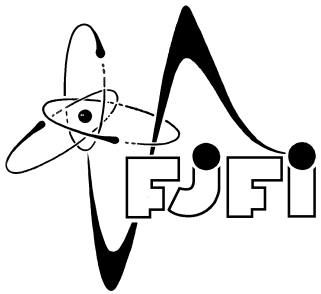
GPGPU

Provádění vědeckých výpočtů prostřednictvím grafických karet osobních počítačů

Jan Vacata



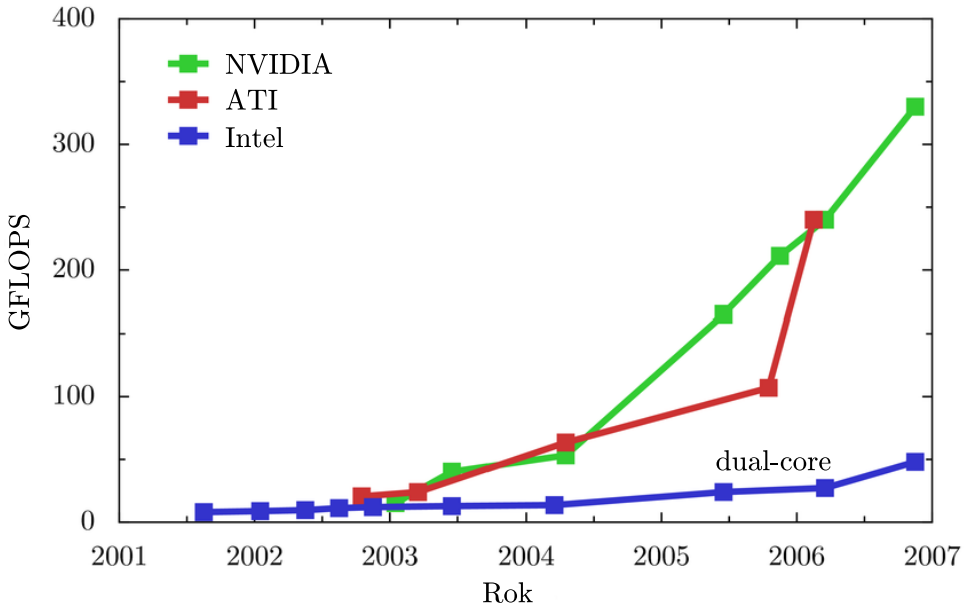
Motivace

3.0 GHz Intel Core2 Quad (QX6850)

- Výkon: 96 GFLOPS
- Propustnost paměti: 21 GB/s
- Orientační cena: 1100 USD

NVIDIA GeForce 8800 GTX

- Výkon: 330 GFLOPS
- Propustnost paměti: 55.2 GB/s
- Orientační cena: 550 USD



Obrázek: Sledování změn výkonu

Motivace - pokračování

Srovnání poměrného ročního nárůstu výkonu:

- Pro CPU: přibližně 1.4
- Pro GPU: přibližně 1.7 ve zpracování fragmentů(pixelů), asi 2.3 ve zpracování vrcholů

