Lars Bremer

Tablebase-Karussell

Wie sich schnelle Festplatten auf Endspielanalysen auswirken

„Festplatten im Computerschach“, das klingt für Laien vielleicht ein bisschen wie „Schweine im Weltall“ – Experten jedoch wissen: Je schneller eine Festplatte ist, desto eher findet sich eine Engine in tablebase-nahen Stellungen zurecht, weil sie die umfangreichen Endspieldatenbanken erst von der Platte laden muss. Aber auf welche Platten-Parameter kommt es dabei genau an, welche sind zu vernachlässigen und wie wichtig ist die Größe des Tablebase-Caches?

Bereits in Ausgabe 4/03 ist CSS der Frage nachgegangen, welchen Einfluss die Festplatte auf die Lösezeiten in Endspielstellungen mit wenigen Steinen und damit auch die Spielstärke ausübt. Es stellte sich heraus, dass die Lösezeiten signifikant voneinander abweichen, je nachdem, auf welcher Art Laufwerk die Tablebases lagern. Damals blieben aber einige Fragen unbeantwortet: Welche Parameter sind bei ähnlich schnellen Platten entscheidend, bringt ein RAID 0 etwas, wie viel MByte Tablebase-Cache sollte man einstellen?

Um all diese Fragen zu klären, hat CSS in Zusammenarbeit mit der c‘t-Redaktion neue Messungen durchgeführt. Die c‘t stellte einen aktuellen Top-Rechner für die mehrere Tage dauernden Messungen zur Verfügung, einen Pentium4 3,2E mit 512 MByte schnellem RAM. Die Wahl der Test-Festplatte(n) fiel auf die Seagate ST3120023AS mit 7200 Umdrehungen pro Minute, 8 MByte Cache und 120 GByte Kapazität. Das Laufwerk schafft eine mittlere Dauertransferrate von 35 MByte/s.

Auf dem Test-Laufwerk, egal ob Einzelplatte oder RAID, haben wir eine 20 GByte große Partition eingerichtet, mit FAT32 formatiert und sämtliche Tablebases mit bis zu fünf Steinen darauf kopiert, um sicherzustellen, dass die Datenbanken auch wirklich im äußersten und mit rund 55 MByte/s schnellsten Bereich der Platte landen.

Bei der für alle Tests verwendeten Schach-Engine handelte es sich um ein Programm, das für Endspiel-Analytiker ohnehin erste Wahl ist: *Yace*. Für unsere Tests programmierte der Autor Dieter Bürßner eine Spezialversion, die alle möglichen Daten über erfolgte Tablebase-Zugriffe ausgibt. Yace lief ohne GUI, von einer Script-Datei gesteuert, mit 150 MByte Hashtables, Zugriff auf sämtliche 5-Steiner und musste sich mit fünf Endspielstellungen herumschlagen, darunter die drei bereits in CSS 4/03 verwendeten. Zwei weitere kamen hinzu, die (noch) mehr Zugriffe auf die Tablebases erfordern – die Messungen werden umso aussagekräftiger, je weniger Zeit das Programm mit seiner Baumsuche „vertrödelt“ und je mehr es in den Endspieldatenbanken herumkramt.

