STUDIA INFORMATICA 2010
Volume 31 Number 2B (90)

Dariusz R. AUGUSTYN, Łukasz WARCHAŁ
Politechnika Śląska, Instytut Informatyki

UDOSTĘPNIANIE ZDALNYCH USŁUG OBLICZEŃ STATYSTYCZNYCH Z UŻYCIEM WIELOWARSTWOWEGO SYSTEMU SERVER

Streszczenie. Artykuł prezentuje system ServeR przeznaczony do udostępniania zdalnych usług statystycznych, realizowanych w ramach modułu R-project. Wykorzystanie wielowarstwowej architektury systemu oraz użycie technologii Windows Communication Foundation pozwala na instalację w wielu różnych konfiguracjach, dostosowanych do wybranej infrastruktury sieciowej. Dzięki temu system jest skalowalny i w pewnym zakresie odporny na awarie. Artykuł pokazuje szczegółowo praktyczne przykłady zastosowania systemu ServeR, pracującego w dwóch trybach – on-line (wsadowy) oraz on-line (zdalna R-sesja). W artykule omówiono nowe rozszerzenie systemu ServeR w zakresie bezpieczeństwa przetwarzania, polegające na użyciu mechanizmu izolowanych magazynów przez procesy obliczeń statystycznych.

Słowa kluczowe: udostępnianie usług obliczeń statystycznych, zastosowanie R-project, infrastruktura Windows Communication Foundation, bezpieczeństwo zdalnego przetwarzania, izolowane magazyny

ENABLING REMOTE STATISTICAL CALCULATION SERVICES BY USING THE MULTILAYER SERVER SYSTEM

Summary. The paper presents ServeR – the system for enabling remote statistical calculation supported by R-project. Multilayer architecture and Windows Communication Foudantion technology usage allow to deploy the system in many configurations adapted to network infrastructure. This makes the system scalable and fault tolerant in some aspect. Usage of ServeR in two modes – off-line (batch mode) and on-line (remote R-sesssion) is shown in detailed examples of statistical tasks. A new feature for increasing system security – an isolated storage mechanism used by statistical task processes – is considered.

Keywords: enabling statistic calculation services, application of R-project, Windows Communication Foundation infrastructure, security of remote processing, isolated storages

1. Wymagania dotyczące systemu udostępnienia usług obliczeń statystycznych

Dziedzinowe systemy informatyczne, na odpowiednio zaawansowanym poziomie rozwoju, rozszerzane są o funkcjonalności związane z analizą danych. W szczególności systemy medyczne rozbudowywane są o moduły realizujące zadania analityczne i statystyczne. Udostępnienie takiej funkcjonalności, np. w Internecie, przyciąga użytkowników zainteresowanych zaawansowaną analizą danych (danych swoich, dostarczonych do sytemu, jak i danych referencyjnych, udostępnianych przez medyczny system dziedzinowy). W ten sposób budowana jest swoista społeczność, zainteresowana dalszym rozwojem całego systemu.

Obecnie jednym z najpopularniejszych wśród społeczności statystyków medycznych narzędzi programowych, realizujących zadania statystyczne, jest R-project [1, 2], udostępniany na zasadzie otwartego oprogramowania. Stąd integracja z modułem R-project jest m.in. przedmiotem niniejszej pracy.

Artykuł opisuje utworzony system informatyczny – ServeR – wykonany w technologiach firmy Microsoft, udostępniający usługi obliczeń statystycznych R-project i pozwalający na łatwą integrację tychże usług z dziedzinowymi systemami informatycznymi, wykonanymi w technologiach bazujących na .NET (lub obsługujących standardy WS-*, SOAP i HTTP).

Wymagania dotyczące systemu wynikały z założeń przyjętych w projekcie „Bon na Innowacje" BNI/0004/2008, organizowanym przez PARP (Polska Agencja Rozwoju Przedsiębiorczości), z którego finansowane były badania nad architekturą oraz prace związane z wytworzeniem i uruchomieniem omawianego systemu ServeR. Podstawowe wymagania dla systemu ServeR są następujące [4]:

a) wykorzystanie technologii .NET,
b) użycie R-project w zakresie realizowanych usług obliczeń statystycznych,
c) możliwość zlecania zadań obliczeń wyrażonych w postaci R-skryptów,
d) zastosowanie wielowarstwowej architektury systemu. Utworzenie niezależnych warstw systemu:
   - klienta,
   - serwera komunikacyjnego (obsługującego żądania aplikacji klienta, odsyłającego odpowiedzi),
   - serwera obliczeniowego (realizującego obliczenia statystyczne w oparciu o system R-project),
e) wykorzystanie technologii WCF (ang. Windows Communication Foundation [5, 6]) w zakresie komunikacji międzywarstwowej oprogramowania wchodzącego w skład infrastruktury systemu,

f) zapewnienie architektury pozwalającej na łatwe (półautomatyczne) wygenerowanie aplikacji klienckiej lub biblioteki programowej, pozwalającej na użycie udostępnianych usług statystycznych,
g) możliwość pracy w trybie wsadowym (ang. off-line) – zlecenie – rozłączenie – połączenie – odbiór wyników,
h) możliwość pracy w trybie połączeniowym (ang. on-line) – tryb zdalnej konsoli systemu R, możliwość interpretacji poleceń R, zadawanych ad hoc, z natychmiastową realizacją i dostarczeniem wyników,
i) możliwość wykonywania zadań obliczeń statystycznych w oparciu o dane przesłane wraz ze zleceniem, jak i dane zgromadzone w relacyjnej bazie medycznego systemu dziedzinowego,
j) zwiększenie bezpieczeństwa przetwarzania w ramach procesu serwera obliczeniowego m.in. poprzez nakładanie ograniczeń na proces obliczeń statystycznych, którego pliki robocze nie tylko mogą być umieszczone w ramach tradycyjnego systemu plików, lecz także w zakresie tzw. izolowanych magazynów (ang. *isolated storage*) [7].
k) wykonanie i weryfikacja prototypowych modułów aplikacji klienckiej dla zlecania i odbierania wyników realizowanych zadań statystycznych (realizacja w 2 technologiach: grubego klienta (WinForms lub WPF – Windows Presentation Foundation) oraz cienkiego klienta (ASP.NET),
l) możliwość udostępniania skryptów realizujących określone zadania statystyczne innym użytkownikom systemu.

Wymaganie (a) wynika z założeń pochodzących od zamawiającego (firmy tworzącej medyczny system dziedzinowy).

Wymagania (b) i (c) są odpowiedzią na zapotrzebowanie społeczności statystyków medycznych (popularność wieloplatformowego R-project). Punkt (c) prowadzi wprost do takiej koncepcji systemu, w której należy realizować całe R-skrypty (uwzględnienie powszechnej znajomości języka skryptowego R-project przez użytkowników), a tym samym eliminuje takie podejście, w którym system udostępniałby pojedyncze wybrane funkcje R-project, np. poprzez mechanizm usług WebService.

