УДК 004.272.34

ЗАМОНАВИЙ ГРАФИК ПРОЦЕССОРЛАР АРХИТЕКТУРАСИ ВА ТЕХНОЛОГИЯЛАРИ
(Обзор)

Хужаяров И.Ш.

Мақолада замонавий график процессорлар архитектураси таҳлили ва уларда дастур бажарилишининг ўзига хусусиятлари, график процессорларда қўлланиладиган дастурлаш платформаларининг таҳлили, дастур кодини оптимизациялаш усуллари келтирилади.

Таянч иборалар: график процессор, тасвирларни қайта ишлаш, варп**,** CUDA, OpenCL, DirectX, хотира иерархияси,параллелизм, сигнал, оптимизациялаш, PCI-Express.

В данной статье анализируется архитектура современных графических процессоров и основные особенности выполнения программ в GPU, анализ платформ программирования используемых в графических процессорах, а также методы оптимизации программного кода.

Ключевые слова: графический процессор, обработка изображение, варп**,** CUDA, OpenCL, DirectX, иерархия памяьти, параллелизм, сигнал, оптимизация, PCI-Express.

The widespread need to implement high-quality, interactive 3D graphics has led in recent years to the significant technological development of graphics processors (GPU)s, which are an integral part of any personal computer. Graphic the processor has acquired the quality of a high-performance device based on the use of parallel technologies. At the same time, the graphical processor provides the ability to program the processing of initial data at the level of direct commands of the graphic processor. High-quality 3D graphics, which requires parallel processing of resource-intensive data from the GPU, led to the creation of a special, in a certain sense, unique GPU architecture. The modern graphics processor is designed so that the majority of its transistors in the processor chip are used for data processing, to the detriment of caching and control of the progress of the program.

This article analyzes the architecture of modern graphics processors (GPUs) and the main particular qualities of executing programs in the GPU. The possibility of using the GPU for solving resource-intensive numerical problems such as the basic algorithms of digital signal processing and images is considered. The problems and limitations that arise when computing from the central processing unit to the graphic processor are transferred, the analysis of the programming platforms used in the GPU is analyzed, and variants of optimizations of

applications using the graphics processor are also offered.

Key words: graphics processor, image processing, warp, CUDA, OpenCL, DirectX, memory hierarchy, parallelism, signal, optimization, PCI-Express.

I. КИРИШ

Яқин вақтларгача умумий мақсадга йўналтирилган масалаларни дастурлаш учун фақат марказий процессорлар билан жиҳозланган компьютерлардан фойдаланиб келинди. Ҳозирги кунга келиб юқори унумдорликга эга компьютерларда марказий процессордан ташқари бошқа ҳисоблаш элементларини қўшилганини кўриш мумкин. Шулардан энг кўп тарқалгани график процессорлар(GPU) ҳисобланади. Охирги йилларда юқори сифатга эга интерактив уч ўлчовли графикаларни тасвирлаш ва улар устида бажариладиган амалларга нисбатан талабнинг ортиши шахсий компьютерларнинг ажралмас қисмига айланган GPU ларнинг мавжуд технологияларини ривожлантиришга олиб келди. График процессор параллел технологияларни қўллашга асосланган юқори унумдорликни таъминловчи қурилмадир. Шу билан бирга, замонавий GPU кирувчи маълумотларни қайта ишлаш дастурларини GPU нинг бевосита бўйруқлар даражасида амалга ошириш имконини беради. GPU ларнинг бевосита марказий процессор(CPU) билан биргаликда ишлаши гетероген ҳисоблаш моделини тақдим этади. Бу гетероген ҳисоблаш модели дастурнинг кетма-кет бажариладиган қисмини CPU да параллел бажариладиган қисмини эса GPU да амалга оширади. Чунки график процессор CPU га нисбатан катта ҳисоблашларни тезкор бажариш имкониятини тақдим этувчи кўп сондаги электрон компонентлардан ташкил топган[1,2].

Кўп ҳисоблаш ресурсини талаб қиладиган юқори сифатга эга уч ўлчовли графикалар GPU дан параллел ҳисоблашни амалга ошириш талабини оширди ва бу эса GPU нинг махсус архитектурасини яратишга туртки бўлди. Замонавий GPU ларнинг архитектурасидаги процессор чипининг асосий қисмини маълумотларни параллел қайта ишлаш учун мўлжалланган ядролардан ташкил топган. Шунинг учун ҳам GPU маълумотлар бўйича параллелизм(data parallel) ни қўллаб қувватлайди. Маълумотлар бўйича параллелизм бир қанча маълумотлар устида битта бўйруқни бажаришни амалга оширади. GPU ларни технологик модернизация қилиш натижасида уларнинг ҳисоблаш қуввати CPU га нисбатан сезиларли даражада ошиб кетишига сабаб бўлди. Мисол учун ҳозирги кунда мавжуд ўрта миёна Kepler архитектурасига тегишли NVIDIA GeForce GTX 680 видеокарталари ўрта ҳисобда 1 Tflops дан юқори максимал унумдорликни таъминловчи секундига трилионлаб қўзғалувчи вергулли сонлар устида амаллар бажара оладиган 2880 та ҳисоблаш ядросига эга. Ҳозирги кундаги замонавий CPU ларнинг энг охирги 7 авлоди саналган Intel Core i7-975 XE 3,33 ГГц микропроцессорларнинг максимал унумдолиги эса 53.28 Gflops га

етади. Бундан кўриниб турибдики, GPU тақдим этадиган унумдолик CPU га нисбатан 20 баравар юқоридир[12]. GPU да тизимли дастурлашни амалга ошириш имкониятиниг пайдо бўлиши уни фақат график иловаларни қайта ишлашда эмас, балки кенг миқёсдаги бошқа масалалар синфини ҳисоблашда фойдаланиш имкониятини берди. Бу эса GPGPU(General Purpose computations on Graphics Processing Unit) номини олди. Ҳозирги кунга келиб GPGPU ахборот технологиялари соҳасининг қўйидаги йўналишларида кенг қўлланилиб келинмоқда: видео маълумотларни қайта ишлаш[3, 19, 20, 22], сигналларга рақамли ишлов бериш[6,7,18], физик ва химик жараёнларни математик моделлаштириш[4], тиббиётдаги визуализация[5] ва бошқа соҳаларни ҳам келтириш мумкин. Бундан ташқари график процессорларни мураккаб нейрон тармоқларини қуришда тадбиқ қилинмоқда [8] адабиётда келтирилган ишда GPU дан фойдаланган ҳолда нейрон тармоқни ўқитишга кетган вақт 150 мартага қисқарганлиги кўрсатиб ўтилган. Яна бир GPU дан фойдаланишнинг истиқболли йўналишларидан бири бу маълумотларни базасини бошқариш тизимларидир[9,10, 11]. Бундай ҳолда GPU маълумотлар базасига янги ёзувни қўшиш, маълумотларни саралаш, сўровларни қайта ишлаш, маълумотларни сиқиш ва жўнатиш каби ёрдами масалаларни бажаради.

Шундай экан GPU архитектурасининг имкониятларини ўрганиш катта ҳисоблаш ресурсларини талаб қиладиган мультимедиа иловалари қайта ишлаш унумдорлигини оширади. Мультимедиа иловалари одатда реал вақт тизимларида қайта ишланади ва уларнинг асосий ташкил этувчилари сифатида нутқ, аудио-видео сигналлар, анимациялар ва тасвирлар қаралади.