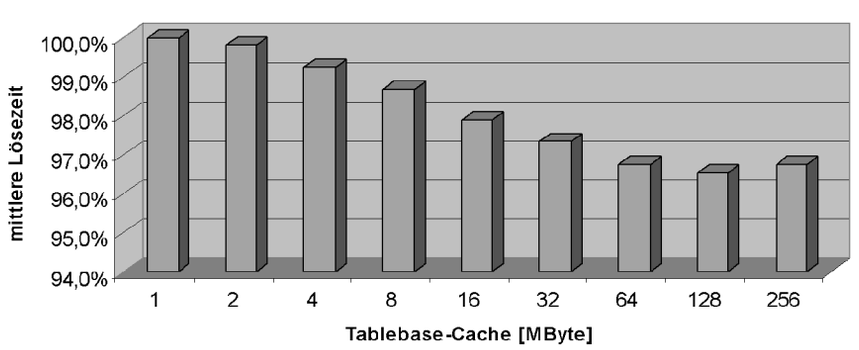
Tablebase-Cache

Die Frage, welche Größe der Tablebase-Cache haben sollte, wird immer wieder gestellt. Um diesen Punkt endgültig zu klären, musste Yace alle fünf Teststellungen mit neun verschiedenen Cache-Größen lösen, angefangen bei 1 MByte und jeweils verdoppelt bis zum Maximalwert 256 MByte. Ein Blick auf die Ergebnis-Tabellen zeigt, wie bedeutungslos der TB-Cache tatsächlich ist – der Lösezeiten-Unterschied zwischen minimalem und maximalem TB-Cache lag nur einmal bei rund acht Prozent, im Mittel bei dreieinhalb Prozent und ist manchmal auch überhaupt nicht zu messen. Es fällt auf, dass ab 64 MByte eine weitere Vergrößerung auf 128 oder 256 MByte nicht einmal mehr einen kleinen Vorteil ergibt – Diagramm 1 zeigt das geometrische Mittel der Lösezeiten bei den verschiedenen TB-Cache-Größen.

Voraussetzung für die Unwichtigkeit des TB-Caches ist aber, dass das Betriebssystem genug Speicher für seinen Cache übrig behält. Windows merkt sich nämlich einmal gelesene Daten in einem eigenen Cache-Speicher. Dessen Größe ist nicht konstant, sondern hängt von der Datenmenge ab – und natürlich vom freien Speicher. Yace hat die Datenmenge protokolliert, die *nicht* aus dem TB-Cache gelesen, sondern vom Betriebssystem angefordert wurde und demzufolge entweder aus dem Windows-Cache oder direkt von der Festplatte stammt. Der größte Unterschied liegt hier bei 417 MByte für die minimale Cache-Größe und 20 MByte bei 256 MByte Cache. Da die Lösezeiten trotz der zwanzigfachen Datenmenge nicht signifikant voneinander abweichen, müssen die meisten Daten schon im Speicher gelegen haben – der Beweis, dass das Betriebssystem, in dem die Tablebases wie auf der Platte komprimiert liegen, ebenso effizient zwischenspeichert wie der Nalimov-Cache, der zwar unkomprimierte Daten enthält, dafür jedoch eben deshalb auch schneller voll ist.

Ein weiterer Nagel im Sarg des Tablebase-Caches ist der Anteil der Tablebase-Zugriffe an der gesamten Lösezeit – es gibt hier keinen signifikanten Unterschied zwischen einem und 256 MByte. Egal also, ob der Anteil der physikalischen TB-Anforderungen bei 64 Prozent liegt oder bei 9 Prozent (wie in der Stellung in Tabelle 2), der Anteil der in der Suche aufgewendeten Zeit unterscheidet sich gerade einmal um zwei Prozent.

Diagramm 1 - Lösezeit mit verschiedenen TB-Cache-Größen



Zugriffszeit

Datenbankanwendungen, und dazu zählen Programme, die die TBs abfragen, profitieren normalerweise besonders von einer kurzen Zugriffszeit. Das ist ganz logisch, weil ein Programm nur selten vorhersehen kann, an welcher Stelle der nächste Zugriff auf die Datenbank erfolgen soll und daher raffinierte Cache-Strategien, die in Ruhephasen vorab schon Daten in den Speicher laden, weniger Erfolg versprechen.

Dies trifft auch auf die Tablebases zu. Sucht das Programm in einer TB-Datei nach einer bestimmten Stellung, berechnet es aus dieser Stellung einen Index, der die Position in der TB-Datei angibt, an der sich die Bewertung dazu findet. In den Tablebases sind also keineswegs Stellungen gespeichert, sondern nur Matt-Distanzen. Ein Beispiel: Das Schachprogramm möchte auf die Dateien für das Endspiel König und Bauer gegen König zugreifen.

Laut Indizierungsschema sind die Könige für die hohen Offsets zuständig, das heißt, ein kleiner Schritt des Königs (große kann er ohnehin nicht machen) führt zu einem Zugriff an einer ganz anderen Stelle in der TB-Datei; zieht nur der Bauer, ändert sich nur wenig an der Dateiposition. In Bauernendspielen ziehen die Könige aber am häufigsten, sodass das Indizierungsschema hier zu wild über die rund 30 MByte große Datei verteilten Zugriffen führen würde.