Realizacja wymagania (h) dotyczącego trybu on-line, pozwala na wykorzystanie systemu do uruchamiania i testowania zadań – użytkownik, uruchamiając pojedyncze komendy, może testować fragmenty złożonych R-skryptów (wykorzystanych później w trybie off-line).

Wielowarstwowa architektura systemu (punkt (d)), pozwalając na niezależne uruchamianie poszczególnych komponentów, prowadzi do spełnienia wymagania o skalowalności systemu. Daje to możliwość zmulitiplikowania instancji serwera obliczeniowego (uruchomienia procesów obliczeniowych w wielu węzłach sieci komputerowej) w celu zwiększenia mocy przetwarzania. Możliwe staje się też podwyższenie dostępności systemu poprzez zwielokrot-

nienie instancji serwera komunikacyjnego. Niezależność komponentów: serwera komunikacyjnego i obliczeniowego pozwala, w szczególności, na pracę całego systemu nawet przy przeciążonym lub chwilowo wyłączonym serwerze obliczeniowym (zapewnienie określonego poziomu odporności na awarie).

Zastosowanie technologii WCF w komunikacji pomiędzy aplikacją klienta – serwerem komunikacyjnym – serwerem obliczeniowym umożliwia pracę systemu w różnych konfiguracjach, pozwalając na niezależność od zastosowanego protokołu komunikacyjnego. Generalnie zakłada się, że mechanizm komunikacji w systemie ServeR oparty jest na HTTP/SOAP i WebSevices, ale dzięki WCF możliwe są inne warianty, np.:

A. możliwość wykorzystania formatu binarnego i protokołu TCP w sieciowej komunikacji pomiędzy serwerem komunikacyjnym i obliczeniowym (jeśli oba komponenty są uruchomione w węzłach sieci lokalnej),
B. możliwość wykorzystania mechanizmów komunikacji międzyprocesorowej (potoki nazwane – ang. named pipes) pomiędzy aplikacją klienta i serwerem komunikacyjnym (jeśli oba procesy są uruchomione na jednym komputerze, a może występować to, gdy klient jest aplikacją WWW wykonaną w technologii ASP.NET),
C. możliwość wykorzystania systemu kolejkowego (Microsoft Message Queuing [8]) w komunikacji pomiędzy serwerem komunikacyjnym i obliczeniowym.

Warianty (A) i (B) służą do podniesienia efektywności systemu. Wariant (C) zapewnia wymagany w punkcie (g), wsadowy tryb pracy (poprzez umożliwienie asynchronicznego działania serwera komunikacyjnego i obliczeniowego).

Skutkiem spełnienia wymagania (k) było powstanie konkretnych, prototypowych modułów programowych, których przykłady użycia w trybie zdalnej R-sesji oraz w trybie wsadowym pokazano w podrozdziałach 3.1 i 3.2.

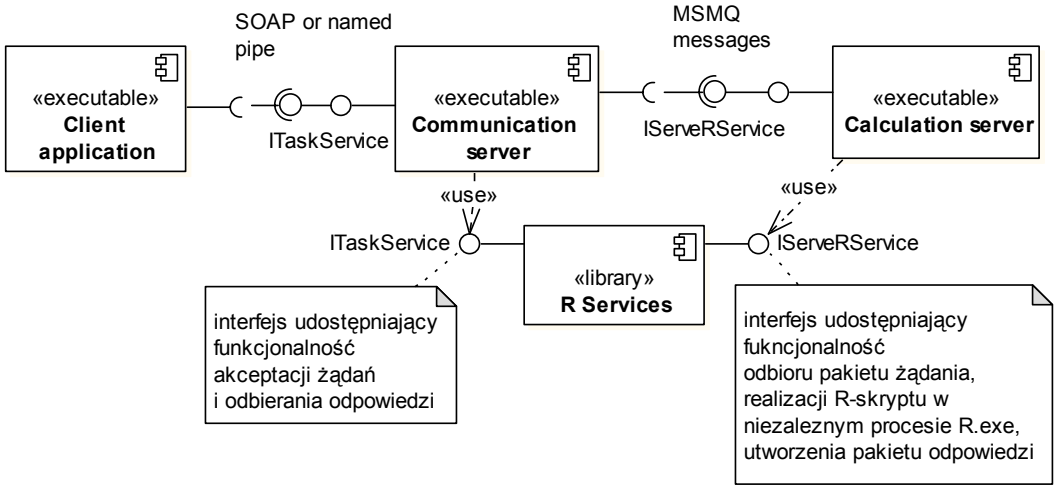
Założenie (k), którego realizacja stanowi rozszerzenie systemu ServeR w stosunku do wersji opisanej w [4], pozwoliło na zabezpieczenie systemu przed nieprawidłowymi R-skryptami. Zastosowanie izolowanych magazynów, opisane w rozdziale 4, pozwoliło na ustawianie ograniczeń rozmiaru przestrzeni dyskowej, przyznawanej dla konkretnego procesu obliczeniowego. W szczególności pozwoliło to na wzajemną izolację przestrzeni, wykorzystywanej przez poszczególne, niezależne zadania (np. zadania zlecane przez różnych użytkowników).

Spełnienie wymagania (i) w zakresie dostępu z poziomu R-skryptów do dziedzinowej, medycznej bazy danych, zostało zrealizowane dzięki rozszerzeniu modułu R o standardową bibliotekę dostępu do relacyjnej bazy danych (tutaj obsługiwanej przez SZBD MS SQLServer), poprzez interfejs ODBC [3]. Przykład użycia, opisany w podrozdziale 3.2, dotyczy właśnie takiego zadania obliczeń statystycznych, które wykorzystuje dziedzinową bazę danych w roli zbioru referencyjnego.

Celem pracy jest pokazanie łatwości praktycznego użycia, stworzonej od podstaw, złożonej infrastruktury, zorientowanej na usługi – systemu ServeR, bazującego na mechanizmach udostępnianych przez technologię komunikacyjną WCF. Przykłady dotyczące pracy w trybie wsadowym i konsolowym (ukrywające szczegóły komunikacji międzywarstwowej w ramach ServeR-a) pozwalają na pokazanie uniwersalności zaproponowanego rozwiązania.

