Paralelizacija algoritama na heterogenim platformama kroz sustav OpenCL

Veljko Dragšić

3. ožujka 2010.

Uvod

- Paralelni algoritmi
  - CPU slijedno izvodi naredbe –> nedovoljno brzo
  - više CPU-ova izvodi naredbe –> ubrzanje
  - trebamo paralelizirati algoritme –> razne tehnologije i principi
- Heterogene platforme
  - CPU: x86, x86_64, *Power*, *ARM*, ...
  - GPU: *Nvidia*, *ATI*, ...
  - *"hibridne"* arhitekture: *Cell*, ...
- OpenCL
  - sustav za paralelizaciju algoritama
  - podržava heterogene platforme (prvi takav)

Uvod

- Paralelni algoritmi
  - CPU slijedno izvodi naredbe –> nedovoljno brzo
  - više CPU-ova izvodi naredbe –> ubrzanje
  - trebamo paralelizirati algoritme –> razne tehnologije i principi
- Heterogene platforme
  - CPU: x86, x86_64, *Power*, *ARM*, ...
  - GPU: *Nvidia*, *ATI*, ...
  - *"hibridne"* arhitekture: *Cell*, ...
- OpenCL
  - sustav za paralelizaciju algoritama
  - podržava heterogene platforme (prvi takav)

Uvod

- Paralelni algoritmi
  - CPU slijedno izvodi naredbe –> nedovoljno brzo
  - više CPU-ova izvodi naredbe –> ubrzanje
  - trebamo paralelizirati algoritme –> razne tehnologije i principi
- Heterogene platforme
  - CPU: x86, x86_64, *Power*, *ARM*, ...
  - GPU: *Nvidia*, *ATI*, ...
  - *"hibridne"* arhitekture: *Cell*, ...
- OpenCL
  - sustav za paralelizaciju algoritama
  - podržava heterogene platforme (prvi takav)