Ob es wirklich so schlimm ist, und ob die Zugriffszeit eine entscheidende Rolle spielt, lässt sich herausfinden, denn es gibt eine Möglichkeit, die Zugriffszeit direkt zu beeinflussen: das Akustik-Management einer Festplatte (siehe Kasten). Unsere Testplatte braucht in der schnellen Betriebsart im Mittel 11,7, in der leisen 12,9 Millisekunden, um einen zufälligen Sektor zu lesen. Die Auswirkung auf die Lösezeit war je nach Stellung verschieden; in den meisten Stellungen gab es kaum einen messbaren Unterschied, in einigen aber bis zu fünf Prozent – nicht die Welt. Der Grund, warum die kürzere Zugriffszeit nicht deutlich zu Buche schlägt: Die Zugriffszeit setzt sich aus zwei Komponenten zusammen: Die Zeit, die der Schreib-/Lesekopf benötigt, um die richtige Datenspur anzusteuern und der Zeit, die er dort auf den gewünschten Sektor warten muss. Die Wartezeit hängt nur von der Drehzahl der Platte ab, von nichts anderem; sie beträgt im Mittel die für eine halbe Umdrehung benötigte Zeit – bei 7200 U/min also ungefähr 8,3 ms.

Mit dem Akustik-Management ändert man nichts an der Drehzahl der Platte, nur an den Kopfbewegungen zwischen den Datenspuren. Moderne Platten bringen auf den Außenspuren bis zu 1000 Sektoren unter. Wenn die Tablebases schön sequenziell auf der Platte liegen, belegen sie nicht allzu viele verschiedene Datenspuren. Darum sind kaum Bewegungen des Schreib-/Lesekopfes nötig. Um das zu beweisen, haben wir, während Yace an einer Stellung rechnete, die Festplattenzugriffe mittels eines speziellen Kerneltreibers mitgeschnitten, und zwar direkt zwischen Windows-Treiber und IDE-Controller, also auf der untersten Ebene, die das Betriebssystem ermöglicht.

Eine Statistik der Tablebase-Zugriffe zeigt, dass immer Blöcke mit 4 oder 8 KByte angefordert werden. Dabei liegen nacheinander angeforderte Blöcke meist auf derselben Spur, landen also ohne Spurwechselzeit im Speicher.

Abstand zum vorigen Block	Anzahl der Zugriffe
direkt anschließend	2308
plus/minus 64 KByte	16249
plus/minus 128 KByte	530
plus/minus 256 KByte	660
weiter entfernt	4575
insgesamt	24322

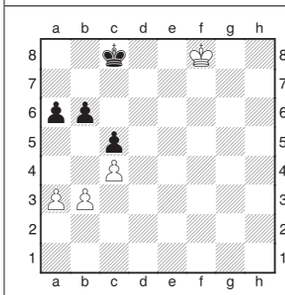
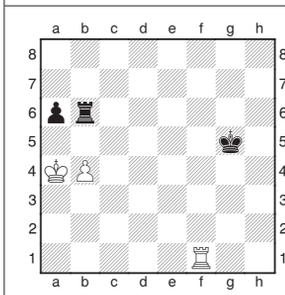
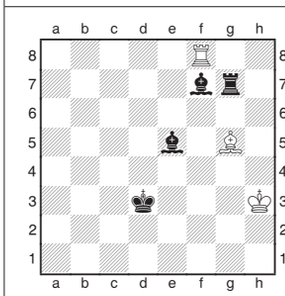
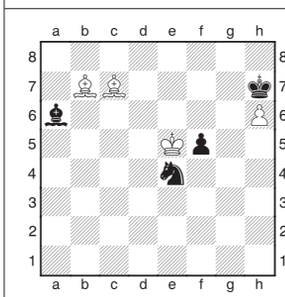
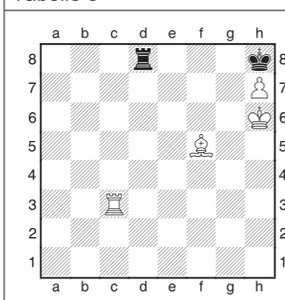
Von mehr als 24.000 Lesevorgängen liegen also beinahe 20.000 auf der Spur, von der auch der vorhergehende Zugriff Daten gelesen hat. Dazu kommt noch ein „read-ahead“ genannter Effekt: Wenn der Lesekopf die gesuchte Spur erreicht hat, liest er, während er auf den gewünschten Sektor wartet, ohnehin alle Daten ein, die unter ihm vorbeirotieren, und schreibt sie in den Plattencache. Das Gleiche geschieht, wenn er alle angeforderten Daten gelesen hat und nicht sofort eine neue Aufgabe wartet – die folgenden Sektoren landen dann ebenfalls „auf Verdacht“ im Plattencache. Jeder Zugriff „in der Nähe“ des vorhergehenden benötigt also kaum Zeit, weil die Platte ihn aus ihrem Cache-Speicher befriedigt.

