SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

OpenCL i njegova primjena

Stjepan Hadjić

Voditelj: *Siniša Šegvić*

Zagreb, lipanj, 2010

Sadržaj

1. Uvod

Kako bi iskoristili sav potencijal procesora pri računanju velikog broja podataka, paralelizam je prihvaćen kao način za ubrzavanje izvođenja zadataka. Današnji procesori opće namjene su dostigli tehničke granice iznad koje je nemoguće efikasno povećati brzinu takta te je počela praksa dodavanja dodatnih jezgara. Grafički procesori su također evoluirali iz jednostavnih naprava koje prikazuju teksture u procesore na kojima je moguće paralelno programirati. Kako današnji računalni sustavi često uključuju paralelne procesore opće namjene i grafičke procesore, bilo je ključno omogućiti programerima način na koji mogu iskoristiti svu snagu heterogenih procesnih platformi.

Programiranje na procesorima opće namjene je različito od programiranja na grafičkim procesorima pa je programerima bilo teško iskoristiti svu snagu heterogenih procesora iz jedne, višeplatformske baze kodova. Stoga je Apple dao prijedlog da se stvori standard koji bi omogućio programiranje neovisno o tipu procesora te je grupa Khronos Group 18.11.2008. izdala specifikaciju za OpenCL prema [1].

2. OpenCL

OpenCL (Open Computing Language) je standard otvorenog koda za paralelno programiranje na procesorima opće namjene, grafičkim procesorima i drugim procesorima (CELL/BE procesori) te omogućava programerima efikasan pristup heterogenim procesnim platformama.

OpenCL sadržava programsko sučelje (engl. API - Application Programming Interface) za paralelno programiranje na heterogenim procesorima. Isto tako sadržava programski jezik za programiranje na različitim platformama.

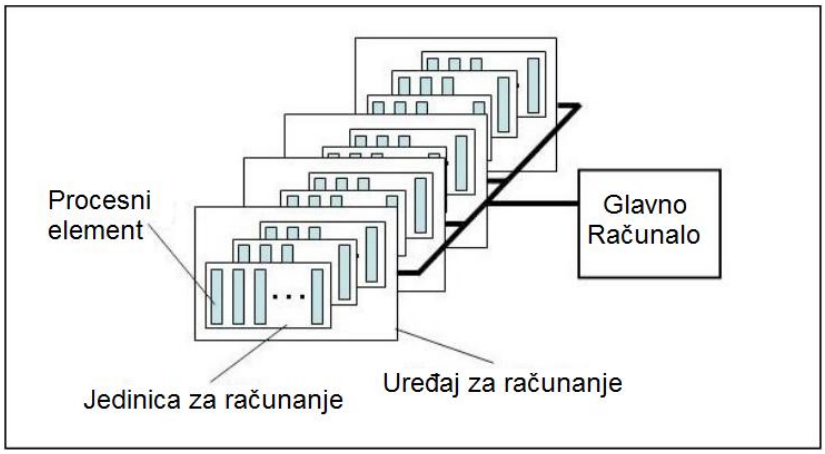
OpenCL standard:

- Zasniva se na programskom jeziku C sa proširenjima za paralelizam
- Podržava podatkovni paralelizam i paralelizam zasnovan na zadacima
- Podržava stvaranje aplikacija za ručne i ugrađene uređaje
- Efikasno surađuje sa OpenGL, OpenGL ES i ostalim grafičkim sučeljima

OpenCL je više od programskog jezika. OpenCL je okvir (engl. framework) za paralelno programiranje i uključuje programski jezik, programsko sučelje, biblioteke i sustav za pokretanje programa. Koristeći OpenCL, programer može napisati program koji se pokreće na grafičkom procesoru bez potrebe preslikavanja algoritma u 3D grafička programska sučelja kao što su OpenGL ili DirectX.

