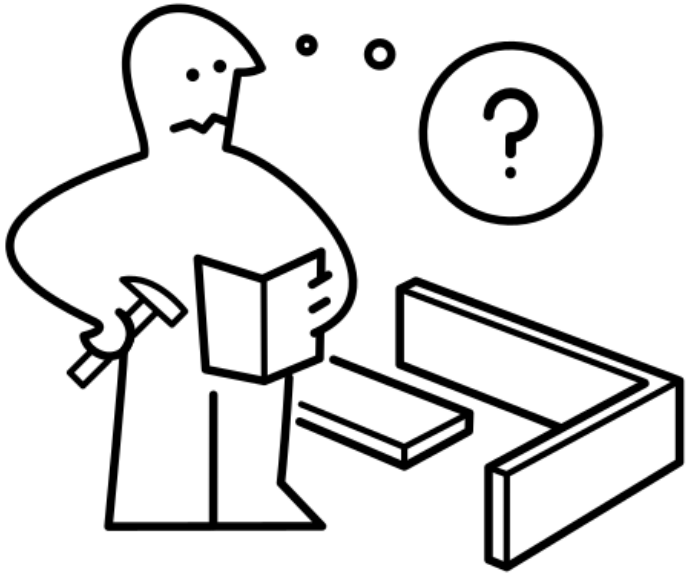
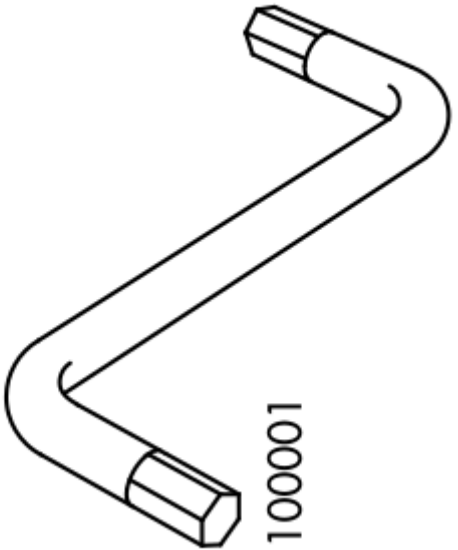
# How poor can a stock perform?

Markus Bilz, 1705042



Greatest, likely loss?




Value at Risk
Simulation

(c) ikea.com

# Value at Risk using a Monte Carlo simulation

```
repeat n times -> first kernel
    repeat t times
        generate normal distributed number
        update interim price
    save end price to path array
extract the 0th rank -> second kernel
print results
```

# 1st kernel

## (generating random prices)

# specification

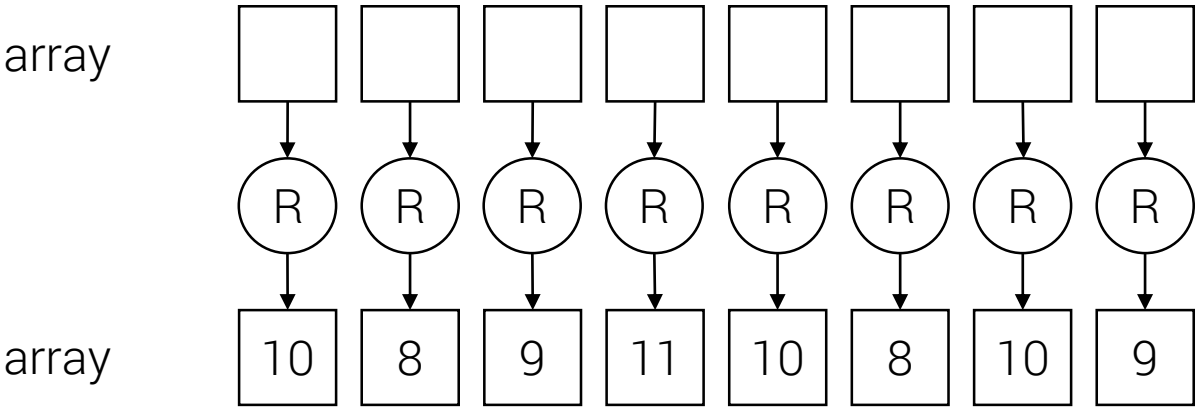
trivial problem / no interaction between threads

All threads run the same code / no thread divergence

Implicitly tiled

...

