Dariusz CHAŁADYNIAK[1]

WYBRANE ZAGADNIENIA PRZETWARZANIA RÓWNOLEGŁEGO I ROZPROSZONEGO ORAZ KLASTRÓW KOMPUTEROWYCH

Streszczenie

W artykule przedstawiono wprowadzenie do zagadnień przetwarzania równoległego. Wyjaśniono jego ideę oraz zaprezentowano najpopularniejsze klasyfikacje. Przybliżono pojęcie przetwarzania rozproszonego we współczesnych sieciach teleinformatycznych. Przedstawiono wybrane implementacje klastrów komputerowych.

Abstract

The article presents the introduction to parallel computing. It explains the idea and presents the most popular classifications. It also explains the concept of distributed processing in today's telecommunications networks. It presents selected cluster computing implementations.

WSTĘP

Przetwarzanie ogromnych ilości danych wymaga zastosowania wydajnych systemów obliczeniowych oraz szybkich, niezawodnych i rozproszonych sieci teleinformatycznych.

Zwiększanie wydajności współczesnych komputerów poprzez podwyższanie częstotliwości ich zegarów wewnętrznych oraz stosowanie szybszych technologii półprzewodnikowych ma swoje wady i w zasadzie dobiega kresu.

Właściwym kierunkiem rozwoju wydaje się być budowa komputerów o strukturach równoległych, w tym systemów składających się z autonomicznych kompletnych komputerów, wymieniających dane za pośrednictwem klasycznych sieci komputerowych.

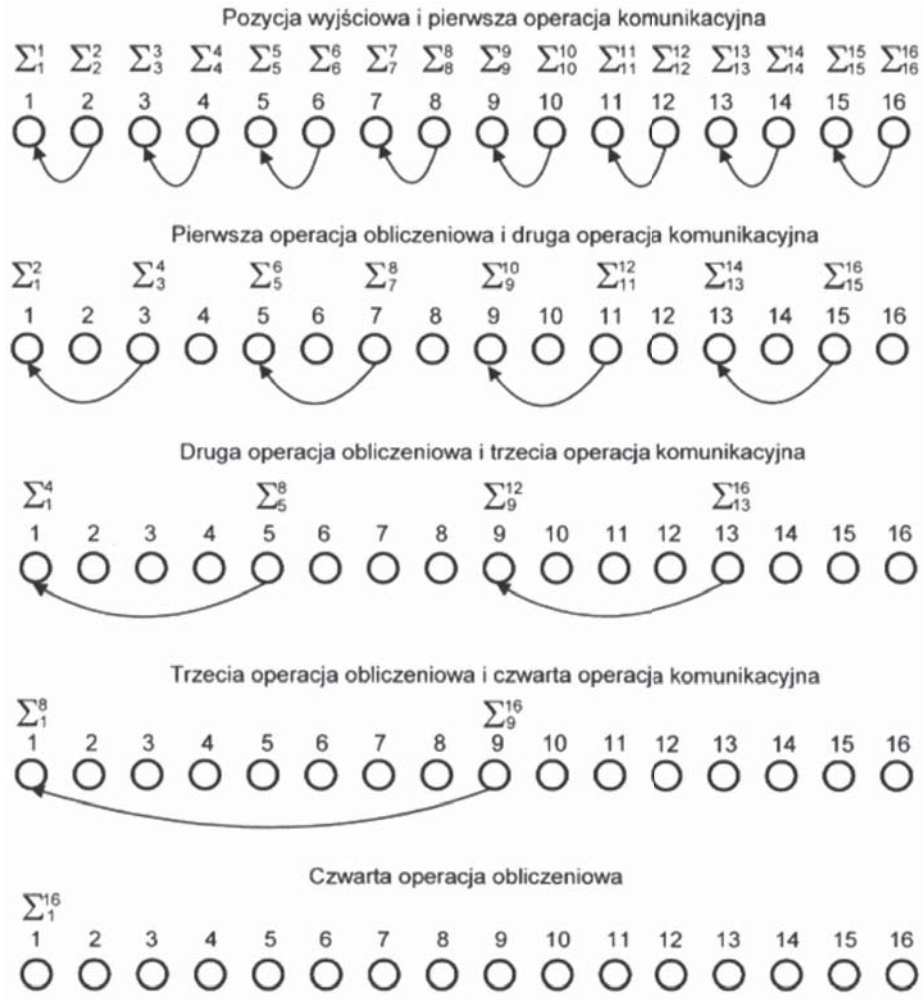
Filozofia przetwarzania równoległego polega na podziale programu na fragmenty, z których każdy wykonywany jest przez inny procesor (lub węzeł). Kod podzielony na „kawałki" wykonywane przez *n* procesorów (węzłów), będzie się wykonywał *n*-razy szybciej niż analogiczny kod, realizowany na maszynie jednoprocesorowej.