2. ServeR – architektura rozwiązania

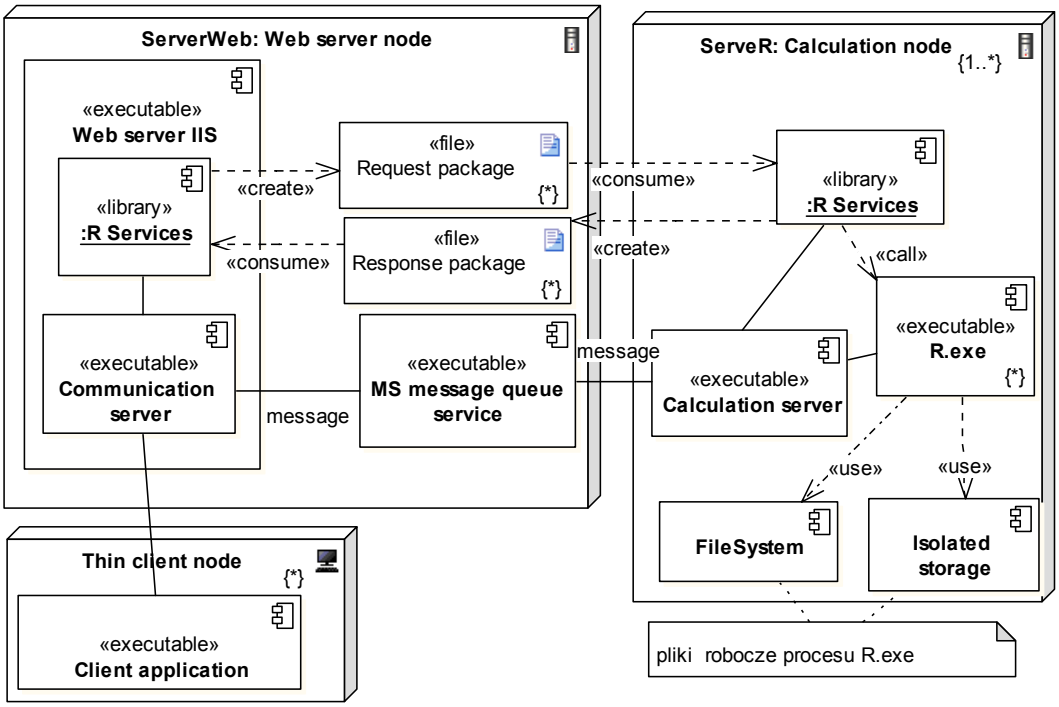
Architektura systemu ServeR została pokazana na rys. 1.



Rys. 1. Diagram podstawowych komponentów systemu ServeR
Fig. 1. Diagram of base components of ServeR system

Typowa konfiguracja systemu ServerR, przeznaczonego do pracy w trybie off-line, została pokazana na diagramie wdrożenia z rys. 2. Aplikacja klienta, zlecająca zadanie, uzyskuje od serwera komunikacyjnego unikalny identyfikator zadania. Przekazywane jest ono w formie skompresowanego pakietu żądania (plik), zawierającego R-skrypty do uruchomienia oraz ewentualne pliki z danymi wejściowymi. Pakiet umieszczany jest we współdzielonym systemie plików komputera, gdzie uruchomiony jest proces serwera komunikacyjnego. Odpowiednia informacja o żądaniu umieszczana jest w kolejce komunikatów. Serwer obliczeniowy odbiera komunikat z kolejki i pobiera pakiet żądania, po czym uruchamia odpowiedni proces silnika obliczeń statystycznych (stworzenie nowej instancji R.exe albo wykorzystanie instancji z dostępnej puli). Proces R.exe, działając w przypisanej przestrzeni roboczej (albo system plików, albo magazyn izolowany), realizuje przesłane R-skrypty. Po zakończeniu działania R.exe, serwer obliczeniowy tworzy i odsyła skompresowany pakiet odpowiedzi (plik) do systemu plików, gdzie uruchomiony jest serwer komunikacyjny. W pakiecie odpowiedzi umieszczony jest plik z zawartością standardowego wyjścia i ewentualnie wynikowe pliki graficzne lub pliki z danymi. Jednocześnie serwer obliczeniowy umieszcza odpowiedni

komunikat zwrotny w kolejce w celu poinformowania serwera komunikacyjnego o zakończeniu zadania. Aplikacja klienta, działając w trybie odpytywania serwera komunikacyjnego, wykrywa (po pewnym czasie) stan gotowości zadania o podanym identyfikatorze i odbiera pakiet odpowiedzi.

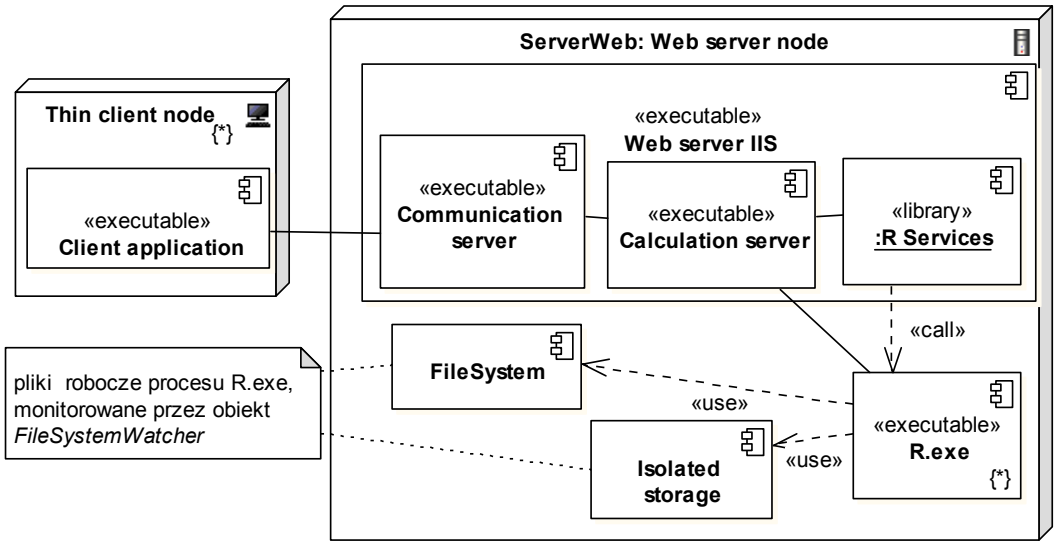


Rys. 2. Diagram wdrożenia systemu ServeR pokazujący konfigurację w trybie wsadowym
Fig. 2. Deployment diagram of ServeR for off-line mode configuration

W trybie bezpośrednim - on-line - serwer komunikacyjny i obliczeniowy działają na tym samym komputerze. Taka konfiguracja pozwala na efektywne zasymulowanie interaktywnego sposobu pracy i nazwana jest zdalną R-sesją. Konfiguracja ta pokazana została na diagramie wdrożenia na rys. 3.

