Przegląd strategii przechowywania danych i odporności na awarie stosowanych w chmurze obliczeniowej

Wstęp

Przetwarzanie w chmurze zyskało w ostatnich latach znaczną popularność. Firmy takie jak Google, Amazon i Microsoft w ciągu ostatnich kilku lat budowały ogromne centra danych. Obejmujące domeny geograficzne i administracyjne, te centra danych są zwykle budowane z typowych komputerów stacjonarnych, a łączna liczba komputerów zarządzanych przez te firmy jest rzędu milionów. Dodatkowo zastosowanie wirtualizacji pozwala na przedstawienie węzła fizycznego jako zestawu węzłów wirtualnych, co daje pozornie niewyczerpany zestaw zasobów obliczeniowych. Wykorzystując ekonomię skali, te centra danych mogą dostarczać procesory, sieci i pamięć masową po znacznie obniżonych cenach, co z kolei stanowi podstawę dla wielu instytucji do hostowania swoich usług w chmurze. W tej części przyjrzymy się najbardziej dominującym strategiom przechowywania i odporności na awarie, które są obecnie używane w środowiskach przetwarzania w chmurze. Istnieje kilka ujednoliconych tematów, które leżą u podstaw analizowanych przez nas systemów.

Temat 1: Obszerne dane

Zbiory danych zarządzane przez te systemy są zwykle bardzo obszerne. Nie jest niczym niezwykłym, że te zestawy danych mają kilka terabajtów. Zbiory danych są również zazwyczaj generowane przez programy, usługi i urządzenia, w przeciwieństwie do tworzenia przez użytkownika po jednym znaku na raz. W 2000 r. raport Berkeley „Ile informacji?" podał, że w sieci było około 25-50 TB danych. W 2003 r. (ta sama grupa podała, że w sieci było około 167 TB informacji. Wielki Zderzacz Hadronów (LHC) ma produkować 15 PB/rok. Ilość generowanych danych rośnie w skali wykładniczej - tam rosną wyzwania nie tylko w zakresie efektywnego przetwarzania tych danych, ale także z podstawowym przechowywaniem.

Temat 2: Sprzęt towarowy

Infrastruktura pamięci masowej dla tych zestawów danych zwykle opiera się na standardowych dyskach twardych z dyskami obrotowymi. Ta mechaniczna natura napędów dysków ogranicza ich wydajność. Podczas gdy szybkości procesorów rosły wykładniczo, czasy dostępu do dysków nie nadążały. Rozbieżność w wydajności między czasem dostępu do procesora i dysku jest rzędu 14 000 000:1 i stale rośnie.

Temat 3: Rozproszone dane

Dany zestaw danych jest rzadko przechowywany w danym węźle i jest zwykle dystrybuowany w zestawie dostępnych węzłów. Dzieje się tak, ponieważ jeden typowy dysk twardy zwykle nie może pomieścić całego zestawu danych. Rozproszenie zbioru danych na zbiorze dostępnych węzłów jest również prekursorem dla późniejszego równoczesnego przetwarzania wykonywanego na zbiorze danych.

Temat 4: Spodziewaj się niepowodzeń

Ponieważ infrastruktura magazynowa opiera się na komponentach towarowych, należy spodziewać się awarii. W związku z tym systemy muszą mieć wdrożony model awarii, który może zapewnić ciągły postęp i akceptowalne czasy reakcji pomimo wszelkich awarii, które mogły mieć miejsce. Często te zestawy danych są replikowane, a poszczególne wycinki tych zestawów danych mają powiązane z nimi sumy kontrolne w celu wykrywania przeskoków bitów i towarzyszących im uszkodzeń danych, które często miały miejsce w zwykłym sprzęcie.

Temat 5: Dostrój dostęp według aplikacji

Chociaż te struktury pamięci masowej są zbudowane na istniejących systemach plików, przechowywane zestawy danych są przeznaczone do przetwarzania przez aplikacje, a nie przez ludzi. Ponieważ zbiór danych jest rozproszony na dużej liczbie komputerów, rekonstrukcja zbioru danych wymaga przetworzenia metadanych (danych opisujących dane) w celu zidentyfikowania dokładnej lokalizacji określonych części zbiorów danych. Ręczne uzyskiwanie dostępu do dowolnego węzła w celu wyszukania części zbioru danych jest daremne, ponieważ te części zostały zmodyfikowane tak, aby zawierały informacje o sumie kontrolnej.

Temat 6: Optymalizacja pod kątem dominującego użytkowania

Inną ważną kwestią w tych strukturach pamięci masowej jest optymalizacja najbardziej ogólnych wzorców dostępu do tych zestawów danych. W niektórych przypadkach oznaczałoby to optymalizację dla długich, sekwencyjnych odczytów, które kładą nacisk na oszczędzanie przepustowości, podczas gdy w innych wiązałoby się to z optymalizacją małych, ciągłych aktualizacji zarządzanych zestawów danych.

Temat 7: Kompromis między spójnością a dostępnością

Ponieważ te zestawy danych są rozproszone (i replikowane) na dużej liczbie maszyn, rozliczanie tych awarii wiąże się z kompromisem między spójnością a dostępnością. Większość z tych struktur pamięci masowej decyduje się na dostępność i opiera się na ostatecznej spójności. Wybór ten ma swoje korzenie w twierdzeniu CAP. W 2000 roku Brewer wysunął teorię, że usługa sieciowa nie może zapewnić pełnej gwarancji spójności, dostępności i tolerancji partycji . W 2002 roku Seth Gilbert i Nancy Lynch z MIT udowodnili to twierdzenie. Chociaż twierdzenie Brewera było nastawione na usługi internetowe, każdy rozproszony system plików może być postrzegany jako taki. W niektórych przypadkach, takich jak Amazon S3, łatwiej jest zobaczyć to połączenie niż inne. Zanim zagłębimy się głębiej, co rozumiemy przez spójność, dostępność i tolerancję na partycje? Posiadanie spójnego systemu rozproszonego oznacza, że bez względu na węzeł, z którym się łączysz, zawsze znajdziesz te same dokładne dane. W tym przypadku dostępność oznacza, że dopóki żądanie jest wysyłane do węzła, który nie zawiódł, zwróci wynik. Ta definicja nie ma ograniczeń czasowych, po prostu stwierdza, że w końcu klient otrzyma odpowiedź. Wreszcie istnieje tolerancja podziału. Partycja występuje, gdy jedna część systemu rozproszonego nie może już komunikować się z inną częścią, ale nadal może komunikować się z klientami. Najprostszym przykładem tego jest system z 2 węzłami, A i B. Jeśli A i B nie mogą już komunikować się ze sobą, ale oba mogą i nadal obsługują klientów, system jest odporny na partycje. W systemie odpornym na partycje nic poza całkowitą awarią systemu nie uniemożliwia poprawnego działania systemu. Jako szybki przykład spójrzmy na system odporny na partycje z dwoma węzłami A i B. Załóżmy, że między A i B wystąpił błąd sieci i nie mogą się już komunikować ze sobą, ale obaj mogą nadal łączyć się z klientami. Jeśli klient miałby zapisać zmianę w pliku v hostowanym zarówno na A, jak i B podczas połączenia z B, zmiana zostałaby wprowadzona na B, ale jeśli klient później połączy się z A i ponownie odczyta v, klient nie zobaczy swoich zmian , więc system nie jest już spójny. Można to obejść, poświęcając dostępność — jeśli zignorujesz zapisy podczas partycji sieciowej, możesz zachować spójność. W tej części dokonamy przeglądu ram przechowywania danych od trzech dominujących dostawców chmury obliczeniowej – Google, Amazon i Microsoft. Profilujemy każdą strukturę pamięci masowej według wymiarów, które obejmują między innymi replikację, model awarii, replikację i bezpieczeństwo. Nasz opis każdej struktury jest samowystarczalny, a czytelnik może przeglądać te struktury w dowolnej kolejności. Dla kompletności zamieściliśmy opis systemu xFS (opracowanego w połowie lat 90.), który badał pomysły, które teraz znalazły drogę do kilku omawianych przez nas systemów.

