Aplicații virtuale Cloudifier [1]

Aplicații virtuale pentru aplicații de analiză predictivă pentru desktop, bazate pe cadrul de procesare a GPU-urilor

Andrei Ionut DAMIAN *(Autor)*
Cloudifier SRL
București, România
Damian@cloudifier.net

Alexandru PURDILA *(Autor)*
Cloudifier SRL
București, România

alex@cloudifier.net Nicolae TAPUS *(Autor)*
Universitatea "Politehnica" București
București, România
Ntapus@cs.pub.ro

***Rezumat* - Nevoia de sisteme capabile de a efectua analiza inferențială și analiză predictivă este omniprezent într - o societate informațională globală. Odată cu progresele recente din domeniul modelelor de învățare automată a mașinilor de predicție și al calculului masiv paralel, este acum disponibil un nou set de resurse pentru comunitatea științelor informatice pentru a cerceta și dezvolta noi aplicații cu adevărat inteligente și inovatoare. În lucrarea de față prezentăm principiile, arhitectura și rezultatele experimentale actuale pentru o platformă on - line capabil de ambele aplicații *inteligente* hosting și generatoare.**

Cuvinte cheie - inteligență artificială, învățare automată, desktop virtual, analiză predictivă

I. INTRODUCERE

Întregul câmp al Inteligenței Artifice tinde să devină tot mai mult despre îmbunătățirea predicțiilor și a modelelor predictive pe care Goldfarb și alții [1] susțin în lucrarea "Gestionarea mașinilor". Modelarea predictivă poate fi acum utilizată pentru sarcini extrem de variate, de la construirea unui sistem expert de chat-bots [2] bazat pe rețelele neuronale recurente de lungă durată pe termen scurt care ar folosi capacitatea de scriere a limbajului natural până la recunoașterea melanomului în imaginile dermoscopice [3].

În lucrarea noastră, vom propune mai multe abordări: (i) un model bazat pe servicii bazate pe servicii de tip web pentru implementarea modelului de pregătire și predicție pe baza backend-ului masiv de calcul paralel al GPU și (ii) angajarea de proiectare, dezvoltare și implementare automată

[1] Cercetare sponsorizată de Cloudifier SRL pe baza Axei 1, Finanțarea Programului Operațional pentru Competitivitate.

a software- Bazat pe un model de învățare a mașinilor superficiale [4] și (iii) o arhitectură unificată și un cadru de bază pentru generarea și găzduirea aplicațiilor de analiză predictivă.

În cadrul lucrărilor, vom susține că cercetarea și experimentarea propuse vor susține dezvoltarea și implementarea unui sistem de producție capabil să răspundă în același timp nevoilor mai multor categorii de clienți, cum ar fi:

- Utilizatorii care au nevoie de o piață de aplicații online orientată pe aplicații de analiză predictivă;
- Dezvoltatorii de software care au nevoie de instrumente de traducere automată a aplicațiilor automate, inclusiv șabloane de învățare automată;
- Utilizatorii științifici din mediul academic și comercial care caută un cadru de aplicație online și un mediu de execuție pentru a efectua experimente predictive în domenii precum sănătatea, finanțele și mediul;

II. LUCRU ÎNRUDIT

Activitatea noastră se referă la cele mai recente progrese atât în domeniul calculului de înaltă performanță, cât și al învățării în mașină.

A. GPU bazate pe computere de înaltă performanță

Utilizarea computerelor bazate pe GPU este în mare parte aplicată în procesul de învățare profundă - atât experimentarea cercetărilor, cât și dezvoltarea producției - oricât de puțin se acordă atenție folosirii resurselor de calcul paralel GPU în tehnicile de învățare superficială a mașinilor, cum ar fi arborii decizionali, Și regresie logistică, modele Naive-Bayes. Cu toate acestea, progresele actuale și scorurile de predicție de ultimă generație obținute prin experimente științifice profunde de învățare dau puțin loc pentru dezvoltarea științifică și avansarea tehnicilor cunoscute de modele superficiale. Chiar și mai mult, companii precum Microsoft, Google sau IBM propun în prezent motoare-cadru on-line care să permită crearea și experimentarea pe internet a modelelor de predicție bazate pe web [5], utilizate în întregime pentru experimentele de formare off-line și predicție, Infrastructuri. Luând în considerare aceste aspecte prezentate, propunem un set de noi abordări moderne, care ar putea împuternici potențialul modelelor de învățare a mașinilor superficiale, cu capacitatea de utilizare a resurselor de calcul paralele bazate pe GPU. De asemenea, un alt aspect al asistenței pe care îl vom folosi în abordarea noastră se bazează pe cele mai importante caracteristici ale modelelor de învățare mecanică superficială: abilitatea modelului de a se auto-explica (deși în detrimentul puterii de predicție a modelelor profunde cu Masive ascunse și spații de greutate mai puțin explicative).

