Prefazione

La seguente trattazione presenta un progetto dedito allo studio dell'efficienza computazionale di cluster a basso consumo energetico adoperati per l'analisi biofisica. In particolare, la ricerca mira a dimostrare come l'utilizzo di macchine a minor dispendio energetico possano essere più convenienti e potenzialemente più potenti rispetto alle odierne macchine sfruttate nel ramo della ricerca biomedica e sanitaria.
Il processo attraverso il quale è stato possibile strutturare una tale ricerca è avvenuto concentrando gli interessi verso uno tra i sistemi più moderni di ricerca delle mutazioni genetiche causanti varie tipologie di tumori: il sistema GATK-LOD$_{\mathrm{n}}$. L'implementazione di una componente di questo metodo in un innovativa versione della nota utility Make, e quindi la creazione di un nuovo eseguibile, ha permesso una gestione più libera dei singoli passaggi del suddetto sistema.
Questo nuovo strumento, denominato Snakemake, è in grado di organizzare i diversi compiti del metodo biofisico su diverse macchine conservando il corpo unico del programma. Proprio questa caratteristica è stata sfruttata per verificare la fruibilità di insiemi di computer che collaborano come un solo apparato: i cluster.
La formazione di gruppi di computer nasce dalla necessità dei moderni organi di ricerca di potenziare le capacità computazionali, a cui deve essere anche alternato un contenimento dei costi sia economici che energetici. In questi tempi infatti, gli studi nel campo biomedico prevedono la collaborazione di professionisti in statistica dotati di computer ad alta potenza, che hanno il compito di fornire i risultati proficuamente e rapidamente. Lo sviluppo e il progresso nei vari settori ha necessitato, e necessita tuttora, di un contemporaneo aumento della potenza di computazione, il quale ha alcuni risvolti problematici. La crescita delle prestazioni dei computer ha come conseguenza di base un'inevitabile innalzamento dei costi di tali servizi e quindi una minor accessibilità alla maggioranza dei gruppi di ricerca. Questo però, non è l'unico effetto negativo poichè all'aumento della potenza segue un aumento del consumo energico, che non sè un fattore trascurabile.
Queste ragioni hanno portato, negli ultimi anni, alcuni studiosi ad interessarsi a metodi alternativi per la computazione e in questa ricerca è approfondito il tema riguardante le simulazioni su cluster a basso consumo energetico. L'idea di fondo è il poter garantire ai ricercatori un risultato in qualità di tempi almeno pari a quello ottenuto con la metodologia tradizionale. Accompagnato però, dal vantaggio di consumare minor energia e spendere una quantità di fondi inferiore.
Nel primo capitolo saranno introdotti e approfonditi gli elementi cardine del

progetto a partire dall'esposizione del metodo GATK-LOD$_n$ sia nel funzionamento che nei risultati. Successivamente sarà evidenziata la componente del metodo che è implementata nell'ambiente di sviluppo del tool Snakemake e, quindi, saranno approfondite le capacità di questo strumento. Infine, sarà specificato il significato di low power e saranno mostrate le macchine adoperate nell'analisi, per poi concludere con un accenno al funzionamento dei cluster.

Nel secondo capitolo sarà spiegato dettagliatamente il funzionamento del programma con alcuni approfondimenti sui parametri utilizzati e verranno evidenziati i passaggi per un corretto uso del sistema, dall'installazione ai dati ottenuti. In più sarà specificato quale tipo di analisi è stata compiuta su tali valori a disposizione.

Nel terzo capitolo verrano mostrati i risultati finali più rilevanti con l'aggiunta di tabelle e grafici utili ad impreziosire le analisi e ad accompagnare l'esposizione. Inoltre sarà presente anche un breve accenno ai dati non considerati proficui e a coloro che sono stati trascurati.

Per terminare sarà presente un paragrafo destinato alle conclusioni, alle considerazioni finali e agli sviluppi futuri del progetto, tutto basato sugli esiti più interessanti tratti dalle indagini svolte.

Capitolo 1

Introduzione

Questo capitolo introduce le componenti principali del progetto ed ha il compito di spiegare in maniera approfondita le operazioni svolte da esse. Per consentire una piena comprensione del sistema, tali componenti sono state raggruppate nei tre campi su cui è stato condotto il lavoro.
Il primo di questi è il campo bioinformatico e, quindi, un approfondimento del metodo GATK-LOD_n riguardante la ricerca sull'origine dei tumori attraverso nozioni di bioinformatica. Inoltre sarà sottolineata l'importanza della fisica nell'ambiente di lavoro e nelle procedure degli algoritmi.
Il secondo è il campo di sviluppo informatico, ovvero la spiegazione del programma Snakemake scelto, le cui funzionalità concendono ampie possibilità sulla gestione delle risorse offerte dalle macchine.
Il terzo è il campo dei dispositivi low power, accompagnati dalla descrizione di coloro che sono stati adoperati per la computazione.
La combinazione tra tali differenti ambiti ha quindi fornito i sufficienti strumenti per proseguire con la costruzione e il funzionamento del programma; i quali saranno affrontati nel prossimo capitolo.

1.1 La ricerca delle mutazioni genetiche

Questa sezione si occuperà di dare un'idea generale su un ramo degli studi riguardanti le mutazioni genetiche che sviluppano e alimentano i tumori.
Negli ultimi anni l'indagine sulle forme cancerogene basata sulle variazioni che avvengono nel codice genetico ha suscitato sempre più interesse e ciò ha portato allo sviluppo di un discreto numero di programmmi, software e metodi atti all'analisi del DNA. Ognuna di queste applicazioni ha come fine la determinazioni di quelle mutazioni nei geni che comportano l'insorgere delle

malattie tumorali ora conosciute. La conoscenza di una tale correlazione è di vitale importanza per la pianificazione di un piano di cura adeguato e, in altri casi, per un intervento anticipato in grado di prevenire il presentarsi della malattia.
Un altro aspetto delicato è la risposta del cancro alle cure a cui è sottoposto il paziente e che, in taluni casi, si concretizza in una forma di resistenza a questi interventi, come ad esempio la resistenza alla chemioterapia. La gestione di una tale conseguenza può essere aiutata dalla comprensione dell'origine genetica del tumore ed è possibile perciò agevolare la scelta tra i percorsi di guarigione più opportuni.
Seppur avendo lo stesso obiettivo, gli innumerevoli algoritmi utilizzati spesso si ritrovano in contrasto tra di loro e queste discrepanze sono rilevate quando si procede con il confronto dei risultati finali, che spesso non concordano o sono proprio differenti. Questi sistemi lavorano i dati sperimentali grezzi attraverso vari processi che, generalmente, sfruttano svariati algoritmi di statistica le cui radici provengono dai campi di matematica e fisica applicata. La netta differenza di prestazioni, insieme alla diverse metodologie operate, lascia ai gruppi di ricerca un arduo compito nella scelta del metodo più ideoneo da seguire.
In questo progetto il metodo considerato prende il nome di GATK-LOD_n ed è ideato dalla combinazione di due tra i tools più comuni nel panorama bioinformatico: GATK e MuTect. Prima di esporre questo algoritmo e le sue fondamenta, è necessario definire il percorso storico che ha determinato le moderne tecniche con cui è analizzato attualmente il materiale genetico e le caratteristiche generali di questi procedimenti. Inoltre, sarà sottolineato l'impatto della fisica nei campi della biologia e della medicina.

