Cloudweaver: Adaptacyjny i oparty na danych menedżer obciążeń dla chmur ogólnych

Przetwarzanie w chmurze oznacza najnowszy trend w rozwoju aplikacji do przetwarzania równoległego na ogromnych ilościach danych. Opiera się na chmurach serwerów do obsługi zadań, które wcześniej były zarządzane przez pojedynczy serwer. Dzięki przetwarzaniu w chmurze dostawcy oprogramowania mogą świadczyć usługi analizy biznesowej i analizy danych dla zestawów danych w skali internetowej. Wiele projektów open source, takich jak Hadoop, oferuje różne komponenty oprogramowania, które są niezbędne do budowania infrastruktury chmury. Obecny Hadoop (i wiele innych) wymaga od użytkowników konfigurowania infrastruktur chmurowych za pomocą programów i interfejsów API, a taka konfiguracja jest naprawiana w czasie wykonywania. W tej części proponujemy menedżera obciążenia (WLM) o nazwie CloudWeaver, który zapewnia automatyczną konfigurację infrastruktury chmurowej do wykonywania runtime. Zarządzanie obciążeniem jest oparte na danych i może dostosowywać się do dynamicznego charakteru przepustowości operatora podczas różnych faz wykonania. CloudWeaver działa dla pojedynczego zadania i obciążenia składającego się z wielu zadań działających jednocześnie, co ma na celu maksymalną przepustowość przy użyciu minimalnego zestawu procesorów.

Wstęp

Cloud Computing to najnowszy trend w tworzeniu aplikacji do przetwarzania równoległego na ogromnych ilościach danych. Opiera się na chmurach serwerów do obsługi zadań, które wcześniej były zarządzane przez pojedynczy serwer. Dzięki Cloud Computing dostawcy oprogramowania mogą świadczyć usługi analizy biznesowej i analizy danych dla zestawów danych w skali internetowej. Ze względu na rozmiar tych zestawów danych, tradycyjne równoległe rozwiązania bazodanowe mogą być zbyt drogie. Aby móc przeprowadzać tego typu analizy na skalę internetową w opłacalny sposób, kilka firm opracowało rozproszone systemy przechowywania i przetwarzania danych na dużych klastrach serwerów towarowych typu „shared-nothing", w tym Google App Engine Antoszenkow (1996), Amazon EC2 /S3/SimpleDB Acker, Roth i Bayer (2008), usługi danych Microsoft SQL Server DeWitt, Naughton, Schneider i Seshadri (1992) oraz IBM „Blue Cloud". Jednocześnie projekty open source, takie jak Hadoop, oferują różne komponenty oprogramowania, które są niezbędne do budowy infrastruktury chmury. Obecny Hadoop (i wiele innych) zapewnia wirtualizację lokalizacji danych, współbieżną koordynację wykonywania i równoważenie obciążenia między serwerami; wymaga jednak od użytkowników skonfigurowania infrastruktury chmury za pomocą programów i interfejsów API, a ponadto taka konfiguracja jest naprawiana w czasie wykonywania. Na przykład programista Hadoop musi ręcznie ustawić stosunek liczby serwerów/zadań wykorzystywanych przez funkcje reduktora do liczby serwerów wykorzystywanych przez funkcje/zadania mapowe Davlid, DeWitt, Shanker (2008). Ponadto model programowania Hadoop ogranicza się do rozszerzonych funkcji SQL zdefiniowanych przez użytkownika i procedur składowanych, co jest mniej przyjazne dla użytkownika i mniej elastyczne w porównaniu z językiem SQL obsługiwanym przez równoległe bazy danych. Jak dotąd Hadoop obsługuje równoległość tylko do prostych zadań, zamiast obliczeń ogólnego przeznaczenia wymaganych przez ETL i DBMS. W tym rozdziale, motywowani ograniczeniami Hadoop, wdrażamy menedżera obciążenia (WLM), CloudWeaver, aby zapewnić automatyczną konfigurację infrastruktury chmury do wykonywania w środowisku wykonawczym. Dzięki temu programiści nie muszą ustawiać proporcji między maperami a reduktorami, gdy jest zarządzany przez CloudWeaver. Co więcej, nie jest możliwe, aby programiści ustawiali takie proporcje, gdy włączone jest potokowanie, ponieważ byłoby wiele warstw serwerów/zadań działających jednocześnie. Obciążenie na każdym poziomie może zmieniać się dynamicznie w zależności od dystrybucji danych i charakterystyki operatorów. Tak więc przetwarzanie w chmurze jest w rzeczywistości oparte na danych, a przepustowość każdego operatora/zadania zmienia się podczas fazy obliczeń. CloudWeaver może dostosować się do

dynamicznego charakteru przepustowości operatora podczas różnych faz wykonania, co więcej, działa zarówno dla pojedynczego zadania, jak i obciążenia wielu zadań działających jednocześnie, dążąc do maksymalnej przepustowości przy minimalnym zestawie procesorów.

