Dr inż. Jacek Markus
Warszawska Wyższa Szkoła Informatyki
jacek.markus@gmail.com

Mechanizmy zapewnienia wysokiej dostępności aplikacji wielowarstwowych korzystających z relacyjnych baz danych Microsoft SQL.

High availability in multi-layer application with relational databases Microsoft SQL.

Streszczenie

Dostępność aplikacji firmowych jest krytyczna, a ich niedostępność może powodować straty zarówno finansowe jak i prestiżu, które w wielu przypadkach mogą nawet doprowadzić do upadku firmy.

Słowa kluczowe: aplikacje wielowarstwowe, wysoka dostępność, przetwarzanie w chmurze, replikacja, mirroring, always-on, Microsoft SQL

Abstract

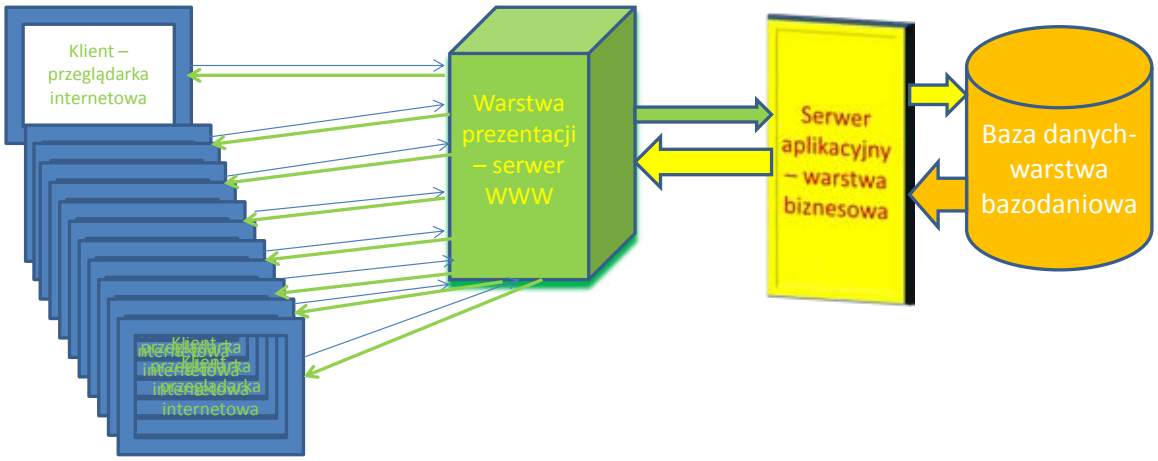
High availability of corporate applications is critical, and their unavailability can cause loss of both financial and prestige, which in many cases may even lead to the collapse of the company.

Keywords: multi-layer application , high availability, cloud computing, replication, mirroring, always-on, Microsoft SQL

1. APLIKACJE WIELOWARSTWOWE.

Większość obecnie tworzonych aplikacji biznesowych ma budowę wielowarstwową.

Najbardziej zewnętrzna warstwa prezentacji bardzo często wykorzystuje technologię webową, co pozwala wykorzystać w wielu przypadkach jako klienta standardową przeglądarkę internetową. Warstwa aplikacji przetwarzająca dane biznesowe w mniejszych systemach bywa w dużym stopniu zintegrowana z warstwą prezentacji i warstwą bazodanową dzięki wykorzystaniu wyzwalaczy, procedur składowanych i obiektów CLR.



Rys.1. Architektura aplikacji wielowarstwowej

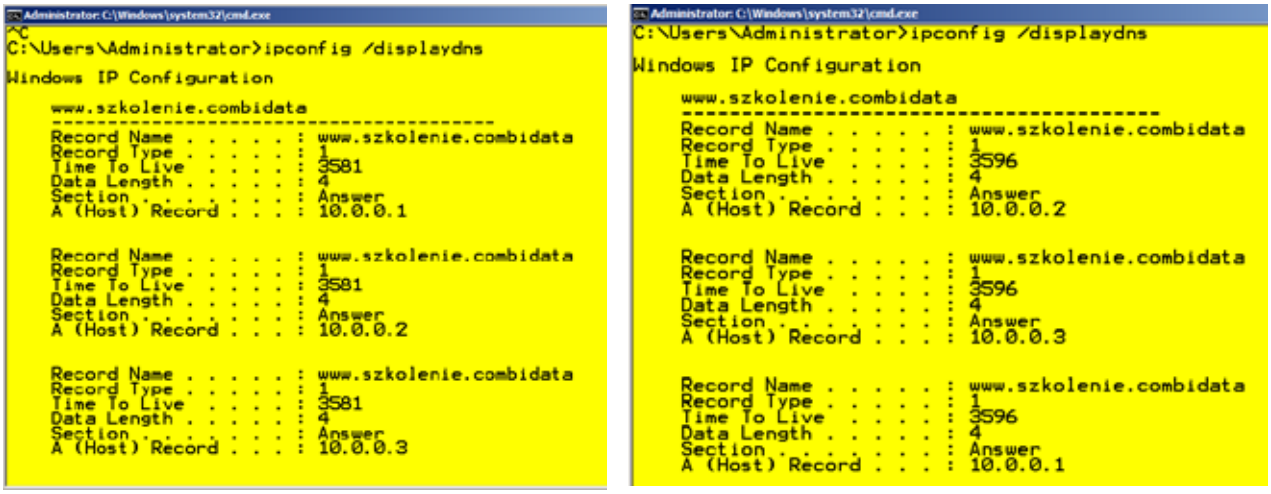
W systemach biznesowych krytyczna jest warstwa bazodanowa, której spójność i bezpieczeństwo przechowywanych danych jest najwyższym priorytetem. Dane te muszą być chronione przed nieautoryzowanym dostępem przy jednoczesnym zapewnieniu ich pełnej dostępności.

