Hier geht nix rein!

Storage-Performance im Virtualisierungsumfeld

Michael Ziegler

it-novum GmbH

15. März 2014

/me

Michael Ziegler

- openATTIC-Entwickler seit 2010
- Ansprechpartner bei Performance-Fragen
  - Storage
  - Virtualisierung

Disclaimer

Keine Optimierung wird einfach so funktionieren.

Größtes Hindernis: Irrelevante Messungen.

Niemand sonst kennt euren Workload.

Messt selbst.

dd

```
dd if=/dev/zero of=/dev/md1 bs=256k count=131072
```

dd = Durchsatz

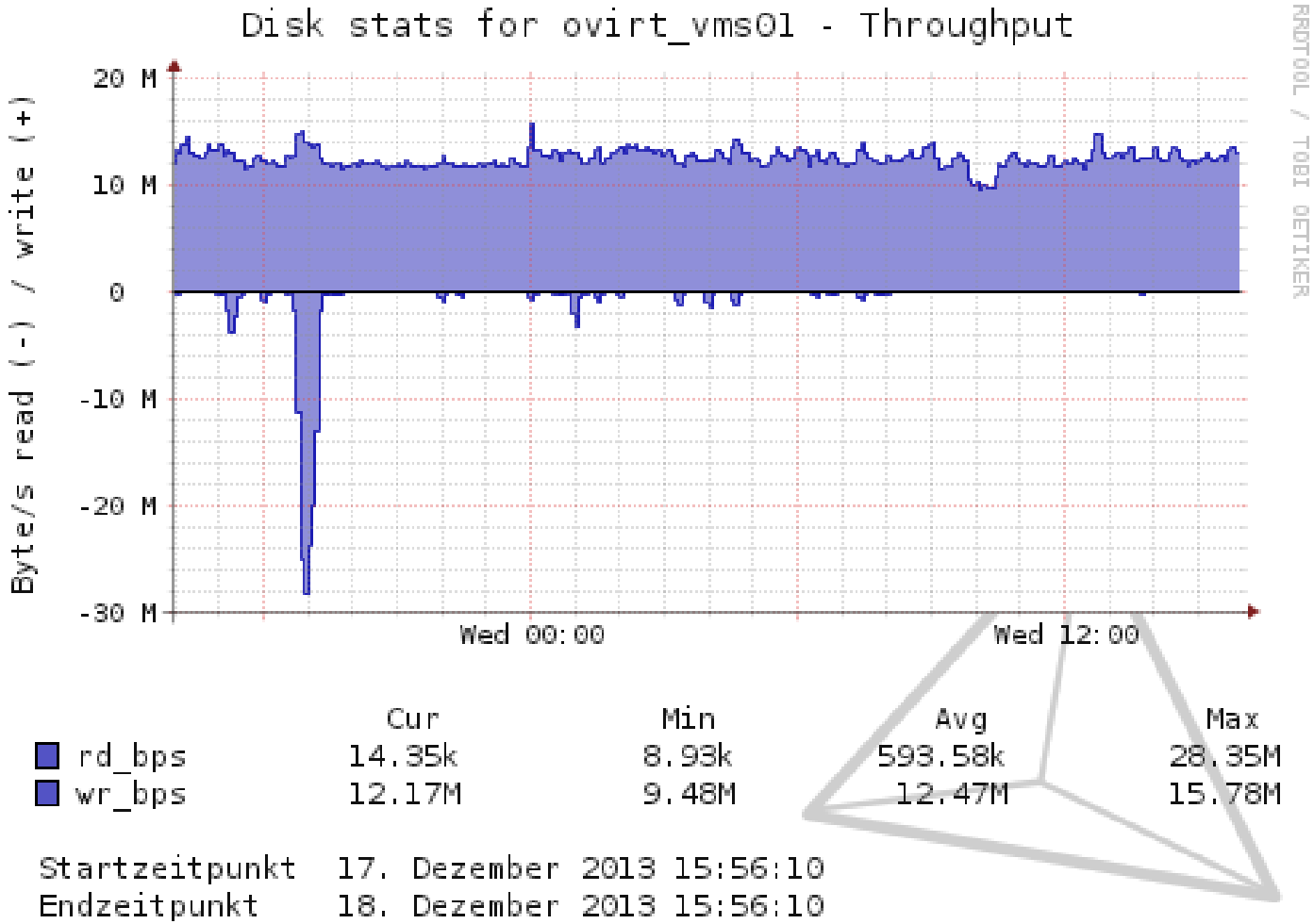
- RAID:
  dd if=/dev/zero of=/dev/md1 bs=256k count=131072
  ...
  34359738368 Bytes (34 GB) kopiert, 42,8475 s, 802 MB/s