Die von Herstellern und Computerzeitschriften angegebene Zugriffszeit ist für Schachprogramme also so wichtig wie die Gehäusefarbe des Rechners – mit anderen Worten: völlig irrelevant. Entscheidend sind vielmehr die Rotationsgeschwindigkeit und die Datendichte, weil es nur von diesen beiden Parametern abhängt, wie schnell eine Spur im Plattencache landet. Die Drehzahl einer neuen Platte ist leicht zu ermitteln, doch wie soll man etwas über die Datendichte herausfinden, zumal diese auch noch in Bits per Inch (BPI) und Tracks (Spuren) per Inch (TPI) zerfällt und nur die BPI für uns wichtig sind? Eine Faustregel: Schauen Sie auf die Transferrate, die hängt ebenfalls entscheidend vom BPI-Wert ab. Drehen zwei Platten gleich schnell, hat die mit der höheren Transferrate wahrscheinlich die größere Datendichte. In der Regel liegen die Daten auch umso dichter, je neuer ein Laufwerk ist. Nebenbei bemerkt ist es ohnehin eine gute Idee, die jeweils modernste Festplatte zu kaufen und keine quietschende alte vom Grabbeltisch.

RAID 0

Viele Motherboards enthalten schon einen RAID-Controller, mit dem man zwei Festplatten zu einem schnellen Verbund zusammenschalten kann. Bei den niedrigen Festplatten-Preisen mag mancher mit dem Gedanken spielen, oft benötigte und leicht ersetzbare Daten auf einem RAID 0 zu parken, immerhin liegt die Transferrate mit nur zwei IDE-Platten bei mehr als 95 MByte/s. Das macht sich im Alltag durchaus positiv bemerkbar, aber bringt es auch etwas für Tablebases? Schließlich bleibt die Zugriffszeit etwa dieselbe wie bei einer Einzelplatte. Und weil die Tablebase-Häppchen, die das Schachprogramm von der Platte fordert, immer nur 4 oder 8 KByte groß sind, liegen sie ohnehin nur auf einer der beiden RAID-Platten, die sich nur alle 64 KByte den Datenstrom teilen.

Trotzdem bringt das RAID, wie aus den Tabellen zu sehen ist, mehr Geschwindigkeit und kürzere Lösezeiten. Gemessen am Aufwand – zwei Platten! – allerdings nicht allzu viel, im Mittel fünf Prozent. Die Transferrate, die um mehr als 40 Prozent höher liegt, kann kaum schuld sein – die Steigerung wäre dann höher ausgefallen. Die Zugriffszeit aber ist bei einem RAID 0 ungefähr dieselbe wie die einer Einzelplatte. Grund ist wieder einmal die Cache-Strategie: Zwar liegen

