Факултет техничких наука

Универзитет у Новом Саду

Архитектуре система великих скупова података

Edge, fog, continuum computing

Аутор:
Александар Стојановић

Индекс:
E2 119/2023

10. април 2024.

Садржај

Списак слика

1 Увод

У данашњем, дигиталном добу, нагли пораст броја уређаја повезаних на интернет (*Internet of Things - IoT*) представља један од најзначајнијих изазова за традиционалне рачунарске парадигме. Од обичних кућних апарата до сложених индустријских сензора, број *IoT* уређаја расте експоненцијално, доносећи са собом потребу за ефикасним и скалабилним приступима обради података.

Претходне парадигме, попут класичних рачунарских архитектура и централизованих *cloud* система суочавају се са многобројним изазовима. Уколико се знатно повећа број крајњих уређаја, количина података који се преносе преко мреже може довести до загушења мреже као и повећања латенције обраде података, што код система који захтевају обраду података у реалном времену може представљати велики проблем. У случајевима рада апликација са осетљиивм корисничким подацима, њихово пренос на *cloud* платформе представља безбедносни ризик. Оно што свакако не треба занемарити је и да константна комуникација великог броја крајњих уређаја са удаљеним рачунарским центрима у неким случајевима може произвести и знатне енергетске губитке.

Како би се ови изазови превазишли , развијају се нови концепти као што су *Edge, Fog* и *Continuum computing* чија је основна идеја приближавање обраде и анализе података њиховим изворима. У даљим поглављима биће детаљније објашњене идеје иза сваке од горе наведене 3 парадигме, архитектуре за њихову реализацију као и неки од примера њихове употребе.

2 *Edge computing*

У контексту *Edge computing*-а, термин *edge* се посебно односи на географску или логичку границу мреже где се обрада података одвија ближе извору података или уређају крајњег корисника, уместо да се ослања само на централизоване центре података или сервере у *cloud*-у. Иако ова идеја постоји већ годинама уназад, оно што је условило њену тренутну популаризацију и развој је долазак 5G мреже на тржиште која омогућава међусобну комуникацију много већег броја уређаја, као и много већи пропусни опсег без којих реализација овакве архитектуре не би била могућа.

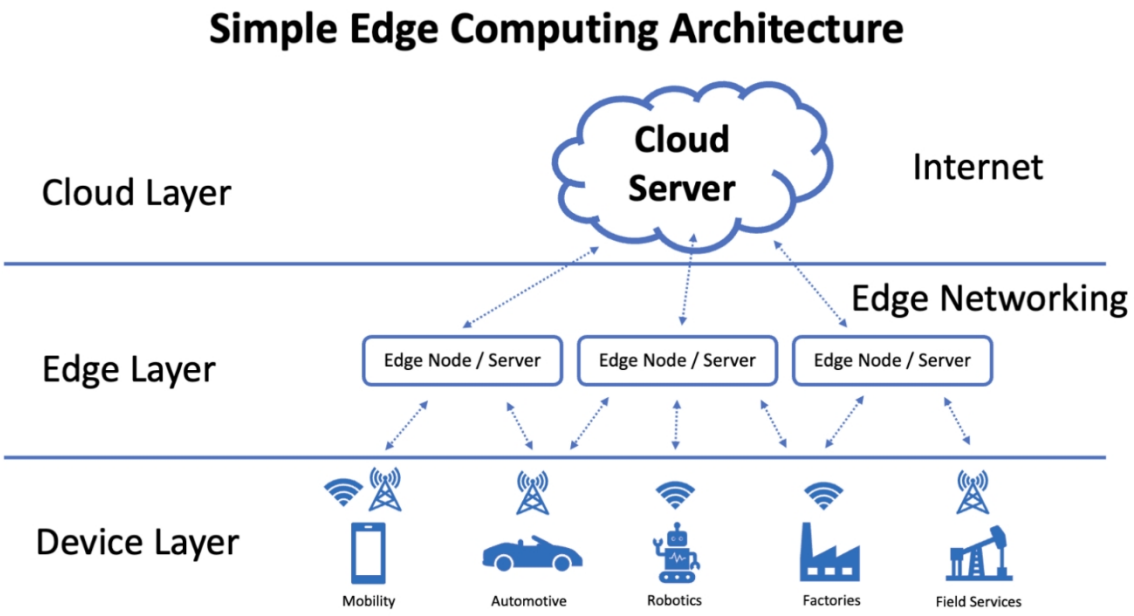
Основне карактеристике *edge comptuing*-а су:

- Близина извора података. Уређаји који обрађују и анализирају податке налазе се веома близу самог извора података, тиме смањујући кашњење и побољшавајући време одговора за критичне апликације.
- Децентрализација. Ресурси за израчунавање и аналитику распоређују се преко *edge* уређаја, сервера и *gateway*-а омогућавајући локално процесирање података и одлучивање.

2.1 Архитектура

Архитектура оваквих система најчешће се састоји из три слоја 1:

- Слој крајњих уређаја, који се састоји од разних сензорских и мобилних уређаја као и уређаја који не морају само да прикупљају информације, него и да реагују у складу са наредбама система.
- *Edge* слој који служи за горе поменуту обраду и анализу података у близини самих крајњих уређаја, која смањује латеницју и оптерећење на мрежи. Обрађујући податке на овом слоју, такође се повећава и безбедност система пошто прикупљени и обрађени подаци ни у једном тренутку не морају напустити интерну мрежу организације.
- *Cloud* слој на који се шаљу задаци који превазилазе моћи edge слоја, као и подаци који су намењени за глобалну употребу, који су додатно филтрирани и шифровани ради боље заштите података јер њиховим слањем на *cloud* они су изложени свакаквим врстама напада који иначе нису могући у крајњим и *edge* слојевима.