3. Arhitektura OpenCL standarda

Model platforme (engl. platform model) OpenCL-a je definiran na slici 1. Model se sastoji od jednog ili više OpenCL uređaja (engl. device) koji su spojeni na glavno računalo (engl. host). OpenCL uređaj se sastoji od jedane ili više jedinica za izračunavanje (engl. compute unit (CU)) koje su dalje podijeljene na jednu ili više procesnih elemenata (engl. processing element (PE)). OpenCL aplikacija se pokreće na glavnom računalu te ona šalje zapovijedi (engl. comands) procesnim elementima unutar uređaja. Zapovijedi se mogu koristiti za prebacivanje podataka između memorije glavnog računala i memorije uređaja, izvšavanje jezgre, ili za sinkronizaciju.



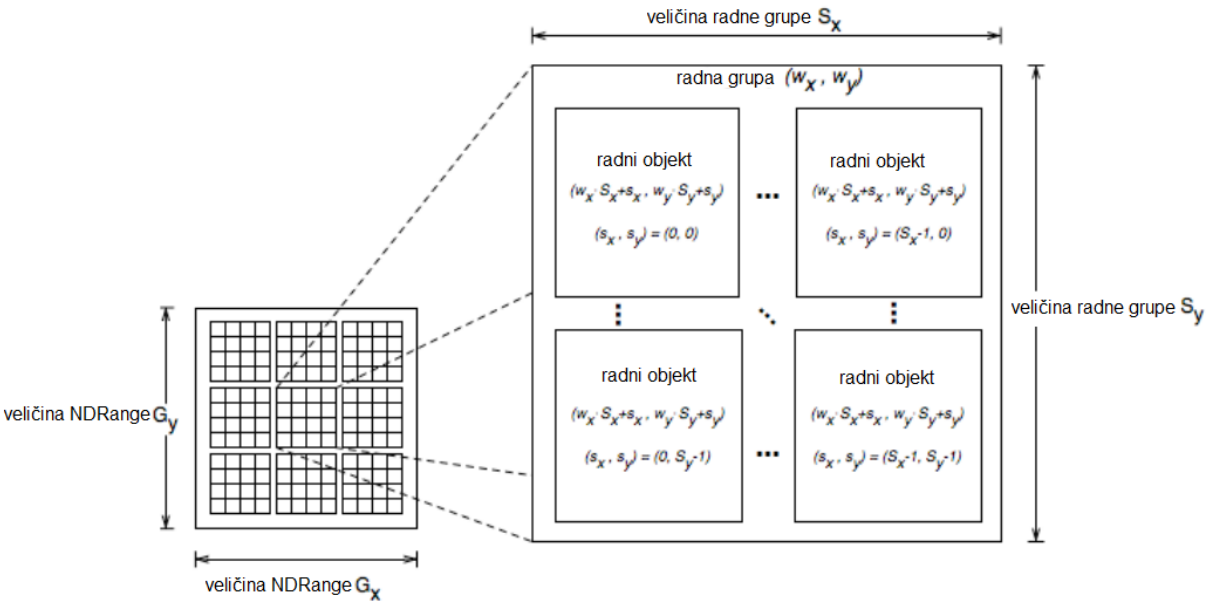
Slika 1. Model platforme (uređaji za računanje koji se sastoje od jedinica za računanje, koji se sastoje od procesnih elemenata, su spojeni na glavno računalo)

OpenCL program sastoji se od dva dijela: jezgre (engl. kernels) koje se pokreću na jednom ili više OpenCL uređaja i glavni program (engl. host program) koji se pokreće na glavnom računalu. Glavni program definira kontekst za jezgre i upravlja njihovim izvršavanjima.

Suština OpenCL modela izvršavanja (engl. execution model) je definirana u tome kako se jezgre izvršavaju. Za svako računanje stvara se posebna kopija jezgre, npr. za množenje elemenata dva vektora veličine 10, stvorit će se 10 kopija jezgre. Te se jezgre nazivaju radni objekti (engl. work-item) te se oni spremaju u prostor

indeksa (engl. index space). U OpenCL standardu takav prostor se naziva NDRange (N-Dimensional Range). Radni objekti su indentificirani mjestom gdje se nalaze u prostoru. Na slici 2. je prikazan jedan primjer prostora indeksa

