
# Wenn’s mal wieder länger dauert

## auf der Suche nach dem Bottleneck

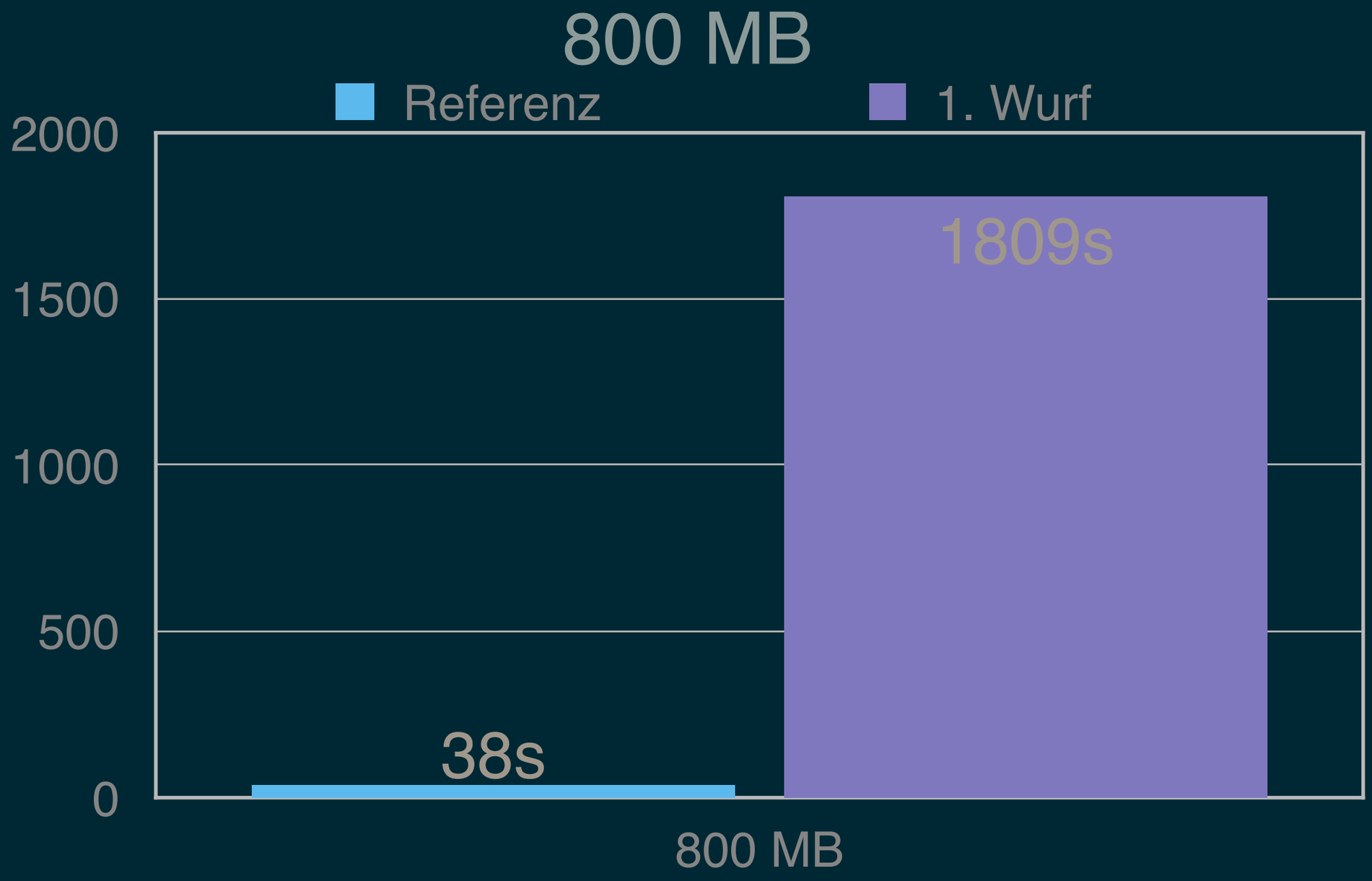
# Applikation

Datei im Streaming Verfahren verschlüsseln

# Erwartungshaltung

Wenn ein optimiertes C Programm 40 Sekunden braucht, wie lange braucht dann ein optimiertes Java Programm?

