Arthur MATEOS Jothy ROSENBERG

MANNING

Chmura obliczeniowa

ROZWIĄZANIA DLA BIZNESU

ODKRYJ WIELKĄ MOC CHMUR OBLICZENIOWYCH I POTENCJAŁ, JAKI KRYJĄ W SOBIE!

- Co to jest chmura obliczeniowa?
- Kiedy korzystać z chmur, a kiedy ich unikać?
- Jak oszacować koszty korzystania z chmury obliczeniowej?

Helion

Tytuł oryginału: The Cloud at Your Service

Tłumaczenie: Justyna Walkowska
Projekt okładki: Jan Paluch

ISBN: 978-83-246-3416-3

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: http://helion.pl (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/chmura
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

2 Klasyfikacja chmur obliczeniowych

W tym rozdziale:

- podstawy technologiczne wspólne dla wszystkich typów chmur,
- klasyfikacja typów chmur i ich możliwości,
- zasady wyboru chmury odpowiedniego typu i najlepszego dostawcy.

Poprzedni rozdział był wprowadzeniem do świata przetwarzania w chmurze. W tym zajmiemy się bardziej szczegółową analizą różnych typów chmur i charakterystycznych dla nich sposobów działania. Za przykład niech nam posłuży samochód — jeśli chmura jest samochodem, to nowoczesne centrum danych odgrywa rolę silnika, a wirtualizacja odpowiada resorom, chroniącym nas na wyboistej drodze. API chmury przypomina deskę rozdzielczą, pozwalającą kontrolować chmurę. Magazyn danych działa jak bagażnik, umożliwiający transportowanie różnych rzeczy. Bazy danych w chmurze to system nawigacyjny — konkretne informacje niezbędne w podróży. W zależności od potrzeb samochód elastycznie przestawia się na inny bieg — można porównać to do elastyczności aplikacji, która może obsługiwać jednego użytkownika, a chwilę później rozszerza się tak, że możliwa jest obsługa miliona zgłoszeń. Istnieje wiele modeli pojazdów i wiele typów chmur. Przyjrzymy się z bliska dominującym dzisiaj typom. Ocenimy, czy potrzebujesz wyścigówki, gdyż zależy Ci na prędkości, czy może wolisz osiemnastokołowego olbrzyma, który zapewni Ci mnóstwo przestrzeni.

Na początek przeanalizujmy najbardziej niezbędne aspekty technologiczne chmury, by dobrze zrozumieć, z jakich elementów jest ona zbudowana. W rozdziale 1. omówiliśmy wstępnie różne typy chmur i porównaliśmy je ze sobą. Teraz rozwiniemy to zagadnienie, by ułatwić Ci podjęcie decyzji na temat wyboru typu chmury oraz wspomóc najbardziej efektywne jej wykorzystanie.

2.1. Podstawy technologiczne przetwarzania w chmurze

Większość kierowców zna podstawy działania samochodu — niektórzy poznali je z czystej ciekawości, innym zależało na byciu lepszymi kierowcami i właścicielami auta. W przypadku chmury również możliwe jest wyodrębnienie ogólnych zasad działania niezależnych od typu. Omówimy je dla lepszego zrozumienia późniejszych kwestii:

- *Chmura potrzebuje serwerów sieciowych, te zaś potrzebują domu.* Serwery zgromadzone w pewnej fizycznej lokalizacji będziemy określali mianem centrum danych.
- *Serwery w chmurze należy zwirtualizować.* Celem tego procesu jest zwiększenie efektywności banku serwerów. W przeciwnym razie koszty utrzymania serwerów obniżą efektywność finansową chmury.
- *Chmura potrzebuje API.* Bez interfejsu dostępowego zwirtualizowane serwery w chmurze tkwiłyby w ciszy i samotności. Użytkownicy muszą mieć dostęp do chmury. Chcą zamawiać nowe serwery, wysyłać i pobierać dane, uruchamiać i zatrzymywać aplikacje, a także pozbywać się serwerów, które przestały już być potrzebne. Wszystko to ma się odbywać zdalnie, gdyż stopa użytkowników nigdy nie postanie w fizycznej serwerowni.

- *Chmura musi gdzieś przechowywać dane*. Musi składować obrazy maszyn wirtualnych, aplikacje użytkowników i wykorzystywane przez nie trwałe dane.
- *Chmura wymaga bazy danych*. Większość aplikacji do działania potrzebuje ustrukturyzowanych danych — w konsekwencji potrzebuje ich także chmura.
- *Chmura musi być elastyczna i skalować się dynamicznie zgodnie z potrzebami aplikacji i ich użytkowników*. Dla użytkowników chmury możliwość dostosowania ilości wykorzystywanych zasobów (oraz płatności za nie) do aktualnego obciążenia aplikacji to jedna z największych korzyści z przejścia na chmurę.

Omówimy teraz bardziej szczegółowo każdy z wymienionych sześciu aspektów technologicznych i infrastrukturalnych.

2.1.1. Duże korzyści skali dzięki centrom danych w chmurze

Wróćmy do analogii samochodowej — centrum danych to silnik samochodu. Centrum danych, posiadane obecnie przez każdą większą instytucję, to obiekt (nierzadko pilnie strzeżony), w którym znajduje się wiele komputerów i innego sprzętu sieciowo-komunikacyjnego. Duże korporacje internetowe, takie jak Amazon, Yahoo, Google, Intuit czy Apple, w ciągu lat zgromadziły *mega*centra danych z tysiącami serwerów. Stanowią one punkt wyjścia infrastruktury przetwarzania w chmurze.

Warto dobrze zrozumieć strukturę i ekonomikę tych ogromnych centrów danych. To na nich oparte są mechanizmy skalowania, niezawodność przetwarzania, bezpieczeństwo danych i przyszłość rozwoju chmur publicznych. Rozważ te kwestie, zwłaszcza jeśli myślisz o stworzeniu własnej chmury prywatnej. Jeszcze w tym rozdziale opowiemy co nieco o chmurach prywatnych. Dodatkowo cały rozdział 4. jest poświęcony chmurom prywatnym i bezpieczeństwu.

Struktura centrum danych

Centrum danych może mieścić się w jednym pokoju, zajmować jedno albo więcej pięter, a nawet objąć we władanie cały budynek. Większość jego wyposażenia z reguły stanowią serwery upakowane w dziewiętnastocalowych szafach, najczęściej ustawianych w rzędach, pomiędzy którymi tworzą się korytarze umożliwiające dostęp do serwerów z dwóch stron. Serwery różnią się rozmiarami. Serwer wielkości 1U (ang. *rack unit*) zajmuje jeden slot (z czterdziestu dwóch) w szafie montażowej, ale istnieją także samostojące serwery, których nie montuje się w szafach. Komputery typu *mainframe* oraz niektóre magazyny danych mogą być tak duże jak szafy i są ustawiane w szeregach razem z nimi. Duże centra danych czasami korzystają z kontenerów po tysiąc lub więcej serwerów każdy. Jeśli potrzebna jest naprawa lub aktualizacja, wymienia się cały kontener, a nie poszczególne serwery.

Niezbędny jest czysty, stabilny dopływ energii. Komputery w centrach danych działają bez przerwy. Muszą być odporne na spadki napięcia, a nawet przerwy w dostawach prądu. Serwerownia musi być zaopatrzona w stabilizator napięcia, zapasowe baterie oraz generatory prądu, zapewniające nieograniczony, nieprzerwany dopływ prądu. Ważne jest także chłodzenie sprzętu. Najczęściej stosuje się do tego schłodzone powietrze, jednak pojawiają się również systemy wykorzystujące wodę, jeśli jest ona łatwo dostępna. Wodą chłodzone są na przykład nowe centra danych położone wzdłuż rzeki Kolumbia w stanie Waszyngton. Klimatyzacja ma za zadanie nie tylko obniżenie temperatury, lecz także kontrolę wilgotności, by nie doszło do skraplania się lub wyładowań elektrostatycznych.

Centrum danych wymaga szerokopasmowego łącza umożliwiającego odbiór i wysyłanie danych. Istnienie serwerów i magazynów danych nie ma sensu, jeśli nikt nie może się z nimi połączyć.

Ważne jest także bezpieczeństwo, zarówno fizyczne, jak i logiczne. Duże centra danych są pod ciągłym ostrzałem hakerów z całego świata. Procedury bezpieczeństwa w niektórych centrach danych zaczynają się od tego, że ukrywane jest samo istnienie centrum w danej lokalizacji. Strażnicy, pułapki i najnowocześniejsze mechanizmy autentykacji mają za zadanie fizycznie powstrzymać nieproszonych gości. Firewalle, bramki VPN, oprogramowanie wykrywające włamania i inne tego typu rozwiązania chronią centrum przed włamaniem przez sieć (więcej na temat bezpieczeństwa w chmurze w rozdziale 4.).

Ponadto centra danych muszą być przygotowane na najgorsze oraz posiadać strategie wznowienia i utrzymania struktury po awarii, włamaniu lub innym zdarzeniu, by uniknąć utraty danych i zminimalizować okres nieaktywności.

Centra danych: skalowanie na potrzeby chmury

Tradycyjne duże centra danych przeznaczone do obsługi jednej dużej korporacji kosztują między 100 a 200 milionami dolarów[1]. Dla porównania: całkowity koszt budowy największych megacentrów danych, świadczących usługi przetwarzania w chmurze, wynosi ponad 500 milionów dolarów[2,3]. Co powoduje taki przyrost kosztów? Co mają największe centra przeznaczone dla chmury, czego nie mają tradycyjne centra dedykowane jednej firmie?

Najwięksi operatorzy, tacy jak Google, Amazon czy Microsoft, lokują swoje centra w niedużej odległości od rejonów, których mieszkańcy intensywnie z nich korzystają, co zmniejsza opóźnienia i ułatwia przełączanie się pomiędzy maszynami w przypadku awarii. Szukają także miejsc, w których energia elektryczna jest tania. W Stanach Zjednoczonych takim obszarem jest Północny Zachód — elektrownie wodne produkują najtańszą energię w kraju, a koszty chłodzenia są bliskie zeru. Sama konsumpcja energii przez duże centrum danych może kosztować więcej niż 30 milionów dolarów rocznie. Obecnie zużycie energii przez takie centra stanowi 1,2 procenta całkowitego zużycia energii w kraju i wciąż rośnie.

---

[1] *http://perspectives.mvdirona.com/2008/11/28/CostOfPowerInLargeScaleDataCenters.aspx*.

[2] *http://www.datacenterknowledge.com/archives/2007/11/05/microsoft-plans-500m-illinois-data-center*.

[3] *http://www.theregister.co.uk/2009/09/25/microsoft_chillerless_data_center*.

Światowe serwery emitują więcej dwutlenku węgla niż cała Holandia

Firma doradcza McKinsey & Co. donosi, że 44 miliony działających na świecie serwerów są odpowiedzialne za 0,5 procenta światowej konsumpcji energii oraz 0,2 procenta emisji dwutlenku węgla, co odpowiada 80 megatonom rocznie. Zbliżone ilości dwutlenku węgla emitują takie kraje, jak Argentyna czy Holandia.

