Università degli Studi di Genova
Facoltà di Ingegneria

Tesi di Laurea Magistrale in Ingegneria Elettronica

Progetto e implementazione di un motore di esecuzione multipiattaforma per applicazioni IoT

Relatore:
Chiar.mo Prof. Riccardo Berta

Candidati:
Luca Lazzaroni
Andrea Mazzara

24 Luglio 2020

Ringraziamenti

Ringraziamenti.

Dedica

Abstract

Le applicazioni dell'Internet of Things (IoT) richiedono spesso una notevole larghezza di banda, bassa latenza e performance affidabili, e al tempo stesso devono rispettare requisiti normativi e di conformità, motivo per cui il Cloud Computing non risulta adatto in questi particolari casi applicativi.
Per ovviare ai problemi sopracitati, negli ultimi anni si sta affermando un nuovo approccio, l'Edge Computing: un'architettura distribuita di micro data center, ciascuno in grado di immagazzinare ed elaborare i dati a livello locale e in seguito trasmetterli ad un data center centralizzato o a un database su Cloud.
Edge Engine nasce allo scopo di realizzare un motore il più generico possibile e slegato dall'hardware, tale da raccogliere dati provenienti dai dispositivi ad esso collegati, elaborarli e inviarli su Cloud.
In questo specifico caso si tratterà lo sviluppo di Edge Engine per dispositivi fissi. Il linguaggio di programmazione utilizzato sarà il C++ con l'intento di ottenere un prodotto finale multipiattaforma, caratteristica concorde con i requisiti preposti di genericità e indipendenza dall'hardware.
DA COMPLETARE CON PARTE UNITY

Indice

Elenco delle figure

Introduzione

Negli ultimi anni, la mole di dati prodotta da aziende e privati sta crescendo esponenzialmente, tanto che, entro il 2022, si stima che in media si avranno 50 dispositivi connessi a Internet per abitazione [1]. Ovviamente questi dati necessitano di essere processati e conservati, oltre che condivisi tra più dispositivi all'occorrenza. A tal fine, la modalità che si è adottata maggiormente negli ultimi anni è quella del Cloud Computing: i dati non vengono processati in locale per mancanza di risorse, ma inviati a specifici data center online in grado di elaborarli e processarli, oltre che conservarli. Tale approccio introduce però alcune criticità:

- **Latenza**: in molti ambiti è richiesta un'elaborazione dei dati in tempo reale, si pensi per esempio ad un'eventuale applicazione che permetta a un veicolo autonomo di riconoscere i pedoni. In questo specifico caso è richiesta una bassissima latenza dato l'enorme rischio in gioco. Tuttavia, proprio l'invio dei dati al Cloud, la successiva elaborazione degli stessi e, infine, l'invio di un feedback al dispositivo in uso, introducono ritardi non trascurabili, pertanto in questi specifici casi il Cloud Computing risulta non essere l'approccio migliore.

- **Scalabilità**: l'invio dei dati al Cloud è problematico in tal senso, dato soprattutto il numero in crescita esponenziale di dispositivi connessi. Inoltre, l'invio di tutti i dati al Cloud è inefficiente in termini di consumo di risorse, in particolare se non tutti i dati sono necessari al Deep Learning.

- **Privacy**: l'invio di dati sensibili a server online aumenta i rischi di furto di tali informazioni, oltre al fatto che l'utente spesso e volentieri non è a conoscenza di come questi dati verranno trattati né tantomeno di dove saranno conservati.

Una possibile soluzione a queste tre criticità, proprie del Cloud Computing, è l'Edge Computing (si veda figura 1.1).

