Wykorzystanie hybrydowych technologii Grid/Cloud Computing do elastycznego przechowywania,

Modele prognozy klimatu i pogody o wysokiej rozdzielczości oraz regionalne i globalne sieci czujników generują coraz większe ilości wielowymiarowych danych środowiskowych. Aby były użyteczne, dane te muszą być przechowywane, zarządzane i udostępniane globalnej społeczności badaczy, decydentów i innych osób. Typowym podejściem do rozwiązywania tych problemów jest korzystanie z dedykowanych obiektów do przechowywania i dystrybucji danych. Na przykład siatka systemu Ziemi (ESG) obejmuje systemy danych w kilku laboratoriach w USA, każde z dużą ilością pamięci masowej i wysokiej klasy serwer skonfigurowany do obsługi żądań wielu zdalnych użytkowników. Usługi rozproszone, takie jak katalogi replik i metadanych, integrują te różne komponenty w jeden system rozproszony. Wraz ze wzrostem zarówno ilości danych środowiskowych, jak i zapotrzebowania na te dane, serwery w systemach takich jak ESG mogą łatwo ulec przeciążeniu. Większe zestawy danych prowadzą również do tego, że konsumenci chcą wykonywać potoki analizy „na miejscu", zamiast pobierać dane do analizy lokalnej, co dodatkowo zwiększa obciążenie serwerów danych. W ten sposób operatorzy stają przed ważnymi decyzjami, jak najlepiej skonfigurować systemy, aby sprostać szybko rosnącym, często bardzo zmiennym w czasie, obciążeniom. Pojawienie się komercyjnych „chmury" lub dostawców infrastruktury na żądanie - operatorów dużych farm pamięci masowej i obliczeniowych wspierających quasi-natychmiastowy dostęp na żądanie i wykorzystujących ekonomię skali w celu obniżenia kosztów – stanowi potencjalną alternatywę dla serwery obsługiwane przez systemy takie jak ESG. Przechowywanie danych środowiskowych w pamięci masowej w chmurze (np. Amazon S3) i uruchamianie potoków analitycznych na komputerach w chmurze (np. Amazon EC2) może potencjalnie obniżyć koszty i/lub poprawić jakość dostarczanych usług, zwłaszcza w przypadku reagowania na szczyty dostępu. Tu przedstawiamy wyniki badania, którego celem jest ustalenie, czy rzeczywiście jest wykonalne i opłacalne zastosowanie usług w chmurze do hostingu i dostarczania danych środowiskowych. Podchodzimy do tego pytania z dwóch perspektyw: architektury i kosztów. Najpierw badamy i przedstawiamy projekt, rozwój i ocenę infrastruktury oprogramowania opartej na chmurze, która wykorzystuje niektóre usługi przetwarzania siatkowego i jest przeznaczone do przechowywania, przetwarzania i dostarczania wielowymiarowych dane środowiskowe. W naszym projekcie wykorzystaliśmy API przetwarzania w chmurze Amazon EC2/S3, aby zapewnić elastyczne możliwości hostingu i przetwarzania danych, oraz Globus Toolkit v4 (GT4) do federacji elementów danych w szerszym kontekście aplikacji gridowych. Skalowalność zapewnia hybrydowa wirtualna/rzeczywista agregacja zasobów obliczeniowych.

Przechowywanie danych środowiskowych na zasobach elastycznych

Naszym celem w tej pracy jest zbadanie, czy możliwe jest wykorzystanie usług chmury Amazon do przechowywania danych środowiskowych. Innymi słowy, chcemy określić trudność, wydajność i koszt ekonomiczny obsługi usługi takiej jak FDDDS z danymi (a być może także z przetwarzaniem) hostowanymi nie na zasobach lokalnych, ale na zasobach w chmurze dostarczanych przez Amazon. Ta usługa, podobnie jak FDDDS, powinna umożliwiać zdalnym użytkownikom żądanie zarówno całych zestawów danych, jak i ich podzbiorów, a docelowo również przeprowadzanie analiz na zestawach danych. To, czy usługa jest wdrażana w natywnym środowisku siatki, czy w siatce w chmurze, powinno być niewidoczne dla konsumenta. W idealnym przypadku usługa odziedziczy bezpieczeństwo i standardowy interfejs połączenia z obliczeń sieciowych i osiągnie skalowalność oraz dostępność dzięki elastycznej mocy chmury. Prowadząc to badanie skupiamy się w szczególności na kwestiach wydajnościowych. Wykorzystanie dynamicznie przydzielanych zasobów w chmurze może mieć słabą wydajność ze względu na środowisko zwirtualizowane, wewnętrzne szczegóły zachowania pamięci masowej w chmurze oraz dodatkową komunikację sieciową w chmurze/w obrębie chmury. Przewidujemy, że pożądane będzie przeniesienie jak największej ilości prac związanych z

przetwarzaniem (podzestawy i analiza danych) do chmury, aby zminimalizować potrzebę przesyłania danych z chmury do świata zewnętrznego. Takie podejście może również pomóc w obniżeniu kosztów, biorąc pod uwagę, że Amazon pobiera opłaty za przesyłanie danych między pamięcią masową w chmurze a światem zewnętrznym.

Usługi Amazon Cloud

Podsumowujemy ważne cechy usług Amazon EC2, S3 i EBS, z których korzystamy w tej pracy. Usługa Elastic Compute Cloud (EC2) umożliwia klientom żądanie utworzenia co najmniej jednego wystąpienia maszyny wirtualnej (VM), z których każde jest skonfigurowane do uruchamiania obrazu maszyny wirtualnej dostarczonego przez klienta. Użytkownik jest obciążany tylko za czas (zaokrąglony w górę do najbliższej pełnej godziny), w którym instancja EC2 jest uruchomiona. Różne typy instancji są obsługiwane w różnych konfiguracjach (liczba wirtualnych rdzeni, ilość pamięci itp.) i kosztują różne kwoty za godzinę. Użytkownik EC2 może konfigurować wiele obrazów maszyn wirtualnych i uruchamiać wiele instancji każdego z nich, aby tworzyć instancje złożonych scenariuszy rozproszonych, które zawierają różne bloki funkcjonalne, takie jak serwery WWW, serwery aplikacji i serwery baz danych. EC2 zapewnia narzędzia, internetowy interfejs użytkownika i interfejsy API w wielu językach, które ułatwiają tworzenie i zarządzanie obrazami i instancjami. Globalna biblioteka obrazów stanowi punkt wyjścia, od którego można rozpocząć instalację i konfigurację obrazu. Simple Storage Service (S3) zapewnia prosty interfejs usługi sieciowej, którego można używać do przechowywania i pobierania obiektów danych (o rozmiarze do 5 GB) w dowolnym czasie iz dowolnego miejsca w sieci. Dozwolone są tylko operacje zapisu, odczytu i usuwania. Liczba obiektów, które można stworzyć, jest praktycznie nieograniczona. Przestrzeń nazw obiektów jest płaska (nie ma hierarchicznego systemu plików): każdy obiekt danych jest przechowywany w zasobniku i jest pobierany za pomocą unikalnego klucza przypisanego przez programistę. Usługa S3 replikuje każdy obiekt w celu zwiększenia dostępności i niezawodności. Fizyczna lokalizacja obiektów jest niewidoczna dla użytkownika, z wyjątkiem tego, że użytkownik może wybrać strefę geograficzną, w której ma utworzyć obiekt (obecnie zachodnie stany USA, wschodnie stany USA i Europa). O ile obiekty nie zostaną wyraźnie przeniesione, nigdy nie opuszczają regionu, w którym zostały utworzone. Użytkownicy S3 mogą kontrolować, kto ma dostęp do danych, lub alternatywnie udostępniać obiekty wszystkim. Dostęp do danych odbywa się za pośrednictwem interfejsów REST i SOAP zaprojektowanych do pracy z dowolnym internetowym zestawem narzędzi programistycznych. Użytkownicy S3 są obciążani opłatami za przechowywanie i transfery między S3 a światem zewnętrznym. Domyślnym protokołem pobierania jest HTTP; udostępniono interfejs protokołu BitTorrent w celu obniżenia kosztów dystrybucji na dużą skalę. Duże ilości danych (np. duży zbiór danych środowiskowych) można przenieść do S3 za pomocą usługi importu/eksportu opartej na fizycznym dostarczaniu przenośnych jednostek pamięci, która jest szybsza i tańsza niż przesyłanie przez Internet. Elastyczny magazyn bloków (EBS) udostępnia woluminy pamięci masowej na poziomie bloków, które można podłączyć do instancji EC2 jako urządzenie, ale które pozostają niezależnie od okresu eksploatacji konkretnej instancji. Ta usługa jest przydatna dla aplikacji, które wymagają bazy danych, systemu plików lub dostępu do surowego magazynu na poziomie bloków, jak w przypadku przechowywania plików NetCDF. Woluminy EBS mogą mieć rozmiar od 1 do 1000 GB. W tej samej instancji można podłączyć wiele woluminów, co pozwala na striping danych. Woluminy pamięci masowej zachowują się jak surowe, niesformatowane urządzenia blokowe z nazwami urządzeń nadanymi przez użytkownika i interfejsem urządzenia blokowego. Instancje i woluminy muszą znajdować się w tej samej strefie. Woluminy są automatycznie replikowane. EBS używa S3 do przechowywania migawek woluminów w celu ochrony danych w celu zapewnienia długoterminowej trwałości i tworzenia instancji tylu woluminów, ile potrzebuje użytkownik. Wydajność EBS może się różnić, ponieważ wymagany dostęp do sieci i interfejs migawek S3 są ściśle powiązane z konkretną aplikacją, dlatego w każdym przypadku potrzebne są testy porównawcze.

