Využití GPGPU pro řešení soustav lineárních rovnic

Vítězslav Žabka

Katedra matematiky, Fakulta jaderná a fyzikálně inženýrská,
České vysoké učení technické v Praze

Diplomová práce
2009/2010

Co to je?

Provádění obecných výpočtů na grafických kartách PC

Co to je?

Provádění obecných výpočtů na grafických kartách PC

Proč grafické karty?

- Obrovský hrubý výpočetní výkon, velká propustnost paměti
- Potenciál pro urychlení numerických výpočtů
- Výhodný poměr cena/výkon
- Vysoce paralelní architektura, desítky až stovky jader
- Obtížnější využít dostupné prostředky
- Vhodné zejména pro řešení výpočetně náročných datově paralelních úloh

GPGPU technologie NVIDIA CUDA

- Představena na konci roku 2006, nyní široce rozšířená, intenzívně se vyvíjí
- Aplikace ve zpracování obrazu, videa a zvuku, fyzikálních simulacích, kryptografii...
- Grafický procesor (GPU) — vysoce paralelní koprocesor k CPU
- Programování GPU bez nutnosti práce s grafikou

Nástrahy programování v CUDA

- GPU zpracovává velké množství vláken paralelně
- Vlákna komunikují skrz sdílenou paměť — rychlá, ale její velikost omezená
- Globální paměť grafické karty velká, ale pomalá
- Nutnost optimalizovat práci s pamětí

Metoda GMRES

- Iterační metoda řešení soustavy lineárních algebraických rovnic

$$Ax = b, \quad A \in \mathbb{R}^{n \times n}$$

- CUDA implementace pro řídkou matici A více než 10krát rychlejší než jednovláknová CPU implementace

Metoda GMRES

- Iterační metoda řešení soustavy lineárních algebraických rovnic

$$Ax = b, \quad A \in \mathbb{R}^{n \times n}$$

- CUDA implementace pro řídkou matici A více než 10krát rychlejší než jednovláknová CPU implementace
- Možnost předpodmínění
  - zleva:

$$M^{-1}Ax = M^{-1}b$$

  - zprava:

$$AM^{-1}u = b, \quad u = Mx$$

- Cílem práce implementovat předpodmínění metody GMRES v CUDA

Paralelizovatelné techniky předpodmínění

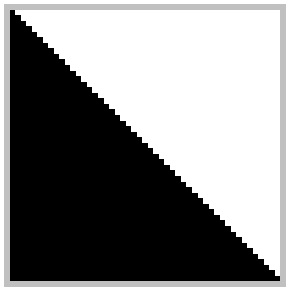
$$Ax = b \quad \longrightarrow \quad M^{-1}Ax = M^{-1}b$$

- Jacobi — jednoduše implementovatelné, levné, ale málo účinné
- Diskrétní kosinová transformace (Mistry, Braganza, Kaeli, Leeser)
- Čebyševovo polynomiální (Asgari, Tate)
- Polynomiální blokově diagonální ILU (Wang, Klie, Parashar, Sudan)
- Blokově diagonální ILU

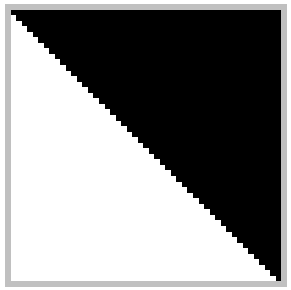
Obecné ILU předpodmínění

- $A = LU - R$, $M = LU$ kde L je dolní a U horní trojúhelníková matice
- V každé iteraci metody je třeba řešit soustavy

$$u = M^{-1}v = U^{-1}\left(L^{-1}v\right) \quad \longrightarrow \quad Ly = v,\ Uu = y$$



L



U

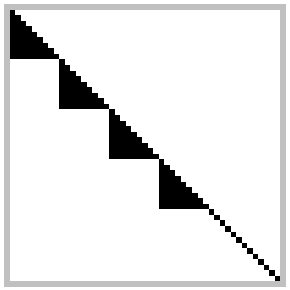
Ryze sekvenční proces

Blokově diagonální ILU předpodmínění

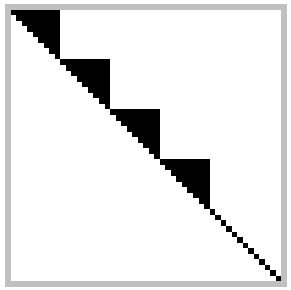
- Chceme paralelizovat řešení soustav

$$Ly = v,\ Uu = y$$

- Blokově diagonální tvar matic soustavy:



L



U

- Umožňuje paralelizaci algoritmu
- Ale omezení na strukturu matic L a U
- Očekávána nižší účinnost předpodmínění

