Politechnika
Śląska

PROJEKT INŻYNIERSKI

System do obsługi pracowni informatycznych z wykorzystaniem maszyn wirtualnych

Wojciech JANOTA
Nr albumu: ⟨**290357**⟩

Kierunek: ⟨Informatyka⟩
Specjalność: ⟨Bazy Danych i Inżynieria Systemów⟩

PROWADZĄCY PRACĘ
⟨**dr inż. Błażej Adamczyk**⟩
KATEDRA ⟨**Katedra Sieci i Systemów Komputerowych**⟩
Wydział Automatyki, Elektroniki i Informatyki

Gliwice 2023

Tytuł pracy

System do obsługi pracowni informatycznych z wykorzystaniem maszyn wirtualnych

Streszczenie

Tematem pracy jest stworzenie systemu do zarządzania pracowniami informatycznymi używając możliwości oferowanych przez maszyny wirtualne. Proponowana solucja umożliwia zdalne, zautomatyzowane przygotowanie pracowni informatycznej do zajęć lekcyjnych, minimalizując wymaganą liczbę manualnych kroków.

Słowa kluczowe

automatyzacja,PXE,Python,Ansible,Linux,QEMU

Thesis title

A system for upporting IT labs with the use of virtual machines

Abstract

Topic of this thesis is a automated system for managing IT labs using virtual machines. Solution allows for remote, automated preparation of the lab for classes, minimizing number of manual steps required.

Key words

automation,PXE,Python,Ansible,Linux,QEMU

Spis treści

Rozdział 1

Wstęp

Wprowadzenie Jednym z wielu problemów, z jakimi musi zmierzyć się administrator systemów (na przykład w placówce edukacyjnej), jest konfiguracja i zarządzanie flotą wielu maszyn, użytkowanych często przez osoby nietechniczne. Powstało wiele narzędzi służących do ułatwienia tego zadania, wśród których zwrócono uwagę na kilka przykładów:

- Microsoft Active Domain jako system do zarządzania już skonfigurowanymi maszynami opartymi o system Microsoft Windows
- Ansible jako system do zarządzania maszynami opartymi o systemy z rodziny UNIX
- Ubuntu Landscape jako system służący do zarządzania flotą maszyn opartych o system Ubuntu Linux
- Microsoft Windows Unattended Install, czyli narzędzie służące do automatycznego konfigurowania instalacji systemu Microsoft Windows

Każde z tych narzędzi pozwala na zautomatyzowanie jednego z podstawowych kroków w procesie zarządzania i utrzymania pracowni informatycznej: instalacji systemu/systemów operacyjnych, konfiguracji systemu, aktualizacji i utrzymania systemu. Pewne przypadki użycia wymagają jednak pewnych cech, które bardzo trudno osiągnąć używając powyższych narzędzi. Przykładem takiej sytuacji jest resetowanie urządzeń po każdych zajęciach lekcyjnych, zapewniając przy tym, że każde środowisko na którym pracują uczniowie/studenci jest identyczne. W takiej sytuacji najczęściej wykorzystuje się maszyny wirtualne, które po zakończeniu zajęć są przywracane do migawki, lub ich obraz dysku jest podmieniany na oryginalny. Wymaga to jednak kilku manualnych kroków, które należy wykonać pomiędzy zajęciami.

Proponowane rozwiązanie automatyzuje proces dystrybucji obrazów maszyn wirtualnych, instalacji oraz konfiguracji systemu operacyjnego, pod kontrolą którego będą pracować maszyny wirtualne. Celem tej pracy było:

- napisanie programu serwera obrazów maszyn wirtualnych, którego zadaniem jest ich rejestrowanie, przypisywanie oraz dystrybuowanie
- stworzenie klienta synchronizującego stan maszyny klienckiej ze stanem obecnym na serwerze, pobierającego obrazy, wyświetlającego ekran wyboru systemy do uruchomienia oraz obsługującego ich uruchmianie poprzez mechanizm QEMU wraz z KVM
- przygotowanie konfiguracji dla serwera opartego o system operacyjny Linux:
  - do automatycznego instalowania systemu nadzorcy dla maszyn klienckich
  - do obsługi sieciowej podłączonych maszyn (przydzielanie adresów IP, wskazywanie na serwer konfiguracji)
  - do zarządzania maszynami klienckimi
- wdrożenie rozwiązania w symulowanym środowisku testowym

Prezentowana praca podzielona została na kilka rozdziałów:

- analiza tematu: przybliżenie wykorzystanych technologii, ich charakterystyka, historia i zastosowania
- wymagania i narzędzia: jakie są wymagania dla aplikacji, jakie narzędzia zostały wykorzystane do rozwiązania problemu, ich opis oraz uzasadnienie wyboru
- specyfikacja zewnętrzna: specyfikacja opisująca całe rozwiązanie z perspektywy użytkowników końcowych, z wyszczególnionymi elementami składowymi rozwiązania
- specyfikacja wewnętrzna: specyfikacja opisująca techniczne aspekty rozwiązania, z perspektywy osoby technicznej zaznamiającej się z kodem źródłowym oprogramowania
- weryfikacja i walidacja: wyniki testów przeprowadzonych na testowym środowisku wdrożeniowym, ich analiza
- podsumowanie i wnioski: prezentacja wniosków wynikających z analizy wyników testów, krytyka proponowanego rozwiązania i propozycje poprawek

Wkład pracy autora Przedmiotem pracy było napisanie aplikacji kontrolera floty urządzeń, aplikacji klienta zarządzającej systemem nadzorcy, konfiguracji fizycznego serwera oraz samo jego skonfigurowanie. Wszystkie te zadania zostały zrealizowane, wraz z zakupem maszyny w roli serwera oraz skonfigurowaniem domowej sieci lokalnej do obsługi rozwiązania.