Przegląd systemu

W tej sekcji opisujemy architekturę systemu CloudWeaver. Najpierw krótko omówimy Hadoop i jego komponenty. Hadoop to zintegrowany system dla zadań Map/Reduce. Działa na dużym klastrze z HDFS (Hadoop Distributed File System). HDFS ma pojedynczy Namenode, który zarządza przestrzenią nazw systemu plików i reguluje dostęp klientów do plików. Każda maszyna ma Datanode, który zarządza pamięcią podłączoną do maszyny. Każdy plik danych w HDFS jest przechowywany jako wiele małych bloków danych, zwykle o stałym rozmiarze. Każdy blok ma 2 lub 3 repliki zlokalizowane w różnych Datanodach. Korzystanie z wielu kopii małych bloków danych zapewnia lepszą dostępność i dostępność. Wykonanie Map/Reduce jest oparte na HDFS. Użytkownik przesyła konfigurację zadania Map/Reduce za pośrednictwem klienta zadania. Węzeł główny będzie utrzymywał śledzenie zadań i rozwidla wiele węzłów podrzędnych w celu wykonania zadań mapowania/redukowania. Każdy węzeł podrzędny posiada moduł do śledzenia zadań, który zarządza mapowaniem lub redukcją instancji zadań na tym węźle. W porównaniu z Hadoop, nasz nowy proponowany system chmur ogólnych, o nazwie CloudWeaver, ma następujące rozszerzenia:

• Dodano Cloud Monitor w celu monitorowania wykorzystania zasobów procesora i stanu zużycia danych wyjściowych procesora (tj. wyników). Służy również do dodawania nowych serwerów do chmury lub wyłączania niektórych narzędzi obliczeniowych.v

• Chmura Hadoop została rozszerzona tak, aby była bardziej ogólna dla komputerów ogólnego przeznaczenia jako chmura ogólna. Aby włączyć ogólną chmurę, zadania dla CloudWeaver zostały również rozszerzone z opisu zadania MapReduce do DAG z operatorami ogólnego przeznaczenia.

• Dodano menedżera obciążenia (WLM), który automatyzuje przydzielanie procesorów do zadań i przypisuje zadania do procesorów.

Ogólna chmura jest dostarczana jako sprzęt komputerowy. Użytkownik może, ale nie musi znać szczegółów swoich konfiguracji, a konfiguracja może zostać zmieniona. Użytkownik prześle zadanie zapytania do ogólnej chmury z komputera klienckiego. Zadanie zapytania można traktować jako oznaczone drzewo operatorów, dzięki czemu znamy przepływ pracy lub przepływ danych. Uwzględniono również nazwy plików wejściowych niektórych węzłów. Zakładamy, że te pliki wejściowe znajdują się w systemie pamięci masowej chmury ogólnej. Innymi słowy, chmura może odczytywać pliki z ich nazw. Łagodzimy również założenie HDFS, obsługując zarówno współużytkowane systemy plików, jak i systemy plików inne niż HDFS. W systemach plików współdzielonych każdy procesor może uzyskać dostęp do pamięci bezpośrednio przez jakiś wspólny interfejs. Inne rozproszone systemy plików mogą mieć inne zasady przechowywania. Zakładamy, że nazwa node i data node w CloudWeaver zapewniają interfejs dostępu do danych w postaci małych bloków podobnych do HDFS. Nasz algorytm planowania jest przeznaczony do uruchamiania wielu zadań na małych blokach danych w celu poprawy wydajności i osiągnięcia równoważenia obciążenia.

Menedżer zadań

Menedżer obciążenia zaakceptuje zadanie zapytania i jest odpowiedzialny za przetwarzanie zapytania. Zna stan całego systemu: gdzie jest nazwa węzła, gdzie są serwery obliczeniowe, a gdzie jest system pamięci masowej. Każda zmiana w środowisku chmury zostanie zauważona przez menedżera obciążenia. WLM przygląda się drzewu operatorów zadaje zapytania i przetwarza zadania w modelu

opartym na danych. Oznacza to, że WLM zaplanuje małe zadania do uruchomienia na serwerach. Każde zadanie zajmie mały blok danych wejściowych i wygeneruje kilka plików wyjściowych. WLM zaplanuje, że plik wyników pośrednich będzie zasilał inne operatory do momentu wygenerowania danych wyjściowych zadania zapytania. W tym procesie WLM zajmuje się ogólną zmianą chmury i postępem pracy, dzięki czemu może dynamicznie wykorzystywać wszystkie dostępne zasoby. Węzeł nazwy utrzymuje katalog wszystkich plików. Można go uznać za interfejs systemu plików. Źródłowe pliki wejściowe znajdują się w systemie pamięci masowej. Gdy serwer przetwarza zadanie, zapyta węzeł nazwy o adres dostępu do plików, a następnie odczyta je lub zapisze nowe pliki. System pamięci masowej może być albo wspólną pamięcią masową, albo strukturą bez współdzielenia. Zakładamy, że istnieje centralny węzeł, który utrzymuje wszystkie powiązane pliki w ogólnej chmurze.

Monitor chmury

Ponieważ system oparty jest na modelu producent-konsument, dane wyjściowe zadań niższego rzędu są wykorzystywane jako dane wejściowe dla zadań wyższego rzędu. Każde zadanie przechowuje swoje dane wyjściowe (tj. wyniki pośrednie) na dysku lokalnym według wstępnie zdefiniowanego rozmiaru bloku. Pośrednie bloki wyników są następnie odczytywane przez zadania wyższego poziomu. Jeśli liczba pośrednich bloków wyników rośnie, Cloud Monitor może zauważyć, że WLM zwiększa liczbę zadań wyższego poziomu, aby zużywać coraz większą liczbę bloków.

