Zkušenosti s GPU výpočty na clusteru Konos v rámci MetaCentra

Jan Vaněk a Jan Trmal

Katedra Kybernetiky
Západočeská Univerzita v Plzni

vanekyj@kky.zcu.cz

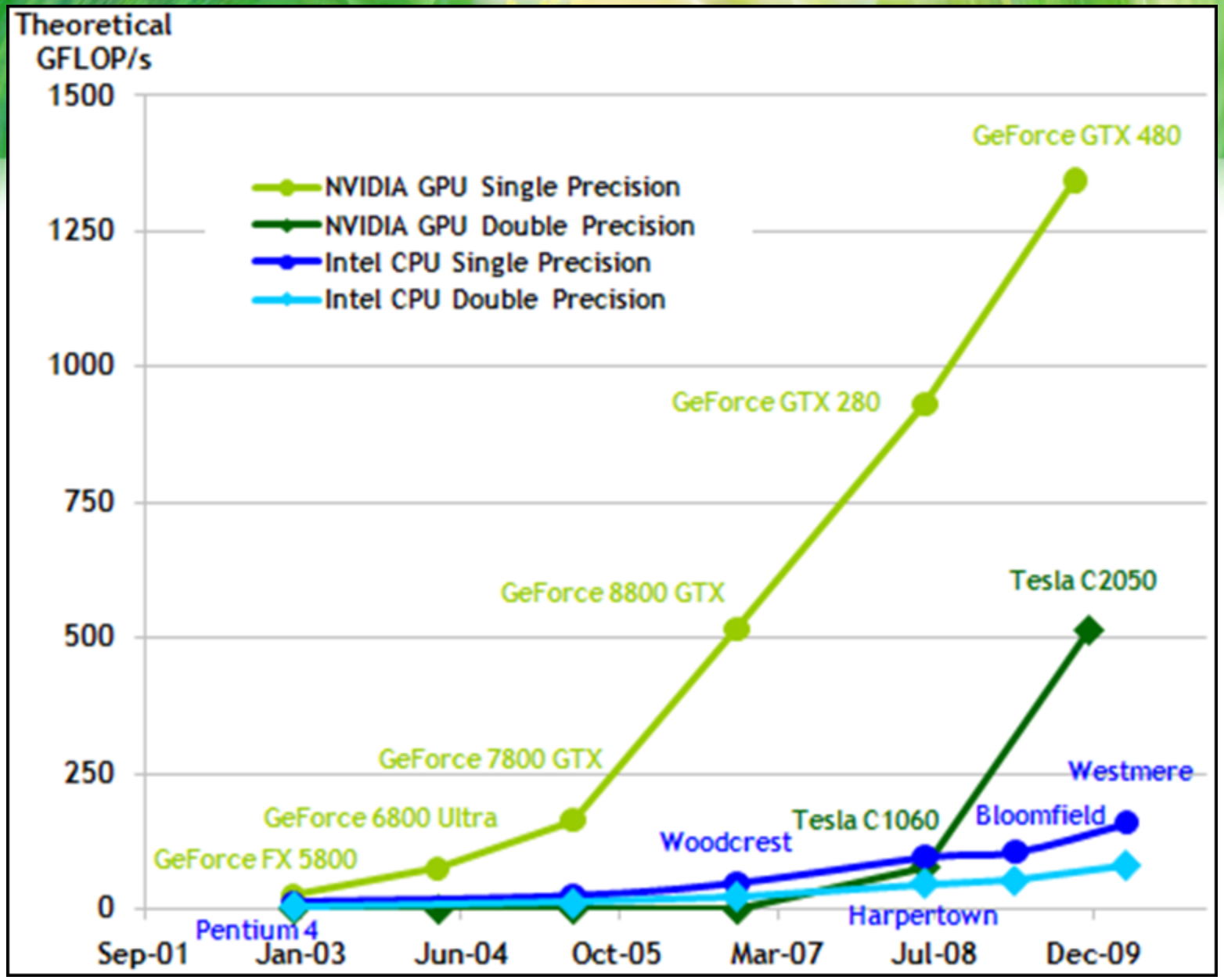
7.11.2011

Obsah

- Obecný pohled
  - Aplikace akcelerovatelné na GPU
  - Implementace úloh na GPU
- Praxe
  - Úlohy implementované na ZČU-KKY
  - Výsledky, porovnání NVidia a ATI/AMD
  - Zkušenosti s clusterem Konos (Metacentrum)
- Pohled vlastníka clusteru