Teststellungen und Ergebnisse				Einzelplatte schnell		Einzelplatte leise		RAID0	
	TB-Cache [Mbyte]	Anteil der phys. TB-Hits	gelesene TB-Daten [Mbyte]	Lösezeit [s]	Anteil der TB-Hits an der Lösezeit	Lösezeit [s]	Anteil der TB-Hits an der Lösezeit	Lösezeit [s]	Anteil der TB-Hits an der Lösezeit
Tabelle 1									
	1	69,9%	144,7	125	89,9%	136	90,3%	116	89,2%
	2	68,2%	141,2	125	90,0%	135	90,8%	117	89,2%
	4	65,9%	136,5	125	90,0%	135	90,8%	116	89,3%
	8	62,8%	129,5	124	90,0%	135	90,8%	116	89,3%
	16	58,0%	119,0	124	90,0%	135	90,8%	115	89,2%
	32	51,3%	104,6	124	90,0%	134	90,8%	115	89,3%
	64	41,8%	85,3	123	90,0%	134	90,8%	115	89,3%
	128	33,3%	68,6	123	90,0%	134	90,7%	115	89,2%
	256	31,2%	64,2	125	90,0%	135	90,8%	116	89,3%
Tabelle 2									
	1	64,4%	332,3	103	73,2%	108	74,3%	101	72,7%
	2	59,1%	305,0	103	73,1%	107	74,2%	100	72,5%
	4	52,9%	272,7	101	72,8%	106	74,0%	99	72,2%
	8	45,3%	233,5	100	72,5%	105	73,8%	98	71,9%
	16	36,6%	187,8	99	72,1%	104	73,4%	97	71,7%
	32	26,1%	133,6	97	71,8%	102	73,1%	96	71,3%
	64	16,4%	83,1	96	71,5%	100	72,7%	94	70,8%
	128	9,8%	49,8	95	71,2%	100	72,6%	93	70,5%
	256	8,7%	44,0	95	71,3%	100	72,6%	93	70,5%
Tabelle 3									
	1	48,3%	57,7	69	39,6%	70	40,7%	65	36,3%
	2	42,9%	51,4	69	39,6%	70	40,7%	65	35,9%
	4	37,4%	44,9	68	39,4%	70	40,6%	65	35,7%
	8	31,6%	38,1	68	39,2%	69	40,2%	64	35,3%
	16	25,1%	30,4	68	38,9%	69	40,1%	64	35,2%
	32	19,2%	23,4	68	38,7%	69	39,6%	64	35,0%
	64	14,0%	17,3	67	38,5%	69	39,4%	64	34,8%
	128	12,8%	15,8	67	38,5%	69	39,6%	63	34,7%
	256	12,8%	15,8	67	38,6%	69	39,8%	64	34,8%
Tabelle 4									
	1	54,0%	148,8	166	42,6%	173	44,8%	157	39,4%
	2	49,2%	134,7	166	42,6%	172	44,6%	157	39,2%
	4	44,1%	119,8	166	42,5%	172	44,6%	156	39,0%
	8	38,5%	103,9	165	42,3%	171	44,4%	156	38,8%
	16	31,9%	85,5	164	42,0%	171	44,3%	156	38,8%
	32	25,0%	66,7	164	41,9%	170	44,0%	155	38,5%
	64	18,0%	47,6	164	41,8%	170	43,9%	154	38,1%
	128	13,6%	36,1	163	41,7%	170	43,9%	154	38,2%
	256	13,5%	35,8	163	41,7%	169	43,8%	154	38,3%
Tabelle 5									
	1	45,3%	417,6	280	19,5%	284	20,6%	273	17,6%
	2	37,4%	345,8	278	18,9%	282	20,0%	272	17,1%
	4	28,9%	266,5	276	18,2%	279	19,4%	269	16,3%
	8	19,8%	181,3	273	17,3%	277	18,7%	266	15,5%
	16	11,2%	100,9	270	16,6%	274	17,9%	264	14,8%
	32	5,2%	46,0	268	16,1%	272	17,4%	262	14,2%
	64	2,8%	24,9	267	15,8%	271	17,2%	261	14,0%
	128	2,3%	20,6	267	15,8%	271	17,1%	260	13,8%
	256	2,3%	20,6	267	15,8%	271	17,1%	261	13,9%

die angeforderten Daten nur auf einer der beiden Platten, aber auf beiden bewegen sich die Köpfe zur fraglichen Spur und lesen Daten – sodass nach erfolgtem TB-Zugriff *zwei* Datenspuren in den Plattencaches lagern statt nur einem. Die doppelte Datenmenge im schnellen Zwischenspeicher, und das, wo doch die meisten TB-Zugriffe nicht allzu weit entfernt vom nächsten entfernt stattfinden. Es ist fast schon verwunderlich, dass die Lösezeit nicht noch kürzer ist.