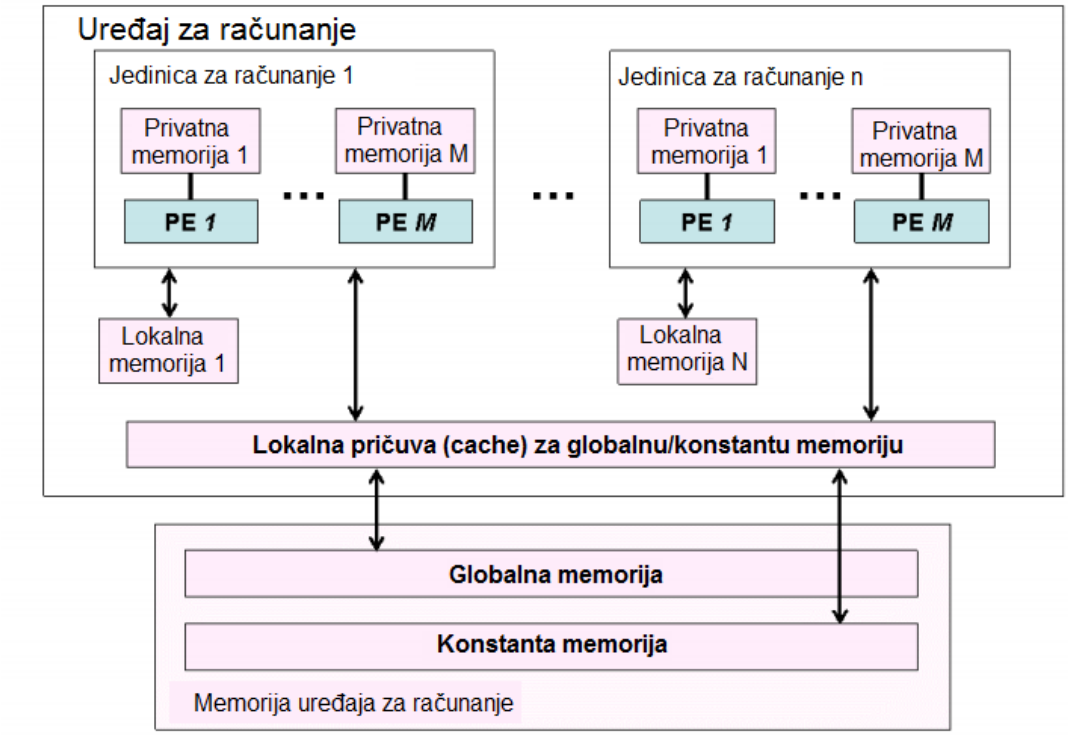
Radni objekti organizirani su u radne grupe (engl. work-group). Radne grupe daju krupnozrnatu dekompoziciju prostora indeksa. Svaki se radni objekt unutar radne grupe izvršava istodobno na procesnim elementima unutar jedne jedinice za izračunavanje (CU).



Slika 2. Primjer NDRange prostora indeksa

Glavno računalo definira kontekst za izvršavanje jezgri te sadrži informacije o uređajima, jezgrama, programskim objektima i memorijskim objektima. Isto tako glavno računalo stvara strukturu podataka koja se zove red zapovijedi (engl. command-queue) za koordinaciju izvršavanja jezgri na uređajima.

Radni objekti kod izvršavanja jezgri imaju pristup četirima različitim memorijskim prostorima: globalna, konstantna, lokalna i privatna memorija. Slika 3. prikazuje kako se navedene memorije odnose prema platformskom modelu.



Slika 3. Konceptualna arhitektura OpenCL uređaja (odnos između konstantne, globalne, lokalne i privatne memorije i modela platforme)

4. Pisanje programa u OpenCL standardu

U OpenCL standardu program mora biti pisan na određen način. Prvi dio program se sastoji od ponalaska svih mogućih platformi (trenutačno NVIDIA CUDA ili ATI STREAM) za izvršavanje OpenCL programa, pronalaska uređaja vezanih za platforme (grafički procesori i procesori opće namjene) te se na kraju stvara kontekst za pronađene uređaje. U drugom djelu se stvara red zapovijedi i memorijski objekti, stvaraju se i prevode jezgre. Postavljaju se argumenti jezgre te se jezgre šalju u prostor indeksa kako bi se izvršile.