W trybie R-sesji użytkownik może uruchamiać pojedyncze komendy. Dzięki połączeniu aplikacji klienta i serwera komunikacyjnego w trybie dwukierunkowym (tryb WFC – wsDualHttpBinding [5, 6]), wyniki wysyłane przez zdalny proces R.exe na standardowe wyjście są od razu przekazywane do aplikacji klienta. Dodatkowo, dzięki zastosowaniu komponentu System.IO.FileSystemWatcher [9], katalog roboczy, w którym działa zdalny proces R.exe, jest stale monitorowany i jeśli realizowany R-skrypt utworzy/zmieni jakikolwiek plik wynikowy (graficzny czy z danymi wynikowymi), to jego odpowiednik (z dokładnością do nazwy i zawartości) zostanie automatycznie utworzony/zmieniony lokalnie po stronie aplikacji klienta. Takie funkcjonowanie ServeR-a pozwala użytkownikowi odnieść wrażenie pracy z lokalnym modułem R.exe.

Szczegółowo architektura systemu ServeR została opisana w [4].



Rys. 3. Diagram wdrożenia systemu ServeR pokazujący konfigurację w trybie on-line (zdalna R-sesja)
Fig. 3. Deployment diagram of ServeR for off-line mode configuration (remote R-session)

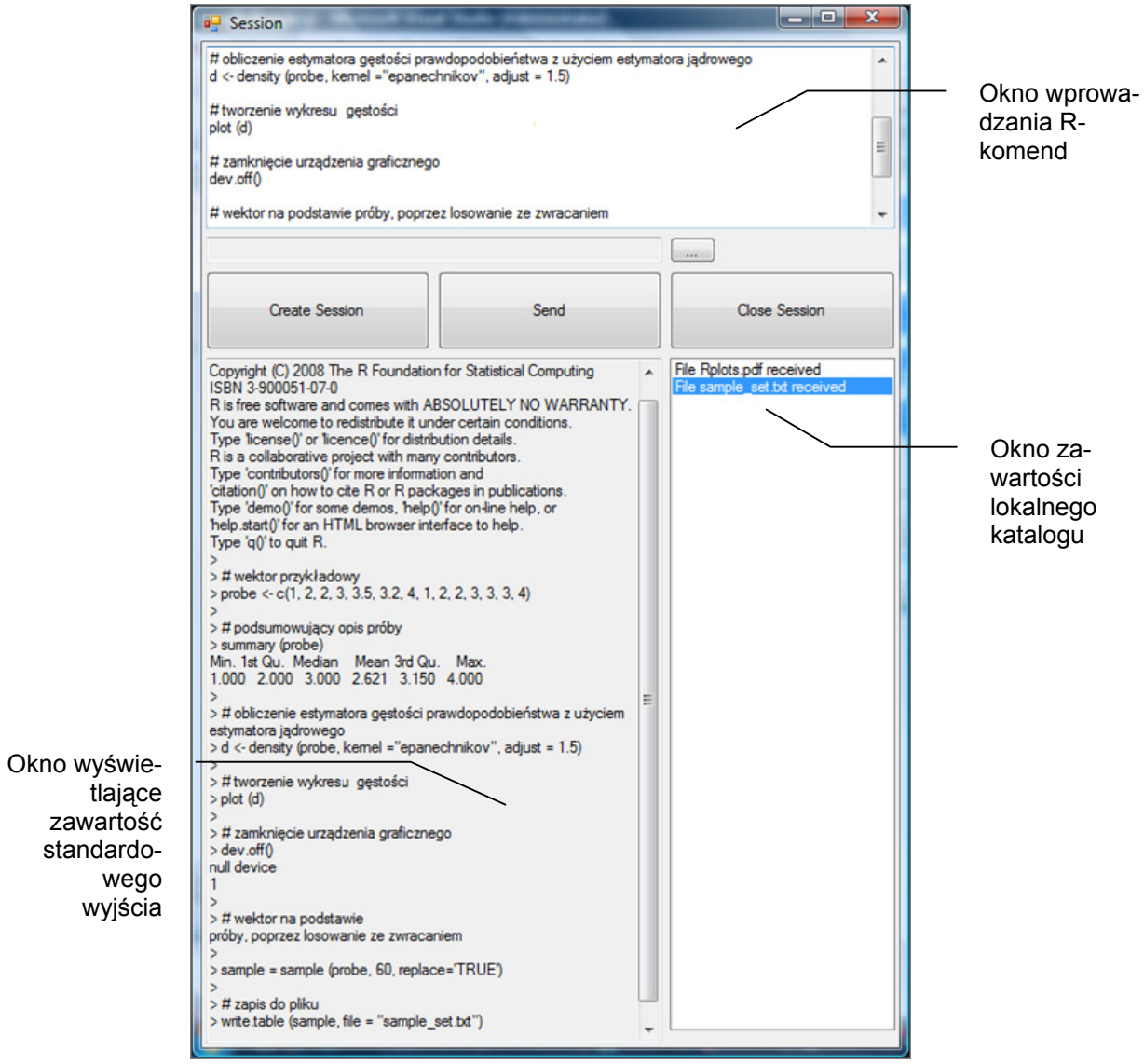
3. Przykłady użycia systemu ServeR

Wykorzystanie systemu ServeR zostanie pokazane na przykładach:

- wykonania sekwencji komend w trybie zdalnej R-konsoli (tryb on-line),
- zlecenia realizacji zadania wsadowego (tryb off-line), wyrażonego R-skryptem, korzystającego z danych dostarczonych oraz danych z medycznej bazy dziedzinowej, udostępnianych pośrednio poprzez infrastrukturę ServeR.

3.1. Praca w trybie zdalnej konsoli systemu R – przykład użycia

Rys. 4 pokazuje ekran interfejsu użytkownika prototypowej aplikacji klienta (wykonany w technologii grubego klienta – WinForms). Użytkownik wprowadza dane obserwacyjne, wyznacza podstawowe charakterystyki próby, tworzy estymator funkcji gęstości prawdopo-

