Sprawozdanie z pracy w grupie „Pi of the Sky”

Przemysław Trędak

25 czerwca 2010

1 Wprowadzenie

Projekt „Pi of the Sky” zajmuje się głównie znajdowaniem, a także badaniem rozbłysków gamma. Używane do tego celu kamery, ze względu na dużo większe pole widzenia niż tradycyjne teleskopy, filmują nocne niebo. Zrobione w ten sposób zdjęcia są następnie przetwarzane, aby oddzielić interesujące przypadki od tych nieciekawych.

Jednak ze względu na potrzebę dobrych warunków do obserwacji, nie można ich robić na miejscu. Poprzednia aparatura do eksperymentu „Pi of the Sky” znajdowała się w Chile, obecnie trwają prace nad montażem nowej, wyposażonej w więcej kamer, a przez to zwiększającej możliwości obserwacyjne. Z tym brakiem dostępu do sprzętu wiąże się wiele problemów, które ograniczyć może jedynie nieustanna kontrola wszystkich podzespołów tego skomplikowanego systemu. Nie może tego jednak robić człowiek, więc do tej kontroli należy wykorzystać system informatyczny.

W projekcie „Pi of the Sky” do monitoringu urządzeń wybrano system Nagios, który jest szeroko stosowany do tego typu zadań. Jest to narzędzie wciąż rozwijane, istnieje też szeroka gama „wtyczek”, które pozwalają na poszerzenie jego funkcjonalności.

W eksperymencie „Pi of the Sky” do zbierania danych, nadesłanych przez system Nagios, zamierzano użyć bazy danych PostgreSQL. Jednak okazało się, że nie istnieje do tej pory żadna wtyczka potrafiąca przesyłać dane z Nagiosa do tego typu bazy, jedyna istniejąca potrafi komunikować się jedynie z bazą MySQL. Jako, że zmiana bazy danych nie była możliwa, moim zadaniem było napisanie programu potrafiącego przesyłać informacje pomiędzy Nagiosem a bazą PostgreSQL, a w przyszłości być może innymi rodzajami baz danych.

2 Specyfikacja programu

Po prześledzeniu kodu istniejącej wtyczki, piszącej do bazy MySQL, doszedłem do wniosku, że sposób, w jaki ona działa jest stosunkowo prosty: Nagios wszystkie istotne informacje o monitorowanych urządzeniach i o swojej aktywności zapisuje do kilku plików: `nagios.log` i `status.dat`, które odczytuje owa wtyczka i zawarte w nich informacje zapisuje do bazy. Napisany przeze mnie program działa w ten sam sposób. Jest on jednak napisany w języku wyższego poziomu, jakim jest Java, przez co ewentualna zmiana rodzaju bazy, z którą się on komunikuje nie jest żadnym problemem.

Plik `nagios.log` ma dość prostą strukturę - każda linijka to osobny wpis, zaczynający się od określenia czasu zdarzenia, później jego kategorii (których jest kilka, np. `SERVICE STATUS` czy `HOST STATUS`), jego parametrów i słownego określenia tego, co się stało. Wszystkie te rodzaje zdarzeń są zapisywane przez mój program do osobnych tabel w bazie danych, co pozwala na łatwe znalezienie tych interesujących.

```
[1275084000] LOG ROTATION: DAILY
[1275084000] LOG VERSION: 2.0
[1275084000] CURRENT HOST STATE: localhost;UP;HARD;1;
PING OK - Packet loss = 0%, RTA = 0.15 ms
[1275084000] CURRENT SERVICE STATE: localhost;Current Load;OK;
HARD;1;OK - load average: 0.01, 0.03, 0.07
[1275084000] CURRENT SERVICE STATE: localhost;Current Users;OK;
HARD;1;USERS OK - 3 users currently logged in
[1275084000] CURRENT SERVICE STATE: localhost;HTTP;OK;HARD;1;
HTTP OK: HTTP/1.1 200 OK - 260 bytes in 0,004 second response time
```

Rysunek 1: Przykładowy fragment pliku nagios.log

Plik `status.dat` ma trochę inną specyfikę - zapisywane są tam wszystkie dane zebrane przez Nagiosa, nawet jeśli nie miały one charakteru zdarzenia (np. jeśli stan jakiejś usługi nie zmienił się od jednego do drugiego sprawdzenia, nie pojawi się wpis w pliku `nagios.log`, natomiast fakt przeprowadzenia takiego pomiaru zostanie odnotowany w `status.dat`). Podzielony jest na sekcje związane z każdym hostem i każdą uruchomioną usługą.

