L4Ka::Pistachio

Mikrojądra mogą być wydajne

Marek Materzok

Instytut Informatyki Uniwersytetu Wrocławskiego

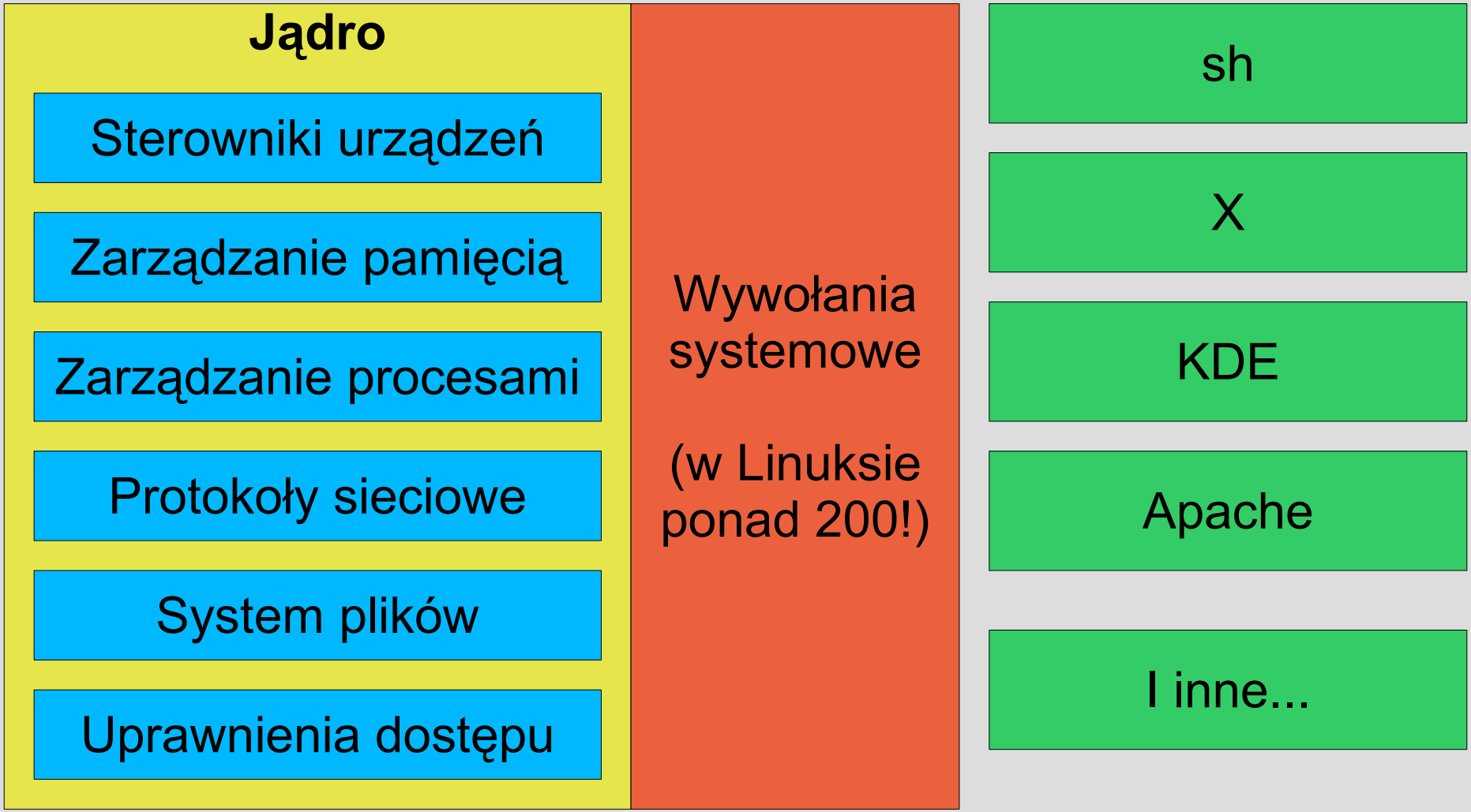
Spotkanie Koła Studentów Informatyki

8 grudnia 2005

Architektura monolityczna

- Jądro jednolitym blokiem kodu pracującym z podwyższonymi uprawnieniami
- W skład jądra wchodzi wiele różnych podsystemów, które komunikują się z aplikacjami przez rozbudowane API
- Aplikacje nie mogą ingerować w funkcjonowanie systemu

Architektura monolityczna



Zalety architektury monolitycznej

- Wysoka wydajność – komunikacja między podsystemami jądra odbywa się przez współdzielenie danych we wspólnej pamięci oraz wywołania funkcji
- Sprawdzona w dużej liczbie prawdziwych, działających systemów

