FSTORAGE PROJECT

Dokumentacja użytkowa

Wersja: 0.5
Autor: Łukasz Marcin Podkalicki

Spis treści

1 Opis projektu

Fstorage (dalej zamiennie FS) został zaprojektowany w połowie roku 2009. Oprogramowanie to ma docelowo rozwiązać problem składowania plików (duszych ilości, pliki multimedialne, etc..) przez wiele aplikacji internetowych (równocześnie; różne technologie). W swej idei Fstorage jest prostym i szybkim serwerem storage'owym dostępnym dla popularnych platform hostujących aplikacje www – systemy linux oraz BSD. Jest również lepsza alternatywa mojego wcześniejszego, komercyjnego rozwiązania fbox (EOL-GROUP; przykładowe wdrożenie – multimedia dla biznesu – vidze.pl) napisanego w języku Python. Głównymi zadaniami Fstorage są: zdeponować plik (o konkretnym identyfikatorze, klucz pliku) na serwerze oraz pozyskać plik z serwera odwołując się do konkretnego klucza. Struktura rozkładu fizycznych serwerów z funkcjonalnością Fstorage, podobnie jak to ma miejsce w programie Memcache, może przypominać strukturę sieci eliptycznej. W obecnej wersji projekt wspiera tylko cechy protokołu IP version 4.

W skład projektu FS wchodzi:

- serwer programowy FSTORAGED. Program znajduje się w pakiecie fstorage-0.5.tar.gz. Po instalacji pakietu jest dostępny w „/usr/bin". Wymaga prostego pliku konfiguracyjnego (przykład w fstoraged.sample.conf). Aby zobaczyć wszystkie dostępne opcje należy wpisać w terminalu „fstoraged -h";
- biblioteki API dla języków formalnych C/C++. Standardowo wraz z pakietem fstorage-0.5.tar.gz dostarczane są bliblioteki dla ANSI C (c99) libfstorage oraz C++ (stdgnu98) libfstorage++ (-lfstorage, -lfstorage++) instalowane razem z pakietem docelowo w „/usr/lib". Pliki nagłówkowe do tych bibliotek kopiowane są docelowo do „/usr/include" (fstorage.h dla ANSI C, fstorage++.h dla C++). Biblioteka dla C++ jest obiektowym wrapperem dla funkcji z biblioteki ANSI C;
- bilbioteka API dla Python. Jest to oddzielny pakiet pyfstorage-0.5.tar.gz, który do instalacji wymaga wcześniejszeg zainstalowania pakietu fstorage-0.5.tar.gz. Python'owa biblioteka używa funkcjonalności CPython (korzysta również z biblioteki libfstorage dla ANSI C) do stworzenia odpowiedniego rozszerzenia dla tego języka;
- Biblioteka API dla PHP. Podobnie jak w przypadku biblloteki dla Python jest to oddzielny pakiet phpfstorage-0.5.tar.gz i wymagana jest wcześniejsza instalacja pakietu fstorage-0.5.tar.gz. FS API dla PHP buduje się jak standardowe moduły dla tego języka (Zend/phpize);

FS obsługuje na jednym porcie dwa rodzaje protokołów:

- FSTP (File Storage Trasfer Protocol) – dedykowany, binarny protokół (dalej zamiennie BTP);
- HTTP – metody HEAD, PUT, GET;

FS we wcześniejszej wersji (0.2) korzystał z autonomicznego modułu serwera WWW napisanego specjalnie dla Nginx - do serwowania plików po HTTP. W aktualnej wersji serwer FSTORAGED wykonuje serwowanie plików po HTTP jak i po FSTP wykorzystując „sendfile". Działanie serwera programowego z założenia jest proste i opiera się o interfejs gniazd BSD – przy każdym (autoryzowanym – simple connection host management - Allowed IP) połączeniu tworzony jest nowy wątek (współbieżność wątkowa niewyprzedzająca). Serwer w pierwszej kolejności sprawdza jakim protokółem chce „rozmawiać" klient, wybiera odpowiedni handler do obslugi połączenia i w razie potrzeby wykonuje pojedynczy Handshake. Następnie serwer wykonuje zlecone operacje i

odpowiada na prośbę klienta. Na tym etapie transakcja się kończy i następuje rozłączenie. Rozwiązanie takie ma zapewnić odejście od problemu CK10 (podobnie jak to wykonał Igor Sysojev w Nginx). Również ważnym elementem w projekcie jest generowanie kluczy. Klucze „fkey" (File Key) generowane są na podstawie zadanych ciągów znaków (np. przy wysyłanie pliku; file_send(„klucz nr 1", „/path/to/file")). W FS wykorzystywany do tego celu jest prosty algorytm CRC16, który zamienia ciąg znaków na system liczbowy dziesiętny. Następnie wynik kodowany jest dedykowanym algorytmem do systemu liczbowego pozycyjnego o podstawie 60 (alfabet składa się kolejno z cyfr, małych i dużych znaków alfabetu łacińskiego).