Ушбу мақолада график процессорлар архитектураси таҳлили ва уларда дастур бажарилишининг ўзига хусусиятлари, дастурлаш платформаларининг таҳлили, дастур кодини оптимизациялаш усуллари келтирилади.

II. АСОСИЙ ҚИСМ

Сигналларга ва тасвирларга рақамли ишлов бериш замонавий ахборот технологиялари, телекоммуникация ва рақамли алоқа тизимлари, IP-телефония ва видеоконференция, рақамли телевидения соҳаларидаги масалаларни ечишда долзарб ҳисобланади. Юқорида келтирилган соҳа масалаларини аксарияти реал вақт режимида ишлашини инобатга олган ҳолда ушбу тизимларда маълумотларни қайта ишлаш ва ўзатиш тезлигига юқори талабларни қўймоқда. Бундай тизимларнинг аппарат қурилмалари ҳажм жиҳатидан катта бўлмаслиги лозим, аксинча энергия самарадорли, ихчам, тезкор ҳатто мобил ишлов бериш воситаларидан ташкил топган бўлиши керак. Бундай талабларни ушбу тизимларда самарали бажариш учун маълумотларни қайта ишлаш ва уларни узатиш алгоритмларини тезкор

амалга оширувчи кўпядроли процессорларни тадбиқ қилиш мақсадга мувофиқдир.

Сигналларга ва тасвирларга рақамли ишлов бериш масалалари асосан вақт соҳасида ва спектрал соҳада амалга оширилади. Ҳар иккала ишлов бериш соҳасининг алгоритмлари ва усуллари мавжуд бўлиб, улар реал вақт тизимида сигналлар ва тасвирлар устида турли хилдаги фильтрлаш , сигнал сифатини яхшилаш, коррелацион ишлов бериш, сигналларнинг энергияси ва қувватини ҳисоблаш, амплитудали ва фазоли характеристикалари аниқлаш, сиқиш, кучли шовқин таъсиридан йўқотилган сигнал қисмларини қайта тиклаш, сигнал параметрларни баҳолаш, таниш ва шунга ўхшаш амалларни бажаради.

Спектрал усуллар асосида сигналларга рақамли ишлов бериш жадал равишда ривожланди, айниқса Фурье таҳлил. Сигналларни спектрга ёйиш эвазига эришиладиган самарадорликдан ташқари Фурье базис ва ундан ҳосил бўлган базислар (Адамар, Хаара, слэнт-алмаштириш, вейвлет алмаштириш) базисларнинг алгоритмлари аввалдан векторлар ва матрицаларни кўпайтириш процедурасини қўллаган. Шунинг учун ҳам бу алгоритмларни параллел қайта ишлаш муваффақиятли амалга оширилади.

Спектрал ишлов бериш соҳаси сигналлар устида бажариладиган мураккаб алгоритмлар синфини кенгайтиради ва уларда қўлланиладиган сонли усуллар, бир вақтнинг ўзида бир нечта амалларни(фильтрлаш, сиқиш ва ҳк) бажариш имкониятини тақдим этади. Сигналларни спектрал соҳада қайта ишлаш катта ҳажмдаги ҳисоблашни талаб қилади, чунки кирувчи маълумотлар ҳажмининг ортиши билан ҳисоблашларнинг бажарилиш вақти ҳам чизиқсиз ўсиб боради. Спектрал қайта ишлаш алгоритмларини ҳисоблаш унумдорлигини ошириш анча вақтлардан буён сонли усулларнинг ҳисоблаш операцияларининг умумий сонини камайтириш ҳисобига амалга оширилиб келинган. Мисол сифатида кўп сондаги тезкор Фурье-базислар вариантлари ва вейвлет-алмаштиришни мисол келтириш мумкин. Аммо кўп ядроли процессорлар алгоритмларни оптимизациялаш жараёнига янги услубларни киритишни талаб этади. Чунки кўп ядроли процессорларда маълумотлар оқимини бошқариш имконияти борлиги сабабли ҳисоблаш юкламаларини бир нечта ҳисоблаш қурилмаларига тенг ажратиш ва оқимли ҳисоблашларни бажариш мумкин[1,2].

Параллел ҳисоблашнинг дастурий ва аппарат жиҳатлари бир-бири билан чамбарчас боғлиқ. Одатда параллел ҳисоблаш ўзида 2 турдаги ҳисоблаш технологиялар соҳасини ўз ичига олади:

- Компьютер архитектураси(аппарат аспектлари).
- Параллел дастурлаш(дастурлаш аспектлари).

Компьютер архитектураси параллелизмни архитектура босқичида қўллаб-қувватлашга эътиборини қаратган. Параллел дастурлаш эса компьютер архитектураси имкониятидан тўлиқ фойдаланиш эвазига масалани бир

вақтнинг ўзида ҳисоблашга мўлжалланган. Дастур кодини параллел бажарилишини таъминлаш учун аппарат таъминот бир нечта оқимлар ёки бир нечта жараёнларни бир вақтда бажарилишини қўллаб-қувватловчи платформани таъминлаш керак бўлади. Шунинг учун ҳам параллел дастурларни тузувчилар турли типдаги параллел процессорлар архитектураларининг(умумий хотирали, тақсимланган хотирали ва гетероген) имкониятларини, ҳамда ҳар бир архитектурага мос келувчи параллелаштиришнинг замонавий технологиялари ва дастурий воситалари саналган (махсуслаштирилган параллел дастурлаш тиллари ва дастурлаш тилларининг кенгайтмалари, махсуслаштирилган параллел кутубхоналар, автоматик параллелаштириш воситалари, параллел дастурларни лойиҳалаштириш ва яратиш воситаларини) мукаммал билишни талаб этади. Биз ушбу ишда гетероген ҳисоблаш архитектурасининг асоси ҳисобланган GPU архитектурасини имкониятларини таҳлил қиламиз[13].

1. График процессорлар архитектураси. График процессор архитектуралари оқимли мультипроцессорлар массиви (Streaming Multiprocessor, SM) асосида қурилган. GPU ядролари ишга туширилганда SM ларда мавжуд турлардаги блоклар ишга туширилади. Кўп оқимли дастур бир-биридан мустақил равишда ишловчи блокдаги оқимларга ажратилади. Шунинг учун GPU кўп сондаги ядролар ҳисобига ҳисоблашларни кичик вақтда бажаради [13].

GPU CUDA да архитектура имкониятларини белгилашда бир жуфт сонлардан иборат Compute capability (CC) тушунчасини ишлатади. CC даги биринчи сон глобал архитектура версиясини, иккинчи сон эса архитектурадаги катта бўлмаган ўзгаришларни англатади. GPU да оптимал дастурни ёзиш учун қурилма архитектурасининг хусусиятларини ва уларнинг дастурлаш моделларини ўрганиш зарур. GPU ларнинг Tesla, Fermi, Kepler, Maxwell ва Pascal архитектураларини таҳлил қиламиз. Ҳозирги кунда GPU архитектуралари уларнинг CC дан келиб чиқган ҳолда 5 та оилага ажралади(1-жадвал).

