Portland akcelerátor (PGI Accelerator)

akcelerátor = (hardwarově) GPU, (softwarově) součást překladače vytvářející kód pro GPU
softwarový akcelerátor
– rozhraní pro Fortran a C ve stylu OpenMP
– direktivy, moduly a procedury, proměnné prostředí
– umožňuje zapsat jeden zdrojový kód a překládat jej jak pro kombinaci CPU/GPU, tak jen pro CPU

Příprava, konfigurace a překlad zdrojového kódu

instalace driveru s CUDA (www.nvidia.com)
testovací utilita pgaccelinfo
překladače Fortranu 95 a C99: pgfortran, pgcc
překlad: pgfortran -ta=nvidia -Minfo file.f90
- -ta … target accelerator flag, -Minfo … informational messages enabled
- -ta=nvidia,suboptions … suboptions: analysis, cc10, cc11, cc13, fastmath, time aj.
- -ta=nvidia,time pro aktivaci profileru
- -ta=host pro emulaci
- -Minfo=accel pro omezení zpráv

kombinace s OpenMP a MPI možná při více GPU

Direktivy akcelerátoru

syntaxe direktiv akcelerátoru
- !$ACC directive-name [clause [,clause] …]

označení akcelerované výpočetní oblasti
- !$ACC REGION // !$ACC END REGION
- klauzule IF (condition), COPY (list), COPYIN (list), COPYOUT (list), LOCAL (list), UPDATE DEVICE (list), UPDATE HOST (list)

označení akcelerované datové oblasti
- !$ACC DATA REGION // !$ACC END DATA REGION
- klauzule COPY (list), COPYIN (list), COPYOUT (list), LOCAL (list), MIRROR (list), UPDATE DEVICE (list), UPDATE HOST (list)

nastavení způsobu akcelerace cyklu
- !$ACC DO, kombinovaná direktiva !$ACC REGION DO
- klauzule CACHE, HOST, INDEPENDENT, KERNEL, PARALLEL, PRIVATE, SEQ, SHORTLOOP, UNROLL, VECTOR

nastavení vlastností dat z implicitně akcelerovaných oblastí (procedury, moduly)
- !$ACC COPY (list) …
- další klauzule COPYIN, COPYOUT, LOCAL, MIRROR, REFLECTED

synchronizace dat v paměti hostitele a zařízení
- !$ACC UPDATE
- klauzule HOST (list), DEVICE (list)

Procedury a proměnné akcelerátoru

modul accel_lib s definicemi konstant a rozhraní procedur
procedury
– nastavení zařízení: acc_set_device, acc_set_device_num
– explicitní start a stop: acc_init, acc_shutdown
– dotazovací procedury: acc_get_device, acc_get_num_devices, acc_on_device

př. explicitní vynucení CPU/GPU: USE accel_lib ; CALL acc_set_device(acc_device_host)
př. explicitní inicializace GPU: USE accel_lib ; CALL acc_init(acc_device_nvidia)

proměnné prostředí
- ACC_DEVICE … hodnoty NVIDIA pro GPU, HOST pro emulaci na CPU
- ACC_DEVICE_NUM … hodnota od 1 výše pro číslo GPU
- ACC_NOTIFY … hodnota 1 pro runtime informaci „launch kernel …"

soubor siteenvrc (?)
- set COMPUTECAP=11 … pro computing capability 1.1 (default 1.3)

Klíčová témata

– paralelizovatelný algoritmus
– přenos dat mezi hostitelem a GPU
– přenos dat v rámci GPU
– rozvrhování cyklů (loop scheduling)
– analýza výkonu (profiling)
pozn. o inicializaci GPU: od 10^{-5} do 10^{1} s, závisí na stavu driveru zařízení (trvalé připojení na pozadí: pgcudainit)

Přenos dat mezi hostitelem a GPU

– potlačení automatického přenosu dat lokálních v oblasti (REGION LOCAL)
– přenášení spojitých (a zarovnaných) sekcí pole (REGION COPY COPYIN COPYOUT)

Přenos dat v rámci GPU

– data jsou poskytována polosvazkům (half-warps, 16 vláken) po superslovech (16 slov po 4 B = 64 B)
– ideál: když polosvazek získá 64 B (spojitých a zarovnaných) dat přenosem jednoho superslova naráz
(sdružený přístup do paměti, memory coalescing)
– opak: když polosvazek vyžaduje nespojitá data pomocí až 16 superslov
– preference spojitých a zarovnaných sekcí polí, tj. cykly s krokem 1, zarovnávání dimenzí polí na násobky 16
– snaha překladače užívat softwarovou cache (lokální paměť SM), též klauzule DO CACHE (list)

Rozvrhování cyklů (klauzule direktivy DO)

– hostitelský mód	DO HOST (width)	pro sekvenční provádění (částí) cyklu na hostiteli
– sekvenční mód	DO SEQ (width)	pro sekvenční provádění (částí) cyklu na GPU
– vektorový mód	DO VECTOR (width)	pro SIMD provádění několika iterací (max. 256) na jednom SM
– paralelní mód	DO PARALLEL (width)	pro paralelní provádění iterací na více SM
– kernel mód	DO KERNEL	specifikuje samotný kernel (cykly uvnitř již sekvenční)