Plusem dla operatorów jest to, że przy takim zapotrzebowaniu na energię mogą negocjować potężne zniżki.

Dodatkowo megacentra mogą negocjować ogromne zniżki na zakup sprzętu, wykraczające daleko poza zniżki oferowane największym nawet firmom budującym prywatne, dedykowane serwerownie. Przykładowo w 2008 roku firma Amazon zapłaciła około 90 milionów dolarów za 50 tysięcy serwerów firmy Rackable/SGI[4]. W normalnym handlu kosztowałyby one 215 milionów.

Serwery stanowią największą część kosztów ponoszonych przez centra danych. Dlatego Google i inni próbują ciąć koszty, samodzielnie konstruując serwery z gotowych komponentów. Google polega na tanich komputerach ze zwykłymi procesorami wielordzeniowymi. Jedno centrum danych Google posiada dziesiątki tysięcy takich niedrogich procesorów i dysków pospinanych rzepami, co umożliwia łatwą wymianę komponentów.

By ograniczyć apetyt maszyn na energię, firma Google wprowadziła mechanizmy optymalizujące dopływ prądu, wentylatory o zmiennej prędkości oraz płyty główne wypatroszone ze wszystkich zbędnych elementów, w tym kart graficznych. Eksperymentowano także z **dynamicznym skalowaniem napięcia i częstotliwości**. Napięcie lub częstotliwość procesora są obniżane w pewnych okresach (np. gdy wynik obliczeń nie jest wymagany natychmiast). Serwer działa wolniej, co zmniejsza konsumpcję energii. Inżynierowie Google twierdzą, że w niektórych testach oszczędność energii wyniosła 20 procent.

W roku 2006 firma Google wybudowała w miejscowości Dalles w Oregonie dwa centra danych, z których każde zajmuje teren wielkości boiska futbolowego, ma cztery piętra i czteropiętrową chłodnię (rysunek 2.1). Strategiczne znaczenie dla centrum danych ma tama w Dalles, zapewniająca dopływ energii oraz chłodzenie. (Niektóre nowe centra danych korzystają z chłodni kominowych, które odparowują wodę wykorzystaną do chłodzenia, co zużywa o wiele mniej energii niż tradycyjne chłodziarki).

Centrum danych w Dalles jest doskonale połączone istniejącymi już wcześniej światłowodami z różnymi lokalizacjami w USA, Azji i Europie. Kable te są spuścizną po bańce internetowej.

W 2007 roku firma Google stworzyła co najmniej cztery nowe centra danych warte średnio 600 milionów dolarów. Weszły one w skład Googleplexu: wielkiej światowej sieci komputerowej, szacowanej na 450 tysięcy serwerów w dwudziestu pięciu lokalizacjach. Firma Amazon również zdecydowała się umieścić swoje największe centrum danych w Dalles, kawałek dalej wzdłuż rzeki.

[4] *http://www.datacenterknowledge.com/archives/2009/06/23/amazon-adds-cloud-data-center-in-virginia.*

Rysunek 2.1. Zdjęcie przedstawia ściśle tajne centrum danych Google w miejscowości Dalles w Oregonie, zbudowane w pobliżu tamy w celu uzyskania dostępu do taniej energii. Zwróć uwagę na duże chłodnie kominowe na końcach znajdujących się po lewej stronie zdjęcia budynków wielkości boiska do futbolu. Kominy chłodzą budynki przez parowanie, bez używania kosztownych energetycznie chłodziarek. Źródło: Melanie Conner, „New York Times"

Yahoo! i Microsoft wybrały miejscowość Quincy w Waszyngtonie. Budynek Microsoftu ma powierzchnię porównywalną niemal z obszarem dziesięciu boisk futbolowych. Przedstawiciele firmy nabrali wody w usta, jeśli chodzi o liczbę serwerów, jednak wiadomo, że wykorzystuje się tam prawie pięć kilometrów rur chłodzących, prawie tysiąc kilometrów przewodów elektrycznych, niemal 10 tysięcy metrów kwadratowych płyt kartonowo-gipsowych i 1,6 tony baterii zapewniających prąd w razie awarii. Centrum zużywa 48 megawatów energii, co wystarczyłoby dla 40 tysięcy domów.

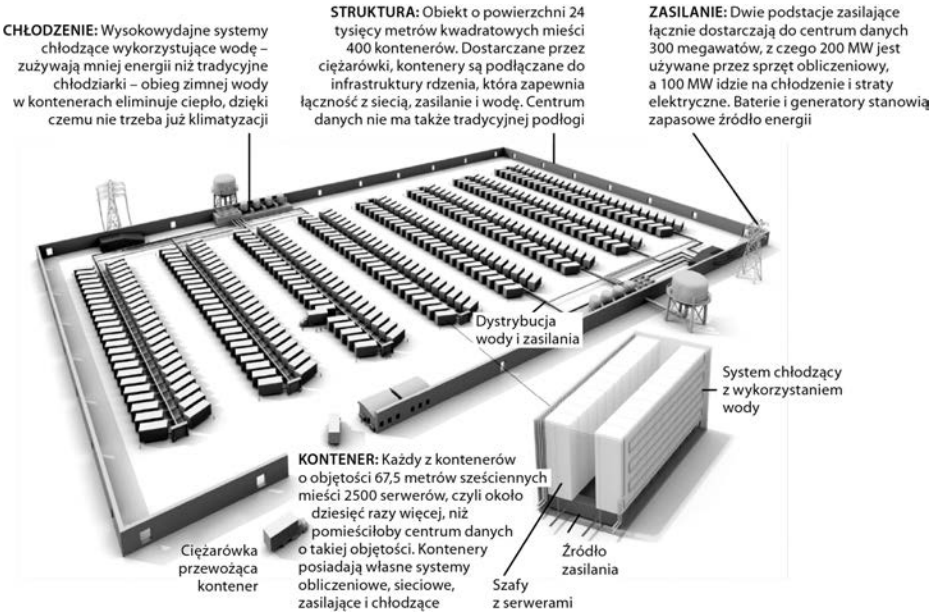
Chmurowe centra danych: większa efektywność i elastyczność dzięki modularyzacji

Już teraz, dzięki zakupom hurtowym, serwerom budowanym na zamówienie i starannie wybranym lokalizacjom, właściciele największych centrów danych ponoszą o wiele mniejsze koszty w przeliczeniu na operacje procesora niż zwykłe firmy. Robią, co mogą, by jeszcze zwiększyć tę różnicę. Centra danych w chmurze stają się coraz bardziej efektywne dzięki modularyzacji. Modułowe, skalowalne, efektywne centra danych są natychmiast dostępne za niewielką cenę z każdego miejsca na świecie.

Rysunek 2.2 przedstawia wizualizację modularnego centrum danych (fotografie takich obiektów są pilnie strzeżone). Firmowe centra danych zostały w tyle za wydajnymi megacentrami, a ich sytuacja będzie się jeszcze pogarszała.

Modularyzacja ma na celu ustandaryzowanie centrów danych i odstąpienie od własnych projektów, co ma ułatwić produkcję sprzętu dla takich centrów. Jedną z najbardziej uderzających postulowanych cech jest to, że takie centra znajdują się pod gołym niebem.

Podobnie jak Google Microsoft stara się ciąć koszty energii, ulega także naciskom ekologów, by ograniczać emisję i poprawić wydajność. Docelowy współczynnik wydajności energetycznej PUE (ang. *Power Usage Effectiveness*) to co najwyżej 1,125 we wszystkich centrach danych Microsoftu do 2012 roku.



Rysunek 2.2. Rozszerzalne, modularne centrum danych dla chmury. Warto zwrócić uwagę na brak zadaszenia. Kontenery z serwerami, zasilaniem, sprzętem chłodzącym i monitorującym mogą być dodawane i usuwane w miarę potrzeb. Źródło: magazyn „IEEE Spectrum"

2.1.2. Efektywne wykorzystanie serwerów w chmurze dzięki wirtualizacji

Pozostańmy przy analogii samochodowej — wirtualizacja to zawieszenie. Zapewnia ona pożądane intensywne wykorzystanie serwerów. Łagodzi różnice pomiędzy aplikacjami, które prawie nie potrzebują mocy obliczeniowej (mogą one dzielić procesor z innymi aplikacjami), a tymi, które domagają się każdego istniejącego cyklu procesora. Z wszystkich rewolucyjnych technologii wykorzystywanych w chmurze to właśnie wirtualizacja i jej udane wdrożenie zadecydowały o przyjęciu się nowego trendu. Bez wirtualizacji i umożliwianego przez nią zużycia procesora wynoszącego ponad 60 procent chmura nie byłaby opłacalna.

Współczynnik wydajności energetycznej (PUE)

Współczynnik wydajności energetycznej (PUE, ang. *Power Usage Effectiveness*) określa wydajność centrum danych. PUE wyznacza się, dzieląc ilość energii dostarczanej przez centrum danych przez ilość niezbędną do działania infrastruktury obliczeniowej wewnątrz. Wydajność poprawia się, gdy PUE zmniejsza się w kierunku wartości 1.

Według organizacji Uptime Institute średnia wartość PUE typowego centrum danych to 2,5. Oznacza to, że z każdego 2,5 wata na liczniku tylko wat jest wykorzystywany do obliczeń. Z szacunków Uptime wynika, iż większość centrów mogłaby osiągnąć wynik 1,6 PUE, stosując najlepszy sprzęt i dobre praktyki. Google i Microsoft zbliżają się do wartości 1,125 — to wynik nieosiągalny dla firmowych, a nawet współdzielonych centrów danych.

WIRTUALIZACJA

W tej książce koncentrujemy się głównie na wirtualizacji *platformy*. Wirtualizacja platformy to technika abstrakcji zasobów komputera oddzielająca system operacyjny od sprzętu. System operacyjny nie odwołuje się bezpośrednio do sprzętu. Zamiast tego odwołuje się on do nowej warstwy programistycznej, nazywanej *monitorem maszyny wirtualnej*, która ma dostęp do sprzętu i przedstawia systemowi wirtualny zestaw zasobów sprzętowych. Oznacza to, że wiele obrazów maszyn wirtualnych lub instancji może działać na jednym fizycznym serwerze, a nowe instancje mogą być tworzone i uruchamiane na żądanie. W ten sposób tworzona jest podstawa elastycznych zasobów obliczeniowych.

Wspomnieliśmy wcześniej, że wirtualizacja wcale nie jest nowym pomysłem. Komputery typu *mainframe* produkowane przez IBM stosowały wirtualizację czasową już w latach sześćdziesiątych, umożliwiając wielu osobom korzystanie z jednej maszyny w różnym czasie bez żadnej interakcji i wchodzenia sobie w drogę. Wcześniej na użytkowników nakładany był obowiązek wykonania całości planowanej pracy w jednym, wyznaczonym okienku czasowym. Pojęcie pamięci wirtualnej, wprowadzone około 1962 roku, choć uznawane wówczas za radykalne, raz na zawsze uwolniło programistów od ciągłego zamartwiania się możliwością przekroczenia wielkości pamięci fizycznej. Dzisiaj wirtualizacja serwerów ma równie znaczący wpływ na wdrażanie i skalowanie aplikacji, będąc zarazem jednym z największych motorów rozwoju chmury. Jak do tego doszło?

