2014 Conference on Advanced Topics and Auto Tuning in High Performance Scientific Computing

Mathematics Research Center
National Taiwan University
March 14-15, 2014

Some Techniques for Optimizing CUDA More

http://web.eecs.utk.edu/~kurzak/talks/ATAT14

Jakub Kurzak

Innovative Computing Laboratory **(CUDA Center of Excellence)**
Electrical Engineering and Computer Science
University of Tennessee

Contents

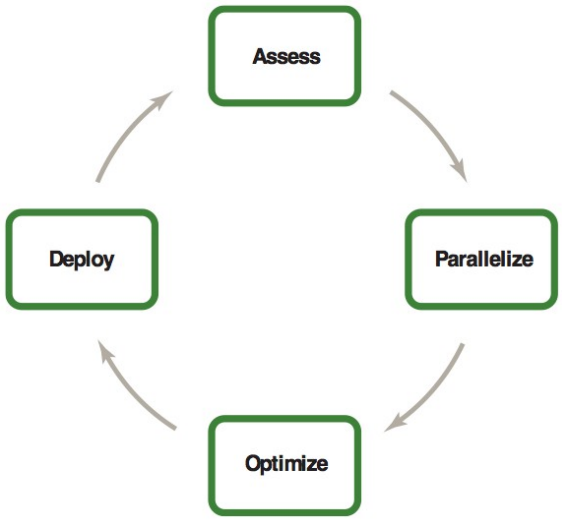
Show 5 different versions of SGEMM

Introduce performance improvements

- using texture caches
- using vector types

Show performance diagnostic tools

- look at PTX
- look at ASM
- look at hardware counters



Performance Optimization

absolute basics

Minimize Thread Divergence

- all threads branch together

Avoid Warp Serialization

- each thread reads a different shared memory bank
- all threads read the same physical memory address

Optimize Global Memory Access

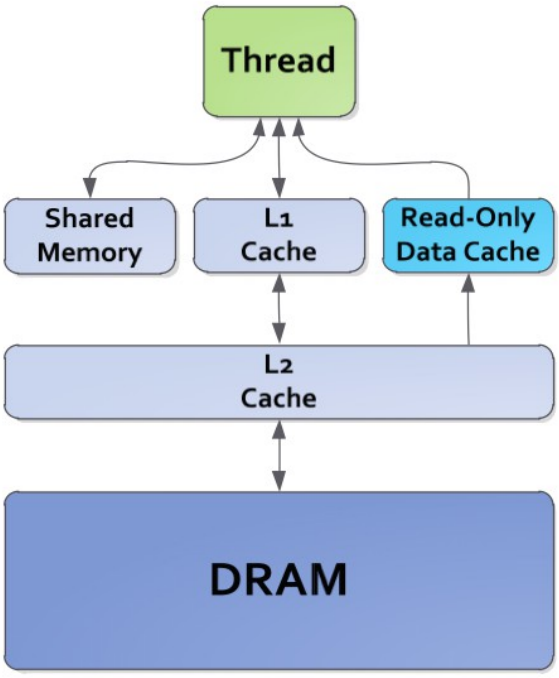
- access is sequential and aligned

Maximize Occupancy

- number of threads is much higher than number of CUDA cores

Texture Cache

read-only data cache



Offers performance advantages.

Can be used through:

- texture reads
- `__LDG()` intrinsic (compute capability 3.5)

Vector Types

Type	Alignment
int1, uint1	4
int2, uint2	8
int3, uint3	4
int4, uint4	16
float1	4
float2	8
float3	4
float4	16
double1	8
double2	16

In principle a GPU is a SIMT device.

It is not really a SIMD device.

But, vector operations allow for:

- better texture reads
- better shared emory access

Hardware Counters

events

tex0_cache_sector_queries	Number of texture cache 0 requests.
tex1_cache_sector_queries	Number of texture cache 1 requests.
tex2_cache_sector_queries	Number of texture cache 2 requests.
tex3_cache_sector_queries	Number of texture cache 3 requests.
l2_subp0_write_sector_misses	Number of write misses in slice 0 of L2 cache.
l2_subp1_write_sector_misses	Number of write misses in slice 1 of L2 cache.
l2_subp2_write_sector_misses	Number of write misses in slice 2 of L2 cache.
l2_subp3_write_sector_misses	Number of write misses in slice 3 of L2 cache.
warps_launched	Number of warps launched on a multiprocessor.
threads_launched	Number of threads launched on a multiprocessor.
shared_load	Number of executed load instructions...
shared_store	Number of executed store instructions...

- raw numbers
- many are cryptic
- **some are clear**

Hardware Counters

metrics

sm_efficiency:	The percentage of time at least one warp is active
achieved_occupancy:	Average nuber of active warps to maximum number of warps
ipc:	Instructions executed per cycle
issued_ipc:	Instructions issued per cycle
shared_load_throughput:	Shared memory load throughput
shared_store_throughput:	Shared memory store throughput
flops_sp:	SP floating point operations executed
flops_sp_add:	SP floating point add operations executed
flops_sp_mul:	SP floating point multiply operations executed
flops_sp_fma:	SP floating point multiply accumulate operations executed

- some are raw numbers
- some are processed numbers
- most are straightforward

Hardware Counters

collecting with NVPROF

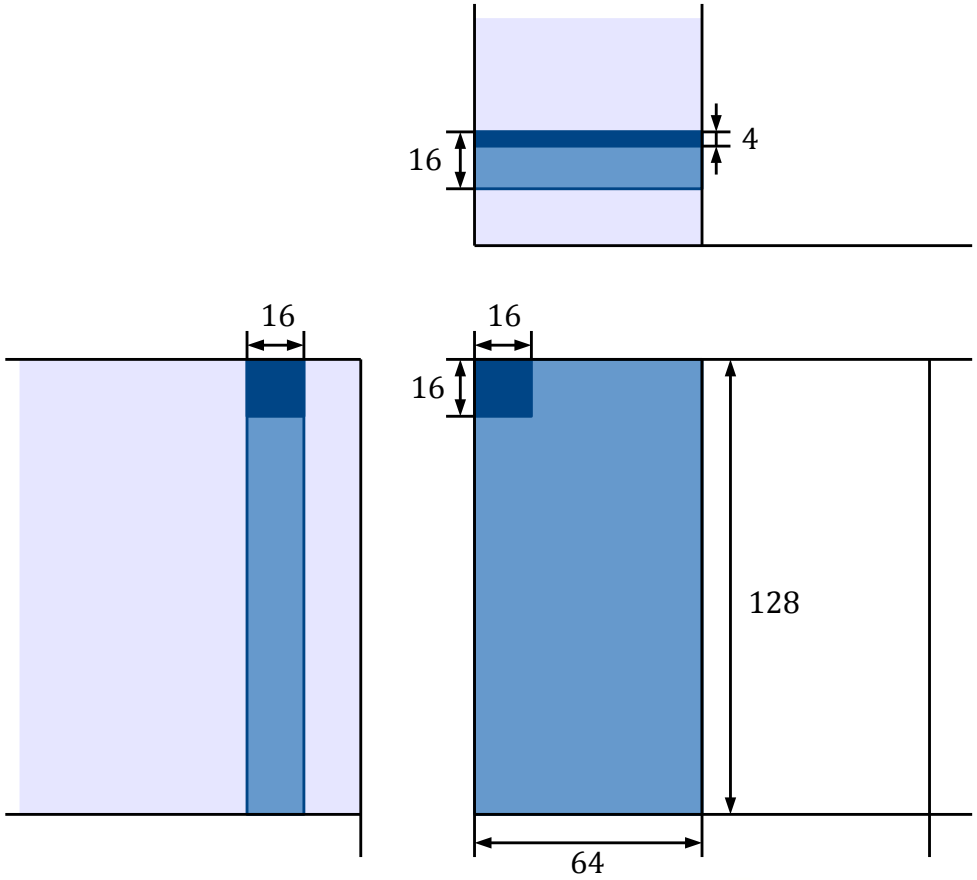
```
cudaProfilerStart();

cublas_stat = cublasSgemm(
    cublas_handle,
    CUBLAS_OP_N, CUBLAS_OP_N,
    m, n, k,
    &alpha, d_A, lda,
            d_B, ldb,
    &beta,  d_C, ldc);

cudaProfilerStop();
```

All counters cannot be collected at one time. If requesting many counters, the kernel will be replayed as many times as necessary to collect all requested counters.

```
> nvprof   --profile-from-start off   --csv   --events warps_launched   a.out

> nvprof   --profile-from-start off   --csv   -metrics achieved_occupancy   a.out
```