2. WYSOKA DOSTĘPNOŚĆ WARSTWY PREZENTACJI I APLIKACJI.

Większość mechanizmów zapewnienia wysokiej dostępności jest związane z wprowadzeniem nadmiarowości prowadzących do wyeliminowania punktów pojedynczej awarii. W sytuacji jeśli dostępność aplikacji zależy od jednego dysku w serwerze, jednego zasilacza, jednego procesora itp., awaria pojedynczego elementu doprowadza do braku dostępności aplikacji. Podniesienie prawdopodobieństwa bezawaryjnej pracy komponentu na wyższy poziom powoduje często logarytmiczny wzrost kosztów wdrożenia i nigdy nie pozwala na pełną gwarancję niezawodności. Wprowadzenie nadmiarowości pozwala często uzyskać obniżenie prawdopodobieństwa niedostępności na wyższym poziomie i często niższym kosztem niż podnoszenie jakości składowych ponad pewien poziom. W przypadku dysków wprowadza się macierze dyskowe z wykorzystaniem różnych poziomów RAID, które związane z nadmiarowymi dyskami, pozwalają na dalszą pracę systemu w razie awarii jednego z dysku, a w niektórych konfiguracjach nawet kilku dysków do czasu wymiany uszkodzonego dysku. W składnicach danych wprowadza się dodatkowo pulę dysków awaryjnych, które automatycznie zastępują uszkodzone dyski. Należy również zwrócić uwagę, że nie zawsze wprowadzenie dwóch komponentów pozwala na wyeliminowanie pojedynczego punktu awarii np. zamontowanie w serwerze o poborze mocy 1KW dwóch zasilaczy o mocy 0,75 KW, awaria jednego z nich spowoduje, że jeden zasilacz nie dostarczy komponentom serwera odpowiedniego zasilania. Analogicznie wprowadzenie dwóch linii dostępu do Internetu, gdy światłowody są poprowadzone obok siebie do dostawców internetu w przypadku uszkodzenia jednego ze światłowodów np. w czasie pracy koparki łączy się z bardzo wysokim prawdopodobieństwem awarii drugiego. Pierwszym etapem zapewnienia wysokiej dostępności jest wyeliminowanie pojedynczych punktów awarii.

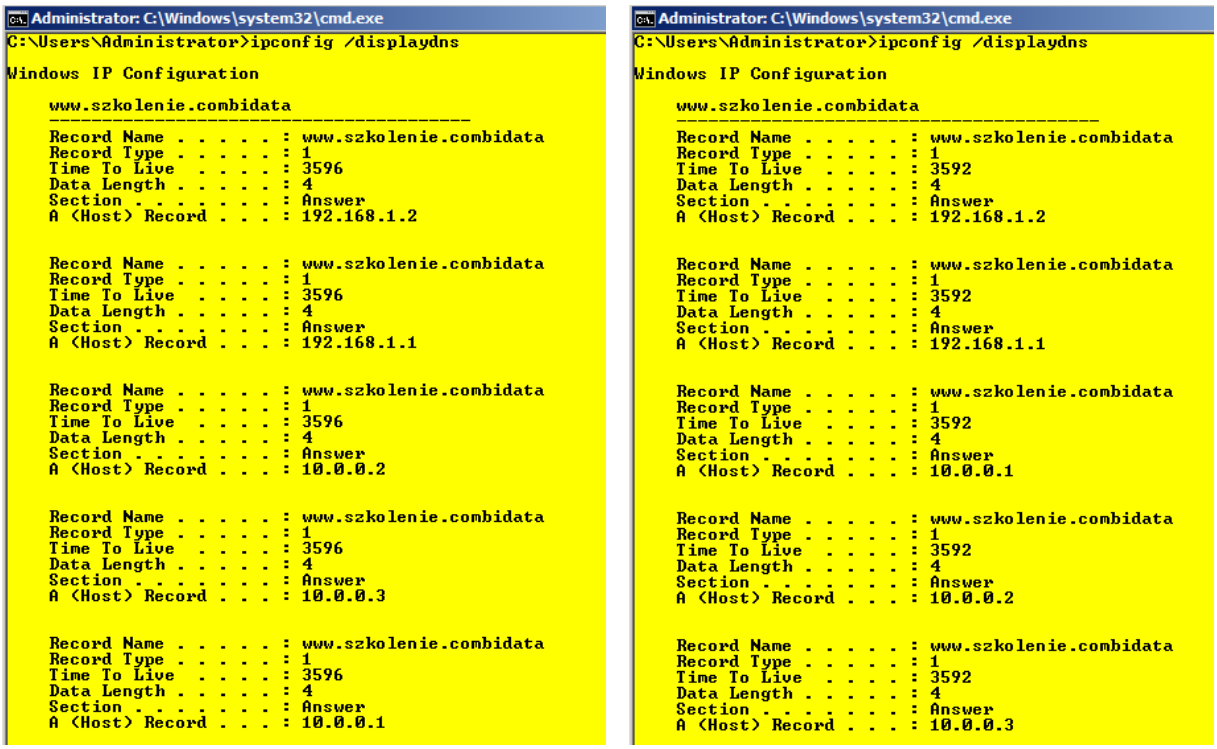
Istnieje wiele mechanizmów zapewnienia wysokiej dostępności warstwy biznesowej i warstwy prezentacji z wykorzystaniem nadmiarowych serwerów.

Najłatwiejszym w implementacji jest wykorzystanie **DNS round robin (RRDNS)**. Polega na utworzeniu na serwerze DNS kilku rekordów DNS wskazujących dla tej samej nazwy hosta kilka rekordów DNS. Lista tych rekordów jest przesyłana po otrzymaniu zapytania posortowana w określonej kolejności. Kolejne zapytanie otrzymuję powyższą listę ze zmienioną kolejnością: pierwszy rekord z poprzedniego zapytania jest przesortowany na koniec listy.