# Laufzeit/Datenmenge

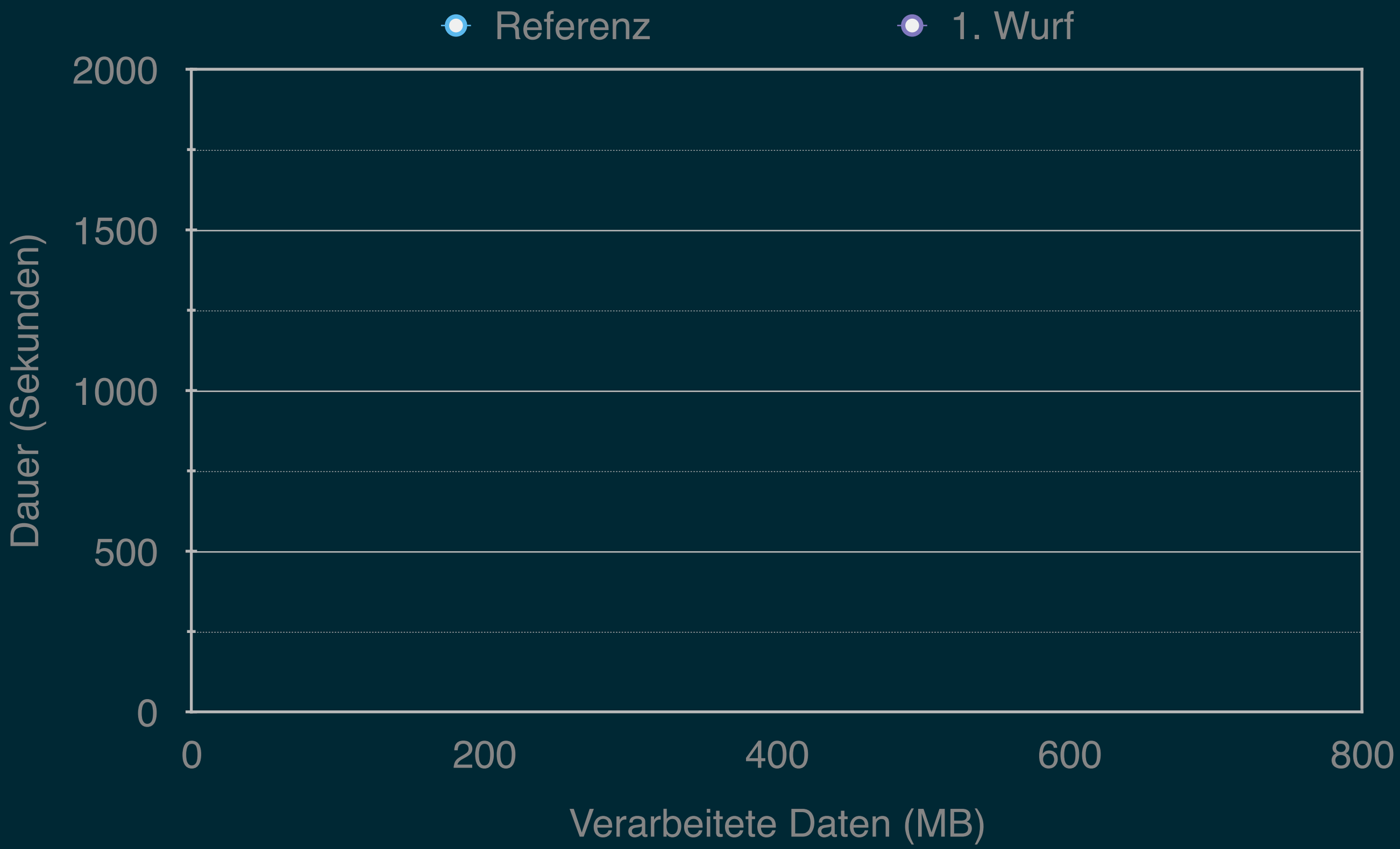



~ Faktor 47 zwischen Referenz und 1. Wurf

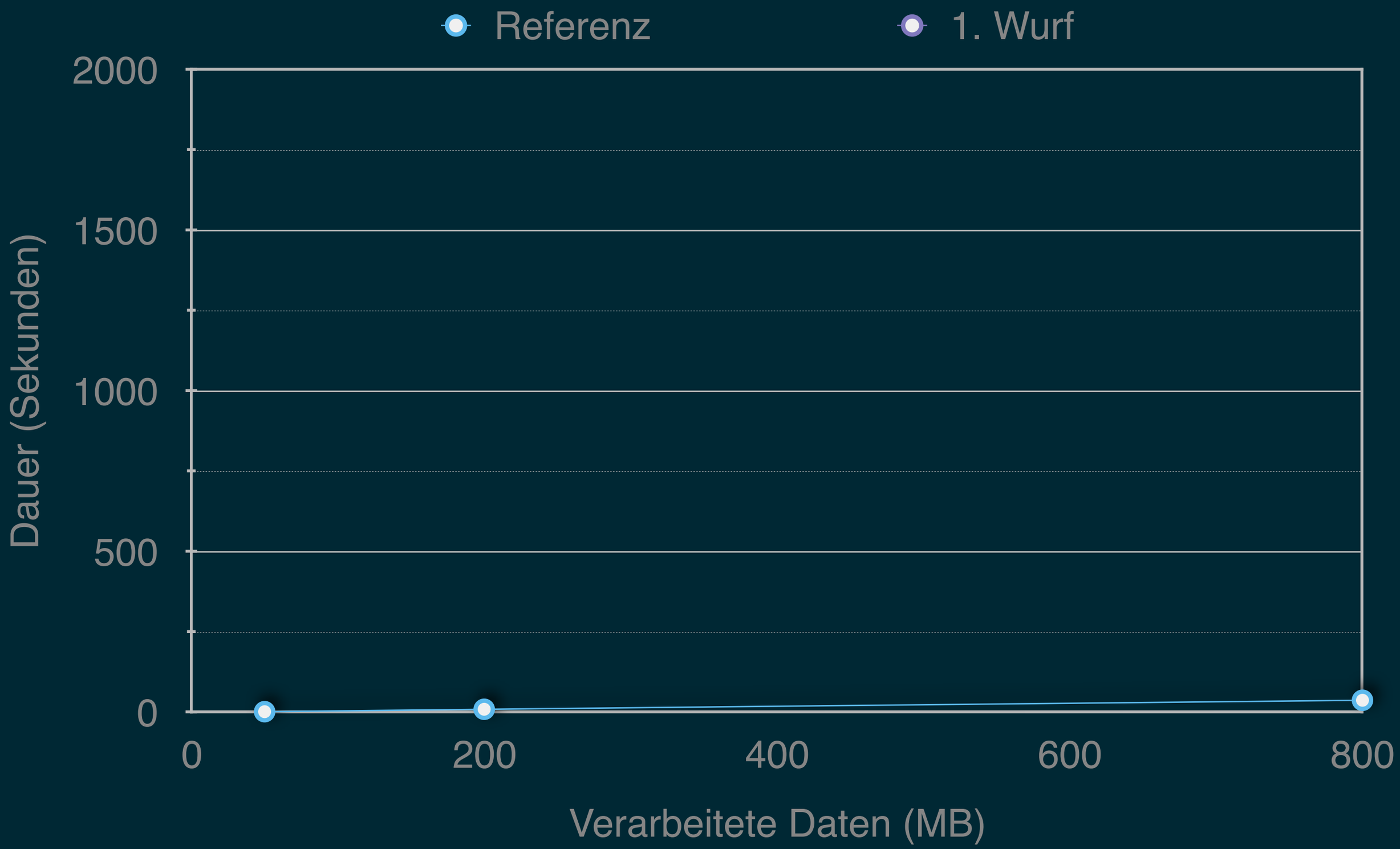
# Happy?