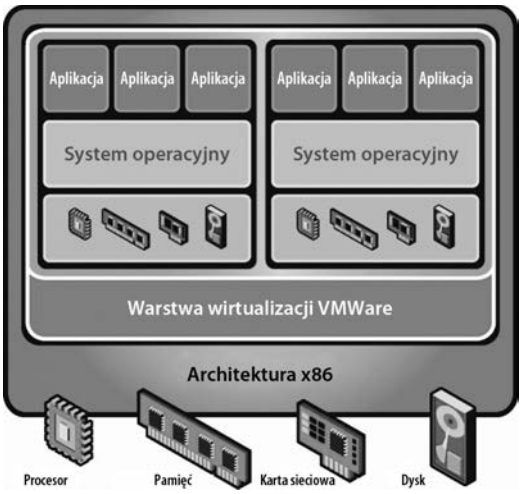
Przeciętny serwer w firmowym centrum danych jest wykorzystywany w około 6 procentach[5]. Nawet w chwilach maksymalnego obciążenia wykorzystanie nie przekracza 20 procent. W najlepiej zarządzanych centrach danych serwery wykorzystują średnio około 15 procent swoich możliwości. Jednak gdy takie centra zdecydują się na pełną wirtualizację, wykorzystanie mocy procesora wzrasta do 65 procent lub więcej. Z tego powodu w ciągu ostatnich paru lat w większości centrów danych wdrożono setki lub tysiące serwerów wirtualnych w miejsce poprzedniego modelu, w którym jeden serwer odpowiadał jednej fizycznej jednostce. Przyjrzyjmy się teraz temu, co sprawia, że wykorzystanie zmienia się aż tak radykalnie.

Jak to działa?

Wirtualizacja serwera przekształca (*wirtualizuje*) zasoby sprzętowe komputera — w tym czas procesora, pamięć RAM, dysk twardy i kontroler sieciowy — by stworzyć w pełni funkcjonalną maszynę wirtualną, na której system operacyjny i inne aplikacje mogą działać tak, jakby działały na fizycznym komputerze. Efekt ten osiąga się, dodając cienką warstwę oprogramowania działającą bezpośrednio nad sprzętem. Warstwa ta zawiera monitor maszyny wirtualnej (VMM, ang. *Virtual Machine Monitor*), który przydziela zasoby sprzętowe w dynamiczny i przejrzysty sposób. Dzięki temu na jednej fizycznej maszynie może działać wiele systemów operacyjnych, które współbieżnie korzystają z zasobów sprzętowych. Enkapsulacja całej maszyny, w tym procesora, pamięci i urządzeń sieciowych,

[5] McKinsey & Company, *2008 Data Center Efficiency* — raport na temat wydajności centrów danych.

sprawia, że maszyna wirtualna staje się w pełni kompatybilna ze standardowymi systemami operacyjnymi, aplikacjami i sterownikami urządzeń. Architektura maszyny wirtualnej VMware dla procesorów x86 została przedstawiona na rysunku 2.3.



Rysunek 2.3. Architektura maszyny wirtualnej na przykładzie VMware. Warstwa wirtualizacji odwołuje się bezpośrednio do komponentów sprzętowych, w tym do procesora. Warstwa ta prezentuje każdemu z goszczących na maszynie systemów operacyjnych jego własny zestaw zasobów sprzętowych. Taki system zachowuje się dokładnie tak samo jak system faktycznie mający dostęp do sprzętu. Dzięki wirtualizacji wiele systemów i działających w nich aplikacji może korzystać z jednej fizycznej maszyny, osiągając lepszą wydajność. Źródło: VMware

Zastosowanie wirtualizacji w chmurze

Wirtualizacja szybko znalazła uznanie u architektów systemów i kierowników działów IT z bardzo prostego powodu — pozwala zaoszczędzić pieniądze. Radykalnie zwiększył się stopień wykorzystania zasobów sprzętowych. Bez większego wysiłku można było przejść od 5 czy 6 do 20 procent. Przy rozsądnym planowaniu możliwe stało się osiągnięcie wyniku 65 procent wykorzystania sprzętu.

Poza lepszym współczynnikiem wykorzystania sprzętu i związanymi z nim oszczędnościami wirtualizacja w firmowych centrach danych w dużej mierze przygotowała grunt dla przetwarzania w chmurze. Użytkownicy zostali oddzieleni od implementacji, poprawiła się szybkość, elastyczność i „zwinność", a do tego zmienił się nienaruszalny dotąd model wyceny i licencjonowania oprogramowania. Tabela 2.1 zawiera objaśnienia poszczególnych elementów.

Tabela 2.1 przedstawia najważniejsze zyski z wprowadzenia chmury. Chmura zdobywa coraz większe uznanie i coraz więcej firm jest gotowych, by przejść na ten model obliczeniowy. Wiele firm już wcześniej korzystało z wirtualizacji, więc przejście na chmurę jest dla nich bezbolesne.

Rozpatrzymy scenariusz z wykorzystaniem tysięcy serwerów fizycznych. Każdy z nich został zwirtualizowany, w związku z czym może na nim działać dowolnie wiele systemów operacyjnych. Konfiguracja i wdrażanie trwają parę

Tabela 2.1. Wpływ wirtualizacji na firmowe centra danych

Zysk	Objaśnienie
Oddzielenie użytkowników od implementacji	Pojęcie serwera wirtualnego sprawia, że użytkownik nie musi, a nawet nie może martwić się o fizyczny serwer lub jego lokalizację. Zamiast tego koncentruje się na umowach i definicjach usług oraz wdrażanych aplikacjach.
Pozyskanie nowego serwera trwa parę minut, a nie kilka miesięcy	Zamówienie, instalacja, konfiguracja i uruchomienie fizycznego serwera w dużych firmach zajmuje 60, 90, a czasem nawet 120 dni. W modelu serwerów wirtualnych czas pomiędzy żądaniem a wdrożeniem gotowej aplikacji jest liczony w minutach lub godzinach, w zależności od stopnia automatyzacji.
Nowy model naliczania opłat i licencjonowania	Centrum danych nie może już naliczać opłat za cały serwer ani za wszystkie serwery, na których działa aplikacja. Zamiast tego nalicza opłaty za faktyczne zużycie — to rewolucja w informatyce.

minut, a opłaty są pobierane za każdą godzinę wykorzystania procesora. Zasoby megacentrów danych stały się dostępne dla zwykłych użytkowników dzięki połączeniu następujących elementów: wirtualizacji, automatycznego dodawania i usuwania sprzętu oraz nowych zasad naliczania opłat. Jako odpowiednik samochodowych resorów wirtualizacja sprawia, że aplikacja może gwałtownie przyspieszyć, nie zabijając przy tym wszystkich siedzących w pojeździe po wjechaniu na pierwszy wybój.

Jednak potężny silnik (centrum danych) i dobre zawieszenie (wirtualizacja) to nie wszystko. Potrzebna jest jeszcze deska rozdzielcza — zestaw urządzeń pozwalających na ruszanie, zatrzymywanie oraz sterowanie autem. Niezbędny jest interfejs dający kontrolę nad chmurą.

2.1.3. Sterowanie zdalnymi serwerami za pośrednictwem API chmury

API (interfejs programowania aplikacji) jest dla chmury tym, czym deska rozdzielcza dla samochodu. Pod maską może czaić się prawdziwy potwór, jednak bez kontrolek i pokręteł nie dowiesz się, co właściwie dzieje się z pojazdem ani jak nim sterować. Potrzebujesz przynajmniej kierownicy, gazu i hamulca. Pamiętaj także o tym, że nie można jeździć szybko bez dobrych hamulców.

Do chmury trzeba się jakoś dostać. Chmury najwyższego poziomu — te oferujące aplikacje SaaS (oprogramowanie jako usługa) — z reguły mają interfejsy oparte na przeglądarce. Chmury niższego poziomu, IaaS (infrastruktura jako usługa), też muszą mieć dostęp do aplikacji. Chmura każdego typu musi oferować API, za pośrednictwem którego da się dodawać zasoby, zarządzać nimi, konfigurować je i zwalniać, gdy przestają być potrzebne.

API jest niezbędne do korzystania z usług dostawcy chmury. W ten sposób sprzedający prezentuje swoją ofertę, którą kupujący może ocenić według swoich potrzeb. Przykładowo API chmury EC2 firmy Amazon jest oparte na protokole SOAP i HTTP, za pomocą którego wysyła się właściwe tej firmie żądania tworzenia, składowania i dodawania obrazów maszyn Amazon (AMI) oraz zarzą-

dzania nimi. Z kolei API oferowanej przez firmę Sun[6] chmury Kenai całkowicie opiera się na architekturze REST. To za pośrednictwem API tworzy się zasoby (moc obliczeniowa, składowanie, komponenty sieciowe) i zarządza nimi.

> **Architektura REST i zgodne z nią API** REST (ang. *Representational State Transfer*, transfer stanów reprezentacyjnych) to styl architektoniczny dla rozproszononych systemów, takich jak World Wide Web. Opiera się on na protokole HTTP. Sieć WWW jest największą znaną implementacją zgodną z architekturą REST — mówi się nawet, że REST to *specyfikacja post hoc* tych właściwości sieci, które są odpowiedzialne za jej sukces. Architektura REST zakłada istnienie klientów i serwerów. Klienci wysyłają żądania do serwerów, a procesy serwerów przetwarzają te żądania i zwracają odpowiednie odpowiedzi. Żądania i odpowiedzi zawierają *reprezentacje zasobów*. Zasób to dowolny spójny i sensowny byt, który jest adresowalny. Reprezentacja zasobu to z reguły dokument opisujący aktualny lub zamierzony stan zasobu. Aplikacje i interfejsy zgodne z zasadami i ograniczeniami architektury REST określa się mianem *RESTful*.

Jako że aplikacja działająca w chmurze będzie głównym żywicielem Twojej firmy, musisz postarać się o jej ochronę przed nieuprawnionymi osobami. Gdyby działała ona w prywatnym, należącym do Twojej firmy centrum danych, chronionym przez wiele fizycznych i logicznych warstw zabezpieczających, miałbyś pewność, że nie dostanie się do niej nikt niepowołany. W chmurze wszystko jest z definicji dostępne przez internet. Amazon i inni rozwiązują ten problem, wydając klucze publiczne X.509 i żądając ich podania przy każdym wywołaniu API. W ten sposób serwer zyskuje pewność, że byt wywołujący API ma prawo dostępu do infrastruktury.