Użytkownik płaci tylko za dane przechowywane w woluminie oraz ilość zużytego S3 na migawki (przestrzeń dyskowa i operacje we/wy). Z perspektywy programisty korzystanie z EBS jest całkowicie przejrzyste, ponieważ wolumeny są postrzegane jako urządzenia blokowe dołączone do instancji EC2. Natomiast funkcje S3 wymagają użycia interfejsów API usług sieci Web. W celu połączenia S3 z NetCDF wybieramy implementację Java bezpłatnie dostępną w kodzie źródłowym.

Wielowymiarowy Standardowy Format Pliku Danych Środowiskowych

Ponieważ mamy tu do czynienia z danymi w formacie NetCDF, krótko opisujemy ten format danych. NetCDF to format i abstrakcja danych, zaimplementowana przez bibliotekę oprogramowania do przechowywania i pobierania wielowymiarowych danych środowiskowych. Opracowany przez UCAR we wczesnych latach 90-tych do zarządzania danymi meteorologicznymi, NetCDF stał się szeroko stosowanym formatem danych dla szerokiej gamy zastosowań w obliczeniach środowiskowych. Implementacja NetCDF zapewnia samoopisujący się i niezależny od komputera format do przedstawiania wielowymiarowych danych naukowych: abstrakcja, biblioteka dostępu i format danych obsługują tworzenie, dostęp i udostępnianie informacji naukowych. Abstrakcja danych NetCDF modeluje zbiór danych naukowych jako zbiór nazwanych zmiennych wielowymiarowych (skalary i tablice bajtów, znaków, liczb całkowitych i liczb zmiennoprzecinkowych) wraz z ich układami współrzędnych i niektórymi nazwanymi właściwościami pomocniczymi. Każda zmienna ma typ, kształt określony przez listę nazwanych wymiarów oraz zestaw innych właściwości opisanych przez pary atrybutów. Interfejs NetCDF umożliwia dostęp do danych poprzez podanie nazwy zmiennej i specyfikacji, która część danych związanych z tą zmienną ma być odczytywana lub zapisywana, a nie poprzez dostęp sekwencyjny i indywidualne odczyty i zapisy. Wymiar to nazwana liczba całkowita używana do określenia kształtu jednej lub więcej zmiennych i zwykle reprezentuje rzeczywisty wymiar fizyczny, taki jak czas, szerokość, długość geograficzna lub poziom atmosferyczny. Zmienna jest tablicą wartości tego samego typu i charakteryzuje się nazwą, typem danych oraz kształtem opisanym przez listę wymiarów. Atrybuty mogą być pojedynczą wartością lub wektorem wartości. Jeden wymiar może być nieograniczony. Zmienna o kształcie zawierającym nieograniczony wymiar może wzrosnąć do dowolnej długości wzdłuż tego wymiaru. Nieograniczony wymiar jest jak numer rekordu w konwencjonalnych plikach; pozwala nam dołączać dane do zmiennych. Implementacje interfejsu oprogramowania NetCDF są dostępne między innymi w językach C, Fortran, Java i MatLab. Pracujemy tutaj z biblioteką NetCDF-Java, frameworkiem 100% Java do odczytu NetCDF i innych formatów plików do Common Data Model (CDM), uogólnieniem modeli danych NetCDF, OpenDAP i HDF5 oraz do zapisu do pliku NetCDF format. Biblioteka NetCDF-Java implementuje również NcML, który umożliwia programiście dodawanie metadanych do zestawów danych CDM, a także tworzenie wirtualnych zestawów danych poprzez agregację. Ta implementacja biblioteki umożliwia dostęp do plików NetCDF za pośrednictwem protokołów sieciowych, takich jak HTTP, a dzięki architekturze wtyczek umożliwia rozwój różnych czytników danych.

Ulepszanie interfejsów API S3

S3 ma dwa ważne ograniczenia, które komplikują jego użycie w przypadku dużych wielowymiarowych zbiorów danych środowiskowych. Pierwszy to maksymalny rozmiar obiektu wynoszący pięć gigabajtów, który jest zbyt mały dla zastosowań środowiskowych. Na przykład aplikacja do prognozowania pogody w czasie rzeczywistym opracowana w DSA/uniParthenope w 2003 r. i nadal działająca do dziś, generuje codziennie 11-gigabajtowy zestaw danych NetCDF tylko z przebiegów przeprowadzonych za pomocą modelu badań i prognoz pogody (WRF) . Drugim jest wymóg, aby obiekt został przeczytany lub napisany w całości. Najbardziej podstawową operacją wykonywaną na wielowymiarowych zbiorach danych środowiskowych jest podzbiór: wyodrębnianie skalarów, tablic i macierzy wzdłuż jednego lub więcej wymiarów. Ta operacja wymaga losowego dostępu do

