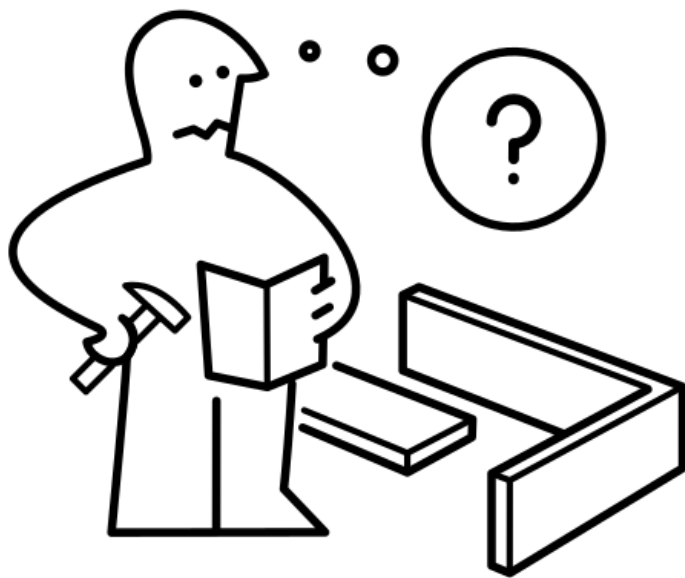
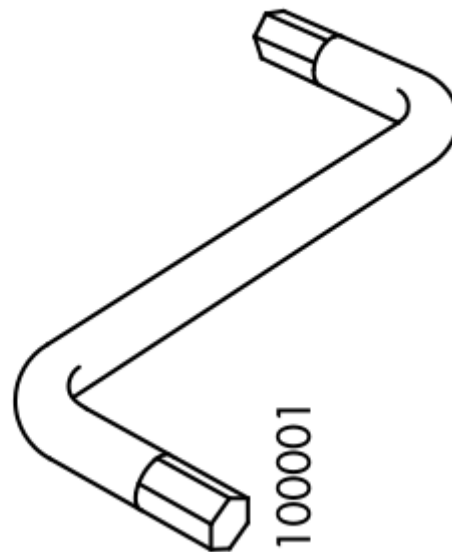


# How worse can a stock perform?

Markus Bilz, 1705042



Greatest, likely loss?



