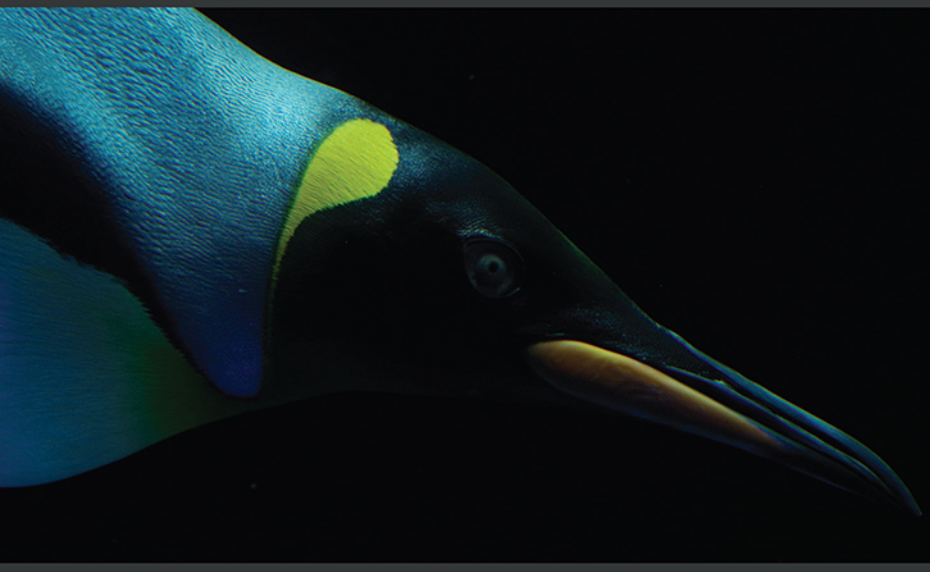
Linux dla admina

Najlepsze praktyki

O czym pamiętać podczas projektowania i zarządzania systemami



Scott Alan Miller

Helion

Tytuł oryginału: Linux Administration Best Practices: Practical solutions to approaching the design and management of Linux systems

Tłumaczenie: Robert Górczyński

ISBN: 978-83-289-0071-4

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

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Printed in Poland.

Spis treści

CZĘŚĆ 2. Najlepsze praktyki związane z technologiami systemu Linux

Najlepsze praktyki dotyczące pamięci masowej

Rozdział 3

Prawdopodobnie najbardziej skomplikowane i jednocześnie najmniej zrozumiałe aspekty **administrowania systemem** obejmują obszar **pamięci masowej**. Temat pamięci masowej jest zwykle omawiany po macoszemu i często traktowany w kategoriach mitu, a nie nauki. Pamięć masowa budzi również największe obawy, ponieważ popełnienie związanego z nią błędu może prowadzić do utraty danych. Nie może być większego niepowodzenia niż utrata danych.

Decyzje dotyczące pamięci masowej mogą wpływać na wydajność działania, pojemność, długowieczność, a co najważniejsze — na *trwałość*. W przypadku pamięci masowej mamy również najmniejszy margines błędu, pomyłka może mieć poważne konsekwencje. W innych obszarach planowania i projektowania często można sobie pozwolić na *drobne błędy*. Popełnione wówczas pomyłki nie będą miały aż tak bolesnych konsekwencji. Na przykład system nie będzie tak wydajny, jak powinien być, lub okaże się znacznie kosztowniejszy, niż przewidywano. Natomiast nadmiarowa pamięć masowa może podwoić koszt, a popełniony błąd bardzo łatwo może prowadzić do powstania niefunkcjonującego systemu. Z kolei awarie zdecydowanie nie będą przebiegały w elegancki sposób.

W rozdziale wyjaśnię, jak w systemie Linux patrzeć na pamięć masową i jak ją interpretować. Dość dokładnie ją omówię, aby umożliwić podejście do niej w sposób metodyczny i empiryczny. Dzięki lekturze rozdziału nauczysz się podejmować najlepsze decyzje dotyczące produktów związanych z pamięcią masową i projektować ją w taki sposób, aby uwzględnić wszystkie potrzebne rozwiązania.

Oto tematy, które zostały omówione w rozdziale:

- najważniejsze czynniki w obszarze pamięci masowej;
- poznanie blokowej pamięci masowej — lokalna i SAN;
- omówienie dyskowych i sieciowych systemów plików;
- poznanie **zarządcy woluminów logicznych** (**LVM**);

- wykorzystanie technologii RAID i RAIN;
- replikowana lokalna pamięć masowa;
- analiza architektury pamięci masowej i związanego z nią ryzyka.

Najważniejsze czynniki w obszarze pamięci masowej

Rozważając aspekty związane z pamięcią masową, podczas administrowania systemami pod uwagę bierze się następujące kwestie: **koszt**, **trwałość**, **dostępność**, **wydajność działania**, **skalowalność** i **pojemność**. Bardzo łatwo można zostać przytłoczonym tak dużą liczbą elementów związanych z pamięcią masową. To prowadzi do niebezpieczeństwa utraty orientacji, na co należy zwrócić szczególną uwagę. Podczas podejmowania każdej decyzji dotyczącej pamięci masowej należy mieć na względzie wymienione wcześniej czynniki. A co najważniejsze: trzeba o nich wszystkich pamiętać. Niezwykle kuszące może być skoncentrowanie się jedynie na kilku wybranych, ale to nie będzie dawało pełnego obrazu sytuacji.

W większości przypadków analiza systemów pamięci masowej, które nie spełniły wymagań biznesowych, wskazuje, że niemal zawsze na etapie planowania pominięto jeden lub więcej wymienionych wcześniej czynników. Bardzo kuszące może być skoncentrowanie się na jednym lub dwóch kluczowych aspektach oraz ignorowanie pozostałych. Jednak w celu osiągnięcia sukcesu związanego z pamięcią masową trzeba się skoncentrować na wszystkich wymienionych czynnikach.

W kolejnych punktach dokładnie omówię poszczególne kwestie.

Koszt

Może się wydawać, że nikt nie pominie aspektu kosztów, kiedy planuje rozwiązanie z zakresu pamięci masowej. Jednak możesz mi wierzyć, że to się zdarza i to zdumiewająco często. Z reguły dział informatyczny to funkcja biznesowa, a z definicji celem każdego biznesu jest zarabianie pieniędzy. Wobec tego koszt dostarczenia infrastruktury zawsze musi być uwzględniany. Dlatego też w informatyce, a także w innych dziedzinach biznesowych, żaden koszt nie powinien być ponoszony bez jego wcześniejszego przeanalizowania. Nigdy nie należy zapominać o kosztach. To jest błędem, podobnie jak stwierdzenie typu *pieniądze to nie problem*, ponieważ to nigdy nie może być prawdą i nie ma sensu. Wprawdzie koszt może mieć znaczenie drugorzędne, a budżet może zapewniać pewną elastyczność działania, ale mimo to koszt zawsze jest ważny.

Ogólnie rzecz biorąc, pamięć masowa to jeden z najdroższych komponentów systemu produkcyjnego. Dlatego też panuje tendencja większej wrażliwości na koszt pamięci masowej niż w innych fizycznych elementach systemu np. w procesorze lub pamięci operacyjnej. Problem związany z pamięcią masową często jest również najłatwiejszy do rozwiązania, wystarczy przeznaczyć na nią większą kwotę. Nic dziwnego, że na etapie planowania wiele osób popełnia błąd, wydając więcej na sprzęt, ponieważ to okazuje się bardzo łatwym rozwiązaniem. Oczywiście zawsze można się na to zdecydować, o ile zrozumiałe są wymagania dotyczące pamięci masowej i sposobu jej *działania*, przy czym takie rozwiązanie będzie kosztowne. Trudno być efektywnym administratorem systemu, jeśli pomija się kwestie efektywnego zarządzania kosztami — te dwa aspekty są ze sobą nierozerwalnie powiązane.

Trwałość

W pamięci masowej nic nie ma większego znaczenia niż trwałość. To oznacza odporność mechanizmu pamięci masowej na potencjalne awarie i utratę danych. Trwałość to jeden z dwóch aspektów niezawodności. W większości zadań i systemów trwałość ma największe znaczenie. Bardzo rzadko chcemy przechowywać pewne dane, których później nie będzie można niezawodnie pobrać, nawet jeśli ta operacja okazuje się powolna, opóźniona bądź kosztowna. Koncepcje takie jak dostępność danych lub wydajność działania oznaczają, że żadne dane nie będą utracone.

Trwałość odnosi się również do odporności na uszkodzenia bądź utratę danych. W systemie pamięci masowej trzeba również pamiętać o potencjalnym niebezpieczeństwie utraty spójności danych, co niekoniecznie może być możliwe do wykrycia. Sama możliwość pobrania danych niekoniecznie oznacza, że są one w oczekiwanej postaci. Uszkodzenie danych może oznaczać, że pliku nie można odczytać, baza danych nie pozwala dłużej na uzyskanie do niej dostępu, system operacyjny się nie uruchamia lub co gorsza — w programie księgowym nastąpiła zmiana wartości na inną, a taka zmiana jest praktycznie niemożliwa do wykrycia.

Dostępność

Niezawodność danych tradycyjnie rozważa się w kategoriach sposobu uzyskania dostępu do danych, gdy trzeba będzie je pobrać. Dostępność często określa się mianem **czasu nieprzerwanej pracy systemu**, tzw. *uptime*, a jeśli pamięć masowa jest niedostępna, wówczas nie będzie można wykonywać żadnych związanych z nią zadań. Dlatego też wprawdzie dostępność schodzi na dalszy plan względem trwałości, ale wciąż ma ważne znaczenie i stanowi jeden z dwóch kluczowych aspektów ogólnej niezawodności pamięci masowej.

Zdarzają się sytuacje, gdy dostępność i wydajność działania łączą się ze sobą. Musi dojść do tak dużego spadku wydajności działania pamięci masowej, że dane praktycznie będą niedostępne. Rozważ np. prysznic, z którego co kilka sekund spada kropla wody. Formalnie rzecz biorąc, z prysznica nadal leci woda, ale to nie ma nic wspólnego z silnym strumieniem wody, który zwykle wydostaje się z prysznica.

W dalszej części rozdziału znacznie dokładniej omówię macierz RAID, ale dostępność i wydajność działania to dobre, rzeczywiste przykłady. Klasyczną ilustracją jest sytuacja występująca w ogromnej macierzy RAID 6, gdy jeden lub dwa dyski uległy awarii. Po ich wymianie macierz jest w trybie online oraz w procesie aktywnego odtwarzania (to proces, w trakcie którego brakujące dane są odtwarzane na podstawie metadanych). W takich przypadkach system macierzy jest bardzo często obciążony ze względu na ilość przetwarzanych danych, które następnie są zapisywane w macierzy. Formalnie rzecz biorąc, macierz działa i jest dostępna, ale funkcjonuje na tyle wolno, że nie można sensownie z niej korzystać. System operacyjny lub aplikacje próbujące używać macierzy będą nie tylko bezużyteczne ze względu na drastyczny spadek wydajności działania, ale mogą również prowadzić do powstawania błędów na skutek zgłaszania informacji o niedostępności systemu pamięci masowej, co będzie wynikało z wyjątkowo długiego czasu udzielania odpowiedzi. Jeżeli nie zostanie zachowana ostrożność, wówczas *dostępność* może okazać się mętną koncepcją.

Wydajność działania

W przypadku komputerów XXI w. pamięć masowa niemal zawsze będzie najwęższym gardłem systemu. Procesor i pamięć operacyjna prawie w 100% będą musiały czekać na pamięć masową, a nie na odwrót. W nowoczesnej pamięci masowej są używane technologie półprzewodnikowe, dzięki którym udało się zmniejszyć przepaść w wydajności działania istniejącą między systemami pamięci masowej a innymi komponentami. Mimo to wspomniana przepaść nadal istnieje i jest dość duża.

Wydajność działania może być dość trudna do zmierzenia, ponieważ można to zrobić na wiele sposobów. Poszczególne rodzaje nośników pamięci masowej zwykle mają odmienne cechy charakterystyczne wydajności działania. Istnieją koncepcje takie jak opóźnienie (czyli czas upływający, zanim rozpocznie się pobieranie danych), **przepustowość** (określająca szybkość strumieniowego przesyłania danych), a także tzw. wartość IOPS, czyli liczba operacji wejścia-wyjścia przeprowadzanych w trakcie sekundy (liczba działań związanych z pamięcią masową, które można przeprowadzić w pewnym czasie). Większość osób rozważa pamięć masową jedynie w kategoriach przepustowości, podczas gdy tradycyjnie wartość IOPS okazuje się najbardziej użytecznym pomiarem wydajności działania w większości zadań.

Zawsze będzie kuszące zredukowanie czynników do postaci czegoś prostego, co można później zrozumieć i porównać. Rozważ przykład trzech pojazdów. Pierwszy to samochód, który ma duże przyspieszenie, ale małą szybkość maksymalną. Drugi to samochód,

który ma małe przyspieszenie, ale dużą szybkość maksymalną. Trzeci to ciągnik siodłowy, który ma małe przyspieszenie i małą szybkość maksymalną, ale za to może przewieźć ogromny ładunek. Pierwszy z wymienionych pojazdów będzie doskonały tylko wtedy, gdy interesuje nas najmniejsze opóźnienie: w pamięci masowej to czas, który musi upłynąć do pojawienia się pierwszego pakietu. Drugi z samochodów będzie użyteczny, jeśli interesuje nas jak najszybsze przemieszczenie się z jednego miejsca do drugiego. W przypadku pamięci masowej to można porównać do sprawdzenia wartości IOPS. Z kolei trzeci pojazd okaże się niezastąpiony, gdy trzeba będzie przewieźć ogromną ilość ładunku. W odniesieniu do pamięci masowej można to porównać do całkowitej ilości danych przekazywanych między systemami. To będzie nasza przepustowość. W przypadku samochodów większość osób myśli o *najszybszym pojeździe* o największej szybkości maksymalnej, natomiast w odniesieniu do pamięci masowej kluczowe znaczenie ma ilość przekazanych danych, a nie *odczucie*, jak szybko są one przekazywane. W rzeczywistości wydajność działania to jedynie kwestia perspektywy. W różnych zadaniach odmiennie postrzega się kwestię wydajności działania.

Na przykład podczas tworzenia kopii zapasowej dane są przekazywane stabilnie i strumieniowo. W takim razie najlepszym rozwiązaniem będzie system zaprojektowany pod kątem przepustowości. Dlatego też napędy taśmowe sprawdzają się podczas tworzenia kopii zapasowej, podobnie jak stare nośniki optyczne typu płyta CD lub DVD. Natomiast w innych zadaniach, np. w bazach danych, wymagających do sprawnego działania dużych wartości IOPS i małego opóźnienia, całkowita przepustowość będzie przynosiła tylko niewielkie korzyści. Największą korzyścią będzie zastosowanie napędu półprzewodnikowego typu SSD. Z kolei w innych zadaniach, np. w serwerach, dość często ważne znaczenie mają inne czynniki i w tych zadaniach świetnie się sprawdzają tradycyjne mechaniczne napędy dysków twardych. Trzeba znać zadania, które mają zostać wykonane, ponieważ to pozwoli na zaprojektowanie poprawnego systemu pamięci masowej.

Wydajność działania okazuje się jeszcze bardziej skomplikowana, gdy będzie rozważana w kategoriach wartości chwilowych i trwałych. Pod tym względem mamy wiele do omówienia i nie można tak po prostu skrócić tego procesu.

Skalowalność

Obecnie przyjmuje się założenie, że typowe fizyczne wdrożenie systemu w środowisku produkcyjnym powinno funkcjonować przez cztery do ośmiu lat. Nierzadko zdarza się słyszeć o systemach, które pozostawały w użyciu przez znacznie dłuższy czas. Jeżeli pozostaniesz w branży informatycznej przez dłuższy okres, prawdopodobnie spotkasz się z systemami, które od ponad 20 lat obsługują w firmie funkcjonalność o znaczeniu krytycznym. Skoro system pamięci masowej ma wytrzymać tak długo, konieczne jest wcześniejsze ustalenie, w jaki sposób ten system będzie mógł być rozbudowywany bądź też modyfikowany.

W większości przypadków oczekuje się, że niezbędna pojemność będzie się zwiększała wraz z upływem czasu. System pamięci masowej musi więc być zaprojektowany w sposób pozwalający na zwiększenie pojemności. Takie rozwiązanie okaże się korzystne nie tylko ze względu na ochronę danych przed nieprzewidzianymi zdarzeniami, ale także pozwoli na ograniczenie początkowego kosztu inwestycji i jego zwiększenie *tylko wtedy*, gdy będzie wymagana większa pojemność. Niektóre systemy pamięci masowej mogą być skalowalne również w kategoriach wydajności działania. To sytuacja rzadziej spotykana i rzadziej uznawana za funkcjonalność o znaczeniu krytycznym. Nawet jeśli wymagany jest jedynie wzrost pojemności pamięci masowych, a nie zwiększenie wydajności jej działania, to większa pojemność z reguły uzasadnia konieczność wzrostu wydajności działania podczas obsługiwania zadań takich jak tworzenie kopii zapasowej. W tym przypadku większa pojemność oznacza dłuższy czas tworzenia kopii zapasowej i przywracania z niej danych.

Teoretycznie można się spotkać z sytuacją, w której potrzeba niezawodności (trwałości, dostępności bądź też i jednego, i drugiego) może się rosnąć w miarę upływu czasu. Takie rozwiązanie również jest możliwe, chociaż będzie znacznie bardziej skomplikowane.

Pamięć masowa to obszar, w którym elastyczność w zakresie dostosowania konfiguracji w miarę upływu czasu jest często najtrudniejsza do zapewnienia, a jednocześnie ma największe znaczenie. Nie zawsze można przewidzieć przyszłe potrzeby. Konieczne jest jak najlepsze planowanie, aby zapewnić sobie elastyczność w zakresie dopasowania rozwiązania do własnych potrzeb.

Pojemność

Ostatnim czynnikiem jest pojemność, czyli ilość danych, które mogą być przechowywane w systemie pamięci masowej. Wprawdzie pojemność może się wydawać prostym aspektem, ale jednocześnie może być dezorientująca. Nawet w prostej macierzy bazującej na dyskach rozwiązanie trzeba przemyśleć w kategoriach pojemności nominalnej (czyli sumy pojemności wszystkich urządzeń) oraz w kategoriach pojemności użytkowej (czyli użytecznej pojemności, którą system może wykorzystać na potrzeby przechowywania danych). Wiele systemów pamięci masowej charakteryzuje się nadmiarowością w celu zapewnienia większej niezawodności i wydajności działania. Jednak to się wiąże z kosztem w postaci większego poziomu użycia pojemności nominalnej. Dlatego też trzeba wiedzieć, w jaki sposób konfiguracja pamięci masowej będzie wpływała na ostateczne rozwiązanie. Administratorzy zajmujący się pamięcią masową używają pojęć w kategoriach pojemności zarówno nominalnej, jak i użytecznej.

W ten sposób omówiłem ogólne aspekty systemu pamięci masowej, o których należy pamiętać. Mogę więc przejść do wyjaśnienia, w jaki sposób współpracują ze sobą komponenty pamięci masowej, a tym samym — jak powstają **podsystemy korporacyjnej pamięci masowej**.

Poznanie blokowej pamięci masowej — lokalna i SAN

Podstawą wszystkich standardowych mechanizmów obecnie używanej pamięci masowej jest koncepcja **urządzeń blokowych**. Urządzenie blokowe to urządzenie pamięci masowej pozwalające na trwałe przechowywanie danych, które mogą być pobierane w dowolnej kolejności. W praktyce przykładem *standardowego* urządzenia blokowego jest dysk twardy, który można uznać za prototypowe urządzenie blokowe, zaś wszystkie pozostałe urządzenia blokowe działają w sposób podobny do dysku twardego. Można to również określić mianem *implementacji interfejsu dyskowego*.

Wiele urządzeń zalicza się do kategorii urządzeń blokowych: tradycyjne, mechaniczne dyski twarde, napędy półprzewodnikowe SSD, napędy dyskietek, napędy optyczne (np. CD-ROM i DVD-ROM), napędy taśmowe, RAM-dyski, macierze RAID i in. Z perspektywy komputera wszystkie te urządzenia są takie same. To znacznie ułatwia pracę administratorowi systemu — wszystko zostało zbudowane na bazie urządzeń blokowych.

Administratorzy systemów często określają urządzenie blokowe po prostu jako *dysk*, ponieważ z perspektywy systemu operacyjnego nie można powiedzieć zbyt wiele na temat urządzenia poza tym, że zalicza się ono do kategorii urządzeń blokowych. Takim urządzeniem blokowym może być fizyczny dysk, urządzenie logiczne zbudowane na podstawie wielu dysków, abstrakcja utworzona na podstawie pamięci operacyjnej, napęd taśmowy, zdalny system itd. Tak naprawdę trudno to określić. To jest po prostu urządzenie blokowe, a skoro urządzenie blokowe, ogólnie rzecz biorąc, przedstawia dysk, to jest nazywane dyskiem. Takie określenie niekoniecznie odzwierciedla faktyczne urządzenie, ale jest użyteczne.

Lokalnie podłączona pamięć masowa

Najprostszy typ blokowego urządzenia pamięci masowej to urządzenie fizycznie podłączone do komputera. Doskonale znasz takie urządzenia, może być nim np. dysk twardy. Obecnie lokalne urządzenia pamięci masowej są podłączone za pomocą interfejsów typu SAS, SATA i NVME. W przeszłości były również stosowane standardy **Parallel SCSI** (nazywany wówczas **SCSI**) i **Parallel ATA** (czyli **PATA**), nazywany wówczas **ATA** lub **IDE**. Wszystkie te technologie oraz inne niszowe pozwalały na fizyczne podłączanie blokowej pamięci masowej bezpośrednio do komputera.

W większości przypadków mamy do czynienia z lokalnie podłączoną pamięcią masową. Każde urządzenie blokowe musi być gdzieś podłączone lokalnie, aby można było z niego korzystać. Dlatego też ta technologia zawsze jest ważna.

