SWAP: czym jest, kiedy czyścić, kiedy nie ruszać

Poziom: średnio **Dotyczy: wszystkie dystrybucje** **Czas: ~18 min**

SWAP to przestrzeń na dysku, do której jądro wypycha rzadziej używane strony pamięci, gdy RAM jest ciasny. Ta strona tłumaczy, kiedy wysoki swap jest zupełnie normalny, kiedy jest objawem realnego problemu — i dlaczego dwa najpopularniejsze „lekarstwa" (`swapoff -a` i `drop_caches`) potrafią zrobić więcej szkody niż pożytku.

Skróty użyte na tej stronie (5)

Skrót	Co znaczy
GB	Gigabyte: miliard bajtów, jednostka pojemności pamięci i plików, tu najczęściej rozmiar modelu albo pamięci karty graficznej
GiB	gibibajt: 1024 MiB, czyli około 1,07 GB; tej jednostki używa dokumentacja MailCow przy wymaganiach pamięci
OOM	Out Of Memory: brak pamięci; OOM killer to mechanizm jądra zabijający proces, gdy jej zabraknie
RAM	Random Access Memory: pamięć operacyjna
WMS	Warehouse Management System: system zarządzania magazynem

Fiszka

TYLKO ODCZYT

Wypisanie aktywnych partycji i plików swap: swapon -s

```
swapon -s
```

Starszy, ale wciąż powszechny sposób sprawdzenia, co system używa jako swap. Format jest zwięzły — nazwa urządzenia, typ, rozmiar i aktualne zużycie w kilobajtach.

Przykład:

```
Filename                                Type            Size            Used
Priority
/dev/dm-1                               partition       4194300         524288
-2
```

swapon = włącz wskazaną partycję albo plik jako przestrzeń wymiany

TYLKO ODCZYT

Wypisanie aktywnego swapu w czytelniejszym formacie: swapon --show

```
swapon --show
```

Ten sam efekt co `-s`, ale w formacie tabeli z jednostkami czytelnymi dla człowieka i bez konieczności dzielenia kilobajtów w głowie.

Przykład:

```
NAME      TYPE      SIZE USED PRIO
/dev/dm-1 partition   4G 512M   -2
```

swapon = włącz wskazaną partycję albo plik jako przestrzeń wymiany

TYLKO ODCZYT

Sprawdzenie swapu w kontekście całej pamięci: free -h

```
free -h
```

Pełny opis kolumn jest na stronie o `free`; tutaj liczy się druga linia wyniku.

Przykład:

```
               total        used        free      shared  buff/cache   available
Mem:            15Gi       8.2Gi       2.1Gi       512Mi       5.3Gi       7.4Gi
Swap:          4.0Gi       512Mi       3.5Gi
```

free = pokaż ile pamięci RAM i przestrzeni wymiany jest wolne, a ile zajęte

□□ TYLKO ODCZYT

Wypisanie surowych danych o swapie: cat /proc/swaps

```
cat /proc/swaps
```

To jest plik, z którego `swapon -s` czyta swoje dane — bezpośrednie źródło, przydatne w skryptach, które nie chcą parsować wyjścia narzędzia.

Przykład:

```
Filename□□□□Type□□Size□□Used□□Priority
/dev/dm-1                                 partition□4194300□□524288□□-2
```

□□ **cat** = concatenate: sklej i wypisz pliki

□□ TYLKO ODCZYT

Wyłączenie całego swapu: swapoff -a

```
swapoff -a
```

Wyłącza wszystkie aktywne obszary swap naraz. **To polecenie przenosi z powrotem do RAM-u każdą stronę pamięci, która w tej chwili leży w swapie** — patrz ostrzeżenie w sekcji *Koniecznie* niżej, zanim je uruchomimy na czymkolwiek produkcyjnym.