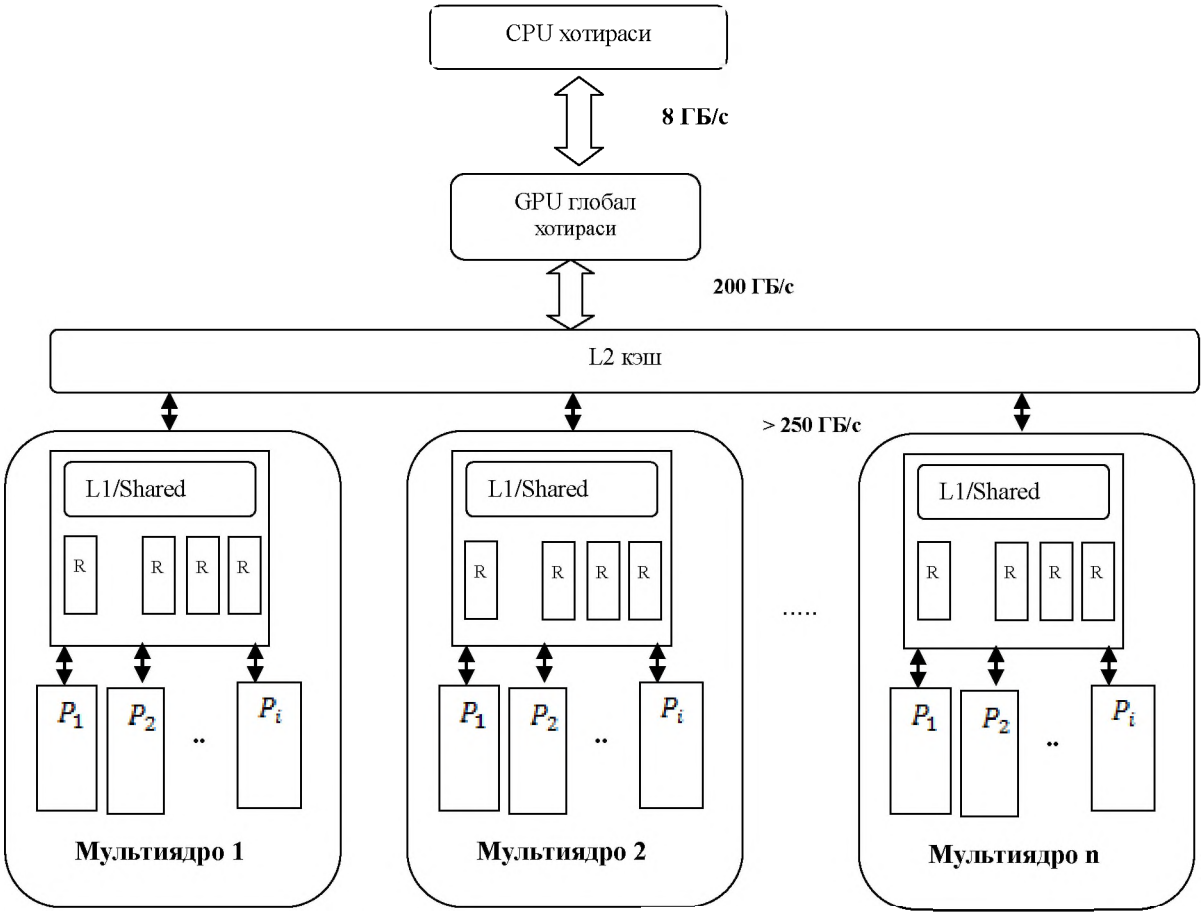
1-жадвал. GPU архитектура оиласи.

CC	Архитектура	Архитектурага кирувчи GPU
1.0	Tesla	Tesla C870, Tesla D870
1.3		Tesla C1060
2.0	Fermi	Tesla C2075, Tesla C2050
2.1		GeForce GT 720M
3.0	Kepler	Tesla K10
3.5		Tesla K40, Tesla K20x/K20
3.7		Tesla K80

5.0	Maxwel	GeForce GTX 850M
5.2		Tesla M40, Tesla M60
6.0	Pascal	Tesla P100
6.1		GeForce GTX 1080, NVIDIA Titan X

GPU глобал архитектура версиясинининг ошиб бориши билан унинг унумдорлиги ўсиб боради. GPU нинг CC ни тавсифловчи асосий кўрсаткичларга архитектурадаги ядролар сони, хотира хажми ва ўтказувчанлик қобиляти киради.

GPU кўп сондаги SIMD(Single Instruction Multiple Data) ядролардан ташкил топган бўлиб, бу ядролар турли типдаги маълумотлар устида битта микродастурни амалга оширади. Бундай микродастур GPU дастурлаш терминологиясида ядро функцияси (kernel function) деб номланади. SIMD ядро 1-расмда P_i деб белгилаш киритилган.



1-расм. График процессор архитектураси.

GPU маълумотларни бир-бири билан ўзаро таксимловчи бир қанча SIMD ядроларини битта ишчи гуруҳга(workgroups) бирлаштиради. Ишчи

гуруҳлар регистрлар, кэш ва умумий хотиралар ёрдамида ишчи гуруҳ ичида қўшимча маълумотлар билан алмашиниши мумкин ва алоҳида SIMD ядролар ўртасида синхронизацияни амалга оширади. Бундай вақтда GPU алоҳида ишчи гуруҳларнинг ўзаро ишлашини синхронизациялаш тақдим этмайди. Ишчи гуруҳлар 1- расмда “Мультиядро n” деб белгиланган[12, 14].

Ҳар бир оилага тегишли GPU архитектуралари таснифи ва уларни ташкил этувчи электрон компоненталарининг хусусиятларини таҳлил қилиш натижасида қўйидаги таҳлилий 2-жадвални ҳосил қилдик. Бу жадвалда дастур унумдорлигини оширишга таъсир кўрсатувчи энг муҳим қўйидаги кўрсаткичлар келтирилган:

- GPU да мавжуд оқимли мультипроцессорлар сони;
- CPU ва GPU ўртасидаги алоқани таъминлаб берувчи PCI-Express шина тури(PCI-e ўтказувчанлик қобиляти : v2 - 8 GB/s, v3 - 15.75 GB/s, NVLink – 80 GB/s);
- GPU да мавжуд хотирлар ҳажми (Глобал, кэшлар, регистрлар);
- ҳар бир оқимли мультипроцессордаги ядролар сони;
- глобал хотирадан юклаш ва сақлаш сўровлар амалга оширувчи Load/Store юнитлар сони;
- оқимли мультипроцессордаги оқимлар гуруҳи саналган варпларни режалаштиргичлар(планировшиклар) сони;
- битта оқимли мультипроцессордаги актив блоклар сони;
- аппаратли ташкил қилинган махсус математик функцияларни бажарувчи SFU(Special Function Unit) лар сони;
- хотиранинг ўтказувчанлик қобиляти;
- транзисторлар сони.
-

2-жадвал. График процессорларнинг кўрсатгичлари

	Fermi	Kepler GK110	Maxwell GM100	Maxwell GM200	Pascal P100
SM лар сони	16	15	16	24	56
PCI-e версияси	v2	v3	v3	v3	v3, NVLink
Глобал хотира	~ 6 Гб	~ 12 Гб	8 Гб	12 Гб	16 Гб
L2 кэш, Кб	768	1536	2048	3072	4096
CUDA ядролар (per SM)	32	192	128	128	64
LD/ST Unit лар (per SM)	16	32	32	32	16
SFU лар (per SM)	4	32	32	32	16
Warp Schedulerлар (per SM)	2	4	4	4	2
Shared хотира (per SM), KB	16/48	16/32/48	96	96	64
Регистрлар (per SM)	32K	65K	65K	65K	65K