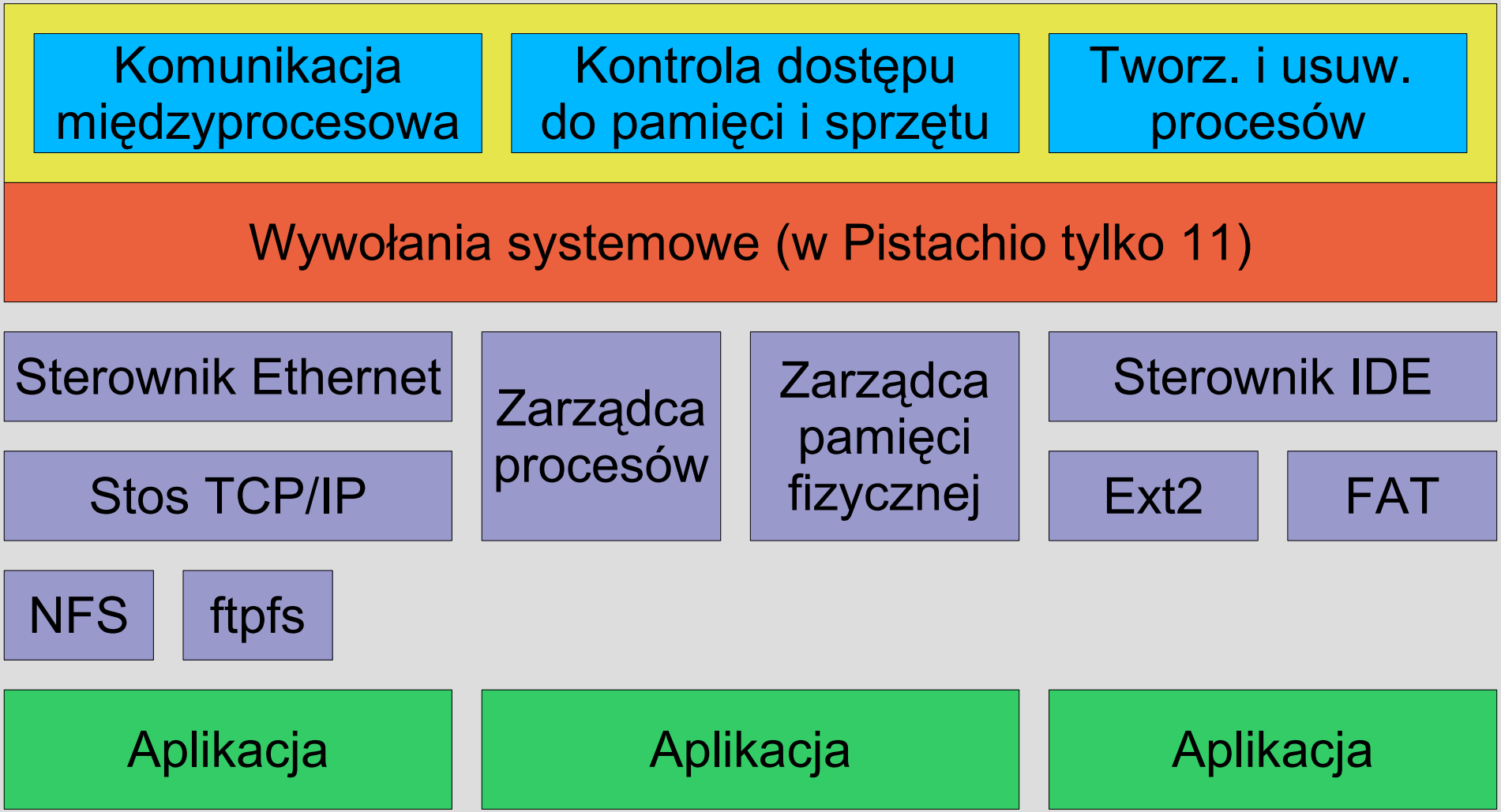
Wady architektury monolitycznej

- Problemy bezpieczeństwa
- Utrudnione pisanie kodu systemowego
- Ograniczone możliwości modyfikacji pracującego systemu
- Brak możliwości rozbudowy systemu przez użytkowników
- Jądro musi przebywać cały czas w pamięci

Architektura z mikrojądrem

- Jądro zawiera jedynie podstawową funkcjonalność: tworzenie i usuwanie wątków i procesów, mechanizmy kontroli dostępu do pamięci i sprzętu, komunikację międzyprocesową
- Wszystkie inne funkcje systemu realizowane są przez procesy w przestrzeni użytkownika, komunikujące się przez mechanizmy mikrojądra