# algorithm




own drawing inspired by Ade Miller cppcon 2014

# 2nd kernel

# (extracting the minimal price)

# specification

non trivial problem / dependence between threads

explicitly tiled

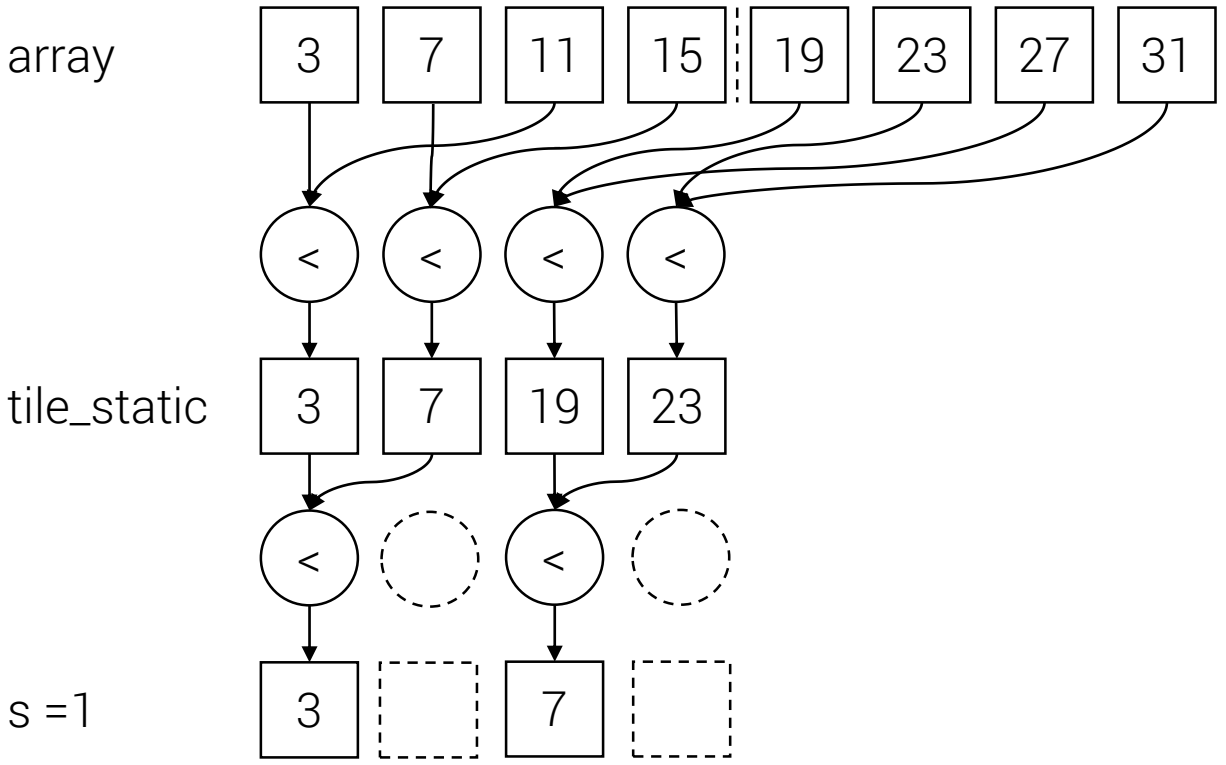
reduce algorithm

potential bank conflicts / idle threads

race conditions resolved through barriers

...

