W1

Dr inż. hab. Siergiej Fialko, IMK-PK,

https://torus.uck.pk.edu.pl/~fialko.sergiy/
sfialko@pk.edu.pl

dr hab inz Sergiy
Fialko
IMK
p. 144/7

Education | Research | Examin. | ????

email: sfialko@pk.edu.pl

Ogloszenia:

Konsultacje

Konsultacje, prowadzone jawnie - od oznaczonej godziny czekam 15 min. Jesli nikt nie przyszedl - uwazam, ze konsultacja nie odbyla sie.

Dzien	Godziny	Tryb	Status
Wtorek	16:15 - 17:15	MS TEAMS	
Czwartek	11:00 - 12:00	p. 144/7	

Materialy Dydaktyczne:
Linki aktualne:

- Modelowanie Zagadnien Technicznych (MZT1.html)
- JiPP cz. I. Studia stacjonarne i niestacjonarne (CC.html)
- JiPP cz. II. Studia stacjonarne i niestacjonarne. (CPP_z_new.html)

Contact Sergiy Fialko

Wyklady:

Kurs MZT: http:\\torus.uck.pk.edu.pl\~pk21

- Kurs MZT(MZT.pdf)
- w1 (w1.pdf)
- w2 (w2.pdf)
- w3 (w3.pdf)
- w4 (w4.pdf)
- w5 (w5.pdf)
- w6 (w6.pdf)
- w7 (w7.pdf)
- w8 (w8.pdf)
- w9 (w9.pdf)
- w10 (w10.pdf)
- w10_1 (w10_1.pdf)

Przyklady do wykladow:

- MemTest_1 (MemTest_1.zip)
- MemTest_2 (MemTest_2.zip)
- Zrownoleglenie petli for za pomoca OpenMP (parallel_1.zip)
- MultThreads (MultThreads.pdf)
- DoesNotDoIt (DoesNotDoIt.zip)
- SendMess (SendMess.zip)
- MatrixMultipl (Mulm_Stud.7z)

Laboratoria: Studia stacjonarne

- Lab_1 (Lab_1_2.7z)
- Lab_2 (Lab_2.zip)
- Lab_3 32 bit application (Lab_3_1_i_ia32.7z)
- Lab_3 64 bit application (Lab_3_1_i_x64.7z)
- Lab_3 64 bit application AVX(Lab_3_1_x64_AVX.7z)
- Lab_3 64 bit application AVX FMA3(Lab_3_1_x64_AVX_FMA.7z)
- CPU_Z_x64 (cpuz64_153.zip)
- CPU_Z_ia32 (cpuz_153.zip)
- Obliczenie dot = X'*Y w trybie wielowatkowym (Lab_dot_prod.zip)
- Obliczenie calki w trybie wielowatkowym (Lab_integr.7z)
- Obliczenie C = A*B w trybie wielowatkowym (L6.doc)
- Macierze rzadkie: format skompresowany (SparseMatr.7z)
- Macierze rzadkie: format skompresowany (SparseMatr_mul.zip)

Warunki zaliczenia:

1. **Zaliczenie wszystkich prac laboratoryjnych na podstawie obrony każdej pracy.**
2. **Napisanie kolokwium zaliczeniowego, który obejmuje ostatnie dwa tematy.**

Osobliwości przedmiotu

- **W podanym kursie główna uwaga będzie przydzielona osobliwościom modelowania komputerowego zagadnień technicznych, czyli aspektom informatycznym, który powstają przy tworzeniu systemów obliczeniowych w zagadnieniach technicznych.**

- **Celą kursu jest zapoznanie z technikami i metodami opracowania algorytmów, często spotykanych przy modelowaniu zagadnień technicznych, przy czym główna uwaga przydzielona podstawom tworzenia takich metod na wielordzeniowych komputerach z pamięcią wspólną.**

Kurs obejmuje:

- Osiągnięcie wysokiej wydajności typowych algorytmów obliczeniowych zagadnień technicznych przy oprogramowaniu sekwencyjnym, a również wielowątkowym dla komputerów wielordzeniowych (SMP) .

- Typowe techniki oprogramowania wielowątkowego dla algorytmów obliczeniowych, a również zrównoleglenia SIMD (wektoryzowanie obliczeń) oraz wspieranie potokowości (pipelines) współczesnych procesorów.

- Macierzy rzadkie. Metody bezpośrednie i iteracyjne rozwiązywania układów równań liniowych algebraicznych z macierzami rzadkimi symetrycznymi.

Osobliwości kursu:

1. **Kurs zawiera elementy matematyki – algebry liniowej, algebry macierzy, teorii grafów, metod numerycznych, a również wymaga wiedzy z podstaw oprogramowania w języku C, C++, systemów operacyjnych i oprogramowania wielowątkowego.**

2. **Brak podręczników. Źródła literaturowe są rozrzucane po wielu książkach (podręcznikach z matematyki, metod numerycznych, podręcznikach z informatyki), artykułach w czasopismach naukowych, stronach www .**

Miara wydajności:

FLOPS = (ilość operacji zmiennoprzecinkowych)/(czas wykonania)

Jest naturalną miarą dla algorytmów, które powstają z zagadnień naukowo-technicznych (dominantowymi są operacji arytmetyczne zmiennoprzecinkowe), przecież nie jest przydatna do oceny wydajności innych rodzajów algorytmów (kompilatory, bazy danych, itp.)

Operacje zmiennoprzecinkowe mogą być szybkimi i wolnymi. Stosunek trwałości różnych operacji zmiennoprzecinkowych:

Operacje zmiennoprzecinkowe	Stosunkowa trwałość
dodawanie, odejmowanie, porównanie, mnożenie	1
dzielenie, obliczenie pierwiastka kwadratowego	4
eksponenta, sinus, ...	8

Pamięć główna systemu równoległego może być wspólną (shared memory) lub rozproszoną (distributed memory).

