Erősen korrelált rendszerek szimulációja NVIDIA CUBLAS könyvtár segítségével: egyszerűen és hatékonyan

Nemes Csaba*, Barcza Gergely†, Nagy Zoltán*, Legeza Örs†

* Információs Technológiai Kar, Pázmány Péter Katolikus Egyetem, Budapest
† Erősen Korrelált Rendszerek "Lendület" Kutatócsoport, Wigner Fizikai Kutatóközpont Szilárdtest-fizikai és Optikai Intézet, Magyar Tudományos Akadémia, Budapest

nemes.csaba@itk.ppke.hu

Jun, 20, 2013

Sűrűségmátrix renormálásicsoport algoritmus (DMRG)

DMRG: Density Matrix Renormalization Group

A DMRG egy

- iteratív, variációs numerikus módszer
- az egydimenziós kölcsönható kvantum rendszerek alapállapotának meghatározására.

Az algoritmus előnye, hogy

- a lánchossz növelése ellenére a tér kezelhető számú bázissal írható le,
- melyek optimálisan választhatók meg
- az alrendszerek összefonódásának csökkentése mellett.

1992, Steven R. White

A DMRG alkalmazási területei

Erősen korrelált rendszerek szimulációja

- Anizotróp anyagok numerikus szimulációja (polimerek)
- Optikailag csapdázott ultrahideg atomok rendszere, kvantumszámítógép
- Kvantumkémiai rendszerek, molekulák elektronszerkezete

Spin-1/2 Heisenberg-lánc (elmélet)

- Anizotróp mágneses rendszerek leírására
- A rendszert rácspontokban rögzített, kölcsönható spinek modellezik
- Egy rácspont a $\{\uparrow, \downarrow\}$ független állapotok lineáris kombinációjaként írható le

Másodkvantált Hamilton:

$$H = \frac{1}{2} \sum_{j=1}^{N-1} \left(S_j^+ S_{j+1}^- + S_j^- S_{j+1}^+ \right) + \Delta \sum_{j=1}^{N-1} S_j^z S_{j+1}^z \tag{1}$$

- S_j^+, S_j^-, S_j^z a léptető és vetület operátorok a j rácspontban
- Δ az interakciós kölcsönhatás

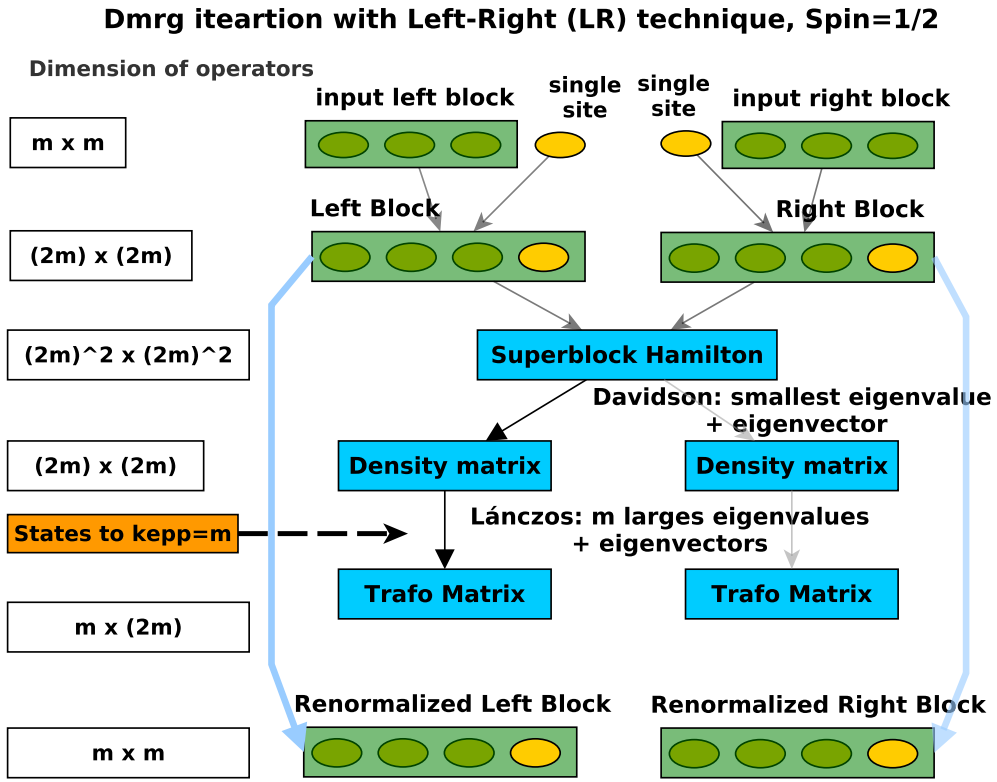
A j rácspontban ható $\mathcal{O}_j$ operátor az N részecskés bázisban:

$$\mathcal{O}_j = \bigotimes_{i=1}^{j-1} \mathbb{I} \otimes \mathcal{O} \otimes \bigotimes_{i=j+1}^{N} \mathbb{I} \tag{2}$$

ahol $\mathcal{O}$ egy rácspont bázisában felírt operátor, $\mathbb{I}$ az identitás operátor.

