Kompresja pamięci w jądrze Linuksa

Piotr Sarna

Uniwersytet Warszawski

piotr.sarna@students.mimuw.edu.pl

7 lutego 2015

Plan prezentacji

1. Wprowadzenie
   - pamięć niedościgła
2. Wykorzystanie kompresji w zarządzaniu pamięcią
   - wydajność
   - zswap
   - zcache
   - zram
   - zsmalloc vs zbud
3. Algorytmy kompresji
   - wydajność
   - LZO
   - LZ4
   - Crypto API

Idea

„What if there was a class of memory that is of unknown and dynamically variable size, is addressable only indirectly by the kernel, can be configured either as persistent or as "ephemeral" (meaning it will be around for awhile, but might disappear without warning), and is still fast enough to be synchronously accessible?"
- Dan Magenheimer

Cechy

- Pule stron o nieznanym rozmiarze
- Strony trwałe (persistent) i nietrwałe (ephemeral)
- Adresowanie przy pomocy uchwytów
- Prosty interfejs
- Operacje mogą się nie powieść

Front-endy

Cleancache

Służy do trzymania stron, które mogą się przydać, ale nic się nie stanie, jeśli znikną.
Interfejs:

- int cleancache_put_page(struct page *page);
- int cleancache_get_page(struct page *page);

Frontswap

Służy do trzymania stron, którym nic złego nie może się stać.
Interfejs:

- int frontswap_store(struct page *page);
- int frontswap_load(struct page *page);

Dowiązanie cleancache

```
/*
 * Delete a page from the page cache and free it. Caller has to make
 * sure the page is locked and that nobody else uses it - or that usage
 * is safe.  The caller must hold the mapping's tree_lock.
 */
void __delete_from_page_cache(struct page *page, void *shadow)
{
        struct address_space *mapping = page->mapping;

        trace_mm_filemap_delete_from_page_cache(page);
        /*
         * if we're uptodate, flush out into the cleancache, otherwise
         * invalidate any existing cleancache entries.  We can't leave
         * stale data around in the cleancache once our page is gone
         */
        if (PageUptodate(page) && PageMappedToDisk(page))
                cleancache_put_page(page);
        else
                cleancache_invalidate_page(mapping, page);

        page_cache_tree_delete(mapping, page, shadow);

        page->mapping = NULL;
```

Dowiązanie frontswap

```
/*
 * We may have stale swap cache pages in memory: notice
 * them here and get rid of the unnecessary final write.
 */
int swap_writepage(struct page *page, struct writeback_control *wbc)
{
        int ret = 0;

        if (try_to_free_swap(page)) {
                unlock_page(page);
                goto out;
        }
        if (frontswap_store(page) == 0) {
                set_page_writeback(page);
                unlock_page(page);
                end_page_writeback(page);
                goto out;
        }
        ret = __swap_writepage(page, wbc, end_swap_bio_write);
out:
        return ret;
}
```

Back-endy

Xen tmem

Usprawnia wykorzystanie pamięci w maszynach wirtualnych.