Charakterystyki pamięci:

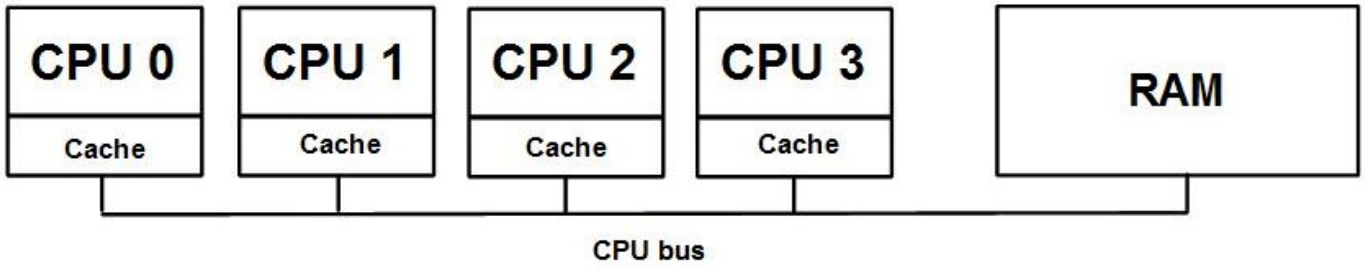
- **Obiętość (MB), Latentność (Latency – ang.) (ns), Przepustowość (b/s).**
- **Wielopoziomowa pamięć podręczna (cache memory) – szybka pamięć dla przechowywania tymczasowych danych.**
- **bufory rozkazów i danych.**

<u>Wspólna pamięć</u> (SMP – symmetric multiprocessing) – procesory mają równoprawny dostęp (każdy z procesorów nie ma żadnej przewagi nad innymi) i równoznaczny (czas dostępu do dowolnej części pamięci jest taki samy).

Zalety: związek między procesami i synchronizacja są proste, oprogramowanie mniej skomplikowane, niż w wypadku architektury z pamięcią rozproszoną.

Wady: mają podstawowe ograniczenie na ilość procesorów – krytycznym jest przepustowość magistrali.

<u>Wspólna pamięć.</u> Wieloprocesorowe systemy z jednorodnym dostępem do pamięci - UMA (Uniform Memory Access)



System symetryczny wieloprocesorowy SMP (Symmetrical Multiprocessing) UMA

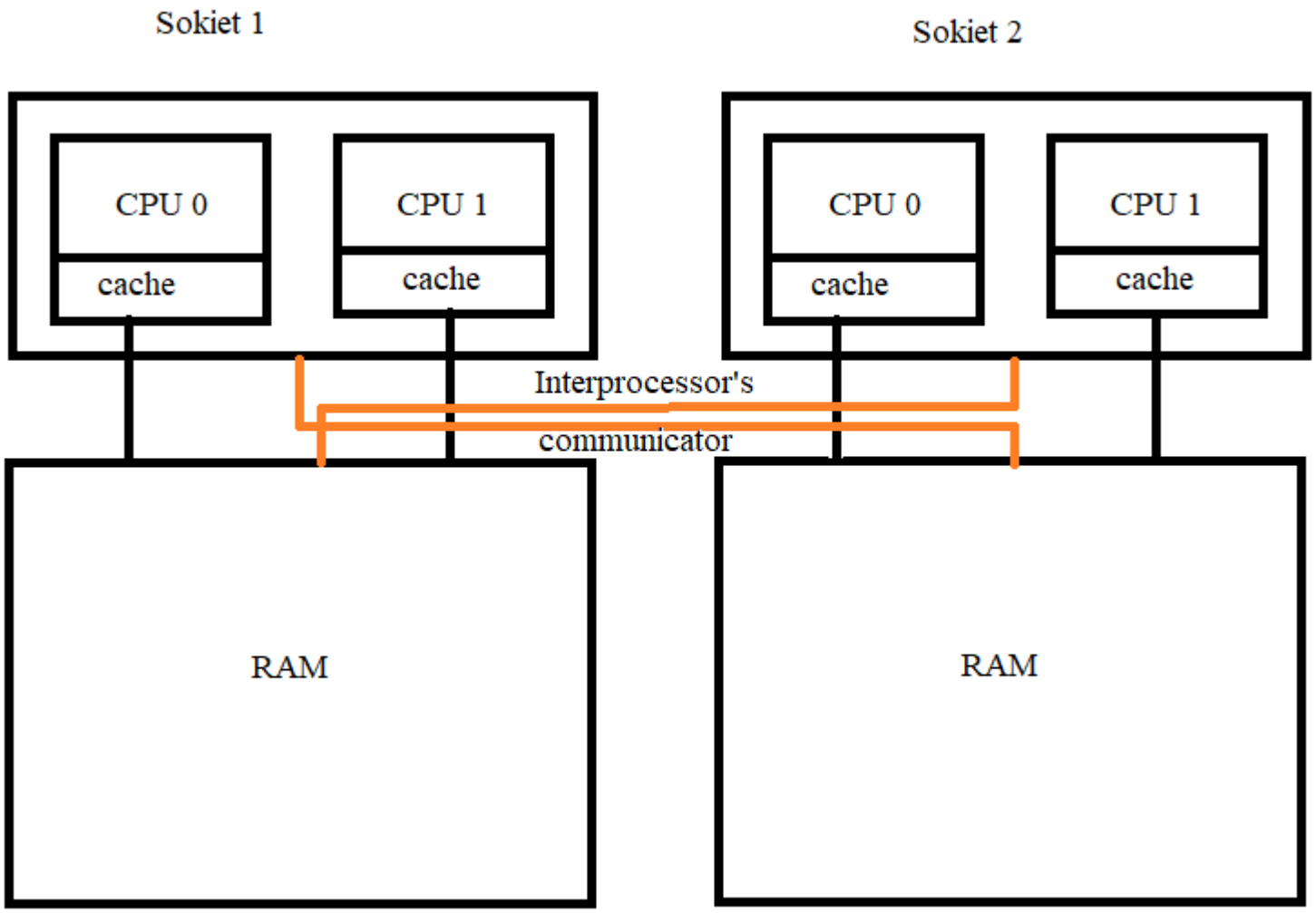
Wieloprocesorowe systemy z niejednorodnym dostępem do pamięci - NUMA (Non - uniform Memory Access)

Wielu procesorów są łączone jedną magistralą z tą samą pamięcią główną.