2 Wymagania środowiska

- System operacyjny linux lub freeBSD (architektura x86, 32 lub 64 bit),
- najnowszy dostępny kompilator gcc/g++ (v. >= 4.4.1),
- (opcjonalnie) Python v. >= 2.5
- (opcjonalnie) PHP v. >= 5.0.1

3 Instalacja pakietu

3.1 Pobranie najnowszych źródeł

Pliki można pobrać ze strony http://sourceforge.net/projects/fstorage/

(najnowszy tarball znajduje się w aktualnym względem tej dokumentacji folderze)

3.2 Rozpakowanie tarball'a

PATH – ścieżka do katalogu, gdzie został zapisany tarball.

```
$ cd $PATH
$ tar -xvf ./fstorage-0.5.tar.gz
$ …
```

3.3 Instalacja

Instalacja wygląda standardowo: configure, make, make install

```
$ cd fstorage-0.5/
$ ./configure
$ make
$ sudo make install
```

4 Konfiguracja

4.1 Serwer fstoraged

Serwer do prawidłowego działania wymaga podania ścieżki do pliku konfiguracyjnego. Plik konfiguracyjny składa się z kilku zmiennych konfiguracyjnych (pisanych z małych liter).

- **debug = {0, 1, 2, 3, 4}** – poziomy logowania, odpowiednio: BRAK, CRITICAL, ERROR, NOTICE, DEBUG; zmienna wymagana.

- **logfile = PATH** – ścieżka do pliku gromadzącego logi (działa tylko w trybie background, w trybie interactive logi przekazywane są na standardowe wyjście błędów); zmienna opcjonalna.
- **host = IP_HOST** – nazwa hosta lub adres IP; zmienna wymagana;
- **port = PORT** – numer portu, liczba ze zbioru <1; 2^16 - 1>; zmienna wymagana;
- **salt = NUM** – duża liczba, ziarno losowe; zmienna opcjonalna;
- **base = NUM** – liczba ze zbioru <2, 65>; zmienna opcjonalna;
- **path = PATH** – ścieżka do katalogu, gdzie będą gromadzone pliki; zmienna wymagana;
- **allowed_ips** = ip1, ip2, …, ipN – prosty system autoryzacji połączeń na podstawie IP; zmienna opcjonalna.

4.2 Client (wykorzystanie API)

Klientem może być aplikacja obsługująca protokół HTTP (np. przeglądarka internetowa, http://127.0.0.1:8001/?fkey=HASH) lub aplikacja wykorzystująca bezpośrednio FS API. Do połączenia się z serwerem fstoraged wymagane jest podanie „fsuri", np. „fstp://127.0.0.1:8001".

Przykład dla języka Python (wymagany jest zainstalowany pakiet fstorage-0.5 oraz pyfstorage-0.5):

```
#!/usr/bin/env python
from pyfstorage import Fstorage

fs = Fstorage("fstp://127.0.0.1:8001")

fs.file_send(...)
fs.file_recv(...)
fs.file_remove(...)
fs.file_replace(...)
```

Pliki z przykładami znajdują się w katalogu „examples".

5 Uruchomienie

5.1 Lista opcji dostępnych do uruchomienia serwera.

```
$ fstoraged -h
Usage: fstoraged [options]

Options:
  -v, --version              show program's version number and exit
  -h, --help                 show this help message and exit
  -d, --daemon               run in background mode
  -c PATH, --config PATH     path to config file
```

5.2 Przykładowy plik konfiguracyjny

```
# Sample configuration file fstoraged
```

```
debug = 4
#logfile = ./fstoraged.log
base = 45
salt = 91029271
host = 127.0.0.1
port = 8001
path = /var/tmp
```

5.3 Tryb intractive

```
$ fstoraged -c ./fstoraged.sample.conf
```

5.4 Tryb background

```
$ fstoraged -c ./fstoraged.sample.conf -d
```

5.5 Przykład

Uruchamiamy serwer w trybie interactive (na początku najlepiej ustawić w pliku konfiguracyjnym debug=4, zeby lepiej zrozumić jak wszystko działa), w tym celu w terminalu należy wpisać komendę:

```
$ fstoraged -c ./docs/fstoraged.sample.conf -d
```

Serwer już działa więc spróbujmy połączyć się z nim prostym programem-klientem, wykorzystującym bibliotekę libfstorage(++). W katalogu „examples” znajdują się przykłady programów w różnych językach programowania (w przypadku Python oraz PHP wymagana jest instalacja dodatkowego oprogramowania – pyfstorage lub phpfstorage). Prostym przykładem będzie wykorzystanie programu-klienta napisanego w C. W pierwszej kolejności należy przejść do katalogu „examples/C” a następnie skompilować przykładowy program i uruchomić:

```
$ cd {fstorage_sources}/examples/C
$ gcc example1.c -o example1 -lfstorage
$ ./example1
```