ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Scuola di Scienze
Corso di Laurea in Ingegneria e Scienze Informatiche

Progettazione di un sistema ETL distribuito per l'elaborazione di immagini satellitari in ambito Precision Farming

Elaborato in
Laboratorio di Basi di Dati

Relatore
Prof. Matteo Golfarelli

Presentata da
Alessandro Cevoli

Seconda Sessione di Laurea
Anno Accademico 2015 – 2016

PAROLE CHIAVE

Big Data

Copernicus

Precision Farming

MoReFarming

ETL

A tutti coloro che mi hanno accompagnato in questo percorso
Alla mia famiglia
Alla mia compagna di merende

Indice

Introduzione

Viviamo in un'era in cui l'informazione è pervasiva e open, accessibile ovunque e on-demand. Sono diventate ormai la normalità, all'interno di ogni settore del mercato e della ricerca, le soluzioni IT che sfruttano in modo efficace queste informazioni analizzandole, catalogandole, elaborandole e distribuendole per gli scopi più vari, spesso di stampo economico. Gli effetti sono di fatto reali e sensibili e perciò non più ignorabili all'interno del panorama sociale ed economico. Si tratta del campo di ricerca e sviluppo dei così detti Big Data, una disciplina della Computer Science dedita allo studio dell'enorme flusso di dati che ogni secondo viene generato dalla moderna società e che costituisce un'informazione vasta, utile e preziosa. Lo scopo delle soluzioni Big Data è quello di imbrigliare l'informazione e renderla utilizzabile, mantenibile, socialmente ed economicamente utile.

Considerare questa vasta gamma di informazioni come i soli dati generati dall'essere umano tramite l'uso di Internet, e quindi social media, transizioni finanziarie, ricerche tramite web browser o su siti di shopping etc., non porta alla sua massima espressione. Mentre gli esseri umani, in quanto creature senzienti e consce della propria esistenza, creano inevitabilmente flussi di dati nel corso del loro vivere quotidiano, esistono altri soggetti che possono essere una potenziale sorgente di informazione, inesauribile, solo utilizzando i giusti strumenti. Un esempio sono i satelliti geo-stazionari che forniscono informazione *in tempo quasi reale ed in maniera globale (e non più locale)* sul nostro pianeta come il livello degli oceani, la composizione e qualità dell'aria, i livelli di Ozono, la temperatura atmosferica e informazioni di carattere climatico come il movimento dei fronti nuvolosi. L'esempio più vicino è l'attuale programma Copernicus nato dalla collaborazione tra Comitato Europeo e l'Agenzia Spaziale Europea e per il quale sono già stati lanciati in orbita 6 delle 12 coppie di satelliti per il monitoraggio dell'ambiente. Altre sorgenti d'informazione "silenziosa" sono date dai sistemi di sensori attraverso i quali si esegue il monitoraggio delle coltivazioni per ricavarne dati utili allo sviluppo e crescita della coltivazione come ad esempio i livelli d'acqua e azoto nel terreno, la presenza di sostanze, batteri, virus o animali nocivi, il peso dei vegetali prodotti e in quale quantità, le qualità intrinseche possedute dal terreno di coltura etc.

Le informazioni da noi prodotte, come ad esempio il nostro stile di vita quotidiano e i nostri gusti (in generale), sono più utili al mercato di consumo, al fine di rilevare i trend di mercato giusti e massimizzare così le vendite, piuttosto che ad un uso rivolto al miglioramento della qualità di vita. Al contrario, le informazioni raccolte dalle fonti come satelliti e sistemi di monitoraggio vengono utilizzate da soluzioni che mirano a migliorare la qualità di vita, in modo sia diretto che indiretto, non solo dal punto di vista economico di chi produce ma anche di chi consuma. Si parla dunque di un nuovo campo di ricerca e di mercato chiamato Agricoltura di Precisione, il cui proposito principale è quello di creare un modello di gestione agraria basato sui Big Data che sia autonomo, automatizzato, efficiente ed ottimizzato, capace di generare un nuovo mercato economicamente, socialmente ed ecologicamente sostenibile, sfruttando tecnologie all'avanguardia e le informazioni raccolte tramite i satelliti e le reti di sensori. Mentre in U.S.A. esistono già da tempo soluzioni orientate verso questa strada, anche per una questione di fabbisogno del paese e di una migliore gestione delle immense coltivazioni estensive, in Italia un esempio è dato dall'attuale progetto in fase di sviluppo chiamato MoReFarming, il quale punta a sfruttare i satelliti del programma Copernicus per fornire dati statistici sulla crescita e qualità dei raccolti.

All'interno di questa tesi progettuale si affronterà il problema dell'automatizzazione e distribuzione di un processo ETL (Transform-Extract-Load) per la piattaforma Open e Big Data chiamata MoReFarming, progetto nato dalla collaborazione tra Università degli studi di Bologna, Università degli studi di Piacenza e il Centro di Ricerca per le Produzioni Vegetali. L'applicativo dovrà scaricare i dati dal server remoto di Copernicus, elaborarli applicando le adeguate correzioni attraverso una pletora di tool (Sen2Cor e GDAL) ed infine caricarli sia su Hadoop Distributed File System sia sul database geo-relazionale PostGIS. Il prototipo di applicativo dovrà inoltre poter eseguire sia in locale che distribuito, in modo tale da ottimizzare i tempi di esecuzione e garantire sempre l'esecuzione del processo ETL. Le operazioni eseguite nel processo di trasformazione saranno relative alla classificazione della scena e correzione atmosferica, conversione in formato GeoTiff e relativa suddivisione in tiles. Su HDFS verranno caricati tutti i dati generati ai vari step mentre su PostGIS i le immagini GeoTiff sezionate.

Questa tesi è strutturata in 3 capitoli: il primo capitolo tratterà dell'Analisi dati e dell'Agricoltura di Precisione, esponendo quali sono le principali tematiche trattate da ciascun campo; il secondo capitolo tratterà delle tecnologie sfruttate all'interno del progetto, spiegandone i meccanismi e le funzioni; il terzo e ultimo capitolo tratterà del prototipo di applicativo realizzato per questo progetto, ne verrà esplicata l'architettura generale e il funzionamento dei principali moduli da cui è composto.

Capitolo 1

Analisi dati e Agricoltura di precisione

1.1 Esa e Sentinel 2

1.1.1 European Space Agency

L'Agenzia Spaziale Europea (ESA) [4] è una organizzazione intergovernativa dedita all'esplorazione spaziale e alla quale collaborano 22 tra gli stati membri dell'unione Europea. Istituita il 30 maggio 1975 e con sede amministrativa a Parigi, Francia, conta uno staff di circa 2000 persone e un budget annuale di Euro 5.25 miliardi. Il programma di volo dell'ESA prevede viaggi spaziali in collaborazione con il programma della Stazione Spaziale Internazionale (ISS), il lancio e operazione di esplorazioni di altri pianeti e satelliti tramite sonde e robot, osservazione terrestre, esperimenti scientifici e di telecomunicazione, progettazione di veicoli di lancio e il supporto economico e amministrativo del Centro Spaziale della Guiana Francese a Kourou.

Le sue strutture principali sono distribuite tra 5 centri di ricerca:

- Le missioni di stampo scientifico hanno luogo al centro ESTEC a Noordwijk, Netherlands;
- Le missioni di osservazione terrestre hanno sede al centro ESRIN a Frascati, Italia;
- Il centro di controllo delle missioni (ESOC) è situato a Darmstadt in Germania;
- Il centro di istruzione e allenamento degli astronauti (EAC) è a Colonia, Germania;

- Il centro di osservazione Astronomica (ESAC) è localizzato a Villanueva de la Cañada, Madrid, Spain.

Missione

La convenzione di Parigi del 30 Maggio del 1975 che istituisce la figura dell'ESA[1] così si esprime a riguardo dell'obiettivo fondamentale dell'agenzia:

> "*ESA's purpose shall be to provide for, and to promote, for exclusively peaceful purposes, cooperation among European States in space research and technology and their space applications, with a view to their being used for scientific purposes and for operational space applications systems:*
>
> - *by elaborating and implementing a long-term European space policy, by recommending space objectives to the Member States, and by concerting the policies of the Member States with respect to other national and international organisations and institutions;*
> - *by elaborating and implementing activities and programmes in the space field;*
> - *by coordinating the European space programme and national programmes, and by integrating the latter progressively and as completely as possible into the European space programme, in particular as regards the development of applications satellites;*
> - *by elaborating and implementing the industrial policy appropriate to its programme and by recommending a coherent industrial policy to the Member States."*

L'ex-direttore generale dell'ESA Jean-Jacques Dordain espresse il seguente parere in una intervista del 2003 a riguardo la missione dell'Agenzia:

> *"Today space activities are pursued for the benefit of citizens, and citizens are asking for a better quality of life on earth. They want greater security and economic wealth, but they also want to pursue their dreams, to increase their knowledge, and they want younger people to be attracted to the pursuit of science and technology. I think that space can do all of this: it can produce a higher quality of life, better security, more economic wealth, and also fulfill our citizens' dreams and thirst for knowledge, and attract the young*

[1] `http://www.kosmos.gov.pl/download/ESA_Convention.pdf`

generation. This is the reason space exploration is an integral part of overall space activities. It has always been so, and it will be even more important in the future."

1.1.2 Copernicus Project

Copernicus [3] è un progetto nato dalla collaborazione tra la Commissione Europea ed ESA con lo scopo di osservare l'ambiente terrestre, collezionare i dati raccolti, analizzare i dati collezionati e rendere infine disponibili queste informazioni a tutti, in maniera libera e gratuita, in modo tale da migliorare l'auto-consapevolezza dell'uomo riguardo agli eventi naturali, e non, che hanno luogo sul nostro pianeta, creare una mappa comprensiva e dettagliata dello stato di salute del nostro pianeta e per elevare i settori industriali ad un nuovo stadio evolutivo sostenibile sia a livello economico che ecologico. Questo sistema di monitoraggio è reso disponibile grazie alla pletora di satelliti geostazionari chiamati Sentinels.

Servizi

I principali servizi che verranno messi a disposizione da Copernicus sono:

- **Monitoraggio Atmosferico** (Es.: Rilevamento dei Particolati detti anche Aerosol e qualità dell'aria).
- **Monitoraggio dell'ambiente marino** (Es: Qualità dell'acqua e livello dei mari).
- **Monitoraggio dell'ambiente terrestre** (Es.: Monitoraggio Aree Boschive).
- **Monitoraggio dei cambiamenti climatici** (Es.: Monitoraggio dello scioglimento dei ghiacci ai poli).
- **Gestione delle Emergenze Ambientali** (Es.: Incendi, Inondazioni, Terremoti)
- **Sicurezza Ambientale**

Le missioni Sentinels

L'ESA sta attualmente sviluppando 7 missioni per il progetto Sentinel. Le varie missioni Sentinels includono la ripresa di immagini radar e spettrali del

terreno e dell'oceano, e monitoraggio atmosferico. Ogni missione è caratterizzata dalla collaborazione di 2 satelliti per garantire e rispettare i requisiti di ogni missione, al fine di fornire dataset robusti per tutti i servizi di Copernicus.

Le missioni Sentinel hanno i seguenti obiettivi:

- La missione Sentinel-1 opera giorno e notte per eseguire la cattura di immagini, per i servizi di monitoraggio del terreno e degli oceani, attraverso il SAR (Synthetic Aperture Radar). Questo strumento rende possibile l'acquisizione d'immagini qualunque sia il tempo atmosferico.

- La missione Sentinel-2 riprende immagini ad alta risoluzione per servizi terrestri (es.: livello vegetazione, suolo e bacini d'acqua, fiumi e canali interni, aree costiere) e per servizi di emergenza. La missione monitorerà la variabilità delle condizioni superficiali del terreno e grazie alla grande area di ripresa ed il suo tempo di rivoluzione (10-5 giorni) rende possibile monitorare le variazioni di crescita della vegetazione con l'avanzare delle stagioni in maniera costante. I satelliti che compongono il sistema di Sentinel-2 possiedono 2 orbite polari con una fase di 180° tra loro.

- La missione Sentinel-3 garantirà servizi di monitoraggio oceanici e del territorio globale. Il suo obiettivo principale è quello di misurare la topografia e temperatura superficiali di mari ed oceani, definire con grande precisione i colori delle superfici di terreno e oceano, garantire un supporto affidabile per i servizi di previsione e di monitoraggio ambientale e climatico.

- Sentinel-4 metterà a disposizione dati per il monitoraggio continuo della composizione atmosfera. La missione si focalizzerà sulla qualità dell'aria, monitorando livelli di Ozono (O3), Diossido di Azoto (NO2), Diossido di Zolfo (SO2), Formaldeide (HCHO) e profondità dei particolati (Aerosol). La missione non è attualmente attiva e il lancio è previsto per il 2021.

- Il Sentinel-5 Precursor è uno dei satelliti che faranno parte di Sentinel-5 ed il cui lancio è previsto nei primi mesi del 2017. Il suo principale scopo è quello di ridurre il gap di raccolta dati (nella fattispecie le osservazioni atmosferiche SCIAMACHY) che si è creato dopo la conclusione della missione EnviSat nel 2012. Inoltre avrà il compito di effettuare misurazioni atmosferiche ad alta risoluzione spazio-temporale in relazione a qualità dell'aria, cambiamenti climatici, livelli di Ozono e radiazioni UV. Il satellite è equipaggiato con lo spettroscopio UV-VIS-NIR-SWIR soprannominato TROPOspheric Monitoring Instrument (TROPOMI).

L'intero sistema Sentinel-5 orbiterà ai poli, il che permetterà di raccogliere dati riguardanti le concentrazioni di gas per applicazioni di stampo chimico e climatico.

- Il satellite Sentinel-6 è un altimetro radar con l'obiettivo di misurare con grande precisione il livello globale del livello del mare. Un secondo obiettivo di Sentinel-6 è quello di eseguire una profilazione delle temperature ai vari livelli dell'atmosfera attraverso l'uso del GNSS (Global Navigation Satellite System) Radio-Occultation sounding technique.

1.1.3 Sentinel 2 Multi Spectral Instrument

Entriamo ora più nello specifico parlando del satellite preso in causa all'interno del progetto in cui è coinvolta questa tesi. I satelliti Sentinel-2 sono una missione di ripresa d'immagini ad ampio raggio (290 Km), ad alta risoluzione e multi-spettrali, all'interno della quale sono inclusi: il monitoraggio della vegetazione, del suolo e dei bacini d'acqua, dei fiumi e dei canali artificiali. I satelliti sono capaci di catturare fino a 15000 Km in modalità di acquisizione continua, catture che verranno poi discretizzate in granuli più piccoli durante le varie fasi di elaborazione dei vari prodotti.

Risoluzioni utilizzate

I satelliti rendono disponibili misurazioni con le seguenti risoluzioni:

- Risoluzione temporale è la frequenza con cui un satellite visita un luogo (ovvero il periodo di rivoluzione del satellite lungo la propria orbita attorno alla terra). Tale frequenza per i singoli satelliti è pari a 10 giorni, mentre quella combinata è pari a 5 giorni.

- La risoluzione spaziale è la singola rappresentazione a livello del suolo catturata da un singolo rilevatore all'interno dell'array di sensori a disposizione di un satellite.

- La risoluzione Radiometrica può essere definita come la capacità di uno strumento di rilevare differenze nell'intensità della componente luminosa o nel coefficiente di riflessione. Essa è determinata come intervallo incrementale dell'intensità luminosa o del coefficiente di riflessione che può essere rappresentata o distinta dal sistema. Maggiore è la risoluzione radiometrica, migliore sarà la capacità dello strumento di rilevare differenze nell'intensità luminosa o nel coefficiente di riflessione.

Zona di copertura fornita dal satellite

Il satellite garantisce la copertura delle seguenti aree:

- Tutte le superfici delle aree continentali, inclusi fiumi e laghi, in una latitudine compresa tra 56° Sud e 83° Nord;
- Tutte le acque costiere fino a 20 Km dalla riva;
- Tutte le isole con una superficie maggiore di $100Km^2$;
- Tutte le isole Europee, indipendentemente dalla dimensione;
- Tutto il Mare Mediterraneo;
- Tutti i mari chiusi (Mar Caspio, Mar Morto).

1.2 Agricoltura di Precisione

L'agricoltura di precisione è un modello di gestione delle colture basato sull'osservazione e la raccolta dati per rispondere al meglio alle necessità mutevoli delle coltivazioni. L'obiettivo della ricerca nell'agricoltura di precisione è quello di definire un sistema di supporto alle decisioni che permetta di gestire l'intera coltura in modo autonomo, automatico e ottimizzato dal punto di vista dello sfruttamento delle risorse. Si tratta di un approccio fito-geomorfologico che studia la stabilità di crescita durante gli anni in congiunzione agli attributi topologici del territorio, i quali sono indice delle risorse necessarie a quel particolare terreno per rendere al meglio.

L'avvento dell'agricoltura di precisione si può rimandare alla nascita dei GPS e GNSS, che hanno consentito agli agricoltori e/o ricercatori di individuare con precisione le zone di interesse del territorio e associarvi la maggior quantità di dati statistici possibili. Inoltre, ciò è stato possibile anche grazie all'avvento di tecnologie per il monitoraggio, per la gestione temporizzata e variabile della semina, dell'irrigazione, della somministrazione di additivi etc., lo sviluppo di sensori per la raccolta dati come livelli di clorofilla e d'acqua.

Obiettivi

L'agricoltura di precisione punta ad ottimizzare la gestione del territorio in relazione alle seguenti tematiche:

- Scienza delle coltivazioni: avvicinare pratiche di coltivazione ai bisogni del raccolto, o personalizzare la coltivazione per una parte di esso.

- Protezione dell'ambiente: ridurre il rischio di disastri ambientali e l'impatto ecologico delle piantagioni, incentivando quindi l'uso di tecniche sostenibili sia a livello ecologico che economico;

- Incremento della competitività economica attraverso il miglioramento dell'efficienza delle tecniche agricole.

Il processo

L'agricoltura di precisione è solitamente caratterizzata da 4 passaggi fondamentali:

- **Collezione dei dati**: La geo-localizzazione consente ai coltivatori di sovrapporre informazioni raccolte dall'analisi del terreno e dei residui di Azoto con le informazioni sui precedenti raccolti e sulla resistenza del suolo. La localizzazione geografica può avvenire in 2 modalità:

  - Si delinea il terreno utilizzando un ricevitore GPS mobile a cui viene fatto percorrere tutto il perimetro dei terreni interessati.
  - Il terreno è delineato attraverso una mappa costruita utilizzando le rilevazioni aeree o immagini satellitari. Le immagini utilizzate per generare la mappa dovranno perciò avere il giusto livello di risoluzione e qualità geometrica per permettere una geo-localizzazione sufficientemente accurata.

- **Valutazione delle variabili in gioco**: La varianza di fattori esterni ed interni raccolti può risultare in una molteplicità non triviale di fattori da tenere in considerazione. Tra questi fattori sono inclusi: condizioni climatiche (neve, grandine, pioggia, siccità), caratteristiche del terreno (profondità, livelli di azoto e consistenza), pratiche di raccolta, erbe infestanti indigene del territorio e malattie. I fattori indicativi delle caratteristiche del suolo permettono di acquisire informazioni riguardo alle costanti ambientali principali. I fattori puntuali consentono di tracciare lo stato del raccolto in maniera precisa nel tempo. Tali informazioni possono essere raccolte attraverso sensori o stazioni meteo. Le statistiche combinate con la resistenza del suolo rendono possibile anche l'identificazione dei livelli di umidità del terreno.

- **Scelta della strategia**: Attraverso l'uso di mappe di rappresentazione del suolo, i coltivatori possono perseguire 2 possibili strategie per regolare i quantitativi di additivi somministrati:

  - **Approccio predittivo**: questa strategia è basata sull'analisi statistica dei dati forniti dagli indicatori durante il ciclo di crescita della coltura.
  - **Approccio controllato**: in questa strategia la raccolta di dati dagli indicatori è regolarmente aggiornata durante il ciclo di crescita della coltura attraverso il campionamento del peso delle biomasse, del livello di clorofilla nelle foglie, del peso dei frutti, l'osservazione remota della temperatura, dell'umidità, del vento o del diametro degli steli dei vegetali, sfruttando una rete di sensori wireless. E' inoltre utilizzata l'osservazione remota da satellite tramite la cattura di immagini multi-spettrali o elaborate in modo tale da misurare i parametri biofisici del raccolto.

  La scelta della migliore strategia può essere affidata ad un sistema di supporto alle decisioni ma l'analisi finale è solitamente fatta dal coltivatore, decidendo in termini di valore economico ed impatto ambientale.

- **Attuazione della strategia**: Le nuove tecnologie per l'informazione e la comunicazione rendono la gestione dei raccolti un'operazione molto più semplice da eseguire per i coltivatori. Le applicazioni per il supporto alle decisioni in un sistema di gestione del raccolto richiedono un equipaggiamento che supporti Variable-Rate Technologies (VRT) per la semina assieme a Variable-Rate Applications (VRA) per la somministrazione di Azoto e prodotti fitosanitari. Viene inoltre fatto uso delle comuni macchine agricole in congiunzione all'uso di sistemi rilevazione della posizione e sistemi informativi geografici (GIS).

Impatto economico ed ambientale

Come implica il nome, l'obiettivo di questo modello di coltivazione è quello di somministrare alla coltivazione la quantità precisa e corretta di acqua, fertilizzante, pesticidi e altri additivi, ai giusti intervalli temporali in modo tale da incrementare la produttività e massimizzare il rendimento. In maniera collaterale è possibile minimizzare lo spreco delle risorse e di emissione di gas inquinanti, generando anche un maggior apporto economico a favore del coltivatore grazie al minor uso di fitosanitari e fertilizzanti e quindi una minore spesa per l'acquisto di tali additivi. Inoltre l'agricoltura di precisione ha un secondo importante beneficio, sia temporale che qualitativo, che riguarda la riduzione su larga scala dell'impatto ambientale. Applicando le giuste quantità di additivi nel luogo giusto e al momento giusto, si beneficia il raccolto, il suolo e riserve idriche sotterranee. L'agricoltura di precisione è divenuta perciò la chiave di volta dell'agricoltura eco-sostenibile in quanto rispetta le colture,

la componente idro-geologica del territorio e anche i coltivatori stessi da un punto di vista economico. L'agricoltura sostenibile tenta di garantire produzioni alimentari nei limiti ecologici, economici e sociali richiesti a sostenere una produzione a lungo termine, attraverso l'uso di tecnologie all'avanguardia.

1.2.1 Progetto MoReFarming

Il progetto MoReFarming nasce come collaborazione tra l'Università degli studi di Bologna e l'Università degli studi di Piacenza per conto del Centro di Ricerca per le Produzioni Vegetali (CPRV). Il progetto rientra nel contesto del monitoraggio territoriale e aziendale di tutti gli aspetti di variabilità, sia spaziale che temporale, che oggigiorno rappresentano gli aspetti chiave per eseguire una gestione efficace e sostenibile delle colture agrarie estensive ed/o intensive (a.k.a. Precision Farming/Agricoltura Sostenibile).

