



AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Replikacja danych

Opracowanie: Sylwia Miedzińska

Zespół w składzie
Sylwia Miedzińska
Piotr Michno
Michał Mielec
Konrad Łata
Paweł Kozub

Wydział Inżynierii Metali i Informatyki Przemysłowej

Kraków, 09.06.2016

Zagadnienie replikacji [3][4]

- Replikacja danych polega na przechowywaniu kopii danych na wielu komputerach.

Podstawowy model zarządzania replikowanymi danymi:

- - Klienci wysyła żądanie do systemu(front-end).
- - System(front-end) zapewnia przejrzystość, ukrywając fakt, że dane są replikowane.
- - System(front-end) komunikuje się z jednym lub więcej menedżerów repliki do pobierania / zapisywania danych.
- - Zarządcy repliką współdziałają w celu zapewnienia, że dane są spójne.

Powody zwielokrotniania [1]

- Skalowalność – możliwe jest rozłożenie obciążenia pomiędzy wieloma serwerami. Operacje zapisu i aktualizacji rekordów odbywają się na jednym serwerze, a pobieranie i przeszukiwanie danych z drugiego.
- Bezpieczeństwo – Zwielokrotnianie w celach efektywnościowych jest ważne w sytuacjach, w których system rozproszony należy skalować w wymiarze liczbowym i geograficznym. Skalowanie w wymiarze liczbowym występuje na przykład wówczas, gdy wzrasta liczba procesów wymagających dostępu do danych zarządzanych przez jeden serwer. W tym wypadku efektywność możemy polepszyć przez zwielokrotnienie serwera i podział pracy

Podział wybranych algorytmów na pasywne i aktywne

Algorytmy pasywne	Algorytmy aktywne
Kopia podstawowa – pisanie/czytanie tylko z podstawowej	Brak kopii podstawowej - z globalnym znacznikiem czasu Lamporta
Kopia podstawowa – zapasowa – czytanie z kopii zapasowej	Brak kopii podstawowej – z procesem porządkowym (koordynatorem nadającym unikalne id operacjom)
	Brak kopii podstawowej – z głosowaniem kworum

Tabela 1. Podział wybranych algorytmów na pasywne i aktywne [10]

Nazwa algorytmu	Nazwisko Autora	Data powstania
Kopia podstawowa – pisanie/czytanie tylko z podstawowej	Budhijara	1993
Kopia podstawowa – zapasowa – czytanie z kopii zapasowej	Li, Hudak	1989
Brak kopii podstawowej - z globalnym znacznikiem czasu Lamporta	Rodrigues	1996
Brak kopii podstawowej – z procesem porządkowym (koordynatorem nadającym unikalne id operacjom)	Fonseca H	1996
Brak kopii podstawowej – z głosowaniem kworum	Thomas, Gifford	1979

Tabela 2. Zestawienie twórców algorytmów i roku powstania.[1]

OPIS ALGORYTMÓW

Kopia podstawowa – pisanie/czytanie tylko z podstawowej[4]

Nazwa algorytmu	Opis algorytmu	Wady i problemy związane z użyciem algorytmu
Kopia podstawowa – pisanie/czytanie tylko z podstawowej	<p>Operacje pisania i czytania oparte są na jednym zdalnym serwerze</p> <p>Rezultat : dane nie są zwielokrotniane lecz umieszczone na jednym zdalnym serwerze.</p> <p>Proces, które chce wykonać operację zapisania jednostki danych x, przekazuje tę operację do serwera głównego x. Serwer ten wykonuje uaktualnienie na lokalnej kopii x, po czym przekazuje uaktualnienie do serwerów zapasowych. Każdy serwer zapasowy dokonuje również aktualizacji i wysyła potwierdzenie z powrotem do serwera podstawowego. Gdy wszystkie serwery zapasowe uaktualnią swoje kopie lokalne, wówczas serwer podstawowy wysyła potwierdzenie do procesu, który zapoczątkował te działania</p>	<p>- Potencjalnym problemem efektywności w tym schemacie może być dość długi czas, który mija, zanim procesowi inicjującemu aktualizację zezwoli się na dalszą pracę. Wskutek tego wszystkie aktualizacje realizujemy jako operacje blokowane. Możemy też zastosować metodę bez blokowania. Gdy tylko serwer główny uaktualni swoją lokalną kopię x, zwraca potwierdzenie. Dopiero potem powiadamia serwery zapasowe, aby też wykonały uaktualnienia.</p> <p>- Główny problem w nieblokowanych protokołach podstawa-zapas dotyczy tolerowania awarii. W schemacie z blokowaniem procesu klient wie na pewno, że aktualizacja została wykonana na kilku innych serwerach zapasowych. Pewności tej nie ma w rozwiązaniu bez blokowania</p>

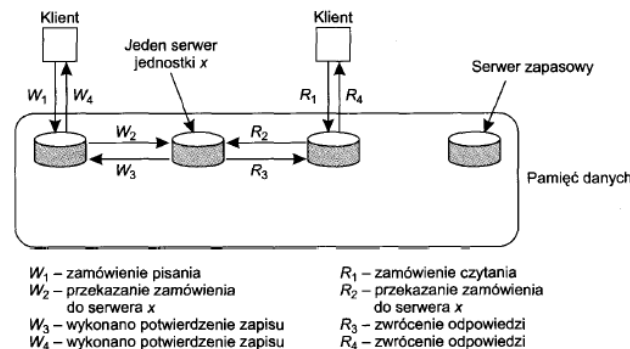


Tabela 3a Opis algorytmów i problemy związane z ich użyciem

Kopia podstawowa – zapasowa – czytanie z kopii zapasowej [5]

**Kopia podstawowa – zapasowa –
czytanie z kopii zapasowej**

Istnieją dwa rodzaje protokołów pisania lokalnego, opartego na kopii podstawowej. W pierwszym rodzaju każda jednostka danych *x* ma tylko jedną kopię. Mówiąc inaczej - nie ma zwielokrotnieli. Ilekroć proces chce wykonać operację na jednostce danych, tylekroć jest do niego najpierw przesyłana ta jedyna kopia, po czym jest wykonywana operacja. Ten protokół tworzy w istocie w pełni rozproszoną, niezwiłokrotnioną wersję pamięci danych. Spójność jest oczywista, gdyż zawsze istnieje tylko jedna kopia każdej jednostki danych.