Rozdział 2

Analiza tematu

Praca została podzielona na dwie główne części: obsługę maszyn wirtualnych i ich obrazów, oraz zarządzanie, instalację i konfigurację systemu zarządcy maszyn wirtualnych.

Istnieje kilka popularnych sposobów synchronizacji i przesyłania plików pomiędzy urządzeniami opartymi o systemy z rodziny Linux. Wśród nich warto zwrócić uwagę na poniższe rozwiązania:

- RSYNC [9]
- SFTP [8]
- serwer HTTP serwujący pliki statyczne

RSYNC RSYNC to narzędzie służące do synchronizacji, kopiowania i przenoszenia plików na maszynach pracujących pod kontrolą systemów z rodziny UNIX. Podstawą działania aplikacji RSYNC jest architektura klient-serwer, gdzie klient najpierw nawiązuje połączenie z serwerem poprzez potok, a w przypadku połączenia sieciowego najpierw uruchamia powłokę na maszynie zdalnej, następnie uruchamia proces serwera i tworzy potok do komunikacji między nimi. Po nawiązaniu połączenia, proces wysyłający dane tworzy listę wszystkich plików do wysłania, wraz z informacjami o własności, wielkości plików, trybie, uprawnieniach i czasie ostatniej zmiany. Następnie, każda ze stron transferu sortuje te listy leksykograficznie, względem ścieżki do bazowego katalogu transferu. W kolejnym kroku tzw. generator iteruje się po liście plików i sprawdza, czy mogą zostać pominięte (czy data ostatniej zmiany i wielkość nie uległy zmianie, lub, jeżeli została wybrana odpowiednia flaga, czy sumy kontrolne poszczególnych plików nie uległy zmianie) oraz tworzy brakujące katalogi. Jeżeli plik nie został oznaczony jako możliwy do pominięcia, jego obecna wersja staje się "plikiem bazowym", a jej obliczone sumy kontrolne bloków plików są wysyłane do procesu odbierającego dane. Proces odczytuje tablicę sum kontrolnych bloków, tworzy indeks funkcji skrótu, aby przyspieszyć operacje wyszukiwania bloków. Następnie plik lokalny jest odczytywany i liczona jest suma kontrolna dla

pierwszego bajtu pliku. Jeżeli wykryta zostanie różnica w sumach kontrolnych, do pierwszego niepasującego bajtu dołaczony zostanie odpowiadający bajt z pliku wysyłanego, a następnie obliczona zostanie suma kontrolna bloku zaczynającego się w kolejnym bajcie. Jeżeli znalezione zostanie dopasowanie w tablicy indeksów funkcji skrótów, dany blok jest traktowany jako pasujący i jest pomijany. W ten sposób proces jest w stanie zrekontruować plik źródłowy. Wszystkie dane z tej analizy wysyłane są do procesu odbierającego, który na ich podstawie odtwarza oryginalny plik. Mechanizm „rolling checksum" opisany powyżej jest integralną częścią algorytmu RSYNC, który sprawia, że nie ma potrzeby transferu całych plików, jedynie zmian, które w nich nastąpiły.[13][19][20]

SFTP SFTP (Secure File Transfer Protocol) to bezpieczny protokół przesyłu plików wykorzystujący protokół SSH do uwierzytelnienia oraz szyfrowania przesyłanych danych. Pierwszym krokiem do przesyłu danych jest nawiązanie połączenia SSH pomiędzy urządzeniami, następnie serwer SFTP sprawdza dostęp do maszyny klienta poprzez SSH (Secure SHell). Jeżeli test przebiegnie pomyślnie, nawiązywane jest połączenie SFTP, przez które rozpoczyna się transfer plików. Protokół obsługuje przesył wielowątkowy, gdzie każda operacja transferu ma przypisany unikalny numer identyfikacyjny, nadawany przez klienta, dzięki czemu serwer może odpowiadać na zapytania klienta asynchronicznie i bez zachowania kolejności. Aby poprawić wydajność, klient wysyła często wiele zapytań zanim zatrzymuje się i czeka na odpowiedzi.[15]

SSH SSH (Secure SHell) to standard protokołów, który miał w założeniu stanowić bezpieczną alternatywę dla protokołu telnet. Do autoryzacji wykorzystuje kryptografię klucz publiczy - klucz prywatny, gdzie w najprostszej postaci pary kluczy sa generowane automatycznie i służą do zabezpieczenia połączenia sieciowego, natomiast sama autoryzacja odbywa się za pomocą hasła użytkownika systemu zdalnego.[12] Obecnie najpopularniejszą implementacją protokołu SSH jest OpenSSH, stworzona przez część członków projektu OpenBSD. Jest wykorzystywana przez większość najpopularniejszych systemów operacyjnych, takich jak różne dystrybucje Linux, macOS, Microsoft Windows, czy OpenBSD.[14]

