Komputery macierzowe

Krzysztof Ostrowski, Piotr Byszewski
Wydział Informatyki Politechniki Szczecińskiej.
Informatyka – studia magisterskie dzienne, rok 2, grupa 20Ć.
Email: {krostrowski, pbyszewski}@wi.ps.pl

Wstęp

Celem referatu będzie omówienie komputerów macierzowych – komputerów wykonujących synchronicznie jednakowe operacje na różnych zestawach danych. Omówione będą najważniejsze zagadnienia związane z tymi maszynami – organizacja, problem spójności pamięci podręcznej. Na przykładzie pierwszego komputera macierzowego ILLIAC IV zostaną zaprezentowane realizacje organizacyjne i architektoniczne SIMD. Przedstawione będą najważniejsze metody przyspieszające obliczenia zastosowane w superkomputerach Cray. Jako rozwinięcie tematu zostaną omówione topologie sieci połączeń.

Komputer równoległy

Większość obecnie produkowanych komputerów należy, wg klasyfikacji Flynna [FLYN72], do kategorii SISD – tzn. wykonują sekwencyjnie jedną instrukcję na jednym strumieniu danych. Jednak przeglądając specyfikacje najnowszych procesorów można się z tym nie zgodzić. Nowe PC są już zaopatrywane w wiele funkcji, które niegdyś były atrybutem wyłącznie wielkich komputerów działających równolegle. Po co, więc, budować komputery równoległe o wielkiej mocy i równie dużych rozmiarach?

Odpowiedź jest prosta: szybkość i skuteczność działania. Mały biurkowy komputer nie jest w stanie wydajnie przetwarzać większych ilości danych, ma relatywnie niską liczbę operacji zmiennoprzecinkowych na sekundę (FLOPS), czy też operacji całkowitych na sekundę (IPS). Pecetem nie jesteśmy w stanie rozwiązywać skomplikowanych zadań symulacji, przeprowadzić zaawansowanego przetwarzania obrazów (przede wszystkim 3D), czy też przewidywać pogody. Za to możemy cieszyć się wystarczającą do potrzeb nieprofesjonalnych wydajnością za stosunkowo niską cenę.

Konstrukcja coraz szybszych procesorów nie nadąża (również technologicznie – nieprzekraczalne wciąż pozostają prawa fizyki) za rosnącymi potrzebami obliczeń w nauce, technice, ekonomii czy zarządzaniu. Kilka wolniejszych procesorów jest tańszych (także w eksploatacji) niż jeden superszybki. Pomimo, że cena paralelizmu sprzętowego w ostatnich latach spadła, a jego elementy można znaleźć nawet w popularnych PC (Intel: MMX, SSE, SSE2, AMD: 3DNow!, PowerPC: AltiVec, HP PA-RISC: MAX, Sun UltraSPARC: VIS, itp.), to jednak jest jeszcze bardzo daleko do w pełni równoległych komputerów biurkowych.

Komputery równoległe mają długą historię – począwszy od późnych lat 60-tych, kiedy zaczęły pojawiać się pierwsze takie maszyny, poprzez lata 80-te aż po dzień dzisiejszy, gdzie można zauważyć stale wzrastające zainteresowanie tego typu komputerami. Duże problemy ze standaryzacją architektur, a więc także brak przenośności oprogramowania, oraz cena tych maszyn skutecznie hamowały i nadal hamują wzrost ich popularności.

Co to jest komputer macierzowy

Zapotrzebowanie na obliczenia na dużych seriach danych skłoniło projektantów komputerów do opracowania superkomputera, który byłby w stanie realizować operacje równolegle – a więc szybciej. We wczesnej fazie rozwoju komputery o dużej mocy znajdowały zastosowanie głównie w wojsku i centrach badawczych (najlepszym przykładem jest ILLIAC IV wykorzystywany przez NASA do „podboju kosmosu"). Początkowo opierano się na modelu SIMD – wspólna pamięć z połączoną macierzą jednostek przetwarzających, które nadzorowała jednostka sterująca. Jak wynika ze specyfikacji SIMD – taki superkomputer wykonywał jeden rozkaz na wielu danych. Stąd też wzięła się nazwa – komputer macierzowy – od macierzy jednostek przetwarzających macierze danych zmiennoprzecinkowych. Wkrótce znaczenie tego terminu uległo znacznemu poszerzeniu.

W literaturze można się spotkać z nieco mylnymi określeniami. Bardzo często procesorem macierzowym (tablicowym) nazywany jest pomocniczy procesor operujący tylko na danych wektorowych. Taki procesor jest użytkowany jako urządzenie peryferyjne przez duże komputery lub minikomputery do wykonywania wektorowych fragmentów programów. Procesorem wektorowym nazywany jest procesor z ALU umożliwiającym przetwarzanie potokowe, mimo, że równoległa organizacja ALU także jest zdolna do przetwarzania wektorowego, podobnie jak i superkomputery. Gdyby tego jeszcze było mało – w literaturze anglojęzycznej terminy *array*, *vector* są używane zamiennie, a termin *processor* jest często synonimem komputera. Dalej: komputer macierzowy bardzo często jest klasyfikowany jako model tylko i wyłącznie SIMD, co oczywiście jest błędne.

Stąd wydaje się bardziej odpowiednim stosowanie terminu superkomputery wieloprocesorowe (nie mylić z wielokomputerowymi) o współużytkowanej pamięci zamiast komputery macierzowe.