Lokalnie podłączane urządzenia blokowe mają wiele zalet. Podłączenie lokalne wiąże się z naturalną zaletą w postaci dużej wydajności działania i jednocześnie z ogromną trwałością — takie rozwiązanie jest możliwie jak najprostsze, więc niewiele może się w nim zepsuć. Gdy wszystkie pozostałe czynniki mają takie same znaczenie, prostota zyskuje przewagę nad złożonością. Pamięć masowa jest tego doskonałym przykładem. Im mniejsza liczba ruchomych elementów i krótsze połączenie, tym mniejsze opóźnienie, wyższa przepustowość, większa niezawodność i niższy koszt.

Oczywiście lokalnie dołączona pamięć masowa ma również pewne wady, ponieważ w przeciwnym razie nikt nie opracowywałby dla niej żadnych alternatyw. Największą wadą jest brak elastyczności. Po prostu zdarzają się sytuacje, w których lokalnie dołączona pamięć masowa się nie sprawdza i wówczas trzeba szukać alternatywnych rozwiązań.

Sieć pamięci masowej (SAN)

Logiczną alternatywą dla lokalnie podłączonego urządzenia pamięci masowej jest zdalnie podłączone urządzenie. Jeżeli sądzisz, że w taki sposób będziemy odwoływać się do tego typu urządzeń blokowych, to jesteś w błędzie. Zdalnie podłączone urządzenie blokowe używa protokołu sieciowego w celu zaimplementowania koncepcji **zdalnego dostępu** do pamięci masowej, a sieć pozwalająca na prowadzenie komunikacji z tym urządzeniem jest nazywana **siecią pamięci masowej** (ang. *storage area network*, SAN). Dlatego też wszystkie zdalne urządzenia blokowej pamięci masowej są określane jako **SAN**.

Okropna terminologia SAN

Formalnie rzecz biorąc, pojęcie sieć pamięci masowej powinno być odniesieniem do oddzielnej sieci, która jest używana do obsługi ruchu urządzeń blokowych. W kręgach technicznych to pojęcie jest stosowane w takim właśnie kontekście. Urządzenie w sieci SAN może być bezpośrednim urządzeniem blokowym, macierzą dysków bądź też innym podobnym *urządzeniem blokowym działającym poprzez sieć*. SAN oznacza tutaj sieć, a nie *produkt*, który można kupić.

W żargonie informatycznym przyjęło się, że SAN to standard odnoszący się do dowolnego urządzenia zapewniającego dostęp do pamięci masowej, implementującego interfejs urządzenia blokowego oraz nawiązującego połączenie z siecią, a nie bezpośrednio z komputerem. Codziennie można się spotkać z następującymi wyrażeniami: *Czy kupiłeś SAN?*; *Czy potrzebujesz inżyniera SAN?*; *Rozmawiałem z producentem SAN*; *Czy powinienem uaktualnić urządzenie SAN?*; *Gdzie jest nasz SAN?* itd. Udaj się do najbliższego sklepu komputerowego i zapytaj, czy możesz kupić SAN. Bez wątpienia uzyskasz odpowiedź twierdzącą. Ta terminologia jest powszechnie używana i daję głowę, że

sprzedawca będzie całkowicie zdezorientowany, jeśli spróbujesz użyć jej, jakby SAN to było urządzenie sprzętowe, w którym umieszcza się dyski twarde, a następnie za pomocą jakiegoś przewodu podłącza się urządzenie do sieci.

Już sama pamięć masowa jest skomplikowana, zagmatwana i przerażająca, sieć pamięci masowej jeszcze bardziej wszystko komplikuje, więc cała dziedzina SAN jest traktowana jako czarne „magiczne" pudełka. Do tego terminologia szybko przestała być precyzyjna, zaś większość opinii dotyczących SAN powstała na skutek niezrozumienia i mitów. Do często powtarzanych mitów zalicza się stwierdzenie, że SAN nie może ulec awarii; że SAN działa szybciej niż ta sama technologia, ale pozbawiona warstwy sieciowej; że technologia SAN jest wymagana przez inne technologie itd.

Aby być efektywnym administratorem systemu, trzeba dobrze poznać sposób działania używanych technologii i unikać przy tym mitów (oraz marketingu). Na przykład nie można przeprowadzić wiarygodnej analizy ryzyka lub nie można podjąć decyzji dotyczącej wydajności działania, wierząc, że urządzenie jest magiczne, i nie uwzględniając rzeczywistego profilu ryzyka.

Teoretycznie SAN to wyjątkowo prosta koncepcja. Wystarczy wziąć dowolne urządzenie blokowe, np. fizyczny dysk twardy, lub znacznie bardziej zaawansowane rozwiązanie, np. macierz dysków, a następnie hermetyzować je za pomocą standardowego protokołu urządzenia blokowego (takiego jak SCSI lub SATA) i przekazać przez protokół sieciowy (taki jak TCP/IP, Ethernet lub *FibreChannel*). Tutaj protokół sieciowy działa w charakterze prostego tunelu w celu zapewnienia obsługi protokołu blokowego na odległość. I na tym koniec, do tego sprowadza się rozwiązanie SAN. Ostatecznie to jest po prostu urządzenie typu SCSI lub ATA, które teraz może być używane na dużą odległość.

Skoro rozwiązanie stało się nieco bardziej skomplikowane, SAN jest automatycznie znacznie bardziej zawodne niż lokalna pamięć masowa. Pozostaje całe ryzyko i poziom skomplikowania znane z lokalnej pamięci masowej, a ponadto dochodzi nowe ryzyko i nowe komplikacje związane z obsługą warstwy sieciowej i sprzętu sieciowego. To ryzyko się kumuluje. Ponadto dodatkowa warstwa sieciowa, przetwarzanie i fizyczna odległość muszą się przekładać na większe opóźnienie transakcji pamięci masowej.

Z powodu wymienionych czynników pamięć masowa bazująca na SAN zawsze będzie wolniejsza i bardziej zawodna niż odpowiadająca jej pamięć masowa, ale podłączona lokalnie. Różne czynniki powtarzane w mitach w celu promowania SAN okazują się słabością takiego rozwiązania.

Oczywiście rozwiązanie bazujące na SAN ma swoje zalety, w przeciwnym razie nie zostałoby nigdy opracowane. SAN zapewnia obsługę trzech funkcjonalności o znaczeniu krytycznym: odległości, konsolidacji i połączeń współdzielonych.

Odległość może być dowolna, od kilku metrów aż po drugi koniec świata. Oczywiście im większa odległość, tym większe opóźnienie. Pamięć masowa jest zwykle bardzo wrażliwa na opóźnienie, więc dość rzadko zdarza się, że zdalna blokowa pamięć masowa

będzie użyteczniejsza niż pamięć masowa podłączona lokalnie. Jeżeli zdecydujesz się używać pamięci masowej poprzez sieć, prawdopodobnie zaobserwujesz duże opóźnienie, które może drastycznie zmniejszyć wydajność działania. Ponadto pojawiają się ograniczenia wynikające z przepustowości sieci. W typowym produkcyjnym rozwiązaniu blokowej pamięci masowej przyjmuje się, że przepustowość jest liczona w GB/s (to gigabajty, a nie gigabity na sekundę), natomiast opóźnienie wynosi poniżej milisekundy. W połączeniu sieciowym jego przepustowość rzadko osiąga gigabit na sekundę (Gb/s), zaś opóźnienie zwykle wynosi co najmniej kilka milisekund.

Konsolidacja była tradycyjnie uznawana za ogromną zaletę SAN. Ponieważ wiele komputerów można fizycznie dość łatwo połączyć z pojedynczą macierzą pamięci masowej, używając do tego sieci, po raz pierwszy pojawiła się możliwość inwestycji w jeden droższy system pamięci masowej, który jednocześnie mógł być używany przez oddzielne komputery. Pamięć masowa w urządzeniu zostanie *podzielona*, zaś każdy komputer podłączony do takiego systemu będzie miał dostęp do unikatowego fragmentu pamięci masowej.

Kiedy lokalna pamięć masowa tak naprawdę nie jest lokalna?

Biorąc pod uwagę te wszystkie interfejsy, abstrakcje i nieprawidłową terminologię stosowaną w informatyce, w większości przypadków można bardzo łatwo stracić rachubę, co tak naprawdę się dzieje. SAN to jedno z tych miejsc, w których można się pomylić. Działanie technologii SAN polega na wykorzystaniu urządzenia blokowego, które znajduje się daleko, i udostępnienie go komputerowi, jakby było podłączone do niego lokalnie. Z drugiej strony można użyć urządzenia lokalnego i udostępnić je jako lokalne, podczas gdy tak naprawdę jest zdalne. Czy ja to naprawdę powiedziałem?

Najlepszym przykładem będzie tutaj zewnętrzny dysk twardy podłączony przez port USB. Praktycznie każdy korzysta z takiego dysku, są one powszechne dostępne. Bez problemu znajdziesz go w lokalnym sklepie komputerowym. Możesz go również zamówić w sklepie internetowym. Prawdopodobnie kilka masz również w szufladzie i nawet o nich nie pamiętasz. Napęd USB wprawdzie jest zewnętrzny, ale oczywiście pozostaje lokalny, prawda?

No cóż, to nie tak łatwo wyjaśnić. Nie ulega wątpliwości, że taki dysk fizycznie znajduje się blisko komputera. Jednak z technologicznego punktu widzenia zdalny oznacza, że *coś jest podłączone przez sieć*, natomiast lokalny to *niepodłączony przez sieć*. Odległość nie ma tutaj żadnego znaczenia, to aspekt sieciowy określa, czy coś będzie uznane za lokalne, czy za zdalne. W przeciwnym razie można byłoby powiedzieć, że mój komputer stacjonarny znajdujący się w Teksasie jest połączony z laptopem mojego taty w Nowym Jorku, ponieważ między tymi dwoma urządzeniami znajduje się mnóstwo przewodów.

W ten sposób powstaje bardzo interesujące zagadnienie, ponieważ USB to w rzeczywistości prosty protokół sieciowy, podobnie jak IEEE 1394 i Thunderbolt. Jeżeli fizycznie przeanalizujesz zewnętrzny dysk twardy, możesz to dość łatwo dostrzec, przynajmniej

do pewnego stopnia. Taki napęd składa się ze standardowego dysku twardego, najczęściej używającego interfejsu SATA, oraz niewielkiej karty sieciowej. Jej zadaniem jest hermetyzacja protokołu SATA w protokole sieciowym USB, który następnie może być przekazywany przez sieć (bardzo często na odległość wynoszącą jedynie około metra).

USB i podobne protokoły mogą nie być odbierane jako protokoły sieciowe, choć w rzeczywistości nimi są. Mamy tutaj warstwę dwóch protokołów sieciowych konkurującą z Ethernetem oraz pozwalającą na podłączanie wielu urządzeń do wielu komputerów, a nawet używanie urządzeń typu np. przełącznik sieciowy. To jest rzeczywista platforma sieciowa, co oznacza, że zewnętrzny dysk twardy podłączony za pomocą USB jest faktycznie małym rozwiązaniem typu SAN. Wprawdzie trudno w to uwierzyć, ale tak właśnie jest. Możesz być oficjalnie zszokowany.

Pamięć masowa to największy koszt większości systemów komputerowych. Jeżeli będzie mogła być efektywnie współdzielona, pozwoli na znaczne obniżenie kosztów wdrożenia nowego systemu komputerowego. Dysk twardy może mieć pojemność np. 1 TB, podczas gdy pojedynczy komputer wymaga jedynie 80 – 300 GB miejsca na dysku twardym. Dzięki współdzielonemu rozwiązaniu typu SAN setki komputerów mogą korzystać z pojedynczej macierzy pamięci masowej, a każdy z nich wykorzysta jedynie tyle miejsce, ile będzie potrzebował. Obecnie największą wydajność lokalnej pamięci masowej można uzyskać dzięki wirtualizacji. Zanim rozwiązania w zakresie wirtualizacji stały się powszechnie dostępne, tylko systemy typu SAN były w stanie zapewnić te oszczędności. Po pojawieniu się pierwszych urządzeń typu SAN największy nacisk był kładziony na zmniejszenie kosztów. Dopiero później utworzono kolejne funkcjonalności. Obecnie te proporcje uległy odwróceniu. Rozwiązanie jest znacznie droższe niż posiadanie nadmiarowej ilości lokalnej pamięci masowej, ale mimo to nadal się sprawdza w niektórych sytuacjach.

Ostatnią zaletą są współdzielone połączenia. To oznacza, że dwa lub więcej komputerów może jednocześnie uzyskać dostęp do tego samego fragmentu pamięci masowej, czyli do tych samych danych. Wprawdzie może to zabrzmieć jak tradycyjne udostępnianie plików, ale w rzeczywistości to zupełnie inne rozwiązanie.

W przypadku współdzielenia plików komputer nawiązuje połączenie ze *sprytnie* działającym urządzeniem odpowiedzialnym za udzielanie dostępu do plików. Należy pamiętać, że rozwiązanie typu SAN zawiera po prostu *zwykłe* urządzenie blokowe, które nie ma zaimplementowanej własnej logiki. Używanie SAN w celu dołączenia jednego lub więcej systemów komputerowych do pojedynczego logicznego urządzenia blokowego oznacza, że z perspektywy danego komputera pamięć masowa należy całkowicie do niego. Pamięć masowa jest więc postrzegana jako prywatny i w pełni odizolowany system nieposiadający żadnych informacji na temat innych systemów, które mogłyby być do niego podłączone. To może prowadzić do wszelkiego rodzaju problemów, począwszy od utraty zmian w plikach poprzez uszkodzenie plików aż po zniszczenie

systemów plików. Oczywiście istnieją mechanizmy pozwalające na współdzielenie pamięci masowej. Jednak z definicji nie są one implementowane w rozwiązaniu typu SAN, a powinny być dostarczone na wyższym poziomie systemów komputerowych.

Współdzielone połączenia SCSI

Przed pojawieniem się SAN i zanim rozwiązania tego typu stały się popularne i powszechnie dostępne, istniała inna technika pozwalająca dwóm komputerom na współdzielenie tej samej puli dysków twardych — współdzielone SCSI.

Dzięki tej technice za pomocą pojedynczego kabla taśmowego SCSI można było podłączyć do 8, 16, a nawet 32 urządzeń. Jedno z tych urządzeń musiało być kontrolerem, najczęściej było one wbudowane na płycie głównej komputera. Pozostałe połączenia były dostępne dla dysków twardych. Za pomocą innego złącza można było połączyć się z kolejnym kontrolerem w drugim komputerze, więc te dwa komputery mogły się nawzajem widzieć i uzyskiwać dostęp do tych samych dysków.

Ze względu na ograniczenia związane z koniecznością dzielenia jednego kabla taśmowego między dwoma fizycznymi urządzeniami ta technika była wyjątkowo ograniczona i niewygodna, choć jak najbardziej możliwa do zastosowania. Podstawową zaletą takiej konfiguracji było to, że jeśli jeden komputer uległ awarii, drugi mógł przejąć jego zadania. Ponadto można było podwoić zasoby procesora lub pamięci operacyjnej przypisane do jednego komputera, a tym samym wykorzystać więcej zasobów, niż mogło się zmieścić w pojedynczej obudowie komputera. Jednak ograniczenia dotyczące niezawodności i wydajności działania komponentu pamięci masowej spowodowały, że na ogół ten system był mniej niż praktyczny, a sama technika rzadko implementowana w rzeczywistości. Mimo to sama technika okazała się bardzo ważna, ponieważ dostarczyła podstaw dla nowoczesnej współdzielonej blokowej pamięci masowej. Jej znajomość była oczekiwana w trakcie szkoleń systemowych prowadzonych w późnych latach 90. ubiegłego wieku, a także pomaga w przedstawieniu sposobu działania obecnych rozwiązań typu SAN — są znacznie bardziej eleganckie i elastyczne, choć pod względem koncepcji to ta sama technika.

Obecnie najważniejszym zastosowaniem połączeń współdzielonej blokowej pamięci masowej są klastrowane systemy, które zostały przeznaczone do używania tego rodzaju pamięci masowej jako współdzielonego komponentu dla wirtualizacji. Około 2010 r. to był szczyt mody, ale od tego czasu ta metoda ustąpiła miejsca innym rozwiązaniom. To raczej będzie specjalny rodzaj projektu systemu. Jednak użyte w nim technologie zostaną wykorzystane w innych modelach pamięci masowej, o czym się wkrótce przekonasz.

Świat rozwiązań typu SAN zawiera wiele popularnych technologii połączeń. Dostępne są niezwykle proste rozwiązania transportowe SAN, które są na tyle nieskomplikowane, że pozostają nierozpoznane przez wielu. Do tych rozwiązań zaliczamy USB, Thunderbolt

i IEEE1394/Firewire. Istnieje także szeroka gama protokołów SAN klasy przemysłowej, takich jak iSCSI (SCSI over IP), FibreChannel, FCoE (Fibre Channel over Ethernet), *FC-NVMe* (NVMe over Fiber Channel) itd. Każdy protokół SAN ma pewne wady i zalety, a producenci zwykle dostarczają jedynie niewielką liczbę wykorzystującego go sprzętu. Dlatego też wybór producenta zazwyczaj będzie oznaczał ograniczenie dostępnych opcji SAN, zaś wybór opcji SAN będzie ograniczał dostępnych producentów. Poznanie tych wszystkich protokołów oznacza przejście ze świata systemów do świata sieci. Bardzo rzadko się zdarza, że administrator systemu będzie wybierał rozwiązania typu SAN lub będzie miał wpływ na wybory dokonywane w projekcie. Z reguły wybory są dokonywane przez zespoły zajmujące się pamięcią masową, siecią i/lub platformą. Jeżeli masz wpływ na ten obszar, musisz koniecznie zapoznać się z tymi technologiami, ich zaletami oraz poznać możliwości ich zastosowania w Twoich zadaniach. Jednak omówienie tych kwestii wykracza poza zakres tematyczny tej książki.

Blokowa pamięć masowa nie odchodzi do lamusa. Wprawdzie nowe technologie dotyczące pamięci masowej, np. obiektowa pamięć masowa, są fascynujące, ale mimo to blokowa pamięć masowa zapewnia fundamenty dla wszystkich pozostałych typów pamięci masowej. Konieczne jest dokładne poznanie urządzeń blokowych zarówno pod względem fizycznym, jak i logicznym, ponieważ na wiele sposobów będą one używane jako elementy konstrukcyjne na naszych platformach pamięci masowej. Blokowa pamięć masowa oferuje potężne możliwości i jest wszechobecna. To także najważniejszy rodzaj pamięci masowej, z którą będziemy się spotykać na etapach tworzenia rozwiązania. Można się spodziewać, że w nadchodzących dekadach pozostanie ona podstawą wszystkiego, co robimy.

Istnieje użyteczna reguła, którą można wykorzystać podczas podejmowania decyzji o wyborze między lokalną a zdalną blokową pamięcią masową: *Zawsze chcesz używać lokalnej pamięci masowej, o ile nie masz potrzeb, których ona nie może spełnić. Nigdy nie chcesz używać zdalnej pamięci masowej, o ile nie masz innego wyjścia.*

Omówienie dyskowych i sieciowych systemów plików

Na warstwie powyżej blokowej pamięci masowej zwykle znajduje się **system plików**. System plików to podstawowy (używając określenia podstawowy, mam na myśli 99,999% lub więcej przypadków użycia) sposób ostatecznego przechowywania danych w systemach komputerowych. W systemach plików są przechowywane pliki, jakie znają użytkownicy, gdy pracują na komputerze.

System plików to format organizacji danych używany w blokowej pamięci masowej i zapewniający mechanizm organizacji, identyfikacji, przechowywania i pobierania

danych za pomocą analogii pliku. Systemu plików używasz każdego dnia i praktycznie do wszystkiego. Są one używane nawet wtedy, gdy ich nie widać na pulpicie, w smartfonie, w telefonie VoIP lub nawet w kuchence mikrofalowej. Systemy plików są wszędzie.

System plików to w rzeczywistości baza danych

Bądźmy szczerzy: skoro czytasz książkę dotyczącą najlepszych praktyk w zakresie administrowania systemem, to doskonale wiemy, że uwielbiasz zagłębiać się w szczegóły. Zatem przekonaj się, czym tak naprawdę jest system plików. U podstaw systemu plików znajduje się baza danych typu NoSQL, w szczególności baza danych pliku (w zasadzie to jest specjalizowana baza danych dokumentu), bezpośrednio używająca urządzenia blokowego jako mechanizmu pamięci masowej oraz posiadająca jedynie możliwości przechowywania i pobierania plików.

Istnieją także inne specjalne bazy danych bezpośrednio używające urządzeń blokowych (w żargonie świata baz danych często nazywane bezpośrednią pamięcią masową), ale są one rzadko spotykane. System plików to tak często występujący rodzaj bazy danych, znacznie częściej niż wszystkie pozostałe typy, że nikt nigdy o nim nie mówi i w ogóle nie traktuje go jako bazy danych. Jednak pod maską mamy prawdziwą bazę danych i to pod każdym względem.

Być może zastanawiasz się nad porównaniem obu wymienionych rodzajów baz danych. Standardowa baza danych niezależnie od pliku ma standaryzowany format pamięci masowej, format pobierania danych, silnik (sterownik) bazy danych oraz w pewnych przypadkach warstwę zarządzania bazą danych (to często pozwala na używanie wielu silników baz danych za pomocą interfejsu pojedynczego systemu) i interfejs zapytań pozwalający uzyskać dostęp do danych. Niezależnie od tego, czy do porównania użyjesz MongoDB, czy MS SQL Server, przekonasz się, że systemy plików działają identycznie. Wybrany system plików na dysku jest formatem pamięci masowej, formatem pobierania jest *plik*, silnik bazy danych to system pliku napędu, system zarządzania bazą danych w Linuksie jest wirtualnym systemem plików (do tego zagadnienia jeszcze powrócę), a język zapytań stanowi lista poleceń zaimplementowanych w języku C (razem z bazującymi na powłoce prostymi abstrakcjami, których można używać dla wygody). Jeżeli porównasz ją ze standardową bazą danych, trudno będzie je rozróżnić. To świetne rozwiązanie.

Po wdrożeniu systemu komputerowego niemal wszystkie przeprowadzone w nim operacje z perspektywy pamięci masowej oznaczają pracę z systemem plików. Na etapie tworzenia dużą wagę przywiązuje się do blokowej pamięci masowej, natomiast na etapie administrowania skupia się uwagę na systemach plików. Nie ulega wątpliwości, że systemy plików muszą być poprawnie zaplanowane jeszcze przed wdrożeniem systemu do środowiska produkcyjnego. Poprawne zaplanowanie systemu plików to w rzeczywistości zadanie pomijane przez większość osób, które po prostu akceptują wartości domyślne i rzadko w ogóle zastanawiają się nad projektem systemu plików.

