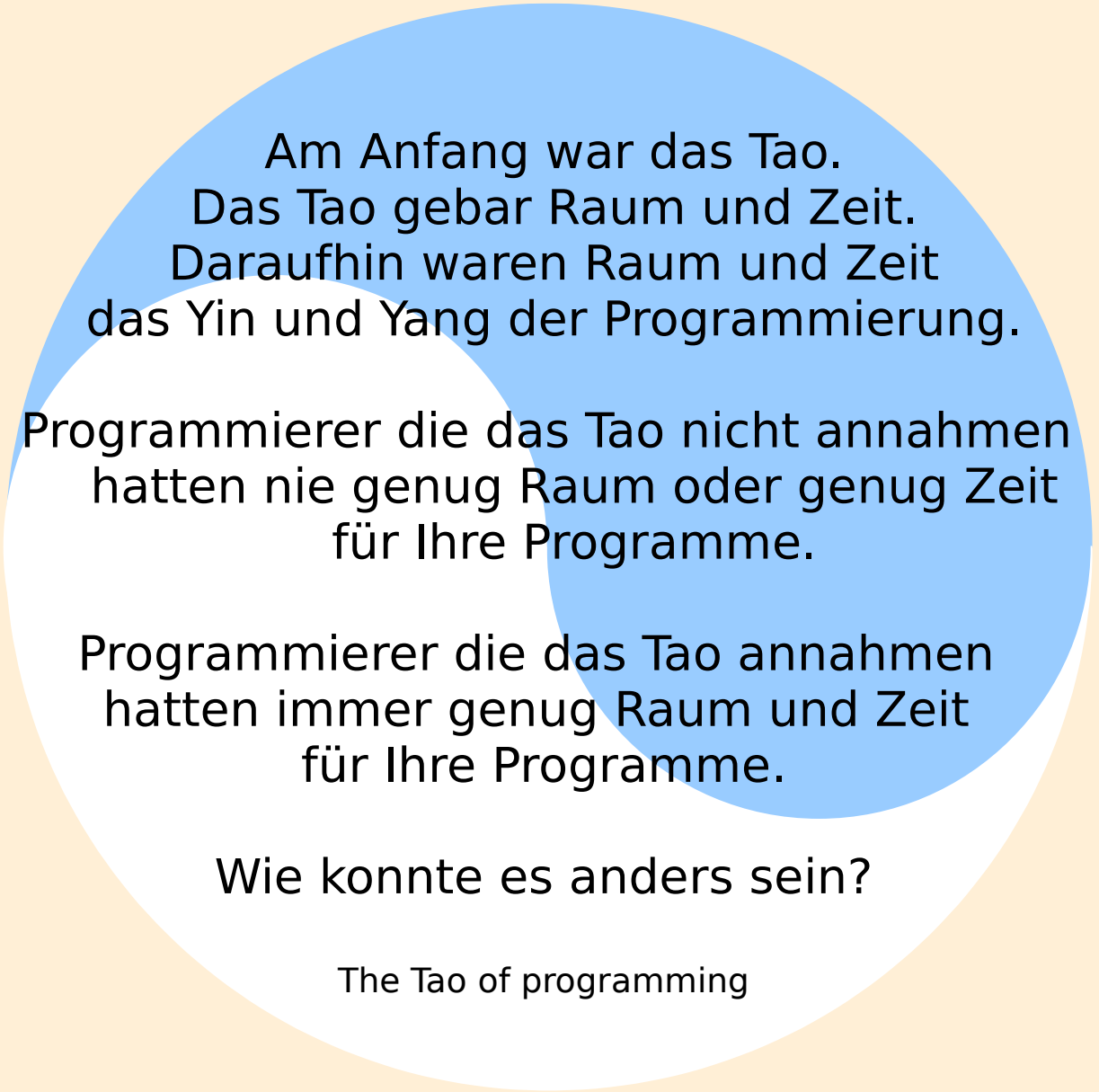

Am Anfang war das Tao.
Das Tao gebar Raum und Zeit.
Daraufhin waren Raum und Zeit
das Yin und Yang der Programmierung.
Programmierer die das Tao nicht annahmen
hatten nie genug Raum oder genug Zeit
für Ihre Programme.
Programmierer die das Tao annahmen
hatten immer genug Raum und Zeit
für Ihre Programme.
Wie konnte es anders sein?
The Tao of programming

Wie schnell ist schnell?

Dr. Volker Jaenisch

inqbus
it consulting

http://inqbus.de
Fragen zum Vortrag bitte an
volker.jaenisch@inqbus.de

- **Linux Performance Analyse**
- **IO-Subsystem: Fehler finden, IO-Sheduler, Hardware vs. Software RAID**
- **Kernel: Load vs. CPU%, Prozess Scheduler**
- **Netzwerk: TCP Performance**

Einleitung

Server 1980 :

- **CPU: 8-Bit : 4.77 MHZ 8068**
- **ST-506 HDD**
- **System-BUS-Rate 4.77 MHZ**

Server 2008 :

- **CPU: 64Bit: 3 GHZ Quadcore AMD 2212**
- **ST Barracuda ES**
- **System-BUS-Rate 133 MHZ**

Fortschritt verschärft Probleme

Die Parameter der Hardware wachsen nicht homogen

Faktorielle Veränderung bezogen auf 1980

HDD	**Kapazität**	**Interfaces**	**Rotation**	**Zugriffszeit**
	6000	**1600**	**5**	**1/10**
Taktrate	**CPU**	**DRAM**	**Busse**	
	1000	**40**	**30**	
Netzwerk:	**Takt**	**Paketgröße**		
	100000	**1**		

HDD

ST-225 ST412 MFM

UNFORMATTED CAPACITY (MB)	25.6
FORMATTED CAPACITY (17 SECTORS) (MB)	21.4
ACTUATOR TYPE	STEPPER
TRACKS	2,460
CYLINDERS	615
HEADS	4
DISCS	2
MEDIA TYPE	OXIDE
RECORDING METHOD	MFM
TRANSFER RATE (mbits/sec)	5.0
SPINDLE SPEED (RPM)	3,600
AVERAGE LATENCY (mSEC)	8.3
INTERFACE	ST412
SECTORS PER DRIVE	41,820
TPI (TRACKS PER INCH)	588
BPI (BITS PER INCH)	9,827
AVERAGE ACCESS (ms)	**65**
SINGLE TRACK SEEK (ms)	20
MAX FULL SEEK (ms)	150
MTBF (power-on hours)	100,000
POWER REQUIREMENTS: +12V START-UP (amps)	2.4
+12V TYPICAL (amps)	0.9
+5V TYPICAL (amps)	0.8
TYPICAL (watts)	14.8
MAXIMUM (watts)	33
BUFFERED STEP PULSE RATE (micro sec)	5-200
WRITE PRECOMP (cyl)	300

Intelligente Tricks

Thus spoke the master programmer:
Without the wind, the grass does not move.
Without software, hardware is useless. (TAOP)

Damit es trotzdem funktioniert wird schon seit alters her mit Tricks gearbeitet:

- RAM, HDDs: Caching, Interleaving, Scheduling
- Kernel/CPU: Caching, Multi-Core, Parallelisierung
- Code: Optimierung, Parallelisierung, Branch-Prediction, Lokalisation

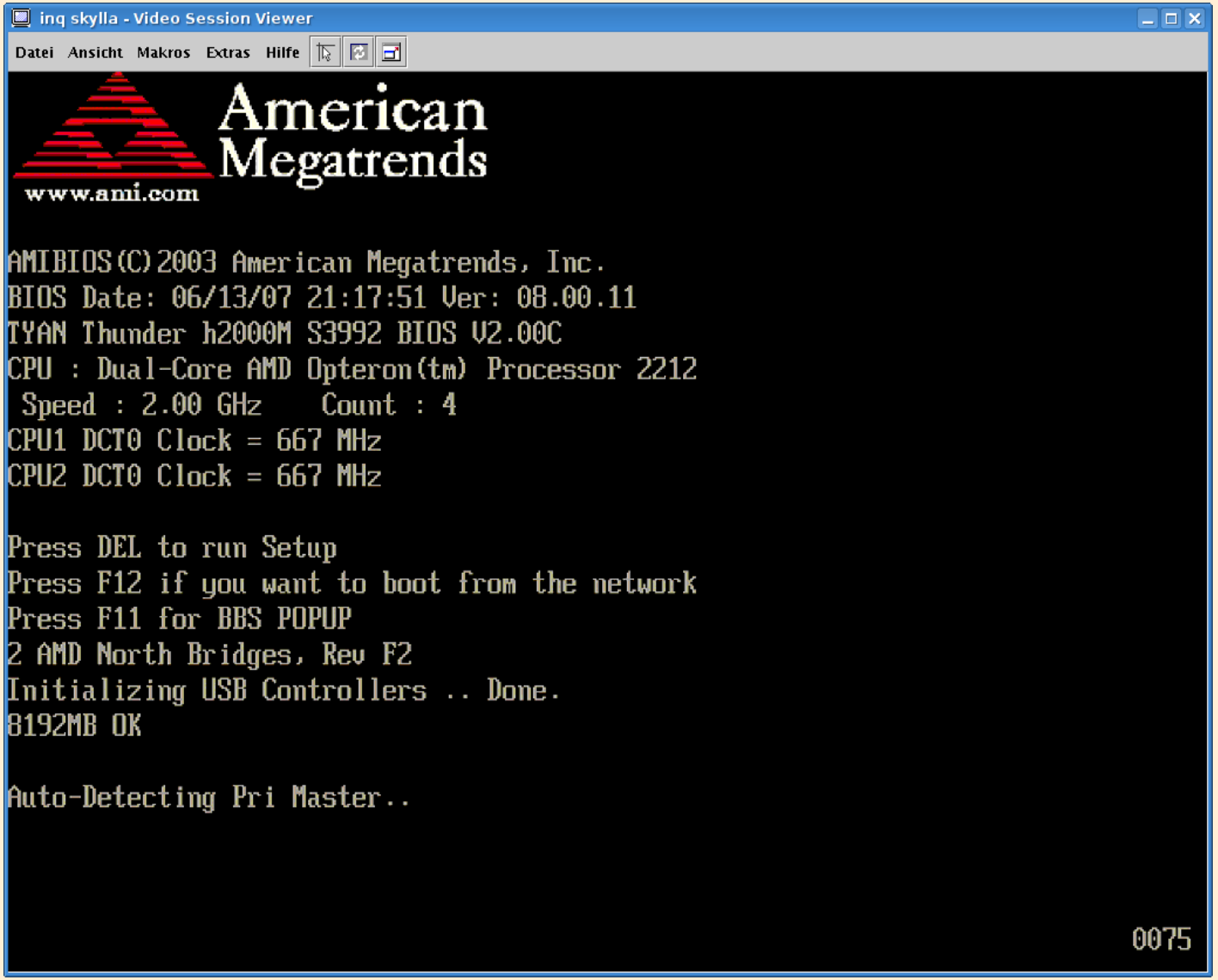
Optimierungs-Potential in Faktoren

- Design 1000
- DatenStrukturen 100
- Algorithmen 100
- Schnittstellen 100
- Coding style 10
- Network 10
- Storage 5
- Hardware 1

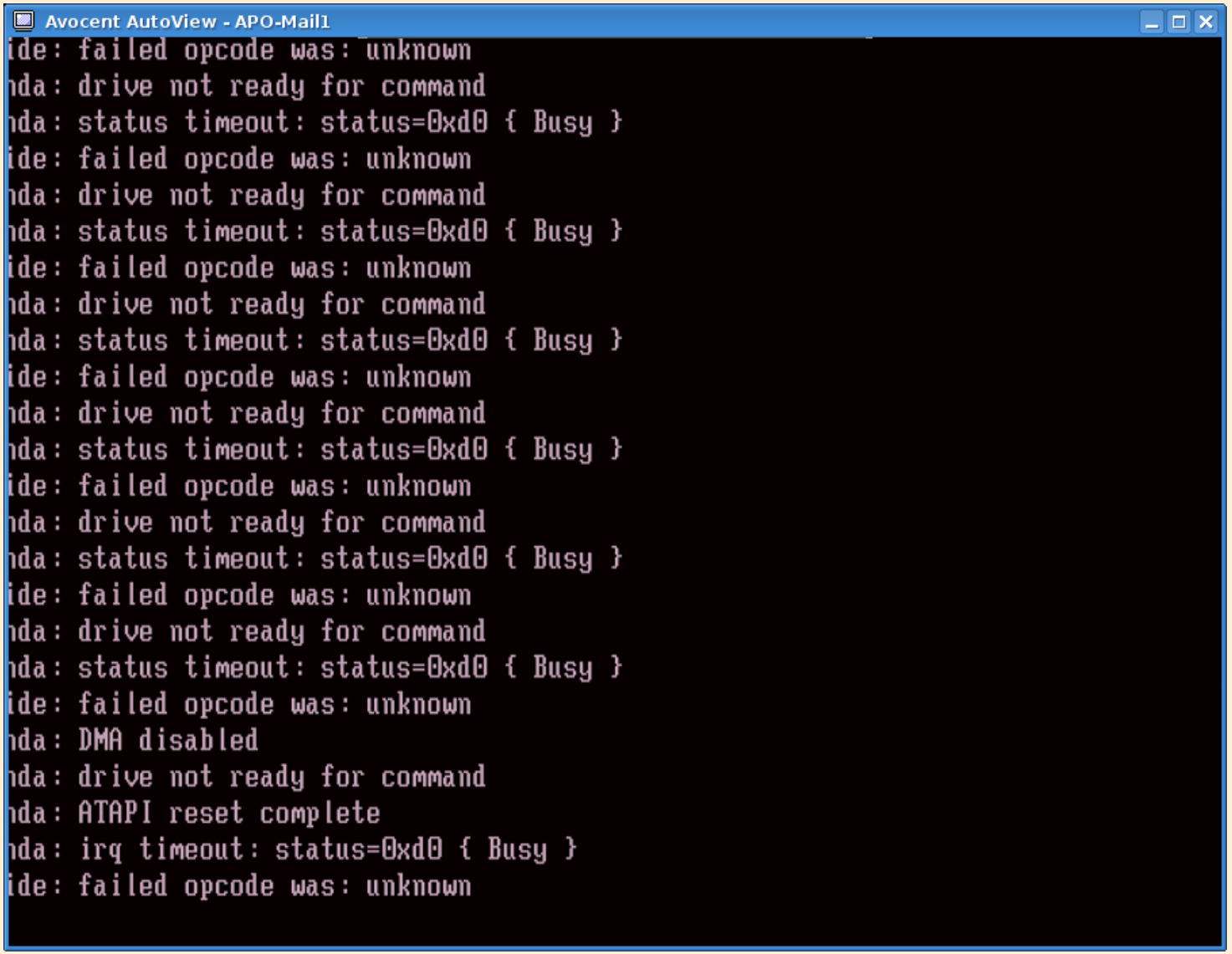
Hardware
Bekam ich was ich mietete/kaufte?

- *lspci* : Test auf Komponenten/Board
- *cat /proc/cpuinfo* : Was für/Wieviel CPU(s)
- *cat /proc/meminfo* : Wieviel RAM
- *decode-dimms.pl* (sensors) : Was für RAMs
- *memtest* (knoppix et.al) : Arbeiten RAMs
- *dmesg* : HDD-Typ
- Über remote-Console: BIOS Meldungen/Einstellungen, Controller BIOS, Boot-Verhalten
- Jede noch so kleine Unstimmigkeit ausräumen!

Server hängt manchmal beim Softreset



Nach Boardtausch (anderer Hersteller)



Ursache: Kabel-Origami

Marken RAMs ohne SPD-Vendor Eintrag?

```
mail2:~# /usr/share/doc/lm-sensors/ \
examples/eeprom/decode-dimms.pl
...
Maximum Access Time (CAS 4)       0.5 ns
Minimum Cycle Time (CAS 3)          5 ns
Maximum Access Time (CAS 3)         0.6 ns
...
---=== Manufacturing Information ===---
Manufacturer                      Undefined
PartNumber                        Undefined
ManufacturingDate                 2033-W07
```

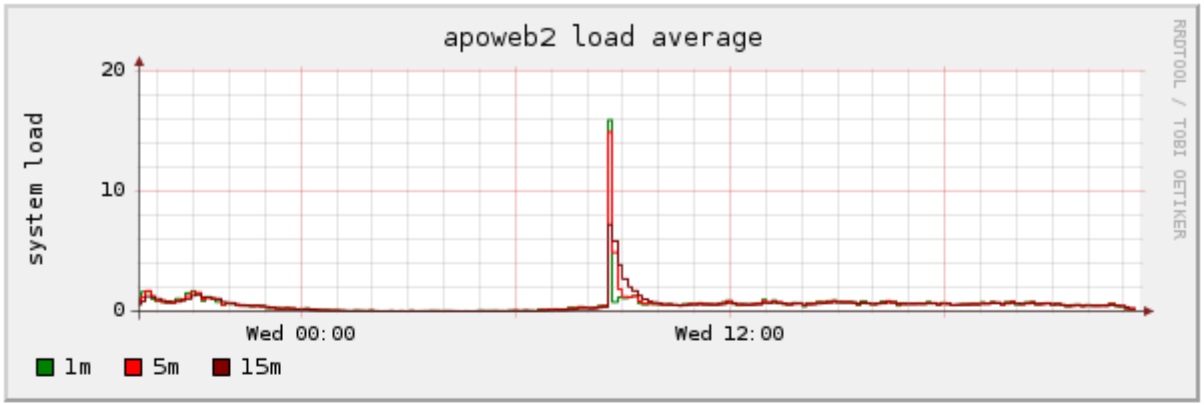
Basis Werkzeuge

- Screen-Shots
- Gnumeric (oder anderes Spreadsheet)
- Scripte für Automatisierung
- Einfache Performance-Tools: hdparm, tiotest
- cat /proc/stat
- cat /proc/loadavg
- cat /proc/meminfo
- top

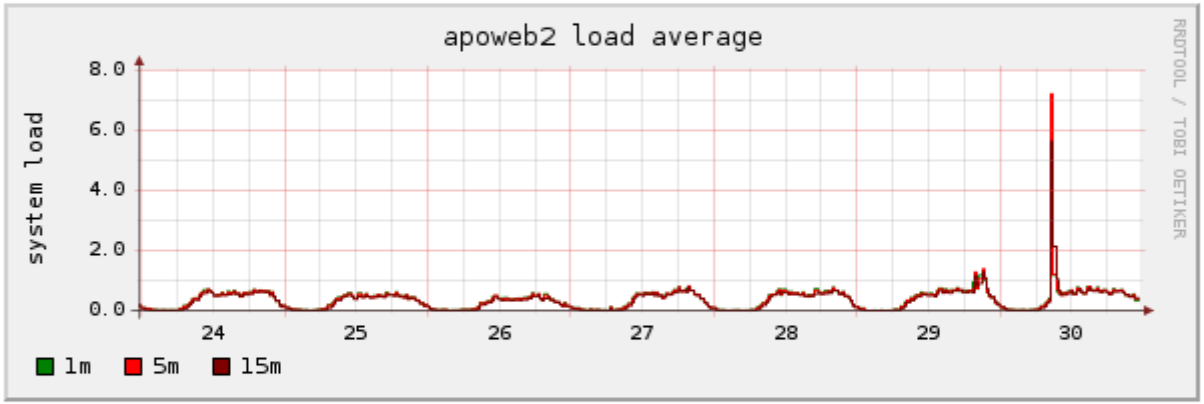
Weitergehende Werkzeuge

- CollectD (Erfasst Performance Systemweit in RRD)
- DRRAW (Simples Web-Frontend zu RRD-Daten)
- Diverse Performance-Suites.

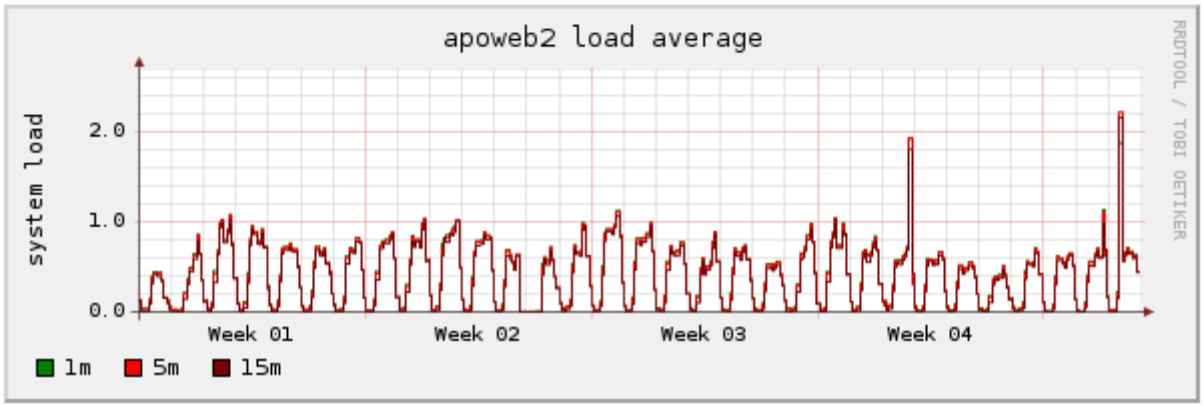
CollectD und DRRAW



Past 28 Hours



Past Week



Past Month

HDD Performance

Mit einfachen Mitteln HDD Performance Probleme finden

1) Suchen eines Performance-Analyse Tools
2) Anwendung nach Gebrauchsanweisung
3) Nachdenken über das Ergebnis
4) Analyse

Die getestete Festplatte WD RE 250YS

Herstellerangaben:

- Umdrehungsgeschwindigkeit 7.200 U/min (Nennwert)
- Cache-Größe 16 MB
- Zugriffszeit 8,7 ms
- Spur-zu-Spur-Suchzeit 0,6 ms (durchschnittlich)
- Full Stroke Seek 21,0 ms (durchschnittlich)
- Cache-zu-Host (Serial ATA) 3 Gb/s (maximal)
- Datenübertragungsrate (aus dem Puffer auf die Platte) 70 MB/s (kontinuierlich)

Suche Performance-Analyse Tool

google: linux festplatte geschwindigkeit

Erster Treffer:
Wer-weis-was: UNIX und Linux -
Festplattengeschwindigkeit testen :
...
Löst zwar nicht dein Problem mit
Schreib/Lesegeschwindigkeit, zeigt dafür aber schön den Einfluß des Caches und **ist ein gutes Maß für die Geschwindigkeit der Festplatte**:

hdparm -tT /dev/hda

Gruß Stefan

Nutze Gebrauchsanweisung

man hdparm

-T Perform timings of cache reads for benchmark and comparison purposes. For meaningful results, this operation should be repeated 2-3 times on an otherwise inactive system (no other active processes) with at least a couple of megabytes of free memory. This displays the speed of reading directly from the Linux buffer cache without disk access. This measurement is essentially an indication of the throughput of the processor, cache, and memory of the system under test. If the -t flag is also specified, then a correction factor based on the outcome of -T will be incorporated into the result reported for the -t operation.

Nachdenken über das Ergebnis

Hdparm -tT [Mb/sec]

Server A	Server B
96,25	106,15
91,99	57,73
78,24	104,38

Abstimmen!

Wer ist für Server A?

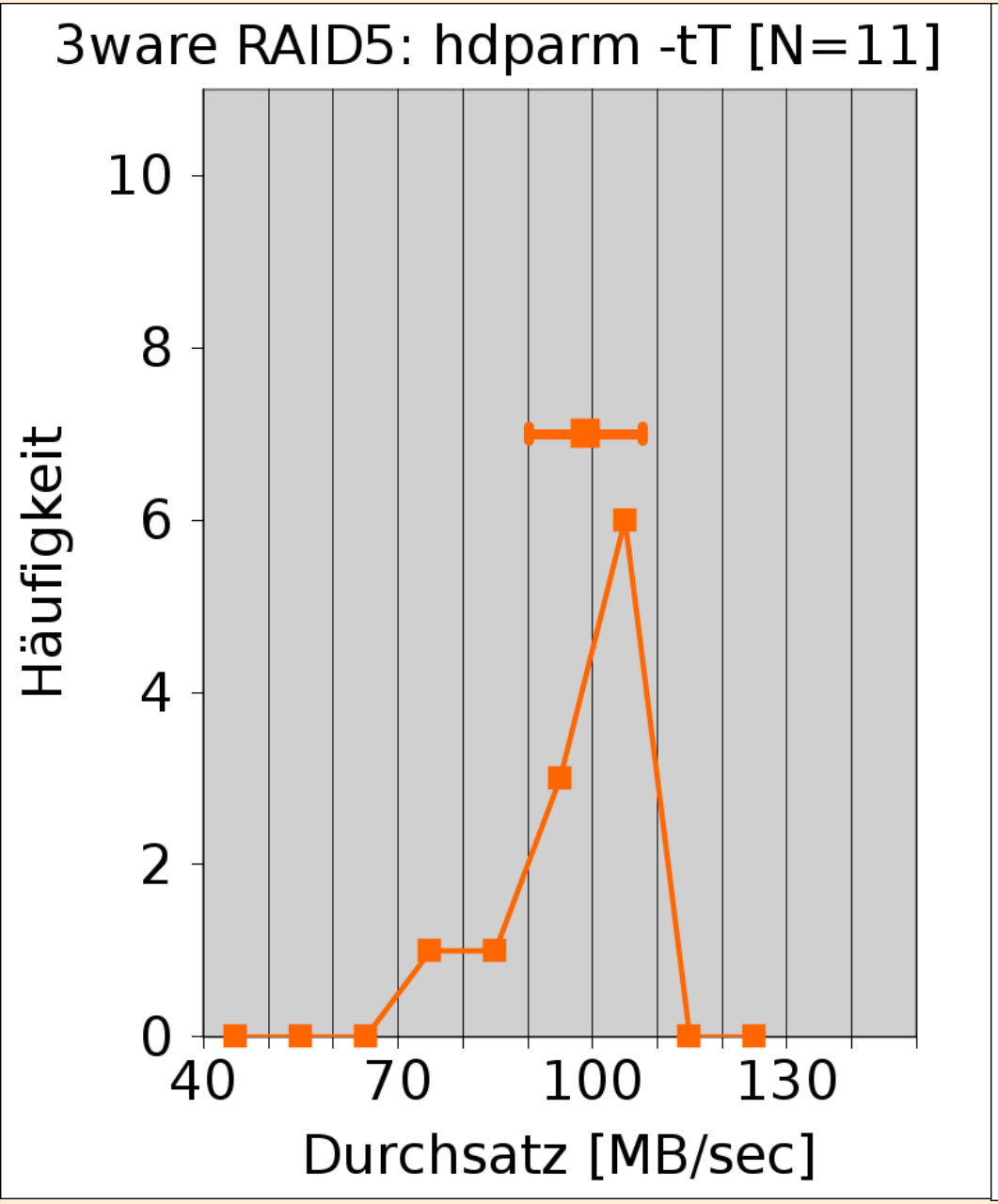
Wer ist für Server B?

