Technologie wymagające dużej ilości danych do przetwarzania w chmurze

Wstęp

W wyniku nieustannej eksplozji informacji wiele organizacji tonie w danych, a wynikająca z tego „luka w danych" lub niemożność przetwarzania tych informacji i ich efektywnego wykorzystania narasta w alarmującym tempie. Przetwarzanie intensywnie korzystające z danych reprezentuje nowy paradygmat obliczeniowy, który może rozwiązać lukę w danych za pomocą skalowalnego przetwarzania równoległego, aby umożliwić rządom, organizacjom komercyjnym i środowiskom badawczym przetwarzanie ogromnych ilości danych i wdrażanie aplikacji, które wcześniej uważano za niepraktyczne lub niewykonalne. Przetwarzanie w chmurze daje organizacjom o ograniczonych zasobach wewnętrznych możliwość wdrażania aplikacji obliczeniowych na dużą skalę, intensywnie korzystających z danych, w opłacalny sposób. Podstawowe wyzwania związane z przetwarzaniem intensywnie przetwarzającym dane to zarządzanie i przetwarzanie wykładniczo rosnących ilości danych, znaczne skrócenie powiązanych cykli analizy danych w celu obsługi praktycznych, aktualnych aplikacji oraz opracowywanie nowych algorytmów, które można skalować w celu wyszukiwania i przetwarzania ogromnych ilości danych. Naukowcy z LexisNexis uważają, że odpowiedzią na te wyzwania jest skalowalna, zintegrowana architektura sprzętu i oprogramowania systemów komputerowych zaprojektowana do równoległego przetwarzania aplikacji obliczeniowych intensywnie korzystających z danych. Ten część analizuje wyzwania związane z intensywnym przetwarzaniem danych i oferuje dogłębne porównanie dostępnych na rynku architektur systemów, w tym superkomputera LexisNexis Data Analytics Supercomputer (DAS) zwanego również klastrem LexisNexis High-Performance Computing (HPCC) oraz Hadoop, Implementacja open source oparta na architekturze Google MapReduce. Przetwarzanie w chmurze kładzie nacisk na możliwość skalowania zasobów obliczeniowych zgodnie z potrzebami bez dużych wstępnych inwestycji w infrastrukturę i związanych z tym bieżących kosztów operacyjnych. Usługi przetwarzania w chmurze są zazwyczaj podzielone na trzy modele:

(1) Infrastruktura jako usługa (IaaS). Usługa obejmuje dostarczanie sprzętu i oprogramowania do przetwarzania, przechowywania danych, sieci i wszelkiej wymaganej infrastruktury do wdrażania systemów operacyjnych i aplikacji, które normalnie byłyby potrzebne w centrum danych zarządzanym przez użytkownika; (2) Platforma jako usługa (PaaS). Usługa obejmuje języki i narzędzia programowania oraz platformę dostarczania aplikacji hostowaną przez dostawcę usług w celu wsparcia rozwoju i dostarczania aplikacji dla użytkowników końcowych; oraz (3) Oprogramowanie jako usługa (SaaS). Hostowane aplikacje oprogramowania są dostarczane i zarządzane przez dostawcę usług dla użytkowników końcowych, którzy zastępują aplikacje uruchamiane lokalnie aplikacjami internetowymi. Aplikacje obliczeniowe intensywnie korzystające z danych są wdrażane przy użyciu modelu IaaS, który umożliwia dostarczanie skalowalnych klastrów procesorów do przetwarzania danych równoległych przy użyciu różnych architektur oprogramowania lub model PaaS, który zapewnia kompletne środowisko przetwarzania i tworzenia aplikacji, obejmujące zarówno elementy infrastruktury, jak i platformy, takie jak języki programowania i narzędzia do tworzenia aplikacji. Przetwarzanie intensywnie wykorzystujące dane można wdrożyć w chmurze publicznej (infrastruktura i platforma chmury jest publicznie dostępna od dostawcy usług w chmurze), takiej jak Elastic Compute Cloud (EC2) i Elastic MapReduce firmy Amazon, lub jako chmura prywatna (infrastruktura i platforma chmury są obsługiwane wyłącznie dla konkretnej organizacji i może istnieć wewnętrznie lub zewnętrznie w stosunku do organizacji). Implementacje IaaS i PaaS do przetwarzania intensywnie przetwarzającego dane mogą być dynamicznie udostępniane w zwirtualizowanych środowiskach przetwarzania w oparciu o wymagania dotyczące harmonogramowania aplikacji i przetwarzania danych lub mogą być wdrażane jako trwała konfiguracja o wysokiej dostępności. Trwała konfiguracja

ma przewagę pod względem wydajności, ponieważ wykorzystuje dedykowaną infrastrukturę zamiast wirtualnych serwerów współdzielonych z innymi użytkownikami.

Aplikacje obliczeniowe intensywnie korzystające z danych

Podejścia przetwarzania równoległego można ogólnie sklasyfikować jako wymagające dużej mocy obliczeniowej lub wymagającej dużej ilości danych. Intensywne obliczeniowo jest używane do opisywania programów aplikacji, które są powiązane z obliczeniami. Takie aplikacje poświęcają większość czasu wykonania na wymagania obliczeniowe, w przeciwieństwie do operacji we/wy, i zazwyczaj wymagają niewielkich ilości danych. Przetwarzanie równoległe aplikacji intensywnie korzystających z mocy obliczeniowej zazwyczaj obejmuje zrównoleglenie poszczególnych algorytmów w ramach procesu aplikacji i rozłożenie całego procesu aplikacji na oddzielne zadania, które następnie mogą być wykonywane równolegle na odpowiedniej platformie obliczeniowej, aby osiągnąć ogólną wyższą wydajność niż przetwarzanie szeregowe. W aplikacjach wymagających dużej mocy obliczeniowej wiele operacji jest wykonywanych jednocześnie, a każda operacja dotyczy określonej części problemu. Jest to często określane jako równoległość funkcjonalna lub równoległość sterowania. Intensywne przetwarzanie danych jest używane do opisywania aplikacji, które są powiązane z operacjami we/wy lub muszą przetwarzać duże ilości danych . Takie aplikacje poświęcają większość czasu przetwarzania na operacje we/wy i przenoszenie danych. Przetwarzanie równoległe aplikacji intensywnie korzystających z danych zazwyczaj obejmuje partycjonowanie lub dzielenie danych na wiele segmentów, które mogą być przetwarzane niezależnie przy użyciu tego samego wykonywalnego programu aplikacji równolegle na odpowiedniej platformie obliczeniowej, a następnie ponowne składanie wyników w celu uzyskania kompletnych danych wyjściowych. Im większa zbiorcza dystrybucja danych, tym większe korzyści z równoległego przetwarzania danych. Gorton i inni stwierdzają, że wymagania dotyczące przetwarzania intensywnego danych zwykle skalują się liniowo w zależności od rozmiaru danych i są bardzo podatne na prostą równoległość. Podstawowe wyzwania dla intensywnego przetwarzania danych według Gortona zarządzają i przetwarzająwykładniczo rosnące ilości danych, znacznie redukujące powiązane cykle analizy danych w celu obsługi praktycznych, aktualnych aplikacji oraz opracowywania nowych algorytmów, które można skalować w celu wyszukiwania i przetwarzania ogromnych ilości danych. Przetwarzanie w chmurze może sprostać tym wyzwaniom dzięki możliwości udostępnienia nowych zasobów obliczeniowych lub rozszerzenia istniejących zasobów w celu zapewnienia możliwości przetwarzania równoległego, które skalują się w celu dostosowania do rosnących ilości danych.

Równoległość danych

Architektury systemów komputerowych, które mogą obsługiwać aplikacje równoległe do danych, stanowią potencjalne rozwiązanie wymagań przetwarzania danych w skali terabajtowej i petabajtowej. Według Agichteina i Ganti (2004) zrównoleglenie uważane jest za atrakcyjną alternatywę dla przetwarzania niezwykle dużych zbiorów danych, takich jak miliardy dokumentów w sieci (Agichtein, 2004). Nyland i in. zdefiniuj równoległość danych jako obliczenie stosowane niezależnie do każdego elementu danych w zbiorze danych, które umożliwia skalowanie stopnia równoległości z objętością danych. Według Nylanda i innych , najważniejszym powodem tworzenia aplikacji do pracy równoległej z danymi jest możliwość skalowania wydajności, która może skutkować poprawą wydajności o kilka rzędów wielkości. Kluczowymi kwestiami związanymi z tworzeniem aplikacji wykorzystujących równoległość danych są wybór algorytmu, strategia dekompozycji danych, równoważenie obciążenia w węzłach przetwarzania, komunikacja przekazująca komunikaty między węzłami oraz ogólna dokładność wyników. Nyland i in. Należy również zauważyć, że rozwój aplikacji dataparallel może wymagać znacznej złożoności programowania w celu zdefiniowania problemu w kontekście dostępnych narzędzi programistycznych i rozwiązania ograniczeń architektury docelowej.

Wyodrębnianie informacji z dokumentów internetowych i indeksowanie ich jest typowe dla przetwarzania intensywnie wykorzystującego dane, które może uzyskać znaczne korzyści w zakresie wydajności z implementacji równoległych danych, ponieważ kolekcje dokumentów internetowych i innych typów mogą być wówczas przetwarzane równolegle .

„Luka w danych"

Szybki rozwój Internetu i sieci World Wide Web doprowadził do ogromnych ilości informacji dostępnych online. Ponadto organizacje biznesowe i rządowe tworzą duże ilości zarówno ustrukturyzowanych, jak i nieustrukturyzowanych informacji, które muszą być przetwarzane, analizowane i łączone. Vinton Cerf z Google opisał to jako „lawinę informacyjną" i stwierdził, że „musimy wykorzystać energię Internetu, zanim informacje, które wyzwolił, nas pogrzebią" (Cerf, 2007). W białej księdze IDC sponsorowanej przez EMC oszacowano ilość informacji przechowywanych obecnie w formie cyfrowej w 2007 roku na 281 eksabajtów, a ogólny złożony wskaźnik wzrostu na 57%, przy czym informacje w organizacjach rosły w jeszcze szybszym tempie (Gantz i in., 2007). . W innym badaniu tak zwanej eksplozji informacji oszacowano, że 95% wszystkich aktualnych informacji istnieje w formie nieustrukturyzowanej o zwiększonych wymaganiach w zakresie przetwarzania danych w porównaniu z informacjami ustrukturyzowanymi. Przechowywanie, zarządzanie, uzyskiwanie dostępu i przetwarzanie tej ogromnej ilości danych stanowi podstawową potrzebę i ogromne wyzwanie w celu zaspokojenia potrzeb wyszukiwania, analizowania, wydobywania i wizualizacji tych danych jako informacji. W 2003 r. LexisNexis określił tę kwestię jako „lukę w danych": zdolność do gromadzenia informacji znacznie przewyższa zdolność organizacji do efektywnego ich wykorzystywania. Organizacje budują aplikacje, aby wypełnić dostępną pamięć, a także budują pamięć, aby dopasować ją do aplikacji i danych, które posiadają. Ale czy organizacje będą w stanie robić przydatne rzeczy z informacjami, które mają, aby uzyskać pełne i innowacyjne wykorzystanie swoich niewykorzystanych zasobów danych? W miarę wzrostu danych organizacji, w jaki sposób „luka w danych" zostanie rozwiązana i pokryta? Naukowcy z LexisNexis uważają, że odpowiedzią jest skalowalna architektura sprzętu i oprogramowania systemów komputerowych zaprojektowana dla aplikacji obliczeniowych intensywnie przetwarzających dane, które mogą być skalowane do przetwarzania miliardów rekordów na sekundę (BORPS) (Uwaga: termin BORPS został wprowadzony przez Seisint, Inc. w 2002. Seisint została przejęta przez LexisNexis w 2004). Jakie są charakterystyki systemów obliczeniowych intensywnie przetwarzających dane i jakie architektury systemów są dostępne dla organizacji w celu wdrożenia aplikacji obliczeniowych intensywnie korzystających z danych? Czy te możliwości można wdrożyć za pomocą chmury obliczeniowej, aby zmniejszyć ryzyko i początkowe inwestycje w infrastrukturę oraz umożliwić model płatności zgodnie z rzeczywistym użyciem? W tej części zbadamy te zagadnienia i przedstawimy porównanie dostępnych na rynku architektur systemów.

Charakterystyka systemów obliczeniowych intensywnie przetwarzających dane

National Science Foundation uważa, że intensywne przetwarzanie danych wymaga „zasadniczo innego zestawu zasad" niż obecne podejścia komputerowe. Poprzez program finansowania w obszarze informatyki i inżynierii, NSF stara się „zwiększyć zrozumienie możliwości i ograniczeń przetwarzania intensywnie wykorzystującego dane". Kluczowe obszary zainteresowania to:

• Podejścia do programowania równoległego w celu rozwiązania problemu równoległego przetwarzania danych w systemach intensywnie przetwarzających dane

• Programowanie abstrakcji, w tym modeli, języków i algorytmów, które pozwalają na naturalną ekspresję równoległego przetwarzania danych

• Projektowanie platform obliczeniowych intensywnie korzystających z danych w celu zapewnienia wysokiego poziomu niezawodności, wydajności, dostępności i skalowalności.