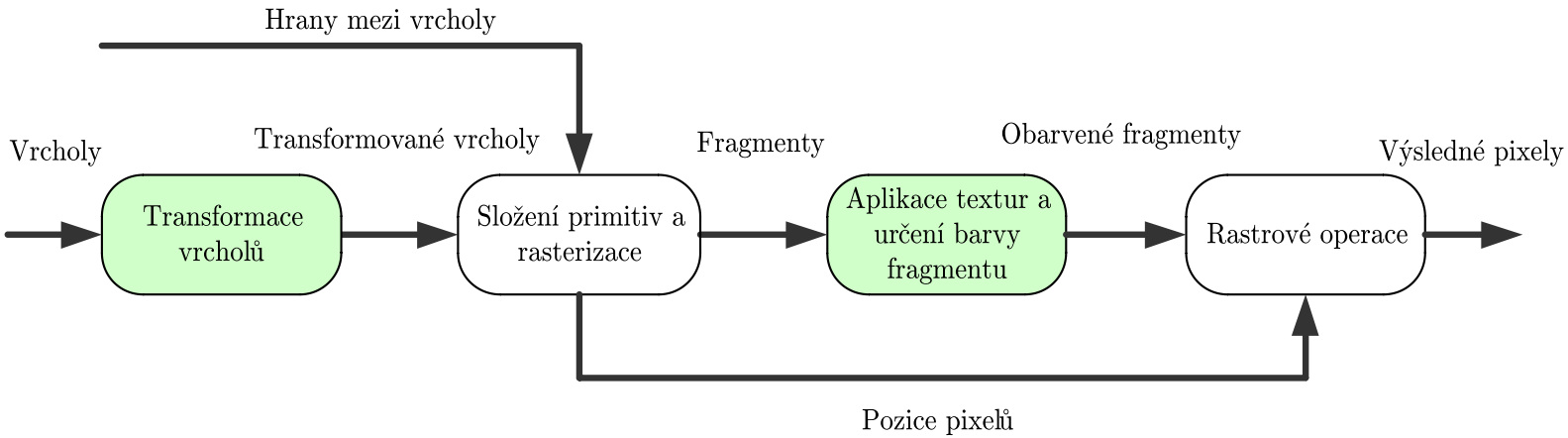
Proč jsou GPU tak rychlé?

- Technologický důvod: Specializace GPU pro rychlé provádění nezávislých paralelních výpočtů
- Ekonomický důvod: Obrovský trh s počítačovými hrami

Základní otázky

1. Bylo by možné využít ohromující výkon GPU pro výpočty, které nemají s grafikou vůbec nic společného?
2. Pokud ano, jak to lze provést?

Grafická pipeline, architektura GPU



Obrázek: Abstraktní pohled na grafickou pipeline

- 5 generací grafických karet (pevná vs. programovatelná pipeline)
- Programovatelné procesory: vertex procesor, fragment procesor (*fp*32 aritmetika, SIMD/MIMD charakter, *scatter*/*gather*, ...)
- Vertex/fragment programy/shadery

Programovací model I

Návrh a implementaci GPGPU[1] aplikace shrneme do několika bodů, pokusíme se vysvětlit na jednoduchém příkladu (násobení všech elementů rozsáhlého pole známou konstantou):

1. Identifikace paralelizovatelných částí problému, jejich implementace prostřednictvím uživatelských fragment programů
2. Nahrání vstupních dat do textur
   - Analogie pole $\longleftrightarrow$ textura

