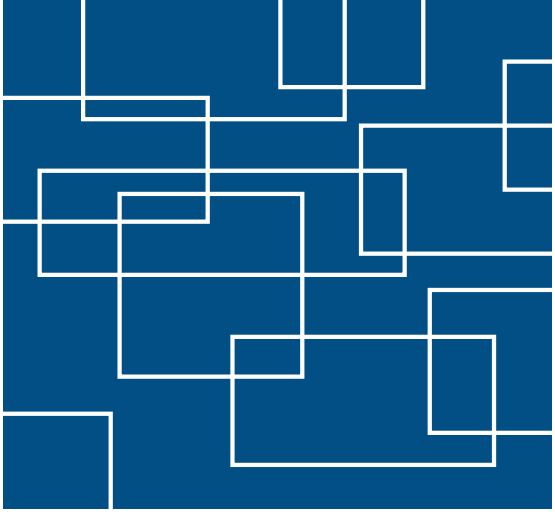
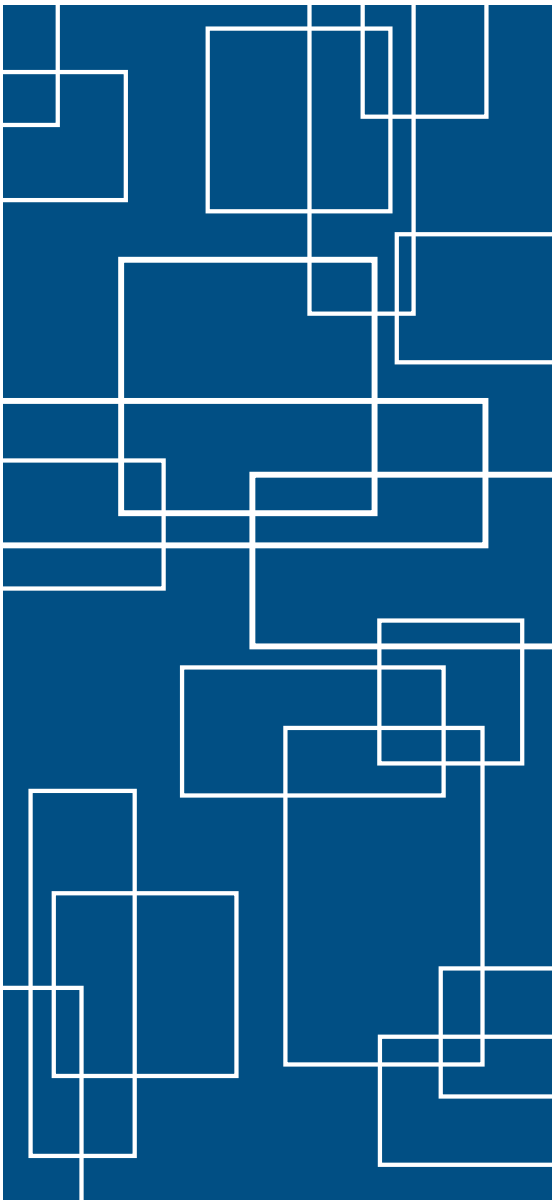


I/O-Scheduler und RAID-Performance



Linux bringt vier verschiedene I/O-Scheduler mit. Nach umfangreichen Tests zeigt dieser Artikel, welche davon mit verschiedenen RAID-Systemen und Controllern am besten harmonieren.

Dr. Volker Jaenisch, Martin Klapproth, Patrick Westphal

Welchen Einfluss hat die Wahl des I/O-Schedulers auf die Festplattenperformance eines Systems? 2006 zeigte eine Studie von IBM (1) je nach Applikation und der gewählten Kombination aus I/O-Scheduler und -Subsystem gewaltige Unterschiede in der Performance. Derartige Untersuchungen betrachten aber meist nur die Antwortzeiten einer komplexen Mess-Suite und gehen nicht näher auf Größen wie Durchsatz (Lesen, Schreiben, Sequenziell, Random), Zugriffszeit oder CPU-Auslastung ein, die für ein I/O-System charakteristisch sind. Deshalb untersucht dieser Artikel die vier Linux-I/O-Scheduler NOOP, Deadline, Anticipatory und CFQ in Kombination mit Hardware RAID 1, RAID 5 (mit 3 und 4 HDDs), Software-RAID 1 und RAID 5 und wirft einen Blick auf den Einfluss des Native Command Queuing (NCQ).

I/O-Scheduler bei der Arbeit

Vor 30 Jahren hatte eine Festplatte wie die Seagate ST-225, eine Kapazität von 20 MByte und eine Zugriffszeit von 65 ms. Heute besitzen die Nachfolger Kapazitäten im Terabyte-Bereich und typischerweise Acces Times um 3.5 ms. Während der Speicherplatz also um einen Faktor 50.000 wuchs, schrumpfte die Zugriffszeit nur auf ein Achtzehntel. Dagegen haben auch die Bussysteme (Faktor 1600) und die CPUs (Faktor 1000) in ihren Taktraten mächtig zugelegt. Aber die Festplatten mit ihren mechanischen Eigenschaften wie der Zugriffszeit folgen nicht der rasanten Entwicklung der Halbleitersysteme getreu dem Mooreschen Gesetz. Daher erhalten heute Software-Lösungen mehr und mehr Beachtung, die den kritischen Faktor Zugriffszeit der I/O-Subsysteme reduzieren, zum Beispiel eben die Scheduler.

I/O-Scheduler versuchen, unnötige Kopfbewegungen zu vermeiden. Sie reorganisieren die Reihenfolge von I/O-Zugriffen, um so den Durchsatz der Festplatten zu erhöhen. Damit diese Reorganisation erfolgreich sein kann, muss der Scheduler wissen, welche Abfolge von I/O-Zugriffen wie lange benötigt. Die I/O-Scheduler gehen typischerweise von einem Zusammenhang zwischen den Blocknummern zweier Datenblöcke und der Zugriffszeit aus. Ist dabei die Differenz der Blocknummern:

- groß, dann ist wahrscheinlich auch die Zugriffszeit groß, ist sie dagegen
- klein, dann ist auch die Zugriffszeit klein.

Abbildung 1 illustriert dies anhand einer vereinfachten Darstellung der Blockverteilung einer Festplatte. Es zeigt sich, dass tatsächlich geringe

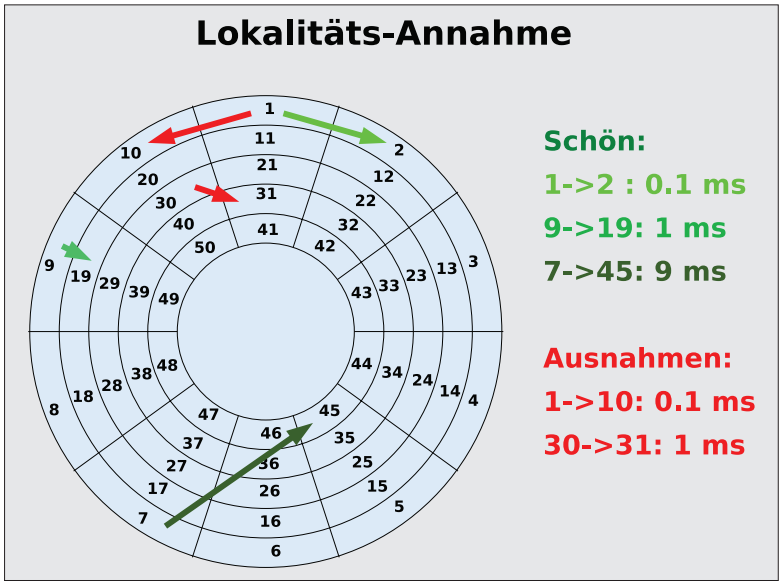


Abbildung 1: Die Lokalitäts-Annahme. Eine Festplatte ist in Sektoren und Spuren unterteilt. Diese formen Blöcke, welche von außen nach innen in Spiralen nummeriert sind. Die grünen Pfeile zeigen Zugriffe, die der Lokalitäts-Annahme der I/O-Scheduler gut entsprechen, rote Pfeile zeigen Abweichungen. Sowohl längere als auch kürzere Wege sind möglich.

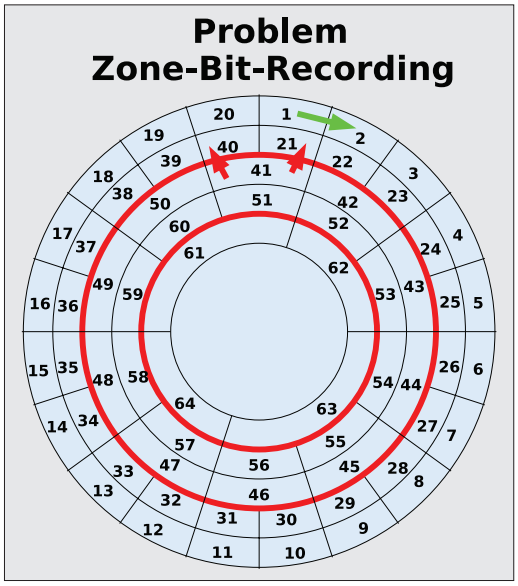
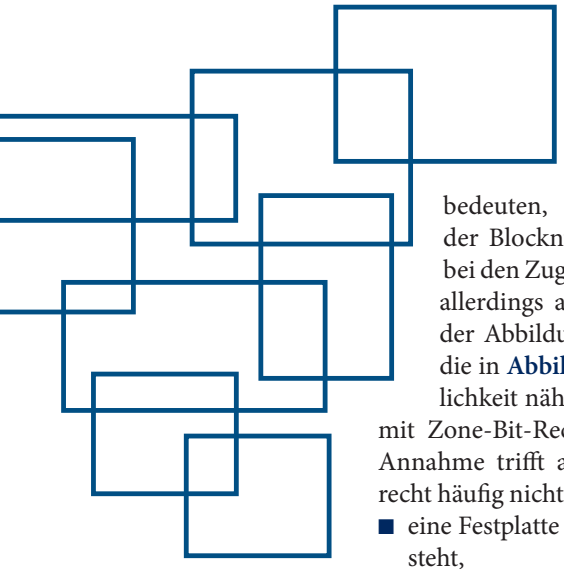


Abbildung 2: Beim heute üblichen Zone-Bit-Recording werden weiter außen liegende Spuren dichter beschrieben als die Inneren. Die roten Linien markieren Blockgrenzen, an denen die Lokalitäts-Annahme der I/O-Scheduler verletzt wird.



Unterschiede in den Blocknummern auch geringe Differenzen in der Zugriffszeit bedeuten, dagegen große Unterschiede der Blocknummern auch große Sprünge bei den Zugriffszeiten verursachen. Es gibt allerdings auch ein paar Ausnahmen (in der Abbildung rot markiert). Aber schon die in **Abbildung 2** dargestellte, der Wirklichkeit näher kommende Blockaufteilung mit Zone-Bit-Recording zeigt: Die Lokalitäts-Annahme trifft an den Block-Dichte-Grenzen recht häufig nicht zu. Dazu kommt, dass:

- eine Festplatte oft aus mehreren Scheiben besteht,
- ein RAID-System Blöcke auf mehrere Platten verteilt,
- niemand die Festplatten-Hersteller dazu zwingt die Blöcke systematisch zu verteilen.

Schon jetzt scheint die Lokalitäts-Annahme auf eher tönernen Füßen zu stehen. Gutes I/O-Scheduling ist aber darüber hinaus auch von Faktoren abhängig wie :

- Hersteller und Typ der Disks und Controller,
- RAID-Level und -Algorithmus
- Typ des Storage-Systems (lokal, SAN, NAS),
- gewähltem I/O-Scheduling Algorithmus.

