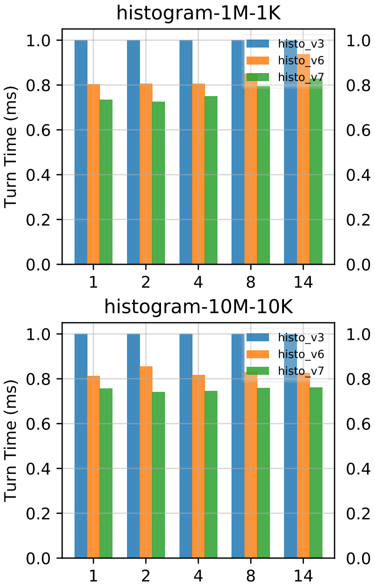
BLonD Meeting

Konstantinos Iliakis

October 13, 2017

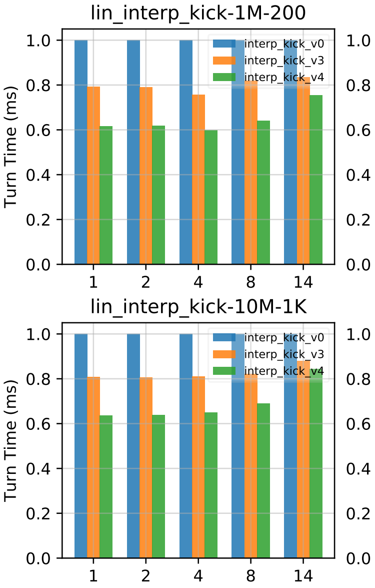
Table of Contents

Histogram



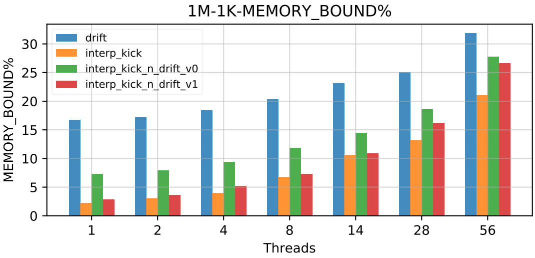
- histo_v3: Original implementation
- histo_v6: New version compiled with gcc5.1
- histo_v7: New version compiled with icc17
- Optimization: Loop tiling + Speculative execution
- 20-25% average speedup

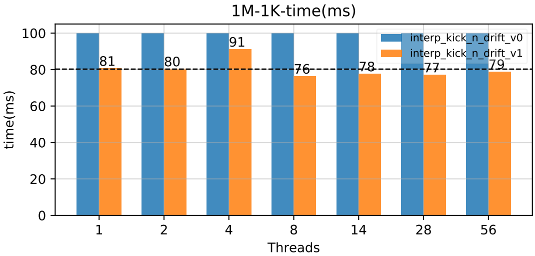
Linear Interpolation



- interp_kick_v0: Original implementation
- interp_kick_v3: Pre-calculate part of the loop independent of the particle coordinates
- interp_kick_v4: + Loop tiling + Speculative execution
- all versions compiled with gcc5.1
- 35-40% average speedup

