

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 multiplatforma, caratteristica concorde con i requisiti preposti di genericità e indipendenza dall'hardware.

DA COMPLETARE CON PARTE UNITY

Indice

1	Introduzione	1
2	Stato dell'Arte	4

Elenco delle figure

1.1	Struttura dell'Edge Computing	2
1.2	Confronto tra Edge e Cloud Computing	2
1.3	Edge Engine per PC	3

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).

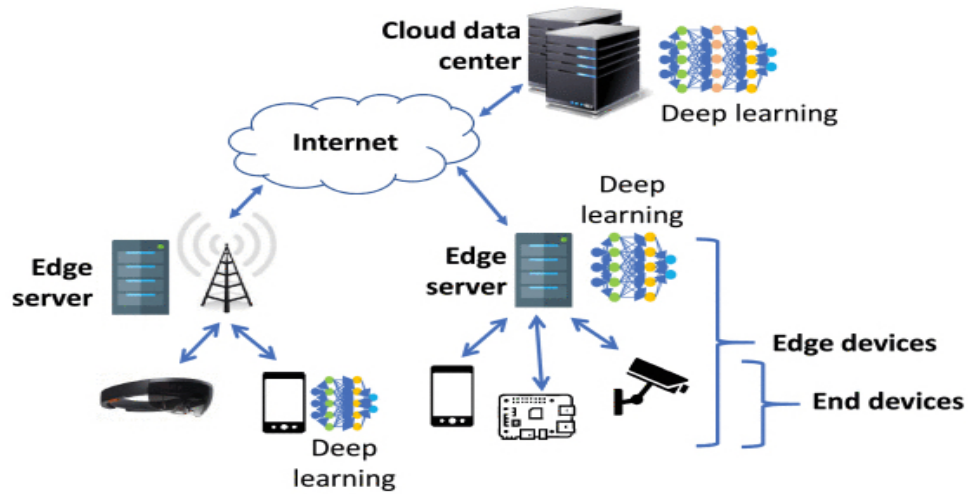


Figura 1.1: Struttura dell'Edge Computing

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.

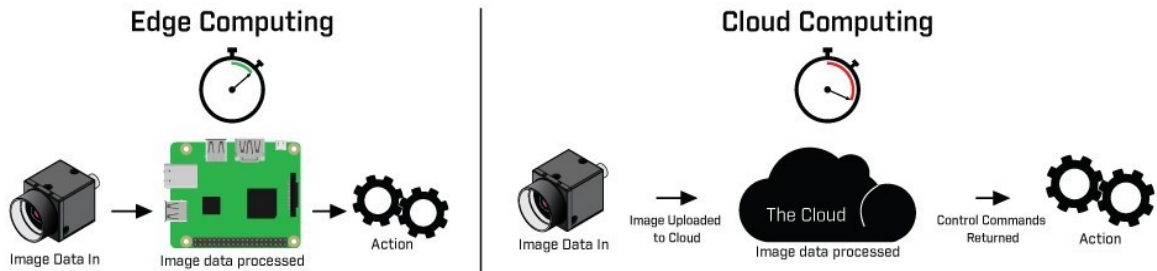


Figura 1.2: Confronto tra Edge e Cloud Computing

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 crittazione.

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 script 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 script 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.