SGEMM Parameters

hand picked




```
BLK_M = 128
BLM_N =  64
BLK_K =  16

DIM_X =  16
DIM_Y =  16

DIM_XA = 16
DIM_YA = 16

DIM_XB =  4
DIM_YB = 64
```

```
extern "C" __global__
void beast_gemm_kernel(
    int M, int N, int K,
    float alpha, float *A, int lda,
                 float *B, int ldb,
     float beta, float *C, int ldc)
{



    __shared__ float sA[BLK_K][BLK_M]; // shared memory for A
    __shared__ float sB[BLK_N][BLK_K]; // shared memory for B
    float rC[THR_N][THR_M];            // registers for C

    int coord_A = (blx*BLK_M     + idyA*lda) + idxA;
    int coord_B = (bly*BLK_N*ldb + idyB*ldb) + idxB;



    for (kk = 0; kk < K; kk += BLK_K)
    {
        #pragma unroll
        for (n = 0; n < BLK_K; n += DIM_YA)
            #pragma unroll
            for (m = 0; m < BLK_M; m += DIM_XA)
                sA[n+idyA][m+idxA] = A[coord_A + (n*lda+m)];

        #pragma unroll
        for (n = 0; n < BLK_N; n += DIM_YB)
            #pragma unroll
            for (m = 0; m < BLK_K; m += DIM_XB)
                sB[n+idyB][m+idxB] = B[coord_B + (n*ldb+m)];

        __syncthreads();

        #pragma unroll
        for (k = 0; k < BLK_K; k++)
            #pragma unroll
            for (n = 0; n < THR_N; n++)
                #pragma unroll
                for (m = 0; m < THR_M; m++)
                    rC[n][m] += sA[k][m*DIM_X+idx] * sB[n*DIM_Y+idy][k];

        __syncthreads();

        coord_A += BLK_K*lda;
        coord_B += BLK_K;
    }


}
```

```
int idx = threadIdx.x;      // thread's m position in C
int idy = threadIdx.y;      // thread's n position in C
int idt = DIM_X*idy + idx;  // thread's number
int blx = blockIdx.x;       // block's m position
int bly = blockIdx.y;       // block's n position
int idxA = idt % DIM_XA;    // thread's m position for loading A
int idyA = idt / DIM_XA;    // thread's n position for loading A
int idxB = idt % DIM_XB;    // thread's m position for loading B
int idyB = idt / DIM_XB;    // thread's n position for loading B
int m, n, k, kk;            // loop counters
```

```
#pragma unroll
for (n = 0; n < THR_N; n++)
    #pragma unroll
    for (m = 0; m < THR_M; m++)
        rC[n][m] = 0.0;
```

```
#pragma unroll
for (n = 0; n < THR_N; n++) {
    int coord_dCn = bly*BLK_N + n*DIM_Y+idy;
    #pragma unroll
    for (m = 0; m < THR_M; m++) {
        int offsC = coord_dCn*ldc + blx*BLK_M;
        C[offsC + m*DIM_X+idx] =
            alpha*rC[n][m] + beta*C[offsC + m*DIM_X+idx];
    }
}
```

```
for (kk = 0; kk < K; kk += BLK_K)
{
    #pragma unroll
    for (n = 0; n < BLK_K; n += DIM_YA)
        #pragma unroll
        for (m = 0; m < BLK_M; m += DIM_XA)
            sA[n+idyA][m+idxA] = A[coord_A + (n*lda+m)];

    #pragma unroll
    for (n = 0; n < BLK_N; n += DIM_YB)
        #pragma unroll
        for (m = 0; m < BLK_K; m += DIM_XB)
            sB[n+idyB][m+idxB] = B[coord_B + (n*ldb+m)];

    __syncthreads();

    #pragma unroll
    for (k = 0; k < BLK_K; k++)
        #pragma unroll
        for (n = 0; n < THR_N; n++)
            #pragma unroll
            for (m = 0; m < THR_M; m++)
                rC[n][m] +=
                    sA[k][m*DIM_X+idx] *
                    sB[n*DIM_Y+idy][k];

    __syncthreads();

    coord_A += BLK_K*lda;
    coord_B += BLK_K;
}
```

```
shl.b32         %r36, %r13, 4;
mad.lo.s32      %r37, %r36, %r55, %r6;
mul.wide.s32    %rd12, %r37, 4;
add.s64         %rd13, %rd2, %rd12;
mul.wide.s32    %rd14, %r2, 512;
add.s64         %rd16, %rd9, %rd14;
mul.wide.s32    %rd17, %r1, 4;
add.s64         %rd18, %rd16, %rd17;
ld.global.f32   %f163, [%rd13];
st.shared.f32   [%rd18], %f163;
ld.global.f32   %f164, [%rd13+64];
st.shared.f32   [%rd18+64], %f164;
ld.global.f32   %f165, [%rd13+128];
st.shared.f32   [%rd18+128], %f165;
ld.global.f32   %f166, [%rd13+192];
st.shared.f32   [%rd18+192], %f166;
ld.global.f32   %f167, [%rd13+256];
st.shared.f32   [%rd18+256], %f167;
ld.global.f32   %f168, [%rd13+320];
st.shared.f32   [%rd18+320], %f168;
ld.global.f32   %f169, [%rd13+384];
st.shared.f32   [%rd18+384], %f169;
ld.global.f32   %f170, [%rd13+448];
st.shared.f32   [%rd18+448], %f170;
shl.b32         %r38, %r55, 4;
add.s32         %r39, %r7, %r38;
mul.wide.s32    %rd19, %r39, 4;
add.s64         %rd20, %rd1, %rd19;
mul.wide.s32    %rd21, %r4, 64;
add.s64         %rd23, %rd11, %rd21;
mul.wide.s32    %rd24, %r3, 4;
add.s64         %rd25, %rd23, %rd24;
ld.global.f32   %f171, [%rd20];
st.shared.f32   [%rd25], %f171;
ld.global.f32   %f172, [%rd20+16];
st.shared.f32   [%rd25+16], %f172;
ld.global.f32   %f173, [%rd20+32];
st.shared.f32   [%rd25+32], %f173;
ld.global.f32   %f174, [%rd20+48];
st.shared.f32   [%rd25+48], %f174;
bar.sync        0;
```

```
MOV         R4, c[0x0][0x158];
MOV32I      R8, 0x4;
ISCADD      R7, R21, R23, 0x4;
SHF.L       R4, RZ, 0x4, R4;
IMAD        R6.CC, R7, R8, c[0x0][0x160];
IMAD        R5, R4, R21, R24;
IMAD.HI.X   R7, R7, R8, c[0x0][0x164];
IMAD        R4.CC, R5, R8, c[0x0][0x150];
LD.E        R11, [R6];
LD.E        R10, [R6+0x10];
LD.E        R9, [R6+0x20];
IMAD.HI.X   R5, R5, R8, c[0x0][0x154];
LD.E        R8, [R6+0x30];
LD.E        R18, [R4];
LD.E        R17, [R4+0x40];
LD.E        R16, [R4+0x80];
LD.E        R15, [R4+0xc0];
LD.E        R14, [R4+0x100];
LD.E        R13, [R4+0x140];
LD.E        R12, [R4+0x180];
LD.E        R6, [R4+0x1c0];
STS         [R3], R18;
STS         [R3+0x40], R17;
STS         [R3+0x80], R16;
STS         [R3+0xc0], R15;
STS         [R3+0x100], R14;
STS         [R3+0x140], R13;
STS         [R3+0x180], R12;
STS         [R3+0x1c0], R6;
STS         [R20], R11;
STS         [R20+0x10], R10;
STS         [R20+0x20], R9;
STS         [R20+0x30], R8;
BAR.SYNC    0x0;
```

```
shl.b32         %r36, %r13, 4;
mad.lo.s32      %r37, %r36, %r55, %r6;
mul.wide.s32    %rd12, %r37, 4;
add.s64         %rd13, %rd2, %rd12;
mul.wide.s32    %rd14, %r2, 512;
add.s64         %rd16, %rd9, %rd14;
mul.wide.s32    %rd17, %r1, 4;
add.s64         %rd18, %rd16, %rd17;
ld.global.f32   %f163, [%rd13];
st.shared.f32   [%rd18], %f163;
ld.global.f32   %f164, [%rd13+64];
st.shared.f32   [%rd18+64], %f164;
ld.global.f32   %f165, [%rd13+128];
st.shared.f32   [%rd18+128], %f165;
ld.global.f32   %f166, [%rd13+192];
st.shared.f32   [%rd18+192], %f166;
ld.global.f32   %f167, [%rd13+256];
st.shared.f32   [%rd18+256], %f167;
ld.global.f32   %f168, [%rd13+320];
st.shared.f32   [%rd18+320], %f168;
ld.global.f32   %f169, [%rd13+384];
st.shared.f32   [%rd18+384], %f169;
ld.global.f32   %f170, [%rd13+448];
st.shared.f32   [%rd18+448], %f170;
shl.b32         %r38, %r55, 4;
add.s32         %r39, %r7, %r38;
mul.wide.s32    %rd19, %r39, 4;
add.s64         %rd20, %rd1, %rd19;
mul.wide.s32    %rd21, %r4, 64;
add.s64         %rd23, %rd11, %rd21;
mul.wide.s32    %rd24, %r3, 4;
add.s64         %rd25, %rd23, %rd24;
ld.global.f32   %f171, [%rd20];
st.shared.f32   [%rd25], %f171;
ld.global.f32   %f172, [%rd20+16];
st.shared.f32   [%rd25+16], %f172;
ld.global.f32   %f173, [%rd20+32];
st.shared.f32   [%rd25+32], %f173;
ld.global.f32   %f174, [%rd20+48];
st.shared.f32   [%rd25+48], %f174;
bar.sync        0;
```

```
for (kk = 0; kk < K; kk += BLK_K)
{
    #pragma unroll
    for (n = 0; n < BLK_K; n += DIM_YA)
        #pragma unroll
        for (m = 0; m < BLK_M; m += DIM_XA)
            sA[n+idyA][m+idxA] = A[coord_A + (n*lda+m)];

    #pragma unroll
    for (n = 0; n < BLK_N; n += DIM_YB)
        #pragma unroll
        for (m = 0; m < BLK_K; m += DIM_XB)
            sB[n+idyB][m+idxB] = B[coord_B + (n*ldb+m)];

    __syncthreads();

    #pragma unroll
    for (k = 0; k < BLK_K; k++)
        #pragma unroll
        for (n = 0; n < THR_N; n++)
            #pragma unroll
            for (m = 0; m < THR_M; m++)
                rC[n][m] +=
                    sA[k][m*DIM_X+idx] *
                    sB[n*DIM_Y+idy][k];

    __syncthreads();

    coord_A += BLK_K*lda;
    coord_B += BLK_K;
}
```