Perché nasce MoReFarming

Il monitoraggio delle risorse è andato evolvendosi con l'utilizzo degli strumenti che la tecnologia nel tempo ha messo a disposizione (capannine meteo, trappole sessuali, ecc.). Un ulteriore salto qualitativo in termini di accuratezza, tempestività ed estensione è offerto dall'uso di sensori (remoti o locali) che, in combinazione con specifici supporti informativi e/o decisionali, permettono l'applicazione di sistemi e tecniche agricole avanzate, con particolare riguardo all'agricoltura di precisione, nelle sue applicazioni puntuali sito-specifiche nell'ambito della concimazione, dell'irrigazione, della difesa fitosanitaria, di lavorazione del terreno e della raccolta.[13]

Il rilevamento satellitare rappresenta uno strumento efficace e all'avanguardia per l'individuazione delle zone che necessitano di una gestione delle risorse personalizzata e specifica. A partire dall'autunno 2015 l'Agenzia Spaziale Europea (ESA) ha reso disponibili gratuitamente i dati raccolti dal primo dei due satelliti che compongono la missione Sentinel-2. Una importante applicazione del rilevamento satellitare è quello rivolto alle tecniche di agricoltura di precisione orientate al miglioramento dell'efficienza della distribuzione di apporti idrici e fitofarmaci.[13]

Per quanto riguarda la sensoristica locale, il principale svantaggio delle centraline esistenti consiste nella elevata specificità delle applicazioni (solo per umidità, temperatura, etc.) che ne restringono il campo di operatività e ne favoriscono una rapida obsolescenza. Inoltre, l'interoperabilità tra le diverse periferiche avviene spesso per comunicazione via cavo a causa degli elevati consumi della rete da cui i dati vengono acquisiti ma da alcuni anni, ed in particolare nel campo della sicurezza, sono in uso telecamere e sistemi di sensori

periferici a basso consumo con comunicazione wireless. Attualmente la maggior parte dei sensori di visione con connettività wireless sono destinati all'uso generico per le applicazioni di rete, alcuni dei quali progettati con specifiche hardware e di connettività ad elevate prestazioni che richiedono una notevole potenza di alimentazione, che risulta poi in una autonomia particolarmente vincolata.[13]

Per la raccolta dati a supporto del processo decisionale, i lavori esistenti in letteratura evidenziano un dislivello tra le caratteristiche degli attuali Farm Management Information Systems (FMIS) e i requisiti necessari per un loro utilizzo reale nell'ambito dell'agricoltura di precisione. La situazione attuale si caratterizza per la disponibilità di un enorme mole di dati raccolti tramite nuove famiglie di sensori e la contemporanea disponibilità delle tecnologie legate ai Big-Data che ne consentono la gestione. Questo contesto determina un forte gap rispetto ai lavori preesistenti e richiede sia la creazione di nuove architetture hardware e software per la gestione dei dati, sia la revisione dei precedenti modelli di previsione e di supporto al processo decisionale alla luce delle nuove potenzialità.[13]

MoReFarming vuole ricoprire tutti questi aspetti appena descritti, in modo tale da ottenere una piattaforma unica e su larga scala, Open ed orientata ai Big Data, per la gestione eco-sostenibile delle coltivazioni attraverso l'uso delle tecnologie più avanzate a disposizione, come ad esempio la moderna piattaforma Copernicus e il programma Sentinels di ESA. Un sistema di agricoltura di precisione che permetterà alle aziende agricole di raccogliere dati con una precisione (temporale e qualitativa) da permettere un utilizzo delle risorse più oculato e sostenibile, sia dal punto di vista ambientale che economico. Infatti, ogni agricoltore potrà discriminare quali zone della coltivazione che necessitano di un maggiore o minore apporto idrico, di concimi e di prodotti fitosanitari per la difesa delle colture. In tal modo il progetto si propone anche di ottimizzare il processo di irrigazione e di riduzione delle emissioni nocive nell'ambiente, in quanto la distribuzione di fitofarmaci e fertilizzanti secondo criteri sito-specifici limiterà il loro uso.

Obiettivi

Obiettivo del progetto è quindi, attraverso la messa a punto e il collaudo di un sistema di sensori remoti e/o locali, definire un sistema di monitoraggio a differente scala di precisione (dal territorio provinciale alla singola azienda agricola) che permetta l'applicazione di sistemi e tecniche agricole avanzate, con particolare riguardo alla Precision Farming, nelle sue applicazioni nell'ambito della irrigazione, concimazione e difesa fitosanitaria. Ulteriore obiettivo è l'automatizzazione, l'organizzazione e ottimizzazione della raccolta di dati

storici o provenienti da fonti di implementazione diverse (sistemi informativi proprietari e isolati), al fine di fornire informazioni e/o supportare processi decisionali.[13]

1.3 Big Data & Open Data

1.3.1 Big Data

L'uso dell'elettronica per implementare soluzioni IT è sempre stato legato alla trattazione di grandi moli di dati sin dal tardo 20° secolo. Che tali soluzioni fossero utilizzate per applicazioni dedicate all'elaborazione o per creare procedure di business avanzate, il loro scopo è sempre stato quello di gestire grandi moli di dati in maniera molto più veloce, consistente e accurata di come gli esseri umani potessero fare. Il campo di ricerca dei Big Data rappresenta sia un cambio di paradigma che un'evoluzione di come i dati sono e verranno raccolti, analizzati ed elaborati. Allo stesso modo di come la computazione distribuita ha portato nuove prospettive economiche per lo sfruttamento delle soluzioni IT, così ora i Big Data sono una nuova strada per lo sviluppo di modelli di business che porteranno un avanzamento tecnologico sia nell'analisi che nella gestione dei dati, strutturati e non [9][16].

Che cosa significa "big data"

I Big Data sono un fenomeno [9] informatico caratterizzato da una rapida espansione dei dati grezzi, dati che vengono collezionati e generati in maniera rapidissima e che stanno inondando sia i governi che la società.

> *"From the dawn of civilization until 2003, humankind generated five exabytes of data. Now we produce five exabytes every two days... and the pace is accelerating."*
>
> Eric Schmidt, Executive Chairman, Google

Questa nuova prospettiva rappresenta sia una sfida che una opportunità: una sfida legata al volume di dati messo in gioco; un'opportunità per incrementare l'efficienza delle istituzioni attraverso un'adeguata analisi delle informazioni prodotte. Le piattaforme Big Data si distinguono comunemente dalle piattaforme IT convenzionali per 4 requisiti o dimensioni fondamentali: Volume, Velocità, Varietà e Veracità. Le soluzioni per applicazioni Big Data sono caratterizzate da complesse elaborazioni e associazioni di dati in real-time, un'avanzata capacità analitica e capacità di ricerca, enfatizzando in particolare il flusso di dati.

Le 4 dimensioni dei Big Data

Volume Le soluzioni Big Data devono poter gestire ed elaborare grandi quantità di dati, nell'ordine degli Zettabytes e oltre. I sistemi Big Data devono dunque essere **scalabili** per poter soddisfare i requisiti di Volume. Le classiche tecnologie di archiviazione (RAID) non sono però più adatte a soddisfare le prestazioni richieste dai sistemi Big Data, sia per resistenza ai guasti sia per throughput, così come le attuali unità di elaborazione non riescono a stare al passo con la crescita dei dati. Si necessità quindi un cambiamento radicale, su come la componentistica è sviluppata, costruita e resa operativa. A risoluzione di questi problemi possono essere adottate le moderne tecniche di *Database sharding*, basate sul principio che "un DB piccolo è anche più performante" e l'uso dei nuovi sistemi di archiviazione come i Solid State Drives che, malgrado questi degradino molto velocemente, scalano in maniera migliore sia a livello hardware che a livello prestazionale.

Velocità Le soluzioni Big Data devono poter elaborare sempre più velocemente i dati, anch'essi in crescita sempre più rapida. Dal momento che le attuali componenti hardware (throughput di HDD, SSD, RAM e CPU) e la rete con possono competere con la velocità con cui i dati vengono generati è necessario perfezionare i dispositivi in termini di latenze d'accesso e tempi di risposta. Le operazioni di accesso ai dati sono operazioni onerose e richiedono infrastrutture di rete efficienti e con una discreta potenza, specie utilizzando paradigmi classici one-to-one e one-to-many. I tempi di risposta per ottenere i dati elaborati costituiscono un punto critico all'interno dei Big Data, specie quando è necessario garantire velocità e feedback real-time, e richiedono nuovi approcci all'elaborazione e allo sfruttamento delle memorie.

Varietà Le soluzioni Big Data devono essere capaci di analizzare ed elaborare qualsiasi tipo di dato, strutturato, non-strutturato o semi-strutturato. Il fatto che un sistema Big Data debba supportare qualsiasi dato impone una sfida non indifferente alle sue prestazioni, sia per la crescita dei dati che per l'aumentare delle sorgenti.

Veracità Le soluzioni Big Data devono possedere meccanismi di validazione per garantire la correttezza delle grandi moli di dati elaborate che rapidamente giungono. I dati raccolti dalle sorgenti sono spesso proni alla presenza di errori e di conseguenza i dati da essi derivati sono errati: è cruciale dunque poter effettuare una discriminazione dei dati appetibili da quelli errati.

Benefici economici

I Big Data hanno a che fare con un uso più vasto dei dati, con l'integrazione di nuove sorgenti di dati e con un'analisi sempre più profonda attraverso l'uso di strumenti sempre più all'avanguardia al fine di incrementare l'efficienza e rendere disponibili nuovi modelli di business. Oggigiorno i Big Data stanno divenendo un imperativo categorico nell'ambito business poiché consente alle organizzazioni di raggiungere numerosi obiettivi prima d'ora impensabili:

- Applicare analisi statistiche che vanno oltre ai tradizionali dati analitici sui casi d'uso, al fine di supportare decisioni real-time, sempre e ovunque.
- Attingere da qualsiasi tipo di informazione che può essere utilizzata all'interno di un sistema di supporto alle decisioni di tipo data-driven (es.: Google Analytics).
- Permettere a tutte le persone in qualsiasi ambito lavorativo di esplorare ed analizzare le informazioni ed offrire previsioni.
- Ottimizzare qualsiasi tipo di decisione, che esse siano effettuate da sistemi individuali o siano integrate in sistemi automatizzati utilizzando previsioni basati sulle analisi statistiche.
- Produrre previsioni da tutti i punti di vista e orizzonti temporali possibili, da investigazioni storiche all'analitica real-time, fino a sistemi di modellazione delle previsioni.
- Perfezionare i guadagni economici e gestire eventuali rischi, attuali e futuri.

In breve i Big Data garantiscono la capacità ad una organizzazione di rimodellarsi in una impresa contestualizzata, che dinamicamente di adatta ai bisogni in continuo cambiamento dei propri utenti attraverso l'uso di informazioni raccolte da un vasto numero di sorgenti.

1.3.2 Open Data

Che cosa sono gli Open Data

Se da un lato i Big Data forniscono una nuova visione riguardo alla gestione, elaborazione ed analisi dell'immenso oceano di dati che la società produce, dall'altro lato non sempre questi dati sono accessibili a tutti. Si distinguono 2 principali categorie: Open Data e Closed Data. Nei Closed Data rientrano tutti quei dati sulla quale vengono imposti vincoli legali sulla loro diffusione,

utilizzo, modifica etc. Gli Open Data, riprendendo le definizioni fornite dal sito `http://opendefinition.org/`, possono essere così definiti:

> *"Open means anyone can freely access, use, modify, and share for any purpose (subject, at most, to requirements that preserve provenance and openness)."*

Perciò gli "Open Data" sono tutti i dati che:

> *"[...] can be freely used, modified, and shared by anyone for any purpose."*

L'Openess (ovvero la disposizione all'apertura), a livello legale, implica dunque l'applicazione di licenze d'uso che permettano, in maniera legalmente riconosciuta, l'accesso, l'uso e la modica in modo libero, gratuito e per qualunque scopo. In particolare devono essere liberi dai vincoli imposti da copyright e brevetti [2]. Gli aspetti più importanti da tenere in considerazione sono:

- **Disponibilità e accesso**: i dati devono essere disponibili nel loro complesso, per un prezzo non superiore ad un ragionevole costo di riproduzione, preferibilmente mediante scaricamento da Internet. I dati devono essere disponibili in un formato utile e modificabile.

- **Riutilizzo e ridistribuzione**: i dati devono essere forniti a condizioni tali da permetterne il riutilizzo e la ridistribuzione. Ciò comprende la possibilità di combinarli con altre basi di dati.

- *Partecipazione universale*: tutti devono essere in grado di usare, riutilizzare e ridistribuire i dati. Non ci devono essere discriminazioni né di ambito di iniziativa né contro soggetti o gruppi. Ad esempio, la clausola 'non commerciale', che vieta l'uso a fini commerciali o restringe l'utilizzo solo per determinati scopi (es. quello educativo) non è ammessa.

[2] in Italia, e più in generale in Europa, la concessione di un brevetto è limitata: « Possono costituire oggetto di brevetto le nuove invenzioni atte ad avere un'applicazione industriale, quali un metodo o un processo di lavorazione industriale, una macchina, uno strumento, un utensile o un dispositivo meccanico, un prodotto o un risultato industriale e l'applicazione tecnica di un principio scientifico, purché essa dia immediati risultati industriali. [...] » Art. 2585 del Codice Civile della Repubblica Italiana.

Pro e Contro

Il dibattito sugli Open Data è in continua evoluzione. Le applicazioni Open a disposizione dei governi degli stati tentano di emancipare il cittadino, aiutare le piccole attività business o di creare valore economico in modi maggiormente costruttivi. Rendere Open i dati governativi è solo il primo passo verso il miglioramento dell'educazione e del governo stesso, e di costruire soluzioni capaci di risolvere, assieme, i problemi reali che affliggono il mondo.

Malgrado siano state effettuate innumerevoli argomentazioni, molte di queste, che siano contro o a favore, dipendono fortemente dal tipo di dato trattato ed il suo potenziale utilizzo. Alcuni esempi di argomentazioni Pro Open Data sono:

- Trasparenza: [11] I cittadini devono sapere cosa fa il governo e per fare ciò rendere Open tutti i dati governativi è indispensabile.
- Molte opere sono state realizzate grazie all'uso di soldi pubblici: i dati di queste opere dovrebbero essere universalmente disponibili.
- Non dovrebbe essere possibile applicare licenze Copyright sui fatti. Spesso aziende impongo tale limitazione d'uso sulle pubblicazioni scientifiche, impedendone in molti casi il riuso.
- Le aziende sponsor non possono trarre pieno vantaggio dei loro investimenti finché i dati non sono resi pienamente accessibili.
- Le restrizioni sul riuso del materiale generano un anti-commons.
- Nel campo della ricerca scientifica, l'Openess dei dati accelera il processo con cui le scoperte scientifiche vengono fatte.
- Permette il miglioramento della preservazione dei dati a lungo termine.

Altre argomentazioni invece a sfavore dell'Openess, spesso poste da istituzioni private, sono:

- I finanziamenti governativi non dovrebbero essere utilizzati per fare concorrenza al settore privato.
- Il governo deve essere responsabile dell'uso efficiente dei soldi derivanti dalle tasse.
- Il guadagno ricavato dalla pubblicazione Open dei dati permette alle organizzazioni no-profit di istituire nuove attività.

- Per questioni di privacy, l'accesso ad alcuni dati dovrebbe essere limitato solo a figure specifiche.
- La collezione, l'analisi, la gestione e la distribuzione dei dati sono tipicamente attività e/o processi costosi per chiunque renda disponibili tali servizi che dovrebbero essere dunque adeguatamente remunerati.
- Spesso molti utenti necessitano di dati finali propriamente elaborati ed analizzati. Se tutti i dati fossero completamente Open allora sarebbero minori gli incentivi ad investire in tali processi di creazione del dato utile.

Capitolo 2

Tecnologie

2.1 Applicazioni Distribuite

Per applicazione distribuita si intende una componente software messa in esecuzione su molteplici macchine in modo simultaneo all'interno di una rete. Queste applicazioni comunicano in maniera asincrona attraverso meccanismi di scambio di messaggi, coordinandosi in modo tale da portare a compimento l'esecuzione di una task comune.

All'interno delle applicazioni distribuite l'esecuzione deve essere resa trasparente all'esterno, ovvero, devono essere sviluppate in modo tale da nascondere all'utente che l'applicazione non esegue totalmente (o affatto) sulla macchina locale ma è dislocata su più nodi. Tali applicazioni devono essere resistenti ai fallimenti: se uno dei nodi su cui è attiva l'applicazione è o va offline allora l'applicazione è in grado di completare in ogni caso la sua esecuzione.

2.2 Hadoop Distributed File System

2.2.1 Hadoop

Apache Hadoop [2] è un framework che consente l'esecuzione e creazione di applicazioni distribuite che lavorano su grandi moli di dati, situate su HDFS, sfruttando modelli di programmazione Object Oriented. Hadoop è sviluppato per essere elastico, efficiente anche su hardware datato e/o a basso costo, e per adattarsi a configurazioni da singole fino a migliaia macchine, ognuna delle quali fornisce risorse di elaborazione e spazio di archiviazione. Hadoop dispone di una modalità ad Alta Affidabilità implementata in modo tale da rilevare e gestire guasti a livello di applicazione anziché a livello hardware tramite l'uso di uno Zookeeper.

Hadoop (2.x.y) è composto da 4 moduli principali:

- **Hadoop Common** che fornisce gli strumenti base per l'utilizzo del framework e l'accesso al file system;

- **HDFS: Hadoop Distributed File-System**, un file-system distribuito progettato per essere altamente tollerante ai guasti e capace di eseguire perfettamente anche su hardware a basso costo e/o datato. HDFS fornisce accesso rapido ai dati ed è adatto ad applicazioni che lavorano su grandi moli di dati. Il suo comportamento è stato basato sul modello ed implementazione dei file-system POSIX [1] fornita dai file-system Linux.

- **Hadoop YARN: Yet Another Resource Negotiator** che, come dice il nome, è un "Negoziatore di Risorse", nato dalla ri-progettazione del framework di Map-Reduce 1 in modo tale da separare la logica di Resource Management e Job Scheduling in daemoni separati (Resource Manager e Application Master). Questo framework mette a disposizione 2 algoritmi di scheduling: un Fair-Scheduler e un Capacity-Scheduler.

- **Hadoop Map Reduce**, framework che fornisce costrutti di alto livello per la creazione di applicazioni concernenti l'elaborazione di gradi moli di dati.

2.2.2 HDFS

Hadoop Distributed File System (HDFS) è un **file-system distribuito, progettato per essere altamente tollerante ai guasti ed eseguibile su hardware a basso costo e/o non all'avanguardia**. E' sviluppato usando il linguaggio Java perciò qualsiasi piattaforma che supporta tale linguaggio può eseguire i servizi messi a disposizione da HDFS.

Assunzioni e obiettivi di HDFS

- Con l'aumentare delle dimensioni di un sistema i guasti hardware divengono una norma, è quindi fondamentale **individuare e risolvere in modo automatico i punti in cui il sistema è guasto**. HDFS sfrutta una tecnica chiamata Heartbeat per apprendere in real-time lo stato di funzionamento dei singoli nodi del cluster. Quando un certo nodo smette di inviare Heartbeat, il Namenode lo segna come offline e non gli trasmette nuove operazioni di lettura o scrittura finché il nodo non ritorna online (atto indicato da un nuovo Heartbeat). Finché il nodo è offline

[1] Portable Operating System Interface (POSIX), famiglia di standard IEEE utilizzati per mantenere la compatibilità tra sistemi operativi diversi. https://en.wikipedia.org/wiki/POSIX

i dati da lui memorizzati non sono accessibili in alcun modo. Per questo motivo il Namenode, non appena il numero di repliche dei blocchi (vedi Archiviazione) scende sotto la soglia impostata, avvia una operazione di replicazione dei blocchi mancanti.

- Le applicazioni che operano su HFDS necessitano di un **accesso continuo ed illimitato ai dati**. HDFS rimuove alcuni vincoli imposti dai file-systems POSIX in modo tale da migliorare la velocità di accesso e reperimento dei dati piuttosto che diminuire la latenza.

- Le applicazioni che operano su HDFS possiedono dataset di dimensioni variabili da i pochi GigaByte a molti TeraByte. Occorre perciò una piattaforma appositamente configurata per gestire queste enormi moli di dati.

- HDFS mette a disposizione delle applicazioni un modello di accesso ai file di tipo ***write-once-read-many***: un file una volta creato, scritto e chiuso non deve essere modificato. Tali assunzioni semplificano la gestione della coerenza dei file e permettono una migliore velocità di accesso ai dati.

- **"Moving Computation is cheaper than moving Data"**: quando una applicazione richiede un certo insieme di dati, è molto più performante dislocare l'esecuzione dell'applicativo "vicino" a dove sono situati tali dati, specie se applicativo richiede di elaborare una grande quantità di dati. HDFS, in tal senso, fornisce alle applicazioni delle interfacce per muoversi tra i vari nodi del cluster.

- Garantire la portabilità su piattaforme hardware e software eterogenee.

Architettura

HDFS è basato su una **architettura centralizzata di tipo master/slave** (vedi Figura 2.1) ed è composto da 2 principali componenti:

- Un singolo **Namenode** (attivo), ovvero un server centrale che gestisce i namespace del file-system, regola l'accesso ai file per le applicazioni client e le operazioni quali apertura, chiusura e rinominazione di file e cartelle. Gestisce inoltre la coerenza tra file e i relativi blocchi di cui sono composti e attraverso i Blockreport può verificare quali sono i blocchi memorizzati su un certo nodo del cluster. Le proprietà del sistema, il namespace e la coerenza file-blocchi sono tutte mantenute in un file locale al Namenode chiamato FSImage. Il Namenode inoltre sfrutta un

EditLog (situato anch'esso nel suo file-system locale) per memorizzare ogni singolo cambiamento che i meta-dati del file-system subiscono a seguito di operazioni di creazione, eliminazione, scrittura e modifica del fattore di replicazione di un file. Il Namenode può mantenere più repliche di questi file per garantirne la consistenza in caso di corruzione di uno di essi. Questo però aumenta il numero di operazioni di aggiornamento che devono essere effettuate.

- HDFS è composto infine da un insieme di **Datanodes**, solitamente uno per nodo, che gestiscono lo storage nei nodi sui quali sono in esecuzione. Essi sono responsabili della gestione delle richieste di lettura e scrittura dai client del file-system. Inoltre gestiscono creazione, cancellazione e replicazione dei blocchi secondo richiesta del Namenode. Ogni Datanode alloca 1 file per ogni blocco nel proprio file-system e li distribuisce in directories differente basandosi su un algoritmo euristico per determinare in giusto quantitativo di file per ogni directory in modo che l'accesso ai file sia ottimale.

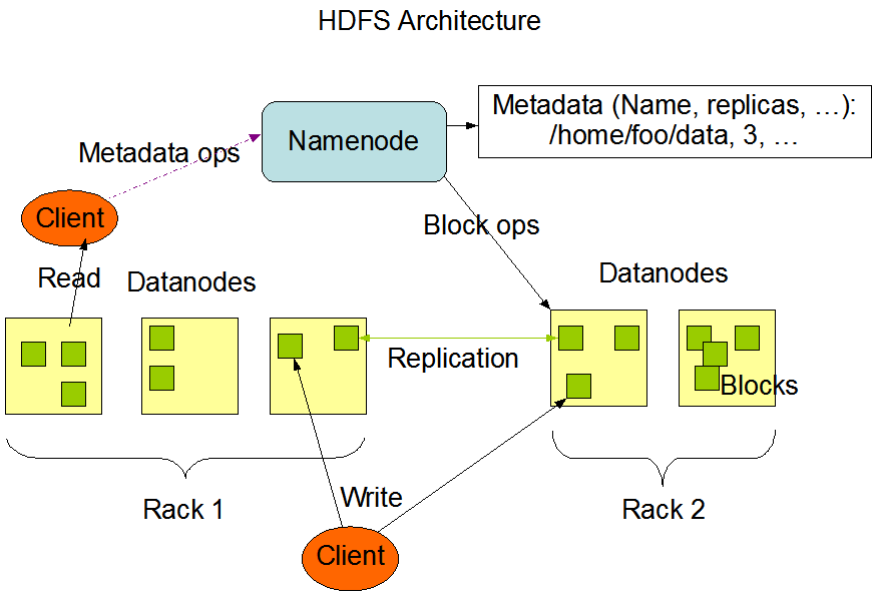

Figura 2.1: Rappresentazione semplificata dell'architettura di HDFS

Archiviazione

HDFS memorizza i dati in file binari, i quali sono suddivisi in una sequenza di uno o più blocchi, tutti della stessa dimensione (eccetto l'ultimo) e che sono

fisicamente memorizzati sui Datanode. La dimensione di default di ogni blocco è di 64MB.