Większość systemów operacyjnych oferuje natywną obsługę dla kilku różnych systemów plików. W większości przypadków system operacyjny ma jeden oczywisty standard i podstawowy system plików, a także specjalne systemy plików przeznaczone do użycia w niszowych urządzeniach sprzętowych bądź też w celu zapewnienia zgodności z innymi systemami. Na przykład system macOS firmy Apple używa systemu plików APFS (Apple File System) do wszystkich standardowych zadań oraz systemu plików ISO 9660 podczas pracy z nośnikami optycznymi. Ponadto zapewnia obsługę systemów plików FAT32 i **exFAT** w celu zapewnienia zgodności z urządzeniami pamięci masowej Windows (np. pendrive USB lub zewnętrzny dysk twardy). W systemie Windows mamy podobną sytuację, przy czym podstawowym systemem plików jest NTFS zamiast APFS. W systemie Windows ostatnio został dodany alternatywny system plików, ReFS, przeznaczony do celów specjalnych. Jednak nie jest powszechnie używany ani znany.

W dystrybucjach Linuksa mamy wiele podstawowych systemów plików oraz wiele różnych specjalnych systemów plików. Nie ma w tym miejscu możliwości omówienia ich wszystkich, wspomnę więc o kilku najważniejszych i wyjaśnię, dlaczego są używane i dlaczego są tak istotne. Na szczęście w systemach produkcyjnych tak naprawdę trzeba się zajmować jedynie paroma kluczowymi produktami. Jeżeli interesują Cię systemy plików, znajdziesz wiele informacji na temat różnych systemów plików dostępnych w Linuksie. Dostępnych jest wiele danych o historii projektu systemów plików i być może znajdziesz nawet projekt, którego będziesz chciał używać do specjalnych celów.

Obecnie w Linuksie istnieje kilka kluczowych i przeznaczonych do obsługi codziennych zadań systemów plików, o których warto wiedzieć: XFS, EXT4, ZFS, BtrFS. Praktycznie wszystko to, czym będziesz się zajmować, będzie wymagało użycia jednego z tych systemów plików. Ponadto istnieje wiele mniej popularnych systemów plików, które zostały doskonale zintegrowane i świetnie się sprawdzają w różnych zadaniach. Zaliczamy do nich np. JFS i ReiserFS, choć praktycznie nigdy nie spotkasz ich w środowiskach produkcyjnych. Istnieją jeszcze starsze wersje formatów takich jak EXT2 i EXT3, ale zostały one wyparte przez ich nowsze wydania. Mamy też wiele systemów plików będących standardami w innych systemach operacyjnych, a jednocześnie możliwych do użycia w Linuksie, np. NTFS z rodziny Windows i UFS z rodziny BSD. Dostępne są też niszowe systemy plików, takie jak ISO 9660 i FAT32, o których wspomniałem już wcześniej. Linux oferuje wiele opcji w zakresie wyboru systemu plików — to doskonały przykład pokazujący, jak szeroki może być wybór.

EXT — rodzina systemów plików Linuksa

Niemalże każdy system operacyjny ma własny specjalny system plików, który jest używany natywnie bądź domyślnie oraz pozostaje ściśle powiązany ze zdalnym systemem operacyjnym. Linux nie jest tutaj wyjątkiem… Żartowałem. Linux jest absolutnie wyjątkowy, co naprawdę zadziwia, jeśli się weźmie pod uwagę znacznie większą niezawodność Linuksa pod względem dostępnych opcji systemów plików w porównaniu z innymi

systemami operacyjnymi. Illumos ma ZFS, FreeBSD ma UFS, Windows ma NTFS, macOS ma APFS, AIX ma JFS, IRIX miał XFS itd. Linux naprawdę nie ma własnego systemu plików, ma za to wszystkie inne systemy plików.

Większość osób uznaje systemy plików z rodziny EXT jako natywne systemy plików Linuksa i tak naprawdę nic innego nie przychodzi na myśl, co pasowałoby do tego opisu. Podczas opracowywania Linuksa, jeszcze na długo przed tym, gdy ktokolwiek miał okazję go wypróbować, został do niego przeniesiony system plików MINIX i stał się on domyślnym systemem plików w czasie powstawania nowego systemu operacyjnego. Jednak jak sama nazwa wskazuje, system plików MINIX był natywny dla systemu MINIX i powstał jeszcze na długo przed Linuksem.

Zaledwie na rok przed pierwszą premierą Linuksa został opracowany system plików EXT (skrót od MINIX Extended File System) na bazie istniejącego systemu plików MINIX. Jak nietrudno zgadnąć, ten system plików zawierał nowe funkcjonalności, przede wszystkim związane ze znacznikami czasu.

Wraz z rozwojem systemu Linux usprawniany był również system plików EXT. Zaledwie rok po wprowadzeniu EXT pojawił się jego następca, EXT2, który okazał się ogromnym usprawnieniem. Dzięki niemu możliwe stało się przekształcenie ekosystemu systemu plików Linuksa z kategorii hobby na prawdziwie korporacyjny. EXT2 praktycznie na wyłączność dominował w ekosystemie Linuksa od chwili jego wprowadzenia w 1993 r. do 2001 r., gdy Linux przeszedł rewolucję systemu plików. System plików EXT2 okazał się tak dużym krokiem naprzód, że został przeniesiony również do systemu MINIX i zaczął być obsługiwany także w innych systemach operacyjnych np. Windows i macOS. Prawdopodobnie żaden inny system plików nie jest tak *silnie identyfikowany* z Linuksem jak EXT2.

Do 2001 r. w wielu systemach operacyjnych szukano bardziej zaawansowanych technologii systemów plików, aby zapewnić im większą konkurencyjność na rynku. Sytuacja wyglądała podobnie również w ekosystemie Linuksa, w którym wprowadzono obsługę kolejnych systemów plików oraz dodano funkcjonalność księgowania do istniejącego systemu plików EXT2 i zmieniono jego nazwę na EXT3. Dzięki temu rodzinie systemów plików EXT zapewniono tak bardzo potrzebną jej stabilność.

Po kolejnych siedmiu latach pojawiło się następne ważne uaktualnienie pseudonatywnego systemu plików Linuksa — EXT4. Co zaskakujące, główny programista systemów plików EXT3 i EXT4 stwierdził, że wprawdzie EXT4 to ogromny krok naprzód, ale praktycznie mamy do czynienia z prowizorycznym dodaniem usprawnień, które w dużej mierze są technologią pochodzącą z lat 80. ubiegłego wieku. Reguły projektowe systemów plików znacznie się zmieniły na początku naszego wieku, a rodzina systemów plików EXT prawdopodobnie znajduje się na końcu stawki. Mimo to wciąż ma przed sobą przyszłość.

Zamierzam dość dokładnie omówić najważniejsze systemy plików w Linuksie. Musisz jednak pamiętać, że te informacje mimo wszystko nie będą aż tak szczegółowe. Szczegóły dotyczące systemu plików mogą się dość często zmieniać i różnić się między poszczególnymi wersjami lub implementacjami. Dlatego też jeśli szukasz konkretnych informacji np. o maksymalnej wielkości pliku, dozwolonej liczbie plików, wielkości systemu plików itd., to sięgnij do artykułów Wikipedii bądź dokumentacji systemu plików. Naprawdę nie musisz zapamiętywać tych szczegółów i rzadko kiedy będziesz do nich wracać. W latach 90. ubiegłego wieku ograniczenia systemu plików były dość rygorystyczne, więc trzeba było je znać, aby można było sobie z nimi poradzić. Obecnie praktycznie każdy system plików używany w najpopularniejszych systemach operacyjnych jest w stanie obsłużyć niemalże dowolne zadanie. Dlatego tak naprawdę należy poznać mocne i słabe strony poszczególnych systemów plików, aby wiedzieć, który z nich powinien zostać użyty w danym rozwiązaniu.

EXT4

Linux, jako kategoria, nie ma domyślnego systemu plików w kontekście znanym z innych systemów operacyjnych. Jeżeli spróbujesz ustalić, który z obecnie dostępnych systemów plików zasługuje na to miano, prawdopodobnie będzie nim **EXT4**. Większość wdrożeń Linuksa obecnie używa EXT4 jako domyślnego systemu plików. Jednak to zaczyna się zmieniać, więc jest mało prawdopodobne, że EXT4 pozostanie dominującym systemem plików przez więcej niż kilka kolejnych lat.

EXT4 to wystarczająco szybki i niezawodny system plików. Do tego całkiem elastyczny, doskonale znany i spełniający wymagania niemalże każdego wdrożenia. Można go uznać za punkt wyjścia podczas wyboru systemu plików dla Linuksa. W przypadku typowego wdrożenia systemu plików EXT4 sprawdza się doskonale.

EXT4 jest określany mianem *czystego systemu plików*, co oznacza, że jest po prostu systemem plików i niczym więcej. To znacznie ułatwia zrozumienie jego sposobu działania i użycia, ponieważ ten system plików ma również znacznie ograniczone możliwości.

XFS

Podobnie jak rodzina EXT, także system plików **XFS** powstał we wczesnych latach 90. ubiegłego wieku i pochodzi od konkurenta Linuksa, w tym przypadku systemu operacyjnego IRIX UNIX opracowanego przez SGI. Jest na tyle solidny i niezawodny, że został przeniesiony do Linuksa w 2001 r., czyli kiedy pojawił się EXT3. Przez 20 lat systemy plików EXT3/4 i XFS próbowały zdobyć serca i dusze administratorów systemu Linux oraz skłonić twórców dystrybucji Linuksa, aby stały się domyślnymi systemami plików.

XFS to również *czysty system plików* i jest powszechnie używany. XFS słynie z wyjątkowej wydajności działania i niezawodności. XFS jest czasami zalecany w aplikacjach wymagających wysokiej wydajności działania, np. w bazach danych, aby zapewnić im maksymalną wydajność.

XFS to prawdopodobnie najczęściej wdrażany system plików, gdy administrator systemu ma wpływ na wybór systemu plików, a nie decyduje się na użycie rozwiązania domyślnego. Ponadto jest prawdopodobnie najczęściej zalecanym systemem plików przez producentów aplikacji, a także moim prywatnym faworytem dla większości rozwiązań, gdy wymagania dotyczące pamięci masowej są nieco bardziej zaawansowane.

Na przestrzeni lat systemy plików EXT4 i XFS zdobywały i traciły popularność. Z moich obserwacji wynika, że to XFS powoli zyskuje przewagę.

Dość często pojawiającym się zastrzeżeniem wobec systemu plików XFS w porównaniu z EXT4 jest to, że EXT4 pozwala na zwiększanie *lub* zmniejszanie woluminu po jego utworzeniu. Z kolei XFS pozwala jedynie na zwiększenie woluminu, ale już nie na zmniejszenie istniejącego. Jednakże praktycznie nie spotyka się informacji o tym, że poprawnie wdrożony system produkcyjny wymaga zmniejszenia wielkości systemu plików. Dlatego też to zastrzeżenie wydaje się naciągane i nieistotne podczas podejmowania decyzji dotyczącej wyboru systemu plików, zwłaszcza wraz z nadejściem elastycznej blokowej pamięci masowej.

ZFS

Opracowany w 2006 r. dla systemu Solaris system plików **ZFS** jest, ogólnie rzecz biorąc, uznawany za fundament dla prawdziwie nowoczesnego projektu systemu plików. Gdy w 2001 r. rozpoczęto prace nad ZFS, w branży informatycznej zaczęto na poważnie podchodzić do projektu systemu plików. Regularnie pojawiało się wiele nowych związanych z tym koncepcji, ale to w systemie plików ZFS tak naprawdę wykorzystano te paradygmaty projektowe i przeniesiono je na wyższy poziom. Obecnie system plików ZFS nadal pozostaje liderem na wielu obszarach.

ZFS ma trzy wysokiego poziomu obszary, na których przeprowadził rewolucje branży systemów plików. Pierwszy to wielkość: ZFS może zaadresować znacznie większą pojemność pamięci masowej niż jakikolwiek inny system plików wcześniej. Drugi to niezawodność: ZFS wprowadził znacznie bardziej niezawodne mechanizmy ochrony danych niż stosowane w innych systemach plików. Ponadto pozwala skutecznie chronić przed utratą danych. Trzeci to integracja: ZFS to pierwszy prawdziwie **nieczysty system plików**: ZFS przedstawia system plików, system macierzy RAID oraz zarządcy woluminów logicznych — ta cała funkcjonalność została wbudowana w pojedynczy sterownik systemu plików. Dokładniejsze omówienie macierzy RAID i zarządcę woluminów logicznych przedstawię w dalszej części rozdziału. Ta integracja okazała się znacząca,

ponieważ pozwoliła warstwie pamięci masowej na prowadzenie komunikacji i koordynacji jak nigdy wcześniej. Czyste systemy plików takie jak EXT4 i XFS mogą wykorzystywać — i to robią — tę technologię, ale odbywa się to za pomocą komponentów zewnętrznych, a nie zintegrowanych.

Wprawdzie ZFS nie jest nowością i pojawiał się w systemach produkcyjnych przez ostatnie 15 lat, mimo to jest nowością w systemie Linux. Minęły lata, zanim przygotowano wersję ZFS dla Linuksa. Następnie musiały minąć kolejne lata, w trakcie których obawy dotyczące licencji uniemożliwiały udostępnienie tego formatu w postaci użytecznej dla systemu Linux. Obecnie jedną z najważniejszych dystrybucji Linuksa oficjalnie obsługującą i udostępniającą system plików ZFS jest Ubuntu. Należy pamiętać, że dominująca pozycja Ubuntu na rynku oznacza również automatycznie powszechne udostępnienie systemu plików ZFS. W chwili powstawania książki nie minęły jeszcze dwa lata, odkąd system plików ZFS może być używany jako rozruchowy główny system plików w Ubuntu. Dlatego też można go uznać za nowość w produkcyjnych i rozpowszechnionych środowiskach Linuksa. Popularność tego systemu plików w Linuksie gwałtownie rośnie.

ZFS to prawdopodobnie najbardziej zaawansowany, niezawodny skalowalny system plików dostępnych obecnie w Linuksie. Warto w tym miejscu dodać, że z perspektywy czystej wydajności działania systemu plików ZFS nie jest orłem. Rzadko się zdarza, aby przeprowadzone na poziomie systemu plików usprawnienia związane z wydajnością działania były uznawane za cenne. Jednak nowoczesne systemy plików charakteryzujące się wyjątkową niezawodnością zwykle nie mogą konkurować ze starszymi i znacznie prostszymi systemami plików. Często przyjmuje się założenie, że znacznie nowsze systemy automatycznie będą również szybsze, ale w omawianym przypadku tak nie jest.

BtrFS

System plików **BtrFS** to wynik obecnie podejmowanej próby utworzenia natywnego systemu plików dla Linuksa. (Wcześniejsza próba tego rodzaju dotyczyła systemu plików ReiserFS i zyskała rozgłos na początku obecnego wieku, ale z powodów technicznych okazała się nieudana). System plików BtrFS ma odzwierciedlać sposób działania systemu plików ZFS, ale jednocześnie ma być rozwiązaniem natywnym dla Linuksa i dostępnym na odpowiedniej licencji.

BtrFS znacznie wyprzedza ZFS, choć wiele funkcjonalności nie zostało jeszcze zaimplementowanych. Jednak prace nad BtrFS wciąż trwają. Projekt systemu plików jest aktywny i obsługiwany przez coraz więcej dystrybucji Linuksa. Zdarza się również, że jest wybierany jako domyślny system plików. Można odnieść wrażenie, że BtrFS to prawdopodobnie długoterminowa przyszłość dla Linuksa.

Podobnie jak ZFS, także BtrFS to nowoczesny i wysoce zintegrowany system plików, który zaczyna oferować funkcjonalność znaną z macierzy RAID i zarządcy woluminów logicznych. Obecnie najsłabszym aspektem systemu plików BtrFS jest wydajność działania.

Stratis

Biorąc pod uwagę obsługę w branży informatycznej, trzeba w tym miejscu wspomnieć o Stratis. To nie jest *typowy* system plików, ale rozwiązanie bardzo do niego podobne. Stratis to wynik podjętej próby opracowania zintegrowanej funkcjonalności (czyli *system plików zarządzania woluminami*) podobnej do znanej z systemów plików ZFS i BtrFS. Stratis używa do tego istniejących komponentów systemu plików XFS oraz standardowej warstwy zarządcy woluminów logicznych (LVM) w Linuksie.

Na początkowym etapie istnienia systemu operacyjnego IRIX system plików XFS został opracowany do użycia razem z natywnym zarządcą woluminów logicznych. Te dwa rozwiązania zostały w naturalny sposób zintegrowane, dostarczając w ten sposób coś przypominającego istniejące obecnie systemy plików ZFS i BtrFS. Po przeniesieniu do Linuksa systemu plików XFS powiązana z nim warstwa LVM nie została przeniesiona, a zamiast tego opracowano natywną dla Linuksa warstwę zarządcy woluminów logicznych. Wobec tego XFS + LVM od dawna jest standardem przemysłowym, zaś Stratis to jedynie próba dostarczenia znacznie wygodniejszego rozwiązania przy jednoczesnej integracji najlepszych praktyk i uproszczonego sposobu zarządzania.

W ten sposób przedstawiłem cztery obecnie stosowane w środowisku produkcyjnym systemy plików, które prawdopodobnie napotkasz bądź spośród których będziesz dokonywać wyboru. Pamiętaj o możliwości łączenia różnych systemów plików w pojedynczym rozwiązaniu. To jest bardzo często stosowane podejście. W rzeczywistości można się spotkać z rozwiązaniami, w których EXT4 jest używany jako rozruchowy system plików do zapewnienia podstawowej funkcjonalności systemu operacyjnego. Z kolei XFS służy jako wysokowydajny system plików pamięci masowej przeznaczonej dla bazy danych, zaś BtrFS jako system plików dla ogromnego serwera plików. Wybieraj to, co ma sens dla danego zadania na warstwie pojedynczego systemu plików. Nie myśl, że musisz używać tylko jednego systemu plików w całym rozwiązaniu bądź też w jednym komputerze.

Większość prawdziwie technicznych aspektów systemu plików kryje się w algorytmach odpowiedzialnych za obsługę wyszukiwania i przechowywania bitów w urządzeniach blokowych. Omawianie szczegółów związanych z tymi algorytmami wykracza poza zakres tematyczny nie tylko tej książki, ale również ogólnie poza administrowanie systemami. Jeżeli interesujesz się systemami plików, wówczas poznanie sposobu przechowywania danych, ich ochrony i pobierania z dysku może być prawdziwie fascynujące. Natomiast w przypadku zadań związanych z administrowaniem systemami w zupełności wystarczająca jest ogólna wiedza dotycząca systemów plików.

Niestety nie istnieje dobry sposób na przekazanie najlepszych praktyk związanych z wyborem systemu plików. Jest mało prawdopodobne, że w środowisku produkcyjnym trzeba będzie na poważnie rozważyć użycie innego systemu plików niż jeden z tutaj omówionych. Wszystkie cztery przedstawione wcześniej są niezwykle dobrymi systemami plików i należy brać je pod uwagę. Bardzo często wybór systemu plików nie odbywa się w całkowitej izolacji, o ile nie pracujesz nad bardzo specyficznym produktem, w przypadku którego wymaga się (bądź zaleca) użycie konkretnego systemu plików z powodu określonej funkcjonalności produktu. Zamiast tego wybór systemu plików zwykle wiąże się z wieloma innymi decyzjami dotyczącymi pamięci masowej, m.in. macierzy RAID, LVM, potrzeb w zakresie fizycznej obsługi urządzeń, nośników itd.

Klastrowane systemy plików

Wszystkie omówione dotychczas systemy plików, niezależnie od tego, jak bardzo są one nowoczesne, zaliczają się do *standardowych* bądź *niestandardowych* systemów plików. Ich stosowanie przynosi korzyści jedynie wtedy, gdy dostęp odbywa się z poziomu tylko jednego systemu operacyjnego. W prawie wszystkich przypadkach takie rozwiązanie sprawdza się doskonale.

Przypomnij sobie wcześniejszą dyskusję dotyczącą rozwiązania typu SAN. Wspomniałem wówczas o przypadkach, w których jednocześnie wiele komputerów może odczytywać i zapisywać dane w tym samym systemie pamięci masowej. Mechanizmem pozwalającym na działanie takiego rozwiązania jest system klastrowany, czyli *system plików współdzielonej pamięci masowej*.

Klastrowany system plików działa podobnie jak tradycyjny system plików, ale oferuje dodatkową funkcjonalność pozwalającą na blokowanie podczas operacji zapisu i współdzielenie informacji systemu plików, aby wiele komputerów mogło koordynować użycie tego systemu plików. W przypadku standardowego systemu plików dostęp do niego uzyskuje tylko jeden komputer w danym momencie. Dlatego też system tego komputera wie, który plik jest otwarty, który jest uaktualniony, kiedy zapis jest buforowany itd. Te wszystkie operacje są obsługiwane w pamięci. Jeżeli z dwóch lub więcej komputerów spróbuję współdzielić dane znajdujące się w tradycyjnym systemie plików, komputery nie będą mogły współdzielić tych danych pamięci, co nieuchronnie doprowadzi do uszkodzenia danych, ponieważ będą one wzajemnie nadpisywały wprowadzone przez siebie zmiany. Tak się dzieje z powodu braku możliwości wykrycia uaktualnienia pliku. Z kolei nieaktualny bufor zapisu może prowadzić do wielu niepożądanych efektów ubocznych.

Skoro jedynym komponentem współdzielonym tych systemów jest system plików, cała komunikacja między węzłami dotycząca dostępu do systemu plików musi się odbywać w samym systemie plików. Dlatego też dosłownie nie ma innej możliwości niż wykorzystanie mechanizmu, który nie jest po prostu współdzieloną pamięcią masową,

ale współdzieloną jednostką obliczeniową wyposażoną w pamięć masową, a takie rozwiązanie jest znacznie bardziej skomplikowane i kosztowne.