- Jedną z głównych trudności w tej metodzie pełnej wędrówki jest śledzenie aktualnego miejsca pobytu każdej jednostki danych

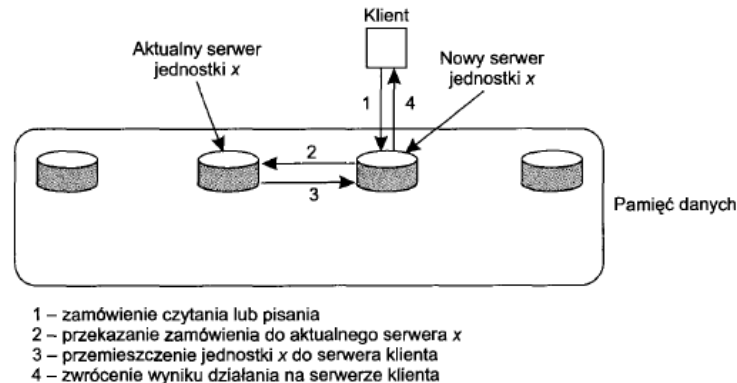


Tabela 3b Opis algorytmów i problemy związane z ich użyciem

Brak kopii podstawowej - z globalnym znacznikiem czasu Lamporta[6]

<p>Brak kopii podstawowej - z globalnym znacznikiem czasu Lamporta</p>	<p>W protokołach zwielokrotnionych zapisów (ang. replicated-writes) operacje pisania możemy wykonywać na wielu kopiach, a nie tylko na jednej, jak w wypadku kopii podstawowych. Potencjalnym problemem aktywnego zwielokrotnienia jest konieczność wykonywania operacji wszędzie w tym samym porządku. Jest więc potrzebny mechanizm całkowicie uporządkowanego rozsyłania. Rozsyłanie takie możemy zrealizować przy użyciu znaczników czasu Lamporta.</p>	<ul style="list-style-type: none"> - konieczność wykonywania operacji wszędzie w tym samym porządku - znaczniki czasu Lamporta źle się skalują w wielkich systemach rozproszonych
--	---	---

Tabela 3c Opis algorytmów i problemy związane z ich użyciem

Brak kopii podstawowej – z procesem porządkowym (koordynatorem nadającym unikalne id operacjom)[1]

Brak kopii podstawowej – z procesem porządkowym (koordynatorem nadającym unikalne id operacjom)	Każdą operacje przekazuje się najpierw porządkowemu Porządkowy przypisuje niepowtarzalny numer Porządkowy przekazuje tą operacje do wszystkich kopii. Uwaga! Operacje wykonywane są w kolejności numerów porządkowych.	- problem zwielokrotnionych wywołań
--	---	--

Tabela 3d Opis algorytmów i problemy związane z ich użyciem

Brak kopii podstawowej – z głosowaniem kworum[1]

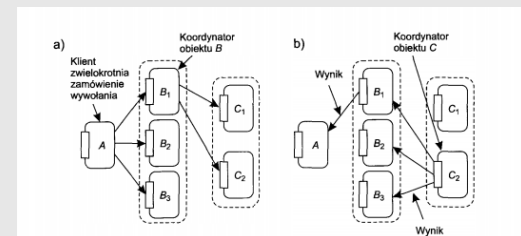
Brak kopii podstawowej – z głosowaniem kworum

Rozważamy rozproszony system plików, zakładamy że plik jest zwielokrotniany na N serwerach W celu zaktualizowania pliku klient musi się skontaktować przynajmniej z połową + 1 serwerów.
Po zgodzie serwerów plik zostaje podmieniony a nowa wersja zostaje zaopatrzona nowym numerem – numer wersji służy do identyfikowania wersji pliku i jest taki sam dla wszystkich nowo zaktualizowanych kopii pliku. Aby przeczytać zwielokrotniony plik klient również musi się skontaktować z ponad połową serwerów i prosić je o wysłanie numerów wersji.

- Uproszczony schemat Gifforda: Do czytania pliku mającego N zwielokrotnień wymaga się od klienta kworum czytania, Czyli dowolnego zbioru N_r lub więcej serwerów, podobnie do zmodyfikowania pliku potrzeba kworum pisania, co najmniej N_w serwerów, Wartości te muszą spełniać następujące ograniczenia

$$N_r + N_w > N$$

$$N_w > N/2$$



Na rysunku a może wystąpić konflikt pisanie – pisanie ponieważ nie jest spełniony warunek 1. W szczególności gdy jeden klient odbierze swój zbiór do zapisu $\{A, B, C, E, F, G\}$, a drugi wybierze $\{D, H, I, J, K, L\}$

Na rysunku b jest szczególnie interesująca sytuacja ponieważ N_r wynosi tu 1 co oznacza możliwość czytania zwielokrotnionego pliku za pomocą dowolnej liczby kopii – co oznacza aktualizowanie wszystkich kopii przy zapisywaniu – nazwa schematu – czytaj jedno zapisuj wszystko.

Tabela 3e Opis algorytmów i problemy związane z ich użyciem.

Zakres stosowalności

Algorytmy, w których każda kopia umożliwia czytanie są wykorzystywane w bardzo rozproszonych serwerach, np. DNS lub CDN.

Aktywa replikacja danych znajduje zastosowanie w rozproszonych systemach plików.(OpenAFS, Google Cloud Starage)

Protokół Gossip, system BAYOU, *Coda* (Constant Data Availability)[2]

Wybór algorytmu

- Wybrany algorytm jest najprostszy i najłatwiejszy w implementacji
- Wybrany algorytm nie zawiera w sobie dodatkowych zagadnień z zakresu systemów rozproszonych.
- Wybrany algorytm pozwala w najbardziej przejrzysty i czytelny sposób przedstawić zasadę działania algorytmów replikacji.

Wybór technologii

SOAP[11]	RMI[12]	Sockety[13]	REST[14]	MPI[18]	CORBA[15]
<ul style="list-style-type: none"> Simple Object Access Protocol, protokół komunikacyjny oparty o XML. Niezwykłą elastyczność protokołu, który pozwala przenosić właściwie dowolne informacje Duży narzut samego języka XML (rozmiar komunikatu jest znacząco większy niż sumaryczny rozmiar danych w nim zawartych) 	<p>Remote Method Invocation, umożliwia programowanie rozproszone w Javie. Mechanizm zdalnych wywołań umożliwia wywołanie metod z obiektów pod kontrolą innych maszyn wirtualnych języka Java. Mogą działać na różnych komputerach.</p>	<p>Narzędzie do komunikacji pomiędzy procesem działającym na tej samej maszynie bądź na innym. Do stworzenia socketów potrzebny jest protokół, domena oraz typ komunikacji.</p>	<p>REpresentational State Transfer, zamiast XML używa prostego URL. Większość zadań można uzyskać poprzez żądania HTTP 1.1 takie jak GET, POST, PUT, DELETE. Dane można przesyłać przez JSON, RSS.</p>	<p>Message Passing Interface, protokół przesyłania komunikatów pomiędzy procesami programów równoległych. Komunikacja może być grupowa bądź punktowa. MPI_Init – inicjalizacja MPI, MPI_Send – wysyłanie blokujące, MPI_Recv – odbiór blokujący MPI_Finalize – zakończenie działania</p>	<p>Common Object Request Broker Architecture, przeznaczona przede wszystkim do wspomagania programowania pomiędzy systemami niekompatybilnymi. Określa metody dostępu do obiektów i komunikacji między obiektami</p>