Details

Die Interna der I/O-Scheduling-Algorithmen finden sich im Linux-Magazin (**(2)**,**(3)**) und in den Vortragsunterlagen zu den Chemnitzer Linuxtagen **(4)**. Daher geht der Artikel im Folgenden nur auf die wesentlichen Charakteristika der Scheduler ein.

- Der **NOOP**-Scheduler (im Folgenden kurz NOOP genannt) ist nur eine einfache Warteschlange ohne jede Intelligenz.
- Der **Deadline**-Scheduler (Deadline) sortiert die I/O-Zugriffe nach ihrer Blocknummer in einer Warteschlange. Um zu verhindern, dass Blöcke mit stark abweichenden Blocknummern verhungern, wird für jeden Block über einen Deadline-Timer eine maximale Auslieferungszeit erzwungen.
- Der **Anticipatory**-Scheduler (Anticipatory) arbeitet wie Deadline, pflegt aber zusätzlich eine Statistik über die Zugehörigkeit der Blöcke zu Prozessen. So kann er die typische Rate schätzen, mit der ein Prozess den I/O-Scheduler bedient. Diese Statistik ermöglicht es dem Scheduler quasi in die Zukunft zu schauen und zu ahnen, wann wohl der nächste Block von welchem Prozess an die Reihe kommen wird. Damit Anticipatory effektiv funktionieren kann, sollte ein Prozess nur aus einer Datei schreiben oder lesen und die Zugriffe müssen sequenziell erfolgen.
- Der aktuelle Standard-I/O-Scheduler von Linux ist der **Complete Fair Queuing Scheduler** (CFQ). Er wählt einen anderen Ansatz als die bisher genannten Scheduler. Ähnlich wie Anticipatory führt der CFQ eine Prozess/Block-Statistik, nutzt diese aber dazu, die verfügbare I/O-Bandbreite so fair wie möglich auf die einzelnen Prozesse zu verteilen. Anders als seine Scheduler-Kollegen garantiert CFQ weder höchsten Durchsatz noch geringste Zugriffszeit. Sein Ziel ist ein ausgewogenes Antwortverhalten über alle Prozesse, wie es bei interaktiven Applikationen, zum Beispiel bei graphischen Desktops benötigt wird. Der **Kasten „I/O-Scheduler einstellen unter Linux“** zeigt, wie Linux-Admins den Scheduler für einzelne Devices auswählen.

Selbst aus dieser stark verkürzten Darstellung ergeben sich sofort zwei wichtige Fragen:

- Sind die Voraussetzungen für Anticipatory Scheduling vielleicht zu naiv? Wie schaut das bei einem Webserver aus, wo ein Prozess beliebig viele Dateien offen haben kann?
- Wieso ist für Linux, das überwiegend im Server-Bereich eingesetzt wird, der eher für Desktops optimierte CFQ-Scheduler der Standard?

I/O-Scheduler einstellen unter Linux

Die I/O-Scheduler aller Block-Devices können zur Bootzeit über den Kernel-Parameter »elevator« gesetzt werden, bei Grub beispielsweise so:

```
kernel /vmlinuz-2.6.23-1-amd64 root=/dev/mapper/vg0-lv0 ro ➥
single mem=256M elevator=deadline
```

Auch zur Laufzeit ist es möglich, über Kernel-Variablen den I/O-Scheduler für jedes Blockdevice zu setzen:

```
echo <schedulername> > /sys/block/<device>/queue/scheduler
```

Die möglichen Schedulernamen liefert

```
cat /sys/block/<device>/queue/scheduler
```

Testaufbau und Messung

Für die Performance-Tests (**Kasten „Testaufbau und Test-Software“**) wurde die Software TIOtest **(5)** eingesetzt. Das Tool liefert Lese- und Schreibraten, Zugriffszeiten und vor allem die

Testaufbau und Test-Software

Zum Einsatz kamen AMD64-Systeme mit Dual Dual Core 2212 Opteron CPUs und 4 GByte RAM. Die drei verwendeten RAID-Controller waren:

```
3ware 9950 SXU-8LP (SATA 2)
Driver Version = 2.26.02.008
Model = 9550SXU-8LP
Memory Installed  = 112 MByte
Firmware Version = FE9X 3.04.00.005
Bios Version = BE9X 3.04.00.002
Monitor Version = BL9X 3.02.00.001
Controller Bus Type = PCIX
Controller Bus Width = 64 bits
Controller Bus Speed = 133 MHz
3ware 9650-2LP (SATA 1)
Driver Version = 2.26.02.009
Model = 9650SE-2LP
Memory Installed  = 112 MByte
Firmware Version = FE9X 3.08.00.016
Bios Version = BE9X 3.08.00.004
Monitor Version = BL9X 3.08.00.001
Serial Number = L325007B7460776
Controller Bus Type = PCIe
Controller Bus Width = 1 lane
Controller Bus Speed = 2.5 Gbps/lane
3ware 9650-4LP
Driver Version = 2.26.02.010
Model = 9650SE-4LPML
Memory Installed  = 224 MByte
Firmware Version = FE9X 3.08.00.016
Bios Version = BE9X 3.08.00.004
Monitor Version = BL9X 3.08.00.001
Serial Number = L326001A7010112
Controller Bus Type = PCIe
Controller Bus Width = 4 lanes
Controller Bus Speed = 2.5 Gbps/lane
```

Die Controller laufen im Modus „Performance". 3ware hat den 9650-4LP für die Messungen mit der aktuellsten Firmware bestückt.

Bei den getesteten Festplatten handelte es sich zum einen um Western-Digital WD RE 250YS (SATA 2) mit 250 GByte Kapazität. Leider hat WD es nicht mehr rechtzeitig geschafft, HDDs mit aktueller Firmware zur Verfügung zu stellen, daher basiert der Test hier auf einfachen, handelsüblichen Geräten.

Als Vergleich kamen Festplatten von Seagate zum Einsatz, hier stellte der Hersteller freundlicherweise Baracuda ES (SATA 2) ES ST3250620NS mit 250 GByte Kapazität und der Firmware-Version 3.AES zur Verfügung.

Software

Als Testsoftware wurde TIOtest verwendet. Dieses Tool simuliert multithreaded Zugriffe auf das I/O-Subsystem, wobei die Manpage dazu rät, den Speicher des Servers auf 16 MByte RAM zu begrenzen, um Caching-Effekte zu eliminieren. Testmessungen mit 64, 128, 256 und 512 MByte RAM zeigen aber keine Abhängigkeit der Messergebnisse vom RAM-Ausbau. Wesentlich ist nur, dass die zur Messung herangezogenen Test-Files deutlich größer sind als der verfügbare Hauptspeicher des Testsystems. Da die I/O-Scheduler und das Software-RAID aber genügend RAM benötigen, haben die Tester mit 256 MByte hier etwas Luft gelassen. Die Testfiles sind mit 1.2 GByte mehr als viermal größer als der verfügbare Hauptspeicher.

Interpretation der Ergebnisse

Die Interpretation der Werte der CPU-Auslastung von TIOtest gestaltet sich schwierig. Ursache ist ein Fehler, der sich in den Code von TIOtest eingeschlichen hat. Als normierende Größe dient die Wall-Clock-Time und nicht die in den Threads verbrachte Zeit. Dies führt dazu, dass unrealistische SYS-CPU-Werte von über 300 Prozent erreicht werden. Daher verwendet dieser Artikel die Größe CPU-Effizienz, die nicht von der falschen Normierung auf die Wall-Clock-Time beeinflusst ist.

Das RAID-Szenario

Die Festplatten stehen bei den Hard- und Software-RAID-Tests vollständig als eine einzige RAID-Partition zur Verfügung. Als Betriebssystem läuft Debian Etch AMD64 mit Kernel 2.6.23 (2.6.24 für NCQ=ON) im Single-User-Mode auf einem separaten RAID 1, um jede unerwünschte Wechselwirkung mit der Messung zu minimieren.

Für jedes Testszenario haben die Tester 16 Messungen durchgeführt: Für jeweils 1, 2, 4 und 8 Threads wurde auf allen beteiligten Blockdevices der gewünschte I/O-Scheduler eingestellt. Zwischen den einzelnen Messungen liegt eine Wartepause von 5 Sekunden. Der Messlauf eines Testszenarios dauerte zwischen 45 und 75 Minuten. Die Auswertung der TIO-Daten und das damit verbundene Erstellen der Grafiken erfolgt vollständig automatisch, um Übertragungsfehler zu eliminieren.

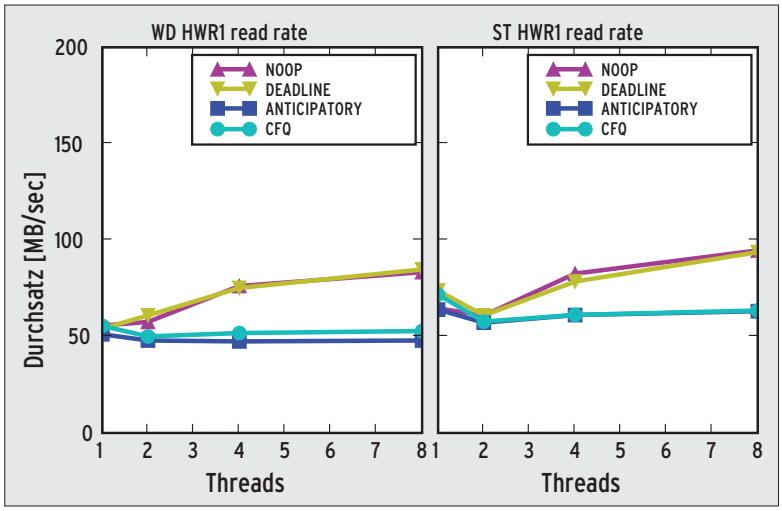


Abbildung 3: Sequenzielle Leserate von 1, 2, 4 und 8 parallelen Threads für die vier I/O-Scheduler bei Hardware RAID 1. Links Messung A: Western-Digital (SATA 2) HDDs auf einem 3ware 9650-2LP (SATA 1), rechts Messung B: Seagate HDDs (SATA 2) auf einem 3ware 9650-4LP (SATA 2).

CPU-Auslastung während des Tests. Weiterhin lässt sich damit der simultane Zugriff paralleler Threads auf das I/O-System messen. Das Verhalten einer Messgröße, wie zum Beispiel der sequenziellen Leserate unter 1, 2, 4 oder 8 Threads, zeigt, wie gut die Scheduler mit steigender Zahl von parallelen Zugriffen klarkommen.