Architektura z mikrojądrem



Zalety architektury z mikrojądrem

- Separacja uprawnień poszczególnych składników systemu
- Łatwa modyfikacja i naprawa pracującego systemu bez przestojów
- Uproszczone programowanie składników systemu – są one normalnymi procesami
- Możliwość budowy systemów heterogenicznych

Wady architektury z mikrojądrem

- Niższa wydajność – komunikacja wymaga przełączenia kontekstu i/lub kopiowania
  - Architektura sprzętowa projektowana pod kątem mikrojąder mogłaby poprawić wydajność
- Niewystarczająco sprawdzone w praktyce

Notka historyczna

- Mikrojądro Mach
  - Asynchroniczne, obrzydliwie nisko wydajne IPC
  - Wbudowane sterowniki sprzętu
  - Wbudowany system zarządzania pamięcią
- Systemy oparte na architekturze z mikrojądrem
  - QNX
  - AmigaOS
  - Symbian

Mikrojądra L4

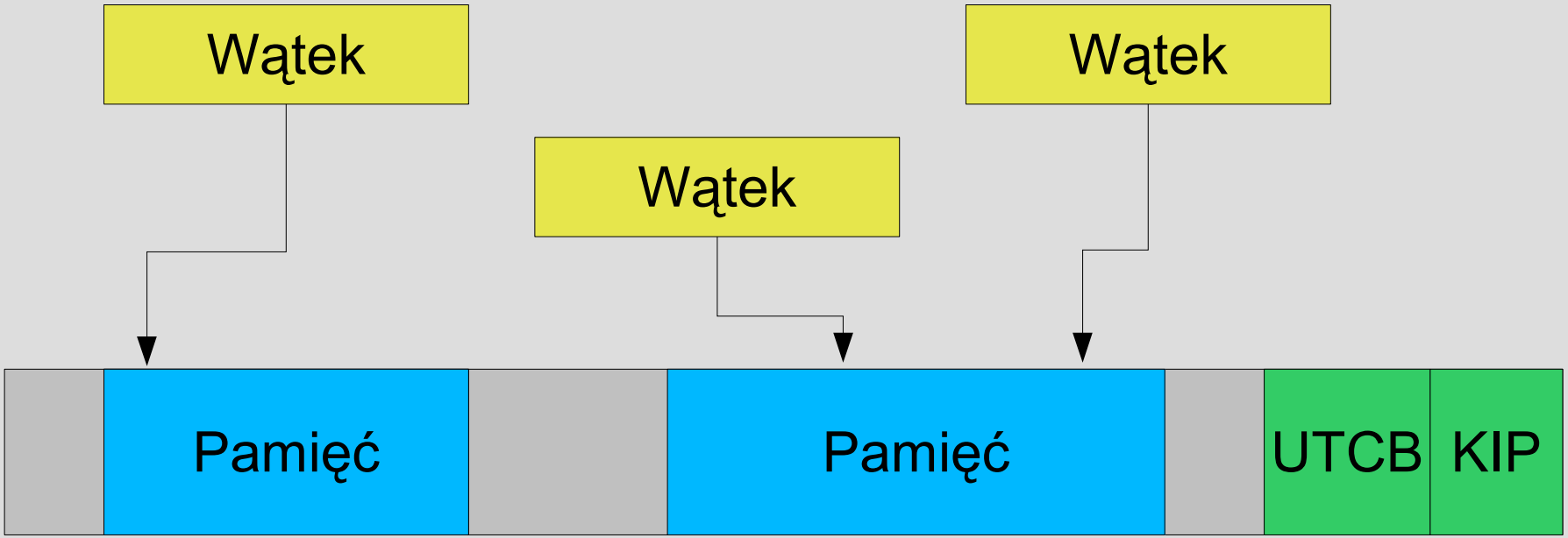
- Pierwsze L4; Lemon Pip (1995)
  - Minimalny kod napisany w asemblerze
  - Bardzo wydajna komunikacja międzyprocesowa
  - Bardzo niewielkie obciążenie cache
- Hazelnut (2001)
  - Hybrydowy kod C++ i asemblerowy
- Pistachio (2004)
  - Usprawnione API
  - Sportowane na wiele architektur sprzętowych (X86, X86-64, Itanium, SPARC, PowerPC...)