Ogólna chmura

Ogólna chmura ma klaster serwerów o mocy obliczeniowej. Duży zestaw danych jest przechowywany w chmurze lub może zostać przesłany do chmury w celu wykonania zadania przetwarzania danych. W dalszej części tej części każde zadanie przetwarzania danych nazywane jest w skrócie zadaniem. Zajmujemy się głównie pytaniami w tym rozdziale. Zadanie można zrównoleglać na małe zadania. Zadania są wykonywane na różnych serwerach. Wydajność została poprawiona dzięki zastosowaniu równoległości. Serwery mogą mieć różną moc obliczeniową i wielkość pamięci. Planowanie zadań w ogólnej chmurze w celu uzyskania najlepszego czasu odpowiedzi to trudny problem z optymalizacją. Predefiniowany algorytm planowania jest trudny do radzenia sobie ze zmieniającym się środowiskiem. W tej części rozwiązujemy problem planowania w czasie wykonywania. Sprawdzamy wymagania dotyczące przetwarzania danych i stan chmury w czasie wykonywania, określamy liczbę zadań i przypisanie zadań do serwerów. Ponieważ każda partycja i krok planowania są oparte na dostępnych danych, które muszą zostać przetworzone, uważamy, że ta metoda oparta na danych może najlepiej zrównoważyć obciążenie serwerów w ogólnej chmurze i osiągnąć najlepszą wydajność. Uważamy, że SQL jest jak przetwarzanie zapytań dla dużego zestawu danych z obsługą MapReduce.

Zapytanie o pracę

Zadanie zapytania użytkownika można opisać za pomocą wykresu operatora. Operatory obejmują funkcje Extract, Join, Aggregate i Map Reduce. Funkcja mapowania i zmniejszania jest dostarczana przez użytkownika. W środowisku równoległym zachowanie operatora można podzielić na kilka małych zadań i uruchomić równolegle na różnych serwerach. Każde zadanie może być uruchamiane przez plik wykonywalny. Polecenie pobiera kilka plików wejściowych i generuje pliki wyjściowe. W ten sposób możemy kierować pliki danych i plik wykonywalny na różne serwery, plik wykonywalny zużywa pliki wejściowe i generuje dane wyjściowe.

Wejście

Zakładamy, że dane wejściowe są bardzo duże. Duże pliki wejściowe mogą być traktowane jako tabele w RDMBS. Każdy plik możemy traktować jak dużą tabelę. Aby WLM mógł korzystać z równoległości,

WLM musi wiedzieć, jak każdy operator może być zrównoleglony. Na przykład prosty wyciąg z tabeli można dowolnie podzielić na wiele małych plików. Każdy plik może być przetwarzany przez zadanie wyodrębniania na dowolny serwer. Operator sortowania może być również zrównoleglony przez wiele małych zadań sortowania z zadaniem łączenia, ale dane wyjściowe nie mogą być ustalone, chyba że wszystkie dane wejściowe zostały przetworzone. Ten rodzaj operatora nazywa się operatorem blokującym. Operator złączenia to kolejny złożony przykład. Możemy podzielić dane wejściowe i użyć wielu serwerów do przetwarzania złączenia. Gdy WLM chce zaplanować więcej serwerów do przetworzenia przyłączenia, status oryginalnych przyłączonych serwerów musi zostać zmigrowany lub zmieniony. Różnice naszej pracy w porównaniu z innymi polegają na tym, że nie zakładamy SMP ani klastra z identycznymi serwerami. Zamiast tego mamy do czynienia z maszynami o różnej mocy. Zajmujemy się dynamicznym charakterem przepustowości danych. Na przykład, gdy mamy sprzężenie, szybkość przetwarzania jest stała, ale szybkość wyjściowa wyniku zmienia się przez cały okres przetwarzania. Nasz framework ma również dużą różnicę w porównaniu z Map/reduce lub Hadoop. System Hadoop ma na celu zapewnienie abstrakcji (wirtualizacji) podstawowego sprzętu/lokalizacji plików/równoważenia obciążenia, tak aby aplikacje mogły skoncentrować się na pisaniu map i ograniczaniu funkcji. W naszej pracy CloudWeaver zapewnia podobną funkcjonalność, ale skupia się na koordynowaniu realizacji złożonych zadań (wirtualizacja zarządzania przepływem realizacji i optymalizacja ze strony programistów). Wykonanie Map/Reduce lub Hadoop jest znacznie prostsze, ponieważ wykonanie ma tylko dwie fazy i nie obejmuje złożonego przepływu danych. Jest to podobne do SQL: użytkownicy muszą określić, czego chcą za pomocą SQL i nie muszą określać, jak wykonywać zapytania i jak je optymalizować. Optymalizacja i wykonanie odbywa się automatycznie przez system bazy danych. Nasz system to potężne wdrożenie do przetwarzania danych w środowisku chmury. Cały system przetwarzania danych może składać się z trzech ważnych elementów jeden na drugim. Pierwszym z nich jest wprowadzenie przez użytkownika opisu zadania. Drugi to zrównoleglenie i wykonanie. Trzecia, podobnie jak pamięć masowa, zapewnia infrastruktura. W naszym systemie menedżer obciążenia koncentruje się na drugiej fazie wykonania, przeprowadzając mapowanie procesora/zadania. Wychodzimy z założenia, że wybór zestawu odpowiednich serwerów jest obsługiwany przez infrastrukturę. Hadoop i Map/Reduce zapewniają wszystkie trzy fazy. To sprawia, że nie jest ogólny. Nasz system może poradzić sobie z zadaniem wejściowym użytkownika i ma dobrą wydajność w dowolnej infrastrukturze. Dryad jest podobny do naszego systemu w tym sensie, że paralelizuje sekwencyjny przepływ danych, ale ma lokalną optymalizację tylko dla operatorów, którzy działają wolniej, podczas gdy nasz algorytm planuje całe zadanie DAG w sposób oparty na danych, który jest bardziej elastyczny i rozszerzalny. Poza tym Driada nie została rozszerzona o planowanie wielu zadań.