Oto jak najprościej można opisać sposób działania klastrowego systemu plików: każdy komputer otrzymuje z systemu plików informacje, że określony fragment urządzenia blokowego (dyski) w niezwykle sztywnym formacie i o podanej wielkości jest zarezerwowany jako obszar, w którym węzły odczytują i zapisują bieżące informacje o stanie pracy z systemem plików. Jeżeli węzeł A musi otworzyć plik X, umieszcza informacje o otwarciu tego pliku. Jeżeli węzeł B usuwa plik, podaje informację o zamiarze jego usunięcia i uaktualnia ją po faktycznym usunięciu pliku. Odczytując te niewielkie informacje z systemu plików, węzeł C ma aktualne dane o bieżącej aktywności. Wszystkie połączone węzły wiedzą, aby nie buforować danych na tym obszarze i informować o wszelkich podejmowanych akcjach, a także rejestrować informacje o przeprowadzonych działaniach. Jeżeli którykolwiek węzeł będzie działał nieprawidłowo, cały system ulegnie awarii i dojdzie do utraty danych.

Oczywiście nie ulega wątpliwości, że takie rozwiązanie wiąże się przynajmniej ze znacznym spadkiem wydajności działania. Poza tym system wymaga absolutnego zaufania między wszystkimi powiązanymi ze sobą węzłami, ponieważ kontrola dostępu do danych i spójności danych odbywa się w poszczególnych węzłach. Nie ma i nie może być żadnego mechanizmu wymuszającego prawidłowe działanie węzłów. Węzły same muszą nad tym czuwać. To oznacza, że jakikolwiek błąd w kodzie, awaria pamięci, administrator z uprawnieniami root i oprogramowanie typu malware zyskujące dostęp do pojedynczego węzła będą miały dostęp również do pozostałych węzłów i będą mogły pominąć *wszelkie* mechanizmy kontroli. W efekcie zyskają możliwość odczytu, modyfikacji, uszkodzenia, szyfrowania itd. w dowolnym stopniu, a wszelkie mechanizmy bezpieczeństwa i kontroli, które powinny zapewnić ochronę, nie będą istniały. Współdzielona pamięć masowa jest wyjątkowo prosta. Jednak przywykliśmy do traktowania abstrakcji pamięci masowej jako skomplikowanej, więc aż trudno uwierzyć, że jakikolwiek system pamięci masowej może być tak prosty.

Podobnie jak w przypadku zwykłych systemów plików, Linux ma wiele klastrowanych systemów plików, które są powszechnie używane. Do najczęściej spotykanych zaliczamy GFS2 i OCFS2.

Ponadto podobnie jak ogólnie w rozwiązaniach typu SAN, ta sama reguła ma zastosowanie również dla klastrowanego systemu plików — nie należy go używać, o ile nie ma takiej konieczności.

Sieciowe systemy plików

Sieciowy system plików zawsze jest trudniejszy do opisania, ale przynosi korzyści dzięki fenomenowi *będziesz o tym wiedzieć, gdy go zobaczysz*. W przeciwieństwie do zwykłego systemu plików działającego na bazie urządzenia blokowego i zapewniającego

dostęp do pamięci masowej w postaci plików, sieciowy system plików wykorzystuje zwykły system plików i udostępnia go poprzez sieć. Wprawdzie to może przypominać rozwiązanie typu SAN, ale jest zupełnie inne. W przypadku rozwiązania SAN mamy współdzielenie zbioru urządzeń blokowych poprzez sieć. Natomiast sieciowy system plików współdzieli zwykły system plików poprzez sieć.

Sieciowe systemy plików są dość często stosowane i prawdopodobnie spotykasz się z nimi każdego dnia, ale często nawet o tym nie wiesz. Można się zetknąć z określaniem sieciowych systemów plików mianem *udziałów* lub *mapowanych napędów*. Do standardowych protokołów używanych przez sieciowe systemy plików zaliczamy NFS i SMB (czasem można napotkać określenie CIFS, które jednak jest niewłaściwe). Serwer implementujący sieciowy system plików jest nazywany serwerem plików, jeśli wbudujesz go w urządzenie, wówczas otrzymujesz tzw. NAS (ang. *network-attached storage*). Z tego powodu sieciowe systemy plików są często traktowane jako *protokoły NAS*, podobnie jak rozwiązanie blokowe przekazywane przez protokół sieciowy jest traktowane jako *protokół SAN*.

W przeciwieństwie do protokołu współdzielonych bloków sieciowy system plików działa *sprytnie*, ponieważ urządzenie udostępniające pamięć masową ma lokalny system plików, który zapewnia obsługę takich koncepcji jak nakładanie blokad, buforowanie, uaktualnianie plików itd. Mogą być one obsługiwane za pomocą pojedynczego komponentu wymuszającego zapewnienie bezpieczeństwa i spójności danych, bez konieczności ufania węzłom uzyskującym dostęp do danych. Największa różnica polega na tym, że SAN to po prostu pamięć masowa dołączona do sieci. Może mieć bardzo prostą postać, np. karty sieciowej wbudowanej w dysk twardy (i często tak się zdarza). Z drugiej strony urządzenie implementujące sieciowy system plików jest serwerem, który do funkcjonowania wymaga procesora, pamięci i systemu operacyjnego. Współdzielona blokowa pamięć masowa jest używana niemal wyłącznie w bardzo ograniczonych wdrożeniach, z wykorzystaniem niezwykle dokładnie kontrolowanych serwerów. Sieciowego systemu plików można używać praktycznie wszędzie tam, gdzie zastosowanie znajduje rozwiązanie typu SAN. Ponadto używa się go powszechnie do udostępniania pamięci masowej bezpośrednio urządzeniom użytkownika końcowego, ponieważ duży poziom bezpieczeństwa, łatwość użycia i brak konieczności ufania urządzeniu docelowemu powodują, że takie rozwiązanie jest niezwykle użyteczne w sytuacjach, w których niemożliwe byłoby wdrożenie rozwiązania typu SAN.

Sieciowy system plików działa jako dodatkowa warstwa sieciowa na bazie tradycyjnego systemu plików i nie zastępuje *dyskowego systemu plików* istniejącego w pamięci masowej. Z perspektywy interfejsu, sieciowy system plików można przedstawić jako używający *interfejsu systemu plików* i dostarczający *interfejs systemu plików*. W zasadzie mamy więc system plików na wejściu i system plików na wyjściu.

Podobnie jak w przypadku tradycyjnego systemu plików, Linux oferuje duży wybór sieciowych systemów plików. Wiele z nich jest historycznych z natury lub wyjątkowo

niszowych. Dobrym przykładem jest tutaj **Apple Filing Protocol**, czyli **AFP** (inaczej *AppleTalk*), oferowany przez Linuksa, ale obecnie nieużywany w żadnym systemie produkcyjnym. Dziś faktycznie używanymi sieciowymi systemami plików są NFS i SMB.

NFS

Pierwotny **sieciowy system plików** (ang. *network file system*, NFS) będący w powszechnym użyciu i od którego dosłownie pochodzi nazwa protokołu, **NFS**, powstał w 1984 r.! NFS nie może natywnie działać w Linuksie, ponieważ jest od niego starszy o siedem lat. Mimo to NFS był domyślnym sieciowym systemem plików we wszystkich systemach operacyjnych z rodziny UNIX/Linux od chwili jego pojawienia się i stanowi raczej ważny standard. Linux jest obecnie ważnym systemem operacyjnym i dlatego większość osób uważa, że NFS to *protokół Linuksa*.

NFS jest dostępny praktycznie w każdym systemie. Jego obsługa jest oferowana przez każdy system mający swoje korzenie w systemie UNIX, nawet macOS, a także Windows Server! NFS to standard otwarty i prawie uniwersalny. Popularność NFS-u wynika z prostoty jego użycia, niezawodnego działania w sieci i ogólnie dobrej wydajności działania. Protokół NFS pozostaje często używany w serwerach, gdy wymagane jest bezpośrednie współdzielenie plików między komputerami, zwłaszcza w zadaniach związanych z tworzeniem kopii zapasowej.

SMB

Protokół **SMB** (ang. *server message block*) jest jeszcze starszy od NFS-u i pojawił się w 1983 r. Oba tutaj omawiane protokoły są faktycznie bardzo starą technologią. Ten protokół nie zdobył większej popularności aż do chwili, gdy firma Microsoft zaczęła go promować mniej więcej w 1990 r. Wraz z pojawieniem się platformy Windows NT protokół SMB zaczął zyskiwać na popularności.

Protokół SMB stał się znany dzięki używaniu go przez firmę Microsoft do mapowania napędów między serwerami i stacjami roboczymi. W ten sposób protokół SMB poznało wielu użytkowników, zarówno tradycyjnych, jak i technicznych.

W systemie Linux obsługę protokołu SMB zapewnia pakiet Samba (nazwa tego pakietu to żart na podstawie liter SMB). Wprawdzie Linux zapewnia dobrą obsługę protokołu SMB, ale jego używanie jest znacznie bardziej skomplikowane niż praca z protokołem NFS.

Wybór między protokołami NFS i SMB na potrzeby współdzielenia plików w systemie Linux zależy, ogólnie rzecz biorąc, od konkretnej sytuacji. Jeżeli w rozwiązaniu dominują systemy z rodziny UNIX, wówczas sensowniejsze będzie użycie protokołu NFS. Natomiast jeśli dominującymi systemami są Windows, wówczas lepiej będzie użyć protokołu SMB. Oba protokołu oferują potężne możliwości, są niezawodne i mogą służyć do wielu zadań.

Podjęcie decyzji może się okazać wyjątkowo trudne, gdy te same potrzeby można spełnić na zupełnie odmienne sposoby. Na przykład jeśli trzeba zapewnić współdzieloną pamięć masową dla wirtualizacji, wówczas można skorzystać z sieciowego systemu plików takiego jak NFS lub klastrowanego systemu plików (np. GFS2) w rozwiązaniu typu SAN. Te dwa podejścia nie mogą być łatwo porównane, ponieważ poszczególne aspekty obu systemów prawdopodobnie będą odmienne. Dotyczy to także np. producentów i sprzętu, więc porównanie trzeba przeprowadzić na poziomie pełnego stosu, a nie jedynie na poziomie technologii sieciowej.

W ten sposób przedstawiłem wybrane technologie systemów plików oraz przypadki ich rzeczywistego zastosowania w systemach Linux. Wyjaśniłem również, że system plików może: być lokalny lub zdalny, zapewniać pojedynczy dostęp bądź też być klastrowany i pozwalać jednocześnie na dostęp wielu klientom. Wspomniałem także o kwestiach, które należy wziąć pod uwagę podczas wyboru systemu plików i jego konfiguracji. Dzięki temu wiesz, kiedy wybierać poszczególne technologie systemów plików, a także na co zwracać uwagę w nowych bądź alternatywnych systemach, które nie zostały tutaj dokładnie omówione. Systemy plików nie muszą być przerażające lub dezorientujące, to raczej cenne narzędzia w naszym arsenale i można je wykorzystać w celu dostosowania systemów do własnych potrzeb pod kątem zapewnienia bezpieczeństwa, skalowalności, dostępu lub wydajności działania. W następnym podrozdziale przejdę do omówienia jednego z najmniej zrozumiałych obszarów pamięci masowej — **woluminu logicznego**.

Poznanie zarządcy woluminów logicznych (LVM)

Nie znoszę używania pojęcia *nowy* względem technologii, która pozostaje w użyciu od późnych lat 80. ubiegłego wieku. Mimo to w porównaniu z większością koncepcji w dziedzinie komputerowej pamięci masowej **zarządca woluminów logicznych** (ang. *logical volume management*, LVM) to dla większości administratorów systemów całkiem nowa koncepcja, znacznie mniej znana niż większość innych standardowych technologii. Była ona używana jedynie w najwyższej klasy systemach serwerowych, dopóki nie została spopularyzowana przez Linuksa w 1998 r. i następnie przez Microsoft w 2000 r. Obecnie zarządca woluminów logicznych jest wszechobecny i powszechnie dostępny, często natywnie i domyślnie w większości systemów operacyjnych.

LVM to obecnie podstawowa technologia używana do wirtualizacji pamięci masowej. LVM pozwala wykorzystać dowolną liczbę urządzeń blokowych (czyli co najmniej jedno, na ogół są one nazywane **woluminami fizycznymi**), połączyć je, podzielić bądź zmodyfikować w jeszcze inny sposób, a następnie przedstawić w systemie jako dowolną

liczbę urządzeń blokowych (nazywanych **woluminami logicznymi**). Wprawdzie to może wydawać się skomplikowane, ale w rzeczywistości takie nie jest. Praktyczny przykład pokaże, jak łatwo można używać takiego rozwiązania.

Rozważmy przypadek komputera zawierającego trzy dyski twarde. Wszystkie dyski mogą być takie same bądź też różne. W rzeczywistości jeden może być tradycyjnym mechanicznym dyskiem twardym, jeden nowoczesnym napędem SSD, a jeden zewnętrznym napędem USB (np. macierz RAID, SAN itd.). Te wszystkie dyski mogą zostać użyte podczas tworzenia LVM i woluminu fizycznego. Trzeba w tym miejscu dodać, że LVM pozwala traktować to jako pojedynczą pulę pamięci masowej i skonfigurować ją w dowolny sposób. Jedną z możliwości jest tworzenie pojedynczego woluminu logicznego poprzez połączenie w jedną całość wszystkich trzech dysków. Ewentualnie można utworzyć wiele oddzielnych woluminów logicznych i wykorzystać je w odmiennych celach. Można mieć dowolną liczbę woluminów fizycznych i na ich podstawie otworzyć dowolną liczbę woluminów logicznych. Wielkość woluminów logicznych może być dowolna (w niektórych przypadkach nawet większa niż całkowita wielkość woluminów fizycznych!). Nie ma ograniczenia do tradycyjnej pojemności dysków twardych. W przypadku woluminu logicznego często użyteczne jest zwiększanie bądź zmniejszanie jego wielkości, co pozwala na większą elastyczność w zakresie zarządzania i izolacji.

Dzięki LVM można myśleć o systemie używającym urządzenia blokowego oraz prezentującym urządzenie blokowe. Skoro zarządcy woluminów logicznych używają urządzeń blokowych (czyli dysków)i dostarczają je, można je *łączyć* ze sobą, jeśli zachodzi potrzeba. Dlatego też urządzenie blokowe zawierające LVM i tworzące wolumin logiczny może być używane przez inny LVM tworzący kolejny wolumin logiczny, który z kolei jest wykorzystywany przez jeszcze inny LVM itd. To nie ma praktycznego zastosowania, ale pomaga wyjaśnić, jak LVM znajduje się na **stosie pamięci masowej**. Zawsze znajduje się gdzieś w środku, ale zapewnia ogromną elastyczność.

LVM musi oferować jedynie podstawową funkcjonalność *blok na wejściu*, *blok na wyjściu*. Jednak istnieją jeszcze inne funkcjonalności powszechnie dodawane do LVM, dzięki którym otrzymujemy niewiarygodnie użyteczną technologię. Do wybranych z najbardziej standardowych funkcjonalności oczekiwanych od LVM zaliczamy niepowodującą utraty danych zmianę wielkości woluminu logicznego, możliwość podłączenia urządzeń fizycznych *w trakcie działania komputera*, obsługę migawek, obsługę buforowania oraz procesu *thin provisioning*.

Po stronie Linuksa podobnie jak w innych przypadkach także w tym mamy wiele rozwiązań w zakresie obsługi zarządców woluminów logicznych. Stały się one powszechniejsze, ponieważ w ostatnich latach trendem jest tworzenie samodzielnych zintegrowanych systemów plików. W środowiskach produkcyjnych Linuksa mamy obecnie LVM2, ZFS i BtrFS. Oczywiście już wiesz, że dwa ostatnie to systemy plików, które zostały

omówione nieco wcześniej w rozdziale. Gdy większość osób mówi o zarządcy woluminów logicznych w Linuksie, zwykle ma na myśli LVM2, ogólnie rzecz biorąc, określany mianem LVM. Jednak coraz większą popularność zyskuje zintegrowana funkcjonalność LVM w systemach plików ZFS i BtrFS.

Z powodu natury LVM pozwalającego *łączyć ze sobą w stos* rozwiązania LVM dostarczające i wykorzystujące urządzenia blokowe mamy możliwość użycia LVM2 w połączeniu z systemami plików ZFS lub BtrFS oraz wyłączenia ich zintegrowanych warstw LVM jako niepotrzebnych bądź też użycie ich jako funkcjonalności przeznaczonej do wykorzystania. Pomyśl, jaka to ogromna elastyczność.

Co się stało z partycjami?

Jeżeli przypomnisz sobie pracę w dziale informatycznym w latach 90. ubiegłego wieku, wiesz, że wówczas partycje dyskowe były dość często używane. Trzeba było zdecydować o sposobie zdefiniowania partycji, liczbie tworzonych partycji, pamiętać o partycjach podstawowych i rozszerzonych, dobrać oprogramowanie przeznaczone do partycjonowania itd. Nie ulega wątpliwości, że partycje wciąż istnieją. Po prostu nie musieliśmy się nimi zajmować od bardzo dawna (od czasu systemu Windows 2000 i jądra Linuksa w wersji 2.4).

Partycje to bardzo *nieelastyczny* system dzielenia dysku fizycznego na oddzielne części, z których każda może być przedstawiona w systemie operacyjnym jako odmienne urządzenie blokowe (inaczej dysk). Pod tym względem partycje można porównać do niezwykle prostego rozwiązania typu LVM, ale pozbawionego elastyczności. Partycje są ograniczone i istnieją jako część pojedynczego urządzenia blokowego. Mapowanie określające, która część urządzenia blokowego należy do danej partycji, jest umieszczone w prostej tablicy partycji znajdującej się na początku urządzenia.

Partycje były prekursorami woluminów logicznych i nadal są wykorzystywane przez niektórych użytkowników, ale tylko dlatego, że nie znają oni woluminów logicznych. Partycje są nieelastyczne i nie zapewniają obsługi ważnych funkcjonalności takich jak *thin provisioning* i migawki, które z kolei są oferowane przez woluminy logiczne. Ponadto wprawdzie zmiana wielkości partycji jest, formalnie rzecz biorąc, możliwa, ale taka operacja okazuje się nieelastyczna, trudna i wyjątkowo ryzykowna.

LVM oferuje wszystko to, co partycje, a nawet jeszcze więcej, nie zabierając nic w zamian. Przez lata znacznie zmniejszyła się potrzeba partycjonowania, czyli tworzenia wielu systemów plików w urządzeniu blokowym. W późnych latach 90. ubiegłego wieku partycjonowanie było standardem. Okazywało się konieczne nawet w przypadku najprostszego serwera. Także podczas konfiguracji komputera biurkowego istniały dobre powody do podzielenia dysku na oddzielne systemy plików. Obecnie znacznie częściej łączy się wiele urządzeń blokowych w pojedynczy system plików. To wynika przede

wszystkim z tego, że wydajność działania systemów plików i ich niezawodność uległy całkowitej zmianie. Zaniknęły więc najważniejsze czynniki skłaniające do partycjonowania. Mimo to obecnie wciąż istnieją dobre powody ku dzieleniu systemów plików. Jednak taka potrzeba zdarza się zdecydowanie coraz rzadziej.

Wiele obecnie dostępnych mechanizmów, np. narzędzi przeznaczonych do tworzenia kopii zapasowych, wykorzystuje potężne możliwości warstwy LVM w celu przeprowadzenia zadań takich jak zamrożenie stanu urządzenia blokowego, co pozwala na wykonanie jego pełnej kopii zapasowej. Ponieważ LVM działa poniżej ostatecznej warstwy systemu plików, ma pewne możliwości, których nie znajdziemy na innych warstwach. LVM to warstwa pamięci masowej, na której otrzymujemy funkcjonalność pozbawioną znaczenia krytycznego, choć jednocześnie to warstwa, na której wykonywane są najważniejsze operacje. LVM zapewnia elastyczność w zakresie modyfikowania układu pamięci masowej po jej wdrożeniu, a także pozwala na współpracę z tymi systemami pamięci masowej na poziomie bloku. LVM można uznać za ważny komponent technologiczny, dzięki któremu mamy poczucie pracy z nowoczesnym systemem operacyjnym XXI w.

Oczywiście każda nowa warstwa technologiczna ma swoje ograniczenia. LVM prowadzi do powstania kolejnego poziomu złożoności oraz wiąże się z dodaniem następnych komponentów, które administrator systemu musi poznać i zrozumieć. Poznanie technologii LVM i sposobów zarządzania nią nie należy do trudnych, ale mimo wszystko to i tak więcej niż w przypadku, gdy nie trzeba się uczyć niczego nowego. LVM oznacza również pojawienie się niewielkiego dodatkowego obciążenia, ponieważ ta technologia zapewnia mapowanie między urządzeniami fizycznymi i logicznymi. Zwykle to obciążenie jest minimalne, ale mimo wszystko istnieje.

Ogólnie rzecz biorąc, zalety technologii LVM są znacznie większe niż jej wady, więc coraz więcej systemów zaczyna wdrażać warstwę LVM bez pytania użytkownika końcowego o zgodę. Wynika to z faktu, że coraz większa funkcjonalność oczekiwana przez użytkowników od systemu operacyjnego po prostu wymaga warstwy LVM. Zatem wdrożenie systemu operacyjnego bez takiej warstwy pozostawiłoby użytkowników w niepewności, dlaczego system nie spełnia oczekiwań — użytkownicy często mogliby nawet nie zdawać sobie z tego sprawy przez miesiące lub lata po wdrożeniu systemu.

Podobnie jak w przypadku innych form wirtualizacji, także wirtualizacja pamięci masowej jest najważniejsza w zakresie *ochrony przed nieznanym*. Jeżeli mielibyśmy więc całkowitą wiedzę na temat sposobu używania systemu w trakcie jego całego cyklu życiowego, wówczas zadania takie jak zmiana wielkości, tworzenie kopii zapasowej, konsolidacja itd. miałyby niewielką wartość. Jednak rzeczywistość jest zupełnie inna.

Ogólnie rzecz biorąc, przyjmuje się, że najlepsze praktyki w zakresie technologii LVM są następujące: o ile nie jesteś w stanie znaleźć solidnych powodów technicznych potwierdzających, że obciążenie powodowane przez LVM przyniesie więcej szkody niż pożytku, to zawsze decyduj się na zastosowanie LVM.