□□ TYLKO ODCZYT

Ponowne włączenie swapu z /etc/fstab: swapon -a

```
swapon -a
```

Włącza z powrotem wszystkie obszary swap wpisane w `/etc/fstab` (albo oznaczone `swap` w innych mechanizmach montowania). Klasyczna para z poprzednią komendą:

```
swapoff -a && swapon -a
```

Ta kombinacja resetuje zawartość swapu do zera, nie zmieniając jego rozmiaru ani konfiguracji — bywa stosowana, gdy swap jest mocno pofragmentowany starymi, już niepotrzebnymi stronami i chcemy zacząć „na czysto", **ale tylko na maszynie, która ma wystarczająco wolnego RAM-u, by to przełknąć**.

swapon = włącz wskazaną partycję albo plik jako przestrzeń wymiany

TYLKO ODCZYT

Zrzut buforów zapisu przed operacją na cache: sync

```
sync
```

Wymusza natychmiastowy zapis wszystkich „brudnych" stron (zmodyfikowanych, jeszcze niezapisanych) na dysk. Zawsze poprzedza `drop_caches` — bez tego jądro mogłoby zwolnić pamięć podręczną razem z niezapisanymi jeszcze zmianami.

ZMIENIA SYSTEM

Zwolnienie page cache: echo 1 > /proc/sys/vm/drop_caches

```
sync; echo 1 > /proc/sys/vm/drop_caches
```

Zwalnia pamięć podręczną stron (pagecache) — czyli treść ostatnio czytanych plików trzymaną w RAM-ie. Komentarz źródłowy z dokumentacji jądra brzmi: *„To free pagecache*".

ZMIENIA SYSTEM

Zwolnienie dentries i inode'ów: echo 2 > /proc/sys/vm/drop_caches

```
sync; echo 2 > /proc/sys/vm/drop_caches
```

Zwalnia pamięć podręczną katalogów (dentries) i i-węzłów (inodes) — struktury opisujące, gdzie w systemie plików leżą pliki i katalogi. Z dokumentacji jądra: *„To free dentries and inodes*".

ZMIENIA SYSTEM

Zwolnienie wszystkiego naraz: echo 3 > /proc/sys/vm/drop_caches

```
sync; echo 3 > /proc/sys/vm/drop_caches
```

Łączy działanie `1` i `2` — czyści całą pamięć podręczną (pagecache, dentries i inodes) jednym poleceniem. Z dokumentacji jądra: *„To free pagecache, dentries and inodes*".

Przykład:

```
$ sync; echo 3 > /proc/sys/vm/drop_caches
$ free -h
               total        used        free      shared  buff/cache   available
Mem:            15Gi       8.2Gi       6.9Gi       512Mi       180Mi       7.0Gi
```

Bufory i cache spadły z 5.3 GiB do 180 MiB, `free` (kolumna wolnej pamięci) skoczyło w górę — ale `available` prawie się nie zmieniło, bo ta pamięć i tak była wcześniej dostępna na żądanie. Zysk jest pozorny, a koszt realny: każdy kolejny odczyt tych samych plików wraca na dysk.

GiB = gibibajt: 1024 MiB, czyli około 1,07 GB; tej jednostki używa dokumentacja MailCow przy wymaganiach pamięci

TYLKO ODCZYT

Sprawdzenie bieżącej skłonności jądra do swapowania: cat /proc/sys/vm/swappiness

```
cat /proc/sys/vm/swappiness
```

Wartość od `0` do `100`, domyślnie zwykle `60`. Im wyższa, tym chętniej jądro wypycha strony anonimowe (pamięć procesów) do swapu, zamiast zmniejszać page cache. Pełne wyjaśnienie mechanizmu w sekcji *Jak to działa*.

Przykład:

```
60
```

cat = concatenate: sklej i wypisz pliki

TYLKO ODCZYT

Zmiana swappiness na gorąco: sysctl vm.swappiness=10

```
sysctl vm.swappiness=10
```

Zmienia wartość natychmiast, bez restartu — ale tylko do najbliższego przeładowania systemu. Sensowne przy testowaniu wpływu przed wpisaniem zmiany na stałe.