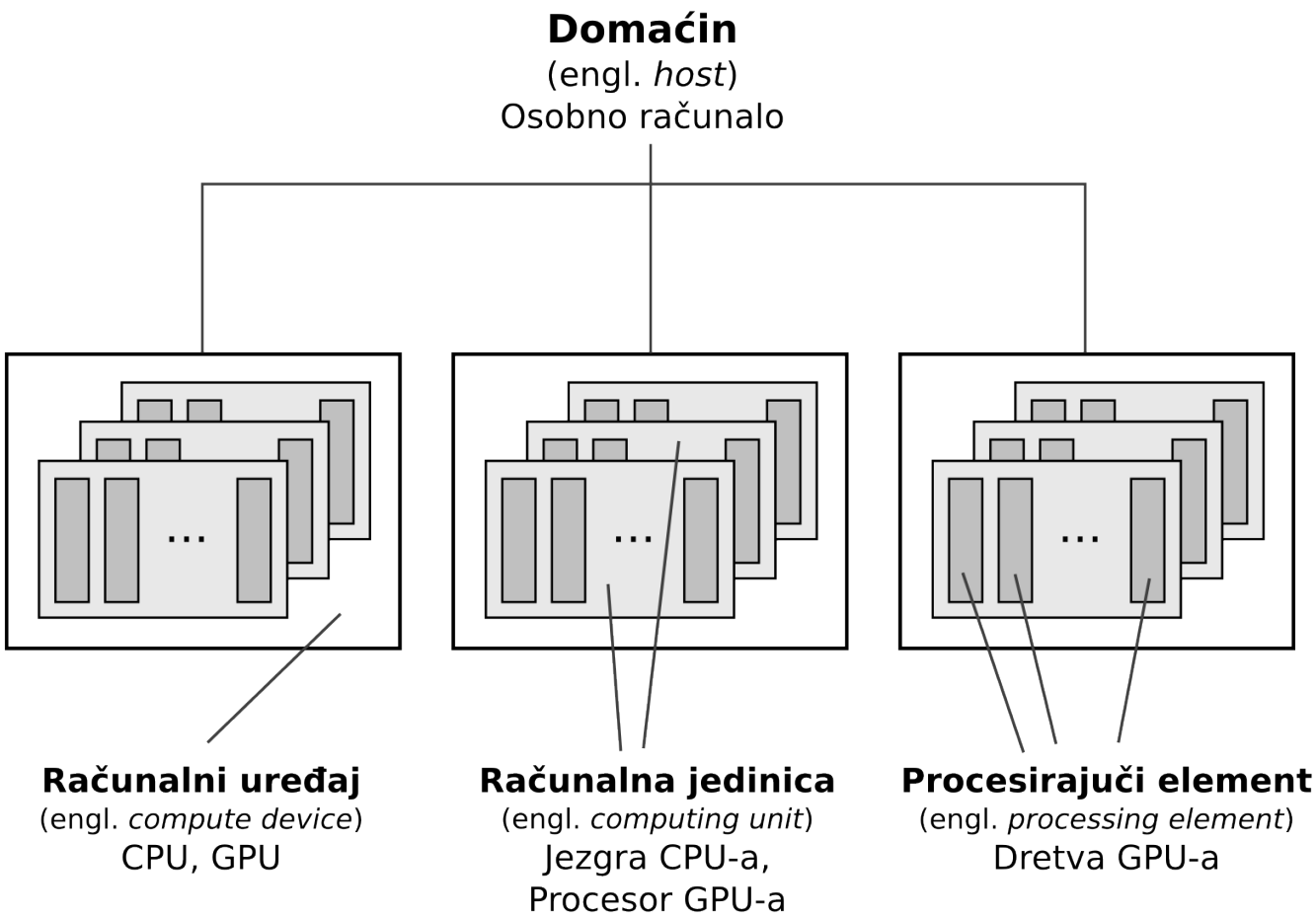
Opis sustava

- projekt započela tvrtka *Apple*
- upravljanje standardom je prepušteno konzorciju *Khronos* (*Apple*, *Nvidia*, *AMD/ATI*, *Intel* i *IBM*)
- otvoren standard, 2009. se pojavile prve implementacije
- namijenjen paralelnom programiranju opće namjene na heterogenim platformama
- cilja na područje od *desktop* aplikacija do računarstva visokih preformansi (engl. *HPC*)
- sustav se sastoji od sučelja za pristup sklopovskim platformama, biblioteka, sustava za raspodjelu poslova i *OpenCL C* programskog jezika

Arhitektura sustava

- platformski model
- izvršni (engl. *execution*) model
- memorijski model
- programski (engl. *programming*) model

Platformski model



Izvršni model

- *OpenCL* jezgre (engl. *kernel*) se izvode na *OpenCL* uređajima
  - napisane su u *OpenCL C* programskom jeziku (proširenje ISO C99)
- *OpenCL* aplikacija se izvodi na domaćinu
  - stvara programske kontekste (engl. *context*), slijedove (engl. *programming queue*s) i memorijske međuspremnike (engl. *buffers*)
  - stvara okolinu i upravlja izvođenjem programskih jezgri

Izvršni model

- *OpenCL* jezgre (engl. *kernel*) se izvode na *OpenCL* uređajima
  - napisane su u *OpenCL C* programskom jeziku (proširenje ISO C99)
- *OpenCL* aplikacija se izvodi na domaćinu
  - stvara programske kontekste (engl. *context*), slijedove (engl. *programming queue*s) i memorijske međuspremnike (engl. *buffers*)
  - stvara okolinu i upravlja izvođenjem programskih jezgri

Izvršni model - indeksni prostor

- za pokretanje programske jezgre je potrebno definirati veličinu problema po dimenzijama (1, 2 ili 3) *(primjer matrice)*
- definira se globalna i lokalna veličina problema
- stvara se indeksni prostor koji odgovara veličini problema po dimezijama
- za svaki element indeksnog prostora se pokreće po jedna radna jednica, tj. programska jezgra
- radne jedinice su grupirane u radne grupe (prema lokalnoj veličini prostora)

