Universitatea Politehnica din București
Facultatea de Automatică si Calculatoare
Departamentul de Automatică și Informatică Industrială

Rezumat Teză de Doctorat

Monitorizarea în Cloud a Clădirilor Inteligente, cu Scopul Detecției Hazardurilor

Prezentat de

Drd.Ing. Florin LĂCĂTUȘU

Supervizat de

Prof.Dr.Ing. Anca Daniela IONIŢĂ

2023

Bucharest, Romania

Abstract

Transformarea vieții oamenilor prin utilizarea clădirilor inteligente a devenit o tendință pentru ansambluri rezidențiale, campusuri universitare și corporative și complexe comerciale, unde este important să se concentreze atât asupra factorilor socio-economici, cât și a celor de mediu care pot fi facilitați de tehnologiile inteligente, inclusiv Internet of Things (IoT) și cloud computing. Interconexiunea dintre aceste două subiecte - clădiri inteligente ce utilizează dispozitivele IoT și cloud computing pentru susținerea administrării componentelor arhitecturii software - a câștigat mult interes din partea comunității științifice; beneficiază în cea mai mare parte de elasticitatea calculului și ușurința de acces pentru operatorii clădirii. Unul dintre obiectivele tezei mele este acela de a proiecta arhitectura pentru clădiri inteligente prin combinarea avantajelor oferite de o rețea edge situată în interiorul unei clădiri și cloud-ul care găzduiește aplicația de monitorizare. Rezultatul este Edge Watcher System (EWS), care a fost conceput pentru monitorizarea unui complex de clădiri inteligente și detectarea evenimentelor periculoase prin intermediul unei aplicații web native din cloud care rulează pe Kubernetes. Prin urmare, ca și contribuții la subiectele menționate mai sus, propun un nou sistem inteligent de monitorizare a clădirilor cu capabilități de detectare a hazardurilor, inclusiv proiectarea soluțiilor de arhitectură distribuite utilizate pentru evaluarea soluției propuse. Astfel, au fost prezentate patru opțiuni principale de proiectare a arhitecturii care acoperă locația software-ului de monitorizare și configurațiile rețelei locale de la marginea clădirii. Pe lângă aceasta, teza prezinta și validarea capabilităților sistemului de monitorizare a clădirii EWS prin integrarea acestuia într-o clădire reală în care au fost definite șaisprezece cazuri de testare pe baza opțiunilor de proiectare prezentate. De asemenea, un alt subiect important pe care îl acoperă teza este reprezentat de evaluarea capacităților de scalare folosind până la o mie de noduri edge simulate pentru diferite scenarii. Fiecare scenariu corespunde unui tip de clădire din lumea reală, cum ar fi un apartament mic, o casă, o clădire rezidențială mică, o clădire de birouri și un complex de clădiri în care ar fi instalat un număr diferit de noduri edge. Totodată, pentru fiecare scenariu, s-a făcut o comparație între serviciul de monitorizare EWS instalat într-un cluster Kubernetes situat în clădirea monitorizată și EWS instalat în serviciul IBM Cloud Kubernetes. Atât configurațiile de cluster Kubernetes locale, cât și cele din cloud utilizate pentru testare au avut rezultate bune în majoritatea cazurilor de testare. Totuși un cluster mai puternic decât cel testat este recomandat pentru un complex de clădiri care conține mai mult de 100 de noduri edge. Pe baza rezultatelor oferite de aceste evaluări atât cele din lumea reală, cât și evaluarea scalării sistemului, teza prezintă și recomandări pentru cea mai bună arhitectură care să fie utilizată pentru EWS.

Mulțumiri

În primul rând, aș dori să mulțumesc conducătorului meu de doctorat, doamna Profesoară Anca Daniela Ioniță pentru îndrumarea, sprijinul, răbdarea și disponibilitatea ei pe parcursul doctoratului. Dumneaei este cel mai bun mentor și îndrumător și nu aș fi putut obține aceleași rezultate fără sprijinul acordat.

De asemenea, vreau să mulțumesc comisiei mele de îndrumare, doamnei Profesoare Daniela Saru, domnului Profesor Florin Daniel Anton și doamnei Profesoare Adriana Olteanu pentru îndrumarea și comentariile ce m-au ajutat să-mi îmbunătățesc munca.

În continuare, vreau să îi mulțumesc fratelui meu, Marian Lăcătușu, și bunului meu prieten Ioan Damian, pentru colaborarea cu succes în publicarea mai multor articole în timpul studiilor doctorale.

Nu în ultimul rând, vreau să mulțumesc familiei mele pentru sprijinul și încurajarea acordată pe parcursul acestui drum.

Cuprins

Listă tabele

Listă figuri

1. Introducere

Unul dintre subiectele cele mai discutate în comunitatea științifică și industria IT este migrarea către tehnologiile cloud a soluțiilor anterioare. Implementările care utilizează tehnologii cloud au devenit mai frecvente datorită avantajelor și prevalenței lor. Această tendință de creștere este determinată de multitudinea de oferte disponibile de la diferiți furnizori de cloud. Un alt factor de influență este reprezentat de flexibilitatea oferită de cloud computing. Prin urmare, preocupările privind alegerile hardware, costurile, instalarea și întreținerea nu există în acest tip de mediu. Avantajele cloud-ului au fost exploatate în mai multe industrii, de la cazuri personale de utilizare până la procesări complexe, rezolvând probleme elaborate care necesită o cantitate semnificativă de resurse de calcul. De exemplu, Carstoiu et al. propune un caz de utilizare cloud care poate fi aplicat în domeniul medical, pentru reabilitarea pacienților neurologici [1]. Cea mai mare problemă care trebuia rezolvată în trecut a fost costul achiziției unui astfel de hardware pentru nevoia respectivă. În cloud, fiecare resursă este taxată pentru timpul în care este utilizată. Astfel, utilizarea echipamentelor performante este accesibilă unui grup mai mare de persoane.

O altă sarcină importantă în contextul dezvoltării de sisteme de monitorizare este implementarea unei rețele de senzori, cu scopul de a colecta diferiți parametri de mediu. Rețelele de senzori sunt utilizate în multe domenii, cum ar fi în domeniul medical, pentru a detecta tumorile [2]. Din ce în ce mai multe implementări vor migra către această nouă practică, datorită flexibilității oferite de aceasta și a avantajelor sale. Conceptul de închiriere hardware nu este nou. A fost folosit de mulți ani în industria IT sub formă de soluții Infrastructure as a Service (IaaS). Implementările care au aderat la această metodă au fost migrate de la serverele locale la mașinile virtuale cloud. Avantajele care au venit cu acest tip de implementare au fost strict legate de costurile hardware ale serverelor. Asadar, achizitia de servere este inlocuita cu planuri cloud care ofera posibilitatea plății pentru hardware-ul respectiv doar pentru timpul in care acesta este utilizat. Această posibilitate de a folosi hardware-ul doar pentru timpul necesar m-a fascinat întotdeauna și reprezintă unul dintre subiectele mele de interes. Încep studiul soluțiilor cloud cu proiectul meu de disertație, unde am migrat un sistem de avertizare timpurie în cloud. Cea mai recentă tendință în lumea aplicațiilor cloud este utilizarea soluțiilor Platform as a Service (PaaS). Diferența dintre aceste două tipuri de metodologii cloud (PaaS și IaaS) este faptul că, în cazul PaaS, utilizatorul este preocupat doar de implementarea aplicației. Toate sarcinile administrative ale sistemului de operare sunt executate de furnizorul de cloud. Drept urmare, acest tip de dezvoltare face cloud-ul accesibil pentru mai mulți dezvoltatori care doresc să-și migreze aplicațiile în cloud fără grijile de administrare a sistemului de operare. În teza mea de doctorat, am abordat cele mai bune metodologii de implementare a unui sistem de monitorizare a clădirii, în cloud, folosind PaaS. Din acest motiv, am studiat implementarea aplicațiilor de monitorizare folosind containere Linux cu rezultatul dezvoltării de soluții cloud-native. Fiecare furnizor important de cloud implementează în catalogul său de oferte un serviciu de containere Linux. Unul dintre cele mai populare servicii de orchestrare a containerelor este Kubernetes. A fost dezvoltat de Google și este cea mai importantă soluție de orchestrare a containerelor din domeniul IT. Cel mai mare avantaj pe care îl oferă o implementare de container este legat de izolarea aplicației și oferă linia de bază pentru implementarea microserviciilor. Această separare a componentelor aplicației în mai multe părți mici este recomandată deoarece dependența dintre ele este redusă și, dacă o componentă este offline,

celelalte vor continua să-și servească scopul. Am ales Kubernetes ca soluție viabilă pentru o aplicație de monitorizare a clădirilor, deoarece oferă o disponibilitate ridicată pentru podurile sale. Un pod este cea mai mică unitate din Kubernetes și poate conține unul sau mai multe containere. Separarea infrastructurii clădirii de software-ul său de monitorizare este un pas important în asigurarea faptului că, în caz de dezastru, acest sistem de urgență va continua să funcționeze și să trimită alerte părților care ajută.

Obiectivul cercetării mele este de a proiecta o arhitectură de clădire inteligentă, combinând avantajele oferite de o rețea de edge situată în interiorul unei clădiri și de cloud-ul care găzduiește aplicația de monitorizare. Similar cu [3], funcționalitățile principale sunt monitorizarea mai multor parametri de mediu folosind diferiți senzori, detectarea situațiilor anormale și trimiterea de notificări atât în mod centralizat, cât și în mod descentralizat. După evaluarea mai multor opțiuni arhitecturale, rezultatul este Edge Watcher System (EWS), care a fost conceput pentru monitorizarea unui complex de clădiri inteligente și detectarea evenimentelor de urgență printr-o aplicație web nativă în cloud. Scopul este de a îmbunătăți fluxul de lucru și accesul sistemelor de management într-o clădire inteligentă, prin furnizarea unei arhitecturi flexibile pentru monitorizarea și configurarea rețelei locale edge a clădirii. Sistemul a fost conceput ca o aplicație nativă în cloud, care poate fi accesată din orice locație folosind mai multe tipuri de dispozitive, cum ar fi smartphone-uri, tablete, laptop-uri etc., folosind implementarea modelului de container într-un cluster Kubernetes. Acest lucru reprezintă avantajul disponibilității ridicate a containerelor, o caracteristică necesară pentru un sistem de monitorizare a clădirii care poate trimite alerte în caz de urgență [4]. Referitor la capacitățile de configurare oferite de sistemul nostru, propun o metodă de generare a fișierelor de configurare pe baza preferințelor utilizatorului; aceste informații vor fi utilizate în continuare în configurarea nodurilor de rețea edge, pentru a automatiza procesul de configurare a acestora și comunicarea cu sistemul cloud.

Noutatea cercetării mele este legată de propunerea unui nou sistem care valorifică avantajele tehnologiilor cloud și cele mai recente metode de dezvoltare, pentru a oferi o platformă care o să fie folosită atât pentru monitorizarea, cât și pentru configurarea nodurilor de edge ale clădirii, cu scopul de a detecta diferite alerte. De asemenea, soluția mea este dezvoltată pentru a fi integrată într-un sistem mai larg compus din mai multe clădiri inteligente cu scopul de a se notifica reciproc despre posibile alerte. Cercetarea a inclus și evaluarea performanței pentru capacitatea de scalare a sistemului reprezentată de simulări corespunzătoare unui apartament mic, unei case, unei clădiri rezidențiale mici, unei clădiri de birouri și unui complex de clădiri.

În ceea ce privește structura tezei, aceasta este compusă din următoarele capitole. Capitolul 2 prezintă analiza stadiului actual, cu investigații pe două subiecte principale – Monitorizarea clădirilor pentru managementul urgențelor și cloud și edge computing. Capitolul 3 se referă la obiectivele și metoda cercetării. Capitolul 4 prezintă contribuțiile originale ale tezei mele privind monitorizarea clădirilor în cloud. În Capitolul 5 sunt discutate verificarea și validarea Edge Watcher System. Capitolul 6 reprezintă contribuțiile privind evaluarea scalării EWS. Teza se încheie cu concluziile (Capitolul 7), contribuțiile originale, perspectivele de viitor și o anexă cu detalii de implementare.

2. Analiza Stadiului Actual

În cadrul unei cercetări, un studiu de literatură oferă baza pentru fiecare investigatie în virtutea creării unei soluții personale care să le îmbunătățească pe cele actuale. Cercetarea mea se bazează pe studiul clădirilor inteligente și a modului în care pot fi implementate sisteme de

monitorizare în acest sens, cu scopul principal de semnalizare de urgență în caz de dezastru. Pe lângă temele menționate anterior, am studiat și implementarea sistemului de monitorizare specificat anterior într-un mediu cloud cu intenția de a separa infrastructura de monitorizare de clădire. Astfel, studiul acestei inițiative de cercetare a început cu investigarea diferitelor articole care oferă perspective asupra evoluțiilor actuale în monitorizarea clădirilor pentru tema managementului pericolelor, care include: monitorizarea clădirilor, managementul pericolelor și situații de urgență, și a continuat cu cloud și edge computing. subiecte, care includ cloud, IoT, edge, and fog computing și monitorizarea performanței (vezi Fig. 2.1). Acestea sunt esențiale pentru obiectivul principal al tezei, care se referă la monitorizarea clădirilor inteligente folosind cloud computing. Prin urmare, software-ul de monitorizare va fi găzduit folosind o soluție cloud care adună date de la diferite dispozitive IoT care comunică într-o rețea de edge pentru a colecta date din clădire.

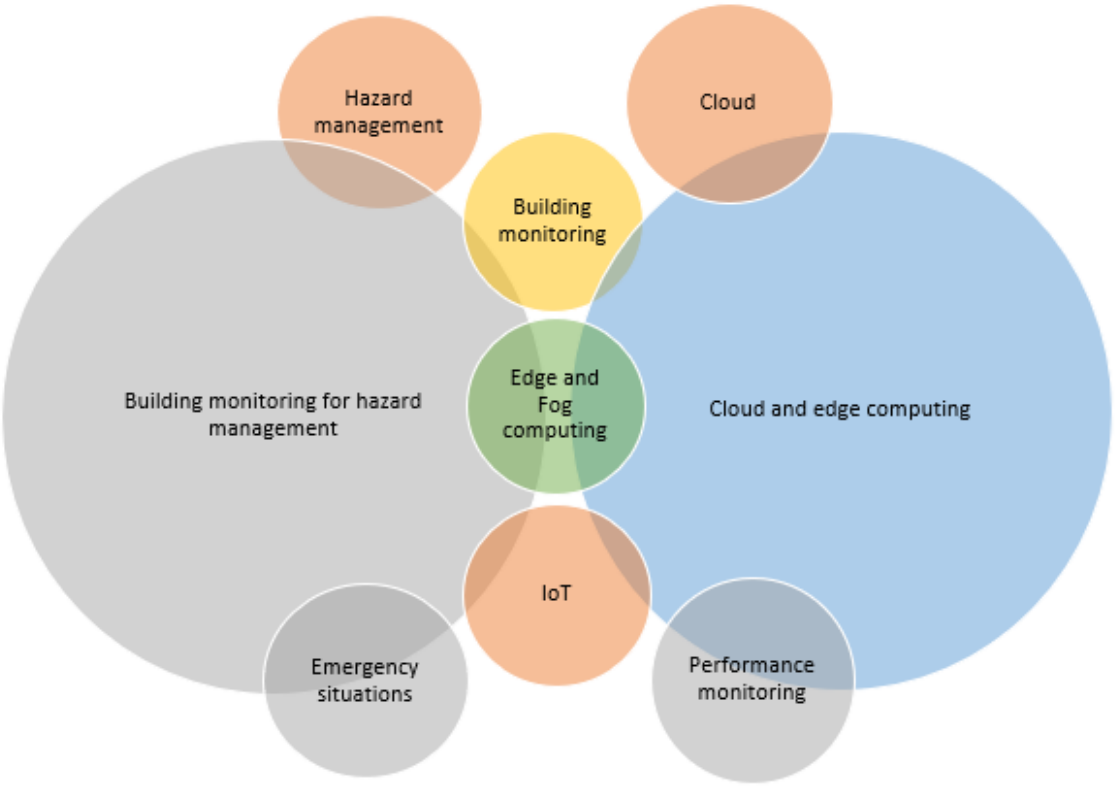

Fig. 2.1 Domenii de cercetare

3. Monitorizarea în cloud a situațiilor de urgență din clădire

3.1 Obiectivele cercetării

Obiectivele tezei mele sunt legate de studiul temei clădirilor inteligente integrate cu soluții cloud-native. Scopul este legat în principal de propunerea unei noi soluții care să fie capabilă să integreze aceste două subiecte și să poată beneficia de elasticitatea și disponibilitatea înaltă a cloud-ului. Aceste avantaje oferite de implementările cloud, împreună cu flexibilitatea dispozitivelor IoT instalate la nivel clădirii, pot oferi o soluție robustă care poate răspunde adecvat în cazul unor posibile alerte detectate la nivelul clădirii. Pe langa acestea, unul dintre cele mai importante obiective ar fi si evaluarea unui astfel de sistem de monitorizare a cladirilor in diferite situatii: de la evaluarea reala care presupune instalarea sistemului de monitorizare

intr-o cladire reala pana la evaluarea simulata de scalare care implementează sute și mii de noduri edge. Prin urmare, mai jos am rezumat principalele obiective ale tezei:

1. Nou sistem inteligent de monitorizare a clădirilor cu capabilități de detectare a evenimentelor periculoase

2. Proiectarea și evaluarea soluțiilor de arhitectură distribuită

3. Validarea sistemului de monitorizare a clădirii prin integrarea acestuia într-o clădire reală

4. Evaluarea capacităților de scalare folosind noduri de margine simulate pentru diferite scenarii

3.2 Metoda

Metoda aplicată pentru realizarea acestei cercetări este prezentată în Fig. 3.1.

