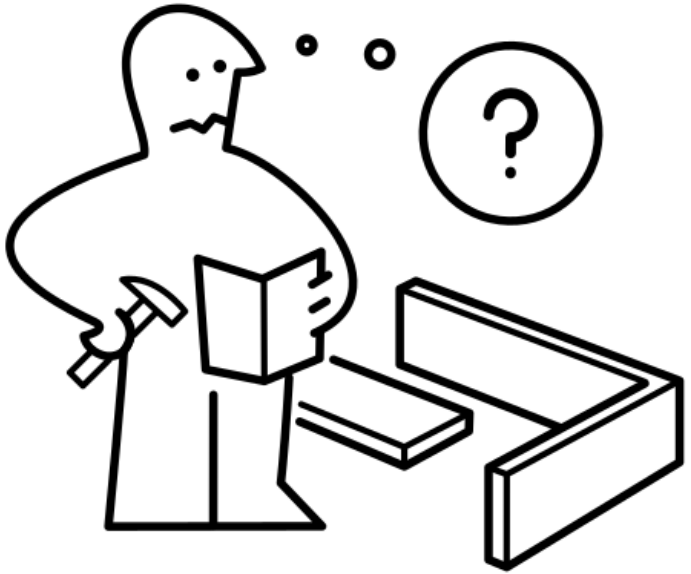
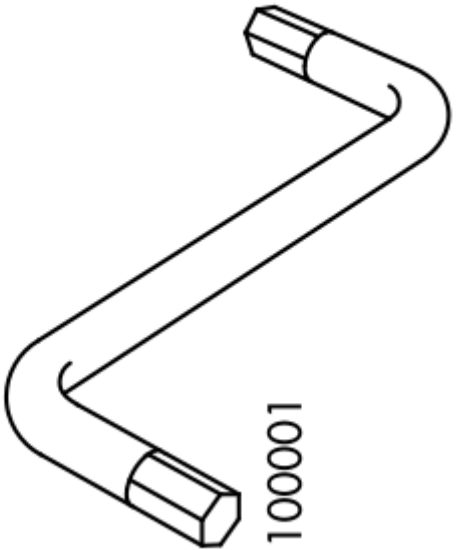
How poor can a stock perform?

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 nth rank -> second kernel
optional: scale value at risk to holding period
print results
```

1st kernel

(generating random prices)

specification