Слика 1: Скица уобичајене *edge* архитектуре

2.2 Могуће примене

Локалност, брзина и безбедност израчунавања *edge computing*-а може бити од користи у разним случајевима:

2.2.1 Предиктивно одржавање машина у погонима

Неочекивани отказ машина у индустријским погонима може значајно угрозити рад, профит фирме као и безбедност радника који њима управљају или се налазе у њиховој близини. Тренутни, најчешћи начин превенције отказа оваквих машина обавања се редовним, заказаним контролама и сервисима. Овакав приступ превенцији се показао као ефикасан, али његов проблем је што тешко и једино искуствено може одредити интервал између две контроле што резултује сувише честим или ретким контролама, што може довести до ненаданог престанка рада машине или узалудног губитка ресурса на сувишне контроле. *Edge computing* овај проблем решава константим праћењем рада и стања сваке појединачне машине у реалном или приближно реалном времену помоћу мноштва сензора који су повезани или интегрисани у машине. На овај начин нема потребе за нагађањем времена контроле или поправке машине, због тога што се информације о стању машине могу добавити у било ком тренутку и по потреби организовати њихово сервисирање.

2.2.2 Видео надзор подржан вештачком интелигенцијом

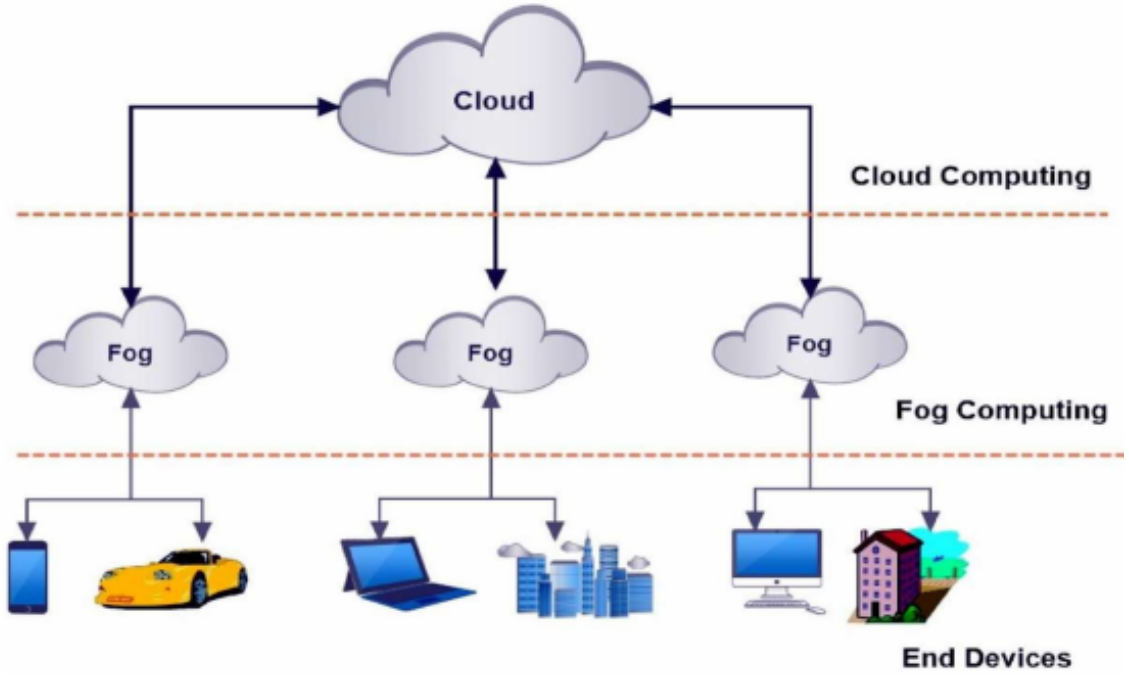
Видео надзор је један од најраспрострањенијих начина константног удаљеног надзора, добављања информација, као и превенције нежељених догађаја. Све већим развојем вештачке интелигенције, могућности видео надзора се проширују са пуког посматрања снимака камера од стране човека на ауто детекцију објеката и препознавање значајних догађаја, попут уласка непознатих лица у објекат или присуства човека у деловима погона који могу угрозити његово здравље или живот. Оваква примена вештачке интелигенције захтева обраду података у реалном времену, као и конекцију са одређеним базама података и серверима који када би се налазили на *cloud* платформама не би успевали да постигну најбоље перформансе и случајеви нестанка интернет конекције доводили би до потпуног отказа система. Због ових потешкоћа *edge computing* је добар кандидат за имплементацију ових система због своје локалности уређаја за обраду података која смањује вероватноћу велике латенције и отказа комуникације између компоненти система.

2.2.3 Пренос догађаја уживо

Видео преноси утакмица, конференција и осталих догађаја временом добијају на квалитету што све више и више оптерећује њихов мрежни пренос. Закшњење које се јавља може представљати проблем гледаоцима који се налазе у непосредној близини самог догађаја. С тога се уместо слања на централни *cloud* сервер, видео садржај поставља на локално распоређене *edge* сервере којима гледаоци могу приступити са веома малом латенцијом.