Zarządca woluminów logicznych to element konstrukcyjny o znaczeniu krytycznym podczas tworzenia niezawodnych rozwiązań w zakresie w pamięci masowej. Pod wieloma względami pozwala oddzielić nowoczesną pamięć masową od klasycznych systemów komputerowych. Zrozumienie tego, jak woluminy logiczne pozwalają na tworzenie pamięci masowej, która będzie działała w oczekiwany sposób, daje nam wiele możliwości i zapewnia dodatkowe opcje, takie jak RAID i RAIN omówione w następnym podrozdziale. Dzięki nim technologię LVM można wykorzystać do ochrony danych, rozszerzenia pamięci masowej i poprawy jej wydajności działania.

Wykorzystanie technologii RAID i RAIN

Przedstawiłem wiele sposobów pracy z pamięcią masową. Jednak prawdopodobnie najbardziej ekscytujące będzie rozpoczęcie pracy z macierzą **RAID** (ang. *redundant array of independent disks*), a co za tym idzie: także z macierzą **RAIN** (ang. *redundant array of independent nodes*). Jednak zanim przejdziemy zbyt daleko, trzeba w tym miejscu podkreślić, że macierz RAID to ogromny temat, którego dokładne omówienie może zająć całą książkę. Poznanie sposobu działania macierzy RAID i wszystkich niezbędnych obliczeń pozwalających na zrozumienie niuansów wydajności działania macierzy i związanego z nią ryzyka jest zagadnieniem samym w sobie. W tym miejscu chcę jedynie zamieścić ogólne wprowadzenie do koncepcji macierzy RAID, wyjaśnić, jak można ją wpasować w projekt, a także przedstawić związane z nią najlepsze praktyki i zapewnić Ci dobry punkt wyjścia do dalszych działań.

RAID i RAIN to mechanizmy pozwalające na wykorzystanie wielu *urządzeń pamięci masowej* (urządzeń blokowych) i zastosowanie naturalnej możliwości łączenia tych urządzeń w celu zapewnienia pewnego rodzaju wariantu usprawnionej wydajności działania, niezawodności lub skalowalności ponad to, co może zaoferować tylko pojedynczy napęd. Podobnie jak w przypadku LVM, także RAID i RAIN to przykłady technologii *środka stosu* wykorzystujące konsumenckie urządzenia blokowe i zapewniające interfejs urządzenia blokowego, a tym samym możliwe do umieszczenia na warstwie LVM, pod warstwą LVM, na bazie innej macierzy RAID, na bazie połączenia różnych urządzeń sprzętowych itd. To jest niezwykle elastyczne rozwiązanie.

W rzeczywistości technologie RAID i RAIN można uznać za bardzo wyspecjalizowane formy LVM. Nikt nigdy nie mówi o tych technologiach w taki właśnie sposób. Jeżeli np. na bożonarodzeniowej imprezie firmowej zaczniesz przedstawiać macierz RAID jako wyspecjalizowaną wersję LVM, inni mogą się na Ciebie dziwnie patrzeć, mimo że tak właśnie jest. RAID i RAIN oferują tylko wyjątkowo mały podzbiór funkcjonalności LVM i na dodatek bardzo się na niej koncentrują. Nierzadko można się spotkać z sytuacją, że ogólnego przeznaczenia warstwa LVM ma wbudowaną funkcjonalność macierzy RAID. Trendem w zintegrowanych systemach plików jest posiadanie warstw LVM i RAID zintegrowanych z danym systemem plików.

RAID

Standard RAID został początkowo wprowadzony jako zestaw technologii działających na poziomie urządzenia blokowego i pozwalających na przedstawienie wielu urządzeń jako jednego. Tego rodzaju technologie pojawiały się już w latach 60. ubiegłego wieku. Pojęcie i nowoczesna definicja tej technologii pochodzą z 1988 r., co oznacza, że macierz RAID faktycznie powstała jeszcze przed opracowaniem ogólnego przeznaczenia zarządcy woluminów logicznych (LVM).

W rozwiązaniu typu macierz RAID grupa urządzeń blokowych jest łączona ze sobą w jedną całość, z wykorzystaniem dowolnej z wielu różnych konfiguracji pamięci masowej, które w przypadku macierzy RAID są nazywane *poziomami*. Poszczególne poziomy macierzy są odmienne i używają różnych mechanizmów do połączenia grupy dysków w pojedynczy dysk wirtualny. Dzięki wykorzystaniu wielu dysków i skonfigurowaniu ich w taki sposób, jakby były pojedynczym napędem, można wedle potrzeb rozszerzyć różne aspekty pamięci masowej. Jednak zysk osiągnięty w jednym obszarze wiąże się z kosztem poniesionym w innym. Dlatego też bardzo ważne znaczenie ma zrozumienie sposobu działania macierzy RAID.

Macierz RAID to obszar, w którym ogromnie istotne jest to, aby administrator dokładnie znał wewnętrzny sposób działania podsystemu pamięci masowej. Za zaskakujący można uznać fakt, że pod tym względem bardzo niewielu administratorów wie, jak tak naprawdę działa używany przez nich system.

Wprawdzie macierz RAID jest definiowana jako seria *poziomów*, ale to nie powinno Cię zwieść. Poziomy to odmienne typy pamięci masowej, współdzielące podstawowe koncepcje macierzy RAID. Te poziomy nie są wbudowane jeden na drugim, zaś wyższy numer poziomu nie oznacza lepszego produktu.

Skoro macierz RAID to tak naprawdę pewna postać LVM, może się znajdować w dowolnym miejscu stosu pamięci masowej. W rzeczywistych rozwiązaniach można ją spotkać niemalże wszędzie. Najpopularniejsze poziomy macierzy RAID to w rzeczywistości *macierze RAID umieszczone na stosie* i wykorzystujące artefakty swojego projektu. Najbardziej znanym przykładem jest tutaj macierz RAID 10.

Macierz RAID może być implementowana sprzętowo lub programowo. Implementacja sprzętowa przypomina kartę graficzną, która jest podłączona bezpośrednio do monitora i odciąża główne systemy komputera, prowadząc bezpośrednią komunikację ze sprzętem. Karta sprzętowej macierzy RAID działa w ten sam sposób, zmniejsza obciążenie głównego systemu komputerowego i bezpośrednio współpracuje z urządzeniami pamięci masowej. Taka karta potencjalnie oferuje również specjalną funkcjonalność, np. buforowanie. Natomiast programowe rozwiązanie w postaci macierzy RAID wykorzystuje ogólnie znacznie potężniejszy system komputerowy, procesor i pamięć operacyjną oraz zapewnia elastyczniejsze konfiguracje. Oba wymienione podejścia świetnie się sprawdzają.

Każdy poziom macierzy RAID ma unikatowy zbiór właściwości, a jego użycie ma sens w odmiennych sytuacjach. Macierz RAID to bardzo zaawansowany temat, którego dokładne omówienie wymagałoby oddzielnej książki. To również nie jest temat, który można zreferować zbyt szybko, ponieważ pobieżne poznanie tego zagadnienia może być zagrożeniem dla danych przechowywanych w pamięci masowej. Czynniki ryzyka związanego z macierzą RAID są często sprowadzane do praktycznie bezsensownych stwierdzeń typu: *Ile dysków może ulec awarii, aby wciąż można było odzyskać macierz RAID poziomu X?* To tak naprawdę nic nie znaczy i stanowi jedynie sposób na uproszczenie niezwykle zaawansowanej koncepcji oraz skonwertowanie jej na postać czegoś bardzo prostego, co można zapamiętać bądź przedstawić na wykresie. Macierz RAID nie działa w taki sposób. Każdy jej poziom ma pewną charakterystykę wydajności działania, niezawodność, koszt, skalowalność, rzeczywiste implementacje itd.

RAIN

Wraz z upływem czasu, gdy systemy stawały się coraz większe i coraz bardziej skomplikowane, ograniczenia dotyczące macierzy RAID zaczęły być widoczne. Macierz RAID jest prosta i łatwa do zaimplementowania, a jednocześnie jest nieelastyczna i istnieją pewne kluczowe funkcjonalności, np. zmiana wielkości, zautomatyzowane przywrócenie równowagi oraz elastyczna zmiana węzłów, które są obsługiwane raczej kiepsko. Dlatego też pojawiła się potrzeba powstania nowej rodziny technologii.

Technologia RAIN wystrzega się podejścia **pełnego urządzenia blokowego** do macierzy, na której bazuje macierz RAID, i zamiast tego przeprowadza podział pamięci masowej na mniejsze fragmenty, często bloki, oraz obsługuje replikację na tym poziomie. W celu efektywnego działania technologia RAIN musi nie tylko zapewniać obsługę koncepcji tych bloków, ale również urządzeń blokowych (czyli *dysków*), na których bazuje koncepcja, a także węzłów zawierających te dyski. Takie podejście doprowadziło do powstania technologii RAIN, której skrót można przedstawić jako **nadmiarowa macierz niezależnych węzłów** (ang. *redundant array of independent nodes*).

Co zaskakujące, w technologii RAIN to nie węzły są koniecznie komponentami nadmiarowymi. W rzeczywistości rozwiązanie typu RAIN można zaimplementować w pojedynczym urządzeniu fizycznym, aby bezpośrednio konkurowało z tradycyjną macierzą RAID w jej najprostszej postaci. Jednak tego rodzaju rozwiązania rzadko są spotykane.

Skoro technologia RAIN obsługuje replikację na poziomie bloku, może się pochwalić wieloma zaletami nieoferowanymi przez macierz RAID. Na przykład umożliwia płynne używanie urządzeń o różnej pojemności. Dyski o różnej pojemności można *dowolnie* umieszczać w serwerze i łączyć je ze sobą, zachowując przy tym doskonałą efektywność.

W przypadku macierzy RAID jeżeli napęd ulegnie awarii, konieczne jest zastąpienie go nowym napędem, który będzie mógł być zastosowany w danej macierzy. To często

okazuje się problemem, zwłaszcza w przypadku starszych macierzy. Rozwiązanie typu RAIN pozwala uniknąć tego problemu — możliwe jest używanie dowolnego połączenia napędów o różnej pojemności w macierzy i zrekompensowanie utraconej pojemności uszkodzonego napędu.

Rozwiązania w technologii RAIN są implementowane w tak różny sposób, że każde z nich jest unikatowe. Dlatego też trudno jest w tym przypadku mówić o jakimkolwiek standardzie. Obecnie większość rozwiązań jest własnościowych. Wprawdzie powstało kilka doskonale znanych produktów typu open source, które stały się standardowymi komponentami ekosystemu Linuksa, ale ogólnie rzecz biorąc, są one uznawane za zewnętrzne dla dystrybucji. Działają na zasadzie podobnej do produktów własnościowych i tak trzeba je traktować.

W przyszłości możemy być świadkami poważnej konsolidacji bądź przynajmniej standaryzacji na rynku technologii RAIN, gdy stanie się ona łatwiej dostępna i lepiej zrozumiała. Do tego czasu korzystając z rozwiązania typu RAIN, trzeba pamiętać, że replikacja bazująca na blokach *może działać*, a ponadto trzeba mieć świadomość istnienia potencjalnie ogromnych różnic między wybranymi podejściami projektowymi. Technologia RAIN może być wbudowana bezpośrednio w jądro bądź też może istnieć w postaci aplikacji uruchomionej na wyższym poziomie stosu. W pewnych przypadkach da się ją nawet uruchomić w maszynie wirtualnej działającej na podstawie danego hipernadzorcy. Sposób reakcji rozwiązania typu RAIN na utratę napędu — podczas równoważenia obciążenia, podczas przywracania równowagi systemu zachwianej na skutek awarii, podczas odtwarzania systemu po naprawie itd. — nie jest zdefiniowany przez żaden standard. Aby korzystać z tego typu rozwiązania, musisz samodzielnie wyszukać informacje i dokładnie poznać konkretne rozwiązanie, którego użycie rozważasz. Ponadto musisz się zastanowić nad tym, jak artefakty tego rozwiązania będą wraz z upływem czasu wpływały na wykonywane zadania.

Można niemalże zagwarantować, że rozwiązanie typu RAIN stanowi przyszłość systemów pamięci masowej. W miarę coraz częstszego przechodzenia w kierunku klastrów, hiperkonwergencji, chmury oraz rozwiązań bazujących na chmurze technologia RAIN wydaje się coraz bardziej naturalna. Im lepiej będzie rozumiana, tym częściej będzie stosowana. To po prostu wymaga czasu, nawet w przypadku technologii, które nie są absolutną nowością.

Niemalże każdy system produkcyjny, którego projektowaniem lub obsługą kiedykolwiek się zajmiesz, będzie zawierał pewną postać technologii typu RAID lub RAIN zaimplementowaną lokalnie bądź zdalnie. Musisz się przygotować do myślenia o tym, jak decyzje dotyczące poziomu macierzy RAID lub jej konfiguracji, a także implementacji RAIN będą wpływały na systemy pozostające pod Twoją opieką. Poświęcenie czasu na dokładne przeanalizowanie czynników pamięci masowej w tego rodzaju frameworkach agregacji wielu urządzeń to jedna z najlepszych rzeczy, które możesz

zrobić, i jednocześnie dostarczy Ci bardzo cennej wiedzy, gdy zajmujesz się administrowaniem systemami. W następnym podrozdziale pokażę, jak można wykorzystać tę technologię w taki sposób, aby lokalna pamięć masowa mogła się stać nadmiarowa dzięki systemom bądź węzłom zewnętrznym.

Replikowana lokalna pamięć masowa

Typem pamięci masowej, który prawdopodobnie ma najbardziej krytyczne znaczenie i jednocześnie pozostaje najmniej znany, jest **replikowana lokalna pamięć masowa** (ang. *replicated local storage*, RLS). To nie jest trudna koncepcja, a wręcz przeciwnie. Jednak wiele mitów wiąże się z innymi koncepcjami, np. SAN, które podobno przesłoniły funkcjonalność RLS. Na przykład wiele osób zaczęło używać wyrażenia **współdzielona pamięć masowa** względem proxy dla *zewnętrznej pamięci masowej* bądź nawet dla rozwiązania typu *SAN*. Mimo to zewnętrzna pamięć masowa wcale nie oznacza, że jest lub może być współdzielona. Podobnie lokalna pamięć masowa nie oznacza, że nie jest lub nie może być współdzielona.

Pojęcie replikowana pamięć masowa odnosi się do dwóch lub więcej systemów komputerowych posiadających lokalną pamięć masową, która jest replikowana między nimi. Z perspektywy poszczególnych komputerów pamięć masowa zostaje podłączona lokalnie, jak w tradycyjnej instalacji. Jednak istnieje proces replikujący dane z jednego systemu w inny, więc zmiany wprowadzone w jednym komputerze będą odzwierciedlone w drugim.

Replikacja lokalnej pamięci masowej może się odbywać na wiele sposobów. Najprostszy i jednocześnie najstarszy z nich polega na wykorzystaniu **Network RAID**, czyli technologii macierzy RAID używanej po prostu poprzez sieć. **Mirrored RAID** (inaczej **RAID 1**) to najprostsze rozwiązanie tego typu i jednocześnie najlepszy przykład.

Mamy dwa sposoby na obsługę takiego rozwiązania. Pierwszy to wykorzystanie pary gorący–zimny. W tym przypadku jeden węzeł jest *gorący* i ma uprawnienia zapisu w pamięci masowej, natomiast drugi może odczytywać dane z tej pamięci masowej i ewentualnie przejąć funkcję węzła zapisującego dane, gdy zajdzie taka potrzeba lub gdy węzeł gorący ulegnie awarii. Ten model jest łatwy i podobny do wielu tradycyjnych modeli współdzielonej pamięci masowej w rozwiązaniu typu SAN. Takie podejście pozwala na używanie zwykłych (tzn. nieklastrowanych) systemów plików takich jak XFS lub ZFS.

Drugi sposób polega na użyciu systemu typu live–live, w którym wszystkie węzły replikujące pamięć masową mogą w każdej chwili odczytywać i zapisywać dane. To wymaga użycia tego samego klastrowanego systemu plików, który będzie potrzebny w przypadku pracy z dowolną współdzieloną blokową pamięcią masową. Podobnie jak w przypadku urządzenia typu SAN używanego jednocześnie przez dwa węzły, węzły

w klastrze RLS będą się musiały komunikować przez przechowywanie informacji o swojej działalności w specjalnie wydzielonym do tego celu obszarze klastrowanego systemu plików.

Replikowana lokalna pamięć masowa zapewnia wiele zalet zwykle oferowanych przez rozwiązanie typu SAN lub inną zewnętrzną pamięć masową. Przede wszystkim będzie to możliwość jednoczesnego uzyskania dostępu do danych przez wiele węzłów. Jednocześnie zachowujemy korzyści wynikające z lokalnego położenia pamięci masowej, czyli większą wydajność działania i niezawodność dzięki istnieniu mniejszej liczby zależności. Oczywiście replikacja wiąże się z pewnym obciążeniem i trzeba je wziąć pod uwagę. Istnieje wiele sposobów, na które można skonfigurować replikację. Część z nich powoduje niewielkie obciążenie, inne zaś wiążą się z dużymi kosztami.

Bardzo często można się spotkać z przekonaniem, że replikowana lokalna pamięć masowa jest nowością lub czymś zupełnie nietypowym. Nic bardziej mylnego. W rzeczywistości rzadko się wie, że w systemach wymagających wysoce niezawodnych pamięci masowych zawsze jest używane rozwiązanie typu RLS. Niezależnie od tego, czy używa się go lokalnie (tzn. jest bezpośrednio dołączony do komputera), czy też zdalnie (tzn. zdalna pamięć masowa używa rozwiązania typu RLS w celu zapewnienia większej niezawodności), technologia RLS to jedna z podstawowych technologii stosowanych w niemalże każdym systemie pamięci masowej charakteryzującym się wysoką dostępnością.

Rozwiązanie typu RLS jest oferowane w różnych odmianach, przede wszystkim Network RAID i RAIN. W takim przypadku można się pokusić o użycie nazwy Network RAIN, ale nie zrobię tego. W przeciwieństwie do macierzy RAID, która niemalże zawsze jest wyłącznie lokalna, rozwiązanie typu RAIN jest niemalże zawsze używane w rozwiązaniu typu RLS i wówczas przyjmuje się, że ma ono naturę sieciową lub przynajmniej użycie sieci stanowi jedną z opcji.

Rozwiązanie typu RLS jest na platformie Linux oferowane w różnych postaciach, w tym miejscu nie mam możliwości omówienia wszystkich dostępnych opcji. Zamiast tego skoncentruję się na kilku najczęściej używanych. RLS to obszar, na którym istnieje wiele rozwiązań zarówno komercyjnych, jak i typu open source, charakteryzujących się różną wydajnością działania, niezawodnością i funkcjonalnością, a także zwykle zaimplementowanych na odmienne sposoby. RLS może dodać nową warstwę złożoności do każdego rozwiązania pamięci masowej, ponieważ trzeba uwzględnić komunikację z lokalną pamięcią masową, komunikację dotyczącą replikacji oraz potencjalnie komunikację sieciową między węzłami i zdalną pamięcią masową (tzn. lokalną względem innego węzła). Poza tym trzeba wiedzieć, jak algorytmy i protokoły współdziałają ze sobą.

DRBD

Pierwszą i najprostszą technologią typu RLS na platformie Linux jest **DRBD** (ang. *distributed replicated block device*), czyli po prostu warstwa Network RAID umieszczona bezpośrednio w jądrze systemu Linux. Według artykułu zamieszczonego w Wikipedii DRBD to nie rozwiązanie typu *Network RAID*, przy czym system DRBD został przedstawiony jako Network RAID. Niezależnie od tego, czy mamy tutaj do czynienia jedynie z czystą semantyką, czy nie, w praktyce system DRBD jest w zasadzie nie do odróżnienia od Network RAID, zarówno w opisie, jak i w implementacji. Podobnie jak każdy rodzaj macierzy RAID, także DRBD używa urządzeń blokowych i jest udostępniane w postaci urządzenia blokowego, które może być umieszczone w dowolnym miejscu na stosie pamięci masowej, jak się to dzieje w rozwiązaniach typu RAID i LVM.

System DRBD został wbudowany w mechanizm macierzy RAID 1 (tzw. Mirrored RAID), a także pozwala, aby w zestawie składającym się z dwóch lub więcej węzłów poszczególne węzły pobierały od innych kopie danych.

DRBD to bardzo elastyczne i niezawodne rozwiązanie. Z powodu jego prostoty większość administratorów systemów może łatwo poznać sposób jego działania i dostrzec możliwość wpasowania w istniejącą infrastrukturę pamięci masowej. Jednak z powodu ograniczenia DRBD do replikacji pełnego urządzenia blokowego przez utworzenie jego lustrzanego odbicia, podobnie jak w przypadku macierzy RAID 1, możliwości w zakresie skalowania są dość ograniczone. Rozwiązanie typu DRBD pozostaje niezwykle mocno skoncentrowane na klasycznym klastrze dwóch węzłów lub, w bardzo niszowych przypadkach zastosowania, ogromna liczba węzłów obliczeniowych musi współdzielić tę samą małą ilość identycznych danych.

Zapewnienie elastyczności rozwiązania typu DRBD

Skoro DRBD jest w rzeczywistości tylko narzędziem programowej macierzy RAID, masz nad nim pełną kontrolę i ponieważ macierz RAID działa jak rozwiązanie typu LVM zapewniające elastyczność, która pozwala na jego implementację w dowolnym miejscu stosu, możesz zamienić DRBD na rozwiązanie charakteryzujące się znacznie większą skalowalnością, niż na początku mogłoby się wydawać. Jednak ten proces jest całkowicie ręczny, choć teoretycznie istnieje możliwość jego skryptowania bądź utworzenia narzędzi pozwalających na automatyzację tego rodzaju procedur.