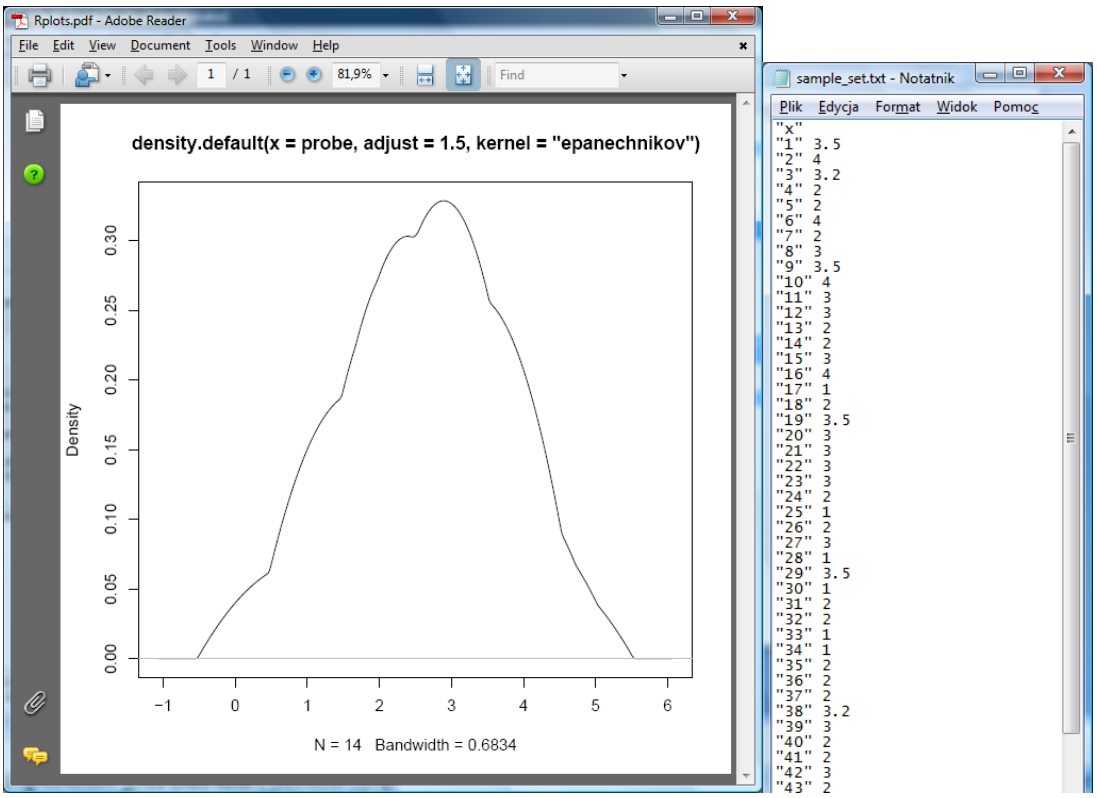


Rys. 4. Przykład użycia aplikacji implementującej funkcjonalność zdalnej R-sesji
Fig. 4. Sample usage of client application that implementing remote R-session functionality

dobieństwa, bazując na kwadratowym jądrze Epanechnikova, wyznacza wykres estymowanej funkcji gęstości, tworzy nowy zbiór danych syntetycznych poprzez realizację losowania ze zwracaniem ze zbioru początkowego. Użytkownik wprowadza do okna R-komend pojedyncze następujące rozkazy:

```
# wektor przykładowy
probe <- c(1, 2, 2, 3, 3.5, 3.2, 4, 1, 2, 2, 3, 3, 3, 4)
# podsumowujący opis próby
summary (probe)
# obliczenie estymatora gęstości prawdopodobieństwa z użyciem estymatora jądrowego
d <- density (probe, kernel ="epanechnikov", adjust = 1.5)
# tworzenie wykresu gęstości
plot (d)
# zamknięcie urządzenia graficznego
dev.off()
# wektor na podstawie próby, poprzez losowanie ze zwracaniem
sample = sample (probe, 60, replace='TRUE')
```

```
# zapis danych do pliku
write.table (sample, file = "sample_set.txt")
```



Rys. 5. Zawartość plików wynikowych – wykres jądrowego estymatora funkcji gęstości (Rplots.pdf)) i syntetyczne dane uzyskane na podstawie próby losowej (sample_set.txt)

Fig. 5. Content of result files – graph of kernel estimator of probability density function (Rplots.pdf) and synthetic data obtained form given simple random sample (sample_set.txt)

Jeśli rozkazy wyprowadzają jakieś napisy, to są one automatycznie (asynchronicznie w stosunku do ewentualnych interakcji użytkownika i aplikacji) wypisywane w oknie standardowego wyjścia. Komendy *dev.off* i *write.table* powodują utworzenie plików wynikowych po stronie systemu ServerR. Odpowiedniki tych plików (wykres – *Rplots.pdf* i dane – *sample_set.txt* z rys. 5) są tworzone po stronie aplikacji klienta (w lokalnym systemie plików), a informacja o tym zdarzeniu jest wyświetlana w oknie zawartości lokalnego katalogu. Ta operacja również odbywa się asynchronicznie w stosunku do ewentualnych interakcji użytkownika i aplikacji. Odświeżanie okna zawartości lokalnego katalogu odbywa się automatycznie i z inicjatywy serwera komunikacyjnego systemu ServeR. Wybranie odpowiedniego pliku (poprzez podwójne kliknięcie myszki na jego nazwie w oknie zawartości lokalnego katalogu) powoduje automatyczne uruchomienie odpowiedniego programu do przeglądnięcia pliku (np. viewer *Adobe Reader* dla wykresów, *notepad* dla danych z rys. 5). Taki tryb działania pozwala wykorzystać ServeR jako odpowiednik lokalnego modułu R-project. Jednak możliwe jest tu użycie skryptów bibliotecznych udostępnianych, współdzielonych przez użytkowników systemu ServeR oraz wykorzystanie referencyjnych danych ze zdalnej bazy

systemu dziedzinowego. Ta ostatnia opcja będzie pokazana w następnym przykładzie z podrozdziału 3.2.

3.2. Praca w trybie wsadowym – przykład użycia

Przykładem wykorzystania trybu wsadowego będzie, opisane poniżej, zadanie zbadania skuteczności zastosowania leku, poprzez porównanie wyników jego zastosowania z danymi ze zbioru referencyjnego. Zakłada się, że wskaźnikiem oceniającym będzie zawartość hemoglobiny – zmienna losowa *Hb*. Im wyższa wartość *Hb*, tym lepsza ocena skuteczności leku. Zadanie polega na stwierdzeniu istotnej statystycznie różnicy w ocenie leków – badanego i referencyjnego – dla zadanego poziomu istotności. W przypadku wystąpienia takiej różnicy, powinno nastąpić stworzenie wykresu pokazującego punktowe estymatory parametrów rozkładu względnego przyrostu hemoglobiny dla każdego z leków (średnie, maksima, minima, kwartyle – pierwszy, trzeci). W celu realizacji ww. zadania, przekazywany jest plik z danymi wejściowymi o hemoglobinie przed i po użyciu badanego leku. W wyniku wykonania zadania powinny zostać wygenerowane następujące pliki:

- plik z zawartością tzw. standardowego wyjścia, gdzie będzie pokazany przebieg realizacji zadania, w szczególności stwierdzenie wystąpienia lub braku statystycznej różnicy zastosowania leków (badanego i referencyjnego),
- plik z wykresem typu *boxplot* (tworzony w przypadku istotnej różnicy rozkładów stwierdzonej w pkt. a), pokazujący wspomniane, punktowe estymatory rozkładów.

