POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica (Computer Engineering)

Tesi di Laurea Magistrale

Progettazione e sviluppo di un'architettura scalabile per il calcolo dei Key Performance Indicators (KPI)

Relatrice
prof.ssa Tania Cerquitelli

Candidato
Gianluca Raffa

Anno Accademico 2018-2019

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Big Data I Big Data sono quei dati, che per il loro volume, la loro complessità e la loro diversità richiedono nuove architetture, tecniche, algoritmi e strumenti di analisi per essere manipolati, in modo da estrarre del valore e della conoscenza da essi.
Sebbene il termine big data sia relativamente nuovo, la tendenza a raggruppare e immagazzinare ampi volumi di informazione, per un'eventuale analisi futura è molto antica. Il concetto prende piede nei primi anni 2000 quando Doug Laney, formula la ormai nota definizione delle tre V dei big data:

- Volume: si riferisce alla quantità di dati(strutturati e non) generati ogni secondo.
- Varietà: indica la differente tipologia dei dati che vengono generati, collezionati ed utilizzati. I dati possono essere di qualunque tipo essendo generati da sorgenti eterogenee quali: sensori, log, eventi, mail, social media, database tradizionali.
- Velocità: i dati fluiscono ad una velocità senza precedenti e vanno pertanto gestiti in maniera tempestiva. L'utilizzo di sensori e la crescita dell'IoT aumentano la necessità di gestire fiumi di dati in tempo reale o quasi.

Successivamente, sono state introdotte altre 2 V:

- Veridicità: considerando la varietà di sorgenti e la velocità alla quale i dati possono variare, è molto probabile che non si possa garantire la stessa qualità dei dati utilizzati nei processi di ETL tradizionale. E' fondamentale quindi assegnare un indice di veridicità ai dati su cui si basano le analisi, in modo da avere una misura dell'affidabilità.
- Valore: si riferisce alla capacità di trasformare i dati in valore.

Big Data e KPI Al giorno d'oggi esistono molte applicazioni di tipo data-intensive che producono grandi moli di dati, come ad esempio quelle utilizzate in ambito industriale grazie al nuovo paradigma Industry 4.0. La sfida più grande consiste nella

capacità di riuscire a lavorare questi grossi volumi di dati per prendere le migliori decisioni aziendali e acquisire un vantaggio competitivo.
L'analisi e l'elaborazione dei Big Data da la possibilità di calcolare i KPI(Key Performance Indicators), cioè gli indicatori chiave di prestazione. I KPI non sono altro che dei valori, estratti dai dati, che consentono di analizzare, comprendere e sintetizzare meglio i piani e i risultati della propria azienda. I KPI forniscono informazioni su ciò che funziona correttamente, identificando al contempo potenziali aree problematiche in tempo utile per l'adozione di azioni correttive.
Al giorno d'oggi, esistono un gran numero di KPI e le nuove tecnologie permettono di raccogliere un'infinità di dati: tuttavia avere troppe informazioni può essere svantaggioso. E', quindi, necessario stabilire, in primis, quale sia l'obiettivo da raggiungere e successivamente selezionare il KPI più utile al raggiungimento dello stesso. Infine, bisogna progettare un'architettura in grado di raccogliere, memorizzare ed elaborare i dati necessari per il calcolo del KPI.
Una volta calcolato il KPI, è necessario osservarne l'andamento in un periodo di tempo sufficientemente lungo, in modo da identificare le problematiche presenti e le azioni da intraprendere per il miglioramento delle prestazioni.

Infrastrutture e framework per i Big Data Il problema che si riscontra nell'utilizzo dei Big Data è dovuto principalmente alla difficoltà di gestire questi dati con le tecnologie tradizionali. Per questo motivo è necessario che un azienda dedichi un'infrastruttura adeguata e specifica, in grado di raccogliere, archiviare ed elaborare i dati per presentarli in forma utile.
Il limite principale delle architetture tradizionali è che sono di tipo "processor bound": vanno, quindi, bene per i datasets di dimensione limitata su cui è eseguita un'elaborazione molto intensiva. I big data richiedono, invece, un'infrastruttura che permetta di trasferire continuamente grandi quantità di dati tra processore e memoria. Purtroppo, la velocità di trasferimento dei dati, non essendosi evoluta tanto velocemente quanto la capacità di storage, è divenuta il collo di bottiglia dell'intero sistema. La soluzione a ciò è stata ideata cambiando totalmente il paradigma su cui si basano le architetture hardware: piuttosto che trasferire i dati tra il disco ed il processore, si preferisce trasferire il processing sui dati. Ciò implica la necessità di distribuire i dati su più dischi, ognuno dei quali contiene una porzione del grande dataset, ed effettuare il processing in parallelo sui vari dischi. Al fine di gestire le problematiche generate da questa nuova infrastruttura, quali la necessità di sincronizzare i vari processi, la gestione dei guasti hardware, la perdita dei dati, la scalabilità e il joining dei dati presenti nei vari dischi, è stato sviluppato il framework Hadoop. Hadoop fornisce le seguenti funzionalità:

- Distribuzione automatica dei dati sui vari nodi.
- Replicazione dei dati, necessaria per aumentare la disponibilità e l'affidabilità del sistema.
- Processing eseguito direttamente sui dati, in modo da minimizzare la quantità di dati trasferiti durante l'elaborazione.

Tuttavia, in presenza di jobs di tipo iterativo, anche Hadoop richiede un elevato numero di I/O per ogni iterazione e stage. La naturale evoluzione di Hadoop è Spark che nasce dall'opportunità di acquistare memorie primarie ad un costo molto più basso rispetto al passato. Spark è basato sull'idea di aumentare la quantità di dati memorizzati nella memoria principale, le quali sono tra le 10 e le 100 volte più veloci rispetto ai dischi secondari. I dati vengono quindi suddivisi e memorizzati nelle memorie primarie dei vari server e memorizzati sotto forma di RDD(Resilient Distributed Datasets): i programmi Spark saranno costituiti da una sequenza di operazioni da eseguire sugli RDD. Il framework Spark gestirà in modo automatico le complessità che nascono dalla necessità di avere un sistema fault tolerant. Inoltre si occuperà dello scheduling e della sincronizzazione dei vari jobs.

Big Data e DBMS NoSQL Un'altra problematica che si riscontra nella progettazione di un'architettura per la gestione dei Big Data è la difficile convivenza tra gli stessi Big Data e i database tradizionali di tipo SQL. Questi DBMS non stati, infatti, ideati per scalare orizzontalmente e per la gestione di insieme di dati così grandi e così vari. In questo contesto, i DBMS NoSQL, come Cassandra e MongoDB, trovano grande utilizzo. Le loro caratteristiche principali sono infatti:

- Rappresentazione dei dati senza uno schema fisso: in tal modo non è necessario definire i dati in anticipo e questi possono cambiare nel tempo;
- Scalabilità: i DBMS NoSQL sono progettati per garantire la scalabilità orizzontale, attuata usando cluster distribuiti di hardware;
- Memorizzazione dei dati in formato diverso: esistono infatti, ad esempio, DBMS di tipo document-based come MongoDB o column-oriented come Cassandra.

Descrizione del lavoro L'obiettivo di questa tesi è quello di progettare e sviluppare un'architettura, attraverso le tecnologie più moderne e scalabili, per il calcolo dei KPI (Key Performance Indicators). L'architettura sarà customizzata per effettuare l'analisi dell'efficienza produttiva in due ambiti distinti: quello industriale e quello universitario.
Il primo caso di studio è l'analisi dell'efficienza produttiva di una delle linee di produzione presenti in Procemsa, un'industria farmaceutica torinese. Il KPI scelto per l'analisi dell'andamento nel tempo delle prestazioni della linea di produzione è l'OEE(Overall Equipment Effectiveness).
Il secondo caso di studio è la produttività degli afferenti al dipartimento di automatica ed informatica del Politecnico di Torino. A tale scopo, è stata progettata una pagina Web, contenente una dashboard nella quale sono presenti vari grafici che indicano l'andamento di diversi KPI.

Il lavoro di tesi è strutturato come segue.
Il presente capitolo funge da introduzione. Descrive il concetto di Big Data e l'importanza dei KPI al giorno d'oggi. Viene, inoltre, posto il focus sull'inadeguatezza delle tecnologie tradizionali nella gestione dei Big Data. Infine, è presentata la struttura della tesi.
Nel capitolo 2 sono descritte le tecnologie ed i software utilizzati per la realizzazione dell'architettura per il calcolo dei KPI. In particolar modo, dapprima sono analizzati i sistemi per la gestione delle basi di dati. Successivamente viene descritto Apache Spark, un framework per la gestione dei Big Data. Vengono, quindi, analizzati i tool per lo sviluppo delle applicazioni Web. Infine vengono illustrate le caratteristiche principali della piattaforma Docker.
Nel capitolo 3 viene descritta la progettazione e lo sviluppo dell'architettura in questione. In particolar modo, vengono illustrati i blocchi logici principali da cui deve essere composta e le due istanze reali sviluppate. Per ognuna delle due customizzazioni sono descritte le varie tecnologie utilizzate e le motivazioni per le quali sono state scelte.
Nel capitolo 4 viene analizzato il caso di studio Procemsa. Dapprima è spiegato cosa è l'OEE, come può essere calcolato ed a cosa è utile. Successivamente è descritto il lavoro effettuato per il calcolo dell'OEE. Infine vengono mostrati i risultati ottenuti.
Nel capitolo 5 viene analizzato il caso di studio Dauin. Viene descritta la procedura di realizzazione della dashboard ed i risultati ottenuti.
Il capitolo 6 presenta le conclusioni della tesi e gli eventuali sviluppi futuri.

Capitolo 2

Tecnologie

In questo capitolo, sono illustrate le principali soluzioni tecnologiche, che oggi possono essere considerate state of the practice, perchè mature dal punto di vista della ricerca e facilmente utilizzabili e integrabili nello sviluppo di applicazioni data-intensive. Le tecnologie presentate possono essere categorizzate in tre macrogruppi: i sistemi per la gestione di basi di dati, le infrastrutture per i big data, i tool per lo sviluppo di applicazioni web. Infine è presentata la piattaforma Docker, una tecnologia per lo sviluppo ed il facile deployment di applicazioni innovative su sistemi già esistenti.

2.1 Sistemi per la gestione di basi di dati

In questa sezione sono illustrate le principali caratteristiche dei due DBMS NoSQL scelti: Apache Cassandra e MongoDB.

2.1.1 Cassandra

Cassandra è un database management system non relazionale distribuito con licenza open source e ottimizzato per la gestione di grandi quantità di dati. Il codice di Cassandra è stato inizialmente sviluppato all'interno di Facebook (per potenziare la ricerca all'interno del sistema di posta) da Avinash Lakshman e Prashant Malik. Nel luglio del 2008 sono stati resi disponibili i sorgenti, su Google Code; dal marzo 2009 è entrato a far parte del progetto Incubator di Apache Software Foundation, data in cui l'intero progetto ha iniziato a essere distribuito sotto la Apache License 2 [7]. Nel seguito analizzeremo le caratteristiche principali di Cassandra [6].

Scalabilità orizzontale La scalabilità orizzontale si riferisce alla possibilità di espandere la capacità di archiviazione e di elaborazione di un database aggiungendo più server a un cluster di database. La capacità di archiviazione di un database tradizionale monomaster è limitata dalla capacità del server che ospita l'istanza master. Se il set di dati supera questa capacità e non è disponibile un server più potente, il set di dati deve essere suddiviso tra più istanze di database indipendenti che non

sanno nulla l'una dell'altra. L'applicazione ha la responsabilità di sapere a quale istanza appartiene un dato pezzo di dati. Cassandra, invece, è sviluppata come un gruppo di istanze consapevoli l'una dell'altra. Dal punto di vista dell'applicazione client, il cluster è una singola entità; l'applicazione non deve sapere, né preoccuparsi, a quale macchina appartiene un dato. I dati potranno essere letti o scritti in una qualsiasi istanza del cluster, chiamata nodo; questo nodo inoltrerà la richiesta all'istanza a cui i dati appartengono effettivamente. Il risultato è che le distribuzioni di Cassandra hanno una capacità quasi illimitata di memorizzare ed elaborare i dati; quando è richiesta una capacità aggiuntiva, è sufficiente aggiungere più macchine al cluster. Quando nuove macchine entrano a far parte del cluster, Cassandra si occupa di riequilibrare i dati esistenti in modo che ogni nodo del cluster ampliato abbia una quota pressoché uguale.

Alta disponibilità I database più semplici vengono eseguiti come un'unica istanza su un singolo server. Questo tipo di configurazione è altamente vulnerabile alle interruzioni: se il server è interessato da un guasto hardware o da un'interruzione della connessione di rete, la capacità dell'applicazione di leggere e scrivere dati è completamente persa fino al ripristino del server. Se il guasto è catastrofico, i dati su quel server potrebbero andare completamente persi. Un'architettura master-follower migliora un po' questa situazione. L'istanza master riceve tutte le operazioni di scrittura, e poi queste operazioni vengono replicate nelle istanze follower. L'applicazione è in grado di leggere i dati dal master o da una qualsiasi delle istanze follower, quindi un singolo host che non è disponibile non impedirà all'applicazione di continuare a leggere i dati. Un guasto del master, tuttavia, impedirà comunque all'applicazione di eseguire qualsiasi operazione di scrittura, per cui, sebbene questa configurazione fornisca un'elevata disponibilità in lettura, non fornisce completamente un'elevata disponibilità. Cassandra, invece, non ha un single point of failure per la lettura o la scrittura dei dati. Ogni dato viene replicato su più nodi, ma nessuno di questi nodi detiene la copia principale. Se una macchina non è più disponibile, Cassandra continuerà a scrivere i dati sugli altri nodi che condividono i dati con quella macchina, terrà una coda delle operazioni eseguite in modo da aggiornare il nodo quando si ricongiungerà al cluster. Ciò significa che in una configurazione tipica, più nodi devono fallire simultaneamente affinchè Cassandra non sia disponibile all'applicazione.

Ottimizzazione della scrittura I database relazionali e document-based sono ottimizzati per le prestazioni in lettura. La scrittura dei dati in un database relazionale comporta tipicamente l'aggiornamento di complesse strutture di dati su disco, al fine di mantenere una struttura di dati che può essere letta in modo efficiente e flessibile. L'aggiornamento di queste strutture dati è un'operazione molto costosa dal punto di vista dell'I/O del disco, che è spesso il fattore limitante per le prestazioni del database. Poiché le scritture sono più costose delle letture, in genere si evitano aggiornamenti inutili di un database relazionale, anche a spese di operazioni di lettura aggiuntive. Cassandra, invce, è altamente ottimizzato per il troughput in scrittura: di fatto non modifica mai i dati sul disco, incrementando solo i file

esistenti o creandone di nuovi. Poiché sia la scrittura che l'archiviazione dei dati in Cassandra sono operazioni poco costose, la denormalizzazione ha un costo contenuto e ciò garantisce che i dati possano essere letti in modo efficiente in vari scenari di accesso.

Strutturazione dei records In Cassandra, i record sono strutturati nello stesso modo in cui sono strutturati in un database relazionale, utilizzando tabelle, righe e colonne. In questo modo, le applicazioni che utilizzano Cassandra possono godere di tutti i vantaggi dello storage distribuito masterless, ottenendo al contempo tutte le funzionalità avanzate di modellazione dei dati e di accesso associate ai record strutturati.

Ordinamento efficiente dei risultati E' un'operazione abbastanza comune voler recuperare un set di record ordinato per un particolare campo. Poiché l'ordinamento dei dati al volo è un'operazione fondamentalmente costosa, i database devono conservare su disco le informazioni relative all'ordine dei record, al fine di restituire in modo efficiente i risultati in ordine. In un database relazionale, questo è uno dei lavori di un indice secondario. In Cassandra, gli indici secondari non possono essere usati per ordinare i risultati, quindi l'ordinamento su colonne arbitrarie al momento della lettura non è possibile. Tuttavia le tabelle possono essere strutturate in modo tale che le righe siano sempre ordinate per una data colonna o per un insieme di colonne: queste sono chiamate colonne di clustering.

Consistenza dei dati Quando scriviamo dei dati in un database, ci auguriamo che i dati siano immediatamente disponibili a qualsiasi altro processo che desideri leggerli. Da un altro punto di vista, quando leggiamo alcuni dati da un database, vorremmo essere certi che i dati che recuperiamo siano aggiornati alla versione più recente. Questa proprietà si chiama consistenza immediata, ed è una proprietà dei più comuni database monomaster. I sistemi distribuiti come Cassandra, invece, in genere non forniscono una garanzia di consistenza immediata. Quindi, gli sviluppatori accettano la cosiddetta consistenza eventuale: ciò significa che quando i dati vengono aggiornati, il sistema rifletterà l'aggiornamento in futuro. Gli sviluppatori sono disposti a rinunciare alla consistenza immediata proprio perché si tratta di un compromesso diretto con l'alta disponibilità. Nel caso di Cassandra, questo compromesso è reso esplicito attraverso la tunable consistency. Ogni volta che si esegue una scrittura o una lettura dei dati, si ha la possibilità di scegliere tra una consistenza immediata con una disponibilità meno resiliente e una consistenza eventuale con una disponibilità estremamente resiliente.

MapReduce Il MapReduce è una tecnica per l'esecuzione di elaborazioni aggregate su grandi quantità di dati in parallelo; è una tecnica particolarmente comune nelle applicazioni di data analytics. Cassandra non offre funzionalità integrate di MapReduce, ma può essere integrata con Hadoop o Spark.

CQL Il CQL(Cassandra Query Language) è l'interfaccia principale del DBMS Cassandra. CQL è fortemente ispirato all'SQL; infatti, molte istruzioni CQL sono istruzioni SQL altrettanto valide. Tuttavia, CQL e SQL non sono interscambiabili. CQL manca di una grammatica per le caratteristiche relazionali come le istruzioni JOIN, che non sono possibili in Cassandra. Al contrario, CQL non è un sottoinsieme di SQL; i costrutti per recuperare il tempo di aggiornamento di una data colonna, o per eseguire un aggiornamento in una transazione leggera, che sono disponibili in CQL, non hanno un equivalente SQL.

Strutturazione dei dati

Keyspace Cassandra è organizzata in keyspace [6]. Un keyspace è una collezione di tabelle collegate tra loro, corrispondente al concetto di database in un sistema relazionale.

Tabelle Cassandra struttura le tabelle in righe e colonne, proprio come un database relazionale. Come in un database relazionale, le colonne disponibili per una tabella sono definite in anticipo. Non è possibile aggiungere nuove colonne al volo quando si inseriscono i dati, anche se è possibile aggiornare lo schema di una tabella esistente. Ogni tabella definisce una o più colonne che fungono da chiave primaria; ogni riga è identificata univocamente dal valore della sua chiave primaria, e quelle colonne non possono essere lasciate vuote in nessuna riga. Cassandra non offre chiavi primarie auto-incrementali; ad ogni riga, quando creata, deve essere esplicitamente assegnata una chiave primaria dal client. Un buon modo per strutturare la chiave primaria è quello di utilizzare una chiave naturale, che è un valore fondamentalmente unico per ogni riga che si vuole memorizzare. Per quanto riguarda le colonne che non fanno parte delle chiave primaria, Cassandra non ha il concetto del valori "NULL": queste colonne o hanno dati o non ne hanno. Cassandra è quindi più flessibile di un database relazionale in quanto può gestire righe con contenuto e numero di colonne variabile: una tabella in Cassandra non può quindi essere pensata come un rettangolo [16] .

Chiave primaria composta Un concetto molto importante in Cassandra è quello di chiave primaria composta, che si ha nel caso in cui la chiave primaria è composta da più colonne. In tal caso, la prima parte della chiave primaria sarà chiamata chiave di partizione: Cassandra memorizza le righe con la stessa chiave di partizione insieme in modo che la ricerca all'interno di una partizione sia molto efficiente. Dall'altra parte, effettuare una query su più partizioni è un operazione molto costosa e quindi da evitare. La seconda parte della chiave primaria sarà, invece, chiamata colonna di clustering, il cui compito è quello di determinare l'ordine delle righe all'interno di una partizione. Ciò permette di leggere i dati all'interno di una partizione in modo già ordinato, senza operazioni costose durante la lettura.

Architettura del sistema

Gran parte del potere di Cassandra risiede nel fatto che si tratta di un database distribuito: piuttosto che memorizzare tutti i dati su una singola macchina, è progettato per distribuire i dati su più macchine. Un'architettura distribuita è estremamente vantaggiosa per la scalabilità in quanto non è vincolata alla capacità hardware di una singola macchina; se si ha bisogno di più storage o maggiore potenza di elaborazione, è sufficiente aggiungere più nodi al cluster Cassandra. È anche un vantaggio per la disponibilità: memorizzando più copie dei dati su più macchine, Cassandra è in grado di resistere al guasto di un particolare nodo.
Il vantaggio di un database distribuito come Cassandra è che, come sviluppatori di applicazioni, raramente dobbiamo pensare al fatto che stiamo lavorando con dati distribuiti su più server: il database si occupa di capire da quale macchina o macchine i dati vengono scritti o letti. Detto questo è, comunque, importante capire come i dati vengono distribuiti e replicati.

Distribuzione dei dati Cassandra ha una funzione TOKEN che genera un valore intero per qualsiasi chiave di partizione; quando recuperiamo i risultati su chiavi di partizione multiple, le righe sono ordinate da questo token. Quando Cassandra distribuisce i dati, assegna ad ogni nodo una serie di token; una riga è memorizzata sul nodo nel cui intervallo di token rientra il token della chiave di partizione. Poiché i token sono generati utilizzando una funzione di hashing, i valori dei token sono distribuiti uniformemente su tutto l'intervallo di valori possibili. Quindi, finché il numero di chiavi di partizione è molto più grande del numero di nodi nel cluster, le chiavi di partizione saranno bilanciate in modo uniforme tra i diversi nodi, se ogni nodo è responsabile di una porzione di uguale dimensione dell'intervallo dei token.
Adesso consideriamo un esempio in cui abbiamo un cluster a tre nodi. I token Cassandra sono interi con segno a 64-bit, quindi il minimo hash possibile è -2^{-63} e il massimo hash possibile è $2^{63} - 1$. Quindi ad ogni nodo corrisponderanno i seguenti intervalli, come mostrato nella figura 2.1: Abbiamo visto in precedenza che

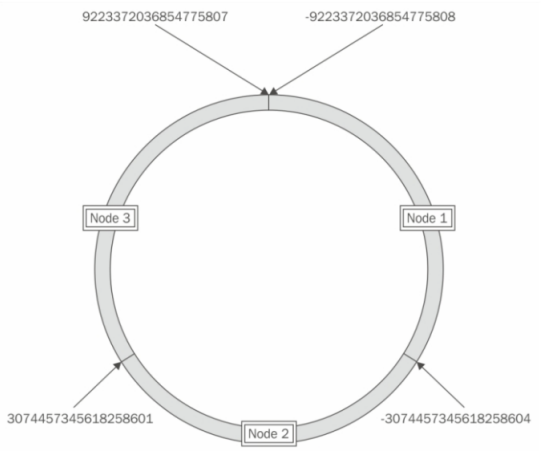

Figura 2.1: Nodi e corrispondenti intervalli di token

le tabelle con chiavi primarie composte memorizzano tutte le righe, che condividono la stessa chiave di partizione in un archivio fisico contiguo. Ciò significa che la ricerca di intervalli di valori delle colonne di clustering all'interno di una singola chiave di partizione è altamente efficiente. Per eseguire questo tipo di ricerca, Cassandra deve solo localizzare l'inizio dell'intervallo su disco e può quindi leggere tutti i risultati a partire da quella posizione. Al contrario, l'interrogazione di righe che si estendono su più chiavi di partizione richiede un'inefficiente scansione casuale del disco per ogni chiave di partizione interrogata.
Tuttavia, il modello di distribuzione dei dati che abbiamo sviluppato finora è una semplificazione del funzionamento di un moderno cluster Cassandra. Nella versione di Cassandra 1.2 sono stati introdotto i nodi virtuali. I nodi virtuali sostituiscono i nodi fisici nell'anello partizionato; ogni nodo virtuale possiede una parte dello spazio del token. I nodi virtuali stessi si trovano nei nodi fisici, ma un nodo fisico non possiede una gamma contigua di nodi virtuali; piuttosto, i nodi virtuali sono distribuiti casualmente tra i nodi fisici. Fondamentalmente, ci sono molti più nodi virtuali che nodi fisici e ogni macchina fisica è responsabile di molti nodi virtuali. Tornando all'esempio precedente del cluster a tre nodi, esaminiamo come vengono distribuiti i dati utilizzando nodi virtuali. Supponiamo di avere dodici nodi virtuali nel cluster - quattro per nodo fisico - anche se in un cluster reale quel numero sarebbe molto più alto. I creatori di Cassandra raccomandano 256 nodi virtuali in un cluster di produzione. La situazione sarà quella mostrata nella figura 2.2.