Aby zrozumieć API chmury — a nie istnieje jeszcze uznany standard — najlepiej przyjrzeć się API chmury Amazon, gdyż firma ta jest liderem i narzuca rozwiązania innym. Tabela 2.2 przedstawia najważniejsze definicje i operacje leżące u podstaw API chmury Amazon.

To oczywiście wierzchołek góry lodowej, jeśli chodzi o terminy i wywołania w API chmury Amazon. Dokumentacja jest dostępna pod adresem: *http://aws.amazon.com/documentation/*. API można podzielić na następujące obszary:

- adresowanie chmur,
- bezpieczeństwo sieciowe,
- regiony i strefy dostępności,
- usługa Amazon Elastic Block Store (EBS),
- automatyczne skalowanie, równoważenie obciążenia i Amazon Cloud Watch,
- publiczne zbiory danych,
- chmura prywatna Amazon.

[6] Obecnie Oracle — *przyp. tłum.*

Tabela 2.2. Podstawowe pojęcia i operacje API chmury obliczeniowej Amazon EC2

Pojęcie	Opis
AMI	Obraz maszyny Amazon (AMI, ang. *Amazon Machine Image*) to zaszyfrowany i podpisany obraz maszyny, który można uruchomić w środowisku serwera wirtualnego. Przykładowo na obraz może składać się następujący zestaw: Linux, Apache, MySQL, PHP oraz aplikacja właściciela AMI.
	AMI mogą być publiczne (zapewniane przez Amazon), prywatne (zaprojektowane przez twórcę aplikacji), zakupione (od zewnętrznego producenta) lub współdzielone (stworzone za darmo przez pewną społeczność).
	AMI można składować w serwisie Amazon Simple Storage Service (S3).
Instancja	System działający w wyniku uruchomienia AMI to właśnie *instancja*. Gdy instancja kończy działanie, jej dane zostają utracone. Instancja pełni taką samą funkcję jak tradycyjny samodzielny komputer.
Standardowy przepływ	1. Dopasuj AMI do swoich potrzeb.
	2. Przygotuj paczkę AMI i pozyskaj identyfikator.
	3. Uruchom jedną lub więcej instancji AMI.
	4. Zarządzaj działającymi instancjami i korzystaj z nich.
Podłączenie	W przeglądarce wpisz adres: *http://<nazwa-hosta>*, w którym *<nazwa-hosta>* to publiczna nazwa Twojej instancji.
	Jeśli chcesz podłączyć się do dopiero co uruchomionego publicznego AMI, który nie został jeszcze zmodyfikowany, skorzystaj z polecenia `ec2-get-console-output`.
	W obu przypadkach masz możliwość zalogowania się jako administrator i sprawowania pełnej kontroli nad instancją — odpowiada to podejściu do komputera w centrum danych i zalogowaniu się do niego.

Do różnych aspektów API chmury będziemy jeszcze wracać na dalszych kartach tej książki. Na razie porzucimy ten temat i zajmiemy się kolejną ważną warstwą: składowaniem danych w chmurze.

2.1.4. Przechowywanie trwałych danych w chmurze

Bagażnik samochodu służy do przechowywania bagażu podróżnego i całej masy innych rzeczy. Analogicznie, chmura zapewnia miejsce, w którym możesz przechować obrazy maszyn, aplikacje i wykorzystywane przez nie dane. Przechowywanie danych w chmurze zyskuje popularność z tych samych powodów, co przetwarzanie. Na żądanie otrzymujesz wirtualny magazyn danych w sieci, wymagasz także usługi określonej jakości. Inaczej niż w firmowym centrum danych nie musisz kupować dysków — czasami nie musisz nawet zamawiać miejsca przed wysłaniem danych. Z reguły płacisz za ilość przesyłanych danych oraz, co jakiś czas, za zajmowane przez nie miejsce.

Z przechowywania w chmurze można korzystać na różne sposoby. Przykładowo możesz tworzyć kopie bezpieczeństwa danych lokalnych, na przykład tych z prywatnego laptopa. Możesz zsynchronizować z chmurą dysk wirtualny i mieć dostęp do danych z różnych maszyn. Możesz również archiwizować dane zgodnie z pewną polityką, na przykład w celu nadzorczym.

Standard przechowywania danych w chmurze

Stowarzyszenie Storage Networking Industry Association powołało specjalną grupę roboczą, która miała zająć się wypracowaniem standardu składowania danych w chmurze. Nowy interfejs CDMI (ang. *Cloud Data Management Interface*, interfejs zarządzania danymi w chmurze) umożliwia uniwersalne, niezależne od środowiska zarządzanie takimi danymi. W CDMI magazyn danych jest widziany przez interfejsy jako abstrakcyjny kontener. Dodatkowo kontener ułatwia grupowanie danych i tworzenie agregatów.

Przechowywanie w chmurze sprawdzi się w przypadku aplikacji, które dostarczają dane bezpośrednio do klientów w sieci. Aplikacja przekierowuje klienta do odpowiedniej lokalizacji, gdzie pobiera on dane, na przykład pliki audio czy wideo. Wymagania związane z przesyłaniem strumieni danych poprzez sieć będą się skalowały niezależnie i nie zaburzą działania aplikacji.

Wykorzystuje się tu interfejs HTTP. Możesz pobrać plik za pomocą przeglądarki bez pisania dodatkowego kodu — automatycznie uruchomi się odpowiednia aplikacja. Tylko jak umieścić plik w chmurze i jak upewnić się, że magazyn jest odpowiedniego typu i zapewnia wymaganą przez Ciebie jakość? Znów mamy do czynienia z szeroką ofertą. W tym wypadku nie dziwi, iż wielu dostawców korzysta z interfejsów zgodnych z zasadami architektury REST. Najczęściej mamy do czynienia z interfejsem pozwalającym na tworzenie, odczytywanie, aktualizację i usuwanie poszczególnych obiektów danych za pośrednictwem metod HTTP.

Kolejny raz potraktujemy API chmury Amazon jako przykład do analizy. Tabela 2.3 przedstawia najważniejsze elementy chmury S3.

Tabela 2.3. Podstawowe pojęcia i operacje chmury danych Amazon S3

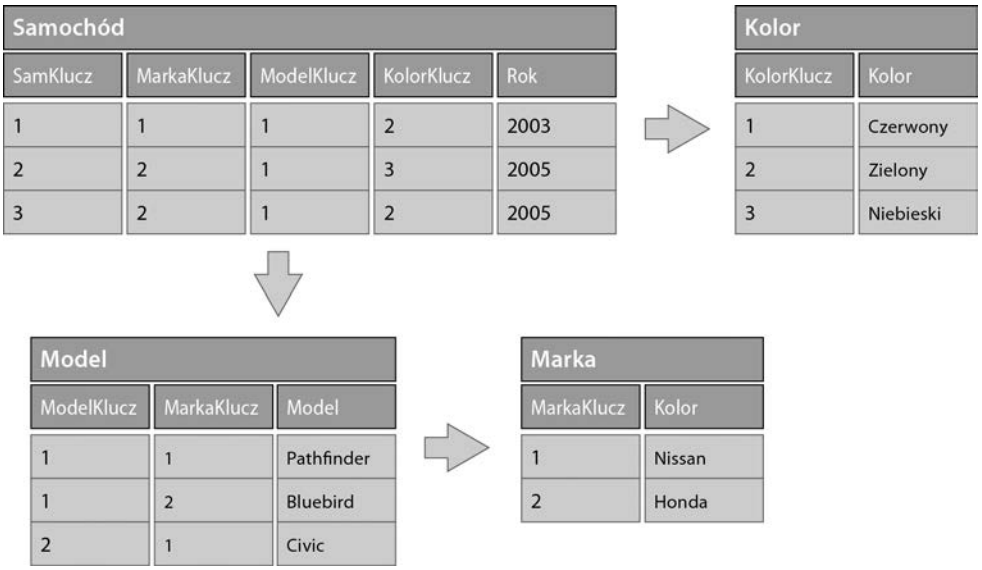
Pojęcie	Opis
Obiekt	Podstawowa jednostka składowana w chmurze danych S3. Obiekt może mieć rozmiar od 1 bajta do 5 gigabajtów. Każdy obiekt ma swoje metadane w postaci par klucz-wartość, opisujących obiekt.
Kubełek	Podstawowy kontener do przechowywania danych w chmurze S3. Obiekty wgrywa się do kubełków. Kubełek (ang. *bucket*) to unikalna przestrzeń nazw, umożliwiająca zarządzanie całą zawartością. Nazwy kubełków są globalne. Jeden programista może korzystać jednocześnie z maksymalnie stu kubełków.
Klucz	Klucz to unikalny identyfikator obiektu wewnątrz kubełka. Nazwa kubełka połączona z kluczem jednoznacznie identyfikuje obiekt wewnątrz całej chmury S3.
Wykorzystanie	1. Stwórz kubełek, w którym będziesz przechowywać dane. 2. Załaduj (zapisz) dane (obiekty) do kubełka. 3. Pobierz (odczytaj) dane z kubełka. 4. Skasuj część danych przechowywanych w kubełku. 5. Wypisz obiekty znajdujące się w kubełku.

W wielu przypadkach „gruboziarnista" i nieuporządkowana natura usług przechowywania w chmurach takich jak S3 okazuje się niewystarczająca — potrzebna jest alternatywna metoda przechowywania o ustalonej strukturze. Spójrzmy zatem, jak działają (i nie działają) chmurowe bazy danych.

2.1.5. Przechowywanie danych aplikacji w chmurowej bazie danych

Samochodowy system nawigacji na bieżąco informuje Cię o Twoim aktualnym położeniu i odległości od celu podróży. Prowadzi Cię drogą, którą musisz przebyć. Tego typu dane, chociaż niezbędne w czasie podróży, po jej zakończeniu przestają być przydatne. System nawigacyjny jest dla samochodu tym, czym chmurowa baza danych dla aplikacji działającej w chmurze: dane transakcyjne są tworzone i wykorzystywane tylko w czasie działania aplikacji. Gdy mówimy o transakcyjnych danych przechowywanych w bazie, z reguły myślimy o relacyjnych bazach danych.

Czym jest RDBMS i dlaczego często słychać, że nie działa on w chmurze? RDBMS (ang. *Relational Database Management System*) to system zarządzania relacyjną bazą danych, w której przechowuje się dane w postaci tabel. Relacje pomiędzy danymi także reprezentowane są przez tabele. Rysunek 2.4 przedstawia prosty przykład.



Rysunek 2.4. Prosty przykład działania bazy relacyjnej. Cztery tabele odwzorowują zależności (relacje) pomiędzy danymi. Osobna tabela służy do prezentowania marek, inna pokazuje kolory — dzięki temu nie zachodzi konieczność oddzielnego reprezentowania na przykład czerwonego Nissana lub niebieskiego Forda. Jednak by w pełni zrozumieć, czym jest samochód o identyfikatorze SamKlucz 1, musisz połączyć tabele Samochód, Kolor, Model i Marka

RDBMS System zarządzania bazą danych (ang. *Database Management System*) opartą na modelu relacyjnym. *Relacyjne* bazy danych to bazy należące do dość rozległej klasy systemów bazodanowych, w których dane są prezentowane użytkownikowi w postaci relacji (zestaw tabel złożonych z wierszy i kolumn spełnia ten warunek), a on może modyfikować te dane za pomocą operatorów relacyjnych. Wszystkie współczesne relacyjne bazy danych wykorzystują język zapytań SQL, przez co bazy RDBMS często nazywa się *bazami SQL*.

