Mariusz Frydrych, Wojciech Horzelski
Wydział Informatyki i Zarządzania
Wyższa Szkoła Informatyki w Łodzi
Wydział Matematyki i Informatyki
Uniwersytet Łódzki

BAZODANOWA LOGIKA SKRZYNEK POCZTY ELEKTRONICZNEJ

Streszczenie – W pracy przedstawiono ideę poczty elektronicznej, w którym system skrzynek pocztowych jest realizowany w postaci bazy danych., a nie w natywnym systemie plików tak jak to ma miejsce w klasycznych systemach pocztowych. Każda wiadomość jest rozkładana na składowe charakterystyczne dla poczty elektronicznej i przechowywana w relacyjnej bazie danych. Z systemem sprzęgnięte są dwa interfejsy protokołów odbioru poczty (dla MUA) IMAP i POP3, dwa dla serwerów (MTA) SMTP i LMTP oraz interfejsy do konfiguracji ustawień i zarządzania zasobami systemu. System został przetestowany i wdrożony w jednostce średniej wielkości z około 15 tysiącami użytkowników w warunkach produkcyjnych.

1 System poczty elektronicznej

Poczta elektroniczna jest jedną z najwcześniejszych i najbardziej powszechnych usług współczesnych sieci komputerowych. Została opracowana w roku 1965 przez Louisa Pouzina, Glenda Schroedera i Pata Crismana. Początkowo służyła jedynie do przesyłania wiadomości pomiędzy użytkownikami tego samego komputera. W 1971 roku Ray Tomlinson rozszerzył ją na użytkowników zdalnych maszyn, i dodał znak @ do rozdzielania nazwy użytkownika od nazwy komputera.

Początkowo do wysyłania poczty służyły protokoły komunikacyjne *CPYNET, FTP, UUCP* i kilka innych. W 1982 roku Jon Postel opracował dedykowany protokół *SMTP* (*Simple Mail Transfer Protocol*) pozwalający na skuteczne dostarczenie wiadomości do właściwej maszyny [1].

Organizacja odbierania i przechowywania wiadomości leży po stronie hosta odbierającego i zazwyczaj są to pliki zawierające wszystkie wiadomości danego użytkownika (*mbox, mailspool*) lub pliki z każdą wiadomością z osobna (*maildir*). Do lepszej organizacji odbierania poczty stosuje się protokoły *POP3* (*Post Office Protocol ver. 3*) oraz *IMAP* (*Internet Message Access Protocol*) [2].

Zaawansowane systemy poczty elektronicznej umożliwiają użytkownikom dostęp do zasobów firmowych w dowolnym czasie i z dowolnego miejsca. Nieodzownym atrybutem takich systemów jest zapewnienie odpowiedniego poziomu bezpieczeństwa, dostępności oraz niezawodności. Należy przy tym dążyć, aby taki system był jak najłatwiejszy w administracji i najbardziej opłacalny. W dzisiejszym świecie trudno wyobrazić sobie funkcjonowanie jakiejkolwiek instytucji bez własnego systemu obsługi poczty elektronicznej.

Elementami systemy poczty elektronicznej są zazwyczaj serwery poczty MTA *(Mail Transfer Agent, np. Exim, SendMail, Posfix, Qmail, Exchange)* oraz klienci poczty MUA *(Mail User Agent, np. Thunderbird, Evolution, Microsoft Outlook).*

Serwery *MTA* wykorzystują protokół komunikacyjny poczty wychodzącej *SMTP*, z kolei klienci poczty *MUA* wykorzystują zarówno protokół serwerowy (*SMTP*) jak i protokoły tzw. poczty przychodzącej (np. *POP3* czy *IMAP*). Prowadzi to czasami do nieprecyzyjnego rozumienia problemu transferu poczty elektronicznej przez jej mniej doświadczonych użytkowników. Mianowicie, ten sam protokół *SMTP* służy zarówno do transferu poczty przez serwery (*MTA*), jak i jej wysyłania przez użytkowników, za pomocą klientów (*MUA*). Jest to po części zrozumiałe, ponieważ nie wykształcił się jeszcze „wydelegowany" protokół wysyłania poczty przez klientów pocztowych. Co prawda, niektóre serwery *MTA* (np. Exim) „potrafią" odróżnić rodzaj klienta po drugiej stronie, tzn. czy jest to inny serwer *MTA* czy klient *MUA*, obsługując klienta w wyspecjalizowanym trybie tzw. submission mode [3 i 4]. Do takiej obsługi protokołu *SMTP* [5] już dawno wydelegowano oddzielny port TCP 587 (poza portem 25).