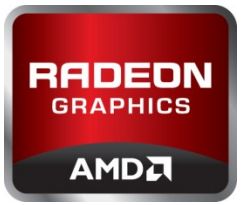
Obecný pohled

- GPU - vysoký výpočetní výkon
  - Hodí se jen pro některé úlohy
- Intel Core i7 6-core:
  - computation ~ **160-210** GFLOPS
  - memory speed ~ **20-40** GB/s
  - Price ~ **12–21** tis. Kč
- NVIDIA GeForce GTX 580
  - computation ~ **1.6** TFLOPS
  - memory speed ~ **192** GB/s
  - Price ~ **11-12** tis. Kč


Theoretical GFLOP/s
1500
1250
1000
750
500
250
0
NVIDIA GPU Single Precision
NVIDIA GPU Double Precision
Intel CPU Single Precision
Intel CPU Double Precision
GeForce GTX 480
GeForce GTX 280
GeForce 8800 GTX
GeForce 7800 GTX
GeForce 6800 Ultra
GeForce FX 5800
Tesla C2050
Tesla C1060
Westmere
Bloomfield
Woodcrest
Harpertown
Pentium 4
Sep-01
Jan-03
Jun-04
Oct-05
Mar-07
Jul-08
Dec-09

Hardware review

GPU	FLOPS(S/D)	Cena*
GF GT 240	260G/-	1 500 Kč
GF GTX465	880G/110G	3 700 Kč
GF GTX580	1.6T/200G	11 000 Kč
Tesla M2050	1.03T/515G	33 000 Kč

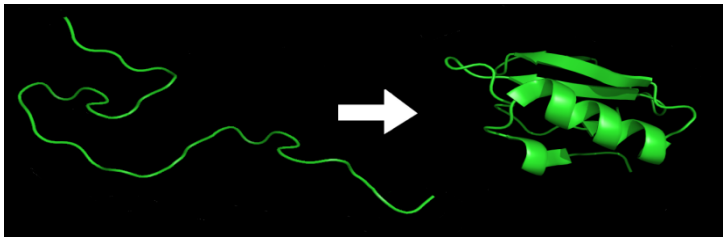


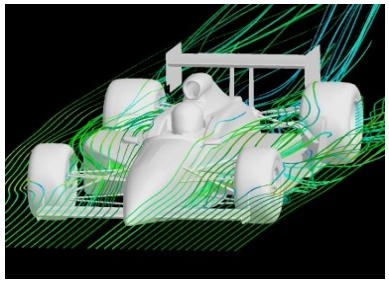
HD5670	620G/-	1 500 Kč
HD6850	1.5T/-	3 500 Kč
HD5870	2.7T/544G	7 000 Kč
HD6970	2.7T/683G	7 500 Kč

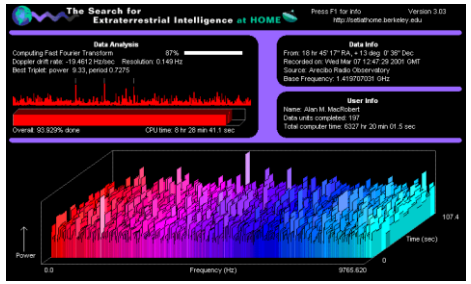
*přibližné ceny (říjen 2011)