Wyzwaniem dla systemów RDBMS działających w chmurze jest skalowalność. Aplikacje o znanej liczbie użytkowników i obciążeniu nie odnotują kłopotów. Większość dostawców chmur oferuje usługi RDBMS dla takich przypadków. Jednak gdy aplikacje działają w środowisku o dużym obciążeniu (jak w przypadku usług sieciowych *web services*), ich wymagania dotyczące skalowalności mogą zmieniać się (a zwłaszcza rosnąć) bardzo szybko. Zmieniające się wymagania są problemem, gdy baza danych jest zainstalowana na pojedynczym serwerze — jeśli ilość danych codziennie się potraja, jak szybko będziesz nadążać z rozbudową sprzętu? Zbyt duża ilość danych może przekroczyć możliwości bazy relacyjnej — stanie się ona wąskim gardłem, uniemożliwiającym skalowanie aplikacji. Istniejącym rozwiązaniom tego problemu przyjrzymy się dokładniej w rozdziale 5.

Powiedzieliśmy już, że jedną z największych zalet chmury jest możliwość szybkiego (i automatycznego, co dopiero zaprezentujemy) dodawania nowych serwerów do aplikacji, gdy wzrasta obciążenie. Jednak skalowanie systemów RDBMS jest o wiele trudniejsze. Konieczne jest albo zreplikowanie danych na wszystkich nowych serwerach, albo podzielenie ich pomiędzy maszyny wirtualne. W obu przypadkach dodanie nowej maszyny wymaga kopiowania lub przenoszenia danych na nowy serwer. Jest to czasochłonny i drogi proces, dlatego bazy danych nie mogą być dynamicznie dostarczane na żądanie.

Duże wyzwanie podczas podziału lub replikacji RDBMS stanowi zachowanie zasady *integralności referencyjnej*. Baza spełnia tę zasadę, jeśli każda wartość atrybutu (kolumny) relacji (tabeli) istnieje jako wartość innego atrybutu w innej (lub tej samej) relacji (tabeli). Mniej formalnie: każde pole tabeli zadeklarowane jako klucz obcy może zawierać tylko wartości z klucza głównego lub potencjalnego tabeli rodzica. W praktyce oznacza to, że usunięcie rekordu zawierającego wartość, do której odwołuje się jakiś klucz obcy z innej tabeli jest złamaniem zasady integralności referencyjnej. Podczas dzielenia lub replikacji bazy danych prawie niemożliwe jest zachowanie integralności we wszystkich bazach. Bardzo przydatna cecha systemu RDBMS, jaką jest konstruowanie relacji z małych, poindeksowanych tabel, do których odwołują się wartości rekordów, okazuje się nierealizowalna, gdy bazy mają się skalować podczas dużych obciążeń, mimo że możliwe jest skalowanie wszystkich innych elementów aplikacji w chmurze.

Ruch NoSQL

Od 1998 roku notuje się powolne, ale nabierające szybkości odchodzenie od baz SQL. Zamiast nich proponowana jest klasa nierelacyjnych magazynów danych, łamiących podstawowe zasady SQL, ale umożliwiających osiągnięcie o wiele większej skali. Oczywiście jest to pożądana cecha w przypadku *niektórych* aplikacji w chmurze. Nierelacyjne bazy danych często nie wymagają sztywnych schematów danych i raczej unikają operacji łączenia (ang. *join*). Mówi się o nich, że skalują się **horyzontalnie** (poziomo). Używane jest także określenie **ustrukturyzowane przechowywanie** (ang. *structured storage*).

Bazy danych NoSQL, najczęściej w postaci klucz-wartość, faktycznie skalują się lepiej. Z tego powodu coraz chętniej są używane w chmurze. Podstawową jednostką takiej bazy są artykuły (ang. *items*) — wszystkie dane na temat pewnego artykułu są przechowywane wraz z nim. Tabela może zawierać bardzo różne artykuły, na przykład marki, modele i kolory samochodów. Oznacza to, że dane są często duplikowane w różnych artykułach w tabeli (przykładowo więcej niż jeden artykuł może mieć ten sam kolor). Przykład został przedstawiony na rysunku 2.5. W systemie RDBMS za coś takiego twórca zostałby wyklęty. W bazie NoSQL takie rozwiązanie jest akceptowane, gdyż przestrzeń dyskowa jest stosunkowo tania. W takim modelu pojedynczy artykuł zawiera wszystkie związane z nim dane, co poprawia skalowalność poprzez eliminację konieczności łączenia tabel. Chcąc przegrupować atrybuty w relacyjnej bazie danych, musielibyśmy połączyć tabele. To podstawowa zasada skalowalności — jeśli konieczne jest złączenie osobnych tabel, replikacja danych jest trudna i blokuje skalowalność.

Architektura NoSQL

Ograniczenia relacyjnych baz danych powodują, że nie najlepiej sprawdzają się one podczas przetwarzania dużych objętości danych w bezprecedensowej współczesnej skali. Przykłady ogromnej skali to nazwy najpopularniejszych stron: zielone plakietki Digg zajmują 3 TB (terabajty), wyszukiwanie w skrzynce odbiorczej na Facebooku to 50 TB, dane portalu eBay zajmują razem 2 PB (petabajty).

Systemy NoSQL z reguły dają słabe gwarancje spójności, na przykład na poziomie pojedynczych artykułów danych. W większości przypadków możliwe jest wymuszenie pełnych gwarancji ACID (ang. *Atomicity*, *Consistency*, *Isolation*, *Durability* — atomowość, spójność, izolacja, trwałość) przez dodanie pośredniczącej warstwy oprogramowania.

Niektóre systemy NoSQL charakteryzują się architekturą rozproszoną — dane są przechowywane w redundantny (nadmiarowy) sposób na kilku serwerach, często z użyciem rozproszonej tablicy mieszającej. Dzięki temu system można łatwo skalować, dodając nowe serwery, a awaria pojedynczego serwera nie jest problemem.

Niektórzy zwolennicy NoSQL postulują stosowanie prostych interfejsów, takich jak tablice asocjacyjne i pary klucz-wartość. Część systemów, jak bazy oparte na XML-u, wspiera się standardem XQuery.

Jak widać, chmura jest na wczesnym etapie ewolucji i wiele decyzji dopiero zostanie podjęte.

Samochód	
Klucz	Atrybuty
1	Marka: Nissan Model: Pathfinder Kolor: Zielony Rok: 2003
2	Marka: Nissan Model: Pathfinder Kolor: Niebieski Rok: 2005 Biegi: Automatyczne

Rysunek 2.5. Dane z rysunku 2.4, tym razem przedstawione w bazie danych typu klucz-wartość. Ponieważ wszystkie dane dla artykułu (wiersza) są zawarte wewnątrz niego, skalowanie staje się trywialne. Podział (przeniesienie części wierszy w inne miejsce) lub replikacja (skopiowanie całości danych) to proste czynności, niezaburzające integralności referencyjnej

Gdy firma decyduje się na stworzenie publicznej chmury obliczeniowej (jak Amazon) albo na budowanie silnie równoległych, redundantnych, a zarazem ekonomicznych aplikacji sterowanych danymi (Google), bazy relacyjne pozostają bezradne. Obie te firmy potrzebowały sposobu zarządzania danymi, który byłby niemalże nieskończenie skalowalny, godny zaufania i efektywny, także finansowo. W rezultacie obie zaproponowały nierelacyjne systemy bazodanowe oparte na parach klucz-wartość. Amazon nazywa swoją chmurową bazę danych SimpleDB, Google ma BigTable. Obie bazy powstały długo przed tym, zanim firmy uruchomiły chmurę. Stworzono je w celu rozwiązania wewnętrznych problemów. Po uruchomieniu chmury te same struktury stały się częścią oferty przetwarzania w chmurze.

Rozwiązanie BigTable to dość prosty system zarządzania danymi zapewniający szybki dostęp do petabajtów danych, które mogą być redundantnie rozproszone na tysiącach maszyn. Fizycznie BigTable przypomina tablicę z indeksem w formie B-drzewa, którego gałęzie i liście są przechowywane na różnych maszynach. Podobnie jak w B-drzewie wierzchołki w miarę przyrostu danych są dzielone. Dzięki temu, ponieważ węzły są rozproszone, baza skaluje się na wiele maszyn. Elementy danych w BigTable są identyfikowane za pomocą klucza głównego, nazwy kolumny i — opcjonalnie — znacznika czasowego. Wyszukiwanie przy użyciu klucza głównego jest przewidywalne i bardzo szybkie. Baza BigTable jest wykorzystywana przez Google App Engine. W dalszej części tego rozdziału dokładniej opiszemy to środowisko PaaS.

Google pobiera comiesięczną opłatę w wysokości 180 dolarów za każdy wykorzystany terabajt przestrzeni w BigTable. Poniżej przedstawiamy kilka przykładów kodu komunikującego się z tą bazą (w języku Python).

Deklaracja klasy przechowującej dane:

```
class Patient(db.Modal);              //pacjent
firstName = db.UserProperty()         //imię
lastName = db.UserProperty()          //nazwisko
dateOfBirth = db.DateTimeProperty()  //data urodzenia
sex = db.UserProperty()               //płeć
```

Kod tworzący obiekt i zapisujący go w bazie:

```
patient = Patient()
patient.firstName = "Jan"
patient.lastName = "Kowalski"
dateOfBirth = "2008-01-01"
sex = "M"
patient.put()
```

Zapytanie o obiekty danej klasy:

```
patients = Patient.all()

for patient in patients:
  self.response.out.write('Osoba: %s %s. ', patient.firstName,
patient.lastName)
```

Zapytanie o stu najmłodszych mężczyzn:

```
allPatients = Patient.all()
allPatients.filter('sex=', 'M')
allPatients.order('dateOfBirth')
patients = allPatients.fetch(100)
```

Baza SimpleDB, stanowiącą kluczową część środowiska obliczeń w chmurze AWS (Amazon Web Services), działa na bardzo podobnej zasadzie (analogiczną funkcjonalność oferuje także Microsoft przy użyciu SQL Server Data Services [SSDS] w chmurze Azure). SimpleDB również opiera się na parach klucz-wartość. Podstawową jednostką organizacyjną jest **domena** (ang. *domain*). Domeny to zbiory artykułów opisanych za pomocą par klucz-wartość. Tabela 2.4 przedstawia skrócony spis wywołań SimpleDB API wraz z ich opisem funkcjonalnym.