Számolás DMRG-vel (gyakorlat)

- Többféle stratégia lehetséges a Hamilton iteratív felépítésére. A cél az, hogy a Hamilton tényleges felépítését elkerüljük.
- A Hamilton csak a Davidson algoritmus projekciós lépésében jelenik meg. Ez a mátrix-vektor szorzás viszont 2 mátrix-mátrix szorzássá szelidíthető, és elvégezhető tényleges felépítés nélkül.



Számolás DMRG-vel (trükkök 1.)

- Az LR stratégia esetén a Hamiltonian előáll a bal és jobb oldalon értelmezett operátorokból az alábbi módon:

$$H_{SB} = \sum_{\alpha} A_{\alpha}^{(L)} \otimes B_{\alpha}^{(R)} \ , \tag{3}$$

  ahol $A^{(L)}$ mindig egy bal, $B^{(R)}$ mindig egy jobb oldali operátor, α pedig az operátor párok indexe.

- Davidson algoritmus projekciós lépése:

$$X' = H_{SB} X. \tag{4}$$

- Kihasználva a Kronecker szorzás tulajdonságait:

$$\tilde{X}' = \sum_{\alpha} A_{\alpha}^{(L)} \tilde{X} B_{\alpha}^{(R)T} \ . \tag{5}$$

Számolás DMRG-vel (trükkök 2.)

- Szimmetria operátorok figyelembevételével a Hilbert teret független alterekké (*szektorok*) bonthatjuk.
- A operátorokat praktikusan az alterek közötti transzformációkat leíró sűrű részmátrixokkal reprezentálhatjuk.
- Így az egyenlet (5) több, kisebb sűrű mátrixokon értelmezett, az alábbi alakú műveletté egyszerűsödik:

$$\tilde{X}'_{jl} \mathrel{+}= A^{(L)}_{i\to j} \tilde{X}_{ik} B^{(R)T}_{k\to l} \tag{6}$$

ahol $A^{(L)}_{i\to j}$ bal oldalon értelmezett operátor az i-ből a j szektorba képez.

Davidson iteráció

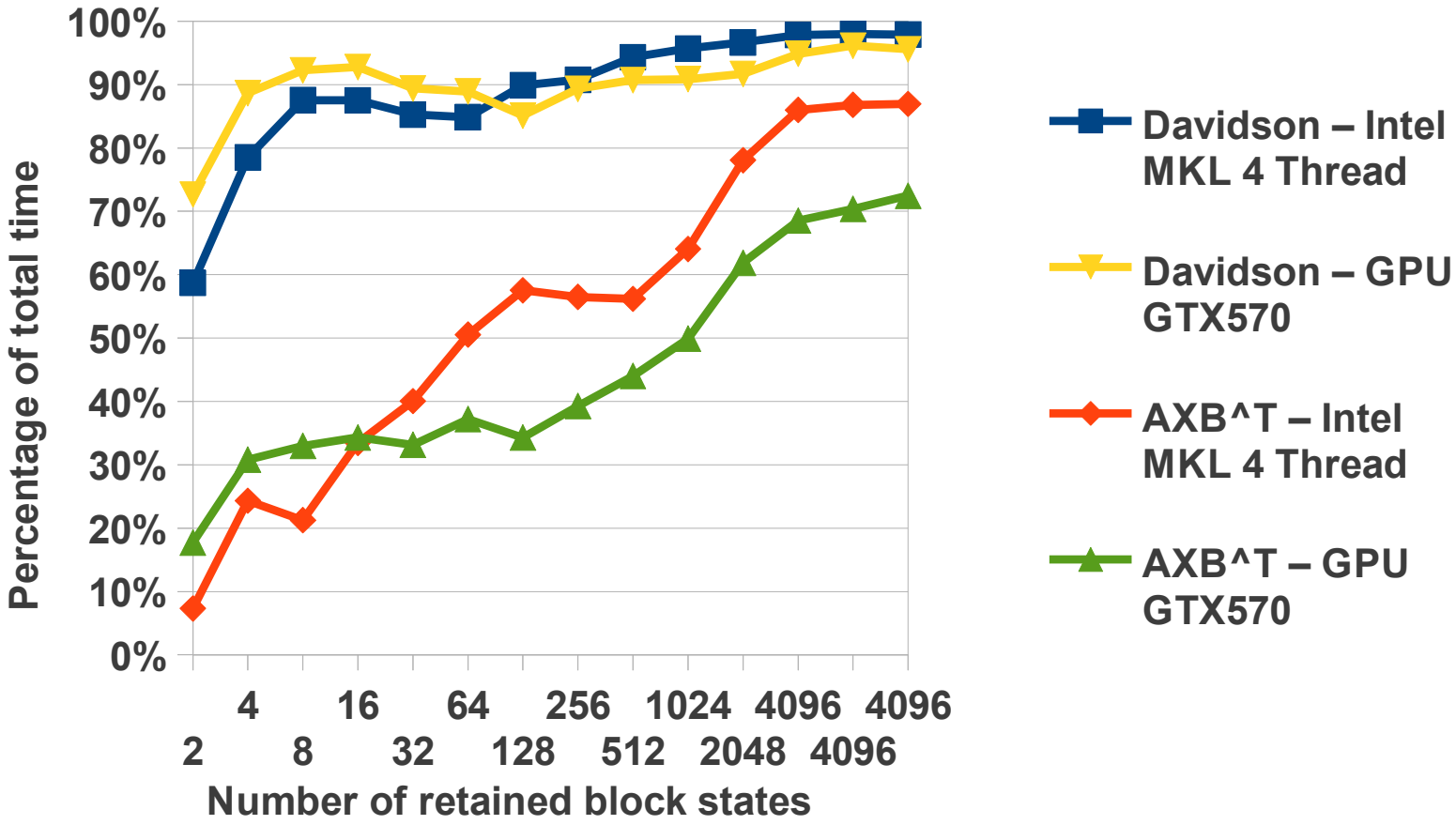
- V: basis vectors
- W: projected basis vectors
- B: matrix for $VH_{SB}V^T$
- y: Ritz vector
- λ: smallest eigenvalue of B
- x: estimate vector ($H_{SB}y$)
- r: residual vector
- i: iteration

