Questo breve documento contiene qualche idea volutamente descritta in modo sintetico. Il lavoro effettivo verrà poi concordato con lo studente in base agli interessi specifici e ai tempi d'esecuzione. La tesi potrà essere scritta/svolta in inglese, ma per semplicità le tracce proposte sono in italiano.

Lista aggiornata agosto 2023

Quantum Serverless Computing

Quantum Computing è un nuovo paradigma computazionale basato sulle leggi della meccanica quantistica. Esistono oggi diversi linguaggi di riferimento e diversi tentativi d'uso, ma uno dei problemi attualmente aperti riguarda la realizzazione di sistemi complessi e la possibilità di scalare soluzioni quantistiche. A fronte di queste difficoltà si stanno sviluppando soluzioni ibride che si propongono di integrare elementi di computazione quantistica in sistemi tradizionali. Solitamente, si tende ad usare soluzioni che estendono il concetto di *worklfow*, ma questo lavoro prende spunto dai principi alla base del paradigma computazionale chiamato Function as a Service (FaaS), o Serverless computing. Si vorrebbe definire il concetto di funzione quantistica e pensare poi alla sua integrazione in sistemi FaaS tradizionali per avere sistemi ibridi che sfruttano il paradigma a funzioni per integrare elementi tradizionali e quantistici.

Spark, Machine Learning e gestione efficiente delle risorse

Le ultime versioni di Spark supportano PyTorch e consentono quindi l'integrazione del modo big data con deep/machine learning. Tuttora però la gestione delle risorse di calcolo in Spark, principalmente core di CPU, non è particolarmente ottimizzata, non è possibile prevedere *deadline* (durate) per il completamento delle diverse attività e neppure gestire le risorse in modo efficiente in caso di attività in parallelo. In passato, avevamo proposto DynaSpark con l'obiettivo di risolvere i problemi appena elencati. Oggi quindi abbiamo due versioni di Spark incompatibili: ognuna ha delle caratteristiche interessanti, ma l'unione non è possibile per le troppe differenze accumulate nel tempo. Lo scopo principale del lavoro è quindi di studiare le due versioni e creare un'opportuna versione integrata. Non è un problema di compilazione o merge di repository diverse, ma l'evoluzione di Spark, rispetto alla versione utilizzata per creare DynaSpark, richiede un'attenta opera di reverse engineering e la progettazione di un'architettura integrata, prima di pensare di sfruttare il codice esistente per la sua realizzazione. Un'attenta valutazione delle prestazioni della soluzione realizzata concluderà il lavoro.

Test metamorfico per applicazioni Deep Learning per Computer Vision

I modelli di Deep Learning per Computer Vision vengono utilizzati ampiamente in varie applicazioni, come la rilevazione degli oggetti, la classificazione delle immagini e il riconoscimento facciale. Questi modelli hanno dimostrato una notevole accuratezza nell'identificazione e nella classificazione degli oggetti nelle immagini, ma sono vulnerabili agli "attacchi avversari" che possono facilmente ingannarli, portando a previsioni errate. Questi attacchi sono una preoccupazione significativa, soprattutto quando i modelli di Deep Learning vengono utilizzati in applicazioni critiche per la sicurezza, come i veicoli autonomi o le immagini mediche. Il testing metamorfico è un approccio di testing che può aiutare a garantire la robustezza dei modelli di Deep Learning contro gli attacchi avversari. Questo approccio prevede l'applicazione di un insieme di regole o trasformazioni predefinite alle immagini in ingresso e la verifica che l'output del modello rimanga coerente. L'approccio può aiutare a rivelare difetti nascosti nell'architettura del modello di Deep Learning e migliorare la sua robustezza. L'obiettivo principale di questa tesi è quello di proporre una soluzione di testing metamorfico per i modelli di Deep Learning per Computer Vision.