Die Tester verwendeten bewusst keine applikationsbasierten Festplatten-Stress-Tester. Diese simulieren zwar effektiv die komplexen Zugriffsmuster von Datenbanken oder Webservern, erlauben aber gerade deshalb keinen differenzierten Blick in das Innere der I/O-Scheduler

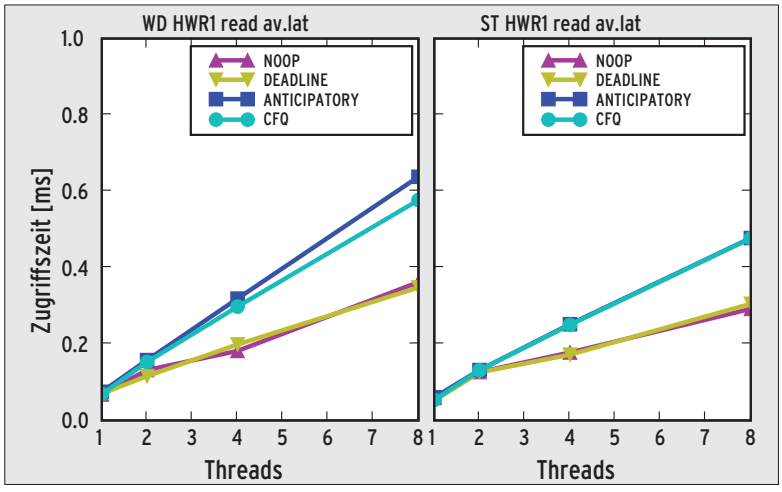


Abbildung 4: Durchschnittliche Zugriffszeiten beim sequenziellen Lesen von 1, 2, 4 und 8 parallelen Threads für die vier I/O-Scheduler bei Hardware RAID 1. Bereits hier zeigen sich zwei Gruppen mit ähnlichem Ergebnis.

und Controller. Die Aussagen großer Mess-Suites sind daher in ihrer Allgemeingültigkeit häufig sehr beschränkt. Sie liefern Aussagen wie: „Mit dieser Kombination aus HDD und Controller schafft eine Postgres-Datenbank 25 Requests pro Sekunde." Das ist zwar eine wahre Aussage, nur leider liefert sie nicht die Erkenntnis, welche Komponente des Systems wie stark und auf welche Weise zu dieser Rate führt. Weil gerade diese Informationen aber bei der Projektierung von IT-Systemen relevant sind, ist dieser Artikel auf der Suche nach eher allgemeingültigen Aussagen wie: „Für Software-RAID 5 ist CFQ der beste Scheduler."

HW RAID 1

Abbildung 3 zeigt die sequenzielle Leseleistung der vier I/O-Scheduler für zwei Hardware RAID Konfigurationen mit RAID 1 und Festplatten unterschiedlicher Hersteller. Im linken Bild (Messung A) sind die Transferraten der vier I/O-Scheduler für sequenzielles Lesen über der Anzahl der simultanen Threads zu sehen. Bei einem Thread ist die Leserate mit rund 55 MByte/s über alle Scheduler gleich. Mit steigender Zahl der Threads spaltet sich die Leserate in zwei Zweige auf: Einen oberen Zweig, der von NOOP und Deadline gebildet wird und rund 80 MByte/s Leserate bringt, und einen unteren Zweig (CFQ und Anticipatory), welcher bei rund 50 MByte/s konstant bleibt.

Im rechten Diagramm (Messung B) sehen wir dieses Verhalten bei einer anderen Hardware-Konfiguration bestätigt, auch wenn die Leserate hier tendenziell etwas höher liegt. Diese leicht erhöhte Rate im rechten Bild ist bedingt durch den Trend von Seagate-HDDs, leicht höhere Leseraten zu liefern als die Konkurrenz von Western-Digital. Die WD-HDDs glänzen dafür mit leicht höheren Schreibraten. Auch sind die WD-HDDs in diesem Messbeispiel durch den kleineren Controller (nur SATA 1) etwas benachteiligt.

Die primitiveren Scheduler NOOP und Deadline schneiden bei beiden Systemen deutlich besser ab als die komplexeren Scheduler Anticipatory und CFQ. Weiterhin sehen wir, dass die Effekte durch unterschiedliche Controller (SATA 1 vs. SATA 2) oder die Marke der Festplatten hinter den Einfluss des I/O-Schedulers zurücktreten.

Betrachtet man die Messwerte noch genauer, dann stellt sich die Frage: Wie kann es sein, dass

die Leserate bei steigender Anzahl von Threads bei NOOP und Deadline zunimmt? Am Algorithmus kann es nicht liegen, da NOOP ja keine Intelligenz besitzt und sich Deadline prinzipiell ähnlich verhält.

Ein ideales Computersystem liefert unabhängig von der Anzahl der Threads die gleiche Leserate. Die verfügbare Bandbreite wird aufgeteilt auf die Threads und entspricht in der Summe wieder der verfügbaren Leserate.

Ein reales Computersystem hat dagegen Reibungsverluste, zum Beispiel durch die Zugriffszeiten der Festplatte und den Verwaltungsaufwand der Threads und I/O-Scheduler. Es ist also bei einem realen System zu erwarten, dass die Leserate mit der Anzahl der Threads abnimmt. Daher erstaunt es sehr, dass beim NOOP- und Deadline-Scheduler die Leserate mit steigender Anzahl von Threads zunimmt, also sogar besser abschneidet als das ideale System.

Des Rätsels Lösung liegt in der RAID-1-Konfiguration. Ein derartiger Verbund kann theoretisch bei zwei Threads doppelt so schnell lesen wie bei einem. Das System verteilt die Lesezugriffe dazu auf die beiden Festplatten so, dass jede Festplatte einen Thread bedient und somit eine minimale Zugriffszeit und maximale Leserate erreicht. Bei mehr als 2 Threads klappt diese Aufteilung aber stetig schlechter und die Leserate eines RAID 1 nimmt mit steigender Anzahl von Threads ab. Die 3ware-Controller folgen diesem theoretischen Verhalten von RAID 1 aber nicht optimal. Bei zwei Threads sollten sie rund 110 MByte/s (2 x 55 MByte/s) liefern. Die maximal erreichte Rate ist aber nur 95 MByte/s, und zwar bei acht Threads. Trotzdem kann das nur Software-RAID 1 übertreffen (siehe unten).

Average Latency

TIOtest gibt eine Größe „Average Latency“ (AL_tio) aus. Diese entspricht aber nicht der Average Latency (AL) welche Festplatten-Hersteller üblicherweise auf den Datenblättern der Festplatten angeben (6). Letztere ist die typische Verzögerung beim Zugriff auf einen Block, die dadurch entsteht, dass die Magnetscheibe sich erst aus der aktuellen Position so weit drehen muss, bis der adressierte Block unter dem Kopf der Platte positioniert ist. Die AL ist ein direkt von der Umdrehungsgeschwindigkeit abhängiges Maß. Für eine Festplatte mit 7200 U/min beträgt die AL=4.2 ms. TIOtest misst für AL_tio die Zeit, die benötigt wird, um einen Block auf die Festplatte zu schreiben oder von der Festplatte zu lesen. Die AL_tio ist daher die Zugriffszeit, also die Summe aus:

- Verarbeitungszeit: Zeit für die Verarbeitung des Requests durch Treiber, I/O-Scheduler, Protokoll,
- Suchzeit: Zeit für die Positionierung des Kopfes in der richtigen Spur,
- Einschwingzeit: Zeit für das Einschwingen des Kopfes in der Spur,
- Latenzzeit: Zeit für die Rotation der Scheibe (im Mittel also die AL).

Abgesehen von der Verarbeitungszeit sind alle anderen Parameter, die in die Zugriffzeit eingehen, direkt an die Hardware der Festplatte gekoppelt. Messungen der Zugriffszeit geben also Aufschluss über die Effizienz der Controller, Treiber und des I/O-Schedulings.

Sequenzielles Lesen

Die Zugriffszeiten beim sequenziellen Lesen (**Abbildung 4**) zeigen die gleiche Verzweigung wie die Messwerte der Leserate. Die Zugriffszeit bei einem Thread ist für alle Scheduler gleich. Sie liegt bei etwa 0.1 ms, was der Zeit für einen Spurwechsel zur nächsten Spur entspricht. Dies zeigt, dass beim sequenziellen Lesen keine großen Bereiche der Platte überstrichen werden. Mit steigender Zahl der Threads werden die Spurwechsel häufiger und die Zugriffszeit steigt linear mit der Zahl der Threads an, allerdings unterschiedlich für die einzelnen Scheduler. Wieder formen sich zwei Zweige: NOOP

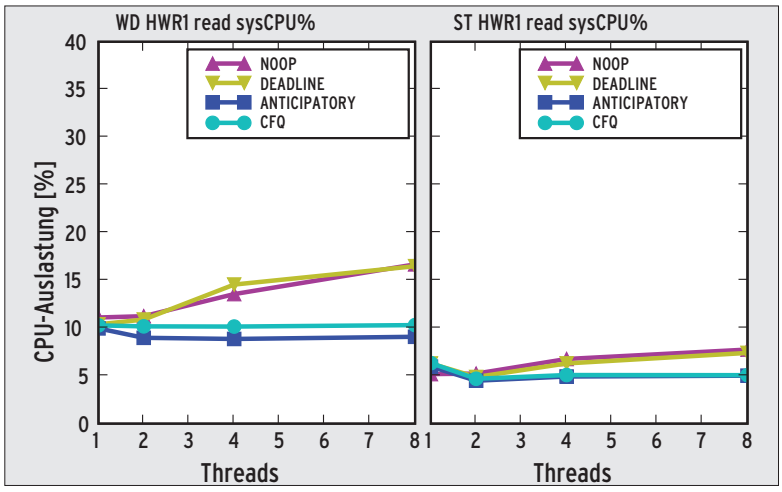


Abbildung 5: SYS-CPU-Auslastung beim sequenziellen Lesen von 1, 2, 4 und 8 parallelen Threads für die vier I/O-Scheduler bei Hardware RAID 1.

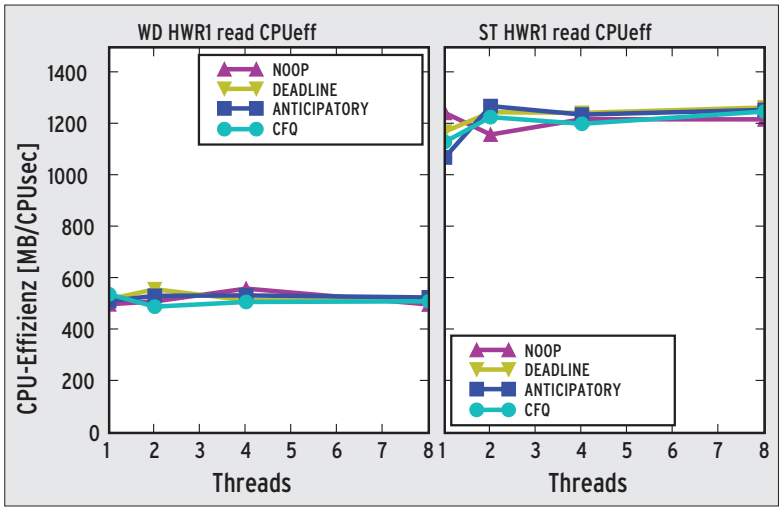


Abbildung 6: CPU-Effizienz beim sequenziellen Lesen von 1, 2, 4 und 8 parallelen Threads für die vier I/O-Scheduler bei Hardware RAID 1.

und Deadline steigen mit rund 0.04 ms/Thread wohingegen die Zugriffszeit bei Anticipatory und CFQ mit rund 0.075ms/Thread fast doppelt so schnell ansteigt.

CFQ hatte nie versprochen, niedrigste Zugriffszeiten und höchste Datenraten zu erzielen und sei damit freigesprochen. Anticipatory zeigt aber kein anderes Verhalten als CFQ und wird sogar von NOOP übertroffen. Für sequenzielles Lesen bei Hardware RAID 1 scheinen also ganz klar NOOP oder Deadline die bessere Wahl zu sein. Aber wie sieht es mit der CPU-Auslastung der Scheduler aus? In **Abbildung 5** sehen wir die prozentuale SYS-CPU-Auslastung, also den Anteil der Zeitscheiben, in denen Kernel-Prozesse ausgeführt wurden. Die Treiber für das I/O-Subsystem wie auch die Treiber für das Ext3-Dateisystem laufen im Kernel-Space. Daher ist der »SYS-CPU%«-Wert ein gutes Maß für die Belastung der CPU durch die Treiber und I/O-Scheduler. Es zeigt sich, dass NOOP und Deadline mehr CPU-Auslastung hervorrufen als Anticipatory und CFQ. Dies verblüfft, wenn man bedenkt, um wie viel simpler NOOP gestrickt ist. Er verwaltet ja nicht mehr als einen einfachen FIFO-Puffer, während CFQ pro Prozessgruppe einen Red-Black-Tree für die Sortierung der Puffer-Warteschlangen pflegen muss.