xFS

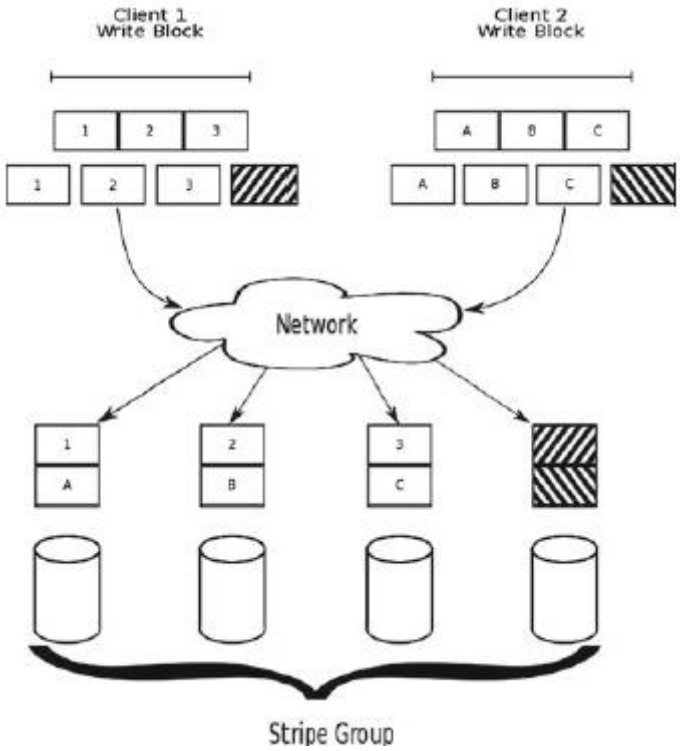
W przeciwieństwie do innych wymienionych tutaj systemów, xFS nigdy nie trafił do środowiska produkcyjnego. xFS to oryginalny „bezserwerowy system plików”, a kilka obecnie produkowanych systemów opiera się na pomysłach, które pojawiły się pierwotnie. xFS został zaprojektowany do pracy na standardowym sprzęcie i powinien obsługiwać duże obciążenia i wielu użytkowników. Opierając się na śledzeniu wzorców użytkowania w systemie NFS przez kilka dni, xFS zakłada, że użytkownicy inni niż twórca pliku rzadko modyfikują pliki.

Model awarii

W xFS, gdy maszyna ulegnie awarii, nie oczekuje się, że wróci do trybu online. W przypadku awarii maszyny dane są automatycznie tasowane, aby zrekompensować utratę. Chociaż zakłada się, że awarie mają charakter trwały, system został zaprojektowany tak, aby był w stanie powrócić po pełnej utracie funkcjonalności.

Replikacja

xFS nie obsługuje replikacji plików. Zamiast tego obsługuje podejście RAID do przechowywania danych, jak pokazano na rysunku. W xFS serwery są zorganizowane w grupy paskowe. Każda grupa rozłożona jest podzbiorem zestawu serwerów danych. Gdy klient zapisuje do pliku, jest on gromadzony w bloku zapisu, który jest przechowywany w pamięci podręcznej klienta. Na rysunku są dwaj klienci, z których każdy buduje własny blok zapisu.



Gdy blok zapisu jest pełny, dane są wysyłane do grupy serwerów w celu zapisania do pliku. W przypadku grupy serwerów z N serwerami plik jest podzielony na części N-1 i rozłożony we wzorze RAID na wszystkich serwerach. N-ty pasek to blok parzystości, który zawiera XOR wszystkich pozostałych elementów i jest pokazany jako blok w paski na ryc. 6.1. Ten blok parzystości zostanie przesłany do serwera parzystości dla grupy. W ten sposób, jeśli serwer zostanie utracony lub część ulegnie uszkodzeniu, można go przywrócić. Jedną z wad tego podejścia jest to, że jeśli wiele serwerów z grupy

ulegnie awarii, dane mogą zostać trwale utracone, a xFS przestanie działać. Ogólnie poziom replikacji pliku nigdy nie może być większy niż liczba serwerów w grupie serwerów.

Dostęp do danych

W xFS klient połączy się z menedżerem systemu, który wyszuka odpowiednią grupę serwerów i połączy klienta z liderem grupy serwerów. Ogólnie rzecz biorąc, zajmuje to około 3 przeskoków (nie licząc rzeczywistej transmisji danych). Ogólnie rzecz biorąc, system będzie próbował przenieść dane, aby były jak najbliżej użytkownika (w wielu przypadkach projekt oczekuje, że klient będzie działał na maszynie, która działa również jako serwer danych), ponosząc krótkoterminową karę w ruch sieciowy związany z przenoszeniem pliku dla długoterminowej premii braku konieczności dalszej interakcji z administratorem systemu.

Uczciwość

Z powodu zaplecza RAID xFS uszkodzenie danych można wykryć i naprawić za pomocą bloku parzystości obliczonego podczas zapisywania danych. xFS wykorzystuje te informacje również do odzyskiwania brakujących bloków rozłożonych, gdy serwer w grupie rozłożonych ulegnie awarii.

Spójność i Gwarancje

xFS gwarantuje spójność odczytu tego, co napisałeś, ale umożliwia także użytkownikom odczytywanie nieaktualnych danych - co oznacza, że ostateczna jest najlepsza ogólna spójność, jaką można osiągnąć. Nie jest również jasne, czy system może efektywnie obsługiwać współbieżne zapisy. xFS nigdy nie trafił do środowiska produkcyjnego, więc nigdy nie było silnej potrzeby ustanowienia jakichkolwiek gwarancji regulujących czas dostępu. Dodatkowo xFS został zaprojektowany do obsługi zmian w liczbie dostępnych serwerów.

Metadane

Główną zaletą xFS jest jego w pełni dynamiczna struktura. Chodzi o to, aby móc przenosić dane, aby poradzić sobie ze zmianami obciążenia i zwiększyć lokalność. System wykorzystuje informacje o metadanych, aby pomóc zlokalizować wszystkie pliki i uporządkować je.

Umieszczenie danych

Menedżerowie w xFS starają się zapewnić, że dane są przechowywane jak najbliżej klienta uzyskującego do nich dostęp - w niektórych przypadkach nawet zmiana lokalizacji danych, gdy klient zaczyna do nich zapisywać. Chociaż wydaje się to nieporęczne, xFS używa metody przechowywania opartej na dziennikach, więc nie ma zbyt dużego trafienia w sieć, ponieważ dane są przesuwane przy nowym zapisie bliżej obecnego klienta.

Bezpieczeństwo

xFS został zaprojektowany do uruchamiania w zaufanym środowisku i oczekuje się, że klienci działają na komputerach, które działają również jako serwery pamięci masowej. Możliwe jest jednak zamontowanie i dostęp do xFS z niebezpiecznego środowiska. Niestety jest to bardziej nieefektywne i skutkuje znacznie większym ruchem w sieci. Możliwe jest również, że nieuczciwy menedżer zacznie bezkrytycznie nadpisywać dane, co może spowodować awarię całego systemu.