Primjer programa:

```
// Pronalaženje platforme (NIDIA CUDA ili ATI Stream)
clGetPlatformIDs(1, &platform_id, &ret_num_platforms);
// Pronalaženje uređaja (GPU, CPU)
clGetDeviceIDs(platform_id,CL_DEVICE_TYPE_DEFAULT, 1,&device_id,
&ret_num_devices);
// Stvaranje konteksta
context = clCreateContext( NULL, 1, &device_id, NULL, NULL, &ret);
// Stvaranje reda zapovijedi
command_queue = clCreateCommandQueue(context, device_id, 0, &ret);
// Stvaranje openCL memorijskog spremnika
memobj = clCreateBuffer(context, CL_MEM_READ_WRITE, MEM_SIZE *
      sizeof(float), NULL, &ret);
// Prebavivanje podataka u openCL memorijski spremnik
clEnqueueWriteBuffer(command_queue, memobj, CL_TRUE, 0, MEM_SIZE *
      sizeof(float), mem, 0, NULL, NULL);
// Stvaranje OpenCL programa iz tekstualnog izvora
program = clCreateProgramWithSource(context, 1, (const char **)
      &source_str, (const size_t *) &source_size, &ret);
// Stvaranje programa (build)
clBuildProgram(program, 1, &device_id, NULL, NULL, NULL);
// Stvaranje OpenCL jezgre
kernel = clCreateKernel(program, "vecAdd", &ret);
// Postavljanje argumenata openCL jezgre
clSetKernelArg(kernel, 0, sizeof(cl_mem), (void *)&memobj);
```

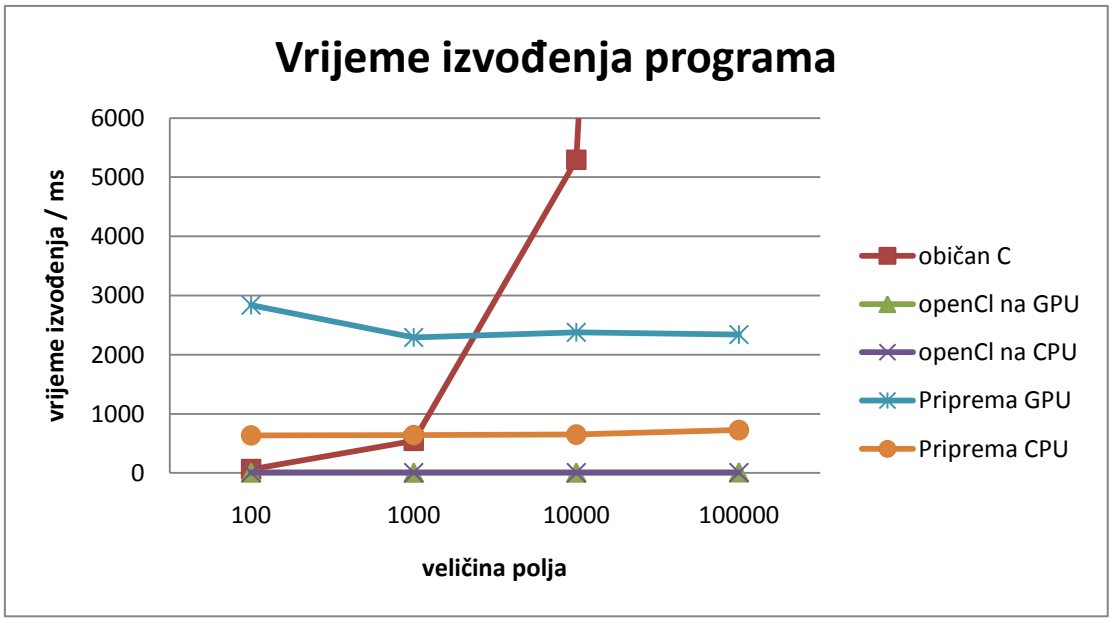
```
// Postavljanje veličine radnih grupa
size_t global_work_size[3] = {MEM_SIZE, 0, 0};
size_t local_work_size[3]  = {512, 0, 0};
// Pokretanje openCL jezgre
clEnqueueNDRangeKernel(command_queue, kernel, 1, NULL,
      global_work_size, local_work_size, 0, NULL, NULL);
// Prebacivanje rezultata iz openCL memorijskog spreminka
clEnqueueReadBuffer(command_queue, memobj, CL_TRUE, 0, MEM_SIZE *
      sizeof(float), mem, 0, NULL, NULL);
```

Primjer jezgre:

```
__kernel void vecMul(__global float* a)
{
    int gid = get_global_id(0);
    a[gid] = a[gid] * a[gid];
}
```

4.1. Usporedba openCL programa sa programom pisanim u čistom C-u

Testovi su vršeni na dvije različite platforme, na grafičkoj kartici NVIDIA 8800 GTS te na AMD X2 6400+ procesoru.