- ZFS (raidz3):
  dd if=/dev/zero of=/tank/test.img bs=256k count=131072
  ...
  34359738368 Bytes (34 GB) kopiert, 41,9683 s, 819 MB/s

dd = Durchsatz

Was ist denn realistisch?

dd = Durchsatz



dd = Durchsatz

- dd = ein Prozess schreibt sequenziell mit großer Blocksize
- VMs = viele Prozesse schreiben random mit kleiner Blocksize
- thin-provisioned Images:

```
$ xfs_db -c frag -r /dev/drbd/by-res/ovirt_vms01
[...] fragmentation factor 90.29%
$ xfs_db -c frag -r /dev/drbd/by-res/ovirt_vms02
[...] fragmentation factor 99.68%
```

dd = Durchsatz = irrelevant

Quintessenz: sequenziell = irrelevant.

postmark

Benchmark that's based around small file operations similar to those used on large mail servers and news servers. Has been ported to NT so should be good for comparing OSs.

"Postmark: a new file system benchmark", Jeffrey Katcher, Network Appliance Technical Report TR-3022, October 1997

postmark

- Volume = mit XFS formatiertes LV auf RAID
- 4 postmark-Prozesse parallel

postmark

Ausgabe:
150087 created (1035 per second)
    Creation alone: 100000 files (7692 per second)
    Mixed with transactions: 50087 files (417 per second)
49995 read (416 per second)
49991 appended (416 per second)
150087 deleted (1035 per second)
    Deletion alone: 100174 files (8347 per second)
    Mixed with transactions: 49913 files (415 per second)

postmark

Interessantere Statistik:

```
iostat -xdm 1 /dev/vgbenrime/postmarkxfs
```

postmark

```
 Device:  rrqm/s wrqm/s r/s w/s rMB/s wMB/s avgrq-sz
avgqu-sz await r_await w_await svctm %util
dm-6 0,00 0,00 0,00 9934,00 0,00 73,07 15,06 226,03
32,47 0,00 32,47 0,10 100,00
dm-6 0,00 0,00 0,00 5901,00 0,00 43,93 15,25 185,99
29,96 0,00 29,96 0,17 100,00
dm-6 0,00 0,00 0,00 4173,00 0,00 29,92 14,68 200,41
46,62 0,00 46,62 0,24 100,00
dm-6 0,00 0,00 0,00 8367,00 0,00 63,14 15,46 200,71
28,35 0,00 28,35 0,12 100,00
dm-6 0,00 0,00 0,00 3388,00 0,00 24,45 14,78 230,90
101,86 0,00 101,86 0,30 100,00
dm-6 0,00 0,00 2,00 9869,00 0,02 72,44 15,03 223,62
28,00 82,00 27,99 0,10 100,00
dm-6 0,00 0,00 2,00 1762,00 0,01 13,55 15,75 168,21
110,37 0,00 110,50 0,57 100,00
dm-6 0,00 0,00 1,00 1772,00 0,00 14,47 16,72 183,03
```

postmark

w/s	wMB/s	avgrq-sz	avgqu-sz	await	%util
9934,00	73,07	15,06	226,03	32,47	100,00
5901,00	43,93	15,25	185,99	29,96	100,00
4173,00	29,92	14,68	200,41	46,62	100,00
8367,00	63,14	15,46	200,71	28,35	100,00
3388,00	24,45	14,78	230,90	101,86	100,00
1772,00	14,47	16,72	183,03	89,59	100,00
0,00	0,00	0,00	188,00	0,00	100,00
5868,00	43,90	15,32	179,32	28,78	100,00

postmark

w/s	wMB/s	avgrq-sz	avgqu-sz	await	%util
9934,00	73,07	15,06	226,03	32,47	100,00
5901,00	43,93	15,25	185,99	29,96	100,00
4173,00	29,92	14,68	200,41	46,62	100,00
8367,00	63,14	15,46	200,71	28,35	100,00
3388,00	24,45	14,78	230,90	101,86	100,00
1772,00	14,47	16,72	183,03	89,59	100,00
0,00	0,00	0,00	188,00	0,00	100,00
5868,00	43,90	15,32	179,32	28,78	100,00