1.1.1 L'analisi del DNA

Lo studio sull'eredità genetica ha inizio con gli studi di Gregory Mendel nella seconda metà del diciannovesimo secolo e nella prima metà del secolo seguente, un serie di contributi, tra cui la fotografia della doppia elica genetica da parte di Roselind Franklin, pongono le basi alla moderna ricerca genetica. Il contributo essenziale è la determinazione della struttura del DNA a cui contribuirono James Watson e Francis Crick, la quale permise di cominciare la decifrazione del codice genetico. I tentativi di decodifica hanno subito una svolta significativa nel 1977 grazie alla tecnica di sequenziamento genetico proposta dal biochimico Frederick Sanger, conosciuta ora come metodo Sanger. Tale procedimento determina l'ordine delle basi in un filamento breve di DNA utilizzando la tecnica dell'elettroforesi e il supporto di alcuni marcatori radioattivi. Questo metodo è considerato rivoluzionario ed è stato utilizzato

come capostipite del sequenziamento del genoma umano per più di 40 anni. Nel 1984 il Department of Energy (DOE) e il National Institutes of Health (NIH) degli Stati Uniti hanno avviato un programma per lo sviluppo e la ricerca di tecnologie e metodi per il sequenziamento genetico e la mappatura del codice genetico umano: lo Human Genome Project(HGP). L'obiettivo essenziale del programma è la decodifica del DNA umano che comprende principalemente la determinazioni delle sequenze, cioè l'ordine delle basi, e la mappatura dei geni, ovvero la localizzazione dei geni per la maggior parte dei cromosomi e le connessioni con le caratteristiche fisiche e germinali. Oltre a ciò, il progetto ha cercato di analizzare anche i genomi di esseri viventi valutati più semplici, come ad esempio le mosche.
Molte tecniche per la trattazione del materiale genetico hanno partecipato alla ricerca, ma il primo procedimento attuato dall'HGP fu il BAC, Bacterial Artificial Chromosome, che sfruttava la clonazione apportata dai batteri per ottenere un numero elevato di fili. Questi subivano un ulteriore frammentazione per poi essere sequenziati con il metodo Sanger ed infine riuniti per formare l'unico filamento di DNA.
Dopo oltre un decennio, nell'Aprile del 2003, l'HGP si è concluso con successo essendo stato decodificato il 99% del genoma umano e visto che gli studi erano stati allargati prematuramente ad altri interessi, soprattutto nelle ricerche sulle malattie. Da quel momento in poi, il termine comune affiliato alle nuove ricerche sul genoma umano è Next-Generation Sequencing(NGS), di cui fanno parte gli algoritmi coinvolti in questa presentazione.

NGS

Il procedimento standard per il sequenziamento genetico durante lo Human Genome Project è il metodo Sanger e, pur essendo efficace, le richieste iniziali sono state conseguite in più di 10 anni di lavoro. La necessità di garantire una maggior rapidità, e anche di ridurre le spese, ha aperto un altro capitolo nella ricerca sul genoma umano: il Next-Generation Sequecing. Questa nuova generazione di tecnologie aspira a dotare i ricercatori di strumenti più rapidi e meno costosi, che sfruttano la parallelizzazione di processi su scale microscopiche.
Un fattore importante che supporta lo sviluppo di tecnologie più veloci è l'esistenza del genoma di riferimento prodotto da HGP, che permette il confronto di certe regioni senza quindi dover estrarre l'intero codice genetico. Un ulteriore contributo è dato dal contemporaneo progresso delle aree che cooperano con la biologia, come ad esempio le tecniche di computazione.
Le numerose applicazioni sviluppate all'interno del NGS devono trovare l'idoneo compromesso tra l'accuratezza dei risultati e la diminuzione delle spese

temporali ed economici. L'uso di questo strumenti deve essere più conveniente del metodo Sanger, il quale rimane artefice di un eccellente connubio tra qualità e quantità di letture(read-lenght). Infatti, nello studio di sequenze a piccola scala la tecnologia di Sanger resta tuttora uno tra i contributi più affidabili, mentre è atteso che le applicazioni del NGS possano vantare in futuro le migliori funzionalità su letture ad ordini superiori.
Pur adoperando tipi di tecnologia diversi, il funzionamento di un generale algoritmo del NGS prevede sempre l'allineamento delle letture con un riferimento e l'investigazione sulle possibili variazioni. La scelte di quale tra le diverse opzioni disponibili è facilitata dal confronto tra i diversi parametri risultati dal processo, i quali includono la qualità dei risultati, le tempistiche previste e altre caratteristiche, come l'assemblaggio e gli allineamenti.
Negli ultimi anni si sono aggiunti alle ricerche del NGS due software chiamati GATK e MuTect, i quali hanno condotto allo sviluppo dell'algoritmo coinvolto nella ricerca presentatata in questo testo: il GATK-LOD_n.

1.1.2 Il metodo GATK-LOD_n e le sue radici

L'ideazione di questo algoritmo é dovuta ad un gruppo di ricercatori del Dipartimento di Fisica e Astronomia dell'Università di Bologna nell'ambito dello studio sulla scoperta di polimorfismi somatici del singolo nucleotide nel sequenziamento dell'esoma. Precisamente, questo genere di polimorfismo è denonimato con la sigla SNP, single nucleotide polymorphism, e indica quelle variazioni nei singoli nucleotidi che si verificano con frequenza significativa in una specifica posizione del genoma. In particolare, l'esoma comprende quelle regioni del genoma in cui sono codificate le istruzioni per l'RNA e per la sintesi delle proteine. Le mutazioni somatiche inoltre, hanno un ruolo chiave nella progressione della malattia e nella resistenza alla chemioterapia.
L'interesse nel proporre questo metodo nasce dal desiderio di poter predisporre di uno strumento che non si aggiunga al gruppo di software già esistenti bensì che ottimizzi e potenzi alcuni tra questi. Perciò, il team di studiosi ha considerato due applicazioni standard, GATK e MuTect, e li ha composti in modo da migliorare i prodotti finali di entrambi. Infatti alla completa esecuzione di GATK è stato applicato una componente di MuTect, ovvero un classificatore Bayesiano conosciuto come LOD_n il cui scopo è verificare un ulteriore volta i risultati ottenuti. I passaggi previsti dall'algoritmo sono vari e di seguito saranno esposti coloro ritenuti più rilevanti per una spiegazione sufficientemente piena del medesimo.
Dopo aver raccolto i campioni normali e quelli per alcune specie di tumori attraverso specifiche metodologie sperimentali, essi sono stati sottoposti ad un controllo di qualità tale da rimuovere le letture considerate a bassa confi-