Tabela 1

Dane o wartościach hemoglobiny przed i po podaniu badanego leku B

	"nr_pacjenta"	"Hb1"	"Hb2"	"HbRel"
"1"	101	12.5	13.4	0.072
"2"	102	7.1	8.4	0.183098592
"3"	103	8.7	9.91	0.13908046
"4"	104	7.95	8.61	0.083018868
"5"	105	12.6	13.5	0.071428571
"6"	106	10.45	10.9	0.043062201
"7"	107	15.5	16.9	0.090322581
"8"	108	12.6	13.9	0.103174603

Dane wejściowe dla zadania (tab. 1) umieszczone są w pliku danych wejściowych *set_Hb_B.dat*, opisujących odpowiednio: nr pacjenta, wartości *Hb*1 przed podaniem badanego *B*, wartość *Hb*2 po podaniu leku, wartość względnego przyrostu *HbRel* wg wzoru:
$HbRel = (Hb2 - Hb1) / Hb1$.

Zawartość tablicy *Pomiary* zawierającej zbiór wyników pomiarów wartości hemoglobiny (*Hb_pomiar*) przed podaniem leku (*nr_wizyty* = 1) i po podaniu leku (*nr_wizyty* = 2) dla referencyjnego leku A pokazuje tabela 2.

Tabela 2

Dane o wartościach hemoglobiny przed i po podaniu referencyjnego leku A

nr_pacjenta	nr_wizyty	Hb_pomiar
101	1	12,25
101	2	12,96
102	1	7,30
102	2	8,01
103	1	8,72
103	2	9,42
104	1	8,01
104	2	8,36
105	1	12,61
105	2	12,61
106	1	10,48
106	2	10,85
107	1	15,79
107	2	16,50
108	1	12,61
108	2	12,96

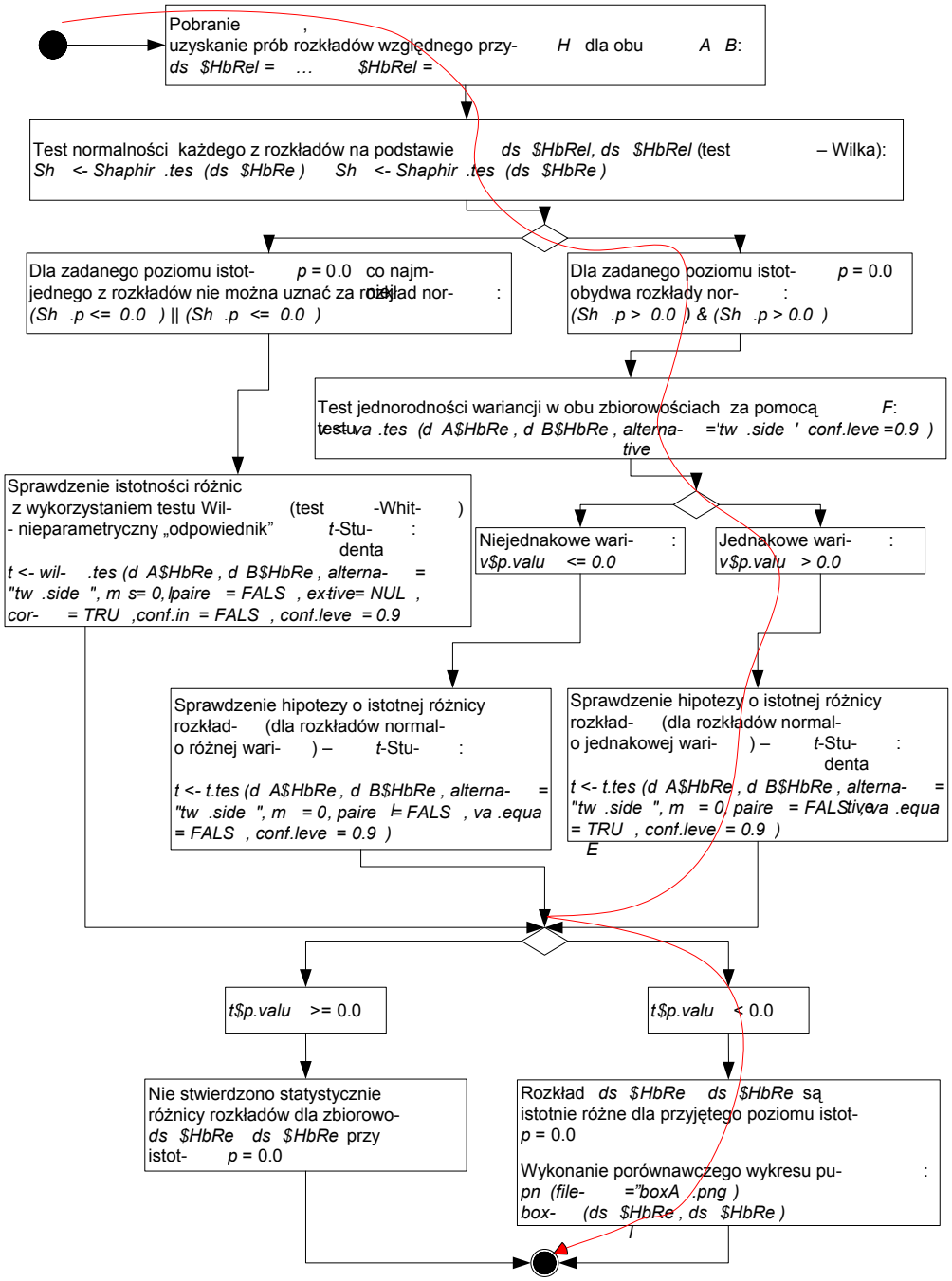
Na rys. 6 został przedstawiony algorytm zadania polegającego na weryfikacji hipotezy o statystycznie istotnej różnicy pomiędzy rozkładami względnej zmiany hemoglobiny przed i po podaniu leku dla leku badanego B i leku referencyjnego A. Hipoteza jest weryfikowana dla przyjętego poziomu istotności o wartości 0,05. W przypadku istotnej różnicy rozkładów stworzony jest w ramach tego zadania odpowiedni wykres typu *boxplot*. Kod zadania wyrażony jest za pomocą skryptu R.