Актив блоклар(per SM)	8	16	32	32	32
Транзисторлар сони	1.5 млрд	3.5 млрд	7 млрд		15.3 млрд
Хотира ўтказувчанлик қобиляти, GB/s	144	250	336	288	720

2. Хотира иерархияси. GPU да самарали дастурларни яратишда дастурчи томонидан билиш зарур бўлган яна бир аппарат қисми бу хотирадир. GPU нинг хотиралари иерархик структурага эга (1-расм). GPU да бир неча кўринишдаги хотиралар тури мавжуд бўлиб, уларнинг бир нечтаси мультипроцессорларда қолганлари эса GPU DRAM хотирасида жойлашган. GPU хотира иерархиясида глобал хотира, локал хотира, текстурали хотира, L2 кэш, L1 кэш, текстурали кэш, умумий хотира(shared memory) ва регистр файллар мавжуд.

Регистр хотира. Ҳисоблаш қурилмасига регистр хотира энг яқин жойлашган бўлиб, у жуда тезкор саналади(SMда ўтказувчанлик тезлиги 0.5-2 ТБ/с етади). Регистрлар анча соддалаштирилган хотира типи саналади ва қуйидаги хусусиятларга эга :

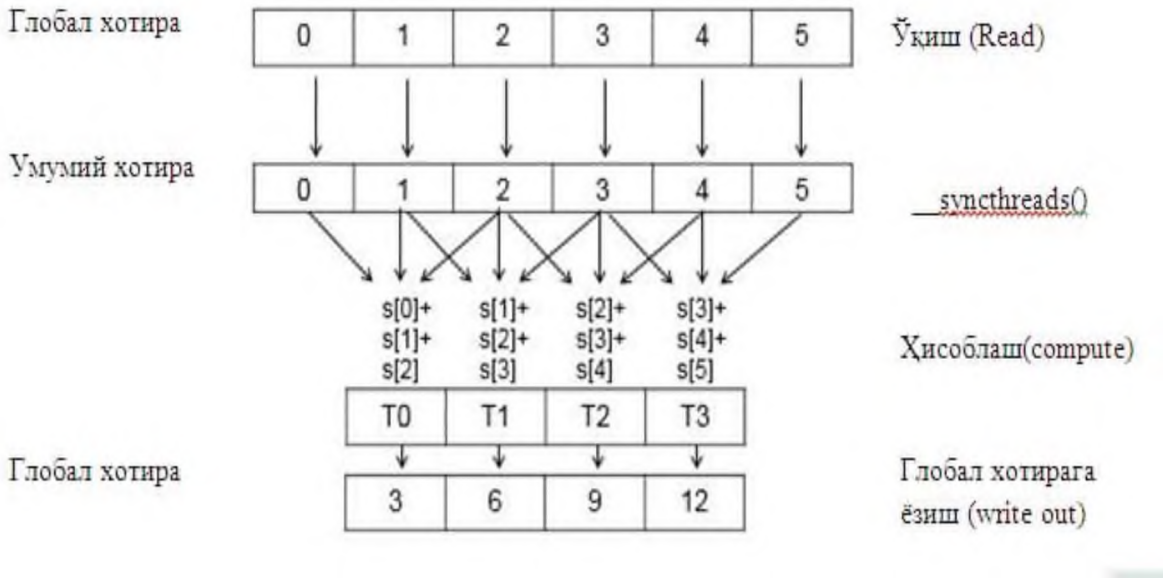
- регистрлар компиляция жараёнида блок оқимлари ўртасида тақсимланади;
- ҳар бир оқимнинг ишчи маълумотларини сақлайди;
- бошқа хотиралардан фарқли равишда ихтиёрий адреслашни қўллаб қувватламайди;
- ҳар бир оқим ядро бажарилишининг барча вақтида бир қанча регистрларда ўқиш ва ёзиш учун манапол равишда фойдаланади;
- оқим бошқа бир оқимнинг регистрларидан фойдалана олмайди;
- регистрлар мультипроцессорларда жойлашган ва минимал латентликга эга;
- SM да ўтказувчанлик тезлиги 0.5-2 ТБ/с етади.

3-жадвал. Регистр хотиранинг турли архитектуралардаги сони ва ҳар бир оқим учун тақсимланиши

GPU архитектураси	**Fermi GF100**	**Fermi GF104**	**Kepler GK104**	**Kepler GK110**
Compute Capability	**2.0**	**2.1**	**3.0**	**3.5**
32-bit Registr/SM	32768	32768	65536	65536
Max Registr/Thread	63	63	63	255

Умумий хотира(shared memory). Умумий хотира GPU даги энг муҳим хотира саналади ва қўйидаги таснифга эга:

- умумий хотира SM да жойлашган ва блок босқичида ажратилади;
- ҳар бир блок учун бир хил сондаги умумий хотира ажратилади;
- хотира ҳажми 48 Кбайт(Fermi архитектурасида 64 Кбайт);
- қўйи латентликга эга;
- битта SM блок оқимлари фойдаланиши мумкин;
- глобал хотирага мурожаатни камайтириш мақсадида умумий хотирадан буфер сифатида фойдаланиш мумкин;
- умумий хотира ҳажми SM да ишга туширилган барча блок оқимлари ўртасида тенг тақсимланади;
- умумий хотира ўлчами CUDA ядро томонидан __shared__ атрибути орқали массивларни аниқлаш ёки ядрони ишга тушириш параметрлари асосида берилиши мумкин;
- умумий хотира бир нечта оқимлар блокида фойдаланганлиги сабабли race condition дан сақланиш учун оқимли баръерли синхронлаштириш __syncthreads() дан фойдаланилади;
- Синхронлаштиришнинг типик алгоритми қўйидаги этапларда амалга оширилади(3-расм):
  1. глобал хотирадан маълумотларни shared хотирага юклаш
  2. __syncthreads() синхронлаштириш;
  3. юкланган маълумотлар устида ҳисоблашни олиб бориш;
  4. Натижаларни shared хотирадан глобал хотирага ёзиш.



3-расм. Shared хотира фойдаланишда синхронлаштириш

Константали ва текстурали хотиралар. Константали ва текстурали хотиралар GPU DRAM да жойлашган ва қўйидаги хусусиятларга эга:

- кэшдан мустақил равишда юқори тезликдаги мурожаат ҳуқуқига эга;
- турдаги барча оқимлар учун фақат ўқиш ҳуқуқини беради;
- константхотиранинг умумий ҳажми 64 кбайтни ташкил этади;
- Текстурали хотира физик жиҳатдан глобал хотиранинг бир қисми саналади, мурожаат ҳуқуқи махсус кэш орқали бажарилади.

Кэшлар. Fermi архитектурасидан бошлаб L1 ва L2 босқичдаги кэшлардан фойдаланади ва қўйидаги хусусиятларга эга.