Terminologia

Opisujemy określone przez użytkownika zadanie za pomocą DAG, gdzie każdy węzeł Ni jest operatorem, który zapewnia abstrakcję podzadań, a każda krawędź Ei,j między dwoma operatorami Ni i Nj wskazuje przepływ danych od Ni do Nj . Przykład pokazano po lewej stronie rys. 9.2. Ma sześć operatorów nazwanych od A do F i dwie gałęzie A → B → \ i E → F, z operatorem znajdującym się najwyżej D. A i F to węzły liści (węzły we/wy) oraz B,C,D,E są węzłami nie będącymi liśćmi (węzłami obliczeniowymi). To zadanie wejściowe można odwzorować na zapytanie, które po wykonaniu pewnych wyborów i projekcji wykonuje łączenie dwóch tabel. Zadanie wejściowe jest abstrakcją wysokiego poziomu i nie mówi, jak zrównoleglić przetwarzanie danych. Oprócz tego wykresu chmura powinna znać nazwę pliku wejściowego oraz funkcję każdego operatora. Używamy równoległości między operatorami, aby przyspieszyć wykonanie każdego operatora. Jak pokazano w prawej części Rys. 9.2, każdy operator może wykonać wiele małych zadań. Zadania są jednostkami przetwarzania na

poziomie systemu operacyjnego i mogą być przeglądane jako wątki/procesy. Zadania są niezależne i nic nie dzielone. Do zadania można przypisać cały procesor lub część procesora współdzieloną z innymi zadaniami. Używamy terminu procesor, aby być bardziej ogólnym. W zależności od środowiska, w którym działa zadanie, procesorem może być serwer (dla klastra), węzeł (dla sieci/chmury) lub procesor/rdzeń dla ustawienia SMP/CMP. Przydzielamy zadania procesorom w chmurze. Ten krok nazywa się mapowaniem procesora/zadań. Każdy procesor może uruchamiać jedno lub więcej zadań od tego samego operatora lub różnych operatorów. . Zakładamy architekturę bez współdzielenia. Każdy procesor ma swoje dyski lokalne. Dane wejściowe są przechowywane na niektórych dyskach. Węzeł może przetwarzać pliki przechowywane w innym węźle poprzez przesyłanie plików przez sieci. Naszym celem jest zaplanowanie postępów wszystkich operatorów, aby zminimalizować pośrednie pliki wyników i maksymalnie wykorzystać procesory w chmurze.

Status równoległości operatora

Menedżer obciążenia będzie utrzymywał status każdego operatora. W szczególności w naszym podejściu opartym na danych zarządza on dwoma zestawami operatora O: kolejką wejściową plików danych wejściowych o nazwach Q.input i kolejką wyjściową plików danych wyjściowych o nazwach Q.output. Specyfikacja użytkownika opisuje, w jaki sposób dane wejściowe mogą być przetwarzane (na przykład łączenie lub agregacja), w szczególności przez określony rodzaj operatora. Pliki wejściowe (dane) zostaną podzielone na wiele bloków danych, tak jak zostało to zrobione w Hadoop. Wiele bloków danych od operatora może być wykonywanych jednocześnie przez wiele zadań w różnych procesorach. Migawka wszystkich danych wejściowych/wyjściowych i uruchomionych zadań związanych z operatorem jest traktowana jako stan operatora. W szczególności status operatora może być następujący: oczekiwanie na dane (nierozpoczęty), w trakcie przetwarzania lub zakończony. Oprócz statusu operatora powinniśmy również odnotować, w jaki sposób operator jest zrównoleglony, ile zadań jest przypisanych do każdego operatora oraz parametry. Na przykład, na początku przypisaliśmy 5 procesorów do wykonania operatora skanowania. Menedżer obciążenia będzie rejestrować, ile zostało zrobione przez każdy procesor. Później, gdy dostępnych będzie więcej zasobów (np. procesorów), menedżer obciążenia może zdecydować o dystrybucji pozostałych danych do przeskanowania do łącznie 10 procesorów. Decyzję należy podjąć zgodnie ze statusem operatora. Monitorując status operatora, menedżer obciążenia będzie wiedział, jak zrównoleglić i zaplanować wykonanie każdego operatora. Kolejka wejściowa do każdego operatora będzie monitorowana jako bufor wejściowy. Gdy bufor wejściowy szybko rośnie, oznacza to, że operator potrzebuje więcej mocy obliczeniowej i jeśli to możliwe, menedżer obciążenia powinien przydzielić więcej procesorów do wykorzystania danych wejściowych. Statystyki działania mogą być zbierane z uruchomionych zadań na różnych procesorach, które mogą być używane przez menedżera obciążenia do mapowania odpowiednich procesorów na operatora/zadanie w celu osiągnięcia lepszego wykorzystania procesora.