Per ogni file può essere specificato (alla creazione o in fasi successive) il numero di repliche di ogni blocco, altrimenti tale *fattore di replicazione* è deciso dal Namenode ed è descritto nel suo file di configurazione. L'allocazione di tali repliche è cruciale per l'affidabilità, la prontezza e le performance sia del sistema che della rete sottostante (*specialmente in cluster sparsi su più reti, formando perciò più "gruppi" di rete o rack*). HDFS cerca di ottimizzare l'allocazione dei file nel cluster poiché, per soddisfare le richieste di lettura di un certo file, indirizza all'applicazione le repliche del file ad essa più vicine.

Si hanno 2 possibili modi di allocazione:

1. Distribuire equamente le repliche su ogni rack: perciò se R sono il numero di repliche e N sono il numero di gruppi di macchine allora avremo R/N repliche per gruppo. Questo però incrementerebbe il costo delle operazioni di scrittura dei file su HDFS poiché per ogni scrittura è necessario trasferire i blocchi in cui è suddiviso il file anche agli altri gruppi di nodi.

2. Distribuire 2/3 delle repliche sul rack locale al Namenode (attivo) ed il restante 1/3 su un altro gruppo di nodi. Questo approccio riduce drasticamente il costo delle scritture ma ha un impatto significativo sulla rete nel momento in cui occorre eseguire una operazione di lettura sui dati.

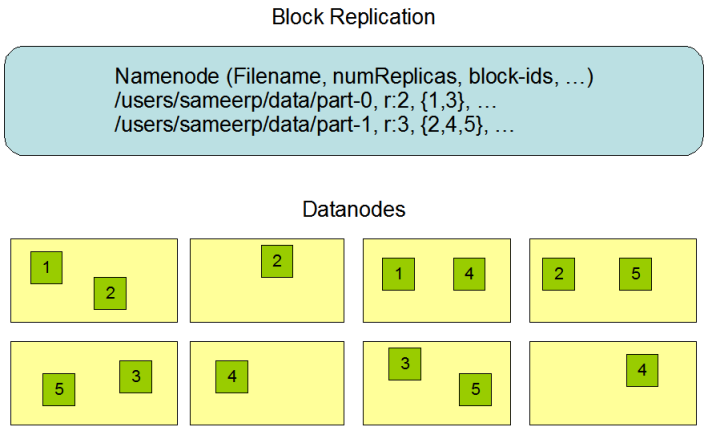

Figura 2.2: Rappresentazione della suddivisione dei blocchi di un file su HDFS

Proprietà A.C.I.D.

Data la sua natura distribuita, HDFS si propone di garantire le seguenti proprietà per le transazioni: **Atomicità**, **Consistenza** (*one-copy-update-semantics*) e fornisce un livello di **Durabilità** *variabile*, ovvero dipendente dalla ridondanza dei blocchi di dati in cui un dato file è stato suddiviso. E' importante però puntualizzare che HDFS **non fornisce alcuna garanzia di Isolamento** delle transazioni, ergo nessun controllo sugli accessi concorrenti da parte delle varie applicazioni allo stesso dato.

2.3 Sentinels Scientific Data Hub

La Scientific Data Hub è una piattaforma che offre accesso completo, libero e gratuito ai dati raccolti dalle missioni Sentinels attraverso un'interfaccia Web interattiva denominata Copernicus oppure attraverso le REST API rese accessibili a questo indirizzo: https://scihub.copernicus.eu/apihub/.

2.3.1 Dati forniti da Sentinel-2

I prodotti Sentinel-2 Multi-Spectral Instrument (MSI) [7][6] consistono di una serie di n granuli ognuno dei quali consiste di 13 immagini (1 per banda spettrale) in formato JPEG-2000, suddivise tra 3 risoluzioni spaziali prestabilite di 10m, 20m e 60m (vedi Figura2.3).

L'area che un singolo un granulo copre varia dal tipo di prodotto:

- Nei prodotti con correzione orto-fotografico/cartografica (ovvero 1C e 2A) ogni granulo, chiamato tile in questo specifico caso, ricopre un'area di $100Km^2$ in proiezione UTM/WGS84 (vedi Figura 2.4). A questi tipi di prodotti sono associati i meta-dati per la geo-referenziazione.

- Nei prodotti di livello 0, 1A e 1B ogni granulo consiste in un immagine che copre un'area di $25Km^2$. Inoltre i prodotti di livello 0 e 1B non sono disponibili al pubblico poiché consistono di immagini grezze e perciò difficilmente utilizzabili.

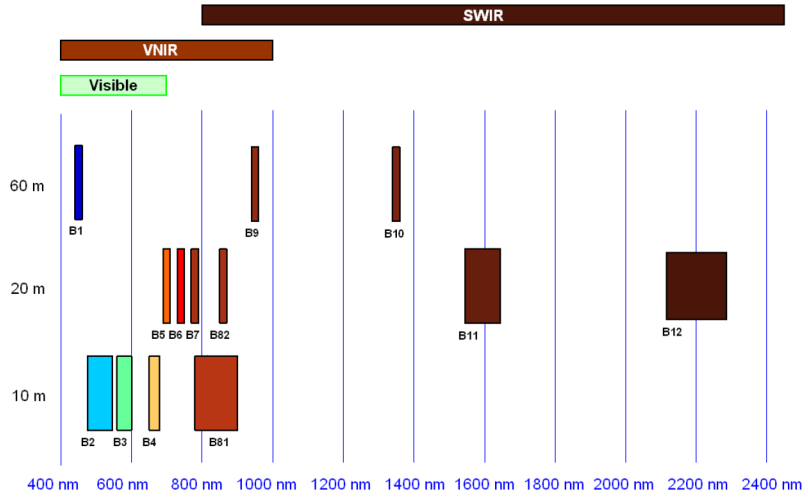

Figura 2.3: Bande spettrali messe a disposizione nei prodotti MSI (VNIR: Visible and Near Infra-Red, SWIR: Short Wave Infra-Red).

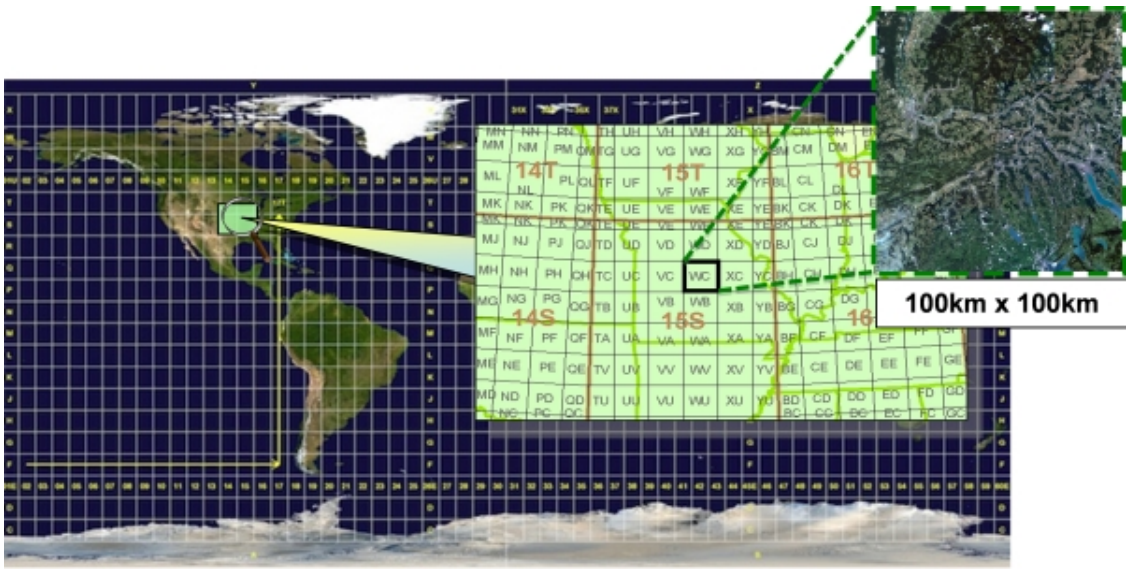

Figura 2.4: Il sistema UTM (Universal Transverse Mercator) divide la superficie terrestre in 60 fusi distinti. Ogni fuso possiede una larghezza verticale di longitudine 6° ed è suddivisa in 20 fasce ciascuna avente una larghezza orizzontale di latitudine 8°, generando così 1200 zone UTM. Le zone sono contrassegnate da un numero che varia da 1 a 60 (indicante il fuso) e una lettera che varia da C a X (indicante la fascia).

Tipi di prodotti a disposizione

- **Prodotti di livello 0**: Questi prodotti consistono in immagini compresse in formato ISP. Al loro interno sono composte da meta-dati descriventi il prodotto, dati per la coerenza dell'immagine compressa e dati necessari ai livelli successivi (per esempio i dati per l'elaborazione del modello geometrico, dati di correlazione, dati termici, dati efemèridi [2] e dati posizionali).

- **Prodotti di livello 1A**: Questi prodotti sono ottenuti effettuando la decompressione delle immagini fornite dai prodotti di livello 0, sulle quali viene poi sviluppato il modello geometrico tale da mappare la posizione di tutti i pixel presente nell'immagine. Le coordinate dei pixel in questi pacchetti sono riferite al loro ipotetico incentro.

- **Prodotti di livello 1B**: Ai prodotti di livello 1A, per essere qualificati al livello 1B, deve essere applicata la correzione radiometrica dell'immagine in termini di correzione del coefficiente di riflessione a Top of Atmosphere (TOA) e della geometria del sensore. La correzione radiometrica consiste in: correzione di segnali scuri, disomogeneità dei pixel, correzione dei pixel difettosi per interpolazione degli stessi con i pixel vicini, ripristino delle bande ad alta risoluzione. Inoltre questo tipo di prodotto contiene i modelli geometrici perfezionati per produrre le immagini di livello 1C.

- **Prodotti di livello 1C**: Questo livello di prodotti sono generati a partire dalle immagini 1B usando modelli di Digital Elevation (DEM) per proiettare l'immagine in un sistema di rifermento cartografico/orto-fotografico. Le misure radiometriche per ogni pixel sono fornite sotto forma di coefficiente di riflessione Top of Atmosphere, assieme ai parametri per trasformare tali valori in radianza. Questi prodotti sono campionati con una distanza di campionamento (Ground Sampling Distance - GSD) costante di 10m, 20m o 60m in base alle bande spettrali a cui sono correlate.

  In questo caso le coordinate dei ogni pixel sono riferite allo spigolo in alto a sinistra del pixel.

  Il prodotti di livello 1C contengono inoltre maschere inerenti a terreno, acqua, nubi, ed ECWF (valori totali di ozono e vapore acqueo, livello medio della pressione atmosferica a livello del mare).

[2] effemèride (o efemèride) s. f. [dal lat. ephemĕris -ĭdis, ...]: 3. Tavola o gruppo di tavole numeriche, dette astronomiche (o anche nautiche, in quanto servono principalmente alle esigenze della navigazione), che forniscono le coordinate degli astri (o altri dati astronomici variabili col tempo) a intervalli prefissati ed uguali fra loro, per es. di giorno in giorno oppure di ora in ora. Per estens., anche i libri, generalmente pubblicati con frequenza annuale, che contengono tali raccolte. [5]

- **Prodotti di livello 2A**: I prodotti di livello 2A mettono a disposizione immagini con coefficiente di riflessione corretto a Bottom of Atmosphere e con correzione delle nubi e vapore acqueo tramite l'uso delle maschere messe a disposizione dai prodotti di livello 1C da cui sono derivati.

Si vuole ora presentare più nello specifico la sequenza di elaborazione dei prodotti 1B in prodotti di livello 1C poiché questi ultimi sono i pacchetti di dati che vengono usati come base all'interno del processo ETL.

1. **Ricampionamento**:
   (a) Selezione delle tile che intersecano l'area a cui l'immagine si riferisce (l'area é specificata nei meta-dati tramite una lista punti geografici, identificati da latitudine e longitudine, della quale identificano i bordi).
   (b) Codifica geografica delle tile tramite il codice di proiezione associato.
   (c) Correzione della geometria dell'immagine (vedi Figura 2.5). In questa fase si cerca di correlare i punti dell'immagine di partenza (1B) con i rispettivi pixel dell'immagine di output (1C) generando cosi l'immagine geometricamente corretta.
   (d) Si genera una griglia di campionamento per ogni banda spettrale che presenta l'immagine. La griglia è calcolata trasformando inizialmente le coordinate native dell'immagine in quelle fornite dal DEM, segue poi l'interpolazione di ogni DEM generato per ottenere l'altitudine. Si procede infine con la geolocalizzazione dei punti dell'immagine.
   (e) Interpolazione dei dati radiometrici di ogni punto per calcolare il valore della radianza.
   (f) Correzione del coefficiente di riflessione Top of Atmosphere per i punti dell'immagine nel nuovo sistema geometrico. Questo passo è effettuato convertendo ogni banda spettrale (k) e il valore numerico (CN) di ogni pixel nel coefficiente di riflessione TOA. Nella misurazione si tiene conto dello spettro della luce esterna (E_s), dell'angolo di incisione della luce rispetto allo Zenith (Θ_S) e il fattore di calibrazione degli strumenti MSI (A_k). La funzione di conversione risulta essere quindi:

$$\rho_k(i,j) = \frac{\pi \cdot CN_{k,NTDI}(i,j)}{A_{k,NTDI} \cdot E_S \cdot d(t) \cdot \cos(\Theta_S(i,j))}$$

2. **Generazione delle Maschere** per nuvole dense, cirri e vapore acqueo.

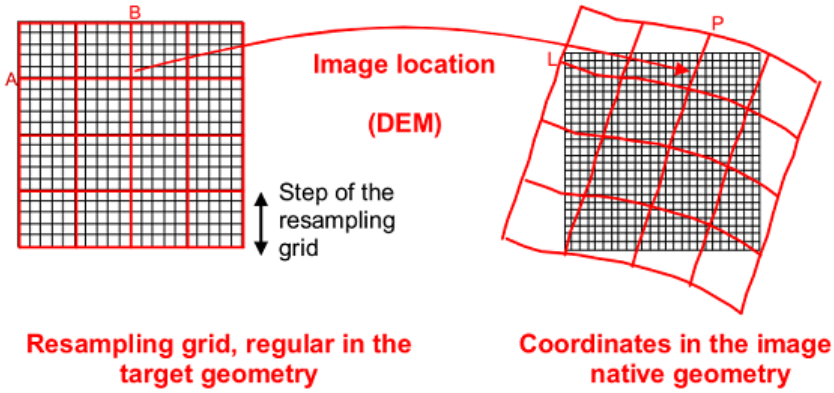

Figura 2.5: Passaggio di sistema geometrico e griglia di campionamento.

Nomenclatura Standard

I pacchetti di dati messi a disposizione dalle missioni Sentinel sono individuati da una nomenclatura standard che li identifica in modo univoco:

MMM_CCCCC_FFF_DDDDDD_SSSS_yyyymmddThhmmss
_ROOO_VyyyymmddThhmmss_YYYYMMDDTHHMMSS.SAFE

1. **MMM** è l'identificatore della missione spaziale indicante da quale satellite è stato prodotto quel pacchetto (Es.: S2A = Sentinel-2A).

2. **CCCCC** indica la classe a cui appartengono i dati (Es.: OPER = Operations Phase).

3. **FFF_DDDDDD** è l'identificatore del tipo di pacchetto, in cui:

   - **FFF** indica la categoria dei pacchetti (Es. di categorie: PRD = Product, MTD = Metadata,).
   - **DDDDDD** identifica la semantica di descrizione (Es.: MSIL2A = Multi Spectral Instrument Level-2A)

4. **SSSS** è l'identificatore del il sito di raccolta: "MTI_" per Matera CGS, "SGS_" per Svalbard CGS, "MPS_" per Maspalomas CGS, "EPA_" per Madrid PAC, "MPC_" per il centro MPC/CC, "UPA_" per il Regno Unito, "EDRS" per gli European Data Relay Satellites. Gli identificatori terminanti con "L" indicano le stazioni terrestri locali. Infine "USER" viene utilizzato per i prodotti Sentinel di livello 2.

5. **yyyymmddThhmmss** è la data di creazione del pacchetto.

6. **ROOO** indica il numero di orbita relativa del satellite.

7. **VyyyymmddThhmmss_YYYYMMDDTHHMMSS** sono rispettivamente le date di inizio e fine della cattura del frammento d'immagine contenuta nel pacchetto.

8. **SAFE**, infine, identifica il formato del pacchetto ed è uguale per tutti i prodotti dei satelliti Sentinels. Questo particolare formato è stato scelto come standard all'interno delle strutture dell'ESA che si occupano di osservazione terrestre. Tale formato consiste di un insieme di cartelle contenenti immagini in formato binario e alcuni file di meta-dati in formato XML. Questa particolare struttura lo rende abbastanza flessibile da poter rappresentare tutti i livelli in cui i prodotti di Sentinel-2 sono suddivisi.

Gli elementi contenuti nei prodotti presentano una nomenclatura leggermente differente in alcune componenti:

MMM_CCCCC_FFF_YYY_ZZ_SSSS_yyyymmddThhmmss_
(SYYYYMMDDTHHMMSS o Afffff)_(Dxx o Txxxxx)_Nxx.yy

1. **YYY** indica il livello del prodotto di cui il dato fa parte.

2. **ZZ** identifica la tipologia del dato ("GR" = Granulo, "DS" = Datastrip, "TL" = Tile, "TC" = True Colour Image, "CO" = Immagine Consolidata).

3. **SYYYYMMDDTHHMMSS** è la data di Inizio di cattura dell'inizio della cattura.

4. **Afffff** è il numero dell'orbita assoluta.

5. **Dxx** è l'identificatore del rilevatore e varia tra 01 e 12.

6. **Txxxxx** indica il numero della tile nella proiezione UTM-WGS84 (Es.: T32TPQ indica la Tile (T) PQ - Bologna - nella zona UTM di fuso 32 e fascia T).

7. **Nxx.yy** è il numero della corrente baseline di elaborazione ed in cui i valori x e y sono cifre decimali.

Formattazione dei prodotti

I pacchetti SAFE resi disponibili da Sentinel-2 possiedono una formattazione interna (vedi Figura: 2.6) composta da una serie cartelle all'interno delle quali le informazioni vengono catalogate per tipologia:

- Un **manifest.safe** file scritto con sintassi XML che contiene i dati generali del prodotto.
- Un'**anteprima dell' immagine**.
- Una cartella contenente i dataset delle rilevazioni e i dati delle immagini (maschere) in formato GML-JPEG2000 (GRANULE).
- Un'altra cartella contenente le informazioni delle datastrip per i vari livelli (DATASTRIP).
- Una ulteriore cartella contenente dati ausiliari (AUX_DATA).
- Infine un file in formato HTML di anteprima.

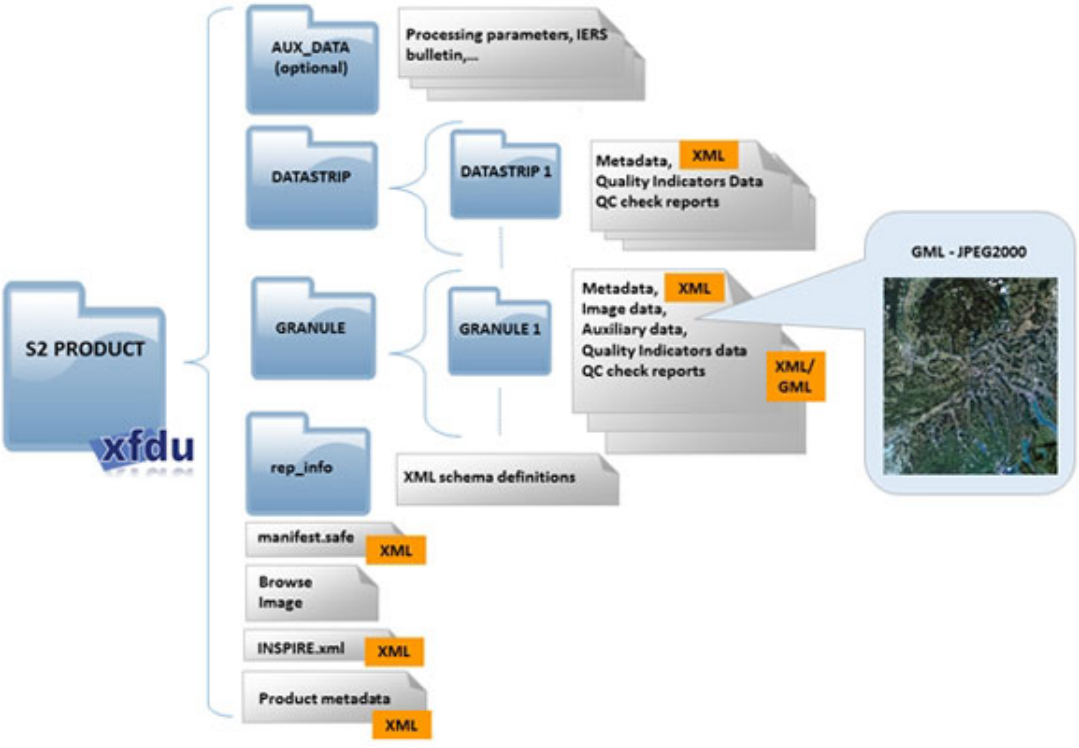

Figura 2.6: Schema della formattazione (superficiale) interna di un pacchetto SAFE.

2.3.2 Scientific Data Hub REST API

Lo Scientific Data Hub espone 2 possibili API per la ricerca e 1 per download dei prodotti:

- **Open Data Protocol**: si tratta di una interfaccia di accesso ai dati sviluppata su protocolli come HTTP e su metodologie ampiamente riconosciute, come REST, le quali possono essere facilmente maneggiate da una vasta gamma di client (Browser web o programmi come cUrl e wget). Il protocollo Open Data permette la creazione di servizi di accesso ai dati basati su metodi REST, i quali permettono la pubblicazione e l'uso delle risorse, identificate tramite URI e definite attraverso un modello dati predefinito, da parte dei client Web, utilizzando dei messaggi in semantica HTTP.

  La Scientific Data Hub, tramite OData, consente sia di ricercare che scaricare i prodotti tramite la costruzione di query personalizzate, dando la possibilità di utilizzare questi servizi anche da remoto.

