1. GPU и CPU: отличия в архитектуре, преимущества и недостатки проведения расчетов на них. Закон Мура, его расширение, учитывающее появление GPU.

Ядра CPU проектируются для выполнения одного потока последовательных инструкций с максимальной производительностью, а GPU — для быстрого исполнения очень большого числа параллельно выполняемых потоков инструкций. В этом и заключается принципиальное отличие графических процессоров от центральных. CPU представляет собой универсальный процессор или процессор общего назначения, оптимизированный для достижения высокой производительности единственного потока команд, обрабатывающего и целые числа, и числа с плавающей точкой. При этом доступ к памяти с данными и инструкциями происходит преимущественно случайным образом.

Для повышения производительности CPU они проектируются так, чтобы выполнять как можно больше инструкций параллельно. Например, для этого в ядрах процессора используется блок внеочередного выполнения команд, позволяющий переупорядочивать инструкции не в порядке их поступления, что позволяет поднять уровень параллелизма реализации инструкций на уровне одного потока. Тем не менее это все равно не позволяет осуществить параллельное выполнение большого числа инструкций, да и накладные расходы на распараллеливание инструкций внутри ядра процессора оказываются очень существенными. Именно поэтому процессоры общего назначения имеют не очень большое количество исполнительных блоков.

Графический процессор устроен принципиально иначе. Он изначально проектировался для выполнения огромного количества параллельных потоков команд. Причем эти потоки команд распараллелены изначально, и никаких накладных расходов на распараллеливание инструкций в графическом процессоре просто нет. Графический процессор предназначен для визуализации изображения. Если говорить упрощенно, то на входе он принимает группу полигонов, проводит все необходимые операции и на выходе выдает пикселы. Обработка полигонов и пикселов независима, их можно обрабатывать параллельно, отдельно друг от друга. Поэтому из-за изначально параллельной организации работы в GPU используется большое количество исполнительных блоков, которые легко загрузить, в отличие от последовательного потока инструкций для CPU.

Графические и центральные процессоры различаются и по принципам доступа к памяти. В GPU доступ к памяти легко предсказуем: если из памяти читается тексель текстуры, то через некоторое время придет срок и для соседних текселей. При записи происходит то же самое: если какой­то пиксел записывается во фрейм­буфер, то через несколько тактов будет записываться пиксел, расположенный рядом с ним. Поэтому GPU, в отличие от CPU, просто не нужна кэш­память большого размера, а для текстур требуются лишь несколько килобайт. Различен и принцип работы с памятью у GPU и CPU. Так, все современные GPU имеют несколько контроллеров памяти, да и сама графическая память более быстрая, поэтому графические процессоры имеют гораздо большую пропускную способность памяти, по сравнению с универсальными процессорами, что также весьма важно для параллельных расчетов, оперирующих огромными потоками данных.

В универсальных процессорах большую часть площади кристалла занимают различные буферы команд и данных, блоки декодирования, блоки аппаратного предсказания ветвления, блоки переупорядочения команд и кэш­память первого, второго и третьего уровней. Все эти аппаратные блоки нужны для ускорения исполнения немногочисленных потоков команд за счет их распараллеливания на уровне ядра процессора.

Сами же исполнительные блоки занимают в универсальном процессоре относительно немного места.

В графическом процессоре, наоборот, основную площадь занимают именно многочисленные исполнительные блоки, что позволяет ему одновременно обрабатывать несколько тысяч потоков команд.

Можно сказать, что, в отличие от современных CPU, графические процессоры предназначены для параллельных вычислений с большим количеством арифметических операций.

<https://compress.ru/article.aspx?id=23724>

<https://www.ixbt.com/video3/cuda-1.shtml>

<https://club.dns-shop.ru/Duff-Mаn/blog/Вычисления-на-графических-процессорах/>

**Зако́н Му́ра** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Moore's law*) — [эмпирическое](https://ru.wikipedia.org/wiki/%D0%AD%D0%BC%D0%BF%D0%B8%D1%80%D0%B8%D0%B7%D0%BC) наблюдение, изначально сделанное [Гордоном Муром](https://ru.wikipedia.org/wiki/%D0%9C%D1%83%D1%80,_%D0%93%D0%BE%D1%80%D0%B4%D0%BE%D0%BD), согласно которому (в современной формулировке) количество [транзисторов](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B8%D1%81%D1%82%D0%BE%D1%80), размещаемых на кристалле интегральной схемы, удваивается каждые 24 месяца. Часто цитируемый интервал в 18 месяцев связан с прогнозом Давида Хауса из [Intel](https://ru.wikipedia.org/wiki/Intel" \o "Intel), по мнению которого, *производительность* процессоров должна удваиваться каждые 18 месяцев из-за сочетания роста количества транзисторов и увеличения тактовых частот процессоров[[1]](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BA%D0%BE%D0%BD_%D0%9C%D1%83%D1%80%D0%B0#cite_note-1).