API L4Ka::Pistachio

- Metody komunikacji z jądrem
  - Strona interfejsu jądra
  - Rejestry wirtualne
  - Jedenaście wywołań systemowych
  - Protokoły komunikacji z wątkami systemowymi
- Podstawowe pojęcia
  - Przestrzenie adresowe
  - Wątki
  - Komunikacja międzyprocesowa
  - Mapowanie pamięci

Przestrzeń adresowa

- Wirtualna przestrzeń pamięci oraz zbiór pracujących wewnątrz niej wątków
- Zawiera stronę interfejsu jądra (KIP) oraz blok kontrolny wątków (UTCB)



Wirtualna przestrzeń adresowa

Wątek

- Obiekt jądra składający się z identyfikatora (TID) lokalnego i globalnego, stanu rejestrów procesora oraz rejestrów wirtualnych mikrojądra
- Identyfikator wątku służy również za identyfikator przestrzeni adresowej, z którą jest związany

Strona interfejsu jądra

- Zawiera podstawowe informacje o jądrze:
  - Wersja jądra
  - Informacje o procesorach
  - Ograniczenia dokładności zegara
  - Rozmiar strony sprzętowej, KIP oraz UTCB
  - Układ pamięci sprzętowej
- Oraz informacje niezbędne do wykonywania wywołań systemowych

Wywołania systemowe

- Nieuprzywilejowane
  - KernelInterface – wyszukanie strony interfejsu
  - Ipc – wymiana komunikatów
  - ExchangeRegisters – wymiana rejestrów
  - SystemClock – odczytanie zegara
  - ThreadSwitch – uwolnienie procesora
  - Schedule – ustawienia szeregowania
  - Unmap – odmapowanie stron pamięci
- Uprzywilejowane
  - ThreadControl – tworzenie i usuwanie wątków
  - SpaceControl – konfiguracja układu pamięci
  - ProcessorControl – konfiguracja procesorów
  - MemoryControl – ustawianie atrybutów pamięci

ThreadControl

- Uprzywilejowane wywołanie systemowe
- Pozwala na tworzenie i usuwanie wątków w obrębie istniejącej przestrzeni adresowej
- Umożliwia również utworzenie nowej przestrzeni adresowej, którą następnie należy skonfigurować wywołaniem SpaceControl
- Pozwala zmienić wątek szeregujący oraz stronicujący

SpaceControl

- Uprzywilejowane wywołanie systemowe
- Pozwala ustalić położenie UTCB oraz KIP w nowo utworzonej przestrzeni adresowej
- Umożliwia ustawienie wątka przekierowującego dla danej przestrzeni adresowej
  - Wątek ten kontroluje całą komunikację między przestrzenią adresową, do której jest przydzielony, a innymi przestrzeniami adresowymi

Ipc

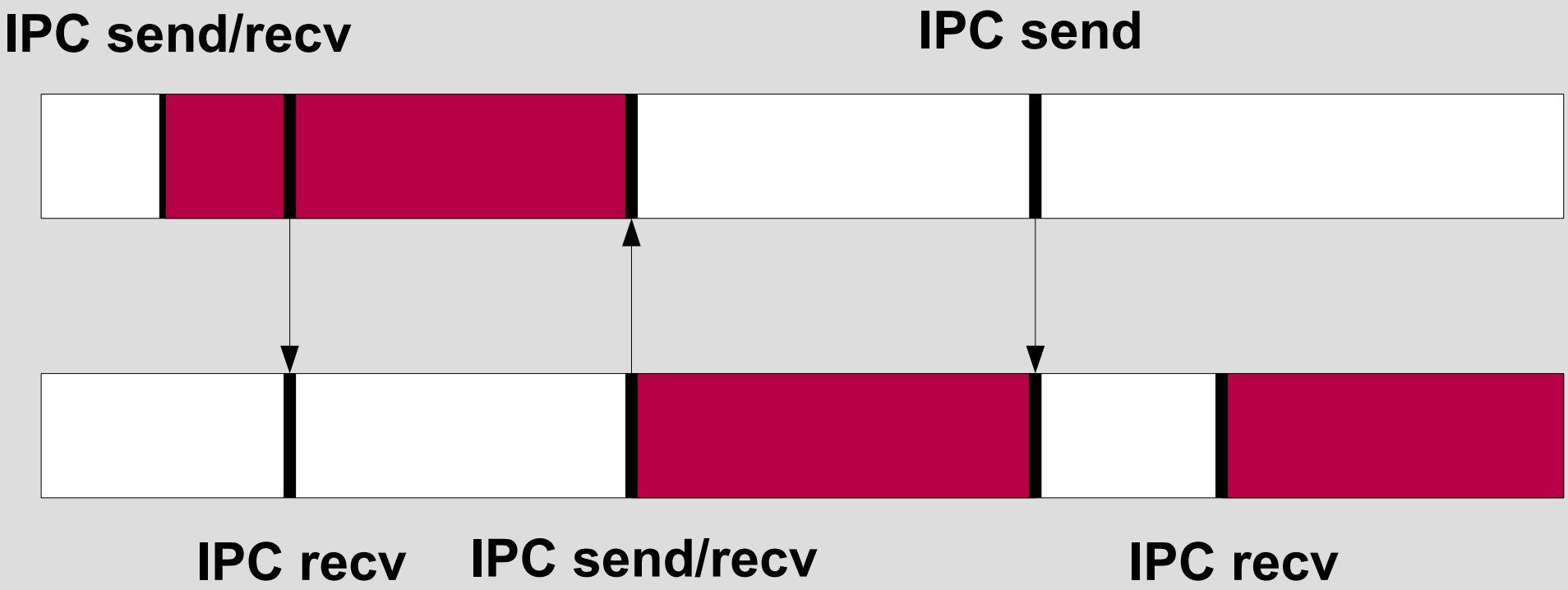
- Najważniejsze wywołanie systemowe L4Ka::Pistachio
- Cała komunikacja między wątkami, zarządzanie pamięcią, obsługa przerwań i wyjątków sprzętowych odbywa się poprzez IPC
- Trzy rodzaje przesyłanych informacji
  - Pojedyncze rejestry
  - Zawartość obszaru pamięci
  - Mapowanie pamięci w przestrzeni adresowej