Organizacje wieloprocesorowe

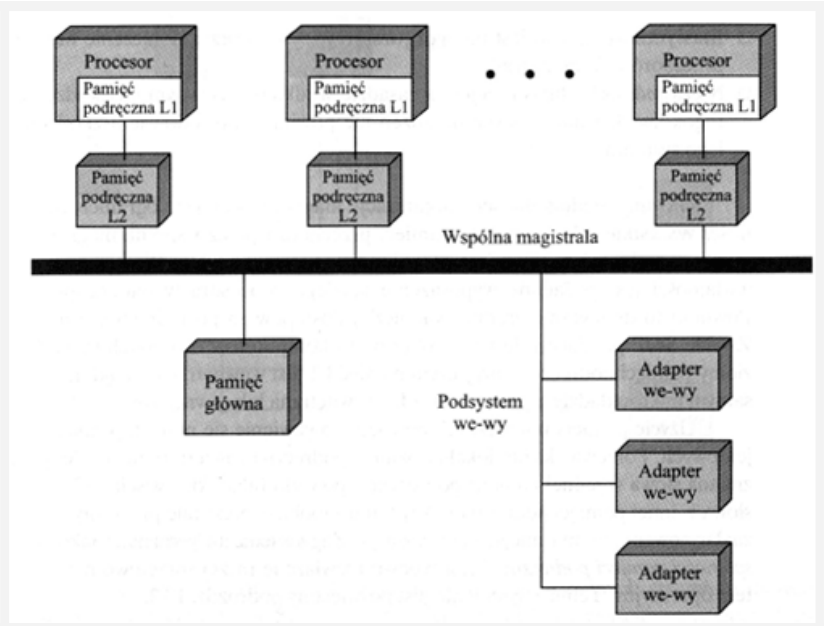
Klasyczna organizacja superkomputerowa cechuje się wykorzystaniem wielu, fizycznie niezależnych, jednostek przetwarzających (PU) połączonych szybką siecią połączeń. Podczas, gdy w procesorze superskalarnym wykorzystuje się możliwość pracy równoległej na poziomie rozkazów, przetwarzanie równoległe korzysta z mechanizmów pozwalających na równoległą pracę współdziałających PU. Tego typu organizacja rodzi wiele problemów. Każdy element przetwarzający posiada własną pamięć podręczną oraz ma dostęp do pamięci wspólnej, toteż musi być zastosowane rozwiązanie sprzętowe (bądź programowe) zapewniające prawidłowe aktualizowanie pamięci, tak aby każdy PU miał jej właściwy (aktualny) obraz. Zagadnienie to znane jest jako problem spójności pamięci podręcznej (*cache coherence*).

Komputery macierzowe (nazywane niekiedy wektorowymi, chyba z racji wykorzystania potokowania) to zazwyczaj maszyny wieloprocesorowe współużytkujące pamięć, zapisane w niej dane i programy, a także komunikujące się za jej pośrednictwem. Najczęściej spotykaną postacią takiego systemu jest wieloprocesor symetryczny (*symmetric multiporcessor*) SMP. Nowszym i obecnie często stosowanym rozwiązaniem jest NUMA (*nonuniform memory access*) – organizacja o niejednorodnym dostępie do pamięci. Tutaj także użytkowana jest wspólna pamięć (jednak nie w sensie fizycznie dosłownym – o czym dalej), jednak czas dostępu do niej dla konkretnego PU zależy od rejonu, którego ten dostęp dotyczy.

Większość obecnie używanych komputerów równoległych to bardzo szybkie klastry o architekturze mieszanej. Klaster jest układem niezależnych jednostek obliczeniowych (węzły) połączonych szybką siecią komunikacyjną. W klastry można łączyć komputery typu SMP, NUMA, komputery biurkowe jak i same klastry. Zwykle każda jednostka posiada niezależną pamięć. Z kolei jednostka taka może zawierać kilka procesorów ze wspólną pamięcią. Zawsze istnieje komputer sterujący, który widzi jedną przestrzeń pamięci, dysków twardych, plików itp. Taki typ architektury równoległej jest bardzo elastyczny. Obecnie najczęściej wykorzystywane są klastry typu Beowulf (szybkie/standardowe procesory oraz szybka/standardowa sieć), gdzie zwykle łączy się wiele PC-tów. Beowulf profesjonalne skłaniają się w stronę bardzo szybkich superklastrów z pamięcią dzieloną. Tanie systemy równoległe o pamięci rozproszonej (*distributed memory*) to znane seti@home połączone siecią Internet. Z racji, że typowy komputer macierzowy nie zalicza się do rozproszonych wielokomputerowych systemów, zagadnienie klastrów nie zostanie tutaj szerzej omówione.

Najstarsze i najczęściej stosowane rozwiązanie wieloprocesorowe – SMP

Organizacja SMP odzwierciedla model MIMD, na który składa się pamięć dzielona (*shared memory*) współużytkowana przez wszystkie synchronicznie działające jednostki przetwarzające oraz jednostkę sterującą działającą pod kontrolą systemu operacyjnego. Na wieloprocesor symetryczny składa się większa niż jeden liczba jednakowych procesorów o porównywalnych możliwościach realizujących jednakowe funkcje (stąd określenie symetryczne). Każdy procesor jest traktowany jako jednostka przetwarzająca (PU), która jest połączona z innym PU za pośrednictwem magistrali lub innego systemu połączeń w taki sposób, że czas dostępu do pamięci dla każdego z nich jest w przybliżeniu jednakowy (UMA – *uniform memory access*). PU może komunikować się z innym PU poprzez określone obszary pamięci lub bezpośrednio sygnałami. Dostęp do urządzeń wejścia/wyjścia jest przydzielany każdej jednostce przetwarzającej. Wyróżnia się trzy główne rozwiązania organizacyjne SMP.



Organizacja SMP – wspólna magistrala. [STAL04]

Wspólna magistrala (z podziałem czasu)

Jest to najprostsze i praktycznie identyczne rozwiązanie jak w systemach jednoprocesorowych. Magistrala składa się z linii adresu, sterowania i danych. Transfery bezpośrednie do pamięci (DMA) ułatwiają mechanizmy adresowania (do wyróżniania modułów na magistrali), arbitrażu (możliwość funkcjonowania modułu jako jednostki nadrzędnej, rozwiązanie problemu rywalizacji o sterowanie magistralą) oraz podziału czasu (w określonej chwili tylko jeden moduł może sterować magistralą, pozostałe są odłączone). Za magistralą z podziałem czasu przemawia elastyczność rozbudowy (wystarczy wpiąć w magistralę kolejny procesor) oraz działanie nawet w przypadku uszkodzenia któregoś z procesorów. Wadą takiego rozwiązania jest ograniczona wydajność, ponieważ wszystkie polecenia przechodzą przez magistralę. W przypadku użycia w procesorach pamięci podręcznych dochodzi jeszcze problem ich spójności.