• Identyfikowanie aplikacji, które mogą wykorzystywać ten paradygmat obliczeniowy i określanie, w jaki sposób powinien on ewoluować, aby wspierać pojawiające się aplikacje intensywnie korzystające z danych.

Laboratoria Pacific Northwest National Labs zdefiniowały intensywne przetwarzanie danych jako „przechwytywanie, zarządzanie, analizowanie i rozumienie danych w ilościach i szybkościach, które przesuwają granice obecnych technologii" . Uważają, że aby sprostać szybko rosnącym wolumenom i złożoności danych, potrzebne są „epokowe postępy w rozwoju oprogramowania, sprzętu i algorytmów", które można łatwo skalować wraz z rozmiarem danych i zapewniać skuteczną i terminową analizę oraz wyniki przetwarzania. Architektura HPCC opracowana przez LexisNexis reprezentuje taki postęp w zakresie możliwości.

Podejście do przetwarzania

Obecne platformy obliczeniowe intensywnie korzystające z danych stosują podejście przetwarzania równoległego typu „dziel i zwyciężaj", łączące wiele procesorów i dysków w duże klastry obliczeniowe połączone za pomocą szybkich przełączników komunikacyjnych i sieci, co pozwala na dzielenie danych między dostępne zasoby obliczeniowe i niezależne przetwarzanie w celu osiągnąć wydajność i skalowalność w oparciu o ilość danych. Buyya, Yeo, Venugopal, Broberg i Brandic (2009) definiują klaster jako „rodzaj równoległego i rozproszonego systemu, który składa się ze zbioru połączonych, niezależnych komputerów pracujących razem jako pojedynczy zintegrowany zasób obliczeniowy". Takie podejście do przetwarzania równoległego jest często określane jako podejście „brak współużytkowania", ponieważ każdy węzeł składający się z procesora, pamięci lokalnej i zasobów dyskowych nie dzieli nic z innymi węzłami w klastrze. Równoległe obliczanie tego podejścia jest uważane za odpowiednie dla problemów przetwarzania danych, które są „żenująco równoległe", tj. gdzie stosunkowo łatwo jest rozdzielić problem na kilka równoległych zadań i nie ma zależności ani komunikacji między zadaniami innych niż ogólne zarządzanie zadaniami. Tego typu problemy z przetwarzaniem danych z natury można dostosować do różnych form przetwarzania rozproszonego, w tym klastrów i siatek danych oraz przetwarzania w chmurze.

Wspólne cechy

Istnieje kilka ważnych wspólnych cech systemów obliczeniowych intensywnie przetwarzających dane, które odróżniają je od innych form przetwarzania. Pierwsza to zasada kolokacji danych i programów lub algorytmów w celu wykonania obliczeń. Aby osiągnąć wysoką wydajność podczas przetwarzania intensywnego danych, ważne jest, aby zminimalizować przepływ danych. W przeciwieństwie do innych rodzajów obliczeń i superkomputerów, które wykorzystują dane przechowywane w oddzielnym repozytorium lub na serwerach i przesyłają je do systemu przetwarzania w celu obliczeń, obliczenia intensywnie wykorzystujące dane wykorzystują dane rozproszone i rozproszone systemy plików, w których dane znajdują się w klastrze węzłów przetwarzania, a zamiast przenoszenia danych, program lub algorytm jest przesyłany do węzłów z danymi, które mają zostać przetworzone. Ta zasada - „Przenieś kod do danych" - która została zaprojektowana w architekturze przetwarzania równoległego danych wdrożonej przez firmę Seisint w 2003 roku, jest niezwykle skuteczne, ponieważ rozmiar programu jest zwykle mały w porównaniu z dużymi zestawami danych przetwarzanymi przez systemy intensywnie korzystające z danych i skutkuje znacznie mniejszym ruchem w sieci, ponieważ dane mogą być odczytywane lokalnie, a nie w sieci. Ta cecha umożliwia wykonywanie algorytmów przetwarzania w węzłach, w których znajdują się dane, zmniejszając obciążenie systemu i zwiększając wydajność. Drugą ważną cechą systemów obliczeniowych intensywnie korzystających z danych jest

wykorzystywany model programowania. Systemy obliczeniowe intensywnie korzystające z danych wykorzystują podejście niezależne od maszyny, w którym aplikacje są wyrażane w postaci operacji wysokiego poziomu na danych, a system wykonawczy w przejrzysty sposób kontroluje planowanie, wykonywanie, równoważenie obciążenia, komunikację oraz ruch programów i danych w rozproszone klastry obliczeniowe. Abstrakcja programowania i narzędzia językowe pozwalają na wyrażenie przetwarzania w kategoriach przepływów danych i przekształceń obejmujących nowe języki programowania przepływu danych i współdzielone biblioteki typowych algorytmów manipulacji danymi, takich jak sortowanie. Konwencjonalne superkomputery i rozproszone systemy obliczeniowe zazwyczaj wykorzystują modele programowania zależne od maszyny, które mogą wymagać niskopoziomowej kontroli programisty nad przetwarzaniem i komunikacją węzłów przy użyciu konwencjonalnych imperatywnych języków programowania i wyspecjalizowanych pakietów oprogramowania, które zwiększają złożoność zadania programowania równoległego i zmniejszają produktywność programistów. Model programowania zależny od maszyny również wymaga znacznego dostrojenia i jest bardziej podatny na pojedyncze punkty awarii. Trzecią ważną cechą systemów obliczeniowych intensywnie korzystających z danych jest skupienie się na niezawodności i dostępności. Systemy na dużą skalę z setkami lub tysiącami węzłów przetwarzania są z natury bardziej podatne na awarie sprzętu, błędy komunikacji i błędy oprogramowania. Systemy obliczeniowe intensywnie korzystające z danych są zaprojektowane tak, aby były odporne na awarie. Obejmuje to nadmiarowe kopie wszystkich plików danych na dysku, przechowywanie pośrednich wyników przetwarzania na dysku, automatyczne wykrywanie awarii węzła lub przetwarzania oraz selektywne ponowne obliczanie wyników. Klaster przetwarzania skonfigurowany do obliczeń intensywnie przetwarzających dane jest zazwyczaj w stanie kontynuować działanie ze zmniejszoną liczbą węzłów po awarii węzła z automatycznym i przejrzystym odzyskiwaniem niekompletnego przetwarzania. Ostatnią ważną cechą systemów obliczeniowych intensywnie korzystających z danych jest nieodłączna skalowalność podstawowej architektury sprzętu i oprogramowania. intensywnie przetwarzający dane systemy obliczeniowe można zazwyczaj skalować w sposób liniowy, aby pomieścić praktycznie dowolną ilość danych lub spełnić wymagania dotyczące wydajności krytycznej czasowo, po prostu dodając dodatkowe węzły przetwarzania do konfiguracji systemu w celu osiągnięcia szybkości przetwarzania miliardów rekordów na sekundę (BORPS) . Liczba węzłów i zadań przetwarzania przypisanych do określonej aplikacji może być zmienna lub stała w zależności od sprzętu, oprogramowania, komunikacji i architektury rozproszonego systemu plików. Ta skalowalność pozwala na problemy obliczeniowe, które kiedyś uważano za trudne do rozwiązania ze względu na ilość wymaganych danych lub czas przetwarzania, który jest obecnie wykonalny, i stwarza możliwości nowych przełomów w analizie danych i przetwarzaniu informacji.

Obliczenia siatkowe

Podobny paradygmat obliczeniowy, znany jako przetwarzanie siatkowe, zyskał popularność przede wszystkim w środowiskach badawczych. Siatka obliczeniowa jest zazwyczaj z natury heterogeniczna (węzły mogą mieć różne procesory, pamięć i zasoby dyskowe) i składa się z wielu różnych komputerów rozproszonych w organizacjach i często geograficznie korzystających z komunikacji sieciowej rozległego obszaru, zwykle o stosunkowo niskiej przepustowości. Siatki są zwykle używane do rozwiązywania złożonych problemów obliczeniowych, które wymagają dużej mocy obliczeniowej i wymagają tylko niewielkich ilości danych dla każdego węzła przetwarzania. Odmiana znana jako siatki danych umożliwia dostęp do współdzielonych repozytoriów danych przez siatkę i wykorzystanie ich w przetwarzaniu aplikacji, jednak niska przepustowość siatek danych ogranicza ich skuteczność w przypadku aplikacji intensywnie przetwarzających dane na dużą skalę. W przeciwieństwie do tego, systemy obliczeniowe intensywnie korzystające z danych są zazwyczaj jednorodne (węzły w klastrze obliczeniowym mają identyczne zasoby procesora, pamięci i dysków), wykorzystują komunikację o

dużej przepustowości między węzłami, na przykład przełączniki Gigabit Ethernet, i są zlokalizowane w bliskiej odległości centrum danych wykorzystujące sprzęt o dużej gęstości, taki jak serwery kasetowe montowane w stojaku. Logiczny system plików zazwyczaj obejmuje wszystkie dyski dostępne w węzłach w klastrze, a pliki danych są rozproszone między węzłami, w przeciwieństwie do oddzielnego współdzielonego repozytorium danych, takiego jak sieć pamięci masowej, która wymagałaby przeniesienia danych do węzłów w celu przetworzenia. Geograficznie rozproszone systemy gridowe są trudniejsze w zarządzaniu, mniej niezawodne i mniej bezpieczne niż systemy obliczeniowe intensywnie korzystające z danych, które zwykle znajdują się w bezpiecznych środowiskach centrów danych.

Zastosowanie do przetwarzania w chmurze

Przetwarzanie w chmurze może przybierać różne formy. Większość z nich wizualizuje chmurę jako Internet lub Sieć, która jest często przedstawiana w ten sposób, ale bardziej ogólna definicja mówi, że przetwarzanie w chmurze przesuwa lokalizację zasobów obliczeniowych i infrastruktury dostarczającej aplikacje obliczeniowe do sieci. Oprogramowanie dostępne za pośrednictwem chmury staje się usługą, platformy aplikacji dostępne za pośrednictwem chmury w celu opracowywania i dostarczania nowych aplikacji stają się usługą, a sprzęt i oprogramowanie do tworzenia infrastruktury i środowisk wirtualnych centrów danych dostępnych za pośrednictwem chmury staje się usługą. Inne cechy zwykle kojarzone z przetwarzaniem w chmurze obejmują redukcję kosztów związanych z zarządzaniem sprzętem i zasobami oprogramowania , dostęp do aplikacji typu pay-per-use lub pay-as-you-go do aplikacji i zasobów obliczeniowych na żądanie , dynamiczne udostępnianie infrastruktury oraz skalowalność zasobów w celu dopasowania do wielkości danych i wymagań obliczeniowych, co ma bezpośrednie zastosowanie do cech przetwarzania intensywnie korzystającego z danych . Buyya i in. podać następującą kompleksową definicję chmury: „Chmura jest rodzajem równoległego i rozproszonego systemu składającego się ze zbioru wzajemnie połączonych i zwirtualizowanych komputerów, które są dynamicznie dostarczane i prezentowane jako jeden lub więcej zunifikowanych zasobów obliczeniowych opartych na usłudze umowy na poziomie ustalane w drodze negocjacji między usługodawcą a konsumentem.” Modele przetwarzania w chmurze mające bezpośrednie zastosowanie do cech obliczeniowych wymagających dużej ilości danych to infrastruktura jako usługa (IaaS) i platforma jako usługa (PaaS). IaaS zazwyczaj obejmuje dużą pulę konfigurowalnych zasobów zwirtualizowanych, które mogą obejmować sprzęt, systemy operacyjne, oprogramowanie pośrednie i platformy programistyczne lub inne usługi oprogramowania, które można skalować w celu dostosowania do różnych obciążeń przetwarzania . W tym modelu można zapewnić klastry obliczeniowe zwykle używane do przetwarzania intensywnego danych. Środowiska przetwarzania, takie jak Hadoop MapReduce i LexisNexis HPCC, które oprócz podstawowej infrastruktury obejmują możliwości platformy do tworzenia aplikacji, implementują model Platform as a Service (PaaS). Aplikacje o wysokim stopniu równoległości danych i wymaganiu przetwarzania bardzo dużych zbiorów danych mogą korzystać z chmury obliczeniowej i IaaS lub PaaS przy użyciu setek komputerów udostępnianych na krótki czas zamiast jednego lub niewielkiej liczby komputerów przez długi czas. Według Armbrust i in. W raporcie badawczym Uniwersytetu Kalifornijskiego w Berkeley ten model przetwarzania jest szczególnie dobrze przystosowany do analizy danych i innych aplikacji, które mogą skorzystać na równoległym przetwarzaniu wsadowym. Jednak analiza kosztów i korzyści dla użytkownika powinna również obejmować koszt przeniesienia dużych zestawów danych do chmury, a także przyspieszenie i niższe koszty przetwarzania oferowane przez modele IaaS i PaaS.