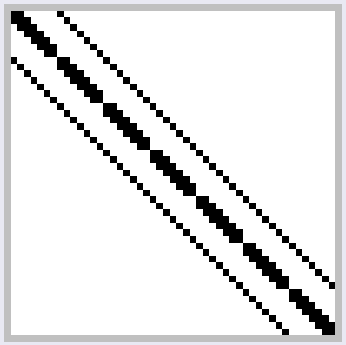
Příklad — rovnice vedení tepla ve 2D

$$\frac{\partial u}{\partial t}(x,t) = \Delta u(x,t)$$

Očíslování uzlů sítě po řadách

0	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	32	33	34
35	36	37	38	39	40	41
42	43	44	45	46	47	48

$\longrightarrow$



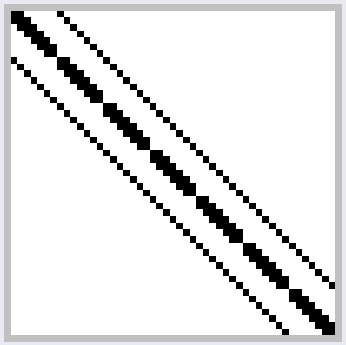
Příklad — rovnice vedení tepla ve 2D

$$\frac{\partial u}{\partial t}(x,t) = \Delta u(x,t)$$

Očíslování uzlů sítě po řadách

0	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	32	33	34
35	36	37	38	39	40	41
42	43	44	45	46	47	48

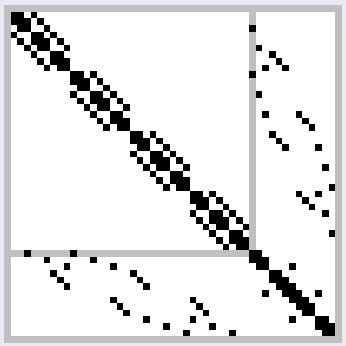
$\longrightarrow$



Blokové očíslování uzlů sítě

0	1	2	36	9	10	11
3	4	5	37	12	13	14
6	7	8	38	15	16	17
39	40	41	42	43	44	45
18	19	20	46	27	28	29
21	22	23	47	30	31	32
24	25	26	48	33	34	35

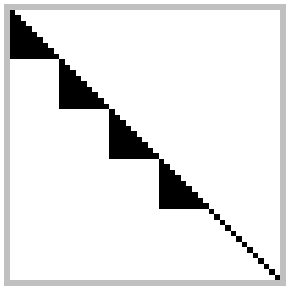
$\longrightarrow$



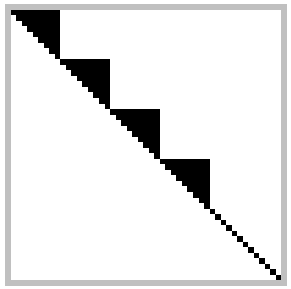
Návrh implementace v CUDA

Paralelizace řešení soustav

$$Ly = v,\ Uu = y$$



L



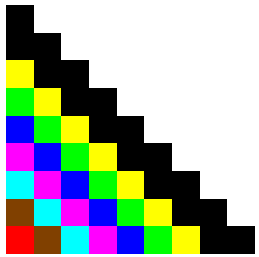
U

Sloupcový způsob

- Jeden blok matice zpracovává několik CUDA vláken
- Nutnost spolupráce vláken
- Vhodné pro větší a více zaplněné bloky matice

Implementace sloupcového algoritmu řešení $Lx = b$

- Vektor b zkopírovaný do x
- Prvky bloků matice se čtou po sloupcích
- Postup výpočtu pro j-tý sloupec:
  1. Vydělit x_j odpovídajícím diagonálním prvkem matice
  2. Pro každý poddiagonální prvek $l_{i,j}$ provést $x_i \leftarrow x_i - l_{i,j}x_j$
- Poddiagonální prvky lze zpracovávat paralelně, každý jedním vláknem:

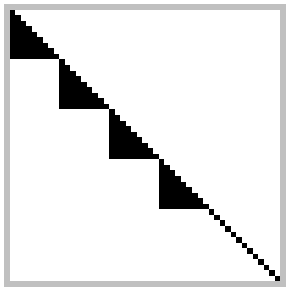


- Matice v paměti uložena ve speciálním formátu kvůli optimalizaci
- Pro složky vektoru x a diagonálu matice použita sdílená paměť GPU jako cache