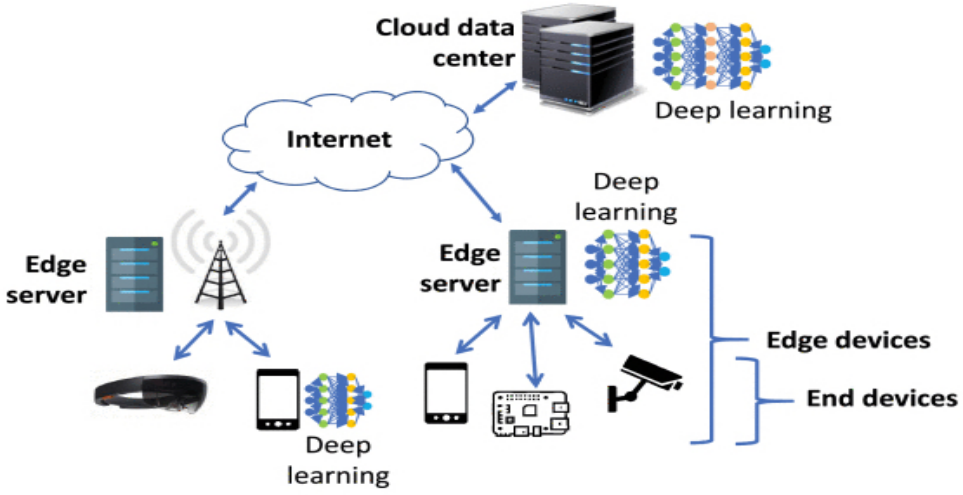

Figura 1.1: Struttura dell'Edge Computing

Tale approccio prevede una rete di micro data center posti nelle vicinanze dei dispositivi che rilevano i dati da elaborare. Proprio questa vicinanza alle sorgenti dei dati permette di ridurre drasticamente la latenza (si veda figura 1.2). Inoltre, al fine di incrementare le prestazioni in termini di scalabilità, è prevista una struttura gerarchica dei dispositivi connessi, oltre al fatto che non è necessario apportare modifiche o espansioni ai data center in Cloud siccome i dati vengono elaborati in locale. Per quanto riguarda infine i vincoli di privacy, l'Edge Computing prevede l'elaborazione dei dati alla sorgente, solitamente grazie a un server locale, perciò i dati non vengono trasmessi sulla rete globale, riducendo dunque i rischi che ne deriverebbero.

Figura 1.2: Confronto tra Edge e Cloud Computing

L'approccio Edge Computing presenta però alcune criticità.
Uno degli aspetti più importanti da considerare è l'elevata quantità di risorse richiesta da determinati algoritmi (come ad esempio quelli di Machine Learning o di elaborazione delle immagini), in contrapposizione con l'utilizzo di nodi locali dotati di ridotta potenza di calcolo rispetto ai server centralizzati.
Un secondo problema è la coordinazione tra i dispositivi Edge e il Cloud, considerando che per ognuno di essi si avranno verosimilmente differenti capacità di calcolo e tipologie di connessione alla rete.
In ultimo, dal lato privacy, anche se le potenziali minacce sono ridotte rispetto a una soluzione unicamente basata sul Cloud Computing, la riservatezza rimane comunque

un punto delicato poiché i dati necessitano di essere condivisi tra i vari dispositivi e pertanto risulta necessario l'utilizzo della comunicazione in rete, oltre al fatto che il dispositivo perimetrale avrà meno potenza di calcolo da dedicare a complessi algoritmi di criptazione.
Con queste premesse nasce l'idea di realizzare Edge Engine: un runtime system generico, slegato dall'hardware, in grado di interpretare codice per dispositivi multipiattaforma, comprese board di sviluppo per microcontrollori. Tale sistema è in grado di elaborare i flussi di dati provenienti dai sensori ad esso collegati grazie all'utilizzo degli script: insiemi di operazioni prestabilite che possono anche essere composte al fine di eseguire calcoli complessi sui dati in ingresso. L'Edge Engine è configurato in modo tale da recuperare dal Cloud gli scripts associati al dispositivo in uso, eseguirli localmente e poi trasmettere nuovamente al Cloud i risultati ottenuti. Per il corretto funzionamento di Edge Engine è dunque necessario un server online che conservi gli scripts e le descrizioni dei vari dispositivi. In questo specifico caso verrà utilizzato Measurify: una piattaforma cloud creata dall'Elios Lab dell'Università di Genova per gestire oggetti smart dell'Internet of Things (IoT).
Lo scopo del progetto in esame sarà la realizzazione di Edge Engine per sistemi PC (Windows/Linux/MacOS) e, successivamente, l'impiego di tale motore in un contesto di realtà virtuale, quale un simulatore medico, in modo da illustrare e testare un'ulteriore modalità di impiego del sistema.