Dieses paradoxe Verhalten erklärt sich aber leicht damit, dass NOOP und Deadline häufiger CPU-Zeitscheiben brauchen, weil sie weniger Zugriffszeit und damit weniger CPU-WAIT-Zeitscheiben erzeugen. Der höhere Durchsatz bei NOOP und Deadline manifestiert sich also in einer stärkeren Nutzung der CPU.

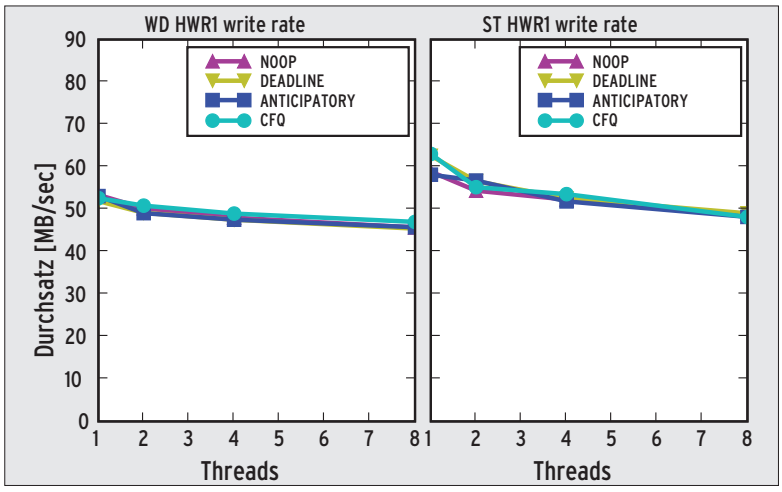


Abbildung 7: Sequenzielle Schreibrate von 1, 2, 4 und 8 parallelen Threads für die vier I/O-Scheduler bei Hardware RAID 1.

CPU-Effizienz

Ein besseres Maß für die CPU-Auslastung durch die I/O-Scheduler ist daher die CPU-Effizienz, die beschreibt, welche Datenmenge der Scheduler pro SYS-CPU-Sekunde bewegt. Wie **Abbildung 6** zeigt, ist die CPU-Effizienz bei allen Schedulern mit zunehmender Zahl von Threads konstant.

Die komplexeren Scheduler verbrauchen also offensichtlich nicht mehr Rechenzeit pro transferierte Datenmenge als die simplen Scheduler. Die deutlich geringere Datenrate bei CFQ und Anticipatory entsteht also durch die schlechtere Passung der Algorithmen, nicht aber durch ineffiziente Programmierung der Scheduler. Zum anderen macht die Konstanz der CPU-Effizienz mit der Anzahl der Threads klar, dass für den Durchsatz bei steigender Zahl von Threads nicht die CPU-Leistung, sondern die Festplatte der begrenzende Faktor ist.

Sequenzielles Schreiben bei Hardware RAID 1

Abbildung 7 zeigt die Raten beim sequenziellen Schreiben. Die Schreibraten fallen bei Messung A (links) von rund 55 MByte/s bis auf rund 48 MByte/s ab, ohne signifikante Unterschiede zwischen den I/O-Schedulern. Bei Messung B (rechts) zeigt sich das gleiche Verhalten: Der Durchsatz fällt bei allen Schedulern von 60 MByte/s auf etwa 50 MByte/s. Die Wahl des I/O-Schedulers scheint also vorrangig das Lesen zu beeinflussen.

Anders als beim sequenziellen Lesen ist beim sequenziellen Schreiben das RAID 1 nicht in der Lage, bei mehr als einem Thread eine höhere Leistung zu erzielen. Beim Schreiben ins RAID 1 muss das System die Information auf beiden Platten möglichst zeitgleich schreiben. Daher sinkt die Schreibrate monoton mit der Anzahl der Threads ab.

Dies wird auch aus den Zugriffszeiten in **Abbildung 8** deutlich. Die Zugriffszeiten nehmen in beiden Messfällen mit rund 0.08 ms/Thread zu. Damit ist der Zuwachs der Zugriffszeit beim sequenziellen Schreiben doppelt so groß wie beim sequenziellen Lesen mit dem besten I/O-Scheduler (Deadline) oder gleich dem Wert beim sequenziellen Lesen mit dem schlechtesten Scheduler (CFQ). Dies legt die Vermutung nahe, dass die Ankopplung des I/O-Schedulers an den RAID-Algorithmus entscheidend für den Erfolg ist. Die CPU-Effizienz ist wie beim sequenziellen Lesen auch beim Schreiben für alle Scheduler gleich.

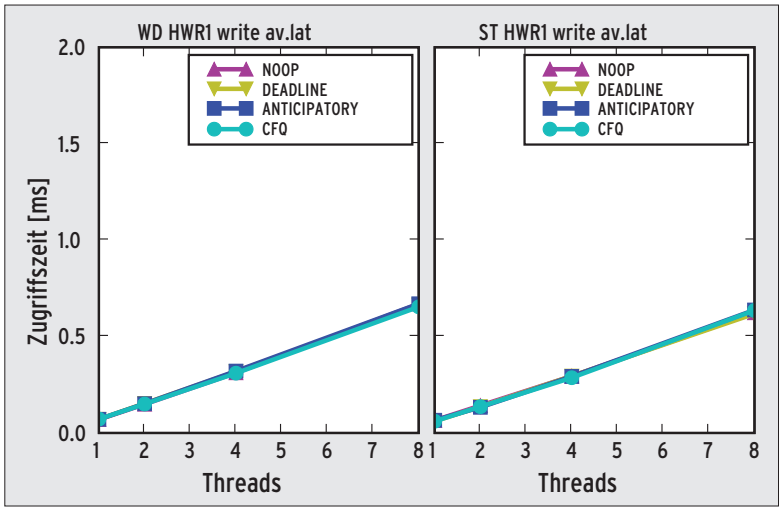


Abbildung 8: Zugriffszeiten beim sequenziellen Schreiben von 1, 2, 4 und 8 parallelen Threads für die vier I/O-Scheduler bei Hardware RAID 1.

Zufälliges Lesen bei Hardware RAID 1

Die I/O-Scheduler unterscheiden sich beim sequenziellen Zugriff zumindest beim Lesen stark in der Performance. Wie verhalten sich die I/O-Scheduler beim zufälligen Lesen? Eigentlich sollten da die hochwertigeren Scheduler mit ihrem größeren Optimierungs-Potenzial zu Ruhm und Ehre kommen.

Abbildung 9 zeigt die vier I/O-Scheduler beim zufälligen Lesen. Zunächst fällt beim Blick auf die Skala auf, dass die Leserate, unabhängig vom Scheduler, um einen Faktor 100 geringer ist als beim sequenziellen Lesen. Die Optimierungsstrategien der Scheduler bauen auf eine den Zugriffen innewohnende Regelmäßigkeit. Diese Regelmäßigkeit ist hier nicht gegeben und alle Scheduler versagen, allen voran Anticipatory.

Dessen Optimierungsstrategie beruht besonders auf der Annahme, dass mehrere Prozesse parallel und jeweils sequenziell lesen. Anticipatory legt bewusst Wartezeiten ein, um die I/O-Zugriffe eines Prozesses zusammenzufassen, die dann eventuell sequenziell aufeinander folgen. Beim zufälligen Lesen sind die Blöcke eines Prozesses aber eben zufällig verstreut, die Wartezeit ist also immer erfolglos investiert und addiert sich für jeden weiteren Thread als zusätzliche Zugriffszeit, was in **Abbildung 10** deutlich zu erkennen ist.

Weder Deadline noch CFQ weisen eine bessere Datenrate als NOOP auf, was zeigt, dass die I/O-Scheduler bei zufälligem Lesen alle keinen Nutzen bringen. Der Effekt, dass die Datenrate mit zunehmender Anzahl der Threads anwächst, ist, wie beim sequenziellen Lesen, dem RAID 1 zuzuschreiben. Ein weiterer Effekt ist die Blockdichte. Je mehr Threads gleichzeitig Blöcke zufällig adressieren, desto höher ist die Wahrscheinlichkeit, dass die zu lesenden Blöcke näher beieinanderliegen und somit die Zugriffszeit sinkt.

Die Zugriffszeit ist mit rund 7ms für einen Thread etwas besser als die im Datenblatt angegebene mittlere Zugriffzeit von rund 8.5 ms. Dies liegt daran, dass der Test-Bereich von TIO-

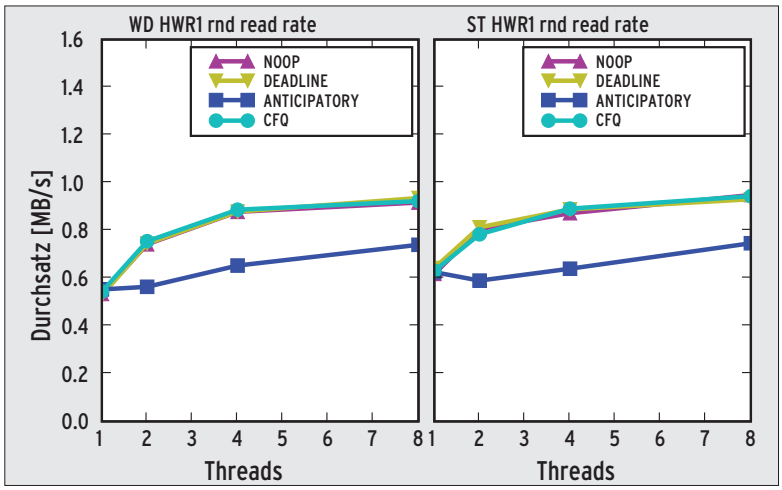


Abbildung 9: Leserate beim zufälligen Lesen von 1, 2, 4 und 8 parallelen Threads für die vier I/O-Scheduler bei Hardware RAID 1.

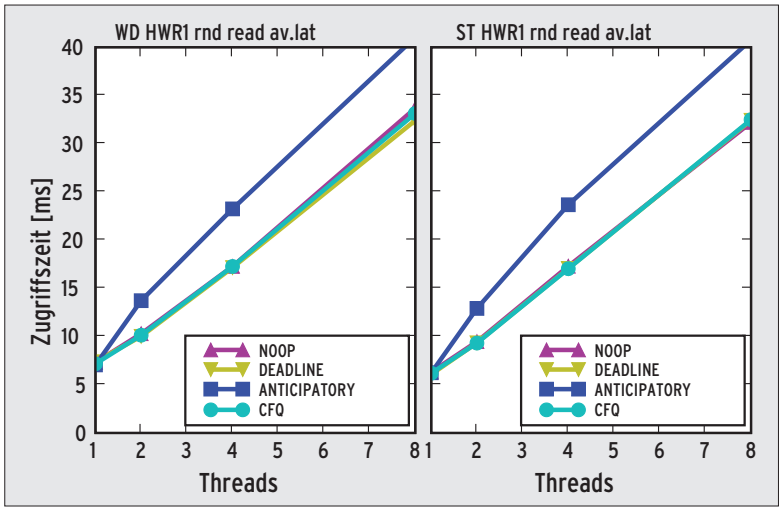


Abbildung 10: Zugriffszeit beim zufälligen Lesen von 1, 2, 4 und 8 parallelen Threads für die vier I/O-Scheduler bei Hardware RAID 1.

test nur einen kleinen Bereich (10 Prozent) der Festplatte umfasst. Pro Thread kommen 3.7 ms Zugriffszeit hinzu, bei Anticipatory sogar 5.1 ms.