denza, cioè non soddisfacenti una soglia minima predefinita. A questo punto, sono state applicati in successione i tools BWA-MEM e Picard, dove il primo allinea le reads e il secondo le ordina, indicizza e in più ne marca i duplicati. Una volta completati sugli allineamenti una fase di riallineamento locale e di ricalibrazione sulla qualità delle letture, possibili grazie all'utilizzo di alcuni strumenti del Genome Analysis Toolkit, infine sono stati eseguiti GATK e MuTect per la ricerca delle varianti sui singoli nucleotidi(SNV).
La differenza procedurale tra queste due applicazioni è che, mentre MuTect ritrova le mutazioni contemporaneamente tra i campioni normali e tumorali; GATK le chiama indipendentemente. Un'osservazione da sottolineare è che i due metodi scovano pure diverse varianti che non sono condivise da entrambi, indicando la natura incompleta dei sistemi utilizzati. Un ulteriore distinzione tra GATK e MuTect è data dal tipo di risultati raccolti poichè, se il primo è dotato di una maggiore sensibilità alle mutazioni, dalle 3 alle 20 volte superiore al secondo; quest'ultimo possiede una maggiore specificità degli SNVs.
Avendo GATK un elevato numero di falsi positivi nella chiamata alle varianti, è stato aggiunto in fondo all'algoritmo il classificatore Bayesiano di MuTect per cercare di ridurre questo errore. Il compito del LOD_n consiste nel calcolare il rapporto tra due eventi probabilistici. Il primo è che la mutazioni nel campione normale è dovuta a rumori di fondi e quindi in realtà non esiste. La seconda invece, considera il caso in cui la mutazione esiste nel campione normale ed è dovuta ad una variante germinativa eterozigote. A questo punto, se il rapporto tra le probabilità (il Log Odds, da cui LOD) eccede un valore di soglia fissato, il classificatore definisce la variante come somatica.
Il gruppo di studiosi ha confrontato i prodotti finali dei tre algoritmi e ha ricavato che l'uso di GATK-LOD_n riduce notevolmente il numero di chiamate degli SNVs di GATK, mantenendo una sensibilità nettamente superiore a MuTect. Questa riduzione è dovuta all'eliminazione di un sostanziale numero di falsi positivi, ovviamente dipendenti dalla tipologia di tumore. Perciò, il vero successo è il raggiungimento dello scopo prefissato, che consisteva nel dotarsi di uno strumento più performante di GATK e MuTect. Il miglioramento non si limita solo alla filtrazione dei falsi positivi ma anche al mantenimento della sensibilità di GATK e all'aumento della specificità. Infatti, GATK-LOD_n ha presentato frequenze di validità superiori a GATK e un miglior PPV, Positive Predictive Value, di GATK su differenti range di VAF, Variant Allelic Frequency. Le metodologie adoperate per confrontare la sensibilità e la specificità non sono state ritenuti utili allo scopo della presentazione e per questo non verranno trattate.
A posteriori delle verifiche sperimentali, GATK-LOD_n si è rivelato uno stru-

mento utile ad allargare le capacità di GATK e a scovare varianti non trovate da MuTect senza dover rinunciare alla specificità e sensibilità.
Visti i risultati positivi ricavati dall'algoritmo, il team di ricercatori è fiducioso che un metodo di questo tipo possa aiutare a definire con maggior dettaglio le mutazioni somatiche di genomi cancerogeni, favorendo le valutazioni mediche e gli approcci sui percorsi di cura.
In modo da fornire una conoscenza più approndita delle operazioni che avvengono nel sequenziamento, saranno esposti brevemente i concetti fondamentali alla base delle applicazioni utilizzate nel metodo GATK-LOD_n: GATK e MuTect.

GATK

Il Genome Analysis Toolkit è un framework di programmazione creato da un gruppo di ricercatori di Boston, Massacchussets, assieme al Broad Institute di Harvard e l'MIT in Cambridge, Massacchussets, per facilitare lo sviluppo di programmi che analizzino l'enorme mole generati dal NGS, Next Generation DNA Sequencing. Infatti, l'utilizzo di questo sistema permette di sviluppare tools più solidi e performanti per il sequenziamento genetico.
I motivi che hanno condotto alla creazione di GATK sono da ricercare nella mancanza di strumenti flessibili e sofisticati dediti alla manipolazione dei dati di sequenziamento in maniera programmatica. Infatti, nonostante l'esistenza di molti software che cooperano con allineatori del tipo di BWA, questi concedono alte prestazioni solo nel loro specifico campo lasciando irrisolte questioni di carattere più specifico. Questo deficit e l'emergere di un formato specifico per i prodotti del sequenziamento (SAM), hanno dato l'opportunità di ideare un software per la semplificazione delle analisi sui data sets.
L'architettura di base per GATK è il MapReduce, il cui funzionamento implica la separazione delle computazioni in due passaggi. Nel primo, l'intero problema è suddiviso in tanti elementi discreti indipendenti, i quali sono correlati alla funzione Map; nel secondo step, l'operatore Reduce riunisce gli esiti di Map in un unico risultato finale. Siccome solo in certi casi, come la ricerca degli SNPs, vi è un adattamento naturale del sistema, GATK è costruito su traversals e walkers. Le traversals sono schemi che provvedono alla preparazione e divisione dei dati; mentre le walkers consistono nei differenti moduli di analisi che computano i dati provenienti dalle prime. GATK ha un ridotto numero di diversi traversali che però soddisfa le esigenze della maggior parte della comunità di ricerca. I due trasversal standard sono il "by each sequencer read" e il "by every read covering single base position in a genome", il cui uso è sfruttato per operazioni standard come la chiamata agli SNPs. Il meccanismo di questi schemi non è riportato poichè non è stato

ritenuto necessario per la comprensione di questo scritto.
Uno dei punti di forza dell'algoritmo è la capacità di gestire l'enorme quantità di materiale ricavato dal sequenziamento. In particolare, GATK divide il tutto in pezzi chiamati "shard" che, al contrario della maggior parte dei sistemi di suddivisione, sono di una dimensione del multikilobase. In questo modo non sono limitate le capacità della memoria e le prestazioni della parallelizzazione. Questi shards contengono tutte le informazioni della regione genomica associata e sono trasmesse al trasversal prescelto.
Altre caratteristiche di GATK sono la possibilità di selezionare solo certe regioni del genoma, la parallelizzazione delle azioni da svolgere, l'organizzazione dei files di input grazie ad un'operazione di merging e la presenza di un walker relativo alla depth of coverage(DoC).
Il metodo stima il genotipo più probabile attraverso un semplice operatore Bayesiano, che è interpretato sia come punto di partenza per sviluppare nuovi classificatori; che per mettere in luce le capacità di parallelizzazione e di ottimizzazione della memoria disposte da GATK. E' importante sottolineare che è proprio la semplicità dell'operatore la causa di molti falsi positivi chiamati.
Il Genome Analysis Toolkit è quindi un framework che, grazie al suo nuovo approccio ai big data e alla piena libertà di sviluppo, fornisce strumenti importanti per la creazione di algoritmi più specifici come il GATK-LOD_n.