przechowywanych obiektów. Dlatego zwykłe przechowywanie każdego pliku NetCDF jako obiektu S3 jest nieefektywne i (w przypadku większych plików) również niewykonalne. Z drugiej strony S3 korzysta z wysoce niezawodnej usługi replikacji i lokalizacji, która jest całkowicie przejrzysta dla użytkownika: każdy obiekt jest nazwany unikalnym identyfikatorem zasobu i dostępny za pomocą adresu URL, który można upublicznić. Możliwy jest wielokrotny jednoczesny dostęp do różnych obiektów należących do tego samego segmentu bez widocznej utraty wydajności. Projekt naszego interfejsu Java API z obsługą S3 ma na celu przezwyciężenie ograniczeń S3 przy jednoczesnym zachowaniu wysokiej wydajności i dostępności S3. Jak wspomniano wcześniej, każdy obiekt danych S3 jest identyfikowany za pomocą adresu URL. Chociaż S3 nie obsługuje hierarchicznego systemu plików, ciąg adresu URL S3 może zawierać większość drukowalnych znaków, w tym ukośnik, który jest powszechnie używany do tworzenia struktury nazw folderów i plików. Wewnętrzne usługi Amazon, które używają S3 do przechowywania, często dzielą duże zestawy danych (np. obrazy maszyn wirtualnych) na wiele obiektów S3, przy użyciu zestawu obiektów tworzących zestaw danych wymienionych w pliku manifestu. Podobne podejście stosujemy w przypadku naszych plików NetCDF. W szczególności implementujemy samozarządzający się „obiekt z ramkami", wirtualny, duży (prawdopodobnie ponad pięć gigabajtów) obiekt, identyfikowany po nazwie (nie jest potrzebny plik manifestu) i podzielony na zestaw fizycznych ramek przechowywanych po jednej na obiekt S3 , z nazwami wybranymi do opisu organizacji przypominającej podfolder. Każda ramka jest identyfikowana przez „pod-nazwę" kodującą numer ramki, całkowitą liczbę ramek i rozmiar ramki. Ponieważ nazwy obiektów mają ograniczony rozmiar do 255 znaków, kodujemy liczby w bazie 62, używając wszystkich drukowalnych znaków zgodnych z internetowymi adresami URL w następującej kolejności: 10 cyfr, 26 małych liter i 26 wielkich liter. W celu zwiększenia niezawodności usługi każda ramka jest podpisana cyfrowo przez MD5. Za każdym razem, gdy ramka jest pobierana, jest sprawdzana pod kątem sygnatury i ponownie wysyłana w przypadku wykrycia błędu. Ponieważ oszczędność miejsca w S3 może zarówno obniżyć koszty (dla przechowywania i przesyłania danych), jak i poprawić wydajność, kompresujemy każdą klatkę za pomocą algorytmu zip. S3 obsługuje współbieżny dostęp o wysokiej wydajności, dlatego wszystkie operacje zapisu i odczytu implementujemy przy użyciu wspólnej kolejki blokowania. Nasze API zarządza obiektami w ramkach zarówno w operacjach zapisu, jak i odczytu. Deweloper postrzega każdą operację odczytu i zapisu jako niepodzielną. Gdy obiekt ma być przechowywany w S3 przy użyciu podejścia ramkowego, jest on dzielony na ramki, każda o rozmiarze wcześniej oszacowanym pod kątem najlepszej wydajności, plus ostatnia mniejsza ramka na pozostałe dane. Następnie każda podpisana i skompresowana ramka MD5 jest dodawana do kolejki, która ma zostać zapisana w S3. Kolejka jest zużywana przez pulę wątków roboczych. Liczba wątków zależy od warunków wdrożenia i można ją dostroić w celu uzyskania najlepszej wydajności. Każdy zhread roboczy może wykonywać zarówno operacje zapisu, jak i odczytu. Operacja zapisywania obiektów w ramkach może być blokująca lub nieblokująca. W przypadku blokowania program wywołujący czeka, aż każda ramka zostanie poprawnie zapisana w S3; w przypadku nieblokującym jest powiadamiany o zakończeniu operacji. Deweloper postrzega również operację odczytu obiektów w ramkach jako operację niepodzielną. Ta operacja najpierw wyodrębnia cechy obiektu (rozmiar, całkowita ilość ramek, rozmiar każdej ramki plus pozostały rozmiar ramki) z nazwy przechowywanej ramki. Następnie obiekt jest alokowany w pamięci klienta i wymagane operacje odczytu są przesyłane do kolejki. Wątki robocze jednocześnie sprawdzają integralność ramki za pomocą sygnatury MD5, dekompresują ramkę i umieszczają ją we właściwej pozycji w pamięci. Operacja odczytu obiektu w ramce, podobnie jak operacja zapisu, może być blokująca lub nieblokująca (rys. 26.2). Opracowany przez nas interfejs Java API z obsługą S3 może być używany jako interfejs oprogramowania pośredniego do S3, na którym możemy następnie budować oprogramowanie wyższego poziomu, takie jak nasz interfejs NetCDF Java obsługujący S3.

Włączanie interfejsu Java NetCDF do S3

Biblioteka NetCDF-Java obsługuje dostęp do zbiorów danych środowiskowych przechowywanych zarówno lokalnie, poprzez operacje systemu plików, jak i zdalnie, za pomocą protokołów takich jak HTTP i OpenDAP. Dostęp do danych jest realizowany przez interfejs dostawcy usług wejścia/wyjścia (IOSP). Deweloper może tworzyć niestandardowe IOSP, które implementują metody sprawdzania poprawności plików, otwierania i zamykania plików oraz odczytu danych, w każdym przypadku określając nazwę zmiennej i sekcję do odczytania. IOSP generuje żądania do komponentu RAF (ang. low level random access file), który jest buforowanym zamiennikiem dla homonimicznego komponentu obecnego w pakiecie wejścia/wyjścia Java. Zastosowanie RAF zapewnia znaczny wzrost prędkości dzięki zastosowaniu buforowania. Niestety, pliki RAF nie są ustrukturyzowane w bibliotece NetCDF-Java jako konfigurowalny dostawca usług, więc programiści nie mogą tworzyć i rejestrować własnych komponentów plików o swobodnym dostępie. Format pliku NetCDF jest samoopisujący: dane zawierają osadzone metadane. NetCDF Markup Language (NcML) jest reprezentacją XML metadanych netCDF. NcML jest podobny do sieci netCDF Common Data Form Description Language (CDL), ale używa składni XML. Plik NetCDF może być postrzegany jako folder, w którym każda zmienna może być uważana za plik. Tak więc opis pliku NcML może być traktowany jako agregator zmiennych, wymiarów i atrybutów. W naszym interfejsie Java NetCDF z obsługą S3 używamy reprezentacji pliku NcML jako pliku manifestu, nazwy pliku NetCDF jako nazwy folderu, a każdej zmiennej jako obiektu w ramce: zasadniczo podfolderu, w którym każda zmienna jest przechowywana w ramkach. Tak więc, jeśli plik NetCDF przechowywany w S3 ma nazwę s3://nazwa_zasobnika/ścieżka/nazwa_pliku.ncml, to jego dane są przechowywane w obiektach s3://nazwa_zasobnika/ścieżka/nazwa_pliku/nazwa_zmiennej/nazwa_ramki. Nasza otwarta metoda S3IOServiceProvider współdziała z naszym pakietem Java z rozszerzeniem S3, aby pobrać reprezentację pliku NcML i utworzyć pusty lokalny obraz zdalnego pliku netCDF przy użyciu składnika NcMLReader. W ten sposób metadane są dostępne lokalnie, ale żadne zmienne dane nie są faktycznie pobierane z S3. Aby zaimplementować to zachowanie, tworzymy niestandardowy RAF, który definiuje trzy metody: canOpen zwraca true, jeśli przekazana nazwa pliku może być uznana za poprawny plik; getName zwraca nazwę RAF; a getRaf zwraca bazowy komponent pliku o dostępie swobodnym. Udostępniamy również abstrakcyjny komponent dostawcy RAF, który definiuje referencję RAF i implementuje metodę getRaf. Na koniec zaimplementowaliśmy składnik S3RandomAccessFileProvider, który zwraca ciąg „S3RAFProvider” jako nazwę i sprawdza, czy nazwa pliku zaczyna się od ciągu „s3:”. Metoda canOpen tworzy instancję S3RandomAccessFile i wykonuje potrzebną inicjalizację. S3RandomAccessFile (S3RAF) to kluczowy składnik naszej usługi NetCDF z obsługą S3. Implementuje niskopoziomową interakcję z naszym S3 Java API. Gdy sekcja zmiennej jest odczytywana, S3RAF pobiera tylko potrzebne ramki danych przechowywane w S3 i zapisuje je w tymczasowym pliku lokalnym NetCDF. Jeśli potrzebna jest więcej niż jedna ramka, nasz interfejs Java S3 używa komponentu obiektu w ramkach do równoczesnego ich odczytywania. Mechanizm buforowania gwarantuje, że nie pobierzemy dwukrotnie tej samej ramki. Ta funkcja minimalizuje czas pobierania i zmniejsza liczbę operacji S3 get, a w konsekwencji całkowity koszt (rys. 26.3). Poniższy kod odczytuje sekcję zmiennej z pliku NetCDF przechowywanego w S3:

```
1: String fileName = "s3://12WREKPN1ZEX2RN4SZG2/

wrfout_d01_2009-02-10_00-00-00.nc_1_test";

2: S3ConnectionManager.setIdAndSecret

("12WREKPN1ZEX2RN4SZG2", ". . .");

3: NetcdfFile.registerIOProvider("ucar.unidata.io.s3.

S3IOServiceProvider");
```

4: NetcdfFile.registerRAFProvider("ucar.unidata.io.s3.