[1] Dr inż. Dariusz Chaładyniak jest wykładowcą w Warszawskiej Wyższej Szkole Informatyki.

Idea obliczeń równoległych narodziła się pod koniec lat sześćdziesiątych, ale dopiero w latach osiemdziesiątych w wyniku gwałtownego rozwoju technik komputerowych, była możliwa praktyczna realizacja tego sposobu przetwarzania danych.

1. IDEA PRZETWARZANIA RÓWNOLEGŁEGO

Załóżmy, że w systemie obliczeniowym, złożonym z N procesorów, wykonywana jest operacja sumowania n liczb. Niech $n = N = 16$, a argumenty zostaną rozłożone pomiędzy procesorami w taki sposób, że każdy z nich powiązany zostanie z niezależnym procesorem. Wtedy sekwencja obliczenia sumy ma postać jak na rysunku 1.1.



Rys. 1.1. Operacja równoległego sumowania liczb [6]

Rozpatrywana operacja wykonywana jest w czterech krokach, z których każdy włącza w siebie operację przesłania danych (operacja komunikacyjna) i operację sumowania pary liczb (operacja arytmetyczna). Tak więc sumowanie 32 liczb zakończone będzie po pięciu krokach, a dla sumowania *n* liczb konieczne jest wykonanie *log n* kroków.

W jednoprocesorowym, sekwencyjnym systemie obliczeniowym rozpatrywana operacja sumowania zostanie wykonana w *n -1* krokach; w każdym z nich ma miejsce tylko jedna operacja obliczeniowa (sumowanie). Niech czas wykonania operacji komunikacyjnej t_c będzie równy czasowi wykonania operacji obliczeniowej t_b ($t_c = t_b = t$), wtedy sumaryczny czas przetwarzania zadania T_s w sekwencyjnym systemie obliczeniowym będzie równy:

$$T_s = (n-1)t \tag{1.1}$$

a czas T_p jego wykonania w równoległym systemie obliczeniowym będzie wynosić odpowiednio:

$$T_p = 2t\log_2 n \tag{1.2}$$

Załóżmy, że przyspieszenie S^+, będące rezultatem metod przetwarzania równoległego, będzie stosunkiem czasu niezbędnego do realizacji operacji w systemie sekwencyjnym do czasu wykonania jej w systemie równoległym. Wtedy S^+ można określić za pomocą wyrażenia [6]:

$$S^+ = \frac{(n-1)}{2t\log_2 n} = \frac{(n-1)}{2\log_2 N} \approx \frac{n}{2\log_2 N} \tag{1.3}$$

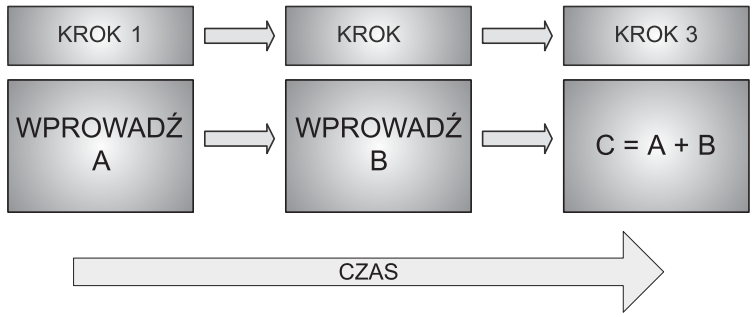
2. KLASYFIKACJA SYSTEMÓW RÓWNOLEGŁYCH

2.1. Klasyfikacja Flynna

W klasyfikacji Michaela Flynna podstawą do rozróżniania poszczególnych rodzajów maszyn obliczeniowych jest zdolność obsługi strumienia danych i strumienia rozkazów. Maszyny równoległe zostały w niej podzielone na 4 grupy.