Technika zrównoleglenia dla SMP systemów zwykłe polega na wielowątkowości.

Stan pamięci podręcznej każdego procesora ma spełnić warunku spójności danych.

Wspólna pamięć. Wieloprocesorowe systemy z jednorodnym dostępem do pamięci - NUMA (Non-Uniform Memory Access)



Wielu procesorów, każdy ma własną fizyczną pamięć, dostęp do której jest szybki, i wspólną wirtualną pamięć. Jeśli procesor adresuje się do pamięci innego procesora, generuje się page fault, i jak obsługa sytuacji wyjątkowej, załadowuje się strona pamięci odległej. Dostęp do pamięci odległej jest wolny.

Cache-konflikty

- Przy czytaniu danych (słowo, bajt) procesor najpierw sprawdza stan swojej pamięci podręcznej. Jeśli te dane są w pamięci podręcznej (read hit – trafienie), oni będą umieszczone w rejestry procesora. Jeśli tych danych (słowa, bajta) niema w pamięci podręcznej (read miss – chybienie), będzie przeładowana cała cache linia pamięci podręcznej.

- Przy pobraniu tych danych (bajta, słowa) z pamięci głównej będą pobrane nie tylko to słowo, bajt, a i bajty sąsiednie tak, żeby wypełnić całą cache linie pamięci podręcznej.

- W zależności od typu procesora pamięć podręczna składa się z linii o kilku słów (32 B, 64 B, 128 B). Dane z pamięci głównej będą umieszczone w pamięci podręcznej przed umieszczeniem w rejestrach procesora. Mówimy o odwzorowaniu bloków pamięci głównej w cache-linie pamięci podręcznej.

- Dane, które będą umieszczone w cache-linie, muszą być wyrównane po granicach, wielokrotnych 32 B, 64 B, 128 B odpowiednio.

- **Ideologia cache-linii polega na tym, że zadania najczęściej pracują ze zbiorem sąsiednich słów (bajtów), więc przy odczycie tych słów (bajtów) często wynika stan read hit, i procesor pobiera te dane w rejestry z cache-linii, a nie przez wolny kanał magistral – pamięć główna.**

- **Przecież cache-linii istotnie komplikują odnowienie pamięci w środowisku wieloprocesorowym z pamięcią wspólną**

Przykład

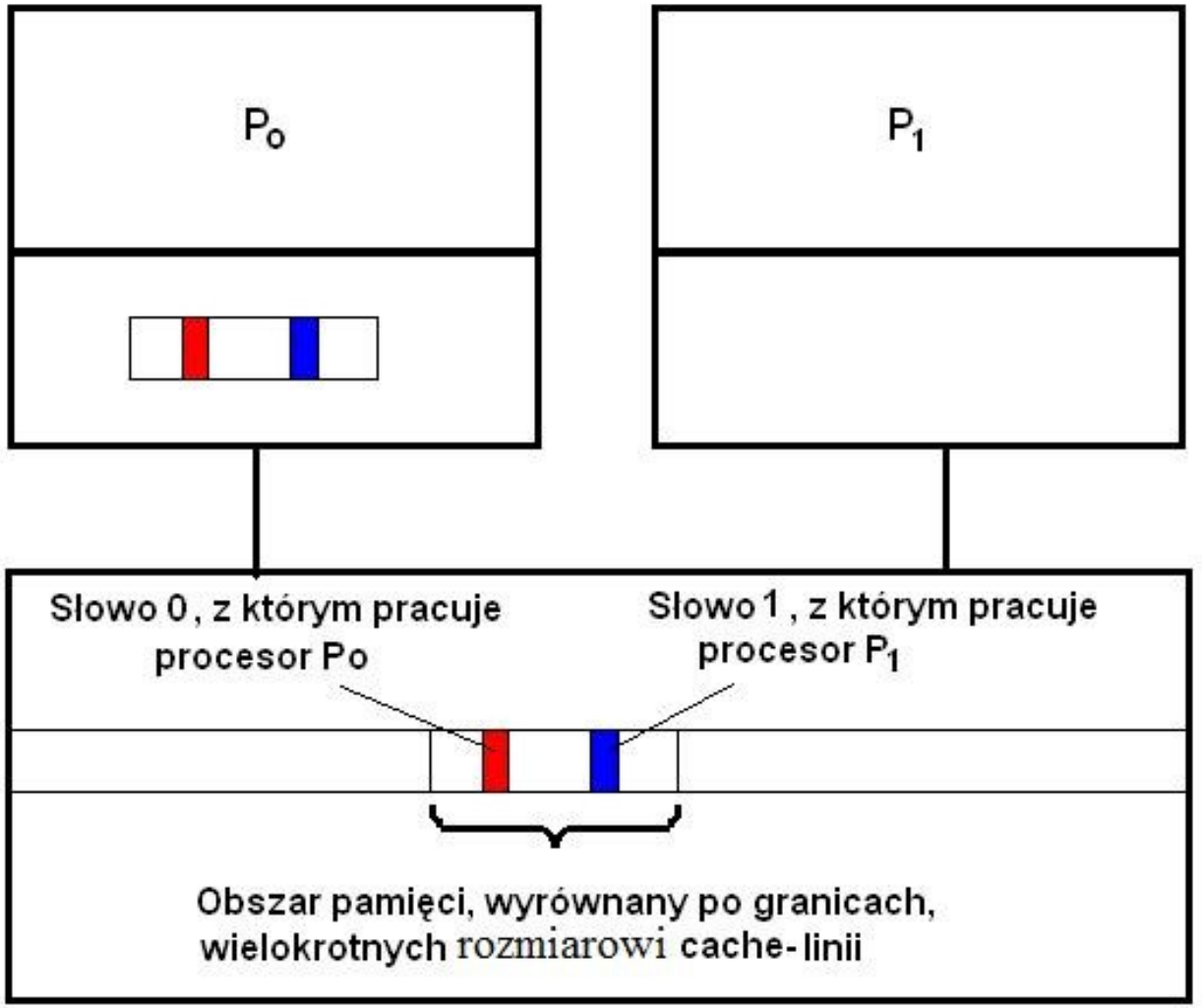
1. **Procesor P0 odczytuje słowo 0, a również i słowa sąsiedni w swój cache**