Algorithm 1 One iteration of the Davidson algorithm

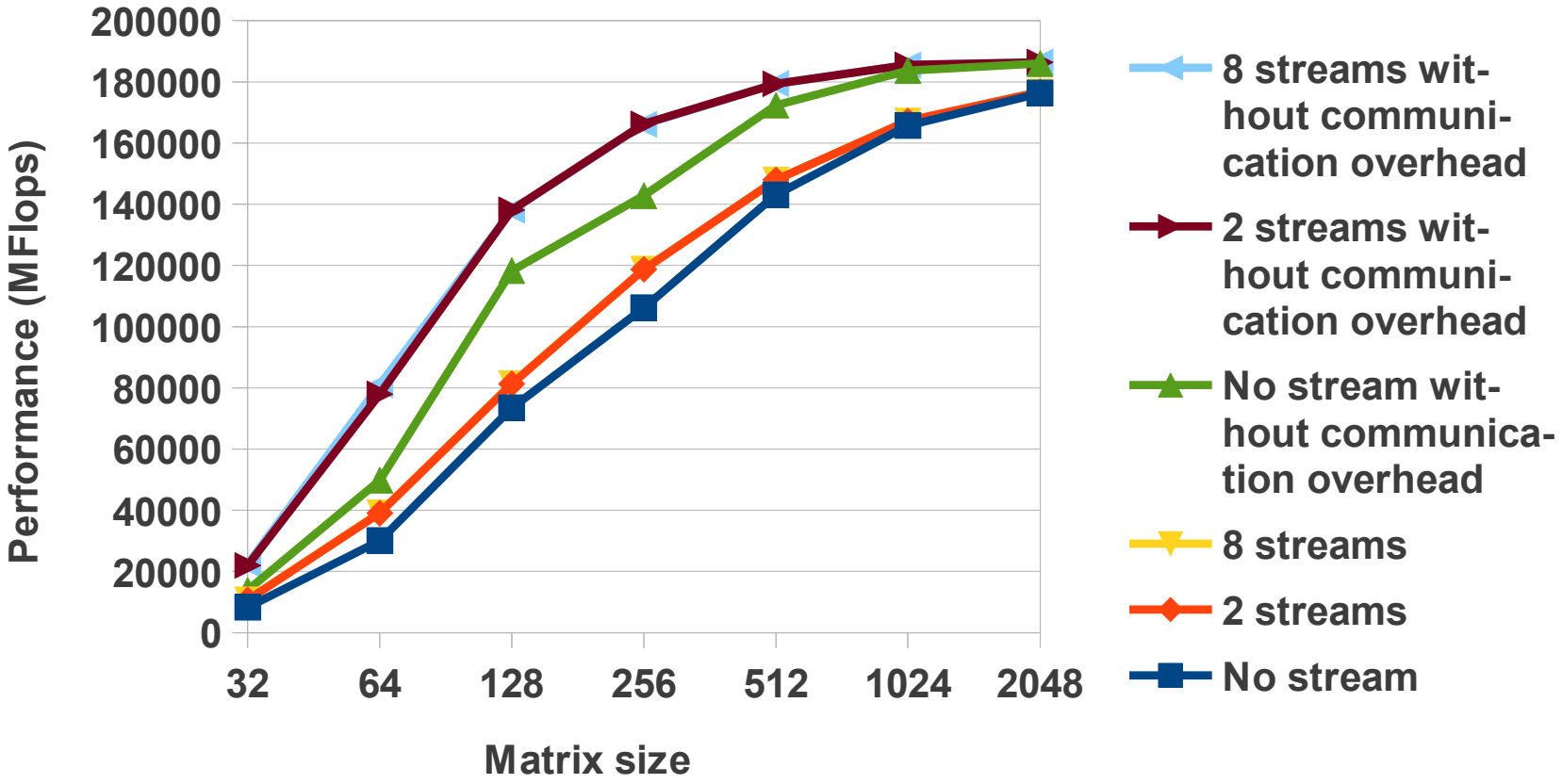
Precond: Previous $(i-1)$ basis vectors already computed.