zswap

Implementacja frontswap wykorzystująca kompresję

zcache

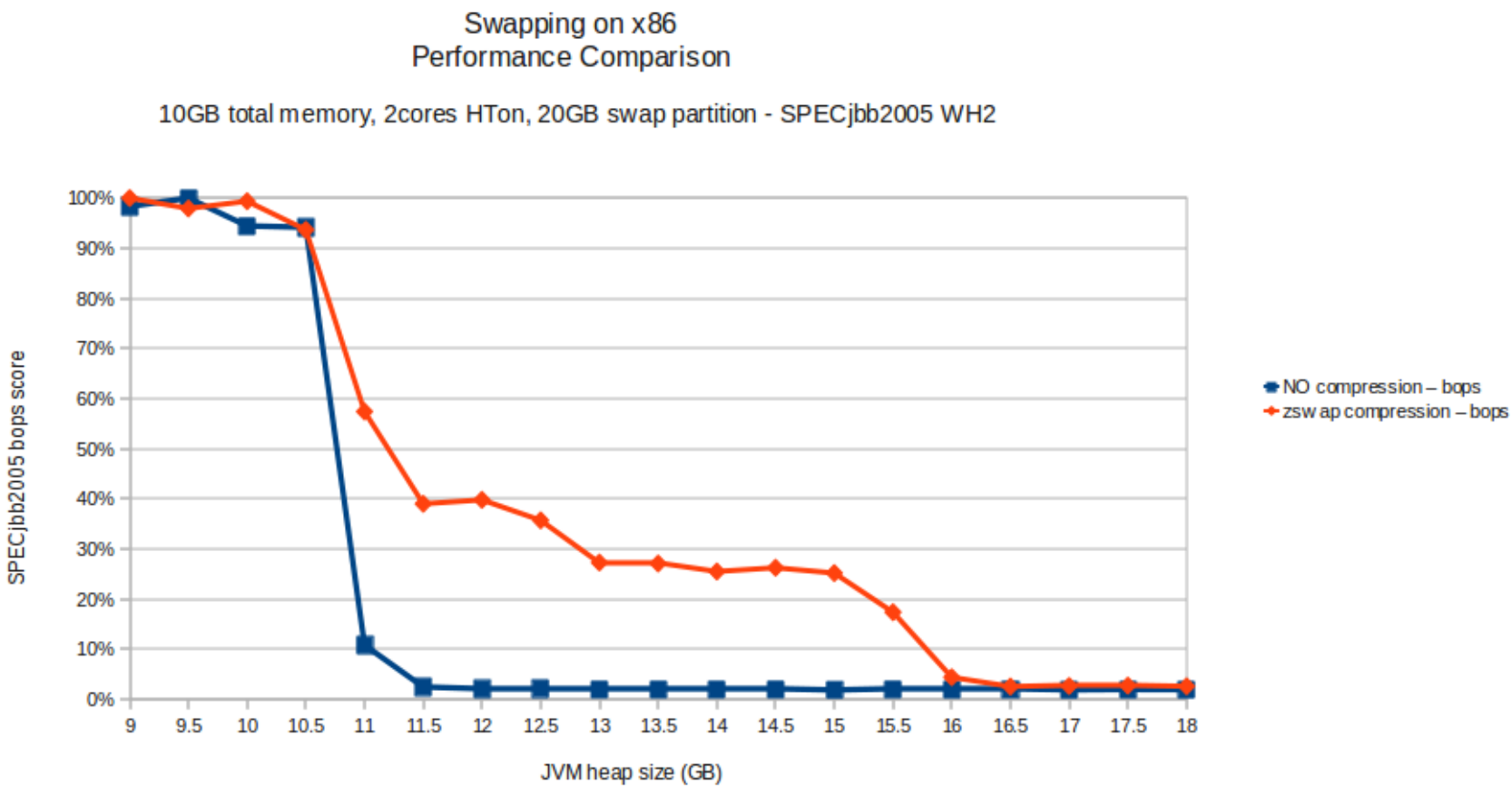
Implementacja frontswap i cleancache wykorzystująca kompresję

Rejestracja zswap jako back-end frontswap

```
static struct frontswap_ops zswap_frontswap_ops = {
        .store = zswap_frontswap_store,
        .load = zswap_frontswap_load,
        .invalidate_page = zswap_frontswap_invalidate_page,
        .invalidate_area = zswap_frontswap_invalidate_area,
        .init = zswap_frontswap_init
};

frontswap_register_ops(&zswap_frontswap_ops);
```

wydajność



https://www.ibm.com/developerworks/community/blogs/fe313521-2e95-46f2-817d-44a4f27eba32/entry/new_linux_zswap_compression_functionality7

zswap

- Implementacja interfejsu frontswap
- Struktura danych: drzewo informacji o skompresowanych stronach
- Kompresja wykorzystuje cryptographic API wbudowane w jądro
- W razie potrzeby przerzuca strony do „prawdziwego” urządzenia swap (resumed writeback)
- Wykorzystywał alokator zsmalloc, obecnie korzysta ze zbud
- Dostępny w głównej gałęzi jądra od wersji 3.11

Jak skorzystać?

demo

zcache

- Implementuje zarówno cleancache, jak i frontswap
- Struktura danych: drzewo informacji o skompresowanych stronach
- Korzysta z alokatora zbud
- Kompresja wykorzystuje cryptographic API wbudowane w jądro
- Dostępny w głównej gałęzi jądra DO wersji 3.11
- Część pomysłów została przeniesiona do zswap

Jak skorzystać?

Używać przestarzałego jądra Linux, które udostępnia zcache jako eksperymentalny dodatek.

zram

Zupełnie inne podejście

- Sterownik wirtualnego urządzenia blokowego
- Włączany ręcznie przy pomocy instrukcji mkswap i swapon
- Działa jako samodzielne urządzenie swapujące
- Można go wykorzystać jako samodzielny swap w systemach wbudowanych
- Inne wykorzystanie: kompresowane /tmp/

Jak skorzystać?

