Die Sicht eines Sysadmins auf DB systeme

Robert Meyer

21. Oktober 2016

Inhaltsverzeichnis

1. Einleitung
2. IO unter Linux
   - typische Zugriffszeiten
   - Caching, das Prinzip
   - Caching
   - Ein kleines Beispiel
3. Transaktionen
   - Eigenschaften
4. Das Problem
   - Das Problem I
   - Das Problem II
5. Die Lösung
   - Die Lösung
   - Write Ahead Log
   - checkpoint
   - Zusammenfassung
6. RAID

Einleitung

Der Betrieb eines DB Systems ist tiemlich anspruchsvoll.
Meistens ist man auf ein performantes IO-Subsystem angewiesen, da oft viele Lese- und Schreiboperationen auftreten. Unglücklicherweise sind diese Operationen vergleichsweise langsam. Dies trifft insbesondere zu, wenn die Daten nicht sequentiell geschrieben werden, wie dies bei DB-Systemen meist der Fall ist.
Andererseits sind die Anforderungen an die Datenintegrität bei Datenbanken sehr hoch. Diese können vom Betriebssystem nicht so ohne weiteres geleistet werden.
Im Folgenden soll gezeigt werden, wie Postgresql versuch, diese beiden sich widersprechenden Anforderungen zu erfüllen, und woran man sonst noch so denken muss.

Zugriffszeiten im Vergleich

Access type	Actual time	Approximated time
1 CPU cycle	0.3 ns	1 s
Level 1 cache access	0.9 ns	3 s
Level 2 cache access	2.8 ns	9 s
Level 3 cache access	12.9 ns	43 s
Main memory access	120 ns	6 min
Solid-state disk I/O	50-150 us	2-6 days
Rotational disk I/O	1-10 ms	1-12 months

Tabelle: typische Zugriffszeiten. Aus https://madusudanan.com/blog/understanding-postgres-caching-in-depth/