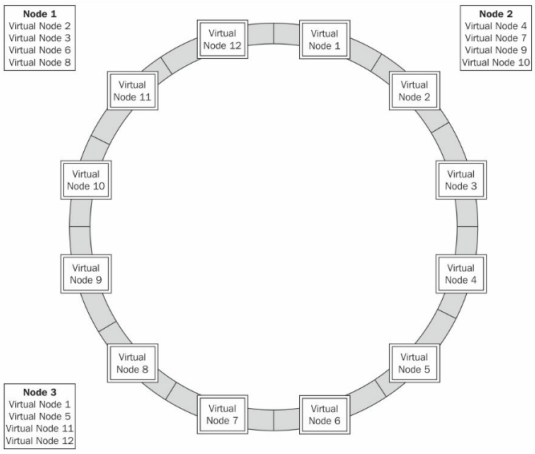

Figura 2.2: Nodi fisici e virtuali

Adesso ogni macchina fisica non sarà più responsabile di un intervallo contiguo di token; ogni macchina è invece responsabile di quattro nodi virtuali, ognuno dei quali copre un diverso intervallo di token.
Il vantaggio principale di avere dei nodi virtuali è il loro comportamento quando il cluster cambia. Supponiamo ad esempio di aggiungere un quarto nodo al nostro cluster a tre nodi. Senza nodi virtuali, il range di token dei tre nodi fisici preesistenti

sarà modificato per fare spazio al quarto nodo: si tratta di un processo noto come riequilibrio, reso superfluo dai nodi virtuali. Quando, infatti, un nuovo nodo si unisce al cluster, gli viene semplicemente assegnata una manciata di nodi virtuali che in precedenza appartenevano ad altre macchine. Piuttosto che ricalcolare direttamente la posizione fisica di ogni singola riga, Cassandra può semplicemente assegnare il numero corretto di nodi virtuali alla nuova macchina - in questo caso, tre e spostare il loro contenuto all'ingrosso. A differenza di uno scenario di riequilibrio, in cui ogni macchina fisica sta perdendo e guadagnando dati, la ridistribuzione dei nodi virtuali richiede solo lo spostamento dei dati dalle tre macchine originali alla sua nuova sede sulla quarta macchina. Ecco come apparirà l'anello:

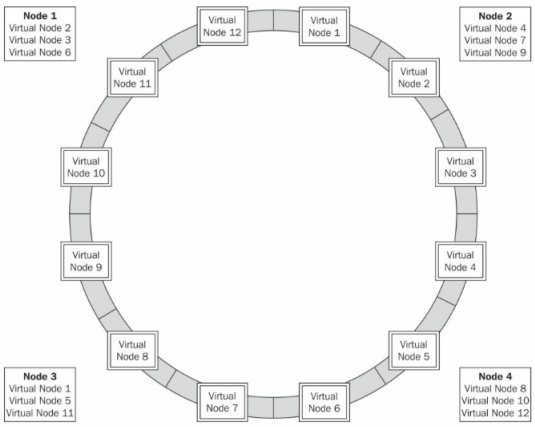

Figura 2.3: Nodi fisici e virtuali

2.1.2 MongoDB

MongoDB (da "humongous", enorme) è un DBMS non relazionale, orientato ai documenti. Classificato come un database di tipo NoSQL, MongoDB si allontana dalla struttura tradizionale basata su tabelle dei database relazionali in favore di documenti in stile JSON con schema dinamico (MongoDB chiama il formato BSON), rendendo l'integrazione dei dati di alcuni tipi di applicazioni più facile e veloce. Oggi MongoDB è un software libero e open source ed è il più popolare database NoSQL [12]. Nel seguito, saranno descritte le caratteristiche principali di MongoDB [4].

Query ad hoc Nonostante il data-model sia molto semplice, MongoDB supporta ricerche per campi, intervalli e regular expression. Le query possono restituire campi specifici del documento e anche includere funzioni definite dall'utente in JavaScript.

Indicizzazione In MongoDB, qualunque campo può essere indicizzato: gli indici secondari sono implementati come B-tree, in modo similare ai tradizionali database relazionali. Questi indici sono ottimizzati per avere una grande varietà di query, inclusi i range scan e le query con ordinamento.

Velocità e persistenza Nell'ambito dei sistemi di database esiste una relazione inversa tra velocità di scrittura e persistenza. La velocità di scrittura consiste nel volume di inserimenti, aggiornamenti e cancellazioni che un database può elaborare in un dato periodo di tempo. La persistenza indica il livello di garanzia con cui l'effetto di una transizione sia memorizzato in modo permanente.
Nel caso del MongoDB, gli utenti possono determinare il trade-off tra velocità e persistenza, scegliendo la semantica di scrittura e decidendo se abilitare o meno il journaling.
Tutte le scritture, di default, sono "fire-and-forget", il che significa che queste scritture sono inviate attraverso un socket TCP senza richiedere una risposta del database. Se gli utenti vogliono una risposta, possono emettere una scrittura usando una modalità sicura: questo forza una risposta, assicurando che la scrittura sia stata ricevuta dal server senza errori. La modalità sicura è configurabile e può anche essere usata per bloccare fino a quando una scrittura non è stata replicata su un certo numero di server. Per dati ad alto volume e basso valore si usa la modalità fire-and-forget, mentre per i dati importanti è preferibile la modalità sicura.
Se il journaling è abilitato, ogni scrittura è committed ad un registro append-only. Se il server viene spento in modo non pulito (ad esempio, in caso di interruzione dell'alimentazione elettrica), il journal sarà utilizzato per garantire che i file dei dati di MongoDB siano ripristinati in uno stato coerente quando si riavvierà il server. Questo è il modo più sicuro per eseguire MongoDB. E' comunque possibile disabilitare il journaling per aumentare le prestazioni.

Scalabilità e replicazione MongoDB, come Cassandra, può scalare orizzontalmente, distribuendo il database su più nodi. Ciò permette di avere prestazioni migliori e la possibilità di minimizzare i problemi in caso di guasto.
Per lo scaling orizzontali, MongoDB sfrutta una tecnica chiamata sharding. L'utente deve scegliere una chiave di sharding, che determina come i dati di una collection saranno distribuiti tra i vari nodi. I dati sono divisi in intervalli (basati sulla chiave di shard) e distribuiti su molteplici shard: infatti MongoDB fornisce la possibilità di replicare il database attraverso i cosiddetti replica-set.
MongoDB funziona con un meccanismo master-slave in cui sul nodo primario è possibile eseguire sia operazioni di lettura che di scrittura, mentre nei nodi secondari solo le operazioni di lettura sono lecite: nel caso in cui il nodo primario si guasta, il cluster sceglierà un nodo secondario e lo promuoverà a master. Un esempio del funzionamento è mostrato nella figura 2.4
Infine MongoDB include un meccanismo di bilanciamento dei dati, che permette di spostare gli intervalli di dati da uno shard troppo carico ad uno shard meno carico, in modo da bilanciare la distribuzione dei dati all'interno del cluster.

Data model

Il data model di MongoDB è document-oriented [4]. Un documento è essenzialmente un insieme di proprietà che sono identificate da un nome a cui è associato un

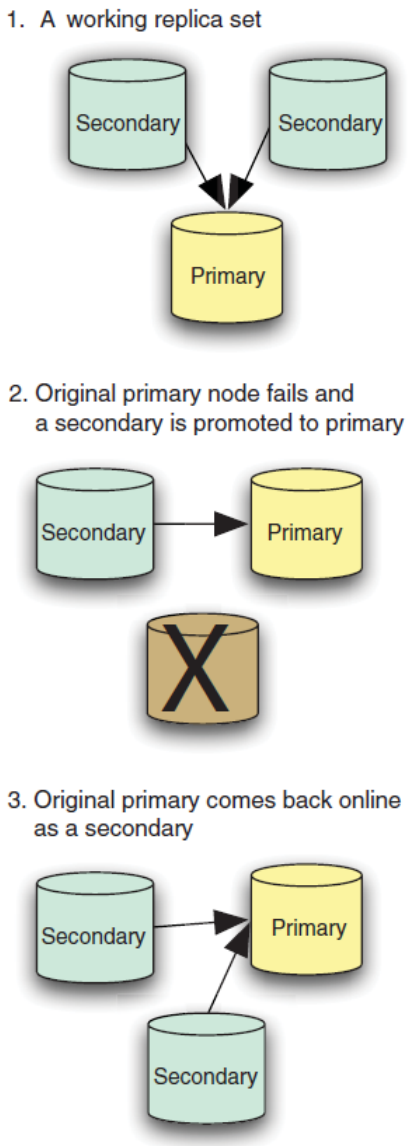

Figura 2.4: Replica set

valore. I valori possono essere semplici tipi di dati, come stringhe, numeri e date oppure arrays o altri documenti JSON: ciò permette ai documenti di rappresentare una grande varietà di strutture dati.
Una delle caratteristiche principali di MongoDB è che i documenti non devono essere conformi ad uno schema predefinito. In un database relazionale, è possibile memorizzare le righe in una tabella, che avrà uno schema rigorosamente definito : se la riga di una tabella ha bisogno di un campo extra, è necessario modificare esplicitamente la tabella. MongoDB invece raggruppa i documenti in collezioni, contenitori che non impongono alcun tipo di schema ed in qualsiasi momento è possibile aggiungere proprietà in modo dinamico. In teoria, ogni documento in una collezione potrà avere una struttura completamente diversa; in pratica i documenti che appartengono ad

una collezione saranno abbastanza uniformi.
Questa mancanza dello schema, conferisce alcuni vantaggi. Il primo è che sarà l'applicazione e non il database a determinare la struttura dei dati: ciò permette di velocizzare lo sviluppo iniziale dell'applicazione. Il secondo è che è possibile rappresentare dati con proprietà variabili: ciò è molto utile quando non si conoscono le proprietà a priori.

2.1.3 MongoDB vs Cassandra

Nei seguenti punti, viene presentata un'analisi delle differenze tra i due DBMS:

- Data Model. MongoDB è un DBMS document-oriented che supporta un object model ricco ed espressivo. Gli oggetti possono avere proprietà e possono essere annidati tra di loro. Questo modello è "orientato agli oggetti" e può facilmente rappresentare qualsiasi struttura dati. Cassandra, di contro, è un DBMS column-oriented, dove i dati vengono meorizzati in righe composte da un numero variabile di colonne. MongoDB è da preferire nel caso in cui non si conosce a priori la struttura del database da realizzare o nel caso in cui si vogliono rappresentare oggetti complessi. Cassandra, invece, richiede la conoscenza iniziale della struttura da sviluppare, sebbene garantisca una maggior flessibilità rispetto ai tradizionali DBMS SQL.

- Disponibilità dei dati. Cassandra supporta un modello "multiple master", in cui se un nodo va down, un altro prenderà il suo posto in modo da rendere il cluster disponibile. MongoDB supporta invece un modello "single master", in cui in caso di problema al master sarà necessario un certo periodo di tempo prima di poter effettuare le operazioni sui vari replica set. Di conseguenza, nel caso in cui è necessaria una disponibilità del 100% è preferibile l'utilizzo di Cassandra, viceversa va bene anche MongoDB.

- Scalabilità. MongoDB, a causa del suo modello "single master" può accettare le scritture solo sul replica set primario. I replica set secondari possono essere usati solo per le letture. Cassandra, invece, grazie al modello "multiple master" consente la scrittura su qualsiasi server e quindi la scalabilità in scrittura è limitata dal numero di server presenti nel cluster: maggior è il numero di server presenti nel cluster, migliore sarà la scalabilità.

2.2 Infrastrutture per i big data: Apache Spark

Apache Spark è un framework opensource per il calcolo distribuito, nato nel 2009 come progetto di ricerca all'universita della California e successivamente diventato un ASF Top-Level Project nel 2014 [2].
Spark consente agli sviluppatori di creare routine per l'elaborazione dei dati complesse e multistage, fornendo un'API di alto livello e un framework fault tolerant che consente ai programmatori di concentrarsi sulla logica piuttosto che su problemi

infrastrutturali o ambientali, come i guasti hardware.
Spark è stato pensato come alternativa all'utilizzo del tradizionale MapReduce di Hadoop, che si è mostrato inadatto per query interattive o in tempo reale e per le applicazioni a bassa latenza. Uno dei principali svantaggi dell'implementazione del MapReduce di Hadoop è la persistenza di dati intermedi sul disco tra la map phase e la reduce phase. Spark supera questi problemi attraverso l'introduzione di una struttura distribuita, fault tolerant, in-memory, basata sugli RDD(Resilient Distribute Dataset). Gli RDD sono distribuiti tra i vari worker nodes del cluster in modo da distribuire l'elaborazione e minimizzare le scritture di dati intermedi sul disco: in questo modo Spark è fino a 100 volte più veloce della corrispondende applicazione MapReduce che lavora sul disco. Inoltre, il riutilizzo da parte di Spark di queste strutture in-memory lo rende adatto ad operazioni iterative, ad operazioni di machine learning e a query interattive.

Versatilità Spark è scritto in Scala ed è eseguito nella Java virtual Machine(JVM). Spark fornisce un supporto nativo per le interfacce di programmazione, tra cui Java, Scala, Python, SQL e R.
Solitamente Spark è utilizzato per elaborare i dati presenti nell'Hadoop File System, tuttavia può essere utilizzato con una moltitudine di altri sistemi come i file system locali, i database relazionali, i database non-relazionali(Cassandra, HBase, Hive, etc.) e le piattaforme di stream processing come Kafka.

2.2.1 Architettura del sistema

Un'applicazione Spark contiene diversi componenti, tutti eseguiti nella Java Virtual Machines(JVM). Ogni componente ha un ruolo specifico nell'esecuzione di un programma Spark: alcuni di questi ruoli sono passivi durante l'esecuzione, come i componenti client, mentre altri ruoli sono attivi nell'esecuzione del programma, tra cui i componenti che eseguono funzioni di calcolo. I principali componenti di un'applicazione Spark sono il driver, il master, il cluster manager e l'esecutore (o gli esecutori). Un esempio dell'architettura è mostrato in figura 2.5.

Driver La vita di un'applicazione Spark inizia e termina attraverso lo Spark driver. Il driver è il processo che i client utilizzano per sottomettere delle applicazioni su Spark. Il driver è, inoltre, responsabile della creazione dello SparkContext, che è l'istanza dell'applicazione che rappresenta la connessione allo Spark master.
Una delle funzioni principali del driver è la pianificazione dell'applicazione: il driver setta l'input dell'applicazione, tutte le trasformazioni (operazioni di manipolazione dei dati) e le azioni richieste (richieste di output o una richiesta di esecuzione del programma) e crea un grafico aciclico diretto (DAG). Un DAG è costituito da task e stage: i task sono la più piccola unità di lavoro schedulabile in un programma Spark, mentre gli stage sono insiemi di task che possono essere eseguiti assieme. Gli stage sono dipendenti l'uno dall'altro. Infine il driver deve restituire lo stato dell'applicazione e/o i risultati al client.

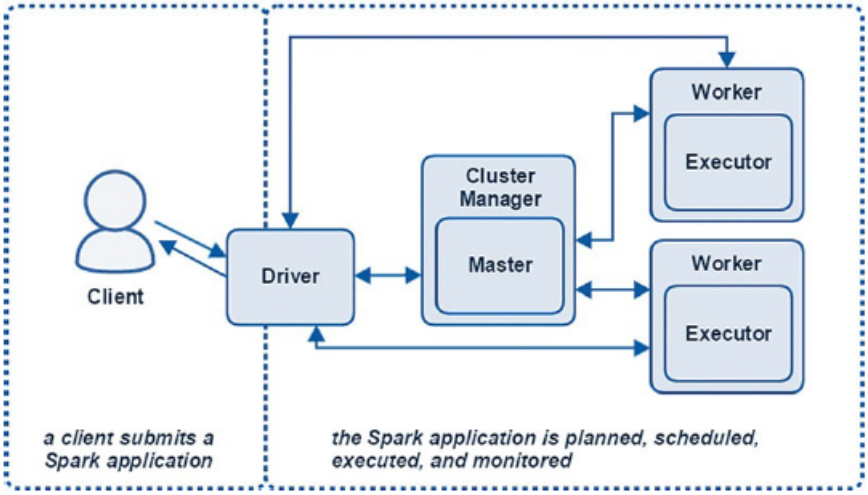

Figura 2.5: Architettura di Spark

Executor e worker Gli Spark executor sono i processi host nei quali i task di un DAG sono eseguiti. Gli executor prenotano una quantità di CPU e di memoria dai worker node dello Spark cluster. Essi sono dedicati ad una specifica applicazione e terminano quando l'applicazione termina. Un executor può eseguire centinaia o migliaia di task all'interno di un programma. Tipicamente ogni nodo del cluster ha un numero finito di executor che possono essere allocati. Worker and executor saranno solamente consapevoli dei task che sono allocati a loro, mentre il driver ha la responsabilità di conoscere l'intero insieme di task e le loro rispettive dipendenze.

Spark Master e Cluster Manager A questo punto abbiamo compreso che lo Spark driver pianifica e coordina l'insieme dei task richiesti da un'applicazione Spark, i task sono eseguiti dagli executor, che sono ospitati dai worker node. Resta da chiarire il ruolo del master e del cluster manager: essi non sono altro che il processo centrale che monitora, setta e alloca le risorse del cluster distribuito in cui i vari executor devono essere eseguiti. Il master e il cluster manager possono essere processi separati o possono essere combinati in un unico processo, nel caso in cui Spark è lanciato in modalità Standalone.

2.2.2 RDD: Resilient Distributed Dataset

Gli RDD sono la più importante struttura dati utilizzata nella programmazione Spark: la maggior parte delle operazioni effettuate in Spark consiste nel creare nuovi RDD eseguendo delle operazioni sugli RDD già esistenti. Sebbene vi sia la possibilità di memorizzare gli RDD permanentemente sul disco, gli RDD sono solitamente pensati per la memorizzazione nella memoria.
Il termine Resilient Distribute Dataset permette di descrivere i seguenti concetti:

- Resilient: gli RDD sono resilienti. Ciò significa che se si guasta un nodo che sta eseguendo un operazione, il dataset può comunque essere ricostruito. Ciò è possibile perchè Spark conosce la sequenza di operazioni necessarie per ricreare l'RDD.

- Distributed: gli RDD sono distribuiti. Ciò significa che i dati presenti negli RDD sono divisi in una o più partizioni e distribuite come collezioni di oggetti tra i vari worker node del cluster. Gli RDD forniscono un modo efficiente per condividere la memoria in modo che sia possibile scambiare dati tra i vari processi(executor) presenti nei vari nodi(worker): ciò da la possibilità di avere un riuso efficiente dei dati, utile nelle operazioni iterative per superare i problemi presenti nel mapReduce di Hadoop.

- Dataset: gli RDD sono dataset composti di un insieme di record. I record sono identificabili univocamente all'interno di un dataset.

Gli RDD sono partizionati in modo tale che ogni partizione contenga un unico insieme di record e possa operare in modo indipendente. Un'altra proprietà chiave degli RDD è la loro immutabilità, che significa che dopo che sono stati instanziati e popolati con i dati, non possono essere aggiornati: a partire da un RDD, potranno essere creati nuovi RDD attraverso l'esecuzione delle trasformazioni. Le azioni invece permettono di ottenere i dati di output a partire dagli RDD.

2.3 Tool per lo sviluppo di applicazioni Web

In questa sezione, vengono illustrate le principali caratteristiche di due tool per lo sviluppo di applicazioni web, quali Apache Zeppelin e Apache HTTP Server.

2.3.1 Apache Zeppelin

Apache Zeppelin è un "notebook" open-source, web-based, che consente un'analisi interattiva dei dati e dei documenti. Il notebook è integrato con sistemi di elaborazione dati distribuiti come Apache Spark, Cassandra, MongoDB e tanti altri. Apache Zeppelin consente di creare documenti interattivi e data-driven con SQL, Scala, R o Python direttamente nel browser [9].

2.3.2 Apache server

Il progetto Apache HTTP Server Project consiste nello sviluppo e il mantenimeto di un server HTTP open-source per i moderni sistemi operativi, inclusi UNIX e Windows. L'obiettivo di questo progetto è quello di fornire un server sicuro, efficiente ed estensibile che fornisca servizi HTTP adatti per gli attuali standard HTTP. L'Apache HTTP Server è un progetto dell'Apache software foundation [1].

Funzionamento del server web Apache

Sebbene chiamato Apache web server, non è un server fisico, ma piuttosto un software eseguito su un server. Il suo compito è quello di stabilire una connessione tra un server e i browser dei visitatori del website attraverso un modello client-server. Quando un visitatore vuole caricare una pagina, il suo browser invia una richiesta al server e Apache restituisce una risposta con tutti i file richiesti(testo, immagini, ecc.). Il server e il client comunicano attraverso il protocollo HTTP e Apache è responsabile per una comunicazione facile e sicura tra le 2 macchine.
Apache è altamente personalizzabile, grazie alla sua struttura module-based. I moduli permettono agli amministratori del server di aggiungere e rimuovere le funzionalità addizionali. Apache ha dei moduli per la sicurezza, per il caching, per la riscrittura degli URL, per la password authentication e altri.

Pro e contro

I vantaggi dell'utilizzo di Apache sono i seguenti [3]:

- Open-source e free, anche per uso commerciale;
- Software stabile e affidabile;
- Aggiornato frequentemente, con patch di sicurezza;
- Flessibilità, grazie alla struttura basata sui moduli;
- Facile da configurare;
- Cross-Platform;
- Supporto in caso di problema, grazie alla enorme community.

I punti deboli di Apache sono:

- Problemi di performance nei website estremamente trafficati;
- Possibili problematiche di sicurezza dovute all'alto numero di opzioni configurabili.

2.4 Docker

Il software Docker è una tecnologia di containerizzazione che consente la creazione e l'utilizzo dei container Linux: Docker considera i container come macchine virtuali modulari estremamente leggere, offrendo la flessibilità di creare, distribuire, copiare e spostare i container da un ambiente all'altro. [8]

2.4.1 Come funziona Docker?

La tecnologia Docker utilizza il kernel di Linux e le sue funzionalità, come Cgroups e namespace, per isolare i processi in modo da poterli eseguire in maniera indipendente. Questa indipendenza è l'obiettivo dei container: la capacità di eseguire più processi e applicazioni in modo separato per sfruttare al meglio l'infrastruttura esistente pur conservando il livello di sicurezza che sarebbe garantito dalla presenza di sistemi separati.
Gli strumenti per la creazione di container, come Docker, consentono il deployment a partire da un'immagine. Ciò semplifica la condivisione di un'applicazione o di un insieme di servizi, con tutte le loro dipendenze, nei vari ambienti. Docker automatizza anche la distribuzione dell'applicazione (o dei processi che compongono un'applicazione) all'interno dell'ambiente containerizzato.
Gli strumenti sviluppati partendo dai container Linux, responsabili dell'unicità e della semplicità di utilizzo di Docker, offrono agli utenti l'accesso alle applicazioni, la capacità di eseguire un deployment rapido e il controllo sulla distribuzione di nuove versioni.
La tecnologia Docker è stata sviluppata partendo dalla tecnologia LXC, quella che in molti associano ai container Linux "tradizionali", discostandosi però fin da subito da tale dipendenza. La tecnologia LXC era una soluzione di virtualizzazione ottimizzata, ma non offriva un'esperienza ottimale a sviluppatori e utenti. La tecnologia Docker non offre solo la capacità di eseguire i container, ma consente, ad esempio, di semplificare il processo di creazione e costruzione dei container, di invio e di versioning delle immagini. I container Linux tradizionali adottano un sistema init in grado di gestire più processi, consentendo di eseguire più applicazioni come una singola entità. La tecnologia Docker agevola la suddivisione delle applicazioni nei loro vari processi e mette a disposizione gli strumenti per farlo: questo approccio granulare ha svariati vantaggi.

