Anyagtudomány GPU-klasztereken: Trendek és lehetőségek

Tóth Gyula, Tegze György, Pusztai Tamás és Gránásy László

MTA Wigner Fizikai Kutatóközpont
Szilárdtestfizikai és Optikai Intézet

GPU nap 2013, Wigner FK, Budapest

Tartalom

1. Kontinuum modellek az anyagtudományban
2. Numerikus megoldási módszerek
3. Implementálás GPU-ra: pro és kontra
4. Módszertani optimalizálás
5. Multi-GPU megoldások

Kontinuum modellek az anyagtudományban

A vizsgálatok célja

Elsőrendű fázisátalakulások, komplex kristályos morfológiák (pl. polikristályos megszilárdulás), általános mintázatképződés (fázisszeparáció, kémiai mintázatok, reakció-diffúzió folyamatok, stb.) vizsgálata NEM atomisztikus módszerekkel.

Kontinuum modellek

A rendszer viselkedését **erősen nemlineáris, magasabbrendű, csatolt** (parabolikus, elliptikus, hiperbolikus) parciális differenciálegyenlet-rendszerek írják le .

Megoldási lehetőségek

- Analitikus megoldás: a legegyszerűbb esetekben, 1 dimenzióban
- Altalában numerikus megoldás

Véges differencia/térfogat vs spektrális módszerek I.

Véges differencia(térfogat) (FD/FV)

$$\frac{\partial \phi(x,t)}{\partial t} = \frac{\partial^2 \phi(x,t)}{\partial x^2} + f_{nl}[\phi(x,t)]$$

- Operátor diszkretizálás (stencil):

$$\frac{\partial^2 \phi}{\partial x^2} \approx \frac{\phi_{i-1}^t + \phi_{i+1}^t - 2\phi_i^t}{\Delta x^2}$$

- Diszkretizálás időben:

$$\frac{\partial \phi}{\partial t} \approx \frac{\phi_i^{t+\Delta t} - \phi_i^t}{\Delta t}$$

- Időléptetési séma:

$$\phi_i^{t+\Delta t} = \phi_i^t + \frac{\Delta t}{\Delta x^2}(\phi_{i-1}^t + \phi_{i+1}^t - 2\phi_i^t)$$

Szomszédos értékek használata!

(Legalább részben implicit) spektrális

$$\frac{\partial \phi(k,t)}{\partial t} = -k^2 \phi(k,t) + \mathcal{F}_k\{f_{nl}[\phi(x,t)]\}$$

- Térbeli operátor $\rightarrow$ lokális függvény:

$$\frac{\partial^2 \phi(x,t)}{\partial x^2} \rightarrow -k^2 \phi(k,t)$$

- Nemlineáris függvény:

  $\mathcal{F}_k\{f_{nl}[\phi(x,t)]\}$ nem analitikus

- Részben implicit időléptetés:

$$\phi_k^{t+\Delta t} = \phi_k^t + \Delta t \frac{-k^2 \phi_k^t + F_k[f_{nl}(\phi_i^t)]}{1 + \Delta t \cdot k^2}$$

Gyors Fourier-transzformáció (FFT)!

Véges differencia/térfogat vs spektrális módszerek II.

Véges differencia (FD/FV)

Előnyök:

- egyszerű (Taylor-sor, Gauss-tétel)
- egyszerű ellenőrzés
- tetszőleges peremfeltétel

Hátrányok:

- bonyolult implementáció
- stabilitási kritérium:

$$\Delta x' = \Delta x/2 \Rightarrow \tau' = 2^{M+D}\tau$$

  τ: teljes számítási idő
  D: dimenzió, M: operátor rend
  pl. $D = 3$, $M = 4 \Rightarrow \tau'/\tau = 128!!!$
- implicit $\rightarrow$ mátrixegyenlet
  (ritka mátrix, iteratív megoldás)
- rácsanizotrópia (operátor pontosság)

Spektrális

Előnyök:

- közel feltétel nélkül stabil sémák
- lokális függvénykiértékelés
  $\Rightarrow$ ideális párhuzamosíthatóság
- FFT: gyári könyvtárak elérhetőek
- pontos operátorok (FFT)

Hátrányok:

- Bonyolult "matek":
  (Operátor-szeletelési technikák)
- bonyolult stabilitás ellenőrzés
  (nemlineáris függvények spektrális térben $\rightarrow$ lim sup tételek)
- pontosság teszt tapasztalati úton
  (esetenkénti ellenőrzés)
- korlátozott peremfeltételek

Implementálás GPU-ra: általános jellemzők

Véges differencia (FD/FV)

- blokkonkénti rendezett adatbeolvasás a globális memóriából
- a peremelemek (*ghost*) beolvasása rosszul kondicionált
- szomszédos elemek használata $\rightarrow$ shared memory + szinkronizálás
- időléptetési séma (shared memory)
- blokkonkénti rendezett kiírás a globális memóriába

pros & cons

- 1 kernel, de esetleg hosszú, bonyolult
- rosszul kondícionált memóriakezelés
- kommunikáció a blokkok között