Aplikace akcelerovatelné na GPU

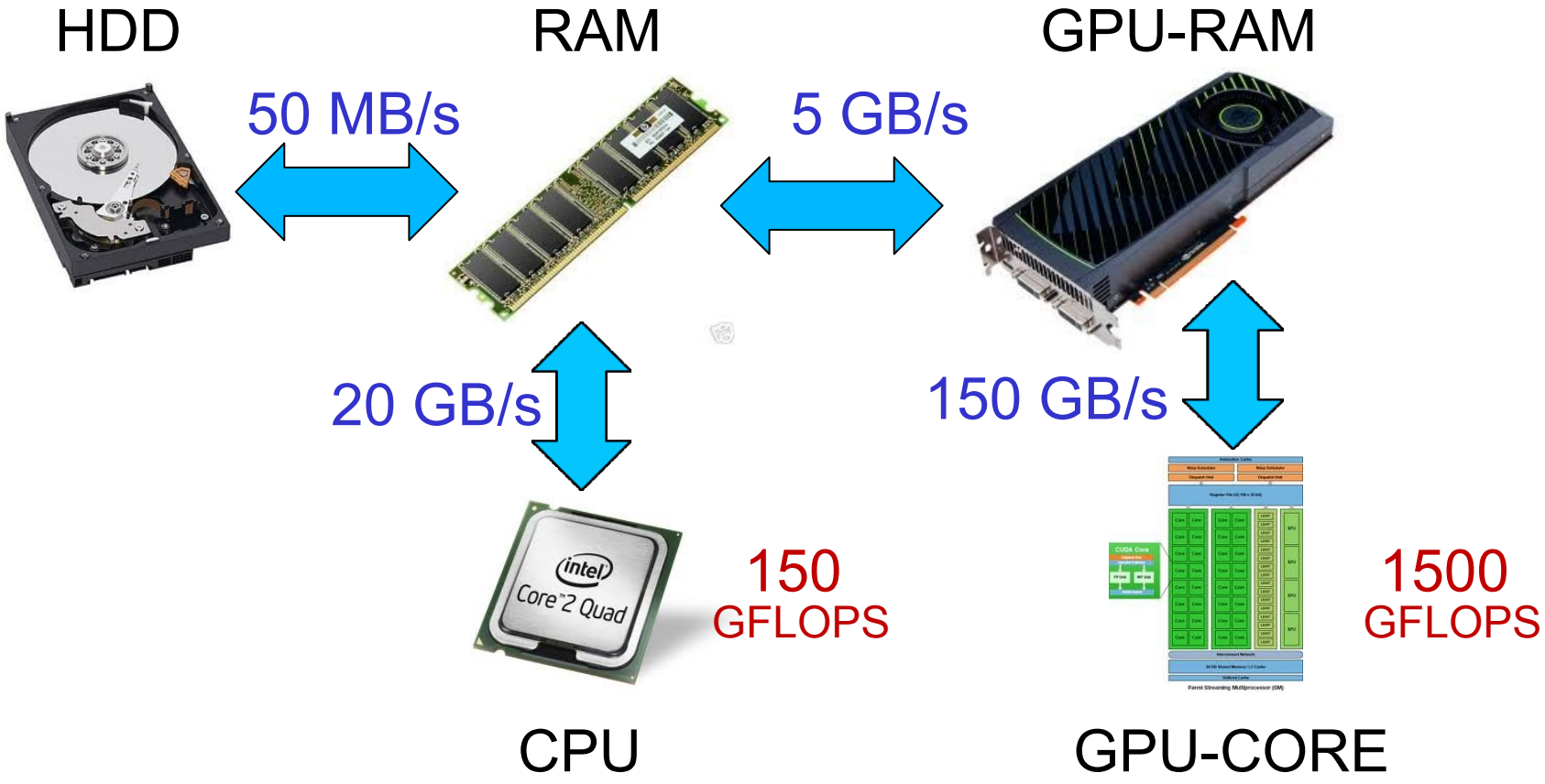
- Počítačová 3D grafika
- Zpracování dat
  - Zpracování obrazu/videa
  - Algoritmy umělé inteligence – např. klasifikátory
  - Statistická analýza/modelování
- Simulace
  - Od atomů po hvězdné soustavy
  - Různé vědecké oblasti (fyzika, chemie, ekonomie)
- Kryptografie, kryptoanalýza