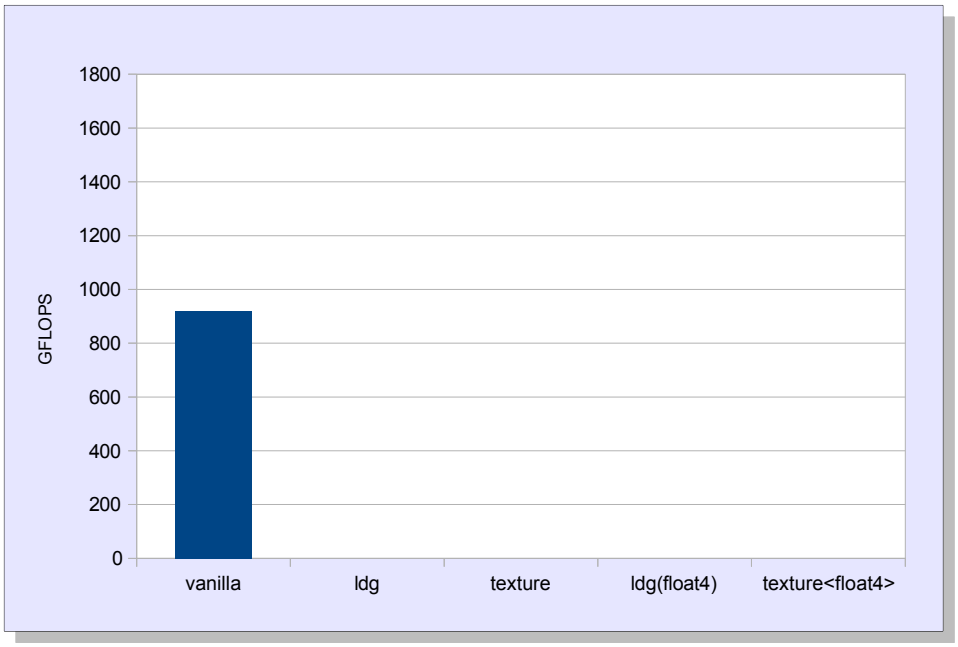
```
ld.shared.f32   %f175, [%rd3];
ld.shared.f32   %f176, [%rd4];
fma.rn.f32      %f177, %f175, %f176, %f976;
ld.shared.f32   %f178, [%rd3+64];
fma.rn.f32      %f179, %f178, %f176, %f975;
ld.shared.f32   %f180, [%rd3+128];
fma.rn.f32      %f181, %f180, %f176, %f974;
ld.shared.f32   %f182, [%rd3+192];
fma.rn.f32      %f183, %f182, %f176, %f973;
ld.shared.f32   %f184, [%rd3+256];
fma.rn.f32      %f185, %f184, %f176, %f972;
ld.shared.f32   %f186, [%rd3+320];
fma.rn.f32      %f187, %f186, %f176, %f971;
ld.shared.f32   %f188, [%rd3+384];
fma.rn.f32      %f189, %f188, %f176, %f970;
ld.shared.f32   %f190, [%rd3+448];
fma.rn.f32      %f191, %f190, %f176, %f969;
ld.shared.f32   %f192, [%rd4+1024];
fma.rn.f32      %f193, %f175, %f192, %f968;
fma.rn.f32      %f194, %f178, %f192, %f967;
fma.rn.f32      %f195, %f180, %f192, %f966;
fma.rn.f32      %f196, %f182, %f192, %f965;
fma.rn.f32      %f197, %f184, %f192, %f964;
fma.rn.f32      %f198, %f186, %f192, %f963;
fma.rn.f32      %f199, %f188, %f192, %f962;
fma.rn.f32      %f200, %f190, %f192, %f961;
ld.shared.f32   %f201, [%rd4+2048];
fma.rn.f32      %f202, %f175, %f201, %f960;
fma.rn.f32      %f203, %f178, %f201, %f959;
fma.rn.f32      %f204, %f180, %f201, %f958;
fma.rn.f32      %f205, %f182, %f201, %f957;
fma.rn.f32      %f206, %f184, %f201, %f956;
fma.rn.f32      %f207, %f186, %f201, %f955;
fma.rn.f32      %f208, %f188, %f201, %f954;
fma.rn.f32      %f209, %f190, %f201, %f953;
ld.shared.f32   %f210, [%rd4+3072];
fma.rn.f32      %f211, %f175, %f210, %f952;
fma.rn.f32      %f212, %f178, %f210, %f951;
fma.rn.f32      %f213, %f180, %f210, %f950;
fma.rn.f32      %f214, %f182, %f210, %f949;
fma.rn.f32      %f215, %f184, %f210, %f948;
fma.rn.f32      %f216, %f186, %f210, %f947;
fma.rn.f32      %f217, %f188, %f210, %f946;
```

```
LDS         R71, [R0];
LDS         R67, [R0+0x40];
LDS         R69, [R0+0x80];
LDS         R65, [R0+0xc0];
LDS         R59, [R0+0x100];
LDS         R57, [R0+0x140];
LDS         R63, [R0+0x180];
LDS         R61, [R0+0x1c0];
LDS.128     R12, [R2+0x400];
LDS.128     R8, [R2+0x800];
FFMA        R25, R71, R12, R25;
LDS.128     R16, [R2];
FFMA        R27, R69, R12, R27;
FFMA        R26, R67, R12, R26;
LDS.128     R4, [R2+0xc00];
FFMA        R29, R65, R12, R29;
LDS         R58, [R0+0x280];
FFMA        R31, R59, R12, R31;
LDS         R64, [R0+0x2c0];
FFMA        R32, R57, R12, R32;
LDS         R62, [R0+0x340];
FFMA        R33, R63, R12, R33;
LDS         R66, [R0+0x380];
FFMA        R34, R61, R12, R34;
LDS         R12, [R0+0x200];
FFMA        R35, R71, R8, R35;
LDS         R68, [R0+0x3c0];
FFMA        R36, R67, R8, R36;
LDS         R73, [R0+0x840];
FFMA        R30, R69, R8, R30;
LDS         R70, [R0+0xac0];
FFMA        R37, R65, R8, R37;
LDS         R76, [R0+0xbc0];
FFMA        R38, R59, R8, R38;
FFMA        R39, R57, R8, R39;
FFMA        R28, R63, R8, R28;
FFMA        R41, R61, R8, R41;
LDS         R8, [R0+0x240];
FFMA        R46, R71, R16, R46;
FFMA        R52, R67, R16, R52;
FFMA        R54, R69, R16, R54;
FFMA        R44, R65, R16, R44;
FFMA        R26, R8, R13, R26;
```

```
ld.shared.f32   %f175, [%rd3];
ld.shared.f32   %f176, [%rd4];
fma.rn.f32      %f177, %f175, %f176, %f976;
ld.shared.f32   %f178, [%rd3+64];
fma.rn.f32      %f179, %f178, %f176, %f975;
ld.shared.f32   %f180, [%rd3+128];
fma.rn.f32      %f181, %f180, %f176, %f974;
ld.shared.f32   %f182, [%rd3+192];
fma.rn.f32      %f183, %f182, %f176, %f973;
ld.shared.f32   %f184, [%rd3+256];
fma.rn.f32      %f185, %f184, %f176, %f972;
ld.shared.f32   %f186, [%rd3+320];
fma.rn.f32      %f187, %f186, %f176, %f971;
ld.shared.f32   %f188, [%rd3+384];
fma.rn.f32      %f189, %f188, %f176, %f970;
ld.shared.f32   %f190, [%rd3+448];
fma.rn.f32      %f191, %f190, %f176, %f969;
ld.shared.f32   %f192, [%rd4+1024];
fma.rn.f32      %f193, %f175, %f192, %f968;
fma.rn.f32      %f194, %f178, %f192, %f967;
fma.rn.f32      %f195, %f180, %f192, %f966;
fma.rn.f32      %f196, %f182, %f192, %f965;
fma.rn.f32      %f197, %f184, %f192, %f964;
fma.rn.f32      %f198, %f186, %f192, %f963;
fma.rn.f32      %f199, %f188, %f192, %f962;
fma.rn.f32      %f200, %f190, %f192, %f961;
ld.shared.f32   %f201, [%rd4+2048];
fma.rn.f32      %f202, %f175, %f201, %f960;
fma.rn.f32      %f203, %f178, %f201, %f959;
fma.rn.f32      %f204, %f180, %f201, %f958;
fma.rn.f32      %f205, %f182, %f201, %f957;
fma.rn.f32      %f206, %f184, %f201, %f956;
fma.rn.f32      %f207, %f186, %f201, %f955;
fma.rn.f32      %f208, %f188, %f201, %f954;
fma.rn.f32      %f209, %f190, %f201, %f953;
ld.shared.f32   %f210, [%rd4+3072];
fma.rn.f32      %f211, %f175, %f210, %f952;
fma.rn.f32      %f212, %f178, %f210, %f951;
fma.rn.f32      %f213, %f180, %f210, %f950;
fma.rn.f32      %f214, %f182, %f210, %f949;
fma.rn.f32      %f215, %f184, %f210, %f948;
fma.rn.f32      %f216, %f186, %f210, %f947;
fma.rn.f32      %f217, %f188, %f210, %f946;
```

Performance

vanilla



__ldg() intrinsic

const float* __restrict__

```
extern "C" __global__
void beast_gemm_kernel(
    int M, int N, int K,
    float alpha, float *A, int lda,
                 float *B, int ldb,
     float beta, float *C, int ldc)
```

```
extern "C" __global__
void beast_gemm_kernel(
    int M, int N, int K,
    float alpha, const float* __restrict__ A, int lda,
                 const float* __restrict__ B, int ldb,
     float beta,       float*               C, int ldc)
```

```
#pragma unroll
for (n = 0; n < BLK_K; n += DIM_YA)
    #pragma unroll
    for (m = 0; m < BLK_M; m += DIM_XA)
        sA[n+idyA][m+idxA] = A[coord_A + (n*lda+m)];

#pragma unroll
for (n = 0; n < BLK_N; n += DIM_YB)
    #pragma unroll
    for (m = 0; m < BLK_K; m += DIM_XB)
        sB[n+idyB][m+idxB] = B[coord_B + (n*ldb+m)];
```

```
#pragma unroll
for (n = 0; n < BLK_K; n += DIM_YA)
    #pragma unroll
    for (m = 0; m < BLK_M; m += DIM_XA)
        sA[n+idyA][m+idxA] = __ldg(A + coord_A + (n*lda+m));

#pragma unroll
for (n = 0; n < BLK_N; n += DIM_YB)
    #pragma unroll
    for (m = 0; m < BLK_K; m += DIM_XB)
        sB[n+idyB][m+idxB] = __ldg(B + coord_B + (n*ldb+m));
```

```
mul.wide.s32  %rd17, %r1, 4;
add.s64       %rd18, %rd16, %rd17;
ld.global.f32 %f163, [%rd13];
st.shared.f32 [%rd18], %f163;
ld.global.f32 %f164, [%rd13+64];
st.shared.f32 [%rd18+64], %f164;
ld.global.f32 %f165, [%rd13+128];
st.shared.f32 [%rd18+128], %f165;
ld.global.f32 %f166, [%rd13+192];
st.shared.f32 [%rd18+192], %f166;
ld.global.f32 %f167, [%rd13+256];
st.shared.f32 [%rd18+256], %f167;
ld.global.f32 %f168, [%rd13+320];
st.shared.f32 [%rd18+320], %f168;
ld.global.f32 %f169, [%rd13+384];
st.shared.f32 [%rd18+384], %f169;
ld.global.f32 %f170, [%rd13+448];
st.shared.f32 [%rd18+448], %f170;
shl.b32       %r38, %r55, 4;
add.s32       %r39, %r7, %r38;
```

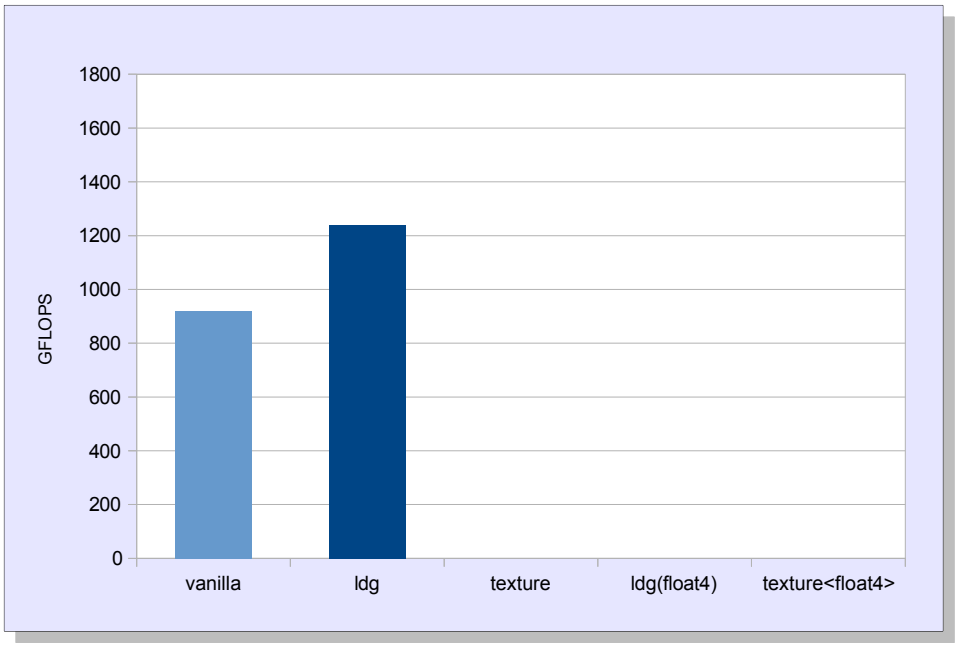
```
mul.wide.s32      %rd30, %r45, 4;
add.s64           %rd12, %rd4, %rd30;
ld.global.nc.f32  %f163, [%rd11];
mul.wide.s32      %rd31, %r3, 512;
add.s64           %rd33, %rd8, %rd31;
mul.wide.s32      %rd34, %r2, 4;
add.s64           %rd35, %rd33, %rd34;
st.shared.f32     [%rd35], %f163;
ld.global.nc.f32  %f164, [%rd12];
st.shared.f32     [%rd35+64], %f164;
ld.global.nc.f32  %f165, [%rd13];
st.shared.f32     [%rd35+128], %f165;
ld.global.nc.f32  %f166, [%rd14];
st.shared.f32     [%rd35+192], %f166;
ld.global.nc.f32  %f167, [%rd15];
st.shared.f32     [%rd35+256], %f167;
ld.global.nc.f32  %f168, [%rd16];
st.shared.f32     [%rd35+320], %f168;
ld.global.nc.f32  %f169, [%rd17];
st.shared.f32     [%rd35+384], %f169;
ld.global.nc.f32  %f170, [%rd18];
st.shared.f32     [%rd35+448], %f170;
mul.wide.s32      %rd36, %r37, 4;
add.s64           %rd19, %rd5, %rd36;
```

```
MOV         R4, c[0x0][0x158];
MOV32I      R8, 0x4;
ISCADD      R7, R21, R23, 0x4;
SHF.L       R4, RZ, 0x4, R4;
IMAD        R6.CC, R7, R8, c[0x0][0x160];
IMAD        R5, R4, R21, R24;
IMAD.HI.X   R7, R7, R8, c[0x0][0x164];
IMAD        R4.CC, R5, R8, c[0x0][0x150];
LD.E        R11, [R6];
LD.E        R10, [R6+0x10];
LD.E        R9, [R6+0x20];
IMAD.HI.X   R5, R5, R8, c[0x0][0x154];
LD.E        R8, [R6+0x30];
LD.E        R18, [R4];
LD.E        R17, [R4+0x40];
LD.E        R16, [R4+0x80];
LD.E        R15, [R4+0xc0];
LD.E        R14, [R4+0x100];
LD.E        R13, [R4+0x140];
LD.E        R12, [R4+0x180];
LD.E        R6, [R4+0x1c0];
STS         [R3], R18;
STS         [R3+0x40], R17;
STS         [R3+0x80], R16;
STS         [R3+0xc0], R15;
STS         [R3+0x100], R14;
STS         [R3+0x140], R13;
STS         [R3+0x180], R12;
STS         [R3+0x1c0], R6;
STS         [R20], R11;
STS         [R20+0x10], R10;
STS         [R20+0x20], R9;
STS         [R20+0x30], R8;
BAR.SYNC    0x0;
```

```
IMAD        R6.CC, R7, R26, c[0x0][0x150];
ISCADD      R19, R0, R21, 0x4;
LDG.E       R8, [R4];
IADD        R5, R14, 0x50;
IMAD.HI.X   R7, R7, R26, c[0x0][0x154];
IMAD R4.CC, R5, R26, c[0x0][0x150];
LDG.E       R9, [R6];
...
LDG.E       R10, [R4];
...
LDG.E       R11, [R6];
...
LDG.E       R13, [R6];
...
LDG.E       R14, [R6];
...
LDG.E       R4, [R4];
IMAD.HI.X   R7, R7, R26, c[0x0][0x164];
LDG.E       R5, [R6];
TEXDEPBAR   0xb;
STS         [R24], R8;
TEXDEPBAR   0xa;
STS         [R24+0x180], R9;
TEXDEPBAR   0x9;
STS         [R24+0x140], R10;
TEXDEPBAR   0x8;
STS         [R24+0x100], R11;
TEXDEPBAR   0x7;
STS         [R24+0xc0], R12;
TEXDEPBAR   0x6;
STS         [R24+0x80], R13;
TEXDEPBAR   0x5;
...
BAR.SYNC    0x0;
```