Spektrális

- gyári FFT könyvtár használata
- lokális kernelek valós és spektrális térben
- rendezett globális memóriaműveletek mindkét irányban (nincs *ghost*)
- láncok, pl $h(x,t) := \partial_x^2 f_{nl}[\phi(x,t)]$

$$[\phi_i^t] \xrightarrow{f_{nl}[\phi_i^t]} [f_i^t] \xrightarrow{F} [-k^2 f_k^t] \xrightarrow{F^{-1}} [h_i^t]$$

pros & cons

- sok, egyszerű kernel, moduláris
- jól kondícionált memóriakezelés
- FFT domináns teljesítmény

Az optimalizálásról általában

Programozási modell: Make it work. Make it right. Make it fast.

"Premature optimization is the root of all evil."

A párhuzamos optimalizálás területei

- Az implementált numerikus módszer teljesítményoptimalizálása
- Optimális numerikus módszer választása az adott problémához
- A numerikus módszer (nem az implementáció!) optimalizálása az architektúrára!
  $\Rightarrow$ Az architektúrák fejlődése visszahat a numerikus módszertan fejlődésére

Megéri?

- **A feladatmegoldás ideje (τ) = programfejlesztési idő (τ_p) + futtatási idő (τ_f)**
- Hibajavítási (**debug**) idő: $\tau_d^{\text{soros}} << \tau_d^{\text{párhuzamos}}$
- Murphy: $\tau_p \approx \tau_d \Rightarrow$ Előfordulhat, hogy $\tau^{\text{soros}} < \tau^{\text{párhuzamos}}$

Módszertani optimalizálás példa: elliptikus egyenletek FD/FV megoldása

Többszintű diszkretizálás

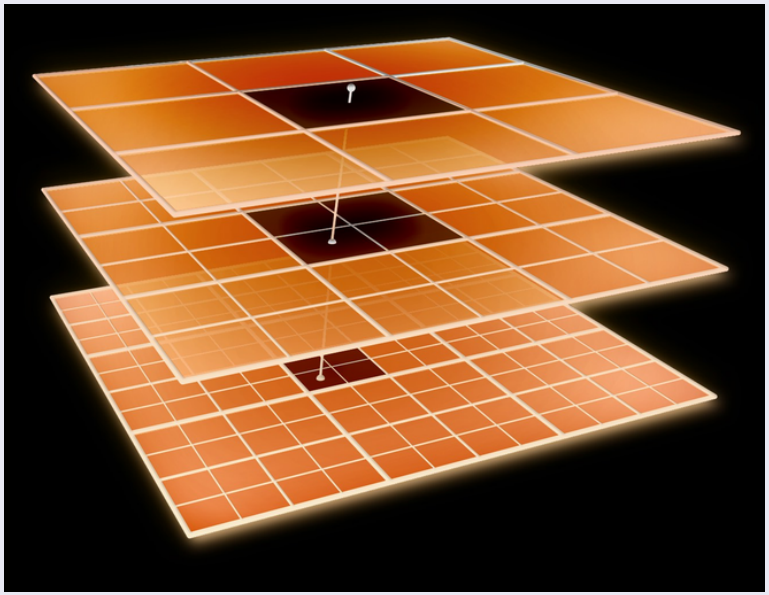


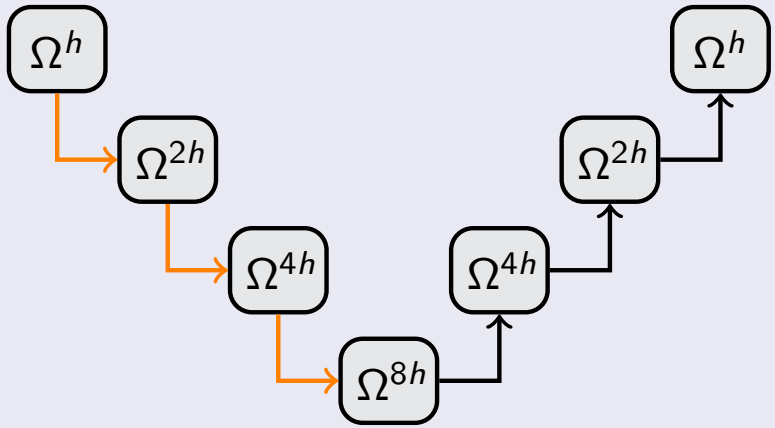
Fig. by Marius Sucan

- Azonos hibarelaxációs sebesség különböző hullámhosszokon: $h, 2h, 4h, 8h, \ldots$
- gyors konvergencia

$$\nabla^2 \phi(x, y) = \nabla \cdot \mathbf{v}(x, y)$$

megoldás Gauss-Seidel iterációval

Additive Correction Multigrid



1. simítás (hiba relaxálás)
2. hiba összegzés: $h \rightarrow 2h$
3. korrekció alkalmazása: $2h \rightarrow h$

Interpolated Stencil Multigrid (Tegze & Tóth, submitted to Comp. Phys. Comm.)