Przekształcenie istniejącej aplikacji tak, by wykorzystywała jedną z chmurowych baz danych, jest albo trudne, albo całkowicie niewarte wysiłku. Lepiej jest w przypadku aplikacji, które już teraz korzystają z frameworków ORM (ang. *Object-Relational Mapping*, mapowanie obiektowo-relacyjne) — w ich wypadku baza w chmurze może z łatwością realizować podstawowe funkcjonalności zarządzania danymi bez negatywnego wpływu na skalowalność (która jest jedną z podstaw istnienia chmury). Jednak, co ilustruje tabela 2.5, nowe typy chmurowych baz danych nie są pozbawione poważnych wad, które należy dokładnie przeanalizować, jeśli rozważa się przejście na chmurę.

Tabela 2.4. Streszczenie API bazy SimpleDB firmy Amazon

Wywołanie API	Opis funkcjonalny
`CreateDomain`	Tworzy domenę do przechowywania zbioru danych.
`DeleteDomain`	Usuwa domenę.
`ListDomains`	Wypisuje wszystkie domeny.
`DomainMetadata`	Pobiera informacje na temat czasu utworzenia domeny, liczbę nazw artykułów i atrybutów oraz całkowity rozmiar w bajtach.
`PutAttribute(s)`	Dodaje lub aktualizuje artykuł lub jego atrybuty, pozwala także dodawać pary klucz-wartość do już istniejących artykułów. Artykuły są automatycznie indeksowane po dodaniu do bazy.
`BatchPutAttributes`	Masowy zapis danych. Wykonuje do dwudziestu pięciu operacji `PutAttribute` w jednym wywołaniu.
`DeleteAttribute(s)`	Usuwa artykuł, atrybut lub wartość atrybutu.
`GetAttribute(s)`	Pobiera artykuł i wszystkie lub tylko niektóre atrybuty i wartości.
`Select`	Umożliwia odpytanie zbioru danych za pomocą składni `Select x from` *`nazwa_domeny`* `where` *`warunki_zapytania`*. Wartości można porównywać za pomocą operatorów `=`, `!=`, `<`, `>`, `<=`, `>=`, `like`, `not like`, `between`, `is null`, `isn't null` oraz `every()`. Przykład: `select * from mydomain where every(keyword) = "Książka"` Kolejność wyników można określić za pomocą operatora `SORT`. Operator `Count` zlicza wyniki pasujące do zapytania.

Tabela 2.5. Minusy chmurowych baz danych

Zastosowanie bazy	Wyzwania w przypadku bazy w chmurze
Transakcyjność i integralność referencyjna	Odpowiedzialność za integralność transakcji i relacji pomiędzy tabelami spoczywa w dużej mierze na aplikacji korzystającej z chmurowej bazy danych, a nie na samej bazie.
Dostęp do złożonych danych	Bazy w chmurze sprawdzają się w przypadku transakcji opartych na pojedynczych wierszach: pobierz wiersz, zapisz wiersz itd. Jednak nietrywialne aplikacje muszą wykonywać złączenia i inne operacje, które nie są możliwe w bazach danych tego typu.
Analityka biznesowa	Dane są potrzebne nie tylko do działania aplikacji — są także podstawą procesu analizy biznesowej. Nikt z własnej woli nie cofnie się do czasów sprzed powstania baz relacyjnych, gdy dane aplikacji były zakopane głęboko w jej trzewiach i nie była możliwa ich analiza i ocena.

Chmurowe bazy danych mogłyby zastąpić bazy relacyjne w sporym segmencie nowej generacji aplikacji korzystających z chmury. Jednak szefowie firm raczej nie podejdą z entuzjazmem do architektury, która utrudnia lub uniemożliwia wykorzystanie danych w procesie analizy biznesowej i podejmowania decyzji — do tego potrzebna jest relacyjna baza danych. Rozwiązaniem byłaby architektura zapewniająca skalowalność i umożliwiająca wykorzystanie wszystkich zalet chmury, a jednocześnie dająca pełen dostęp do ustrukturyzowanych danych. W ciągu kilku kolejnych lat można się spodziewać dużych postępów i wielu innowacji.

Ostatnią z technologicznych podstaw chmury, której chcemy poświęcić więcej uwagi, jest elastyczność. W naszym przykładzie z samochodem elastyczność odpowiada skrzyni biegów.

2.1.6. Elastyczność: skalowanie aplikacji w miarę zwiększania się lub zmniejszania popytu

Automatyczna skrzynia biegów łagodnie dostosowuje prędkość kół pojazdu do prędkości silnika, gdy kierowca przyspiesza lub hamuje. Podobnie elastyczność w chmurze pozwala aplikacji na bezproblemowe radzenie sobie w chwilach różnego zapotrzebowania użytkowników na aplikację. Elastyczność polega na dodawaniu zasobów, gdy zwiększa się obciążenie, i usuwaniu ich, gdy nie są już potrzebne. Wiele dużych firm stanęło w obliczu porażki, kiedy nie były w stanie sprostać zainteresowaniu użytkowników.

Skalowalność w chmurze to zdolność platformy do obsłużenia zwiększonej liczby użytkowników korzystających z aplikacji. **Elastyczność** to zdolność skalowania w górę lub w dół bez przerywania normalnego działania aplikacji. Bez elastyczności przeniesienie aplikacji i działalności firmy w chmurę byłoby nieopłacalne.

Poniższy zestaw wywołań pokazuje, jak skonfigurować aplikację w chmurze EC2, by była automatycznie skalowana (czyli elastyczna) z wykorzystaniem co najmniej dwóch, a najwyżej dwudziestu instancji. Określamy, że aplikacja ma otrzymać dodatkową jedną instancję, gdy średnie wykorzystanie procesora przekroczy 80 procent. Jedna instancja ma zostać odjęta, gdy wynik ten spadnie poniżej 40 procent na dłużej niż dziesięć minut.

Elastyczność a śmierci gwiazd

W czerwcu 2009 roku tego samego dnia odeszły dwie gwiazdy. Najpierw umarła znana z *Aniołków Charliego* aktorka Farrah Fawcett — spowodowało to lekkie poruszenie w mediach. Trochę później, w godzinach popołudniowych, rozszalał się prawdziwy sieciowy sztorm, gdy w społecznej sieci pojawiły się doniesienia o śmierci Michaela Jacksona. Nieoczekiwanie okazało się, że serwis Twitter napotkał ogromne trudności ze skalowaniem, kiedy pojawiły się setki tysięcy wpisów o śmierci artysty. Twitter nie był jednak sam.

Na blogu TechCrunch podano, że: „Należąca do firmy AOL strona TMZ, która jako pierwsza opublikowała informację, co jakiś czas przestawała działać. Prawdopodobnie w wyniku tego popularny blog Pereza Hiltona również odnotował problemy, gdy ludzie rzucili się tam, by potwierdzić wiadomość. Następnie strona LA Times poinformowała, że Jackson nie umarł, tylko jest w śpiączce, więc wszyscy przenieśli się tam, ponownie doprowadzając do przeciążenia strony (w końcu reporterzy LA Timesa potwierdzili informację o śmierci)".

Znane są liczne przypadki wiadomości, oświadczeń prasowych, nawet materiałów, takich jak niechlubna reklama Victoria's Secret podczas finałów Super Bowl, które spowodowały, że użytkownicy przypuścili szturm na stronę, a ta nie udźwignęła obciążenia. Zbyt duży ruch w połączeniu z niewystarczającymi zasobami oznacza katastrofę. Gdy użytkownik wchodzi na stronę, a ona nie działa, jest duże prawdopodobieństwo, że odpuści sobie dalsze wizyty. Takie problemy wyraźnie odciskają się na zyskach firmy. Wniosek jest taki: skalowanie w miarę wzrostu obciążenia ma kluczowe znaczenie dla powodzenia przedsięwzięcia.

Wywołaj `CreateLoadBalancer` z następującymi parametrami:

```
AvailabilityZones = eu-west-1a
LoadBalancerName = MyLoadBalancer
Listeners = lb-port=80,instance-port=8080,protocol=HTTP
```

Wywołaj `CreateLaunchConfiguration` z następującymi parametrami:

```
ImageId = myAMI
LaunchConfigurationName = MyLaunchConfiguration
InstanceType = m1.small
```

Wywołaj `CreateAutoScalingGroup` z następującymi parametrami:

```
AutoScalingGroupName = MyAutoScalingGroup
AvailabilityZones = us-east-1a
LaunchConfigurationName = MyLaunchConfiguration
LoadBalancerNames = MyLoadBalancer
MaxSize = 20
MinSize = 2
```

Wywołaj `CreateOrUpdateScalingTrigger` z następującymi parametrami:

```
AutoScalingGroupName = MyAutoScalingGroup
MeasureName = CPUUtilization
Statistic = Average
TriggerName = MyTrigger1a
Namespace = AWS/EC2
Period = 60
LowerThreshold = 40
LowerBreachScaleIncrement = -1
UpperThreshold = 80
UpperBreachScaleIncrement = 1
BreachDuration = 600
```

W rozdziale 1. napisaliśmy, że istnieje więcej niż jeden typ przetwarzania w chmurze. Spróbujmy połączyć przedstawione wówczas informacje na temat różnych typów chmur z tym, co już wiesz o sześciu technologiach, bez których nie mogłaby istnieć chmura. W następnym podrozdziale postaramy się ułatwić Ci zrozumienie tego, czym różnią się typy, co mają do zaoferowania i jak działają. Umożliwi Ci to świadomy wybór najlepszego dla Ciebie rozwiązania.

2.2. Zrozumienie różnych typów chmur

Wiesz już, jakie są technologiczne podstawy chmury obliczeniowej. Rozumiesz, czym jest wirtualizacja, elastyczność, jakie są problemy ze składowaniem danych w chmurze. Przyjrzymy się teraz rozwiązaniom zastosowanym w różnych typach chmur. Wróćmy do taksonomii z rozdziału 1.: IaaS, PaaS, DaaS i spróbujmy zaklasyfikować chmury największych przemysłowych dostawców.

2.2.1. Amazon EC2: IaaS

Chmura EC2 należy do klasy IaaS (infrastruktura jako usługa). Niektórzy mówią, że to HaaS (sprzęt jako usługa), jednak firma Amazon wprowadziła tyle dodatkowych usług, że nazywanie tej chmury HaaS to nieporozumienie. Jest to pierwsza i jak dotąd największa chmura tej kategorii. Została udostępniona w 2006 roku, gdy firma chciała jakoś spożytkować nadmiar sprzętu niewykorzystywanego do operacji handlowych. Pod koniec 2008 roku firma miała już ponad pół miliona użytkowników.