MuTect

Il metodo MuTect è un algoritmo per la rilevazione delle mutazioni genomiche, che è stato creato per superare le scarse prestazioni dei meccanismi fino ad allora presenti. Infatti, quest'ultimi fornivano livelli di sensibilità e specificità considerati insoddisfacenti per una sufficiente comprensione delle anomalie.
Il sistema focalizza gli sforzi soprattutto sull'individuazione delle varianti a bassa frazione allelica, ovvero quelle regioni del DNA che originano e nutrono il tumore. Queste frazioni sono tanto importanti quanto difficili da scovare poichè, mentre il meccanismo al loro interno determina il tipo di alterazione genomica, sia le regioni in cui si manifestano che la frequenza di occorrenza sono basse. La conoscenza di queste strutture specifiche favorisce lo studio sulle evoluzioni delle forme cancerogene e aiuta a proporre terapie di cura più affidabili.
L'esistenza di metodi a scarsa sensibilità e specificità ha quindi indotto un team di ricercatori dell'università di Boston, Massachussetts, assieme al Broad Insitute di Harvard e all'MIT in Cambridge, Massacchussets, a sviluppare il tool MuTect. Questo strumento si è dimostrato molto più sensibile

e specifico nella scoperta degli eventi a bassa frequenza allelica, mantenendo alte prestazioni di specificità anche a frequenza superiori.
Nessuno dei modelli precedenti sopporta tutti gli errori dei processi di sequenziamento ed è per questo che MuTect sfrutta due approcci di benchmarking per migliorare la performance: il downsampling e i 'virtual tumors'. Il primo misura la sensibilità con cui le mutazioni vengono chiamate, grazie a subset di dati con mutazioni già riconosciute. Questo approccio è però limitato da alcuni aspetti che comprendono il basso numero di eventi verificati, la sovrastima della sensibilità e l'impossibilità alla misurazione della specificità. A causa dei limiti del downsampling è presente pure un secondo approccio, 'virtual tumors', che genera dei tumori virtuali conosciuti in ogni dettaglio. Infatti, i due metodi sono complementari e mentre il primo usa dati reali ma è limitato, il secondo è libero ma consuma materiale virtuale sufficientemente adatto. La sintesi dei due approcci permette di ricavare valori più veritieri della sensibilità, misurare la specificità e colmare la maggior parte delle lacune derivate dal downsampling.
Previo allineamento ed esecuzione dei processi standard preanalisi, MuTect riceve i dati sequenziati dei campioni normali e cancerogeni per poi eseguire quattro principali operazioni: rimozione di dati a bassa qualità, ricerca delle varianti, filtraggio per i falsi positivi e classificazione delle varianti. La ricerca delle varianti consiste nell'applicazione di un primo operatore bayesiano, detto $\mathrm{LOD_T}$, che adopera il rapporto tra due modelli per determinare se è presente un variante. Siccome il calcolo è condizionato da errori di sequenziamento, prima di ricavare la probabilità del variante è necessario applicare una serie di filtri che ne elimina la maggior parte, evitando la sottostima dei falsi positivi. Successivamente si utilizza il secondo classificatore bayesiano $\mathrm{LOD_n}$, implementato nel metodo GATK-$\mathrm{LOD_n}$, per definire se il variante è somatico, germinale o indeterminato.
Gli esperimenti di verifica sulla sensibilità hanno evidenziato come MuTect sia uno strumento ad alto rilevamento soprattutto nelle mutazioni a frazioni alleliche basse. Riguardo alla specificità sono due le fonti principali di falsi positivi: l'eccessiva chiamata a varianti, dovuta ad errori di sequenziamento, e la scarsa individuazione di eventi germinali, causato dalle insufficienti letture sul campione normale. La gestione di tali errori è risolta con il miglior compromesso verificato tra sensibilità e specificità, che risulta nella scelta di mantenere alta la seconda a discapito della prima.
Complessivamente MuTect è un metodo che valorizza il compromesso tra il grado di rilevazione delle varianti e la loro corretta classificazione, producendo risultati affidabili. In aggiunta, riporta sostanziali miglioramenti sulle analisi delle mutazioni a bassa frequenza allelica, la cui importanza è essenziale per le future ricerche biomediche.

1.1.3 Il ruolo della fisica nella ricerca biomedica

In modo da capire chiaramente quale sia il ruolo di una branca scientifica come la fisica nel campo della ricerca in biologia e in medicina, è necessario innanzitutto distinguere il ruolo della fisica come materia e dei fisici. Infatti, nella prima metà del Novecento, dopo che l'introduzione della relatività di Einstein e la nascita della meccanica quantistica avevano ribaltato il pensiero della comunita scientifica, numerosi fisici si interessarono a problemi di biologia. Il contributo fornito ad una così diversa area era sia di tipo matematico, dove la necessità di utilizzare equazioni e formule continuava ad aumentare nel corso degli anni, che di tipo teorica, ove l'uso di modelli già presenti nella fisica erano utili ad una comprensione più approfondita dei quesiti di biologia. Pur coltivando interessi all'apparenza lontani dai propri, i fisici non snaturavano il fine della fisica, dato che l'intento restava la ricerca ai principi primi della natura. Il libro "What is life?" del fisico austriaco Erwin Schroedinger promosse il ruolo della fisica negli studi sull'ereditarietà a tal punto che gli scienziati Watson e Crick, che svelarono la struttura del DNA, lo accreditarono nelle loro pubblicazioni.
Anche se alcuni tra i fondatori della biologia molecolare erano fisici, il ruolo della materia col passare del tempo è passato sempre più inosservato pur rimanendo viva la partecipazione dei fisici. Questa conseguenza è basata prevalentemente sul fatto che mentre la fisica è governata da concetti teorici, la biologia su eventi empirici e tale discrepanza ha ipotizzato un allontanamento tra i due campi. Nonostante ciò, la biologia molecolare ha continuato a beneficiare della fisica a causa dell'approccio diverso, che propone nuovi aspetti, e alla confidenza riguardo alla modellizzazione dei sistemi complessi. Al passare del tempo e con il crescere delle ricerche, soprattutto nel campo biomedico, la presenza della fisica ha acquistato importanza sia nel ramo sperimentale che in quello teorico. Nel primo, difatti, per anni le tecniche della cristallografia a raggi X e della risonanza magnetica sono stati essenziali per i biologi molecolari, e, tuttora, gli innovativi programmi della NGS concentrano gli sforzi su scovare quelle differenze fisiche che permettono di stabilire più velocemente i nucleotidi. Nel secondo, invece, la fisica coopera con i bioinformatici sia nell'individuazione di modelli efficaci con cui affrontare i big data provenienti dai laboratori, che per l'analisi delle sequenze di DNA, sfruttando metodi provenienti esattamente dalla fisica statistica.
La richezza della fisica nel saper gestire i sistemi complessi, l'elasticità nell'ampliare le proprie conoscenze a differenti campi di studio, insieme ad un approccio critico ma propositivo, rende quest'area della scienza un ingrendiente fondamentale per il proseguimento della ricerca biomedica, oltre che di qualsiasi altro settore di sviluppo.