– vektorový mód vytváří blok, paralelní mód mřížku
– default mód není, překladač volí automaticky; hodnoty width mohou být omezeny (př. mocniny 2)
– více klauzulí nebo uvedení (width) vede k restrukturalizaci cyklu (strip-mining) na vnější a vnitřní
– klauzule DO INDEPENDENT zvýrazní, že data v jednotlivých iteracích jsou vzájemně nezávislá
– klauzule
– př. DO HOST (16) PARALLEL pro 16 iterací na hostiteli, během kterých běží vnitřní cyklus na různých SM
DO PARALLEL VECTOR (256) rozdělí cyklus na vnitřní o 256 iteracích (blok 256 vláken běžící na 1 SM)
a vnější, který provádí své iterace paralelně (mřížka bloků běžících na různých SM)
platí relace typu: i = BlockIdx%x * BlockDim%x + ThreadIdx%x, x od 0

Analýza výkonu (time profiling)

– profiler akcelerátoru: -ta=nvidia,time
informace o počtu oblastí a kernelů a čase pro inicializaci, datové přenosy a běh kernelů
– fortranské podprogramy:
call cpu_time(time) … real time
call system_clock(count,count_rate,count_max) … integer count, count_rate, count_max
– cudaprof od NVIDIA

Překážky paralelizovatelnosti cyklů

loop carried scalar dependence for 'x'
- (ú)sporné skaláry v cyklech (přiřazení v jedné iteraci, použití v druhé)
- podmíněné užívání skaláru (ve více blocích, byť s toutéž podmínkou)
- redukční cykly (př. skalární součin, prý jen dočasný problém překladače)

scalar last value needed after loop for 'x'
- skalár za cyklem pracující s poslední hodnotou z cyklu (často zbytečné)

loop carried dependence of 'a' prevents parallelization
- může pomoci rozložení cyklu do více jednodušších cyklů
- rekurentní relace (existují paralelizovatelné algoritmy pro jejich vyčíslování)
- vektorové indexy na levé straně přiřazení (na pravé straně vadit nemusí)

parallelization would require privatization of array 'a'
- překladač v cyklech automaticky privatizuje skaláry, pole nikoliv

podmíněné příkazy ve vnořených cyklech
- překladač neparalelizuje podmíněné vnitřní cykly (podmínku však lze přesunout do vnitřního cyklu)

a jiná omezení akcelerátoru nebo překladače
- unsupported operation
- function/procedure calls not supported
- struct/member references not supported
- datatype not supported

loop is parallelizable
- paralizovatelný, ale neparalelizovaný cyklus

accelerator kernel generated
- cyklus zparalelizován, z těla cyklu vytvořen kernel, další téma: rozvržení cyklu (loop scheduling)

pgaccelinfo na našich GPU

GT260 Twin Frozr/Batman

```
CUDA Driver Version           3000/2030
Device Number:                0
Device Name:                  GeForce GTX 260
Device Revision Number:       1.3
Global Memory Size:           939196416
Number of Multiprocessors:    27    (má být 24?)
Number of Cores:              216
Concurrent Copy and Execution: Yes
Total Constant Memory:        65536
Total Shared Memory per Block: 16384
Registers per Block:          16384
Warp Size:                    32
Maximum Threads per Block:    512
Maximum Block Dimensions:     512 x 512 x 64
Maximum Grid Dimensions:      65535 x 65535 x 1
Maximum Memory Pitch:         2147483647B/262144B
Texture Alignment             256B
Clock Rate:                   1408 MHz
Current free memory           894635776
```

8400GS

```
CUDA Driver Version           3000
Device Number:                0
Device Name:                  GeForce 8400 GS
Device Revision Number:       1.1
Global Memory Size:           536543232
Number of Multiprocessors:    1     (má být 2?)
Number of Cores:              8
Concurrent Copy and Execution: No
Total Constant Memory:        65536
Total Shared Memory per Block: 16384
Registers per Block:          8192
Warp Size:                    32
Maximum Threads per Block:    512
Maximum Block Dimensions:     512 x 512 x 64
Maximum Grid Dimensions:      65535 x 65535 x 1
Maximum Memory Pitch:         2147483647B
Texture Alignment             256B
Clock Rate:                   1400 MHz
Current free memory           516685824
```

Odkazy

PGI Fortran & C Accelerator Programming Model (v. 1.0 June 2009)
PGI User's Guide – Parallel Fortran, C and C++ for Scientists and Engineers (v. 10.2 February 2010)
PGI CUDA Fortran Programming Guide and Reference (v. 1.2 March 2010)

NVIDIA CUDA Programming Guide (v. 2.3.1 August 2009)
NVIDIA CUDA CUBLAS Library (v. 2.3 June 2009)
NVIDIA CUDA CUFFT Library (v. 2.3 June 2009)

http://www.root.cz/serialy/uvod-do-technologie-cuda