2.4.2 I vantaggi di Docker

- Modularità. L'approccio Docker alla containerizzazione si basa sulla capacità di estrarre i singoli componenti di un'applicazione, da aggiornare o riparare. Oltre a questo approccio basato sui microservizi, è possibile condividere i processi tra più applicazioni in modo molto simile a quello usato dalla Service-Oriented Architecture (SOA).

- Strati e controllo delle versioni delle immagini. Ogni file immagine Docker è composto da più strati, che insieme costituiscono una singola immagine. Uno strato viene creato ad ogni modifica dell'immagine. Ogni volta in cui un utente specifica un comando (es. run o copy), viene creato un nuovo strato. Docker riutilizza questi strati per velocizzare il processo di creazione dei container. Le modifiche sono condivise tra le immagini, migliorando ulteriormente la velocità, la dimensione e l'efficienza. Il controllo delle versioni fa parte del processo di stratificazione. Ogni volta che viene apportata una modifica, si

crea sostanzialmente un registro delle modifiche incorporato, che garantisce un controllo completo sulle immagini del container.

- Rollback. Uno dei maggiori vantaggi della stratificazione è la capacità di eseguire il rollback. Ogni immagine è composta da strati. Se l'iterazione di un'immagine non è soddisfacente, è possibile riportala alla versione precedente. Ciò consente uno sviluppo agile e aiuta a ottenere l'integrazione e il deployment continui.

- Deployment rapido. In passato, la configurazione, l'esecuzione e il provisioning di un nuovo hardware richiedevano giorni e notevoli investimenti. I container basati su Docker possono ridurre la durata del deployment a pochi secondi. Creando un container per ogni processo, puoi condividere con rapidità i processi simili con le nuove applicazioni. Poiché non è necessario riavviare un sistema operativo per aggiungere o spostare un container, i tempi per il deployment sono sostanzialmente più brevi. Inoltre, grazie alla velocità del deployment, attraverso i container è possibile creare ed eliminare dati in modo sicuro, semplice ed economico.

Capitolo 3

Architettura del sistema

L'obiettivo di questo capitolo è quello di descrivere la fase di progettazione concettuale dell'architettura. Successivamente saranno descritte le due customizzazioni dell'architettura in funzione dei due casi di studio in cui saranno utilizzate.

3.1 Progettazione concettuale

Il primo step da effettuare per la progettazione dell'architettura consiste nell'identificazione dei blocchi logici da cui deve essere costituita, ognuno dei quali avrà una funzione specifica.
Il blocco di partenza ha il compito di effettuare la raccolta dei dati. I dati sono raccolti a partire da applicazioni di tipo data-intensive, in un contesto che può essere o non essere industry 4.0.
Dopo aver raccolto i dati, il secondo blocco ha la funzionalità di data ingestion. Con il nome data ingestion si intende il processo di importazione dell'enorme mole di dati all'interno dell'architettura.
Il terzo blocco è rappresentato dal DBMS: il DBMS dovrà essere altamente scalabile, quindi sarà di tipo NoSQL. Esso dovrà essere progettato in modo da permettere la memorizzazione dei dati raccolti dalle applicazioni data-intensive e dei risultati ottenuti dall'elaborazione degli stessi.
Il quarto blocco ha il compito di effettuare l'elaborazione dei dati per il calcolo dei KPI. Esso leggerà i dati su cui effettuare l'elaborazione dal DBMS e successivamente scriverà i risultati ottenuti sullo stesso DBMS.
Infine, l'ultimo blocco ha il compito di mostrare i risultati ottenuti, attraverso delle dashboard informative navigabili sul Web e interattive.
Nella figura 3.1 sono rappresentati i principali blocchi logici da cui dovrà essere composta l'architettura.
Dell'architettura appena descritta, saranno create due istanze:

- La prima per l'analisi dell'efficienza produttiva in un contesto Industry 4.0;
- La seconda per l'analisi dell'efficienza produttiva degli afferenti ad un dipartimento universitario.

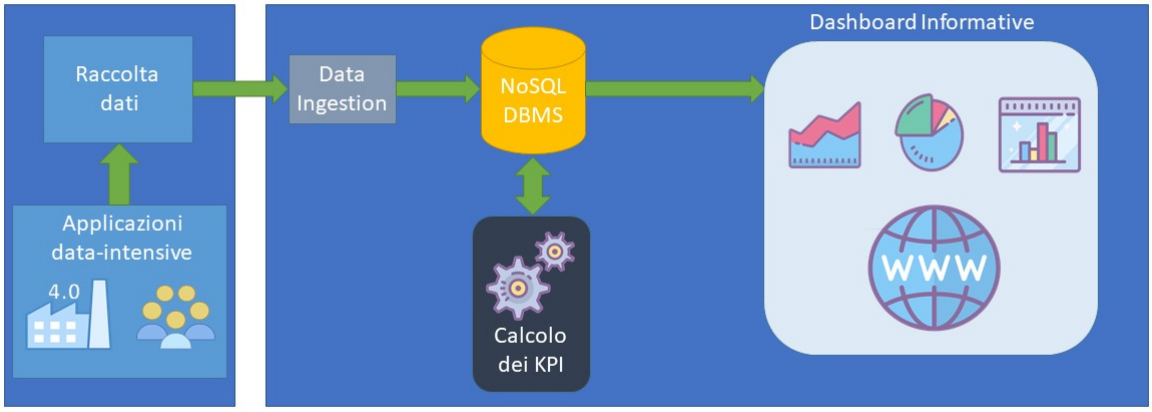

Figura 3.1: Schema a blocchi dell'architettura

Le architetture reali sviluppate saranno descritte nelle sezioni 3.3 e 3.4. Nella sezione 3.2 sarà invece descritta la piattaforma Docker, utilizzata per effettuare il deployment di entrambe le istanze dell'architettura.

3.2 Realizzazione dell'architettura

L'architettura del sistema dovrà essere sviluppata attraverso la piattaforma Docker. Come descritto nel capitolo 2, Docker è un sistema per il deployment e la gestione dei container: esso permette infatti di creare ed eseguire applicazioni come container isolati.

Motivazioni La scelta del sistema Docker è motivata da una serie di ragioni.
In primo luogo, Docker permette di effettuare un deployment delle applicazioni molto più rapido rispetto ai processi di configurazione ed installazione tradizionali.
In secondo luogo, sebbene concettualmente sembri simile all'idea di virtual machine, in realtà è profondamento diverso. Infatti, ogni macchina virtuale ha uno strumento chiamato hypervisor che si preoccupa di riservare un certo quantitativo di risorse dal sistema operativo host: in tal modo l'operazione è molto costosa ed il sistema operativo guest sarà completamente isolato dal sistema host. I container, invece, hanno un isolamento molto più leggero, in quanto condividono tutti il kernel del sistema operativo sottostante. Il vantaggio che ne consegue è che è un sistema molto più scalabile rispetto alle macchine virtuali tradizionali.

Infine, attraverso la piattaforma Docker Hub è molto facile trovare sia l'immagine dell'applicazione di cui si ha bisogno che il supporto necessario per effettuare eventuali modifiche all'immagine di base.

Docker-compose Lo step immediatamente successivo è la creazione del file Docker-compose, in formato yml. Questo file facilita la vita del sistemista in quanto permette di installare, cancellare, lanciare o terminare tutti i container descritti al suo interno postponendo rispettivamente le parole "up", "down", "start" e "stop" al comando docker-compose, da shell di comando. Ad esempio per lanciare i container è sufficiente scrivere, su shell di comando, "docker-compose start".
Per ogni container indicato nel docker-compose, possono essere specificate delle impostazioni aggiuntive come ad esempio la porta attraverso cui l'applicazione sarà raggiungibile dall'esterno oppure è possibile effettuare il mounting di una cartella presente nel sistema operativo sottostante all'interno del filesystem del container stesso.

3.3 Architettura sviluppata per l'analisi dell'efficienza produttiva in un contesto Industry 4.0

La prima istanza dell'architettura è sviluppata per l'analisi dell'efficienza produttiva di una linea di produzione dell'industria farmaceutica Procemsa. Il progetto è effettuato insieme ad altri partner che si occupano delle fasi di raccolta dei dati e data ingestion. Il focus, in questa sezione, sarà quindi incentrato sulle parti dell'architettura di cui mi sono occupato personalmente.

Memorizzazione dei dati Il DBMS scelto per la memorizzazione dei dati è Cassandra. Tale scelta è motivata dal fatto che i dati sono ben strutturati ed è quindi semplice stabilire la struttura del keyspace prima dello sviluppo dell'applicazione. In secondo luogo, Cassandra è un database altamente scalabile sia in scrittura che in lettura ed ha una disponibilità pari al 100%.

Elaborazione dei dati I dati relativi al caso di studio Procemsa saranno elaborati attraverso il framework Spark. Spark è l'unico componente installato manualmente senza l'ausilio di Docker. Il compito di Spark sarà quello di leggere i dati da Cassandra, elaborarli in modo semplice e scalabile grazie all'utilizzo degli RDD e scrivere i risultati nuovamente in Cassandra.
Al fine di utilizzare Spark per la lettura e la scrittura di dati da Cassandra, è stato sfruttato lo Spark-Cassandra-connector realizzato da Datastax

Visualizzazione dei risultati Il tool di visualizzazione grafica dei risultati scelto nel caso di studio Procemsa è Apache Zeppelin. Tale scelta è motivata, principalmente, da 3 ragioni:

- Zeppelin è un sistema open-source, a differenza di altri software simili come Tableau.
- Zeppelin è molto semplice da usare e permette di effettuare le query attraverso il linguaggio CQL. Inoltre, è possibile usare Spark e Spark SQL per l'elaborazione dei dati e per l'interrogazione del DB.
- Zeppelin contiene di default l'interpreter per utilizzare Cassandra o MongoDB.

Attraverso Apache Zeppelin è stata quindi creata una dashboard riassuntiva, all'interno della quale sono stati rappresentati i vari grafici che mostrano l'andamento dei KPI.
Un esempio grafico dell'architettura realizzata è mostrato in figura 3.2.

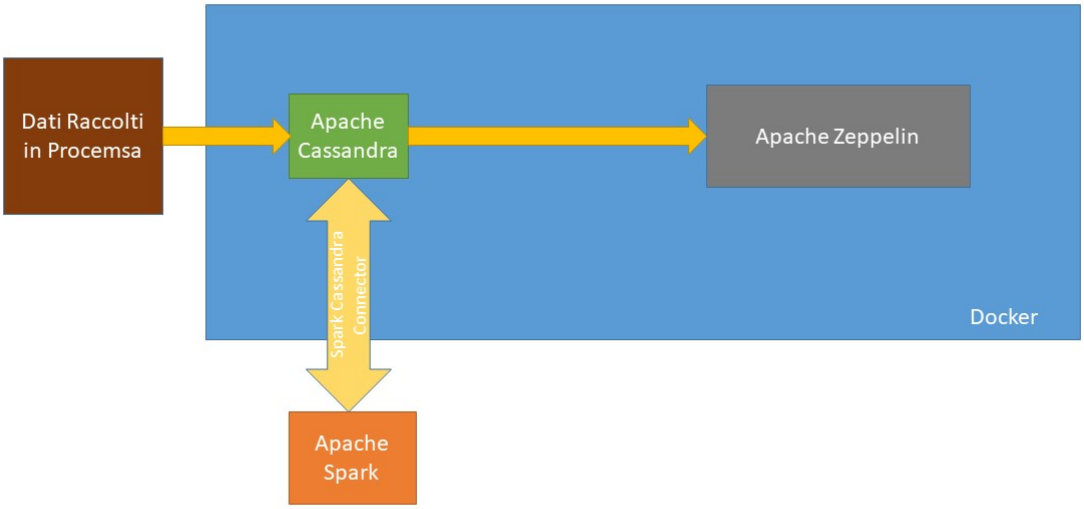

Figura 3.2: Schema a blocchi dell'architettura realizzata nel caso di studio Procemsa

3.4 Architettura sviluppata per l'analisi dell'efficienza produttiva degli afferenti ad un dipartimento universitario

La seconda istanza dell'architettura è sviluppata per l'analisi dell'efficienza produttiva degli afferenti al dipartimento di automatica ed informatica del Politecnico di Torino. Anche in questo caso la fase di raccolta dei dati è effettuata da altri, mentre la fase di data ingestion è momentaneamente effettuata in modo manuale: in un futuro prossimo questa fase sarà automatizzata attraverso l'utilizzo di uno script.

Memorizzazione dei dati Il DBMS scelto in questo caso è MongoDB. Ciò è dettato dal fatto che la struttura dei dati è abbastanza complessa ed è stata definita durante il processo di sviluppo dell'applicazione. Un'ulteriore motivazione è che la quasi totalità delle operazioni da eseguire sul database sono operazioni di lettura ed è noto dal capitolo 2 che MongoDB garantisce un'ottima scalabilità in lettura.

Elaborazione dei dati I dati del dipartimento di automatica ed informatica sono invece elaborati attraverso un back-end ed un front-end presenti all'interno dell'Apache HTTP Server. Il back-end è stato realizzato in linguaggio PHP ed ha il compito di esporre i dati presenti in MongoDB ed effettuarne una prima elaborazione. Il front-end è realizzato, invece, in linguaggio Javascript ed interrogherà le API esposte dal back-end per ottenere i dati ed effettuare un'elaborazione più complessa grazie anche all'ausilio della libreria linq.js .

Visualizzazione dei risultati Per la rappresentazione grafica dei KPI , è stata realizzata, invece, una dashboard interattiva navigabile sul web utilizzando il front-end Javascript già citato in precedenza con l'ausilio di varie librerie quali ApexCharts o Chart.js.
Un esempio grafico dell'architettura realizzata è mostrato in figura 3.3.

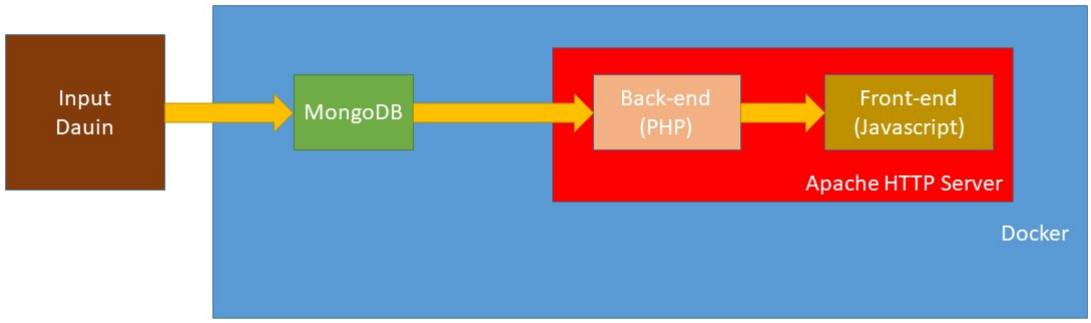

Figura 3.3: Schema a blocchi dell'architettura realizzata nel caso di studio Dauin

Capitolo 4

Analisi dell'efficienza produttiva in un contesto Industry 4.0

L'obiettivo di questo capitolo è quello di descrivere l'analisi dell'efficienza produttiva nel caso di studio Procemsa. Il KPI ritenuto più significativo per un analisi dell'andamento delle prestazioni è l'OEE(Overall Equipment Effectiveness).
L'obiettivo di questo capitolo è quello di spiegare cosa è l'OEE e come è stato calcolato in Procemsa. In particolar modo, saranno descritte le varie formule utilizzabili per il calcolo dell'OEE ed il significato dei vari fattori attraverso cui si ottiene l'OEE. Successivamente saranno dettagliate le fasi di raccolta dei dati, di calcolo dell'OEE e di rappresentazione grafica dei risultati ottenuti per ognuna delle 2 macchine che sono state analizzate.

4.1 OEE: overall equipment effectiveness

L'OEE(Overall Equipment Effectiveness), letteralmente "efficienza generale dell'impianto", è un indicatore percentuale che rappresenta il rendimento globale di una risorsa produttiva o di un insieme di risorse, siano esse umane o tecniche, durante il tempo nel quale queste sono disponibili a produrre. L'OEE è l'indicatore più "esigente" ed omnicomprensivo che esista, in quanto considera tutte le tipologie di inefficienze che portano ad una minore produttività: dalla mancanza di materiali alla cattiva pianificazione, dai setup ai tempi morti, dalle microfermate ai guasti, dalle rilavorazioni alle non conformità [11].

4.1.1 Fattori e calcolo

In questa sottosezione saranno descritte le due modalità per il calcolo dell'OEE [18].

All time e tempo di produzione pianificato Il primo punto da chiarire è la differenza tra "all time" e il tempo di produzione pianificato. L' "all time", detto anche "tempo 24/7", è il tempo che include tutti i minuti di tutti i giorni.
Da questo valore bisogna sottrarre la perdita pianificata, ovvero tutto il tempo

Figura 4.1: All time

che deve essere escluso dall'analisi dell'OEE perchè non vi è intenzione di produrre durante questo periodo(fermi impianto, pranzi, interruzioni, periodi in cui non vi sono ordini). Il tempo rimanente è il tempo di produzione pianificato: l'OEE deve analizzare tutte le perdite di efficienza e di produttività che vi sono in questo periodo, con l'obiettivo di ridurle o eliminarle.

Figura 4.2: tempo di produzione pianificato

I 3 fattori dell'OEE I 3 fattori da prendere in considerazione per il calcolo dell'OEE sono i seguenti:

- Disponibilità: indica la percentuale dell'effettivo tempo di attività rispetto al tempo disponibile. Tiene conto di tutti gli eventi che fermano la produzione per un periodo di tempo non trascurabile. Un esempio di eventi che causano perdita di disponibilità sono gli stop non programmati, come un guasto all'impianto o la carenza di materiale, o gli stop programmati, come il tempo di conversione. Sottraendo dal tempo di produzione pianificato il tempo perso a causa delle perdite di disponibilità, si ottiene il tempo di esecuzione.

- Rendimento: indica la percentuale di parti prodotte rispetto alla potenzialità teorica, quando l'impianto è attivo. Questo parametro tiene conto di tutto ciò(inclusi i piccoli stop e i cicli lenti) che rende l'impianto meno veloce rispetto alla massima velocità che potrebbe avere. Le perdite di rendimento possono essere causate da usura della macchina, materiali scadenti, inceppamenti delle macchine. Sottraendo dal tempo di esecuzione il tempo perso a causa delle perdite di rendimento, si ottiene il tempo di esecuzione netto.

- Qualità: indica la percentuale di parti conformi rispetto al totale delle parti prodotte. Questo parametro valuta le parti prodotte che non rispettano gli standard di qualità. Sottraendo dal tempo di esecuzione netto il tempo perso a causa delle perdite di qualità, si ottiene il tempo pienamente produttivo.

Figura 4.5: Tempo pienamente produttivo

Questi tre fattori includono quelle che sono definite le "sei maggiori perdite", ovvero le cause più comuni che causano una perdita di produttività [14]. Le perdite di

Figura 4.3: Tempo di esecuzione

Figura 4.4: Tempo di esecuzione netto

disponibilità includono i guasti e i tempi di setup; le perdite di rendimento includono gli arresti dovuti a piccoli inconvenienti e la ridotta velocità di lavorazione; le perdite di qualità tengono in considerazione la resa ridotta, gli scarti e le rilavorazioni.

Calcolo semplice Il modo più semplice per calcolare l'OEE è come rapporto tra il tempo pienamente produttivo e il tempo di produzione pianificato. Il tempo pienamente produttivo non significa altro che produrre parti buone il più velocemente possibile senza mai fermarsi. Da ciò ne consegue che:

$$OEE = \frac{\text{Parti buone} \cdot \text{Tempo ideale di ciclo}}{\text{Tempo di produzione pianificato}}$$

Tuttavia, sebbene sia un metodo totalmente valido per il calcolo dell'OEE, non fornisce informazioni relativamente ai tre fattori legati alla perdita: disponibilità, rendimento e qualità. Useremo quindi un calcolo alternativo.

Calcolo preferito Il metodo preferito consiste nel calcolare l'OEE attraverso il prodotto dei tre fattori:

$$OEE = \text{Disponibilità} \cdot \text{Rendimento} \cdot \text{Qualità}$$

A questo punto bisogna stabilire come ottenere i valori di disponibilità, rendimento e qualità.
Per calcolare la disponibilità, dobbiamo in primis calcolare il tempo di esecuzione: esso è uguale al tempo di produzione pianificato meno il tempo di arresto, che, a sua volta, è uguale alla somma di tutti gli intervalli di tempo in cui il processo era destinato ad essere in esecuzione, ma non lo è stato a causa di stop programmati o non programmati.

$$\text{Tempo di esecuzione} = \text{Tempo di produzione pianificato} - \text{Tempo di arresto}$$

A questo punto è possibile calcolare la disponibilità, come rapporto tra il tempo di esecuzione e il tempo di produzione pianificato.

$$\text{Disponibilità} = \frac{\text{Tempo di esecuzione}}{\text{Tempo di produzione pianificato}}$$

Il rendimento può essere calcolato nel modo seguente:

$$\text{Rendimento} = \frac{\text{Tempo netto di esecuzione}}{\text{Tempo di esecuzione}}$$

oppure, considerando che il tempo ideale di ciclo moltiplicato per il numero di parti totali è uguale al tempo netto di esecuzione:

$$\text{Rendimento} = \frac{\text{Tempo ideale di ciclo} \cdot \text{Numero di parti totali}}{\text{Tempo di esecuzione}}$$

Il rendimento non può mai essere superiore al 100%.
Infine, la qualità può essere calcolata come rapporto tra numero di parti buone e il numero di parti totali.

$$\text{Qualità} = \frac{\text{Numero di parti buone}}{\text{Numero di parti totali}}$$

Si preferisce calcolare l'OEE con questo secondo metodo, in quanto l'OEE fornisce una visione d'insieme di quanto sia efficace il processo di produzione, ma non fornisce un'analisi relativa alle cause alla base della perdita di produttività: questo è il ruolo di disponibilità, rendimento e qualità.

Granularità del calcolo Usualmente è conveniente usare la misura più granulare che esista: questo è il modo attraverso cui avremo le informazioni più dettagliate e precise relativamente alle perdite. L'alternativa è usare la misura che è più facilmente comprensibile e significativa per il team [10].
Per quanto riguarda il tempo è possibile scegliere qualsiasi periodo si preferisce. Comunemente si sceglie un periodo basato sul ciclo di lavoro della fabbrica(ad esempio 8 ore o 480 minuti). Altre possibilità sono il calcolo per un giorno, per un compito o eventualmente un monitoraggio continuo [17]. Un'indagine in corso sulle aziende manifatturiere indica che le migliori aziende del settore non solo sono più vigili e persistenti nei loro sforzi di misurazione, ma monitorano e misurano le loro prestazioni più frequentemente. L'indagine indica che le misure operative come l'OEE, per tali aziende, sono monitorate quotidianamente, come minimo, al fine di attivare efficaci azioni correttive o preventive [15].

Calcolo dell'OEE per l'intero impianto Il calcolo dell'OEE per un intero impianto può essere utile per monitorare le tendenze (ad esempio se un dato impianto sta migliorando). Tuttavia, si dovrebbe essere molto cauti nell'uso dell'OEE per confrontare differenti impianti, prodotti o beni. A meno che non si producano prodotti identici su apparecchiature identiche in condizioni identiche, non ha senso confrontare i punteggi OEE. Non è possibile calcolare un'OEE aggregata per un impianto, ma solo una media [13]: la via più semplice consiste nel calcolare la media normale dell'OEE. L'alternativa consiste nel calcolo di una media pesata [17] [10].

Calcolo dell'OEE per linea di produzione Se voglio uno score OEE per l'intera linea di produzione, dove devo calcolarlo? L'OEE deve essere misurato nella fase che rappresenta un collo di bottiglia per il processo: infatti è questa fase che determina l'output del sistema. Se abbiamo un sistema in cui tutta l'attrezzatura è bilanciata

per lavorare allo stessa velocità, allora la prassi è monitorare l'OEE sull'attrezzatura che effettua l'attività principale. Se in seguito ai miglioramenti, l'attrezzatura che prima rappresentava il collo di bottiglia non lo è più, allora bisogna spostare il calcolo dell'OEE sulla attrazzatura che rappresenta la nuova limitazione. Se la limitazione si spostasse nella produzione di prodotti differenti, sarebbe corretto cambiare anche il punto in cui misuriamo l'OEE [10].

