netherlands eScience center

# Kernel Tuner

## Faster development of faster GPU code

### Intro

Kernel Tuner is an easy to use tool for testing and auto-tuning CUDA, OpenCL, and C kernels.

Kernel Tuner simplifies the software development of optimized GPU programs, by enabling Python-based unit testing of GPU code and supporting output verification of all benchmarked kernels during tuning.

Easy to use also means no extensive changes and no new dependencies are required in the kernel code. The kernels can still be compiled and used as normal from any host programming language.

Kernel Tuner aims to provide a comprehensive solution for auto-tuning GPU programs, supporting tuning of user-defined parameters in both host and device code, as well as many optimization strategies to speed up the tuning process.

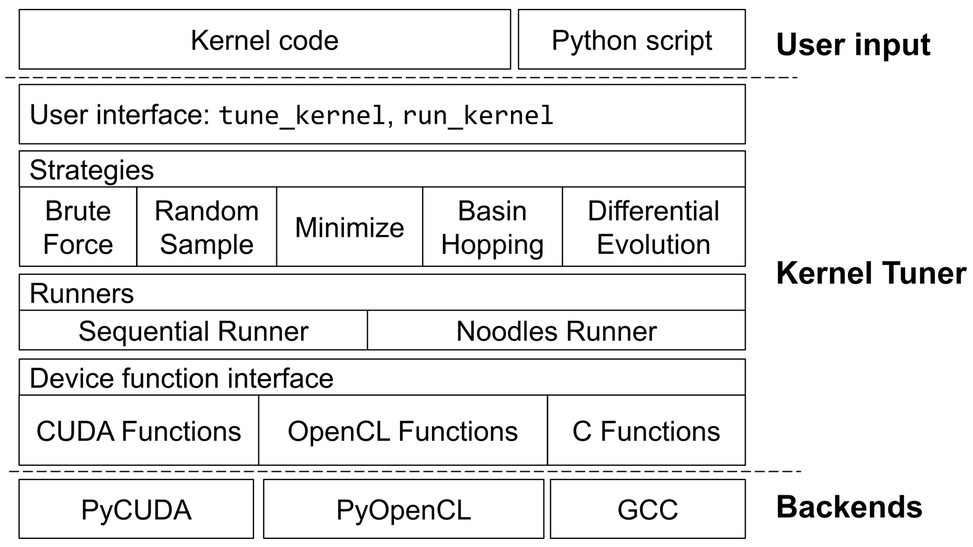
### Test GPU code from Python

```python
import numpy
from kernel_tuner import run_kernel

def test_vector_add():

    kernel_string = """
    __global__ void vector_add(float *c, float *a, float *b, int n) {
        int i = blockIdx.x * block_size_x + threadIdx.x;
        if (i<n) {
            c[i] = a[i] + b[i];
        }
    }
    """

    size = 10000000
    a = numpy.random.randn(size).astype(numpy.float32)
    b = numpy.random.randn(size).astype(numpy.float32)
    c = numpy.zeros_like(b)
    n = numpy.int32(size)
    args = [c, a, b, n]

    params = {"block_size_x": 512}
    answer = run_kernel("vector_add", kernel_string, size, args, params)

    assert numpy.allclose(answer[0], a+b, atol=1e-8)
```

Open Source repository with many examples, extensive documentation, and tutorials available at:

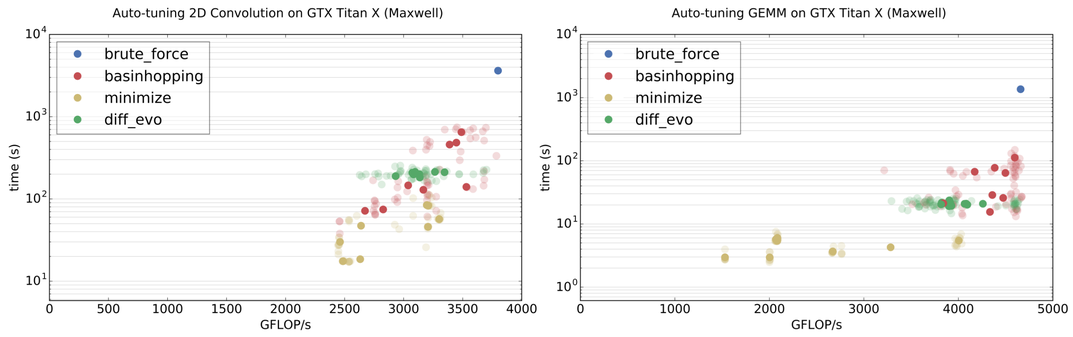
**github.com/benvanwerkhoven/kernel_tuner**

### Design




### Save time auto-tuning

Kernel Tuner supports many search strategies that can save a lot of time spent on auto-tuning. Using any of the strategies is easy, you only need to specify to Kernel Tuner's `tune_kernel()` function which strategy you would like to use, for example: `strategy="basinhopping", method="TNC"`.




The figures above show the performance of the best configuration found during auto-tuning versus the time spent to find it using different search strategy+method combinations for auto-tuning 2D Convolution and GEMM (matrix-matrix multiply) on Nvidia GTX Titan X.

## Used in eScience projects:

- Real-time detection of neutrinos (with Nikhef and Leiden University)
- Jungle Computing for Forensics (with NFI and VU University Amsterdam)
- 3D Geospatial Data Exploration
- Localization Microscopy (with Delft University of Technology)
- Distributed Radio Astronomical Computing (with ASTRON)