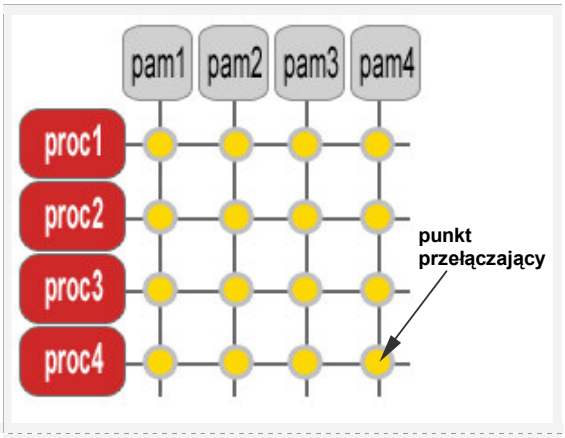
Pamięć wieloportowa

Użycie pamięci wieloportowej umożliwia bezpośredni i niezależny dostęp każdego procesora i każdego urządzenia we/wy do modułów pamięci głównej (własne, niezależne ścieżki). W celu uniknięcia konfliktów stosowane są skomplikowane sieci układów logicznych i system priorytetów. Jest to rozwiązanie wydajniejsze niż wspólna magistrala. Istnieje możliwość zarezerwowania określonych obszarów pamięci wyłącznie dla określonego procesora (bez dostępu dla innych). Zmiana danych w *cache* procesora wymusza natychmiastową zmianę tych danych w pamięci głównej (zapis jednoczesny), ponieważ nie ma możliwości poinformowania pozostałych procesorów o aktualizacji pamięci.

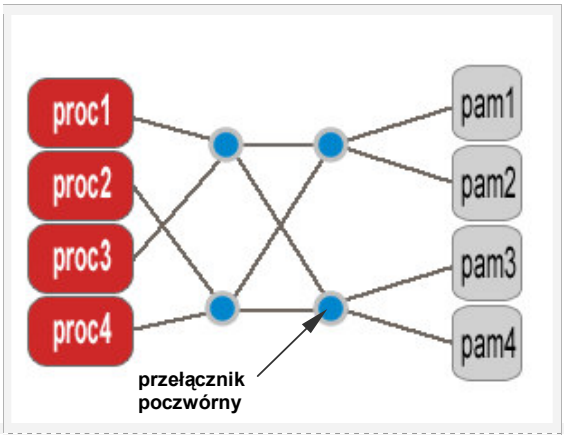
Aby umożliwić każdemu procesorowi dostęp do każdego modułu pamięci stosowane są rozwiązania ogólnie zwane architekturą przełączaną. Dwa najczęstsze rozwiązania to wybierak krzyżakowy (*crossbar switch*) i sieć omega (*omega network*).

Zaletą wybieraka krzyżowego jest fakt, że umożliwia on szybki dostęp wielu procesorów jednocześnie do różnych bloków pamięci. Główną wadą takiego rozwiązania jest wymagana ilość przełączników – dla n procesorów i n bloków pamięci potrzeba aż n^2 tych urządzeń. Szybkość dostępu do pamięci spada, kiedy wiele procesorów odwołuje się do tego samego bloku jednocześnie.

Sieć omega wymaga zastosowania przełączników poczwórnych, a ich ilość dla n procesorów i n bloków pamięci wynosi zaledwie $(n/2)\log_2 n$. Rozwiązanie to nie jest bez wad. Tutaj także spada szybkość dostępu do pamięci, kiedy wiele procesorów odwołuje się jednocześnie do tego samego bloku. Istotny jest czas przełączania – dla większych n mogą wystąpić istotne opóźnienia.



Wybierak krzyżakowy (*crossbar switch*).



Sieć omega (*omega network*).

Centralna jednostka sterująca (CU)
Jest to najstarsze rozwiązanie organizacyjne SMP, obecnie bardzo rzadko stosowane. CU steruje przepływem danych pomiędzy procesorem, pamięcią i we/wy. Może dokonywać arbitrażu, buforowania, realizować taktowanie. Zarządzanie PU odbywa się tylko w jednostce sterującej, co sprawia, że prostota tego rozwiązania jest porównywalna z użyciem wspólnej magistrali. Wadą jest tutaj oczywiście ograniczona wydajność CU. W taki system był wyposażony pierwszy komputer macierzowy ILLIAC IV.

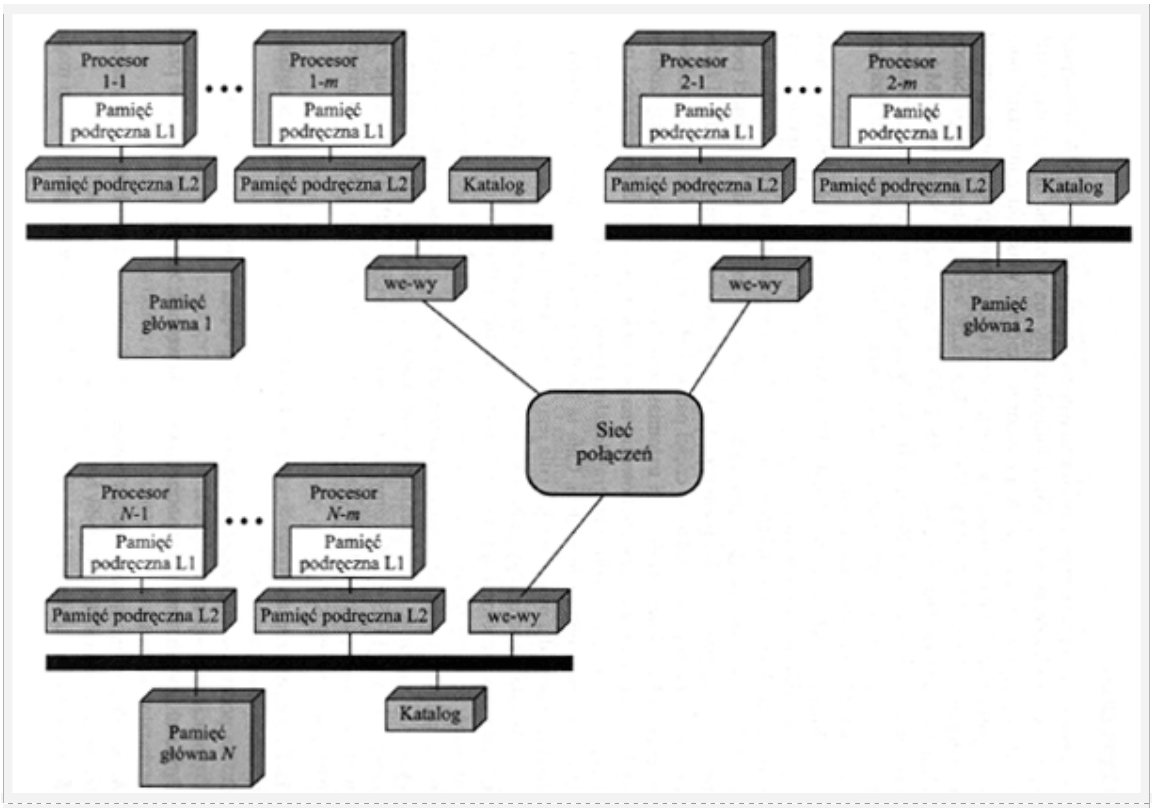
Nowoczesne rozwiązanie wieloprocesorowe – CC-NUMA