B. Arborele arhitectural

Arhitectura propusă se bazează pe activitatea noastră anterioară desfășurată în domeniul migrării automate a aplicațiilor software și al programării automate asistate automat [4]. Bazat pe arhitectura noastră automată de migrare software [4] vom fi ab le pentru a realiza următoarele deziderate în timpul fazei de producție a proiectului nostru:

- Creați o arhitectură scalabilă pentru un mediu online ușor de utilizat, în care utilizatorii externi vor putea să genereze și să personalizeze rapid aplicațiile pornind de la aplicațiile vechi existente;
- Combinați puterea caracteristicilor de generare automată a aplicațiilor cu modele bazate pe șabloane pentru a crea un depozit de aplicații orizontale pentru afaceri care să

includă aplicații cum ar fi (a) modele multi-verticale de predicție a clienților; (B) modelul de inferență a comportamentului multi-vertical; (C) aplicații de explorare a datelor generate de fluxul de afaceri; (D) aplicații pentru învățarea mașinilor / șablon

- În cele din urmă, suntem capabili să creăm un depozit de aplicații de cercetare care să crească în mod continuu prin adaptarea aplicațiilor augmentate de învățare în mașină în domenii precum inferența / predicția diagnosticului medical, analiza predictivă financiară, modelele inferențiale pentru mediul înconjurător. Aplicațiile științifice propuse vor fi dezvoltate în cadrul comunității "Cloudifier" și vor utiliza pe deplin modelele scalabile de modele de învățare a mașinilor scalabile de GPU;
-

III. ARHITECTURA PROPUSĂ

Principalele puncte ale arhitecturii propuse sunt următoarele:

A. Cadru general de execuție bazat pe serviciile GPU HPC REST;

B. Extinderea aplicațiilor de învățare a mașinilor generate într-o infrastructură Cloud provizorie prin migrarea automată a software-ului [4];

C. Aplicații virtuale de analiză predictivă a aplicațiilor desktop, care constau în aplicații de afaceri dezvoltate de șablon și experimente științifice;

A. REST bazate pe servicii masive de calcul paralel

Bazându-ne pe faptul că unul dintre domeniile noastre de inovare este augmentarea modelelor de învățare mecanică superficială cu tehnici de procesare paralelă bazate pe procesarea GPU, propunem un serviciu bazat pe REST pentru calculul paralel pe backend-ul GPU. În prezent, pe baza cunoștințelor noastre, avansarea infrastructurilor hardware bazate pe GPU capabile să furnizeze computere masive paralele pentru sarcini de învățare în mașină la costuri foarte mici a fost exploatată doar de domeniul învățării profunde și a rețelelor neuronale profunde în special. Propunerea noastră abordează acest decalaj între resursele de calcul disponibile pentru învățarea profundă și cele disponibile pentru învățarea științifică superficială științifică. În abordarea noastră propusă pentru avansarea procesului de învățare a mașinilor superficiale prin amplificarea computerizată paralelă cu GPU vom susține că conductele modelului superficial pot obține rezultate comparabile cu modelele de învățare profundă pentru diverse aplicații. Deși majoritatea cercetărilor de ultimă oră în domeniul învățării în mașină se axează pe modele de învățare profundă, modelele clasice de învățare a mașinilor cu o structură superficială ascunsă a straturilor, care sunt două caracteristici principale pe care le lipsesc modelele adânci: capacitatea de auto-explicare și robustețea algoritmică Pentru medii online și în timp real.