Przykład:

```
vm.swappiness = 10
```

sysctl = odczytaj albo zmień parametr jądra Linuksa w czasie działania systemu, tu limit vm.max_map_count wymagany przez OpenSearch

ZMIENIA SYSTEM

Zapisanie trwałej zmiany swappiness w /etc/sysctl.d/

```
echo 'vm.swappiness = 10' > /etc/sysctl.d/99-swappiness.conf
sysctl --system
```

Plik w `/etc/sysctl.d/` przetrwa restart; `sysctl --system` wczytuje go od razu, bez czekania na reboot, i stosuje wszystkie pliki `sysctl.d` w poprawnej kolejności.

sysctl = odczytaj albo zmień parametr jądra Linuksa w czasie działania systemu, tu limit vm.max_map_count wymagany przez OpenSearch

ZMIENIA SYSTEM

Przygotowanie pliku lub partycji jako obszaru swap: mkswap

```
fallocate -l 2G /swapfile
chmod 600 /swapfile
mkswap /swapfile
```

`mkswap` zapisuje na wskazanym pliku lub partycji format rozpoznawany przez jądro jako obszar wymiany — analogicznie do `mkfs` dla systemu plików. `chmod 600` jest obowiązkowe: plik swap czytelny dla innych użytkowników to potencjalny wyciek danych z pamięci.

Przykład:

```
Setting up swapspace version 1, size = 2 GiB (2147479552 bytes)
no label, UUID=3a1f9c02-88f1-4e2a-9e77-1c1b9a1e2233
```

chmod = change mode: zmień prawa dostępu

TYLKO ODCZYT

Aktywacja nowego obszaru swap: swapon /swapfile

```
swapon /swapfile
```

Włącza przygotowany plik od razu, bez restartu.

swapon = włącz wskazaną partycję albo plik jako przestrzeń wymiany

ZMIENIA SYSTEM

Zapisanie wpisu swap w /etc/fstab dla trwałości po restarcie

```
echo '/swapfile none swap sw 0 0' >> /etc/fstab
```

Bez wpisu w `/etc/fstab` plik swap zniknie po restarcie — `swapon -a` (uruchamiane automatycznie przy starcie systemu) czyta właśnie ten plik, żeby wiedzieć, co włączyć.

TYLKO ODCZYT

Sprawdzenie, które procesy trzymają dane w swapie: smem -s swap

```
smem -s swap -r | head -6
```

Sortuje procesy według ilości danych realnie leżących w swapie, nie w RAM-ie — inaczej niż `ps`, które w standardowych kolumnach swapu w ogóle nie pokazuje.

Przykład:

```
  PID User     Command                         Swap      USS      PSS      RSS
 3311 java     java -Xmx2g -jar legacy.jar    412000   890000   910400   980000
 1203 postgres postgres: autovacuum worker         0    45000    48200    52000
```

ZMIENIA SYSTEM

Ręczne przeszukanie /proc/*/status w poszukiwaniu swapu bez dodatkowych pakietów

```
for f in /proc/[0-9]*/status; do
  awk -v pid="${f}" '/^VmSwap/ && $2+0 > 0 {print pid, $2, $3}' "$f"
done | sort -k2 -n -r | head
```

Gdy `smem` nie jest zainstalowany i nie chcemy go doinstalowywać na produkcyjnej maszynie, ta pętla robi to samo ręcznie: przechodzi po katalogach `/proc/PID/`, wyciąga pole `VmSwap` ze `status` i sortuje malejąco.

Przykład:

```
/proc/3311/status 412000 kB
/proc/8842/status 6200 kB
```

awk = od nazwisk autorów Aho, Weinberger, Kernighan: język do danych w kolumnach

TYLKO ODCZYT

Wszystkie flagi

Polecenie	Co robi
`swapon -s` / `swapon --show`	pokazuje aktywne obszary swap
`cat /proc/swaps`	to samo, źródło surowe
`swapoff -a`	wyłącza cały swap, przenosi jego zawartość do RAM
`swapon -a`	włącza swap zdefiniowany w `/etc/fstab`
`sysctl vm.swappiness=N`	zmienia skłonność do swapowania (tymczasowo)
`mkswap` / `swapon plik`	tworzy i aktywuje nowy obszar swap

Uważaj

- `swapoff -a` na maszynie z zajętym swapem może wywołać OOM killera — sprawdź `free -h` **przed**, nie po.
- `drop_caches` nie zwalnia trwale niczego — cache odbuduje się przy pierwszym kolejnym odczycie tych samych plików, kosztem dodatkowego I/O.
- Wysoki `used` w linii `Swap:` z `free -h` sam w sobie nic nie mówi — liczy się aktywny ruch `si`/`so` z `vmstat`.

