Оптимизация

Передача данных Host / Device

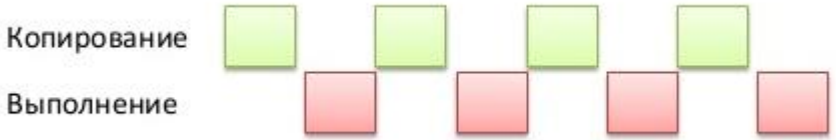
- Является дорогостоящей операцией
- Возможна асинхронная передача

cudaMemcpy()

cudaMemcpyAsync()

- Синхронная передача

- Асинхронная передача



Типы памяти устройства

- Streaming Multiprocessor
  - Регистровая память
  - Разделяемая память
  - Константная память
- DDRAM
  - Локальная память
  - Глобальная память
- Текстурный блок
  - Текстурная память

Объединенное чтение DRAM

- Выравнивание исходных данных по границе слова

- Потоки warp-а должны осуществлять одновременное чтение DDRAM

Разделяемая память и конфликты

- Общая для всех потоков блока
- Распределяется между блоками
- Разбивается на банки (32-битные слова)

Float vs Double

- Арифметика с float числами быстрее чем с double числами

Использовать “f” при объявлении числовых констант (3.14f)

Умножение / степень

- По возможности использовать оператор сдвига

- Для известных целых значений степеней использовать явное перемножение вместо вызова pow()

Аппаратные аналоги функций

__sinf()	sinf()
__cosf()	cosf()
__expf()	expf()

…

Операторы ветвления

Операторы управления потоком команд (if, switch, for, while, do-while) отрицательно сказываются на производительности

следить за исполнением внутри варпа, в идеале все нити идут по одному пути, иначе возможно последовательное выполнение

Процесс оптимизации

Определить что ограничивает производительность

- Пропускная способность памяти
- Исполнение инструкций
- Задержка
- Комбинации

Процесс оптимизации

Исследовать ограничители в порядке важности

- Насколько эффективно работает
- Анализ и нахождение проблем
- Применить оптимизации

Анализ через модификацию кода

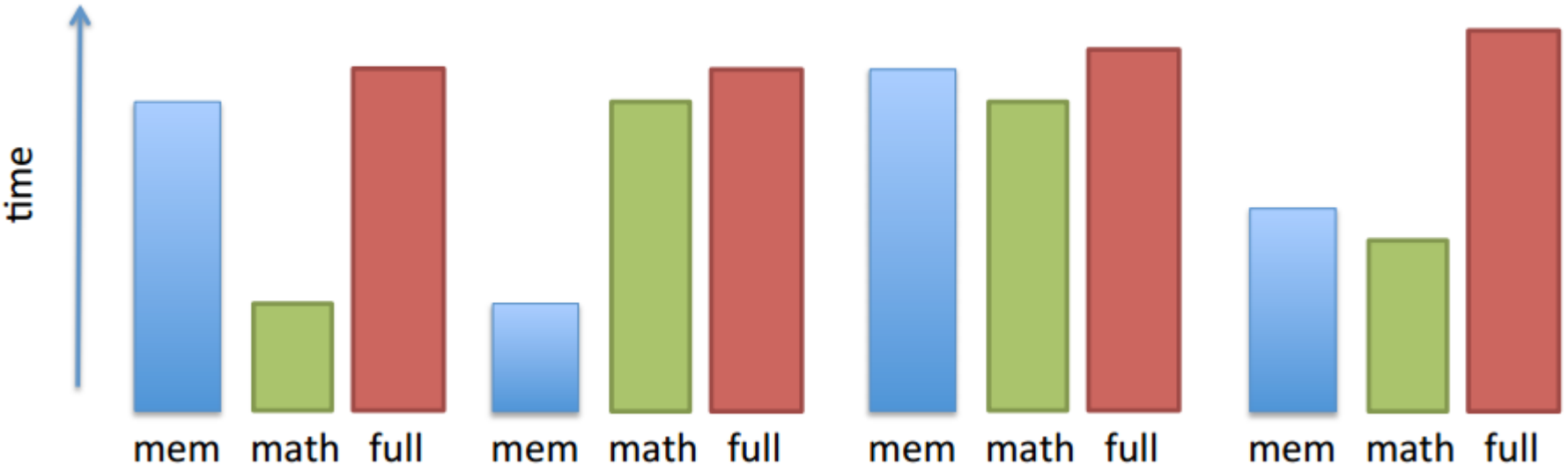
Измеряем время работы 2-х модификаций:

- Минимум инструкций, только доступ к памяти(mem)
- Минимум обращений к памяти, только инструкции(math)

Удобно для алгоритмов не использующих зависимость по данным для ветвления или адресации

Сравниваем время работы модифицированных ядер

- Помогает определить во что мы упираемся
- Насколько хорошо идет перекрытие арифметических операций и чтений из памяти



Ограничитель – память

Хорошее перекрытие математики и памяти:

нет проблем с задержками

Ограничитель – инструкции

Хорошее перекрытие математики и памяти:

нет проблем с задержками

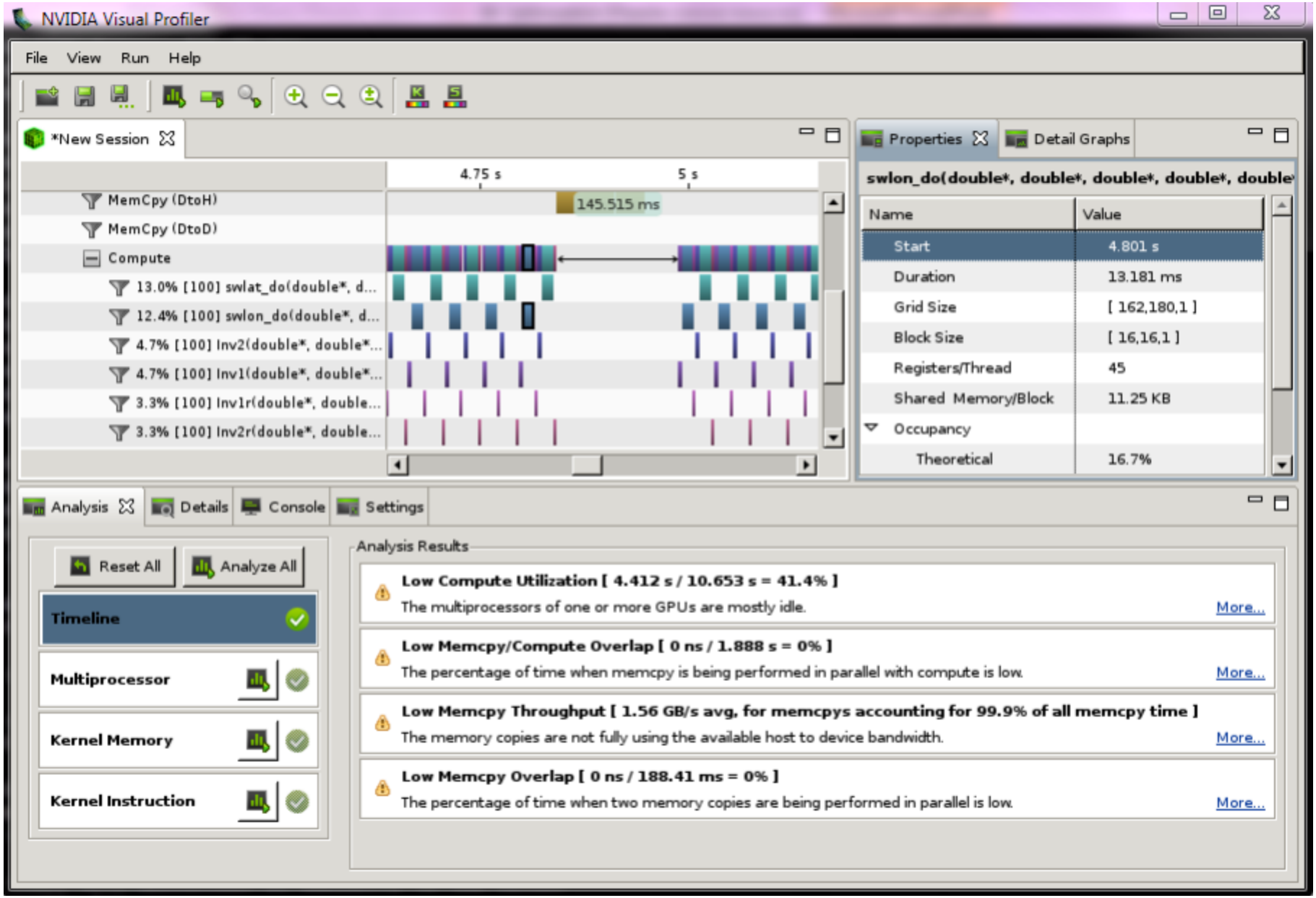
Баланс

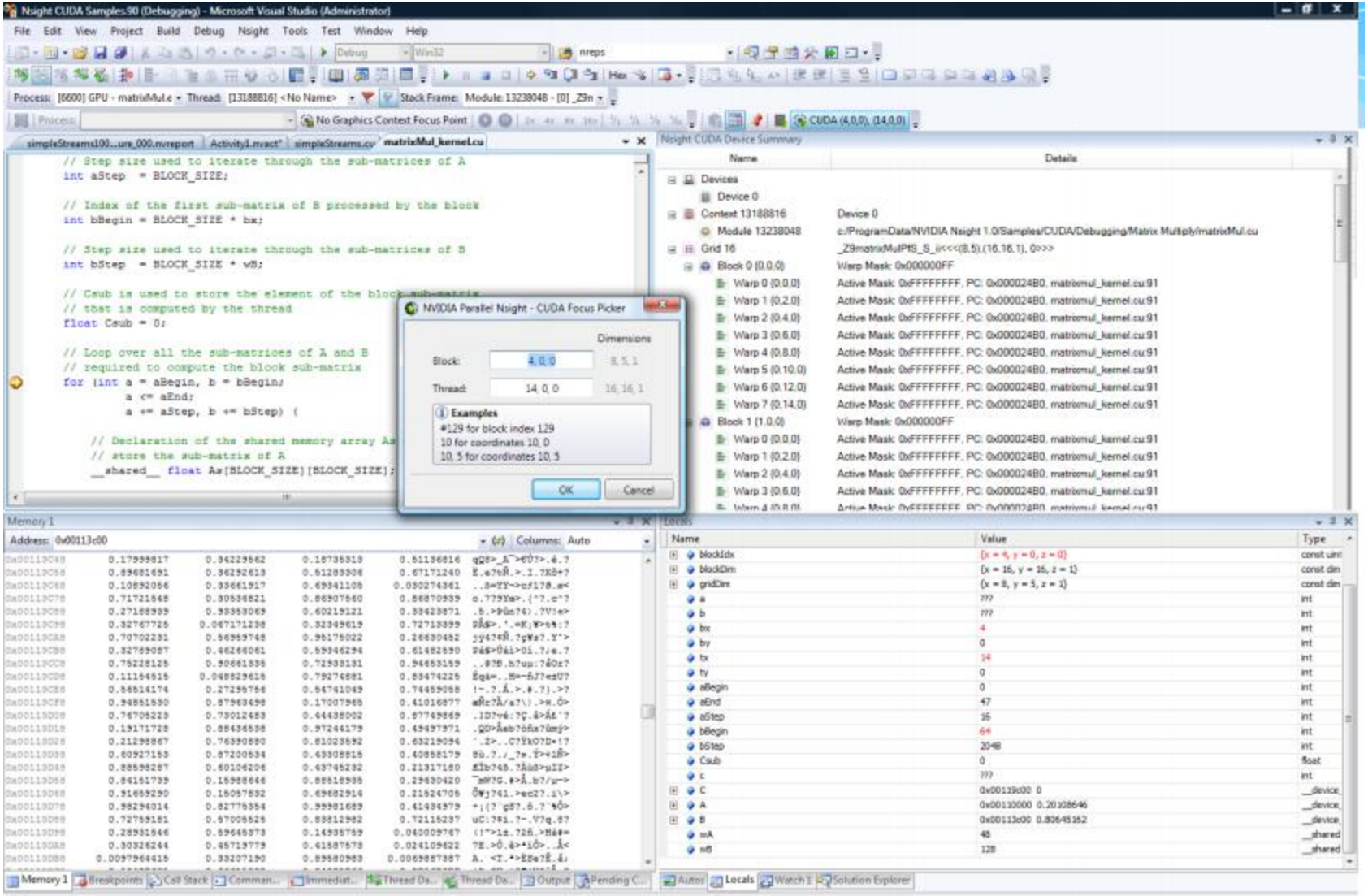
Хорошее перекрытие математики и памяти:

нет проблем с задержками

Плохое перекрытие математики и памяти:

проблемы с задержкамивремя


NVIDIA Visual Profiler
File View Run Help
*New Session
4.75 s
5 s
145.515 ms
MemCpy (DtoH)
MemCpy (DtoD)
Compute
13.0% [100] swlat_do(double*, d...
12.4% [100] swlon_do(double*, d...
4.7% [100] Inv2(double*, double*...
4.7% [100] Inv1(double*, double*...
3.3% [100] Inv1r(double*, double...
3.3% [100] Inv2r(double*, double...
Properties
Detail Graphs
swlon_do(double*, double*, double*, double*, double
Name
Value
Start
4.801 s
Duration
13.181 ms
Grid Size
[162,180,1]
Block Size
[16,16,1]
Registers/Thread
45
Shared Memory/Block
11.25 KB
Occupancy
Theoretical
16.7%
Analysis
Details
Console
Settings
Reset All
Analyze All
Timeline
Multiprocessor
Kernel Memory
Kernel Instruction
Analysis Results
Low Compute Utilization [4.412 s / 10.653 s = 41.4%]
The multiprocessors of one or more GPUs are mostly idle.
More...
Low Memcpy/Compute Overlap [0 ns / 1.888 s = 0%]
The percentage of time when memcpy is being performed in parallel with compute is low.
More...
Low Memcpy Throughput [1.56 GB/s avg, for memcpys accounting for 99.9% of all memcpy time]
The memory copies are not fully using the available host to device bandwidth.
More...
Low Memcpy Overlap [0 ns / 188.41 ms = 0%]
The percentage of time when two memory copies are being performed in parallel is low.
More...


Nsight CUDA Samples:90 (Debugging) - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Nsight Tools Test Window Help
simpleStreams100...ure_000.nvreport
Activity1.nvact*
simpleStreams.cu
matrixMul_kernel.cu
// Step size used to iterate through the sub-matrices of A
int aStep = BLOCK_SIZE;
// Index of the first sub-matrix of B processed by the block
int bBegin = BLOCK_SIZE * bx;
// Step size used to iterate through the sub-matrices of B
int bStep = BLOCK_SIZE * wB;
// Csub is used to store the element of the block sub-matrix
// that is computed by the thread
float Csub = 0;
// Loop over all the sub-matrices of A and B
// required to compute the block sub-matrix
for (int a = aBegin, b = bBegin;
a <= aEnd;
a += aStep, b += bStep) {
// Declaration of the shared memory array As
// store the sub-matrix of A
__shared__ float As[BLOCK_SIZE][BLOCK_SIZE];
NVIDIA Parallel Nsight - CUDA Focus Picker
Dimensions
Block:
4, 0, 0
8, 5, 1
Thread:
14, 0, 0
16, 16, 1
Examples
#129 for block index 129
10 for coordinates 10, 0
10, 5 for coordinates 10, 5
OK
Cancel
Nsight CUDA Device Summary
Name
Details
Devices
Device 0
Context 13188816
Device 0
Module 13238048
c:/ProgramData/NVIDIA Nsight 1.0/Samples/CUDA/Debugging/Matrix Multiply/matrixMul.cu
Grid 16
_Z9matrixMulPfS_S_ii<<<(8,5),(16,16,1), 0>>>
Block 0 (0,0,0)
Warp Mask: 0x000000FF
Warp 0 (0,0,0)
Active Mask: 0xFFFFFFFF, PC: 0x000024B0, matrixmul_kernel.cu:91
Warp 1 (0,2,0)
Warp 2 (0,4,0)
Warp 3 (0,6,0)
Warp 4 (0,8,0)
Warp 5 (0,10,0)
Warp 6 (0,12,0)
Warp 7 (0,14,0)
Block 1 (1,0,0)
Warp 0 (0,0,0)
Warp 1 (0,2,0)
Warp 2 (0,4,0)
Warp 3 (0,6,0)
Memory 1
Address: 0x00113c00
Columns: Auto
Locals
Name
Value
Type
blockIdx
{x = 4, y = 0, z = 0}
uint3
blockDim
{x = 16, y = 16, z = 1}
dim3
gridDim
{x = 8, y = 5, z = 1}
dim3
a
???
int
b
???
int
bx
4
int
by
0
int
tx
14
int
ty
0
int
aBegin
0
int
aEnd
47
int
aStep
16
int
bBegin
64
int
bStep
2048
int
Csub
0
float
c
???
int
C
0x00113c00 0
__device__
A
0x00110000 0.20108646
__device__
B
0x00113c00 0.80645162
__device__
wA
48
__shared__
wB
128
__shared__
Memory 1
Breakpoints
Call Stack
Comman...
Immediat...
Thread De...
Thread De...
Output
Pending C...
Autos
Locals
Watch 1
Solution Explorer