Rys.2. Wynik zapytania z wykorzystaniem round robin

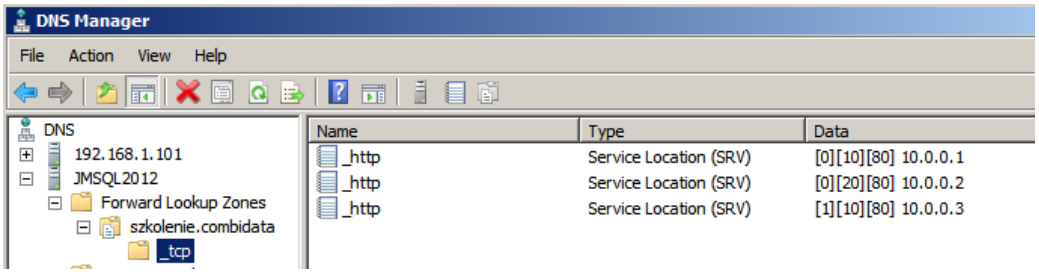
Dodatkowo można użyć netmask ordering polegającego na grupowaniu adresów z jednej klasy i sortowaniu w obrębie grupy. Dzięki temu klient otrzymuje na „górze listy" adresy z „bliższego" adresu sieciowego.



Rys.3. Wynik zapytania z wykorzystaniem round robin i netmask ordering

RRDNS pozwala na rozłożenie obciążenia na wiele serwerów, jednak awaria jednego z serwerów powoduje, że część zapytań trafia na niedziałający serwer.

Mechanizmem DNS pozwalającym rozłożyć zapytania na wiele serwerów z równoczesną obsługa awarii jednego lub kilku z nich jest użycie **DNS Load Balancing (DNSLB)**. Mechanizm ten wykorzystuje **rekordy SRV** do zdefiniowania preferencji rozłożenia obciążenia na wiele serwerów. Jedną z bardziej znanych usług korzystających z tego mechanizmu jest usługa Active Direktory. Z uproszczonej wersji tego mechanizmu opartego o rekordy MX korzysta również usługa SMTP, z której korzystają wszystkie współczesne i popularne systemy pocztowe (są oczywiście rozwiązania niszowe, które wykorzystują inne mechanizmy). Rekord SRV odwołuje się do odpowiedniego hosta poprzez definicję jego IP lub wskazanie rekordu A albo rekordu CNAME. Rekord SRV wskazuje usługę protokół TCP, UDP, TLS… i port na którym usługa pracuje na podstawie zarejestrowanych mapowań nazwy usługi oraz aliasów jej nazwy na port w pliku **%systemroot%\System32\drivers\etc\services**.



Rys.4. Rekordy SRV

Kolejność przesortowania rekordów określa priorytet, który grupuje rekordy, czyli rekord o priorytecie 0 obsługuje cały ruch, jeśli wszystkie serwery wskazane przez rekordy o priorytecie O są niedostępne, aplikacja odpytuje serwery usługi wskazane przez rekordy o priorytecie 1… Waga określa proporcje rozłożenia ruchu pomiędzy serwery o tym samym priorytecie.

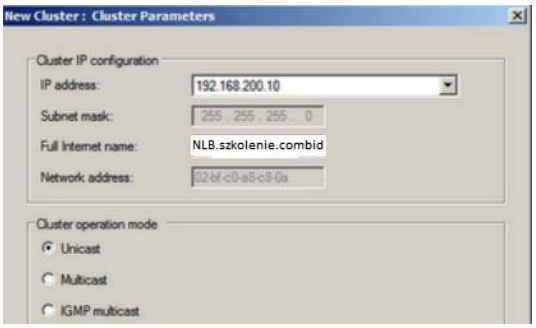
Wykorzystanie DNSLB nie tylko do rozłożenia obciążenia na wiele serwerów, ale również z obsługą

awarii jednego lub kilku z nich jest związane z możliwościami aplikacji klienckich, które po przekroczeniu czasu oczekiwania odpowiedzi z serwerów wskazanych przez pierwszy rekord z listu odpytują kolejny oraz wbudowaniem w aplikację mechanizmów automatycznego wznawiania sesji. Choć wiele aplikacji klienckich oraz usług aplikacyjnych obsługuje mechanizm DNSLB, to obecnie popularne przeglądarki internetowe go nie obsługują.

Alternatywą jest użycie. W klastrze NLB jest kilka hostów, które korzystają ze wspólnej wirtualnej karty sieciowej.

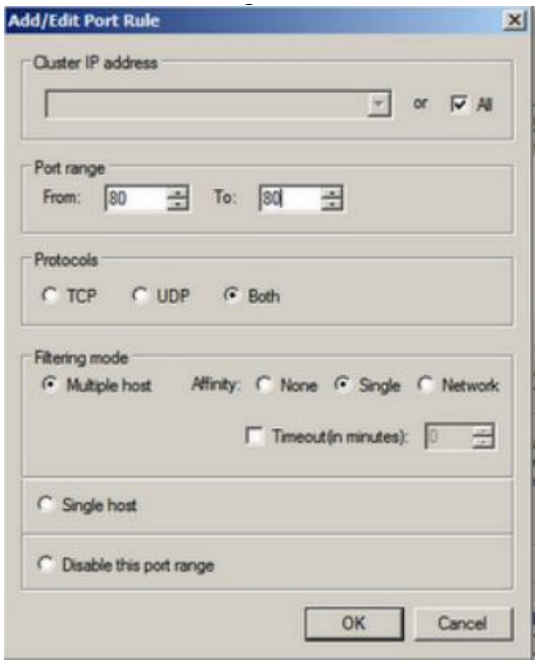
Rys.5. Wykorzystanie NLB

Zaletą rozwiązanie jest taki sam adres IP na wszystkich hostach w klastrze, co uniezależnia równoważenie obciążenia od usługi DNS, a obsługę awarii jednego z hostów w klastrze NLB od aplikacji klienckiej. Rozwiązanie pomimo, że jest stosunkowo tanie w implementacji (np. wersje serwerowe Windows poczynając od 2003 Standard i Web Edition go obslugująm jak również jego obsługa może być wkompilowana w jądro dystrybucji Linux) i nie wymaga zwykle inwestycji w drogi sprzęt. Przy konfiguracji NLB należy pamiętać, że wszystkie hosty korzystają z wirtualnej karty sieciowe o adresie MAC i IP wspólnym dla całego klastra NLB. Dlatego należy pamiętać o odpowiedniej konfiguracji przełącznika, aby do wszystkich portów obsługujących hosty z klastra NLB przypisać statycznie MAC adres klastra, w przeciwnym przypadku przełącznik automatycznie zapamięta pierwszy port, z którego otrzymał odpowiedź i cały ruch będzie kierowany tylko na jednego z hostów w klastrze NLB. Klaster NLB skonfigurowany w trybie unicast korzysta z adresu MAC w postaci 02:BF: + szesnastkowo zapisany adres IP klastra NLB.