1: **function** DAVIDSONITER(i)
2: $W(:,i) = H_{SB} \cdot V(:,i)$ ▷ BLAS-3: dgemm()
3: $B(:,i) = V^T \cdot W(:,i)$ ▷ BLAS-2: dgemv(trans)
4: $[\lambda, y] \longleftarrow$ get smallest eigvalue and vector of B
5: $x = V \cdot y$ ▷ BLAS-2: dgemv()
6: $r = -\lambda \cdot x + W \cdot y$
7: **if** norm(r) ≈ 0 **then**
8: return with x and success
9: **else**
10: orthonormalize r against V ▷ BLAS-2: dgemv(trans),dgemv()
11: append r to V
12: return without success

A DMRG futási idejének analízise



Az AXB^T gyorsíthatósága GPU-n. (GTX 570)



Az AXB^T gyorsítása

CUDA driveren keresztül elérhető speciális tulajdonságok:

- CUDA kernelek párhuzamosan futtathatók a GPU-n
- I/O műveletek átlapolhatók

Programozás szempontjából mindkettő a CUDA streamek segítségével érhető el. (CuBLAS függvénykönyvtár támogatja.)

Stream

Műveletek sorozata, melyek egymás után, az utasítás sorrendjében hajtódnak végre. Különböző stream-ekhez tartozó műveletek viszont határozatlan sorrendben, párhuzamosan futnak.

Megvalósítás:

- Az $(AX)B^T$ művelet esetén az A és B mátrixok a Davidson előtt a GPU memóriájába tölthetők, és az iterációk során nem változnak.
- A trükköket használva $(AX)B^T$ helyett sok $A^{(L)}_{i\to j}\tilde{X}_{ik}B^{(R)T}_{k\to l}$ művelet.
- $A^{(L)}_{i\to j}, B^{(R)T}_{k\to l}$ mátrix párok (*operáció*) *rekordokba* szedése
- Mivel a CPU teljesítménye nem elhanyagolható, a hibrid megoldás előnyös.
- Memóriaigény és a streamek száma közötti egyensúly

Az AXB^T gyorsítása II.

Algorithm 2 Az operáció rekordok kezelése

1: Az operáció rekordok elkészítése, az elvégzendő munka (FLOP) kimenetenként történő kiértékelése
2: Az operáció rekordok **particionálása**: a legtöbb munkát igénylő kimenetekhez tartozó rekordok kerülnek a GPU-hoz, egyetlen megosztott kimenet lehetséges, amely esetben az eredmények a végén összefésülődnek.
3: Az operáció rekordok ütemezési stratégiájának a kiválasztása
4: A kiválasztott stratégia alapján a GPU-nak szánt munka összeállítása

Az AXB^T gyorsítása - Implementált stratégiák

"small" stratégia

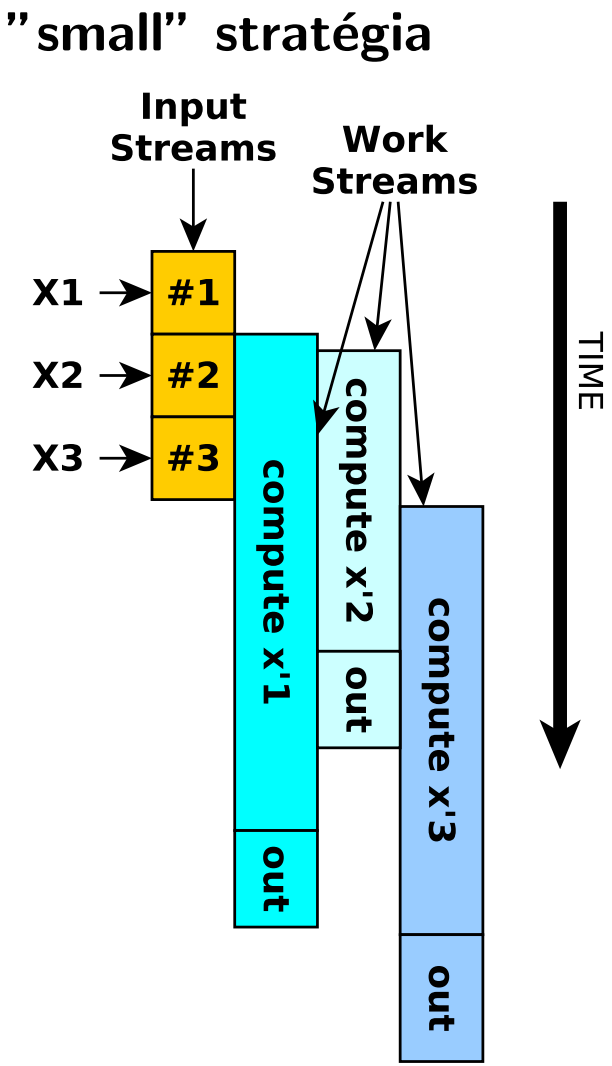
- Egy I/O stream. Ahány bemenet, annyi buffer. Ahány kimenet, annyi munka stream.
- Ha a munka stream végzett, elkezdi visszamásolni az eredményt.
- Nagyobb memóriaigény. (Kis méretnél nem számít.)
- Kisebb mátrixok esetén a több stream, párhuzamos munkavégzés, előny.

