Modyfikowanie Minix File System w celu optymalizacji korzystania z maszyn wirtualnych

Łukasz Raszkiewicz
lr371594@students.mimuw.edu.pl

1. PROBLEM

Obrazy maszyn wirtualnych mogą zawierać partycje, które nie są do końca zapełnione. Taka partycja będzie zazwyczaj zawierała bloki wypełnione samymi zerami. Dopóki nic w tym miejscu faktycznie nie zapiszemy, zajmuje ono niepotrzebne miejsce na naszym (fizycznym) dysku twardym, a odczytywanie z niego bloków samych zer to niepotrzebne, powolne operacje odczytu danych z dysku.

1.1 Czy tak faktycznie jest?

Powyższe założenia wydają się oczywiste, ale warto sprawdzić, czy taka sytuacja faktycznie występuje w obrazach maszyn wirtualnych. W tym celu zbadałem plik `minix.img` udostępniony w scenariuszach do Systemów Operacyjnych. Podzieliłem ten obraz na 5120 części o rozmiarze 1 MB. Aż 3044 z nich były w pełni wypełnione zerami. Zatem odpowiednia modyfikacja systemu plików mogłaby zaoszczędzić około 60% miejsca zajmowanego przez ten obraz.

2. IMPLEMENTACJA

Implementacja funkcji `rw_chunk` służącej do oczytywania i zapisywania danych nie większych niż blok w Minix File System już zawiera taką funkcjonalność, że jeżeli próbujemy odczytać coś z nieistniejącego bloku, to zwracane są same zera. Za to gdy zachodzi próba zapisania danych do nieistniejącego bloku, tworzony jest nowy blok i dane są zapisywane do niego, co wymaga kosztownych operacji korzystania z dysku twardego.

Moja łatka do Minix File System zmienia właśne to, co dzieje się przy zapisywaniu do nieistniejącego bloku. W takiej sytuacji sprawdzam, czy zapisywane dane to same zera. Jeżeli tak, to po prostu nic nie robię — nie tworzę nowego bloku, nie zapisuję nic na dysk. Wtedy próba odczytania takich danych zakończy się zwróceniem samych zer, ponieważ będzie to próba czytania z nieistniejącego bloku.

Taka implementacja zapewnia, że przy zapisywaniu bloku pełnego zer nie oznaczymy żadnego bloku na dysku twardym jako zajętego. Dzięki temu miejsce na dysku, które byłoby zajęte przez pełen zer fragment pliku, może zostać wykorzystane na przechowywanie innych plików. Poza tym zmniejsza się liczba zajmujących dużo czasu operacji zapisu danych na dysk.

3. WYDAJNOŚĆ

Skrypt `prepare_tests.sh` generuje (poprzez skompilowanie i uruchomienie `gen_test_files.c`) trzy pliki:

- `only_zeroes` — każdy bajt to `0`
- `only_ones` — każdy bajt to `1`
- `sandwich` — pierwsze i ostatnie ćwierć pliku to same bajty `1`, środkowe pół to tylko bajty `0`

3.1 test_generate.sh

Poniżej znajduje się czas (w sekundach) działania programu generującego te trzy pliki o rozmiarze 128 MB na maszynie wirutalnej ze zmodyfikowanym systemem plików oraz na takiej z oryginalnym MFS.

	MFS z modyfikacją	**Oryginalny MFS**
Próba 1	0.96	1.26
Próba 2	0.95	1.25
Próba 3	0.93	1.25
Próba 4	0.93	1.26
Próba 5	0.93	1.26
Średnia	**0.94**	**1.26**

W tym czasie wykonywane są też operacje generowania zawartości tych plików, a nie tylko ich zapisu. Dlatego też zaobserwowane przyspieszenie nie jest tak duże, jak będzie w następnym teście, ale pokazuje, że zastosowanie łatki może zmniejszyć czas działania faktycznych programów (np. służących do generowania pustego obrazu maszyny wirtualnej).

3.2 test_copy.sh

Teraz porównam wydajność kopiowania plików na zmodyfikowanym MFS. Kopiowanie plików opiera się w większości na operacjach czytania i zapisywania na dysk, zatem szybkość działania systemu plików powinna być czynnikiem istotnie wpływającym na czas kopiowania. Jeżeli łatka faktycznie działa, plik `only_zeroes` powinien kopiować się szybciej niż `sandwich`, a ten szybciej niż `only_ones`, ponieważ zawierają one kolejno coraz mniej fragmentów z samymi zerami. W teście korzystam ze zdefiniowanych wcześniej plików o rozmiarze 256 MB.