Pobranie danych do eksperymentu (dane w tabelach 1 i 2), wykorzystanych w pierwszej czynności algorytmu z rys. 6, dla leku B pochodzą z przesłanego pliku, a dla leku A z relacyjnej bazy danych (użycie biblioteki RODBC [3]), co wyraża się następującym skryptem R:

```
# wczytanie zbioru badanego
dsB <- read.table ("set_Hb_B.dat")

# wczytanie zbioru referencyjnego (baza danych)
library(RODBC)
myConn <- odbcConnect("dbReferences")

strSQL = "select P1.nr_pacjenta, P1.Hb_pomiar As Hb1,P2.Hb_pomiar As Hb2,
(P2.Hb_pomiar - P1.Hb_pomiar)/P1.Hb_pomiar As HbRel
from
(select * from Pomiary where nr_wizyty =1) P1,
(select * from Pomiary where nr_wizyty =2) P2
where P1.nr_pacjenta = P2.nr_pacjenta
order by P1.nr_pacjenta"

dsA = sqlQuery(myConn, strSQL)
odbcClose (myConn)
```

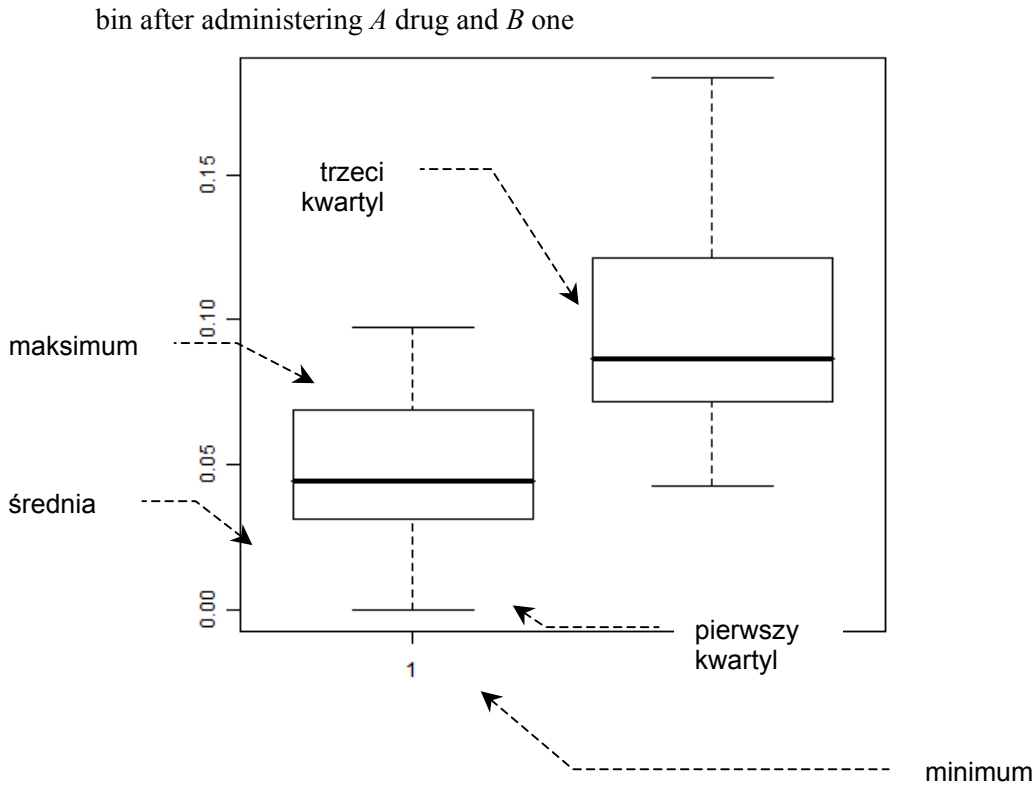
Dla danych przykładowych (tabela 1 i 2) realizacja zadania przebiega wg scenariusza opisanego pseudokodem:

1. Pobranie danych z zewnętrznego pliku z danymi – rozkład badany dla leku *B*
2. Pobranie danych z relacyjnej bazy danych – rozkład referencyjny dla leku *A*
3. Sprawdzenie hipotezy o normalności obu rozkładów *(shapiro.test)*
4. Jeśli przyjęto (na przyjętym poziomie istotności), że oba rozkłady normalne to
   4.1 Testowanie jednorodności wariancji *(var.test)*
   4.2 Jeśli przyjęto(na przyjętym poziomie istotności), że wariancje jednakowe
      4.2.1 Sprawdzenie statystycznie istotnej różnicy pomiędzy rozkładami *(t.test)*
      4.2.2 Jeśli rozkłady istotnie różne (na przyjętym poziomie istotności)
         4.2.2.1 Stworzenie wykresu pudełkowego *(box.plot)* ilustrującego różnicę rozkładów



Rys. 6. Schemat czynności zadania prowadzącego do stwierdzenia, że rozkłady opisujące względne przyrosty hemoglobiny po podaniu leku *A* i *B* są statystycznie różne

Fig. 6. Activity diagram of the task for verification of the hypothesis of significant statistical dereference between probabilistic distribution of relative change of hemoglo-



Rys. 7. Wynikowy wykres pudełkowy – porównanie rozkładów względnej zmiany hemoglobiny dla leku *A*(1) i *B*(2)
Fig. 7. Result boxplot diagram – the comparison of distributions of relative change of hemoglobin for drugs *A*(1) and *B*(2)

Przepływ sterowania dla danych przykładowych, opisany powyższym scenariuszem, został oznaczony czerwoną linią na schemacie algorytmu z rys. 6.

W wyniku realizacji zadania powstał plik tekstowy (*stdout.txt*) z zawartością standardowego wyjścia, opisujący przebieg pokazanego powyżej scenariusza, gdzie nastąpiło stwierdzenie istotnej różnicy rozkładów dla leku A i B. Dodatkowo powstał plik graficzny (*boxAB.png*) z wykresem pudełkowym, opisującym rozkłady względnych przyrostów hemoglobiny dla obu leków, pokazany na rys. 2 (1 - lek referencyjny A, 2 – lek badany B). Nawet pobieżna, nieformalna analiza danych wskazuje na istotnie różnice rozkładów, tzn. lek B jest lepszy od leku A – wszystkie parametry minimum, maksimum, średnia, pierwszy i trzeci kwartyl mają wartości wyższe.

4. Rozszerzenie systemu w zakresie bezpieczeństwa wynikające z wykorzystania mechanizmu izolowanych magazynów

Istotnymi składnikami każdego systemu informatycznego są jego mechanizmy bezpieczeństwa. Powinny one zapewniać dostęp do jego funkcji tylko wybranym użytkownikom

(posiadającym odpowiednie uprawnienia). Ważne jest także zapewnienie integralności i poufności przetwarzanych danych – użytkownik nie powinien widzieć (i móc zmienić) prywatnych danych innych użytkowników. W omawianym rozwiązaniu jest to szczególnie istotne. Wykonywane zadania obliczeniowe powinny być od siebie maksymalnie odseparowane, tak, aby nie dało się (przy użyciu specjalnie spreparowanego R-skryptu) odczytać danych/wyników z innego zadania. Ten aspekt bezpieczeństwa systemu zrealizowano w oparciu o mechanizm izolowanych magazynów (ang. Isolated Storage).

