Ćwiczenie nr: 8
Temat: Optymalizacja serwerów WWW :Apache

1. Informacje ogólne

Serwer WWW jest to program obsługujący żądania protokołu HTTP. Z serwerem WWW łączy się przeglądarka internetowa ładując żądaną przez użytkownika stronę internetową. Serwery WWW pośredniczą czasem także w realizacji innych zadań lub usług np. zleca przetworzenie pliku źródłowego serwerowi PHP. Przykładowymi serwerami są: najbardziej popularny wśród użytkowników **Apache** oraz **Lighttpd**.
Apache jest otwartym serwerem HTTP dostępnym dla wielu systemów operacyjnych np. Microsoft Windows oraz Linux. Wraz z obsługą PHP i silnikiem baz danych MySQL na platformie Linux, Apache stanowi tzw. platformę **LAMP** (Linux, Apache, MySQL, PHP). LAMP jest to akronim określający zestaw oprogramowania *open source* stanowiący popularną platformę serwerową dynamicznych stron WWW.

2. Protokół http

Protokół **HTTP** (ang. Hypertext Transfer Protocol) udostępnia znormalizowany sposób komunikowania się komputerów ze sobą. Określa on formę żądań klienta dotyczących danych oraz formę odpowiedzi serwera na te żądania. Jest zaliczany do protokołów bezstanowych, ponieważ nie zachowuje żadnych informacji o poprzednich transakcjach z klientem. Pozwala to znacznie zmniejszyć obciążenie serwera, jednak jest kłopotliwe w sytuacji, gdy np. trzeba zapamiętać konkretny stan dla użytkownika, który wcześniej łączył się już z serwerem. Najczęstszym rozwiązaniem tego problemu jest wprowadzenie mechanizmu cookies. Protokół HTTP standardowo korzysta z portu 80 (TCP).

a) Wybrane metody HTTP
GET - pobranie zasobu wskazanego przez URI
HEAD - pobiera informacje o zasobie, stosowane do sprawdzania dostępności zasobu
PUT - przyjęcie danych w postaci pliku przesyłanych od klienta do serwera
POST - przyjęcie danych przesyłanych od klienta do serwera
DELETE - żądanie usunięcia zasobu, włączone dla uprawnionych użytkowników
OPTIONS - informacje o opcjach i wymaganiach istniejących w kanale komunikacyjnym
TRACE - diagnostyka, analiza kanału komunikacyjnego
b) Wybrane zapytania HTTP
GET / HTTP/1.1 (prośba o zwrócenie dokumentu o URI / zgodnie z protokołem HTTP 1.1)
User-Agent: Mozilla/5.0 (X11; U; Linux i686; pl; rv:1.8.1.7/20070914 Firefox/2.0.0.7 (nazwa aplikacji klienckiej)
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8 (akceptowane (bądź nieakceptowane dla q=0) przez klienta typy plików)
Accept-Language: pl,en-us;q=0.7,en;q=0.3 (preferowany język strony)
Accept-Charset: ISO-8859-2,utf-8;q=0.7,*;q=0.7 (preferowane kodowanie znaków)
c) Wybrane odpowiedzi HTTP
Date: Sun, 11 Jul 2004 12:04:30 GMT (czas serwera)
Server: Apache/2.0.50 (Unix) DAV/2 (opis aplikacji serwera)

3. Serwer Apache (Linux)

Pliki konfiguracyjne znajdują się w :
`/etc/httpd/conf/httpd.conf`
znajdują się ustawienia serwera.
Kontrola serwera Apache:

```
service httpd status – sprawdzenie stanu
service httpd start – uruchomienie serwera
service httpd stop – zatrzymanie serwera
service httpd restart – restart serwera
```

Po uruchomieniu przeglądarki i wpisaniu **http://localhost** uruchomi się nasza strona postawiona na serwerze Apache

Serwis WWW umieszczamy najczęściej w katalogu **htdocs.**
`/var/WWW/html/`
Po wydaniu polecenia:
`#ps -u root`
powinien być widoczny proces o nazwie httpd.

4. Optymalizacja serwerów WWW

Skuteczna optymalizacja serwerów WWW jest częstym problemem administratorów serwerów WWW. Przyczyn powodujących zakłócenie płynności działania serwisów jest bardzo wiele. Sposobów na zidentyfikowanie „wąskiego przejścia" jak też i sposobów na rozwiązanie problemu jest bardzo wiele. Przyczyn powodujących ten stan rzeczy może być wiele. Do najczęstszych powodów powodujących opóźnienia można zaliczyć:

- **Problemy z siecią (obciążenie sieci, routing)**
- **Przeciążenie serwera**
- **Bardzo duża liczba generowanych zapytań przez użytkowników vs słaba wydajność serwera**

W codziennej pracy administratora serwera WWW najlepszą strategią jest systematyczna kontrola systemu a w związku z tym zapobieganie potencjalnym problemom. Do najczęstszych technik poprawy wydajności serwerów WWW jest ich optymalizacja na 4 poziomach:

- **Optymalizacja na poziomie sieci oraz routingu**
- **Optymalizacja na poziomie konfiguracji serwerów WWW**
- **Optymalizacja na poziomie protokołu HTTP**
- **Optymalizacja na poziomie kodu HTML**

5. Optymalizacja serwera Apache

W pliku httpd.conf w katalogu /etc/httpd/conf znajdują się następujące parametry:

- Parametr ***Timeout***

Jest to czas jaki serwer potrzebuje na zamknięcie połączenia nie doczekawszy się nowego pakietu lub zapytania. Zbyt duża wartość powoduje, że takie zapytania blokują procesy podrzędne i uniemożliwiają przyjmowanie nowego połączenia które serwer może obsłużyć w międzyczasie.