Serwer HTTP Protokół HTTP (HyperText Transfer Protocol) powstał jako część WWW (World Wide Web) i miał służyć do transferu dokumentów hipertekstowych. Prace nad nim były prowadzone przez Tima Berners-Lee zatrudnionego w CERN (Conseil Européen pour la Recherche Nucléaire) od 1989 roku (pierwsza propozycja systemu hipertekstowego opartego o Internet), gdzie implementacja nastąpiła w 1990 roku, by już rok później pojawiły się pierwsze serwery poza organizacją macierzystą twórcy[4]. W 1997 roku powstała pierwsza ustandaryzowana wersja protokołu, HTTP/1.1, która została wykorzystana w tej pracy.[7] HTTP to obecnie bardzo wszechstronne narzędzie, pozwalające obecnie na przesyłanie dowolnych danych poprzez sieć Internet. Jako protokół

jest podstawą współczesnej sieci Web, służy do obustronnego transferu danych pomiędzy serwerami i klientami. Komunikacja jest rozpoczynana zawsze przez klienta. Tok takiej tranzakcji jest następujący (opis dotyczy wersji HTTP/1.1):

- otwarcie połączenia TCP (Transmission Control Protocol) z serwerem
- wysłanie zapytania HTTP
- odebranie odpowiedzi HTTP od serwera
- zamknięcie połączenia, lub ponowne jego użycie (do wysłania kolejnego zapytania)

Wiadomości HTTP (dla wersji HTTP/1.1 i starszych) mają ustandaryzowany format, czytelny dla człowieka i zapisywalny w postaci tekstu. Format różni się dla zapytań i odpowiedzi, gdzie zapytanie posiada następujące pola[3]:

- metoda HTTP: definiuje operację, którą chce przeprowadzić klient. Może to być np. operacja pobrania danych (metoda GET), czy wysłania danych (metoda POST). Standard HTTP/1.1 zawiera poniższe metody[5]:
  - CONNECT
  - DELETE
  - GET
  - HEAD
  - OPTIONS
  - POST
  - PUT
  - TRACE
- ścieżka dostępu do danych
- wersja protokołu HTTP
- opcjonalne nagłówki
- ciało zapytania, dla metod obsługujących przesył danych z klienta do serwera

Natomiast struktura odpowiedzi jest następująca[3]:

- wersja protokołu HTTP
- kod statusu: jeden z kodów HTTP oznaczający czy zapytanie zostało obsłużone oraz jaki był efekt jego obsługi

- wiadomość statusu: krótka wiadomość opisująca znaczenie kodu statusu
- nagłówki HTTP, podobnie do zapytania
- ciało odpowiedzi, jeżeli wymagane

Z perspektywy prezentowanego projektu ważną cechą protokołu HTTP jest możliwość przesyłu dowolnych danych w ten sam sposób. Zostało to wykorzystane do stworzenia serwera, który potrafi przesyłać dane do klientów w formacie JSON (JavaScript Object Notation), czyli tekstowym formacie opisującym (w tym konkretnym przypadku) konfigurację oraz w formacie binarnym, co jest wykorzystywane do przesyłu obrazów maszyn wirtualnych. Powyższe rozwiązanie zostało wybrane właśnie z powodu łatwości połączenia części przesyłającej konfigurację z serwera do klientów oraz części przesyłającej obrazy. W dalszej części pracy opisano dokładną architekturę tego rozwiązania, jej zalety oraz wady.

Maszyny wirtualne Maszyna wirtualna to w najprostszym ujęciu programowa implementacja fizycznego urządzenia, która wykonuje zadany program w sposób identyczny do tego urządzenia. Aplikacja, która to umożliwia jest nazywana nadzorcą (ang. hypervisor). Jej zadaniem jest tłumaczenie zapytań programów do natywnego sprzętu na zapytania do sprzętu, na którym działa uruchomiona maszyna wirtualna. Istnieje kilka popularnych rodzajów wirtualizacji[2]:

- pełna wirtualizacja z translacją binarną, rys. 2.2 przedstawia schemat działania takiego systemu
- wirtualizacja ze wsparciem sprzętowym, rys. 2.3 przedstawia schemat działania takiego systemu
- wirtualizacja wspierana przez system operacyjny i parawirtualizacja, rys. 2.1 przedstawia schemat działania takiego systemu

W prezentowanym projekcie wykorzystano model ze sprzętowym wsparciem dla wirtualizacji poprzez KVM (Kernel Virtual Machine) wraz z QEMU (Quick EMUlator) i technologie Intel VT-d/AMD-V, pod kontrolą systemu operacyjnego Ubuntu Linux w wersji 22.04 LTS.

KVM KVM to technologia obsługi akceleratorów sprzętowych wirtualizacji wbudowana w systemy z rodziny Linux oparte o jądro w wersji co najmniej 2.6.20, a jej część przeznaczona dla użytkownika została wbudowana w narzędzie QEMU w wersji 1.3.[11]

Intel VT-d/AMD-V Technologie Intel VT-d/AMD-V to zbiór rozwiązań technicznych zaimplementowanych podczas tworzenia procesora oraz chipsetu płyty głównej, które wspierają i ułatwiają używanie maszyn wirtualnych. Sposób działania został opisany na podstawie technologii Intel VT-d.[10] Wsparcie dla wirtualizacji zostało tam osiągnięte poprzez zapewnienie:

- zestawu rozszerzeń dla procesora (nazwanych Intel VMX, Virtual Machine eXtensions)
- wirtualizacji urządzeń We/Wy (Wejścia/Wyjścia)
- wirtualizacji dostępu do pamięci

Wirtualizacja dostępu do pamięci jest zapewniana poprzez mechanizm MMU (Memory Management Unit). Jest to układ tłumaczący adresy logiczne pamięci operacyjnej na jej fizyczne adresy i zapewniający ochronę dostępu procesów do pamięci.