Fazit

Eine schnelle Festplatte zahlt sich aus, das war schon im letzten Tablebase-Karussell klargeworden. Dabei sind nicht die reine Zugriffszeit wichtig und auch nicht die Dauertransferrate, sondern einzig und allein die Drehzahl und die Datendichte innerhalb einer Spur (natürlich hängen alle diese Parameter auch zusammen und beeinflussen sich gegenseitig). Das stimmt aber nur, wenn die Tablebases weitgehend unfragmentiert auf der Platte liegen. Defragmentierprogramme mögen in 99 Prozent aller Fälle so überflüssig sein wie ein Magnetstreifen im Kissen (man muss dran glauben, damit es wirkt!), die Endspieldatenbanken gehören zum restlichen Prozent.

Am besten ist es, sämtliche TB-Dateien auf eine frisch eingerichtete FAT32-Partition zu kopieren. Diese sollte idealerweise so weit außen auf der Platte wie möglich beginnen, was man am einfachsten dadurch erreicht, dass man sie möglichst frühzeitig anlegt, am besten gleich nach der Systempartition. Dies gilt insbesondere für die neuen 6-Steiner, die ohnehin schon auf mehrere Dateien verteilt daherkommen und damit per definitionem schon fragmentiert sind.

Die TBs auf einem RAID 0 zu lagern scheint wenig sinnvoll. Zwar steigert das die Leistung geringfügig, doch lange nicht so sehr, wie der doppelte Preis hoffen ließe – man benötigt immerhin zwei identische Festplatten. Wer wirklich maximale Leistung ohne Kompromisse möchte, sollte zu SCSI greifen. Ohrenschützer gibt es im Bundle.

Wie bedeutungslos der Tablebase-Cache wirklich ist, haben die Experimente deutlich gezeigt. Die Schlussfolgerung für Engine-Matches ist klar: Statt einen großen Nalimov-Cache zu spendieren, den nur jeweils eine Engine nutzen kann und den jede geladene Engine separat anfordert, ist der Speicherplatz besser angelegt, wenn er beim Betriebssystem bleibt. Schon bei nur einer Engine ist der Windows-Cache praktisch ebenso effektiv wie der Nalimov-Cache, bei zwei Engines ist er mit Abstand effizienter. Es ist also ungünstig, den gesamten Speicher für Hashtables zu verbraten, weil dann nicht genug für den Betriebssystem-Cache übrig bleibt.

Puristen, die während eines Engine-Matches nicht einmal die Maus zu bewegen wagen, könnten einwenden, dass es dadurch eine gegenseitige Beeinflussung der Engines gibt. Schließlich hängt der Zeitbedarf für einen Zug unter anderem davon ab, welche TB-Zugriffe die Gegner-Engine bereits getätigt hat, weil dadurch bestimmte Daten im Windows-Cache schon vorhanden sind. Wer eine so rigorose Einstellung hat, tut am besten daran, ganz ohne Tablebases zu spielen, denn ganz kann man den Filecache nicht abschalten.

Rund um Festplatten

Akustik-Management: IDE-Laufwerke kann man in einem *schnellen* und einem *leisen* Modus betreiben. In der leisen Betriebsart beschleunigt die Platte ihre Schreib-/Leseköpfe weniger stark und bremst sie auch sanfter wieder ab, was die Lautstärke des bei der Kopfpositionierung entstehenden Klickens deutlich reduziert. Auf das reine Laufgeräusch hat dies aber keinen Einfluss, und auch die Transferrate bei rein sequenziellem Zugriff bleibt gleich. Nur bei über die Platte verstreuten Zugriffen ist die Zugriffszeit länger und die Transferrate niedriger.