Amazon S3

Simple Storage Service (S3) firmy Amazon jest używany przez użytkowników domowych, małe firmy, instytucje akademickie i duże przedsiębiorstwa. Dzięki S3 (Simple Storage Service) dane mogą być

rozłożone na wielu serwerach w Stanach Zjednoczonych i Europie (S3-Europe). S3 oferuje niskie opóźnienia, nieskończoną trwałość danych i dostępność na poziomie 99,99%.

Dostęp do danych i zarządzanie

S3 przechowuje dane na 2 poziomach: najwyższy poziom zasobników i obiektów danych. Zasobniki są podobne do folderów i mogą zawierać nieograniczoną liczbę obiektów danych. Każde konto Amazon Web Services (AWS) może mieć do 100 segmentów. Opłata za S3 jest obliczana na poziomie zasobnika. Wszystkie poziomy kosztów są wielopoziomowe, ale podstawowe koszty od stycznia 2010 r. są następujące: koszty przechowywania 0,15 USD/GB/miesiąc w USA, 0,165 USD w północnej Kalifornii i 0,15 USD/GB/miesiąc w Europie; 0,10 USD/GB za przesyłanie (bezpłatnie do lipca 2010 r.) i 0,17 USD/GB za pobieranie; oraz 0,01/1000 USD operacji PUT, COPY, POST lub LIST, 0,001/10 000 GET i wszystkich innych operacji. Każdy obiekt danych ma nazwę, obiekt blob danych (do 5 GB) i metadane. S3 narzuca mały zestaw wstępnie zdefiniowanych wpisów metadanych i pozwala na dodanie do tych metadanych do 4 KB wygenerowanych przez użytkownika par {nazwa, wartość}. Chociaż użytkownicy mogą tworzyć, modyfikować i usuwać obiekty w zasobniku, S3 nie obsługuje zmiany nazwy obiektów danych ani przenoszenia ich między zasobnikami — te operacje wymagają od użytkownika najpierw pobrania całego obiektu, a następnie zapisania całego obiektu z powrotem w S3. Funkcje wyszukiwania są również poważnie ograniczone w obecnej implementacji. Użytkownicy mogą wyszukiwać obiekty tylko według nazwy zasobnika — nie można przeszukiwać metadanych ani samego obiektu BLOB danych. Amazon S3 obsługuje trzy protokoły dostępu do danych: SOAP, REST i BitTorrent. Podczas gdy REST jest najczęściej używany do przesyłania dużych danych, BitTorrent może być bardzo przydatny do przesyłania dużych obiektów.

Bezpieczeństwo

Podczas gdy klienci używają schematu opartego na infrastrukturze klucza publicznego (PKI) do uwierzytelniania podczas wykonywania operacji z S3, klucze publiczne i prywatne użytkownika są generowane przez Amazon, a klucz prywatny jest dostępny za pośrednictwem witryny AWS użytkownika. Oznacza to, że skuteczne bezpieczeństwo zależy od hasła użytkownika AWS, które można zresetować za pośrednictwem poczty e-mail. Ponieważ konta S3 są połączone bezpośrednio z kartą kredytową, może to potencjalnie spowodować wiele problemów dla użytkownika. Kontrola dostępu jest określana za pomocą list kontroli dostępu (ACL) zarówno na poziomie zasobnika, jak i obiektu danych. Każda lista ACL może określać uprawnienia dostępu dla maksymalnie 100 tożsamości i obsługiwana jest tylko ograniczona liczba atrybutów kontroli dostępu: odczyt dla segmentów lub obiektów danych, zapis dla segmentów i wreszcie odczyt i zapis samej listy ACL. Użytkownik może skonfigurować zasobnik do przechowywania rekordów dziennika dostępu. Dzienniki te zawierają typ żądania, obiekt, do którego uzyskano dostęp, oraz czas przetworzenia żądania.

Uczciwość

Wewnętrzne działanie Amazon S3 nie zostało opublikowane. Trudno określić ich podejście do wykrywania i usuwania błędów. Na podstawie zgłoszonego użycia nie doszło do trwałej utraty danych.

Dynamo

Dynamo to zaplecze dla większości usług świadczonych przez Amazon. Podobnie jak S3 jest to system rozproszonej pamięci masowej. Dynamo przechowuje dane w parach klucz-wartość i poświęca spójność na rzecz dostępności. Dynamo został zaprojektowany do przechowywania stosunkowo małych plików (~1 MB) i bardzo szybko je odzyskać. Strona internetowa może mieć kilka usług, z których każda ma własną instancję Dynamo działającą w tle - to prowadzi do konieczności upewnienia

się, że opóźnienia podczas pobierania danych są niewielkie. Dynamo używa spójnego hashowania, aby stworzyć skalowalny system. Każdy plik w systemie identyfikowany przez klucz jest zaszyfrowany, a ta wartość skrótu jest używana do określenia, do którego węzła w systemie jest przypisany. Ta przestrzeń mieszająca jest traktowana jako pierścień, który jest podzielony na partycje o równej wielkości Q. Każdy węzeł (serwer) w systemie ma przypisaną taką samą liczbę partycji. Na tym rysunku jest łącznie 8 partycji. Węzły A, B i C są odpowiedzialne za przechowywanie kopii wszystkich plików, w których zaszyfrowany klucz znajduje się na partycji rozłożonej, którą zarządzają.

Punkt kontrolny

Węzły Dynamo udostępniają informacje za pośrednictwem protokołu opartego na plotkach. Nie ma regularnych pulsów przesyłanych między węzłami. Cała komunikacja jest popychana przez żądania klientów. Jeśli nie ma żądania danych, węzły nie komunikują się i nie dbają o to, czy inny węzeł jest wyłączony. Okresowe testy sprawdzające, czy węzeł jest dostępny, mają miejsce tylko wtedy, gdy podczas żądania klienta okaże się, że węzeł jest nieosiągalny.

Replikacja

W Dynamo system podobny do kworum jest używany do określenia, czy odczyt lub zapis zakończył się powodzeniem. Jeśli wystarczająca liczba węzłów odpowie, że zapis/odczyt się powiódł, cała operacja jest uznawana za udaną - nawet jeśli nie wszystkie N replik są zapisywane lub odczytywane. Dynamo pozwala autorowi usługi określić nie tylko N, ale także R i W. R to liczba udanych odczytów potrzebnych do powodzenia całej operacji, a W to liczba zapisów. Dynamo zgłosi udany zapis, jeśli węzły W-1 zgłoszą sukces, więc aby stworzyć system, który zawsze działa i nigdy nie odrzuci zapisu, W można ustawić na 1. Ogólnie, W i R są mniejsze niż N, więc że system może robić postępy w przypadku awarii. Sugerowana konfiguracja dla Dynamo to R + W > N. Ogólna konfiguracja (N,R,W) to (3,2,2).