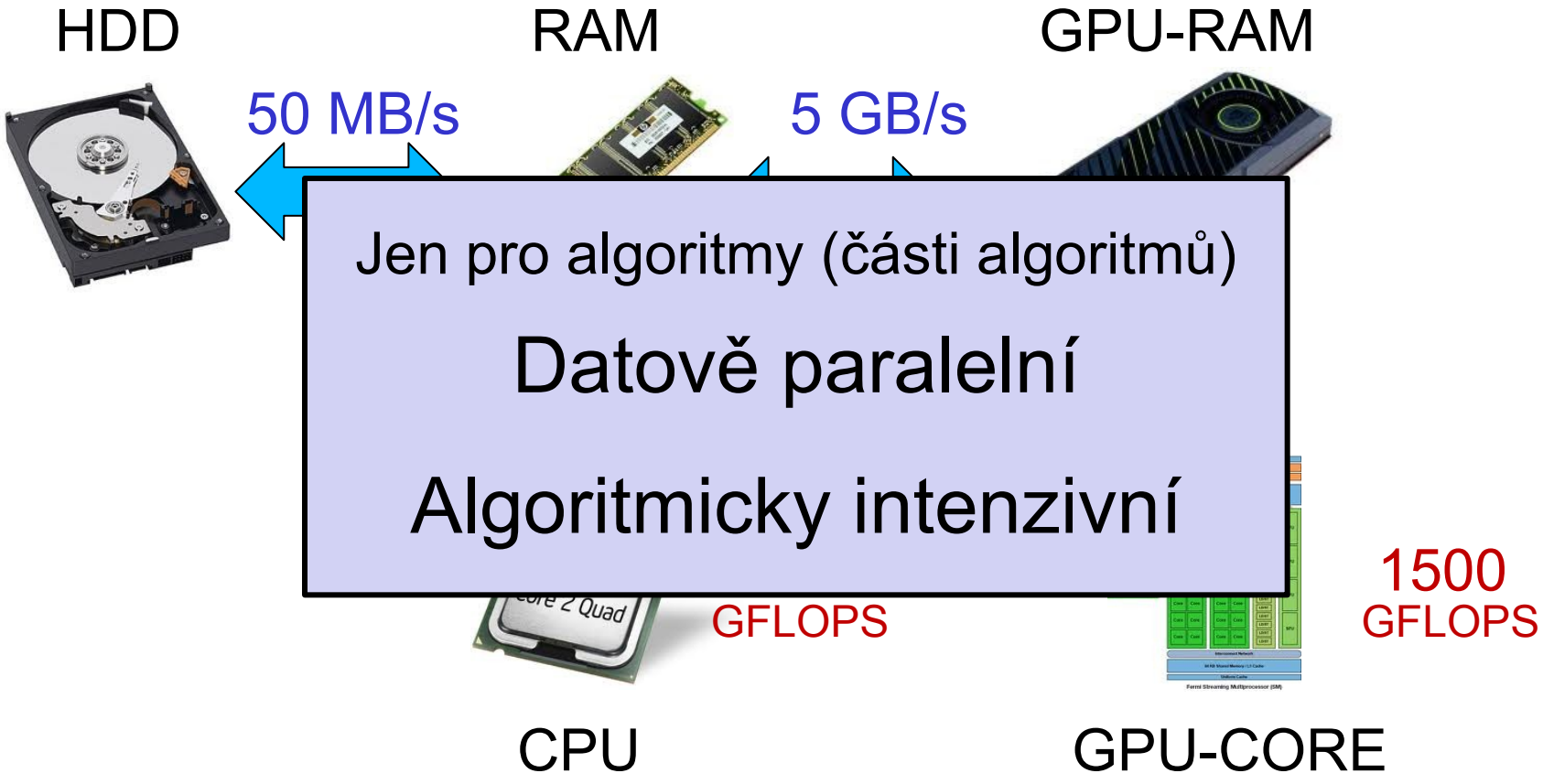




Ne všechny aplikace se hodí na GPU



Ne všechny aplikace se hodí na GPU



Úlohy dobře řešitelné na GPU

- Datově paralelní úlohy
  - Zcela nezávislé zpracování jednotlivých částí
  - Data na sobě závislá
    - Ignorování části závislostí – iterativní konvergující řešení
    - Iterativní úlohy (simulace, modelování, statistika)
      - Částečné překrývání dat, synchronizace jednou za čas
- Numericky intenzivní úlohy
  - Velký počet výpočetních operací, malá vstupní data
    - Kombinatorické úlohy (násobení matic, N^3 vs. N^2)
    - Iterativní úlohy

Jak získat GPU akcelerované programy

- Postupný nárůst GPU aplikací v posledních letech

- Akcelerované aplikace
- Akcelerované knihovny funkcí/algoritmů
- Matlab parallel computing toolbox
- Vlatní implementace v CUDA nebo OpenCL