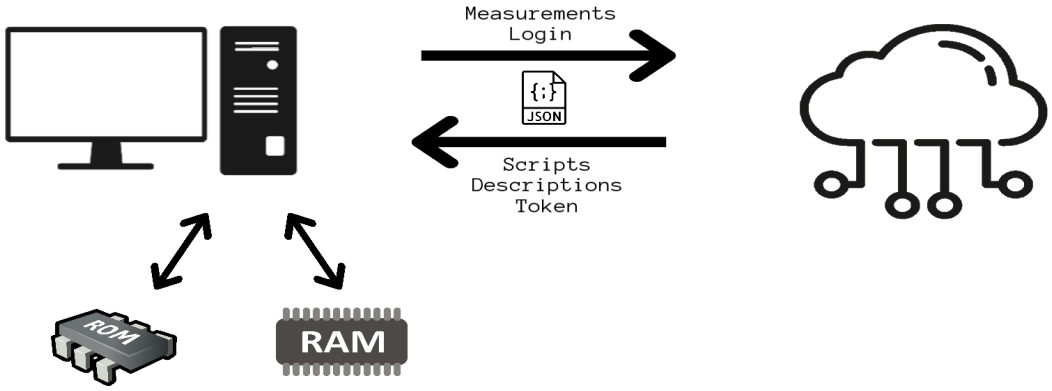

Figura 1.3: Edge Engine per PC

Stato dell'Arte

In un mondo nel quale sempre più dispositivi necessitano di connessione alla rete e i dati personali richiedono un alto livello di protezione, oltre che di elaborazione attraverso l'impiego di algoritmi più o meno complessi, le potenzialità offerte in tale direzione dall'Edge Computing hanno attirato l'attenzione dei colossi del settore.
Google, ad esempio, ha recentemente rilasciato Edge TPU e Cloud IoT Edge [2]: il primo è un ASIC creato specificamente per eseguire l'IA a livello periferico, mentre il secondo è una piattaforma per l'Edge Computing che estende le capacità di elaborazione dei dati e Machine Learning di Google Cloud ai dispositivi perimetrali. L'idea di fondo è quella di costruire i propri modelli sul Cloud, per poi utilizzarli su dispositivi Cloud IoT Edge sfruttando le potenzialità offerte dall'acceleratore hardware Edge TPU.
Amazon, all'interno della sua offerta di servizi Cloud (AWS), mette a disposizione la soluzione IoT Greengrass [3], che semplifica l'inferenza di Machine Learning in locale sui dispositivi, mediante modelli creati, formati e ottimizzati nel Cloud. L'utente può inoltre utilizzare modelli il cui training viene fatto in prima persona. L'AWS IoT Greengrass dispone del runtime Lambda [4]: un gestore di messaggi, accesso alle risorse, ecc. I requisiti minimi a livello hardware sono 1 GHz di frequenza del processore e 128 MB di RAM.
Microsoft mette a disposizione Azure IoT Edge [5]: un servizio che permette di distribuire i carichi di lavoro del Cloud per eseguirli su dispositivi perimentrali dell'IoT. Il codice di IoT Edge supporta numerosi linguaggi tra cui C, C#, Java, Node.js e Python, inoltre la latenza è ridotta siccome i dati vengono elaborati in locale, con la possibilità di usare l'architettura hardware Microsoft, Project Brainwave [6]. I dispositivi perimetrali possono poi anche funzionare in condizioni di connessione a internet scarsa, grazie alla gestione dei dispositivi di Azure che sincronizza in automatico lo stato più recente degli apparecchi dopo la riconnessione a internet.
In ultimo, IBM ha sviluppato IBM Edge Application Manager [7]: una piattaforma intelligente, sicura e flessibile che fornisce uno strumento di gestione per l'elaborazione perimetrale. La soluzione proposta è autonoma, ossia consente ad un singolo amministratore di gestire scalabilità, variabilità e frequenza di modifica degli ambienti delle applicazioni su decine di migliaia di endpoint. Gli endpoint perimetrali si eseguono su contenitori Red Hat OpenShift [8]. IBM Edge Application Manager supporta inoltre tool di IA per Deep Learning e riconoscimento di voce e immagini, oltre all'analisi video e acustica.