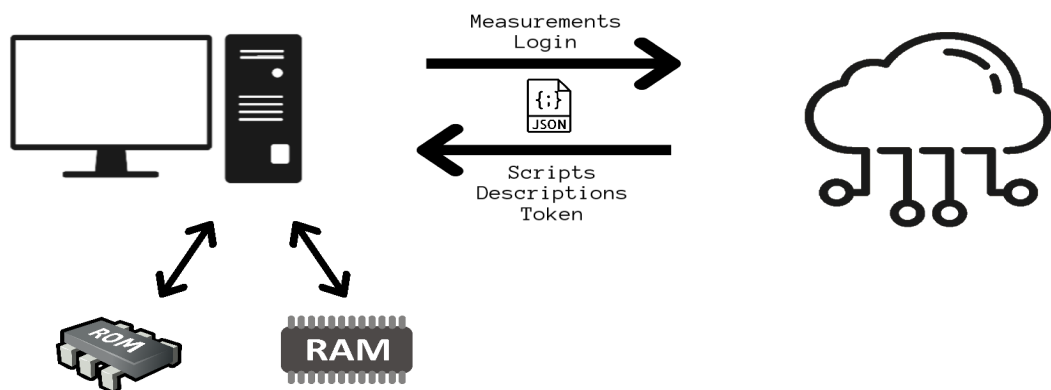


Figura 1.3: Edge Engine per PC

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 (ML) 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. Tale circuito è inoltre in grado di eseguire TensorFlow Lite [3]: una piattaforma che fornisce un set di strumenti i quali permettono all'utente di convertire modelli di reti neurali TensorFlow in versioni semplificate e ridotte, adatte ai dispositivi Edge [4]. Amazon, all'interno della sua offerta di servizi Cloud (AWS), mette a disposizione la soluzione IoT Greengrass [5], che semplifica l'inferenza di ML 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 [6]: 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 [7]: un servizio che permette di distribuire i carichi di lavoro del Cloud per eseguirli su dispositivi perimetrali 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 [8]. 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. Microsoft ha inoltre rilasciato EdgeML [9]: una suite di algoritmi di ML progettata per un utilizzo in situazioni di risorse ridotte. I risultati pubblicati sull'uso di EdgeML per il training su Cloud in condizioni di limitata potenza di calcolo [10]. Al momento tale libreria prevede algoritmi di tipo k-Nearest Neighbors (kNN) per classificazione, regressione e ranking noti con i nomi di Bonsai, ProtoNN e Robust PCA [11].

In ultimo, IBM ha sviluppato IBM Edge Application Manager [12]: 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 [13]. IBM Edge Application Manager supporta inoltre tool di IA per Deep Learning e riconoscimento di voce e immagini, oltre all'analisi video e acustica.

Anche nella letteratura scientifica è possibile reperire numerosi articoli pubblicati nel periodo recente, a indicare un orientamento in direzione Edge Computing.

In particolare, riguardo la possibilità di eseguire codice multiplatforma che sfrutti le potenzialità offerte da questo approccio, si fa riferimento ad un articolo riguardante lo sviluppo di algoritmi di allocazione risorse per migliorare le prestazioni delle Vehicular Networks [14]. Per la selezione della piattaforma di esecuzione (es. Cloud Computing, Mobile Edge Computing, o Local Computing) viene utilizzato l'algoritmo k-Nearest Neighbor (kNN), mentre, per il problema di allocazione delle risorse computazionali, il Reinforcement Learning (RL). I risultati della simulazione mostrano che, rispetto all'algoritmo di base in cui tutte le attività vengono eseguite sul server di Edge Computing locale o mobile, lo schema di allocazione delle risorse consente una riduzione significativa della latenza che si attesta intorno all'80%.

Per far fronte al problema del consumo di energia dei device IoT, l'approccio Edge Computing è stato trattato all'interno di un articolo di Olli Väänänen e Timo Hämäläinen [15]. Aspetto importante di tale ricerca è la focalizzazione sul concetto di virtualizzazione. L'approccio consigliato è quello della container-based virtualization: ambienti di sviluppo virtuali emulano un determinato sistema operativo (OS), in questo modo è possibile produrre codice indipendente dalla piattaforma. I container possono essere eseguiti da dispositivi dotati di risorse limitate (es. Raspberry Pi), tuttavia richiedono una certa potenza computazionale e un generico OS.

Un ulteriore aspetto critico riguardante l'Edge Computing è il coordinamento tra dispositivi. Nell'ambito dei vari modelli che è possibile adottare, quelli tuple-based sono i più conosciuti e utilizzati, principalmente per la loro flessibilità [16]. Ne è un esempio il modello Tusow (Tuple Spaces over the Web) [17]. Le tuple rappresentano i messaggi o i dati scambiati tra i componenti e l'interazione tra gli stessi è gestita definendo come e quando i vari agenti coinvolti sono in grado di inserire, leggere o elaborare i dati. Tusow permette ai vari clients di rappresentare le tuple in diversi formati: YAML, JSON, XML, FOL e plain text. Il supporto multiplatforma offerto da tale sistema mira a fornire un mezzo di interazione ad alto livello per clients eterogenei, permettendo agli sviluppatori di non curarsi delle complessità intrinseche della rete di basso livello.