Счетчики профайлера

- timestamp
- gridsize
- threadblocksize
- dynsmemperblock
- stasmemperblock
- regperthread
- memtransferdir
- memtransfersize
- Streamid
- gputime
- cputime
- Occupancy
- gld_incoherent
- gld_coherent
- gld_32b / gst_32b
- gld_64b / gst_62b
- gld_128b / gst_128b
- gld_request
- gst_incoherent
- gst_coherent
- gst_request
- local_load
- local_store
- branch
- divergent_branch
- instructions
- warp_serialize
- cta_launched

- Значение имеют не цифры, а их приращение и соотношение

- Для ядер нужно стремиться чтобы стремились к нулю непоследовательные обращения к памяти

(gld_incoherent, gld_coherent, gst_incoherent, gst_coherent)

Арифметика - Память

- Оптимальное соотношение для Tesla C2050

инструкции : байты

~ 3.6 : 1, float при включенном ECC

~ 4.5 : 1, float при выключенном ECC

Счетчик профилировщика (на варп)

– сколько инструкций исполнилось

Instructions executed

– Конфликты банков разделяемой памяти

l1_shared_bank_conflict,

shared_load, shared_store

Конфликты существенны если:

l1_shared_bank_conflict >> (shared_load+ shared_store)

l1_shared_bank_conflict >> instructions_issued

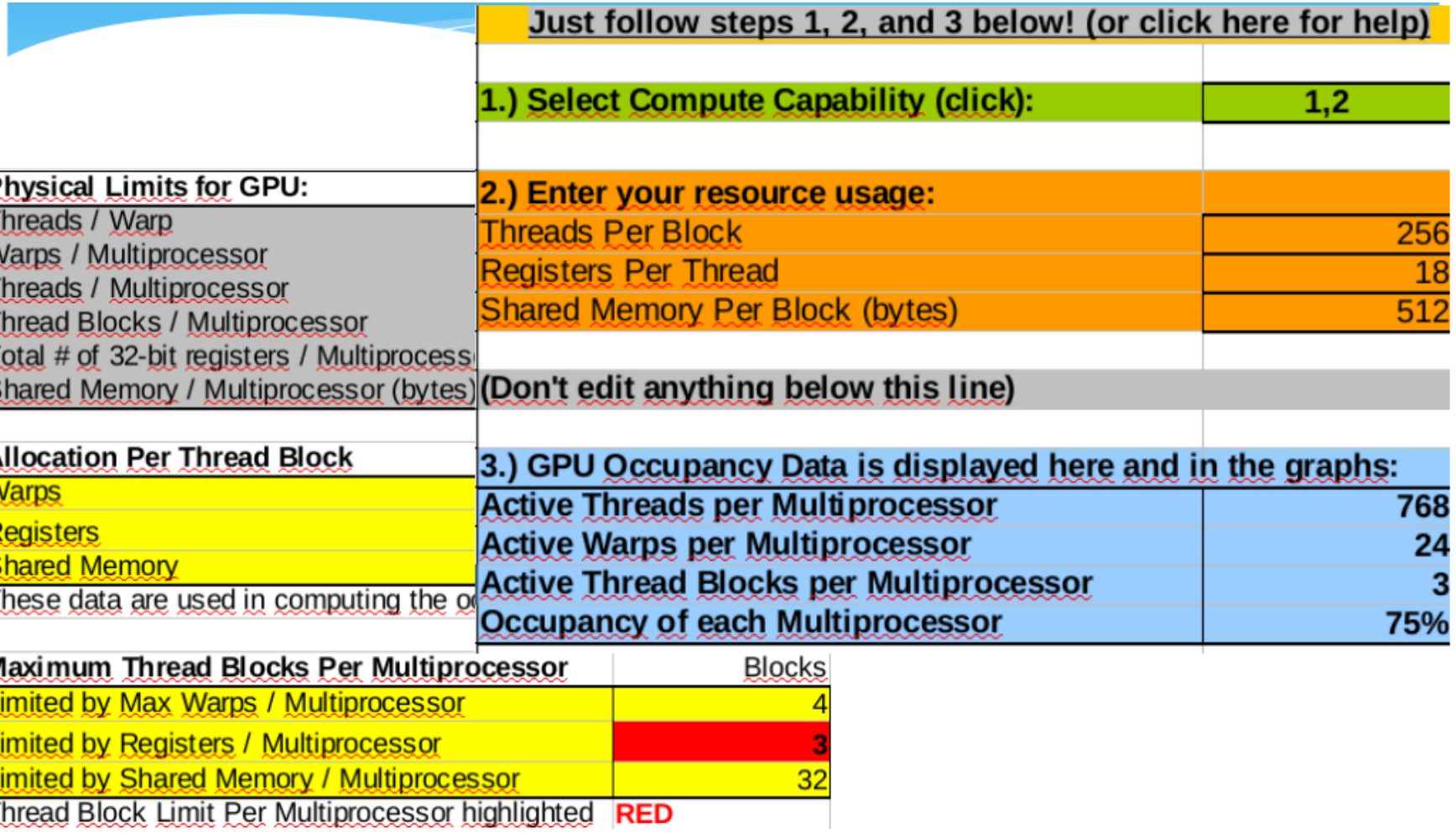
Спиллинг регистров