2. **Procesor P1 odczytuje słowo 1, a również i słowa sąsiedni w swój cache, przy czym słowo 0 procesora P0 w skutek wyrównania po granicach, wielokrotnych rozmiary cache-linii, znajduje się w cache-linii procesora P1 też.**

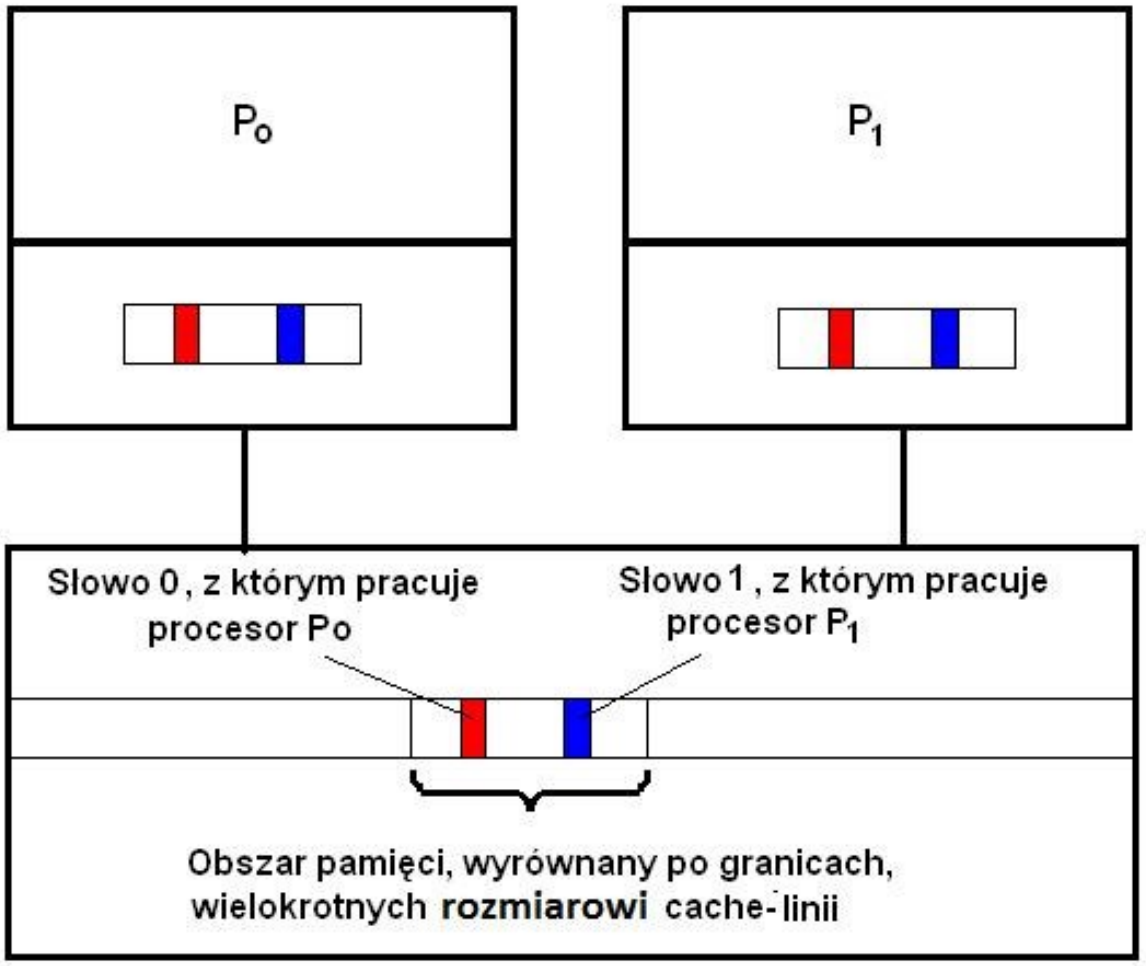
3. **Procesor P0 modyfikuje słowo 0 i umieszcza ten wynik w swój cache. Wartość tego słowa w pamięci głównej na razie pozostaje bez zmian.**

4. **Procesor P1 nic nie wie o tym, że zawartość słowa 1 podległa zmianie. Teraz wartość słowa 0 zależy od tego, cache którego procesora będzie wyładowany w pamięć główną później.**