Awarie

Dynamo działa z założeniem, że spodziewane są awarie sprzętu, i wymienia gwarancje spójności danych na dostępność. Wykorzystuje system oparty na plotkach do wykrywania awarii węzłów. Gdy węzeł przestanie odpowiadać, inne węzły w końcu przekażą wiedzę o awarii. Jako cecha projektowa, węzły nie są uważane za usunięte, chyba że administrator wyda wyraźne polecenie usunięcia - oznacza to, że system z łatwością poradzi sobie z przejściowymi przestojami. Jeśli koordynator nie może dotrzeć do węzła w celu zapisu, po prostu przekaże dane do następnego dostępnego węzła w pierścieniu mieszającym. Będzie on zawierał dodatkowy bit metadanych, który oznaczy go jako należący do innego miejsca. Gdy węzeł wróci do trybu online, informacje te mogą zostać do niego przekazane. Jeśli węzeł nie jest dostępny, dane, które przypuszczalnie znajdują się w tym węźle, nie są natychmiast replikowane w innym węźle – dzieje się tak tylko wtedy, gdy administrator jawnie usuwa węzeł za pomocą polecenia. Dynamo jest budowane w oparciu o oczekiwanie, że wystąpi wiele przejściowych awarii, więc nie ma żadnych problemów, aby zapewnić spełnienie poziomów replikacji, gdy węzeł przestanie odpowiadać na żądania. Z tego powodu niektóre odczyty mogą się nie powieść, jeśli R jest ustawione na N. Po jawnym usunięciu węzła wszystkie zakresy kluczy poprzednio posiadane przez ten węzeł są ponownie przypisywane do innych węzłów, zapewniając jednocześnie, że dany węzeł nie jest przeciążony w wyniku ta redystrybucja.

Dostęp do danych

Oparty na plotkach protokół Dynamo do wykrywania węzłów zapewnia, że wszystkie węzły w jednym kroku znają dokładny węzeł, do którego mają wysłać żądanie odczytu lub zapisu. Istnieją dwie główne metody dostępu do danych: (1) użycie dedykowanego węzła do obsługi żądań klientów lub (2)

posiadanie kilku dedykowanych węzłów lub koordynatorów, które przetwarzają żądania klientów i przekazują je do odpowiednich węzłów. Pierwsze podejście może prowadzić do niezrównoważonych węzłów sieci, podczas gdy drugie podejście skutkuje bardziej zrównoważoną siecią i można zapewnić mniejsze opóźnienia.

Integralność danych

Nie ma konkretnej wzmianki o wykrywaniu uszkodzeń w danych ani o tym, jak może wystąpić odpowiednie odzyskiwanie po błędzie. Ponieważ dane są przechowywane jako obiekt binarny, twórcom aplikacji można pozostawić wykrycie uszkodzenia danych i obsługę wszelkiego rodzaju odzyskiwania. Zgłaszane wyniki w ustawieniach na żywo nie wskazują na trwałą utratę danych. Amazon wymaga regularnej archiwizacji każdego systemu - istnieje szansa, że te archiwalne dane zostaną użyte do odzyskania, jeśli zostaną znalezione błędy w danych

Spójność i Gwarancje

Dynamo gwarantuje ostateczną spójność - istnieje szansa, że nie wszystkie replikacje zawierają te same dane. Z powodu przejściowych awarii sieci i współbieżnych zapisów niektóre zmiany mogą nie być w pełni propagowane. Aby rozwiązać ten problem, każdy obiekt zawiera również kontekst. Ten kontekst zawiera wektor wersji, dający możliwość śledzenia zmian i ustalenia, która wersja obiektu powinna mieć największy priorytet. Istnieje kilka różnych schematów radzenia sobie z tym. Samo Dynamo obsługuje kilka prostych schematów, w tym metodę last-write-wins. Dostępny jest również interfejs, który umożliwia programistom wdrażanie bardziej złożonych i specyficznych dla danych technik łączenia. Scalanie różnych wersji obiektów jest obsługiwane podczas odczytów. Jeśli koordynator pobiera wiele wersji obiektu podczas odczytu, może próbować scalić różnice przed wysłaniem go do klienta. Wszystko, czego koordynator nie może rozwiązać, jest przekazywane klientowi. Zakłada się, że każdy kolejny zapis od tego klienta rozwiązał wszystkie pozostałe konflikty. Koordynator upewnia się, że rozwiązany obiekt zapisze z powrotem do wszystkich węzłów, które odpowiedziały na zapytanie o obiekt. Jedyną inną podstawową gwarancją zapewnianą przez Dynamo jest wydajność ukierunkowana na 99,99. percentyl użytkowników - gwarantowane są milisekundowe opóźnienia. Poza tym twórcy usług mogą dostosowywać system, aby dopasować gwarancje niezbędne dla ich aplikacji poprzez ustawienia N, R i W.

Metadane

W Dynamo metadane obiektu nazywane są kontekstem. Za każdym razem, gdy zapisywane są dane, dołączany jest kontekst. Kontekst zawiera metadane systemowe i inne informacje specyficzne dla obiektu, takie jak informacje o wersji. Może również istnieć dodatkowe pole binarne, które umożliwia programistom dodawanie wszelkich dodatkowych informacji potrzebnych do działania ich aplikacji. Metadanych nie można przeszukiwać i wydaje się, że wchodzą w interakcję z Dynamo tylko podczas rozwiązywania konfliktów wersji, jak wspomniano powyżej.

Rozmieszczenie danych

Według DeCandii i innych istnieją gwarancje zapewniające rozmieszczenie replik w różnych centrach danych. Prawdopodobnie Amazon ma szczególny schemat, który pozwala Dynamo na sprawne określanie lokalizacji węzłów. Klucz obiektu jest najpierw haszowany, aby znaleźć jego lokalizację w pierścieniu sieci. Poruszając się po pierścieniu zgodnie z ruchem wskazówek zegara od tego punktu, pierwszy napotkany węzeł to miejsce, w którym umieszczana jest pierwsza kopia danych. Następne węzły N-1 (wciąż poruszające się zgodnie z ruchem wskazówek zegara) będą zawierały repliki danych. W Dynamo nie ma obecnie metod segregacji danych - jest po prostu interfejs get() i put() dla

programistów, nie ma wsparcia dla struktury hierarchicznej. Każda usługa korzystająca z Dynamo ma swoją indywidualną instancję. Na przykład Twój koszyk nie będzie mógł uzyskać dostępu do listy bestsellerów. Z drugiej strony Dynamo nie ma gwarancji, że różne instancje nie działają na tym samym komputerze.

Bezpieczeństwo

Dynamo zostało zaprojektowane do pracy w zaufanym środowisku, więc nie ma struktury, która mogłaby poradzić sobie z kwestiami bezpieczeństwa. Zgodnie z projektem, każda usługa korzystająca z Dynamo ma własną uruchomioną instancję. Z tego powodu użytkownicy mają pewne poczucie bezpieczeństwa, ponieważ istnieje naturalna separacja danych, a jedna aplikacja nie może uzyskać dostępu do danych innej.

System plików Google

System plików Google (GFS) został zaprojektowany przez Google jako zaplecze dla wszystkich systemów Google. Podstawowym założeniem leżącym u podstaw jego projektu jest to, że komponenty mogą ulec awarii. Potrzebny jest solidny system do wykrywania i obchodzenia tych awarii bez zakłócania obsługi plików. GFS jest zoptymalizowany pod kątem najczęstszych operacji - długich, sekwencyjnych i krótkich, losowych odczytów, a także dużych, dołączających i małych, dowolnych zapisów. Ponadto głównym celem przy projektowaniu GFS było wydajne umożliwienie współbieżnego dołączania do tego samego pliku. Jako cel projektowy uznano, że wysoka trwała przepustowość jest ważniejsza niż niskie opóźnienia w celu obsługi dużych zestawów danych. Instancja GFS zawiera serwer główny i wiele serwerów porcjowych. Serwer główny jest odpowiedzialny za utrzymanie wszystkich metadanych systemu plików i zarządzanie porcjami (zapisanymi fragmentami plików). Zwykle istnieje również kilka replik głównych, a także wzorców cienia, które mogą obsługiwać odczyty klienta, aby zmniejszyć obciążenie serwera głównego. Serwery porcji przechowują dane w porcjach o rozmiarze 64 MB.