Performance

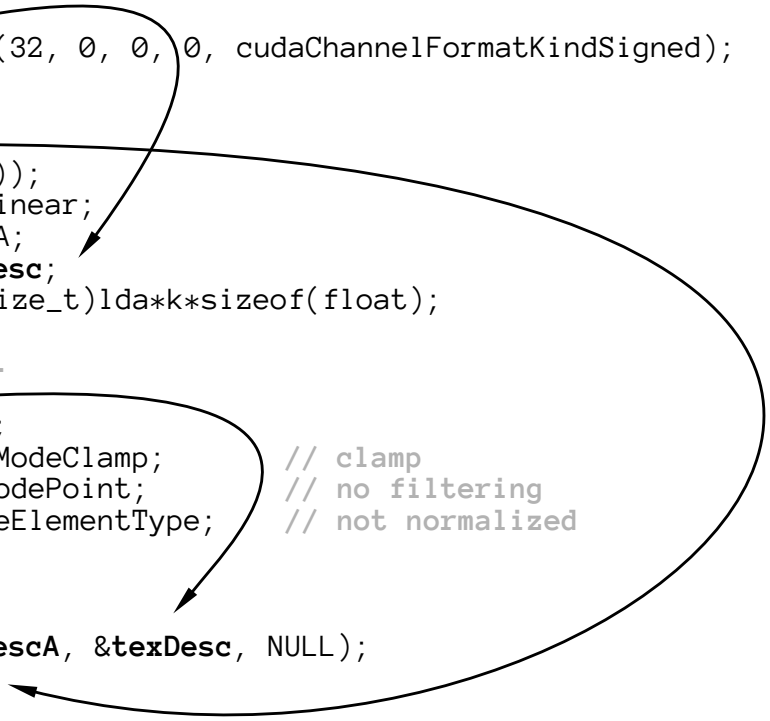
__ldg() intrinsic



Textures

setup

```
// Create channel.
cudaChannelFormatDesc channel_desc;
channel_desc = cudaCreateChannelDesc(32, 0, 0, 0, cudaChannelFormatKindSigned);

// Create resource descriptor.
struct cudaResourceDesc resDescA;
memset(&resDescA, 0, sizeof(resDescA));
resDescA.resType = cudaResourceTypeLinear;
resDescA.res.linear.devPtr = (void*)A;
resDescA.res.linear.desc = channel_desc;
resDescA.res.linear.sizeInBytes = (size_t)lda*k*sizeof(float);

// Specify texture object parameters.
struct cudaTextureDesc texDesc;
memset(&texDesc, 0, sizeof(texDesc));
texDesc.addressMode[0] = cudaAddressModeClamp;      // clamp
texDesc.filterMode     = cudaFilterModePoint;       // no filtering
texDesc.readMode       = cudaReadModeElementType;   // not normalized

// Create texture object.
cudaTextureObject_t texA;
cudaCreateTextureObject(&texA, &resDescA, &texDesc, NULL);
```



Textures

tex1Dfetch<float>

```
channel_desc = cudaCreateChannelDesc(32, 0, 0, 0, cudaChannelFormatKindSigned);
```

```
extern "C" __global__
void beast_gemm_kernel(
    int M, int N, int K,
    float alpha, float *A, int lda,
                 float *B, int ldb,
     float beta, float *C, int ldc)
```

```
extern "C" __global__
void beast_gemm_kernel(
    int M, int N, int K,
    float alpha, cudaTextureObject_t A, int lda,
                 cudaTextureObject_t B, int ldb,
     float beta,               float* C, int ldc)
```

```
#pragma unroll
for (n = 0; n < BLK_K; n += DIM_YA)
    #pragma unroll
    for (m = 0; m < BLK_M; m += DIM_XA)
        sA[n+idyA][m+idxA] = A[coord_A + (n*lda+m)];

#pragma unroll
for (n = 0; n < BLK_N; n += DIM_YB)
    #pragma unroll
    for (m = 0; m < BLK_K; m += DIM_XB)
        sB[n+idyB][m+idxB] = B[coord_B + (n*ldb+m)];
```

```
#pragma unroll
for (n = 0; n < BLK_K; n += DIM_YA)
    #pragma unroll
    for (m = 0; m < BLK_M; m += DIM_XA)
        sA[n+idyA][m+idxA] = tex1Dfetch<float>(A, coord_A + (n*lda+m));

#pragma unroll
for (n = 0; n < BLK_N; n += DIM_YB)
    #pragma unroll
    for (m = 0; m < BLK_K; m += DIM_XB)
        sB[n+idyB][m+idxB] = tex1Dfetch<float>(B, coord_B + (n*ldb+m));
```

```
mul.wide.s32    %rd17, %r1, 4;
add.s64         %rd18, %rd16, %rd17;
ld.global.f32   %f163, [%rd13];
st.shared.f32   [%rd18], %f163;
ld.global.f32   %f164, [%rd13+64];
st.shared.f32   [%rd18+64], %f164;
ld.global.f32   %f165, [%rd13+128];
st.shared.f32   [%rd18+128], %f165;
ld.global.f32   %f166, [%rd13+192];
st.shared.f32   [%rd18+192], %f166;
ld.global.f32   %f167, [%rd13+256];
st.shared.f32   [%rd18+256], %f167;
ld.global.f32   %f168, [%rd13+320];
st.shared.f32   [%rd18+320], %f168;
ld.global.f32   %f169, [%rd13+384];
st.shared.f32   [%rd18+384], %f169;
ld.global.f32   %f170, [%rd13+448];
st.shared.f32   [%rd18+448], %f170;
shl.b32         %r38, %r55, 4;
add.s32         %r39, %r7, %r38;
```

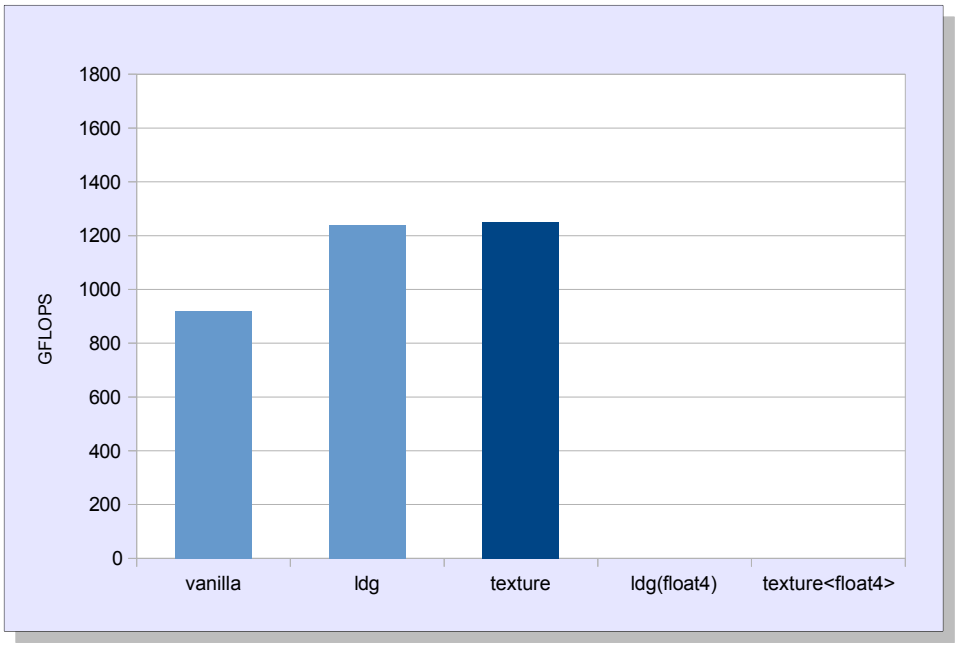
```
add.s32             %r38, %r36, 32;
add.s32             %r37, %r36, 16;
tex.1d.v4.f32.s32   {%f163, %f164, %f165, %f166}, [%rd4, {%r36}];
mul.wide.s32        %rd23, %r3, 512;
add.s64             %rd25, %rd8, %rd23;
mul.wide.s32        %rd26, %r2, 4;
add.s64             %rd27, %rd25, %rd26;
st.shared.f32       [%rd27], %f163;
tex.1d.v4.f32.s32   {%f167, %f168, %f169, %f170}, [%rd4, {%r37}];
st.shared.f32       [%rd27+64], %f167;
tex.1d.v4.f32.s32   {%f171, %f172, %f173, %f174}, [%rd4, {%r38}];
st.shared.f32       [%rd27+128], %f171;
tex.1d.v4.f32.s32   {%f175, %f176, %f177, %f178}, [%rd4, {%r39}];
st.shared.f32       [%rd27+192], %f175;
tex.1d.v4.f32.s32   {%f179, %f180, %f181, %f182}, [%rd4, {%r40}];
st.shared.f32       [%rd27+256], %f179;
tex.1d.v4.f32.s32   {%f183, %f184, %f185, %f186}, [%rd4, {%r41}];
st.shared.f32       [%rd27+320], %f183;
tex.1d.v4.f32.s32   {%f187, %f188, %f189, %f190}, [%rd4, {%r42}];
st.shared.f32       [%rd27+384], %f187;
tex.1d.v4.f32.s32   {%f191, %f192, %f193, %f194}, [%rd4, {%r43}];
st.shared.f32       [%rd27+448], %f191;
```

```
MOV         R4, c[0x0][0x158];
MOV32I      R8, 0x4;
ISCADD      R7, R21, R23, 0x4;
SHF.L       R4, RZ, 0x4, R4;
IMAD        R6.CC, R7, R8, c[0x0][0x160];
IMAD        R5, R4, R21, R24;
IMAD.HI.X   R7, R7, R8, c[0x0][0x164];
IMAD        R4.CC, R5, R8, c[0x0][0x150];
LD.E        R11, [R6];
LD.E        R10, [R6+0x10];
LD.E        R9, [R6+0x20];
IMAD.HI.X   R5, R5, R8, c[0x0][0x154];
LD.E        R8, [R6+0x30];
LD.E        R18, [R4];
LD.E        R17, [R4+0x40];
LD.E        R16, [R4+0x80];
LD.E        R15, [R4+0xc0];
LD.E        R14, [R4+0x100];
LD.E        R13, [R4+0x140];
LD.E        R12, [R4+0x180];
LD.E        R6, [R4+0x1c0];
STS         [R3], R18;
STS         [R3+0x40], R17;
STS         [R3+0x80], R16;
STS         [R3+0xc0], R15;
STS         [R3+0x100], R14;
STS         [R3+0x140], R13;
STS         [R3+0x180], R12;
STS         [R3+0x1c0], R6;
STS         [R20], R11;
STS         [R20+0x10], R10;
STS         [R20+0x20], R9;
STS         [R20+0x30], R8;
BAR.SYNC    0x0;
```

```
IADD        R9, R11, 0x20;
IADD        R10, R11, 0x30;
TLD.LZ.T    R7, R11, 0x54, 1D, 0x1;
TLD.LZ.T    R8, R8, 0x54, 1D, 0x1;
TLD.LZ.T    R9, R9, 0x54, 1D, 0x1;
TLD.LZ.P    R10, R10, 0x54, 1D, 0x1;
IADD        R12, R11, 0x40;
IADD        R13, R11, 0x50;
IADD        R14, R11, 0x60;
IADD        R15, R11, 0x70;
 ...
TLD.LZ.T    R15, R15, 0x58, 1D, 0x1;
TLD.LZ.T    R16, R16, 0x58, 1D, 0x1;
TLD.LZ.T    R17, R17, 0x58, 1D, 0x1;
TLD.LZ.P    R18, R18, 0x58, 1D, 0x1;
TEXDEPBAR   0xb;
STS         [R24], R7;
TEXDEPBAR   0xa;
STS         [R24+0x40], R8;
TEXDEPBAR   0x9;
STS         [R24+0x80], R9;
TEXDEPBAR   0x8;
STS         [R24+0xc0], R10;
TEXDEPBAR   0x7;
STS         [R24+0x100], R11;
TEXDEPBAR   0x6;
STS         [R24+0x140], R12;
TEXDEPBAR   0x5;
STS         [R24+0x180], R13;
TEXDEPBAR   0x4;
STS         [R24+0x1c0], R14;
 ...
BAR.SYNC    0x0;
```

