Fortran na GPU: CUDA Fortran a OpenACC direktivy

Nvidia GPU

Grafické procesory (GPU, device) firmy Nvidia mohou sloužit jako výpočetní koprocesory neboli akcelerátory, na které může hlavní procesor (CPU) odkládat k provedení výpočetně náročné části programů. GPU obsahuje multiprocesory (streaming multiprocessors), ty obsahují jádra (CUDA cores); jader v GPU jsou stovky až tisíce. Pro počítání na GPU musí programátor vyčlenit vhodnou část zdrojového kódu, explicitně nebo pomocí direktiv. Tato část (kernel) se pak provádí v mnoha paralelních vláknech (threads), sdružených do bloků (blocks) a s nimi do mřížky (grid); členění hardwarových jader do relativně nezávislých multiprocesorů koresponduje s členěním softwarových vláken do bloků.
Algoritmy vhodné pro GPU obsahují paralelizovatelné cykly o velkém počtu průchodů (tisíce, milióny a více) s výhodným (vysokým) poměrem float-to-byte (více operací než dat); velikost zpracovávaných dat by neměla přesáhnout velikost paměti GPU, tedy jednotky GB. Vhodné k přenosu na GPU bývají např. algoritmy lineární algebry; např. násobení matice s vektorem však zahrnuje $O(n^2)$ operací i datových prvků a výpočetní síla GPU tak nemusí být využita naplno, zatímco maticové násobení s $O(n^3)$ operacemi na $O(n^2)$ datech je nadějnější. Jiné nadějné úlohy spadají do rodiny Monte Carlo metod, provádějících totožné, ale vzájemně nezávislé operace pro mnoho různých výchozích konfigurací. Netriviální je aplikace GPU na metodu konečných diferencí, kde počet operací je úměrný počtu zpracovávaných dat; výpočetní oblast bývá potřeba rozdělit mezi bloky vláken, vlákna je pak třeba synchronizovat, navíc se výkonné implementace musejí postarat o explicitní využití cache paměti multiprocesorů.
Vývojové generace GPU, z pohledu programátora popsané pomocí compute capability (cc):
2006: cc 1.1 (pouze single precision real)
2008: cc 1.3 (poprvé double precision); př. GeForce GTX 260; u nás geof10-20
2010: Fermi alias cc 2.0, 2.1 (stovky jader, vysoký výkon DP, cache paměť); př. GTX 470, 570; geof30-50, lojzik
2012: Kepler alias cc 3.0 (až tisíce jader); př. GTX 650, 680; geof60-90, vaclav
2013: cc 3.5; př. GTX Titan (2688 SP jader, třetina DP jader, první GeForce s 6 GB, 25 tis. Kč); geoffg
dál jen 2 směry: výše... Nvidia Tesla karty, př. Kepler K20 (2496 jader, 5 GB, vysoký DP výkon, 94 tis. Kč)
jinam... Intel Xeon Phi (MIC, many integrated core architecture, ~50 tis., zatím nedostupné)
Mezigeneračně se zvyšují počty (zvláště SP) jader na kartě i v multiprocesorech (1.x: 8, 2.x: 32/48, 3.x: 192 jader na multiprocesor); paměť GPU řady GeForce dosud nepřesáhla 2 GB, výjimkou je nový Titan s 6 GB.
Přehled našich GPU: geo.mff.cuni.cz/documents/hw-geof.pdf, další podrobnosti: programy pgaccelinfo, nvidia-smi.

Software pro GPU

volný Nvidia CUDA Toolkit pro Linux i Windows
- Nvidia C překladač (nvcc)
- Nvidia knihovny pro lineární algebru (cublas), FFT (cufft) ad. (pro lib. překladač C a Fortranu, Linux, Win)

knihovna CULA Tools Dense & Sparse s algebraickými řešiči pro plné a řídké matice (lib. překladač, Linux, Win)
knihovna MAGMA: hybridní implementace BLASu a LAPACKu pro CPU a GPU
komerční vývojový systém Portland (PGI Accelerator) pro Linux (geof30) i Windows
- překladače Fortran a C (pgfortran a pgcc), převzatá GUI prostředí, ACML a IMSL knihovny, OpenMP, MPICH
- rozšíření fortranského překladače CUDA Fortran (CUF) a Kernel Loop direktiva (!$CUF KERNEL DO)
- akcelerační OpenACC direktivy (!$ACC KERNELS LOOP aj.)

Na výběr je tedy buď nativní Nvidia C, nebo hotové knihovny, nebo PGI CUDA Fortran, nebo OpenACC direktivy realizované rovněž (jen) v PGI Fortranu.

Ukázky programů

Zdrojové kódy následujících programů jsou k nalezení na webu semináře. K jejich překladu je nezbytný PGI Fortran.
Č. 1. Syntaktická ukázka: kopie pole

```
CPU:    real a(nmax),b(nmax)
        do i=1,nmax ; b(i)=a(i) ; enddo
```

Č. 2. Nejopuštěnější: nalezení bodu s největší nejmenší vzdáleností od ostatních bodů, $\max_p(\min_q(\mathrm{dist}(\mathrm{bod}_p,\mathrm{bod}_q)))$. Výpočet zahrnuje stanovení vzdáleností mezi každými dvěma body, poměr float-to-byte je tedy nadějný: $O(n^2)$ operací, $O(n)$ dat.
Č. 3. Lorenzův atraktor: posun mnoha nezávislých bodů po trajektorii popsané systémem obyčejných diferenciálních rovnic pro Lorenzův atraktor. Každý bod může být řešen jedním vláknem; počet operací je sice úměrný počtu dat, ale konstanta úměrnosti je velká. Výstup: vizualizace atraktoru prostřednictvím koncových poloh bodů. Pro zkrácení doby běhu kernelu (např. pro udržení grafické interaktivity počítače) lze kernel volat opakovaně pro malý počet kroků; režie jednoho volání kernelu je 10^{-5} s.