Architektury systemów intensywnie korzystających z danych

Zaimplementowano różne architektury systemowe dla aplikacji do analizy danych na dużą skalę i intensywnie przetwarzających dane, w tym równoległe i rozproszone systemy zarządzania relacyjnymi bazami danych, które od ponad dwóch dekad można uruchamiać na wspólnych klastrach węzłów

przetwarzania. Należą do nich systemy baz danych firm Teradata, Netezza, Vertica i Exadata/Oracle oraz inne, które zapewniają wydajne równoległe platformy baz danych. Chociaż te systemy mają możliwość uruchamiania równoległych aplikacji i zapytań wyrażonych w języku SQL, zazwyczaj nie są platformami przetwarzania ogólnego przeznaczenia i zwykle działają jako zaplecze dla oddzielnego systemu przetwarzania aplikacji typu front-end. Chociaż takie podejście oferuje korzyści, gdy wykorzystywane dane mają przede wszystkim strukturę i łatwo dopasowują się do ograniczeń relacyjnej bazy danych, a często doskonale sprawdzają się w zastosowaniach do przetwarzania transakcji, większość przyrostu danych dotyczy danych w formie nieustrukturyzowanej i nowych paradygmatów przetwarzania z bardziej elastycznymi danymi potrzebne były modele. Firmy internetowe, takie jak Google, Yahoo, Microsoft, Facebook i inne, wymagały nowego podejścia do przetwarzania, aby skutecznie radzić sobie z ogromną ilością danych internetowych dla aplikacji takich jak wyszukiwarki i sieci społecznościowe. Ponadto wiele organizacji rządowych i biznesowych zostało przytłoczonych danymi, których nie można było skutecznie przetwarzać, łączyć i analizować za pomocą tradycyjnych metod obliczeniowych. Pojawiło się kilka rozwiązań, w tym architektura MapReduce zapoczątkowana przez Google, a teraz dostępna w implementacji open source o nazwie Hadoop, używanej przez Yahoo, Facebook i inne. LexisNexis, uznany lider branży usług informacyjnych, opracował i wdrożył również skalowalną platformę do przetwarzania intensywnego danych, która jest wykorzystywana przez LexisNexis oraz inne organizacje komercyjne i rządowe do przetwarzania dużych ilości ustrukturyzowanych i nieustrukturyzowanych danych. Podejścia te zostaną wyjaśnione i skonfrontowane pod względem ich ogólnej struktury, modelu programowania, systemów plików i zastosowania w chmurze obliczeniowej w kolejnych sekcjach. Podobne podejścia wykorzystujące klastry obliczeniowe, w tym Sektor/Sfera, opisane niedawno w literaturze, są również odpowiednie dla aplikacji przetwarzania w chmurze intensywnie korzystających z danych i stanowią dodatkowe alternatywy.

Google MapReduce

Architektura i model programowania MapReduce zapoczątkowane przez Google jest przykładem nowoczesnej architektury systemowej przeznaczonej do przetwarzania i analizy dużych zbiorów danych i jest z powodzeniem wykorzystywana przez Google w wielu aplikacjach do przetwarzania ogromnych ilości danych rawWeb. Architektura MapReduce umożliwia programistom używanie funkcjonalnego stylu programowania do tworzenia funkcji mapowania, która przetwarza parę klucz-wartość skojarzoną z danymi wejściowymi w celu wygenerowania zestawu pośrednich par klucz-wartość oraz funkcji redukcji, która łączy wszystkie wartości pośrednie powiązane z ten sam klucz pośredni. Według Deana i Ghemawata programy MapReduce mogą służyć do obliczania danych pochodnych z dokumentów, takich jak odwrócone indeksy, a przetwarzanie jest automatycznie zrównoleglone przez system, który działa na dużych klastrach maszyn typu towarowego, wysoce skalowalnych do tysięcy maszyn . Ponieważ system automatycznie zajmuje się szczegółami, takimi jak partycjonowanie danych wejściowych, planowanie i wykonywanie zadań w klastrze przetwarzania oraz zarządzanie komunikacją między węzłami, programiści bez doświadczenia w programowaniu równoległym mogą z łatwością korzystać z dużego środowiska przetwarzania rozproszonego. Model programowania dla architektury MapReduce to prosta abstrakcja, w której obliczenia pobierają zestaw wejściowych par klucz-wartość skojarzonych z danymi wejściowymi i tworzą zestaw wyjściowych par klucz-wartość. W fazie mapowania dane wejściowe są dzielone na podziały wejściowe i przypisywane do zadań mapowania powiązanych z węzłami przetwarzania w klastrze. Zadanie mapowania jest zwykle wykonywane w tym samym węźle zawierającym przypisaną do niego partycję danych w klastrze. Te zadania mapowania wykonują określone przez użytkownika obliczenia na każdej wejściowej parze klucz-wartość z partycji danych wejściowych przypisanych do zadania i generują zestaw wyników pośrednich dla każdego klucza. Faza tasowania i sortowania następnie pobiera dane

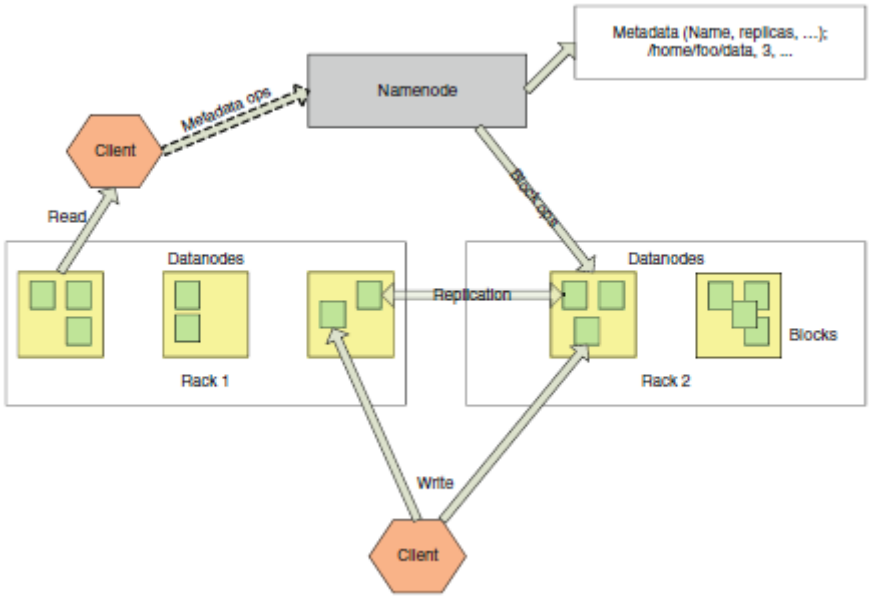
pośrednie generowane przez każde zadanie Mapy, sortuje te dane z danymi pośrednimi z innych węzłów, dzieli te dane na regiony do przetworzenia przez zadania redukcji i dystrybuuje te dane w razie potrzeby do węzłów, w których funkcja Redukcja zadania zostaną wykonane. Wszystkie zadania na mapie muszą zostać ukończone przed przetasowaniem oraz posortowaniem i skróceniem faz. Liczba zadań Redukcja nie musi być taka sama jak liczba zadań Mapy. Zadania Zmniejszanie wykonują dodatkowe operacje określone przez użytkownika na danych pośrednich, prawdopodobnie scalając wartości skojarzone z kluczem z mniejszym zestawem wartości w celu wygenerowania danych wyjściowych. W przypadku bardziej złożonych procedur przetwarzania danych wiele wywołań MapReduce może być połączonych ze sobą w sekwencji. Dane wejściowe mogą składać się z wielu plików wejściowych. Każde zadanie Mapy wygeneruje pośredni plik wyjściowy dla każdego obszaru klucza przypisanego na podstawie liczby zadań Redukcja R przypisanych do procesu (moduł skrótu (klucza) R). Funkcja zmniejszania następnie „wyciąga" pliki pośrednie, sortując i łącząc pliki dla określonego regionu ze wszystkich zadań mapy. Aby zminimalizować ilość danych przesyłanych przez sieć, można określić opcjonalną funkcję Combiner, która jest wykonywana na tym samym węźle, który wykonuje zadanie Map. Kod łączenia jest zwykle taki sam, jak kod funkcji redukującej, która wykonuje częściowe scalanie i redukowanie danych dla partycji lokalnej, a następnie zapisuje pliki pośrednie, które mają być dystrybuowane do zadań Zmniejsz. Dane wyjściowe funkcji Reduce są zapisywane jako końcowy plik wyjściowy. W implementacji Google MapReduce funkcje są kodowane w języku programowania C++. Podstawą i nałożoną na architekturę MapReduce jest system plików Google (GFS). GFS został zaprojektowany jako wysokowydajny, skalowalny rozproszony system plików dla bardzo dużych plików danych i aplikacji intensywnie korzystających z danych, zapewniający odporność na błędy i działający na klastrach zwykłego sprzętu. GFS jest zorientowany na bardzo duże pliki, dzieląc je i przechowując domyślnie w porcjach o stałym rozmiarze 64 Mb, które są zarządzane przez węzły w klastrze zwane chunkserverami. Każdy system GFS składa się z pojedynczego węzła głównego działającego jako serwer nazw i wielu węzłów w klastrze działających jako serwery fragmentów przy użyciu standardowej maszyny opartej na systemie Linux (węzeł w klastrze) z uruchomionym procesem serwera na poziomie użytkownika. Kawałki są przechowywane w zwykłych plikach Linuksa, które są rozszerzane tylko w razie potrzeby i replikowane na wielu węzłach, aby zapewnić wysoką dostępność i poprawić wydajność. Dodatkowe serwery nazw zapewniają kopię zapasową węzła głównego. Duży rozmiar fragmentu zmniejsza potrzebę interakcji programów klienckich MapReduce z węzłem głównym, umożliwia przechowywanie metadanych systemu plików w pamięci w węźle głównym, poprawiając wydajność i umożliwia wykonywanie wielu operacji za pomocą jednego odczytu fragmentu danych przez klient MapReduce. W idealnym przypadku podziały wejściowe dla operacji MapReduce mają rozmiar porcji GFS. GFS okazał się bardzo skuteczny w przypadku intensywnych obliczeń na bardzo dużych plikach, ale jest mniej skuteczny w przypadku małych plików, które mogą powodować gorące punkty, jeśli wiele zadań MapReduce uzyskuje dostęp do tego samego pliku. Firma Google wdrożyła dodatkowe narzędzia wykorzystujące architekturę MapReduce i GFS, aby poprawić produktywność programistów oraz usprawnić analizę danych i przetwarzanie danych ustrukturyzowanych i nieustrukturyzowanych. Ponieważ system plików GFS jest głównie zorientowany na sekwencyjne przetwarzanie dużych plików, Google wdrożył również skalowalny, rozproszony system przechowywania danych o wysokiej dostępności dla uporządkowanych danych z dynamiczną kontrolą nad formatem danych z możliwością losowego dostępu z kluczem. Dane są przechowywane w Bigtable jako rzadka, rozproszona, trwała wielowymiarowa posortowana mapa o strukturze indeksowanej według klucza wiersza, klucza kolumny i znacznika czasu. Wiersze w Bigtable są uporządkowane według klucza wiersza, a zakresy wierszy stają się jednostką dystrybucji i równoważenia obciążenia zwaną tabletem. Każda komórka danych w Bigtable może zawierać wiele wystąpień indeksowanych według sygnatury czasowej. Bigtable używa GFS do przechowywania zarówno plików danych, jak i dzienników. Interfejs API dla Bigtable jest elastyczny i zapewnia funkcje

zarządzania danymi, takie jak tworzenie i usuwanie tabel, oraz funkcje manipulacji danymi według klucza wiersza, w tym operacje odczytu, zapisu i modyfikacji danych. Informacje indeksowe dla Bigtables wykorzystują informacje z tabletu przechowywane w strukturach podobnych do B+Drzewo. Aplikacje MapReduce mogą być używane z Bigtable do przetwarzania i przekształcania danych, a firma Google wdrożyła wiele aplikacji na dużą skalę, które wykorzystują Bigtable do przechowywania, w tym Google Earth. Firma Google wdrożyła również język wysokiego poziomu do przeprowadzania równoległej analizy danych i eksploracji danych przy użyciu architektury MapReduce i GFS o nazwie Sawzall oraz infrastruktury zarządzania przepływem pracy i harmonogramowania dla zadań Sawzall o nazwie 5 Data-Intensive Technologies for Cloud Computing 95. Chociaż C++ w standardowych zadaniach MapReduce jest w stanie obsłużyć zadania analizy danych, jest trudniejszy w użyciu i wymaga znacznego wysiłku programistów. W przypadku większości aplikacji zaimplementowanych przy użyciu Sawzall kod jest znacznie prostszy i mniejszy niż odpowiednik C++ o współczynnik 10 lub więcej. Program Sawzall definiuje operacje na pojedynczym rekordzie danych, język nie pozwala na badanie wielu rekordów wejściowych jednocześnie i jeden rekord wejściowy nie może wpływać na przetwarzanie innego. Instrukcja emit umożliwia wyprowadzenie przetworzonych danych do zewnętrznego agregatora, który umożliwia przetwarzanie całych plików rekordów i danych za pomocą programu Sawzall. System działa w trybie wsadowym, w którym użytkownik przesyła zadanie, które wykonuje program Sawzall na ustalonym zestawie plików i danych oraz zbiera dane wyjściowe na koniec przebiegu. Zadania Sawzall można łączyć w łańcuch, aby obsługiwać bardziej złożone procedury. Programy Sawzall są kompilowane do kodu pośredniego, który jest interpretowany podczas wykonywania w czasie wykonywania. Kilka powodów jest cytowanych przez Pike′ i innych dlaczego nowy język jest korzystny w zastosowaniach do analizy danych i eksploracji danych: (1) język programowania dostosowany do określonej dziedziny problemu sprawia, że powstałe programy są „wyraźniejsze, bardziej zwarte i bardziej wyraziste”; (2) agregacje są określone w języku Sawzall, dzięki czemu programista nie musi ich podawać w zadaniu Reduce standardowego programu MapReduce; (3) język programowania zorientowany na analizę danych zapewnia bardziej naturalny sposób myślenia o problemach z przetwarzaniem danych w przypadku dużych rozproszonych zbiorów danych; oraz (4) programy Sawzall są znacznie mniejsze niż równoważne programy C++ MapReduce i znacznie łatwiejsze do zaprogramowania. Google nie udostępnia obecnie swojej architektury MapReduce w środowisku chmury publicznej IaaS lub PaaS. Google dostarcza jednak Google Apps Engine jako publiczne środowisko PaaS do przetwarzania w chmurze

