Zcache

Andor Daam, Stefan Hengelein, Florian Schmaus

Friedrich-Alexander Universität Erlangen-Nürnberg

Einführung

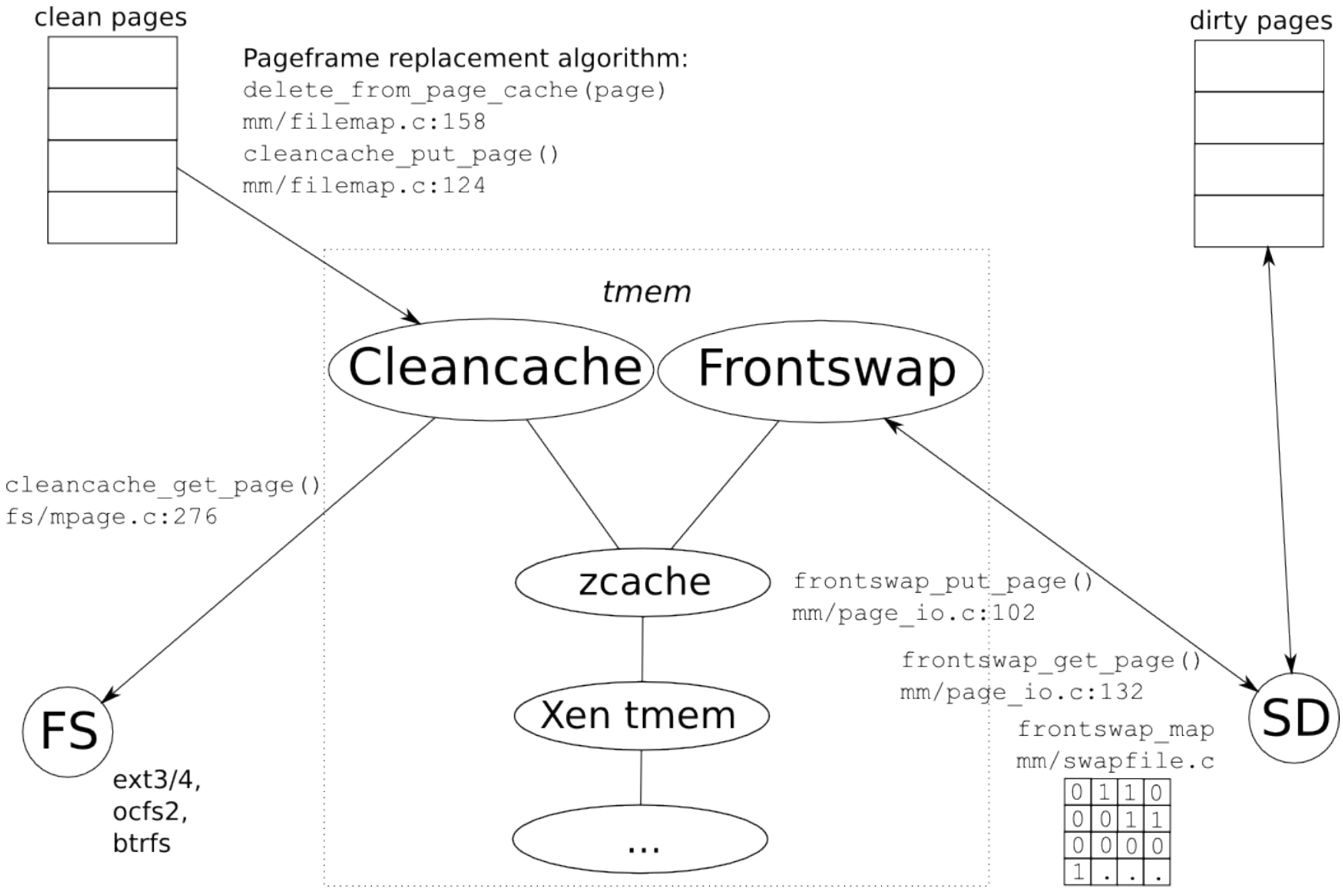
Zcache

- realisiert komprimierte Speicherseiten im RAM
- Backend für Cleancache und Frontswap
- Seiten die normalerweise verdrängt würden, werden komprimiert $\Rightarrow$ weniger Speicherbedarf, Aus- und Wiedereinlagern schneller (keine I/O-Operation)

Einführung

- Zcache ist derzeit noch im Stagingbereich. Ein Grund dafür ist, dass er nicht als Modul (ent)ladbar ist.
- Man tauscht durch die (De-)Komprimierung I/O-Zugriffszeit gegen CPU-Rechenzeit.
- Es stellt sich also die Frage, ob durch Nutzung von Zcache mehr Strom verbraucht wird.