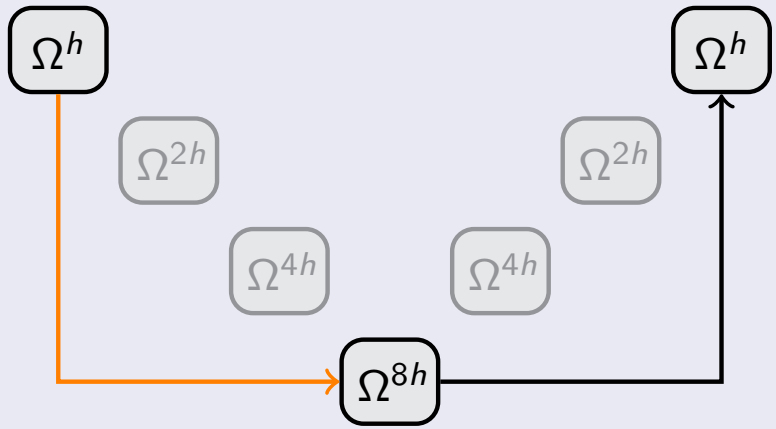
Problémák a GPU-n

- A 2x2-es összegzés és korrekció alkalmazás nem hatékony
- Csökkenő teljesítmény, amint $h \to 2h \to 4h \to 8h \to \ldots$

Lehetséges megoldás

- 8x8, 16x16, 32x32 jellemző csempeméretek (shared memory)
- 1 csempe / 1 redukciós kernel hatékonyabb
- 2 szint: h (GPU) $\Leftrightarrow$ (~16h) (CPU)
- additív korrekció $\to$ bilineáris interpoláció

Ritka multigrid ciklus



- Feltehetjük, hogy nem minden szint szükséges
- Egyszerre több szintet kihagyunk (átugrunk)
- Jellemző durvítás: lineáris csempeméret!

Interpolated Stencil Multigrid (Tegze & Tóth, submitted to Comp. Phys. Comm.)

Elvégzendő műveletek

- ACM: Gauss-Seidel iteráció + $2x2$ redukció-korrekció **szintenként**
- ISM: Gauss-Seidel + redukció (GPU) - Gauss-Seidel (CPU) - interpoláció (GPU)

Tapasztalat

- A szükséges GS itarációk száma (nyers számításigény): $N_{ACM} \approx N_{ISM}$
- Az ISM $5-6\times$ kevesebbet kommunikál ($\sim$ kernel hívások száma)

Következtetések

- Az új elméleti numerikus módszer kifejlesztését a GPU architektura motiválta
- Az ISM módszer elméleti teljesítménye közel azonos a hagyományos ACM-évelt
- Az ISM jelentősen gyorsabb a memória sávszélesség limitált architekturán!

SimGPU 2013, Freudenstadt, Deutschland

A workshop témája

- fizikai alkalmazások (PDE, rács QCD, . . .) muli-GPU implementációja
- skálázódás és optimalizálás nagy klasztereken (akár 10.000 GPU)

Tapasztalatok - NVIDIA

- GPUDirect+mpi technológia továbbra sem teljes:
  1. GPU global → page locked (CPU) → infini buffer (CPU) → infini card
  2. GPUDirect 1.0: GPU global → infini buffer → infini card
  3. GPUDirect 2.0: GPU global → infini card - CSAK VIRTUÁLISAN (UVA)
- Hiába a P2P, a QPI hídon keresztül nagyon lassan kommunikál

  ⇒ 1 CPU-hoz 1 GPU + 1 infiniband kártya azonos PCI express buszon

Nagy problémák skálázódása

Véges differencia (FD/FV)

Hagyományos szeletelés

N^3/n méretű szelet / node
n: node-ok száma (közöttük mpi)
+ node-on belül OpenMP párhuzamosítás

Transzfer igény

- x szelet: $2 \times N^2$ transzfer [per GPU]
- (x, y) hasáb: $4\frac{N^2}{\sqrt{n_{GPU}}}$ [per GPU]
- (x, y, z) kocka: $6\frac{N^2}{n_{GPU}^{2/3}}$ [per GPU]

$\Rightarrow$ $n_{GPU} > 256$ esetén a kocka optimális!
(a kis kommunikációigény legyőzi a tömbátrendezések idejét)

Spektrális (2D/3D FFT)

Módszer

1. $1D/2D$ batched FFT_{R2C}
2. blokk átrendezés (GPU-n)
3. blokk transzponálás (GPU-k között)
4. $1D$ batched & strided FFT_{C2C}

4 × Tesla C2050, OpenMP, 3D FFT