# algorithm




own drawing inspired by Ade Miller cppcon 2014

# performance evaluation

# gpu specification

## gpu specification

NVIDIA GeForce 940MX

## misc

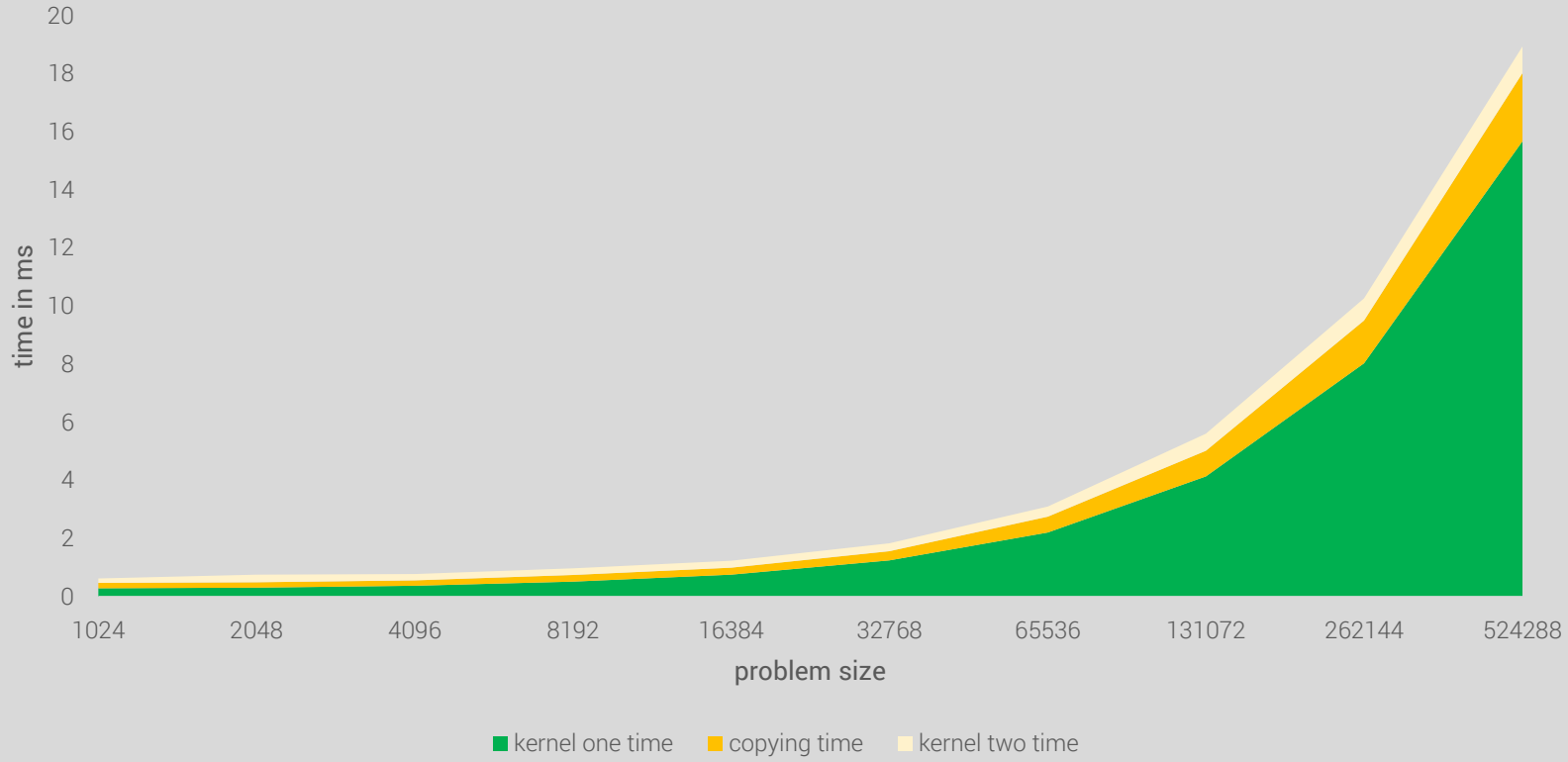
using wakeup call to prevent JIT and lazy initialization