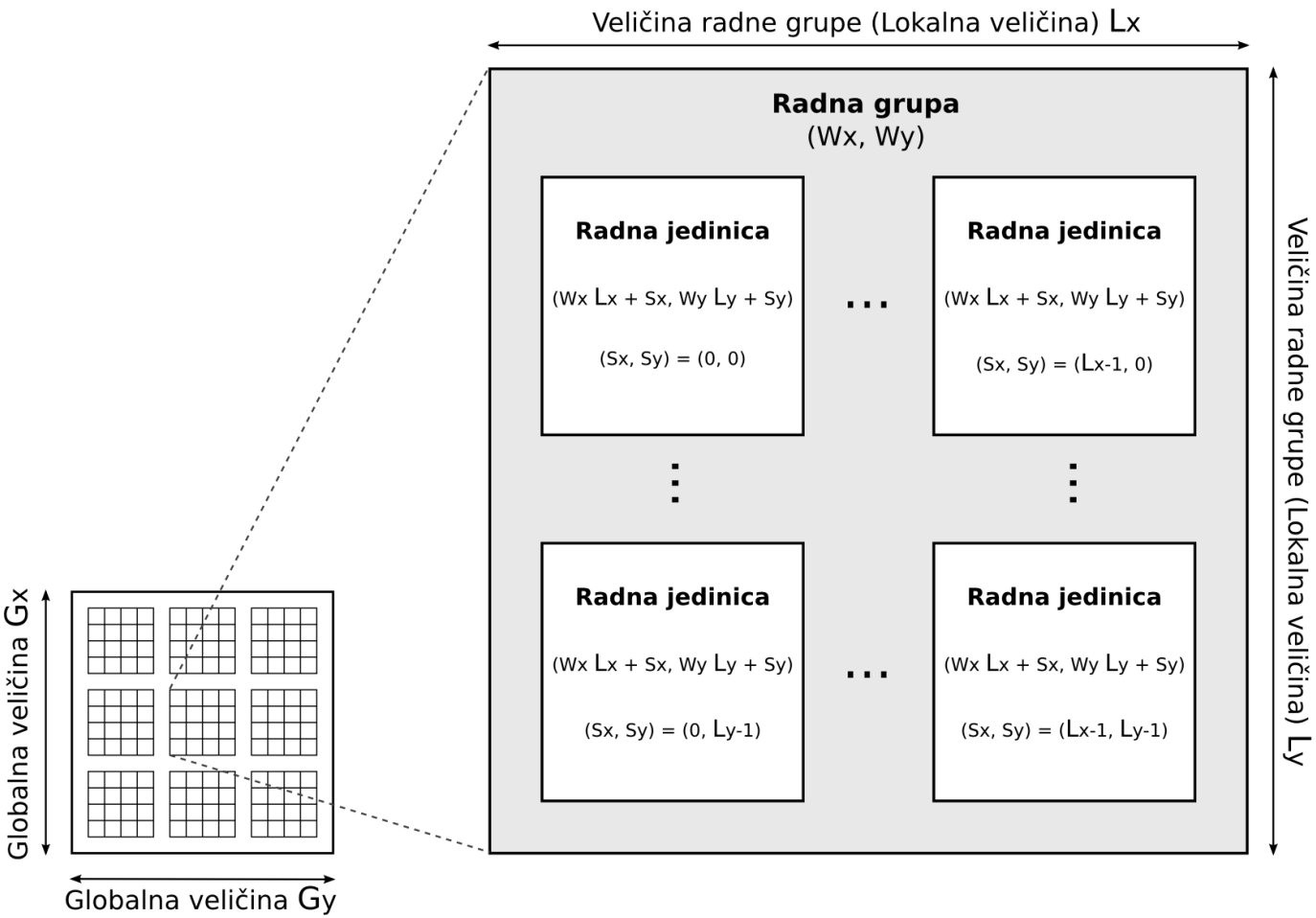
Izvršni model - globalni i lokalni indeksi

- svaka radna jedinica u indeksnom prostoru se može identificirati:
  - preko globalnih indeksa (engl. *global ID*)
  - preko lokalnih indeksa (engl. *local ID*) i indeksa radnih grupa (engl. *work-group ID*)
- navedeni princip omogućava bolju granulaciju u rješavanju problema, te olakšava paralelizaciju algoritama jer se sustav donekle sam pobrine oko raspodjele posla
- svaka radna jedinica izvodi isti programski kod na različitim podacima –> SIMD, tj. SIMT (*Single Instruction, Multiple Thread*)

Izvršni model - globalni i lokalni indeksi

- svaka radna jedinica u indeksnom prostoru se može identificirati:
  - preko globalnih indeksa (engl. *global ID*)
  - preko lokalnih indeksa (engl. *local ID*) i indeksa radnih grupa (engl. *work-group ID*)
- navedeni princip omogućava bolju granulaciju u rješavanju problema, te olakšava paralelizaciju algoritama jer se sustav donekle sam pobrine oko raspodjele posla
- svaka radna jedinica izvodi isti programski kod na različitim podacima –> SIMD, tj. SIMT (*Single Instruction, Multiple Thread*)

Izvršni model - primjer



Memorijski model (1.)

- Globalna memorija
  - sve radne jedinice (neovisno o grupama) mogu čitati/pisati po globalnoj memoriji
  - fizički odgovara radnoj memoriji (engl. *RAM*) na matičnoj ploči ili grafičkoj kartici
- Konstantna memorija
  - područje globalne memorije koje ostaje nepromijenjeno za vrijeme izvođenja programskih jezgri
  - za pisanje/čitanje se brine domaćin

Memorijski model (1.)