- `w/s` = Operationen pro Sekunde
- Sozusagen “Arbeit pro Zeit”

postmark

w/s	wMB/s	avgrq-sz	avgqu-sz	await	%util
9934,00	73,07	15,06	226,03	32,47	100,00
5901,00	43,93	15,25	185,99	29,96	100,00
4173,00	29,92	14,68	200,41	46,62	100,00
8367,00	63,14	15,46	200,71	28,35	100,00
3388,00	24,45	14,78	230,90	101,86	100,00
1772,00	14,47	16,72	183,03	89,59	100,00
0,00	0,00	0,00	188,00	0,00	100,00
5868,00	43,90	15,32	179,32	28,78	100,00

- `wMB/s` = Durchsatz
- Erinnerung: dd erreicht 800MB/s

postmark

w/s	wMB/s	avgrq-sz	avgqu-sz	await	%util
9934,00	73,07	15,06	226,03	32,47	100,00
5901,00	43,93	15,25	185,99	29,96	100,00
4173,00	29,92	14,68	200,41	46,62	100,00
8367,00	63,14	15,46	200,71	28,35	100,00
3388,00	24,45	14,78	230,90	101,86	100,00
1772,00	14,47	16,72	183,03	89,59	100,00
0,00	0,00	0,00	188,00	0,00	100,00
5868,00	43,90	15,32	179,32	28,78	100,00

- `avgrq-sz` = Request-Größe in Sektoren (512B = $\frac{1}{2}$KiB)
- Groß bei sequenziellem IO
- Klein bei random-IO

postmark

w/s	wMB/s	avgrq-sz	avgqu-sz	await	%util
9934,00	73,07	15,06	226,03	32,47	100,00
5901,00	43,93	15,25	185,99	29,96	100,00
4173,00	29,92	14,68	200,41	46,62	100,00
8367,00	63,14	15,46	200,71	28,35	100,00
3388,00	24,45	14,78	230,90	101,86	100,00
1772,00	14,47	16,72	183,03	89,59	100,00
0,00	0,00	0,00	188,00	0,00	100,00
5868,00	43,90	15,32	179,32	28,78	100,00

- `avgqu-sz` = Queue-Länge
- Viel bei sequenziellem IO = OK
- Viel bei random-IO = Storage ist am Limit

postmark

w/s	wMB/s	avgrq-sz	avgqu-sz	await	%util
9934,00	73,07	15,06	226,03	32,47	100,00
5901,00	43,93	15,25	185,99	29,96	100,00
4173,00	29,92	14,68	200,41	46,62	100,00
8367,00	63,14	15,46	200,71	28,35	100,00
3388,00	24,45	14,78	230,90	101,86	100,00
1772,00	14,47	16,72	183,03	89,59	100,00
0,00	0,00	0,00	188,00	0,00	100,00
5868,00	43,90	15,32	179,32	28,78	100,00

- `await` = Latenz in Millisekunden
- Zeit pro Arbeitsschritt

postmark

w/s	wMB/s	avgrq-sz	avgqu-sz	await	%util
9934,00	73,07	15,06	226,03	32,47	100,00
5901,00	43,93	15,25	185,99	29,96	100,00
4173,00	29,92	14,68	200,41	46,62	100,00
8367,00	63,14	15,46	200,71	28,35	100,00
3388,00	24,45	14,78	230,90	101,86	100,00
1772,00	14,47	16,72	183,03	89,59	100,00
0,00	0,00	0,00	188,00	0,00	100,00
5868,00	43,90	15,32	179,32	28,78	100,00