Punkt kontrolny

W GFS serwer główny będzie prowadzić dzienniki śledzące wszystkie mutacje porcji. Gdy plik dziennika staje się zbyt duży, serwer główny utworzy punkt kontrolny. Te punkty kontrolne mogą służyć do odzyskiwania serwera głównego i są używane przez repliki główne do uruchomienia nowego procesu głównego.

Replikacja

Domyślnie wszystkie GFS utrzymują poziom replikacji 3. Jest to jednak cecha konfigurowalna: „users może wyznaczyć różne poziomy replikacji dla różnych regionów przestrzeni nazw plików”. Na przykład katalog tymczasowy ma zazwyczaj poziom replikacji 1 i jest używany jako przestrzeń magazynowa. Serwer główny jest odpowiedzialny za zapewnienie spełnienia poziomu replikacji. Obejmuje to nie tylko kopiowanie fragmentów w przypadku awarii serwera fragmentów, ale także usuwanie replik po ponownym uruchomieniu serwera. Z reguły serwer główny będzie próbował umieszczać repliki na różnych stojakach. Dzięki konfiguracji sieci Google urządzenie główne może wywnioskować topologię sieci na podstawie adresów IP.

Awarie

Jeśli chodzi o awarie, GFS zawsze oczekuje najgorszego. Serwer główny regularnie wymienia pulsy z serwerami porcji. Jeśli serwer główny nie odbierze na czas pulsu z serwera porcji, założy, że serwer nie działa i natychmiast rozpocznie rozpowszechnianie porcji znajdujących się na tym serwerze na inne

serwery w celu przywrócenia poziomów replikacji. Jeśli serwer porcji powróci do stanu normalnego, zacznie ponownie wysyłać pulsy i powiadomi mastera, że jest w kopii zapasowej. W tym momencie serwer główny będzie musiał usunąć porcje, aby powrócić do poziomu replikacji i nie marnować miejsca. Dzięki takiemu podejściu możliwe byłoby sianie spustoszenia w instancji GFS poprzez wielokrotne włączanie i wyłączanie serwera chunk. Awaria serwera głównego jest wykrywana przez zewnętrzny system zarządzania. Gdy to nastąpi, jedna z replik serwera głównego jest promowana i uruchamiany jest na niej proces serwera głównego. Pełne ponowne uruchomienie zwykle zajmuje około 2 minut – większość tego czasu poświęca się na odpytywanie serwerów porcji, aby dowiedzieć się, jakie porcje zawierają

Dostęp do danych

Klienci początkowo kontaktują się z serwerem głównym, aby uzyskać dostęp do pliku, po czym klient wchodzi w interakcję bezpośrednio z niezbędnymi serwerami porcji. W przypadku pliku wieloterabajtowego klient może śledzić wszystkie serwery porcji w swojej pamięci podręcznej. Serwer porcji bezpośrednio komunikujący się z klientami otrzymuje dzierżawę porcji od serwera głównego i jest teraz nazywany podstawowym. Główny jest wtedy odpowiedzialny za szeregowe porządkowanie wszelkich operacji na danych. Następnie jest odpowiedzialny za propagację tych zmian do innych serwerów porcji, które przechowują porcję. Jeśli klient chce tylko odczytać dane, możliwe jest, że będzie przechodził przez serwer cienia, a nie przez serwer główny. Możliwe jest, że współbieżne zapisy będą się przeplatać w nieoczekiwany sposób lub jeśli nieudane próby zapisu będą wyświetlane jako powtarzające się dane w porcjach. GFS zakłada, że każda używająca go aplikacja jest w stanie poradzić sobie z tymi możliwymi problemami, chociaż nadmiarowe dane mogą obniżyć wydajność odczytów.

Integralność danych

Każda porcja w GFS śledzi własne informacje o sumie kontrolnej. Informacje te są unikalne dla każdej porcji — nie ma gwarancji, że będą takie same, nawet we wszystkich replikach. Serwery porcji są odpowiedzialne za sprawdzanie sum kontrolnych przechowywanych przez nie porcji. Dzięki temu system może wykryć uszkodzone pliki. W przypadku wykrycia uszkodzonej porcji porcja jest usuwana i kopiowana z innej repliki.

Spójność i Gwarancje

GFS jest zbudowany do obsługi wielu równoczesnych dołączeń w jednym pliku. Zadaniem podstawowego serwera porcji jest uporządkowanie przychodzących żądań permutacji od wielu klientów w kolejności sekwencyjnej, a następnie przekazanie tych zmian do wszystkich pozostałych replik. Z tego powodu możliwe jest, że klient nie zobaczy dokładnie tego, co napisał podczas odczytu sekwencyjnego - istnieje możliwość, że permutacje od innych klientów zostały przeplatane własnymi. Google opisuje ten stan jako spójny, ale niezdefiniowany – wszyscy klienci zobaczą te same dane, niezależnie od tego, która replika jest podstawowa, ale mutacje mogą być przeplatane. W przypadku błędu zapisu fragment może stać się niespójny. Jest to przypadek, w którym w niektórych, ale nie we wszystkich replikach, mogą występować nadmiarowe linie danych. Ponieważ GFS został zbudowany w celu utrzymania przepustowości, w przeciwieństwie do osiągnięcia celu docelowego opóźnienia, nie ma gwarancji dotyczących opóźnień. GFS gwarantuje utrzymanie określonego poziomu replikacji, który jest osiągany za pomocą pulsów systemowych. GFS nie może również zagwarantować pełnej spójności w obliczu błędów zapisu. Nieco luźniejsza definicja spójności – przynajmniej jedna kopia wszystkich danych jest w pełni przechowywana w każdej replice – jest tym, co dostarcza GFS. Każda aplikacja zbudowana na bazie GFS, która może obsłużyć te możliwe niespójności, powinna być w stanie zagwarantować większą spójność.

Metadane

W GFS serwer główny zawiera metadane dotyczące wszystkich porcji zawartych w systemie. W ten sposób serwer główny śledzi, gdzie znajdują się porcje. Każda porcja ma również swój własny zestaw metadanych. Fragment ma numer wersji, a także własną informację o sumie kontrolnej.

Rozmieszczenie danych

Serwer główny próbuje umieścić repliki na osobnych stojakach, co jest możliwe dzięki schematowi sieciowemu Google. Serwer główny próbuje również zrównoważyć obciążenie sieci, więc spróbuje równomiernie rozproszyć wszystkie porcje.

Schemat bezpieczeństwa

GFS oczekuje, że będzie działać w zaufanym środowisku i nie ma żadnych znaczących podejść do bezpieczeństwa. Gdyby użytkownik mógł wyłączyć serwer porcji, zmodyfikować przechowywane na nim wersje porcji i ponownie podłączyć go do systemu, GFS powoli zatrzymałby się, ponieważ uważa, że ten serwer ma najbardziej aktualne porcje i zaczyna usuwać i przepisując wszystkie te kawałki. Spowodowałoby to duży ruch w sieci i teoretycznie obniżyłoby nie tylko każdą usługę, która opiera się na GFS, ale także wszystko inne, co wymaga przepustowości sieci do działania.