2 Bazodanowa organizacja systemu pocztowego

Przedstawimy teraz pomysł systemu bazodanowego organizacji skrzynek pocztowych. Architektura taka oparta jest o system bazy danych sprzęgniętej za pomocą dobrze zdefiniowanych interfejsów z elementami „klasycznej" poczty elektronicznej. Cała logika użytkowników, ról, skrzynek, aliasów, przekazywania (*forwarding*) oraz medium magazynowania wiadomości wraz ze strukturą nagłówków i załączników została tu zaimplementowana w dość złożonej strukturze relacyjnej bazy danych.

Bazodanowy system doskonale się sprawdza w środowisku poczty „korporacyjnej" czy „hostingowej", tzn. w środowisku ze znaczną ilością użytkowników, w którym zajmowane stanowiska nie posiadają cech trwałości, co prowadzi do wykształcenia relacji pomiędzy „prawdziwymi" użytkownikami, a ich rolami zwanymi „wirtualnymi" użytkownikami

systemu. Środowiska tego typu są charakterystyczne dla instytucji o charakterze edukacyjnym (np. wyższe uczelnie, instytucję szkoleniowe).

Dodatkowo podejście bazodanowe (listy są rozkładane na elementy i składowane w tabelach) pozwala na łatwe przeszukiwanie całości poczty i jej analizę.

Można go łatwo wpisać w strukturę klastrową - wiele serwerów *MTA* może być w relacji z wieloma bazami danych, a cały klaster składa się w warstwie logicznej z połączenia tych elementów za pomocą odpowiednich interfejsów.

Nie do przecenienia jest też kwestia bezpieczeństwa tak skonstruowanego systemu, przede wszystkim użytkownicy nie mają dostępu do systemu plikowego, a jedynie dostęp pośredni przez mechanizmy bazy danych. Taka architektura znacznie upraszcza zarządzanie, archiwizację oraz migrację systemu, co z kolei znacznie ułatwia pracę administratorom.

W celu realizacji tych cech zaprojektowano system pocztowy o budowie modularnej, jego elementami są: serwery poczty przychodzącej z obsługą *IMAP oraz POP3*, serwer protokołu komunikacyjnego *LMTP* (lub *SMTP*), interfejs dla protokołu *SMTP*, moduł do zarządzania użytkownikami i ich zasobami, moduł do zarządzania zasobami baz danych, narzędzie do migracji systemu oraz sterowniki do silnika baz danych: *Sqlite, MySQL, PostgreSQL*.

Fundamentalnym elementem systemu jest ostatni z tych składników, tzn. interfejs pośredniczący pomiędzy warstwą poczty elektronicznej, a silnikiem baz danych.

Zarządzanie bazą możne odbywać się poprzez odpowiednie dla wybranego systemu narzędzia (np. *phpPgAdmin* dla bazy *PostgreSQL*). Z łatwością można zauważyć, że mamy oddzielne tabele dla użytkowników, ich skrzynek, folderów, aliasów, przekierowań, nagłówków i kopert wiadomości oraz „fizycznej” treści wraz z załącznikami. Pozwala to na transformację, z pozoru amorficznego obiektu, jaką jest wiadomość *email*, na w pełni restrukturyzowany element relacyjnej bazy danych. Interpretowanie zasobów poczty elektronicznej w postaci obiektów bazy danych, skutkuje znakomitą wydajnością systemu pocztowego, m.in. szybkim wyszukiwaniem kontekstowym, łatwym i efektywnym zarządzaniem, skalowalnością i elastycznością.

3 Przykład implementacji

Jednym z przykładów takiego podejścia jest open-source’owy system *DBmail*. W trakcie jego projektowania szczególny nacisk położony został na skalowalność, zarządzalność, wydajność, bezpieczeństwo i

elastyczność. W celu realizacji tych cech zaprojektowano system pocztowy o budowie modularnej.

System *DBmail* wdrożono w celach testowych w jednostce (wyższa uczelnia) z około 15 tysiącami użytkowników w warunkach produkcyjnych, jako równoległy system pocztowy. W tym celu wykorzystano komputer wyposażony w dwurdzeniowy procesor *AMD64* o mocy 1.86 GHz z pamięcią RAM 2GB, na którym został zainstalowany system operacyjny FreeBSD-7.2-STABLE.

System *DBmail* testowany był w okresie miesiąca równolegle z podstawowym systemem pocztowym. Obydwa systemy pocztowe były równomiernie obciążone

System sprzęgnięto z serwerem *MTA Exim* w wersji 4.69 i badano jego zachowanie z bazami danych *Sqlite3 3.6.19*, *PostgreSQL 8.4.2* oraz *MySQL 5.0.89*.