PGI CUDA Fortran

Programátor vyhledá část kódu vhodnou k provedení na GPU. Přetvoří ji na modulový podprogram, podprogram přetvoří na kernel, deklaruje kopie svých dat na GPU, zabezpečí přenosy dat mezi CPU a GPU, ovlivní rozložení dat v interní paměti GPU, konfiguruje vlákna do mřížky bloků. Více na dřívějších seminářích.

Př.

```
module m ; contains
attributes(global) subroutine Kernel(a,b,nmax)
real,device :: a(nmax),b(nmax)
integer,value :: nmax
i=threadidx%x+blockdim%x*(blockidx%x-1)
if (i<=nmax) b(i)=a(i)
end subroutine
end module
```

```
use m
integer,parameter :: nmax=100000
integer,parameter :: block=64,grid=(nmax-1)/block+1
real a(nmax),b(nmax)
real,device :: ad(nmax),bd(nmax)
a=1. ; ad=a
call Kernel<<<grid,block>>>(ad,bd,nmax)
b=bd
```

Postupy blízké originálnímu Nvidia C překladači; pracné, spolehlivé.
Překlad: pgfortran -Mcuda file.f90; pgfortran file.cuf; jiné volby: -fast, -Mcuda=cc20,cc30,cc35 (geof30 default: cc20).

PGI Kernel Loop direktiva

Programátor vyhledá cyklus, deklaruje data na GPU, zabezpečí jejich přenosy, přidá direktivu s volitelnou konfigurací vláken do mřížky. Systém sám přetvoří cyklus na kernel.

Př.

```
real a(nmax),b(nmax)
real,device :: ad(nmax),bd(nmax)
ad=a
!$CUF KERNEL DO <<<*,*>>>
do i=1,nmax ; bd(i)=ad(i) ; enddo
b=bd
```

Explicitní velikost bloku: př. <<<*,64>>>.
Překlad: pgfortran -Mcuda file.f90. Potíž s překladem na 32bitových Windows.

OpenACC direktivy

Programátor vyhledá paralelizovatelný kód, obloží ho direktivami, jejich klauzulemi popíše požadované přenosy dat a volitelnou konfiguraci mřížky. Systém se pokusí o zbytek. Zdrojový kód lze i nadále překládat pro chod na CPU libovolným překladačem. Direktiv i klauzulí je celá řada (viz specifikace nebo přehledová karta, odkazy níže).

Př.

```
real a(nmax),b(nmax)
!$ACC KERNELS LOOP COPYIN (a) COPYOUT (b)
do i=1,nmax ; b(i)=a(i) ; enddo
!$ACC END KERNELS
```

Klauzule pro explicitní velikost bloku: př. VECTOR (64).
Překlad: pgfortran -acc file.f90. Pozor na překlad bez -acc, program pak běží jen na CPU.

Drobné tipy

Vhodný počet vláken v bloku: násobky 32, max. 1024 (vlákna jednoho bloku běží na jednom multiprocesoru, jejich počet je proto shora omezen).
Vhodný počet bloků v mřížce: násobky počtu multiprocesorů; počet multiprocesorů pro GTX 260: 27, 470: 14, 650: 2, 680: 8, Titan: 14 (počet bloků shora prakticky omezen není).
Linux: pgcudainit pro eliminaci inicializační prodlevy, service gdm stop pro eliminaci timeoutu (~10 s) pro kernel.
Windows: timeout (~2 s) eliminovatelný nastavením v registrech.

Odkazy

web semináře	http://geo.mff.cuni.cz/jednooci_slepym
naše GPU	http://geo.mff.cuni.cz/documents/hw-geof.pdf
Nvidia CUDA	http://developer.nvidia.com/cuda
CULA Tools	http://www.culatools.com
Portland CUDA Fortran	http://www.pgroup.com/doc/pgicudaforug.pdf
OpenACC, specifikace a dokumenty	http://www.openacc.org
	http://www.openacc.org/sites/default/files/OpenACC_API_QuickRefGuide.pdf

Příloha: zdrojové soubory a doba běhu

	CPU	GPU+CUDA Fortran	GPU+!$CUF	GPU+!$ACC
pgfortran	-fast	-Mcuda	-Mcuda	-acc
ukázka č. 1	1.f90	1C.f90	1K.f90	1A.f90
ukázka č. 2	nej.f90	nejC.f90	nejK.f90	nejA.f90
i7/470/650/680/Titan	...			
ukázka č. 3	lor.f90	lorC.f90	lorK.f90	lorA.f90
i7/470/650/680/Titan	...			

Testy: hardware: i7-3930K/3.2 GHz (1 jádro), Nvidia GeForce GTX 470, GTX 650, GTX 680, GTX Titan
č. 2 nmax=10^5, č. 3 nmax=...

L. H., 7. 5. 2013