Jedną z potężniejszych technik, którą można wykorzystać, jest koncepcja *rozbudowy* macierzy RAID 1 za pomocą dodatkowego urządzenia blokowego w celu odwzorowania macierzy RAID 1E działającej na zasadzie podobnej do RAID 1, ale zapewniającej skalowalność. Ta technika wygląda następująco: fizyczna przestrzeń w pamięci masowej węzła zostaje logicznie podzielona na dwie (lub teoretycznie więcej) sekcje z wykorzystaniem technologii LVM. W standardowej konfiguracji Network RAID w przestrzeni pamięci masowej węzła drugiego zostaje utworzone lustrzane odbicie całej

przestrzeni pamięci masowej węzła pierwszego. Jednak teraz gdy pamięć masowa w poszczególnych węzłach została podzielona, lustrzane odbicie pierwszej części węzła pierwszego jest tworzone w pierwszej części węzła drugiego. W przypadku węzła drugiego dzieje się to samo, ale względem węzła trzeciego. Następnie dla węzła trzeciego mamy to samo, ale względem węzła czwartego. Ten proces zachodzi w nieskończoność aż do chwili, gdy ostatni węzeł nie zrobi tego samego z węzłem pierwszym, zamykając tym samym *krąg*. Wówczas każdy system będzie miał zdefiniowaną macierz RAID 1 dla danych, utworzoną razem z dwoma innymi węzłami jako jego parą lustrzanego odbicia. Dzięki temu rozwiązanie typu Network RAID 1 może mieć nieograniczoną wielkość.

Nie ulega wątpliwości, że to jest naprawdę potężna technika. Mimo to okazuje się wyjątkowo uciążliwa podczas dokumentowania rozwiązania i jego późniejszej obsługi. Jeżeli będzie zastosowana w klastrze statycznym, który nigdy nie ulega zmianie, wówczas sprawdzi się doskonale. Natomiast jeśli regularnie powiększasz lub modyfikujesz klaster, wtedy może się okazać, że zastosowanie tej techniki niesie wiele problemów.

DRBD i większość technologii Network RAID zwykle charakteryzuje się dobrą ogólną wydajnością działania i, co prawdopodobnie znacznie ważniejsze, jest to raczej przewidywalna wydajność. Ponieważ DRBD przedstawia ostateczne urządzenie blokowe, więc to rozwiązanie jest z natury lokalne. W celu zdalnego uzyskania dostępu do zasobów DRBD konieczne będzie użycie DRBD jako elementu konstrukcyjnego rozwiązania typu SAN, które następnie będzie zdalnie współdzielone. To oczywiście jest jedynie semantyka. Rozwiązanie DRBD zawsze jest lokalne, ponieważ dla DRBD rozwiązanie SAN to węzeł lokalny, a interfejs SAN to kolejna warstwa w górnej części przysłowiowego stosu. Dlatego też choć rozwiązanie SAN będzie zdalne, to DRBD okaże się lokalne.

Gluster i CEPH

Wprawdzie **Gluster** i **CEPH** to dwie zupełnie różne technologie, ale obie to bezpłatne i udostępnione jako open source nowoczesne rozwiązania typu RAIN. Zostały opracowane dla Linuksa, aby zapewnić wysoki poziom niezawodności i skalowalności. Oba te rozwiązania przynajmniej oferują możliwość posiadania lokalnej pamięci masowej dla danego węzła obliczeniowego. W obu przypadkach mamy do czynienia z bardzo skomplikowanymi rozwiązaniami zawierającymi wiele opcji wdrożenia. Nie można po prostu przyjąć założenia, że użycie jednej z tych technologii wskazuje na lokalną bądź zdalną pamięć masową. Lokalna pamięć masowa to dotychczas znacznie częściej spotykane rozwiązanie. Oba te rozwiązania pozwalają na bezpośrednie tworzenie oddzielnej warstwy pamięci masowej, która będzie dostępna poprzez sieć, o ile została do tego zaprojektowana.

Te technologie są znacznie bardziej skomplikowane i oferują znacznie więcej opcji, niż jestem w stanie tutaj omówić. Z konieczności mogę je omówić tylko ogólnie, ale to powinno być wystarczające do naszych potrzeb.

Rozwiązanie typu RAIN to najczęściej stosowane podejście podczas obsługi ogromnej puli serwerów (węzłów obliczeniowych), które współdzielą pulę pamięci masowej. Dzięki tej technice pamięć masowa może być lokalna oraz ma możliwość automatycznego przywrócenia równowagi w przypadku wystąpienia awarii. Jednak to rzadko będzie gwarantowało lokalność danych. Pamięć masowa w grupie to jedynie pula. Dlatego może być zalecenie zachowania lokalności danych, ale to nie jest ściśle wymagane jak w przypadku rozwiązania typu DRBD. Dzięki DRBD zachowujesz większą kontrolę, ale znacznie większą elastyczność i lepszy poziom wykorzystania zasobów uzyskasz za pomocą rozwiązań Gluster i CEPH.

Zewnętrzne rozwiązania własnościowe i typu open source

Pomijając to, co zostało bezpośrednio wbudowane bądź dołączone do dystrybucji Linuksa, istnieje jeszcze wiele opracowanych przez podmioty zewnętrzne komponentów, które można zainstalować. Niemal wszystkie te produkty zaliczają się do kategorii rozwiązań typu RAIN i różnią się ceną, obsługą techniczną oraz możliwościami. Kilka wartych uwagi produktów to: *LizardFS*, *MooseFS* i *Lustre*.

Niemożliwe jest omówienie potencjalnej gamy produktów komercyjnych, które już istnieją lub dopiero będą oferowane. Pamięć masowa typu RAIN to obecnie intensywnie rozwijany obszar i wciąż mogą się pojawiać producenci oferujący nowe produkty tego typu, ale niekoniecznie będą one powszechnie dostępne. W niektórych przypadkach można znaleźć komercyjny system typu RAID lub RAIN, który jest dostępny jedynie w połączeniu z urządzeniem danego typu bądź też w ramach konkretnego projektu. Jednak we wszystkich tego rodzaju systemach pamięci masowej stosowane są te same podstawowe koncepcje. Jeżeli poznasz je i sposób ich działania, wówczas możesz podejmować dobre decyzje dotyczące systemów pamięci masowej, nawet gdy nie masz doświadczenia pracy z konkretnymi implementacjami.

Abstrakcja wirtualizacji pamięci masowej

Bardzo łatwo się zgubić podczas omawiania pamięci masowej i zapomnieć, że w większości przypadków pamięć masowa to nie jest kwestia, którą musi zajmować się administrator systemu. Przynajmniej nie w stopniu, w którym została tutaj omówiona.

Administrator pamięci masowej

W większych organizacjach nierzadko zdarza się, że występują oddzielne stanowiska dla osób zajmujących się systemami i pamięcią masową. To wynika z faktu istnienia wielu złożonych kwestii dotyczących pamięci masowej, więc jak najbardziej ma sens

powołanie zespołu, którego członkowie rozumieją niuanse związane z pamięcią masową i zajmują się nią. Jeżeli jednym z Twoich pracodawców była firma z listy Fortune 500, wówczas takie rozwiązanie nie jest Ci obce.

Do wybranych z największych problemów związanych z takim podejściem zalicza się oddzielenie osób dokładnie znających zadania do wykonania (obciążenie) i tym samym odsunięcie ich od najważniejszych czynników, które określają wydajność działania i niezawodność tego obciążenia. Separacja często powoduje, że ważne decyzje architekturalne mają podłoże polityczne, a nie techniczne. Jeżeli używasz lokalnej pamięci masowej, wówczas tak do końca nie możesz rozdzielić zespołów zajmujących się pamięcią masową i systemami. Z tego powodu w wielu firmach często podejmowano koszmarne z technicznego punktu widzenia decyzje projektowe prowadzące do powstania w organizacji silosów umiejętności, bez zastanowienia się, jaki to będzie miało wpływ na wykonywane zadania. Wdrażanie technologii SAN dość często zdarza się właśnie z tego powodu.

Niezależnie od efektywności takiego podejścia, gdy jest ono stosowane, na ogół oznacza zabranie pamięci masowej z rąk administratorów systemów. Z jednej strony to stanowi ułatwienie pracy administratora systemu, natomiast z drugiej podcina mu skrzydła i nie pozwala pokazać pełni możliwości. Administratorzy systemów mogą wymagać określonych poziomów wydajności działania lub niezawodności oraz muszą ufać, że ich potrzeby zostaną spełnione lub przynajmniej że nie będą pociągnięci do odpowiedzialności, gdy to się nie uda.

Często się zdarza rozdzielanie zespołów odpowiedzialnych za system i platformy. W takim przypadku można zaobserwować dokładnie ten sam efekt. Zespół odpowiedzialny za platformę, zarządzający w tle hipernadzorcą, będzie zapewniał pojemność pamięci masowej zespołowi systemów, który z kolei musi wykorzystać to, co jest dla niego dostępne.

W obu tych przypadkach mamy do czynienia z abstrakcją pamięci masowej z systemu oraz z dostarczeniem zespołowi zajmującemu się systemem **ślepego urządzenia blokowego**. W takich sytuacjach nadal trzeba znać sposób działania poszczególnych komponentów, wiedzieć, jakie pytania powinny być zadawane, a także potrafić zarządzać systemem plików znajdującym się w dostarczonym urządzeniu blokowym. Interfejs urządzenia blokowego pozostaje uniwersalny i *niezależny* od zespołu pamięci masowej bądź platformy.

Warto w tym miejscu dodać, że to samo często zdarza się również w przypadku zespołu odpowiedzialnego za platformę. Może on otrzymywać ślepą pamięć masową od zespołu zajmującego się pamięcią masową, wykorzystać ją na warstwie hipernadzorcy, a następnie podzielić to urządzenie blokowe na mniejsze i przekazać zespołowi zajmującemu się systemem.

Obecnie w większości przypadków systemy Linux mogą być w pewien sposób wirtualizowane. Konieczne jest poznanie tematu pamięci masowej na różnych warstwach stosu, ponieważ system Linux sam może pełnić funkcję hipernadzorcy (np. w KVM), być wykorzystywany do kontrolowania hipernadzorcy (jak to się dzieje w rozwiązaniach typu Xen) bądź też dostarczać pamięć masową hipernadzorcom wysokiego poziomu (takim jak VirtualBox). W wymienionych przypadkach to Linux potencjalnie zarządza wszystkimi aspektami dotyczącymi pamięci masowej. Linuksa można również wykorzystać do utworzenia urządzenia SAN bądź pewnej innej warstwy pamięci masowej. Należy poznać i zrozumieć kwestie dotyczące pamięci masowej, choć w większości przypadków system Linux będzie pobierał pamięć masową z hipernadzorcy, nawet jeśli jesteś menedżerem tego hipernadzorcy.

Wprawdzie istnieje wiele różnych sposobów konfiguracji, ale większość osób decyduje się zdefiniować hipernadzorcę do pracy podobnie jak warstwa LVM w przypadku pamięci masowej. W pewnym stopniu to jest jak przypadek specjalny, ponieważ zachodzi konwersja z bloku na system plików, a następnie z powrotem na blok w celu przekazania maszynie wirtualnej, ale koncepcja pozostaje taka sama. W niektórych konfiguracjach hipernadzorcy mamy po prostu przekazanie przez bezpośrednie połączenie blokowe do pamięci masowej, niezależnie od tego, czy będzie nią dysk lokalny, rozwiązanie typu SAN, czy wolumin logiczny utworzony za pomocą LVM. To wszystko są dostępne podejścia i zapewniają maszynie wirtualnej więcej opcji określających sposób pracy z pamięcią masową. Jednak ogólnie rzecz biorąc, warstwa blokowa pamięci masowej kończy się na hipernadzorcy i zostaje przekształcona na system plików. Na bazie systemu plików tworzy **kontener urządzenia blokowego** i pozwala maszynom wirtualnym na używanie tych urządzeń w taki sam sposób, jakby były zwykłymi urządzeniami blokowymi. Takie rozwiązanie jest oczekiwane od wirtualizacji. Wiele osób w rzeczywistości odwołuje się do tego artefaktu wirtualizacji jako nieodłącznego elementu samej wirtualizacji, chociaż tak nie jest.

Tę technikę można wykorzystać również wewnątrz systemu. Przykłady obejmują montowanie systemu plików takich jak *qcow2*, *vhdx* i *iso*. To są zadania wykonywane codziennie, przy czym rzadko o nich myślimy i nie do końca zdajemy sobie sprawę, na czym tak naprawdę polegają.

Oczywiście podczas pobierania pamięci masowej z hipernadzorcy pojawiają się obawy dotyczące standardowej (niereplikowanej) lokalnej pamięci masowej, replikowanej lokalnej pamięci masowej, standardowej (niereplikowanej) zdalnej pamięci masowej lub replikowanej zdalnej pamięci masowej — wszystkie one mogą znajdować się na innej warstwie niż system. Mimo to decyzje wciąż są podejmowane oraz ostatecznie mają wpływ na sposób działania systemów.

W ten sposób omówiłem wiele podejść w zakresie abstrakcji pamięci masowych oraz paradygmatów takich jak LVM, RAID i RAIN. Trzeba się teraz zastanowić, jak można połączyć te technologie podczas tworzenia własnych rozwiązań w zakresie pamięci masowej.

Analiza ryzyka i architektury pamięci masowej

Nic nie powoduje większego ryzyka dla systemu niż pamięć masowa. To powinno być oczywiste, ale mimo to trzeba o tym wyraźnie powiedzieć. Pamięć masowa to obszar, w którym administrator systemu ma wiele możliwości do wprowadzenia zmian. To również obszar, gdzie można ponieść porażkę i to sromotną.

Aby zapanować nad ryzykiem i możliwościami związanymi z pamięcią masową, konieczne jest poznanie całego stosu pamięci masowej oraz sposobów, na jakie poszczególne warstwy i komponenty ze sobą współdziałają. Pamięć masowa może być przytłaczająca, ponieważ istnieje w niej tak wiele zmiennych i opcjonalnych komponentów.

Do pewnego stopnia to uczucie przytłoczenia można wyeliminować poprzez wykorzystanie wzorców projektowych pozwalających na osiągnięcie sukcesu oraz zdobycie wiedzy, która pomaga w ustaleniu, kiedy powinno być rozważone użycie poszczególnych wzorców.

Ogólna architektura pamięci masowej

W **architekturze pamięci masowej** tak naprawdę istnieją dwa ogólne aspekty: *lokalny* kontra *zdalny* oraz *standardowy* kontra *replikowany*.

Naturalnym założeniem większości osób jest natychmiastowe przekonanie, że replikowany i zdalny to oczywiste punkty wyjścia. W rzeczywistości to nieprawda. To prawdopodobnie są najmniej sensowne punkty wyjścia dla pamięci masowej z powodu najmniej użytecznego połączenia czynników.

Prosta lokalna pamięć masowa — klocek

Możesz wierzyć lub nie, ale dla firm różnej wielkości najbardziej odpowiednim projektem z zakresu pamięci masowej jest lokalna **niereplikowana pamięć masowa**. Pamiętaj, że w kontekście architektury pamięci masowej termin *replikowana nie* oznacza braku kopii zapasowej bądź lokalnej replikacji (np. macierz RAID 1), ale odwołuje się do replikacji pamięci masowej w czasie rzeczywistym bądź niemalże w czasie jednoczesnym do drugiego, całkowicie oddzielnego systemu.

Do tego zagadnienia jeszcze powrócę i omówię je nieco inaczej podczas prezentacji ogólnego projektu systemu, a nie omawiania samej pamięci masowej. Podobnie jak w wielu innych sytuacjach życiowych, także tutaj zachowanie prostoty ma największy sens.

Replikacja brzmi fantastycznie i wydaje się absolutnie niezbędną funkcjonalnością, ale wiąże się z kosztem, często nawet stosunkowo wysokim. Ponadto replikacja zwykle wpływa na wydajność działania, potencjalnie nawet dość znacząco.

Replikowanie katastrofy

Często popełnianym błędem podczas projektowania pamięci masowej jest działanie pod wpływem emocji i przekonanie, że im bardziej rozbudowana replikacja, tym większa ochrona przed katastrofą. Do pewnego stopnia to będzie prawdą. Lokalne replikowanie pewnych plików za pomocą macierzy RAID 1 zapewnia dość dużą ochronę przed awarią pojedynczego dysku twardego, zaś replikacja zdalna może zapewnić ochronę w przypadku awarii całego węzła. Jednak w żadnym z wymienionych rozwiązań nie mamy ochrony przed zdarzeniami takimi jak: przypadkowe usunięcie pliku, złośliwe uszkodzenie pliku, uszkodzenie pliku bądź przeprowadzony atak typu ransomware.

W przypadku wykonania prostej operacji, takiej jak usunięcie pliku, który nie powinien zostać usunięty, nasz potężnym mechanizm replikacji zagwarantuje, że w ciągu milisekundy bądź dwóch operacja usunięcia zostanie replikowana do całego systemu. Tak więc zamiast zapewnić ochronę, ten mechanizm może replikować błąd popełniony przez użytkownika i to szybciej, niż będzie on w stanie na niego zareagować. Nadmiernie rozbudowane mechanizmy replikacji zwykle chronią jedynie przed awarią sprzętową i bardzo szybko może się okazać, że utworzony w taki sposób system nie był warty poniesionych nakładów.

Pierwszy poziom macierzy (RAID 1) będzie niezwykle cenny, ponieważ awaria dysku twardego to bardzo poważne i realne zagrożenie, a nawet drobna awaria może prowadzić do utraty znacznej ilości danych. Jednak replikacja może chronić jedynie przed utratą całego systemu, a taka sytuacja zdarza się znacznie rzadziej. Ochrona zapewniona przez macierz RAID jest względnie tania, a jej implementacja często może się zamknąć w kwocie zaledwie kilku tysięcy złotych. Z kolei replikacja węzłowa będzie wymagała o wiele większych inwestycji w sprzęt, a to już oznacza ogromne nakłady finansowe, liczone nawet w dziesiątkach tysięcy złotych, za jedynie ułamek ochrony zapewnianej już przez macierz RAID.

Mechanizm typu macierz RAID, zwłaszcza RAID 1 (lustrzane odbicie), jest wyjątkowo prosty do implementacji, a samo rozwiązanie pozostaje nieskomplikowane. Rzadko spotykamy się z sytuacją utraty danych spowodowaną przez błąd człowieka popełniony w macierzy RAID 1. Tego samego nie można powiedzieć w przypadku replikowanej pamięci masowej między węzłami. Tutaj możliwości popełnienia błędu jest znacznie więcej, zaś sam błąd człowieka może prowadzić do o wiele większych strat. Decydując się na tego rodzaju kosztowne rozwiązanie, nie ograniczasz ryzyka, a po prostu wprowadzasz kolejne zagrożenia, które również będą musiały być złagodzone.

Wielu administratorów systemów jest przekonanych, że nie mogą używać prostej, lokalnej i niereplikowanej pamięci masowej, a problemy wynikające z polityki stosowanej przez firmy nie mogą być przeoczone. Jeżeli firma będzie *uprawiać politykę* i obwiniać administratora systemu, nawet jeśli to nie on popełnił błąd, a podjęta przez niego decyzja była najlepsza z perspektywy biznesowej, wówczas administrator będzie zmuszony do podejmowania ryzykownych decyzji, które nie leżą w interesie firmy. Nad tym administrator systemów nie ma kontroli.

W niektórych sytuacjach administrator systemu może się zająć takim problemem politycznym przez przedstawienie (i doskonałe udokumentowanie) decyzji związanych z ryzykiem i finansami, aby w ten sposób pokazać, dlaczego decyzja, która ostatecznie mogła doprowadzić do utraty danych, mimo to była właściwa. Żadna decyzja nie jest w stanie eliminować wszystkich zagrożeń. Specjaliści w dziale IT, a zwłaszcza administratorzy systemów, zawsze podejmują decyzje dotyczące wielkości ryzyka, które należy złagodzić, a także decyzje o tym, jakim kosztem to zrobić.

Ocena ryzyka

Jednym z najtrudniejszych i jednocześnie najważniejszych aspektów w branży informatycznej, a zwłaszcza w zakresie administrowania systemami, jest ocena ryzyka pozwalającego na prawidłowe planowanie. Ryzyka nie można się nauczyć formalnie lub organoleptycznie. To jest obszar, na którym niemalże wszystkie jednostki biznesowe spektakularnie zawodzą. W branży informatycznej, w której ryzyko ma absolutnie krytyczne znaczenie w odniesieniu do wszystkich podejmowanych działań, umiejętność oceny ryzyka jest pozostawiona bez szkoleń, bez zasobów i bez pomocy technicznej.

Nauczenie się oceny ryzyka to zajęcie samo w sobie. Istnieje jednak kilka technik, które powinny być używane we wszystkich przypadkach. Te techniki zostaną tutaj omówione. W gruncie rzeczy ryzyko sprowadza się do ustalenia kosztów, które jesteśmy gotowi ponieść w celu realizacji projektu.

Istnieją dwa kluczowe aspekty ryzyka. Pierwszy to niebezpieczeństwo, że coś złego może się zdarzyć. Drugi to wpływ, jaki będzie miało to złe zdarzenie. Pierwszy z wymienionych aspektów można wyrazić w postaci np. *zdarza się X razy w ciągu roku*, o ile okaże się to użyteczne. Natomiast ten drugi można wyrazić w kategoriach pieniężnych, np. *to będzie kosztowało 5000 zł*. Jeżeli coś zdarza się raz na 10 lat, wówczas można powiedzieć, że występuje 0,1 raza w roku. Z kolei jeśli coś będzie się wiązało z kosztem 5000 zł w ciągu 10 lat, to można powiedzieć, że rocznie kosztuje 500 zł. Mamy tutaj do czynienia z wręcz absurdalnym uproszczeniem, w rzeczywistości ryzyko tak nie działa. Mimo wszystko jest to wręcz niewiarygodnie użyteczny sposób wyrażenia decyzji związanych z ryzykiem, w którym to milion różnych czynników zostało sprowadzonych do pojedynczej wartości.

Wykorzystam teraz wymienioną wcześniej liczbę do przedstawienia kosztu strategii łagodzenia ryzyka. Na przykład jeśli zostanie zaimplementowana technologia replikacji wymagająca licencji kosztującej 300 zł rocznie i 10 godzin pracy administratora systemu pobierającego 120 zł za godzinę pracy, przewidywany koszt takiego projektu może wynieść 1500 zł rocznie.

Następną kwestią do rozważenia jest efektywność. Nic nie jest pewne w 100%. Dobra strategia replikacji może zapewnić efektywność na poziomie 95%, natomiast typowa efektywność wynosi mniej więcej 65%. Te wartości można teraz wykorzystać do wykonania działań matematycznych.