Рост числа транзисторов на кристалле микропроцессора показан на графике справа. Точки соответствуют наблюдаемым данным, а прямая — периоду удвоения в 24 месяца.

В последнее время, чтобы получить возможность задействовать на практике ту дополнительную вычислительную мощность, которую предсказывает закон Мура, стало необходимо задействовать [параллельные вычисления](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F). На протяжении многих лет производители процессоров постоянно увеличивали [тактовую частоту](https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%BA%D1%82%D0%BE%D0%B2%D0%B0%D1%8F_%D1%87%D0%B0%D1%81%D1%82%D0%BE%D1%82%D0%B0) и параллелизм на уровне инструкций, так что на новых процессорах старые однопоточные приложения исполнялись быстрее без каких-либо изменений в программном коде.[[10]](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BA%D0%BE%D0%BD_%D0%9C%D1%83%D1%80%D0%B0#cite_note-10) Примерно с середины десятилетия 2000-х годов по разным причинам производители процессоров предпочитают [многоядерные](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D1%8F%D0%B4%D0%B5%D1%80%D0%BD%D0%BE%D1%81%D1%82%D1%8C) архитектуры, и для получения всей выгоды от возросшей производительности ЦП программы должны переписываться в соответствующей манере. Однако не каждый алгоритм поддается распараллеливанию, определяя, таким образом, фундаментальный предел эффективности решения вычислительной задачи согласно [закону Амдала](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BA%D0%BE%D0%BD_%D0%90%D0%BC%D0%B4%D0%B0%D0%BB%D0%B0).

<https://habr.com/company/madrobots/blog/405413/>

<http://isicad.ru/ru/news.php?news=18203>

<https://dtf.ru/gamedev/10882-nvidia-zakon-mura-poteryal-aktualnost-a-gpu-stanut-zamenoy-cpu>

1. Основные понятия архитектуры CUDA (сетка, блок, поток, warp, ядро, хост, устройство).

<http://cmnd.bashedu.ru/docs/lectures/CUDA/Lection%2001%20-%20Intro.pdf>

<https://proglib.io/p/cuda/>

5) <https://habr.com/post/55461/>

<https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#shared>

<http://qaru.site/questions/1198692/dynamic-shared-memory-in-cuda>

<http://ssd.sscc.ru/sites/default/files/content/attach/332/cuda-2-program-arch.pdf>

http://hpc.mipt.ru/wp-content/uploads/2012/11/Виды-памяти.pdf

6) <http://ssd.sscc.ru/sites/default/files/content/attach/332/cuda-3-arch.pdf>

<https://proglib.io/p/cuda/>

<https://indico-hlit.jinr.ru/event/6/attachments/104/129/talk1_intro2gpu.pdf>

7) <https://devblogs.nvidia.com/unified-memory-cuda-beginners/>

<https://streamhpc.com/blog/2013-11-14/cuda-6-unified-memory-explained/>

Унифицированная память - это одно адресное пространство памяти, доступное из любого процессора в системе (см. Рис. 1). Эта аппаратно-программная технология позволяет приложениям выделять данные, которые можно читать или записывать из кода, работающего на процессорах или графических процессорах. Выделение единой памяти так же просто, как замена вызовов malloc () или new на вызовы cudaMallocManaged (), функция распределения, которая возвращает указатель, доступный с любого процессора (в следующем случае ptr).

cudaError\_t cudaMallocManaged (void \*\* ptr, size\_t size);

Когда код, выполняющийся на процессоре или графическом процессоре, получает доступ к данным, которые распределены таким образом (часто называемые управляемыми данными CUDA), системное программное обеспечение CUDA и / или аппаратное обеспечение заботятся о переносе страниц памяти в память процессора доступа. Важным моментом здесь является то, что архитектура GPU Pascal является первой с аппаратной поддержкой для сбоев и миграции страниц виртуальной памяти через свой механизм миграции страниц. Старые GPU, основанные на архитектурах Kepler и Maxwell, также поддерживают более ограниченную форму Unified Memory.

8) <http://cmnd.bashedu.ru/docs/lectures/CUDA/Lection%2001%20-%20Intro.pdf>

<http://www.geeks3d.com/20100606/gpu-computing-nvidia-cuda-compute-capability-comparative-table/>

<http://www.ksmlab.ru/page/cuda-osnovy-primery>

9) <https://habr.com/post/151897/>

12) <https://wrf.ecse.rpi.edu/wiki/ParallelComputingSpring2014/nvidia/cuda6doc/html/kepler-tuning-guide/index.html#shared-memory-and-warp-shuffle>

13) <https://stackoverflow.com/questions/5041328/in-cuda-what-is-memory-coalescing-and-how-is-it-achieved>

<https://www.hotchips.org/wp-content/uploads/hc_archives/hc20/1_Sun/HC20.24.240.pdf>