Algorytm wykonywania zadań

Nasz algorytm oparty na danych ma na celu określenie procesorów i harmonogramu zadań dla każdego operatora. Biorąc pod uwagę zadanie przetwarzania danych, mamy dwa typy operatorów blokujących operatorów i operatorów nieblokujących. Jeśli operator nie może wygenerować żadnych danych wyjściowych, zanim wszystkie jego dane wejściowe są gotowe, nazywamy go operatorem blokującym. Operatory następcze operatora blokującego nie mogą zostać uruchomione, dopóki operator blokujący nie zakończy pracy, na przykład operator sortujący. Wręcz przeciwnie, operator nieblokujący i operatory jego sukcesu mogą być wykonywane w sposób potokowy, na przykład operator skanowania, po którym następuje operator łączenia. Mając drzewo operatora odpowiadające opisowi zadania użytkownika, rozważamy trzy scenariusze, na które nasz algorytm będzie celował. Pierwsze dwa omówione są dla operatorów nieblokujących, a ostatni dla operatorów blokujących. Menedżer

obciążenia ma na celu dynamiczne przypisanie zbioru procesorów do każdego operatora, zwanego zbiorem procesorów kandydujących operatora. Zadania peratora są zaplanowane w kandydującym zestawie procesorów. Kontrolując rozmiar kandydującego zestawu procesorów, menedżer obciążenia faktycznie kontroluje moc wykonania dla określonego operatora. Operator bez zadania do wykonania będzie miał pusty zestaw procesorów kandydujących.

1. Załóżmy, że mamy wystarczające zasoby, a operatorzy są operatorami nieblokującymi. W takim przypadku za każdym razem, gdy operator ma określoną ilość danych wejściowych, menedżer obciążenia przydzieli operatorowi niepusty procesor kandydujący i zaplanuje zadania. Menedżer obciążenia będzie dążył do pełnego wykorzystania każdego procesora. Gdy zmienia się szybkość odbierania danych, a bieżący kandydujący zestaw procesorów nie pasuje do wejściowej kolejki danych, menedżer obciążenia może zwiększyć/zmniejszyć kandydujący zestaw procesorów.

2. Załóżmy, że mamy ograniczone zasoby, a operatorzy nie blokują się. W takim przypadku, gdy operator ma określoną ilość danych wejściowych, menedżer obciążenia może nie zawsze być w stanie znaleźć dostępne procesory dla operatora. Załóżmy, że wszystkie operatory są operatorami nieblokującymi, a wykonywanie oparte na danych rozpocznie się od operatora liścia. Gdy istnieją dane dla innych operatorów, menedżer obciążenia uruchomi operatora, jeśli zostanie znaleziony możliwy zestaw procesorów kandydujących. Menedżer obciążenia spróbuje zaplanować więcej operatorów w tym samym czasie w sposób potokowy, a nie jeden po drugim. Jeśli zasoby są ograniczone, możemy zmniejszyć rozmiar zestawu kandydatów dla każdego operatora węzła liścia i przypisać procesory do operatorów w drzewie danych. Gdy operator zakończy całą swoją pracę, jego kandydujący zestaw procesorów może zostać zwolniony i używany przez innych operatorów.

3. Załóżmy, że niektórzy operatorzy blokują operatory. W tym przypadku nie możemy rozszerzyć wykonywania potoku dalej na operatory nadrzędne operatorów blokujących. Dla operatora Ni, wszystkim operatorom pod jego węzłem w drzewie operatorów przypisano kandydujący zestaw procesorów. Wszystkie procesory przypisane do tej gałęzi nazywamy zestawem procesorów gałęzi. Jeśli Ni jest operatorem blokującym, menedżer obciążenia może rozpocząć przypisywanie procesorów do nowych gałęzi w drzewie operatorów, które mają dane wejściowe. Jeśli nie ma innej gałęzi, pozostałe dostępne procesory można dodać do gałęzi Ni. Dodajemy procesor od węzła liścia do Ni, co oznacza, że przydzielamy więcej mocy procesora operatorom źródłowym i przesyłamy dane w stylu bottom-up .Gdy operator liścia jest kompletny, ograniczony zasób można przenieść do operatorów wyższego poziomu.

Może istnieć wiele możliwych podejść do określenia, których operatorów należy zaplanować w pierwszej kolejności. Można zastosować wiele rozwiązań optymalizacyjnych. W naszej pracy najpierw skupiamy się na prostym sposobie oddolnym i wybieramy operatorów w określonej kolejności (na przykład post-order), o ile operator może zostać uruchomiony. Dążymy do osiągnięcia pełnego wykorzystania mocy procesora i lepszego zarządzania postępem potoku wzdłuż drzewa operatorskiego.

Dynamiczna równoległość do wykonywania zadań

Podano zadanie DAG z n operatorami, N_1, … ,N_n oraz zbiór m procesorów o mocy obliczeniowej P_1, … ,P_n zakładamy odpowiednio, że krawędź $E_{i,j}$ pomiędzy N_i i N_j jest krawędzią rurociągu, jeśli N_j nie jest operatorem blokującym. Krawędź potoku wskazuje, że częściowe wyjście operatora N_i może zostać przekazane do N_j w celu wykonania następnego poziomu. Jeśli N_j jest operatorem blokującym, nazywamy krawędź $E_{i,j}$ krawędzią blokującą, co oznacza, że operator N_j musi czekać na wszystkie dane ze swoich operacji wejściowych, aż będzie można uzyskać jakikolwiek wynik. Dynamiczny planista ma na celu maksymalizację równoległości i zrównoważenie obciążeń między procesorami. Ponieważ