4.1.2 Interpretare l'OEE

Dopo aver capito a cosa serve l'OEE e come si può calcolare, bisogna capire come intrepretare il risultato ottenuto. L'obiettivo World-Class, secondo il libro "Introduction to TPM" di Seiichi Nakajima, è quello di ottenere un'OEE dell'85%, grazie ad una disponibilità del 90%, ad un rendimento del 95% e ad una qualità del 99% [18]. Tuttavia lo studio di Nakajima è relativo ad un paese particolare(il Giappone), ad un particolare periodo storico(gli anni '70) e ad una particolare industria(quella automobilistica). Oggi la maggior parte delle aziende ha un'OEE del 60%. Raggiungere la condizione ideale del 100% è virtualmente impossibile, in quanto rappresenterebbe un sistema che non si ferma mai e che non effettua mai attrezzaggi/setup. Se l'OEE risultasse maggiore del 100%, anzi, sarebbe sintomo di inaccuratezza del modello impostato (ad esempio, tempi standard sovradimensionati e quindi inesatti). Anche valori alti (maggiori del 70%), se rilevati in contesti che non hanno mai affrontato un processo strutturato di miglioramento dell'efficienza, devono essere validati approfonditamente [11].

4.2 Data Ingestion

I dati sono stati forniti, per entrambe le macchine, tramite un foglio di lavoro Excel. Quest'ultimo è stato convertito in un documento in formato csv, contenente solo i campi necessari per il calcolo dell'OEE. Il documento in formato csv è quindi caricato su una tabella nel DBMS Cassandra.

4.3 Descrizione dello scenario

L'analisi dell'OEE è stata effettuata sulla linea Pluridose, all'interno della quale sono state considerate 2 macchine: una tappatrice, sita nel reparto di imbottigliamento, che ha il compito di controllare il riempimento del barattolo e di chiudere il barattolo; nel reparto di confezionamento, che si trova a valle rispetto al precedente, vi è, invece, un'etichettatrice SL200.
Tra le 2 macchine, vi è un serbatoio che funge da buffer, in modo tale che in caso di fermo macchina di una delle 2 macchine, l'altra possa continuare il proprio compito: la tappatrice sarà, comunque, costretta a fermarsi nel caso in cui l'etichettatrice resta ferma per un periodo di tempo tale per cui il serbatoio diventa pieno, mentre l'etichettatrice si fermerà nel caso in cui il serbatoio resta vuoto. E' possibile vedere

uno schema esemplificativo nella figura 4.6.

Lungo la linea sono stati installati vari sensori, in grado di monitorare il com-

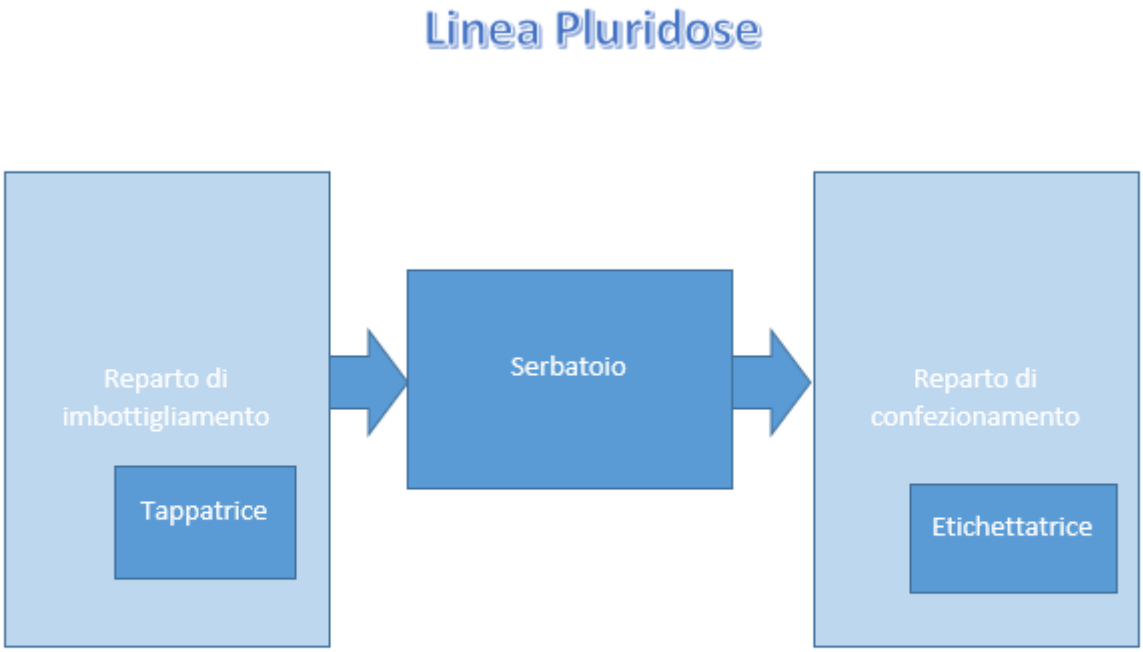

Figura 4.6: Schema linea pluridose

portamento dei macchinari, analizzando i segnali elettrici da essi derivati. I dati ricavati da questi sensori vengono salvati in tempo reale all'interno di un database, sviluppato da Orchestra, che supporta API REST per la lettura e la scrittura dei dati al suo interno.

La corrispondenza tra i segnali fisici e quelli campionati è stata confermata a macchina ferma o a ciclo ridotto, mentre l'acquisizione deve essere tarata con il segnale fisico con la macchina in moto.

4.4 Descrizione dei dati: etichettatrice

I dati raccolti dai sensori presenti nell'etichettatrice sono relativi al periodo tra il 24 Aprile 2018 e il 2 Maggio 2018 e sono stati forniti tramite un foglio di lavoro Excel. Nel periodo di studio sono stati prelevati 31368 campioni.
Per ogni campione sono presenti i seguenti dettagli:

- Timestamp: indica la data e l'orario in cui è stato acquisito il campione;
- Carico minimo generico
- Termico intervenuto/allarme generico: indica se si ha un allarme generico;
- Contatore aggregato dei pezzi in ingresso nella macchina;
- Contatore aggregato dei pezzi in uscita dalla macchina;
- Velocità macchina.

I dati sono stati acquisiti in parte in modalità "Time Based" ogni 20 secondi ed in parte in modalità "Event Based" con trigger sugli allarmi.

4.5 Preprocessing dei dati: etichettatrice

Lo step successivo, alla raccolta dei dati, è stato quello di definire la granularità di calcolo dell'OEE e capire quali dei dati presenti per ogni campione sono utili per il calcolo dell'OEE.

Granularità di calcolo Al fine di dettagliare con maggior precisione l'efficienza della macchina, si è scelto di calcolare l'OEE dinamico in 2 modalità:

- calcolo istantaneo: consiste nel calcolo per ogni intervallo di 10 minuti;
- calcolo aggregato: consiste in un calcolo per ogni intervallo di 10 minuti a partire dall'inizio della giornata.

Questa scelta permette di avere sia un dato istantaneo, relativo al singolo intervallo, che indica l'efficienza dei 10 minuti presi in considerazione, sia di avere un dato aggregato, che permette di capire quanto la macchina sia stata efficiente in quella giornata fino all'intervallo considerato. Ciò significa che il dato aggregato dell'ultimo intervallo di tempo di una giornata coincide con l'OEE relativo a quella stessa giornata.

Estrazione dei dati Dopo aver stabilito la granularità di calcolo, bisogna estrarre da ogni campione le misure realmente necessarie per calcolare l'OEE.
Nel capitolo 2, è stato spiegato che l'OEE si può ottenere tramite la seguente formula:

$$OEE = \text{Disponibilità} \cdot \text{Rendimento} \cdot \text{Qualità}$$

La disponibilità può essere ottenuta come rapporto tra tempo di esecuzione e tempo di produzione pianificato. Nel calcolo istantaneo il tempo di produzione pianificato è uguale a 10 minuti, tranne per il primo e l'ultimo intervallo della giornata la cui durata può essere inferiore ai 10 minuti. Nel calcolo aggregato, il tempo di produzione pianificato è uguale al tempo trascorso tra il primo campione della giornata e il termine dell'intervallo considerato.
Per ottenere il tempo di esecuzione è necessario calcolare il tempo di arresto, che a sua volta corrisponde alla somma degli intervalli di tempo in cui la macchina è ferma. Per calcolare il tempo di arresto, in un primo momento, era stato ipotizzato di usare il campo "allarme generico" dei vari campioni. Tuttavia, l'unica informazione restituita dal campo "allarme generico" è un bit che indica se la macchina è ferma o meno nel timestamp corrispondente, senza specificare l'eventuale durata dello stop. Per questo motivo, è stata pensata una via alternativa per il calcolo dei tempi di arresto che consiste nel considerare ferma la macchina, nel caso in cui ho più campioni consecutivi in cui il contatore dei pezzi prodotti resta invariato: il tempo di arresto sarà uguale all'intervallo di tempo tra l'ultimo campione e il primo campione in cui il contatore resta invariato. Nel calcolo instantaneo dovrò sommare i tempi di arresto relativi al singolo intervallo, mentre nel calcolo aggregato dovrò sommare i tempi di arresto a partire dall'inizio della giornata sino all'intervallo considerato.
Il secondo fattore necessario per calcolare l'OEE è il rendimento. Questa misura può essere ottenuta dal tempo ideale di ciclo, dal numero di parti prodotte e dal tempo di esecuzione, come visto nel capitolo 2.
Il numero di parti prodotte, nel calcolo istantaneo, è uguale alla differenza del valore del contatore aggregato dei pezzi prodotti tra il termine e l'inizio dell'intervallo scelto; nel calcolo aggregato, invece corrisponde al valore del contatore aggregato dei pezzi prodotti al termine dell'intervallo considerato.
Il tempo ideale di ciclo, indica la velocità massima teorica della macchina. Questa informazione non è presente nei dati forniti, per cui è stato ipotizzato un calcolo approssimato per ottenerne un valore plausibile. Il calcolo consiste nel selezionare il minuto in cui la macchina ha avuto la massima produttività e, quindi, ottenere il tempo ideale di ciclo come rapporto tra 60 secondi e il numero di pezzi prodotti in quel minuto.
L'ultimo fattore necessario per il calcolo dell'OEE è la qualità. La qualità corrisponde al rapporto tra parti buone e parti totali. Tuttavia nei dati forniti, non sono presenti informazioni relative al numero di parti scartate: ho, quindi, supposto un numero di scarti pari a 0 e di conseguenza una qualità del 100%.
Nella tabella 4.1 sono esplicitati i campi necessari per il calcolo dell'OEE.

Misura	Campo corrispondente
Tempo di produzione pianificato	Timestamp
Tempo di arresto	Timestamp e contatore aggregato dei pezzi in uscita
Numero di parti totali	Timestamp e contatore aggregato dei pezzi in uscita
Tempo ideale di ciclo	Timestamp e contatore aggregato dei pezzi in uscita
Numero di parti scartate	Non presente

Tabella 4.1: Campi necessari per il calcolo dell'OEE dell'etichettatrice

Dalla tabella emerge che sono necessari solamente il timestamp e il contatore aggregato dei pezzi in uscita: è stato, quindi, creato un file in formato csv a partire dal foglio di lavoro Excel, contenente i suddetti campi.

4.6 Modello dei dati: etichettatrice

Dopo aver estratto i dati utili per calcolare l'OEE, il file in formato csv è stato caricato nel DBMS Cassandra.
Le tabelle relative all'etichettatrice sono state inserite all'interno di un keyspace con replication_factor pari a 1. Nel keyspace è stata creata la tabella "records", all'interno della quale è stato caricato il contenuto del file csv. La struttura della tabella "records" prevede 2 colonne:

- la colonna time di tipo timestamp, che funge da Primary Key;
- la colonna pezzi_prodotti di tipo int, che indica il numero di pezzi prodotti nel corrispondente timestamp.

Successivamente, è stata creata la tabella "risultati", nella quale in un secondo momento saranno salvati i dati elaborati attraverso Spark. La tabella "risultati" sarà composta dai campi indicati nella tabella 4.2.

Campo	Tipo	Descrizione
Data	date	Data
Tempo	time	Orario

Disponibilità	float	Disponibilità nei 10 minuti che terminano all'orario indicato da Tempo
Disponibilità_agg	float	Disponibilità dall'inizio della giornata all'orario indicato da Tempo
Oee	float	Oee nei 10 minuti che terminano all'orario indicato da Tempo
Oee_agg	float	Oee dall'inizio della giornata all'orario indicato da Tempo
Pezzi_prodotti	int	Pezzi prodotti nei 10 minuti che terminano all'orario indicato da Tempo
Pezzi_prodotti_agg	int	Pezzi prodotti dall'inizio della giornata all'orario indicato da Tempo
Qualita	float	Qualità nei 10 minuti che terminano all'orario indicato da Tempo
Qualita_agg	float	Qualità dall'inizio della giornata all'orario indicato da Tempo
Rendimento	float	Rendimento nei 10 minuti che terminano all'orario indicato da Tempo
Rendimento_agg	float	Rendimento dall'inizio della giornata all'orario indicato da Tempo
Tempo_fermi_macchina	int	Totale tempo di arresto nei 10 minuti che terminano all'orario indicato da Tempo, espresso in secondi

Tempo_fermi_macchina_agg	int	Totale tempo di arresto dall'inizio della giornata all'orario indicato da Tempo, espresso in secondi
Timestamp	timestamp	Timestamp che indica data e ora

Tabella 4.2: Contenuto della tabella risultati dell'etichettatrice

La chiave primaria della tabella "risultati" è costituita dalla coppia "data-tempo", mentre la clustering column corrisponderà alla colonna tempo. In tal modo i records saranno raggruppati per data ed all'interno di ogni gruppo i dati saranno ordinati per orario.

4.7 Elaborazione dei dati: etichettatrice

L'ultimo step è l'elaborazione dei dati attraverso Spark. Il codice per l'elaborazione dei dati è stato scritto in linguaggio Java e utilizza i Resilient Distributed Dataset. Le principali azioni e trasformazioni svolte sono le seguenti:

1. Lettura dei dati da Cassandra e memorizzazione in un JavaRDD di stringhe;

2. Calcolo del minuto di massima produttività corrispondente al tempo ideale di ciclo. Per eseguire questa operazione è state utilizzata la trasformazione mapToPair, attraverso cui associo ad ogni stringa il minuto a cui corrisponde la misura. Successivamente attraverso il reduceByKey è stato calcolato per ogni minuto il valore minimo e massimo di pezzi_prodotti. Lo step successivo è consistito in un mapToPair per il calcolo del numero di pezzi prodotti per ogni minuto. Infine attraverso l'azione reduce è stato ottenuto il minuto con la massima produttività.

3. Calcolo dei fermi macchina e del numero di pezzi prodotti. Attraverso la trasformazione "groupByKey" vengono raggruppate le misure in base al numero di pezzi prodotti. Nel caso in cui la lista ottenuta per un dato numero di pezzi prodotti contiene più di un elemento allora viene considerato come tempo di fermo macchina, la differenza tra l'ultimo e il primo istante in cui il contatore resta invariato. Si esegue un mapToPair: la chiave dell'RDD ottenuto sarà un oggetto che indica un intervallo da 10 minuti, estratto dal timestamp, mentre il valore sarà un oggetto che contiene il numero di pezzi prodotti e il tempo di fermo macchina.

4. Calcolo dell'OEE, della disponibilità e del rendimento. Attraverso la trasformazione reduceByKey, viene calcolato per ogni intervallo da 10 minuti il tempo totale di fermi macchina ed il numero di pezzi prodotti. Infine attraverso la funzione Map vengono calcolati OEE, disponibilità e rendimento.

5. Salvataggio dei risultati ottenuti in Cassandra.

4.8 Risultati: etichettatrice

I risultati sono stati rappresentati attraverso vari grafici all'interno di una dashboard realizzata tramite Apache Zeppelin. Il primo grafico è un grafico a barre che rappresenta l'andamento dell'oee istantaneo per ognuno dei 3 giorni analizzati: i grafici ottenuti sono mostrati nelle figure 4.7, 4.9 4.11.
Gli stessi grafici sono stati realizzati per rappresentare l'andamento dell'oee aggregato: i grafici ottenuti sono mostrati nelle figure 4.8, 4.10, 4.12.
La seconda tipologia di grafico è un grafico a linea che mostra l'andamento dell'oee in funzione dell'orario. Sull'asse delle ascisse è rappresentato l'orario, mentre sull'asse delle ordinate l'oee medio per quell'orario. Questo grafico è molto utile per capire quali siano i momenti della giornata più produttivi. Il suddetto grafico è rappresentato nella figura 4.13.

4.9 Descrizione dei dati: tappatrice

I dati raccolti dai sensori presenti sulla tappatrice sono relativi al periodo tra il 16 Aprile 2018 e il 20 aprile 2018 e sono stati forniti tramite un foglio di lavoro Excel. Nel periodo di studio sono stati prelevati 52563 campioni.
Per ogni campione sono presenti i seguenti dettagli:

- Timestamp: indica la data e l'orario in cui è stato acquisito il campione;
- Fotocellula conteggio ingresso dosata: indica il numero di pezzi in ingresso nella macchina;
- Fotocellula conteggio uscita dosata: indica il numero di pezzi in uscita dalla macchina;
- Carico minimo generico: indica se si ha un allarme di carico minimo;
- Termico intervenuto/allarme generico: indica se si ha un allarme generico;
- Fotocellula carico minimo ingresso flacone;
- Fotocellula carico massimo uscita flacone;
- Contatore aggregato dei pezzi in ingresso nella macchina;

- Contatore aggregato dei pezzi in uscita dalla macchina;
- Velocità macchina tappatura.

Anche in questo caso, i dati sono stati acquisiti in parte in modalità "Time Based" ogni 20 secondi ed in parte in modalità "Event Based" con trigger sugli allarmi.

4.10 Preprocessing dei dati: tappatrice

Lo step successivo, alla raccolta dei dati, è stato quello di definire la granularità di calcolo dell'OEE e capire quali dei dati presenti per ogni campione sono utili al fine di calcolare l'OEE.

Granularità di calcolo Anche in questo caso, come per l'etichettatrice, sono state scelte 2 modalità per il calcolo dell'OEE:

- calcolo istantaneo: consiste nel calcolo per ogni intervallo di 10 minuti;
- calcolo aggregato: consiste in un calcolo per ogni intervallo di 10 minuti a partire dall'inizio della giornata.

Estrazione dei dati Dopo aver stabilito la granularità di calcolo, bisogna estrarre da ogni campione le misure realmente necessarie per calcolare l'OEE.
Nel capitolo 2, è stato spiegato che l'OEE si può ottenere tramite il prodotto di disponibilità, rendimento e qualità.
Tuttavia, per calcolarlo in questo modo, è necessario avere dati relativi ai fermi macchina. Come per l'etichettatrice, l'unica informazione relativa ai fermi macchina è un bit che indica se la macchina sta lavorando o è ferma: questa informazione non è quindi molto utile ai fini del calcolo dell'OEE poichè non indica l'intervallo di tempo in cui la macchina è ferma. In questo caso però, a differenza dell'etichettatrice, non è stato neanche possibile supporre che la macchina sia ferma quando il contatore dei pezzi prodotti resta invariato per più campioni consecutivi: infatti da un'analisi dei dati è emerso che vi sono molti records consecutivi in cui il contatore è invariato e di conseguenza la macchina risulterebbe sempre ferma. Per questi motivi, l'OEE è stato calcolato attraverso quello che nel capitolo 2 è stato definito "calcolo semplice":

$$OEE = \frac{\text{Parti buone} \cdot \text{Tempo ideale di ciclo}}{\text{Tempo di produzione pianificato}}$$

Il tempo di produzione pianificato equivale all'intervallo di tempo in cui è stata effettuata l'osservazione: per il calcolo istantaneo corrisponderà a 10 minuti, tranne per primo ed ultimo intervallo, in cui la durata può essere inferiore; per il calcolo aggregato corrisponderà all'intervallo di tempo trascorso dall'inizio della giornata al termine dell'intervallo considerato.
Il tempo ideale di ciclo, indica la velocità massima teorica della macchina. Questa

informazione non è presente nei dati forniti, per cui è stato ipotizzato un calcolo approssimato per ottenerne un valore plausibile. Il calcolo consiste nel selezionare il minuto in cui la macchina ha avuto la massima produttività e, quindi, ottenere il tempo ideale di ciclo come rapporto tra 60 secondi e il numero di pezzi prodotti in quel minuto.
Infine l'informazione sul numero di parti buone è stata ricavata dal contatore aggregato dei pezzi in uscita: nel calcolo istantaneo considerando la differenza del valore del contatore aggregato dei pezzi in uscita tra il primo e l'ultimo timestamp dell'intervallo considerato. Nel calcolo aggregato, il numero di parti buone corrisponde al valore del contatore aggregato nell'ultimo timestamp relativo all'intervallo scelto. Purtroppo, anche in questo caso, non sono presenti informazioni sul numero di scarti. Ne consegue che il numero di scarti è stato supposto essere uguale a 0: il numero di parti buone corrisponderà al numero di parti prodotte. Nella tabella 4.3 è mostrato quali sono i campi necessari per il calcolo delle varie misure.

Misura	**Campo corrispondente**
Tempo di produzione pianificato	Timestamp
Numero di parti buone	Timestamp e contatore aggregato dei pezzi in uscita
Tempo ideale di ciclo	Timestamp e contatore aggregato dei pezzi in uscita

Tabella 4.3: Campi necessari per calcolare l'OEE della tappatrice

Dalla tabella emerge che sono necessari solamente il Timestamp e il contatore aggregato dei pezzi in uscita: è stato, quindi, creato un file in formato csv a partire dal foglio di lavoro Excel, contenente i suddetti campi.

4.11 Modello dei dati: tappatrice

Dopo aver determinato i campi utili per il calcolo dell'OEE, il file in formato csv è stato caricato nel DBMS Apache Cassandra. Le tabelle relative alla tappatrice sono state inserite all'interno di un keyspace con replication_factor pari a 1. Nel keyspace è stata creata la tabella "records", all'interno della quale è stato caricato il contenuto del file csv. La struttura della tabella "records" prevede 2 colonne:

- la colonna time di tipo timestamp, che funge da Primary Key;
- la colonna pezzi_prodotti di tipo int, che indica il numero di pezzi prodotti nel corrispondente timestamp.

Successivamente, è stata creata la tabella "risultati", nella quale in un secondo momento saranno salvati i dati elaborati attraverso Spark. La tabella "risultati" sarà composta dai campi indicati nella tabella 4.4.

Campo	Tipo	Descrizione
Data	date	Data
Tempo	time	Orario
Oee	float	Oee nei 10 minuti che terminano all'orario indicato da Tempo
Oee_agg	float	Oee dall'inizio della giornata all'orario indicato da Tempo
Pezzi_prodotti	int	Pezzi prodotti nei 10 minuti che terminano all'orario indicato da Tempo
Pezzi_prodotti_agg	int	Pezzi prodotti dall'inizio della giornata all'orario indicato da Tempo
Timestamp	timestamp	Timestamp che indica data e ora

Tabella 4.4: Contenuto della tabella risultati della tappatrice

La chiave primaria della tabella "risultati" è costituita dalla coppia "data-tempo", mentre la clustering column corrisponderà alla colonna tempo. In tal modo i records saranno raggruppati per data ed all'interno di ogni gruppo i dati saranno ordinati per orario.

4.12 Elaborazione dei dati: tappatrice

L'ultimo step è l'elaborazione dei dati attraverso Spark. Il codice per l'elaborazione dei dati è stato scritto in linguaggio Java e utilizza i Resilient Distributed Dataset. Le principali azioni e trasformazioni svolte sono le seguenti:

1. Lettura dei dati da Cassandra e memorizzazione in un JavaRDD di stringhe;

2. Calcolo del minuto di massima produttività corrispondente al tempo ideale di ciclo. Per eseguire questa operazione è state utilizzata la trasformazione mapToPair, attraverso cui associo ad ogni stringa il minuto a cui corrisponde la misura. Successivamente attraverso il reduceByKey è stato calcolato per ogni minuto il valore minimo e massimo di pezzi_prodotti. Lo step successivo è consistito in un mapToPair per il calcolo del numero di pezzi prodotti per ogni minuto. Infine attraverso l'azione reduce è stato ottenuto il minuto con la massima produttività.

3. Calcolo del numero di pezzi prodotti. Attraverso la funzione mapToPair si associa ad ogni intervallo da 10 minuti il contatore dei pezzi prodotti.