Tabela 4 Zestawienie wybranych technologii

Diagramy UML

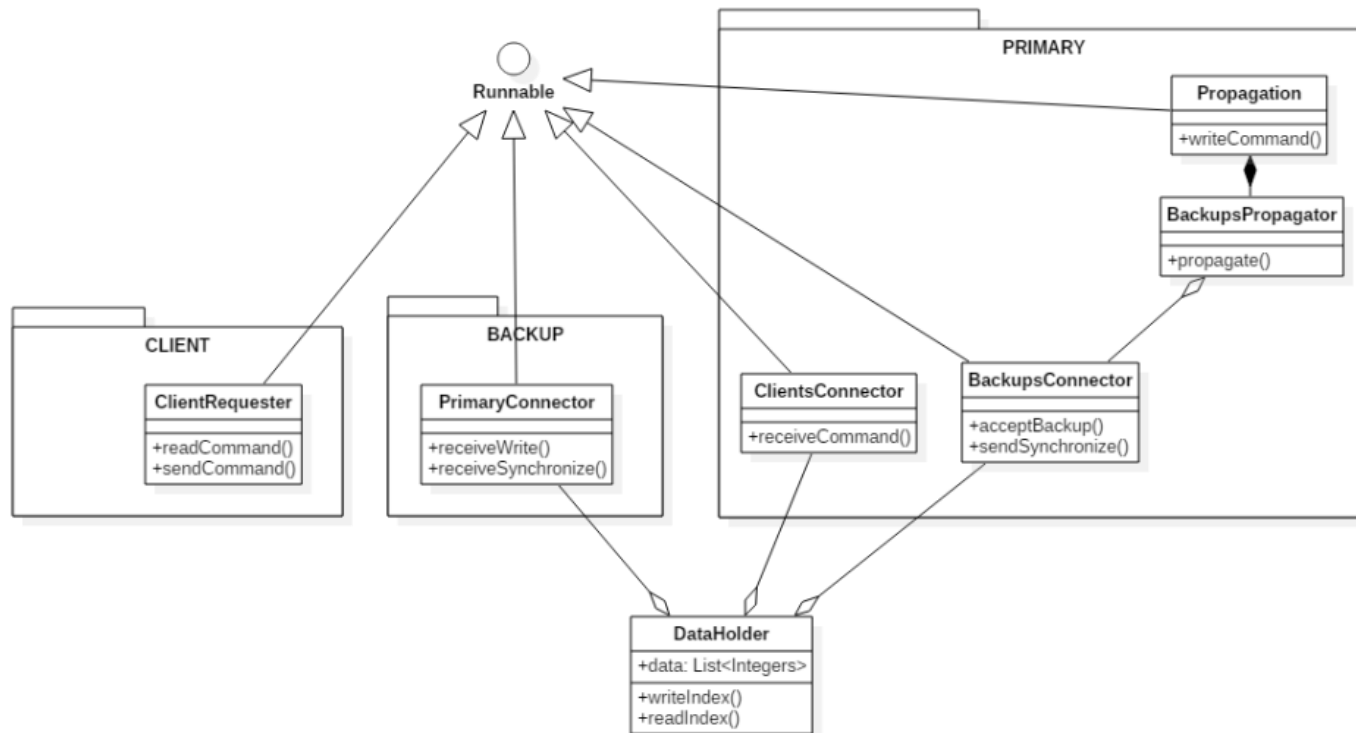


Diagram1. Diagram UML Kopia podstawowa – pisanie zdalne (technologia sockety)

```
C:\Windows\System32\cmd.exe - java -jar replication-sockets-1.0-SNAPSHOT.jar client
D:\Projects\replication-sockets\target>java -jar replication-sockets-1.0-SNAPSHOT.jar client
write 3 9
Podaj komendę: (exit kończy)
write 0 7
Podaj komendę: (exit kończy)
write 5 5
Podaj komendę: (exit kończy)
write 5 8
Podaj komendę: (exit kończy)
write 0 1
Podaj komendę: (exit kończy)
write 1 8
Podaj komendę: (exit kończy)

C:\Windows\System32\cmd.exe - java -jar replication-sockets-1.0-SNAPSHOT.jar primary
D:\Projects\replication-sockets\target>java -jar replication-sockets-1.0-SNAPSHOT.jar primary
BackupsConnector::run Started!
ClientsConnector::run Started!
ClientsConnector::run Client accepted on port: 51885
ClientsConnector::start Reading from 51885: write 3 9
[0, 0, 0, 9, 0, 0, 0, 0, 0, 0]
ClientsConnector::run Client accepted on port: 51887
ClientsConnector::start Reading from 51887: write 0 7
[7, 0, 0, 9, 0, 0, 0, 0, 0, 0]
ClientsConnector::run Client accepted on port: 51888
BackupsConnector::run Backup accepted on port: 51888
BackupsConnector::run Backup accepted on port: 51890
ClientsConnector::start Reading from 51888: write 5 5
[7, 0, 0, 9, 0, 5, 0, 0, 0, 0]
ClientsConnector::run Client accepted on port: 51891
ClientsConnector::start Reading from 51891: write 5 8
[7, 0, 0, 9, 0, 5, 0, 0, 0, 0]
ClientsConnector::run Client accepted on port: 51892
ClientsConnector::start Reading from 51892: write 0 1
[1, 0, 0, 9, 0, 0, 0, 0, 0, 0]
ClientsConnector::run Client accepted on port: 51893
ClientsConnector::start Reading from 51893: write 1 8
[1, 0, 0, 9, 0, 0, 0, 0, 0, 0]
ClientsConnector::run Client accepted on port: 51894

C:\Windows\System32\cmd.exe - java -jar replication-sockets-1.0-SNAPSHOT.jar backup
D:\Projects\replication-sockets\target>java -jar replication-sockets-1.0-SNAPSHOT.jar backup
PrimaryConnector::run Started!
PrimaryConnector::start Reading from 10000: synchronize 7 0 0 0 0 0 0 0
[7, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[7, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[7, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[7, 0, 0, 9, 0, 0, 0, 0, 0, 0]
[7, 0, 0, 9, 0, 0, 0, 0, 0, 0]
[7, 0, 0, 9, 0, 0, 0, 0, 0, 0]
[7, 0, 0, 9, 0, 0, 0, 0, 0, 0]
[7, 0, 0, 9, 0, 0, 0, 0, 0, 0]
[7, 0, 0, 9, 0, 0, 0, 0, 0, 0]
[7, 0, 0, 9, 0, 0, 0, 0, 0, 0]
[7, 0, 0, 9, 0, 0, 0, 0, 0, 0]
[7, 0, 0, 9, 0, 0, 0, 0, 0, 0]
PrimaryConnector::start Reading from 10000: write 5 5
[7, 0, 0, 9, 0, 5, 0, 0, 0, 0]
PrimaryConnector::start Reading from 10000: write 5 8
[7, 0, 0, 9, 0, 8, 0, 0, 0, 0]
PrimaryConnector::start Reading from 10000: write 0 1
[1, 0, 0, 9, 0, 0, 0, 0, 0, 0]
PrimaryConnector::start Reading from 10000: write 1 8
[1, 0, 0, 9, 0, 0, 0, 0, 0, 0]

C:\Windows\System32\cmd.exe - java -jar replication-sockets-1.0-SNAPSHOT.jar backup
D:\Projects\replication-sockets\target>java -jar replication-sockets-1.0-SNAPSHOT.jar backup
PrimaryConnector::run Started!
PrimaryConnector::start Reading from 10000: synchronize 7 0 0 9 0 0 0 0
[7, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[7, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[7, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[7, 0, 0, 9, 0, 0, 0, 0, 0, 0]
[7, 0, 0, 9, 0, 0, 0, 0, 0, 0]
[7, 0, 0, 9, 0, 0, 0, 0, 0, 0]
[7, 0, 0, 9, 0, 0, 0, 0, 0, 0]
[7, 0, 0, 9, 0, 0, 0, 0, 0, 0]
[7, 0, 0, 9, 0, 0, 0, 0, 0, 0]
[7, 0, 0, 9, 0, 0, 0, 0, 0, 0]
[7, 0, 0, 9, 0, 0, 0, 0, 0, 0]
[7, 0, 0, 9, 0, 0, 0, 0, 0, 0]
PrimaryConnector::start Reading from 10000: write 5 5
[7, 0, 0, 9, 0, 5, 0, 0, 0, 0]
PrimaryConnector::start Reading from 10000: write 5 8
[7, 0, 0, 9, 0, 8, 0, 0, 0, 0]
PrimaryConnector::start Reading from 10000: write 0 1
[1, 0, 0, 9, 0, 0, 0, 0, 0, 0]
PrimaryConnector::start Reading from 10000: write 1 8
[1, 0, 0, 9, 0, 0, 0, 0, 0, 0]
```