OOM = Out Of Memory: brak pamięci; OOM killer to mechanizm jądra zabijający proces, gdy jej zabraknie

Z życia wzięte

Z ŻYCIA WZIĘTE

Czwartek, 6:00 w Swarzędzu. WMS na ora01 muli dokładnie w momencie, gdy czterdzieści skanerów loguje się naraz na start zmiany. Tomek widzi w vmstat rosnącą kolumnę swap i tłumaczy Markowi, że system zaczął odkładać rzadziej używane strony pamięci na dysk, co przy starcie zmiany jest właśnie najgorszym możliwym momentem.

Po co to istnieje

Pojęcie: przestrzeń wymiany (swap). Obszar na dysku — dedykowana partycja albo zwykły plik — który jądro traktuje jak rozszerzenie RAM-u. Gdy fizycznej pamięci zaczyna brakować, jądro wybiera strony pamięci używane najrzadziej i zapisuje je na tym obszarze, zwalniając RAM dla czegoś pilniejszego. Nazwa „swap" (zamiana) odnosi się dokładnie do tej operacji: strona pamięci jest zamieniana miejscami — z RAM-u na dysk, i z powrotem, gdy proces znów jej zażąda.

Administrator sprawdza swap z dwóch przeciwstawnych powodów: albo obawia się, że system go nadużywa (spowolnienie, bo dysk jest tysiące razy wolniejszy niż RAM), albo — rzadziej, ale częściej niż się wydaje — nieopatrznie wyłącza go w złym momencie i wywołuje kaskadę problemów gorszych niż ta, którą próbował rozwiązać.

Analogia

Wyobraź sobie **biurko w małym biurze i szafę z archiwum w piwnicy**.

Biurko (RAM) mieści tylko kilka teczek naraz — to, nad czym pracujesz teraz, leży w zasięgu ręki. Kiedy przychodzi nowa sprawa, a biurko jest zawalone, sekretarka nie wyrzuca żadnej teczki — znosi do piwnicy (swap) tę, po którą nikt nie sięgał najdłużej, i kładzie na jej miejscu nową. To działa, dopóki ktoś nie zażąda z powrotem teczki, która akurat jest w piwnicy — wtedy trzeba zejść po schodach, co zajmuje zauważalnie więcej czasu niż sięgnięcie po coś z blatu.

System, w którym od czasu do czasu jedna rzadko używana teczka trafia do piwnicy, działa normalnie — to zwykła gospodarka miejscem. Problem zaczyna się, gdy sekretarka biega po schodach non stop, bo *wszystko*, czego ludzie akurat potrzebują, akurat leży w piwnicy — to jest odpowiednik ciągłego, aktywnego swapowania (`si`/`so` w `vmstat`), a nie samego faktu, że piwnica jest zapełniona starymi teczkami.

A wyłączenie swapu w złym momencie to zabranie piwnicy w chwili, gdy biurko jest już zawalone pod sufit — wszystkie teczki z piwnicy trzeba natychmiast wnieść z powrotem na biurko, które nie

ma na nie miejsca.

Jak to działa

Rozwiń: jak to działa pod spodem

> **Pojęcie: wypieranie stron.** Jądro dzieli pamięć procesów na strony (zwykle 4 KB każda) i śledzi, jak dawno każda była używana. Gdy potrzebuje zwolnić RAM, wybiera strony najdawniej nietknięte (algorytm zbliżony do LRU — *least recently used*) i albo po prostu je odrzuca (jeśli to dane z pliku, można je zawsze odczytać ponownie z dysku — to jest właśnie odzyskiwanie page cache, opisane na stronie o `free`), albo, jeśli to pamięć anonimowa (dane procesu, których nie ma skąd odczytać poza samym procesem), zapisuje je na obszar swap, zanim zwolni stronę w RAM-ie. To zapisanie i późniejsze ewentualne odczytanie z powrotem nazywa się właśnie wypieraniem stron.