[1] z angl. General-Purpose computation on Graphics Processing Units

Programovací model II

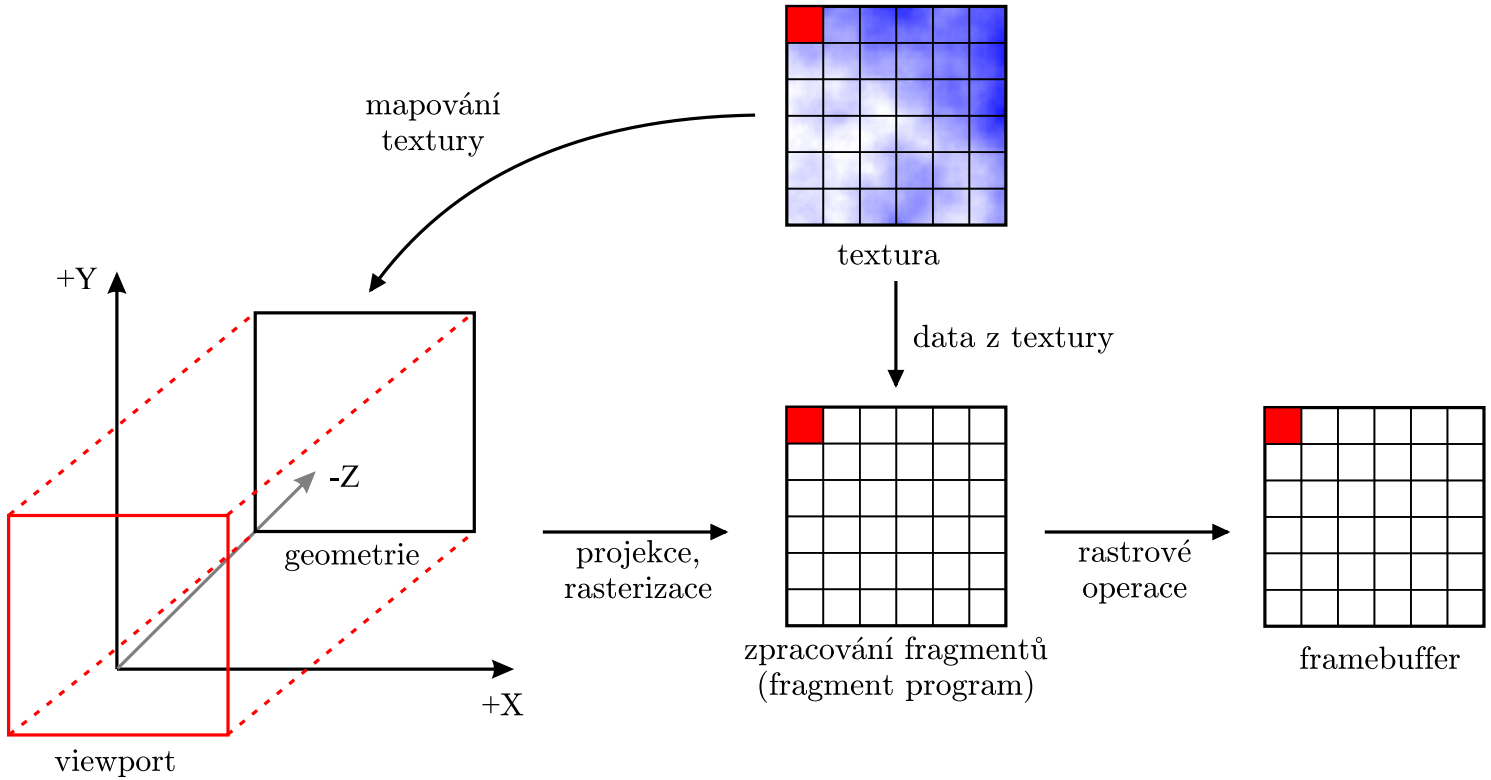
3. Renderování vhodné geometrie zajistí aplikaci požadovaného programu na fragmenty
   - Analogie tělo smyčky $\longleftrightarrow$ fragment programy

tělo smyčky na CPU

```
for (i=0;i<height;i++) {
  for (j=0;j<width;j++) {
    grid[i][j] *= k;
  }
}
```

Programovací model II

3. Renderování vhodné geometrie zajistí aplikaci požadovaného programu na fragmenty
   - Analogie tělo smyčky $\longleftrightarrow$ fragment programy



Programovací model II

3. Renderování vhodné geometrie zajistí aplikaci požadovaného programu na fragmenty
   - Analogie tělo smyčky $\longleftrightarrow$ fragment programy

tělo smyčky na CPU

```
for (i=0;i<height;i++) {
  for (j=0;j<width;j++) {
    grid[i][j] *= k;
  }
}
```

fragment program na GPU

```
struct SInput {
  uniform samplerRECT tex;
  float2 cr : TEXCOORD0;
};

float main(SInput in) : COLOR {
  flout out;
  out = texRECT(in.tex,in.cr)*k;
  return out;
}
```

Programovací model III

4. Rasterizér rozdělí vstupní geometrii na fragmenty, hodnoty ve vrcholech jsou interpolovány
5. Každý fragment potom nezávisle zpracován fragment programem
6. Výsledek ve framebufferu, následuje zpracování výsledku nebo další renderování
   - Analogie zpětná vazba $\longleftrightarrow$ renderování do textury

Další informace o programování GPU

- Které problémy lze efektivně mapovat na GPU?
  - Aritmetická intenzita
  - Sekvenční přístup do paměti
- Omezení vyplývající z architektury (mnohá odpadají u nejnovější generace GPU)
  - Scattering/gathering
  - Řízení toku programu
  - Celočíselná aritmetika
- Jazyky vyšší úrovně pro programování shaderů (Cg, HLSL, GLSL)