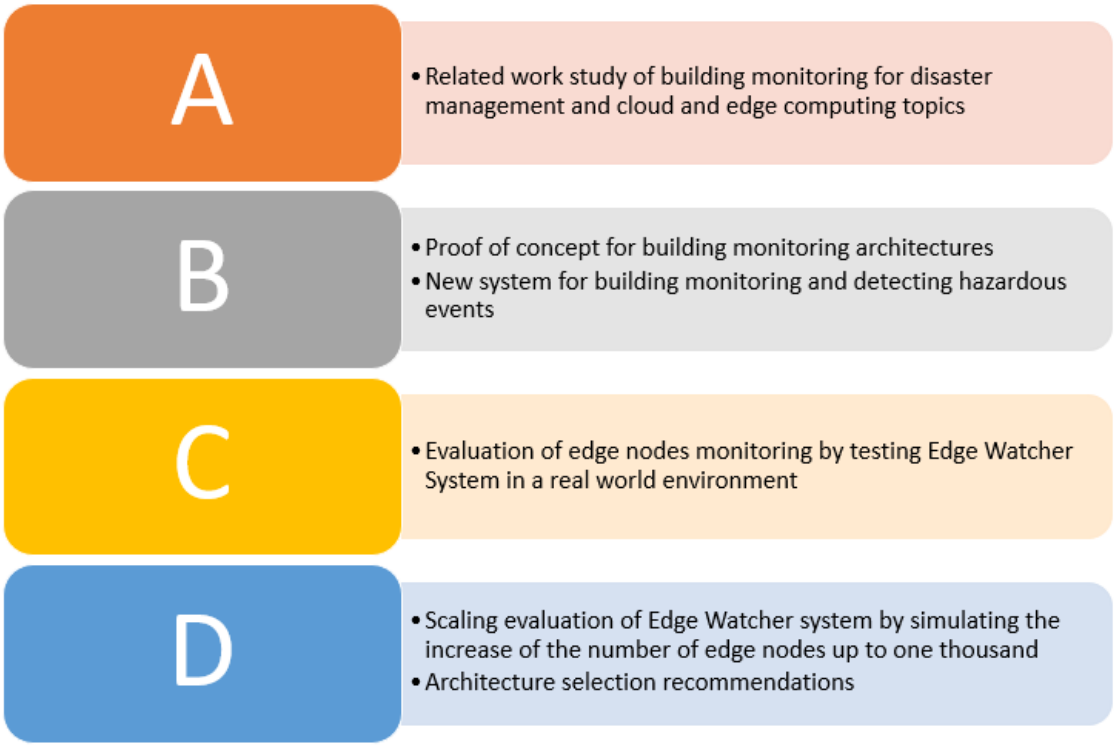

Fig. 3.1 Etapele metodei de cercetare

4. Contribuții de cercetare pentru monitorizarea clădirilor în cloud

4.1 Prototip pentru o clădire universitară

În timpul studiului meu, am creat mai întâi un prototip pentru un sistem de monitorizare și am realizat o comparație între posibilitățile multiple de implementare pentru EWS într-un mediu universitar, cu scopul de a detecta eventualele urgențe care pot apărea. Prin urmare, am făcut o comparație între două implementări ale unui sistem de monitorizare a clădirilor universitare, folosind o abordare de server local și, respectiv, o implementare nativă în cloud. Pentru ambele abordări, rezultatul a fost un sistem capabil să monitorizeze clădirea țintă și să trimită notificări. Serviciile oferite sunt utile persoanelor care pot accesa acest sistem și pot primi notificări de urgență. De asemenea, datele culese de la senzori sunt, de asemenea, esențiale, deoarece pot reprezenta evenimente care pot caracteriza situații periculoase. Următorul pas logic pentru acest tip de structură a fost implementarea unui sistem de monitorizare care poate trimite rapoarte

utilizatorilor. Scopul sistemului a fost să afișeze datele de la senzori și rapoartele utilizatorului și să le afișeze pe o pagina web. Informațiile primite sunt afișate în timp real. Sistemele constau dintr-o rețea de senzori ce colectează date, un software de monitorizare și notificare și o aplicație care colectează rapoarte de la utilizatorii săi.

Sistemul de monitorizare oferă informații despre parametrii de mediu ai clădirii universității; dacă unul dintre acești parametri iese dintr-un prag stabilit, generează alerte. Scopul aplicației de raportare este de a colecta rapoarte ale utilizatorilor în caz de urgență. Rapoartele sunt trimise prin Internet sau prin SMS. Sistemul de monitorizare colectează rapoarte de la utilizatori și le compară cu datele care provin de la senzori.

4.2 Edge Watcher System (EWS)

Monitorizarea și detectarea automată a hazardurilor sunt esențiale pentru fiecare clădire cu un număr mare de locatari și poate fi vitală în detectarea evenimentelor care pot afecta viața locuitorilor săi. Edge Watcher System este soluția pe care o propun pentru această monitorizare a clădirii, cu capacitatea de a detecta evenimente periculoase și de a anunța persoanele responsabile. Există multe tipuri de dispozitive de detectare care pot fi utilizate într-o clădire, cum ar fi senzori de mediu, monitorizare video, colectare de informații despre echipamente și senzori virtuali. Senzorii virtuali sunt reprezentați de rapoarte umane care descriu ce se întâmplă atunci când are loc un anumit eveniment. Acest tip de raport este esențial deoarece provine de la locatarii clădirii și reprezintă un aspect foarte important datorită faptului că arată starea actuală a persoanelor care locuiesc sau lucrează în prezent în clădirea monitorizată. De asemenea, în contextul fiecărui sistem de monitorizare, intrarea utilizatorului este cea mai importantă și poate avea un impact critic asupra posibilei intervenții deoarece descrie în detaliu evenimentele exacte care au loc la un moment dat într-o clădire care pot afecta eventual viețile locatarilor săi. Pe lângă raportarile umane și hardware care servesc ca parte de colectare a unui sistem de monitorizare, o parte importantă este software-ul propriu-zis de monitorizare care prelucrează datele și pe baza rezultatelor trimite alerte sau afișează datele colectate într-o formă structurată ce poate fi citită de către personalului autorizat. Într-o arhitectură de monitorizare software, mai multe servicii compun soluția, cum ar fi serviciile care rulează pe noduri și adună date de la senzori împreună cu serviciul care colectează toate datele de la noduri. De asemenea, alături de componenta software menționată anterior, o altă parte importantă a sistemului de monitorizare este software-ul dedicat ce analizează datele și afișează alerte și pagină web care este accesată de administratori cu scopul de a afișa datele procesate. Aplicatia Web reprezintă interacțiunea personalului cu diferite date care provin de la senzori. În soluția pe care o propun partea software de monitorizare este reprezentată de o aplicație Web și are două scopuri: de monitorizare și configurarea bazei de date pentru aplicație. Prin urmare, administratorii sistemului au posibilitatea de a adăuga mai multe clădiri de monitorizat împreună cu toate componentele necesare care vor fi interogate pentru date precum noduri și senzori. De asemenea, aplicația oferă configurația caracteristicilor structurale cu privire la locația fiecărui dispozitiv de detectare, cum ar fi etajul la care se află fiecare nod. Datele de monitorizare au o pagină dedicată sunt afișate statistici privind clădirea, cum ar fi numărul de etaje, noduri, senzori și utilizatori înregistrați.

4.2.1 Arhitectura EWS

Arhitectura sistemului Edge Watcher este compusă din mai multe partiții. Prin urmare, în Fig. 4.1, arhitectura EWS este prezentată prin sublinierea celor trei partiții care compun soluția: Notificari evenimente hazard, Cloud, Rețea Edge.

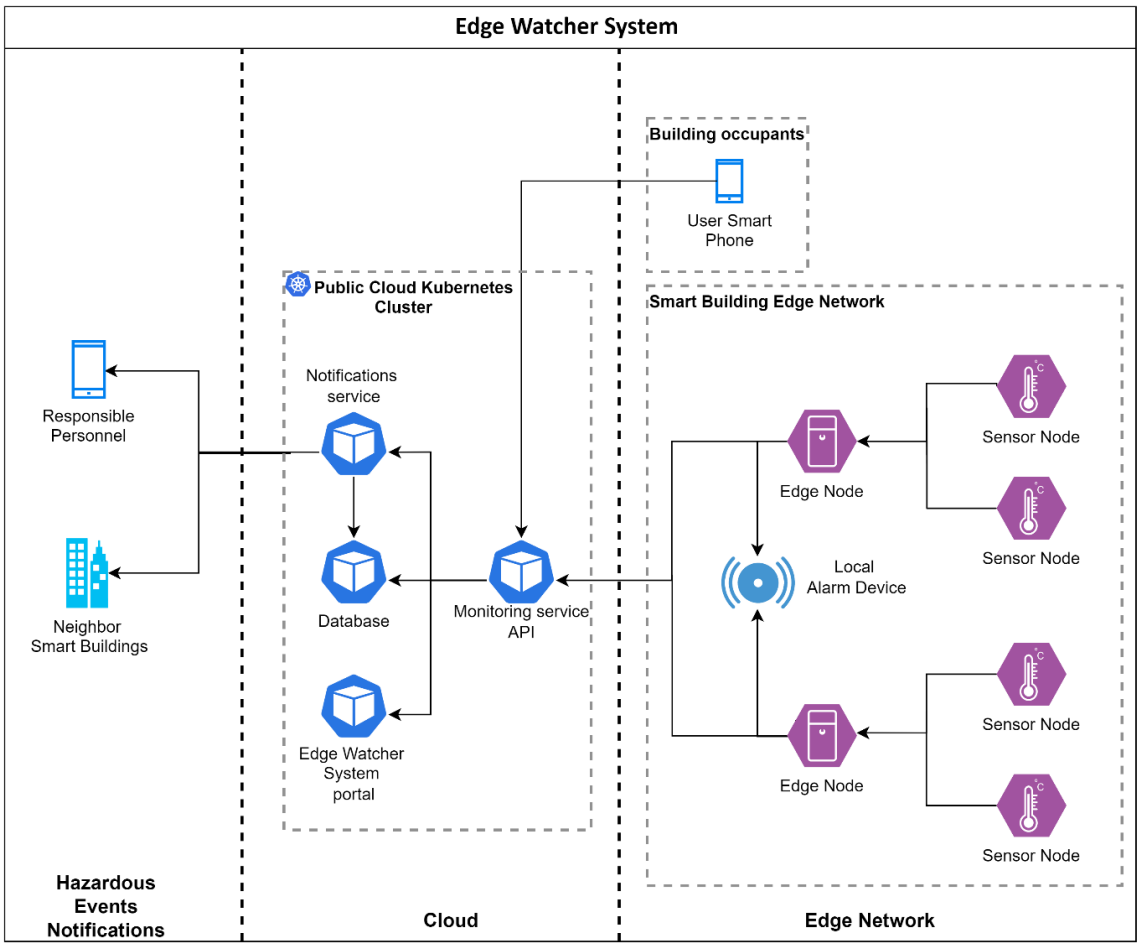

Fig. 4.1. Arhitectură logică EWS

Partiția cloud este reprezentată de implementarea bazată pe microservicii a sistemului de monitorizare software care prelucrează datele ce provin din rețeaua edge. Mai multe servicii compun acest sistem, cum ar fi serviciul de monitorizare care adună datele de la senzori împreună cu rapoartele utilizatorilor. EWS centralizează datele de la celelalte servicii și este foarte flexibil și configurabil pentru a adăuga mai multe clădiri, noduri, senzori și utilizatori. Poate fi adaptabil pentru a adăuga orice clădire și pentru a se integra cu diferite servicii și dispozitive de monitorizare. Baza de date reprezintă partea de persistență a sistemului de monitorizare în care toate configurațiile și datele sunt adăugate pentru a fi interogate de către celelalte servicii, în special de către EWS. De asemenea, partiția cloud este reprezentată de componentele software Edge Watcher System implementate pe un cluster Kubernetes situat într-un cloud public, cum ar fi Amazon Web Services, Microsoft Azure sau IBM Cloud. Arhitectura software este reprezentată de servicii slab cuplate implementate pe podurile Kubernetes [5]. Există un pod pentru fiecare dintre serviciile prezentate în Fig. 4.1. În ceea ce privește fluxul de date între componente, pot fi descrise două scenarii, care sunt reprezentative pentru cele două funcționalități principale pe care le oferă această aplicație:

(i) **Scenariul 1** - colectarea datelor din clădirea monitorizată și alertare,

(ii) **Scenariul 2** - configurarea nodurilor edge din cadrul EWS.

Software-ul EWS are patru componente principale:

(1) o interfață Web scrisă folosind Angular

(2) un backend Node.js (API-ul serviciului de monitorizare)
(3) o bază de date MySQL pentru a păstra setările utilizatorului, configurari, raportări senzori și utilizatori
(4) serviciul de notificare.

Toate acestea sunt implementate pe containere și rulează în Kubernetes. Ca rezultat, există un pod dedicat pentru fiecare dintre aceste componente. Această aplicație este flexibilă și poate rula în orice cloud cu modificări minime.

4.2.2 Algoritm de detectare a hazardurilor utilizat în testare

Algoritmul de detectare a hazardurilor a fost utilizat pentru a verifica dacă valorile senzorilor colectate au depășit un prag predefinit (configurabil). Pentru testarea performanței EWS, performanța algoritmului va fi luată în considerare folosind o bibliotecă dedicată Node.js care măsoară timpul de rulare.

Scopul algoritmului, este de a filtra datele primite de la senzori pentru a detecta o eventuală urgență (Fig. 4.2) și apoi de a transmite alerte personalului responsabil. Prin urmare, datele de mediu au fost comparate cu un prag predefinit care a fost stabilit în timpul configurării EWS pentru fiecare clădire/complex de clădiri la care a fost aplicat. Valorile critice au fost stocate în baza de date, iar alertele au fost trimise personalului responsabil pe baza acestor valori.

```
1   INPUT sensorId, sensorValue from edge node
2   CALL readDatabase with sensorId RETURNING criticalThreshold, location
3   IF sensorValue >= criticalThreshold THEN
4       CALL writeDatabase with sensorValue
5       CALL initiateAlert with location, sensorValue
6   END IF
```

Fig. 4.2 Algoritm pentru detectarea hazardurilor.

Pentru algoritm, metrica folosită a fost timpul mediu de execuție, care a fost calculat pe fișierul CSV generat folosind biblioteca Node.js în timpul execuției. Pentru fiecare dintre probele de performanta sunt prezentate doua cazuri corespunzătoare celor doua opțiuni arhitecturale.

4.2.3 Detectarea și notificarea hazardurilor

Edge Watcher System oferă capacitatea de a trimite notificări de urgență atunci când este detectat un posibil eveniment critic. Sistemul de notificare este implementat în cadrul metodei principale de detectare a EWS, în care datele sunt primite de la clădirea monitorizată țintă. Prin urmare, atunci când este detectat un eveniment critic – valoarea provenită de la senzor este egală sau depășește cea care este setată ca prag, se trimite o alertă de urgență personalului autorizat (Fig.4.3). Sistemul este implementat în așa fel încât ID-ul senzorului de unde a fost detectată urgența să fie salvat temporar pe server, astfel încât dacă senzorul detectează o nouă alertă în timpul configurat (de exemplu, 5 minute), o nouă alertă nu va fi trimisă deoarece este considerată legată de alerta care a fost deja trimisă. Acest sistem de notificare este esențial pentru un sistem de monitorizare a clădirii, deoarece detectează posibile date critice care provin de la senzorii care alertează personalul autorizat al clădirii.

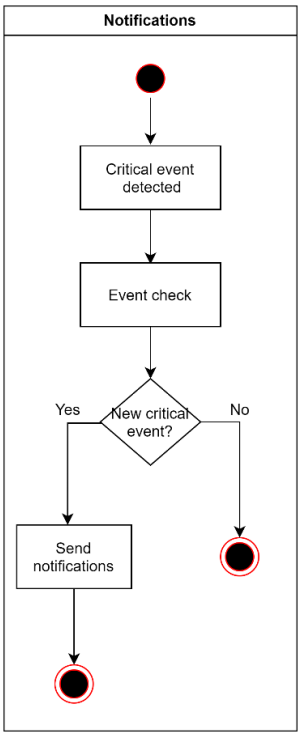

Fig. 4.3 Notificări de urgență

4.3 Evaluarea opțiunilor de proiectare

4.3.1. Opțiuni arhitecturale

Serviciile EWS se bazează pe containerizare pentru a automatiza implementarea aplicațiilor și pentru a permite o configurare ușoară pentru diferite modele de clădiri inteligente. Prin urmare, în ceea ce privește locația serviciilor de gestionare a clădirii, există două posibilități de implementare: într-un cloud public sau într-un centru de date local al clădirii.

4.3.1.1 Opțiunea arhitecturală A — Cluster Kubernetes într-un cloud public

Prima soluție analizată localizează serviciul EWS într-un cluster Kubernetes implementat într-un cloud public. Datele sunt colectate de sistemul de monitorizare cloud direct de la dispozitivele de detectare distribuite în întreaga clădire cu scopul de a detecta situațiile de urgență și de a sesiza personalul responsabil (vezi Fig. 4.4). Avantajul implementării la distanță a clusterului Kubernetes este legat de principiul separării sistemului de monitorizare de clădirea țintă monitorizată. În acest caz, sistemul de monitorizare a clădirii nu ar fi afectat de diferitele întreruperi care pot apărea atunci când apare o urgență. Cu toate acestea, există și alte aspecte importante, cum ar fi costurile de întreținere și costurile inițiale de achiziție a hardware-ului, care sunt zero cu această opțiune de cloud public. Există, într-adevăr, costuri de utilizare care, în cele din urmă, sunt mai mici în comparație cu cele necesare achiziției și întreținerii unui mic datacenter, inclusiv personalului implicat în aceste operațiuni. Cu toate acestea, un dezavantaj care se aplică unui sistem care utilizează această abordare este dependența sistemului de monitorizare de o conexiune la Internet fiabilă pentru trimiterea datelor către cloud. Această problemă poate fi atenuată cu ușurință prin furnizarea de conectivitate de rezervă în cazul în care conexiunea principală la Internet nu este disponibilă prin cuplarea sistemului cu conectivitate la Internet mobil, care ar trebui să fie prezentă pe fiecare nod de edge/nod senzor inteligent, pentru a trimite în mod independent date către sistemul de monitorizare, dacă alte opțiuni de Internet nu sunt disponibile.