Hadoop

Hadoop to projekt oprogramowania open source sponsorowany przez The Apache Software Foundation (http://www.apache.org). Po opublikowaniu w 2004 roku artykułu badawczego opisującego Google MapReduce podjęto wysiłki w połączeniu z istniejącym projektem Nutch, aby stworzyć implementację open source architektury MapReduce. Później stał się niezależnym podprojektem Lucene, został przyjęty przez Yahoo! po tym, jak główny programista Hadoopa został pracownikiem i stał się oficjalnym projektem najwyższego poziomu Apache w lutym 2006 roku. Hadoop obejmuje teraz wiele podprojektów oprócz podstawowego rdzenia, MapReduce i rozproszonego systemu plików HDFS. Te dodatkowe podprojekty zapewniają ulepszone możliwości przetwarzania aplikacji w podstawowej implementacji Hadoop i obecnie obejmują Avro, Pig, HBase, ZooKeeper, Hive i Chukwa. Więcej informacji można znaleźć na stronie internetowej Apache. Architektura Hadoop MapReduce jest funkcjonalnie podobna do implementacji Google, z wyjątkiem tego, że podstawowym językiem programowania dla Hadoop jest Java zamiast C++. Implementacja jest przeznaczona do wykonania na klastrach procesorów standardowych wykorzystujących Linux jako środowisko systemu operacyjnego, ale może być również uruchomiona na pojedynczym systemie jako środowisko edukacyjne. Klastry Hadoop wykorzystują również paradygmat przetwarzania rozproszonego „brak

współużytkowania", łączący poszczególne systemy z lokalnym procesorem, pamięcią i zasobami dyskowymi przy użyciu funkcji szybkiego przełączania komunikacji, zwykle w konfiguracjach montowanych w szafie. Elastyczność konfiguracji Hadoop pozwala na tworzenie małych klastrów do testowania i rozwoju przy użyciu systemów stacjonarnych lub dowolnego systemu z systemem Unix/Linux udostępniającym środowisko JVM, jednak klastry produkcyjne zazwyczaj wykorzystują jednorodne procesory montowane w stojaku w środowisku centrum danych. Architektura Hadoop MapReduce jest podobna do implementacji Google, która tworzy podziały wejściowe o stałym rozmiarze z danych wejściowych i przypisuje podziały do zadań Map. Lokalne dane wyjściowe z zadań mapy są kopiowane do węzłów Zmniejsz, gdzie są sortowane i scalane w celu przetworzenia przez zadania Zmniejsz, które dają ostateczny wynik. Hadoop implementuje rozproszone przetwarzanie danych w środowisku planowania i wykonywania oraz framework dla zadań MapReduce. Zadanie MapReduce to jednostka pracy składająca się z danych wejściowych, skojarzonych programów Map i Reduce oraz informacji konfiguracyjnych określonych przez użytkownika. Framework Hadoop wykorzystuje architekturę master/slave z jednym serwerem głównym zwanym jobtracker i serwerami podrzędnymi zwanymi trackerami zadań, po jednym na węzeł w klastrze. Jobtracker jest interfejsem komunikacyjnym między użytkownikami a frameworkiem i koordynuje wykonanie miejsc pracy MapReduce. Użytkownicy przesyłają zadania do narzędzia do śledzenia zadań, które umieszcza je w kolejce zadań i wykonuje je na zasadzie „kto pierwszy, ten lepszy". Jobtracker zarządza przypisaniem zadań mapowania i zmniejszania do węzłów zadania trackera, które następnie wykonują te zadania. Moduły do śledzenia zadań obsługują również przenoszenie danych między fazami Map i Zmniejszenie wykonywania zadania. Struktura Hadoop przypisuje zadania Map do każdego węzła, w którym znajdują się podziały danych wejściowych w procesie zwanym optymalizacją lokalizacji danych. Liczba zadań Zmniejsz jest określana niezależnie i może być określona przez użytkownika i może wynosić zero, jeśli cała praca może zostać wykonana przez zadania Mapy. Podobnie jak w przypadku implementacji Google MapReduce, wszystkie zadania mapy muszą zostać ukończone, zanim będzie mogła nastąpić faza odtwarzania losowego i sortowania oraz zainicjowanie redukcji zadań. Struktura Hadoop obsługuje również funkcje Combinera, które mogą zmniejszyć ilość przenoszenia danych w zadaniu. Platforma Hadoop udostępnia również interfejs API o nazwie Streaming, który umożliwia pisanie funkcji Map i Reduce w językach innych niż Java, takich jak Ruby i Python, oraz zapewnia interfejs o nazwie Pipes for C++. Hadoop zawiera rozproszony system plików o nazwie HDFS, który jest analogiczny do GFS w implementacji Google MapReduce. Blok w HDFS jest odpowiednikiem fragmentu w GFS i jest również bardzo duży, domyślnie 64 Mb, ale w niektórych instalacjach używane jest 128 Mb. Duży rozmiar bloku ma na celu zmniejszenie liczby wyszukiwań i skrócenie czasu przesyłania danych. Każdy blok jest niezależną jednostką przechowywaną jako dynamicznie przydzielany plik w lokalnym systemie plików Linux w katalogu datanode. Jeśli węzeł ma wiele dysków, można określić wiele katalogów datanode. Dodatkowy plik lokalny na blok przechowuje metadane bloku. HDFS jest również zgodny z architekturą master/slave, która składa się z jednego serwera głównego, który zarządza przestrzenią nazw rozproszonego systemu plików i reguluje dostęp do plików przez klientów nazywanych Namenode. Ponadto istnieje wiele Datanodów, po jednym na węzeł w klastrze, które zarządzają pamięcią dyskową dołączoną do węzłów i przypisaną do usługi Hadoop. Namenode określa mapowanie bloków do Datanodes. Datanody są odpowiedzialne za obsługę żądań odczytu i zapisu od klientów systemu plików. Zadania MapReduce, a także wykonują tworzenie, usuwanie i replikację bloków na podstawie poleceń z Namenode. System HDFS może zawierać dodatkowe drugorzędne Namenody, które replikują metadane systemu plików, jednak nie ma usług przełączania awaryjnego na gorąco. Każdy blok datanode posiada również repliki na innych węzłach w oparciu o parametry konfiguracyjne systemu (domyślnie dla każdego bloku datanode są 3 repliki). W środowisku wykonawczym Hadoop MapReduce węzeł w fizycznym klastrze często działa zarówno jako Tasktracker, jak i jako węzeł danych. Architekturę systemu HDFS pokazano tu



Środowisko wykonawcze Hadoop obsługuje dodatkowe możliwości rozproszonego przetwarzania danych, które są zaprojektowane do działania przy użyciu architektury Hadoop MapReduce. Kilka z nich stało się oficjalnymi podprojektami Hadoop w ramach Apache Software Foundation. Należą do nich HBase, rozproszona baza danych zorientowana na kolumny, która zapewnia podobne możliwości odczytu/zapisu o dostępie swobodnym, jak i jest wzorowana na Bigtable zaimplementowanym przez Google. HBase nie jest relacyjny i nie obsługuje języka SQL, ale udostępnia interfejs API Java i powłokę wiersza polecenia do zarządzania tabelami. Hive to system hurtowni danych zbudowany na bazie Hadoop, który zapewnia podobne do SQL funkcje zapytań do podsumowania danych, zapytań ad hoc i analizy dużych zestawów danych. Inne projekty sankcjonowane przez Apache dla Hadoop to Avro – system serializacji danych zapewniający dynamiczną integrację z językami skryptowymi, Chukwa – system gromadzenia danych do zarządzania dużymi systemami rozproszonymi, ZooKeeper – wysokowydajna usługa koordynacji dla aplikacji rozproszonych oraz Pig – wysoka -poziom języka przepływu danych i ramy wykonawcze dla obliczeń równoległych. Pig to wysokopoziomowy język i środowisko wykonawcze zorientowane na przepływ danych, pierwotnie opracowane przez Yahoo! rzekomo z tych samych powodów, dla których Google opracował język Sawzall dla swojej implementacji MapReduce - aby zapewnić specyficzną notację językową dla aplikacji do analizy danych oraz poprawić produktywność programistów i skrócić cykle rozwoju podczas korzystania ze środowiska Hadoop MapReduce. Opracowanie sposobu dopasowania wielu aplikacji do analizy i przetwarzania danych do paradygmatu MapReduce może być wyzwaniem i często wymaga wielu zadań MapReduce (White, 2009). Programy Pig są automatycznie tłumaczone na sekwencje programów MapReduce, jeśli są potrzebne w środowisku wykonawczym. Ponadto Pig obsługuje znacznie bogatszy model danych, który obsługuje wielowartościowe, zagnieżdżone struktury danych z krotkami, torebkami i mapami. Pig obsługuje wysoki poziom dostosowywania użytkownika, w tym funkcje specjalne zdefiniowane przez użytkownika i zapewnia możliwości w języku do ładowania, przechowywania, filtrowania, grupowania, deduplikacji, porządkowania, sortowania, agregacji i łączenia operacji na danych (Olston, Reed, Srivastava, Kumar i Tomkins, 2008a). Pig jest imperatywnym językiem zorientowanym na przepływ danych (wyrażenia w języku definiują przepływ danych do przetwarzania). Pig działa jako aplikacja po stronie klienta, która tłumaczy programy Pig na zadania MapReduce, a następnie uruchamia je w klastrze Hadoop. Etapy kompilacji i wykonania programu Pig obejmują parser, optymalizator logiczny, kompilator MapReduce, optymalizator MapReduce i Hadoop Job Manager. Według Yahoo! gdzie ponad 40% zadań produkcyjnych Hadoop i 60% zapytań ad-hoc jest obecnie

wdrażanych przy użyciu Pig, programy Pig są 1/20 wielkości równoważnego programu MapReduce, a ich opracowanie zajmuje 1/16 czasu. Yahoo! używa 12 standardowych benchmarków (zwanych PigMix) do testowania wydajności Pig w porównaniu z równoważną wydajnością MapReduce od wydania do wydania. W obecnej wersji programy Pig zajmują około 1,5 razy dłużej niż odpowiednik MapReduce (http://wiki.apache.org/pig/PigMix). Wprowadzane są dodatkowe optymalizacje, które powinny jeszcze bardziej zmniejszyć tę lukę w wydajności. Hadoop jest dostępny zarówno w publicznych, jak i prywatnych środowiskach chmury obliczeniowej. Platforma przetwarzania w chmurze EC2 firmy Amazon obejmuje teraz Amazon Elastic MapReduce (http://aws.amazon.com/elasticmapreduce/), który umożliwia użytkownikom zapewnienie takiej pojemności, jaka jest potrzebna dla aplikacji obliczeniowych intensywnie korzystających z danych. Dane dla aplikacji MapReduce mogą być ładowane do HDFS bezpośrednio z S3 (Simple Storage Service) firmy Amazon.