S3RandomAccessFileProvider");

5: NetcdfFile ncfile = NetcdfFile.open(testFileName);

6: String section = "1:1:1,1:19:,1:190:1,1:254:1";

7: Array arrayResult = ncfile.findVariable("U").read(range);

W linii 1 definiujemy nazwę pliku. Ciąg „s3://” identyfikuje protokół, umożliwiając komponentowi S3RAFProvider rozpoznanie, że lokalizacja jest dostępna dla S3RAF i że musi utworzyć instancję tego obiektu. Ciąg następujący bezpośrednio po nazwie protokołu to nazwa zasobnika. (Ponieważ nazwy zasobników muszą być unikalne dobrą strategią jest użycie identyfikatora użytkownika S3.) Ostatnią częścią ciągu jest rzeczywista nazwa pliku.

W wierszu 2 używamy komponentu S3ConnectionManager, aby ustawić identyfikator użytkownika i hasło wymagane przez infrastrukturę S3. Składnik S3ConnectionManager umożliwia również programiście skonfigurowanie szczegółów wdrożenia i wydajności, takich jak ścieżka pliku tymczasowego, rozmiar kolejek odczytu i zapisu oraz liczba wątków roboczych.

W wierszu 3 rejestrujemy S3IOServiceProvider przy użyciu standardowego interfejsu NetCDF Java, podczas gdy w wierszu 4 rejestrujemy S3RAFProvider, który, jak wspomniano powyżej, ulepsza standardowy interfejs NetCDF-Java, aby zaimplementować całkowicie przezroczysty dostęp do zbiorów danych przechowywanych w S3.

W linii 5 otwieramy plik, używając tej samej składni, co w przypadku operacji lokalnej. Linia 6 określa, że chcemy odczytać tylko jeden krok czasowy całej zmiennej dwuwymiarowej. Odczytujemy zmienną w wierszu 7, pobierając odwołanie do obiektu Array do danych w taki sam sposób, jak w przypadku lokalnego dostępu do danych. Zauważ, że podstawowa złożoność chmury jest całkowicie ukryta.

Hybrydyzacja w chmurze i sieci: usługa NetCDF

Usługa NetCDF opracowana przez Montella, Agrillo, Mastrangelo i Menna (2008) jest usługą internetową opartą na GT4. Wykorzystuje przydatne funkcje GT4 i przechwytuje wiele wcześniejszych doświadczeń w dostarczaniu danych środowiskowych za pomocą narzędzi gridowych. Usługa integruje wiele źródeł danych i trybów interakcji z serwerem danych, interfejsy z usługą indeksowania w celu umożliwienia wykrywania oraz obsługuje wbudowane przetwarzanie danych. Wreszcie, jest przeznaczony do pracy w hybrydowym środowisku chmury/sieci.

Architektura usługi NetCDF

Usługa NetCDF zapewnia swoim klientom dostęp do zasobów: abstrakcyjnych reprezentacji obiektów danych, które są całkowicie oddzielone od bazowego przechowywania danych związanego z obiektami danych. Łącznik łączy zasób usługi NetCDF z określonym podstawowym systemem przechowywania danych. Dostępne konektory obejmują konektor plików NetCDF, który przy użyciu naszego interfejsu NetCDF Java z rozszerzeniem S3 może obsługiwać pliki lokalne, pliki obsługiwane przez DODS, pliki obsługiwane przez HTTP i pliki przechowywane w S3; złącze GDS, które może obsługiwać pliki Grib i Grads obsługiwane przez serwer danych Grads; oraz złącze Hyrax dla serwerów opartych na OpenDAP Hyrax. Opracowujemy również złącze instrumentalne jako bezpośredni interfejs do instrumentów akwizycji danych w oparciu o nasz Abstract Instrument Framework. Głównym celem konektora jest wysyłanie żądań do różnych serwerów danych i konwersja wszystkich odpowiedzi na zestawy danych NetCDF . Gdy żądany podzbiór zestawu danych zostanie dostarczony przez łącznik danych i zapisany

lokalnie, użytkownik może przetwarzać ten podzbiór za pomocą oprogramowania lokalnego. Ta funkcja jest implementowana przy użyciu innej w pełni konfigurowalnej wtyczki. Dzięki podejściu fabryka/instancja każdy klient usługi sieciowej ma do czynienia z własnymi danymi w tymczasowym prywatnym obszarze przechowywania fizycznie blisko usługi sieciowej. Misją komponentu łącznika procesora jest sprzęganie innych niż proces procesorów zestawu danych NetCDF, realizujących standardowy sposób wprowadzania danych, wysyłania zadań przetwarzania i wyprowadzania danych . Dostępne są dwa złącza procesora. Złącze procesora GraDS zapewnia dostęp do danych NetCDF w postaci danych siatkowych. Konsument może wysłać do łącznika procesora złożone sekwencje poleceń przy użyciu wspólnego interfejsu Java lub bezpośrednio za pomocą skryptów GraDS. Złącze procesora GrADS jest oparte na interfejsie GrADSj Java, który opracowaliśmy w poprzednich pracach . Łącznik NetCDF Operator to interfejs Java do pakietu oprogramowania o tej samej nazwie. Operatory netCDF lub NCO to zestaw samodzielnych programów wiersza poleceń, z których każdy przyjmuje pliki netCDF jako dane wejściowe, operują na tych plikach (np. wyprowadzają nowe dane, obliczają średnie, wyodrębniają hiperpłyty, manipulują metadanymi) i tworzą plik wyjściowy netCDF . NCO przede wszystkim pomaga w manipulacji i analizie danych naukowych podzielonych na siatkę. Styl NCO z jednym poleceniem pozwala użytkownikom interaktywnie manipulować plikami i analizować je za pomocą prostych skryptów, które pozwalają uniknąć pewnych kosztów (i mocy) środowisk programowania wyższego poziomu. Podobnie jak w przypadku złącza procesora GraDS, konsument usługi sieciowej wchodzi w interakcję ze złączem procesora NCO za pomocą prostego interfejsu Java lub bezpośrednio ze skryptami podobnymi do powłoki. Ponieważ wewnętrznie GrADSj, złącze procesora NCO wykorzystuje AbstractExecutionFramework (AEF), który opracowaliśmy w celu zarządzania wykonywaniem oprogramowania poza procesem ze środowiska Java jest standardową modą na wysokim poziomie (Montella & Agrillo, 2009b). Po wyodrębnieniu i przetworzeniu podzbioru danych głównym aktorem potoku danych jest łącznik przesyłania danych. Ten składnik podobny do wtyczki umożliwia programiście dostosowanie sposobu udostępniania wybranych danych użytkownikowi. Wszystkie łączniki przesyłania współdzielą system zarządzania pamięcią podręczną i automatyczne publikowanie opisów danych w usłudze indeksowania. (Te funkcje są stosowane automatycznie tylko wtedy, gdy nie ma problemów z prywatnością). Ogólnie dane podzbioru z publicznie dostępnego zestawu danych są nadal publiczne, ale użytkownik może ustawić wynik podzbioru jako prywatny; z kolei wynik przetwarzania jest domyślnie prywatny, ale użytkownik może go zadeklarować jako publiczny.