	`only_zeroes`	`sandwich`	`only_ones`
Próba 1	0.21	0.61	0.96
Próba 2	0.23	0.60	0.96
Próba 3	0.21	0.60	1.00
Próba 4	0.23	0.60	0.95
Próba 5	0.23	0.60	0.95
Średnia	**0.22**	**0.60**	**0.96**

Testy pokazują ogromne przyspieszenie operacji na plikach, które mają dużo fragmentów z samymi bajtami `0`. Kopiowanie pliku w całości zerowego jest ponad czterokrotnie szybsze niż kopiowanie pliku, który w ogóle nie ma fragmentów zerowych.

3.3 test_only_ones.sh

Należy jednak pamiętać, że do systemu plików zostały dodane dodatkowe operacje. Jak już zostało sprawdzone, przyspieszają one operacje na plikach z blokami zerowymi, ale dodany kod mógłby spowolnić pracę na plikach, które tych zer nie mają. W celu sprawdzenia, czy faktycznie jest taki efekt, zmierzyłem czas kopiowania pliku bez żadnych bloków zerowych na systemie ze zmodyfikowanym systemem plików oraz na systemie z oryginalnym MFS.

	MFS z modyfikacją	**Oryginalny MFS**
Próba 1	0.98	0.95
Próba 2	0.98	0.93
Próba 3	0.96	0.96
Próba 4	0.96	0.95
Próba 5	0.96	0.95
Próba 6	0.96	0.95
Próba 7	0.98	0.96
Próba 8	0.96	0.93
Próba 9	0.96	0.93
Próba 10	1.00	0.95
Średnia	**0.970**	**0.946**

W powyższych czasach można zaobserwować nieznaczny wzrost czasu kopiowania w zmodyfikowanym systemie plików. Trudno jednak określić, czy jest to faktyczny spadek wydajności, czy tylko błąd pomiaru.

Istnieje teoretyczna możliwość spreparowania złośliwego pliku, który będzie zawierał tylko bloki mające same zera oraz inny bajt na końcu. Zapisując taki plik w zmodyfikowanym MFS będzie trzeba go przejrzeć w całości, gdyż dodana tam pętla nie przerwie się przedwcześnie po znalezieniu niezerowego bajtu. Jednak taka sytuacja nie powinna się zdarzyć w realistycznych zastosowaniach systemu plików i uważam, że nie ma sensu jej testować. Nawet gdybyśmy z jakiegoś powodu musieli zapisać taki plik, to wpływ tych dodatkowych operacji na wydajność powinien być minimalny - potrzebny jest tylko procesor i pamięć operacyjna, a głównym wąskim gardłem są operacje zapisu na dysk.

4. WNIOSKI

Wprowadzenie wcześniej opisanej modyfikacji do Minix File System może istotnie poprawić wydajność operacji na plikach zawierających dużo bloków zerowych i do tego pozwoli zaoszczędzić miejsce na dysku dla takich plików. Szczególnie znanym przypadkiem takich plików są obrazy maszyn wirtualnych i zastosowanie mojej łatki na MFS powinno być opłacalne na dyskach przechowujących takie obrazy.

Stosowanie łatki na dyskach, które nie przechowują tego typu plików, nie powinno sprawiać żadnych problemów (być może nieznaczną utratę wydajności), ale też nie będzie miało żadnych pozytywnych efektów.

Potencjalnym problemem takiego rozwiązania jest to, że podczas korzystania z maszyny wirtualnej może skończyć się miejsce na partycji fizycznego dysku twardego, ponieważ puste miejsce na dysku maszyny wirtualnej nie jest wcześniej zadeklarowane. Może to doprowadzić do uszkodzenia obrazu maszyny.

Warto też zauważyć, że nie wszystkie formaty obrazów maszyn wirtualnych będą przechowywały zerowe bloki w obrazie. Sprawdziłem obrazy moich maszyn VirtualBoxa z Sieci Komputerowych i nie zawierały one żadnych bloków zerowych. Zatem przed zastosowaniem łatki należy sprawdzić, czy będzie ona przydatna dla obrazów z używanego programu do wirtualizacji.