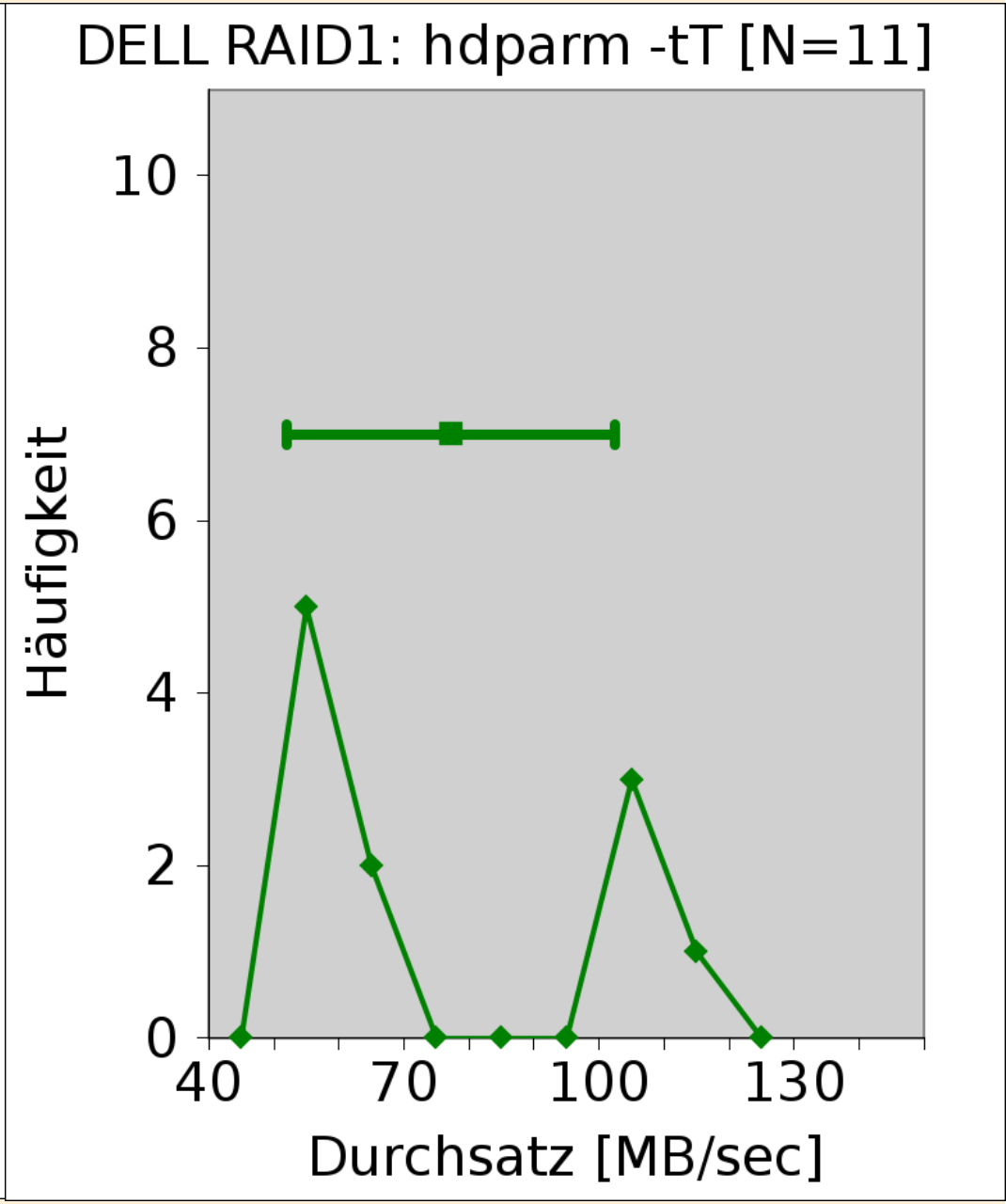
Analyse

Hdparm -tT [MB/sec]

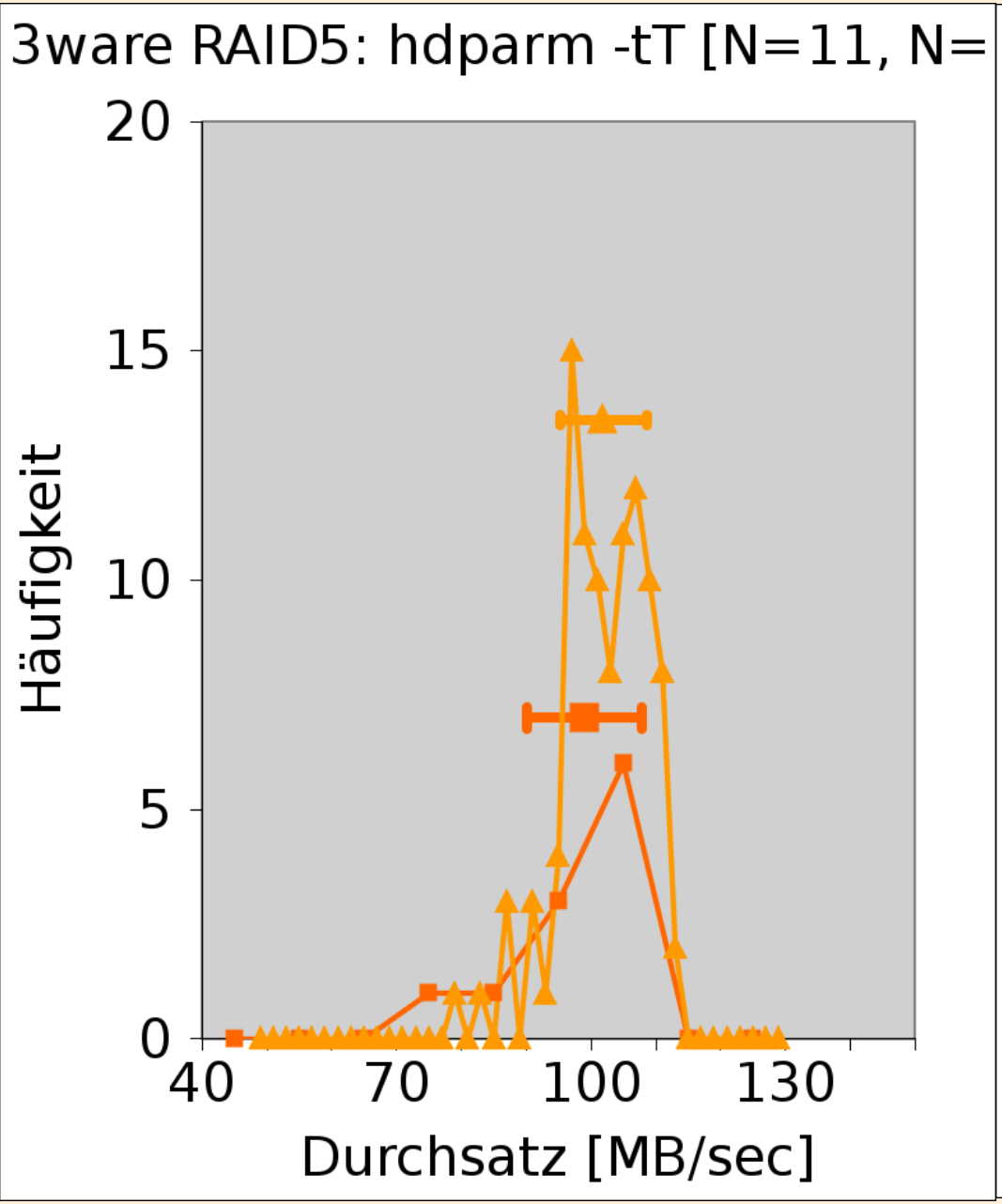
Server A	Server B
105,75	108,43
104,27	55,64
96,71	62,37
99,57	107,52
79,96	58,87
100,37	61,35
85,73	111,10
99,71	57,87
101,37	58,17
106,89	109,02
108,59	59,19
---	---
98,99	77,23
+/-	+/-
8,83	25,27

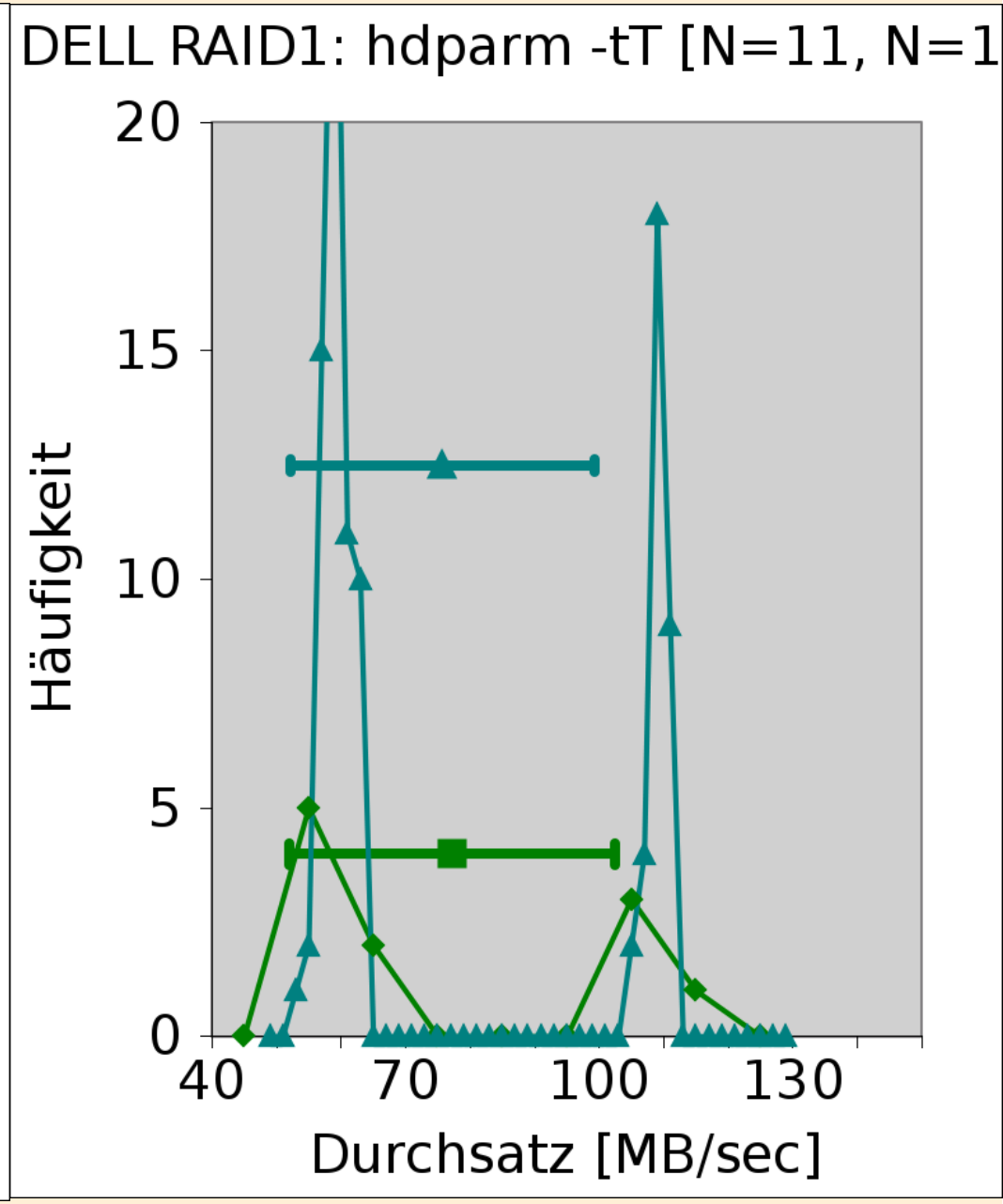
Histogram von 11 Messungen





Histogram von 100 Messungen





Neues Werkzeug TIOtest

- IO-Benchmarks unter simulierter Last mehrerer simultaner Threads
- Observiert dabei auch das System
- To get usefull results the used file sizes should be a lot larger than the physical amount of memory you have. A good idea is to boot with 16 Megs of RAM (Try passing the "mem=16M" option to the kernel to limit Linux to using a very small amount of memory) and into Single User mode only.

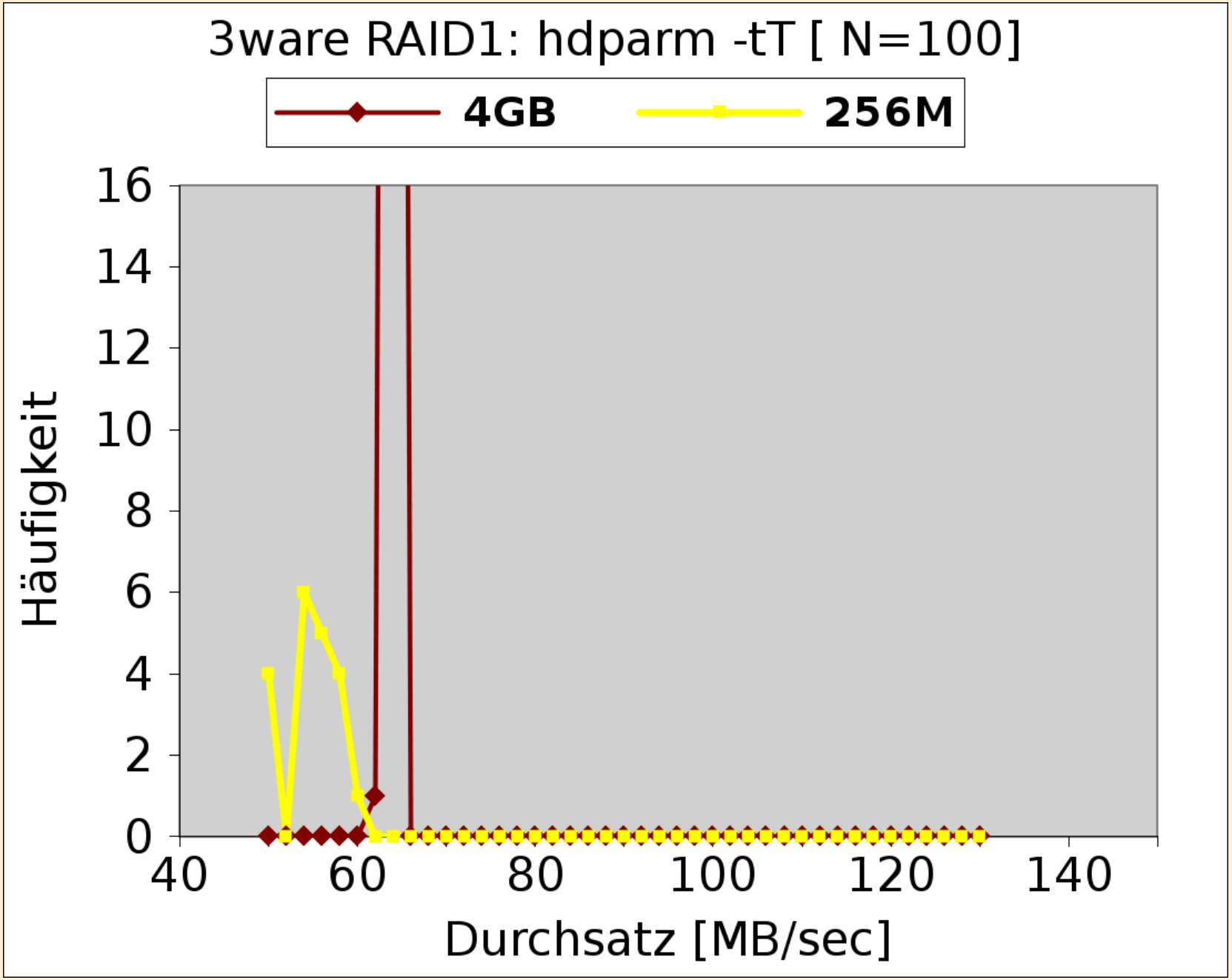
Moment mal!

- hdparm sagte :

  Viel Speicher sonst geht es schief
- tiobench sagt:

  Wenig Speicher sonst geht es schief

Speicherabhängigkeit: hdparm

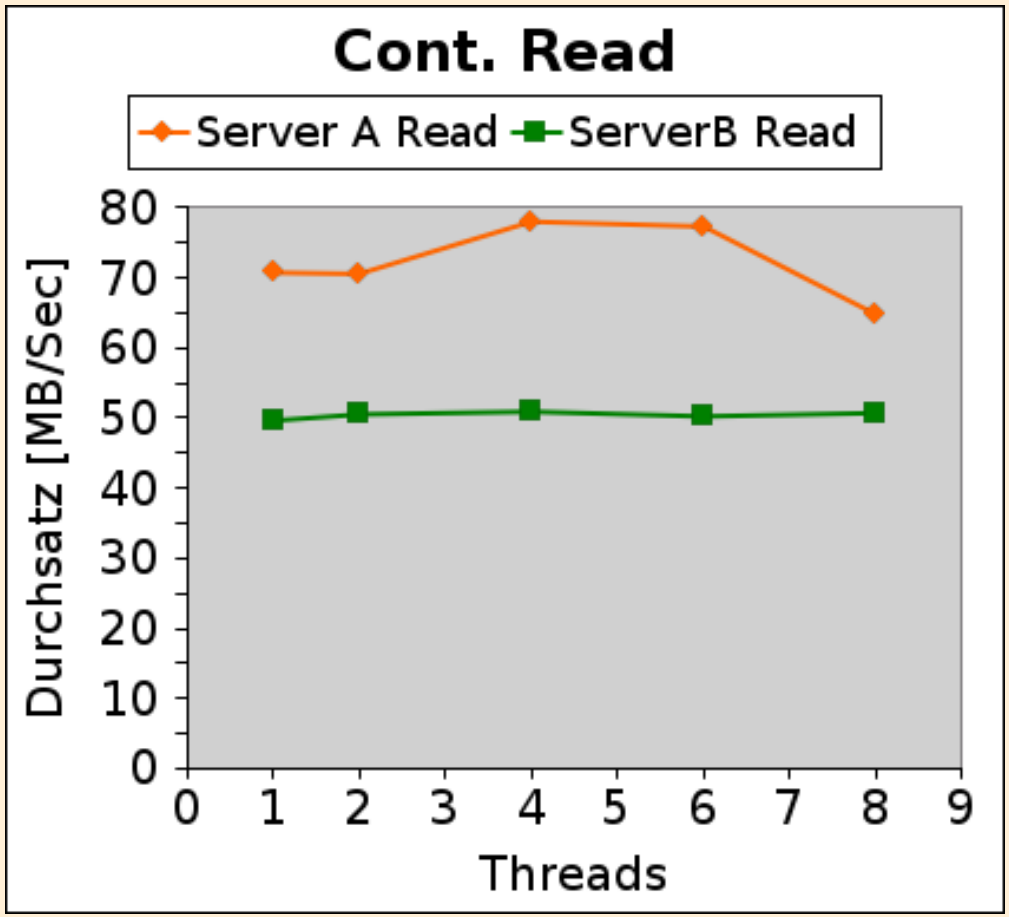


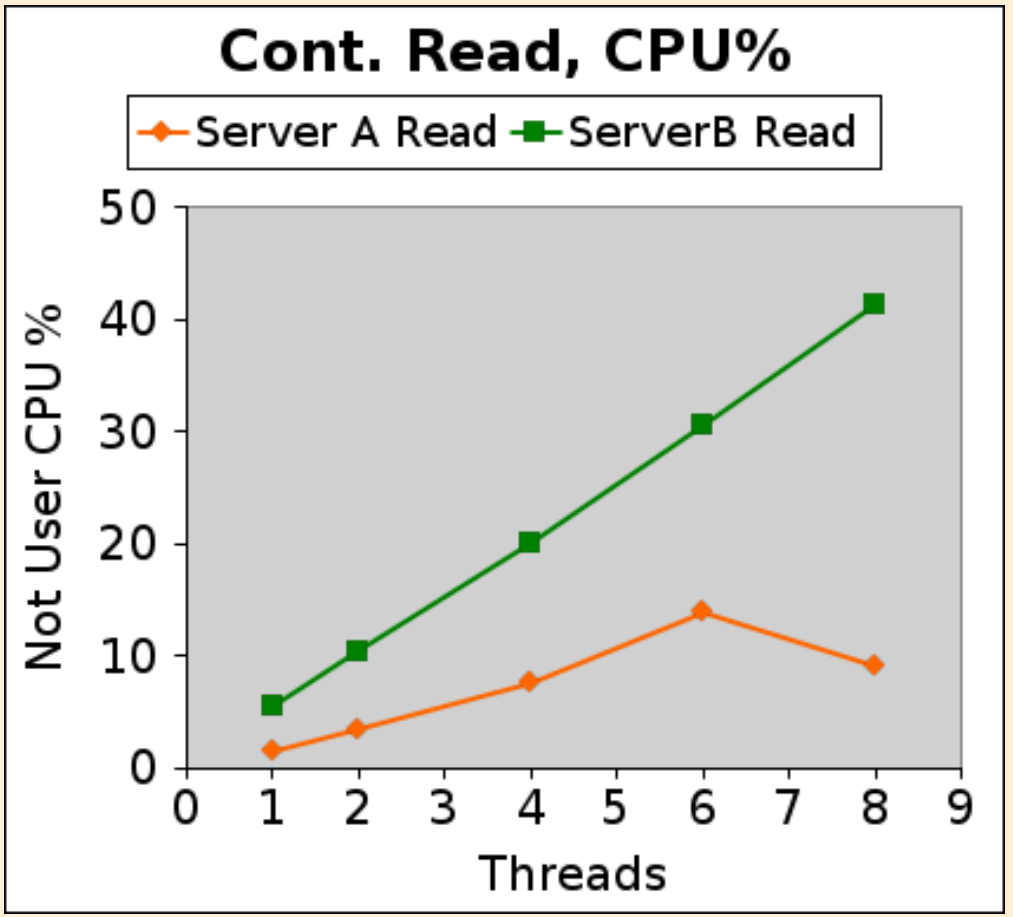
TIOTest

```
Tiotest results for 4 concurrent io threads:
,-----------------------------------------------------------------.
| Item                  | Time     | Rate          | Usr CPU  | Sys CPU |
+-----------------------+----------+---------------+----------+---------+
| Write        4800 MBs |  117.2 s |  40.972 MB/s |   1.1 %  | 124.9 % |
| RandomWrite    16 MBs |   15.2 s |   1.028 MB/s |   0.2 %  |   2.1 % |
| Read         4800 MBs |  105.3 s |  45.586 MB/s |   1.4 %  |  35.8 % |
| RandomRead     16 MBs |   32.7 s |   0.478 MB/s |   0.1 %  |   1.2 % |
`-----------------------------------------------------------------'
Tiotest latency results:
,-------------------------------------------------------------------------.
| Item         | Average latency | Maximum latency | % >2 sec | % >10 sec |
+--------------+-----------------+-----------------+----------+-----------+
| Write        |        0.367 ms |     1173.328 ms |  0.00000 |   0.00000 |
| RandomWrite  |        7.592 ms |     1529.972 ms |  0.00000 |   0.00000 |
| Read         |        0.340 ms |     4082.375 ms |  0.00033 |   0.00000 |
| RandomRead   |       31.625 ms |      376.053 ms |  0.00000 |   0.00000 |
|--------------+-----------------+-----------------+----------+-----------|
| Total        |        0.416 ms |     4082.375 ms |  0.00016 |   0.00000 |
`--------------+-----------------+-----------------+----------+-----------'
```