trivial problem / no interaction between threads

transform algorithm

All threads run the same code / no thread divergence

...

{demo}

2nd kernel

(extracting the minimal price)

specification

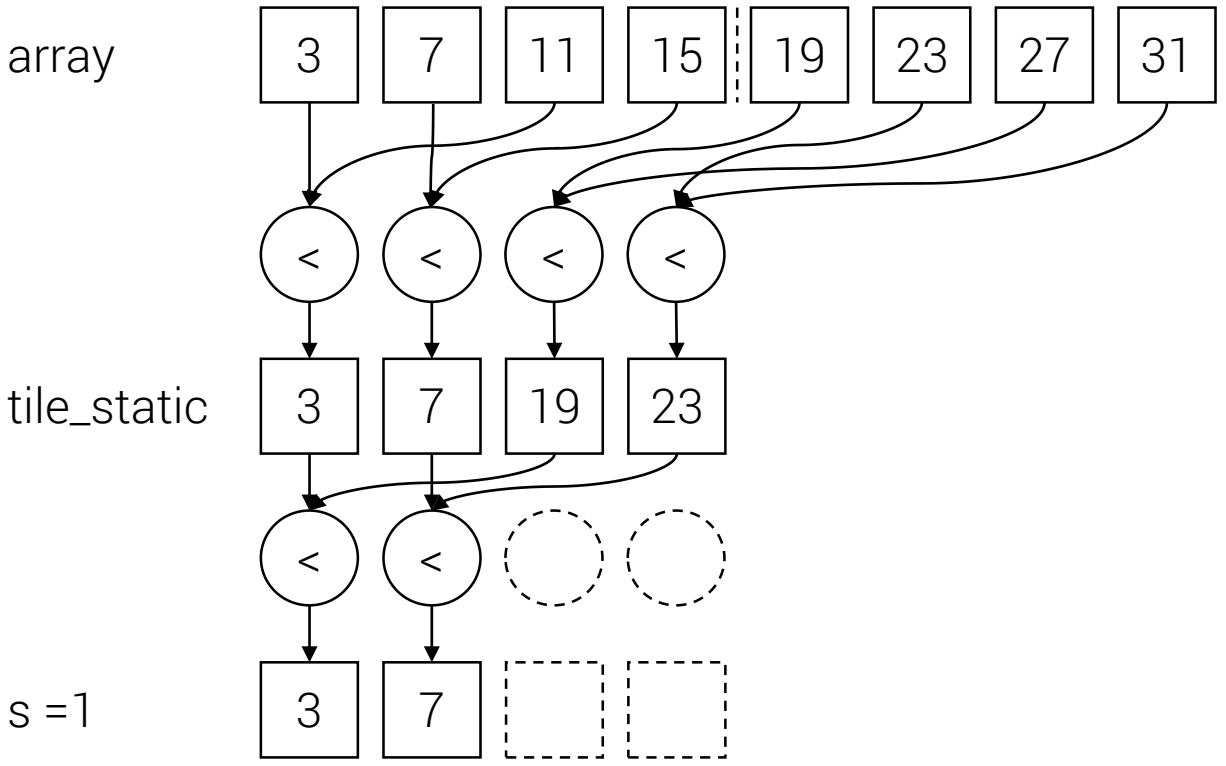
non trivial problem / dependence between threads

reduce algorithm

potential bank conflicts / idle threads

...