demo

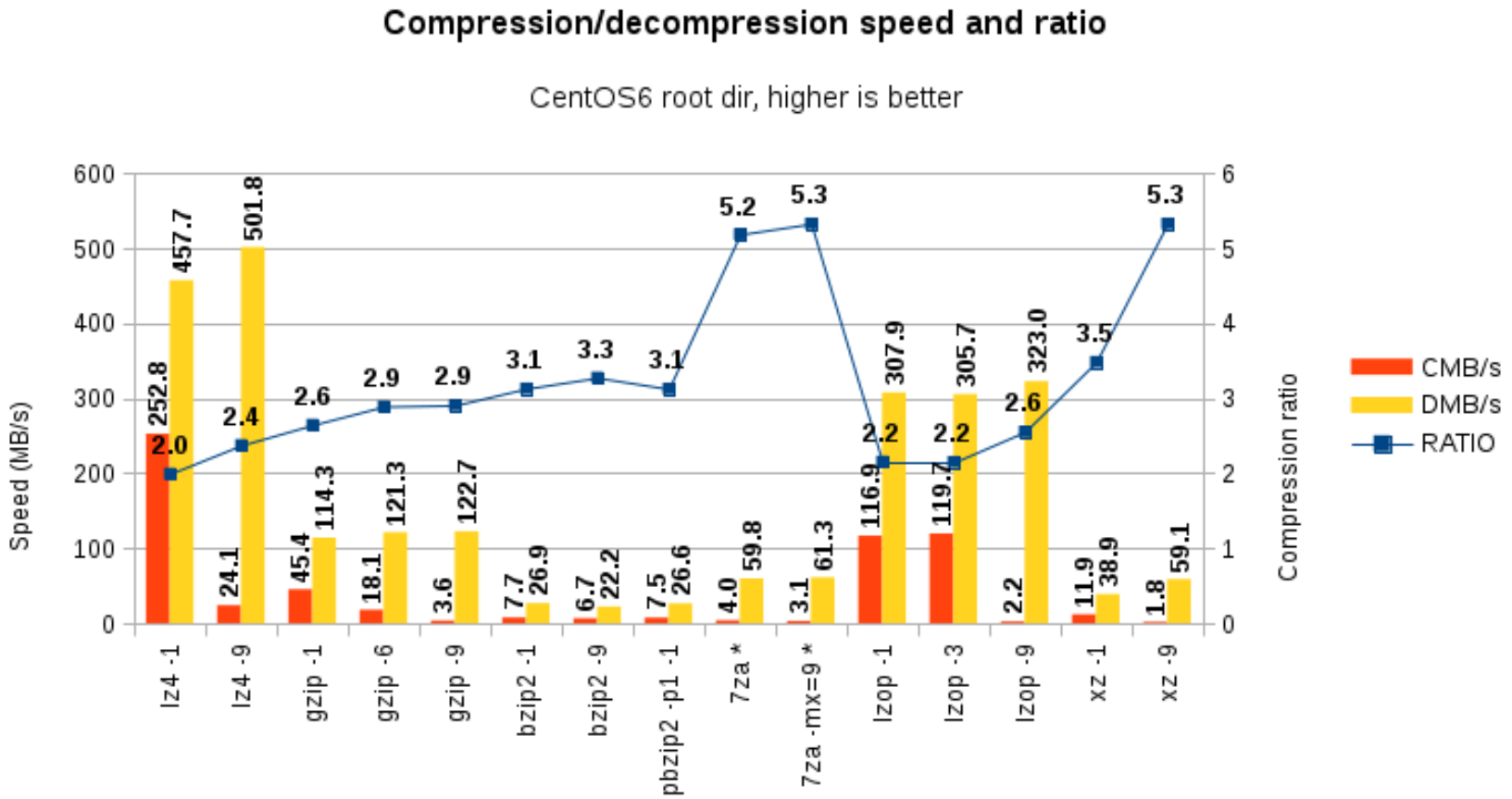
zsmalloc vs zbud

zsmalloc

- Umożliwia fragmentację obiektów
- Nie wymaga ciągłego kawałka pamięci

zbud

- Dobiera kompresowane strony w pary
- Ma lepszą wydajność niż zsmalloc



http://linuxaria.com/article/linux-compressors-comparison-on-centos-6-5-x86-64-lzo-vs-lz4-vs-gzip-vs-bzip2-vs-lzma

LZO

Źródła

http://www.oberhumer.com/opensource/lzo/

- oferuje bardzo szybką dekompresję
- nie potrzebuje dodatkowej pamięci przy dekompresji
- można dostosować balans pomiędzy współczynnikiem kompresji a jej prędkością, nie zmieniając jednocześnie szybkości dekompresji.

LZ4

Źródła

https://code.google.com/p/lz4/

- oferuje jeszcze szybszą dekompresję
- kosztem dłuższej kompresji można poprawić zarówno współczynnik kompresji, jak i szybkość dekompresji.

Crypto API

Wbudowane w jądro Linuksa implementacje algorytmów, w tym także algorytmy kompresujące.

- generyczny interfejs
- prosty przepływ (alloc, compress, decompress, free)
- dostępne m. in. lzo, lz4, 842

Dziękuję za uwagę.

Źródła:

1. : Seth Jennings, The zswap compressed swap cache (http://lwn.net/Articles/537422/)
2. : Dan Magenheimer, In-kernel memory compression (http://lwn.net/Articles/545244/)