4. Calcolo dell'OEE.. Attraverso la trasformazione reduceByKey, viene calcolato per ogni intervallo da 10 minuti il numero totale di pezzi prodotti. Infine attraverso la funzione Map viene calcolato l'OEE.

5. Salvataggio dei risultati ottenuti in Cassandra.

4.13 Risultati: tappatrice

Anche in questo, come per l'etichettatrice si è utilizzato Apache Zeppelin per la realizzazione di una dashboard in cui vengono mostrati i risultati.
Per la rappresentazione dei dati sono stati ideati vari grafici. La prima tipologia consiste in un grafico a barre che rappresenta l'andamento dell'oee istantaneo per ognuno dei 5 giorni analizzati: i grafici ottenuti sono mostrati nelle figure 4.14, 4.16, 4.18, 4.20, 4.22.
Gli stessi grafici sono stati realizzati per rappresentare l'andamento dell'oee aggregato: i grafici ottenuti sono mostrati nelle figure 4.15, 4.17, 4.19, 4.21, 4.23.
La seconda tipologia di grafico è un grafico a linea che mostra l'andamento dell'oee in funzione dell'orario. Sull'asse delle ascisse è rappresentato l'orario, mentre sull'asse delle ordinate l'oee medio. Questo grafico è molto utile per capire quali siano i momenti della giornata più produttivi. Il suddetto grafico è rappresentato nella figura 4.24.

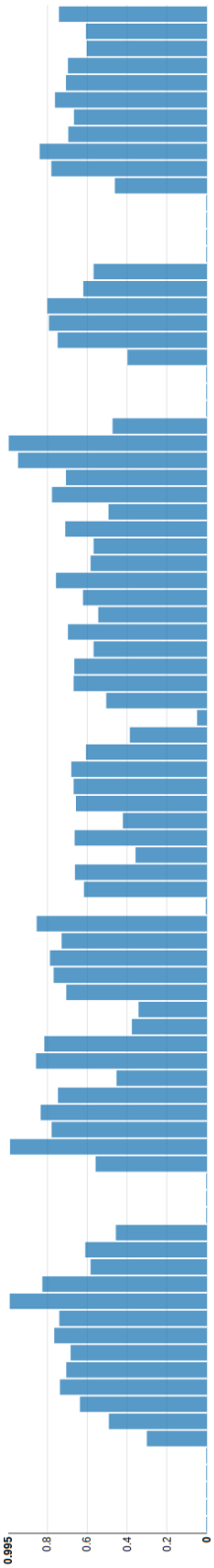

Figura 4.7: Andamento oee istantaneo dell'etichettatrice del 24 aprile 2018 tra le ore 9 e le ore 24. Valore minimo oee: 0; valore massimo oee: 0.995.

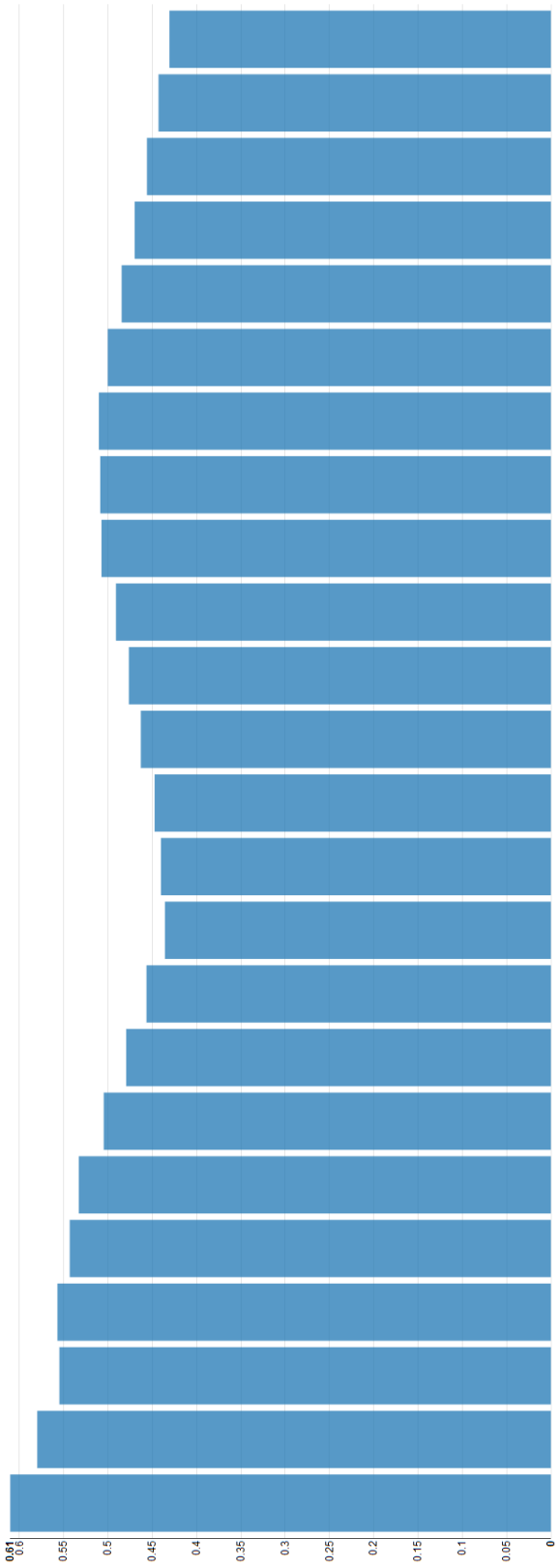

Figura 4.8: Andamento oee aggregato dell'etichettatrice del 24 aprile 2018 tra le ore 9 e le ore 24. Valore aggregato massimo: 0.554. Oee del 24 aprile 2018: 0.534.

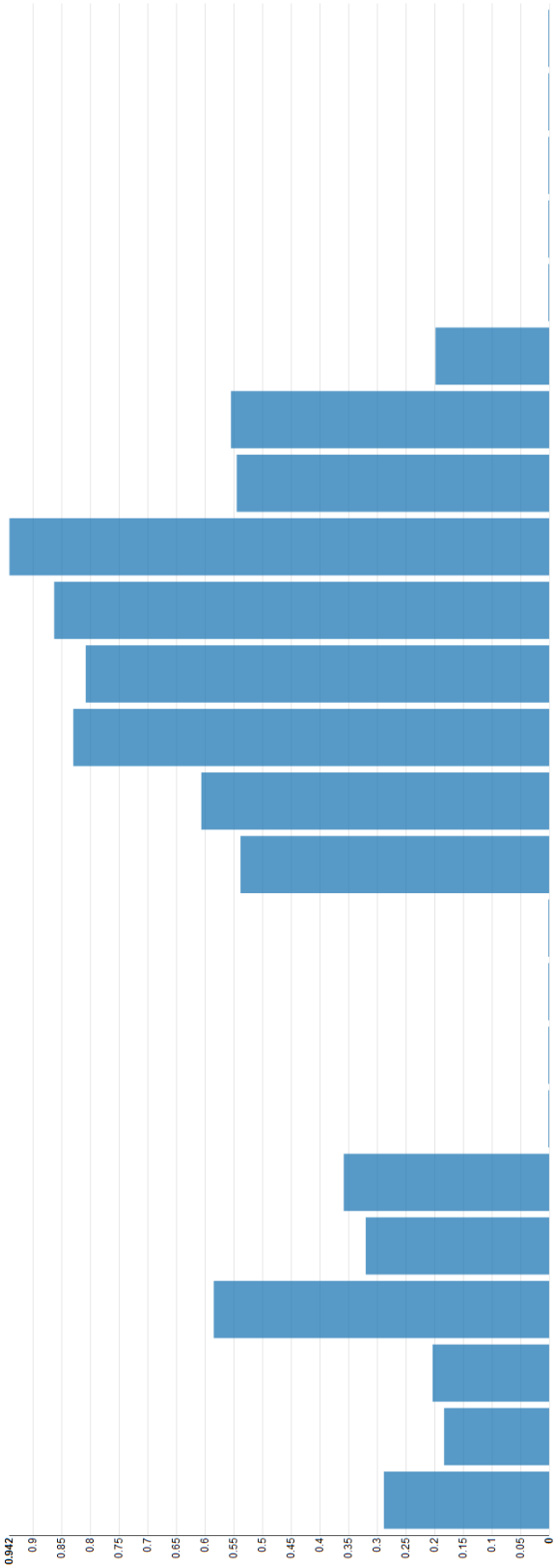

Figura 4.9: Andamento oee istantaneo dell'etichettatrice del 25 aprile 2018 tra le ore 24 e le ore 4. Valore minimo oee: 0; valore massimo oee: 0.942.

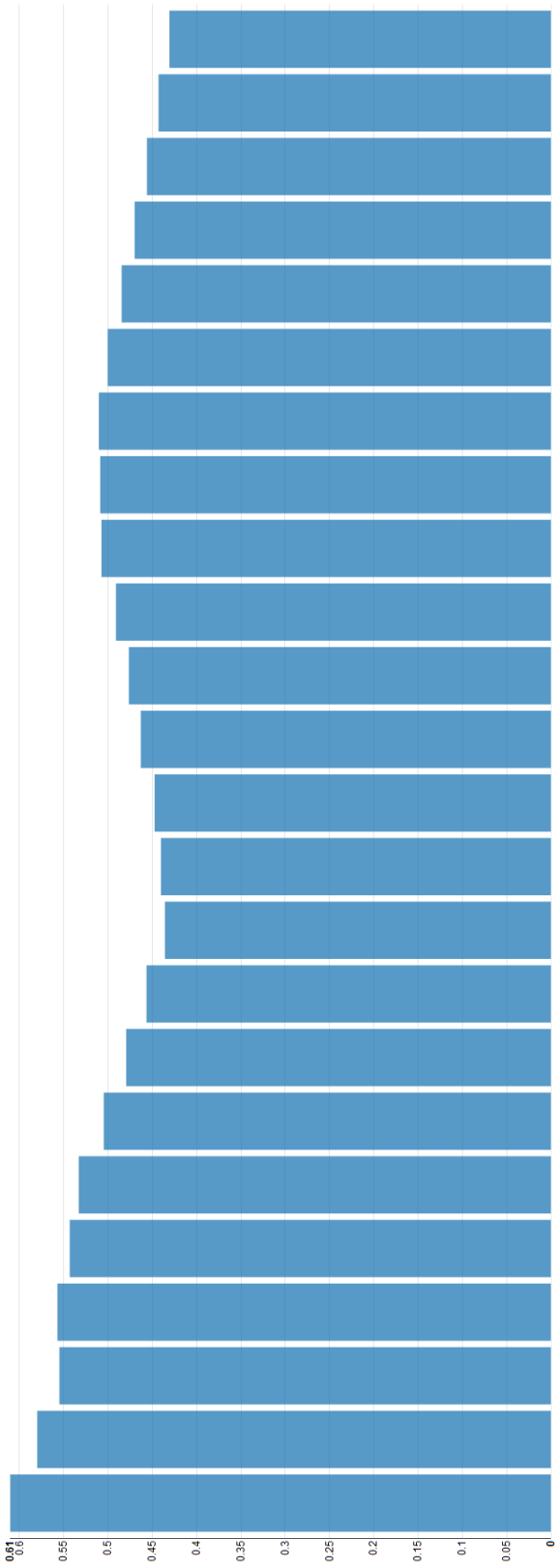

Figura 4.10: Andamento oee aggregato dell'etichettatrice del 25 aprile 2018 tra le ore 24 e le ore 4. Valore aggregato massimo: 0.61. Oee del 25 aprile 2018: 0.431.

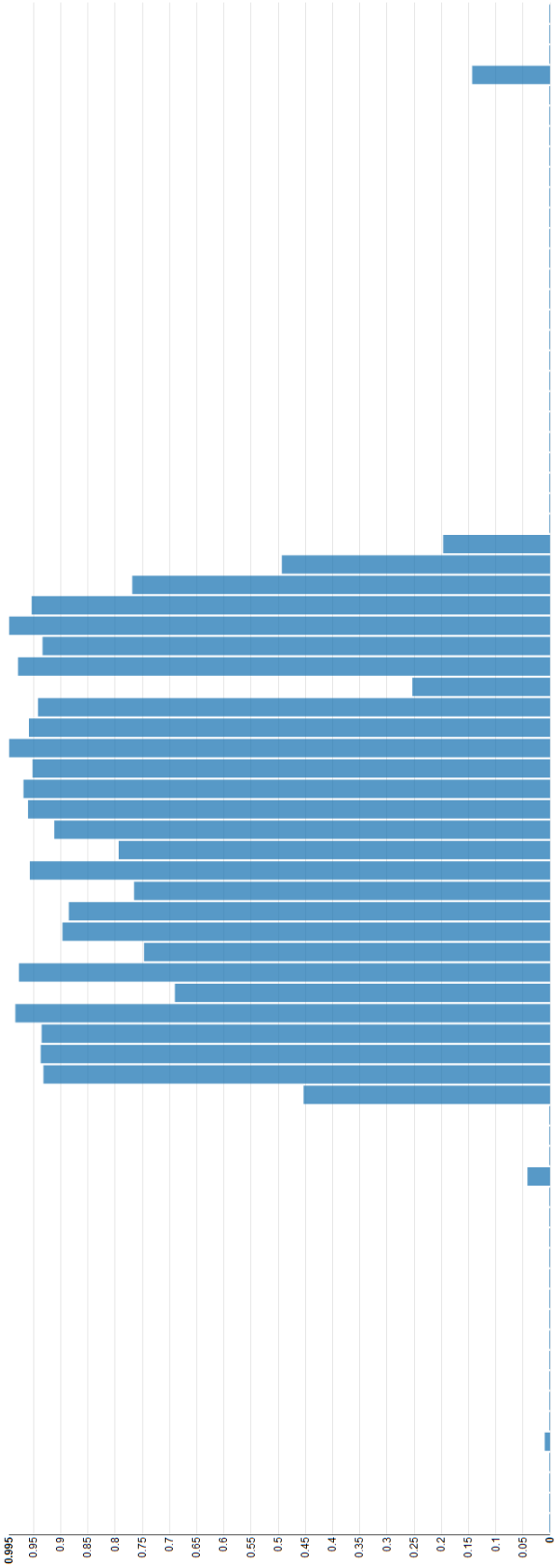

Figura 4.11: Andamento oee istantaneo dell'etichettatrice del 2 maggio 2018 tra le ore 09:30 e le ore 22. Valore minimo oee: 0; valore massimo oee: 0.995.

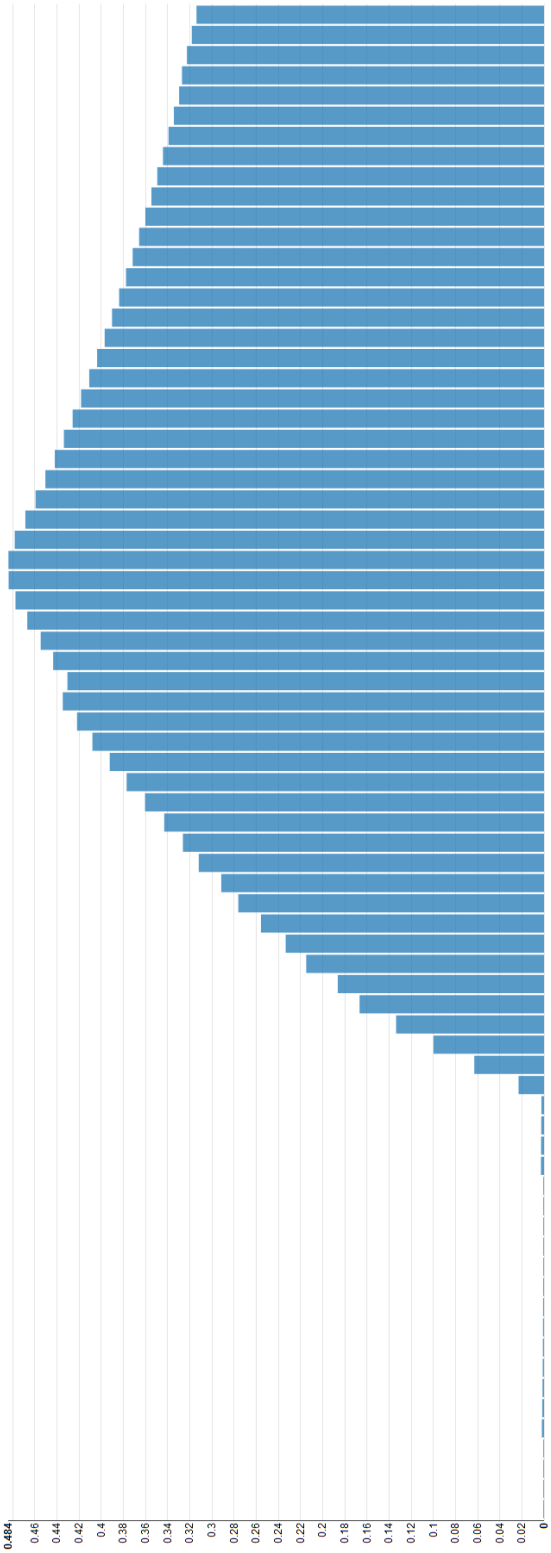

Figura 4.12: Andamento oee aggregato dell'etichettatrice del 2 maggio 2018 tra le ore 09:30 e le ore 22. Valore aggregato massimo: 0.484. Oee del 2 maggio 2018: 0.314.

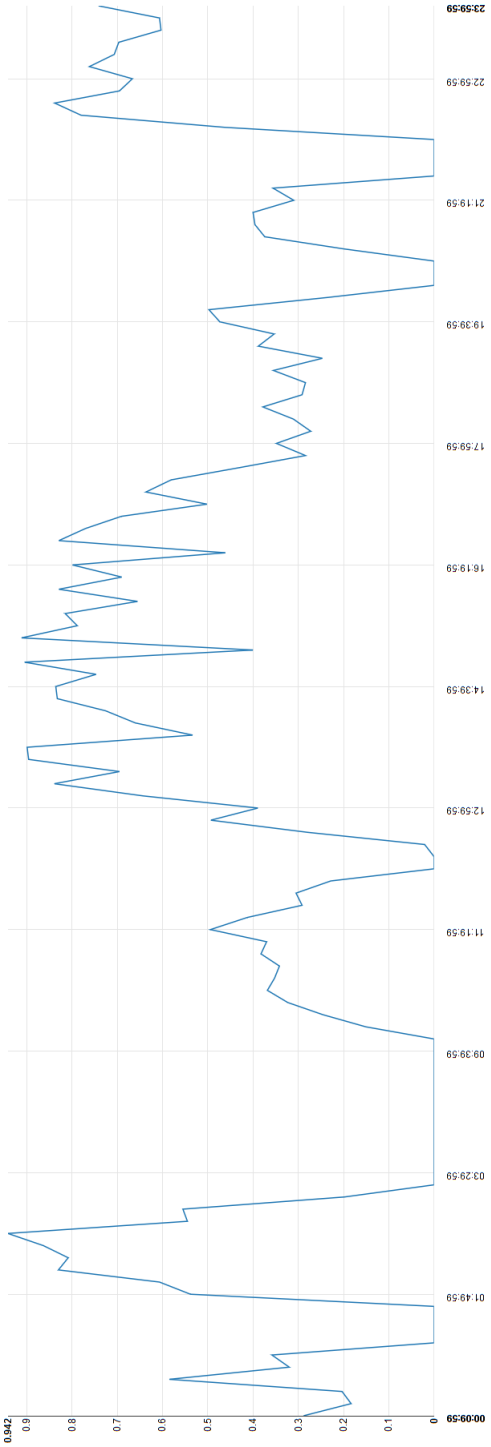

Figura 4.13: Andamento medio dell'oee dell'etichettatrice tra le ore 00:00 e le ore 24. Il momento più produttivo della giornata è l'intervallo tra le 02:30 e le 02:40, in cui l'oee ha un valore medio di 0.942.

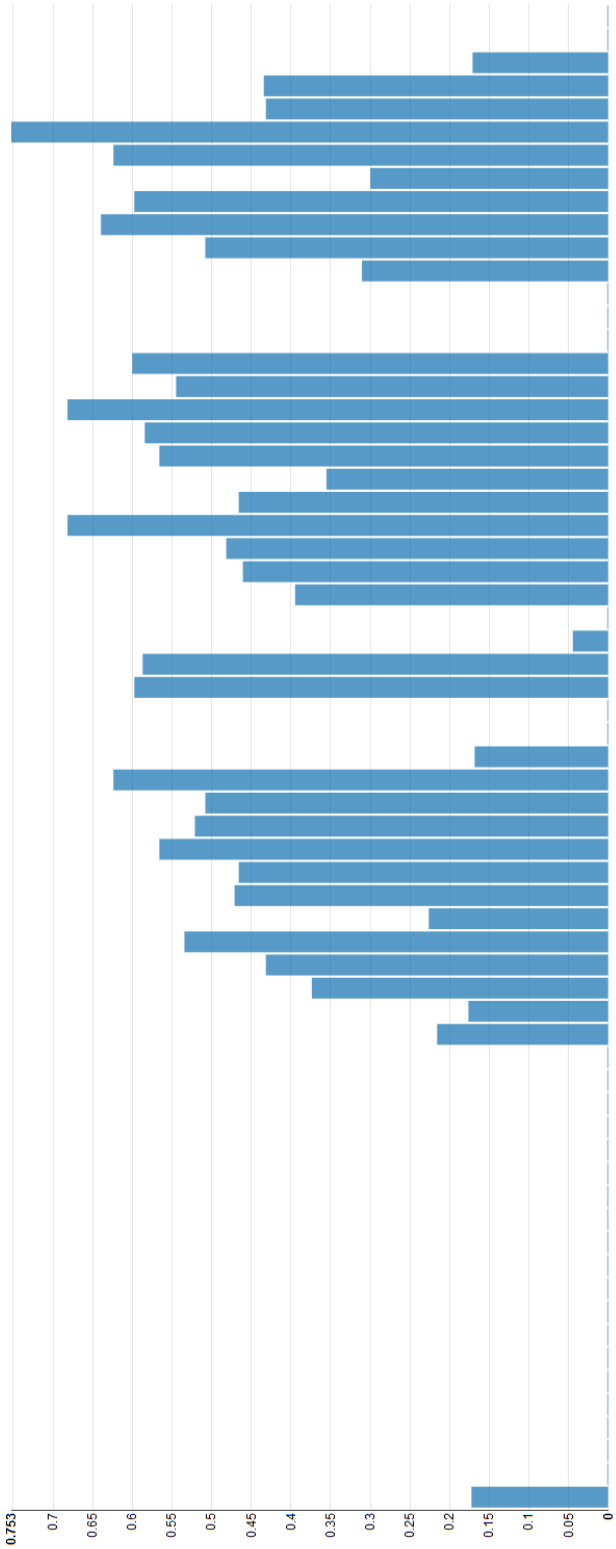

Figura 4.14: Andamento oee istantaneo della tappatrice del 16 aprile 2018 tra le ore 11:10 e le ore 22. Valore minimo oee: 0; valore massimo oee: 0.753.

