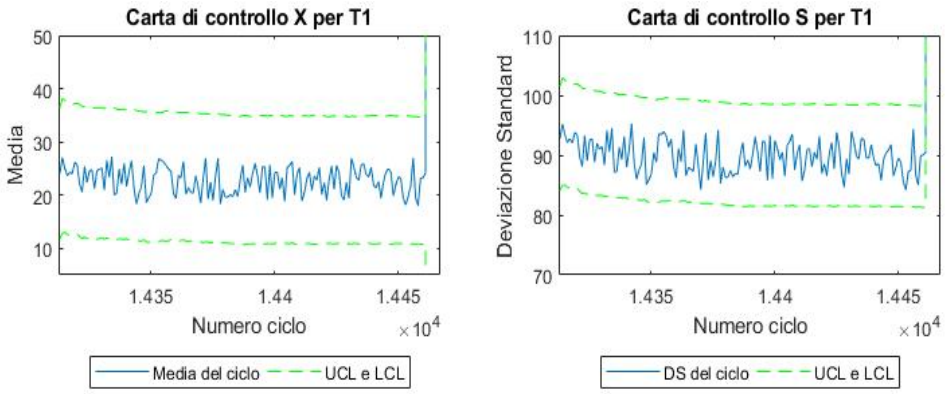

Figura 4.3: Dati ottenuti con anomalia iniettata

Come previsto allo scattare dell'ultimo ciclo anomalo si sono ricevuti i due allarmi relativi all'uscita del confine dei valori di media e deviazione. Dal timestamp sull'allarme si è verificato quanto questo è stato reattivo confrontandolo con il "CURRTIMESTAMP" di partenza dalla macchina, il risultato è di 1,5s di latenza. Tale risultato non è preoccupante se si parla di semplice monitoraggio, ma se occorresse

un particolare tipo di retroazione, tale tempo è piuttosto critico e bisognerebbe sommarvi anche il tempo con qui il messaggio di azione verso la macchina ci mette ad arrivare. Ciononostante si intuisce che tale risultato sia colpa del collo di bottiglia di SPCProcessingBolt, che appunto dovrebbe contenere una logica parallelizzata, piuttosto che il passaggio della tupla sull'EsperBolt che processa tale allarme, dato che Esper viene pubblicizzato come motore CEP con un throughput di un milione di eventi al secondo. Non si vuole però dare un impressione sbagliata invece del controllo sensoristico, difatti per far risaltare l'anomalia e far scattare l'allarme, sono stati immessi dei valori esagerati. Ciononostante occorre fare comunque attenzione a possibili errori sensoristici che potrebbero debilitare il controllo, produrre falsi positivi e che potrebbero danneggiare le regole sino a quando per il calcolo ponderato questi errori diventano irrilevanti.

Come ultime misurazioni, sono state campionate le prestazioni della macchina remota che ospita il prototipo del sistema di Stream Processing. Le specifiche di tale macchina sul Cloud Aruba sono le seguenti:

- **Sistema Operativo:** Ubuntu Server 16.04.4 LTS 64bit
- **Memoria RAM:** 1GB
- **Spazio disco:** 20GB
- **Processore:** 1 vCPU di Aruba, prevede una potenza minima garantita corrispondente a metà di un core fisico di una CPU Intel XEON (Serie 5600)

Per quanto riguardano le prestazioni misurate, sono state calcolate la percentuale di memoria occupata dal sistema sulla macchina e la percentuale di CPU occupata utilizzando il tool *top* di Ubuntu, con periodo di campionamento di due secondi, ricordando che la macchina è pulita da qualsiasi altra applicazione a meno di quelle di base del sistema operativo:

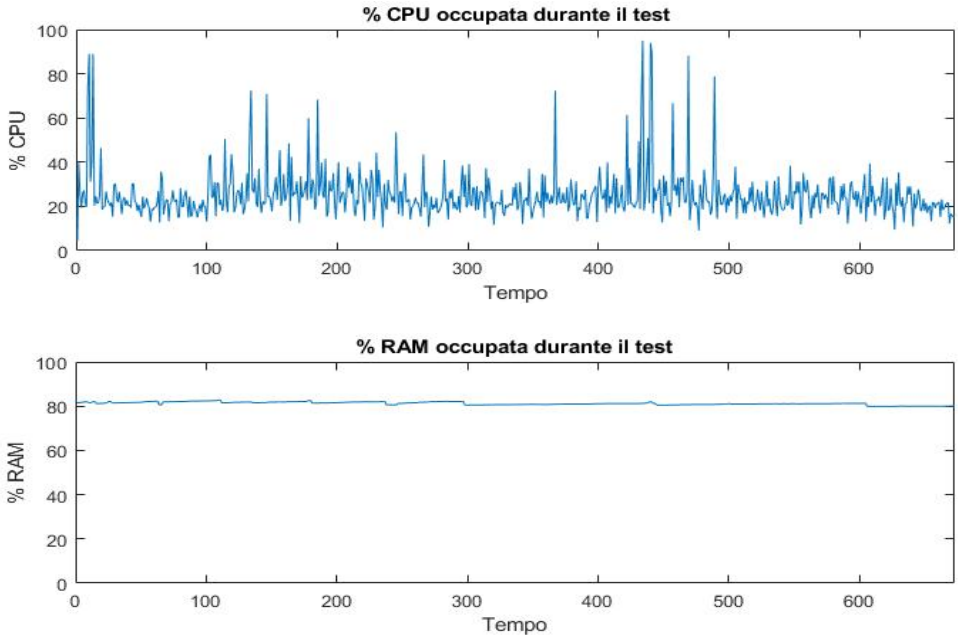

Figura 4.4: Percentuali di RAM e CPU occupate durante il test

Si può notare un andamento costante della RAM occupata anche a partire da prima del test, dato che le applicazioni Apache sono state avviate precedentemente e queste occupano una memoria già preallocata dato che vengono eseguite su JVM. Comunque la memoria occupata è piuttosto alta, quindi forse risulta necessario destinare ciascuno dei servizi Apache su macchine differenti o utilizzarne di più per ciascun servizio in modo da avere più scalabilità delle prestazioni. Il processore comunque non ne risente granché, a meno di qualche picco, soprattutto all'avvio del test e durante i momenti di concentrazione nell'arrivo dei cicli.

4.2 Sviluppi futuri e implementazioni

Premettendo i test, i risultati ottenuti non sono piuttosto promettenti. Questo però non pregiudica il fatto della non possibilità di integrare i servizi usati per ottenere un sistema che si avvicini ad un CPS. Perciò occorre effettuare diverse osservazioni per

gli sviluppi futuri di questo progetto ed il prototipo comunque ha suscitato interesse nell'ambiente lavorativo. In occasione c'è ancora piuttosto lavoro da fare e questo sarà riassunto nei seguenti punti:

- **Edge Computing o Cloud Computing:** i test sul prototipo hanno dato informazioni sulla latenza e considerando un sistema parallelizzato e scalato orizzontalmente, del quale ciascuna macchina si occupa ad esempio dei cicli di una sola macchina o di una sua parte, ed escludendo la parte di logica del processing, occorre considerare la latenza della comunicazione in remoto. L'Edge Computing si basa su utilizzare le macchine instanziate in azienda per sollevare del carico al Cloud Application Layer, non è detto però che un servizio di tale genere possa essere fatto girare internamente all'azienda come applicazione di Edge Computing. Dipende quindi dall'uso che si vuole fare di questo progetto di CPS, se quindi appunto si vogliono fare interagire più aziende o se si vuole effettivamente racchiudere il sistema solo all'interno dell'azienda. Industria 4.0 vuole dire però anche collaborazione, quindi si proseguirà sulla strada di un CPS Cloud per quanto riguarda le interazioni di gestione che la partnership tra aziende si scambia ed un altro focalizzato sul mondo impianti ed interno all'azienda. Si cercherà quindi di proseguire su entrambe le strade, in primis il raggio di azione sarà corto solo per quanto riguarda l'ambiente aziendale, poi si proseguirà per un ambiente Cloud più esteso. Per l'Edge Computing si è pensato di acquistare dispositivi appositi e più potenti rispetto a quello utilizzato per il prototipo che permettono di eseguire il sistema all'interno di un Container per Docker o LXD.