Performance

tex1Dfetch<float>

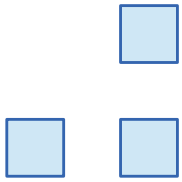


__ldg(float4)

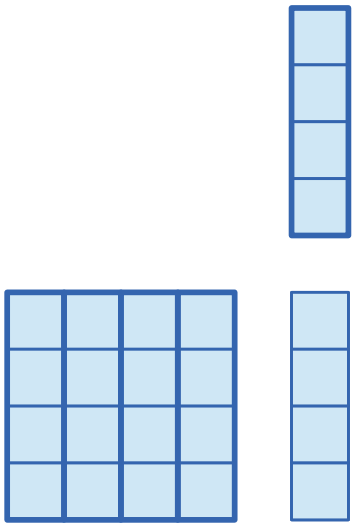
```
extern "C" __global__
void beast_gemm_kernel(
    int M, int N, int K,
    float alpha, const float* __restrict__ A, int lda,
                 const float* __restrict__ B, int ldb,
     float beta,       float*              C, int ldc)
{
     ...

    __shared__ float sA[BLK_K][BLK_M];      // shared memory buffer for A
    __shared__ float sB[BLK_N][BLK_K];      // shared memory buffer for B
    float rC[THR_N][THR_M];                 // registers for C

    int coord_A = (blx*BLK_M     + idyA*lda) + idxA;
    int coord_B = (bly*BLK_N*ldb + idyB*ldb) + idxB;

     ...

    for (kk = 0; kk < K; kk += BLK_K)
    {
        #pragma unroll
        for (n = 0; n < BLK_K; n += DIM_YA)
            #pragma unroll
            for (m = 0; m < BLK_M; m += DIM_XA)
                sA[n+idyA][m+idxA] = __ldg(A + coord_A + (n*lda+m));

        #pragma unroll
        for (n = 0; n < BLK_N; n += DIM_YB)
            #pragma unroll
            for (m = 0; m < BLK_K; m += DIM_XB)
                sB[n+idyB][m+idxB] = __ldg(B + coord_B + (n*ldb+m));

        __syncthreads();
```

__ldg(float4)

```
extern "C" __global__
void beast_gemm_kernel(
    int M, int N, int K,
    float alpha, const float4* __restrict__ A, int lda,
                 const float4* __restrict__ B, int ldb,
     float beta,        float*              C, int ldc)
{
     ...

    __shared__ float4 sA[BLK_K][BLK_M/4];   // shared memory buffer for A
    __shared__ float4 sB[BLK_N][BLK_K/4];   // shared memory buffer for B
    float rC[THR_N][THR_M];                 // registers for C

    int coord_A = (blx*BLK_M     + idyA*lda)/4 + idxA;
    int coord_B = (bly*BLK_N*ldb + idyB*ldb)/4 + idxB;

     ...

    for (kk = 0; kk < K; kk += BLK_K)
    {
        #pragma unroll
        for (n = 0; n < BLK_K; n += DIM_YA)
            #pragma unroll
            for (m = 0; m < BLK_M; m += 4*DIM_XA)
                sA[n+idyA][m/4+idxA] = __ldg(A + coord_A + (n*lda+m)/4);

        #pragma unroll
        for (n = 0; n < BLK_N; n += DIM_YB)
            #pragma unroll
            for (m = 0; m < BLK_K; m += 4*DIM_XB)
                sB[n+idyB][m/4+idxB] = __ldg(B + coord_B + (n*ldb+m)/4);

        __syncthreads();
```

```
        #pragma unroll
        for (k = 0; k < BLK_K; k++)
            #pragma unroll
            for (n = 0; n < THR_N; n++)
                #pragma unroll
                for (m = 0; m < THR_M; m++)
                    rC[n][m] += sA[k][m*DIM_X+idx] * sB[n*DIM_Y+idy][k];



        __syncthreads();

        coord_A += BLK_K*lda;
        coord_B += BLK_K;
    }
```

```
    #pragma unroll
    for (k = 0; k < BLK_K; k+=4)
        #pragma unroll
        for (n = 0; n < THR_N; n++)
            #pragma unroll
            for (m = 0; m < THR_M; m+=4)
            {
                float4 rB  = sB[n*DIM_Y+idy][k/4];

                float4 rA0 = sA[k  ][m*DIM_X/4+idx];
                float4 rA1 = sA[k+1][m*DIM_X/4+idx];
                float4 rA2 = sA[k+2][m*DIM_X/4+idx];
                float4 rA3 = sA[k+3][m*DIM_X/4+idx];

                rC[n][m  ] += rA0.x * rB.x;
                rC[n][m  ] += rA1.x * rB.y;
                rC[n][m  ] += rA2.x * rB.z;
                rC[n][m  ] += rA3.x * rB.w;

                rC[n][m+1] += rA0.y * rB.x;
                rC[n][m+1] += rA1.y * rB.y;
                rC[n][m+1] += rA2.y * rB.z;
                rC[n][m+1] += rA3.y * rB.w;

                rC[n][m+2] += rA0.z * rB.x;
                rC[n][m+2] += rA1.z * rB.y;
                rC[n][m+2] += rA2.z * rB.z;
                rC[n][m+2] += rA3.z * rB.w;

                rC[n][m+3] += rA0.w * rB.x;
                rC[n][m+3] += rA1.w * rB.y;
                rC[n][m+3] += rA2.w * rB.z;
                rC[n][m+3] += rA3.w * rB.w;
            }

    __syncthreads();

    coord_A += BLK_K*lda/4;
    coord_B += BLK_K/4;
}
```



```
#pragma unroll
for (n = 0; n < THR_N; n++) {
    int coord_dCn = bly*BLK_N + n*DIM_Y+idy;
    #pragma unroll
    for (m = 0; m < THR_M; m++) {
        int offsC = coord_dCn*ldc + blx*BLK_M;
        C[offsC + m*DIM_X+idx] = alpha*rC[n][m] + beta*C[offsC + m*DIM_X+idx];
    }
}
```

```
#pragma unroll
for (n = 0; n < THR_N; n++) {
    int coord_dCn = bly*BLK_N + n*DIM_Y+idy;
    #pragma unroll
    for (m = 0; m < THR_M; m+=4) {
        int offsC = coord_dCn*ldc + blx*BLK_M;
        C[offsC + m*DIM_X+4*idx  ] = alpha*rC[n][m  ] + beta*C[offsC + m*DIM_X+4*idx  ];
        C[offsC + m*DIM_X+4*idx+1] = alpha*rC[n][m+1] + beta*C[offsC + m*DIM_X+4*idx+1];
        C[offsC + m*DIM_X+4*idx+2] = alpha*rC[n][m+2] + beta*C[offsC + m*DIM_X+4*idx+2];
        C[offsC + m*DIM_X+4*idx+3] = alpha*rC[n][m+3] + beta*C[offsC + m*DIM_X+4*idx+3];
    }
}
```

```
ld.global.nc.f32 %f163, [%rd11];
mul.wide.s32     %rd31, %r3, 512;
add.s64          %rd33, %rd8, %rd31;
mul.wide.s32     %rd34, %r2, 4;
add.s64          %rd35, %rd33, %rd34;
st.shared.f32    [%rd35], %f163;
ld.global.nc.f32 %f164, [%rd12];
st.shared.f32    [%rd35+64], %f164;
ld.global.nc.f32 %f165, [%rd13];
st.shared.f32    [%rd35+128], %f165;
ld.global.nc.f32 %f166, [%rd14];
st.shared.f32    [%rd35+192], %f166;
ld.global.nc.f32 %f167, [%rd15];
st.shared.f32    [%rd35+256], %f167;
ld.global.nc.f32 %f168, [%rd16];
st.shared.f32    [%rd35+320], %f168;
ld.global.nc.f32 %f169, [%rd17];
st.shared.f32    [%rd35+384], %f169;
ld.global.nc.f32 %f170, [%rd18];
st.shared.f32    [%rd35+448], %f170;
mul.wide.s32     %rd36, %r37, 4;
add.s64          %rd19, %rd5, %rd36;
add.s32          %r46, %r37, 12;
mul.wide.s32     %rd37, %r46, 4;
add.s64          %rd22, %rd5, %rd37;
add.s32          %r47, %r37, 8;
mul.wide.s32     %rd38, %r47, 4;
add.s64          %rd21, %rd5, %rd38;
add.s32          %r48, %r37, 4;
mul.wide.s32     %rd39, %r48, 4;
add.s64          %rd20, %rd5, %rd39;
ld.global.nc.f32  %f171, [%rd19];
mul.wide.s32     %rd40, %r5, 64;
add.s64          %rd42, %rd10, %rd40;
mul.wide.s32     %rd43, %r4, 4;
add.s64          %rd44, %rd42, %rd43;
st.shared.f32    [%rd44], %f171;
ld.global.nc.f32 %f172, [%rd20];
st.shared.f32    [%rd44+16], %f172;
ld.global.nc.f32 %f173, [%rd21];
st.shared.f32    [%rd44+32], %f173;
ld.global.nc.f32 %f174, [%rd22];
st.shared.f32    [%rd44+48], %f174;
bar.sync         0;
```

```
mul.wide.s32          %rd14, %r66, 16;
add.s64               %rd11, %rd4, %rd14;
ld.global.nc.v4.f32   {%f163,%f164,%f165,%f166}, [%rd11];
mul.wide.s32          %rd15, %r3, 512;
add.s64               %rd17, %rd10, %rd15;
mul.wide.s32          %rd18, %r2, 16;
add.s64               %rd19, %rd17, %rd18;
add.s32               %r50, %r66, 16;
mul.wide.s32          %rd20, %r50, 16;
add.s64               %rd12, %rd4, %rd20;
st.shared.v4.f32      [%rd19], {%f163, %f164, %f165, %f166};
ld.global.nc.v4.f32   {%f167,%f168,%f169,%f170}, [%rd12];
mul.wide.s32          %rd21, %r65, 16;
add.s64               %rd13, %rd5, %rd21;
st.shared.v4.f32      [%rd19+256], {%f167, %f168, %f169, %f170};
ld.global.nc.v4.f32   {%f171,%f172,%f173,%f174}, [%rd13];
mul.wide.s32          %rd22, %r5, 64;
add.s64               %rd24, %rd8, %rd22;
mul.wide.s32          %rd25, %r4, 16;
add.s64               %rd26, %rd24, %rd25;
st.shared.v4.f32      [%rd26], {%f171, %f172, %f173, %f174};
bar.sync              0;
```

```
  ...
IMAD      R6.CC, R7, R26, c[0x0][0x150];
ISCADD    R19, R0, R21, 0x4;
LDG.E     R8, [R4];
  ...
LDG.E     R9, [R6];
  ...
LDG.E     R10, [R4];
  ...
LDG.E     R11, [R6];
  ...
LDG.E     R12, [R4];
  ...
LDG.E     R13, [R6];
  ...
LDG.E     R17, [R4];
  ...
LDG.E     R16, [R6];
  ...
LDG.E     R15, [R4];
  ...
LDG.E     R14, [R6];
  ...
LDG.E     R4, [R4];
IMAD.HI.X R7, R7, R26, c[0x0][0x164];
LDG.E R5, [R6];
TEXDEPBAR 0xb;
STS       [R24], R8;
TEXDEPBAR 0xa;
STS       [R24+0x180], R9;
TEXDEPBAR 0x9;
STS       [R24+0x140], R10;
TEXDEPBAR 0x8;
STS       [R24+0x100], R11;
TEXDEPBAR 0x7;
STS       [R24+0xc0], R12;
TEXDEPBAR 0x6;
STS       [R24+0x80], R13;
TEXDEPBAR 0x5;
STS       [R24+0x40], R17;
TEXDEPBAR 0x4;
STS       [R24+0x1c0], R16;
TEXDEPBAR 0x3;
STS       [R25], R15;
TEXDEPBAR 0x2;
STS       [R25+0x30], R14;
TEXDEPBAR 0x1;
STS       [R25+0x20], R4;
TEXDEPBAR 0x0;
STS       [R25+0x10], R5;
BAR.SYNC  0x0;
```

```
MOV32I      R20, 0x10;
IADD        R5, R3, 0x10;
IMAD        R6.CC, R3, R20, c[0x0][0x150];
IMAD.HI.X   R7, R3, R20, c[0x0][0x154];
IMAD        R4.CC, R5, R20, c[0x0][0x150];
LDG.E.128   R12, [R6];
IMAD.HI.X   R5, R5, R20, c[0x0][0x154];
IMAD        R16.CC, R40, R20, c[0x0][0x160];
IMAD.HI.X   R17, R40, R20, c[0x0][0x164];
LDG.E.128   R8, [R4];
LDG.E.128   R4, [R16];
TEXDEPBAR   0x2;
STS.128     [R42], R12;
TEXDEPBAR   0x1;
STS.128     [R42+0x100], R8;
TEXDEPBAR   0x0;
STS.128     [R43], R4;
BAR.SYNC    0x0;
```