Iz rezultata vidimo kako se vrijeme izvođenja programa eksponencijalno povećava za program pisan u čistom C-u (preko jedne minute), dok je vrijeme izvođenja programa na openCL platformi ujednačeno (oko 7 ms). Problem kod programa pisanog u openCL-u jest vrijeme potrebno za pripremu podataka prije izvođenja. Uočavamo, iako je vrijeme pripreme ujednačeno i neovisno o veličini podataka, ipak je nekoliko redova veličine veće od vremena izvršavanja računanja.

5. Zaključak

Potreba za aplikacijama koje mogu brzo računati sa milijunima podataka, svakim danom je sve veća. Zbog postojanja puno procesora sa različitim arhitekturama, teško je predvidjeti na kakvim će se svim sustavima pokretati takva aplikacija. Taj problem se pokušava riješiti stvaranjem openCL standarda. OpenCL standard omogućuje lakše paralelno programiranje na heterogenim procesorima.

Iz rezultata usporedbi možemo vidjeti kako je vrijeme izvršavanja openCL programa nekoliko redova veličine manje nego programa pisanog u čistom C-u. No zbog relativno dugog vremena potrebnog za pripremu programa za izvršavanje na openCL platformama, openCL bi se trebao koristiti samo za paralelni dio programa. Slijedni dio programa bi se trebao programirati u čistom C-u kako bi se najviše optimizirao program.

6. Literatura

[1] OpenCL 1.0 Specification, Khronos Group, listopad 2009.
http://www.khronos.org/registry/cl/specs/opencl-1.0.pdf

[2] OpenCL 1.0 Reference page, Khronos Group, veljača 2010.
http://www.khronos.org/registry/cl/sdk/1.0/docs/man/xhtml/

[3] Introductory Tutorial to OpenCL, Benedict R. Gaster, veljača 2010.
http://developer.amd.com/gpu/ATIStreamSDK/pages/TutorialOpenCL.aspx

[4] OpenCL Tutorials, MacReshearch, veljača 2010.
http://www.macresearch.org/opencl

[5] OpenCL Programming Guide, NVIDIA, veljača 2010.
http://developer.download.nvidia.com/compute/cuda/3_0/toolkit/docs/NVIDIA_OpenCL_ProgrammingGuide.pdf

[6] OpenCL, Wikipedia, the free encyclopedia, veljača 2010.
http://en.wikipedia.org/wiki/OpenCL

7. Sažetak / Abstract

OpenCL (Open Computing Language) je standard otvorenog koda za paralelno programiranje na procesorima opće namjene, grafičkim procesorima i drugim procesorima (CELL/BE procesori) te omogućava programerima efikasan pristup heterogenim procesnim platformama.

OpenCL je okvir za paralelno programiranje i uključuje programski jezik, programsko sučelje, biblioteke i sustav za pokretanje programa.

Paralelni algoritmi pisani u openCL mogu biti i do 100 puta brži nego slijedno pisani algoritmi.

Ključne riječi: OpenCL, programiranje na grafičkim procesorima (GPGPU), paralelno programiranje, NVIDIA CUDA, ATI STREAM

OpenCL

OpenCL (Open Computing Language) is an open standard for general purpose programming across CPUs, GPUs and other processors, giving software developers efficient access to these heterogeneous processing platforms.

OpenCL is a framework for parallel programming and includes a language, API, libraries and a runtime system.

Parallel algorithms written in openCL can be up to 100 times faster than sequential algorithms.

Key words: OpenCL, General purpose computation of Graphics Processing Units (GPGPU), parallel programming, NVIDIA CUDA, ATI STREAM