Screen z działania w technologii Socket

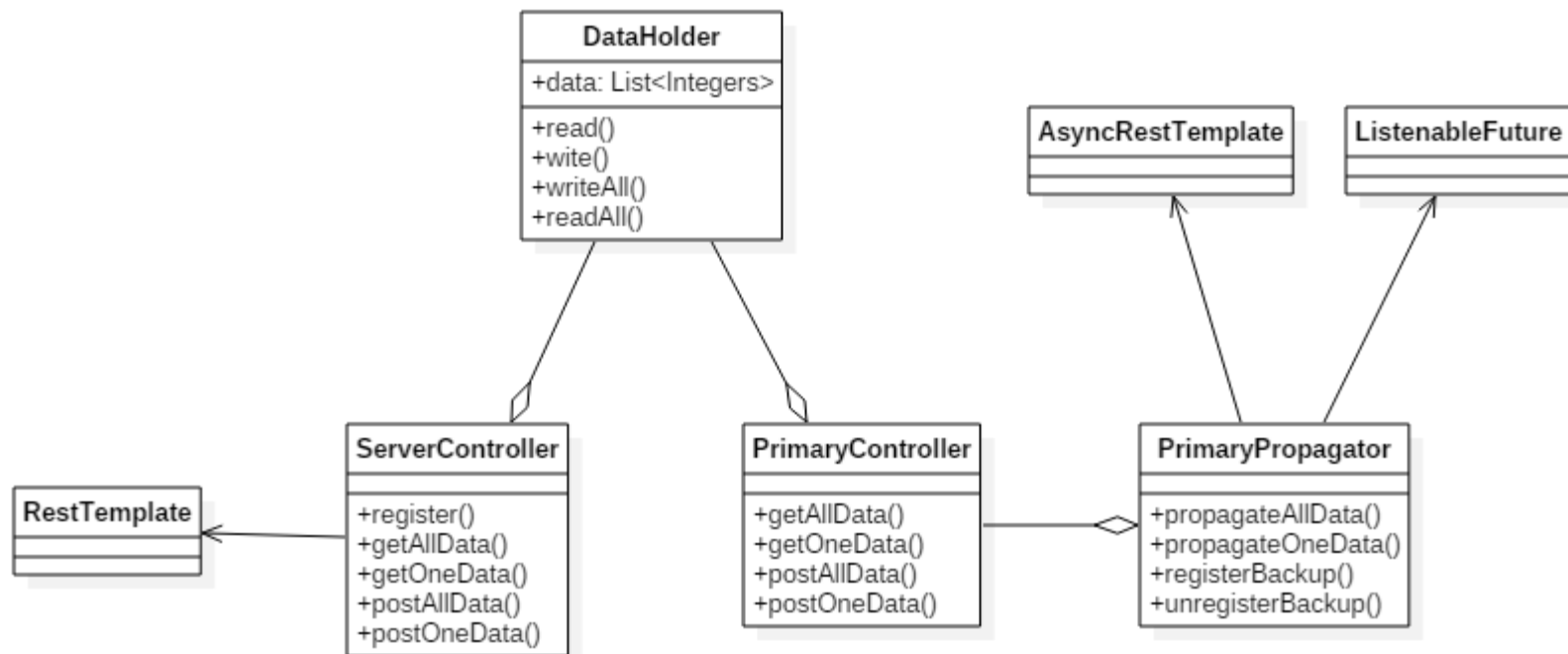
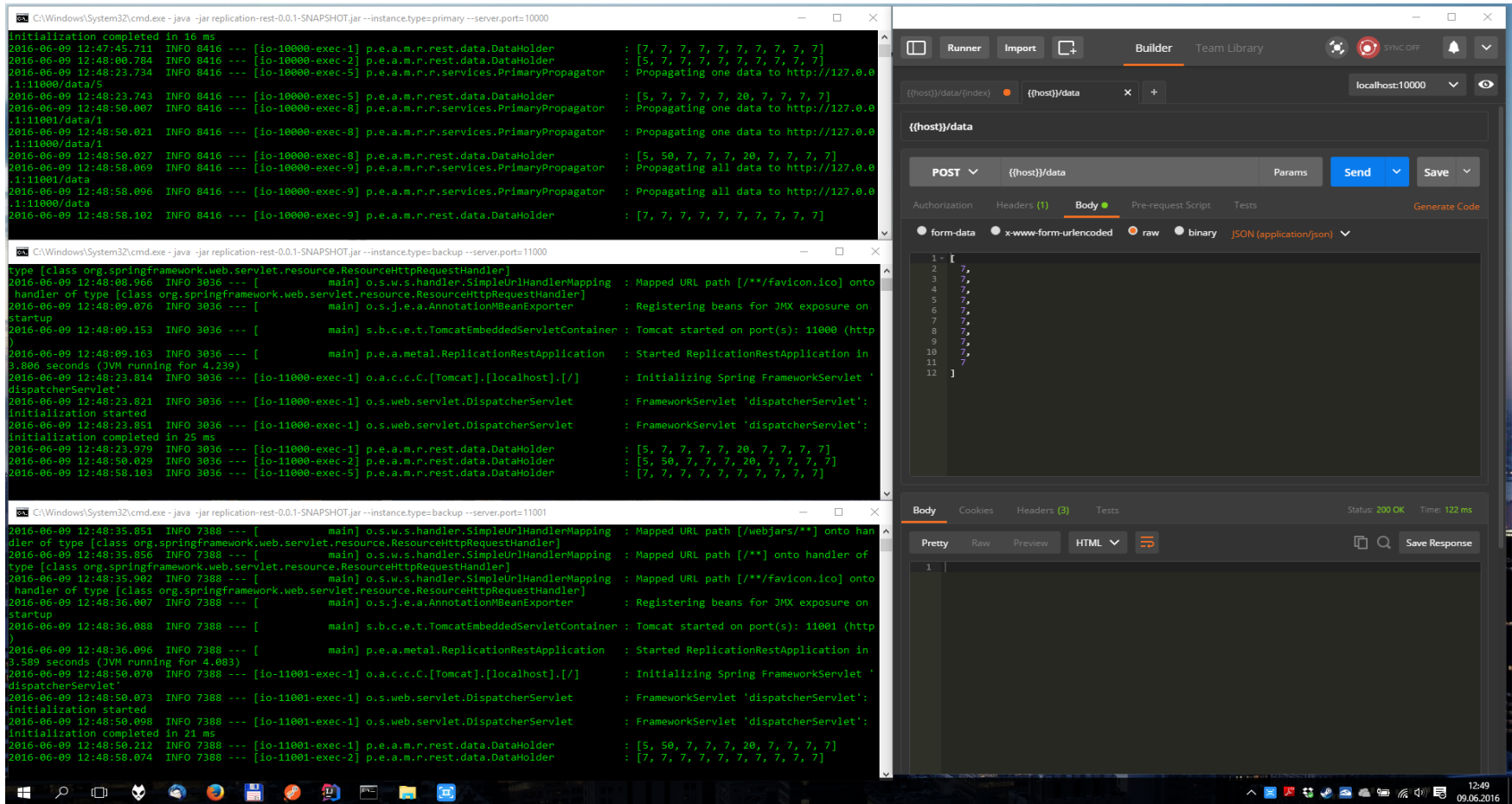