În prezent, sunt disponibile mai multe opțiuni de programare pentru programarea motoarelor de calcul paralel GPU atât pentru scopuri științifice, cât și pentru inginerie / producție și probabil cele mai răspândite sunt OpenCL [6] și CUDA [7]. Practic, ambele modele menționate utilizează aceeași tehnică, cu doar mici diferențe. Cele două aspect fundamental în ceea ce privește abordarea de calcul constă în cele două entități centrale constând în *kernel - ul* - care este funcția care conține codul actual , care urmează să fie executate în paralel - și *dispozitivele* - dispozitivele de calcul reale care vor rula *kernel - ul* în paralel . În mod tradițional *kernel - ul* este o funcție descrisă de următorul algoritm generic:

Algoritmul 1 *Kernel (Data, BlockInformation)*

cid ∪ get coordonatele tuplu de la *BlockInformation*
Modificare *date* [Cid] cu greu codificate <*operație*>
Întoarcere

De notat că , în exemplul de mai sus *BlockInformation* definește segmentul de date reale pe care un anumit *dispozitiv* este de procesare în cadrul *datelor* și <*operație*> denotă operația codificate (compilat) că *kernel - ul* este de calcul pentru blocul particular al *datelor* primite. Pentru cazul particular al produsului dot matrice putem avea fiecare compute *dispozitiv* cu *kernel* produsul scalar a doi vectori (vectorul rând de 1 matrice st și vectorul coloană de 2 matrice nd).

(1)

În cele din urmă, avem următoarele două fragmente de cod pentru *nucleele* de calcul (1) , în OpenCL și CUDA pe bază de :

Cod 1 OpenCL Kernel pentru Matrix-Matrix produs scalar

```
__kernel void OpenCLMatrixDotKernel (const int A_ROWS,
  Const int BC_COLUMNS,
  Const int A_COLUMNS,
  Const __global float * A,
  Const __global float * B,
__ float global * C)
{{
Const int Row = get_global_id (0);
Const int Col = get_global_id (1);
Float acc = 0.0f;
Pentru (int c = 0; c <A_COLUMNS; c ++) {
Acc + = A [c * A_ROWS + rând] * B [Col * A_COLUMNS + c];
}
C [Col * A_ROWS + Row] = acc;
}
```

Cod 2 CUDA Kernel pentru Matrix-Matrix produs scalar

```
__global__ void CUDAMatrixDotKernel (Matricea A, Matricea B, Matricea C)
{{
        Float acc = 0;
        Int rând = blocIdx.y * blockDim.y + threadIdx.y;
        Int col = blocIdx.x * blockDim.x + threadIdx.x;
        Dacă (rând> A.height || col> B.width)
                întoarcere;
        Pentru (int e = 0; e <A.width; ++ e)
                Acc + = elemente A. [rând * A.width + e] *
                        B. elemente [e * B. lățime + col];
        C. elemente [rând * C.width + col] = acc;
}
```

Așa cum am menționat anterior, în lucrarea de față vom prezenta o abordare REST [8] pentru furnizarea bazată pe cloud computing a serviciilor de calcul paralel. Această abordare va permite implementarea și consumul unui serviciu web care va avea următorul algoritm generic bazat pe alocarea FIFO.

Algoritmul 2 *ParallelProcessingREST (Request)*

Pentru fiecare Kernel, perechea de date din Solicitare
Kernel αυα lua nucleu codul sursă de la *Cerere*
Datele de la ⇐ obține date CCD *Solicitare* de valoare sau prin referință la sursa externă
Disponibil ⇐ obține dispozitive disponibile de la cluster existent pe baza **algoritmului** adaptiv 3
Dacă este *disponibil* este valid apoi
Off-încărcare *Kernel* și *date* pe dispozitive *disponibile*
Ia - off-încărcare *Rezultat* și adăugați la *răspunsul*
Altfel

Răspuns ⇐ timpul Overhead depășește timpul de calcul

Răspuns de **retur**

Strategia actuală de execuție a unităților de procesare GPU a nucleelor descărcate se va baza pe un algoritm adaptiv care va lua în considerare sarcina serverului, complexitatea sarcinilor și sarcina / pregătirea GPU.