using std::chrono::steady_clock

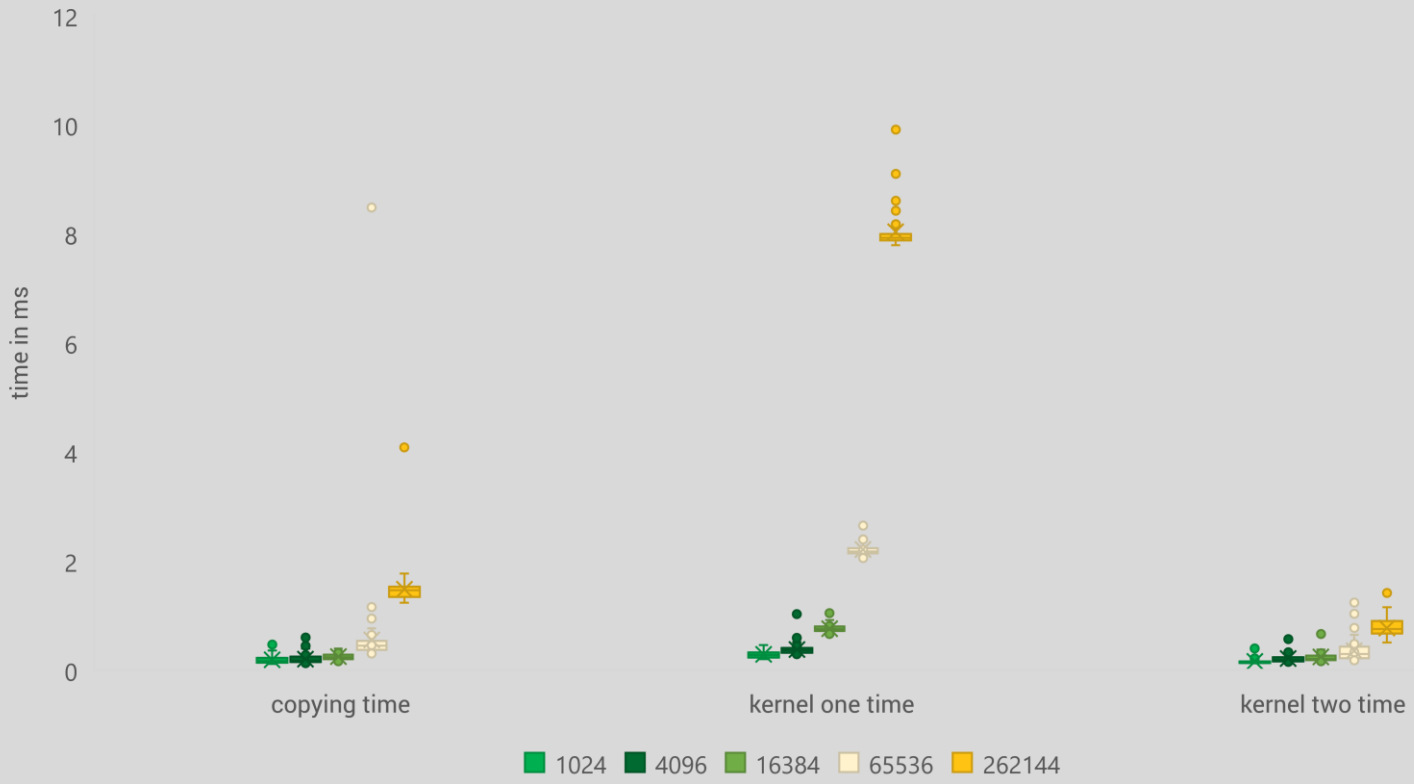
using tile_size with multiple of two