3 *Fog computing*

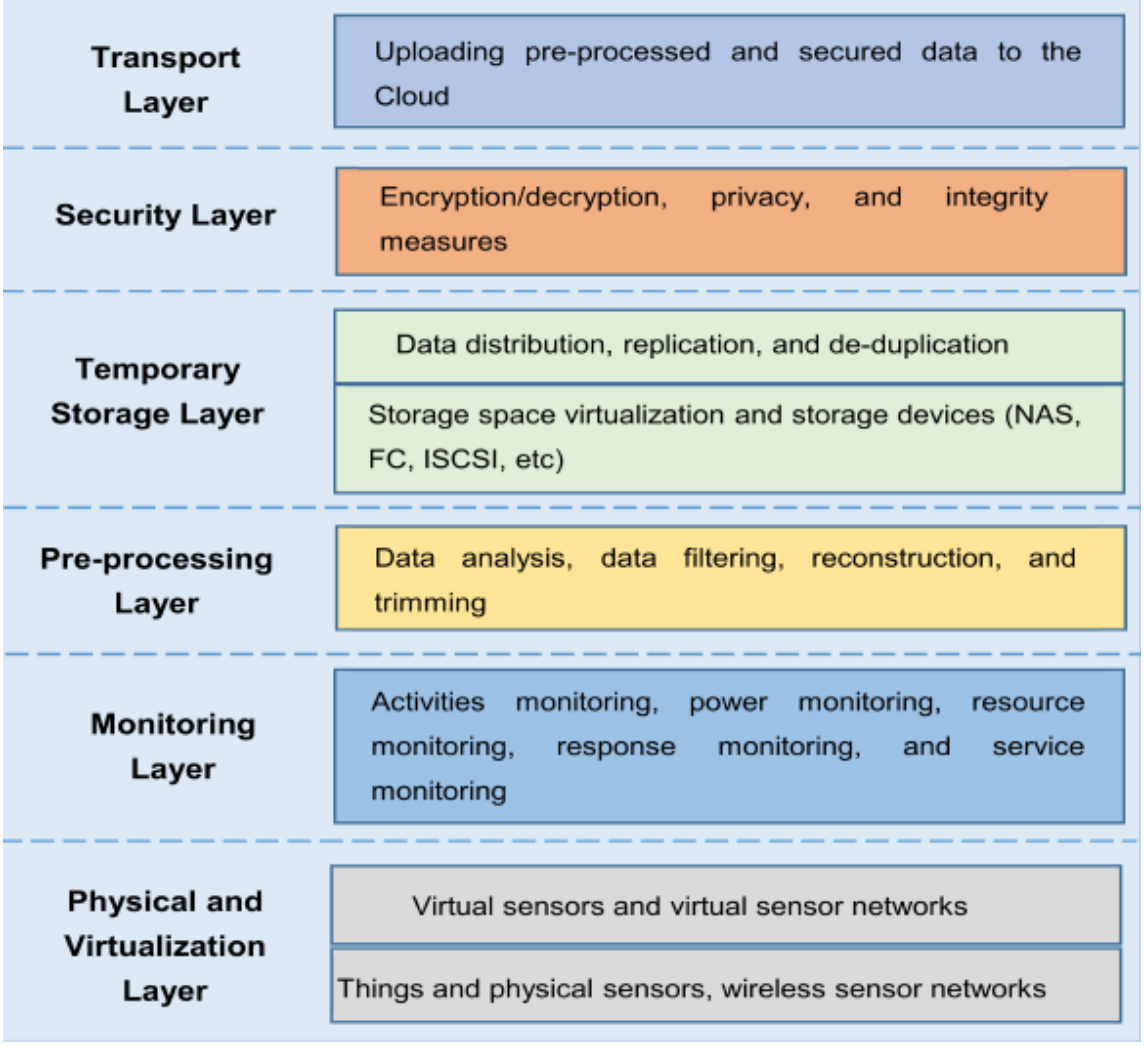
Иако се *edge* и *fog computing* често спомињу у истом контексту и неретко користе као синоними, постоји значајна разлика између ова два појма. *Fog computing* представља унапређење *edge computing*-а проширујући његове могућности обраде и складиштења података увођењем *fog* чворова који се налазе између *edge* и *cloud* система. У суштини, он је екстензија *cloud*-а с тиме да је много ближи уређајима *edge* система 2, што великим делом очувава све погодности *edge computing*-а као што су брза мрежна комуникација, мала латенција и безбедност, а опет омогућава обраде података које захтевају много веће рачунарске ресурсе. Овако *edge* систем још увек може самостално да обрађује најкритичније податке по безбедност и брзину рада апликације, а да остатак посла делегира *fog* чворовима. Још једна од погодности је могућност много већег географског распростирања једног система користећи горе поменуте *fog* чворове.



Слика 2: *Fog* је попут *cloud*-а који је много ближи крајњим уређајима

3.1 Архитектура

Архитектура *fog computing*-а састоји се из шест слојева 3:



Слика 3: Слојевита архитектура *fog computing*-а

Физички и виртуелизациони слој укључује различите типове чворова попут физичких, виртуелних чворова или виртуелних сензорских мрежа. Различити чворови географски су дистрибуирани да би прикупили информације из њихове околине и послали их вишим слојевима уз помоћ *gateway*-а на даље филтрирање и процесирање.