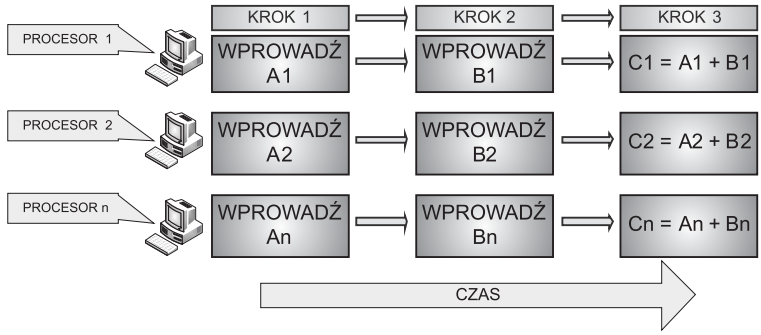
1. **SISD** (ang. *Single Instruction stream, Single Data stream*) – pojedynczy strumień rozkazów (instrukcji) jest wykonywany sekwencyjnie w pojedynczym

strumieniu danych. Jest to architektura komputera sekwencyjnego jednoprocesorowego, zwana też architekturą Neumanna. W tym modelu pojedyncza jednostka przetwarzająca (ALU) wykonuje pojedyncze instrukcje.



Rys. 2.1. Architektura SISD

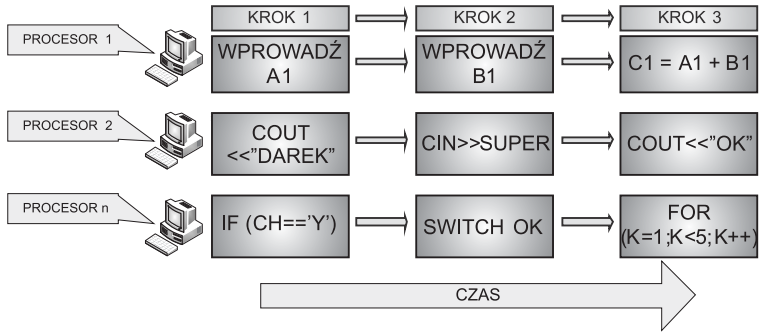
2. **SIMD** (ang. *Single Instruction stream, Multiple Data stream*) – metoda zastosowana we wczesnych systemach równoległych. W tym modelu każdy z procesorów przetwarza ten sam zestaw instrukcji na własnym zestawie danych. Jest to zwielokrotnienie jednostek przetwarzających. Każda jednostka realizuje ten sam, pojedynczy strumień rozkazów, dekodowany przez jedną, wspólną jednostkę sterującą. Wszystkie jednostki przetwarzające pracują w sposób synchroniczny: w danym momencie każda z nich wykonuje ten sam rozkaz, ale na innych danych, dostarczanych jej oddzielnym strumieniem ze wspólnej pamięci. Model SIMD nadaje się do rozwiązywania problemów rachunku macierzowego, algebry liniowej, sortowania, przeszukiwania oraz teorii grafów.



Rys. 2.2. Architektura SIMD

3. **MIMD** (ang. *Multiple Instruction stream, Multiple Data stream*) – każdy z procesorów pracuje z własnym zestawem instrukcji, operując na własnym

zestawie danych. Praca systemu opartego na architekturze MIMD ma charakter asynchroniczny: w danym momencie jednostki przetwarzające mogą wykonywać różne rozkazy na różnych zestawach danych. Procesory pracują całkowicie oddzielnie, realizując oddzielne i najczęściej różne strumienie rozkazów, nazywane procesami. W modelu MIMD mogą istnieć zależności między wykonującymi się równolegle procesami, co wymaga komunikacji między nimi i synchronizacji. Jest to jednak synchronizacja procesów a nie strumieni ze wspólnym zegarem jak w SIMD. Jest to komunikacja między procesami przez wspólną pamięć albo sieć połączeń.



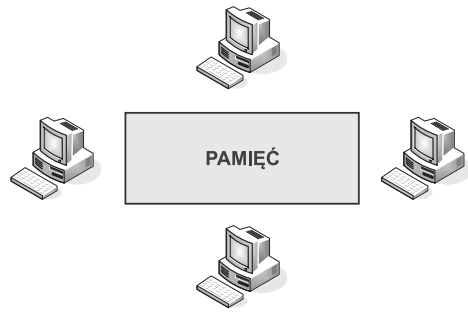
Rys. 2.3. Architektura MIMD

W ramach grupy MIMD wyróżnia się następujące kategorie: MIMD z pamięcią wspólną, MIMD z pamięcią rozproszoną, MIMD z pamięcią rozproszono-wspólną.

