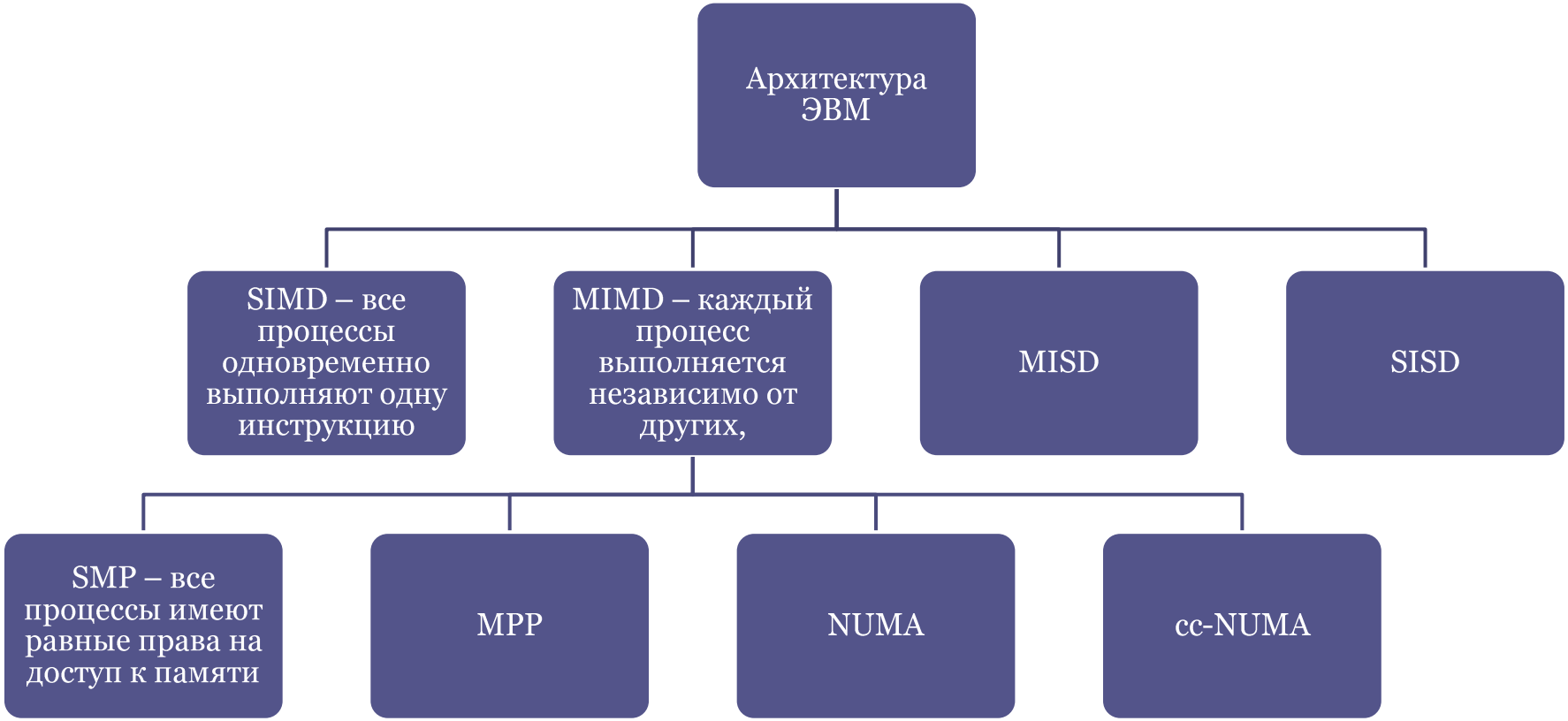
Модель исполнения SIMT

Как реализовать выполнение миллионов нитей на имеющейся архитектуре?

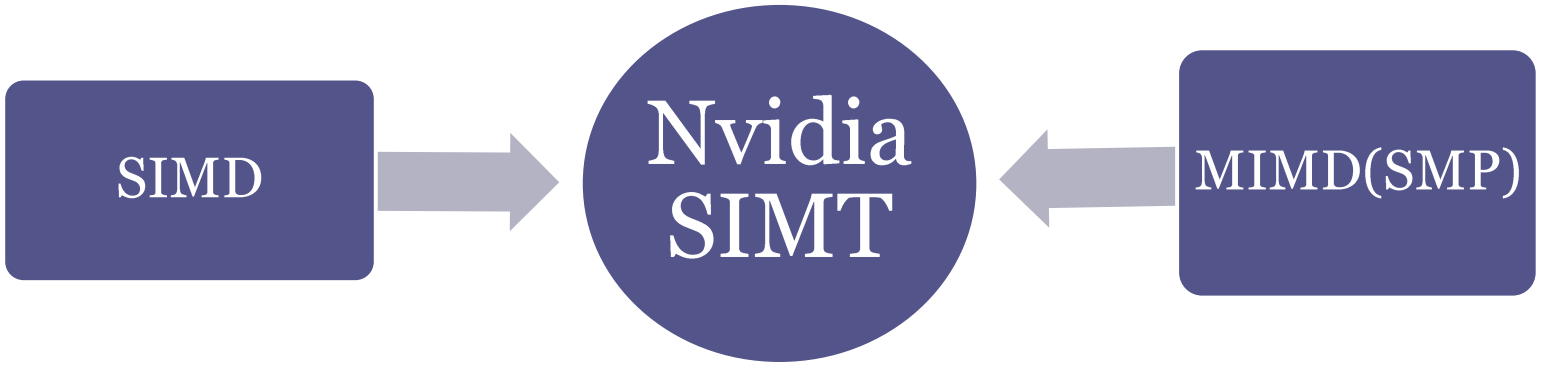
CUDA и классификация Флинна



CUDA и классификация Флинна

- У Nvidia собственная модель исполнения, имеющая черты как SIMD, так и MIMD:

- **Nvidia SIMT: Single Instruction – Multiple Thread**

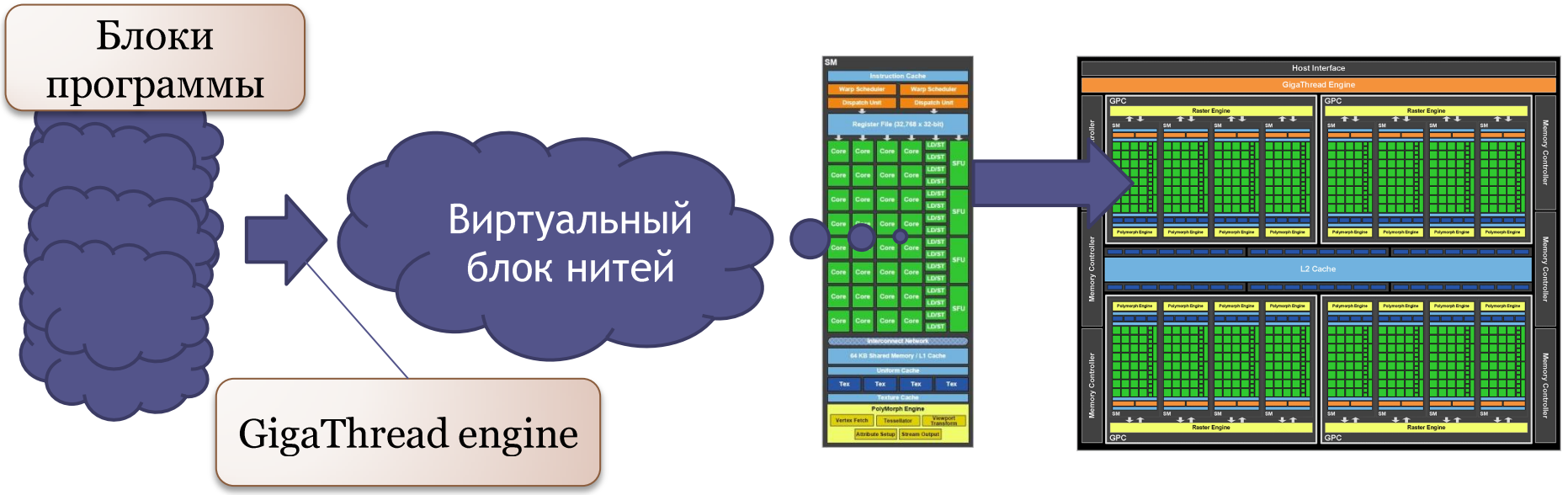


SIMT: виртуальные нити, блоки

- Виртуально все нити:
  - выполняются параллельно (MIMD)
  - Имеют одинаковые права на доступ к памяти (MIMD :SMP)
- Нити разделены на группы одинакового размера (блоки):
  - В общем случае (есть исключение) , глобальная синхронизация всех нитей невозможна, нити из разных блоков выполняются полностью независимо
  - Есть локальная синхронизация внутри блока, нити из одного блока могут взаимодействовать через специальную память
- Нити не мигрируют между блоками. Каждая нить находится в своём блоке с начала выполнения и до конца.

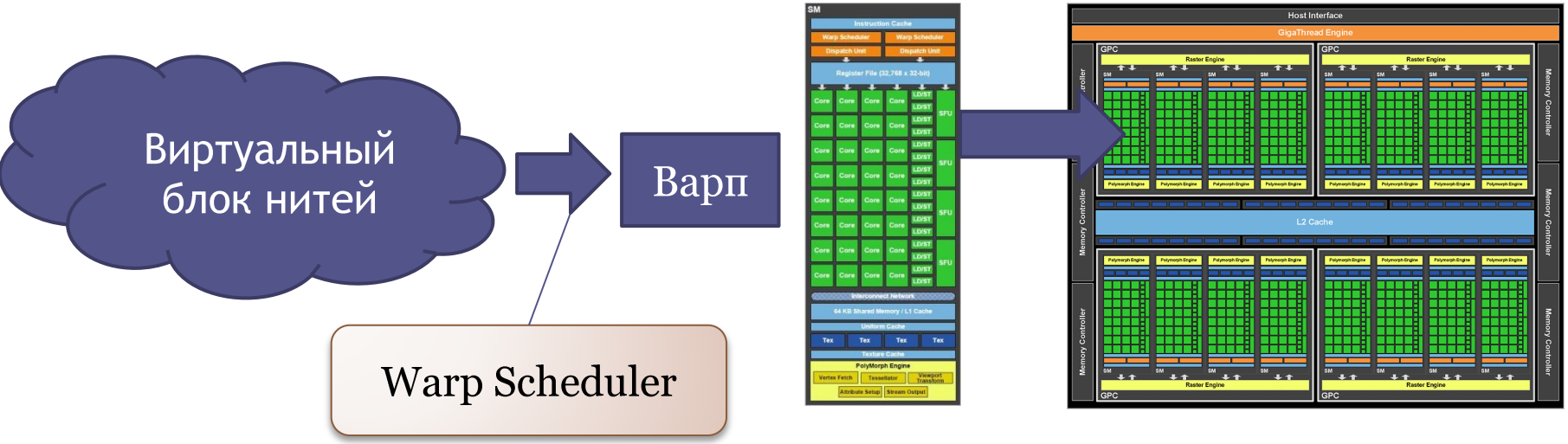
SIMT: аппаратное выполнение

- Все нити из одного блока выполняются на одном мультипроцессоре (SM)
- Максимальное число нитей в блоке – 1024
- Блоки не мигрируют между SM
- Распределение блоков по мультипроцессорам непредсказуемо
- Каждый SM работает независимо от других



Блоки и варпы

- Блоки нитей по фиксированному правилу разделяются на группы по 32 нити, называемые варпами (warp)
- Все нити варпа одновременно выполняют одну общую инструкцию (в точности SIMD-выполнение)
- Warp Scheduler на каждом цикле работы выбирает варп, все нити которого готовы к выполнению следующей инструкции и запускает весь варп



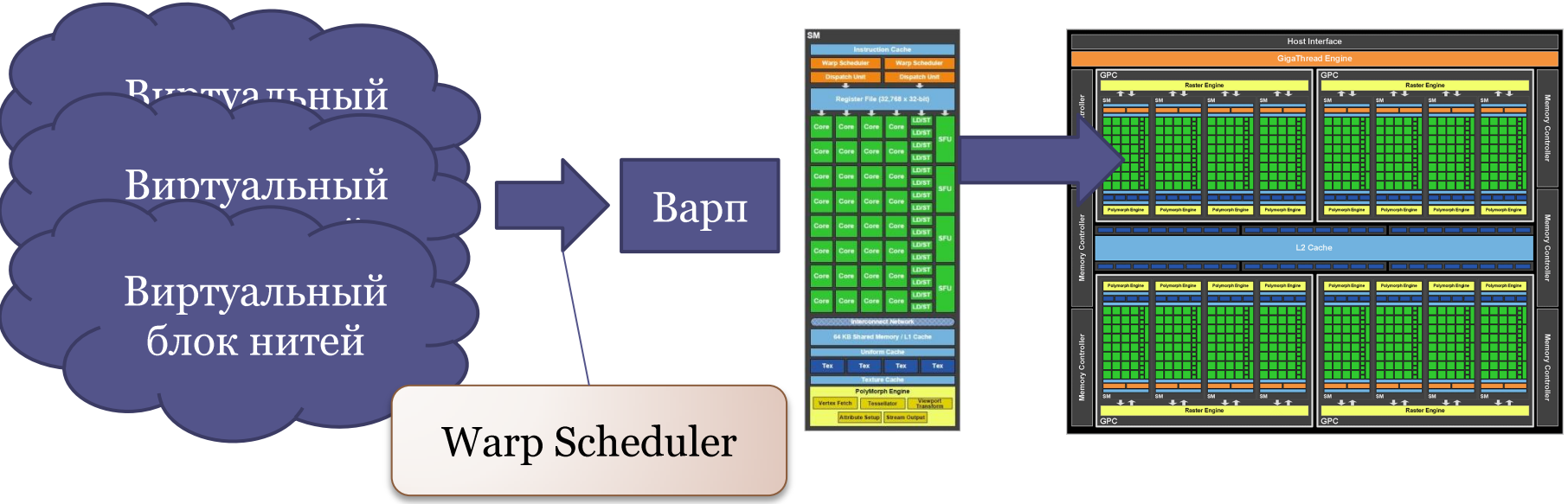
Ветвление (branching)

- Все нити варпа одновременно выполняют одну и ту же инструкцию.
- Как быть, если часть нитей эту инструкцию выполнять не должна?
  - if(<условие>), где значение условия различается для нитей одного варпа

Эти нити «замаскируются» нулями в специальном наборе регистров и не будут её выполнять, т.е. будут простаивать

Несколько блоков на одном SM

- SM может работать с варпами нескольких блоков одновременно
  - Максимальное число резидентных блоков на одном мультипроцессоре – 8
  - Максимальное число резидентных варпов – 48 = 1536 нитей



Загруженность (Occupancy)

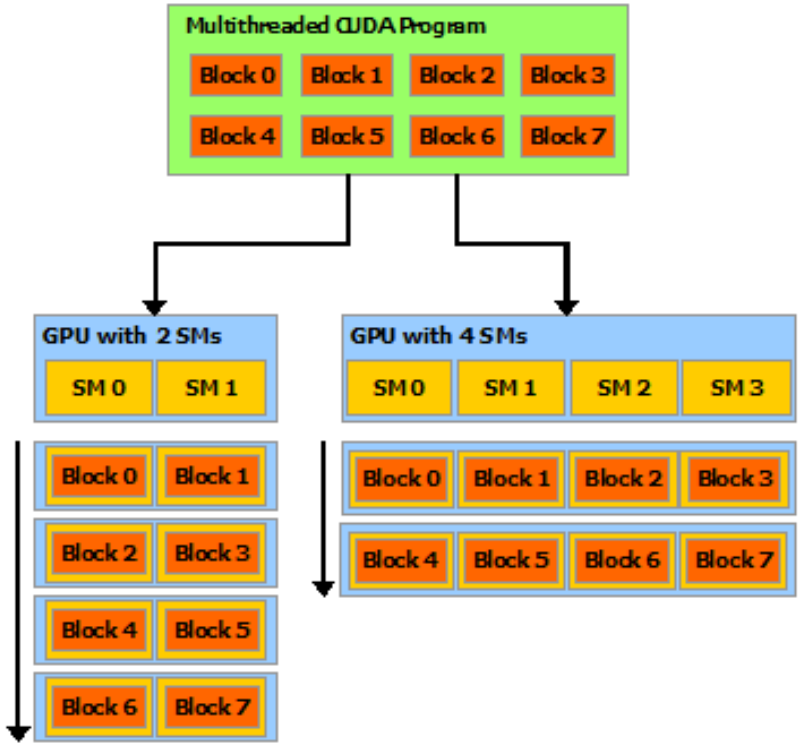
- Чем больше нитей активно на мультипроцессоре, тем эффективнее используется оборудование
  - Блоки по 1024 нити – 1 блок на SM, 1024 нити, 66% от максимума
  - Блоки по 100 нитей – 8 блоков на SM, 800 нитей, 52%
  - Блоки по 512 нитей – 3 блока на SM, 1536 нитей, 100%

SIMT и глобальная синхронизация

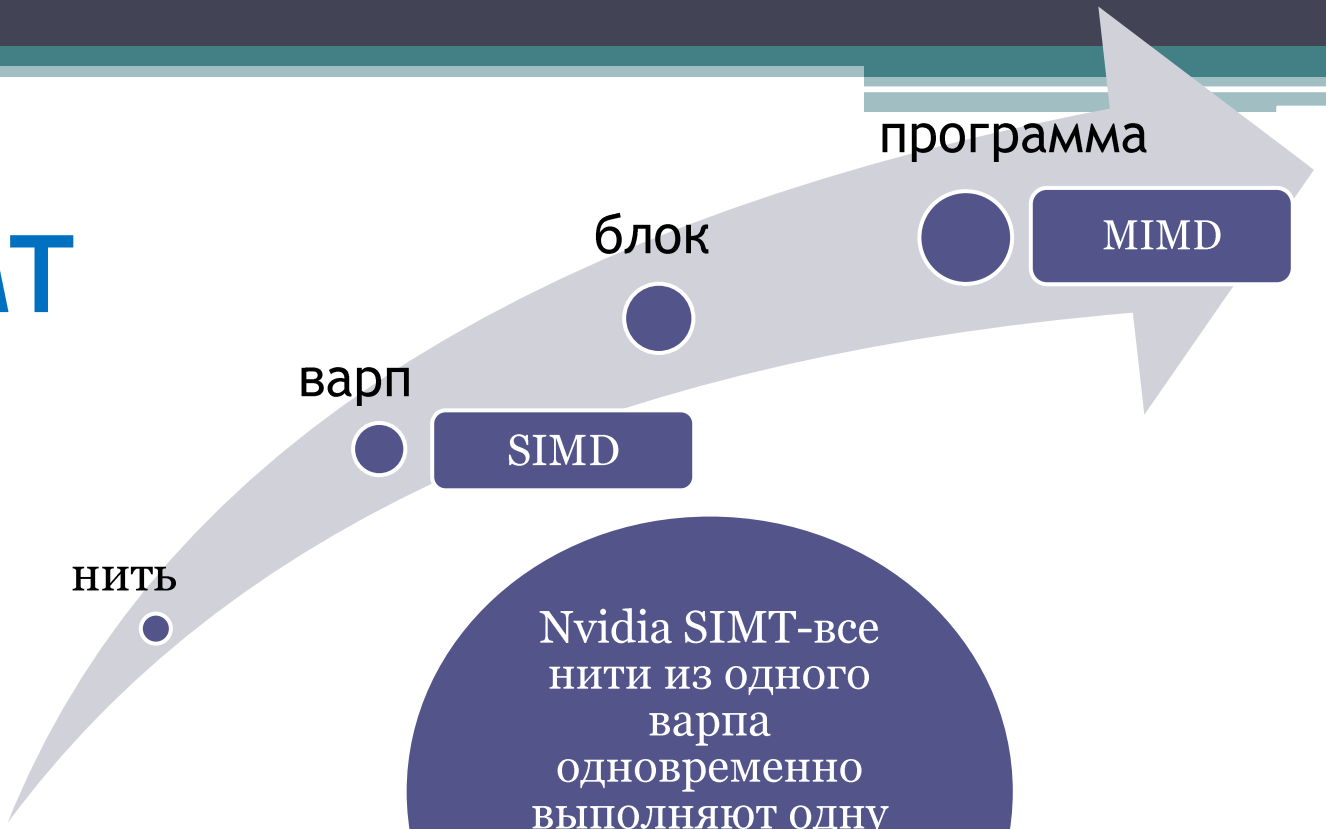
- В общем случае, из-за ограничений по числу нитей и блоков на одном SM, не удаётся разместить сразу все блоки программы на GPU
  - Часть блоков ожидает выполнения
    - Поэтому в общем случае невозможна глобальная синхронизация
  - Блоки выполняются по мере освобождения ресурсов
    - Нельзя предсказать порядок выполнения блоков
- Глобальная синхронизация частично возможна через атомарные операции
    - Вручную, специальная техника «Persistent Threads»

SIMT и масштабирование

- Виртуальное
  - GPU может поддерживать миллионы виртуальных нитей
  - Виртуальные блоки независимы
    - Программу можно запустить на любом количестве SM
- Аппаратное
  - Мультипроцессоры независимы
    - Можно «нарезать» GPU с различным количеством SM



SIMT



Nvidia SIMT-все нити из одного варпа одновременно выполняют одну инструкцию, варпы выполняются независимо

SIMD – все нити одновременно выполняют одну инструкцию

MIMD – каждая нить выполняется независимо от других, SMP – все нити имеют равные возможности для доступа к памяти

Выводы

Хорошо распараллеливаются на GPU задачи, которые:

- Имеют параллелизм по данным
  - Одна и та же последовательность вычислений, применяемая к разным данным
- Могут быть разбиты на подзадачи одинаковой сложности
  - подзадача будет решаться блоком нитей
- Каждая подзадача может быть выполнена независимо от всех остальных
  - нет потребности в глобальной синхронизации

Выводы

Хорошо распараллеливаются на GPU задачи, которые:

- Число арифметических операций велико по сравнению с операциями доступа в память
  - для покрытия латентности памяти вычислениями
- Если алгоритм итерационный, то его выполнение может быть организовано без пересылок памяти между хостом и GPU после каждой итерации
  - Пересылки данных между хостом и GPU накладны