> **Pojęcie: swappiness.** Parametr jądra (`/proc/sys/vm/swappiness`, zakres 0–100) sterujący tym, jak chętnie jądro sięga po wypieranie stron anonimowych do swapu zamiast po prostu kurczyć page cache. Wartość `0` nie wyłącza swapu całkowicie, ale mówi jądru: „unikaj swapowania, dopóki się da, kurcz najpierw cache". Wartość `100` mówi: „swapuj równie chętnie, jak zwalniasz cache". Domyślne `60` jest kompromisem dobranym pod typowe stacje robocze; na serwerach bazodanowych czy z dużą ilością RAM-u administratorzy często obniżają tę wartość do `10`–`20`, bo wolą, żeby jądro trzymało dane aplikacji w RAM-ie maksymalnie długo, a page cache kurczyło pierwsze — page cache i tak da się odtworzyć z dysku bez żadnej kary poza czasem odczytu.

Dlaczego `swapoff -a` bywa niebezpieczne: to polecenie nie „wyłącza możliwości swapowania w przyszłości" — ono **natychmiast** czyta z powrotem do RAM-u każdą stronę, która w danej chwili leży na obszarze swap, zanim go wyłączy (bo inaczej dane by przepadły). Jeśli w swapie leżało np. 3 GB danych, a w RAM-ie zostało tylko 1 GB wolnego miejsca, jądro musi znaleźć te dodatkowe 2 GB skądś — najpierw próbuje skurczyć cache, a gdy i to nie wystarcza, może uruchomić OOM killera, dokładnie ten sam mechanizm opisany na stronie o `free`.

Krok po kroku

1. Sprawdź, ile swapu jest w użyciu i jak bardzo.

```
swapon --show
free -h
```

```
NAME      TYPE      SIZE USED PRIO
/dev/dm-1 partition   4G 3.2G   -2
```

3.2 GB z 4 GB w użyciu samo w sobie nie jest wyrokiem — następny krok rozstrzyga, czy to problem.

2. Sprawdź, czy swap jest *aktywny*, czy tylko *zajęty*.

```
vmstat 1 5
```

```
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0 3355000 512000  51200 2048000    0    0     2     1   50   30  5  2 92  1  0
 0  0 3355000 498000  51200 2051000    0    0     4     0   48   28  4  1 94  1  0
```

Kolumny `si` (swap in) i `so` (swap out) to zera w każdym wierszu — dane leżą w swapie, ale nic nie jest do niego aktywnie zapisywane ani z niego odczytywane. To jest **stary, „zimny" swap** — najprawdopodobniej dane trafiły tam kiedyś, gdy było ciaśniej (np. przy starcie usługi albo chwilowym szczycie), i od tamtej pory po prostu tam leżą, bo nikt nie zażądał ich z powrotem, a przenoszenie ich z powrotem do RAM „na zapas" nie miałoby żadnej korzyści.

3. Jeśli `si`/`so` są aktywne, to jest prawdziwy sygnał alarmowy.

```
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 3  2 3900000 102000  48000 1800000  820  650    12     8   60   45 15  8 55 22  0
```

`si` i `so` w setkach oznaczają, że system w tej chwili aktywnie wymienia strony między RAM-em a dyskiem — to realnie spowalnia każdy proces, który akurat czeka na swoje dane. Tu reakcja to szukanie procesu żarłocznego pamięciowo (`ps aux --sort=-%mem`), nie ruszanie samego swapu.

4. Jeśli musisz bezpiecznie zresetować zawartość swapu (np. po dużym, jednorazowym szczycie, który już minął).

```
free -h   # upewnij się, że jest zapas w available
swapoff -a
swapon -a
```

Rób to tylko wtedy, gdy `free -h` pokazuje realny zapas w `available` — w praktyce co najmniej tyle, ile aktualnie zajmuje swap, powiększone o margines bezpieczeństwa.

Częste błędy

`swapoff -a` na serwerze z ciasnym RAM-em.

```
$ swapoff -a
Killed
```

Powłoka, w której uruchomiono komendę, sama pada ofiarą OOM killera w trakcie próby przeniesienia zawartości swapu z powrotem do RAM-u. Naprawa: przed `swapoff -a` zawsze `free -h` — jeśli `available` jest mniejsze niż aktualne zużycie swapu, nie wykonuj tej operacji, dopóki nie zwolnisz pamięci albo nie zwiększysz RAM-u.

