МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКА НАЦІОНАЛЬНА АКАДЕМІЯ ХАРЧОВИХ ТЕХНОЛОГІЙ

XI МІЖНАРОДНА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ І АВТОМАТИЗАЦІЯ – 2018

Збірник доповідей

Частина I

Одеса,
4-5 жовтня 2018

ЗМІСТ

УДК 781.6.18

VYATKIN SERGEY I., ROMANYK ALEXANDER N.[1]
Institute of Automation and Electrometry SB RAS,
[1]Vinnytsia National Technical University

SEPARATION OF THE COMPLEX TASK OF PLANNING STREAMING MULTIPROCESSORS INTO SEVERAL SIMPLER WARP SCHEDULERS

Introduction

GPU architectures have emerged that use a stream as the basic computational unit [1-3]. What is a processor dependency (CPU dependency), to 3D-accelerators with reference to? This is because no existing 3D-accelerators of user class do not to accelerate the whole process of visualization three-dimensional scene (geometric transformations, calculation of luminosity, removing the invisible surfaces etc.). Different video cards accelerate different stages, usually lying closer to the calculation chain end sometime, for instance, rasterization (i.e. translation of two-dimensional vector expressing in the two-dimensional raster scene) and imposition of textures. The whole rest work lies on the shoulder of central processor (CPU). Herewith for organizations of interaction with 3D-accelerator is usually required certain additional number of steps. Processor dependency, thereby, consist from two component - a percent of calculations, not accelerated by the card, and percent of additional calculations, which are produced already by the graphic processor. As a processor dependency is connected with scalable, i.e. possibility of growing of card output with the processor output growing. In most cases there is certain length of processor power, under which card demonstrates nearly single-line growing of output. Bound this with that processor does not be able to prepare data for the graphic accelerator, and any data are processed before the arrival of following portions. Herewith, obviously that than greater area of tract of calculations undertakes a graphic accelerator, that load less on the processor, but signifies CPU quicker prepares data, and, as a result, is narrowed single-line scaling area. Thence follows that than less processor-dependent card-accelerator, that it usually less scalable. As from some moment graphics accelerator begins more and more to hold up a processor, and under the unlimited growing of output of last, velocity of rendering becomes constant, speed to peak accelerator output. Certainly a model several simplified. In real systems to the processor will always find, than will be occupied, additionally reception capacity of buses, sending data, on today highly limited. That apropos is one more narrow bottleneck on way of achievement of maximum output. As evaluations of limiting output of accelerator possible to use known features of amount of triangles, which it can process at a second and velocity a fill-rate, i.e. amount texturing and filtering pixels taken out at one second.

Begin from existing raster graphic system analysis and will show that nor one of these architectures does not allow single-line to scale output. In the machine graph enormous computing difficulty of algorithms and potentially endless difficulty of expressing models require a single-purpose hardware support. Big computing syntheses cost of photorealistic expressing results from complex geometric transformations, using the complex illuminating models, displaying a texture and surrounding ambiences, as well as methods of eliminating the distortion, appearing because of the discrete nature of devices of conclusion of expressing. In scientific visualizations by reasons of greater computing expenses are an enormous size visual models and algorithmic difficulty of separation significant information from possible multivariate given. Typical operation of ray tracing or ray-casting for the scalar field comprises of itself recovering a value to functions, calculation of gradient and its modular, separation of features of ambience, painters and compositions for each spot of ray. For this are required solely greater amounts, high reception capacity of memory, and enormous speed. Problems appear any time requests reach a limit of one graphic pipeline output, in most cases because of limited reception capacity of memory. Then for getting a desired speedup, it is necessary to put parallel several pipelines and select a suitable method of sharing data. In the ideal event system output grows single-line with number of parallel pipelines. However, in practice, graphs of output are asymptotically drawn near to certain value, but increasing output is in general founded on certain suggestions, which can and be not executed for all exhibits. The simple pipeline can execute calculations that necessary for visualization of polygonal surfaces. Logical sequence of calculations in the pipeline prompts a first method of multisequencing of problem to visualizations in the hardware pipeline. Such partitioning a problem on stage possible to find nearly in all produced today graphic systems. Method of multisequencing insufficient, if multiplies canonical scheme by means of repetitions of all components. Regrettably, duplicated "pipelines" cannot work independently. An ex-

change data must occur in certain spot between pipelines.

We consider the control logic, which distributes and plans to work and exchange of data for the cores of three well-known architectures.

Separation of the complex task

Fermi:

In computing architecture «Fermi» used the third generation of streaming multiprocessors (Streaming Multiprocessor). The Amount cores (CUDA cores), compared with the previous architecture, and has more than doubled.

Used several Polymorph Engines and ROP units (Raster Engines), working in parallel. Caches the first and second levels provide quick access to the geometric attributes of stream processors and blocks tessellation. Fast switching context between graphical and non-graphical calculations, the competitive performance of computing programs and improved architecture caching. Manager GigaThread is the center of the chip; it creates and distributes blocks streams for different multiprocessors. Multiprocessors distribute warp (warps, a group of 32 threads) among stream processors (CUDA cores) and other execution units. Each Streaming Multiprocessor (SM) supports up to 48 simultaneous execution of the warp and CUDA core can perform all types of programs: vertex, pixel, geometry, computational.

Multiprocessors running thread in groups of 32 pieces, these groups are called the warp. Each multiprocessor contains two Warp Scheduler (Warp Scheduler) and two controller instructions (Instruction Dispatch Unit), which allows to simultaneously performing two warps on each SM.