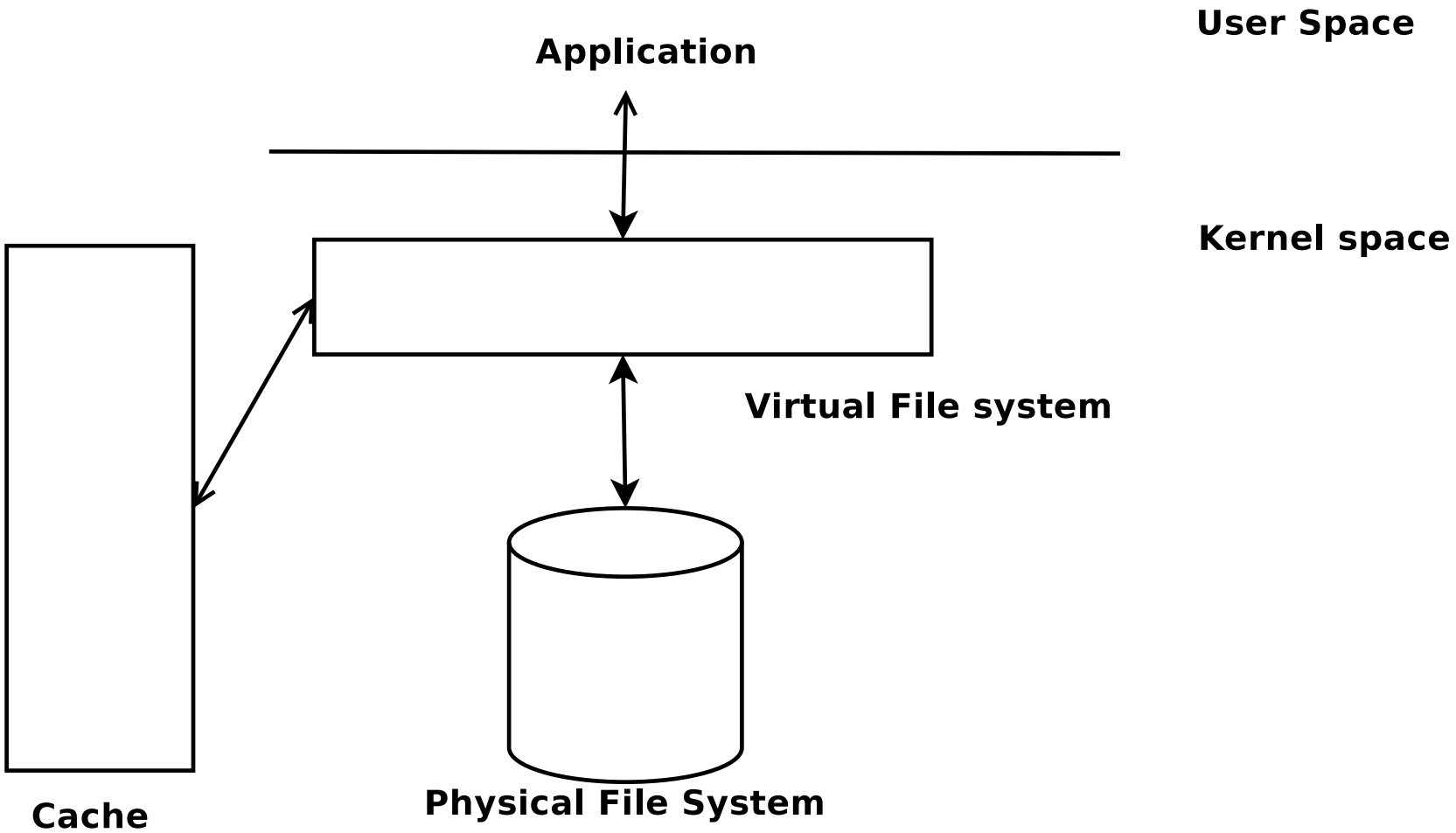


Abbildung: Caching

- Des Betriebssystem stellt ein virtuelles Dateisystem zur Verfügung.
- Die Applikation sieht nur diese Schicht.
- Beim öffnen einer Datei wird eine Kopie der Daten in den Arbeitsspeicher geladen (falls sie nicht bereits dort war).
- Änderungen werden nun im Arbeitsspeicher vorgenommen.
- Periodisch (oder beim unmounten) werden Bereiche, die sich geändert haben (dirty pages), auf die Disk geschrieben.
- Dabei wird der Eintrag entfernt, der am längsten nicht mehr benutzt wurde.
- Wird der Arbeitsspeicher knapp, wird automatisch Speicher aus dem cache zurückgegeben.
- Die Applikation weiss nicht, ob die Daten bereits auf die HDD geschrieben worden sind.
- http://www.linuxatemyram.com/

Ein kleines Beispiel

```
meyer@peregrin /media $ time dd if=/dev/zero of=/media/big_file bs=10M cou

100+0 records in
100+0 records out
1048576000 bytes (1,0 GB) copied, 0,495104 s, 2,1 GB/s

real    0m0,499s
user    0m0,001s
sys     0m0,495s
meyer@peregrin /media $ time sync

real    3m38,869s
user    0m0,000s
sys     0m0,005s
meyer@peregrin /media $
```

Abbildung: Ein kleines Beispiel

Eigenschaften

Die Anforderungen an Transaktionen in einer DB können folgendermassen zusammemgedasst werden:

- Atomic: Eine Transaktion wird entweder ganz durchgeführt, oder gar nicht.
- Consistent: Nach einer Transaktion ist die DB in einem konsistenten Zustand, wenn sie dies davor auch schon war.
- Isolated: Es ist irrelevant, ob Transaktionen gleichzeitig, oder nachheinander ausgeführt werden.
- Durable: Wenn eine Transaktion durchgeführt wurde, bleibt sie bestehen.