- `%util` = Auslastung in Prozent

postmark

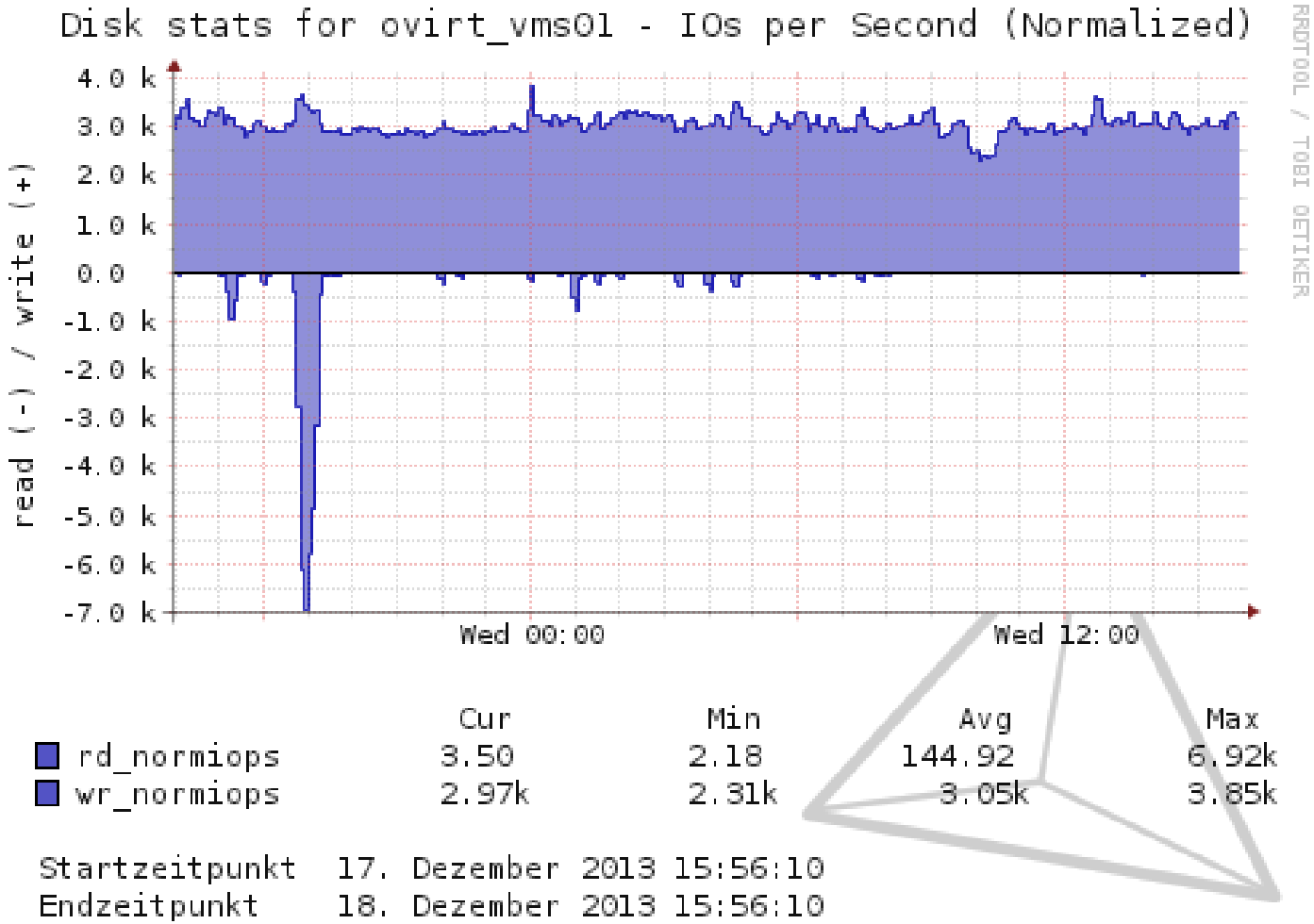
w/s	wMB/s	avgrq-sz	avggu-sz	await	%util
9934,00	73,07	15,06	226,03	32,47	100,00
5901,00	43,93	15,25	185,99	29,96	100,00
4173,00	29,92	14,68	200,41	46,62	100,00
8367,00	63,14	15,46	200,71	28,35	100,00
3388,00	24,45	14,78	230,90	101,86	100,00
1772,00	14,47	16,72	183,03	89,59	100,00
0,00	0,00	0,00	188,00	0,00	100,00
5868,00	43,90	15,32	179,32	28,78	100,00

- Versch. IOPS = 100%

IOPS

Was ist denn realistisch?

