# CUDA Pro Tip: The Fast Way to Query Device Properties

CUDA applications often need to know the maximum available shared memory per block or to query the number of multiprocessors in the active GPU. One way to do this is by calling cudaGetDeviceProperties(). Unfortunately, calling this function inside a performance-critical section of your code lead to huge slowdowns, depending on your code. We found out the hard way when cudaGetDeviceProperties() caused a [20x slowdown in the Random Forests algorithm in cuML](https://github.com/rapidsai/cuml/issues/927).

Here is a very simple CUDA “pro tip”: cudaDeviceGetAttribute() is a much faster way to query device properties.

## Just the Facts You Need

Typically you don’t need to know all the properties of the GPU you are running on. Often you just need one or two, like the maximum block size, the number of multiprocessors, or the maximum shared memory per block. But cudaGetDeviceProperties() gives you everything, whether you need it or not. So it’s usually overkill to call this function, and you will pay for it, because some device properties requires PCIe reads to query, which is expensive.

In contrast, cudaDeviceGetAttribute gives you one attribute per call—just the one you ask for. That makes it much faster for most attributes. We are talking orders of magnitude faster: nanoseconds vs. milliseconds. Let’s get some numbers.

## Benchmarking Device Attribute Queries

We wrote a simple benchmark to compare the performance of cudaGetDeviceProperties() and cudaDeviceGetAttribute(). The timings were captured using a single Tesla V100 in an NVIDIA DGX-1 with driver v410.79 and CUDA Toolkit 10.0. The benchmark compares getting a full cudaDeviceProp struct using cudaGetDeviceProperties() to just querying the maximum shared memory per block and number of multiprocessors using two calls to cudaDeviceGetAttribute(). It averages the run-time over 25 iterations. Here’s the test code for cudaGetDeviceProperties().

auto start = chrono::high\_resolution\_clock::now();

cudaDeviceProp prop;

for(int i = 0; i < 25; ++i) {

  cudaGetDeviceProperties(&prop, devId);

}

auto end = chrono::high\_resolution\_clock::now();

cout << "cudaGetDeviceProperties -> "

    << chrono::duration\_cast<chrono::microseconds>(end - start).count() / 25.0

    << "us" << endl;

**Output:**

cudaGetDeviceProperties -> 1150.56us

Here’s the test code for cudaDeviceGetAttribute().

int smemSize, numProcs;

auto start = chrono::high\_resolution\_clock::now();

for (int i = 0; i < 25; ++i) {

  cudaDeviceGetAttribute(&smemSize,

    cudaDevAttrMaxSharedMemoryPerBlock, devId);

  cudaDeviceGetAttribute(&numProcs,

    cudaDevAttrMultiProcessorCount, devId);

}

auto end = chrono::high\_resolution\_clock::now();

cout << "cudaDeviceGetAttribute -> "

     << chrono::duration\_cast<chrono::microseconds>(end - start).count() / 25.0

     << "us" << endl;

**Output:**

cudaDeviceGetAttribute -> 0.08us

As you can see, `cudaDeviceGetAttribute() is four orders of magnitude faster than cudaGetDeviceProperties() for these attributes: 80 nanoseconds vs. 1.15 milliseconds. You can find the code used in this experiment [here](https://gist.github.com/teju85/9521e2224f0c31f71a93b593ff64e8da).

## Caution: Some Attributes are Expensive

As we mentioned, some device properties require expensive PCIe reads, which is why cudaGetDeviceProperties() is slow. For the same reason, the following properties are much slower than others to query using cudaDeviceGetAttribute(): cudaDevAttrClockRate, cudaDevAttrKernelExecTimeout, cudaDevAttrMemoryClockRate, and cudaDevAttrSingleToDoublePrecisionPerfRatio.