Rezumatul Etapei I

Identificare:

Contract de finanțare nr. 124/01/04.2020
Contractor: Universitatea de Vest din Timișoara (UVT)
Cod: ERANET-CHISTERA3-DIPET
Etapa 1: 1 Aprilie – 30 Octombrie 2020
Nume etapă: Cazuri de utilizare și metrici

Autori ai raportului și membrii ai proiectului:

Lect. Dr. Gabriel Iuhasz (Co-PI)
Dr. Silviu Panica
Lect. Dr. Marian Neagul
Asist.cercetare Alexandru Munteanu

Director proiect:

Prof. Dr. Dana Petcu (PI)

Pagini web:

https://dipet.hpc.uvt.ro/
https://dipet.eeecs.qub.ac.uk/

Obiectivul proiectului

Proiectul DiPET investighează maparea dinamică și transparentă a aplicațiilor de procesare a fluxurilor de date în medii de calcul de tip Fog și Edge și folosește calculul cu trans-precizie cu scopul final de a îmbunătăți aspectele operaționale in termeni de utilizare a resurselor și consum de energie și de a îmbunătății experiența utilizatorului.

Rezultatele estimate ale proiectului

Proiectul DiPET va dezvolta strategii de distribuție care îmbunătățesc performanța aplicațiilor prin utilizarea coalcululului cu trans-precizie pentru a maximiza adaptabilitatea aplicațiilor la resursele disponibile. Această abordare va crea noi oportunități de îmbunătățire a mai multor aspecte ale sistemelor de tip Fog și Edge, precum creșterea eficienței energetice și a performanței acestor sistemelor (se vizează o îmbunătățire a performanței de 2 ori prin trans-precizie și o îmbunătățire a energiei similare). Mai mult, DiPET își propune să simplifice radical programarea, configurarea și desfășurarea procesării fluxului, permițând programatorilor să declare restricții de plasare la resursele dinamice, controlul de planificare și trans-precizie și / sau aproximarea, toate într-o singură specificație concisă. Planificatorul va fi unul în premieră, întrucât în prezent nu există un planificator bun pentru procesarea fluxurilor de date în medii de tip Fog, cu atât mai puțin conștient de trans-precizie, și va oferi transparență de funcționare deopotrivă pentru dezvoltatorii de aplicații și utilizatori. Sistemul DiPET va fi partajat comunității sub licențe open source și distribuit ca software reutilizabil, prin pachete software instalabile. În plus, vor fi dezvoltate cazuri de utilizare, alese dintre domeniile de aplicație relevante, centrate pe utilizator: detectarea intruziunilor în rețea și analiza video distribuită.

Rezumatul Etapei 1, Cazuri de utilizare si metrici

Conform planului de lucru a propunerii focusul principal al etapei raportate este crearea unui cadru științific care va sta la baza cercetării viitoare. În principal munca a fost concentrată în jurul creării unor cerințe clar definite pentru realizarea obiectivelor proiectului. Acest lucru a fost realizat prin elaborarea și proiectarea unor cazuri de utilizare pertinente.

Elaborarea cazurilor de utilizarea în această etapă a fost focusată pe definirea unor metrici și cerințe atât funcționale cât și nefuncționale. Cu ajutorul acestora putem în viitor să quantificăm precis performanțele și beneficiile aduse modelelor predictive create în timpul proiectului prin aplicarea de metode „transprecise".

Au fost definite 5 cazuri de utilizare pentru testare. Dintre acestea echipa UVT oferă suport pentru două dintre ele, iar un caz de utilizare este propus de către echipa UVT. Cazul de utilizare propus de către echipa UVT are la bază unealta creată în timpul unui alt proiect de cercetare numit *ML4EO*[1] finanțat de către Agenția Spațială Europeana (ESA). Unealta numindu-se *HuginEO*[2]. Descrierea amănunțită a acestor cazuri de utilizare este realizată în secțiunile următoare.