Informacje z obu tych plików (a także z archiwów Nagiosa, do których trafiają pliki `nagios.log` po 24 godzinach użytkowania), są przez mój program procesowane, a następnie znajdujące się tam dane są zapisywane w danej bazie. Ze względu na to, że program działa jako daemon, tzn. przez cały czas bez udziału użytkownika, wszystkie dane o ewentualnych błędach

```
hoststatus {
host_name=localhost
modified_attributes=0
check_command=check-host-alive
check_period=24x7
notification_period=workhours
check_interval=5.000000
retry_interval=1.000000
event_handler=
has_been_checked=1
should_be_scheduled=1
check_execution_time=6.255
check_latency=0.194
check_type=0
current_state=0
last_hard_state=0
last_event_id=18
current_event_id=20
current_problem_id=0
last_problem_id=0
plugin_output=PING OK - Packet loss = 0%, RTA = 0.16 ms
long_plugin_output=
performance_data=rta=0.155000ms;3000.000000;
5000.000000;0.000000 pl=0%;80;100;0

                                    ⋮

}
```

Rysunek 2: Przykładowy fragment pliku status.dat

związanych np. z brakiem dostępu do bazy danych czy problemem z dostępem do jakiegoś pliku są odnotowywane w odpowiednim pliku.

3 Obsługa programu

Obsługa programu jest bardzo prosta - razem z programem jest dostarczony skrypt uruchamiający program, w którym należy ustawić jedynie ścieżkę do folderu `var` Nagiosa (znajdującego się wprost w głównym folderze Nagiosa,

dla domyślnej instalacji system Nagios jest to `/usr/local/nagios/var`) i do sterownika jdbc, o którym powiem później. Program startuje jako daemon, jeżeli wszystko zakończy się powodzeniem, w katalogu zawierającym program pojawi się plik `activity_data uruchomienia.log`, w którym znajdzie się informacja o udanym zakończeniu procedury odłączania od konsoli. Dzięki temu program jest w stanie działać w tle bez interakcji z użytkownikiem, a także po wylogowaniu się danego użytkownika.

Do poprawnego działania programu potrzebna jest maszyna wirtualna Javy, sterownik JDBC PostgreSQL, np. postgresql-8.1.414.jdbc3.jar, do którego należy podać ścieżkę w skrypcie uruchomieniowym (lokalizacja domyślna ustawiona w skrypcie to `/usr/local/pgsql/share/postgresql-8.1.414.jdbc3.jar`). Program należy uruchamiać po uruchomieniu systemu Nagios.

Po uruchomieniu program co 10 sekund sprawdza, czy nie wystąpiły jakieś zmiany w odczytywanych plikach, i stara się zapisać je do bazy danych. Zapisuje przy tym datę ostatniego udanego wpisu do bazy, dzięki czemu nawet w przypadku awarii bazy danych trwającej dość długo, by odpowiedni plik został przeniesiony do archiwum, przy następnym udanym połączeniu z bazą zmiany i tak zostaną naniesione.

Ze względów bezpieczeństwa użytkownik i hasło dostępu do bazy danych są na stałe wpisane w kod programu, więc zmiana tych parametrów wymaga ponownej kompilacji kodu, jednak zmiana tego stanu rzeczy nie jest trudna.

4 Podsumowanie

Napisałem program, który stanowi pomost pomiędzy systemem monitoringu Nagios, a używaną w eksperymencie „Pi of the Sky” bazą danych PostgreSQL. Było to niezbędnym krokiem w stronę stworzenia nowej aparatury do badania błysków gamma.