Ipc – schemat komunikacji

- Komunikacja jest synchroniczna: przesył danych odbywa się tylko wtedy, gdy obie strony wykonają wywołanie Ipc
- Składa się z dwóch opcjonalnych faz: nadawania i odbierania
- Każda z faz może być: blokująca, nieblokująca lub blokująca z ograniczeniem czasowym

Ipc – schemat komunikacji

- Przykładowa sesja komunikacji pomiędzy dwoma wątkami



Ipc – przesył rejestrów

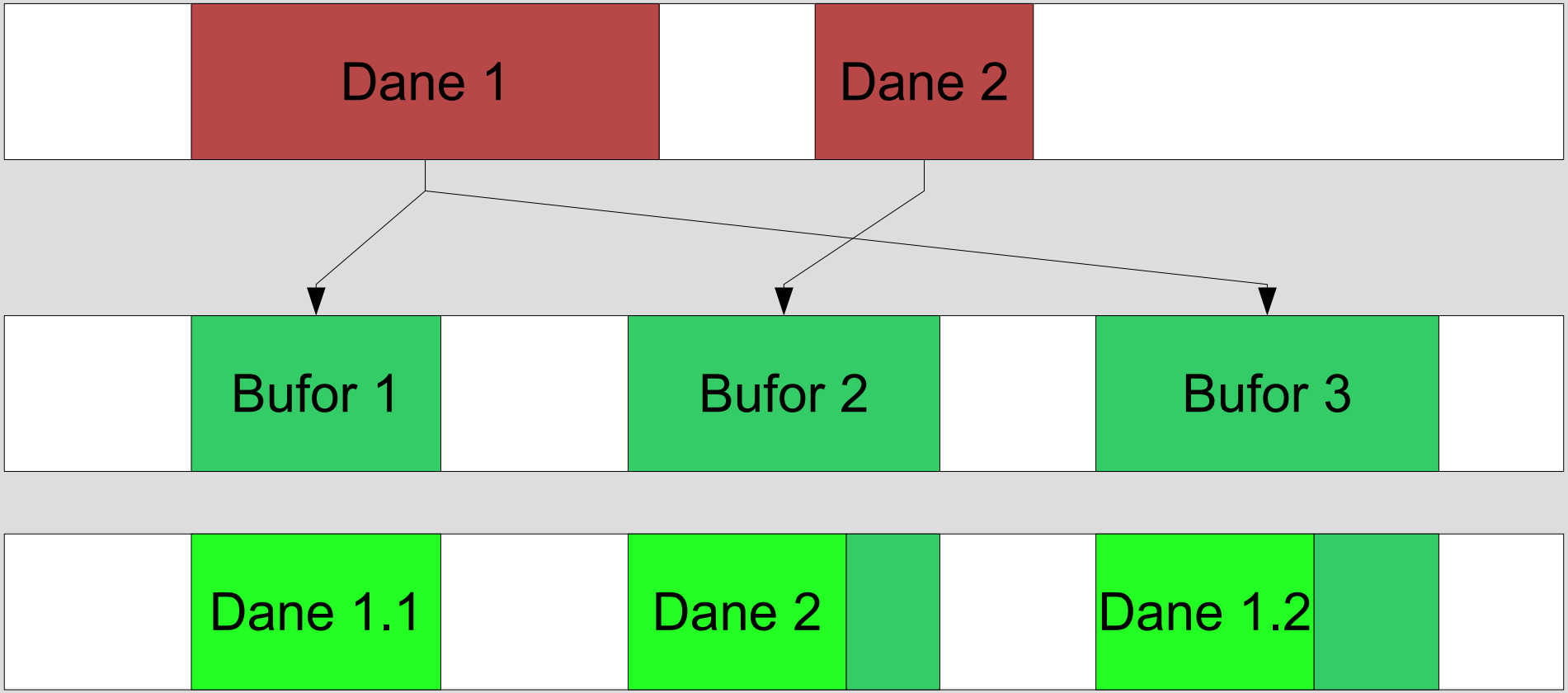
- Nadawca umieszcza dane w wirtualnych rejestrach MR (Message Register) i wywołuje Ipc
- Odbiorca otrzymuje kopię tych danych do własnych rejestrów MR
- Rejestry MR mogą być, w zależności od architektury, obsługiwane przez rejestry procesora lub obszar pamięci w UTCB