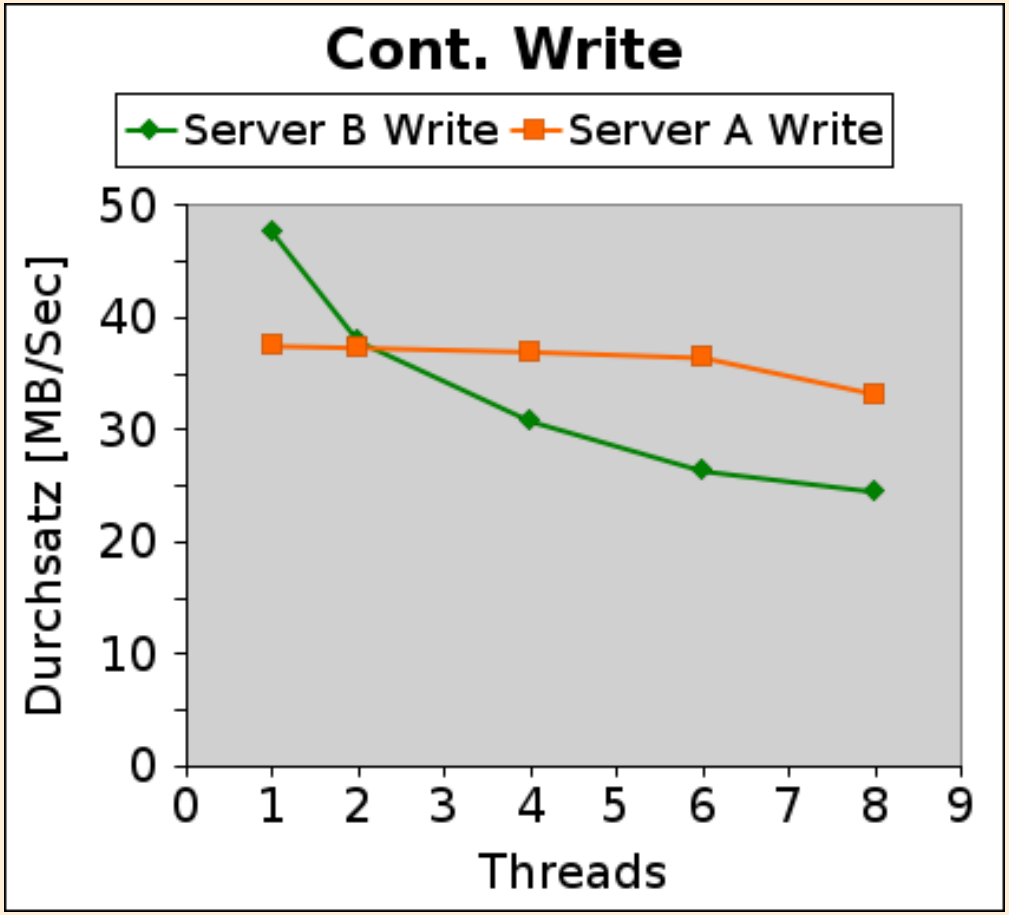
Schalter -T macht es maschinenlesbarer

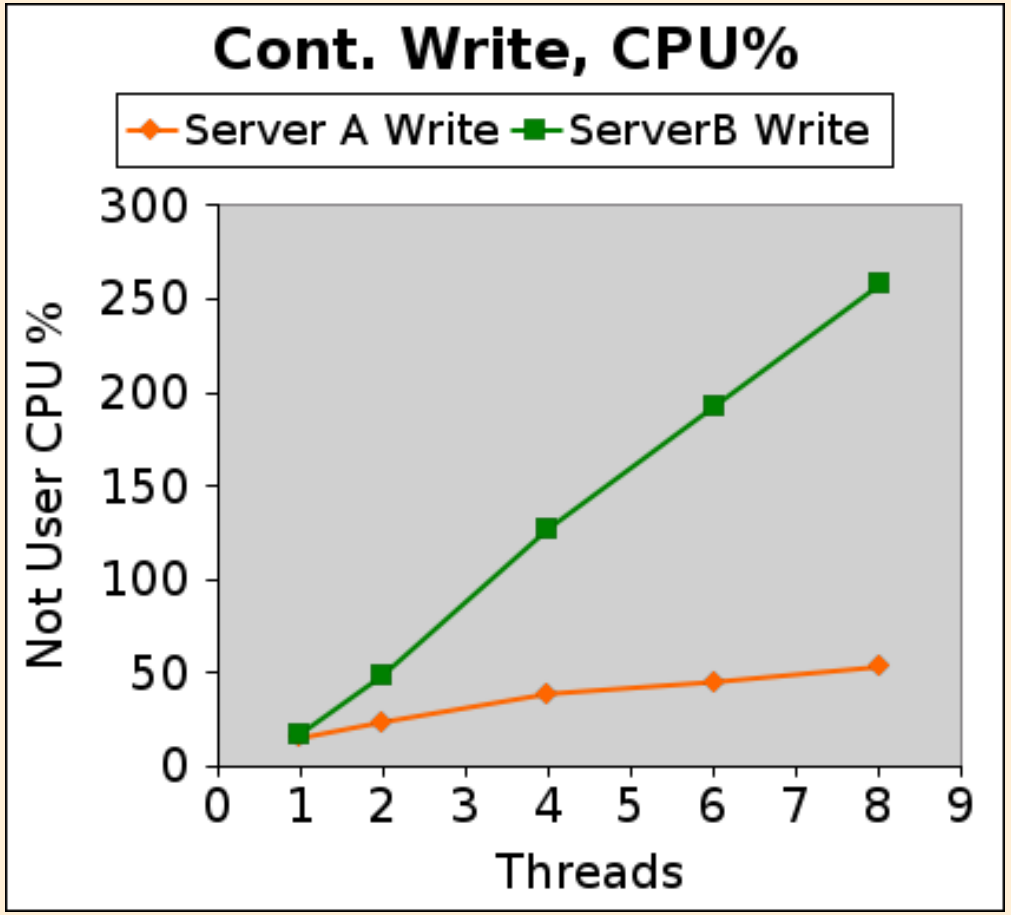
TIOTest lesen



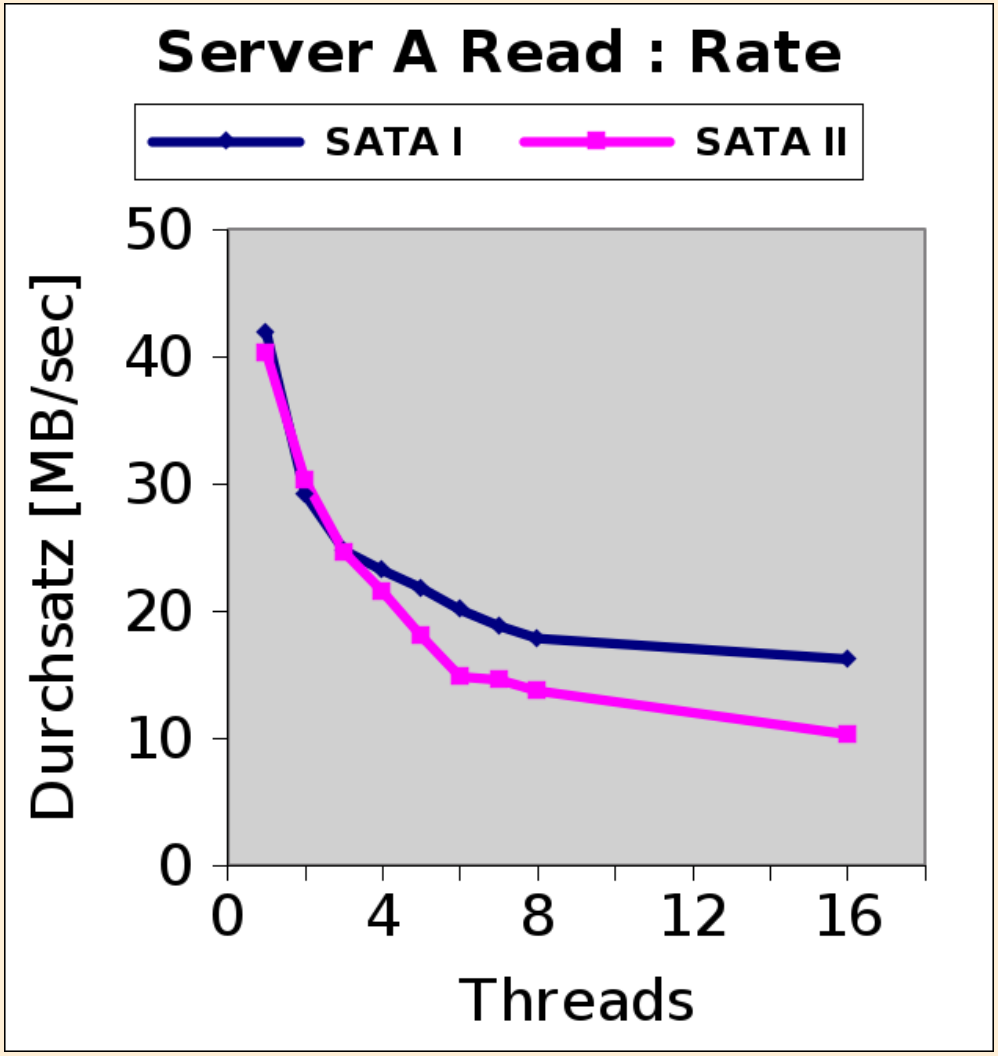


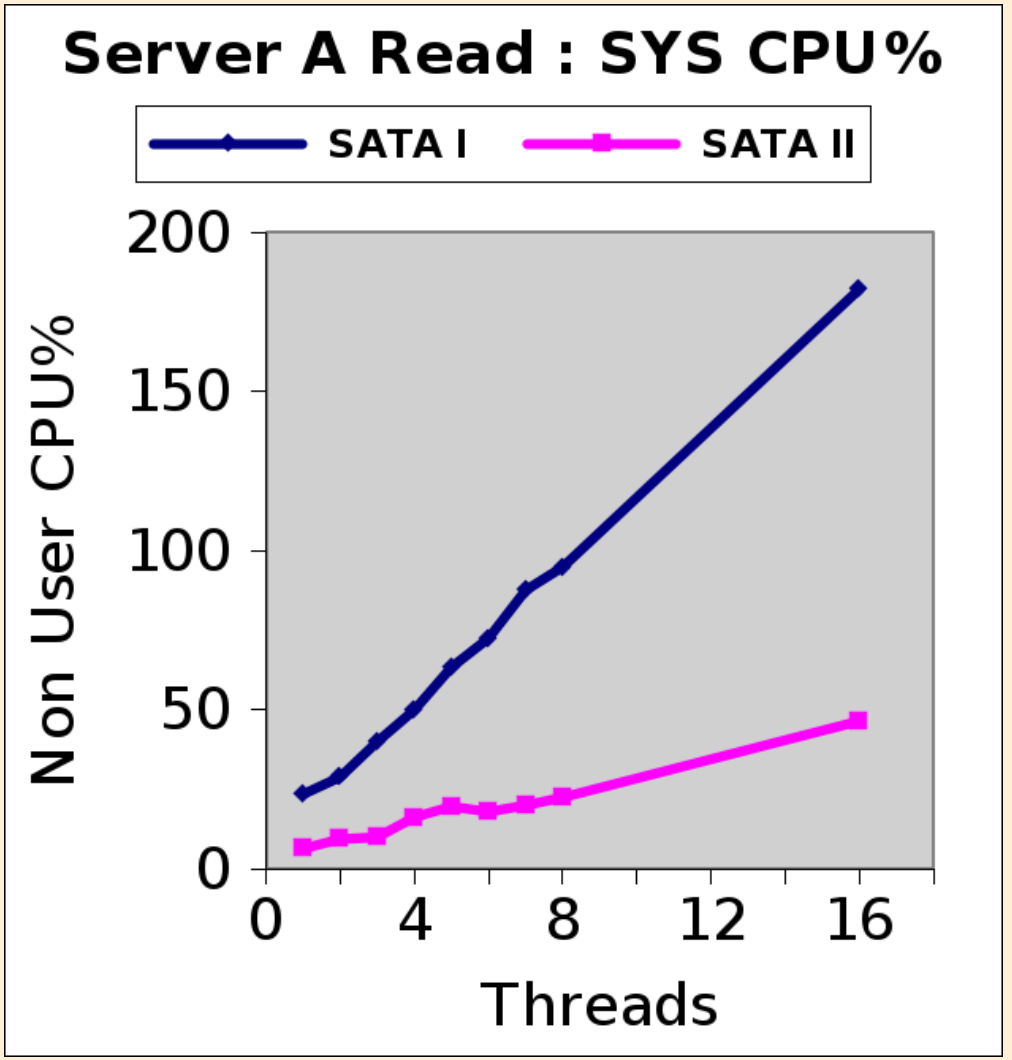
TIOTest schreiben





SATA II Controller:
SATA II HDDs auf SATA I/II gejumpert

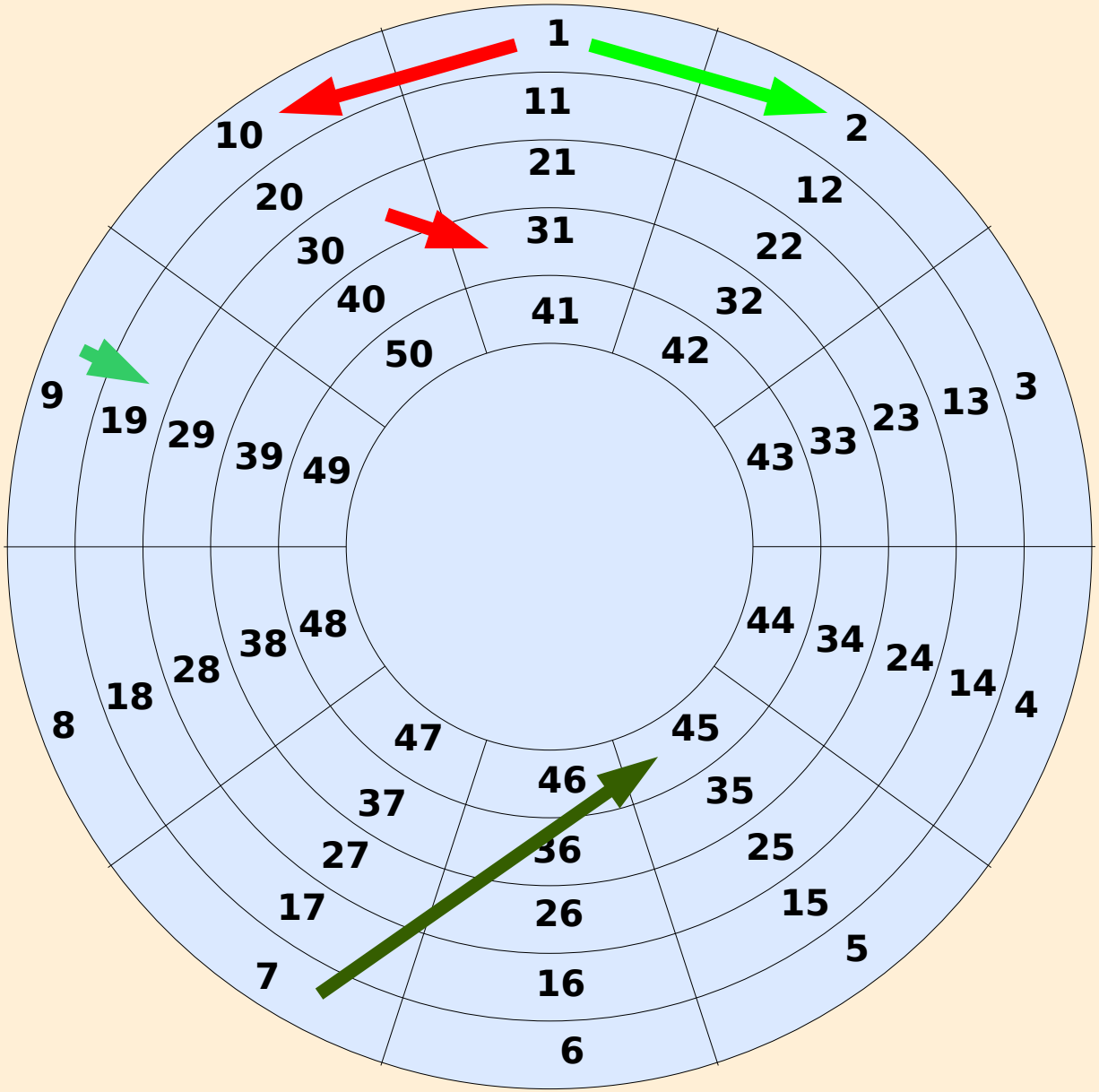




IO Scheduler

- **IO Scheduling: Kosten von IO-Zugriffen durch zeitliche Reorganisation der Zugriffe minmieren.**
- **Voraussetzung: Kosten der IO-Zugriffe sind unterschiedlich und bekannt.**
- **Lokalitäts-Annahme:**
  - **IO-Zugriff auf Blöcke ähnlicher Blocknummer ist billig.**
  - **IO-Zugriff auf Blöcke stark abweichender Blocknummer dagegen teuer.**

Lokalitäts-Annahme



Schön:

1>2 : 0.1 ms

9>19: 1 ms

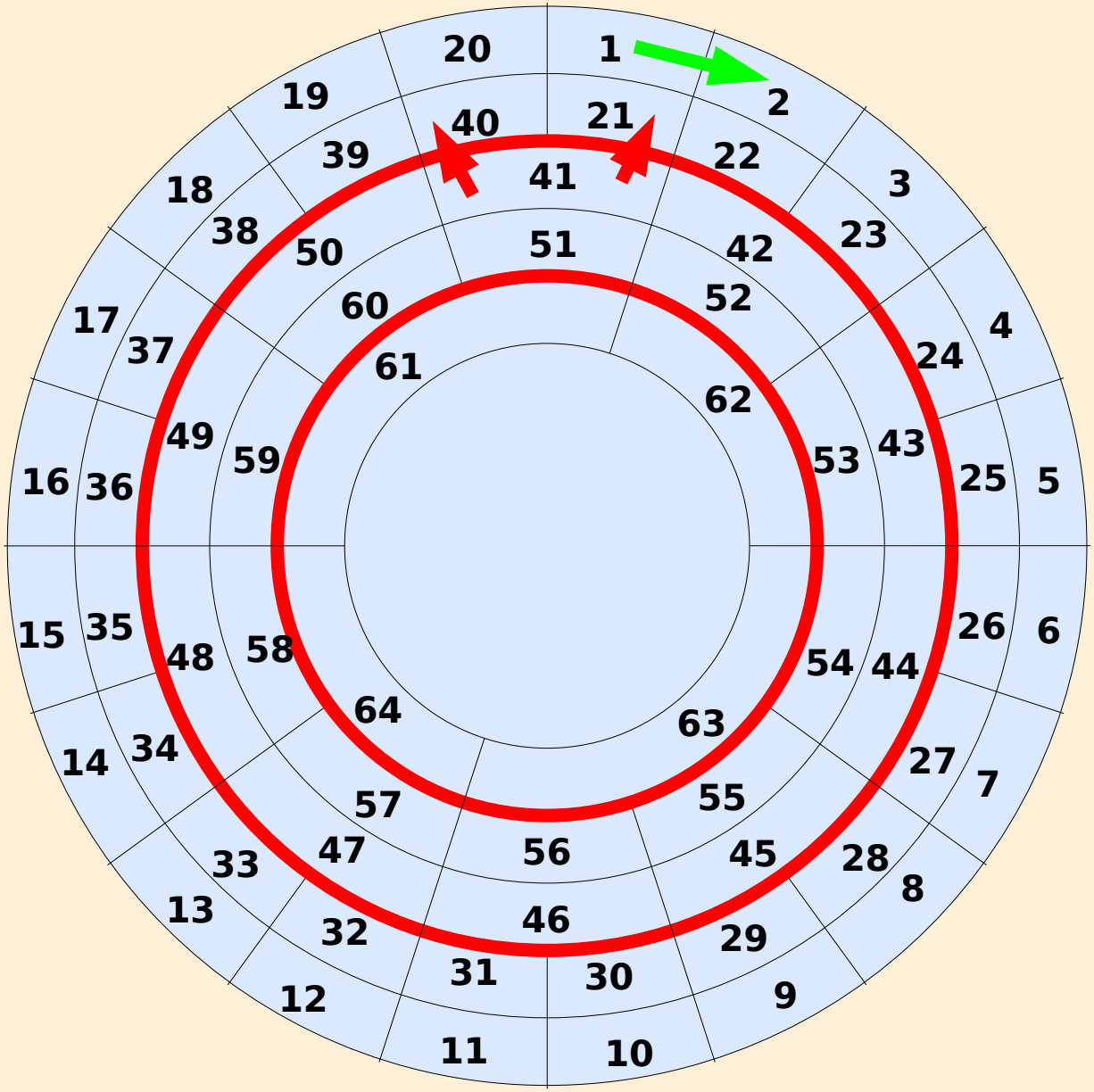
7>45: 9 ms

Ausnahmen:

1>10: 0.1 ms

30>31: 1 ms

Problem Zone-Bit-Recording



Fazit:

Die Annahme der Lokalität muss nicht stimmen.

Die aktuellen IO Scheduler

0 Elevator (Historisch von Linus)

1 NOOP

2 Deadline

3 Anticipatory

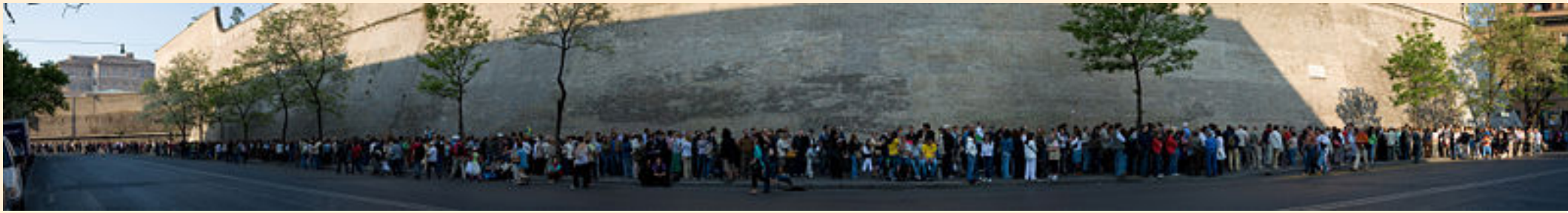
4 CFQ (Complete Fair Queuing)

***elevator*=<*Scheduler*> kernel-param**

oder zur Laufzeit

 # echo <Scheduler> >
/sys/block/<device>/queue/scheduler

NOOP IO Scheduler



Elevator IO Scheduler

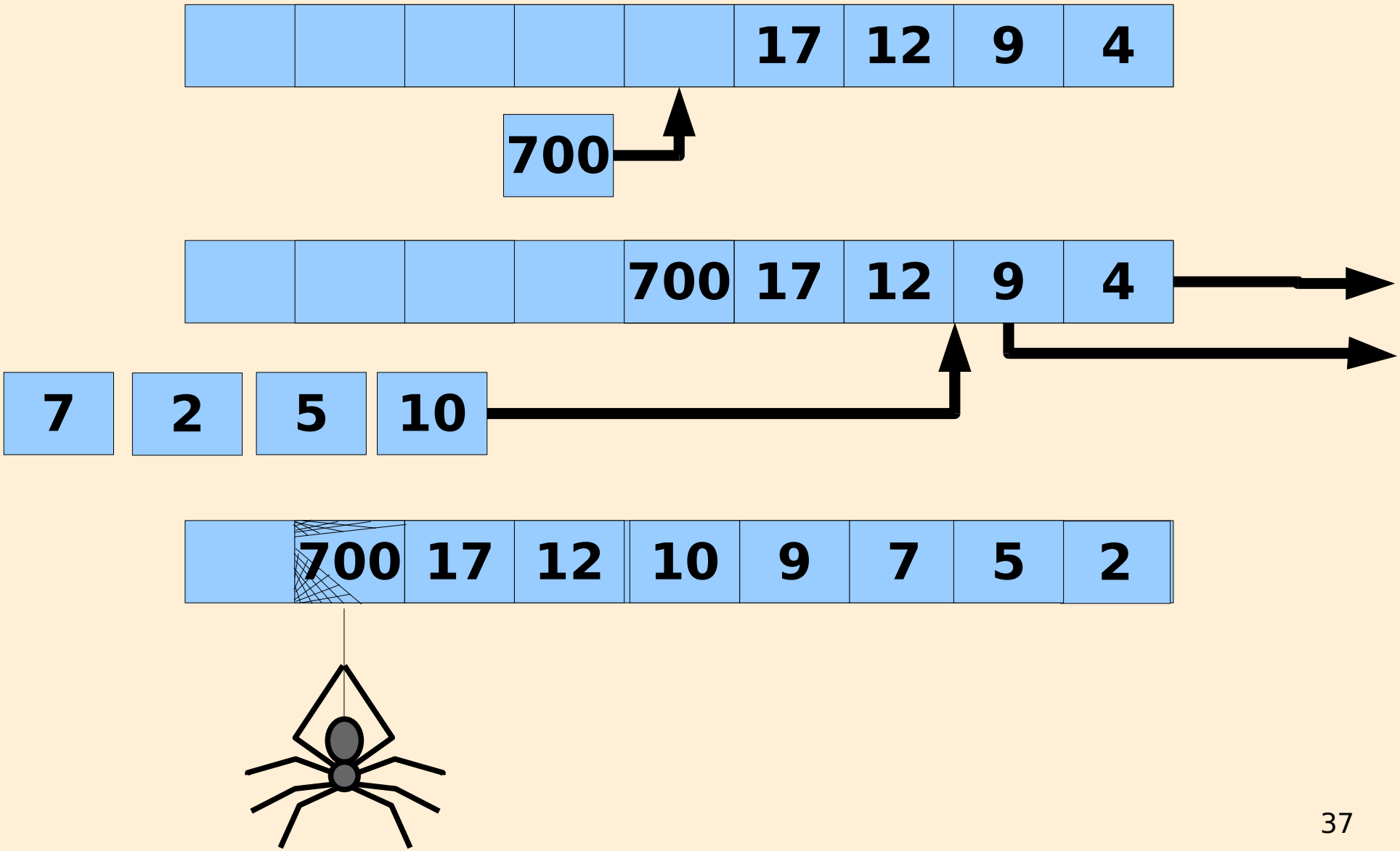
IO Requests mit Blocknummern

In einen Buffer

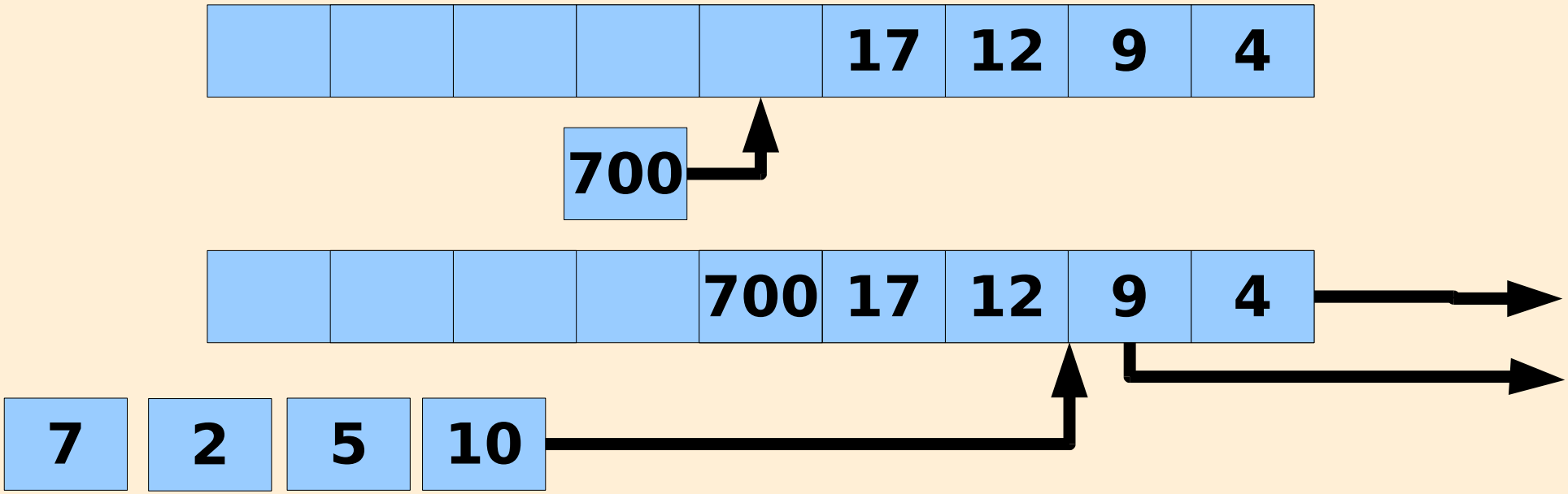
nach Blocknummern einsortieren

Buffer voll : Senden!

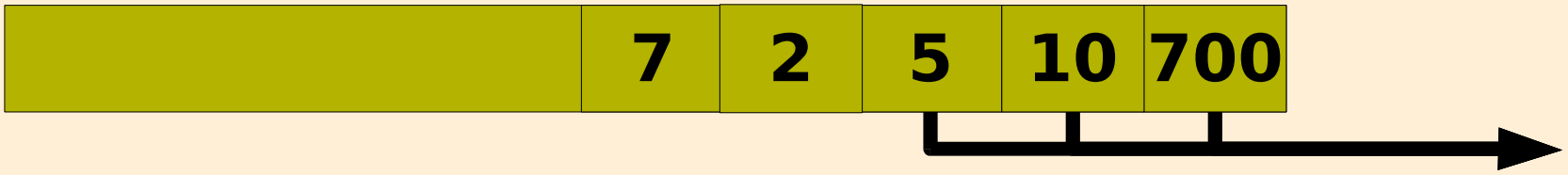
Probleme mit dem Elevator



Deadline verbessert Elevator

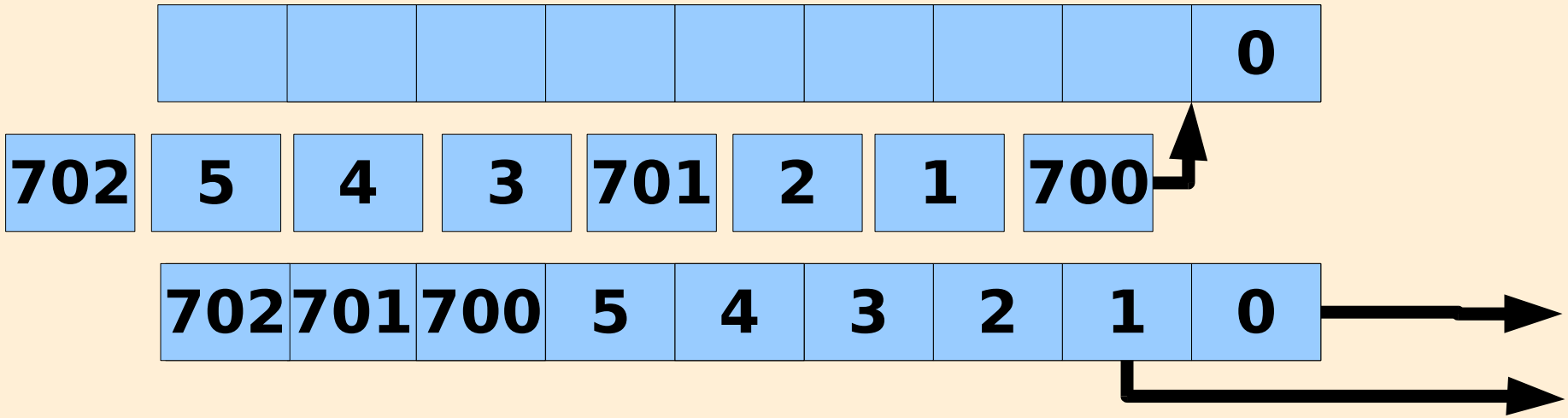


Zusätzlicher FIFO mit Timeout Timern

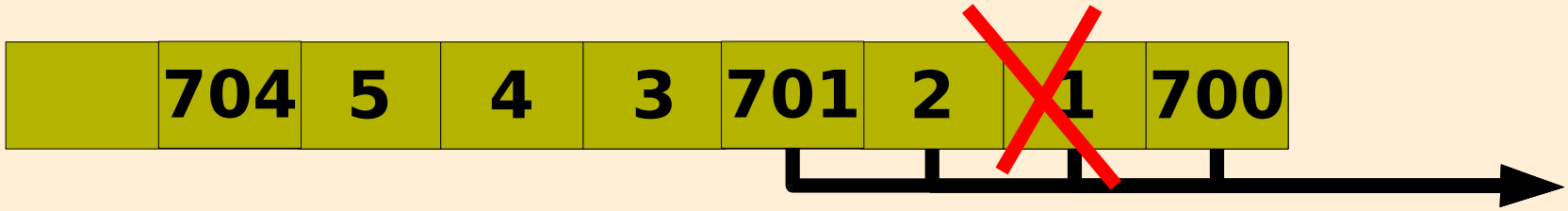


Timeout abgelaufen: einige Einträge aus FIFO senden

Problem bei Deadline



Zusätzlicher FIFO mit Timeout Timern



Enstehende IO Requests

Bessere Anordnung der IO Requests

Anticipatory Scheduler : AS

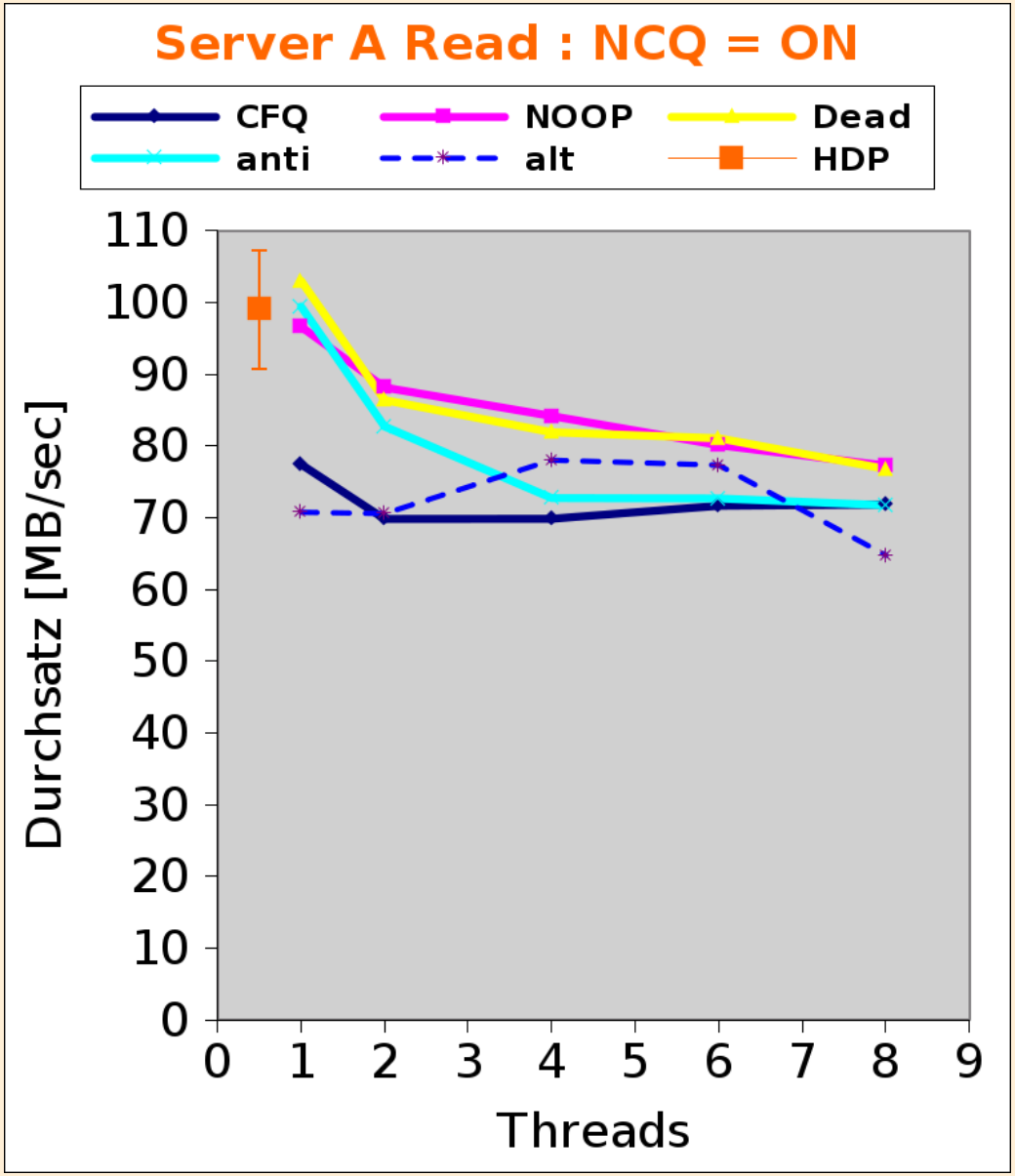
- **Pro Prozess : Statistik über Read/Write Frequenz**
- **Wartepause in Ordnung dieser Frequenz, um Requests eines Prozesses zusammenzufassen**