Diagram2. Diagram UML Kopia podstawowa – pisanie zdalne (technologia REST)

Działanie algorytmu w technologii rest



The screenshot displays the execution of a REST API client application in a Java IDE. The terminal window shows the application's startup logs, including the initialization of the Spring Framework and the starting of the Tomcat server on port 11000. The REST client window shows a POST request to the endpoint `{(host)}/data` with a body of `[7, 7, 7, 7, 7, 7, 7, 7, 7]`. The response status is 200 OK, and the response body is `[5, 7, 7, 7, 7, 7, 7, 7, 7]`.

```
2016-06-09 12:47:45.711 INFO 8416 --- [io-10000-exec-1] p.e.a.m.r.rest.data.DataHolder : [7, 7, 7, 7, 7, 7, 7, 7, 7]
2016-06-09 12:48:00.784 INFO 8416 --- [io-10000-exec-2] p.e.a.m.r.rest.data.DataHolder : [5, 7, 7, 7, 7, 7, 7, 7, 7]
2016-06-09 12:48:23.734 INFO 8416 --- [io-10000-exec-5] p.e.a.m.r.r.services.PrimaryPropagator : Propagating one data to http://127.0.0.1:11000/data/5
2016-06-09 12:48:23.743 INFO 8416 --- [io-10000-exec-5] p.e.a.m.r.rest.data.DataHolder : [5, 7, 7, 7, 7, 20, 7, 7, 7]
2016-06-09 12:48:50.007 INFO 8416 --- [io-10000-exec-8] p.e.a.m.r.r.services.PrimaryPropagator : Propagating one data to http://127.0.0.1:11000/data/1
2016-06-09 12:48:50.021 INFO 8416 --- [io-10000-exec-8] p.e.a.m.r.r.services.PrimaryPropagator : Propagating one data to http://127.0.0.1:11000/data/1
2016-06-09 12:48:50.027 INFO 8416 --- [io-10000-exec-8] p.e.a.m.r.rest.data.DataHolder : [5, 50, 7, 7, 7, 20, 7, 7, 7]
2016-06-09 12:48:50.069 INFO 8416 --- [io-10000-exec-9] p.e.a.m.r.r.services.PrimaryPropagator : Propagating all data to http://127.0.0.1:11000/data
2016-06-09 12:48:58.096 INFO 8416 --- [io-10000-exec-9] p.e.a.m.r.r.services.PrimaryPropagator : Propagating all data to http://127.0.0.1:11000/data
2016-06-09 12:48:58.102 INFO 8416 --- [io-10000-exec-9] p.e.a.m.r.rest.data.DataHolder : [7, 7, 7, 7, 7, 7, 7, 7, 7]
```

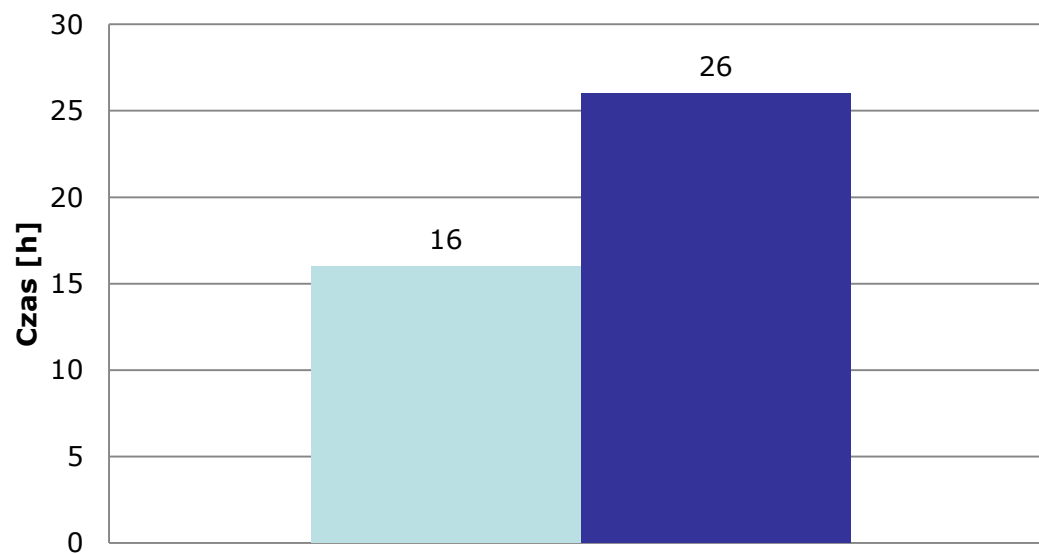
```
2016-06-09 12:48:08.966 INFO 3036 --- [main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/**/favicon.ico] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2016-06-09 12:48:09.076 INFO 3036 --- [main] o.s.j.e.a.AnnotationMBeanExporter : Registering beans for JMX exposure on startup
2016-06-09 12:48:09.153 INFO 3036 --- [main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 11000 (http)
2016-06-09 12:48:09.163 INFO 3036 --- [main] p.e.a.metal.ReplicationRestApplication : Started ReplicationRestApplication in 3.806 seconds (JVM running for 4.239)
2016-06-09 12:48:23.814 INFO 3036 --- [io-11000-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring FrameworkServlet 'dispatcherServlet'
2016-06-09 12:48:23.821 INFO 3036 --- [io-11000-exec-1] o.s.web.servlet.DispatcherServlet : FrameworkServlet 'dispatcherServlet':
2016-06-09 12:48:23.851 INFO 3036 --- [io-11000-exec-1] o.s.web.servlet.DispatcherServlet : FrameworkServlet 'dispatcherServlet':
2016-06-09 12:48:23.979 INFO 3036 --- [io-11000-exec-1] p.e.a.m.r.rest.data.DataHolder : [5, 7, 7, 7, 7, 20, 7, 7, 7]
2016-06-09 12:48:50.029 INFO 3036 --- [io-11000-exec-2] p.e.a.m.r.rest.data.DataHolder : [5, 50, 7, 7, 7, 20, 7, 7, 7]
2016-06-09 12:48:58.103 INFO 3036 --- [io-11000-exec-5] p.e.a.m.r.rest.data.DataHolder : [7, 7, 7, 7, 7, 7, 7, 7, 7]
```

```
2016-06-09 12:48:35.851 INFO 7388 --- [main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/**/favicon.ico] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2016-06-09 12:48:35.856 INFO 7388 --- [main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/**] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2016-06-09 12:48:35.902 INFO 7388 --- [main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/**/favicon.ico] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2016-06-09 12:48:36.007 INFO 7388 --- [main] o.s.j.e.a.AnnotationMBeanExporter : Registering beans for JMX exposure on startup
2016-06-09 12:48:36.088 INFO 7388 --- [main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 11001 (http)
2016-06-09 12:48:36.096 INFO 7388 --- [main] p.e.a.metal.ReplicationRestApplication : Started ReplicationRestApplication in 3.589 seconds (JVM running for 4.083)
2016-06-09 12:48:50.070 INFO 7388 --- [io-11001-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring FrameworkServlet 'dispatcherServlet'
2016-06-09 12:48:50.073 INFO 7388 --- [io-11001-exec-1] o.s.web.servlet.DispatcherServlet : FrameworkServlet 'dispatcherServlet':
2016-06-09 12:48:50.098 INFO 7388 --- [io-11001-exec-1] o.s.web.servlet.DispatcherServlet : FrameworkServlet 'dispatcherServlet':
2016-06-09 12:48:50.212 INFO 7388 --- [io-11001-exec-1] p.e.a.m.r.rest.data.DataHolder : [5, 50, 7, 7, 7, 20, 7, 7, 7]
2016-06-09 12:48:58.074 INFO 7388 --- [io-11001-exec-2] p.e.a.m.r.rest.data.DataHolder : [7, 7, 7, 7, 7, 7, 7, 7, 7]
```