3.1. MIMD-SM – architektura z pamięcią wspólną (ang. *Shared Memory*) – w modelu tym procesory są połączone specjalizowaną siecią interconnect, poprzez którą komunikują się ze wspólnym obszarem pamięci. W ramach systemów SM można wyróżnić dwie kolejne kategorie, różniące się podejściem do problemu współdzielenia zasobów:

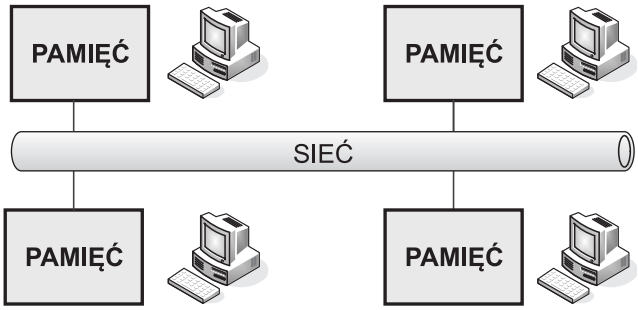
- **Shared everything** – Komunikacja między procesorami odbywa się poprzez operacje zapisu i odczytu pamięci dzielonej, przy czym wymagane jest, aby wszystkie dane programu były przechowywane we wspólnym obszarze pamięci. Wadą tego rozwiązania jest fakt, że wszelkie operacje, wykonywane przez jeden z procesorów, mają wpływ na pracę pozostałych, zaletą jest zaś łatwość przenoszenia oprogramowania.
- **Shared something** – ten model również zakłada komunikację międzyprocesorową, realizowaną poprzez dostęp do obszarów pamięci dzielonej,

przy czym wymaga od użytkownika zdefiniowania, które dane powinny się w tym obszarze znaleźć.



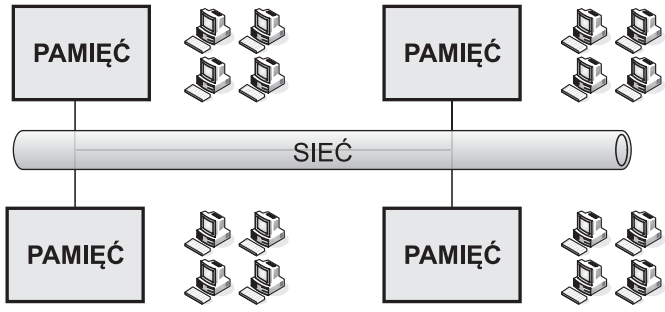
Rys. 2.4. Architektura MIMD-SM

3.2. MIMD-DM – w tym modelu każdy procesor posiada własną pamięć, dostępną tylko dla niego samego. Procesory przekazują sobie nawzajem informacje poprzez komunikaty. Są to zazwyczaj maszyny wieloprocesorowe.



Rys. 2.5. Architektura MIMD-DM

3.3. MIMD-HDSM – rozwiązanie hybrydowe.

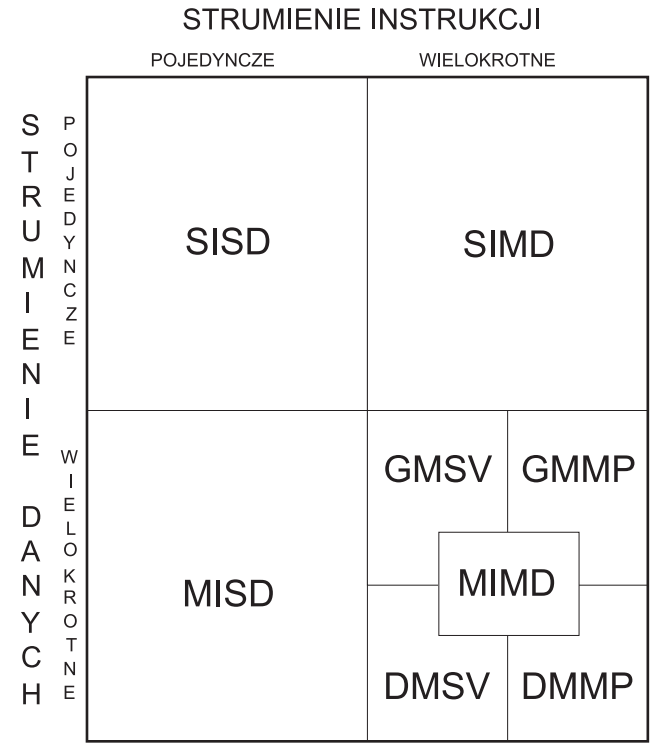


Rys. 2.6. Architektura MIMD-HDSM

4. **MISD** (ang. *Multiple Instruction stream, Single Data stream*) – maszyny tego typu wykonują różne operacje na tych samych danych. Zbudowano niewiele takich maszyn, nie są one używane do celów komercyjnych. Znajdują zastosowanie dla uzyskania bardzo wysokiej niezawodności, np. sterowanie reaktorami jądrowymi.