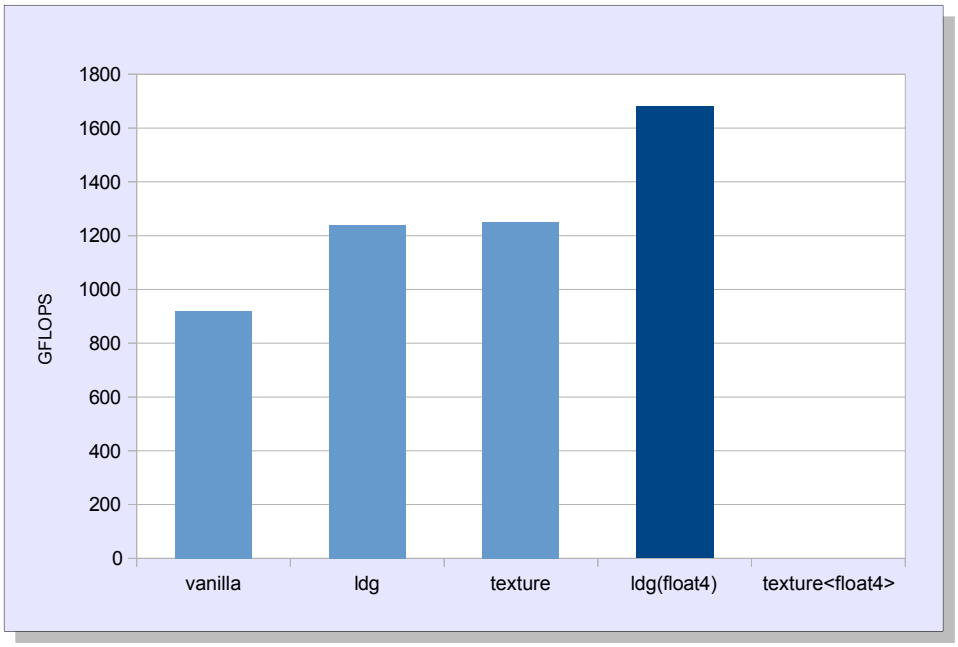
```
ld.shared.f32     %f175, [%rd2];
ld.shared.f32     %f176, [%rd3];
fma.rn.f32        %f177, %f175, %f176, %f976;
ld.shared.f32     %f178, [%rd2+64];
fma.rn.f32        %f179, %f178, %f176, %f975;
ld.shared.f32     %f180, [%rd2+128];
fma.rn.f32        %f181, %f180, %f176, %f974;
ld.shared.f32     %f182, [%rd2+192];
fma.rn.f32        %f183, %f182, %f176, %f973;
ld.shared.f32     %f184, [%rd2+256];
fma.rn.f32        %f185, %f184, %f176, %f972;
ld.shared.f32     %f186, [%rd2+320];
fma.rn.f32        %f187, %f186, %f176, %f971;
ld.shared.f32     %f188, [%rd2+384];
fma.rn.f32        %f189, %f188, %f176, %f970;
ld.shared.f32     %f190, [%rd2+448];
fma.rn.f32        %f191, %f190, %f176, %f969;
ld.shared.f32     %f192, [%rd3+1024];
fma.rn.f32        %f193, %f175, %f192, %f968;
fma.rn.f32        %f194, %f178, %f192, %f967;
fma.rn.f32        %f195, %f180, %f192, %f966;
fma.rn.f32        %f196, %f182, %f192, %f965;
fma.rn.f32        %f197, %f184, %f192, %f964;
fma.rn.f32        %f198, %f186, %f192, %f963;
fma.rn.f32        %f199, %f188, %f192, %f962;
fma.rn.f32        %f200, %f190, %f192, %f961;
ld.shared.f32     %f201, [%rd3+2048];
fma.rn.f32        %f202, %f175, %f201, %f960;
fma.rn.f32        %f203, %f178, %f201, %f959;
fma.rn.f32        %f204, %f180, %f201, %f958;
fma.rn.f32        %f205, %f182, %f201, %f957;
fma.rn.f32        %f206, %f184, %f201, %f956;
fma.rn.f32        %f207, %f186, %f201, %f955;
fma.rn.f32        %f208, %f188, %f201, %f954;
fma.rn.f32        %f209, %f190, %f201, %f953;
ld.shared.f32     %f210, [%rd3+3072];
fma.rn.f32        %f211, %f175, %f210, %f952;
fma.rn.f32        %f212, %f178, %f210, %f951;
fma.rn.f32        %f213, %f180, %f210, %f950;
fma.rn.f32        %f214, %f182, %f210, %f949;
fma.rn.f32        %f215, %f184, %f210, %f948;
fma.rn.f32        %f216, %f186, %f210, %f947;
fma.rn.f32        %f217, %f188, %f210, %f946;
fma.rn.f32        %f218, %f190, %f210, %f945;
fma.rn.f32        %f258, %f238, %f176, %f256;
fma.rn.f32        %f260, %f244, %f177, %f258;
fma.rn.f32        %f262, %f250, %f178, %f260;
fma.rn.f32        %f264, %f233, %f175, %f1162;
fma.rn.f32        %f266, %f239, %f176, %f264;
fma.rn.f32        %f268, %f245, %f177, %f266;
fma.rn.f32        %f270, %f251, %f178, %f268;
fma.rn.f32        %f272, %f234, %f175, %f1161;
fma.rn.f32        %f274, %f240, %f176, %f272;
fma.rn.f32        %f276, %f246, %f177, %f274;
fma.rn.f32        %f278, %f252, %f178, %f276;
ld.shared.v4.f32  {%f279, %f280, %f281, %f282}, [%rd2+1024];
fma.rn.f32        %f284, %f180, %f279, %f1160;
fma.rn.f32        %f286, %f187, %f280, %f284;
fma.rn.f32        %f288, %f194, %f281, %f286;
fma.rn.f32        %f290, %f201, %f282, %f288;
fma.rn.f32        %f291, %f181, %f279, %f1159;
fma.rn.f32        %f292, %f188, %f280, %f291;
fma.rn.f32        %f293, %f195, %f281, %f292;
fma.rn.f32        %f294, %f202, %f282, %f293;
fma.rn.f32        %f295, %f182, %f279, %f1158;
fma.rn.f32        %f296, %f189, %f280, %f295;
fma.rn.f32        %f297, %f196, %f281, %f296;
fma.rn.f32        %f298, %f203, %f282, %f297;
fma.rn.f32        %f299, %f183, %f279, %f1157;
fma.rn.f32        %f300, %f190, %f280, %f299;
fma.rn.f32        %f301, %f197, %f281, %f300;
fma.rn.f32        %f302, %f204, %f282, %f301;
fma.rn.f32        %f303, %f231, %f279, %f1156;
fma.rn.f32        %f304, %f237, %f280, %f303;
fma.rn.f32        %f305, %f243, %f281, %f304;
fma.rn.f32        %f306, %f249, %f282, %f305;
fma.rn.f32        %f307, %f232, %f279, %f1155;
fma.rn.f32        %f308, %f238, %f280, %f307;
fma.rn.f32        %f309, %f244, %f281, %f308;
fma.rn.f32        %f310, %f250, %f282, %f309;
fma.rn.f32        %f311, %f233, %f279, %f1154;
fma.rn.f32        %f312, %f239, %f280, %f311;
fma.rn.f32        %f313, %f245, %f281, %f312;
fma.rn.f32        %f314, %f251, %f282, %f313;
fma.rn.f32        %f315, %f234, %f279, %f1153;
fma.rn.f32        %f316, %f240, %f280, %f315;
fma.rn.f32        %f317, %f246, %f281, %f316;
fma.rn.f32        %f318, %f252, %f282, %f317;
```

```
LDS     R69, [R22+0x40];
LDS.128 R16, [R23];
LDS     R61, [R22+0x180];
FFMA    R58, R69, R16, R51;
LDS     R67, [R22+0x80];
FFMA    R60, R61, R16, R57;
LDS.128 R12, [R23+0x400];
FFMA    R52, R67, R16, R52;
LDS     R63, [R22];
FFMA    R51, R69, R12, R29;
LDS     R75, [R22+0xc0];
LDS     R73, [R22+0x100];
FFMA    R53, R75, R16, R53;
LDS     R71, [R22+0x140];
FFMA    R33, R73, R12, R33;
LDS.128 R8, [R23+0x800];
LDS     R65, [R22+0x1c0];
FFMA    R55, R71, R16, R55;
LDS.128 R4, [R23+0xc00];
FFMA    R32, R75, R12, R32;
FFMA    R57, R67, R12, R31;
LDS     R64, [R22+0x200];
FFMA    R27, R63, R12, R27;
LDS     R62, [R22+0x2c0];
FFMA    R34, R71, R12, R34;
LDS     R66, [R22+0x300];
FFMA    R35, R61, R12, R35;
LDS     R68, [R22+0x340];
FFMA    R29, R69, R8, R30;
LDS     R70, [R22+0x3c0];
FFMA    R31, R75, R8, R38;
FFMA    R12, R65, R12, R37;
FFMA    R30, R67, R8, R36;
FFMA    R38, R61, R8, R43;
FFMA    R28, R63, R8, R28;
FFMA    R36, R73, R8, R39;
LDS     R39, [R22+0x400];
FFMA    R37, R71, R8, R41;
FFMA    R43, R67, R4, R45;
LDS     R45, [R22+0x480];
FFMA    R8, R65, R8, R44;
LDS     R44, [R22+0x240];
FFMA    R41, R63, R4, R40;
LDS     R40, [R22+0x280];
```

```
LDS.128 R32, [R2];
LDS.128 R4, [R0];
LDS.128 R24, [R2+0x400];
LDS.128 R20, [R2+0x800];
LDS.128 R16, [R2+0xc00];
FFMA    R9, R4, R20, R39;
FFMA    R8, R4, R32, R31;
FFMA    R49, R6, R20, R49;
FFMA    R10, R4, R24, R37;
LDS.128 R12, [R0+0x100];
FFMA    R30, R5, R32, R30;
FFMA    R36, R5, R24, R36;
FFMA    R48, R5, R20, R48;
FFMA    R47, R6, R32, R47;
FFMA    R50, R12, R20, R50;
FFMA    R31, R4, R16, R55;
FFMA    R53, R14, R20, R53;
FFMA    R52, R5, R16, R52;
FFMA    R45, R6, R24, R45;
FFMA    R59, R6, R16, R59;
FFMA    R38, R7, R32, R38;
FFMA    R46, R7, R24, R46;
FFMA    R56, R7, R20, R56;
FFMA    R58, R7, R16, R60;
LDS.128 R4, [R0+0x200];
FFMA    R37, R12, R32, R61;
FFMA    R39, R13, R32, R54;
FFMA    R55, R14, R32, R63;
FFMA    R61, R15, R32, R62;
FFMA    R48, R5, R21, R48;
FFMA    R51, R13, R20, R51;
FFMA    R59, R6, R17, R59;
FFMA    R57, R15, R20, R57;
FFMA    R32, R13, R24, R68;
FFMA    R56, R7, R21, R56;
FFMA    R20, R13, R16, R70;
FFMA    R13, R4, R33, R8;
FFMA    R63, R4, R25, R10;
FFMA    R64, R4, R21, R9;
LDS.128 R8, [R0+0x300];
FFMA    R54, R12, R24, R67;
FFMA    R12, R12, R16, R65;
FFMA    R31, R4, R17, R31;
FFMA    R30, R5, R33, R30;
```