Wirtualizacja urządzeń We/Wy jest realizowana poprzez mechanizm IOMMU (Input/Output Memory Management Unit). Działa on na analogicznej zasadzie co MMU, zapewniając kontrolę oraz tłumaczenie adresów logicznych urządzeń do ich adresów fizycznych.[1] Pozwala to na przekierowanie urządzenia bezpośrednio pod kontrolę systemu operacyjnego uruchomionego w maszynie wirtualnej, pomijając sterownik systemu zarządcy. Technologia ta może zostać wykorzystana do przekazania w taki sposób urządzeń wymagających bardzo niskich opóźnień i dużej przepustowości, na przykład kart graficznych, lub kart sieciowych. Wtedy pełną kontrolę nad sprzętem otrzymuje system operacyjny gościa w maszynie wirtualnej, a narzut wirtualizacji na wydajność jest zminimalizowany.

DHCP Protokół DHCP (Dynamic Host Configuration Protocol) odpowiada za automatyczne przypisywanie adresów IP oraz wysyłanie konfiguracji sieciowej do urządzeń. Jego założeniem była także iteroperatywność z już wtedy istniejącym protokołem BOOTP (BOOTstrap Protocol), przez co pojedyncze paczki danych (nazwane wiadomościami) są te same dla obu protokołów (z kilkoma odstępstwami).[6] Format wiadomości DHCP jest następujący:[6]

- OP (1 bajt): typ wiadomości
- HTYPE (1 bajt): typ adresu karty sieciowej
- HLEN (1 bajt): długość adresu karty sieciowej
- HOPS (1 bajt): ustawiane przez klienta na 0 (zero); używane przez agentów pośredniczących
- XID (4 bajty): numer identyfikacyjny (ID) tranzakcji; używany do rozpoznania wiadomości przeznaczonych dla konkretnego klienta

- SECS (2 bajty): liczba sekund od rozpoczęcia procesu odnawiania lub nabywania adresu; ustawiane przez klienta
- FLAGS (2 bajty): flagi
- CIADDR (4 bajty): adres IP klienta; uzupełniany tylko, jeżeli klient może odpowiedzieć na zapytanie ARP (Address Resolution Protocol) i jest w stanie BOUND, RENEW, lub REBINDING
- YIADDR (4 bajty): „twój" (klienta) adres IP
- SIADDR (4 bajty): adres następnego serwera do użycia podczas pobierania
- GIADDR (4 bajty): adres agenta pośredniczącego
- CHADDR (16 bajtów): adres fizyczny klienta
- SNAME (64 bajty): (opcjonalnie) nazwa serwera, łańcuch znaków zakończony zerem (0)
- FILE (28 bajtów): nazwa pliku uruchomieniowego, łańcuch znaków zakończony zerem (0)
- OPTIONS (zmienna długość): opcjonalne parametry

Zdefiniowano następujące wiadomości w protokole DHCP:[6]

- DHCPDISCOVER - Klient wysyła pakiet rozgłoszeniowy, aby zlokalizować dostępne serwery
- DHCPOFFER - Serwer odpowiada klientowi na zapytanie DHCPDISCOVER z proponowaną konfiguracją
- DHCPREQUEST - Klient odpowiada serwerowi (a) żądając oferowanych parametrów od jednego serwera i ignorując pozostałe, (b) potwierdzając prawidłowość otrzymanej wcześniej konfiguracji, lub (c) przedłużając dzierżawę adresu IP
- DHCPACK - Serwer odpowiada klientowi parametrami konfiguracji, wraz z przyznanym adresem IP
- DHCPNAK - Serwer odpowiada klientowi zaznaczając, że konfiguracja żądana przez klienta jest nieprawidłowa, lub dzierżawa adresu wygasła
- DHCPDECLINE - Klient odpowiada serwerowi, że dany adres sieciowy jest już w użytku

- DHCPRELEASE - Klient odpowiada serwerowi, zwalniając dany adres sieciowy i anulując bieżącą dzierżawę
- DHCPINFORM - Klient wysyła zapytanie serwerowi, prosząc o konfigurację lokalną; adres sieciowy skonfigurowany zewnętrznie

Proces konfiguracji w takiej sieci jest następujący[6]:

- klient wysyła zapytanie `DHCPDISCOVER`
- serwer wysyła klientowi proponowaną konfigurację poprzez wiadomość `DHCPOFFER`
- klient żąda zadanej konfiguracji poprzez wiadomość `DHCPREQUEST`
- serwer odpowiada klientowi wiadomością `DHCPACK`, jeżeli żądana konfiguracja jest poprawna, w przeciwnym przypadku odpowiada `DCHPNAK`
- przy odłączaniu od sieci, klient wysyła wiadomość `DHCPRELEASE`, zwalniając zajmowany adres sieciowy

TFTP TFTP (Trivial File Transfer Protocol) to nieskomplikowany protokół przesyłu plików. Jego jedynym zadaniem jest odczyt pliku po stronie serwera i przesłanie go do klienta. Nie obsługuje listowania plików, ani uwierzytelniania.[17]

Protokół obsługuje następujące tryby przesyłu danych[17]:

- `NETASCII`, czyli zmodyfikowana forma formatu `ASCII`, rozszerzona do długości 8 bitów i zawierająca dodatkowe znaki kontrolne
- `OCTET`, czyli format przesyłu surowych bajtów danych
- `MAIL`, uznany za przestarzały w specyfikacji RFC1350[17]

Proces transferu plików jest następujący[17]:

- klient wysyła zapytanie RRQ (read request/żądanie odczytu), lub WRQ (write request/żądanie zapisu), zawierające nazwę pliku i tryb transferu do serwera na port 69
- serwer odpowiada pakietem OPTIONS ACK (jeżeli były użyte) i pakietem ACK w przypadku zapytania WRQ; w przypadku zapytania RRQ odpowiedź to od razu pierwszy pakiet żądanych danych
- maszyna będąca źródłem pliku wysyła go w ponumerowanych segmentach (domyślnie o wielkości 512 bajtów); maszyna odbierająca na każdy segment odpowiada pakietem ACK

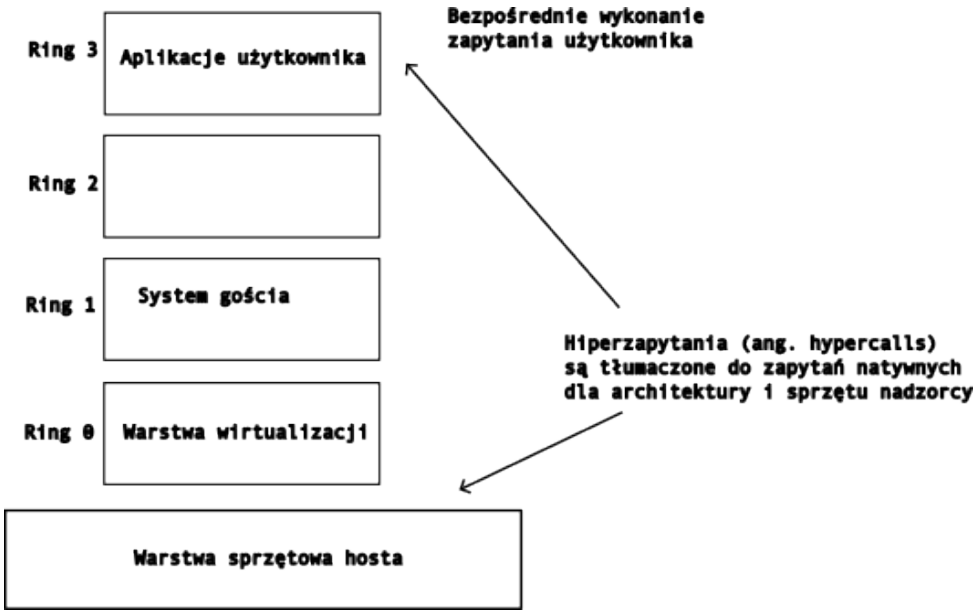
- ostatni segment jest rozpoznawany po wielkości, która jest mniejsza od poprzednich (domyslnie musi mieścić się w przedziale od 0 bajtów do 511 bajtów)
- jeżeli odpowiedź ACK dla danego segmentu nigdy nie została otrzymana przez maszynę wysyłającą, po upływie określonego czasu (timeout) segment jest retransmitowany

PXE PXE (Preboot eXecution Environment) to specyfikacja ustandaryzowanego środowiska pozwalającego na uruchamianie oprogramowania poprzez sieć. Rozwiązanie takie pracuje w architekturze klient-serwer, gdzie od klienta wymagane jest jedynie posiadanie odpowiedniej karty sieciowej (wspierającej to rozwiązanie) oraz wsparcie w oprogramowaniu inicjalizującym maszyny, odpowiedzialnym za uruchamianie oprogramowania (na przykład BIOS/UEFI dla maszyn o architekturze X86). Część serwera jest zbiorem już wcześniej istniejących rozwiązań, składającym się z serwera DHCP oraz serwera TFTP. [18]

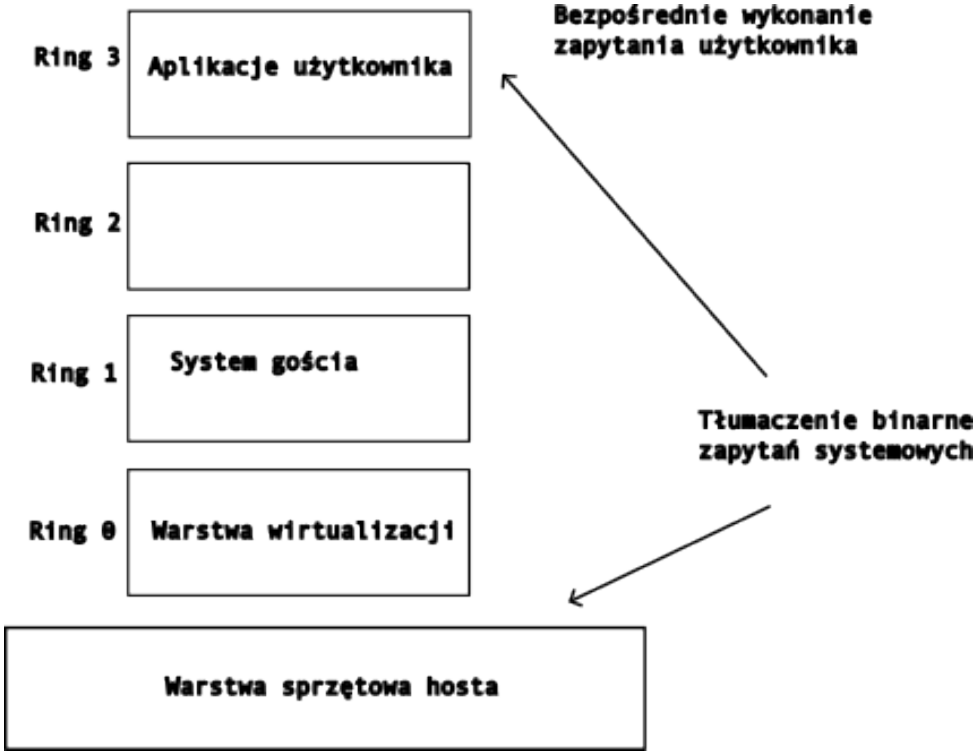
Proces uruchamiania składa się z następujących kroków[18]:

- wysłanie zapytania `DHCPDISCOVER` w trybie BROADCAST przez klienta
- serwer DHCP odpowiada poprzez wiadomość `DHCPOFFER`; jeżeli serwer DHCP nie implementuje obsługi PXE, wysyła poprawną wiadomość nie zawierającą informacji o lokalizacji plików potrzebnych do uruchomienia minimalnego systemu, a maszyna klienta nie uruchamia się poprzez PXE. W przeciwnym przypadku, serwer DHCP poza zwykłymi informacjami przesyłanymi we wiadomości `DHCPOFFER` dodaje także informacje o lokalizacji serwera TFTP, gdzie znajdują się pliki potrzebne do uruchomienia minimalnego środowiska uruchomienowego, oraz o nazwach tych plików
- klient PXE konfiguruje się danymi przesłanymi przez serwer DHCP oraz pobiera pliki uruchomieniowe do pamięci RAM (Random Access Memory); uruchamia otrzymane pliki
- minimalne środowisko uruchomieniowe powinno pobrać pełen obraz instalacyjny lub obraz systemu do uruchomienia i obsłużyć jego urchomienie/instalację; ten krok najczęściej jest wykonywany poprzez bardziej skomplikowane i niezawodne protokoły, takie jak HTTP, lub NFS

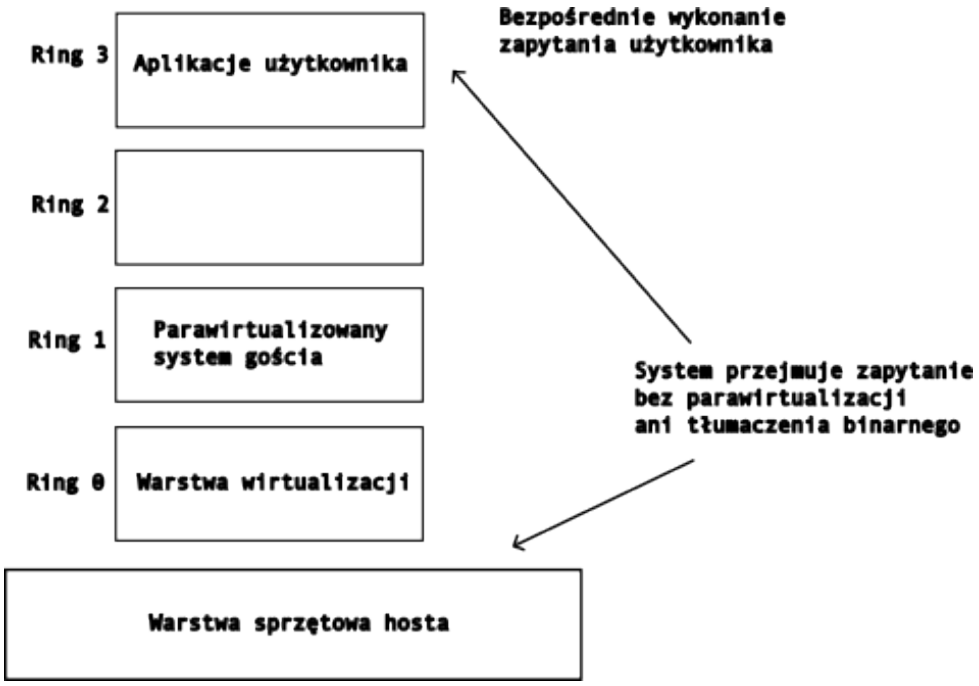
Jako minimalne środowisko uruchomieniowe wykorzystuje się często zestaw jądra Linux oraz prostego obrazu initrd, które mają za zadanie pobrać i uruchomić instalator pełnego systemu operacyjnego.



Rysunek 2.1: Diagram przedstawiający schemat działania systemu z parawirtualizacją (za [2], rys. 1.6)



Rysunek 2.2: Diagram przedstawiający schemat działania systemu z pełną wirtualizacją (za [2], rys 1.5)



Rysunek 2.3: Diagram przedstawiający schemat działania systemu ze sprzętowym wsparciem wirtualizacji (za [2], rys. 1.7)

Rozdział 3

Wymagania i narzędzia

Do implementacji pierwszej części projektu wyszczególnionej powyżej wykorzystano język Python, do zaimplementowania prostego serwera oraz klienta, synchronizujących obrazy maszyn wirtualnych, QEMU wraz z KVM do uruchamiania samych maszyn oraz system operacyjny nadzorcy Ubuntu Linux w wersji 22.04 LTS.