Ipc – przesył buforów

- Nadawca deklaruje obszary pamięci, z których będą kopiowane dane
- Odbiorca deklaruje, gdzie dane zostaną zapisane przy odbiorze
- Obszary pamięci nie muszą być spójne – jądro realizuje scatter&gather
- Odbiorca i nadawca deklarują maksymalny czas trwania transferu

Ipc – przesył buforów

- Przykład kopiowania danych pomiędzy dwoma przestrzeniami adresowymi

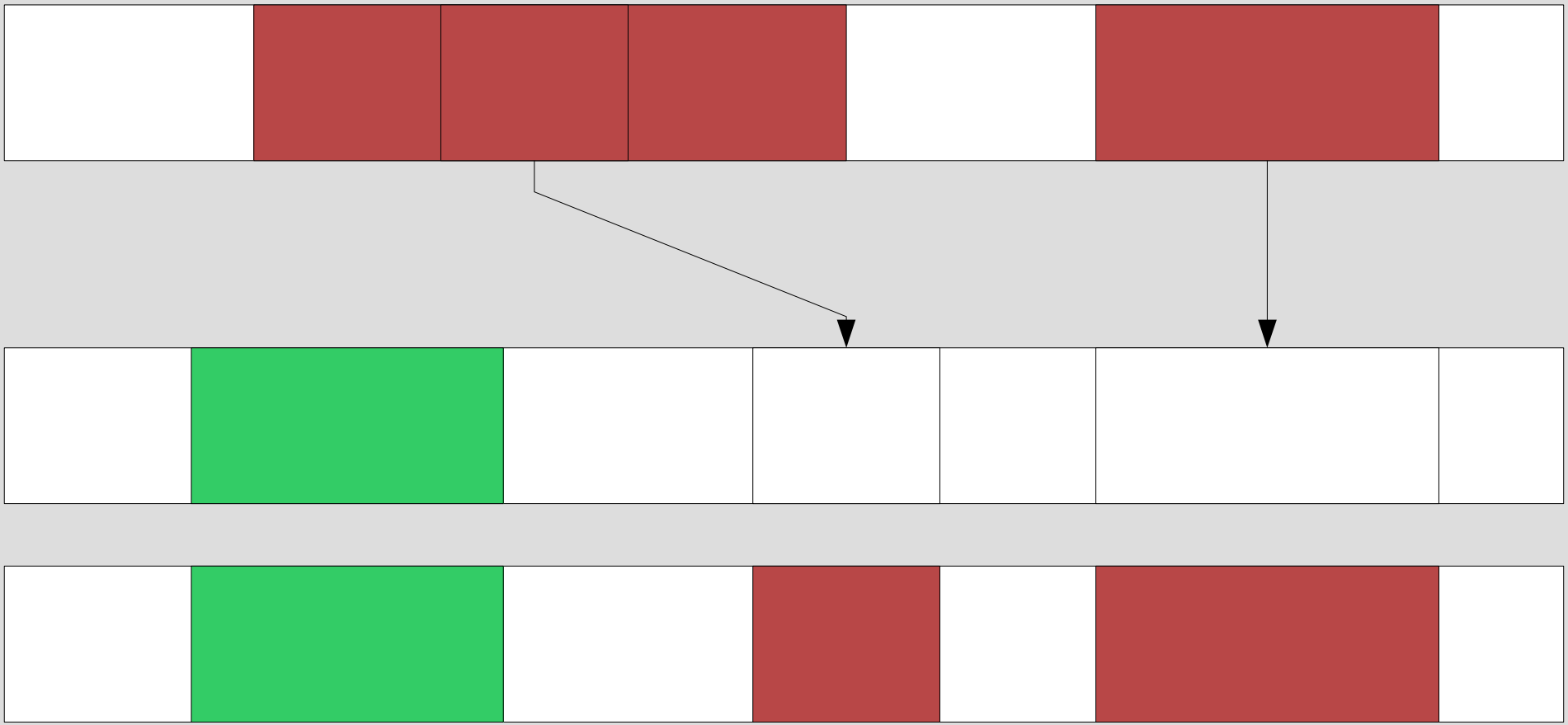


Ipc – mapowanie pamięci

- Część pamięci nadawcy jest mapowana w przestrzeni adresowej odbiorcy
- Nie kopiowanie, lecz współdzielenie
- Nadawca może wybrać uprawnienia dostępu (odczyt, zapis, wykonanie) do mapowanego obszaru
- Ograniczenie położenia i rozmiaru mapowanej pamięci: dwójkowa wielokrotność strony pamięci
- Oddanie pamięci (grant) – fragment pamięci jest po operacji odmapowywany od nadawcy