În concluzie, principalul avantaj care susține această opțiune de proiectare este că sistemul de monitorizare nu depinde de resursele clădirii pentru a funcționa, iar costurile de întreținere a unui datacenter local sunt eliminate. Ca un dezavantaj, datacenter-ul cloud este situat mai departe de clădire, iar solicitările de la dispozitivele de detectare durează mai mult pentru a fi îndeplinite.

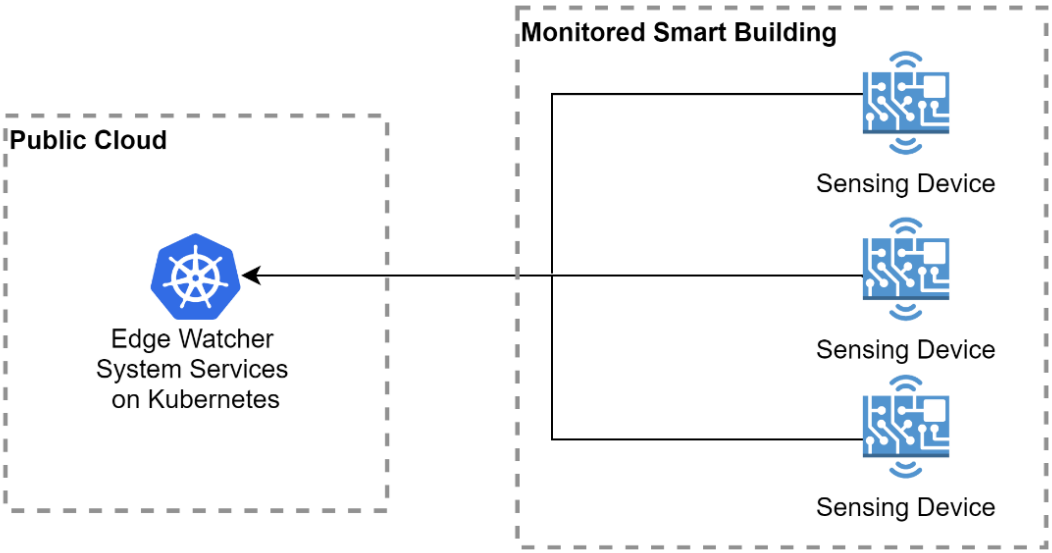

Fig. 4.4. EWS cu un cluster Kubernetes în cloud public

4.3.1.2. Opțiunea arhitecturală B — Cluster Kubernetes într-un datacenter local

În a doua soluție, datacenter-ul local implică instalarea sistemului de monitorizare pe hardware-ul situat în interiorul clădirii (Fig. 4.5). Avantajul acestei abordări este o comunicare mai rapidă între senzori și sistemul de monitorizare. Comunicarea în rețeaua locală este mai rapidă decât în cea care funcționează prin Internet. Acest lucru este cuplat cu faptul că sistemul nu depinde de a avea o conexiune fiabilă la Internet pentru a trimite date de mediu către sistemul de monitorizare. Marele dezavantaj care vine cu implementarea acestei soluții este dependența sistemului de monitorizare de rețeaua electrică a clădirii. Când există o problemă cu sistemul electric, sistemul de monitorizare nu poate fi ținut online. Această problemă se aplică numai hardware-ului centrului de date. Dispozitivele de detectare formate din noduri de margine și senzori sunt compuse din dispozitive de putere redusă care pot funcționa pe o baterie o perioadă foarte lungă de timp, furnizând datele necesare din mediul clădirii.

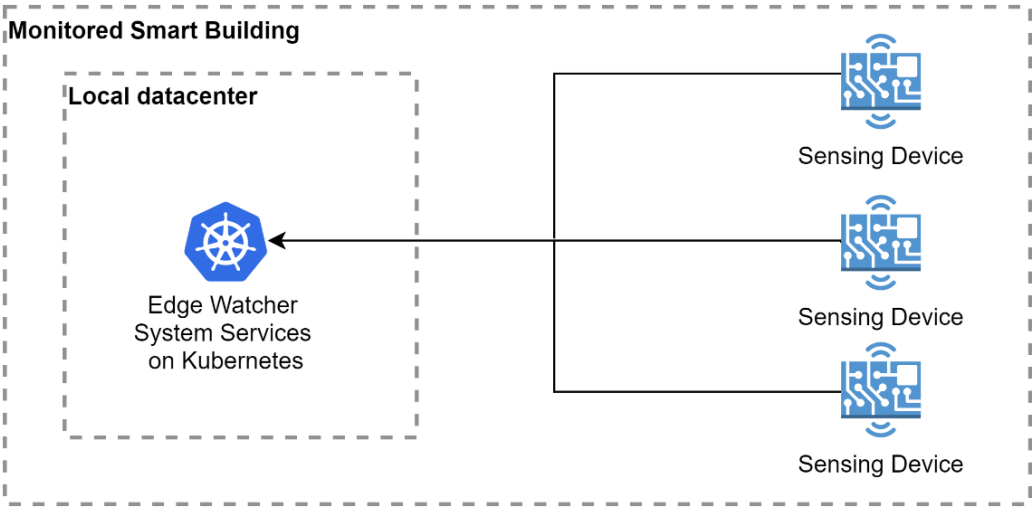

Fig. 4.5. EWS cu un cluster Kubernetes intr-un datacenter local

În comparație cu opțiunea locației din prima metodă, cu această abordare, principalul avantaj este perioada de latență mai mică în transmiterea datelor între rețeaua edge locală și clusterul local. Principalul dezavantaj este că, dacă are loc un eveniment critic, sistemul de monitorizare poate fi și el afectat, deoarece este situat în aceeași unitate.

4.3.2 Opțiuni pentru dispozitive de detectare

Secțiunea anterioară a analizat opțiunile de proiectare bazate pe locația sistemului de monitorizare, și anume, cloud public și implementări ale datacenter-ului local. Alte aspecte importante privind topologia edge, opțiunile pentru dispozitivele de detectare și modul în care aceste dispozitive sunt conectate la cloud și utilizate pentru a colecta date pentru sistemul de monitorizare sunt discutate ulterior.

4.3.2.1. Optiunea Edge A — Noduri Edge

Prima opțiune de proiectare luată în considerare pentru dispozitivele de detectare se bazează pe o arhitectură compusă din noduri edge bazate pe microprocesoare care adună date de la noduri senzor wireless, de putere redusă, bazate pe microcontrolere (Fig. 4.6). Scopul nodurilor edge este de a aduna date de la mai multe dispozitive de detectare și de a le trimite către sistemul de monitorizare cloud. Această abordare poate fi instalată în orice clădire pentru a monitoriza parametrii de mediu. Nodurile de margine sunt conectate la Internet și pot funcționa și pe o baterie pentru perioade de timp mai scurte în comparație cu dispozitivele cu senzor.

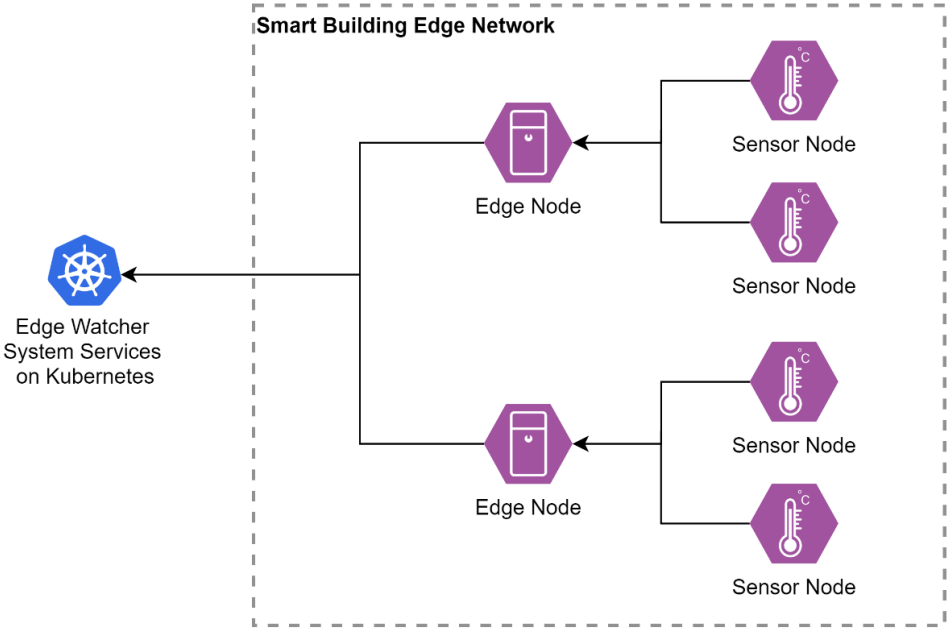

Fig. 4.6. Proiectare EWS cu noduri edge.

4.3.2.2. Opțiunea Edge B — Dispozitive de detectare edge.

A doua opțiune de proiectare luată în considerare pentru implementarea dispozitivelor de detectare a fost conectarea dispozitivelor edge de detectare direct la cloud (Fig. 4.7). Acest sistem este similar cu nodul edge prezentat mai devreme, dar senzorii sunt conectați printr-o conexiune fizică la nod. Dispozitivele edge de detectare se bazează pe microprocesoare, iar numărul de senzori ar fi apropiat de cel al nodurilor de senzori bazate pe microcontroler din alegerea anterioară de proiectare. Deoarece microprocesoarele sunt mai scumpe decât microcontrolerele și consumă mai multă energie, această abordare ar fi mult mai costisitoare decât cea anterioară, oferind în același timp aceeași funcționalitate în acest caz de utilizare.

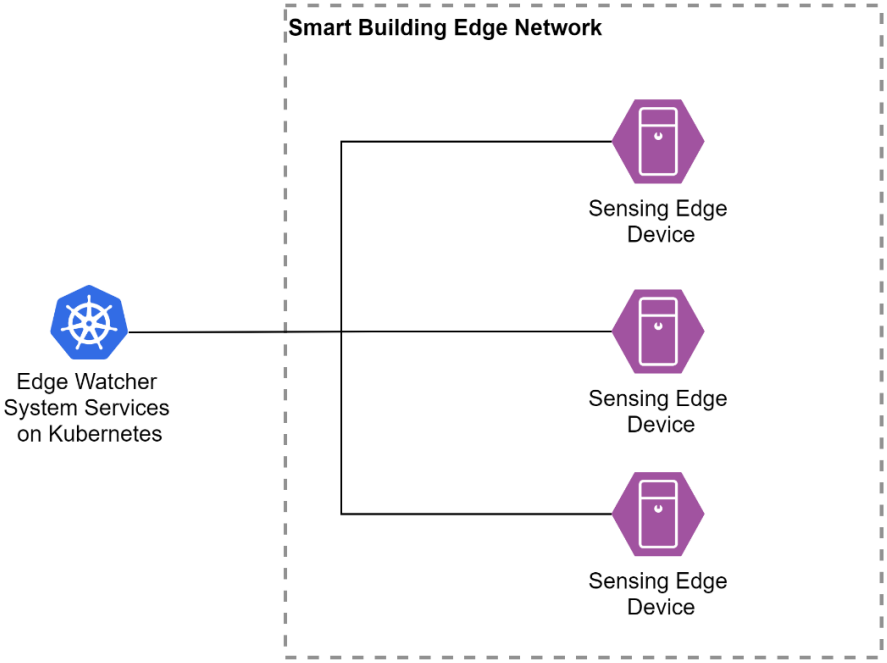

Fig. 4.7. Proiectare EWS cu dispositive edge de detectare.

5. Evaluare și validare pentru o clădire universitară

Acest capitol prezintă experimentele de validare pentru Edge Watcher System. Verificarea și validarea sunt foarte importante deoarece sistemul este testat cu diferite scenarii și oferă cea mai bună vedere despre cum funcționează și se comportă în mediul clădirii. Acest capitol are două părți principale: experimentul realizat pentru o implementare EWS cu un singur nod și un experiment de scenariu real în care EWS a fost testat într-o clădire Universitara.

5.1 Experiment de monitorizare a clădirilor și de detectare a hazardurilor

5.1.1 Prezentare generală a experimentului

În ceea ce privește opțiunile dispozitivelor de detectare, acest capitol descrie experimentele efectuate pentru cele două alternative prezentate în 4.3.1: Edge Nodes și Sensing Edge devices. Pentru a face acest lucru, am creat două teste diferite.

EWS cu un singur nod se bazează pe implementarea unui singur nod edge care va colecta date de la senzori sau de la un dispozitiv bazat pe microcontroler, cum ar fi Node MCU. Mai jos sunt prezentate două opțiuni.

5.1.2 Opțiunea Edge A. Implementare cu noduri edge

Această opțiune de arhitectură a fost descrisă mai detaliat în Capitolul 4 și implică integrarea a două dispozitive, un nod de margine și un nod senzor bazat pe microcontroler. Prin urmare, pentru acest experiment am implementat următorul sistem care a fost compus din:

- Raspberry Pi 3 ca nod edge (MQTT Broker și Subscriber)
- NodeMCU – nod senzor și MQTT publisher
- Senzorul BMP 180– folosit ca exemplu pentru a obține date din mediu
- Cluster local Kubernetes implementat în aceeași rețea virtuală

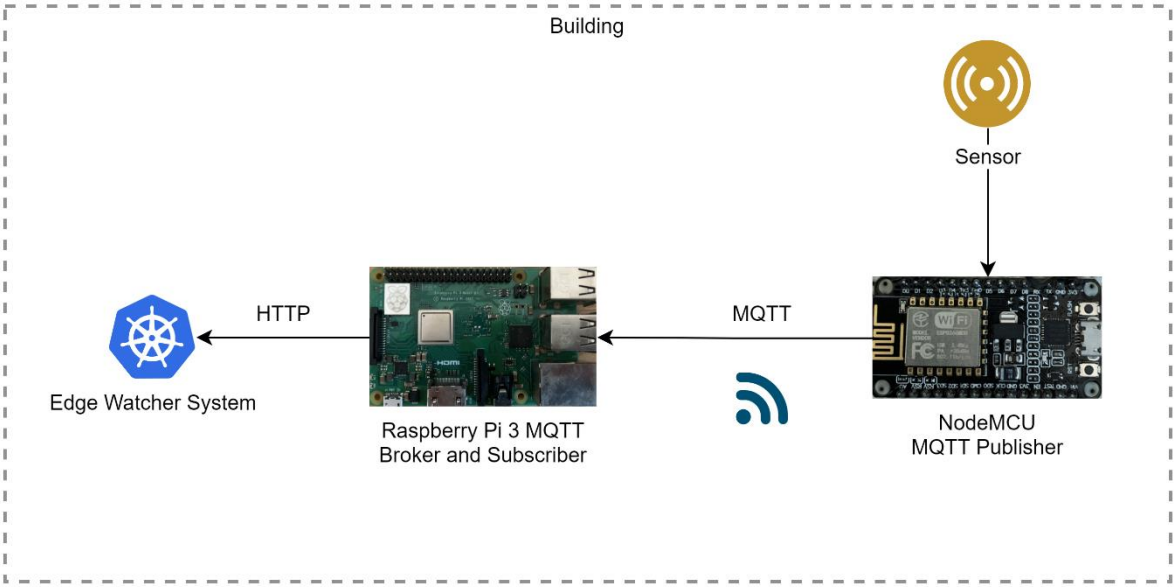

Fig. 5.1 Raspberry Pi nod edge și Node MCU nod senzor

Scenariul acestui experiment este următorul: Nodul bazat pe microcontroler NodeMCU adună date de la senzorul de temperatură BMP 180. După aceasta, datele sunt publicate folosind MQTT în nodul edge Raspberry Pi 3. Rolul acestui nod broker este de broker și subscriber

MQTT. Prin urmare, datele sunt trimise către acest nod de la una sau mai multe plăci NodeMCU. Acest nod de edge acționează și ca un subscriber MQTT. Prin urmare, se abonează și la subiectele în care editorul NodeMCU trimite datele senzorului ca mesaj JSON. În cele din urmă, datele sunt analizate pe nodul edge și sunt trimise printr-un apel HTTP POST către Edge Watcher System. Mesajul care vine de la nodurile senzorului conține atât valoarea detectată, cât și id-ul senzorului. Astfel, Edge Watcher System va cunoaște identitatea și locația senzorului pentru a afișa valorile pe Dashboard sau pentru a alerta personalul responsabil dacă valoarea detectată este critică.

5.1.2.3 Evaluarea performanței EWS pentru Opțiunea Edge A

Pentru testul de analiză a performanței, am măsurat timpul de răspuns și timpul de rulare al algoritmului de detectare a hazardurilor pentru 50 de apeluri de la nodul edge la EWS. Fiecare apel a fost efectuat la fiecare 5 secunde când datele au fost primite de la nodul senzor NodeMCU. În tabelul de mai jos, există rezultatul testelor. Așadar, pentru cele 50 de apeluri efectuate, timpul de răspuns a fost între 27 și 54 ms cu un răspuns mediu de 32,25 ms. Timpul de rulare al algoritmului de decizie este de 11,27 ms.