Studio di soluzioni self-adaptive per sistemi basati su ML

Per anni, abbiamo provato ad aggiungere capacità auto-adattative (self-adaptive) a sistemi software tradizionali. L'uso di soluzioni basate su ML sta cambiando molti sistemi software, ma alcuni dei problemi che avevamo motivato l'idea dell'adattività a runtime restano. Ad esempio, il modello creato durante la fase di training può essere ottimo in un certo contesto, ma non (sufficientemente) adatto in altri. Chiaramente, si potrebbe pensare di ricominciare da capo, ma costi e tempi non sono sempre praticabili. La ricerca quindi si propone di studiare soluzioni efficienti per poter adattare il modello di riferimento in corso d'opera, considerando che quanto fatto per i sistemi tradizionali non è più utilizzabile, il processo di adattamento è fortemente guidato dai dati a disposizione ed i tempi sono spesso non trascurabili. È quindi ipotizzabile pensare di far coesistere più soluzioni che possano gestire eventuali transitori per il passaggio da una configurazione alla successiva. Il lavoro parte da alcune esperienze già fatte cercando di ricreare il ciclo di controllo ad anello chiuso tipico dei sistemi self-adaptive tradizionali attorno al modello ML e vorrebbe spingersi a creare modello ML che possano includere le capacità self-adaptive al loro interno.

Deployment avanzato di funzioni di applicazioni edge

Mentre quando si parla di cloud, si adotta l'ipotesi di risorse infinite, quando si gestiscono server edge, le risorse sono limitate e vanno gestite in modo appropriato. Non è quindi pensabile che ogni server possa mettere sempre a disposizione tutti componenti (servizi, funzioni, o altro) che gli utenti potrebbero voler usare. Molti lavori in letteratura, oggi, non affrontano il problema, ma trovare il giusto deployment per consentire agli utenti di usare in modo efficiente le funzionalità richieste non è banale. Nello specifico, Neptune, la nostra soluzione per edge computing, sfrutta il paradigma function-as-a-service e, quindi, il problema non è solo l'allocazione delle diverse funzioni, ma anche la gestione delle immagini per consentire il deployment delle funzioni dove e quando è richiesto. Lo scopo del lavoro, quindi, è di estendere Neptune con la gestione del deployment delle funzioni richieste, senza ipotizzare, come avviene ora, che le funzioni siano (sempre) "magicamente" disponibili. Il lavoro si compone di due parti: la gestione dei repository delle diverse immagini, considerando architetture distribuite e altamente decentralizzate, e il deployment vero e proprio delle funzioni stesse, gestendo i problemi di *cold start* e l'ottimizzazione del deployment.

Gestione dei dati di applicazioni edge

Ad oggi, Neptune, la nostra infrastruttura basata sul paradigma function-as-a-service per edge computing non gestisce i dati: è in grado di instradare le richieste degli utenti e gestire le risorse, ma senza considerare che i dati, se lontani dalle computazioni che dovrebbero usarli, diventerebbero il collo di bottiglia dell'intero sistema. Poiché oggi abbiamo applicazioni sempre più basate su grandi quantità di dati, decentralizzare e ottimizzare la computazione non è più sufficiente. Mentre per anni abbiamo pensato di "avvicinare" i dati alla computazione, il contesto attuale ci invita a fare il contrario, ma anche a pensare a repliche, partizionamenti e migrazioni che consentano all'utente di "restare" vicino ai dati necessari, usando l'archiviazione sul cloud come ultima ipotesi. Il lavoro si propone quindi di analizzare lo stato dell'arte relativo alla gestione dei dati in federazioni di server edge, identificandone limiti e problemi irrisolti. La prima fase dovrebbe essere funzionale alla definizione di una soluzione innovativa ed efficace, che possa poi essere implementata, come opportuna estensione di Neptune, e valutata con dati ed in contesti realistici.

Soluzioni per MLOps adattativo

Machine Learning Operations (MLOps) si propone di definire un insieme di regole pratiche per gestire modelli machine learning in modo efficiente e affidabile. MLOps aiuta a gestire e monitorare il processo di sviluppo e la pipeline adottata con l'obiettivo di migliorare la qualità dei modelli prodotti.

Recentemente, AutoML è diventato uno strumento molto popolare per facilitare l'uso di soluzioni di machine learning su problemi reali. I diversi framework esistenti (TPOT, H2O, AutoWEKA, auto-sklearn, AutoGluon, and Google AutoML Tables) consentono di fare il training di modelli di base e di combinarli in meta-modelli opportuni. Questo lavoro si propone di studiare ed estendere questi framework per farli diventare sistemi auto-adattativi, ovvero sistemi che dato un budget (in termine di costi, tempo, o risorse) ed un obiettivo di qualità (ad esempio, in termini di accuratezza, robustezza e *fairness*) sfruttano al meglio il budget per raggiungere l'obbiettivo. In particolare, si tratterebbe di identificare i modelli di cui rifare il training, selezionare i dati da usare e stimare, e poi allocare, le risorse richieste.