Specifika GPU architektury

- velké množství procesorových jednotek
- rozdělené po částech do skupin s vlastní cache
- několik druhů pamětí se specifickým přístupem:
  - paměť RAM počítače – pro GPU přímo nepřístupná
  - paměťGPU – pomalá, umožňuje pro všechny SP Read/Write
  - Texture/constant cache – pro všechny SP, rychlejší, read-only
  - Shared (sdílená) paměť – pro jednu skupinu SP, R-W, rychlá, 16kB
- umožňuje rychle přepínat jednotlivá vlákna
  - jedno vlákno čeká na data, jiné počítá
- umožňuje načítat data na pozadí výpočtu – *memory latency hiding*

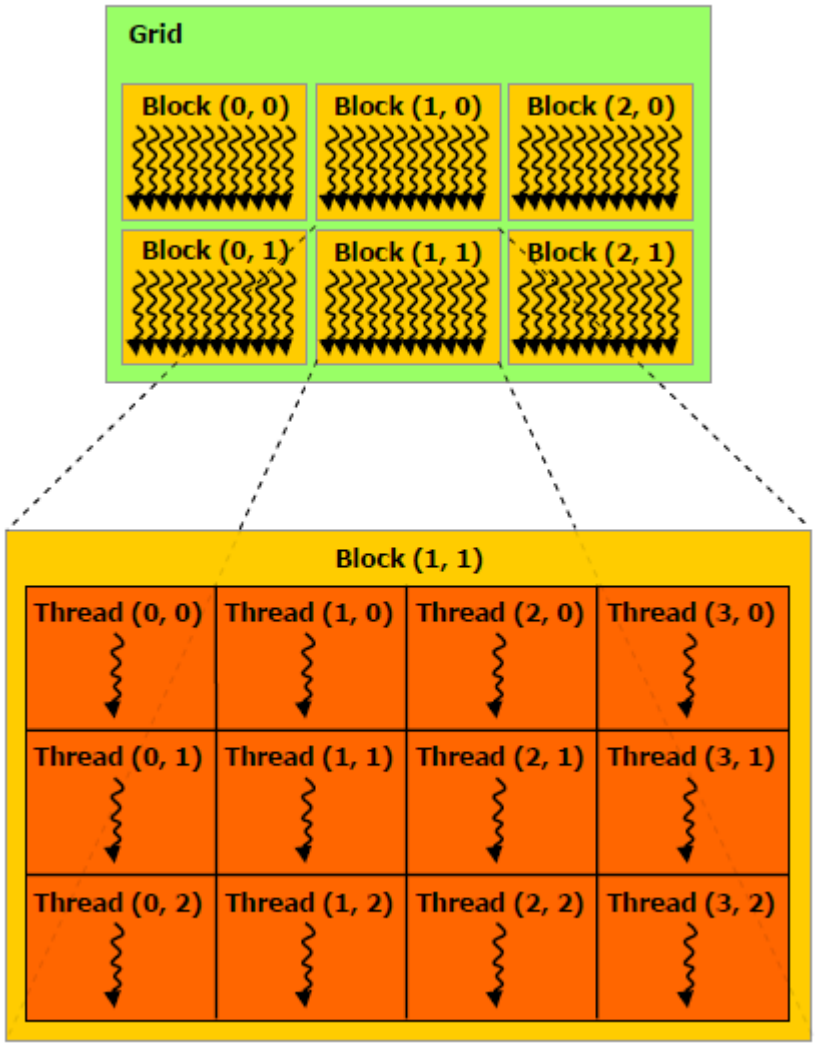
Obecné řešení

- na GPU se implementuje jenom výpočetní jádro algoritmu
- data se připraví předem v paměti RAM
- přenesou se do paměti GPU
- spustí se výpočet na GPU (CPU může dělat něco jiného)
- výsledky se přenesou zpět z GPU do RAM
- přenáší se větší bloky dat – větší datová prostupnost

- na GPU implementujeme výpočetně náročné části, které se dají snadno rozdělit na mnoho nezávislých částí
- paměťově náročné části programu (random read/write) ponecháváme na CPU (větší a rychlejší R/W cache)