NUMA określa sposób dostępu poszczególnych procesorów do pamięci, która jest podzielona na rejony. W zależności od wybranego rejonu pamięci określany jest czas dostępu do niej. CC-NUMA dodatkowo zachowuje spójność pamięci podręcznych. System NUMA można porównywać z klastrem, jednak CC-NUMA jest równie odmienne od klastrów jak i od SMP.

W przypadku SMP wraz ze wzrostem ilości PU wzrastał ruch na magistrali, a więc i spadała wydajność – mimo, że zamierzenie było odwrotne. SMP osiąga najlepszą wydajność przy ilości PU między 16 a 64. Takie ograniczenia sprzyjały powstaniu klastrów. Jednak i one nie są bez wad. Spójność pamięci podręcznych jest realizowana programowo, co znowu źle się odbija na wydajności. Lekarstwem jest system NUMA.

Na system NUMA składa się wiele niezależnych węzłów (zazwyczaj SMP) połączonych siecią (lub innym rozwiązaniem, np. przełącznikami czy pierścieniem). Pamięć wszystkich węzłów tworzy jedną pamięć adresowalną z unikatowymi adresami. Dostęp do pamięci „cudzej” jest dużo wolniejszy niż do pamięci „własnej”. Spójność pamięci podręcznej węzła jest realizowana na zasadzie „katalogu”, który wskazuje położenie różnych części pamięci i zawiera informacje o statusie pamięci podręcznych. Zapisy w katalogach są modyfikowane wraz ze zmianami w pamięci.

Wydajność CC-NUMA jest dużo większa niż SMP jeżeli jest mało (lub nie ma) odwołań do odległych węzłów. Oprogramowanie dla SMP nie będzie działać w CC-NUMA.



Organizacja CC-NUMA. [STAL04]

Utrzymywanie spójności pamięci podręcznych

Spójność pamięci podręcznych można kontrolować programowo i sprzętowo. Realizacja spójności na poziomie kompilatora polega na określeniu, które dane mogą potencjalnie prowadzić do niespójności pamięci podręcznej i uniemożliwieniu kierowania ich do pamięci *cache*. Najwygodniej jest w ogóle nie kierować wspólnych zmiennych do pamięci podręcznej, jednak jest zbyt pochopne. Efektywniej jest przeanalizować program i określić okresy bezpieczne dla wspólnych zmiennych, a w okresach krytycznych wymusić spójność.

Rozwiązania sprzętowe (protokoły spójności pamięci podręcznej). Dynamiczne rozpoznawanie momentów potencjalnej niespójności i rozwiązywanie problemu tylko wtedy, kiedy rzeczywiście zaistnieje jest wydajniejsze niż rozwiązanie programowe. Wykorzystywane są protokoły katalogowe oraz podglądania. Protokół katalogowy opiera się na centralnym sterowniku pamięci (zwykle jest to część sterownika pamięci głównej) oraz katalogu umieszczonego w tej pamięci. Zasada działania została omówiona w rozdziale poświęconym systemowi NUMA. W protokołach podglądania utrzymywaniem spójności pamięci podręcznych zajmują się sterowniki tych pamięci każdego procesora. W przypadku aktualizacji bloku pamięci wspólnej, o fakcie zostają powiadomione pamięci podręczne za pośrednictwem np. magistrali. Każdy sterownik podgląda sieć w poszukiwaniu takich powiadomień i odpowiednio reaguje.

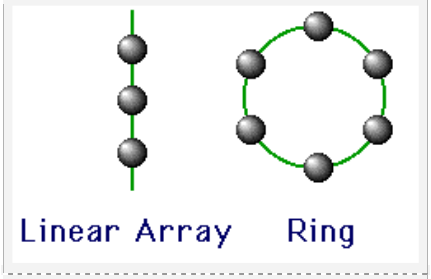
Organizacje SMP stosują protokół MESI do zachowywania spójności. Wykorzystanie tego protokołu nadaje „świadomość" pamięci podręcznej. Każdy wiersz tej pamięci jest zaopatrzony w dwa bity stanu. Mogą to być zmodyfikowany (*modified*) – wiersz pamięci różni się od tego w pamięci głównej i jest dostępny tylko w tej pamięci *cache*, wyłączony

(*exclusive)* – wiersz w pamięci *cache* jest taki sam jak w głównej i nie występuje w żadnej innej pamięci podręcznej, wspólny (*shared*) – wiersz jest pamięci głównej i *cache* oraz może występować w *cache* innych procesorów, nieważny (*invalid*) – taki wiersz w *cache* jest traktowany jako pusty. Wraz ze zmianami w pamięci modyfikowane są bity stanu.