Performance

__ldg(float4)



Textures

tex1Dfetch<float4>

```
channel_desc = cudaCreateChannelDesc(32, 32, 32, 32, cudaChannelFormatKindSigned);
```

```
extern "C" __global__
void beast_gemm_kernel(
    int M, int N, int K,
    float alpha, const float4* __restrict__ A, int lda,
                 const float4* __restrict__ B, int ldb,
     float beta,       float*               C, int ldc)
```

```
extern "C" __global__
void beast_gemm_kernel(
    int M, int N, int K,
    float alpha, cudaTextureObject_t A, int lda,
                 cudaTextureObject_t B, int ldb,
     float beta,              float* C, int ldc)
```

```
#pragma unroll
for (n = 0; n < BLK_K; n += DIM_YA)
    #pragma unroll
    for (m = 0; m < BLK_M; m += 4*DIM_XA)
        sA[n+idyA][m/4+idxA] = __ldg(A + coord_A + (n*lda+m)/4);

#pragma unroll
for (n = 0; n < BLK_N; n += DIM_YB)
    #pragma unroll
    for (m = 0; m < BLK_K; m += 4*DIM_XB)
        sB[n+idyB][m/4+idxB] = __ldg(B + coord_B + (n*ldb+m)/4);
```

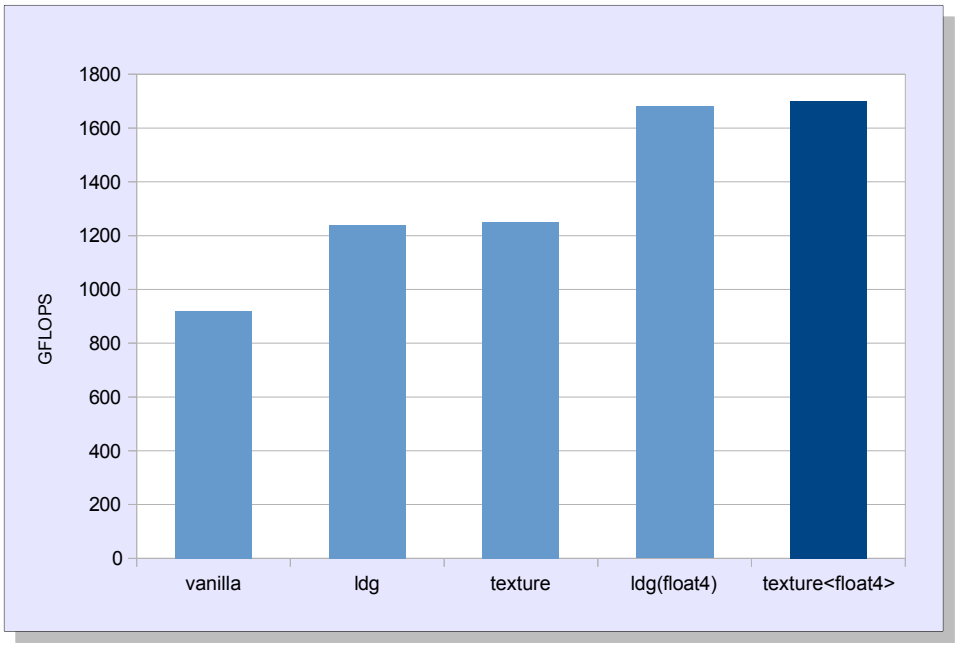
```
#pragma unroll
for (n = 0; n < BLK_K; n += DIM_YA)
    #pragma unroll
    for (m = 0; m < BLK_M; m += 4*DIM_XA)
        sA[n+idyA][m/4+idxA] = tex1Dfetch<float4>(A, coord_A + (n*lda+m)/4);

#pragma unroll
for (n = 0; n < BLK_N; n += DIM_YB)
    #pragma unroll
    for (m = 0; m < BLK_K; m += 4*DIM_XB)
        sB[n+idyB][m/4+idxB] = tex1Dfetch<float4>(B, coord_B + (n*ldb+m)/4);
```

```
...
ld.global.nc.v4.f32     {%f163,%f164,%f165,%f166}, [%rd11];
...
st.shared.v4.f32        [%rd19], {%f163, %f164, %f165, %f166};
ld.global.nc.v4.f32     {%f167,%f168,%f169,%f170}, [%rd12];
...
st.shared.v4.f32        [%rd19+256], {%f167, %f168, %f169, %f170};
ld.global.nc.v4.f32     {%f171,%f172,%f173,%f174}, [%rd13];
...
st.shared.v4.f32        [%rd26], {%f171, %f172, %f173, %f174};
bar.sync                0;
```

```
tex.1d.v4.f32.s32       {%f163, %f164, %f165, %f166}, [%rd4, {%r68}];
...
st.shared.v4.f32        [%rd18], {%f163, %f164, %f165, %f166};
tex.1d.v4.f32.s32       {%f167, %f168, %f169, %f170}, [%rd4, {%r51}];
st.shared.v4.f32        [%rd18+256], {%f167, %f168, %f169, %f170};
tex.1d.v4.f32.s32       {%f171, %f172, %f173, %f174}, [%rd5, {%r67}];
...
st.shared.v4.f32        [%rd23], {%f171, %f172, %f173, %f174};
bar.sync                0;
```