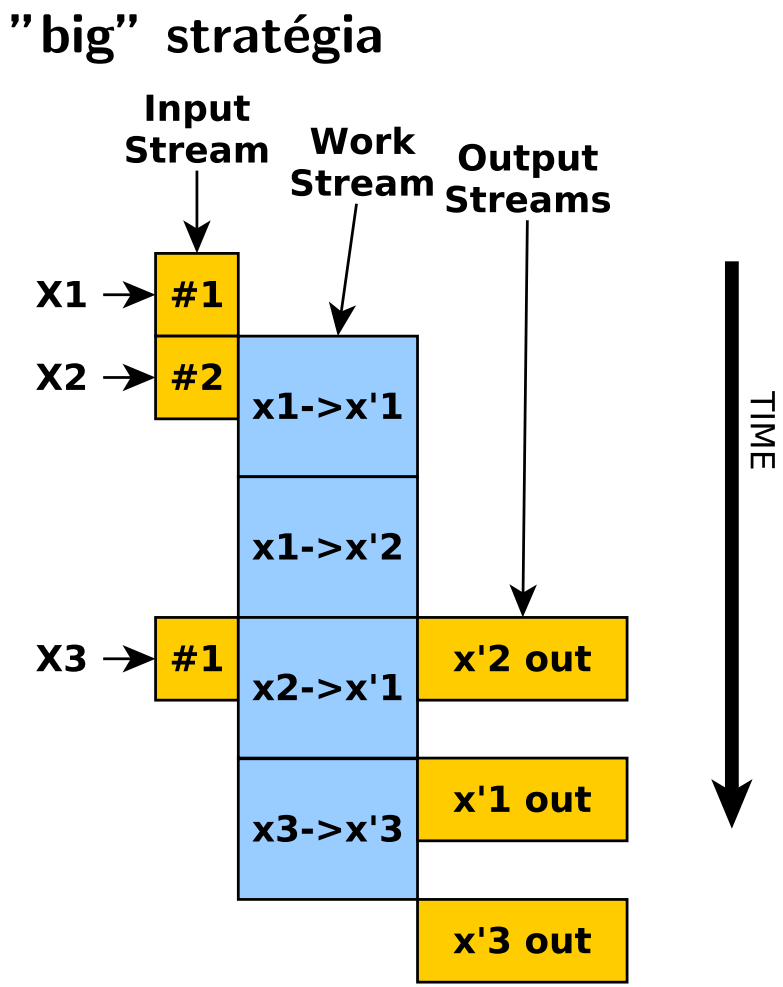
Kérdés, hogy mi számít nagy méretnek!

"big" stratégia

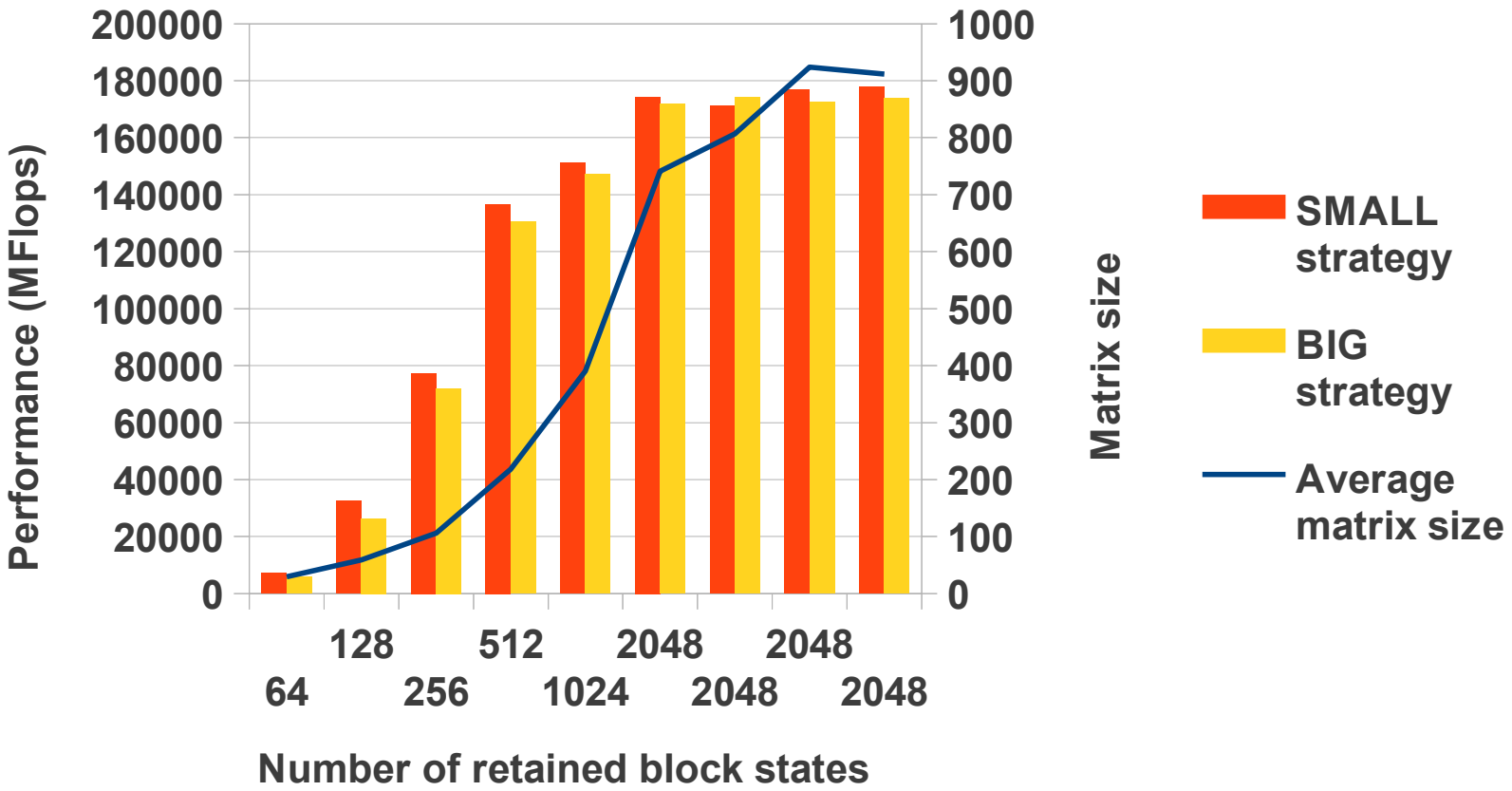
- Egy I/O stream. Két bemeneti buffer. Egy munka stream. (Dupla bufferelés.)
- A rekordok a bemenetek alapján sorbarendezve.
- Kevesebb memóriaigény. (Nagy méretnél számít.)
- Kisebb mátrixok esetén a GPU nincs kihasználva, lassabb.