Screen z działania algorytmu w technologii REST

Analiza porównawcza implementacji

Czasowy nakład pracy



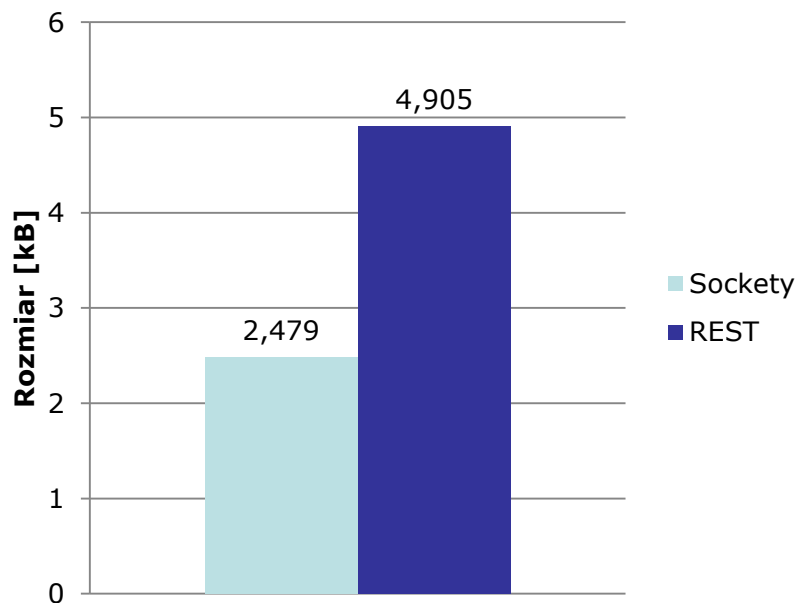
Nakład pracy	Czas [h]
Sockety	16
REST	26

Tabela 5 Nakład czasu pracy w poszczególnych technologiach

Wykres 1 Nakład czasu pracy w poszczególnych technologiach

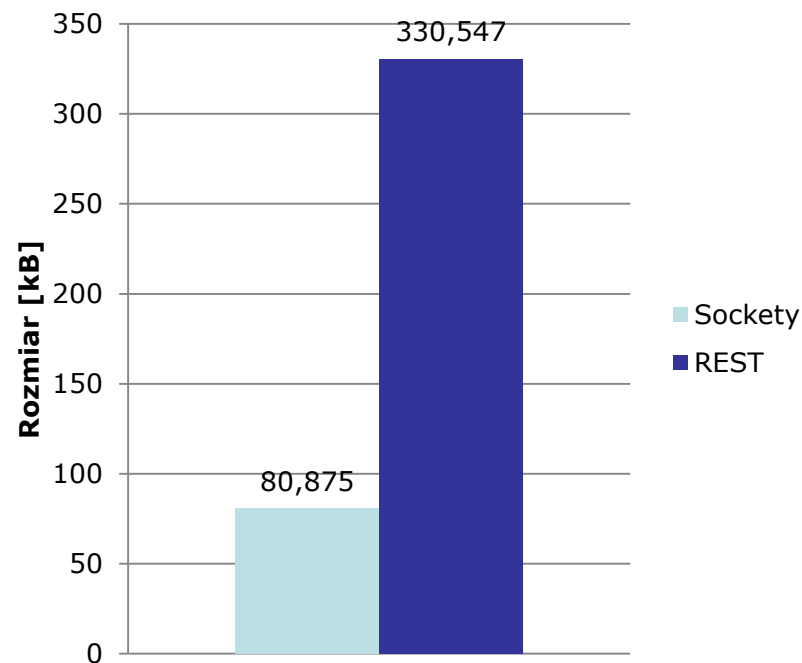
porównania szybkości działania tego samego algorytmu w różnych technologiach

Wielkość wiadomości dla
przykładowego łańcucha
znakowego



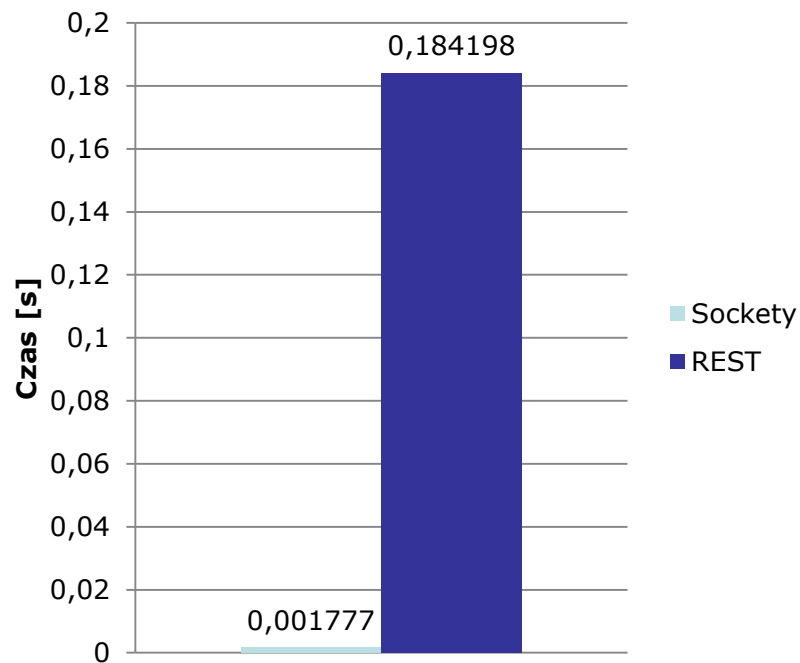
Wykres 2 Wielkość wiadomości dla
przykładowego łańcucha znakowego