W przypadku większej ilości poziomów pamięci podręcznych wykorzystuje się rozszerzony protokół MESI. Rozszerzenie polega m.in. na wykorzystaniu jednoczesnego zapisu z jednej pamięci *cache* do innej.

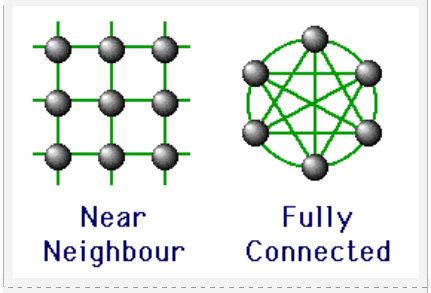
Topologie sieci połączeń

Sposób łączenia układu procesorów w sieć to topologia sieci (przedstawiana w postaci nieskierowanych grafów). Wiele nowoczesnych komputerów MIMD wyposażanych jest w bardzo szybkie linie komunikacyjne między procesorami, jednak zwiększanie liczby procesorów znacznie obniża ich wydajność. Pewnym rozwiązaniem wpływającym na polepszenie szybkości komunikacji jest zastosowanie odpowiedniej topologii sieci. Znajomości zastosowanej topologii umożliwia wydajne sterowanie procesami (np. redukcja opóźnienia dzięki operowaniu na blisko siebie położonych procesorach).



Najprostsze topologie sieci połączeń
http://www.cems.uwe.ac.uk/

Najprostszymi sposobami łączenia procesorów jest szereg (*linear array*) oraz pierścień (*ring*). Takie topologie są nieefektywne – raz ze względu na rosnące wraz ze wzrostem ilości procesorów opóźnienie, a dwa za brak niezawodności. Uszkodzenie któregoś z procesorów w szeregu spowoduje zatrzymanie pracy całego układu, o ile nie zastosowało się alternatywnych połączeń przygotowanych specjalnie na obsługę tego rodzaju zdarzeń.



Kratowy schemat sieci połączeń.
http://www.cems.uwe.ac.uk/

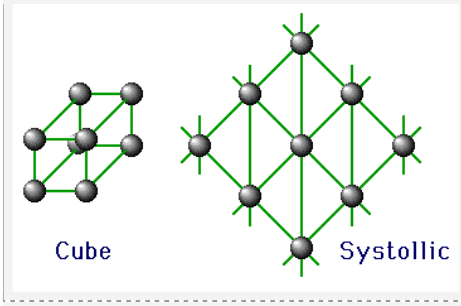
Istnieje wiele hybrydowych schematów połączeń, które wywodzą się wprost od najprostszych topologii. Przykładem może być drzewo (znane z algorytmiki) czy też gwiazda. Ciągle jednak uszkodzenie (w tym przypadku głównego) jednego procesora wyłącza układ. Rozwiązaniem tego problemu jest zastosowanie kratowych schematów połączeń.

Idea kratowego schematu połączeń (*the mesh*) opiera się na stałej ilości połączeń między każdymi dwoma procesorami. Dąży się do połączenia każdego procesora z każdym, jednak jest to bardzo kosztowne. Liczba połączeń rośnie wraz z liczbą procesorów, a więc rośnie także opóźnienie w komunikacji. Rozsądne rozmieszczanie procesów może utrzymać współpracujące procesy w jednej lokacji, a więc zniwelować opóźnienie w komunikacji.

Jedną z wariacji sieci kratowej jest torus – krata z zawiniętymi narożnymi połączeniami. Umożliwia to skierowanie zawiniętych „rogów” do następnego poziomu połączeń (o ile

istnieje), tworzy się w ten sposób macierz procesorów wewnątrz kraty (przykładowo: takie były założenia konstrukcji komputera ILLIAC IV).

Najtrudniej jest zaprojektować sieć połączeń, która będzie niezawodna („odporna" na uszkodzenia procesorów) i szybka w działaniu. Naiwnym i kosztownym rozwiązaniem będzie połączenie każdego procesora z każdym. Kompromisem między układem kratowym (*near neighbour*) a układem pełnych połączeń (*fully connected*) jest przykładowo hiperkostka n-wymiarowa (*n-dimensional hypercube*). Jest to bardzo wydajne rozwiązanie (małe opóźnienia), ale trudne w projektowaniu w przypadku większych wymiarów.

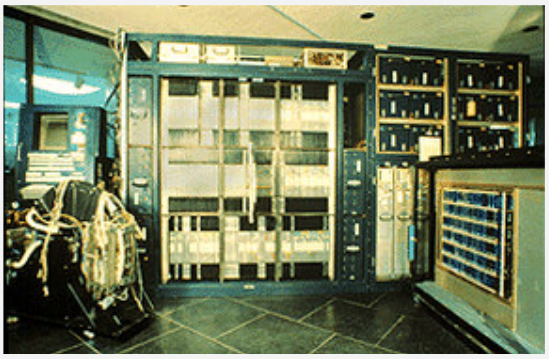


Kompromis dla sieci pełnych połączeń.
http://www.cems.uwe.ac.uk/

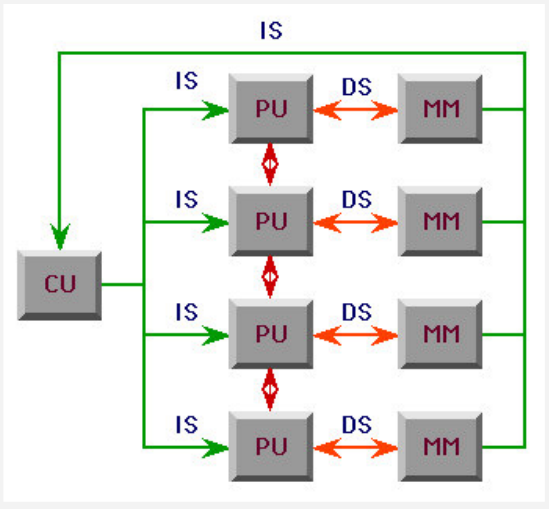
Ciekawym przykładem łączenia topologii jest The MIT Connection Machine (realizacje: CM-1, CM-2). Tutaj 1-bitowe procesory połączone są w kratę (256x256). W celu przyspieszenia komunikacji wydzielone grupki 16-sto procesorowe łączy się dodatkowo w 12-wymiarową kostkę, gdzie, z kolei każdy procesor połączony jest z dwoma innymi w szereg. Procesory miały bardzo ograniczone możliwości (złożone instrukcje rozkładał na mniejsze specjalny mikrokontroler), ale ich liczba jest spora (65535 procesorów, w przypadku CM-2 dochodzą jeszcze koprocesory). Dzięki takiemu rozwiązaniu i zastosowaniu skomplikowanej topologii osiągnięto szczytową wydajność 20GFLOPS-ów. Tego typu maszyny budowano głównie na potrzeby sztucznej inteligencji.