Postupné cíle této práce

1. Pochopit architekturu GPU a způsob jejího využití pro obecné výpočty
2. Nastudovat nutné principy OpenGL API
3. Pochopit vlastní GPGPU implementaci a ukázat ji na jednoduchém případě
   - Ukázkové GPGPU řešení operace konvoluce
4. Studovat a naučit se používat tuto techniku na složitější aplikace, hledat další možnosti využití
5. Provést úvodní průzkum v oblasti nejnovějšího HW a moderního GPGPU pojetí

Rovnice vedení tepla

Formulace problému

Hledáme řešení rovnice

$$\frac{\partial u}{\partial t} = k\Delta u \qquad \text{na} \qquad \Omega \times (0, T)$$

za podmínek

$$u|_{\partial\Omega} = \gamma,\ \gamma : \partial\Omega \longrightarrow \mathbb{R}, u(\cdot, 0) = u_0$$

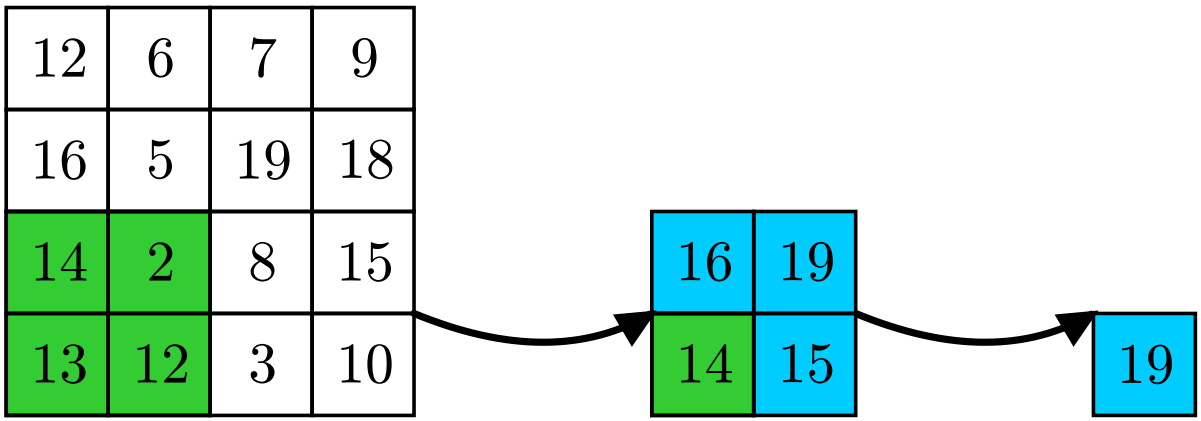
Řešení metodou přímek:

- Metoda konečných diferencí pro prostorovou diskretizaci
- Runge-Kuttova-Mersonova metoda 4. řádu s automatickou volbou integračního kroku pro diskretizaci časovou

GPGPU implementace

Hlavní znaky GPGPU implementace:

- Mapování na GPU velmi přirozené
  - Diskrétní 2D mřížka
  - Sekvenční přístup k datům
  - Laplaceův operátor a pětibodové schéma (příznivé vzhledem k 2D organizaci cache paměti na GPU)
  - Malá aritmetická intenzita
- Ping-pong technika
- Operace redukce (maximum z absolutních hodnot)



LU rozklad husté matice

Formulace problému

Hledáme rozklad

$$\mathbf{A} = \mathbf{LU},$$

kde **L** je dolní trojúhelníková matice s jedničkami na diagonále a **U** je horní trojúhelníková matice.