- **Sicurezza prima di tutto:** l'ambiente del prototipo si ritrova su una macchina remota, ciononostante non vi è stata la premura di configurare Kafka per garantire le opportune proprietà di sicurezza, oppure bloccare gli accessi indesiderati alle porte aperte dai servizi di Zookeeper e Storm oppure anche

per quanto riguarda l'accesso via SSH. Per quanto riguarda l'integrità nella comunicazione non vi sono problemi e ciò è gestito direttamente dal funzionamento alla base di Kafka. Ciononostante non sono stati scelti gli opportuni algoritmi simmetrici per autenticazione del messaggio e riservatezza e non è stato costruito nemmeno un canale VPN sicuro. Occorre fare però attenzione alle possibilità di avere dei problemi ulteriori di latenza con le misure di sicurezza, quindi si continueranno i test in locale o in un ambiente sicuro fino a quando non si avranno le certezze di risolvere i problemi di criticità.

- **Machine Learning Online o Offline:** i risultati della latenza a causa del processing delle tecniche SPC un po' hanno spaventato la possibilità di eseguire algoritmi di Machine Learning online, però effettivamente occorre pensare che la topologia non è stata scalata orizzontalmente. Perciò questa possibilità è ancora da tenere in conto. Ciononostante occorre pensare che applicando delle logiche apposite per un particolare tipo di dato con una sua struttura emessa dalla macchina, possa ledere la generalità del sistema e quindi si ritorna al contesto prevalentemente della macchina. Per questo motivo si è cercato di integrare il CEP, in maniera tale che, qualora fosse stato scelto un effettivo algoritmo per il Machine Learning atto alla produzione di regole, si possano installare direttamente con delle "add" a delle mappe le regole in maniera on-the-fly e senza ledere tutta la struttura della topologia. Si pensa quindi di utilizzare un database apposta per i tipi di evento, le regole e le azioni a seguire (indirizzi specifici degli interessati, URL per richieste a Web Service etc. per l'invio di allarmi e notifiche) e di sviluppare un produttore di regole a partire dall'algoritmo di Machine Learning come nell'esempio di autoCEP. Oltre a questo si potrebbe realizzare un client a disposizione dell'utente per facilitare la scrittura delle regole EPL. Ovviamente sarà necessario utilizzare più dataset possibile per verificare il funzionamento per più casi d'uso, possibilmente anche

con i dati di output sulle caratteristiche qualitativi del prodotto o di vita media se si parla di manutenzione predittiva.

- **Standardizzazione del dato:** in azienda sono stati eseguiti degli studi su OPC-UA per integrare l'ambiente MES prodotto in AEC Soluzioni S.R.L con le macchine dei clienti. Il progetto della control-room potrebbe sfruttare lo standard per adeguarsi almeno a quel contesto e facilitare la mappatura della topologia riuscendo a coprire più casi d'uso possibile.

- **Interfaccia di monitoraggio:** per il progetto si pensava di realizzare una dashboard utilizzando HTML5 e JavaScript per visualizzare l'avanzamento del processo più eventuali anomalie. Al momento sono state effettuate delle prove con JavaFX per creare dei client che permettessero di monitorare il dato inviato in opportuni grafici, soprattutto istogrammi a barre e a linee.

Come si è visto il lavoro da fare è ancora molto, ma questa soluzione ha portato a seguire una strada nuova per adeguarsi all'ambiente di Industria 4.0 e continuare ad evolvere, senza costi particolari, l'ambiente industriale.

Conclusioni

Con questo capitolo si va a chiudere la tesi discutendo brevemente i risultati ottenuti dal lavoro ed esponendo critiche su quello che si è venuto a realizzare.