O cerință clar identificată încă de la scrierea proiectului este necesitatea de monitorizare a aplicațiilor cât și modelarea performanțelor acestora. În această etapă echipa UVT a început implementarea unei unelte de detecție a evenimentelor și anomaliilor de performanță cât și modelarea performanțelor acestora, unealta numindu-se *Event Detection Engine* (EDE)[3].

De asemenea, în cadrul proiectului au fost achiziționate o serie de componente hardware specializate (Nvidia Jetson[4]) pentru testarea uneltelor și a aplicațiilor pe resurse limitate. Acestea permit testarea metodelor „transprecise" create în cadrul proiectelor pe hardware care de regulă se află la limita unei soluții de tip Fog/Edge Computing. Componentele urmează au fost instalate în centrul de calcul alături de infrastructura existentă în UVT[5].

Descriere științifică și tehnică

Cazuri de utilizare

HuginEO

După cum am menționat în rezumat cazul acesta de utilizare este bazat pe unealta HuginEO. Aceasta a fost creată pentru a ușura aplicarea metodelor de tip Deep Learning asupra domeniului teledetecției. Acesta permite procesarea imaginilor satelitare cât și crearea unor modele predictive pentru diverse problematici precum detecția de obiecte, segmentare semantică, clasificare etc. Un alt lucru important de menționat este faptul că HuginEO este capabil sa asimileze date dintr-o serie considerabilă de surse: Local, URL, AWS S3, Google Cloud Storage cât și surse video („video streams").

În contextul DIPET se dorește crearea și optimizarea modelelor bazate pe Deep Learning care să permită rularea acestor modele pe infrastructură limitată din punct de vedere computațional. În principal se dorește crearea unor modele care pot fi instanțiate pe Nvidia Jetson pentru diverse cazuri de utilizare precum segmentarea semantică, detecția de obiecte, urmărirea de obiecte. Este de precizat faptul că nu se dorește

[1] http://sage.ieat.ro/projects/ML4EO/
[2] https://hugin-eo.readthedocs.io/en/latest/
[3] https://github.com/DIPET-UVT/EDE-Dipet
[4] https://developer.nvidia.com/embedded/jetson-nano-2gb-developer-kit
[5] http://hpc.uvt.ro

antrenarea pe aceste resurse de Edge/Fog ci doar execuția părții de inferență. Acestea putând fii instanțiate pe platforme mobile precum drone autonome.

În perioada curentă sau realizat o serie de experimente, rezultatele acestora fiind prezentate și publicate în cadrul conferinței SYNASC 2020. Titlul articolului este „*Optimizing Deep Learning Models for Object Detection*". În acest articol au fost investigate diferite metode de optimizarea a topologiilor unor rețele Deep Learning. Sau folosit operatori „transprecise" precum *minifloat (FP8)* și *versiunea hibridă (HFP8)*. Adițional sau folosit și metode bazate pe algoritmi genetici pentru optimizarea parametrilor unei rețele de tip *YOLOv3*. Domeniul problemei ales a fost unul de actualitate, obiectele detectate fiind reprezentate de persoane dintr-o filmare și măsurarea distanței dintre aceștia. Practic se poate verifica respectarea distanțării sociale în contextul COVID 19.

Guifi Net

În acest caz de utilizare se dorește realizarea unui sistem bazat pe metode de învățare automată (machine learning) pentru detecția de evenimente și anomalii a unui *mesh network*. Acesta făcând parte din Guifi net[6], proiect început în 2004. Fiecare nod este deținut de o persoană și este lega la celelalte noduri fie prin conexiuni terestre fie prin conexiuni Wifi. Aceste noduri interconectate permit ca accesul la internet să fie partajat între noduri. Astfel infrastructura necesară pentru conectarea la internet este una comunitară.