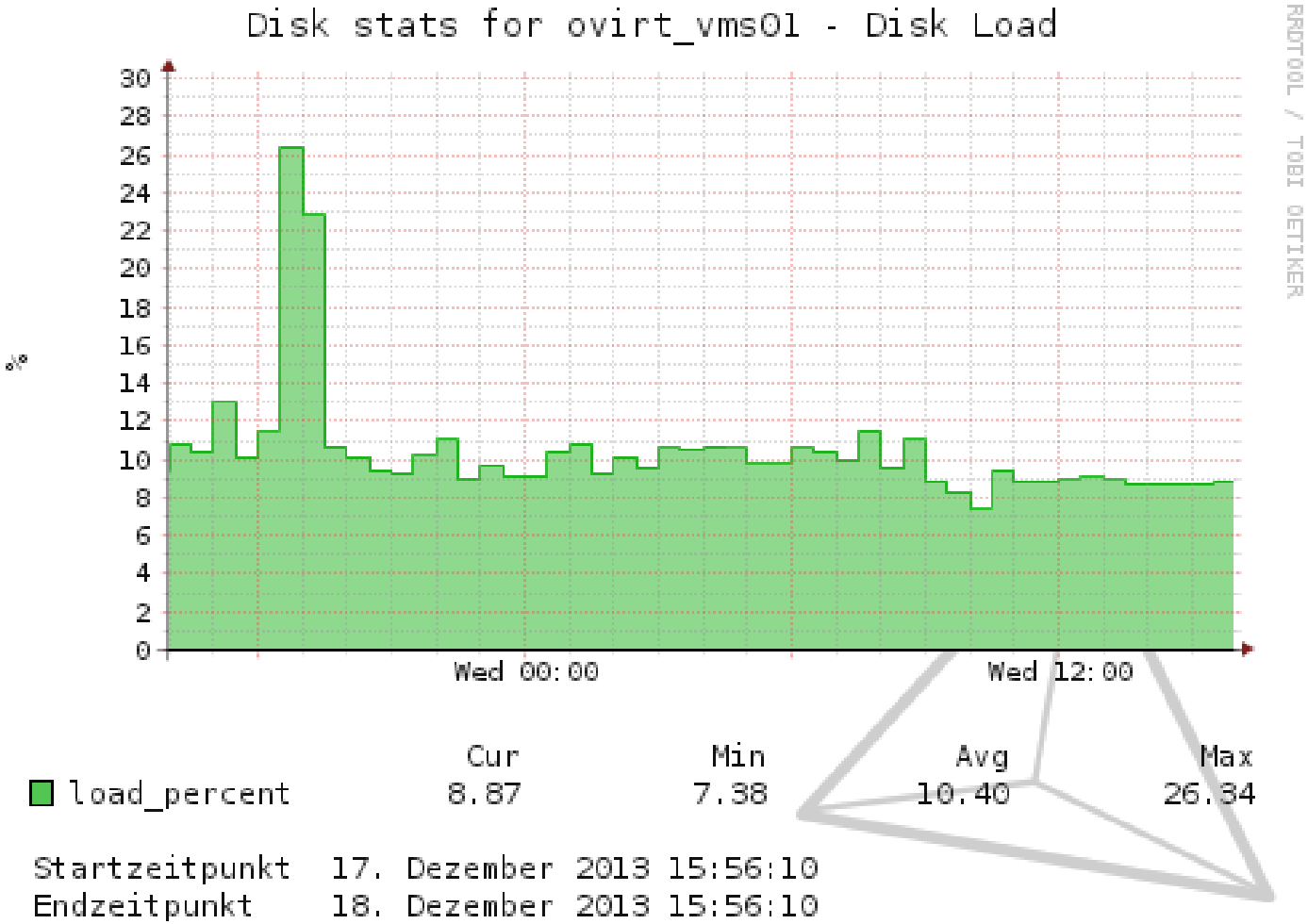
IOPS


Disk stats for ovirt_vms01 - IOs per Second (Normalized)
read (-) / write (+)
4.0 k
3.0 k
2.0 k
1.0 k
0.0
-1.0 k
-2.0 k
-3.0 k
-4.0 k
-5.0 k
-6.0 k
-7.0 k
Wed 00:00
Wed 12:00
RRDTOOL / TOBI OETIKER
Cur
Min
Avg
Max
rd_normiops 3.50 2.18 144.92 6.92k
wr_normiops 2.97k 2.31k 3.05k 3.85k
Startzeitpunkt 17. Dezember 2013 15:56:10
Endzeitpunkt 18. Dezember 2013 15:56:10

IOPS

30%

Auslastung des Live-Systems



Live-System

w/s	wMB/s	avgrq-sz	avgqu-sz	await	%util
254.19	13.95	112.95	0.34	1.26	13.52
947.00	53.94	116.66	0.26	0.27	21.60
140.00	7.26	106.17	0.01	0.06	0.80
89.00	3.55	81.76	5.67	0.04	72.80
363.00	15.91	89.30	1.31	19.11	36.40
418.00	17.23	84.43	0.18	0.44	14.40
1062.00	31.58	60.89	0.16	0.15	6.80
1054.00	44.83	87.11	0.23	0.22	22.40
1274.00	71.01	114.16	0.06	0.05	5.60