Rys.6. Ustawienie klastra NLB - równoważenia obciążenia sieciowego

W trybie unicast jest wymagana dodatkowa karta sieciowa w każdym z hostów klastra do komunikacji pomiędzy hostami niezbędnej do negocjacji, który host ma obsłużyć przychodzące połączenie oraz wykrycie hosta, który uległ awarii. W przypadku braku możliwości zestawienia komunikacji pomiędzy hostami klastra NLB na dodatkowych interfejsach sieciowych, można użyć trybu multicast, gdzie MAC adres sklastrowanej mechanizmem NLB MAC adres karty sieciowej nie jest ukrywany, a wirtualna karta sieciowa NLB ma adres z prefiksem 03:BF.



Rys.7. Ustawienie roli portu

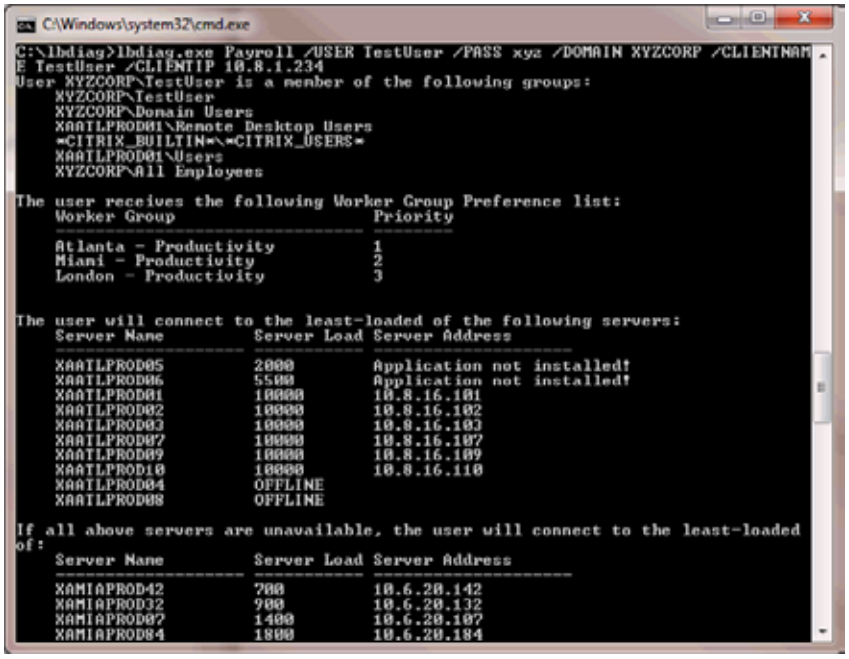
W trybie single host preferencję użycia serwerów w klastrze NLB określają priorytety. Dla danego protokołu, portu TCP/UDP lub zakresu portów cały ruch obsługuje host o najniższym priorytecie, w razie jego awarii ruch obsługuje aktywny host o aktualnie najniższym priorytecie. Jeśli zdefiniujemy tryb pracy Multiple host (najczęściej stosowany, ponieważ pozwala na dynamiczne równoważenie obciążenie) zamiast priorytetu definiujemy wagę, ruch jest rozkładany na wszystkie aktywne hosty w klastrze NLB proporcjonalnie do przypisanej do hosta wagi.

Rys.8. Waga portu

Domyślnie dla trybu Multiple host jest ustawienie Equal, czyli równomierne rozłożenie sesji pomiędzy serwery w klastrze NLB. Przychodzące połączenie obsługuje host w klastrze, który „zadeklaruje" najniższy koszt przyjęcia połączenia. Trochę jak w życiu, dodatkową pracę otrzymuje zwykle ten, kto najmniej broni się przed jej przyjęciem (najniższy koszt przyjęcia).

W przypadku równoważenia obciążenia na podstawie wag problemem mogłyby być protokoły z zabezpieczeniem transmisji poprzez SSL lub TLS (zwłaszcza wielosesyjne jak http, https). W trakcie negocjowania połączenia klient otrzymuje od serwera klucz publiczny, który wykorzystuje do zaszyfrowania symetrycznego klucza sesji. Zaszyfrowany kluczem publicznym symetryczny klucz sesji jest wysyłany do serwera, który za pomocą klucza prywatnego dokonuje deszyfracji symetrycznego klucza sesji. Dalsza komunikacja pomiędzy klientem i serwerem jest szyfrowana za pomocą symetrycznego klucza sesji, który „zna" klient i serwer- jeden z serwerów w klastrze NLB. Gdyby sesja została przeniesiona na inny serwer w klastrze NLB, który nie posiada aktualnego klucza symetrycznego sesji nastąpiłoby zerwanie sesji. Przed takimi zdarzeniami zabezpiecza mechanizm Affinity: serwer w klastrze NLB zapamiętuje IP klienta, który z nim nawiązał połączenie (IP klienta – w domyślnym ustawieniu Single Host, a w szczególnej konfiguracji pierwsze trzy bajty adresu IPv4 - Klasy C, stosowanej gdy obsługujemy klientów

łączących się zza Array Proxy) i deklaruje dla niego najniższy koszt przyjęcia połączenia (0). W takim przypadku sesja zostanie przeniesiona na inny serwer w klastrze NLB wyłącznie w przypadku awarii serwera dotychczas obsługującego połączenie. Implementacja NLB monitoruje na serwerach Windows pracę usługi WLBS, która weryfikuje dostępność hosta w klastrze NLB, ale nie monitoruje pracy usług. Niezależnymi mechanizmami, należy również zapewnić spójność zawartości witryn i ustawień aplikacji. Z klastra NLB mogą korzystać aplikacje korzystające wyłącznie z protokołów TCP i UDP. Inne protokoły nie mogą korzystać z NLB, np. niestosowane już od bardzo dawna w nowych aplikacjach NetBT, to jednak z powodu zaszłości historycznych spotykany w wielu firmach (był cenny w czasach, gdy używano w sieciach różnych protokołów IPX/SPX, NetBEUI, Appletalk, a protokół TCPIP jeszcze nie pozwolił na stworzenie globalnego i powszechnie dostępnego internetu, ponieważ uniezależniał aplikacje klient-serwer od podanych przykładów protokołów transportowych, wystarczyło użycie dodatkowej warstwy NETBIOS).