2.2. Klasyfikacja Johnsona

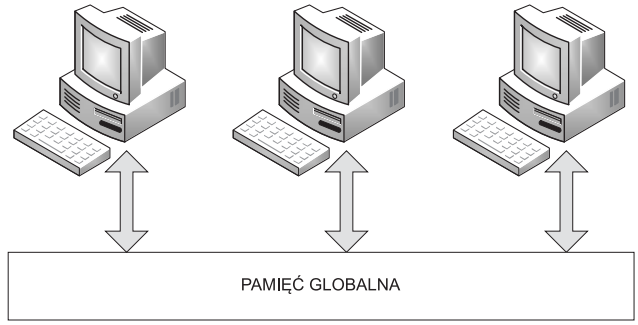
Ponieważ większość systemów przeznaczonych do obliczeń równoległych jest klasyfikowana jako maszyny MIMD, Johnson usystematyzował tę grupę, wprowadzając jako kryterium podziału strukturę pamięci.



Rys. 2.7. Rozszerzona klasyfikacja Johnsona [7]

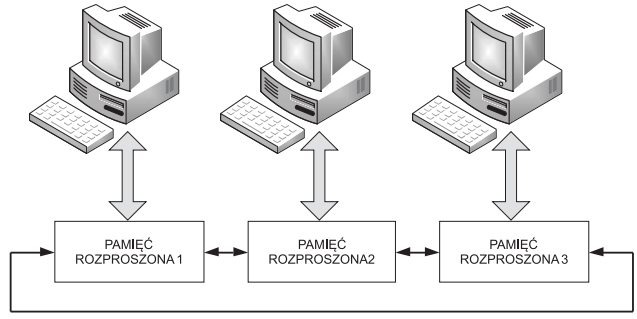
Według Johnsona w ramach architektury MIMD możemy wyróżnić 4 grupy:

1. **GMSV** (ang. *Global Memory Shared Variables*) systemy z pamięcią globalną i współdzielonymi zmiennymi. Maszyny tego typu stanowią ściśle powiązane wieloprocesory.



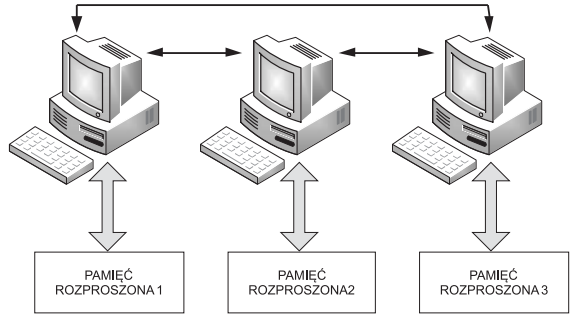
Rys. 2.8. Architektura GMSV [7]

2. **GMMP** (ang. *Global Memory Message Passing*) systemy z pamięcią globalną i przekazywaniem wiadomości. Brak praktycznych rozwiązań.
3. **DMSV** (ang. *Distributed Memory Shared Variables*) systemy z pamięcią rozproszoną i współdzielonymi zmiennymi.



Rys. 2.9. Architektura DMSV [7]

4. **DMMP** (ang. *Distributed Memory Message Passing*) systemy z pamięcią rozproszoną i przekazywaniem komunikatów między sobą.



Rys. 2.10. Architektura DMMP [7]

2.3. Klasyfikacja Tanenbauma

Odmienne, od przedstawionego przez Johnsona, spojrzenie na klasę MIMD przedstawił Andrew S. Tanenbaum. W swojej klasyfikacji koncentruje się on na sposobie w jaki komunikują się poszczególne systemy. Stopień rozproszenia systemów określany jest na podstawie szybkości przesyłu danych pomiędzy systemami.

W systemach ściśle powiązanych będą to setki i tysiące bitów na sekundę. Takie systemy typowe są dla przypadków, w których wiele jednostek pracuje jednocześnie nad tym samym zadaniem.