  L'URI attraverso il quale sono rese disponibili le informazioni dei prodotti è $https://scihub.copernicus.eu/dhus/odata/v1/Products$. Mentre a questo URI $https://scihub.copernicus.eu/dhus/odata/v1/\$metadata$ è reso disponibile il meta-modello rappresentante tutte le proprietà che i pacchetti di dati (definiti entity) reperibili espongono. Il protocollo OData definito sull'hub permette anche di specificare il formato in cui si desidera ricevere la risposta alla richiesta dei pacchetti. I formati disponibili sono Atom (XML - RFC4287) e JSON.

  L'interfaccia di OData consente di scaricare un prodotto attraverso il seguente URI: $https://scihub.copernicus.eu/dhus/odata/v1/Products('uuid')/\$value$, dove 'uuid' è una stringa alfanumerica del tipo xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx identificativa del prodotto (non si tratta del nome del prodotto precedentemente discusso, che invece è identificato da 'identifier').

- **Open Search**: OpenSearch offre un insieme di tecnologie che permettono di pubblicare risultati di ricerca in un formato standard ed accessibile (XML e JSON). Anche OpenSearch come OData sfrutta metodologie RESTful per garantire i servizi. Infatti il framework stesso risulta essere complementare a OData e può essere dunque utilizzato per l'esecuzione delle query (di ricerca) al posto di OData. La sintassi di OSearch risulta maggiormente user-friendly, mettendo a disposizione costrutti in stile SQL. Inoltre arricchisce i messaggi di risposta con delle informazioni supplementari, come ad esempio il numero di elementi totali trovati

con la query. L'implementazione di OpenSearch utilizzata dal Data Hub sfrutta l'engine Apache Solr.

2.4 Sentinel-2 Corrector

Il Sentinel-2 Corrector [1], in breve Sen2Cor, è un tool per la generazione e formattazione dei prodotti di livello 2A per Sentinel-2. Esso esegue la conversione dei dati geo-referenziati Top of Atmosphere (TOA) di livello 1C in dati orto-rettificati Bottom of Atmosphere (BOA) di livello 2A, ai quali vengono applicate correzione atmosferica, del terreno e delle nubi. Possono inoltre essere applicati algoritmi per correggere il coefficiente di riflessione, la profondità ottica dei particolati, il vapore acqueo e/o generare una mappa di classificazione della scena assieme a dati indicanti la qualità complessiva dell'immagine.

Architettura

Il tool di Sen2Cor si suddivide nei seguenti componenti:

1. **L2A_Process**: Questo modulo coordina le interazioni tra i vari moduli di Sen2Cor e genera la struttura dei meta-dati per il prodotto di livello 2A.

2. **L2A_Schedule**: Modulo che si occupa dello scheduling e coordinazione delle esecuzioni parallele dei varie unità di elaborazione.

3. **L2A_ProcessTile**: Questa componente rappresenta una singola unità di elaborazione. Ad essa viene affidata l'esecuzione delle task di classificazione della scena, correzione atmosferica e creazione dei meta-dati sulla base della tile che si sta elaborando.

4. **L2A_SceneClass**: Questo modulo si occupa di eseguire una prima classificazione ed analisi statistica delle immagini in base al loro contenuto: nuvole, neve, acqua, terreno, etc...

5. **L2A_AtmCorr**: Tale componente che esegue la trasformazione dell'immagine da TOA a BOA e applica la correzione atmosferica al prodotto di input.

6. **L2A_Config**: Modulo che gestisce i parametri di configurazione.

7. **L2A_Tables**: Modulo di supporto che gestisce la conversione dei file da JPEG-2000 al formato interno di Sen2Cor e vice versa, inoltre garantisce accesso efficiente ai dati da parte dei moduli di elaborazione.

8. **L2A_Manifest**: A questo componente è affidata la generazione del manifest.safe del nuovo livello del prodotto.

9. **L2A_XmlParser**: Un modulo che mette a disposizione utility per effettuare il parsing dei meta-dati e dei file di configurazione.

10. **L2A_Library**: Una collezione di strumenti d'uso comune tra i vari moduli.

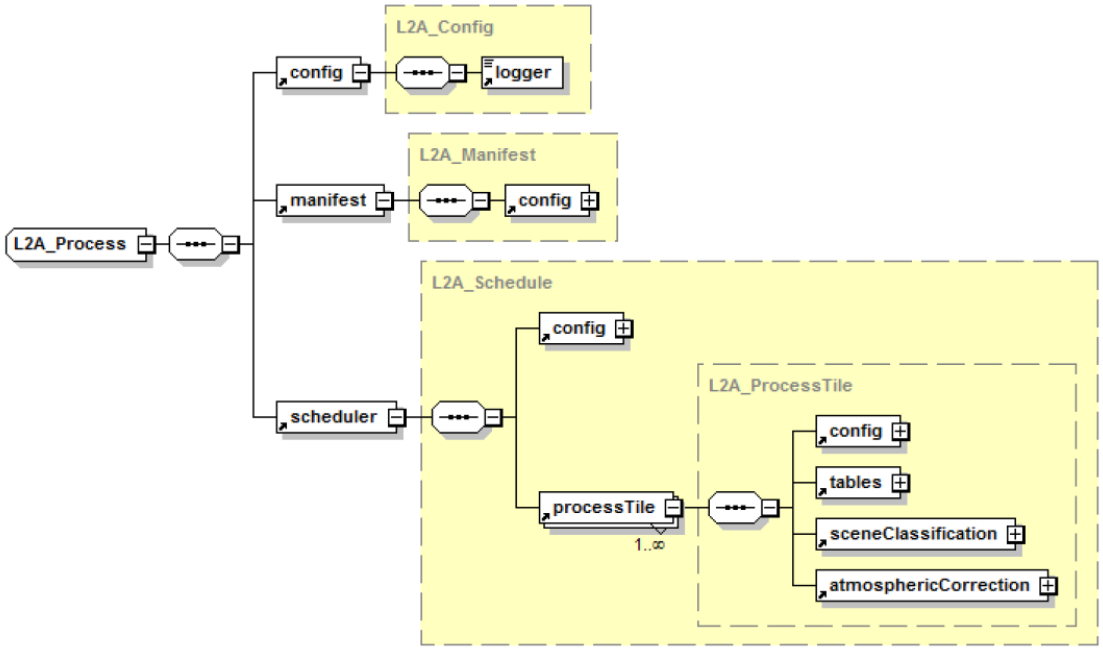

Figura 2.7: Rappresentazione ad alto livello dell'architettura del tool Sen2Cor

Workflow

Per ogni livello di risoluzione spaziale e per ogni tile presente nel prodotto Sentinel-2 di livello 1C vengono eseguite le seguenti trasformazioni (vedi anche Figura: 2.10):

1. **Classificazione della Scena**: L'algoritmo di classificazione della scena permette di rilevare la presenza di nuvole, neve e ombre generate dalla presenza di nubi, generando una mappa di classificazione (vedi Figura 2.8). Una mappa di classificazione consiste di 4 diverse classi di nuvole e 6 differenti classificazioni di ombre (di nuvole, della vegetazione, terreno, deserto, acqua e neve). L'algoritmo sfrutta le proprietà riflessive degli elementi della scena per stabilire la presenza, o l'assenza, delle componenti di disturbo.

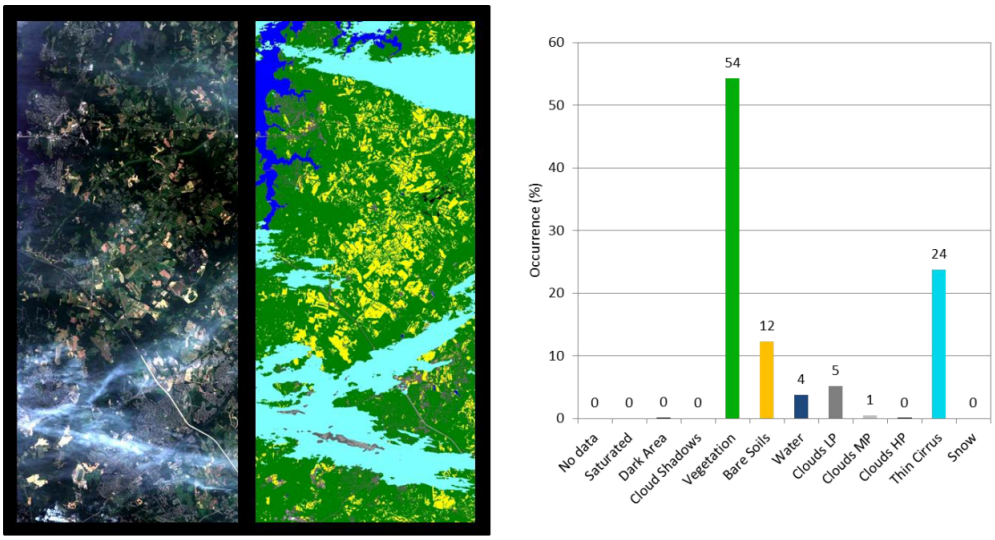

Figura 2.8: L'immagine a sinistra è la ripresa satellitare mentre a fianco è rappresentata la mappa di classificazione della scena. Il grafico a lato mostra le percentuali in pixel degli elementi che comprono la mappa

2. **Correzione Atmosferica**: Il processo di correzione atmosferica è composto da 4 diverse subtask: identificazione della profondità ottica dei particolati (Aerosol Optical Thickness: AOT), recupero del livello di vapore acqueo, recupero del coefficiente di riflessione superficiale del terreno e correzione dei Cirri. Il procedimento ci correzione produce delle mappe che descrivono la profondità ottica dei particolati e il livello di vapore acqueo a livello di pixel e l'immagine Bottom of Atmosphere (BOA) corretta per tutte le bande richieste (le bande sono dipendenti dalla risoluzione spaziale).

   (a) **Identificazione della profondità ottica dei particolati**: Questo procedimento è utilizzato per rilevare il livello di trasparenza, ovvero la qualità visiva, dell'atmosfera. Questo dato è calcolato a partire dall'algoritmo Dense Dark Vegetation (DDV) [3], usando la banda 12 dell'infrarosso (a lunghezza d'onda corta) e correlando il relativo coefficiente di riflessione con le bande del Rosso (4) e del Blu (2). Questo particolare procedimento richiede però che la scena contenga almeno un'area di riferimento il cui comportamento riflessivo sia conoscibile a priori. Per la scelta dell'area si prediligono zone ad

[3]DDV Algorithm - www.mdpi.com/2072-4292/7/3/2668/pdf

alta vegetazione, terreni di gamma bassa (scuri) o bacini d'acqua, ovvero zone che contengano un basso coefficiente di riflessione. In caso queste aree non siano rilevate, l'algoritmo elabora la banda 12 dell'infrarosso in modo tale che includa il livello di luminosità medio dei pixel nel campione in analisi.

(b) **Recupero del livello di vapore acqueo**: Questa task adopera l'algoritmo di Atmospheric Pre-corrected Differential Absorption (APDA) [4] sulle bande 8_a e 9 messe a disposizione da S2-MSI. La banda 8_a rappresenta il canale di riferimento in una regione dell'atmosfera. La banda 9 rappresenta invece il canale su cui è misurato il grado di Adsorbimento [5] della regione. Il livello d'Adsorbimento del vapore è valutato inizialmente calcolando la radianza spettrale dell'atmosfera in assenza di vapore acqueo e assumendo che il coefficiente di riflessione per il canale di riferimento sia lo stesso di quello del canale di misurazione. Successivamente il livello è calcolato misurando il contenuto della colonna del vapore acqueo per pixel contenuta nella mappa generata.

(c) **Correzione dei Cirri**: Questo algoritmo sfrutta la banda 10 dello spettro infrarosso messa a disposizione da Sentinel-2 (poiché i corpi nuvolosi influenzano maggiormente le bande della luce visibile vicine all'infrarosso). Dal momento che sono parzialmente trasparenti, esse risultano problematiche da individuare per gli strumenti multi-spettrali a banda larga, specialmente se la superficie in analisi è particolarmente disomogenea. Mentre il vapore acqueo è presente massivamente nella bassa troposfera (0-5 Km di altitudine), una banda spettrale ridotta all'interno di una regione di spettro in cui l'adsorbimento di vapore acqueo è molto elevato (B10), assorbirà di conseguenza anche la luce riflessa dal terreno ma disperderà il segnale della componente nuvolosa. Ciò ci permette di correlare il coefficiente di riflessione nella banda 10 dell'infrarosso con le bande spettrali del visibile, del quasi-infrarosso e dell'infrarosso a lunghezza d'onda corta (B12) (tali valori sono individuabili nella Figura 2.3 come VNIR - Visible and Near Infra-Red - e SWIR - Short Wave Infra-Red -). In questo modo è possibile ottenere la correzione dell'entità nuvolose semplicemente rimuovendo il loro contributo informativo dallo spettro luminoso emesso dalla scena.

(d) **Recupero del coefficiente di riflessione superficiale**

[4] APDA - http://www.sciencedirect.com/science/article/pii/S0034425798000443

[5] Adsorbimento: Fenomeno in virtù del quale la superficie di una sostanza solida, detta adsorbente, fissa molecole provenienti da una fase gassosa o liquida con cui è a contatto...[5]

3. **Formattazione prodotto**: Il prodotto finale di livello 2A generato da Sen2Cor consiste, similmente ai prodotti di livello 1C, di 13 immagini (una per banda spettrale, vedi Figura: 2.3), in formato JPEG-2000, suddivise nelle 3 risoluzioni spaziali supportate (10m/20m/60m). Inoltre a ciascuna immagine sono associati i meta-dati relativi alla geo-referenziazione. Ogni livello delle tile (area di $100Km^2$), in relazione alla risoluzione spaziale, contiene le seguenti informazioni:

   - Un prodotto di livello 2A a 10 m di risoluzione che contiene le bande spettrali 2, 3, 4 e 8 e una mappa di identificazione della profondità dei particolati alla risoluzione di 20m.
   - Un prodotto con risoluzione a 20 m contenente le bande spettrali 2, 7, 8a, 11 e 12. Inoltre include anch'esso una mappa di identificazione della profondità dei particolati e in più una mappa dei livelli di vapore acqueo.
   - Infine il prodotto a 60m di risoluzione spaziale che include al suo interno le bande 1 e 9, oltre che a tutti gli elementi del prodotto a 20m (adattati alla risoluzione a 60m).

   Le immagini JPEG-2000 del prodotto finale vengono rigenerate attraverso l'uso delle L2A_Tables precedentemente descritte in combinazione alla libreria OpenJPEG ritrasformando il contenuto delle tabelle HDF5 in JPEG-2000.

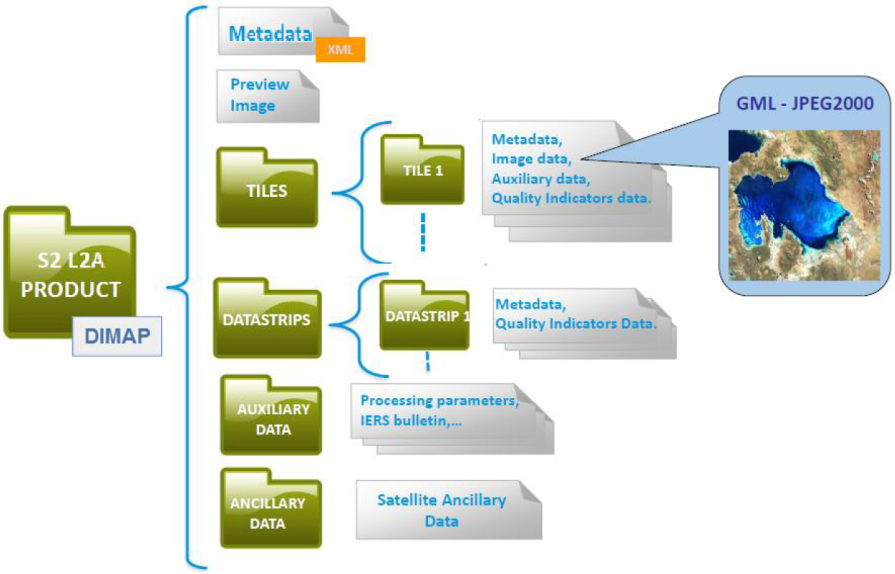

Figura 2.9: Esemplificazione della composizione interna di un prodotto L2A.

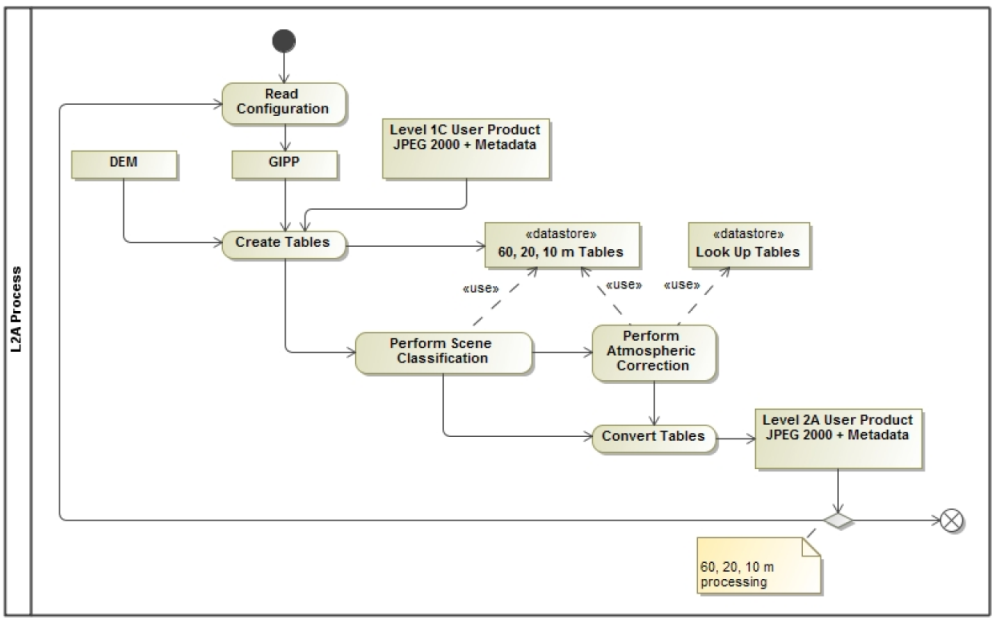

Figura 2.10: Flusso esemplificativo delle operazioni di elaborazione effettuate da Sen2Cor per la trasformazione di prodotti di livello 1C in livello 2A.

2.5 Geospatial Data Abstraction Library

La suite GDAL/OGR [8] è un compendio di librerie multi-piattaforma per la conversione di dati geo-spaziali in formato raster e vettoriale, sviluppate in linguaggio C++ e rilasciate sotto la licenza Open Source X/MIT dalla Open Source Geospatial Foundation.

Come libreria, essa presenta un singolo modello di dati astratto per tutti i formati supportati ed è accompagnata da una pletora di strumenti utilizzabili a linea di comando per la conversione ed elaborazione dei dati.

Garantisce accesso a dati compatibile con varie piattaforme, tra le quali MapServer, GRASS, QGIS e OpenEV. Inoltre è utilizzato dai seguenti software/framework OSSIM, Cadcorp SIS, FME, Google Earth, VTP, Thuban, ILWIS, MapGuide e ArcGIS.

Caratteristiche

- Compatibilità con Python, Java, C#, Ruby, Visual Basic e Perl.
- Motore del sistema di coordinate sviluppato su PROJ.4 e OGC.
- Funzioni di conversione, trasformazione, suddivisione.
- Accesso altamente efficiente ai dati raster attraverso l'utilizzo di tile e viste.
- Supporto a file di dimensioni superiori a 4GB.

La Geospatial Data Abstraction Library (v.2.1.x) supporta 148 formati raster e 84 formati vettoriali. Le principali caratteristiche messe a disposizione sono per i formati raster sono: supporto a file di grandi dimensioni, metodi per effettuare trasformazioni affini, tecniche di caching dei pixel, un motore ottimizzato per le proiezioni e numerosi algoritmi di rasterizzazione, vettorializzazione, interpolazione di pixel e di filtraggio. Per i formati vettoriali sono disponibili: modelli geometrici basati su OGC/ISO, librerie GEOS per le operazioni geometriche, un motore per le proiezioni e supporto del linguaggio SQL.

Modello Dati

Il modello di dataset reso disponibile da GDAL è basato sulle specifiche OpenGIS Grid Coverages. Tale modello è composto dall'unione delle bande raster relative al formato e alcune informazioni comuni a tutti i formati. Tutti i dataset modellano al loro interno il concetto della dimensione raster, espressa in pixel e numero di linee, che è poi applicato a tutte le bande.

Il dataset è inoltre responsabile per le trasformazioni geo-referenziate e per la definizione del sistema di coordinate di tutte le bande raster. Al dataset possono essere associati dei meta-dati sotto forma di una lista di stringhe contenenti coppie chiave/valore.

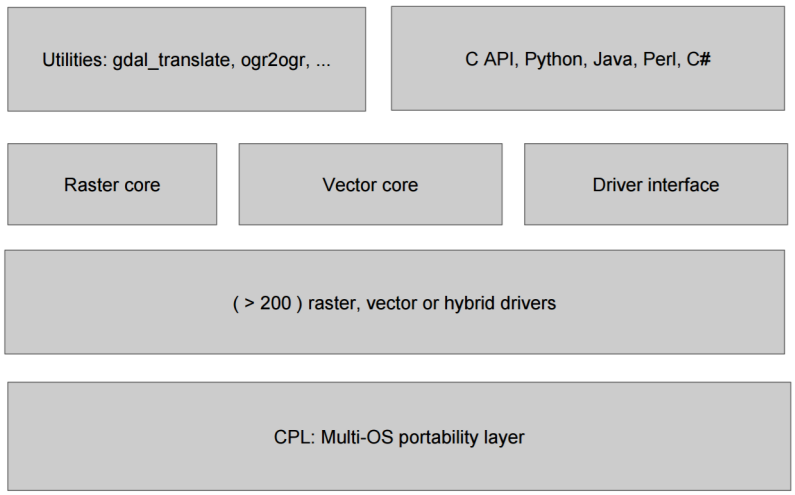

Figura 2.11: Schema che riassume lo stack architetturale di cui è composta la libreria di GDAL

Command Line Tools

All'interno del progetto sono state sfruttate in particolare 2 funzioni a riga di comando messe a disposizione da GDAL:

- **gdal_translate**: è una funzione che permette la conversione da un formato raster all'altro cercando di mantenere intatta la componente informativa espressa da dato. Alla versione attuale di GDAL (2.1.x) questo comando supporta la conversione dei formati SAFE dei satelliti SENTINEL ed il formato JPEG2000 (codifica usata dalle immagini prese dai Sentinel) tramite l'uso della libreria OpenJPEG.

- **gdal_retile**: è uno strumento utilizzato per effettuare il "tiling" del file in input (se è di un formato supportato). L'output è un insieme di nuovi file, nello stesso formato del file di input, ordinati per riga e colonna (pensiamo al file di input come ad una matrice di dimensioni $height/tileHeight \cdot width/tileWidth$ arrotondate per difetto) a ciascuno dei quali è associato un file di meta-dati per mantenere la coerenza e informazioni di supporto. Questo comando consente anche di generare

una piramide di zoom di cui possiamo specificare i livelli. Il numero ottimale di livelli di una immagine è calcolabile tramite l'uso della seguente funzione:

$$numberOfPyramids = log(imageSizeInPixel)/log(2) - log(specifiedTileSizeInPixel)/log(2)$$

2.6 PostGIS

PostGIS [14][15] è un'estensione gratuita e open source al database relazionale PostgreSQL che fornisce supporto per gli elementi geografici e spaziali, in modo che possano essere manipolati e memorizzati come qualsiasi altro tipo di dato. Questa estensione segue le specifiche Simple Feature per SQL messe a disposizione dal Open Geospatial Consortium. PostGIS è basato su modelli geometrici "leggeri" e di indicizzazione ottimizzati per ridurre l'impatto su disco e memoria.

