Uniwersytet Mikołaja Kopernika
Wydział Fizyki, Astronomii i Informatyki Stosowanej
Katedra Informatyki Stosowanej

Rafał Szklarski
nr albumu: 189226

Praca magisterska
na kierunku Fizyka Techniczna

Planista całkowicie sprawiedliwy. Algorytm, śledzenie działania i strojenie

Opiekun pracy dyplomowej
dr hab. Jacek Kobus
Zakład Mechaniki Kwantowej

Toruń 2009

Pracę przyjmuję i akceptuję

.....
data i podpis opiekuna pracy

Potwierdzam złożenie pracy dyplomowej

.....
data i podpis pracownika dziekanatu

Składam serdeczne podziękowania Panu dr hab. Jackowi Kobusowi za okazywaną cierpliwość i pomoc w trakcie przygotowywania niniejszej pracy, za cenne porady, a także za życzliwą i miłą atmosferę sprzyjającą pracy naukowej. Dziękuję również moim bliskim za cierpliwość oraz wiarę i wsparcie w dążeniu do celu.

Spis treści

1 Wstęp

Po włączeniu komputera do jego pamięci jest ładowane jądro systemu operacyjnego, czyli specjalny program odpowiedzialny za zarządzanie jego zasobami: pamięcią, jednostką centralną oraz urządzeniami wejścia-wyjścia. Współczesne systemy operacyjne są złożonymi programami, które pozwalają na realizację wieloprocesowości i tym samym na równoczesne korzystanie z systemu komputerowego wielu użytkownikom. W przypadku systemu komputerowego wyposażonego w pojedynczą jednostkę centralną w danej chwili może się wykonywać tylko kod jednego procesu – może to być proces systemu operacyjnego, bądź proces użytkownika. Jądro musi zatem dysponować jakimś sposobem rozdziału cykli jednostki centralnej pomiędzy konkurujące ze sobą procesy. Przydziałem jednostki centralnej poszczególnym procesom zajmuje się podsystem jądra systemu operacyjnego zwany planistą.

Celem niniejszej pracy jest omówienie działania nowego planisty jądra systemu Linux, tzw. planisty całkowicie sprawiedliwego (ang. *Completely Fair Scheduler*). Planista CFS został opracowany w 2007 r. przez Ingo Molnara i został włączony do jądra 2.6.23 zastępując dotychczasowego planistę O(1) [1]. Planista O(1) pojawił się w jądrze 2.5 w 1995 roku, więc był używany na tyle długo, aby ujawniły się wszystkie jego mocne i słabe strony. Podstawową jego wadą jest bardzo złożony kod, który utrudnia poprawianie wszelkich usterek i dodawanie nowych funkcji. Ponadto planista O(1) stosuje skomplikowane heurystyki do oceny stopnia interaktywności zadań, które w pewnych sytuacjach zawodziły. Skutkowało to przydzielaniem przez planistę nieadekwatnych kwantów czasu dla zadań, co powodowało pogorszenie ogólnej wydajności systemu.

Nowy planista rezygnuje z heurystyk i ma całkowicie przebudowany mechanizm decyzyjny dla zadań zwykłego typu (zadań `SCHED_OTHER`). Implementując ten typ zadań Molnar wykorzystał pomysł Cona Kolivasa [2] pozwalający na sprawiedliwy podział cykli jednostki centralnej między wszystkie zadania w systemie (z pominięciem zadań czasu rzeczywistego). Ta idea została zastosowana po raz pierwszy w napisanym przez Kolivasa planiście RSDL (ang. *Rotating Staircase DeadLine*) [2]. Ten planista traktuje wszystkie zadania jednakowo, tzn. zadania dostają tyle samo czasu niezależnie od swojego charakteru (znika rozróżnienie na zadania uzależnione od jednostki centralnej oraz od operacji wejścia-wyjścia). Planista RSDL sprawdził się doskonale i dlatego Molnar postanowił na nim oprzeć swoją implementację nowego planisty jądra systemu Linux.[1]

Praca składa się z trzech zasadniczych części.

Po pierwsze, w pracy omówiony został przydział czasu jednostki centralnej w systemach z podziałem czasu na przykładzie planisty O(1) oraz CFS. Szczególny nacisk położono na wskazanie kluczowych różnic w mechanizmie planowania stosowanym przez planistów O(1) i CFS.

Po drugie, praca omawia najważniejsze zmiany związane z implementacje nowego planisty. Wyjaśnia na czym polega modularność planisty, która przyczyniła się do zasadni-

[1] Informacje o kontrowersjach jakie to wywołało można znaleźć pod adresem `http://kerneltrap.org/node/8059`.

czego uporządkowania i uproszczenia kodu zarządzającego poszczególnymi klasami zadań. Praca opisuje także główne zmiany jakie Molnar wprowadził do mechanizmu planowania zadań typu `SCHED_OTHER`. W opisie implementacji najważniejszą kwestią jest prezentacja mechanizmu wyboru nowego zadanie do uruchomienia oraz charakterystyka tych momentów działania systemu operacyjnego, kiedy ten wybór jest dokonywany. Kwestie związane z realizacją przełączenia procesora od jednego zadania do drugiego są pomijane. Warto podkreślić, że w planiście CFS zadbano o to, aby planowanie na serwerach mogło przebiegać równie sprawnie jak na komputerach biurkowych i przenośnych. Osiągnięto to udostępniając w nowym planiście szereg parametrów, poprzez które można wpływać w czasie rzeczywistym na jego działanie i tym samym na działanie całego systemu. Praca zawiera szczegółowe omówienie znaczenia tych parametrów, dzięki czemu administrator uzyskuje możliwość strojenia działania planisty CFS.

Po trzecie wreszcie, praca zawiera opis architektury, implementacji oraz działania modułu *schedtrace* służącego do śledzenia pracy planisty. Dzięki niemu administrator ma możliwość oceny wpływu wybranych parametrów planisty na jego działanie. Moduł ten pobiera dane z wewnętrznych struktur jądra wykorzystując mechanizm **tracepoints**, który umożliwia śledzenie działania wybranych funkcji jądra. Te dane są udostępniane programom z przestrzeni użytkownika poprzez pseudosystem plików **proc**, a także możliwe jest wpływanie z tego poziomu na rodzaj danych gromadzonych przez moduł. W ten sposób uzyskano możliwość drobiazgowego śledzenia pracy planisty. Dodatkowy program *schedtrace* pozwala na uruchamianie programów testowych i jednoczesne włączanie rejestrowania wybranych danych. W ten sposób użytkownik może obserwować jak zmiana parametrów pracy planisty oraz priorytetów zadań i polityki ich przetwarzania wpływa na sposób traktowania tych zadań przez system. Zarówno moduł jak i program są udostępniane (w formie paczki tgz) w ramach licencji GNU GPL [3]. Pracę zamykają dodatki, w których opisano implementację modułu i programu oraz instalację tych narzędzi.