Soluzioni model-based per Machine Learning

Oggi esistono molte soluzioni diverse per realizzare soluzioni basate su machine learning. Pur essendo interessanti, molte di queste soluzioni vincolano l'utente all'uso dello strumento specifico. Questa "dipendenza" è dovuta al fatto che molte soluzioni usano formati specifici per salvare i modelli ottenuti. Ad esempio, TensorFlow, Caffe, Theano e PyTorch usano tutti soluzioni particolari. Il lavoro di tesi dovrebbe quindi identificare un insieme significativo di soluzioni di riferimento, studiare i loro formati e le soluzioni adottate, studiare anche eventuali proposte di standardizzazione (ad esempio, Open Neural Network Exchange, Apple Core ML format e Neural Network Exchange Format) e proporre poi una soluzione astratta che integri l'esistente e consenta di generare (automaticamente) i diversi formati richiesti.

Bridge tra soluzioni blockchain layer 2

I rollup sono soluzioni "layer 2" per scalare sistemi blockchain. Questo tipo di soluzioni aggiungono alla sicurezza delle soluzioni principali (e.g., Ethereum), throughput elevati e costi bassi sia per gli utenti che per gli sviluppatori di DApp (Decentralized Applications). Purtroppo, la comunicazione e la gestione degli asset (fondi, NFT) tra diverse soluzioni "layer 2", e tra una soluzione "layer 2" e una blockchain principale, richiedono spesso soluzioni ad-hoc che dipendono fortemente dalle piattaforme usate. L'idea di questo lavoro è quindi di studiare soluzioni generiche che possano sistematizzare, semplificare e automatizzare la comunicazione suddetta. Oltre a studiare a fondo le soluzioni esistenti, in un mondo in forte evoluzione, il lavoro si propone di astrarre dalle tecnologie particolari e studiare una soluzione che attraverso l'uso di modelli, o elementi simili, possa poi essere "calata" nei diversi contesti tecnologici in modo automatico e senza richiedere dispendio di energie e risorse.

Framework model-driven per applicazioni IoT

Ogni framework per applicazioni IoT (ad esempio, UML IoT, AndroidThings, NodeRED) ha la propria notazione, il proprio runtime ed il proprio linguaggio di programmazione. Una notazione unica ed un approccio model-driven aiuterebbero a semplificare il problema e consentirebbero al progettista/programmatore di concentrarsi sul cuore dell'applicazione e non sui problemi relativi alla sua implementazione. In questo modo si darebbe anche la possibilità di fare facili sperimentazioni con tecnologie diverse. La tesi dovrebbe studiare le soluzioni disponibili, realizzare un meta-modello unificato, ovvero un'unica notazione di progetto, e realizzare le trasformazioni richieste per generare il codice richiesto dai framework target a partire dal modello (istanze del meta-modello) dell'applicazione di interesse.

Strumenti per l'analisi di modelli BIM

Un modello BIM (Building Information Modeling) cattura tutti gli aspetti di un edificio. Poiché nella sostanza è possibile creare un modello ad oggetti dell'edificio di interesse e di tutte le sue parti, la tesi si propone di creare un linguaggio specifico (domain-specific) per la definizione di vincoli sul modello (ad esempio, per le

distanze dai vicini, per le dimensioni minime di una camera da letto o per il volume di aria scambiata da un impianto di condizionamento) ed un motore di verifica ispirato ai tanti metodi e modelli formali disponibili per l'analisi e la progettazione del software. L'integrazione con un tool di progettazione (ad esempio, AutoCAD o Revit) potrebbe essere un plus. L'idea di fondo è di iniziare a pensare ad un gemello digitale (digital twin) vivo dell'edificio di interesse.

JML come libreria funzionale

Java Modeling Language (JML) è il linguaggio di riferimento usato per insegnare la programmazione per contratto nel corso di Ingegneria del Software. Purtroppo, dopo diversi anni, e dopo varie versioni di Java, oggi non esiste un tool utilizzabile per definire e valutare asserzioni scritte in JML; esistono diverse soluzioni parziali che funzionano solamente in casi molto specifici. Al tempo stesso, Java è stato esteso in modo significativo con la possibilità di definire funzioni generiche e con costrutti funzionali. Il lavoro si propone quindi di ripensare JML e di ridefinirlo attraverso opportuni elementi funzionali generici. L'obiettivo è di realizzare un'opportuna libreria indipendente dai diversi IDE e compilatori Java e che possa essere utilizzata in modo semplice ed intuitivo a prescindere dalla versione di Java effettivamente usata.