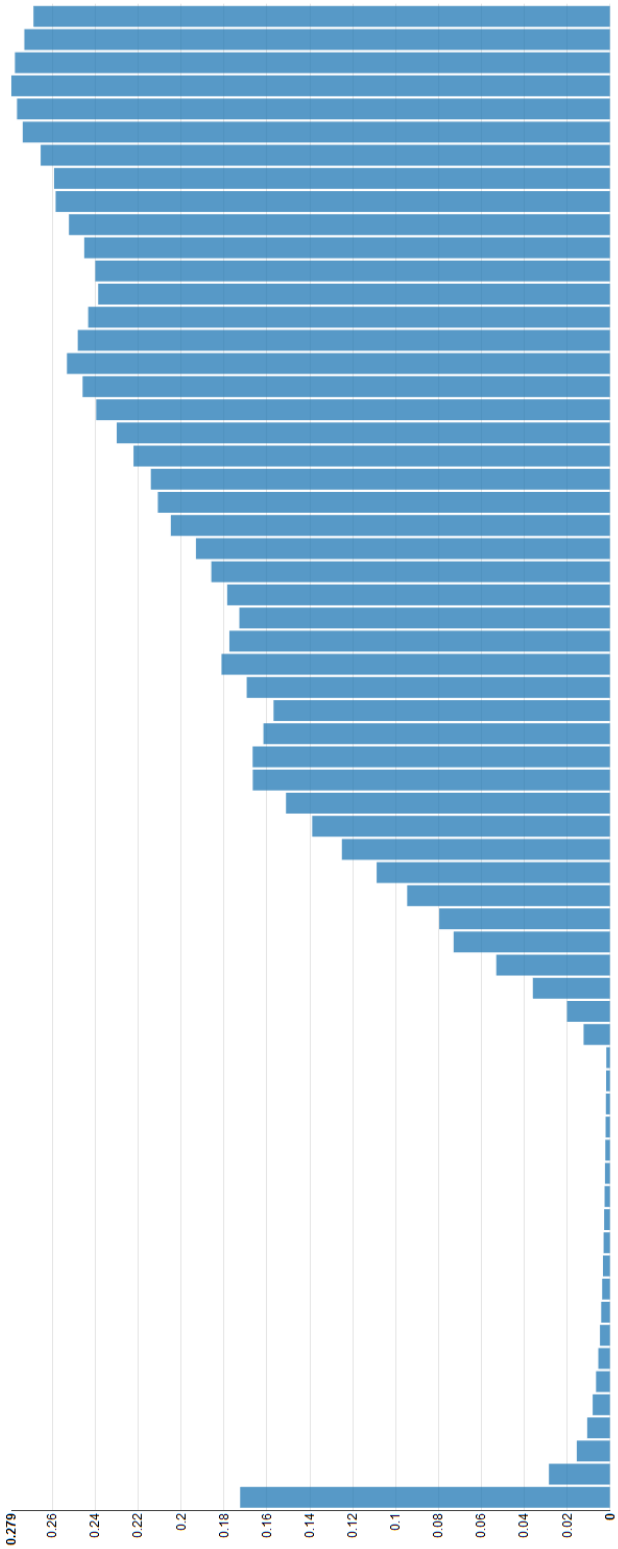

Figura 4.15: Andamento oee aggregato della tappatrice del 16 aprile 2018 tra le ore 11:10 e le ore 22. Valore aggregato massimo: 0.279. Oee del 16 aprile 2018: 0.269.

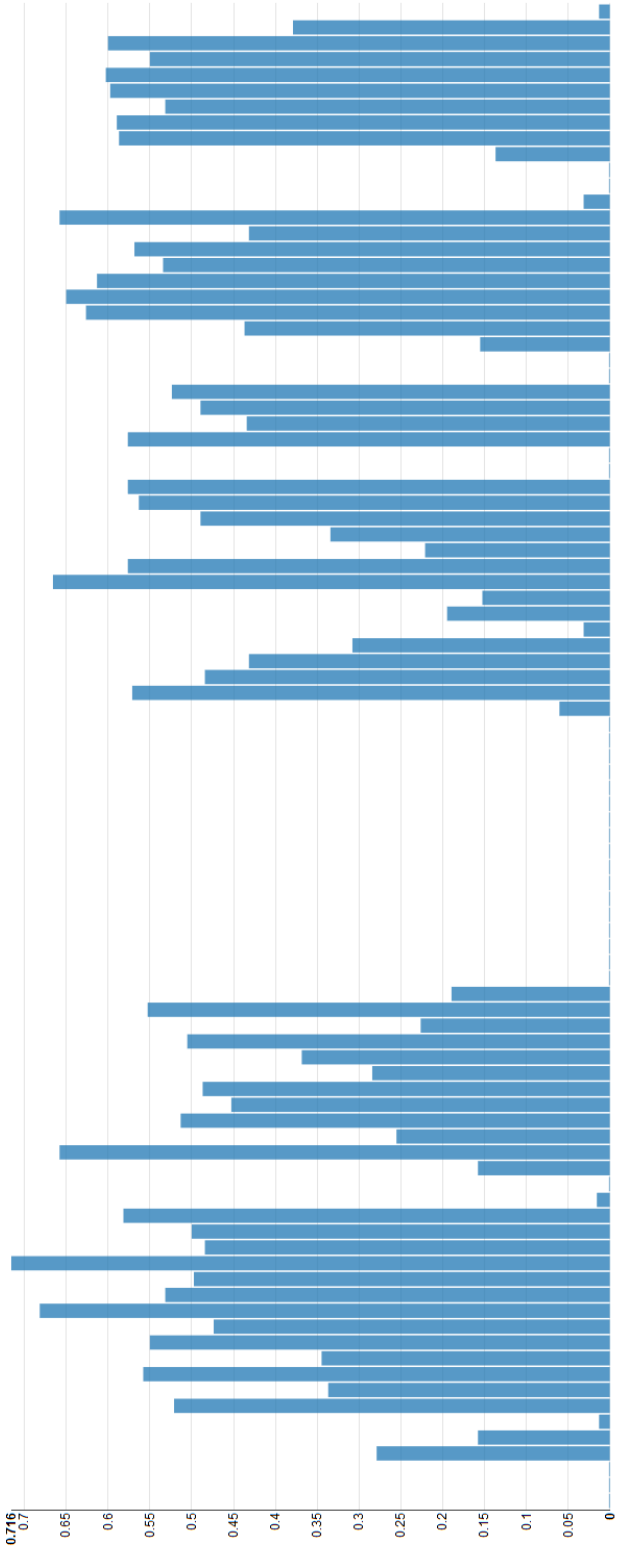

Figura 4.16: Andamento oee istantaneo della tappatrice del 17 aprile 2018 tra le ore 05:50 e le ore 21:40. Valore minimo oee: 0; valore massimo oee: 0.716.

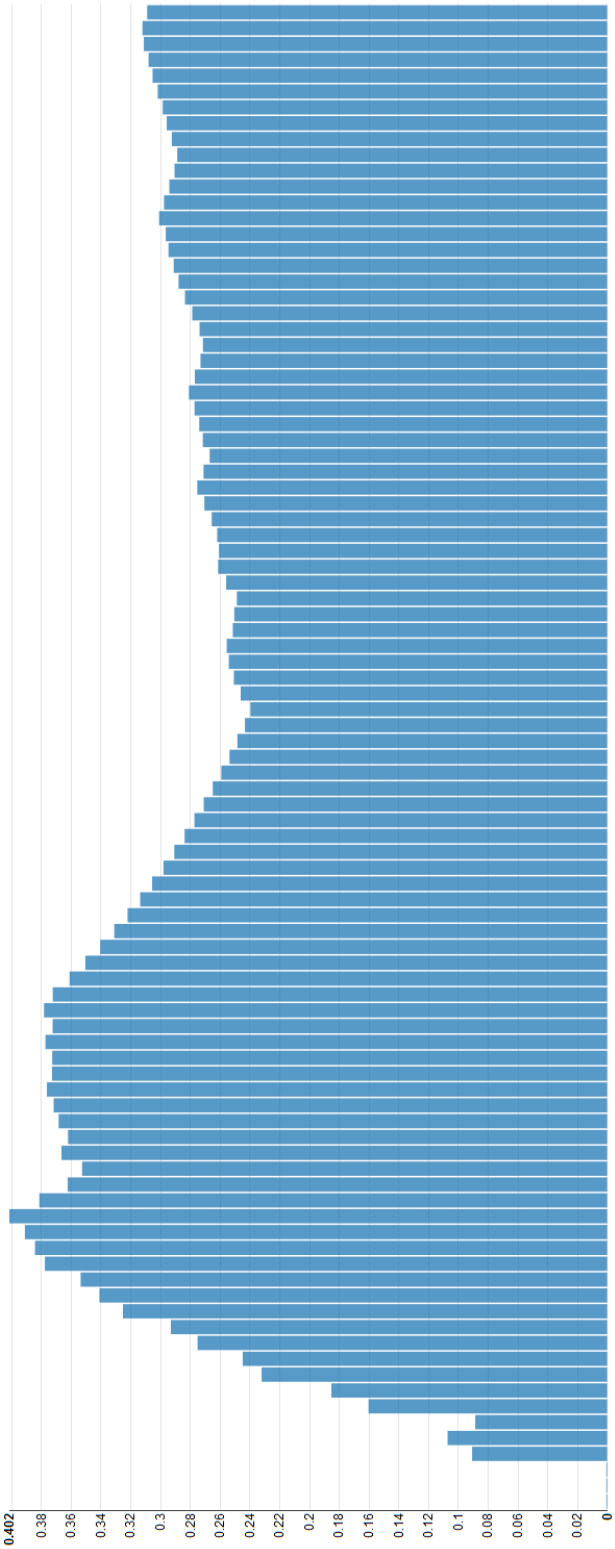

Figura 4.17: Andamento oee aggregato della tappatrice del 17 aprile 2018 tra le ore 05:50 e le ore 21:40. Valore aggregato massimo: 0.402. Oee del 17 aprile 2018: 0.309.

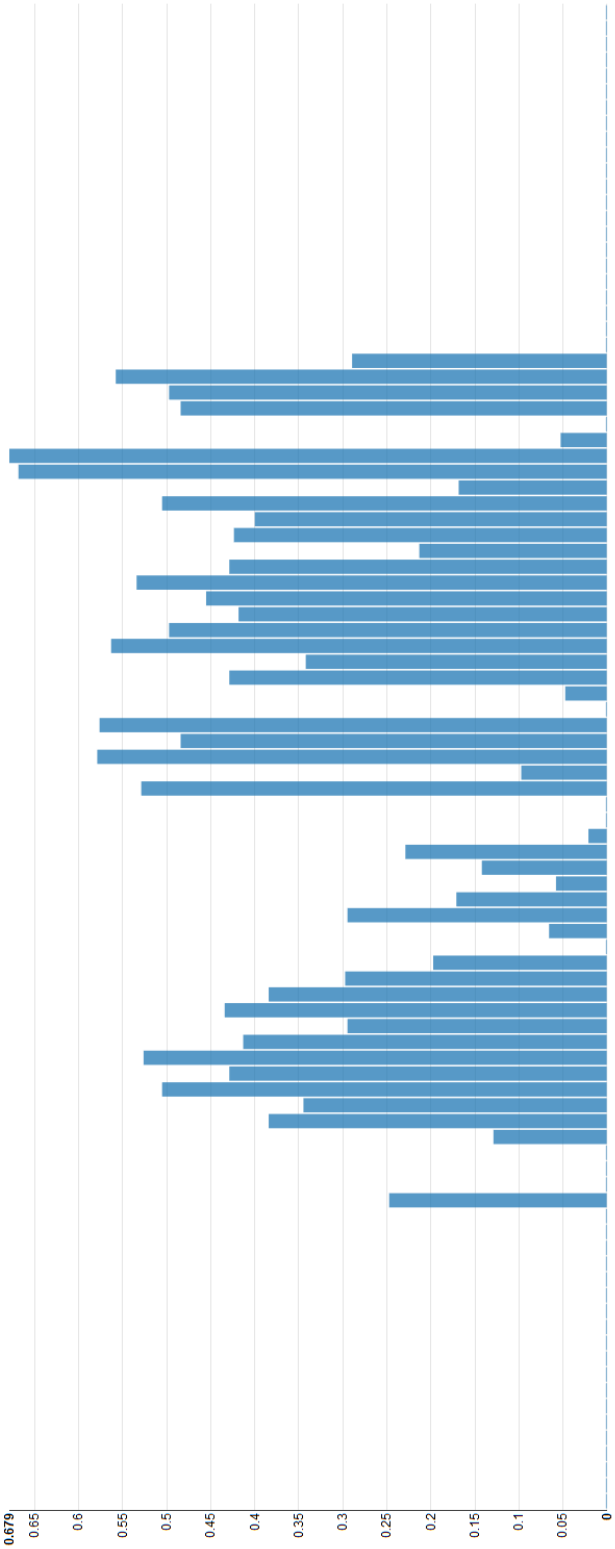

Figura 4.18: Andamento oee istantaneo della tappatrice della tappatrice del 18 aprile 2018 tra le ore 05:30 e le ore 21:20. Valore minimo oee: 0; valore massimo oee: 0.679.

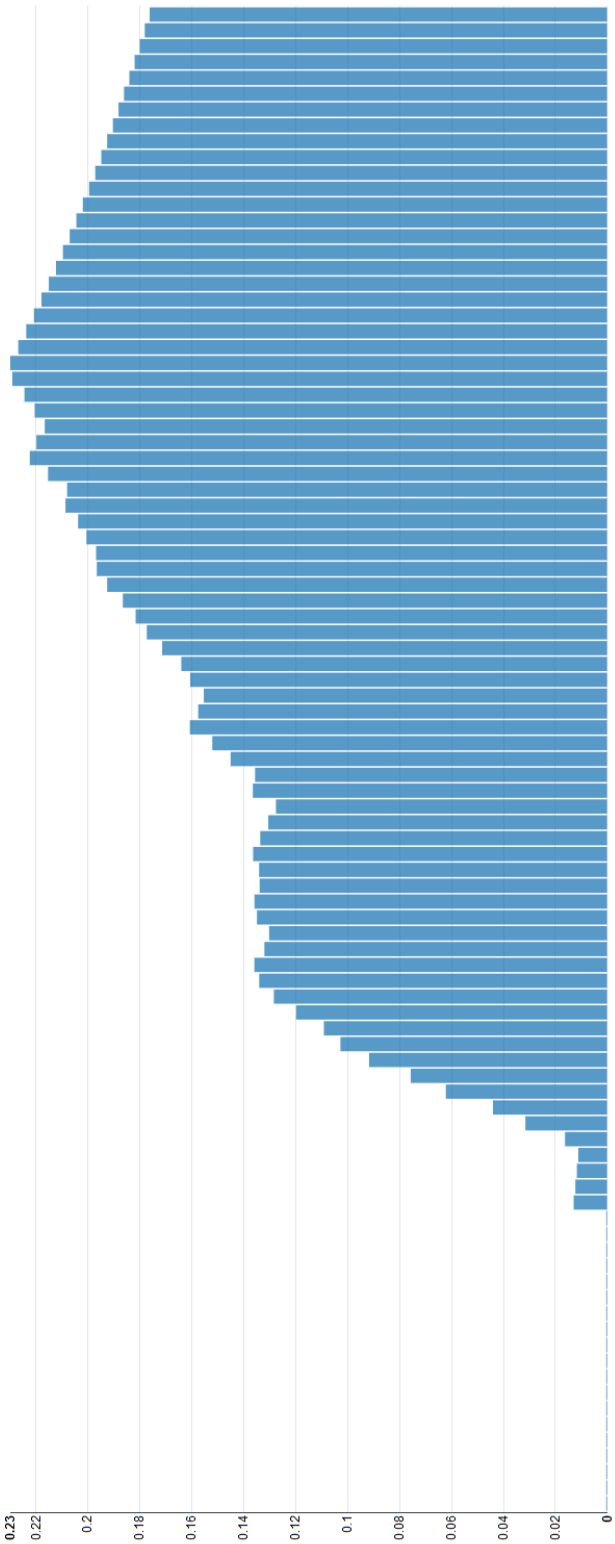

Figura 4.19: Andamento oee aggregato della tappatrice del 18 aprile 2018 tra le ore 05:30 e le ore 21:20. Valore aggregato massimo: 0.23. Oee del 18 aprile 2018: 0.176.

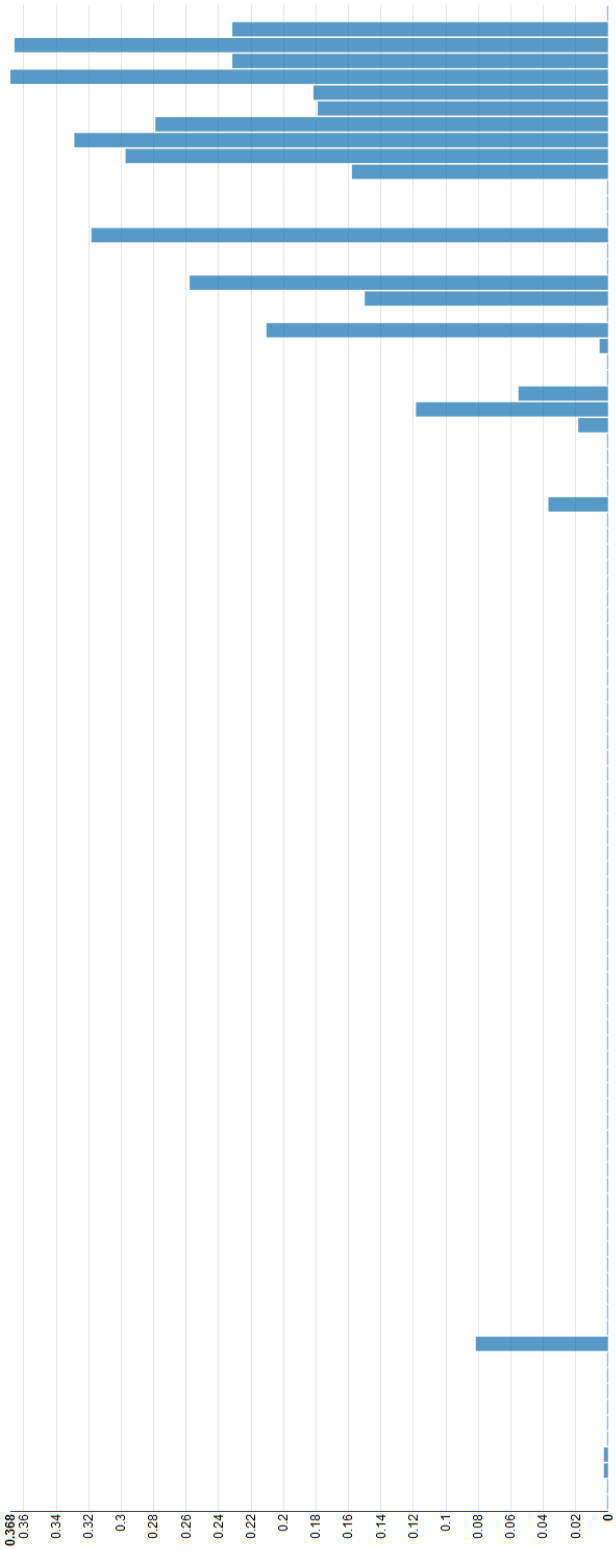

Figura 4.20: Andamento oee istantaneo della tappatrice del 19 aprile 2018 tra le ore 6 e le ore 21:50. Valore minimo oee: 0; valore massimo oee: 0.368.

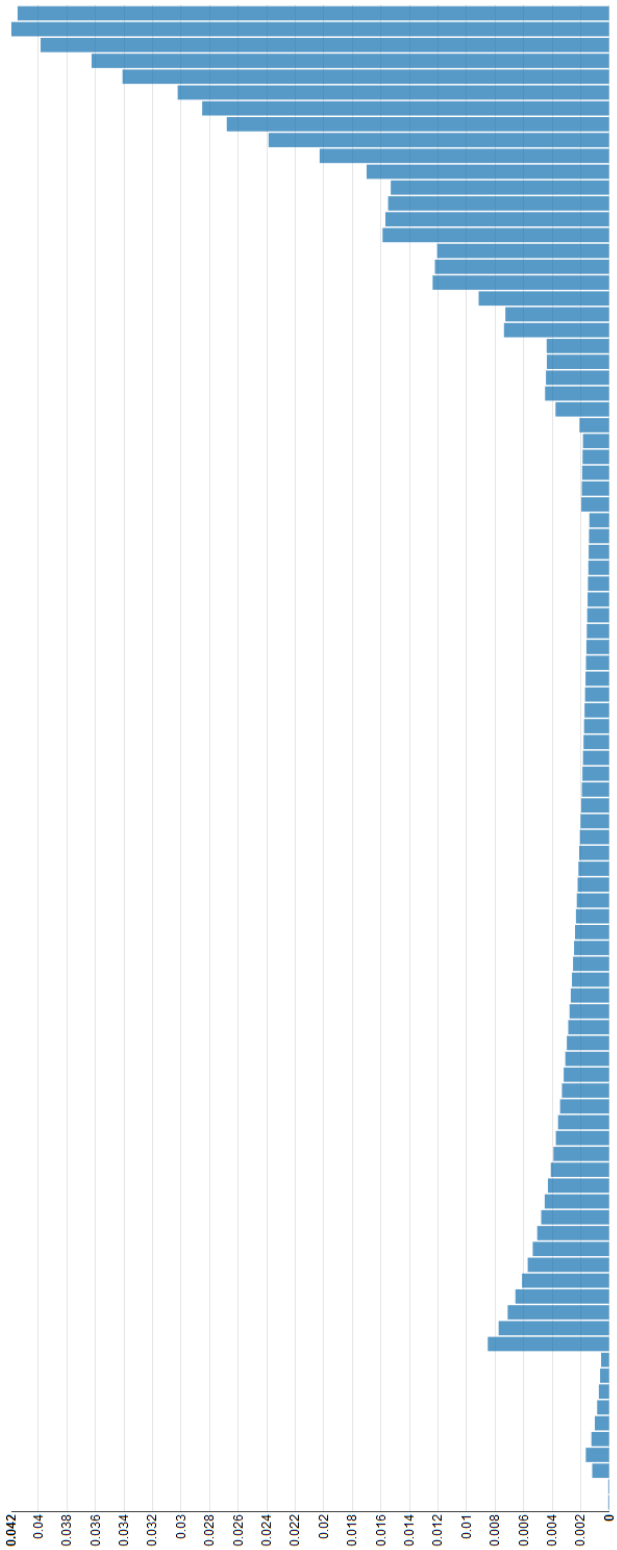

Figura 4.21: Andamento oee aggregato della tappatrice del 19 aprile 2018 tra le ore 6 e le ore 21:50. Valore aggregato massimo: 0.042. Oee del 19 aprile 2018: 0.041.

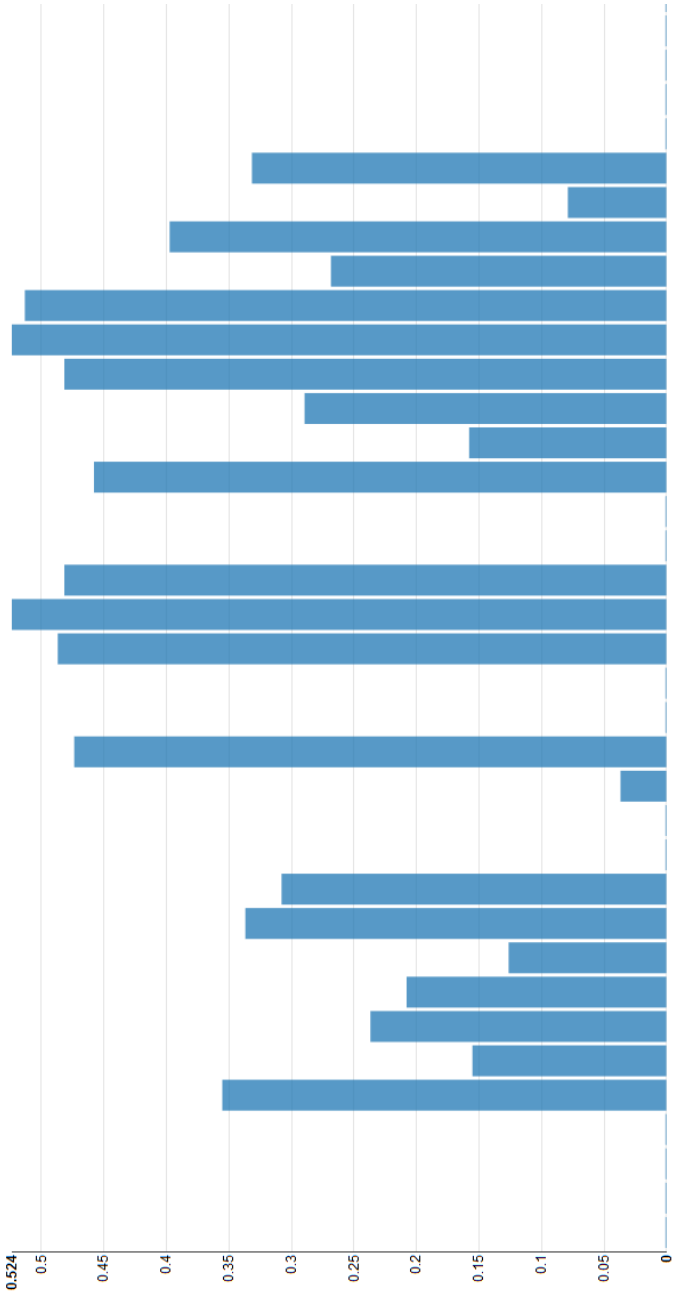

Figura 4.22: Andamento oee istantaneo della tappatrice del 20 aprile 2018 tra le ore 05:40 e le ore 11:40. Valore minimo oee: 0; valore massimo oee: 0.524.