- L1 кэш ҳар бир SM да мавжуд;
- L1 кэш ва умумий хотира битта физик ташувчида жойлашган. Унинг ҳажми 64 Кбайт;
- кэш линиялар узунлиги 128байт;
- агар ҳар бир оқим учун сўз ўлчами 4 байтни ташкил этса, унда барча 32 та варп оқимларининг хотирага сўрови биттага бирлаштирилади;
- L1 ва L2 кэшларнинг SM да ўтказувчанлик қобиляти 256-512Гб/с етади;
- L2 кэшнинг умумий ҳажми 256 КБ дан 4096 КБ (қурилма архитектурасига боғлиқ) етказилган.

Локал хотира. Хотира иерархиясининг кейинги босқичида умумий хотира жойлашган. Умумий хотира OpenCL терминларида локал хотира хам дейилади. Локал хотира SM оқимларига регистлар етишмаганда фойдаланилади ва қўйидаги хусусиятларга эга:

- локал хотира GPU DRAM да жойлашганлиги сабабли унда юқори латентлик пайдо бўлади(400-800 такт);
- локал хотирага одатда union (бирлашмалар), компиляция жараёнида ўлчами аниқ бўлмаган катта структуралар ва массивлар жойлашади;
- агарда ядро барча регистрларни ҳисоблаш жараёнида эгалласа барча ўзгарувчилар ҳам локал хотирага жойлаштирилади;
- GPU нинг CC си 2.0 ва ундан катта қурилмалардан бошлаб локал хотира ҳар бир SM учун L1 кэш томонидан умумий қурилма учун L2 кэш томонидан кэшлашланади.

Глобал хотира. Глобал хотира GPU нинг DRAM хотираси саналади қўйидаги хусусиятларга эга:

- ўтказувчанлик қобиляти 140-260 ГБ/с. Юқори латентликга эга 400-800 такт атрофида;
- CUDA API ёрдамида malloc функцияси асосида хотира ажратилади;
- Fermi архитектурасидан бошлаб кэшланади;
- турдаги барча оқимлар учун мавжуд;

- глобал хотирага мурожаат қилишни минимизациялаш самарали дастур яратишнинг асосий усули ҳисобланади;
- Глобал хотирага қўйидаги функциялар ёрдамида ажратилади ва бўшатилади:
  - cudaError_t cudaMalloc(void **devPtr, size_t size);
  - cudaError_t cudaFree(void *devPtr);
- массивларга мурожаат қилишда маълумотлар типини бараварлаштириш ҳисобига тезлаштириш мумкин;
- Бараварлаштирилган хотира ҳажмини ажратиш учун функция силжишни pitch параметри орқали қайтаради:
  - cudaError_t cudaMallocPitch(void *dst, const void *src, size_t size, enum cudaMemcpyKind kind);
  - cudaError_t cudaFree(void *devPtr);
- Нусхалаш йўналиши kind параметри орқали аниқланади:
  - Host->device- cudaMemcpyHostToDevice;
  - Device->host- cudaMemcpyDeviceToHost;

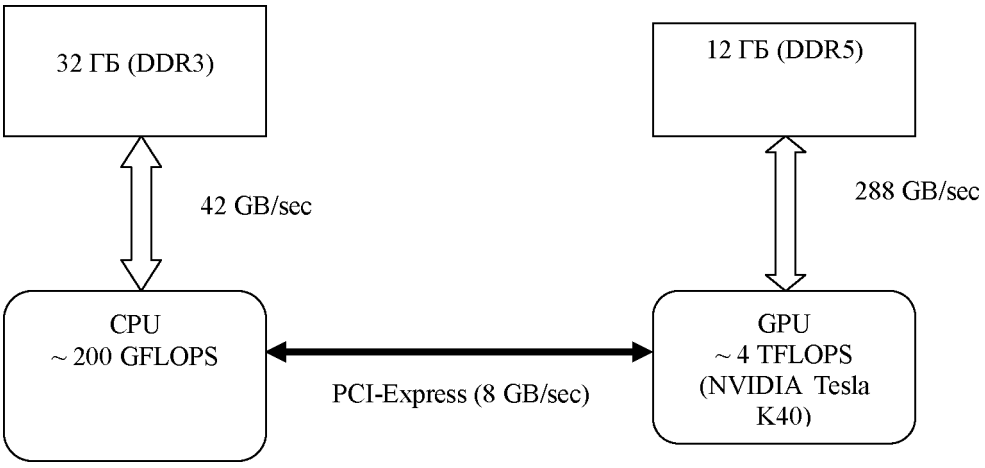
Юқорида келтирилган хотираларни таҳлили шуни кўрсатадики, глобал хотира паст латент лик эга. Ҳисоблаш жараёнини оптимизациялаш учун ушбу хотирага мурожаатни минималлаштириш керак бўлади[21].

3. Маълумотларни самарали ўзатиш. Бизга маълумки, гетероген ҳисоблаш тизимлари CPU ва GPU ресурсларини биргаликда фойдаланади. Чунки бундай ҳисоблаш тизимида дастурнинг параллел бажариладиган қисми GPU глобал хотирасига нусхаланади, ҳисоблашлар бажарилади ва олинган натижалар CPU хотирасига қайта юкланади. Бундай юклаш жараёни PCI-Express шинасида амалга оширилади. Шундай экан PCI-Express шинаси CPU ва GPU ни боғловчи нозик қисм(узкий часть) саналади. Шунинг учун ҳам шинанинг хоссаларини билишимиз зарур. Қўйида келтирилган 4- расмда қурилмалар ўртасидаги ўзаро маълумот ўзатиш тезликлари келтирилган.

Юқоридаги 4-расмдан кўриниб турибдики, маълумотларни кучириб ўтказиш дастурнинг умумий унумдолигига катта таъсир кўрсатади. Айниқса PCI-Express шинасининг тезлиги график процессорда ҳисоблашларни амалга оширишнинг камчиликларидан биридир.

CPU ва GPU ўртасида маълумотларни кучириб ўтказиш қўйидаги ҳолларда амалга оширилади:

- Кирувчи/чиқувчи массивларни ўзатишда.
- Ядронинг бинар кодларини юклашда(драйвер бошқаруви остида).
- Ядро аргументларини юклашда(ядрони ишга туширишдан олдин GPU константали хотирага жўнатади).
- Олинган натижаларни host га юклаш жараёнида.



4-расм. Қурилмалар ўртасидаги маълумот ўзатиш схемаси

Ҳозирги кунда PCI-Express шиналари x2, x4, x8, x16, x32 линиялардан иборат. Битта (x1) линия 500Мб/сек ўтказувчанлик қобилятига эга. PCI-Express шиналарининг ривожланиш авлодлари қўйидаги 4-жадвалда келтирилган.

4-жадвал. PCI-Express шиналарининг ривожланиш авлодлари.

PCI Express версияси	**Бир томонлама ўтказувчанлик тезлиги(Per Lane Bandwidth)**	**x16 Bandwidth**
1.0 (2003)	250 MB/s	4 GB/s
2.0 (2007)	500 MB/s	8 GB/s
3.0 (2010)	984 MB/s	15 GB/s
4.0 (2017)	1969 MB/s	31 GB/s
5.0 (?)	3125-4000 MB/s	50-64 GB/s

Жадвалдан кўриниб турибдики, PCI-Express шиналарининг янги версияси аввалги версиясига нисбатан тезлиги 2 марта ошиб бораётганлигини кўришимиз мумкин. Ушбу шиналаридан унумли фойдаланиш учун GPU хотирасига ўзатиладиган маълумотлар миқдори катта бўлиши керак. Кичик миқдорда маълумотларни GPU хотирасига ўзатиш шина линияларидан тўлиқ фойдаланиш имкониятини бермайди[14].

