PM1/PM7

Hard Disk Power Management: Implementierung eines Timeout–Mechanismus

In dem PC `i30pm1` bzw. `i30pm7` ist eine Notebook–Festplatte eingebaut; in Linux erscheint sie als `/dev/hdc`. Als erster Schritt soll der „DDT–Algorithmus“ (*device–dependent timeout*) implementiert werden:

DDT-Algorithmus
Festplatte in den Standby–Modus versetzen, falls
$$t_{la} + t_{be} \leq t$$

Die Variablen bedeuten:
- t : die aktuelle Zeit
- t_{la} : wann wurde zuletzt auf die Festplatte zugegriffen
- t_{be} : die break–even–Zeit

Die break–even–Zeit soll über das `/proc`-Dateisystem konfigurierbar sein (implementiert im IDE-Treiber: `drivers/ide/ide.c` und `ide-proc.c`).

Dazu soll der Kernel–Thread `ide_power_task` geschrieben werden, der die Moduswechsel der Festplatte kontrolliert. Dieser Thread wird aufgeweckt (z.B. über einen Mutex), wenn die Festplatte aktiviert oder deaktiviert werden soll oder sich der Modus der Festplatte ändert (z.B. durch einen Reset). Die Struktur `ide_drive_t` (`include/linux/ide.h`) soll dazu um folgende Informationen erweitert werden:

```
unsigned long break_even_period;
unsigned long last_access;
int power_mode;     /* 0=standby, 1=running */
int new_power_mode; /* 0=standby, 1=running, 2=changed, 3=no_change */
```

Die Variable `power_mode` stellt den Zustand der Festplatte dar (es wird nur zwischen `standby` und `running` unterschieden); `new_power_mode` ist der neue Zustand bei einem Moduswechsel. Bei einem Zugriff auf die Festplatte (IDE-Treiber) soll die Variable `last_access` aktualisiert werden. Über einen Timer soll einmal die Sekunde die Funktion `ide_check_for_standby()` aufgerufen werden, die die aktuelle Zeit mit (`last_access + break_even_period`) vergleicht und bei einer Überschreitung die Festplatte in den Standby–Modus versetzt (`new_power_mode` auf 0 setzen, `ide_power_task` aufwecken). In manchen Situationen ist der Zustand der Festplatte unbekannt. Dann soll der `ide_power_`

`task` mit dem Modus `changed` aufgeweckt werden und selbst herausfinden, in welchem Modus sich die Festplatte befindet.

Seiten, die den Inhalt von Disk–Blöcken enthalten und modifiziert wurden (als *dirty* markiert), werden vom *pdflush*–Dämon auf die Platte zurückgeschrieben (implementiert in in `mm/pdflush.c` und `mm/page-writeback.c`). Der Dämon überprüft alle 5 Sekunden (*dirty writeback centisecs*), ob modifizierte Seiten vorliegen und schreibt diese auf Platte, sobald sie mindestens 30 Sekunden alt sind (*dirty expire centisecs*). Damit soll vermieden werden, dass bei kurz aufeinanderfolgenden Schreibzugriffen jedesmal auf die Platte zugegriffen wird. Die Zeiten können unter `/proc/sys/vm/dirty_expire_centisecs` bzw. `dirty_writeback_centisecs` geändert werden.

Dieser Mechanismus verfügt über einen *laptop mode*, welcher einen energiegewahren Update–Prozess implementiert. Dieser *laptop mode* soll so ergänzt werden, dass vor einem Herunterfahren der Platte sämtliche als *dirty* markierte Blöcke auf die Platte geschrieben werden. Damit soll ein Wiederanfahren der Platte durch den Update–Prozess vermieden werden.

Bei regelmäßigen Festplattenzugriffen mit ungefähr gleicher Größe bzw. Dauer (z.B. Audioplayer) kann der Algorithmus noch verbessert werden, indem nach einem Datentransfer sofort in den Standby–Modus gewechselt wird (und nicht gewartet wird, bis die break–even–Zeit abgelaufen ist). Der Algorithmus soll dann zum DDT/ES–Algorithmus (device–dependent timeout with early shutdown) erweitert werden:

DDT/ES-Algorithmus

Festplatte in den Standby-Modus setzen, falls

$$t_{la} + t_{be} \leq t$$

oder

$$t_{fa} + t_{lb} \leq t \leq t_{fa} + t_{lb} + t_1 \quad \text{und} \quad t_{la} + t_2 \leq t$$

Die Variablen bedeuten:

- t : die aktuelle Zeit
- t_{lb} : die Länge der letzten aktiven Phase (Dauer eines I/O-Transfers, `busy_period`)
- t_{fa} : der Zeitpunkt des ersten Zugriffs in der gerade aktiven Phase (`first_access`)
- t_{la} : wann wurde zuletzt auf die Festplatte zugegriffen
- t_{be} : die break-even-Zeit
- t_1 : Toleranzwert beim Vergleich des letzten mit dem aktuellen Interval (2 Sekunden)
- t_2 : kleiner Timeout, um das Ende der aktiven Phase zu erkennen (1 Sekunde)

Als *busy period* wird die Dauer zusammengehörender Festplattenzugriffe verstanden; sie beginnt und endet mit einem Festplattenzugriff. Dazu muss in der Struktur `ide_drive_t` (`include/linux/ide.h`) noch um folgende Variablen ergänzt werden:

```
unsigned long busy_period;
unsigned long first_access;
```

Die Funktion `ide_check_for_standby` muss nun entsprechend modifiziert werden.