Слој за надзор прати искоришћење ресурса, доступност сензора и *fog* чворова, као и осталих елемената мреже. Сви задаци које обављају чворови се прате у овом

слоју, пратећи који чвор обавља који задатак, у које време, и шта ће му бити потребно у наредном кораку. Прате се перформансе и статус свих апликација и услуга које су имплементиране унутар система. Додатно, прати се и потрошња енергије *fog* чворова.

У пре-процесном слоју сакупљени подаци се анализирају, врши се филтрирање и редуковање података како би се извукле значајне информације. Претпроцесирани подаци се затим складиште у слоју за привремено складиштење. Када се подаци пренесу на *cloud*, више им није потребно локално складиштење и могу бити уклоњени са привремених складишних медија.

Безбедносни слој врши шифровање и дешифровање података, а често и примењује мере заштите њиховог интегритета, како их потенцијални нападачи не би могли неприметно прочитати или изменити приликом њиховог транспорта на *cloud*.

На самом крају, у транспортном слоју, претпроцесирани подаци се транспортују на *cloud* како би се издвојили и креирали додатни корисни сервиси. Ради ефикасне употребе енергије и очувања безбедности података, само део сакупљених података се транспортује на *cloud*.

3.2 Могуће примене

Тренутна централизована архитектура *cloud computing*-а суочава се са озбиљним изазовима за *IoT* апликације. На пример, не може да подржи временски осетљиве *IoT* апликације, као што су *video streaming, gaming* и *augmented reality*. Такође, недостаје свест о локацији процеса, јер је у питању централизовани модел. *Fog computing* може да реши ове изазове. *Fog computing* делује као мост између *IoT* уређаја и великих *cloud* рачунарских и складишних услуга. Он пружа високо виртуализовани модел рачунања, складиштења и мрежних ресурса између крајњих уређаја и класичних *cloud* сервера.

3.2.1 Аутономна возила

Постоје многе корисне функционалности, које зависе од *fog*-а и интернет конекције, које могу бити додате аутомобилима као што су "*hands free*" режим вожње или функција самопаркирања која више не би захтевала возача за воланом како би се аутомобил паркирао. У следећих неколико година се очекује да ће сви нови аутомобили имати могућност комуникације са блиским аутомобилима и интернетом. *Fog computing* ће бити најефикасније решење за сва возила повезана са интернетом, јер омогућава комуникацију у реалном времену. Такође, омогућиће аутомобилима, приступним тачкама и семафорима да комуницирају једни са другима како би обезбедили добру услугу корисницима. Уз коришћење *fog*-а уместо *cloud*-а, судари и друге несреће могле би се свести на минимум.