- Globalna memorija
  - sve radne jedinice (neovisno o grupama) mogu čitati/pisati po globalnoj memoriji
  - fizički odgovara radnoj memoriji (engl. *RAM*) na matičnoj ploči ili grafičkoj kartici
- Konstantna memorija
  - područje globalne memorije koje ostaje nepromijenjeno za vrijeme izvođenja programskih jezgri
  - za pisanje/čitanje se brine domaćin

Memorijski model (2.)

- Lokalna memorija
  - memorijsko područje namijenjeno radnim jedinicama unutar iste radne grupe
  - u praksi odgovara priručnoj (engl. *cache*) memoriji mikroprocesora
    (ovisno o implementaciji, primjer za CPU na *OS X*-u)
  - značajno manja od globalne memoriji, ali i značajno brži pristup
- Privatna memorija
  - područje namijenjeno svakoj radnoj jedinici pojedinačno
  - svaka radna jedinica ima pristup samo svojoj privatnoj memoriji

Memorijski model (2.)

- Lokalna memorija
  - memorijsko područje namijenjeno radnim jedinicama unutar iste radne grupe
  - u praksi odgovara priručnoj (engl. *cache*) memoriji mikroprocesora
    (ovisno o implementaciji, primjer za CPU na *OS X*-u)
  - značajno manja od globalne memoriji, ali i značajno brži pristup
- Privatna memorija
  - područje namijenjeno svakoj radnoj jedinici pojedinačno
  - svaka radna jedinica ima pristup samo svojoj privatnoj memoriji

Programski model

- podatkovna paralelizacija (do sada opisani princip, češći)
- paralelizacija po poslovima
  - programske jezgre se pokreću neovisno o indeksnom prostoru
  - sinkronizacija radnih jedinica unutar iste grupe (i kod podatkovne)
  - sinkronizacija stavljanjem poslova u programski slijed

Programski model

- podatkovna paralelizacija (do sada opisani princip, češći)
- paralelizacija po poslovima
  - programske jezgre se pokreću neovisno o indeksnom prostoru
  - sinkronizacija radnih jedinica unutar iste grupe (i kod podatkovne)
  - sinkronizacija stavljanjem poslova u programski slijed

Osnovne grupe funkcija

- detektiranje platformi
- dohvaćanje informacija o platformama
- baratanje sa kontekstom (engl. *context*)
- baratanje sa programskim slijedovima (engl. *programming-queues*)
- rad sa memorijskim međuspremnicima (engl. *buffers*)
- stvaranje i izvođenje programskih jezgri (engl. *kernels*)
- rad sa događajima
- povezivanje sa OpenGL-om

OpenCL C (1.)

- proširenje *"C"*-a (ISO C99)
- služi za pisanje programskih jezgri
- predovi ga programski prevodilac sadržan u *driverima*
- izbačene neke mogučnosti
  (pokazivači na funkcije, rekurzije, varijabilne duljine nizova, ...)
- proširenja za paralelizaciju
  (rad sa radnim jedinicama i grupama, vektorima podataka, sinkronizacija, OpenGL, ...)

OpenCL C (2.)

- kvalifikatori za varijable:
  - *__global*, *__local*, *__constant*, *__private*
- funkcije za rad sa radnim jedinicima i grupama:
  - *get_work_dim()*
  - *get_global_size(D)*, *get_global_id(D)*
  - *get_local_size(D)*, *get_local_id(D)*
  - *get_num_groups(D)*, *get_group_id(D)*
- funkcije za sinkronizaciju:
  - *barrier(flags)*, *[read|write]_mem_fence(flags)*

Druge tehnologije

- CUDA (engl. *Compute Unified Device Architecture*)
  - *GP/GPU* na *Nvidia* GPU-ovima –> ograničenje
- ATI Stream - pandan *CUDA*-i
- MPI (engl. *Message Passing Interface*)
  - *manje-više* standard u paralelizaciji danas, više procesa unutar jednog ili više računala (CPU)
- OpenMP
  - paralelizacija algoritama predprocesorskim naredbama prevodiocu
  - koristi višedretvenost, jednostavan, ograničen na jedno računalo
- Cell Broadband Engine
- DirectCompute
- Intel Ct