În momentul de față există 36.718 noduri operaționale din 65.872 planificate. Numărul mare de noduri și conexiunile între acestea rezultă într-o rețea extrem de complexă. Fiecare nod din rețea realizează conexiuni între unul sau mai multe noduri. Astfel orice perturbare a rețelei este amplificată.

La momentul începerii proiectului nu exista un sistem de colectare si stocare a metricilor. Astfel sa decis implementarea unei metode de colectare și stocare a datelor asupra unui număr limitat de noduri. Mai precis în momentul de față se colectează date de la 67 de noduri.

Împreună cu partenerii de la Universitat Politècnica de Catalunya (UPC) am reușit să colectăm în medie 120 de metrici cu o rezoluție de 5 secunde între măsurători. Dimensiunea variabilă a metricilor se datorează faptului că nodurile nu sunt omogene. Acestea au hardware de capacități diferite. În plus numărul conexiunilor este si ea variabilă între noduri. Unele noduri se pot considera noduri frunză iar altele având 20+ conexiuni active. Pentru rutarea traficului între noduri dintr-un *mesh network* precum Guifi net este necesară folosirea unui protocol de rutarea specializat care poate să facă față schimbărilor neprevăzute în conectivitatea nodurilor.

Ținând cont de caracteristicile enumerate mai sus, crearea unui set de antrenare și testare a unor modele predictive este un demers extrem de complicat, necesitând o cunoștință aprofundată a infrastructurii. Astfel a există o colaborare strânsă între echipa UVT și cea de la UPC. Sa realizat o analiză aprofundată a metricilor și a contextului dat de acestea.

Împreună cu partenerii de la UPC urmează să realizăm o serie de experimente pentru detecția anomaliilor contextuale. În prima fază experimentele se vor focusa pe detecția acestor anomalii folosind metode nesupervizate. În principiu experimentele vor încerca creare de modele predictive capabile să prezică parametri normali la care ar trebui să funcționeze Guifi net, orice valori care sunt diferite de ce a prezis modelul creat vor fi considerate anomalii. Următoarea fază va implica crearea unor seturi de date care să conțină diferite clase de anomalii. Scopul final este crearea unor modele predictive folosind metode „transprecise" care pot fi instanțiate pe nodurile din Guifi net, limitând necesitatea de crearea a unei infrastructuri centralizate de monitorizare si analiză a datelor.

[6] https://guifi.net/

Este important de menționat participarea echipei UVT la o serie de ședințe legate și de un alt caz de utilizare pentru detecția de evenimente și anomalii legate de o rețea cu o infrastructură tradițională (de tip HPC/Cloud). În acest caz se dorește identificarea evenimentelor de securitate (*intrusion detection*). În prezent există limitări legate de GDPR, datele obținute până în momentul de față necesită anonimizare. Astfel o analiză similară cu cea realizată pentru Guifi net nu a putu fi realizată până în momentul de față. Anonimizarea datelor necesitând mai mult timp.

Soluția software care va fi folosită pentru procesarea datelor obținute și crearea de modele predictive în cazul ambelor cazuri de utilizarea va fi descrisă în secțiunea următoare.

Event Detection Engine (EDE)

EDE este bazat pe o unealtă dezvoltată pentru sisteme Exascale în cadrul proiectului H2020 ASPIDE, cu funcționalități extinse pentru a permite funcționarea uneltei în contextul DIPET. Mai precis posibilitatea de a folosii metode „transprecise" cât și distribuirea proceselor de antrenare și a predicției pe noduri de tip Edge/Fog. Folosirea acestui component și în cazuri de utilizare reprezintă doar o primă etapă în implementarea acestor cazuri de utilizare. Modelele create cu ajutor EDE vor fi integrate într-o soluție separat implementată. Similaritatea între problematica platformei DIPET (Pachetul de lucru 3 din propunere) și soluția implementată de echipa UVT (EDE) și a cazurilor de utilizare a permis acest lucru fără modificări majore.