LexisNexis HPCC

LexisNexis, lider branży w zakresie zawartości danych, agregacji danych i usług informacyjnych, niezależnie opracował i wdrożył rozwiązanie do przetwarzania intensywnego danych o nazwie HPCC (High-Performance Computing Cluster), zwane również superkomputerem analizy danych (DAS). Rozwój tej platformy obliczeniowej przez spółkę zależną Seisint firmy LexisNexis rozpoczął się w 1999 r., a aplikacje były w produkcji pod koniec 2000 r. Podejście LexisNexis wykorzystuje również klastry towarowe sprzętu z systemem operacyjnym Linux. Niestandardowe oprogramowanie systemowe i komponenty oprogramowania pośredniego zostały opracowane i ułożone warstwowo na podstawowym systemie operacyjnym Linux, aby zapewnić środowisko wykonawcze i obsługę rozproszonego systemu plików wymagane do przetwarzania intensywnego danych. Ponieważ firma LexisNexis dostrzegła potrzebę nowego paradygmatu obliczeniowego w celu zaadresowania rosnących ilości danych, podejście projektowe obejmowało definicję nowego języka wysokiego poziomu do równoległego przetwarzania danych o nazwie ECL (Enterprise Data Control Language). Moc, elastyczność, zaawansowane możliwości, szybkość rozwoju i łatwość użycia języka programowania ECL to główny czynnik odróżniający LexisNexis HPCC od innych rozwiązań obliczeniowych intensywnie korzystających z danych. Poniżej przedstawiono przegląd architektury systemów HPCC i języka ECL oraz ogólne porównanie z architekturą i platformą Hadoop MapReduce. Deweloperzy LexisNexis zauważyli, że spełnienie wszystkich wymagań aplikacji obliczeniowych intensywnie korzystających z danych wymaga zaprojektowania i wdrożenia dwóch odrębnych środowisk przetwarzania, z których każde można niezależnie zoptymalizować pod kątem równoległego przetwarzania danych. Pierwsza z tych platform nazywa się rafinerią danych, której ogólnym celem jest ogólne przetwarzanie ogromnych ilości surowych danych dowolnego typu w dowolnym celu, ale zwykle używane do czyszczenia i higieny danych, przetwarzania ETL surowych danych (wyodrębnianie, przekształcanie, ładowanie), łączenie rekordów i rozwiązywanie jednostek, wielkoskalową analizę danych ad hoc oraz tworzenie danych z kluczami i indeksów w celu obsługi wysokowydajnych zapytań strukturalnych i aplikacji hurtowni danych. Data Refinery jest również określana jako Thor, co jest odniesieniem do mitycznego nordyckiego boga piorunów z wielkim młotem symbolizującym zmiażdżenie dużych ilości surowych danych w użyteczne informacje. System Thor jest podobny pod względem konfiguracji sprzętowej, funkcji, środowiska wykonawczego, systemu plików i możliwości do platformy Hadoop MapReduce, ale oferuje znacznie wyższą wydajność w równoważnych konfiguracjach. Druga z zaprojektowanych i wdrożonych przez LexisNexis równoległych platform przetwarzania danych to Data Delivery Engine. Platforma ta została zaprojektowana jako wysokowydajna platforma zapytań strukturalnych i analizy online lub hurtownia danych zapewniająca równoległe przetwarzanie dostępu do danych wymagane przez aplikacje online za pośrednictwem interfejsów usług sieci Web obsługujących tysiące jednoczesnych zapytań i użytkowników z czasem odpowiedzi poniżej sekundy. Z platformy tej

korzystają znane aplikacje internetowe opracowane przez firmę LexisNexis, takie jak Accurint. Mechanizm dostarczania danych jest również określany jako Roxie, co jest akronimem od Rapid Online XML Enquiry Engine. Roxie używa specjalnego rozproszonego indeksowanego systemu plików, aby zapewnić równoległe przetwarzanie zapytań. System Roxie jest podobny pod względem funkcji i możliwości do Hadoop z dodanymi funkcjami HBase i Hive, ale zapewnia znacznie wyższą przepustowość, ponieważ wykorzystuje bardziej zoptymalizowane środowisko wykonawcze i system plików do wysokowydajnego przetwarzania online. Co najważniejsze, zarówno systemy Thor, jak i Roxie wykorzystują ten sam język programowania ECL do wdrażania aplikacji, zwiększając ciągłość i produktywność programistów. Klaster systemu Thor jest implementowany przy użyciu podejścia master/slave z pojedynczym węzłem głównym i wieloma węzłami podrzędnymi do równoległego przetwarzania danych. Każdy z węzłów podrzędnych jest również węzłem danych w rozproszonym systemie plików klastra. Jest to podobne do koncepcji Jobtracker, Tasktracker i Datanode w konfiguracji Hadoop. W środowisku HPCC może istnieć wiele klastrów Thor, a kolejki zadań mogą w razie potrzeby obejmować wiele klastrów w środowisku. Zadania wykonywane w klastrze Thor w środowisku wieloklastrowym mogą również w razie potrzeby odczytywać pliki z rozproszonego systemu plików w klastrach obcych. Warstwa oprogramowania pośredniego zapewnia dodatkowe procesy serwerowe do obsługi środowiska wykonawczego, w tym agentów ECL i serwerów ECL. Proces klienta przesyła zadanie ECL do agenta ECL, który koordynuje całościowe wykonanie zadania w imieniu procesu klienta. Zadanie ECL jest kompilowane przez serwer ECL, który współdziała z dodatkowym serwerem zwanym Repozytorium ECL, które jest repozytorium kodu źródłowego i zawiera współużytkowany kod ECL. Programy ECL są kompilowane do zoptymalizowanego kodu źródłowego C++, który jest następnie kompilowany w kod wykonywalny i dystrybuowany do węzłów podrzędnych klastra Thor przez węzeł główny Thor. Urządzenie główne Thor monitoruje i koordynuje czynności przetwarzania węzłów podrzędnych oraz przekazuje informacje o stanie monitorowane przez procesy agenta ECL. Po zakończeniu zadania agent ECL i proces klienta są powiadamiane, a wynik procesu jest dostępny do przeglądania lub dalszego przetwarzania. Dane wyjściowe mogą być przechowywane w rozproszonym systemie plików klastra lub zwracane do procesu klienta. ECL jest analogiczny do języka Pig, który może być używany w środowisku Hadoop. Rozproszony system plików używany w klastrze Thor jest zorientowany na rekordy, co różni się od formatu blokowego używanego przez klastry Hadoop. Rekordy mogą mieć stałą lub zmienną długość i obsługują różne standardowe (stały rozmiar rekordu, CSV, XML) i niestandardowe formaty, w tym zagnieżdżone podrzędne zestawy danych. Zapisy we/wy są buforowane w dużych blokach w celu zmniejszenia opóźnień i zwiększenia szybkości przesyłania danych zi na dysk Pliki, które mają zostać załadowane do klastra Thor, są zazwyczaj najpierw przesyłane do strefy lądowania z jakiejś lokalizacji zewnętrznej, a następnie odbywa się proces zwany „rozpylaniem". służy do partycjonowania pliku i załadowania go do węzłów klastra Thor. Początkowy proces natryskiwania dzieli plik na określone przez użytkownika granice rekordów i rozprowadza dane tak równomiernie, jak to możliwe, w kolejności dostępnych węzłów w klastrze. Pliki można również „rozpraszać", gdy są potrzebne do przesłania plików wyjściowych do innego systemu lub można je bezpośrednio kopiować między klastrami Thor w tym samym środowisku. Usługi nazw i przechowywanie metadanych o plikach, w tym informacje o formacie rekordu w Thor DFS, są utrzymywane na specjalnym serwerze zwanym serwerem Dali (nazwanym tak od zwierzaka szynszyli dewelopera), który jest analogiczny do Namenode w HDFS. Użytkownicy Thor mają pełną kontrolę nad dystrybucją danych w klastrze Thor i mogą w razie potrzeby redystrybuować dane w zadaniu ECL według określonych kluczy, pól lub kombinacji pól, aby ułatwić charakterystykę lokalizacji przetwarzania równoległego. Serwer nazw Dali używa dynamicznego magazynu danych dla metadanych systemu plików zorganizowanego w hierarchiczną strukturę odpowiadającą zakresowi plików w systemie. Thor DFS wykorzystuje lokalny system plików Linux do fizycznego przechowywania plików, a zakresy plików są tworzone przy użyciu struktur katalogów plików lokalnego systemu plików.

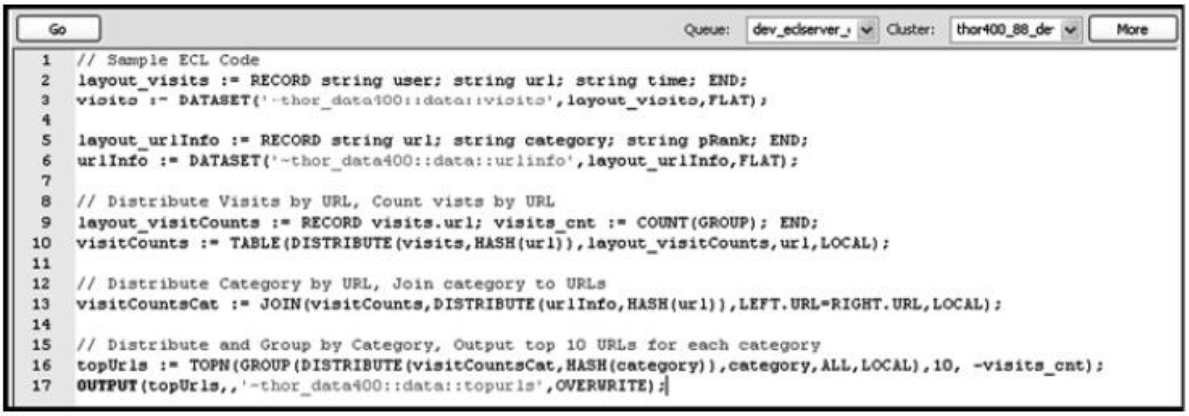
Części pliku rozproszonego są nazywane zgodnie z numerem węzła w klastrze, dzięki czemu plik w klastrze z 400 węzłami będzie zawsze miał 400 części, niezależnie od rozmiaru pliku. Stały rozmiar bloku Hadoop może doprowadzić do podziału rekordów logicznych między węzłami, co oznacza, że węzeł może potrzebować odczytać niektóre dane z innego węzła podczas przetwarzania zadania Map. Dzięki Thor DFS zachowana jest integralność rekordów logicznych, a przetwarzanie we/wy jest całkowicie zlokalizowane w węźle przetwarzania na potrzeby lokalnych operacji przetwarzania. Ponadto, jeśli rozmiar pliku w usłudze Hadoop jest mniejszy niż pewna wielokrotność rozmiaru bloku razy liczba węzłów w klastrze, przetwarzanie Hadoop będzie mniej równomiernie rozłożone i będzie potrzebny dostęp do dysków między węzłami. Jeśli podziały wejściowe przypisane do zadań Map w Hadoop nie są przydzielone w całych rozmiarach bloków, wyniknie dodatkowy węzeł do węzła we / wy. Możliwość łatwej redystrybucji danych równomiernie do węzłów w oparciu o wymagania dotyczące przetwarzania i charakterystykę danych podczas zadania Thora może zapewnić znaczną poprawę wydajności w porównaniu z podejściem Hadoop. Thor DFS obsługuje również koncepcję „superplików", które podczas uzyskiwania dostępu są przetwarzane jako jeden plik logiczny, ale składają się z wielu plików Thor DFS. Każdy plik tworzący superplik musi mieć taką samą strukturę rekordów. Nowe pliki mogą być dodawane, a stare usuwane z superpliku dynamicznie, ułatwiając procesy aktualizacji bez konieczności przepisywania nowego pliku. Klastry Thor są odporne na uszkodzenia, a co najmniej jedna replika każdej części pliku w pliku Thor DFS jest przechowywana w innym węźle w klastrze. Klastry Roxie składają się z konfigurowalnej liczby węzłów połączonych równorzędnie, działających jako platforma zapytań przetwarzania równoległego o wysokiej wydajności i wysokiej dostępności. Kod źródłowy ECL dla zapytań strukturalnych jest wstępnie kompilowany i wdrażany w klastrze. Rozproszony system plików Roxie to rozproszony, indeksowany system plików, który wykorzystuje niestandardową strukturę B + Tree do przechowywania danych. Indeksy i zapytania obsługujące dane są wstępnie zbudowane w klastrach Thor i wdrażane w systemie Roxie DFS z częściami indeksu i danymi przechowywanymi w każdym węźle. Zazwyczaj dane związane z kluczami logicznymi indeksu są osadzone w strukturze indeksu jako ładunek. Klucze indeksu mogą być wielopolowe i wielowymiarowe, a ładunki mogą zawierać dowolny typ danych strukturalnych lub niestrukturalnych obsługiwanych przez język ECL. Zapytania mogą używać tylu indeksów, ile jest wymaganych dla zapytania i zawierać łączenia i inne złożone przekształcenia danych z pełnymi możliwościami wyrażeń i przetwarzania języka ECL. Na przykład kompleksowy raport dotyczący osoby LexisNexis Accurint, który generuje wiele stron danych wyjściowych, jest generowany przez pojedyncze zapytanie Roxie. Klaster Roxie wykorzystuje koncepcję serwerów i agentów. Każdy węzeł w klastrze Roxie uruchamia procesy serwera i agenta, które mogą być konfigurowane przez administratora systemu w zależności od wymagań przetwarzania klastra. Proces serwera czeka na żądanie zapytania z interfejsu usług WWW, a następnie określa węzły i skojarzone procesy agenta, które mają lokalnie dane potrzebne do zapytania lub część zapytania. Żądania zapytań Roxie można przesyłać z aplikacji klienckiej jako wywołanie SOAP, żądanie protokołu HTTP lub HTTPS z aplikacji sieci Web lub przez bezpośrednie połączenie z gniazdem. Każde żądanie zapytania Roxie jest powiązane z określonym wdrożonym programem zapytań ECL. Zapytania Roxie mogą być również wykonywane z programów działających w klastrach Thor. Proces serwera Roxie, który odbiera żądanie, jest właścicielem przetwarzania programu ECL dla zapytania, dopóki nie zostanie ono zakończone. Serwer wysyła części zadania zapytania do węzłów w klastrze i procesy Agenta , które mają dane potrzebne do zapytania przechowywane lokalnie w razie potrzeby i czeka na wyniki. Gdy serwer otrzyma wszystkie potrzebne wyniki ze wszystkich węzłów, zestawia je, wykonuje dodatkowe przetwarzanie, a następnie zwraca zestaw wyników do żądającego klienta. Wydajność przetwarzania zapytań różni się w zależności od takich czynników, jak szybkość maszyny, złożoność danych, liczba węzłów i charakter zapytania, ale wyniki produkcyjne wykazały przepustowość tysiąca wyników na sekundę lub więcej. Klastry Roxie oferują elastyczne opcje przechowywania danych z indeksami i danymi przechowywanymi lokalnie w