Demony nasłuchujące połączeń od *MUA dbmail-imapd* i *dbmailpop3d* oddzielono szyfrującym tunelem *SSL stunnel* w celu podwyższenia stopnia bezpieczeństwa. Sprzężenie systemu z serwerem *MTA Exim* dokonano na poziomie transportu za pomocą protokołu *LMTP* (*Local Mail TransferProtocol*) poprzez lokalne gniazdo uniksowe */var/tmp/dbmail-lmtpd.socket*. W tym celu wykorzystano element systemu *dbmail dbmail-lmtpd* jako tzw. *LDA* (*Local Delivery Agent*) lub *MDA* (*Mail Delivery Agent*). Osiągnięto to przez następującą konfigurację (fragment pliku konfiguracyjnego *Exim*) [4]:

```
localuser_dbmail:
              driver = accept
              transport = dbmail_delivery
              unseen = true

localuser_dovecot:
              driver = accept
              check_local_user
              transport = dovecot_delivery
              cannot_route_message = Unknown user
```

oraz odpowiednie konfiguracje transportów (*Exim*):

```
dovecot_delivery:
              driver = pipe
              command = /usr/local/libexec\
                      /dovecot/deliver
              message_prefix =
              message_suffix =
              log_output
              delivery_date_add
              envelope_to_add
```

```
                return_path_add
                group = mail
dbmail_delivery:
                driver = lmtp
                socket = /var/tmp/dbmail-lmtpd.socket
                delivery_date_add
                envelope_to_add
                return_path_add
                user = mailnull
                group = mail
```

Oprócz normalnej pracy z „prawdziwą” pocztą, system był testowany za pomocą programu generującego około 100 wiadomości na sekundę o losowej treści *test-mail.c*[6]:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <assert.h>
#include <unistd.h>
#define MAX_WORD 0377
typedef enum { false, true } boolean;
int draw(int, int);
char *rand_word(int, int, boolean);
main(int argc, char *argv[])
{
   char cmd[MAX_WORD + 1];
   int k, n;
   if (argc < 1 + 1) {
        fprintf(stderr,    "%s    num_e-mails\n\n",
   argv[0]);
        return 1;
        }
   srandom(time(0));
   n = atoi(argv[1]);
   for (k = 0; k < n; ++k) {
        FILE *po;
        int nw, nowrap = 0, nl;
```

```
            snprintf(cmd,   MAX_WORD,"mail   -s   \"test
      %010d some-host\" "
                                                "dbmailuse
                                          r@some-host",  k
                                          + 1);
            if (!(po = popen(cmd, "w"))) {
                  fprintf(stderr, "popen( \"%s\", \"w\"
            ): ""error!\n\n", cmd);
                  return 2;
                  }
            nw = draw(128, 512);
            printf("cmd: \"%s\",nw=%d\n", cmd, nw);
            while (nw-- > 0) {
                  ++nowrap;
                  nowrap %= 10;
                  nl = nowrap ? false : true;
                  fprintf(po,   nl   ?   "%s.\n"   :   "%s
            ",rand_word(3, 12, nowrap == 1)
                  );
            }
      fprintf(po, ".\n");
      pclose(po);
      printf("END.: cmd: \"%s\",nw=%d\n\n", cmd, nw);
      usleep(10000);
   }
   return 0;
}
int draw(int a, int b)
{
   return a + random() % (b - a + 1);
}
char *rand_word(int a, int b, boolean firstCapital)
{
   static char sbuff[MAX_WORD + 1];
```

```
   int len;
   char *p;
   assert(b < MAX_WORD);
   memset(sbuff, 0, sizeof sbuff);
   len = draw(a, b);
   assert(len > 0);
   p = sbuff;
   if (firstCapital) {
      *p++ = (char) draw('A', 'Z');
      --len;
      }
   while (len-- > 0) {
      *p++ = (char) draw('a', 'z');
      --len;
      }
   *p = '\0';
   return sbuff;
}
```

Przy bardzo dużym obciążeniu (praktycznie nie spotykanym w rzeczywistych warunkach pracy), system *DBmail* zachowywał dużą wydajność przy współpracy z bazą danych *PostgreSQL 8.4.2*, natomiast z bazami *MySQL* i *Sqlite3* następowała blokada bazy po kilkunastu wiadomościach. Serwer *MTA* zwracał tzw. „przejściowy” błąd o kodzie 430 (*transient negative completion reply*) w wyniku czego wiadomość wędrowała do kolejki *Exima*, by w następnym cyklu (tj. po około 30 minutach) zostać poprawnie przetworzona. Ilustrują to odpowiednie fragmenty dzienników systemu (kolejka *Exima*):