Das Problem I

Arbeitsspeicher ist grundstzlich flüchtig. Wenn inmitten einer Transaktion der Strom ausfällt, und keine Zeit mehr vorhanden war, die Daten auf die Festplatte zu schreiben, sind diese unwiederruflich verloren. Das widerspricht nun aber der Bedingung, dass Transaktionen atomar sein müssen. Man kann in einem derartigen Fall auch nicht davon ausgehen, dass die DB konsistent bleibt.
Schauen wir uns einen deratigen Fall enmal genauer an:

Das Problem II

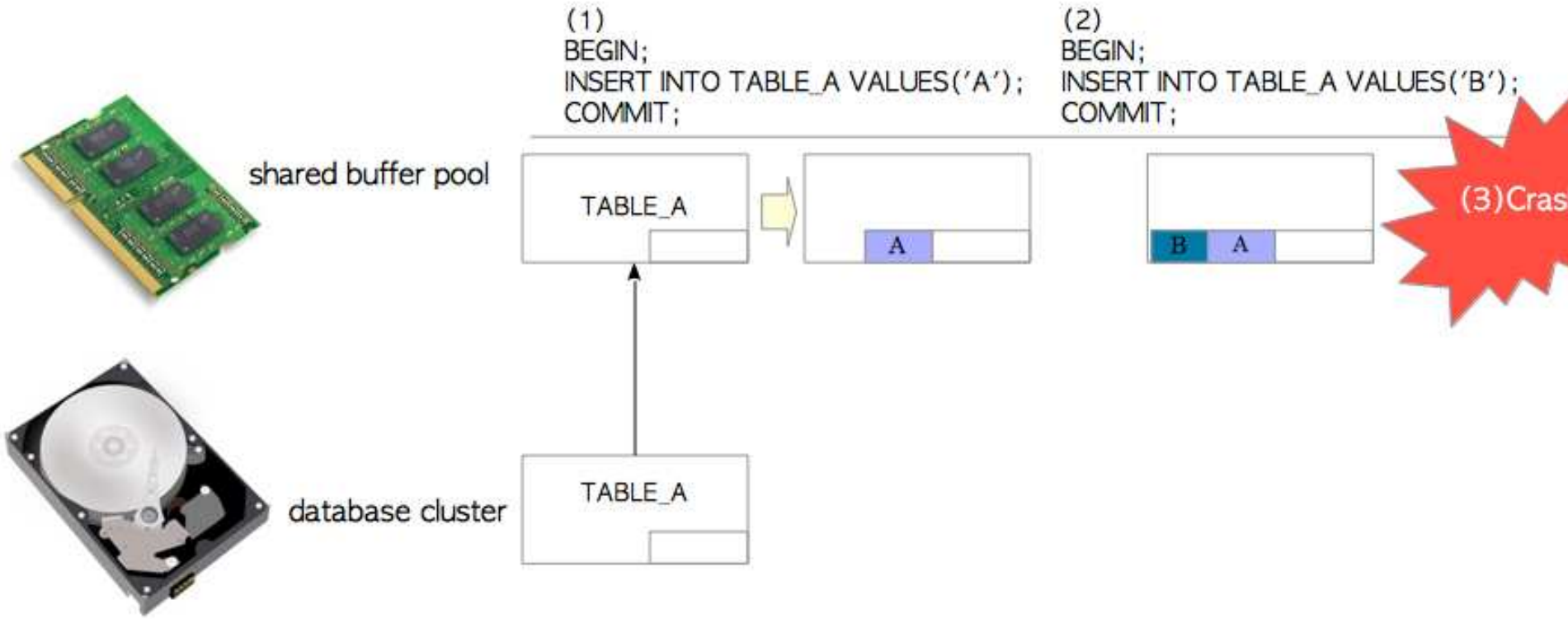


Abbildung: From: http://www.interdb.jp/pg/pgsql09.html

Die Lösung

Um die Bedingungen, die an Transaktionen gestellt werden, auch im Falle eines Systemausfalles zu garantieren, wird im Wesentlichen Buch geführt über alle Änderungen, die gemacht werden. Zudem wird sicher gestellt, das man immer wieder zum letzten konsistenten Zustand der DB zurück kann. Im Fehlerfall, werden dann alle Transaktionen seit dem letzten konsistenten Zustand wiederholt.