Wśród największych istniejących chmur EC2 jest chmurą najogólniejszego typu. Ma ona najmniejsze wsparcie automatyczne dla skalowania i korekty błędów — te wymagania musi obsłużyć sama aplikacja. Odróżnia to EC2 od automatycznie skalujących aplikacje chmur typu PaaS, takich jak Google App Engine, o której będzie mowa w punkcie 2.2.3. W chmurach typu IaaS, takich jak EC2, elastyczność musi zostać starannie zaprogramowana w oparciu o API chmury. Plusem jest to, że programista może wybrać dowolny język programowania i ma całkowitą kontrolę nad aplikacją. Konieczna jest ręczna obsługa, jednak w zamian dostaje się odpowiednik fizycznej maszyny, na której można zainstalować system i wszystkie inne wymagane elementy. Najpopularniejsza konfiguracja EC2 to tak zwany stos LAMP (tabela 2.6).

Tabela 2.6. Komponenty stosu LAMP w chmurze IaaS

L	Linux	System operacyjny
A	Apache	Serwer sieciowy
M	MySQL	Relacyjna baza danych
P	PHP	Obsługa strony internetowej po stronie serwera

Amazon zapewnia bardzo obszerne API dla wszystkich swoich usług, z których kilka przedstawiono w tabeli 2.7. API ma dwie wersje: jedna jest oparta o SOAP, druga o czysty HTTP (`GET`, `POST`). Żądanie i konfiguracja maszyny wirtualnej realizowane są w mniej niż dwudziestu wywołaniach.

Chmura EC2 i parawirtualizacja Xen

Chmura EC2 firmy Amazon korzysta ze zmodyfikowanej wersji otwartego monitora (VMM) Xen, stosując tak zwaną parawirtualizację. System operacyjny goszczący na maszynie parawirtualnej nie wykonuje samodzielnie operacji wymagających uprzywilejowanego dostępu, tylko korzysta z pośrednictwa monitora. W ten sposób zmniejsza się obciążenie procesora.

Parawirtualizacja jest wydajniejsza od zwykłej wirtualizacji, w której system operacyjny nie jest modyfikowany. Jednak system ten trzeba dostosować do parawirtualnego środowiska. Monitor powinien „oddać" niektóre wolniej przebiegające zadania systemowi operacyjnemu, by wykonał je bezpośrednio. Z tego powodu w chmurze Amazon nie można uruchomić dowolnego systemu operacyjnego, tylko te, które producent systemu dopasował i w pełni przetestował.

Tabela 2.7. Inne usługi Amazon związane z chmurą (zapewniające część funkcjonalności PaaS)

Zastosowanie bazy	Wyzwania w przypadku bazy w chmurze
Simple Storage Service (S3)	Magazyn danych w chmurze, w którym można przechowywać duże ilości danych. Dostęp jest możliwy z każdego miejsca w sieci za pośrednictwem prostego API. Dobrze zintegrowany z EC2: w S3 są przechowywane AMI, przesyłanie danych pomiędzy S3 i EC2 jest zwolnione z opłat.
SimpleDB	Zapewnia podstawowe funkcje bazodanowe: indeksy (specjalne struktury organizacyjne przyspieszające wyszukiwanie) i zapytania. Pozwala uniknąć kosztów związanych z licencjami na relacyjne bazy danych i administracją, a także nie wymaga złożonej konfiguracji. Nie jest to relacyjna baza danych: nie ma schematu i nie można jej odpytywać za pomocą SQL.
CloudFront	Usługa sieciowa dostarczająca treści, konkurująca z Akamai. Pozwala na wygodną i szybką dystrybucję treści między użytkowników w modelu opłat w miarę zużycia.
Simple Queue Service (SQS)	Zarządzana kolejka komunikatów przesyłanych pomiędzy komputerami. Ułatwia przesyłanie danych między rozproszonymi komponentami aplikacji wykonującymi różne zadania bez ryzyka utraty komunikatu i bez nakładania na komponenty obowiązku ciągłej dostępności.

Ceny EC2 zaczynają się od paru centów za godzinę używania procesora niewielkiej instancji linuksowej, najdroższe instancje to koszt około pół dolara za godzinę[7]. Ceny S3 zaczynają się od 15 centów za 1 GB na miesiąc. Im więcej przestrzeni wykorzysta użytkownik, tym mniejsza cena za 1 GB.

2.2.2. Microsoft Azure: IaaS

Azure, podobnie jak EC2, należy do klasy IaaS, ale oferuje także dodatkowe usługi, działające bardziej na poziomie PaaS. Wiele aplikacji dostarczanych przez Microsoft użytkownikom końcowym przenosi się obecnie do chmury. W rezultacie cała platforma zmierza w kierunku poziomu SaaS (oprogramowanie jako usługa), by zablokować zapędy Google do przejęcia rynku należącego do Microsoft Office za pomocą Dokumentów Google i Google Apps.

Element opisany jako Windows Azure na rysunku 2.6. to system Windows Server 2008 zmodyfikowany pod kątem pracy w chmurze. Jest to kolejny przykład parawirtualizacji, której celem jest efektywne działanie systemu w środowisku wirtualnym pod kontrolą narzędzia monitorującego Microsoft Hypervisor, uruchamianego na czystym sprzęcie w centrach danych Microsoftu.

Wewnętrznie warstwa systemowa — odziedziczona po Windows Server 2008 — składa się z czterech filarów. Są to: składowanie danych (jak system plików), kontroler odpowiadający za zarządzanie modelowaniem, wdrażaniem i realizacją zapotrzebowania na zasoby, maszyna wirtualna i środowisko programistyczne, które umożliwia programistom emulację Windows Azure na ich maszynach

[7] *http://aws.amazon.com/ec2/pricing/*.

Rysunek 2.6. Architektura i framework Windows Azure. Na najniższym poziomie działa system operacyjny Azure. Jest on uruchamiany w wirtualnym środowisku stworzonym przez monitor Microsoft Hypervisor na czystym sprzęcie. W najwyższej warstwie działają aplikacje przeznaczone dla użytkowników końcowych, które Microsoft przekształca do postaci SaaS. Źródło: Microsoft

oraz wykorzystanie do pisania aplikacji takich narzędzi, jak Visual Studio czy Eclipse. Dzięki takiej architekturze wystarczy wdrożyć Azure na pojedynczej maszynie i zduplikować instancje na pozostałych serwerach z wykorzystaniem technologii wirtualizacyjnej.

Aplikacje działające w chmurze Azure pisze się w środowiskach programistycznych tej firmy, na przykład w Visual Studio, z wykorzystaniem bibliotek .NET i kompiluje się je do Common Language Runtime, czyli środowiska uruchomieniowego niezależnego od języka.

Windows Azure API

API Windows Azure jest oparte na architekturze REST i korzysta z certyfikatów X.509 do autentykacji użytkowników. Tabela 2.8 przedstawia fragment API, który daje pojęcie o tym, jak zarządza się aplikacjami działającymi w chmurze Azure. Są to wywołania związane z podstawowymi operacjami na wdrożonych usługach.

Podobnie jak w chmurze Amazon EC2 nad systemem Azure działa zestaw usług PaaS, które można wykorzystywać jako bloki do budowy aplikacji. Podstawowy zestaw takich usług to:

- Live Services,
- SQL Services,
- .Net Services,
- SharePoint Services,
- CRM Services.

Tabela 2.8. Fragment API Windows Azure. API jest zgodne z architekturą REST

Usługa	Opis
Wypisz dostępne usługi	Wypisanie usług dostępnych w ramach aktualnej subskrypcji.
`GET https://management.core.windows.net/<id-subskrypcji>/services/hostedservices`	
Wypisz właściwości usługi	Zwraca właściwości systemowe wskazanej usługi. Obejmują one nazwę i typ usługi oraz nazwę grupy, do której należy, lub lokalizację usługi, jeśli usługa nie jest częścią grupy. Opcjonalnie podawana jest informacja o wdrożeniach usługi.
`GET https://management.core.windows.net/<id-subskrypcji>/ ↪services/hostedservices/<nazwa-usługi>`	
Utwórz wdrożenie	Wdrożenie specyfikuje się w pokazany niżej sposób. Można je usunąć, podając nazwę slotu wdrożenia (przygotowanie lub produkcja) albo unikalną nazwę wdrożenia.
`GET https://management.core.windows.net/<id-subskrypcji> ↪/services/hostedservices/<nazwa-usługi>/deploymentslots/ ↪<slot-wdrożenia>/`	
`GET https://management.core.windows.net/<id-subskrypcji>/services/ ↪hostedservices/<nazwa-usługi>/deployments/<nazwa-wdrożenia>/`	
Przerzuć wdrożenie	Rozpoczyna proces wymiany IP pomiędzy slotem przygotowawczym a slotem produkcyjnym usługi. Jeśli usługa obecnie działa w środowisku przygotowawczym, zostanie przeniesiona do środowiska produkcyjnego. Jeśli działa w środowisku produkcyjnym, trafi do przygotowawczego. Jest to operacja asynchroniczna, której status należy sprawdzać przy użyciu wywołania specjalnego wywołania.
`POST https://management.core.windows.net/<id-subskrypcji>/hostedservices/ ↪<nazwa-usługi>`	
Usuń wdrożenie	Usuwa dane wdrożenie. Jest to operacja asynchroniczna.
`DELETE https://management.core.windows.net/<id-subskrypcji>/ ↪services/hostedservices/<nazwa-usługi>/deploymentslots/ ↪<slot-wdrożenia>`	
`DELETE https://management.core.windows.net/<id-subskrypcji>/ ↪services/hostedservices/<nazwa-usługi>/deployments/ ↪<nazwa-wdrożenia>`	

Usługi te można traktować jako API (nie mają one interfejsów użytkownika) do budowy aplikacji w chmurze.

Ceny Azure są zbliżone do cen w chmurze Amazon. Czas obliczeniowy to koszt 12 centów za godzinę, przechowywanie danych to 15 centów za 1 GB, a przesyłanie danych zaczyna się od 1 centa za 10 KB. Baza danych to koszt 9,99 dolara za 1 GB miesięcznie w przypadku edycji sieciowej, a 99,99 dolara miesięcznie za 10 GB w edycji biznesowej. Wkrótce ma pojawić się warstwowy model typu „wszystko, co zdołasz zjeść" (w pewnych granicach).

2.2.3. Google App Engine: PaaS

App Engine to chmura całkowicie zgodna z definicją „platforma jako usługa". Jest ona przeznaczona do uruchamiania tradycyjnych aplikacji internetowych, które są zmuszane do wyraźnego oddzielenia bezstanowej warstwy obliczeniowej od stanowej warstwy danych. Wirtualizacja i elastyczność, dobrze widoczne w modelu IaaS, tutaj pozostają prawie całkowicie niezauważalne, jednak za kulisami odgrywają naprawdę ważną rolę. Jedną z zalet tego modelu jest automatyczna elastyczność przy zmianie zapotrzebowania.

Języki programowania dozwolone w App Engine to Python i Java. App Engine nie jest chmurą ogólnego przeznaczenia. Najlepiej sprawdza się w przypadku aplikacji internetowych o strukturze żądanie-odpowiedź, w których pomiędzy wywołaniami występują długie okresy bez używania procesora (przykładowo użytkownik zastanawia się nad wyborem opcji). Z tego powodu Google surowo racjonuje czas korzystania z procesora przeznaczony do obsługi każdego z żądań.