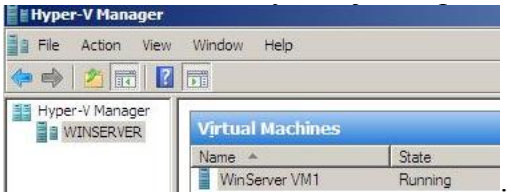
Rys.9. Równoważenie ładowania aplikacji

Na rynku dostępne są rozwiązania **Aplication Load Balancing (ALB)** stosowane w celu rozłożenia obciążenia i obsługę awarii serwerów warstwy aplikacji. Działanie rozwiązań polega zwykle na odpytywaniu usługi z listy serwerów w klastrze aplikacyjnym i sortowanie tej listy według rosnących czasów odpowiedzi. „Na górze" listy jest więc serwer, który najszybciej odpowiedział, więc najprawdopodobniej jest najmniej obciążony w stosunku do posiadanych zasobów. Jest to stosunkowo efektywne rozwiązanie, które w przeciwieństwie do NLB monitoruje pracę usługi (jeśli usługa z serwera nie odpowiedziała, to nie ma go na liście serwerów do odpytania dla warstwy prezentacji, pojawi się, gdy na kolejne zapytanie usługa odpowie w zdefiniowanym przedziale czasu.

Rys. 10. Sprzętowe równoważenie obciążenia HLB

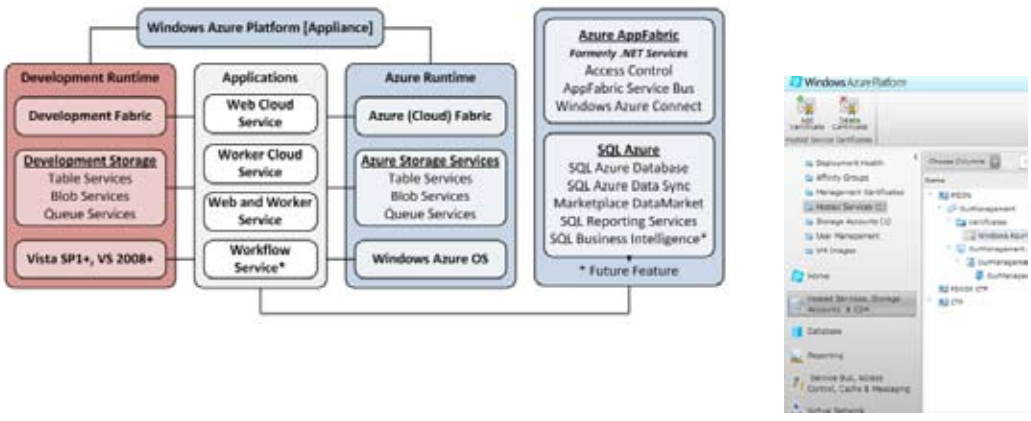
Oprócz rozwiązań softwarowych jest dostępnych na rynku wiele rozwiązań sprzętowych **Hardware Load Balancing (HLB)**, które pozwalają zarówno na wykorzystanie mechanizmów NLB jak ALB zależnie od wybranego modelu urządzenia. Rozwiązanie sprzętowe pozwalają na użycie równoważenia obciążenia usług dla aplikacji, które nie obsługują NLB np. pula enterprise serwerów FrontEnd Office Communication Server, albo Lync (Lync 2010 obsługuje DNSLB dla protokołu SIP, ale nie HTTPS).

W warstwach prezentacji i aplikacyjnej rzadziej stosuje się usługi klastrowe **cluster services (CS)**, ponieważ w wielu rozwiązaniach są to klastry niezawodnościowe failover pracujące w modelu shared-nothing, gdzie grupa zasobów w tym klastrowana usługa pracuje na jednym węźle klastra. W razie jego awarii jest przenoszona na inny węzeł klastra, podobnie jak NLB w trybie single host. Usługi klastrowe zdecydowanie częściej stosuje się do zapewnienia wysokiej dostępności warstwy bazodanowej, o czym później. W rozwiązaniach wysokiej dostępności stosuje się wirtualizację.



Rys. 11. Wirtualizacja

Wirtualna usługa lub cała wirtualna maszyna pracuje na jednym z serwerów hypervizora, w przypadku jego awarii lub przeciążenia wirtualna maszyna lub wirtualna usługa jest przenoszona na inny serwer hypervizora. Dostępne są rozwiązania pozwalające na przeniesienie wirtualnej usługi lub maszyny bez jej restartu – Live Migration. Do dominujących na rynku rozwiązań można zaliczyć: VMware. Microsoft Hyper-V, Citrix XenServer.

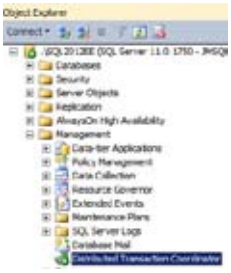


Rys. 12. Chmura obliczeniowa

Do zrównoważenia obciążenia oraz zapewnienia wysokiej dostępności coraz częściej wykorzystuje się usługi **w chmurze obliczeniowej Cloud Computing**, będące rozwinięciem wirtualizacji. Do bardziej znanych należą rozwiązania Cloud computing firm: Amazon, Rackspace, Salesforce, Microsoft, IBM, Google. Wykorzystywana chmura obliczeniowa może być prywatna, zbudowana na sprzęcie będącym w posiadaniu firmy lub wydzierżawionym albo publiczna, korzystająca ze sprzętu dostawcy chmury obliczeniowej. Planując użycie chmury do równoważenia obciążenia i zapewnienia wysokiej dostępności usług, szczególnie w modelu hybrydowym, gdzie szczytowe i zasobożerne obliczenia wykonywane są w chmurze publicznej (np. przeliczanie grup miar), a dane słownikowe (np. hierarchie wymiarów) na serwerach firmowych, należy uwzględnić wyeliminowanie pojedynczych punktów awarii połączenia do internetu oraz jego odpowiednią przepustowość.