Część projektu odpowiedzialna za obsługę systemów zarządców maszyn wirtualnych została także oparta o system operacyjny Ubuntu Linux w wersji 22.04 LTS, serwer TFTP oraz PXE do serwowania medium instalacyjnego systemu bazowego nadzorców, cloud-init oraz Ansible do automatyzacji instalacji i konfiguracji hostów.

QEMU QEMU to zintegrowany system do emulacji procesorów poprzez tłumaczenie binarne, potrafi także korzystać z technologii KVM do uruchamiania aplikacji/maszyn wirtualnych z minimalnym narzutem wydajnościowym. [16]

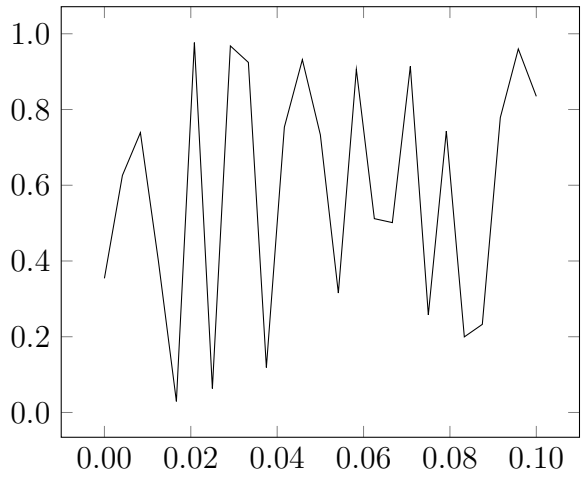
- wymagania funkcjonalne i niefunkcjonalne
- przypadki użycia (diagramy UML) – dla prac, w których mają zastosowanie
- opis narzędzi, metod eksperymentalnych, metod modelowania itp.
- metodyka pracy nad projektowaniem i implementacją – dla prac, w których ma to zastosowanie

Rozdział 4

[Właściwy dla kierunku – np. Specyfikacja zewnętrzna]

Jeśli „Specyfikacja zewnętrzna”:

- wymagania sprzętowe i programowe
- sposób instalacji
- sposób aktywacji
- kategorie użytkowników
- sposób obsługi
- administracja systemem
- kwestie bezpieczeństwa
- przykład działania
- scenariusze korzystania z systemu (ilustrowane zrzutami z ekranu lub generowanymi dokumentami)



Rysunek 4.1: Podpis rysunku po rysunkiem.

Rozdział 5

[Właściwy dla kierunku – np. Specyfikacja wewnętrzna]

Jeśli „Specyfikacja wewnętrzna”:

- przedstawienie idei
- architektura systemu
- opis struktur danych (i organizacji baz danych)
- komponenty, moduły, biblioteki, przegląd ważniejszych klas (jeśli występują)
- przegląd ważniejszych algorytmów (jeśli występują)
- szczegóły implementacji wybranych fragmentów, zastosowane wzorce projektowe
- diagramy UML

Krótka wstawka kodu w linii tekstu jest możliwa, np. `int a;` (biblioteka `listings`). Dłuższe fragmenty lepiej jest umieszczać jako rysunek, np. kod na rys 5.1, a naprawdę długie fragmenty – w załączniku.

```
1  class test : public basic
2  {
3      public:
4        test (int a);
5        friend std::ostream operator<<(std::ostream & s,
6                                       const test & t);
7      protected:
8        int _a;
9
10 };
```

Rysunek 5.1: Pseudokod w `listings`.

Rozdział 6

Weryfikacja i walidacja

- sposób testowania w ramach pracy (np. odniesienie do modelu V)
- organizacja eksperymentów
- przypadki testowe zakres testowania (pełny/niepełny)
- wykryte i usunięte błędy
- opcjonalnie wyniki badań eksperymentalnych

Tabela 6.1: Nagłówek tabeli jest nad tabelą.

	metoda						
			alg. 3			alg. 4, $\gamma = 2$	
ζ	alg. 1	alg. 2	$\alpha = 1.5$	$\alpha = 2$	$\alpha = 3$	$\beta = 0.1$	$\beta = -0.1$
0	8.3250	1.45305	7.5791	14.8517	20.0028	1.16396	1.1365
5	0.6111	2.27126	6.9952	13.8560	18.6064	1.18659	1.1630
10	11.6126	2.69218	6.2520	12.5202	16.8278	1.23180	1.2045
15	0.5665	2.95046	5.7753	11.4588	15.4837	1.25131	1.2614
20	15.8728	3.07225	5.3071	10.3935	13.8738	1.25307	1.2217
25	0.9791	3.19034	5.4575	9.9533	13.0721	1.27104	1.2640
30	2.0228	3.27474	5.7461	9.7164	12.2637	1.33404	1.3209
35	13.4210	3.36086	6.6735	10.0442	12.0270	1.35385	1.3059
40	13.2226	3.36420	7.7248	10.4495	12.0379	1.34919	1.2768
45	12.8445	3.47436	8.5539	10.8552	12.2773	1.42303	1.4362
50	12.9245	3.58228	9.2702	11.2183	12.3990	1.40922	1.3724

Rozdział 7

Podsumowanie i wnioski

- uzyskane wyniki w świetle postawionych celów i zdefiniowanych wyżej wymagań
- kierunki ewentualnych danych prac (rozbudowa funkcjonalna . . .)
- problemy napotkane w trakcie pracy

Bibliografia

[1] AMD. *AMD64 Technology; AMD64 Architecture Programmer's Manual Volume 2: System Programming, Rev. 3.3*. Architecture Specification. 2022. DOI: 24593.