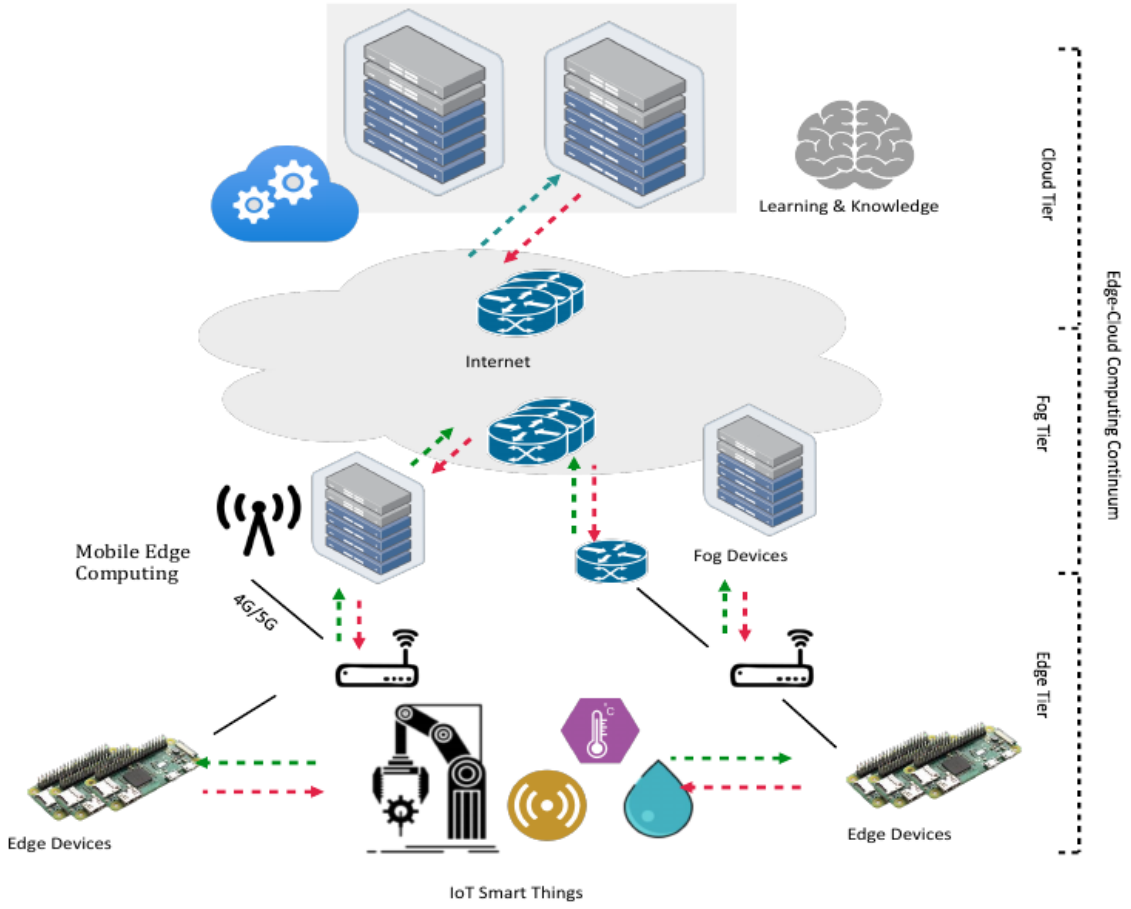
3.2.2 Паметне куће

IoT има много повезаних сензора и уређаја унутар домаћинстава. Међутим, ови уређаји долазе од различитих произвођача и користе различите платформе, што отежава њихово међусобно повезивање. Такође, неки задаци захтевају велику количину рачунарских ресурса и простора за складиштење података. *Fog computing* решава многе од ових проблема, интегрише све различите платформе и омогућава паметним кућним апликацијама приступ флексибилним ресурсима. *Fog computing* такође пружа много предности за апликације за обезбеђивање домаћинстава.

4 *Continuum computing*

Continuum computing представља еволуцију претходно поменутих парадигми, интегришући *edge, fog* и *cloud computing* у један кохезиван динамички систем, трудећи се да усклади локализовани *edge*, средишњи *fog* и централизовани *cloud* пружајући споља кориснику једну јединствену, свеприсутну мрежу уређаја, олакшавајући му коминикацију и интеракцију без потребе за размишљањем о сложености позадинске технологије. Чине га мноштво разноликих уређаја попут мобилних уређаја, сензора и *IoT* уређаја.

Интегришући ресурсе *cloud*-а, *edge* уређаје и *IoT*, *continuum computing* омогућава ефикасне, *real time* и динамичке рачунске процесе који задовољавају потребе данашњих разноврсних апликација. Он обавља рачунске операције дистрибуирајући оптерећење преко више уређаја у систему. Сваки уређај обавља део посла, а резултати се комбинују како би се произвео крајњи резултат. Ово омогућава брже време обраде и повећану скалабилност. Са *continuum computing*-ом, рачунске операције се ефикасно обављају прилагођавајући се променљивим захтевима и оптимизујући искоришћеност ресурса ван традиционалних граница. Ова алокација ресурса се заснива на факторима као што су близина ресурса, рачунарска способност и давање приоритета задацима који захтевају брзу реакцију. У зависности од задатка, одговори у реалном времену могу бити пренесени на *edge* уређаје, док се комплексна аналитика подразумевано може обавити у *cloud*-у. Ова динамичка дистрибуција задатака побољшава перформансе система и ефикасност обраде док смањује латенцију. Општа архитектура *continuum computing*-а је приказана на слици 4.



Слика 4: Архитектура *continuum computing*-а

Агилност овакве архитектуре доноси многе предности укључујући оптимизацију мрежног протока, скалабилност, малу латенцију, ефикасно искоришћене ресурса, балансирање оптерећења, флексибилност и поузданост. Упркос свим овим предностима, наилази се и на одређене потешкоће и изазове као што су међусобна комуникација уређаја различитих произвођача који функционишу служећи се различитим семнатичким правилима, сложеност управљања великог броја уређаја и синхронизација података.

4.1 Могуће примене

4.1.1 Паметни градови

Паметни град користи мрежу сензора и уређаја за прикупљање информација у реалном времену о транспорту, потрошњи енергије, управљању отпадом и јавним услугама. Подаци са ових извора могу се анализирати и користити за доношење одлука, као средство за повећање комфора, унапређење јавних услуга и квалитет живота грађана. Пошто *continuum computing* има подразумевану скалабилност, он се може динамички повећавати или смањивати као одговор на промене у екосистему паметног града.

4.1.2 Здравство

Здравство обухвата различите медицинске услуге, технологије и системе који су дизајнирани да спрече, дијагностикују, лече и управљају болестима и здравственим стањима. У последњих неколико година развијено је неколико медицинских уређаја, од мобилних сензора до високопрофесионалних машина (постављених у болницама и здравственим центрима) који се користе за прикупљање података о пацијентима и њихово процесирање путем паметних телефона или у *cloud*-у. Здравствена индустрија захтева тачне и брзе аналитичке резултате од рачунарских уређаја. Понекад је неопходно анализирати захтевне задатке као што су медицински снимци (рентген или ЦТ снимци) или геномско секвенцирање, али се резултат очекује у кратком временском периоду. Понекад је потребно користити вештачку интелигенцију или машинско учење за предвиђање стања пацијента, што захтева више рачунарских ресурса.