Arhitectura EDE care constă în 5 componente distincte care sunt prezentate în secțiunile următoare.

Data Ingestion

Această componentă este folosită pentru ingestia de date din diferite surse. Momentan sunt suportate 2 tipuri de surse persistente; ElasticSearch[7] și Prometheus[8]. Amândouă tehnologiile sunt des utilizate în soluții de monitorizare. EDE poate prelua în mod automat date din aceste surse prin generarea interogărilor necesare în mod automat. În plus există suport pentru încărcarea de date locale în diferite formate, precum: HDF5[9], CSV și JSON.

Prepocessing

Datele provenite din diverse surse sunt în cele mai multe cazuri incompatibile cu soluții de învățare automată, astfel necesită diverse preprocesări. Această componentă este responsabilă cu formatarea datelor, împărțirea datelor (ex. set de antrenarea, validare, holdout), metode de analiză și explorarea a datelor (rezultatul fiind comparabil cu cazul de utilizare Guifi net), augmentarea datelor (folosind metode precum ADASYN sau SMOTE).

Aceste preprocesări sunt implementate în Python folosind la baza librăria scikit-learn[10], Pandas[11] NumPy[12], SciPy[13] etc. Pentru a aplica metode de tip „transprecise" în cadrul proiectului DIPET sau adăugat diverse optimizări bazate pe librăria Intel MKL[14]. Astfel a fost necesară modificarea codului EDE pentru a suporta

[7] https://www.elastic.co/
[8] https://prometheus.io/
[9] https://www.hdfgroup.org/solutions/hdf5/
[10] https://scikit-learn.org/
[11] https://pandas.pydata.org/
[12] https://numpy.org/
[13] https://www.scipy.org/
[14] https://software.intel.com/content/www/us/en/develop/tools/math-kernel-library.html

„run time”-ul Python[15] bazat pe Intel MKL care permite aceste optimizări ale performanței librăriilor științifice mai sus menționate.

Prin utilizarea funcționalităților expuse de către librăria *PyYAML.load*[16] este posibilă încărcarea de funcții arbitrare definite de către utilizator. Aceste funcții pot fi preprocesări, metode de antrenare, postprocesări sau metode de analiză. Astfel se pot adăuga extrem de ușor metode noi (inclusiv cele „transprecise”) de către utilizator. Singura cerință este ca funcțiile noi definite să fie encapsulate de funcții de tip *wrapper.* Scopul acestora este de a păstra o formă comună a intrărilor și ieșirilor fiecărei funcții. Astfel ele pot fi adăugate in „pipeline”-ul de execuție EDE. Dacă funcțiile definite nu sunt conforme cu cerințele de encapsulare acestea nu sunt executate din motive de siguranță.

Antrenarea

Pentru modul de antrenare din EDE au fost necesare modificări semnificative pentru a permite rularea de metode compatibile în contextul DIPET. Similar cu preprocesarea, antrenarea a fost modificată ca să fie compatibilă cu librăria și „run time”-ul Python din Intel MKL, aducând aceleași beneficii. Similar, si encapsularea metodelor de învățare automată prin mecanismul bazat pe facilitățile *PyYAML.load* este posibilă.

Metodele „transprecise” necesită utilizarea unor funcții de evaluare speciale. Aceste noi metode de evaluare trebuie să poată evalua nu doar precizia metodelor predictive dar și metrici precum, energia folosită, resursele hardware folosite, dimensiunea modelului, timpul de inferență etc. Astfel mecanismul bazat pe facilitățile *PyYAML.load* este folosit pentru a definii aceste noi funcții de evaluare.

De cele mai multe ori timpul de inferență, dimensiunea, resursele hardware utilizate ale modelelor predictive sunt influențate de parametrii setați în timpul antrenării. Astfel am adăugat metode de optimizare a parametrilor („hyper-parameter optimization”) care pot fi folosite împreună cu funcțiile de evaluare.

