CUDA 7.5

- 16-bit Floating-Point Storage
  - 2x larger datasets in GPU memory
  - Great for Deep Learning

- cuSPARSE Dense Matrix * Sparse Vector
  - Speeds up Natural Language Processing

- Instruction-Level Profiling
  - Pinpoint performance bottlenecks
  - Easier to apply advanced optimizations

- *Experimental* GPU Lambdas

Release Schedule:
- 7/6: Release Candidate
- ~Sept: Production Release
FP16 SUPPORT IN CUDA 7.5

fp16 Datatype and fp16↔fp32 Conversions

Store up to 2x larger models in GPU memory.

- Kernels can perform vector loads of fp16 data and convert to fp32
- New mixed-precision cublas_SgemmEx() routine supports 2x larger matrices.
- Great for Deep Learning

Reduce memory bandwidth requirements by up to 2x.
FP16 FORMAT

cuda_fp16.h

- half: \texttt{SxEEEEMMMMMMMMMM}
  - 1 Sign bit + 5-bit Exponent
  - 10-bit Mantissa
  - Range \~ 6*10^{-8} .. 6*10^{4}

- half2: \texttt{SxEEEEMMMMMMMMMM SxEEEEMMMMMMMMM}

CUSPARSE: (DENSE MATRIX) X (SPARSE VECTOR)

**Speeds up Natural Language Processing**

- `cusparse<T>gemvi()`
  - `y = α * op(A)*x + β*y`
  - `A = dense matrix`
  - `x = sparse vector`
  - `y = dense vector`

Sparse vector could be frequencies of words in a text sample
KERNEL LEVEL PROFILING - (PRIOR TO 7.5)

Compute and memory utilization are low

Instruction latency reasons at kernel level

Kernel perf limiter

<table>
<thead>
<tr>
<th>Stall Reasons</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory latency</td>
<td>332</td>
</tr>
<tr>
<td>Instruction latency</td>
<td>19</td>
</tr>
<tr>
<td>Synchronization</td>
<td>19</td>
</tr>
<tr>
<td>Load/Store</td>
<td>19</td>
</tr>
<tr>
<td>Instruction fetch</td>
<td>19</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel</td>
<td>3.932 ms (3.931 s)</td>
</tr>
<tr>
<td>Invocations</td>
<td>19</td>
</tr>
<tr>
<td>Importance</td>
<td>63.3%</td>
</tr>
</tbody>
</table>
KERNEL LEVEL PROFILING - (PRIOR TO 7.5)

Instruction execution counts
INSTRUCTION-LEVEL PROFILING - 7.5

Stall analysis via PC Sampling
INSTRUCTION-LEVEL PROFILING - 7.5

After Code Tuning: Stall - Execution Time goes from ~4ms to ~1.5ms

Synchronization stalls reduced, Memory dependency stalls are removed
*EXPERIMENTAL* GPU LAMBDAS

Define __device__ lambdas in host code

- Can pass __device__ lambdas to kernel launch
- Enable with nvcc flag: --expt-extended-lambda
- See Programming Guide for restrictions

```c
__device__ int result;
template <typename T>
__global__ void kernel(T in) { result = in(4); }

void foo(void) {
  int x = 30;
  kernel<<<1,1>>>( [=] __device__ (int in) {
    return in * x; // x captured by value
  });
}
```
*EXPERIMENTAL* C++ “PARALLEL FOR”

Build parallel algorithms with familiar coding style

**CPU** Sequential `for_each()`

```cpp
void saxpy(int N, float a, float *x, float *y) {
    using namespace std;
    auto r = range(0, N);
    for_each (begin(r), end(r), [=] (int i) {
        y[i] = a * x[i] + y[i];
    });
}
```

**CPU/GPU** Thrust Parallel `for_each()`

```cpp
void saxpy(int N, float a, float *x, float *y) {
    using namespace thrust;
    auto r = counting_iterator<int>(0);
    for_each (device, r, r+N, [=] __device__ (int i) {
        y[i] = a * x[i] + y[i];
    });
}
```