Live-System

w/s	wMB/s	avgrq-sz	avgqu-sz	await	%util
254.19	13.95	112.95	0.34	1.26	13.52
947.00	53.94	116.66	0.26	0.27	21.60
140.00	7.26	106.17	0.01	0.06	0.80
89.00	3.55	81.76	5.67	0.04	72.80
363.00	15.91	89.30	1.31	19.11	36.40
418.00	17.23	84.43	0.18	0.44	14.40
1062.00	31.58	60.89	0.16	0.15	6.80
1054.00	44.83	87.11	0.23	0.22	22.40
1274.00	71.01	114.16	0.06	0.05	5.60

- Queue Length und Latenz in der Regel < 1

Live-System

w/s	wMB/s	avgrq-sz	avgqu-sz	await	%util
254.19	13.95	112.95	0.34	1.26	13.52
947.00	53.94	116.66	0.26	0.27	21.60
140.00	7.26	106.17	0.01	0.06	0.80
89.00	3.55	81.76	5.67	0.04	72.80
363.00	15.91	89.30	1.31	19.11	36.40
418.00	17.23	84.43	0.18	0.44	14.40
1062.00	31.58	60.89	0.16	0.15	6.80
1054.00	44.83	87.11	0.23	0.22	22.40
1274.00	71.01	114.16	0.06	0.05	5.60

- Keine erkennbare Korrelation zwischen `w/s` und `%util`!

Live-System

w/s	wMB/s	avgrq-sz	avgqu-sz	await	%util
254.19	13.95	112.95	0.34	1.26	13.52
947.00	53.94	116.66	0.26	0.27	21.60
140.00	7.26	106.17	0.01	0.06	0.80
89.00	3.55	81.76	5.67	0.04	72.80
363.00	15.91	89.30	1.31	19.11	36.40
418.00	17.23	84.43	0.18	0.44	14.40
1062.00	31.58	60.89	0.16	0.15	6.80
1054.00	44.83	87.11	0.23	0.22	22.40
1274.00	71.01	114.16	0.06	0.05	5.60

- Hier übrigens auch nicht...

IOPS

- Das Verhalten unter Volllast ist nicht aussagekräftig.
- maximale IOPS = irrelevant.

Also?

Also was hilft?

Also?

Wer von euch hat eine SSD?

Also?

Wer will seine HDD wiederhaben?

Vergleich: Volllast vs. Normalbetrieb

await	await
32.47	1.26
29.96	0.27
46.62	0.06
28.35	0.04
101.86	19.11
89.59	0.44
0.00	0.15
28.78	0.22

Die Latenz spürt man bei **jeder einzelnen** IO-Operation.

Latenz verringern

- Caching in Hardware (BBU, Flash)
- Hardware-nahe Berechnung der Parity
- Disk-Alignment:

```
Units = sectors of 1 * 512 = 512 bytes

   Device       Start          End      Blocks ...
/dev/sda1        2048    102402047    51200000 ...
```

Latenz umgehen

w/s	wMB/s	avgrq-sz	avgqu-sz	await	%util
254.19	13.95	112.95	0.34	1.26	13.52
947.00	53.94	116.66	0.26	0.27	21.60
140.00	7.26	106.17	0.01	0.06	0.80
89.00	3.55	81.76	5.67	0.04	72.80
1062.00	31.58	60.89	0.16	0.15	6.80
1054.00	44.83	87.11	0.23	0.22	22.40
1274.00	71.01	114.16	0.06	0.05	5.60

- Request-Größe von Linux-VMs = konstant 4KiB (8 Sektoren).
- Die Storage merged ca. 10 VM-Requests zu einem.
- Wir warten nur **einmal** statt zehnmal auf die Latenz!

Richtig messen

1. Kein sequenzielles IO.
   - Achtung: “Random-IO” mit großer Blocksize ($\geq$ 256 KiB) ist nicht mehr random!
2. Max. 30% Last.
3. Durchschnittswerte über einen langen Zeitraum (Nagios-Graphen) verdecken kurze Ausschläge.
4. iostat mit kleinem Intervall sagt nichts über die durchschnittliche Performance.

Richtig messen: Distmark

- Simuliert VMs
  - Mehrere Prozesse
  - Random-IO
  - konstant 4KiB
- Limitierung der max. IOPS
- https://bitbucket.org/Svedrin/distmark

Kontakt

- @just_svedrin
- #openattic in Freenode
- michael@open-attic.org
- openATTIC-Stand