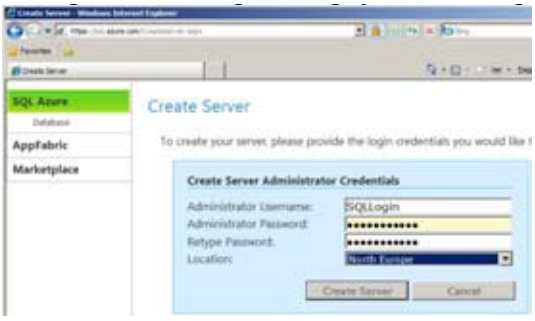
3. WYSOKA DOSTĘPNOŚĆ WARSTWY BAZODANOWEJ

Dostępność warstwy składowania danych jest krytyczna dla działania aplikacji biznesowych. Omówione wcześniej rozwiązania zapewnienia wysokiej dostępności warstw prezentacji i aplikacji zwykle nie mają zastosowania dla warstwy bazodanowej, ze względu na konieczność zapewnienia spójności danych, której utrzymanie np. w klastrze NLB w wielu przypadkach może być bardziej kosztowne od korzyści, choć każde wdrożenie należy rozpatrywać indywidualnie. Umieszczenia baz danych na serwerach aplikacyjnych pracujących w klastrze NLB lub ALB ma najczęściej zastosowanie, gdy bazy są w trybie do odczytu, zasilane z serwera centralnego, alternatywą jest lokalny serwer, na którym zapisywane są dane pochodzące z serwera aplikacyjnego, a następnie replikowane z centralną składnicą danych. Wadą takiego rozwiązania mogą być opóźnienia w synchronizacji baz lokalnych z centralną składnicą danych, jednak ze względów wydajnościowych są stosowane. Alternatywnym rozwiązaniem jest użycie transakcji rozproszonych, których spójność zapewnia usługa **Microsoft Distributed Transaction Coordinator (MsDTC)**. Należy jednak rozważyć efekt w postaci potencjalnego wydłużenia czasu trwania transakcji, zwiększoną ilość blokad, co może prowadzić do spadku wydajności.



Rys.13. Koordynator tranzakcji rozproszonychMicrosoft

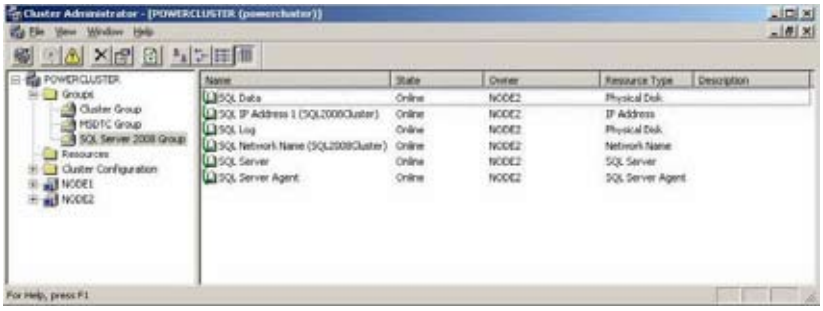
Jest dostępna wersja **Microsoft SQL Azur**e, który jest składnicą danych przechowywaną w chmurze. W wielu rozwiązaniach może to być dobry wybór zapewniający wysoką dostępność, szczególnie, gdy warstwa aplikacji i prezentacje są umieszczone w chmurze.



Rys.14. SQL Azure

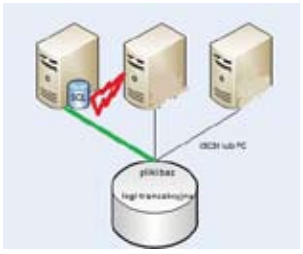
Należy jednak pamiętać, że nie jest to silnik Microsoft SQL 2012 uruchomiony w chmurze, tylko niezależny produkt i przeniesienie składnicy danych ze „zwykłego" Microsoft SQL Server do SQL Azure może wymagać istotnych zmian zarówno w bazie danych jak i aplikacji, która z tej bazy korzysta.

Jednym z bardziej znanych sposobów zapewnienia dostępności bazy danych jest instalacja **Microsoft SQL Serwer na usługach klastrowych - Microsoft Clustering Services (MSCS)**.



Rys.15. SQL na usłudze klastrowej

W klasycznym rozwiązaniu klastrowym pomiędzy węzłami klastra jest współdzielona magistrala, do której podpięte są składnice danych. Najczęściej wykorzystywane są technologie Fibre Chanel i iSCSI. Bazy danych i logi transakcyjne znajdują się na macierzach na magistrali współdzielonej.

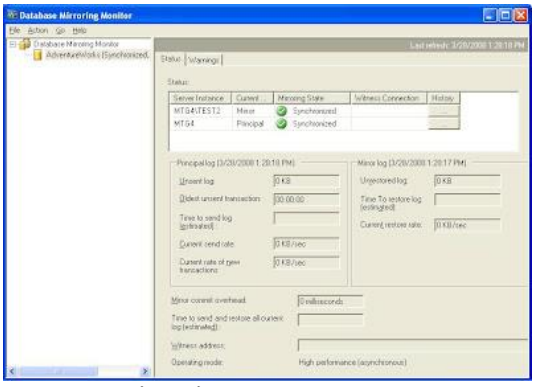


Rys.16. Baza na magistrali współdzielonej klastra

W przypadku awarii węzła klastra, na którym pracuje usługa SQL Server (i inne usługi SQL), w wyniku failover następuje restart usług na innym węźle klastra. Przy starcie bazy i logi są montowane z dysku współdzielonego. W trakcie startu następuje naprawa bazy. Po failover instancja SQL jest dostępna pod tą samą nazwą i tym samym adresem IP. Jednak czas przełączenia po failover, zależny od czasu startu usługi, w tym naprawy baz danych.