Die CPU-Effizienz (**Abbildung 11** und **12**) ist im Falle des zufälligen Lesens bei allen Schedulern deutlich geringer als beim sequenziellen Lesen. Sie streut stark bei wenigen Threads, aber spätestens bei acht Threads sind alle Scheduler gleich auf. Diese starke Streuung wird durch die Granularität der CPU-Zeitscheiben hervorgerufen. Jede davon dauert einen Jiffy, also 4 ms. Wenn die Zeitdauer von I/O-Zugriffen in der Größenordnung von Jiffies abläuft, kommt es folglich zu Interferenz-Effekten und damit zu einer starken Streuung der Messwerte.

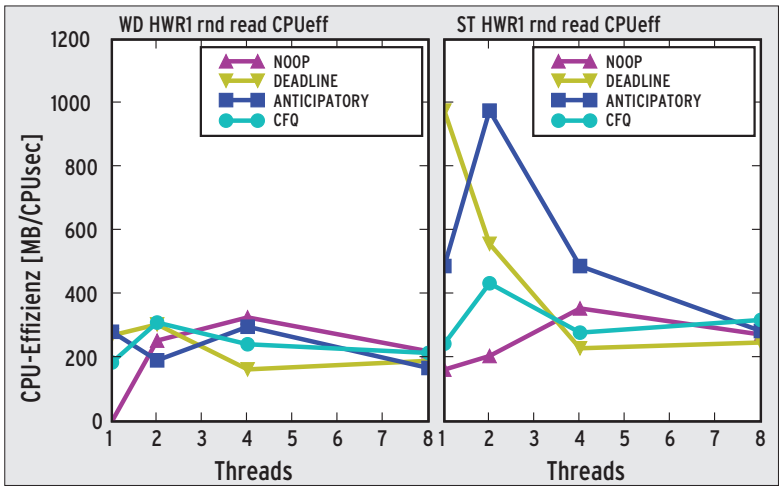


Abbildung 11: CPU-Effizienz bei zufälligem Lesen von 1, 2, 4 und 8 parallelen Threads für die vier I/O-Scheduler bei Hardware RAID 1.

Zufälliges Schreiben bei Hardware RAID 1

Wie beim Lesen bricht die Rate auch beim zufälligen Schreiben bei mehreren Threads zusammen, wobei sie aber nur auf etwa ein Fünfzigstel abfällt. Die Scheduler zeigen bei wenigen Threads eine starke Auffächerung um zum Teil 400-500 Prozent. Gerade Anticipatory, der beim zufälligen Lesen völlig versagte, tut sich hier positiv hervor und auch CFQ hat hier die Nase vorn. Diese Effekte sind keine statistischen Schwankungen.

Mit sinkendem RAM-Ausbau verringert sich aber die Auffächerung. Hier zeigt sich, dass die statistischen Methoden von Anticipatory und CFQ durchaus funktionieren, sofern nicht zu viele Threads parallel arbeiten. Allerdings ist fraglich, ob dieser Effekt in der Realität spürbar ist, da auf einem Linux-Server sicher mehr als 4 Prozesse parallel das I/O-System nutzen. Trotzdem ist zu vermerken, dass CFQ und Anticipatory bei zufälligem Schreiben tendenziell bessere Ergebnisse liefern und Deadline weit abgeschlagen zurückbleibt.

Auch NOOP schlägt sich bei einem Thread besser oder mindestens so gut wie Deadline. Deadline ist bei WD-Platten deutlich besser als bei Seagate-Platten. Dies ist der einzige Punkt im Test, wo signifikant unterschiedliche Ergebnisse zwischen den Festplatten-Herstellern erkennbar sind.

Testergebnisse für RAID 1

Tabelle 1 fasst die Ergebnisse der Tests von Hardware RAID 1 zusammen. Wir wählen dabei Deadline stellvertretend für den Zweig NOOP und Deadline sowie CFQ für den Zweig CFQ und Anticipatory.

Für Hardware RAID 1 ist Deadline beim sequenziellen Zugriff die bessere Wahl gegenüber CFQ. Aber selbst Deadline liegt beim sequenziellen Lesen deutlich unterhalb der theoretisch möglichen Leistung von RAID 1. Beim zufälligen Lesen oder Schreiben brechen alle Scheduler auf einen Bruchteil der Leistung des sequenziellen Zugriffs ein.

Die höherwertigeren Scheduler erreichen beim zufälligen Schreiben einen spürbaren Vorteil zu Deadline. Insgesamt entsteht so ein inkonsistentes Gesamtbild. Deadline bringt eine um 40 Prozent höhere sequenzielle Leserate als CFQ, während dieser eine um Faktoren höhere zufällige Schreibrate zumindest unterhalb von 4

Threads erzielt. Es ist also schwer, hier eine klare Empfehlung für die Wahl des Schedulers auszusprechen. Für einen File-Server würde man in dieser Konfiguration wohl Deadline verwenden, für einen DB-Server mit vielen Inserts wohl eher CFQ. Vor allem aber geben die Tests Anlass, über Software RAID 1 neu nachzudenken.

Software RAID

Software RAID fristet im Server-Bereich ein Schattendasein und gilt als eine Art Notnagel für geringe Budgets. Die landläufige Meinung reicht von CPU-Verschwendung über Instabilität bis zu miserabler Performance. Wir ließen Software RAID 1 gegen unseren Favoriten 3ware 9650-4LP in der Disziplin RAID 1 antreten. Wieder wurden alle vier I/O-Scheduler getestet. Die Domäne von RAID 1 ist das Lesen. **Abbildung 13** zeigt den Vergleich von Hardware- und Software-Variante, **Tabelle 2** fasst die bevorzugten I/O-Scheduler für Hard- und Software RAID zusammen.

Siegten bei Hardware RAID 1 die Scheduler Deadline und NOOP, so ist es beim Software RAID umgekehrt: CFQ und Anticipatory liegen vorne, und zwar mit Abstand. SW RAID 1 ist dabei sogar nahe am theoretischen Optimum von RAID 1.

Bei zwei Threads verdoppelt sich die Leserate beinahe. Schließlich bleibt Sie auf einem erfreulich hohen Niveau von 120 MByte/s über den ganzen getesteten Bereich von Threads. Selbst bei acht Threads ist Software RAID 1 noch 20 MByte/s schneller als Hardware RAID 1. Auch die Zugriffszeiten sind dementsprechend niedriger, einzig bei der CPU-Effizienz (**Abbildung 14**) ist Hardware RAID 1 um rund 40 Prozent besser, was angesichts der Tatsache, dass bei SW RAID die CPU das Verwalten des RAIDs übernimmt, nicht verwundert.

Tabelle 1: Welcher Scheduler ist der beste für HW RAID 1?

	Rate	Zugriffs-zeit	CPU-Eff.
9650-2LP Western Digital HDDs			
Lesen (Seq.)	Dead	Dead	=
Schreiben (Seq.)	Dead	Dead	=
Lesen (Rnd.)	=	=	=
Schreiben (Rnd.)	CFQ	CFQ	=
9650-4LP Seagate HDDs			
Lesen (Seq.)	Dead	Dead	=
Schreiben (Seq.)	Dead	=	=
Lesen (Rnd.)	=	=	=
Schreiben (Rnd.)	CFQ	CFQ	=

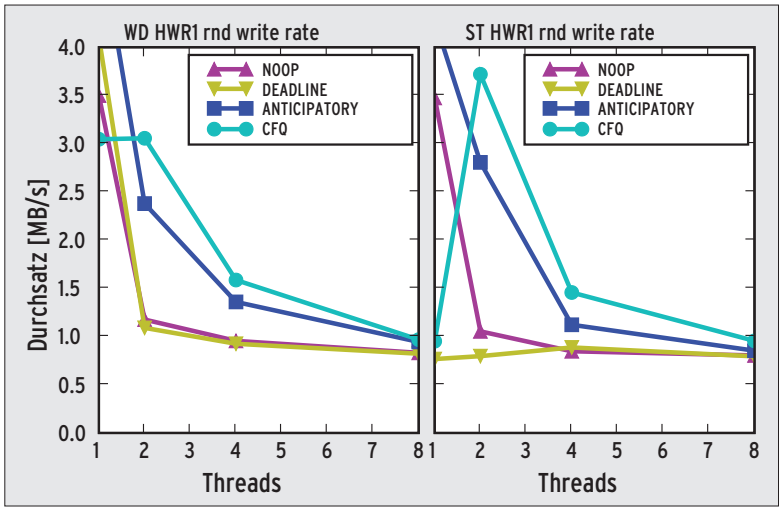


Abbildung 12: CPU-Effizienz bei zufälligem Schreiben von 1, 2, 4 und 8 parallelen Threads für die vier I/O-Scheduler bei Hardware RAID 1.

Vorsicht vor unechten HW RAID

Allerdings liegt die Sache für Hardware RAID nicht ganz so einfach, hier gibt es echte und unechte Hardware-RAID-Controller. In den echten ist ein eigener Microcontroller mit Firmware angesiedelt, der die CPU entlastet. Im Gegensatz dazu delegieren die unechten Hardware-RAID-Controller die RAID-Logik an die Treiber und damit an die CPU. Gerade bei Onboard-RAID-Controllern ist schwer zu

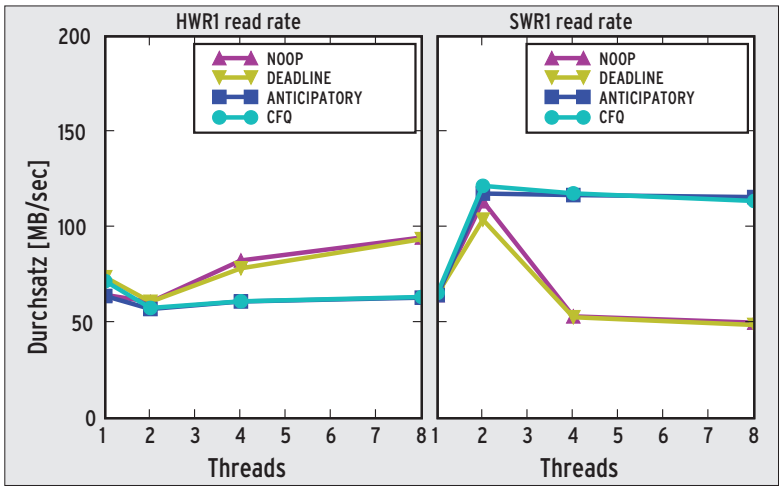


Abbildung 13: Sequenzielle Leserate von 1, 2, 4 und 8 parallelen Threads für die vier I/O-Scheduler bei RAID 1 mit Seagate HDDs (SATA 2). Links: 3ware 9650-4LP (SATA 2). Rechts: Software RAID 1.

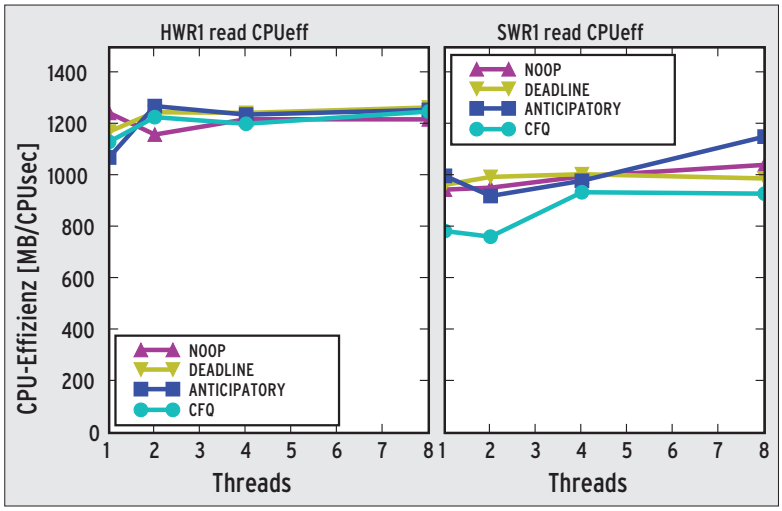


Abbildung 14: CPU-Effizienz beim sequenziellen Lesen von 1, 2, 4 und 8 parallelen Threads für die vier I/O-Scheduler bei RAID 1 mit Seagate HDDs (SATA 2). Links: 3ware 9650-4LP (SATA 2). Rechts: Software RAID 1.

entscheiden, zu welcher Kategorie ein Controller gehört. Da bleibt nur im Einzelfall bei jedem RAID-Controller einen Test auf Durchsatz und CPU-Effizienz.

