Struktura jąder systemów operacyjnych

Lista zadań nr 9

Na zajęcia 17 czerwca 2020

UWAGA! W trakcie prezentacji rozwiązań należy zdefiniować i wyjaśnić pojęcia, które zostały oznaczone **wytłuszczoną** czcionką.

Zadanie 1. Rozważmy procedury zarządzania tablicą stron wymienione w `pmap(9)` (ang. physical map). Które z nich wykorzystuje się do implementacji wywołań systemowych związanych z zarządzaniem pamięcią procesu (`mmap(2)`, `munmap(2)`, `mprotect(2)`) i tworzeniem procesów (`fork(2)` i `execve(2)`), a które przez algorytm zastępowania ramek i **demon stronicowania**? Które z podanych przez Ciebie procedur muszą wyczyścić część lub całą zawartość TLB? Jakie informacje niesie ze sobą lista `pv_entry`? Na podstawie pliku `pmap.c` powiedz, które z procedur interfejsu pmap wymagają użycia tej listy?

Zadanie 2. Procesory z rodziny MIPS nie posiadają **bitów monitorowania dostępu** we wpisach tablicy stron. Co więcej nie oferują również sprzętowego przeglądania tablicy stron, więc jądro systemu odpowiada za zarządzanie zawartością TLB. Wpisy przechowywane w TLB posiadają bity uprawnień: bit modyfikacji (ang. dirty) i bit ważności (ang. valid). Procesor nie zmienia samodzielnie wartości tych bitów. Instrukcje generują odrębne wyjątki procesora przy chybieniu w TLB (ang. TLB refill exception); przy próbie dostępu, gdy V=0 (ang. TLB invalid exception); przy próbie modyfikacji strony, gdy D=0 (ang. TLB modified exception). Zaproponuj algorytm realizacji `pmap_is_referenced(9)` i `pmap_is_modified(9)` dla architektury MIPS.

Zadanie 3. Zarządzanie buforem translacji adresów komplikuje się na maszynach SMP (ang. Shared Memory Processor). Czemu zawartość TLB poszczególnych rdzeni nie jest automatycznie synchronizowana? Na podstawie artykułu „Translation Lookaside Buffer Consistency: A Software Approach”[1] wyjaśnij działanie prostego algorytmu **zestrzeliwania wpisów TLB** (ang. TLB shootdown) z systemu Mach.

Wskazówka: Problem i rozwiązanie opisano również w [8, §15.9] i [8, §15.10].

Zadanie 4. Dla podanego poniżej fragment wydruku polecenia «`x86info -a`» podaj **zasięg bufora translacji adresów** (ang. TLB reach) pierwszego i drugiego poziomu.

```
L1 Data TLB: 4KiB pages, 16 entries
L1 Data TLB: 4MiB pages, 4 entries
L2 Data TLB: 4KiB pages, 256 entries
L2 Data TLB: 4MiB pages, 32 entries
```

Z jakich przyczyn producenci procesorów wprowadzili **duże strony** (ang. superpages, huge pages)? Na podstawie [4, §6.11] opowiedz jak jądro FreeBSD automatycznie wypromowuje ciągły obszar stron do superstrony. Jakie problemy sprawia zarządzanie dużymi stronami?

Podpowiedź: Więcej informacji można znaleźć w publikacji „Practical, transparent operating system support for superpages”[2].

Zadanie 5. Najprostszy wariant algorytmu przydziału i zwalniania stron fizycznych `vm_page` jest realizowany odpowiednio przez procedurę `vm_phys_alloc_freelist_pages` oraz `vm_phys_free_pages`. Dla uproszczenia przyjmujemy, że wartość makrodefinicji «MAXMEMDOM», «VM_NFREELIST» i «VM_NFREEPOOL» wynosi 1. Posługując się kodem tych procedur przedstaw działanie **algorytmu bliźniaków**. W szczególności pokaż jak przebiega podział i złączanie bloków.

[1] https://www.cs.rice.edu/~alc/comp521/Papers/p113-black.pdf
[2] https://people.mpi-sws.org/~druschel/publications/superpages.pdf

Zadanie 6. Algorytm przydziału i zwalniania bloków stron wirtualnej przestrzeni adresowej jądra [4, §6.3] jest realizowany odpowiednio przez procedurę `vmem_xalloc` i `vmem_xfree`. Zakładamy, że parametry procedury «`vmem_xalloc`» przyjmują odpowiednio wartości: «`PAGE_SIZE`» dla «`align`», «`VMEM_ADDR_MIN`» dla «`minaddr`», «`VMEM_ADDR_MAX`» dla «`maxaddr`», 0 dla «`nocross`» i «`phase`». Przydział przebiega według strategii «`M_FIRSTFIT`». Wiemy, że mamy wystarczająco dużo pamięci, żeby spełnić żądanie przydziału.

Posługując się kodem tych procedur przedstaw działanie **algorytmu vmem** przedstawionego w [1, 4]. Jaka motywacja stała za wprowadzeniem nowego algorytmu zarządzania przedstrzenią adresową do jądra? Do czego służą argumenty «`vmem_xalloc`», które pominęliśmy w rozważaniach?

Zadanie 7. Alokator strefowy (ang. zone allocator) systemu `FreeBSD` [4, §6.3] stosuje **wiaderka**, zwane dalej magazynkami (zgodnie z [1, 3]). Co przechowujemy w magazynkach? Czemu każdy procesor posiada zestaw dwóch magazynków? W jaki sposób system dobiera dynamicznie rozmiar magazynków? Skąd algorytm bierze pamięć, jeśli procesor wystrzela obydwa magazynki?

Zadanie 8. Przypomnij do czego służą `vm_object` i **obiekty przesłaniające** (ang. shadow object)? Co się dzieje z łańcuchami obiektów przesłaniających w trakcie obsługi wywołań systemowych `fork(2)` i `mmap(2)` dla odwzorowań prywatnych? Na podstawie ustępu „Collapsing of Shadow Chains” z [4, §6.5] zreferuj algorytm skracania łańcuchów obiektów przesłaniających. Kiedy zachodzi potrzeba jego wykonania?

Literatura

[1] „Magazines and Vmem: Extending the Slab Allocator to Many CPUs and Arbitrary Resources”[3]
Jeff Bonwick, Jonathan Adams

[2] „Modern Operating Systems”
Andrew S. Tanenbaum, Herbert Bos;
Pearson; 4th edition, 2015

[3] „Operating Systems Internals and Design Principles”
William Stallings; Pearson; 9th edition, 2018

[4] „The Design and Implementation of the FreeBSD® Operating System”
Marshall Kirk McKusick, George V. Neville-Neil, Robert N.M. Watson;
Addison-Wesley Professional; 2nd edition, 2014

[5] „Linux Kernel Development”
Robert Love; Addison-Wesley; 3rd edition, 2010

[6] „Solaris Internals: Solaris 10 and OpenSolaris Kernel Architecture”
Richard McDougall, Jim Mauro; Prentice Hall; 2nd edition, 2006

[7] „FreeBSD Device Drivers: A Guide for the Intrepid”
Joseph Kong; No Starch Press; 2012

[8] „UNIX Internals: The New Frontiers”
Uresh Vahalia; Prentice Hall; 1996

[9] „Mac OS X and iOS Internals: To the Apple’s Core”
Jonathan Levin; Wrox; 2012

[3] `https://www.usenix.org/legacy/publications/library/proceedings/usenix01/full_papers/bonwick/bonwick.pdf`