Wielkości wiadomości dla
przykładowego ciągu liczb



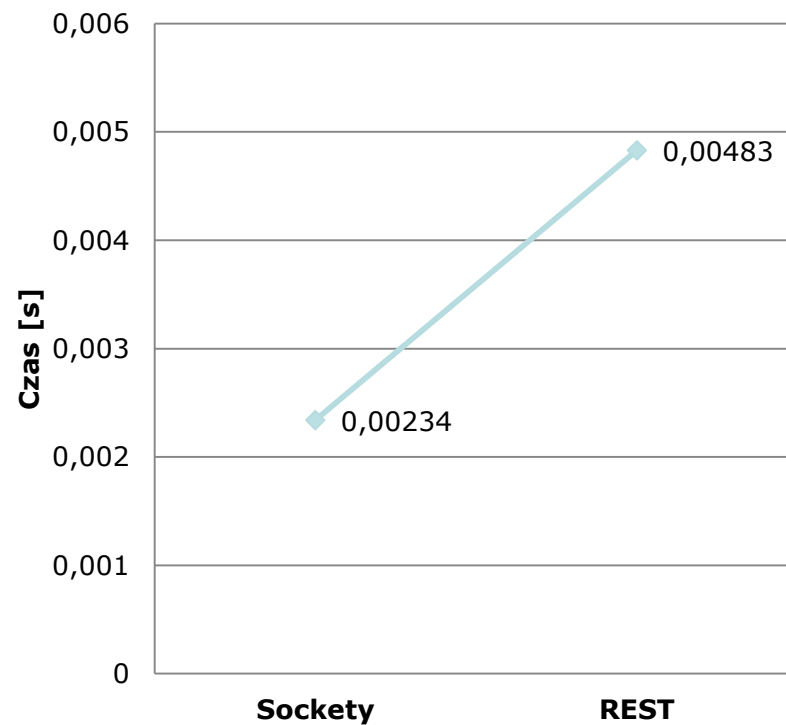
Wykres 3 Wielkości wiadomości dla
przykładowego ciągu liczb

Czas przesyłania dużego łańcucha znaków



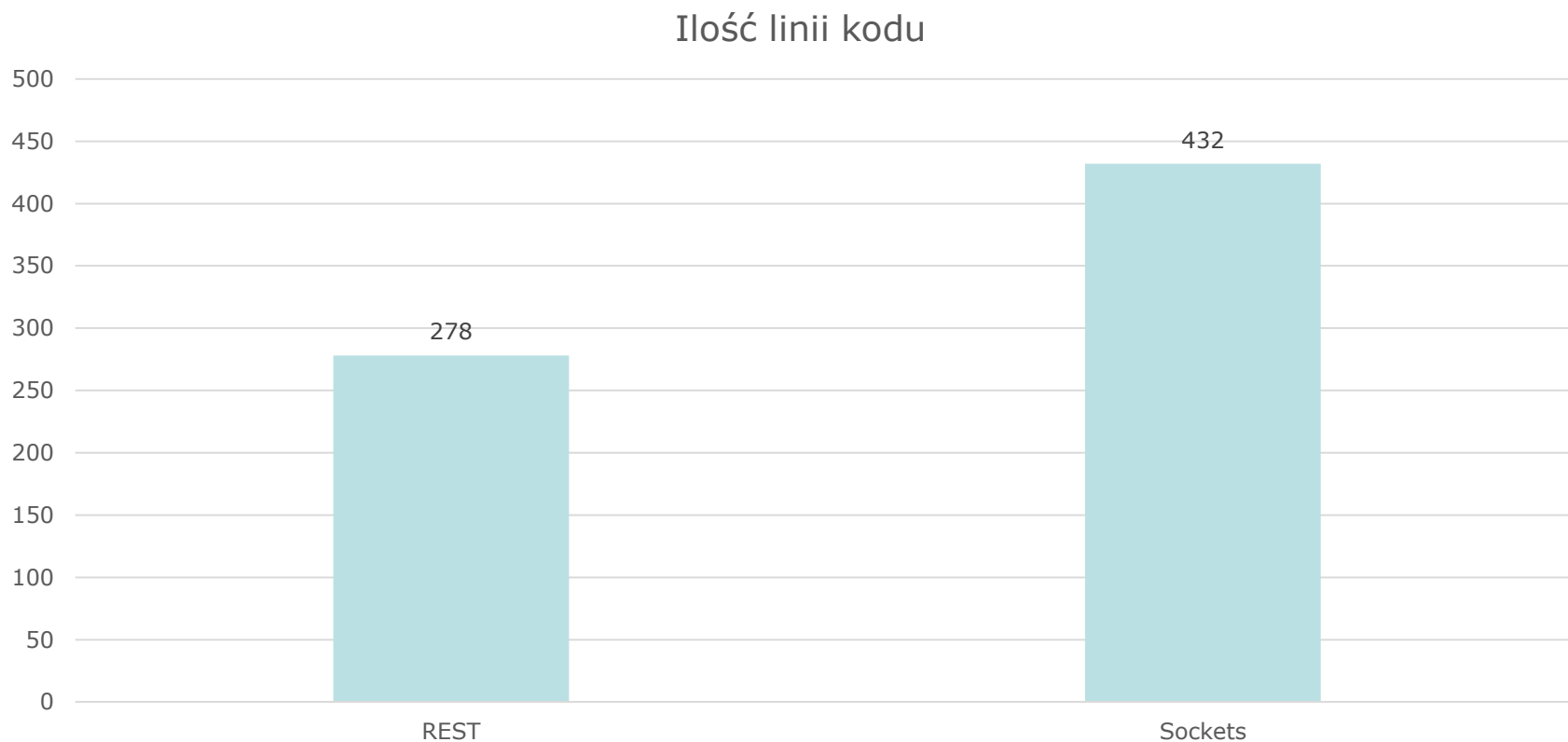
Wykres 5 Czas przesyłania dużego łańcucha znaków