Write Ahead Log

Eine Transaktion besteht aus folgenden Schritten:

- Die Daten, die sich ndern sollen, werden in den Arbeitsspeicher geladen.
- Im Arbeitsspeicher wird die Änderung durchgeführt.
- Die Änderung wird in das WAL Log geschrieben.
- Die Datenbank gibt ein commit zurück
- Erst jetzt gilt die Transaktion als durchgeführt.

checkpoint

In periodischen Abständen läuft der sogenannte checkpoint Prozess im Hintergrund ab. Dabei geschieht folgendes:

- Schreibe einen checkpoint record in das Logfile, um wieder an diesen Zustand zurückkehren zu können.
- Schreibe alle dirty pages auf den Datenträger.

Man kann also davon ausgehen, dass die DB an jedem checkpoint konsistent ist.

Zusammenfassung

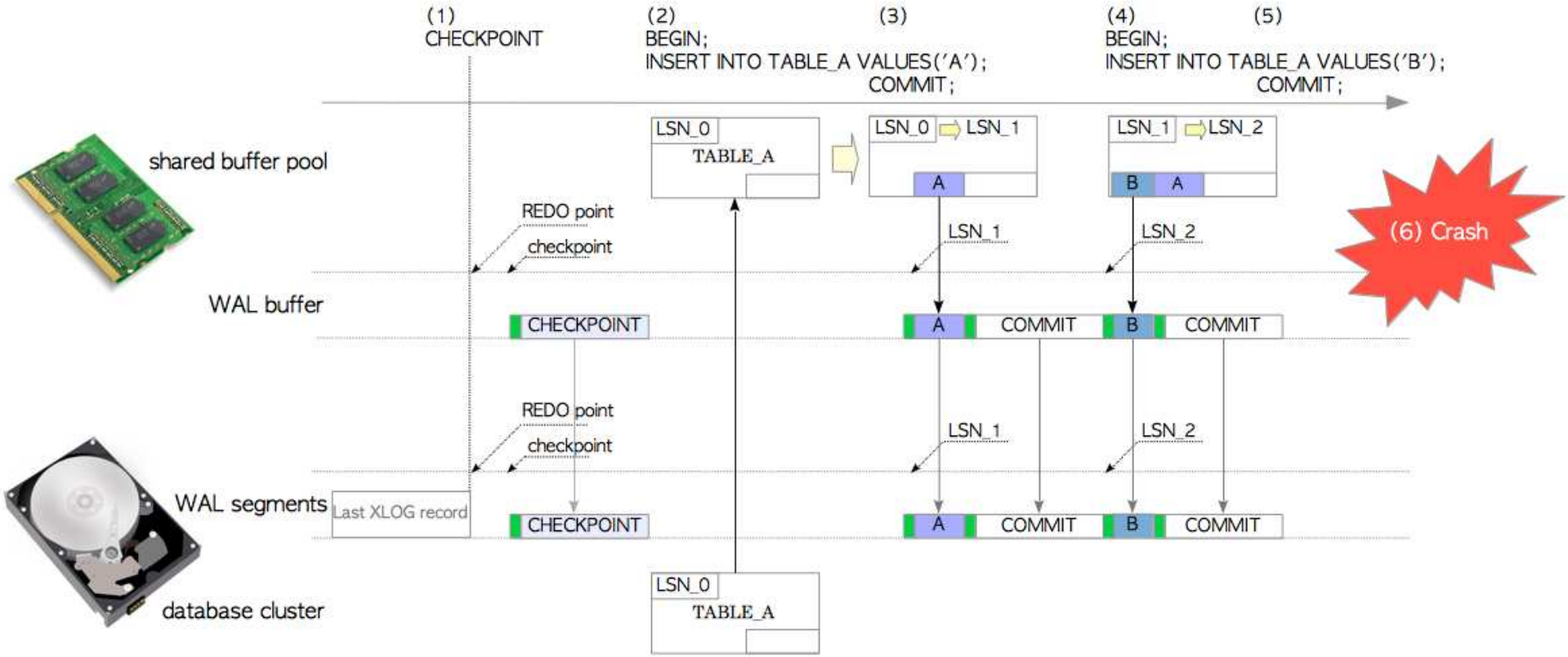


Abbildung: From: http://www.interdb.jp/pg/pgsql09.html

RAID

Um sich gegen den Ausfall einer HDD zu schützen, verbindet man am besten mehrere Disks zu einem RAID-Verbund.