ILLIAC IV

Pierwszym komputerem macierzowym kategorii SIMD był zaprojektowany dla NASA przez University of Illinois ILLIAC IV (1969, (1972)). Procesor tej maszyny operował na tablicy danych wejściowych (wykonanie rozkazu dla maksymalnie 64 elementów jednocześnie).



ILLIAC IV (1969, NASA).
http://www.cems.uwe.ac.uk/



Uproszczona architektura ILLIAC IV – SIMD. CU – jednostka sterująca, PU – jednostka przetwarzająca, MM – pamięć, IS – strumień rozkazów, DS – strumień danych.
http://www.cems.uwe.ac.uk/

ILLIAC mógł być programowany językiem wysokiego poziomu bazującym na Fortranie – CFD (Computational Fluid Dynamics). Przygotowanie programów użytkowych odbywało się na osobnym procesorze (host firmy Burroughs) z wykorzystaniem zaawansowanego systemu operacyjnego – w porównaniu z prostym systemem kontrolującym CU, umożliwiającym jedynie realizację ładowanych programów.

Jednostka sterująca (CU) rozsyłała instrukcję do każdej aktywnej jednostki procesorowej. Inicjalizuje wykonanie rozkazu przez procesory oraz kontroluje zapis wyników do pamięci (ma również dostęp do pamięci dowolnego PU). Rozkazy sterujące (np. skoki) oraz rozkazy przetwarzające skalary wykonywał sam CU. Dopiero rozkazy wektorowe (po zdekodowaniu) są transmitowane do macierzy PU. Cykl wykonywania rozpoczynał się w momencie, kiedy każdy procesor posiadał ten sam rozkaz, a dane zostały załadowane do pamięci. Cykl ten mógł trwać do 400ns.

Każdy procesor (PU) był w stanie kontrolować dane pobrane ze swojej własnej pamięci lokalnej (ponieważ z każdym PU jest związany jeden moduł pamięci – jednak nie jest to zasadą), pobierać dane poprzez sieć połączeń lub pośrednio poprzez CU mieć dostęp do pamięci innego PU lub do pamięci globalnej. Kratowa (*mesh*) struktura sieci połączeń mogła być użyta do przesłania pośrednich wyników operacji do innych procesorów w celu wykonania następnej instrukcji. Liczba zaangażowanych PU do wykonania danej instrukcji zależy od rozmiaru wektora danych (pozostałe, nieużywane, PU są blokowane przez CU).

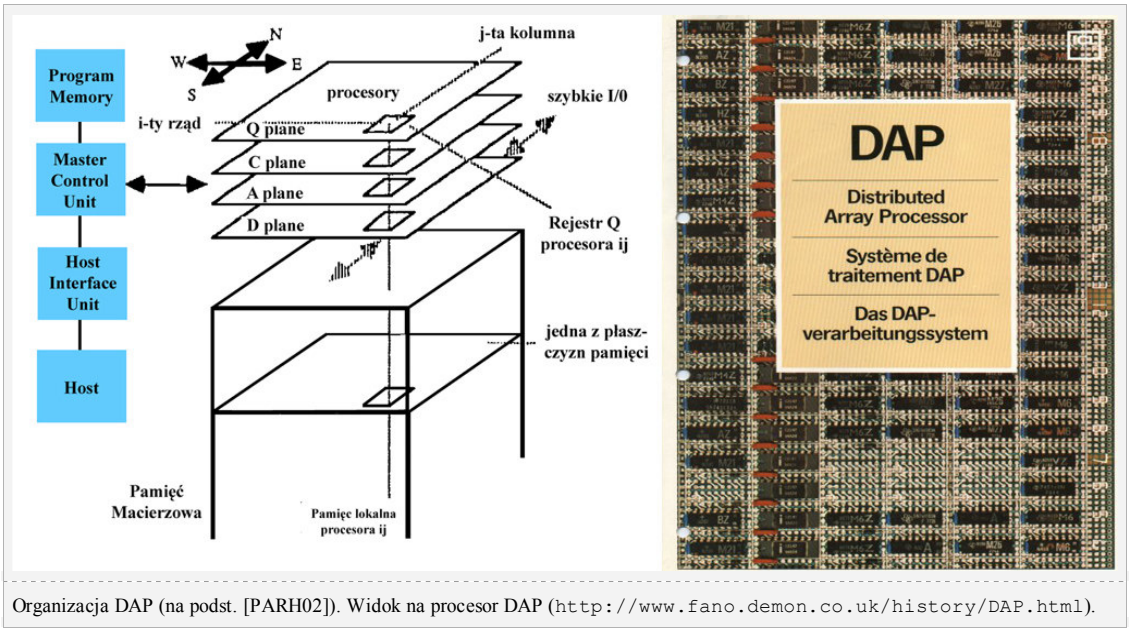
ILLIAC IV osiągał teoretyczną maksymalną wydajność 500 MFLOPS-ów (chociaż podczas pracy ciągłej było to dużo mniej – od 40 do 100 MFLOPS-ów) i 200 MIPS-ów. Założeniem projektantów było osiągnięcie 1GFLOPS-ów i zaangażowanie do pracy 256 PU, można więc uznać projekt ILLIAC IV za „nieudany". Każdy PU taktowany był 13Mhz zegarem, miał 2048 słów pamięci do zapamiętywania wyników oraz 6 rejestrów ogólnego przeznaczenia. ILLIAC to synchroniczna (komputery macierzowe zawsze są synchroniczne) maszyna 64-

bitowa. Na pamięć główną składało się 13 zsynchronizowanych dysków magnetycznych, dając łącznie 1GB przestrzeni dyskowej.