5 Примери имплементација

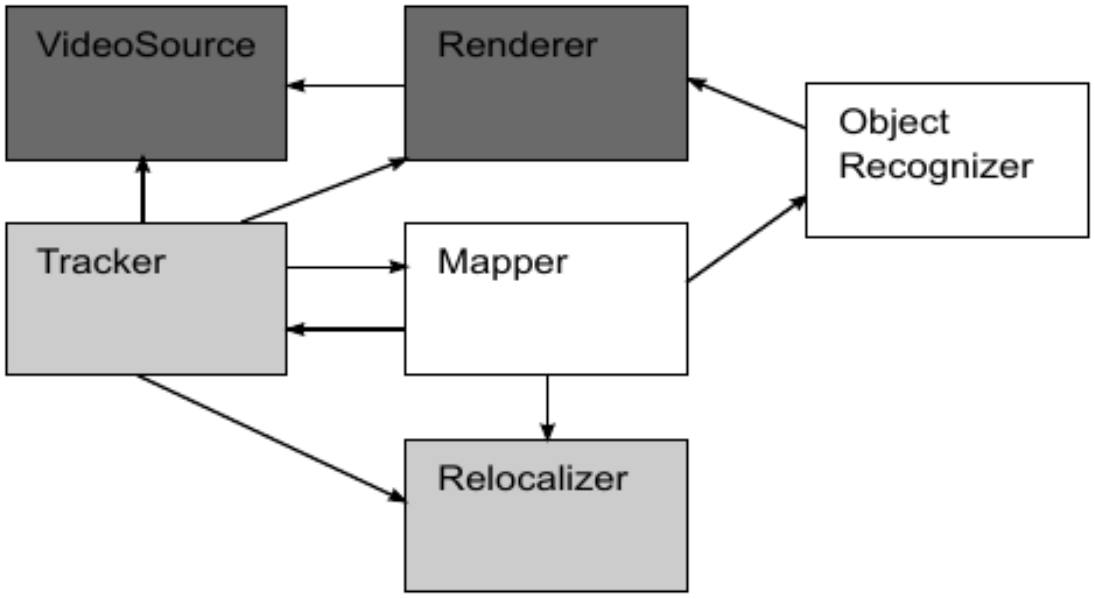
5.1 *Augmented reality offloading*

У овом примеру тема је апликација за проширену стварност која у реалном времену камером анализира околину и детектује познате објекте. Апликација, на снимку уживо, препознатим објектима задебљава спољну ивицу како би се лакше уочили на слици 5.

Слика 5: Са десне стране приказан је међукорак у раду апликације, док се на левој страни може приметити оивичена књига коју је апликација препознала

Будући да је ова апликација намењена за коришћење у покрету, мобилни телефони се намећу као идеални уређаји за ову намену, али они сами по себи не поседују довољно хардверских ресурса за захтевна процесирања у реалном времену која су неопходна за несметан рад. Такође, делегирање читавог процесирања рачунарима на *cloud*-у није адекватно због великог времена одзива, које не иде у корист обради у реалном времену. У даљем тексту биће описан процес разлагања апликације на независне процесе који се затим распоређују на *cloudlet* архитектуру уређаја која се налази близу уређаја на коме је покренута апликација.

5.1.1 Разлагање апликације на процесе



Слика 6: Процеси од којих је сачињена апликација

На слици 6 може се приметити неколико процеса који заједно чине апликацију. *Video source* представља камеру телефона која генерише видео садржај и прослеђује га *Renderer*-у који преко њега поставља помоћну мрежу која је поравната са положајем камере који се прати уз помоћ *Tracker*-а. Тако обрађене фрејмове *Mapper* добија и покушава да пронађе и постави значајне тачке које су од помоћи *Object recognizer*-у који на самом крају проналази и истиче објекте на слици. Улога *Relocalizer*-а је да у случају да ниједна значајна тачка није пронађена на слици промени позицију камере како би се праћење наставило.

Сваки од наведених процеса има различите захтеве у погледу хардверских ресурса. *Video source* и *Renderer* се из јасних разлога морају налазити на мобилном уређају који покреће апликацију, док се остали процеси могу делегирати екстерним уређајима. *Tracker* и *Relocalizer* захтевају више хардверских ресурса као што је случај и са *Mapper*-ом и *Object recognizerom* с тиме да *Tracker* и *Relocalizer* морају имати много бржи одзив од преостала два.

Како би било могуће покретати ове процесе на разним врстама уређаја, процеси су имплементирани у *Java* програмском језику уз помоћ *OSGi* сервисно оријентисаног система за управљање модулима. Имплементација процеса критичних по перформансе рађена је у *C/C++*-у, а затим, због компатибилности са остатком система, обмотана *Java* адаптером.

5.1.2 *Cloudlet* архитектура

Претходно дефинисани процеси представљају основну компоненту овог система и они се покрећу унутар извршних окружења *(EE)*. Више извршних окружења комуницира међусобно путем *RPC* протокола. Процеси могу да дефинишу додатна ограничења која се тичу перформанси у виду *XML* шеме, где је то најчешће брзина извршавања коју уређај на коме се процес извршава мора да испуни.

```
<component name="Tracker">
  <service name="TrackerService">
    <method name="trackFrame">
      <constraint type="maxTime">50</constraint>
    </method>
  </service>
  <parameter name="MaxFeatures"
             type="uint"
             range="50..1000"
             default="200"/>
</component>
```

Слика 7: Пример описа процеса који додатно намеће ограничења у виду брзине извршавања

Један или више извршних окружења покрећу се у оквиру једног оперативног система који је покренут на правом или виртуелизованом хардверу. Један овакав оперативни систем са покренутим извршним окружењима представља чвор и њиме управља *node agent (NA)*, који управља извршним окружењима, креира нова и стопира их. *node agent (NA)* такође надзире искоришћење ресурса у оквиру комплетног чвора.