Procesy buforowania i automatycznej publikacji działają na dwóch rodzajach zestawów danych w ten sam sposób. Ponieważ każdy plik NetCDF jest w pełni opisany przez jego metadane, każdy zestaw danych można jednoznacznie zidentyfikować za pomocą skrótu MD5 tych metadanych. W ten sposób podpisujemy każdy zestaw danych wygenerowany przez podzbiór lub operację przetwarzania i kopiujemy go do obszaru pamięci podręcznej. Następnie przy każdym żądaniu pliku NetCDF najpierw oceniamy sygnaturę metadanych MD5 i sprawdzamy pamięć podręczną. W przypadku braku pamięci podręcznej zestaw danych jest wymagany do podzbioru lub przesyłany do przetwarzania. Po trafieniu w pamięć podręczną zestaw danych jest kopiowany z obszaru pamięci podręcznej do tymczasowego obszaru pamięci żądającego, a ponadto komponent menedżera pamięci podręcznej zestawu danych NetCDF zwiększa wskaźnik wykorzystania wybranego zestawu danych. Jeśli ten indeks przekroczy wybrany próg, zestaw danych jest promowany jako przechowywany zestaw danych. Menedżer zasobów usługi sieci Web eksploruje zasoby i automatycznie anonsuje je w usłudze indeksowania. Składnik menedżera pamięci podręcznej zestawu danych NetCDF okresowo bada pamięć podręczną, zmniejszając licznik użycia każdego buforowanego zestawu danych NetCDF. Zestaw danych jest usuwany, jeśli jego liczba osiągnie zero, w celu zaoszczędzenia lokalnej pamięci masowej. Ten proces sprawdzania użycia jest wykonywany nawet na promowanych zestawach danych. Rzeczywista

sekwencja priorytetów urządzenia pamięci masowej, uszeregowana od najczęściej używanych do najmniej, to: lokalny system plików, EBS, S3, a następnie usuwana. Zasady używane do zarządzania przenoszeniem danych z tych urządzeń pamięci masowej są w pełni konfigurowalne i oparte na liczbie żądań na jednostkę czasu pomnożoną przez zapotrzebowanie na miejsce dla konkretnego urządzenia. Łącznik transferowy działa jako główny składnik dostarczania danych implementujący w jaki sposób użytkownik może uzyskać dostęp do wyniku usługi sieciowej. Domyślnym łącznikiem transferu jest TransferServiceConnector. W ten sposób przedstawiamy wynik przez odniesienie do punktu końcowego (EPR) do tymczasowego zasobu zarządzanego przez TransferService. Ta usługa jest opakowaniem usługi GridFTP. TransferService korzysta z Infrastruktury Bezpieczeństwa Grid i działa w wydajny i efektywny sposób, zapewniając bezpieczny transfer danych. Inne dostępne łączniki transferu obejmują HTTPTransferConnector i DODSTransferConnetor odpowiednie dla publicznie dostępnych zestawów danych wyników. Wreszcie S3TransferConnector przechowuje wyniki w S3, a następnie użytkownik może uzyskać do nich bezpośredni dostęp.

Scenariusze wdrażania usługi NetCDF

Usługa NetCDF reprezentuje składnik agregatora siatki dla hostowanych w chmurze wielowymiarowych zasobów danych środowiskowych. Badamy trzy różne scenariusze wdrażania, w których usługa NetCDF jest wdrażana w różny sposób:

1. na komputerze poza chmurą (Grid+S3);

2. na instancji EC2 (Cloud); lub

3. w konfiguracji proxy, w której usługa działa na komputerze poza chmurą i automatycznie uruchamia jedną lub więcej instancji EC2 do zarządzania operacjami na zbiorach danych (Hybrid)

W scenariuszu wdrażania Grid+S3 samodzielna usługa NetCDF działa na serwerze zewnętrznym względem chmury. Ten serwer musi być wystarczająco wydajny i mieć wystarczającą ilość miejsca do przechowywania, aby obsługiwać operacje na podzbiorach i przetwarzaniu. Dane mogą być hostowane lokalnie, przez serwery DODS w zabezpieczonych sieciach prywatnych i/lub w usługach chmurowych opartych na S3. Dzięki komponentowi AbstractExecutionFramework oprogramowanie przetwarzające (GrADS i NCO) może działać jako zadania przesyłane do lokalnej kolejki i wykonywane w klastrze obliczeniowym o wysokiej wydajności. W tym scenariuszu używamy dostawcy chmury do przechowywania danych, tak jak w przypadku lokalnie przechowywanych dużych zestawów danych i S3 przechowywanych automatycznie promowanych w pamięci podręcznej zestawów danych. W scenariuszu Cloud usługa NetCDF jest wdrażana na instancji EC2, która jest bezpośrednio dostępna z zewnątrz chmury za pomocą usługi Elastic IP firmy Amazon. Dane mogą być przechowywane w EBS i dostępne lokalnie przez instancję, podczas gdy automatycznie promowane buforowane zestawy danych mogą być przechowywane w EBS i/lub S3, w zależności od częstotliwości użytkowania. Instancja EC2 musi zapewnić wystarczającą moc obliczeniową, aby zaspokoić potrzeby procesora zbioru danych. Zaletą tego rodzaju wdrożenia jest szybki dostęp do elastycznie przechowywanych zbiorów danych za pomocą EBS (szybszy) i S3 (wolniejszy, ale dostępny spoza chmury). Główną wadą tego podejścia jest to, że użytkownik jest obciążany kosztami stale działającej instancji EC2 i konieczności posiadania przypisanego elastycznego adresu IP. Scenariusz hybrydowy ma najwyższy poziom hybrydyzacji sieci/chmury. Tutaj wdrażamy usługę NetCDF na prawdziwym komputerze poza infrastrukturą chmury. Usługa ta działa jako proxy dla innej instancji usługi działającej (jako instancja EC2) w infrastrukturze chmury. Gdy usługa NetCDF otrzyma żądanie, sprawdza, czy ta instancja EC2 jest już uruchomiona. Jeśli nie ma lub jeśli działająca jest zbyt zajęta (obciążenie wirtualnego procesora przekracza próg), tworzona jest nowa instancja. Jeśli dane, do których chcesz uzyskać dostęp, są hostowane przez EBS, a aktywna instancja EC2 jest podłączona do EBS, w którym przechowywane są

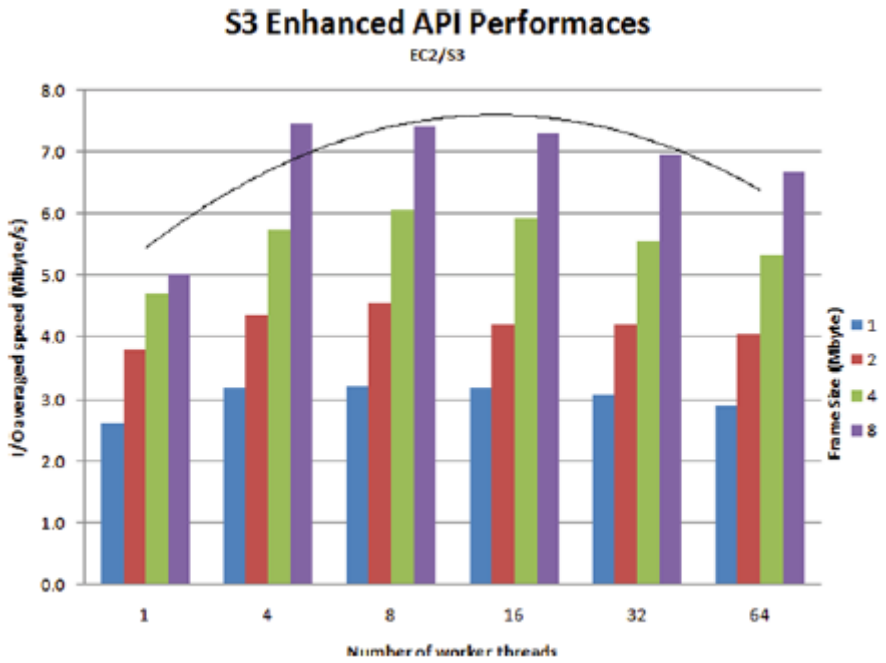
dane. Ponieważ wolumen EBS może być jednocześnie połączony tylko z jedną instancją EC2, żądanie musi być skierowane właśnie do tej instancji. W tym scenariuszu konsument wchodzi w bezpośrednią interakcję tylko z usługą NetCDF uruchomioną na rzeczywistym komputerze; interakcja z usługą NetCDF uruchomioną na elastycznie przydzielonym zasobie obliczeniowym jest całkowicie przejrzysta. Główną zaletą tego podejścia jest to, że (a) użytkownik płaci za uruchamianie AMI tylko podczas przetwarzania zbiorów danych oraz (b) nie ma potrzeby przydzielania elastycznego adresu IP. Co więcej, ilość danych, które muszą zostać przesłane z chmury do sieci, jest ograniczona, ponieważ większość przetwarzania danych odbywa się wewnątrz chmury. Ponadto elastyczność zapewnia skalowalność, a infrastruktura chmury zapewnia przewidywalną jakość usług, dostępność danych, tworzenie kopii zapasowych i odtwarzanie po awarii. Wreszcie maszyna gridowa obsługująca usługę NetCDF nie potrzebuje ogromnej mocy obliczeniowej ani przestrzeni dyskowej, ponieważ działa jedynie jako proxy zapewniające przejrzysty interfejs do infrastruktury chmury. Główną wadą jest złożoność wdrożenia i zwiększona liczba warstw interfejsu.