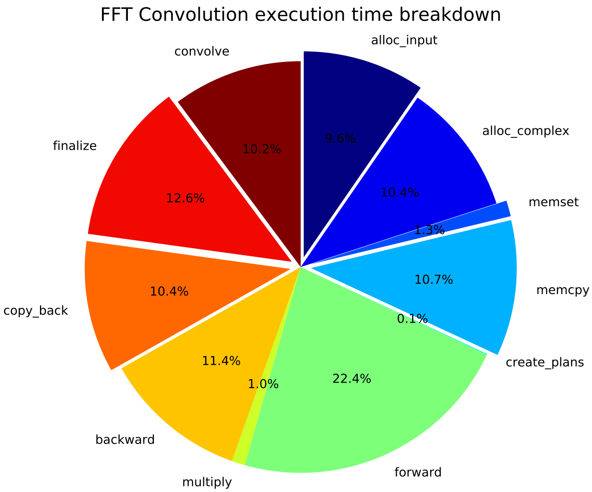
Overlapping kick and drift





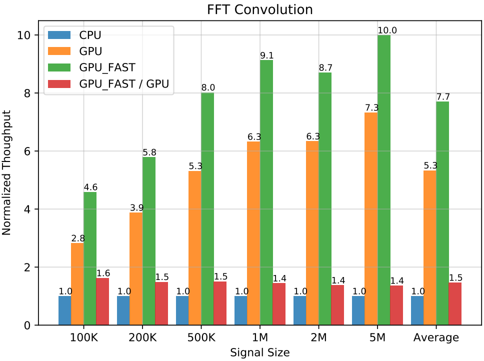
- v0: calculate kick and drift separately
- v1: overlap the calculation of kick and drift
- drift is memory bound while interp_kick is not
- The overlap gives better results: 20% average Speedup
- Considering the previous slide: 48-52% total Speedup

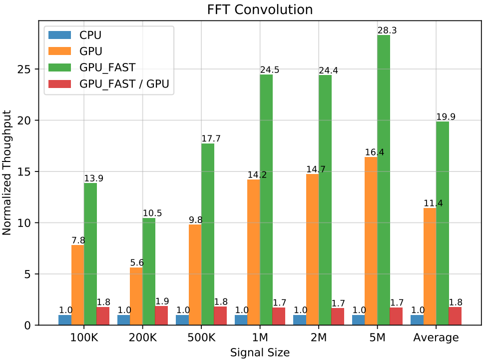
GPU FFT Convolution Time Breakdown



- The separated parts can be eliminated if the arrays reside on the GPU (44.6%)
- Convolve prepares the data for the CUDA invocation
- alloc_complex can be skipped too by reusing memory
- Actual computations: 34.8%

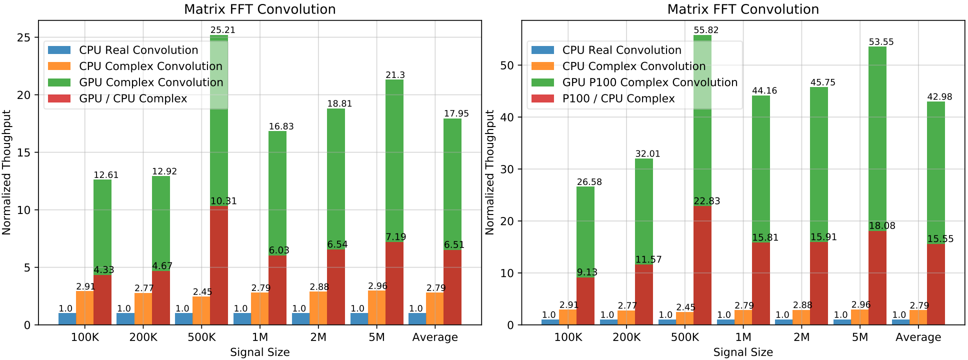
CPU/GPU FFT Convolution Benchmark





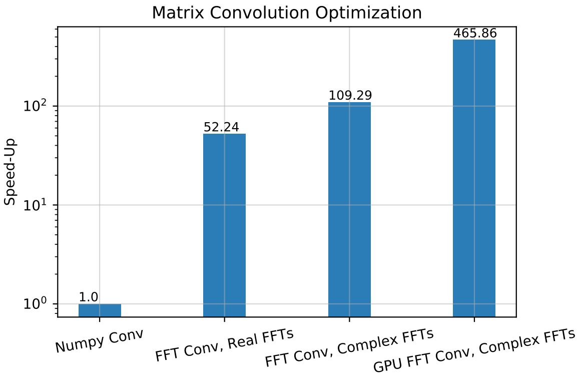
- Figure on the left: Kepler K20x, 2688 SP/ 896 DP cores, 3.95 SP/ 1.31 DP TFlops
- Figure on the right: Pascal P100, 3584 SP/ 1792 DP cores, 10.6 SP/ 5.3 DP Tflops