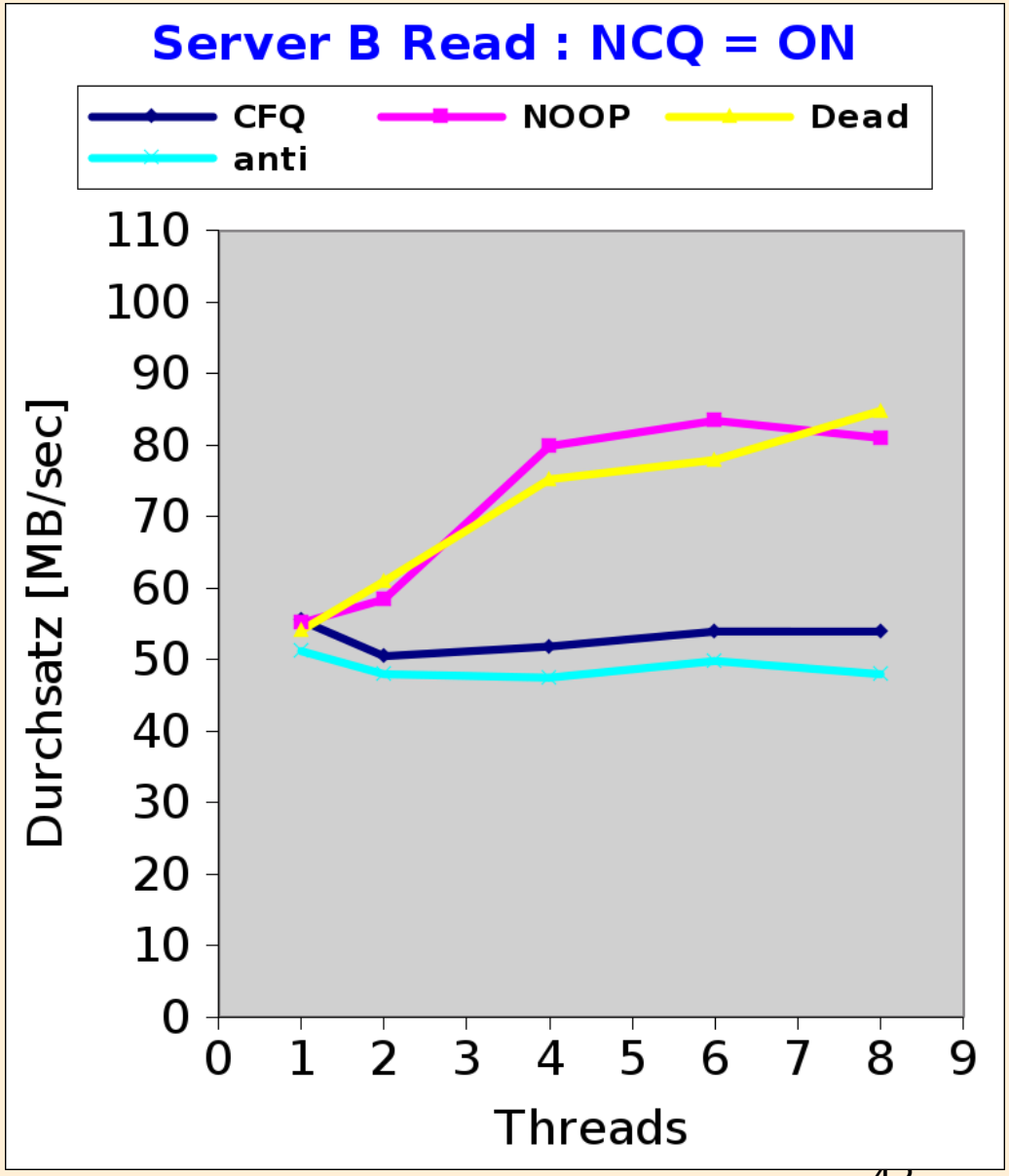
CFQ Scheduler

- **Anderer Ansatz als Deadline und AS**
- **IO-Bandbreite wird gleichmässig allen Prozessen zugeteilt**
- **Prozessgruppen werden betrachtet**
- **Prioritäten können definiert werden**
- **Garantiert Bandbreite aber nicht minimale Latenzzeiten und maximalen Durchsatz.**
- **Gut für interaktive Applikationen/Desktops**
- **Standard bei vielen Distributionen**