1.2 Lo strumento di sviluppo: Snakemake

Snakemake è un sistema di gestione del flusso di lavoro che semplifica l'esecuzione di algoritmi particolarmente complessi grazie all'utilizzo di un ambiente di sviluppo nitido ed intuitivo. Inoltre, questo software è specializzato nella scalabilità da lavori su singoli core fino all'uso di cluster e la transizione tra gli apparecchi non implica modifiche al procedimento del sistema.
Questo programma è basato sul linguaggio di programmazione Python e la sua formazione è fortemente influenzata dal noto tool Make del sistema operativo Linux. Visto il largo uso di Python e l'affermata conoscenza di Make nella comunità di sviluppatori, queste due caratteristiche rendono tale strumento ancora più affidabile e prossimo agli interessi dei ricercatori.
Il contenitore del codice Python in Snakemake è chiamato di default *Snakefile* e l'ordine di esecuzione predefinito da linea di comando è:

```
$ snakemake
```

Il sistema è strutturato, come Make, da un insieme di regole che rappresentano i compiti da svolgere nell'algoritmo, dove ognuna di esse contiene le tre informazioni fondamentali input, output e azione.

```
rule 'nome':
        input:

        output:

        shell/run/script:
```

Come si può vedere dal codice riportato, l'operazione avviene etichettando la regola con un certo 'nome' e inserendone all'interno le keyword *input*, *output* ed una tra *shell*, *run* e *script*. Rispettivamente le tre chiavi implicano l'utilizzo di comandi da terminale, l'esecuzione di un codice Python e l'avvio di uno script esterno. La dichiarazione dei file di input e di output esprime le condizioni iniziali per la regola e il risultato atteso, così permettendo al programma di riconoscere le relative dipendenze e stabilire l'ordine di successione dei singoli lavori. In aggiunta, è possibile creare un diagramma che mostri la sequenza delle regole, grazie al comando seguente.

```
$ snakemake --dag | dot -Tpdf > dag.pdf
```

Per ottenere questa illustrazione è però indispensabile l'installazione del pacchetto *graphviz* tramite *Conda*. Due notevoli proprietà di Snakemake sono la portabilità e un innovativo meccanismo di inferenza. La prima manifesta le poche dipendenze di installazione, dato che è in generale sufficiente dotarsi di Python; mentre la seconda rappresenta un moderno supporto all'inferenza

per nome che si compie grazie a wildcards nominate nelle regole.
Una peculiarietà particolarmente utile di Snakemake è la capacità di riprendere l'esecuzione di una pipeline interrotta, esattamente dalla regola in cui si trovava al momento dell'interruzione. Ciò è garantito dal fatto che il sistema attiva una regola solo se è assente il relativo output e, quindi, solo coloro che non sono state ultimate , o rimaste intoccate, prendono parte alla nuova esecuzione senza ripartire dall'inizio. Chiaramente, questo approccio è modificabile richiendo esplicitamente che alla richiesta di una nuova istanza vengano sovrascritti i file di output e quindi acconsentita una diversa realizzazione.
Nella pipeline creata nell'ambito di questa ricerca, sono stati implementati altri attributi forniti da Snakemake. Infatti, eccetto le chiavi di base, esistono diverse funzionalità che arrischiscono l'impostazione e la realizzazione delle regole, tra cui *params*, *threads*, *benchmark* e *conda*. I primi due contengono rispettivamente i parametri indispensabili introdotti nell'azione e il numero di threads su cui l'azione può essere eseguita. Quest'ultima proprietà richiama ad uno tra gli aspetti più importanti che hanno determinato la scelta di Snakemake, che è l'eccezionale capacità di gestire l'esecuzione della pipeline sulle risorse tecniche dei dispositivi adoperati. Le istruzioni sul numero di core da utilizzare, il numero di threads e la quantità di memoria da mettere a disposizione consentono di ricavare le configurazioni più efficienti, il tutto a beneficio dello sviluppo di metodi più potenti e ottimizzati.
A differenza delle prime due direttive, *benchmark* e *conda* richiedono una trattatazione più ampia.
In seguito alla spiegazioni di questi, è necessario dedicare un breve accenno ai file di configurazione spesso affiancati agli Snakefile.

benchmark

Quando è utilizzata la direttiva *benchmark* in una regola, Snakemake trascrive su un file di testo i dettagli tecnici dell'operazione svolta. Per l'invocazione del benchmark è sufficiente aggiungere nella regola:

```
benchmark:
        "path/to/directory/nome_del_file.txt"
```

A questo punto, il file creato sarà strutturato come una tabella con le seguenti colonne.

```
s h:m:s max_rss max_vms max_uss max_pss io_in io_out mean_load
```

Sotto la prima e la seconda voce sarà riportato il tempo impiegato dall'apparecchio per completare la regola rispettivamente in secondi e in ore, minuti

e secondi.
Dalla terza alla sesta sono mostrati i particolari sull'uso della memoria. In particolare:

- max_rss è la massima memoria fisica non scambiata, che il processo usa(Resident Set Size);
- max_vms è la massima quantità totale di memoria virtuale utilizzata(Virtual Memory Size);
- max_uss è il massimo di memoria affidata unicamente al singolo lavoro, che esso impiega(Unique Set Size);
- max_pss è il massimo della quantità condivisa tra tutti i processi, che la regola sfrutta(Proportional Set Size).

Riguardo agli ultimi tre, io_in e io_out identificano le caratteristiche di input e output del processo; mentre mean_load descrive il carico medio sulla CPU.

Conda

Conda è una piattaforma per la gestione di svariati pacchetti ed è un pratico amministratore degli ambienti di elaborazione. La cooperazione tra Snakemake e Conda, che avviene, come mostrato nel codice sottostante, inserendo la direttiva *conda* nel dominio, favorisce un uso più flessibile degli ambienti.

```
conda:
        "path/to/directory/config_file.yaml"
```

Difatti, prima che lo Snakefile sia eseguito, Snakemake riconosce la voce *conda*itconda nella regola e richiama Conda per la creazione dell'ambiente su cui essa sarà completata. Le informazioni sull'ambiente da formare sono procurate da un file di configurazione indicato nel dominio ed è questo che consente a Conda di generare l'ambiente e dotarlo dei requisiti richiesti. Una volta che Conda ha terminato la creazione, l'ambiente è attivato e comincia l'esecuzione. Chiaramente, questa fase di creazione non avviene ad ogni istanza dello stesso Snakefile, poichè se l'ambiente è già presente, per Conda è sufficiente attivarlo.
Una tale opzione non costringe altri utilizzatori ad impostare l'ambiente principale come voluto dallo Snakefile, dato che ne sarà creato un apposito per la regola, e ne alleggerisce l'utilizzo. In aggiunta, questo meccanismo rende plastica la realizzazione del codice, vista la possibilità di dotare ogni regola di un proprio ambiente.