Przykładem systemów słabo powiązanych mogą być dwa komputery połączone za pomocą modemów o niskiej prędkości przesyłu danych. Te systemy również mogą pracować na rzecz jednego projektu, jednak zadania dla nich dzielone są na oddzielne części.

Komunikacja za pomocą szyny współdzielonej to przypadek, gdy wszystkie jednostki podłączone są do współdzielonej sieci, szyny, bądź łącza danych (przykładowo: kilka procesorów korzystających ze wspólnej szyny bądź grupa komputerów wymieniających informacje za pośrednictwem segmentu sieci Ethernet).

Systemy komunikujące się za pomocą przełączania charakteryzują się tym, że poszczególne jednostki mogą dowolnie zestawiać połączenia między sobą. Przykładem takiej sieci może być sieć telefoniczna z komutacją kanałów lub grupa procesorów komunikująca się za pośrednictwem przełącznicy krzyżowej.

3. PRZETWARZANIE ROZPROSZONE

Przetwarzanie rozproszone należy postrzegać jako jedno, zintegrowane środowisko obliczeniowe, bez względu na jego fizyczną lokalizację lub użytą do jego stworzenia ilość komputerów (węzłów), która waha się od paru stacji roboczych, poprzez kilkanaście – kilkaset w sieci lokalnej, aż po tysiące, a nawet miliony w sieci Internet.

Dokładnie przetwarzanie rozproszone (proces rozproszony) polega na zlecaniu często bardzo niejednolitym środowiskom obliczeniowym (heterogenicznym) wykonania różnych zadań. Zadania te, aby mogły być wykonane w środowisku rozproszonym należy odpowiednio przygotować tzn. poddać dekompozycji. Następnie po wykonaniu dekompozycji zadania, do węzłów wysyłane są odpowiednio dla nich przygotowane „fragmenty danych" w celu przetworzenia. W chwili, gdy dany węzeł zakończy przetwarzanie, wysyła jego wynik, a sam pobiera kolejny „fragment danych" do przetworzenia. Dla uzyskania wyniku, przetworzone „fragmenty danych" z węzłów są łączone w całość.

Łatwy dostęp do sieci Internet sprawił, że powszechne stało się udostępnianie zasobów komputerów osobistych na potrzeby systemów rozproszonych takich jak np. projekt SETI@home mający na celu znalezienie kontaktu z istotami pozaziemskimi poprzez poszukiwanie sztucznych sygnałów pochodzących z przestrzeni kosmicznej, czy też projekt PrimeGrid, który uczestniczy w konkursie kryptograficznym „The RSA Challenge Numbers".

Na obecnym poziomie rozwoju technologii informatycznych systemy rozproszone są normą przy tworzeniu systemów ukierunkowanych na uzyskanie jak najlepszych możliwości obliczeniowych. Mogą być wykorzystywane do tworzenia lub wspomagania systemów handlowych, przemysłowych, czy multimedialnych.

Systemy rozproszone, są także nadzieją na sprostanie wymaganiom tzw. „Grand Challenge problems", które określa się jako zagadnienia, których nie da się obliczyć (rozwiązać) w rozsądnym czasie, przy użyciu obecnie wykorzystywanych technologii informatycznych.

Przykładem „Grand Challenge problems" mogą być różnego rodzaju symulacje związane z klimatem dotyczące np. obliczeń związanych z globalnym ociepleniem, przewidywaniem pogody, czy ruchem planet w przestrzeni.

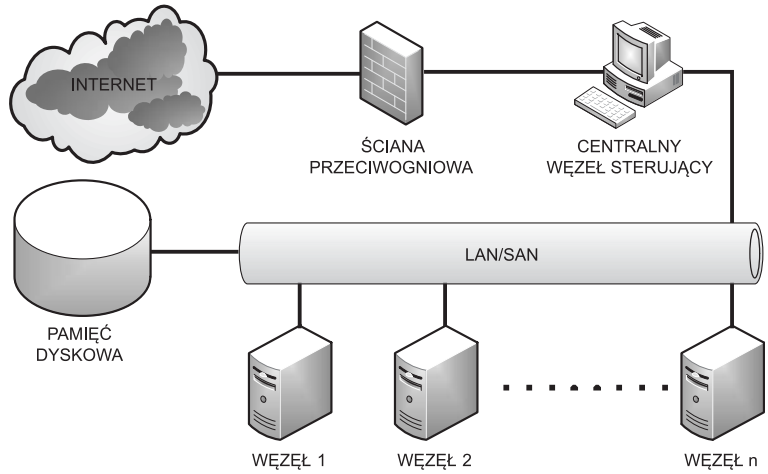
4. WYBRANE ROZWIĄZANIA KLASTROWE

4.1. Definicja klastra

Klaster komputerowy (ang. *cluster*) jest to grupa wspólnie działających połączonych maszyn, które tworzą pojedynczy zespół obliczeniowy, może być też postrzegany jako pojedyncza maszyna.