klastrze, a także możliwość korzystania z indeksów przechowywanych zdalnie w tym samym środowisku w klastrze Thor. Serwer Dali zapewnia również usługi nazw dla klastrów Roxie. Klastry Roxie są odporne na awarie, a nadmiarowość danych jest wbudowana przy użyciu systemu równorzędnego, w którym repliki danych są przechowywane na dwóch lub więcej węzłach, a wszystkie dane, w tym repliki, są dostępne do wykorzystania w przetwarzaniu zapytań przez procesy agenta. Klaster Roxie zapewnia automatyczne przełączanie awaryjne w przypadku awarii węzła, a klaster będzie nadal działać, nawet jeśli jeden lub więcej węzłów nie działa. Dodatkową nadmiarowość można zapewnić, włączając w środowisko wiele klastrów Roxie. Równoważenie obciążenia żądań zapytań w klastrach Roxie jest zwykle implementowane przy użyciu zewnętrznych urządzeń komunikacyjnych do równoważenia obciążenia. Rozmiar klastrów Roxie można dostosować do potrzeb, aby spełnić wymagania dotyczące przepustowości przetwarzania zapytań i czasu odpowiedzi, ale zazwyczaj są mniejsze niż klastry Thor. Implementacja dwóch typów platform równoległego przetwarzania danych (Thor i Roxie) w środowisku przetwarzania HPCC obsługujących różne potrzeby przetwarzania danych pozwala na optymalizację i dostrojenie tych platform do ich konkretnych celów, aby zapewnić użytkownikom najwyższy poziom wydajności systemu. Jest to wyraźna zaleta w porównaniu z platformą i architekturą Hadoop MapReduce, które muszą być nałożone na różne systemy, takie jak HBase, Hive i Pig, które mają różne cele i wymagania dotyczące przetwarzania i nie zawsze są łatwo mapowane do paradygmatu MapReduce. Ponadto podejście LexisNexis HPCC obejmuje ideę środowiska przetwarzania, które może integrować klastry Thor i Roxie w zależności od potrzeb w celu zaspokojenia pełnych potrzeb organizacji w zakresie przetwarzania. W rezultacie skalowalność może być zdefiniowana nie tylko pod względem liczby węzłów w klastrze, ale także pod względem liczby klastrów i jakiego typu jest potrzebnych do spełnienia celów wydajności systemu i wymagań użytkowników. Zapewnia to wyraźną przewagę w porównaniu z klastrami Hadoop, które zwykle są niezależnymi wyspami przetwarzania. LexisNexis HPCC jest komercyjnie dostępny do wdrażania środowisk chmury prywatnej (http://risk.lexisnexis.com/Article.aspx?id=51). Ponadto LexisNexis zapewnia klientom zewnętrznym hostowane, trwałe środowiska HPCC. W przyszłości planowana jest oferta PaaS w chmurze publicznej z wykorzystaniem platformy HPCC.

ECL

Język programowania ECL jest kluczowym czynnikiem elastyczności i możliwości środowiska przetwarzania HPCC. ECL został zaprojektowany jako przejrzysty i niejawnie równoległy język programowania dla aplikacji intensywnie korzystających z danych. Jest to wysokopoziomowy, deklaratywny, nieproceduralny język zorientowany na przepływ danych, który pozwala programiście zdefiniować, jaki powinien być wynik przetwarzania danych oraz przepływy danych i przekształcenia, które są niezbędne do osiągnięcia wyniku. Wykonanie nie zależy od kolejności instrukcji języka, ale od sekwencji przepływów danych i przekształceń reprezentowanych przez instrukcje języka. Łączy reprezentację danych z implementacją algorytmu i jest fuzją zarówno języka zapytań, jak i języka przetwarzania danych równoległych. ECL wykorzystuje intuicyjną składnię, która wzoruje się na innych znanych językach, obsługuje modułową organizację kodu z wysokim stopniem ponownego wykorzystania i rozszerzalności oraz wspiera wysoką produktywność programistów pod względem ilości kodu wymaganego do typowych aplikacji w porównaniu z tradycyjnymi językami jak Java i C++. Podobnie do korzyści, jakie Sawzall zapewnia w środowisku Google, a Pig zapewnia użytkownikom Hadoop, 20-krotny wzrost produktywności programisty jest typowym, znacznie skracającym cykle rozwoju. ECL jest kompilowany w zoptymalizowany kod C++ do wykonania na platformach systemowych HPCC i może być używany do złożonych zadań przetwarzania i analizy danych w klastrze Thor lub do kompleksowego przetwarzania zapytań i raportów i klaster Roxie. ECL umożliwia włączanie wbudowanych funkcji C++ do programów ECL, a programy zewnętrzne w innych językach mogą być włączane i równoległe za pomocą funkcji PIPE. Usługi zewnętrzne napisane w C++ i innych językach

generujących biblioteki DLL można również włączyć do biblioteki systemowej ECL, a programy ECL mogą uzyskać dostęp do zewnętrznych usług Web poprzez standardowy interfejs SOAPCALL. Podstawowa jednostka kodu ECL to atrybut. Atrybut może zawierać kompletne zapytanie lub program wykonywalny lub fragment kodu, który można udostępniać i ponownie wykorzystać, taki jak funkcja, definicja rekordu, definicja zestawu danych, makro, definicja filtra itp. Atrybuty mogą odwoływać się do innych atrybutów, które z kolei mogą odwoływać się do innych atrybutów, dzięki czemu kod ECL można zagnieżdżać i łączyć w razie potrzeby w sposób wielokrotnego użytku. Atrybuty są przechowywane w repozytorium kodów ECL, które jest podzielone na moduły zwykle związane z projektem lub procesem. Każdy atrybut ECL dodany do repozytorium skutecznie rozszerza język ECL, np. dodając nowe słowo do słownika, a atrybuty mogą być ponownie wykorzystywane jako część wielu zapytań i programów ECL. Wraz z ECL dostarczany jest bogaty zestaw narzędzi programistycznych, w tym interaktywne IDE podobne do Visual C++, Eclipse i innych środowisk programowania kodu. Język ECL zawiera rozbudowane możliwości definiowania danych, filtrowania, zarządzania danymi i przekształcania danych oraz zapewnia szeroki zestaw wbudowanych funkcji do operowania na rekordach w zestawach danych, które mogą obejmować funkcje transformacji zdefiniowane przez użytkownika. Funkcje transformacji działają na pojedynczym rekordzie lub parze rekordów na raz, w zależności od operacji. Wbudowane operacje transformacji w języku ECL, które przetwarzają całe zestawy danych, obejmują PROJECT, ITERATE, ROLLUP, JOIN, COMBINE, FETCH, NORMALIZE, DENORMALIZE i PROCESS. Funkcja transformacji zdefiniowana dla operacji JOIN na przykład otrzymuje dwa rekordy, po jednym z każdego połączonego zbioru danych, i może wykonywać dowolne operacje na polach w parze rekordów oraz zwraca rekord wyjściowy, który może być całkowicie różny od któregokolwiek z danych wejściowych dokumentacja. Inne ważne operacje na danych zawarte w ECL, które działają na zestawach danych i indeksach, obejmują TABLE, SORT, MERGE, MERGEJOIN, DEDUP, GROUP, APPLY, ASSERT, AVE, BUILD, BUILDINDEX, CHOOSESETS, CORRELATION, COUNT, COVARIANCE, DISTRIBUTE, DISTRIBUTION, ENTH, EXISTS, GRAPH, HAVING, KEYDIFF, KEYPATCH, LIMIT, LOOP, MAX, MIN, NONEMPTY, OUTPUT, PARSE, PIPE, PRELOAD, PULL, RANGE, REGROUP, SAMPLE, SET, SOAPCALL, STEPPED, SUM, TOPN, UNGROUP i VARIANCE. System Thor umożliwia wykonywanie operacji transformacji danych lokalnie na każdym węźle niezależnie w klastrze lub globalnie we wszystkich węzłach w klastrze, co może być określone przez użytkownika w języku ECL. Niektóre operacje, takie jak PROJECT, na przykład, są z natury operacjami lokalnymi w części rozproszonego pliku przechowywanego lokalnie w węźle. Inne, takie jak SORT, można w razie potrzeby przeprowadzić lokalnie lub globalnie. Jest to znacząca różnica w porównaniu z architekturą MapReduce, w której operacje mapowania i zmniejszania są wykonywane tylko lokalnie w podziale wejściowym przypisanym do zadania. Lokalna operacja SORT w klastrze HPCC posortowałaby rekordy według określonego klucza w części pliku w węźle lokalnym, w wyniku czego rekordy byłyby posortowane w węźle lokalnym, ale nie w pełnej kolejności obejmującej wszystkie węzły. W przeciwieństwie do tego, globalna operacja SORT spowodowałaby, że pełny plik rozproszony byłby posortowany według określonego klucza obejmującego wszystkie węzły. Wymaga to przeniesienia danych z węzła do węzła podczas operacji SORT. Rysunek przedstawia przykładowy program ECL wykorzystujący tryb pracy LOCAL.



Zwróć uwagę na jawną kontrolę programisty nad dystrybucją danych między węzłami. Operator równy dwukropkowi „: =" w programie ECL jest odczytywany jako „jest zdefiniowany jako". Jedyną akcją w tym programie jest instrukcja OUTPUT, pozostałe instrukcje to definicje. Dodatkową ważną funkcją dostępną w języku programowania ECL jest obsługa przetwarzania języka naturalnego (NLP) za pomocą instrukcji PATTERN i wbudowanej funkcji PARSE. Funkcja PARSE cam akceptuje jednoznaczną gramatykę zdefiniowaną przez wyrażenia WZORZEC, TOKEN i REGUŁA z karami lub preferencjami, aby zapewnić deterministyczny wybór ścieżki, co może znacznie zmniejszyć trudność aplikacji NLP. Instrukcje PATTERN umożliwiają definiowanie pasujących wzorców, w tym wyrażeń regularnych, i używanie ich do analizowania informacji z danych nieustrukturyzowanych, takich jak nieprzetworzony tekst. Instrukcje PATTERN można łączyć w celu implementacji złożonych operacji analizowania lub kompletnych gramatyk z definicji BNF. Funkcja operacji PARSE w zbiorze danych rekordów w określonym polu w rekordzie, to pole może być na przykład całym wierszem w pliku tekstowym. Korzystając z tej możliwości języka ECL, można wdrożyć przetwarzanie równoległe dla aplikacji do ekstrakcji informacji z plików dokumentów, w tym dokumentów opartych na XML lub stron internetowych. Kluczowe korzyści płynące z ECL można podsumować w następujący sposób:

• ECL obejmuje transparentną i niejawną równoległość danych niezależnie od wielkości klastra obliczeniowego i zmniejsza złożoność programowania równoległego, zwiększając produktywność twórców aplikacji.

• ECL umożliwia wdrażanie aplikacji intensywnie korzystających z danych z ogromnymi ilościami danych, które wcześniej uważano za niewykonalne lub niewykonalne. ECL został specjalnie zaprojektowany do manipulowania danymi i przetwarzaniem zapytań. Możliwy jest wzrost wydajności rzędu wielkości w porównaniu z innymi podejściami.

• ECL zapewnia wszechstronne środowisko IDE i narzędzia programistyczne, które zapewniają wysoce interaktywne środowisko do szybkiego tworzenia i wdrażania aplikacji ECL.

• ECL to potężny, wysokopoziomowy język programowania równoległego, idealny do implementacji ETL, wyszukiwania informacji, ekstrakcji informacji i innych aplikacji intensywnie korzystających z danych.

• ECL jest językiem dojrzałym i sprawdzonym, ale wciąż ewoluuje w miarę pojawiania się nowych postępów w przetwarzaniu równoległym i przetwarzaniu intensywnie przetwarzającym dane.