# Laufzeit/Datenmenge

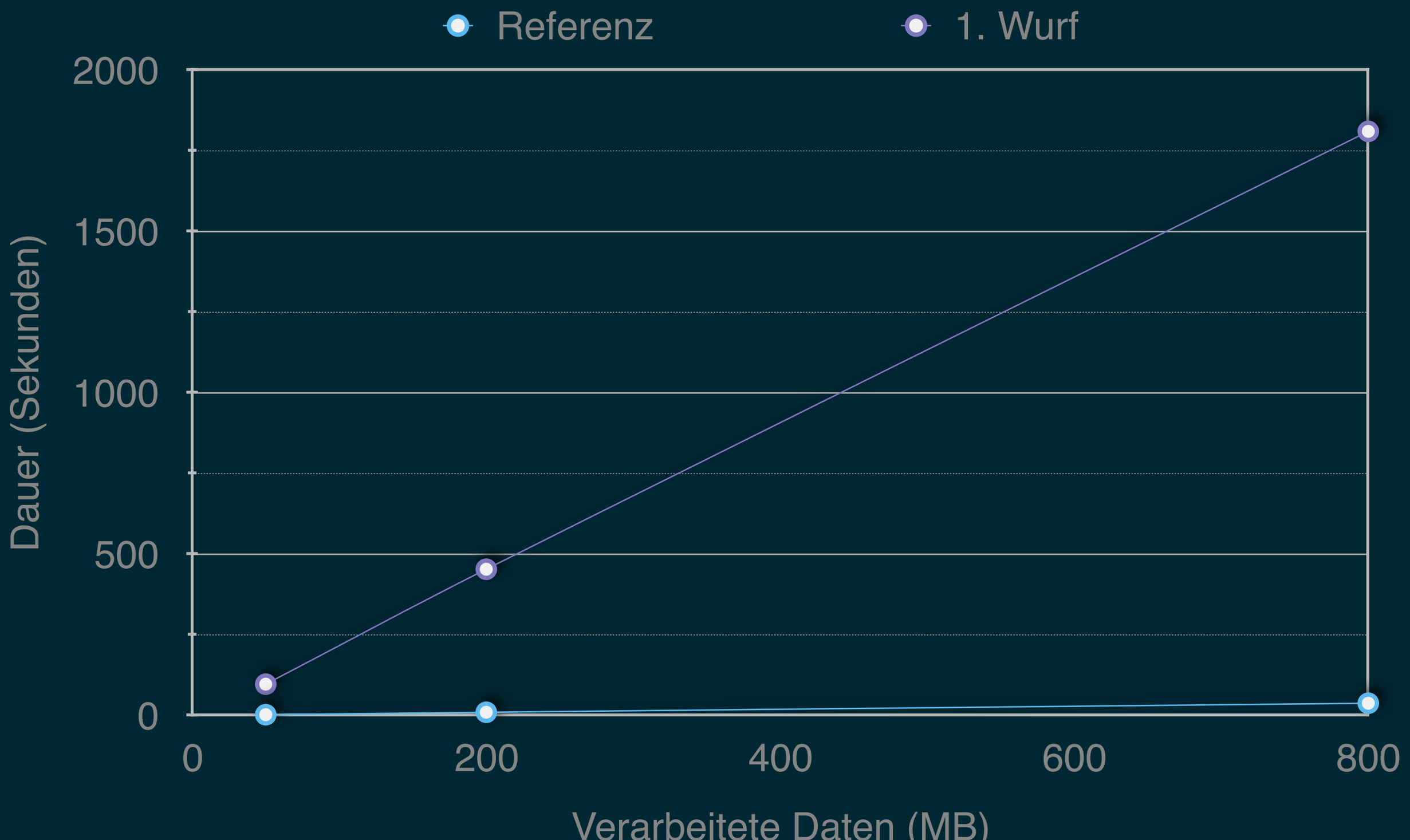
# Laufzeit/Datenmenge



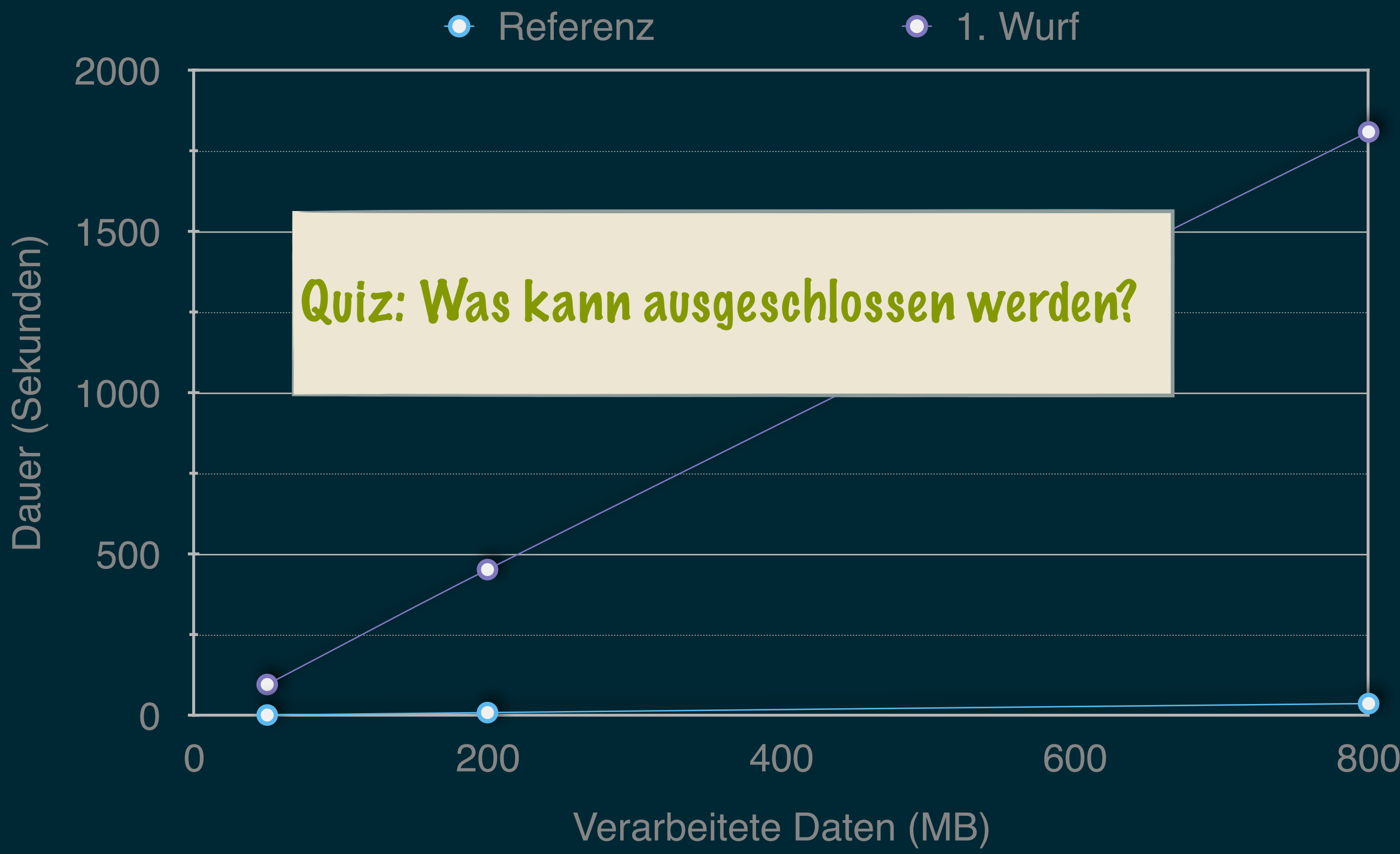
# Laufzeit/Datenmenge



# Laufzeit/Datenmenge



# Laufzeit/Datenmenge




**Quiz: Was kann ausgeschlossen werden?**

# Und jetzt?

# Und jetzt?

# Was tut der Rechner?

* Rechnen (CPU)*
* I/O (Platte, Netz)
* Blockieren (z.B. Mutex)

Bottleneck?

* Nicht Thema:
  Multi-Core, CPU Caches, TLB thrashing, …
  Effiziente Algorithmen

# Was tut der Rechner?

## Begriffe

* *Ressource*: CPU, Disk, Network, …
* ***U**tilisation*: Wie viel Zeit (in %) ist eine Ressource beschäftigt?
* ***S**aturation*: Wie viel Arbeit “bleibt liegen”?

* ***E**rrors*: Wie viele Fehler treten auf?

# Was soll er tun?

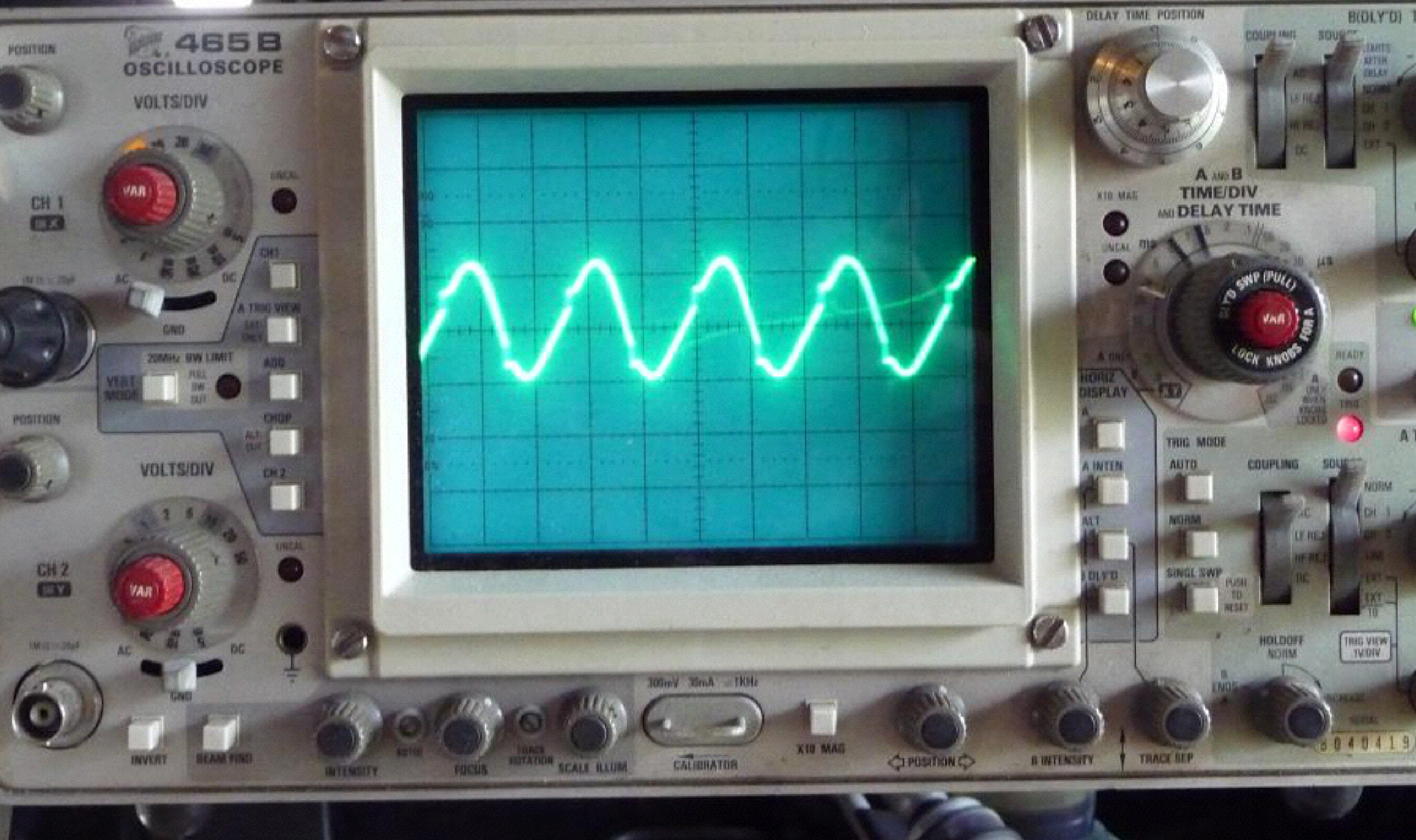
oder: Wo ist der Bottleneck?