Druge tehnologije

- CUDA (engl. *Compute Unified Device Architecture*)
  - *GP/GPU* na *Nvidia* GPU-ovima –> ograničenje
- ATI Stream - pandan *CUDA*-i
- MPI (engl. *Message Passing Interface*)
  - *manje-više* standard u paralelizaciji danas, više procesa unutar jednog ili više računala (CPU)
- OpenMP
  - paralelizacija algoritama predprocesorskim naredbama prevodiocu
  - koristi višedretvenost, jednostavan, ograničen na jedno računalo
- Cell Broadband Engine
- DirectCompute
- Intel Ct

Druge tehnologije

- CUDA (engl. *Compute Unified Device Architecture*)
  - *GP/GPU* na *Nvidia* GPU-ovima –> ograničenje
- ATI Stream - pandan *CUDA*-i
- MPI (engl. *Message Passing Interface*)
  - *manje-više* standard u paralelizaciji danas, više procesa unutar jednog ili više računala (CPU)
- OpenMP
  - paralelizacija algoritama predprocesorskim naredbama prevodiocu
  - koristi višedretvenost, jednostavan, ograničen na jedno računalo
- Cell Broadband Engine
- DirectCompute
- Intel Ct

Centralni mikroprocesor

- mikroprocesor opće namjene <–> brzina/jednostavnost
- već par desetaka godina osnova PC-a, x86 arhitektura (*CISC*), kompatibilnost unatrag
- slijedno izvodi naredbe –> pokušaji ubrzavanja rada (osim dizanja radnog takta)
- cjevovodi, superskalarnost, višedretvenost, priručna memorija, *out-of-order-execution*
- dostizanje fizičke granice radnog takta + *mooreov* zakon –> višejezgrenost
- potreba za paralelizacijom (postojećih) algoritama

Grafički mikroprocesor

- pojavili se kasnije od CPU-a zbog potrebe ubrzavanja prikaza 3D grafike
- paralelizirane arhitekture namijenjene izvođenju velikog broja (istovjetnih) matematičkih kalkulacija na velikoj količini podataka –> specijaliziranost
- u odnosu na CPU: manja *"logika"* i priručna memorija, veći dio zauzimaju ALU jedinice
- značajno brže napredovali u odnosu na CPU (takt, tranzistori)
- *GeForce8* (2006.), 128 procesora dretvi, grupe od 8 sa 16kB memorije, svaki može izvršavati po 96 dretvi istovremeno –> 12,288 dretvi (*latency hiding*)
- u zadnjih par godina pojava *GP/GPU*-a

Buduće arhitekture

- imamo višejezgrene CPU-ove opće namjene –> neizbježni zbog kompatibilnosti (*OS*, korisničke aplikacije, ...)
- imamo GPU-ove koji preuzimaju ulogu matematičkog koprocesora
- mogućnost *"hibridnih"* procesora ili CPU + akcelerator (GPU)
- OpenCL je prvi sustav za heterogene platforme!
- budućnost: IBM Cell (1 PPE + 8 SPE jezgri), Intel Larrabee (x86)

Primjer množenja matrica

- jednostavan za realizaciju u svim sustavima: *MPI*, *OpenMP* i *OpenCL*
- korišteno *python* sučelje za *OpenCL*, *pyOpenCL*, programske jezgre u *OpenCL C*-u
- isprobane različite veličine matrica, tipovi podataka, odnos globalne i lokalne *OpenCL* memorije, veličina lokalnog indeksnog prostora

Programska jezgra 1

```
__kernel void matrix_mul(const __global float* A,
                         const __global float* B,
                         __global float* C,
                         uint m, uint n, uint p)
{
// odredivanje retka i stupca u indeksnom prostoru
    uint row = get_global_id(0);
    uint col = get_global_id(1);

    // mnozenje elemenata matrica, C = A * B
    C[row * p + col] = 0;
    for (uint k = 0; k < n; ++k)
        C[row*p+col] += A[row*n+k] * B[k*p+col];
}
```