Porównanie Hadoop vs. HPCC

Hadoop i HPCC można porównać bezpośrednio, ponieważ oba systemy mogą być wykonywane na identycznych konfiguracjach sprzętowych klastra. Umożliwia to bezpośrednią analizę porównawczą

wydajności systemu przy użyciu standardowego obciążenia lub zestawu aplikacji zaprojektowanych do testowania możliwości równoległego przetwarzania danych w każdym systemie. Standardowym testem porównawczym dostępnym dla platform obliczeniowych intensywnie przetwarzających dane jest test porównawczy Terasort zarządzany przez grupę branżową kierowaną przez Microsoft i HP. Sortowanie terabajtowe przekształciło się w GraySort, który mierzy liczbę terabajtów na minutę, które można sortować na platformie, która umożliwia wykorzystanie klastrów z dowolną liczbą węzłów. Jednak przy porównywaniu skuteczności i równoważnych kosztów/wydajności systemów przydatne jest przeprowadzanie testów porównawczych na identycznych konfiguracjach sprzętowych systemu. Tutaj zostanie zaprezentowane bezpośrednie porównanie oryginalnego sortowania terabajtowego w klastrze 400 węzłów. Dodatkową metodą porównywania platform systemowych jest porównanie cech i funkcjonalności, które jest subiektywną oceną na podstawie czynników określonych przez oceniającego. Chociaż takie porównanie zawiera nieodłączną stronniczość, jest przydatne przy określaniu mocnych i słabych stron systemów.

Test porównawczy terabajtów

Benchmark sortowania Terabyte ma swoje korzenie w sortowaniu testów porównawczych przeprowadzanych na systemach komputerowych od lat 80. XX wieku. Niedawno witryna internetowa pierwotnie sponsorowana przez firmę Microsoft i jednego z jej naukowców zajmujących się badaniami, Jima Graya, co roku przeprowadzała formalne konkursy z wynikami prezentowanymi na konferencji SIGMOD (Special Interest Group for Management of Data) sponsorowanej co roku przez ACM (http: //sortbenchmark.org). Istnieje kilka kategorii sortowania w systemach, w tym sortowanie terabajtowe, które miało mierzyć, jak szybko można posortować plik 1 terabajta danych sformatowanych w 100-bajtowych rekordach (łącznie 10 000 000 rekordów). Dopuszczono dwie kategorie: Daytona (standardowy komercyjny system komputerowy i oprogramowanie bez modyfikacji) oraz Indy (niestandardowy system komputerowy z dowolnym rodzajem modyfikacji). Nie istniały żadne ograniczenia dotyczące rozmiaru systemu, więc test porównawczy sortowania można było przeprowadzić na tak dużym systemie, jak jest to pożądane. Obecnym rekordzistą 2009 roku w kategorii Daytona jest Yahoo! przy użyciu konfiguracji Hadoop z 1460 węzłami z 8 GB pamięci RAM na węzeł, 8000 zadań map i 2700 zadaniami redukcji, które sortowały 1 TB w 62 sekundy (O'Malley & Murthy, 2009). W 2008 roku Yahoo! wykonał benchmark w 3 minuty 29 sekund. W 2008 r. firma LexisNexis wykorzystująca architekturę HPCC tylko w systemie 400-węzłowym przeprowadziła test porównawczy terabajtów w ciągu 3 minut i 6 sekund. W 2009 r. LexisNexis ponownie, używając tylko konfiguracji 400-węzłowej, przeprowadził test porównawczy terabajtów w 102 sekundy. Jednak uczciwym i bardziej logicznym porównaniem możliwości systemów obliczeniowych intensywnie korzystających z danych i architektur oprogramowania wykorzystujących klastry obliczeniowe byłoby przeprowadzenie tego testu porównawczego na tej samej konfiguracji sprzętowej. Należy również ocenić inne czynniki, takie jak ilość kodu wymaganego do wykonania testu porównawczego, który jest silnym wskaźnikiem produktywności programisty, który sam w sobie jest istotnym czynnikiem wydajności przy wdrażaniu aplikacji obliczeniowych intensywnie korzystających z danych. W dniu 8 sierpnia 2009 r. przeprowadzono test porównawczy Terabyte Sort na konfiguracji rozwojowej znajdującej się w biurach LexisNexis Risk Solutions w Boca Raton, Floryda we współpracy z Lawrence Livermore National Labs (LLNL) i przez nie zweryfikowane. Klaster testowy obejmował 400 węzłów przetwarzania, każdy z dwoma lokalnymi dyskami SCSI o pojemności 300 MB, jednordzeniowe procesory Dual Intel Xeon działające z częstotliwością 3,00 GHz, 4 GB pamięci na węzeł, wszystkie podłączone do jednego przełącznika Gigabit Ethernet o przepustowości 1,4 terabajta/s. W klastrze wdrożono Hadoop Release 0.19, a do testu porównawczego użyto standardowego testu porównawczego Terasort napisanego w języku Java, zawartego w wydaniu. Hadoop potrzebował 6 minut 45 sekund na utworzenie danych testowych, a test porównawczy Terasort wymagał łącznie 25

minut 28 sekund na ukończenie testu sortowania, jak pokazano na rys. 5.13. Oprogramowanie systemowe HPCC wdrożone na tej samej platformie i przy użyciu standardowego ECL wymagało 2 minut i 35 sekund na utworzenie danych testowych oraz łącznie 6 minut i 27 sekund na ukończenie testu sortowania . W związku z tym implementacja Hadoop przy użyciu Javy działającej na tej samej konfiguracji sprzętowej zajęła 3,95 razy dłużej niż implementacja HPCC przy użyciu ECL. Wersja testu porównawczego Hadoop wykorzystywała ręcznie dostrojony kod Java, w tym niestandardowe klasy TeraSort, TeraInputFormat i TeraOutputFormat z łącznie 562 wierszami kodu wymaganymi do sortowania. System HPCC wymagał do sortowania tylko 10 wierszy kodu ECL, co oznacza 50-krotne zmniejszenie ilości wymaganego kodu.

Pig kontra ECL

Chociaż wiele instalacji Hadoop implementuje aplikacje bezpośrednio w Javie, język Pig Latin jest obecnie używany do zwiększenia produktywności programistów i dalszego uproszczenia programowania aplikacji intensywnie korzystających z danych w Yahoo! i innych głównych użytkowników Hadoop. Google dodał też język wysokiego poziomu z podobnych powodów wezwał Sawzall do wdrożenia MapReduce w celu ułatwienia analizy danych i eksploracji danych (Pike i in., 2004). Platforma HPCC zawiera omówiony wcześniej język wysokiego poziomu, analogiczny do Pig i Sawzall, zwany ECL. ECL jest podstawowym językiem programowania używanym w aplikacjach na platformie HPCC, mimo że jest skompilowany do C++ w celu wykonania. Porównując platformy Hadoop i HPCC, warto porównać cechy i funkcjonalność tych języków wysokiego poziomu. Zarówno Pig, jak i ECL są wewnętrznie równoległe, wspierając transparentną równoległość danych na platformie bazowej. Pig i ECL są tłumaczone na programy, które automatycznie przetwarzają dane wejściowe dla procesu równolegle z danymi rozproszonymi w klastrze węzłów. Programiści obu języków nie muszą znać bazowego rozmiaru klastra ani używać go do wykonywania zadań równolegle do danych. Zarówno Pig, jak i ECL są zorientowane na przepływ danych, ale Pig jest imperatywnym językiem programowania, a ECL jest deklaratywnym językiem programowania. Język deklaratywny pozwala programistom skupić się na przekształceniach danych wymaganych do rozwiązania problemu z aplikacją i ukrywa złożoność podstawowej platformy i szczegółów implementacji, zmniejsza skutki uboczne i ułatwia optymalizację kodu i planu wykonania przez kompilator. Imperatywny język programowania dyktuje przepływ sterowania programem, co może nie skutkować idealnym planem wykonania w środowisku równoległym. Deklaratywne języki programowania pozwalają programiście określić „co" program powinien osiągnąć, zamiast „jak" to zrobić. Aby uzyskać więcej informacji, zapoznaj się z dyskusjami deklaratywnymi http://en.wikipedia.org/wiki/Declarative_programming) i imperatywne (http://en.wikipedia.org/wiki/Imperative_programming) języki programowania w Wikipedii. Kod źródłowy zarówno Pig, jak i ECL jest kompilowany lub tłumaczony na inny język – programy źródłowe Pig są tłumaczone na język Java zadania MapReduce do wykonania i programy ECL są tłumaczone na kod źródłowy C++, który jest następnie kompilowany do biblioteki DLL w celu wykonania. Programy Pig są ograniczone do architektury MapReduce i HDFS Hadoop, ale ECL nie ma ustalonej struktury innej niż DFS (Distributed File System) używanej przez HPCC i dlatego może być bardziej elastyczna we wdrażaniu operacji na danych. Jest to widoczne w dwóch kluczowych obszarach: (1) ECL umożliwia operacje globalne lub lokalne, przy czym standardowe MapReduce jest ograniczone do operacji lokalnych tylko w fazach Map i Reduce. Operacje globalne przetwarzają rekordy w zbiorze danych w kolejności we wszystkich węzłach i powiązanych częściach plików w kolejności, utrzymując rekordy w kolejności posortowanej, w przeciwieństwie do tylko rekordów zawartych w każdym węźle lokalnym, które mogą być ważne dla procedury przetwarzania danych; (2) ECL ma elastyczność we wdrażaniu operacji, które mogą przetwarzać więcej niż jeden rekord jednocześnie, takich jak operacja ITERATE, która wykorzystuje przesuwane okno i przekazuje jednocześnie dwa rekordy do powiązanej funkcji transformacji. Pozwala to na współzależności między

rekordami pole po polu i decyzje, które nie są dostępne w Pig. Na przykład operacja DISTINCT w Pig, która służy do usuwania duplikatów, nie pozwala na to w podzbiorze pól. ECL zapewnia zarówno operacje DEDUP, jak i ROLLUP, które są zwykle poprzedzone sortowaniem i działają na sąsiednich rekordach w trybie okna przesuwnego, a wszelkie warunki dotyczące zawartości pól lewego i prawego rekordu sąsiednich rekordów mogą być wykorzystane do określenia, czy rekord jest usunięty. ROLLUP umożliwia zastosowanie niestandardowej transformacji w procesie deduplikacji. Ważnym elementem każdej architektury oprogramowania dla danych jest podstawowy model danych. Pig zawiera bardzo elastyczny zagnieżdżony model danych, który umożliwia nieatomowe typy danych (atomowe typy danych obejmują liczby i ciągi), takie jak zestaw, mapa i krotka, występują jako pola tabeli. Krotki to sekwencje pól, torby to kolekcje krotek, a mapy to zbiór elementów danych, w których każdy element danych ma klucz, za pomocą którego można go wyszukać. Rekord danych w Pig nazywa się relacją, która jest zewnętrzną torbą, torba jest zbiorem krotek, każda krotka jest uporządkowanym zestawem pól, a pole jest fragmentem danych. Do relacji odwołuje się nazwa przypisana przez użytkownika. Typy mogą być przypisane przez użytkownika do każdego pola, ale jeśli nie zostaną przypisane, domyślnie będą to tablice bajtów, a konwersje są stosowane w zależności od kontekstu, w którym pole jest używane. Model danych ECL oferuje również zagnieżdżoną strukturę danych przy użyciu podrzędnych zestawów danych. Określona przez użytkownika definicja RECORD definiuje zawartość każdego rekordu w zestawie danych, który może zawierać pola o stałej lub zmiennej długości lub podrzędne zestawy danych, które z kolei zawierają pola lub podrzędne zestawy danych itp. Za pomocą tego formatu można przedstawić dowolny typ struktury danych. ECL oferuje specyficzną obsługę formatów CSV i XML oprócz płaskich formatów plików. Każde pole w rekordzie ma określony przez użytkownika identyfikator i typ danych oraz opcjonalną wartość domyślną i opcjonalne modyfikatory pól, takie jak MAXLENGTH, które poprawiają typ i używają sprawdzania podczas kompilacji. ECL wykona niejawne rzutowanie i konwersję w zależności od kontekstu, w którym używane jest pole, a także obsługiwane jest jawne rzutowanie użytkownika. ECL umożliwia również wbudowane zestawy danych, co pozwala na łatwe definiowanie przykładowych danych i włączanie ich do kodu w celu testowania, a nie oddzielnie w pliku. Środowisko Pig oferuje kilka narzędzi programistycznych do tworzenia, wykonywania i debugowania programów Pig Latin (Pig Latin to formalna nazwa języka, a środowisko wykonawcze nazywa się Pig, chociaż oba są powszechnie określane jako Pig). Pig zapewnia wykonywanie skryptów z wiersza poleceń i interaktywną powłokę o nazwie Grunt, która umożliwia wykonywanie poszczególnych poleceń Pig lub wykonanie skryptu Pig. Programy Pig można również osadzać w programach Java. Chociaż Pig nie zapewnia określonego IDE do tworzenia i wykonywania programów PIG, dostępne są dodatki dla kilku środowisk edycji programów, w tym Eclipse, Vim i Textmate, umożliwiające sprawdzanie i podświetlanie składni (White, 2009). PigPen to wtyczka do środowiska Eclipse, która umożliwia edycję programu, przykładowy generator danych oraz możliwość uruchamiania skryptu Pig w klastrze Hadoop. Platforma HPCC zapewnia obszerny zestaw narzędzi do opracowywania ECL, w tym kompleksowe IDE o nazwie QueryBuilder, które umożliwia edycję, wykonywanie i interaktywną wizualizację wykresów w celu debugowania i profilowania programów ECL. Wspólne drzewo repozytorium kodu jest wyświetlane w QueryBuilder, a narzędzia są dostarczane do kontroli źródła, uzyskiwania dostępu i przeszukiwania repozytorium. Zadania ECL można uruchamiać w środowisku HPCC lub określonym klastrze, a wykonanie można monitorować bezpośrednio z QueryBuilder. Dostępne są również narzędzia zewnętrzne, w tym ECLWatch, który zapewnia pełny dostęp do bieżących i historycznych jednostek roboczych (zadania wykonywane w środowisku HPCC są pakowane w jednostki robocze), zarządzanie kolejką i monitorowanie, wizualizację grafu wykonania, funkcje narzędzi rozproszonego systemu plików oraz monitorowanie i analizę wydajności systemu. Chociaż Pig Latin i środowisko wykonawcze Pig zapewniają podstawowe środowisko językowe wysokiego poziomu do przetwarzania i analizy dużej ilości danych oraz zwiększają produktywność programistów i użytkowników środowiska Hadoop MapReduce, ECL jest