wymagane procesory dla każdego operatora (z wyjątkiem węzłów liści) są nieznane, zanim dane będą gotowe, dynamiczny planista musi wykonać fazę „eksploracji", aby oszacować szybkość wejścia i wyjścia każdego operatora. W szczególności, program planujący najpierw przypisuje jeden procesor Pi do każdego operatora skanowania (inaczej operator liścia) N_i. W tym kroku możemy losowo wybrać procesor i przypisać do zadań operatora na poziomie liścia. Załóżmy, że szybkość wyjściowa operatora N_i działającego na procesorze P_j wynosi $R_{i,k}$, a wyniki wyjściowe są podawane do innego operatora N_k jako jego dane wejściowe. Jeśli krawędź pomiędzy N_i i N_k jest krawędzią potoku, aby dwa operatory N_i i N_k działały jednocześnie, planista musi przydzielić procesory, aby zużywały dane wyjściowe z N_i z w przybliżeniu taką samą szybkością $R_{i,k}$. Podobnie, po przetworzeniu operatora N_k, program planujący będzie znał szybkość wyjściową N_k i odpowiednio przyporządkuje procesor. Ta sama procedura jest powtarzana aż do napotkania krawędzi blokującej. W takim przypadku operator blokowania, taki jak sortowanie/agregacja, będzie czekał na przybycie wszystkich danych źródłowych z jego operatorów podrzędnych. Gdy wszystkie źródła są gotowe do przetworzenia, planujemy odpowiednie zadania, aby dopasować dane wejściowe operatora blokującego. Omówione dotychczas przypisanie procesorów zakłada jedynie, że istnieje wystarczająca liczba procesorów do przypisania. Gdy procesory nie są wystarczające, planista powinien zredukować procesory przypisane do węzłów liściowych i odpowiednio dostosować przypisanie procesorów do węzłów innych niż liście. Teraz krótko opiszemy, jak menedżer obciążenia planuje zadania przykładowego w chmurze z X maszynami. Każdy operator może być zrównoleglony i wykonywany przez wiele zadań. A → B → C lub E → F mogą być wykonywane jako potok, a D może działać tak długo, jak długo dostępne są odpowiednie dane z C i E. Zadanie jest zaplanowane w czasie rzeczywistym przez menedżera obciążenia. Wyjaśniamy, w jaki sposób podejście oparte na danych poradzi sobie z potencjalnymi ograniczeniami we/wy poprzez wykorzystanie potoku i nakładających się zadań we/wy i procesora. Jest wykonywany w następujących krokach.

Krok 1: określ zadania dla operatora A.

Krok 2: wybierz odpowiednie węzły serwera a1, a2, a3 do wykonania zadań A

Krok 3: węzły a1, a2, a3 wykonują zadania operatora A i mogą być postrzegane jako pojedynczy węzeł „1".

Krok 4: znajdź zestaw serwerów do obsługi danych wyjściowych z węzła „1". Powiedzmy, przydzielamy b1, b2, b3, b4.

Krok 5: te 7 węzłów można oglądać razem jako węzeł „2".

Krok 6: znajdź zestaw serwerów wystarczający do obsługi danych wyjściowych z węzła „2".

Krok 7: cały serwer w lewej gałęzi może być postrzegany jako węzeł „3", a w tym węźle 3 wszystkie serwery są zajęte, ponieważ dopływ danych i moc obliczeniowa są dopasowane, a zatem wykorzystanie zasobów jest wysokie.

Krok 8: skonstruuj węzeł „4", „5" podobnie jak konstruujemy węzeł „3".

Krok 9: następnie węzły 3 i 5 można połączyć w węzeł „6"

Krok 10: znajdź zestaw serwerów do obsługi wyjścia węzła „6", czyli d1, d2, d3.

Zauważ, że staramy się znaleźć zestaw właściwych „stosunków" dla serwerów pomiędzy sąsiednimi poziomami, zilustrowanymi przez węzły liczbowe. Kiedy już znajdziemy taki zestaw wskaźników, dla dowolnej liczby maszyn, wiemy jak przypisać serwery (lub moce obliczeniowe poprzez współdzielenie serwerów) każdemu operatorowi. Takie podejście idealnie zapewnia najlepszy czas odpowiedzi przy użyciu minimalnej liczby serwerów (większość czasu podczas wykonywania), ponieważ wszystkie

maszyny są w pełni wykorzystane. Alternatywnie możemy użyć wielu maszyn dla operatora A i F, a następnie kolejno kończyć B,C,E,D. To naiwne podejście jest najwyraźniej nieefektywne, jeśli nie możemy wykorzystać wszystkich maszyn dla jednego z operatorów. Aby określić najlepszy stosunek między dwoma poziomami, dążymy do zrównoważenia wejścia/wyjścia lub wielkości środkowego wyniku. Innymi słowy, chcemy utrzymać ilość oczekujących bloków danych poniżej progu T. Gdy jest więcej bloków danych niż próg, staramy się dodać więcej zasobów, aby więcej bloków danych zostało przetworzonych. Gdy liczba bloków danych jest mniejsza niż próg, możemy zmniejszyć liczbę procesorów. Więc ostatecznie stosunek można osiągnąć w okolicach progu. Najwyraźniej dla małej wartości T całe zadanie zostanie wykonane szybciej.