```
   some-host# exim -Mvl 1NgSt4-0005ZA-CE
            2010-02-14 02:03:02 Received from some-
            user@some-host
            U=some-user P=local
            S=2277 T="test 0000000100 some-host"
   2010-02-14 02:03:04 dbmailuser@some-host
            R=localuser_dbmail
            T=dbmail_delivery defer (-46):
            LMTP error after end of data:
            430 Message not received
```

oraz informacja interfejsu *dbmail-lmtpd*

```
some-host# tail -f /var/log/dbmail.err
Feb 14 02:13:18 some-host dbmail-lmtpd[20390]:
        Error:[sql] dbsqlite.c,db_query(+330):
        sqlite3_get_table failed:
        database is locked
Feb 14 02:13:18 some-host dbmail-lmtpd[20389]:
        Debug:[sql] dbsqlite.c,db_query(+324):
        database locked, retrying...
Feb 14 02:13:18 some-host dbmail-lmtpd[20390]:
        Debug:[db] dbmodule.c,db_query(+145):
        last query took [0] secondo
```

DBmail pracujący z bazą danych *PostgreSQL* pozostawił w kolejce niespełna 50 wiadomości, natomiast przy wykorzystaniu *Sqlite3* zakolejkowanych zostało prawie 1000 wiadomości (przy jednoczesnej obsłudze 1024 listów). Wyniki takie zostały osiągnięte przy niemalże identycznym obciążeniu systemu (co ilustrują poniższe fragmenty plików opisujących obciążenie zasobów systemowych):

```
#PostgreSQL
real 191.069
user 1.34
sys 3.86
      1404  maximum resident set size
        78  average shared memory size
       444  average unshared data size
       114  average unshared stack size
    260190  page reclaims
         0  page faults
         0  swaps
         3  block input operations
         8  block output operations
         0  messages sent
         0  messages received
         0  signals received
     12011  voluntary context switches
     48243  involuntary context switches

# Sqlite3
real 175.21
user 1.18
sys 3.68
      1404  maximum resident set size
        76  average shared memory size
       436  average unshared data size
```

```
       111  average unshared stack size
    260190  page reclaims
         0  page faults
         0  swaps
         0  block input operations
         0  block output operations
         0  messages sent
         0  messages received
         0  signals received
     10989  voluntary context switches
     13388  involuntary context switches
```

4 Wnioski

Przedstawiona tu organizacja poczty e-mail stanowi funkcjonalne rozwiązanie do zarządzania pocztą użytkowników w rozbudowanych instytucjach. Przeprowadzone w środowisku produkcyjnym testy potwierdziły jego skuteczność oraz wydajność przy zachowaniu cech założonych przy projektowaniu systemu.

W systemach poczty typu „korporacyjnego” czy „hostingowego” jako końcowego transportu serwera *MTA* (tzw. *LDA* lub *MDA*) korzystniej jest użyć elementu systemu leżącego „bliżej” serwera odbioru poczty (jak IMAP i POP3), ponieważ ten element rozkłada, indeksuje i przechowuje wiadomości w skrzynkach i folderach użytkowników. Wygenerowane indeksy są potem wykorzystywane przez bliźniaczy element systemu przy przeglądaniu i organizacji folderów pocztowych przez użytkowników. Takie postępowanie znacznie skraca obsługę poczty przez klientów *MUA*. Ponadto taka architektura systemu pozwala w łatwy sposób zbudować bardziej skomplikowaną strukturę klastra pocztowego.

Literatura

[1] Hazel P., The Exim SMTP Mail Server. UIT Cambridge Ltd., 2007.

[2] Wood D., Programming Internet Mail. O’Reilly, 1999.

[3] Hughes L., Internet e-mail Protocols, Standards and Implementation. Artech House Publishers, 1998.

[4] Loshin P., Essential Email Standards: RFCs and Protocols Made Practical. John Wiley and Sons, 1999.

[5] Klensin J., Simple Mail Transfer Protocol (RFC 2821), 2001.

[6] Rhoton J., Programmer’s Guide to Internet Mail: SMTP, POP, IMAP, and LDAP. Digital Press, 1999.

DATABASE LOGIC OF EMAIL BOXES

Summary – The paper presents an idea of an email system where storage of messages is entirely implemented on the database system. Each message is splitted into components specific to e-mail and stored in a relational database The system is connected with two MUA interfaces (IMAP and POP3) and two MTA interfaces (SMTP and LMTP) for easier configuration and system management. The presented system has been successfully implemented and tested in an average volume production environment of about 15 thousands of users.