Ocena wydajności

Wszystkie komponenty oprogramowania opracowane w ramach tej pracy to wysokiej jakości prototypy, gotowe do zastosowania w rzeczywistych warunkach testowych, a nawet do użytku produkcyjnego. Aby ocenić wydajność różnych metod wdrażania, zmierzyliśmy wydajność w trzech różnych scenariuszach: odczyt i zapis danych z S3 przy użyciu obiektu w ramce zaimplementowanego w naszym interfejsie Java; odczyt danych NetCDF z S3 porównujący wydajność wewnątrz i na zewnątrz chmury; oraz zestaw danych NetCDF służący do porównywania pamięci EBS i S3.

Wybór parametrów dla ulepszonego interfejsu Java S3

Musimy ocenić wydajność we/wy interfejsu Java z obsługą S3, aby określić optymalną konfigurację frameworka. Dwa parametry, którymi może sterować programista, to rozmiar ramki i liczba współbieżnych wątków używanych jako elementy robocze przez składniki obiektów w ramkach. (Trzeci parametr, rozmiar kolejki blokującej, został empirycznie oceniony jako czterokrotność liczby wątków roboczych). Wybór rozmiaru ramki jest krytyczny, ponieważ po ustawieniu przechowywany obiekt w ramce musi zostać ponownie przesłany, aby go zmienić. W przeciwieństwie do tego liczba wątków roboczych może się różnić nawet podczas działania aplikacji i potencjalnie może być dostosowywana automatycznie na podstawie zaobserwowanego przeciążenia sieci i wykorzystania procesora przez komputer. Do oceny wydajności przesyłania i pobierania użyliśmy pliku NetCDF o pojemności 130 MB dla różnych rozmiarów ramek (1, 2, 4 i 8 MB) oraz liczby wątków (1–64 wątków). W każdym przypadku przeprowadzaliśmy eksperyment 100 razy i uśrednialiśmy wyniki. Zastosowaliśmy podejście najgorszego przypadku, przechowując plik w strefie USA i uzyskując do niego dostęp z Europy.



Najlepszą wydajność zarówno odczytu, jak i zapisu uzyskuje się przy użyciu ramek o wielkości 8 MB. Najlepsza liczba współbieżnych wątków różni się w zależności od operacji: przy zapisie (wysyłaniu) najlepszą wydajność osiąga się od 8 do 16 wątków roboczych, natomiast przy odczytywaniu (pobieraniu) maksymalna liczba testowanych wątków roboczych, czyli 64, była najlepsza. Zastosowaliśmy te lekcje w ulepszonej wersji naszego interfejsu Java z rozszerzeniem S3, który korzysta z narzędzia do oceny wydajności na żywo, aby zmieniać liczbę wątków roboczych w czasie.

Ocena interfejsów NetCDF Java z obsługą S3 i EBS

Nasz drugi zestaw eksperymentów porównuje wydajność naszego interfejsu NetCDF Java z obsługą S3 z analogicznymi operacjami przy użyciu EBS. Używamy czterowymiarowego pliku NetCDF o rozmiarze ∼11 GB wyprodukowanego przez model WRF na siatce o wymiarach 256 × 192 komórek i 28 poziomach pionowych. Plik ten zawiera sześć dni (144 godz.) 111 zmiennych: 14 jednowymiarowych, zmiennych w czasie; 10 dwuwymiarowych, różniących się czasem i poziomem; 72 trójwymiarowy, zmienny w czasie, szerokości i długości geograficznej; oraz 15 czterowymiarowych, różniących się czasem, poziomem, szerokością i długością geograficzną. Definiujemy pięć oddzielnych testów, odpowiadających odczytaniu odpowiednio pierwszego (5,5 MB), 24 (132 MB), 48 (264 MB), 72 (396 MB) i 144 h (792 MB) jednego cztero- zmienna wymiarowa. Zmienna podzbioru jest składową prędkości wiatru z zachodu na wschód U(X, Y, Z, T), gdzie X, Y i Z to wymiary przestrzenne, a T to wymiar czasowy, w kolejności T, Z, Y , X, co oznacza, że plik dyskowy utworzony dla tej zmiennej zawiera dane dla wszystkich okresów dla (X = 0, Y = 0, Z = 0), a następnie dane dla wszystkich okresów dla (X = 0, Y = 0, Z = 1) i tak dalej. Tak więc żądanie 144 h zmiennej obejmuje pojedynczą ciągłą sekwencję bajtów, podczas gdy żądanie 1 h zmiennej obejmuje wiele małych odczytów. Tak jak poprzednio, każdy test przeprowadzamy 100 razy i uśredniamy wyniki. Przeprowadziliśmy zestaw testów w czterech konfiguracjach, w których operacja podzbioru jest wykonywana w różny sposób na:

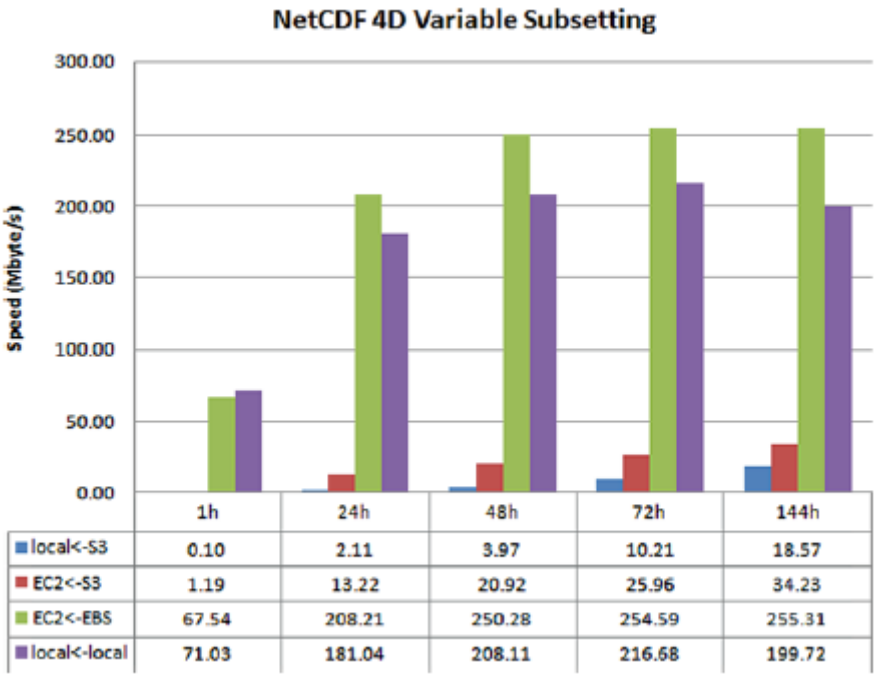
1. komputer w Europie, poza chmurą, z danymi na S3 w USA;

2. maszyna wirtualna EC2 z danymi na S3 w tej samej strefie;

3. maszyna wirtualna EC2 z danymi na dołączonym wolumenie EBS; oraz

4. serwer poza chmurą (serwer oparty na czterordzeniowym Xeon Linux) z danymi na lokalnym dysku tego komputera.

