Chapter 4: GPU programming

Version 1: OpenCL

In the OpenCL version, the change is of course equally trivial:

```
const char* kernel_source =
R""(
__kernel void mykernel(__global float* r, __global const float* d, int n) {
    int i = get_global_id(0);
    int j = get_global_id(1);
    if (i >= n || j >= n)
        return;
    float v = HUGE_VALF;
    for (int k = 0; k < n; ++k) {
        float x = d[n*j + k];
        float y = d[n*k + i];
        float z = x + y;
        v = min(v, z);
    }
    r[n*j + i] = v;
}
)"";
```

What is perhaps much more interesting is the fact that we get very similar performance improvements across a wide variety of platforms. Here are examples of the benchmarks of this code and the previous version (total running time for two iterations, input size n = 4000):

GPU:	AMD Radeon R9 M390	NVIDIA Quadro K2200	NVIDIA Quadro K2200
OS:	macOS	Linux	Linux
API:	OpenCL	OpenCL	CUDA
V0:	9.2 s	14.9 s	14.4 s
V1:	1.9 s	3.9 s	3.6 s
speedup:	**4.9 ×**	**3.8 ×**	**4.0 ×**

While the APIs are superficially different, and the technical details of the hardware differ a lot too, the same key principles related to the memory access patterns still hold.