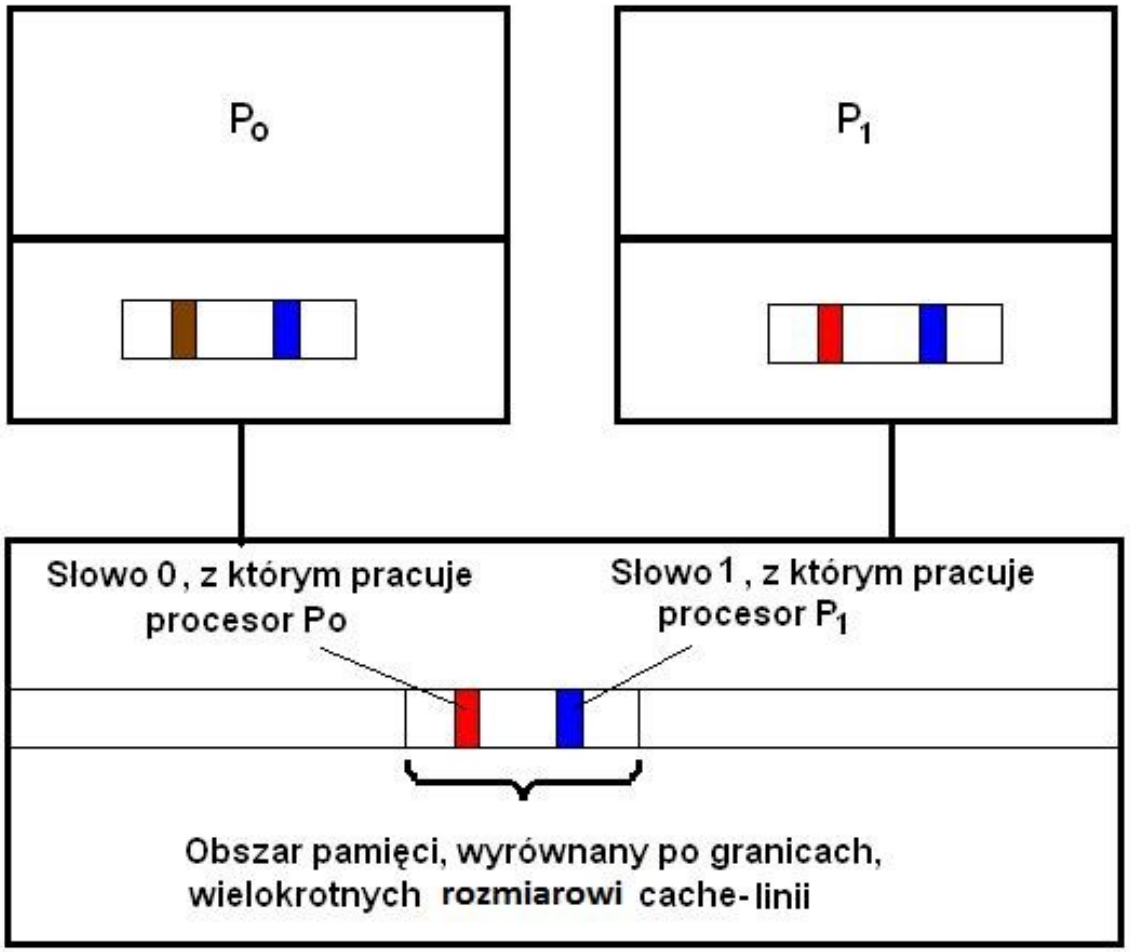
Procesor P0 odczytuje słowo 0, a również i słowa sąsiedni w swój cache



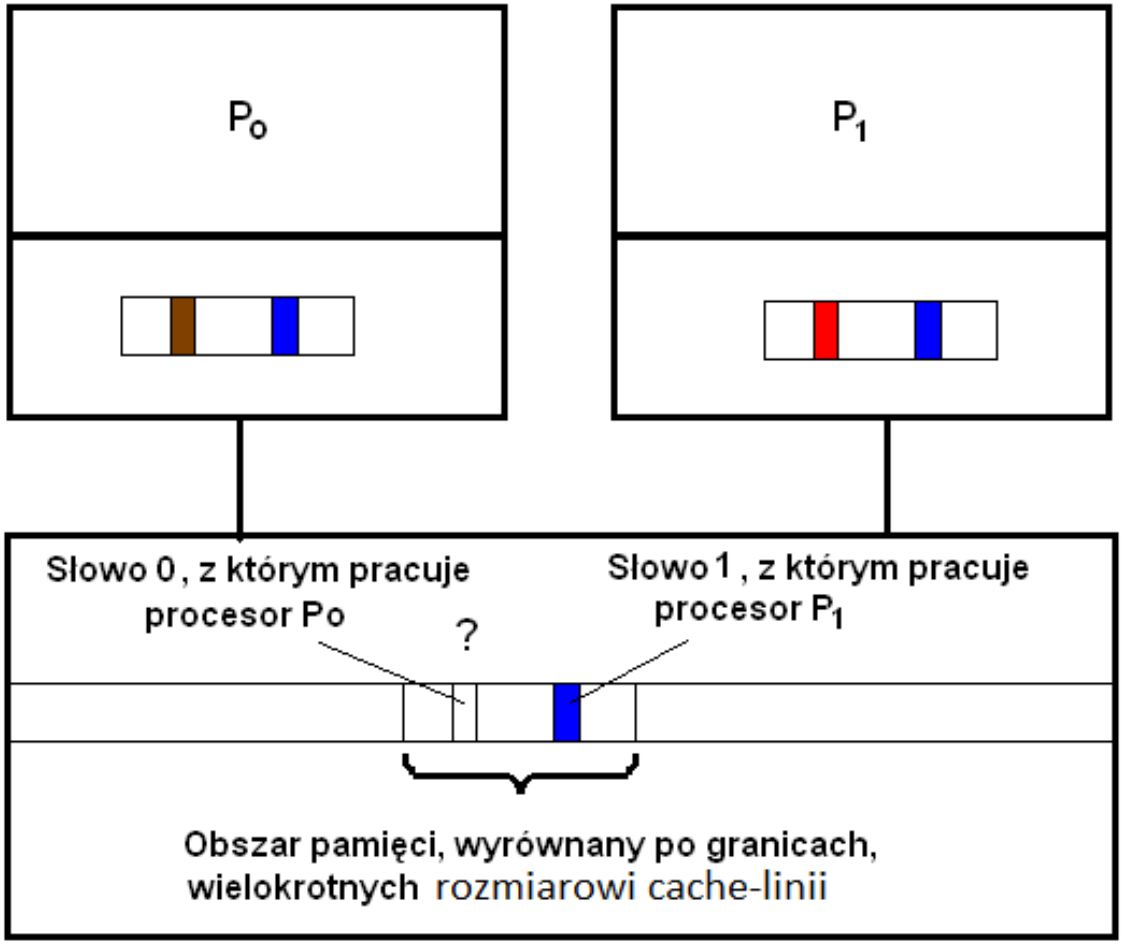
Procesor P1 odczytuje słowo 1, a również i słowa sąsiedni w swój cache, przy czym słowo 0 procesora P0