Klastry składają się z węzłów (ang. *node*), czyli pojedynczych maszyn, które są połączone za pomocą wydajnej sieci – zwykle szybkiej sieci lokalnej (ang. *Local Area Network – LAN*). Ilość węzłów w klastrze może wahać się od kilku do kilku tysięcy.

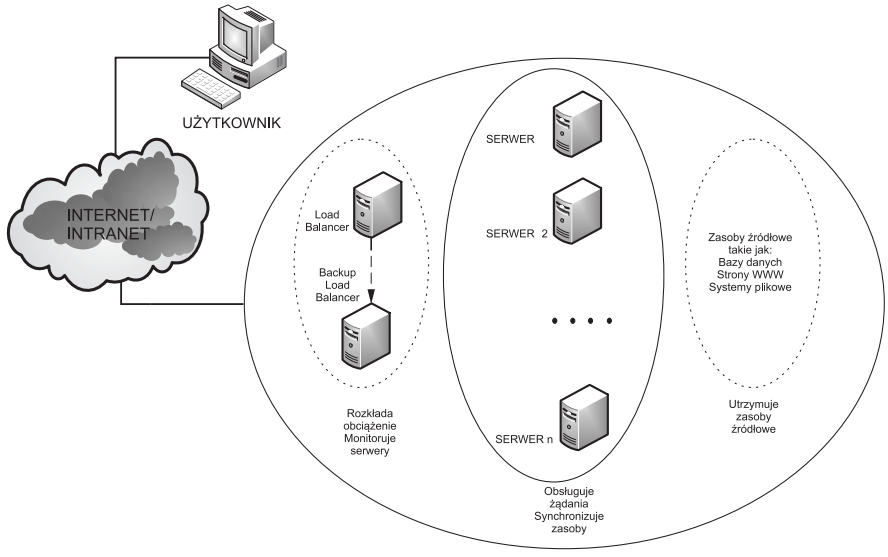
Architektura klastra zależy od danej implementacji. Większość architektur klastra opiera się na modelu, w którym mamy do czynienia z węzłami połączonymi w jeden zespół obliczeniowy za pomocą sieci/magistrali. W modelu tym wyróżnia się jeden węzeł, który jest centralnym punktem klastr a, a przez który możliwe jest sterowanie całym klastrem. Niekiedy wyposaża się klaster w dodatkowe terminale dostępowe oraz systemy pamięci masowych.



Rys. 4.1. Ogólny model klastra komputerowego

4.2. Klastry równoważące obciążenie (ang. *Load Balancing Cluster*)

Nazywane są też klastrami serwerowymi (ang. *server clusters*). Służą do utrzymywania w działaniu mocno obciążonych usług sieciowych takich jak np. serwery WWW, czy bazy danych. Działanie ich polega na równoważnym dystrybuowaniu obciążenia pomiędzy poszczególne węzły klastra. Klastry tego typu implementuje się w przypadku, gdy bardzo istotny jest czas reakcji usługi na żądania klienta.

