Struktura jąder systemów operacyjnych

Lista zadań nr 6

Na zajęcia 6 i 13 maja 2020

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

Zadanie 1. Na podstawie [1, §8.1.3] opowiedz czemu koszt operacji atomowych (np. `compare-and-swap`) jest wysoki w **systemach SMP** (ang. *Symmetric MultiProcessing*). Zdefiniuj problem **fałszywego współdzielenia** (ang. *false sharing*) i **gry w ping-pong** (ang. *cache ping-pong*). Czemu w danej linii pamięci podręcznej powinna znajdować się tylko jedna blokada wirująca? W procedurze `mutex_vector_enter` zidentyfikuj kod odpowiadający za opisany w książce algorytm exponential backoff i wyjaśnij jego działanie.

Zadanie 2. Na podstawie [5, §7.4.2] zreferuj problem **wygłodniałej hordy** (ang. *thundering herd*). Czy zwalnianie blokady `mutex(9)` z użyciem «`turnstile_broadcast`» nie prowadzi do tego problemu?

Zadanie 3. Na stronie 105 [2] pojawia się następująca uwaga:

> When releasing a turnstile lock, all waiting threads are released. Because the threads are ordered from highest to lowest priority, that is the order in which they will be awakened. (...) And because they are released from highest to lowest priority, the highest priority thread will usually be the first to acquire the lock. There will be no need for, and hence no overhead from, priority propagation.

Załóżmy, że w trakcie zdejmowania blokady `mutex(9)` typu «`MTX_DEF`» wybudzamy tylko jeden wątek zamiast wszystkich. Czemu jest to mniej wydajne rozwiązanie od wybudzenia wszystkich? Pokaż, że implementacja «`mutex_unlock`» z użyciem «`turnstile_broadcast`» unika kosztów związanych z wielokrotnym odpożyczaniem priorytetów.

Zadanie 4. Jakie dane zawiera struktura muteksa adaptacyjnego w jądrze `NetBSD`? Do czego służy bit «`WAITERS`»? Zapoznaj się z procedurami `MUTEX_ACQUIRE` i `MUTEX_RELEASE`. Czemu ich implementacja używa **barier pamięciowych** (ang. *memory barrier*)? Pomiń rozważanie makra «`MUTEX_INHERITDEBUG`».

Wskazówka: Implementacja «`kmutex_t`» w jądrze `NetBSD` bazuje na wersji opisanej w [3, §17.5.2].

Zadanie 5. Przeprowadź uczestników zajęć przez procedury zakładające i zwalniające blokadę «`kmutex_t`» odpowiednio `mutex_vector_enter` i `mutex_vector_exit`. Rozważamy wieloprocesorową adaptacyjnę wersję blokady, tj. zostało zdefiniowane makro «`MULTIPROCESSOR`» i «`FULL`». W trakcie omawiania implementacji całkowicie pomiń moduł odpluskwiania blokad (`LOCKDEBUG`) i zbierania statystyk dot. blokad (`LOCKSTAT`). Jakie jest zadanie procedury `mutex_oncpu`? Wskaż miejsce wykorzystania procedur rogatek (ang. *turnstile*).

UWAGA! Należy zredagować kod źródłowy przed prezentacją usuwając wiersze zaciemniające przepływ sterowania.

Zadanie 6. Odpowiedz na następujące pytania dotyczące implementacji **rogatek** w systemie `FreeBSD`:

- W jakim celu rogatka śledzi właściciela `ts_owner` blokady?
- Czemu istnieją dwie listy wątków zablokowanych `ts_blocked` na danej rogatce?
- Względem jakiego parametru posortowane są listy «`ts_blocked`» i dlaczego?
- W jakim celu wątek przechowuje listę rogatek `td_contested` wszystkich posiadanych blokad, o które istnieje współzawodnictwo?
- Czy wybudzanie wątków byłyby prostsze, gdyby rogatki były przypisane na stałe do wątków?

Wskazówka: W ostatnim pytaniu przyjrzyj się pierwszemu wątkowi, który użyczył swojej rogatki i włożył ją do łańcucha.

Zadanie 7. Na podstawie [3, §17.6] opisz semantykę pól słowa maszynowego `rw_owner` blokady współdzielonej `rwlock(9)`. Następnie odpowiedz na poniższe pytania:

- Czy pisarze i czytelnicy pożyczają sobie nawzajem priorytet?
- Czy blokada dopuszcza głodzenie czytelników lub pisarzy?
- Czy po wybudzeniu z oczekiwania na zwolnienie blokady czytelnicy i pisarze mogą wejść do sekcji krytycznej bez wykonywania dodatkowych operacji na blokadzie?

UWAGA! Należy pominąć opisywanie pola «`NODEBUG`».

Zadanie 8. Na podstawie komentarzy zawartych w kodzie, omów algorytm zakładania i zwalniania blokad «`krwlock_t`» zaimplementowany przez procedury `rw_vector_enter` i `rw_vector_exit`. Podobnie jak w zadaniu o «`kmutex_t`» zignoruj referencje do modułu odpluskwiania i zbierania statystyk. Jakie różnice zauważasz w stosunku do algorytmu opisanego w [3, §17.6.1]?

UWAGA! W trakcie prezentacji proszę nie wchodzić w szczegóły, o ile fragment kodu nie budzi wątpliwości wśród słuchaczy.

Literatura

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

[2] „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

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

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

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

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