Jak pokazano na rysunku, wydajność wzrasta we wszystkich przypadkach z podzestawem okresu czasu, a tym samym z rozmiarem odczytu.



	1h	24h	48h	72h	144h
local<-S3	0.10	2.11	3.97	10.21	18.57
EC2<-S3	1.19	13.22	20.92	25.96	34.23
EC2<-EBS	67.54	208.21	250.28	254.59	255.31
local<-local	71.03	181.04	208.11	216.68	199.72

Widzimy również, że wydajność różni się znacznie w czterech różnych konfiguracjach. Dostęp do S3 z EC2 (EC2←S3) jest szybszy niż w przypadku dostępu do S3 spoza chmury (local←S3), przy czym różnica jest proporcjonalnie większa dla mniejszych odczytów. Ten wynik sugeruje, że mogą być korzyści z wykonywania operacji na podzbiorach w chmurze, a następnie przenoszenia tylko podzbioru danych do zdalnego klienta. Widzimy również, że wydajność EBS (EC2←EBS) jest znacznie lepsza niż wydajność S3 - w rzeczywistości lepsza niż dostęp dysk lokalny (lokalny←lokalny). Z tego wyniku możemy wywnioskować, że dane powinny zawsze znajdować się w EBS. Jednak przy wyborze między pamięcią S3 i EBS należy wziąć pod uwagę inne subtelne kwestie. Jeśli dostęp do danych ma być możliwy tylko z maszyny wirtualnej EC2, EBS jest najlepszym wyborem. Z drugiej strony, S3 pozwala na bezpośredni dostęp do danych (na przykład przy użyciu naszego interfejsu Java NetCDF z ulepszoną wersją S3) spoza chmury i pozwala na jednoczesne uzyskiwanie wielu dostępów. Należy również wziąć pod uwagę koszty: uruchamianie instancji na maszynach EC2 może, ale nie musi, być opłacalne, w zależności od obciążenia. Dynamiczne tworzenie instancji maszyn wirtualnych tylko wtedy, gdy jest to potrzebne do celów podzbiorów, analizy i przesyłania danych, może być najlepszym wyborem. Wracamy do tych kwestii poniżej.

Ocena wydajności usług NetCDF

Nasze ostatnie eksperymenty mają na celu ocenę naszej usługi NetCDF z perspektywy zarówno kosztów, jak i wydajności. Na potrzeby tych badań opracowaliśmy Amazon Web Service Simulator (AWSS), platformę Java, która symuluje zachowanie komponentów Amazon EC2, EBS i S3. Przy określonym dostępie do danych i profilach cenowych ten symulator generuje prognozę wydajności i kosztów. Aby ocenić dokładność symulatora, porównaliśmy jego przewidywany koszt z wynikami uzyskanymi, w których uruchomiliśmy instancje hostingu danych i maszyn wirtualnych bez

jakiejkolwiek aplikacji w chmurze. Uzyskaliśmy profile kosztów porównywalne z prostym miesięcznym symulatorem Amazon. Oceniliśmy trzy scenariusze. W Grid+S3 usługa NetCDF jest hostowana na komputerze poza chmurą i uzyskuje dostęp do danych hostowanych na S3 za pośrednictwem protokołu S3 opartego na Amazon HTTP. W chmurze usługa NetCDF jest hostowana na „standard.extralarge" instancji EC2 (15 GB pamięci, 8 jednostek obliczeniowych EC2, 1690 GB pamięci lokalnej instancji, platforma 64-bitowa), która działa nieprzerwanie przez cały miesiąc. Dane są hostowane na 15 jednoterabajtowych woluminach EBS dołączonych do tej instancji EC2. W Hybrid usługa NetCDF jest hostowana na instancji EC2 z taką samą konfiguracją jak w Cloud. Jednak ta instancja nie jest uruchamiana w sposób ciągły, ale zamiast tego jest tworzona po nadejściu żądania, a następnie zamykana po dwóch godzinach, chyba że w tym okresie nadejdzie dodatkowe żądanie. Ponieważ chmura i hybryda generują dane podzbioru na instancjach EC2, dane te muszą być przesyłane z chmury do świata zewnętrznego. Zmierzyliśmy wydajność dla trzech różnych protokołów — HTTP (~2,5 MB/s), gridFTP (~16 MB/s) i scp (~0,56 MB/s) - i wykorzystaliśmy te liczby w naszych symulacjach.

Prezentujemy wyniki dla poniższej konfiguracji i obciążenia. Zakładamy 15 TB danych środowiskowych, które są już przechowywane w chmurze; w związku z tym nie są ponoszone żadne koszty przesyłania danych. Bierzemy pod uwagę 30-dniowy czas trwania (w tym czasie ponoszone są koszty przechowywania danych), przy czym żądania różnych podzbiorów tego samego czterowymiarowego zestawu danych są następujące:

• Dni 0–2: Brak żądań.

• Dni 3–5: 144 żądania, każde na 1 godzinę podzbioru (tj. 5,5 MB).

• Dni 6–8: 144 żądania, każde dla podzbioru 24-godzinnego (132 MB).

• Dni 9–11: 144 żądania, każde dla 48-godzinnego podzbioru (264 MB).

• Dni 12–14: 144 żądania, każde na 72-godzinny podzbiór (396 MB).

• Dni 15–17: 144 żądania, każde na 144 godz. podzbiór (792 MB).

• Dni 18–29: Brak próśb.

Żądania przychodzą zgodnie z rozkładem normalnym ze średnią 0,5 godz.; łącznie 280 GB w ciągu 30 dni. Zakładamy najgorszą sytuację z komputerem w Europie uzyskującym dostęp do zasobów AWS hostowanych w strefie zachodniej Stanów Zjednoczonych. Korzystamy z kosztów Amazon według stanu na marzec 2010 r. Grid+S3 jest najdroższy ze względu na użycie S3 i związane z nim koszty przechowywania i transferu danych. Jego wydajność jest dobra, szczególnie w przypadku większych podzbiorów, ponieważ interfejs S3 Enhanced Java, który służy do przenoszenia danych poza chmurę, działa dobrze przy dużych wyborach danych . Chmura jest tańsza niż Grid+S3 i ma lepszą wydajność niż Grid+S3 (w przypadku korzystania z GridFTP do transferów zewnętrznych), z wyjątkiem przypadku 144 h (bez podzbiorów). Hybryda jest najbardziej opłacalna, ponieważ wykorzystuje EBS do hostowania danych i tworzy instancje EC2 tylko wtedy, gdy jest to wymagane. Korzystanie z GridFTP do przesyłania danych poza chmurą jest bardziej ekonomiczne, ponieważ zwiększa się rozmiar problemu (przechowywanie, podzbiór i przesyłanie danych). Grid+S3 jest drogi i powolny dla małych podzbiorów, ale staje się konkurencyjny, gdy dane podzbioru rosną. Jednym z czynników nieuwzględnionych w tych wynikach jest to, co się dzieje, gdy wielu zdalnych klientów żąda danych w tym samym czasie. W takiej sytuacji Grid+S3 może stać się bardziej konkurencyjny, ponieważ fakt, że wolumen EBS może być dołączony tylko do jednej instancji EC2 w czasie, ogranicza wydajność Hybrid. Pouczające jest również porównanie tych wyników z tym, co wiemy o ESG. Na początku 2010 r. systemy ESG przechowują około 150 terabajtów danych symulacji klimatu. W 2009 roku jeden z dwóch głównych serwerów działających

w tym czasie (w NCAR) dostarczył łącznie 112 TB w odpowiedzi na 170 886 żądań, co daje średnio 20 żądań na godzinę i 654 MB na plik. Nasza eksperymentalna konfiguracja miała 15 TB (około 1/10 całkowitego rozmiaru) i zwróciła łącznie 280 GB w odpowiedzi na 720 żądań, co daje średnio 1 żądanie na godzinę i 388 MB na plik. Musimy powtórzyć nasze symulacje przy rzeczywistym obciążeniu ESG, ale szacunki na podstawie Rys. 26.11 sugerują, że hostowanie ESG na Amazon kosztuje około 20 000 USD miesięcznie.