Základní verze algoritmu bez pivotingu

```
for k = 1,...,n-1 {
    for i = k+1,...,n
        a(i,k) = a(i,k) / a(k,k);
    for i = k+1,...,n
        for j = k+1,...,n
            a(i,j) = a(i,j) - a(i,k)*a(k,j);
}
```

GPGPU implementace

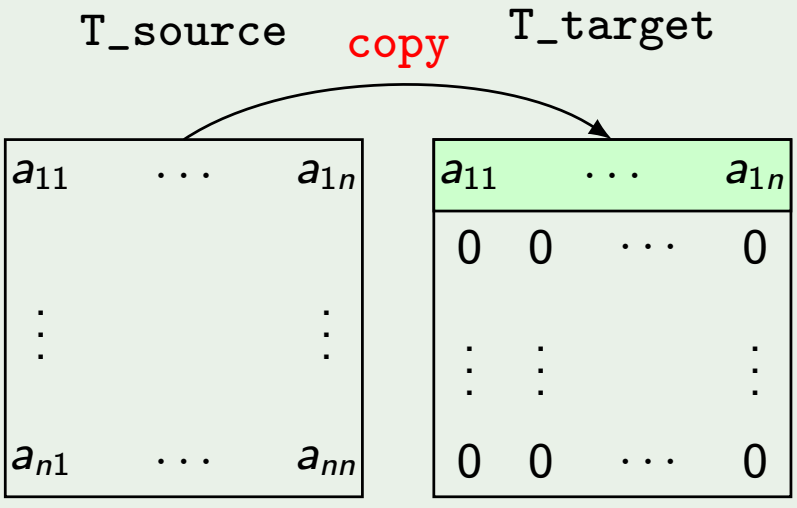
První krok LU rozkladu

`T_source`

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$$

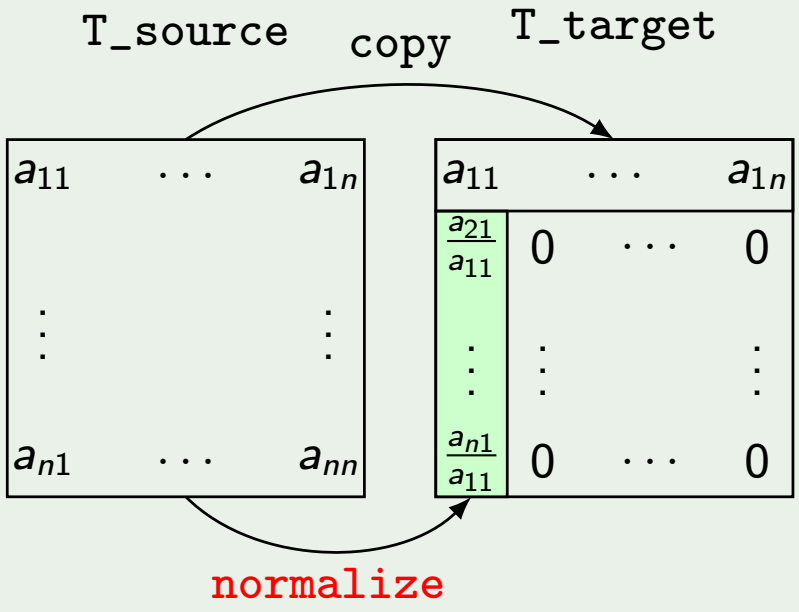
GPGPU implementace

První krok LU rozkladu



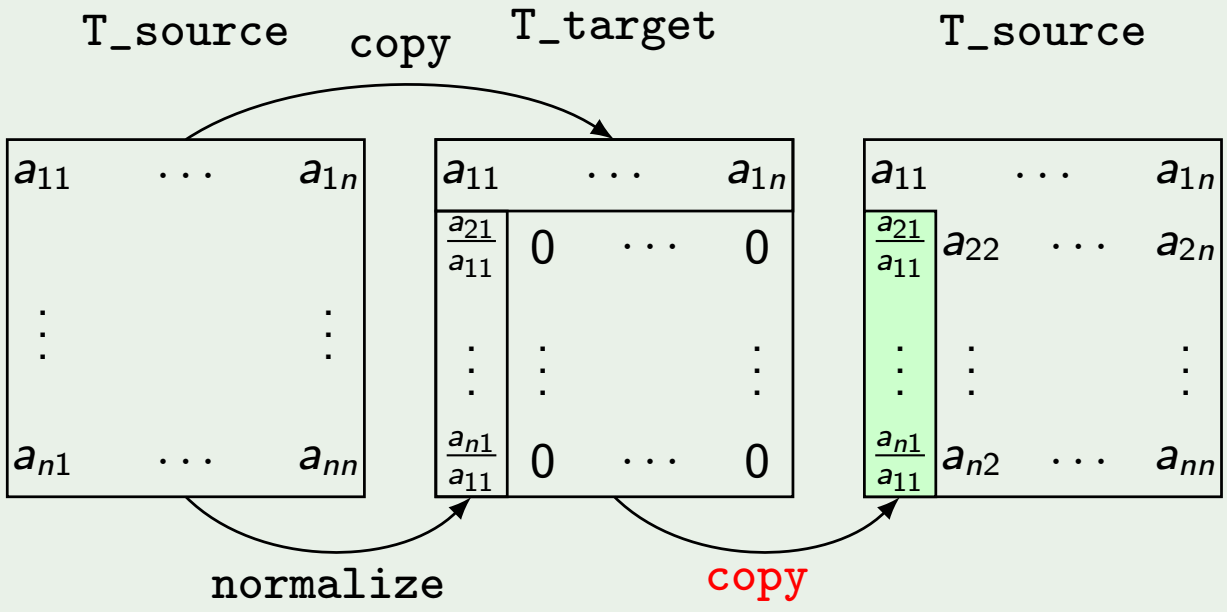