Einführung


clean pages
Pageframe replacement algorithm:
delete_from_page_cache(page)
mm/filemap.c:158
cleancache_put_page()
mm/filemap.c:124
dirty pages
tmem
Cleancache
Frontswap
cleancache_get_page()
fs/mpage.c:276
zcache
frontswap_put_page()
mm/page_io.c:102
frontswap_get_page()
mm/page_io.c:132
Xen tmem
frontswap_map
mm/swapfile.c
FS
ext3/4,
ocfs2,
btrfs
...
SD
0 1 1 0
0 0 1 1
0 0 0 0
1 . . .

Einführung

Unsere angestrebten Ziele waren:

- dynamisches An- und Ausschalten der Komprimierung über sysfs (evtl. auch automatisch für Notebooks)
- Messen des Stromverbrauchs mit und ohne Zcache
- dynamisches Nachladen des Zcache-Backends (insmod) erlauben
- wenn möglich: Entladen von Zcache (rmmod) erlauben

Kernelpatches

- Add r/w sysfs attribute for zcache_freeze [1]
- Added counters for compression and decompression [2]
- Added a missing newline at the end of zbud_unbuddied_list_counts statistic sysfs entry [3]
- cleancache and frontswap: allow backends to register after cleancache init [4] [5]

[1] http://driverdev.linuxdriverproject.org/pipermail/devel/2012-March/025034.html

[2] http://driverdev.linuxdriverproject.org/pipermail/devel/2012-March/025035.html

[3] http://driverdev.linuxdriverproject.org/pipermail/devel/2012-March/025032.html

[4] http://marc.info/?l=linux-mm&m=133174533205576&w=4

[5] http://marc.info/?l=linux-mm&m=133174533405587&w=4

sysfs-node

- Ein Patch der die Komprimierung von Seiten ausschaltet, wurde an die entsprechende Mailingliste geschickt und wartet auf Validierung.
- Das Ausschalten der Dekomprimierung ist komplizierter, weil dafür die vorliegenden Seiten invalidiert werden müssten. Dies ist insbesondere für "dirty pages"(Frontswap) schwer, da diese nicht einfach verworfen werden können.

insmod

- Um das Nachladen von Backends für Cleancache und Frontswap zu erlauben, wurde ein Patch an die Mailingliste geschickt.
- Das Entladen zu realisieren trifft auf die gleichen Probleme wie das Ausschalten der Dekomprimierung durch einen sysfs-Eintrag. Die Seiten im Frontswap können nicht einfach gelöscht werden.
- Resultat: insmod für Backends ist nun möglich. Vorher mussten sich Backends registrieren, bevor Dateisysteme eingehängt oder Swap-Devices aktiviert wurden.

Herausforderungen

Erste Herausforderung: Den Kernel auf den Testsystemen zum Laufen zu bekommen.
Zweite Herausforderung: Können wir eine Verbesserung im Durchsatz, die bei Nutzung von Zcache gezeigt wurde, reproduzieren?
Dritte Herausforderung: Können wir eine positive oder negative Auswirkung auf die Akkulaufzeit messen?

Benchmark und Evalutionsframework

Drei Testsysteme mit aktuellem Linux-next, Kernel 3.3 32 Bit / 64 Bit

- EEEPC 1005PX, CPU: Intel Atom N450(2x1,6 GHz), 1 GB RAM
- Pentium 4 Dualcore, 3,2 GHz, 1 GB RAM
- SubNotebook, Intel-i5 Prozessor, 4 GB RAM

Benchmark und Evalutionsframework

Komponenten:

- C Helper
  - `malloc [-s mbytes] [-r rnd_read_count] [-w wait_secs]`
  - `mmap [-s mbytes] [-r rnd_read_count] [-w wait_secs] [-f file]` (`PROT_READ`)
- Bash Scripts
  - `function.sh` Liest `sysfs`- und `debugfs`-Werte aus und erstellt Statistiken
  - `iozone.sh` startet iozone mit Profilen aus `iozone_profiles.sh`
  - `kernel-unpack-compile.sh` startet einen Kernel-Compile-Lauf mit n Iterationen

Zeitmessung Pentium 4 Rechner

Kernelcompile Test

mittlere Laufzeit bei Acht Testläufen

- ohne zcache: 1558s
- mit zcache: 1541s

Iozone Automatic Test

gemittelte Dauer pro Test 16389sec (∼4,5 Std) bei 6 Testläufen

zcache	write	read	rnd read	bkwd read	rec. rewrite	stride read
aus	40347	31969	12517	24349	19750	17628
an	43315	33350	13264	25959	20987	19234
Ratio:	1.074	1.043	1.060	1.066	1.063	1.091

- Laufzeit Verbesserung nur marginal
- Durchsatz Verbesserung ebenso kaum erkennbar

Allerdings....