Wnioski i kierunki na przyszłość

Staraliśmy się odpowiedzieć na pytanie, czy wykorzystanie usług chmury Amazon do hostowania dużych zbiorów danych środowiskowych jest wykonalne, opłacalne i wydajne. Wierzymy, że odpowiedź na to pytanie brzmi „tak", choć z pewnymi zastrzeżeniami. Aby ocenić wykonalność, opracowaliśmy usługę NetCDF, która wykorzystuje kombinację usług zewnętrznych i opartych na chmurze, aby umożliwić zdalny dostęp do danych przechowywanych w pamięci masowej Amazon S3. Podstawą tej usługi jest NetCDF Java API z obsługą S3, który wykorzystuje abstrakcję obiektów w ramkach, aby umożliwić prawie swobodny dostęp do odczytu i zapisu do zestawów danych NetCDF przechowywanych w obiektach S3. Sygnatury MD5 i kompresja zwiększają niezawodność i wydajność. Interfejs NetCDF-Java z obsługą S3 umożliwia programiście dostęp zarówno do plików lokalnych, jak i do danych NetCDF przechowywanych w S3 (lub przechowywanych w EBS) w identyczny sposób. Korzystanie z magazynu Amazon jest przejrzyste z jedyną różnicą polegającą na konieczności podania danych dostępowych. Ogólnie rzecz biorąc, to doświadczenie prowadzi nas do wniosku, że ESG oparty na Amazon jest wykonalny. Usługa NetCDF jest produktem naszych wcześniejszych, znaczących prac nad usługami sieciowymi dostawcy danych środowiskowych (opartych na GT4). Usługa ta jest wysoce modułowa i oparta na architekturze typu plug in. Wielowymiarowe zbiory danych środowiskowych są ujawniane jako zasoby, które ponadto są automatycznie ogłaszane za pośrednictwem usługi indeksowania. Konsument usługi WWW wchodzi w interakcję z prywatnym, dynamicznie przydzielanym wystąpieniem zasobu usługi. Dostawca operacji zezwala na wybór, podzbiór, przetwarzanie i przesyłanie danych. Każda funkcja jest implementowana przez składnik dostawcy, który umożliwia ulepszanie, rozbudowę i dostosowywanie. Usługę można wdrożyć na trzy główne sposoby: samodzielnie na maszynie należącej do sieci obliczeniowej w celu dystrybucji danych hostowanych lokalnie, na zabezpieczonych serwerach niestandardowych lub przy użyciu usługi S3; samodzielna praca na instancji maszyny wirtualnej działającej w ekosystemie EC2 udostępniająca dane przechowywane na EBS lub S3; oraz w trybie proxy działającym na zewnętrznym komputerze, który działa jako proxy do obsługi instancji zarządzających zasobami na dynamicznie przydzielanych instancjach EC2. Aby ocenić wydajność, przeprowadziliśmy szczegółowe badania eksperymentalne wydajności dostępu do danych dla naszej usługi NetCDF w tych różnych konfiguracjach. Aby oszacować koszty, opracowaliśmy i zastosowaliśmy prosty symulator chmury. Na podstawie tych wyników uważamy, że rozwiązanie hybrydowe, reprezentowane przez trzeci scenariusz wdrażania usługi NetCDF, Hybrid, zapewnia najlepszą równowagę między zasobami zewnętrznymi i hostowanymi w chmurze. Ta strategia minimalizuje koszty, ponieważ instancje EC2 są uruchamiane tylko wtedy, gdy jest to wymagane. Kolejną redukcję kosztów uzyskuje się, ponieważ nie ma potrzeby stosowania elastycznego adresu IP: wystąpienie usługi uruchomione na maszynie wirtualnej łączy się z tym na rzeczywistej maszynie identyfikowanej przez w pełni kwalifikowaną nazwę domeny. Dwa inne źródła kosztów to operacje put and get danych na S3 oraz dane przesyłane przez granicę infrastruktury chmury. W Hybrid większość przetwarzania danych odbywa się w chmurze, a zatem konsument żądający podzbiorów lub wyników analizy pobiera mniej danych. Mniej transferu danych oznacza również lepszą wydajność. Zastosowanie EBS zamiast S3 zapewnia zwiększoną wydajność dostępu do danych. Chociaż takie podejście pozwala na wykorzystanie S3 jako efektywnego sposobu na przeprowadzanie dużych zbiorów danych w chmurze w efektywny i wydajny sposób, wykorzystując

współbieżny dostęp do obiektów S3 za pomocą opracowanego przez nas interfejsu Java API. Wreszcie, ponieważ zapotrzebowanie na pamięć masową jest skalowane przez elastyczne podejście obliczeniowe typowe dla infrastruktury chmurowej, nawet moc obliczeniowa przetwarzania skaluje się wraz z potrzebami dzięki możliwości wystąpienia tylu instancji EC2 obsługujących usługi NetCDF, ile potrzeba. Bardziej szczegółowe porównanie kosztów i wydajności dla ESG czeka na udostępnienie szczegółowych dzienników dostępowych ESG. Wydaje się jednak, że koszt utrzymania obecnych zasobów ESG na Amazon i reagowania na bieżące obciążenia może wynosić 20 000 USD miesięcznie. Ustalenie aktualnych kosztów hostingu danych ESG byłoby trudne, ale biorąc pod uwagę wszystkie koszty, nie może się one zbytnio różnić. ESG przygotowuje się teraz do następnej generacji modeli klimatycznych, które będą generować petabajty mocy wyjściowej. Do oceny przydatności komercyjnych dostawców chmury, takich jak Amazon, dla takich zbiorów danych potrzebne będzie szczegółowe porównanie wydajności i kosztów – zadanie, do którego nasz symulator jest dobrze przygotowany. Niestety porównanie to jest obecnie trudne lub niemożliwe do wykonania ze względu na brak danych na temat przyszłych konfiguracji chmury i kosztów oraz przyszłych obciążeń klientów. Zauważamy jednak, że oczekuje się, że analiza po stronie serwera będzie stawać się coraz ważniejsza wraz ze wzrostem rozmiarów danych, a infrastruktury, takie jak Amazon, są dobrze zaprojektowane do intensywnej obliczeniowej analizy dużych ilości danych. Praca ta jest wynikiem rocznego prototypowania i eksperymentów z wykorzystaniem hybrydowych technologii gridowych i przetwarzania w chmurze do elastycznego przechowywania, przetwarzania i dostarczania danych środowiskowych. Usługa NetCDF integruje wiele warstw oprogramowania w celu określenia wygodnego scenariusza wdrożenia hybrydowego. W następnym kroku zastosujemy tę technologię do bardziej realistycznych scenariuszy prognozy pogody i symulacji klimatu. W tym procesie poprawimy stabilność poszczególnych komponentów i rozwiniemy kolejne cechy zidentyfikowane przez doświadczenie.