Az AXB^T gyorsítása - Implementált stratégiák

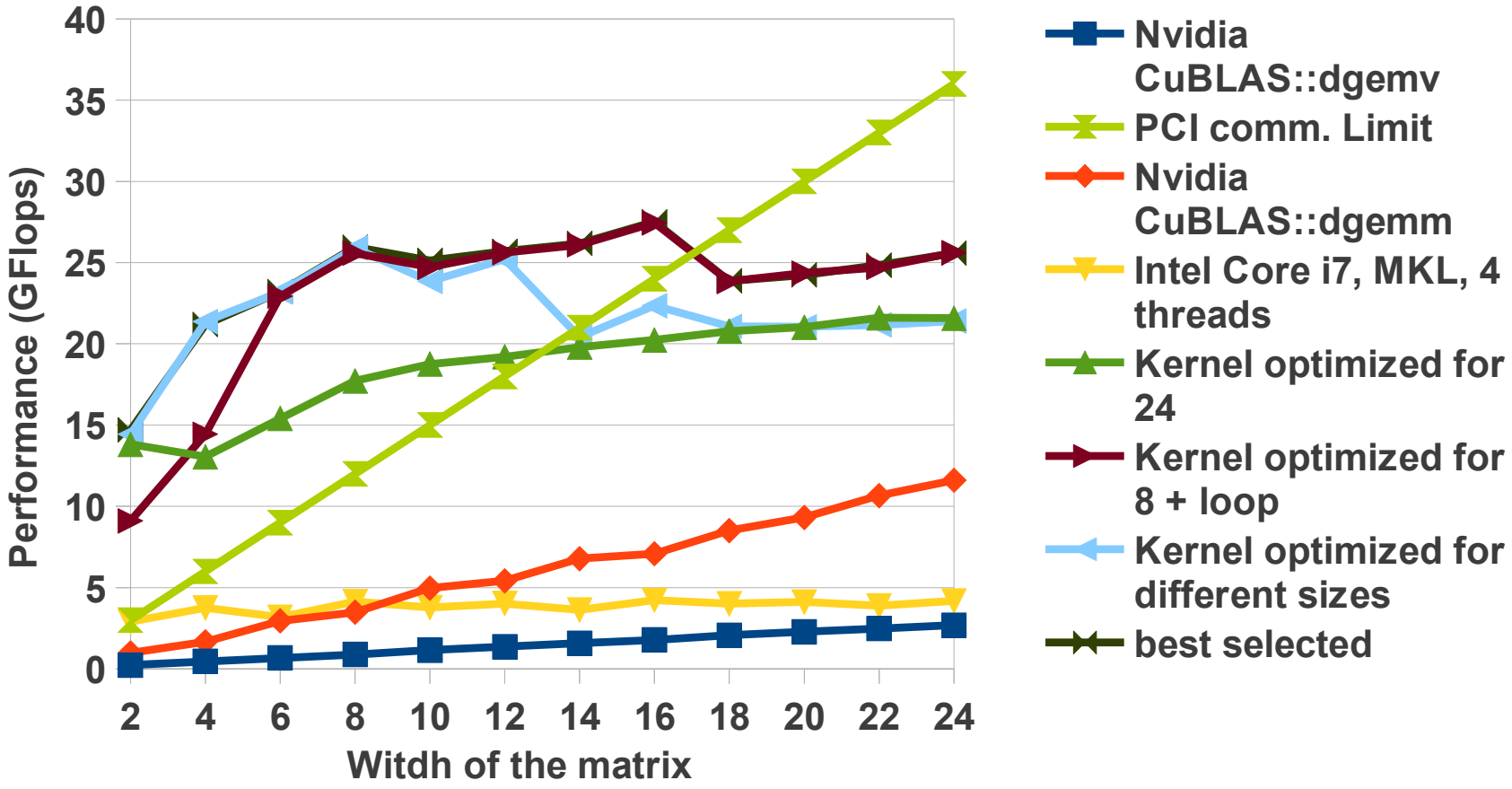