W przypadku użycia **mirroringu baz** czasy przełączenia w przypadku awarii liczone są w milisekundach. W przeciwieństwie do usług klastrowych mirrorowana nie jest cała instancja, a jedynie baza użytkowników. Niezależnie należy zapewnić spójność informacji przechowywanych w bazie master np. loginów. Przy zastosowaniu mirroringu w trybie synchronicznym można użyć dodatkowego serwera świadka (Witness Server) monitorującego stan pracy baz i w razie awarii automatycznie przełączającego bazy. Po failover następuje zmiana nazwy instancji do której łączy się aplikacja. Zapewnia to Native Client, gdy w connection string podamy połączenie do głównego i awaryjnego serwera np.

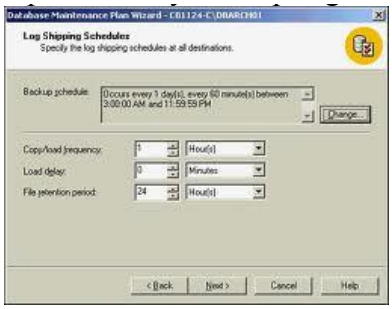
Data Source=myServerAddress;
Failover Partner=myMirrorServerAddress;
Initial Catalog=myDataBase;Integrated Security=True;



Rys.17. Mirroring

W mirroringu transakcje zachodzą w bazie principal i są replikowane do bazy mirror. Baza danych ma więc swoją pełną kopię na drugim serwerze, czyli należy przewidzieć dwie macierze dyskowe. Mirroring może pracować w trybie asynchronicznym, gdy transakcja jest zamykana natychmiast w bazie principal, a zmiany są replikowane do bazy mirror. Użycie trybu asynchronicznego, który choć jest szybszy, może doprowadzić w razie awarii do utraty części transakcji w bazie mirror, które zostały zamknięte w bazie principal. Ponadto w trybie asynchronicznym zachodzi konieczność ręcznego przełączenie w razie awarii. Przy synchronicznym trybie pracy transakcja w bazie principal jest zamykana po zreplikowaniu skompresowanego logu do serwera z bazą mirror (od wersji SQL 2008, w SQL 2005 dopiero po zamknięciu transakcji w bazie mirror, ponadto transmitowany log nie był kompresowany). W przypadku mirroringu (SQL 2008/2008R2/2012 Enterprice Edition) została podniesiona również odporność na uszkodzenie stron danych. Wykryta uszkodzona strona danych w bazie principal jest automatycznie naprawiana poprzez skopiowanie z bazy mirror, a uszkodzona strona danych w bazie mirror naprawiana poprzez skopiowanie z bazy principal. Mechanizm ten minimalizuje problem związany z czasochłonnym naprawianiem uszkodzonych stron danych poprzez odzyskiwanie bazy z backupu, w czasie którego grupa plików z uszkodzoną stroną danych lub cała baza jest niedostępna produkcyjnie. W mirroringu biorą udział dwie bazy, z których baza mirror nie jest dostępna dla użytkowników wprost. Istnieje możliwość wykonania na bazie mirror snapshotu, który jest dostępny produkcyjnie w trybie do odczytu i może być wykorzystany do robienia raportów.

Mechanizmem znanym już w SQL 2000 był **log shipping**. Baza primary ma wprowadzony backup logów transakcyjnych, które są odtwarzane na bazach secondary.

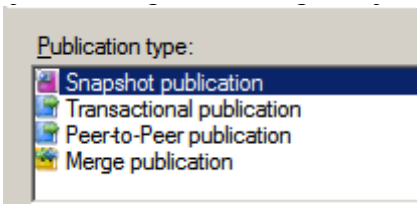


Rys.18. Log shipping

Baz secondary może być kilka. Opóźnienia w odtworzeniu transakcji w bazach secondary mogą być liczone w minutach w stosunku do bazy primary. W log shippingu brak mechanizmu automatycznego przełączenia bazy i przełączenie w razie awarii musi być wykonywane ręcznie. Bazy secondary w czasie odtwarzania logu są niedostępne produkcyjne, poza odtworzeniami mogą być dostępne w trybie do

odczytu, ale przy rozpoczęciu odtwarzania backupu logu połączenia są zrywane, ponadto zwiększają się opóźnienia baz secondary w stosunku do primary wynikające z oczekiwania na zamknięcia połączeń i przełączenia bazy przed rozpoczęciem odtwarzania logu transakcyjnego.

Jeśli nie chcemy synchronizować całej bazy użytkownika można rozważyć użycie replikacji. Przy definiowaniu publikacji można wybrać obiekty do **replikacji**, ponadto wybrane tablice (lub widoki) można odfiltrować zarówno poziomo jak i pionowo. W replikacji bierze udział: **publisher** – serwer na którym jest baza, z której udostępniana jest publikację, **distributor** – na którym udostępniona jest publikacja, często jest na tej samej instancji co publisher oraz **subscriber** – który pobiera publikację do bazy. Do wyboru jest kilka sposobów replikacji.



Rys.19. Typy replikacji

Replikacja **snapshot** polega na wykonaniu kopii i załadowaniu jej do baz subskryberów. Zwykle poprzedzana jest czyszczeniem baz subskryberów.

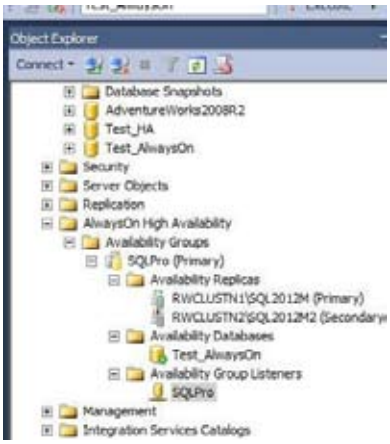
Replikacja **merge** pozwala na dwustronną replikację asynchroniczną. Może być wykorzystana do synchronizacji danych z SQL CE zabezpieczonej protokołem HTTPS. Cechą replikacji merge jest konieczność dodania do każdej replikowanej tablicy kolumny z typem danych rowguid. Jest on wykorzystywany do rozwiązywania konfliktów, gdy wiersz został jednocześnie zmieniony na kilku subscryberach, lub subscryberach i publischerze.