znacznie bardziej wszechstronnym i dojrzałym językiem, który generuje wysoce zoptymalizowany kod, oferuje bardziej zaawansowane możliwości w solidnej, sprawdzonej, zintegrowanej architekturze przetwarzania intensywnie przetwarzającej dane.

Porównanie architektury

Hadoop MapReduce i platforma LexisNexis HPCC to skalowalne architektury skierowane do rozwiązań obliczeniowych intensywnie korzystających z danych. Każda z tych platform systemowych ma mocne i słabe strony, a ich ogólna skuteczność w przypadku dowolnego problemu lub dziedziny aplikacji jest z natury subiektywna i można ją określić jedynie poprzez dokładną ocenę wymagań aplikacji w porównaniu z możliwościami rozwiązania. Hadoop to platforma typu open source, która zwiększa jej elastyczność i możliwość dostosowania do wielu problematycznych dziedzin, ponieważ użytkownicy korzystający z tej technologii mogą łatwo dodać nowe możliwości. Jednak, podobnie jak w przypadku innych platform open source, niezawodność i wsparcie mogą stać się problemami, gdy wielu różnych użytkowników wnosi nowy kod i zmiany do systemu. Hadoop znalazł uznanie w wielu dużych firmach internetowych, w tym Yahoo!, Facebooku i innych, w których możliwości przetwarzania dużej ilości danych mają kluczowe znaczenie dla powodzenia ich działalności. Amazon wdrożył nowe usługi przetwarzania w chmurze przy użyciu Hadoop w ramach swojego EC2 o nazwie Amazon Elastic MapReduce. Niedawno powstała firma Cloudera, która ma świadczyć usługi szkoleniowe, wsparcia i konsultingowe dla społeczności użytkowników Hadoop oraz dostarczać pakiety i przetestowane wersje, które można wykorzystać w środowisku Amazon. Chociaż na platformie Hadoop zbudowano wiele różnych narzędzi aplikacji, takich jak Pig, HBase, Hive itp., narzędzia te zwykle nie są dobrze zintegrowane, oferując różne powłoki poleceń, języki i cechy operacyjne, które utrudniają łączenie zdolności w efektywny sposób. Jednak Hadoop oferuje wiele korzyści potencjalnym użytkownikom oprogramowania open source, w tym łatwo dostępne dystrybucje oprogramowania i dokumentację online, łatwą instalację, elastyczne konfiguracje oparte na powszechnie dostępnym sprzęcie, środowisko wykonawcze oparte na sprawdzonym paradygmacie obliczeniowym MapReduce, możliwość planowania zadań za pomocą konfigurowalnego liczba zadań Map and Reduce, dostępność dodatkowych funkcji, takich jak Pig, HBase i Hive, aby rozszerzyć możliwości platformy podstawowej i poprawić produktywność programistów, oraz szybko rozwijający się użytkownik społeczności zaangażowanej w open source. Doprowadziło to do gwałtownego rozwoju i akceptacji platformy Hadoop oraz jej implementacji do obsługi aplikacji obliczeniowych intensywnie korzystających z danych. Platforma LexisNexis HPCC to zintegrowany zestaw systemów, oprogramowania i innych elementów architektonicznych zaprojektowanych w celu zapewnienia możliwości przetwarzania dużej ilości danych, od przetwarzania danych pierwotnych i aplikacji ETL po wysokowydajne przetwarzanie zapytań i eksplorację danych. Język ECL został specjalnie zaimplementowany w celu zapewnienia języka przetwarzania równoległego przepływu danych wysokiego poziomu, który jest spójny we wszystkich komponentach systemu i ma rozbudowane możliwości opracowane i zoptymalizowane przez okres prawie 10 lat. LexisNexis HPCC to dojrzała, niezawodna, dobrze sprawdzona, komercyjnie obsługiwana platforma systemowa wykorzystywana w instalacjach rządowych, laboratoriach badawczych i przedsiębiorstwach komercyjnych. Przedstawione tutaj porównanie języka Pig Latin i systemu wykonawczego dostępnego na platformie Hadoop MapReduce z językiem ECL używanym na platformie HPCC pokazuje, że ECL zapewnia znacznie bardziej zaawansowane możliwości i funkcjonalność bez potrzeby korzystania z rozbudowanych funkcji zdefiniowanych przez użytkownika napisanych w innym języku lub uciekając się do natywnej aplikacji MapReduce zakodowanej w Javie. Poniższe porównanie ogólnych funkcji zapewnianych przez architektury systemów Hadoop i HPCC pokazuje, że architektura HPCC oferuje wyższy poziom integracji komponentów systemu, środowisko wykonawcze nieograniczone określonym paradygmatem obliczeniowym, takim jak MapReduce, elastyczne konfiguracje i zoptymalizowane środowiska przetwarzania, które może dostarczać aplikacje

wymagające dużej ilości danych, od analizy danych po hurtownie danych i wysokowydajne przetwarzanie zapytań online, a także wysoką produktywność programistów dzięki wykorzystaniu języka programowania i narzędzi ECL.

Podsumowanie

W wyniku nieustannej eksplozji informacji wiele organizacji tonie w danych, a luka w danych lub niemożność ich przetwarzania i efektywnego wykorzystania rośnie w zastraszającym tempie. Przetwarzanie intensywnie wykorzystujące dane reprezentuje nowy paradygmat obliczeniowy, który może zająć się luką w danych i umożliwić organizacjom rządowym i komercyjnym oraz środowiskom badawczym przetwarzanie ogromnych ilości danych i wdrażanie aplikacji, które wcześniej uważano za niepraktyczne lub niewykonalne. Niektóre organizacje przewidujące wcześnie zauważyły, że potrzebne są nowe architektury przetwarzania równoległego, w tym Google, który początkowo opracował architekturę MapReduce i LexisNexis, który opracował architekturę HPCC. Niedawno platforma Hadoop pojawiła się jako alternatywa open source dla podejścia MapReduce. Hadoop szybko nabrał rozpędu i opracowano dodatkowe funkcje rozszerzające platformę, w tym język programowania przepływu danych i środowisko wykonawcze o nazwie Pig. Te architektury, ich krewni . W niniejszej Części opisano mocne i słabe strony oraz ich zastosowanie w chmurze obliczeniowej, a także przedstawiono bezpośrednie porównanie języka Pig Hadoop z językiem ECL używanym z platformą LexisNexis HPCC. Dostępność języka programowania wysokiego poziomu zorientowanego na równoległy przepływ danych okazała się krytycznym czynnikiem sukcesu w obliczeniach intensywnie przetwarzających dane. Przydatność platformy przetwarzania i architektury dla organizacji oraz jej wymagania dotyczące aplikacji można określić tylko po starannej ocenie dostępnych alternatyw. Wiele organizacji przyjęło platformy open source, podczas gdy inne preferują platformę opracowaną komercyjnie i wspieraną przez uznanego lidera branży. Platforma Hadoop MapReduce jest obecnie z powodzeniem stosowana w wielu tak zwanych firmach internetowych, których dane jako źródło danych obejmują ogromne ilości informacji internetowych. Platforma LexisNexis HPCC jest sercem wiodącego dostawcy usług informacyjnych i lidera w branży i została przyjęta przez agencje rządowe, organizacje komercyjne i laboratoria badawcze ze względu na jej wydajne i opłacalne wdrożenie. Istniejące aplikacje HPCC obejmują przetwarzanie surowych danych, ETL i łączenie ogromnych ilości danych ze wspieraniem internetowe usługi informacyjne, takie jak LexisNexis i wiodące w branży aplikacje do wyszukiwania informacji, takie jak Accurint; wyodrębnianie jednostek i rozwiązywanie jednostek nieustrukturyzowanych i częściowo ustrukturyzowanych danych, takich jak dokumenty internetowe, w celu wsparcia ekstrakcji informacji; analiza statystyczna logów internetowych dla aplikacji zabezpieczających, takich jak wykrywanie włamań; przetwarzanie analityczne online w celu wsparcia systemów Business Intelligence (BIS); oraz analiza danych z ogromnych zbiorów danych w środowiskach edukacyjnych i badawczych oraz przez agencje rządów stanowych i federalnych. Istnieje wiele kompromisów przy podejmowaniu właściwej decyzji przy wyborze nowej architektury systemów komputerowych, a często najlepszym podejściem jest przeprowadzenie określonego testu porównawczego z aplikacją klienta w celu określenia ogólnej skuteczności i wydajności systemu. Należy wziąć pod uwagę względną charakterystykę kosztów i wydajności systemu, oprócz przydatności, elastyczności, skalowalności, zajmowanego miejsca i zużycia energii, które wpływają na całkowity koszt posiadania (TCO). Jeśli zasoby wewnętrzne są ograniczone, należy również rozważyć alternatywy przetwarzania w chmurze, które ograniczają lub eliminują początkowe inwestycje w infrastrukturę. Porównanie architektury Hadoop MapReduce z architekturą HPCC w tym rozdziale ujawnia wiele podobieństw między platformami, w tym użycie języka programowania wysokiego poziomu zorientowanego na przepływ danych w celu implementacji przejrzystego przetwarzania równoległego danych. Obie platformy można dostosować do przetwarzania w chmurze, aby zapewnić platformę jako usługę (PaaS). Kluczową zaletą korzystania z architektury Hadoop jest jej dostępność w ofercie usługi

chmury publicznej. Jednak przetwarzanie w chmurze prywatnej, które wykorzystuje trwałe konfiguracje z dedykowaną infrastrukturą zamiast serwerów zwirtualizowanych współdzielonych z innymi użytkownikami typowymi dla chmury publicznej, może mieć znaczną przewagę wydajności w przypadku aplikacji obliczeniowych intensywnie korzystających z danych. Niektóre dodatkowe korzyści wynikające z wyboru platformy LexisNexis HPCC, którą można wykorzystać w prywatnej chmurze obliczeniowej, obejmują: (1) architekturę, która implementuje wysoce zintegrowane środowisko systemowe z możliwościami od przetwarzania surowych danych po wysokowydajne zapytania i analizę danych przy użyciu wspólnego języka; (2) architektura, która zapewnia równoważną wydajność przy znacznie niższych kosztach systemu w oparciu o liczbę wymaganych węzłów przetwarzania, jak wykazano w teście Terabyte Sort, w którym platforma HPCC była prawie 4 razy szybsza niż Hadoop działająca na tym samym sprzęcie, co skutkuje znacznie niższym całkowity koszt posiadania (TCO); (3) architektura, która okazała się stabilna i niezawodna w wysokowydajnych aplikacjach produkcyjnych do przetwarzania danych dla różnych organizacji przez okres 10 lat; (4) architektura wykorzystująca język programowania przepływu danych (ECL) z rozbudowanymi wbudowanymi możliwościami przetwarzania równoległego danych, która umożliwia złożone operacje bez potrzeby korzystania z rozbudowanych funkcji zdefiniowanych przez użytkownika i automatycznie optymalizuje wykresy wykonania z setkami kroków przetwarzania w jednym wydajne jednostki robocze; (5) architekturę o wysokim poziomie odporności na błędy i możliwości językowych, które zmniejszają potrzebę ponownego przetwarzania w przypadku awarii systemu; oraz (6) architekturę, która jest dostępna i wspierana przez znanego lidera usług informacyjnych i rozwiązań zarządzania ryzykiem (LexisNexis), który jest częścią jednego z największych na świecie wydawców informacji, ReedElsevier.