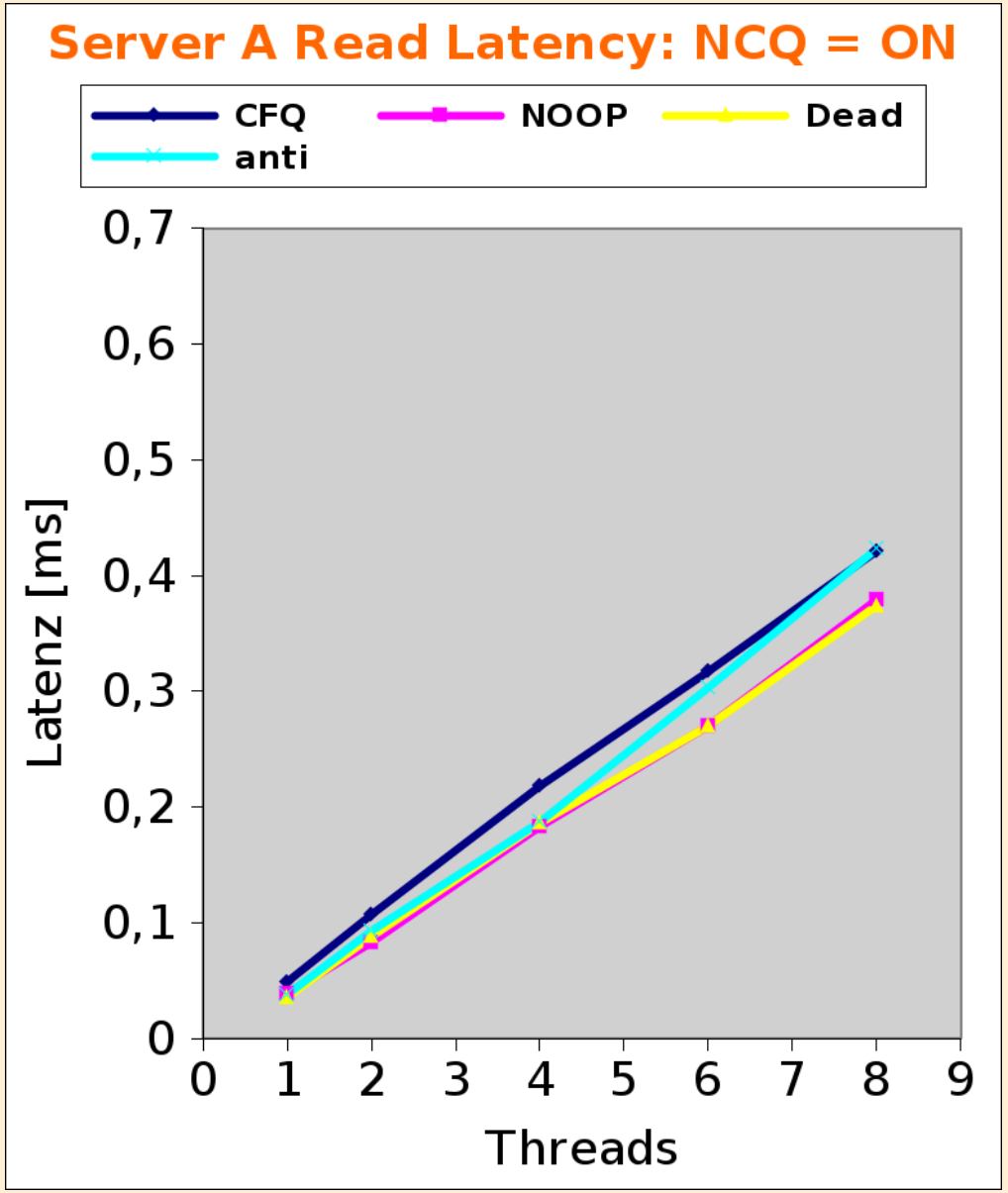
Scheduler Read Rate

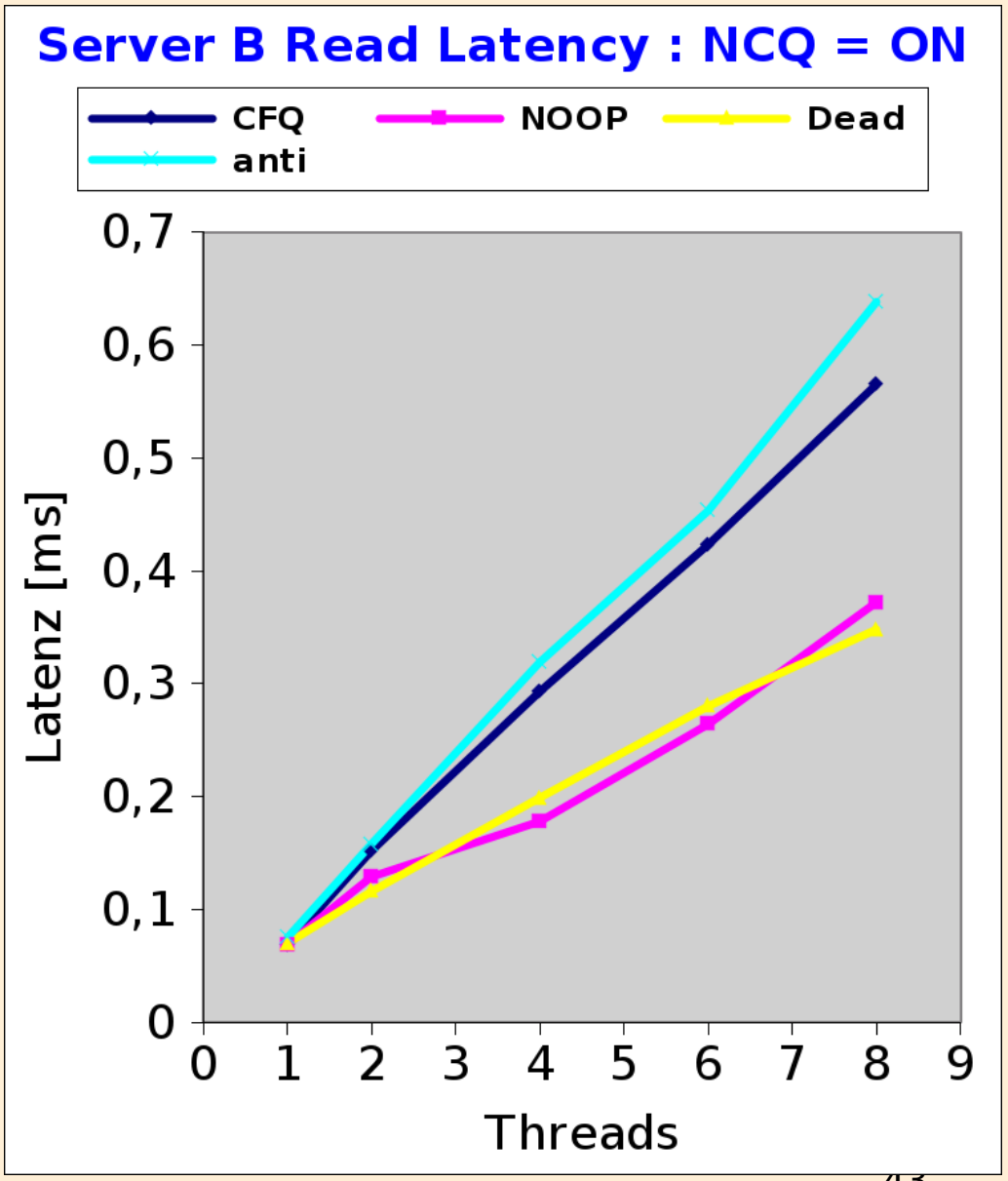
3ware 9550 RAID 5 vs. 3ware 8006 RAID 1



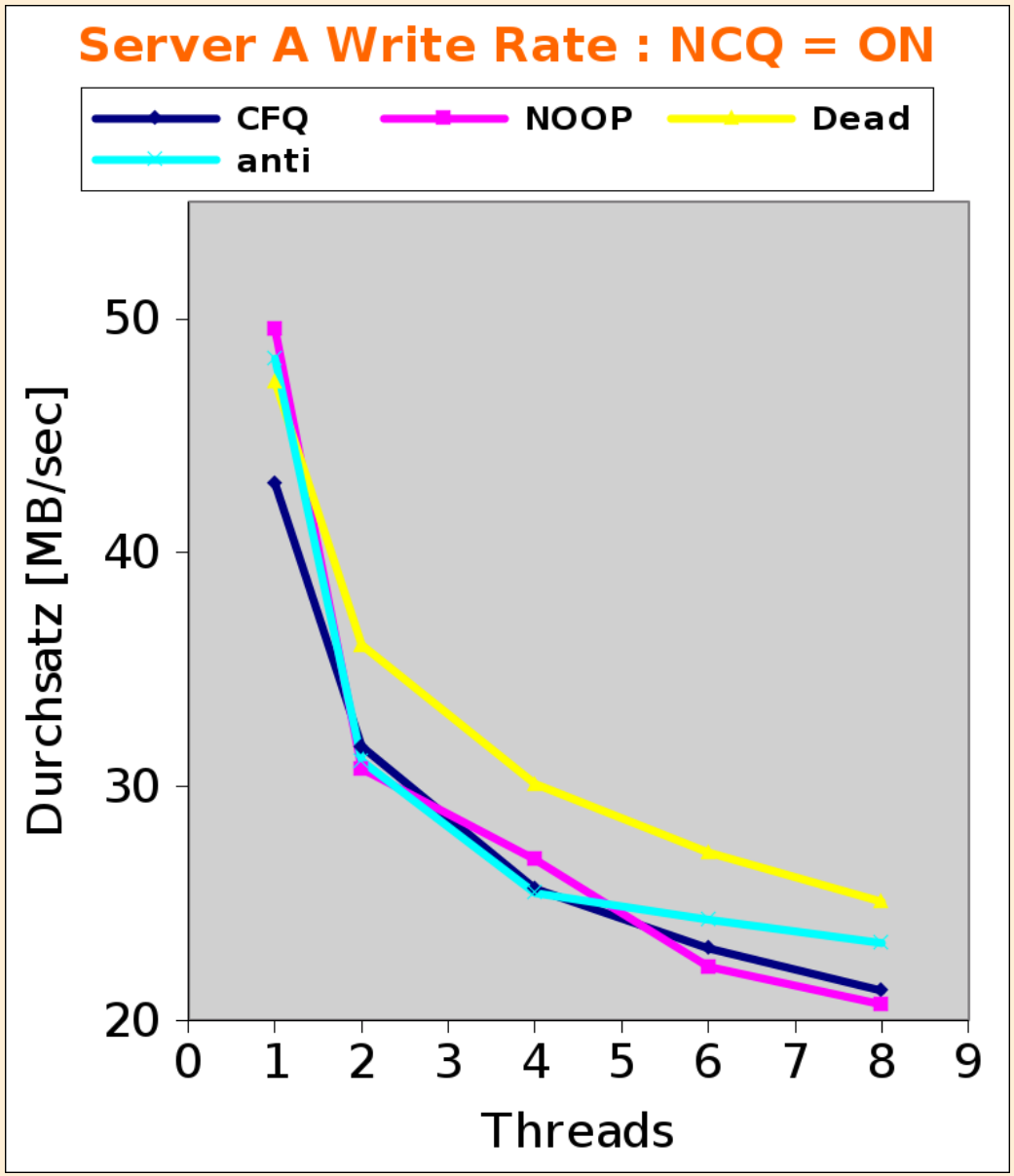


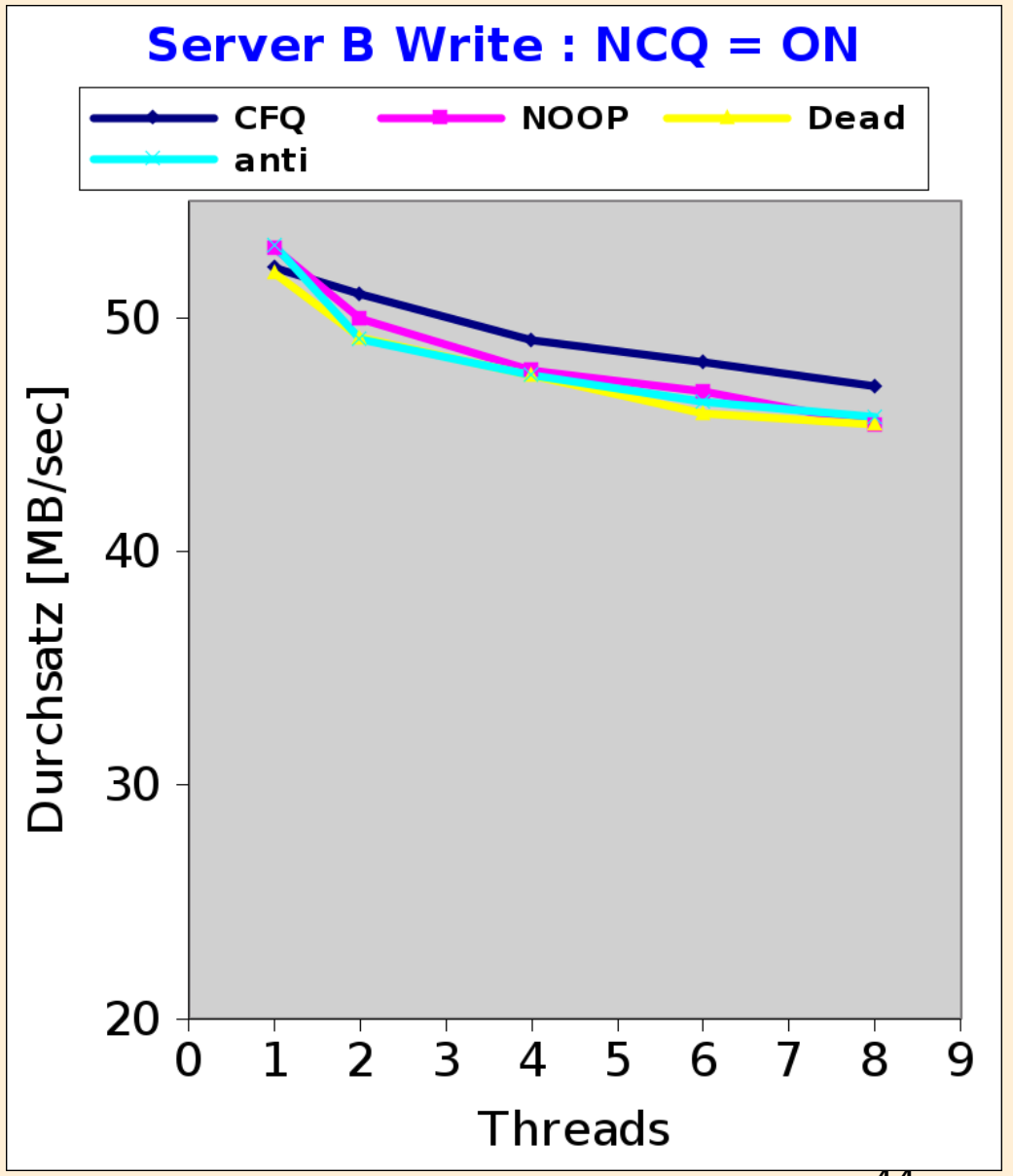
Scheduler Read Latenz



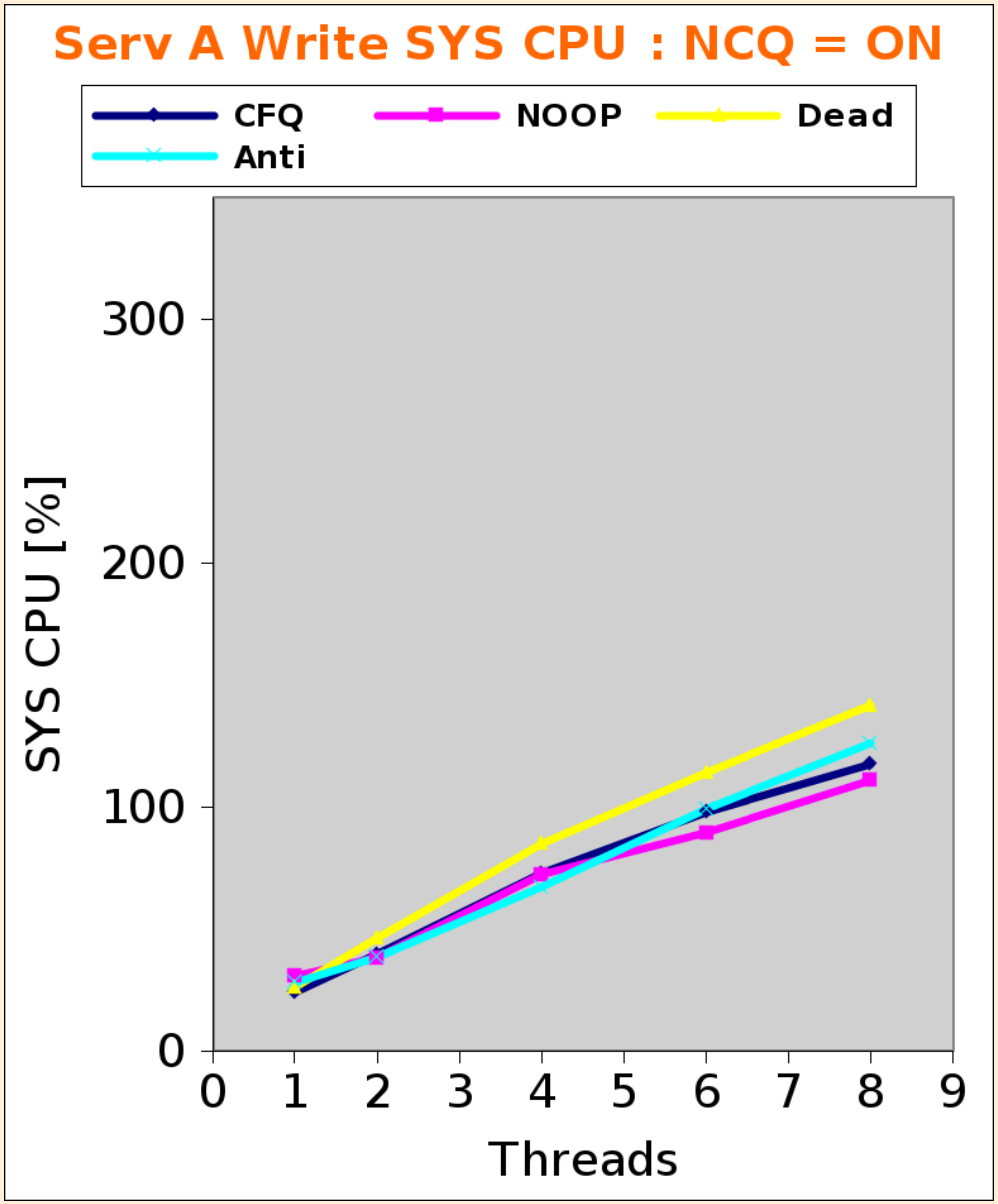


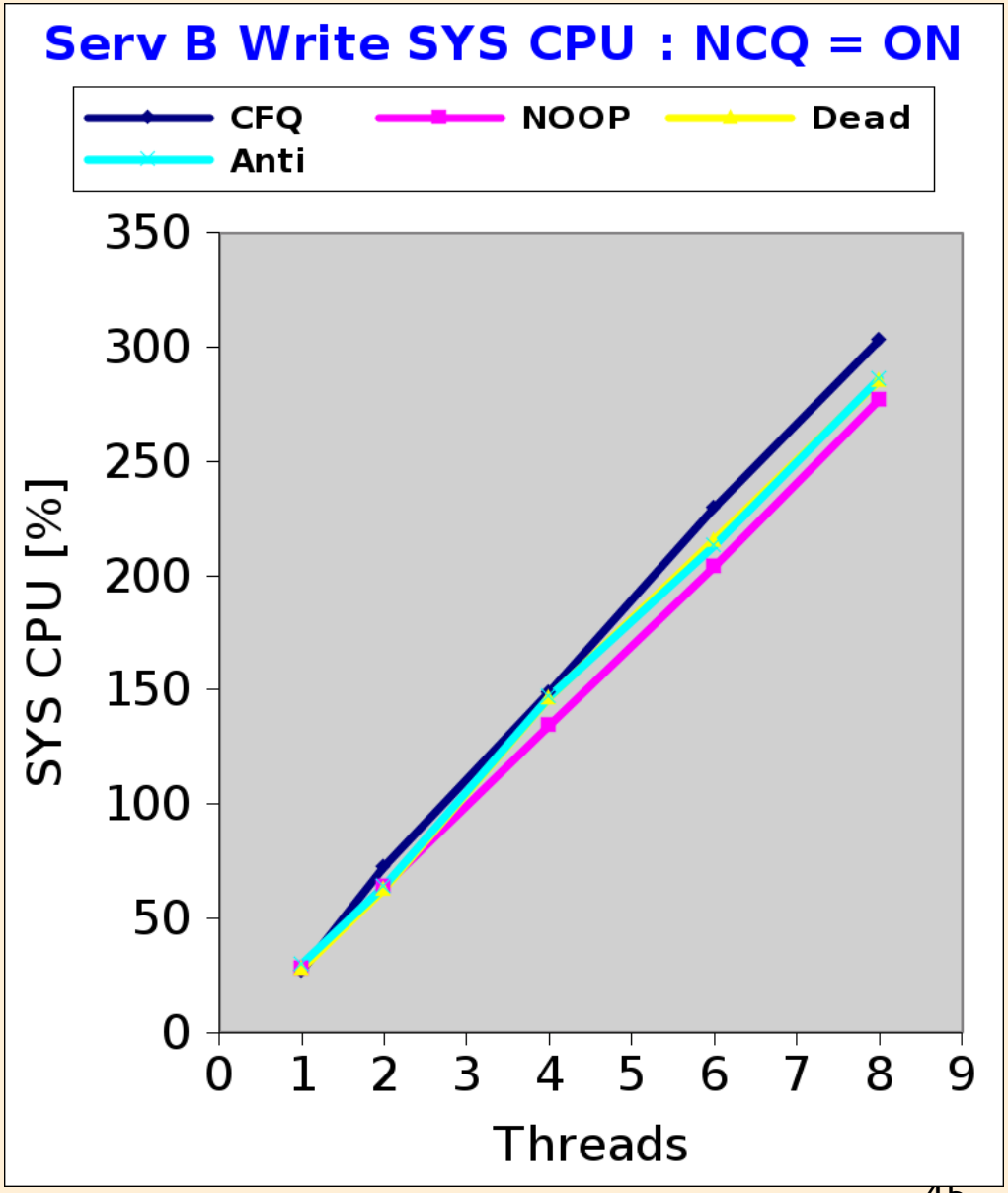
Scheduler Write Rate



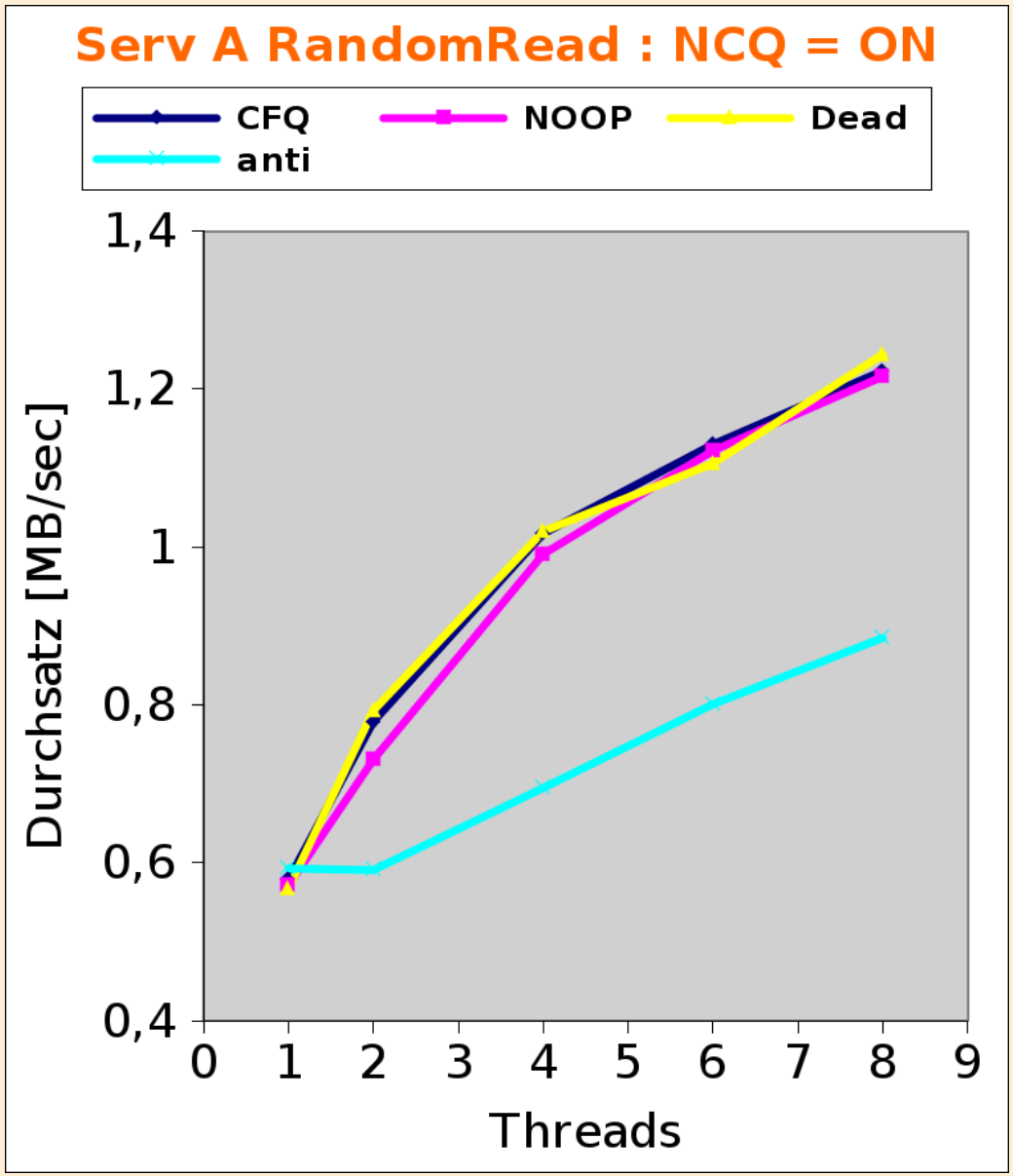


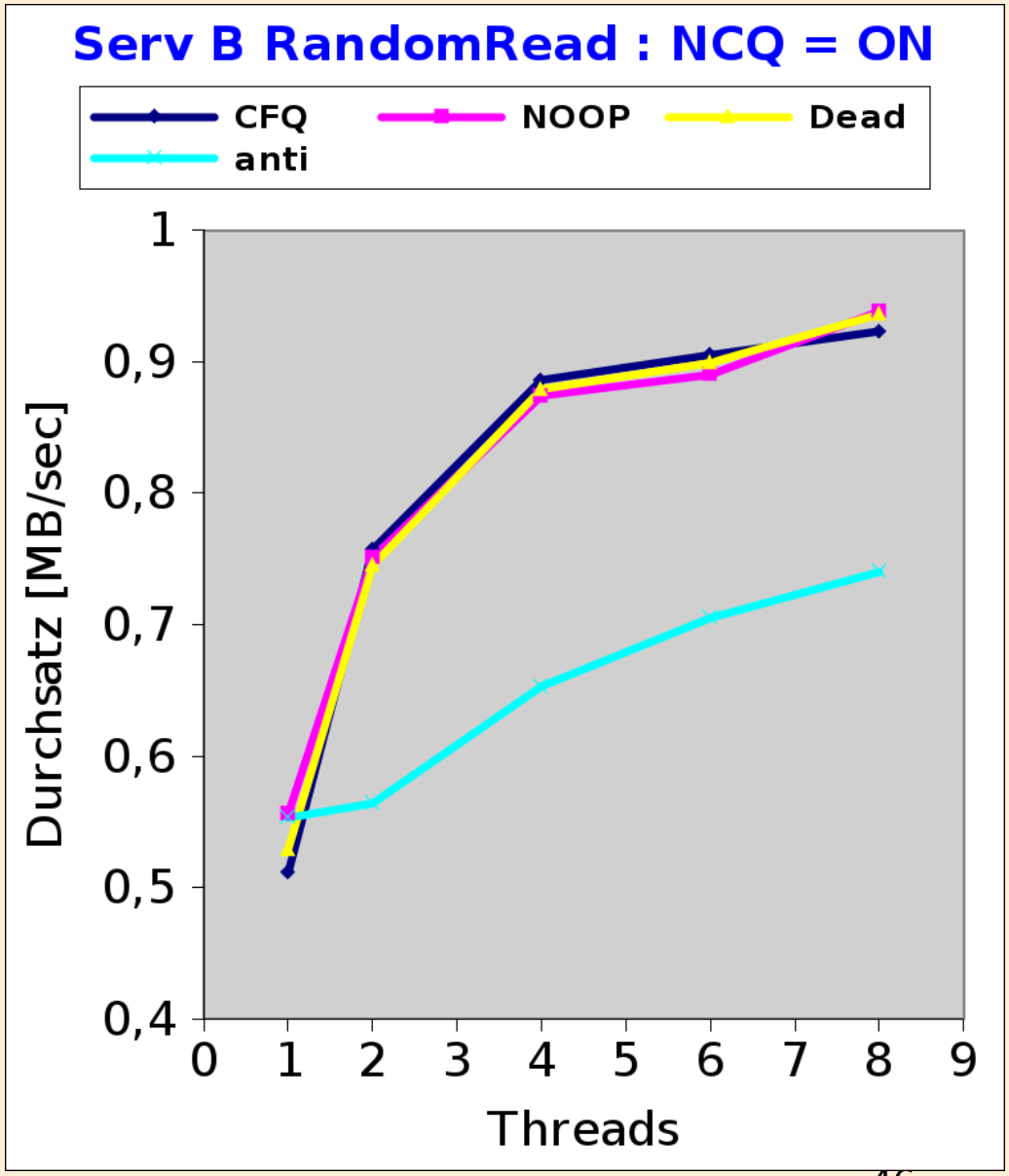
Scheduler Write SYS CPU%



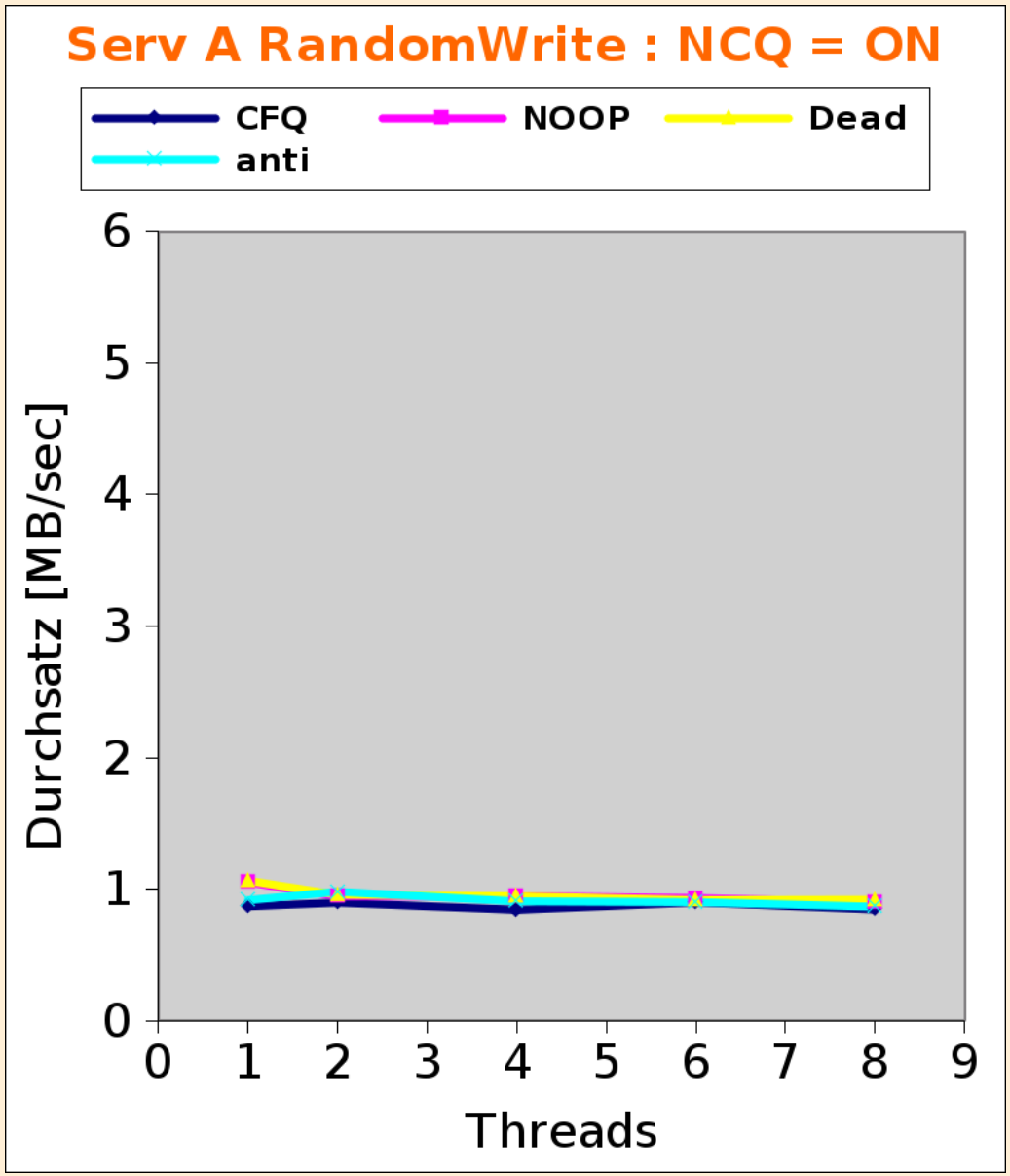


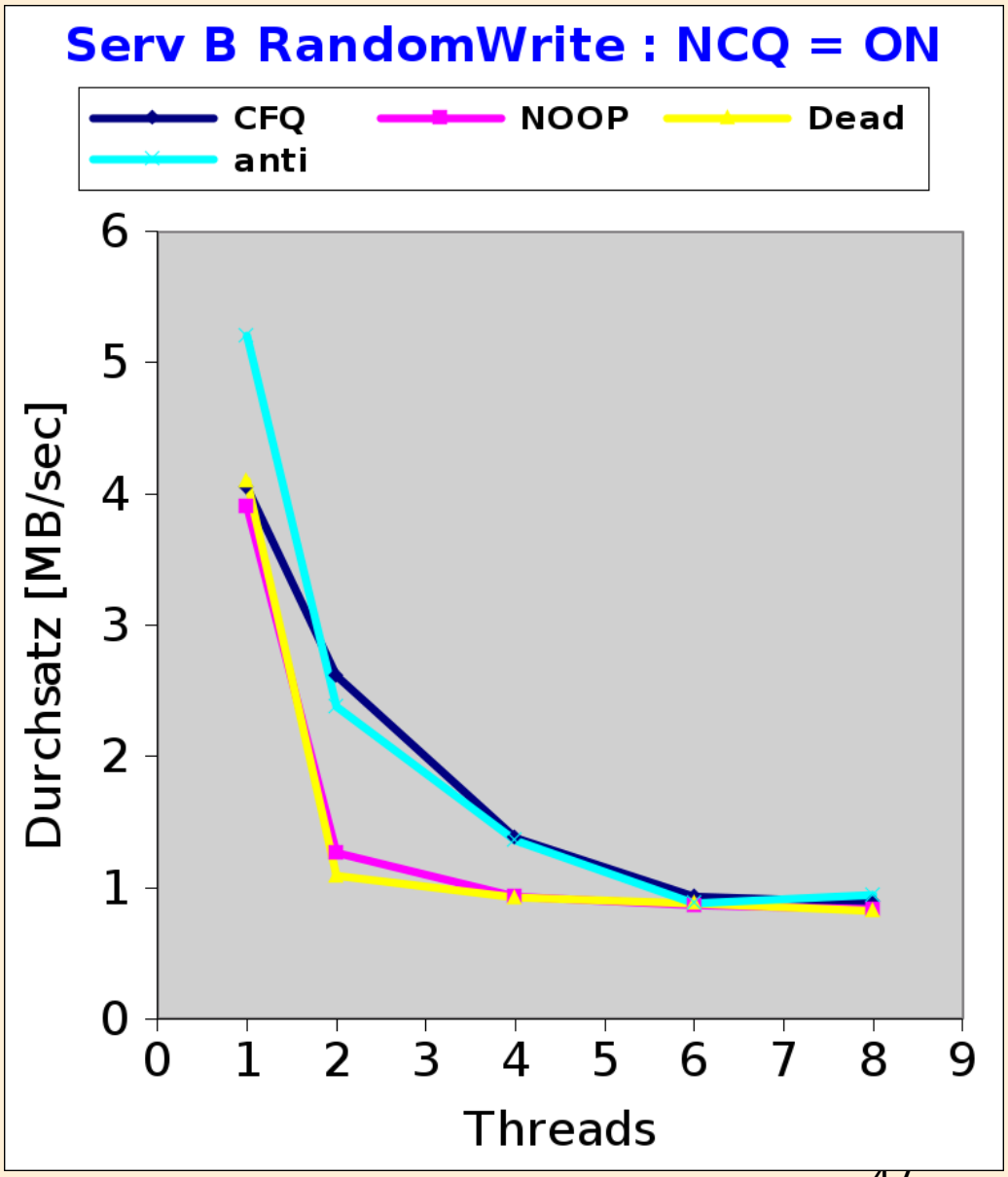
Scheduler RandomRead Rate





Scheduler RandomWrite Rate





Native Command Queuing (NCQ)
Doppeltes Scheduling!

Kernel

IO Scheduler:
NOOP
Deadline
Anticipatory
CFQ

Kennt Prozess-Blocknumber Beziehung
Hat nur Vermutungen über IO-Kosten

RAID Controller
NCQ-Fähig

Blocknummer nach RAID Umsetzung

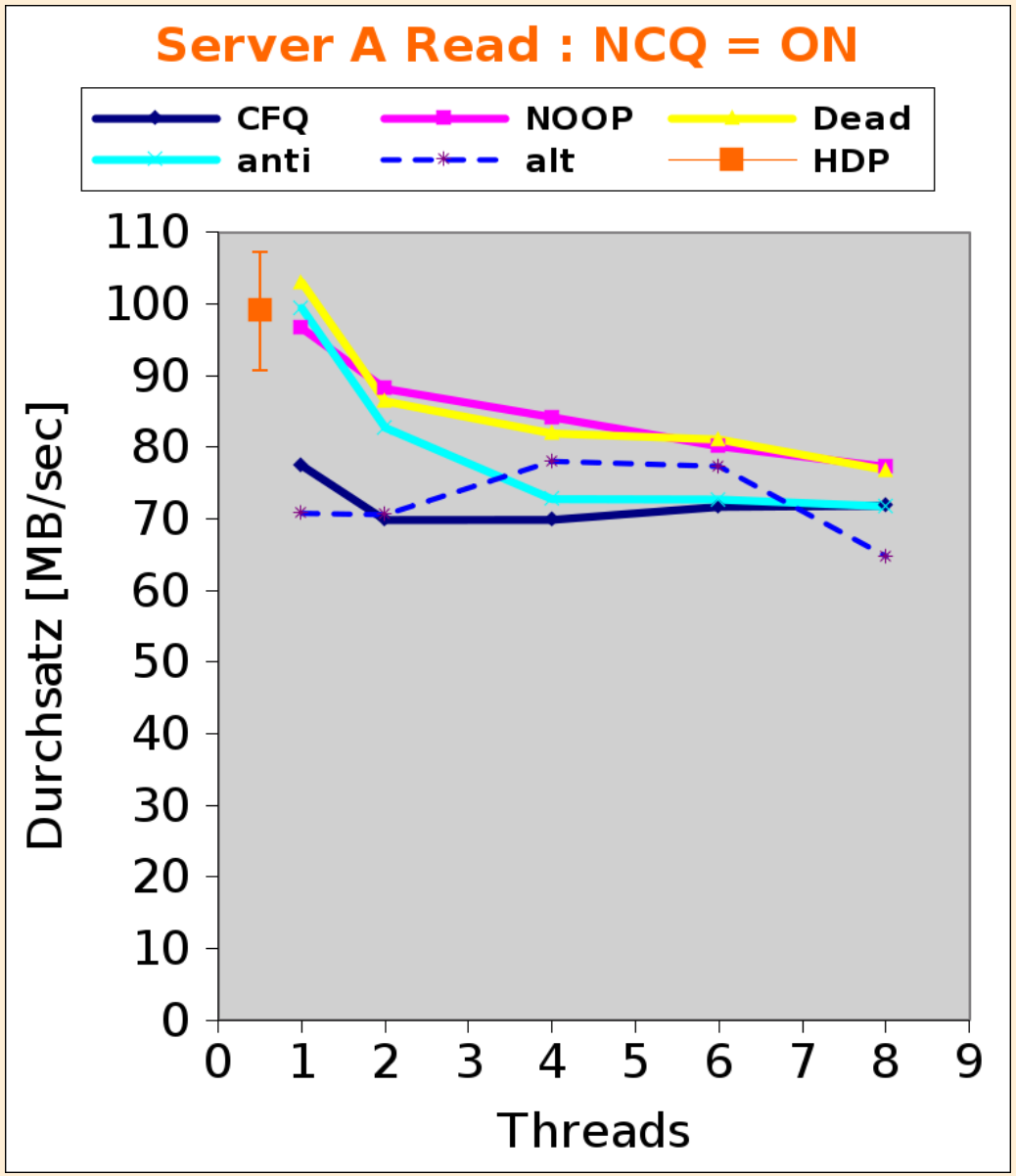
Sorgt für Redundanz der Blöcke

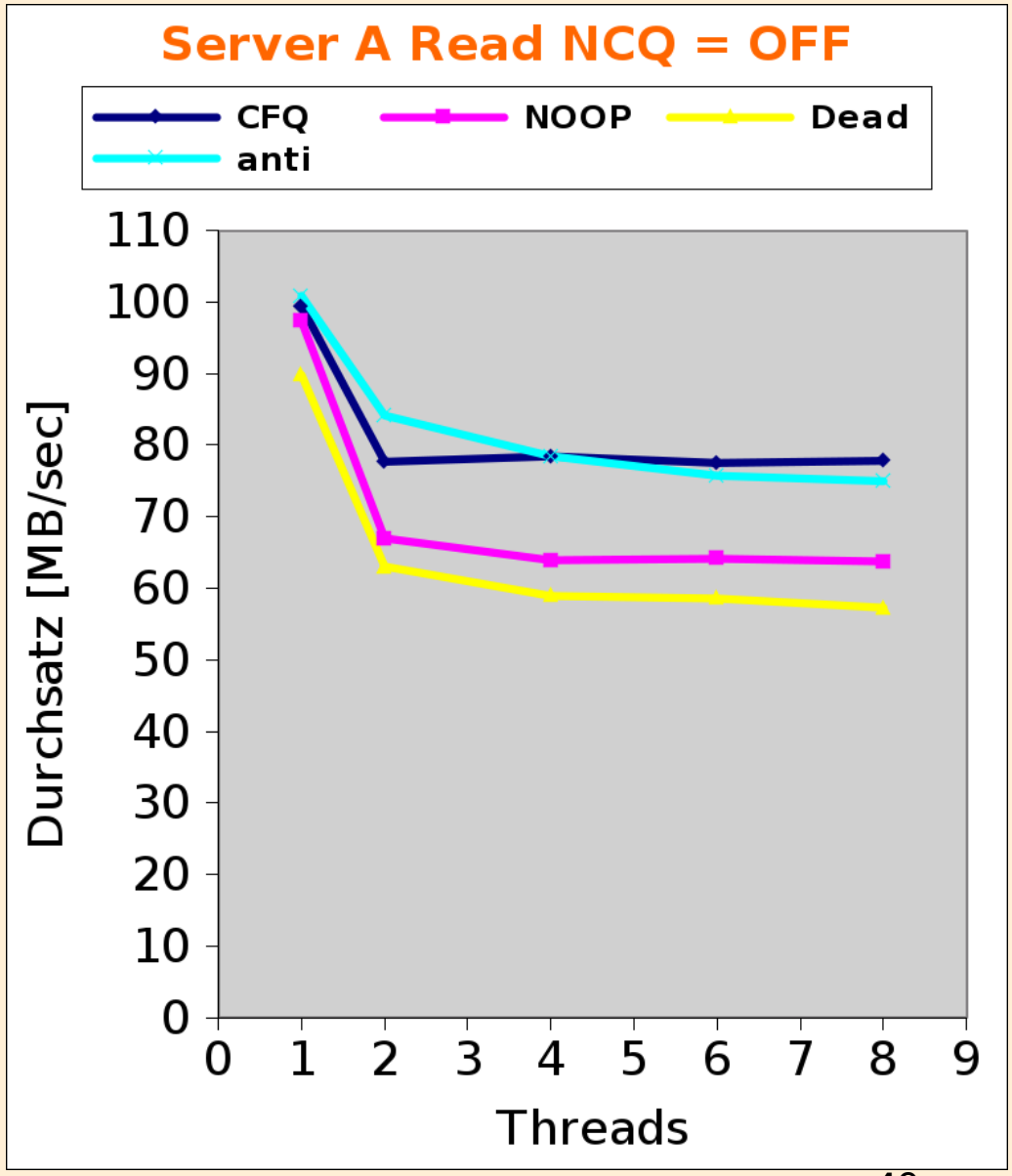
HDDs
NCQ:IOSched

Blocknummer nach Kopf, Spur, Scheibe Umsetzung

Kennt Kosten für Zugriff auf Blöcke ganz genau
Keine Ahnung von Prozessen

Scheduler Read NCQ=ON vs. NCQ=OFF





3ware 9550: NCQ ON vs. NCQ OFF

CFQ

	Rate	Lat.	CPU%
Write	**OFF**	**OFF**	**ON**
Read	**OFF**	**OFF**	**ON**
WriteRand	**ON**	**ON**	**OFF**
ReadRand	**ON**	**ON**	**OFF**

DEADLINE

	Rate	Lat.	CPU%
Write	**ON**	**ON**	**OFF**
Read	**ON**	**ON**	**OFF**
Write Rand	**ON**	**ON**	**OFF**
Read Rand	**ON**	**ON**	**OFF**