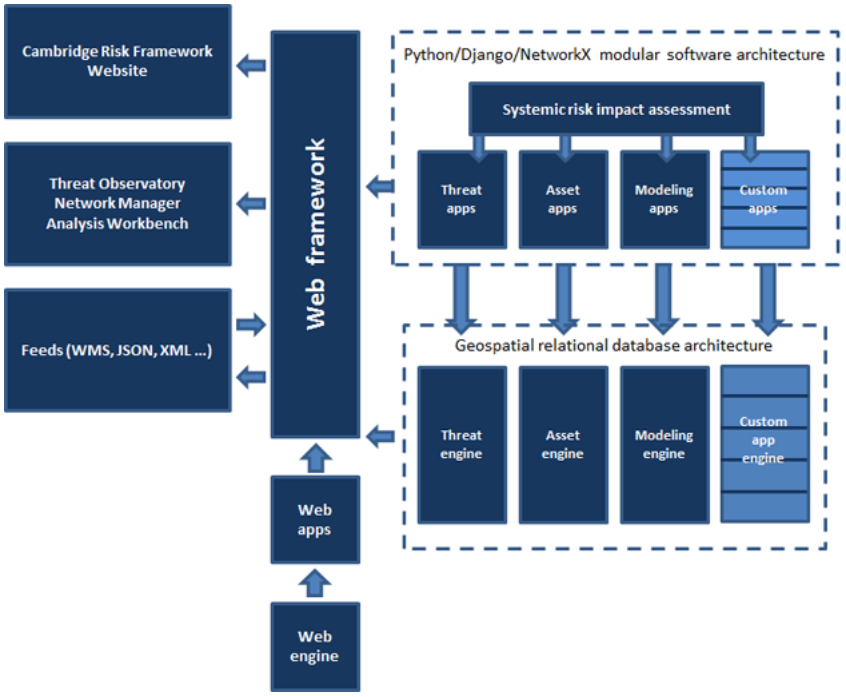

Figura 2.12: Gerarchia dei tipi dato a disposizione di PostGIS.

PostgreSQL

Introduzione: PostgreSQL è un potente Object-Relational Database Management System (O-RDBMS), rilasciato sotto licenza BSD e di conseguenza

gratuito ed open source. Come molti altri programmi open source. PostgreSQL non è controllato da una sola azienda ma è gestito da una comunità globale di sviluppatori e aziende. E' stato principalmente pensato per essere estensibile, lasciando libera la possibilità di creare nuovi tipi di dato, funzioni e modalità di accesso ai dati.

Perchè PostgreSQL? PostgreSQL garantisce le proprietà ACID per le transazioni, supporta pienamente gli standard SQL92, fornisce la possibilità di utilizzare estensioni e plugin. Come già detto, offre un modello di sviluppo orientato alla community. Inoltre, a livello strutturale, non ha limiti imposti sulla dimensione delle colonne, rendendogli possibile il supporto agli oggetti GIS e agli indici generici (GiST).

Tipi di dato e sottotipi

Il DBMS PostGIS dispone di 4 tipi di dato fondamentali: Geometrie, Topologie, Geografie e Raster. PostGIS integra la libreria WKT Raster per il supporto alla gestione dei dati in formato raster.

PostGIS dispone di sottotipi di dato modellanti entità quali:

- Point, identificati da un'ascissa ed un'ordinata.
- LineString, ovvero un oggetto che modella un segmento identificato da 2 punti. Vengono utilizzati per per interpretare un percorso tra 2 coordinate.
- LinearRing, una LineString identificata da 3 o più punti, in cui il primo e l'ultimo punto coincidono.
- Polygon, oggetto che modella un poligono ed è identificato da 2 o più LineString.
- MultiPoint, oggetto che modella un insieme geometrico di punti.
- MultiLineString, oggetto che modella un insieme geometrico di linee.
- MultipPolygon, oggetto che modella un insieme geometrico di poligoni
- GeometryCollection, oggetto che modella un insieme di elementi geometrici misti.

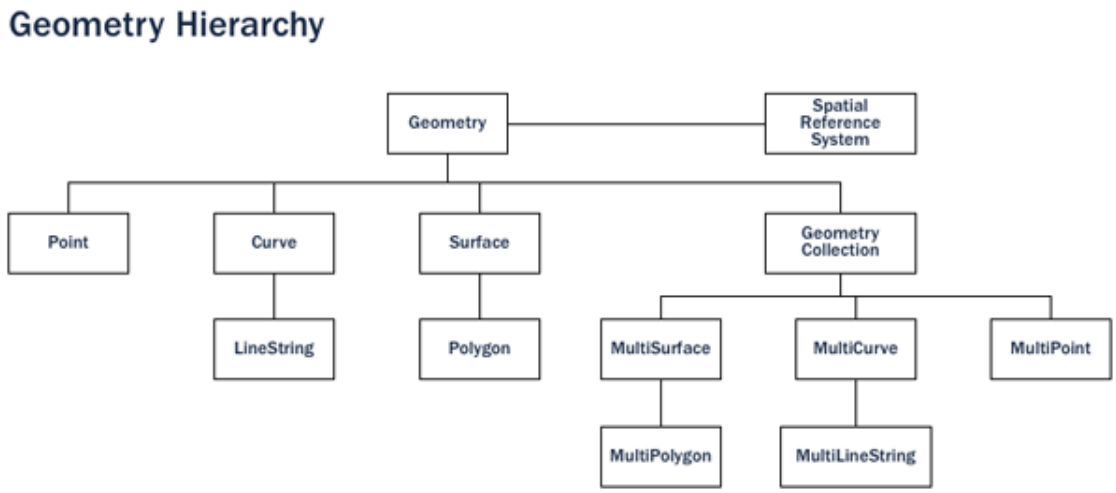

Figura 2.13: Gerarchia dei tipi dato a disposizione di PostGIS.

Indici

PostGIS oltre agli indici classici (B+, B*) ammette anche i seguenti tipi di indici:

- Indici R-tree-over-GiST [6] per ottimizzare l'esecuzione delle query con contenuti solo geo-spaziali.
- Indici Quad-tree [7], per l'indicizzazione di una superficie bidimensionale.
- Indici Grid [8] per l'indicizzazione di superfici tridimensionali.

Tutti gli indici utilizzati da PostGIS sfruttano la tecnica di Bounding Box: il più piccolo poligono regolare (solitamente rettangoli o cerchi) all'interno della quale è contenuto contenuto il mio oggetto geometrico. PostGIS inoltre supporta la selettività degli indici per garantire performance ottimali per l'esecuzione di query miste.

Predicati, Operatori e Funzioni

- Offre predicati specificatamente orientati alla geometria spaziale, per determinare le interazioni (quali Intersection, Boundary, Buffer, Equals e Contains) tra modelli geometrici attraverso l'uso dell'algoritmo 3x3 DE-9IM [9] fornito dalla libreria GEOS.

[6] http://web.eecs.umich.edu/ michjc/eecs584/notes/lecture04-rtree-gist.pdf
[7] https://en.wikipedia.org/wiki/Quadtree
[8] https://en.wikipedia.org/wiki/Grid_(spatial_index)
[9] http://docs.geotools.org/stable/userguide/library/jts/dim9.html

- Operatori orientati orientati alla geometria spaziale per determinare misure geo-spaziali quali area, distanza, lunghezza e perimetro.
- Operatori Unione, Differenza, Simmetria, etc. orientati alla geometria spaziale

La maggior parte delle funzioni geo-spaziali può essere raggruppata in una delle seguenti categorie:

- **Conversione**: funzioni per il passaggio da dati geometrici a formati esterni a PostGIS.
- **Gestione**: funzioni utilizzati per la gestione delle informazioni relativi alle mappe dello spazio geometrico e di amministrazione di PostGIS.
- **Recupero**: funzioni che permettono di ricavare le proprietà e le misure da un dato Geometrico.
- **Comparazione**: funzioni per la confrontare 2 geometrie, rispettando le loro relazioni nello spazio geometrico.
- **Generazione**: funzioni utilizzati per generare nuove geometrie a partire dalle geometrie primitive.

Capitolo 3

Il prototipo

3.1 Descrizione del progetto

Questo applicativo rientra all'interno del progetto MoReFarming, un progetto nato dalla collaborazione tra Università degli studi di Bologna, Università degli studi di Piacenza e il Centro di Ricerca per le Produzioni Vegetali. MoReFarming si pone come obiettivo quello di fornire una piattaforma Big-Data, aperta e gratuitamente usufruibile a tutti, per il monitoraggio delle agricolture e della qualità del territorio sia regionale che, in futuro, nazionale.

L'applicativo realizzato vuole modellare un primo prototipo del processo ETL che il progetto MoReFarming dovrà sfruttare per popolare la piattaforma Big-Data. Lo scopo di questo applicativo è quindi quello di automatizzare: il processo di estrazione dei Dati dal server remoto Scientific Data Hub messo a disposizione da ESA usufruendo delle loro RESTful API, il processo trasformazione dei pacchetti di livello 1C forniti dal satellite Sentinel-2 ed infine il processo di caricamento dei dati elaborati sulle piattaforme HDFS e PostGIS, per ora con istanza nelle macchine a disposizione del laboratorio di Business Intelligence nella sede della facoltà di Ingegneria e Scienze Informatiche di Cesena.

I principali requisiti dunque richiesti da tale applicativo sono:

- Robustezza delle fasi appena presentate che compongono il processo. Si cercherà quindi di garantire il funzionamento dell'applicativo anche in caso di guasti.
- Flessibilità e scalabilità dell'architettura e delle sue componenti in modo tale da facilitare possibili integrazioni e sviluppi futuri.
- Rendere disponibile una interfaccia di configurazione semplice che possa essere modificabile da riga di comando.

3.2 Architettura

3.2.1 Architettura Funzionale

Per la realizzazione di questo applicativo si è deciso di suddividerlo in 3 principali componenti (vedi Figura 3.1): un componente modellante il processo di estrazione dei dati dal server remoto (Extractor), un secondo modulo determina le varie azioni di trasformazione a cui i dati dovranno essere sottoposti (Transformer) ed infine una componente che si occupa di modellare il processo di caricamento dei dati elaborati e grezzi sulle piattaforme di storage (Loader).

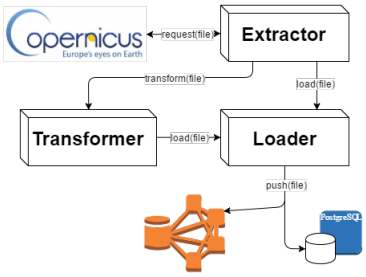

Figura 3.1: Architettura Funzionale dell'applicativo.

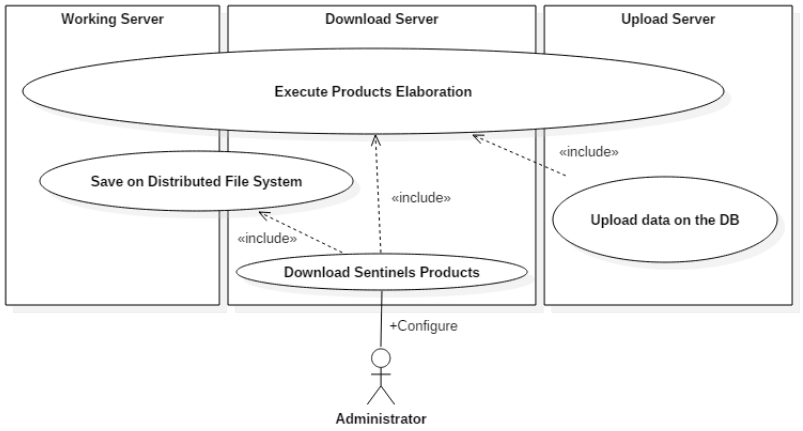

Figura 3.2: Diagramma dei casi d'uso dell'applicativo.

Come possiamo osservare dal diagramma dei casi d'uso (vedi Figura: 3.2 riportata sopra), vi è un unico punto d'interazione attraverso cui l'utente - amministratore - può interagire con l'applicativo. Più nello specifico tale interazione sarà data attraverso il file di configurazione che l'applicativo utilizzerà

per completare la sua inizializzazione. Il processo di Load, come si nota, è suddiviso su 2 istanze: il cluster su cui è presente il file system distribuito e il server su cui è presente il database geo-relazionale, i quali risultano essere di fatto su due sistemi differenti. Il modulo di Transform è invece distribuito sulle 3 istanze principali in cui è suddiviso il sistema: si è deciso in tal modo sia per motivi di bilanciamento del carico e di ottimizzazione di tutto il procedimento, sia a causa di limiti imposti dalle piattaforme, come vedremo poi in seguito.

3.2.2 Master-Slave Model

Più nello specifico, per la modellazione architetturale dell'applicativo si è deciso di sfruttare una **architettura centralizzata di tipo Master-Slave** (vedi Figura 3.3). Questo tipo di architettura ci permette fisicamente di suddividere le componenti del processo ETL in singole unità di elaborazione e di cedere l'onere della parte computazionale agli altri nodi del cluster.

In particolare, per questo applicativo, si è deciso di realizzare l'architettura master-slave attraverso l'uso di un modello client-server. Tutti i processi client sono localizzati sulla medesima macchina, che sarà quindi identificata come nodo master. I nodi server, di conseguenza, rappresentano gli slave e il loro numero può variare in maniera potenzialmente non definita. Ad ogni processo client verrà sottoposto il caricamento di uno o più file scaricati dal server remoto sul file-system distribuito, a seguito del quale il client comunicherà ad uno dei server slave di eseguire l'elaborazione del file appena caricato. Il server, a sua volta, quando avrà completato una serie di task di elaborazione, caricherà i nuovi file generati negli appositi storage condivisi.

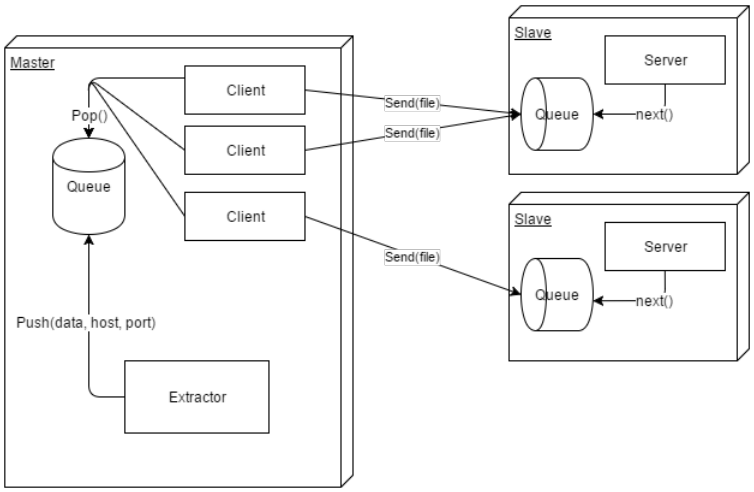

Figura 3.3: Immagine rappresentate il modello architetturale di tipo Master-Slave usato all'interno del progetto.

Dall'immagine 3.3 possiamo osservare che il processo di Estrazione comunica con i servizi client attraverso una coda, eseguendo una push dei dati necessari (dati, hostname e porta dello slave) all'interno del buffer. I client, a loro volta eseguono una pop dalla coda per reperire i dati più vecchi presenti. I server, infine, presentano anch'essi una coda, all'interno della quale vengono posti dati inviati dal servizio client in attesa dell'esecuzione. **Ma come fa il nodo master a sapere chi sono i server slave?**

3.2.3 Publish-Subscribe Pattern

All'interno del modello dell'architettura Master-Slave, le modalità con cui il nodo master identifica i nodi slave sono state definite attraverso l'uso di un pattern Publish-Subscribe semplificato (vedi Figura: 3.4). Con questo pattern è possibile ora rispondere alla domanda effettuata nella sezione precedente, ovvero come è possibile al nodo master conoscere quali sono i suoi slave.

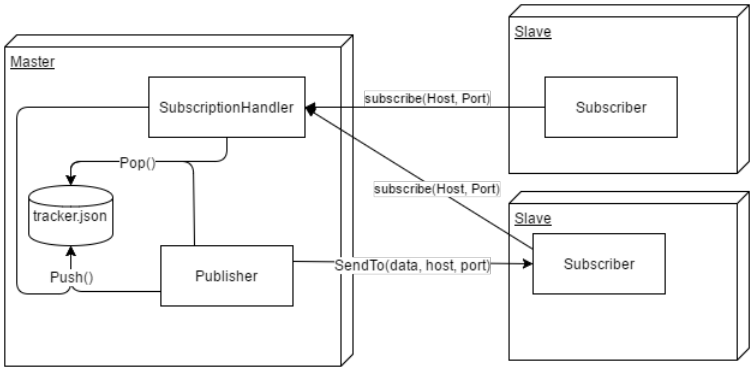

Figura 3.4: Schema del pattern Publish-Subscribe semplificato utilizzato all'interno del prototipo.

Normalmente, in un pattern publish-subscribe classico, l'elemento che modella il subscriber si iscrive ad una serie di topic e riceve notifiche e richieste solo riguardanti ai quegli specifici interessi. Dal momento che questo sistema contiene un solo topic, ovvero l'esecuzione delle operazioni di trasformazione dei dati, si è deciso di non modellare la gestione dei topic ma di eseguire direttamente l'iscrizione alla lista dei server (tracker) che il publisher, nel nostro caso il processo di estrazione, userà per scegliere i server da contattare. Il nodo che vuole iscriversi al servizio invia al SubscriptionHandler il proprio hostname e porta attraverso i quali sarà possibile contattarlo. Il gestore delle iscrizioni aggiungerà, dopo aver verificato la correttezza del formato dell'iscrizione, tale host al file tracker.

3.3 I Processi di caricamento

In questo capitolo verranno esposte le metodologie utilizzate per implementare le varie componenti del processo ETL.

3.3.1 Overview

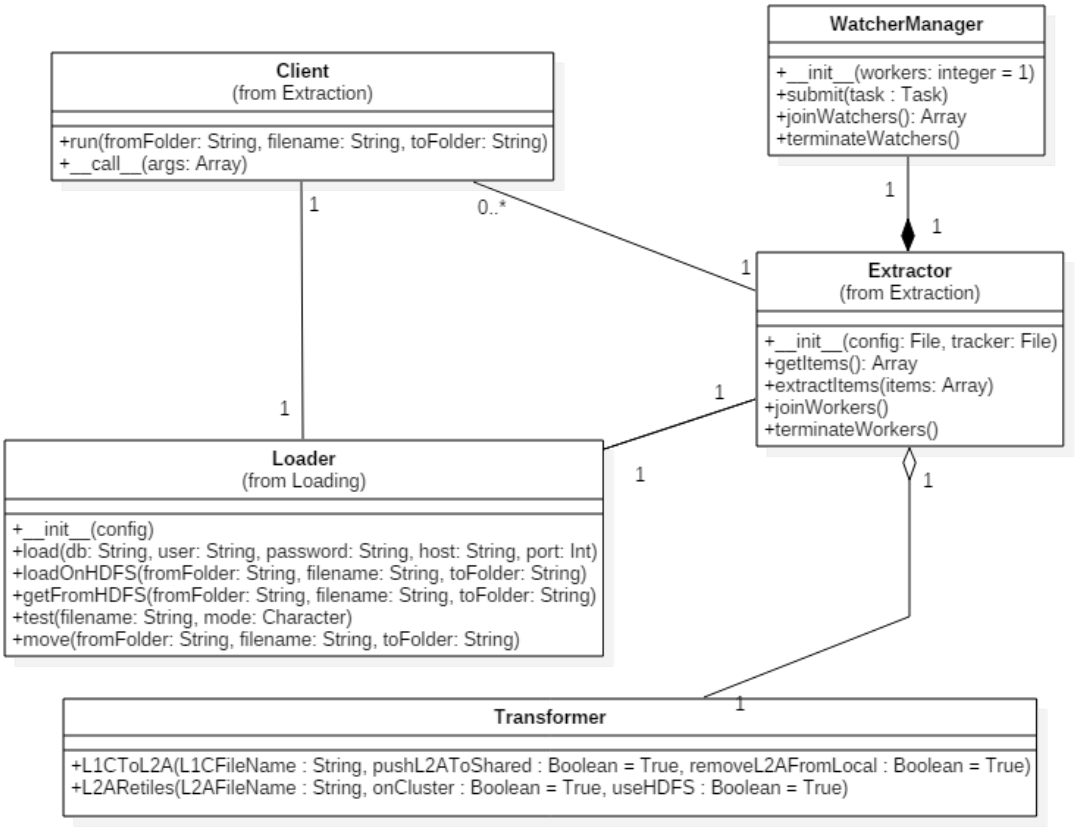

Figura 3.5: Da questa immagine possiamo osservare come la classe Extractor risulti essere il fulcro dell'intero sistema. Essa esegue il download e subordina l'assegnamento dei file scaricati ai client che dovranno caricarli su HDFS. La classe Loader fornisce i metodi necessari ad interfacciarsi con Hadoop Distributed File-System, per eseguire l'upload automatico dei file su PostGIS e per spostare le risorse da una directory del file-system all'altra. Infine la classe Transformer, wrappata dalla classe Extractor, fornisce i metodi necessari ad eseguire le trasformazioni dei file. La classe WatchersManager è il componente a cui l'Extractor affida la gestione dei processi implementanti i servizi client.

