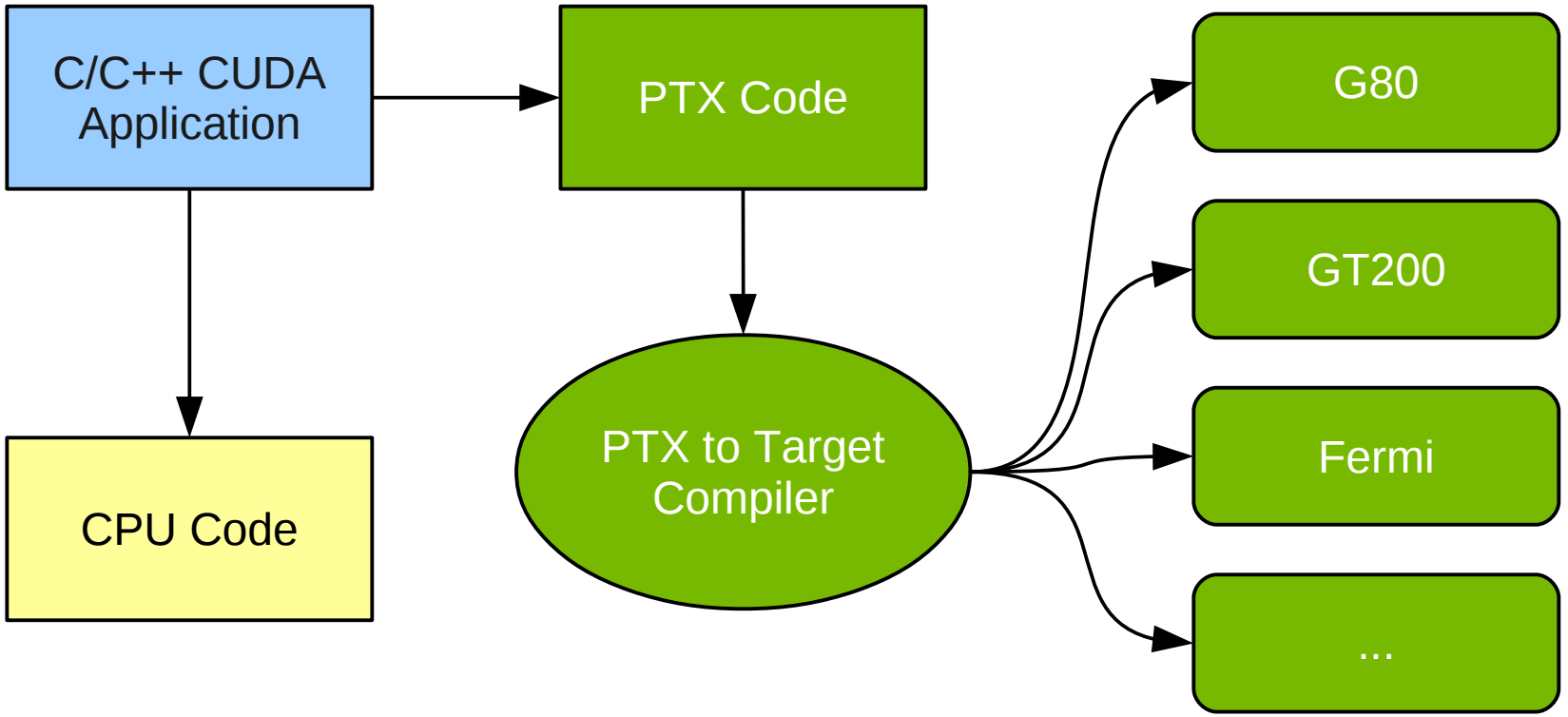
CUDA Driver API

Дмитрий Микушин, Александр Харламов

CUDA C Runtime



Доступ к PTX

--keep (-keep)

Keep all intermediate files that are generated during internal compilation steps.

```
__global__ void kernel ( float * data )
{
   int idx = blockIdx.x * blockDim.x + threadIdx.x ;

   data [idx] = idx;
}
```

Доступ к PTX

```
.entry _Z6kernelPf (.param .u32 __cudaparm__Z6kernelPf_data)
{
  .reg .u16 %rh<4>;
  .reg .u32 %r<8>;
  .reg .f32 %f<3>;
  .loc  14  6   0
  $LBB1__Z6kernelPf:
  .loc  14  10  0
  mov.u16   %rh1, %ctaid.x;  //
  mov.u16   %rh2, %ntid.x;   //
  mul.wide.u16  %r1, %rh1, %rh2;    //
  cvt.u32.u16   %r2, %tid.x;        //
  add.u32   %r3, %r2, %r1;   //
  cvt.rn.f32.s32    %f1, %r3;       //
  ld.param.u32  %r4, [__cudaparm__Z6kernelPf_data]; // id:14
  mul.lo.u32    %r5, %r3, 4;     //
  add.u32   %r6, %r4, %r5;   //
  st.global.f32     [%r6+0], %f1;   // id:15
  .loc  14  11  0
  exit;
  $LDWend__Z6kernelPf:
} // _Z6kernelPf
```

Запуск ядра на CUDA

```
float * a   = new float [N];
float * dev = NULL;

cudaMalloc( (void**)&dev, N * sizeof ( float ) );

dim3 threads = dim3( 512, 1 );
dim3 blocks  = dim3( N / threads.x, 1 );


kernel<<<blocks, threads>>> ( dev );
cudaThreadSynchronize();

cudaMemcpy(a, dev, N*sizeof(float), cudaMemcpyDeviceToHost);

cudaFree( dev );
```

Запуск ядра на Driver API

```
CUdevice   device;
CUcontext  context;
CUmodule   module;
CUfunction function;
CUdeviceptr pData;

float * pHostData = new float[N];

cuInit(0);

cuDeviceGetCount(&device_count);
cuDeviceGet( &device, 0 );

cuCtxCreate( &context, 0, device );

cuModuleLoad( &module, "hello.cuda_runtime.ptx" );
cuModuleGetFunction( &function, module, "_Z6kernelPf" );

cuMemAlloc( &pData, N * sizeof(float) );

// ...
```

Запуск ядра на Driver API

```
// ...
cuFuncSetBlockShape( function, N, 1, 1 );
cuParamSeti( function, 0, pData );
cuParamSetSize( function, sizeof(void *) );
cuLaunchGrid( function, 1, 1 );
cuMemcpyDtoH( pHostData, pData, N * sizeof( float) );
cuMemFree( pData );
```