La ricerca scientifica si sta orientando verso l'implementazione di algoritmi di ML applicati all'Edge Computing, come anche dimostrato dalle proposte di Google e Microsoft in tale ambito [4], [10].

È stata ad esempio testata la possibilità di eseguire algoritmi di ML su Raspberry Pi [18]. In particolare sono stati implementati tre algoritmi: Support Vector Machine (SVM), Multi-Layer Perceptron e Random Forest, raggiungendo una precisione oltre l'80% mantenendo però un basso consumo di energia.

È stato inoltre analizzato un approccio di Deep Learning in cui vengono utilizzati autoencoder a livello Edge in modo da operare una riduzione delle dimensioni dei dati [19], portando benefici in termini di tempo e spazio richiesti. Tale progetto illustra tre differenti scenari. Nel primo, dati provenienti da sensori vengono mandati a dei "nodi Edge", dove viene applicata la riduzione delle dimensioni, per poi applicare le tecniche

di ML su Cloud. Nel secondo, i dati che sono stati ridotti in Edge vengono trattati sul Cloud in modo da riottenere i dati originari per poi successivamente applicare il ML. Nel terzo ed ultimo caso invece, la tecnica utilizzata è quella del Cloud Computing: i dati vengono mandati dai sensori direttamente al Cloud. I risultati finali mostrano come l'utilizzo di autoencoder a livello Edge riduca il numero di features e, di conseguenza, l'ammontare di dati inviati al Cloud.

In ultimo, Respiro è un inalatore smart prodotto da Amiko [20] che contiene un processore ultra-low-power ARM Cortex-M. Tale inalatore fa affidamento sul ML per interpretare le vibrazioni provenienti dal sensore posto al suo interno. Il dispositivo è addestrato per riconoscere diverse tipologie di respiro e calcolare di conseguenza importanti parametri in ambito medico come la capacità polmonare e la tecnica di inalazione. I dati così prodotti vengono poi processati da un'applicazione e, se la connessione lo permette, inviati al Cloud per essere conservati e elaborati potendoli confrontare inoltre con i risultati altrui a fini statistici. Infine, attraverso l'app, viene inviato un feedback all'utente affinché possa avere sotto controllo i dati estrapolati.

Queste sono le principali soluzioni disponibili nel mercato tecnologico e nelle più recenti ricerche accademiche. È possibile notare come gli aspetti ritenuti più importanti siano l'ottimizzazione delle risorse, il basso consumo di energia, l'indipendenza dalla piattaforma, la privacy e la bassa latenza, oltre alla facilità di utilizzo.

L'Edge Engine da noi proposto nasce con l'idea di base di creare un runtime system slegato dalla piattaforma che soddisfi i requisiti cardine dell'Edge Computing sopracitati. A differenza delle soluzioni appena presentate, questo progetto cerca di fornire un software completamente personalizzabile e adattabile a qualsiasi apparato IoT.