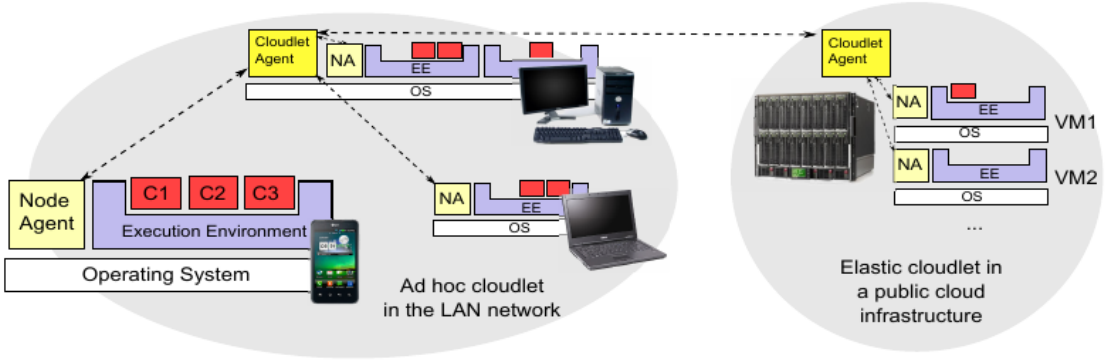
Више чворова који се налазе физички близу једни других (имају малу међусобну латенцију), чине *cloudlet*. *Cloudlet*-ом управља *cloudlet agent* који комуницира са свим интерним *node agent*-има. Више *cloudlet*-а такође могу међусобно да комуницирају у циљу размене процеса. Унутар роја суседних чворова *cloudlet agent* се покреће на чвору са највише ресурса.

За време извршавања, ако извршно окружење уочи да је ограничење неког од процеса прекршено, оно обавештава *node agent*-а који даље обавештава *cloudlet agent*-а који у том тренутку покушава да измени распоред процеса у оквиру његовог *cloudlet*-а и тако оптимизује рад апликације. Оваква хијерархијска архитектура има неколико предности. Много је склабилније да једно тело унутар роја доноси одлуку о реорганизацији него да сваки чвор учествује у дистрибуираном гласању. Калкулисање

расподеле посла увек се врши на чвору са највише ресурса и који има увид у све остале чворове, као и могућност комуникације са осталим *cloudlet*-има.

Постоје две врсте *cloudlet*-а, *ad hoc* и еластични. *ad hoc cloudlet* чине динамички пронађени чворови у локалној мрежи. Када се нови чвор придружи мрежи његов *node agent* се активира и покреће извршна окружења, док *cloudlet agent* врши ребалансирање посла при сваком уласку и изласку неког од чворова из мреже. Еластични *cloudlet* се извршава на виртуализованој инфраструктури, где се чворови извршавају унутар виртуалних машина. У овом случају *cloudlet agent* има могућност да креира и стопира чворове по потреби.

На слици 8 може се видети пример горе описане архитектуре где *ad hoc cloudlet* чине десктоп рачунар, лаптоп и паметни телефон, док се еластични *cloudlet* налази на јавној *cloud* инфраструктури.



Слика 8: Пример *cloudlet* архитектуре са еластичним и *ad hoc cloudlet*-има

5.2 Обезбеђивање отпорности на отказе у здравственим установама

У здравственим установама, захтева се беспрекоран рад апликације. Ове апликације управљају критичним задацима као што су чување података о пацијентима, праћење њиховог тренутног стања и давање лекова. Сваки застој или квар може директно утицати на негу и безбедност пацијената.

Cloud решења нуде скалабилност и лакоћу управљања за апликације у здравству, међутим постоје проблеми у вези са кашњењем, поузданошћу мреже, безбедношћу података као и несметаним наставком рада у случају отказа *cloud* система или мрежних проблема.

Fog computing представља могуће решење ових проблема, коришћењем сензорских и серверских уређаја који се налазе у непосредној близини здравствене установе, тиме испуњавајући високе захтеве оваквог типа установе.

У конкретној имплементацији као адекватни механизам изабрана је реактивна отпорност на грешке детекцијом отказа ослушкивањем *heartbeat* сигнала главног *cloud* сервера и затим пребацивање на привремени *fog* сервер као и реплицирану локалну базу података. Овакав приступ је изабран јер крајњи корисници који се налазе на рубу мреже често могу имати проблема са конекцијом на главни сервер или немогућношћу повезивања услед његовог пада, док *fog* сервер не пати од таквих проблема јер се налази у локалној мрежи.

Читав систем састоји се из три модула: детекција отказа, репликација сервиса и делегирање посла резервном чвору. Као *fog* уређај кориштен је *Raspberry Pi 4*.