Ipc – mapowanie pamięci

- Przykład operacji mapowania pamięci

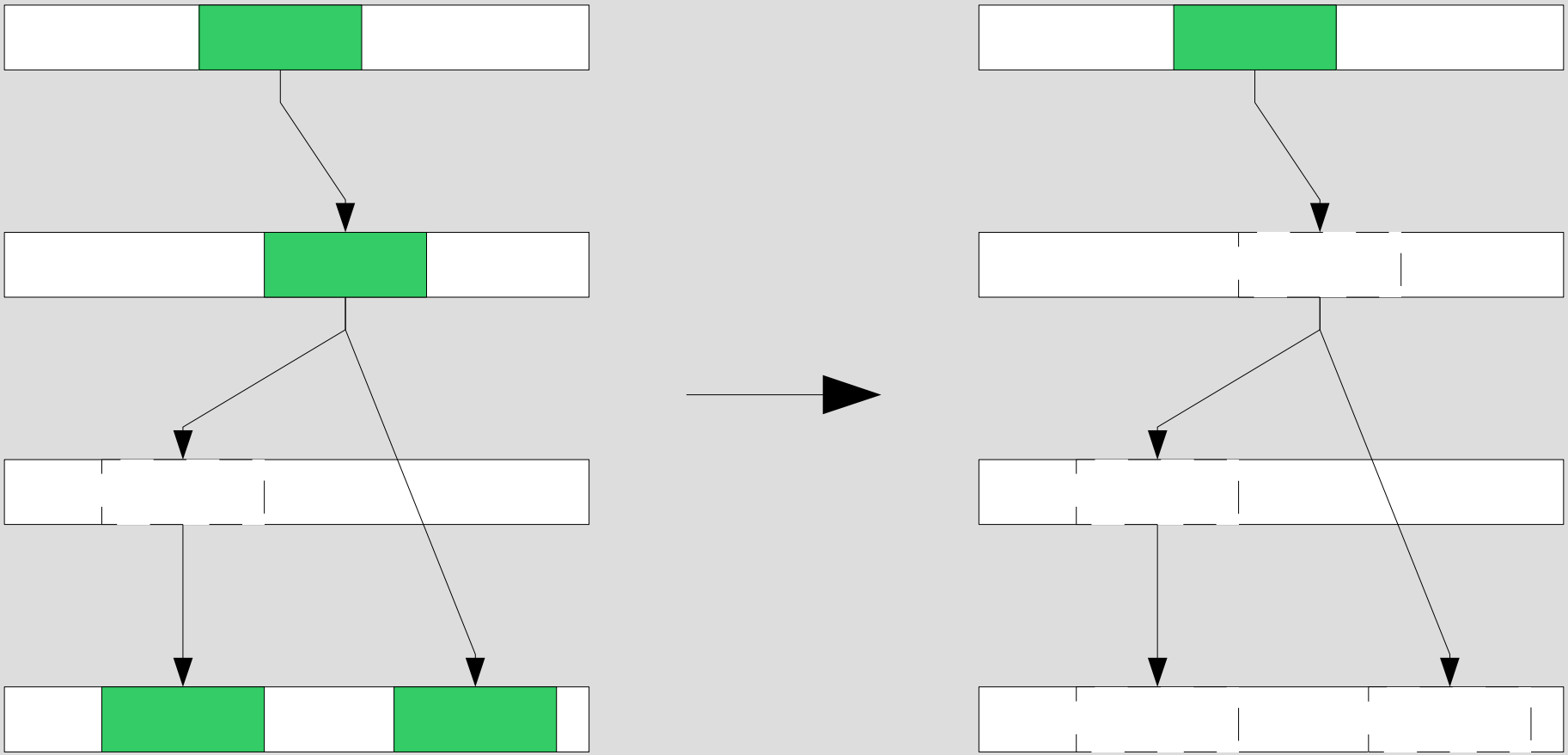


Unmap

- Odmapowanie fragmentu pamięci od wszystkich przestrzeni adresowych, którym została zmapowana od aktualnej przestrzeni (i rekursywnie)
- Opcjonalnie można też odmapować pamięć również od własnej przestrzeni adresowej

Unmap

- Przykład działania operacji Unmap



Stronicowanie

- Każdy wątek posiada przydzielony wątek stronicujący
- W przypadku błędu stronicowania jądro komunikuje to wątkowi stronicującemu za pomocą IPC
- Wątek stronicujący odpowiada, mapując właściwą pamięć

Zarządzanie pamięcią fizyczną

- Przy starcie systemu prawie całą pamięć fizyczną otrzymuje wątek sigma0
- Zadaniem sigma0 jest oddanie pamięci fizycznej systemowi operacyjnemu
- Sigma0 jest wątkiem stronicującym wątku głównego systemu
- Wątek główny pobiera pamięć od sigma0 i rozdziela ją innym składnikom systemu

Wyjątki sprzętowe

- Każdemu wątkowi przyporządkowany jest wątek obsługi wyjątków
- W przypadku wystąpienia wyjątku sprzętowego (błędu wykonania) wątek obsługi wyjątków otrzymuje komunikat zawierający rejestr IP i opcjonalnie inne rejestry
- Wątek obsługi wyjątków odpowiada komunikatem zawierającym nowe wartości rejestrów

Obsługa przerwań

- Każdemu numerowi przerwania przyporządkowany jest pewien identyfikator wątku (TID)
- Jeśli wątek przerwania ma przydzielony wątek stronicujący, to ten wątek będzie otrzymywać komunikaty IPC o przerwaniach
- Uprawnienia dostępu do zasobów sprzętowych (np. porty I/O) są przekazywane poprzez IPC

Szeregowanie

- Każdemu wątkowi jest przyporządkowany wątek szeregujący
- Wątek szeregujący może używać wywołania Schedule na podległych mu wątkach
