Spiking Neural P systems on CUDA

Github

Huma Stefan-Dorian
Sanda Marian-Tiberiu
Vulpoi Gabriel
Ivascu Ioan-Andrei
Florescu Bogdan-Ilie

Cuprins

Introducere

Sisteme Neuronale cu Spikes (SNP Systems)

- Modele computaționale inspirate de functionarea neuronilor biologici.
- Bazate pe principiile calculului in membrane.
- Utilizeaza impulsuri ("spikes") pentru procesarea informatiei.

CUDA: Calcul Paralel pe GPU-uri

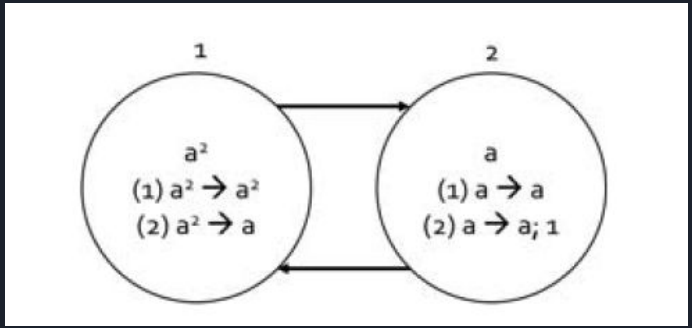
- CUDA este o platforma de calcul paralel dezvoltata de NVIDIA.
- Permite procesare rapida prin unitati grafice (GPU-uri).
- Ideala pentru accelerarea simularilor complexe si proceselor computationale.

Impulsurile Neuronale & Modele Computationale

SNP systems simuleaza modul in care neuronii biologici transmit impulsuri electrice, procesand informatia prin evenimente discrete numite "spikes".

Fiecare neuron urmeaza reguli specifice pentru a determina momentul emiterii impulsurilor, asigurand o procesare eficienta a datelor.

Aceste modele sunt folosite in cercetarea AI, retele neuronale si recunoasterea de modele datorita eficientei lor in gestionarea datelor secventiale si dinamice.

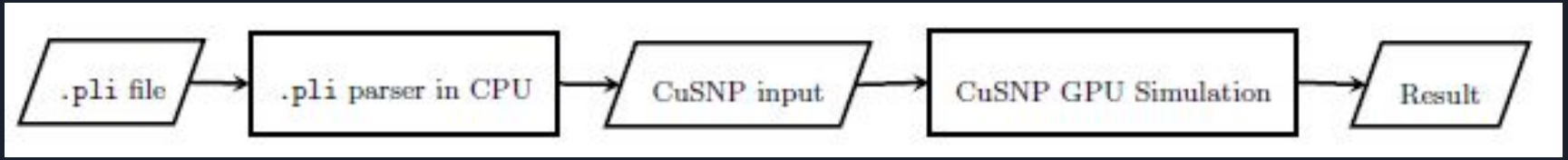


De ce GPU-uri & CUDA?

Simularea sistemelor SNP necesita calcule intense, deoarece modelele de mari dimensiuni presupun procesarea simultana a multor neuroni si impulsuri.

CPU-urile proceseaza datele secvential, ceea ce le face mai putin eficiente in gestionarea operatiunilor masiv paralele.

CUDA, o platforma de calcul paralel de la NVIDIA, foloseste puterea GPU-urilor pentru a executa mai multe operatii simultan, imbunatatind semnificativ performanta simularilor SNP.



Obiectivul Proiectului

- Dezvoltarea unui simulator pentru SN P Systems.
- Implementare pe CPU si GPU folosind CuPy pentru accelerare.
- Crearea unui parser pentru fisiere .snps.
- Analiza comparativa CPU vs GPU pe retele de diferite dimensiuni.

Arhitectura Proiectului

Structura proiectului:

- **src/**: Cod sursa (Neuron, Synapse, SNSystem)
- **cuda/**: Kernel CUDA scris cu CuPy
- **tests/**: Teste de diferite forme pentru verficarea sistemului + testare CPU/GPU
- **docs/**: Documentatie + plot-uri

Implementare SN P System

Clasa Neuron:

- Memoreaza numarul de spike-uri si regulile neuronului.
- Poate primi spike-uri si aplica reguli de firing.
- tick(): executa un pas de simulare pentru neuron.

Clasa Synapse:

- Leaga un neuron sursa de un neuron tinta. (bidirectional)

Clasa SNSystem:

- Gestioneaza toati neuronii si sinapsele.
- Permite rularea sistemului atat pe CPU, cat si pe GPU.
- Functii principale: tick_cpu(), tick_gpu(), run(), load_from_file().

Accelerare CPU (CUDA)

- Kernel CuPy scris manual pentru aplicarea regulilor in paralel.
- Fiecare neuron este procesat in paralel pe GPU.
- Crestere majora de performanta pentru retele mari.

```
import cupy as cp

apply_rules_multi_kernel = cp.RawKernel(r'''
extern "C" __global__
void apply_rules_multi(long long* spike_counts, const long long* thresholds, const long long* consumes,
                       const long long* produces, const long long* delays, long long* fire_counts, long long num_neurons)
    long long idx = blockDim.x * blockIdx.x + threadIdx.x;
    if (idx >= num_neurons) return;

    long long spikes = spike_counts[idx];
    long long threshold = thresholds[idx];
    long long consume = consumes[idx];

    if (spikes >= threshold && spikes >= consume) {
        long long times = spikes / consume;
        spike_counts[idx] -= times * consume;
        fire_counts[idx] = times;
    } else {
        fire_counts[idx] = 0;
    }
}
```

Parsarea fisierelor .snps

Formate de fisiere:

- Sectiune neuroni (*N)
- Sectiune sinapse (*S)
- Incarcare automata in obiecte Python.

```python
n1 = Neuron("N1", spike_count=1,verbose=True, rules=[
    {"consume": 1, "produce": 1, "delay": 1, "condition": lambda x: x >= 1}
])
n2 = Neuron("N2",verbose=True, rules=[
    {"consume": 1, "produce": 1, "delay": 1, "condition": lambda x: x >= 1}
])
n3 = Neuron("N3",verbose=True)

sn = SNSystem(verbose=True)
sn.add_neuron(n1)
sn.add_neuron(n2)
sn.add_neuron(n3)
sn.add_synapse(Synapse("N1", "N2"))
sn.add_synapse(Synapse("N2", "N3"))
```

```
#
*N
N1 1 1 1 1 1 1 1
N2 0 1 1 1 1 1 1
N3 0 1 0
*S
N1 N2
N2 N3

```

Teste Unitare

Teste pentru:

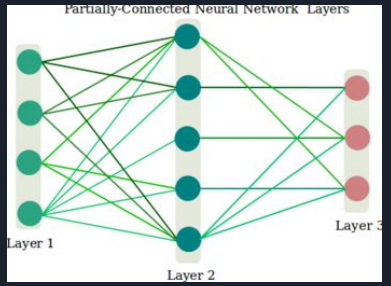
- Test initializare neuron
- Test primire spike neuron
- Test tick neuron: livrare spike imediat
- Test aplicare regula neuron: consum si productie
- Test regula neuron neaplicata daca conditia nu e indeplinita
- Test creare sinapsa
- Test adaugare neuron si sinapsa in sistem
- Test tick CPU: transmitere spike-uri intre neuroni
- Test rulare mai multe tick-uri si salvare istoric spike-uri
- Test incarcare fisier invalid in SNSystem

```
unit_tests.py .........                                                  [100%]

============================================================ 10 passed in 1.55s ============================================================
```

Benchmark-uri CPU vs GPU

Idee: Partially connected neural network



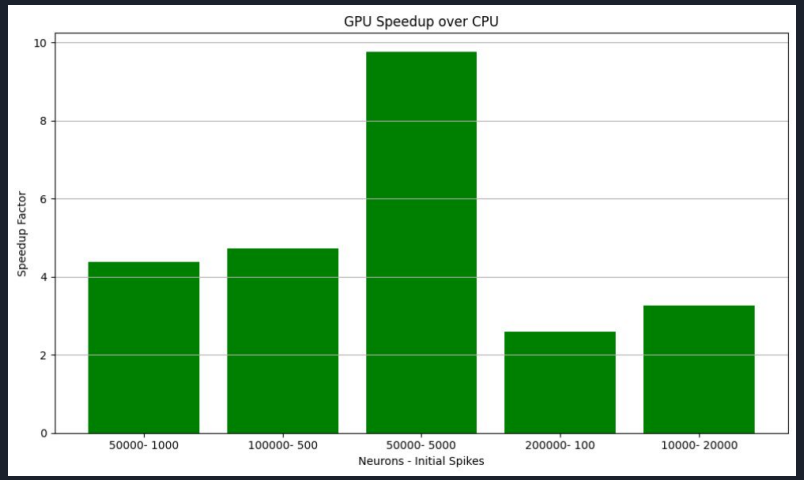
Neurons	Initial Spikes	Connection Gap	Ticks	CPU Time (s)	GPU Time (s)	Speedup (GPU/CPU)
50000	1000	10	5	45.1735	10.3243	4.3754
100000	500	20	5	91.8732	19.4763	4.7171
50000	5000	10	10	102.8080	10.5440	9.7503
200000	100	50	3	75.0574	28.9639	2.5914
10000	20000	2	5	13.1474	4.0302	3.2621

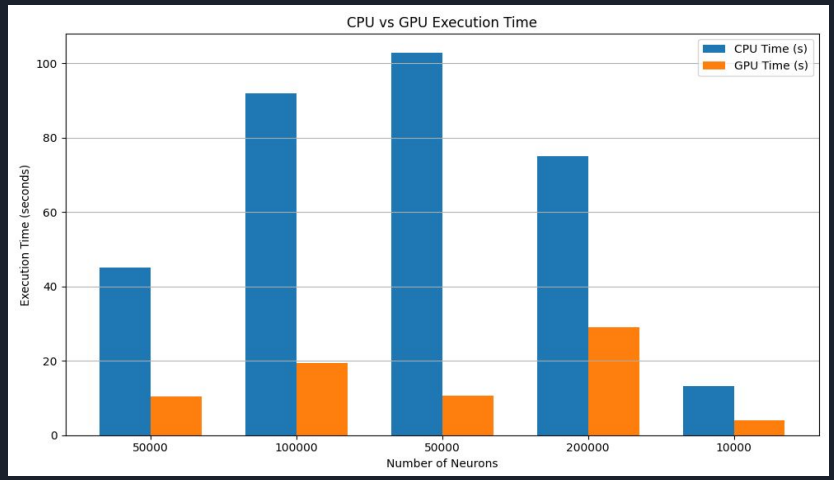
Performanta

GPU ofera accelerari considerabile, in special cand:

- Avem multe spike-uri procesate simultan.
- Reteaua are o densitate medie-mare (connection gap mic).

La retele foarte mari (ex: 200k neuroni), overhead-ul devine semnificativ, reducand usor speedup-ul.





Referinte

CuSNP: Spiking Neural P Systems Simulators in CUDA

Optimizations in CuSNP Simulator for Spiking Neural P Systems on CUDA GPUs

Improving Simulations of Spiking Neural P Systems in NVIDIA CUDA GPUs: CuSNP