Wykorzystanie izolowanych magazynów pozwala zapisywać dane w specjalnym folderze, którego fizyczne położenie na dysku nie jest jawnie zapisane w kodzie źródłowym. Oprócz elastyczności (aplikacja nie wymaga, by istniał ściśle określony katalog na dysku), takie rozwiązanie zapewnia, że inne aplikacje (lub R-skrypty) będą miały utrudniony dostęp do prywatnych danych. Mechanizm izolowanych magazynów ma jeszcze jedną właściwość istotną z punktu widzenia omawianego systemu - administrator ma możliwość ustalenia maksymalnego rozmiaru zgromadzonych w magazynie plików. Zapobiega to wykonaniu złośliwych
R-skryptów, które zapełniają dysk niepotrzebnymi plikami.

5. Podsumowanie

Artykuł przedstawia architekturę oraz przykłady użycia systemu ServeR służącego do udostępniania usług obliczeń statystycznych zaimplementowanych w module R-project. Popularność R-project oraz języka skryptowego w docelowym środowisku użytkowników systemu, tj. wśród statystyków medycznych, zadecydowała o koncepcji udostępniania realizacji całych skryptów, zamiast „sztywnej” implementacji wybranego zestawu funkcji R-project.

Architektura omawianego systemu opiera się na wykorzystaniu trzech niezależnych składników – aplikacji klienta – serwera komunikacyjnego – serwera obliczeniowego. Ten ostatni zarządza pulą uruchomionych procesów R.exe, realizując pośrednio zlecone zadania obliczeń statystycznych. Wykorzystanie takiej wielowarstwowej architektury oraz technologii Windows Communication Foundation w zakresie komunikacji między składnikami systemu pozwala na uruchamianie systemu ServeR w różnych konfiguracjach, dostosowanych do wymogów infrastruktury sieciowej. Możliwe jest wykorzystanie różnych protokołów sieciowych (binarny/TCP w sieci lokalnej, potoki nazwane gdy procesy aplikacji klienta i serwera komunikacyjnego są uruchomione na tym samym komputerze) w celu podniesienia wydajności rozwiązania. Na potrzeby trybu klienta odłączonego, możliwe jest wykorzystanie systemu kolejkowego w komunikacji pomiędzy serwerem komunikacyjnym i obliczeniowym. W opisywanym trybie wsadowym aplikacja klienta zleca obliczenia, wysyłając R-

skrypty i ewentualnie pliki z danymi wejściowymi. W odpowiedzi uzyskuje plik standardowego wyjścia i ewentualnie pliki wynikowe (np. graficzne).

ServeR umożliwia również pracę w trybie bezpośrednim, czyli on-line, zwanym inaczej trybem zdalnej R-sesji. W tym trybie użytkownik może zlecać wykonanie pojedynczych R-komend, zadawanych ad hoc. Komendy te wykonywane są natychmiastowo – ich wynik dostarczany jest do aplikacji klienta w taki sposób, że użytkownik ma wrażenie pracy z lokalnym programem R-project. Tryb przeznaczony jest do uruchamiania i testowania komend wykorzystanych później do budowy skryptów użytych w trybie off-line.

Artykuł pokazuje praktyczne zastosowania systemu ServeR – w szczególności poprzez użycie prototypowych aplikacji klienta. Pierwszy z przykładów pokazuje wykorzystanie zdalnej R-konsoli, drugi użycie systemu w trybie off-line, czyli wykonanie zleconego R-skryptu. Artykuł przedstawia sposób wykorzystania w zleconym zadaniu obliczeniowym danych pochodzących ze zdalnej, relacyjnej bazy danych medycznego systemu dziedzinowego.

W opracowaniu omówiono rozszerzenie związane z użyciem mechanizmu izolowanych magazynów, mając na uwadze zwiększenie poziomu bezpieczeństwa systemu. Mechanizm ten użyty został przy implementacji obsługi procesu przetwarzania przez moduł R.exe, dzięki czemu uzyskano wzajemną izolację dyskowej przestrzeni roboczej procesów przetwarzających.

Z powodu swojej modularności ServeR może być wykorzystywany zarówno jako oddzielny komponent (zaimplementowano moduły aplikacji klienta w technologii .NET), jak i jako integralny element dowolnego, dziedzinowego systemu (dzięki użyciu WCF, usługi ServeR-a dostępne dla projektantów/programistów są łatwe w integracji).

BIBLIOGRAFIA

1. R Development Core Team R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna 2008.
2. The R Project for Statistical Computing Page: http://www.r-project.org.
3. Lapsey M., Riplay B. D.: The RODBC Package. ODBC Database Access Page,
4. http://cran.r-project.org/web/packages/RODBC/RODBC.pdf.
5. Augustyn D.R., Warchał Ł.: ServeR: .NET-based Infrastructure for Remote Services of Statistical Computing with R-Project, Communications in Computer and Information Science, Computer Networks , Springer-Verlag, Berlin Heidelberg 2009.
6. Peiris C., Mulder D., Cicoria S., Bahree, A., Pathak N.: Pro WCF: Practical Microsoft SOA Implementation. Springer-Verlag, New York 2007.

7. Microsoft Developer Network Page: http://msdn.microsoft.com.
8. Freeman A., Jones A.: Programming .NET Security. O'Reilly Media, Inc.
9. Boyd S., Costall R., Rabold K., Redkar A., Redkar T., Walzer C.: Pro MSMQ: Microsoft Message Queue Programming. Apress, USA 2004.
10. Jones A: C# Programmer's Cookbook, Microsoft Press USA 2003.

Recenzenci: Dr inż. Bożena Małysiak-Mrozek
Dr hab. Zygmunt Mazur, prof. Pol. Wrocławskiej

Wpłynęło do Redakcji 31 stycznia 2010 r.

Abstract

The paper presents ServeR – the system for enabling remote statistical calculation supported by R-project. Multilayer architecture (client application – communication server – calculation server) and Windows Communication Foundation technology usage allow to deploy the system in many configurations adapted to a network infrastructure. This makes system scalable e.g. by possibility of multiplying instances of calculation server hosted by independent network nodes. Using Microsoft Message Queuing between the communication server and the calculation server makes ServeR system usable when the calculation server is overloaded or even down.

Using ServeR in two modes – off-line and on-line (remote R-console) is shown in detailed examples of statistical tasks. The paper presents the example task of using referential data from remote domain medical system.

A new feature – an isolated storage mechanism used by statistical task processes – is considered. Using isolated storage enhances system security that different computation tasks run independently can`t access data of each other. A system administrator can explicitly set a disk quota used by a task of calculation process spawned by ServeR.

Adresy

Dariusz R. AUGUSTYN: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, Polska, draugustyn@polsl.pl.
Łukasz WARCHAŁ: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-101 Gliwice, Polska, lukasz.warchal@polsl.pl.