Każdy PU miał ograniczone możliwości, jednak liczba zaangażowanych procesorów była na tyle duża, że komputer macierzowy ILLIAC był bardzo drogi. ILLIAC IV do połowy lat ’80 XX wieku był najszybszym komputerem macierzowym kategorii SIMD (pomimo tego, że był „nieudany”).

DAP

Pierwszym masowo produkowanym i szeroko dostępnym procesorem macierzowym był DAP (*Distributed Array Procesor*). Opracowany w 1976 roku przez International Computers Limited w Anglii występował pod nazwą ICL DAP. Nieco później powstała amerykańsko-angielska firma Cambridge Parallel Processing, która za cel postawiła sobie zbudowanie i rozreklamowanie systemów opartych na DAP. Ostatnie produkty z tej serii zawierały 1024 lub 4096 procesorów 1-bitowych skonfigurowanych w 32x32 lub 64x64 kraty, każdy procesor miał odpowiednio przydzielone do 16 lub 64MB bloku pamięci. Mniejszy model mieścił się pod biurkiem. DAP zazwyczaj hostowała stacja robocza firmy Sun. Droższe modele procesorów DAP były wyposażone w 8-bitowe koprocesory – jeden na każdy element przetwarzający. Służyły do przyspieszania operacji na liczbach całkowitych i zmiennoprzecinkowych.



Organizacja DAP (na podst. [PARH02]). Widok na procesor DAP (`http://www.fano.demon.co.uk/history/DAP.html`).

Kod i dane były przechowywane osobno. Strukturę kontrolującą stanowiła jednostka MCU (*Master Control Unit*). Jej głównym zadaniem było rozsyłanie instrukcji do procesorów. MCU pełniła również rolę procesora skalarnego. Aplikacje składały się z części działającej na hoście, oraz skompilowanej osobno, działającej na DAP.

Organizacja DAP była podobna do tej, znanej z ILLIAC IV. DAP składał się z procesorów typu *bit-serial* (przetwarzane jest po jednym bicie każdego słowa równocześnie). Każdy procesor był połączony ze swoimi czterema sąsiadami. W porównaniu z ILLIAC IV blok

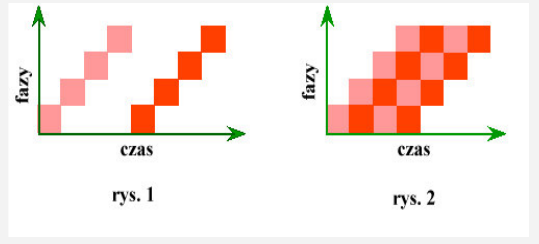
pamięci przydzielony każdemu procesorowi był większy. Jasne jest, że MCU była w stanie adresować wszystkie bloki pamięci.

Każdy z procesorów jest wyposażony w pięć jednobitowych rejestrów (kontroli aktywności – A, akumulacji – Q, I/O danych – D, buforowania danych – S, flaga przesunięcia – C). Jedynie rejestr S nie był widoczny dla programisty, pozostałe były widoczne w postaci płaszczyzn (*planes*). Dane załadowane do płaszczyzny D mogły zostać asynchronicznie przesłane na zewnątrz bez przerywania przetwarzania.

Możliwość pracy na arytmetyce zmiennej długości (*variable-length*), czyniła DAP idealnym do pracy z aplikacjami zarówno numerycznymi jak i nienumerycznymi, zawierającymi duże ilości danych. Głównie były to aplikacje przetwarzające obraz oraz aplikacje bazodanowe. Komputer host pracował zazwyczaj pod Uniksem, z kolei wewnętrzny system operacyjny DAP (sprzęgający jednostkę sterującą i procesor macierzowy) nie był widoczny dla użytkownika. Standardowym dla tej maszyny językiem wysokiego poziomu był Fortran Plus.

Do roku 1992 stworzone zostały 4 generacje systemów DAP, ostatnią były DAP 510 (32x32) i 610 (64x64). Maksymalną wydajność osiągał DAP 610c (model 610 z koprocesorem 8-mio bitowym), było to 560 MFLOPS-ów.

Cray-1



1 – przetwarzanie wieloetapowe (bez potokowania), 2 - przetwarzanie potokowe (*pipelining*).
Na podstawie: `http://www.cems.uwe.ac.uk/`

W procesorze sekwencyjnym pojedyncza operacja $x := a + b$, jest przetwarzana wieloetapowo. Procesor pobiera a i b, porównuje ich wykładniki, odpowiednio przesuwa mantysę, dodaje liczby, normalizuje wynik i zapisuje w x. Jeżeli taka operacja miałaby zostać wykonana na elementach tablicy, wówczas n-ty element tablicy nie byłby przetwarzany dopóki nie zostałyby wykonane operacje na wszystkich wcześniejszych elementach. Zamiast czekać na ukończenie bieżących obliczeń można czekać na zakończenie ich pierwszej fazy, po czym zacząć wykonywać operacje na następnym elemencie tablicy. Takie podejście do obliczeń zwane jest ogólnie potokowaniem (*pipelining*).

Wkrótce po pojawieniu się na rynku komputera ILLIAC IV światło dzienne ujrzał Cray-1, superkomputer (taka etykieta rzeczywiście widniała na jego obudowie) projektu Seymoura Cray'a (1976). Komputer ten prezentował alternatywne do zastosowanego w ILLIAC IV podejście do obliczeń równoległych. Po dzień dzisiejszy Cray-1 jest najbardziej znanym komputerem, w którym wykorzystano potokowanie na szeroką skalę.