algorithm



Drawing adapted from Ade Miller cppcon 2014

{demo}

performance evaluation

gpu specification gpu specification

NVIDIA GeForce 940MX

3 Mb dedicated memory

384 cores

misc

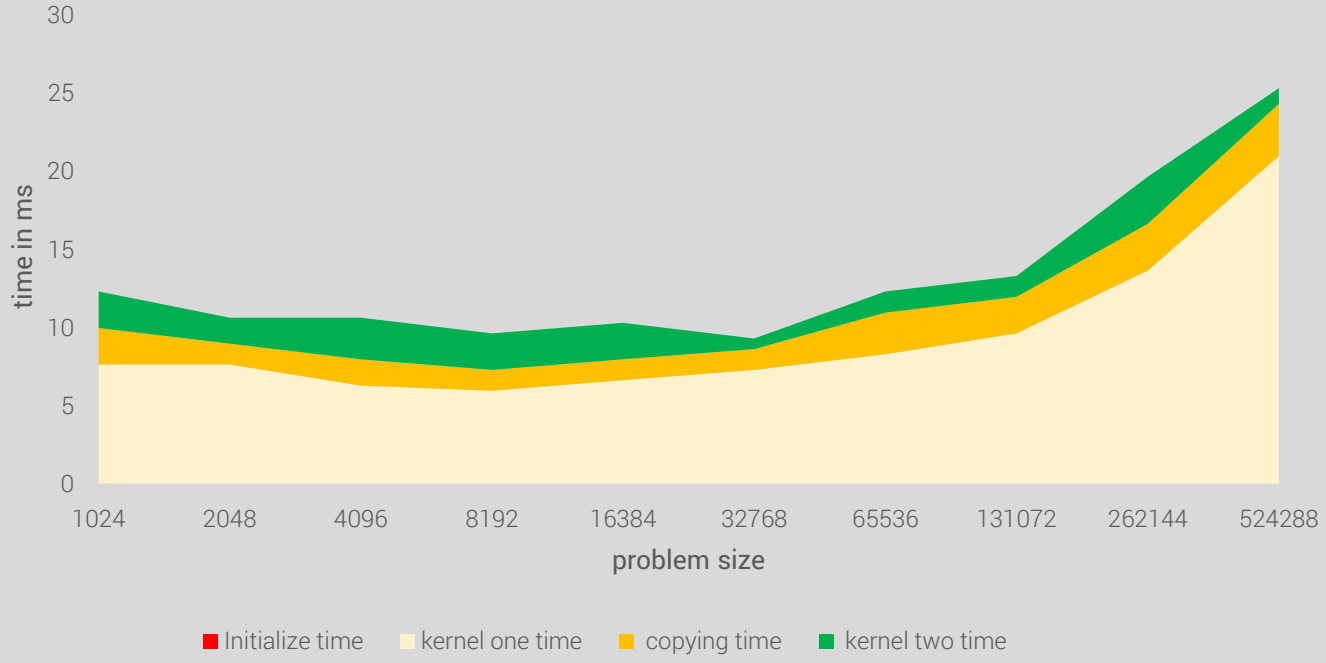
using wakeup call to prevent JIT and lazy initialization

using std::chrono::steady_clock