1.2.1 I file di configurazione in Snakemake

I file di configurazione sono oggetti, solitamente in formato yaml, dedicati alle istruzioni sui parametri contenuti nei codici principali. Il ruolo di tali file è stabilire il valore delle chiavi che rappresentano i parametri. Ad esempio, nel caso di Snakemake, un tipico file di configurazione ha forma: `chiave: 'valore'`. Il valore può essere un numero, una parola, un percorso o un file, ed esso rimane costante se non modificato direttamente nel codice sorgente o da linea di comando.
Nell'ambito di Snakemake, il file di configurazione deve essere citato inizialmente con la linea `configfile:'config_file.yaml'` e l'associazione tra chiavi e parametri è conseguita inizializzando una variabile secondo la seguente modalità.

```
parametro = configfile['chiave']
```

I valori dei parametri possono essere modificati da terminale nel comando di avvio dello Snakefile, grazie all'argomento *–config* seguito da una nuova inizializzazione, ad esempio `chiave='nuovo_valore'`.
Il file di configurazione che si indica nella direttiva *conda* ha, invece, una composizione differente, come mostrato nel codice riportato.

```
channels:
  - esempio_canale
dependencies:
 -   esempio_dipendenza
```

Il dominio *channels* determina su quale canale Conda deve lavorare, mentre *dependencies* specifica quali pacchetti devono essere installati nell'ambiente. In questo modo, dopo che Conda è chiamato da Snakemake per l'attivazione del particolare ambiente, esso controlla se ne è già presente uno con tali proprietà e, se esistente, lo attiva o, al contrario, prima lo istanzia.
Il contenuto dei file di configurazione è, in conclusione, determinante per il corretto, oltre che miglior, funzionamento del sistema.

1.3 I dispositivi low power

I gruppi di ricerca che insistono nel proporre metodi avanzati per l'analisi dei big data in campo biomedico, non si limitano ad ottimizzare la sola componente software, bensì affrontano pure la scelta del tipo di componente hardware da adoperare. Lo sviluppo di applicazioni sempre più raffinate ha richiesto infatti, nel corso del tempo, un progresso tecnologico che procedesse di pari passo e che permettesse di soddisfare in maniera esaustiva le richieste

dei ricercatori. Normalmente, i macchinari prediletti sono quelli più potenti e performanti che, pur essendo abili nel terminare i lavori previsti, hanno il difetto di consumare massicce quantità di energia e di raggiungere costi molto elevati. Questi due aspetti negativi hanno spinto alcuni team di studiosi ad interessarsi a differenti risorse computazionali, tra cui quella relativa ai dispositivi low power.
I macchinari low power sono strumenti che mirano ad abbassare il consumo energetico e soprattutto ad incrementare l'efficienza. Il dispendio di energia di uno strumento elettrico avviene, in particolare, alla conversione da corrente analogica (AC) a corrente digitale (DC): questo passaggio implica un'innalzamento della temperatura che causa lo sprigionamento di calore nell'ambiente circostante, da cui si ha una perdita di lavoro. Tale effetto equivale ad uno spreco di energia ed esso si riproduce in ogni macchinario elettrico. Il contenimento del calore, e quindi in generale una diminuzione dello spreco energetico, consente non solo una diminuzione dei costi per il sostentamento dell'apparecchio, che può penalizzare grandi aziende, ma anche il miglioramento della resa del dispositivo. A riprova di ciò, una società come Google è passata da un'iniziale efficienza del 60% ad una attuale superiore al 90%.
I dispositivi low power, però, non sono una vera novità nel campo tecnologico, dato che alcuni oggetti, come gli orologi, possono essere già considerati tali. La vera innovazione è l'impiego di strumenti di questo calibro nei processi computazionali sui big data. Ovviamente, la differenza di performance tra gli apparecchi low power e i potenti macchinari utilizzati giornalmente è abissale. Nonostante ciò, la cooperazione fra diversi dispositivi di tale genere, il cluster, potrebbe ambire a sostituire gli apparecchi moderni. Questo testo descrive proprio una ricerca dedita alla verifica di una questione così rilevanti, che considera inizialmente la computazione non sul cluster ma sulle singole macchine: i nodi.
I dispositivi coinvolti in questi studio sono stati scelti con caratteristiche tecniche differenti, in modo da evidenziare fra taluni una crescita delle prestazioni. In accordo con il centro nazionale per la ricerca e lo sviluppo sulle tecnologie per l'informazione e la comunicazione (CNAF) dell'Istituto Nazionale di Fisica Nucleare, avente sede a Bologna, sono state utilizzate alcuni server appartenti al bastion dell'INFN. I dettagli sui nodi dei cluster utilizzati per la computazioni sono esposti nelle tabelle sottostante.

$Hostname$	$Hostname(10Gb/s)$	ISA	$Microarchitecture(Platform)/litho$
$xeond$	$xeond$	$x86_64$	$Broadwell/14nm$
$avoton$	$avoton$	$x86_64$	$Silvermont(Avoton)/22nm$
$n3700-00,2$	$-$	$x86_64$	$Airmont(Braswell)/14nm$

Tabella 1.1:

CPU	$Freq(GHz)$	$Cores$	$Cache(MB)$
Xeon D-1540	2.0(2.60)	8(16)	12
Atom C2750	2.40(2.60)	8	4
Pentium N3700	1.60(2.40)	4	2

Tabella 1.2:

$Memory(GB)$	$Ubuntu$	TDP
16	16.04.2	$45W$
16	16.04.1	$20W$
8	16.04.2	$6W$

Tabella 1.3:

Capitolo 2

Procedimento

Questo capitolo ha lo scopo di presentare il procedimento seguito per valutare l'ottimizzazione di una pipeline per l'analisi computazionale biofisica su apparecchi a basso consumo energetico.
In primo luogo sarà spiegato il corpo creato per l'elaborazione, che implementa una componente del metodo GATK-LOD$_{\text{n}}$, e saranno approfondite le singole componenti.
A seguire, saranno specificate quali computazioni sono state svolte, spiegando i motivi che hanno spinto a considerare talune, e il tipo di operazioni statistiche effettuate per modellare il prodotto delle analisi.
Nel complesso questo capitolo crea i presupposti per argomentare in maniera critica ed esaustiva i risultati finali che saranno mostrati nell'ultimo capitolo.