Cray-1 nadzorowała, podobnie jak w przypadku ILLIAC IV, jednostka sterująca MCU inicjująca i kontrolująca pracę całego systemu (jednostki przetwarzające, I/O, pamięć zewnętrzna). Pośrednikiem między systemem Cray-1 a człowiekiem był komputer komunikacyjny (*front end*, *host*). Stosowanym językiem wysokiego poziomu był CFT (*Cray-1 Fortran Translator*), który świetnie sprawdzał się w operowaniu na dużych porcjach danych. Schemat blokowy Cray -1 można podzielić na cztery główne części (sekcja wektorowa, skalarna, adresowa, rozkazów). Sekcja wektorowa wykonuje obliczenia na

wektorach danych w sposób potokowy wykorzystując do tego potokowe jednostki funkcjonalne (ALU potokowe), zestaw ośmiu rejestrów wektorowych oraz rejestry specjalne. Sekcja skalarna składa się z rejestrów skalarnych, bufora między rejestrami skalarnymi a pamięcią oraz również z potokowych ALU.

Każda potokowa jednostka funkcjonalna ma przydzielone jedno działanie (np. dodawanie) i może pracować niezależnie (o ile nie korzysta ze wspólnych danych). Jednostki skalarne wykonują operacje całkowitoliczbowe i logiczne na rejestrach skalarnych, a wektorowe na elementach wektorów. Z kolei jednostki zmiennoprzecinkowe działają na elementach wektorów oraz skalarach. Cray -1 to maszyna 64-bitowa, która wykonuje operacje arytmetyczne wyłącznie w obrębie rejestrów (architektura rejestr-rejestr).

Widok na superkomputer Cray-1.
http://en.wikipedia.org/

Architektura rejestr-rejestr znacznie przyspiesza szybkość obliczeń (dostęp do pamięci jest tutaj trzykrotnie wolniejszy niż praca potokowych ALU). Rejestry wektorowe mogą pomieścić po 64 elementy – dłuższe wektory dzielone są sprzętowo na segmenty po 64 elementów, które są w dalszej kolejności przetwarzane (spada jednak wydajność).

Cray-1 umożliwiał łańcuchowanie potoków (*pipeline chaining*). Ta nowatorska technika polegała na wykorzystaniu wyników produkowanych przez jedne potokowe ALU bezpośrednio jako danych wejściowych dla innego potokowego ALU. Dodatkowo pierwszym lub ostatnim elementem łańcucha mógł być potok realizujący transmisję z/do pamięci co jeszcze bardziej przyspieszało obliczenia. Aby optymalnie wykorzystać łańcuchowanie stosowano specjalne kompilatory, które wykrywały miejsca potencjalnego zastosowania łańcuchowania i reorganizowały kod.

Cray-1 może być uważany za multipleksowane w czasie implementacje SIMD, co sprawia że klasyfikuje się jako hybryda SIMD i MIMD. Cray-1 osiągał od 27 do 160MFLOPS-ów w zależności od kombinacji operacji oraz zapotrzebowania na dostęp do pamięci.

I co dalej?

W 1952 roku von Neumann wykazał, że dwuwymiarowa macierz 29 stanowych procesorów może symulować zachowanie maszyny Turinga. Od tamtego czasu do roku 1992 wybudowano ponad dwieście maszyn przetwarzających dane równolegle. Wiele z tych maszyn okazało się finansową porażką, przykładem może tu być projekt ILLIAC, który znacznie przekroczył planowany budżet dając przy tym moc obliczeniową cztery razy mniejszą od zakładanej. Jednak praktycznie każdy z tych komputerów miał swój wkład w rozwój projektowania komputerów przetwarzających równolegle, a także w postęp w tworzeniu oprogramowania i algorytmów. Oczywistym jest, że mocy obliczeniowej nie można w nieskończoność zwiększać przez zwiększanie taktowania układu i miniaturyzację, przyszłość stanowią komputery przetwarzające równolegle. Badania nad tymi maszynami są obecnie prowadzone w olbrzymiej liczbie uniwersytetów, jednostek badawczych oraz firm komercyjnych.

Obecnie bardzo rzadko można spotkać superkomputer oparty o tradycyjny model SIMD. Po wprowadzeniu Cray-1 z szeroko rozumianym potokowaniem kierunek rozwoju superkomputerów wskazuje nieuchronnie na MIMD. Sprzyja temu na pewno mniejszy (albo zerowy – przykładowo rozproszone obliczenia przez Internet) koszt wyprodukowania takiej maszyny. Z drugiej strony maszyn typowo SIMD nie można zastąpić żadnymi innymi w choćby realizacji systemów czasu rzeczywistego.

Dla przykładu najnowszy Cray XD1, zbudowany z 12 procesorów 64-bitowych AMD Opteron (w zależności od opcji mogą być dwurdzeniowe) zgrupowanych w węzły o organizacji SMP, osiąga 106GFLOPS-ów. Komputer działa pod systemem Linux. Co ciekawe, jest w stanie wykonywać aplikacje 32-bitowe lub 64-bitowe zgodne z x86. Dodatkowo w celu przyspieszenia obliczeń jest do dyspozycji 6 koprocesorów opartych o Xilinx Virtex-4. Administracja Cray XD1 jest bardzo przyjemna – mamy do dyspozycji bajecznie kolorowy menedżer oraz standardowo linię poleceń. Maszyna obsługuje oprócz języków Fortran także C/C++ oraz Javę. Jedyne, co odstrasza od kupna Cray XD1 to pobór mocy (2200 W) i oczywiście cena. [CRAY05]

Maj 2006

Polecane strony

The History of the Development of Parallel Computing
`http://ei.cs.vt.edu/~history/Parallel.html`
Aktualna lista 500 największych superkomputerów na świecie
`http://top500.org/`
Computer History Museum
`http://www.computerhistory.org/`

Bibliografia

CRAY05 Cray XD1 Datasheet, Cray Inc. 2005, `http://www.cray.com/`
FLYN72 Flynn M. J.: „Some computer organizations and their effectiveness". *IEEE Transactions on Computers*, C-21 (9), str. 114-118, Wrzesień 1972
PARH02 Parhami B.: „Plenum Series in Computer Science. Introduction to Parallel Processing. Algorithms and Architectures". Kluwer Academic Publishers 2002, str. 445-448, 484-485, 488-490, 501-515
STAL04 Stallings W.: „Organizacja i architektura systemu komputerowego". Wydawnictwa Naukowo-Techniczne, str. 710-758, Warszawa 2004