Procesor P0 modyfikuje słowo 0 i umieszcza ten wynik w swój cache. Wartość tego słowa w pamięci głównej na razie pozostaje bez zmian.



Procesor P1 nic nie wie o tym, że zawartość słowa 1 podległa zmianie. Teraz wartość słowa 0 zależy od tego, cache którego procesora będzie wyładowany w pamięć główną później.



Taki scenariusz był by katastrofą. Producenci sprzętu komputerowego bardzo dobrze wiedzą o tym, i dla tego rzeczywisty komputer nigdy tak nie działa.

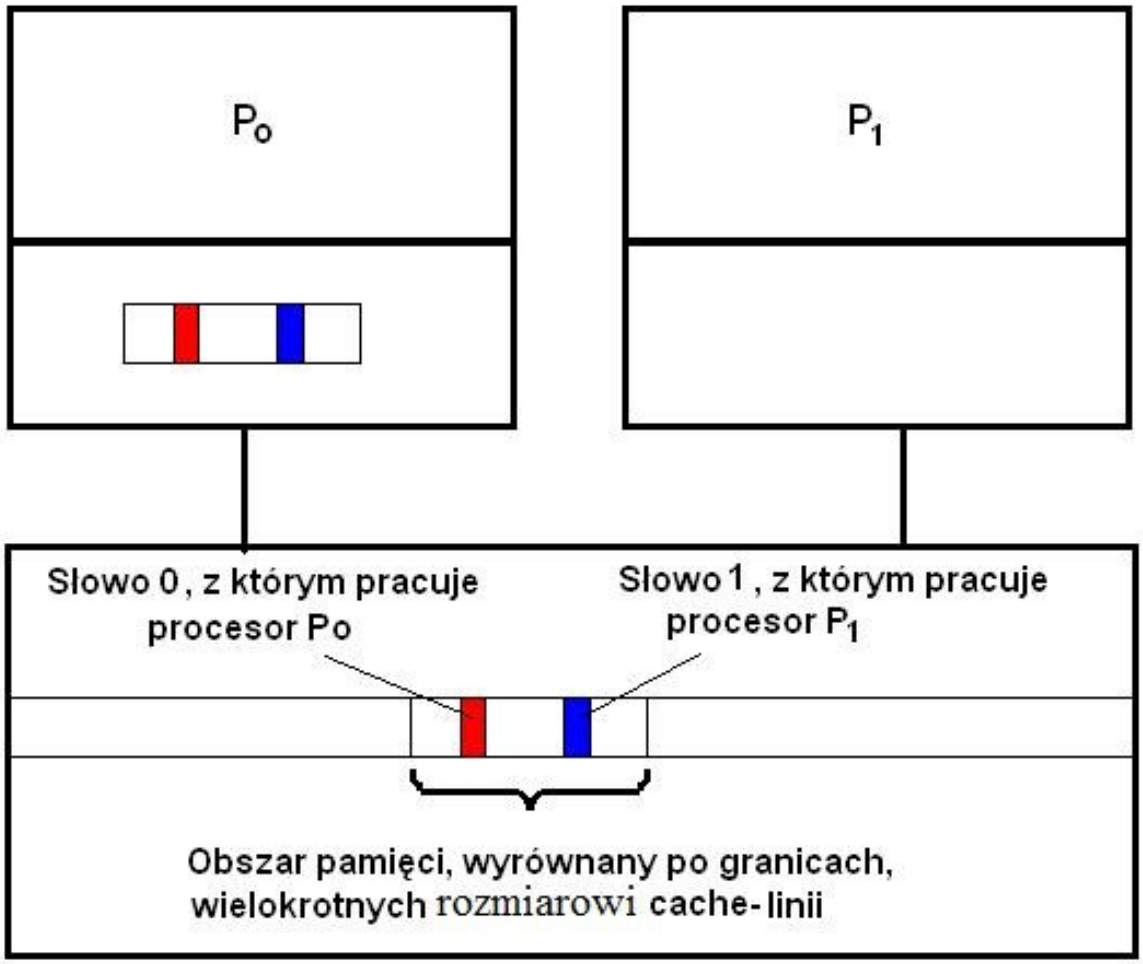
1. Procesor P0 odczytuje słowo 0, a również i słowa sąsiedni w swój cache

2. Procesor P1 odczytuje słowo 1, a również i słowa sąsiedni w swój cache, przy czym słowo 0 procesora P0 w skutek wyrównania po granicach, wielokrotnych rozmiary cache-linii, znajduje się w cache-linii procesora P1 też.