Replikacja **transakcyjna** jest replikacją synchroniczną, szczególnie efektywną, gdy chcemy zmniejszyć ilość replikowanych danych, ponieważ zmiany są replikowane na podstawie logu transakcyjnego

Szczególnym przypadkiem replikacji transakcyjnej jest replikacja **Peer to Peer**, wprowadzona w Microsoft SQL 2008. Pozwala ona na transakcyjną, synchroniczną replikację wielostronną, replikowane mogą być nie tylko zmiany z bazy publishera, ale również zmiany z baz subscryberów.

Rys.20. Replkacja peer to peer

W Microsoft SQL 2012 wprowadzony kolejny mechanizm zapewnienia wysokiej dostępności **always -on**. Technologia ta łączy zalety mirroringu, replikacji i MSCS, którego komponenty podobnie jak DAG Exchange 2010 wykorzystuje do obsługi faiklover. W always-on Microsoft SQL 2012 definiowana jest **AlwaysOn Availability Group**, w obrębie której jest podstawowa replika bazy **primary replica** w trybie zapisu i odczytu, oraz do czterech dodatkowych replik **secondary replica**, które mogą być udostępnione w trybie tylko do odczytu i backupowane. Jest obsługiwany zarówno automatyczny, jak manualny planowany lub wymuszony failover. Usługa Availability group listeners pozwala na przekierowanie żądania aplikacji w zależności od typu zapytania do primary replica, która jest w trybie zapisu lub secondary replica w trybie odczytu. Listener zapewnia również szybki failover w razie awarii poprzez zamianę jednej z secondary replika w primary replika. Failover jest definiowana w **failover policy**.



Rys.21. Always-on

Analogicznie do mirroringu następuje automatyczne detekcja i naprawa uszkodzonych stron danych. Replikacja pomiędzy primary a secondary replika wspiera kompresję i szyfrowanie. Always-on w stosunku do mirroringu wyróżnia się dwoma istotnymi zaletami: zamiast jednej bazy mirror można użyć od jednej do czterech secondary replika, ponadto te repliki sa dostępne w trybie odczytu bez potrzeby robienia snspshotu.

4. PODSUMOWANIE

Jest dostępny cały szereg mechanizmów pozwalających podnieść parametry wysokiej dostępności zarówno warstwy prezentacji i aplikacji jak i szczególnie nas interesującej warstwy bazodanowej. Wybór odpowiedniego mechanizmu podyktowany jest zarówno potrzebami danego wdrożenia jak i budżetem. Planując wysoką dostępność należy zadać sobie pytanie, jakie będą konsekwencje niedostępności aplikacji. Istnieje wiele aplikacji, które przetwarzają dane w trybie asynchronicznym i niedostępność warstwy bazodanowej owocuje przyrostem długości kolejek czy to w MsMQ, service broker, czy katalogu na dysku oraz późniejszym przetworzeniem zgromadzonych danych. W takich przypadkach niedostępność warstwy bazodanowej jest akceptowalna przez czas niezbędny do odtworzenia serwera bazodanowego, który uległ awarii i można uznać, że koszty wdrożenia wysokiej dostępności są wyższe od potencjalnych strat wynikających z awarii pojedynczego serwera bazodanowego, zwłaszcza gdy nie grozi to kompromitacją w oczach klienta. Jednak istnieje wiele systemów, w których brak dostępności składnicy danych nawet przez krótki czas może spowodować nie tylko olbrzymie straty finansowe np. w wyniku wstrzymania sprzedaży w internetowym kanale dystrybucji, ale co często istotniejsze kompromitację w oczach klientów. Planując aplikacje biznesowe oprócz uwzględnienia mechanizmów wysokiej dostępności należy uwzględnić bezpieczeństwo danych. Mechanizmy wysokiej dostępności minimalizują konsekwencje pojedynczej awarii sprzętowej, czy aplikacyjnej, nie zapobiegają jednak wszystkim błędom użytkownika, czy złośliwym atakom w tym wirusom, które mogą być przyczyna skasowania lub podmiany danych. Należy więc rozważyć użycie dodatkowych mechanizmów kontrolnych takich jak np. logowanie transakcji za pomocą wyzwalaczy, cdc czy audytu w tym C2. Dodatkowo należy pamiętać o ryzyku zagłodzeniu instancji SQL przez zasobożerne zapytanie. Problem ten można zminimalizować poprzez użycia Resorce Governor.

Innym problemem może być atak DDOS na warstwę prezentacji, który może doprowadzić do zablokowania usług poprzez przeciążenie zasobów. Masowy atak może być problemem nawet dla dużych farm serwerów, teoretycznie można użyć cloud computing, dzięki czemu potrzebne zasoby do obsłużenia ogromnej liczby zapytań mogą zostać dynamicznie przydzielone, należy jednak pamiętać, że w tym przypadku płacimy za zużytą moc obliczeniową, analogicznie jak płacimy wysoki rachunek za energię

elektryczną po długotrwałym używaniu silnika wysokiej mocy, czy elektrycznym ogrzewaniu. Przy planowaniu rozwiązań o wysokiej dostępności należałoby uwzględnić detekcję takich ataków i ograniczenie przesyłania pakietów na ruterach brzegowych pochodzących od puli podejrzanych adresów IP. Np. dla aplikacji, gdzie zwykle dominują zapytania z Polski przy niewielkim udziale zapytań z UE nagle pojawiają się duże ilości pakietów z chińskich adresów IP. Planując wysoką dostępność aplikacji należy wiec uwzględnić bardzo wiele czynników. Wybrane mechanizmy zostały zaprezentowane w niniejszej pracy, które można uwzględnić przy projektowaniu nowych wdrożeń i modernizacji istniejących systemów.