Bigtable

Jak sama nazwa wskazuje, Bigtable przechowuje duże ilości danych w tabeli. Chociaż nie jest to pełny model relacyjny, jest to zasadniczo wielowymiarowa baza danych. Tabele są indeksowane kluczami wierszy i kolumn (ciągi), a także znacznikiem czasu (int64). Wartości wewnątrz komórek są niezinterpretowaną tablicą bajtów, a tabele mogą być łatwo używane jako dane wejściowe lub wyjściowe MapReduce. Każda tabela jest podzielona na rzędy na tabletki. Każdy tablet będzie zawierał sekcję kolejnych wierszy, zwykle o wielkości około 100–200 MB. Bigtable został zaprojektowany przez Google do obsługi bardzo dużych plików, ogólnie mieszczących się w zakresie petabajtów. Jest używany w kilku produktach, w tym w Google Analytics i Google Earth. Bigtable został zaprojektowany do działania w systemie plików Google (GFS) i dziedziczy jego funkcje i ograniczenia. Wąskie gardła związane z GFS bezpośrednio wpływają na wydajność Bigtable, dlatego podjęto środki, aby uniknąć nadmiernego zwiększania ruchu sieciowego. Dodatkowo Bigtable opiera się na Chubby w zakresie podstawowej funkcjonalności. Chubby to usługa blokowania, która implementuje twierdzenie Lamporta Paxosa używane w Google, aby pomóc klientom dzielić się informacjami o stanie ich środowiska. Różne systemy wykorzystują Chubby do synchronizacji poszczególnych komponentów. Jeśli Chubby upadnie, to samo zrobi Bigtable. Biorąc to pod uwagę, Chubby był odpowiedzialny za mniej niż 0,001% przestojów Bigtable, jak donosi Chang procesy Bigtable zwykle działają na serwerach GFS, a inne procesy Google działają obok siebie. Zapewnienie małego opóźnienia w tym środowisku jest wyzwaniem. Implementacja Bigtable składa się z 3 elementów: Po pierwsze, z każdym klientem jest powiązana biblioteka, która pomaga klientom znaleźć właściwy serwer podczas wyszukiwania danych. Po drugie, istnieje jeden serwer główny. Ten serwer główny zazwyczaj nie wchodzi w interakcje z klientami, przez co jest zwykle tylko lekko obciążony. Wreszcie istnieje wiele serwerów tabletów. Serwery tabletów odpowiadają za komunikację z klientami i niekoniecznie obsługują kolejne tablety; po prostu to, co jest potrzebne. Każdy tablet jest obsługiwany tylko na jednym serwerze tabletów naraz. Nie jest również konieczne, aby wszystkie tablety były obsługiwane — master przechowuje listę tabletów, które nie są obecnie obsługiwane, i przypisze te tablety do serwera, jeśli klient zażąda do niego dostępu. Tablety są przechowywane w GFS, podobnie jak w formacie SSTable, a na tablecie jest zazwyczaj kilka SSTables. SSTable zawiera zestaw par klucz/wartość, gdzie zarówno klucz, jak i wartość są dowolnymi ciągami. Aktualizacje tabletów są przechowywane w dzienniku zatwierdzenia. Ostatnio

wprowadzone zmiany są przechowywane w pamięci, a starsze rekordy aktualizacji tabletu są przechowywane jako SSTable. Zarówno dzienniki zatwierdzenia, jak i pliki SSTable są przechowywane w GFS. Przechowywanie plików zatwierdzenia w GFS oznacza, że wszystkie zatwierdzenia można odzyskać w przypadku śmierci serwera tabletu. Te dzienniki zmian są tak blisko, jak Bigtable zbliża się do rzeczywistego punktu kontrolnego - dokładniejsze sprawdzanie jest przeprowadzane przez GFS.

Replikacja

Jak wspomniano powyżej, serwer główny Bigtable zapewnia, że tylko jeden serwer faktycznie modyfikuje tablet naraz. Chociaż wygląda na to, że Bigtable całkowicie ignoruje replikację, pliki SSTable każdego tabletu są w rzeczywistości przechowywane w GFS. Bigtable zgrabnie omija problem replikacji i pozwala GFS go obsłużyć. Bigtable odziedziczy poziom replikacji folderów, w których przechowywane są pliki SSTables.

Awarie

Wszystkie wykrywane awarii dla Bigtable ostatecznie sprowadza się do Chubby. Gdy serwer tabletu uruchamia się po raz pierwszy, kontaktuje się z Chubbym i tworzy plik specyficzny dla serwera, po czym uzyskuje na nim blokadę na wyłączność. Ta blokada jest aktywna tak długo, jak tablet ma połączenie z Chubbym i natychmiast przestanie podawać tabletki, jeśli zgubi tę blokadę. Jeśli serwer tabletów kiedykolwiek skontaktuje się z Chubbym i stwierdzi, że plik zniknął, sam się zabije. Serwer główny jest odpowiedzialny za okresowe odpytywanie serwerów tabletów i sprawdzanie, czy nadal działają. Jeśli master nie może skontaktować się z serwerem tabletu, najpierw sprawdza, czy serwer tabletu może nadal komunikować się z Chubby. Czyni to, próbując uzyskać blokadę na wyłączność w pliku serwera tabletu. Jeśli mistrz zdobędzie zamek, Chubby żyje, a tablet nie może się z nim komunikować. Następnie master usuwa plik serwera, zapewniając, że serwer nie będzie próbował ponownie udostępniać. Jeśli sesja Chubby mistrza wygaśnie, mistrz natychmiast zabija się bez wpływu na podawanie tabletek. System zarządzania klastrem działający razem z Bigtable jest odpowiedzialny za uruchomienie nowego serwera głównego, jeśli tak się stanie. Chociaż nie określa wyraźnie, co się stanie, jeśli Chubby przestanie działać, prawdopodobnie bieżący serwer główny sam się zabije, a menedżer klastra będzie wielokrotnie próbował uruchomić nowy serwer główny, dopóki Chubby nie zacznie ponownie odpowiadać.

Dostęp do danych

Do każdego klienta wysyłana jest początkowo biblioteka lokalizacji tabletów, więc początkowo powinni mieć możliwość bezpośredniego kontaktu z właściwym serwerem tabletów. Z biegiem czasu serwery tabletów umierają, niektóre mogą zostać dodane lub tablety mogą zostać usunięte lub podzielone. Bigtable ma trzypoziomową hierarchię lokalizacji tabletu. Po pierwsze, w Chubby znajduje się plik, który zawiera lokalizację tabletu głównego. Każda instancja Bigtable ma swój własny tablet główny. Tablica główna określa lokalizację wszystkich tabletów w tabeli METADATA. Ta tabela METADATA zawiera lokalizacje wszystkich tabel użytkowników, a także niektóre informacje dotyczące tabletu przydatne do celów debugowania. Tablet root to po prostu pierwsza tablet tabeli METADATA. Tablet główny jest traktowany specjalnie - nigdy nie jest dzielony, aby hierarchia lokalizacji tabletu nie rosła. Za pomocą tego schematu można zaadresować 234 lokalizacje tabletów. Biblioteka klienta buforuje lokalizacje tabletów z tabeli METADATA i rekursywnie śledzi hierarchię, jeśli nie ma tabletu lub lokalizacja tabletu jest skalowana. Z pustym cache zajmie to 3 podróże w obie strony, ale może zająć do 6 z przestarzałym cache. Żadna z tych operacji nie wymaga odczytu z GFS, więc czas jest znikomy. Serwery tabletów mają dostęp do posortowanych tablic SSTable, więc zazwyczaj mogą lokalizować wymagane dane (jeśli nie znajdują się już w pamięci) za pomocą jednego dostępu do dysku.