* *Für jede Ressource*:
* ***U****tilisation*:  beschäftigt! *
* ***S****aturation*: Es bleibt keine Arbeit liegen!
* ***E****rrors*: Es treten keine Fehler auf!

* Je nach System/Messung: Utilisation sollte ~80% sein
  Durchsatz vs. Latenz: beachten!

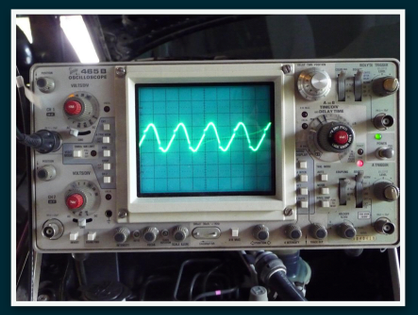
  Die *USE Methode* ist von Brendan Gregg

# Und jetzt? Messen!



# Disk

## messen



* ***U****tilisation:* iostat (busy time, IOPS)
* ***S****aturation*: iostat (wait time, IOPS), vmstat (b)
* ***E****rrors*: iostat

# Disk

```
$ iostat -x -p dm-3 60

rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
0.00     0.00       0.17    6.50    0.67   141.07  42.52     0.05     6.91    4.00    6.98   0.70   0.47
```

```
$ iostat -x -p dm-3 60

[…] r/s     w/s    rkB/s    wkB/s […] svctm %util
[…] 0.17    6.50    0.67   141.07 […]  0.70  0.47
```

# Disk

```
$ iostat -x -p dm-3 60

rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
0.00     0.00       0.17    6.50    0.67   141.07  42.52     0.05     6.91    4.00    6.98   0.70   0.47
```

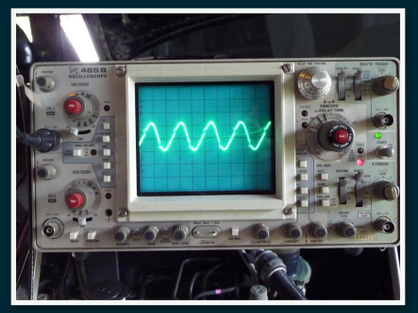
```
$ iostat -x -p dm-3 60

[…] r/s      w/s     rkB/s     wkB/s […] svctm %util
[…] 0.17    6.50     0.67    141.07 […]  0.70  0.47
```

* I/O: kein Problem

# CPU

## messen



* ***U****tilisation:* time, top, vmstat, mpstat
* ***S****aturation*: vmstat run queue > #cores
* ***E****rrors*: z.B. mcelog

# CPU

```
$ time ./encrypt.sh […]
[…]
Encryption took 452.47 s

real    7m32.730s
user    0m16.597s
sys     7m17.231s
```

```
$ mpstat -P ALL 60
CPU     %usr      %nice   %sys   %iowait   […]   %idle
 all     0.68      0.02   12.41     0.15    […]    86.73
   0     0.32      0.02    1.39     0.60    […]    97.68
   1     0.25      0.02    0.13     0.02    […]    99.58
   2     0.82      0.03   15.59     0.03    […]    83.53
   3     0.50      0.00    6.67     0.08    […]    92.74
   4     1.69      0.02   43.23     0.03    […]    55.04
   5     1.05      0.00   26.95     0.15    […]    71.83
   6     0.52      0.07    5.13     0.32    […]    93.97
   7     0.30      0.02    0.18     0.00    […]    99.50
```

# CPU

```
$ time ./encrypt.sh […]
[…]
Encryption took 452.47 s

real    7m32.730s
user    0m16.597s
sys     7m17.231s
```

```
$ mpstat -P ALL 60
CPU     %usr      %nice   %sys   %iowait   […]   %idle
 all     0.68      0.02   12.41     0.15   […]    86.73
   0     0.32      0.02    1.39     0.60   […]    97.68
   1     0.25      0.02    0.13     0.02   […]    99.58
   2     0.82      0.03   15.59     0.03   […]    83.53
   3     0.50      0.00    6.67     0.08   […]    92.74
   4     1.69      0.02   43.23     0.03   […]    55.04
   5     1.05      0.00   26.95     0.15   […]    71.83
   6     0.52      0.07    5.13     0.32   […]    93.97
   7     0.30      0.02    0.18     0.00   […]    99.50
```

# CPU

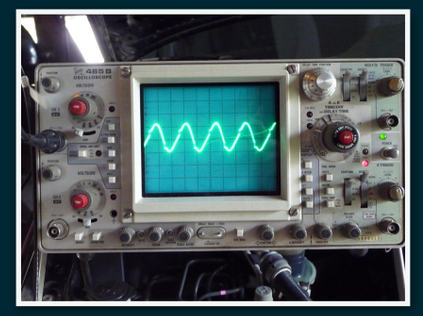
```
$ time ./encrypt.sh […]
[…]
Encryption took 452.47 s

real    7m32.730s
user    0m16.597s
sys     7m17.231s
```

```
$ mpstat -P ALL 60
CPU     %usr      %nice   %sys    %iowait   […]   %idle
 all     0.68     0.02    12.41     0.15    […]    86.73
   0     0.32     0.02     1.39     0.60    […]    97.68
   1     0.25     0.02     0.13     0.02    […]    99.58
   2
   3
   4
   5
   6     0.52     0.07     5.13     0.32    […]    93.97
   7     0.30     0.02     0.18     0.00    […]    99.50
```

* I/O: kein Problem
* Single Threaded (12.5% von 8 Cores = 1 Core)
* 96% Kernel Time?