2.1 Il corpo delle analisi

Per assicurare un'esposizione adeguata del funzionamento del sistema, è opportuno suddividere il procedimento in fasi: installazione, esecuzione e configurazione.
La prima di queste, la fase di installazione, evidenzia quali sono i fattori che permettono all'esecuzione di avvenire senza problemi.
In seguito, con la fase di esecuzione, è chiarita la struttura fondamentale del procedimento, ovvero la parte relativa all'algoritmo genetico, ed è definita la successione dei diversi passaggi: dalle estrazioni dei subset di DNA, fino alla produzione e semplicazione dei dati generati dal metodo.
Infine, sono esposte, nella fase di configurazione, le modalità di impostazione del programma, così da garantire una corretta gestione dei parametri e delle istruzioni da trasmettere ad esso.

2.1.1 Installazione

Il passaggio iniziale consiste nel predisporre l'ambiente di lavoro soddisfacendo i requisiti indispensabili per una valida esecuzione del sistema.
Le due prerogative più importanti coinvolgono l'installazione di *Conda*, che è inevitabile per l'attivazione degli ambienti, e ovviamente quella di *Snakemake* per l'avvio del programma.
In seguito, sono richieste alcune librerie per Python, pandas e matplotlib, che sono necessarie per una fase di semplificazione dei benchmark e per le future analisi statistiche. In più, pur non avendo uno spessore di primo piano, sono essenziali i tool PyYAML e psutil, che colmano alcune lacune dei contenuti principali installati.
Dopo aver superato tali punti, è fondamentale equipaggiarsi del genoma umano di riferimento e dei dati genetici che si desiderano sequenziare.
In relazione al progetto ivi presentato, il genoma umano di riferimento è stato ottenuto via web dal sito ufficiale del IGSR(International Genome Sample Resource); mentre il campione di DNA esaminato è stato condiviso dall'Azienda Ospedaliero-Universitaria Sant'Orsola-Malpighi(Bologna, Italia), in accordo con i canoni dettati dalla dichiarazione di Helsinki.
Sempre nell'ambito di questo progetto, è stato scritto uno script apposito, denominato *installer.sh*, per la fase preliminare appena esposta, in modo da rendere questo procedimento compatto e più rapido. Questo eseguibile può risultare utile per coloro che sono già in possesso dei requisiti ma che preferiscono evitare di mischiare diversi ambienti di lavoro. Infatti, è importante specificare che l'installer aggiunge al bash script *.bashrc* il percorso della directory Miniconda prodotta nell'installazione, così da indurre l'utilizzo di quei tool, tra cui *Conda* e *Snakemake*, inclusi in tale cartella.
Conclusi gli interventi preparatori, può essere avviata l'esecuzione del procedimento senza dover curare altri aspetti.

2.1.2 Esecuzione

La fase esecutiva comprende una serie di processi che si possono raggruppare in tre macro sezioni ben definite. La prima di esse descrive il processo di estrazione dei sottogruppi, o subset, di sequenze di DNA che si sceglie analizzare. La seconda contiene il fulcro del procedimento, quindi illustra il tipo di operazioni svolte dall'algoritmo genetico e dichiara quali sono i prodotti attesi. La terza racchiude un'operazione di riordinamento e di semplificazione di tali ultimi prodotti per agevolare le successive analisi statistiche.
E' importante sottolineare che per favci

Estrazione

Gli oggetti delle analisi sono i subset genetici che si sceglie di estrarre dall'intero genoma del soggetto e per questo è lecito accennare a come avviene l'estrazione.
Il genoma del soggetto è solitamente contenuto in un file in formato di testo *fastq* che contiene le sequenze di nucleotidi rilevate durante le analisi in laboratorio. Spesso però, la rilevazione non è singola ma molteplice e, come nel caso di questo progetto, i dati del paziente sono suddivisi in due file fastq accoppiati.
Brevemente, ogni singola lettura contenuta nel fastq è descritta da quattro linee aventi i seguenti ruoli: la prima linea marca la sequenza, la seconda mostra la sequenza, la terza identifica il punteggio di qualità e la quarta riporta tale punteggio.
Siccome il numero di letture contenute è enorme, ed essendo che ad ognuna corrispondono quattro linee, la dimensione del file fastq è di conseguenza molto grande; per questo risulta funzionale dotarsi di un meccanismo per l'estrazione di sottogruppi.
All'interno del progetto è stato scritto uno script in Python(*split.py*) che svolgesse l'operazione appena presentata e per il quale, senza entrare in dettagli tecnici, è sufficiente trasmettere da linea di comando gli estremi della sezione che si vuole ricavare; come indicato di seguito.

```
$ python split.py {inizio} {fine}
```

Grazie all'inserimento degli estremi, il programma calcola il numero di reads desiderate, seleziona le linee corrispondenti(il quadruplo del numero di letture) e le trascrive in un nuovo file fastq.
Ai fini del progetto, la disposizione di sottogruppi sia con diversi range che in diverse regioni consente di analizzare nei particolari le proprietà di scalabilità delle computazioni.

Algoritmo genetico

Una volta creati i subsets, può avere inizio l'applicazione dell'algoritmo genetico, il quale è interamento contenuto nell'apposito Snakefile. E' importante notare che la procedura seguita è ripresa dal metodo GATK-LOD_n solamente in alcuni passaggi, come specificato fra non molto. Tali passaggi, in particolare, sono concretizzati nello Snakefile da regole che si susseguono l'un l'altra in base alle dipendenze di ciascuna(**Sezione Snakemake Cap1**).
I primi step consistono nell'indicizzazione del genoma umano di riferimento per i software che partecipano al sequenziamento. Questi coinvolgono il software di allineamento BWA, il tool di manipolazione Picard, il gestore di

strumenti Samtools e i programmi affini a GATK. Per le prime tre applicazioni, questa operazione è portata avanti da una specifica opzione delle stesse, mentre per l'ultimo essa è inclusa già nell'uso di Samtools.
Terminate le indicizzazioni, è sfruttato BWA MEM per la mappatura del DNA del soggetto sul genoma di riferimento che, inoltre, produce un file di etichetta, in formato *SAM*, che è sottoposto ad un riordinamento da parte Picard con l'assistenza della modalità SortSam.
Dopo essersi procurati i file `sorted_bam`, è eseguito il comando MarkDuplicates di Picard per identificare le letture doppie ed è generato un nuovo file `dedup_bam`. Elaborando quest'ultimo, la regola successiva crea un indice(in un file *bai*) per il file *bam* in modo da velocizzare l'analisi dei dati nel *bam*; ciò è realizzato da un'altra funzionalità di Picard chiamata BuildBamIndex.
Per concludere, supportati dalla direttiva RealignerTargetCreator, propria del pacchetto GATK, il contenuto del file *bam* è riallineato localmente in modo da diminuire ed evidenzire il numero di variazioni presenti.
Per riassumere l'intero processo è possibile osservare la raffigurazione seguente.