3ware 8006-2: NCQ ON vs. NCQ OFF

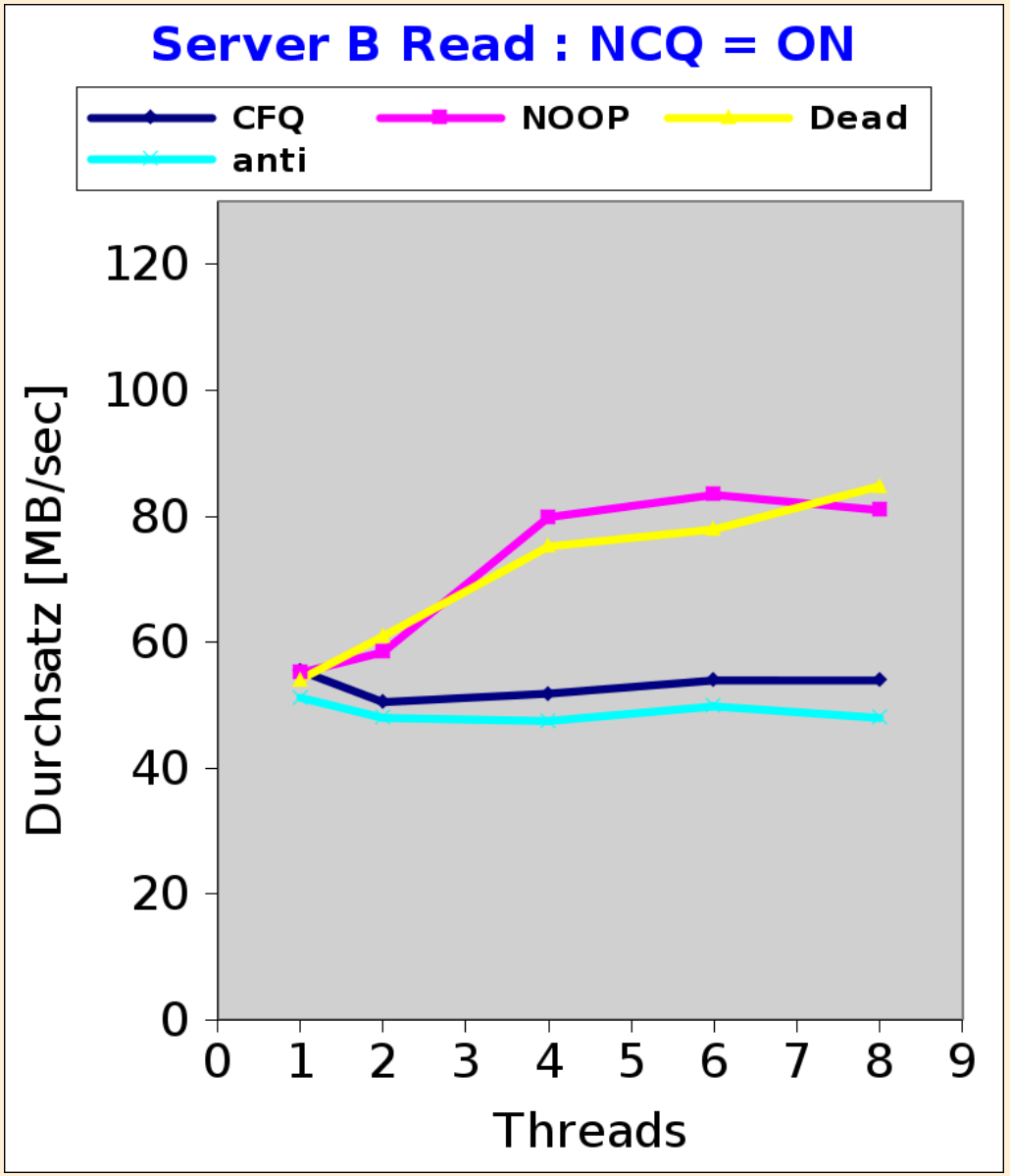
CFQ

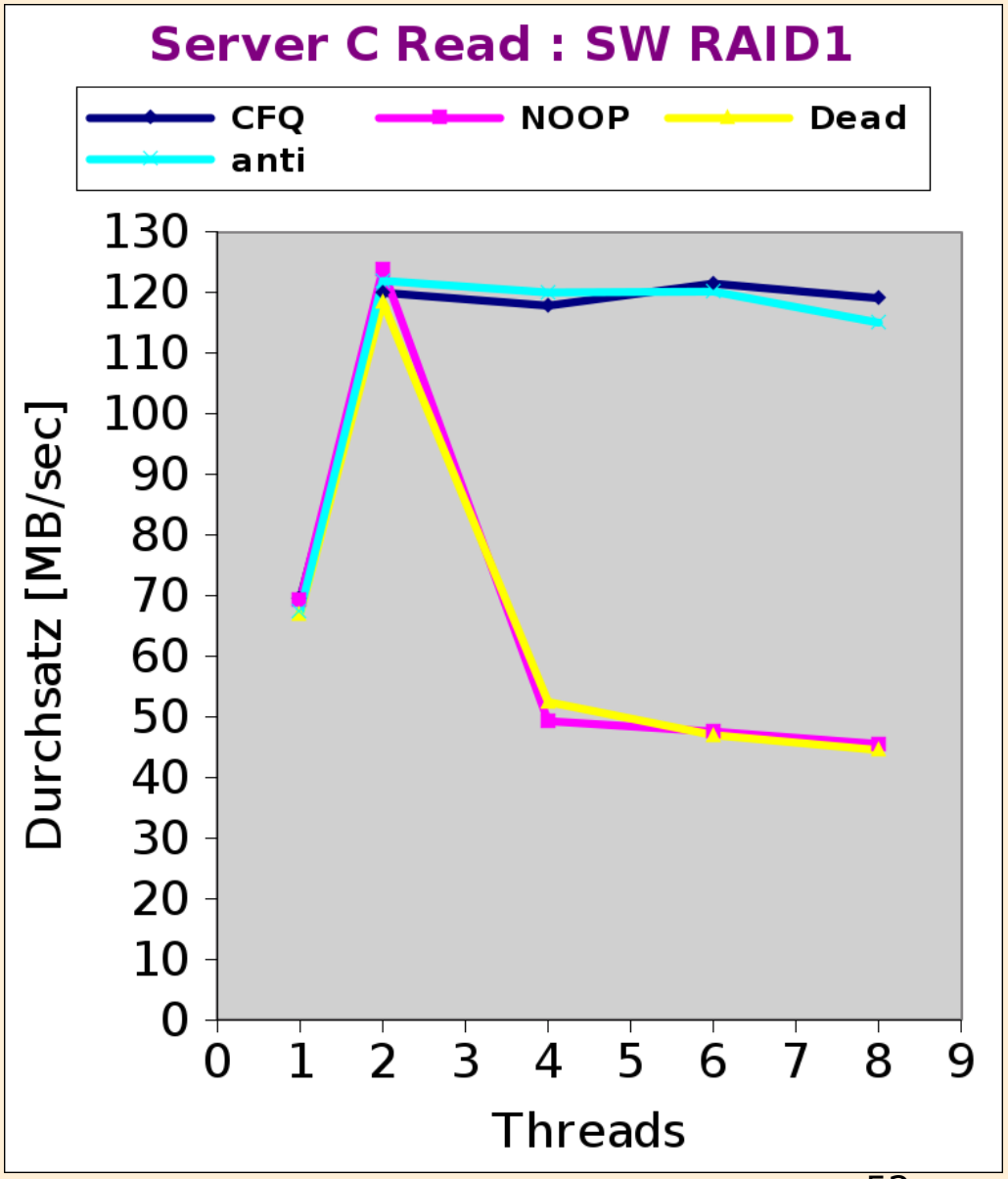
	Rate	Lat.	CPU%
Write	=	ON	=
Read	=	=	=
WriteRand	ON	ON	=
ReadRand	ON	ON	=

DEADLINE

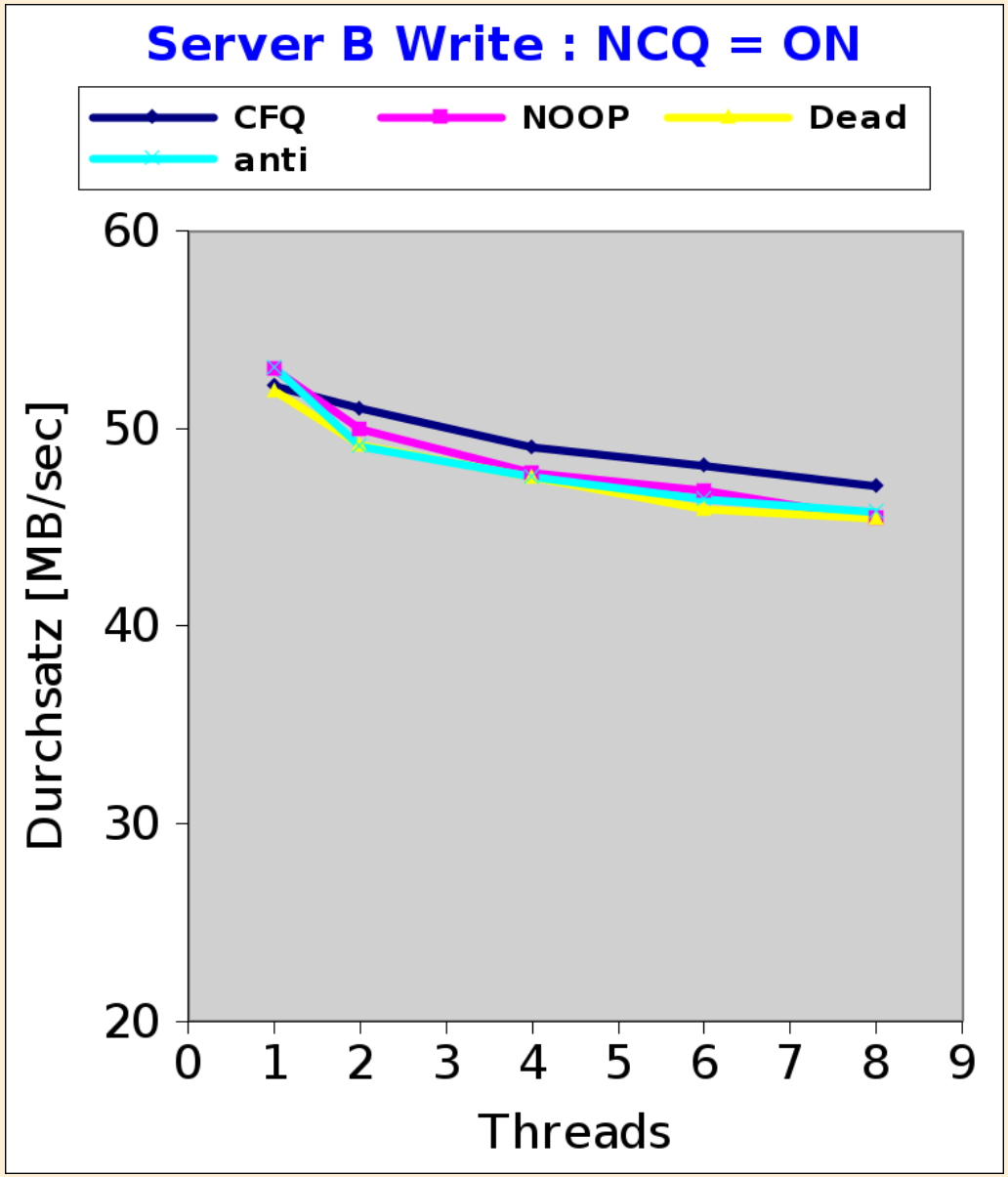
	Rate	Lat.	CPU%
Write	ON	ON	OFF
Read	ON	ON	OFF
Write Rand	ON	=	OFF
Read Rand	ON	ON	=

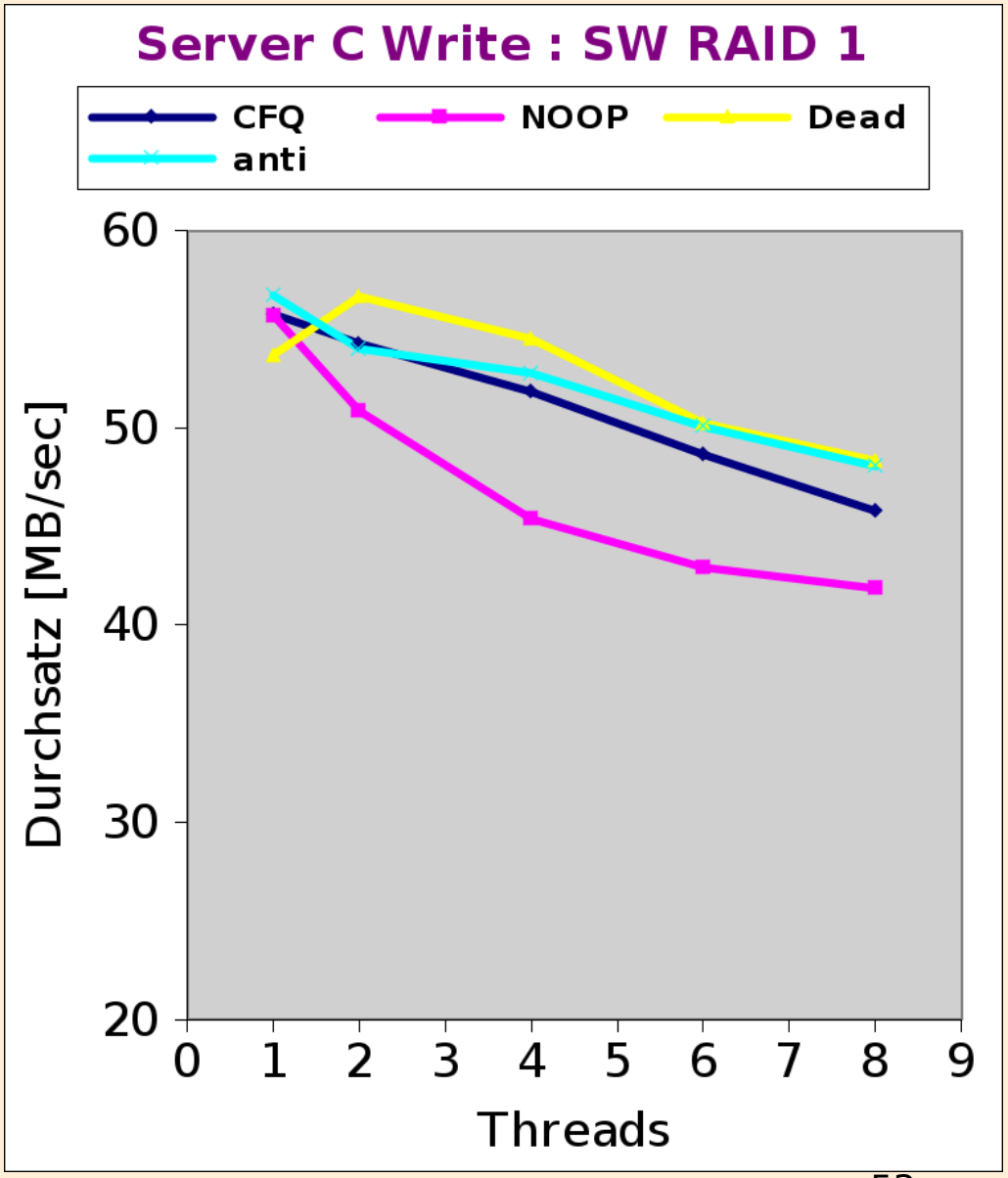
3ware RAID 1 vs. SW RAID 1 : READ



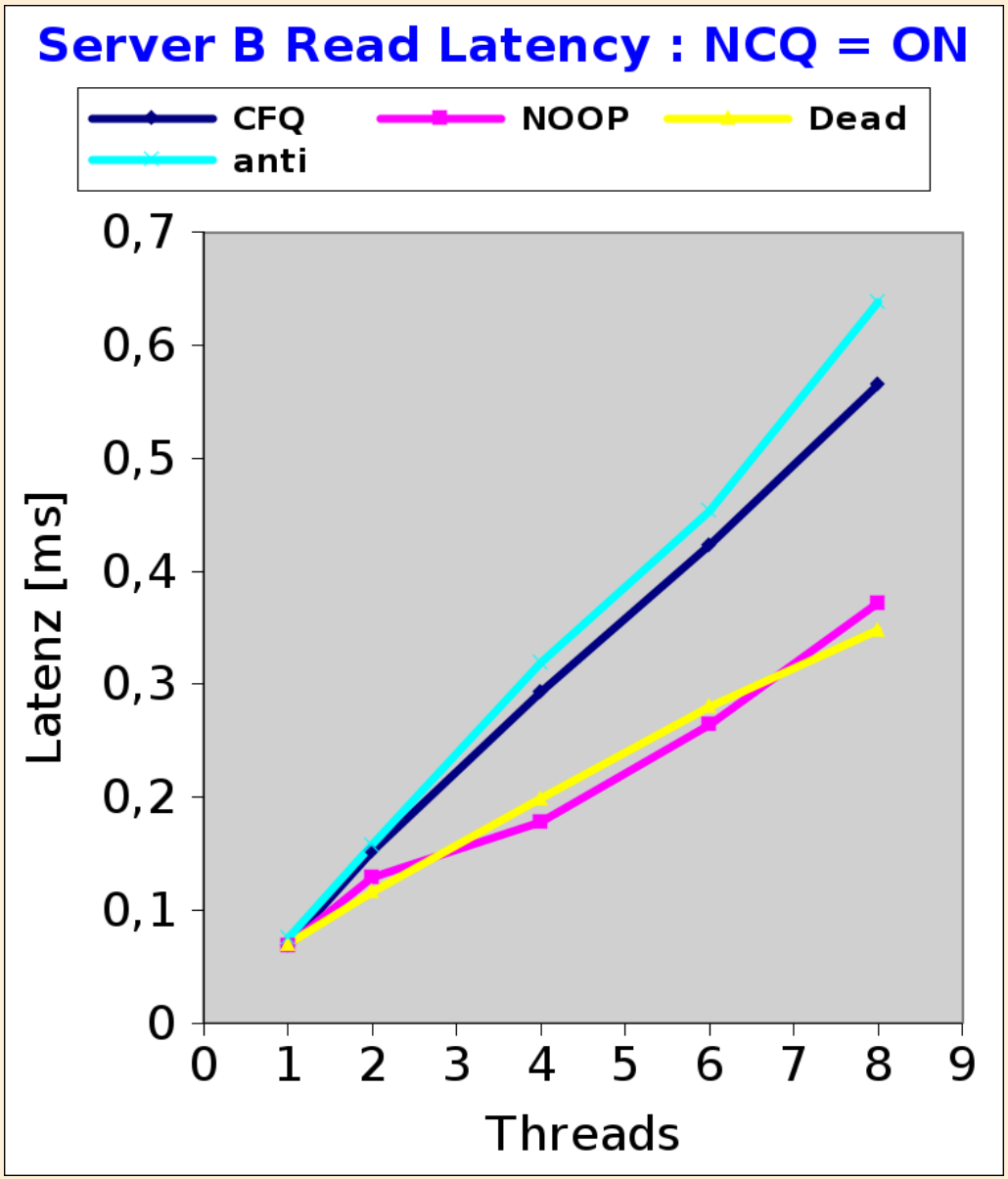


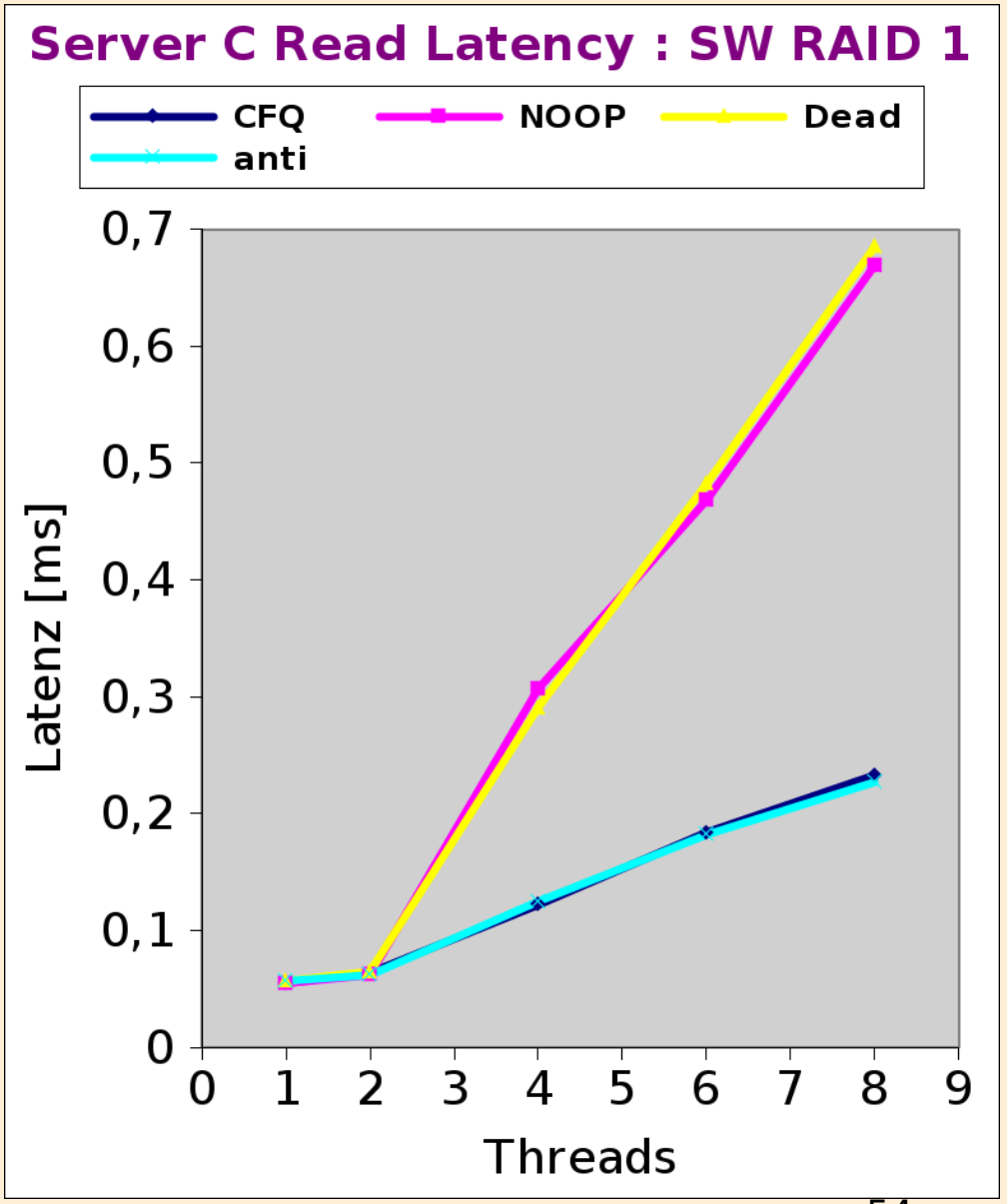
3ware RAID1 vs. SW RAID1: Write Rate



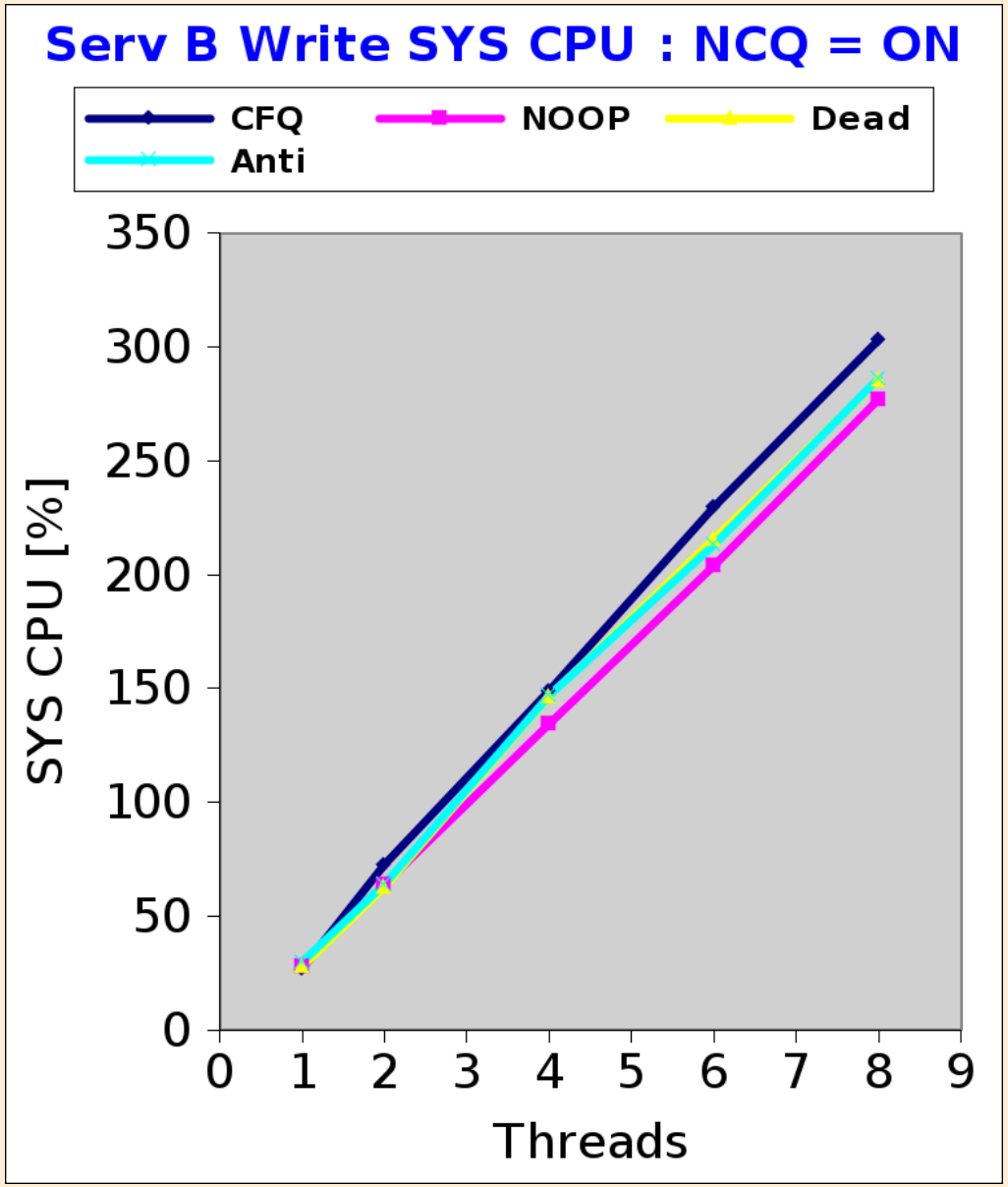


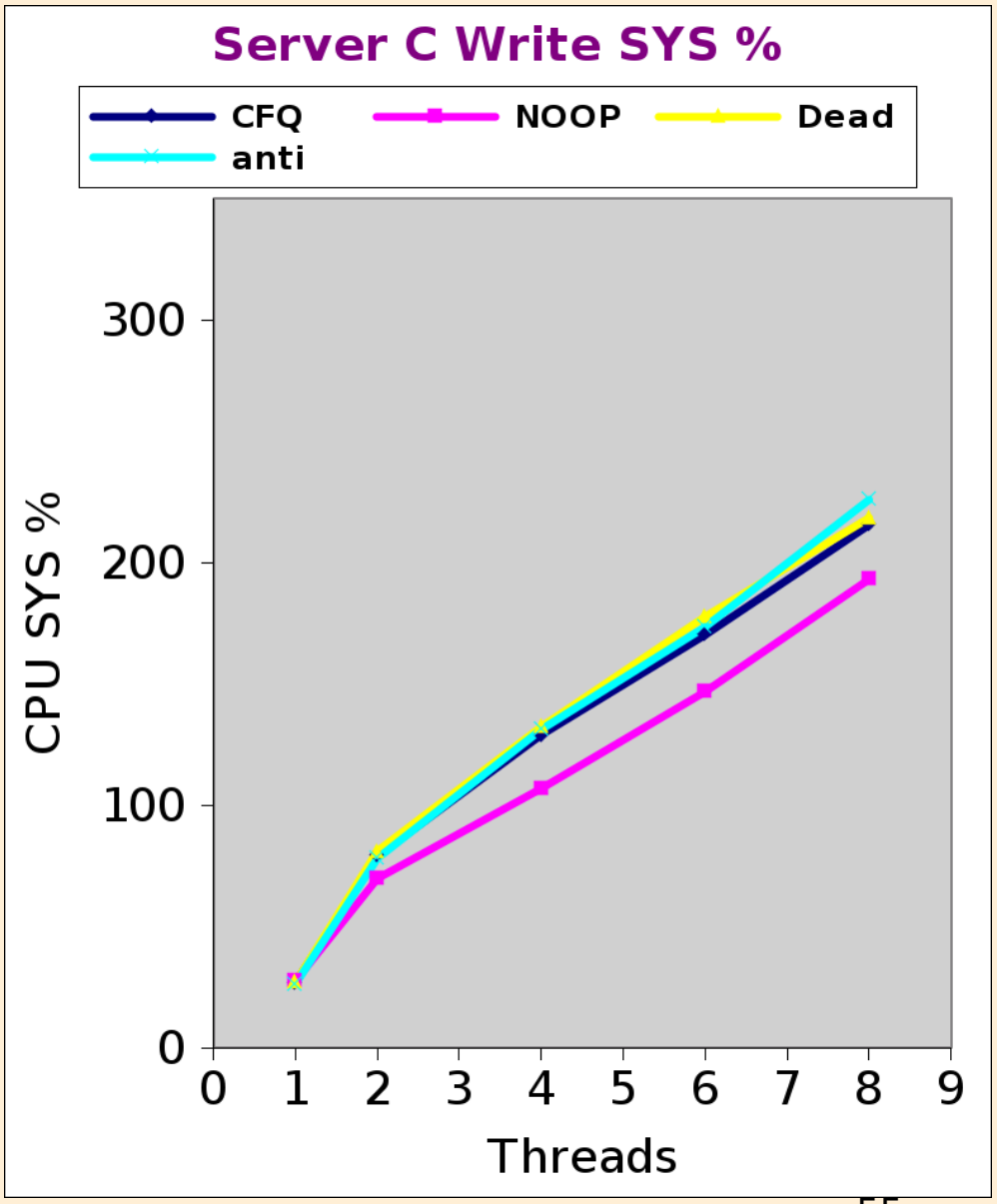
3ware RAID1 vs. SW RAID1: Read Lat.