Az AXB^T gyorsítása - eredmények



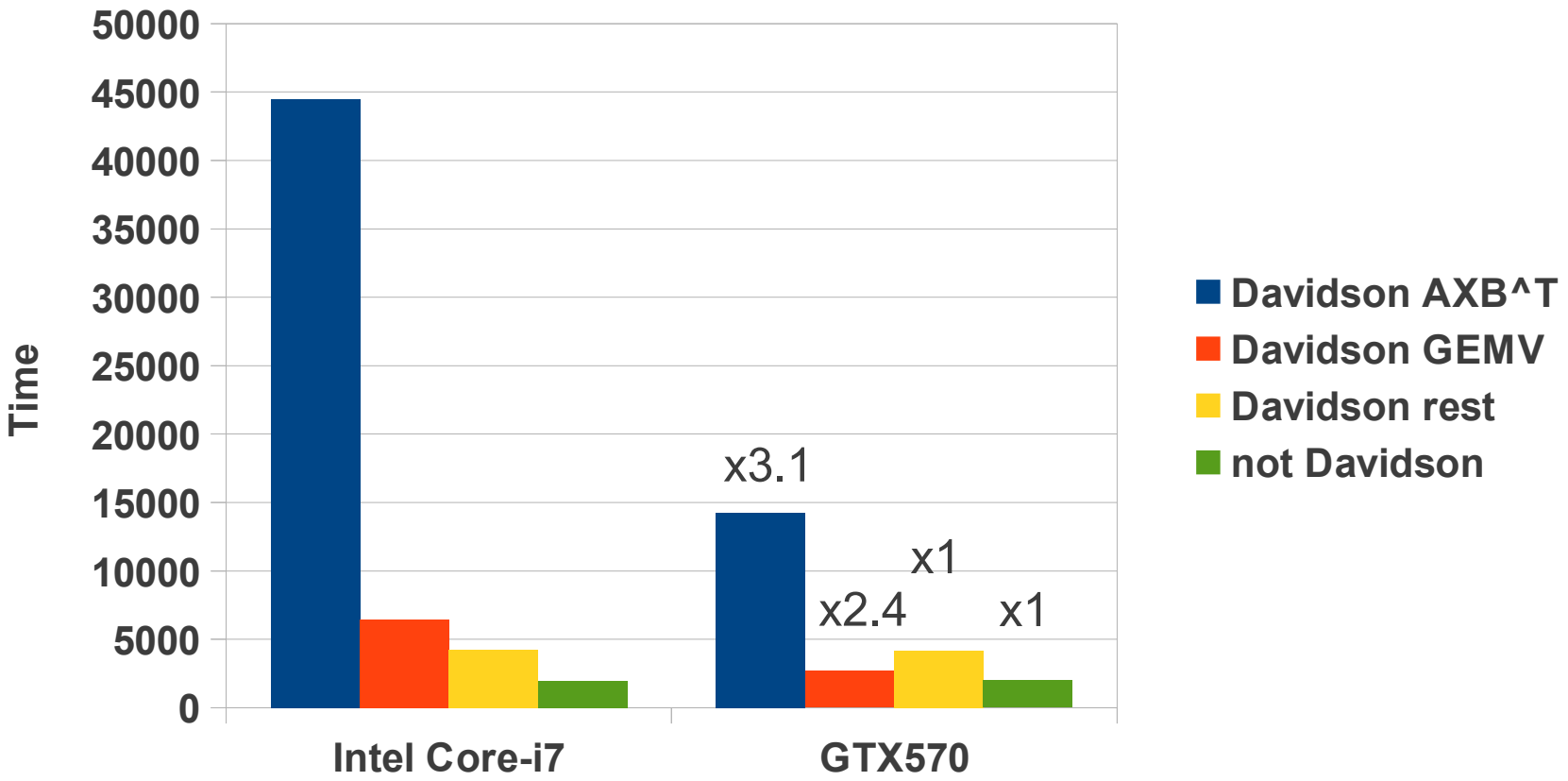
Az gemv(transzponált) gyorsítása - tesztek