# sys == Syscall



## Messen z.B. mit systemtap *

```
#! /usr/bin/env stap
#
# This script lists the top 20 systemcalls in total
# start with stap -c command / stap -x pid

global syscalls_count

probe syscall.* {
  if (target() == pid())  {
     syscalls_count[name] <<< 1
  }
}

function print_systop () {
  printf ("%25s %10s\n", "SYSCALL", "COUNT")
  foreach (syscall in syscalls_count- limit 20) {
    printf("%25s %10d\n", syscall, @count(syscalls_count[syscall]))
  }
}

probe end {
  print_systop ()
}
```

* https://sourceware.org/systemtap/

# Syscalls?

# ~125,000 write calls/s

bei einem Durchsatz von 141 kB/s

```
$ iostat -x -p dm-3 60

[…] r/s     w/s    rkB/s    wkB/s […]  %util
[…] 0.17    6.50    0.67   141.07 […]  0.47
```

# Syscalls?

# ~125,000 write calls/s

## bei einem Durchsatz von 141 kB/s

```
$ iostat -x -p dm-3 60

[…] r/s     w/s     rkB/s     wkB/s […]  %util
[…] 0.17    6.50    0.67    141.07 […]  0.47
```

* I/O: kein Problem
* 96% Kernel Time?
  * 125k system calls für 141 kB!


DINOCAR

# write calls?

```
final OutputStream outputStream = new FileOutputStream(destFile);
pgp.encryptAndSign(new FileInputStream(sourceFile), outputStream);
```

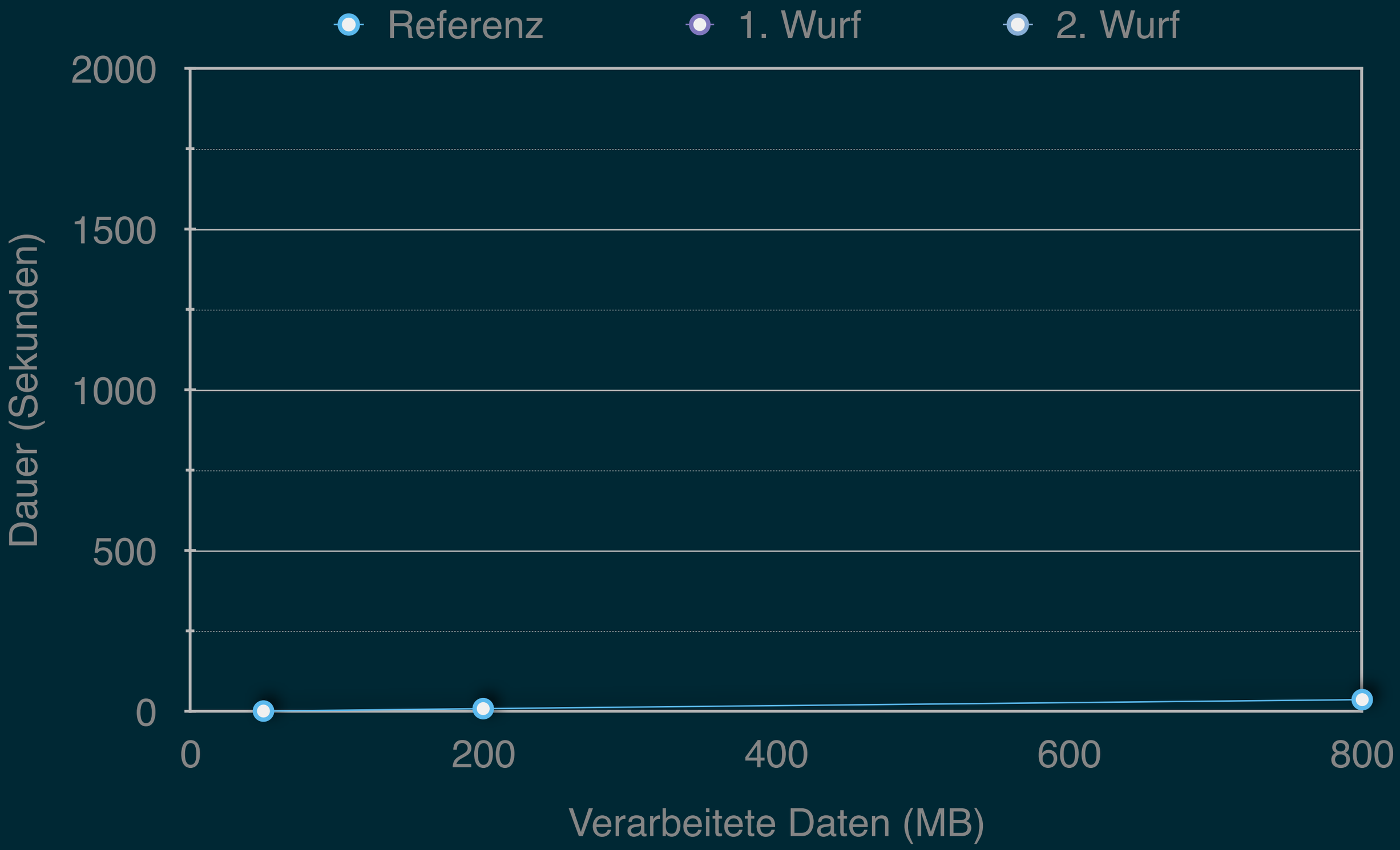
```
final OutputStream outputStream = new FileOutputStream(destFile);
final OutputStream bufferedOutputStream = new BufferedOutputStream(outputStream, 8192);

pgp.encryptAndSign(new FileInputStream(sourceFile), bufferedOutputStream);
```