Bei Software RAID 1 ist CFQ die bessere Wahl. Beim sequenziellen Lesen ist Deadline in der CPU-Effizienz besser als CFQ. Letzterem sei dies allerdings verziehen, da er in dieser Disziplin rund 70-100 MByte/s mehr Durchsatz bringt. CFQ ist damit also für Software RAID 1 der beste I/O-Scheduler. **Tabelle 3** zeigt den Vergleich von Hardware RAID 1 (Deadline) mit Software RAID 1 (CFQ).

Bei den Raten gewinnt Software RAID beim sequenziellen Lesen und der Zugriffszeit beim Lesen. Einzig beim zufälligen Schreiben ist die Rate von Hardware RAID 1 im Schnitt doppelt so hoch wie bei Software RAID. Dieser Vorsprung verliert sich aber bei 8 Threads.

Hardware RAID 1 gewinnt bei der CPU-Effizienz mit etwa 20 Prozent Vorsprung, was aber nicht weiter verwundert. Software RAID 1 liefert eine im Schnitt 40 MByte/s höhere sequenzielle Leserate als Hardware RAID 1. Beim zufälligen Schreiben gewinnt Hardware RAID 1 mit doppelt so hoher Schreibrate wie Software RAID.

Eine wesentliche Erkenntnis ist auch, dass im Gegensatz zum Hardware RAID 1 die Wahl des I/O-Schedulers bei Software RAID 1 unabhängig vom Anwendungsfall ist. CFQ ist bei Software RAID in allen Disziplinen genauso gut oder besser als alle anderen Scheduler.

Tabelle 2: Welcher Scheduler für welches RAID?

	Rate	Zugriffs-zeit	CPU-Eff.
Scheduler für 9650-4LP HW RAID 1			
Lesen (Seq.)	Dead	Dead	=
Schreiben (Seq.)	Dead	=	=
Lesen (Rnd.)	=	=	=
Schreiben (Rnd.)	CFQ	CFQ	=
Scheduler für Software RAID 1			
Lesen (Seq.)	CFQ	=	Dead
Schreiben (Seq.)	=	=	=
Lesen (Rnd.)	=	=	=
Schreiben (Rnd.)	=	=	CFQ

Hardware RAID 5

Bisher hat dieser Artikel RAID 1 Konfigurationen untersucht, aber wie verhalten sich die I/O-Scheduler bei RAID 5? Jetzt kommt RAID 5 mit drei und vier Festplatten an zwei 3ware Controllern unter die Lupe. Für Hardware RAID 5 haben Tester zwei weitere Testsysteme aufgebaut. Testsystem C: 3ware 9550-SXU-8LP (SATA 2) mit WD HDDs (SATA 2) RAID 5 tritt an gegen Testsystem D: 3ware 9650-4LP(SATA 2) mit Seagate HDDs (SATA 2) RAID 5.

Abbildung 15 zeigt die beiden Testsysteme beim sequenziellen Lesen. Zunächst fällt wieder auf, dass beide Systeme eine Leserate in gleicher Größenordnung produzieren. Auch hier findet, wie bei Hardware RAID 1, eine Aufspaltung in die Zweige NOOP/Deadline und Anticipatory/CFQ statt.

Allerdings schneiden diese Zweige in den beiden Testkonfigurationen völlig unterschiedlich ab. Testfall C verhält sich wie die bisher präsentierten Ergebnisse von RAID 1: NOOP und Deadline gewinnen gegen Anticipatory und CFQ. In Testfall D ist es aber genau umgekehrt. Hier erreichen NOOP und Deadline nicht ein-

Tabelle 3: Hardware (H) vs. Software RAID 1(S)

	Rate	Zugriffs-zeit	CPU-Eff.
Lesen (Seq.)	S	S	H
Schreiben (Seq.)	=	=	=
Lesen (Rnd.)	=	=	H
Schreiben (Rnd.)	H	=	H

mal die Hälfte des Durchsatzes von Anticipatory und CFQ. Dieses Verhalten bestätigt erneut die Vermutung, dass die Effizienz des I/O-Scheduling maßgeblich durch den Controller bestimmt ist. Wieso sich die beiden Testfälle so komplementär verhalten, lässt sich wohl nur durch eine Analyse der Firmware der Controller und Festplatten ermitteln. Es findet sich aber in der FAQ zum 9550-SXU-Controller (7) der Hinweis, dass Deadline als Scheduler empfohlen wird, was die Beobachtungen stützt.

Beim 9650-4LP (Testfall D) ist bei einem Thread die Leserate gegenüber dem RAID 1 desselben Controllers mehr als doppelt so hoch. Allerdings erfolgt schon bei zwei Threads ein starker Einbruch auf rund 80 MByte/s. An diesem Wert ändert sich auch bei einer steigenden Anzahl von Threads nichts.

Abbildung 16 zeigt für Testfall D, dass der Durchsatz von NOOP und Deadline sowohl mit 3 als auch mit 4 HDDs unverändert schlecht bleibt. Allerdings profitieren CFQ und sogar noch stärker Anticipatory von der weiteren Festplatte. Es erscheint logisch, dass für RAID-Optimierung mit vielen Platten CFQ und Anticipatory die bessere Wahl darstellen.

Die **Tabelle 4** fasst diese Ergebnisse zusammen. Dazu dienen CFQ und Deadline stellvertretend für die beiden Zweige aus NOOP/Deadline und Anticipatory/CFQ. Bewertet werden Rate, Zugriffszeit und CPU-Effizienz. Im Testfall C siegt Deadline, im Testfall D siegt CFQ klar. Eindeutig ist Deadline beim 9550-SXU-8LP die richtige Entscheidung. Beim 9650-4LP mit einem RAID 5 ist der I/O-Scheduler der Wahl CFQ. Im Vergleich mit **Tabelle 1** zeigt sich, dass selbst bei identischer Hardware (gleicher Controller, gleiche HDDs) je nach Konfiguration (RAID 1 oder RAID 5) ein anderer I/O-Scheduler die bessere Performance liefert.

Tabelle 4: Welcher Scheduler bei Hardware Raid 5 ?

	Rate	Zugriffs-zeit	CPU-Eff.
9550-SXU-8LP			
Lesen (Seq.)	Dead	Dead	=
Schreiben (Seq.)	Dead	Dead	=
Lesen (Rnd.)	=	=	=
Schreiben (Rnd.)	=	=	=
9650-4LP			
Lesen (Seq.)	CFQ	CFQ	=
Schreiben (Seq.)	CFQ	=	=
Lesen (Rnd.)	=	=	CFQ
Schreiben (Rnd.)	=	CFQ	CFQ

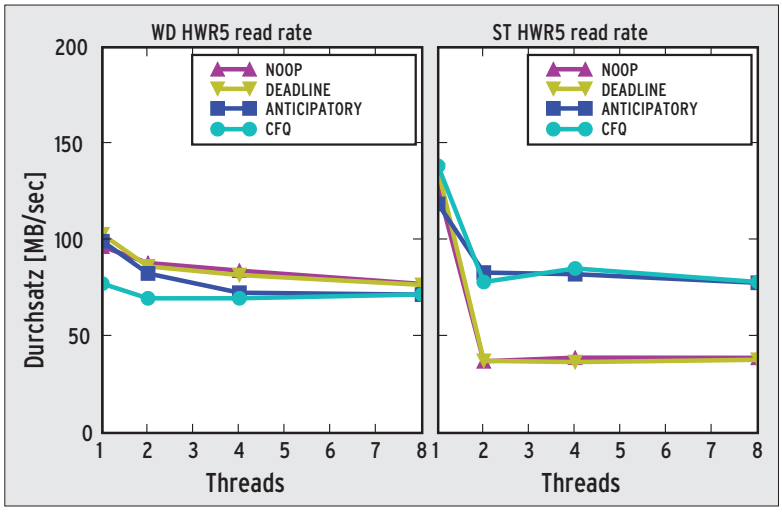


Abbildung 15: Sequenzielles Lesen von 1, 2, 4 und 8 parallelen Threads für die vier I/O-Scheduler bei Hardware RAID 5 (3HDDs). Links Testfall C: Western-Digital (SATA 2) HDDs auf einem 3ware 9550-SXU-8LP (SATA 2), rechts Testfall D: Seagate HDDs (SATA 2) auf einem 3ware 9650-4LP (SATA 2).

SW RAID 5 gegen HW RAID 5

Bei RAID 1 hat Software RAID das Hardware RAID deklassiert. Kann Software-RAID trotz der höheren Belastung durch die größere Rechenarbeit auch bei RAID 5 punkten? **Abbildung 17 und 18** zeigen: Beim sequenziellen Lesen sind Software RAID 5 und Hardware RAID 5 gleich auf (**Abbildung 17**). Die sequenzielle

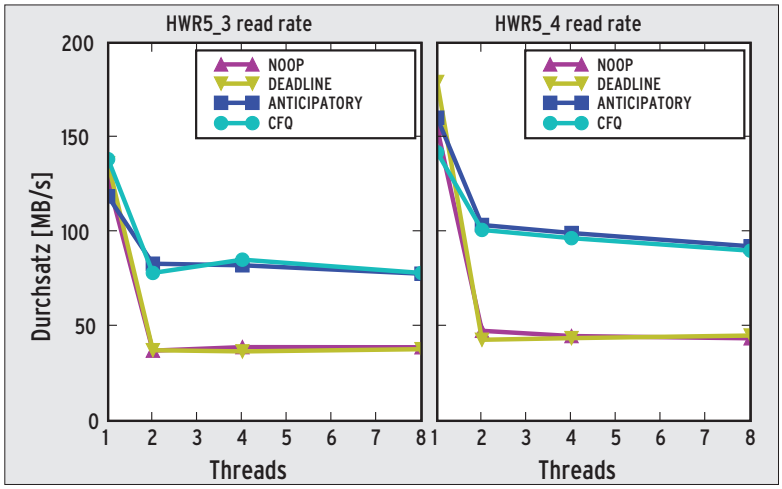


Abbildung 16: Sequenzielles Lesen von 1, 2, 4 und 8 parallelen Threads für die vier I/O-Scheduler bei Hardware RAID 5: Links 3HDDs und rechts 4HDDs. Seagate HDDs (SATA 2) auf einem 3ware 9650-4LP (SATA 2).