Wiadomo, że w omawianym przykładzie ryzyko wiąże się z kosztem wynoszącym około 500 zł rocznie. Wydając 1500 zł rocznie, mamy 95% szans na uniknięcie utraty 500 zł. Zatem 500 zł·0,95 = 475 zł, więc 1500 zł – 475 zł = 1025 zł rocznie kosztuje strategia złagodzenia ryzyka. Te liczby można przedstawić dyrektorowi finansowemu. Przeprowadź przedstawione obliczenia — będziesz w stanie podać wysokość oszczędności, czyli koszt ochrony, a nie straty. Jeżeli zamierzasz pokazać stratę, naprawdę musisz zmienić plan. To oznacza, że mechanizm obronny przed ryzykiem jest, praktycznie rzecz biorąc, przepisem na *katastrofę*.

Być może zabrzmi to banalnie, ale matematyka jest ważna. Typowy administrator systemu i nawet typowy dyrektor finansowy często uciekają się do prostej matematyki, aby w ten sposób pokazać, czy pomysł jest dobry, czy zły, kierując się przy tym wyłącznie emocjami. Matematyka może Ci pomóc. Dzięki niej możesz pójść do dyrektora finansowego lub generalnego i postawić na swoim. Trudno się spierać z matematyką. Przedstaw im obliczenia, a jeśli uznają je za błędne, niech przeprowadzą własne. Jeżeli zdecydują się na zignorowanie wyników obliczeń matematycznych, będziesz już wiedział, dla jakiego rodzaju organizacji pracujesz. W takim przypadku należy się poważnie zastanowić nad przyszłością firmy, dla której osiąganie zysków nie jest podstawowym celem. Jeżeli w przyszłości coś pójdzie źle i wina spadnie na Ciebie, możesz wyciągnąć z szuflady wspomniane wcześniej obliczenia i zapytać: *Skoro to nie była właściwa decyzja, dlaczego nie uwzględniono tych danych podczas jej podejmowania?*

Nie ma lepszego uczucia niż skuteczna obrona szalonej, wydawałoby się, decyzji, której słuszność założeń potwierdzają obliczenia matematyczne. Pokaż, że starasz się pracować jak najlepiej. Nie wystarczy tylko tak mówić i wysuwać bezpodstawne argumenty. Matematyczne obliczenia wykorzystaj do potwierdzenia słuszności swoich decyzji. Podnieś poziom podejmowanych decyzji, aby nie przedstawiały jedynie założeń, ale były podparte naukowo.

Nie każde obciążenie można potraktować w tak prosty sposób. Jednak w większości przypadków da się zastosować tego rodzaju podejście. To powinno być Twoim standardowym założeniem, o ile nie masz mocnych matematycznych dowodów pokazujących, że jest inaczej. Ewentualnie możesz mieć do czynienia z sytuacją, w której nie

można przeprowadzić obliczeń matematycznych, np. system pomocy technicznej, w której czas nieprzerwanego działania i niezawodność mają o wiele większe znaczenie niż pieniądze. We wszystkich pozostałych przypadkach kieruj się matematyką.

Najprostsze podejście ze wszystkich związanych z pamięcią masową można bardzo łatwo potraktować jak *klocek*. Takie podejście jest proste, stabilne, niezawodne, łatwe podczas tworzenia kopii zapasowej i przywracania z niej, a także łatwe do poznania i późniejszej obsługi technicznej. To rozwiązanie pozostaje obecnie niezwykle efektywne, zwłaszcza w nowoczesnych urządzeniach i technologiach pamięci masowej. Można się pokusić o stwierdzenie, że będzie odpowiednie dla 85–90% wszystkich zadań produkcyjnych, niezależnie od wielkości firmy, oraz dla co najmniej 99% małych firm.

RLS — wysoce niezawodne rozwiązanie

Obciążenia i sytuacje, które nie mają sensu w przedstawionej wcześniej prostej architekturze, niemal zawsze zaliczają się do kategorii **replikowana lokalna pamięć masowa** (**RLS**). RLS pozwala na utworzenie wysoce dostępnej pamięci masowej charakteryzującej się świetną wydajnością działania i rozsądnym kosztem. Nic nie jest w stanie równać się wydajności działania i kosztowi bezpośrednio lokalnej pamięci masowej, o czym już wcześniej wspomniałem. Jeżeli jednak potrzebujesz większej dostępności i lepszej niezawodności niż możliwa do zaoferowania przez dane rozwiązanie, wówczas rozwiązanie omawiane w tym punkcie zdecydowanie będzie warte rozważenia.

RLS zapewnia największy możliwy poziom niezawodności, ponieważ składa się z najmniejszej liczby „ruchomych elementów". W przypadku wysokiej dostępności rozwiązania zdalnego trzeba sobie poradzić z odległością, okablowaniem i protokołami sieciowymi, ponieważ przynajmniej będą potrzebne komponenty dodatkowe, a najczęściej wymagany jest dodatkowy sprzęt sieciowy (np. przełączniki). To znacznie wykracza poza ryzyko związane z RLS. Pamiętaj, że jeśli rozwiązanie zdalnej pamięci masowej będzie chciało zapewnić wysoką dostępność, musi to zrobić poprzez wykorzystanie RLS lokalnie w klastrze. Następnie ten klaster pamięci masowej musi być dostępny zdalnie przez sieć. W efekcie zajmujesz się wszystkimi komplikacjami i potencjalnymi problemami dotyczącymi RLS, a ponadto pojawia się obciążenie i ryzyko nieodłącznie towarzyszące technologii zdalnego dostępu.

Jeżeli rozwiązanie w postaci prostej lokalnej pamięci masowej bez zdalnej replikacji sprawdza się w 90% wszystkich przypadków, to RLS zabiera 90% z pozostałych sytuacji (wobec tego oba te rozwiązania zapewniają łącznie obsługę 99% sytuacji). Dzięki odpowiedniemu planowaniu te dwa podejścia są tak proste, efektywne pod względem kosztu i bezpieczne, że w normalnych warunkach trudno o lepsze rozwiązanie. To są Twoje najlepsze możliwości.

Niezawodność alternatywnej pamięci masowej

Wprawdzie RLS może się wydawać ostateczną możliwością w zakresie zapewnienia ochrony pamięci masowej, ale tak nie jest. Dobrze jest, że trzeba polegać na warstwie pamięci masowej w celu zapewnienia niezawodności. Jednak w tych kategoriach to jest raczej przykład rozwiązania niekonwencjonalnego, a nie ostatecznego.

W doskonałym świecie będą istniały mechanizmy pamięci masowej będące warstwami znajdującymi się ponad rzeczywistą warstwą pamięci masowej z komponentami takimi jak baza danych. System zarządzania bazą danych ma o wiele lepsze możliwości wykonania zadania związanego z konserwacją techniczną i koordynacją replikacji danych między węzłami niż te oferowane przez mechanizm replikacji bloków. Sensowne jest zaimplementowanie replikacji tam, gdzie znajduje się logika aplikacji.

Najlepszym rozwiązaniem byłoby, aby aplikacje używały swoich baz danych jako warstwy pamięci masowej i zapewnienia niezawodności stanu, pozostawiając obsługę replikacji systemom znającym dane. Baza danych to jeden z najbardziej idealnych mechanizmów przeznaczonych do replikacji, ponieważ zna dane i ma możliwość podejmowania dobrych decyzji dotyczących danych.

Dlatego też wiele aplikacji korporacyjnych nie używa żadnej postaci pamięci masowej lub nawet systemów replikacji. Używanie *zawodnych* systemów i wysoce niezawodnych *aplikacji* to dobra strategia oferująca korzyści, których nie da się osiągnąć w inny sposób. Z tego powodu czasami można zignorować potrzebę wysokiej dostępności na czystej warstwie pamięci masowej i po prostu skoncentrować się na wydajności działania.

Środowisko testowe — zdalna współdzielona standardowa pamięć masowa

Taka architektura jest bardzo popularna, ponieważ pomaga handlowcom osiągnąć wszystkie cele: *wysoki koszt*, *ryzyko* i *dezorientację*. Nie ulega wątpliwości, że handlowcy starają się zastosować to podejście zamiast każdego innego, ponieważ pozwala im wyciągnąć dodatkowe pieniądze od klienta za kolejne usługi, a jednocześnie przerzuca całą odpowiedzialność na klienta.

Każda architektura ma mniej więcej swoje miejsce w ekosystemie i to tutaj nie jest wyjątkiem. Jednak zanim przejdę do przedstawienia przykładu użycia, najpierw chciałbym wymienić jej zalety: niereplikowana zdalna pamięć masowa jest *wolniejsza*, *obarczona większym ryzykiem* i *znacznie kosztowniejsza* (przynajmniej na pierwszy rzut oka) w porównaniu z mechanizmami lokalnej pamięci masowej. Na czym może polegać wartość takiego rozwiązania?

Największe korzyści pojawiają się w środowisku roboczym, testowym oraz we wszelkich innych, w których nie zachodzi potrzeba trwałego zachowania danych. Korzyści mogą się także pojawiać w ogromnych środowiskach, w których pamięć masowa może zostać sensownie podzielona, aby w ten sposób zapewnić maksymalne oszczędności na dużą skalę.

Niemalże każde ogromne środowisko na pewnym etapie wykonywania swoich zadań potrzebuje tego rodzaju pamięci masowej. Kluczowe znaczenie ma tutaj określenie, w którym miejscu zachodzi taka potrzeba, i uniknięcie próby stosowania takiego środowiska tam, gdzie nie ma to sensu. Skoro zastosowanie tego rodzaju pamięci masowej na dużą skalę jest tanie (i jednocześnie szokująco kosztowne na mniejszą skalę) i skoro menedżerowie często popełniają błąd, uznając zdalną współdzieloną pamięć masową za rodzaj *magicznego pudełka, które nigdy nie ulegnie awarii*, to omawiany rodzaj pamięci masowej zbyt często znajduje zastosowanie tam, gdzie ma to najmniejszy sens. W tym rodzaju pamięci masowej nie ma żadnej magii. To jest po prostu słaba technologia dla większości rozwiązań produkcyjnych i powinna być wybierana z zachowaniem ogromnej ostrożności.

Elegancka awaria

Uniknięcie awarii nie zawsze jest możliwe. Tak naprawdę umiejętność omijania awarii ma krytyczne znaczenie w naszej pracy. Aby sobie dobrze poradzić w sytuacji kryzysowej, trzeba wiedzieć, jak dobrze przetrwać awarię, a kluczowe znaczenie ma tutaj idea **eleganckiej awarii**. W przypadku pamięci masowej ta koncepcja ma jeszcze większe znaczenie niż w innych obszarach.

Idea eleganckiej awarii polega na tym, że jeśli już musi do niej dojść, to powinna mieć postać małych i przyrostowych zdarzeń, a nie pełnej i totalnej katastrofy. Wiele decyzji dotyczących pamięci masowej jest podejmowanych właśnie pod tym kątem. Doskonale wiemy o tym, że coś może pójść źle. Dlatego też staramy się zapewnić możliwość usunięcia problemu, a jednocześnie bierzemy pod uwagę to, *co się może zdarzyć* w razie awarii.

Z tego powodu dość często używa się macierzy RAID 10 i lokalnej pamięci masowej w architekturze. Oczekujemy utworzenia rozwiązania, które w razie niepowodzenia pozwoli zachować bezpieczeństwo, a nie doprowadzić do utraty danych, ponieważ ktoś uznał, że to nie ma znaczenia.

Zgodnie z praktyczną zasadą ten rodzaj pamięci masowej powinien być używany tylko wtedy, gdy jesteś w stanie udowodnić, że ma to sens — nawet standardowa niezawodność i wydajność działania nie mają tutaj znaczenia. W razie jakichkolwiek wątpliwości, Twoich bądź organizacji, zdecyduj się na zupełnie inną architekturę pamięci masowej, zanim popełnisz błąd, wybierając tę i tym samym niepotrzebnie zwiększając ryzyko. Zmniejszenie wydajności działania to tylko *drobne niepowodzenie*. Dość łatwo można je naprawić już po fakcie i zwykle będzie miało znikomy wpływ, jeśli

się pomylisz. Natomiast utrata danych to *poważna awaria*, a popełniony tutaj błąd może nie prowadzić do drobnej awarii, ale do katastrofy, po wystąpieniu której będzie istniała niewielka możliwość ratunku.Wprawdzie do określenia całej architektury potrzeba nieco więcej niż tylko błędnych decyzji dotyczących pamięci masowej, ale zdalna niereplikowana pamięć masowa jest podstawowym punktem wyjścia dla popularnych projektów używanych przez producentów i resellerów. Oni zwykle używają określenia **3-2-1 architektura** względem tego, co praktycy IT nazywają **odwróconą piramidą przeznaczenia**.

Taka architektura jest tradycyjnie powszechnie wdrażana, ale jednocześnie pozostaje najmniej prawdopodobną architekturą, jaką kiedykolwiek zastosowałem w środowisku produkcyjnym. Jest wolna, skomplikowana, kosztowna i wiąże się z większym ryzykiem. Jej zastosowanie ma sens przede wszystkim w środowiskach roboczych, w których odtworzenie przechowywanych danych będzie co najwyżej czasochłonne. Ta architektura naprawdę została opracowana z myślą o typowych zadaniach nieprodukcyjnych.

Skala gigantyczna — zdalnie replikowana pamięć masowa

Ostatnia z najważniejszych omówionych tutaj architektur jest jednocześnie *największą* z nich. To zdalnie replikowana pamięć masowa. Może się wydawać, że tego rodzaju architektura pamięci masowej będzie powszechnie spotykana w przedsiębiorstwach. Wprawdzie nie jest stosowana tak rzadko, ale jednocześnie nie tak często, jak można byłoby sądzić.

Implementacja zdalnej replikowanej pamięci masowej jest kosztowna w małych rozwiązaniach i dość przystępna w dużych. Wiąże się z tym dodatkowy poziom złożoności względem rozwiązania typu RLS (co oznacza mniejszą niezawodność) i niższa wydajność działania niż rozwiązanie typu RLS bądź bezpośrednia lokalna pamięć masowa. Dlatego też takie podejście jest sensowne, gdy ograniczanie kosztu ma ważne znaczenie. To rozwiązanie zdecydowanie niszowe, choć pewien poziom niezawodności nadal będzie zapewniony.

Biorąc pod uwagę to, że dwie omówione wcześniej architektury znajdują zastosowanie w 99% wdrożeń (spotykanych przeze mnie), dla tej architektury pozostał 1%, przynajmniej jeśli chodzi o rozwiązania używane w produkcji.

Najbezpieczniejszy system nie zapewnia pełnego bezpieczeństwa

Jedna z moich bardziej dramatycznych historii z dekad pracy jako administratora systemu pochodzi z czasów, gdy jako konsultant pracowałem w bibliotece uniwersyteckiej. Razem z innymi administratorami zajmowaliśmy się obsługą ogromnych

baz danych. Administrator senior był na wakacjach i pozostał tylko administrator junior. Środowisko zawierało wysoce nadmiarowe rozwiązanie typu SAN charakteryzujące się wysoką dostępnością. Od tego rozwiązania zależała funkcjonalność wszystkich pozostałych systemów. Przygotowano wyjątkowo dużą ilość zabezpieczeń systemu SAN, m.in. począwszy od UPS-ów po generatory zapewniające nadmiarowość sprzętu. To wszystko było bardzo kosztowne.

Na mojej zmianie administrator junior postanowił przeprowadzić pewne prace konserwacyjne w rozwiązaniu typu SAN i z niewiadomego mi powodu przypadkowo wybrał opcję usunięcia wszystkich woluminów w SAN. To oczywiście był wypadek, bardzo nieostrożny błąd popełniony przez osobę przyjmującą założenie, że cały system został starannie zabezpieczony przed każdym możliwym scenariuszem niepowodzenia. Mimo to wysoka dostępność, duża nadmiarowość oraz wszelka technologia specjalna nie uchroniły przed prostym błędem popełnionym przez człowieka.

Jedno kliknięcie myszką wystarczyło, aby usunąć zawartość wszystkich systemów znajdujących się w bibliotece. Usunięte zostały aplikacje, bazy danych, konta użytkowników i dzienniki zdarzeń. Dosłownie wszystko. Działanie wszystkich systemów zależało od pojedynczego systemu pamięci masowej, który można wyłączyć, w omawianym przykładzie usunąć, w zasadzie bez żadnego wysiłku i przez osobę posiadającą dostęp do tego systemu. To można porównać do następującej sytuacji: wszystkie jajka zostają umieszczone w pojedynczym pojemniku, który wprawdzie jest wykonany niezwykle solidnie, ale ma ogromny otwór na górze, a sam pojemnik można bardzo łatwo odwrócić do góry dnem.

Sytuację jeszcze bardziej pogorszył fakt, że administrator junior nie był emocjonalnie przygotowany na taki błąd. Przerażenie możliwością utraty pracy spowodowało, że człowiek, który popełnił ten błąd, musiał zostać hospitalizowany. Wszyscy pracownicy działu informatycznego, którzy mogliby pomóc w odpowiedniej reakcji na tę awarię, zamiast zająć się złagodzeniem jej skutków, byli zaangażowani w sprowadzanie pomocy medycznej dla administratora. Niewłaściwe planowanie technologiczne plus niewłaściwe przygotowanie administratora spowodowały, że łatwa do uniknięcia katastrofa, która powinna mieć minimalny zasięg, przekształciła się w ogromną awarię. Na szczęście były tworzone dobre kopie zapasowe i system udało się przywrócić we względnie krótkim czasie. Jednak ta sytuacja pokazuje, że często wiele inwestuje się w ochronę przed awariami mechanicznymi, a jednocześnie niewiele w przeszkolenie personelu. W rzeczywistości błąd popełniony przez człowieka prawdopodobnie znacznie częściej będzie powodem katastrofy niż awaria urządzenia.

Podczas projektowania systemu pamięci masowej najtrudniejsze są kwestie związane z *architekturą pamięci masowej* i *ryzykiem*. Zagłębianie się w szczegóły związane z systemem plików na ogół zapewnia niezłą zabawę i wiąże się z minimalnym ryzykiem dla administratora systemu. W sytuacji, w której najlepszym rozwiązaniem

będzie system plików BtrFS, a Ty wybierzesz EXT4, negatywne skutki tej decyzji będą minimalne. Okażą się na tyle małe, że prawdopodobnie nikt nigdy się nie dowie o podjęciu niedoskonałej decyzji. Natomiast wybór niewłaściwej architektury pamięci masowej może prowadzić do ogromnego kosztu, długich przestojów bądź też utraty ogromnej ilości danych.

Naprawdę trzeba poświęcić nieco czasu na zrozumienie potrzeb biznesowych i tych związanych z wykonywanymi zadaniami. Konieczne jest określenie akceptowanego poziomu ryzyka, oczekiwanego poziomu wydajności działania, sposobu spełnienia wymagań przy optymalnym koszcie. Jeżeli nie znasz odpowiedzi na te pytania, musisz je znaleźć.

Najlepszą praktyką jest jak zwykle poświęcenie czasu na poznanie wszystkich potrzeb biznesowych, poznanie wszystkich dostępnych czynników i wykorzystanie tej wiedzy. Jednak w praktyce to okazuje się bardzo trudne do zrobienia. Przydatnych więc będzie kilka reguł, które mogą w tym pomóc.

Najlepsze praktyki dotyczące pamięci masowej

Próba przełożenia pamięci masowej na zbiór **najlepszych praktyk** jest raczej trudnym zadaniem. Ogólnie rzecz biorąc: za najważniejszą regułę dotyczącą pamięci masowej można uznać to, że nie istnieje droga na skróty. Konieczne jest zrozumienie wszystkich aspektów infrastruktury pamięci masowej, poznanie wykonywanych zadań, a także wykorzystanie wiedzy z uwzględnieniem ryzyka i idealnego poziomu obciążenia.

Idąc dalej, najlepsze praktyki można przedstawić następująco:

- RAID. Jeżeli dane są warte ich przechowywania, dobrze jest wykorzystać macierz RAID (lub RAIN). Jeżeli zastanawiasz się nad wartością posiadania macierzy RAID (absolutne minimum) w serwerach, ponownie się zastanów nad zasadnością przechowywania danych.
- RAID (kwestie fizyczne). Implementacje sprzętowe i programowe macierzy RAID są równie dobre. Ustal, które czynniki najlepiej się sprawdzają w przypadku Twoich potrzeb.
- LVM. Podobnie jak z ogólną wirtualizacją, która zostanie dokładnie omówiona w dalszej części książki, tak samo wirtualizacja pamięci masowej nie wiąże się z dostarczeniem konkretnego zestawu funkcjonalności potrzebnej już pierwszego dnia działania systemu. Zamiast tego dotyczy dostarczenia mechanizmów zapewniających ochronę przed nieznanym oraz elastyczność niezależnie od tego, co się zdarzy w przyszłości. O ile nie jesteś w stanie przedstawić niewiarygodnie silnych argumentów przeciwko rozwiązaniu typu LVM, korzystaj z tej technologii.

- System plików. Nie daj się złapać na krzykliwą reklamę bądź trendy. Dokładnie określ funkcjonalność, która dzisiaj ma dla Ciebie największe znaczenie oraz zapewni Ci w przyszłości ochronę przed nieznanym. Zdecyduj się na system plików, który jest niezawodny, solidny i doskonale przetestowany. Wybierz taki system, który zapewni Ci komfort pracy i na dłuższą metę nie będzie Ci sprawiał problemu.
- Architektura pamięci masowej. O ile nie masz absolutnej pewności, że jest wymagane bądź też (z perspektywy finansowej) zaleca się użycie zdalnego systemu pamięci masowej, pozostań wyłącznie przy lokalnej pamięci masowej. Ponadto jeśli nie jesteś w stanie potwierdzić wyraźnych korzyści płynących z replikacji węzła, nie stosuj jej. Proste jest lepsze niż skomplikowane.

Jako administrator systemu rzadko będziesz mieć do czynienia z decyzjami projektowymi dotyczącymi pamięci masowej. Niezależnie od częstotliwości podejmowania takich decyzji, w przeciwieństwie do innych rodzajów decyzji te mają ogromny wpływ na długoterminowe działanie systemu. Poświęć czas i starannie ustal, co jest niezbędne do wykonywania danego zadania.