[2] Diane Barrett i Gregory Kipper. „1 - How Virtualization Happens". W: *Virtualization and Forensics*. Red. Diane Barrett i Gregory Kipper. Boston: Syngress, 2010, s. 3–24. ISBN: 978-1-59749-557-8. DOI: https://doi.org/10.1016/B978-1-59749-557-8.00001-1. URL: https://www.sciencedirect.com/science/article/pii/B9781597495578000011.

[3] Mozilla MDN Contributors. *An overview of HTTP*. ostatnia modyfikacja: Nov 22, 2022. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview (term. wiz. 31. 12. 2022).

[4] Mozilla MDN Contributors. *Evolution of HTTP*. ostatnia modyfikacja: Oct 28, 2022. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP#http1.1_%E2%80%93_the_standardized_protocol (term. wiz. 31. 12. 2022).

[5] Mozilla MDN Contributors. *HTTP request methods*. ostatnia modyfikacja: Sep 9, 2022. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods (term. wiz. 31. 12. 2022).

[6] Ralph Droms. *Dynamic Host Configuration Protocol*. RFC 2131. Mar. 1997. DOI: 10.17487/RFC2131. URL: https://www.rfc-editor.org/info/rfc2131.

[7] Roy T. Fielding i in. *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2068. Sty. 1997. DOI: 10.17487/RFC2068. URL: https://www.rfc-editor.org/info/rfc2068.

[8] Joseph Galbraith i Oskari Saarenmaa. *SSH File Transfer Protocol*. Internet-Draft draft-ietf-secsh-filexfer-13. Work in Progress. Internet Engineering Task Force, lip. 2006. 60 s. URL: https://datatracker.ietf.org/doc/draft-ietf-secsh-filexfer/13/.

[9] Steve Newsted (Red Hat). *Keeping Linux files and directories in sync with rsync*. 2021. URL: https://www.redhat.com/sysadmin/sync-rsync (term. wiz. 31. 12. 2022).

[10] Intel. *Intel® Virtualization Technology for Directed I/O; Architecture Specification, rev. 4.0*. Architecture Specification. 2022. DOI: D51397-015.

[11] KVM. *Main page — KVM,* 2016. URL: https://www.linux-kvm.org/index.php?title=Main_Page&oldid=173792 (term. wiz. 31. 12. 2022).

[12] Chris M. Lonvick i Tatu Ylonen. *The Secure Shell (SSH) Protocol Architecture*. Request for Comments 4251. 2006. 30 s. DOI: 10.17487/RFC4251.

[13] N/A. *How Rsync Works; A Practical Overview*. N/A. URL: https://rsync.samba.org/how-rsync-works.html (term. wiz. 31. 12. 2022).

[14] N/A. *OpenSSH Users*. N/A. URL: https://www.openssh.com/users.html (term. wiz. 31. 12. 2022).

[15] N/A. *SSH File Transfer Protocol (SFTP): Get SFTP client and server*. N/A. URL: https://www.ssh.com/academy/ssh/sftp-ssh-file-transfer-protocol (term. wiz. 31. 12. 2022).

[16] QEMU. *QEMU; A generic and open source machine emulator and virtualizer*. N/A. URL: https://www.qemu.org (term. wiz. 31. 12. 2022).

[17] Dr. Karen R. Sollins. *The TFTP Protocol (Revision 2)*. RFC 1350. Lip. 1992. DOI: 10.17487/RFC1350. URL: https://www.rfc-editor.org/info/rfc1350.

[18] Intel with special contributions from SystemSoft. *Preboot Execution Environment (PXE) Specification, rev. 2.1*. Architecture Specification. 1999.

[19] Andrew Tridgell i Paul Mackerras. „The rsync algorithm". W: (1996).

[20] Andrew Tridgell i in. „Efficient algorithms for sorting and synchronization". W: (1999).

Dodatki

Spis skrótów i symboli

DNA kwas deoksyrybonukleinowy (ang. *deoxyribonucleic acid*)

MVC model – widok – kontroler (ang. *model–view–controller*)

N liczebność zbioru danych

μ stopnień przyleżności do zbioru

$\mathbb{E}$ zbiór krawędzi grafu

$\mathcal{L}$ transformata Laplace'a

Źródła

Jeżeli w pracy konieczne jest umieszczenie długich fragmentów kodu źródłowego, należy je przenieść w to miejsce.

```
1 if (_nClusters < 1)
2     throw std::string ("unknown number of clusters");
3 if (_nIterations < 1 and _epsilon < 0)
4     throw std::string ("You should set a maximal number of
          iteration or minimal difference — epsilon.");
5 if (_nIterations > 0 and _epsilon > 0)
6     throw std::string ("Both number of iterations and minimal
          epsilon set — you should set either number of iterations
          or minimal epsilon.");
```

Lista dodatkowych plików, uzupełniających tekst pracy

W systemie do pracy dołączono dodatkowe pliki zawierające:

- źródła programu,
- dane testowe,
- film pokazujący działanie opracowanego oprogramowania lub zaprojektowanego i wykonanego urządzenia,
- itp.

Spis rysunków

Spis tabel