3.3.2 Iterazione dei Componenti

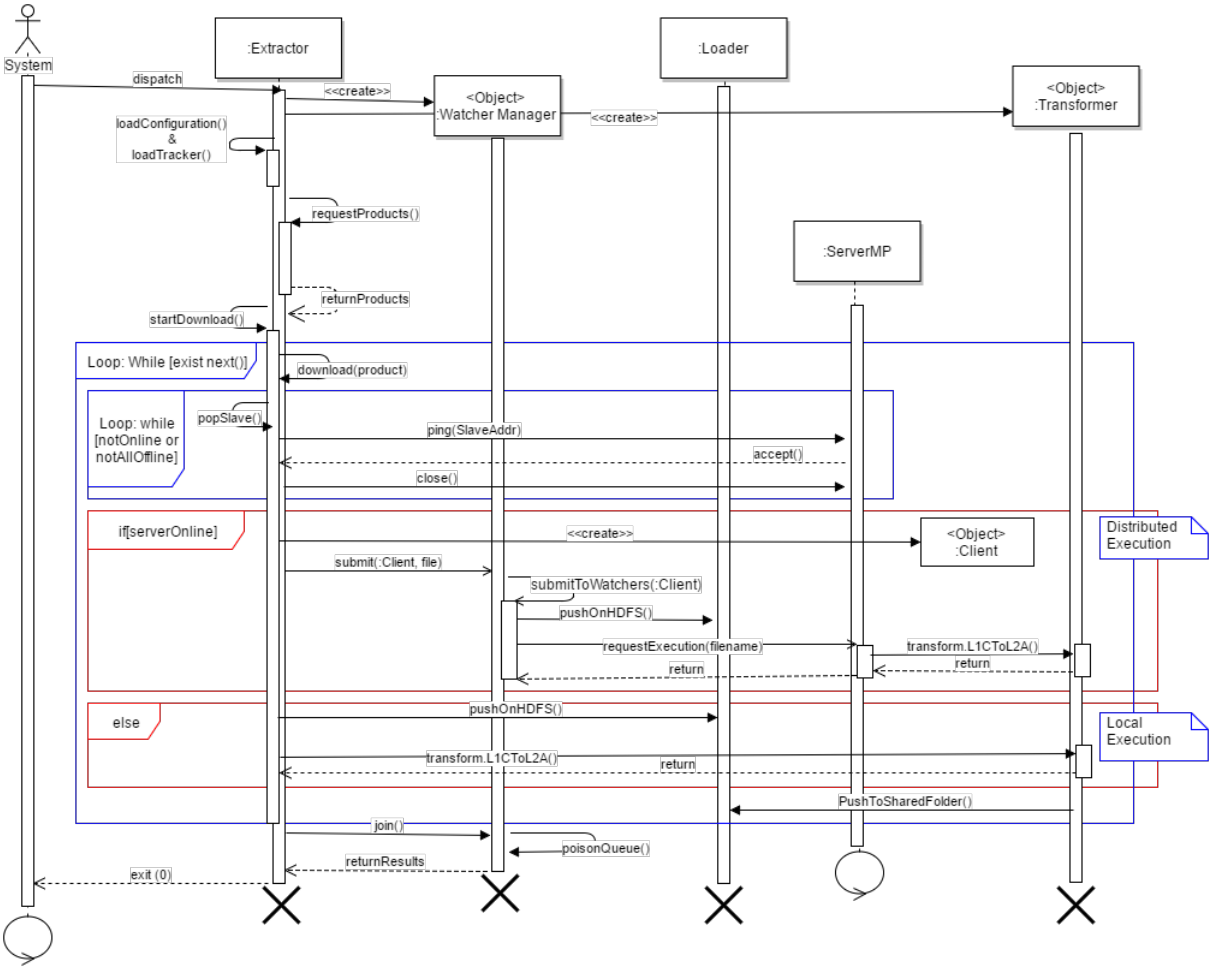

Figura 3.6: Questa immagine riassume l'insieme delle interazioni tra le componenti del sistema. Il sistema operativo, ad intervalli di tempo predefiniti (operazione possibile utilizzando le crontab di CentOS), avvia l'esecuzione dello script che implementa la classe Extractor, il quale eseguirà le sue routine preliminari e darà il via alla fase di download. Il WatchersManager è la componente che intercede nelle comunicazioni tra l'Extractor e i processi Client, essa incapsula tutte le logiche di gestione dei servizi. Come puntualizzato nella precedente figura 3.5, la classe Extractor, avendo associazioni con tutte le classi, risulta essere il fulcro del sistema. Questo perché l'Extractor, in caso non vi siano slave capaci di soddisfare le richiste di elaborazione, esegue le computazioni in locale ed in modo sequenziale.

3.3.3 Extract

In questa sezione verrà esaminato nello specifico il processo di estrazione dei dati, presentandone le logiche e la sequenza di operazioni eseguite dal processo (vedi Figura: 3.7).

Workflow

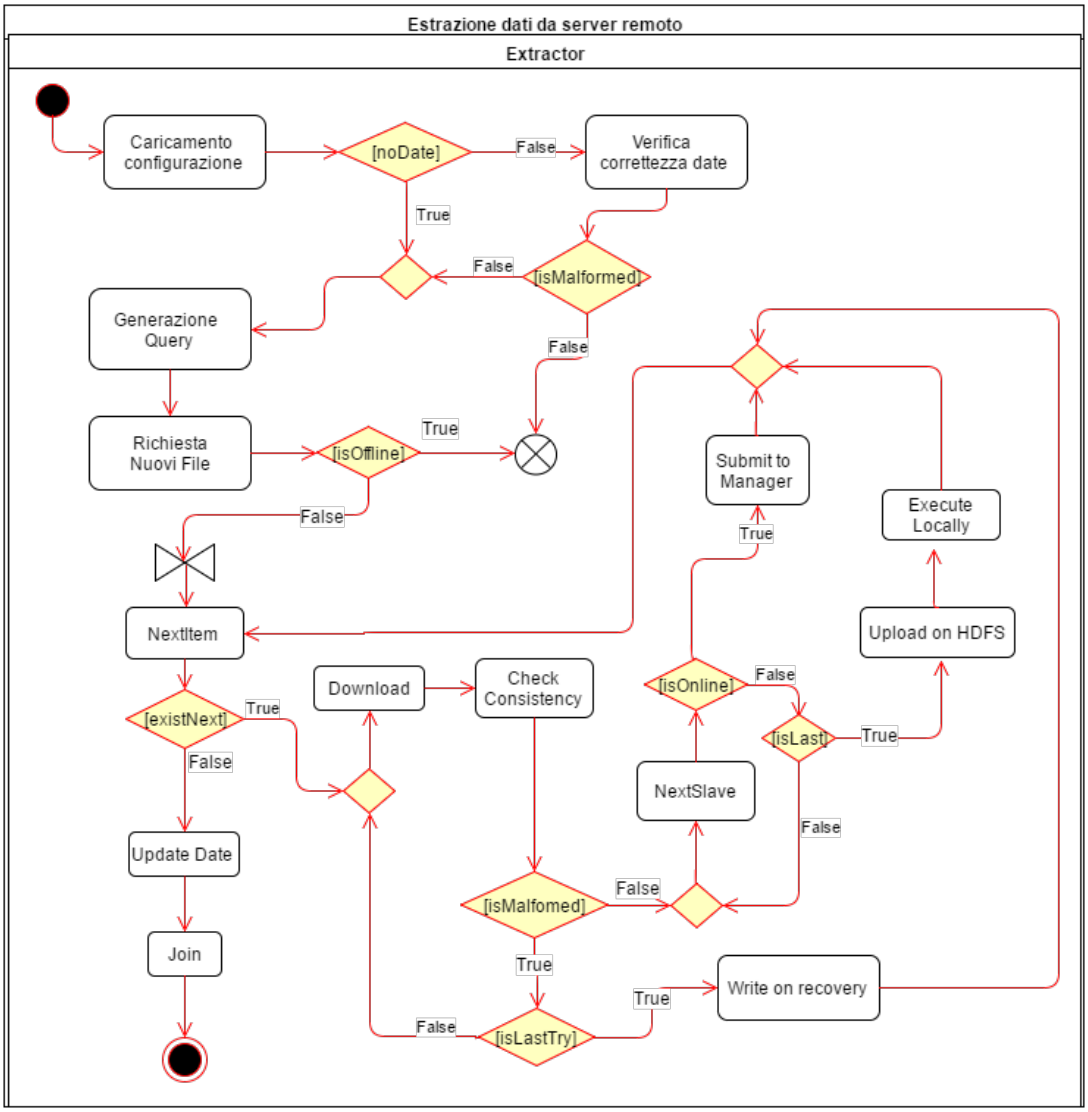

Figura 3.7: Diagramma delle attività svolte dal processo di estrazione dei dati.

Nel momento in cui l'Extractor viene istanziato, esegue l'inizializzazione caricando il file di configurazione (in formato .json), nella quale sono presenti tutti i dati necessari per conoscere in quali directory allocare e leggere i dati e quali sono i file di supporto (come ad esempio il file di recovery dei download falliti). Nel file di configurazione può essere specificata, su decisione dell'utente, una flag che indica se utilizzare o meno un range di date all'interno della quale si desiderano dei particolari dati (di default è settata a True). Nel caso in cui tale flag non sia specificata verrà eseguito il download di tutti i file presenti fin'ora sul server remoto e che soddisfino le condizioni imposte dalla query. Anche le condizioni di ricerca della query sono situate all'interno del file di configurazione, questo permette una flessibilità notevole nell'esecuzione delle richieste, rendendo possibile personalizzare quali file verranno scaricati di volta in volta. Nel caso le date specificate non siano corrette si riceverà un errore.

Prima di poter iniziare il download dei pacchetti è necessario richiedere al server remoto dell'ESA quali siano tali pacchetti (identificati da un identifier - nome del pacchetto - e/o un uuid, entrambi i valori sono univoci). Infatti la query che è specificata nel file di configurazione, non serve ad eseguire il download diretto di tutti i file che soddisfano i requisiti richiesti, bensì è una richiesta attraverso cui il server remoto ci comunica ***quali*** sono i *metadati* dei pacchetti che soddisfano le caratteristiche richieste. La risposta del server è disponibile in 2 formati: JSON e XML. Per facilità d'uso, più che per performance è stato deciso di usare il formato XML. Inizialmente il server permetteva, attraverso le API di OpenSearch, di richiedere un qualsiasi numero di entry nel file di risposta ma, da fine Ottobre 2016, hanno modificato le policy di servizio, limitando il numero di entry per ogni richiesta a 100. Di conseguenza l'attività di reperimento dei nuovi elementi esegue tante richieste fino a quando non arriva un file privo di elementi entry.

Il nucleo del processo inizia quando viene letto il primo elemento presente nella lista di entry appena scaricata: si procede inizialmente con la fase di download, a cui segue il controllo di integrità del file. Se il pacchetto risulta malformato, ne si riesegue il download. Il download di uno stesso file verrà rieseguito fino ad un massimo di 3 volte, dopo di che verrà aggiunto al file di recovery.

Completata questa fase l'Extractor estrae dal tracker, utilizzando una politica di tipo **Round Robin**, il primo server e valuta che sia online: se il server esegue l'handshake alla richiesta di connessione allora l'Extractor chiude immediatamente la socket, riposiziona il server in fondo al tracker, genera un servizio client che si connetta al server appena testato e infine richiede al WatchersManager di mandarlo in esecuzione. Se invece il server estratto dal tracker è offline, viene posizionato subito in coda e si procede con il test dello slave successivo. Se tutti gli slave sono offline, l'Extractor effettuerà tutte le

operazioni in locale, invocando direttamente il modulo di Trasformazione.

Volendo essere più precisi il tracker che il processo di estrazione utilizza è una copia del tracker originale, questo per evitare accessi concorrenti in lettura e scrittura da parte dei due servizi. Questa gestione non permette quindi un hot-join dei nodi slave ad una pool di esecuzione che è già attiva.

Giunti al termine dell'esecuzione il processo aggiornerà la data di inizio nel file di configurazione con la data dell'ultimo file scaricato, mentre quella di fine sarà settata a "NOW". Successivamente eseguirà il join dei task.

Fault-Tolerance

Per la gestione del processo di download i principali scenari di fallimento sono:

- L'assenza del file di configurazione nella cartella del progetto, in questo l'applicativo esce dall'esecuzione;
- Le date impostate nel file di configurazione non rispettano lo standard ISO-8601/RFC3339, anche in questo caso l'applicativo termina;
- La risposta dal server remoto contenente gli uuid dei file da scaricare risulta malformata, quando il server invia una risposta in formato XML che il parser non può leggere può essere a causa che la query inviata non è conforme agli standard o che il server stesso sia in mantenimento. In ambo i casi l'applicativo termina l'esecuzione;
- Il server remoto va offline durante durante il download di un pacchetto. Questo è lo scenario peggiore, specialmente se l'Extractor ha mandato in esecuzione differita l'elaborazione dei pacchetti scaricati: in questo caso occorre non far fallire l'intero processo ma limitare i danni portandolo allo stato di terminazione, ovvero dove effettuerà il join dei task client, prima del dovuto. In questo modo lo script attenderà il completamento di tutti servizi client ancora in coda prima di terminare la sua esecuzione, assicurando il completamento dei task di elaborazione ancora in esecuzione.

Durante l'esecuzione in locale, dal momento che l'Extractor invoca direttamente il modulo di elaborazione, occorre gestire anche eventuali errori che possono incorrere durante l'esecuzione del modulo di elaborazione e che verranno approfonditi nella sezione seguente.

3.3.4 Transform

In questa sezione si esaminerà nello specifico il processo di elaborazione dei dati assieme alle logiche utilizzate.

Workflow

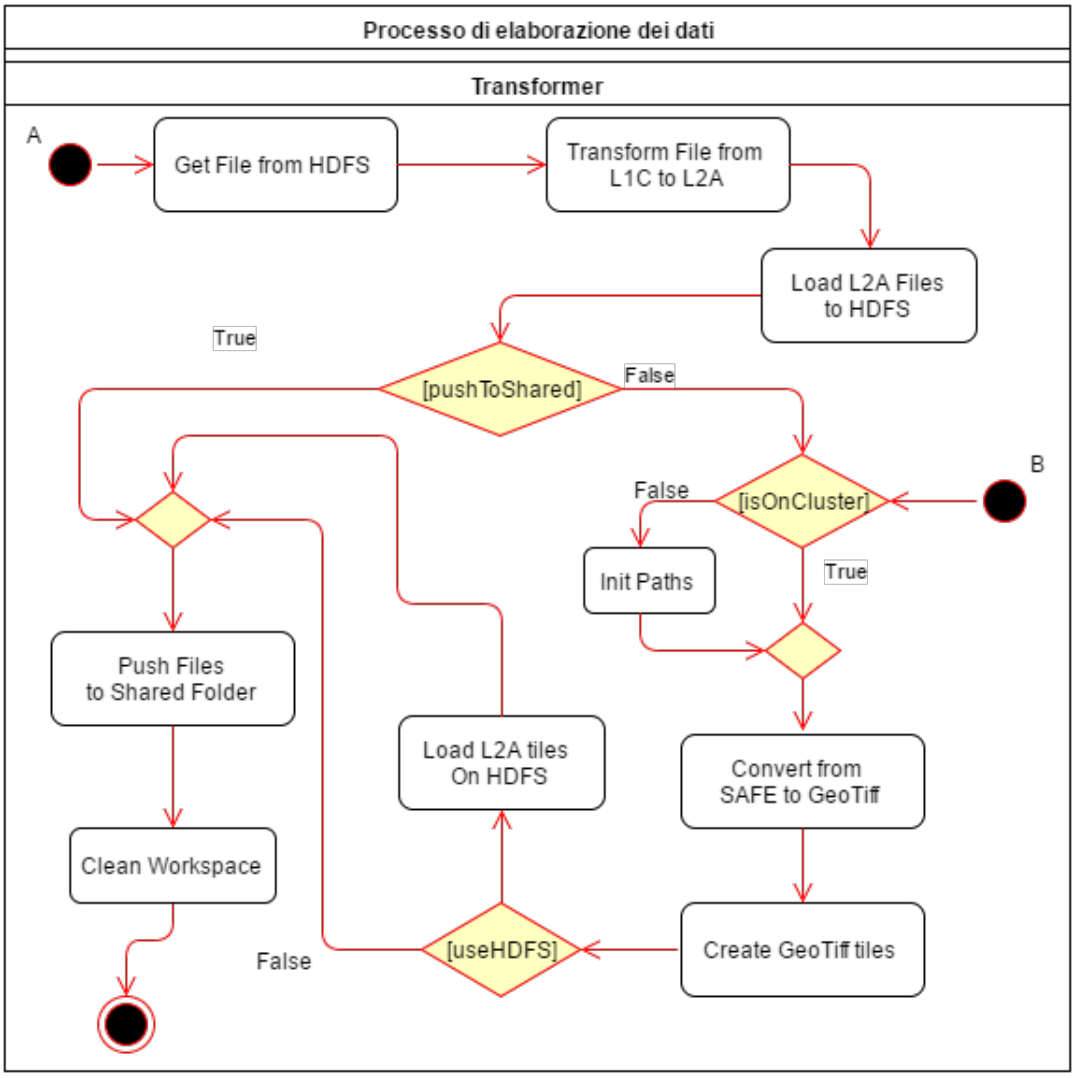

Figura 3.8: Diagramma delle attività svolte dal processo di elaborazione dei dati scaricati.

Il processo di trasformazione dei pacchetti può essere utilizzato da 3 possibili moduli:

- Dal processo di estrazione dei file durante l'esecuzione locale dell'elaborazione.
- Dal RequestHandler del nodo slave.
- Oppure dallo script che esegue il caricamento dei dati su PostGIS.

Nei primi due casi il processo di trasformazione inizia l'esecuzione partendo dal nodo A, esegue la trasformazione dei dati di livello 1C a livello 2A tramite lo strumento di correzione Sen2Cor, carica i dati su HDFS e procede poi al bivio decisionale che, allo stato attuale dell'applicativo, porta sempre ad eseguire una push nella directory condivisa con il server PostGIS. Nel secondo caso (nodo B), ovvero nello script che esegue il Loader (e non nel Loader stesso, come è possibile vedere in figura 3.5), il processo di trasformazione è utilizzato per completare la parte mancante dell'elaborazione dei dati, ovvero la generazione dell'immagine .tif, attraverso il comando gdal_translate, e della sua suddivisione in tile, attraverso il comando gdal_retile. Una volta completate le operazioni, i file generati verranno spostati su un'apposita cartella presente nella directory condivisa.

L'intero processo di trasformazione è stato pensato anche per essere anche eseguito interamente sul cluster, sulla quale però non è possibile utilizzare, per ora, le istruzioni di GDAL.

Incapsulamento dei tool esterni

Gli strumenti utilizzati per l'elaborazione delle immagini non sono stati importati direttamente nell'applicativo, bensì sono utilizzati attraverso un modulo, chiamato Subprocess, fornito da Python che permette di comunicare con il sistema operativo tramite l'uso di costrutti Pipe. Subprocess.call() ci consente di invocare in maniera sincrona un applicativo, specificando la stringa di comando bash da inviare al sistema. I comandi verranno eseguiti su processi separati, figli del processo che ha invocato la chiamata di subprocess, e ritorneranno l'output del comando o un errore, in caso ve ne siano. Malgrado questo metodo renda molto facile sfruttare elementi esterni, rende anche l'applicativo molto meno flessibile e non permette di gestire il flusso di controllo dei comandi utilizzati, rendendo il tutto meno resistente ai guasti.

Il subprocessing è stato utilizzato per incapsulare le 3 componenti esterne utilizzate dall'applicativo: HDFS, Sen2Cor, GDAL e le utility di GeoServer per caricare i dati (jdbc-mapping). Si è scelto di agire in questo modo per molteplici motivazioni:

- I sorgenti di gdal_translate non sono forniti assieme alla distribuzione di OSGeo4 (e sono in ogni caso scritti in linguaggio C++)

- Quelli di Sen2Cor non sono stati pensati per invocare i suoi metodi da un altro codice Python, rendendo impossibile il passaggio di parametri al metodo main.

- Similmente a Sen2Cor, anche i sorgenti di gdal_retile sono scritti linguaggio Python (questo perché gdal_retile non fa parte ufficialmente della libreria) e sono messi a disposizione in questo caso da OSGeo4W, una suite che mette a disposione GDAL ed altri software per l'elaborazione di dati geo-spaziali per piattaforme Windows, ma pur importandoli nel progetto non è possibile sfruttarli.

- La medesima condizione vale per l'incapsulamento delle utility a riga di comando di HDFS (altrimenti accessibili tramite le REST API di Hadoop) e quelle di GeoServer per caricare i dati su PostGIS.

In particolare, l'incapsulamento di gdal_translate per essere eseguito correttamente ha richiesto che l'applicativo fosse avviato attraverso una particolare shell messa a disposizione da OSGeo4W, la quale avvia una serie di driver necessari a creare l'ambiente di esecuzione per il tool.

Inizialmente si puntava ad eseguire tutta la parte di elaborazione sul cluster CentOS ma attualmente non è stato possibile effettuare un build e installazione, tramite CMake dei sorgenti, di GDAL che fosse funzionante e non entrasse in conflitto con le variabili d'ambiente per la versione di GDAL custom utilizzata ed inglobata da Sen2Cor. Inoltre la piattaforma CentOS, attualmente, non risulta presente tra le piattaforme supportate da GDAL.

Limiti imposti dai sistemi: Windows e CentOS

Come già accennato nella precedente sezione, l'idea di iniziale prevedeva di eseguire l'intera operazione di elaborazione sul cluster CentOS ma a causa della incompatibilità di alcuni componenti esterni utilizzati (gdal_translate in primis) si è ritenuto necessario spostare una parte della computazione all'interno del server Windows. Il server eseguirà perciò le operazioni di conversione del pacchetto in geotiff e la sua suddivisione in tile oltre a quelle di caricamento sul database geo-relazionale PostGIS. Alla fine delle operazioni, lo script salverà i file sezionati in una apposita cartella della directory condivisa. Questo dirottamente di una parte di elaborazione ha costretto ad aggiungere delle funzionalità di recupero al processo Loader, le quali verranno illustrate nella prossima sezione.

Inoltre, va sottolineato che sul server, pur essendo attualmente l'unica piattaforma sulla quale è possibile utilizzare lo strumento di conversione gdal_translate, esiste sempre il limite che l'esecuzione è subordinata all'ambiente fornito dalla shell di comando di OSGeo4W.

Fault-Tolerance

Dal momento che i tool esterni sono utilizzati attraverso il servizio di subprocessing, la gestione dei fallimenti all'interno dell'applicativo non può fare altro che gestire gli errori esternati da tali processi, mentre quelli interni sono forzatamente lasciati alla bontà del software esterno. E poiché che queste operazioni devono eseguire su dei server slave, la cui prerogativa è quella di restare sempre online, ci si è concentrati per prima cosa nel catturare gli errori che potevano causare il totale arresto del servizio slave, esternando nel log eventuali errori ed inviando un apposito messaggio al servizio client presente sul nodo master. Similmente, nell'esecuzione locale, si è fatto in modo di non fare arrestare lo script in maniera istantanea, se non prima di aver effettuato il join dei task in esecuzione.

I principali casi di fallimenti presentabili possono dunque essere:

- La sintassi errata del comando di elaborazione (sen2cor, gdal o jdbc-mapping).
- Sen2Cor risulta molto sensibile ai guasti, i quali possono derivare sia dal file elaborato (malgrado non siano necessariamente malformati) sia da particolari combinazioni di valori all'interno del suo file di configurazione.
- La gdal_translate non è eseguita nell'ambiente di OSGeo4W.

Per i casi appena descritti si hanno 2 possibili approcci:

- Se l'esecuzione del processo di trasformazione è eseguita in locale allora si procede aggiungendo il file in questione all'interno di un file di archiviazione, il cui nome è definito nel file di configurazione:
- Altrimenti, ovvero con l'elaborazione assegnata ad un nodo slave, l'errore porterà alla terminazione del handler di richieste del servizio slave e l'invio del tipo d'errore catturato al servizio client del master. Il client valuterà l'errore: nel caso sia un errore causato dalla mancanza di uno strumento esterno allora scriverà il nome del file sul file di archiviazione, altrimenti tenterà fino ad un massimo di 10 volte la rielaborazione del pacchetto al nodo slave. Al decimo tentativo il file sarà scritto sul file di archiviazione.

3.3.5 Load

In questa sezione si esaminerà nello specifico il processo di caricamento dei dati su PostGIS, assieme alle logiche da esso utilizzate.