3ware RAID1 vs. SW RAID1: Write CPU





3ware RAID1 vs. SW RAID1

CFQ

	Rate	Lat.	CPU%
Write	SW	=	SW
Read	SW	SW	3ware
WriteRand	3ware	=	SW
ReadRand	3ware	3ware	3ware

DEADLINE

	Rate	Lat.	CPU%
Write	SW	=	SW
Read	3ware	3ware	SW
Write Rand	3ware	SW	SW
Read Rand	3ware	3ware	3ware

IO-Subsysteme: Fazit

- **CFQ gut bei dummer Hardware oder SW RAID**
- **Deadline gut bei intelligenter Hardware**
- **Anticipatory schlecht bei Random Read?**
- **SW RAID1 und HW RAID1 vergleichbar**
- **Geringere CPU-Auslastung bei SW RAID!**

Linux Prozess Scheduling

- **Was sagt top?**
- **Keine Angst vor Load**
- **Modellierung vs. Load-Messung**
- **Was bringt der neue Prozess Scheduler**

top

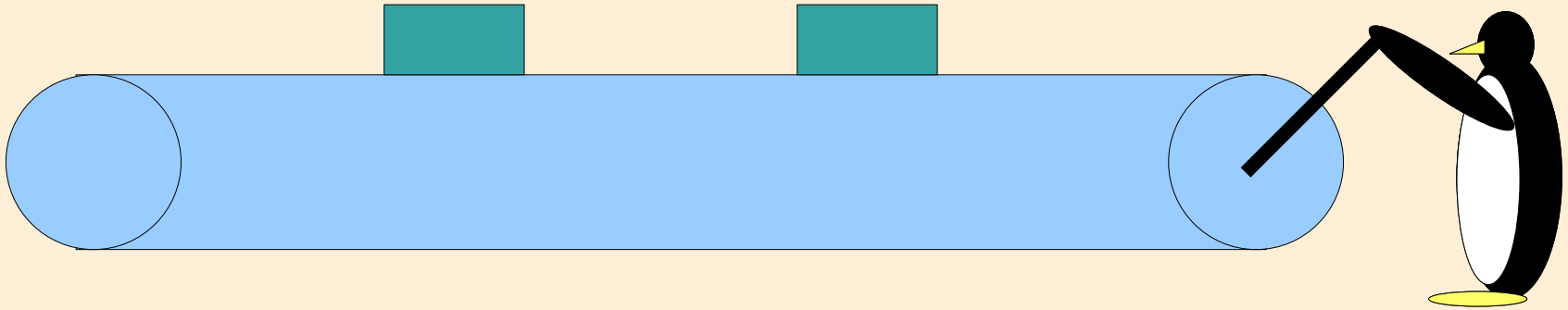
```
top - 06:06:46 up 4 min, 1 user, load average: 108.93, 52.97, 20.15
Tasks: 115 total, 1 running, 114 sleeping,  0 stopped, 5 zombie
Cpu(s): 0.2% us, 0.0% sy, 0.0% ni
      ,99.8% id ,0.0% wa, 0.0% hi, 0.0%si ,8.0%st
Mem:   4055916k total, 451984k used, 3603932k  free,  32068k buffrs
Swap: 15631160k total,      0k used, 15631160k free, 219996k cached

PID USER       PR  NI  VIRT  RES  SHR S %CPU %MEM   TIME+   COMMAND
3451 root      20   0  320m  17m 6420 S    1  0.4   0:07.58 Xorg
3625 volker    20   0  147m  10m 7448 S    1  0.3   0:03.42 artsd
3631 volker    20   0  197m  20m  13m S    1  0.5   0:00.84 konsole
```

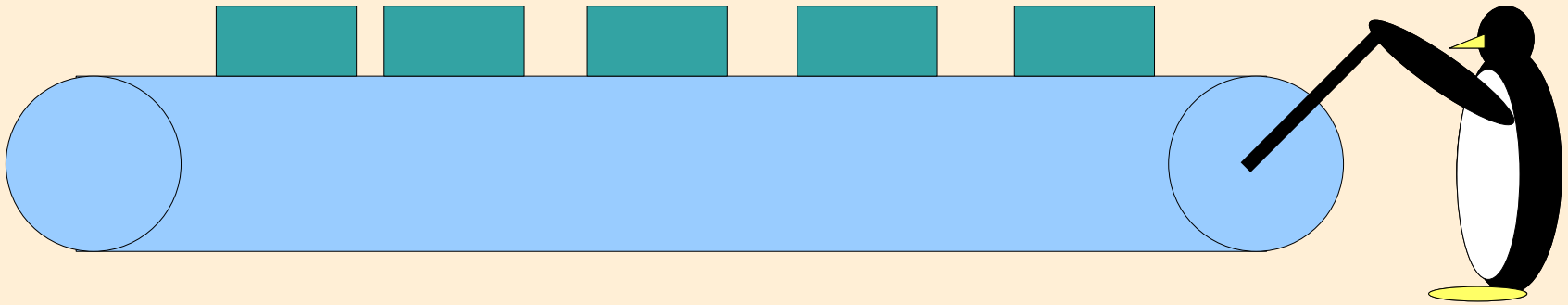
Was ist Load?

Länge der CPU-Warteschlange

- Load 2



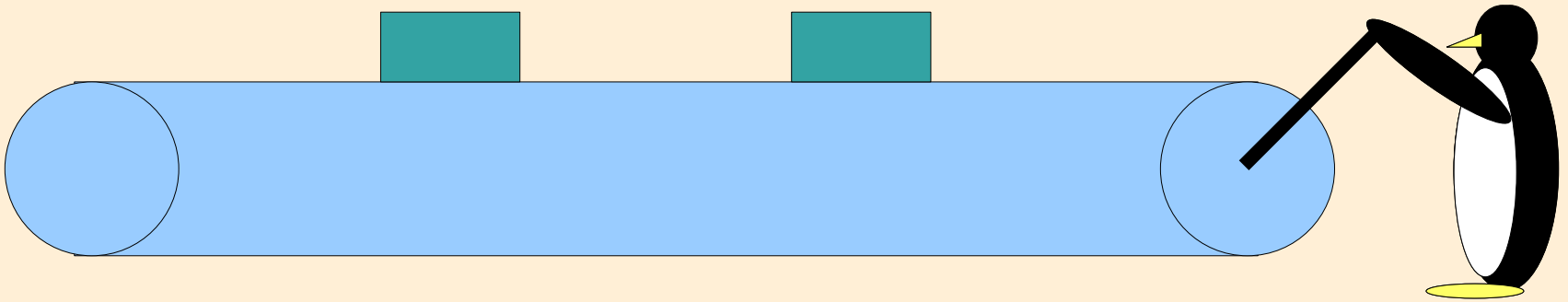
- Load 5



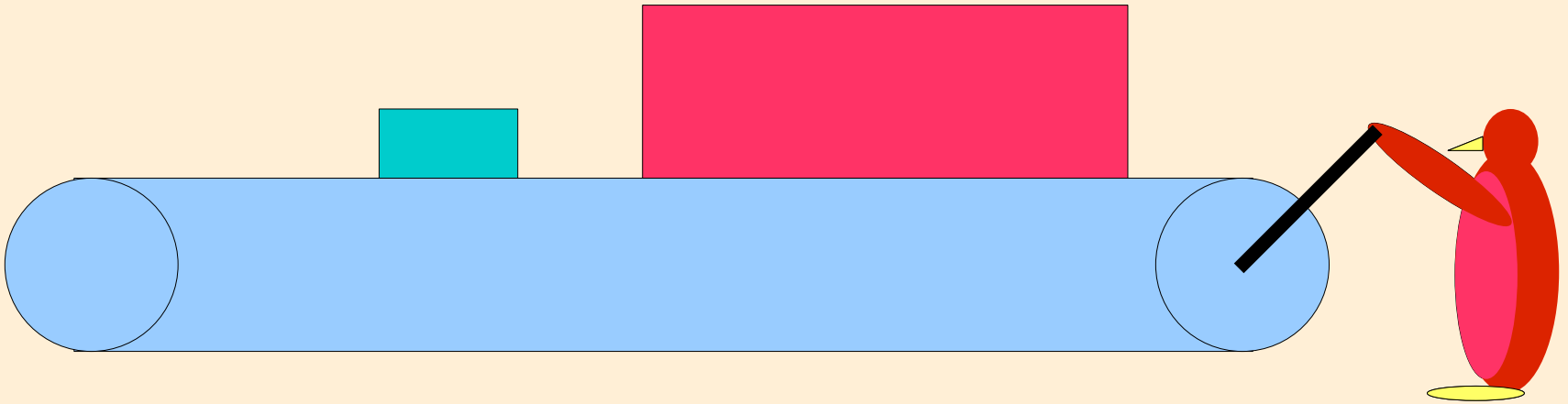
Was ist CPU %

Auslastung der CPU

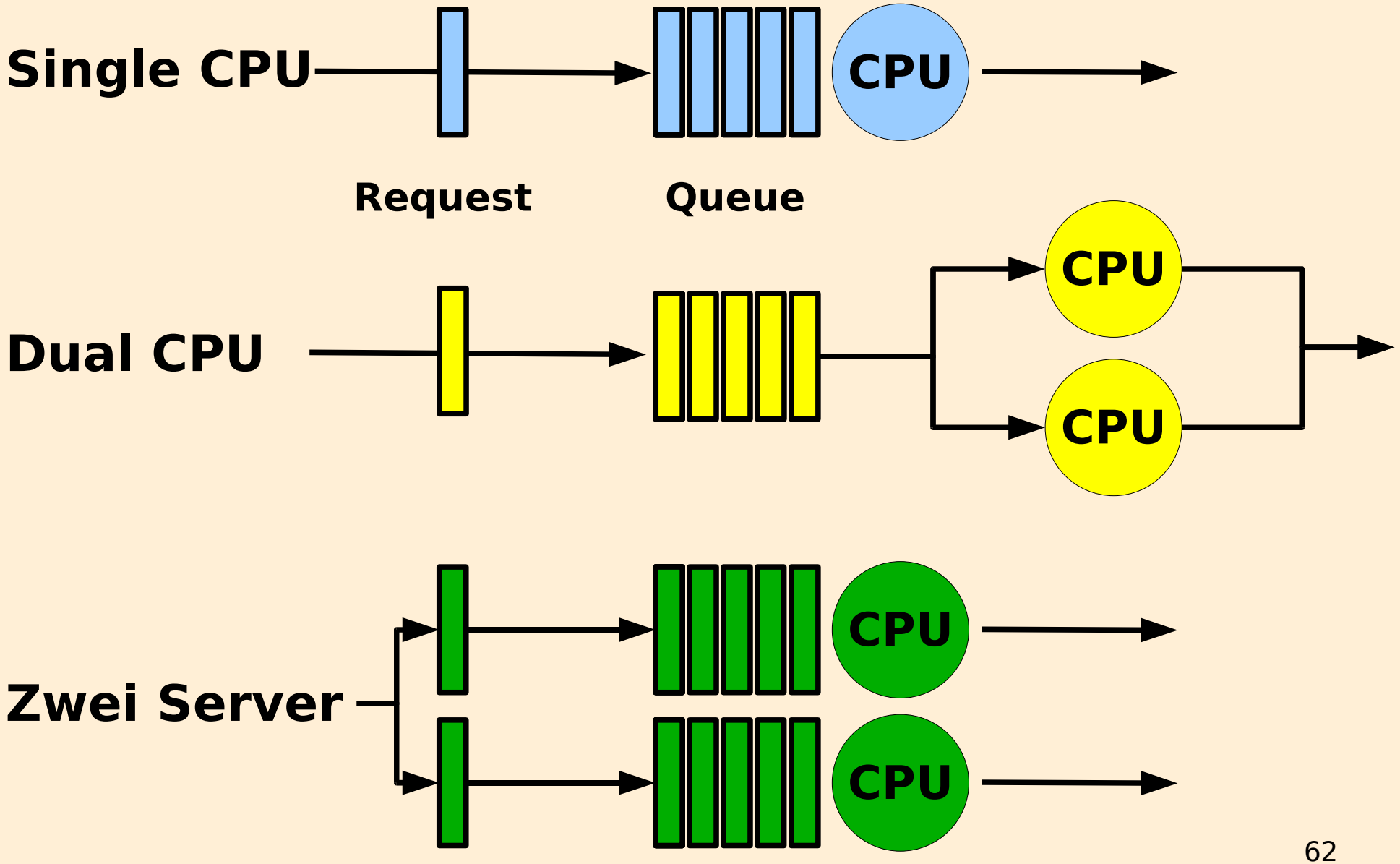
- CPU % =20, Load = 2



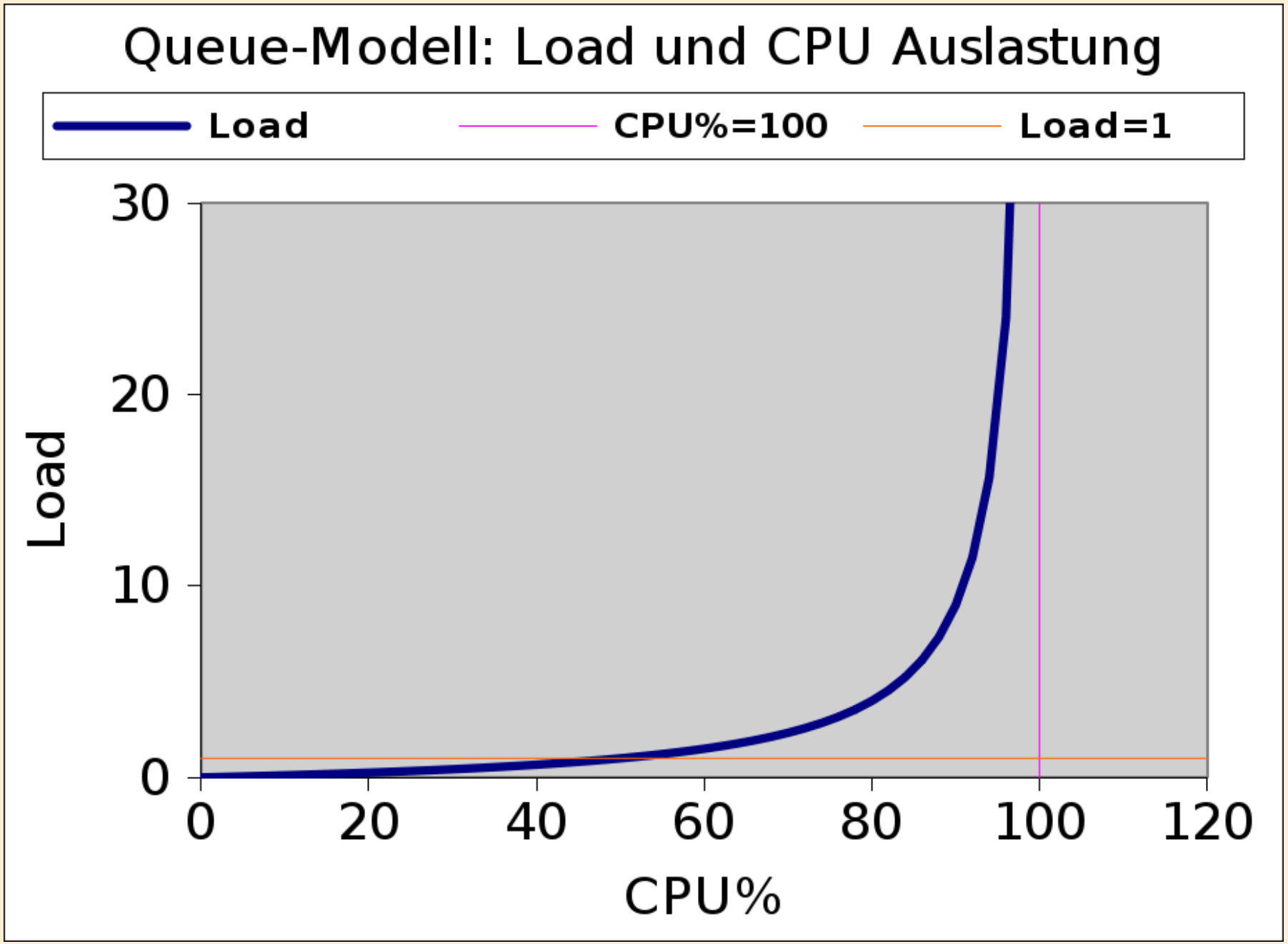
- cpu% = 95, Load = 2



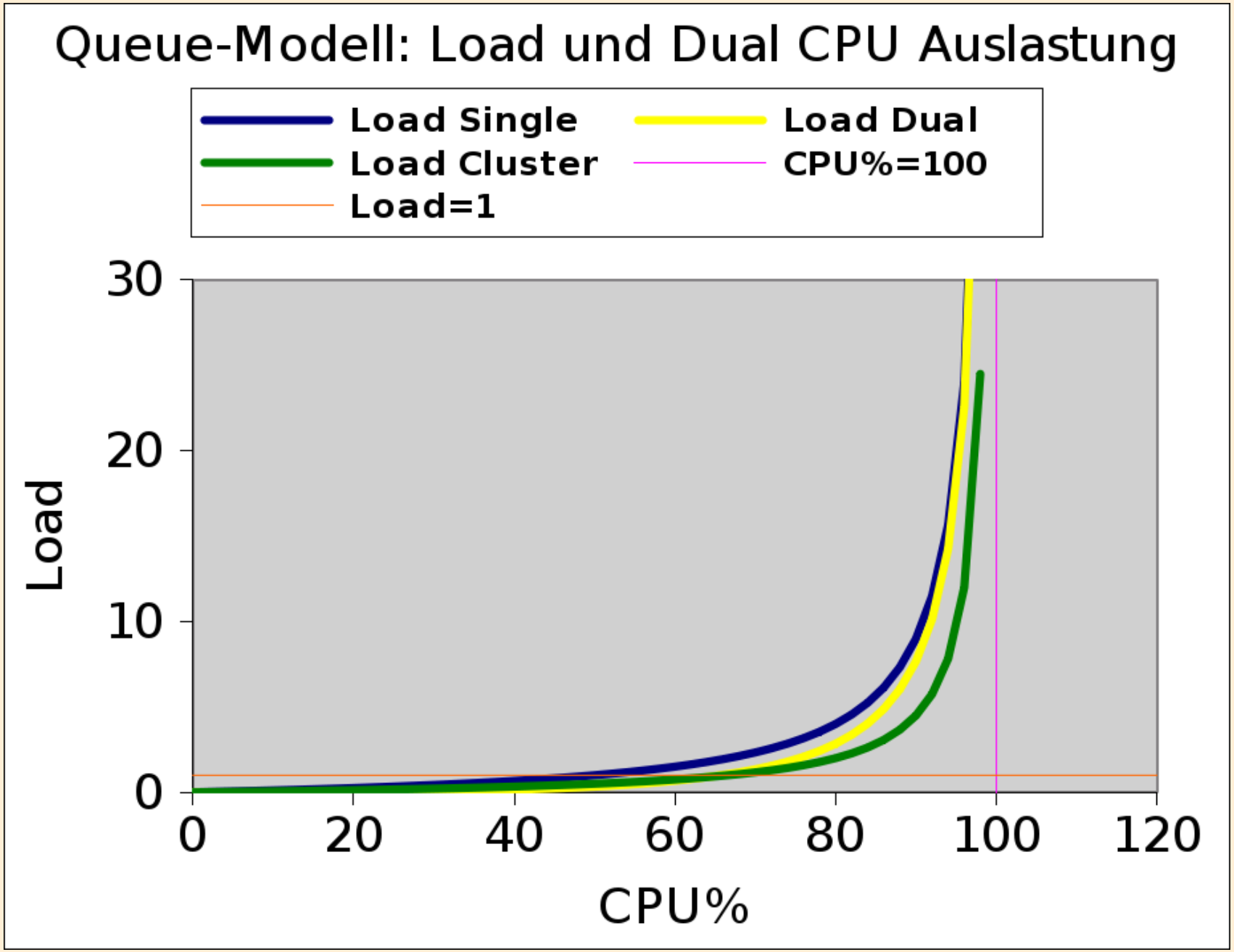
Single-CPU vs Dual-CPU vs Cluster Load



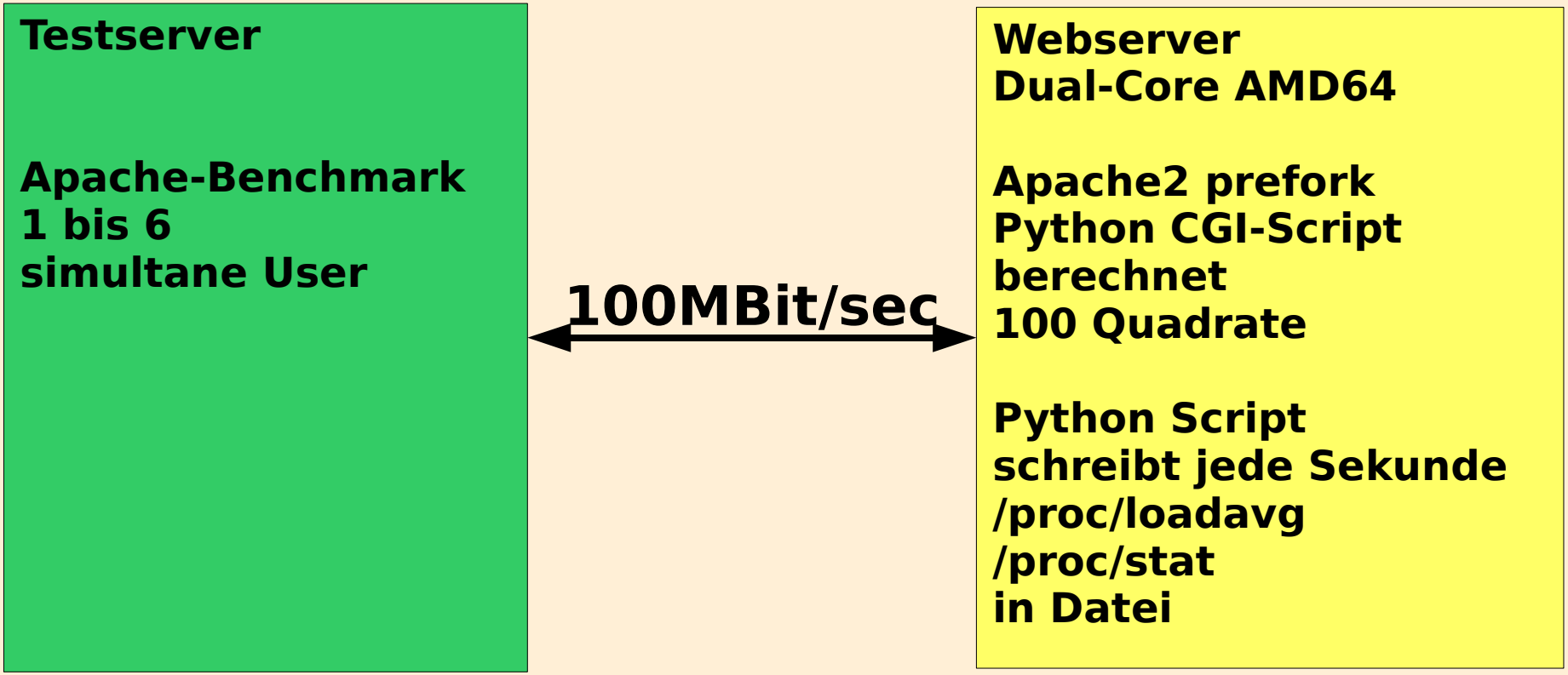
Load und CPU%


Queue-Modell: Load und CPU Auslastung
Load
CPU%=100
Load=1
30
20
10
0
Load
0
20
40
60
80
100
120
CPU%