Algoritmul 3 ResourceAllocation *(Kernel, date)*

cFLOPS ⇐ evalua necesare FLOPS pentru *kernel - ului* și a *datelor*
Resursele ⇐ Compute necesare dispozitive bazate pe *cFLOPS*
ComputeTime ⇐ Aproximate timpul total de execuție pe baza *resurselor, cFLOPS*
WaitTime ⇐ Off-timpul de încărcare deasupra capului + timp până când dispozitivele sunt disponibile
Dacă *ComputeTime*> *WaitTime*
Rezultatul poziției ⇐ Coadă sau reale *dispozitive* alocate
Altceva
Alocarea *Rezultatul* ⇐ invalidată
Rezultatul de **retur**

B. Aplicații utilizator - repository de migrare și dezvoltare rapidă

Aplicațiile create de utilizator vor consta atât din aplicații generate, cât și din aplicații bazate pe șabloane. Aplicațiile generate vor fi efectiv rezultatele motorului nostru automat de migrare a software-ului [4], găzduit pe deplin în mediul de execuție propus de proiectul nostru.

Aplicațiile bazate pe șabloane vor consta în principal din aplicații online "schelet" care au fost proiectate pentru un anumit scop orizontal în domeniul analizei predictive. Utilizatorul va putea să configureze un anumit șablon sau să propună un nou șablon pentru publicarea online și testarea mulțimii.

C. Aplicații de afaceri și experimente în cercetare

Ca și în depozitul de aplicații utilizator, aplicațiile de afaceri și depozitele de experimente de cercetare vor găzdui diverse aplicații online de analiză predictivă care vor putea rula modelele de învățare a mașinilor propuse în cadrul procesorului de procesare GPU propus.

IV. EXPERIMENTARE ȘI CONCLUZII

În prezent desfășurăm activități de experimentare în următoarele domenii:

- Migrarea și furnizarea automată a software-ului moștenit în cadrul infrastructurii cloud propuse. A fost construit un set de aplicații gata de migrare bazate pe desktop, constând în aplicații scrise în FoxPro, Visual Basic, Visual C ++. Pentru acest set special de aplicații folosim software-ul nostru de migrare automată și atât codul sursă existent al aplicației moștenite. Codul sursă este necesar pentru validarea reală a procesului inferențial de migrare automată rezultat.

- Analiza aplicatiilor orizontale pentru mediul de afaceri. În prezent, experimentele de analiză predictivă a comportamentului clienților sunt efectuate în cadrul cadrului nostru de învățare automată propus de GPU. Validarea încrucișată și benchmarking-ul rezultatelor se realizează utilizând următoarele cadre științifice și generale de învățare automată:
  - Python Sci-Kit Machine Learning pachet
  - Pachetul R "caret" (găzduit de CRAN)

V. BIBLIOGRAFIE

[1] A. Agrawal, J. Gans și A. Goldfarb, "Gestionarea Roboților" *HBr,* 2016.

[2] Vinyals, "Un model Neural conversațională," în *ICML adânc* de *învățare Workshop 2015,* 2015.

[3] C. V, Q.-B. Nguyen și S. Pankanti, „Ansambluri de învățare profundă pentru Recunoașterea Melanomul în dermatoscopie Imagini,“ *Jurnalul de Cercetare și Dezvoltare*, 2016.

[4] NT AI Damian, "Model Arhitectură pentru traducere automată și Migrației Legacy aplicații Cloud Computing medii," în *CSCS,* București, 2017.

[5] M. Bihis și S. Roychowdhury, "Un flux generalizat pentru multi-clasă și sarcini de clasificare binare: O abordare Azure ML," în *Big Data (Big Data), 2015 IEEE International Conference on* 2015.

[6] J. Stone, D. Gohara și G. Shi, "OpenCL: O programare paralelă standard pentru heterogeni Computing Systems" , *în Computing Science & Engineering,* vol. 12, nr. 3, 2010.

[7] J. Ghorpade, J. Parande și M. Kulkarni, "PROCESAREA GPGPU IN ARHITECTURA CUDA," *Advanced Computing: An International Journal (ACIJ),* vol. 3, 2012.

[8] RT Fielding, "Stiluri arhitecturale și proiectarea arhitecturilor software bazate pe rețea", University of California, Irvine, 2000.

Proiect ID: P_38_543, Nr. Ctr. 98 / 09.09.2016, MySMIS: 104349, Apel - POC-A1-A1.2.1-C-2015,