GPGPU implementace

První krok LU rozkladu



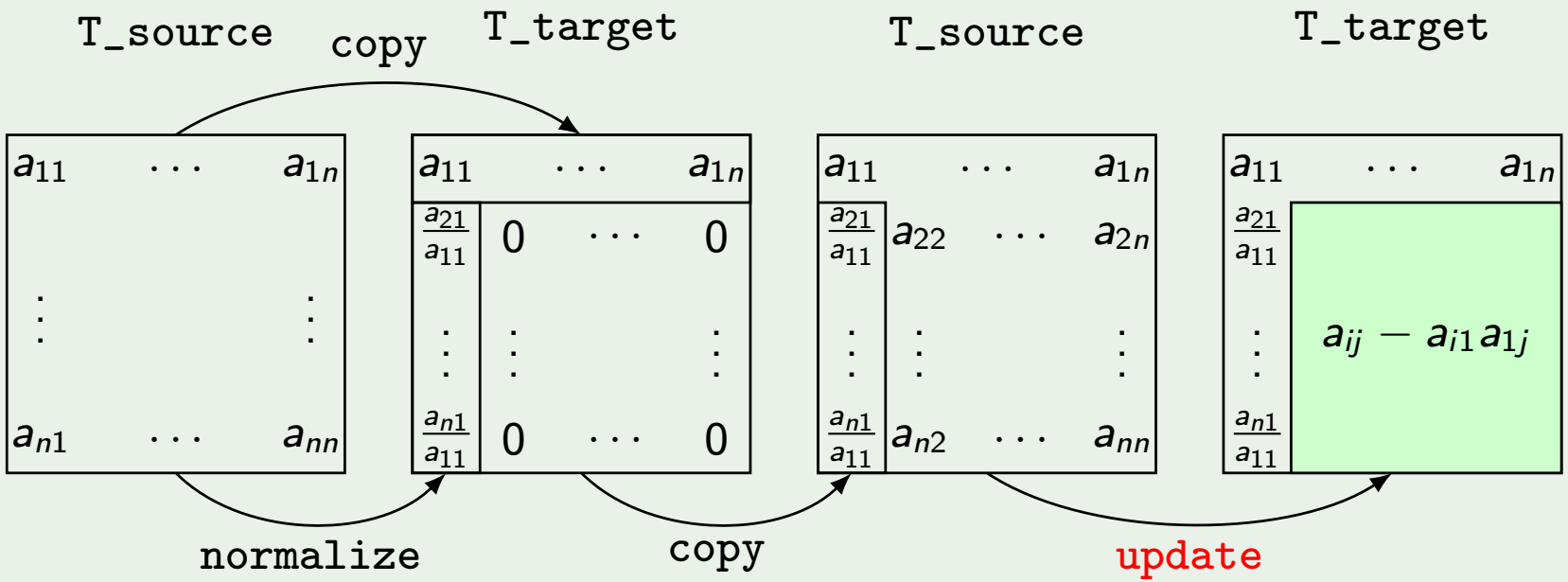
GPGPU implementace

První krok LU rozkladu



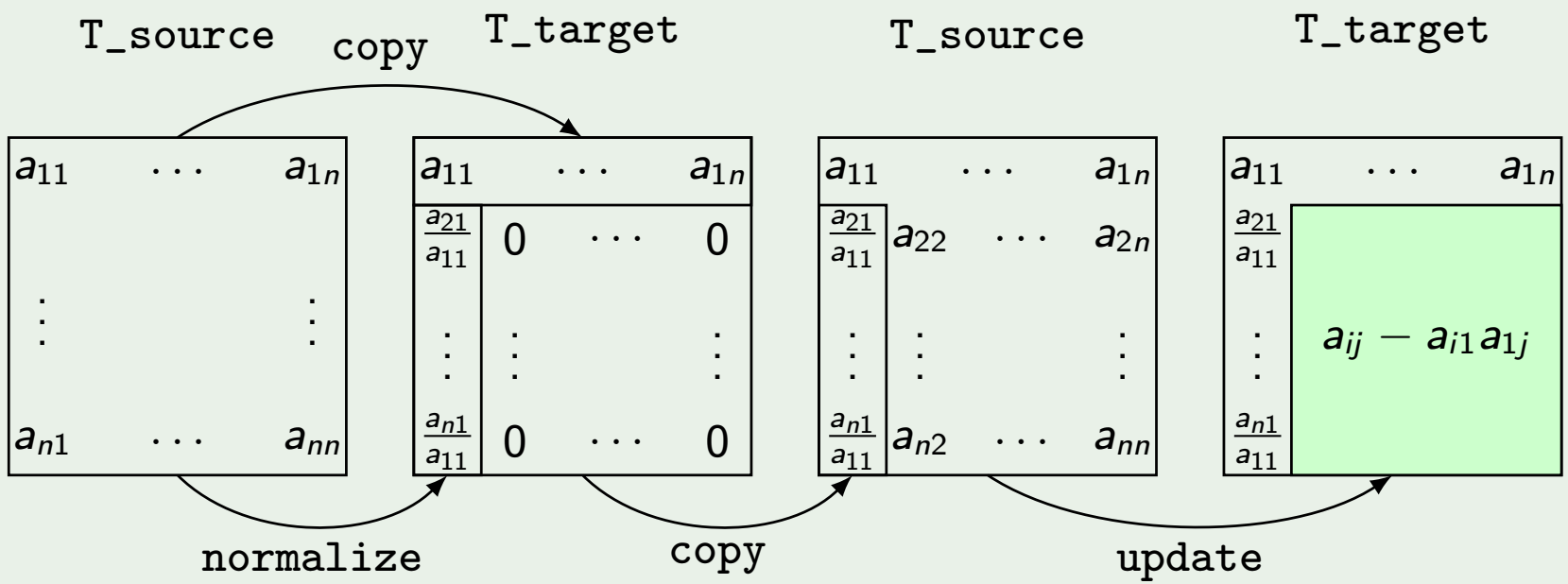
GPGPU implementace

První krok LU rozkladu



GPGPU implementace

První krok LU rozkladu



Hlavní znaky GPGPU implementace:

- Aritmetická intenzita
- Lze překonat optimalizované CPU implementace

Závěr

- Moderní GPGPU pojetí
  - NVIDIA CUDA
- Plánované pokračování práce
  - Měření výkonu a efektivity
  - Hledání dalších komplexnějších problémů, které lze efektivně řešit na GPU

Literatura


Jan Vacata. *GPGPU - Provádění vědeckých výpočtů prostřednictvím grafických karet osobních počítačů.* výzkumný úkol, 2007.


Matt Pharr (Ed.). *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation.* Addison-Wesley, 2005.


John D. Owens et al. *A Survey of General-Purpose Computation on Graphics Hardware.* Computers Graphics Forum 26(1):80-113, 2007.


`http://www.gpgpu.org`