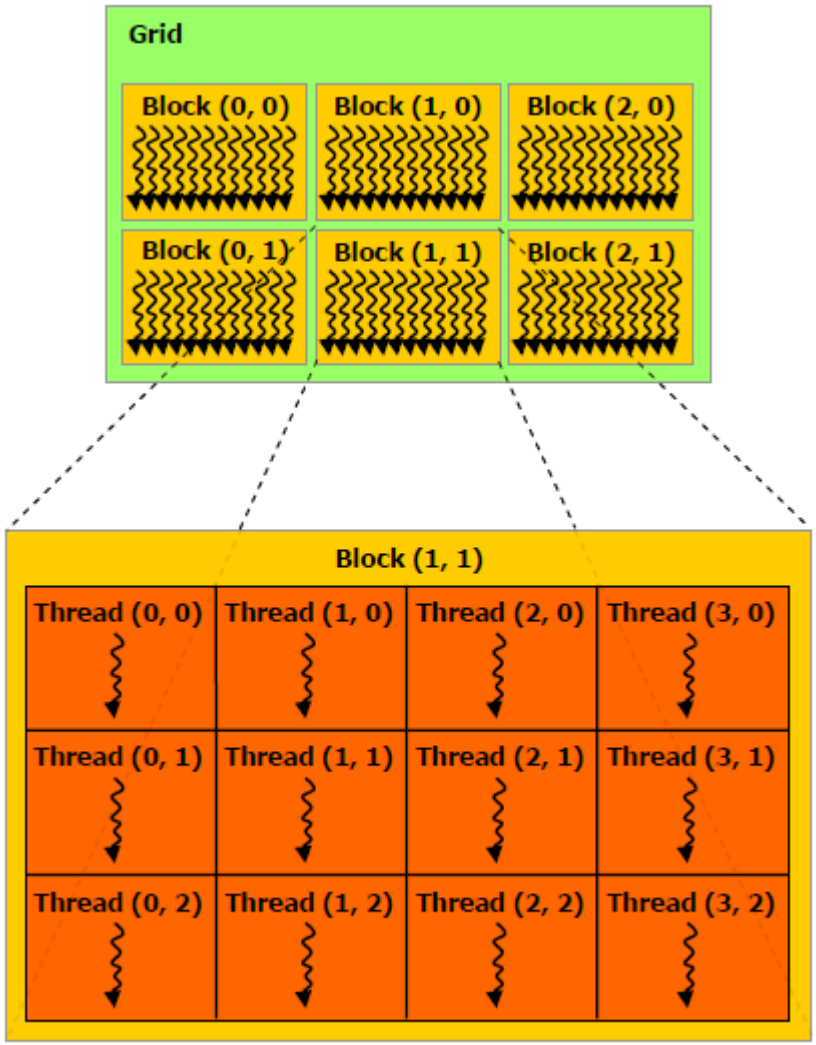
Datově-paralelní program

- Zpracovávaná data rozdělena do nezávislých bloků
- tyto bloky tvoří vektor či matici – *grid*
- programátor implementuje *kernel*, jádro výpočtu
- *kernel* definuje co se má s jedním prvkem matice dělat
  - co má dělat jedno vlákno
- rozdělení úloh mezi GPU procesory řídí ovladače a HW



Dvojitá hierarchická paralelizace

- Rozdělení do bloků
  - Jednotlivé multiprocesory
- Rozdělení na vlákna
  - Jednotlivé SP se SIMD
- Globální synchronizace jen na úrovni jednotlivých kernelů
  - vysoká režie
- Synchronizace vláken uvnitř jednoho bloku je možná
  - malá/zanedbatelná režie



Klíčové vlastnosti GPU implementací

- „Kulaté“ počty vláken, bloků paměti (32, 64, …)
  - přidávání virtuálních dat/výpočtů
  - speciální kód pro okrajové části
- Všechna vlákna *kernelu* dělají stejnou práci
  - nepoužívat nevhodné příkazy *if* a *cykly*
- Načítání dat v blocích
  - příprava dat podle algoritmu zpracování
- Využívání registrů, lokálních a cache-pamětí
- Načítání dat na pozadí - velké read/write a výpočetní bloky
- rozbalování cyklů (loop unrolling)