- Opcja ***Keep Alive***

Włączenie tej opcji pozwala klientom używać jednego połączenia do obsługi wielu zapytań. Jeżeli opcja jest wyłączona użytkownik musi korzystać z nowego połączenia dla każdego nowego zapytania co w efekcie prowadzi za każdym razem do przejścia całej procedury nawiązania połączenia.

- Parametr ***KeepAliveTimeout***
- Parametr ***MaxSpareServers***

Parametr ten ogranicza liczbę bezczynnych procesów httpd. Powoduje to zmniejszenie liczby procesów httpd w czasie gdy serwer jest mniej obciążony. W wyniku tego Apache będzie dzielił nowe procesy httpd w celu obsłużenia zwiększającego się opóźnienia.

- Parametr ***MinSpareServers***

Parametr ten określa liczbę bezczynnych procesów które powinien podtrzymać w pamięci serwer. Stanowią one bufor w razie pojawienia się dużych skoków obciążenia serwera. W przypadku używania modułu MPM dla Apache 2 wykorzystujemy parametry:
o ***MaxSpareThreads***
o ***MinSpareThreads***
W tym przypadku wartości odnoszą się do liczby wątków w procesie Apache. Jest to o tyle lepsze rozwiązanie, że Apache będzie miał mniejsze problemy z tworzeniem nowych wątków niż procesów.

Wątek nie musi posiadać własnego obszaru pamięci do którego proces musiałby być skopiowany. Wątek też nie posiada własnego identyfikatora (process ID) ale wraz z pozostałymi wątkami współdzieli ID głównego procesu.

- Parametr ***MaxRequestsPerChild***

Parametr ten usuwa procesy potomne po przetworzeniu pewnej liczby zapytań i tworzy nowe. Włączenie tego parametru nie zawsze jest konieczne, dlatego też należy wykonać wcześniej testy.

6. Oprogramowanie do testowania serwerów pod dużym obciążeniem

ab

Program ab jest dostarczany razem z Apache. Program symuluje dużą liczbę jednoczesnych zapytań a następnie mierzy liczbę zapytań obsługiwanych przez serwer w czasie 1s oraz czas potrzebny
serwerowi do ich obsługi.
Przykładowa linia komend:
ab –n 1000 –c 20 http://127.0.0.1/
W powyższym przykładzie program uruchamia 1000 zapytań wykonując w każdej fazie jednocześnie po 20 zapytań.

SCENARIUSZ

1. Zrobić kopię zapasową pliku httpd.conf (nazwać httpd_kopia.conf)
2. Otworzyć plik httpd.conf edytorem tekstu, np. pluma
3. Na końcu pliku dopisać w kolejnych wierszach parametry, które będą podlegać zmianom w trakcie trwania laboratorium: KeepAlive, MaxClients, TimeOut
4. Do podanych wyżej parametrów dopisać ich parametry (odpowiednio: On, 10, 100).
5. Sprawdzić stan serwera Apache poleceniem podanym w części teoretycznej laboratorium, jeśli serwer nie jest uruchomiony należy go uruchomić.
6. Ustalić z prowadzącym parametry serwera do testów oraz ich wartości.
   a. KeepAlive On/Off
   b. Max Clients ……. ….. …… ……. …… …….
   c. Time Out ……..
7. Wartości parametrów zapisać w pliku i zrestartować serwer.
8. Uruchomić program ab: przykład ab –n 1000 –c 20 http://127.0.0.1/, z uwzględnieniem parametrów: liczba żądań do wykonania oraz liczba żądań, które będą wykonywane jednocześnie (domyślnie: jedno).
9. Przeprowadzić testy dla zmienionych parametrów z uwzględnieniem wszystkich kombinacji. Dla każdej kombinacji przeprowadzić test trzy razy.
   W czterech kolejnych tabelach należy notować informacje: Time taken for tests [s], Requests per second [#/sec], Time per request [ms] (mean), Time per request [ms] (mean, across all conc. req)

	Parametr np. MaxClients				
Liczba żądań do wykonania		**10**	**50**	**100**	**1000**
	1000				
	5000				
	10000				
	100000				

10. Po wykonaniu wszystkich testów porównać wyniki i zaproponować 2 optymalne zestawy parametrów i jeszcze raz przeprowadzić dla nich serie testów.

Wyniki pomiarów:

- Opracować statystycznie uzyskane wyniki
- Wykonać wykresy zależności wartości zmierzonych od ustawionych, porównać z domyślnymi
- Wyciągnąć i opracować wnioski na podstawie uzyskanych danych