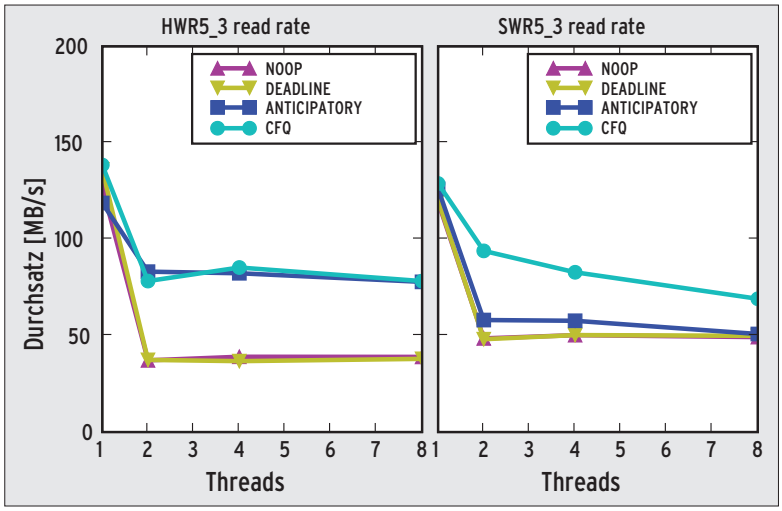


Abbildung 17: Sequenzielles Lesen von 1, 2, 4 und 8 parallelen Threads für die vier I/O-Scheduler auf 3 Seagate HDDs bei HW RAID 5 und Software Raid 5. Links: 3ware 9650-4LP Hardware RAID 5. Rechts: Software RAID 5.

Schreibrate bei Software-RAID 5 (**Abbildung 18**) ist allerdings durchgängig 20 bis 10 MByte/s schlechter als beim Hardware RAID 5 und auch beim zufälligen Lesen und Schreiben hat Hardware RAID knapp die Nase vorn. **Tabelle 5** fasst die Ergebnisse zusammen:

Bei Hardware RAID 5 und Software RAID 5 ist folglich definitiv CFQ der richtige Scheduler. Hardware RAID 5 gewinnt aber nur knapp gegen Software RAID 5. Dass die CPU-Effizienz bei Hardware RAID größer ist als bei Software RAID verwundert dabei nicht weiter: Die Berechnung der Parität erfolgt beim Hardware RAID im Controller und beim Software RAID in

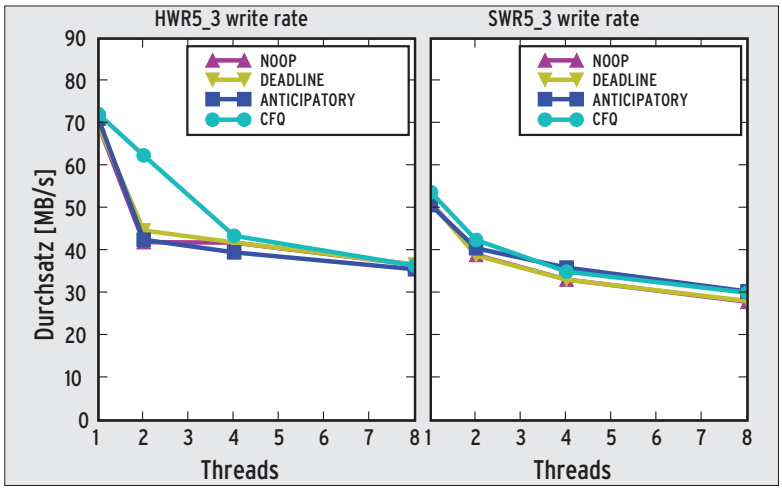


Abbildung 18: Sequenzielles Schreiben von 1, 2, 4 und 8 parallelen Threads für die vier I/O-Scheduler bei HW RAID 5 und SW RAID 5. 3ware 9650-4LP Hardware RAID 5 (links) und Software RAID 5 (rechts) auf 3 Seagate HDDs.

der CPU. Der Vorsprung des Hardware-RAID-Systems beträgt hier aber maximal 20 Prozent. Gönnt man dem Software RAID 5 jedoch eine vierte Festplatte, so überrundet es Hardware RAID 5 mit 3 Festplatten (wie aus **Tabelle 6** ersichtlich, zumindest bei den Leseraten). Dieser Performance-Gewinn wird allerdings auf Kosten der CPU-Auslastung erkauft.

Die Intelligenz der Controller und Festplatten.

Die I/O-Scheduler scheinen recht selektiv in ihrer Wirkung. Bei manchen Controllern oder RAID-Konfigurationen bringen sie große Performance, bei anderen versagen sie regelrecht.

Offenbar gibt es zwischen den I/O-Schedulern des Linux-Kernels und den I/O-Subsystemen, also RAID-Controllern und Festplatten, unterschiedliche Auffassungen darüber, in welcher Reihenfolge die I/O-Zugriffe erfolgen sollen.

Abbildung 19 zeigt vereinfacht die Situation. Auf der linken Seite steht der Linux-Kernel, der die vier I/O-Scheduler zur Verfügung stellt. Der RAID-Controller (in der Mitte) verbindet das Betriebssystem mit den Festplatten (rechts). Letztere sind ebenfalls mit einem eigenen Scheduler bestückt. SCSI-Platten besitzen das als Tagged Command Queuing (TCQ) schon seit Jahrzehnten und SATA 2 hat dies nun auch als Native Command Queueing (NCQ) nachempfunden. Alle Messungen mit Hardware oder

Tabelle 5: Testergebnisse für Hard- und Software RAID 5

	Rate	Zugriffs-zeit	CPU-Eff.
Scheduler für 9650-4LP RAID 5			
Lesen (Seq.)	CFQ	CFQ	=
Schreiben (Seq.)	CFQ	=	=
Lesen (Rnd.)	=	=	CFQ
Schreiben (Rnd.)	=	CFQ	CFQ
Scheduler für Software RAID 5			
Lesen (Seq.)	CFQ	CFQ	=
Schreiben (Seq.)	=	=	=
Lesen (Rnd.)	Dead	=	=
Schreiben (Rnd.)	=	=	=
Hardware Raid 5 (H) vs. Software RAID 5 (S)			
Lesen (Seq.)	=	=	H
Schreiben (Seq.)	H	H	H
Lesen (Rnd.)	H	=	H
Schreiben (Rnd.)	H	=	H

Software RAID in diesem Artikel bisher benutzten NCQ, also doppeltes Scheduling: Zuerst wurden die I/O-Zugriffe vom Linux-Kernel reorganisiert, worauf sie der Controller/NCQ erneut organisiert hat. Der Kernel hat aber keine Information darüber, wie NCQ oder TCQ von den Festplatten-Herstellern implementiert ist.

Das Kommunikationsdefizit ...

Hier offenbart sich eine gewaltige Kommunikationslücke zwischen dem Betriebssystem, den Festplatten und deren Controllern. Das Betriebssystem weiß, welcher Block zu welchem Prozess gehört, es kann mit CFQ anhand von Nice-Level und Prozess-Scheduler optimale Bandbreitenzuordnungen für die I/O-Ströme der Prozesse ermitteln. NCQ hat von Prozessen und Nice-Leveln dagegen keine Ahnung.
Die Festplatte wiederum kennt die Blockaufteilung, also deren Low-Level-Formatierung bis aufs Bit. Weiterhin kennt NCQ den aktuellen Drehwinkel der Magnetscheiben und kann diesen in die Zugriffs-Optimierung einbeziehen, um die Average-Latency zu minimieren. Der **Kasten „NCQ unter Linux aktivieren/deaktivieren“** beschreibt, wie ein Admin unter Linux NCQ an- und ausschalten kann. **Abbildung 20** zeigt idealisiert die Möglichkeiten, die NCQ hat, um die I/O-Zugriffe zu optimieren.

... ließe sich mit offenen Protokollen beheben.

Einzig die Hersteller von RAID- oder HDD-Controllern hätten eine Chance, hier eine Mittlerrolle einzunehmen. Würden die Festplatten und das Betriebssystem befähigt, über den Controller Parameter für das Scheduling auszutauschen, dann könnte die Aussage „Doppelt hält besser“ auch beim Scheduling wahr werden. Die Controller-Produzenten wie auch die Festplatten-Hersteller stehen aber unter dem Druck, Hardware für Windows, Linux, Mac und Co.

Tabelle 6: HW Raid 5 (3HDDs) (H) vs. SW RAID 5 (4HDDs) (S)

	Rate	Zugriffs-zeit	CPU-Eff.
Lesen (Seq.)	S	S	H
Schreiben (Seq.)	=	=	H
Lesen (Rnd.)	S	S	H
Schreiben (Rnd.)	=	=	H

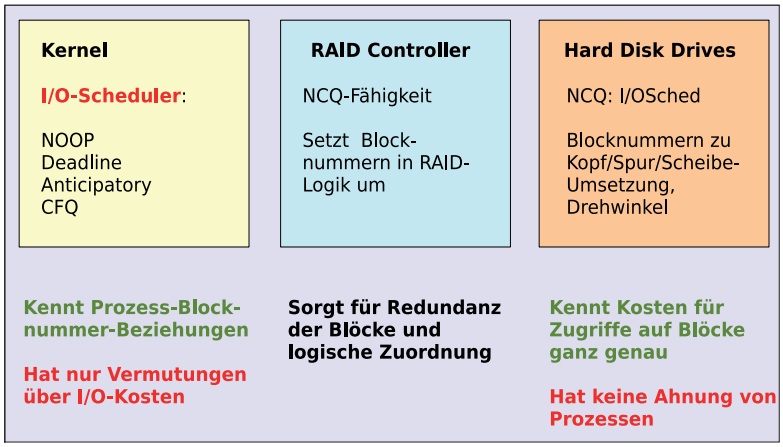


Abbildung 19: Die drei Komponenten eines I/O-Systems. Links der Linux Kernel mit seinen vier Schedulern. In der Mitte setzt der RAID-Controller die Daten für die Festplatten (rechts) um. Leider haben die beiden Enden dieser Kette keinerlei Wissen über die am anderen Ende auftretenden Kosten.

NCQ unter Linux aktivieren/deaktivieren.

Mit »dmesg |grep NCQ« lässt sich der Linux Admin anzeigen, ob der Treiber NCQ-fähige Festplatten erkannt hat:

```
$ dmesg | grep NCQ
sata_nv 0000:00:05.0: Using SWNCQ mode
ata1.00: 488397168 sectors, multi 16: LBA48 NCQ (depth 31/32)
ata2.00: 488397168 sectors, multi 16: LBA48 NCQ (depth 31/32)
```

In diesem Beispiel hat Linux zwei SATA-Platten mit NCQ erkannt und aktiviert. Die erste Zeile der Ausgabe gibt dabei den Treiberstatus aus:

```
$ dmesg | grep NCQ
ata1.00: 488397168 sectors, multi 16: LBA48 NCQ (depth 0/32)
ata2.00: 488397168 sectors, multi 16: LBA48 NCQ (depth 0/32)
```

In diesem Fall ist NCQ festplattenseitig zwar vorhanden, aber im Treiber nicht aktiviert. Hier kann Root versuchen, mit

```
echo 31 > /sys/block/<device>/device/queue_depth
```

das NCQ zu aktivieren.
Schlägt diese Operation mit der Meldung „Permission denied” fehl, so ist häufig der SATA-Kernel-Treiber das Problem.

NCQ bei 3ware-Controllern

Für 3ware-Controller bietet sich das Kommandozeilen-Tool »tw_cli« **(8)** an. Damit kann der Admin NCQ für jede RAID-Unit einzeln aktivieren oder deaktivieren, sofern der Controller und die Platten NCQ unterstützen. Der Befehl

```
/cx/uy set qpolicy=on/off
```

schaltet NCQ für Controller »x« und RAID-Unit »y« ein oder aus,

```
/cx/pz show NCQ
```

zeigt für Controller »x« und Device »z« den aktuellen NCQ-Status an.