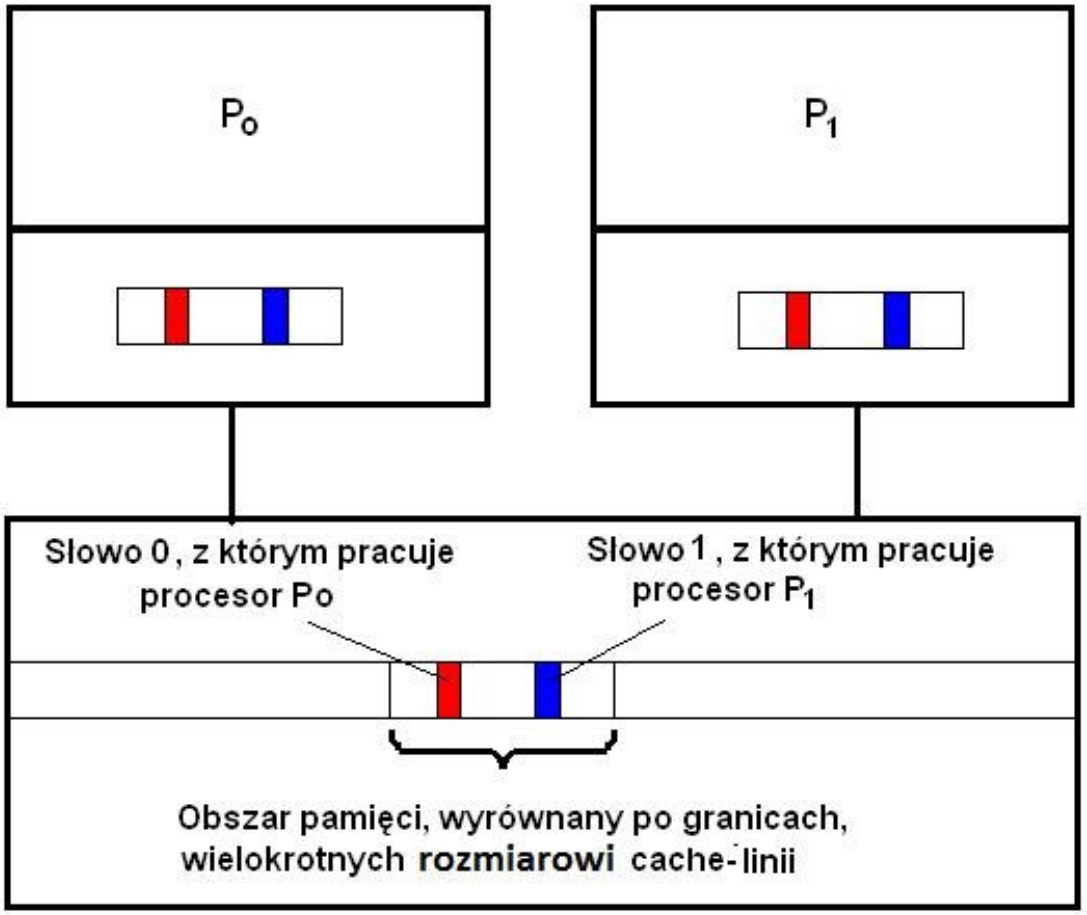
3. Procesor P0 modyfikuje słowo 0 i umieszcza ten wynik w swój cache. Wartość tego słowa w pamięci głównej na razie pozostaje bez zmian, ale procesor P1 w skutek snooping'u zgłasza zawartość swojego cache unieważnioną.

4. Procesor P0 wyładuje zawartość swojego cache w pamięć główną, a procesor P1 ponownie załadowuje swoją cache-linie. Teraz dane w pamięci głównej i w cache-liniach procesorów P0, P1 są spójne (koherentne). Przecież płacą za tą spójność – nadliczbowe przeładowania cache-linij procesorów P0, P1.

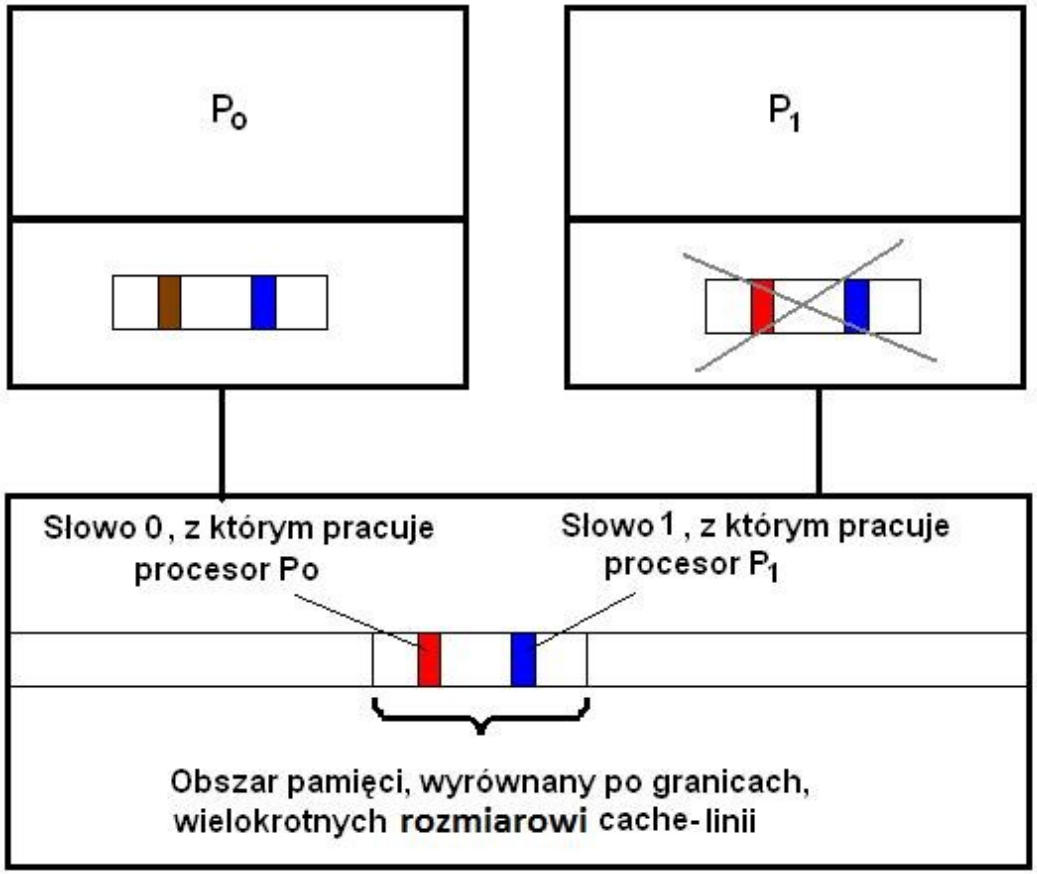
Procesor P0 odczytuje słowo 0, a również i słowa sąsiedni w swój cache