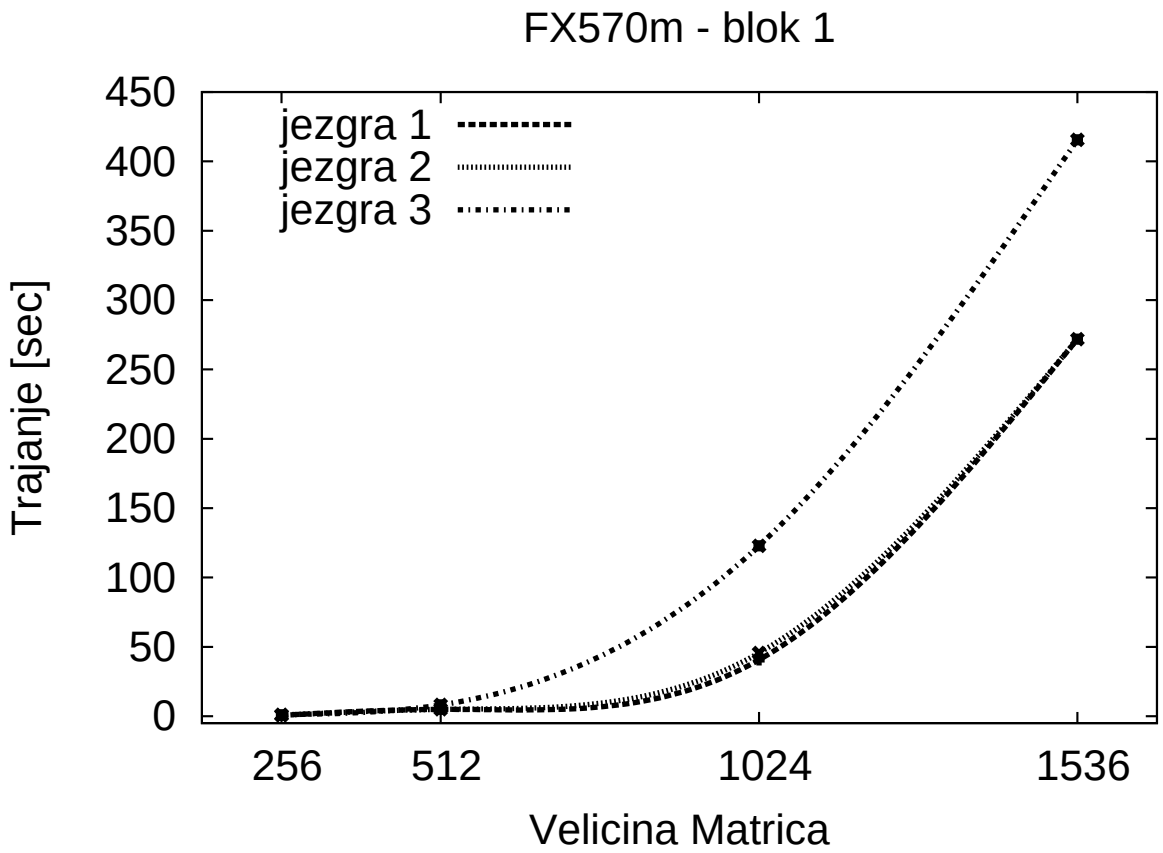
Programska jezgra 2

```
// odredivanje velicine bloka
uint blockSize = get_local_size(0);

// odredivanje indeksa radne grupe
uint row = get_group_id(0);
uint col = get_group_id(1);

// odredivanje lokalnih indeksa u grupi
uint x = get_local_id(0);
uint y = get_local_id(1);

// odredivanje indeksa u matrici
uint pos_x = (row * blockSize + x) * p;
uint pos_y = col * blockSize + y;
```