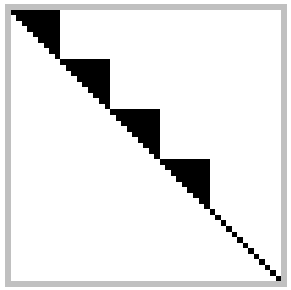
Návrh implementace v CUDA

Paralelizace řešení soustav

$$Ly = v,\ Uu = y$$



L



U

Řádkový způsob

- Jeden blok matice zpracovává jedno CUDA vlákno
- Absolutní nezávislost vláken
- Výhodné při velkém počtu bloků

Implementace řádkového algoritmu řešení $Lx = b$

- Vektor b zkopírovaný do x
- Prvky bloků matice se čtou po řádcích
- Postup výpočtu pro i-tý řádek:
  1. Pro každý předdiagonální prvek $l_{i,j}$ provést $x_i \leftarrow x_i - l_{i,j}x_j$
  2. Vydělit x_i odpovídajícím diagonálním prvkem matice
- Matice v paměti uložena ve speciálním formátu kvůli optimalizaci
- Pro složky vektoru x použita sdílená paměť GPU jako cache

Řešení soustav $Ux = b$

- Řešeno převodem na soustavu s dolní trojúhelníkovou maticí soustavy

$$\begin{pmatrix} u_{1,1} & u_{1,2} & u_{1,3} \\ & u_{2,2} & u_{2,3} \\ & & u_{3,3} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$\begin{pmatrix} u_{3,3} & & \\ u_{2,3} & u_{2,2} & \\ u_{1,3} & u_{1,2} & u_{1,1} \end{pmatrix} \begin{pmatrix} x_3 \\ x_2 \\ x_1 \end{pmatrix} = \begin{pmatrix} b_3 \\ b_2 \\ b_1 \end{pmatrix}$$

- Převod matice lze provést při ukládání do formátu vhodného pro postupné dosazování na GPU
- Vektor x nutné transformovat vždy před řešením soustavy $Ux = b$ a pak zpět

Účinnost blokově diagonálního předpodmínění

- Dva testovací příklady — rovnice vedení tepla a Allenova-Cahnova rovnice
- Velikosti sítí — 256×256, 512×512, 1024×1024 uzlů
- Různé velikosti bloků a zaplnění při ILU rozkladu

Účinnost blokově diagonálního předpodmínění

- Dva testovací příklady — rovnice vedení tepla a Allenova-Cahnova rovnice
- Velikosti sítí — 256×256, 512×512, 1024×1024 uzlů
- Různé velikosti bloků a zaplnění při ILU rozkladu

Výsledky

- Jacobiho předpodmínění neúčinné
- Nejvýhodnější zaplnění — 4 mimodiagonální nenulové prvky
- Ušetřeno 6–54 % iterací metody GMRES
- Účinnost výrazně horší než neblokové ILU předpodmínění

Rychlost řešení soustav $Lx = b$

Síť	Bloky	CPU [ms]	GPU-S [ms]	GPU-R [ms]	Urychlení
256×256	8×8	3,6	0,59	0,35	10,28×
	12×12	3,7	0,69	0,63	5,87×
	16×16	3,9	0,87	1,05	3,71×
	32×32	3,6	1,05	-	-
512×512	8×8	15,0	2,19	1,08	13,88×
	12×12	15,4	2,43	1,43	10,76×
	16×16	15,9	2,64	1,31	12,13×
	32×32	15,8	3,35	-	-
	64×64	14,6	4,13	-	-
1024×1024	8×8	59,4	8,35	3,78	15,71×
	12×12	61,8	9,31	4,07	15,18×
	16×16	63,7	9,71	4,47	14,25×
	32×32	66,3	10,61	-	-
	64×64	63,8	13,18	-	-

- Řádková implementace nezvládá bloky větší než 16×16 uzlů (omezení velikostí sdílené paměti GPU)
- Sloupcová implementace pomalejší
- Urychlení řádkové implementace oproti CPU 3–16násobné
- Transformace vektoru x při řešení soustavy $Ux = b$ trvala na síti 256×256 0,08 ms, 512×512 0,25 ms, 1024×1024 0,92 ms

Rychlost předpodmíněné metody GMRES na GPU

- Rovnice vedení tepla, Allenova-Cahnova rovnice
- Sítě o velikostech 256×256 a 512×512 uzlů
- Všechny implementace metody GMRES v CUDA uspěly při řešení soustav lineárních rovnic
- Zpomalení oproti nepředpodmíněné metodě při každé volbě parametrů, někdy až trojnásobné

Závěr

- CUDA implementace blokově diagonálního předpodmínění metody GMRES nepřinesly urychlení
- Horší účinnost předpodmínění
- Nedostatek sdílené paměti GPU
- Nízká aritmetická intenzita algoritmu
- Rezervy ve vytíženosti GPU
- Část problémů by mohla zmírnit nová generace GPU