using templated functions for changing tile_size

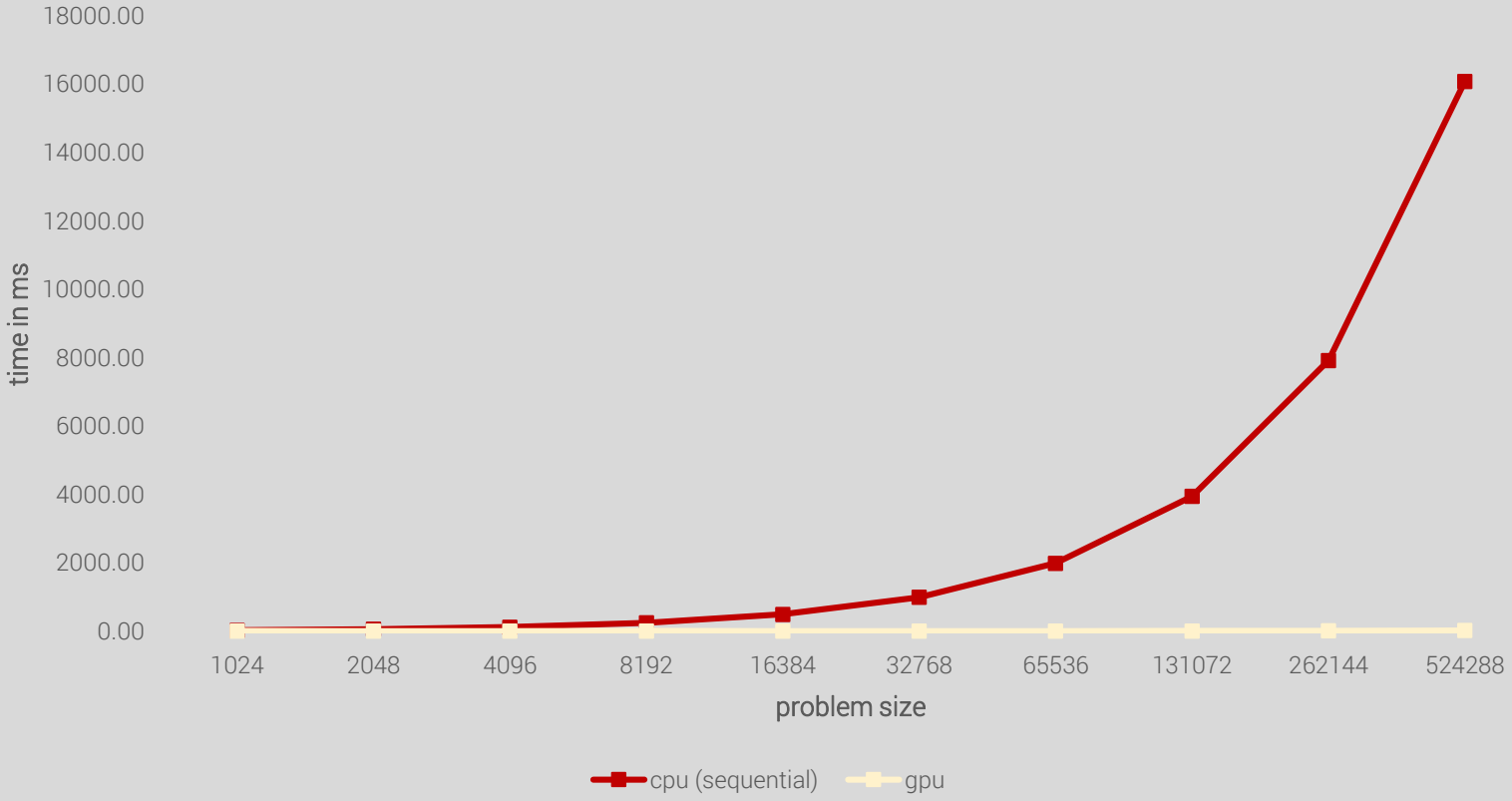
accelerator_view::wait()