CPU/GPU Matrix Convolution Benchmark



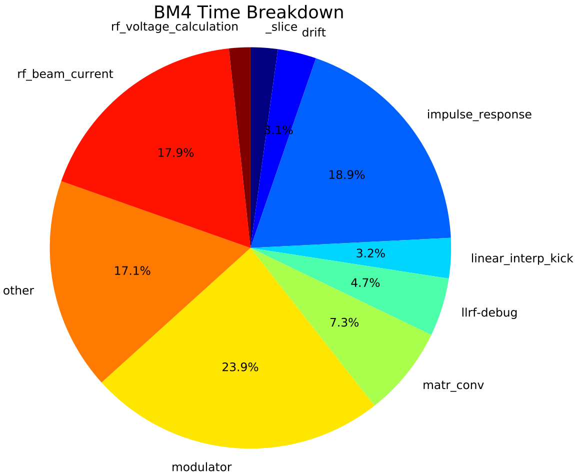
- Matrix Convolution: Convolution of two complex signals, requires 4 Real-Real Convolutions
- K20x (left) 6.51 Avg. Speedup over the best CPU version
- P100 (right) 15.55 Avg. Speedup over the best CPU version
- K20X Pie Chart P100 Pie Chart

Benchmark_4 Total Speedup



- FFT Size: ˜100K points
- x52.4 from `np.covolve()` to `scipy.signal.fftconvolve()`
- x2.1 from real to complex FFTs
- x4.26 from CPU to GPU K20x

Time Breakdown



- Convolution is no longer the bottleneck
- Other functions such as `modulator()`, `impulse_response()`, `rf_beam_current()` are in the critical path
- Configuration: 100K particles, 144 bunches, 46200 Slices

Python PAPI Library

Motivation

- Need a way to extract info about processor counters in python.
- Counters are meaningful only when combined in metrics.

Implementation

1. Build a C library (backend) that uses the PAPI (Performance API) interface to extract native and preset events.
2. Expose the C library to Python with ctypes.
3. Build a Python module with preset metrics that will communicate with the backend to read the necessary counters and compute the requested metrics.
4. Metrics found in Intel64 and IA-32 Architecture Optimization Manual

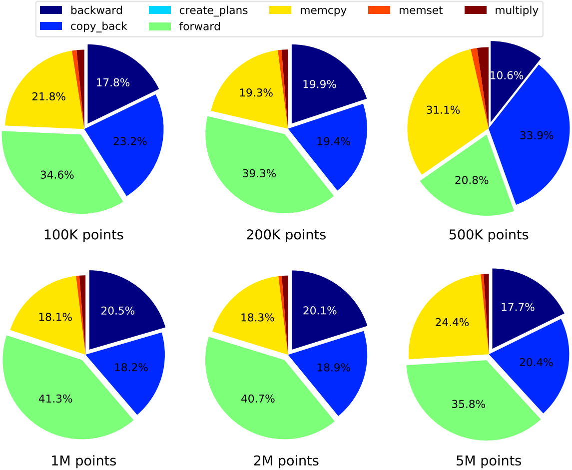
```
@papiprof(['CPI', 'MEM_BOUND', 'CORE_BOUND'])
def foo():
    compute()

foo()
papiprof_report_metrics()
```

Simple Use-Case

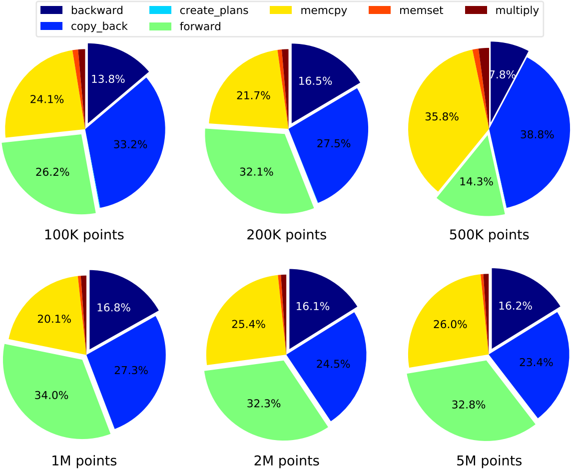
Thank you for your attention

K20x Matrix Convolution Time Breakdown



Back to Back to presentation

P100 Matrix Convolution Time Breakdown



Back to Back to presentation