Value at Risk  
Simulation

# 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
scale value at risk to holding period
print results
```

1<sup>st</sup> kernel

(generating random prices)

# specification

trivial problem / no interaction between threads

All threads run the same code / no thread divergence...

{demo}

2<sup>nd</sup> kernel

(extracting the minimal price)

{demo}



# specification

non trivial problem / dependence between threads

potential bank conflicts / idle threads

solved through efficient reduction

...

performance evaluation

## **gpu specification**      gpu specification

NVIDIA GeForce 940MX

3 Mb dedicated memory

384 cores

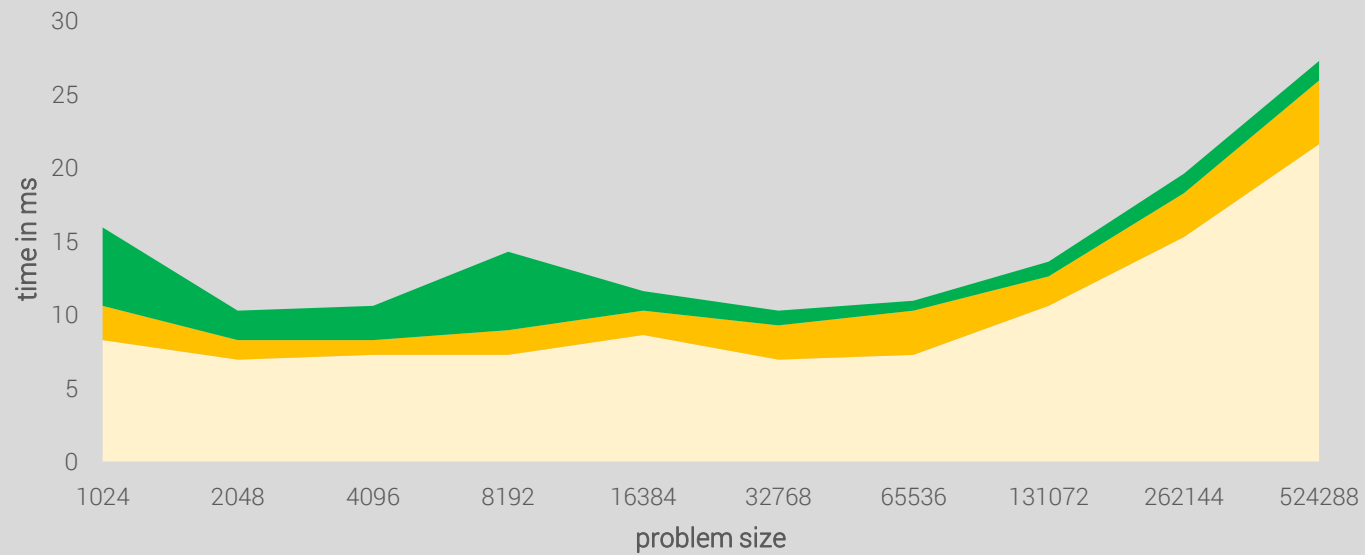
### **misc**

using wakeup call to prevent JIT and lazy initialization

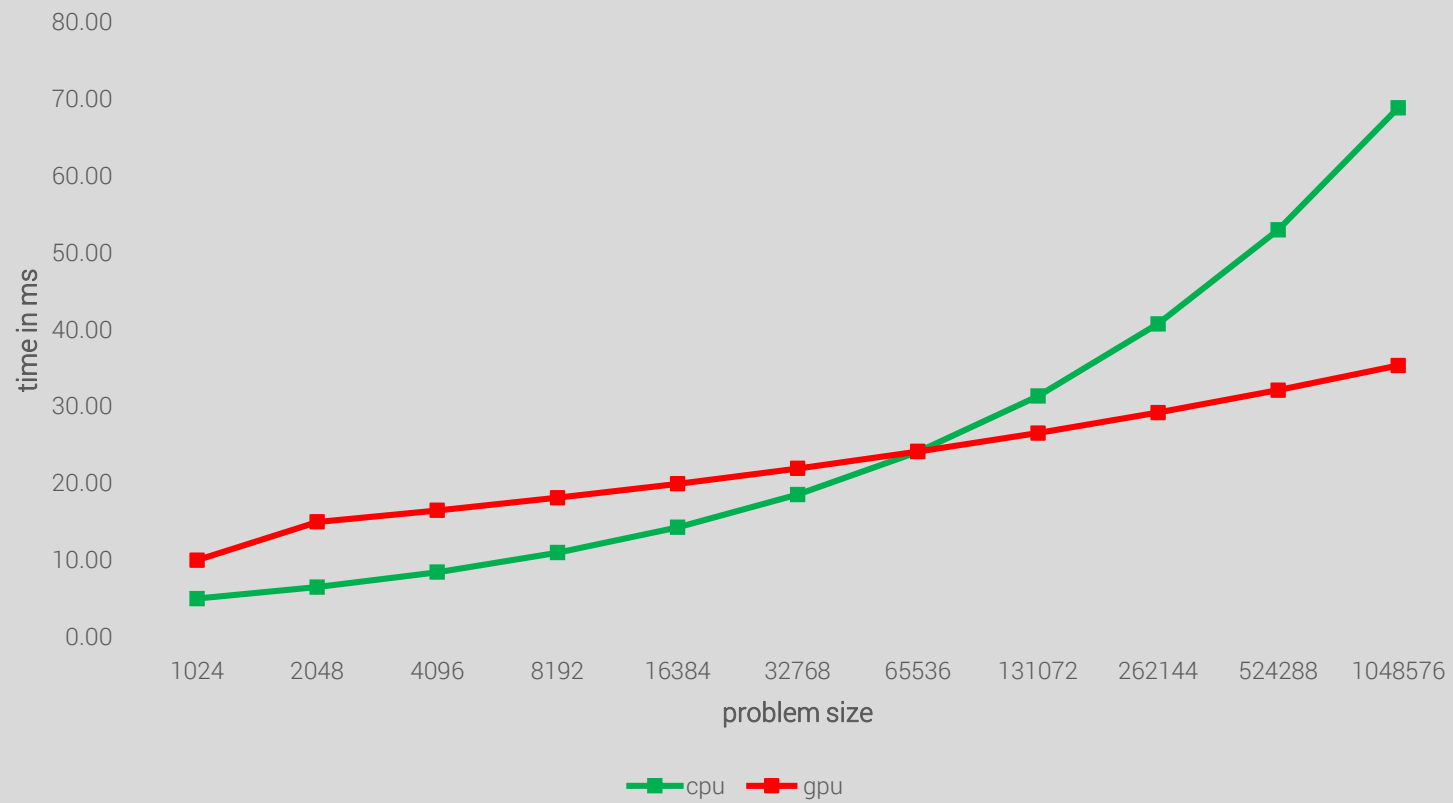
using `std::chrono::steady_clock`

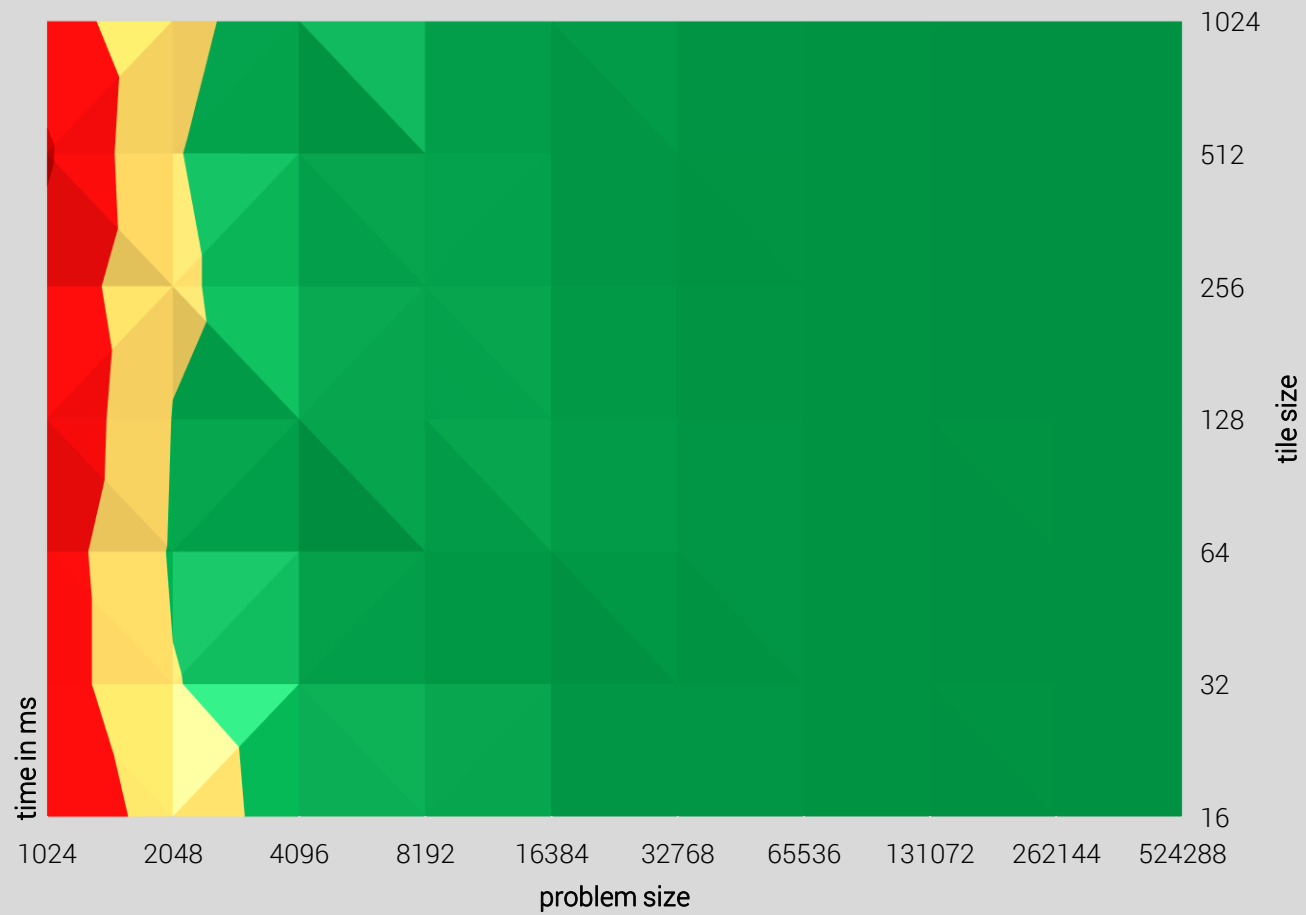
using `tile_size` with multiple of two

`accelerator_view::wait()`



Initialize time   kernel one time   copying time   kernel two time





0-0.005 0.005-0.01 0.01-0.015 0.015-0.02

<https://github.com/KarelZe/MC-VAR-Sim>

Slides shared with  under cc-by 4.0