Double warp scheduler selects two warps and executes one instruction from each of them in a group of 16 cores, 16 blocks LSU or four SFU. Since warp executed independently of each other, the scheduler should not GPU check the flow of instructions dependent on the commands. Using this model, the simultaneous execution of two instructions (dual-issue) per cycle to achieve high performance close to the theoretical values of the peak.

Most instructions can be executed simultaneously in two: a pair of integer instructions, two floating point instructions, or a combination of integer, floating point instructions, load data, store data, special instructions SFU. Is this applies only to single-precision instructions.

For a modern GPU is very important and effective organization of the memory subsystem. Especially when more and more attention given to non-graphical computing. Was improved memory model. There is a dedicated cache of the first level in each multiprocessor (SM).

Cache memory is working with a shared (common) memory multiprocessor and complements it. Shared memory improves the speed with predictable memory access and cache L1 faster access when the address of the requested data is not known beforehand.

The unified cache is more efficient than separate caches for different purposes. When selected caches might get position when one of them is used in full, but to take advantage of idle volumes of other types of cache memory at the same time is impossible.

In addition, the effectiveness of caching is below the theoretically possible. A unified L2 cache dynamically allocates space below different needs, to achieve high efficiency.

One L2 cache replaces the texture cache and L2 cache ROP. Second-level cache is used to read and write data, and is fully consistent (coherent).

This ensures a more efficient exchange of data between pipeline stages. As well as significant savings in bandwidth capacity external memory.

Kepler:

As in the case of Fermi, Kepler architecture has in its composition a few blocks GPC (clusters of graphics processing - Graphics Processing Clusters), which are composed of independent devices GPU. These units can operate as separate units, since them composition have all the necessary own resources: rasterizer geometric calculators and texture units. That is, most performed within the functional blocks GPC. To download the data processing units SMX, each comprising four schedule block warp (warp scheduler), each of which, in turn, processes the two instructions per clock cycle per warp. On Compared with the SM Fermi, Kepler SMX architecture reduces the number of control logic in the chip. In Fermi complicated logic, and then Kepler to simplify it.

Although Kepler and Fermi contain similar hardware units that manage data loading and warp, flow control instructions, but the scheduler Fermi also contains more complex and hardware logic designed to prevent conflicts of access to data. Special table registers (multi-port register scoreboard) monitors registers, in which data are not yet ready, and the block check dependencies (dependency check)

analyzes using them, depending checking teams. However, once information about delays in Access is known in advance and they do not change, then a similar analysis can be carried out even in the compiler. Moreover, Kepler part of the control logic decided to transfer from GPU to the compiler, which is partly responsible for planning. Dependency checking and ordering instructions on Fermi implemented in hardware inside the GPU, in the case of Kepler compiler performs these tasks. Of course, this reduced the effectiveness of streaming data in some problems. However, in most applications, it is little different from the efficiency Fermi. However, taken solution allowed to remove the complex and energy-intensive units, replacing them with simple, easy to take predefined data about delays the compiler and use them in your planning. One of the most interesting features of the architecture is the technology of Kepler GPU Boost. It should accelerate performance. This is a combined hardware and software technology, which dynamically changes the frequency of GPU, based on the conditions of his work and some of the characteristics.

Maxwell:

One of the most interesting changes in the architecture of Maxwell became new streaming multiprocessors (Streaming Multiprocessor - SM), which have both better efficiency and productivity relative to the chip area. Despite the fact that SMX design multiprocessors in Kepler and so was quite effective in the development of architecture was modified Maxwell multiprocessors, giving them the name of the SMM. Has been improved a lot, including control units and planning, load balancing between the blocks, the number of issued for execution of instructions per clock cycle, and more. The organization changed multiprocessors very seriously. While multiprocessor SMX in Kepler is a big block, at Maxwell each multiprocessor further divided into four distinct logical computing sections, each of which has its own instruction buffer, the scheduler Warp and consists of 32 cores. Kepler architecture approach with the number of stream cores, not powers of two, was abolished, and a partition of SMM on computing topics similar to what it was in the Fermi. Separation of computational units simplified overall design and control logic chip, reduced latency, and chip area of energy to power.

REFERENCES

1. Duca, N., Cohen, J., and Kirchner, P., 2003. Stream caching: A mechanism to support multi-record computations within stream processing architectures. DIMACS

Working Group on Streaming Analysis II, March.

2. Feigenbaum, J., Kannan, S., Strauss, M., and Viswanathan. M. 1999. An approximate l1-difference algorithm for massive data streams. In Proc. 40th Symposium on Foundations of Computer Science, IEEE.

3. Kapasi, U. J., Dally, W. J., Rixner, S., Owens, J. D., and Khailany, B. 2002. The imagine stream processor. In Proc. IEEE International Conference on Computer Design, 282–288.

XI МІЖНАРОДНА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ І АВТОМАТИЗАЦІЯ – 2018

ОДЕСА
4 – 5 ЖОВТНЯ, 2018

Збірник включає доповіді учасників XI Міжнародної науково-практичної конференції «Інформаційні технології і автоматизація – 2018»

Редакційна колегія: Котлик С.В., Хобін В.А.

Комп'ютерний набір і верстка: Шамрай О.А.

Відповідальний за випуск: Котлик С.В.

ДЛЯ НОТАТОК