Load Dual-Core, Cluster

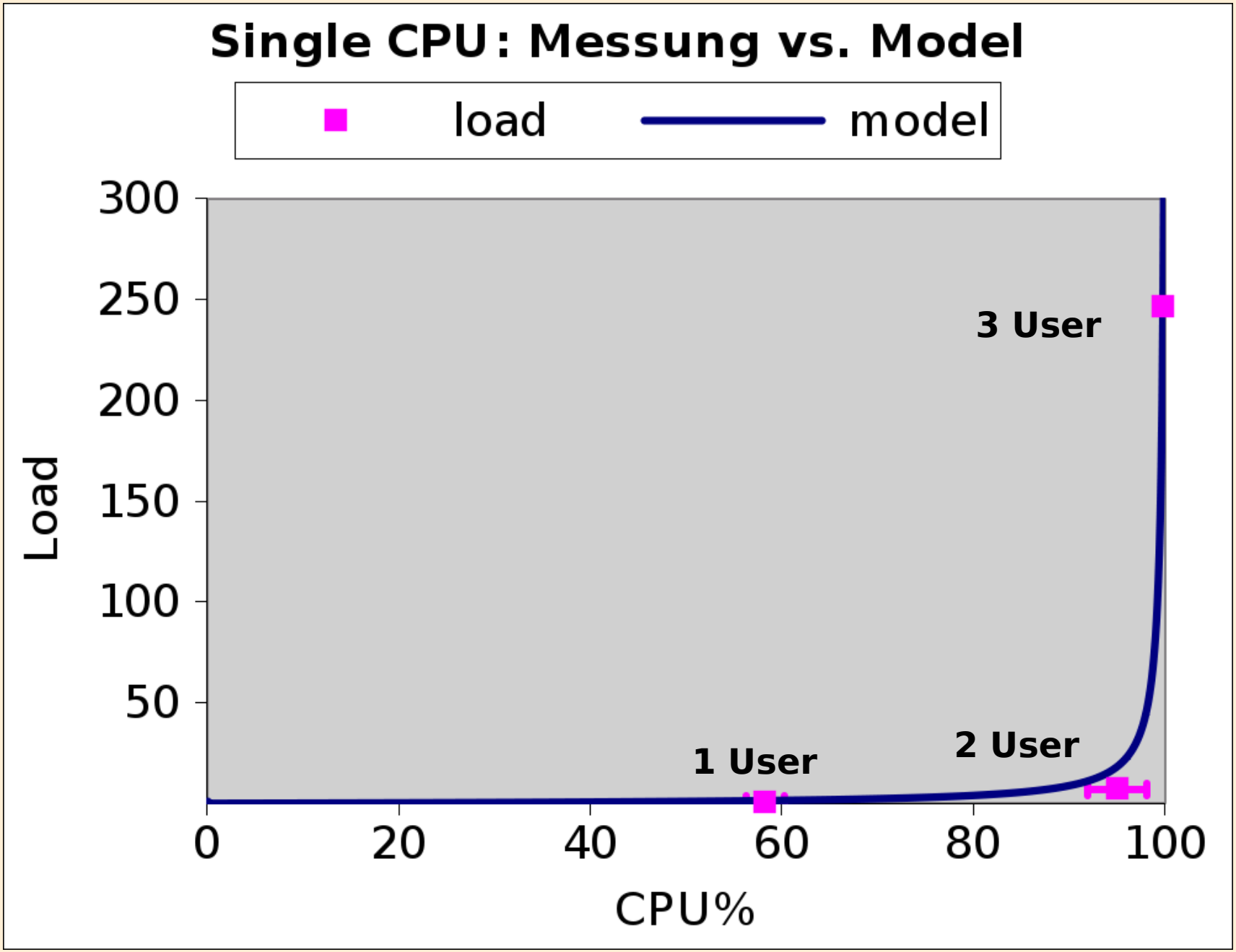


Versuchsaufbau

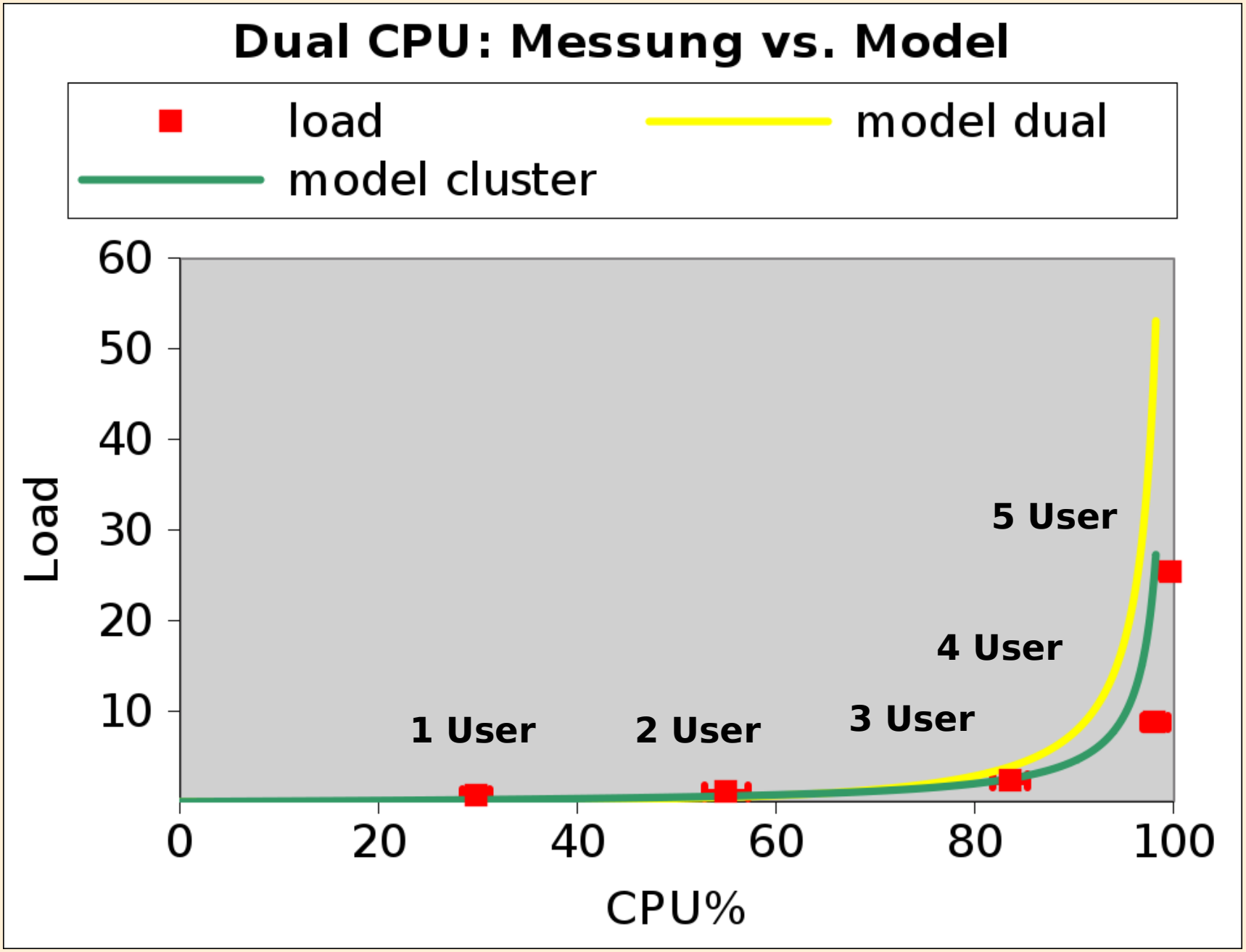


Messung von Load: Einschwingen

Single CPU


Single CPU: Messung vs. Model
load
model
300
250
200
150
100
50
Load
3 User
2 User
1 User
0
20
40
60
80
100
CPU%

Dual-Core CPU

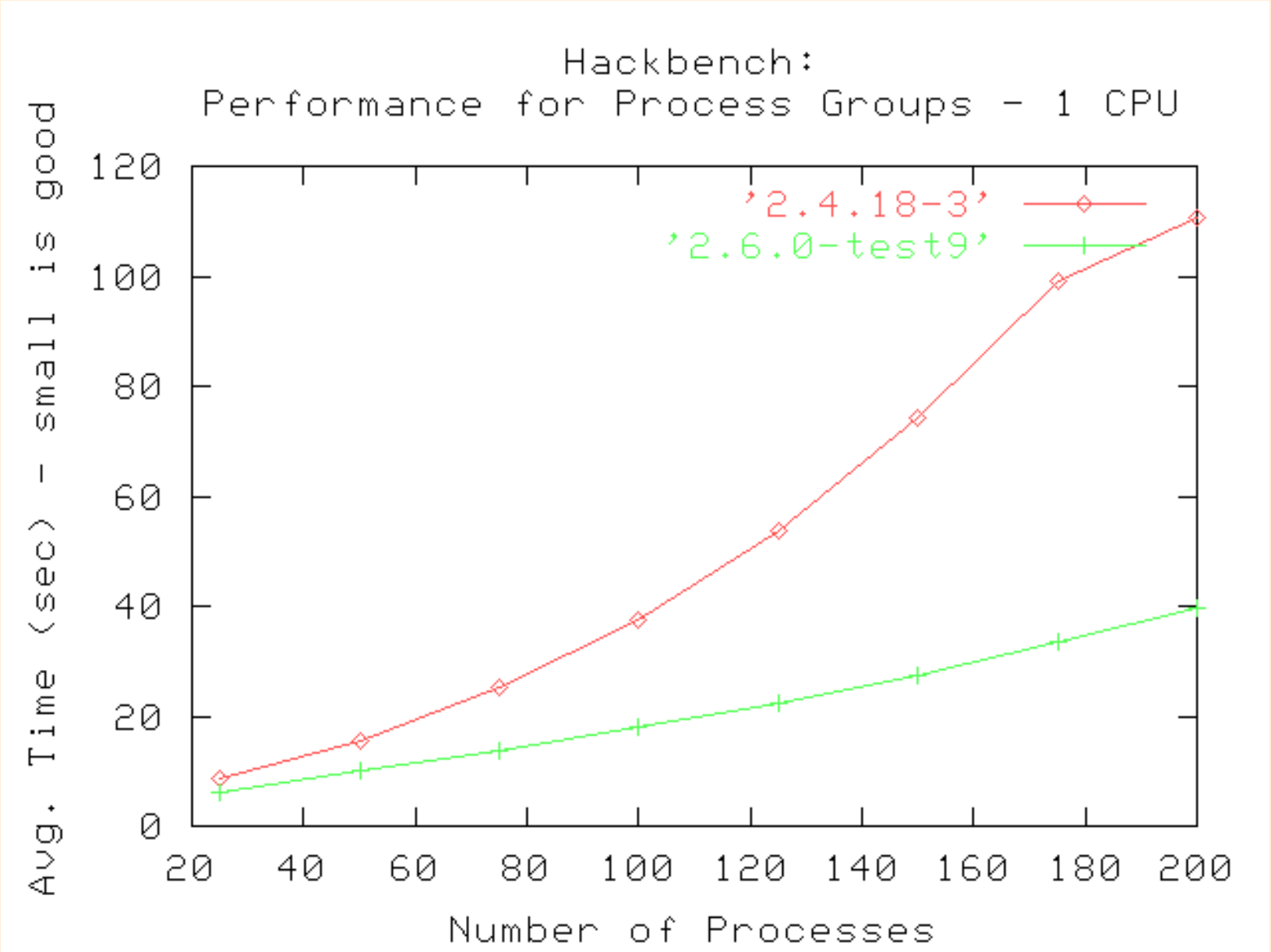


Kernel Scheduler Performance hackbench.c

- Simuliert ein Netzwerk von interagierenden Prozess-Gruppen.
- Verwaltungs-Aufwand des Schedulers:
  - Kernel 2.4: wächst mit Anzahl der Prozesse
  - Kernel > 2.6.0, O(1): konstant viel Heuristik
  - Kernel > 2.6.23, CFQ: konstant viel Wirbel!

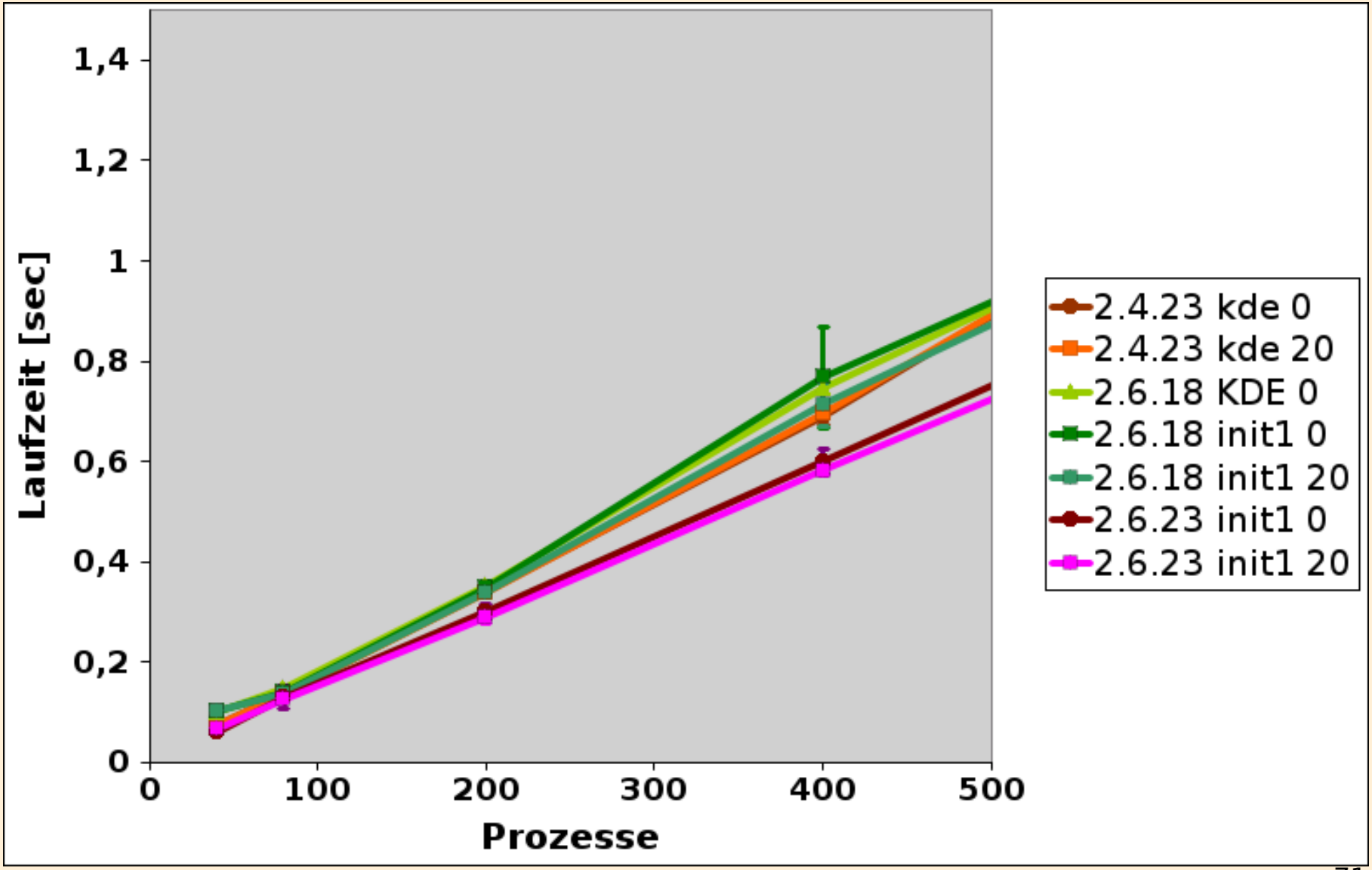
Kernel Sheduler Performance

hackbench.c (Russel, Craiger 2004)



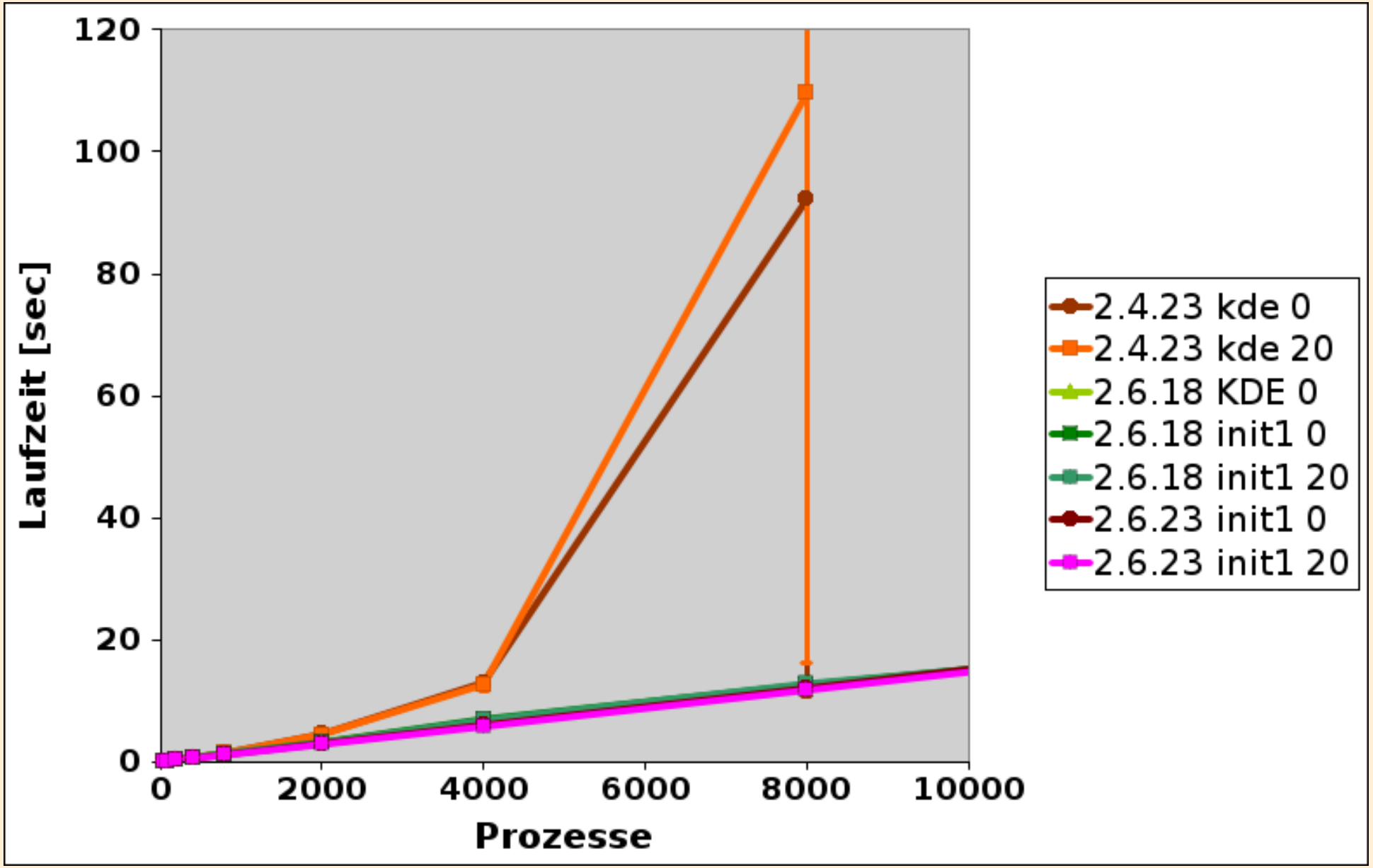
Kernel Scheduler Performance

hackbench.c



Kernel sheduler performance

hackbench.c



Netzwerk TCP Performance

- **Messungen mit iperf**
- **Messungen mit npad**
- **TCP über WAN**

Iperf Gbit Messung

charyptis:~# iperf -c skylla -p5555 -d -m

Server listening on TCP port 5555

TCP window size: 85.3 KByte (default)

Client connecting to 192.168.0.1, TCP port 5555

TCP window size: 237 KByte (default)

[5] local 192.168.0.2 port 41364 connected with 192.168.0.1 port 5555

[4] local 192.168.0.2 port 5555 connected with 192.168.0.1 port 43630

[5] 0.0-10.0 sec 1.01 GBytes 870 Mbits/sec

[5] MSS size 1448 bytes (MTU 1500 bytes, ethernet)

[4] 0.0-10.0 sec 1.01 GBytes 863 Mbits/sec

[4] MSS size 1448 bytes (MTU 1500 bytes, ethernet)

Iperf Gbit Messung II

mail1:~# iperf -c mail2 -p5555 -d -m

[5] 0.0-10.0 sec 201 MBytes 169 Mbits/sec

[5] MSS size 1448 bytes (MTU 1500 bytes, ethernet)

[4] 0.0-10.1 sec 318 MBytes 265 Mbits/sec

[4] MSS size 1448 bytes (MTU 1500 bytes, ethernet)

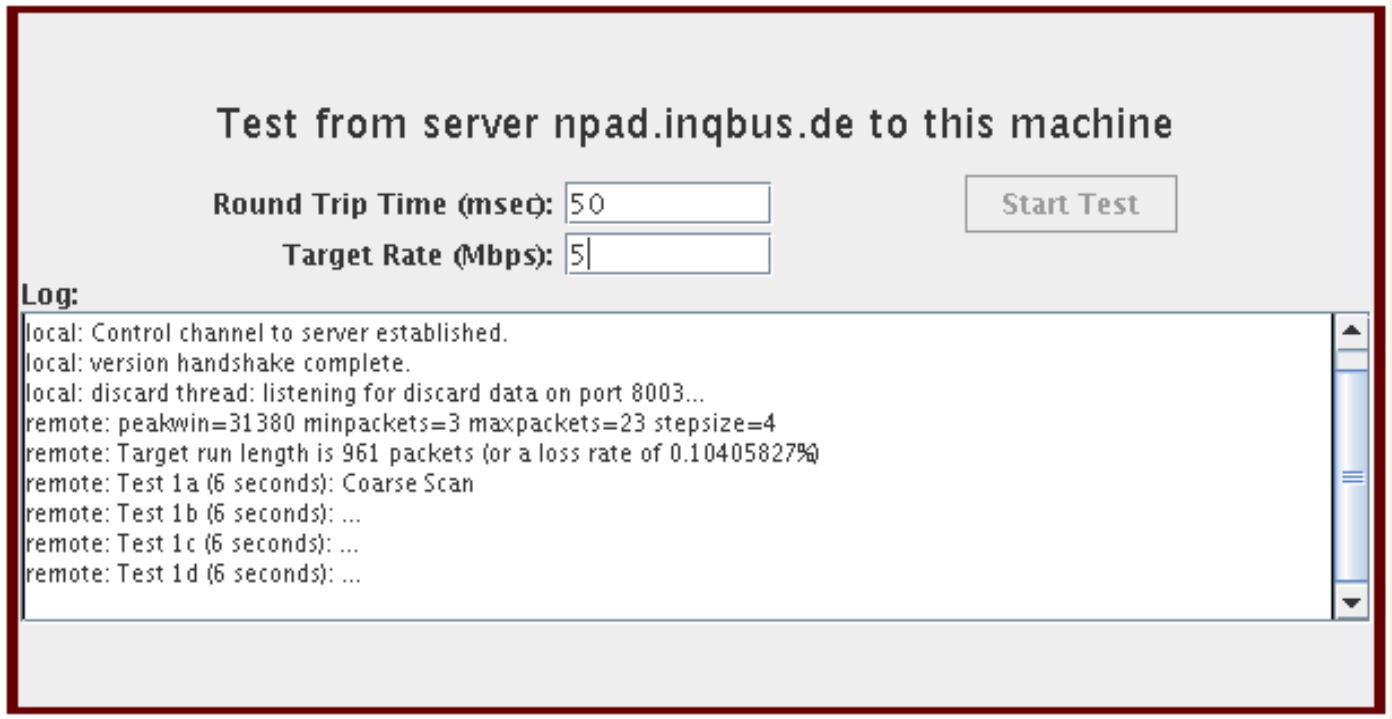
Ursache: Billige 32Bit-Gbit-Karte bzw. Anbindung von Gbit-fähigen Onboard Chips an 32Bit PCI-Bus. Blockzeichnung der Boards anschauen!

NPAD

One-click network diagnostics

Server located at Inqbus in Leipzig, Germany

Documentation | Results Summary | About NPAD



Command line client

```
cc diag-client.c -o diag-client
```

Run it:

./diag-client <server_name> <port> <target_RTT> <target_data_rate>

Target host TCP configuration test: Pass! [?]

TCP negotiated appropriate options: WSCALE=7, SACKok, and Timestamps. [?]
The target passed all tests! See tester caveats: [?]

Path measurements [?]

Data rate test: Fail! [?]

The maximum data rate was 77.909309 Mb/s. [?]
This is below the target rate (90.000000 Mb/s). [?]
This test did not complete due to other problems with the path, target or tester.
> Correct other problems first, and then rerun this test. [?]

Loss rate test: Pass! [?]

Pass: zero losses in 107043 packets, loss rate less than 0.000934%. [?]
FYI: To get 90 Mb/s with a 1448 byte MSS on a 3 ms path the total end-to-end loss budget is 0.090253% (1108 packets between losses). [?]

Suggestions for alternate tests

FYI: This path may pass with a less strenuous application: [?]
Try rate=77 Mb/s, rtt=34 ms
Or if you can raise the MTU: [?]
Try rate=77 Mb/s, rtt=211 ms, mtu=9000 bytes

Network buffering test: Pass! [?]

This test did not complete due to other problems with the path, target or tester.
> Correct other problems first, and then rerun this test. [?]
Pass: The network bottleneck has sufficient buffering (queue space) in routers and switches. [?]
Estimated queue size is at least: Pkts: 156 Bytes: 225888
This is probably an underestimate of the actual queue size. [?]
This corresponds to a 23.210388 ms drain time. [?]
To get 90 Mb/s with on a 3 ms path, you need 33750 bytes of buffer space. [?]

> Localize all path problems by testing progressively smaller sections of the full path. [?]

TCP übers WAN : RZ L.E. -> RZ RO

test:~# iperf -s -p5555
Server listening on TCP port 5555
TCP window size: 85.3 KByte (default)
[4] 0.0-10.5 sec 9.00 MBytes 7.21 Mbits/sec
[5] 0.0-10.3 sec 9.47 MBytes 7.67 Mbits/sec
[4] 0.0-10.3 sec 11.0 MBytes 8.98 Mbits/sec

Maximale TCP Transferrate

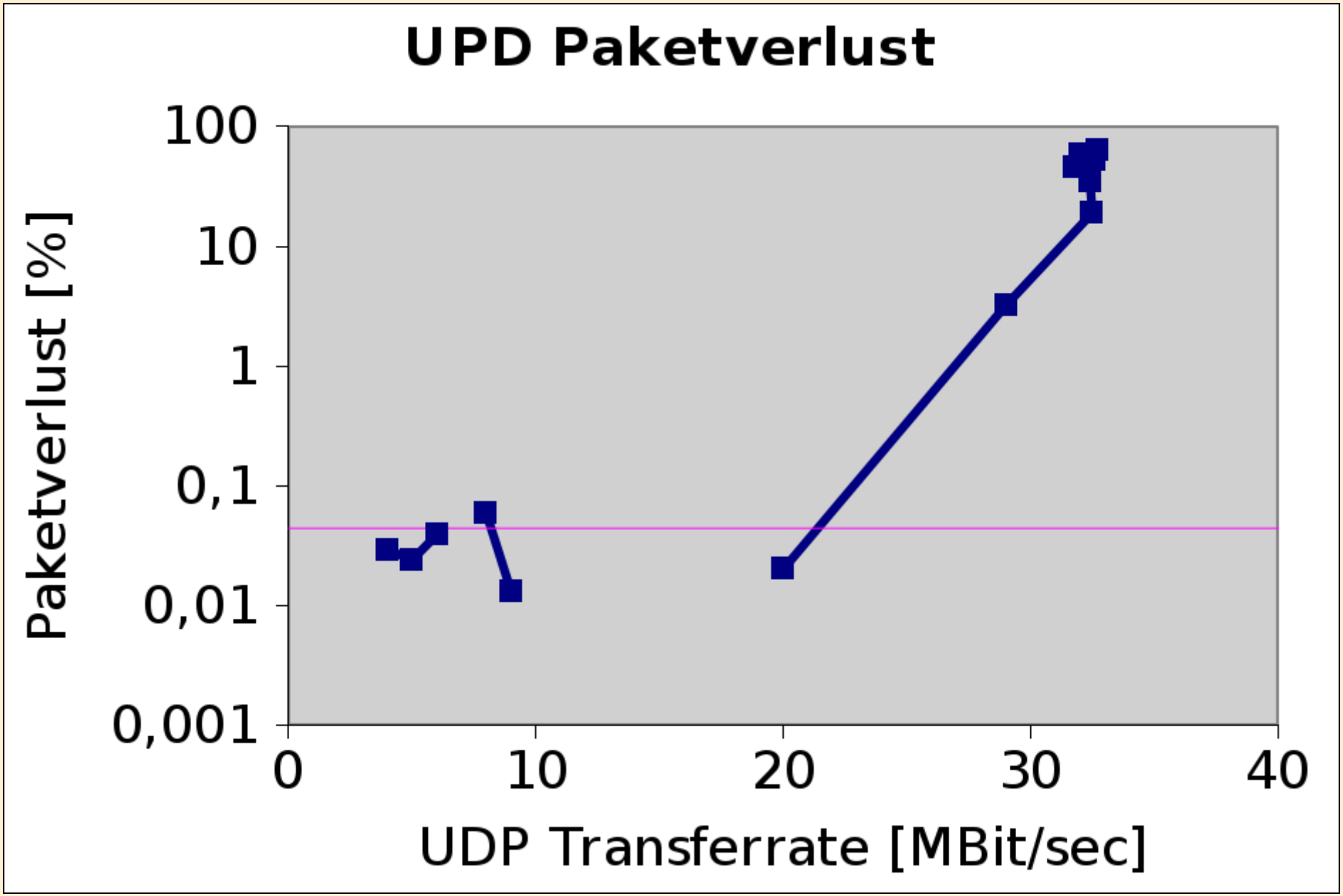
$$R_{max} = \frac{MSS}{RTT} \times \frac{1}{\sqrt{Loss}}$$

$$Loss_{max} = \left(\frac{MSS}{RTT} \times \frac{1}{R} \right)^2$$

MSS=1460 Byte
RTT=61 ms
R=9Mbit/sec

$$Loss_{max} = 0{,}044\%$$

Lossrate bestimmen mit iperf UDP



path_char

```
debian-test01:~# ./pathchar 1.2.3.4

 0 localhost
 |    86 Mb/s,   325 us (790 us)
 1 3.4.5.4 (3.4.5.4)
 |    34 Mb/s,   302 us (1.75 ms)
 2 44.54.67.67 (44.54.67.67)
 |    112 Mb/s,   3.86 ms (10.5 ms)
 3 34.43.34.43 (34.43.34.43)
 |   ?? b/s,   1.67 ms (13.7 ms)
 4 rb1-g0-2.tau.envia-tel.net (83.221.252.242)
 |   124 Mb/s,   6.14 ms (26.1 ms),  +q 1.09 ms (16.9 KB)
 5 rmws-frnk-de07.nw.telefonica.de (80.81.192.89)
 |   ?? b/s,   82 us (26.2 ms)
 6 rmwc-frnk-de01-ge-2-1-0-0.nw.mediaways.net
(213.20.249.201)
```

Danke!

- Patrick Westphal und Martin Klapproth für Daten-Erhebung und -Auswertung
- Elfrun Jaenisch: Für geduldiges Warten bis Papa mal wieder Zeit für sie hat.
- Der Firma DELL für die angeregte Diskussion nach dem Vortrag.

Thus spake the master programmer:
It is time for you to leave. (TAOP)