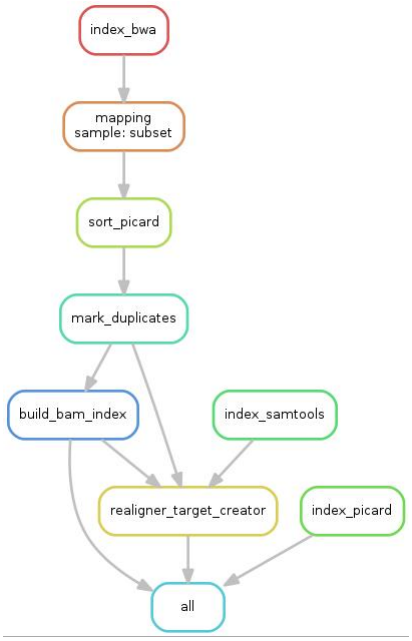

Figura 2.1: Workflow

L'algoritmo GATK-LOD$_n$ è stato integrato solo fino al riallineamento di GATK, escludendo quindi i passaggi di riallineamento Indel e di ricalibrazione dei punteggi di qualità. Allo stesso tempo non sono state considerate le fasi fondamentali di chiamata alle varianti e l'implementazione del classifica-

tore LOD_n di MuTect, vera novità del metodo GATK-LOD_n.
Questa forte selezione degli step è stata voluta appositamente per non appesantire lo studio sull'ottimizzazione computazionale e, quindi, è stato scelto di concentrare le analisi solo su quelle operazioni che formano le basi dell'algoritmo genetico. In tale maniera, è facilitato sia l'indagine sulla scalabilità che il confronto tra i diversi apparecchi adoperati, così da poter trarre rapidamente le prime conclusioni.

Organizzazione dei prodotti

Il materiale su cui sono condotte le analisi statistiche sono i particolari tecnici e tempistici di ognuna delle regole completate. Tali dati sono procurati, come già spiegato nel **capitolo Snakemake**, dalla direttiva benchmark in ogni passaggio. Ciò significa che al termine dell'algoritmo sono prodotti tanti singoli file di benchmarking quante regole sono state completate, contando pure il caso in cui esse venissero ripetute ricorsivamente. Per controllare la quantità di file e agevolare il futuro studio statistico, è stato scritto uno script in Python, denominato `script_benchmark`.
Ogni file benchmark è dotato di un nome avente funzione di etichetta; in particolare, la forma è la seguente.

benchmark_{name}_subset_{sample}_n_sim_{n_sim}_cputype_{cpu_type}_thrs

Gli attributi contenuti nel nome indicano ordinatamente: il nome della regola completata, il tipo di campione analizzato, il numero della simulazione, il tipo di cpu utilizzata e il numero di threads e di cpu adoperati. Mentre i primi due sono ricavati in automatico, gli altri fanno riferimento al file di configurazione dello Snakefile, come sarà spiegato nel paragrafo seguente.
Il lavoro svolto dallo script consiste quindi nel leggere ognuno dei file benchmark generati, considerare le etichette delle simulazioni e trasferire i dati incolonnati in un'unica tabella. La tabella risultante non è dotata, quindi, solo delle colonne predefinite da *Snakemake* nell'operazione di benchmarking(presentate nella sezione **benchmark** della sezione 1.2), bensì è arricchita dai dettagli contenuti nell'etichettatura. In particolare, i dati sono organizzati nella tabella in ordine crescente rispetto al numero di simulazione.
Il nome prestabilito per la tabella si può ritrovare nel codice sorgente dell'eseguibile ed è necessario accedere ad esso e modificare tale nominativo se si desidera crearne una nuova. Difatti, all'avvio dello script è controllato se sia presente una tabella con lo stesso nome: in caso affermativo, i nuovi dati sono accodati ai precedenti; al contrario, ne è inizializzata un'altra.
Attraverso questo sistema, il prodotto finale della fase di esecuzione, ed un

unico oggetto dell'analisi statistica, è una tabella che racchiude le proprietà relative alle simulazioni ultimate.
Prima di procedere con l'esposizione dello studio effettuato su tali tabelle, è indispensabile perfezionare la descrizione del procedimento, specificando le opzioni di configurazione su cui essa si struttura.

2.1.3 Configurazione

Le ultime componenti del procedimento da delineare sono le modalità che definiscono le condizioni sotto cui deve essere portata avanti la computazione. Queste componenti sono riportate nei file di configurazione e, in questo progetto, sono stati scritti due file di questo tipo che corrispondono rispettivamente a *Snakemake* e a *Conda*.
Le informazioni necessarie allo Snakefile sono procurate dal file *config.yaml* che comunica a *Snakemake* alcuni dettagli della simulazione e le indicazioni su certe dipendenze.
Gli attributi della simulazione completano semplicemente l'etichettatura dei file di benchmarking e, pur rivestendo uno ruolo marginale, tali etichette permettono allo script di organizzazione dei dati una lettura rapida e quindi un'istanza immediata della tabella. Come già citato precedentemente, i dettagli trascritti nel file di configurazione sono il tipo di cpu sfruttata, il numero della simulazione, il numero di threads adoperati e il numero di cpu coinvolte.
E' stato scelto di gestire queste informazioni come parametri per non irrigidire il programma, visto che gli apparecchi e i meccanismi usati possono essere innumerevoli. Nello specifico, per modificare questi dettagli nelle etichette è sufficiente agire da linea di comando, come indicato nel paragrafo 1.2.1.
Sempre nello stesso file di configurazione sono presenti alcune chiavi che rappresentato le dipendenze non implementabili direttamente negli ambienti di *Conda* per *Snakemake*. Innanzitutto, sono presenti due indicazioni necessarie per il meccanismo di mapping, che sono l'utilizzo di *Illumina* come piattaforma e di *WES-Nextera-Rapid-Capture* come libreria. A seguire, è indicato l'indirizzo in cui si può trovare il Genome Analysis ToolKit, da applicare nella procedura di riallineamento.
Altre istruzioni sulla configurazione del procedimento sono richieste da *Conda* per la creazione, attivazione e disattivazione degli ambienti di lavoro durante l'esecuzione di *Snakemake*(paragrafo 1.2.1).
Nell'ambito di questa ricerca, è stato scritto solo un documento di configurazione relativo a *Conda*, dato che la maggior parte dei processi necessita di un ambiente con le stesse caratteristiche. Nello specifico, quest'unico file di riferimento è contenuto nella directory *envs* ed è chiamato `config_conda.yaml`.

I requisiti richiesti per l'ambiente sono soddisfatti istruendo espressamente *Conda* all'utilizzo del canale *bioconda* per procedere con l'installazione di tre software coinvolti nel sequenziamento: *BWA*, *Picard* e *Samtools*.
Ora che l'ossatura del sistema è stata argomentata e sono stati approfonditi pure le procedure configurative, è possibile illustrare il percorso di studio statistico effettuato, accompagnato da una presentazione del genere di simulazioni conseguite.

2.2 Studio statistico