Workflow

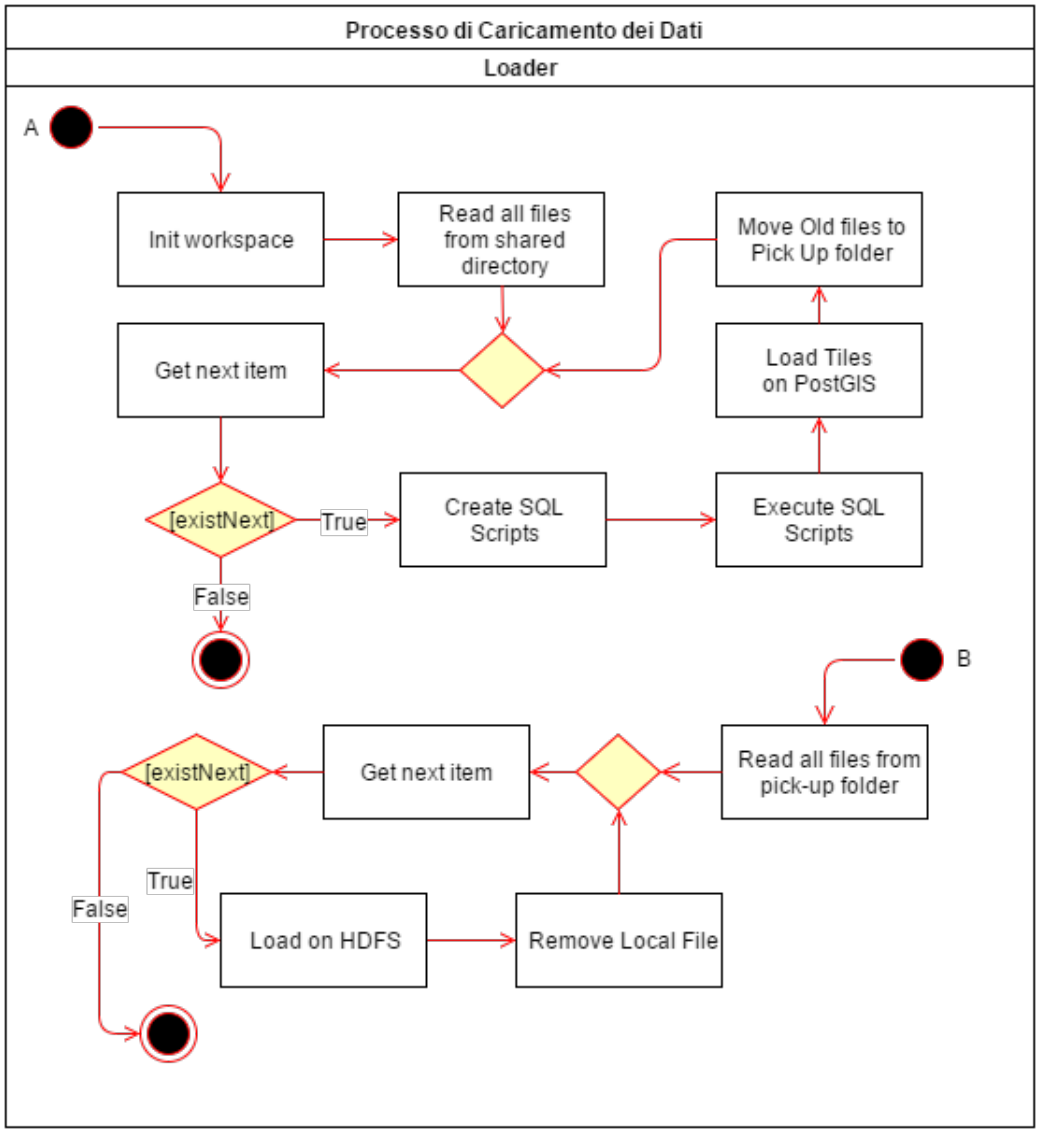

Figura 3.9: Diagramma delle attività svolte dal processo di caricamento dei dati sezionati su PostGIS.

Il processo di caricamento dei pacchetti sul database geo-relazionale PostGIS (diagramma A della figura 3.9) è messo in esecuzione dal sistema operativo Windows ad intervalli alternati rispetto al processo di estrazione. I due processi non comunicano tra di loro, l'operazione svolta dal Loader consiste nel verificare la presenza di nuovi file all'interno della cartella Tiles situata nella directory condivisa dai due sistemi. Tale directory è situata sul server Windows ed è montata come disco di rete nei nodi del cluster CentOS. Il Loader esegue il caricamento dei soli pacchetti di immagini sezionate e generate con il comando gdal_retile. Per l'operazione di caricamento **è necessario un tool esterno messo a disposizione da GeoServer, chiamato gt-image-mosaic-jdbc**, il quale è utilizzato anche per la generazione delle query SQL che creeranno le tabelle della base dati.

Inoltre la classe Loader fornisce, ingloba e modella al suo interno le utility per interfacciarsi con HDFS. Attualmente sono state implementate le operazioni di put (file-system locale a HDFS), get (HDFS a file-system locale) e test (ritorna true o false a seconda che un file sia presente o meno su HDFS, qualunque sia la sua directory su di esso).

Come menzionato precedentemente, a causa dello spostamento di una parte delle operazioni di elaborazione sul server Windows, si è ritenuto necessario aggiungere una ulteriore funzionalità al Loader. Tale funzionalità aggiuntiva è mostrata in figura 3.9 dal diagramma di attività B. Il suo scopo è quello di salvare i pacchetti di immagini sezionate, precedentemente salvate nella Pick-up folder nella directory condivisa, sul file system distribuito. **Questo però porta come conseguenza la necessità di schedulare anche questo processo all'interno delle *crontab* di CentOS**.

Limiti imposti dai tool utilizzati

Come già detto le operazioni di generazione delle query SQL e di caricamento delle immagini suddivise in tile sul database PostGIS sono eseguite attraverso l'uso di un tool esterno. Questo tool, purtroppo, si è scoperto non avere il comportamento pensato. Il tool è impostato per generare **un database per immagine**, all'interno del quale sono presenti una tabella *Mosaic* e *tante tabelle Tiles quanti sono i livelli di zoom della piramide* costruita con la gdal_retile. La tabella Mosaic contiene come entry delle tuple così composte: (Name, LevelName, Date), dove la colonna LevelName è **il nome della tabella contenente uno specifico livello di zoom della piramide**. Di conseguenza Mosaic conterrà tante entry quante sono le tabelle Tiles. Mentre le tuple delle tabelle Tiles, le quali oltretutto possiedono tutte la stessa struttura, sono così formate: (name, data, checksum). Questa (orribile) struttura comporta che *le entry delle tabelle relative ai livelli di zoom non contengo-*

no affatto, come chiave importata, la chiave primaria dell'entry nella tabella Mosaic relativa alla loro immagine. Di conseguenza non è possibile nemmeno caricare più immagini all'interno delle stesse tabelle poiché la struttura stessa della DB non consente di capire a quale immagine corrispondano le entry delle tabelle Tiles poiché non ne esiste affetto un riferimento! L'unico "riferimento" alle Tiles è quello presente nelle entry della tabella Mosaic il quale però non punta ad un array di entry (il che sarebbe comunque male poiché il diagramma ER associato non sarebbe in Prima forma normale) ma bensì ad una intera tabella.

Attualmente si è riusciti a utilizzare gli script creando un solo database ma al cui interno sono presenti tante tabelle mosaic (identificate da {data caricamento su DB remoto}_{data inizio cattura}_orbita_{data fine cattura}_{risoluzione atmosferica}_mosaic) quante sono le immagini e tante tabelle di zoom per ogni immagine e per ogni livello. Benché questa struttura sia meno dispersiva, resta in ogni caso **inutilizzabile in prospettiva ad un uso reale del database**.

Fault-Tolerance

Il principale caso di fallimento è legato al processo di caricamento delle immagini sezionate: l'applicativo può fallire a causa di un formato file non riconosciuto o a causa di tabelle inesistenti. Infatti come è stato menzionato sopra, le tabelle hanno un nome formattato a partire dal nome originale del file ma a cui sono state rimosse tutte le lettere in modo tale da rientrare nei limiti di lunghezza imposta da PostGIS per i nomi delle tabelle.

Questi tipi di fallimento sono stati gestiti in maniera molto semplice, ovvero saltando direttamente il caricamento del file. Visto che il file resterà memorizzato nella directory condivisa finché non verrà caricato, si attende semplicemente che il file verrà caricato al prossimo caricamento schedulato dal sistema. In caso non sia possibile caricarlo occorre agire manualmente.

3.4 Concorrenza e Parallelismo

In questa sezione verranno trattate in maniera approfondita le metodologie attraverso cui sono stati gestisti i servizi client e slave all'interno dell'applicativo.

3.4.1 Gestione degli slave

Come già spiegato, il processo di estrazione testa le connessioni con i server presenti nel tracker finché non ne trova uno online capace di soddisfare la propria richiesta. Dopodiché crea un task contenente il Client da eseguire e lo invia al WatchersManager, il quale si occuperà della sua esecuzione. Occorre ora spiegare che cos'è l'applicativo slave a cui il client generato si connette e qual è la sequenza di azioni che lo coinvolgono.

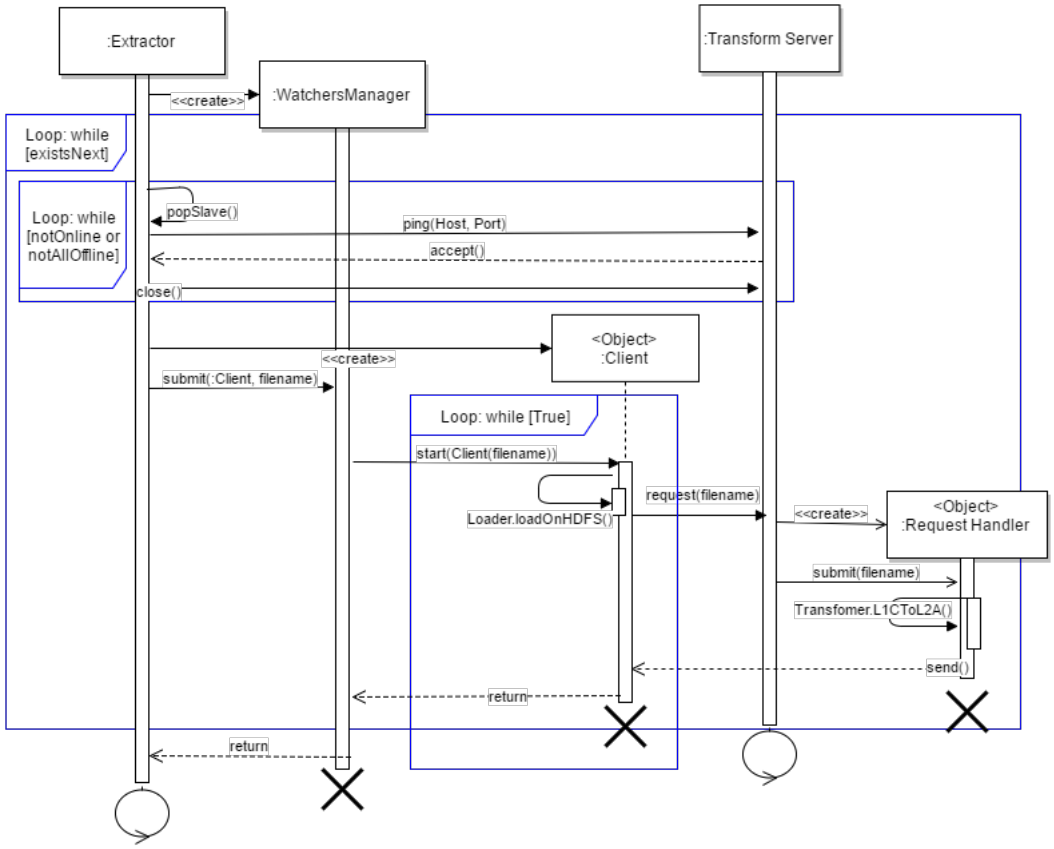

Figura 3.10: Diagramma di sequenza delle interazione tra le componenti modellanti i servizi client del Master e i servizi server modellanti gli slave.

Analizziamo inizialmente le interazioni che coinvolgono lo slave attraverso la figura 3.10. Il processo slave è un processo multi-server, ovvero capace di soddisfare più richieste dei client contemporaneamente. Quando una richiesta giunge al server, essa viene dirottata verso il primo RequestHandler attivo, il quale si occuperà anche di inviare il messaggio di risposta. **Il RequestHandler è l'oggetto che fisicamente realizza il processo di Trasformazione**, qualora l'elaborazione sia stata affidata ai nodi slave.

Focalizziamoci ora sull'implementazione effettiva delle classi che compongono il servizio slave attraverso il diagramma delle classi proposto di seguito (Figura: 3.11).

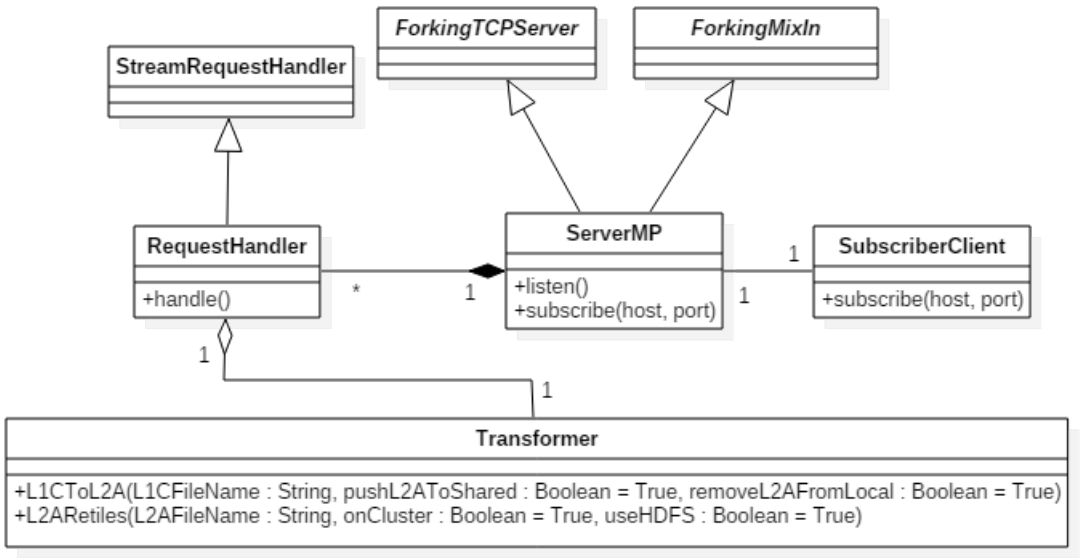

Figura 3.11: Diagramma delle classi relativo alla modellazione del servizio server.

Come si nota dalla figura 3.11, la classe implementante il servizio server, ServerMP, deriva da 2 superclassi fornite dal linguaggio Python: ForkingMixIn e ForkingTCPServer. La prima rende disponibili le metodologie di creazione e gestione dei processi; la seconda, invece, definisce tutte le strutture necessarie all'uso degli stessi. Inoltre ForkingTCPServer necessita di una classe handler che gestisca le richieste generate ad ogni nuova connessione. La classe RequestHandler svolge tale scopo: essa deriva la superclasse StreamRequestHandler, la quale consente la gestione di uno stream di dati anziché di un singolo pacchetto. Nel nostro specifico caso la classe RequestHandler sfrutta l'entità Transformer per eseguire le elaborazioni dei dati. In tutto questo la gestione

delle richieste da parte del servizio server è completamente **asincrona**. Infatti è il processo figlio stesso, il RequestHandler, ad eseguire l'invio della risposta al client, che invece rimarrà in attesa della risposta del server in maniera sincrona. Questo tipo di server è stato ideato per funzionare su piattaforme Linux, sia perché la gestione dei processi è più performante sia perché queste socket sfruttano il come metodo di generazione il fork dal processo padre, proprio delle piattaforme Posix e non presente in Windows.

Si vuole ora volgere lo sguardo sul servizio che gestisce le richieste di subscribe (Figura 3.12) da parte dei server slave.

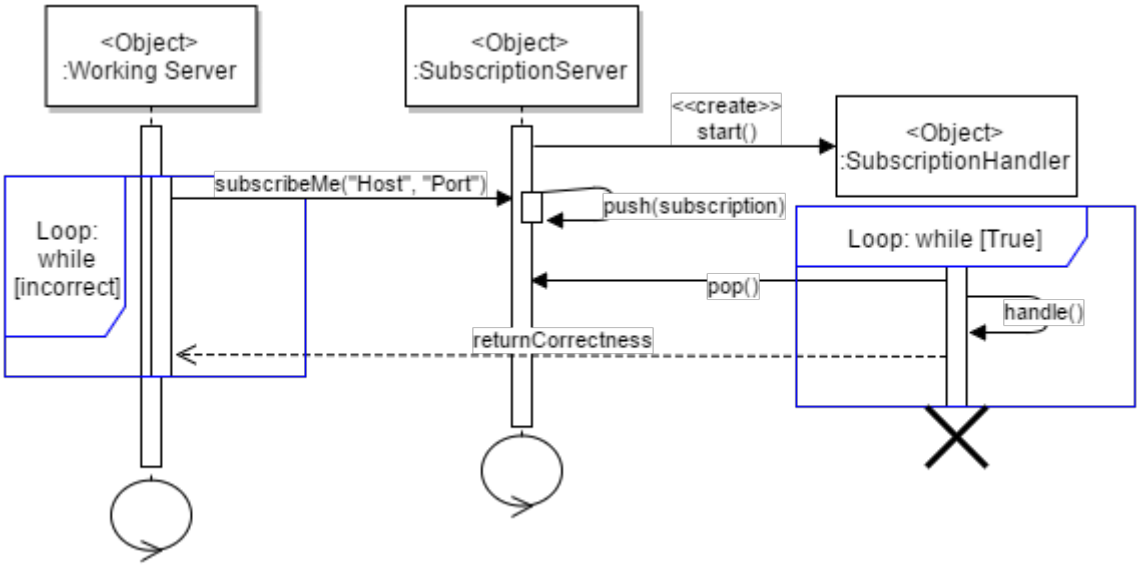

Figura 3.12: Diagramma di sequenza che mostra le interazioni tra i componenti che implementano il pattern publish-subscribe

Come è possibile osservare dalla figura 3.12 il gestore delle iscrizioni è anch'esso implementato in modo tale da poter gestire più richieste contemporaneamente. In questo caso però, data la mole di lavoro poco elevata, il tutto è stato implementato con una socket multi-thread. Il server in ascolto, al giungere di una nuova iscrizione, esegue una push del dato in una coda condivisa con il thread che gestisce il request handler. Il thread esegue una pop dalla coda fin tanto che sono presenti elementi, valutando se le iscrizioni sono corrette o meno, e aggiunge il nuovo host al tracker. Il risultato dell'iscrizione viene inviato al servizio client che l'ha effettuata. Se un subscriber effettua una nuova iscrizione ma con una porta differente allora il SubscriptionManager aggiornerà il relativo host del tracker con il nuovo valore della porta.

3.4.2 Gestione dei servizi client

Come si già è detto, il processo di estrazione una volta scaricato un pacchetto lo sottomette al WatchersManager, ma come lavora al suo interno tale componente?

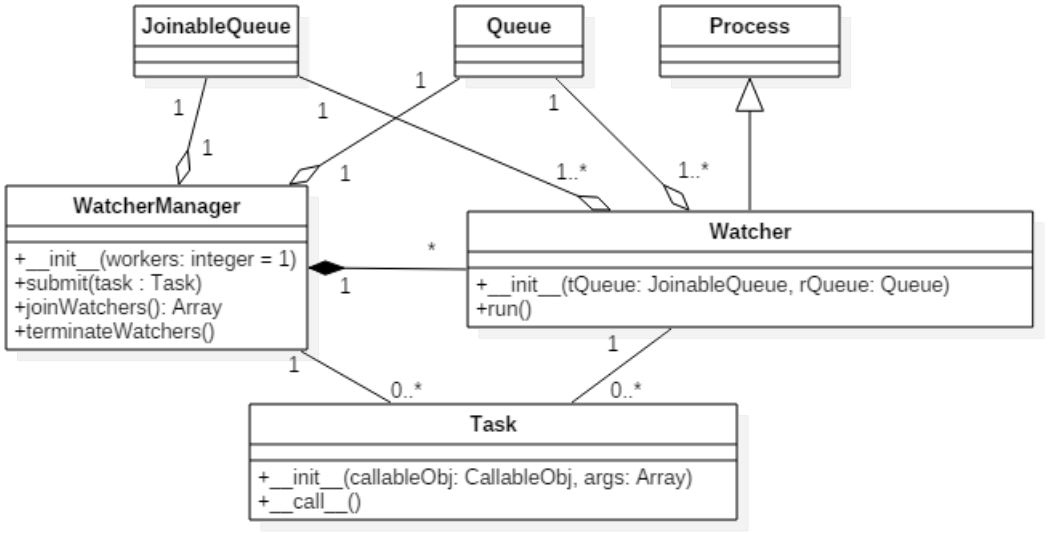

Figura 3.13: Diagramma delle classi per le componenti watcher.

La figura 3.13 ci mostra l'insieme di classi che compongono il WatchersManager. La classe WatchersManager implementa al suo interno 2 code: una JoinableQueue ed una Queue, costrutti entrambi forniti da Python e che modellano una coda contenente elementi di controllo della concorrenza tra Processi o Threads. La peculiarità della JoinableQueue che la differenzia dalla Queue è quello di permettere il join dei processi che attendono in lettura su quella coda. La JoinableQueue è utilizzata dal WatchersManager per memorizzare gli oggetti di tipo Task che dovranno essere eseguiti dai processi Watcher, mentre la Queue è utilizzata dai Watcher per inserire i valori di ritorno degli oggetti Task eseguiti. Il parametro passato al WatchersManager al momento dell'inizializzazione indica qual è il numero massimo di oggetti Watcher che il Manager dovrà gestire. La classe Watcher modella un processo che, in un loop, legge dalla JoinableQueue finché non diviene vuota, esegue in maniera **sincrona** il task estratto ed esegue infine una push del valore di ritorno all'interno della coda dei risultati.

Il WatcherManager può effettuare la Terminazione o il Join dei suoi processi Watcher, i quali sono dunque gestiti in maniera **asincrona**. L'operazione di Join è effettuata attraverso la tecnica della Poison Pill: si un oggetto all'interno

della coda che permetta al Watcher di uscire dal loop e concludere in modo non brusco la sua esecuzione. L'oggetto utilizzato nel nostro caso è None, oggetto che modella il NULL in Python. Il numero di pillole avvelenate è ovviamente in proporzione 1:1 con il numero di Watcher.

La classe Task modella un oggetto generico Callable che wrappa un altro oggetto/elemento anch'esso Callable. In linguaggio Python gli oggetti Callable sono tutti quegli elementi (Classi o Metodi) che implementano la proprietà *call*, la quale è già definita per i metodi (che per loro natura sono Callable) mentre deve essere esplicitata all'interno delle classi. Questo particolare costrutto permette di creare oggetti evocabili come se fossero dei normali metodi, ciò permette di rendere trasparente la distinzione tra metodi e classi da parte della classe Task. Alla classe Task al momento dell'inizializzazione, oltre al riferimento dell'elemento Callable, vanno specificati anche gli argomenti per il metodo *call*: in questo modo nono è necessario modificare il costruttore della classe Callable per passargli tali valori, che altrimenti avrebbe esposto semantiche inutili, e di lasciare tutta la logica del richiamo all'oggetto Task.