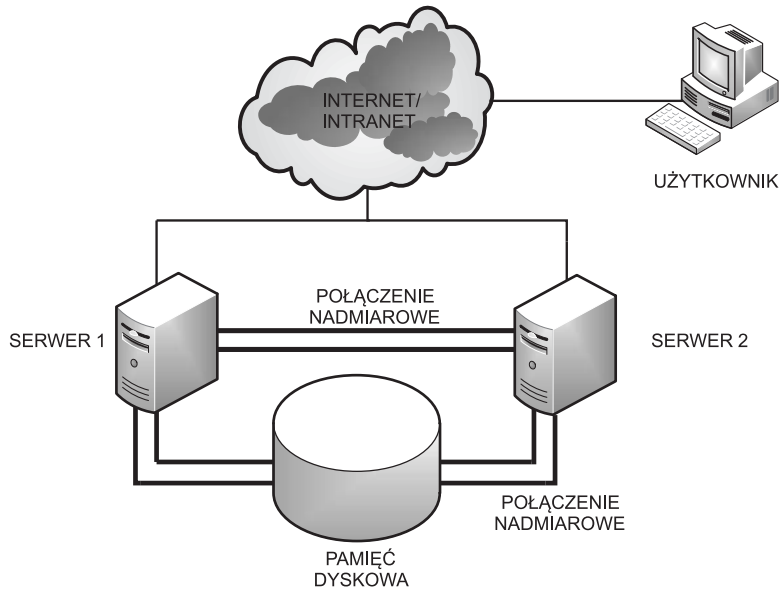


Rys. 4.2. Przykładowa budowa klastra równoważącego obciążenie

W powyższym przykładzie serwery *1, 2 … n* dysponują własną mocą obliczeniową, ale posiadają też pewne zasoby, które są synchronizowane z serwerem do tego przeznaczonym lub np. z pamięcią masową. Serwery *1, 2 … n* są stale monitorowane przez Load Balancer. Użytkownik chcący skorzystać z zasobów łączy się z Load Balancerem, a ten znając obciążenie serwerów *1, 2 … n* przekierowuje go do tego, który w danej chwili ma najmniejsze obciążenie, przy czym cały ten proces nie jest widoczny dla użytkownika korzystającego z usług serwera. Na koniec w przypadku zmiany dokonanej na zasobach serwer synchronizuje swoje zasoby z zasobami źródłowymi.

4.3. Klastry niezawodnościowe (ang. *High Availability Clusters*)

Zadaniem tych klastrów nie jest zwiększanie wydajności, a wyeliminowanie tak zwanego pojedynczego punktu awarii (ang. *Single Point of Failure – SPOF*). Działanie ich polega na rozłożeniu (w przypadku wystąpienia awarii któregoś z serwerów) jego zadań na pozostałe serwery, w taki sposób, aby nie było to widoczne dla użytkowników systemu. Klastry te wykorzystywane są w systemach o znaczeniu krytycznym (ang. *Mission Critical Systems*). Gdy zostanie stwierdzona niedostępność usług serwera 1, serwer 2 przejmie jego rolę. Łącze nadmiarowe znajdujące się pomiędzy serwerami pozwala serwerowi 2 na próbę ponownego uruchomienia serwera 1.



Rys. 4.3. Przykładowa budowa klastra niezawodnościowego

4.4. Klastry wydajnościowe (ang. *High Performance Clusters*)

Nazywane są też klastrami do przetwarzania równoległego. Najczęściej wykorzystuje się je do uruchamiania obliczeń aplikacji inżynierskich oraz aplikacji symulacyjnych. Używane są do masowego przetwarzania danych jednego rodzaju, przy czym równoważenie obciążenia najczęściej leży w gestii samych aplikacji, jednak nie jest to regułą. W klastrach tych duże znaczenie ma aplikacja, którą najczęściej należy przygotować w jednej z dostępnych często specjalizowanych bibliotek programistycznych. Jej duże znaczenie wynika z faktu, iż to zazwyczaj od niej zależy w jakim stopniu będzie możliwe odpowiednie rozłożenie obciążenia, a przez to i przyśpieszenie obliczeń.

Literatura

[1] R. Buyya, *High Performance Cluster Computing: Architectures & Systems,* Published by Prentice Hall PTR, June 1999.
[2] D. Chaładyniak, *The role of the cluster servers in parallel computing of hydrometeorological data*, XIV-th International Scientific and Technical Conference, The Part of Navigation in Support of Human Activity on the Sea, Gdynia, 2004.
[3] D. Chaładyniak, *Some aspects of parallel computing in geodesic and cartographic applications*, Polish Journal of Environmental Studies, Vol. 15, No. 3c, 2006.
[4] D. Culler, J. Singh P., A. Gupta, *Parallel Computer Architecture: A Hardware/Software Approach,* Published by Morgan Kaufmann, August 1998.
[5] J. Dongarra, I. Foster, G. Fox, K. Kennedy, A. White, L. Torczon, W. Gropp, *The Sourcebook of Parallel Computing*, Published by Morgan Kaufmann, November 2002.
[6] M. Hajder, H. Loutskii, W. Stręciwilk, *Informatyka – wirtualna podróż w świat systemów i sieci komputerowych*, Wydawnictwo WSIiZ, Rzeszów, 2002.
[7] K. Lal, T. Rak, *Linux a technologie klastrowe*, MIKOM, Warszawa, 2005.