Przykład pamięci masowej

Warto wykonać krok wstecz i połączyć ze sobą wszystkie informacje zamieszczone w rozdziale, aby je wykorzystać w rzeczywistym scenariuszu. Nie mogę w tym miejscu podać rozsądnych przykładów dla każdego powszechnie spotykanego scenariusza, nie mówiąc już o tych mniej prawdopodobnych. Jednak mogę przedstawić ogólne informacje podsumowujące materiał zamieszczony w rozdziale, aby w ten sposób jakoś go usystematyzować.

W celu zachowania maksymalnej prostoty postaram się przedstawiać jak najbardziej ogólne informacje dotyczące najczęściej spotykanych konfiguracji w sektorze małych i średnich przedsiębiorstw. To prawdopodobnie są konfiguracje, z którymi będziesz się spotykać w pracy administratora systemu.

W małych przedsiębiorstwach ogólną korzyścią będzie zachowanie maksymalnej prostoty projektu. Brak ogromnego i doświadczonego personelu, który często jest podatny na ryzyko rotacji, oznacza, że w małych przedsiębiorstwach preferowane są systemy wymagające mniejszej obsługi technicznej. Taka obsługa może być łatwo zapewniona przez konsultantów bądź pracowników, którzy nie mają dogłębnej wiedzy o danym środowisku.

W tego rodzaju środowiskach, a także w wielu większych, ważne znaczenie ma sprzętowa macierz RAID pozwalająca na wymianę dysków w trakcie działania systemu. Dzięki temu konserwacja sprzętu będzie mogła być prowadzona przez specjalistów,

bez konieczności wykonywania zadań typowych dla administratorów systemów. To zaczyna nabierać krytycznego znaczenia w przypadku centrów danych lub stosowania innych zdalnych rozwiązań. W takich okolicznościach pracownik działu informatycznego być może nie będzie mógł mieć fizycznej możliwości zaangażowania się w rozwiązywanie problemu.

Należy rozpocząć od przyjęcia pewnych ogólnych założeń. Warstwa fizyczna zawiera osiem dysków twardych. To mogą być tradycyjne dyski twarde, napęd SSD, napędy NVMe bądź też dowolne urządzenia blokowe. Tak naprawdę to nie ma znaczenia. Natomiast znaczenie ma to, że istnieje wiele napędów, a chcemy je widzieć w postaci pojedynczej jednostki.

To oznacza dodanie sprzętowego kontrolera macierzy RAID. Do tego kontrolera zostaną podłączone wszystkie napędy, a następnie skonfigurowane z użyciem odpowiedniego poziomu macierzy RAID. Można użyć dowolnego poziomu macierzy, przy czym w omawianym przykładzie przyjmujemy założenie o zdefiniowaniu macierzy RAID 10.

Od samego początku omawianego tutaj przykładu, nawet bez zainstalowania systemu Linux lub czegokolwiek innego tego rodzaju, wykorzystaliśmy sprzęt do implementacji dwóch pierwszych warstw pamięci masowej! To są urządzenia fizyczne i pierwsza warstwa abstrakcji.

Nie omawiam tutaj rzeczywistego wkładania dysków do komputera, ponieważ to proces typowo manualny i istnieje spore prawdopodobieństwo, że dostawca serwera już się tym zajął.

Jeżeli chodzi o konfigurowanie karty kontrolera macierzy RAID: poszczególne produkty w tej kategorii są zwykle odmienne, natomiast podstawy pozostają takie same, a zadanie do wykonania zawsze jest niezwykle proste. O to właśnie chodzi w kontrolerze macierzy RAID — zmniejszenie do absolutnego minimum ilości wymaganej wiedzy i planowania niezbędnych do przeprowadzenia operacji związanych z kartą macierzy RAID, zarówno podczas jej początkowego uruchamiania, jak i później w czasie jej codziennego działania. W omawianym tutaj przykładzie i w celu zachowania prostoty przyjmujemy założenie, że mamy do czynienia z pojedynczą macierzą. To *nie* jest macierz przeznaczona do uruchomienia systemu operacyjnego, co pozwoli na znacznie łatwiejsze przedstawienie wybranych kroków koniecznych do wykonania w powłoce. To tylko jest przykład.

Pamiętaj, że poszczególne kontrolery macierzy RAID są różne w zależności od producentów, a potencjalnie także różnią się między poszczególnymi modelami danego produktu. Dlatego też zawsze trzeba znać dokładny sposób działania konkretnego produktu.

To w tym miejscu należy dodać, że w kontrolerze macierzy RAID poszczególne napędy są dołączone jako urządzenia blokowe. To unikatowy przypadek, w którym interfejs urządzenia blokowego udający fizyczny dysk twardy w rzeczywistości jest nim. We wszystkich kolejnych przykładach będę używać oprogramowania do implementacji interfejsu dysku twardego, przy czym przedstawiony dysk będzie logiczny, a nie fizyczny:

```
=>ctrl slot=0 create type=ld drives=2I:1:5,2I:1:6,2I:1:7,2I:1:8
raid=1+0
```

To jest faktycznie stosowana składnia w rzeczywistym kontrolerze macierzy RAID. Tego rodzaju zadanie zwykle wykonuje się graficznie za pomocą narzędzia wyposażonego w graficzny interfejs użytkownika. Jednak czasami zachodzi potrzeba użycia narzędzia działającego w powłoce. Gdy tylko istnieje możliwość, staram się pracować w powłoce. Zapewnia ona większą powtarzalność, a ponadto dokumentowanie rozwiązania staje się łatwiejsze.

Po przejściu fazy początkowej konfiguracji sprzętowej można przystąpić do przeprowadzenia dalszej konfiguracji w systemie Linux.

Kontroler sprzętowej macierzy RAID zwykle stosuje własną konwencję nazw w systemie plików */dev*. W omawianym przykładzie kontroler używa składni `cciss` i w ramach tego systemu tworzy urządzenie `c0d1`. To wszystko będzie się różniło w zależności od używanego kontrolera i konfiguracji.

Następnym krokiem jest utworzenie warstwy woluminu logicznego na warstwie macierzy RAID, aby w ten sposób zapewnić elastyczność w systemie pamięci masowej. Trzeba rozpocząć od dodania nowo utworzonego urządzenia jako *urządzenia fizycznego* do systemu LVM Linuksa. W tym celu można wykorzystać polecenie `pvcreate` i podać ścieżkę dostępu do nowego urządzenia:

```
# pvcreate /dev/cciss/c0d1
```

To bardzo krótka i łatwa operacja. Po jej przeprowadzeniu podsystem LVM wie o istnieniu nowego urządzenia macierzy RAID. Oczywiście system LVM wie o istnieniu tego urządzenia i traktuje je jako blokowe. Formalnie rzecz biorąc, to jest macierz RAID, system LVM nie jest w stanie tego wykryć i tak naprawdę to nie ma żadnego znaczenia. W tym miejscu najważniejsze jest użycie abstrakcji, którą będzie można zastosować niezależnie od rodzaju urządzenia fizycznego. Szybkość działania, pojemność i kwestie związane z zapewnieniem bezpieczeństwa są hermetyzowane na warstwie macierzy RAID. Pracując dalej nad tym przykładem, można traktować pamięć masową jako zwykły dysk twardy.

Kolejnym interesującym aspektem jest tutaj to, że podczas używania sprzętowego kontrolera macierzy RAID ta abstrakcyjna i wirtualizowana reprezentacja dysku twardego to jedynie napęd logiczny, ale urządzenie blokowe jest w rzeczywistości fizyczne. Oszołamiające, wiem. To naprawdę sprzęt, ale nie napęd sprzętowy. Zastanów się nad tym przez chwilę.

W jaki sposób macierz RAID jest zarządzana przez LVM, aby było możliwe dodanie napędu do grupy woluminu? W omawianym przykładzie dodajemy napęd do nowej grupy woluminu tworzonej na potrzeby tego rozwiązania. To nowa grupa otrzymuje nazwę `vg1`. Spójrz na przykład polecenia, które należy wydać w powłoce:

```
# vgcreate vg1 /dev/cciss/c0d1
```

W ten sposób otrzymujemy pewne urządzenie. Pojemność poszczególnych fizycznych dysków twardych zostaje połączona przez kontroler macierzy RAID w pojedynczy logiczny napęd zarządzany przez LVM. W ten sposób powstaje pula pojemności, czyli **grupa woluminu**, którą można dzielić na użyteczne fragmenty o pojemności niezbędnej dla zadania wykonywanego przez dany serwer.

Następnie w utworzonej grupie woluminu trzeba przystąpić do utworzenia rzeczywistych *woluminów logicznych*. Pamiętaj, że woluminy logiczne zastąpiły partycje jako podstawowy sposób dzielenia urządzenia blokowego na mniejsze fragmenty. W prezentowanym tutaj rozwiązaniu zostanie użyte najprostsze możliwe podejście — system LVM otworzy tylko jeden wolumin logiczny o maksymalnej pojemności. To oznacza użycie 100% dostępnej pojemności grupy woluminu (obecny poziom wykorzystania wynosi 0%, ponieważ to będzie pierwsza operacja przeprowadzona w tej grupie woluminu):

```
# lvcreate -l 100%FREE -n lv_data vg1
```

Wymienione polecenie nakazuje utworzenie nowego woluminu logicznego obejmującego całą wolną przestrzeń w grupie woluminu `vg1` i nadanie mu nazwy `lv_data`. Na tym koniec. W ten sposób został utworzony wolumin logiczny, którego można używać. Istnieje możliwość utworzenia mniejszego woluminu logicznego, np. o pojemności wynoszącej 50% dostępnej przestrzeni, a następnie utworzenie drugiego, który będzie obejmował 100% pozostałej wolnej przestrzeni. W ten sposób powstaną dwa woluminy logiczne o identycznej wielkości.

Pamiętaj, że system LVM, taki jak istniejący w Linuksie, zapewnia większą elastyczność niż w przypadku bezpośredniego tworzenia systemów plików w napędach fizycznych bądź nawet w napędach wirtualnych dostarczonych przez sprzętowy kontroler macierzy RAID. System LVM pozwala np. na dodawanie większej liczby urządzeń fizycznych do grupy woluminu, co pozwala otrzymać większą pojemność dla tworzonych woluminów logicznych. Zyskujesz także możliwość zmiany wielkości (w górę bądź w dół) poszczególnych woluminów logicznych. System LVM pozwala też na tworzenie migawki woluminu logicznego, co okazuje się niezwykle użyteczne podczas tworzenia mechanizmu kopii zapasowej bądź przygotowania systemu do ryzykownej modyfikacji, którą będzie można łatwo wycofać. LVM oferuje bardzo ważną funkcjonalność.

Po utworzeniu grupy woluminu `lv_data` w większości przypadków trzeba ją sformatować i utworzyć system plików, aby można było zacząć z niej korzystać. Sformatujemy ją z użyciem systemu plików XFS. Obecnie to najczęściej zalecany system plików do ogólnego przeznaczenia:

```
# mkfs.xfs /dev/vg1/lv_data
```

Bardzo proste. Po upływie kilku sekund otrzymasz w pełni sformatowany wolumin logiczny. Sformatowanie systemu plików powoduje zatrzymanie łańcucha interfejsów urządzenia blokowego. W tym momencie przedstawiony będzie interfejs systemu plików pozwalający na używanie pamięci masowej w standardowy sposób zamiast urządzenia blokowego, które jest używane przez komponenty systemu pamięci masowej.

Ostatnim koniecznym do wykonania krokiem jest zamontowanie nowego systemu plików w katalogu, aby można było z niego korzystać:

```
# mkdir /data
# mount /dev/vg1/lv_data /data
```

I to tyle. W ten sposób została zaimplementowana składająca się z wielu warstw abstrakcja, bazująca na systemie pamięci masowej i przeznaczona do użycia w jednym z najczęściej spotykanych scenariuszy. Otworzyliśmy system plików XFS na podstawie zarządcy woluminu logicznego (LVM), istniejącego na bazie sprzętowego kontrolera macierzy RAID obejmującego wiele fizycznych napędów dyskowych.

Skoro każda warstwa używa interfejsu urządzenia blokowego, można łączyć i dopasowywać wiele funkcjonalności dodatkowych. Przykładem może być tutaj użycie dwóch kontrolerów macierzy RAID i połączenie ich pojemności w jednej grupie woluminu. Ewentualnie utworzenie wielu grup woluminów. Można mieć wiele woluminów logicznych. Programową macierz RAID (w systemie Linux nosi nazwę MD RAID) można wykorzystać do utworzenia macierzy RAID na podstawie dwóch fizycznych kontrolerów! Możliwości są praktycznie niczym nieograniczone, jedynym ograniczeniem pozostaje praktyczne zastosowanie danego rozwiązania.

W tym momencie po wydaniu polecenia `cd /data` możesz już używać nowego systemu plików, jakby zawsze tam był. To jest nowy system plików zbudowany na bazie tych wszystkich warstw abstrakcji, a wiele urządzeń fizycznych powoduje, że cała „magia" na tym etapie jest całkowicie ukryta przed użytkownikiem. To po prostu działa.

W przeszłości, np. jakby to był 2004 r., w tym miejscu zatrzymałbym się i powiedział, że opisałem prawdziwy serwer, który będzie się tak przedstawiał, jeśli wszystko zostanie poprawnie zaimplementowane. Jednak to nie jest 2004 r., więc trzeba naprawdę się zatrzymać i wyjaśnić, jak nasza pamięć masowa będzie używana w większości przypadków. Obecnie zastanawiamy się, jak warstwy wirtualizacji będą stosować tę pamięć masową, ponieważ w tym miejscu sytuacja naprawdę robi się interesująca.

Przyjąłem założenie, że utworzony przed chwilą system plików */data* będzie używany do przechowywania obrazów dysków kilku maszyn wirtualnych. Oczywiście te obrazy dysków to po prostu poszczególne pliki znajdujące się w systemie plików. To bardzo proste założenie. Niczym się nie różni od utworzenia i przechowywania np. plików tekstowych w */data* (poza tym, że obrazy w maszynach wirtualnych są o wiele większe).

Świetną cechą obrazu dysku (takiego jak QCOW, VHD, ISO itd.) jest to, że znajduje się w systemie plików, ale po otworzeniu za pomocą specjalnego sterownika, który potrafi go odczytywać, ponownie przedstawia interfejs blokowy. To prawda. Przebyliśmy drogę z bloku do bloku do bloku do bloku do systemu plików i ponownie do bloku. W niektórych przypadkach nawet może nie być używany hipernadzorca. Natomiast tego nowego pliku urządzenia blokowego można używać gdzieś w zwykłym serwerze. Tego rodzaju rozwiązanie jest powszechnie spotykane w systemie Windows, który w pewnych okolicznościach przekazuje pliki typu VHD. Z kolei w systemie macOS takie rozwiązanie wykorzystuje się powszechnie do tworzenia pakietów instalacyjnych oprogramowania. Natomiast w systemie Linux to rozwiązanie jest znacznie rzadziej spotykane, ale mimo to dostępne.

Przyjmując założenie o robieniu czegoś normalnego, mamy pewność co do wykorzystania hipernadzorcy, którym niemal na pewno będzie KVM, a uruchamiane w nim maszyny wirtualne będą używały plików obrazów pamięci masowej w nowo przygotowanym systemie plików. W takim przypadku większość omówionych tutaj procesów i operacji będzie przeprowadzona ponownie, tym razem wewnątrz maszyny wirtualnej.

Niektóre fragmenty nie będą wymagały ponownego odtworzenia. Na przykład napędami fizycznymi już zarządza fizyczny kontroler macierzy RAID. Szybkość, pojemność i niezawodność pamięci masowej zostały już określone przez system i nie trzeba tego tutaj powielać. Standardowe podejście polega na przedstawieniu systemowi operacyjnemu uruchomionemu w maszynie wirtualnej pojedynczego pliku obrazu dysku jako pojedynczego urządzenia blokowego. Nasz system operacyjny hipernadzorcy nie inaczej postrzega urządzenie blokowe ze sprzętowego kontrolera macierzy RAID.

W maszynie wirtualnej często będą wykonywane te same zadania. Urządzenie blokowe zostaje dodane jako wolumin fizyczny LVM. Następnie ten wolumin fizyczny zostaje dodany do grupy woluminów. Kolejnym krokiem jest utworzenie na podstawie tej grupy jednego lub więcej woluminów logicznych. Następnie wolumin logiczny zostaje sformatowany z użyciem wybranego systemu plików oraz zamontowany w katalogu. Oczywiście to najczęściej nie odbywa się ręcznie, jak przedstawiłem w rozdziale — te zadania są zwykle automatyzowane przez proces instalacyjny.

Używając MD RAID, można dodać kolejne kroki. Istnieje również możliwość użycia mniejszej liczby kroków, np. przez całkowite pominięcie LVM. Wszystkie te same kroki można wykonać dla pojedynczego napędu dysku twardego bez użycia kontrolera macierzy RAID. Wprawdzie pod względem fizycznym takie rozwiązanie będzie miało mniejsze możliwości, ale wszystkie przedstawione przykłady będą działały tak samo na poziomie technicznym. LVM można używać w fizycznym komputerze, ale nie w maszynach już wirtualnych. Elastyczność zapewnia nam ogromne możliwości w tym zakresie. Trzeba poznać sposób działania urządzeń blokowych, tworzenie systemów plików w urządzeniach blokowych, a także wiedzieć, jak pliki blokowe pozwalają skonwertować system plików z powrotem na postać urządzenia blokowego.

Abstrakcja i hermetyzacja to w naszym arsenale narzędzia o potężnych możliwościach, które rzadko są tak bardzo namacalne.

Podsumowanie

Jeżeli wytrwałeś do końca tego rozdziału i wciąż ze mną jesteś, to gratuluję, udało Ci się! Pamięć masowa to ważny temat podczas administrowania systemem. Prawdopodobnie żaden inny obszar, którym musisz zarządzać, nie będzie miał aż tak dużej wartości dla Twojej organizacji.

Omówiłem podstawy pamięci masowej, opierając się na koncepcjach interfejsów urządzeń blokowych, technik abstrakcji, systemów plików i ich interfejsów. Wykorzystałem te koncepcje do przedstawienia nadmiarowości pamięci masowej oraz wyjaśnienia, jak tworzyć złożone i niezawodne magazyny danych. Ponadto wyjaśniłem, jak można obsłużyć dostęp do pamięci masowej w różnych urządzeniach, aby w ten sposób zapewnić spełnienie wymagań. Moim celem było dostarczenie wiedzy niezbędnej do samodzielnego zastanowienia się nad własnymi potrzebami z zakresu pamięci masowej dla danego zadania. Przedstawiłem również dostępne technologie i wyjaśniłem, jak można je stosować w celu jak najefektywniejszego spełnienia tych wymagań.

Nigdy więcej nie musisz postrzegać pamięci masowej jako magicznego czarnego pudełka bądź też uciążliwego zadania, z którym jesteś zmuszony się zmagać. Zamiast tego możesz patrzeć na pamięć masową jako na możliwość do zabłyśnięcia i pokazania, jak mogą być zastosowane poprawne najlepsze praktyki z zakresu administrowania systemem. To pozwoli na maksymalizację tych czynników pamięci masowej, które mają największe znaczenie dla wykonywanych przez Ciebie zadań. Unikniesz tym samym wyrzucania pieniędzy w błoto na niepotrzebny sprzęt bądź jeszcze gorzej: zignorowania tematu sprzętu i żywienia nadziei, że uda ci się znaleźć inną pracę, zanim bieżące środowisko, którym się zajmujesz, się rozleci.

W następnym rozdziale przedstawię architekturę systemu na jeszcze wyższym poziomie. Wiele interesujących koncepcji z tego rozdziału pojawi się ponownie w następnym. Architektura systemu w dużej mierze zależy od architektury pamięci masowej, współdzielonych jest wiele paradygmatów nadmiarowości i ochrony systemu. Za niezwykle ekscytujące można uznać sprawdzenie, jak dobre elementy projektu pamięci masowej mogą prowadzić do uzyskania prawdziwie wysokiej wydajności, wysokiej dostępności i powstania ostatecznego rozwiązania, które będzie efektywne pod względem kosztów.

Zostań mistrzem wśród adminów Linuksa!

Popularność systemów linuksowych cały czas rośnie. Mimo to bardzo niewielu administratorów stara się dokładnie opanować sztukę zarządzania Linuksem, większość ogranicza się do kilku rutynowych zadań. Tak administrowane systemy oczywiście mogą dłuższy czas działać poprawnie, jednak dopiero dogłębne zapoznanie się ze sposobem działania Linuksa pozwoli na pełniejsze skorzystanie z jego niesamowitych możliwości.

To książka przeznaczona dla profesjonalnych administratorów i użytkowników Linuksa. Dzięki niej szybciej zrozumiesz, w jakim stopniu dobre zarządzanie systemami na poziomie systemu operacyjnego może wynieść działanie infrastruktury biznesowej na zupełnie inny poziom. Znajdziesz tu najlepsze praktyki zarządzania systemami — począwszy od wyboru optymalnej dystrybucji Linuksa, poprzez zaprojektowanie architektury systemu, skończywszy na strategiach zarządzania przeprowadzanymi w nim poprawkami i aktualizacjami. Sporo miejsca poświęcono różnym metodom automatyzacji części zadań administratora, a także schematom tworzenia kopii zapasowych i odzyskiwania danych po awarii. Zaproponowano również ciekawe podejście do rozwiązywania problemów, dzięki któremu można szybciej uzyskać satysfakcjonujące rozwiązanie i uniknąć poważniejszych szkód.

Najciekawsze zagadnienia:

- kim jest administrator systemu i znaczenie tej roli w organizacji
- ocena ryzyka podczas administrowania systemami
- najlepsze praktyki w pracy z technologiami związanymi z Linuksem
- nadawanie priorytetów i prowadzenie segregacji
- planowanie katastrofy i procedury odzyskiwania po awarii

Scott Alan Miller od ponad 25 lat zajmuje się systemami UNIX i Linux. Pracował w firmach o różnej wielkości, w wielu branżach; był technikiem, liderem, menedżerem, konsultantem, autorem, prelegentem i mentorem. Od ponad 20 lat kieruje zespołem IT w NTG. Mieszka w Nikaragui.

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl
KOD KORZYŚCI
Sięgnij po więcej!
ISBN 978-83-289-0071-4
9 788328 900714
Cena: 89,00 zł
Packt