Table 5.1 Opțiunea A - analiza performanței

Noduri Edge	**Apeluri**	**Interval între apeluri (s)**	**Timp de răspuns mediu (ms)**	**Min**	**Max**	**Median**	**95th percentile**	**Timp rulare (ms)**
1	50	5	32.25404	27.24	53.801	30.924	37.52415	11.27152

5.1.3 Opțiunea Edge B. Implementarea dispozitivului edge de detectare

Această opțiune arhitecturală implică faptul că un senzor este conectat fizic la nodul de margine Fig. 5.2. Prin urmare, pentru acest experiment am implementat următorul sistem care a fost compus din:

- Raspberry Pi 3 ca dispozitiv de detectare
- Senzor BMP 180 – folosit ca exemplu pentru a obține date din mediu
- Cluster local Kubernetes implementat în aceeași rețea virtuală

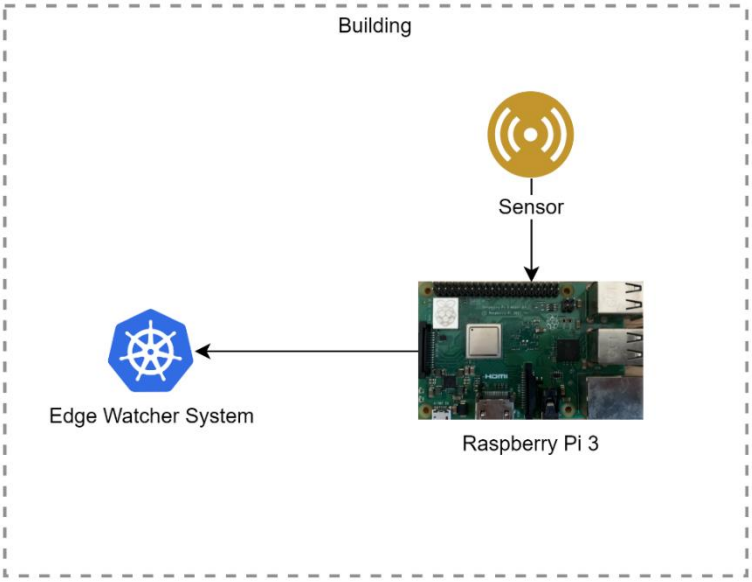

Fig. 5.2 Diagrama nodului senzor Raspberry PI

Scenariul acestui experiment este următorul: dispozitivul Raspberry Pi 3 de detectare va colecta citiri de la senzorul de temperatură BMP 180. La nivelul EWS, dispozitivul edge este configurat pentru a-și înregistra locația, pragul critic. EWS va înregistra numai valorile care trec de pragul configurat.

5.1.3.1 Evaluarea performanței EWS pentru Opțiunea Edge B

Pentru testarea acestui scenariu, senzorul este conectat direct la dispozitivul edge. Prin urmare, au fost executate 50 de apeluri de la acesta către EWS. Astfel, pe EWS, valorile înregistrate sunt comparate cu pragul selectat din baza de date. Dacă valoarea depășește pragul, datele sunt adăugate în baza de date. Timpul mediu de răspuns este de 37,93 ms, iar timpul de rulare al algoritmului este de aproximativ 12 ms. Aceste valori înregistrate arată o performanță bună pentru sarcina curentă (Tabelul 5.2).

Table 5.2 Opțiunea B - analiza performanței

Noduri Edge	**Apeluri**	**Interval între apeluri (s)**	**Timp de răspuns mediu (ms)**	**Min**	**Max**	**Median**	**95th percentile**	**Timp rulare (ms)**
1	50	5	37.93	24.6	155.2	30.06	91.06	12.10

5.1.4 Discuţie

Pentru EWS cu un singur nod, am prezentat un experiment de bază din lumea reală pentru EWS. Tema principală a acestui capitol a fost testarea performanței sistemului cu date care provin dintr-o rețea edge fizică. Pentru testele efectuate în această lucrare, implementarea locală a EWS a primit apeluri din două configurații ale rețelei edge: Opțiunea edge A și B. În cazul Opțiunii A, arhitectura este compusă dintr-un dispozitiv edge și un nod senzor. Datele sunt colectate de nodul senzor și publicate prin MQTT către broker (dispozitiv edge). Dispozitivul edge este atât un broker MQTT, cât și un subscriber. Când un mesaj nou este publicat, subscriber-ul trimite valoarea înregistrată către EWS prin HTTP. EWS compară valoarea cu un prag predefinit și trimite o alertă dacă este necesar. Pentru Opțiunea edge B, arhitectura este compusă dintr-un dispozitiv edge de detectare care are o conexiune fizică la un senzor. După ce datele sunt colectate, acestea sunt trimise la EWS unde sunt procesate în același mod în care a fost descris anterior.

După cum s-a observat din experiment, rezultatele de performanță în cazul unei implementări cu un singur nod, au fost foarte asemănătoare pentru cele două opțiuni, ambele înregistrând un timp mediu de răspuns sub 40 ms. Avantajul Opțiunii B este reprezentat doar de simplitatea implementării și este recomandat pentru un apartament mic unde nu este necesar un număr limitat de senzori. Opțiunea edge A este mai complexă de implementat, dar este mai ieftină în comparație cu Opțiunea edge B, unde vor fi necesari mai mulți senzori.

5.2 Testarea performanței pentru mai multe opțiuni de proiectare

Testarea performanței într-un mediu real este esențială pentru a valida faptul că un sistem funcționează la parametrii necesari. Prin urmare, trebuie implementate mai multe scenarii care să testeze funcționarea unui sistem. Pentru teza mea, am realizat mai multe scenarii de testare în clădirea Universității Politehnica București „Precis”.

5.2.1. Scenarii de testare

Testele au constat în principal în implementarea rețelei edge fizice în cadrul acestei clădiri și crearea mai multor configurații pentru testare. Prin urmare, testul include setările pentru rețeaua edge și, de asemenea, pentru locația Edge Watcher System, care a fost instalat pe o mașină la nivelul clădirii și, de asemenea, în IBM Cloud. Un alt set de scenarii a inclus tipul de dispozitiv edge care a fost folosit pentru a forma rețeaua edge. Rolul rețelei edge este de a culege date de mediu din clădire și de a le trimite către sistemul de monitorizare. În cadrul EWS, de îndată ce datele sunt primite de la rețeaua edge a clădirii, acestea sunt comparate cu o valoare de prag stabilită anterior de administrator. Pentru acest caz, există două scenarii au fost acoperite în cadrul testelor.

Primul scenariu este despre implementarea algoritmului de detectare a hazardurilor pe nodul edge. În acest caz, Raspberry Pi, atunci când primește datele de la senzor, le compară cu valoarea de prag care a fost setată anterior de un administrator.

Al doilea scenariu care a fost testat se referă la numărul de noduri edge care au fost instalate în zona țintă. În consecință, am efectuat teste atât cu un nod edge, cât și cu două configurații de nod edge. Pentru testul cu două noduri edge, au fost folosite două dispozitive Raspberry Pi pentru a trimite datele senzorului către EWS. De asemenea, pe tema rețelei edge, un alt tip de scenariu care a fost realizat a fost implementarea a două opțiuni de arhitectură: una în care senzorul a fost conectat direct la nodul edge printr-o conexiune fizică și alta în care a fost introdus un nou tip de dispozitiv edge: Node MCU. În cazul unei clădiri mari, precum Universitatea Politehnica din București „Precis”, construirea abordarii cu dispozitive bazate pe microcontrolere poate fi mai potrivită, deoarece costă mai puțin de implementat.

5.2.2 Testarea EWS

Obiectivul experimentului meu este de a testa aplicația de monitorizare a clădirii Edge Watcher System față de configurații multiple de dispozitive edge. Aceste dispozitive ar urma să fie instalate apoi într-o clădire a universității pentru a furniza datele necesare pentru detectarea eventualelor evenimente periculoase. Pentru testul planificat am identificat mai multe configurații de arhitectură care pot fi integrate într-un mediu de clădire și bazate pe Opțiunile edge A și B și Opțiunile arhitecturale A și B.

5.2.2.1. Opțiuni de proiectare

Arhitecturile pe care le propun pentru acest test de performanță sunt clasificate în două categorii care vor fi prezentate în detaliu mai jos.

1. Opțiuni de proiectare pentru scenariul 1

Cea mai importantă caracteristică a acestei opțiuni de arhitectură este reprezentată de un senzor care este conectat fizic la un nod edge (Edge Opțiunea B). O altă clasificare ar fi legată de locația sistemului de monitorizare care poate fi implementat local (Opțiunea de proiectare 1.1 - Fig.5.3) sau în cloud (Opțiunea de proiectare 1.2 - Fig. 5.4) pe baza Opțiunii de arhitectură B și A.

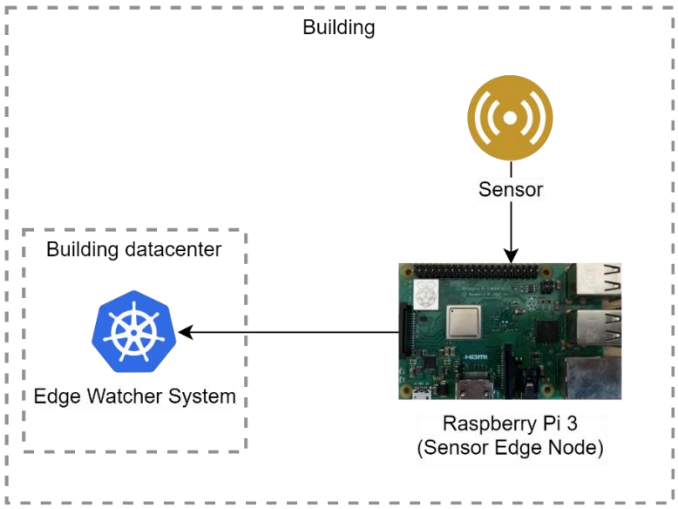

Fig. 5.3 Opțiunea de proiectare 1.1 Sensor Edge Node, EWS implementat local

În imaginea de mai jos este reprezentat EWS-ul implementat în cloud care primește datele despre mediul clădirii de la nodul edge legat la senzor. Datele sunt trimise de la nodul edge, care în testele noastre este reprezentat de o placă Raspberry Pi 3, sunt trimise prin HTTP către sistemul de monitorizare.

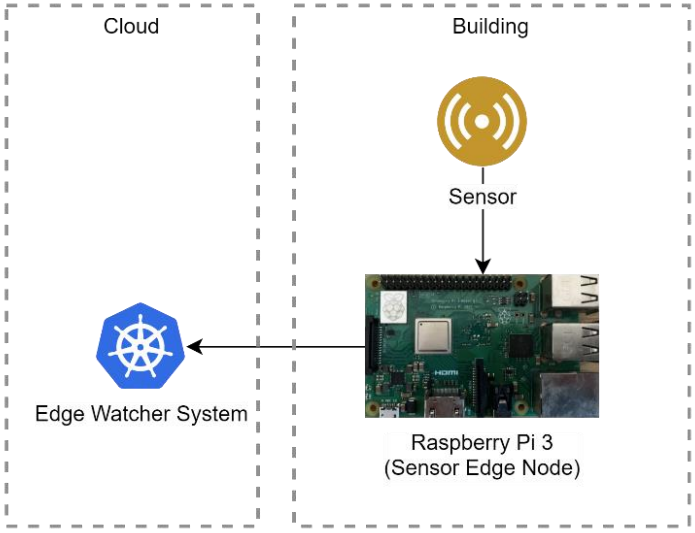

Fig. 5.4 Opțiunea de proiectare 1.2 - Sensor Edge Node, EWS implementat în cloud

2. Opțiuni de proiectare pentru scenariul 2

Principala diferență care apare este introducerea unui nou dispozitiv hardware în cadrul rețelei edge – un nod senzor dedicat (Opțiunea edge A). Acest nod senzor, care este reprezentat de o placă bazată pe microcontroler - Node MCU colectează datele de la senzori și le publică prin MQTT către nodul edge Raspberry Pi 3, care acționează atât ca broker, cât și ca subscriber. După ce datele sunt primite pe nodul edge, acestea sunt apoi trimise către sistemul de monitorizare prin HTTP.

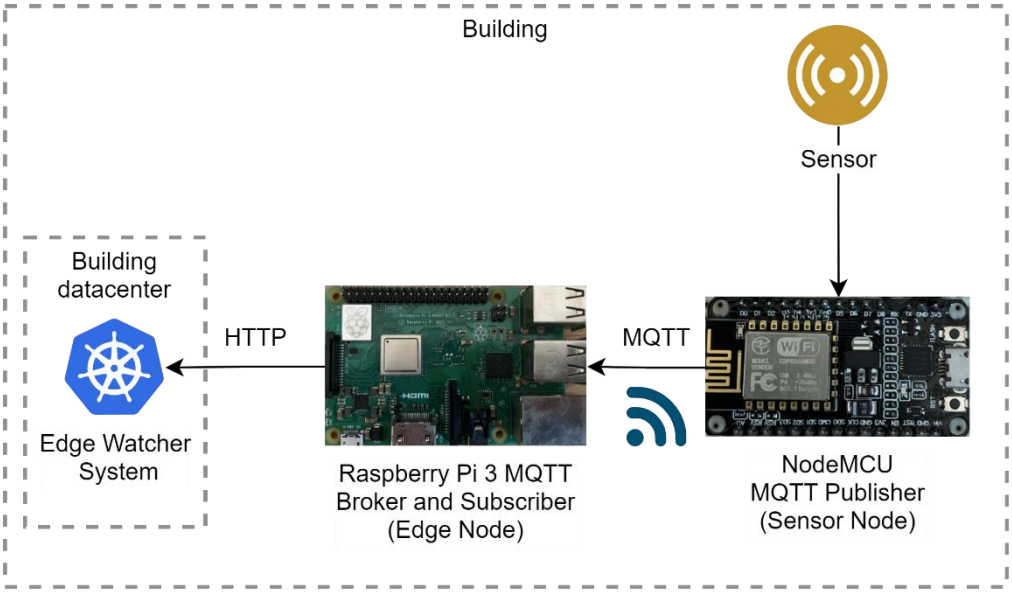

Fig. 5.5 Opțiunea de proiectare 2.1 – nod edge MQTT broker cu nod senzor, EWS implementat local

Ca și în cazul opțiunii de proiectare 1, există două configurații disponibile – sistem de monitorizare implementat local (Fig. 5.5) și implementat în cloud (Fig. 5.6).

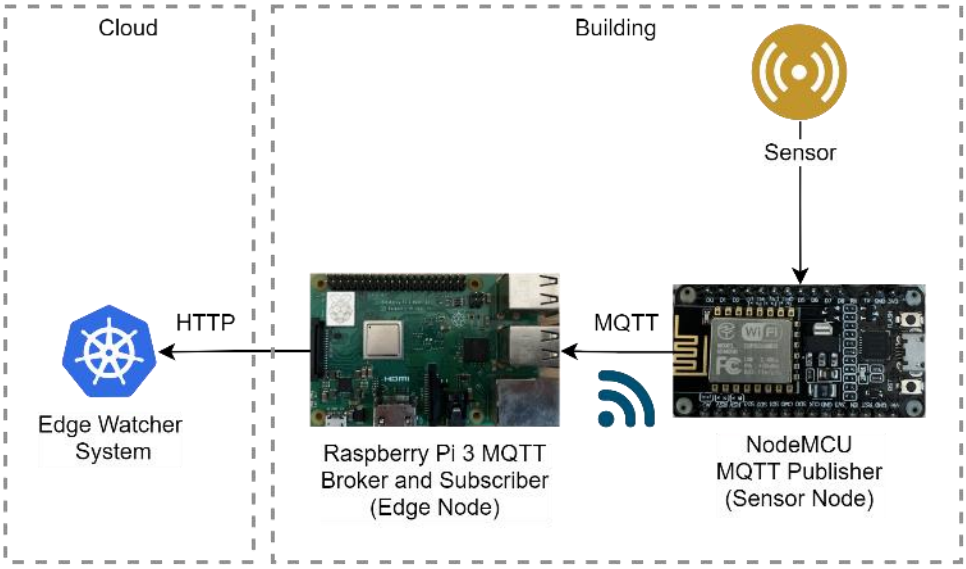

Fig. 5.6 Opțiunea de proiectare 2.2 – nod edge MQTT broker cu nod senzor, EWS implementat în cloud

3. Lista de opțiuni de proiectare testate

În total au fost definite șaisprezece cazuri de testare, pornind de la opțiunile de proiectare prezentate mai sus. Lista este următoarea:

Opțiunea de proiectare 1

• Opțiunea de proiectare 1.1

1.1.1. Un nod edge senzor, un senzor, algoritm de detectare a hazardurilor pe nodul edge

1.1.2. Un nod edge senzor, un senzor, algoritm de detectare a hazardurilor pe EWS

1.1.3. Două noduri edge senzor, un senzor pe nod, algoritm de detectare a hazardurilor pe EWS
1.1.4. . Două noduri edge senzor, un senzor pe nod, algoritm de detectare a hazardurilor pe nodul edge

• Opțiunea de proiectare 1.2
1.2.1. Un nod edge senzor, un senzor, algoritm de detectare a hazardurilor pe nodul edge
1.2.2. Un nod edge senzor, un senzor, algoritm de detectare a hazardurilor pe EWS
1.2.3. Două noduri edge senzor, un senzor pe nod, algoritm de detectare a hazardurilor pe EWS
1.2.4. Două noduri edge senzor, un senzor pe nod, algoritm de detectare a hazardurilor pe nodul edge

Opțiune de proiectare 2

• Opțiunea de proiectare 2.1
2.1.1. Un nod edge, un nod senzor, un senzor, algoritm de detectare a hazardurilor pe nodul edge
2.1.2. Un nod edge, un nod senzor, un senzor, algoritm de detectare a e hazardurilor pe EWS
2.1.3. Două noduri edge, un nod senzor, un senzor pe nod, algoritm de detectare a hazardurilor pe EWS
2.1.4. Două noduri de edge, un nod de senzor, un senzor per nod, algoritm de detectare a hazardurilor pe nodul edge
• Opțiunea de proiectare 2.2
2.2.1. Un nod edge, un nod senzor, un senzor pe nod, algoritm de detectare a hazardurilor pe nodul edge
2.2.2. Un nod edge, un nod senzor, un senzor pe nod, algoritm de detectare a hazardurilor pe EWS
2.2.3. Două noduri edge senzor, un senzor pe nod, algoritm de detectare a hazardurilor pe EWS
2.2.4. Două noduri edge senzor, un senzor pe nod, algoritm de detectare a hazardurilor pe nodul edge

5.2.2.2. Detalii de implementare

Testele au fost efectuate în clădirea Universității „Politehnica” din București „PRECIS” unde am implementat cele patru arhitecturi prezentate în capitolele precedente. În cadrul testelor au fost definite mai multe scenarii care includ și variațiile numărului de noduri edge. Pe lângă acestea, algoritmul de detectare a hazardurilor, un algoritm simplu folosit pentru a stabili dacă există o urgență a fost implementat fie pe nod, fie în cloud, în funcție de scenariu.