Bilansowanie operatorów potokowych

W ogólnym przepływie pracy możemy wyeliminować wyniki pośrednie, jeśli przedłużymy wykonanie operatorów potoku. Jeśli zasoby obliczeniowe są nieograniczone, zawsze możemy przydzielić nowe zadania procesorom. Ogólnie zasoby obliczeniowe dla jednego zadania są ograniczone. W takim przypadku WLM musi inteligentnie przydzielać zadania do ograniczonej liczby zasobów. Trzech operatorów ma różne szybkości przetwarzania dla tego samego rozmiaru danych. Nie możemy dokładnie wiedzieć, jak szybko dane zostaną wygenerowane przed rozpoczęciem zadania, dlatego alokację zasobów najlepiej przeprowadzać w czasie rzeczywistym. Skupiamy się na równoważeniu operatorów potokowych W ogólnym przepływie pracy możemy wyeliminować wyniki pośrednie, jeśli możemy przedłużyć wykonanie operatorzy rurociągu. Jeśli zasoby obliczeniowe są nieograniczone, zawsze możemy przydzielić nowe zadania procesorom. Ogólnie zasoby obliczeniowe dla jednego zadania są ograniczone. W takim przypadku WLM musi inteligentnie przydzielać zadania do ograniczonej liczby zasobów. Trzech operatorów ma różne szybkości przetwarzania dla tego samego rozmiaru danych. Nie wiemy dokładnie, jak szybko dane zostaną wygenerowane przed rozpoczęciem pracy, dlatego alokację zasobów najlepiej przeprowadzać w czasie rzeczywistym.

Poziomy równoważenia

Następnie pokażemy, jak wybrać serwery, aby zrównoważyć obciążenie między sąsiednimi warstwami. Ponieważ jest oparty na danych, najprościej jest – przypisać jeden serwer lub minimalną liczbę serwerów do podzadań związanych z I/O (powiedzmy, operatorzy A i B) – wtedy możemy określić, ile serwerów należy przypisać do B, C, E, D odpowiednio obsługują dane wyjściowe odpowiednio z A i F. Pomysł jest taki: przypisz jeden węzeł I/O; przechodzenie od węzła liścia do góry, aż do węzła złączenia w węźle złączenia, wiele gałęzi musi być „zsynchronizowanych" we wszystkich gałęziach; w tym celu należy dodać serwery przemierzające gałęzie; po tym, jak wszystkie gałęzie w węźle dołączenia są zsynchronizowane (zrównoważone), kontynuuj ruch w górę; po dotarciu na szczyt określamy proporcje między serwerami między sąsiednimi poziomami. Możemy pomnożyć liczbę (tj. 2, 3, 4, ...) do liczby serwerów w każdym węźle i rozwinąć wykres wykonania. W przykładzie wspominamy o „stosunku" dla węzłów numerycznych. Rzeczywisty stosunek zmieni się w czasie przebiegu i zmieni się z jednego przebiegu na drugi. Współczynnik może nic nie znaczyć, chyba że wszystkie jednostki obliczeniowe zostaną „znormalizowane". Stosunek daje nam odniesienie, jak przydzielić moc procesora do różnych podzadań, aby można było zrównoważyć obciążenie.

Planowanie wielu prac

Aby zaplanować wiele zadań, najpierw mapujemy określoną liczbę procesorów dla każdego zadania, które można uznać za jego indywidualną pulę procesorów. W przypadku każdego zadania menedżer obciążenia przypisze swoje zadanie tylko do maszyn we własnej puli. To mapowanie procesora/zadania jest wykonywane przez WLM podczas inicjowania zadania. Jeśli zasoby są ograniczone, pozwalamy, aby procesor był współdzielony przez wiele zadań, ponieważ niektóre z procesorów mogą być potężne.

Procesor 2 jest współdzielony przez oba zadania. Podejście polegające na początkowej puli pracowników jest dobre w przypadku chmury na dużą skalę, ponieważ możemy wtedy łatwo kontrolować zasoby wykorzystywane przez każde zadanie. Pulę można dynamicznie zmieniać w czasie wykonywania, aby osiągnąć wiele celów związanych z planowaniem. Na przykład możemy po prostu zawiesić zadanie, czyszcząc jego pulę pracowników. Możemy przyspieszyć/zwolnić przetwarzanie, dodając/zmniejszając procesory w puli pracowników zadania. Początkowy rozmiar puli może nie być odpowiedni dla wszystkich zadań. W czasie rzeczywistym WLM może przydzielić więcej/mniej pracowników do zadania, które ma więcej/mniej zadań do zaplanowania. Na przykład, jeśli zadanie jest najpierw wykonywane przez 10 maszyn w celu wykonania skanowania we/wy, a ostatni krok jest wykonywany przez pojedyncze zadanie na jednym komputerze, nieużywane procesory mogą zostać wcześniej zwolnione do przetwarzania innych zadań z ograniczonymi zasobami. Podajemy wewnętrzny projekt WLM do planowania zadania DAG w Algorytmie 1. WLM działa jako centralny program planujący. Po przesłaniu nowego żądania zadania, WLM najpierw zainicjuje zadanie i zażąda początkowego zestawu pracowników zadania, a następnie , WLM zaplanuje zadanie. Po zakończeniu zaplanowanego zadania WLM zaplanuje inne zadania odpowiadające mu zadaniu.