Traktowanie `drop_caches` jako standardowego narzędzia do „odzyskania pamięci".

```
sync; echo 3 > /proc/sys/vm/drop_caches
```

Polecenie kończy się bez błędu, `free` (kolumna wolnej pamięci) rośnie efektownie — administrator uznaje zadanie za wykonane. W rzeczywistości `available` prawie się nie zmieniło (ta pamięć i tak była policzalna jako dostępna), a każdy kolejny odczyt plików, które były w cache, teraz musi wrócić na dysk. Na produkcji, zwłaszcza pod bazą danych, to bywa zauważalny spadek wydajności zaraz po operacji. Naprawa: prawie nigdy nie jest to właściwa odpowiedź — jeśli `available` jest niskie, szukaj procesu zajmującego pamięć (`ps aux --sort=-%mem`), nie czyść cache.

Interpretacja wysokiego `Swap: used` jako dowodu na problem z wydajnością.

```
Swap:          4.0Gi       3.2Gi       800Mi
```

Administrator widzi 3.2 GiB w użyciu i od razu planuje restart usługi albo dokupienie RAM-u. Bez sprawdzenia `si`/`so` w `vmstat` to jest strzał w ciemno — te dane mogą leżeć w swapie od tygodni, zupełnie nieaktywne, bez żadnego wpływu na bieżącą wydajność. Naprawa: zawsze `vmstat 1 5` przed wyciąganiem wniosków z samej wielkości `used`.

Zmiana `swappiness` przez `echo` bezpośrednio do `/proc`, myląc to z trwałą konfiguracją.

```
echo 10 > /proc/sys/vm/swappiness
```

Działa natychmiast, ale znika po najbliższym restarcie — administrator odkrywa to miesiąc później, gdy po planowym restarcie serwer wraca do zachowania sprzed zmiany. Naprawa: `sysctl vm.swappiness=10` do szybkiego testu, a trwałą zmianę zapisz w `/etc/sysctl.d/99-swappiness.conf` i zastosuj przez `sysctl --system`.

Zapomniany wpis w `/etc/fstab` po ręcznym `mkswap`/`swapon`.

```
mkswap /swapfile
swapon /swapfile
# reboot
free -h
```

```
               total        used        free      shared  buff/cache   available
Swap:            0B          0B          0B
```

Plik swap istniał, ale zniknął z listy aktywnych po restarcie, bo `swapon -a` (uruchamiane automatycznie przy starcie) nie wiedziało o jego istnieniu — nikt nie dopisał go do `/etc/fstab`. Naprawa: `echo '/swapfile none swap sw 0 0' >> /etc/fstab`, zanim ktokolwiek zapomni.

□□ Ćwiczenia

> Wszystko poniżej wykonujesz na własnej maszynie, w katalogu `/tmp`. Operacje na `/proc/sys/vm/drop_caches` i `swapon`/`swapoff` wymagają uprawnień administratora — na maszynie testowej/wirtualnej, nigdy na cudzej produkcji.

Ćwiczenie 1 — plik swap od zera

```
cd /tmp
sudo fallocate -l 256M /tmp/test.swap
sudo chmod 600 /tmp/test.swap
sudo mkswap /tmp/test.swap
sudo swapon /tmp/test.swap
swapon --show
```

Sprawdź wynik, a potem posprzątaj:

```
sudo swapoff /tmp/test.swap
sudo rm /tmp/test.swap
```

Dlaczego `chmod 600` jest tu istotne, a nie tylko kosmetyczne?

Odpowiedź

Plik swap zawiera surowe strony pamięci innych procesów — potencjalnie hasła w pamięci, klucze prywatne, fragmenty danych aplikacji. Jeśli plik miałby uprawnienia do odczytu dla innych użytkowników systemu, każdy z lokalnym dostępem mógłby przeszukać jego zawartość binarnie i wyłowić wrażliwe dane, które nigdy nie trafiły na dysk w żadnej innej, bardziej oczywistej formie.