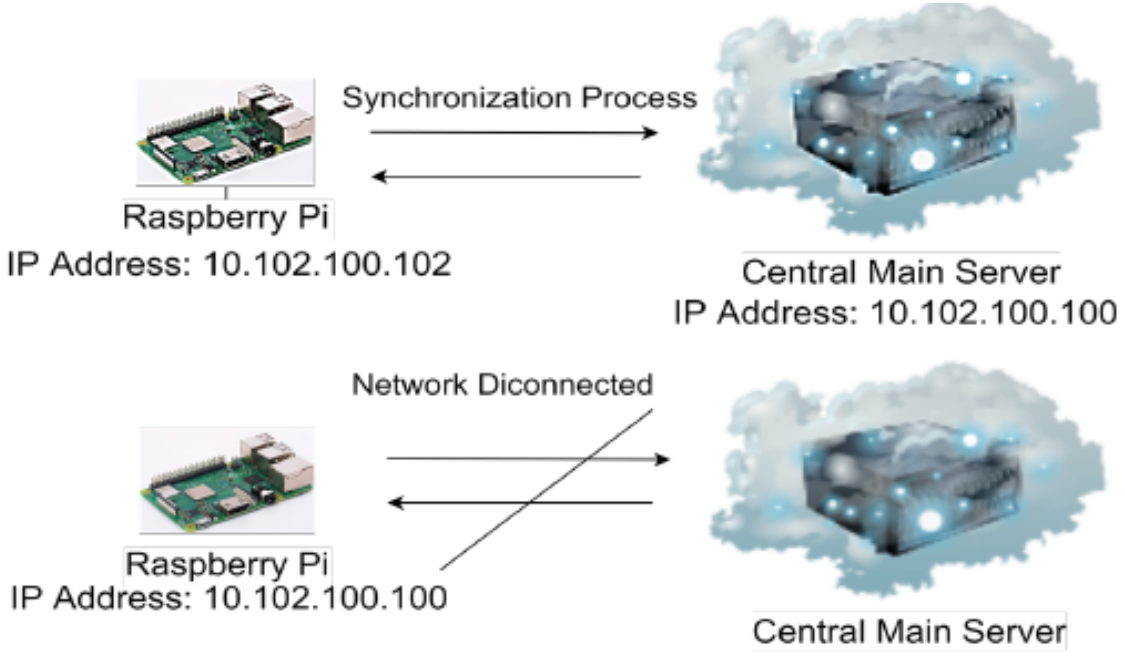
Детекција отказа врши се ослушкивањем *heartbeat* сигнала *cloud* севера. Модул је имплементиран *python* скриптом која на сваких 5 секунди шаље упит на адресу *cloud* сервера.

```
Input: CentralServerAddress,FogAddress
Output: CentralServerRespond
CentralServerRespond ← fnSendHeartbeatMsg
if CentralServerRespond == false then
    FogAddress ← CentralServerAddress
else if CentralServerRespond == true then
    fnUpdateServices
    fnFetchData
```

Слика 9: Код за синхронизацију локалне базе података

Услед ограничених ресурса *fog* уређаја, репликација сервиса своди се само на најважније и неопходне сервисе, што би био *Apache Tomcat* сервис на коме се покреће *backend Java Swing* апликација и *MySQL* база података. Реплицирани сервис представља хладну реплику, што би значило да му се у току нормалног рада апликације корисници уопште не обраћају, све док је могућ приступ примарном сервису. База података синхронизује се са главном базом података на прилично стандардан начин 9, где *python* скрипта периодично шаље упит главној бази података и уколико је она доступна, подаци се ажурирају, а уколико није *fog* бази се додељује адреса налик централној и на тај начин крајњи корисници не примећују било какву промену у случају пада базе података.

Делегирање посла са главног сервера на *fog* сервер врши се по сличном принципу као и синхронизација бази података, где у случају недоступности главног сервера *fog* сервер поставља као своју адресу адресу главног сервера и преузима посао, с тиме да наставља да ослушкује *heartbeat* сигнал главног сервера како би му вратио контролу када се његово стање стабилизује. Исти алгоритам користи се и за делегирање посла у случају преоптерећења главног сервера. 10



Слика 10: Илустрација рада система када је главни сервер у и ван функције

6 Закључак

У овом семинарском раду, истражене су архитектуре система великих скупова података које обухватају *Edge, Fog* и *Continuum computing*. Свака од ових парадигми има своје јединствене карактеристике и примене у модерном дигиталном окружењу.

Edge computing се фокусира на обраду података крај самог извора, што доводи до смањења латенције и бољег одзива за крајње кориснике. Оно омогућава примену специфичних алгоритама и решења на уређајима на месту самих података.

Fog computing нуди додатни слој за процесну обраду између уређаја на рубу мреже и централизованих *cloud* система. Овакав приступ побољшава скалабилност и могућност ресурсно захтевне анализе података пре њиховог слања на *cloud*, што је посебно значајно у условима великог броја података и аналитички захтевних задатака.

Continuum computing представља еволуцију ових парадигми интегришући их у кохезиван динамички систем. Он омогућава усклађивање локализованог *edge*-а, средишњег *fog*-а и централизованог *cloud-a*. Оваква интеграција отвара врата за унапређене могућности обраде података, побољшавање ефикасности мрежа и оптимизацију ресурса.

Захваљујући овим технологијама, предстојећи развој информационих система имаће значајне користи у областима као што су *IoT*, мобилна комуникација, паметни градови, индустрија и здравство што се већ и сад може наслутити узимајући у обзир тренутне имплементације.

7 Библиографија

1. Edge computing, A grounded theory study
2. Edge computing and deployment strategies for communication service providers
3. Edge computing brings data and insight closer to customers
4. An Overview on Edge Computing Research
5. Fog Computing and the Internet of Things: A Review
6. Exploring the Potential of Distributed Computing Continuum Systems
7. Fundamental Research Challenges for Distributed Computing Continuum Systems
8. Cloudlets: Bringing the cloud to the mobile user
9. Fault Tolerance Mechanism for Software Application Through Fog Computing as Middleware