4. GPU да дастурлаш платформалари. Ҳозирги кунда GPU да ҳисоблашларни амалга ошириш учун бир нечта дастурлаш платформалари мавжуд. Бу платформаларни дастурлашда қўллаш турлича ёндашувни талаб этади. Бунда дастурлаш платформаларига Microsoft DirectX, OpenCL, CUDA киради.

Microsoft DirectX - Microsoft платформаларида мультимедиага алоқадор масалаларни, айниқса видео ва уйинларни дастурлаш учун мўлжалланган амалий дастурлаш интерфейслари(APIs) тўпламидир[16]. Microsoft DirectX платформасининг ютуқ ва камчилик томонларини қараб чиқамиз.

Microsoft DirectX платформасининг афзалликлари:

1. API- интерфейсларни ўзлаштириш дастурчилар учун осонлаштирилган.
2. График ресурс компоненталари билан самарали ишлай олади.
3. DirectX принципларидан фойдаланувчи уйинлар учун яхши қўлланилади.
4. Текстуралаштириш функцияларига мурожаат қилиш хуқуқига эга.
5. DX10 ва DX11 график процессорларида кўпроқ ишлатилади.

Камчиликлари:

1. Фақат Windows 7 ва Vista операцион тизимлари қўллаб-қувватлайди.
2. DirectX бўйича адабиётлар, мисоллар ва кутубхоналарнинг камлиги.
3. CPU томонидан откатка қилинмайди.

OpenCL(Open Computing Language)- CPU ва GPU лардан ташкил топган гетероген платформаларда дастур ёзиш учун мўлжалланган фреймворкдир[12]. Ушбу дастурий платформанинг ютуқ ва камчилик томонларини қараб чиқамиз.

OpenCL платформасининг афзалликлари:

1. Кенг спектрдаги турли аппарат воситаларда ва бир нечта платформаларда ишлатиш мумкин. AMD, Nvidia ва Intel GPU ларини бир-хилда қўллаб қувватлайди. Ҳозирда ушбу платформани замонавий мобил телефонларда ва бошқа электрон қурилмаларда ҳам фойдаланмоқда.
2. Агарда GPU аппарат таъминоти бўлмаганда, у кўрсатилган ишни бажариш учун CPU га мурожаат қилиш имконияти мавжуд.
3. Бир нечта қурилмалар ўртасида синхронизацияни амалга ошириш имкониятининг мавжудлиги.
4. Технологияни тез ўзлаштириш мумкин. Ишлаб чиқарувчи томонидан имконияти чекланмаган очиқ стандартни тақдим этади.
5. Ўзлаштириш учун етарлича ресурслар мавжуд.

Камчиликлари:

1. Ҳисоблаш аниқлигига боғлиқ чекловлар мавжуд. OpenCL фақат 32-битлик ҳақиқий сонлар(single-precision floating point) билан ишлай олади.
2. Атомар операциялар билан боғлиқ чекловлар. Атомар операция бу бутун типга эга ўзгарувчи қийматини ошириш ёки камайтириш жараёни тушунилади. Шунинг учун ҳам бундай счетчикларни локал гуруҳлар ичида ишлатиш синхронлаштириш примитивлардан(барьерлар) фойдаланишни талаб қилади.
3. Хотирага мурожаат қилиш ҳуқуқидаги чекловлар. Хотирага кетма-кет бўлмаган мурожаат қилиш унумдорликни йўқотишга олиб келади. График процессорларда самарали параллел ҳисоблашларни амалга ошириш учун хотирага махсус тарзда мурожаат қилиш керак.

4. OpenCL да дастур кодини профиллаштириш инстурментлари бошқа платформаларга нисбатан камлиги.

CUDA (Compute Unified Device Architecture) – бу дастурий-аппарат архитектура ҳисобланиб, NVIDIA компаниясининг график процессорлардан фойдаланган ҳолда видеокартада ҳисоблашларни бажарилишига имконият яратадиган технологиялардан биридир[13]. Ушбу дастурий платформанинг ютуқ ва камчилик томонларини қараб чиқамиз.

CUDA дастурий-аппарат архитектурасининг афзалликлари:

1. CUDA ядро функциясини Си дастурлаш тилида ёзилиши сабабли дастурчилар учун қийинчилик тўғдирмайди.
2. Ушбу технология бир нечта дастурлаш тиллари билан интеграцияланаолади.
3. Ядро коди кўрсаткичлар билан ишлашни тўлиқ қўллаб қувватлайди. Бу эса хотира сарфини камайишига сабаб бўлади.
4. C ++ конструкцияларини қўллаб қувватлайди.
5. GPU да тезкорликни оширувчи тайёр кутубхоналарнинг мавджудлиги.
6. CUDA nvcc компилятори бошқа компиляторларга(Microsoft Visual Studio) нисбатан яхши оптимизацияланган.
7. Турли платформалар учун документациялар кўринишдаги адабиётлар ва дастур кодлари етарлича берилган.
8. CUDA да дастурчиларга етарлича унумдоликни таҳлил қилувчи инстурментал воситлар(Visual profiler, NVIDIA® Nsight™, TAU Performance System®) мавжуд.
9. Ҳисоблаш ресурслари(блоклар ва оқимлар) имкониятидан тўлиқ фойдалана олиш ва хотира модулини бошқариш имконияти(shared memory).

Камчиликлари:

1. Фақат Nvidia GPU ларида ишлайди.
2. GPU ресурси етишмаса ҳисоблашлар бажарилмайди.
3. CUDA дастурлашни бошловчилар учун дастур соҳасини созлашдаги қийинчиликларнинг мавжудлиги.
4. CUDA да самарали дастурларни яратиш учун дастурловчилардан қурилма архитектурасини яхши билишни талаб этади.

5. Оптимизациялаш усуллари. GPU да ҳисоблашларни оптимизациялаш қўйида келтирилган принциплар асосида амалга оширилади:

1. **Дастур кодини параллелаштириш усулларини танлаш.** GPU да энг яхши унумдорликга эришиш учун масалани шундай қисм масалаларга ажратиш керакки, маълумотлар бўйича параллелизмдан тўлиқ фойдалана олишимиз керак. Бундай вазиятда параллелизм қоидалари(мисол учун оқимлар маълумотларни ўзаро бўлиб олиш учун синхронлаштириш

зарурияти туфайли) бузилиши мумкин. Шунинг учун қўйидаги ҳаракатларни қўллаш мақсадга мувофиқдир :

a) Агар оқимлар битта блокга тегишли бўлса, маълумотларни самарали алмашиш учун умумий хотирадан(shared memory) фойдаланиш мақсадга мувофиқ саналади. Бунинг учун биз __syncthreads() функциясини битта ядро ичидаги оқимларни синхронлаштириш учун ишлатамиз.
b) Агар оқимлар турли блокларга тегишли бўлса, унда глобал хотирада оралиқ ёзувлари мавжуд бўлган 2 та турли ядролардан фойдаланиш самарали ҳисобланади.