Mechanizmy automatycznego skalowania i dużej dostępności oraz własny system składowania danych MegaStore (oparty na BigTable) zakładają zachowanie opisanych powyżej ograniczeń. Jednak jeśli Twoja aplikacja wpisuje się w te warunki, to najprawdopodobniej nie znajdziesz szybszego i tańszego sposobu budowania jej w taki sposób, by automatycznie się skalowała i działała w największej chmurze na planecie.

Środowisko programistyczne App Engine

- **Piaskownica** (ang. *sandbox*) — Aplikacje działają w bezpiecznym środowisku zapewniającym jedynie ograniczony dostęp do systemu operacyjnego. Te ograniczenia pozwalają App Engine na rozpraszanie żądań wysyłanych do aplikacji na wiele serwerów, uruchamianych i zatrzymywanych stosownie do zmieniającego się zapotrzebowania. Piaskownica izoluje aplikację we własnym, bezpiecznym środowisku niezależnym od sprzętu, systemu operacyjnego i fizycznej lokalizacji na serwerze.
- **Środowisko uruchomieniowe języka Java** — Możesz pisać aplikacje oparte o Javę 6 z użyciem popularnych narzędzi programistycznych i API do tworzenia aplikacji internetowych. Aplikacja komunikuje się ze środowiskiem uruchomieniowym przy użyciu standardu Java Servlet i może korzystać z popularnych technologii, takich jak Java Server Pages (JSP). Aplikacje odwołują się do większości usług App Engine za pośrednictwem standardowych interfejsów języka Java. Środowisko zawiera platformę i biblioteki Java SE Runtime Environment (JRE) 6. Ograniczenia związane z istnieniem piaskownicy zostały zaimplementowane wewnątrz maszyny wirtualnej Javy (JVM). Aplikacja może korzystać z dowolnej funkcjonalności kodu bajtowego lub biblioteki, o ile nie złamie w ten sposób narzuconych ograniczeń.

- **Środowisko uruchomieniowe języka Python** — Możesz pisać aplikacje w języku Python 2.5 i uruchamiać je przy użyciu interpretera. App Engine zawiera narzędzia i API do tworzenia aplikacji internetowych w Pythonie, w tym API do modelowania danych, framework aplikacji webowych i narzędzia umożliwiające zarządzanie danymi i dostęp do nich. Środowisko Pythona posiada standardowe biblioteki dostosowane do ograniczeń piaskownicy. Kod aplikacji uruchamianych w tym środowisku musi być napisany w czystym Pythonie. Środowisko Pythona daje dostęp do API przeznaczonego do magazynowania danych, pozyskiwania URK, Google Accounts i usług poczty elektronicznej.
- **Magazyn danych** — App Engine oferuje rozproszoną usługę składowania danych, obsługującą język zapytań i transakcje. Rozproszony magazyn skaluje się w zależności od zapotrzebowania. Zgodnie z tym, co powiedzieliśmy wcześniej na temat chmurowych baz danych, magazyn danych App Engine nie przypomina tradycyjnej relacyjnej bazy danych. Obiekty danych (**encje**) mają typ oraz zbiór właściwości. Zapytania pobierają encje danego typu, odfiltrowane i posortowane według wartości właściwości. Wartości właściwości mogą należeć do jednego ze wspieranych typów właściwości. Encje w magazynie danych są **pozbawione schematu**. To kod aplikacji odpowiada za określenie struktury encji. Może wykorzystać do tego interfejsy JDO/JPA albo interfejs magazynu danych Pythona.

Chmura App Engine jest darmowa, jeśli spełnione są następujące warunki: 6,5 godziny czasu procesora i 1 GB przesyłanych w obie strony danych. Po przekroczeniu tych progów za dane wysyłane przez aplikację płaci się 12 centów za 1 GB, 10 centów za dane przychodzące. Czas procesora kosztuje 10 centów za godzinę, a składowanie danych 15 centów za 1 GB na miesiąc. Wysłanie wiadomości e-mail do jednej osoby kosztuje 0,0001 dolara.

2.2.4. Ruby on Rails w chmurze: PaaS

Ruby on Rails (RoR) to otwarty framework do tworzenia aplikacji internetowych w języku Ruby. Z założenia powinien być używany zgodnie z metodologią *agile* („zwinną"). Programiści często wybierają go ze względu na łatwość tworzenia niewielkich projektów skoncentrowanych wokół klienta. Podobnie jak w przypadku Google App Engine aplikacje muszą działać na zasadzie żądanie-odpowiedź.

Otwarte oprogramowanie (open source) Oprogramowanie *open source* wyróżnia się tym, że kod źródłowy oraz pewne prawa, które normalnie są chronione, udostępnia się użytkownikom na pewnej specjalnej licencji, umożliwiającej analizę, wprowadzanie zmian, a także poprawianie oprogramowania. Niektórzy nazywają *open source* filozofią, inni pragmatyczną metodologią. Zanim termin ten się przyjął, programiści i producenci używali różnych nazw na opisanie tego pomysłu. To określenie zyskało

popularność wraz z rozwojem internetu i związaną z nim potrzebą przekształcenia istniejących narzędzi i kodu. Otwarte oprogramowanie najczęściej jest tworzone publicznie przez grupę współpracujących programistów.

Język Ruby zaprojektowano tak, by łączył konceptualną elegancję Smalltalka, łatwość nauki i użycia Pythona oraz pragmatyzm Perla. Wiele zespołów odnotowało dziesięciokrotne przyspieszenie wytwarzania aplikacji internetowych z użyciem frameworku RoR. Jednak wielu programistów donosi o problemach ze skalowaniem, które mimo wszystko mają chyba źródło w architekturze i innych wyborach projektowych, a nie wynikają z charakterystyki samego języka.

Wiele małych firm wykorzystało sytuację, oferując stosy RoR działające w chmurze Amazon EC2. Przykłady to Heroku, Aptana i EngineYard.

2.2.5. Salesforce.com i Force.com: PaaS

Firma Salesforce.com stworzyła odnoszącą największe sukcesy aplikację SaaS, służącą do zarządzania relacjami z klientami (CRM). Działa ona jako typowa aplikacja w chmurze już od 1999 roku.

Force.com to oferta PaaS tej samej firmy. Programiści mogą tworzyć w języku Apex aplikacje — nakładki na podstawową aplikację Salesforce, działające także w chmurze. Firmy Google i Salesforce zintegrowały swoje chmury App Engine i Force.com w taki sposób, że możliwe jest stworzenie w oparciu o dowolną z nich aplikacji, która ma dostęp do repozytorium danych firmowych.

Force.com utrzymuje katalog aplikacji o nazwie AppExchange, umożliwiający zakup i dodanie do środowiska Salesforce aplikacji napisanych przez zewnętrzne podmioty. Dostępne jest ponad 800 aplikacji od ponad 450 niezależnych producentów. Cena za korzystanie z Force.com to 5 dolarów za logowanie, przy czym użytkownik ma prawo zalogować się maksymalnie pięć razy w miesiącu. Na stronie firmy napisano: „Chmura Force.com jest przeznaczona dla okazjonalnie korzystających z niej użytkowników i dużych aplikacji, jest dostępna jako platforma i nie służy do tworzenia aplikacji CRM".

Opiszemy teraz jeszcze jedną, dość dziwną klasę chmur. Różni się ona od poprzednich typów nie środowiskiem działania aplikacji, ale strukturą własności. Nazwiemy tę klasę „centrum danych jako usługa" (DaaS, ang. *Datacenter as a Service*). Należą do niej prywatne chmury firm, przeznaczone do użytku wewnętrznego.

2.2.6. Chmury prywatne: DaaS (centrum danych jako usługa)

Chmura prywatna (nazywana także **chmurą wewnętrzną**, **korporacyjną** bądź **firmową**) to termin określający architekturę, w której usługi są dostępne dla ograniczonej liczby użytkowników znajdujących się za firewallem. W takiej chmurze stosuje się te same rozwiązania co w chmurach Amazon, Microsoft czy Google — wirtualizację, automatyzację, rozproszone przetwarzanie — tyle że jedynie na potrzeby klientów wewnątrz firmy.

Skorowidz

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

Ś

T

U

V

W

X

Z

Jeszcze parę lat temu udostępnienie dużej, popularnej aplikacji wiązało się z ogromnymi wydatkami na infrastrukturę. Konieczne było posiadanie własnej serwerowni, wynajmowanie przestrzeni w centrum danych lub uciekanie się do innych kosztownych rozwiązań. W tej chwili na zawołanie można otrzymać dokładnie tyle mocy obliczeniowej i przestrzeni dyskowej, ile w danej chwili jest potrzebne. Zmartwienia związane z nagłymi i chwilowymi wzrostami obciążenia odeszły na zawsze, a dostępność Twoich aplikacji na poziomie bliskim 100% przez okrągły rok jest w zasięgu ręki. Jak to możliwe?

Książka ta wprowadzi Cię w świat, jakiego nie znałeś. Dowiesz się, czym są chmury obliczeniowe, kiedy z nich korzystać i co w nich umieszczać. Poznasz obecnych na rynku dostawców i ich platformy: Google App Engine, Amazon EC2, Windows Azure oraz Salesforce.com i Force.com. Każda z nich ma swoje mocne i słabe strony oraz sprawdza się najlepiej w innych rozwiązaniach. Po lekturze tej książki bezbłędnie wybierzesz najlepsze z nich — idealnie dopasowane do Twoich potrzeb. W kolejnych rozdziałach autorzy poruszają kwestie związane z bezpieczeństwem w chmurze, omawiają najlepsze wzorce dla działających w niej aplikacji oraz sposoby szacowania kosztów przechowywania danych. Znajdziesz tu też sposoby prowadzenia testów i wdrażania aplikacji w chmurach. Chmury obliczeniowe są przyszłością świata informatyki — głównie w sferze biznesu. Nie pozostawaj w tyle i już dziś sięgnij po kompendium, które otworzy przed Twoją firmą nowe możliwości!

- Zasady definiujące przetwarzanie danych w chmurze
- Historia chmur obliczeniowych i ich klasyfikacja
- Najnowsze technologie, najwięksi dostawcy, najlepsze praktyki
- Kiedy korzystać z chmury, a kiedy jej unikać
- Tworzenie skalowalnych i niezawodnych aplikacji w chmurze
- Przenoszenie do chmury istniejących aplikacji i infrastruktury
- Niezawodność i bezpieczeństwo chmur obliczeniowych
- Testy, wdrożenia i działanie w chmurze

WYJDŹ NAPRZECIW NOWYM TECHNOLOGIOM I PRZENIEŚ SWÓJ BIZNES DO CHMURY!

MANNING

Nr katalogowy: 7450

Księgarnia internetowa
http://helion.pl

Zamówienia telefoniczne:
0 801 339900
0 601 339900

Helion

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

helion.pl
księgarnia internetowa

Informatyka w najlepszym wydaniu

Cena 49,00 zł

ISBN 978-83-246-3416-3

9 788324 634163