5 Testläufe ohne zcache und nur 1 Testlauf mit zcache..... wieso?

KB	reclen	write	rewrite	read	reread	random read	random write	bkwd read	record rewrite
972800	4	45497	19084	46791	48122				

Error in file: Found ?5c0405055c040505? Expecting ?3a3a3a3a3a3a3a3a? addr b6400000
Error in file: Position 72327168
Record # 17658 Record size 4 kb
where b6400000 loop 0

Allerdings...

Testet man mit den richtigen Einstellungen

Statistik	zcache aus	zcache an	Speicher
compressed_pages	0	4044384	15798 MB
decompressed_pages	0	1836987	7175 MB
cc puts	9612914	4044384	-
cc succ_gets	0	1837062	7176 MB
cc failed_get	9678269	2359034	-
time	5469 sec	2766 sec	-

Frontswap hat hier garnicht gearbeitet...

Allerdings...

Testet man mit den richtigen Einstellungen

Statistik	zcache aus	zcache an	Speicher
compressed_pages	0	4044384	15798 MB
decompressed_pages	0	1836987	7175 MB
cc puts	9612914	4044384	-
cc succ_gets	0	1837062	7176 MB
cc failed_get	9678269	2359034	-
time	5469 sec	2766 sec	-

Frontswap hat hier garnicht gearbeitet...

Was wurde getestet?

- write / rewrite
- random read / write
- Wortlängen 4kb / 1024kb wobei insgesamt 972MB gelesen / geschrieben wurden

Stromverbrauch- und Zeitmessung Subnotebook

System

X220, i5-2520M (4 Kerne), 4 GB RAM, SSD

Benchmark

Kernel entpacken und `make defconfig && make -j4`

Ergebnis

zcache	off	on
mem free (start)	753MB	218MB
mem free (end)	826MB	762MB
time	264s	265s
compressed pages #	0	172829
decompressed pages #	504	7161
mAh/s	-1162	-1169
swap used (start)	697MB	0MB
swap used (end)	746MB	697MB

170000 pages $\sim$ 670 MB

Stromverbrauch- und Zeitmessung Subnotebook

Benchmark

`iozone -M -B -r 4 -r 1024 -i 0 -i 1`

3 × `malloc -s 1024 -w 60`

Ergebnis

zcache	off	on
comp pages #	0	337588
decomp pages #	578551	342747
time	415	446
mAh/s	730	683
swap used (start)	1444MB	136MB
swap used (end)	1816MB	1444MB

600000 pages ∼ 2340MB

Stromverbrauch- und Zeitmessungen EEE PC

iozone mit Satzlängen: 8, 128, 1024, 4096 KB; re-/write; re-/read; bkwd read

1 GB Daten

zcache	aus	an	+%
Laufzeit (sek)	2024	2480,66	22,6
komp. Seiten	0	5308806	–
dekomp. Seiten	0	98	–
Strom (mAh/s)	237,66	256	8,5

1,2 GB Daten

zcache	aus	an	+%
Laufzeit (sek)	2316	3042	31,3
komp. Seiten	0	6703150	–
dekomp. Seiten	0	2077	–
Strom (mAh/s)	310	343	10,6

Stromverbrauch- und Zeitmessungen EEE PC

- da keine Seiten aus dem zcache geholt werden, ist kein Speedup durch Umgehen von I/O-Operationen bemerkbar (ist sogar langsamer)
- Batterieverbrauch (pro Sekunde) steigt jedoch um $\sim$10% an
- In einem solchen Anwendungsszenario wäre es sinnvoll, zcache_freeze vorab zu aktivieren

Fazit

- Zcache ist besonders vorteilhaft bei Systemen mit unvorteilhaften CPU/RAM Verhältnis (schnelle und viele Kerne, wenig RAM)
- Bei schwachen Systemen mit wenig RAM (z.B. Netbooks), kann sich Zcache negativ auf die Akkulaufzeit/Stromverbrauch auswirken
  - Vor allem bei Anwendungsfällen, in denen Seiten hauptsächlich ausgelagert werden

Vielen Dank für die Aufmerksamkeit