Hardware used:

1. Opțiunea de proiectare 1

- Nodul edge senzor: Raspberry Pi 3
- Senzor: Senzor de temperatură BMP 180
- Cluster local: Cluster Kubernetes plasat în rețeaua locală a clădirii
- Cloud Cluster: IBM Cloud Kubernetes Cluster

2. Opțiunea de proiectare 2:

- Edge Node: Raspberry Pi 3

- Nod senzor: Node MCU
- Senzor: Senzor de temperatură BMP 180
- Cluster local: Cluster Kubernetes plasat în rețeaua locală a clădirii
- Cloud Cluster: IBM Cloud Kubernetes Cluster

5.2.2.3 Procedura de testare

Scopul testelor de performanță este de a demonstra capacitatea EWS de a funcționa într-un scenariu din lumea reală. Prin urmare, pentru a testa performanța EWS, am evaluat apelurile de HTTP care au fost efectuate către acesta prin diferite posibilități de configurare a rețelei edge. Prin urmare, pe nodurile edge, a fost măsurat timpul de răspuns de la apelul HTTP pentru inserarea datelor în EWS. De obicei, timpul de răspuns HTTP este influențat de locația serverului, încărcarea și procesarea care se face cu fiecare apel. În ceea ce privește procesarea, în scenariile de testare am inclus posibilitatea în care codul care implementează algoritmul de decizie mică a fost rulat pe server și alte scenarii în care a rulat pe nodul edge..

5.2.3 Rezultate

Rezultatele au fost obținute pentru fiecare dintre opțiunile de proiectare prezentate. Prin urmare, în tabelul de mai jos vă prezint timpul mediu de răspuns pentru apelurile executate de la nodurile edge către EWS. Pentru scenariile prezentate în capitolul anterior, există unele în care au fost utilizate două noduri edge. Astfel, în tabelul de mai jos timpul mediu de răspuns, E2 (Edge Node 2) va apărea numai pentru scenariile în care au fost implementate două noduri edge. Pentru fiecare scenariu au fost efectuate 50 de apeluri de la nodul edge către EWS pentru a trimite datele de mediu care au fost colectate de la Clădirea UPB „PRECIS”. EWS este implementat în două locații, IBM Cloud (locația din Milano) și în rețeaua clădirii, prin urmare rezultatele vor diferi și în funcție de locația sistemului de monitorizare. Ultima coloană arată unde rulează algoritmul de decizie – pe nodul de margine sau în EWS. Algoritmul de detectare a evenimentelor periculoase compară valoarea critică care a fost setată de administratorul clădirii în timpul configurării inițiale a senzorului cu valoarea reală care a fost colectată din mediu. Dacă valoarea este critică, atunci datele sunt introduse în baza de date EWS și, de asemenea, trimite o notificare personalului autorizat al clădirii.

Table 5.3 Rezultate

Opțiuni proiectare	**Timp de răspuns mediu E1 (ms)**	**Timp de răspuns mediu E2 (ms)**	**Locație EWS**	**Algoritm detecție hazarduri**
1.1.1	64.21662	-	Building	Edge Node
1.1.2	67.37904	-	Building	EWS
1.1.3	78.00704	92.02102	Building	EWS
1.1.4	76.4661	89.14806	Building	Edge Node
1.2.1	196.2808	-	IBM Cloud	Edge Node
1.2.2	207.4164	-	IBM Cloud	EWS
1.2.3	259.2715	222.0848	IBM Cloud	EWS
1.2.4	242.1902	222.6137	IBM Cloud	Edge Node
2.1.1	60.96322	-	Building	Edge Node
2.1.2	85.6524	-	Building	EWS
2.1.3	69.35771	92.97539	Building	EWS

2.1.4	65.2284	80.05818	Building	Edge Node
2.2.1	210.3509	-	IBM Cloud	Edge Node
2.2.2	213.7616	-	IBM Cloud	EWS
2.2.3	196.2808	222.6137	IBM Cloud	EWS
2.2.4	214.7974	220.3171	IBM Cloud	Edge Node

Edge Node 2 are mai puține valori, deoarece a fost testat doar în scenariile cu două noduri edge. Ceea ce este comun pentru ambele noduri este că au înregistrat valori sub 300 ms, ceea ce indică o performanță adecvată pentru un sistem de monitorizare.

6. Evaluarea scalării și discuții

6.1 Descriere

Evaluarea performanței este foarte importantă în edge și cloud computing și cu atât mai mult pentru utilizarea acestor sisteme în managementul hazardurilor. Aceasta necesită teste multiple de „fiabilitate, interoperabilitate și scalabilitate” sub o anumită sarcină de lucru [6]. Haseeb-Ur-Rehman și colab. a dezvoltat o taxonomie de nor de senzori care acoperă aspectele legate de rețea, comunicare, managementul datelor, arhitectură, eterogenitate și securitate [7].

6.2 Evaluarea scalării

Testarea performanței este o parte esențială în dezvoltarea unei aplicații, deoarece oferă mijloacele pentru a afla dacă sistemul testat rulează pe parametrii doriti cu intrări diferite. Importanța acestui pas critic se bazează în principal pe ideea că fiecare piesă nouă de dezvoltare trebuie testată pe diferite scenarii predefinite în care sunt monitorizate diferite metrici. În cazul studiului meu, care are ca obiectiv cercetarea sistemelor de monitorizare a clădirilor în cloud, există multiple capabilități ale acestui sistem care trebuie testate și validate în raport cu un plan de testare predefinit. Unele dintre aceste capabilități care vor fi detaliate în continuare sunt legate de monitorizarea performanței acestei soluții. Un obiectiv pe care îl propun pentru această parte este acela de a simula diverse configurații de noduri edge care efectuează diferite solicitări HTTP către aplicația cloud. Unele dintre aceste solicitări sunt reprezentate de datele care sunt trimise către EWS pentru a fi analizate în continuare. Un alt aspect important al acestui test este că, cu această configurație, configurațiile reale ale nodurilor edge ale clădirii sunt simulate, clădirile mai mari necesită mai multe noduri edge pentru a acoperi o zonă mai mare. Testul va fi realizat cu scenarii pentru apartamente mici, o casă, clădire rezidențială mică, clădiri de birouri și un complex de clădiri împreună cu o comparație detaliată a rapoartelor generate pentru toate aceste cazuri. Un alt aspect care va fi luat în considerare cu testele menționate anterior este comparația dintre soluția găzduită în cloud și un cluster local. Prin urmare, aceleași teste vor fi simulate pentru aplicația care este implementată local și pentru implementarea cloud pe un serviciu bazat pe containere, cum ar fi un cluster Kubernetes. Această comparație va arăta în ce măsură implementarea cloud va afecta timpii de răspuns pentru aplicație în comparație cu aceeași soluție implementată pe aceeași rețea cu nodurile edge. Următorul test pe care îl propun pentru această soluție este evaluarea performanței pentru algoritmii de detectare a evenimentelor periculoase care sunt utilizați pentru a detecta dacă datele colectate sunt critice sau nu. Pentru acest test, o parte foarte esențială care ar trebui menționată este importanța implementării cloud și cât de repede ar rula algoritmii pe o soluție de container bazată pe cloud, în comparație cu aceeași implementare pe un cluster Kubernetes local. Un alt factor care este foarte important pentru acest test, este performanța algoritmului de decizie față de un număr

mare de solicitări reprezentate de o rețea de noduri edge mai mare situată într-un complex de clădiri.

Rezultatele testelor care au fost prezentate mai sus reprezintă, de asemenea, o parte importantă a procesului de validare care va atesta că soluția de cloud cercetată atinge unele metrici de performanță critice pentru funcționarea optimă a unui astfel de sistem.

6.2.1 Evaluarea performanței opțiunilor de arhitectură containerizată

Pe baza opțiunilor de proiectare prezentate în Capitolul 4.3, prezint metoda utilizată pentru a determina performanța sistemului cu mai multe dispozitive de detectare simulate edge. Testele au comparat performanțele celor două opțiuni de proiectare arhitecturală: (A) cluster-ul Kubernetes din cloud-ul public; și (B) cluster-ul local Kubernetes. Acestea au fost conectate la dispozitive de detectare prin noduri edge pe baza primei opțiuni din capitolul 4.3. Scopul acestei comparații a fost de a obține o vedere detaliată a cerințelor de performanță necesare pentru sistemul cloud atunci când datele sunt primite de la mai multe noduri. Fiecare test efectuat a corespuns unei configurații reale pentru un tip de clădire, cu nevoi specifice privind numărul de noduri edge și senzori necesari.

6.2.2 Scenarii de testare

Pentru ambele opțiuni arhitecturale, preocuparea mea a fost să identific scenarii de testare bazate pe configurații de exemple de clădiri din lumea reală, de la un apartament mic până la un întreg complex de clădiri, cum ar fi o universitate sau un campus corporativ. Prin urmare, testele de performanță au fost executate pentru serviciile EWS găzduite în medii containerizate urmând scenariile descrise mai jos. Am selectat exemple inspirate din clasificarea gradului de ocupare și definițiile date în Codul Internațional al Construcțiilor [8] și din cele zece clase de clădiri stabilite în [9].

- **Apartament mic.** Acest scenariu se referă la o unitate individuală dintr-o clădire rezidențială. Am luat în considerare configurația pentru un apartament mic cu două camere și un nod edge. În acest scenariu, nodul de margine trebuie să colecteze date de mediu de la senzori pentru a detecta mișcarea, temperatura și contactul atunci când ușa se deschide sau se închide. În scopuri de testare, un apel API este simulat de la nodul edge la EWS, ceea ce are ca rezultat adăugarea unei noi citiri în baza de date. Algoritmul de detectare a hazardurilor compară fiecare valoare cu un prag predefinit pentru a verifica dacă ar trebui luată în considerare o alertă. Scenariul este la fel de aplicabil pentru un magazin dintr-un centru comercial dacă monitorizarea senzorului este realizată separat de chiriașul magazinului.
- **Casă.** Al doilea scenariu a vizat o clădire rezidențială de sine stătătoare. Am luat în considerare exemplul unei case decomandate cu cinci camere și două etaje. În acest scenariu, un nod edge este instalat pe fiecare etaj pentru a colecta date de mediu și a le trimite către EWS. Diferența față de primul scenariu este în utilizarea a două noduri edge.
- **Clădire rezidențială mică.** Scenariul clădirii mici încapsulează un total de cinci noduri edge simulate instalate pe fiecare etaj al clădirii. Fiecare nod primește date de mediu transmise de senzori instalați în spațiul public și în interiorul apartamentelor individuale situate la același etaj. Monitorizarea și sesizările evenimentelor periculoase sunt gestionate pentru întreaga clădire de către personalul responsabil.
- **Cladire de birouri.** În acest scenariu, am simulat cazul unei clădiri cu 10 etaje și 20 de noduri edge. În acest scenariu, două noduri edge sunt instalate pe fiecare etaj pentru a primi date de mediu de la senzori și a le trimite la EWS pentru procesare ulterioară. Am

luat în considerare acest exemplu de clădire nerezidențială folosită în scop profesional sau comercial deoarece acest tip de clădire este folosit mai des ca clădire inteligentă; prin urmare, poate profita de serviciile de detectare și alertare a evenimentelor periculoase, cum ar fi cele luate în considerare în studiul nostru.

- **Un complex de clădiri**. Am luat în considerare un scenariu la scară mai mare, care corespunde unui grup de clădiri inteligente asociate cu utilizare rezidențială, de afaceri sau instituțională. Acestea pot corespunde unui centru comercial, unei universități, unui campus corporativ sau unui complex rezidențial. Acestea pot ocupa o zonă mai mică sau mai mare, cu senzori multipli utilizați pentru a măsura datele de mediu conectate la EWS prin mai multe noduri edge. Am luat în considerare mai întâi trei cazuri pentru efectuarea testelor:
    - 50 de noduri de edge – împrăștiate în jurul mai multor clădiri care cuprind complexul
    - 100 de noduri edge — pentru a testa capacitatea de a lucra sub o sarcină mare prin înregistrarea unui număr mare de puncte de date de mediu trimise într-perioadă scurtă de timp
    - 1000 de noduri edge — pentru a testa limitele configurației clusterului și capacitatea de a simula 1000 de solicitări trimise sistemului; cererile sunt trimise imediat către server, iar acesta trebuie să abordeze fiecare cerere imediat ce cea anterioară a fost îndeplinită.

Apoi, am executat și teste pentru noduri edge între 100 și 1000 cu pași de 100.

6.2.3 Setări teste de performanță

Pentru EWS, testarea performanței a fost efectuată folosind Apache JMeter versiunea 5.4.1, pentru a simula solicitările de la nodurile edge pentru a verifica dacă acestea au fost adresate fără eroare și dacă timpul de răspuns a fost suficient de mic. Mașina folosită pentru a rula testele JMeter conține un procesor Intel i7 cu 4 nuclee, cuplat cu 8 GB de RAM.

Au fost dezvoltate cazuri de testare pentru diferite dimensiuni, începând de la un apartament mic și terminând cu un complex de clădiri. Configurarea a constat în crearea de fire de execuție pentru a simula grupuri de noduri edge și executarea solicitărilor către API-ului de raportare EWS. În ceea ce privește configurația JMeter, au fost trei parametri importanți de configurat:

- Numărul de fire de execuție: numărul de noduri edge utilizate pentru a trimite date de mediu către aplicație
- Ramp-up period: timpul necesar pentru a ajunge la numărul complet de fire
- Loop count: numărul de teste care trebuie executate.

Un alt obiectiv pentru testarea performanței, pe lângă apelurile HTTP JMeter este testarea efectivă a algoritmului de decizie care verifică dacă valoarea senzorului colectată depășește pragul predefinit. Pentru testarea performanței algoritmului de decizie, am folosit o bibliotecă dedicată Node.js care măsoară timpul de rulare a diferitelor funcții. Numele bibliotecii este „execution-time” și poate fi instalat din managerul de pachete node (npm).

6.2.4 Configurarea mediului containerizat

Pentru a testa opțiunile de arhitectură containerizate pentru EWS, am implementat două configurații, corespunzătoare analizei prezentate în Capitolul 4.3.:

(A) Clusterul IBM Cloud Kubernetes a fost implementat pentru opțiunea de proiectare a clusterului cloud public Kubernetes cu următoarele detalii tehnice:

- Implementare în datacenterele IBM Cloud
- Cluster Kubernetes cu un singur nod, cu două nuclee și 4 GB de RAM (nivel gratuit implicit)
- Versiunea Kubernetes: 1.21.7 (implicit).

(B) Clusterul local Docker Desktop a fost implementat pentru opțiunea de proiectare a clusterului Kubernetes pentru datacenter-ul local cu următoarele detalii tehnice:

- Implementat pe o mașină Windows cu procesor Intel i7 cu 4 nuclee, cuplat cu 8 GB RAM- Windows Subsystem for Linux (WSL) 2
- Docker Desktop WSL 2 versiunea backend 4.1.1 cu Kubernetes
- Memoria și CPU sunt alocate dinamic pentru a îmbunătăți consumul de resurse
- Kubernetes versiunea 1.21.5 (implicit).

6.2.5 Metrici de performanță

Apache JMeter face posibilă obtinerea unui rezultat sub formă de pagină HTML care oferă mai mulți parametri ce sunt legați de timpul de răspuns al cererii testate. Atributele care sunt relevante pentru studiul meu sunt erorile de execuție, timpul de răspuns și debitul (pentru a determina performanța sistemului). Acestea sunt asociate cu următoarele valori:

- Error % (for the execution)
- Average response time
- Minimum response time
- Maximum response time
- Median response time
- Percentiles
- Transactions/s (for the throughput)

6.3 Evaluarea scalării - rezultate

În această secțiune, rezultatele evaluării scalarii sunt prezentate și discutate. Pentru fiecare dintre cele două configurații de mediu containerizat (adică, clusterul IBM Cloud Kubernetes și clusterul local Docker Desktop), au fost executate mai multe teste de performanță pentru a simula scenariile relevante identificate în Capitolul 6.2. Rezultatele monitorizate corespund metricilor obținute pentru apelurile HTTP JMeter, simulând informațiile adunate de la un număr de noduri edge plus timpul de rulare a algoritmului de decizie pentru detectarea evenimentelor periculoase, care a fost măsurat cu o bibliotecă Node.js, așa cum este explicat în Capitolul 4.2. Astfel, testarea performanței a fost executată atât pentru implementarea clusterului local, cât și a celui public. JMeter poate fi folosit și pentru a testa performanța aplicațiilor de servicii web, cum ar fi cele de la [10], [11], [12].

Astfel, am generat 30 de rapoarte JMeter, fiecare conținând o multitudine de grafice și date. Pentru a le analiza comparativ, câteva rezultate importante sunt rezumate în Tabelul 6.1 pentru clusterul IBM Cloud Kubernetes și în Tabelul 6.2 pentru clusterul local Docker Desktop.

Table 6.1. Rezultatele testelor pentru clusterul IBM Cloud Kubernetes.