**CPU** Lambda

**GPU** Lambda
MORE C++ PARALLEL FOR LOOPS

GPU Lambdas Enable Custom Parallel Programming Models

Kokkos

Kokkos::parallel_for(n, KOKKOS_LAMBDA (int i) {
    y[i] = a * x[i] + y[i];
});

https://github.com/kokkos

RAJA

RAJA::forall<cuda_exec>(0, N, [=] __device__ (int i) {
    y[i] = a * x[i] + y[i];
});

https://e-reports-ext.llnl.gov/pdf/782261.pdf

Hemi

Hemi::parallel_for(0, N, [=] HEMI_LAMBDA (int i) {
    y[i] = a * x[i] + y[i];
});

http://github.com/harrism/hemi
### LOP3: ARBITRARY 3-INPUT LOGIC OPERATION

**Supported in PTX Inline Assembly (lop3.b32)**

- Supported on SM50 or higher
- \( D = \text{Oper}[\text{immLut}](A,B,C) \)

<table>
<thead>
<tr>
<th>ta</th>
<th>tb</th>
<th>tc</th>
<th>Oper 0 (False)</th>
<th>Oper 1 (ta &amp; tb &amp; tc)</th>
<th>Oper 2 (ta &amp; tb &amp; ~tc)</th>
<th>...</th>
<th>Oper 254 (ta</th>
<th>tb</th>
<th>tc)</th>
<th>Oper 255 (True)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![immLut](0x00, 0x80, 0x40, ..., 0xFE, 0xFF)

An awesome developer platform

GeForce GTX TITAN, TITAN Black, TITAN Z and TITAN X

GeForce TITAN series GPUs now support:

- TCC Mode
- Multi-process server (MPS)
- CUDA Stream Priorities
- All relevant nvidia-smi commands

*Most of these features will across the entire GeForce product line
WINDOWS REMOTE DESKTOP

» CUDA will work with Remote Desktop starting r352

» CUDA apps will be able to run “as a service” on Windows

» Will work across all GPU products supported on Windows
ADDITIONAL IMPROVEMENTS

See release notes and documentation for more details

- 64-bit API for cuFFT
- n-dimensional Euclidian norm floating-point math functions
- Bayer CFA to RGB conversion functions in NPP
- Faster double-precision square-roots (sqrt)
- CUDA Samples for the cuSOLVER library
- Nsight Eclipse Edition supported on POWER platform
- Nsight Eclipse Edition supports multiple CUDA Toolkit versions
x86_64 PLATFORM SUPPORT

- **Linux**
  - RHEL & CentOS 6, 7
  - Fedora 21 Workstation
  - SLES 11 SP3, 12
  - OpenSUSE 13.2
  - Ubuntu 14.04 LTS, 15.04

- **Windows**
  - 7, 8.1, 10, Server 2008 R2, 2012 R2

- **Mac OSX**
  - 10.9, 10.10, 10.11 (~Sept)

- ** Alternative Linux host compilers**
  - Clang 3.5, 3.6
  - Intel icc 15.0.0
  - PGI pgc++ 15.4(+)

- **CUDA 7.5 drops support for:**
  - Ubuntu 12.04 LTS on x86
  - cuda-gdb native debugging on Mac

- **CUDA 7.5 announces deprecation of:**
  - “Legacy” profiler
    - Will be dropped in a future release
    - Use nvprof instead
POWER8 PLATFORM SUPPORT

- Linux
  - RHEL 7.2
  - Ubuntu 14.04.3

- Alternative Linux POWER8 compilers
  - IBM xlc/xlC 13.1.x
UPCOMING WEBINARS

- **Getting Started with PGI OpenACC Compiler**
  - Tuesday, August 11 at 9AM Pacific
  - Register at [www.gputechconf.com/gtc-webinars](http://www.gputechconf.com/gtc-webinars)

- **Deep Learning Course**
  - Weekly Series of webinars and Office Hours - started yesterday and goes until Sept. 23rd
  - For more information and to register [https://developer.nvidia.com/deep-learning-courses](https://developer.nvidia.com/deep-learning-courses)