Достигнут предел доступных регистров, компилятор начинает использовать локальную память

- Дополнительная нагрузка шины памяти
- Дополнительные инструкции
- Не всегда проблема

счетчики l1_local_load_hit, l1_local_load_miss

CUDA Occupancy calculator



	Just follow steps 1, 2, and 3 below! (or click here for help)	
	1.) Select Compute Capability (click):	1,2
Physical Limits for GPU:	2.) Enter your resource usage:	
Threads / Warp	Threads Per Block	256
Warps / Multiprocessor	Registers Per Thread	18
Threads / Multiprocessor	Shared Memory Per Block (bytes)	512
Thread Blocks / Multiprocessor		
Total # of 32-bit registers / Multiprocess		
Shared Memory / Multiprocessor (bytes)	(Don't edit anything below this line)	
Allocation Per Thread Block	3.) GPU Occupancy Data is displayed here and in the graphs:	
Warps	Active Threads per Multiprocessor	768
Registers	Active Warps per Multiprocessor	24
Shared Memory	Active Thread Blocks per Multiprocessor	3
These data are used in computing the o	Occupancy of each Multiprocessor	75%

Maximum Thread Blocks Per Multiprocessor	Blocks
Limited by Max Warps / Multiprocessor	4
Limited by Registers / Multiprocessor	3
Limited by Shared Memory / Multiprocessor	32
Thread Block Limit Per Multiprocessor highlighted	RED