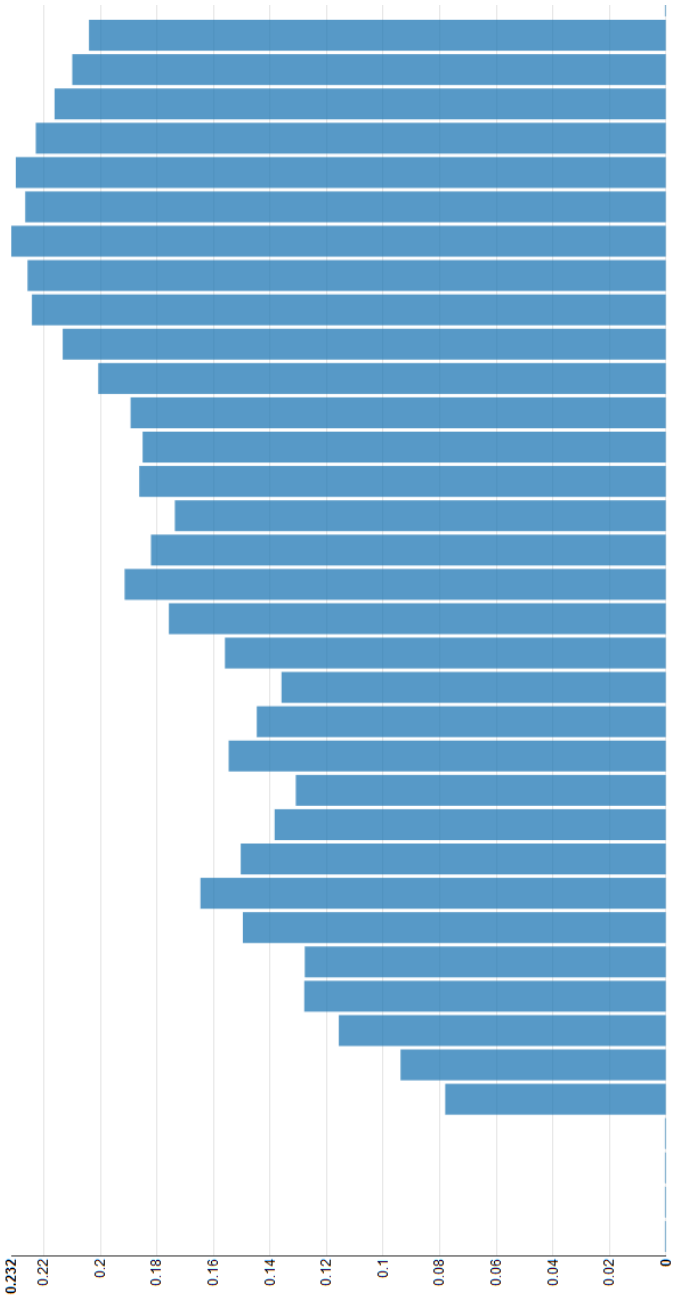

Figura 4.23: Andamento oee aggregato della tappatrice del 20 aprile 2018 tra le ore 05:40 e le ore 11:40. Valore aggregato massimo: 0.232. Oee del 20 aprile 2018: 0.204.

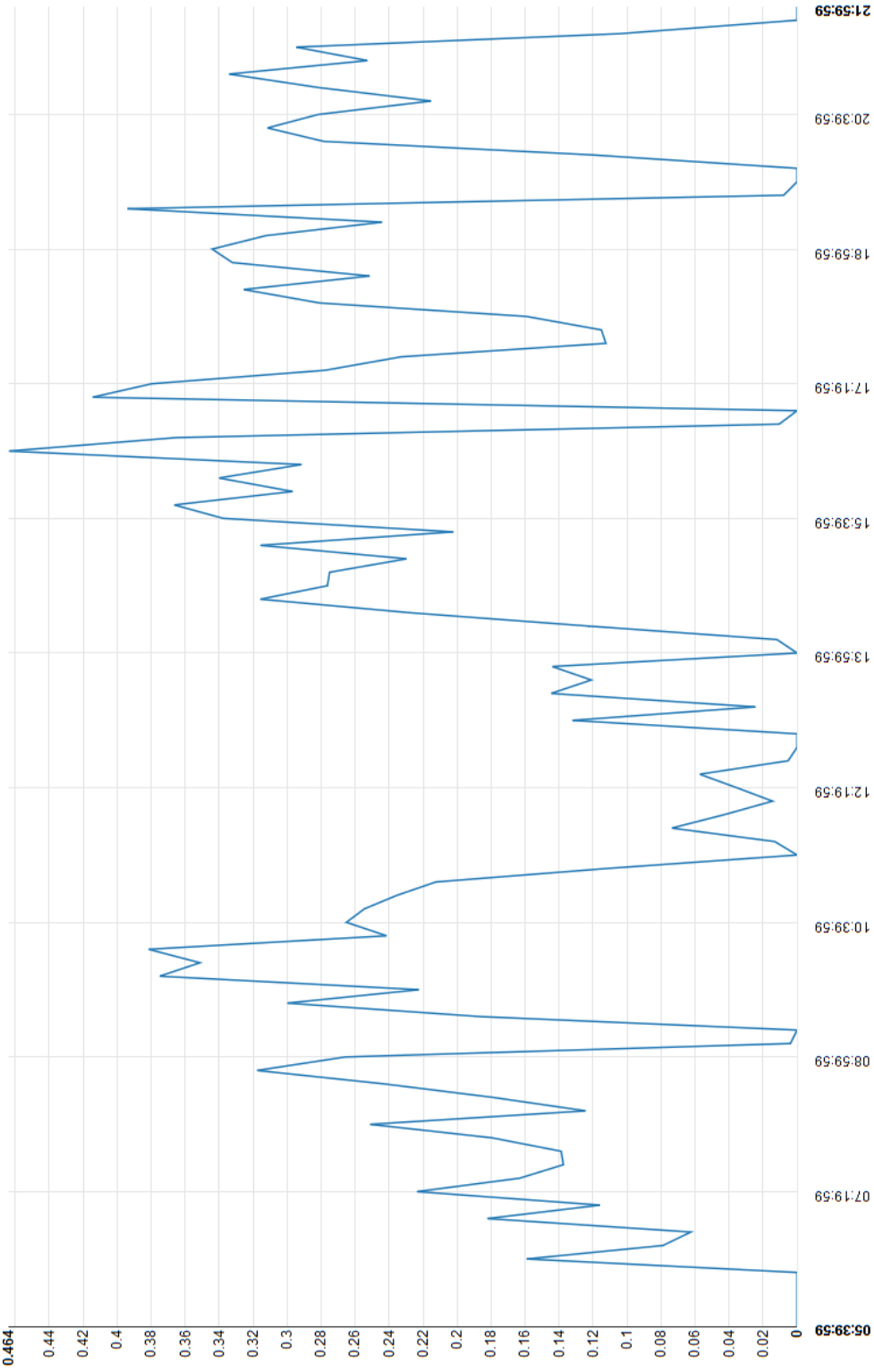

Figura 4.24: Andamento medio dell'oee della tappatrice tra le ore 05:30 e le ore 22. Il momento più produttivo della giornata è l'intervallo tra le 16:20 e le 16:30, in cui l'oee ha un valore medio di 0.464.

Capitolo 5

Analisi dell'efficienza produttiva degli afferenti ad un dipartimento universitario

L'idea di creare una dashboard per il Dauin nasce dalla necessità di avere uno strumento accessibile a tutti e di facile utilizzo, attraverso cui i membri del dipartimento possano visualizzare, attraverso varie tipologie di grafici, l'andamento di vari KPI i quali riassumono alcune informazioni relative alla produttività e alla qualità delle loro pubblicazioni.
L'obiettivo di questo capitolo è quello di descrivere la realizzazione della dashboard informativa del Dauin. In particolar modo, saranno descritte la fase di importazione dei dati sul DBMS, l'elaborazione dei dati e la visualizzazione dei risultati.

5.1 MongoDB: importazione e strutturazione dei dati

In questa sezione sono descritte tutte le fasi necessarie per popolare MongoDB.
I dati raccolti dalle varie fonti(servizio Pauper,Iris.Polito, Scimago, WoS) sono forniti attraverso un dump SQL: il primo step da effettuare è quello di lanciare lo script SQL su un server SQL locale, in modo da poter osservare la struttura dei dati in SQL e capire quali di essi sono necessari per la realizzazione della dashboard.
La struttura del db generato dal dump è rappresentata in figura 5.1.
Tra le tabelle presenti, quelle di interesse per la realizzazione della dashboard sono le seguenti:

- La tabella "people" contiene le informazioni sui membri del dipartimento.
- La tabella "servizio" indica il ruolo accademico di ogni membro del dipartimento per ogni anno in cui ha fatto parte del dipartimento.
- La tabella "qualifica" contiene l'insieme di tutti i possibili ruoli accademici.

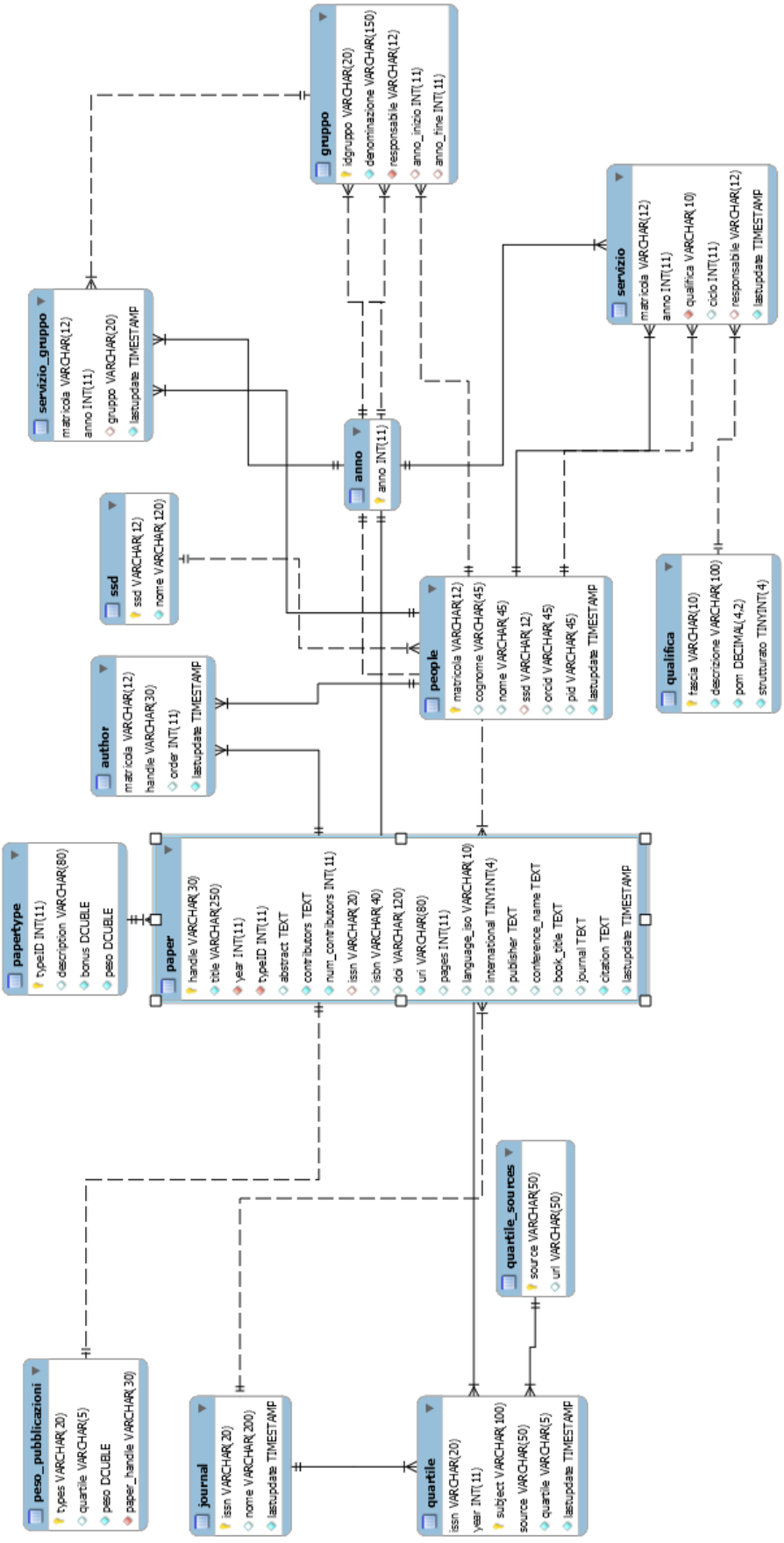

Figura 5.1: Struttura del db generato dal dump sql

- La tabella "quartile" indica uno o più quartili per ogni pubblicazione di tipo rivista.
- La tabella "servizio_gruppo" indica il gruppo di afferenza dei mebri del dipartimento.

Oltre alle tabelle è descritte, è di particolare interesse il contenuto della vista paper_author che è realizzata attraverso un join tra le tabelle paper, author e people. Tale vista conterrà un record per ogni coppia "pubblicazione,autore": sarà quindi possibile avere per la stessa pubblicazione più record, nel caso in cui la pubblicazione è stata redatta da più autori. La fase successiva consiste nell'effettuare le query sul server SQL per ottenere i dati delle tabelle descritte in precedenza. Le query effettuate sono descritte nella tabella 5.1.

Query	**Significato**
Select * from paper_author	Seleziona tutto il contenuto della vista paper_author
Select * from people	Seleziona tutto il contenuto della tabella people
Select * from qualifica	Seleziona tutto il contenuto della tabella qualifica
Select * from quartile	Seleziona tutto il contenuto della tabella quartile
Select * from servizio s, people p where s.matricola = p.matricola	Effettua il join tra le tabelle servizio e people e seleziona tutto il risultato
Select * from paper_author p, quartile q where p.issn = q.issn and p.year = q.year and description LIKE "%rivista"	Effettua il join tra le tabelle paper_author e quartile e seleziona tutti le righe in cui il campo description contiene la parola rivista.
Select * from servizio_gruppo	Seleziona tutto il contenuto della tabella servizio_gruppo

Tabella 5.1: Query effettuate sull'istanza locale del server MySQL

I risultati di tutte le query effettuate sono salvati su dei file in formato json: i file json vengono importati su mongoDB attraverso il comando "mongoimport" all'interno del database chiamato Database_dauin. Attraverso questa procedura, all'interno del db sarà creata una collection per ogni file json e sarà caricato all'interno di esso il contenuto ottenuto tramite la query. Questa procedura è attualmente realizzata in modo manuale: successivamente dovrà essere automatizzata attraverso uno script

in modo che il server mongoDB possa essere aggiornato in modo periodico con i dati presenti sul server MySQL.

5.2 Struttura del server web

Il server Web-Apache, come descritto nel capitolo 3, è costituito un back-end e da un front-end:

- Il back-end è realizzato attraverso linguaggio php e ha il compito di esporre i dati presenti in MongoBD.
- Il front-end è realizato attraverso linguaggio javascript e ha il compito di caricare l'interfaccia grafica e gestire l'interazione con l'utente.

5.3 Back-end php

Il back-end del sistema è realizzato attraverso il linguaggio PHP. Il compito del back-end è quello di leggere ed esporre i dati presenti in mongoDB. Ciò garantisce vari vantaggi:

- Una maggior sicurezza del sistema: l'utente si interfaccerà con le API esposte dal back-end e non avrà di conseguenza alcun interazione con il db.
- E' possibile effettuare operazioni di elaborazione, manipolazione ed aggregazione dei dati del db, minimizzando la quantità di dati da inviare all'utente e la complessità dello script che deve essere eseguito sul client.

L'url di base attraverso cui si può accedere alle API è "indirizzo_IP:porta\Server-web\", dove l'indirizzo IP è l'indirizzo della macchina sui cui è installato il sistema e porta è la porta che è stata settata nel docker-compose file. Nella tabella 5.2, sono indicate le API esposte dal back-end.

API	**Parametri aggiuntivi inviabili attraverso le variabili $_GET**	**Descrizione**
\pubblicazione\readAll.php	\	Restituisce tutte le pubblicazioni presenti nel DB.
\pubblicazione\readOne.php	matricola, da_anno, a_anno	Restituisce tutte le pubblicazioni presenti nel DB, in funzione della matricola e/o degli anni inseriti.

\pubblicazione\readRivista.php	\	Restituisce tutte le pubblicazioni di tipo rivista presenti nel DB.
\coautori\readAll.php	matricola	Restituisce i coautori con cui ha collaborato la matricola cercata. In particolar modo si ha un record per ogni collaborazione.
\coautori\readAnni.php	matricola,da_anno,a_anno	Restituisce i coautori con cui ha collaborato la matricola cercata nel range di anni inserito. In particolar modo si ha un record per ogni collaborazione.
\servizio_gruppo\readOne.php	matricola	Restituisce il gruppo di afferenza della matricola cercata.
\people\readAll.php	\	Restituisce tutti i membri del dipartimento.
\people\readOne.php	matricola	Restituisce tutte le informazioni relative alla persona cercata.
\pubblicazione_quartile\readAll.php	\	Restituisce tutte le pubblicazioni presenti nel DB, con i relativi quartili.
\pubblicazione_quartile\ readOne.php	matricola	Restituisce tutte le pubblicazioni presenti nel DB, con i relativi quartili, in funzione della matricola inserita.
\qualifica\readAll.php	\	Restituisce tutte le qualifiche esistenti nel db.
\quartile\readAll.php	\	Restituisce tutti quartili presenti nel DB.

\quartile\readOne.php	matricola, year	Restituisce tutti i quartili per l'issn e l'anno inserito.
\servizio\readAll.php	\	Restituisce tutte le informazioni relative al ruolo assunto dai membri del dipartimento nel corso del tempo.
\servizio\readOne.php	matricola	Restituisce tutte le informazioni relative al ruolo assunto dalla persone cercata nel corso del tempo.
\wordcloud\readMatricola.php	matricola	Restituisce le 20 parole più frequenti presenti negli abstract e nei title della persona cercata.
\wordcloud\ readMatricolaAnni.php	matricola,da_anno,a_anno	Restituisce le 20 parole più frequenti presenti negli abstract e nei title della persona cercata nel range di anni inserito.
\wordcloud\ readMatricolaAnniTitle.php	matricola	Restituisce le 20 parole più frequenti presenti nei title della persona cercata nel range di anni inserito.

Tabella 5.2: API esposte dal backend

Effettuando una query ad una delle API appena descritte, si otterrà una risposta in formato JSON.

5.4 Front-end javascript

Il front-end del sistema è realizzato in linguaggio javascript. Il suo compito è quello di gestire l'interazione con l'utente, interrogare il back-end per ottenere i dati richiesti e rappresentare graficamente i risultati.

Per raggiungere gli obiettivi descritti in precedenza, sono state utilizzate varie librerie:

- linq.js è una libreria che permette di effettuare query SQL-like su dati in formato json (il formato in cui il back-end php restituisce i dati);
- chart.js e apexcharts sono delle librerie che permettono la rappresentazione grafica dei dati.

Il codice html è realizzato sfruttando il framework bootstrap, che contiene molti stili predefiniti per numerosi elementi html e classi. Nelle sotto-sezioni successive saranno descritte le varie parti della dashboard.

5.4.1 Input form

L'input form ha il compito di gestire l'interazione con l'utente. E' costituito da una serie di menù a tendina:

- Ruolo: permette di specificare il ruolo accademico della persona da ricercare. I ruolo sono raggruppati e ordinati in base al rango. Il ruolo di default è "Tutti": in tal caso nel menù persona saranno incluse tutte le persone afferenti al dipartimento. Nel caso in cui venga selezionato un ruolo diverso, il menù persona sarà aggiornato includendo solamente le persone del dipartimento che hanno quello specifico ruolo.
- Anni: contiene 2 opzioni "Tutti" e "Intervallo". Nel caso in cui è selezionata l'opzione "Intervallo", compariranno attraverso un'animazione i menù "Da" e "A" che danno la possibilità di specificare l'intervallo di ricerca e la checkbox "Includi pubblicazioni in stampa" che permette di specificare se mostrare le pubblicazioni in corso di stampa. L'opzione "Tutti", invece, dà la possibilità di cercare tutte le informazioni presenti nel database per la persona cercata.
- Persona: permette di specificare la persona da ricercare. Il form di default è vuoto.

Il tasto "CERCA" avvierà la ricerca e aggiornerà il contenuto della pagina.
Un esempio dell'input form è rappresentato nella figura 5.2.

Figura 5.2: Form di input della dashboard

5.4.2 Profilo

La sezione "Profilo" ha il compito di riassumere le informazioni principali relative alla persona cercata. In particolar modo, oltre al nome e al cognome, sarà indicato il ruolo al 31/12 dell'anno precedente a quello corrente e il numero di pubblicazione totali. Infine è presente il link alla pagina Iris e alla pagina Orcid del ricercatore. Un'esempio è mostrato in figura 5.3.

Figura 5.3: Profilo

5.4.3 Wordcloud

La sezione successiva al profilo è dedicata alla rappresentazione della wordcloud. Una wordcloud è una rappresentazione visiva di key-words.

- La wordcloud è calcolata sui titoli e sugli abstract di tutte le pubblicazioni della persona cercata negli anni selezionati. L'elenco di parole ottenuto dai titoli e dagli abstract sarà filtrato in modo da eliminare le stopwords. Successivamente saranno selezionate le 50 parole più frequenti: queste vengono ordinate dalla più frequente alla meno frequente e la loro dimensione sarà scalata in base alla posizione nel ranking ottenuto.

La wordcloud è stata realizzata attraverso l'utilizzo della libreria raggiungibile al seguente link https://github.com/jasondavies/d3-cloud. Un esempio della suddetta sezione è rappresentato in figura 5.4.

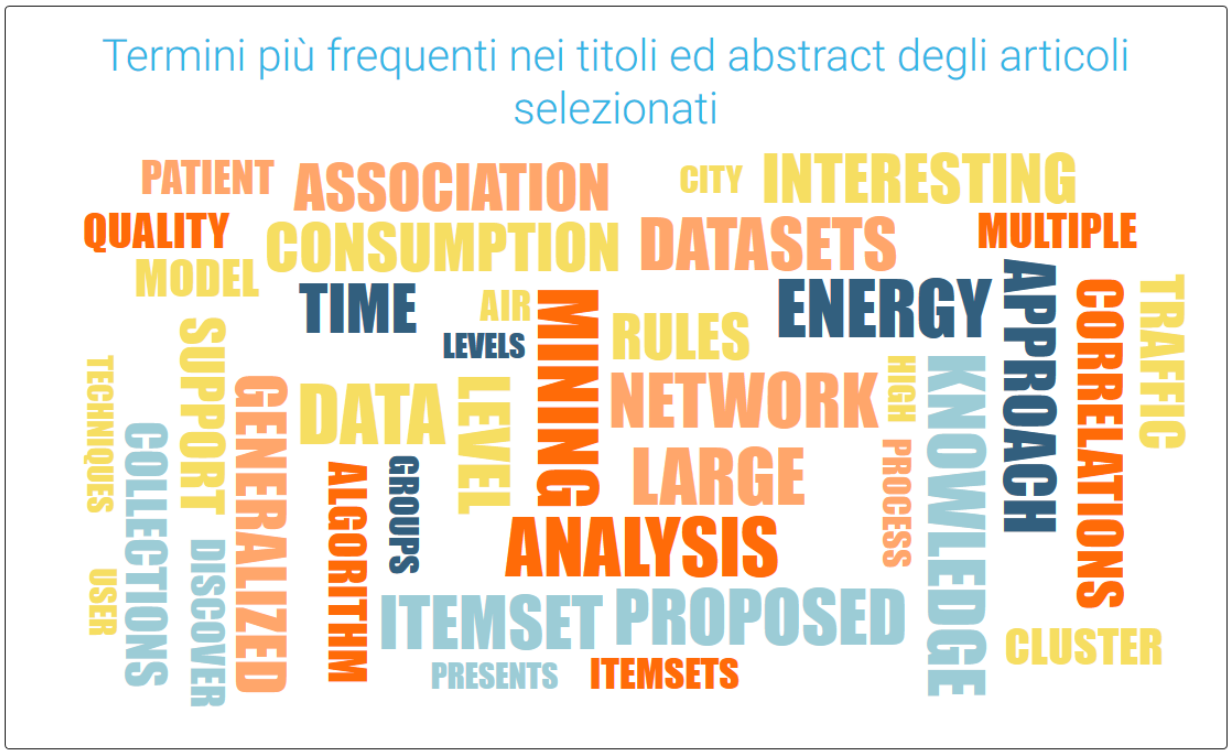

Figura 5.4: Wordcloud

5.4.4 Istogramma delle collaborazioni

In questa sezione è rappresentato un'istogramma orizzontale, il cui scopo è quello di rappresentare i coautori della persona cercata ed indicare il numero di collaborazioni con ognuno di essi. Nell'istogramma i coautori saranno ordinati in base al numero di collaborazioni effettuate e nel caso in cui il numero di contributori sia superiore a 20, non saranno rappresentati coloro che hanno effettuato una sola collaborazione. L'istogramma è stato realizzato attraverso la libreria chart.js. Un esempio del grafo è rappresentato in figura 5.5.