Test	**Executions**		**Response Time (ms)**					**Throughput**	**Hazardous Event Detection Algorithm**
Scenario	Samples (Edge Nodes)	Error (%)	Average	Min	Max	Median	95th Percentile	Transactions /s	Run time (ms)

Small apartment	1	0	389	389	389	389	389	2.57	3.52
House	2	0	375.5	371	380	375.5	380	5.26	4.42
Small residential building	5	0	380	371	385	382	385	12.89	3.49
Office building	20	0	405.75	374	437	405.5	436.8	43.8	21.8
A complex of buildings	50	0	472.84	391	560	461	553.45	81.04	167.43
	100	0	574	376	748	574	729	115.74	144.42
	200	0	556.83	375	815	563.50	789.00	203.87	278.9
	300	0	906.49	446	1336	947.50	1309.95	196.34	564.4
	400	0	981.00	380	1461	936.00	1412.85	236.27	554.4
	500	0	1393.26	420	2090	1394.50	2014.85	212.04	560.3
	600	0	1614.98	391	2282	1602.00	2225.95	223.46	846.57
	700	0	1584.66	374	2373	1620.00	2288.90	246.05	448.3
	800	0	1832.57	398	2759	1855.0	2651.95	250.16	677
	900	0	2026.60	499	3056	1994.0	2954.85	230.00	1036.5
	1000	0	1992	384	3538	1927	3359	189.9	939.89

Table 6.2. Rezultatele testelor pentru clusterul local Docker Desktop.

Test	**Executions**		**Response Time (ms)**					**Throughput**	**Hazardous Event Detection Algorithm**
Scenario	Samples (Edge Nodes)	Error (%)	Average	Min	Max	Median	95th Percentile	Transactions /s	Run time (ms)
Small apartment	1	0	9	9	9	9	9	111.11	3.22
House	2	0	11	9	13	11	13	153.85	3.59
Small residential building	5	0	24	16	32	23	32	151.52	19.62
Office building	20	0	68.2	31	109	69.5	107.7	176.99	57.3
A complex of buildings	50	0	228.44	97	288	239	285.8	130.89	155.5
	100	0	322.33	78	460	311	447.9	175.75	288.86
	200	0	543.18	138	978.6	553.7	930.7	176.3	447.3
	300	0	768.05	147	1390	769.3	1322.1	179.2	574.5
	400	0	995.9	164	1821.6	1002.5	1741.1	179.92	762.5
	500	0	1316	181	2378.3	1318.3	2293.3	177.93	1030.9
	600	0	1597.8	360	2855.6	1515.6	2737.8	175.79	1211.2
	700	0	2296.3	426	3262.4	2094.1	3151.5	164.04	1881.5
	800	0	2589.2	363	3698.7	2515.5	3560.75	173.05	2232.5
	900	0	2849.8	147	3994.4	2842.3	3757.9	162.14	2588.6
	1000	0	3098	243	4431	2914	4284	171.59	2975.23

De asemenea, prezint rezultatele în grafice comparative pentru a arăta diferența referitoare la timpul mediu de răspuns pentru cele două opțiuni de proiectare arhitecturală: clusterul IBM Cloud Kubernetes și clusterul local Docker Desktop. Fig. 6.1 prezintă timpii de răspuns față de numărul de noduri edge corespunzător primelor patru scenarii: un nod edge pentru apartamentul mic, două noduri edge pentru casă, cinci noduri edge pentru clădirea rezidențială mică și 20 de noduri edge pentru cladirea de birouri. Fig. 6.2 reprezintă timpii de răspuns pentru un complex de clădiri față de numărul de noduri edge, variind de la 50 la 1000. Se poate observa, astfel, influența descentralizării orchestrației containerului. Fig. 6.3 și Fig. 6.4 ilustrează comparații similare pentru timpul de rulare al algoritmului de decizie pentru detectarea hazardurilor.

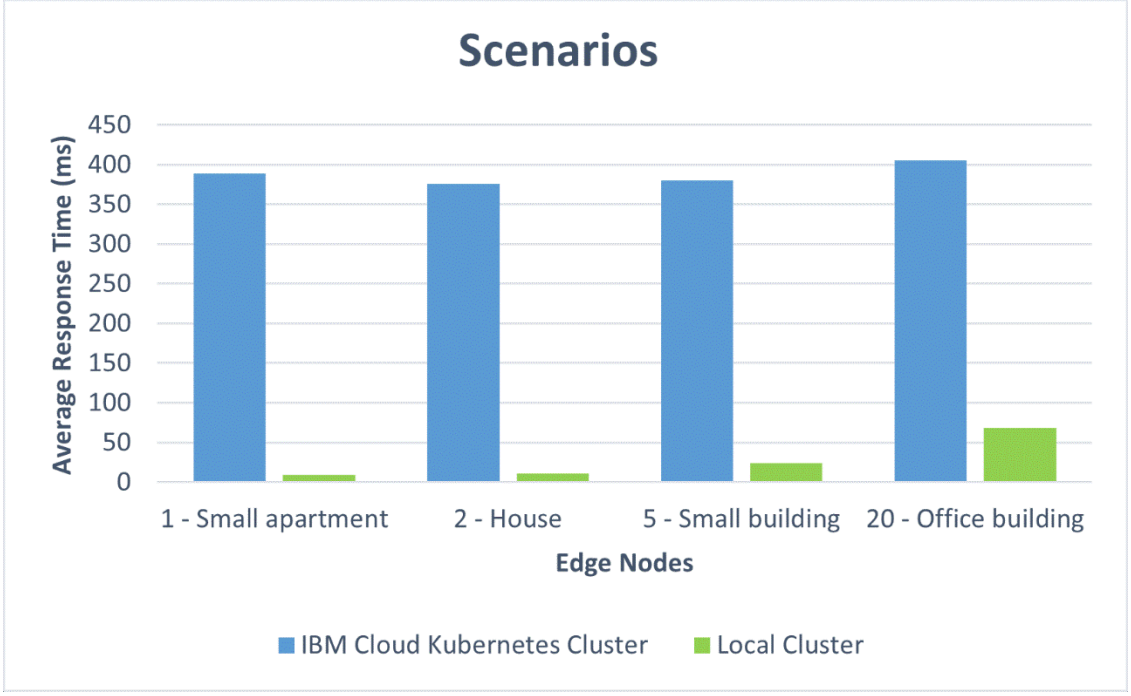

Fig. 6.1 Timp de răspuns comparativ pentru diferite scenarii

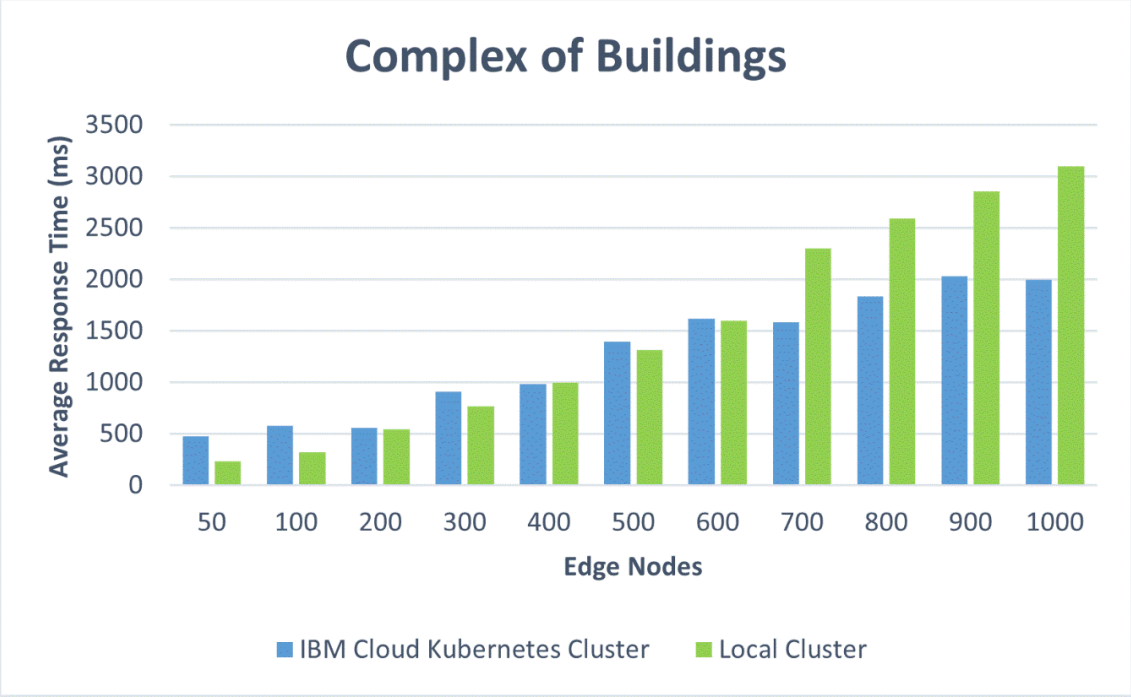

Fig. 6.2 Timp de răspuns comparativ pentru complexul de clădiri.

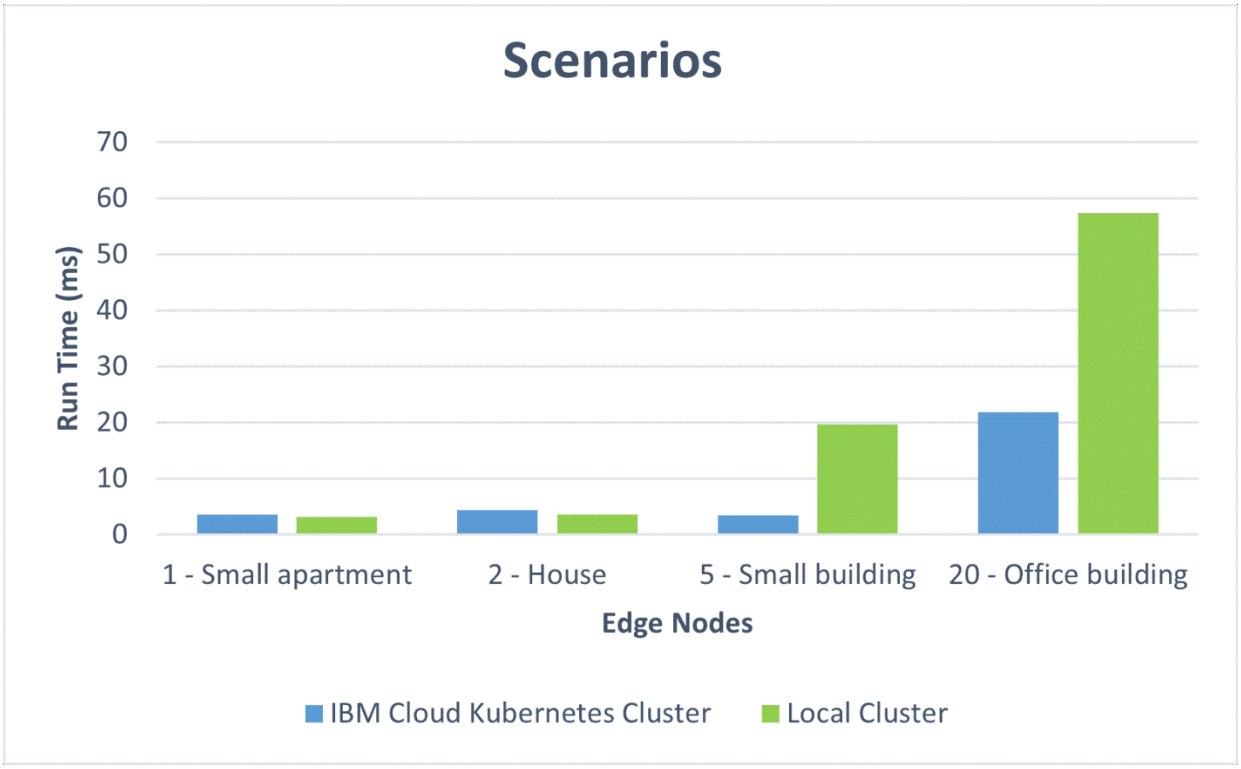

Fig. 6.3. Timpi de rulare comparați pentru diferite scenarii.

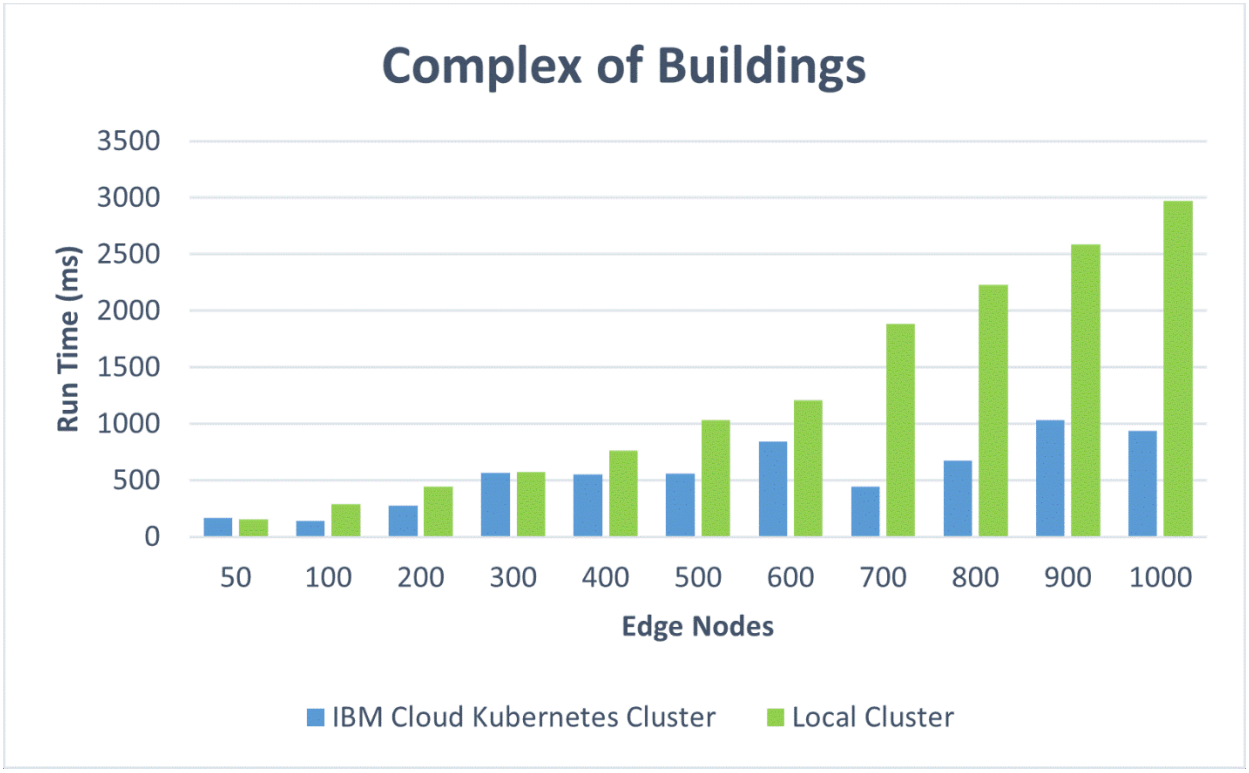

Fig. 6.4. Timpi de rulare comparați pentru complexul de clădiri.

Am observat că clusterul local Docker Desktop a funcționat mai rapid pentru primele patru scenarii cu până la 20 de noduri edge ca urmare a locației sale. Cu toate acestea, clusterul IBM Cloud Kubernetes a funcționat foarte bine și pentru primele patru scenarii, păstrând un timp mediu de răspuns sub 500 ms [13].

Clusterul IBM Cloud Kubernetes (corespunzător opțiunii arhitecturale A) a funcționat mai bine decât clusterul local Docker Desktop (opțiunea arhitecturală B) pe cele mai solicitante cazuri de testare, de la 700 la 1000 de noduri edge (cu aproape o secundă mai rapid). Opțiunea A a fost reprezentată de un cluster dedicat cloud, care era mai puternic decât Opțiunea B, care rulează pe o stație de lucru. Cu toate acestea, în cazurile cu mai mult de 100 de noduri edge, ambele au avut rezultate mai slabe decât limita de 500 ms, care este de obicei folosită pentru aplicațiile web. Ca o concluzie a acestor teste, într-un mediu de producție, se recomandă un cluster mai puternic pentru a oferi performanțe satisfăcătoare pentru cazurile cu mai mult de 100 de noduri edge.

6.4 Discuție și lecții învățate

6.4.1. Compararea performanței pe baza scenariilor

Această secțiune discută rezultatele obținute pentru implementarea celor două opțiuni arhitecturale pentru fiecare dintre opțiunile prezentate în Capitolul 4.3. De asemenea, prezint aceste rezultate în [16].

Apartament mic. Pentru un nod edge, diferențele care pot fi observate în timpii de răspuns (Fig. 6.1) se datorează în principal faptului că clusterul local Docker Desktop este implementat într-un mediu local și latența este mai mică în comparație cu clusterul IBM Cloud Kubernetes înființat. Chiar dacă au existat diferențe mari între timpii de răspuns respectivi, rezultatele pentru implementarea în cloud public încă se încadrează sub un timp de răspuns acceptabil, adică sub 500 ms. Timpii de rulare ai algoritmului de detectare a urgențelor au fost similare.

Casă. Pentru configurațiile cu două noduri edge, citirile nu au fost foarte diferite de cele obținute cu abordarea cu 1 nod edge (din scenariul apartamentului mic). Rezultatele pentru ambele opțiuni (cluster local Docker Desktop și cluster IBM Cloud Kubernetes) au fost foarte asemănătoare cu cele obținute pentru scenariul anterior, oferind o performanță bună.

Clădire rezidențială mică. Odată cu creșterea numărului de noduri edge testate, majoritatea modificărilor legate de timpul de răspuns au fost înregistrate pe clusterul local Docker Desktop (Fig. 6.3). Cea mai notabilă diferență și cauza acestui răspuns mai lent a fost legată de timpul de rulare al algoritmului de detectare a urgențelor, care a crescut semnificativ în cazul clusterului local Docker Desktop. Pentru clusterul IBM Cloud Kubernetes, valorile au rămas similare cu cele obținute cu abordarea cu două noduri edge.

Cladire de birouri. După creșterea numărului la 20 de noduri edge, timpii de răspuns au crescut pentru ambele abordări testate: clusterul local Docker Desktop și clusterul IBM Cloud Kubernetes. Cu toate acestea, rezultatele arată că atât abordările locale, cât și cele din cloud sunt capabile să fie instalate într-o clădire de birouri în care 20 de noduri edge trimit date de mediu în același timp.