Integralność danych

Bigtable nie jest bezpośrednio zaangażowany w utrzymanie integralności danych. Wszystkie dane Bigtable są przechowywane w GFS i to jest odpowiedzialne za wykrywanie i naprawianie wszelkich błędów występujących w danych. Gdy serwer tabletów ulegnie awarii, istnieje ryzyko, że modyfikacja tabeli nie została zatwierdzona lub podział tabletu nie został prawidłowo przekazany z powrotem do Chubby. Przechowywanie wszystkich dzienników operacji tabletu w GFS również rozwiązuje pierwszy problem: nowy serwer tabletu może odczytywać dzienniki i zapewniać, że wszystkie tablety są aktualne. Podziały tabletów są jeszcze mniejszym problemem, ponieważ serwer tabletów zgłasza wszystkie posiadane tablety, do których Chubby się nie odwołuje.

Spójność i Gwarancje

Bigtable gwarantuje ostateczną spójność - wszystkie repliki są ostatecznie zsynchronizowane. Serwery tabletów przechowują w pamięci wszelkie modyfikacje tabletów i zapisują permutacje w dzienniku, ale niekoniecznie czekają, aż GFS potwierdzi, że zapis się powiódł przed potwierdzeniem tego przez użytkowników. Pomaga to skrócić opóźnienia i zapewnić użytkownikom bardziej interaktywne wrażenia, na przykład podczas korzystania z programu Google Earth. Bigtable dziedziczy wszystkie gwarancje GFS dotyczące replikacji danych, odzyskiwania błędów i umieszczania danych.

Metadane

Tabela METADATA zawiera metadane dla wszystkich tabletów przechowywanych w instancji Dynamo. Te metadane obejmują listy SSTables, które tworzą tablet, oraz zestaw wskaźników do zatwierdzania dzienników dla tabletu. Gdy serwer tabletu zaczyna udostępniać plik, najpierw odczytuje metadane tabletu, aby dowiedzieć się, które pliki SSTable należy załadować. Po załadowaniu SSTables do pamięci, działa poprzez dzienniki zatwierdzenia i przenosi wersję w pamięci do punktu, w którym była podczas ostatniego dostępu do tabletu.

Rozmieszczenie danych

Całe umieszczanie danych w Bigtable jest obsługiwane przez GFS – nie ma on bezpośredniego wpływu na umieszczanie danych. Jeśli chodzi o Bigtable, istnieją tylko pojedyncze kopie plików – używa uchwytów GFS, aby uzyskać dostęp do potrzebnych plików. Chociaż Bigtable nie jest bezpośrednio świadomy istnienia wielu wersji plików, nadal może korzystać z replik za pośrednictwem GFS.

Bezpieczeństwo

Bigtable został zaprojektowany do działania w zaufanym środowisku i tak naprawdę nie ma zbyt wielu środków bezpieczeństwa. Teoretycznie użytkownik może mieć zaszyfrowane nazwy wierszy i kolumn, a także dane w polach. Byłoby to możliwe, ponieważ są to wszystkie arbitralne ciągi. Podczas gdy szyfrowanie nazw wierszy oznacza, że możesz potencjalnie użyć niektórych możliwości grupowania, nie ma powodu, dla którego użytkownik nie byłby w stanie uzyskać pewnego bezpieczeństwa za pomocą tej metody.

Microsoft Azure

Azure to rozwiązanie do przetwarzania w chmurze firmy Microsoft. Składa się z trzech części: pamięci masowej, skalowalnego przetwarzania i podstawowej struktury, która utrzymuje wszystko razem w heterogenicznej sieci. Zarówno poziom przetwarzania, jak i przechowywania opierają się na warstwie sieci szkieletowej, która działa na wielu maszynach. Skalowalny komponent obliczeniowy platformy Azure jest poza zakresem tego artykułu, ale ze względu na kompletność wspomniano o nim tutaj. Rozwiązanie komputerowe firmy Microsoft zostało zaprojektowane tak, aby upewnić się, że działa

dobrze z pamięcią masową, ale nie jest konieczne używanie jednego do korzystania z drugiego. Microsoft nie opublikował zbyt wielu szczegółów dotyczących Azure. Usługa magazynu Azure pozwala użytkownikowi wybrać jeden z trzech formatów przechowywania: BLOB, tabel i kolejek. Obiekty BLOB to zasadniczo kontenery, które mogą pomieścić do 5 GB danych binarnych. Format BLOB Azure jest bardzo podobny do S3 — istnieją kontenery do przechowywania BLOBów i nie ma hierarchicznej obsługi (nie można umieścić kontenera wewnątrz kontenera). Nazwy BLOB nie mają jednak żadnych ograniczeń, więc nic nie stoi na przeszkodzie, aby użytkownik wstawił „/" w nazwie BLOB, aby pomóc w organizacji danych. Tabele na platformie Azure nie są prawdziwymi tabelami relacyjnymi, ale bardziej podobnymi do Bigtable — tabele zawierają jednostki, a jednostka jest listą nazwanych wartości. Chociaż tracisz możliwość wykonywania zapytań dotyczących tabel platformy Azure, tak jak w przypadku prawdziwej relacyjnej bazy danych, istnieje możliwość skutecznego skalowania na wielu komputerach. Kolejki Azures są przeznaczone głównie do użytku z usługą obliczeniową. Kolejki umożliwiają różnym aplikacjom uruchomionym przez użytkownika komunikowanie się ze sobą. Na przykład, użytkownik może zaprojektować internetową aplikację typu front-end, która może komunikować się z kilkoma aplikacjami roboczymi w celu wykonywania przetwarzania zaplecza. Ten pakiet aplikacji używałby kolejek do wymiany informacji między interfejsem internetowym a różnymi pracownikami.

Replikacja

Niezależnie od typu magazynu wszystkie dane mają poziom replikacji 3, którego utrzymanie jest koordynowane przez samą usługę przechowywania. Według Chappell , usługa sieci szkieletowej nie jest nawet świadoma poziomów replikacji, po prostu postrzega usługę pamięci masowej jako inną aplikację. Więcej o tym, jak to się dzieje, znajduje się w sekcji awarii.

Awaria

Warstwa tkaniny Azure składa się z maszyn w Microsoft Data Center. Centrum danych podzielone jest na domeny błędów. Microsoft definiuje domenę błędów jako zestaw maszyn, które mogą zostać zniszczone przez awarię pojedynczego elementu sprzętu. Wszystkie maszyny dedykowane dla Azure są sterowane przez 5–7 kontrolerów Fabric. Każda maszyna wewnątrz sieci szkieletowej ma uruchomiony proces kontrolera sieci szkieletowej, który raportuje stan wszystkich aplikacji uruchomionych na tej maszynie (obejmuje to aplikacje użytkownika na różnych maszynach wirtualnych oraz usługę magazynu). Chociaż nie wiemy dokładnie, w jaki sposób pamięć masowa jest obsługiwana w sieci szkieletowej, wiemy, że kontrolery sieci szkieletowej widzą usługę pamięci masowej jako kolejną aplikację. Jeśli aplikacja z jakiegoś powodu umrze, kontrolerzy są odpowiedzialni za uruchomienie kolejnej instancji aplikacji. Jest zrozumiałe, że jeśli instancja usługi pamięci masowej uruchomiona na maszynie umrze lub jeśli sama maszyna umrze, te kontrolery uruchomią inną instancję na innej maszynie. Dzięki warstwie sieci szkieletowej, która zapewnia rozproszenie aplikacji w domenach błędów, gwarantuje to rozproszenie replik.