Historie GPU výpočtů na ZČU-KKY

- Začátkem roku 2008 první pokusy v CUDA a BrookGPU (BrookGPU+)
- Opuštění BrookGPU a karet ATI, přechod k NVidia a CUDA
- Postupná implementace vybraných metod
- Rozšiřovaní GPU výpočtů mezi ostatní kolegy
- NVidia a CUDA nyní jako standard v rámci naší skupiny
- Rozšíření clusteru Konos o CPU-GPU nody
- Implementace v OpenCL pro ATI/AMD

Implementované úlohy (1)

- Rozpoznávání řeči
  - výpočet pravděpodobností akustického modelu
- Trénování akustických modelů
  - výpočet pravděpodobností
  - Forward-Backward algoritmus
  - Akumulace statistik
- Rozpoznávání řečníka
  - Gaussian Mixture Models - GMM
  - Suport Vector Machines - SVM

Implementované úlohy (2)

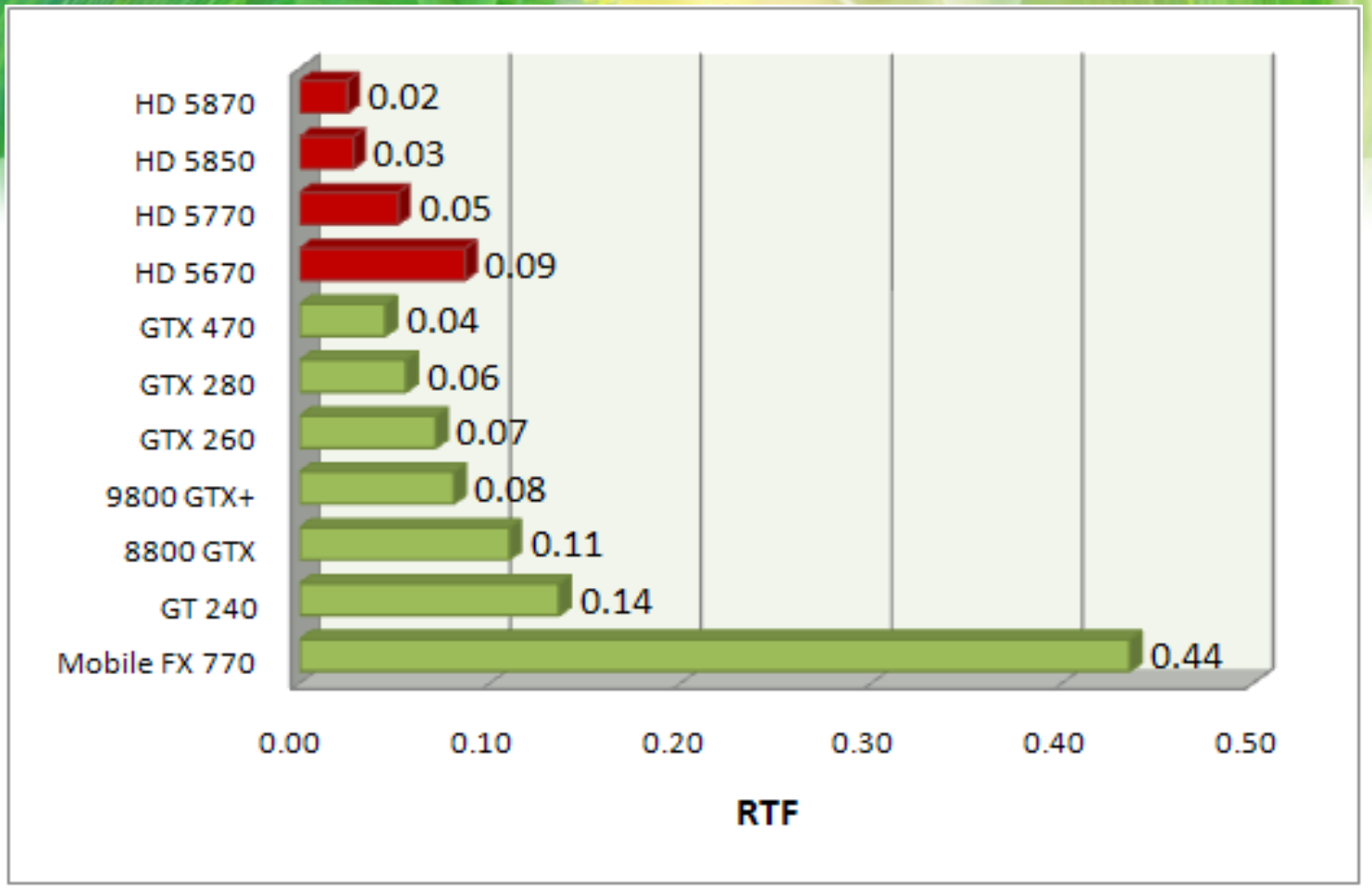
- Neuronové sítě
  - pro hybridní rozpoznávání řeči
  - trénování NN
    - několik milionů parametrů
    - desítky až stovky milionů trénovacích vektorů
    - iterativní optimalizační algoritmus (L-BFGS)
    - potřeba mnoha iterací (tisíce)
  - 32x zrychlení oproti CPU (1core, Intel LAPACK)