Un complex de clădiri. Pentru acest scenariu, dimensiunea complexului și numărul de noduri edge pot face o diferență importantă din punct de vedere al performanței; cazurile care au fost simulate sunt discutate separat. Pentru ambele implementări ale configurației mediului containerizat, experimentele cu 50 de noduri edge au arătat o creștere a timpului mediu de răspuns, cu o schimbare mai vizibilă pentru clusterul local Docker Desktop, unde a fost instalată o mașină virtuală pe o stație de lucru. Creșterea mai mică a timpului de răspuns pentru clusterul IBM Cloud Kubernetes a fost influențată de utilizarea unui cluster dedicat pentru a oferi stabilitate. Două treimi dintre solicitări au avut un timp de răspuns sub valoarea recomandată de 500 ms (majoritatea între 400 și 500 ms), iar cealaltă treime au avut timpi de răspuns între 500 și 575 ms. Timpul mediu de răspuns a fost mai mare atunci când numărul de fire active era

scăzut, deoarece erau doar câțiva utilizatori care așteptau executarea apelurilor lor; celelalte fuseseră deja servite, adică încărcătura era mare. Timpul de rulare al algoritmului de detectare a hazardurilor a crescut pentru ambele abordări, oferind o valoare similară. Pentru testul de stres cu 100 de noduri edge, ambele configurații au arătat o creștere a timpilor medii de răspuns (Fig. 6.2). Pentru clusterul IBM Cloud Kubernetes, timpul de rulare a algoritmului de detectare a hazardurilor pentru sistemul cloud a fost foarte similar cu rezultatul obținut pentru experimentul cu 50 de noduri edge. Pentru clusterul local Docker Desktop, timpul de rulare a crescut (Fig. 6.4), ilustrând din nou avantajul oferit de un cluster dedicat în comparație cu stația de lucru locală. Între 200 și 600 de noduri, timpii de răspuns pentru cele două opțiuni de proiectare au fost destul de similare; totuși, pornind de la 700 de noduri edge, s-a arătat un avantaj clar față de clusterul IBM Cloud Kubernetes, datorită faptului că atât timpii de răspuns, cât și timpii de rulare au fost mai buni. Experimentul cu 1000 de noduri edge a verificat comportamentul sistemului atunci când i-au fost trimise 1000 de apeluri. Rata de eroare pentru fiecare dintre sisteme a fost 0, ceea ce indică faptul că toate datele de mediu au fost adăugate cu succes la baza de date EWS. O mare diferență care a apărut doar pentru acest experiment, dar nu și pentru celelalte scenarii, este că clusterul local Docker Desktop a oferit un timp mediu de răspuns mai mare în comparație cu clusterul IBM Cloud Kubernetes. Pentru celelalte scenarii, rezultatul a fost invers; motivul pentru aceasta este că cererea a fost trimisă local, iar diferența de timp a fost explicată de faptul că, pentru opțiunea cloud public, cererea a fost transmisă prin Internet. În ceea ce privește algoritmul de detectare a hazardurilor, implementarea clusterului local Docker Desktop a oferit un timp de rulare de aproape trei ori mai mare decât cel obținut de clusterul IBM Cloud Kubernetes.

O limitare importantă atunci când se lucrează cu o soluție cloud poate fi legată de locația centrului de date. Această alegere influențează latența rețelei prezentă în apelul către aplicație. Această observație poate fi luată în considerare și pentru alte aplicații legate de cloud, cum ar fi [14], [15], [17], [18].

Acest lucru subliniază faptul că utilizarea unui cluster cloud dedicat poate oferi o performanță mai consistentă sub o sarcină mare. Totuși, ca o remarcă generală pentru rezultatele legate de opțiunea public cloud, un timp mediu de răspuns de aproape 2000 ms este prea mare pentru acest tip de sistem. Pe baza acestor rezultate, pentru un complex de clădiri inteligente, se recomandă o configurație de cluster mai puternică decât cea testată în acest studiu și descrisă în Capitolul 6.2.4. Cele 1000 de noduri edge reprezintă un experiment de încărcare care poate apărea în situații reale pentru complexe mari de clădiri, cum ar fi campusuri inteligente sau centre comerciale mari.

6.4.2 Recomandări

În capitolul 4, am analizat mai multe opțiuni de proiectare pentru EWS, luând în considerare două criterii: arhitectura containerizată și topologia rețelei de margine. Am prezentat și arhitectura EWS în [19]. După evaluarea calitativă, metoda aplicată pentru testarea performanței celor două opțiuni de proiectare arhitecturală, (A) cluster-ul Kubernetes din cloud-ul public și (B) cluster-ul local Kubernetes, a fost descrisă în Capitolul 6. În continuare, această secțiune prezintă proiectare alegerile făcute pentru dezvoltarea EWS.

Alegerea arhitecturală pentru containere: Pe baza rezultatelor testelor din capitolul 5 și, de asemenea, prezentat în [20] și a comparației performanței pentru cele cinci scenarii, concluzionez că alegerea arhitecturală A (clusterul Kubernetes în cloud public) oferă o performanță mai bună pentru o încărcare corespunzătoare unui complex de clădiri inteligente, unde numărul de noduri edge utilizate pentru colectarea informațiilor este mare. Un alt motiv

pentru această alegere este separarea sistemului de monitorizare a clădirii de clădirea monitorizată propriu-zisă. Principalul avantaj în cazul unui eveniment periculos care ar putea produce o întrerupere în interiorul clădirii este că sistemul nu ar fi afectat și ar reține informațiile cu privire la eveniment. Un alt motiv major pentru această alegere este legat de costurile inițiale și de întreținere, care sunt mai mici cu abordarea cloud public. Serviciile cloud au, de asemenea, un acord privind nivelul de servicii garantat, care asigură că sistemul poate funcționa mai mult de 99,9 la sută din timp. Acest aspect este crucial pentru un sistem de monitorizare a clădirii care include detectarea diferitelor urgențe care pot apărea în clădire în sfera sa de aplicare.

Alegerea Edge: Pe baza analizei calitative prezentate în Capitolul 4.3, sunt în favoarea alegerii de proiectare care utilizează noduri senzor conectate la noduri edge cu microprocesor și nu dispozitive de detectare bazate pe microprocesoare care sunt conectate direct la serviciile EWS de pe Kubernetes. Rolul nodului edge este de a culege date, de a efectua unele procesări de bază descentralizate, de exemplu, pentru a detecta când să activeze dispozitivele locale de alarmă și să trimită date către EWS pentru a fi procesate într-un mod centralizat folosind algoritmi care necesită mai multe resurse. Pentru a estima costurile corespunzătoare celor două opțiuni pentru rețeaua edge am presupus că s-au folosit aceleași tipuri și număr de senzori și am omis alte costuri, precum cele legate de arhitectura containerizată. Să luăm în considerare un nod bazat pe microprocesor rulat cu Raspberry Pi [21] - o placă de dezvoltare foarte populară, cu capacitatea de a acționa atât ca dispozitiv de detectare (Opțiunea edge B), cât și ca broker și nod edge (Opțiunea edge A) . Pentru ambele opțiuni, Raspberry Pi oferă suficientă putere de calcul pentru a rula algoritmi și mai complexi în viitor [22]. Costul mediu pentru un Raspberry Pi 3 (4× CPU ARM, 1,2 GHz, 1 GB RAM, 10/100 Ethernet, 2,4 GHz 802.11n wireless) este de 35 USD. Pentru Opțiunea edge A, un nod senzor bazat pe microcontroler poate fi implementat cu NodeMCU v3 (procesor de 32 de biți, 80 MHz, 128 KB RAM), o placă bazată pe microcontroler cu capacitate Wi-Fi integrată, pentru a trimite date către un nod de margine . Costul mediu pentru o placă NodeMCU este de aproximativ 7 USD. Astfel, estimările mele arată că o soluție care conține doar dispozitive de detectare bazate pe microprocesor (Opțiunea edge B) ar costa de cinci ori mai mult decât una care include și noduri senzori bazate pe microcontroler (Opțiunea edge A). Acest lucru este deosebit de important în cazul monitorizării unui complex de clădiri cu un număr mare de noduri edge. Prin urmare, Opțiunea edge A, care conține noduri edge, a fost selectată pentru că, în general, este mai puțin costisitoare decât cealaltă opțiune și pentru că necesită un număr mai mic de noduri edge, ceea ce este o premisă pentru a oferi performanțe mai bune.

7. Concluzii

7.1 Discuţie

Această teză a prezentat o soluție pentru tema monitorizării clădirilor pentru detectarea hazardurilor, și a propus un nou set de arhitecturi pentru evaluarea performanței sistemului. Prin urmare, pentru a obține aceste rezultate, au fost investigate articole referitoare la diferite subiecte precum monitorizarea clădirilor, urgențe, IoT, cloud, managementul dezastrelor, fog și edge computing. M-am concentrat pe prezentarea arhitecturii de EWS - software și rețea edge care vor fi utilizate pentru evaluarea sistemului. Partea de contribuție la cercetare a fost folosită pentru a prezenta EWS, aplicația pe care am dezvoltat-o pentru a rezolva problema de monitorizare a clădirii. Integrarea unui sistem de configurare dedicat oferă mijloacele necesare pentru a edita dispozitivele edge care sunt împrăștiate în întreaga clădire monitorizată pentru a colecta date de mediu. Combinația acestor opțiuni dezvoltate ca aplicație web oferă personalului clădirii mijloacele necesare pentru a fi mereu conectat la clădirea lor și pentru a

gestiona parametrii necesari. Edge Watcher System este aplicația pe care am propus-o și poate satisface cerințele menționate mai sus într-un mediu inteligent. Pe lângă această problemă, integrarea unui sistem de configurare dedicat oferă mijloacele necesare pentru a configura dispozitivele de margine care sunt împrăștiate în întreaga clădire monitorizată pentru a colecta date de mediu.

Tema de verificare și validare este reprezentată de testarea în lumea reală care validează implementarea EWS în scenarii din lumea reală. În consecință, în primul rând am testat configurația cu un singur nod a EWS instalat pe un server local. Cele două arhitecturi implicate în test au prezentat rezultate similare, ambele având un timp mediu de răspuns sub 40 ms, ceea ce este considerat foarte rapid pentru instalarea EWS locală. Avantajul primei arhitecturi prezentate a fost faptul că este mai ieftină atunci când este implementată cu un număr foarte mare de senzori, comparativ cu cealaltă (Opțiunea edge B) al cărei principal avantaj este simplitatea. Ultimul pas în validarea EWS a fost implementarea sistemului într-un mediu real. Prin urmare, am ales să o implementez în cadrul Universității „Politehnica" din București „Precis" unde am implementat multiple opțiuni de proiectare ale EWS utilizate pentru testarea performanței. Rezultatele pe care le-am obținut în urma rulării testelor de performanță arată că toate testele de performanță au atins timpi medii de răspuns sub 300 ms pentru datele trimise de la nodurile edge la EWS. Ca o remarcă generală, ambele Opțiuni 1 și 2 au înregistrat rezultate similare. Din această perspectivă, fie Opțiunea de proiectare 1, fie Opțiunea de proiectare 2 vor putea funcționa similar într-un mediu real. Diferența dintre ele va apărea atunci când costul va fi luat în discuție. Opțiunea 1 implementează o conexiune fizică de la un dispozitiv bazat pe microprocesor (Raspberry Pi 3) la senzor. Trebuie achiziționat un număr mai mare de astfel de dispozitive pentru a implementa Opțiunea 1 în comparație cu Opțiunea 2. Pentru Opțiunea 2, sunt necesare mai puține dispozitive bazate pe microprocesor, deoarece acest nod edge este utilizat ca broker MQTT care primește date de la mai multe dispozitive mai ieftine, cum ar fi Node MCU. Dacă studiem mai în profunzime scenariile prezentate, observ că situațiile în care algoritmul de detectare a hazardurilor a fost implementat pe nodul edge primesc răspunsul de la EWS mai rapid, deoarece are mai puțin de procesat. Cea mai mare diferență se înregistrează pentru cazul în care EWS este implementat în Cloud. În acest caz, instalarea locală este în medie de 3 ori mai rapidă. Acest rezultat era de așteptat, deoarece versiunea Cloud a EWS a fost implementată în regiunea Milano, ceea ce este foarte departe, în comparație cu cazul rețelei locale. Un alt aspect este numărul de noduri edge implicate. Când au fost implicate două noduri edge, timpii de răspuns tind să fie puțin mai mari decât cazul unui singur nod. Cu toate acestea, diferențele dintre aceste două opțiuni nu sunt foarte mari și ambele au demonstrat că funcționează conform intenției în aceste situații.

În ceea ce privește subiectul de evaluare a scalării, în primul rând am măsurat performanța EWS cu noduri edge simulate. Această parte a evaluat mai multe opțiuni de proiectare pentru un sistem care monitorizează una sau mai multe clădiri inteligente cu scopul de a culege informații de la o mare varietate de senzori, de a detecta situații anormale (cum ar fi flăcări, gaze toxice, scurgeri etc.) și de a notifica personalul responsabil atunci când apar evenimente de urgență. Mai multe detalii despre acest subiect de notificări pot fi găsite în [23]. Opțiunile de proiectare au ținut cont de arhitectura software bazată pe containere și de dispozitivele edge de detectare. Pe lângă o analiză calitativă, am prezentat lucrări bazate pe medii containerizate pentru a testa performanța, care are o pondere importantă în sistemele de alertă. Timpii de răspuns furnizați trebuie să rămână sub un anumit prag pentru ca soluția să fie aprobată și implementată într-un mediu de producție. În concluzie, evaluarea performanței este o etapă importantă pentru un proiect deoarece oferă rezultatele necesare pentru a determina capacitatea unei aplicații și pe

baza rezultatelor configurația exactă a sistemului care este necesară pentru capacitatea dorită. Prin urmare, aceasta testare reproduce diferite scenarii din lumea reală pentru a testa limitele sistemului și modul în care acesta se comportă față de diferite intrări. Pe lângă aceste aspecte, un alt fapt important este rata de eroare care se înregistrează pentru fiecare caz. Pentru ca o aplicație să fie considerată stabilă, această valoare ar trebui să fie aproape de 0 pentru toate apelurile. Aceste argumente sunt validate de diferite experimente care au fost efectuate asupra și prezentate în capitolul de cercetare. Rezultatele înregistrate au oferit un caz foarte puternic pentru fiecare dintre sisteme de a defini capacitatea maximă și, de asemenea, mijloace de îmbunătățire pentru a crește această valoare. Pentru a realiza acest lucru, pot fi utilizate mai multe instrumente pentru testarea performanței, Apache JMeter este o alegere foarte populară datorită modelului său de distribuție gratuită și, de asemenea, datorită utilității pe care o oferă. În ceea ce privește obiectivul meu principal – Edge Watcher System au fost create mai multe scenarii care măsoară eficiența acestei aplicații implementate în două medii diferite: un cluster local și un cluster Cloud gratuit. Prin urmare, am implementat două setări de mediu containerizat (un cluster IBM Cloud Kubernetes și un cluster local Docker Desktop) și am simulat comportamentul pentru mai multe scenarii corespunzătoare configurațiilor din lumea reală cu 1 până la 1000 de noduri edge. Pentru setările corespunzătoare unui apartament mic, unei case, unei clădiri rezidențiale mici și unei clădiri de birouri, timpul mediu de răspuns a fost cu 250 ms mai mare pentru cloud-ul public decât pentru cluster-ul local. Cu toate acestea, pentru un complex de clădiri cu peste 600 de noduri edge, timpul de răspuns a fost cu 700 ms mai mic pentru cloud decât pentru soluția locală. Am folosit constatările evaluării performanței și analiza opțiunilor edge pentru a face alegeri de proiectare pentru EWS, o soluție cu noduri senzor bazate pe microcontroler, noduri edge bazate pe microprocesor și servicii de monitorizare, configurare și notificare cu un cluster IBM Cloud Kubernetes. Valorile care au fost vizate pentru aceste teste sunt: raportul de eroare (pentru a valida dacă toate solicitările au avut succes), timpul de răspuns (min, max, mediu și percentile, acestea sunt cele mai importante valori care subliniază experiența reală de a efectua apeluri). Pe lângă testarea JMeter, a fost efectuată o altă măsurătoare pentru algoritmul de decizie utilizat de Edge Watcher System pentru a decide dacă solicitarea nodului edge conține sau nu o posibilă alertă. Pentru acest aspect a fost folosită o bibliotecă Node.js numită „execution-time”. Rezultatul afișat de această bibliotecă este foarte important deoarece arată timpul exact petrecut de algoritm și poate fi extras din timpul total de răspuns oferit de JMeter pentru a decide dacă este nevoie de o configurație mai puternică. Rezultatele obținute în urma acestor teste au fost satisfăcătoare pentru ambele abordări. Diferența de bază dintre rezultatele înregistrate pentru cele două sisteme este legată în principal de faptul că, clusterul local a oferit un timp de răspuns mai rapid deoarece apelurile au fost efectuate local. Pentru clusterul IBM Cloud Kubernetes, solicitările au durat mai mult deoarece sistemul este situat în Cloud. O altă remarcă este că sistemul Cloud, fiind prevăzut cu un mediu de lucru dedicat, tinde să ofere rezultate mai consistente chiar și pentru percentila 95, fiind aproape de timpul mediu de răspuns. Valoarea dorită pentru timpul de răspuns ar trebui să fie în apropierea a 500 ms. Acest lucru a fost ușor de realizat pentru primele cinci scenarii. Dacă sunt conectate mai mult de o sută de noduri edge, atunci ar trebui utilizat un sistem mai puternic

7.2 Rezumatul contribuțiilor originale

1. Lecții învățate din studiul literaturii despre clădirile inteligente în cloud, situații de urgență, managementul hazardurilor și cloud computing.