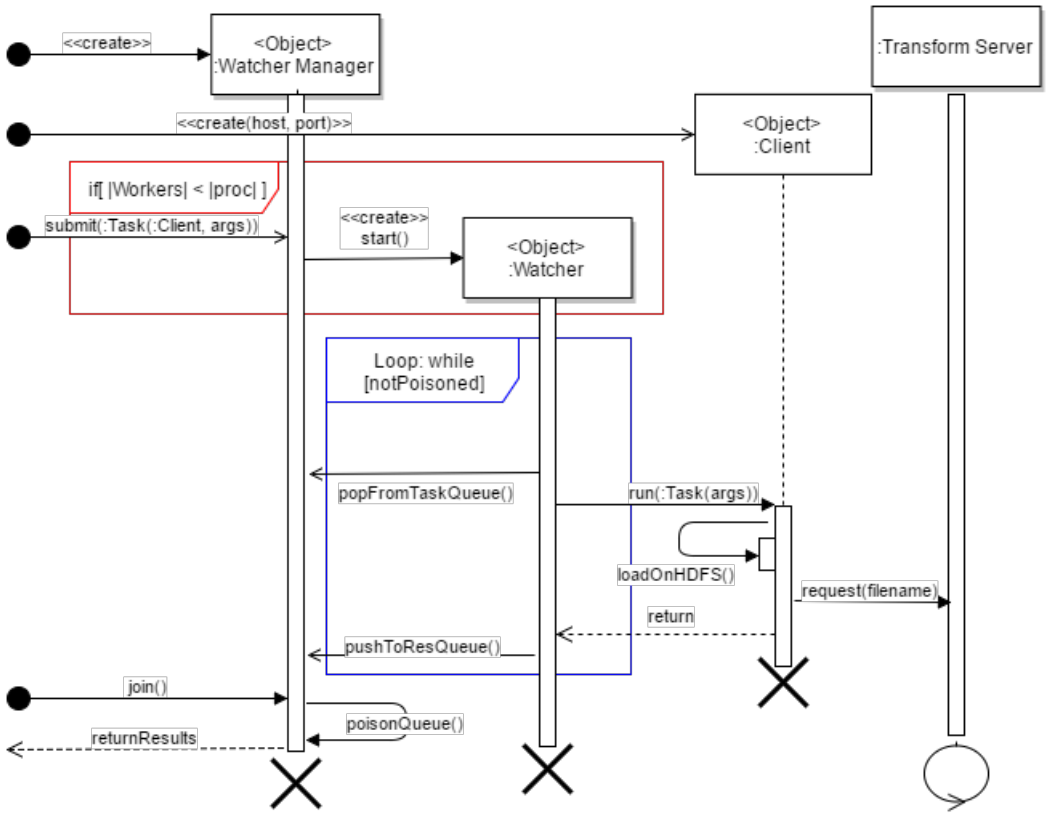

Figura 3.14: Diagramma di sequenza delle interazione che coinvolgono il WatchersManager.

Abbiamo ora tutti gli elementi per comprendere in toto come è gestita l'esecuzione dei servizi client (vedi Figura 3.14): il processo di estrazione, non appena ha completato l'esecuzione di un download, mette in *Pipeline* l'esecuzione del caricamento su HDFS e l'elaborazione dei dati. L'Extractor genera un oggetto Task inizializzato con un oggetto Client (il quale esegue le operazioni sopra menzionate) e con esso i parametri necessari al metodo __call__(): identificatore del file appena scaricato, cartella in cui è situato il file e directory di destinazione di HDFS. L'oggetto è passato al WatcherManager che a sua volta lo posiziona nella JoinableQueue. Il processo Watcher non fa altro che leggere dalla coda ed eseguire un generalissimo oggetto Callable, ignaro dell'astrazione chiamata Client. L'oggetto Client è quindi un elemento monouso il cui unico compito è quello di eseguire l'upload su HDFS e, successivamente, richiedere allo slave l'esecuzione del pacchetto appena caricato e attenderne il risultato.

3.4.3 Python: Processes over Threads

Per la realizzazione di questo applicativo, come metodo di esecuzione asincrona dei task, si è scelto per la maggior parte l'utilizzo dei processi anziché i thread. Questa scelta è dovuta ad un limite imposto dal linguaggio di programmazione scelto per la realizzazione dell'applicativo: Python. Python è un linguaggio interpretato e dinamico, che offre modelli di programmazione sia strutturata che Object-Oriented. Proprio questa sua proprietà di essere interpretato anziché compilato porta con se un importante svantaggio: GIL - Global Interpreter Lock. Questo lock comporta che l'interprete Python può eseguire il codice di un solo thread alla volta e, sebbene si possa eliminare o scavalcare (ad esempio usando Jython - Python su una macchina virtuale Java), è necessario poiché **la gestione della memoria di Python non è thread-safe**. Fortunatamente ogni processo possiede la propria istanza dell'interprete Python ed è quindi possibile eliminare il problema del GIL utilizzando un paradigma di programmazione a processi. D'altro canto però i processi sono dei costrutti la cui implementazione e performance varia da un sistema operativo all'altro: mentre sulle piattaforme Linux risultano molto leggeri (ma sempre meno dei thread), sulle piattaforme Windows i processi sono pesanti e comportano un overhead considerevole per il sistema. Inoltre la comunicazione tra processi non è semplice come quella tra i thread poiché non condividono alcuna area di memoria: è necessario utilizzare i costrutti Pipe per rendere possibile la comunicazione tra processi.

Nell'applicativo i servizi che sfruttano i processi sono principalmente il WatchersManager (che sfrutta le API del sistema per creare i processi figli ed

è quindi multipiattaforma) e il ServerMP (che al contrario utilizza i metodi fork forniti dal linguaggio Python, i quali non sono supportati da Windows).

3.4.4 Limiti

Nella strutturazione di una applicazione che fosse parallela e concorrente si sono dovuti affrontare alcune limitazioni, sia imposte che per tolleranza del sistema. Iniziamo con la limitazione sul numero di download imposta dalla piattaforma online di Copernicus: attualmente ogni account è limitato ad un numero massimo di download paralleli pari a 2. Inizialmente si pensava di distribuire ad ogni nodo l'esecuzione di download (Extraction), elaborazione (Transform) e caricamento (Load), *rendendo di fatto ogni singolo slave un micro-processo ETL completo ed autonomo.* Ciò avrebbe reso anche a livello architetturale un'applicazione meglio strutturata e maggiormente flessibile. Ma, data la limitazione, è solo 1 nodo, il master, quello che esegue i download dei file e ne distribuisce l'elaborazione. E' necessario però puntualizzare che attualmente sarebbe stato controproducente eseguire più download contemporanei, sia a causa della velocità della rete che della tolleranza del server remoto ad un numero di richieste elevato (capita spesso che il server vada down). Ma, nel futuro, avere un'architettura come quella sopra descritta comporterebbe delle migliori prestazioni.

Esiste inoltre una limitazione, necessaria, per l'esecuzione delle task di elaborazione utilizzando l'applicativo esterno Sen2Cor. Infatti l'esecuzione di una sola elaborazione consuma dal 99% al 100% delle risorse CPU up-time e, in dipendenza dalla dimensione della *(*singola) immagine elaborata (che può variare da 100MB fino a 900MB), può occupare fino a 12GB di RAM nei momenti più critici, con un andamento crescente con il procedere dell'elaborazione. Si è scelta dunque una politica per il quale uno slave può eseguire una sola elaborazione alla volta, riducendo perciò il numero di elaborazioni contemporanee al numero di nodi slave. Tale limitazione risulta necessaria poiché il sistema, sebbene ogni nodo sia equipaggiato con 32GB di RAM (di cui 10-12 occupati up-time da risorse del sistema e applicativi vari) non può rischiare di andare in RAM overflow. Anche se possiede un'area di swap relativamente grande, le prestazioni ne risentirebbero considerevolmente e di conseguenza il tempo di elaborazione totale aumenterebbe, tempo che è già abbastanza elevato. In media un pacchetto di circa 600MB viene elaborato in circa 20 minuti, mentre con i formati di pacchetti più vecchi (la cui taglia varia da 3GB fino a 7GB) [1] impiegano circa 10 volte tanto.

[1] I vecchi pacchetti contengono un numero di tile pari a quello delle aree UTM che incidono con i punti geografici descritti dalla footprint. Ma dal 2016/09/29 queste immagini sono distribuite singolarmente.

3.5 Sperimentazione

Si vuole inizialmente presentare una stima teorica delle prestazioni distribuite dell'applicativo in confronto alle prestazioni sequenziali per poi verificare tali dati con le prestazioni sperimentali, calcolate attraverso la funzione `time` di bash. La stima teorica prenderà la nuova composizione dei pacchetti, introdotta in data 2016/09/29, e normalizzerà i pacchetti aventi la vecchia composizione, dividendone la dimensione (MB) per il numero di tile contenute. Inoltre per il calcolo delle statistiche si assume che le prestazioni della rete sottostante e verso il server remoto non siano affette da ritardi e che il tempo di invio di un messaggio da Master a Slave, essendo nella stessa rete, sia irrisorio. Non sono inoltre considerati i tempi di caricamento e reperimento degli oggetti da HDFS.

Definiamo con $P = \{P_1 ... P_n\}$ l'insieme dei pacchetti da noi considerato, normalizzato rispetto al numero di tile contenute all'interno:

$$TotPkts = \sum_{1}^{|P|} |P_i|$$

$$MeanSize = \frac{1}{TotPkts} \sum_{1}^{|P|} Size(P_i)$$

La Varianza relativa alle variabili composte è ricavabile applicando alle rispettive funzioni le seguenti trasformazioni [2] mentre lo scarto quadratico medio sarà dato dalla radice della Varianza:

$$X = aA => \Delta^2_{aA} = a^2 \Delta^2_A$$

$$X = aA + bB => \Delta^2_{aA+bB} = a^2 \Delta^2_A + b^2 \Delta^2_B$$

$$X = A \cdot B => \Delta^2_{A \cdot B} = \Delta^2_{\frac{aA}{bB}} = X^2 \cdot [\frac{\Delta^2_A}{A^2} + \frac{\Delta^2_B}{B^2}]$$

$$\sigma^2_X = \sqrt{\Delta^2_X}$$

Dove X è la funzione di cui si vuole calcolare l'errore e $(\Delta X)^2$ è lo scarto quadratico medio.

[2] `https://en.wikipedia.org/wiki/Propagation_of_uncertainty`

Si presenta ora il modello delle formule utilizzate per effettuare i calcoli:

$$\Delta^2_{MeanSize} = \frac{1}{TotPkts} \cdot \sum_{1}^{|P|} (MeanSize - Size(P_i))^2$$

La velocità media con cui ogni pacchetto sarà scaricato sarà:

$$MeanDLTime = \frac{MeanSize}{DLSpeed}$$

$$\Delta^2_{MeanDLTime} = \frac{1}{DLSpeed^2} \cdot \Delta^2_{MeanSize}$$

Il tempo di elaborazione media per ogni pacchetto può essere calcolato nel seguente modo:

$$MeanExecTime = \frac{1}{TestPkts} \cdot \sum_{1}^{|P|} ExecTime(P_i)$$

$$\Delta^2_{MeanExecTime} = \frac{1}{TotPkts} \cdot \sum_{1}^{|P|} (MeanExecTime - ExecTime(P_i))^2$$

Definiamo quindi qualitativamente la velocità dell'applicativo Sen2Cor attraverso il tempo medio di esecuzione come:

$$RelExecSpeed = \frac{MeanSize}{MeanExecTime}$$

$$\Delta^2_{RelExecSpeed} = RelExecSpeed^2 \cdot \left[\frac{\Delta^2_{MeanSize}}{MeanSize^2} + \frac{\Delta^2_{MeanExecTime}}{MeanExecTime^2} \right]$$

Si descrive ora la formula utilizzata per stimare i tempi di esecuzione seriale e con pipeline:

$$SerialTime = TotPkts \cdot \left(MeanDLTime + \frac{MeanSize}{RelExecSpeed} \right)$$

$$PipelineTime = \left\lceil \frac{TotPkts}{nSlave} \right\rceil \cdot \left(MeanDLTime + \frac{MeanSize}{RelExecSpeed} \right)$$

Ogni nodo scaricherà perciò $\left\lceil \frac{TotPkts}{nSlave} \right\rceil$ pacchetti ad intervalli di $MeanDLTime+$ $MeanExecTime$ poiché ogni download è eseguito ad una distanza di circa $MeanDLTime$ l'uno dall'altro.

La varianza Δ delle formule di costo, nel nostro caso, saranno date dalla composizione delle precedenti:

$$\Delta^2_{Serial} = TotPkts^2 \cdot \left[\Delta^2_{MeanDLTime} + \Delta^2_{\frac{MeanSize}{RelExecSpeed}} \right] =$$

$$= TotPkts^2 \cdot \left\{ \frac{1}{DLSpeed^2} \cdot \Delta^2_{MeanSize} + \frac{MeanSize^2}{RelExecSpeed^2} \cdot \left[\frac{\Delta^2_{MeanSize}}{MeanSize^2} + \frac{\Delta^2_{RelExecSpeed}}{RelExecSpeed^2} \right] \right\} =$$

$$= TotPkts^2 \cdot \left\{ \frac{1}{DLSpeed^2} \cdot \Delta^2_{MeanSize} + \frac{MeanSize^2}{RelExecSpeed^2} \cdot \left[\frac{\Delta^2_{MeanSize}}{MeanSize^2} + \frac{RelExecSpeed^2}{RelExecSpeed^2} \cdot \left(\frac{\Delta^2_{MeanSize}}{MeanSize^2} + \frac{\Delta^2_{MeanExecTime}}{MeanExecTime^2} \right) \right] \right\} =$$

$$= TotPkts^2 \cdot \left\{ \frac{1}{DLSpeed^2} \cdot \Delta^2_{MeanSize} + \frac{MeanSize^2}{RelExecSpeed^2} \cdot \left[2 \cdot \frac{\Delta^2_{MeanSize}}{MeanSize^2} + \frac{\Delta^2_{MeanExecTime}}{MeanExecTime^2} \right] \right\}$$

$$\Delta^2_{Pipeline} = \frac{\Delta^2_{Serial}}{nSlaves^2}$$

Prendiamo come campione i pacchetti contenuti attualmente nella cartella `/morefarming/datalake/L1C` di HDFS. Valutiamo, inoltre, per ogni variabile la relativa deviazione standard [3]:

[3] Le statistiche calcolate sono disponibili presso la sheet al seguente indirizzo `https://goo.gl/82EVNq` o alternativamente in allegato a questa tesi.

$$TotPkts_P = 124$$

$$MeanSize_P = 465.031MB$$

$$\Delta^2_{MeanSize_P} = 159.462MB^2$$

$$\sigma_{Serial} = \sqrt{\Delta^2_{MeanSize_P}} = 12.62MB(Err_{rel} = 2.72\%)$$

La velocità della rete a cui il cluster è collegato è di 84Mb/s in Download e 70Mb/s in Upload (testato attraverso lo speed test da riga di comando presente nella mia home directory del cluster).

$$DLSpeed = 8.4MB/s$$

$$MeanDLTime_P = 55.360s$$

$$\Delta^2_{MeanDLTime_P} = 2.2995s$$

Il calcolo del tempo di esecuzione medio, per semplificare i calcoli, è stato calcolato utilizzando un insieme $T \subset P$ più ristretto di pacchetti ma che potesse esserne rappresentativo. Successivamente è stata stimata una velocità di esecuzione relativa ($RelExecSpeed$), indicativa della velocità dell'applicativo Sen2Cor.

$$MeanExecTime_T = 951.636s$$

$$\Delta MeanExecTime_T{}^2 = 108554.757s^2$$

$$RelExecSpeed = 0.446MB/s$$

$$\Delta^2_{RelExecSpeed} = 0.024s^2$$

Come verifica calcoliamo il tempo di esecuzione medio per i file di P utilizzando l'indice di velocità di Sen2Cor appena calcolato:

$$MeanExecTime_P = 1042.543s$$

Come possiamo osservare i due tempi medi di esecuzione non differiscono di molto (8.7%), possiamo perciò constatare che il sotto-insieme T di pacchetti

preso in valutazione è rappresentativo dell'insieme P. Possiamo perciò valutare la relativa Varianza.

$$\Delta^2_{MeanExecTime_P} = 2 \cdot \frac{\Delta^2_{MeanSize}}{MeanSize^2} + \frac{\Delta^2_{MeanExecTime}}{MeanExecTime^2} = 120907.402s^2$$

$$\sigma_{Serial} = \sqrt{\Delta^2_{MeanExecTime}} = 347.717(Err_{rel} = 33.353\%)$$

Si valutino ora i tempi di esecuzione, sequenziale e distribuito, finali:

$$SerialTime = 136140.0194s = 37.81h$$

$$\Delta^2_{Serial} = 1856657523.12s^2$$

$$\sigma_{Serial} = \sqrt{\Delta^2_{Serial}} = 42972.753s = 11.936h(Err_{rel} = 31.5\%)$$

Se si suppone che l'applicativo sia avviato su un cluster avente 2 nodi slave ed 1 nodo master, il valore del tempo di esecuzione con pipeline + distribuzione dei task sarà:

$$PipelineTime = 68070.0097s = 18.908h$$

$$\Delta^2_{Pipeline} = 464163930.78s^2$$

$$\sigma_{Pipeline} = \sqrt{\Delta^2_{Pipeline}} = 21544.464s = 5.984h(Err_{rel} = 31.65\%)$$

Lo speed-up teorico risulta quindi essere strettamente minore a:

$$SpeedUp = \frac{SerialTime}{PipelineTime} < 2.0 = nSlave$$

Minore poiché nel calcolo non sono state prese in considerazione le latenze di rete e di HDFS che in particolare affliggono l'elaborazione distribuita.

Passiamo ora ai dati derivanti dalla sperimentazione distribuita dell'applicativo. I test sono stati fatti su un insieme di 107 file ed il cluster utilizzato possiede 2 nodi Slave tra cui uno eseguiva il ruolo di nodo Master. Il tempo di elaborazione risultante per tali dati è pari a:

$$PipelineTime_{Real} = 62402.24s = 17.33h$$

Manteniamo ora in proporzione il tempo di esecuzione distribuita con il numero di pacchetti scaricati nella sperimentazione:

$$PipelineTime_{pkts=107} = 58737.83s = 16.316h$$

L'errore relativo Err_{rel} è invariante.

I due dati relativi all'esecuzione distribuita si discostano del 5.851%, discrepanza che rientra pienamente all'interno dell'errore relativo calcolato (31.65%). Possiamo affermare perciò che i dati sperimentali confermano i dati teorici e quindi che lo speed-up dell'applicativo è strettamente inferiore al numero dei nodi slave attivi.

Conclusioni

L'obiettivo principale di un processo ETL è quello di organizzare e facilitare il trasferimento ed elaborazione dei dati dalla sorgente alla destinazione attraverso un procedimento robusto, flessibile e scalabile.

Nella realizzazione di questo progetto si è dovuto affrontare un numero di sfide non indifferente, in particolare l'integrazione di strumentazioni esterne con il progetto è senz'altro una di queste. L'utilizzo delle interfacce di Subprocess fornite da Python ha reso l'integrazione semplice ma ha minato in parte il requisito di *robustezza* richiesta dall'applicativo poiché è stato limitato il potere di gestione che il programma possiede sul flusso di controllo.

Tra i requisiti principali dell'applicativo è presente quello di creare una soluzione flessibile a livello architetturale, obiettivo che si può dire raggiunto grazie alla suddivisione del progetto differenti moduli autonomi ognuno dei quali concerne funzionalità uniche. E' stato inoltre possibile sviluppare una piattaforma scalabile sul numero di nodi grazie all'uso di una architettura master-slave anche se conserva il problema di un punto unico di fallimento tipico delle architetture client-server.

Nell'ambito Big Data si è creato un processo automatizzato che popola le due piattaforme di storage dei dati di HDFS e PostGIS. Mentre in ambito degli Open Data si è creata una piattaforma Open caratterizzata dall'uso di strumenti liberamente utilizzabili ed espandibili.Lo stesso progetto è distribuito sotto GNU General Public License con Linking.

Gli sviluppi futuri dell'applicativo prevedono il passaggio della parte di elaborazione distribuita sul framework YARN fornito da Hadoop, consentendo di sfruttare anche uno dei concetti principali di Hadoop: *"Moving computation is cheaper than moving data"*. Vi sarà inoltre la necessità di creare una base dati più solida e performante, utilizzabile negli aspetti d'uso reale, eliminando quindi l'uso degli applicativi esterni per la generazione delle query e per il caricamento automatizzato. Altro possibile sviluppo futuro è la ricerca di metodi di integrazione maggiormente performanti rispetto all'uso del Subprocessing e di rimozione dei vincoli di esecuzione imposti dalla piattaforma OSGeo4W per GDAL.

Ringraziamenti

Vorrei ringraziare il mio relatore, Matteo Golfarelli, per avermi dato la possibilità di partecipare a questo progetto e Tommaso Pirini, per la sua disponibilità e per il supporto ricevuto durante questo percorso.

Un immenso grazie ai miei colleghi che mi hanno accompagnato in questi anni e quelli che continueranno ad accompagnarmi nei prossimi avvenire.

Alla mia famiglia che è sempre stata disponibile a supportarmi.

Ed infine ai miei amici di sempre per aver alleviato i giorni più difficili.

Bibliografia

[1] Uwe Müller-Wilm, *Sentinel-2 MSI – Level-2A Prototype Processor Installation and User Manual* , Telespazio VEGA Deutschland GmbH , 2016.

[2] Apache, *Documentazione Ufficiale Apache Hadoop 2.7.2*, Apache, `http://hadoop.apache.org/docs/stable/`.

[3] CopernicusEU, *Copernicus Project*, `http://www.copernicus.eu/main/overview`.

[4] European Spatial Agent ESA, *European Spatial Agent ESA*, `http://www.esa.int/About_Us/Welcome_to_ESA/ESA_s_Purpose`, `https://en.wikipedia.org/wiki/European_Space_Agency`.

[5] Treccani Online, *Il Treccani - Vocabolario Online*, Roma - Istituto della Enciclopedia Italiana, 2003, `http://www.treccani.it/vocabolario/`.

[6] ESA - European Spatial Agent, *Documentazione Tecnica di Sentinel-2 Multi Spectral Instrument*, ESA - European Spatial Agent, 2016, `https://earth.esa.int/web/sentinel/technical-guides/sentinel-2-msi`.

[7] ESA - European Spatial Agent, *Documentazione Utente del Sentinel-2 Multi Spectral Instrument*, ESA - European Spatial Agent, 2016, `https://earth.esa.int/web/sentinel/user-guides/sentinel-2-msi`.

[8] Open Source Geospatial Foundation, *Documentazione Ufficiale Geospatial Data-Abstraction Library - GDAL*, Open Source Geospatial Foundation, 2016, `http://www.gdal.org/index.html`.

[9] IBM, *Performance and Capacity Implications for Big Data*, IBM, `http://www.redbooks.ibm.com/redpapers/pdfs/redp5070.pdf`.

[10] Wikipedia Community, *Critique to Big Data*, Wikipedia Community, `https://en.wikipedia.org/wiki/Big_data`.

[11] Open Knowledge International, *Open Knowledge Definition*, Open Knowledge International, https://okfn.org/opendata/.

[12] Open Knowledge International, *Open Data Definition*, Open Knowledge International, http://opendefinition.org/

[13] Regione Emilia-Romagna - MoReFarming, *POR-FESR 2014-2020 - Bando per progetti di ricerca industriale strategica rivolti agli ambiti prioritari della Strategia di Specializzazione Intelligente*, Regione Emilia-Romagna, .

[14] Open Source Geospatial Foundation, *Documentazione Ufficiale PostGIS 2.2*, Open Source Geospatial Foundation, http://postgis.net.

[15] BoundlessGeo, *Introduction to PostGIS*, BoundlessGeo, http://workshop.boundlessgeo.com/postgis-intro/introduction.html.

[16] Bernard Marr, *Big Data: What is it?*, http://www.slideshare.net/BernardMarr/140228-big-data-slide-share/16-The_Dataficationof_our_World_Activities.