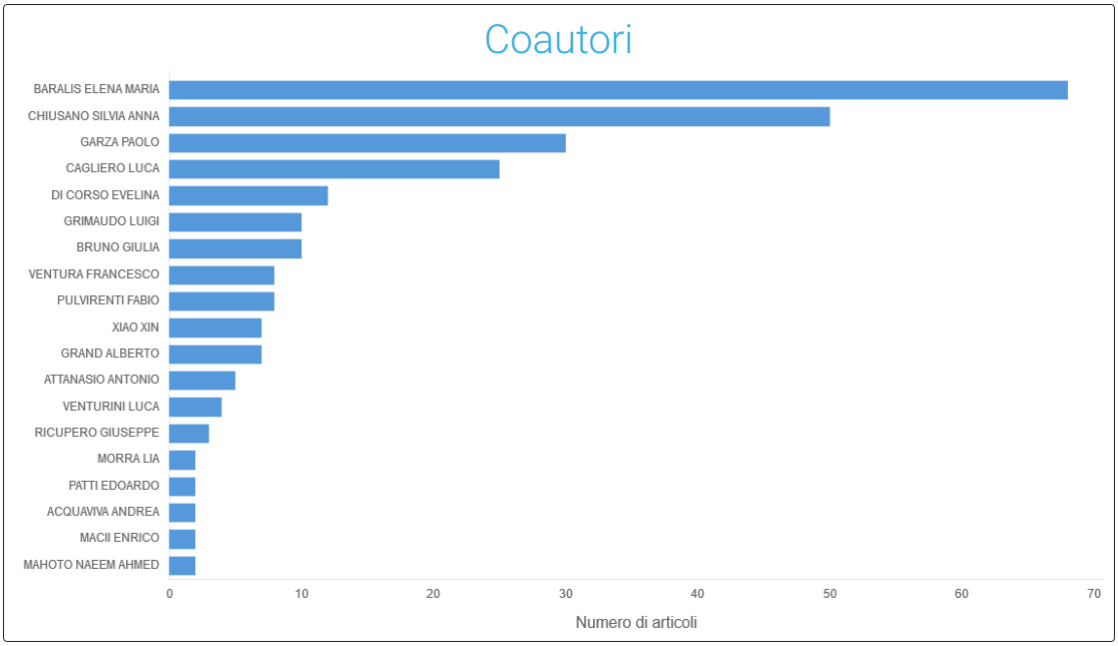

Figura 5.5: Coautori

5.4.5 Grafici sul numero di pubblicazioni

Nella sezione successiva, sono rappresentati 3 grafici:

- Il primo grafico è un grafico a barre, realizzato attraverso la libreria chart.js. Questo grafico rappresenta il numero di pubblicazioni della persona cercata nell'intervallo scelto separatamente per anno e tipo di pubblicazione. Un esempio è mostrato in figura 5.6.

- Il secondo grafico è una heatmap, realizzata attraverso la libreria apexcharts. Questo grafico rappresenta il numero di pubblicazioni della persona cercata nell'intervallo scelto separatamente per anno e numero di coautori. Sull'asse delle ascisse è rappresentato l'anno, mentre sull'asse delle ordinate è rappresentato il numero di coautori: per ogni punto è indicato il numero di pubblicazioni in quell'anno con quel numero di coautori; l'intensità del colore indica la percentuale di pubblicazioni con quel numero di coautori rispetto al numero totale di pubblicazioni di quell'anno. Un esempio è mostrato in figura 5.7.

- Il terzo grafico è uguale al precedente, ma tiene in considerazione solamente le pubblicazioni di tipo rivista. Un esempio è mostrato in figura 5.8.

Al termine della sezione è presente un button che permette di scaricare i dati relativi ai grafici visualizzati in formato json.

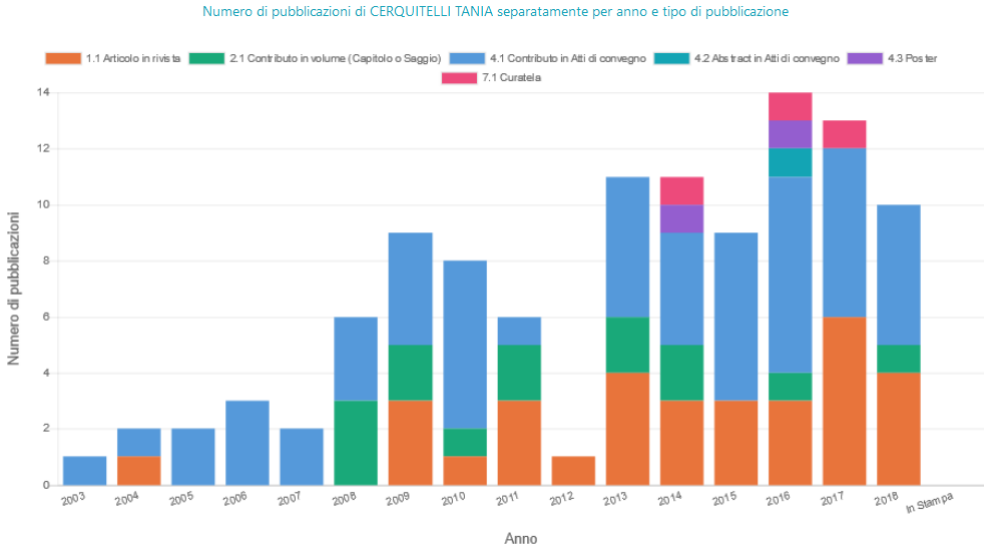

Figura 5.6: Grafico a barre che rappresenta il numero di pubblicazioni separatamente per anno e tipo di pubblicazione.

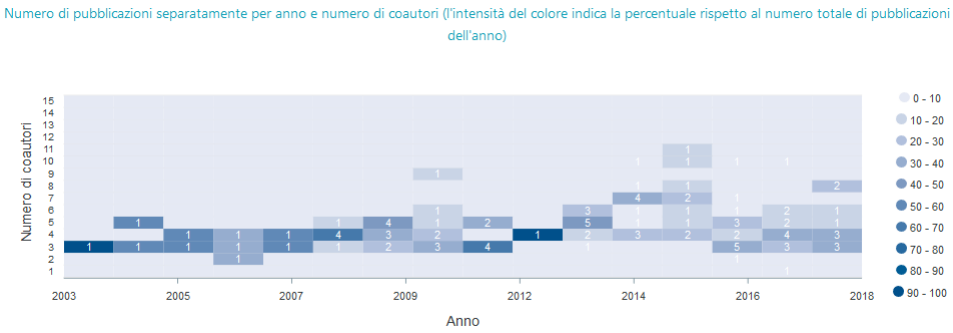

Figura 5.7: Heatmap che rappresenta il numero di pubblicazioni separatamente per anno e numero di coautori.

5.4.6 Grafici sui quartili

La penultima parte della dashboard ha lo scopo di rappresentare le informazioni relative ai quartili delle pubblicazioni. A tale scopo è presente un primo grafico a barre che rappresenta il numero di pubblicazioni della persona cercata nell'intervallo scelto separatamente per anno e quartile. Se ad una pubblicazione, in un determinato anno, sono stati assegnati più quartili, viene scelto quello più alto tra quelli pubblicati su Scimago e WoS.
Al di sotto del grafico è presente un pulsante che permette di scaricare i dati relativi al grafico in formato json.
Un esempio è mostrato in figura 5.9.

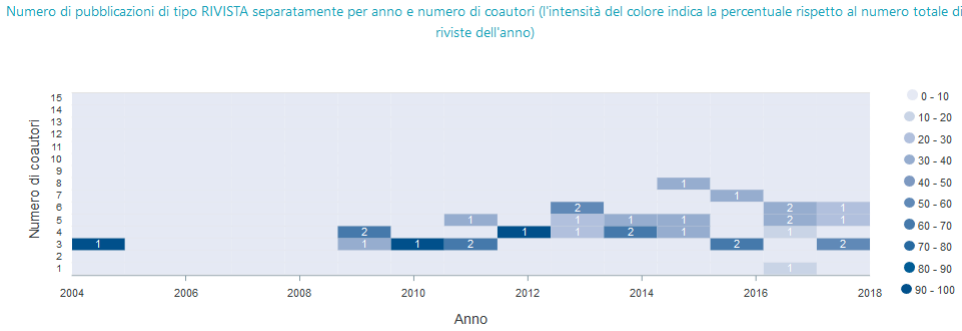

Figura 5.8: Heatmap che rappresenta il numero di pubblicazioni separatamente per anno e numero di coautori, per le pubblicazioni di tipo rivista.

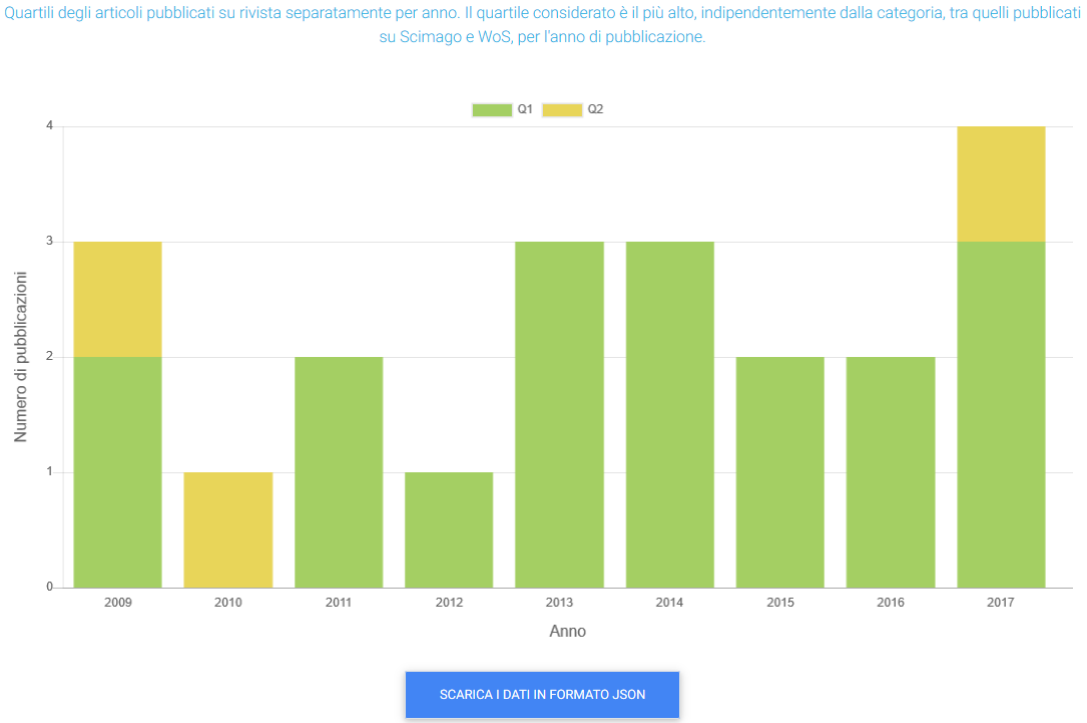

Figura 5.9: Grafico a barre che rappresenta il numero di pubblicazioni separatamente per quartile e anno.

Successivamente è possibile selezionare attraverso una griglia di checkbox le categorie per cui le pubblicazioni sono indicizzate. In tal caso sarà mostrato, alla pressione del tasto "CERCA", un nuovo grafico a barre, analogo al precedente: il quartile considerato sarà il più alto tra quelli pubblicati su Scimago e WoS per le categorie selezionate. E' anche presente il tasto "Seleziona tutto/Deseleziona tutto" che permette di selezionare o deselezionare simultaneamente tutte le checkbox. Un esempio della griglia di checkbox è rappresentato in figura 5.10.

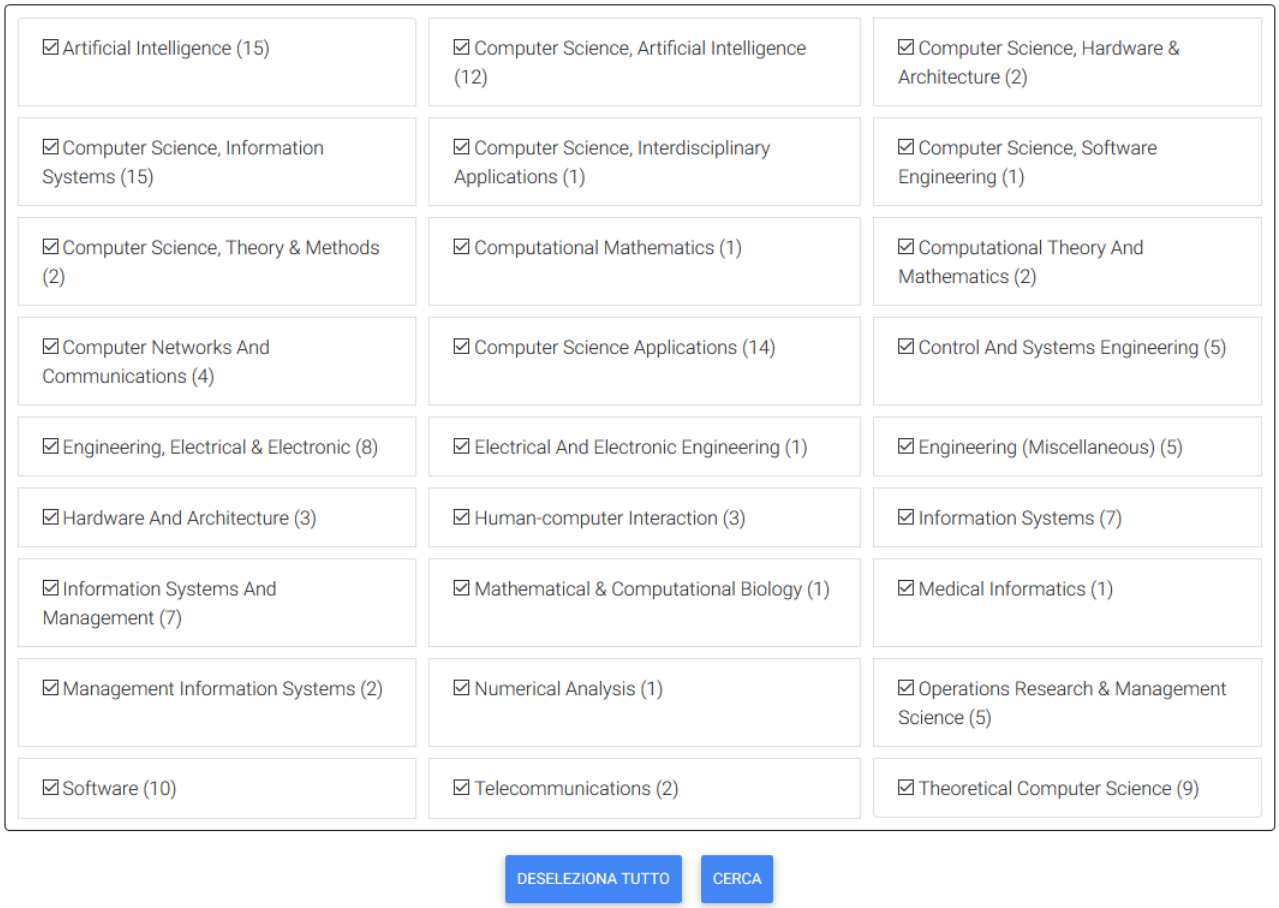

Figura 5.10: Griglia categorie Scopus/Wos degli articoli selezionati.

5.4.7 Ricerca Pubblicazioni

L'ultima sezione della dashboard è stata ideata per permettere all'utente di ricercare specifiche pubblicazioni della persona cercata in funzione di anno e tipo di pubblicazione. In particolare, sono presenti 2 menù a tendina: il primo permette di selezionare l'anno di ricerca, mentre il secondo permette di specificare il tipo di pubblicazione. Alla pressione del tasto "Cerca" comparirà la lista delle pubblicazioni che specificano quei requisiti. Per ogni pubblicazioni saranno mostrati il titolo, il numero di coautori, i coautori e il link della pagina iris.polito relativa alla pubblicazione. Un esempio è mostrato in figura 5.11.

Figura 5.11: Ricerca pubblicazioni della persona ricercata.

Capitolo 6

Conclusioni e sviluppi futuri

In questo lavoro di tesi è stata proposta un'architettura scalabile per il calcolo dei KPI. L'architettura general purpose è stata progettata per essere facilmente utilizzabile in diversi contesti.
L'architettura proposta è stata customizzata in due ambiti specifici:

- Analisi dell'efficienza produttiva in un contesto Industry 4.0;
- Analisi dell'efficienza produttiva degli afferenti ad un dipartimento universitario.

Attraverso l'architettura realizzata è stato possibile effettuare un'analisi su dati reali.
Per quanto riguarda il caso di studio Procemsa, è stato scelto l'OEE come KPI più significativo per il monitoraggio dell'andamento delle prestazioni della linea di produzione. L'OEE è stato calcolato in modo dinamico, ad intervalli da 10 minuti, per l'intera collezione di dati. Sono stati analizzati due dataset reali generati su due delle macchine presenti sulla linea Pluridose, con un numero di record pari a 83931. Le tecnologie utilizzate, come Spark o Cassandra, hanno consentito la realizzazione di un'architettura altamente scalabile, requisito fondamentale, vista la grande mole di dati prodotti dai sensori presenti lungo l'impianto. Zeppelin ha, invece, permesso, grazie all'innata compatibilità con Apache Spark e Apache Cassandra, la produzione di una dashboard riassuntiva, costituita da una serie di grafici facili da leggere ed efficaci nell'obiettivo di monitorare le prestazioni.
Il cruscotto pubblicazioni del Dipartimento di Automatica e Informatica è stato, invece, ideato per realizzare un unico strumento accessibile a tutti che riuscisse a inglobare le varie informazioni presenti sul web. Navigare i dati relativi alle proprie pubblicazioni è, infatti, una funzione già disponibile nel web attraverso i vari portali, quali IRIS.polito, Scimago, Wos. L'obiettivo era quello di riuscire a sintetizzare le informazioni raccolte dalle varie fonti all'interno di una singola dashboard, attraverso dei grafici immediati e di facile lettura. L'analisi è effettuata sulla produzione scientifica degli afferenti al DAUIN in un periodo che va dal 1962 al 2019. In particolare sono stati analizzati:

- 7318 pubblicazioni(articoli in rivista, contributi in volume, monografie, trattati scientifici, contributi in atti di convegno, poster, brevetti, ecc);
- 1987 articoli pubblicati su 702 riviste differenti e associati a 269 categorie distinte su Scimago e Wos;
- 511 afferenti;

Anche in questo caso, si è sfruttato un DBMS molto scalabile come MongoDB ed è stato di grande aiuto l'elevato numero di librerie disponibili nel Web, fondamentali per la realizzazione dei vari grafici.
Per uno sviluppo molto più rapido dell'architettura è stato di grande ausilio l'utilizzo dell'innovativa piattaforma Docker la quale ha permesso di minimizzare il tempo necessario per il deploy delle varie applicazioni e per il trasferimento delle immagini dalla macchina di sviluppo e testing a quella reale.

Sviluppi Futuri In ottica futura, potranno essere effettuati degli aggiornamenti sulle applicazioni progettate.
L'applicazione realizzata per il calcolo dei KPI in Procemsa dovrà essere dapprima testata su un insieme di dati più grande, i quali saranno inseriti in Cassandra in modo automatico attraverso delle API rest sviluppate da un partner del progetto. Ciò permetterà la possibilità di valutare la validità delle varie supposizioni effettuate nel calcolo dell'OEE. In un secondo momento dovrà essere estesa in modo da permettere il calcolo dell'OEE anche per la macchina astucciatrice, situata a valle dell'etichettatrice, e per l'intera linea di produzione.
Per quanto riguarda la dashboard del DAUIN, è previsto un aggiornamento attraverso cui saranno calcolati nuovi KPI legati alla produttività del gruppo, piuttosto che alle attività contrattuali e didattiche svolte dai vari afferenti. Inoltre sarà necessario automatizzare, attraverso uno script, la fase di data ingestion, attraverso cui avviene il caricamento dei dati su MongoDB a partire da quelli originali, presenti sul server SQL.

Bibliografia

[1] *Apache http server project*. URL: https://httpd.apache.org/.

[2] Jeffrey Aven. *Apache Spark in 24 Hours, Sams Teach Yourself*. Sams Publishing, 2016.

[3] Gediminas B. *What is Apache? An In-Depth Overview of Apache Web Server*. URL: https://www.hostinger.com/tutorials/what-is-apache.

[4] Kyle Banker. *MongoDB in action*. Manning Publications Co., 2011.

[5] *Big Data*. URL: https://it.wikipedia.org/wiki/Big_data.

[6] Mat Brown. *Learning Apache Cassandra*. Packt Publishing Ltd, 2015.

[7] *Cassandra (database)*. URL: https://it.wikipedia.org/wiki/Cassandra_(database).

[8] *Cos'è Docker?* URL: https://www.redhat.com/it/topics/containers/what-is-docker.

[9] *Data Visualization Using Apache Zeppelin – Tutorial*. 2017. URL: https://scalegrid.io/blog/data-visualization-using-apache-zeppelin/.

[10] *FAQ*. URL: https://www.oee.com/faq.html.

[11] Paolo Di Medio. *Significato, definizione e calcolo dell'OEE*. URL: https://www.organizzazioneaziendale.net/oee-significato-definizione-calcolo/2671.

[12] *MongoDB*. URL: https://it.wikipedia.org/wiki/MongoDB.

[13] Ellis New. *OEE – Learn How to Use It Right*. URL: https://www.industryweek.com/quality/oee-learn-how-use-it-right.

[14] *OEE: Overall Equipment Effectiveness*. URL: https://www.leanmanufacturing.it/strumenti/oee.html.

[15] *Performance measurement using overall equipment effectiveness (OEE): Literature review and practical application discussion*. URL: https://hal.archives-ouvertes.fr/hal-00512968/document.

[16] Christof Strauch, Ultra-Large Scale Sites e Walter Kriha. "NoSQL databases". In: *Lecture Notes, Stuttgart Media University* 20 (2011).

[17] *the fast guide to oee*. URL: https://www.vorne.com/pdf/fast-guide-to-oee.pdf.

[18] *What is Overall Equipment Effectiveness?* URL: `https://www.oee.com/`.

Ringraziamenti

In questo giorno così speciale vorrei ringraziare tutte le persone che mi sono state vicine in questo percorso universitario.

Innanzitutto, desidero ringraziare la mia famiglia, il mio punto di riferimento. Grazie ai miei genitori e a mia sorella per essermi stati sempre vicini. Grazie per il supporto, per i tanti sacrifici e per i consigli. Il traguardo da me raggiunto è merito vostro.

Un grazie speciale alla mia Chiara. Grazie per avermi supportato, ascoltato e consigliato durante questi cinque anni. Grazie per esserci sempre stata nonostante la distanza.

Un sentito ringraziamento a mia nonna Giovanna per l'affetto e per avermi sempre sostenuto in questi anni.

Grazie a John, di gran lunga il miglior coinquilino che potessi scegliere. Grazie per i tanti momenti insieme e per le nostre infinite chiacchierate.

Un grazie a tutti gli amici che mi hanno accompagnato durante questo percorso.

Un sentito ringraziamento alla professoressa Cerquitelli per la gentilezza, la pazienza e la competenza. Grazie per il supporto e l'aiuto fornitomi nella realizzazione di questa tesi.