2. Prototipul unui sistem de monitorizare a hazardurilor în mediu universitar

3. Comparație între sistemul implementat pe o mașină virtuală și același sistem implementat pe cloud.
4. Monitorizarea implementării sistemului folosind containere
5. Arhitecturi atât pentru sistemele locale (folosind o mașină virtuală) cât și în cloud (folosind un cluster Kubernetes)

Monitorizarea este esențială pentru fiecare clădire cu un număr mare de locatari și poate fi vitală în detectarea evenimentelor periculoase care pot afecta viața locuitorilor săi. Există multe tipuri de dispozitive de detectare care pot fi utilizate într-o clădire, cum ar fi colectoare de date de mediu, monitorizare video, colectare de informații despre echipamente și senzori virtuali.

6. Propunerea unui sistem și arhitectură de monitorizare a clădirii și de detectare a hazardurilor (Edge Watcher System)
7. Propunerea unui model de date pentru Sistemul Edge Watcher care subliniază toată funcționalitatea aplicației.
8. S-a prezentat un API bine documentat pentru o soluție de clădire inteligentă care poate fi utilizată în continuare pentru a integra diferite servicii.
9. S-a prezentat un frontend web responsive care este dedicat configurării si monitorizării unei clădiri inteligente simplificând operațiile realizate la nivelul ei
10. Propunerea unei abordări cloud bazată pe containere pentru implementarea acestui sistem
11. S-a propus un set de tehnologii care se integrează și servesc ca o soluție robustă pentru un sistem de gestionare a clădirii.
12. S-a propus un cadru unificat pentru sistemele de monitorizare a clădirilor care pot fi utilizate pentru mai multe clădiri.
13. S-a propus o arhitectură pentru un manager de clădire inteligent bazat pe cloud, care poate monitoriza clădirea locală și poate trimite notificări în cazul detectării hazardurilor
14. S-a creat un instrument de configurare bazat pe cloud care oferă posibilitatea de a genera configurații pentru nodurile de margine a clădirii locale.
15. Furnizarea de scenarii multiple care pot fi urmate pentru utilizarea în lumea reală a aplicației de gestionare a clădirilor inteligente.
16. S-a propus o implementare Kubernetes bazată pe cloud pentru sistemul de monitorizare a clădirilor și de detectare a hazardurilor.
17. S-a prezentat un algoritm de notificare care ia în considerare dacă evenimentul periculos detectat a fost trimis de la același nod

Scenariile de testare din viața reală sunt esențiale în validarea funcționării unui sistem. În cazul Edge Watcher System, un scenariu de testare a performanței în viața reală este reprezentat de implementarea unei rețele edge fizică compusă din dispozitive IoT care vor colecta datele de mediu din clădirea țintă. În acest domeniu, există mai multe arhitecturi posibile care pot fi implementate pentru acest tip de sistem care implică selectarea tipurilor de dispozitive ce pot

îndeplini această acțiune. Pentru teza mea, am comparat două tipuri de arhitecturi de rețea edge și am măsurat performanța legată de colectarea datelor.

18. Am creat arhitecturile pentru cele două opțiuni de rețea edge bazate (Opțiunea edge A și Opțiunea edge B)
19. Am implementat ambele soluții (Opțiunea edge A și Opțiunea edge B) și le-am conectat la Edge Watcher System
20. Am testat performanța EWS în legătură cu cele două opțiuni de rețea edge implementate
21. Compararea rezultatelor celor două soluții propuse

Am implementat EWS într-un mediu universitar unde am testat performanța EWS utilizând o versiune locală și o versiune Cloud a sistemului împreună cu două arhitecturi de rețea edge diferite.

22. Au fost luate în considerare două opțiuni de proiectare care conțineau două versiuni diferite (pe baza opțiunii edge A și B și a opțiunii arhitecturale A și B)
23. Măsurarea timpului mediu de răspuns al EWS luând în considerare locația software-ului, locația algoritmului de decizie și arhitectura rețelei edge implicată.
24. Compararea arhitecturilor EWS utilizate pentru testare în clădirea Universității

Testarea performanței este o parte esențială în dezvoltarea unei aplicații, deoarece oferă mijloacele de a afla dacă sistemul testat rulează la parametrii doriți cu intrări diferite. Importanța acestui pas critic se bazează în mare parte pe ideea că fiecare nouă piesă de dezvoltare trebuie testată pe diferite scenarii predefinite în care sunt monitorizate diverse metrici.

25. Testarea performanței sistemelor de management al clădirilor
26. Arhitectura sistemului pentru mediul de testare.
27. Implementări în cloud și locale pentru testarea performanței.
28. Testare bazată pe scenarii care acoperă mai multe tipuri de clădiri și medii, cum ar fi: un apartament mic, o casă, o clădire rezidențială mică, o clădire de birouri și un complex de clădiri
29. Simularea solicitărilor API ale nodurilor edge ale clădirii cu Apache JMeter.
30. Măsurarea performanței algoritmului de detectare a hazardurilor
31. Integrarea unei soluții de monitorizare native în cloud.
32. Evaluarea de scalare a Edge Watcher System până la o mie de noduri edge
33. Discuție arhitecturală bazată pe diferite opțiuni de proiectare pentru nodurile edge și locația sistemului cloud
34. Discuție despre latența rețelei pentru sistemul cloud
35. Discuție de performanță pentru hardware-ul oferit în cloud și hardware-ul nodurilor edge
36. Discuție de cost bazată pe alegerile nodurilor edge

37. Discuție aprofundată legată de rezultatele testării
38. Recomandări de arhitectură bazate pe evaluarea scalarii sistemului și pe rezultatele testelor din lumea reală

7.3 Lista publicațiilor

În timpul redactării tezei, am folosit conținut din articolele 1, 4, 6 și 9 care pot fi găsite mai jos:

1. **Florin Lacatusu**, A. D. Ionita, Marian Lacatusu, I. Damian and D. Saru, "A Comparison of Cloud Edge Monitoring Solutions for a University Building," *2022 IEEE 18th International Conference on Intelligent Computer Communication and Processing (ICCP)*, Cluj-Napoca, Romania, 2022, pp. 253-257, doi: 10.1109/ICCP56966.2022.10053978.

2. Marian Lacatusu, A. D. Ionita, **Florin Lacatusu** and I. Damian, "Decision support for multicloud deployment of a modeling environment," *2022 IEEE 18th International Conference on Intelligent Computer Communication and Processing (ICCP)*, Cluj-Napoca, Romania, 2022, pp. 247-251, doi: 10.1109/ICCP56966.2022.10053951.

3. Ioan Damian, Marian Lacatusu, **Florin Lacatusu** and A. D. Ionita, "Web Services for Guiding Persons with Locomotor Impairments in Public Spaces," 2022 26th International Conference on System Theory, Control and Computing (ICSTCC), 2022, pp. 639-644, doi: 10.1109/ICSTCC55426.2022.9931861, **WOS:000889980600107**

4. **Lăcătușu, Florin**; Ionita, A.D.; Lăcătușu, Marian.; Olteanu, A. Performance Evaluation of Information Gathering from Edge Devices in a Complex of Smart Buildings. *Sensors* **2022**, *22*, 1002. https://doi.org/10.3390/s22031002 , *Impact factor:* 3.84 – Q2, **WOS:000760176500001**

5. Lăcătușu, Marian; Ionita, A.D.; Anton, F.D.; **Lăcătușu, Florin** Analysis of Complexity and Performance for Automated Deployment of a Software Environment into the Cloud. *Appl. Sci.* **2022**, *12*, 4183. https://doi.org/10.3390/app12094183, *Impact factor:* 2.84 - Q2, **WOS:000794735600001**

6. **Florin Lacatusu**, I. Damian, A. D. Ionita and Marian Lacatusu, "Smart Building Manager Software in Cloud", U.P.B. Sci. Bull., *Series C, vol. 83, no. 4*, 2021, *Impact factor:* 0.37, **WOS:000741473700003**

7. Marian Lacatusu, **Florin Lacatusu**, Ioan Damian, Anca Daniela Ionita, Multicloud Deployment to Support Remote Learning, *15th International Technology, Education and Development Conferenc*e (INTED 2021) Proceedings (2021), pp. 4601-4606, IATED Digital Library, doi: 10.21125/inted.2021.0936

8. Ioan Damian, Marian Lacatusu, Anca Daniela Ionita, **Florin Lacatusu**, Software Services to Support Faculty Management in Times of Pandemic, *15th International*

*Technology, Education and Development Conferenc*e (INTED 2021) Proceedings (2021), pp. 4634-4639, IATED Digital Library, doi: 10.21125/inted.2021.0943

9. **Lăcătuşu, Florin**; Ionita, A.D. Architecture for Monitoring Risk Situations in a University Environment. Rev. Roum. Sci. Tech. Electrotech. **2020**, 65, 259–263, Impact factor: 0.44, **WOS:000608261900017**

10. Adriana Olteanu, **Florin Lacatusu**, Marian Lacatusu, Iulian Craciun, Anca Daniela Ionita, ”Mobile Application for Crisis Situations in a University Campus” - The International Scientific Conference eLearning and Software for Education; Bucharest Vol. 2, Bucharest: "Carol I" National Defence University. (2018): 280-287. **WOS:000467466800038**

11. Olteanu, Adriana**;** Lacatusu, Marian; Ionita, Anca Daniela; **Lacatusu, Florin**, ”Platform for Informal Education and Social Networking to Increase Awareness Regarding Nuclear Vulnerabilities” - The International Scientific Conference eLearning and Software for Education; Bucharest Vol. 2, Bucharest: "Carol I" National Defence University. (2018): 333-340. **WOS:000467466800045**

7.4 Perspective de viitor

În ceea ce privește lucrările viitoare, următorul pas pentru Edge Watcher System ar fi să fie instalat într-o clădire ca soluție principală pentru monitorizarea și testarea performanței acestuia atunci când funcționează pe o perioadă lungă de timp. Un alt fapt interesant ar fi testarea performanței EWS atunci când este integrat într-o infrastructură de oraș inteligent. Prin urmare, mai multe instanțe ale EWS ar fi implementate pentru fiecare dintre clădiri, iar funcționarea EWS poate fi testată și în acest scenariu.

În plus, având în vedere faptul că Edge Watcher System este o aplicație nativă în cloud, va rula în mod similar pe orice furnizor de cloud fără modificări sau cu modificări minime. În consecință, o altă perspectivă viitoare ar fi testarea performanței software-ului de monitorizare care rulează pe serviciile Kubernetes furnizate de un alt furnizor de cloud pentru a vedea dacă performanța variază între ele. În prezent, în timpul testelor efectuate în studiul meu, IBM Cloud a fost folosit ca furnizor de cloud pentru serviciul EWS Kubernetes. Performanța înregistrată în acest caz a fost satisfăcătoare pentru opțiunea de hardware aleasă. Prin urmare, compararea mai multor oferte de la mai mulți furnizori ar fi interesantă deoarece soluția optimă poate fi obținută prin studierea raportului performanță-preț.

Serviciile Kubernetes implementate de la mai mulți furnizori de cloud având același preț ar trebui comparate și ar trebui aleasă cea mai bună opțiune performanță-preț. Ulterior, un experiment foarte interesant ar fi, pe lângă preț, testarea performanței EWS care rulează pe serviciile Kubernetes furnizate de diferiți furnizori de cloud public. Principala cerință pentru acest tip de comparație ar fi implementarea clusterului Kubernetes într-o locație similară pentru toți furnizorii de cloud. În acest fel, ar fi diferență mică atunci când luăm în considerare latența rețelei ce ar putea fi influențată de locația clusterului Kubernetes. Dacă folosim altceva in loc de Kubernetes, o altă posibilă implementare care poate fi luată în considerare pentru backend-ul EWS este reprezentată de funcțiile serverless. Aceste tipuri de resurse sunt pe deplin gestionate de furnizorul de cloud, iar dezvoltatorul lucrează numai la codul aplicației. Prin

urmare, o comparație foarte interesantă ar fi între soluția Kubernetes și funcțiile serverless împreună cu servicii de baze de date PaaS. Comparația care s-ar face între aceste soluții ar trebui să acopere, în primul rând, tema performanței și, de asemenea, diferența de cost între ele. În final, desigur, această comparație poate fi generalizată și pentru a experimenta cele mai importante opțiuni ale furnizorului de cloud în ceea ce privește funcțiile serverless și a testa oferta acestora pentru a găsi cea mai potrivită opțiune.

Bibliografie selectivă

[1] D. Cârstoiu, V.E. Oltean, S.M. Nica, G. Spiridon, A cloud-based architecture proposal for rehabilitation of aphasia patients, Rev. Roum. Sci. Techn. – Électrotechn. et Énerg., 62, 3, pp. 332–337 (2017)

[2] G.M. Vasilescu, I. Bârsan, G. Kacso, M.E. Marin, M. Maricaru, L.N. Demeter, Two devices equipped with temperature sensors used to detect and locate incipient breast tumors, Rev. Room. Sci. Technol. – Électrotechn. et Énerg., 63, 4, pp. 441–445 (2018).

[3] Kurniawan, F.; Meidia, H.; Salim, S. Building Monitoring System Based on Zigbee. JCSI 2013, 6, 65–69

[4] Kim, D.; Muhammad, H.; Kim, E.; Helal, S.; Lee, C. TOSCA-Based and Federation-Aware Cloud Orchestration for Kubernetes Container Platform. Appl. Sci. 2019, 9, 191.

[5] Kubernetes Concepts - Pods: https://kubernetes.io/docs/concepts/workloads/pods/, Accessed October 5 2020.

[6] Erinle, B. Performance Testing with JMeter 2.9; Packt Publishing: Birmingham, UK, 2013

[7] Haseeb-Ur-Rehman, R.M.A.; Liaqat, M.; Mohd Aman, A.H.; Ab Hamid, S.H.; Ali, R.L.; Shuja, J.; Khurram Khan, M. Sensor Cloud Frameworks: State-of-the-Art, Taxonomy, and Research Issues. *IEEE Sens. J.* **2021**, *21*, 22347–22370.

[8] International Building Code; International Code Council: Country Club Hills, IL, USA, 2018; ISBN 978-1-60983-735-8.

[9] National Construction Code 2019 Building Code of Australia, Volume One. Available online: https://ncc.abcb.gov.au/sites/default/files/ncc/NCC_2019_Volume_One_Amendment%201.pdf (accessed on 6 December 2021).

[10] I. Damian, M. Lacatusu, F. Lacatusu and A. D. Ionita, "Web Services for Guiding Persons with Locomotor Impairments in Public Spaces," 2022 26th International Conference on System Theory, Control and Computing (ICSTCC), 2022, pp. 639-644, doi: 10.1109/ICSTCC55426.2022.9931861.

[11] I. Damian, M. Lacatusu, A.D. Ionita, F. Lacatusu (2021) Software Services to Support Faculty Management in Times of Pandemic, *INTED2021 Proceedings*, pp. 4634-4639. INTED 2021

[12] Olteanu, Adriana**;** Lacatusu, Marian; Ionita, Anca Daniela; Lacatusu, Florin, ”Platform for Informal Education and Social Networking to Increase Awareness Regarding Nuclear Vulnerabilities” - The International Scientific Conference eLearning and Software for Education; Bucharest Vol. 2, Bucharest: "Carol I" National Defence University. (2018): 333-340.

[13] Serrano, D.; Bouchenak, S.; Kouki, Y.; Alvares de Oliveira, F., Jr.; Ledoux, T.; Lejeune, J.; Sopena, J.; Arantes, L.; Sens, P. SLA guarantees for cloud services. Future Gener. Comput. Syst. 2016, 54, 233–246.

[14] Lăcătușu, F.; Ionita, A.D.; Lăcătușu, M.; Olteanu, A. Performance Evaluation of Information Gathering from Edge Devices in a Complex of Smart Buildings. *Sensors* **2022**, *22*, 1002.

[15] M. Lacatusu, A. D. Ionita, F. Lacatusu and I. Damian, "Decision support for multicloud deployment of a modeling environment," *2022 IEEE 18th International Conference on Intelligent Computer Communication and Processing (ICCP)*, Cluj-Napoca, Romania, 2022, pp. 247-251, doi: 10.1109/ICCP56966.2022.10053951.

[16] Lăcătușu, M.; Ionita, A.D.; Anton, F.D.; Lăcătușu, F. Analysis of Complexity and Performance for Automated Deployment of a Software Environment into the Cloud. *Appl. Sci.* **2022**, *12*, 4183.

[17] M. Lacatusu, F. Lacatusu, I. Damian, A.D. Ionita (2021) Multicloud Deployment to Support Remote Learning, *INTED2021 Proceedings*, pp. 4601-4606. INTED 2021

[18] Lăcătuşu, F.; Ionita, A.D. Architecture for Monitoring Risk Situations in a University Environment. Rev. Roum. Sci. Tech. Electrotech. **2020**, 65, 259–263

[19] F. Lacatusu, I. Damian, A. D. Ionita and M. Lacatusu, "Smart Building Manager Software in Cloud", U.P.B. Sci. Bull., *Series C, vol. 83, no. 4*, 2021

[20] F. Lacatusu, A. D. Ionita, M. Lacatusu, I. Damian and D. Saru, "A Comparison of Cloud Edge Monitoring Solutions for a University Building," *2022 IEEE 18th International Conference on Intelligent Computer Communication and Processing (ICCP)*, Cluj-Napoca, Romania, 2022, pp. 253-257, doi: 10.1109/ICCP56966.2022.10053978.

[21] Johnston, S.J.; Cox, S.J. The Raspberry Pi: A Technology Disrupter, and the Enabler of Dreams. *Electronics* **2017**, *6*, 51

[22] Cloutier, M.F.; Paradis, C.; Weaver, V.M. A Raspberry Pi Cluster Instrumented for Fine-Grained Power Measurement. Electronics 2016, 5, 61

[23] Adriana Olteanu, Florin Lacatusu, Marian Lacatusu, Iulian Craciun, Anca Daniela Ionita, ”Mobile Application for Crisis Situations in a University Campus” - The International Scientific Conference eLearning and Software for Education; Bucharest Vol. 2, Bucharest: "Carol I" National Defence University. (2018): 280-287.