Dostęp do danych

Jeśli użytkownik korzysta z aplikacji .NET działającej w usłudze obliczeniowej platformy Azure, można użyć interfejsów ADO .NET. Z drugiej strony, jeśli użytkownik próbuje uzyskać dostęp do danych w usłudze Azure Storage za pośrednictwem aplikacji Java, użyjesz standardowego REST. Jako przykład dostępu do obiektu BLOB z:

http://<StorageAccount>.blob.core.windows.net/<Container>/<BlobName>

Gdzie <StorageAccount> to identyfikator przypisany podczas tworzenia nowego konta magazynu, używany do identyfikowania własności obiektów. <Container> i <BlobName> to nazwy kontenera i obiektu BLOB, do których uzyskuje dostęp to żądanie. Nie ma konkretnej wzmianki o jakichkolwiek gwarancjach dotyczących opóźnień, ale ponieważ oczekuje się, że będzie to część aplikacji internetowej, prawdopodobnie jest ona niska.

Spójność i Gwarancje

Magazyn platformy Azure gwarantuje spójność odczytu i zapisu — wątki robocze i klienci będą mogli natychmiast zobaczyć zmiany, które właśnie napisał. Niestety nie ma jasnego obrazu tego, co to oznacza dla innych wątków/klientów. Gwarantuje również poziom replikacji 3 dla wszystkich przechowywanych danych. Nie ma również żadnych konkretnych gwarancji co do opóźnień ani konkretnej wzmianki o umowach SLA.

Rozmieszczenie danych

Za umieszczanie danych odpowiada warstwa sieci szkieletowej platformy Azure. Chociaż nie jest bezpośrednio świadomy replik, jest w stanie zapewnić, że wystąpienia usługi magazynu działają w różnych domenach błędów. Z dokumentów udostępnionych przez Microsoft wynika, że kontroler sieci szkieletowej działa tylko w jednym centrum danych. Istnieje szansa, że użytkownicy będą mogli wybrać, z którego centrum danych korzystać.

Bezpieczeństwo

Cały dostęp do składnika magazynu platformy Azure jest obsługiwany przez klucz prywatny wygenerowany przez platformę Azure dla określonego użytkownika. Chociaż nie ma szczegółowych informacji na temat tego, jak to się dzieje, prawdopodobnie jest to podatne na te same problemy, co S3 - inna osoba może być w stanie przejąć ten klucz. W usłudze Azure Storage nie ma list ACL, tylko ten jeden klucz dostępu - od deweloperów oczekuje się, że zapewnią własny program uwierzytelniania.

Debata transakcyjna i analityczna

Żaden z omawianych tutaj systemów pamięci masowej nie jest w stanie obsłużyć złożonych informacji relacyjnych. Ponieważ przechowywanie danych przenosi się do chmury, gdzie pozostają bazy danych? Posiadanie instalacji do zarządzania danymi na miejscu może być bardzo trudne w utrzymaniu, wymagając konserwacji administracyjnych i sprzętu, a także początkowych kosztów sprzętu i oprogramowania. Możliwość przeniesienia tych aplikacji do chmury pozwoliłaby firmom bardziej skoncentrować się na tym, co faktycznie produkują – być może przynosząc takie same skutki, jak sieć energetyczna 100 lat temu. Zarządzanie danymi transakcyjnymi jest tym, o czym zazwyczaj myśli się jako pierwsze — podstawą banków, linii lotniczych i witryn handlu elektronicznego. Systemy transakcyjne zazwyczaj mają dużo zapisów, a pliki zwykle mieszczą się w zakresie GB. Zwykle potrzebują gwarancji ACID, a tym samym mają problemy z dostosowaniem się do ograniczeń twierdzenia Brewera CAP. Systemy transakcyjne zazwyczaj zawierają również dane, które muszą być bezpieczne, takie jak numery kart kredytowych i inne prywatne informacje. Z tych powodów trudno jest przenieść system transakcyjny do chmury. Podczas gdy kilka baz danych firmy, takie jak Oracle, mają wersje, które mogą działać w środowisku rozproszonym, takim jak chmura EC2 firmy Amazon, problemem może stać się licencjonowanie. Zamiast potrzebować tylko jednej licencji, obecna implementacja wymaga osobnej licencji dla każdej instancji maszyny wirtualnej: wraz ze skalowaniem aplikacji może to stać się zbyt kosztowne. Zarządzanie danymi analitycznymi jest nieco inne. W systemie analitycznym jest na ogół więcej odczytów niż zapisów, a zapisy występują w dużych partiach. Tego typu systemy są używane do analizowania dużych zbiorów danych w poszukiwaniu wzorców lub

trendów. Pliki w systemie analitycznym mają też zupełnie inną skalę – klient może potrzebować przesiewać petabajty danych. W przypadku tego typu systemu dopuszczalna jest luźniejsza spójność ostateczna - co czyni go dobrym rozwiązaniem dla przetwarzania rozproszonego. Ponadto analizowane dane zwykle mają mniejszą potrzebę zapewnienia bezpieczeństwa, więc akceptacja hostingu danych przez stronę trzecią, taką jak Amazon lub Google.

Wnioski

Przeanalizowaliśmy kilka podejść do przechowywania danych w ustawieniach przetwarzania w chmurze. Centra danych mają i będą nadal budowane z komponentów towarowych. Stosowanie komponentów towarowych w połączeniu z kwestiami związanymi z ustawieniami, w których te komponenty działają, takimi jak rozpraszanie ciepła i zaplanowane przestoje, powodują, że awarie są częstym zjawiskiem i należy je tak traktować. W takich środowiskach nie jest już kwestią tego, czy system lub komponent ulegnie awarii, ale po prostu kiedy. Zbiory danych są rozproszone na zestawie maszyn, aby poradzić sobie z ich obszerną naturą i umożliwić jednoczesne przetwarzanie na nich. Aby poradzić sobie z awariami, każdy wycinek zestawu danych jest replikowany określoną liczbę razy; Replikacja umożliwia aplikacjom podtrzymywanie awarii maszyn, które przechowują określone fragmenty zestawu danych, a także inicjowanie korekcji błędów spowodowanych uszkodzeniem danych. Europejska Agencja ds. Bezpieczeństwa Sieci i Informacji (ENISA) opublikowała niedawno dokument przedstawiający zagrożenia bezpieczeństwa związane z przetwarzaniem w chmurze. Wśród obaw poruszonych w tym dokumencie znajdują się ochrona danych, niepewne lub niekompletne usuwanie danych oraz możliwość złośliwych informacji wewnętrznych. Inne zgłoszone problemy związane z bezpieczeństwem obejmują segregację danych, kontrolę nad lokalizacją danych i wsparcie dochodzeniowe. Większość systemów, które tutaj opisaliśmy, nie rozwiązuje w odpowiedni sposób kilku z wyżej wymienionych problemów związanych z bezpieczeństwem, a także zaostrza problem, projektując systemy, które, jak się zakłada, działają w zaufanym środowisku: pozwala nam to na konstruowanie sytuacji, w niektórych z tych systemów, gdzie złośliwy byt może siać spustoszenie. Kwestie związane z bezpieczeństwem i zaufaniem muszą zostać dokładnie rozwiązane, zanim te ustawienia będą mogły być używane do informacji o znaczeniu krytycznym i wrażliwym.