- Redundant Array of Inexpensive Disks
- Hardware vs Software RAID
- RAID und Performanz

RAID 0

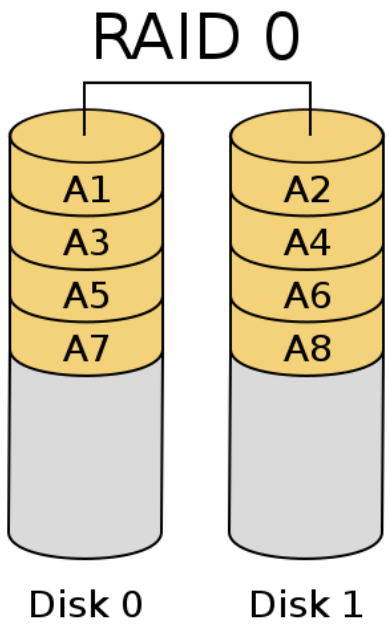


Abbildung: From Wikipedia

RAID 1

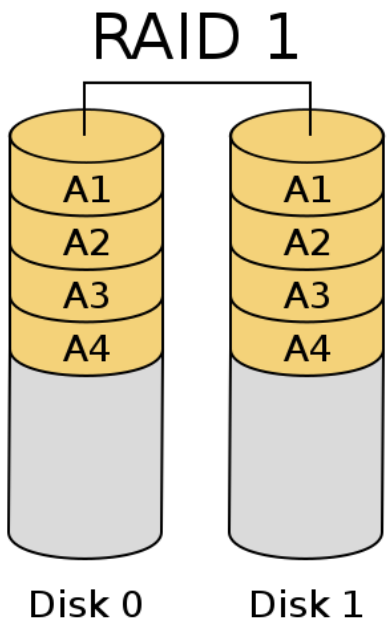


Abbildung: From Wikipedia

RAID 5

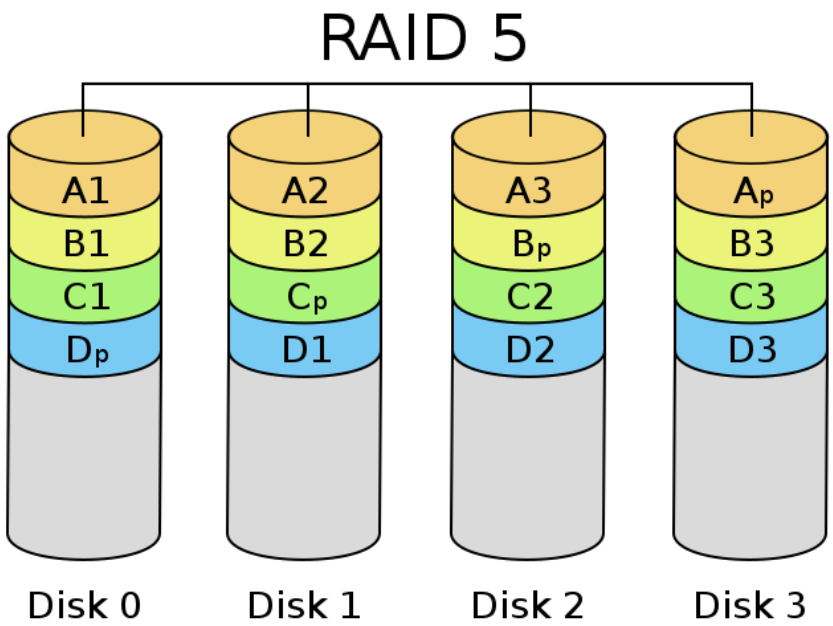
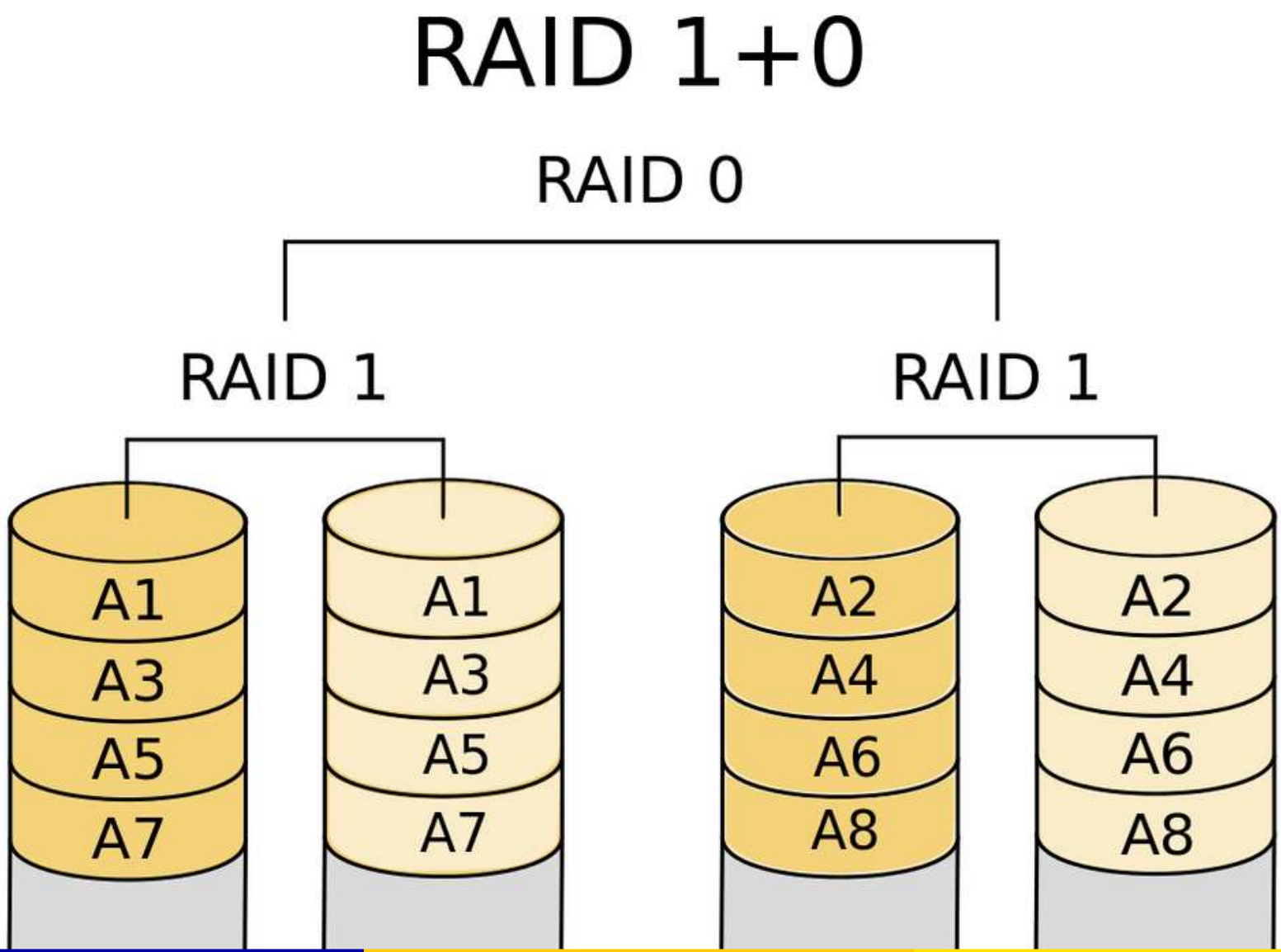


Abbildung: From Wikipedia

RAID 10


RAID 1+0
RAID 0
RAID 1
RAID 1
A1
A3
A5
A7
A1
A3
A5
A7
A2
A4
A6
A8
A2
A4
A6
A8

Backup

- Ein redundantes Backend ist kein Ersatz für Backups.
- Im Fall von Fehlmanipulationen bringt die Redundanz nichts.
- Man sollte tägliche Backups machen.
- Dabei sollte man immer die tools der DB verwenden