Acest moment este oportun pentru a amintii de faptul că toate componentele EDE sunt executate folosind librăria Dask[17]. Aceasta permite executarea EDE în mod distribuit. Astfel este posibilă executarea componentei de ingestie, preprocesare, antrenarea și predicție pe noduri diferite. Acest lucru este extrem de util în cazul metodelor de optimizare. Acestea în prima fază generează configurații candidat pe urmă executându-le. Această execuție este în general una secvențială dar în cazul EDE poate fii executată ușor în paralel folosind Dask, astfel scurtând semnificativ timpul de optimizare.

Prediction

Componenta de predicție are ca și scop instanțierea metodelor predictive și analizarea datelor de monitorizare. Modificările aduse acestei componente în contextul DIPET sunt legate de modul în care sunt planificate instanțele metodelor predictive. După cum sa menționat mai sus componentele EDE sunt executate folosind librăria Dask. Cu ajutorul acesteia este posibilă planificarea execuției pe noduri desemnate de către utilizator. Astfel se poate realiza selecția metodelor predictive bazat pe capacitățile hardware-ului existent (reprezentat de Dask workers). Mai exact, se poate specifica execuția unei metode cu cerințe hardware minimale pe hardware limitat precum Nvidia Jetson sau RaspberyPi.

EDE este folosit și pentru analiza datelor din cazul de utilizare Guifi net și cel de „intrusion detection”. Datorită faptului că metodele de învățare automată sunt aproape în totalitate bazate pe „batch processing”,

[15] https://software.intel.com/content/www/us/en/develop/tools/distribution-for-python.html
[16] https://github.com/yaml/pyyaml
[17] https://dask.org/

componentele EDE pot fi folosite fără modificări. În schimb componenta de predicție trebuie să suporte si modul de „stream processing" via Apache Flink[18]. În momentul de față acest lucru nu este implementat dar există posibilitatea exportării modelelor și instanțierea lor manuală peste Flink.

Platforme de prototipare

În cadrul proiectului DIPET se dorește crearea unei soluții care permite execuția unor aplicații de „stream processing" bazate pe metode „transprecise". Tehnologiile selectate pentru realizarea acestei soluții sunt bazate pe Apache Flink și Kubernetes[19]. Aceste tehnologii trebuie să fie configurate și instalate inclusiv pe hardware care constituie o soluție de tip Edge/Fog.

Pentru a ușura testarea și integrarea solutiilor dezvoltate, echipa UVT a creat două platforme de prototipare. Prima platformă[20] este una bazată pe Vagrant[21]. Aceasta permite crearea locală a 4 mașini virtuale care conțin platforma Kubernetes peste care se instanțiază folosind containere Docker Apache Flink. Monitorizarea parametrilor se realizează folosind Prometheus.

A doua platformă de prototipare este una bazată pe Nvidia Jetson. Cele două platforme sunt similare, singura diferență majoră este folosirea unei imaginii docker care permite accesarea capacităților GPU a hardware-ului. În momentul de față folosim imaginea *nvidia-docker*[22]. În momentul de față este creat un cluster bazat pe Kubernetes cu 5 noduri (4 workeri si un master). Urmează a fi integrată cu infrastructura existentă UVT și extinsă cu încă 5 noduri Nvidia Jetson.

Diseminare

Rezultatele etapei au fost parțial publicate într-o lucrare de conferință internațională (in print, indexată WoS Proceedings) și o platformă software (în GitHub). Co-laterale sunt o teză de master și o lucrare într-o revistă (open-access, indexată WoS ESCI).

Decembrie 1, 2020

[18] https://flink.apache.org/
[19] https://kubernetes.io/
[20] https://github.com/DIPET-UVT/dipet-test-cluster
[21] https://www.vagrantup.com/
[22] https://github.com/NVIDIA/nvidia-docker