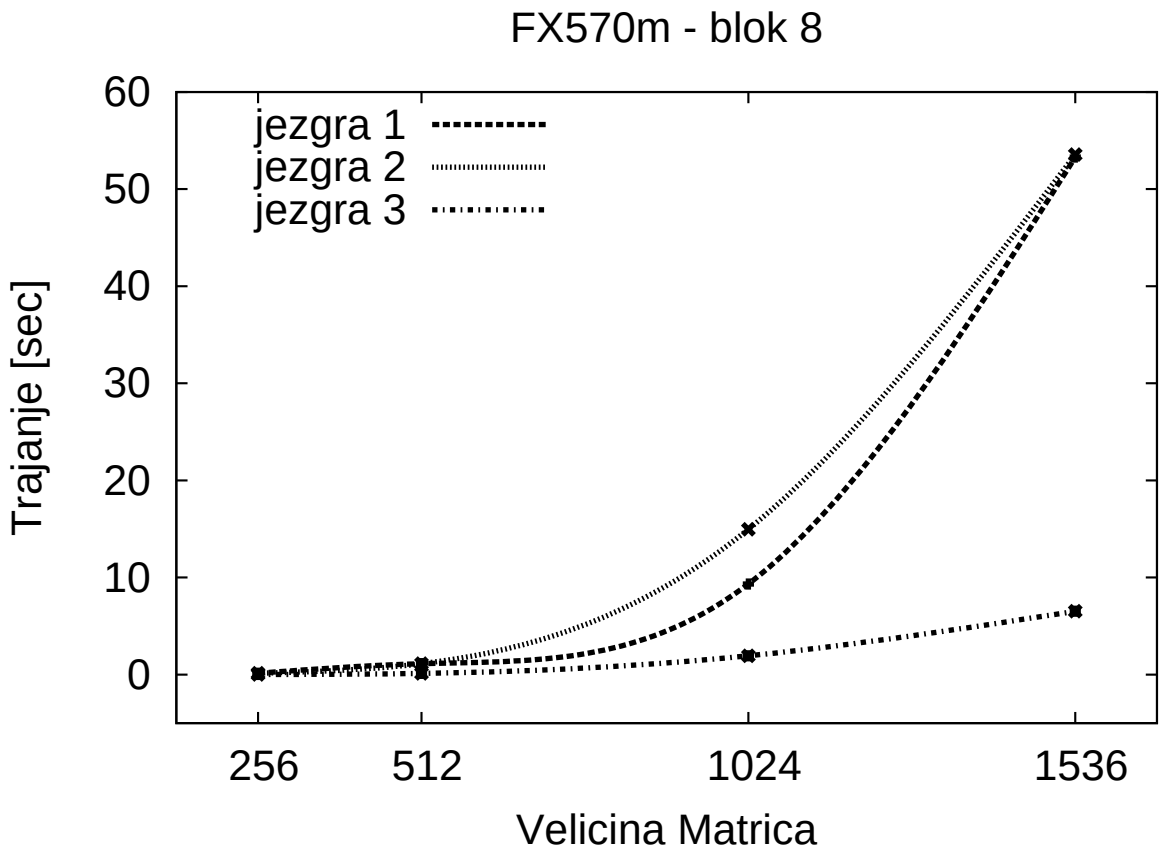
Programska jezgra 3

```
// iteracija kroz sve elemente ...

// kopiranje (pod)matrica A i B u lokalnoj mem.
// svaka radna jedinica kopira po jedan element
subA[x * blockSize + y] = A[blockA + x * n + y];
subB[x * blockSize + y] = B[blockB + x * p + y];

// cekanje da sve radne jednice zavrse kopiranje
barrier(CLK_LOCAL_MEM_FENCE);

// mnozenje elemenata podmatrica A i B
for(int k = 0; k < blockSize; ++k)
 subC += subA[x*blockSize+k] * subB[k*blockSize+y];
```