In primis si sono introdotte le materie innovative che racchiudono lo spirito dell'Industria 4.0. Per realizzare comunque un prodotto per tale iniziativa occorre ancora molto tempo e soprattutto una particolare attenzione a quelle che sono le problematiche presentate da queste materie delle scienze informatiche. Comunque le opportunità realizzabili sono molteplici e si potrebbero estendere a qualsiasi caso d'uso nel settore manifatturiero. Si punterà molto quindi sulla proattività nelle decisioni da prendere per il Business Model aziendale ed il Machine Learning verrà sfruttato per ottenere un modello per poter predire la scelta da effettuare. Oltre ad un punto di vista puramente gestionale, si parla molto di macchine intelligenti, che permettono di prevedere, in base ai dati delle proprie lavorazioni, la qualità del prodotto in uscita e anche se deve effettuare operazioni di natura manutentiva. Non esiste poi Machine Learning senza una base di BDA, che viene utilizzata sia sui dati "at rest" sia sui flussi di dati. Tale attività di analizzare e preelaborare il flusso di dati e prendere decisioni su questo viene svolto dallo Stream Processing. In alternativa, se nel flusso di dati si rileva un evento o una serie di eventi che risultano rilevanti per il loro carico di informazioni, il Complex Event Processing permette di rilevare facilemente i pattern di eventi e di inoltrarli agli agenti interessati. Questa serie di materie serve appunto per riuscire a trovare tecnologie e prodotti che permettano di

realizzare una Control-Room in pieno stile Industria 4.0 e di permettere un controllo in tempo reale dell'avanzamento del processo nel reparto produttivo.

Quindi si sono identificate le opportunità rilevanti dell'Industria 4.0 e ci si è focalizzati su quattro prodotti che appunto occorrono per l'acquisizione dei dati e della loro elaborazione in real-time per fornire una base per realizzare l'ambiente di monitoraggio. Quindi è stato implementato un prototipo che potesse definire come questi prodotti potessero essere integrati per fornire tale servizio. Dopo aver esposto le aspettative dei quattro prodotti all'interno del sistema di Stream Processing, è stato effettuato un test per rilevare e interpretare le misurazioni che ve ne sono uscite. Queste interpretazioni possono portare a nuovi sviluppi e strade nel progetto e riuscire a realizzare un prodotto che si possa avvicinare al sistema CPS tanto richiesto dal paradigma Smart Factory.

Bibliografia

[1] S. Madden. «From Databases to Big Data». In: *IEEE Internet Computing* 12 (2012), pp. 4–6.

[2] C. Min, M. Shiwen e L. Yunhao. «Big Data: A Survey». In: *Mobile Network Application* 19 (2014), pp. 171–209. DOI: 10.1007/11036-013-0489-0.

[3] D. Laney. «3D Data Management: Controlling Data Volume, Velocity, and Variety». In: *Application Delivery Strategies* (2001).

[4] S. Hadi et al. «Big Data Analytics for Wireless and Wired Network Design: A Survey». In: *School of Electronic and Electrical Engineering, University of Leeds* (2018).

[5] C. Philip Chen e C. Zhang. «Data-intensive application, challenges, techniques and technologies: A survey on Big Data». In: *Information Sciences* 275 (2014), pp. 314–347.

[6] K. Vassakis, E. Petrakis e I. Kopanakis. «Big Data Analytics: Applications, Prospects and Challenges». In: *Mobile Big Data* 10 (2018). DOI: 10.1007/978-3-3-319-67925-9_1.

[7] C. Greleck, S. Scholz e Shafarenko A. «Asynchronous Stream Processing with S-Net». In: *IntJ Parallel Program* (2010), pp. 38–67. DOI: 10.1007/s10766-009-0121-x.

[8] D. Robins. «Complex Event Processing». In: *Proceeding of the 17th international conference on WWW* (2010). DOI: 10.1145/1367497.1367741.

[9] Dekkers P. «Complex Event Processing». In: *Radboud University Nijmegen* (2007).

[10] M. Kantardzic. *Data Mining: concepts, models, methods and algorithms*. A cura di IEEE. second edition. Wiley, 2011.

[11] S. Weyer et al. «Towards Industry 4.0 - Standardization as the crucial challenge for higly modular, multi-vendor production systems». In: *IFAC* (2015), pp. 579–584.

[12] K.D. Thoben, S. Wiesner e T. Wuest. «"Industrie 4.0" and Smart Manufactoring - A Review of Research Issues and Application Examples». In: *Int. J. of Automation Technology* 1 (2017).