Bibliografia

[1] IoTedge: *L'edge computing può fare la differenza nell'analisi dei dati*, https://www.iotedge.it/edge-platform/ledge-computing-puo-fare-la-differenza-nellanalisi-dei-dati/

[2] Injong Rhee: *Bringing intelligence to the edge with Cloud IoT*, https://cloud.google.com/blog/products/gcp/bringing-intelligence-edge-cloud-iot

[3] Amazon Web Services - AWS IoT Greengrass, http://www.aws.amazon.com/greengrass/ml/

[4] Amazon Web Services - AWS Lambda, https://docs.aws.amazon.com/it_it/lambda/latest/dg/welcome.html

[5] Microsoft - Azure IoT Edge, https://azure.microsoft.com/it-it/services/iot-edge/

[6] Microsoft - Project Brainwave, https://www.microsoft.com/en-us/research/project/project-brainwave/

[7] IBM - IBM Edge Application Manager, https://www.ibm.com/it-it/cloud/edge-application-manager

[8] Red Hat - OpenShift, https://www.redhat.com/it/technologies/cloud-computing/openshift

[9] Piero Todorovich: L'Internet delle cose (IoT): cos'è e come rivoluzionerà prodotti e servizi, https://www.zerounoweb.it/analytics/big-data/internet-of-things-iot-come-funziona

[10] Giampiero Carli Ballola: Edge computing per IoT: ecco a cosa serve e come utilizzarlo, https://www.zerounoweb.it/techtarget/searchdatacenter/edge-computing-cose-come-implementarlo/

[11] Amazon Web Service. 2019: AWS Lambda@Edge, https://aws.amazon.com/it/lambda/edge/

[12] Amazon CloudFront. 2019, https://aws.amazon.com/cloudfront/

[13] Kunal Yadav: What is AWS Lambda or Serverless?
`https://hackernoon.com/what-is-aws-lambda-or-serverless-f0a006e9d56c`

[14] Villamizar, M.; et al., *Infrastructure Cost Comparison of Running Web Applications in the Cloud Using AWS Lambda and Monolithic and Microservice Architectures*. 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), Cartagena, 2016, pp. 179-182.

[15] William Tärneberg, Vishal Chandrasekaran, and Marty Humphrey, *Experiences creating a framework for smart traffic control using AWS IOT*. In Proceedings of the 9th International Conference on Utility and Cloud Computing (UCC '16). ACM, New York, NY, USA, 63-69.

[16] Microsoft: Azure IoT Edge,
`https://docs.microsoft.com/it-it/azure/iot-edge`

[17] Forsström, S.; Jennehag, U., *A performance and cost evaluation of combining OPC-UA and Microsoft Azure IoT Hub into an industrial Internet-of-Things system*. 2017 Global Internet of Things Summit (GIoTS), Geneva, 2017, pp. 1-6.

[18] Familiar, B., *Microservices, IoT, and Azure: Leveraging DevOps and Microservice Architecture to Deliver SaaS Solutions*. Apress, 2015.

[19] Internet of Things intelligente con Google Cloud IoT,
`https://www.01net.it/internet-of-things-intelligente-google-cloud-iot/`

[20] Jain, R.; Tata, S., *Cloud to Edge: Distributed Deployment of Process-Aware IoT Applications*. 2017 IEEE International Conference on Edge Computing (EDGE), Honolulu, HI, 2017, pp. 182-189.

[21] Fan, K.; Pan, Q.; Wang, J.; Liu, T.; Li, H.; Yang, Y., *Cross-Domain Based Data Sharing Scheme in Cooperative Edge Computing*. 2018 IEEE International Conference on Edge Computing (EDGE), San Francisco, CA, 2018, pp. 87-92.

[22] Cirani, S.; Ferrari, G.; Iotti, N.; Picone, M., *The IoT hub: a fog node for seamless management of heterogeneous connected smart objects*. 2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking - Workshops (SECON Workshops), Seattle, WA, 2015, pp. 1-6.

[23] Noghabi, S. A.; Kolb, J.; Bodik, P.; Cuervo, E, *Steel: Simplified Development and Deployment of Edge-Cloud Applications*. 10th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 18).

[24] Xu, X.; Huang, S.; Feagan, L.; Chen, Y.; Qiu Y.; Wang, Y., *EAaaS: Edge Analytics as a Service*. 2017 IEEE International Conference on Web Services (ICWS), Honolulu, HI, 2017, pp. 349-356.

[25] Yuan, J.; Li, X., *A Reliable and Lightweight Trust Computing Mechanism for IoT Edge Devices Based on Multi-Source Feedback Information Fusion.* in IEEE Access, vol. 6, pp. 23626-23638, 2018.

[26] Brian Park: AUnit,
`https://github.com/bxparks/AUnit`