Postignuti rezulati - veličina bloka 1



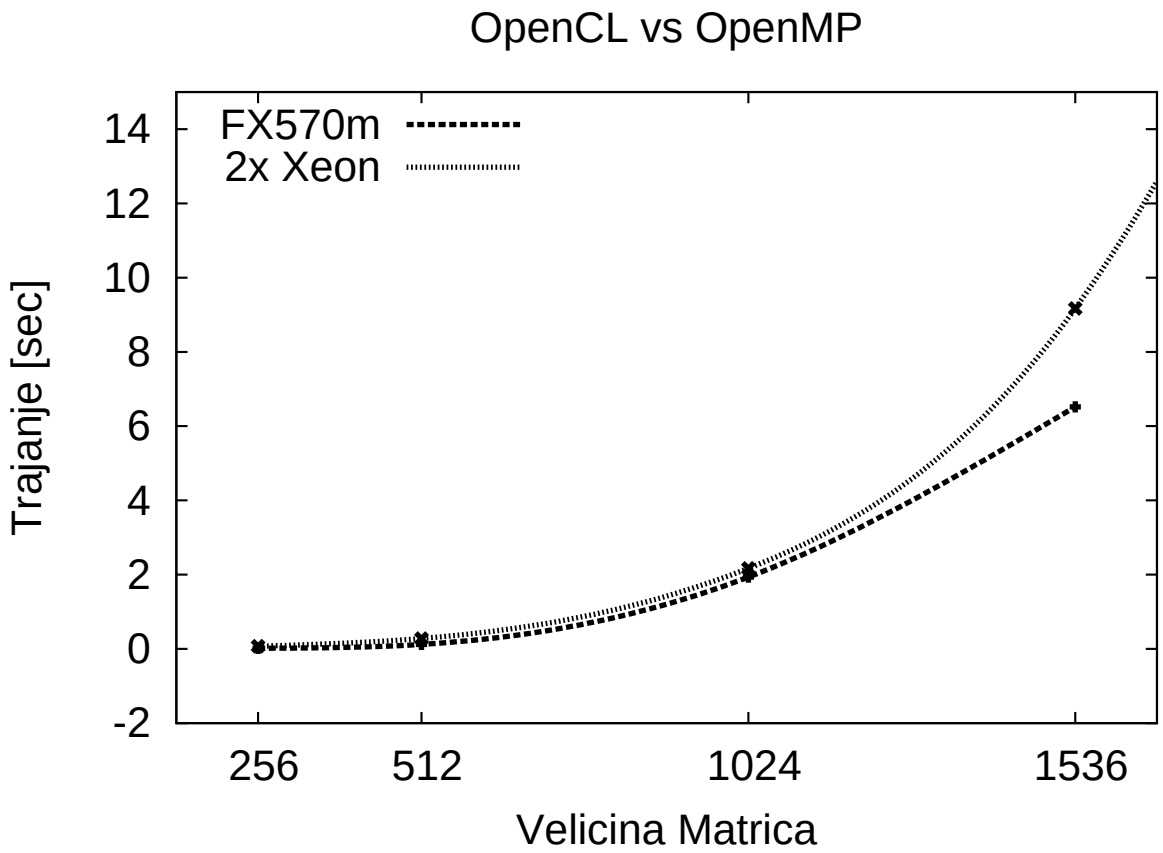
Postignuti rezultati - veličina bloka 8



N-body simulation

- velika količina čestica se raspodijeli po prostoru, te se kroz niz iteracija promatra njihovo međudjelovanje
- uobičajeni primjeri su fizikalne pojave poput gibanja tekućine/plinova –> procesorski iznimno zahtjevni poslovi
- u navedenom primjeru se uzima trenutna brzina i gravitacijske sile, svakom iteracijom se računa novi položaj svih čestica
- korištenje vektorskih tipova podataka (brzina i pozicija)
- ubrzanje korištenjem lokalnog indeksnog prostora i memorije
- primjer sam po sebi jednostavan za paralelizaciju

Množenje matrica: OpenCL vs. OpenMP



Osvrt na rezultate: CPU *vs* GPU

- hardverske platforme:
  - *OpenCL*: *Nvidia FX570m*, 475MHz, 256MB, 2007.
    programska jezgra 3, veličina bloka 8
  - *OpenMP*: 2x *Xeon E5504* (4 jezgre), 2GHz, 8GB, 2009.
- GPU je postigao nešto bolje rezultate!?
- pa zapravo očekivano zbog odnosa opće namjene CPU-a spram specijaliziranog GPU-a
- postavlja se pitanje koje platforme je danas optimalno koristiti (naravno ovisno o tipu problema)
- *hint! Tesla vs CPU cluster*

Osvrt na rezultate: CPU *vs* GPU

- hardverske platforme:
  - *OpenCL*: *Nvidia FX570m*, 475MHz, 256MB, 2007.
    programska jezgra 3, veličina bloka 8
  - *OpenMP*: 2x *Xeon E5504* (4 jezgre), 2GHz, 8GB, 2009.
- GPU je postigao nešto bolje rezultate!?
- pa zapravo očekivano zbog odnosa opće namjene CPU-a spram specijaliziranog GPU-a
- postavlja se pitanje koje platforme je danas optimalno koristiti (naravno ovisno o tipu problema)
- *hint! Tesla vs CPU cluster*

Zaključak

- hardverske platforme
  - neophodnost (x86) CPU-a zbog OS-a, korisničkih aplikacija
  - kombiniranje sa ugradnjom 1 ili više GPU-a kao akceleratora
- OpenCL
  - prvi sustav za paralelizaciju namijenjen heterogenim hardverskim platformama
  - cilja na područje od korisničkih aplikacija (obrada slike, videa, zvuka, ...) pa sve do *HPC-a*
  - pojednostavljuje paralelizaciju algoritama, te omogučava bolje iskorištavanje hardvera koji danas nalazimo u osobnim računalima
  - nedostatak mrežnog sloja, još uvijek upitna podrška

Zaključak

- hardverske platforme
  - neophodnost (x86) CPU-a zbog OS-a, korisničkih aplikacija
  - kombiniranje sa ugradnjom 1 ili više GPU-a kao akceleratora
- OpenCL
  - prvi sustav za paralelizaciju namijenjen heterogenim hardverskim platformama
  - cilja na područje od korisničkih aplikacija (obrada slike, videa, zvuka, ...) pa sve do *HPC-a*
  - pojednostavljuje paralelizaciju algoritama, te omogučava bolje iskorištavanje hardvera koji danas nalazimo u osobnim računalima
  - nedostatak mrežnog sloja, još uvijek upitna podrška