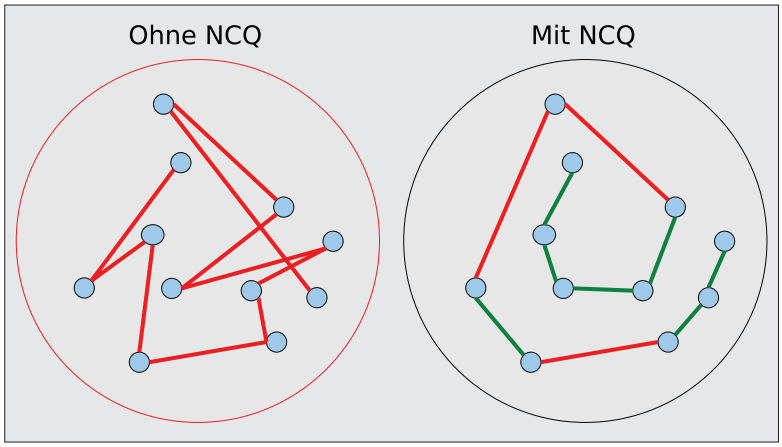


Abbildung 20: Nativ Command Queuing. Durch Einbeziehen der Rotationsposition der Magnetscheiben kann NCQ unnötige teure Zugriffe weitgehend eliminieren und so den Datenstrom beschleunigen.

anzubieten. Ohne offene plattformunabhängige Standards wird die Kommunkationslücke zwischen Festplatten und Kernel wohl nicht so bald geschlossen, und zwar für kein Betriebssystem.

I/O-Scheduling ohne NCQ

Was passiert wenn die Tester das NCQ ausschalten? Steigt oder sinkt die Performance? **Abbildung 21** stellt sequenzielle Leseraten mit und ohne NCQ gegenüber (vgl. **Abbildung 3**). Sofort springt ins Auge, dass die intelligenten Scheduler ihren Platz behaupten, die primitiveren Varianten aber stark zurückfallen. Die einfachen Scheduler profitieren also stark vom NCQ der Festplatten. Ohne NCQ sind die höherwertigen Scheduler vergleichsweise effektiver. Allerdings

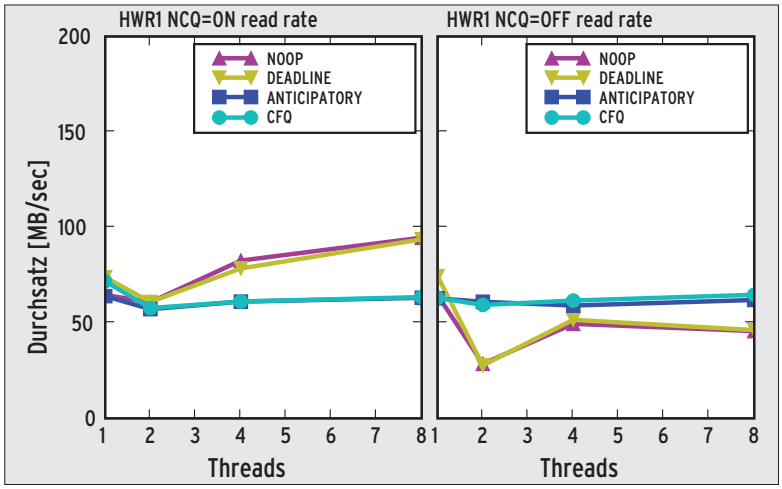


Abbildung 21: Sequenzielles Lesen von 1, 2, 4 und 8 parallelen Threads für die vier I/O-Scheduler bei Hardware RAID 1 (3ware 9650-4LP Hardware RAID 1) mit (links) und ohne Native Command Queuing (rechts).

schaffen Sie es nicht, die Leseraten auf das Niveau der Messungen mit eingeschaltetem NCQ zu heben.

CFQ liefert ab 2 Threads eine nahezu konstante Leserate von rund 65 MByte/s über alle Threads, unabhängig von NCQ. Bei Hardware RAID 1 erweist sich Deadline als performantester Scheduler.

Software RAID 1 profitiert weniger von NCQ

Sowohl Seagate als auch Western-Digital haben mit NCQ gute Arbeit geleistet. Mit einer einzigen Ausnahme profitiert jedes RAID 1 System vom NCQ der Festplatten (**Tabelle 7**).

Abbildung 22 zeigt die sequenzielle Leserate bei Software RAID 1 mit und ohne NCQ. Das Command Queuing scheint das Software RAID zu behindern: Ohne NCQ steigt die Leserate bei allen Schedulern an. **Tabelle 8** zeigt für den CFQ-Scheduler, welche NCQ-Einstellung bei einem Software RAID 1 die besten Ergebnisse liefern.

Es zeigt sich eine Tendenz, dass Software RAID 1 mit abgeschaltetem NCQ bessere Leistung erzielt. Die Unterschiede durch NCQ sind allerdings deutlich geringer als die von den I/O-Scheduler verursachten. Bei Software RAID 5 zeigt sich eine ähnliche, wenn auch schwächere Ausprägung des Einflusses von NCQ. Gleiches gilt für Hardware RAID 5.

Es lohnt sich also, Controller und Festplatten zu kaufen, die NCQ unterstützen. Allerdings sollte in jedem Einzelfall geprüft werden, ob die Abstimmung wirklich funktioniert und das

Tabelle 7: HW RAID 1 Deadline (NCQ=ON) vs. (NCQ=OFF)

	Rate	Zugriffs-zeit	CPU-Eff.
ST Festplatten (3ware 9650-4LP)			
Lesen (Seq.)	ON	ON	ON
Schreiben (Seq.)	=	=	ON
Lesen (Rnd.)	ON	ON	ON
Schreiben (Rnd.)	=	=	=
WD Festplatten (3ware 9650-2LP)			
Lesen (Seq.)	ON	ON	ON
Schreiben (Seq.)	=	=	ON
Lesen (Rnd.)	ON	ON	OFF
Schreiben (Rnd.)	ON	ON	=

gewünschte Ergebnis liefert. Für die Entscheidungsfindung hilfreich: Auf der Website der Autoren (9) finden sich noch detailliertere und weiterführende Testergebnisse und Diagramme zum Download.

Gemischte Gefühle

Die I/O-Scheduler von Linux liefern auf unterschiedlicher Controller-Hardware stark unterschiedliche Leistung. Erstaunlich gering ist hierbei der Einfluß der verwendeten Festplatten.

Aber selbst auf identischer Hardware spielt die RAID-Konfiguration eine entscheidende Rolle. Hardware RAID 1 und Hardware RAID 5 erfordern unterschiedliche I/O-Scheduler, Software RAID 1 zeigt sich Hardware RAID 1 ebenbürtig, und in der sequenziellen Leserate sogar überlegen. Bei RAID 5 haben jedoch die Hardware-RAID-Controller deutliche Vorteile gegenüber Software RAID 5. Dieser Vorsprung der Hardware-Variante kann aber schon durch eine weitere Festplatte im Software RAID 5 ausgeglichen werden.

Software RAID sollte also nicht pauschal vorverurteilt werden. Es ist eine gute Alternative bei der Projektierung gerade kostengünstiger Server-Lösungen. Im High-End-Bereich, wo jedes Quentchen CPU-Leistung gefordert ist, entlasten die Hardware-RAID-Controller die CPU und machen sich dort unverzichtbar. Dies gilt natürlich nur für echte Hardware-RAID-Controller mit eigener Intelligenz.

Gerade bei Onboard-RAID-Controllern lohnt es sich zu testen, ob sie wirklich die CPU entlasten wie die hier im Test gezeigten 3ware-Modelle. Als geeignetes Kriterium dafür bietet sich der Vergleich mit Software RAID im Punkt CPU-Effizienz an.

Native Command Queuing hat sich im Test sowohl bei Software wie bei Hardware RAID als effektive Technologie erwiesen. Die Festplatten von Western-Digital und Seagate zeigten überzeugend das Optimierungs-Potenzial von NCQ. Leider bleibt es jedem Administrator selbst überlassen herauszufinden, welcher I/O-Scheduler für seinen Controller, RAID-Level und NCQ-Einstellung (ON/OFF) die beste Leistung liefert. *(mfe)*

Tabelle 8: SW Raid 1, CFQ, (NCQ=ON) vs. (NCQ=OFF)

	Rate	Zugriffs-zeit	CPU-Eff.
Lesen (Seq.)	OFF	OFF	OFF
Schreiben (Seq.)	=	=	OFF
Lesen (Rnd.)	=	ON	ON
Schreiben (Rnd.)	=	=	ON

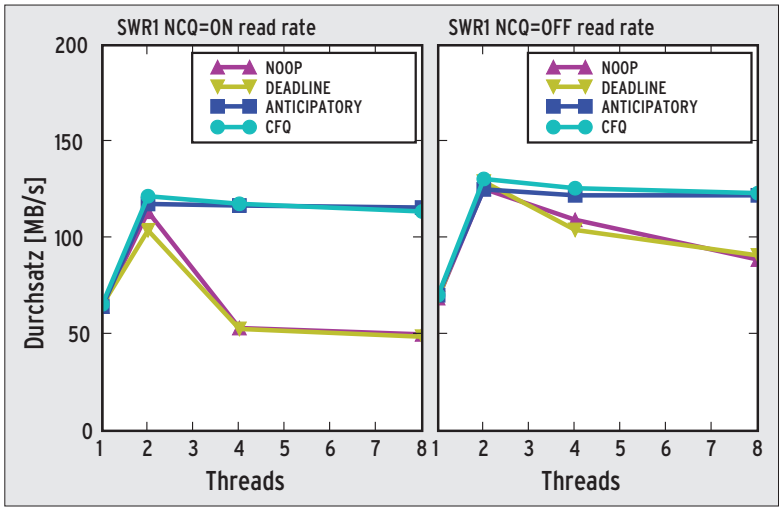


Abbildung 22: Sequenzielles Lesen von 1, 2, 4 und 8 Threads für die vier I/O-Scheduler bei Software RAID 1 mit (links) und ohne NCQ (rechts).

Infos

(1) IBM Applikations-I/O-Benchmark: (http://www.linuxinsight.com/ ols2004_workload_dependent_performance_evaluation_of_the_linux_2_6_i_o_schedulers.html)

(2) Linux-Magazin 03/2005, Kerntechnik 19: (http://www.linux-magazin.de/heft_abo/ausgaben/2005/03/kern_technik)

(3) Linux-Magazin 04/2005, Kerntechnik 20: (http://www.linux-magazin.de/heft_abo/ausgaben/2005/04/kern_technik)

(4) Chemnitzer Linuxtage 2008 Vortragsunterlagen: (http://chemnitzer.linux-tage.de/2008/vortraege/detail.html?idx=92#folien)

(5) TIOtest: (http://sourceforge.net/projects/tiobench)

(6) Storage Review Reference Guide zu Hard Disk Drives: (http://www.storagereview.com/guide2000/ref/hdd/index.html)

(7) 3ware SXU FAQ: (http://www.3ware.com/KB/article.aspx?id=15244)

(8) Tw_cli für 3ware Controller: (https://twiki.cern.ch/twiki/bin/view/FIOgroup/Tw_cli)

(9) Details der Performance Analysen bei Inqbus: (http://performance.inqbus.de)

Die Autoren

Martin Klapproth (links), Dr. Volker Jaenisch (Mitte) und Patrick Westphal (rechts). Dr. Volker Jaenisch ist Geschäftsführer der Inqbus IT-Consulting. Die Inqbus IT-Consulting beschäftigt sich mit Performance-Analyse und -Optimierung; sie projektiert und realisiert Web-, E-Mail- und Zope-HA-Server-Cluster. Darüber hinaus ist Volker Jaenisch tätig als Performance-Spezialist für wissenschaftliche EU-Projekte zur Analyse von Feinstaub-Messungen.

Martin Klapproth (links) und Patrick Westphal (rechts) studieren Informatik in Leipzig.