Ćwiczenie 2 — obserwacja `si`/`so` w praktyce

```
free -h
vmstat 1 5
```

Uruchom to na własnej maszynie w spoczynku, a potem — jeśli masz do dyspozycji maszynę testową z małym RAM-em — pod obciążeniem (np. kompilacja dużego projektu albo `stress-ng --vm 2 --vm-bytes 80% --timeout 30s`, jeśli pakiet jest dostępny). Porównaj kolumny `si`/`so` w obu przypadkach.

Odpowiedź

W spoczynku `si` i `so` powinny być zerami przez cały czas trwania pomiaru, nawet jeśli `swpd` (ilość danych w swapie) jest większe od zera — to potwierdza, że dane w swapie są „zimne". Pod realnym obciążeniem pamięciowym, gdy RAM zaczyna być zbyt ciasny, powinieneś zobaczyć niezerowe wartości `si`/`so` pojawiające się w kolejnych próbkach — to jest moment, w którym system faktycznie płaci koszt wydajnościowy za brak pamięci.

Pytania kontrolne

1. Serwer ma `Swap: used` na poziomie 2 GB od trzech tygodni, bez zmian. Czy to powód do restartu usługi?

Odpowiedź

Nie automatycznie. Trzeba sprawdzić `vmstat 1 5` — jeśli `si`/`so` są zerami, te 2 GB leżą w swapie nieaktywnie i nie mają żadnego wpływu na bieżącą wydajność. Restart usługi tylko po to, żeby „zwolnić swap", nic nie naprawia i wprowadza niepotrzebną przerwę w działaniu.

2. Dlaczego `sync` zawsze poprzedza `echo N > /proc/sys/vm/drop_caches`, a nie odwrotnie?

Odpowiedź

`drop_caches` zwalnia tylko strony „czyste" — takie, które są identyczne z tym, co leży na dysku. Jeśli w cache są jeszcze niezapisane zmiany (strony „brudne"), `sync` wymusza ich zapis na dysk, zanim spróbujesz zwolnić pamięć — bez tego jądro po prostu pominie brudne strony przy czyszczeniu, a efekt polecenia będzie mniejszy niż oczekiwany.

3. Ustawiłeś `sysctl vm.swappiness=10` na produkcyjnym serwerze bazy danych i po tygodniu widzisz mniej ruchu w swapie. Co się stało z page cache w tym czasie?

Odpowiedź

Niższa `swappiness` sprawia, że jądro preferuje kurczenie page cache przed wypieraniem stron anonimowych procesów do swapu. W praktyce oznacza to, że page cache będzie relatywnie mniejszy niż przy domyślnym `60`, a pamięć procesu bazy danych zostanie w RAM-ie dłużej — co dla bazy danych jest zwykle pożądane, bo jej własne bufory (np. `shared_buffers` w PostgreSQL) są cenniejsze niż cache systemu plików dla przypadkowych odczytów.

4. Kolega proponuje dodanie `swapoff -a && swapon -a` do cotygodniowego crona „dla higieny". Jakie ryzyko mu wskażesz?

Odpowiedź

Że taki cron w nieprzewidzianym momencie zresetuje cały swap na produkcyjnym systemie, który akurat może mieć w danej chwili mało wolnego RAM-u — dokładnie sytuacja opisana w sekcji *Częste błędy*. Bez sprawdzenia `free -h` bezpośrednio przed operacją, zautomatyzowany `swapoff -a` może w środku nocy przenieść kilka gigabajtów danych do RAM-u i wywołać OOM killera na serwerze, na którym nikt akurat nie patrzy na ekran. „Higiena" swapu nie jest czymś, co trzeba robić rutynowo — zimny, nieaktywny swap nie szkodzi.

Powiązane: Pamięć naprawdę: free i /proc/meminfo · Trendy zamiast migawek: vmstat, iostat, sar, pidstat · Obciążenie systemu: load average, uptime, dmesg

Źródła: `man 8 swapon` · `man 5 proc` (`drop_caches`) · Documentation/admin-guide/sysctl/vm.rst w źródłach jądra Linux · wewnętrzna notatka „Czyszczenie SWAP"

Revision #16
Created 2026-09-17 21:43:28 UTC by Tomasz Konieczny
Updated 2026-10-03 19:09:14 UTC by Tomasz Konieczny