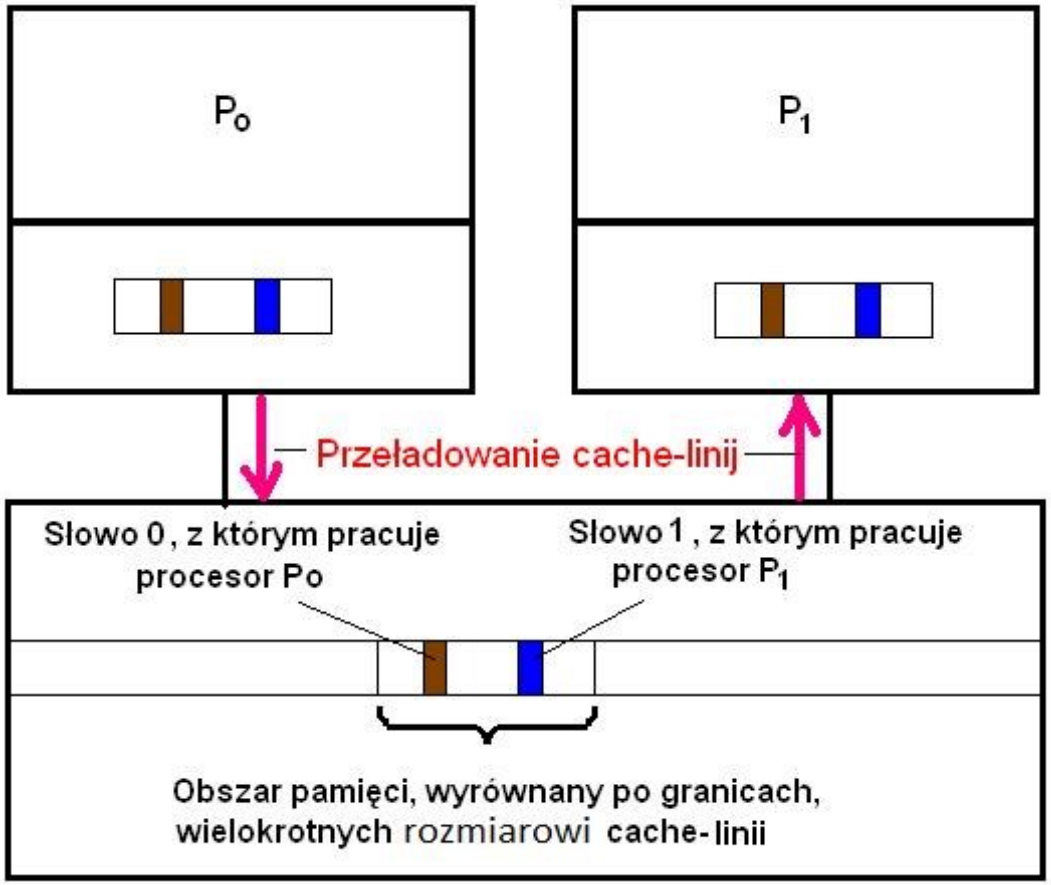
Procesor P1 odczytuje słowo 1, a również i słowa sąsiedni w swój cache, przy czym słowo 0 procesora P0



Procesor P0 modyfikuje słowo 0 i umieszcza ten wynik w swój cache. Wartość tego słowa w pamięci głównej na razie pozostaje bez zmian, ale procesor P1 w skutek snooping'u zgłasza zawartość swojego cache unieważnioną.



Procesor P0 wyładuje zawartość swojego cache w pamięć główną, a procesor P1 ponownie załadowuje swoją cache-linie. Teraz dane w pamięci głównej i w cache-liniach procesorów P0, P1 są spójne (koherentne).



W1 **To się okazuje, że obecność pamięci podręcznej, przeznaczonej dla przyspieszenia obliczeń w skutek zmniejszenia ilości odczytów-zapisów bezpośrednich z/w pamięć(i) głównej, może stać przyczyną istotnego zmniejszenia wydajności obliczeń, jeśli dane nie będą wyrównane z uwzględnieniem podanej powyżej specyfiku pracy z cache-liniami.**

To leży na programiście.

Środki, które zapobiegają wyniknięciu cache-konfliktów:

- **wyrównanie rozmieszczenia danych w pamięci głównej tak, żeby różne procesory mieli dostęp do różnych adresów pamięci, rozdzielonej granicą przynajmniej jednej cache-linii.**

- **Rozdzielenie danych tylko dla odczytu od danych dla odczytu i zapisu. Dane dla odczytu nie wymagają wyrównania na granicy cache-linii.**

- **Użycie pamięci lokalnej dla każdego wątku. Wtedy wyrównanie na granicy cache linii odbywa się automatycznie.**

UWAGA! Cache-konflikty powstają dla architektury ze wspólną pamięcią. Dla architektury z pamięcią rozproszoną każdy procesor odczytuje-zapisuje dane, pobierane z pamięci lokalnej – cache konflikty nie występują.