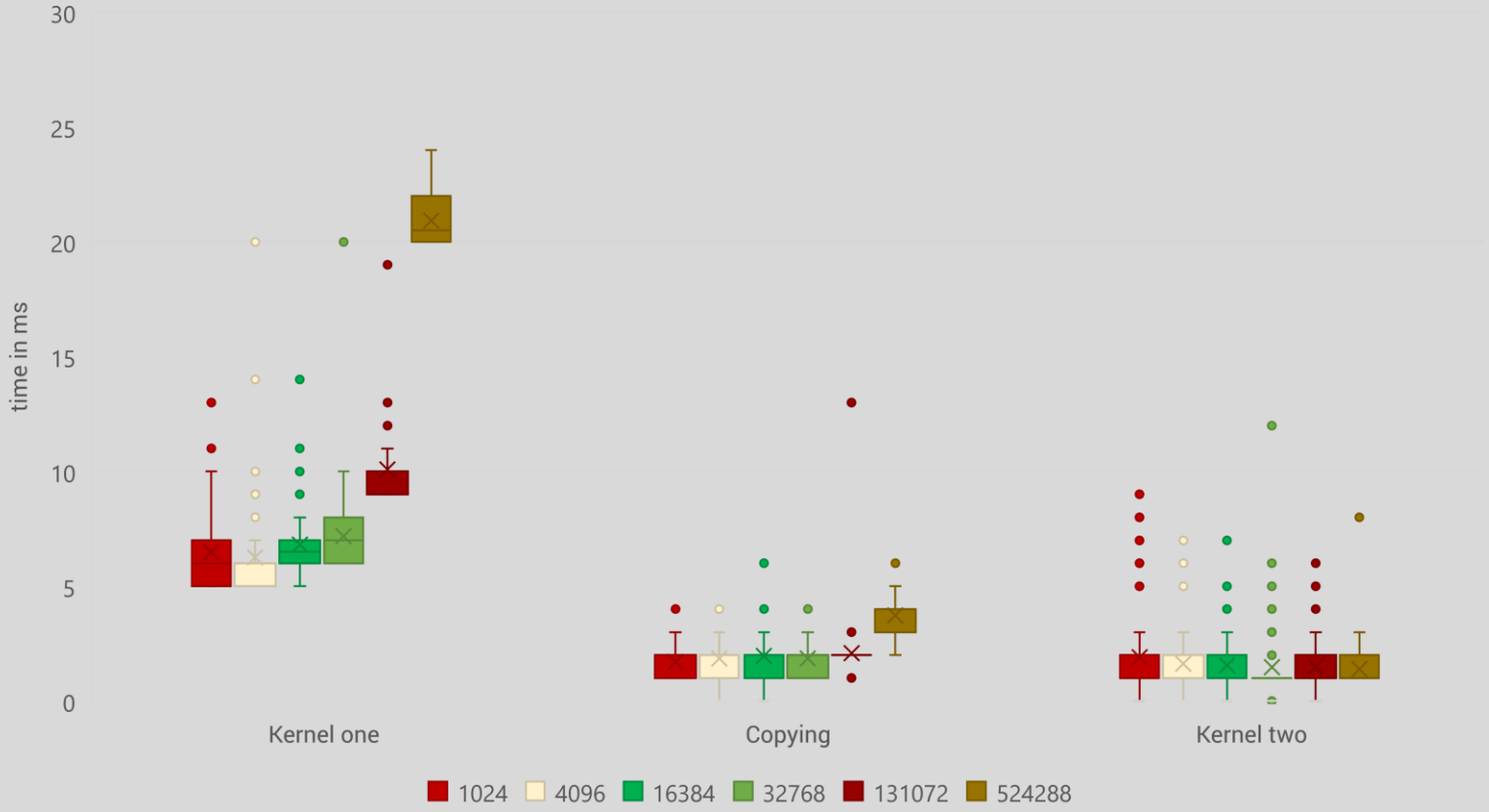
using tile_size with multiple of two

using templated functions for changing tile_size

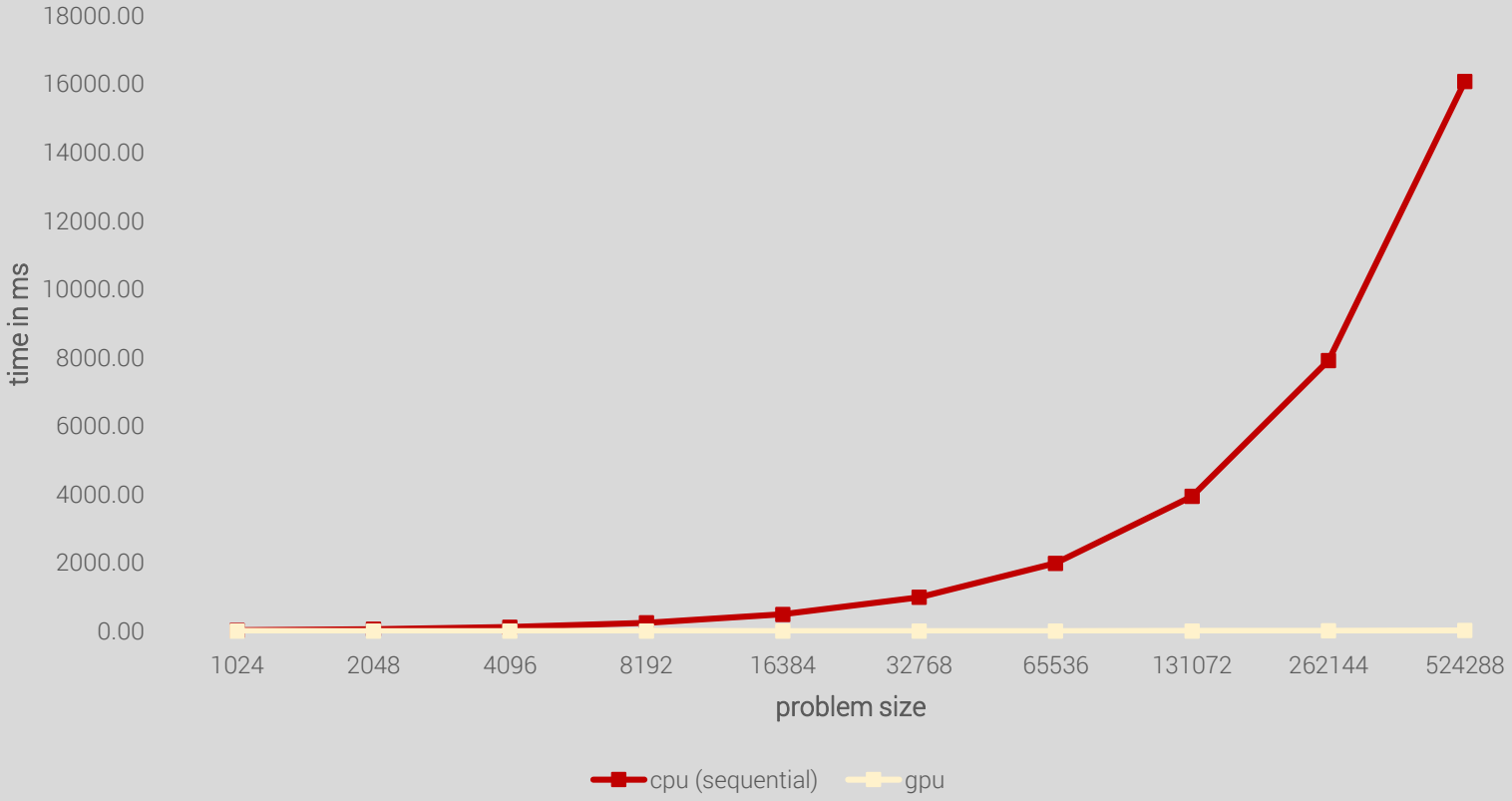
accelerator_view::wait()


30
25
20
15
10
5
0
time in ms
1024
2048
4096
8192
16384
32768
65536
131072
262144
524288
problem size
Initialize time
kernel one time
copying time