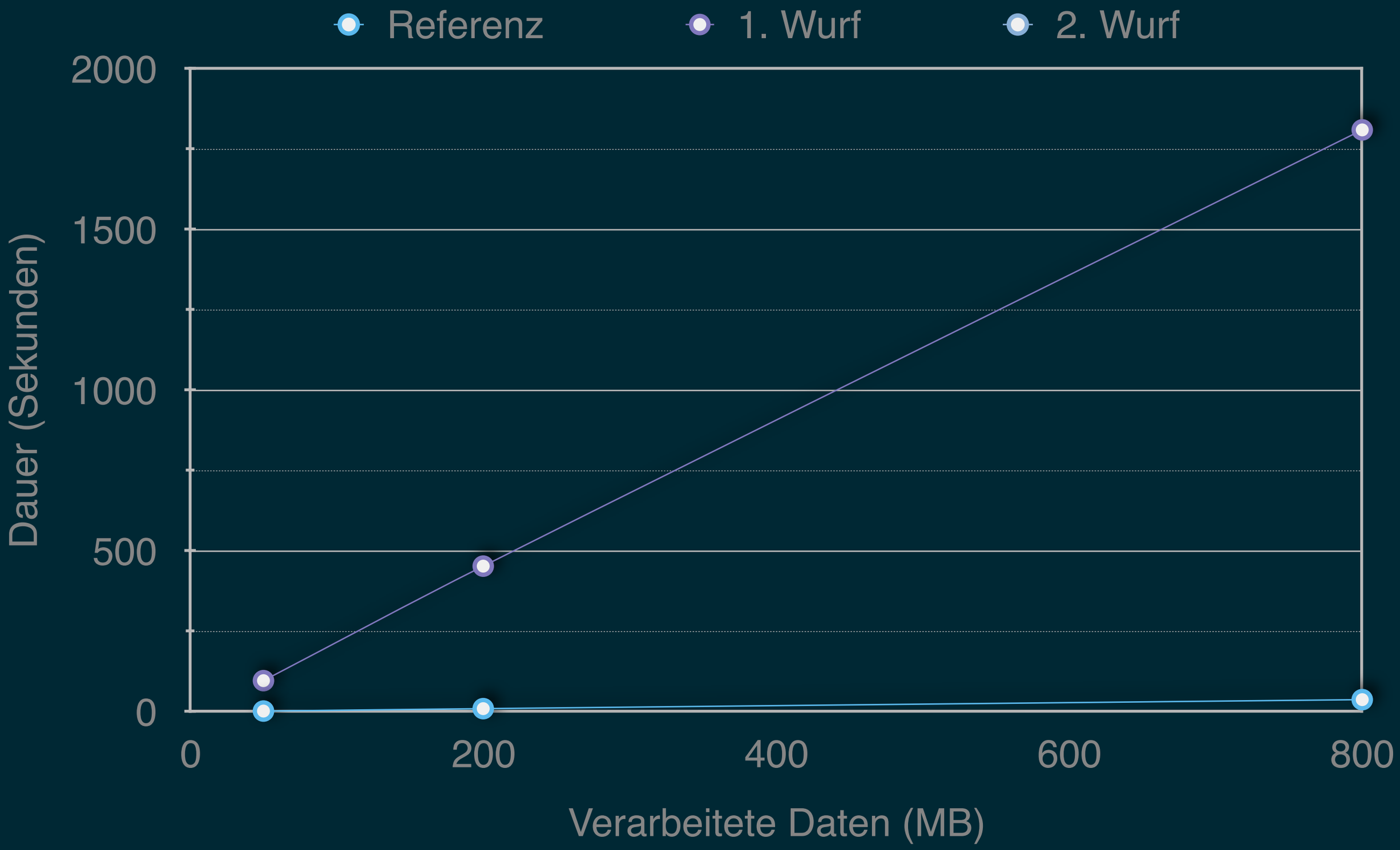
# Lastverhalten

# Lastverhalten

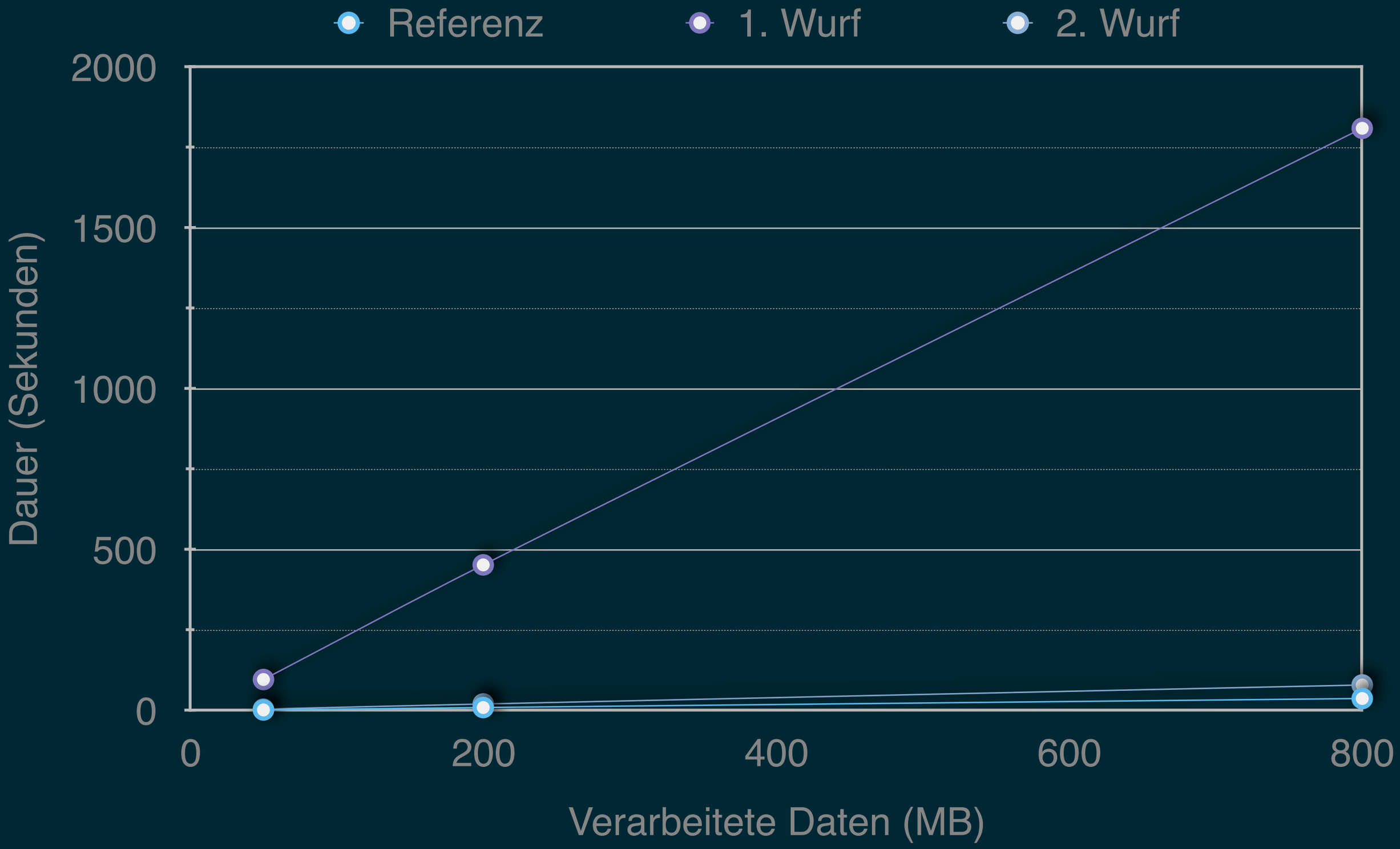
# Lastverhalten



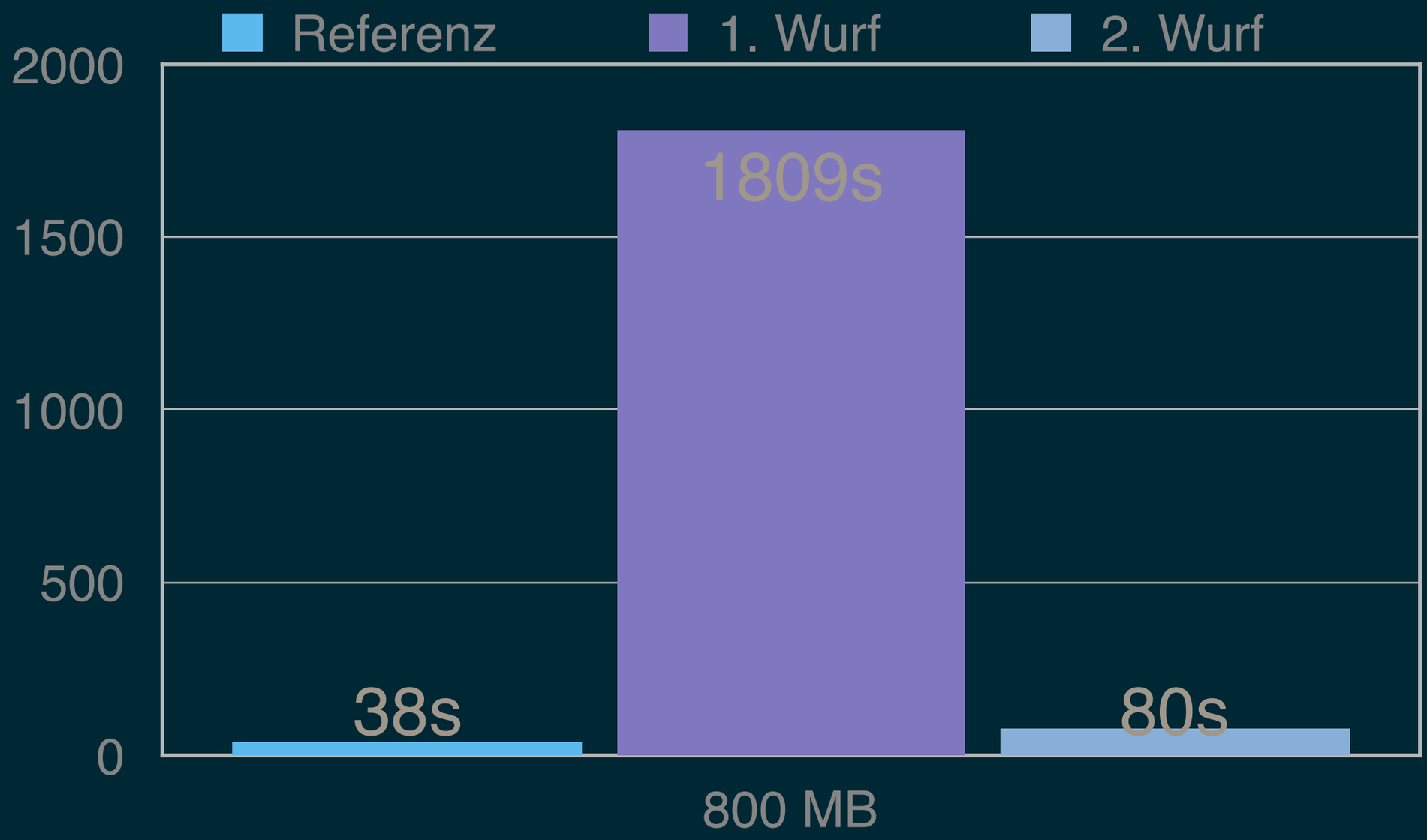
# Lastverhalten



# Lastverhalten



# Lastverhalten




~ Faktor 47 zwischen Referenz und 1. Wurf
~ Faktor 2 zwischen Referenz und 2. Wurf

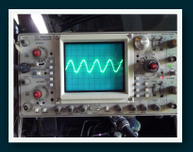
https://images.unsplash.com/40/x6YzbWWRq2sRhAacMjnl_Bangkok%20Indra%20market.jpg?ixlib=rb-0.3.5&q=80&fm=jpg&s=a3d1f05b8862f07db7a569c1bf1ecbc1 (Creative Commons Zero)

https://images.unsplash.com/photo-1433616174899-f847df236857?ixlib=rb-0.3.5&q=80&fm=jpg&s=c1e8fe89e406303598e6d449f9b685eb (Creative Commons Zero)

https://en.wikipedia.org/wiki/Hard_disk_drive#/media/File:Hdd_icon.svg (LGPL)

https://www.dropbox.com/s/dyf59rjfhpo9jaw/78H.jpg?dl=1 (Creative Commons Zero)

https://commons.wikimedia.org/wiki/File:Singapore_Road_Signs_-_Warning_Sign_-_Road_Narrows_On_Right.png (CC-BY 3)



https://upload.wikimedia.org/wikipedia/commons/3/39/RAV4_Tach_750rpm.JPG (public domain)

http://www.gratisography.com/pictures/77_1.jpg (Creative Commons Zero)