Powiązana praca

Bazy danych równoległych

Wykazano, że przetwarzanie równoległe na wielu procesorach lub w architekturze typu „shared-nic" cechuje się wysokim stopniem skalowania i przyspieszania. Niektóre z najstarszych systemów to: GAMMA, Bubba, PRISMA/DB. Równoległość można wykonać za pomocą równoległości między operatorami (użyj potoku lub zrównolegnij drzewo krzaczaste) i równoległości między operatorami (podziel na partycje i zrównoleglenie wewnątrz operatora relacyjnego). Wiele wysiłku włożono w zrównoleglenie operatorów relacyjnych, takich jak algorytm łączenia. Optymalizacja zrównoleglonego planu i harmonogramowania zapytań była szeroko badana, np. planowanie potokowych operatorów zapytań, statyczne planowanie zadań i alokacja zasobów, równoważenie obciążenia z obsługą skośną, zarządzanie operator równoległość z wieloma użytkownikami, dynamiczna optymalizacja planu zapytań i migracja według bieżących statystyk. W celu optymalizacji dynamicznego planu zapytań możemy również przygotować wiele różnych planów zapytań i wybrać jeden w czasie wykonywania lub użyć parametrów. Podejmowane są również próby poprawy wydajności baz danych na innych architekturach komputerowych, takie jak równoległe przetwarzanie zapytań w środowisku wielordzeniowym, przetwarzanie baz danych przeznaczone dla procesorów wielowątkowych symultanicznych itp. W dzisiejszych rzeczywistych aplikacjach internetowych, nad przesyłaniem strumieniowym danych potrzebna jest duża analiza danych. Wiele wysiłku włożono w spełnienie takiego wymogu zapytania.

Przetwarzanie danych w klastrze

MapReduce firmy Google jest wdrażany w dużym klastrze z systemem plików Google. W tym systemie duży zestaw danych jest przechowywany jako wielokrotne kopie małych standardowych bloków w różnych lokalizacjach w klastrze. MapReduce to standardowy model programowania, wykonujący dwie funkcje udostępnione przez użytkownika dla zadań mapowania i zadań redukcji. Wiele problemów związanych z przetwarzaniem danych można przekształcić w ten model programowania, a harmonogramowanie zadań w specjalnie zaprojektowanym systemie bardzo dobrze poprawia wydajność. Ale ze względu na ograniczenia modelu programowania nie nadaje się do wszystkich rodzajów zadań, takich jak łączenie relacji, które pobiera dwa pliki jako dane wejściowe. Ze względu na jego użyteczność i prostotę wiele pionierskich systemów bazodanowych zaczyna integrować jego funkcjonalność, w tym Aster Data i GreenPlum. Implementacja MapReduce typu open source jest

dostarczana przez Hadoop. Dyrad implementuje uniwersalny silnik wykonywania danych równoległych w klastrze. Wykorzystuje język programowania niskiego poziomu do reprezentowania DAG przepływu danych. Plan statyczny zostanie przekazany executorowi wykonawczemu w celu zaplanowania w klastrze. Ma jednak pewne wady. Po pierwsze, użytkownik musi opanować złożony język opisu wykresów. Wymaga również od użytkownika znajomości szczegółów klastra w celu określenia stopnia równoległości (przynajmniej jako sugestia). Optymalizacja polega na zmianie kształtu z góry określonego planu wykresu w czasie wykonywania. Obecnie może zajmować się tylko jednym zadaniem lub zapytaniem użytkownika. Ostatnio Robinson i DeWitt oraz Shankar i DeWitt zaproponowali planowanie przepływu pracy oparte na danych w klastrze. Clustera to niedawno opracowany prototyp zintegrowanego systemu obliczeń i zarządzania danymi. Ma dobrą rozszerzalność i może wykonywać zadania wymagające dużej mocy obliczeniowej, zadanie MapReduce, a także zapytania SQL w środowisku równoległym. Najpierw skompiluje dowolny rodzaj opisu zadania w DAG konkretnych zadań, a następnie harmonogram zadań zoptymalizuje wykonanie w chmurze. Przyjmuje również ideę wykorzystania bazy danych do problemu zarządzania klastrami. Ma porównywalną wydajność w porównaniu do Hadoop pod względem wydajności zadania MapReduce i dobrą skalowalność dla zapytań SQL. Zainspirowany programem MapReduce, zdano sobie sprawę, że do przetwarzania równoległego w środowisku bez udostępniania danych potrzebny jest język opisowy dla SQL, taki jak przetwarzanie danych. Swój pomysł zaproponowały branże, takie jak Pig Latin firmy Yahoo i SCOPE firmy Microsoft.

Wniosek

Przetwarzanie w chmurze oznacza najnowszy trend w rozwoju aplikacji do przetwarzania równoległego na ogromnych ilościach danych. Opiera się na chmurach serwerów do obsługi zadań, które wcześniej były zarządzane przez pojedynczy serwer. Zauważamy, że obecny Hadoop (i wiele innych) wymaga od użytkowników konfigurowania infrastruktur chmurowych za pomocą programów i interfejsów API w czasie wykonywania. Twierdzimy, że takie konfiguracje ad hoc mogą skutkować mniejszym wykorzystaniem i wydajnością całego systemu. Tu zapewniamy zautomatyzowaną konfigurację infrastruktury chmury dostosowującej się do wykonywania w czasie wykonywania, a także rozszerzamy obsługiwany zestaw operatorów. Oceniamy proponowaną strukturę na obu syntetycznych zadaniach obliczeniowych. Przyszłe prace obejmują automatyczną translację SQL do interfejsów API Hadoop (lub w razie potrzeby rozszerzonych interfejsów API) w prototypowym systemie do dalszej oceny.