  - Zmiana długości kwantu czasu
  - Ustawienie maksymalnego czasu wykonania
  - Ustawienie docelowego procesora (SMP)
  - Zmiana priorytetu
- Wyczerpanie przydzielonego czasu jest sygnalizowane wątkowi szeregującemu

Niedostatki Pistachio

- Rozmiar bloku UTCB stały – maksymalna liczba wątków przypadająca na przestrzeń adresową jest niezmienna
- Brak mechanizmu wymiany pamięci pomiędzy mikrojądrem a przestrzenią użytkownika

Zastosowania: HURD

- GNU HURD – system operacyjny kompatybilny z Uniksem, zaprojektowany z myślą o rozszerzalności
- System podzielony na serwery: procesów, autentyfikacji, haseł, systemów plików
- Montowanie poprzez translatory
- Aktualnie oparty na Mach, trwa eksperymentalny port na Pistachio

Zastosowania: Mungi

- System operacyjny z pojedynczą przestrzenią adresową
  - Przestrzeń adresowa podzielona jest na ciągłe obszary – obiekty
  - Wprawdzie przestrzeń jest wspólna, ale różne procesy mają uprawnienia do różnych obiektów
  - Obiekty mogą być trwałe – pozostają w systemie nawet po restarcie
- Nie ma systemu plików!
  - Rolę plików pełnią obiekty trwałe
  - System katalogowy pozwala odnajdywać obiekty

Zastosowania: wirtualizacja

- Kilka niezależnych systemów może pracować jednocześnie na jednym mikrojądrze
- Projekt Marzipan: zarządca wirtualizacji
  - Rozdziela zasoby pomiędzy maszyny wirtualne
  - Kontroluje dostęp do uprzywilejowanych wywołań systemowych
- Projekt Afterburner: prewirtualizacja – wydajność dorównująca parawirtualizacji bez modyfikacji kodu systemu

Bibliografia

- Specyfikacja API L4 X.2 oraz publikacje na stronie http://www.l4ka.org/
- „Podstawy systemów operacyjnych” A.Silberschatz, P.Galvin