Riferimenti bibliografici e sitografici

- [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] TensorFlow Lite, <https://www.tensorflow.org/lite>
- [4] Louis M., Azad Z., Delhadtehrani L., Gupta S.L., Warden P., Reddi V., Joshi A.: ***Towards deep learning using TensorFlow Lite on RISC-V***. Workshop Comput. Archit. Res. RISC-V 2019
- [5] Amazon Web Services - AWS IoT Greengrass, <http://www.aws.amazon.com/greengrass/ml/>
- [6] Amazon Web Services - AWS Lambda, https://docs.aws.amazon.com/it_it/lambda/latest/dg/welcome.html
- [7] Microsoft - Azure IoT Edge, <https://azure.microsoft.com/it-it/services/iot-edge/>
- [8] Microsoft - Project Brainwave, <https://www.microsoft.com/en-us/research/project/project-brainwave/>
- [9] Microsoft - EdgeML, Machine learning algorithms for resource constrained devices, <https://microsoft.github.io/EdgeML/>
- [10] Dennis D.K., Gopinath S., Gupta C., Kumar A., Kusupati A., Patil S.G., Simhadri H.V.: ***EdgeML Machine Learning for Resource-Constrained Edge Devices***, <https://github.com/Microsoft/EdgeML>
- [11] EdgeML Algorithms, <https://github.com/Microsoft/EdgeML/wiki/Algorithms>
- [12] IBM - IBM Edge Application Manager, <https://www.ibm.com/it-it/cloud/edge-application-manager>

- [13] Red Hat - OpenShift,
<https://www.redhat.com/it/technologies/cloud-computing/openshift>
- [14] Yaping Cui, Yingjie Liang, Ruyan Wang: *Resource Allocation Algorithm With Multi-Platform Intelligent Offloading in D2D-Enabled Vehicular Networks*, in IEEE Access (Volume: 7), pag. 21246 - 21253, 14 Febbraio 2019,
<https://ieeexplore.ieee.org/abstract/document/8642506>
- [15] Olli Väänänen, Timo Hämmäläinen: *Requirements for Energy Efficient Edge Computing: A Survey*, in Internet of Things, Smart Spaces, and Next Generation Networks and Systems, vol. 11118, 29 Settembre 2018,
https://link.springer.com/chapter/10.1007/978-3-030-01168-0_1
- [16] G. Ciatto, S. Mariani, A. Omicini, F. Zambonelli, M. Louvel: *Twenty years of coordination technologies: State-of-the-art and perspectives*, in Coordination Models and Languages, Springer, vol. 10852, pag. 51-80, 2018
- [17] G. Ciatto, L. Rizzato, A. Omicini, S. Mariani, *TuSoW: Tuple Spaces for Edge Computing*, 2019 28th International Conference on Computer Communication and Networks (ICCCN), Valencia, Spagna, 2019, pag. 1-6, doi: 10.1109/ICCCN.2019.8846916,
<https://ieeexplore.ieee.org/abstract/document/8846916>
- [18] Yazici M.T., Basurra S., Gaber M.M.: ***Edge Machine learning: Enabling smart Internet of Things applications***. Big Data Cogn. Comput. 2018, 2, 26
- [19] Ghosh A.M., Grolinger K.: ***Deep Learning: Edge-Cloud Data Analytics for IoT. IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)***, Edmonton, AB, Canada, 5-8 Maggio 2019
- [20] Brian Fuller, ***AI Technology Helping Asthma Sufferers Breathe Easier***,
<http://www.hackster.io/news/ai-technology-helping-asthma-sufferers-breathe-easier-50775aa7b89f>
- [21] 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>
- [22] 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/>
- [23] Amazon Web Service. 2019: AWS Lambda@Edge,
<https://aws.amazon.com/it/lambda/edge/>
- [24] Amazon CloudFront. 2019,
<https://aws.amazon.com/cloudfront/>

- [25] Kunal Yadav: What is AWS Lambda or Serverless?
<https://hackernoon.com/what-is-aws-lambda-or-serverless-f0a006e9d56c>
- [26] 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.
- [27] 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.
- [28] Microsoft: Azure IoT Edge,
<https://docs.microsoft.com/it-it/azure/iot-edge>
- [29] 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 (GloTS), Geneva, 2017, pp. 1-6.
- [30] Familiar, B., *Microservices, IoT, and Azure: Leveraging DevOps and Microservice Architecture to Deliver SaaS Solutions*. Apress, 2015.
- [31] Internet of Things intelligente con Google Cloud IoT,
<https://www.01net.it/internet-of-things-intelligente-google-cloud-iot/>
- [32] 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.
- [33] 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.
- [34] 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.
- [35] 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).
- [36] 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.

- [37] 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.
- [38] Brian Park: AUnit,
<https://github.com/bxparks/AUnit>