Бундан ташқари GPU да дастур кодини параллелаштириш учун мўлжалланган сигналларга ва тасвирларга ишлов бериш учун қўлланиладиган жуда катта тўпламдаги минглаб примитивлардан ташкил топган NVIDIA Performance Primitives(NPP), тезкор Фурье алмаштиришни амалга оширувчи cudaFFT, юқори даражадаги машинали ўқитишни(машинного обучения) нейрон тармоқлари билан интеграциялаш учун мўлжалланган NVIDIA cuDNN, матрциалар устида амаллар бажариш, чизиқли алгебра масалаларини тезкор ечишга мўлжаллаган кутубхоналар, параллелаштиришни автоматлаштирувчи OpenACC, OpenMP 4.0 директивалар ва дастурлаш платформалари (CUDA, OpenCL, DirectX) мавжуд [17].

2. **Ҳисоблаш қурилмаси имкониятларидан максимал даражада фойдаланиш учун ядро конфигурациясини(блоклар ва оқимлар сони) тўғри созлаш.** GPU да ҳисоблаш блоклари ва оқимларини бир ўлчовли(1D), икки ўлчовли(2D), уч ўлчовли(3D) ҳосил қилиш мумкин. GPU ҳисоблаш имкониятидан тўлиқ фойдаланиш учун қўйилган масалани турига боғлиқ ҳолда оқим ва блокларни ҳосил қилишимиз лозим. Шуни эътибордан қочирмаслик керакки, GPU архитектураси графикага доир масалаларни ечиш учун кўпроқ мўлжалланган. Бизга маълумки тасвир 2D характерга эга, шундай экан тасвирлар устида бажариладиган амалларни самарали бажаришимиз учун 2D ўлчамли блоклар ва 2D оқимлар ҳосил қилишимиз керак. Бу эса GPU нинг максимал имкониятларидан фойдаланишни таъминлайди. Энди шу ўринда 2D блок ва 2D оқимлар ўлчами қандай бўлса умумдорлик юқори бўлади деган савол пайдо бўлади. Битта блокдаги оқимлар сонини архитектура имкониятидан келиб чиққан ҳолда танлашимиз керак. Мисол учун Nvidia GeForce GT 630 видеокарта архитектураси CUDA учун энг кўпи билан битта блокда бир вақтда бажариладиган 512 та оқим ҳосил қилиб беришга қодир. Биз битта блокда NxN (N^2<=512) ўлчамли оқим ҳосил қилсак, энг самарали бўлади. Яъни битта блокдаги барча оқимлар бир вақтда бажарилади ва бир вақтда якунланади. Блок ичида оқимларда навбат кутиш бўлмайди. Турдаги(grid) блоклар ўлчами масалани тулиқ ҳисоблаш учун керак бўладиган оқимларни ҳосил қилиб бериш учун етарли

бўлиши керак. Мисол учун 2048 та оқим ҳосил қилишимиз учун бизга тўрта блок кифоя саналади.

3. Глобал хотирага бўладиган сўровларни бирлаштиришни(coalescing) таъминлаш. Глобал хотирани адреслашда параллел фойдаланиш ҳуқуқи ёки сўровларни бирлаштириш(coalescing) қоидасига риоя қилиш керак. Чунки глобал хотирадаги маълумотлар кэшланади. Кэш линиялар бир мурожаатда 128 байт(қурилманинг Compute capabilite >=3.0) ҳажмдаги глобал хотирада кетма-кет адресланган маълумотларни кэшлайди. Агар глобал хотирада маълумотлар кетма-кет адресланмаса хотирага мурожаатлар сони ошиб кетади. Глобал хотирадан фойдаланишда яхши унумдорликга эришиш учун варпдаги оқимлар 4, 8 ёки 16 байтли маълумотлар типини кетма-кет адреслаш орқали жойлаштириш лозим ҳамда барча маълумотлар блокини 32, 128 ёки 256 байт чегарада бараварлаштириш керак. Глобал хотирага бўладиган сўровларни бирлаштириш эвазига латентлиги юқори бўлган хотирага ортиқча мурожаатларни камайтириш мумкин бўлади.

4. Хост ва қурилма ўртасидаги маълумот узатишни оптимизациялаш. Хост ва қурилма ўртасидаги маълумотлар алмашинув асосан PCI-E шинасида амалга оширилади. Чунки энг тезкор вариантдаги маълумотларни нусхалашни амалга оширадиган шинанинг(PCI-E 3.0×16) ўтказувчанлик қобиляти 16 ГБ/с га тенг. Қурилма ичидаги динамик хотиранинг ўтказувчанлик қобиляти ҳозирги вақтда 192 ГБ/с га етади(GeForce GTX 550 Ti график процессори учун). Хост ва қурилма ўртасидаги маълумотлар алмашинуви GPU да параллел ҳисоблашнинг энг нозик жойи саналади. Бу жараённи яхшилаш учун қўйидаги усуллар ёрдам беради:

a) PCI-E шинаси орқали маълумотларни транзакциялаш узатиладиган маълумотларнинг етарлича катта хажмда(>=10Мб) бўлса самарали амалга оширилади. Кичик ҳажмга эга маълумотларни жўнатиш PCI-E шинасининг линияларидан тўлиқ фойдалана олмаслиги сабаб бўлади. Бу эса маълумотларни транзакциялаш тезлиги пасайишига олиб келади[14].

b) Одатда CUDA да маълумотларни қурилма хотирасига нусхалашда *cudaMemcpy()* функцияси ишлатилади. Бу функцияда синхрон нусхалаш механизми мавжуд бўлиб, ядро бошқарувини нусхалаш тугаллангандан сўнг қайтаради. Бу вақтда ядро бошқа ишларни бажара олмайди. CUDA да яна битта маълумотларни нусхалаш функцияси мавжуд, бу функция *cudaMemcpyAsync()* деб номланади. Бу функцияда асинхрон нусхалаш механизми мавжуд. Маълумотларни асинхрон нусхалаш pinned-хотира билан биргаликда ишлайди. Pinned хотира бу хостнинг физик хотирасидан CUDA драйвер томонидан ажратиладиган виртуал хотира саналади. Асинхрон нусхалашда операцион тизим томонидан маълумотларни CPU оператив хотирасидан эмас балки pinned хотирадан юклайди.

5. Профайлер дастурларидан фойдаланиш. Дастур кодидаги унумдорликни пасайтирувчи омилларни топиш учун CUDA toolkit таркибида махсус visual profiler дастурий воситаси мавжуд. Профайлер ишга туширилиш жараёнида дастурнинг exe файли жойлашган йўл курсатилади ва кузатиш керак бўладиган счетчиклар(counters) танланади. Ҳар бир счетчик дастур бажарилишидаги ҳодисаларни кўрсатади. Профайлер ўз ишини тугатгандан сўнг оптимизациялаш мумкин бўлган счетчик қийматларини чиқаради. Қуйида келтирилган 5- жадвалда оптимизациялашда керак бўладиган энг мухим счетчик кийматлари келтирилган. Бу счетчик кийматлари асосида дастур унумдорлигини ошириш мумкин[13,17].