kernel two time

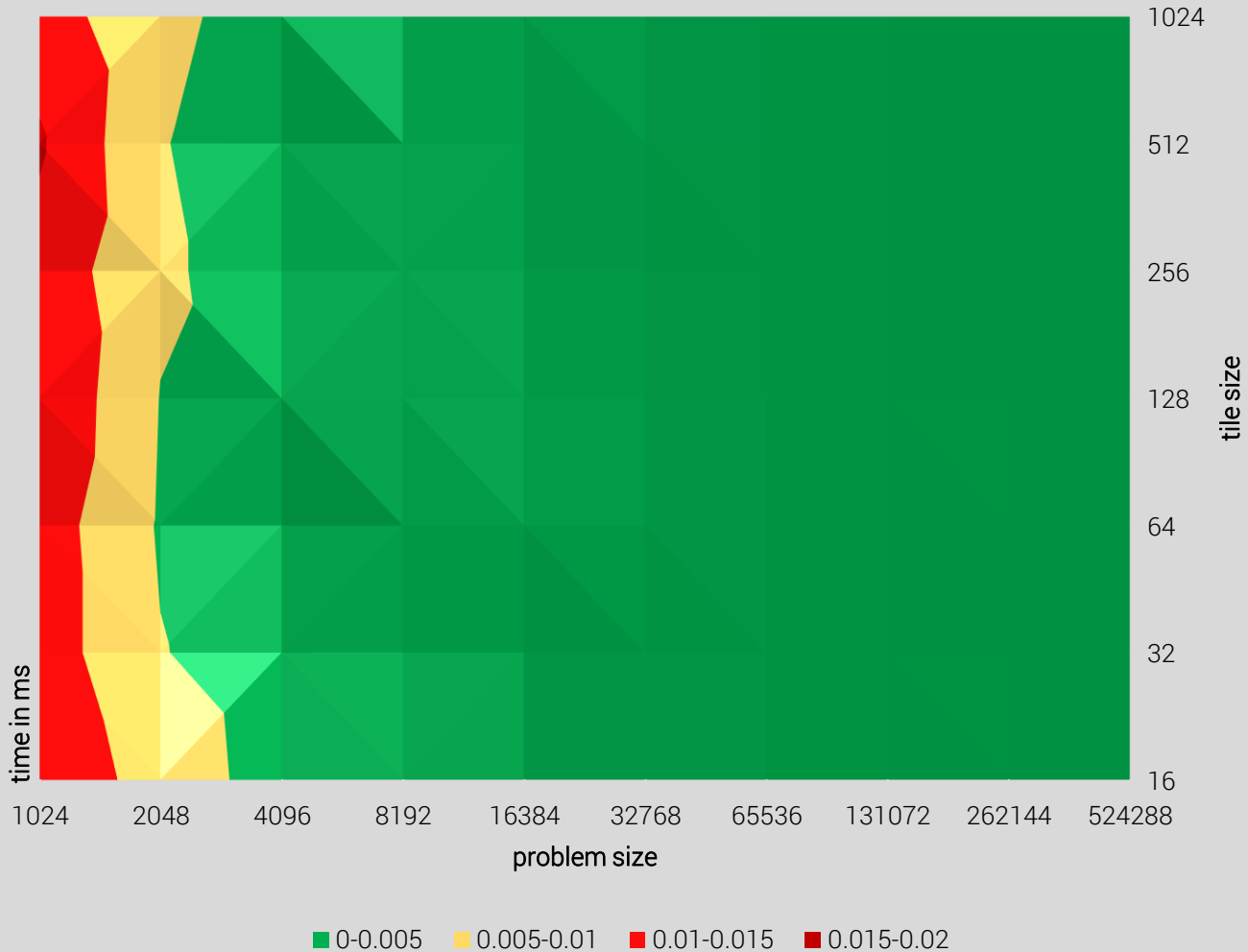
(avg. of 3 runs)



(100 runs, ts = 16)


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)


1024
512
256
128
64
32
16
tile size
time in ms
1024
2048
4096
8192
16384
32768
65536
131072
262144
524288
problem size
0-0.005
0.005-0.01
0.01-0.015
0.015-0.02

(avg. of 3 runs)