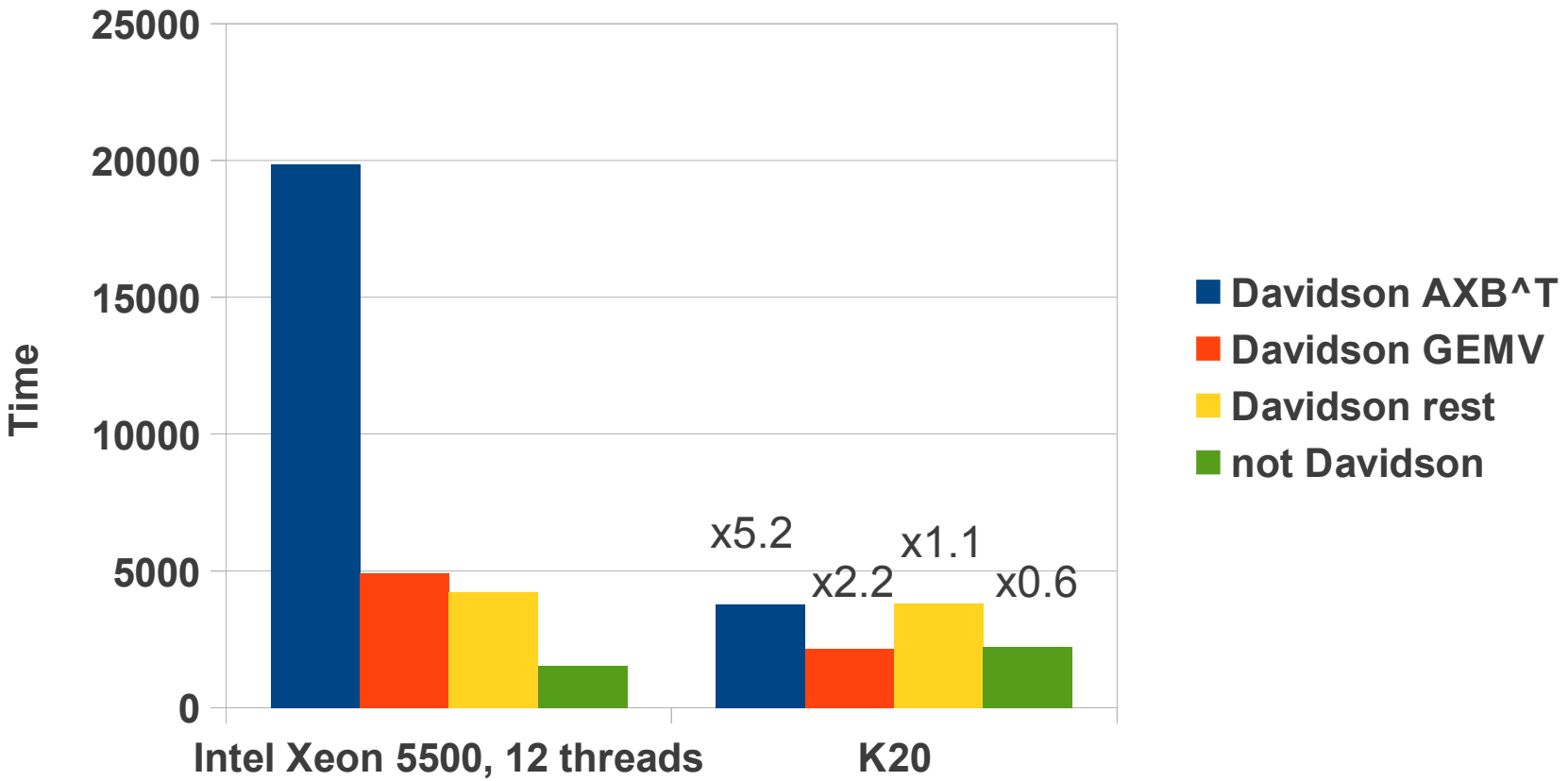
Az gemv(transzponált) gyorsítása - kernel részlet

```
#define GEMV_GRID_WIDTH 128
#define GEMV_TB_WIDTH 256
#define GEMV_FRACTION 8

__global__ void myGemvKernelTransponated(...){
    __shared__ double sharElem[GEMV_FRACTION*GEMV_TB_WIDTH];

    for(int iBelt_c = 0;iBelt_c<iBelt_o;iBelt_c+=GEMV_FRACTION) {
        int iBelt   = min(GEMV_FRACTION,iBelt_o-iBelt_c);
        int iOffGlobal   = iBelt_c * iLda;

        #pragma unroll 8
        for(int j=0; j < iBelt;++j) {
            sharElem[ j*GEMV_TB_WIDTH + threadIdx.x] = 0;
        }

        int iFull = blockDim.x * gridDim.x;
        for(int i = (blockIdx.x*blockDim.x) + threadIdx.x; i < len;i+=iFull) {
            double privVE = dVec[i];

            #pragma unroll 8
            for(int j = 0; j < iBelt;++j) {
                sharElem[ j*GEMV_TB_WIDTH + threadIdx.x]
                                += dMtx[i + iOffGlobal + j*iLda] * privVE;
            }
        }
        ...
    }
}
```

Eredmények - GTX570



Eredmények - K20



Eredmények - Összegzés