Czas połączenia



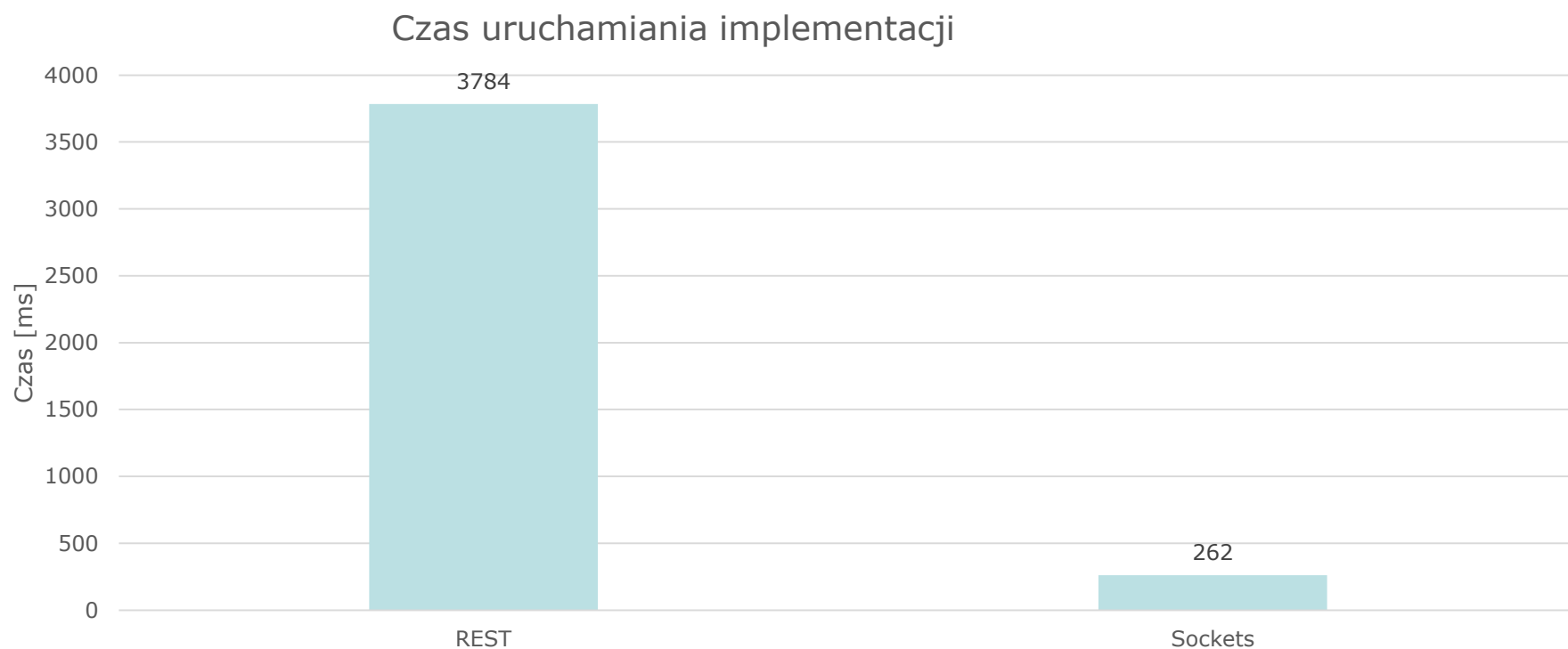
Wykres 4 Czas połączenia

Wykres ilości kodu



Wykres 6 Ilości kodu w poszczególnych technologiach.

Czas uruchamiania implementacji



Wykres 7 Czas uruchamiania implementacji.

1. Andrew S. Tanenbaum, Maarten Van Steen, „Distributed Systems – Principles and Paradigms” Second Edition, 2007, ISBN 0-13-239227-5
2. Sukumar Ghosh, „Distributed Systems – An Algorithmic Approach”, 2007, ISBN 1-58488-564-5
3. George Coulouris, „Distributed Systems – Concepts and Design” Fifth Edition, 2012, ISBN 0-13-214301-1
4. Mullender Sape, „Distributed Systems” Second Edition, 1993, ISBN 978-0201624275
5. Li K., Hudak P., „Memory Coherence in Shared Virtual Memory Systems”, 1989
6. Fonseca H., Verissimo P., „Totally Ordered Multicast in Large-Scale Systems”, Opublikowano: „Proceedings of the 16th International conference on Distributed Computing Systems”, 1996, ISBN 0-8186-7399-0
7. Stanford University, „Chapter 14 – Replication”, <http://www-cs-students.stanford.edu/~dbfaria/quals/summaries/Coulouris-chap14.txt> (dostęp 6.05.2016)
8. Sami Rollins, „Replication”, 10.2008 <http://www.cs.usfca.edu/~srollins/courses/cs682-s08/web/notes/replication.html> (dostęp 10.05.2016)
9. K. Banas „Systemy Równoległe i Rozproszone - Wykład 13”, 03.2016 http://www.metal.agh.edu.pl/~banas/SRR/SRR_W13_Rozglaszanie_Uzgadnianie.pdf (dostęp 12.05.2016)

10. Politechnika Warszawska, „Rozproszone systemy operacyjne”, 06.2007,
http://www.ia.pw.edu.pl/~tkruk/edu/rsob2010/rso_proj2007/rso2007 (dostęp 1.06.2016)
11. Oracle, „Simple Object Access Protocol Overview”, 2001,
https://docs.oracle.com/cd/A97335_01/integrate.102/a90297/overview.htm (dostęp 12.05.2016)
12. Polsko-Japońska Akademia Technik Komputerowych, „RMI – programowanie rozproszone”, 2010,
<http://edu.pjwstk.edu.pl/wyklady/mpr/scb/W11/W11.html> (dostęp 20.05.2016)
13. M.Zakrzewicz, „Wprowadzenie do technologii Web Services: SOAP, WSDL i UDDI”, 05.2006,
<http://www.cs.put.poznan.pl/mzakrzewicz/pubs/ploug06ws.pdf> (dostęp 20.05.2016)
14. J.Brzeziński, C.Sobaniec – Politechnika Poznańska „Usługi sieciowe REST”, 2013 -
https://www.soa.edu.pl/c/document_library/get_file?uuid=46b0faf6-6743-4184-ab16-dbddfd413685&groupId=10122 (dostęp 20.05.2016)
15. T.Olas – Politechnika Częstochowska, 2011 „Oprogramowanie systemów równoległych i rozproszonych”, <http://icis.pcz.pl/~olas/srr/wyklad8.4.pdf> (dostęp 2.06.2016)
16. Robert Werembel – „Rozproszone bazy danych – replikacja danych (Wykład 1)
<http://wazniak.mimuw.edu.pl/images/5/55/ZSBD-2st-1.2-lab1.tresc-1.1.ppt>(dostęp 8.06.2016)
17. Ph. D. Simon Tuffs, „How fast is your network today?”, 2004 <http://soap-stone.sourceforge.net/>
(dostęp 08.06.2016)
18. K.Banaś „Programowanie równoległe – wykład 10” 2015,
http://www.metal.agh.edu.pl/~banas/PR/PR_W10_MPI_wstep.pdf (dostęp 08.06.2016)
19. University of California, „Architectural Styles and the Design of Network-based Software Architectures”, 2000, <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm> (dostęp 03.06.2016)
20. Ph. D. Martin Fowler, „Richardson Maturity Model”, 2010,
<http://martinfowler.com/articles/richardsonMaturityModel.html> (dostęp 02.06.2016)
21. Oracle Java Documentation „All About Sockets”, 2015,
<https://docs.oracle.com/javase/tutorial/networking/sockets/> (dostęp 04.06.2016)