5-жадвал. Энг муҳим счетчик қийматлари

Счетчик қийматлари	Мазмуни
gld_incoherent /gst__incoherent	Coalescing қоидаси бузилган глобал хотирадан ўқиш
gld_incoherent /gst__coherent	Coalescing қоидасга амал қилинган ҳолда глобал хотирадан ўқиш
gld_32b gld_64b gld_128b	32 байтли, 64 байтли ва 128 байтли хотирага сўровлар(ўқиш)
gst_32b gst_64b gst 128 b	32 байтли, 64 байтли ва 128 байтли хотирага сўровлар(ёзиш)
local_load / local_store	Локал хотирага ўқиш/ёзиш
Instructions	Умумий бажарилган инструкциялар сони
warp_serialize	Умумий хотирага ёки константали хотирага мурожаат қилишда оқимларни сериализациялаш
branch / divergent_branch	Шартли операторлар сабабли шохланишлар сони

III. ХУЛОСА

Ушбу мақолада энг яхши дастурлаш воситаларига ва кутубхоналар тўпламига эга NVIDIA курилмасининг график процессорлар архитектурлари ва уларнинг ўзига хос хусусиятлари, дастур кодини оптимизациялаш усуллари ўрганилади ва тахлил қилинди. Тахлил натижалари асосида қўйидаги хулосалар олинди:

1. Сигналларни ва тасвирларни қайта ишлаш алгоритмлари аввалдан векторлар ва матрицаларни кўпайтириш процедурасини қўллаган. Шунинг учун ҳам бу алгоритмларни GPU да параллел қайта ишлаш муваффакиятли амалга оширилади.

2. Замонавий GPU архитектураси юқори унумдорли самарали параллел дастурларни яратиш имконияти тақдим этади. GPU да юқори унумли дастурларни ёзиш учун қурилма архитектураси имкониятларини, хотиралардан тўғри фойдаланиш ва оптимизациялаш усулларини билиши шарт.
3. Ҳисоблашларни CPU дан GPU ўтказилиши катта ҳисоблаш ресурсларини талаб қиладиган масалаларни тезлигини оширади. Лекин GPU да амалга оширилган ҳамма алгоритмлар ҳам CPU да амалга ошириш тезлиги билан солиштирганда юқори самарадорликни бермайди. Бунинг учун GPU да ечиладиган масалалар синфини аниқлаштириш керак.
4. GPU да параллел ҳисоблаш 2 та асосий сабаб туфайли секин бажарилиши мумкин буларга: ҳисоблаш қувватидаги чекловлар ва қайта ишланадиган маълумотлар ҳажмига боғлиқ чекловлар.
5. GPU да параллел ҳисоблаш жараёнини оптимизациялаш глобал хотирага мурожаатлар сонини қисқартириш ҳисобига эришиш мумкин.
6. Дастур кодини оптимизациялашда профайлер дастурлардан фойдаланиш дастур унумдорлигини оширишга ёрдам беради.

АДАБИЁТЛАР

[1] Мусаев М.М., Кардашев М.С., “Спектральный анализ сигналов на многоядерных процессорах//” Цифровая обработка сигналов. 2014. №2. С. 82-86.

[2] Хужаяров И.Ш., Очилов М.М. CUDA технологияси ёрдамида тасвирларни спектрал усулда кайта ишлаш.// Вестник ТУИТ 2017, № 1(41). С. 54-63.

[3] Bart Pieters et al. Motion estimation for H.264/ AVC on multiple GPUs using NVIDIA CUDA. Proc. SPIE, Vol. 7443, 74430X (2009).

[4] John E. Stone et al. High Performance Computation and Interactive Display of Molecular Orbitals on GPUs and Multi-core CPUs. ACM International Conference Proceeding Series; Vol. 383 (2009).

[5] Hongsheng Li “Actin Filament Tracking Based on Particle Filters and Stretching Open Active Contour Models”, Int’l Conf. Medical Image Computing and Computer Assisted Intervention (MICCAI), 2009.

[6] Пантелеев А.Ю. Цифровая обработка сигналов на современных графических процессорах. 15-я Международная конференция «Цифровая обработка сигналов и ее применение- DSPA’2013». стр. 68-75.

[7] C.Clemente. Processing of synthetic aperture radar data with GPGPU. IEEE Workshop on Signal Processing Systems. 2009. p.

309-314.

[8] Guzhva A., Dolenko S., Persiantsev I. Multifold Acceleration of Neural Network Computations Using GPU. Artificial Neural Networks – ICANN 2009, DOI 10.1007/978-3-642-04274-4, 2009.

[9] Govindaraju N. K., Gray J., Kumar R., Manocha D. Gputerasort: High performance graphics coprocessor sorting for large database management. In SIGMOD, 2006.

[10] Govindaraju N. K., Lloyd B., Wang W., Lin M., Manocha D. Fast computation of database operations using graphics processors. In SIGMOD, 2004.

[11] He B., Lu M., Yang K., Fang R., Govindaraju N. K., Luo Q., Sander P. V. Relational query coprocessing on graphics processors. In TODS, 2009.

[12] С. А. Запрягаев, А. А. Карпушин. Применение графического процессора в ресурсоемких вычислениях на базе библиотеки OpenCL. Вестник ВГУ, серия: Системный анализ и информационные технологии, 2010, № 2. стр. 79-87.

[13] John Cheng, Max Grossman, Ty McKercher. Professional CUDA® C Programming. Published by John Wiley & Sons, Inc.10475. 2014.

[14] А.Ивахненко. Оптимизации на основе архитектуры и управление потоками данных (https://compsciclub.ru/media/slides/gpucomputation_2017_spring/2017_03_04_gpucomputation_2017_spring.pdf).2017.

[15] Jayshree Ghorpade, Jitendra Parande "GPGPU PROCESSING IN CUDA ARCHITECTURE" Advanced Computing: An International Journal (ACIJ), Vol.3, No.1, January 2012.

[16] Direct Compute Programming Guide. 2010. (http://developer.download.nvidia.com/compute/DevZone/docs/html/DirectCompute/doc/DirectCompute_Programming_Guide.pdf).

[17] Боресков А. В., Харламов А. А. Основы работы с технологией CUDA. М.: ДМК Пресс, 2010. 232 с.: ил. ISBN 9785940745785.

[18] In Kyu Park, Nitin Singhal, Man Hee Lee, Student Sungdae Cho, and Chris Kim Members IEEE. Design and Performance Evaluation of Image Processing Algorithms on GPUs. IEEE Transactions on Parallel and Distributed Systems (Volume: 22, Issue: 1, Jan. 2011). p. 91 - 104.

[19] G. Shen, G.P. Gao, S. Li, H. Shum, and Y. Zhang, "Accelerate video decoding with generic GPU," IEEE Transactions on Circuits and Systems for Video Technology, vol. 15, no. 5, pp. 685–693, May 2005.

[20] J. Fung, S. Mann, and C. Aimone, "OpenVIDIA: Parallel GPU

computer vision," in Proc. of ACM international conference on Multimedia, November 2005, pp. 849–852.

[21] Wong H., Papadopoulou M. et. al. Demystifying GPU Microarchitecture through Microbenchmarking//IEEE ISPASS, pp. 235-246. IEEE(2010).

[22] Sidi Ahmed Mahmoudi and Pierre Manneback. Efficient Exploitation of Heterogeneous Platforms for Images Features Extraction. 2012 3rd International Conference on Image Processing Theory, Tools and Applications (IPTA).