Cache: Schneller Zwischenspeicher direkt an der Plattenelektronik – beim Lesen von Daten als Read-Ahead-Cache wichtig. Dabei liest die Festplatte bei jeder Datenanforderung die dem gerade gelesenen Sektor folgenden auf Verdacht mit ein. Verlangt das Betriebssystem wenig später genau diese Daten resp. Sektoren, kann das Laufwerk sie ohne Zeitverlust liefern. Das ist auch der Grund, warum der Cache des Betriebssystems den Plattencache nicht ersetzen kann – Windows und Linux cachen nur Daten, die wirklich einmal benötigt wurden, während der Plattencache zu erraten versucht, welche Daten als Nächstes benötigt werden. Oft genug mit Erfolg.

***Drehzahl*:** Eine Festplatte arbeitet intern mit rotierenden Magnetscheiben, auf denen die Daten in Sektoren à 512 Byte, angeordnet in konzentrischen Ringen (Spuren oder Tracks genannt) abgelegt werden. Je schneller diese sich drehen, desto eher kommt ein angeforderter Sektor unter dem Schreib-/Lesekopf vorbei und desto kürzer ist demzufolge die Zugriffszeit.

Sektor: Kleinste logische Einheit auf der Festplatte, enthält 512 Byte Daten.

Spur, Datenspur, Track: Konzentrische Ringe auf der Festplattenoberfläche, die in Sektoren unterteilt sind. Weil die Magnetscheiben einer Festplatte außen mehr Platz bieten, sind die Spuren dort länger und können mehr Sektoren aufnehmen, also mehr Daten.

***Stripe Set, RAID 0*:** Zwei oder mehr zusammen geschaltete identische Festplatten, die vom RAID-Controller die Daten häppchenweise bekommen. Die Größe dieser Häppchen ist einstellbar, liegt jedoch bei den meisten Adaptern standardmäßig bei 64 KByte. Versucht das Betriebssystem also beispielsweise eine ein MByte große Datei zu schreiben, landen die ersten 64 KByte auf Platte eins. Während diese damit beschäftigt ist, die Bits in die Magnetscheibe zu meißeln, schickt der RAID-Controller die nächsten 64 KByte schon an Platte zwei im Verbund. Genau dasselbe passiert beim Lesen, sodass die Datentransferrate deutlich höher ist als bei einer einzelnen Platte. Zudem ist ein RAID 0 auch noch so groß wie die Summe der Kapazitäten der eingebundenen Platten. Die Sicherheit leidet jedoch, denn wenn eine Platte ausfällt, sind alle Daten futsch, auch die auf der intakten Platte. Nur ein Wahnsinniger würde darum sensible Daten oder das Betriebssystem auf ein RAID 0 ablegen.

Windows beherrscht auch andere RAID-Level, etwa RAID 5, ohne einen speziellen Controller zu benötigen. Um Tablebases zu beschleunigen, ist ein so genanntes Soft-RAID 5 aber nicht besonders geeignet, weil der Treiber im Unterschied zum RAID 0 vieles zu berechnen hat und damit dem Schachprogramm Rechenzeit stibitzt.

***Transferrate*:** Die Übertragungsrate in MByte/s, mit der im Mittel Daten von der Platte ins RAM geschaufelt werden können. Festplatten haben im äußeren Bereich mehr Sektoren, sodass die Transferrate in diesem Bereich am größten ist. Kommt es auf hohe Transferraten an, ist es eine gute Idee, die benötigten Daten im Außenbereich der Platte zu lagern. Viele Defragmentierprogramme, so sinnlos sie sonst sein mögen, bieten eine derartige Funktion. Auf einer neu angelegten FAT32-Partition landen die ersten geschriebenen Daten ganz außen. Gibt es mehrere Partitionen, liegt die zuerst angelegte im schnellsten Bereich der Platte.

***Zugriffszeit*:** Die im Mittel benötigte Zeit, um einen beliebigen Sektor einer Festplatte einzulesen. Sie ist abhängig von der Zeit, die der Schreib-/Lese-Kopf benötigt, um die gewünschte Datenspur anzufahren und der Zeit, die er dort auf den richtigen Sektor warten muss.