[13] «PwC - Global Industry 4.0 Survey». In: (2016).

[14] «Piano Nazionale Industria 4.0». In: (2017).

[15] B. Chen et al. «Smart Factory of Industry 4.0: Key Technologies, Application Case, and Challenges». In: *IEEE* 6 (2017), pp. 6505–6519.

[16] *Sito AEC Soluzioni S.R.L.* URL: https://www.aecsoluzioni.it/wp/it/.

[17] M. Syafrudin et al. «An Open Source-Based Real-Time Data Processing Architecture Framework for Manufacturing Sustainability». In: *Sustainability* 9 (2017).

[18] P. Dobbelaere e S.E. Kyumars. «Industry Paper: Kafka versus RabbitMQ». In: *DEBS '17* (2017).

[19] *Sito Apache Kafka*. URL: http://kafka.apache.org/.

[20] *Sito Apache Zookeeper*. URL: https://zookeeper.apache.org/.

[21] S. Kamburugamuve. «Survey of Distribuited Stream Processing for Large Stream Sources». In: (2013).

[22] *Sito Apache Storm*. URL: http://storm.apache.org/.

[23] L.J. Fulop et al. «Survey on Complex Event Processing and Predictive Analytics». In: (2010).

[24] *Sito Espertech Esper*. URL: http://www.espertech.com/.

[25] S. Agrawal et al. «Survey on MongoDB: an open-source document database». In: *IJARET* 6 (2015), pp. 01–11.

[26] *Sito MongoDB*. URL: https://www.mongodb.com/.

[27] M.A. Mohamed, O.G. Altrafi e M.O. Ismail. «Relational vs. NoSQL Databases: A Survey». In: *IJCIT* 3 (2014).

[28] *Introduction to Anomaly Detection*. URL: https://www.datascience.com/blog/python-anomaly-detection.

[29] D.C. Montgomery. *Introduction to Statistical Quality Control*. A cura di Arizona State University. sixth edition. Wiley, 2009.

Ringraziamenti

La famiglia prima di tutto! Si ringrazia mia madre **Susanna** e mio padre **Leonardo**, a cui sono grato di aver creduto sempre in me e di avermi supportato nel mio percorso. Sempre fiero di essere vostro figlio.

Ringrazio mia sorella **Francesca** e il suo fidanzato **Marco** per il loro continuo incoraggiamento. Piuttosto mi dovrei congratulare io con loro per avermi fatto conoscere il piccolo **Alessandro**, il bimbo più felice e rumoroso mai apparso sulla faccia della Terra. Quanto cuore ragazzi!!

Per completare il quadretto, si ringraziano i nonni **Ivano** e **Bruna**, gli zii **Graziano** e **Paola** e mia cugina **Irene**.

Voglio ringraziare in ordine alfabetico **Cubo** perché siamo amiXi, **Giugy** per l'oroscopo, **Katia** per la taverna, **Lisa** perché non capisce niente di calcio, **Marco** perché ha scelto Ingegneria (Meccanica ☹), **Matte** per le offerte telefoniche, **Vale** per il fondo pensione integrativo e **Viki** per la casa al mare. Grazie ragazzi, gruppo fantastico, cene ottime! Adepti del tonno TOP!!

Ringrazio **Anto** e **Mafy** amici sin dalla triennale a Ferrara. Li ringrazio per le avventure passate in questi tre anni a Torino. Grazie a voi per le ignorantate fatte in Via Buenos Aires 60. I migliori film li ho "visti" con loro. Ma soprattutto li ringrazio per avermi fatto realizzare che la Puglia è un posto fantastico.

Ringrazio i ragazzi di **AEC Soluzioni**, per avermi dato il coraggio a finire questo lavoro e perché senza questa esperienza non avrei mai assaggiato le mini-sacher.

Ringrazio tutti quelli che non ho ringraziato.

Ringrazio tutti i ragazzi che ho conosciuto a Torino, il Politecnico e questa meravigliosa città. Oh sì, bel posto, bella gente, però...

VIVA FERRARA E FORZA SPAL!!

Grazie.