Implementované úlohy (3)

- Zpracování časových sérií mikroskopických snímků
  - Ústav Fyzikální Biologie, Nové Hrady
  - Analýza chování buněčných linií
  - 5 Mpix foto každou minutu – po dobu dnů
  - Fázový kontrast – potřeba speciálních metod
    - výpočet entopie pro každý pixel
  - 3600x zrychlení oproti první verzi v Matlabu
  - 40x zrychlení oproti C++ SSE verzi

Výsledky vybraných metod (1)

- Vyhodnocení akustického modelu
- Porovnání se skutečnou délkou zpracovávané řeči
  - Real Time Factor – RTF
  - doba zpracování / skutečná délka řeči
- Rozpoznávání řeči pracující v reálném čase
  - na CPU – kompromis rychlost/přesnost
  - na GPU – i low-end GPU jakýkoli model



- extrémně velký model (1,25 milionů Gaussianů)
- CPU RTF 10 – 30 v závislosti na použité aproximaci
- ATI/AMD cca 2x rychlejší

Porovnání NVidia vs. ATI/AMD

- Různá architektura
  - NVidia (Fermi based – GTX 4XX, GTX 5XX)
    - dvojí takt – jádro/zbytek GPU
    - 32x 1D výpočetní jednotky
    - 128kB registry
    - 48/16kB shared (local) memory
  - ATI/AMD
    - 16x 4D (5D) výpočetní jednotky
    - 256kB registry
    - 16kB shared (local) memory

- Nutné optimalizovat implementaci pro každou architekturu zvlášť!

Cluster Konos

- Náhrada části starých nodů novými
- Řešení projektu LINDAT-Clarin
- Celkem 10 nodů (8x KKY, 2x KMA)
  - Výběrové řízení zaměřeno na CPU výkon
  - GPU jen na akceleraci některých úloh
  - Konfigurace:
    - 2x 6-core Intel Xeon 3.2GHz
    - 24 GB RAM
    - 2x NVidia GeForce GTX 465
  - Reálná spotřeba cca 600W při plném zatížení

Spuštění GPU úlohy v MetaCentru (1)

- Připraven modul cuda-3.2-kky
  - CUDA i OpenCL
  - proměnné prostředí s cestami a definicemi
- Fronta GPU
- GPU - exkluzivní režim
  - Používaná grafická karta není dostupná pro další uživatele

Spuštění GPU úlohy v MetaCentru (2)

- Získáme členství ve frontě GPU
- Spustíme interaktivní job

```
qsub -l nodes=1:ppn=1 -q gpu -W grouplist=gpu -I
```

- Přidáme balíček cuda

```
module add cuda-3.2-kky
```

- Ověříme funkčnost (nepovinné, užitečné)

```
deviceQuery
bandwidthTest
```

- Spustíme svůj program

Reálné výsledky na Konosu1

- Trénování akustických modelů (200 hodin řeči)
  - 6-core CPU (Intel compiler, IPP), **3** hodiny/iteraci
- 1x GPU + 1-core CPU, **25** minut/iteraci
  - > 7x rychlejší
  - jen částečně vytížené 1-core CPU
  - možnost spuštění na více GPU
  - GPU kolem 80 C
- Rozpoznávání řeči (slovník 280 tisíc slov, trigram)
  - 8-core CPU – RTF 0,36
  - 8-core CPU+GPU – RTF 0,19 (GPU vytížení 20%)

Dekuji Vám za pozornost.