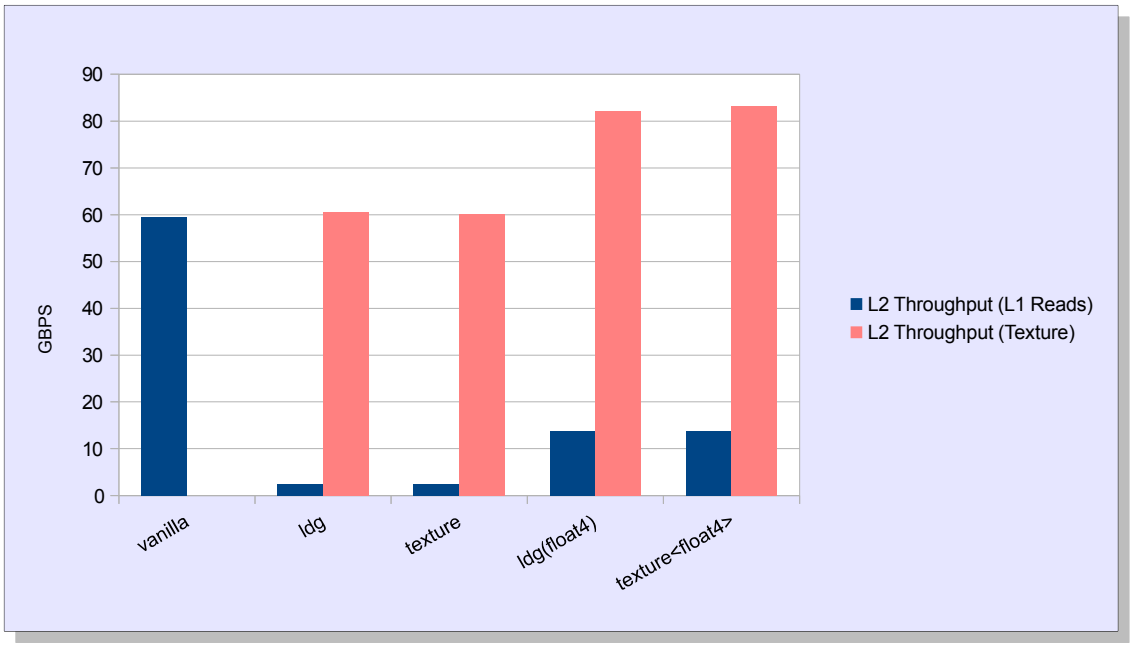
Performance

tex1Dfetch<float4>



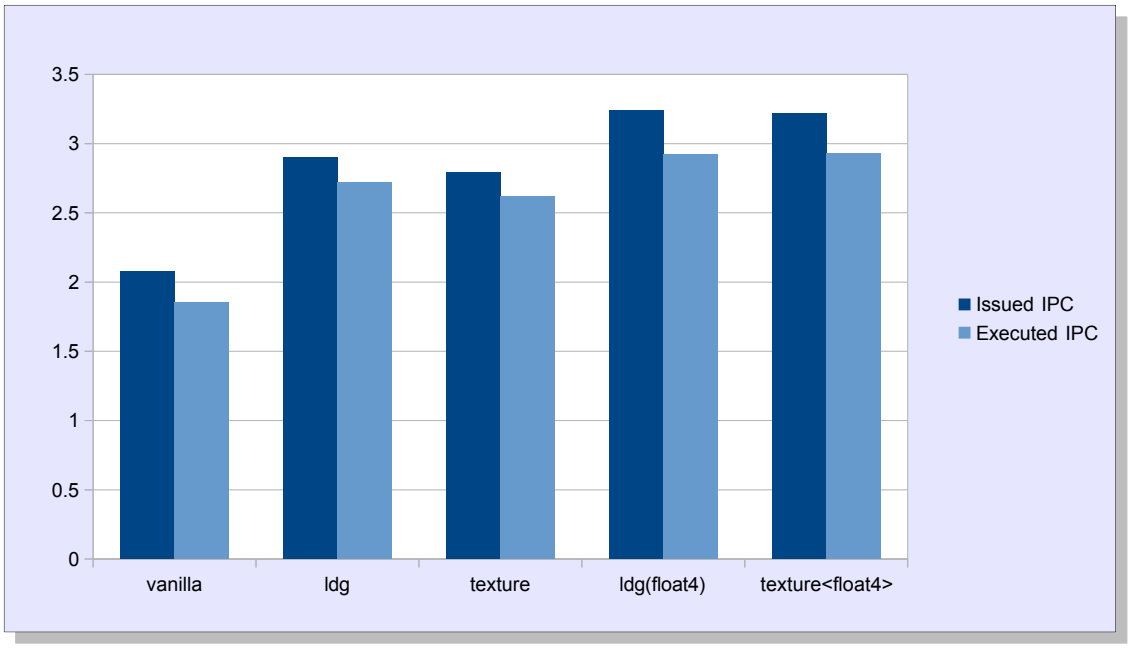
L2 Throughput

L1 vs texture



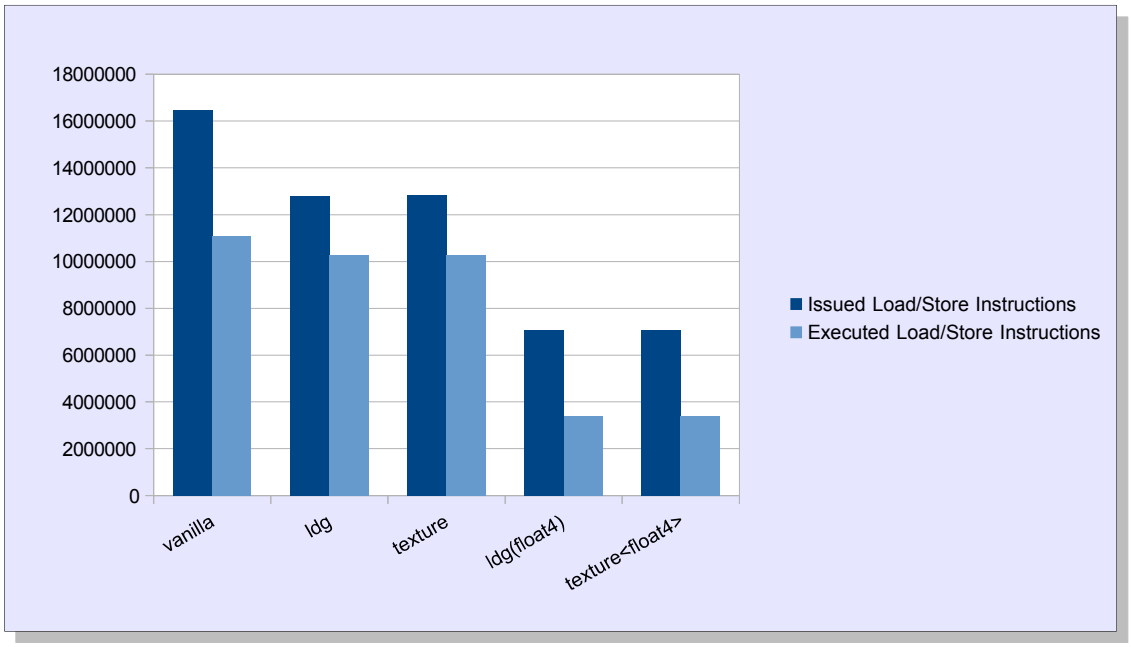
Instructions Per Clock

issued and executed



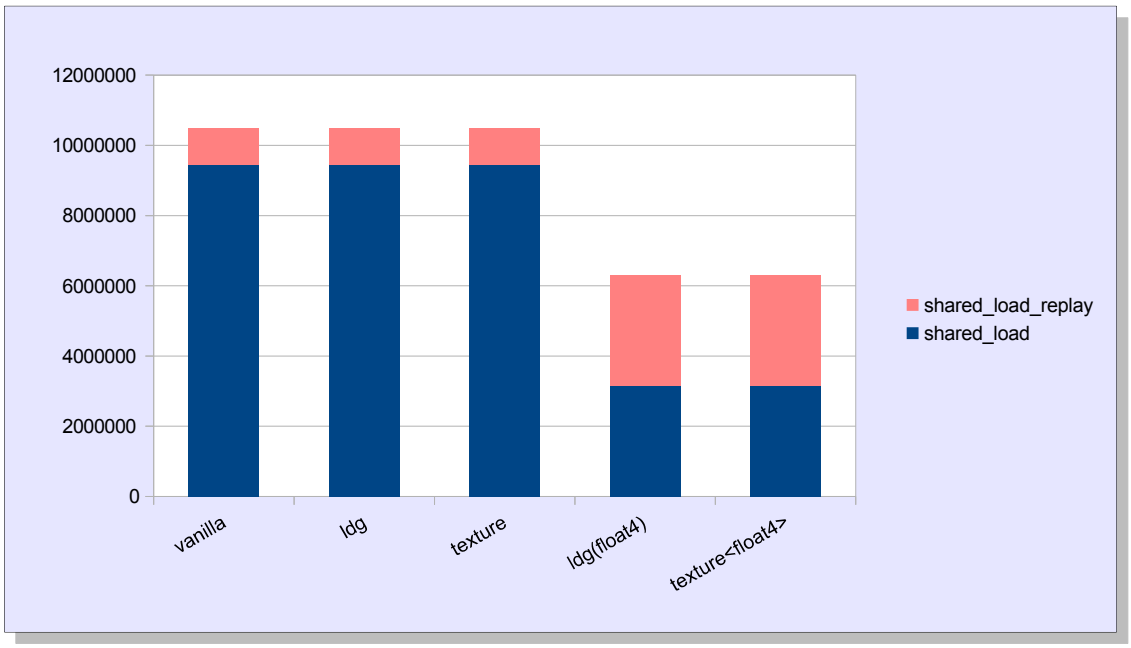
Load/Store Instructions

issued and executed



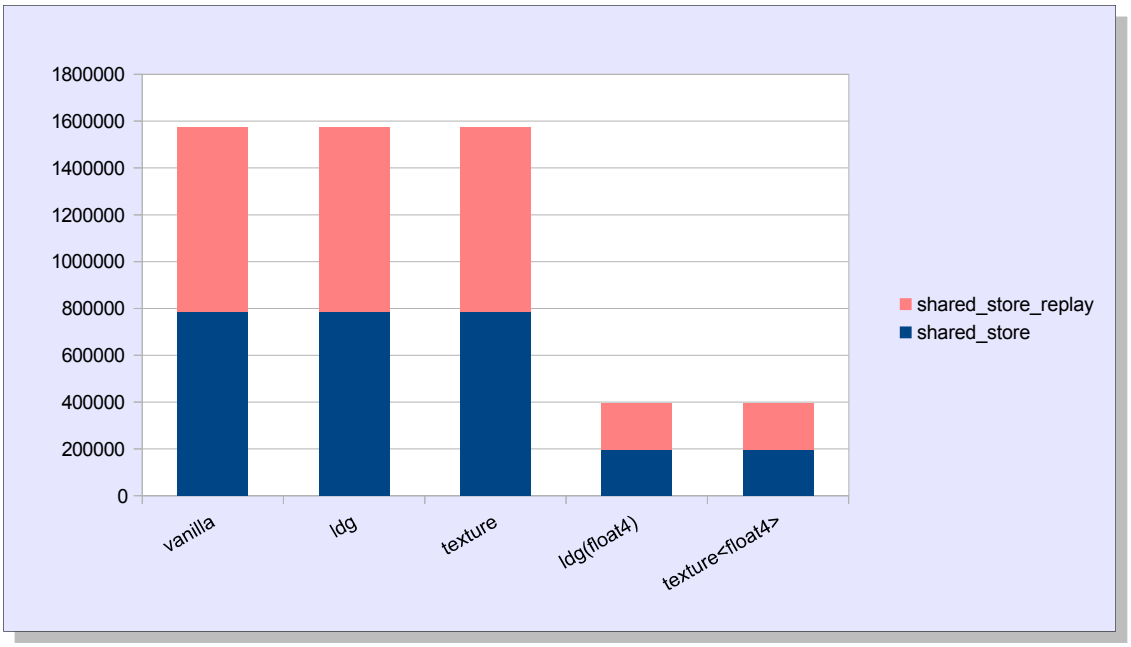
Shared Load

& shared load replay



Shared Store

& shared store replay



Conclusions

- Try using read-only data path for read-only data
- Try utilizing vector types

- Look at PTX
- Look at assembly

- Collect events
- Collect metrics

- Write tunable codes and auto-tune them

Sources


NVIDIA
CUDA C PROGRAMMING GUIDE
CUDA C BEST PRACTICES GUIDE
DG-05603-001_v5.5 | July 2013
Design Guide
CUDA COMPILER DRIVER NVCC
PROFILER USER'S GUIDE
CUDA RUNTIME API
v5.5 | July 2013
API Reference Manual
PARALLEL THREAD EXECUTION ISA
INLINE PTX ASSEMBLY IN CUDA
CUDA BINARY UTILITIES
DA-06762-001_v5.5 | July 2013
Application Note
TUNING CUDA APPLICATIONS FOR KEPLER
DA-06288-001_v5.5 | July 2013
Application Note


ICL UT
INNOVATIVE
COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE