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 2022

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 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[2]
- SFTP[1]
- 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.[3][8][9]

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. Jeżeli test przebiegnie pomyślnie, nawiązywane jest połączenie SFTP, przez które rozpoczyna się transfer plików. SFTP obsługuje transport wielu

Do implementacji pierwszej części 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 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.

- sformułowanie problemu
- osadzenie tematu w kontekście aktualnego stanu wiedzy (*state of the art*) o poruszanym problemie
- studia literaturowe [6, 7, 5, 4] - opis znanych rozwiązań (także opisanych naukowo, jeżeli problem jest poruszany w publikacjach naukowych), algorytmów,

Wzory

$$y = \frac{\partial x}{\partial t} \tag{2.1}$$

jak i pojedyncze symbole x i y składa się w trybie matematycznym.

Rozdział 3

Wymagania i narzędzia

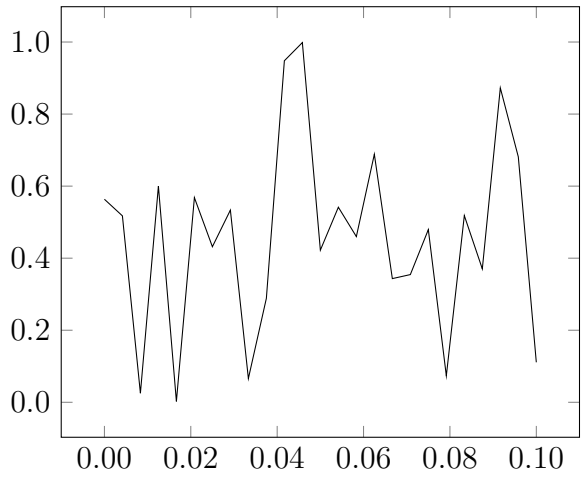
- wymagania funkcjonalne i niefunkcjonalne
- przypadki użycia (diagramy UML) – dla prac, w których mają zastosowanie
- opis narzędzi, metod eksperymentalnych, metod modelowania itp.
- metodyka pracy nad projektowaniem i implementacją – dla prac, w których ma to zastosowanie

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] 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/`.

[2] 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).

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

[4] Imię Nazwisko i Imię Nazwisko. *Tytuł strony internetowej*. 2021. URL: `http://gdzies/w/internecie/internet.html` (term. wiz. 30. 09. 2021).

[5] Imię Nazwisko, Imię Nazwisko i Imię Nazwisko. „Tytuł artykułu konferencyjnego". W: *Nazwa konferecji*. 2006, s. 5346–5349.

[6] Imię Nazwisko, Imię Nazwisko i Imię Nazwisko. „Tytuł artykułu w czasopiśmie". W: *Tytuł czasopisma* 157.8 (2016), s. 1092–1113.

[7] Imię Nazwisko, Imię Nazwisko i Imię Nazwisko. *Tytuł książki*. Warszawa: Wydawnictwo, 2017. ISBN: 83-204-3229-9-434.

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

[9] 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