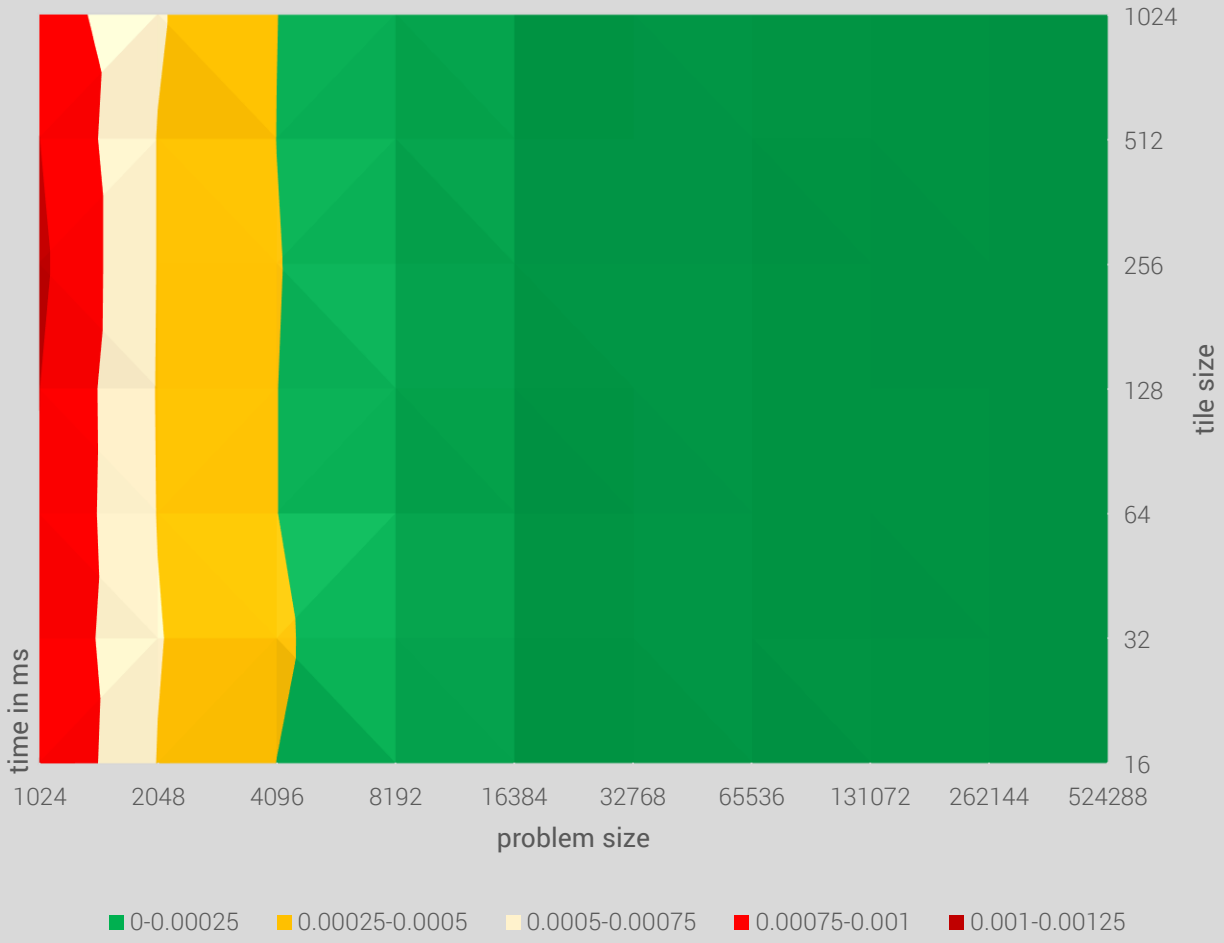

(avg. of 100 runs, ts = 1024)




(100 runs, ts = 1024)


18000.00
16000.00
14000.00
12000.00
10000.00
8000.00
6000.00
4000.00
2000.00
0.00
time in ms
1024
2048
4096
8192
16384
32768
65536
131072
262144
524288
problem size
cpu (sequential)
gpu

(avg. of 3 runs)




(avg. of 100 runs)